Symantec
by Broadcom Software

Symantec Identity Security Platform - 4.0

Table of Contents

Release Notes 15
- **Release Notes 4.0.2** 15
- **Release Notes 4.0.1** 20
- **Release Notes - 4.0** 23
  - Oracle SQL File Updates 34
- **Identity Security Platform Kit Location and Build Details** 35
  - Identity Security Platform Release 4.0.2 Build Details 35
  - Identity Security Platform Release 4.0.1 Build Details 37
  - Identity Security Platform Release 4.0 build location 39

Overview 43

Administering 47
- **Administrative Model** 47
  - Roles and Responsibilites of Bootstrap Clients 50
- **Resource Authorization** 57
  - Defining a Resource Authorization Model 59
    - Resource Authorization Scoping Constraints 61
  - Building A Resource Model 62
    - Scoping Constraints Overview 62
    - Supported Resource Data Scope (Attribute Permissions) 65
    - Supported Resource Instance Scope (RIS) 68
    - Understanding and Using the Default Constraints 71
    - Adding or Removing a Constraint 74
    - Enabling Resource Admin Scope Enforcement for Resource Authorization 75
    - Determining Applicable RDS and RIS for a Given Principal Using Helper APIs 75
  - Running the Postman API Resource Authorization Demo 76
- **Managing Authorization** 77
  - Scopes and APIs Table 87
  - Authorization API 91
  - Application Roles and Privileges 133
  - Functional Administration 136
    - Role and Resource Access 137
    - Delegation 138

Installing 140
- **Overview of Symantec Identity Security Platform Deployment** 140
  - Kubernetes Support 143
  - Prerequisites 144

Helm Charts ... 145
Sizing the Deployment ... 146
Preparing the Infrastructure Before Deploying the Solution ... 149
Preparing the Kubernetes Cluster Environment ... 149
Preparing the OpenShift Cluster Environment ... 151
Database Considerations ... 154
Out-of-the-Box Demo Database Mode ... 154
Customer Database Mode ... 157
Database Connection Pool ... 161
Oracle JDBC Setup for Symantec Identity Security Platform ... 161
Database Lifecycle Management ... 162
Using Database in SSL Mode ... 162
Information Stored in the Symantec Identity Security Platform Database ... 165
Configuring Connectivity to Amazon Web Service (AWS) MySQL DB Without Using a DB Password ... 165
Deploying the Services ... 170
Pre-Installation Checklist ... 171
Required Ports in Symantec Identity Security Platform ... 172
Checking Entropy Level in the Kubernetes Cluster ... 174
Using Private Image Registry ... 176
Using Image Digest in Symantec Identity Security Platform Charts ... 178
Ingress Overview ... 179
Deploying Ingress Controller ... 180
Deploying NGINX Ingress in Symantec Identity Security Platform Namespace ... 186
Ingress Controller on OpenShift ... 190
Configuring Gateway API for Ingress ... 193
Ingress vs Gateway API: Traffic Flow Comparison ... 194
Gateway API Controller Capabilities ... 197
Gateway API Controller Selection and Prerequisites ... 200
Ingress Configuration Reference ... 204
Already Have a Gateway? Use It! ... 209
Supported Gateways/Ingress Controllers in IDSP ... 213
Constructing the Service FQDN ... 264
Symantec Identity Security Platform Components and URLs ... 264
Master Encryption Key ... 265
DR Deployment Steps ... 270
Cache Considerations ... 273
Deploying Enclave Services ... 274
Deploying Elastic Operator and Kibana ... 275
Setting-Up Data Retention Policy in Elastic Search ... 280
Deploying Prometheus ... 285

Deploying Symantec Identity Security Platform....294
IDSP Infra Helm Chart Configuration....298
IDSP OpenShift Configuration....301
Deploying IDSP Data Chart....304
Post-Installation Checklist....305
Using the Symantec Identity Security Platform postman collection....306
Configuring Enclave Services to Show Symantec Identity Security Platform Content....310
Configuring Administrative Access....318
Managing Tenant Administrator Access Using Admin Console....320
Upgrading Symantec Identity Security Platform....322
Uninstalling the Solution and Enclave Services....329
Operating....331
Security Considerations....331
Managing Observability....342
Logs and Security Audit Data....343
Observing Nginx Ingress Traffic....346
Operational Metrics Data....348
Custom Metrics Exposed by Symantec Identity Security Platform....348
Symantec Identity Security Platform Monitoring Information in Grafana Dashboard....353
Summary....354
Authentication Statistics....355
Integrations....356
Resource Statistics....357
Policy Usage Statistics....359
Services Health....361
Insights....363
Cache Health....364
Database Health....366
Policy Engine Health....368
Key Performance Indicator (KPI)....371
Enclave Services Health....373
Kubernetes Cluster....375
Managing Deployment....376
Meeting Topology Requirements....376
Managing Configuration Updates to the Symantec Identity Security Platform Helm Chart....382
Recovering Initial Admin Credentials....383
Handling Database Password Rotation....384
Scheduler Job....384
Risk Active User Event Data Clean-up Scheduler....387
Risk Inactive User Event Data Clean-up Scheduler....388

Known IA Risk Tags Purging Scheduler ... 389
Inactive Device Purging Scheduler ... 390
Populating the Internal Email Domain Column ... 391
Symantec Identity Security Platform Upgrade Procedure with No Active Database Transactions ... 391
Troubleshooting ... 394
Using Logs to Troubleshoot Operational Errors ... 394
General Issues ... 394
Enclave Service Issues ... 395
GCP Related Issues ... 396
Database Connection Issues ... 396
Tracing ... 397
Understanding Resource Authorization Results using Explain ... 399
Case-specific Exceptions to the Explain Results ... 403
Configuring ... 408
Configuring the Solution ... 408
Configuring the User Lifecycle Features ... 412
User Change Event Hook ... 414
Configuring the ID Store ... 418
Configuring the Internal ID Store ... 426
Configuring the Zero Footprint Identity ... 431
Configuring the User Lockout ... 438
Configuring the User Auto Enrollment and Credential Synchronization ... 443
Configuring Inline Enrollment ... 449
Configuring Trusted Devices ... 453
Encrypting the Device Signature ... 460
Configuring SoftMFA ... 462
Advanced Configurations ... 466
Creating Vanity URLs ... 466
Using Custom CA for Intra Cluster SSL Communications ... 474
Operating with Multiple Data Centers ... 475
Dynamic Substitutions ... 476
Identity Service ... 484
User and Group Schemas ... 484
User Schema Extensions for the Identity Service ... 489
Defining Additional User Attributes for the Identity Service ... 490
Manage Users and Groups in the Identity Service ... 490
Manage Users in the Identity Service ... 491
Manage Groups in the Identity Service ... 495
Constraining User and Group Administration by Origin ID ... 497

Managing the Identity State....499
Password Management in the Identity Service....499
Self Service Account Activation in the Identity Service....509
Self Service Password Recovery in the Identity Service....511
Self Service Account Unlock in the Identity Service....514
Self Service Password Recovery and Account Unlock API....516
Setting-Up the Demo Environment....519
Symantec Identity Security Platform Concepts....519
Understanding the Demo Use Cases....534
Deploying Sample App....570
Deploying Sample App in Symantec Identity Security Platform (IDSP) Namespace....570
Custom Deployment of the Sample App....575
Deploying Demo Mobile Application (Android and iOS)....585
Upgrading Symantec Identity Security Platform Sample Web App....591
Developer Resources....593
API Security Using OAuth2....593
Introduction to Symantec Identity Security Platform Rest APIs and Scopes....594
ACCESS APIs....597
ADMIN API Privileges....597
AUTH API Privileges....601
CAEP API Privileges....602
FACTOR API Privileges....603
IARISK API Privileges....604
IASERVICES API Privileges....605
IDENTITY API Privileges....605
IDSTORE API Privileges....607
LDAP API Privileges....607
ME API Privileges....607
OAUTH2 API Privileges....607
RISK API Privileges....608
SCIM 2.0 API Privileges....608
UI API Privileges....610
Tokens....611
Access Token....611
Identity Token....615
Refresh Token....621
Custom Token Claims....631
Revoke Token....641
Using Tokens to Drive Generic Authorization....644

Client Types....652
Grant Types....653
Client Credential Grant....654
Authorization Code Grant....656
JWT Bearer Grant....658
Device Code Grant....661
Refresh Token Grant....666
Resource Owner....669
Authorization Server Endpoints....670
OAuth2 Token Endpoint....685
JSON Web Encryption (JWE)....692
Supported JWE Use Cases....693
Settings and Application Configurations for JWE....694
Keys and Certs required for JWE....696
Requesting Encrypted JWE in various AuthN and AuthZ flows....696
Secure Communication for Clients Using PKCE....702
Detailed Examples of Authorization Endpoints....703
Authorization Flows Supported....704
Discovery Profile of Provider Metadata....717
Making the First REST API call....720
Integrating Sample Web App with Symantec Identity Security Platform....723
Internals of the Sample Application....735
Integrating Sample Mobile App with Symantec Identity Security Platform....737
Steps to Build and Install Sample Mobile Application....746
Using the Sample Mobile Application....747
Implementing Registration of Multiple FIDO Credentials....748
Integrating Custom Web Applications with Symantec Identity Security Platform....749
Working with ACR....760
Custom Web App with Identity Token Hint....764
Integrating with Custom Service Provider Interface (SPI)....767
Integrating with External Risk Provider....781
Integrating with External SMS Provider....795
Integrating with External Email Provider....800
Integrating with External Push Provider....806
Integrating with External IVR Provider....814
Integrating with Custom Authentication Factor....820
Examples of Using a Custom Authentication Factor....829
Integrating with Custom Password Authenticator....877
Integrating with Custom Claims Provider....885
Discovery of User Groups by Applications....912

Resource Server Application Flow....921
Securing MCP Servers with Symantec Identity Security Platform....951
Onboarding MCP Server as a Resource Server Application....954
Onboarding MCP Clients....958
Symantec Identity Security Platform as CAEP Transmitter/Receiver....962
Auditing in Symantec Identity Security Platform....963
Resource Deletion in Symantec Identity Security Platform....967
Symantec Identity Security Platform SDKs....969
Client Release Notes....970
iOS SDK 3.1.5....972
iOS SDK 3.1.4....973
iOS SDK 3.1.3....973
iOS SDK 3.1.2....974
iOS SDK 3.1.1....974
iOS SDK 3.1....975
iOS SDK 3.0....975
iOS SDK 2.3.6....976
iOS SDK 2.3.5....977
iOS SDK - 2.3.4....977
iOS SDK 2.3.3....978
iOS SDK 2.3.2....978
Android SDK - 3.0.3....979
Android SDK - 3.0.2....980
Android SDK 3.0.1....981
Android SDK 3.0....981
Android SDK 2.3.3....982
Android SDK 2.3.2....982
Android SDK 2.3.1....983
Client-Server Compatibility Matrix....984
Symantec Identity Security Platform iOS SDK Framework....987
Integration with iOS App....989
Integration with iOS App with dependent .xcframeworks (without running pod commands)....992
Symantec Identity Security Platform iOS Core SDK....993
FIDO iOS SDK....1009
iOS SDK Error Codes....1023
Symantec Identity Security Platform Android SDK....1026
Integration with Android App....1027
Symantec Identity Security Platform Android Core SDK....1030
FIDO Android SDK....1040
Symantec Identity Security Platform Web SDK....1050

SDK for Transaction Signing (Web)..... 1050
SDK for Password Encryption (Web)..... 1052
SDK for Device Signature Encryption..... 1056
VIP SDKs..... 1057
SDK for Risk Android..... 1058
SDK for Risk iOS..... 1061
SDK for Risk Web..... 1064
SDK for Mobile OTP Android..... 1065
SDK for Mobile OTP iOS..... 1067
Password Encryption iOS SDK..... 1071
Using the REST APIs..... 1079
Authentication APIs..... 1081
Initiate Authentication..... 1081
Understanding the Authentication Flow..... 1090
Next Actions Table..... 1092
Reauthentication/Step-Up..... 1103
Custom Headers for End-User IP Address Usage..... 1110
Enhanced OTP Authentication Method References (AMR)..... 1111
Authentication Factors..... 1112
Password Authentication..... 1112
Platform and Cross-Platform Authenticators..... 1133
VIP Access Push Authenticator..... 1152
OTP Authenticator..... 1160
Identity Provider (IDP) as an Authentication Factor..... 1180
Custom Authentication Factor..... 1182
Attribute-Based Authentication Factors..... 1183
Performing ID Store Search..... 1193
Transaction Signing using FIDO..... 1194
Transaction Signing using OCRA..... 1197
Self Service APIs..... 1199
Self Service Console APIs..... 1201
Self Service Console Preferences API (for End Users)..... 1201
Self Service Console UserPreferences API (for Administrators)..... 1202
Self Service Console AppTiles API (for End Users)..... 1204
Self Service Console UserAppTiles API (for Administrators)..... 1206
Self Service Console MyNotifications API..... 1208
Self Service Console Notifications API..... 1209
Management APIs..... 1212
REST API Filter..... 1212
Managing LDAP Directory Configuration..... 1215

Managing SCIM Configuration....1225
Managing Applications....1235
Creating a Resource Server Application....1256
Application Metadata properties....1263
Client Secret Rotation....1273
Launcher App....1277
Symantec Identity Security Platform Operating as an Identity Provider....1283
Managing Application Groups....1304
Managing Policies....1312
Authentication Policy....1313
Identity Provider (IDP) Discovery Policy....1383
Risk Policy....1401
Admin Login Policy....1414
Self-Service Login Policy....1416
Activate Account Policy....1418
Unlock Account Policy....1419
Reset Password Policy....1421
Managing Signing Keys....1422
Managing Certificates....1437
Certificate Rotation....1442
Managing Secrets....1443
Symantec Identity Security Platform Risk Engine....1447
Managing Risk....1452
Using Advanced Authentication Risk Service....1488
Custom Risk Providers....1491
Managing Credentials....1495
Monitoring Services....1498
Managing Settings....1499
Managing IDSP User Operations....1516
Managing VIP Configuration Settings....1518
Managing Message Templates....1520
Managing Message Provider Configuration....1536
Managing Identity Providers (IDP)....1547
Symantec Identity Security Platform Operating as a Service Provider....1550
Managing Tenants....1574
Managing Service Accounts....1579
MeServiceAccounts API....1579
MeServiceAccountsLocker API....1580
ServiceAccounts API....1581
Managing Token Revocation....1582

Managing Scoping Constraints....1583
Managing Sessions....1586
Authentication Policies Evaluator API....1588
Downloading and Uploading Fraud Data....1592
Uploading Fido Attestation Metadata....1596
Dynamic Client Registration....1597
Identity REST APIs....1600
Invoking the Identity REST APIs....1600
The Internal Identity Store Configuration API....1601
SCIM Schema and Resource Types API....1604
User API....1606
Group API....1611
Group Operator API....1615
Bulk API....1616
Bulk Import Credentials....1622
User Self Service API....1626
Password Management API....1628
Scoping Constraints API....1634
Delegations APIs....1643
Helper APIs....1652
Me Schema Helper API....1652
User and Group Schema Helper APIs....1653
Resource Schema Helper API....1654
Username Generation Helper API....1655
Referencing Objects Helper API....1657
Revoke All App Roles Helper....1659
Access Helper API....1660
Resource Counter Helper API....1662
SignIn User Interface (UI)....1664
Integrating Identity Provider with Custom SignIn UI....1668
Use a Custom SignIn UI Application....1670
Custom Header and Footer....1673
Editing CSS to Modify Elements in the Sign In UI application....1674
Supporting Legacy Browsers....1681
Support Username and Password On Initial Login Screen....1682
Enable Custom Content Embeds in SignIn UI....1685
Signin UI Consent for Resource Server Applications....1686
Support for Factor Enrollment Status Indicators with Tenant Configuration and CSS Customization....1688
Localizing the Symantec Identity Security Platform UI Consoles....1690

Localization through IDSP APIs **1692**
Additional Information on localizing the Self-Service Console **1698**
Admin Console **1699**
User and Group Management **1700**
Managing Users 1700
Managing Groups 1705
Applications and Application Groups **1708**
Applications 1708
General 1711
OAuth/OpenID 1714
OAuth Resource Server 1720
SAML 1723
Launcher App 1726
Application Groups 1726
Policies **1728**
Authentication and Access Policies 1728
Identity Provider Discovery Policies 1738
Risk Policies 1744
Identity Providers **1749**
Using Identity Federation to Access Admin Console 1757
Certificates **1763**
Secrets **1766**
Accounts **1768**
Diagnostics **1770**
Profiles **1772**
Settings **1772**
Administrators and Roles 1773
Branding 1781
Service Configuration 1782
Identity Stores 1791
Identity Store - LDAP Configuration 1792
Identity Store Configuration - SCIM Protocol 1797
Internal Identity Store Configuration 1800
Integration Setup 1809
Risk Service 1810
Basic Rules 1810
IP Rules 1815
Country Rules 1817
User Menu **1818**
Generation of Personal Access Token 1819

Self-Service Console **1821**
Home **1821**
My Applications **1822**
Personal Information **1823**
Security **1824**
My Credentials 1824
Authentication Methods 1825
My Trusted Devices 1828
My Known Devices 1828
Change Password 1828
Third-Party Applications **1829**
My Access **1830**
User Menu **1830**
Generation of Personal Access Token (PAT) 1830
Self Service Console for Admins **1831**
Cookies Used in Symantec Identity Security Platform **1833**
API Error Codes **1835**
Auth Manager Error Codes **1835**
ID Store Error Codes **1836**
Application Error Codes **1837**
Factor Error Codes **1838**
Provider Error Codes **1842**
Policy Error Codes **1842**
Credential Error Codes **1842**
OAuth/OIDC Error Codes **1843**
Risk Error Codes **1844**
Signing Key and Certificate Error Codes **1846**
API Security Error Codes **1847**
Message Error Codes **1847**
Settings Error Codes **1847**
VIP Integration Error Codes **1848**
assetlinks.json error codes **1849**
Identity Service SCIM Error Codes **1849**
Password Management API Error Codes **1851**
Deleting Symantec Identity Security Platform Objects Error Codes **1851**
Integrating **1853**
Integration of SiteMinder with Symantec Identity Security Platform **1853**
Release Information of SiteMinder Integration with Symantec Identity Security Platform 1857
Integration Limitations and Known Issues 1862

Prerequisites to Implement the Integration....1863
Summary of Features Supported in Each Build....1864
Configure the Integration of SiteMinder with Symantec Identity Security Platform....1867
Configure the Integration in 12.8.0700.2989 Build....1867
Configure the Integration in 12.8.0700.2947 Build....1873
Support for Symantec Identity Security Platform as a Fallback Authentication Scheme in Authentication Chain Template....1879
Manage Authentication Policies in SiteMinder....1880
Support for Password Policies in Multi Factor Authentication Chain....1881
Support for Symantec Identity Security Platform Risk Service....1883
Support for SiteMinder Locale in IDSP....1884
Send Custom Data from Application to IDSP....1885
Identity Propagation in SiteMinder....1886
AMR Integrity and Continuity....1894
Track the Session Activity of a User in the Integration of SiteMinder with IDSP....1896
Configure Re-authentication for Sensitive Resources Protected by the Multi Factor Authentication or IDSP Authentication Schemes....1899
Additional Configuration to Use SecurID HTML Form for Risk-based Authentication During Re-authentication of a Sensitive Resource....1903
Support for jwt-bearer Token Grant....1903
Validate Revocation Status of ID Token....1905
Validate Claims Using JWT Authentication Scheme....1905
Use ID Token Hint to Generate an SMSESSION....1906
Manage Trusted OIDC Provider and ID Token Policy Using REST APIs....1911
AMR Mapping....1914
Mitigate the Binary Attribute Issue in an Encrypted Password Blob in Oracle Unified Directory 12c....1917
Define a Sitename to Retrieve KDCs in Kerberos Authentication Scheme....1917
Support for Front-Channel SLO in SiteMinder OpenID Connect Provider....1919
Support for Change Password in Authentication and Authorization Web Services....1922
Use an Existing Virtual Host for Authentication and Authorization Web Services....1934
Support for SiteMinder Password Policies in IDSP....1935
Troubleshooting....1936
Symantec Identity Security Platform Training Videos on Support Site....1938
End User License Agreement (EULA)....1939
Documentation Legal Notice....1950

Release Notes

Contains link to the 4.0 Symantec Identity Security Platform release and patches.

This topic includes the links to all the major and minor releases of IDSP:

- Release Notes - 4.0.2
- Release Notes - 4.0.1
- Release Notes - 4.0

Release Notes 4.0.2

View information about the Symantec Identity Security Platform 4.0.2 release.

New Features and Improvements

- The Kubernetes community has announced the retirement of the ingress-nginx project. Starting March 2026, the community stops providing bug fixes and security updates for ingress-nginx. In response, the Identity Security Platform (IDSP) is transitioning from the legacy Nginx Ingress controller to the Kubernetes Gateway API standard.
  The Gateway API is the standard alternative replacing the Nginx Ingress controller across all platform clusters. It provides a unified, extensible framework for managing ingress traffic, enabling advanced routing, improved security integrations through declarative custom resources such as BackendTLSPolicy, and better native support across multi-cloud environments.
  Symantec Identity Security Platform supports Envoy Gateway (recommended for cross-cloud portability), GKE L7 Gateway, Azure Application Gateway for Containers, AWS Load Balancer Controller, and Avi AKO for VMware environments.
  For more information on the Gateway API, see Configuring Gateway API for Ingress. (F162847)
- A new automated scheduler has been introduced to manage inactive device cleanup, helping maintain optimal database performance by preventing device tables from growing indefinitely. See Inactive Device Purging Scheduler. (F162848)
- Added the ability to customize scheduler configurations during installation or upgrade operations. Users can now configure custom execution times and batch sizes for various system schedulers using command-line parameters. For detailed information on scheduler configuration parameters, refer to the "**Customizing Scheduler Settings" section within each Scheduler Job** topic under the Scheduler Jobs documentation. (DE661080)
- The Device Name ("deviceDetails" in Known Devices and "name" attribute in the Trusted Devices API) framework has been enhanced to include additional information in the Known and Trusted Devices API response.

See Trusted Devices API response, and Known Devices API response examples. (DE649741) (DE661898)

- A new application metadata property, **Delay IDP Rule**, has been introduced to support device-specific identity provider configurations. For more information, see App Metadata. (F164347)

App Metadata	Description
• urn:iam:app:delayProtocolLevelIdpDiscoveryUntilSignin (API terminology) • Delayed IDP Discovery (UI terminology)	Identity Provider discovery can now be deferred until the ".../auth/v1/authenticate" API request, allowing discovery to run with full user and client context instead of during the initial protocol redirect. Client context from authenticate requests is now processed earlier in the authentication flow, ensuring availability during IDP discovery and subsequent steps.

- This release enhances the customizable **New Device Sign-in** email template to include machine details. The out-of-the-box template includes the placeholders, $$DEVICE_DETAILS$$, $$OS_DETAILS$$, and $$BROWSER_DETAILS$$.

  You can customize the email template contents using the Message Template API. See Managing Message Templates. (F162563)
- Added the ability for end users to initiate a password change using the Forgot Password link on the login screen, or in a self-initiated Reset Password flow in your client app for SiteMinder implementations. See Changing Password Using Forgot Password or Self-initialized Reset Password Policy-based Flows. (F141954)
- Previously, the Reset Password Policy supported only the sign-in client app. With this release, you can now change the application that is assigned to this policy. See Reset Password Policy for more details about this policy. (F141954)
- This release introduces a new application metadata attribute (`urn:iam:app:samlStrictCompliance`). When this metadata attribute is set to `true`, the following attributes are not sent as part of the SAML assertion issued by Symantec Identity Security Platform. The default value is `false`. (F162393)
  - app_id
  - session_state
  - user_guid
  - idp_guid
  - user_loginid
  - user_riskscore
  - sid
- Identity and Access Tokens include granular Authentication Methods References (AMRs), such as `["SMS", "OTP", "PWD"]` or `["EMAIL", "OTP", "PWD"]`, by default for all valid Refresh Tokens, regardless of the `useGranularAMRForOTP` tenant setting. Previously, granular AMRs were only included when the `useGranularAMRForOTP` setting was enabled, and even then, only Refresh Tokens issued after enabling the setting produced Identity Tokens with granular AMRs. Pre-existing Refresh Tokens continued to produce non-granular values (for example, `["OTP", "PWD"]`), requiring users to re-authenticate. With this enhancement, granular AMRs are applied universally to all valid Refresh Tokens — both new and pre-existing — without requiring the tenant setting to be enabled or forcing user re-authentication.

  For more information on Granular OTP, see Enhanced OTP Authentication Method References (AMR). (DE651244)
- Symantec Identity Security Platform can now be placed into a Read-Only state during database maintenance, enabling administrators to perform maintenance operations without requiring a complete service shutdown. For more information, see Upgrade Procedure with No Active Database Transactions (DE657507).
- The `.../factor/v1/FIDORegChallengeVerifier` API has been enhanced to provide additional metadata about authenticator. The **`"authenticatorAaguid"`** attribute value is now returned only when the

enableFIDOMetadataAttestation tenant setting is enabled during Identity Security Platform (IDSP) deployment. See ".../factor/v1/FIDORegChallengeVerifier" API response. (F165058)

- A new application metadata setting, "**skipAuthenticationForTrustedDevice**", has been added. When enabled, this feature allows users on trusted devices to safely and seamlessly bypass the complete authentication flow. This enhancement improves the user experience by treating the recognized device as a valid authentication factor, removing the need for repeated interactive logins on known, secure devices. To accurately reflect this authentication method, the system includes the `POP` (Proof of Possession) value in the AMR claim when it produces the Identity Token Hint (ITH).This application metadata is covered in the Application Metadata topic. (F166010)
- The IDSP Admin service now includes a "**strikeCount**" field (integer) in the "get user" GET ".../admin/v1/Users/{{REPLACE-BY-USERID-OR-USERLOGINID}}" endpoint response. This field displays the user's current strike count from authentication failures, enabling administrators to monitor user strike status directly without additional API calls. (DE661957)

Fixed Issues

- The ivr_phone_number attribute disappears from the admin console when configured as multi-value for auto enrollment, preventing voice OTP delivery to multiple phone numbers.
  As part of this fix, added the following:
  - New app metadata, "urn:iam:app:**allowIvrCredsOfTag",** see Application Metadata properties
  - Support for ivr_phone_number in the "mvAttributeMapping" LDAP attribute, see Managing LDAP Directory Configuration

  (DE661784)
- The credential exclusion (urn:iam:app:**excludeCredsNotUsedForXDays**) app metadata setting that filters out credentials not used within a specified number of days does not properly exclude credentials that have never been used (those with blank/null lastUsed values). (DE659218)
- In an integration flow, the Credential Usage metrics in the Grafana dashboard show SMS and Email OTP as "Primary" authentication methods, when these factors are configured as second factors. (DE658403)
- A duplicate entry was created for an existing trusted device for a user. This issue occurred when a single device was shared between two users. If one user removed the shared device and then re-accessed it, the **IA Risk** device tag rotation triggered the creation of a duplicate entry for an existing trusted device. (DE661315)
- An issue was found where users were unable to log in immediately after creating a custom tenant. A race condition in the db transaction during tenant creation and incorrect caching entry was leading to this issue. (DE663688)
- An issue was observed in JIT provisioning where case-variant Multi-Value Attributes (emails, phones, addresses) were not correctly deduplicated, leading to redundant record creation during federated authentication. (DE664062)
- Introduced a new app metadata configuration:

App Metadata	Description
• urn:iam:app:includeAdditionalContextInLaunchApp (API terminology) • Enhanced Launch Context (UI terminology)	Applications can receive additional context parameters (logout URL, session state) when users are redirected to the launch URL after successful authentication.

  For more information on these Application Metadata, see Application Metadata Properties.(DE663788)
- AMR Not Handled Correctly for Attribute-Based Authentication Factor in Policy Rules. When a Policy Rule includes an Attribute-Based Authentication Factor, the Authentication Methods Reference (AMR) is not functioning as expected in two ways:

  - The AMR value associated with the Attribute-Based Authentication Factor is not included in the generated ID Token after authentication.
  - When the ".../authentication" endpoint is called with an existingIDToken payload that already contains the AMR for the Attribute-Based Authentication Factor, the system does not recognize it — causing the factor to be re-prompted instead of being skipped. (DE666159)
- In the mandatory authentication flow, if the obligation sequence is interrupted in the setting,`inlineEnrollmentFactorsRequired` value, all subsequent inline enrollment factors with a priority equal to or higher than the break point are skipped. For example, when the policy enforces PASSWORD (priority 1) and FIDO (priority 2, unregistered), and `inlineEnrollmentFactorsRequired` is set to "EMAILOTP:1, SMSOTP:1, MOBILEOTP:3, PUSH:3" — only factors with priority lower than the break point (EMAILOTP, SMSOTP) are presented. MOBILEOTP and PUSH (priority 3) are silently skipped, resulting in an incomplete authentication challenge. (DE660411)
- The Symantec Identity Security Platform Swagger documentation took an excessively long time to respond (often over 30 seconds). This delay frequently resulted in 503 errors for clients due to proxy timeouts. (DE662755)
- The Swagger API specifications for the Authentication Factor APIs (`/admin/v1/AuthenticationFactor`) and the Authenticator API (`/factor/v1/Authenticator`) were missing from the technical documentation. (DE666398)
- Improved the performance of app tile rendering in the Self Service Console. (DE662603), (DE662604)
- When the user being authorized is assigned a high number of applications, the Self Service Console dashboard experienced issues loading. (DE662598)
- The text "enrollment required" would sometimes erroneously appear on the sign-in screen for users. (DE664413)
- The provided username is ignored when application metadata to delay idpProtocol discovery is set to `true` . (DE664775)
- Password validation rules now enforce that passwords cannot contain the user displayName (Full Name Value) composed by the displayName Rule on the internalIdStore. If the displayName contains delimiters (such as commas, periods, underscores, or dashes), Symantec Identity Security Platform splits the displayName and confirms that each section is not included in the password. This validation ignores any displayName (and its sections) smaller than three characters. (DE664231)
- On the Symantec Identity Security Platform external tenant, the internal *cred sync initiate* audit event obscures the password. (DE666718)
- Symantec Identity Security Platform does not update the deleted email/phone/device credentials list and does not send notifications for deleted email/phone/device credentials when syncing credentials using the scim2/v1/Me APIs. (DE665556)
- An *internal.cred.sync.initiate* audit event is created even if the Service Account creation failed, due to duplicate account 409 errors. (DE665532)
- The `T_USER` table column that stores Apptile Group and Notification data (including banner and admin notifications) has been altered from its previous data type to `LONGTEXT` . This change accommodates larger payloads such as HTML-formatted banner content and Apptile Group JSON data that previously exceeded the column size limit. (DE662925)
- Updated Nginx-Ingress version from 4.14.0 to.4.14.3 (DE662438)
- Third-party libraries identified as vulnerable in the previous release of Symantec Identity Security Platform have been upgraded to mitigate security risks in external components. (DE662732), (DE662936), (DE663510), (DE664246), (DE666336).

Known Issues

- When an Identity Provider (IDP) is force-deleted, the background scheduler fails to remove its stale references from the authorization policies. As a result, the deleted IDP incorrectly persists in the UI under the Administrators and Roles screens. (DE669313)
- Trusted Device Identification (TDI) flow does not enforce password reset when `passwordMustChange` is enabled
  - **Password reset screen is not displayed during Trusted Device Identification (TDI) flow when `passwordMustChange` is enabled**

When a user with the `passwordMustChange` attribute set to `true` authenticates through the TDI flow, the password reset screen is not presented after completing PASSWORD authentication (primary obligation). Instead, the user is logged in to the AdminConsole without being prompted to change their password. This bypasses the intended password reset enforcement.

– **Authentication is skipped entirely in TDI flow when both `passwordMustChange` and `skipAuthenticationForTrustedDevice` are configured**
When the `passwordMustChange` attribute is enabled for a user and the application metadata `urn:iam:app:skipAuthenticationForTrustedDevice` is also set, the PASSWORD challenge is skipped entirely during the TDI flow. This allows the user to log in without any authentication challenge, bypassing both the trusted device authentication and the password reset requirement.

Upgrade Considerations

IMPORTANT

- **IDSP Supports Incremental Upgrades**:
  IDSP facilitates incremental upgrade paths. If you plan to upgrade to the latest release, IDSP v4.0.2, and you are currently using an earlier version, it is advisable to first upgrade to the latest patch of the subsequent minor release before moving on to IDSP 4.0.2.
  IDSP uses the following format for naming release versions: MAJOR.MINOR.PATCH. For example, in the release version 4.0.x, "4" is the MAJOR version, "0" is the MINOR version, and "x" is the PATCH version.
  Upgrade Use Cases:
  - ***Upgrade Scenario 1***: Upgrade from IDSP v3.3 to IDSP v4.0
    Recommended Upgrade Path: You should follow this upgrade path- **3.3.x > 3.4.x > 4.0.2**
    Example: 3.3.6 > 3.4.8 > 4.0.2
  - ***Upgrade Scenario 2***: Upgrade from IDSP v3.4.x to IDSP v4.0.x
    Recommended Upgrade Path: You can directly proceed with the upgrade to v4.0.2 without any intermediate steps.
    Example: 3.4.8 > 4.0.2
- The dataseed job may take longer than usual during the upgrade due to the `ALTER TABLE` statement on the `T_USER` table. The duration depends on the number of records in the table. Plan the maintenance window accordingly.
- **Oracle SQL file Updates**: This release includes Oracle database schema changes that may require manual SQL file updates for environments with separate database administration. For detailed upgrade procedures and SQL file application order, see the Oracle SQL File Updates.

See Upgrading IDSP.

Rollback Considerations

Hazelcast 5.7.0 Upgrade — Impact on Zero Downtime

The Hazelcast upgrade from 5.3.5 to 5.7.0 introduces the following considerations for Zero Downtime (ZDT) deployments.

Upgrade

The Helm upgrade command remains unchanged and performs a rolling update, recreating Hazelcast pods sequentially. However, deployments with a single Hazelcast pod will experience a brief service disruption, as the existing pod is terminated before the replacement pod becomes ready.

Rollback

Rolling back from Hazelcast 5.7.0 to 5.3.5 requires manual intervention. During rollback, newly created pods running 5.3.5 cannot establish a connection with existing pods still running 5.7.0, causing the new pods to enter a crash loop. To resolve this, all existing Hazelcast pods must be forcefully deleted so that every pod restarts cleanly on the rolled-back version.

Rollback Command

Run the following command to perform the rollback. The command executes three steps sequentially, each step proceeds only if the previous step succeeds.

```
helm rollback ${RELEASENAME} <revision> -n ${NAMESPACE} && \
kubectl delete pods -l app.kubernetes.io/name=hazelcast-entp -n ${NAMESPACE} && \
kubectl rollout status statefulset/ssp-hazelcast-entp -n ${NAMESPACE} --timeout=5m
```

Step	Description
Step 1: Helm Rollback `helm rollback ${RELEASENAME} <revision> -n ${NAMESPACE}`	Rolls back the Hazelcast Helm release to the specified revision number, reverting all chart configurations, image versions, and Helm values to their state at that revision.
Step 2: Delete Pods `kubectl delete pods -l app.kubernetes.io/name=hazelcast-entp -n ${NAMESPACE}`	Force-deletes all running Hazelcast pods matching the label `app.kubernetes.io/name=hazelcast-entp`. Because these pods are managed by a StatefulSet, Kubernetes automatically recreates them using the rolled-back configuration, ensuring a clean restart on the correct version.
Step 3: Verify Rollout `kubectl rollout status statefulset/ssp-hazelcast-entp -n ${NAMESPACE} --timeout=5m`	Monitors the StatefulSet rollout and waits up to five minutes for all pods to reach a ready state. If the pods do not become healthy within this period, the command exits with an error.

Symantec Identity Security Platform Kit Location and Build Details

See Release 4.0.2 Build Details.

IDSP Knowledge Articles

All IDSP knowledge articles can be found here, Symantec Identity Security Platform Knowledge Articles.

Release Date

The IDSP v4.0.2 release was distributed on May 19, 2026.

Release Notes 4.0.1

View information about the Symantec Identity Security Platform 4.0.1 release.

This topic includes the following sections:

- New Features and Improvement
- Fixed Issues
- Upgrade Considerations
- Symantec Identity Security Platform Kit Location and Build Details
- Symantec Identity Security Platform Knowledge Articles
- Release Date

New Features and Improvements

- Enhanced password expiration warning dialog in the Sign-in UI with a new "Change Now" button that provides users with immediate access to password reset functionality. (F163525)
- The enhanced factor selection screen in the Sign-in UI clearly distinguishes between authentication factors that are already enrolled and those that require initial setup. Selecting an unenrolled factor now seamlessly initiates an inline enrollment workflow. For more information, see Support for Factor Enrollment Status Indicator with Tenant Configuration and CSS Customization. (F163537)
- Starting from Symantec Identity Security Platform v4.0.1, the `requireAccessControl` application setting is now mandatory for all new application creations replacing the previous optional configuration. This setting supports three values, true, false, and tenant. The default value is `tenant` . Applications created previously with null value are handled through backward compatibility mechanisms. (DE660807),(DE660782)
- The Service Account option in the Self-Service Console is now dynamic. It appears only for users with an existing service account and remains hidden for those without one. (F163549)
- The new Diagnostics feature introduces a comprehensive logging and troubleshooting capability within the Admin Console, enabling administrators to capture, view, and analyze detailed logs for specific user sessions in real-time. See Diagnostics. (DE660155)
- **For upgrades to 4.0.1:** If you previously configured the **End User Password Reset Vanity URL** parameter of the internal identity store (`endUserPasswordResetVanityUrls`), you must complete the following procedure to correct it after upgrading to 4.0.1:
  a. Using the The Internal Identity Store Configuration API, clear the parameter value by setting the `endUserPasswordResetVanityUrls` attribute to double quotes ("").
  b. Reconfigure your vanity URLs using the Admin Console UI or the API.

  If you do not make this change, you will see validation errors in the Admin Console and be unable to save the identity store.

Fixed Issues

- Vanity URL configuration in IDStore breaks Self-Service Console access due to missing "/common/" path. (DE660788)
- iOS browsers experienced navigation failures in the Self-Service Console when administrative banners containing HTML anchor tags (<a href="">) were enabled. (DE660792)
- Users experienced display issues with the Self-Service Console's Personal Information page. The email type, phone type, and address type dropdown lists were not rendering correctly, preventing users from selecting appropriate contact information types when updating their profiles. (DE660477)
- Multiple MCP (Model Context Protocol) clients experienced integration failures when attempting to connect to the IDSP MCP Server. (DE660569)
- The authentication flow incorrectly triggered external identity token revocation validation checks. (DE660045)
- App Launcher failed to save flow state information to the database when configured for DB Persistence mode. (DE660791)
- Random failures in SCIM APIs may occur due to thread safety violations. (DE660786)
- SCIM requests with invalid JSON bodies may throw a 500 internal server error, rather than a 400 bad request error. (DE660793)
- Changed the control element for the **Service Configuration** > **End User Notification** option in the Admin Console to a selection box. (DE660195)
- The configured request timeout variable (GEOLOCATION_REQUEST_TIMEOUT) was not honored by any of the Symantec Identity Security Platform services, potentially increasing the overall time of the transactions. (DE661053)
- In Active/Active (A/A) deployments with browser geo-steering, access tokens issued in one region may be redeemed in a different region, causing session validation failures. (DE660790)
- The geolocation service exhibits high latency, resulting in long response time of other services that are dependent on the geolocation service. (DE660784)
- The geolocation service returns HTTP 404 status for invalid or unknown IP addresses instead of the expected HTTP 200 with appropriate response body. (DE660785)
- Access policy creation and update operations fail to process principal conditions containing "appRoles" with wildcard format "/*", causing these conditions to be omitted from the final policy configuration. (DE661232)
- In Bring Your Own Identity (BYOI) flow, "RememberMe" functionality was not working. (DE660787)
- Following the upgrade to IDSP v4.0, users experienced authentication failures when calling SPI (Service Provider Interface) tokens from SCIM (System for Cross-domain Identity Management) operations. (DE660781)
- The BYOI flow was not enforcing maximum session limits per user, allowing users to exceed configured session thresholds. (DE661313)
- User disambiguation process failed when groups mapping was configured in Claims but the Identity Provider (IDP) returned no group information. (DE661166)
- Third-party libraries identified as vulnerable in the previous release of Symantec Identity Security Platform have been upgraded to mitigate security risks in external components. (DE660653), (DE661226), (DE661032)

Upgrade Considerations

IMPORTANT

- **IDSP Supports Incremental Upgrades**:
  IDSP facilitates incremental upgrade paths. If you plan to upgrade to the latest release, IDSP v4.0.1, and you are currently using an earlier version, it is advisable to first upgrade to the latest patch of the subsequent minor release before moving on to IDSP 4.0.1.
  IDSP uses the following format for naming release versions: MAJOR.MINOR.PATCH. For example, in the release version 4.0.x, "4" is the MAJOR version, "0" is the MINOR version, and "x" is the PATCH version.
  Upgrade Use Cases:
  - ***Upgrade Scenario 1***: Upgrade from IDSP v3.3 to IDSP v4.0
    Recommended Upgrade Path: You should follow this upgrade path- **3.3.x > 3.4.x > 4.0.1**

Example: 3.3.6 > 3.4.8 > 4.0.1

- ***Upgrade Scenario 2***: Upgrade from IDSP v3.4.x to IDSP v4.0.x
  Recommended Upgrade Path: You can directly proceed with the upgrade to v4.0.1 without any intermediate steps.
  Example: 3.4.8 > 4.0.1

- **Oracle SQL file Updates**: This release does not include any Oracle database schema changes. However, if you plan to upgrade from Authentication Hub v3.4.8 to Symantec Identity Security Platform v4.0.1, you need to apply all the SQL files mentioned in v4.0. See Oracle SQL File Updates.

See Upgrading IDSP.

Symantec Identity Security Platform Kit Location and Build Details

See Release 4.0.1 Build Details.

IDSP Knowledge Articles

All IDSP knowledge articles can be found here, Symantec Identity Security Platform Knowledge Articles.

Release Date

The IDSP v4.0.1 release was distributed on February 2, 2026.

Release Notes - 4.0

View information about the Symantec Identity Security Platform 4.0 release.

This topic includes the following sections:

- Product Rebranding: Introducing Symantec Identity Security Platform
- New Features and Improvement
- Fixed Issues
- Known Issues
- Upgrade Considerations
- Rollback Considerations
- Symantec Identity Security Platform Kit Location and Build Details
- Symantec Identity Security Platform Knowledge Articles
- Release Date

Product Rebranding: Introducing Symantec Identity Security Platform

VIP Authentication Hub is now ***Symantec Identity Security Platform (IDSP)***.

This rebrand reflects the solution's significant evolution over the past five years from a focused authentication product into a comprehensive, modern identity and access management platform. What began as VIP Authentication Hub —delivering strong authentication and FIDO2/passkey capabilities—has expanded to include local identity services, advanced federation (SAML, OIDC) flows, granular administration, access policies, OAuth resource scope configuration, customizable self-service console, personal access tokens, and application dashboards. The new name, Symantec Identity Security Platform, accurately represents these broader capabilities and positions the solution within the Symantec product family.

What this means for you: All functionality remains intact. Existing deployments, integrations, and APIs continue to work without modification. Documentation for version 4.0 and later will reference "Symantec Identity Security Platform" or "IDSP," while previous version documentation will retain the "VIP Authentication Hub" name.

VIP Authentication Hub v3.4 and prior releases continue to reside at their original location: VIP Authentication Hub Homepage.

New Features and Improvements

- IDSP 4.0 introduces a new Identity Credential Verifier role that grants permission to update the verified status of an email or phone number in the User Profile. This role can be granted to a User/Client.

  IMPORTANT
  You must add the Identity Credential Verifier role to allow administrators to set a user's email and phone verified state using SCIM. See Administrators and Roles.
- Resource Server: This new feature enables seamless creation and configuration of Resource Server applications, offering robust control over protected resources through the IDSP Admin Console and can also be executed using REST API. Users can now define essential attributes, and a set of resource server-specific scopes. It also facilitates the configuration of consent policies with dedicated URLs for privacy policies. This capability is particularly useful for securing protected resources by enabling granular access control and ensuring only authorized client applications can access sensitive data and functionalities.
  - Complete Application Lifecycle - Streamlined provisioning of Resource Server applications with unique scope identifiers, primary audience configuration, and custom resource server-specific scopes with detailed descriptions and consent requirements
  - Advanced Authorization Management - This release introduces Application Roles, that enables fine-grained access control for Resource Server applications. Application Roles provide administrators with the ability to delegate resource-specific privileges by updating the authorization policies that act as application roles with precise privilege assignment, and built-in role delegation capabilities.
  - Enhanced Security & Compliance - Comprehensive consent management framework with configurable requirements per scope, tenant-level controls for consent persistence.
  - Full OIDC Flow Support - Complete implementation of authorization code flow with resource server integration, and seamless token exchange processes.
  - The Signin UI Consent feature introduces a consent management system that enhances user privacy and application security by requiring explicit user approval for resource server application access. Administrators can configure these consent experiences through the Admin Console's OAuth Resource Server tab, where they define scopes with custom consent labels and descriptions that appear to users at runtime. When applications request specific privileges (scopes), users are presented with a dynamic consent screen that displays the application name, detailed descriptions of requested permissions, and optional usage policy links. See Signin UI Consent for Resource Server Applications, Creating Resource Server using Admin Console.

  See Creating Resource Server using Admin Console, Creating a Resource Server Application using REST API, for complete resource server flow using REST API, see Resource Server Application Flow.
- Self-Service Console (UI) include the following updates:

- Application Dashboard: Presents a collection of application tiles, organized into sections of your choice. It provides the users with a personalized dashboard of applications they can access. This help users quickly find, organize, and launch their various work-related applications. See My Applications.
- Notifications System: Provides two types of notification, one provides system-wide alerts and other provides personal notifications for application assignments. Provides real-time updates and alerts. See My Applications.
- My Known Devices: Provides a security feature allowing users to review and revoke access from devices they previously marked as known, enhancing control over their account security. See My Known Devices.
- My Refresh Tokens: Allows to view and manage refresh tokens associated with your account. It provide control over your active sessions by enabling them to revoke individual or multiple refresh tokens. See My Refresh Tokens.
- My Sessions: Provides a comprehensive overview of all devices and applications currently signed into your account. Real-time session monitoring displaying all active user sessions with detailed information including IP addresses and browsers. See My Sessions.
- Change Password: This feature behavior has been changed to support two types of password change flows, authentication policy driven flow or Verified credential flow based on whichever the admin configures for the user. See Change Password.
- Third-Party Apps: Complete management of external application connections with detailed permission visibility and access history. Enables easy revocation of third-party access. See Third-Party Applications.
- My Access: Displays a list of applications that the user has access to. See My Access.
- Generation of Personal Access Token (PAT): Allows PAT generation and lifecycle management with privilege-based token creation. Provides secure API authentication, granular permission control, and administrative token capabilities. See Generation of Personal Access Token.
- Verified Credentials in Personal Information: Multi-factor credential verification system supporting email and phone verification with real-time status tracking. Offers self-service verification processes. See Personal Information.
- For Self-Service Console Admins - A number of tenant settings have been introduced to control the Self-Service Console Homepage. See Self-Service Console Tenant Settings.
- Symantec Identity Security Platform Localization - The Self-Service Console now supports multiple languages, enabling users worldwide to manage their accounts in their native language. This feature includes support for out-of-the-box languages, with the ability to add custom locales. Administrators can configure different languages for different tenants and users will automatically see the console in their preferred language based on browser settings or tenant configuration. See Localizing the Symantec Identity Security Platform UI Consoles and Additional Information on Localizing the Self-Service Console.

See Self-Service Console for more information on these UI enhancements.

- Self-Service Console (API) include the following updates:
  By default, all the applications that are authorized or granted to an end user appear (as application tiles) on the Self Service Console dashboard for an end user. The following Self Service Console REST APIs are added to allow administrators and end users to manage the groups, apps, and notification settings of the dashboard:
  - AppTiles: As an end user, use this API to manage how apps are grouped on your Self Service Console dashboard. You can manage (add, remove, or rename) your groups or move apps into and out of your groups.
  - UserAppTiles: As an administrator, use this API to manage how apps are grouped for all of your end users on the Self Service Console dashboard. You can manage (add, remove, or rename) groups or move apps into and out of the groups.
  - Preferences: As an end user, use this API to set how apps are displayed on your dashboard, such as the app order, size of the app tiles, and more.
  - UserPreferences: As an administrator, use this API to set how apps are displayed for all of your end users, such as the app order, size of the app tiles, and more.
  - MyNotifications: Use this API to view all Self Service Console notifications sent to you as an end user.
  - Notifications: By default, the Self Service Console provides notifications when apps are removed or added to a user, app-specific notifications, and general notifications. The Self Service Console generates the app addition and removal emails. However, as an administrator, use the Notifications API to manage (view, create, or delete) app-

specific and general notifications. When creating notifications, you can choose the audience, the dates that they are sent and expire, their priority, and more.

See Self-Service Console APIs for more information on these APIs.

- The following features are related to Risk Service:
  - Risk Policy Management is a comprehensive security feature that empowers administrators with granular control over authentication risk assessment and access management. This advanced capability allows organizations to create, configure, and deploy sophisticated risk evaluation policies that operate seamlessly in the background without impacting user experience. The feature supports complex rule structures including action-based conditions (such as "authenticate" operations), application-specific targeting through precise ID matching, and channel-specific configurations for web, mobile, and other platforms. Risk policies integrate with different risk rules. Each policy supports configurable risk scores across multiple risk rules for comprehensive assessment. You can now configure Risk policies from the Admin Console. See Risk Policy in Admin Console.The system includes complete REST API management with CREATE, READ, UPDATE, and DELETE operations protected by the urn:iam:t.configs API scope. See Risk Policies using REST API.
  - Enhanced Admin Console with Risk Service configuration, providing administrators with granular control over risk assessment and rule management. This feature allows you to fine-tune how risk is calculated and applied to authentication events, improving your security posture. See Risk Service.
  - Two new configuration settings, "maxNumberOfDevicesPerUser", and "deviceLimitAction" have been added to the existing Device Recognition functionality to provide administrators with granular control over device management per user. These settings enable limiting the number of registered devices per user and defining the action to take when the device limit is exceeded. See Device Recognition.
  - A new configuration setting, "enableStaticScore" has been added to the Behavioral Engine risk rule that provides administrators with enhanced control over risk scoring for behavioral anomalies. See Behavioral Engine
  - Enterprise IP List Management capabilities - New APIs facilitate the upload of Allowed or Blocked IP lists. Administrators can monitor the real-time status of these uploads, download existing IP lists, and delete them entirely, ensuring flexible and efficient management of network access policies. See Allowed or Blocked IP List for REST API Usage (REST API), and Allowed and Risky IP Address section in the Admin Console documentation.
  - A new automated scheduler system has been introduced to manage the cleanup of old risk event records from the IDSP database. This feature helps maintain optimal database performance by removing outdated records and reducing storage overhead.
    - Risk Active User Event Data Clean-up: Runs every 30 minutes to remove risk events that are older than 720 days for active users. Enable using "isRiskCleanupEnabled" global setting. Disabled by default.
    - Risk Inactive User Event Data Clean-up: Runs on Sundays to remove risk events that are older than 720 days for inactive users. Enable using "isRiskInactiveUsersEventsCleanupEnabled" global setting. Disabled by default.

    See Scheduler Job.
- Service accounts are specialized accounts for service-level users such as trusted clients. Service accounts are not subject to the same limits that an interactive user would be for authentication session lifetime and credential lifetime. This release improves the visibility and management of service accounts through the Admin Console and through APIs.
  - In the Admin Console:
    - If a user is marked as a service account, the **Service Account** control is set to **Yes** on the **Users** page or the **Account** > **Users** page.
    - The **Service Account** tab is added to the **Accounts** page for a user in the Admin Console to provide information about service accounts. Managers can view, lock/unlock, and disable/enable service accounts for their users in the **Service Account** tab.

    See Accounts.
  - Through the Service Account APIs. Three new service account management APIs are added:
    - MeServiceAccounts API allows end users to view and validate their service accounts
    - MeServiceAccountsLocker API allows end users to lock or unlock their service accounts
    - ServiceAccounts API allows administrators to view and validate the service accounts for their users

See Managing Service Accounts.

You can manage a service account if you are the assigned manager (as defined in the `user.manager` property) or have the `urn:iam:t.serviceaccountsadmin` privilege.

- Introduced an enhancement to the signing key management that significantly improves the robustness and continuity of service during cryptographic key rollovers. The system now automatically retains previous signing keys for a configurable grace period (default: 3 days) after new keys are activated, allowing JWT token validation to gracefully fall back to previous keys when primary verification fails.
  This enhancement includes two new configuration settings:
  – "successiveSigningKeyUpdateIntervalMins" (global setting, default: 30 minutes) to prevent rapid successive key update calls
  – "gracePeriodToRetainPreviousSigningKeyDays" (tenant setting, default: 3 days) to control the retention period for previous keys.
  The feature supports enhanced API operations through POST and PATCH endpoints for signing key management and dynamically manages JWK entries to support current, previous, and next key pairs simultaneously. This improvement eliminates service interruptions during key rollovers, provides dependent applications ample time to transition to new keys, See Managing Signing Key using REST API.
- Previously, self-service and recovery actions were initiated using Verified Credentials (a token is sent to the end user email or phone that the end user redeems to complete the self-service or recovery action). With this release, three pre-configured policies are added to control user authentication when they perform the following self-service recovery and activation actions (policy-based approach):
  – Activating an account (ActivateAccountPolicy)
  – Unlocking an account (UnlockAccountPolicy)
  – Resetting a password (ResetPasswordPolicy)
  These policies apply only to your sign-in client app. You cannot change the application that is assigned to this policy. Also, you cannot add any custom actions to this policy. See Authentication and Access Policies for more details about these new policies.
  Use one of the following methods to set whether to use the Verified Credential approach or the policy-based approach:
  – In the Admin Console, select **Settings** > **Identity Store Configuration** > **Internal Identity Store** and select **Verified Credential** or **Policy** for **Password Recovery Flow Type**, **Account Unlock Flow Type**, and **Account Activation Flow Type**. See Internal Identity Store Configuration
  – In the InternalIdStoreConfig API, set `identity` (Verified Credential) or `policy` (policy-based) for `passwordRecoveryFlowType` , `accountUnlockFlowType` , or `accountUnlockFlowType` . See Internal Identity Store Configuration.
- A new configuration setting, "persistAdditionalContextInRiskEvent" has been introduced to control the persistence of additional context information for risk events in the VIP Authentication Hub database. This setting controls whether additional context information related to risk events should be stored in the VIP Authentication Hub database. Enabling this setting will result in consumption of additional DB storage. DB should be sized appropriately before enabling this setting. By default, this setting is false. See Managing Settings.
- The existing "trackLastLogin" tenant setting has been enhanced to also allow tracking of user credential usage by recording the timestamp of every authentication attempt. When this setting is enabled, the system updates the "lastUsedDatetime" attribute for all those user credentials that are used during the authentication flow, capturing only successful logins attempts to provide visibility into credential activity. See Managing Settings.
- A new enhancement has been implemented to support IP address persistence throughout the complete trusted client authentication flow. This feature enables access to the user's IP address during token operations that occur after the authentication flow completion, providing enhanced security context and audit capabilities.
- A new security enhancement has been implemented to restrict Multi-Factor Authentication (MFA) factor enrollment using a new authentication policy attribute called, "inlineEnrollmentAllowed". This optional property enables you to explicitly permit or prevent enrollment based on the security context (for example allowing it on-network but disabling it off-network to prevent unauthorized additions). See Authentication Policy - inlineEnrollmentAllowed attribute.
- The following features are related to Passkey:

  - Enhanced Security with FIDO Metadata Attestation and Device Identification: This release introduces FIDO Metadata Attestation capabilities that not only verify the authenticity and security of FIDO Alliance certified security keys and passkey authenticators during registration, but also provide detailed device identification and tagging functionality. The system now captures and displays specific authenticator information including device origin, model details (such as "YubiKey 5C NFC"), platform type (Mac, Windows), browser information, and other pertinent device characteristics through the Authenticator Attestation Globally Unique Identifier (AAGUID).

    A new "enableFIDOMetadataAttestation" tenant setting enables FIDO attestation, while the ".../system/admin/v1/FIDOMetadataUploader" API facilitates secure upload of FIDO metadata. This enhancement boosts trust, strengthens security, and helps ensure only genuine devices are enrolled. See FIDO Attestation and Uploading FIDO Attestation Metadata.
  - Starting with IDSP release 4.0, the system implements automatic passkey overwrite logic to prevent multiple passkeys (which are stale passkeys on client password manager due to deletion of passkeys from server side) for the same domain. When a user creates a new passkey for a domain where they already have an existing passkey stored, the client-side passkey storage will automatically overwrite the previous passkey with the newly created one for the same user, same domain at the time of passkey re-registration.

    This enhancement eliminates potential confusion from having multiple stale passkeys in the passkey manager on the given platform. The overwrite occurs seamlessly on the client side during the passkey creation process, with the new passkey being the active credential for that domain while the previous passkey is automatically removed from local storage.
- App Groups - Application Group Management enables administrators to better organize and control application access. A comprehensive set of new and updated REST APIs now facilitates the full lifecycle of application groups, from creation and retrieval to detailed updates and deletion, including the ability to add or remove individual applications. These API improvements are also reflected within the Admin Console's intuitive user interface. See Application Group Rest APIs and Admin Console - Application Groups.
- "Enhanced Resource Deletion" feature, empowers administrators to forcefully delete in-use IDSP resources like Applications and IDPs. By adding a "force=true" query parameter to existing DELETE APIs, resources are immediately removed from primary storage and hidden from the core operations. The resourcesCleanupScheduler job then systematically cleans up all outdated references from dependent policies and configurations. Additionally, the /admin/v1/ReferencingObjectsHelper API now includes an "allowResourceToBeForcefullyDeletable" parameter, guiding administrators on permitted force deletions. This feature is also supported using Admin Console. See Resource Deletion in IDSP.
- This release introduces enhancements for managing and controlling the use of user credentials, particularly phone numbers and email addresses, by allowing them to be categorized with specific tags and enabling applications to enforce usage policies based on these tags.
  - Users and administrators can now apply tags (for example, "business", "personal", "work", "home") to phone and email credentials. This allows for a clear distinction between different types of contact information for a single user.
  - LDAP Integration: A new LDAP configuration attribute, mvAttributeMapping, enables the mapping of multi-valued LDAP attributes to these tags. This flexible JSON-based configuration ensures that credential tags are accurately sourced and synchronized from your LDAP directory.
  - Applications can now define specific metadata to control which tagged credentials are permissible for various authentication and communication flows:
    - urn:iam:app:allowSmsCredsOfTag
    - urn:iam:app:allowEmailCredsOfTag
    - urn:iam:app:excludeCredsNnotUsedForXDays

  For descriptions, see Application Metadata topic.
- In addition to the email notifications sent when an administrator performs an action, the Admin Console can now send email or SMS notifications to end users for the following system-generated actions:
  - Account activation (sign in on new device)
  - Account lock or unlock
  - Password reset

The Admin Console can now also send email or SMS notifications to end users for the following actions performed either by administrators or users:
  - Credential activity (credential added, updated, or removed)
  - End user email changed
  - End user account deleted

  End-user email and SMS messages are sent using default message templates. However, you must configure which messages are sent and customize the content of the messages:
  - Configure which end-user notifications are sent in the Admin Console (see End-user Notifications under Service Configuration) or in tenant settings (using the `notifyEventsConfig` property described in Managing Settings).
  - Customize the email and SMS message contents using the Message Template API (see Managing Message Templates or #unique_80).
- The following features are related to Admin Console (UI):
  - The Admin Console provides the option to set a custom password reset URL that is provided to end users when you reset a password from the the Admin Console. With this release, you are also able to configure multiple vanity URLs on the **Internal Identity Store** configuration page. When end users request that a password be reset, the administrator can select which of these URLs to send to the end user in the password reset email.

    If end user password reset vanity URLs are configured on the **Internal Identity Store** configuration page, the **Account Activation Link** field on the **Add New User** page also uses these URLs.

    Note that any custom URLs that you set for the **Password Reset Changer URL** supersedes the vanity URLs. If you do not set any vanity URLs, or have configured a custom URL for the **Pasword Reset Changer URL**, vanity URLs are not displayed to the administrator when resetting an end user password nor in the **Account Activation Link** during user creation.
  - The **"Override max search results limit"** enhancement introduces a new checkbox in most of the resource filters (such as - application, certificates, policies, secrets, certs, IDP etc) that allows tenant administrators to bypass the standard search result limitations in the Admin console. When enabled, this feature overrides the default maximum of 500 Admin Resources displayed, which is normally enforced through the "maxAdminResourcesReturnedByGetAll" tenant setting. See Adding an Application .
  - The "Accounts" user interface has been enhanced with advanced search and filtering capabilities, providing administrators to quickly locate specific accounts using multiple search criteria and filter combinations. This enhancement significantly improves the user experience for managing large account databases. See Accounts.
- The new password expiration warning feature is governed by coordinated settings across two configuration sections in the Admin Console, ensuring consistent enforcement and user awareness.
  - Password validity is controlled through the **Internal Identity Store** under **Password Policies**, where the **“Password Expires After”** attribute defines the password lifespan. This configuration includes the **“Evaluate Password Expiration Rules On Authentication”** option, which enforces real-time validation by checking password expiration during every user authentication attempt. See Setting a Password Policy.
  - User-facing notifications are managed through the **Service Configuration** setting **“Days before password expiry to show a sign-in warning (0 disables warning)”**, which determines how far in advance users are alerted about an upcoming password expiration during sign-in. See Service Configuration.
- If you manually configure an Identity Provider, this release lets you configure Symantec Identity Security Platform to sign SAML authentication requests sent to the Identity Provider. You can choose whether to sign the request using the key pair provided by the Identity Provider or your tenant key. See Identity Providers (Admin Console configuration) or Creating Identity Provider using "/IdentityProviders" API (API configuration) for details.
- MTLS - This release introduces OAuth Mutual Transport Layer Security (mTLS), bolstering security for client interactions with critical OAuth endpoints (Token, Revoke, Introspect) by authenticating clients via their certificates. See Configuring OAuth mTLS between Symantec Identity Security Platform Clients and Cluster .
- Improved LDAP Configuration Updates with bindPasswordSecretAlias: Previously, when modifying an LDAP configuration using the PUT method, the bindPassword field was often pre-filled with "" if bindPasswordSecretAlias was in use. This caused updates to fail unless the bindPassword field was manually cleared before each PUT operation, as "" is not a valid password.

With this enhancement, if the bindPasswordSecretAlias value is supplied in the LDAP configuration during a POST/PUT/GET operation, the bindPassword field is not shown in the response. The response only includes the bindPasswordSecretAlias. See Managing LDAP Configuration.

- This release introduces comprehensive input validation for all REST API endpoints (includes POST, PUT, and PATCH operations), providing robust protection against malicious input and improving overall system security. See Using REST APIs.
- Custom user attributes of type `string` , `int` , and `date` support multiple, comma-separated values. See User Schema Extensions for the Identity Service.
- Automatic creation of the user display name is streamlined. The user display name (`displayName`) is generated from the Display Name Template defined in your identity store. See Internal Identity Store Configuration.

  By default, the Display Name Template is set to `givenName` and `familyName` . You can customize this template to specify user attributes of your choice. If some or all of the values specified in the DisplayName Template are present, the display name is set using the values that are present. If none of the values specified in the Display Name Template are present, the display name defaults to the user `Login` value.

  For details about display name generation, see the following topics:
  - User and Group Schemas
  - Identity Provider Discovery - Just-In-Time (JIT) User Provisioning
  - Configuring the Zero Footprint Identity
- This release adds a Resource Counter Helper API that allows clients to retrieve a count of all resources in a tenant, by resource type. This API also allows clients to retrieve a count of tenants in the system. See Resource Counter Helper API.
- You can now set whether groups are read-only using the GroupOperator API. You can also use the GroupOperator API to view the read-only status of a group. See Group Operator API.
- When using the Group API or Bulk API to perform a PUT, POST, or PATCH operation to add multiple members to a group, you can set the **allowPartialSuccess** flag to continue processing if the API cannot complete the group membership add operation on all users. Without this flag, the APIs returns an error and make no changes if it cannot correctly process all group membership add operations. Setting the **allowPartialSuccess** flag allows the API to capture invalid operations while still processing valid operations. See Group API and Bulk API.
- The User object is updated to include the `groupMembershipLastModified` attribute. This attribute identifies the date and time that the user was added to or removed from a group.
- If a user that is assigned as a manager to other accounts (users, service accounts, and so on) is deleted using the User API, the accounts are not automatically assigned a new manager. The new `replacementManager` parameter assigns the manager that is included in the parameter as the new manager for all accounts that were previously assigned to the deleted manager. See User API.
- The behavior of ABAC and GBAC is updated to reflect the concept of unconditional access, unconditional deny, and conditional access. For example, if a qualifier (C, R, U, or D) is missing from a resource type, the operation is treated as unconditionally denied. If resource types have multiple scopes, the priority order is: unconditional access, unconditional deny, and conditional access. See Scoping Constraints Overview.
- Added the field `denyOps` to scoping constraints. If any scoping constraint for a resource type has `denyOps` defined, the operation is denied for that resource type. For example, if a resource type has `denyOps=U` , the principal cannot update any resources of this type. See Scoping Constraints Overview.
- Previously, if the value of `allowStartIndexSearchParam` in the internal Identity Store is set to `false` , and you included a `startIndex` value in your search parameters, the search threw an error. Now, the search uses the `startIndex` provided in the search parameters if the principal has the `t.allowstartindexsearchparam` privilege. Otherwise, the search sets the start index to 1. See Scopes and APIs Table.
- Added two optional search parameters to the User API and the Group API:
  - The `skiplimit` parameter allows principals with the correct privileges to retrieve all users and groups that match the filter without bounding to the value of the `scimSearchLimit` setting that is defined in the Identity Store configuration.
  - The parameter `countonly` allows a SCIM search to return the number of users or groups present in the tenant.

See User API and Group API.

- Added Identity Service SCIM error code 89410860 to identify when user and group searches time out. See Identity Service SCIM Error Codes. This topic has also been updated to include previously undocumented error codes.
- Added two search response headers:
  - X-MAX-RESOURCES-LIMIT-REACHED: This response header is returned in the following circumstances:
    - For SCIM users and groups, the response header is shown when the number of matching users or groups is equal to or greater than the value of:
      - The `scimSearchLimit` value defined in Internal Identity Store configuration; or,
      - The value of `count` in the API request.
    - For other resources, the header is shown when the number of matching resources equals or is greater then the `maxAdminResourcesReturnedByGetAll` value defined in Tenant Settings.
  - X-TOTAL-COUNT: This response header is returned for user and group SCIM searches. Adding this header brings the user and group searches into parity with all other resource searches.
- The Admin and Scheduler pods max memory limit has been increased to 3Gi (Gibibyte).
- A new OIDC Grant type called, "Resource Owner" has been added to the list of grant types that can be used. However, this grant type is explicitly noted as a legacy grant type that should only be used when a legacy component specifically requires it. See Resource Owner grant type.
- Additional validation for Identity Provider (IDP) configuration using Zero Footprint Identity (ZFP). When modifying existing IDP used as an authentication factor to enable Zero Footprint Identity and Identity Verification, the system now requires the "allowZfpIdpVerifyIdentity" metadata field to be specified on the IDP as "true". Such identity verification is used only in IDP-as-a-Factor flows and not in Bring-Your-Own-Identity flows, hence to minimize confusion regarding identity verification with ZFP IDPs, use of IDP metadata is required. See IDP Metadata properties.
- SAML Application now supports the ${saml_idp_init} placeholder for the launcherUrl field, which automatically populates the URL with the "app.samlEffectiveIdpInitiatedUrlPreferred" value during launcherUrl invocation, eliminating the need for manual URL entry.
- The following four tenant settings are being deprecated and will no longer be exposed through the ".../auth/v1/BrandingSettings" API response:
  - loginLabel
  - loginHint
  - passwordLabel
  - passwordHint
- ***New key-value pairs added in the OOTB locales in SignIn UI***

  To know how to update these key-value pairs in your custom locale, see Updating the custom locale content manually when additions are made in the OOTB Locales.
  - The following new key pairs have been added in this release for Signin UI:

Index	Value
passwordExpiry.passwordExpiresInHeader	Password Expires In
passwordExpiry.daysLabel	{{count}} Days
passwordExpiry.message	For security reasons, you'll be required to set
passwordExpiry.btnOK	OK
identity.passwordreset.screen_pwreset.submsg_success_activate	Your account has been successfully acti
actionResult.title_unlockAccount	Unlock Account
actionResult.unlockSuccess	Your account has been successfully unl
actionResult.title_resetPassword	Reset Password
actionResult.resetpwSuccess	Your password has been reset success

actionResult.title_activateAccount	Activate Account
actionResult.activationSuccess	Your account has been successfully act
identity.verifyPage.screen_title_email	Verify Your Email
identity.verifyPage.screen_title_phone	Verify Your Phone
identity.verifyPage.emailVerified	Your email has been successfully verifie
identity.verifyPage.phoneVerified	Your phone has been successfully verifi
userConsent.header	{{appName}} wants the following access
userConsent.processDataNote	Review the <span class=\"consentUsagePolic understand how this data will be used.
userConsent.btnCancel	Cancel
userConsent.btnAllow	Allow
userConsent.btnClose	Close

Fixed Issues

- An error was discovered in the IDSP risk service's device recognition component. This flaw bypassed intended device recognition security measures, potentially leading to inaccurate risk assessments. (DE641058)
- Continued use of Bitnami images in IDSP after Bitnami has announced a significant change to their container image licensing model, transitioning from free open-source images to a paid subscription model. (DE643277)
- iOS SDK was returning incorrect codes and messages for Passkeys Login. (DE648159)
- Biometric error flows return identical error codes for both Enrollment and Authentication scenarios when duplicate passkeys are detected and the biometric popup is closed, resulting in incorrect error messages being displayed to users. (DE648771)
- Custom Providers created prior to the 3.4.2 release do not contain the "skipUserStrikeCountIncrement" field in their configuration. When these legacy providers are used in version 3.4.2, the system throws a 400 Bad Request exception, preventing authentication operations from completing successfully. (DE651841)
- Identified issues with the passkey enrollment process in mobile applications, including creation of multiple passkeys from the same device and missing AAGUID (Authenticator Attestation Globally Unique Identifier) in the FIDORegChallengeVerifier service, resulting in passkeys without platform identification. (DE644244)
- If an administrator initiated a workflow (such as password reset) from an internal vanity URL that generated end user-facing emails, the links in the emails did not include the end-user vanity URL. (DE645499)
- OOTB Sign-in application did not automatically detect device type (desktop, mobile, tablet) and include this information in the clientContext of the .../auth/v1/authenticate endpoint. This prevented Authentication Policies from utilizing Context conditions to implement device-specific authentication factor requirements and policy configurations. (DE644726)

Known Issues

- After upgrading from a previous version of IDSP to IDSP v4.0, all existing user sessions are automatically terminated and users are required to re-authenticate. (DE659394)

Upgrade Considerations

IMPORTANT

- **IDSP Supports Incremental Upgrades**:
  IDSP facilitates incremental upgrade paths. If you plan to upgrade to the latest release, IDSP v4.0, and you are currently using an earlier version, it is advisable to first upgrade to the latest patch of the subsequent minor release before moving on to IDSP 4.0.

IDSP uses the following format for naming release versions: MAJOR.MINOR.PATCH. For example, in the release version 4.0.x, "4" is the MAJOR version, "0" is the MINOR version, and "x" is the PATCH version.

Upgrade Use Cases:

- ***Upgrade Scenario 1***: Upgrade from IDSP v3.3 to IDSP v4.0
  Recommended Upgrade Path: You should follow this upgrade path- **3.3.x > 3.4.x > 4.0**
  Example: 3.3.6 > 3.4.8 > 4.0
- ***Upgrade Scenario 2***: Upgrade from IDSP v3.4.x to IDSP v4.0
  Recommended Upgrade Path: You can directly proceed with the upgrade to v4.0 without any intermediate steps.
  Example: 3.4.8 > 4.0

- **Oracle SQL file Updates**: This release includes Oracle database schema changes that may require manual SQL file updates for environments with separate database administration. For detailed upgrade procedures and SQL file application order, see the Oracle SQL File Updates topic.

See Upgrading IDSP.

Rollback Consideration

- When rolling back from IDSP v4.0.0 to any previous version, special attention must be paid to Authorization Policy PUT operations to ensure compatibility.
  - The "effect" field introduced in version 4.0.0 is not supported in earlier releases of IDSP.
  - Before performing Authorization Policy PUT operations after a rollback, ensure that the "effect" field is removed from all request payloads, otherwise requests containing the "effect" field will be rejected.
- The "class" path for the message providers have changed in IDSP v4.0. Therefore, before rolling back from IDSP v4.0 to any previous version, you must revert the old class path values in the message provider configuration as they are not backward compatible.
  The following table shows the Message Provider Class Path migration introduced in v4.0:

Old Path	New Path
com.broadcom.layer7authentication.mds.service.providers.sms.TwilioSmsService	com.broadcom.layer7authentication.core.service.mds.pr
com.broadcom.layer7authentication.mds.service.providers.email.TwilioSendGridEmailService	com.broadcom.layer7authentication.core.service.mds.pr EmailService
com.broadcom.layer7authentication.mds.service.providers.push.PushFCMService	com.broadcom.layer7authentication.core.service.mds.pr e
com.broadcom.layer7authentication.mds.providers.email.EmailIEService	com.broadcom.layer7authentication.core.service.mds.pr
com.broadcom.layer7authentication.mds.service.providers.email.TwilioSendGridEmailService	com.broadcom.layer7authentication.core.service.mds.se ndGridEmailService
com.broadcom.layer7authentication.mds.service.providers.sms.ClickATellSMSService	com.broadcom.layer7authentication.core.service.mds.se llSMSService
com.broadcom.layer7authentication.mds.service.providers.email.SMTPEmailService	com.broadcom.layer7authentication.core.service.mds.se mailService

For more information, see Managing Message Providers Configurations.

Symantec Identity Security Platform Kit Location and Build Details

See Release 4.0 Build Details.

IDSP Knowledge Articles

All IDSP knowledge articles can be found here, Symantec Identity Security Platform Knowledge Articles.

Release Date

The IDSP v4.0 release was distributed on January 12, 2026.

Oracle SQL File Updates

Manual Oracle SQL file updates required for environments with separate database administration.

This release introduces Oracle database schema changes that may require manual SQL updates in environments where database administration is handled separately. While the schema is normally upgraded automatically as part of the dataseed job, certain Oracle database instances may need to be updated manually outside the standard process when database administration is decoupled from the Symantec Identity Security Platform deployment.

Fresh Install of Symantec Identity Security Platform

For fresh installations of IDSP, the database schema setup is streamlined and simplified.

If manual updates of the schema are required for a fresh install, follow these steps:

1. **Download and run** the consolidated Oracle SQL file from the Symantec Identity Security Platform Support Portal > My Downloads (Formarly known as VIP Authentication Hub in the portal and select the appropriate version number). For fresh installations, use the single consolidated Oracle SQL file (**IDSP_4.0_Oracle_SQL_Fresh_Install.sql**) that contains all necessary schema updates, rather than applying multiple individual files.
2. Once the **consolidated DB file** has run successfully, proceed with the installation.
3. **Enable the property**, `"ssp.featureFlags.dataseed.skipSchemaInit"="true"` before completing the installation of IDSP.

Note: Fresh installations benefit from using a single, comprehensive SQL file that includes all schema changes, eliminating the need to apply multiple incremental updates sequentially.

Upgrade to IDSP from Previous Releases

For upgrades from previous versions of IDSP, incremental SQL files must be applied in the correct sequence to ensure proper schema migration.

If manual updates of the schema are required for upgrades, instead of performing an automatic upgrade (step 12), follow these steps:

1. **Download and run** the individual Oracle SQL files from the Symantec Identity Security Platform Support Portal > My Downloads (Formarly known as VIP Authentication Hub in the portal and select the appropriate version number). This manual update is required due to changes in the Oracle SQL database in this release.
2. Apply the SQL files in the correct sequential order as shown in the table below.
3. Once all required **DB files** have run successfully, proceed with the upgrade.
4. **Enable the property**, `"ssp.featureFlags.dataseed.skipSchemaInit"="true"` before upgrading to the latest version of IDSP.

Example Upgrade Scenario:
To upgrade from IDSP v3.4 to IDSP v4.0, you must apply 3.4.x, and 4.0 SQL files depending on which patch you are moving from. If you are on a specific patch version of IDSP and you intend to upgrade to another patch version, download and apply only the Oracle SQL files relevant to the patches you are moving to.

This is the order to apply the SQL files for upgrades:

Seq. No.	File version
A	VIP_Authentication_Hub_3.4_Oracle_SQL_file.sql
B	VIP_Authentication_Hub_3.4.1_Oracle_SQL_file.sql
C	VIP_Authentication_Hub_3.4.2_Oracle_SQL_file.sql
D	VIP_Authentication_Hub_3.4.3_Oracle_SQL_file.sql
E	VIP_Authentication_Hub_3.4.4_Oracle_SQL_file.sql
F	VIP_Authentication_Hub_3.4.5_Oracle_SQL_file.sql
G	VIP_Authentication_Hub_3.4.6_Oracle_SQL_file.sql
H	VIP_Authentication_Hub_3.4.7_Oracle_SQL_file.sql
I	IDSP_4.0_Oracle_SQL_upgrade_file.sql
J	IDSP_4.0.2_Oracle_SQL_upgrade_file.sql

Reference Topics

- Upgrading IDSP.

Identity Security Platform Kit Location and Build Details

Contains Symantec Identity Security Platform release build details.

This topic includes the links to all the major and minor releases of Symantec Identity Security Platform:

- IDSP v4.0.2 - Build No 1075 - May-2026
- IDSP v4.0.1 - Build No 1015 - Jan-2026
- IDSP v4.0 - Build No 1157 - Jan-2026

Identity Security Platform Release 4.0.2 Build Details

Symantec Identity Security Platform Release 4.0.2 Build Details

The latest SDK and Sample Source Code kit can be downloaded from the Support Portal > My Downloads.

Build Details for this Release

Chart Name	Chart Version	Chart Description
ssp_helm_charts/ssp-infra	4.0.2+1075	Helm chart to deploy Fluent Bit, Prometheus Metrics Adapter, Arango DB operator, Database (internal mode)
ssp_helm_charts/ssp	4.0.2+1075	Helm chart to deploy Security Services Platform
ssp_helm_charts/ssp-sample-app	4.0.2+1075	Helm chart to deploy a sample application for the Secure Services Platform
ssp_helm_charts/ssp-data "ssp-data" chart gets updated on a weekly basis. It is recommended to perform a "**helm repo update**" to get the latest chart.	2026.20	Helm chart to deploy Security Services Platform Data Loader

ssp_helm_charts/ssp-symantec-dir	4.0.2+1075	Helm chart to deploy a Demo ID Store (CA Directory)

Broadcom Container Image IDs

The following is the list of container images along with their corresponding image digests. You can use this list to pull the images required for this version of IDSP:

- securityservices.packages.broadcom.com/admin-svc:4.0.2.1075@sha256:43648b6824a475efeec8a00eb87c100af0359cc2096b57390457a27d3a713b4a
- securityservices.packages.broadcom.com/authhub-adminconsole:4.0.2.1075@sha256:8b35c7c140ca37a3147d3299ba51fcbf75975b83814c477bd2158f23646bd1ca
- securityservices.packages.broadcom.com/authhub-selfserviceconsole:4.0.2.1075@sha256:1ac14c4d8c7a2c996398957f781392f730d0f69387a0e8db0158d6f1cd0aac02
- securityservices.packages.broadcom.com/authhub-signin-legacy:4.0.2.1075@sha256:1a71c570d8095b4a57c4043e0a33c459fe3e1dca27bb8974faf37586041e1dca
- securityservices.packages.broadcom.com/authhub-signin:4.0.2.1075@sha256:30c4ed2a053c37c0c1e3b4e2d26a01552ea5709879934566b098cc4c72b63844
- securityservices.packages.broadcom.com/authmgr:4.0.2.1075@sha256:4ff314559fbe1161884497e52096113411dd9c6ff034f0246c8f665d53e14f5b
- securityservices.packages.broadcom.com/authswagger:4.0.2.1075@sha256:47dd561ca7d1abcfc7607fb429a513390e2a0fc2cc2fa297d323b997a38dfd3f
- securityservices.packages.broadcom.com/factors-svc:4.0.2.1075@sh256:9616fb79db52076b7df5fdbb6bab78e92968efa765b55bc86ba8c91c678da958
- securityservices.packages.broadcom.com/geolocation-svc:4.0.2.1075@sha256:19bf381aeb699b81b353134556d44d8a09a2e14e9872f310a1ce7b307bad1c11
- securityservices.packages.broadcom.com/iaservice:4.0.2.1075@sha256:9897516da8730c6fc4ed9cb74a4ee0fd40f95cd4436f96fd3c8b61ac659b9e5f
- securityservices.packages.broadcom.com/identity-svc:4.0.2.1075@sha256:e20284ba458872539392f1154e1a959cf606c896c6c541d4f357d849be30197c
- securityservices.packages.broadcom.com/machineid-svc:4.0.2.1075@sha256:61f15dbc0b40620546f5d6ec657be1c3ad70308e4e5752e973de11c33af06f80
- securityservices.packages.broadcom.com/oidc-svc:4.0.2.1075@sha256:1473ab147ce6b71cb63a8f641886d439f1625760390d3aa5ad1d9760478948d5
- securityservices.packages.broadcom.com/opa:4.0.2.1075@sha256:95e4425e7061a290bd44fca19f057f07f895c09aa67f9d3928f2f455e1c8329e
- securityservices.packages.broadcom.com/proxysidecar:4.0.2.1075@sha256:4faaeaf53f1937acb5d17c030c1f45fd425c4af39b6dee51b0f94352408294a0
- securityservices.packages.broadcom.com/ssp-db-initializer:4.0.2.1075@sha256:34a3df9005b1d6bcc8427a4419499590bf088250d44568f5f0ba3c8e54b55559
- securityservices.packages.broadcom.com/ssp-vault-secrets-initializer:4.0.2.1075@sha256:ab3abdc0e41877ab6d50a47f8b3acc747ff740068d74054d3827a18cfe0c1319
- securityservices.packages.broadcom.com/ssprisk-svc:4.0.2.1075@sha256:8b843d1450e6dc8ed0dd7c5205e46e6964d589f4e557bf4d50cb824e3b7e16a3
- securityservices.packages.broadcom.com/ssprouter:4.0.2.1075@sha256:7b5e02b3854477d3865240665866622e80a37ba0530de2a9aa2c2caecf95847e
- securityservices.packages.broadcom.com/ui-helper-svc:4.0.2.1075@sha256:38232491ee97181ab8aa8dbcf3f5083cb6399a9b14a17bbdc77a7c3888c56345
- securityservices.packages.broadcom.com/hazelcast-enterprise:5.7.0-p0@sha256:0a65a82f346aeba8a2648f2dbea9db9f72cc212c2b056986e6e8b4d61ac10505

The network data image contains fraud details and is released every week. There may be a newer version than the following version:

- securityservices.packages.broadcom.com/network-data-loader:2026.20@sha256:06b3351d8be2e0e7f48f3b4b250547c3763e4604bc97df4c3abe806c7e9e3da7

The following is the list of optional images used for the sample applications:

- securityservices.packages.broadcom.com/cadir:4.0.2.1075@sha256:77525d39e49559a477ecad4432e65aaefbd85f1c18ff6f8bf41053761979da2b
- securityservices.packages.broadcom.com/sample-app:4.0.2.1075@sha256:520f8c65a4a7d6de2deb5622a8386271a81e818a038fe39f7a1fbc7888fc09a7
- securityservices.packages.broadcom.com/sample-rp:4.0.2.1075@sha256:5860fe1da0769f88e6f0bbc5b744f0111009ef57c512a94a1aa96ff91197ab6b
- securityservices.packages.broadcom.com/spi-demo:4.0.2.1075@sha256:57d809c674dabd6ddeb007dcbe00f80953c7d77b83fa97c888ccab6448471599

To know how these container images can be pulled from the Broadcom registry, see Pulling images from the Broadcom registry.

Oracle SQL DB file

- The IDSP v 4.0.2 does not include any Oracle database schema changes.

Postman Collection

- The IDSP **v4.0.2** Postman Collection is available for download from the Support Portal > My Downloads
- Best Practice for using the IDSP Postman collection, see Using IDSP postman collection

Identity Security Platform Release 4.0.1 Build Details

Symantec Identity Security Platform Release 4.0.1 Build Details

The latest SDK and Sample Source Code kit can be downloaded from the Support Portal > My Downloads.

This topic includes the following sections:

- Build Details for this release
- Broadcom Container Image ID's
- Oracle SQL DB file
- Postman Collection

Build Details for this Release

Chart Name	Chart Version	Chart Description
ssp_helm_charts/ssp-infra	4.0.1+1015	Helm chart to deploy Fluent Bit, Prometheus Metrics Adapter, Arango DB operator, Database (internal mode)
ssp_helm_charts/ssp	4.0.1+1015	Helm chart to deploy Security Services Platform
ssp_helm_charts/ssp-sample-app	4.0.1+1015	Helm chart to deploy a sample application for the Secure Services Platform

ssp_helm_charts/ssp-data "ssp-data" chart gets updated on a weekly basis. It is recommended to perform a "**helm repo update**" to get the latest chart.	2026.04	Helm chart to deploy Security Services Platform Data Loader
ssp_helm_charts/ssp-symantec-dir	4.0.1+1015	Helm chart to deploy a Demo ID Store (CA Directory)

Broadcom Container Image IDs

The following is the list of container images along with their corresponding image digests. You can use this list to pull the images required for this version of IDSP:

- securityservices.packages.broadcom.com/admin-svc:4.0.1.1015@sha256:a3ff08da001713986c93ab2f84381eaf667936857d17e184d13f315728283923
- securityservices.packages.broadcom.com/authmgr:4.0.1.1015@sha256:1af54cf8f7d7ad25f9774a65b9212f1af3c68bddf77d88945a9716f9fea68529
- securityservices.packages.broadcom.com/factors-svc:4.0.1.1015@sha256:9fe99d6840570f66c360dbd1eec16795f11391d3b82765cfa6ccd5084959e534
- securityservices.packages.broadcom.com/oidc-svc:4.0.1.1015@sha256:07d5664cd6b2d1e1450031bd1852ab36cf9975076e3c233d3dafe5f753f14410
- securityservices.packages.broadcom.com/opa:4.0.1.1015@sha256:c24bf995cf6a2e895d6dd1a72e6a24cd5c9771a629436e47c884b0b0013ea792
- securityservices.packages.broadcom.com/authswagger:4.0.1.1015@sha256:1a3785595959a8dfdf4e501ad5cc52f2dc3d77a4406afab30ed7be01f4e02cdb
- securityservices.packages.broadcom.com/ssp-db-initializer:4.0.1.1015@sha256:2d0076ab5a9ebff4c23b50903825b9192ff44a8437acf3887336f3f09e91218c
- securityservices.packages.broadcom.com/authhub-adminconsole:4.0.1.1015@sha256:b2716281ce771137559d626987c6170b11ad6b14b978b824d6ae9dc11da8c137
- securityservices.packages.broadcom.com/authhub-signin:4.0.1.1015@sha256:d2d5e250df9f61df355f17d209bc98e29b96ded463b86b0aab70c3109f0ef459
- securityservices.packages.broadcom.com/authhub-signin-legacy:4.0.1.1015@sha256:ec9d5e7081cb774b483f21dd99004a37f33ee5815524ab015e2f43f9714907cb
- securityservices.packages.broadcom.com/iaservice:4.0.1.1015@sha256:4fe64044a29d553d7aeff23267dee392d55c0918c65f7c3616c24ad54ff60ad9
- securityservices.packages.broadcom.com/ssp-vault-secrets-initializer:4.0.1.1015@sha256:90a34be7a5028b75f1d1926e6efc49bd5a6cbb723019e9fd71f2fb5660ad1926
- securityservices.packages.broadcom.com/identity-svc:4.0.1.1015@sha256:9f776773da7c2a4db4f489ad8281f1bee3553a2a0f023358848d926d97e285be
- securityservices.packages.broadcom.com/ssprisk-svc:4.0.1.1015@sha256:bd8c1fc3f3889dc3377f079c58da072566d7b310ba2e2bbc7f9a90225c6f692c
- securityservices.packages.broadcom.com/ui-helper-svc:4.0.1.1015@sha256:65a288e38d4b6f0abccd7398a9625effe4f0fd21412e97d97a589c7ef464712e
- securityservices.packages.broadcom.com/geolocation-svc:4.0.1.1015@sha256:6971630bcd12eb3b50a6763d7a7fc3600f95a80d2e2a46364452e036b3746561
- securityservices.packages.broadcom.com/authhub-selfserviceconsole:4.0.1.1015@sha256:935a051748159a3ff223e5f180c00597833ec287f6aabad858a841cb9b46d046
- securityservices.packages.broadcom.com/machineid-svc:4.0.1.1015@sha256:b90b3a5ef9b8d92c31824d01b72e0f4fd605bdee6a06591314a165f38d83ea91
- securityservices.packages.broadcom.com/proxysidecar:4.0.1.1015@sha256:bd931d6e27aedacdc04880d9b485cf2ff6f571def518024694d5d3dfae30d598
- securityservices.packages.broadcom.com/hazelcast-enterprise:5.3.5-m16-p3@sha256:52478dc65468bc18a36641a688b1c93586f682b945609b57dc5ae0f7ea8fe2ad

The network data image contains fraud details and is released every week. There may be a newer version than the following version:

- securityservices.packages.broadcom.com/network-data-loader:2026.04@sha256:44e4faab4b91a724091c402820042ca97a0f9b53f7236d9ad0deb0e0a7b03f14

The following is the list of optional images used for the sample applications:

- securityservices.packages.broadcom.com/cadir:4.0.1.1015@sha256:77525d39e49559a477ecad4432e65aaefbd85f1c18ff6f8bf41053761979da2b
- securityservices.packages.broadcom.com/sample-app:4.0.1.1015@sha256:40a793dea06bc3ff396d6adb89cc83fec8787bd1c3550fa08600167d60041c7d
- securityservices.packages.broadcom.com/sample-rp:4.0.1.1015@sha256:72551e29e7853c2a2d743101cb48075da3ba343ebfb3ee17cc4b0e7bfc9a8236
- securityservices.packages.broadcom.com/spi-demo:4.0.1.1015@sha256:fc8527f0e2a42be336c6ae2cb043d5051584be48b5583b12d5c8130c958fdc86

To know how these container images can be pulled from the Broadcom registry, see Pulling images from the Broadcom registry.

Oracle SQL DB file

- The IDSP v 4.0.1 does not include any Oracle database schema changes.

Postman Collection

- The IDSP **v4.0.1** Postman Collection is available for download from the Support Portal > My Downloads
- Best Practice for using the IDSP Postman collection, see Using IDSP postman collection

Identity Security Platform Release 4.0 build location

Symantec Identity Security Platform Release 4.0 Build Details

The latest SDK and Sample Source Code kit can be downloaded from the Support Portal > My Downloads.

This topic includes the following sections:

- Build Details for this release
- Broadcom Container Image ID's
- Oracle SQL DB file
- Postman Collection

Build Details for this Release

Chart Name	Chart Version	Chart Description
ssp_helm_charts/ssp-infra	4.0+1157	Helm chart to deploy Fluent Bit, Prometheus Metrics Adapter, Arango DB operator, Database (internal mode)
ssp_helm_charts/ssp	4.0+1157	Helm chart to deploy Security Services Platform
ssp_helm_charts/ssp-sample-app	4.0+1157	Helm chart to deploy a sample application for the Secure Services Platform

ssp_helm_charts/ssp-data "ssp-data" chart gets updated on a weekly basis. It is recommended to perform a "**helm repo update**" to get the latest chart.	2026.01	Helm chart to deploy Security Services Platform Data Loader
ssp_helm_charts/ssp-symantec-dir	4.0+1157	Helm chart to deploy a Demo ID Store (CA Directory)

Broadcom Container Image IDs

The following is the list of container images along with their corresponding image digests. You can use this list to pull the images required for this version of IDSP:

- securityservices.packages.broadcom.com/admin-svc:4.0.0.1157@sha256:b8b89508e70df48e912446f3a7b3eec10bead4eecbae3b69a2e62306ec4b1bd8
- securityservices.packages.broadcom.com/authmgr:4.0.0.1157@sha256:a5b6a822f650a97cf3fd8f3e0213136e7472a10129a8e553ece80fad4acf195a
- securityservices.packages.broadcom.com/factors-svc:4.0.0.1157@sha256:420c7c8c482693f8ffa4078222628a556bff6a53019592048e7d8e762f8449eb
- securityservices.packages.broadcom.com/oidc-svc:4.0.0.1157@sha256:72f5b4513964b618bf990cb35937656b63685df0d8a34306063afc16226bf3be
- securityservices.packages.broadcom.com/opa:4.0.0.1157@sha256:4f96cdd0b48c7339e2c396f52e870ddf14b706839f829a02d103bb2ff0207117
- securityservices.packages.broadcom.com/cadir:4.0.0.1157@sha256:77525d39e49559a477ecad4432e65aaefbd85f1c18ff6f8bf41053761979da2b
- securityservices.packages.broadcom.com/authswagger:4.0.0.1157@sha256:907ce0e58960c80f4b14b87968bc1fe37e79b658119833fa67b3e9720715ad49
- securityservices.packages.broadcom.com/ssp-db-initializer:4.0.0.1157@sha256:cb16b3f3632ee8861796e3ad38c96dbc1540f7de16f5f3679849f62c613c03fe
- securityservices.packages.broadcom.com/authhub-adminconsole:4.0.0.1157@sha256:f9d8ddc01294cf4127cfed2b525601c5edd9bee75ef0c66a7d0b2ab8dd6cfd59
- securityservices.packages.broadcom.com/authhub-signin:4.0.0.1157@sha256:fc58e25b4c22e792c57cb0f22279e006737842067f2adae64fac0b16a87e3401
- securityservices.packages.broadcom.com/authhub-signin-legacy:4.0.0.1157@sha256:8e6f2f678156cd06864982ac480719c0bcd57f6f7a43411d77403d1827d71b05
- securityservices.packages.broadcom.com/iaservice:4.0.0.1157@sha256:e67f4817b22a1094bb73161bc00f5a0906f372f157a78bc2e028a623fa5a0a98
- securityservices.packages.broadcom.com/ssp-vault-secrets-initializer:4.0.0.1157@sha256:fcb68047ddf6815a8b1688e5e4c3efe64d7ddc4427989706581fd60cf751700d
- securityservices.packages.broadcom.com/identity-svc:4.0.0.1157@sha256:5b46fa5ee5cd773438b5a06a41c218dec587ef1ee7b09360279f701c77d7ed10
- securityservices.packages.broadcom.com/ssprisk-svc:4.0.0.1157@sha256:ba693b5bb24d008b3c69cb6f361912a83a2d606ce4ccae5774fdeecbaaffbeb2
- securityservices.packages.broadcom.com/ui-helper-svc:4.0.0.1157@sha256:924882c5bd28ace1ce8f7530cbc97213aed84a9ff1e0ca4688ec3af628097a1a
- securityservices.packages.broadcom.com/geolocation-svc:4.0.0.1157@sha256:67d1b0972769b101dbabccfef2646309779a664f68f26ffda2ead960e3edb53c
- securityservices.packages.broadcom.com/authhub-selfserviceconsole:4.0.0.1157@sha256:0db043116c1fa1a8a026cbcbd69140b0e269bc8980d41a06a36734b78b106ae1
- securityservices.packages.broadcom.com/machineid-svc:4.0.0.1157@sha256:68cbe0e52fb96c516d0a6967c65c50b45ab6da4b67e217c77917c6f80fe729a5
- securityservices.packages.broadcom.com/proxysidecar:4.0.0.1157@sha256:bc5e404c77a1c991653f3b42db450d1eb6a0d57f8e676fffdc3f816dbd435c81
- securityservices.packages.broadcom.com/hazelcast-enterprise:5.3.5-m16-p2@sha256:2ffaf52e3b0f4ba9c29cf9b0fb7a2098c8eb4b08450dcc87ab2eadeffc556454

The network data image contains fraud details and is released every week. There may be a newer version than the following version:

- securityservices.packages.broadcom.com/network-data-loader:2026.01@sha256:c6294398d4b4d7d85ad305821cb3d5e6f526412dae0e5efecf5aa7e9858196fd

The following is the list of optional images used for the sample applications:

- securityservices.packages.broadcom.com/cadir:4.0.0.1157@sha256:77525d39e49559a477ecad4432e65aaefbd85f1c18ff6f8bf41053761979da2b
- securityservices.packages.broadcom.com/sample-app:4.0.0.1157@sha256:51514191fa47a0c8a15aec43833d0f4ec98ffcef7661f29a16cc82799082f7ec
- securityservices.packages.broadcom.com/sample-rp:4.0.0.1157@sha256:d586747b92ce7f3470578d7ac38cb8085813a50e9b70bcf48f8d93adaf0263b5
- securityservices.packages.broadcom.com/spi-demo:4.0.0.1157@sha256:4bbd9d4c8bc00a65ef540afc564bf6267f93b756515164b6a3e5b4f68b4f083a

To know how these container images can be pulled from the Broadcom registry, see Pulling images from the Broadcom registry.

Oracle SQL DB file

- The IDSP **4.0** Oracle SQL DB file can be downloaded from the Support Portal > My Downloads (Search for Identity Security Platform in the portal and select the appropriate version number). For more information, see Oracle SQL File Updates topic.

Postman Collection

- The IDSP **v4.0** Postman Collection is available for download from the Support Portal > My Downloads
- Best Practice for using the IDSP Postman collection, see Using IDSP postman collection

Overview

Learn about the Symantec Identity Security Platform (IDSP).

About Symantec Identity Security Platform (IDSP)

The Symantec Identity Security Platform (IDSP) is a comprehensive, modern identity and access management solution built on cloud-native, container-based architecture. Designed with an API-first approach, IDSP delivers enterprise-grade authentication, authorization, and identity services through a flexible, policy-driven framework that adapts to your evolving security requirements.

Whether you are implementing passwordless authentication, securing APIs with OAuth, managing user identities, or building zero trust architectures, IDSP provides the complete toolkit administrators and developers need to deliver trusted access across modern applications and legacy systems alike.

This topic includes the following sections:

- Core Capabilities
- Architecture Highlights
- Use Cases
- Getting Started

Core Capabilities

Policy-Driven Security Framework

At the heart of IDSP is a unique policy orchestration engine that eliminates hard-coded security logic. Define and modify authentication journeys, identity federation workflows, and risk evaluation rules through flexible policies rather than configuration files or code changes. This architecture allows security teams to adapt quickly to new threats and business requirements without vendor dependencies or extensive customization efforts.

- **Value for Administrators**: Adjust security postures in response to emerging threats or compliance requirements without architectural changes or system downtime.
- **Value for Developers**: Build applications that leverage centralized authentication policies without embedding security logic in application code.

Passwordless Authentication and Strong Security

IDSP supports industry-leading authentication methods based on FIDO2 standards, including biometric authentication, hardware security keys, and passkeys (supporting cross-device scenarios). Users can authenticate using fingerprint, facial recognition, or physical security keys—eliminating password vulnerabilities while delivering frictionless user experiences across desktop, mobile, and web applications. Traditional authentication factors such as SMS and Email-delivered security codes and software authenticator applications are also supported.

- **Value for Administrators**: Reduce help desk burden from password resets and account lockouts with “self-service capabilities”, while strengthening the security posture by eliminating credential theft vectors.
- **Value for Developers**: Implement modern authentication in your applications without managing the complexity of FIDO2 protocols, device attestation, or certificate management.

Comprehensive Identity Services

The platform includes an integrated local identity store that is exposed through SCIM (System for Cross-domain Identity Management) standards and persisted in a database-backed service. Support for advanced federation protocols (SAML

2.0, OpenID Connect) enables bring-your-own-identity scenarios with just-in-time provisioning, allowing seamless integration with external identity providers while maintaining local control.

- **Value for Administrators**: Maintain a single authoritative identity source while federating with external providers, reducing identity sprawl and simplifying user lifecycle management.
- **Value for Developers**: Access identity data through standard REST APIs and SCIM endpoints, enabling rapid integration without proprietary protocols or vendor lock-in.

OAuth Resource Server and API Security

IDSP functions as the OAuth authorization server to support third party resource servers, enabling you to define custom scopes, application roles, and user consent workflows tailored to your applications' specific authorization requirements. You can generate access tokens with fine-grained permissions, implement user consent for sensitive operations, and control API access at the scope level.

- **Value for Administrators**: Centrally manage API authorization policies and monitor token usage across all applications from a single console.
- **Value for Developers**: Protect your APIs with industry-standard OAuth flows, define custom permission scopes for your services, and leverage built-in token validation without building an authorization infrastructure.

Personal Access Tokens and Zero Trust

To enforce zero trust and least-privilege access, users and developers can create Personal Access Tokens (PATs) with only the necessary privileges.

These tokens support secure, automated workflows and developer tool integration without the need to share privileged credentials. Access tokens have configurable lifecycles, up to one hour, and can be optionally paired with refresh tokens that last up to one week and are linked to specific clients.

- **Value for Administrators**: Eliminate the proliferation of service accounts and shared credentials while maintaining audit trails of who generated which tokens with what permissions.
- **Value for Developers**: Securely authenticate development tools, scripts, and CI/CD pipelines with scoped tokens instead of embedding passwords or long-lived credentials in code.

Adaptive Risk Management

The integrated risk service evaluates authentication attempts in real-time based on contextual factors including device posture, geolocation, user behavior patterns, and threat intelligence. It is possible to configure risk policies per application to apply appropriate security controls—for stricter evaluation for sensitive financial systems, streamlined access for low-risk internal tools. Support for Continuous Authentication Event Protocol (CAEP) enables coordinated responses across your security ecosystem when risk conditions change.

- **Value for Administrators**: Balance security and user experience by applying risk-based policies that challenge suspicious access while streamlining trusted scenarios.
- **Value for Developers**: Integrate risk signals into your applications through CAEP events, allowing applications to respond dynamically to security state changes without building risk engines.

Self-Service Console and End-User Experience

IDSP includes a fully customizable self-service console where end users manage their own credentials, enroll authentication factors, reset passwords through policy-driven recovery flows, view and launch authorized applications, and manage active sessions. The application dashboard provides a centralized launchpad for all authorized applications with single sign-on support.

- **Value for Administrators**: Dramatically reduces help desk ticket volume while improving security through user-managed strong authentication enrollment and white-label branding that matches corporate identity.
- **Value for Developers**: Provide end users with pre-built UI components for credential management and app discovery, or leverage the APIs to build custom self-service experiences integrated into your applications.

Granular Administration and Compliance

Role-based administration combined with resource authorization constraints enables separation of duties and least-privilege administration. Help desk staff can manage only their assigned user populations, application owners can configure only their integrations, and compliance requirements for segregated responsibilities are met through enforced permission boundaries and comprehensive audit logging.

- **Value for Administrators**: Meet regulatory compliance requirements with provable separation of duties, delegated administration, and complete audit trails of all administrative actions.
- **Value for Developers**: Programmatically manage administrative delegation through APIs, enabling self-service application onboarding and automated provisioning workflows.

Enterprise Scalability and High Availability

Built on container-based, cloud-native architecture, IDSP supports enterprise-scale deployments with multi-region configurations, active-active high availability, and simplified disaster recovery. Deploy across geographic regions to ensure that authentication services remain operational despite regional outages, network issues, or disaster scenarios. Built-in support for multi-tenancy provides separation for business units or external partners needing isolated, customizable identity environments.

- **Value for Administrators**: Meet SLA requirements for mission-critical authentication infrastructure with proven resilience patterns and operational simplicity.
- **Value for Developers**: Build applications with confidence that the authentication layer scales to support growth and remains available during infrastructure disruptions.

API-First Architecture and Developer Tools

Every capability in IDSP is exposed through comprehensive REST APIs, enabling full programmatic control of authentication, authorization, identity management, and administration. Dynamic client registration allows applications to self-register, SDK support accelerates integration, and extensive API documentation provides the guidance developers need.

- **Value for Administrators**: Automate operational tasks, integrate with IT service management tools, and enable self-service scenarios without manual intervention.
- **Value for Developers**: Integrate IDSP into applications, DevOps pipelines, and automation workflows with well-documented APIs, reducing time-to-production from months to days.

Legacy Integration and Modernization Bridge

IDSP provides native integration with Symantec SiteMinder and other Broadcom solutions, protecting existing technology investments while enabling gradual migration to modern patterns. Support for SAML, OIDC, and bring-your-own-identity scenarios means you can modernize incrementally without rip-and-replace disruption.

- **Value for Administrators**: Protect infrastructure investments while creating a clear path toward modern identity architectures aligned with business priorities.
- **Value for Developers**: Build new applications using modern protocols while maintaining compatibility with legacy systems during transition periods.

Architecture Highlights

Container-Based Deployment:

- Kubernetes-native architecture for cloud, on-premises, or hybrid environments
- Horizontal scaling to meet demand
- Rolling updates with zero-downtime maintenance

Standard-Based Integration:

- FIDO2/WebAuthn for passwordless authentication
- OAuth 2.0 and OpenID Connect for authorization and federation
- SAML 2.0 for legacy federation scenarios
- SCIM 2.0 for identity provisioning
- CAEP for continuous authentication events

Database-Backend Persistence:

- No LDAP dependencies—modern database architecture
- Support for enterprise database platforms - MySQL, Oracle, PostgreSQL

Use Cases

For Administrators:

- Replace aging authentication infrastructure with a modern, cloud-native platform
- Implement zero trust architectures with identity as the control plane
- Enable passwordless authentication across enterprise applications
- Reduce operational costs through self-service and automation
- Meet compliance requirements with granular access controls and audit trails

For Developers:

- Secure APIs with OAuth 2.0 authorization and custom scopes
- Integrate authentication into applications through REST APIs
- Build modern web and mobile applications with passkey support
- Automate development workflows with personal access tokens
- Create custom user experiences leveraging IDSP's identity services

Getting Started

This documentation provides comprehensive guidance for planning, deploying, configuring, and operating the Symantec Identity Security Platform:

- **Planning & Architecture**: Understand deployment models, capacity planning, and integration patterns
- **Installation & Configuration**: Step-by-step guides for deploying IDSP in your environment
- **Administrator Guide**: Complete reference for configuring policies, managing identities, and delegating administration
- **Developer Guide**: API documentation, SDK references, and integration tutorials
- **Operations & Maintenance**: Monitoring, backup, disaster recovery, and troubleshooting procedures
- **Release Notes**: Latest features, enhancements, and known issues

Administering

Contains information on Symantec Identity Security Platform Administration.

The administration function in IDSP is overseen by a group of administrators, comprising Infrastructure Administrators, System Administrators, and Tenant Administrators, among others. In the Administrative model, each administrator is assigned distinct responsibilities primarily centered around access management and restrictions. Tasks are executed via APIs and the Admin Console. Detailed roles and responsibilities are outlined in the Administrative Model documentation.

In the Resource Authorization model, you define policies, called scoping constraints, that can declare resource attribute permissions and resource instance rules. When combined with your Administrative model, Resource Authorization provides finer-grained administrator access across the granted Resource privileges.

This section includes the following topics:

- Administrative Model
- Resource Authorization
- Managing Authorization

Administrative Model

Covers the Roles and Responsibilities, and Operations performed by the Infra Administrators, System Administrators, and Business Administrators.

Overview

The Symantec Identity Security Platform business capability supports an administrative model that includes the following roles, each with specific responsibilities and access levels:

Client	Tasks	Notes
Infrastructure Administrator (Infra Admin)	• Bootstrap initial System Administrators	This is the "Super User" role. The single credential that is created for this role should be secured and made available only when needed.
System Administrator (System Admin)	• Configure operational parameters such as expiry settings, LDAP ID Stores, etc. • Assist Tenant Admins in case of issues such as lost account, accidental removal of privileges, etc.	System Admin can perform certain configuration operations on behalf of the Tenant Admin. Operations that are performed in the context of business tenants are subject to constraints imposed by system-level settings.
***Tenant Administrator** (Tenant Admin)	• Manage Security Policies • Manage tenant-level settings, such as CORS, etc. • Integrate with applications	Currently, this role also covers tasks that a ***Developer*** would perform to integrate IDSP into an application.

*** A Default Tenant Admin user is created with similar privileges.**

The OAuth2-based API security model aligns with these administrative roles through security privileges granted to application clients. Out-of-the-box authorization policies regulate access based on assigned roles.

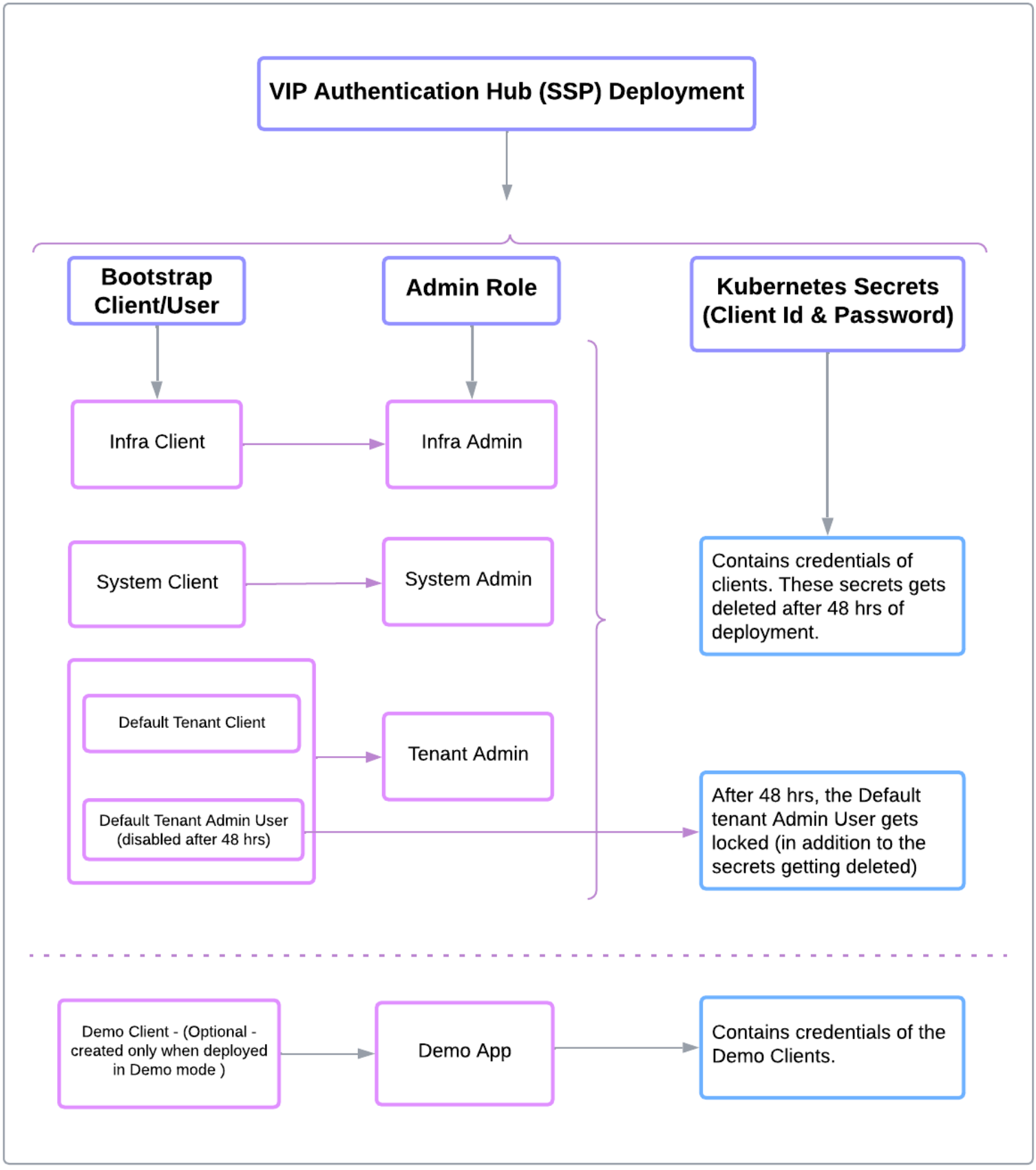
The bootstrap clients' credentials for the Infra, System and the Default Tenant clients are created by IDSP during the initial deployment and are made available to you through the Kubernetes secrets.

The admin client (defaultTenantClient) and admin user (defaultTenantAdminUser) are client and user credentials, respectively, for the "default" tenant bootstrapped by the initial deployment.

Apart from these bootstrap admin clients, IDSP provides various out-of-the-box admin roles, each with distinct permissions and restrictions. These roles are governed by Authorization Policies, with each role corresponding to a specific policy. Each admin role is assigned particular privileges (scopes) that enable the various administrator types to perform their regular tasks. See Admin Roles.

Initial Deployment of Clients and Roles

The following diagram illustrates the **IDSP Deployment clients and roles,** and highlights how various clients credentials are handled within the deployment.


VIP Authentication Hub (SSP) Deployment
Bootstrap Client/User
Admin Role
Kubernetes Secrets (Client Id & Password)
Infra Client
Infra Admin
System Client
System Admin
Contains credentials of clients. These secrets gets deleted after 48 hrs of deployment.
Default Tenant Client
Tenant Admin
Default Tenant Admin User (disabled after 48 hrs)
After 48 hrs, the Default tenant Admin User gets locked (in addition to the secrets getting deleted)
Demo Client - (Optional - created only when deployed in Demo mode)
Demo App
Contains credentials of the Demo Clients.

The breakdown of the components is explained here:

- **IDSP** - This is the IDSP solution, when deployed includes the Infra, System and Default Clients.
- **Clients/Users** - Three client types are deployed automatically with the IDSP solution.
  - **Infra Client**- Represents infra-level client. The credentials for this client is stored in Kubernetes secrets.
  - **System Client**- Represents system-level client. The credentials for this client is stored in Kubernetes secrets.
  - **Default Tenant Client**- Represents tenant-level client. The credentials for this client is stored in Kubernetes secrets. Default client created for tenant-level operations during IDSP deployment includes-
    - **Default Tenant Client** - Tenant client created for handling tenant-specific operations.
    - **Default Tenant Admin User** - An admin user for the default tenant, created during deployment.

      NOTE
      This user is automatically disabled after 48 hours for security reasons. Credentials for this user are stored in Kubernetes secrets, which will be deleted after 48 hours.
  - **(Optional) Demo Client** - Created for testing purposes only when the demo mode is enabled during the IDSP deployment. The credentials for this client is stored in Kubernetes secrets.
- **Admin Roles**- This has already been explained in the Overview section.
- **Kubernetes Secrets** - These credentials are stored as Kubernetes secrets and are available for 48 hours after deployment. After this period, due to security concerns with exposing sensitive credentials for a prolonged time, these credentials are automatically removed. To prevent loss of access, administrators must securely store these credentials immediately upon retrieval.

To know how to retrieve the bootstrap client credentials, see Retrieving Initial Credentials for the Bootstrap clients .

To see few examples of how these roles work in IDSP, see Roles and Responsibilities of Bootstrap clients .

Reference Links

- Managing Authorization
- Configuring Administrative Access

Roles and Responsibilites of Bootstrap Clients

The following content shows few examples of the responsibilities shared by the Bootstrap clients (Infra, System and Tenant Administrators).

This topic includes the following sections:

- Infrastructure Administrator
- System Administrator
  - Obtaining Access Token for System Admin Tasks
  - Example 1: Managing System Settings
  - Example 2: Assist Tenant Admin to fix a tenant specific problem
- Tenant Administrator

Infrastructure Administrator

As an Infra Admin, you will be provided with a single "Infra Client" credential immediately after the initial deployment, to assist you with configuring the first User System Administrator by defining the LDAP Directory and granting appropriate users or groups to the System Admin role. Also, you may choose to create a client having the System Admin role instead of, or in addition to, designating users to be System Admins.

For this release, we automatically seed the System Client having the System Admin role, so that you do not have to perform any specific steps. However, you may be required to use the InfraClient in case of issues with the SystemClient, such as:

- when the secret for the system client has been lost and must be reset
- authorization policy has been mistakenly updated to lock out administrators, and must be reset

In general, this credential is the "super user" credential and you should store it in a secure location.

System Administrator

As a System Admin, you may need to perform certain system admin tasks post installation to configure the environment and meet administration and/or security/governance needs. For example, you may want to configure maximum expiration interval for tokens, or delegate the right to perform system administration to some admin users, etc. These tasks are performed by Users or Applications having the System Admin role. For convenience, you can use the"SystemClient" credential seeded out of the box to perform such tasks.

Obtaining Access Token for System Admin Tasks

Obtaining the access token is the first step towards performing the tasks that are related to your role. The following example shows how to get the access token.

1. Use the following cURL command to get the access token(default tenant) using system admin credentials provided out of the box.
   Request:

```
curl -L -X POST 'https://iam.acme.com/system/oauth2/v1/token' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-u '<systemClientID>:<systemClientSecret>'
--data-urlencode 'grant_type=client_credentials' \
--data-urlencode 'scope=urn:iam:myscopes'
```

   NOTE
   In the example, replace ***<iam.acme.com>*** with your deployment's ingress host. You can retrieve it by using this command:
   `kubectl get ingress -n <namespace>`

   Response: The following is the response that gives you the access token. Use this access token value in the next examples to perform System Admin tasks.

```
{
"access_token": "<access token value>",
"token_type": "Bearer",
"expires_in": 360
}
```

Example 1: Managing System Settings

As a System Admin, you may want to update a setting in the System Configuration. For example, you may want to update the maximum lifetime for the issued access token. As a security consideration, the lifetime of an access token should be kept short, and your organization may have specific policies in place regarding this.

Follow these steps:

1. To get all system settings, execute the following request from POSTMAN (or CURL):
   NOTE
   In the example, replace <iam.acme.com> with your deployment's ingress host and enter the access token that you have obtained in the previous step.
   Request:

```
curl -L -X GET 'https://iam.acme.com/system/admin/v1/Settings' \
-H 'Content-Type: application/json' \
```

```
-H 'Authorization: Bearer <token>' \
--data-raw '
```

Response:


```
[
  {
      "name": "persistEnrolledFactorsinIDP",
      "value": "false",
      "origin": "global"
  },
  {
      "name": "azCodeExpirySecs",
      "value": "120",
      "origin": "global"
  },
  {
      "name": "accessTokenExpiryMaxMins",
      "value": "60",
      "origin": "global"
  },
  {
      "name": "isInlineOTPEnrollmentToBePersisted",
      "value": "true",
      "origin": "global"
  },
  {
      "name": "refreshTokenExpiryDays",
      "value": "7",
      "origin": "global"
  },
  {
      "name": "isInlineEnrollmentAllowed",
      "value": "true",
      "origin": "global"
  },
  {
      "name": "idTokenExpirySecs",
      "value": "21600",
      "origin": "global"
  },
  {
      "name": "pushAuthTimeoutSecs",
      "value": "300",
      "origin": "global"
  },
  {
      "name": "accessTokenExpirySecs",
      "value": "3600",
      "origin": "global"
  },
  {
      "name": "idTokenExpiryMaxMins",
      "value": "720",
      "origin": "global"
```

```
  },
  {
      "name": "useOriginalIdentityTokenExpiry",
      "value": "false",
      "origin": "global"
  },
  {
      "name": "inlineEnrollmentFactorsRequired",
      "value": "PASSWORD",
      "origin": "global"
  },
  {
      "name": "riskProviderToBeUsed",
      "value": "SSP",
      "origin": "global"
  },
  {
      "name": "logLevel",
      "value": "INFO",
      "origin": "global"
  },
  {
      "name": "pushRegTimeoutSecs",
      "value": "300",
      "origin": "global"
  },
  {
      "name": "signAlgo",
      "value": "RS256",
      "origin": "global"
  },
  {
      "name": "logRequestPayloads",
      "value": "true",
      "origin": "global"
  },
  {
      "name": "cacheRefreshIntervalMins",
      "value": "1",
      "origin": "global"
  }
]
```

2. To update the access token expiry system setting, execute the following PATCH request from POSTMAN (or CURL):

   NOTE
   In the example, replace <iam.acme.com> with your deployment's ingress host and enter the access token that you have obtained in the previous step.

Request:

```
curl -L -X PATCH 'https://iam.acme.com/system/admin/v1/Settings' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <token>' \
--data-raw '[
  {
```

```
        "name": "accessTokenExpirySecs",
        "value": "360"
    }
]
```

Response:

```
[
    {
        "name": "persistEnrolledFactorsinIDP",
        "value": "false",
        "origin": "global"
    },
    {
        "name": "azCodeExpirySecs",
        "value": "120",
        "origin": "global"
    },
    {
        "name": "accessTokenExpiryMaxMins",
        "value": "60",
        "origin": "global"
    },
    {
        "name": "isInlineOTPEnrollmentToBePersisted",
        "value": "true",
        "origin": "global"
    },
    {
        "name": "refreshTokenExpiryDays",
        "value": "7",
        "origin": "global"
    },
    {
        "name": "isInlineEnrollmentAllowed",
        "value": "true",
        "origin": "global"
    },
    {
        "name": "idTokenExpirySecs",
        "value": "21600",
        "origin": "global"
    },
    {
        "name": "pushAuthTimeoutSecs",
        "value": "300",
        "origin": "global"
    },
    {
        "name": "accessTokenExpirySecs",
        "value": "360",
        "origin": "global"
    },
    {
        "name": "idTokenExpiryMaxMins",
```

```
        "value": "720",
        "origin": "global"
    },
    {
        "name": "useOriginalIdentityTokenExpiry",
        "value": "false",
        "origin": "global"
    },
    {
        "name": "inlineEnrollmentFactorsRequired",
        "value": "PASSWORD",
        "origin": "global"
    },
    {
        "name": "riskProviderToBeUsed",
        "value": "SSP",
        "origin": "global"
    },
    {
        "name": "logLevel",
        "value": "INFO",
        "origin": "global"
    },
    {
        "name": "pushRegTimeoutSecs",
        "value": "300",
        "origin": "global"
    },
    {
        "name": "signAlgo",
        "value": "RS256",
        "origin": "global"
    },
    {
        "name": "logRequestPayloads",
        "value": "true",
        "origin": "global"
    },
    {
        "name": "cacheRefreshIntervalMins",
        "value": "1",
        "origin": "global"
    }
        ]
```


Example 2: Assist Tenant Admin to fix a tenant specific problem

There may be a situation where tenant admins accidentally lock themselves out, for example, by mistakenly updating the Admin Role policy to exclude their own identity, or by mistakenly updating the LDAP Config, hence disabling connectivity to that LDAP Directory. To fix such issues, you as a System Admin have the right to execute cross-tenant operations (APIs) to fix the problem.

Executing cross-tenant operations requires the request for Access Token to be performed in the context of the business tenant while specifying the System Client's tenancy as the request header.

You must perform a GET operation on the tenant's LDAP Config:

Get a cross-tenant access token (for the "default" tenant) using system admin credentials of the client in the "system" tenant:

> **NOTE**
> In the example, replace <iam.acme.com> with your deployment's ingress host.

Request:

```
curl -L -X POST 'https://iam.acme.com/default/oauth2/v1/token' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H 'X-CLIENT-TENANT-NAME: system' \
-u '<systemClientID>:<systemClientSecret>'
--data-urlencode 'grant_type=client_credentials' \
--data-urlencode 'scope=urn:iam:myscopes'
```

Response:

```
{
"access_token": "<access token value>",
"token_type": "Bearer",
"expires_in": 360
}
```

Get Tenant LDAP config using cross-tenant access token:

> **NOTE**
> In the example, replace <iam.acme.com> with your deployment's ingress host and enter the access token that you have obtained in the previous step.

Request:

```
curl -L -X GET 'https://iam.acme.com/default/admin/v1/LDAPconfigs' \
-H 'Authorization: Bearer <token>'
```

Response:

```
[
  {
      "configName": "TestConfig",
      "configDescription": "CA DIR k8s with default data",
      "sslEnabled": "NO",
      "certificate": null,
      "certificateAlias": null,
      "ldapQueryStart": "(&(cn=",
      "ldapQueryEnd": ")(objectClass=*))",
      "attributes": "uid,telephoneNumber,employeeType,mail,displayName,givenName,cn,sn,memberof,title",
      "objectClasses": "string",
      "hostName": "example-216-ssp.sspdev.dev.broadcom.com",
      "portNumber": "30306",
      "baseDN": "o=democorp,c=us",
      "userDN": "cn=DominicMAJOR,ou=Administration,ou=Corporate,o=democorp,c=US",
      "bindPassword": "/OTqCj2yPWVvhl6AjpJ/DQ==",
      "attributeMapping": "name=cn,family_name=sn,given_name=givenName,
            middle_name=givenName,nickname=displayName,
            preferred_username=displayName,profile=title,email=mail,
            phone_number=telephoneNumber,user_universalid=cn,firstName=cn,
      status=employeeType,memberof=memberof",
```

```
        "passwordResetURL": "http://userPasswordResetURL",
        "createdDateTime": "2020-06-11 09:43 AM UTC",
        "updatedDateTime": "2020-06-11 09:43 AM UTC",
        "createdBy": {
            "principalType": "CLIENT",
            "principalId": "97509c25-6546-46c4-9e4d-12fbbca60f3a",
            "principalName": "defaulttenantclient"
        },
        "ldapConfigId": "c6070959-9117-4a9b-8591-62030aa76e12"
    }
]
```


Tenant Administrator

As a Tenant administrator, your tasks cover what is necessary to integrate applications with security services provided by Symantec Identity Security Platform. For example:

- Creating an "application" entity representing protected applications
- Creating and managing Authentication Policies defining the level of authentication required by the application
- Managing configuration settings such as certificates, message templates, available authentication factors, etc.
- Creating and Managing a "Identity Stores" entity representing the identity store with users to be authenticated by IDSP
- Delegating Tenant Admin role to users and groups. Also to applications that wish to invoke IDSP APIs

Resource Authorization

Describes how Resource Authorization provides fine-grained administrator access by using scoping constraints.

The default Administrative model provides course-grained administrator access by using granted Administrative roles and their associated privileges. In this default model, Administrators can see all resource instances and all of the resource instance attributes. The Resource Authorization model lets you define policies, called scoping constraints, that can declare resource attribute permissions and resource instance rules. When combined with your Administrative model, Resource Authorization provides finer-grained administrator access across the granted Resource privileges.

Resource authorization allows you to control the level of access an entity (a user, group, app, or instance), has to a particular resource in the system. That is, resource authorization allows you to set the following for any entity:

- What can the entity see and manage?
- What can the entity grant or revoke?
- What does the entity have granted to it?

When you create your policies, you naturally create separate resources, such as when you create policies for SAML apps and policies for OIDC apps. However, defining all of the possible separate resources manually through policies is prohibitive. Instead, you can use resource authorization to segregate privileges more granularly, and with more reusabilities.

The Resource Allocation APIs support three access control levels:

- **Resource Privilege Scope (RPS):** RPS controls privilege-level access to the resource, which is secured by an OAuth2 scope, and delegated by virtue of assigning an Admin Role. RPS constitutes the default course-grained access model.

  For example: urn:iam:t.apps, urn:iam:t.session, urn:iam:t.users

  See Managing Authorization for details about defining administrator roles.

NOTE
You can use wildcards in the resource attribute.

- **Resource Data Scope (RDS):** RDS is a control that defines data level access for User attributes such as the user ID, given name, or email address. Resource Data Scopes are defined as data elements with specific permissions in a ScopingConstraint policy.
  You can extend Resource data by adding additional tags, which can be used for Resource Authorization criteria in addition to other tagging purposes. For example, if you tag a resource with a scope, you can include that scope in a constraint statement for any policy that affects that resource. See Scoping Constraints Overview and Managing Scoping Constraints for more detail on adding a scoping constraint.
- **Resource Instance Scope (RIS):** RIS is a control that defines instance-level access to a resource such as a User, Group, App, Identity Provider, Secret, Certificate, and so on. The RIS is enforced by policy enforcement points that control access to a resource (typically a resource controller). An RIS is defined as instance elements containing admin scope rules (controls) in a ScopingConstraint policy.
  You add criteria (attribute tags) directly into your API schema to define the scope attribute for each resource. Once a resource is tagged with a criteria, you can include that criteria in a constraint statement for any policy that affects that resource. See Scoping Constraints Overview and Managing Scoping Constraints for more detail on adding a scoping constraint.

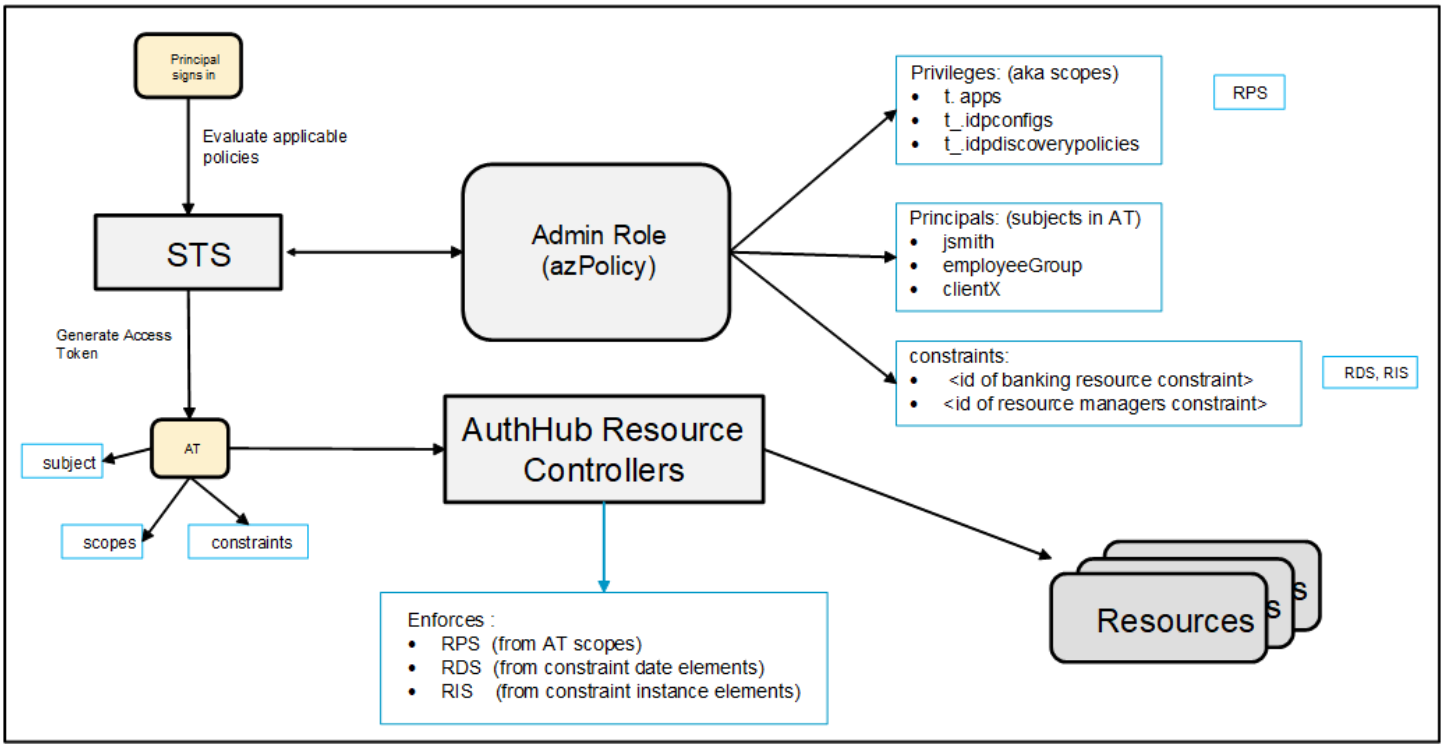
When an entity attempts to perform an operation against a resource, the requesting API checks the policy for the following:

- Conditions (when the policy applies)
- Principals (to whom the policy applies)
- RPS or scopes (what privileges the administrator is assigned)
- Constraints based on RDS and RIS (to which attributes does the administrator can access)

The combinations of these are included in the Access Token that defines what the administrator has access to and can manage.

The following image illustrates the resource access flow with Resource Authorization constraints in place.

Figure 1: Resource Access Flow



Next:

Defining a Resource Authorization Model

Defining a Resource Authorization Model

Describes how to define your Resource Authorization model.

The Resource Authorization model for a deployment can vary depending upon the several factors, such as:

- The balance between functional administration and business need (for example, whether it should be centralized, decentralized, or a combination of both)
- The state of the resource data and data consistency (for example, whether the deployment has common schemas and consistent data values)

Resource Authorization provides flexible capabilities to handle various authorization models. The two main approaches are typically devised based on either a centralized or a decentralized model.

Centralized Administration

A centralized administration model is defined by top-down alignment of resources based on some business-driven partitioning, such as by organization, business or product line, department, and so on. Administrators and resources are typically provisioned along these lines. This partitioning allows you to construct policies based on data attribute values that are directly aligned to your business needs.

Administrative privileges are typically centralized within the business partition. For example, a banking admin can administer banking resources. Resource management tends to maintain the centralized model. In this model, Administrators can only create and manage resource within their partition. However, these rigid centralized tendencies can sometimes be relaxed, such as allowing an Administrative scope to cross partitions.

The following figures illustrate the centralized administration model for two different organizations.

Figure 2: Centralized Banking Organization Administration Model

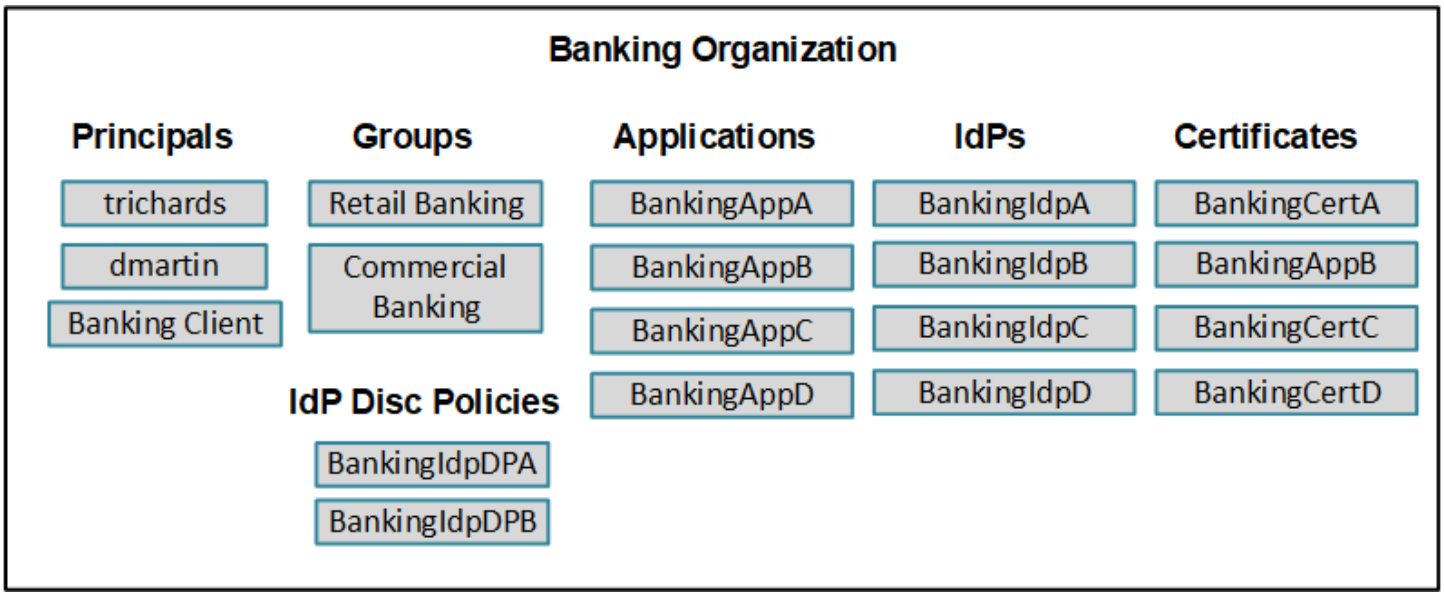
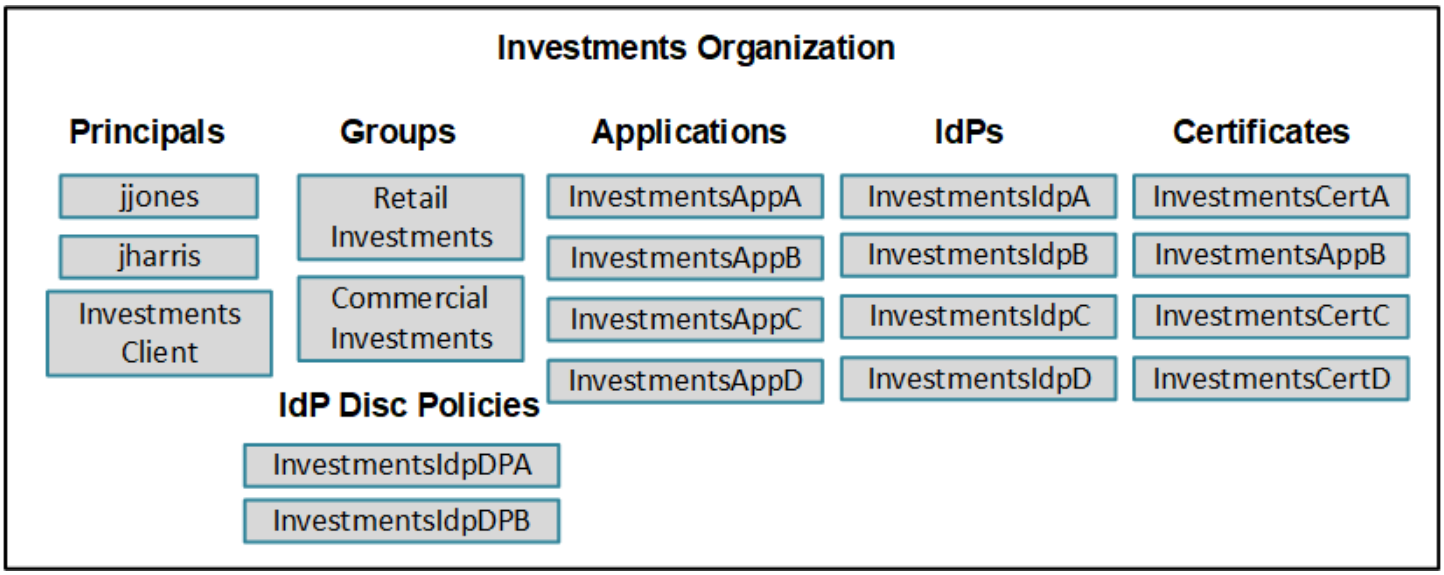


Figure 3: Centralized Investments Organization Administration Model



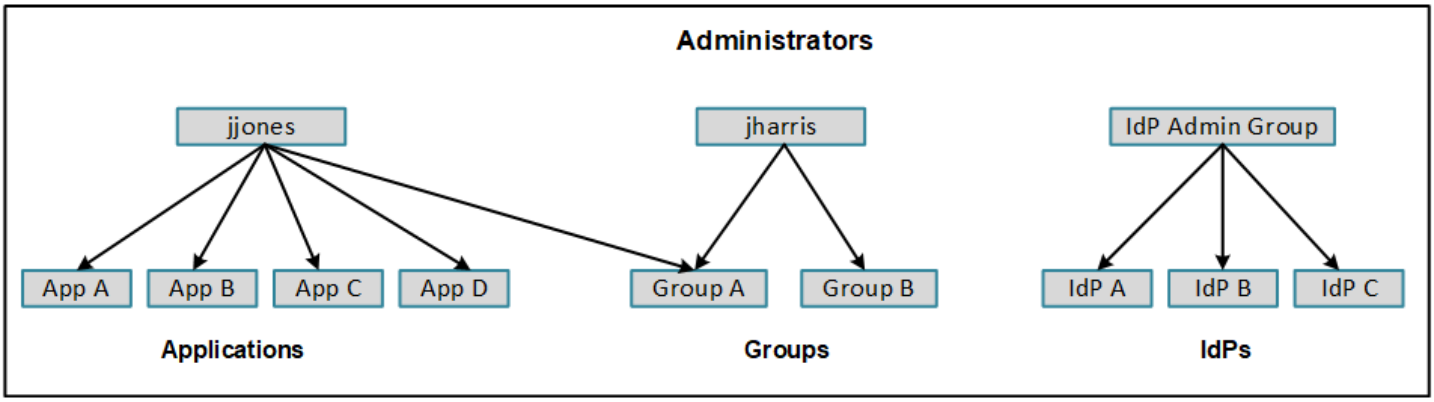
Decentralized Administration

A decentralized administration model has no unifying, top-level, business-defined structure. Administration is spread out based on other functional administrative needs. The decentralized model allows different areas of the business to define administrative needs differently. Decentralization relies on a named manager model where access is granted to specific resource instances for a user, group, or client.

The decentralized administration model requires a higher level of manual administration. However, using groups as the basis of access can alleviate this administrative burden.

The following figure illustrates the decentralized administration model.

Figure 4: Decentralized Administration Model



Resource Authorization Scoping Constraints

Describes the access control mechanisms available in the Resource Authorization model.

The following table describes the access control mechanisms that are supported by the Resource Authorization model:

Table 1: Resource Authorization access control mechanisms

Access	Resources	Usage	Notes
User attributes (ABAC)	User	User - any User Schema attribute	It is highly recommended to use **User.organization** or User custom attributes since these attributes are indexed in the Internal Id Store.
adminscope.criteria (ABAC)	ABAC for Group, Application, IDP, IDP Discovery, AuthN/ AuthZ Policy, Secrets, Certificates	**adminscope.criteria** is a schema extension that is added for tagging non-user Resources.	**adminscope.criteria** is an attribute that can be used as an ABAC constraint in a Scoping Constraint policy or as a general resource tagging mechanism. The attribute takes the form of: `name=value` Multiple name=value pairs can be defined.
adminscope.managedby	All	Named manager of a resource. Supports several bindings: • Non-user Resources: **adminscope.managedby** • User resources: **User.managerIdUser.Groups**	**adminscope.managedby** relies on delegation of access to resources for a subject (user, client, group) using the Subdelegation APIs. • **User.managerId** is the SCIM supported manager/ user relationship • **User.groups** allows an Admin who belongs to a group to manage user members in that same group.

Access	Resources	Usage	Notes
Owner	All	The default constraint when none of the previous access control mechanisms apply.	

Building A Resource Model

Describes how to build a Resource Authorization model using Scoping constraints.

Resource Authorization is based on enforcement of applicable constraints (or *Admin Scopes*) as defined in the Scoping Constraints associated with an Administrative Role. While Administrative Roles define resource privilege scope (RPS), any associated Scoping Constraints can restrict those privileges by defined resource instance scope (RIS) rules, or by resource data scope (RDS) rules.

If no Scoping Constraints are added to Administrative Roles, or a Scoping Constraint has been added but is marked inactive, then no restrictions are applied to the privileges of those Administrative Roles.

> **NOTE**
> RDS applies only to User resources.

You manage (create, read, update, and delete) Scoping Constraints using the Scoping Constraints API. Once a Scoping Constraint is defined, it can be added/removed from an Administrative Role using the Scoping Constraints API or the **Administrative Role > Constraints** page of the Admin Console.

- For information on using the Scoping Constraints API to manage Scoping Constraints, see Scoping Constraints API.
- For information on using the Admin Console to add or remove constraints, see Administrators and Roles

For detailed procedures on building and managing a Resource Authorization Model, see the following topics:

- Scoping Constraints Overview
- Supported Resource Data Scope (Attribute Permissions)
- Supported Resource Instance Scope (RIS)
- Adding or Removing a Constraint
- Enabling Resource Admin Scope Enforcement for Resource Authorization
- Understanding and Using the Default Constraints
- Determining Applicable RDS and RIS for a Given Principal Using Helper APIs

Resource Authorization results for Symantec Identity Security Platform Resource APIs (such as User, Group, applications, and so on) can often return complex outcomes. For information on using the `explain` parameter to understand Resource Authorization results, see Understanding Resource Authorization Results using Explain.

Scoping Constraints Overview

Overview

A Scoping Constraint is a policy that defines access controls for resources. A Scoping Constraint, which are associated with specific resource types, defines resource data access and resource instance access. These access controls are collectively referred to as scope or admin scope.

Resource Authorization is based on the enforcement of applicable admin scope as defined by the Scoping Constraints associated with an Administrative Role. While Administrative Roles define resource privilege scope (RPS), any associated Scoping Constraints can restrict those privileges by defined resource instance scope (RIS) rules, or by resource data scope (RDS) rules.

NOTE
Note RDS is restricted to User resources only.

Scoping Constraints are created, read, updated, and deleted via the Scoping Constraints API. Once a Scoping Constraint is defined, it can be added/removed from an Administrative Role using the Scoping Constraints API or the Admin Console's Administrative Role's Constraints page.

- For information on the using the Scoping Constraints API to manage Scoping Constraints, see Scoping Constraints API.
- For information on using the Admin Console to add or remove constraints, see Administrators and Roles

Scoping Constraints Definition

The following table lists the components of a Scoping Constraint.

Table 2: Scoping Definition

Scoping Constraint fields	Description	Values
name	A unique name for the constraint.	Varies
description (optional)	A description of the constraint.	Varies
resource	The associated resource type	Must be one of the following: • urn:iam:res:app • urn:iam:res:authnpolicy • urn:iam:res:authzpolicy • urn:iam:res:cert • urn:iam:res:group • urn:iam:res:idp • urn:iam:res:idppolicy • urn:iam:res:secret • urn:iam:res:user • urn:iam:res:* - all resources
status	The state of the constraint (active or inactive). Inactive constraints are ignored during enforcement.	• active • inactive

Scoping Constraint fields	Description	Values
abacOps (optional)	A control that defines the operations that are enforced when an ABAC constraint applies. You can add one of the following qualifiers to this control: • **+** This prefix allows the operation without enforcing the ABAC constraint. • **-** This prefix denies the control for the ABAC constraint. Providing no prefix enforces the ABAC constraint for the operation. If any scoping constraint includes the **+** prefix, it takes precedence over other constraints. If + is not provided in all scoping constraints for the resource type and one scoping constraint includes the - prefix in abacOps, the - prefix takes precedence over other constraints. If a qualifier is not explicitly added, it is treated as unconditionally denied. If the resource type has multiple scopes, the priority order is: unconditional access, unconditional deny, and conditional access.	• C - enforce constraints on create operations • R - enforce constraints on read operations • U - enforce constraints on update operations • D - enforce constraints on delete operations
gbacOps (optional)	A control that defines the operations that are enforced when a GBAC constraint applies. gbacOps is associated with the managedBy constraint conditions. Since managedBy is not applicable during resource create operations, 'C' is not enforced. You can add one of the following prefixes to this scoping constraint: • **+** This prefix allows the operation without enforcing the GBAC constraint. • **-** This prefix denies the control for the GBAC constraint. Providing no prefix enforces the GBAC constraint for the operation. If any scoping constraint includes the **+** prefix, it takes precedence over other constraints. If + is not provided in all scoping constraints for the resource type and one scoping constraint includes the - prefix in gbacOps, the - prefix takes precedence over other constraints. If a qualifier is not explicitly added, it is treated as unconditionally denied. If the resource type has multiple scopes, the priority order is: unconditional access, unconditional deny, and conditional access.	• C - enforce constraints on create operations • R - enforce constraints on read operations • U - enforce constraints on update operations • D - enforce constraints on delete operations

Scoping Constraint fields	Description	Values
denyOps (optional)	A control used to explicitly deny an operation (matching is case-insensitive). The default value is null. If any scoping constraint for a resource type has this control, the operation is denied for that resource type. For example, if a resource type has denyOps=U, the principal cannot update any resources of this type.	• C - enforce constraints on create operations • R - enforce constraints on read operations • U - enforce constraints on update operations • D - enforce constraints on delete operations
adminScopeApi (optional)	A control override that, when set to **true**, enforces RPS and constraint RDS rules, but ignores constraint RIS rules	• true • false
data	For RDS only: A list of attributes and their associated permissions.	See specific details in Supported Resource Data Scope (Attribute Permissions). If the data list is empty, then no attribute permissions are defined and all attributes are considered RW.
instance	For RIS only: a list of access rules.	See specific details in Supported Resource Instance Scope (RIS). If the instance list is empty, no instance rules are applied for this policy.
disableOwner	If disableOwner = true, whether the current principal is the owner of a resource is not considered when enforcing the scoping constraint.	• true • false (default)

Supported Resource Data Scope (Attribute Permissions)

Describes how to use Resource Data Scope (RDS), or *attribute permissions*.

The data element of a Scoping Constraint defines the permissions that are allowed on User attributes. Note that data is allowed on User resources only. The following example illustrates the format of the data section attribute permissions:

```
"data":  [
         {
            "attr": "attribute_name | * " ,
            "priv": "<permission>"
            },
]
```

If the data list is empty or not present in the constraint, then the attribute access is *unconstrained*.

The following table describes the resource attribute permissions:

Table 3: Resource Attribute Permissions

Permission	Description
r	Administrator can read/view the attribute. This setting is the default unless explicitly declared otherwise.
w	Administrator can write/update the attribute.
rw	Administrator can read/view and write/update the attribute.
-	The administrator has no permission (cannot read/view or write/update the attribute).

The POST and PUT APIs take a JSON payload as in the following example:

```
{
        "name": "<the unique constraint name>",
        "description": "<some description>",
        "status":  "active",
        "resource": "urn:iam:res:user",
            "data": [
                    "attr": "given_name",
                    "priv": "r"
                },
                {
                    "attr": "is_locked",
                    "priv": "w"
                },
                {
                    "attr": "family_name",
                    "priv": "r"
                },
                {
                    "attr": "email",
                    "priv": "rw"
                }

                    ]
```

The `attr` element must contain an attribute name conforming to the logical attribute set. These attributes can be returned from the `UserSchemaHelper` API. The name attribute is the value of the attribute name that is defined in a Scoping Constraint. The following example shows the birthdate attribute output from the schema helper:

```
"birthdate": {
        "account": false,
        "searchable": false,
        "visible": true,
        "operational": false,
        "mva": false,
        "name": "birthDate",
        "type": "DATE",
        "visibleToDisambiguationQuery": false,
        "scimUri": "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User:birthDate",
        "scimAttributeName": "birthDate",
        "scimWritable": true,
        "values": null,
        "custom": false
```

Attribute Permissions

By default, the attribute permissions of a resource are assumed to be unconstrained. An Administrator applying a Scoping Constraint can use the following semantics:

1. If a set of attribute permissions are explicitly defined in a Scoping Constraint, then those defined permissions are enforced.
2. If an attribute permission is not explicitly declared in a Scoping Constraint, then the default permission is ‘r’.

3. Within a given Scoping Constraint, you can use wildcards for the attribute name to define the default permission, with other declared permissions overriding the wildcard default.

In the following Scoping Constraint definition example, an HR Administrator needs ‘rw’ permissions on HR-based attributes, and all other attributes default to ‘r':

```
{
        "name": "HR Support",
        "description": "HR Support needs RW on specific HR attrs",
        "status":  "active",
        "resource": "urn:iam:res:user",
                "data": [
                        {
                        "attr": "given_name",
                        "priv": "rw"
                        },
                        {
                        "attr": "family_name",
                        "priv": "rw"
                        },
                        {
                        "attr": "email",
                        "priv": "rw"
                        },
                        {
                        "attr": "phone_number",
                        "priv": "rw"
                        },
                        {
                        "attr": "address",
                        "priv": "rw"
                        },
                        {
                        "attr": "title",
                        "priv": "rw"
                        },
                        {
                        "attr": "organization",
                        "priv": "rw"
                        }
                        // all other attributes not listed explicitly, have ‘r’ permission
                ]
        }
```

As another example, an Application Owner needs ‘rw’ permissions on a small set of application-based attributes, and ‘r’ permission on others to help identify the User. The rest of the attributes should not be visible. The following Scoping Constraint definition example supports this use case:

```
{
        "name": "App Support",
        "description": "Applications support ", "resource": "urn:iam:res:user",
                "data": [
                        {
                        "attr": "*",    //Declare No permission by default
```

```
                    "priv": "-"
                    },
                    {
                    "attr": "AcmeAppLoginId",    //Declare RW on App attrs
                    "priv": "rw"
                    },
                    {
                    "attr": "AcmeAppSegment",
                    "priv": "rw"
                    },
                    {
                    "attr": "WidgetAppProjects",
                    "priv": "rw"
                    },
                    {
                    "attr": "WidgetAppDefaultProject",
                    "priv": "rw"
                    },
                    {
                    "attr": "user_loginId",    //Declare R on user identifying attrs
                    "priv": "r"
                    },
                   "attr": "given_name",
                    "priv": "r"
                    },
                    {
                    "attr": "family_name",
                    "priv": "r"
                    },
                    {
                    "attr": "email",
                    "priv": "r"
                    },
                    {
                    "attr": "title",
                    "priv": "r"
                    },
                    {
                    "attr": "organization",
                    "priv": "r"
                    }
            ]
    }
```

Supported Resource Instance Scope (RIS)

Describes how to use Resource Instance Scope (RIS).

The instance section defines scope rules that are based on resource attributes, and takes the following form:

```
 attr": "<some attribute name>",
 "scope": [
           {
```

```
            "op": "eq | sw",
            "value": "<some attribute value>"
        }
    ]
```

Attributes that are defined in the instance element are subject to frequent query operations. For user resources, this attribute can be any attribute that is defined in the User schema. However, it is highly recommended to choose attributes that are marked as `searchable=true` . Searchable attributes have high-performance indexing enabled. See the UserSchemaHelper API or the Internal Identity Store Configuration configuration for the Admin Console for a list of searchable User attributes.

The adminscope.criteria attribute that is defined for non-user resources is searchable and indexed.

Comparison Operators

The following table shows the comparison operators that are supported by instance attribute scope rules.

Table 4: Supported comparison operators, for instance, attribute scope rules

Operator	Comparison	Supported Data Types	Examples
eq	equals	string, integer, boolean date	• attr: organization • op: eq • value: retail
sw	starts with	string	• attr: organization • op: sw • value: retail • matches: retail • banking, retailInvestments Special match semantics, comparison matches value portion: • attr. adminscope.criteria • op: sw • value: organization=retail Matches: • organization=retailBanking • organization=retailInvestments

Common Resource Access Controls

You can use the following common attributes:

- **adminscope.criteria**

Table 5: adminscope.criteria attribute

Attribute	Description	User Binding

adminscope.criteria	Sets of {name=value} pairs. For example:
	
	`"attr": "adminscope.criteria",`
	`"scope": [`
	`{`
	`"op": "eq",`
	`"value": "enrollment=platinum"`
	`}`
	`]`
		An optional mapping that resolves to a User-based attribute. For example:
		`"attr": "adminscope.criteria",`
		`"scope": [`
		`{`
		`"op": "eq",`
		`"value": "enrollment=platinum"`
		`"userAttr": "enrolledIn"`
		`}`
		`]`
		In this example, users with an **enrolledIn** attribute value that is equal to *platinum* match the constraint condition.

- **adminscope.managedby**

Table 6: adminscope.managedby attribute

Attribute	Description	User Binding

adminscope.managedby	A definition of a manager with specialized values, used to resolve various resource manager models. For example:
	
	`"attr": "adminscope.managedby",`
	`"scope": [`
	`{`
	`"op": "eq",`
	`"value":`
	`"urn:iam:adminscope:me",`
	`"userAttr": "managerId"`
	`},`
	`{`
	`"op": "eq",`
	`"value":`
	`"urn:iam:adminscope:megroups",`
	`"userAttr": "groups"`
	`}`
	`]`
		• value: "urn:iam:adminscope:me”
		A declaration that resolves to manager records (of type **user/client**) that match the ID of the current principal.
		• value: "urn:iam:adminscope:megroups”
		A declaration that resolves to manager records (of type **group**) that match one of the current principal groups where that principal is a member.
		• userAttr: managerId
		Resolves to users whose **managerid** value matches the Id of the current principal
		• userAttr: groups
		Resolves to groups that match one of the **User.groups** values of the current principal.

Evaluating Multiple Scopes in RIS

In constraint definitions, if the same attribute is defined with different values, these values are ORed during evaluation. For example, given the following constraint, the evaluation matches resources who contain either **silver** or **platinum** values for the enrollment criteria.

```
"attr": "adminscope.criteria",
 "scope": [
   {
```

```
     "op": "eq",
     "value": "enrollment=platinum"
    },
 {
     "op": "eq",
     "value": "enrollment=silver"
    },
   ]
```

If two different attributes are defined in a constraint, then the two attribute and their values are evaluated using an AND operation. For example, given the following constraint, the evaluation matches resources who contain **enrollment=platinum** and **organization=banking**.

```
 "instance": [
          {
              "attr": "enrollment",
              "scope": [
                 {
                    "op" : "eq",
                    "value" : "platinum"
                 }
              ]
          },
          {
              "attr": "organization",
              "scope": [
                 {
                    "op" : "eq",
                    "value" : "banking"
                 }
              ]
          }
         ]
```

Understanding and Using the Default Constraints

When deploying a new tenant or upgrading an existing tenant to version 3.3 or later, the system deploys default constraints. These constraints are deployed in an inactive state and are added to out-of-the-box Administrative Roles that contain privileges for resources that can be constrained using Resource Authorization. To enable the use of the constraint, the status must be changed from **inactive** to **active** using the Scoping Constraints API.

The default constraints cover the full set of supported constraint styles. The deployment of the default constraints and the roles they have been added to are shown in the following table.

Table 7: Default constraints and their roles

Constraint Name	Resource	Administrative Roles	See
Default Constraint	urn:iam:res.*	• IdentityClientPolicy • IntegrationAdminPolicy • IntegrationAuditorPolicy • ConfigurationAdminPolicy • ConfigurationAuditorPolicy • IdentityAdminPolicy	Default Constraint
Default Me Constraint	urn:iam:res.me	Self Administration	Default ME Constraint
Unrestricted Constraint	urn:iam:res.*	Deployment specific. Only used during 3.3 upgrades, and deployed when out-of-the-box Administrative Roles have been granted to principals.	Unrestricted Constraint

Default Constraint

Describes the Default Constraint added to Administrative Roles with new tenants or when upgrading.

The default constraint supports the full spectrum of Authorization styles for centralized and decentralized models. This topic describes the specifics of each constraint style.

The adminscope.criteria Constraint

The following adminscope.criteria definition assumes that partitioning of resources for ABAC is done by organization. This constraint assumes that there is some organization value that is provisioned for resources which matches the organization of principals requesting access to those resources. This concept is a core concept of the centralized administration model that is achieved by attribution.

The adminscope.criteria constraint uses the following format:

```
{
  "attr": "adminscope.criteria",
  "scope": [
     {
         "op": "eq",
         "value": "organization=${requestprincipal.organization}",
         "userAttr": "organization"
     }
   ]
}
```

The following table lists the provisioned resources with matching organization values.

Table 8: Provisioned resources with matching org vales

Resource Type	adminscope.criteria
group, applications, IDPs, and so on.	`"adminscope.criteria": [ "organization=banking ]`
User Principal	`"urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": { "organization": "banking },`
Client Principal (such as Application)	`"metadata": [ { ”name": "organization", "value": "banking" }, ... ]`

The `userAttr` specifier is used to look up the matching organization value to resolve the matching organization for requested Users and for the requesting User principal.

Using the `requestprincipal` Substitution Variable

When resolving an attribute value for the requesting principal, you can use a substitution variable at runtime to determine the value for that principal. For example:

```
Format:  ${requestPrincipal.atttibuteName}
```

In the default constraint, the given organization is the provisioned attribute, and the **organization** is the `attributeName` .

The adminscope.managedby Constraint

The adminscope.managedby constraint covers the various manager models that can be deployed for resource management.

- Individual named managers matching the requested principal are evaluated using `urn:iam:adminscope:me` value processing.
- Groups acting as resource managers evaluate the set of groups that the principal has against the group-based managers during `urn:iam:adminscope:megroups` value processing.

The adminscope.managedby constraint uses the following format:

```
{
  "attr": "adminscope.managedby",
  "scope": [
     {
        "op": "eq",
        "value": "urn:iam:adminscope:me",
        "userAttr": "managerId"
      },
      {
         "op": "eq",
         "value": "urn:iam:adminscope:megroups",
         "userAttr": "groups"
```

```
        }
    ]
}
```

Default ME Constraint

Describes the Default ME Constraint added to Administrative Roles with new tenants or when upgrading.

The Default ME Constraint is used by, and enforced by, User self-modification operations that are invoked using `${SSP_FQDN}/{{apiPathTenant}}/scim2/v1/Me` .

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/scim2/v1/Me`
  should look similar to:
  `https://example.com/default/scim2/v1/Me`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

This constraint contains RDS constraints only. The attributes that are defined in the data section of this constraint define what attributes a User can modify on their own profile.

Unrestricted Constraint

Describes the default Unrestricted Constraint added to Administrative Roles with new tenants or when upgrading.

The Unrestricted Constraint is a special purpose constraint that is used to preserve the unconstrained behavior of principals that are assigned to Administrative Roles before v3.3. This constraint is only deployed during upgrades to v3.3. The constraint only applies to principals that have been assigned to any of the out-of-the-box Administrative Roles whose privileges include resources that can be constrained. The unconstrained semantics of this constraint is achieved using the following attribute:

```
"adminScopeApi" : true,
```

This setting defers enforcement to the API scopes presented in the Access Token and ignores any RIS-defined constraint.

Adding or Removing a Constraint

Describes how to add a scoping constraint to, or remove a scoping constraint from, an Administrative Role.

After you define a Scoping Constraint, you can bind it to an Administrative Role:

- You can add or remove a scoping constraint using the AdminScopeConstrainer API. For information about adding or removing a scope constraint using the APIs, see Managing Scoping Constraints.
- The Admin Roles Detail page of the Admin Console provides a Constraints page that allows you to view or manage (add/remove) existing constraints that are defined by the Scoping Constraints API to a role. For information on using the Admin Console to add or remove constraints, see Administrators and Roles

Enabling Resource Admin Scope Enforcement for Resource Authorization

Describes Admin Scope enforcement for the Resource Authorization Model.

By default, the setting for Admin Scope enforcement is set to **false**. Enforcement is controlled by a global setting with an optional tenant-specific override. The following table describes the setting and the enforcement behavior when the setting (`isAdminScopeEnforced`) is set to **false** (the default). Using this setting allows deployments to build their authorization model over time and roll out the enforcement when fully tested.

Modify the `isAdminScopeEnforced` setting on the Managing Settings page.

Table 9: Default behavior for Admin Scope enforcement

Operation	Effect
Resource Access	No defined constraints are enforced
Constraint Definition (API only) Admin Role to constraint bindings	Allowed
Admin Role and Resource Delegations	Not allowed (APIs return 403 error)
Delegations Discovery	Allowed
Admin Console Resource page attribute value restrictions	No value restrictions are enforced

Determining Applicable RDS and RIS for a Given Principal Using Helper APIs

Describes how to use the Helper APIs to determine the applicable RDS and RIS for a given principal.

When the Resource Authorization model is ready to be evaluated, you can use Resource Schema Helpers to determine if the evaluations of the deployed constraints alongside deployed resource data are working properly. The following schema helpers return the evaluation of RDS and RIS values when passed an Access Token of the current principal to be evaluated:

Table 10: RDS and RIS evaluation schema helpers

Helper	Token	Output showing attribute constraint allowed values (RIS) and allowed permissions (RDS)	Interpreting the Results
admin/v1/UserSchemaHelper	Access Token for principal under test	"organization": { "account": false, "searchable": true, "visible": true, "operational": false, "mva": false, "name": "organization", "type": "STRING", **"permissions": "rw",** "custom": false … **"values"**: ["banking"], **"valueSchema"**: ["{\"op\":\"eq\",\"name\":null, \"value\":\"banking\"}"], "displayName": "Organization", }	The highlighted and underlined output shows evaluated RDS and RIS User constraints of the organization attribute for this principal as follows: • RDS permissions: rw • RIS – values: allowed values: banking – valueSchema: evaluated RIS “value eq banking”
		"groups": { "account": **false**, "searchable": **false**, "visible": **true**, "operational": **false**, "mva": **true**, **"values"**: ["CommercialBanking", "RetailBanking"], **"valueSchema"**: ["{\"op\":\"eq\",\"name\":null, \"value\":\"CommercialBanking \"}",	The highlighted and underlined output shows evaluated RDS and RIS User constraints of the groups attribute for this principal as follows: • RDS permissions: r • RIS – values: allowed values: • CommercialBanking • RetailBanking • valueSchema: evaluated RIS – “value eq CommercialBanking” – “value eq RetailBanking”

Running the Postman API Resource Authorization Demo

Describes how to run the step-by-step Resource Authorization demo using the Postman API.

The Resource Authorization demo provides a comprehensive demonstration of the following scenarios:

- A centralized model based on provisioning of resources and principals for two organizations (banking and investments)
- A decentralized model using delegation of managers to individual resources
- Admin Role and Resource Delegation

The demo provides setup and teardown of all data including:

- Organization-based clients
- Organization-based functional admins
- End Users, Groups Applications, IDPs. IDP Discovery Policies, Certificates
- Roles and Constraints

The demo creates a clone of the ‘Default Constraint’, and relies on these out-of-the-box constraints for most the use cases.

Running the Demo Setup

The Resource Authorization demo is part of the `AuthHub.postman_collection.json`. Run the Setup folder as a runnable collection using `Run Folder` as a run-once option. You should check **Persist Responses for a session** to check for any specific API failure responses.

Running the Resource Authorization Requests

The folders in this section execute specific authorization use cases. Each folder has a set of requests that are intended to be executed in the sequence defined. To ensure expected results, follow the sequence; Grant, Execute Operations, Revoke.

Verifying Expected Results

The Resource Authorization Folder has a subfolder that is named `Helper Authorization Verifications`. These helpers can be used to verify and troubleshoot results. The Helper requests use the same token that is used by the invoking principals in Resource Authorization requests. The attribute permissions and attribute values returned reflect the evaluated constraints for that principal.

Resource Authorization results for Symantec Identity Security Platform Resource APIs (such as User, Group, applications, and so on) can often return complex outcomes. Any IDSP Resource API that is subject to Resource Authorization supports the `explain` parameter. Use the `explain` parameter to provide detailed analysis and explanations of these outcomes.

See Understanding Resource Authorization Results using Explain.

Managing Authorization

Learn about Symantec Identity Security Platform Authorization Policies, administrator roles, and scopes that define the access permissions to be granted to an identity to access protected resources.

Within the IDSP, the process of authorization is managed through authorization policies. A collection of preloaded authorization policies controls access to administrative capabilities represented by individual permissions. These permissions serve as the basis for granting administrative capabilities to users, groups, or client applications. The following is the overview of the authorization methodology in IDSP:

- ***Permissions***: Determines the extent of access or the specific operations that the users, groups, or client applications can perform in IDSP. These permissions are modeled as OAUTH scopes used to secure IDSP APIs protected by an OAUTH security model. See Table of Scopes and its corresponding APIs.
- ***Policies***: Authorization Policies, (also referred to as AuthZ policies), defines the permissions that are granted to users, groups, or client apps within IDSP. These policies are governed by rules that specify the conditions for access permissions. IDSP comes preloaded with various tenant admin policies. However, only admins with the "urn:iam:t.adminroles" privilege, granted through a set of defined Admin Role APIs, can modify the "principal" condition within these policies. To know more about the Admin Roles API endpoints, see Authorization APIs.
- ***Roles***: Allows specific tasks or access rights to be delegated to users, groups, and client apps within IDSP. The solution offers various tenant admin roles, which are implemented through authorization policies. Each admin role is

equipped with distinct scopes, granting administrators the authority to execute their assigned tasks effectively. Details about the various tenant admin roles are covered in the next section. See Admin Roles.
Within the IDSP ecosystem, these administrators have the capability to perform their tasks using IDSP APIs.
The Admin Console UI has also been adjusted to support the Tenant Administrator (full access), Tenant Auditor (everything read-only), Credential administrator, and Identity administrator. All other admin roles are only supported by using APIs.
This comprehensive role-based access enables them to execute a range of administrative duties, including user lifecycle management, application provisioning, and configuration customization.

Important notes about Admin Roles:

- The admin policies that are preloaded into the solution are not removable by any of the administrators.
- The System Admin Policy Role has been entrusted with the authority to perform GET/POST/DELETE/PUT/PATCH operations using the AuthZ policies that are not embedded (seeded) in the solution. Additionally, they have the capability to modify ***ONLY*** the "***principal***" condition for any seeded policies (for example, Account Administrator Role). However, it is important to note that the System Administrator does not have the authority to delete any embedded IDSP policies.
- The Tenant Admin Policy Role is granted permission to modify **ONLY** the "***principal***" within an embedded policy (for example, Account Administrator Role). To know more about Policy condition, see Principle Condition.
- The ***PUT https://${SSP_FQDN}/<tenantName>/admin/v1/AuthZPolicies/{{authZPolicyID}}*** operation that a tenant admin could perform to update the "***principal***" condition for any embedded policy has been deprecated. Instead the POST operation using the https://${SSP_FQDN}/<tenantName>/admin/v1/AdminRolesDelegator API can be used.

Administrative Roles

The various predefined admin roles which are implemented by their respective policies have been explained in the following sections:

- Infrastructure Administrator
- System Administrator
- Signin Client
- Account Administrator
- Authentication Client
- Configuration Administrator
- Configuration Auditor
- Credential Administrator
- Identity Administrator
- Idstore Administrator
- Identity Client
- Integration Administrator
- Integration Auditor
- Risk Client
- Tenant Administrator
- Tenant Auditor
- Diagnostics Administrator
- User Password Administrator
- Registration Client Administrator

Infrastructure Administrator

Table 11: Infrastructure Administrator and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
InfraPolicy	The Infrastructure Administrator role is associated with the client called, **InfraClient** which performs infra-level operations such as tenant management and also seeds the first System Administrator identity and grants appropriate users or groups to the System Admin role. This role is empowered to oversee various tasks related to Applications, ID Stores, Identity Providers, Authentication and Authorization Policies, Authentication Factors, Risk Providers, and more. The "***Break Glass***" privilege is assigned to the Infra Admin Policy (InfraPolicy) role with the purpose of handling critical situations in which admins holding the System Admin Policy role encounter difficulties in performing their tasks. For example, • when the secret for the system client has been lost and must be reset • authorization policy has been mistakenly updated to lock out administrators, and must be reset In general, this credential is the "***super user***" credential and you should store it in a secure location.	• "urn:iam:t.apps" • "urn:iam:t.ldapconfigs" • "urn:iam:t.idstoreconfigs" • "urn:iam:t.idpconfigs" • "urn:iam:t.authpolicies" • "urn:iam:t.authzpolicies" • "urn:iam:g.settings" • "urn:iam:t.configs" • "urn:iam:t.signingkey" The privileges that are associated with the ***Infrastructure Administrator*** role are restricted to "***Client Only***," with no extension of these privileges to regular "User" and "Group" entities.

System Administrator

Table 12: System Administrator Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
SystemAdminPolicy	The System Administrator role is granted permission to perform all activities that are related to system configurations. Using this role, the admin can invoke any system APIs defined in the solution to assess certain system admin tasks post to configure the environment and meet administration and/or security/ governance needs. For example, you may want to configure maximum expiration interval for tokens, or delegate the right to perform system administration to some admin users, etc. These tasks are performed by Users or Applications having the System Admin role. For convenience, you can use the"SystemClient" credential that is seeded out of the box to perform such tasks. This role is empowered to oversee various tasks that are related to Applications, ID Stores, Identity Providers, Authentication and Authorization Policies, Authentication Factors, Tenant Configurations, Global Settings, Admin Roles, and many more. The "***crosstenant***" ("urn:iam:g.crosstenant") scope granted to the System Administrator allows the admin to carry out operations across tenants by obtaining a cross-tenant Access Token (AT). For example, the System Admin can perform GET, POST, DELETE, PUT, or PATCH operations on policies that are not embedded (seeded) in the solution. Additionally, they have the capability to modify ***ONLY*** the "***principal***" condition for any embedded policies. However, it is important to note that the System Administrator does not have the authority to delete any embedded IDSP policies	• "urn:iam:g.tenant" • "urn:iam:g.crosstenant" • "urn:iam:t.authenticate" • "urn:iam:t.apps" • "urn:iam:t.certs" • "urn:iam:t.secrets" • "urn:iam:t.secretsgetter" • "urn:iam:t.ldapconfigs" • "urn:iam:t.idstoreconfigs" • "urn:iam:t.idpconfigs" • "urn:iam:t.authpolicies" • "urn:iam:t.authzpolicies" • "urn:iam:g.settings" • "urn:iam:t.configs" • "urn:iam:t.signingkey" • "urn:iam:t.adminroles" • "urn:iam:g.authzpolicywriter" The privileges that are associated with the ***System Administrator*** role extends to "***Client***", "***User***" and "***Group***" entities

Tenant Administrator

Table 13: Tenant Administrator Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
TenantAdminPolicy	The Tenant Administrator role is granted permission to perform all activities that are related to a tenant. The tenant policy that represents this role is associated with the client called, **DefaultTenantClient** and the scope notation, `urn:iam:t.*` represents tenant-level privileges. A user named, "***DefaultTenantAdminUser"*** with full administrative rights is also provisioned. The Tenant Administrator role is empowered to oversee various tasks that are related to Applications, ID Stores, Identity Providers, Authentication and Authorization Policies, Authentication Factors, Tenant Configurations, Global Settings, Admin Roles, Secrets, Signing Keys, and many more. Some of the tasks that a tenant admin performs are listed here: • Creating an "application" entity representing protected applications • Creating and managing Authentication Policies defining the level of authentication that is required by the application • Managing configuration settings such as certificates, message templates, available authentication factors, etc. • Creating and Managing a "LDAPConfig" entity representing the identity store with users to be authenticated by IDSP • Delegating Tenant Admin role to users and groups. Also to applications that wish to invoke IDSP APIs.	• "urn:iam:t.all" The privileges of ***Tenant Administrator*** role extends to "***Client***", "***User***" and "***Group***" entities

Signin Client

Table 14: Signin Client Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
SigninClient	The Signin Client role is granted permission to view and manage the out-of-the-box SignIn application that is provided by IDSP. The Signin Client policy that represents this role is associated with the client called, **SigninClient**. The Signin Client Role is empowered to oversee various tasks that are related to Factor Authentication, Branding, Password Reset, UI Locale, and more. The crosstenant ("urn:iam:g.crosstenant") scopes granted to the Signin Client allows the admin to carry out operations across tenants by obtaining a cross-tenant Access Token (AT).	• "urn:iam:t.authenticate" • "urn:iam:g.crosstenant" The privileges that are associated with the ***Signin Client*** role are restricted to "***Client Only***," with no extension of these privileges to regular "User" and "Group" entities.

Account Administrator

Table 15: Account Administrator Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
AccountAdminPolicy	The Account Administrator role is granted permission to manage user accounts in IDSP. For example, the Account Administrator can check the user's "Status", and also update the user's account status as "Locked" or "Unlocked" in the ID Store.	"urn:iam:t.accountadmin" The privileges that are associated with the ***Account Administrator*** role extends to "***Client***", "***User***" and "***Group***" entities.

Authentication Client

Table 16: The Authentication Client Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
AuthClientPolicy	The Authentication Client role is granted permission to perform operations that are related to authentication to implement a custom Signin app. The Authentication Client policy that represents this role is associated with the client called, **SigninClient** which performs authenticate operations for the SignIn app. The Authentication Client role grants clients the capability to initiate and execute an authentication flow in IDSP. It does not grant any additional privileges, and the desired purpose is to obtain an access token for the user with their designated scopes. The user may be a tenant administrator or a user with a self-service API. .	"urn:iam:t.authenticate" The privileges that are associated with the ***Authentication Client*** role are restricted to "***Client Only***," with no extension of these privileges to regular "User" and "Group" entities.

Configuration Administrator

Table 17: Configuration Administrator Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
ConfigurationAdminPolicy	The Configuration Administrator role is empowered to oversee various tasks that are related to Security Settings, Authentication policies, ID Stores, Tenant Configurations, Certificates, Secrets, Signing Keys, and more.	• "urn:iam:t.configs" • "urn:iam:t.authpolicies" • "urn:iam:t.idstoreconfigs" • "urn:iam:t.certs" • "urn:iam:t.secrets" • "urn:iam:t.signingkey" • "urn:iam:t.idstorebrowser" • "urn:iam:t.adminroles" The privileges that are associated with the **Configuration Administrator** role extends to "***Client***", "***User***" and "***Group***" entities.

Configuration Auditor

Table 18: Configuration Auditor Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
ConfigurationAuditorPolicy	The Configuration Auditor role is granted permission to audit tasks that are related to Security Settings, Authentication policies, ID Stores (Users and Groups), Tenant Configurations, Certificates, Secrets, Signing Keys, Admin Roles, and more.	• "urn:iam:t.configs" • "urn:iam:t.authpolicies" • "urn:iam:t.idstoreconfigs" • "urn:iam:t.certs" • "urn:iam:t.secrets" • "urn:iam:t.signingkey" • "urn:iam:t.auditonly" • "urn:iam:t.adminroles" • "urn:iam:t.idstorebrowser" The privileges that are associated with the ***Configuration Auditor*** role extends to "***Client***", "***User***" and "***Group***" entities.

Credential Administrator

Table 19: Credential Administrator Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
CredentialAdminPolicy	The Credential Administrator role is granted permission to oversee various aspects in an organization, including managing user account lock status, credentials, trusted and recognized devices, and overseeing user sessions.	• "urn:iam:t.userdata" • "urn:iam:t.session" The privileges that are associated with the ***Credential Administrator*** role extends to "***Client***", "***User***" and "***Group***" entities.

Identity Administrator

Table 20: Identity Administrator Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
IdentityAdminPolicy	The Identity Administrator role is granted the authority to oversee the management of "users" and "groups" within the internal Identity Store. Additionally, users who are entrusted with this role have the permission to manage various other aspects such as user account lock status, credentials, trusted devices, known devices, and sessions.	• "urn:iam:t.users" • "urn:iam:t.groups" • "urn:iam:t.session" The privileges that are associated with the ***Identity Administrator*** role extends to "***Client***", "***User***" and "***Group***" entities.

Idstore Administrator

Table 21: Idstore Administrator Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
IdstoreAdminPolicy	The Idstore Administrator role is granted the permission to oversee the management of both "users" and "groups" within the configured identity stores.	• "urn:iam:t.idstoreadmin" The privileges that are associated with the ***Idstore Administrator*** role extends to "***Client***", "***User***" and "***Group***" entities.

Identity Client

Table 22: Identity Client Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
IdentityClientPolicy	The Identity Client role is granted the permission to oversee the identity-related operations.	• "urn:iam:t.identityclient" The privileges that are associated with the ***Identity Client*** role are restricted to "***Client Only***," with no extension of these privileges to regular "User" and "Group."

Integration Administrator

Table 23: Integration Administrator Role and Associated Privileges given to this role (scopes)

Policy Name	Purpose	Privileges
IntegrationAdminPolicy	The Integration Administrator role is granted the authority to manage Applications, IDPs (Identity Providers), and IDP Discovery policies. Additionally, they manage Identity Stores, Certificates, and Secrets.	• "urn:iam:t.apps" • "urn:iam:t.idpconfigs" • "urn:iam:t.idstoreconfigs" • "urn:iam:t.certs" • "urn:iam:t.secrets" • "urn:iam:t.idstorebrowser" • "urn:iam:t.idpdiscoverypolicies" • "urn:iam:t.dcr" The privileges that are associated with the ***Integration Administrator*** role extends to "***Client***", "***User***" and "***Group***" entities.

Integration Auditor

Table 24: Integration Auditor Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
IntegrationAuditorPolicy	The Integration Auditor role is granted the authority to audit Applications, IDPs, IDP Discovery policies. Additionally, admins entrusted with this role can audit Identity Stores, Certificates, and Secrets. This role is equipped with the read-only access, and while cannot execute any operations, they are able to view all App, IDPs, and IDP Discovery policies.	• "urn:iam:t.apps" • "urn:iam:t.idpconfigs" • "urn:iam:t.idstoreconfigs" • "urn:iam:t.certs" • "urn:iam:t.secrets" • "urn:iam:t.auditonly" • "urn:iam:t.idpdiscoverypolicies" The privileges of ***Integration Auditor*** role extends to "***Client***", "***User***" and "***Group***" entities.

Risk Client

Table 25: Risk Client Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
RiskClientPolicy	The Risk Client role is granted the authority to perform risk-related operations.	• "urn:iam:t.risk" The privileges of the ***Risk Client*** role are restricted to "***Client Only***," with no extension of these privileges to regular "User" and "Group."

Tenant Auditor

Table 26: Tenant Auditor Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
TenantAuditorPolicy	The Tenant Auditor role is given the authorization to conduct audits on all operations within a tenant. They are equipped with the read-only access, and while they cannot execute any operations, they are able to view all tenant-related activities.	• "urn:iam:t.all" • "urn:iam:t.auditonly" • "urn:iam:t.adminroles" The privileges of ***Tenant Auditor*** role extends to "***Client***", "***User***" and "***Group***" entities.

Diagnostics Administrator

Table 27: Diagnostics Administrator Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
DiagnosticsAdminPolicy	This is for internal use only.	• "urn:iam:t.diagnosticsadmin"

User Password Administrator

Table 28: User Password Administrator Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
UserPasswordAdministratorPolicy	The User Password Administrator is granted permission to manage user passwords.	• "urn:iam:t.userpasswordadmin" The privileges of ***User Password*** role extends to "***Client***", and "***User***" entities.

Registration Client Administrator

Table 29: Registration Client Administrator Role and Associated Privileges (scopes)

Policy Name	Purpose	Privileges given to this role
RegistrationClientPolicy	The Registration Client Administrator is granted the authority to call the dynamic client registration API with the initial access token.	• "urn:iam:t.dcr" The privileges that are associated with the ***Registration Client*** role are restricted to "***Client Only***," with no extension of these privileges to regular "User" and "Group" entities.

See the following topics:

- Table of Scopes and its corresponding APIs
- Authorization APIs

Scopes and APIs Table

Learn about the Symantec Identity Security Platform Scopes and the APIs associated with these scopes that can be triggered by authorized admins to get required information and perform related tasks.

Table of Scopes with associated APIs:

API Scope	API URI	API Description	Global / Tenant / Me
"urn:iam:g.authzpolicywriter"	"/admin/v1/AuthZPolicies"		G
"urn:iam:g.crosstenant",	All APIs with the scope urn:iam:t.*		G
urn:iam:g.settings	"/admin/v1/MessageProvider" "/admin/v1/FlushProviderCache" "/admin/v1/CustomProviders" "/admin/v1/hc" "/admin/v1/GroupOperator" "/admin/v1/Settings" "/admin/v1/VIPConfiguration" "/admin/v1/Factorsettings" "/admin/v1/RiskProviders" "/admin/v1/EncryptionProviders" "/admin/v1/LocaleContent" "/admin/v1/FraudIPUploader"	Allows to perform CRUD operations to manage message providers, custom providers, global and tenant settings, VIP configurations,	G
"urn:iam:g.tenant"	"/admin/v1/Tenants" "/admin/v1/hc" "/sm/v1/TenantDisabler"		T
"urn:iam:t.accountadmin"	"/admin/v1/Users" "/admin/v1/UserLocker" "/admin/v1/GroupOperator"		T
"urn:iam:t.adminroles"	"/admin/v1/AdminRolesDelegator"		T
"urn:iam:t.allowstartindexsearchparam"	"/scim2/v1/Users" "/scim2/v1/Groups"	Allows to perform searches with start index.	T
urn:iam:t.apps	"/admin/v1/Apps"	Allows to perform CRUD operations on applications	T

API Scope	API URI	API Description	Global / Tenant / Me
urn:iam:t.authenticate	"/auth/v1/authenticate" "/geolocation/v1/IPAddress" "/auth/v1/UserAuthenticator" "/auth/v1/SelectedFactor" "/auth/v1/UILocaleContent" "/auth/v1/BrandingSettings/idp" "/auth/v1/BrandingSettings" "/auth/v1/NonceGenerator" "/factor/v1/PasswordAuthenticator" "/factor/v1/CustomAuthenticator" "/factor/v1/OTPRegistrar" "/factor/v1/OTPGenerator" "/factor/v1/OTPVerifier" "/factor/v1/MobileOTPRegistrar" "/factor/v1/VIPCredentialRegistrar" "/factor/v1/MobileOTPVerifier" "/factor/v1/PushRegistrarInitiator" "/factor/v1/PushRegistrarStatus" "/factor/v1/PushRegistrar" "/factor/v1/PushGenerator" "/factor/v1/PushVerifier" "/factor/v1/PushStatus" "/factor/v1/FIDOAuthChallengeGenerator" "/factor/v1/FIDOAuthChallengeVerifier" "/factor/v1/FIDORegChallengeGenerator" "/factor/v1/FIDORegChallengeVerifier" "/oauth2/v1/idp" "/oauth2/v1/byoi" "/identity/v1/MeUserPasswordChanger" "/identity/v1/MeUserPasswordValidator" "/identity/v1/EmailVerificationInitiator" "/identity/v1/EmailVerifier" "/identity/v1/PhoneVerificationInitiator" "/identity/v1/PhoneVerifier" "/identity/v1/VerifiedCredentials" "/identity/v1/PasswordResetInitiator" "/identity/v1/PasswordResetValidator" "/identity/v1/PasswordResetChanger" "/admin/v1/BrandingSettings/idp" "/admin/v1/BrandingSettings" "/factor/v1/ FIDOTransactionSigningChallengeGenerator" "/factor/v1/ FIDOTransactionSigningChallengeVerifier" "/admin/v1/UILocaleContent"		T
urn:iam:t.authpolicies	"/admin/v1/AuthPolicies"	Allows to perform CRUD operations on authentication policies.	T

API Scope	API URI	API Description	Global / Tenant / Me
urn:iam:t.authzpolicies	"/admin/v1/AuthZPolicies" "/admin/v1/AuthZPoliciesEvaluator"	Allows to perform CRUD operations on authorization policies.	T
"urn:iam:t.auditonly"			T
urn:iam:t.authzpolicywriter	"/admin/v1/AuthZPolicies" "/admin/v1/AuthZPoliciesEvaluator"	Allows to perform CRUD operations on authorization policies.	T
"urn:iam:t.certs"	"/admin/v1/Certs"		T
urn:iam:t.configs	"/admin/v1/Messages" "/admin/v1/MessageProviders" "/admin/v1/FlushProviderCache" "/admin/v1/CustomProviders" "/admin/v1/Configs" "/admin/v1/SigningCert" "/admin/v1/VIPConfiguration" "/admin/v1/RiskFactors" "/admin/v1/NegativeCountries" "/admin/v1/UntrustedIps" "/admin/v1/ExceptionUsers" "/admin/v1/RiskProviders" "/admin/v1/PasswordPolicies" "/admin/v1/PasswordDictionary" "/admin/v1/SAMLIdpMetadata" "/iarisk/v1/RiskThreshold" "/iarisk/v1/RiskyCountries" "/iarisk/v1/IPList" "/iarisk/v1/Policies" "/iarisk/v1/RiskRules" "/admin/v1/EncryptionProviders" "/admin/v1/ProviderHelper" "/admin/v1/IntegrationHelper" "/admin/v1/InternalIdStoreConfig"		T
"urn:iam:t.diagnosticsadmin"	"/admin/v1/Diagnostics"		T
"urn:iam:t.exceedmaxscimsearchlimit"	"/scim2/v1/Users" "/scim2/v1/Groups"	Allows to perform searches beyond the search limit.	T
"urn:iam:t.groups"	"/scim2/v1/Groups" "/scim2/v1/Groups/.search" "/scim2/v1/.search" "/scim2/v1/Bulk" "/admin/v1/GroupOperator" "/scim2/v1/Schemas" "/scim2/v1/ServiceProviderConfig" "/scim2/v1/ResourceTypes"		T

API Scope	API URI	API Description	Global / Tenant / Me
"urn:iam:t.identityclient"	"/scim2/v1/Users/.search" "/scim2/v1/Groups/.search" "/scim2/v1/Schemas" "/scim2/v1/ServiceProviderConfig" "/scim2/v1/ResourceTypes" "/scim2/v1/PasswordAuthenticator" "/identity/v1/UserPasswordValidator" "/identity/v1/UserPasswordChanger" "/identity/v1/GenerateTemporaryPassword" "/identity/v1/EmailVerificationInitiator" "/identity/v1/EmailVerifier" "/identity/v1/PhoneVerificationInitiator" "/identity/v1/PhoneVerifier" "/identity/v1/VerifiedCredentials" "/identity/v1/PasswordResetInitiator" "/identity/v1/PasswordResetValidator" "/identity/v1/PasswordResetChanger"		T
urn:iam:t.idpconfigs	"/admin/v1/IdentityProviders" "/admin/v1/SAMLIdpMetadata"	Allows to perform CRUD operations on Identity Providers and additionally, allows to import and modify the SAML metadata received from an Identity Provider.	T
"urn:iam:t.idpdiscoverypolicies"	"/admin/v1/IdpDiscoveryPolicies"		T
"urn:iam:t.idstoreadmin"	"/idstore/v1/Users" "/idstore/v1/Groups" "/idstore/v1/UserLocker"		T
"urn:iam:t.idstorebrowser"	"/idstore/v1/Users" "/idstore/v1/Groups"		T
urn:iam:t.idstoreconfigs	"/admin/v1/LDAPconfigs" "/admin/v1/SCIMconfigs"	Allows to perform CRUD operations on ID Store, LDAP, and SCIM.	T
urn:iam:t.ldapconfigs	"/admin/v1/LDAPconfigs" "/admin/v1/SCIMconfigs"	Allows to perform CRUD operations on ID Store, LDAP, and SCIM.	T
"urn:iam:t.risk"	"/risk/v1/UserRiskScoreEvaluator" "/risk/v1/UserRiskScoreUpdater" "/risk/v1/UserRiskScoreGetter" "/iarisk/v1/UserRiskScoreEvaluator" "/iarisk/v1/PostUserRiskScoreEvaluator" "/iaservices/risk/v1/confirm" "/iarisk/v1/UserRiskScore"		T
"urn:iam:t.secrets"	"/admin/v1/Secrets" "/admin/v1/Keys"		T
"urn:iam:t.secretsgetter"	"/admin/v1/SecretsGetter"		T
"/identity/v1/ServiceAccounts"	Allows viewing and validating of service accounts for a user.	T	

API Scope	API URI	API Description	Global / Tenant / Me
"urn:iam:t.serviceaccountsadmin"	"/scim2/v1/Users" "/scim2/v1/Bulk"	Allows administrators to set the user `serviceaccount` attribute to `true` /`false` .	T
"urn:iam:t.session"	"/admin/v1/revoke"		T
urn:iam:t.signingkey	"/admin/v1/SigningKey"		T
"urn:iam:t.userdata"	"/admin/v1/Creds" "/admin/v1/Users" "/admin/v1/TrustedDevices" "/iarisk/v1/KnownDevices"		T
"urn:iam:t.userpasswordadmin"	"/identity/v1/UserPasswordChanger"		T
"urn:iam:t.users"	"/admin/v1/Creds" "/scim2/v1/Users" "/scim2/v1/Users/.search" "/scim2/v1/.search" "/scim2/v1/Bulk" "/scim2/v1/Schemas" "/scim2/v1/ServiceProviderConfig" "/scim2/v1/ResourceTypes" "/scim2/v1/PasswordAuthenticator" "/admin/v1/TrustedDevices" "/admin/v1/Users" "/admin/v1/UserLocker" "/iarisk/v1/KnownDevices" "/admin/v1/PasswordUpdater"		T

Authorization API

Learn about the Symantec Identity Security Platform Authorization Policy Stricture, Authorization APIs, its operations, and examples.

This topic includes the following sections:

- Authorization Policy Structure
- Authorization APIs
- Authorization API Examples
  - Performing GET operation on Tenant Authorization Policy
  - Performing GET operation on Admin Roles
  - Performing POST operation on Admin Roles
  - Performing DELETE operation on Admin Roles
- Examples of Managing System Admin Operations
- Steps to grant "$AUTHCLIENT_PRIVILEGES" privileges to a client app
- Performing GET operation on My Admin Roles

Authorization Policy Structure

Authorization Policies, also referred to as AuthZ policies, determine the permissions that should be assigned to a user, group, or client application for accessing and performing actions on a protected resource. An administrator can create an authorization policy for an application, and can perform CRUD operations on it. Each policy comprises of rules, and each rule comprises of "**conditions**" and "**result**", which helps in determining how the admins should be granted permissions

and access to the solution. Eventually, the authorization policy lays out an authorization guideline for the admins which is governed by the "***principal***" condition that is specified within the rules of the policy. The "***principal***" condition supports three values, "clientApp", "user" and "group".

NOTE

- Symantec Identity Security Platform provides a default OOTB authorization policy for OOTB application (called SSP_APP). This policy cannot be updated or edited. However, with Symantec Identity Security Platform v4.0 onwards, you can now create authorization policy for custom applications, such as a Resource Server app.
- Application group is not supported in Authorization policy.

The following sample is an example of a request payload for an Authorization Policy:

```
{
"policyName": "SampleAdminAuthzPolicy",
"description": "Sample admin authorization policy",
"status": "active",
"isGrantableClientOnly": false,
"isSubdelegatable": false,

"displayName": "Sample admin authorization policy",
"apps": [
{
"name": "SSP",
"id": "$SSP_APP"
}
],
"rules": [
{
"conditions": {
  "principal": {
      "clientApp": {
          "operator": "in",
          "value": [
              "<CHANGE-ME>"
          ]
      },
      "user": {
          "operator": "in",
          "value": [
              "<CHANGE-ME>"
          ]
      },
      "group": {
          "operator": "in",
          "value": [
              "<CHANGE-ME>"
          ]
      }
  }
},
"result": {
  "msg": "This is a sample admin authorization policy",
  "privileges": [
```

```
        "urn:iam:t.idpconfigs",
        "urn:iam:t.ldapconfigs",
        "urn:iam:t.idstoreconfigs"
    ]
  }
  }
  ]
  }
```

Authorization Policy Attributes

The details of the policy attributes are described here:

- **policy name** – The name displays the title of the policy
- **description** – The description details a brief about the policy
- **status** – The status displays whether a policy is **active** or **inactive**
- **isGrantableClientOnly** - This attribute is a boolean value that restricts the values that can be passed when set to "true" or "false."
  - When "isGrantableClientOnly" is set to "***true***," it only permits "***clientApp***" in the principal condition.
  - When "isGrantableClientOnly" is set to "***false***," it allows "***User***," and "***Group***" in the principal condition.
- **isSubdelegatable** - This attribute is a boolean value that restricts whether a principal who has been granted a role can then delegate that role to another principal.
  - When "isSubdelegatable " is set to "***true***," the principal can delegate the role to another principal. The Delegate option for that role appears in the Administrative Console, and the role can be delegated using the `AdminRolesSubdelegator` API.
  - When "isSubdelegatable " is set to "***false***," the principal cannot delegate the role to another principal. The Delegate option for that role does not appear in the Administrative Console, and any attempts to delegate the role using the `AdminRolesSubdelegator` API fail.
- **displayName** - This attribute is used to provide a descriptive title or identifier for the policy, making it easier for administrators and users to understand the function of the policy.
- **apps** – This attribute denotes the name and id of the "clientApp" to which this authorization policy is applied. An OOTB authorization policy with client app (called $SSP_APP) is provided by Symantec Identity Security Platform. This authorization policy cannot be updated or edited.
  - With Symantec Identity Security Platform v4.0 onwards, you can create separate authorization policy for a custom application such as OAuth resource server application.

  For more information on apps in general, see Application API.
- **rules** - Rules include **conditions** and **result**. If the specified conditions in the rule are met, then the corresponding result is applied.
  - **conditions**: This attribute allows you to specify a "***principal***" condition to a rule. The "***principal***" condition supports these values: "***clientapp"***, "***group"*** and "***user***". To know more about users and groups, see Principle Condition.
  - **result** - This property displays the following two sub-properties:
    - **msg**- This property allows for a brief, informative description of the authorization policy. This property can include details about the objectives of the policy, usage guidelines, or any other relevant information to help admins understand its role and functionality."
    - **privileges**- This attribute outlines the set of permissions that are granted by the authorization policy. This attribute specifies the actions, operations, or capabilities that users, groups, or clients are authorized to perform on the protected resource.

Authorization APIs

As a System Administrator, you can use the following Security APIs to retrieve and update privileges of the client apps or users in the system.

Operation Description	Admin Roles	Operation Type	Endp
Retrieves all AuthZ Policies	System/Tenant Admin	GET	https https
Retrieves the AuthZ policy of the given policy Id.	System/Tenant Admin	GET	https https
Creates a new AuthZ Policy	System Admin	POST	https
Updates existing AuthZ Policy of the given policy Id.	System Admin	PUT	https ICY_
Deletes an AuthZ Policy by specifying of the given policy Id. ***Note that System admins cannot delete the embedded policies.***	System Admin	DELETE	https POL To fo Reso
Admin Role APIs			
Retrieves all the admin roles that are configured in the solution	System/Tenant Admin	GET	https https
Grants access to an admin role. With this API, the admin can overwrite only the "***principal"*** condition in the policy configuration. (The "***principal"*** condition can include Group, User, and Client App).	System/Tenant Admin	POST (Grants admin access)	https https
Revokes admin access from an administrator role. With this API, the admin can overwrite only the "***principal"*** condition in the policy configuration. (The "***principal"*** condition can include Group, User, and Client App).	System/Tenant Admin	DELETE (Revokes admin access)	https https
Returns the list of admin capabilities that are associated with the given user.	User Access Token	GET	https

Important notes about Admin Roles:

- The admin policies that are preloaded into the solution are not removable by any of the administrators.
- The System Admin Policy Role has been entrusted with the authority to perform GET/POST/DELETE/PUT/PATCH operations using the AuthZ policies that are not embedded (seeded) in the solution. Additionally, they have the capability to modify ***ONLY*** the "***principal***" condition for any seeded policies (for example, Account Administrator Role). However, it is important to note that the System Administrator does not have the authority to delete any embedded IDSP policies.
- The Tenant Admin Policy Role is granted permission to modify **ONLY** the "***principal***" within an embedded policy (for example, Account Administrator Role). To know more about Policy condition, see Principle Condition.
- The ***PUT https://${SSP_FQDN}/<tenantName>/admin/v1/AuthZPolicies/{{authZPolicyID}}*** operation that a tenant admin could perform to update the "***principal***" condition for any embedded policy has been deprecated. Instead the POST operation using the https://${SSP_FQDN}/<tenantName>/admin/v1/AdminRolesDelegator API can be used.

See Managing Authorization in IDSP using the Admin Roles.

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default". "system" is the default value for <systemTenantName>.
- Replace the text {text}/<text> including {}/<> characters with appropriate values in all the endpoints.

For example: `https://${SSP_FQDN}/<tenantName>/admin/v1/AuthenticationFactor` should look similar to: `https://example.com/default/admin/v1/AuthZPolicies`

- To know about the scopes that are supported by these endpoints, see Rest APIs and Associated Privileges section.

Example of Authorization API Operations

Performing GET operation on Tenant Authorization Policy

The following example shows the request and response of ***Tenant Authorization Policy API***.

Perform a GET request using the following Authorization API:

```
https://${SSP_FQDN}/<tenantName>/admin/v1/AuthZPolicies
```

The response fetches the authorization policies.

```
[
{
"policyId": "87cd3d02-02ec-413a-a541-783d059e200f",
"policyName": "Sample policy1",
"description": "Sample policy for TestApp ",
"status": "active",
"apps": [
{
  "name": "DemoClient",
  "id": "1520154f-94f6-4c2f-8b25-b7fcd6cd4cf7"
}
],
"rules": [
{
  "id": "45159f8b-9b39-4098-bb7b-23ffd9ddeda5",
  "conditions": {
      "principal": {
          "group": {
              "operator": "in",
              "value": [
                  "HR"
              ]
          }
      },
      "action": {
          "operator": "in",
          "value": [
              "login"
          ]
      },
      "ipAddress": {
          "operator": "in",
          "value": [
              "10.0.0.0/24"
          ]
      },
      "channel": {
          "operator": "in",
```

```
            "value": [
                "mobile"
            ]
        },
        "riskThreshold": {
            "operator": "ge",
            "value": 60
        }
    },
    "result": {
        "effect": "allow",
        "reAuthenticate": "false",
        "obligations": [
            "PASSWORD",
            "SMSOTP",
            "EMAILOTP"
        ],
        "msg": "This policy is for login in mobile channel",
        "authLevel": 100
    }
}
],
"createdBy": {
"principalType": "CLIENT",
"principalId": "8b3bef5b-b8ab-4ba8-b56b-0a2807694b5a",
"principalName": "DefaultTenantClient"
},
"createdDate": "2023-09-28 05:27 AM UTC"
},
{
"policyId": "91701a86-f2fa-4dc1-b66d-7596b2025d84",
"policyName": "Feature Policy for Demo client",
"description": "Feature Policy Demo client",
"status": "active",
"apps": [
{
  "name": "DemoClient",
  "id": "1520154f-94f6-4c2f-8b25-b7fcd6cd4cf7"
}
],
"rules": [
{
  "id": "34f2bdbe-8c4b-4c74-8c0c-6a7c45eea624",
  "conditions": {
      "principal": {
          "group": {
              "operator": "in",
              "value": [
                  "Employees",
                  "Security Administrators"
              ]
          }
      },
```

```
      "action": {
          "operator": "in",
          "value": [
              "authenticate"
          ]
      },
      "channel": {
          "operator": "in",
          "value": [
              "web"
          ]
      }
  },
  "result": {
      "effect": "allow",
      "reAuthenticate": "false",
      "obligations": [
          "PASSWORD:1"
      ],
      "msg": "This condition is for the Employees Group with Password Only - Login with nbruce. Also for the
 Security Administrators Group when using OAuth/OIDC Postman flows - Login with mhunter, rchase, or vhickey."
  }
},
{
  "id": "db05058b-3480-4edf-8bbd-66a4b413a018",
  "conditions": {
      "principal": {
          "group": {
              "operator": "in",
              "value": [
                  "Employees",
                  "Human Resources",
                  "Finance",
                  "Development",
                  "Customers"
              ]
          }
      },
      "action": {
          "operator": "in",
          "value": [
              "authenticate"
          ]
      },
      "riskThreshold": {
          "operator": "ge",
          "value": 60
      }
  },
  "result": {
      "effect": "allow",
      "reAuthenticate": "false",
      "obligations": [
```

```
            "PASSWORD:1",
            "EMAILOTP:2",
            "SMSOTP:2"
        ],
        "msg": "This condition is for Risk > 60 on mobile and web channels. Login with nbruce, swise, or vburn
 with untrusted IP."
    }
},
{
    "id": "11d2a19a-732f-48fe-b8a4-8aef85f412ac",
    "conditions": {
        "principal": {
            "group": {
                "operator": "in",
                "value": [
                    "Employees"
                ]
            }
        },
        "action": {
            "operator": "in",
            "value": [
                "wealth"
            ]
        },
        "channel": {
            "operator": "in",
            "value": [
                "web"
            ]
        }
    },
    "result": {
        "effect": "allow",
        "reAuthenticate": "false",
        "obligations": [
            "MOBILEOTP:2"
        ],
        "msg": "This condition is for testing Continuous authentication on the Web Channel. Login with nbruce
 and Select Wealth Management to invoke this Condition."
    }
},
{
    "id": "65f5bd46-9932-4514-a50b-db221996ffeb",
    "conditions": {
        "principal": {
            "group": {
                "operator": "in",
                "value": [
                    "Development"
                ]
            }
        },
```

```
        "action": {
            "operator": "in",
            "value": [
                "authenticate"
            ]
        },
        "channel": {
            "operator": "in",
            "value": [
                "web"
            ]
        }
    },
    "result": {
        "effect": "allow",
        "reAuthenticate": "false",
        "obligations": [
            "FIDO:1",
            "SECURITYKEY:1"
        ],
        "msg": "This condition is Passwordless for the Development Group. Login with cmoran."
    }
},
{
    "id": "d591163e-3ac7-499c-b4f0-73825899bd13",
    "conditions": {
        "principal": {
            "group": {
                "operator": "in",
                "value": [
                    "Customers"
                ]
            }
        },
        "action": {
            "operator": "in",
            "value": [
                "authenticate"
            ]
        },
        "channel": {
            "operator": "in",
            "value": [
                "web"
            ]
        }
    },
    "result": {
        "effect": "allow",
        "reAuthenticate": "false",
        "obligations": [
            "PASSWORD:1",
            "FIDO:1",
```

```
            "SECURITYKEY:1",
            "SMSOTP:2",
            "EMAILOTP:2",
            "MOBILEOTP:2",
            "PUSH:2"
        ],
        "msg": "This condition is PW + OTP/MOBILEOTP/PUSH or FIDO + OTP/MOBILEOTP/PUSH or SECURITYKEY + OTP/
MOBILEOTP/PUSH with choices for the Customers Group on the Web Channel. Login with droth."
    }
},
{
    "id": "86d2cf6b-abd9-414b-aee0-99ae20080653",
    "conditions": {
        "principal": {
            "group": {
                "operator": "in",
                "value": [
                    "Human Resources"
                ]
            }
        },
        "action": {
            "operator": "in",
            "value": [
                "authenticate"
            ]
        },
        "channel": {
            "operator": "in",
            "value": [
                "web",
                "mobile"
            ]
        }
    },
    "result": {
        "effect": "allow",
        "reAuthenticate": "false",
        "obligations": [
            "PASSWORD:1",
            "EMAILOTP:2",
            "SMSOTP:2",
            "FIDO:2",
            "SECURITYKEY:2"
        ],
        "msg": "This condition is for the Human Resources Group with PW+OTP or PW+FIDO or PW+SECURITYKEY on the
 Web and Mobile Channels. Login with edunn for autoenrol and erussell for inline enrol."
    }
},
{
    "id": "710e97ec-22f1-4596-bad1-04e73c05a2bc",
    "conditions": {
        "principal": {
```

```
            "group": {
                "operator": "in",
                "value": [
                    "Employees"
                ]
            }
        },
        "action": {
            "operator": "in",
            "value": [
                "investment",
                "business"
            ]
        },
        "channel": {
            "operator": "in",
            "value": [
                "web",
                "mobile"
            ]
        }
    },
    "result": {
        "effect": "allow",
        "reAuthenticate": "false",
        "obligations": [
            "SMSOTP:2",
            "EMAILOTP:2"
        ],
        "msg": "This condition is for testing Continuous authentication on the Web Channel. Login with nbruce
 and Select Business Banking or Details to invoke this Condition."
    }
},
{
    "id": "a9d28629-9e08-449a-8039-f167599599ac",
    "conditions": {
        "principal": {
            "group": {
                "operator": "in",
                "value": [
                    "Employees"
                ]
            }
        },
        "action": {
            "operator": "in",
            "value": [
                "commercial"
            ]
        },
        "channel": {
            "operator": "in",
            "value": [
```

```
                "web",
                "mobile"
            ]
        }
    },
    "result": {
        "effect": "allow",
        "reAuthenticate": "false",
        "obligations": [
            "FIDO:2",
            "SECURITYKEY:2"
        ],
        "msg": "This condition is for testing Continuous authentication on the Web Channel. Login with nbruce
 and Select Commercial Banking to invoke this Condition."
    }
},
{
    "id": "d628d56d-7c38-4eff-9ea8-6dd5f23b111a",
    "conditions": {
        "principal": {
            "group": {
                "operator": "in",
                "value": [
                    "Finance"
                ]
            }
        },
        "action": {
            "operator": "in",
            "value": [
                "authenticate"
            ]
        },
        "channel": {
            "operator": "in",
            "value": [
                "web"
            ]
        }
    },
    "result": {
        "effect": "allow",
        "reAuthenticate": "false",
        "obligations": [
            "PASSWORD:1",
            "FIDO:2",
            "SECURITYKEY:2"
        ],
        "msg": "This condition is for the Finance Group PW+FIDO or PW+SECURITYKEY on the Web Channels. Login
 with bsteer."
    }
},
{
```

```
  "id": "5698d509-d981-4d5b-8c42-2a23e94a9dbb",
  "conditions": {
      "principal": {
          "group": {
              "operator": "in",
              "value": [
                  "Employees"
              ]
          }
      },
      "action": {
          "operator": "in",
          "value": [
              "authenticate"
          ]
      },
      "channel": {
          "operator": "in",
          "value": [
              "mobile"
          ]
      }
  },
  "result": {
      "effect": "allow",
      "reAuthenticate": "false",
      "obligations": [
          "PASSWORD:1",
          "FIDO:2",
          "SECURITYKEY:2"
      ],
      "msg": "This condition is for the Employees Group PW+FIDO+SECURITYKEY on the Mobile Channel. Login with
 nbruce"
  }
}
],
"createdDate": "2023-09-06 08:22 AM UTC"
},
{
"policyId": "d35a6936-4b11-4d3f-b822-157f17f7bb0b",
"policyName": "AdminLoginPolicy",
"description": "Default policy for Tenant Admin Console",
"status": "active",
"matchAnyApp": false,
"apps": [
{
  "name": "AdminConsole",
  "id": "$TENANT_ADMINCONSOLE"
},
{
  "name": "DefaultTenantClient",
  "id": "61b286b7-8c9a-4f67-9e87-6dd86dee833e"
}
```

```
],
"rules": [
{
  "id": "bf493373-7128-4d58-a711-82de3328ac80",
  "conditions": {
      "channel": {
          "operator": "in",
          "value": [
              "web"
          ]
      }
  },
  "result": {
      "effect": "allow",
      "reAuthenticate": "false",
      "obligations": [
          "PASSWORD:1"
      ],
      "msg": "This policy is for admin login"
  }
}
],
"createdDate": "2023-09-06 08:22 AM UTC",
"updatedBy": {
"principalType": "CLIENT",
"principalId": "8b3bef5b-b8ab-4ba8-b56b-0a2807694b5a",
"principalName": "DefaultTenantClient"
},
"updatedDate": "2023-09-22 10:27 AM UTC",
"preventedOperations": "Delete"
}
]
```


For sample request and response payloads of all the mentioned URLs, refer to Security API Swagger documentation.

Performing GET operation on Admin Roles

The following example shows the GET request and response of the ***Admin Roles Delegator API***.

Perform a GET request using the following Authorization API:

```
https://${SSP_FQDN}/<tenantName>/admin/v1/AdminRolesDelegator
```

The response fetches all the embedded policies of the Admin roles in IDSP. Response:


```
[

{

"policyName": "SystemAdminPolicy",

"description": "Role for the System Administrator",

"displayName": "System Administrator",
```

```
"status": "active",
"isGrantableClientOnly": false,
"apps": [
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
"rules": [
{
  "conditions": {
      "principal": {
          "clientApp": {
              "operator": "in",
              "value": [
                  "$SYSTEM_CLIENT_APP_ID"
              ]
          }
      }
  },
  "result": {
      "msg": "This is System admin authorization policy",
      "privileges": [
          "urn:iam:g.tenant",
          "urn:iam:g.crosstenant",
          "urn:iam:g.settings",
          "urn:iam:t.configs",
```

```
            "urn:iam:t.idpconfigs",
            "urn:iam:t.ldapconfigs",
            "urn:iam:t.idstoreconfigs",
            "urn:iam:t.authzpolicies",
            "urn:iam:t.authpolicies",
            "urn:iam:t.authenticate",
            "urn:iam:t.apps",
            "urn:iam:t.certs",
            "urn:iam:t.secrets",
            "urn:iam:t.secretsgetter",
            "urn:iam:t.signingkey",
            "urn:iam:t.adminroles",
            "urn:iam:g.authzpolicywriter",
            "urn:iam:t.idstorebrowser"
        ]
    }
}
]
},
{
"policyName": "InfraPolicy",
"description": "Role for the Infrastructure Administrator",
"displayName": "Infrastructure Administrator",
"status": "active",
"isGrantableClientOnly": true,
"apps": [
```

```
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
"rules": [
{
  "conditions": {
      "principal": {
          "clientApp": {
              "operator": "in",
              "value": [
                  "$INFRA_CLIENT_APP_ID"
              ]
          }
      }
  },
  "result": {
      "msg": "This is Infra authorization policy",
      "privileges": [
          "urn:iam:t.apps",
          "urn:iam:t.ldapconfigs",
          "urn:iam:t.idstoreconfigs",
          "urn:iam:t.idpconfigs",
          "urn:iam:t.authpolicies",
          "urn:iam:t.authzpolicies",
          "urn:iam:g.settings",
```

```
          "urn:iam:t.configs",
          "urn:iam:t.signingkey"
      ]
  }
}
]
},
{
"policyName": "SigninClient",
"description": "Role for the Default Signin Client",
"displayName": "Signin Client",
"status": "active",
"isGrantableClientOnly": true,
"apps": [
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
"rules": [
{
  "conditions": {
      "principal": {
          "clientApp": {
              "operator": "in",
              "value": [
```

```
                    "$SIGNINCLIENT"
                ]
            }
        }
    },
    "result": {
        "msg": "Default Signin Client authorization policy",
        "privileges": [
            "urn:iam:t.authenticate",
            "urn:iam:g.crosstenant"
        ]
    }
}
]
},
{
"policyName": "SystemAdminConsole",
"description": "Role for the Default System Administrator Console",
"displayName": "System Administrator Console",
"status": "active",
"isGrantableClientOnly": true,
"apps": [
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
```

```
"rules": [
{
  "conditions": {
      "principal": {
          "clientApp": {
              "operator": "in",
              "value": [
                  "$SYSTEM_ADMINCONSOLE"
              ]
          }
      }
  },
  "result": {
      "msg": "Default System Admin Console authorization policy",
      "privileges": [
          "urn:iam:g.crosstenant"
      ]
  }
}
]
},
{
"policyName": "TenantAdminConsole",
"description": "Role for the Default Tenant Administrator Console",
"displayName": "Tenant Administrator Console",
"status": "active",
```

```
"isGrantableClientOnly": true,
"apps": [
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
"rules": [
{
  "conditions": {
      "principal": {
          "clientApp": {
              "operator": "in",
              "value": [
                  "$TENANT_ADMINCONSOLE"
              ]
          }
      }
  },
  "result": {
      "msg": "Default Tenant Admin Console authorization policy",
      "privileges": [
          "urn:iam:g.crosstenant"
      ]
  }
}
]
```

```
},
{
"policyName": "TenantAdminPolicy",
"description": "Role for the Tenant Administrator",
"displayName": "Tenant Administrator",
"status": "active",
"isGrantableClientOnly": false,
"apps": [
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
"rules": [
{
  "conditions": {
      "principal": {
          "clientApp": {
              "operator": "in",
              "value": [
                  "$DEFAULT_TENANT_CLIENT_APP_ID"
              ]
          }
      }
  },
  "result": {
```

```
      "msg": "This is Tenant Admin authorization policy",
      "privileges": [
          "$TENANT_ADMIN_PRIVILEGES"
      ]
  }
}
]
},
{
"policyName": "AuthClientPolicy",
"description": "Role for the Authentication Client",
"displayName": "Authentication Client",
"status": "active",
"isGrantableClientOnly": true,
"apps": [
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
"rules": [
{
  "conditions": {
      "principal": {
          "clientApp": {
              "operator": "in",
              "value": [
```

```
                    "$DEMO_CLIENT_APP_ID"
                ]
            }
        }
    },
    "result": {
        "msg": "This policy allows clients to call authentication APIs",
        "privileges": [
            "$AUTHCLIENT_PRIVILEGES"
        ]
    }
}
]
},
{
"policyName": "TenantAuditorPolicy",
"description": "Role for the Tenant Auditor",
"displayName": "Tenant Auditor",
"status": "active",
"isGrantableClientOnly": false,
"apps": [
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
```

```
"rules": [
{
  "result": {
      "msg": "This is Tenant Auditor authorization policy",
      "privileges": [
          "urn:iam:t.all",
          "urn:iam:t.auditonly"
      ]
  }
}
]
},
{
"policyName": "IntegrationAdminPolicy",
"description": "Role for the Integration Administrator",
"displayName": "Integration Administrator",
"status": "active",
"isGrantableClientOnly": false,
"apps": [
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
"rules": [
{
  "result": {
```

```
      "msg": "This is Integration Admin authorization policy",
      "privileges": [
          "urn:iam:t.apps",
          "urn:iam:t.idpconfigs",
          "urn:iam:t.idstoreconfigs",
          "urn:iam:t.certs",
          "urn:iam:t.secrets",
          "urn:iam:t.idpdiscoverypolicies",
          "urn:iam:t.idstorebrowser"
      ]
  }
}
]
},
{
"policyName": "IntegrationAuditorPolicy",
"description": "Role for the Integration Auditor",
"displayName": "Integration Auditor",
"status": "active",
"isGrantableClientOnly": false,
"apps": [
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
```

```
"rules": [

{

  "result": {

      "msg": "This is Integration Auditor authorization policy",

      "privileges": [

          "urn:iam:t.apps",

          "urn:iam:t.idpconfigs",

          "urn:iam:t.idstoreconfigs",

          "urn:iam:t.certs",

          "urn:iam:t.secrets",

          "urn:iam:t.idpdiscoverypolicies",

          "urn:iam:t.idstorebrowser",

          "urn:iam:t.auditonly"

      ]

  }

}

]

},

{

"policyName": "ConfigurationAdminPolicy",

"description": "Role for the Configuration Administrator",

"displayName": "Configuration Administrator",

"status": "active",

"isGrantableClientOnly": false,

"apps": [

{

  "name": "ssp",
```

```
  "id": "$SSP_APP"
}
],
"rules": [
{
  "result": {
      "msg": "This is Configuration Admin authorization policy",
      "privileges": [
          "urn:iam:t.configs",
          "urn:iam:t.authpolicies",
          "urn:iam:t.idstoreconfigs",
          "urn:iam:t.certs",
          "urn:iam:t.secrets",
          "urn:iam:t.signingkey",
          "urn:iam:t.idstorebrowser",
          "urn:iam:t.adminroles"
      ]
  }
}
]
},
{
"policyName": "ConfigurationAuditorPolicy",
"description": "Role for the Configuration Auditor",
"displayName": "Configuration Auditor",
"status": "active",
```

```
"isGrantableClientOnly": false,
"apps": [
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
"rules": [
{
  "result": {
      "msg": "This is Configuration Auditor authorization policy",
      "privileges": [
          "urn:iam:t.configs",
          "urn:iam:t.authpolicies",
          "urn:iam:t.idstoreconfigs",
          "urn:iam:t.certs",
          "urn:iam:t.secrets",
          "urn:iam:t.signingkey",
          "urn:iam:t.idstorebrowser",
          "urn:iam:t.adminroles",
          "urn:iam:t.auditonly"
      ]
  }
}
]
},
{
```

```
"policyName": "CredentialAdminPolicy",
"description": "Role for the Credential Administrator",
"displayName": "Credential Administrator",
"status": "active",
"isGrantableClientOnly": false,
"apps": [
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
"rules": [
{
  "result": {
      "msg": "This is Credential Administrator policy",
      "privileges": [
          "urn:iam:t.userdata",
          "urn:iam:t.session"
      ]
  }
}
]
},
{
"policyName": "IdentityAdminPolicy",
"description": "Role for the Identity Administrator",
```

```
"displayName": "Identity Administrator",

"status": "active",

"isGrantableClientOnly": false,

"apps": [

{

  "name": "ssp",

  "id": "$SSP_APP"

}

],

"rules": [

{

  "result": {

      "msg": "This is Identity Admin authorization policy",

      "privileges": [

          "urn:iam:t.users",

          "urn:iam:t.groups",

          "urn:iam:t.session"

      ]

  }

}

]

},

{

"policyName": "IdstoreAdminPolicy",

"description": "Role for the IDStore Administrator",

"displayName": "Idstore Administrator",

"status": "active",
```

```
"isGrantableClientOnly": false,
"apps": [
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
"rules": [
{
  "result": {
      "msg": "This is Idstore Admin authorization policy",
      "privileges": [
          "urn:iam:t.idstoreadmin"
      ]
  }
}
]
},
{
"policyName": "AccountAdminPolicy",
"description": "Role for the Account Administrator",
"displayName": "Account Administrator",
"status": "active",
"isGrantableClientOnly": false,
"apps": [
{
```

```
  "name": "ssp",
  "id": "$SSP_APP"
}
],
"rules": [
{
  "result": {
      "msg": "This is Account Admin authorization policy",
      "privileges": [
          "urn:iam:t.accountadmin"
      ]
  }
}
]
},
{
"policyName": "IdentityClientPolicy",
"description": "Role for the Identity Client",
"displayName": "Identity Client",
"status": "active",
"isGrantableClientOnly": true,
"apps": [
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
```

```
"rules": [
{
  "result": {
      "msg": "This policy allows clients to call identity APIs",
      "privileges": [
          "urn:iam:t.identityclient"
      ]
  }
}
]
},
{
"policyName": "RiskClientPolicy",
"description": "Role for the Risk Client",
"displayName": "Risk Client",
"status": "active",
"isGrantableClientOnly": true,
"apps": [
{
  "name": "ssp",
  "id": "$SSP_APP"
}
],
"rules": [
{
  "result": {
```

```
      "msg": "This policy allows clients to call risk APIs",
      "privileges": [
          "urn:iam:t.risk"
      ]
   }
}
]
}
]
```

Performing POST operation on Admin Roles

The following example shows the POST request and response of the ***Admin Roles Delegator API***.

Perform a POST request using the following Authorization API:

```
https://${SSP_FQDN}/<tenantName>/admin/v1/AdminRolesDelegator
```

Request: The following example shows the payload where the admin is requesting to add a user who is known as "droth".

```
{
"id": "d9122c83-7ec5-4450-a31c-7b99c88a5e51",
"principal": {
"user": {
"operator": "in",
"value": [
  "droth"
]
}
}
}
```

Response: The response returns the name of the role which has been updated with a new user (droth).

```
{
"id": "d9122c83-7ec5-4450-a31c-7b99c88a5e51",
"name": "ConfigurationAdminPolicy",
"displayName": "Configuration Administrator",
"description": "Manage security Settings, Signing Keys, Authentication Policies. Also manage Identity Stores,
 Certificates, Secrets.",
"principal": {
"user": {
"operator": "in",
"value": [
  "droth",
"nbruce"
]
}
```

```
},
"isGrantableClientOnly": false
}
```

Performing DELETE operation on Admin Roles

The following example shows the DELETE request and response of ***Admin Roles Delegator API***.

Perform a DELETE request using the following Authorization API:

```
https://${SSP_FQDN}/<tenantName>/admin/v1/AdminRolesDelegator
```

API Request: In the following example, the admin is revoking the access from the user, "droth":

```
{
"id": "d9122c83-7ec5-4450-a31c-7b99c88a5e51",
"principal": {
"user": {
"operator": "in",
"value": [
  "droth"
]
}
}
}
```

API Response: The response displays the role from which the user (droth) has been revoked.

```
{
"id": "d9122c83-7ec5-4450-a31c-7b99c88a5e51",
"name": "ConfigurationAdminPolicy",
"displayName": "Configuration Administrator",
"description": "Manage security Settings, Signing Keys, Authentication Policies. Also manage Identity Stores,
 Certificates, Secrets.",
"principal": {
"user": {
"operator": "in",
"value": [
"nbruce"
]
}
},
"isGrantableClientOnly": false
}
```

Examples of Managing System Admin Operations

Follow these steps:

You must configure the LDAP Directory containing your system administrators and must grant the System Admin Role to the user or group.

1. Run the following API to create an LDAP Configuration to connect with the LDAP Id store and authenticate users:
   Request:

   ```
   curl -L -X POST 'https://iam.example.com/system/admin/v1/LDAPconfigs' \
   -H 'Content-Type: application/json' \
   -H 'Authorization: Bearer <token>' \
   --data-raw '{
   ```

```
"configName": "SytemAdminStore",
"configDescription": "SystemAdministrator Store",
"sslEnabled": "NO",
"ldapQueryStart": "(&(cn=",
"ldapQueryEnd": ")(objectClass=*))",
"attributes": "uid,displayName,telephoneNumber,mail,memberof",
"objectClasses": "string",
"hostName": "cadir-ssp-symantec-dir.idstore.svc.cluster.local",
"portNumber": "30306",
"baseDN": "o=democorp,c=us",
"userDN": "cn=DominicMAJOR,ou=Administration,ou=Corporate,o=democorp,c=US",
"bindPassword": "password",
"attributeMapping":
 "name=cn,family_name=sn,given_name=givenName,middle_name=givenName,nickname=displayName,preferred_username=displayN
"passwordResetURL": "http://userPasswordResetURL"
}'
```

Response:

```
{
"tenantId": "cce2a696-5941-11ea-8e2d-0242ac130003",
"configName": "SytemAdminStore",
"configDescription": "SystemAdministrator Store",
"sslEnabled": "NO",
"certificate": null,
"certificateAlias": null,
"ldapQueryStart": "(&(cn=",
"ldapQueryEnd": ")(objectClass=*))",
"attributes": "uid,telephoneNumber,employeeType,mail,displayName,givenName,cn,sn,memberof,title",
"objectClasses": "string",
"hostName": "cadir-ssp-symantec-dir.idstore.svc.cluster.local",
"portNumber": "30306",
"baseDN": "o=democorp,c=us",
"userDN": "cn=DominicMAJOR,ou=Administration,ou=Corporate,o=democorp,c=US",
"bindPassword": "/OTqCj2yPWVvhl6AjpJ/DQ==",
"attributeMapping":
 "name=cn,family_name=sn,given_name=givenName,middle_name=givenName,nickname=displayName,preferred_username=displayN
"passwordResetURL": "http://userPasswordResetURL",
"createdDateTime": "2020-06-11 09:34 AM UTC",
"updatedDateTime": "2020-06-11 09:34 AM UTC",
"createdBy": {
"principalType": "CLIENT",
"principalId": "8df411c3-466a-46b9-a8bd-21a655a5314c",
"principalName": "systemclient"
},
"ldapConfigId": "9e87ff23-b120-4322-b2aa-9d5f24a0a6d3"
  }
```

2. Grant a System Admin Role to a user or a group in the LDAP Directory.
   Get the system app id to authzclient system policy:
   Request:

```
curl -L -X GET 'https://iam.example.com/system/admin/v1/Apps' \
-H 'Authorization: Bearer <token>'
```

   Response:

```
[
```

```
{
"status": "active",
"name": "infraclient",
"description": "client performs infra-level operations",
"clientType": "CONFIDENTIAL",
"allowedGrantTypes": [
"client_credentials"
],
"appId": "55742ded-92c1-43a6-8659-d6250081e3f6",
"createdBy": null,
"createdDateTime": "2020-06-10 05:38 AM UTC",
"secret": "00f16009-b931-4a41-a8b6-41644c7061f1",
"clientId": "8089b534-b97d-499f-a1c6-67eccdca07ed"
},
{
"status": "active",
"name": "systemconsoleclient",
"description": "Client for SystemConsole",
"clientType": "CONFIDENTIAL",
"allowedGrantTypes": [
"client_credentials"
],
"allowedOperations": [
"onbehalfof"
],
"appId": "534b4af3-f53c-4a01-a4e3-2103f2cd5d0e",
"createdBy": null,
"createdDateTime": "2020-06-10 05:36 AM UTC",
"secret": "36e7cd48-18b4-4d99-b331-ffdae5da528f",
"clientId": "031d0553-5469-4484-ba85-4f7847532d39"
},
{
"status": "active",
"name": "loginappclient",
"description": "Client for LoginApp",
"clientType": "CONFIDENTIAL",
"allowedGrantTypes": [
"client_credentials"
],
"appId": "d2c34e3f-3a15-4e16-9fde-52b25194bec7",
"createdBy": null,
"createdDateTime": "2020-06-10 05:36 AM UTC",
"secret": "299d9c4c-23d2-4f00-bbb1-34920900b832",
"clientId": "5f0ac4dd-9f6e-4d6a-bd89-6a4075070e1a"
},
{
"status": "active",
"name": "systemclient",
"description": "client performs various system level operations",
"clientType": "CONFIDENTIAL",
"allowedGrantTypes": [
"client_credentials",
"urn:ietf:params:oauth:grant-type:jwt-bearer"
```

```
],
"appId": "256fe4b4-1d37-4c30-91fa-f8d8d1262e47",
"createdBy": null,
"createdDateTime": "2020-06-10 05:38 AM UTC",
"secret": "4180b346-e3dd-4d1d-9203-5e023db33db9",
"clientId": "8df411c3-466a-46b9-a8bd-21a655a5314c"
},
{
"status": "active",
"name": "adminconsoleclient",
"description": "Client for AdminConsole",
"clientType": "CONFIDENTIAL",
"allowedGrantTypes": [
"client_credentials"
],
"allowedOperations": [
"onbehalfof"
],
"appId": "c7a80f17-6ac7-4d01-9c6a-547b1c823060",
"createdBy": null,
"createdDateTime": "2020-06-10 05:36 AM UTC",
"secret": "a6d6c87c-4eea-4a88-9023-1795c8f99f1d",
"clientId": "6a240801-5506-4a83-b5b7-06dbad630889"
}
]
```


3. Update the Authorization (AuthZ) policy with the system admin group:
   Request:

```
curl -L -X PATCH 'https://iam.example.com/system/admin/v1/AuthZPolicies/27d6ac59-387c-4311-81eb-
c6632c2abe47' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <token>' \
--data-raw '{
"rules": [
{
"conditions": {
  "principal": {
      "clientApp": {
          "value": [

              "256fe4b4-1d37-4c30-91fa-f8d8d1262e47"
          ]
      }

  ,
      "group": {
          "value": [
              "Admin"
          ]
      }
}
}
}
```

```
]
}'
```

Response:


```
{
"policyId": "27d6ac59-387c-4311-81eb-c6632c2abe47",
"policyName": "SystemAdminPolicy",
"description": "Policy for System Admin",
"status": "active",
"matchAnyApp": false,
"apps": [
{
"name": "ssp",
"id": "3690ac36-c805-43ef-8318-627023b16d64"
}
],
"rules": [
{
"conditions": {
  "principal": {
      "group": {
          "operator": "in",
          "value": [
              "Admin"
          ]
      },
      "clientApp": {
          "operator": "in",
          "value": [
              "256fe4b4-1d37-4c30-91fa-f8d8d1262e47"
          ]
      }
  }
},
"result": {
  "msg": "This is System admin authorization policy",
  "privileges": [
      "urn:iam:g.crosstenant",
      "urn:iam:g.settings",
      "urn:iam:t.ldapconfigs",
      "urn:iam:t.authzpolicies",
      "urn:iam:t.authenticate",
      "urn:iam:t.apps",
      "urn:iam:t.certs",
      "urn:iam:t.signingkey"
  ]
}
}
],
"createdDate": "2020-06-10 05:36:21",
"updatedDate": "2020-06-10 05:36:21"
}
```


Once the Admin role has been updated, it is now possible to invoke APIs in the context of a specific user.

Steps to grant "$AUTHCLIENT_PRIVILEGES" privileges to a client app, click here.

1. Perform a GET https://${SSP_FQDN}/<tenantName>/admin/v1/AuthZPolicies to retrieve all the authorization policies configured in the solution.
2. Look for the "AuthClientPolicy" and copy the policy Id and the rule Id of the conditions>principal>clientApp. A sample of "AuthClientPolicy" is shown in the following example:

```
{
"policyId": "fac76b22-7555-4cf2-86f0-6a6028acba00",
"policyName": "AuthClientPolicy",
"description": "Policy for Authentication Client",
"status": "active",
"matchAnyApp": false,
"apps": [
{
  "name": "SSP",
  "id": "$SSP_APP"
}
],
"rules": [
{
  "id": "2d7978fb-5bed-4a26-90b3-cc3788c27d26",
  "conditions": {
      "principal": {
          "clientApp": {
              "operator": "in",
              "value": [
                  "42753c4a-08be-4db6-a589-09fd883c1efe"
              ]
          }
      }
  },
  "result": {
      "msg": "This policy allows clients to call authentication APIs",
      "privileges": [
          "$AUTHCLIENT_PRIVILEGES"
      ]
  }
}
],
"createdDate": "2021-04-28 05:15:49",
"preventedOperations": "Delete"
}
```

3. Go to the PATCH https://${SSP_FQDN}/<tenantName>/admin/v1/AuthZPolicies/<REPLACE-BY-POLICY-ID>
   a. Replace the policy id in the header. Use the policy Id of the AuthClientPolicy.
   b. In the following payload, replace the value of {{RuleId}} from the preceding authClient Policy.
   c. Replace the value of the app Id that you have got after creating the app.

```
{
"op": "add",
"path": "/rules/[id eq \"{{RuleId}}\"]/conditions/principal/clientApp",
"value": "<REPLACE-BY-CLIENT-APPID>"
},
```

4. The response grants "$AUTHCLIENT_PRIVILEGES" to the application.

```
{
"policyId": "fac76b22-7555-4cf2-86f0-6a6028acba00",
"policyName": "AuthClientPolicy",
"description": "Policy for Authentication Client",
"status": "active",
"matchAnyApp": false,
"apps": [
{
"name": "SSP",
"id": "$SSP_APP"
}
],
"rules": [
{
"id": "2d7978fb-5bed-4a26-90b3-cc3788c27d26",
"conditions": {
  "principal": {
      "clientApp": {
          "operator": "in",
          "value": [
              "9a53235f-e64f-4cb3-b0a9-bffa4e8367e7",
              "42753c4a-08be-4db6-a589-09fd883c1efe"
          ]
      }
  }
},
"result": {
  "msg": "This policy allows clients to call authentication APIs",
  "privileges": [
      "$AUTHCLIENT_PRIVILEGES"
  ]
}
}
],
"createdDate": "2021-04-28 05:15:49",
"updatedBy": {
"principalType": "CLIENT",
"principalId": "b57baefb-cea4-4459-86ee-c19fcefc8c45",
"principalName": "defaulttenantclient"
},
"updatedDate": "2021-05-14 09:57:07",
"preventedOperations": "Delete"
}
```


Performing GET operation on My Admin Roles

- Perform a GET request using the following Authorization Admin API:

  NOTE
  Before performing this step, ensure that you have retrieved the **User Access Token**.

  ```
  GET https://${SSP_FQDN}/<tenantName>/admin/v1/MyAdminRoles
  ```

  The response fetches the list of admin capabilities that are associated with the given user.


  ```
  [
  ```


```
{
"id": "0b3d26ac-6b32-42c6-b3ef-7c8cae61883f",
"name": "TenantAdminPolicy",
"displayName": "Tenant Administrator",
"description": "Manage everything in a tenant",
"privileges": [
"urn:iam:t.all"
]
},
{
"id": "1b38a3ef-3dd8-49be-9c86-c3acc92ecb68",
"name": "CredentialAdminPolicy",
"displayName": "Credential Administrator",
"description": "Manage user account credentials, trusted devices, known devices. Also manage user
 sessions.",
"privileges": [
"urn:iam:t.userdata",
"urn:iam:t.session"
]
}
]
```


Application Roles and Privileges

This topic explains how applications can retrieve user roles and privileges through multiple integration protocols.

Applications serve multiple types of consumers, including human users and non-human entities. Each consumer requires specific authorization and personalization to ensure the application behaves appropriately for their assigned access level.

The system determines the appropriate access level using access control concepts including:

- App Roles
- App Privileges
- Enterprise Groups
- User Attributes
- Business Rules
- Other access control mechanisms

Applications can discover app-specific access control concepts (such as App Roles and App Privileges) through multiple integration patterns. The application's architecture determines which integration pattern to choose from the available options.

Application Roles and Privileges

App Integration Approach	App Integration Protocol	App Cardinality	App Roles	App Privileges

User attributes returned by SCIM contains SCIM-standard roles & entitlement attributes	SCIM /Users	All apps	<pre>"roles": [
			{
			"display": "Sell Role",
			"primary": false,
			"type": "",
			"value": "Bank Sell Policy" → Role Name
			},
			{
			"display": "Buy Role",
			"primary": false,
			"type": "",
			"value": "Bank Buy Policy"
			},
			{
			"display": "Bank Deposit app role",
			"primary": false,
			"type": "",
			"value": "Bank Deposit"
			},
			{
			"display": "Bank Withdraw app role",
			"primary": false,
			"type": "",
			"value": "Bank Withdraw"
			}</pre>
				<pre>"entitlements": [
				{
				"display": "can sell shares",
				"primary": false,
				"type": "",
				"value": "sell" → privilege
				},
				{
				"display": "can buy shares",
				"primary": false,
				"type": "",
				"value": "buy"
				},
				{
				"display": "Deposit money",
				"primary": false,
				"type": "",
				"value": "deposit"
				},
				{
				"display": "Withdraw money",
				"primary": false,
				"type": "",
				"value": "withdraw"
				}
				]</pre>

Custom Claims in SAML-based security tokens	SAML Assertion	Client app representing connecting app	<saml2:Attribute Name="appRoles" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"> <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xsd:string">Bank Sell Policy</saml2:AttributeValue> <saml2:AttributeValue. → Role Name xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xsd:string">Bank Buy Policy</saml2:AttributeValue> </saml2:Attribute> Custom Claim: • name: clientAppRoles • value: $[clientAccess.appRoles]	<saml2:Attribute Name="appPrivileges" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"> <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xsd:string">sell</saml2:AttributeValue> → privilege <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xsd:string">buy</saml2:AttributeValue> </saml2:Attribute> Custom Claim: • name: clientAppPrivileges • value: $[clientAccess.appPrivileges]
Custom Claims in JWT-based security tokens	OpenID Identity Token OAuth Access Token	Client app representing connecting app	"**clientAppRoles**": ["Bank Sell Policy", → Role Name "Bank Buy Policy"] Custom Claim: • name: clientAppRoles • value: $[clientAccess.appRoles]	"**clientAppPrivileges**": ["sell", → privilege "buy"] Custom Claim: • name: clientAppPrivileges • value: $[clientAccess.appPrivileges]
		Resource app representing resource mapped to by requested scopes	"**rsAppRoles**": ["Bank Sell Policy", → Role Name "Bank Buy Policy"] Custom Claim: • name: rsAppRoles • value: $[resourceAccess.appRoles]	"**rsAppPrivileges**": ["sell", → privilege "buy"] Custom Claim: • name: rsAppPrivileges • value: $[resourceAccess.appPrivileges]
Claims returned by Identity Provider	OpenID /Userinfo	All apps	N/A	N/A

HTTP API	HTTP REST API	All apps or specific app	/access/v1/MeAppRoles?app=DemoClient *returns* `[{ "appId": "a16f9414-9714-4bf3-bdc7-5d8637fb0452", "appName": "DemoClient", → App Name "roleId": "52525404-8faf-4025-bb75-00af0f12f87e", "roleName": "Bank Sell Policy", → Role Name "roleDescription": "Sell @ Bank", "displayName": "Sell Role", "effect": "grant", "privileges": ["sell". → privilege] },` `{ "appId": "a16f9414-9714-4bf3-bdc7-5d8637fb0452", "appName": "DemoClient", "roleId": "45bbe5a9-82bd-4d9d-91a3-4e24e9933297", "roleName": "Bank Buy Policy", "roleDescription": "Buy @ Bank", "displayName": "Buy Role", "effect": "grant", "privileges": ["buy"] }]`	/access/v1/MeAppPrivileges?app=DemoClient *returns* *["sell", → privilege "buy"]*

Functional Administration

Describes how tenant administrators (or select administrators with higher privileges) define Administrative Roles, Policies, User Schemas, Scoping Constraints, and assignment of privileges to Functional Administrators.

Functional Administration is the process during which the overall structure and security model is defined that will allow Business Administration to then perform its functions within this structure. Defining a deployment's Resource Authorization model and how the model will be deployed becomes a pre-requisite for functional administration. For Symantec Identity Security Platform, functional administration is the process in which tenant administrators (or select administrators with higher privileges) define administrative roles, policies, user schemas, scoping constraints, and assignment of privileges to functional administrators. This process may include certain resources that need tighter administrative control such as Groups, Applications, Certificates, and Secrets.

Transitioning to Business Administration

One critical aspect of functional administration is the ability to transition rights and responsibilities to business administrators. The transition typically involves granting initial access to administrators responsible for certain operations of the business such as managing particular user populations, or managing sets of applications and Identity Providers needed for their business responsibilities.

The transition from functional administration to business administration then allows business administrators to scale their administration capacity by delegating their rights and responsibilities. Under the delegation model, higher privileges typically required during functional administration are not required. Delegation allows you to delegate the privileges that you have without requiring special privileges. This allows business administration to scale up or down as needed.

- Role and Resource Access
- Delegation

Role and Resource Access

Describes how to use the criteria and scoping constraints of Resource Authorization to assign roles and grant access to resources.

Role and resource access is managed by the administrative grant role capability or by Resource Authorization delegation.

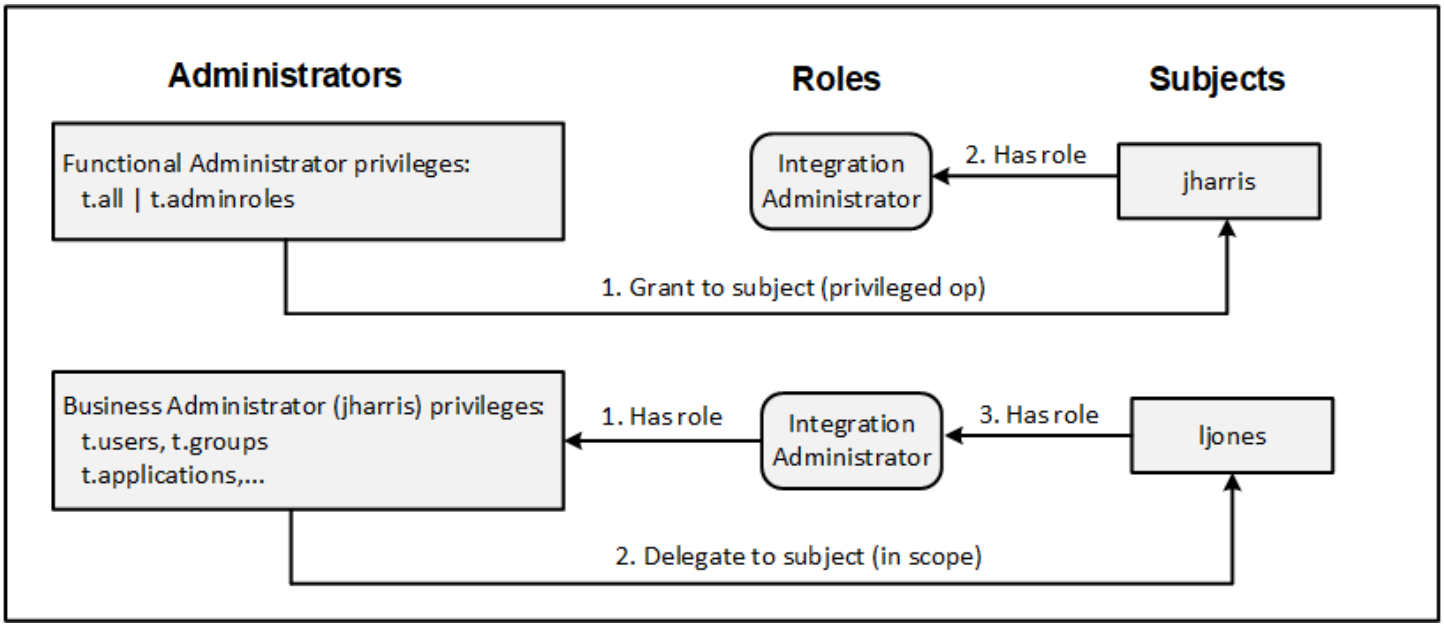
Granting Role Access

The grant role capability is restricted to privileged administrators. Resource Authorization introduces a delegation capability which allows administrators to delegate the roles that they have without requiring any higher privilege.

Administrator role delegation does impose enforcement over to whom an administrator can delegate privileges. The subject of the delegation must be within the admin scope of the administrator. However, administrators with higher privileges and that invoke delegation operations are not subject to these admin scope restrictions.

The role access options are shown in the following diagram:

Figure 5: Role Access Options



Granting Resource Access

The Resource Authorization capabilities include the ability for administrators to grant access to resources. This ability applies to the following resource types:

- Groups
- Applications
- Identity Providers
- Identity Provider Discovery Policies
- Certificates
- Secrets
- Authorization Policies

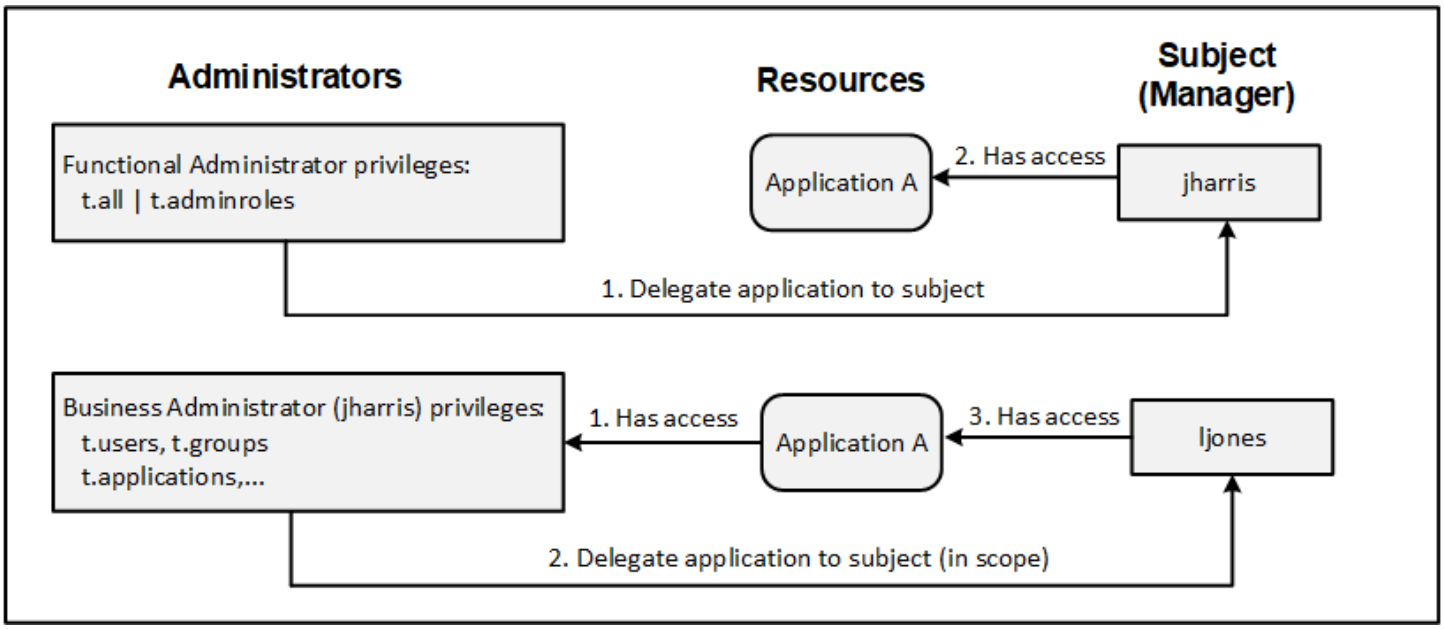
The granting of resources supports the `managedby` access control. When granting access to a resource, the subject is then added to the list of managers for the resource. When resource access is restricted by `managedby` access controls, only managers that are added through delegation are allowed access.

Resource delegation does impose enforcement over to whom an administrator can delegate privileges. The subject of the delegation must be within the admin scope of the administrator. However, administrators with higher privileges that invoke delegation operations are not subject to these admin scope restrictions. The higher privileges that are required for the grant role capability must include one of the following scopes:

- urn:iam:t.all
- urn:iam:t.adminroles

The resource access options are shown in the following diagram:

Figure 6: Resource Access Options



Delegation

Describes how delegation allows you to assign the roles or privileges of one entity to another entity.

Delegation is the ability to grant your roles or resources to others. This ability allows business administrators to scale administrative capability up or down as the business needs change.

There are two types of delegations: Admin Role Delegation and Resource Delegation. Adding or removing a delegation is subject to administrative scope checks. When attempting to delegate a role or a resource to a subject, the administrator must have administrative scope over the subject or else the delegation is denied. However, if the administrator initiated

the delegation attempts to remove a delegation, no administrative scope check is required; you can remove what you delegated.

However, if an administrator has the following privileges, no administrative scope checks are enforced:

- urn:iam:t.all
- urn:iam:t.adminroles

You can manage role and resource delegations (view, add, or remove) directly through the Delegation APIs. See Delegations APIs.

> **NOTE**
> You can also view or remove the roles and resources that are assigned by you or to you from the *Profiles* page of the Admin Console. See Profiles.

Delegations Definitions

The following list provides Symantec Identity Security Platform delegation-related terms and their definitions.

Principal (grantor) — The user (administrator) or client doing the delegation

Subject (grantee) — The user (administrator) or client/group to which access is being delegated

Resource Type — The type of the IDSP resource. The Resource Type must be one of the following options:

- Admin Role
- Group
- Application
- Identity Providers
- Identity Provider Discovery Policies
- Certificates
- Secrets
- Authorization Policies

Resource — The ID of the role or resource

Installing

Includes links to deployment overview, deployment sizing, cluster environment preparation, database considerations, deployment procedures, configuring admin access, upgrade instructions, and uninstallation guidelines for the Symantec Identity Security Platform solution.

IDSP enables the deployment of its instances in Kubernetes to manage containerized applications in a clustered environment. The solution enables container replication, scaling, high availability and simplifies the process of deployment, configuration, versioning, and updates.

To know more about the deployment of IDSP on Kubernetes, see the following topics:

- Overview of IDSP Deployment
- Sizing the Deployment
- Preparing the Infrastructure Before Deploying the Solution
- Database Considerations
- Deploying the Services
- Configuring Administrative Access
- Upgrading IDSP
- Uninstalling the Solution and Enclave Services

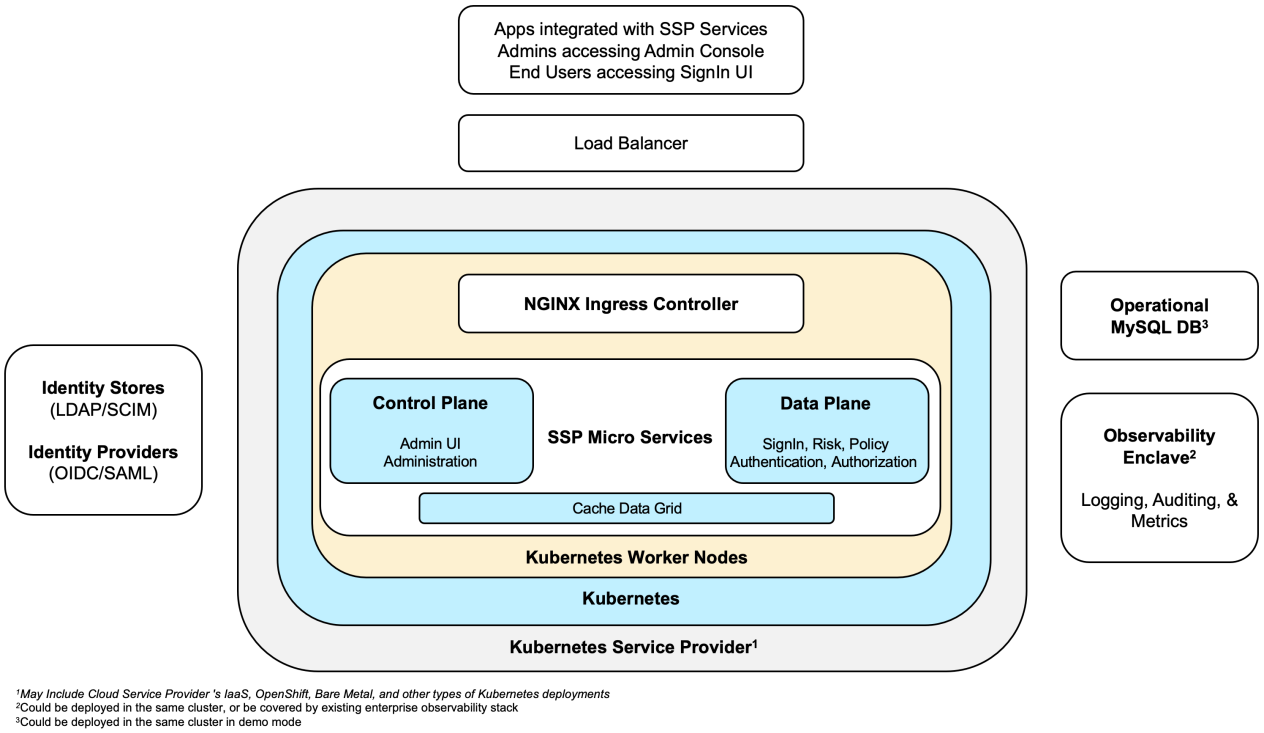
Overview of Symantec Identity Security Platform Deployment

Details the Kubernetes deployment of Symantec Identity Security Platform, covering its architectural components, namespace organization, and the roles and dependencies of its microservices.

This topic includes the following sections:

- Overview of IDSP Deployment in a Kubernetes Cluster
- IDSP Cluster Information
- IDSP Micro-Services Descriptions

The following image provides an overview of how IDSP is deployed in a Kubernetes Cluster environment.



[1]*May Include Cloud Service Provider 's IaaS, OpenShift, Bare Metal, and other types of Kubernetes deployments*
[2]Could be deployed in the same cluster, or be covered by existing enterprise observability stack
[3]Could be deployed in the same cluster in demo mode

IDSP Cluster Information

IDSP solution is composed of multiple microservices, each deployed as a container within a Kubernetes Pod. Readiness probes are configured for each microservice to manage the container lifecycle and ensure the Pod's availability to serve requests.

The following table outlines the IDSP namespaces and the Pods deployed within each.

Suggested namespaces	Services running under each namespace	Description
Ingress	NGINX Ingress	The external traffic to the IDSP is handled t configuration which is read by the Ingress C Chart provides the ingress configuration to the URI path.
logging	• Elasticsearch • Kibana	IDSP writes the logging to standard out stre default. All the logs that are generated by th stack (ElasticSearch, Fluent-bit, and Kibana
monitoring	• Prometheus and Grafana	The solution uses Prometheus for metric ins metrics data and alerts.

ssp	• hazelcast • opa • geolocation • admin • iarisk • azserver • auth-mgr • factor • identity • scheduler • SignIn UI • Admin Console • Self-Service Console • Infrastructure (Fluent-bit, Database) • Custom Metrics Prometheus Adapter	• IDSP microservices comprise Factor se • Security Services Platform comprises a • Infrastructure Services: — IDSP services write log messages in stream all messages to Elasticsearch — Custom Metrics Prometheus Adapte scaler.

IDSP Microservices Descriptions

The following table provides detailed descriptions and dependencies for each IDSP microservice.

Micro-Service Name	Description
hazelcast	**Cache service for IDSP** Provides high-performance distributed caching for the IDSP platform. Manages session state, configur tokens, and shared data across the microservices cluster. Ensures data consistency, supports cache in tolerance for cached data across multiple nodes in the Kubernetes deployment.
opa	**Open Policy Agent (Policy Engine)** Serves as the centralized policy decision point implementing fine-grained authorization policies. Evalua authorization policies, risk policies, and identity provider discovery policies. Processes policy queries fr as-code approach and returns policy decisions based on configured rules, contextual information, and
geolocation	**Geolocation Service** Provides IP-based geolocation services for risk assessment and location-based security analysis. Dete authentication requests, identifies country changes, detects VPN/proxy usage, and supports the risk en factors. Essential for compliance with geographic access restrictions and fraud detection.
admin	**Administration Operations Service** Comprehensive administrative service handling all tenant and system management operations. Provid applications, users, groups, policies, certificates, signing keys, identity stores, and system settings. Ma configurations, resource lifecycle, and administrative workflows through both REST APIs and the Admi
iarisk	**Intelligent Adaptive Risk Engine** IDSP risk assessment engine that evaluates user, client, and machine risk scores using behavioral ana location analysis, and historical patterns. Implements adaptive risk policies, manages risk thresholds, p authentication decisions.
azserver	**OAuth Authorization Server & SAML Identity Provider** Implements OAuth 2.0/OpenID Connect authorization server and SAML 2.0 identity provider functional (access tokens, ID tokens), authorization code flows, client authentication, SAML assertions, federation authentication. Manages signing keys for token and assertion and JWKS/SAML metadata publishing.
auth-mgr	**Authentication Orchestrator Service** Central authentication orchestration service that coordinates the complete multi-factor authentication fl policy evaluation, factor selection, risk-based authentication decisions, and integration with external ide communication between risk engine, policy engine, factor services, and identity stores to provide seam

factor	**Multi-Factor Authentication Service** Comprehensive authentication factor service supporting various authentication methods including pass push notifications, FIDO/WebAuthn, TOTP, custom authenticators, and biometric factors. Manages fac lifecycle management, and integrates with external authentication systems and IDSPcredential service
identity	**Identity Operations Service** Manages identity lifecycle operations for both internal identity store and external identity sources. Hand management, group operations, identity data persistence, and service account management. Provides synchronization, LDAP integration, and supports both disambiguation and authentication identity stores
scheduler	**Task Scheduling Service** Provides comprehensive job scheduling and background task execution capabilities for the IDSP platfo maintenance tasks, certificate renewals, cleanup operations, policy synchronization, audit log processi operations across the distributed system.
adminconsole	**Administrative Management Console** A web-based administrative interface for system and tenant administrators. Offers administrative workf comprehensive management of users, applications, policies, configurations, certificates, identity stores
signIn	**Authentication User Interface** Modern, responsive web-based user interface for authentication flows supporting multi-language locali accessibility features, and mobile-responsive design. Handles user interactions during authentication, f and self-service operations. Integrates with the authentication orchestrator for seamless user experien
selfserviceconsole	**End-User Self-Service Portal** A web-based self-service portal enabling end users to manage their authentication settings, view and r manage authentication factors, update profile information, perform password operations, and view auth intuitive interface for personal account management and credential lifecycle.

Kubernetes Support

This page mentions the supported Kubernetes and Helm versions in Symantec Identity Security Platform.

IDSP supports running the solution on Kubernetes. More specifically, we support all active Kubernetes releases, including equivalent OpenShift versions.

Kubernetes Distributions

IDSP specifically tests and supports the following Kubernetes distributions when running supported Kubernetes versions as indicated earlier:

- Azure Kubernetes Service (AKS)
- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- OpenShift Container Platform (OCP)
- VKS (VCF with VMware vSphere Kubernetes Services)

The tested versions of Kubernetes and OpenShift are noted in the next section.

Recommended Kubernetes and Helm Version Requirements

The recommended Kubernetes requirements to deploy IDSP are mentioned in the following table:

Software	Description	Providers	Tested Versions
Kubernetes	Kubernetes (K8s) is an open-source system to manage containerized applications in a clustered environment.	Google Kubernetes Engine (GKE)	1.32, 1.33, and 1.34
		Amazon Kubernetes Service (EKS)	1.32, 1.33, and 1.34
		Azure Kubernetes Service (AKS)	1.32, 1.33, and 1.34
		OpenShift Container Platform (OCP)	OpenShift 4.18

Software	Description	Providers	Tested Versions
		VKS (VCF with VMware vSphere Kubernetes Services)	v1.31.4+vmware.1-fips
HELM	Helm is an application package manager running on top of Kubernetes which is used to deploy Helm charts.	_	3.10 and above

Prerequisites

Familiarize with the recommended software and hardware requirements before deploying a Symantec Identity Security Platform solution.

Before setting up the Kubernetes environment for IDSP, ensure that the following recommended prerequisites are met.

- Kubernetes and Helm Requirements
- Cloud Cluster Environment Requirements
- Sizing Your Deployment
- Supported Databases
- Supported User Repository
- LDAP Reasonable Commercial Effort Statement
- Security Consideration

Kubernetes and Helm Requirements

For supported Kubernetes and Helm requirements to deploy IDSP, see Kubernetes Support.

Cloud Cluster Environment Requirements

To deploy IDSP in a **production environment**, there are a number of considerations to be made to ready a cloud cluster. These are described in Preparing the Cloud Cluster Environment page.

Sizing Your Deployment

To effectively size a **production deployment**, see the "Sizing the Deployment" page for crucial information about your system's requirements. This page includes an attached, interactive sizing sheet. By entering details about your system into the sheet, the sizing sheet automatically computes and generates the necessary information for sizing your system's CPU and memory, database, and storage needs.

To deploy IDSP in a **demo mode**, see the table with minimum cluster infrastructure requirements here.

Supported Databases

To know about the supported databases in IDSP, see Database Considerations.

Supported User Repository

User Repository	Versions
Internal Identity Store	Embedded
Symantec Directory	14.1
Microsoft Active Directory	2016
SCIM	2.0

LDAP Reasonable Commercial Effort Statement

This Support Statement applies to IDSP product that offers documented support for one or more LDAP v3 compliant user stores. Broadcom strives to meet our clients' diverse and ever changing needs. Broadcom products support and manage many of today's leading platforms, operating systems and various user stores across the IT enterprise. Many of our clients use variants of the LDAP server implementations as user store, for example, Symantec Directory, Microsoft Active Directory, Oracle Directory services. Broadcom does not test every LDAP v3 compliant directory server and cannot certify specific client configurations. To facilitate a quick resolution and isolate the root cause of any potential product issue encountered running on LDAP v3 compliant user store, Broadcom is establishing the following support protocol for IDSP operated in these environments:

- The client is responsible for properly configuring their LDAP user store that must be compatible with LDAP v3 standard.
- The client is responsible for having an active maintenance agreement for both their IDSP product and for the LDAP v3 directory server.
- While Broadcom does not require that clients recreate each issue before contacting support, we can request that the client diagnose and troubleshoot specific issues on a supported LDAP v3 directory server. Broadcom will only do this when we have reason to believe that the issue is directly related to LDAP server variant that is in use.
- Compatible LDAP v3 variants are diverse; Broadcom may require extra time to understand, collect data, troubleshoot, and possibly reproduce reported issues.
- If Broadcom Support cannot directly identify the root cause as a Broadcom or an LDAP v3 compatible directory server issue, client can open a support issue with their LDAP vendor and any other necessary third-party vendors to expedite the resolution of the issue. Broadcom, the vendor(s), and the client will work together toward a quick resolution where there is business relationship mechanism to do so. Note if the customer does not have a Vendor support agreement for their LDAP directory server variant, there is no third-party Broadcom may work with.
- Any known issues with running IDSP on specific LDAP v3 compatible variants will be noted in the respective product documentation.

Security Considerations

See the Security Considerations section for guidelines to be applied on day one of the deployments.

Helm Charts

The Symantec Identity Security Platform uses Kubernetes container-orchestration system to deploy its application package. The package includes a set of helm charts that comprises a collection of Kubernetes manifest files. The helm charts help you to define, install, and upgrade services within the cluster.

Helm is a native tool of the Kubernetes that functions as a package manager and streamlines the deployed services within a cluster. To know more about HELM, refer to the HELM website.

A HELM chart includes the following files:

File Name	Description
Templates folder containing Kubernetes manifest files	The templates folder contains YAML file templates defining Kubernetes resources, for example, Deployments, Services, Ingress. When Helm evaluates a chart, it sends all the files in the templates folder through the template rendering engine along with a set of deployment configuration values. It then collects the results of those templates and sends them to Kubernetes.
values.yaml	This file contains the default values for a chart, including container image, service type, port, and so on. These values may be overridden by users during helm installation or helm upgrade.

Helm Charts

The IDSP package includes the following Helm Charts:

Helm Chart Name	Description
ssp	This chart deploys IDSP services.
ssp-sample-app	This chart deploys a sample application in IDSP.
ssp-infra	This is the IDSP Infrastructure chart that supplies optional Fluent-Bit log collector and demo database.
ssp-data	This is the IDSP ssp-data chart that uploads the fraud IP data into the IDSP (solution) database which is required for risk evaluation. ***Note***: You can deploy the "ssp-data" chart only after deploying the IDSP chart.

Reference Links:

- To know more about deploying the ssp, ssp-infra, and ssp-data chart, see Deploying the Symantec Identity Security Platform.
- To know about ssp-sample-app, see Deploying the Sample Application.

Sizing the Deployment

Learn about deployment sizing.

It is critical to ensure that Symantec Identity Security Platform is properly implemented within the time frame that is determined by your requirements. The more thoroughly you plan your deployment, the more solid your configuration is, and you meet timelines and milestones while staying within your budget.

This topic includes the following sections:

- Deployment Sizing Considerations
- Sizing Systems (includes an interactive deployment sizing sheet)
- Recommended Cluster Infrastructure Requirements for Demo Mode

Deployment Sizing Considerations

When sizing a deployment in production, consider reading and gathering information about the following important aspects regarding your system:

- ***Number of users supported***

Determine the number of users that are supported in your deployment based on system usage. Once you have determined the number of users, you can project future growth.

- ***Number of users that can authenticate concurrently***

  Determine the average number of users that can concurrently authenticate using IDSP. Equally important is to determine the maximum number of users that can concurrently (Peak Load) authenticate using IDSP so that the system can be configured to scale accordingly.
- ***Percentage of authentications that require MFA***

  Depending on the type of application and its associated policies, authentication can require a single factor or multiple factors (MFA). Determine what % of authentications require MFA as it impacts the overall requests throughput (req/sec), hence impacting the authentication throughput (auth/sec) that the system can support.
- ***High Availability Requirements***

  High Availability (HA) refers to the ability of your system to operate continuously for a specified length of time. Design your system to prevent single points of failure and support continuous availability. Before deciding IDSP deployment in HA mode, decide how critical the IDSP service is to warrant the deployment in HA mode and across multiple sites.

  If HA is required, based on the size of your deployment, you can create a deployment architecture using a single zone configuration. For larger deployments, consider implementing a multi zone configuration. If the service experiences a complete failure at one zone, you can redirect client traffic to another zone and you can continue with your operation.
- ***Protected Applications***

  Determine the type of applications that IDSP must protect. Applications can be web applications or programmatic clients (example, Mobile apps). Web applications are integrated using OIDC protocol and programmatic applications use the API-based integration with IDSP (OAuth2).

  The client type determines the components that are required for deployment and sizing. For example, if the customer integrates with IDSP only using APIs then the SignIn component sizing can be ignored as it is not involved in the API-based authentication flows. Similarly, if the customer decides not to use Risk in authentication flows, then the Risk component installation and sizing can be ignored.
- ***Security Requisites***

  Firewalls provide an additional layer of security for your deployment. If you are planning to deploy IDSP behind a firewall, deploy a reverse proxy, such as Gateway. For another level of security, consider using multiple DNS infrastructures using zones; one zone for internal clients, another zone for external clients. To provide additional performance, you can deploy the DNS zones behind a load balancer.
- ***Ingress Deployment***

  One of the pre-requisites for IDSP is to have an ingress controller deployed in the Kubernetes cluster environment. Each enterprise has its own solution around ingress. In some environments, ingress is already configured in the cluster and must be shared by all applications. In some of the enterprises, ingress deployment and configuration are the responsibility of the product team.
- ***Observability and Monitoring Requisites***

  Observability (Elastic/Kibana) and monitoring (Prometheus/Grafana) tools are referred to as enclave services in IDSP. Enclave services can be deployed as part of IDSP deployment or IDSP can be linked to an existing enclave service in your environment. Enclave services must have its own CPU/Memory/Storage requirements. If enclave services are deployed as part of IDSP deployment, then it must share the Kubernetes cluster with the IDSP service.
- ***Disaster Recovery Requisites***

  Determine disaster recovery procedures to recover from events such as server hardware failures, network connections down-time, site failures.
  - Determine if the Disaster Recovery must be active always and the active load must be distributed between primary and Disaster Recovery nodes in the Active-Active mode, or if the Disaster Recovery must be in the passive mode.
  - Govern the impact of losing the transaction data (For example, flow state of in-progress authentications).

Sizing Systems

Once you have gathered the required information regarding your system, enter those details into the deployment sizing sheet.

Click the deployment sizing sheet.

The sizing sheet computes and generates the following information:

- **System CPU and Memory:** A sizing sheet computes the total CPU and memory that is required to handle the load. The sizing sheet also provides the following details:
  - Services or components of IDSP that are required to be deployed.
  - Minimum/maximum number of pods required for each service.
  - Minimum/Maximum memory and CPU for each pod of all the deployed services.
  - Total number of IP addresses required across all pods and services.
  - Total CPU and memory that is required to deploy various services of IDSP including enclave services.
- **Sizing Database:** IDSP requires MySQL database for storing the operational and transactional data that is required for authenticating users. The sizing sheet generates the following sizing details that are required for the database.
  - CPU required
  - Memory required
  - Disk size for the first year (GB)
  - Data growth per year after first year (GB)
  - Maximum number of database connections required
- **Storage**
  Enclave services provide the observability (Elastic/Kibana) and monitoring (Prometheus/Grafana) capabilities for IDSP. These components require persistent volumes to store data generated by various IDSP services. Based on the load supported by IDSP, the data storage for these components varies. Sizing sheet computes the size that is required per day for the Elasticsearch and Prometheus PVCs. As per the customer retention policy, the PVCs should be created with the appropriate size. Policies should be defined in the enclave services as per the customer's retention policy.
  Example: If the storage requirement for Elasticsearch is 1GB per day and the customer retention policy requires the log data to be available for 90 days, then the PVC of Elasticsearch should be 90 * 1 GB/Day = 90 GB.

Recommended Cluster Infrastructure Requirements for Demo Mode

To deploy IDSP in a demo mode, see the following table for minimum cluster requirements:

Parameter	Minimum Cluster Requirements
CPU	14 cores available in the cluster (7 cores in case enclave services are not deployed).
RAM	44 GB available in the cluster. (28 GB in case enclave services are not deployed).

NOTE

- The number of worker nodes that are required will be determined by the number of cores and RAM used per machine.
  For example, when using 4-core machines with 16GB RAM and, a 100-GB hard drive, as your worker nodes, you need three worker nodes cluster that can scale to four nodes (to accommodate for rolling upgrade).
- In the Cloud environment, the control planes, also known as Master Node(s), could be handled by their respective Cloud platforms. If your cloud provider does not transparently handle the control plane, or you deploy the solution in Baremetal, you need additional machines to be used as Master nodes.

Preparing the Infrastructure Before Deploying the Solution

Includes links to sections on how to prepare the Cloud Infrastructure to deploy Symantec Identity Security Platform.

Before deploying IDSP in your environment, ensure you set up these environments with the necessary steps and also check the database considerations. This topic includes the following information:

- Preparing the Cloud Environment
- Preparing the OpenShift Environment

Preparing the Kubernetes Cluster Environment

This topic outlines the essential steps to prepare your Kubernetes environment for deploying Symantec Identity Security Platform.

The solution is designed to operate on Kubernetes Services from various providers, including:

- Google Kubernetes Engine (GKE)
- Amazon Elastic Kubernetes Service (EKS)
- Azure Kubernetes Service (AKS)
- VKS (VCF with VMware Kubernetes Services)

NOTE

- Before proceeding with the deployment, ensure you consult the general Prerequisites for deploying IDSP in your Kubernetes environment.
- While this document focuses on setting up IDSP, specific Kubernetes cluster provisioning details should be referred to on the Kubernetes site or your chosen external cluster provisioning site. Our instructions only include how to set up the IDSP on the cluster that you have provisioned.
- **Administrative Acess**: During deployment, a bootstrap deployer user account is created as a Kubernetes Secret, expiring in 48 hours. See Retrieving Initial Credentials for the clients section for more details. It is critical to use the Admin Console tool to create your own administrative account promptly. To ensure you have sufficient time to establish ingress and access the Admin Console, consider extending the bootstrap credential's expiry period by deploying the chart with the parameter: `--set ssp.global.seedData.disclosureExpiryHours=96`
- **Security Best Practice**: Once your administrative account is established, it is recommended to remove the generic deployer account. In the event of an administrative lockout, this deployer account can be re-created by re-running the Helm chart.
- **Firewall Configuration**: Your organization's firewall policy must permit outbound SMTP traffic on TCP ports 25, 587, or 465, depending on your message provider configuration. See Managing Message Provider Configuration for more info. Additionally, outbound HTTPS traffic on port 443 must be allowed to connect to the Twilio REST API at https://api.twilio.com/. For more details, see Requirements-for-Connecting-to-the-Twilio-REST-API.
- For the rest of the options in the cluster, take the defaults.

The basic guidelines to create a cluster to deploy IDSP in demo mode in any of these environments have been covered in this topic.

NOTE
For details about resource requirements to deploy IDSP to support production loads, see Sizing the Deployment.

- Minimum Cluster Requirement for Google Kubernetes Engine (GKE)
- Minimum Cluster Requirement for Amazon Elastic Kubernetes Service (EKS)
- Minimum Cluster Requirement for Azure Kubernetes Service (AKS)
- Minimum Cluster Requirement for VCF with VMware Kubernetes Services (VKS)
- Validating the Cluster

Minimum Kubernetes Cluster Requirement for Google Kubernetes Engine (GKE)

When you create a GKE cluster to deploy IDSP, follow these **minimum** requirements:

Cluster Fields	Description
Location Type	Zonal (recommended)
Kubernetes Version	1.32.x (This is the minimum supported version. For other supported Kubernetes Versions, see here.)
Node Pool Size	This refers to the configuration of worker nodes in a node pool when deploying a cluster. Enable auto-scale and set min as 5 and max as 6.
Operating System (OS)	Container-Optimized OS with containerd (cos_containerd)
Instance Type	n2d-standard-4 (Resource Capacity- 4 vCPU / 16 GB RAM on each node)

NOTE

- To grant cluster-admin permission to your gcp user account, see Granting "Cluster-Admin" Permission.

Minimum Kubernetes Cluster Requirement for Amazon Elastic Kubernetes Service (EKS)

When you create an EKS cluster to deploy IDSP, follow these **minimum** requirements:

Cluster Fields	Description
Location Type	Region

Kubernetes Version	1.32.x (This is the minimum supported version. For other supported Kubernetes Versions, see here.) If you are using Kubernetes version 1.30 or later, ensure your EKS cluster includes the following annotation in the default storage class:
	`apiVersion: storage.k8s.io/v1`
	`kind: StorageClass`
	`metadata:`
	`annotations:`
	`storageclass.kubernetes.io/is-default-class: "true"`
	Note: This annotation is necessary to avoid issues with Persistent Volume Claim creation, especially when using services like Kibana.

Node Pool Size	This refers to the configuration of worker nodes in a node pool when deploying a cluster. Enable auto-scale and set min as 5 and max as 6.
Operating System (OS)	Linux (amd64)
Instance Type	t3.xlarge (Resource Capacity- 4 vCPU / 16-GB RAM on each node)

NOTE

- From AWS EKS version 1.23 onwards, the in-tree Amazon EBS storage provisioner has been deprecated. A storage class driver ("AWS EBS CSI driver") has to be installed on the cluster to allow the provisioning of persistent storage. For more info see, AWS EKS Storage Classes and Amazon EBS CSI driver.
- IDSP supports both IMDSv1 and IMDSv2 instances in an AWS cluster. However, it is recommended to use IMDSv2 for its enhanced security. See Security Considerations for more info.

Minimum Kubernetes Cluster Requirement for Azure Kubernetes Service (AKS)

When you create an AKS cluster to deploy IDSP, follow these **minimum** requirements:

Cluster Fields	Description
Location Type	region
Kubernetes Version	1.32.x (This is the minimum supported version. For other supported Kubernetes Versions, see here.)
Node Pool Size	This refers to the configuration of worker nodes in a node pool when deploying a cluster. Enable auto-scale and set min as 5 and max as 6.
Operating System (OS)	Azure Linux / Ubuntu Linux
Instance Type	Standard_D4s_v3 (Resource Capacity- 4 vCPU / 16-GB RAM on each node)

Minimum Kubernetes Cluster Requirement for VCF with VMware Kubernetes Services (VKS)

When you create an VKS cluster to deploy IDSP, follow these **minimum** requirements:

Cluster Fields	Description
Location Type	region
Kubernetes Version	1.32.x (This is the minimum supported version. For other supported Kubernetes Versions, see here.)
Node Pool Size	This refers to the configuration of worker nodes in a node pool when deploying a cluster. Enable auto-scale and set min as 5 and max as 6.
Operating System (OS)	Ubuntu
Instance Type	Standard_D4s_v3 (Resource Capacity- 4 vCPU / 16-GB RAM on each node)

Validating the Cluster

To validate that the cluster is operational, run the following command:

```
kubectl get nodes
```

This command retrieves all the nodes. For a cluster to be successfully operational, check that the nodes in the cluster are in the "Ready" state.

Similarly, run the following command to check that all the default system pods in the cluster are in *running state and ready:*

```
kubectl get pods --all-namespaces
```

Next Step

Deploying the Services

Preparing the OpenShift Cluster Environment

Learn the prerequisite steps to perform before deploying Symantec Identity Security Platform in your OpenShift cluster environment.

This topic includes the following sections:

- Minimum Cluster Requirement for OpenShift
- Required Security Context Constraints (SCC) for IDSP Infra Helm Chart
- Validating the Cluster

NOTE

- Refer to the Prerequisites before deploying IDSP in your OpenShift environment.
- To set up your OpenShift cluster, visit the OpenShift site. Our instructions only include how to set up the IDSP on the cluster that you have provisioned.
- For details about resource requirements to deploy IDSP to support production loads, see Sizing the Deployment.

Minimum Cluster Requirement for OpenShift

When you create an OpenShift cluster to deploy IDSP, follow these **minimum** requirements:

Cluster Fields	Description
Location Type	region/zonal
OpenShift Version	OpenShift 4.18 (This is the minimum supported version. For other supported Kubernetes Versions, see here.)
Node Pool Size	5-6 worker nodes
Instance Size	4 CPUs 16-GB RAM on each worker node

NOTE

- Based on the TCP port that your organization uses for outgoing SMTP communication, your firewall policy should allow the outbound traffic on one of the following ports: TCP 25, 587, or 465. See Managing Message Provider Configuration for more info.
- Twilio uses HTTPS on standard port 443. Therefore, firewalls and proxies must allow outbound HTTPS traffic on port 443 to connect to the REST API hosted in https://api.twilio.com/. For more details, see Requirements-for-Connecting-to-the-Twilio-REST-API.
- For the rest of the options in the cluster, take the defaults.

Required Security Context Constraints (SCC) for IDSP Infra Helm Chart

IDSP runs under the most secure out-of-the-box OpenShift security profile, Security Context Constraints (SCC), "restricted-v2" SCC. Therefore, deploying IDSP on OpenShift does not call for any additional SCC. However, some of the prerequisite IDSP infrastructure components, like the NGINX Ingress controller or the Infra chart which deploys fluent bit for log streaming, requires a custom SCC.

You may have to deploy SCCs for the following reasons:

- ***Ingress NGINX***- If you intend to deploy NGINX Ingress in an OpenShift cluster. See Deploying Ingress in OpenShift.
- ***Fluent-bit*** - IDSP Infra helm chart deploys fluent bit for log streaming that requires the following SCC. The SCC is deployed automatically by the ssp-infra chart but can also be manually applied. In case you must deploy it manually, use the following command:

```
cat <<EOF | kubectl apply -f -
apiVersion: security.openshift.io/v1
kind: SecurityContextConstraints
metadata:
  name: fluent-bit-scc
```

```
allowPrivilegedContainer: true
allowPrivilegeEscalation: true
allowHostDirVolumePlugin: true
defaultAllowPrivilegeEscalation: false
# forbid host namespaces
allowHostNetwork: false
allowHostIPC: false
allowHostPorts: false
allowHostPID: false
allowedCapabilities: []
forbiddenSysctls:
- "*"
readOnlyRootFilesystem: false
requiredDropCapabilities:
- MKNOD
runAsUser:
  type: RunAsAny
seLinuxContext:
  type: MustRunAs
supplementalGroups:
  type: RunAsAny
volumes:
  - configMap
  - emptyDir
  - hostPath
  - persistentVolumeClaim
  - secret
EOF
```

Once you create the SCC manually, add the following parameter to the IDSP Infra Helm chart deployment:

```
--set fluent-bit.openShift.securityContextConstraints.create=false
```

Validating the Cluster

To validate that the cluster is operational, run the following command:

```
kubectl get nodes
```

This command retrieves all the nodes. For a cluster to be successfully operational, check that the nodes in the cluster are in the "Ready" state.

Similarly, run the following command to check that all the default system pods in the cluster are in the *running state and ready:*

```
kubectl get pods --all-namespaces
```

Reference links:

- Deploying Ingress in OpenShift

Next Step

Deploying the Services

Database Considerations

This topic provides an overview of the database options and configuration considerations for deploying Symantec Identity Security Platform.

Deployment Modes

IDSP database subsystem supports the following deployment modes, giving you flexibility based on your needs:

These modes have been covered in detail in the following links:

- Out-of-the-Box Demo Database Mode - This is a simplified, default setup, typically used for initial evaluation or non-production environments. It comes pre-configured to work seamlessly with MySQL.
- Customer Database Mode - This mode is for production environments where you connect to your own, externally managed database.

Supported Database Types

IDSP is designed to work with several popular database platforms. The following table summarises the supported databases and their compatible platforms and versions.

Platform	Database Type	Supported DB Version	Infra chart configuration	SSP chart configuration
• Google Cloud • Microsoft Azure • Amazon Web Services • OpenShift	MySQL (default) See Demo Database mode, and Customer Database mode for more info.	8.0, 8.4	• "db.enabled=true" (in Demo mode) • "db.enabled=false" (in Customer Database Mode)	--set ssp.db.type=mysql
• Google Cloud • Microsoft Azure • Amazon Web Services	PostgreSQL See Customer Database mode for more info.	14.19, 15.14, 17.7	"db.enabled=false" (In Customer Database Mode).	--set ssp.db.type=postgresql
• Google Cloud • Microsoft Azure • Amazon Web Services	Oracle See Customer Database mode for more info.	19c	"db.enabled=false" (in Customer Database Mode)	--set ssp.db.type=oracle
• Amazon Web Services	Aurora MySQL	8.0	"db.enabled=false" (in Customer Database Mode)	--set ssp.db.type=mysql

By following these guidelines, you can ensure your IDSP deployment is correctly configured for your chosen database environment.

Database in SSL Mode

IDSP supports SSL mode in databases. To know more about it, see Using Database in SSL Mode - In this mode, the solution deploys an SSL connection by default (sslMode=REQUIRED), however does not validate the server's issuer or server's identity in the server certificate.

Out-of-the-Box Demo Database Mode

The OOTB Demo database is a database created for demonstration purposes. Learn how to create it in Symantec Identity Security Platform by following the steps provided here.

In this mode, the IDSP infra chart creates a single MySQL database instance. This is the default mode of operation that is used only for demo purposes and not in production. By default, the infra chart's database configuration value *"db.enabled=true"* results in the creation of a single database instance.

NOTE
The IDSP infra chart does not use the memory requests and limits for the database deployment. However, if you want to deploy the database with any memory limit, use a minimum of 4G as a "***requests***" and "***limits***" value. The 4G limit is due to the requirement of the IDSP data chart deployment. The data chart requires a minimum of 4G limit for its job to complete successfully.

The following is an example from the values.yaml showing the values that can be used in "***requests***" and "***limits***":

```
pxc:
   crVersion: 1.10.0
   resources:
     requests:
       memory: 1G
       cpu: 600m
       # ephemeral-storage: 1Gi
     limits:
       memory: 4G
       cpu: "2000m"
       # ephemeral-storage: 1Gi
```

The following table displays the key database-related artifacts that are created by the infra chart deployed in the demo mode.

NOTE
The examples mentioned in the following section use *"ah-dev"* as the release name and *"dev"* as the namespace.

- The "ah-dev-ssp-db-pxc-0" pod is where the database service is running.
- The "infra-ah-dev-create-db-job-r5vns" pod is where the job to create the database schema is running.
- The "datadir-ah-dev-ssp-db-pxc-0" (PVC) Persistent Volume Claim represents the database persistence volume claim point to the persistence volume
- The "ah-dev-ssp-db-pxc" secret is where the database passwords are stored.

```
$ kubectl get pods -n dev
NAME                                                    READY   STATUS      RESTARTS   AGE
...
infra-ah-dev-create-db-job-r5vns                        0/1     Completed   0          2d15h
ah-dev-ssp-db-pxc-0                                     2/2     Running     0          2d15h


$ kubectl get pvc -n dev
NAME                           STATUS   VOLUME                                     CAPACITY   ACCESS MODES
 STORAGECLASS   AGE
...
datadir-ah-dev-ssp-db-pxc-0    Bound    pvc-36b7f547-6be7-4cdd-82be-d01dbafb7958   20Gi       RWO
  standard       109m


$ kubectl get pv -n dev
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
                          STORAGECLASS   REASON   AGE
```

```
...
pvc-36b7f547-6be7-4cdd-82be-d01dbafb7958   20Gi        RWO            Delete           Bound    dev/datadir-
ah-dev-ssp-db-pxc-0           standard                109m


$ kubectl get secrets -n dev
NAME                                                         TYPE                                  DATA
  AGE
...
ah-dev-ssp-db-pxc                                           Opaque                                8
 18d
```

Deployment of Infra Helm Chart

Deploy the Infra chart normally without passing database-specific parameters. The database is deployed automatically with the default values as shown in the following table, with passwords randomized.

Optional parameters to pass into the Infra chart are as follows:

Parameter	Default Value	Purpose
db.schema.name	iamauth	Name of the database schema
db.schema.user	iamauth	Name of the database schema user
db.schema.password	N/A	Password of the database schema user. You can provide this through a password or secret. If you do not provide the password, it is randomized.
db.existingSecret	N/A	Secret containing database passwords to use (in base64)
db.enabled	true	Enable or disable embedded database.
db.sslMode	REQUIRED	Enable SSL. See Database in SSL mode for more info.
db.existingSslSecret	N/A	Secret containing SSL certs to use for SSL.

Once the infra chart is deployed, the Percona MySQL operator creates a single Database Service that is named, *ah-dev-ssp-db-pxc-0*.

- The infra chart deploys the Percona MySql Operator from the DockerHub registry. The actual location can be updated via infra chart's values.yaml, for example, using your own images from a "secure" registry.

Once the database becomes available, the *infra-ah-dev-create-db-job* is executed to create the database schema based on the values that are provided and/or defaulted to.

- The database passwords are placed inside **ah-dev-ssp-db-pxc** secret, containing various Percona passwords and "mysql-password" value that is used by IDSP to connect to its database schema.

Once you deploy the Infra chart and the db-creation-job is complete, you can execute the IDSP chart normally.

- For full automation, you should align the deployment of the IDSP chart with the completion of the database creation job.
- Use the following Kubernetes command:"`kubectl wait --for=condition=complete --timeout=5m job/<release-name>-create-db-job> -n <namespace>`"

Database Lifecycle Management

When you uninstall the IDSP chart, it does not perform any action on the database. Upgrading or uninstalling the infra chart does **not** touch the database file (DB PVC) and any upgrade or reinstall of the infra and IDSP charts continue to reuse the same PVC. If you must delete the database file at some point, then delete the PVC.

NOTE

- Uninstalling the infra chart keeps the database secret. Reinstalling the infra chart detects the database secret on installation and refuses to proceed until both the secret and related PVCs are removed. Otherwise, redeploy the infra chart by setting the parameter, *"--set db.existingSecret=**ah-dev-ssp-db-pxc**"* to reuse the existing database persistence.
- If you have deployed the infra chart in the external 'Customer Database Mode' and want to switch to the 'Internal Demo Mode' at a later stage, or you must uninstall and reinstall, make sure to run this command- *"kubectl delete secret ah-dev-ssp-db-pxc -n dev"* and *"kubectl delete secret internal-ah-dev-ssp-db -n dev"*.

Next Step

Deploy IDSP

Customer Database Mode

Learn how to connect a pre-created Custom Database to Symantec Identity Security Platform by configuring the necessary database connection parameters.

This topic includes the following sections:

- Creating a MySQL Database instance
- Creating a PostgreSQL Database instance
- Integrating an Oracle Database instance with IDSP
- Deployment Instructions
- Database connection related issues covered in troubleshooting section

Supported Database Types and Versions

The supported database types and versions are mentioned in the Database Considerations page.

Preparing the Database Instance

To prepare a database, perform the steps mentioned in this section based on your specific database type. These commands create a database, a user, and grant the necessary permissions.

IDSP requires a pre-created database instance. In the customer provided database mode, point the IDSP chart to your own database.

NOTE

- When running in a cloud environment, consider using a ***MySQL***, ***PostgreSQL***, or ***Oracle*** compatible cloud provider database service.
- When using Azure Database for MySQL server, set the "Server timezone" to UTC, by typing “+00:00”.

Creating a MySQL Database Instance

Follow these steps to create an instance of the MySQL database.

1. Connect to the MySQL service by running a MySQL client container on your Kubernetes cluster where you plan to deploy IDSP.
2. Running the following SQL commands creates a database, a user within the MySQL database server and grant permissions to the user.

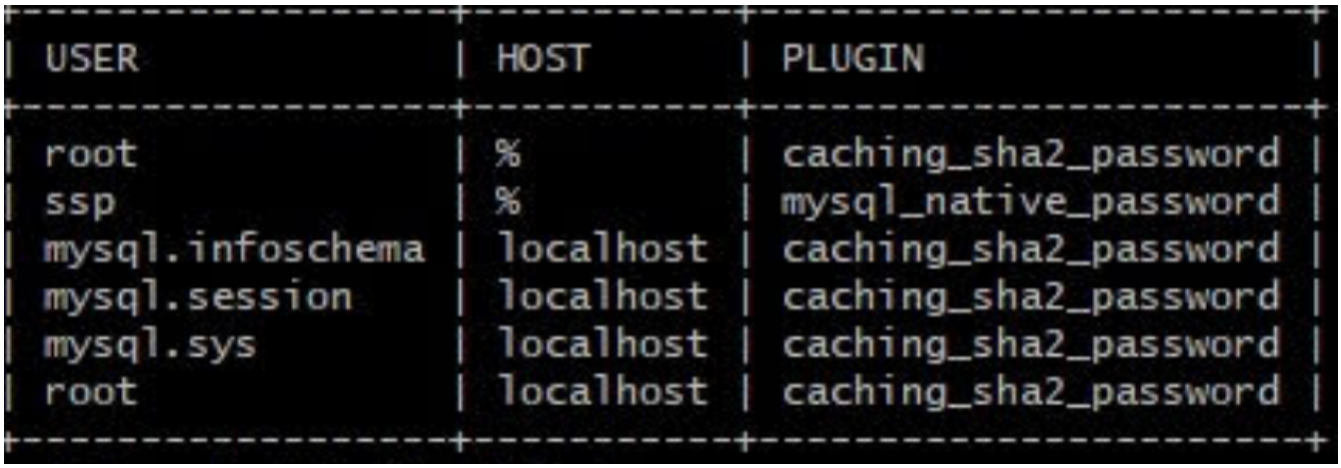
   NOTE
   IDSP supports connecting to a MySQL database using a "**mysql_native_password**" user.

```
export SCHEMA_NAME=<db_name>
export SCHEMA_USER=<db_user>
export SCHEMA_PWD=<db_user_password>;

kubectl run -it --rm --image=mysql:8.0.26 --restart=Never mysql-client -- mysql -h<db_host_fqdn> -
u<db_admin_user> -p<db_admin_password> -e "DROP DATABASE IF EXISTS ${SCHEMA_NAME}; CREATE DATABASE IF NOT
 EXISTS ${SCHEMA_NAME}; CREATE USER IF NOT EXISTS ${SCHEMA_USER} IDENTIFIED WITH mysql_native_password
 BY '${SCHEMA_PWD}'; ALTER USER ${SCHEMA_USER} IDENTIFIED WITH mysql_native_password BY '${SCHEMA_PWD}';
 GRANT ALL ON ${SCHEMA_NAME}.* TO ${SCHEMA_USER};"
```

3. (**Optional Step**) Before proceeding with the next steps, confirm that the user password is stored with the "**mysql_native_password**" mechanism.
   Run the following SQL command to display the password plugin mechanism for the created ${SCHEMA_USER}:

```
kubectl run -it --rm --image=mysql:8.0.26 --restart=Never mysql-client -- mysql -h <db_host_fqdn> -
u<db_admin_user> -p<db_admin_password> -e"SELECT USER, HOST, PLUGIN from mysql.user;"
```

   The following sample screenshot shows that the MySQL password plugin for the newly created "ssp" user is set to "mysql_native_password" (instead of "caching_sha2_password").



USER	HOST	PLUGIN
root	%	caching_sha2_password
ssp	%	mysql_native_password
mysql.infoschema	localhost	caching_sha2_password
mysql.session	localhost	caching_sha2_password
mysql.sys	localhost	caching_sha2_password
root	localhost	caching_sha2_password

   NOTE
   The Database user account should have no limits for "max_questions" and "max_queries" server status variables. See https://dev.mysql.com for more details. To see current parameters, sign in to the database as the DB user and execute the following queries:

   - SHOW SESSION STATUS LIKE 'questions';
   - SHOW SESSION STATUS LIKE 'queries';

Creating a PostgreSQL Database Instance

Follow these steps to create an instance of the PostgreSQL database in your Kubernetes cluster.

1. Connect to your Kubernetes cluster.
2. Connect to the PostgreSQL service by running a PostgreSQL client container on your Kubernetes cluster where you plan to deploy IDSP.

Running the following SQL commands creates a database, a user within the PostgreSQL database server, and grants permissions to the user to perform the required db tasks:

```
export SCHEMA_NAME=<db_name>
export SCHEMA_USER=<db_user>
export SCHEMA_PWD=<db_user_password>;
kubectl run -i --rm --tty percona-client --image=perconalab/percona-distribution-postgresql:15
 --restart=Never -- psql postgres://<db_admin_user>:<db_admin_password>@<db_host_fqdn> -
c \"DROP  DATABASE IF EXISTS ${SCHEMA_NAME};\"; kubectl run -i --rm --tty percona-client
 --image=perconalab/percona-distribution-postgresql:15 --restart=Never -- psql postgres://
<db_admin_user>:<db_admin_password>@<db_host_fqdn> -c \"CREATE DATABASE ${SCHEMA_NAME}
 ENCODING='UTF8';\"; kubectl run -i --rm --tty percona-client --image=perconalab/percona-distribution-
postgresql:15 --restart=Never -- psql postgres://<db_admin_user>:<db_admin_password>@<db_host_fqdn>
 -c \"DROP USER IF EXISTS ${SCHEMA_USER};\"; kubectl run -i --rm --tty percona-client --
image=perconalab/percona-distribution-postgresql:15 --restart=Never -- psql postgres://
<db_admin_user>:<db_admin_password>@<db_host_fqdn> -c \"CREATE USER ${SCHEMA_USER} WITH ENCRYPTED PASSWORD
 '${SCHEMA_PWD}';\"; kubectl run -i --rm --tty percona-client --image=perconalab/percona-distribution-
postgresql:15 --restart=Never -- psql postgres://<db_admin_user>:<db_admin_password>@<db_host_fqdn> -c
 \"GRANT ALL PRIVILEGES ON DATABASE ${SCHEMA_NAME} TO ${SCHEMA_USER};\
```

Where,

- db_admin_user - Name of the database schema user.
- db_admin_password - Password of the database schema user.
- db_host_fqdn - Hostname of the server that has the PostgreSQL deployed.

Integrating Oracle Database Instance with IDSP

To integrate Oracle databse with IDSP, ensure that the user has required privileges such as DDL (Data Definition Language) including creating procedures, DML (Data Manipulation Language), and the ability to establish connections.

For any database connection related issues, see the troubleshooting link at the end of this page.

Deployment Instructions

Infra Chart Configuration

Set the parameter, `db.enabled=false` in the `infra chart` to prevent the system from creating an internal database.

> **NOTE**
> If you configure "db.enabled=false", ensure external database details are prepared for input during the IDSP deployment.

IDSP Chart Configuration

To deploy the IDSP chart, configure the database connection using the parameters outlined in the provided table, ensuring it references your designated database for this deployment.

Parameter	Default Value	
ssp.db.serviceHost		FQDN host of the database
ssp.db.servicePort	• MySQL- 3306	Port of the database service Port values in other DBs are • postgresql - 5432 • oracle - 1521 (SSL port

Parameter	Default Value	
ssp.db.schema	iamauth	Name of the database sche ssp.db.schema parameters
ssp.db.name	iamauth	• Name of the database s (in MySQL db) • Name of the database (in PostgreSQL db) • Name of the database s (in Oracle db) **Note:** If you plan to use **Note:** see these section • Using Oracle JDBC • Using Oracle JDBC
ssp.db.user	iamauth	Name of the database sche
ssp.db.password	N/A	Password of the database s This parameter should not b The DB password should ei both values are specified, th
ssp.db.existingSecret	N/A	The name of the secret cont The name of the key in the s use this option, the secret is deployed. IDSP mounts this secret into This means that whenever t the password in the mounte password to be used to con The secret is expected to ha • 'db-password' (for read/ • 'read-db-password' (for r • when not found, the file To know more about Databa
ssp.db.existingVaultFolder	""	When using a vault injected the file named, 'mysql-passv the file is expected to contai IDSP does not use the ssp. Passwords are reloaded fro
ssp.db.sslMode	REQUIRED	Enable SSL. See Database
ssp.db.existingSslSecret	""	Secret containing SSL certs
ssp.db.type	mysql	Specify the database type to Supported types are: • mysql • postgresql • oracle

Database connection-related issues covered in troubleshooting section

To view database connection-related issues that are covered in the troubleshooting section, see Database Connection Issues.

Reference Links

- Database Connection pool
- Oracle JDBC Setup (with LDAP examples)
- Database Lifecycle Management

Database Connection Pool

Explore the configurable parameters for the Database Connection Pool in Symantec Identity Security Platform.

IDSP supports the use of Database Connection Pool configured at the time of deploying the IDSP chart for its operational database.

The operating parameters of the DB pool are global and must be configured at the time of deploying the IDSP chart. The default Database Connection Pool parameters are maintained in IDSP helm chart's values.yaml.

The DB pool parameters that you can pass while deploying IDSP helm chart using ***--set <name>="<value>"*** helm parameters, are as follows:

Database Pool Config Value	Default	Description
ssp.global.db.maxPoolSizeRead	200	Max number of database connections for reads
ssp.global.db.maxPoolSizeWrite	200	Max number of database connections for writes
ssp.global.db.minIdleSize	5	Minimum number of idle database connections
ssp.global.db.connectionTimeoutMillis	30000	Maximum amount of time to wait for a database connection, in milliseconds
ssp.global.db.idleTimeoutMillis	12000	Maximum lifetime of an idle database connection before its closed, in milliseconds
ssp.global.db.maxLifetimeMillis	1800000	Maximum lifetime of a database connection, in milliseconds. Usually this number should be less than the firewall's TCP keepalive timeout if one is present.

Oracle JDBC Setup for Symantec Identity Security Platform

Learn about Oracle JDBC DB configurations when integrating with Symantec Identity Security Platform.

This topic includes the following sections:

- Using Oracle JDBC
- Using Oracle JDBC with LDAP

Using Oracle JDBC

When using Oracle JDBC (Java Database Connectivity), configure the following parameters:

1. Set the parameter "***ssp.db.connection.type***" to "***oracle***".
2. Set the parameter "***ssp.db.jdbcUrl***" to specify your ***JDBC url.***
   The following example demonstrates a sample JDBC url:

```
"jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=OFF)(FAILOVER=ON)(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
(HOST=<your-host1>)(PORT=50001))(ADDRESS=(PROTOCOL=TCP)(HOST=<your-host2>)(PORT=50001)))
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=<your-service-name>)))"
```

Using Oracle JDBC with LDAP

In scenarios where the Oracle database host is managed within LDAP, the JDBC driver interacts with LDAP to fetch the required connection information.

When using Oracle JDBC (Java Database Connectivity) with LDAP, configure the following parameters:

1. Set the parameter "***ssp.db.connection.type***" to "***oracleldap***".
2. Set the parameter "***ssp.db.jdbcUrl***" to specify the ***complete JDBC url***

NOTE

- If the JDBC URL is passed as a "--set" parameter in the Helm command and contains a comma (,), it should be escaped with a backslash (\) and the value should be encapsulated with single quotes (').
- The following example demonstrates how to use (\) and (') in the Helm command:
  `--set ssp.db.jdbcUrl='jdbc:oracle:thin:@ldap://oidhost1.test.com:46301/cn=OracleContext\,dc=oracle\,dc=com/devd b'`

Database Lifecycle Management

This section covers how to manage your database and its connection details after the initial deployment.

- **Upgrades and Uninstalls**: Upgrading or uninstalling the Symantec Identity Security Platform chart does **not** impact your external database. The data and settings in your database will remain intact.
- **Password Changes**: To change the database password after the initial installation, use the IDSP Helm chart upgrade command.
  adding, *"--set reuse-values=true"* parameter and specify the new password using the following option:
  – `--set ssp.db.password=<schema-password>` (passed in a clear text)
- If you prefer to pass the password in the IDSP chart using a secret, create the secret and pass it in the following command-line parameter: `--set sp.db.existingSecret-<existing-k8s-secret-with-schema-user-password-in-base64>`

NOTE
If you previously used the existingSecret option to specify the secret name, you should change the password directly in the specified secret object and not use the "ssp.db.password" option.

Handling Database Passwords

For better security, it is highly recommended to use a Kubernetes secret to manage database passwords. This allows for seamless password rotation without impacting the IDSP chart.

Using Database in SSL Mode

Symantec Identity Security Platform (solution) supports SSL mode in database.

The solution deploys an SSL connection by default (sslMode=REQUIRED), however does not validate the server's issuer or server's identity in the server certificate.

NOTE

- The solution does not support mutual SSL where the MySQL server validates the client's certificate.
- ***The information in this topic applies to PostgreSQL and Oracle databases as well. While the examples focus on MySQL, they are also relevant to PostgreSQL and Oracle DB types.***

In the solution, the SSL database mode is controlled by the configuration parameter, "`ssp.db.sslMode=<SSL Mode>`" which is specified in the values.yaml file.

The different values that are supported by SSL mode are:

- DISABLED - IDSP connects without SSL assuming MySQL Server allows it.
- PREFERRED - IDSP expects SSL assuming MySQL Server allows it. However, it connects without SSL, if MySQL server does not have ssl enabled.

  NOTE
  This mode is not supported in Oracle DB.
- REQUIRED - IDSP requires SSL assuming MySQL Server allows it (***default mode***).
- VERIFY_CA - IDSP requires validation of the MySQL server's SSL certificate to be signed by a trusted CA.
- VERIFY_IDENTITY - In addition to VERIFY_CA, IDSP requires validation of the MySQL server's SSL certificate to contain the MySQL Server's hostname used during the connection.

  NOTE
  In general, it is sufficient to provide one certificate representing the trusted CA that was used to sign MySQL Server's certificate during SSL negotiation.

Creating secret with DB SSL Certificate

SSL Modes, ***Verify_CA*** and ***Verify_Identity*** require CA certificates to verify the DB server certificate. When using customer DB ("external db mode"), one or more SSL certificates of the DB service has to be imported into the secret for the solution to establish the SSL connection to MySQL server. The solution requires each certificate to be provided in a separate file inside the secret. Each file within the secret should have the ".crt" extension, other file extensions are ignored. Once the SSL secret is created, pass the ssl certificates secret name by adding this attribute, --set **ssp.db.existingSslSecret**="*<secret name>*".

A database provider must provide the required certificate for the database service. This certificate is provided to IDSP through a Secret that is created in the same namespace where the solution is to be deployed. When initializing the MySQL connection, the solution loads all the certificates with the ".crt" extension into its trust store. If multiple certificates are required by the db provider, then create the secret using multiple .crt files.

Example: Using Azure's MySQL ("Customer Database" mode) with VERIFY_CA or VERIFY_IDENTITY ssl mode

1. Download and create a secret with DB SSL certs for the right region, using this command:

   NOTE

   - IDSP mounts the SSL secret and loads all files with "**.crt**" extension. Any file with an extension other than "**.crt**" is ignored.
   - To download the right certificate bundle to be used by IDSP, see the Azure documentation.

   ```
   kubectl create secret generic aazure-mysql-ssl-cert -n ${NAMESPACE} --from-file=us-east-1-bundle.crt=us-east-1-bundle.pem
   ```

2. The following parameters are added to the IDSP deployment to use an SSL connection to connect to the Azure MySQL DB. The parameters specify the name of the secret that contains the certificate (created above) and the SSL mode to be used.

   ```
   --set ssp.db.existingSslSecret="azure-mysql-ssl-cert" \
   --set ssp.db.sslMode="VERIFY_CA"
   ```

   In the following example, the CA cert is obtained from Azure's MySQL service as the "BaltimoreCyberTrustRoot.crt.pem" certificate file.

   ```
    -----BEGIN CERTIFICATE----
   MIIDdzCCAl+gAwIBAgIEAgAAuTANBgkqhkiG9w0BAQUFADBaMQswCQYDVQQGEwJJ
   RTESMBAGA1UEChMJQmFsdGltb3JlMRMwEQYDVQQLEwpDeWJlclRydXN0MSIwIAYD
   VQQDExlCYWx0aW1vcmUgQ3liZXJUcnVzdCBSb290MB4XDTAwMDUxMjE4NDYwMFoX
   DTI1MDUxMjIzNTkwMFowWjELMAkGA1UEBhMCSUUxEjAQBgNVBAoTCUJhbHRpbW9y
   ZTETMBEGA1UECxMKQ3liZXJUcnVzdDEiMCAGA1UEAxMZQmFsdGltb3JlIEN5YmVy
   VHJ1c3QgUm9vdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKMEuyKr
   mD1X6CZymrV51Cni4eiVgLGw41uOKymaZN+hXe2wCQVt2yguzmKiYv60iNoS6zjr
   IZ3AQSsBUnuId9Mcj8e6uYi1agnnc+gRQKfRzMpijS3ljwumUNKoUMMo6vWrJYeK
   mpYcqWe4PwzV9/lSEy/CG9VwcPCPwBLKBsua4dnKM3p31vjsufFoREJIE9LAwqSu
   ```

```
XmD+tqYF/LTdB1kC1FkYmGP1pWPgkAx9XbIGevOF6uvUA65ehD5f/xXtabz5OTZy
dc93Uk3zyZAsuT3lySNTPx8kmCFcB5kpvcY67Oduhjprl3RjM71oGDHweI12v/ye
jl0qhqdNkNwnGjkCAwEAAaNFMEMwHQYDVR0OBBYEFOWdWTCCR1jMrPoIVDaGezq1
BE3wMBIGA1UdEwEB/wQIMAYBAf8CAQMwDgYDVR0PAQH/BAQDAgEGMA0GCSqGSIb3
DQEBBQUAA4IBAQCFDF2O5G9RaEIFoN27TyclhAO992T9Ldcw46QQF+vaKSm2eT92
9hkTI7gQCvlYpNRhcL0EYWoSihfVCr3FvDB81ukMJY2GQE/szKN+OMY3EU/t3Wgx
jkzSswF07r51XgdIGn9w/xZchMB5hbgF/X++ZRGjD8ACtPhSNzkE1akxehi/oCr0
Epn3o0WC4zxe9Z2etciefC7IpJ5OCBRLbf1wbWsaY71k5h+3zvDyny67G7fyUIhz
ksLi4xaNmjICq44Y3ekQEe5+NauQrz4wlHrQMz2nZQ/1/I6eYs9HRCwBXbsdtTLS
R9I4LtD+gdwyah617jzV/OeBHRnDJELqYzmp
-----END CERTIFICATE-----
```

Example: Using AWS's MySQL ("Customer Database" mode) with VERIFY_CA or VERIFY_IDENTITY in SSL mode

1. Download and create a secret with DB SSL certs for the right region, using this command:

   NOTE

   - IDSP mounts the SSL secret and loads all files with "**.crt**" extension. Any file with an extension other than "**.crt**" is ignored.
   - To download the right certificate bundle to be used by IDSP, see the Amazon documentation.

   ```
   kubectl create secret generic aurora-mysql-ssl-cert -n ${NAMESPACE} --from-file=us-east-1-bundle.crt=us-east-1-bundle.pem
   ```

2. The following parameters are added to the IDSP deployment to use an SSL connection to connect to the AWS MySQL DB. The parameters specify the name of the secret that contains the certificate (created above) and the SSL mode to be used.

   ```
   --set ssp.db.existingSslSecret="aurora-mysql-ssl-cert" \
   --set ssp.db.sslMode="VERIFY_CA"
   ```

Example: Sample showing the creation of a secret with multiple certs

In the following example, a secret type generic, called "azure-mysql-ssl-cert" is created with two certificates, "BaltimoreCyberTrustRoot.crt" and "DigiCertGlobalRootG2.crt". This method of including two certificates is useful when you must specify a certificate chain or when your database is transitioning from one certificate to another.

The command loads the certificate files into the secret while renaming their extensions from ".crt.pem" to ".crt".

```
$ kubectl create secret generic azure-mysql-ssl-cert -n dev \
   --from-file=BaltimoreCyberTrustRoot.crt=BaltimoreCyberTrustRoot.crt.pem \
   --from-file=DigiCertGlobalRootG2.crt=DigiCertGlobalRootG2.crt.pem
```

Run the following command to verify the content of the newly created secret:

```
kubectl get secret mysql-ssl-cert -n <namespace> -o yaml
```

The following is a sample secret containing the certificates:

```
apiVersion: v1
  kind: Secret
  metadata:
    name: azure-mysql-ssl-cert
    namespace: dev
  type: Opaquedata:
    BaltimoreCyberTrustRoot.crt: LS0tLS1CRU ... 0tLS0tCgo=
    DigiCertGlobalRootG2.crt: LS0tLS1CRUdJTiBDRVJ ... RS0tLS0tCg==
```

See the Configuring the Environment Variables section for more information.

To troubleshoot issues related to db ssl connection, see here.

Information Stored in the Symantec Identity Security Platform Database

Learn about the user-account related information that Symantec Identity Security Platform may store in its database. These details are required by the solution to run its operations.

User information is stored in the database

The following user account-related details are stored by the solution in the database:

- User information such as the login identifier, first name, and last name may be stored if so configured administratively for the purpose of sending messages
- Status of the user (active or inactive)
- Email Addresses and Phone numbers in encrypted form may be stored if configured administratively for the purpose of using as authentication credentials
- Credential details include public certificates for PKI-based credentials (like FIDO2 devices), Push, TOTP in encrypted form
- User's Trusted Device details: As part of this, the IDSP saves a unique identifier known as TDI, that is unique to the end user's device. Subsequently, the identifier is also stored in the user's device (either as a cookie or secure storage in mobile devices). Basic information about the user's browser or mobile OS is also stored.
- When the risk is enabled, the following details are stored in the database:
  - signature of the devices used by the users: The device signature is a collection of device-specific attributes like OS details, browser details, screen details, time zone, language details. Similarly, for mobile devices, the device signature consists of mobile OS details, device name, model, wifi mac address, screen size, etc. These are collected using js or mobile SDK that is invoked from client applications and passed to the IDSP server in encoded form.
  - IP address from where the users accessed the system if and as may have been provided by the network infrastructure
- The data extracted by the "ssp-data" Helm chart, which is used to load the network watchlist and fraud IP data into the IDSP database for evaluating risk, is stored in the IDSP database.

  NOTE
  When using the Internal Identity Store, the entire user profile, including custom defined attributes, is stored in the database.

Risk evaluation data stored in the database

The data extracted by the "ssp-data" Helm chart, which is used to load the network watchlist and fraud IP data into the IDSP database for evaluating risk, is stored in the IDSP database.

Configuring Connectivity to Amazon Web Service (AWS) MySQL DB Without Using a DB Password

Learn the process of setting up connectivity to an AWS MySQL DB without the need for a database password.

This topic includes the following sections:

- Amazon EKS Pod Identity Prerequisite
- Verifying Access to the MySQL DB Using Access Token (AT)
- Deploying Symantec Identity Security Platform using AWS Pod Identity for database identification

Amazon EKS Pod Identity Prerequisite

In the following section, we have provided a few step highlights that would help you while you perform those steps before deploying Symantec Identity Security Platform:

> **NOTE**
> The following instructions use ***eksctl***- A command-line tool for working with EKS clusters that automates many individual tasks. The instructions require that you use version 0.62.0 or later. Refer to this link to install The eksctl command line utility.

1. **Connect to your AWS MySQL database** and create a ***DB user*** (that uses **AWSAuthenticationPlugin** for authentication).

   > **NOTE**
   > Before running the following command,
   >
   > - Replace <DB_HOST_FQDN> with the MySQL DB host FQDN.
   > - Replace <DB_PORT> with the MySQL DB port.
   > - Replace <DB_ADMIN_USER> by the database admin user name.
   > - Replace <DB_ADMIN_PASSWORD> by the database admin password

   a. Connect to your AWS MySQL DB instance using the database's *admin* user. You can either use the GUI approach or a command-line approach as specified below:

   ```
   kubectl run -it --rm --image=mysql:8.0.26 --restart=Never mysql-client -- mysql --enable-cleartext-
   plugin --host=<DB_HOST_FQDN>  --port=<DB_PORT> --user=<DB_ADMIN_USER>  --password=<DB_ADMIN_PASSWORD>
   ```

   b. Run the following MySQL command to create the DB schema (if not already created), the DB USER, and grant access permissions to the user (the following example uses a db name of "**iamauth**" and db user named, "**iamauth**"):

   ```
   CREATE DATABASE IF NOT EXISTS iamauth;

   CREATE USER IF NOT EXISTS iamauth IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';

   GRANT ALL ON iamauth.* TO iamauth;

   SELECT user FROM mysql.user;

   exit;
   ```

2. **Create a DB connect IAM Policy that allows connection to DB with IAM Authentication**.

   > **NOTE**
   > Before running the following command,
   >
   > - Replace <AWS_REGION> with the AWS region of the MySQL DB.
   > - Replace <AWS_ACCOUNT_ID> with the AWS Account ID.
   >   To get the AWS Account info, use this command:
   >   ```
   >   aws sts get-caller-identity
   >   ```
   > - Replace <AWS_DB_CLUSTER_RESOURCE_ID> with the DB cluster resource ID.
   >   To get DB Clusters Name and ID, use this command:
   >   ```
   >   aws rds describe-db-clusters --query "DBClusters[*].
   >   [DBClusterIdentifier,DbClusterResourceId]"
   >   ```

   a. **Create a policy.json file** with the following content:

   ```
   {
      "Version": "2012-10-17",
      "Statement":
      [
          {
   ```

```
            "Effect": "Allow",
            "Action": [ "rds-db:connect" ],
            "Resource": [ "arn:aws:rds-
db:<AWS_REGION>:<AWS_ACCOUNT_ID>:dbuser:<AWS_DB_CLUSTER_RESOURCE_ID>/iamauth"]
        }
    ]
}
```

NOTE
If you want to configure AWS KMS for password encryption, then add the following action, along with the AWS MySQL DB action in the above policy.json file.

```
"Action": [
        "kms:ListKeys",
        "kms:Encrypt",
        "kms:Decrypt"
          ],
```

See Configuring Connectivity to AWS KMS for Password Encryption

b. **Create the IAM Policy** using this command:

```
aws iam create-policy --policy-name ssp-aws-policy --policy-document file://policy.json
```

3. **Create an OIDC provider for your Kubernetes EKS cluster**.
   a. Get the OIDC provider issuer name for your EKS cluster:

```
aws eks describe-cluster --name <cluster-name> --query "cluster.identity.oidc.issuer" --output text
```

   b. View your existing OIDC providers to see if there is one with the ID matching the ID for the above OIDC provider issuer name:

```
aws iam list-open-id-connect-providers
```

   c. If the OIDC provider does not exist, create one (see AWS documentation for more information):

```
eksctl utils associate-iam-oidc-provider --cluster <cluster-name> --approve
```

   d. Validate that your OIDC provider got created by listing the providers again:

```
aws iam list-open-id-connect-providers
```

4. **Create a Kubernetes Service Account and an AWS IAM role with the DB Connect Policy.**
This step associates the Kubernetes service account with the IAM Role. After performing this step, any pod that is running as this service account would be able to request an access token to connect to the database.
This step does the following:

- creates a Kubernetes Service account
- creates an IAM Role and assigns the DB Connect Policy to the role
- links the Kubernetes service account with the IAM role

NOTE
Before running the commands in this step,

- Make sure to define the NAMESPACE environment variable containing the name of the Kubernetes namespace where your IDSP is going to be deployed. See Configuring the Environment Variables section.
- Replace <cluster-name> with the name of your EKS cluster.

```
eksctl create iamserviceaccount \
  --name ssp-eks-iam-sa \
  --namespace ${NAMESPACE} \
  --cluster <cluster-name> \
  --attach-policy-arn arn:aws:iam::688201620683:policy/ssp-aws-policy \
```

```
--approve \
--role-name cluster-<cluster-name>-ssp-eks-iam-sa
```

Verifying Access to the MySQL DB Using Access Token (AT)

Validate that the AWS Pod Identity is properly deployed and configured before deploying the IDSP helm chart using the following command:

NOTE
Before running the commands in this step,

- Make sure to define the NAMESPACE environment variable containing the name of the Kubernetes namespace where your IDSP is going to be deployed. See Configuring the Environment Variables section.
- Replace <DB_HOST_FQDN> with the MySQL DB host FQDN.
- Replace <DB_PORT> with the MySQL DB port.
- Replace <AWS_REGION> with the AWS region of the MySQL DB.
- The following example uses the db user as "iamauth".

1. Start a MySql pod running as the Kubernetes service account created in step 4 above (in this example, the service account used is "**ssp-eks-iam-sa**"):

```
kubectl run -n ${NAMESPACE} -it --rm --image=mysql:8.0.XX --restart=Never --overrides='{ "spec":
 { "serviceAccount": "ssp-eks-iam-sa" }  }'  mysql-client -- bash -c "apt update -y; apt install curl
 zip -y; curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip" ; unzip
 awscliv2.zip ; ./aws/install; bash"
```

2. While inside the container shell, use the command "**env**" to ensure that the AWS_ROLE_ARN, and AWS_WEB_IDENTITY_TOKEN_FILE environment variables are defined in the container's shell environment.

```
env
```

3. Acquire an access token for the MySQL DB using the following command:

```
export AT="'aws rds generate-db-auth-token \
            --hostname <DB_HOST_FQDN> \
            --port <DB_PORT> \
            --region <AWS_REGION> \
            --username iamauth'"

echo ${AT}
```

4. Use the access token retrieved in the previous step to authenticate to your MySQL database:

```
mysql --enable-cleartext-plugin --host=<DB_HOST_FQDN> --port=<DB_PORT>  --user=iamauth  --
password="${AT}"
```

Deploying IDSP using AWS Pod Identity for database identification

1. Create a *Kubernetes Secret* containing the **IDSP bootstrap metadata** to instruct IDSP how to acquire the Access Token required to connect to the Operational DB (resource type OP_DB).

NOTE

- If the metadata for this resource type cannot be located, IDSP falls back to using a regular DB password, if provided.
- Before running the commands in this step, make sure to define the NAMESPACE environment variable containing the name of the Kubernetes namespace where your IDSP is going to be deployed. See Setting Up SSP Services section.

- Replace <DB_HOST_FQDN> with the MySQL DB host FQDN.
- Replace <DB_PORT> with the MySQL DB port.
- Replace <DB_REGION> with the DB region, for example, - "us-east-1"
- The following example uses the db user as "iamauth".

a. Create a file (for example, aws_rds_iam_auth_pod_identity_md.json) with the following content:

```
[
  {
    "version": "1",
    "comment": "Native AWS Aurora MySQL Identity",
    "resourceType": "OP_DB",
    "responseType": "AWS_RDS_SDK",
    "url": "",
    "method": "",
    "headers": [],
    "params": [
      "region=<DB_REGION>",
      "host=<DB_HOST_FQDN>",
      "port=<DB_PORT>",
      "user=iamauth"
    ],
    "azType": "NONE",
    "azBasicAuthUsername": "",
    "azBasicAuthPassword": "",
    "azBearer": ""
  }
]
```

b. Create a Kubernetes Secret with the content of the preceding file, **make sure the name of the file in the Kubernetes Secret is *"bootstrapmetadata.json"***. The following command uses the example file name, "aws_rds_iam_auth_pod_identity_md.json" (created in the previous step):

NOTE

- Before running this command, ensure that the variables, ${RELEASENAME}, ${NAMESPACE} are set as mentioned in step 1 of Configuring the Environment Variables section.
- The following command uses ${RELEASENAME}-ssp-bootstrapmetadata for the name of the Kubernetes secret. You can use any name to the Kubernetes secret, however, ensure that the secret name is provided during the IDSP helm chart deployment (--set ssp.global.bootstrapmetadata.existingSecret).

```
kubectl create secret generic ${RELEASENAME}-ssp-bootstrapmetadata -n ${NAMESPACE} --from-file=bootstrapmetadata.json=aws_rds_iam_auth_pod_identity_md.json
```

2. Deploy the IDSP chart following the steps mentioned in the Configuring the Environment Variables section. When deploying the IDSP helm chart and the ssp-data chart , use the following additional parameters, making sure to set the DB host, port, and username appropriately for the database schema created in step 1(b) of the "AWS Pod Identity Prerequisite" section and instruct IDSP pods to run as the Kubernetes Service Account created in the preceding steps, and set the DB password to an empty string:

– Pass the DB username created in step (1) of the "AWS Pod Identity Prerequisite" section.

```
--set ssp.db.user="iamauth"
```

– Pass the secret name containing the bootstrap metadata JSON file (created in the previous step)

```
--set ssp.global.bootstrapmetadata.existingSecret=${RELEASENAME}-ssp-bootstrapmetadata
```

NOTE

When you configure IDSP to use bootstrap metadata, the access token retrieved by it will be used in place of a password to connect to your MySQL database. When passing the " --

set ssp.global.bootstrapmetadata.existingSecret", make sure to set the DB password to an empty string: --set ssp.db.password=""

– Ensure to specify existingServiceAccount to instruct IDSP pods to run as the Kubernetes Service Account created in the preceding steps:

```
--set ssp.global.existingServiceAccount="ssp-eks-iam-sa"
```

Example: The following parameters are required to set up an IDSP to connect to the AWS MySQL DB using AWS Pod Identity:

NOTE
Before running the commands in this step,

- Replace <BOOTSTRAP_METADATA_SECRET_NAME> with the file name created in "step 1(a)" of Deploying IDSP using AWS Pod Identity for database identification section.
- Replace <DB_HOST_FQDN> with the MySQL DB host FQDN.
- Replace <DB_PORT> with the MySQL DB port.
- The following example uses the db user and db name as "iamauth".

```
--set ssp.global.bootstrapmetadata.existingSecret="<bootstrap-metadata-secret-name>"
--set ssp.global.existingServiceAccount="ssp-eks-iam-sa" \
--set ssp.db.serviceHost="<DB-HOST>" \
--set ssp.db.servicePort="<DB-PORT>" \
--set ssp.db.name="iamauth" \
--set ssp.db.user="iamauth"
```

Reference Links to:

- Configuring Connectivity to AWS KMS for Password Encryption

Deploying the Services

Learn the sequence of steps to be followed to deploy Symantec Identity Security Platform in your cluster environment.

It is recommended that you follow the sequence of steps that are outlined here to deploy IDSP in your cluster.

1. **Deploy Ingress Controller** - The first step is to deploy a new or reuse an existing ingress controller to route the incoming traffic to different microservices.
2. **Deploy Enclave services** (also known as Observability services) - The second step is to deploy the enclave services. These are a group of external services (Elastic Search, Kibana, Prometheus, Grafana) that are plugged into our solution to collect information that is related to searching, logging, and monitoring from microservices that are running within the IDSP cluster.
   Deploy IDSP solution, after you have successfully completed the deployment of the preceding services. You can deploy the solution in your Cloud environment, or OpenShift. Before deploying, ensure that you check the **Pre-installation checklist**.
3. **Configure Enclave Services** - Perform this step to configure the enclave services to display IDSPcontent.
4. **Post Installation Steps** - Consider the tasks in the checklist after deploying the IDSP helm chart.
5. ***Configure Administrative Access*** - Covers the initial security credentials for the admin clients (Infra client, System client, and Default Tenant client) and admin user (defaultTenantAdminUser) that are provided out-of-the-box by IDSP during the initial deployment

Before going into the specifics of the deployment steps and getting the solution up and running, refer to the following table to view the external components and their capabilities that are part of the deployment process.

Software	Description	Chart Name	
NGINX Ingress Controller	An ingress controller is responsible for reading the ingress resource information and routing the incoming traffic to different microservices.	ingress-nginx	https://kubern
Elastic Operator	Elastic Cloud on Kubernetes (ECK) is a Kubernetes operator to orchestrate Elastic applications (Elasticsearch, Kibana, and so on) on Kubernetes.	eck-operator	https://helm.e
Elasticsearch	Elasticsearch is an open-source full-text search and analytics engine. It allows to store, search, and analyze logs.	elasticsearch	https://helm.e
Kibana	Kibana is an open-source analytics and visualization platform that is designed to work with Elasticsearch. It helps in analyzing and monitoring the logging information.	Kibana	https://helm.e
Prometheus Operator	Prometheus is an open-source monitoring and alerting toolkit for containers and micro-services.	kube-prometheus	https://charts.
Grafana Operator	A Kubernetes Operator for creating and managing Grafana instances.	grafana-operator	https://charts.

NOTE
Ensure that you review the software and cluster infrastructure requirements under the Prerequisites section before going ahead with the deployment.

Next Step

Deploying Ingress Controller

Pre-Installation Checklist

Consider the tasks mentioned in this checklist before deploying the Symantec Identity Security Platform helm chart.

Before you begin installing IDSP helm charts, prepare, configure, and review the following information, and perform the pre-installation tasks as required.

#	Pre-installations Tasks
1.	Check the entropy in your cluster to ensure the cluster has sufficient performance. See Checking Entropy Level.
2.	Prepare the private image registry. If your organization requires the IDSP helm chart to use images from your local image registry. Push the required images to your local image registry, see Using Local Image Registry. 1. If your image registry requires authentication, create a secret of type, docker-registry. See the Additional parameters section.
3.	Ensure that you go through the IDSP deployment guidelines to help you understand your requirement and plan your deployment so, see Sizing the Deployment.

4.	Ensure that the Ingress is deployed. See Deploying Ingress section. If you want a separate ingress for management, see Using an Existing Ingress Controller.
5.	**(Optional)** To set up the enclave (external) services offered by IDSP to be deployed in it is Kubernetes cluster and gather data for searching, logging, and monitoring from tools such as Kibana, Prometheus, and Grafana, see Deploying Enclave Services. If you want to use your own observability services, see Managing Observability. Before deploying IDSP, we recommend you to deploy the enclave services. However, if you must deploy the enclave services following the deployment of IDSP, you can do so by re-running the IDSP helm chart that makes the necessary adjustments. Broadcom team can assist you, if needed.
6.	Identify the host fqdn to be assigned to the IDSP and register it in the DNS (using the name and Ingress IP). See Constructing the Service FQDN.
7.	Obtain signed trusted certificates from your CA for external access. See the point on trusted certificates in the Additional parameters table.
8.	Choosing a database (internal, external), getting the db certificate, and create a secret. See Database Consideration. For MySQL DB, make sure that the db user used for IDSP deployment is not limited in "max_questions" and "max_queries". See DB user account note.
9.	Advanced Configuration: • Using your own TLS certificate for the internal cluster communication. See Using Custom CA for Intra Cluster SSL Communication. • If using an existing IDSP operational DB, configure an existing MEK secret for connecting to the database. See Master Encryption Key. • If you intend to update parameters, such as additionalLabels, nodeSelector, affinity, tolerations, or AWS AWS_STS_REGIONAL_ENDPOINTS for IDSP deployment, see Meeting Topology Requirements section.
10.	Check which container runtime is used in your cluster to set the fluent-bit parser mode. See the Table with additional parameters section.
11.	Configure your firewall to allow access to the following services: • enable access from the Internet to the end-user ingress, and (when using a separate ingress controller for system management) from the internal network to the mgmt ingress. • if you intend to use an LDAP ID store or SCIM ID store, ensure you allow access from the cluster to your id store. • If you intend to use SMTP, Twilio, or VIP Access Push to deliver email, SMS, or push messages from IDSP, ensure that your cluster has access to these services. See Required Ports in IDSP.
12.	When using LDAP Identity Store, IDSP supports the use of LDAP Connection Pool, see LDAP Connection Pool Considerations section.
13.	Review the Security Considerations section to know about various security related information that will help you in taking better decisions during the deployment process. See Security Considerations.
14.	Once IDSP has been deployed, the ***ssp-data helm*** chart may be optionally deployed, however it is necessary in the following cases. • when IA Risk service is operational (operational by default) • when location information associated with the user/client IP address is required in the audit events See Installing the ssp-data helm chart.
15.	Ensure that your network components are configured to preserve the client IP address, for IP-based and Country-based policies to function correctly. See Verifying the deployment section.

Required Ports in Symantec Identity Security Platform

Know about the port numbers and their respective purposes used by the Symantec Identity Security Platform solution.

This page provides a list of port numbers and their corresponding network traffic used by the IDSP solution. Opening these ports is essential to facilitate communication through the firewall, enabling the applications to function properly. If the port numbers differ from the default values, adjust the firewall configuration accordingly to ensure compatibility.

Table 30: Network Traffic Ports

Port Description	Default Port Numbers	Purpose	Inbound/ Outbound
Ingress (controller)	443	Enables communication with IDSP APIs, Admin Console, Signin UI, Kibana, Grafana, etc.	Inbound
External Database	• Mysql - 3306 • Postgress db - 5432	Enables communication between IDSP and External Database.	Outbound
External Service Provider	SMS • *Clicktell • Twilio- 443 Email • SMTP - 25, 587, 465 • *Emailie • *Twilio SendGrid • Corporate mail server - 25 IVR • 443	Enables communication between IDSP and external SMS, Email provider, IVR - required for sending OTP messages, password recovery messages, etc.	Outbound
Default Push and TOTP Provider	Default Push Provider - 443 (https://userservices-auth.vip.symantec.com/)	Enables communication between IDSP and VIP Symantec (Default) Push provider.	Outbound
External Push Provider	Custom Push - 443	Enables communication between IDSP and external Push provider.	Outbound
External Risk Engine	443	Enables communication between IDSP and external Risk Engine	Outbound
External LDAP Identity Store	389	Enables communication between IDSP and external LDAP user store	Outbound
External SCIM Identity Store	443	Enables communication between IDSP and external SCIM user store.	Outbound
Elastic Search	9200, 5601	Enables communication between Fluentbit and Elastic Search, if using elastic search for log storage.	Outbound
Siteminder Access Gateway	443	Enables communication between IDSP and Siteminder Access Gateway (REST API), if using SiteMinder Password services.	Outbound

NOTE
*If you are using services such as external IDP, Webhooks, or STS, and these services use non-standard ports, ensure that you open those ports for outbound connections form the cluster.

Table 31: Inter Pod Communication Ports

Port Description	Default Port Numbers	Purpose	Inbound/Outbound
IDSP services (REST API)	443	Enables communication to IDSP services from the Ingress(controller) and other IDSP services.	Inbound / Outbound
Hazelcast	5701	Enables communication between IDSPservices and the Hazelcast cluster.	Inbound
Hazelcast nodes inter-communication	5702	Enables communication between Hazelcast cluster member to member.	Inbound / Outbound
Internal Database	Mysql (demo db) - 3306	Enables communication between IDSP and Internal DB.	Inbound
IDSP Custom Authenticator (SPI webhook)	443	Enables communication between IDSP and Custom Authenticator	IDSP service - Outbound SPI webhook - Inbound
Jaeger	6831	Enables communication between IDSP and Jaeger	Inbound
Fluentbit	2020	Enables communication to Fluent Bit's built-in HTTP Server used to query internal information and monitor metrics.	Inbound

Checking Entropy Level in the Kubernetes Cluster

Learn how to check the entropy level in your Kubernetes cluster to avoid slowness in the performance of the Symantec Identity Security Platform solution.

Running IDSP in your Kubernetes cluster requires entropy for crypto and random number generation. You should confirm that the entropy level of your Kubernetes cluster nodes is sufficient. Lack of entropy may lead to slowness in the solution's performance.

Before deploying IDSP, ensure that the nodes in the cluster are able to generate random numbers in adequate rate to avoid performance issues in IDSP microservices.

To check the rate at which random numbers can be generated in the cluster nodes, use the following command:

```
kubectl run busybox --image=busybox:1.28 --rm -ti --restart=Never --command -- dd if=/dev/random of=/dev/null
 bs=1 count=1M
```

Or for older versions of the dd command use:

```
kubectl run busybox --image=busybox:1.28 --rm -ti --restart=Never --command -- time dd if=/dev/random of=/dev/
null bs=1 count=1M
```

The expected time for generating 1 Megabyte of random data should be about 2 seconds.

For example, the following command uses one of the Prometheus pods to execute the dd command:

NOTE

- This command assumes that Prometheus is deployed in the monitoring namespace. If the Prometheus is not deployed in the monitoring namespace, then replace "monitoring" with the correct namespace.
- The Prometheus-node-exporter is a DaemonSet and runs on all the nodes in the Kubernetes cluster. Therefore, executing the following command would run the dd tool on every node in the Kubernetes cluster.
- Depending on your Prometheus deployment, the labels set on your pods may be different. The following command uses the parameter, `--selector='app=prometheus-node-exporter'` which uses the label 'app=prometheus-node-exporter'. However, your environment may use the label "app.kubernetes.io/name=node-exporter" or even a different label. To check the labels of a specific pod run: `kubectl get pod -n monitoring prometheus-operator-node-exporter-td9x8 --show-labels`

```
(NAMESPACE=monitoring; kubectl get pods \
--namespace="${NAMESPACE}" \
--selector='app=prometheus-node-exporter' \
--output=name | while read POD; do echo "${POD}"; kubectl exec --namespace="${NAMESPACE}" "${POD}" \
-- dd if=/dev/random of=/dev/null bs=1 count=1M; echo; done)
```

Increasing the Entropy in the Kubernetes Cluster

The following methods can be used to increase the entropy at each of the ***Kubernetes nodes*** in your Kubernetes Cluster:

- If you are using a hosted Kubernetes service, consult with your cloud provider to find out if the entropy could be generated or the cluster nodes kernal could be updated so that entropy would not be required for random number generation.
- On Linux Kernel 5.6, /dev/random and getrandom do not require running an entropy generator. Consider upgrading your Kubernetes nodes to a Linux OS that uses kernel 5.6 or higher. For example, upgrading to Ubuntu 20.04.2 (which uses a Linux kernel newer than 5.6) should solve the entropy issue.
- Run an **Entropy Gathering Daemon (EGD)** that helps provide entropy to the Linux kernel. EGDs support multiple entropy sources including hardware sources. Running an EGD (such as rngd and havened) on each Kubernetes node in your cluster helps increase your cluster's entropy.
- If you are unable to change your Kubernetes nodes, you can choose to run an entropy generator as a DaemonSet in your Kubernetes cluster. The following is an example of such a deployment:

  NOTE

  - The image used in the following example is a third-party image (hortonworks/docker-haveged) and the pod would run in privileged mode to allow it access to the Kubernetes node it is running on. Therefore, before using this example in your Kubernetes cluster make sure to exercise caution and perform due diligence.
  - The tolerations section in the following DaemonSet manifest allows the pods of the DaemonSet to run on all nodes including the ones with a NoSchedule taint (for example, master nodes on BareMetal clusters).

```
$ cat <<EOF > haveged-ds.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  namespace: kube-system
  labels:
    run: haveged
  name: haveged
spec:
  selector:
    matchLabels:
      run: haveged
  template:
    metadata:
```

```
      labels:
        run: haveged
    spec:
      containers:
      - image: hortonworks/haveged:1.1.0
        name: haveged
        securityContext:
          privileged: true
      tolerations:
      - effect: NoSchedule
        operator: Exists
EOF

kubectl apply -f haveged-ds.yaml
```

Using Private Image Registry

Steps to pull images from Broadcom registry, push them to your private registry, and deploy Symantec Identity Security Platform using the images stored in your private registry.

A private image registry is used for enhanced security, as it provides you with more control and security over your image repositories. With a private image registry, you can ensure that only authorized users can push images to the registry, scan the images for vulnerabilities regularly so that it meets your company's security standards.

Migrating the images from the Broadcom image registry to your private image registry includes the following steps:

- Pulling images from the Broadcom registry
- Pushing images to your Private registry
- Deploying IDSP using the images stored in your private registry

Pulling images from Broadcom registry

1. To pull the IDSP images, authenticate to the Broadcom image registry (securityservices.packages.broadcom.com) . The following command is an example that shows how to authenticate to the Broadcom image registry using docker.

   ```
   docker login \
       --username "<BROADCOM_REGISTRY_USERNAME>" \
       --password "<BROADCOM_REGISTRY_PASSWORD>" \
       'securityservices.packages.broadcom.com'
   ```

   Where,

   – Password is the registry access token obtained from the support portal.

   The following are the docker pull images retrieved from the Broadcom image registry:

- docker pull securityservices.packages.broadcom.com/admin-svc:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/authmgr:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/factors-svc:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/oidc-svc:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/opa:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/authswagger:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/ssp-db-initializer:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/authhub-adminconsole:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/authhub-signin:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/authhub-signin-legacy:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/iaservice:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/ssp-vault-secrets-initializer:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/identity-svc:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/ssprisk-svc:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/ui-helper-svc:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/geolocation-svc:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/authhub-selfserviceconsole:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/machineid-svc:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/proxysidecar:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/ssprouter:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/hazelcast-enterprise:5.7.0-p0

The network data image contains fraud details and is released every week. There may be a newer version than the following version:

- docker pull securityservices.packages.broadcom.com/network-data-loader:2026.20

The following are the optional images used for the sample applications:

- docker pull securityservices.packages.broadcom.com/sample-rp:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/sample-app:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/spi-demo:4.0.2.1075
- docker pull securityservices.packages.broadcom.com/cadir:4.0.2.1075

Pushing images to your private registry

> **NOTE**
> While migrating the images from the Broadcom registry to your private registry, it is recommended to keep the same base path for all the IDSP images, image names and keep the original image tags. Doing this allows you to use the IDSP Helm chart parameter, "***imageRepositoryBase***" to configure IDSP chart to use the images from your private registry.

To push images to your private registry, follow these steps:

1. Tag the images to include your private image registry name.
   The following command is an example that shows how to tag the images before pushing them to the private image registry using the docker command:
   ```
   docker tag "<IMAGE_FULL_NAME>" "<PRIVATE_REGISTRY>/<IMAGE_FULL_NAME>"
   ```
   Where, PRIVATE_REGISTRY - this is the full name of the private registry image where you are storing the IDSP images. It is recommended to keep the origin of the image as part of the image path. An example of an image full name in the private registry: "images.example.com/securityservices.packages.broadcom.com/admin-svc:1.0.3211".
2. Push the images to the private registry using the new tag name.
   The following command is an example that shows how to push the images to the private image registry using the docker command:

```
docker push "<PRIVATE_REGISTRY>/<IMAGE_FULL_NAME>"
```

Deploying IDSP using the images stored in your private registry

To deploy IDSP using the images stored in your private image repository, add the following commands to the Deploying IDSP Helm Chart command:

```
--set ssp.global.ssp.registry.imageRepositoryBase="<PRIVATE_REGISTRY>/securityservices.packages.broadcom.com/"
 \
--set hazelcast-enterprise.image.repository="<PRIVATE_REGISTRY>/securityservices.packages.broadcom.com/
hazelcast-enterprise"
```

NOTE

- This configuration parameter assumes that all images are stored in a common base path.
- The "`--set ssp.global.ssp.registry.imageRepositoryBase` " command should be used when deploying the IDSP Infra helm chart, IDSP (ssp) helm chart, ssp-data helm chart, and Sample Web Application helm chart.
- The "`--set hazelcast-enterprise.image.repository` " command is needed ONLY when deploying IDSP helm chart. It is not required for deploying the IDSP Infra, ssp-data, or Sample Web Application helm charts.

Using Image Digest in Symantec Identity Security Platform Charts

Symantec Identity Security Platform charts' values.yaml contains both the image tag and the image digest of each image. Learn to use Image Digest to have the correct version of a container image in your environment.

As image tags are mutable, and multiple images can share the same tag, this makes it difficult to track which version of the container image is running in the cluster. To ensure that a pod is using the correct version of a container image, you can specify the image digest. The image digest uniquely identifies a specific version of the image. Therefore, to improve security, image digest (hash value) is preferred over using image tag.

This topic includes the following sub-topics:

- Using Image Digest
- Enabling useImageDigest Setting
- Updating the values.yaml with the Image Digest

You can refer to an image in a Kubernetes pod by using the image tag or by using the image digest:

Example of Image tag:

```
image: securityservices.packages.broadcom.com/admin-svc:1.0.3160
```

Example of Image digest:

```
image: securityservices.packages.broadcom.com/admin-svc:1.0.3160@sha256:aa8d179c015b01ef...
 985acd76b3e019189c88cb9b9510d0f1
```

NOTE
The separator between the image repository and the image tag is (:), whereas the separator between the image repository and the digest is (@).

Using Image Digest

IDSP charts support the use of image digest. The chart's values.yaml indicates the image for every component and for every image the chart's values.yaml specifies the image repository, tag, and digest (hash).

For Example:

```
admin:
```

```
imageRepository: "securityservices.packages.broadcom.com/admin-svc"
imageTag: 1.0.3170
imageDigest: "sha256:7a5538f3a3bcaf24eb7fd37e889078879651a6a02bc8ba85d01d570fdf43d6e7"
```

By default, IDSP charts' values.yaml contains both the image tag and the image digest of each image.

Enabling useImageDigest Setting

To use the image digest in the IDSP helm charts, set the global setting, "ssp.global.useImageDigest" to true in IDSP as shown in the following example:

```
ssp:
global:
useImageDigest: true
registry:
  credentials:
    password: pwd
    username: user
...
```

The IDSP helm charts use the following logic to determine whether to use an image digest, image tag, or the chart's AppVersion:

- If the "useImageDigest" parameter is set to true and the image digest is available, IDSP helm charts use it.
- If there is no image digest, the charts look for the image tag. IDSP helm charts use the Image tag if it is available.
- IDSP helm charts use the chart's AppVersion as the image tag if no image tag is available.

> **NOTE**
> To use the image digests for Hazelcast, you should set the hazelcast-enterprise.image.tag to null or "" as it is part of the Hazelcast Enterprise sub chart as shown in the following example: hazelcast-enterprise:
>
> ```
> image:
>    repository: "securityservices.packages.broadcom.com/hazelcast-enterprise"
>    tag: "5.1.3-ssl-D9P5"
>    # HC uses imageDigest only if image tag is empty (pass tag as "")
>    digest: "sha256:e653a5b73a515d553616f635939d8014c576e77920e5bad05f36a6553236482a"
> ```

Updating the values.yaml with the Image Digest

You can use the following command to retrieve the image digest of a container image:

```
export imageDigest =$(docker inspect --format='{{index .RepoDigests 0}}' us.gcr.io/saasdev-sed-ssp-hp/spi-demo:1.0.uc604043-1|cut -d'@' -f 2)
```

The following command can be used to update the values.yaml: This particular example uses: yp from: here.

```
yq -i ".ssp.factor.imageDigest = "$imageDigest"" 'ssp/values.yaml'
```

Ingress Overview

Symantec Identity Security Platform provides an Ingress resource that can be read by NGNIX Ingress controller to route the incoming traffic to the different microservices. The Ingress resource contains the rules that govern the traffic to the microservices.

IDSP supports different sets of operational APIs that are represented by endpoints that are exposed through an Ingress:

- **System management operations**: This set includes all the API endpoints that manage the infrastructure of the system such as managing the signing keys, managing the system-wide settings, creating tenants, exposing the

Enclave services. These operations execute in the context of the "system" tenant. The system management operations are mostly restricted to an internal audience.

- **Tenant management operations**: This set includes all the API endpoints that administer the security artifacts of a tenant such as policies, applications. These operations are multi-tenant. The tenant management operations can be exposed to an internal and/or external audience.
- **End user operations**: This set includes all the API endpoints that authenticate tenant end users and support the self-service credential management by tenant end users. These operations are multi-tenant. The end-user operations are typically exposed to an external audience.

Ingress Partitioning

IDSP provides different out-of-the-box Ingress objects to expose the system management and tenant management operations on the same Ingress controller. Due to security reasons as described in Security Considerations, you can choose to expose these operations on different Ingress controllers too.

As the end-user operations do not have an out of the box Ingress object defined, you can create such an Ingress object to expose this set of operational APIs *after deploying* IDSP. These operations are tied to a tenant vanity URL.

In a production environment, you can choose to expose the operational APIs sets using the following Ingress partitioning options:

- Expose system management and tenant management operations on the same Ingress controller on an internal network, while exposing end-user operations on a different Ingress controller on an external network.
- Expose system management on an Ingress controller on an internal network, while exposing the tenant management operations and end-user operations on a different Ingress controller on an external network.

Reference Links:

- Deploying one or more Ingress controllers, see Deploy Ingress Controller.
- Deploying Nginx Ingress controller in IDSP Namespace, see here.
- Deploying Ingress Controller in OpenShift Cluster, see here.
- Managing Custom Vanity Ingress, see Managing Custom Vanity Ingress.
- How Ingress works in Kubernetes, see Kubernetes.

Deploying Ingress Controller

Learn how to deploy an Ingress Controller in your cluster environment.

The default ingress deployment of Symantec Identity Security Platform exposes the system management and tenant management operations on the same Ingress Controller. Due to security reasons as described in Security Considerations, you can choose to expose these operations on different Ingress Controllers too.

This topic includes the following:

- Using an Existing Ingress Controller
- Deploying a New Ingress Controller
- Retrieving Ingress IP
- Setting "topologySpreadConstraints" for Ingress deployment

Using an Existing Ingress Controller

IDSP requires that you have an ingress controller of type, NGINX. The Ingress Helm Chart should be of version 4.14.3 and the Ingress-Nginx app version should be v1.14.3.

If you have an existing ingress controller that meets the preceding requirement, you can skip the ingress controller deployment steps that are mentioned in the following section **(Using a New Ingress Controller)** and use the ingressClassName during the IDSP deployment.

Your existing ingress controller deployment is associated with an Ingress Class. The ingress class name of your ingress controller should be used when deploying the IDSP chart. By default, the IDSP uses the **ingressClassName**, "nginx". In case, your ingress controller uses a different class name, specify it during IDSP (SSP helm chart) deployment using the following parameter:

```
--set ssp.ingress.ingressClassName=<ingress-class-name>
```

Deploying a New Nginx Ingress Controller

The following procedure describes how to deploy the Nginx ingress controller. The procedure also includes information about the parameter that must be set to expose the tenant management operations on a different ingress controller.

Follow these steps:

1. Create an ingress namespace.

   ```
   kubectl create ns ingress
   ```

2. Create a secret if you want to use specific credentials to pull images from the container registry. The following example shows how to create a secret for the Docker Hub registry with a name - "***docker-hub-reg-pullsecret***":

   ```
   kubectl create secret docker-registry docker-hub-reg-pullsecret -n <namespace> \
   --docker-server="https://index.docker.io/v2/" \
   --docker-username="<dockerhub-username>" \
   --docker-password="<dockerhub-password>"
   ```

   NOTE

   Docker Hub applies rate limiting to anonymous and free authenticated accounts for pulling images. If your deployment is going to pull images from Docker Hub, we recommend that you provide the deployment with your docker hub credentials. This is done by creating a secret and specifying it in the helm install command. For example, in the ingress-nginx Helm chart deployment command, add the following parameter:

   ```
   --set imagePullSecrets[0].name=docker-hub-reg-pullsecret
   ```

   where, docker-hub-reg-pullsecret is the name of a Kubernetes secret of type, docker-registry.

   If you exceed the limit for pulling images, your pod may not initialize and the pod gets into a state of "ImagePullBackOff". The following error messages may show up when you perform a kubectl describe pod:

   ```
   ERROR: toomanyrequests: Too Many Requests.
   OR
   You have reached your pull rate limit. You may increase the limit by authenticating and
    upgrading.
   ```

3. Locate the Helm chart and get the latest Helm chart updates.

   ```
   helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
   helm repo update
   ```

4. ***[Only for OpenShift]*** Create a Security Context Constraints (SCC) policy to allow Ingress to use the NET_BIND_SERVICE capability and runAsUser 101 (default controller user).

   ```
   cat <<EOF | kubectl apply -f -
   apiVersion: security.openshift.io/v1
   kind: SecurityContextConstraints
   metadata:
   name: nginx-ingress-scc

   priority: null
   users:
   - system:serviceaccount:ingress:ingress-nginx-admission
   - system:serviceaccount:ingress:ingress-nginx
   groups: []

   allowHostDirVolumePlugin: false
   ```

```
allowHostIPC: false
allowHostNetwork: false
allowHostPID: false
allowHostPorts: false
allowPrivilegeEscalation: true
allowPrivilegedContainer: false
allowedCapabilities: null
defaultAddCapabilities:
- NET_BIND_SERVICE
readOnlyRootFilesystem: false
requiredDropCapabilities:
- ALL
runAsUser:
type: MustRunAs
uid: 101
fsGroup:
type: MustRunAs
seLinuxContext:
type: MustRunAs
supplementalGroups:
type: MustRunAs
volumes:
- secret
EOF
```

In the next step, select the platform hosting the OpenShift environment by choosing either the Public or Virtual Private Cloud (VPC) steps.

5. Deploy Ingress Controller.

NOTE

- If deploying the ingress-nginx controller in an ***OpenShift*** cluster environment, add the following parameter to the ingress-nginx helm deployment command:

```
--set controller.admissionWebhooks.patch.securityContext.runAsUser=101 \
--set controller.admissionWebhooks.patch.securityContext.fsGroup=$(kubectl get ns ingress -o
 jsonpath='{.metadata.annotations.openshift\.io/sa\.scc\.uid-range}'|cut -d '/' -f1) \
--set controller.allowSnippetAnnotations=true
```

- To enable the ingress-nginx controller metrics in the ingress deployment for monitoring purpose, add the following parameters to the ingress-nginx helm deployment command:

```
--set controller.metrics.enabled=true \
--set controller.metrics.serviceMonitor.enabled=true \
--set-string controller.metrics.serviceMonitor.additionalLabels."release"="prometheus-
operator"
```

- If you are planning to deploy multiple ingress controllers in order to create network separation between system management operations, tenant admin operations, and end-user operations, ensure you assign each of these deployments a dedicated ingress-class-name by adding the following parameter to the ingress-nginx helm deployment command:

```
--set controller.ingressClassResource.name=<new-ingress-class-name> \
--set controller.ingressClassResource.controllerValue=<controller name (default "k8s.io/
ingress-nginx")>
```

- The deployment parameters used in the AWS (EKS) environment command require "AWS load balancer controller" to be deployed in the AWS EKS cluster. For more info see, Installing the AWS Load Balancer Controller add-on.

– ***Deploy Ingress Controller in a cloud environment.***

- Deploy an ingress controller that is **publicly accessible on the Internet** by using a Load Balancer with a public IP address.

  Command to run in Amazon (EKS) environment:

```
helm install ingress-nginx -n ingress ingress-nginx/ingress-nginx \
--set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-
scheme"="internet-facing" \
--set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-
type"="external" \
--set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-proxy-
protocol"="*" \
--set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-nlb-target-
type"="instance" \
--set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-target-group-
attributes"="preserve_client_ip.enabled=true" \
--set-string controller.config.use-proxy-protocol="true" \
--set-string controller.config.use-forwarded-headers="true" \
--set imagePullSecrets[0].name=docker-hub-reg-pullsecret \
--set-string controller.config.annotation-value-word-blocklist="load_module\,lua_package\,_by_lua
\,location\,root\,proxy_pass\,serviceaccount\,{\,}\,\'\,\\\\" \
--version=4.14.3 \
--set controller.allowSnippetAnnotations=true \
--set controller.service.externalTrafficPolicy="Local" \
--set controller.config.annotations-risk-level=Critical
```

  Command to run in Google Cloud - GKE environments:

```
helm install ingress-nginx -n ingress ingress-nginx/ingress-nginx \
--set-string controller.config.use-forwarded-headers="true" \
--set imagePullSecrets[0].name=docker-hub-reg-pullsecret \
--set-string controller.config.annotation-value-word-blocklist="load_module\,lua_package\,_by_lua
\,location\,root\,proxy_pass\,serviceaccount\,{\,}\,\'\,\\\\" \
--version=4.14.3 \
--set controller.allowSnippetAnnotations=true \
--set controller.service.externalTrafficPolicy="Local" \
--set controller.config.annotations-risk-level=Critical
```

  Command to run in Azure - AKS environments:

```
helm install ingress-nginx -n ingress ingress-nginx/ingress-nginx \
--set-string controller.config.use-forwarded-headers="true" \
--set imagePullSecrets[0].name=docker-hub-reg-pullsecret \
--set-string controller.config.annotation-value-word-blocklist="load_module\,lua_package\,_by_lua
\,location\,root\,proxy_pass\,serviceaccount\,{\,}\,\'\,\\\\" \
--version=4.14.3 \
--set controller.allowSnippetAnnotations=true /
--set controller.config.annotations-risk-level=Critical
```

  NOTE
  To define a specific IP address for the load balancer of the Ingress service, use the **--set controller.service.loadBalancerIP=<IP-address>** parameter in the helm install command. For information about using this parameter and reserving an IP address, see the documentation of your cloud environment.

  Example:

```
helm install ingress-nginx -n ingress ingress-nginx/ingress-nginx \
--set-string controller.config.use-forwarded-headers="true" \
--set imagePullSecrets[0].name=docker-hub-reg-pullsecret \
--set controller.service.loadBalancerIP=192.168.255.255 \
```

```
--set-string controller.config.annotation-value-word-blocklist="load_module\,lua_package
\,_by_lua\,location\,root\,proxy_pass\,serviceaccount\,{\,}\,\'\,\\\\" \
--version=4.14.3 \
--set controller.allowSnippetAnnotations=true \
--set controller.service.externalTrafficPolicy="Local" \
--set controller.config.annotations-risk-level=Critical
```

Command to run in VKS - (VCF with VMware Kubernetes Services) Environment:

```
helm install ingress-nginx -n ingress ingress-nginx/ingress-nginx
--set controller.allowSnippetAnnotations=true \
--set controller.service.externalTrafficPolicy="Local" \
--set-string controller.config.use-forwarded-headers="true" \
--set controller.service.type=LoadBalancer \
--set controller.service.loadBalancerIP=<IPADDR> \
--set controller.image.pullPolicy="IfNotPresent" \
--set-string controller.config.annotation-value-word-blocklist="load_module\,lua_package\,_by_lua
\,location\,root\,proxy_pass\,serviceaccount\,{\,}\,\'\,\\\\" \
--version=4.14.3 \
--set controller.config.annotations-risk-level=Critical \
--set controller.service.loadBalancerClass=ako.vmware.com/avi-lb
```

- **Deploy ingress controller that is only accessible by applications in the same VPC network and compute region.**

  Command to run in Google Cloud (GKE) environment:

```
helm install ingress-nginx -n ingress ingress-nginx/ingress-nginx \
--set controller.service.annotations."cloud\.google\.com/load-balancer-type"="Internal" \
--set controller.service.externalTrafficPolicy="Local" \
--set-string controller.config.use-forwarded-headers="true" \
--set imagePullSecrets[0].name=docker-hub-reg-pullsecret \
--set-string controller.config.annotation-value-word-blocklist="load_module\,lua_package\,_by_lua
\,location\,root\,proxy_pass\,serviceaccount\,{\,}\,\'\,\\\\" \
--version=4.14.3 \
--set controller.allowSnippetAnnotations=true \
--set controller.config.annotations-risk-level=Critical
```

  Command to run in Amazon (EKS) environment:

```
helm install ingress-nginx -n ingress ingress-nginx/ingress-nginx \
--set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-
internal"="true" \
--set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-
type"="external" \
--set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-proxy-
protocol"="*" \
--set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-nlb-target-
type"="instance" \
--set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-target-group-
attributes"="preserve_client_ip.enabled=true" \
--set-string controller.config.use-proxy-protocol="true" \
--set-string controller.config.use-forwarded-headers="true" \
--set imagePullSecrets[0].name=docker-hub-reg-pullsecret \
--set-string controller.config.annotation-value-word-blocklist="load_module\,lua_package\,_by_lua
\,location\,root\,proxy_pass\,serviceaccount\,{\,}\,\'\,\\\\" \
--version=4.14.3 \
--set controller.allowSnippetAnnotations=true \
```

```
--set controller.service.externalTrafficPolicy="Local" \
--set controller.config.annotations-risk-level=Critical
```

Command to run in Azure (AKS) environment:

```
helm install ingress-nginx -n ingress ingress-nginx/ingress-nginx \
--set controller.service.annotations."service\.beta\.kubernetes\.io/azure-load-balancer-
internal"="true" \
--set-string controller.config.use-forwarded-headers="true" \
--set imagePullSecrets[0].name=docker-hub-reg-pullsecret \
--set-string controller.config.annotation-value-word-blocklist="load_module\,lua_package\,_by_lua
\,location\,root\,proxy_pass\,serviceaccount\,{\,}\,\'\,\\\\" \
--version=4.14.3 \
--set controller.allowSnippetAnnotations=true \
--set controller.config.annotations-risk-level=Critical
```

6. ***[Only for OpenShift]*** If you do not see the ingress pod, ensure that you have created the SCC object correctly by running the following command to inspect the Kubernetes events:

```
$ kubectl get events -n ingress

ingress 0s Warning FailedCreate job/ingress-nginx-admission-create Error creating: pods "ingress-
nginx-admission-create-" is forbidden: unable to validate against any security context constraint:
 [provider "anyuid": Forbidden: not usable by user or serviceaccount, provider nginx-ingress-
scc: .spec.securityContext.fsGroup: Invalid value: []int64{2000}: 2000 is not an allowed group,
 spec.containers[0].securityContext.runAsUser: Invalid value: 2000: must be: 101, provider
 restricted: .spec.securityContext.fsGroup: Invalid value: []int64{2000}: 2000 is not an allowed group,
 spec.containers[0].securityContext.runAsUser: Invalid value: 2000: must be in the ranges: [1000650000,
 1000659999], provider "nonroot": Forbidden: not usable by user or serviceaccount, provider "hostmount-
anyuid": Forbidden: not usable by user or serviceaccount, provider "machine-api-termination-handler":
 Forbidden: not usable by user or serviceaccount, provider "hostnetwork": Forbidden: not usable by user or
 serviceaccount, provider "hostaccess": Forbidden: not usable by user or serviceaccount, provider "node-
exporter": Forbidden: not usable by user or serviceaccount, provider "privileged": Forbidden: not usable
 by user or serviceaccount]
```

7. To validate the deployment of Ingress Controller, run this command for both cloud provider and on-premise:

```
kubectl get nodes
kubectl get pods -n ingress
```

Verify that all the pods are in Running State and *Ready (1/1)*. It may take a little while for the pods to get to the ready state.

Retrieving the Ingress IP Address

1. To get the Ingress IP address, run the following command:

```
kubectl get svc -n ingress ingress-nginx-controller
```

Where, ingress-*nginx-controller* is the name of the ingress controller Kubernetes service. The ingress IP address is listed under the "EXTERNAL_IP" column. When running this command on a cloud provider cluster, the "EXTERNAL_IP" may take a few minutes to be displayed since the service load balancer must be provisioned.

Setting "topologySpreadConstraints" for Ingress deployment

To improve ingress deployment reliability when the "controller.kind" is set to "Deployment" (default), include the following topology constraint. This ensures that the ingress pods are spread across different cluster nodes, and when deployed on a multi-zone cluster, the pods will also be spread across different zones. This parameter contains multiple lines and therefore is easier to specify in an override values yaml file. For example:

```
controller:
topologySpreadConstraints:
```

```
- maxSkew: 1
  topologyKey: topology.kubernetes.io/zone
  whenUnsatisfiable: ScheduleAnyway
  labelSelector:
    matchLabels:
      app.kubernetes.io/name: '{{ include "ingress-nginx.name" . }}'
      app.kubernetes.io/instance: '{{ .Release.Name }}'
      app.kubernetes.io/component: controller
- maxSkew: 1
  topologyKey: kubernetes.io/hostname
  whenUnsatisfiable: ScheduleAnyway
  labelSelector:
    matchLabels:
      app.kubernetes.io/name: '{{ include "ingress-nginx.name" . }}'
      app.kubernetes.io/instance: '{{ .Release.Name }}'
      app.kubernetes.io/component: controller
```

Next Step

Deploying Enclave Services

Deploying NGINX Ingress in Symantec Identity Security Platform Namespace

Learn how to deploy Ingress in Symantec Identity Security Platform (IDSP)Namespace.

The standard option to deploy IDSP uses NGINX Ingress Controller deployed as a shared service in a global Kubernetes namespace. In a cluster where the operator has limited permissions, it may be required to deploy NGINX Ingress Controller as a private service in the IDSP Kubernetes namespace. Whether you want to deploy NGINX Ingress globally or scoped to a namespace, deploying on OpenShift requires use of custom SCC that may not always be welcome by organization's security/governance policy.

> **NOTE**
> The minimum required Kubernetes cluster privilege to deploy Ingress in the IDSP namespace is **Cluster Admin in the namespace**. This can be done by creating a RoleBinding to the cluster-admin ClusterRole in the namespace.

Follow these steps to deploy Nginx Ingress in the IDSP namespace on ***OpenShift***:

1. Creating the SCC (Security Context Constraints)
2. Deploying NGINX Ingress in IDSP namespace
3. Create the OpenShift Routes for IDSP's Business and Management Ingress hosts
4. Update the DNS to point IDSP Business

Creating the Security Context Constraints (SCC)

Use the following command to create a Security Context Constraints (SCC). Ensure that you update the "NAMESPACE" environment variable with the name of the **IDSP Kubernetes Namespace** before running it:

```
export NAMESPACE="<ssp_namespace>"
```

```
cat <<EOF | kubectl apply -f -
apiVersion: security.openshift.io/v1
kind: SecurityContextConstraints
```

```
metadata:
  name: nginx-ingress-scc

priority: null
users:
- system:serviceaccount:${NAMESPACE}:ingress-nginx
groups: []

allowHostDirVolumePlugin: false
allowHostIPC: false
allowHostNetwork: false
allowHostPID: false
allowHostPorts: false
allowPrivilegeEscalation: false
allowPrivilegedContainer: false
allowedCapabilities:
- NET_BIND_SERVICE
defaultAddCapabilities: null
readOnlyRootFilesystem: false
requiredDropCapabilities:
- ALL
runAsUser:
  type: MustRunAs
  uid: 101
fsGroup:
  type: MustRunAs
seLinuxContext:
  type: MustRunAs
supplementalGroups:
  type: MustRunAs
seccompProfiles:
- '*'
volumes:
- secret
EOF
```

Deploying NGINX Ingress in IDSP namespace

NOTE

- Minimum admin privileges require cluster-admin role scoped to the namespace where IDSP is going to be deployed.
- Do not apply the Role and RoleBining yaml manually.

Use the following commands to deploy NGINX Ingress in IDSP Namespace. Before proceeding with the Ingress deployment, ensure that you update the following environment variables:

- SSP_FQDN - IDSP FQDN
- NAMESPACE - IDSP Kubernetes Namespace
- RELEASENAME - IDSP Helm Releasename
- INGRESS_SERVICE_TYPE - It is recommended to use a service-type as LoadBalancer. If you do not have the permission to expose the ingress service as a load balancer, use cluster IP (This option requires you to create an

OpenShift Route object that will route traffic to this service, see Create the OpenShift Routes for IDSP's Business and Management Ingress hosts.

```
export SSP_FQDN="<ssp-FQDN>"
export NAMESPACE="<ssp-namespace>"
export INGRESS_SERVICE_TYPE="<LoadBalancer or ClusterIP>"

helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update
helm install ingress-nginx -n ${NAMESPACE} ingress-nginx/ingress-nginx \
  --version=4.14.3 \
  --set imagePullSecrets[0].name=docker-hub-reg-pullsecret \
  --set-string controller.config.annotation-value-word-blocklist="load_module\,lua_package\,_by_lua\,location
\,root\,proxy_pass\,serviceaccount\,{\,}\,\'\,\\\\" \
  --set controller.service.type=${INGRESS_SERVICE_TYPE} \
  --set controller.scope.enabled=true \
  --set controller.image.runAsUser=101 \
  --set controller.image.allowPrivilegeEscalation=false \
  --set controller.admissionWebhooks.enabled=false \
  --set controller.ingressClassResource.enabled=false \
  --set controller.watchIngressWithoutClass=true \
  --set controller.allowSnippetAnnotations=true \
  --set controller.service.externalTrafficPolicy="Local"
```

Create the OpenShift Routes for IDSP's Business and Management Ingress Hosts

If you chose INGRESS_SERVICE_TYPE="LoadBalancer" in the previous Helm command, you can ***skip*** this step. However, if you chose INGRESS_SERVICE_TYPE="ClusterIP", the Nginx Ingress service would not be exposed as a load balancer.

Perform the following step to expose the ingress service through an OpenShift route.

NOTE

- The environment variables used in the following command are defined in the previous step.
- The following command creates two route objects, one for tenant operations and the other for the system management operations. You may want to expose the system management route only to the intern network.
- For more information on exposing apps with OpenShift routes, see here.

- Use the following commands to create the IDSP's business and management routes:

```
cat <<EOF | kubectl apply -f -
kind: Route
apiVersion: route.openshift.io/v1
metadata:
  name: ${RELEASENAME}-ssp-route
  namespace: ${NAMESPACE}
  labels:
    app.kubernetes.io/component: route
    app.kubernetes.io/instance: ${RELEASENAME}
spec:
  host: ${SSP_FQDN}
  to:
    kind: Service
    name: ingress-nginx-controller
    weight: 100
```

```
  port:
    targetPort: https
  tls:
    termination: passthrough
    insecureEdgeTerminationPolicy: None
  wildcardPolicy: None
EOF


cat <<EOF | kubectl apply -f -
kind: Route
apiVersion: route.openshift.io/v1
metadata:
  name: ${RELEASENAME}-ssp-route-mgmt
  namespace: ${NAMESPACE}
  labels:
    app.kubernetes.io/component: route
    app.kubernetes.io/instance: ${RELEASENAME}
spec:
  host: mgmt-${SSP_FQDN}
  to:
    kind: Service
    name: ingress-nginx-controller
    weight: 100
  port:
    targetPort: https
  tls:
    termination: passthrough
    insecureEdgeTerminationPolicy: None
  wildcardPolicy: None
EOF
```

After creating the routes, confirm the routes are associated with the newly deployed NGINX Ingress Controller service in the IDSP namespace.

```
bash-4.4 ~ $ k get routes -n ssp1
NAME                      HOST/PORT                                                PATH   SERVICES                   PORT    TERMINATION          WILDCARD
sspvipdbos-ssp-route      nightly-os-ssp.clusters.demo.sspdev.dev.broadcom.com            ingress-nginx-controller   https   passthrough/None     None
sspvipdbos-ssp-route-mgmt mgmt-nightly-os-ssp.clusters.demo.sspdev.dev.broadcom.com       ingress-nginx-controller   https   passthrough/None     None
bash-4.4 ~ $
```

Updating the DNS to point to IDSP Service Business

Perform this step to update the DNS to point to the IDSP Business and the Management Host to OS Ingress Controller's external IP Address. The IP Address belongs to the "router-default" service in the "openshift-ingress" namespace that can be viewed through the OS Console UI or Kubernetes CLI.

```
bash-4.4 ~ $ kubectl get svc -n openshift-ingress
NAME                      TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                      AGE
router-default            LoadBalancer                                   80:31395/TCP,443:32088/TCP   95d
router-internal-default   ClusterIP                      <none>          80/TCP,443/TCP,1936/TCP      95d
bash-4.4 ~ $
```

Reference Links to:

- Preparing the OpenShift Cluster
- OpenShift Deployment commands used during IDSP Deployment

Ingress Controller on OpenShift

Learn about various options to deploy Ingress in the OpenShift environment.

This topic includes the following sections:

- Multi-Tenant URL Patterns and Operations
- Deployment Configuration
- Advanced Configuration
- Multi-Tenancy Management

Multi-Tenant URL Patterns and Operations

IDSP is a multi-tenant service, even if the deployment is planning to use a single tenant. When using OpenShift Routes, URLs for the business ingress follow different patterns covering construction of the host FQDN such as SSP_FQDN = "authentication.corporate.acme.com".

NOTE

- For the management ingress, the URL always starts with "https://system.<SSP_FQDN>/system/...". The SSL certificate must be issued for the exact host FQDN, or a wildcard certificate must be issued for the SSP_FQDN host.
- "<tenant-placeholder-in-url-path>" in the API path is just a placeholder defaulting to "common". It does not need to correspond to the actual tenant name associated with the vanity host FQDN. This placeholder can be any URL-friendly string passed into the chart deployment.

URL Pattern Options

Routing Strategy	When to use this strategy	Business Ingress URL	Notes	Postman Configurations
Subdomain under SSP-FQDN host	When using tenant "default".	https://<tenant-name>**.<SSP_FQDN>**/<tenant-placeholder-in-url-path>/...	This is the default, out-of-the-box ingress configuration generated by the standard helm chart when deployed in the OpenShift Routes mode. SSL certificates must be issued for the exact host FQDNs, or a wildcard certificate must be issued for the SSP_FQDN host. Additional tenants created via /Tenants API will not be routable as no ingress/route definitions can be created supporting this strategy.	• pm.environment.set("apiPathTen "common"); • pm.environment.set("sspHost", "default.authentication.corporate • pm.environment.set("sspMgmtH "system.authentication.corporate
End User Vanity host	When using a tenant exposed via custom vanity hosts. Any tenant can be exposed using this strategy.	https://**login.acme.com**/<tenant-placeholder-in-url-path>/...	End User Vanity host created as per documentation enables use of custom vanity host domains. The tenant identified by the vanity host FQDN must exist. SSL certificate must be issued for the vanity host FQDN. Additional tenants to be exposed as vanity hosts are created by /Tenants system management API.	• pm.environment.set("apiPathTen "iam"); • pm.environment.set("sspHost", "login.acme.com"); • pm.environment.set("sspMgmtH "system.authentication.corporate

Routing Strategy	When to use this strategy	Business Ingress URL	Notes	Postman Configurations
X-TENANT-NAME header asserted on the API request exposed on SSP-FQDN host	When multi-tenancy is required. Any tenant can be exposed using this strategy.	https://**<SSP_FQDN>**/<tenant-placeholder-in-url-path>/... header: X-TENANT-NAME = <tenant-name>	APIs exposed by domain <SSP-FQDN> indicate resource tenancy using X-TENANT-NAME header. If the header is not provided, API request will fail with 400 BAD REQUEST status. The tenant identified by X-TENANT-NAME must exist. SSL certificate must be issued for the SSP_FQDN host. Additional tenants are created by /Tenants system management API. Clients using those tenants are responsible for sending X-TENANT-NAME header.	• pm.environment.set("apiPathTen "common"); • pm.environment.set("sspHost", "authentication.corporate.acme. • pm.environment.set("sspMgmtH "system.authentication.corporate

Deployment Configuration

Specifying "Ingress Type" and "Ingress ClassName"

- If you intend to use **OpenShift Routes** when deploying IDSP, pass these additional parameters in the IDSP helm deployment command.
- **Note that OpenShift Route requires OpenShift version 4.18 and above.**

Basic Deployment Parameters

For a basic OpenShift Routes deployment, you need these parameters:

```
--set ssp.ingress.type="openshiftroute" \
--set ssp.ingress.ingressClassName="openshift-default" \
--set ssp.ingress.host="${SSP_FQDN}" \
--set ssp.ingress.mgmt.host="${SSP_MGMT_FQDN}"
```

Optional Deployment Parameters

To deploy IDSP chart in "OpenShift Routes" mode, the following optional helm parameters can be provided during helm install/upgrade:

```
--set ssp.ingress.mgmt.ingressClassName="openshift-default"  # (optional, if not provided defaults to
 ssp.ingress.ingressClassName)
--set ssp.ingress.tls.host="${SSP_FQDN}"  # (optional, if not provided defaults to ssp.ingress.host)
--set ssp.ingress.mgmt.tls.host="${SSP_MGMT_FQDN}"  # (optional, if not provided defaults to
 ssp.ingress.mgmt.host)
--set ssp.ingress.tls.secretName="default-tls-secret-name"  # (optional, if not provided defaults to ssp-
general-tls)
--set ssp.ingress.mgmt.tls.secretName="system-tls-secret-name"  # (optional in case wildcard certificate is
 used for the SSP_FQDN host, if not provided defaults to ssp.ingress.tls.secretName)
--set ssp.ingress.openShiftRoute.defaultTenant="<default-tenant>"  # (optional, defaults to "default")
```

```
--set ssp.ingress.openShiftRoute.defaultTenantPlaceholderInPath="<tenant-placeholder-in-url-path>"  #
 (optional, defaults to "common")
```

Advanced Configuration

Supporting IDSP Tenant Name in the URL Path

From IDSP v3.2.1 onwards, to provision OpenShift Routes to use the IDSP FQDN host with a default tenant name included in the API path, the IDSP deployment process requires specifying the parameter:

`--set ssp.ingress.openShiftRoute.isDefaultTenantInPath=true`

This configuration lets OpenShift Routes to be provisioned as:

`https://<SSP-FQDN>/<tenant-name>/`

The specified default tenant can be routed to using a tenant in **host or in the URL path**. All other tenants are accessible using the vanity URLs only.

Configuration Options

- If "`ssp.ingress.openShiftRoute.isDefaultTenantInPath=false` " - routing for the default tenant uses the format: `https://<defaultTenant>.${SSP_FQDN}/common`
  Example: "`https://default.${SSP_FQDN}/common/... `"
- If "`ssp.ingress.openShiftRoute.isDefaultTenantInPath=true` " - routing for the default tenant uses the format: "`https://${SSP_FQDN}/<defaultTenant>`"
  Example: "`https://${SSP_FQDN}/default/...`"

To know the value of the ${SSP_FQDN} environment variable, see the Configuring IDSP Environment Variables documentation.

Multi-Tenancy Management

The "/Tenants" system management API creates a new tenant or updates an existing tenant. After creating a tenant, follow steps covered in Expose End User Operations on a Vanity Ingress section to create ingress definitions for the custom vanity host.

Configuring Gateway API for Ingress

Overview of transitioning to and configuring the Kubernetes Gateway API for ingress in IDSP.

The Symantec Identity Security Platform (IDSP) is transitioning from the legacy NGINX Ingress controller to the Kubernetes Gateway API standard. The Gateway API provides a unified, extensible framework for managing ingress traffic, enabling advanced routing, improved security integrations like BackendTLSPolicy, and better native support across multi-cloud environments.

Before deploying IDSP, you must configure a Gateway API controller for your cluster. Choose the deployment based on the platform that corresponds to your cloud provider or on-premises environment to prepare your infrastructure, configure your GatewayClass, establish secure traffic policies, and ensure seamless connectivity to the platform.

For more information about the Gateway API configuration in IDSP, see the following topics:

- Ingress vs Gateway API: Traffic Flow Comparison
- Gateway API Controller Capabilities
- Gateway API Controller Selection and Prerequisites
- Gateway API Setup and Ingress Configuration Reference
- Already Have a Gateway? Use It!
- Supported Gateways in IDSP

Ingress vs Gateway API: Traffic Flow Comparison

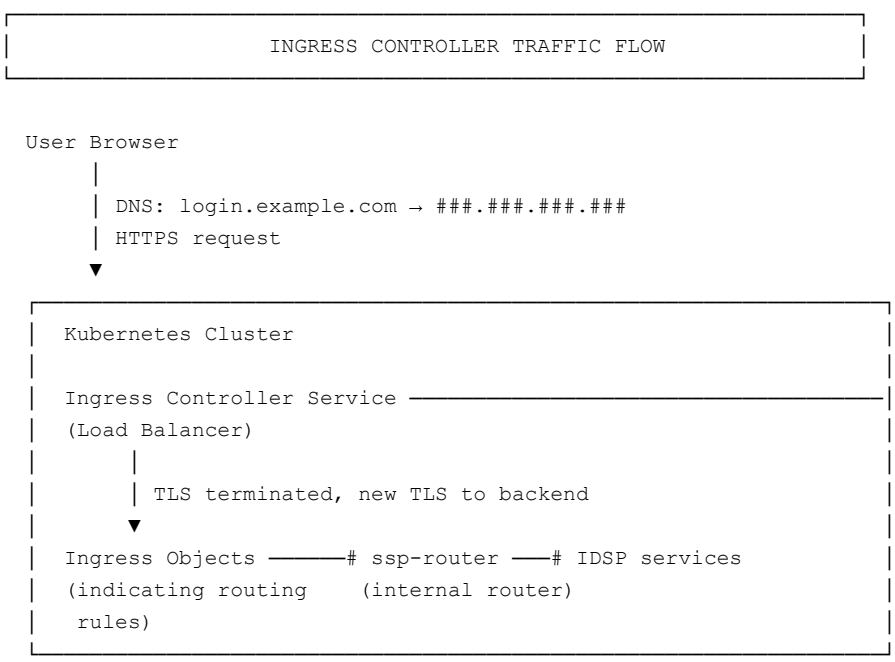
This topic compares the traditional Ingress-based traffic routing model with the newer Gateway API approach in Kubernetes, illustrating how each method directs external HTTPS traffic to IDSP services.

The following diagrams illustrate how traffic routing evolves from the traditional Ingress model to the newer Gateway API approach in Kubernetes. Understanding both models helps teams plan and execute a smooth migration path.

Ingress Traffic Flow

With the Ingress-based model, an Ingress Controller Service acts as the load balancer and Ingress Objects define the routing rules that direct traffic to IDSP services:

This diagram assumes that you have migrated from legacy routes to ssp-router. See IDSP upgrade section (Step 12)

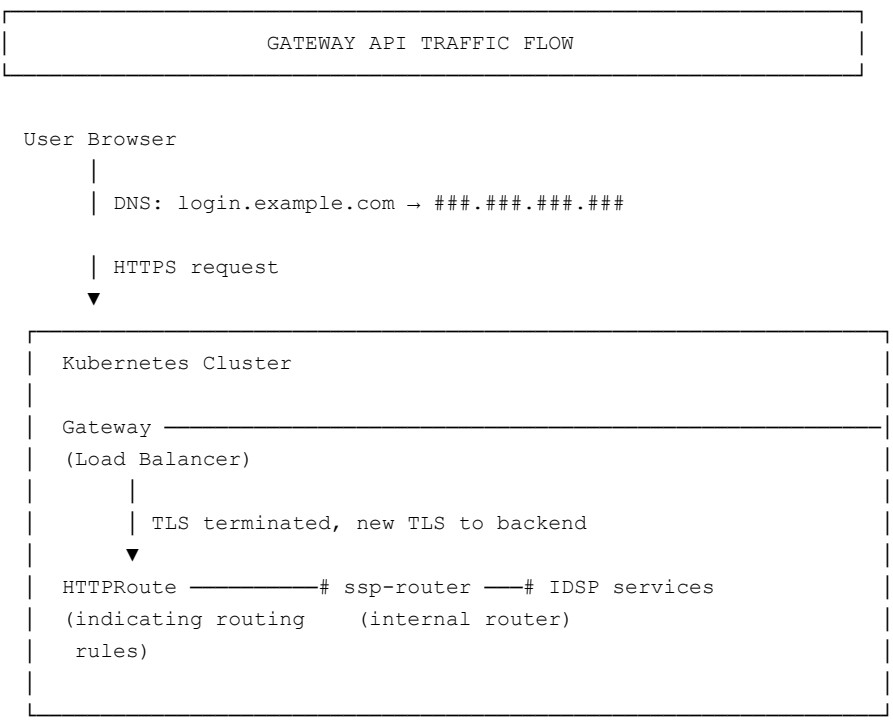


In this model, the key components are:

- **Ingress Controller Service**
  Acts as the external-facing load balancer that receives incoming HTTPS traffic from user browsers. The controller terminates TLS and establishes new TLS connections to backend services within the cluster.
- **Ingress Objects**
  Kubernetes Ingress resources that define routing rules such as hostname matching and path-based routing. These rules determine how traffic is forwarded from the Ingress Controller to the internal ssp-router service.
- **SSP Router**
  IDSP's internal router service that receives traffic from the Ingress Controller and performs tenant extraction, path normalization, and microservice routing to direct requests to the appropriate IDSP services.

Gateway API Traffic Flow

Gateway API is the Kubernetes-native replacement for Ingress that routes external HTTPS traffic to IDSP. Here, a Gateway resource serves as the load balancer and HTTPRoute defines the routing rules:



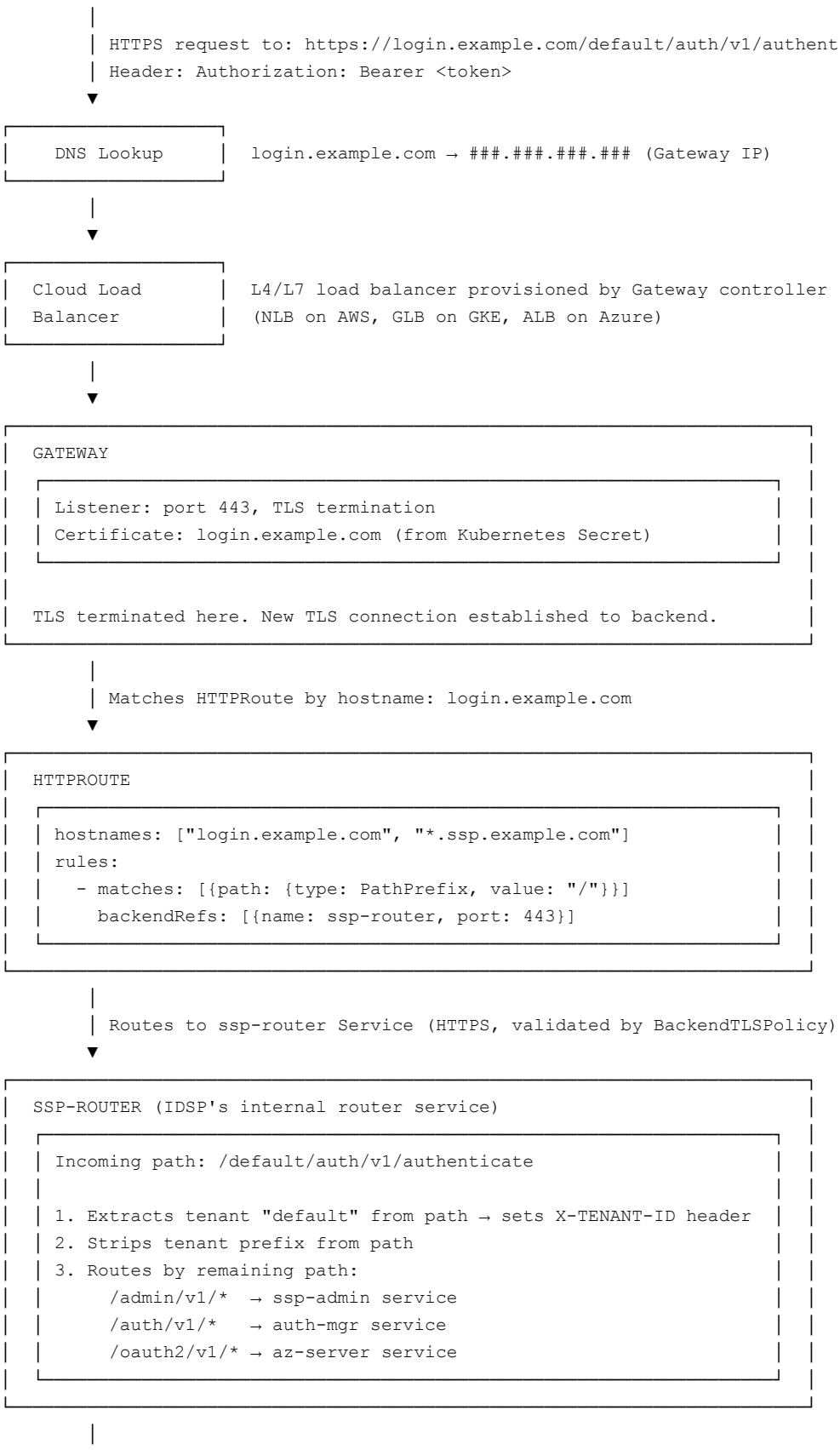
In the Gateway API model, the key components are:

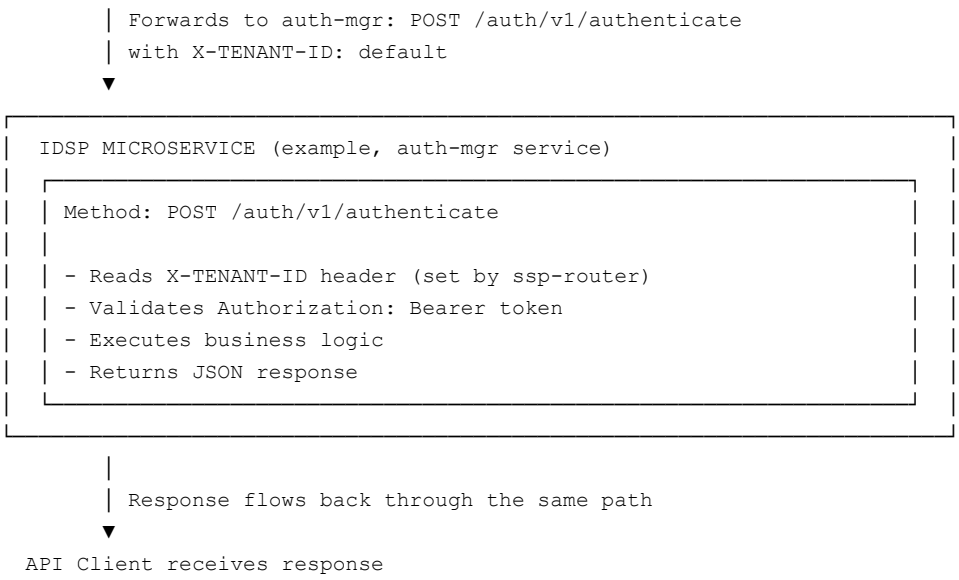
- **Gateway**
  A Gateway API resource that provisions and configures a cloud load balancer. The Gateway listens on ports 80 and 443, holds TLS certificates stored in Kubernetes Secrets, and receives an external IP address for DNS resolution. It terminates TLS and establishes new encrypted connections to backend services.
- **HTTPRoute**
  A Gateway API resource that defines application-layer routing rules. HTTPRoute specifies hostname matching patterns (e.g., `ssp.example.com` , `*.ssp.example.com`), path prefix matching (e.g., `/admin/v1/*` , `/auth/v1/*`), and backend service references pointing to `ssp-router:443` .
- **SSP Router**
  IDSP's internal router service that functions identically in both models — performing tenant extraction, path normalization, and microservice routing to direct requests to the appropriate IDSP services.

  NOTE
  Both models route traffic to the same internal ssp-router service. The migration from Ingress to Gateway API changes only the external traffic entry point and routing configuration — IDSP's internal routing logic remains unchanged.

End-to-End Request Flow in IDSP

This diagram shows how a request flows from an end user through Gateway API to an IDSP microservice:


HTTPS request to: https://login.example.com/default/auth/v1/authenticate
Header: Authorization: Bearer <token>
DNS Lookup
login.example.com → ###.###.###.### (Gateway IP)
Cloud Load Balancer
L4/L7 load balancer provisioned by Gateway controller
(NLB on AWS, GLB on GKE, ALB on Azure)
GATEWAY
Listener: port 443, TLS termination
Certificate: login.example.com (from Kubernetes Secret)
TLS terminated here. New TLS connection established to backend.
Matches HTTPRoute by hostname: login.example.com
HTTPROUTE
hostnames: ["login.example.com", "*.ssp.example.com"]
rules:
- matches: [{path: {type: PathPrefix, value: "/"}}]
backendRefs: [{name: ssp-router, port: 443}]
Routes to ssp-router Service (HTTPS, validated by BackendTLSPolicy)
SSP-ROUTER (IDSP's internal router service)
Incoming path: /default/auth/v1/authenticate
1. Extracts tenant "default" from path → sets X-TENANT-ID header
2. Strips tenant prefix from path
3. Routes by remaining path:
/admin/v1/* → ssp-admin service
/auth/v1/* → auth-mgr service
/oauth2/v1/* → az-server service



Flow Notes:

- **Single Entry Point**
  All external traffic enters through the Gateway and routes to `ssp-router` .
- **SSP Router as Internal Router**
  IDSP uses its own router (not the Gateway) for microservice routing because:
  – Path-based routing requires IDSP-specific logic (tenant extraction, token validation)
  – Microservices are internal-only and not exposed directly
- **Backend HTTPS**
  Gateway connects to ssp-router over HTTPS (port 443), validated via BackendTLSPolicy.
- **Vanity Domains**
  End-user login portals can use custom domains (for example., `login.example.com`) that route to the same ssp-router.
- **ISK Encryption**
  All intra-cluster communications are encrypted using the Internal Signing Key (ISK), including between the Gateway and IDSP Router (Gateway Backend TLS).

Gateway API Controller Capabilities

This topic compares Gateway API controller capabilities across supported providers, covering routing, TLS validation, mTLS certificate forwarding, and traffic policy controls.

Gateway API Controller Capabilities Comparison

The following table compares all supported ingress controllers and gateway providers across the capabilities that matter most for IDSP deployments. Use it to select the right controller for your environment.

Controller	ingress.type	Platforms	Backend TLS Validation	mTLS Cert Forw
Envoy Gateway	`gatewayapi`	GKE, EKS, AKS, VKS	# Full CA (BackendTLSPolicy)	# XFCC header
GKE L7 Gateway	`gatewayapi`	GKE	HTTPS only, no validation	# Metadata only (fing
Azure AGC	`gatewayapi`	AKS	# Full CA (vendor BackendTLSPolicy)	# Validates but does forward
AWS ALB	`gatewayapi`	EKS	HTTPS only, no validation	# No optional verify m
Avi AKO	`gatewayapi`	VKS, vSphere	# Full CA (RouteBackendExtension)	## Via Avi API (manu may not persist acros reconciliation)
OpenShift HAProxy	`openshiftroute`	OpenShift	HTTPS only, no validation	# X-SSL-Client-DER

NOTE

- **Envoy Gateway:** If the optimal Gateway selection is undetermined, use Envoy Gateway. It operates across all platforms and provides the most complete feature set, including configurable headers, backend TLS validation, and mTLS forwarding.
- **OpenShift:** OpenShift uses its native HAProxy Router with OpenShift Routes (`ingress.type: openshiftroute`) instead of the Gateway API. It supports mTLS via `IngressController.spec.clientTLS` . The default header limit is 24 KB; increase this limit to at least 128 KB for IDSP (see the OpenShift HAProxy Router section below).
- **AWS ALB header limits:** The limits of 16 KB per header and 64 KB total are strict AWS infrastructure constraints that cannot be overridden. OIDC `id_token` values with numerous claims can exceed the 16 KB limit. If this limitation affects the deployment, use Envoy Gateway on EKS, which operates behind a Network Load Balancer (NLB) without header limits.
- **Avi AKO:** Avi AKO is the default choice for VMware Kubernetes Service (VKS). Header size limits and mTLS certificate forwarding are configurable through the Avi Controller REST API (using a custom Application Profile for headers and an HTTP Policy Set for certificate forwarding). However, these require manual modifications to the Avi Virtual Service (VS), and it is undetermined whether AKO reconciliation might revert them. For VKS deployments requiring mTLS, use Envoy Gateway on VKS instead.

Frontend mTLS (Client Certificate Authentication)

IDSP supports OAuth2 `tls_client_auth` — clients authenticate to the `/token` endpoint using a TLS client certificate. This requires two things from the Gateway:

1. **Request/validate the client certificate** during the TLS handshake
2. **Forward the certificate to the backend** via an HTTP header so the authorization server can read it

The backend services expect the client certificate in the `X-CLIENT-CERT` header as URL-encoded PEM. With Gateway API, ssp-router sits between the Gateway and the backend services, so ssp-router must normalize the provider-specific cert header into `X-CLIENT-CERT` before forwarding to backends.

IMPORTANT

The Gateway API standard defines frontendValidation (GEP-91) for client certificate validation at the Gateway level (`Gateway.spec.tls.frontendValidation` , `gateway.networking.k8s.io/v1`). However, this only covers **validation** — it does **not** define how to forward the certificate to backends. Forwarding is entirely implementation-specific. Most implementations have not adopted `frontendValidation` yet and use their own CRDs instead.

For a summary of which controllers support `tls_client_auth` , see the Choose Your Controller table above. The following table provides additional detail on header names, formats, and Helm automation — including OpenShift HAProxy and nginx Ingress which are not Gateway API controllers but also support mTLS.

Controller	Forwards client cert to backend	Validation CRD	Cert header na
Envoy Gateway	# Yes	`ClientTrafficPolicy`	`x-forwarded-client-ce`
AWS LB Controller	# No	N/A	N/A — verify mode has no optic passthrough doesn't validate ch
nginx Ingress	# Yes	`auth-tls-*` annotations	`X-CLIENT-CERT`
OpenShift HAProxy	# Yes	`IngressController.spec.clientTLS`	`X-SSL-Client-DER`
Avi AKO	# No (EVH limitation)	Application Profile + PKI Profile (Avi Controller)	`X-Whole-Cert` (via HTTP P
GKE L7 Gateway	# No	N/A	N/A — only forwards fingerprint
Azure AGC	# No	`FrontendTLSPolicy`	N/A — validates but does not fo

NOTE

- **Why the full certificate PEM is required:** The gateway must forward the full certificate PEM (not just a fingerprint). IDSP needs the full certificate to extract Subject DN and SANs for `tls_client_auth` authentication, and to compute certificate-bound access token claims (`x5t#S256` , `kid`). Controllers that only forward metadata (like GKE's SHA-256 fingerprint) cannot support `tls_client_auth` .
- **Avi AKO mTLS:** Avi Load Balancer can forward client certificates using an HTTP Policy Set, but this requires manual configuration on the Avi Controller. It is not yet Kubernetes-managed through CRDs. For VKS deployments requiring mTLS, Envoy Gateway on VKS is recommended instead.
- **Gateway type naming change:** The `gatewayType` values were renamed to be controller-specific rather than platform-specific. The old names (`eks` , `aks` , `gke`) are still accepted by the Helm chart for backward compatibility. They are silently normalized to the new names (`aws-alb` , `azure-agc` , `gke-l7`).

Helm Configuration

```
ssp:
  ingress:
    mtls:
      # Enable frontend mTLS (client certificate authentication / tls_client_auth).
      # This is the single master gate for mTLS. When enabled:
      #   - Sets SSP_MTLS_ENABLED=true on ssp-router (enables cert extraction)
      #   - For Envoy Gateway: creates ClientTrafficPolicy with clientValidation
      #     and XFCC header forwarding (SanitizeSet mode)
      #   - For AWS ALB: mTLS not supported (use Envoy Gateway on EKS)
      #   - For nginx/openshift: configures client cert request and forwarding
      #   - For "avi" and "custom" providers, the gateway/LB admin must:
      #       1. Configure the gateway to request and forward client certificates
      #       2. Ensure the gateway sanitizes cert headers to prevent spoofing
```

```
#
# The gateway provider is derived automatically from ingress.type + gatewayType.
# ssp-router uses SSP_GATEWAY_IDENTIFICATION to determine which header carries
# the client certificate (e.g., XFCC for Envoy, X-Amzn-Mtls-Clientcert for ALB).
enabled: false
caCertSecretName: ""    # Secret containing CA cert for client validation

# Per-request gateway detection via X-SSP-Gateway-Type header.
# Default (false): ssp-router uses the provider derived from ingress.type + gatewayType
# for all requests (explicit mode — secure, recommended).
# Set to true only for multi-provider setups where ALL providers inject and sanitize
# the X-SSP-Gateway-Type header. NOT safe with aws-alb, azure-agc, openshift, or
# customIngress controllers that cannot set the header.
enableGatewayDetectionByHeader: false

# Custom cert header/regex — only used when provider resolves to "custom".
# certHeader: ""    # Header name containing the client certificate
# certRegex: ""     # Regex to extract cert from header value (group 1)
```

Gateway API Controller Selection and Prerequisites

This topic helps you choose the right Gateway API controller and covers the prerequisites, RBAC permissions, and TLS configuration required before deployment.

Choose Your Controller

The following table compares all supported ingress controllers and gateway providers across the capabilities that matter most for IDSP deployments. Use it to select the right controller for your environment.

Controller	ingress.type	Platforms	Backend TLS Validation	mTLS Cert Forw
Envoy Gateway	`gatewayapi`	GKE, EKS, AKS, VKS	# Full CA (BackendTLSPolicy)	# XFCC header
GKE L7 Gateway	`gatewayapi`	GKE	HTTPS only, no validation	# Metadata only (fing
Azure AGC	`gatewayapi`	AKS	# Full CA (vendor BackendTLSPolicy)	# Validates but does forward
AWS ALB	`gatewayapi`	EKS	HTTPS only, no validation	# No optional verify n
Avi AKO	`gatewayapi`	VKS, vSphere	# Full CA (RouteBackendExtension)	## Via Avi API (manu may not persist acros reconciliation)
OpenShift HAProxy	`openshiftroute`	OpenShift	HTTPS only, no validation	# X-SSL-Client-DER
nginx	`nginx`	Any	# Full CA (proxy-ssl-secret annotation)	# X-CLIENT-CERT h

NOTE

- Not sure?Use **Envoy Gateway** - it works everywhere and has the most features.
- AWS Note:AWS Load Balancer Controller does not support BackendTLSPolicy for backend certificate validation. HTTPS to backends still works (encrypted), but the ALB does not verify the backend's TLS certificate. If backend TLS validation is required, use Envoy Gateway on EKS instead.
- Avi AKO: Default choice for **VKS (VMware Kubernetes Service)**. Also use when you already run Avi and want Gateway API on VKS, vSphere/NSX-T. Backend TLS is configured via Avi's `RouteBackendExtension` CRD rather than the standard BackendTLSPolicy CRD.
- For deploying Envoy Gateway on AKS version older than 1.35, refer to this section, AKS - Envoy Gateway.

Controller Name	Description	Controller Version	Doc URL
Envoy Gateway	Cross-cloud portable Gateway API controller. Supports BackendTLSPolicy, mTLS with client cert forwarding, and configurable header/buffer limits via ClientTrafficPolicy and BackendTrafficPolicy CRDs.	1.6.0 (minimum)	https://gateway.envoyproxy.io/
GKE L7 Gateway	Native Google Cloud Gateway API controller using Google Cloud Load Balancer. Does not support BackendTLSPolicy or mTLS client cert forwarding.	GKE-managed	https://cloud.google.com/kubernetes-engine/docs/concepts/gatewayapi
Azure AGC	Native Azure Gateway API controller using Application Gateway for Containers. Uses Azure-specific BackendTLSPolicy and HealthCheckPolicy CRDs (alb.networking.azure.io/v1). Does not support mTLS.	1.9.13	https://learn.microsoft.com/en-us/azure/application-gateway/for-containers/overview
AWS LB Controller	Native AWS Gateway API controller using Application Load Balancer. Uses LoadBalancerConfiguration and TargetGroupConfiguration CRDs. Does not support BackendTLSPolicy or mTLS.	3.x	https://kubernetes-sigs.github.io/aws-load-balancer-controller/
Avi AKO	Gateway API controller for VMware VKS and vSphere/NSX-T environments. Uses RouteBackendExtension CRD for backend TLS. Default choice for VKS (VMware Kubernetes Service).	1.12+	https://techdocs.broadcom.com/us/en/vmware-security-load-balancing/avi-load-balancer.html
OpenShift HAProxy	OpenShift's built-in HAProxy Router using OpenShift Routes (not Gateway API). Supports mTLS via IngressController.spec.clientTLS. Header buffer sizes are adjustable at the IngressController level.	4.18	https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html/networking/configuring-routes
nginx Ingress	Kubernetes Ingress controller using nginx. Supports mTLS via auth-tls-* annotations with client cert forwarding in X-CLIENT-CERT header. Used as the legacy/default ingress controller for IDSP.	4.14.3	https://kubernetes.github.io/ingress-nginx/

Prerequisites (All Deployments)

Before deploying IDSP with Gateway API, ensure you have:

Requirement	How to Check	Notes
Gateway API CRDs	`kubectl get crd gateways.gateway.networking.k8s.io`	Install
Gateway Controller	`kubectl get gatewayclass`	Envoy
GatewayClass	`kubectl get gatewayclass`	Must s
TLS Certificate	`kubectl get secret <tls-secret> -n ssp`	For yo
DNS (planned)	-	Know
RBAC Permissions	Information provided in the **Required RBAC Permissions** table	Requi

Install Gateway API CRDs

If Gateway API CRDs are not installed:

```
kubectl apply -f https://github.com/kubernetes-sigs/gateway-api/releases/download/v1.2.0/standard-install.yaml
```

Required RBAC Permissions

The user or service account running `helm install` needs permissions to create Gateway API resources. The **core resources** are required for all controllers; **controller-specific resources** depend on your chosen controller.

Core resources (all controllers):

Resource	API Group	Required When
`Gateway`	`gateway.networking.k8s.io`	`createGatewa`
`HTTPRoute`	`gateway.networking.k8s.io`	Always
`BackendTLSPolicy`	`gateway.networking.k8s.io`	`createBacken`
`Secret`	`core`	Always (for TLS ce
`Service`	`core`	Always

> **NOTE**
> If using an existing Gateway (`createGateway: false`), you only need permissions for `HTTPRoute` and `BackendTLSPolicy`, not `Gateway`.

Each controller's section includes the controller-specific RBAC requirements and ClusterRole YAML. See the section, **Pre-requisite Check** in your chosen controller section.

TLS Certificates

The Gateway requires a TLS certificate for HTTPS termination. Create a Kubernetes Secret with your certificate:

```
# From certificate files
kubectl create secret tls ssp-tls-cert \
  -n ssp \
  --cert=path/to/tls.crt \
  --key=path/to/tls.key

# Verify the secret exists
kubectl get secret ssp-tls-cert -n ssp
```

Certificate requirements:

- Must cover your IDSP hostname(s):
  - Main host (e.g., `idsp.example.com`)
  - Wildcard (e.g., `*.idsp.example.com`) for tenant subdomains
  - Management host if separate (e.g., `mgmt.idsp.example.com`)
- For vanity domains, create separate secrets or use a SAN certificate

Alternative provisioning methods:

- **cert-manager**: Automates certificate issuance from Let's Encrypt or internal CA
- **Cloud provider**: GCP Certificate Manager, AWS ACM, Azure Key Vault
- **Manual**: Upload certificates from your enterprise CA
- **Chart-generated self-signed**: See Self-Signed TLS Certificates below

Self-Signed TLS Certificates

For test and dev environments, the chart can generate self-signed TLS certificates automatically. This removes the need to create certificates manually before deploying.

> **NOTE**
> **Not for production.** Self-signed certificates are not trusted by browsers or clients without explicit CA trust configuration.

How It Works

When `generateSelfSignedCert: true` is set, the chart:

1. Generates a self-signed CA and a TLS certificate signed by that CA
2. Stores the cert, key, and CA in a Kubernetes Secret with a chart-owned name (e.g., `<release>-ssp-tls`)
3. References that Secret in Gateway listeners and/or Ingress TLS sections
4. On upgrades, reuses the existing certificate from the cluster (via Helm `lookup()`) to avoid regeneration

Configuration

Main certificate — covers the primary hostname, wildcard (unless separate), and mgmt (unless separate):

```
ssp:
  ingress:
    tls:
      generateSelfSignedCert: true   # Generates <release>-ssp-tls
      # secretName is ignored when generateSelfSignedCert=true
```

Separate wildcard certificate — use a real cert for the main FQDN but generate a self-signed cert for `*.host` :

```
ssp:
  ingress:
    tls:
      wildcardTls:
        generateSelfSignedCert: true # Generates <release>-ssp-wildcard-tls
```

Separate mgmt certificate — generate a self-signed cert for the management hostname:

```
ssp:
  ingress:
    tls:
      generateSelfSignedCert: true
    mgmt:
```

```
tls:
  generateSelfSignedCert: true # Generates <release>-ssp-mgmt-tls
```

Certificate Names and SANs

The chart uses deterministic naming and automatically adjusts SANs to avoid duplication:

Cert	Generated Secret Name	SANs
Main	`<release>-ssp-tls`	`host` , `*.host` (if no separate wildcard), `mgmt-host`
Wildcard	`<release>-ssp-wildcard-tls`	`*.host`
Mgmt	`<release>-ssp-mgmt-tls`	`mgmt-host`

Resolution Priority

For each cert type, the chart resolves the secret name in this order:

Cert type	generateSelfSignedCert: true	secretName set
Main	`<release>-ssp-tls`	Use `secretName`
Wildcard	`<release>-ssp-wildcard-tls`	Use `wildcardTls.secretName`
Mgmt	`<release>-ssp-mgmt-tls`	Use `mgmt.tls.secretName`
FullVanity	n/a	Per-host `tlsSecretName`
EndUser	n/a	Per-host `tlsSecretName`

NOTE

On AWS, the ALB controller ignores Kubernetes TLS Secrets and uses ACM auto-discovery instead. After deploying with `generateSelfSignedCert: true` , you must import the generated certificate into ACM.

Ingress Configuration Reference

This topic introduces the values.yaml configuration file and provides a complete reference of all ingress parameters for IDSP.

About values.yaml Files

Throughout the documentation, Helm commands reference `-f your-values.yaml` . This file contains your IDSP deployment configuration including:

- Database connection settings
- License and tenant configuration
- Service replicas and resource limits
- Other IDSP-specific settings

The Gateway API settings in this doc are additions to your existing values.yaml, not replacements.

The following table describes the ingress configuration parameters available in the values.yaml file. These parameters control how external traffic is routed to your IDSP services and can be configured based on your specific ingress controller type (nginx, OpenShift Route, or Gateway API).

Ingress Parameters

Parameter	Description
ssp.ingress.enabled	enabled = true | false **Default Value**: true
ssp.ingress.type	• type = "nginx" for nginx ingress controller (IngressAPI) • type = "openshiftroute" for openshift ingress controller (IngressAPI) • type = "gatewayapi" for ingress controllers compatible with Gateway API **Default Value**: nginx
ssp.ingress.ingressClassName	Specifies the ingress class name to be used for the ingress controller. **Default Value**: nginx
ssp.ingress.keepLegacyRoutes	• **Phase 1**: Keep pre-ssp-router legacy routes during migration to ssp-router. • When true, both legacy and ssp-router Ingress routes are created on the same controller. **Default Value**: false
ssp.ingress.previousController.enabled	• **Phase 2**: Previous controller settings (used during migration to a new ingress controller or Gateway API). • When enabled, keeps the current ssp-router Ingress unchanged on the old controller and creates an additional Ingress (named with the new class suffix) on the new controller. • This allows gradual DNS switchover with zero downtime. **Default Value**: false
ssp.ingress.previousController.ingressClassName	• IngressClass of the previous (old) controller (e.g., "nginx"). • Required when enabled=true. **Default Value**: ""
ssp.ingress.additionalLabels	additional labels for ingress objects **Default Value**: {}
ssp.ingress.additionalAnnotations	additional annotations for ingress objects **Default Value**: {}
ssp.ingress.timeoutSecs	• Default timeout in seconds for proxy read/send (used by Gateway API and legacy templates). • For nginx-specific override, use nginx.timeoutSecs. **Default Value**: "60"
ssp.ingress.host	host is an fqdn. For example: `iam.example.com` or `iam.<yourdomain>.com`. **Default Value**: (empty)
ssp.ingress.tls.enabled	Enables or disables TLS configuration for ingress. **Default Value**: true
ssp.ingress.tls.secretName	Name of the TLS secret containing the certificate and private key. **Default Value**: ssp-general-tls
ssp.ingress.tls.host	Host for the TLS certificate. **Default Value**: (empty)

ssp.ingress.tls.generateSelfSignedCert	• Generate a self-signed TLS certificate and store it in the Secret named by secretName. • The cert includes {host} and *.{host} as SANs (unless wildcardTls.secretName is set). • Useful for test/dev environments. Not for production. Has no effect on AWS ALB (uses ACM). **Default Value**: false
ssp.ingress.tls.wildcardTls.secretName	• Separate TLS secret for the wildcard (*.host) Gateway listener. • If empty, the wildcard listener uses the main tls.secretName. • Set this when you have a real cert for the main FQDN but need a self-signed wildcard cert. **Default Value**: ""
ssp.ingress.tls.wildcardTls.generateSelfSignedCert	Generate a self-signed certificate for the wildcard TLS secret. **Default Value**: false
ssp.ingress.mtls.enabled	Enables or disables mutual TLS (mTLS) authentication. **Default Value**: false
ssp.ingress.mtls.caCertSecretName	Name of the secret containing the CA certificate for mTLS authentication. **Default Value**: (empty)
ssp.ingress.gatewayApi.gatewayType	• Gateway controller type: auto, envoy, gke, aks, eks, avi, custom • "auto" detects from gatewayClassName prefix (gke-l7-* → gke, envoy-* → envoy, etc.) **Default Value**: auto
ssp.ingress.gatewayApi.createGateway	If true, Helm creates the Gateway resource **Default Value**: true
ssp.ingress.gatewayApi.existingGateway	Name of pre-existing Gateway (required if createGateway=false) **Default Value**: ""
ssp.ingress.gatewayApi.existingGatewayNamespace	Namespace of pre-existing Gateway (defaults to Release namespace if empty) **Default Value**: ""
ssp.ingress.gatewayApi.gatewayClassName	GatewayClass name (required if createGateway=true) **Default Value**: ""
ssp.ingress.gatewayApi.infrastructureRef	Infrastructure parametersRef for cloud-specific load balancer config (e.g., AWS LoadBalancerConfiguration) **Default Value**: {}
ssp.ingress.gatewayApi.additionalListeners	• Additional listeners to add to the main Gateway (for sharing with other apps). • allowedRoutesFrom defaults to "All" for cross-namespace sharing; a ReferenceGrant in THIS namespace is still required to authorize the cross-namespace reference. **Default Value**: []
ssp.ingress.gatewayApi.httpToHttpsRedirect	• Add an HTTP listener (port 80) and redirect HTTPRoute to automatically redirect HTTP → HTTPS (301). Equivalent to nginx ssl-redirect: "true". • Values: "auto" (default), true, false. • "auto" enables for envoy/custom, disables for cloud controllers (gke/aks/eks/avi) that handle redirect at the infrastructure level. **Default Value**: "auto"

ssp.ingress.gatewayApi.skipWildcardListener	Skip the wildcard (*.host) listener on the Gateway. Set true when the TLS certificate does not cover the sub-wildcard (e.g., cert is *.example.com but host is app.example.com, so *.app.example.com is not covered). Tenant routing still works via ssp-router path-based routing. **Default Value**: false
ssp.ingress.gatewayApi.createBackendTLSPolicy	Create BackendTLSPolicy resource for ssp-router (set false if managing separately) **Default Value**: true
ssp.ingress.gatewayApi.additionalLabels	Labels for Gateway and HTTPRoute resources **Default Value**: {}
ssp.ingress.gatewayApi.envoy.connectionBufferLimitBytes	• Per-connection buffer limit in bytes for Envoy Gateway ClientTrafficPolicy. • Equivalent to nginx proxy-buffer-size × proxy-buffers-number (128k × 4 = 512KB). **Default Value**: 524288
ssp.ingress.gatewayApi.envoy.maxRequestBodySize	• Maximum request body size. Requests exceeding this limit receive HTTP 413. • Equivalent to nginx proxy-body-size: 50m. Set to "" to disable the limit. • Uses Kubernetes resource.Quantity format (e.g., "50Mi", "10Mi"). **Default Value**: "50Mi"
ssp.ingress.gatewayApi.avi.sslProfile	• SSL profile on Avi Controller for backend re-encrypt TLS. • Configures the cipher suites and TLS version for the Avi pool → ssp-router connection. **Default Value**: {}
ssp.ingress.gatewayApi.gatewayAnnotations	Annotations for Gateway resource (e.g., static IP, load balancer settings) **Default Value**: {}
ssp.ingress.gatewayApi.addresses	• Static addresses for the Gateway (standard Gateway API spec.addresses). • Supported by: Envoy Gateway, Istio, Contour, Cilium. • NOT supported by cloud-managed controllers (GKE, AKS AGC, EKS ALB) — use gatewayAnnotations for those instead (e.g., networking.gke.io/static-ip). **Default Value**: []
ssp.ingress.gatewayApi.httpRouteAnnotations	Annotations for HTTPRoute resources (e.g., konghq.com/strip-path) **Default Value**: {}
ssp.ingress.fullVanity.enabled	Enables or disables full vanity host configuration. **Default Value**: false
ssp.ingress.fullVanity.hosts	List of full vanity hosts with their TLS secret names. **Default Value**: []
ssp.ingress.endUsersVanity.enabled	Enables or disables end users vanity host configuration. **Default Value**: false
ssp.ingress.endUsersVanity.ingressClassName	Ingress class name for end users vanity hosts **Default Value**: (empty)
ssp.ingress.endUsersVanity.allowScim	Allow SCIM endpoints on end users vanity hosts **Default Value**: false

ssp.ingress.endUsersVanity.hosts	List of end users vanity hosts with their TLS secret names **Default Value**: []
ssp.ingress.mgmt.enabled	keep mgmt host and/or mgmt tls empty to use values from the main ingress section **Default Value**: true
ssp.ingress.mgmt.ingressClassName	Ingress class name for management interface **Default Value**: (empty)
ssp.ingress.mgmt.host	Host for management interface **Default Value**: (empty)
ssp.ingress.mgmt.tls.enabled	Enables or disables TLS for management interface **Default Value**: true
ssp.ingress.mgmt.tls.secretName	TLS secret name for management interface **Default Value**: (empty)
ssp.ingress.mgmt.tls.host	TLS host for management interface **Default Value**: (empty)
ssp.ingress.mgmt.tls.generateSelfSignedCert	• Generate a self-signed cert for the mgmt hostname. Only used when mgmt has a separate secretName. • If secretName matches the main tls.secretName, the mgmt host is added to the main cert's SANs instead. **Default Value**: false
ssp.ingress.nginx.timeoutSecs	Nginx-specific timeout in seconds for proxy read/send operations. **Default Value**: "60"
ssp.ingress.nginx.proxySslVerifyDepth	Sets the verification depth in the proxied server certificates chain. **Default Value**: 1
ssp.ingress.nginx.configSnippetsEnabled	Enables or disables nginx configuration snippets. **Default Value**: true
ssp.ingress.nginx.clientHeaderBuffersNumber	Sets the number of buffers used for reading large client request headers. **Default Value**: "4"
ssp.ingress.nginx.clientHeaderBufferSize	Sets buffer size for reading client request headers. **Default Value**: "128k"
ssp.ingress.nginx.clientHeaderBusyBufferSize	• Buffer math: clientHeaderBusyBufferSize must be < (clientHeaderBuffersNumber × clientHeaderBufferSize) - clientHeaderBufferSize • Example: 4 × 128k = 512k, so busy-buffers-size must be < 512k - 128k = 384k (256k is valid) **Default Value**: "256k"

ssp.ingress.openShiftRoute.isDefaultTenantInPath	• Note: OpenShift HAProxy does not support per-route header buffer size annotations. • The effective max request header size is headerBufferBytes - headerBufferMaxRewriteBytes (default: 32768 - 8192 = 24KB), which is too small for SSP's large JWT tokens. • Increase at the IngressController level (OpenShift 4.10+): • oc patch ingresscontroller/default -n openshift-ingress-operator --type=merge -p '{"spec":{"tuningOptions": {"headerBufferBytes":131072,"headerBufferMaxRewriteBytes":1638 • Note: Wildcard hosts (*.fqdn) require wildcardPolicy: WildcardsAllowed on the IngressController. • If wildcards are disabled (default in OpenShift 4.x), the wildcard route shows RouteNotAdmitted. • isDefaultTenantInPath:false implies routes uses <defaultTenant>.<ssp-fqdn>/<defaultTenantPlaceholderInPath>, otherwise routing uses <ssp-fqdn>/<defaultTenant> **Default Value**: false
ssp.ingress.openShiftRoute.defaultTenant	Default tenant name for OpenShift route configuration. **Default Value**: default
ssp.ingress.openShiftRoute.defaultTenantPlaceholderInPath	Default tenant placeholder in path for OpenShift route configuration. **Default Value**: common
ssp.ingress.openShiftRoute.timeout	Timeout for OpenShift route connections. **Default Value**: "60s"
ssp.ingress.openShiftRoute.tunnelTimeout	Tunnel timeout for OpenShift route connections. **Default Value**: "1h"

Already Have a Gateway? Use It!

Configuration details for using an existing Gateway, Traffic Policies and mTLS requirements.

If your platform team already manages Gateway resources, IDSP can attach to your existing Gateway, you do not need to create a new one. IDSP creates only the HTTPRoutes and BackendTLSPolicy, attaching them to your existing Gateway.

Quick setup with existing Gateway:

```
ssp:
  ingress:
    type: gatewayapi
    gatewayApi:
      createGateway: false                    # Don't create new Gateway
      existingGateway: "your-gateway-name"    # Name of your existing Gateway
      existingGatewayNamespace: ""            # Namespace of existing Gateway (defaults to release namespace)
      gatewayClassName: "your-gateway-class"  # Required for HTTPRoute parentRef
```

NOTE
Cross-namespace Gateways: If your existing Gateway lives in a different namespace than IDSP, set `existingGatewayNamespace` to that namespace. The Gateway must have `allowedRoutes.namespaces.from: All` (or a matching Selector) for cross-namespace HTTPRoutes to be accepted.

Requirements for your existing Gateway:

Requirement	Details
HTTPS Listener	Must have a listener on port 443 with `protocol: HTTPS`
TLS Certificate	Listener must reference a Secret with TLS cert for IDSP's hostname(s)
Same Namespace	Gateway must be in the same namespace as IDSP (or use `ReferenceGrant` for cross-namespace)
Hostname Coverage	TLS cert must cover: main host, `*.{main-host}` (wildcard for tenants), mgmt host, vanity hosts

NOTE
Vanity Host Impact: When using an externally-managed Gateway (createGateway=false), the chart creates HTTPRoutes but cannot add listeners to the Gateway. Gateway API listeners match HTTPRoutes by hostname, an HTTPRoute is rejected if no listener matches its hostname. This means that adding a new vanity host requires two steps:

1. Edit the Gateway to add a new HTTPS listener for the vanity hostname.
2. Run helm upgrade to create the HTTPRoute.
   Note that the same applies to any additional hostnames (example, sample-app hostnames if sharing the Gateway).

For detailed information on how to create and use Vanity Hosts in IDSP, see Creating Vanity URLs.

Traffic Policies and mTLS with an Existing Gateway

When the chart creates its own Gateway (`createGateway: true`), it automatically creates all controller-specific resources: traffic policies (timeouts, buffer limits, connection settings), health checks, and mTLS configuration. When you bring your own Gateway, **you are responsible for some of these resources**, depending on your setup.

What you may need to create

When the existing Gateway is in the **same namespace** as IDSP, the chart handles everything automatically.

When the existing Gateway is in a **different namespace**, you may need to create:

Resource	When needed	Notes
ReferenceGrant	Cross-namespace Gateway	Authorizes IDSP's HTTPRoute to attach to the Gateway. Set `allowCrossNamespaceCreate: true` to let the chart create or create it manually in the Gateway's namespace
Traffic policies	Cross-namespace Gateway	Controller-specific policies (e.g., Envoy `ClientTrafficPolicy`) must be in the Gateway's namespace. Set `allowCrossNamespaceCreate: true` to let the chart create these, or create them manually
mTLS policies	Cross-namespace Gateway + `mtls.enabled: true`	Same as traffic policies, requires `allowCrossNamespaceCreat` `true`

NOTE
Why `allowCrossNamespaceCreate` ? Controller-specific policies must be in the **same namespace as the Gateway** they target. When `allowCrossNamespaceCreate: true` , the chart creates these resources in the Gateway's namespace. When `false` (default), the Helm service account may not have RBAC outside the release namespace — create the resources manually instead.

NOTE
Mgmt and enduser gateways: For separate management or end-user Gateways using `existingGateway`, traffic policies are only created when you explicitly set `createTrafficPolicies: true`. This is because a shared existing Gateway may already have traffic policies managed by another team.

Per-controller requirements

The chart creates traffic policies automatically — in the release namespace for same-namespace setups, and in the **Gateway's namespace** for cross-namespace setups (when `allowCrossNamespaceCreate: true`). The following table lists the controller-specific resources and their purpose. If you manage these policies yourself (for example, `allowCrossNamespaceCreate: false` with a cross-namespace Gateway), refer to the following examples.

Envoy Gateway:

Resource	Purpose	Required?
`ClientTrafficPolicy`	Connection buffer limits, request timeout, header case preservation	Recommended
`ClientTrafficPolicy` — mTLS section	Client cert validation + XFCC header forwarding	Required for mTLS
`BackendTrafficPolicy`	Backend timeouts, connection idle timeout	Recommended

Example `ClientTrafficPolicy` with mTLS (the chart creates this automatically in the Gateway's namespace):

```
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: ClientTrafficPolicy
metadata:
  name: ssp-client-traffic
  namespace: <gateway-namespace>      # Must match Gateway's namespace
spec:
  targetRef:
    group: gateway.networking.k8s.io
    kind: Gateway
    name: <your-gateway-name>
  connection:
    bufferLimit: 524288               # 512KB — matches nginx proxy-buffer-size × proxy-buffers-number
  timeout:
    http:
      requestReceivedTimeout: 60s
  http1:
    preserveHeaderCase: true
  # --- mTLS section (only needed when mtls.enabled: true) ---
  tls:
    clientValidation:
      caCertificateRefs:
        - kind: Secret
          group: ""
          name: <ca-cert-secret>      # Secret containing CA cert for client validation
      optional: true                  # Accept requests without client cert (optional mTLS)
  headers:
    xForwardedClientCert:
      mode: SanitizeSet               # Strip client-injected XFCC, set from real cert
      certDetailsToAdd:
        - Cert                        # Include full URL-encoded PEM in XFCC header
```

NOTE
How Envoy mTLS forwarding works: When `clientValidation` is configured, Envoy sets the `x-forwarded-client-cert` (XFCC) header with `Cert="<URL-encoded PEM>"` . The `mode: SanitizeSet` ensures the header is sanitized (client-injected values are stripped). ssp-router extracts the cert from the XFCC `Cert=` field and normalizes it to `X-CLIENT-CERT` for backend services.

Example `BackendTrafficPolicy` (create in IDSP's namespace, targets HTTPRoute):

```
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: BackendTrafficPolicy
metadata:
  name: ssp-backend-traffic
  namespace: <idsp-namespace>         # Same namespace as the HTTPRoute
spec:
  targetRefs:
    - group: gateway.networking.k8s.io
      kind: HTTPRoute
      name: <release>-ssp-router-full
  timeout:
    tcp:
      connectTimeout: 15s
    http:
      requestTimeout: 60s
      connectionIdleTimeout: 3600s
```

NOTE
Note: `BackendTrafficPolicy` targets the **HTTPRoute** (not the Gateway), so it is created in IDSP's namespace and works even with cross-namespace Gateways. The chart creates this automatically.

AWS Load Balancer Controller:

Resource	Purpose	Required?
`LoadBalancerConfiguration`	Listener configuration (e.g., idle timeout)	Optional
`TargetGroupConfiguration`	Health check, target type, protocol, timeouts	Recommended

The `LoadBalancerConfiguration` must be in the Gateway's namespace. It is used for listener-level settings (for example, idle timeout). The chart creates it automatically when needed.

NOTE
mTLS note: ALB does not support mTLS for IDSP. ALB's verify mode requires all requests to present a client cert (no optional mode), and passthrough mode does not validate the cert chain. For mTLS on EKS, use Envoy Gateway instead.

The `TargetGroupConfiguration` targets the Service (in IDSP's namespace), the chart creates this automatically.

Avi AKO:

Resource	Purpose	Required?
Application Profile (Avi Controller)	Client cert request mode + PKI Profile	Required for mTLS
HTTP Policy Set (Avi Controller)	Forward cert via `X-Whole-Cert` header	Required for mTLS

Avi mTLS configuration is done on the **Avi Controller** (not through Kubernetes CRDs).

GKE L7 Gateway / Azure AGC:

These controllers do **not** support forwarding client certificates to backends. Use Envoy Gateway instead if you need mTLS. See Frontend mTLS (Client Certificate Authentication) for workarounds.

Quick reference: existing Gateway checklist

Use this checklist when setting up IDSP with an existing Gateway:

- [ ] Gateway has HTTPS listener(s) covering all IDSP hostnames
- [ ] TLS certificate covers main host + wildcard (`*.main-host`) + mgmt host
- [ ] `allowedRoutes.namespaces.from: All` (or matching Selector) if cross-namespace
- [ ] **Cross namespace?** → Set `allowCrossNamespaceCreate: true` to let the chart create resources in the Gateway's namespace
- [ ] **Cross namespace?** → `ReferenceGrant` created by chart (set `createReferenceGrant: false` if managing separately)
- [ ] **Same namespace?** → Traffic policies created automatically by chart
- [ ] **Cross namespace?** → Traffic policies created in Gateway's namespace (requires `allowCrossNamespaceCreate: true`)
- [ ] **mTLS needed?** → mTLS policy included in traffic policies when `mtls.enabled: true`
- [ ] **mTLS needed?** → CA cert Secret accessible from Gateway's namespace
- [ ] Set `ssp.ingress.mtls.enabled: true` and `caCertSecretName` in Helm values

Supported Gateways/Ingress Controllers in IDSP

This topic includes the links to all the supported Gateway platforms:

- AWS Gateway API Controller
- GKE L7 Gateway Controller
- Azure Application Gateway
- AVI AKO Gateway API Controller
- Envoy Gateway
- OpenShift

AWS Gateway API Controller

Amazon Web Services (AWS) provides native Gateway API support through the AWS Load Balancer Controller (v3.x), which provisions Application Load Balancers (ALBs) for Gateway resources in EKS environments.

> **NOTE**
> AWS Load Balancer Controller does **not** support BackendTLSPolicy for backend certificate validation. HTTPS to backend services still works (traffic is encrypted), but the ALB does not verify the backend's TLS certificate. If backend cert validation is required, use Envoy Gateway on EKS instead.

For Official documentation, see:AWS Load Balancer Controller.

Overview

This topic details the configuration and deployment of the Gateway API on Amazon EKS using the AWS Load Balancer Controller. Deploying on AWS introduces unique requirements compared to standard Ingress setups, such as relying on specific custom resources (`LoadBalancerConfiguration` and `TargetGroupConfiguration`) instead of standard annotations, and utilizing AWS Certificate Manager (ACM) for TLS.

To streamline deployment, the IDSP Helm chart auto-detects AWS environments (via the `amazon*` or `aws*` gatewayClassName prefix) and applies necessary AWS-specific workarounds automatically:

- Skips `BackendTLSPolicy` creation, as it is not supported by the AWS Load Balancer Controller.
- Sets `appProtocol: HTTPS` on the ssp-router Service to enable HTTPS backend connections.
- Creates a `TargetGroupConfiguration` resource with the required `targetType: ip` , health check (`/actuator/health`), and deregistration delay settings (provided the CRD exists in the cluster).

Aspect	Details
Gateway API support	GatewayClass (v1), Gateway (v1), HTTPRoute (v1)
BackendTLSPolicy	Not supported — HTTPS to backends works via `appProtocol: HTTPS` but without cert validation
Health Check	`TargetGroupConfiguration` CRD (`elbv2.k8s.aws/v1beta1`) — auto-created by Helm chart
Traffic Policy	`TargetGroupConfiguration` — target type, health check, deregistration delay
Platforms	EKS only
EKS Requirements	AWS LB Controller v3.x with `ALBGatewayAPI` feature gate enabled, IRSA

IMPORTANT
AWS Load Balancer Controller Gateway API mode behaves **very differently** from its Ingress mode. Configuration that works with Ingress (annotations, tag-based subnet discovery) does **not** work with Gateway API. The controller uses dedicated CRDs instead.

Infrastructure Setup for AWS Gateway API Controller

Step-by-Step Setup

1. Set environment variables:
```
export CLUSTER_NAME="<your-eks-cluster-name>"
export REGION="<aws-region>"
export ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
```
2. Verify the OIDC provider is registered. IRSA requires an OIDC provider associated with the EKS cluster. Most EKS clusters have this enabled by default.
```
OIDC_ID=$(aws eks describe-cluster --name $CLUSTER_NAME --region $REGION \
  --query "cluster.identity.oidc.issuer" --output text | cut -d'/' -f5)

# Check if OIDC provider exists in IAM:
aws iam list-open-id-connect-providers | grep $OIDC_ID

# If not found, create it:
# eksctl utils associate-iam-oidc-provider --cluster $CLUSTER_NAME --region $REGION --approve
```
3. Create the IAM policy (if it does not already exist):
```
# Download the official IAM policy document:
curl -o /tmp/iam-policy.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/main/docs/install/iam_policy.json

# Create the policy:
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerIAMPolicy \
  --policy-document file:///tmp/iam-policy.json
```

```
# If policy already exists, note the ARN:
POLICY_ARN=$(aws iam list-policies --scope Local \
  --query "Policies[?PolicyName=='AWSLoadBalancerControllerIAMPolicy'].Arn" --output text)
echo "Policy ARN: $POLICY_ARN"
```

4. Create the IAM role with an IRSA trust policy:

```
OIDC_ISSUER=$(aws eks describe-cluster --name $CLUSTER_NAME --region $REGION \
  --query "cluster.identity.oidc.issuer" --output text)
OIDC_PROVIDER=${OIDC_ISSUER#https://}

cat > /tmp/trust-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::${ACCOUNT_ID}:oidc-provider/${OIDC_PROVIDER}"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "${OIDC_PROVIDER}:aud": "sts.amazonaws.com",
                    "${OIDC_PROVIDER}:sub": "system:serviceaccount:kube-system:aws-load-balancer-
controller"
                }
            }
        }
    ]
}
EOF

ROLE_NAME="AmazonEKSLoadBalancerControllerRole-${CLUSTER_NAME}"

aws iam create-role \
  --role-name $ROLE_NAME \
  --assume-role-policy-document file:///tmp/trust-policy.json

aws iam attach-role-policy \
  --role-name $ROLE_NAME \
  --policy-arn $POLICY_ARN

ROLE_ARN=$(aws iam get-role --role-name $ROLE_NAME --query "Role.Arn" --output text)
echo "Role ARN: $ROLE_ARN"
```

5. Create the Kubernetes service account:

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: ServiceAccount
metadata:
  name: aws-load-balancer-controller
  namespace: kube-system
  annotations:
```

```
    eks.amazonaws.com/role-arn: ${ROLE_ARN}
EOF
```

6. Install the Gateway API CRDs:

```
kubectl apply -f https://github.com/kubernetes-sigs/gateway-api/releases/download/v1.2.0/standard-
install.yaml

# Verify:
kubectl get crd gateways.gateway.networking.k8s.io
```

7. Install the AWS Load Balancer Controller via Helm. Use `controllerConfig.featureGates.ALBGatewayAPI=true` to enable Gateway API support.

```
helm repo add eks https://aws.github.io/eks-charts
helm repo update eks

VPC_ID=$(aws eks describe-cluster --name $CLUSTER_NAME --region $REGION \
  --query "cluster.resourcesVpcConfig.vpcId" --output text)

helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=$CLUSTER_NAME \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller \
  --set region=$REGION \
  --set vpcId=$VPC_ID \
  --set "controllerConfig.featureGates.ALBGatewayAPI=true"

# Verify controller pods are running:
kubectl get pods -n kube-system -l app.kubernetes.io/name=aws-load-balancer-controller

# Verify feature gate is enabled:
kubectl get deployment -n kube-system aws-load-balancer-controller \
  -o jsonpath='{.spec.template.spec.containers[0].args}'
```

8. Create the GatewayClass. The controller does not auto-create a GatewayClass. The `controllerName` must be exactly `gateway.k8s.aws/alb` .

```
cat <<EOF | kubectl apply -f -
apiVersion: gateway.networking.k8s.io/v1
kind: GatewayClass
metadata:
  name: amazon-alb
spec:
  controllerName: gateway.k8s.aws/alb
EOF

# Verify GatewayClass is Accepted:
kubectl get gatewayclass
```

9. Verify the Gateway API controllers are running:

```
kubectl logs -n kube-system -l app.kubernetes.io/name=aws-load-balancer-controller \
  --tail=50 | grep "Starting Controller"
```

You should see `gateway.k8s.aws/alb` and `aws-lbc-gateway-class-controller` among the started controllers.

AWS-Specific CRDs and Configuration

LoadBalancerConfiguration CRD (Required)

The `LoadBalancerConfiguration` CRD controls ALB settings. It is **required** for Gateway API mode.

```
apiVersion: gateway.k8s.aws/v1beta1
kind: LoadBalancerConfiguration
metadata:
  name: <release>-ssp-gateway-config
  namespace: <namespace>
spec:
  scheme: internet-facing               # or "internal"
  loadBalancerSubnets:
    - identifier: subnet-aaaa           # Public subnets with internet gateway
    - identifier: subnet-bbbb
    - identifier: subnet-cccc
  # Optional: use existing security group instead of creating new one
  securityGroups:
    - sg-xxxx
  manageBackendSecurityGroupRules: false
```

Attach it to the Gateway via `spec.infrastructure.parametersRef` :

```
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: <release>-ssp-gateway
spec:
  gatewayClassName: amazon-alb
  infrastructure:
    parametersRef:
      group: gateway.k8s.aws
      kind: LoadBalancerConfiguration
      name: <release>-ssp-gateway-config
  listeners:
    # ...
```

Finding subnets:

```
# List subnets in the cluster VPC
VPC_ID=$(aws eks describe-cluster --name <cluster> --region <region> \
  --query "cluster.resourcesVpcConfig.vpcId" --output text)

aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" \
  --query 'Subnets[].{ID:SubnetId,AZ:AvailabilityZone,CIDR:CidrBlock,Public:MapPublicIpOnLaunch}' \
  --output table
```

Use **public** subnets (MapPublicIpOnLaunch=true) for internet-facing ALBs.

TargetGroupConfiguration CRD (Required)

The `TargetGroupConfiguration` CRD controls how the ALB connects to ssp-router pods.

```
apiVersion: gateway.k8s.aws/v1beta1
kind: TargetGroupConfiguration
metadata:
  name: <release>-ssp-router-tgc
  namespace: <namespace>
spec:
  targetReference:
    group: ""
    kind: Service
    name: <release>-ssp-router
  defaultConfiguration:
    targetType: ip                        # Required: ssp-router uses ClusterIP
    protocol: HTTPS                       # Required: ssp-router listens on HTTPS
    protocolVersion: HTTP1
    healthCheckConfig:
      healthCheckPath: /actuator/health  # Required: must work without Host header
      healthCheckProtocol: HTTPS
      healthCheckInterval: 15
      healthyThresholdCount: 2
      unhealthyThresholdCount: 3
      matcher:
        httpCode: "200-399"
```

Why `targetType: ip` Is Required:

The default target type is `instance` , which requires a NodePort service. ssp-router is a ClusterIP service. Set `targetType: ip` to register pod IPs directly as ALB targets. Without it you get: `TargetGroup port is empty. When using Instance targets, your service must be of type 'NodePort' or 'LoadBalancer'.`

Why `/actuator/health` Is the Health Check Path:

ALB health checks do **not** send a `Host` header. ssp-router requires a valid Host header to resolve the tenant. Without it, any tenant-scoped path returns 403. `/actuator/health` is a Spring Boot actuator endpoint that returns 200 regardless of Host header.

Health Check Path	Works Without Host Header?	Notes
`/`	No (403)	ssp-router returns 403
`/actuator/health`	No (403)	Requires Host header for tenant
`/actuator/health`	**Yes (200)**	Spring Boot actuator, no tenant context needed

AWS-Specific CRDs Summary

CRD	API Group	Purpose	Requir
`LoadBalancerConfiguration`	`gateway.k8s.aws/v1beta1`	ALB settings: subnets, scheme, security groups	**Yes**
`TargetGroupConfiguration`	`elbv2.k8s.aws/v1beta1`	Target settings: IP mode, protocol, health checks	**Yes** (Helm-ma
`ListenerRuleConfiguration`	`gateway.k8s.aws/v1beta1`	Custom listener rule settings	No (not neede IDSP)

NOTE
`TargetGroupConfiguration` is now created by the Helm chart automatically when `gatewayType=eks` (via `ssprouter-traffic-policy.yaml`). It configures `targetType: ip` , health check path `/actuator/health` , and deregistration delay. `LoadBalancerConfiguration` must still be created manually outside the Helm chart, as it contains environment-specific settings (subnet IDs, security groups).

Deploying or Migrating to Gateway API

Learn how to deploy a fresh installation of IDSP using the Gateway API on AWS, or migrate an existing NGINX Ingress deployment to the Gateway API standard.

Pre-requisite Check

NOTE
Before proceeding with the specific checks mentioned in this section, review and complete all items in the Prerequisites for All Deployments list.

Run the following commands to verify all prerequisites are in place before deploying IDSP:

- **Verify Gateway API CRDs are installed:**
  ```
  kubectl get crd gateways.gateway.networking.k8s.io
  ```
- **Verify AWS Load Balancer Controller pods are running:**
  ```
  kubectl get pods -n kube-system -l app.kubernetes.io/name=aws-load-balancer-controller
  ```
- **Verify the ALBGatewayAPI feature gate is enabled:**
  ```
  kubectl get deployment -n kube-system aws-load-balancer-controller \
    -o jsonpath='{.spec.template.spec.containers[0].args}'
  ```
  Expected output: The output should contain `ALBGatewayAPI=true` .
- **Verify the GatewayClass is available and accepted:**
  ```
  kubectl get gatewayclass
  ```
  Expected output: The list should display `amazon-alb` with a status of `Accepted=True` .
- **Verify the LoadBalancerConfiguration is created:**
  ```
  kubectl get loadbalancerconfiguration -n ssp
  ```
- **Verify the TLS certificate Secret exists:**
  ```
  kubectl get secret <tls-secret-name> -n ssp
  ```
- **Verify the ACM certificate exists for hostnames (AWS):**
  ```
  aws acm list-certificates --region <region>
  ```
  If no ACM certificate exists, import your TLS certificate into ACM. The ALB controller auto-discovers ACM certificates by matching Gateway listener hostnames against the certificate SANs.
  ```
  aws acm import-certificate \
    --certificate fileb://tls.crt \
    --private-key fileb://tls.key \
  ```

```
    --certificate-chain fileb://ca.crt \
    --region <region>
```

Minimum ClusterRole for AWS Load Balancer Controller:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: gateway-api-admin
rules:
  - apiGroups: ["gateway.networking.k8s.io"]
    resources: ["gateways", "httproutes"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
  - apiGroups: ["elbv2.k8s.aws"]
    resources: ["targetgroupconfigurations"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
  - apiGroups: ["gateway.k8s.aws"]
    resources: ["loadbalancerconfigurations"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

Before You Deploy

NOTE

- Gateway ownership: The Gateway can be created by Helm (`createGateway: true` , default) or pre-created by the platform team (`createGateway: false` + `existingGateway`). To know how to use an already existing Gateway, see Already Have a Gateway? section for details.
- Set these Environment Variables:

```
export RELEASENAME="<SSP_RELEASENAME>"
export NAMESPACE="<SSP_NAMESPACE>"
export HELM_REPO='ssp_helm_charts'
```

- Run the following command to export the current override values of your Helm release to `ssp-override.yaml` . This lets you review, back up, or reuse your existing configuration:

```
helm get values ${RELEASENAME} -n ${NAMESPACE} >ssp-override.yaml
```

Deploy IDSP with Gateway API (Fresh Install)

1. **Ensure AWS LB Controller is installed**
   Verify that the AWS Load Balancer Controller is installed with the `ALBGatewayAPI` feature gate enabled and that the GatewayClass exists. See the Infrastructure Setup section.
2. **Create LoadBalancerConfiguration (manual — not Helm-managed)**
   Create the `LoadBalancerConfiguration` resource with your environment-specific subnet IDs. See the LoadBalancerConfiguration CRD section for the full spec.

```
cat <<EOF | kubectl apply -n ssp -f -
apiVersion: gateway.k8s.aws/v1beta1
kind: LoadBalancerConfiguration
metadata:
  name: <release>-ssp-gateway-config
spec:
  scheme: internet-facing
```

```
    loadBalancerSubnets:
      - identifier: <subnet-1>
      - identifier: <subnet-2>
      - identifier: <subnet-3>
  EOF
```

3. **Deploy IDSP with Gateway API**
   Run the Helm install with Gateway API settings. The chart auto-detects the AWS gateway class and skips `BackendTLSPolicy` creation.

```
helm upgrade --install ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.gatewayApi.gatewayClassName=amazon-alb \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.createBackendTLSPolicy=false
```

4. **Verify resources**
   Confirm that the Gateway, HTTPRoute, and TargetGroupConfiguration resources were created.

```
kubectl get gateway,httproute -n ${NAMESPACE}
kubectl get targetgroupconfiguration -n ${NAMESPACE}
```

5. **Verify Gateway references the LoadBalancerConfiguration**
   Check that the Gateway has an `infrastructure.parametersRef` pointing to your `LoadBalancerConfiguration`.

```
kubectl get gateway -n ${NAMESPACE} -o yaml | grep -A5 parametersRef
```

6. **Test connectivity**
   Retrieve the ALB DNS name from the Gateway status and send a test request.

```
ALB_DNS=$(kubectl get gateway -n ${NAMESPACE} -o jsonpath='{.items[0].status.addresses[0].value}')
curl -sk --resolve "your-host:443:$ALB_DNS" \
"https://your-host/default/.well-known/openid-configuration"
```

What Helm creates for EKS:

Resource	Name Pattern	Purpose
Gateway	`<release>-ssp-gateway`	HTTPS listeners with TLS termination
HTTPRoute	`<release>-ssp-router-full`	Routes all traffic to ssp-router
TargetGroupConfiguration	`<release>-ssp-router-tgc`	Target type (IP), health check, deregistration delay

> **NOTE**
> `LoadBalancerConfiguration` must be created manually. See AWS-Specific CRDs section.

Migrate IDSP from Nginx to Gateway API

> **NOTE**
> Before proceeding, complete all steps in the Before You Deploy section.

1. **Upgrade the existing deployment:**
   Run the following command to upgrade an existing NGINX-based IDSP deployment to Gateway API on AWS:

```
helm upgrade ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.previousController.enabled=true \
  --set ssp.ingress.previousController.ingressClassName=nginx \
  --set ssp.ingress.gatewayApi.gatewayClassName=amazon-alb \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.createBackendTLSPolicy=false
```

2. **Remove the old controller:**
   After verifying that the Gateway API works and switching your DNS, run the following command to remove the old controller:

```
helm upgrade ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.previousController.enabled=false \
  --set ssp.ingress.gatewayApi.gatewayClassName=amazon-alb \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.createBackendTLSPolicy=false
```

Deploy the Sample App to Test Connectivity

Deploy the sample-app with AWS ALB to test connectivity before or alongside IDSP:

1. **Create the LoadBalancerConfiguration (manual — not Helm-managed):**

```
cat <<EOF | kubectl apply -n <namespace> -f -
apiVersion: gateway.k8s.aws/v1beta1
kind: LoadBalancerConfiguration
metadata:
  name: <release>-ssp-sample-app-gateway-config
spec:
  scheme: internet-facing
  loadBalancerSubnets:
    - identifier: <subnet-1>
    - identifier: <subnet-2>
    - identifier: <subnet-3>
EOF
```

2. **Deploy the sample app with Gateway API (EKS):**

```
helm upgrade --install ${RELEASENAME} ${HELM_REPO}/ssp-sample-app -n ${NAMESPACE} \
  -f your-sample-values.yaml \
  --set ingress.type=gatewayapi \
  --set ingress.gatewayApi.gatewayClassName=amazon-alb \
  --set ingress.gatewayApi.createGateway=true \
  --set ingress.gatewayApi.createBackendTLSPolicy=false
```

3. **Verify resources:**

```
kubectl get gateway,httproute,targetgroupconfiguration -n <namespace>
```

4. **Test connectivity:**

```
ALB_DNS=$(kubectl get gateway -n <namespace> -o jsonpath='{.items[0].status.addresses[0].value}')
curl -sk --resolve "your-host:443:$ALB_DNS" "https://your-host/sample-app/"
```

> **NOTE**
> All sample-app services use ClusterIP, so `targetType: ip` is required. The chart creates a TargetGroupConfiguration for each backend service automatically when `gatewayType=aws-alb` is detected.

Troubleshooting AWS Deployments

Condition

The following issues may occur during the deployment or migration of the Gateway API on AWS.

Cause: GatewayClass stays Unknown

Check: Run `kubectl get gatewayclass -o yaml`

Remedy

Verify `controllerName` is exactly `gateway.k8s.aws/alb` and the `ALBGatewayAPI` feature gate is enabled.

Cause: No gateway controllers in logs

Check: Run `kubectl logs ... | grep gateway`

Remedy

Enable the feature gate by running `--set "controllerConfig.featureGates.ALBGatewayAPI=true"` and restart.

Cause: Controller pods CrashLoopBackOff

Check: Run `kubectl logs -n kube-system -l app.kubernetes.io/name=aws-load-balancer-controller`

Remedy

Check IRSA: verify the service account annotation and IAM role trust policy.

Cause: ALB not created after Gateway

Check: Run `kubectl describe gateway <name>`

Remedy

Check IAM permissions (AWSLoadBalancerControllerIAMPolicy), VPC subnets, and security groups.

Cause: Target group unhealthy

Check: Run `aws elbv2 describe-target-health --target-group-arn <arn>`

Remedy

Verify ssp-router pods are running and security groups allow traffic from the ALB.

Cause: Gateway API CRDs not found

Check: Run `kubectl get crd | grep gateway`

Remedy

Install the CRDs by running: `kubectl apply -f https://...gateway-api/.../standard-install.yaml`

Cause: "Evaluated 0 subnets"

Check: Review the Controller logs.

Remedy

Create a `LoadBalancerConfiguration` with explicit subnet IDs — tag-based discovery does not work.

Cause: SecurityGroupLimitExceeded

Check: Review the Controller logs.

Remedy

Specify an existing security group in the `LoadBalancerConfiguration` .

Cause: "no certificate found for host"

Check: Review the Controller logs.

Remedy

Import the certificate to ACM, then trigger reconciliation — see the Verifying ACM Certificates section.

Cause: 502 Bad Gateway after TGC change

Check: Run `aws elbv2 describe-target-health --target-group-arn <arn>`

Remedy

Ensure the TargetGroupConfiguration (TGC) has `protocol: HTTPS` — without it, the ALB connects via HTTP to ssp-router (which is HTTPS-only).

Cause: Gateway Accepted: False

Check: Run `kubectl get gateway <name> -o yaml` and check `status.conditions` .

Remedy

Check the controller logs; this is usually a certificate discovery failure blocking reconciliation.

GKE L7 Gateway Controller

Learn how to configure and deploy the Gateway API on Google Kubernetes Engine (GKE) using the native GKE Gateway Controller.

The GKE L7 Gateway Controller is Google's native implementation of the Kubernetes Gateway API. It provisions Cloud Load Balancing resources to manage HTTP and HTTPS traffic routing into your GKE clusters. By using the Gateway API on GKE, you benefit from a standardized, declarative model for service networking that integrates seamlessly with Google Cloud infrastructure.

> **NOTE**
> ## **Limitation:** GKE L7 Gateway does **not** validate backend certificates. If backend certificate validation is a strict requirement for your deployment, consider using Envoy Gateway on GKE instead.

For comprehensive official documentation on the GKE Gateway API, see: About GKE Gateway Controller.

Overview

When deploying the Identity Security Platform (IDSP) using the GKE L7 Gateway Controller, the deployment leverages native Google Cloud resources. The IDSP Helm chart streamlines this process by automatically detecting the GKE environment and applying several necessary configurations:

- **Backend TLS Policy:** Skips `BackendTLSPolicy` creation (which is not currently supported by GKE) and instead sets `appProtocol: HTTPS` on the `ssp-router` Service to enable encrypted backend connections.
- **Health Checks:** Automatically generates a `HealthCheckPolicy` custom resource (`networking.gke.io/v1`) for native HTTPS health checks to the `ssp-router` .
- **Traffic Policies:** Creates a `GCPBackendPolicy` for managing backend timeouts and connection draining.

The following table summarizes the Gateway API resource support and specific platform configurations when using the GKE Gateway Controller:

Aspect	Details
Gateway API support	GatewayClass (v1), Gateway (v1), HTTPRoute (v1)
BackendTLSPolicy	Not supported. HTTPS to backends works via `appProtocol: HTTPS` but without cert validation
Health Check	`HealthCheckPolicy` CRD (`networking.gke.io/v1`) — auto-created by Helm chart
Traffic Policy	`GCPBackendPolicy` , backend timeout, connection draining
Platforms	GKE only

GatewayClass Options

GKE offers different GatewayClasses depending on your traffic routing needs (such as global, regional, or internal access). Choosing the right GatewayClass determines how the Google Cloud Load Balancer is provisioned and whether a proxy-only subnet is required.

GatewayClass	Type	Proxy Subnet Required	Description
`gke-l7-global-external-managed`	Global External	#	Anycast IP, g
`gke-l7-regional-external-managed`	Regional External	#	Single region
`gke-l7-rilb`	Internal	#	VPC-only acc

Choosing a GatewayClass:

- Use `gke-l7-global-external-managed` for multi-region or when you want the simplest setup (no proxy subnet needed).
- Use `gke-l7-regional-external-managed` when your deployment is single-region and you want a regional external IP. Requires a proxy-only subnet in the same region.
- Use `gke-l7-rilb` for internal-only access within the VPC.

The IDSP Helm chart auto-detects GKE based on the GatewayClass and automatically does the following:

- Creates a `HealthCheckPolicy` for HTTPS health checks
- Sets `appProtocol: HTTPS` on the ssp-router Service
- Skips BackendTLSPolicy creation (not supported by GKE)

Infrastructure Setup for GKE L7 Gateway API Controller
These steps describe how to prepare your Google Kubernetes Engine (GKE) environment for the Gateway API.

1. **Enable Gateway API on GKE cluster:**

```
gcloud container clusters update <CLUSTER_NAME> \
  --gateway-api=standard \
  --zone=<ZONE>
```

2. **Create proxy-only subnet (required for regional/internal only):**

```
gcloud compute networks subnets create proxy-only-subnet \
  --purpose=REGIONAL_MANAGED_PROXY \
  --role=ACTIVE \
  --region=<REGION> \
  --network=<VPC_NAME> \
  --range=<CIDR_RANGE>
```

Deploying or Migrating to GKE L7 Gateway API
Learn how to deploy a fresh installation of the Gateway API on Google Kubernetes Engine (GKE), or migrate an existing Nginx-based deployment to the GKE L7 Gateway API.

Pre-requisite Check

Run the following commands to verify that all prerequisites are correctly configured before deploying IDSP:

1. **Verify GatewayClass:** Ensure that the required GKE GatewayClass (such as `gke-l7-global-external-managed`) is available and has been accepted.
   ```
   kubectl get gatewayclass
   ```
   Expected Output: Look for GKE classes with `Accepted=True` .
2. **Verify TLS Certificate Secret:** Confirm that the secret containing your TLS certificates exists in the `ssp` namespace.
   ```
   kubectl get secret <tls-secret-name> -n ssp
   ```
3. **Check RBAC Permissions:** Ensure you have the necessary permissions to create Gateway API and GKE-specific networking resources.

```
kubectl auth can-i create gateways.gateway.networking.k8s.io -n ssp
kubectl auth can-i create httproutes.gateway.networking.k8s.io -n ssp
kubectl auth can-i create healthcheckpolicies.networking.gke.io -n ssp
kubectl auth can-i create gcpbackendpolicies.networking.gke.io -n ssp
```

4. **Verify Proxy-Only Subnet:** (Required only for regional or internal gateways) Check that a proxy-only subnet exists in your VPC.
   ```
   gcloud compute networks subnets list --filter="purpose=REGIONAL_MANAGED_PROXY"
   ```

Minimum ClusterRole for GKE L7 Gateway:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: gateway-api-admin
rules:
  - apiGroups: ["gateway.networking.k8s.io"]
    resources: ["gateways", "httproutes"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
  - apiGroups: ["networking.gke.io"]
    resources: ["healthcheckpolicies", "gcpbackendpolicies"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

Before You Deploy

NOTE

- Gateway ownership: The Gateway can be created by Helm (`createGateway: true` , default) or pre-created by the platform team (`createGateway: false` + `existingGateway`). To know how to use an already existing Gateway, see Already Have a Gateway? section for details.
- Set these Environment Variables:

```
export RELEASENAME="<SSP_RELEASENAME>"
export NAMESPACE="<SSP_NAMESPACE>"
export HELM_REPO='ssp_helm_charts'
```

- Run the following command to export the current override values of your Helm release to `ssp-override.yaml` . This lets you review, back up, or reuse your existing configuration:

```
helm get values ${RELEASENAME} -n ${NAMESPACE} >ssp-override.yaml
```

Deploying Symantec Identity Security Platform with Gateway API (Fresh Install)

1. Ensure GKE Gateway API is enabled. For regional or internal deployments, also ensure that a proxy-only subnet exists. See the Infrastructure Setup section for details.
2. Deploy Symantec Identity Security Platform with Gateway API. Choose the option that matches your deployment type:

   Option A — Global External (no proxy subnet needed):

```
helm install ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.gatewayApi.gatewayClassName=gke-l7-global-external-managed \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.createBackendTLSPolicy=false
```

   Option B — Regional External (requires proxy-only subnet):

```
helm install ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.gatewayApi.gatewayClassName=gke-l7-regional-external-managed \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.createBackendTLSPolicy=false
```

3. Verify that the Gateway API resources were created:

```
kubectl get gateway,httproute -n ${NAMESPACE}
kubectl get healthcheckpolicy,gcpbackendpolicy -n ${NAMESPACE}
```

4. Wait for the Gateway to receive an external IP address. This typically takes 2–5 minutes on GKE:

```
kubectl get gateway -n ${NAMESPACE} -w
```

5. Test connectivity through the Gateway:

```
GATEWAY_IP=$(kubectl get gateway -n ${NAMESPACE} -o jsonpath='{.items[0].status.addresses[0].value}')
curl -sk --resolve "your-host:443:$GATEWAY_IP" \
  "https://your-host/actuator/health"
```

What Helm creates for GKE:

Resource	Name Pattern	Purpose
Gateway	`<release>-ssp-gateway`	HTTPS listeners with TLS termination
HTTPRoute	`<release>-ssp-router-full`	Routes all traffic to ssp-router
HealthCheckPolicy	`<release>-ssp-router-healthcheck`	HTTPS health check for ssp-router
GCPBackendPolicy	`<release>-ssp-router-backend-policy`	Backend timeout, connection draining

Migrating Symantec Identity Security Platform from Nginx to Gateway API

> **NOTE**
> Before proceeding, complete all steps in the Before You Deploy section.

If upgrading an existing nginx-based Symantec Identity Security Platform deployment to Gateway API on GKE:

1. Upgrade your deployment (choose Option A or Option B):

   Option A — Global External (no proxy subnet needed):

```
helm upgrade ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.previousController.enabled=true \
  --set ssp.ingress.previousController.ingressClassName=nginx \
  --set ssp.ingress.gatewayApi.gatewayClassName=gke-l7-global-external-managed \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.createBackendTLSPolicy=false
```

   Option B — Regional External (requires proxy-only subnet):

```
helm upgrade ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.previousController.enabled=true \
  --set ssp.ingress.previousController.ingressClassName=nginx \
  --set ssp.ingress.gatewayApi.gatewayClassName=gke-l7-regional-external-managed \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.createBackendTLSPolicy=false
```

2. After verifying Gateway API works and switching DNS, remove the old controller:

```
# Use the same gatewayClassName you chose above

helm upgrade ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.previousController.enabled=false \
  --set ssp.ingress.gatewayApi.gatewayClassName=<same-class-as-above> \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.createBackendTLSPolicy=false
```

Deploying the Sample App to Test Connectivity

Deploy the sample-app with GKE Gateway to test connectivity before or alongside Symantec Identity Security Platform:

Option A — Global External (no proxy subnet needed):

```
helm upgrade --install ${RELEASENAME} ${HELM_REPO}/ssp-sample-app -n ${NAMESPACE} \
  -f your-sample-values.yaml \
  --set ingress.type=gatewayapi \
  --set ingress.gatewayApi.gatewayClassName=gke-l7-global-external-managed \
  --set ingress.gatewayApi.createGateway=true \
  --set ingress.gatewayApi.createBackendTLSPolicy=false
```

Option B — Regional External (requires proxy-only subnet):

```
helm upgrade --install ${RELEASENAME} ${HELM_REPO}/ssp-sample-app -n ${NAMESPACE} \
  -f your-sample-values.yaml \
  --set ingress.type=gatewayapi \
  --set ingress.gatewayApi.gatewayClassName=gke-l7-regional-external-managed \
  --set ingress.gatewayApi.createGateway=true \
  --set ingress.gatewayApi.createBackendTLSPolicy=false

# Wait for Gateway IP (GKE takes 2-5 minutes)
kubectl get gateway -n ${NAMESPACE} -w

# Test connectivity
GATEWAY_IP=$(kubectl get gateway -n ${NAMESPACE} -o jsonpath='{.items[0].status.addresses[0].value}')
curl -sk --resolve "your-host:443:$GATEWAY_IP" "https://your-host/sample-app/"
```

> **NOTE**
> **Note:** GKE does not support BackendTLSPolicy (`createBackendTLSPolicy=false`). GKE uses `appProtocol: HTTPS` on the spi-demo Service to connect via HTTPS. The chart creates a HealthCheckPolicy for spi-demo because it returns 302 on `/` .

Troubleshooting GKE L7 Deployments

"proxy-only subnetwork required" error

Remedy

Create a proxy-only subnet or use `gke-l7-global-external-managed` which does not require one.

Backend unhealthy (503 errors)

Remedy

Check the health check policy and the backend health:

```
kubectl get healthcheckpolicy -n ssp
# For global external:
gcloud compute backend-services get-health <backend-service-name> --global
```

```
# For regional external:
gcloud compute backend-services get-health <backend-service-name> --region=<REGION>
```

Backends showing HTTP instead of HTTPS

Remedy

Check the appProtocol on the service:

```
kubectl get service <release-name>-ssp-router -n ssp -o jsonpath='{.spec.ports[0].appProtocol}'
```

SPI demo returns 503 — health check type or port mismatch

GKE NEG health checks go directly to the pod IP, so the HealthCheckPolicy must use:

- `type: HTTPS` (not `HTTP`) — spi-demo is an HTTPS-only backend
- `port:` set to the **container port** (e.g., 8092), not the service port (e.g., 443)

Remedy

1. Check the GCE health check configuration:

   ```
   gcloud compute health-checks list --filter="name~spi" --format="table(name,type)" --regions=<REGION>
   ```

   If type shows HTTP instead of HTTPS, this is the issue.

2. Check backend health:

   ```
   gcloud compute backend-services get-health <backend-name> --region=<REGION>
   ```

   If healthState is UNHEALTHY, the health check is failing.

3. Verify the HealthCheckPolicy:

   ```
   kubectl get healthcheckpolicy -n <ns> -o yaml
   ```

   Check: config.type should be HTTPS, httpsHealthCheck.port should be the container port.

4. Fix the issue:

   Update the HealthCheckPolicy to use `type: HTTPS` with `httpsHealthCheck` and the container port. The Helm chart (v1034+) does this automatically.

Azure Application Gateway

Learn about Azure Application Gateway for Containers (AGC) and its integration with the Kubernetes Gateway API on AKS.

Overview

Azure Application Gateway for Containers (AGC) is a Microsoft-managed Layer 7 load balancing and traffic management service built specifically for Azure Kubernetes Service (AKS). It natively implements the Kubernetes Gateway API to securely and efficiently route incoming traffic to your applications. By leveraging AGC, you can take advantage of advanced routing capabilities, robust TLS termination, and seamless integration with the Azure ecosystem.

For comprehensive Official documentation, see Application Gateway for Containers.

To streamline the setup process on AKS, the IDSP Helm chart automatically detects the Azure environment via the `azure*` `gatewayClassName` prefix and does the following:

- Creates the Azure-specific BackendTLSPolicy CRD (`alb.networking.azure.io/v1`) for backend TLS validation
- Creates the Azure-specific HealthCheckPolicy CRD (`alb.networking.azure.io/v1`) with HTTPS health checks to `/actuator/health` , **recommended** because AGC's default health check hits `/` which returns 404 on ssp-router (though AGC can also fall back to the Kubernetes readiness probe when no HCP is present)
- Uses `gatewayAnnotations` to link the Gateway to the ApplicationLoadBalancer resource (the ALB instance in Azure)

> **NOTE**
> AGC assigns a unique FQDN (example, #.####.alb.azure.com) to the Gateway instead of a static IP. You must use "`dig`" or "`nslookup`" to resolve it when testing with "`curl --resolve`" .

Aspect	Details
Gateway API support	GatewayClass (v1), Gateway (v1), HTTPRoute (v1), BackendTLSPolicy (Azure CRD)
BackendTLSPolicy	Azure-specific CRD (`alb.networking.azure.io/v1`) — full CA certificate validation
HealthCheckPolicy	Azure-specific CRD (`alb.networking.azure.io/v1`) — **recommended** (AGC falls back to K8s readiness probe without it, but explicit HCP is safer)
Traffic Policy	Managed by Azure — no CRD needed
Platforms	AKS only
AKS Requirements	Azure CNI networking (flat or overlay), Workload Identity enabled
CNI Overlay Support	Requires ALB Controller ≥1.7.9 (recommended ≥1.9.13). Tested and working on both flat and overlay CNI.

Infrastructure Setup for Azure Gateway API Controller
Review the infrastructure requirements for the Azure Gateway API Controller, including required RBAC roles, provider registration, and a step-by-step setup guide for Azure Application Gateway for Containers (AGC).

Introduction

Before deploying the Gateway API, your Azure Kubernetes Service (AKS) cluster and Azure subscription must be properly configured. The following sections detail the mandatory Role-Based Access Control (RBAC) assignments for the ALB Controller's managed identity, the required subscription-level registration for the `Microsoft.ServiceNetworking` provider, and a complete step-by-step deployment procedure to provision the Azure Application Gateway for Containers (AGC) from scratch.

RBAC Roles Required

The ALB Controller managed identity needs **three** roles (not two — this is a common pitfall):

#	Role	Scope	Purpose
1	**AppGw for Containers Configuration Manager**	**MC resource group** (not AGC subnet)	Create and manage Traffic Controller resources (`Microsoft.ServiceNetworking/trafficCo` `write`). If scoped to the subnet, the ALB CR fails with `AuthorizationFailed.`
2	**Network Contributor**	AGC subnet	Manage network interfaces in the AGC subnet
3	**Reader**	MC resource group	Read resource group properties (required for AGC provi

NOTE

- **Critical:** Role #1 must be scoped to the **MC resource group**, not the AGC subnet. Many Azure docs and tutorials show the subnet scope, but this causes `403 AuthorizationFailed` when the controller tries to create the Traffic Controller resource. The error appears in the ApplicationLoadBalancer CR status (not in controller logs). .
- **CNI Overlay Note:** Azure CNI Overlay works with AGC. The ALB Controller (≥1.7.9) detects overlay automatically and manages overlay tunnels via `OverlayExtensionConfig` . Network Contributor scoped to the AGC subnet is sufficient — VNet scope is not required. If backends are UNHEALTHY on an overlay cluster, check the BackendTLSPolicy port first .

Microsoft.ServiceNetworking Provider

The `Microsoft.ServiceNetworking` resource provider must be registered at the subscription level. If not, the ALB controller fails with a `403` error about `Microsoft.ServiceNetworking/register/action` .

```
# Check registration status
az provider show --namespace Microsoft.ServiceNetworking --query registrationState -o tsv

# Register if needed (takes 1-2 minutes)
az provider register --namespace Microsoft.ServiceNetworking

# Wait for registration to complete
az provider show --namespace Microsoft.ServiceNetworking --query registrationState -o tsv
# Expected: Registered
```

After registering, restart the ALB controller to pick up the change:

```
kubectl rollout restart deployment alb-controller -n azure-alb-system
```

Step-by-Step Setup

Follow these steps to set up Azure AGC:

1. **Set Environment Variables**
   Define the environment variables used throughout the remaining steps.

   ```
   export RESOURCE_GROUP="<your-resource-group>"
   export AKS_NAME="<your-aks-cluster-name>"
   export LOCATION="<region>"
   export ALB_IDENTITY_NAME="alb-identity"
   ```

2. **Verify OIDC Issuer and Workload Identity**
   Both OIDC Issuer and Workload Identity must be enabled on the AKS cluster for the ALB Controller to authenticate to Azure.
   a. Check the current status:

      ```
      az aks show -g $RESOURCE_GROUP -n $AKS_NAME \
        --query "{oidcEnabled:oidcIssuerProfile.enabled,
       workloadIdentity:securityProfile.workloadIdentity.enabled}"
      ```

   b. If either value is `false` , enable them:

```
az aks update -g $RESOURCE_GROUP -n $AKS_NAME \
  --enable-oidc-issuer --enable-workload-identity
```

3. **Install Gateway API CRDs**
   a. Install the standard Gateway API Custom Resource Definitions:

```
kubectl apply -f https://github.com/kubernetes-sigs/gateway-api/releases/download/v1.2.0/standard-
install.yaml
```

   b. Verify the CRDs are installed:

```
kubectl get crd gateways.gateway.networking.k8s.io
```

4. **Create Managed Identity for ALB Controller**
   a. Create a user-assigned managed identity that the ALB Controller uses to manage Azure resources:

```
az identity create -g $RESOURCE_GROUP -n $ALB_IDENTITY_NAME -l $LOCATION
```

   b. Retrieve the identity details for use in later steps:

```
ALB_IDENTITY_CLIENT_ID=$(az identity show -g $RESOURCE_GROUP -n $ALB_IDENTITY_NAME --query clientId -o
 tsv)
ALB_IDENTITY_PRINCIPAL_ID=$(az identity show -g $RESOURCE_GROUP -n $ALB_IDENTITY_NAME --query
 principalId -o tsv)
echo "Client ID:    $ALB_IDENTITY_CLIENT_ID"
echo "Principal ID: $ALB_IDENTITY_PRINCIPAL_ID"
```

5. **Create Federated Credential for Workload Identity**
   Link the managed identity to the ALB Controller's Kubernetes service account via a federated credential.
   a. Retrieve the OIDC Issuer URL for your AKS cluster:

```
AKS_OIDC_ISSUER=$(az aks show -g $RESOURCE_GROUP -n $AKS_NAME --query "oidcIssuerProfile.issuerUrl" -o
 tsv)
```

   b. Create the federated credential:

```
az identity federated-credential create \
  --name alb-federated-credential \
  --identity-name $ALB_IDENTITY_NAME \
  -g $RESOURCE_GROUP \
  --issuer "$AKS_OIDC_ISSUER" \
  --subject system:serviceaccount:azure-alb-system:alb-controller-sa
```

6. **Create AGC Subnet**
   AGC requires a dedicated subnet with `Microsoft.ServiceNetworking/trafficControllers` delegation. The subnet must have at least 250 IPs (/24 or larger).
   a. Get the AKS node resource group and VNet name:

```
AKS_NODE_RG=$(az aks show -g $RESOURCE_GROUP -n $AKS_NAME --query nodeResourceGroup -o tsv)
AKS_VNET=$(az network vnet list -g $AKS_NODE_RG --query "[0].name" -o tsv)
```

b. Check if an AGC subnet already exists:

```
az network vnet subnet list -g $AKS_NODE_RG --vnet-name $AKS_VNET \
  --query "[?delegations[0].serviceName=='Microsoft.ServiceNetworking/trafficControllers'].name" -o tsv
```

c. If no subnet exists, create one (adjust the address prefix to avoid conflicts with existing subnets):

```
az network vnet subnet create \
  -g $AKS_NODE_RG \
  --vnet-name $AKS_VNET \
  -n agc-subnet \
  --address-prefixes 10.225.0.0/24 \
  --delegations Microsoft.ServiceNetworking/trafficControllers
```

d. Get the AGC subnet ID (replace `<agc-subnet-name>` with your subnet name):

```
AGC_SUBNET_ID=$(az network vnet subnet show -g $AKS_NODE_RG --vnet-name $AKS_VNET \
  -n <agc-subnet-name> --query id -o tsv)
echo "AGC Subnet ID: $AGC_SUBNET_ID"
```

7. **Register Microsoft.ServiceNetworking Provider**
   The `Microsoft.ServiceNetworking` resource provider is required at the subscription level. Without it, the ALB controller logs show `403` for `Microsoft.ServiceNetworking/register/action`.
   a. Check the registration status:

```
az provider show --namespace Microsoft.ServiceNetworking --query registrationState -o tsv
```

   b. If the status is `NotRegistered`, register it (takes 1-2 minutes):

```
az provider register --namespace Microsoft.ServiceNetworking
```

   c. Wait and verify registration is complete:

```
az provider show --namespace Microsoft.ServiceNetworking --query registrationState -o tsv
# Expected: Registered
```

8. **Assign RBAC Roles to ALB Identity**

   IMPORTANT
   This step requires **Owner** or **User Access Administrator** permissions. If you do not have these permissions, ask your Azure admin to run these commands.

   Three roles are needed: **Reader** (MC resource group), **AppGw for Containers Configuration Manager** (MC resource group), and **Network Contributor** (AGC subnet).
   a. Get the subscription ID:

```
SUBSCRIPTION_ID=$(az account show --query id -o tsv)
```

   b. Assign **Role 1: Reader** on the MC resource group (required for `Microsoft.Resources/subscriptions/resourcegroups/read`, which is NOT included in the Configuration Manager role):

```
az role assignment create \
  --assignee-object-id $ALB_IDENTITY_PRINCIPAL_ID \
  --assignee-principal-type ServicePrincipal \
  --role "Reader" \
  --scope /subscriptions/$SUBSCRIPTION_ID/resourceGroups/$AKS_NODE_RG
```

c. Assign **Role 2: AppGw for Containers Configuration Manager** on the MC resource group.

> **IMPORTANT**
> **Critical:** Scope this role to the **MC resource group**, not the AGC subnet. Scoping to the subnet causes `403 AuthorizationFailed` when the controller tries to create `Microsoft.ServiceNetworking/trafficControllers` resources. The error appears in the ApplicationLoadBalancer CR status, not in logs.

```
az role assignment create \
  --assignee-object-id $ALB_IDENTITY_PRINCIPAL_ID \
  --assignee-principal-type ServicePrincipal \
  --role "AppGw for Containers Configuration Manager" \
  --scope /subscriptions/$SUBSCRIPTION_ID/resourceGroups/$AKS_NODE_RG
```

d. Assign **Role 3: Network Contributor** on the AGC subnet (for standard Azure CNI flat networking):

```
az role assignment create \
  --assignee-object-id $ALB_IDENTITY_PRINCIPAL_ID \
  --assignee-principal-type ServicePrincipal \
  --role "Network Contributor" \
  --scope $AGC_SUBNET_ID
```

e. Verify all three roles are assigned:

```
az role assignment list --assignee $ALB_IDENTITY_PRINCIPAL_ID --all \
  --query "[].{role:roleDefinitionName, scope:scope}" -o table
# Expected: Reader, AppGw for Containers Configuration Manager, Network Contributor
```

9. **Install ALB Controller via Helm**

a. Install the ALB Controller using the Microsoft container registry Helm chart:

> **NOTE**
> Use v1.9.13 or later. Versions older than v1.7.9 do not support Azure CNI Overlay networking.

```
helm install alb-controller oci://mcr.microsoft.com/application-lb/charts/alb-controller \
  --namespace azure-alb-system --create-namespace \
  --version 1.9.13 \
  --set albController.podIdentity.clientID=$ALB_IDENTITY_CLIENT_ID
```

b. Verify ALB Controller pods are running:

```
kubectl get pods -n azure-alb-system
```

c. Verify GatewayClass is created and accepted:

```
kubectl get gatewayclass
# Expected:
```

```
# NAME                 CONTROLLER                                ACCEPTED
# azure-alb-external   alb.networking.azure.io/alb-controller   True
```

10. **Create ApplicationLoadBalancer Resource**
    This creates the AGC instance in Azure. RBAC roles (Step 8) must be assigned before this step.
    a. If you previously deleted an ApplicationLoadBalancer CR, verify the Azure resource is fully deleted before proceeding:

CAUTION
The Kubernetes CR status can show `Deployment: True` while the Azure resource is still stuck in `provisioningState: Deleting`. Recreating the CR while the Azure resource is still deleting causes a failure loop: the controller tries to create child resources (frontends) under a deleting parent, Azure rejects with "PUT is not allowed as previous operation on parent resource was DELETE", and the Gateway stays in Unknown state.

```
az network alb show --resource-group $AKS_NODE_RG --name <alb-name>
# Must return "ResourceNotFound" before proceeding.
# If you don't know the ALB name, list all traffic controllers:
az network alb list --resource-group $AKS_NODE_RG -o table
```

    b. Create the ApplicationLoadBalancer resource:

```
cat <<EOF | kubectl apply -f -
apiVersion: alb.networking.azure.io/v1
kind: ApplicationLoadBalancer
metadata:
  name: idsp-alb
  namespace: azure-alb-system
spec:
  associations:
    - $AGC_SUBNET_ID
EOF
```

    c. Wait for AGC to be provisioned (3-5 minutes):

```
kubectl get applicationloadbalancer -n azure-alb-system -w
```

    d. If the `DEPLOYMENT` column stays empty, check controller logs for RBAC errors:

```
kubectl logs -n azure-alb-system deployment/alb-controller --tail=20
```

11. **Verify Setup Is Complete**
    Run the following commands to confirm all infrastructure components are ready.
    a. Confirm GatewayClass is accepted:

```
kubectl get gatewayclass
```

    b. Confirm ALB Controller pods are running:

```
kubectl get pods -n azure-alb-system
```

c. Confirm ApplicationLoadBalancer is provisioned:

```
kubectl get applicationloadbalancer -n azure-alb-system
```

Azure Specific CRD's
Review the Azure-specific Custom Resource Definitions (CRDs), including their associated API groups, purposes, and creation methods.

Overview

The following table outlines the Azure-specific CRDs used in the configuration:

CRD	API Group	Purpose	Create
`ApplicationLoadBalancer`	`alb.networking.azure.io/v1`	AGC instance in Azure	Manu
`BackendTLSPolicy`	`alb.networking.azure.io/v1`	Azure-specific backend TLS config	Helm
`HealthCheckPolicy`	`alb.networking.azure.io/v1`	HTTPS health check for ssp-router backend	Helm
`GatewayClass`	`gateway.networking.k8s.io/v1`	`azure-alb-external` / `azure-alb-internal`	ALB C

NOTE
Note: The `ApplicationLoadBalancer` resource is created manually outside the Helm chart, as it contains environment-specific settings (subnet associations). The ALB Controller automatically creates the GatewayClass resources. The `HealthCheckPolicy` is required because AGC's default health check hits `/` which returns 404 on ssp-router.

Deploying or Migrating to Azure Gateway API
Learn to deploy a fresh instance of IDSP or migrate an existing configuration of IDSP to use the Azure Gateway API.

Introduction

This topic outlines the necessary steps to successfully deploy or migrate to the Azure Gateway API. It covers strict pre-deployment checks, fresh installation procedures, Nginx to Gateway API migration logic, and a sample app validation.

Pre-requisite Check

NOTE
Before proceeding with the specific checks mentioned in this section, review and complete all items in the Prerequisites for All Deployments list.

Run these commands to verify all prerequisites are in place before deploying IDSP:

- **ALB Controller pods running:**
  `kubectl get pods -n azure-alb-system`
- **GatewayClass available and accepted** (Expected: `azure-alb-external` with `Accepted=True`):
  `kubectl get gatewayclass`
- **ApplicationLoadBalancer provisioned** (`DEPLOYMENT` must be `True` . If `False` , check controller logs for RBAC errors - Gotcha #16):
  `kubectl get applicationloadbalancer -n azure-alb-system`
- **OverlayExtensionConfig exists in correct namespace** (CNI Overlay clusters only. Expected: one entry in `azure-alb-system` with `Succeeded` status):

```
kubectl get overlayextensionconfigs -A
```

- **Microsoft.ServiceNetworking provider registered** (Expected: `Registered`):
  ```
  az provider show --namespace Microsoft.ServiceNetworking --query registrationState -o tsv
  ```
- **RBAC roles assigned** (3 roles needed. Expected: `Reader`, `AppGw for Containers Configuration Manager`, `Network Contributor`):
  ```
  az role assignment list --assignee <ALB_IDENTITY_PRINCIPAL_ID> --all \
    --query "[].{role:roleDefinitionName, scope:scope}" -o table
  ```
- **TLS certificate Secret exists:**
  ```
  kubectl get secret <tls-secret-name> -n ssp
  ```
- **RBAC permissions:**
  ```
  kubectl auth can-i create gateways.gateway.networking.k8s.io -n ssp
  kubectl auth can-i create httproutes.gateway.networking.k8s.io -n ssp
  kubectl auth can-i create healthcheckpolicies.alb.networking.azure.io -n ssp
  kubectl auth can-i create backendtlspolicies.alb.networking.azure.io -n ssp
  ```

Minimum ClusterRole for Azure AGC:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: gateway-api-admin
rules:
  - apiGroups: ["gateway.networking.k8s.io"]
    resources: ["gateways", "httproutes", "backendtlspolicies"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
  - apiGroups: ["alb.networking.azure.io"]
    resources: ["healthcheckpolicies", "backendtlspolicies"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

Before You Deploy

NOTE

- Gateway ownership: The Gateway can be created by Helm (`createGateway: true`, default) or pre-created by the platform team (`createGateway: false` + `existingGateway`). To know how to use an already existing Gateway, see Already Have a Gateway? section for details.
- Set these Environment Variables:
  ```
  export RELEASENAME="<SSP_RELEASENAME>"
  export NAMESPACE="<SSP_NAMESPACE>"
  export HELM_REPO='ssp_helm_charts'
  ```
- Run the following command to export the current override values of your Helm release to `ssp-override.yaml`. This lets you review, back up, or reuse your existing configuration:
  ```
  helm get values ${RELEASENAME} -n ${NAMESPACE} >ssp-override.yaml
  ```

Deploying Symantec Identity Security Platform with Gateway API (Fresh Install)

1. **Ensure Azure ALB Controller is installed and GatewayClass exists.**
   See Infrastructure Setup.

2. **Deploy IDSP with Gateway API**
   a. Run the helm upgrade command with inline `--set` arguments (note: the escaped dots are required for the command line):

```
helm upgrade --install ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.gatewayApi.gatewayClassName=azure-alb-external \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.createBackendTLSPolicy=true \
  --set ssp.ingress.gatewayApi.gatewayAnnotations."alb\.networking\.azure\.io/alb-name"=<alb-name> \
  --set ssp.ingress.gatewayApi.gatewayAnnotations."alb\.networking\.azure\.io/alb-namespace"=azure-alb-
system
```

   b. Alternatively, `gatewayAnnotations` can be set via a values override file (recommended for readability). Override file example (`gateway-overrides.yaml`):

```
ssp:
  ingress:
    type: gatewayapi
    gatewayApi:
      gatewayClassName: azure-alb-external
      createGateway: true
      gatewayAnnotations:
        alb.networking.azure.io/alb-name: <alb-name>
        alb.networking.azure.io/alb-namespace: azure-alb-system
```

3. **Verify resources**

```
kubectl get gateway,httproute,backendtlspolicy,healthcheckpolicy -n ssp
```

4. **Wait for Gateway to be Programmed**
   AGC takes 3-5 minutes on first deploy. Wait until STATUS shows `Programmed=True` and ADDRESS shows an FQDN.

```
kubectl get gateway -n ssp -w
```

5. **Verify all policies are accepted**

```
kubectl get backendtlspolicy -n ssp -o jsonpath='{range .items[*]}{.metadata.name}:
 Accepted={.status.conditions[?(@.type=="Accepted")].status}{"\n"}{end}'
kubectl get healthcheckpolicy -n ssp -o jsonpath='{range .items[*]}{.metadata.name}:
 Accepted={.status.conditions[?(@.type=="Accepted")].status}{"\n"}{end}'
```

   Expected: `Accepted=True` for both.
6. **Test connectivity**
   AGC returns FQDN, not IP — you must resolve it first.

```
GATEWAY_FQDN=$(kubectl get gateway -n ssp -o jsonpath='{.items[0].status.addresses[0].value}')
GATEWAY_IP=$(dig +short $GATEWAY_FQDN | head -1)
curl -sk --resolve "your-host:443:$GATEWAY_IP" \
```

```
"https://your-host/actuator/health"
```

What Helm creates for Azure AGC:

Resource	Name Pattern	Purpose
Gateway	`<release>-ssp-gateway`	HTTPS listeners with TLS termination
HTTPRoute	`<release>-ssp-router-full`	Routes all traffic to ssp-router
HTTPRoute	`<release>-ssp-router-mgmt`	Management hostname route (if mgmt enab
BackendTLSPolicy	`<release>-ssp-router-backend-tls`	Azure-specific CA validation for HTTPS to s
HealthCheckPolicy	`<release>-ssp-router-health-check`	HTTPS health check (required — AGC defa

Migrating Symantec Identity Security Platform from Nginx to Gateway API

NOTE
Before proceeding, complete all steps in the " Before you Deploy" section.

1. If upgrading an existing nginx-based Symantec Identity Security Platform deployment to Gateway API on Azure, run the following:

```
helm upgrade ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.previousController.enabled=true \
  --set ssp.ingress.previousController.ingressClassName=nginx \
  --set ssp.ingress.gatewayApi.gatewayClassName=azure-alb-external \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.createBackendTLSPolicy=true \
  --set ssp.ingress.gatewayApi.gatewayAnnotations."alb\.networking\.azure\.io/alb-name"=<alb-name> \
  --set ssp.ingress.gatewayApi.gatewayAnnotations."AZ_ALB}alb\.networking\.azure\.io/alb-namespace"=azure-
alb-system
```

2. After verifying Gateway API works and switching DNS, remove the old controller:

```
helm upgrade ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.previousController.enabled=false \
  --set ssp.ingress.gatewayApi.gatewayClassName=azure-alb-external \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.createBackendTLSPolicy=true \
  --set ssp.ingress.gatewayApi.gatewayAnnotations."alb\.networking\.azure\.io/alb-name"=<alb-name> \
  --set ssp.ingress.gatewayApi.gatewayAnnotations."alb\.networking\.azure\.io/alb-namespace"=azure-alb-
system
```

Deploying the Sample App to Test Connectivity

1. Deploy the sample-app with Azure AGC to test connectivity before or alongside Symantec Identity Security Platform:

```
helm upgrade --install ${RELEASENAME} ${HELM_REPO}/ssp-sample-app -n ${NAMESPACE} \  -f your-sample-
values.yaml \
  --set ingress.type=gatewayapi \
  --set ingress.gatewayApi.gatewayClassName=azure-alb-external \
  --set ingress.gatewayApi.createGateway=true \
  --set ingress.gatewayApi.createBackendTLSPolicy=true \
  --set ingress.gatewayApi.gatewayAnnotations."alb\.networking\.azure\.io/alb-name"=<alb-name> \
  --set ingress.gatewayApi.gatewayAnnotations."alb\.networking\.azure\.io/alb-namespace"=azure-alb-system

# Verify resources
kubectl get gateway,httproute,backendtlspolicy,healthcheckpolicy -n
${NAMESPACE}
```

NOTE
Note: The chart creates a HealthCheckPolicy for spi-demo (which returns 302 on `/`) pointing to `/actuator/health` .

Troubleshooting Azure Deployments

General Diagnostics

If you encounter issues during your Azure AGC deployment, use these diagnostic commands to check the state of your resources:

ALB Controller not creating resources:

```
kubectl logs -n azure-alb-system deployment/alb-controller --tail=50
```

Check Gateway status:

```
kubectl get gateway -n ssp -o yaml
kubectl describe gateway <gateway-name> -n ssp
```

Verify BackendTLSPolicy and HealthCheckPolicy:

```
kubectl get backendtlspolicy -n ssp -o yaml
kubectl get healthcheckpolicy -n ssp -o yaml
```

Testing connectivity with curl:

AGC requires proper TLS SNI. Use `--resolve` (not `-H Host:`):

```
# Get Gateway FQDN and resolve to IP
GATEWAY_FQDN=$(kubectl get gateway -n ssp -o jsonpath='{.items[0].status.addresses[0].value}')
GATEWAY_IP=$(dig +short $GATEWAY_FQDN | head -1)

# CORRECT - uses --resolve for proper SNI:
curl -sk --resolve "your-host:443:$GATEWAY_IP" \
  "https://your-host/actuator/health"

# WRONG - -H Host: does NOT set TLS SNI, AGC resets the connection:
curl -sk -H "Host: your-host" "https://$GATEWAY_IP/actuator/health"
```

Cause: 403 AuthorizationFailed / Missing Reader Role

The ALB controller logs report a `403 AuthorizationFailed` specifically for `Microsoft.Resources/subscriptions/resourcegroups/read`.

Remedy

Cluster Administrator

1. Ensure the ALB Managed Identity has the **Reader** role assigned to the AKS node resource group (MC resource group).
2. Ensure the user running the RBAC commands has Owner or User Access Administrator permissions.
3. Restart the ALB controller to pick up the cached RBAC changes: `kubectl rollout restart deployment alb-controller -n azure-alb-system`

Cause: 403 AuthorizationFailed / ServiceNetworking Registration

The ALB controller logs report a `403 AuthorizationFailed` specifically for `Microsoft.ServiceNetworking/register/action`.

Remedy

Cluster Administrator

1. Run the following command to register the required provider: `az provider register --namespace Microsoft.ServiceNetworking`
2. Restart the ALB controller: `kubectl rollout restart deployment alb-controller -n azure-alb-system`

Cause: 403 AuthorizationFailed / Traffic Controller Not Provisioning

Running `kubectl get applicationloadbalancer -o yaml` shows an `AuthorizationFailed` error, and the Traffic Controller fails to provision.

Remedy

Cluster Administrator

Verify that the **AppGw for Containers Configuration Manager** role is scoped correctly to the **MC resource group**, and NOT the AGC subnet.

Cause: 503 "no healthy upstream" / Stale AGC Data Plane State

Curl returns a 503 error. All backends show `UNHEALTHY` but pods are running, responsive, and policies are `Accepted=True`.

Remedy

Cluster Administrator

1. Delete all AGC objects (HTTPRoute, BackendTLSPolicy, HealthCheckPolicy) to clear the stale data plane state.
2. Wait 5 minutes for full deprovisioning and verify entries are gone from the API.
3. Recreate the objects from scratch.

Cause: 503 "no healthy upstream" / BackendTLSPolicy Port Mismatch

Curl returns a 503 error. All backends show `UNHEALTHY` even after a clean recreate of AGC objects (on either flat CNI or CNI Overlay).

Remedy

Cluster Administrator

1. Check the `port` configured in your BackendTLSPolicy. It must be set to the service port (`443`), not the container port (e.g., `8095`).
2. Verify the policy has `verify.caCertificateRef` pointing to the correct CA secret if using self-signed certificates.

Cause: 503 "no healthy upstream" / BackendTLSPolicy Name Collision

Curl returns a 503 error, but all resources and policies read as `Accepted=True` .

Remedy

Cluster Administrator

1. Run `kubectl get backendtlspolicies.alb.networking.azure.io -A` and look for duplicate names across namespaces.
2. If duplicates exist, AGC uses the wrong CA certificate. Fix by ensuring your BackendTLSPolicy names include the namespace to make them unique.

Cause: HealthCheckPolicy Rejected

Running `kubectl describe healthcheckpolicy -n <ns>` shows the policy state as `Accepted=False` .

Remedy

Cluster Administrator

Check the protocol schema in your HealthCheckPolicy yaml. You must use `http` + `useTLS: true` instead of `https` .

Cause: Gateway Not Accepted / Missing Annotations

Running `kubectl describe gateway` reports that the Gateway "does not have annotation".

Remedy

Cluster Administrator

Ensure you have added the `alb.networking.azure.io/alb-name` and `alb.networking.azure.io/alb-namespace` annotations to the Gateway via the `gatewayAnnotations` block in your Helm values.

Cause: OverlayExtensionConfig Issues

The `OverlayExtensionConfig` is not created, logs report `nodenetworkconfigs` watch errors, `overlayextensionconfigs.acn.azure.com is forbidden`, or `extension IP range overlaps with existing`.

Remedy

Cluster Administrator

1. Verify your ALB controller is version ≥1.7.9. Upgrade if older.
2. Run `kubectl get clusterrole alb-controller-cluster-role -o yaml` and ensure the `watch` verb exists for `nodenetworkconfigs.acn.azure.com`.
3. Check the ClusterRoleBinding target namespace: `kubectl get clusterrolebinding alb-controller-cluster-role-binding -o yaml | grep namespace`. If it points to `kube-system` but the controller runs in `azure-alb-system`, create a supplementary binding.
4. If overlapping IP ranges are reported, check for stale configurations across namespaces using `kubectl get overlayextensionconfigs -A`. Delete the stale config, delete the failed one in `azure-alb-system`, and restart the controller.

Cause: Stuck Traffic Controller / Gateway Unknown Status

The Gateway is stuck in `Unknown` status, or ALB controller logs report "Failed to parse Frontend" / "invalid frontend" after recreating an ALB CR. `az network alb show` indicates the Azure resource is stuck in `Deleting` state.

Remedy

Cluster Administrator

1. Remember that ALB CR status can be stale—always verify actual Azure ARM resource state.
2. Wait for the Traffic Controller to fully finish deleting and return a `ResourceNotFound` response via the Azure CLI.
3. Once fully deleted, recreate the ALB CR in Kubernetes.
4. Restart the ALB controller.

Cause: ALB Controller CrashLoopBackOff

The ALB controller pods fail to start or crash repeatedly.

Remedy

Cluster Administrator

Verify that the federated credential matches the ALB Controller service account name and namespace exactly.

Cause: AGC Subnet Creation Fails

Attempts to create the `agc-subnet` using the Azure CLI fail.

Remedy

Cluster Administrator

1. Run `az network vnet subnet list -g $AKS_NODE_RG --vnet-name $AKS_VNET` to review the existing subnets.
2. Check for address space conflicts and adjust the `--address-prefixes` parameter in your create command to use a non-overlapping CIDR block.

AVI AKO Gateway API Controller

Review the features, requirements, and architecture mapping of the Avi AKO Gateway API Controller.

Overview

Avi AKO is Broadcom's Kubernetes operator for Avi (NSX Advanced Load Balancer). It implements Gateway API by translating Gateway API resources into Avi Controller objects (Virtual Services, Pools, Pool Groups, SSL profiles).

> **NOTE**
> Avi AKO is the **default and recommended** Gateway API controller for **VKS (VMware Kubernetes Service)**.

Aspect	Details
Gateway API support	GatewayClass (v1), Gateway (v1), HTTPRoute (v1). Supported from AKO 1.12.1+. Enable via `featureGates.GatewayAPI: true` in values.yaml.
Default GatewayClass	`avi-lb` with `controllerName: ako.vmware.com/avi-lb` (installed when Gateway API is enabled).
Gateway	HTTP/HTTPS listeners; TLS termination with `certificateRefs` to Kubernetes Secrets; optional static IP via `spec.addresses` (single IPv4).
HTTPRoute	Hostname, path (including `PathPrefix` , `RegularExpression`), and header matching; traffic splitting (canary/blue-green).
HTTPRoute filters	RequestHeaderModifier, RequestRedirect, URLRewrite (path/hostname), ResponseHeaderModifier. `ExtensionRef` filters under `backendRefs` are supported for `RouteBackendExtension` .
Backend TLS	Backend HTTPS and certificate validation are configured via Avi's `RouteBackendExtension` CRD (referenced as an `ExtensionRef` filter in HTTPRoute `backendRefs`), not the standard BackendTLSPolicy CRD. Requires the AKO CRD Operator to be enabled.
Platforms	Kubernetes on VMware vSphere/NSX-T; supports ClusterIP, NodePort, NodePortLocal; CNI: Calico, Antrea.
Limitation	AKO High Availability (HA) is not supported when Gateway API is enabled. GRPCRoute, TLSRoute, TCPRoute, UDPRoute, and ReferenceGrant are not supported.

For Official documentation, see Avi Kubernetes Operator – Gateway API v1.

Avi Architecture: How Gateway API Maps to Avi Objects

```
Gateway (K8s)
  └── Parent VS (Avi) — VS_TYPE_VH_PARENT
        ├── VSVIP — holds the VIP (e.g., 172.30.128.214)
        ├── SSL Certs — from Gateway listener TLS secretRef
        ├── Port 443, enable_ssl=true
        └── Child VS entries (one per HTTPRoute)
              ├── Child VS 1 — VS_TYPE_VH_CHILD, vh_type=VS_TYPE_VH_ENHANCED
```

```
│     ├── vh_matches: [{host: "app.example.com", rules: [...]}]
│     └── pool_group_ref → Pool → Servers (NPL node:port pairs)
└── Child VS 2 — VS_TYPE_VH_CHILD
      └── ...
```

Key points:

- One parent VS per Gateway, one child VS per HTTPRoute
- Parent handles SSL termination; children handle host/path matching and backend routing
- AKO uses **EVH mode** for Gateway API regardless of the `enableEVH` AKO config setting
- Child VS uses `vh_matches` (not `vh_domain_name`) — this is the EVH-specific field
- With NodePortLocal (NPL), pool servers are `node_IP:NPL_port` pairs, not pod IPs

Infrastructure Setup for AVI ACO Gateway API Controller
Review the configuration steps required to enable and set up Gateway API support for the Avi Kubernetes Operator (AKO), including feature gate configuration, CRD installation, and ConfigMap verification.

Introduction

Configure the necessary infrastructure to support the AVI AKO Gateway API Controller. The following steps cover enabling Gateway API support in your Helm values, installing the required RouteBackendExtension Custom Resource Definition (CRD) for backend TLS, and verifying the final ConfigMap settings.

Setup Steps

1. **Enable Gateway API Support**
   To enable Gateway API support in AKO, set `featureGates.GatewayAPI: true` in the AKO Helm values:

   ```
   # AKO Helm values
   featureGates:
     GatewayAPI: true
   ```

   This automatically creates the `avi-lb` GatewayClass:

   ```
   apiVersion: gateway.networking.k8s.io/v1
   kind: GatewayClass
   metadata:
     name: avi-lb
   spec:
     controllerName: ako.vmware.com/avi-lb
   ```

2. **Install RouteBackendExtension CRD (required for backend TLS)**
   If you need backend TLS re-encryption (i.e., `createBackendTLSPolicy: true`), the `RouteBackendExtension` CRD must be installed. This CRD is part of the AKO CRD Operator and is **not** included in the base AKO Helm chart.

   WARNING
   If the CRD is not installed, the IDSP Helm chart silently skips creating the `RouteBackendExtension` resource and the corresponding `ExtensionRef` filter in the HTTPRoute. **ssp-router serves HTTPS only** — without backend TLS, the Avi pool connects with plain HTTP, ssp-router rejects the connection, and all requests return **503**. This is the most common cause of 503 errors with AKO Gateway API. The RouteBackendExtension CRD is effectively **required**, not optional.

   a. To install, deploy the AKO CRD Operator Helm chart (available from the same AKO distribution):

   ```
   helm install ako-operator oci://projects.packages.broadcom.com/ako/helm-charts/ako-operator \
     -n avi-system --version <ako-version>
   ```

b. Alternatively, apply the CRD YAML directly from the AKO release artifacts:

```
kubectl apply -f https://raw.githubusercontent.com/vmware/load-balancer-and-ingress-services-for-kubernetes/master/ako-crd-operator/config/crd/bases/ako.vmware.com_routebackendextensions.yaml
```

c. Verify the CRD is installed:

```
kubectl get crd routebackendextensions.ako.vmware.com
```

3. **Verify AKO ConfigMap settings**
(especially on VKS with Antrea CNI):

```
kubectl get configmap avi-k8s-config -n avi-system -o yaml | grep -E "serviceType|cniPlugin|enableEVH"
# Expected for VKS: serviceType=NodePortLocal, cniPlugin=antrea
```

Deploying or Migrating to AVI ACO Gateway API

Review the deployment and migration steps for the AVI AKO Gateway API, including prerequisite validation, fresh installations, Nginx migrations, and sample app connectivity testing.

Introduction

Deploy or migrate to the AVI AKO Gateway API using the end-to-end process outlined below. The following sections cover essential prerequisite validations, configuration steps for both fresh installs and Nginx migrations, and a sample application deployment to verify connectivity.

Pre-requisite Check

Execute the following steps to verify that your cluster and resources are properly configured before attempting a deployment.

1. Check that the AKO pods are running:

```
# AKO pods running
kubectl get pods -n avi-system -l app.kubernetes.io/name=ako
```

2. Verify the GatewayClass is available and accepted:

```
# GatewayClass available and accepted
kubectl get gatewayclass avi-lb
```

3. Ensure the TLS certificate Secret exists:

```
# TLS certificate Secret exists
kubectl get secret <tls-secret-name> -n ssp
```

4. Confirm the RouteBackendExtension CRD is installed (required because ssp-router is HTTPS-only):

```
# RouteBackendExtension CRD installed (required — ssp-router is HTTPS-only)
kubectl get crd routebackendextensions.ako.vmware.com
```

5. After deployment, verify backend TLS is active:

```
# Verify RouteBackendExtension has backendTLS (not null/missing)
kubectl get routebackendextension -n ssp
kubectl get routebackendextension <release>-ssp-router-backend-ext -n ssp -o jsonpath='{.spec.backendTLS}'
# Expected: {} or {"hostCheckEnabled":...}
# If empty/missing: ssp-router returns 503
```

6. Ensure your ClusterRole meets the minimum requirements for Avi AKO:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: gateway-api-admin
rules:
  - apiGroups: ["gateway.networking.k8s.io"]
    resources: ["gateways", "httproutes"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
  - apiGroups: ["ako.vmware.com"]
    resources: ["routebackendextensions"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

Before You Deploy

Review and configure the following environmental considerations before executing the IDSP Helm installation.

NOTE

- Gateway ownership: The Gateway can be created by Helm (`createGateway: true` , default) or pre-created by the platform team (`createGateway: false` + `existingGateway`). To know how to use an already existing Gateway, see Already Have a Gateway? section for details.
- Set these Environment Variables:

```
export RELEASENAME="<SSP_RELEASENAME>"
export NAMESPACE="<SSP_NAMESPACE>"
export HELM_REPO='ssp_helm_charts'
```

- Run the following command to export the current override values of your Helm release to `ssp-override.yaml` . This lets you review, back up, or reuse your existing configuration:

```
helm get values ${RELEASENAME} -n ${NAMESPACE} >ssp-override.yaml
```

Deploying Symantec Identity Security Platform with Gateway API (Fresh Install)

Perform the following steps to deploy a fresh instance of Symantec Identity Security Platform using the Gateway API with Avi.

1. Run the Helm upgrade command to deploy the application:

```
helm upgrade --install ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.gatewayApi.gatewayClassName=avi-lb \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.gatewayType=avi
  # createBackendTLSPolicy defaults to true — creates RouteBackendExtension for Avi
```

2. Verify the Gateway and HTTPRoute resources were successfully created:

```
# Verify resources
kubectl get gateway,httproute -n ssp
```

Migrating Symantec Identity Security Platform from Nginx to Gateway API

Follow these steps if you are upgrading an existing nginx-based <keyword keyref="pnameShort"/> deployment to the Gateway API using Avi AKO.

1. Upgrade the deployment while leaving the previous Nginx controller enabled:

```
helm upgrade ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.previousController.enabled=true \
  --set ssp.ingress.previousController.ingressClassName=nginx \
  --set ssp.ingress.gatewayApi.gatewayClassName=avi-lb \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.gatewayType=avi
```

2. After verifying Gateway API works and switching DNS, run the upgrade command again to remove the old controller:

```
helm upgrade ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.previousController.enabled=false \
  --set ssp.ingress.gatewayApi.gatewayClassName=avi-lb \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.gatewayType=avi
```

Deploying the Sample App to Test Connectivity

Perform these steps to deploy a sample application with Avi AKO to test connectivity before or alongside IDSP.

1. Install the sample application using Helm:

```
helm upgrade --install ${RELEASENAME} ${HELM_REPO}/ssp-sample-app -n ${NAMESPACE} \
  -f your-sample-values.yaml \
  --set ingress.type=gatewayapi \
  --set ingress.gatewayApi.gatewayClassName=avi-lb \
  --set ingress.gatewayApi.createGateway=true \
  --set ingress.gatewayApi.gatewayType=avi
```

2. Verify the created resources:

```
# Verify resources
kubectl get gateway,httproute -n ${NAMESPACE}
```

3. Retrieve the Gateway VIP and test connectivity:

```
# Get Gateway VIP and test
GATEWAY_VIP=$(kubectl get gateway -n ${NAMESPACE} -o jsonpath='{.items[0].status.addresses[0].value}')
curl -sk --resolve "your-host:443:$GATEWAY_VIP" "https://your-host/sample-app/"
```

NOTE
Note: Avi AKO does not use BackendTLSPolicy or HealthCheckPolicy CRDs. spi-demo health checks use Avi's default TCP health monitor.

Backend TLS with AVI ACO
Review the configuration, verification, and troubleshooting steps for setting up backend TLS with the Avi Kubernetes Operator (AKO) using the RouteBackendExtension CRD.

Introduction

Review the concepts and prerequisites for configuring backend TLS with Avi AKO.

Avi AKO does **not** use the standard BackendTLSPolicy CRD. Instead, backend TLS is configured via Avi's **RouteBackendExtension** CRD, which is attached to the HTTPRoute as an `ExtensionRef` filter on the `backendRefs` entry.

Pre-requisite: The `RouteBackendExtension` CRD must be installed via the AKO CRD Operator (see Section 5.2 for installation instructions).

The Helm chart automatically creates the `RouteBackendExtension` resource and references it from the HTTPRoute when `createBackendTLSPolicy: true` and `gatewayType: avi` . The chart gates this on CRD availability — if the CRD is not installed, the resource and the ExtensionRef filter are silently skipped.

> **NOTE**
> **Critical:** The `backendTLS` field in the RouteBackendExtension **must** be present (even as empty `{}`). Without it, the Avi pool has no `ssl_profile_ref` and connects to ssp-router with plain HTTP. Since ssp-router is HTTPS-only, this results in **503 errors** — and is extremely hard to diagnose because the Gateway shows `Programmed=True` , the VS is `OPER_UP` , and the pool servers are healthy (TCP health checks pass, but application-layer communication fails). If you see 503 with everything else looking healthy, check the RouteBackendExtension first.

How backendTLS works — toggle by existence

Understand how the presence of the `backendTLS` field functions as a toggle to enable HTTPS to the backend.

The `backendTLS` field acts as a **toggle**: its *presence* enables HTTPS to the backend, not its contents. The optional sub-fields only refine TLS validation strictness:

`backendTLS` state	Avi pool behavior
Absent or `null`	No `ssl_profile_ref` → **plain HTTP** to backend → **503** (ssp-router is HTTPS-only)
`{}` (present, empty)	Adds `ssl_profile_ref: System-Standard` → **HTTPS** to backend (default validation)
`{pkiProfile: ...}`	HTTPS + CA certificate pinning / mutual TLS
`{hostCheckEnabled: true}`	HTTPS + SNI hostname verification against backend cert
`{domainName: [...]}`	HTTPS + expected SAN/CN matching

For most deployments, `backendTLS: {}` is sufficient — it enables re-encryption with Avi's default `System-Standard` SSL profile without imposing additional certificate validation requirements.

Resource Configurations

Review the YAML configurations created by Helm for the RouteBackendExtension and HTTPRoute.

RouteBackendExtension resource (created by Helm):

```
apiVersion: ako.vmware.com/v1alpha1
kind: RouteBackendExtension
metadata:
  name: <release>-ssp-router-backend-ext
  namespace: ssp
```

```
spec:
  backendTLS: {}   # presence enables HTTPS; sub-fields are optional refinements
```

HTTPRoute ExtensionRef filter (added by Helm to backendRefs):

```
backendRefs:
  - name: <release>-ssp-router
    port: 8095
    filters:
      - type: ExtensionRef
        extensionRef:
          group: ako.vmware.com
          kind: RouteBackendExtension
          name: <release>-ssp-router-backend-ext
```

For full details, see: AKO RouteBackendExtension

How to Verify Backend TLS is Working

Execute the following steps to confirm that backend TLS is properly enabled and functioning.

1. **Check RouteBackendExtension exists and has backendTLS:**

   ```
   kubectl get routebackendextension -n <namespace>
   kubectl get routebackendextension <name> -n <namespace> -o yaml
   # Verify: spec.backendTLS is present (not missing or null-without-fields)
   ```

2. **Check the Avi pool has ssl_profile_ref:**
   Log into Avi Controller → Applications → Pools → find the pool → check SSL Profile. Or verify via API:

   ```
   curl -sk -b cookies.txt "https://<avi-controller>/api/pool/<pool-uuid>?fields=name,ssl_profile_ref" \
     -H 'X-Avi-Version: 30.2.4'
   # ssl_profile_ref should be a non-empty URL
   ```

3. **Perform a quick backend protocol test (from inside a router pod):**

   ```
   kubectl exec -n <namespace> <router-pod> -- wget -qO- --no-check-certificate https://localhost:8095/
   actuator/health
   # Should return: {"status":"UP",...}

   kubectl exec -n <namespace> <router-pod> -- wget -qO- http://localhost:8095/actuator/health
   # Should fail with: "error getting response" (confirms HTTPS-only)
   ```

When backendTLS Can Go Missing

Identify scenarios where the `backendTLS` field may go missing and apply the necessary fixes.

There are three scenarios where the pool ends up without `ssl_profile_ref` :

1. **Old chart version** — the IDSP chart template was updated to include `backendTLS` in the RouteBackendExtension. Deployments from an older chart version have the RouteBackendExtension without `backendTLS` .

2. **AKO full rebuild** — when AKO recreates all Avi objects from scratch (after VS deletion + AKO restart), it reads the current K8s objects. If the RouteBackendExtension doesn't have `backendTLS` , the new pool won't have SSL. Previous pool SSL settings are NOT preserved across recreation.
3. **Manual RouteBackendExtension edits** — if someone patches the CRD and removes `backendTLS` , AKO updates the pool accordingly.

To resolve a missing `backendTLS` configuration, perform one of the following fixes:

1. **Emergency fix (without Helm upgrade):**
   Apply a merge patch to the resource. Note that `hostCheckEnabled:false` is just a carrier field — the merge-patch requires at least one key to create the backendTLS object. `backendTLS:{}` is what matters.

```
kubectl patch routebackendextension <name> -n <namespace> \
  --type=merge -p '{"spec":{"backendTLS":{"hostCheckEnabled":false}}}'
```

2. **Template fix (already applied):**
   The Helm chart template was fixed to render `backendTLS: {}` (empty object) instead of `backendTLS:` (YAML null) when no AVI-specific sub-fields are configured. YAML null is treated as "field not present" by the CRD, which means backend TLS is not enabled. The empty object `{}` correctly signals "enable backend TLS with defaults".

Troubleshooting AVI ACO Deployments

General Diagnostics and Maintenance

Use the following commands and procedures to check the general health of your Avi AKO deployment, force reconciliations, or enable debug logging.

Check AKO logs (AKO runs two containers — `ako` for Ingress and `ako-gateway-api` for Gateway API):

```
# Gateway API container logs (filter out NPL noise)
kubectl logs -n avi-system -l app.kubernetes.io/name=ako -c ako-gateway-api | grep -v "annotation not found"

# Main AKO container logs
kubectl logs -n avi-system -l app.kubernetes.io/name=ako -c ako
```

Verify GatewayClass:

```
kubectl get gatewayclass avi-lb
```

Check Gateway status:

```
kubectl get gateway -n ssp -o wide
kubectl describe gateway <gateway-name> -n ssp
```

Check HTTPRoute status:

```
kubectl get httproute -n ssp
kubectl describe httproute <name> -n ssp
```

Verify Virtual Service in Avi Controller:

- Log into Avi Controller UI
- Navigate to Applications > Virtual Services
- Verify the VS corresponding to your Gateway is created and healthy

Nudge AKO to re-reconcile a Gateway or HTTPRoute:

AKO sometimes caches stale state (e.g., after another Gateway is deleted, hostname conflict checks may not re-evaluate). To trigger re-processing without restarting AKO:

```
kubectl annotate gateway <name> -n <ns> ako.vmware.com/reconcile=$(date +%s) --overwrite
# Or for an HTTPRoute:
kubectl annotate httproute <name> -n <ns> ako.vmware.com/reconcile=$(date +%s) --overwrite
```

Enable debug logging:

```
# Set log level to DEBUG (shows AKO internal model and REST API payloads)
kubectl patch configmap avi-k8s-config -n avi-system \
  --type=merge -p '{"data":{"logLevel":"DEBUG"}}'

# Trigger reconciliation (e.g., delete and re-create an HTTPRoute)
kubectl delete httproute <name> -n <ns>
kubectl apply -f /tmp/httproute-backup.yaml

# Read debug logs
kubectl logs -n avi-system -l app.kubernetes.io/name=ako -c ako-gateway-api --tail=200

# IMPORTANT: Reset to WARN when done
kubectl patch configmap avi-k8s-config -n avi-system \
  --type=merge -p '{"data":{"logLevel":"WARN"}}'
```

In debug logs, look for:

- `AviVsNode` graph dumps — verify `VHMatches` has correct hostnames
- `REST API POST/PUT` payloads — verify `vh_matches` and `pool_group_ref`
- `servers for port` — verify NPL resolution found servers with correct ports

Cause: 503 errors with Gateway Programmed=True

This is most commonly caused by missing `backendTLS` on the RouteBackendExtension. The Avi pool connects to ssp-router with plain HTTP instead of HTTPS. Everything else looks healthy — the Gateway is Programmed, the VS is OPER_UP, the pool servers pass health checks (TCP-only).

Remedy

Cluster Administrator

1. Verify the RouteBackendExtension has backendTLS and patch it if it is missing.

   ```
   # 1. Verify RouteBackendExtension has backendTLS
   kubectl get routebackendextension <release>-ssp-router-backend-ext -n ssp -o jsonpath='{.spec.backendTLS}'
   # If empty/null: this is the cause. Fix with:
   # (hostCheckEnabled:false is just a carrier — any field works; backendTLS:{} is
   #  what matters, but kubectl merge-patch requires at least one key)
   kubectl patch routebackendextension <release>-ssp-router-backend-ext -n ssp \
     --type=merge -p '{"spec":{"backendTLS":{"hostCheckEnabled":false}}}'
   ```

2. Verify the Avi pool has `ssl_profile_ref` via the Avi Controller UI or API.

   ```
   # 2. Verify the Avi pool has ssl_profile_ref (via Avi Controller UI or API)
   # Navigate to: Applications > Pools > find the pool > check SSL Profile is set
   ```

Cause: "no servers for port" in AKO logs

NPL port resolution failure. The HTTPRoute references a service port that differs from the container port, and AKO can't find the NPL annotation.

Remedy

Cluster Administrator

1. Check which port the HTTPRoute uses.

```
# Check which port the HTTPRoute uses
kubectl get httproute <name> -n <ns> -o yaml | grep -A2 backendRefs
# Should show port: 8095 (not 443)
```

2. Ensure the HTTPRoute references the `https-direct` port (8095) where `port == targetPort` . The IDSP chart handles this automatically for `gatewayType: avi` .

Cause: "no listeners matching" in AKO logs

An HTTPRoute's hostname doesn't match any Gateway listener hostname.

Remedy

Cluster Administrator

Ensure the Gateway has a listener for every hostname used in HTTPRoutes. For shared gateways, add listeners via `additionalListeners` .

Cause: Gateway shows Programmed=True but no VIP

IPAM/DNS allocation failure. Avi IPAM might be exhausted or misconfigured.

Remedy

Cluster Administrator

Check the AKO gateway-api container logs to diagnose IPAM issues.

```
# Check AKO gateway-api container logs
kubectl logs -n avi-system -l app.kubernetes.io/name=ako -c ako-gateway-api | grep -i "ipam\|vsvip\|address"
```

Cause: Child VS has empty vh_matches

AKO cache corruption. The child VS was created from stale internal state.

Remedy

Cluster Administrator

Use the AKO Disaster Recovery procedure — delete the Gateway, clean up Avi, restart AKO, and recreate.

Known Issues in AVI ACO Deployments

Review known issues related to AVI AKO Gateway API deployments, including NPL resolution failures, missing CRDs, and cached hostname conflicts, along with their respective workarounds.

Introduction

Familiarize yourself with the common known issues and behaviors when deploying the AVI AKO Gateway API Controller.

NPL (NodePortLocal) with Gateway API

Understand the known issue regarding NodePortLocal resolution and how the Helm chart automatically mitigates it.

Issue: When AKO uses `AKOSettings.serviceType: NodePortLocal` (common on VKS with Antrea CNI), the Gateway API code path has a known issue where service port → targetPort resolution does not work correctly for NPL annotation lookup. If the service port differs from the container/target port (e.g., `port: 443, targetPort: 8095`), AKO logs `no servers for port` and creates an empty pool, resulting in 503 errors.

Workaround: This is handled automatically by the Helm chart — no manual configuration needed.

1. The ssp-router Service includes a second port where `port == targetPort` (e.g., `8095 → 8095`).
2. The HTTPRoute references this port when `gatewayType: avi` .

```
# ssp-router Service (automatic)
ports:
  - name: https          # Used by nginx Ingress
    port: 443
    targetPort: 8095
  - name: https-direct   # Used by AVI Gateway API (NPL workaround)
    port: 8095
    targetPort: 8095
```

RouteBackendExtension CRD not installed

Review the consequences of a missing RouteBackendExtension CRD and how to prevent it.

Issue: If the AKO CRD Operator is not enabled, the `RouteBackendExtension` CRD is not present in the cluster. The Helm chart detects this automatically and skips creating the CRD resource and the ExtensionRef filter. **This results in 503 errors** — ssp-router is HTTPS-only and rejects plain HTTP connections from Avi.

Workaround:

1. The RouteBackendExtension CRD must be installed before deploying IDSP with `gatewayType: avi` .

AKO Caches Hostname Conflicts Across Gateways

Learn how to force AKO to re-evaluate listener hostnames when conflicts are cached.

Issue: When two Gateways in the same namespace have overlapping listener hostnames, AKO rejects the duplicate listeners. If you later delete the conflicting Gateway, AKO does **not** automatically re-evaluate the rejected listeners.

Workaround:

1. You must nudge the Gateway to trigger a manual re-evaluation:

```
kubectl annotate gateway <name> -n <ns> ako.vmware.com/reconcile=$(date +%s) --overwrite
```

Envoy Gateway

Envoy Gateway provides a complete Gateway API implementation with full BackendTLSPolicy support and consistent behavior across all platforms.

Overview

Envoy Gateway supports GatewayClass, Gateway, HTTPRoute, and BackendTLSPolicy (all v1). It provides configurable buffers and header sizes via ClientTrafficPolicy, mTLS with XFCC forwarding, backend timeouts via BackendTrafficPolicy, and uses Kubernetes readiness probes for health checks. It runs on any Kubernetes cluster including GKE, EKS, AKS, and on-premises environments.

The following table summarizes the key aspects and supported capabilities of Envoy Gateway.

Aspect	Details
Gateway API support	GatewayClass (v1), Gateway (v1), HTTPRoute (v1), BackendTLSPolicy (v1)
GatewayClass	`envoy-gateway-class` with `controllerName: gateway.envoyproxy.io/gatewayclass-controller`
BackendTLSPolicy	Full support — standard `gateway.networking.k8s.io/v1` CRD with CA certificate validation. **Required** for backend HTTPS (`appProtocol` alone is not sufficient). Alternative: Envoy Gateway `Backend` resource with `tls` config.
Traffic Policy	`ClientTrafficPolicy` (buffer limit, header preservation, mTLS with XFCC) + `BackendTrafficPolicy` (timeouts)
Health Checks	Uses Kubernetes readiness probes by default (no extra CRD needed)
Platforms	Any Kubernetes cluster: GKE, EKS, AKS, on-premises

NOTE

Envoy Gateway version: Minimum **v1.6.0**. This is the first Envoy Gateway release to support the stable `gateway.networking.k8s.io/v1` BackendTLSPolicy schema. Earlier versions (v1.4.x, v1.5.x) were built against older Gateway API versions and only support `v1alpha3` for BackendTLSPolicy, which our Helm templates do not target. Get releases from: https://github.com/envoyproxy/gateway/releases

Infrastructure Setup for Envoy Gateway API Controller

Install the Gateway API CRDs, deploy the Envoy Gateway controller, and create the GatewayClass for your platform.

Complete the following steps to set up the Envoy Gateway infrastructure on your Kubernetes cluster.

Step 1: Install Gateway API CRDs

Install the standard Gateway API Custom Resource Definitions:

```
kubectl apply -f https://github.com/kubernetes-sigs/gateway-api/releases/download/v1.4.0/standard-install.yaml
```

Step 2: Install Envoy Gateway

Install the Envoy Gateway Helm chart for your platform.

GKE / AKS / On-premises:

```
helm install eg oci://docker.io/envoyproxy/gateway-helm --version v1.6.0 \
  -n envoy-gateway-system --create-namespace --skip-crds
```

EKS (internet-facing):

```
helm install eg oci://docker.io/envoyproxy/gateway-helm --version v1.6.0 \
  -n envoy-gateway-system --create-namespace \
  --set service.annotations.service\.beta\.kubernetes\.io/aws-load-balancer-scheme="internet-facing" \
  --set service.annotations.service\.beta\.kubernetes\.io/aws-load-balancer-type="external" \
  --set service.annotations.service\.beta\.kubernetes\.io/aws-load-balancer-nlb-target-type="ip" \
  --skip-crds
```

EKS (internal, behind DMZ):

```
helm install eg oci://docker.io/envoyproxy/gateway-helm --version v1.6.0 \
  -n envoy-gateway-system --create-namespace \
  --set service.annotations.service\.beta\.kubernetes\.io/aws-load-balancer-scheme="internal" \
  --set service.annotations.service\.beta\.kubernetes\.io/aws-load-balancer-type="external" \
  --set service.annotations.service\.beta\.kubernetes\.io/aws-load-balancer-nlb-target-type="ip" \
  --skip-crds
```

Step 3: Apply Oversized CRD (GKE and AKS)

The `envoyproxies` CRD exceeds the 262KB annotation size limit used by `kubectl apply` (client-side apply). Since Helm applies CRDs from the `crds/` directory using client-side apply, this CRD may fail to install or update. Apply it separately using server-side apply:

```
kubectl apply --server-side -f https://raw.githubusercontent.com/envoyproxy/gateway/v1.6.0/charts/gateway-helm/crds/generated/gateway.envoyproxy.io_envoyproxies.yaml
```

Run this **after** the `helm install` command above. If you are upgrading Envoy Gateway, run it after `helm upgrade` as well. The other Envoy Gateway CRDs are small enough to work with standard client-side apply.

Step 4: Create GatewayClass

Envoy Gateway's Helm chart may auto-create a GatewayClass named `eg` . Verify with `kubectl get gatewayclass` . If no GatewayClass exists, or you want a custom name, create one:

```
apiVersion: gateway.networking.k8s.io/v1
kind: GatewayClass
metadata:
  name: envoy-gateway-class
spec:
  controllerName: gateway.envoyproxy.io/gatewayclass-controller
```

Deploying or Migrating to Envoy Gateway API Controller

Verify prerequisites, deploy Symantec Identity Security Platform with Envoy Gateway API, or migrate an existing nginx-based deployment to Gateway API.

Pre-requisite Check

> **NOTE**
> Before proceeding with the specific checks mentioned in this section, review and complete all items in the Prerequisites for All Deployments list.

Run these commands to verify all prerequisites are in place before deploying Symantec Identity Security Platform:

1. Verify Gateway API CRDs are installed:
   ```
   kubectl get crd gateways.gateway.networking.k8s.io
   ```
2. Verify Envoy Gateway pods are running:

```
kubectl get pods -n envoy-gateway-system
```

3. Verify GatewayClass is created and accepted:

```
kubectl get gatewayclass
# Expected: envoy-gateway-class with Accepted=True
```

4. Verify BackendTLSPolicy CRD has v1 version (required by Envoy Gateway v1.6+):

```
kubectl get crd backendtlspolicies.gateway.networking.k8s.io \
  -o jsonpath='{.spec.versions[*].name}'
# Expected: should include "v1". If only "v1alpha3" is listed,
# upgrade Gateway API CRDs to v1.4.0+
```

5. Verify TLS certificate Secret exists:

```
kubectl get secret <tls-secret-name> -n ssp
```

6. Verify RBAC permissions (current user can create Gateway API resources):

```
kubectl auth can-i create gateways.gateway.networking.k8s.io -n ssp
kubectl auth can-i create httproutes.gateway.networking.k8s.io -n ssp
kubectl auth can-i create backendtlspolicies.gateway.networking.k8s.io -n ssp
kubectl auth can-i create clienttrafficpolicies.gateway.envoyproxy.io -n ssp
kubectl auth can-i create backendtrafficpolicies.gateway.envoyproxy.io -n ssp
```

Minimum ClusterRole for Envoy Gateway:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: gateway-api-admin
rules:
  - apiGroups: ["gateway.networking.k8s.io"]
    resources: ["gateways", "httproutes", "backendtlspolicies"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
  - apiGroups: ["gateway.envoyproxy.io"]
    resources: ["clienttrafficpolicies", "backendtrafficpolicies"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

Before You Deploy

Complete the following before deploying or migrating Symantec Identity Security Platform with Gateway API.

NOTE

1. Gateway ownership: The Gateway can be created by Helm (`createGateway: true` , default) or pre-created by the platform team (`createGateway: false` + `existingGateway`). To know how to use an already existing Gateway, see Already Have a Gateway? section for details.
2. Set these Environment Variables:

```
export RELEASENAME="<SSP_RELEASENAME>"
export NAMESPACE="<SSP_NAMESPACE>"
export HELM_REPO='ssp_helm_charts'
```

3. Run the following command to export the current override values of your Helm release to `ssp-override.yaml` . This lets you review, back up, or reuse your existing configuration:

```
helm get values ${RELEASENAME} -n ${NAMESPACE} >ssp-override.yaml
```

Deploying Symantec Identity Security Platform with Gateway API (Fresh Install)

Follow these steps to deploy Symantec Identity Security Platform with Envoy Gateway API on a fresh installation.

1. Ensure Envoy Gateway is installed and GatewayClass exists (see Infrastructure Setup section).

2. Deploy Symantec Identity Security Platform with Gateway API:

```
helm upgrade --install ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.gatewayApi.gatewayClassName=envoy-gateway-class \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.createBackendTLSPolicy=true
```

3. Verify resources:

```
kubectl get gateway,httproute,backendtlspolicy -n ${NAMESPACE}
kubectl get clienttrafficpolicies,backendtrafficpolicies -n ${NAMESPACE}  # Envoy-specific
```

4. Test connectivity:

```
GATEWAY_IP=$(kubectl get gateway -n ${NAMESPACE} -o jsonpath='{.items[0].status.addresses[0].value}')
curl -sk --resolve "your-host:443:$GATEWAY_IP" \
  "https://your-host/actuator/health"
```

What Helm creates for Envoy:

Resource	Name Pattern	Purpose
Gateway	`<release>-ssp-gateway`	HTTPS listeners with TLS termination
HTTPRoute	`<release>-ssp-router-full`	Routes all traffic to ssp-router
HTTPRoute	`<release>-ssp-router-mgmt`	Management hostname route (if mgmt enabled)
BackendTLSPolicy	`<release>-ssp-router-backend-tls`	CA validation for HTTPS to ssp-router
ClientTrafficPolicy	`<release>-ssp-client-traffic`	Buffer limit (512KB), client timeout, header preservation, mTLS client validation
BackendTrafficPolicy	`<release>-ssp-backend-traffic`	Backend request timeout, idle timeout

NOTE

Envoy mTLS — how cert forwarding works:

Client validation: When `mtls.enabled=true` , the `ClientTrafficPolicy` includes `tls.clientValidation` with `optional: true` (allows requests without client certs) and a CA certificate bundle for validation.

Certificate forwarding: Envoy forwards the validated client certificate via the `x-forwarded-client-cert` (XFCC) header using the following settings:

- `forwardClientCertDetails: SanitizeSet` — sanitizes any client-injected XFCC values.
- `setCertificateDetails: Cert` — includes the full URL-encoded PEM in the XFCC header as `Cert="<URL-encoded PEM>"` .

Header normalization: ssp-router extracts the PEM from the XFCC header and normalizes it into the `X-CLIENT-CERT` header for backend services.

Migrating Symantec Identity Security Platform from Nginx to Gateway API

Follow these steps to migrate an existing nginx-based Symantec Identity Security Platform deployment to Envoy Gateway API.

NOTE
Before proceeding, complete all steps in the Before You Deploy section.

1. Deploy with both nginx and Gateway API active using `previousController.enabled: true` to maintain both paths during transition:

```
helm upgrade ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.previousController.enabled=true \
  --set ssp.ingress.previousController.ingressClassName=nginx \
  --set ssp.ingress.gatewayApi.gatewayClassName=envoy-gateway-class \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.createBackendTLSPolicy=true
```

2. After verifying Gateway API works and switching DNS, remove the old controller:

```
helm upgrade ${RELEASENAME} ${HELM_REPO}/ssp -n ${NAMESPACE} \
  -f your-values.yaml \
  --set ssp.ingress.type=gatewayapi \
  --set ssp.ingress.previousController.enabled=false \
  --set ssp.ingress.gatewayApi.gatewayClassName=envoy-gateway-class \
  --set ssp.ingress.gatewayApi.createGateway=true \
  --set ssp.ingress.gatewayApi.createBackendTLSPolicy=true
```

Deploying the Sample App to Test Connectivity

Deploy the sample-app with Envoy Gateway to test connectivity before or alongside Symantec Identity Security Platform.

1. Deploy the sample-app:

```
helm upgrade --install ${RELEASENAME} ${HELM_REPO}/ssp-sample-app -n ${NAMESPACE} \
  -f your-sample-values.yaml \
  --set ingress.type=gatewayapi \
  --set ingress.gatewayApi.gatewayClassName=envoy-gateway-class \
  --set ingress.gatewayApi.createGateway=true \
  --set ingress.gatewayApi.createBackendTLSPolicy=true
```

2. Verify resources:

```
kubectl get gateway,httproute,backendtlspolicy -n ${NAMESPACE}
```

3. Get Gateway IP and test:

```
GATEWAY_IP=$(kubectl get gateway -n ${NAMESPACE} -o jsonpath='{.items[0].status.addresses[0].value}')
curl -sk --resolve "your-host:443:$GATEWAY_IP" "https://your-host/sample-app/"
```

Troubleshooting Envoy Deployments

Condition

Issues with Envoy Gateway deployments including Gateway IP assignment, HTTPRoute routing, backend TLS, ClientTrafficPolicy status, pod readiness, and NLB target health.

Cause: Gateway not getting IP address

Remedy

Run the following commands:

```
kubectl get gateway -n ssp -o yaml
kubectl describe gateway <gateway-name> -n ${NAMESPACE}
kubectl get gatewayclass
```

Cause: HTTPRoute not routing traffic

Remedy

Run the following commands:

```
kubectl get httproute -n ssp -o yaml
kubectl get httproute -n ssp -o custom-columns='NAME:.metadata.name,PARENT:.spec.parentRefs[0].name'
```

Cause: Backend TLS errors

Remedy

Run the following commands:

```
kubectl get backendtlspolicy -n ${NAMESPACE} -o yaml
kubectl get secret ${RELEASENAME}-ssp-keys-isk-ca -n ${NAMESPACE}
```

Cause: ClientTrafficPolicy has no status (mTLS not working despite correct config)

This is a known race condition: if the Envoy Gateway controller starts **before** the `ClientTrafficPolicy` CRD is installed on the cluster, it skips the CTP watch entirely. CTP resources are created and appear correct (`kubectl get ctp -A` shows them), but they have **no status conditions** — meaning the controller never processed them. The data plane won't apply any CTP settings (mTLS client validation, XFCC forwarding, buffer limits, etc.).

Symptoms:

- `kubectl get ctp -A` shows the resource but no `ACCEPTED` or `PROGRAMMED` status
- Controller log: `ClientTrafficPolicy CRD not found, skipping ClientTrafficPolicy watch`
- Client certificates not forwarded to backends despite correct CTP config

Remedy

1. Restart the Envoy Gateway controller so it re-discovers the CRD:

   ```
   kubectl rollout restart deployment/envoy-gateway -n envoy-gateway-system
   ```

2. After restart, verify the CTP is processed:

   ```
   kubectl get ctp -A
   # STATUS should show Accepted: True / Programmed: True
   ```

Cause: Envoy Gateway pods not ready (EKS)

Remedy

Run the following commands:

```
kubectl get pods -n envoy-gateway-system
```

```
kubectl describe pod -n envoy-gateway-system -l app.kubernetes.io/name=envoy-gateway
```

Cause: Verify NLB target health (EKS)

Remedy

Run the following command:

```
aws elbv2 describe-target-health --target-group-arn <target-group-arn>
```

AKS - Envoy Gateway
Describes how to configure Envoy Gateway on Azure Kubernetes Service (AKS) to support client certificate forwarding and backend TLS, including workarounds for BackendTLSPolicy CRD limitations on Kubernetes versions 1.34 and earlier.

> **NOTE**
> Azure AGC cannot forward client certificates. Use Envoy Gateway on AKS instead.

AKS prerequisite — BackendTLSPolicy CRD

Envoy Gateway v1.6+ requires the standard BackendTLSPolicy CRD (`gateway.networking.k8s.io/v1`), which became GA in Gateway API v1.4.0. On AKS, the available Gateway API CRDs are tied to the Kubernetes version via a managed validating webhook (`managed-gateway-api-ccp-validating-webhook`). This webhook blocks manual installation of Gateway API CRDs.

AKS with K8s 1.35+

Gateway API v1.4.1 is bundled — BackendTLSPolicy is available automatically. No action needed.

AKS with K8s ≤1.34

The bundled Gateway API version does not include BackendTLSPolicy.

> **WARNING**
> In this case, the Helm chart skips creating the BackendTLSPolicy resource and displays a warming message. This results in Envoy Gateway NOT connecting to ssp-router over HTTPS, which fails with 503 upstream connect errors. Envoy Gateway does not use `appProtocol` for backend TLS; it requires either a BackendTLSPolicy or a Backend resource.

Two options:

Option A — Delete the managed webhook and install CRDs manually

1. Delete the managed webhook (takes over CRD lifecycle from AKS):
   ```
   kubectl delete validatingwebhookconfiguration managed-gateway-api-ccp-validating-webhook
   ```
2. Install Gateway API v1.4.0 CRDs (includes BackendTLSPolicy):

   ```
   kubectl apply -f https://github.com/kubernetes-sigs/gateway-api/releases/download/v1.4.0/standard-install.yaml
   ```

> **CAUTION**
> After deleting the managed webhook, AKS no longer manages Gateway API CRD upgrades automatically. You are responsible for keeping the CRDs up to date when upgrading Envoy Gateway.

Option B — Use an Envoy Gateway Backend resource (no webhook change required)

Instead of BackendTLSPolicy, use Envoy Gateway's own Backend CRD for backend TLS. This CRD ships with Envoy Gateway and is not affected by the AKS managed webhook.

Follow these steps:

1. **Enable the Backend API in Envoy Gateway**:
   In the **Envoy Gateway Helm chart deployment** (not the IDSP chart), enable the Envoy Gateway Backend API support:
   ```
   --set config.envoyGateway.extensionApis.enableBackend=true
   ```
2. **Create a Backend resource for ssp-router:**
   ```
   apiVersion: gateway.envoyproxy.io/v1alpha1
   kind: Backend
   metadata:
     name: ssp-router-backend
     namespace: <namespace>
   spec:
     endpoints:
       - fqdn:
           hostname: <release>-ssp-router.<namespace>.svc.cluster.local
           port: 8095
     tls:
       sni: <release>-ssp-router.<namespace>.svc
       caCertificateRefs:
         - group: ""
           kind: Secret
           name: <release>-ssp-keys-isk-ca   # Secret with ca.crt key
   ```

   IMPORTANT
   The `sni` field must be set to the ssp-router service's short name (`<release>-ssp-router.<namespace>.svc`) to match the router certificate's SAN (`*.<namespace>.svc`). If you omit `sni` , Envoy uses the full FQDN (`...svc.cluster.local`) which does not match the certificate's wildcard SAN, causing `CERTIFICATE_VERIFY_FAILED` errors.

   NOTE
   Troubleshooting: If you need to temporarily bypass backend certificate validation (e.g., to isolate TLS issues), you can replace the `tls` block with `tls: { insecureSkipVerify: true }` .
3. The **IDSP Helm chart deployment** should route HTTPRoutes to the Backend instead of the Service:

   ```
   ssp:
     ingress:
       gatewayApi:
         createBackendTLSPolicy: false
         backendRef:
           group: gateway.envoyproxy.io
           kind: Backend
           name: ssp-router-backend
           port: 8095
   ```

Constructing the Service FQDN

Learn how to construct the Service FQDN by specifying the domain, and the service-hostname. Decide on a Releasename and Namespace that can be used in the commands throughout the release process by creating environment variables.

- Setup steps assume that you specify a root domain <DOMAIN> in the Configuring the Environment Variables section. For example, the domain could look like this- "myiam.example.com". By default, the deployment steps assume services are deployed under the same DNS domain name, and the services FQDN is added to the DNS (or configured in /etc/hosts) in the form of "<service-hostname>.<DOMAIN>".
- Setup steps assume that you specify the value for "<PREFIX>" in the Configuring the Environment Variables section. The <PREFIX> specifies the Symantec Identity Security Platform service-hostname which results in the FQDN in the form of "<PREFIX>.<DOMAIN>".
- It is recommended that you acquire a TLS certificate signed by a well-known and trusted Certificate Authority (CA), for example; Let's Encrypt, Digicert. The commands in the following sections use a wildcard certificate for *. <DOMAIN> (for example, "*.myiam.example.com"). However, you could also use host-specific certificates.

  NOTE
  If you do not provide a certificate, the deployment uses a self-signed certificate for all ingress objects. However, note that not all clients are able to work with self-signed certificates. For example, some clients may reject self-signed SSL connections unless those clients have a way to enable the "ignoreSSL" capability.
- The key file and the certificate that is used in the deployment steps refer to the key file as (.key) and the cert file as (.crt), as provided by the CA. Once you have obtained the issuer certificate package containing the CA public certificate series, copy the contents of the package to the bottom of the issued .crt file (domain certificate) and save it as a separate .crt file.
- <RELEASENAME> refers to the helm Release-Name and <NAMESPACE> refers to the Kubernetes Namespace of the IDSP deployment.

The FQDN is used during the creation of environment variables for the deployment of IDSP. See Configuring the Environment Variables section.

Symantec Identity Security Platform Components and URLs

Includes the URLs of the Symantec Identity Security Platform Components.

The following table displays the IDSP components and their respective URLs:

Service	Description	Kubernetes Namespace	Service URL
IDSP	IDSP Services	<user-defined>	https://<PREFIX>.<DOMAIN>/default/<API>
SignIn UI	IDSP SignIn UI	<user-defined>	https://<PREFIX>.<DOMAIN> /default/ui/v1/adminconsole/
Admin Console	IDSP Admin Console	<user-defined>	https://<PREFIX>.<DOMAIN> /default/ui/v1/adminconsole/
IDSP Swagger	IDSP Swagger Documentation	<user-defined>	https://<PREFIX>.<DOMAIN> /default/swagger-ui/index.html
Kibana	Logging of IDSP APIs and 3rd party components	Logging	https://kibana.<DOMAIN>
Grafana	Monitoring of IDSP APIs health and Cluster health	Monitoring	https://grafana.<DOMAIN>
Alert Manager	Alerting based on IDSP KPIs	Monitoring	https://alertmanager.<DOMAIN>

Ingress	Ingress	Ingress	n/a

Master Encryption Key

Creating a master encryption key and backing it up for future retrievals

This topic includes the following sub-sections:

- Backing-up and Restoring the MEK
- New deployment with existing database
- Disaster Recovery deployment

- The Master Encryption Key is used to encrypt the tenants' data encryption keys of the VIP Authentication Hub. Every VIP Authentication Hub installation requires an MEK. The MEK is stored as a Kubernetes secret in a file called, "mek.key" which contains the MEK key value.
- If you do not provide an MEK during deployment, an MEK is dynamically generated as a Kubernetes secret name -"`<RELEASENAME>-ssp-keys-mek`" where the "<RELEASENAME> is the name of the release that is used during the deployment of IDSP helm chart.
- To use an existing MEK, create a Kubernetes secret that contains a file that is called "mek.key" which contains the MEK key and specify `--set ssp.keys.mek.existingSecret="<name of the Kubernetes Secret containing MEK.key>"` parameter during the IDSP helm chart deployment.
  The following command is an example of creating a MEK secret named "ssp-mek-secret" in a namespace called "ssp" with the MEK value of "Th1s!sMyT0pS3cretM3K":
  ```
  kubectl create secret generic ssp-mek-secret -n ssp --from-literal=mek.key='Th1s!sMyT0pS3cretM3K'
  ```
  Once the secret is created, add `--set ssp.keys.mek.existingSecret="ssp-mek-secret"` to your Authentication Hub helm chart deployment command.

  NOTE

  - It is important to create a backup of the MEK because it is the only key that can be used to decrypt the solution's encrypted data stored in the Authentication Hub database. If you lose the MEK, there is no other way to recover the encrypted data.
  - If you are pointing to a replicated instance of an existing Authentication Hub database or restoring an existing database from a backup, ensure to use that MEK.

Important: If you are upgrading from 2021.April.01 or a prior release to the current release, VIP Authentication Hub automatically generates a MEK (if a name of a MEK existing Kubernetes secret was not provided), and this MEK re-encrypts the tenant data encryption key.

Backing-up and Restoring the MEK

It is important to back up your MEK into a safe place for future retrievals. If you have created your own MEK, make sure you save it in a secure location. If the MEK was automatically generated by IDSP during the deployment of the Authentication Hub, ensure to export and save it.

There are two ways to back up the MEK:

- Export the complete secret object using the following command:
  ```
  kubectl get secret ${RELEASENAME}-ssp-keys-mek -n ${NAMESPACE} -o yaml >mek-secret_backup.yaml
  ```
  This command saves the content of the MEK secret into a yaml file called, mek-secret_backup.yaml. In the future when you want to restore it, use the following command:
  ```
  kubectl apply -f <mek secret yaml file> -n ${NAMESPACE}
  ```
- Export the MEK value only, using the following command:
  ```
  kubectl get secret ${RELEASENAME}-ssp-keys-mek -n ${NAMESPACE} -o jsonpath="{.data.mek\.key}" | base64 --
  decode; echo
  ```
  In the future, when you want to restore the MEK value only, use the following command:

```
kubectl create secret generic <secretname> -n ${NAMESPACE} --from-literal=mek.key=<mek value>
```

New Deployment with Existing Database

The MEK is the key to unlock the Authentication Hub database. If you are pointing to a replicated instance of an existing Authentication Hub database or restoring an existing database from a backup, ensure to use that database's MEK.

1. Before deploying the new instance of the Authentication Hub, create a Kubernetes secret, using the MEK value of the existing deployment in the new Authentication Hub deployment location (deployment cluster and Kubernetes namespace). To perform this step, follow the restore steps mentioned in the section- **Backing-up and Restoring the MEK**.
2. Create an Authentication Hub deployment, pointing to the database of the existing deployment and to the MEK secret created in the previous step, by adding the following parameter to the Authentication Hub helm chart install command:

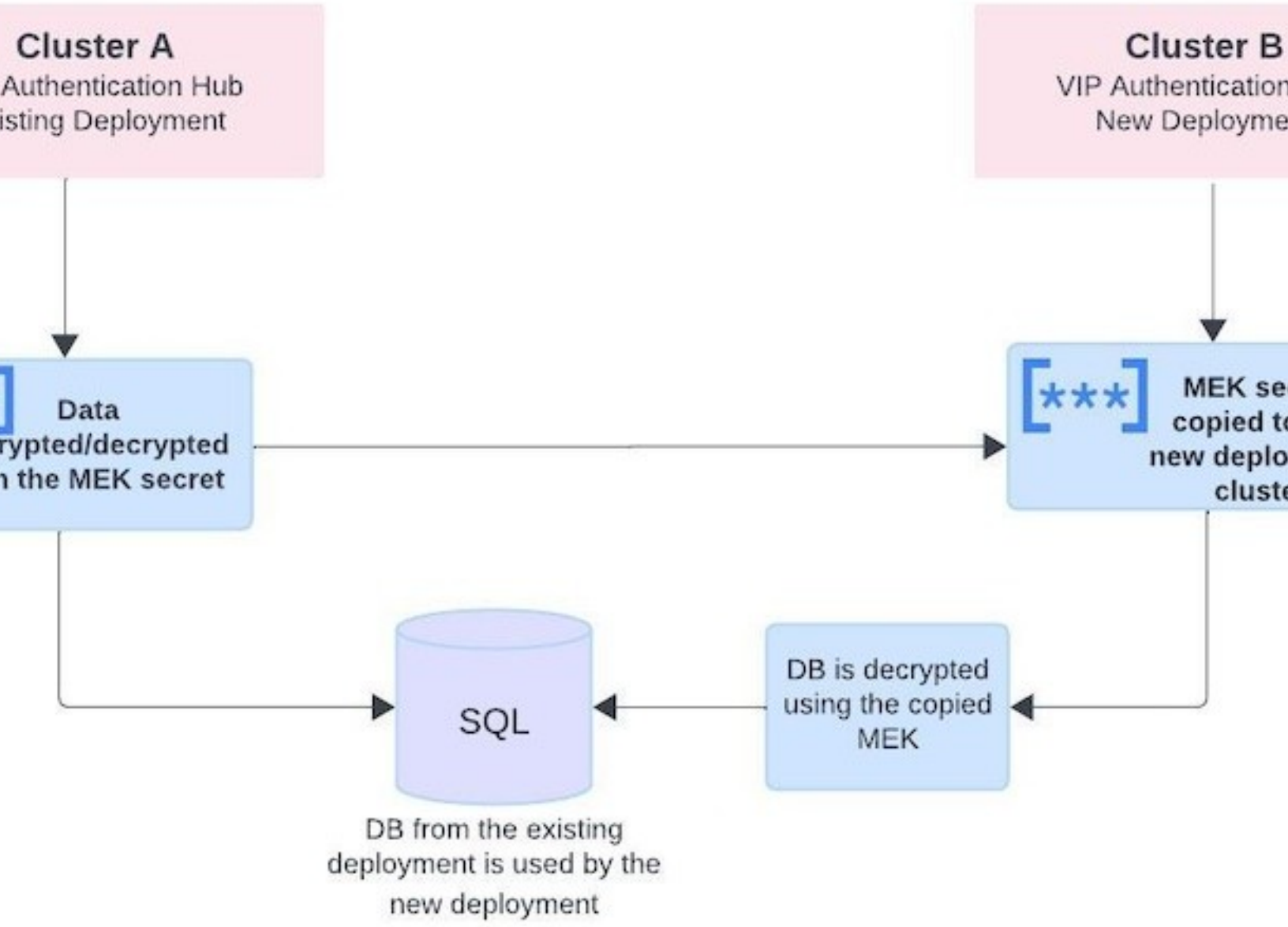
```
--set ssp.global.drSite=true --set ssp.keys.mek.existingSecret="<name of the Kubernetes Secret containing
 MEK.key>"
```

NOTE

- When adding the parameter, ssp.global.drSite=true, no updates are made to the existing database.
- When performing an Authentication Hub version upgrade or deployment changes that require an update to the database, always perform the update or the upgrade on the primary deployment, and not on a DR deployment site (where ssp.global.drSite=true).

Two separate deployments that are using the same database, with the same MEK value

The following image illustrates the process of securely managing database access in a VIP Authentication Hub deployment with two clusters (Cluster A and Cluster B). Cluster A is the existing deployment, while Cluster B is the new deployment. Both clusters use the same database, which is encrypted with a Master Encryption Key (MEK). The MEK secret is copied from Cluster A to Cluster B, allowing Cluster B to decrypt and access the same database.



Disaster Recovery (DR) setup for the VIP Authentication Hub using a load balancer

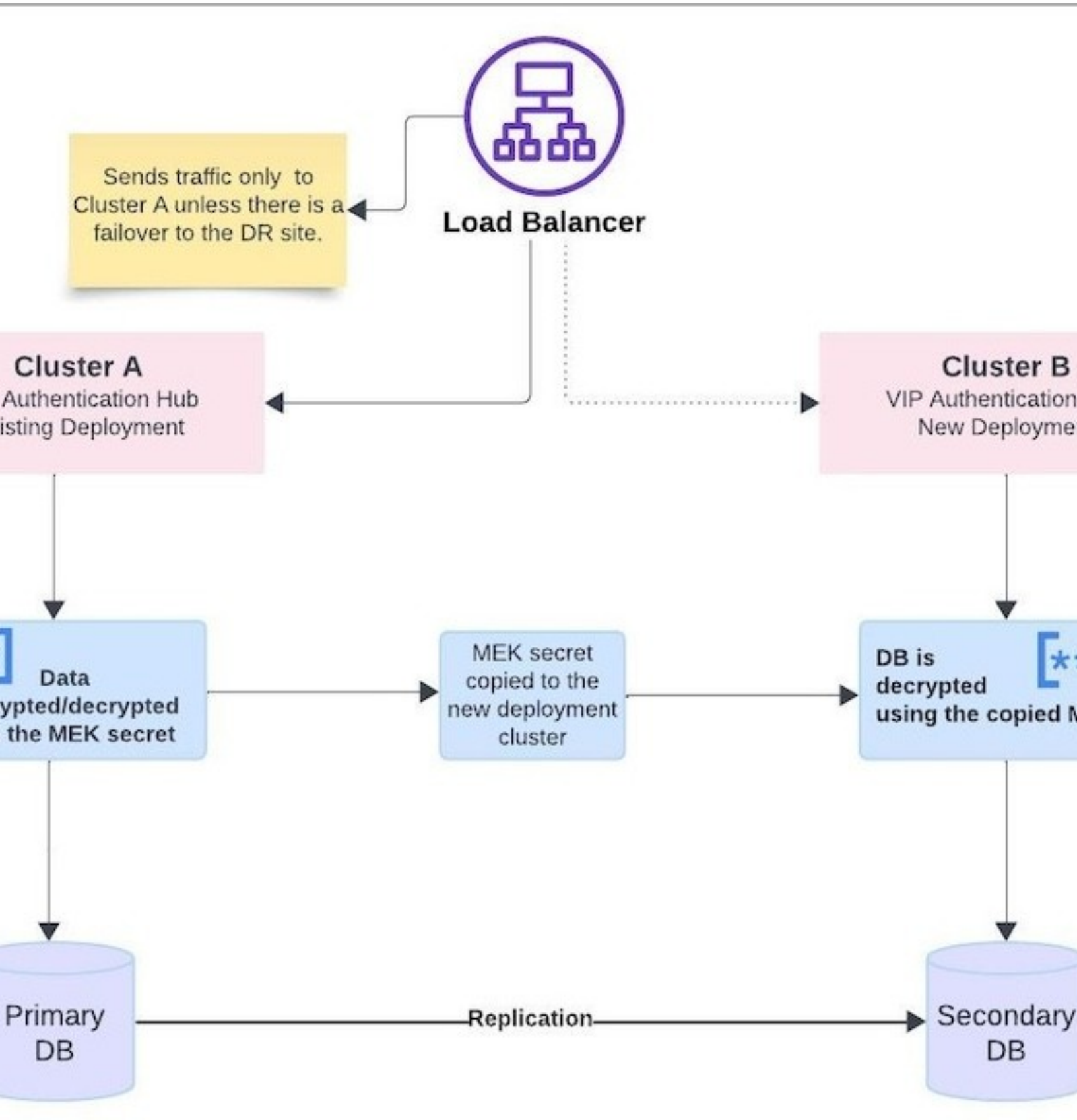
The following image illustrates a DR setup for the VIP Authentication Hub using a load balancer to manage traffic between two clusters (Cluster A and Cluster B). Cluster A uses the primary db serving as the active deployment that handles all traffic unless a failover occurs. Data in the primary database is encrypted using a Master Encryption Key (MEK) secret. This MEK secret is copied to Cluster B, the DR deployment, which uses the same MEK.

In case of a failover, the secondary db must be promoted to a standalone primary db, only then the load balancer starts sending traffic to Cluster B. When Cluster A is available again, a failback procedure should take place which includes the following steps:

1. Creating Cluster A's db as a replica of Cluster B's db.
2. Promote Cluster B's db to primary and switchover load balancer to send traffic to Cluster A.
3. Recreate Cluster B as a readonly db of Cluster A.

IMPORTANT

- The MEK is copied to the DR cluster because both clusters use the same MEK, which is essential for encrypting and decrypting sensitive data in the database.
- If the secondary database is a read-only replica of the primary database, set the parameter, "***drsite=true***" during Authentication Hub deployment to prevent Authentication Hub from updating the database during installation and upgrades.
- In the following diagram, Cluster A and Cluster B are deployed in two separate regions. These clusters should be created as multi-zone clusters to leverage cloud providers infrastructure to get better resiliency within their region.


Sends traffic only to Cluster A unless there is a failover to the DR site.
Load Balancer
Cluster A
Authentication Hub
isting Deployment
Cluster B
VIP Authentication
New Deployme
Data
ypted/decrypted
the MEK secret
MEK secret copied to the new deployment cluster
DB is decrypted using the copied
Primary DB
Replication
Secondary DB

DR Deployment Steps

View information about the Disaster Recovery (DR) deployent steps.

This section describes the steps taken for testing Cloud SQL MySQL failover to another cross-region.

Assume a primary instance is deployed in **us-east-1**, and a replica is deployed in **us-west-1.**

The following steps simulate an instance that "stopped" in **us-east-1**.

General Approach:

1. Ensure that a cross-region replica has been created and can be accessed from the cluster.
2. Capture a fresh backup to ensure that no data has been lost.
3. Simulate a Disaster Recovery (DR) Scenario.
4. Implement DR.
5. Delete the old primary instance (Clean-up).

Detailed Steps:

Step	Steps Description	Detailed Steps
1	Create a cross-region replica from the primary instance, in this scenario, a replica is created in us-west-1	This can be done either directly in the console or by using the following `gcloud` command: `gcloud sql instances create $cross_region_replica_name \ --master-instance-name=$primary_name \ --region=$cross_region_replica_region`

1a	Confirm if us-west-1 Replica instance can be connected from mysql client pod	Create MySQL Pod in preprod jumphost (jumphost that is used for the testing environment) by running the following command:
		`kubectl run mysql-client -n ssp --image=mysql:5.7 -it --rm --restart=Never -- /bin/bash`
		In POD, connect by running the following command:
		`mysql -u iamauth -p '=*HA@=_z@<8qht$.' -h 10.254.91.2`
		k delete pod mysql-client

2	Capture a backup of the primary instance	Either click "Create Backup" in the console or run the following gcloud command: `gcloud sql backups create --async --instance=$primary_name`
2a	List the Backup ID created in the previous step and confirm if backup has created successfully.	Use the following command to output the Backup ID and check the status, which should be shown as SUCCESSFUL: `gcloud sql backups list --instance sample-ah-db-prd`
3	Disable the deletion protection of the primary instance.	Edit the master instance and click "Disable Deletion Protection." This must be done before the instance is stopped, as GCP will not allow you to edit the instance afterward.
3a	Remove the cross-region replica from the primary instance by running the following command:	`gcloud sql instances patch sample-ah-db-prd-replica --no-enable-database-replication` After the command executes successfully, the replica state should display as "replication disabled."

Step	Steps Description	Detailed Steps
3b	Stop the primary database instance	Click `STOP` or run the following gcloud command: `gcloud sql instances patch $primary_name --activation-policy NEVER`
4	Promote the cross-region read replica to a standalone instance	Click `Promote` or run the following gcloud command: `gcloud sql instances promote-replica $cross_region_replica_name`
4a	Start the new primary (the replica that was promoted in step 4).	The `$primary_name` now refers to the replica name that was promoted in step 4. Use the following command to set the activation policy to ALWAYS: `gcloud sql instances patch $primary_name --activation-policy ALWAYS`
4b	Enable the new primary as an HA Regional instance.	The`$primary_name` is now the replica name that was promoted in step 4. `gcloud sql instances patch $primary_name \ --availability-type REGIONAL \ --enable-bin-log \ --backup-start-time=$backup_start_time`
4c	Update the values.yaml file with the new host IP address (replica address).	1. Navigate to the correct branch (environment) in the repository sample-authhub-cd, specifically in the Symantec Identity Security Platform branch. For example: `pipeline_ssp_int_gke-authhub-$ENVNAME-$ENVREGION-int` 2. Update the `spec.db.serviceHost` field to the new primary (replica) address.
4d	Create a cross-region read-replica in a different region (us-east).	Click `Create Replica` or run the following gcloud command `gcloud sql instances create $cross_region_replica_name \ --master-instance-name=$primary_name \ --region=$cross_region_replica_region`
4e	Ensure the dataseed pod in the IDSP namespace is marked as "COMPLETED" and that new IDSP pods are spinning up. Additionally, verify that the admin console is still accessible and that you can log in.	See the following output, all IDSP pods are in a RUNNING state. Use the following command: `kubectl get pods -n ssp`
5	Delete the old primary instance	Click Delete or run the following gcloud command:
5a	(Optional) If you're creating a standalone instance and want to back up data from a previous instance, run the following command:	`gcloud sql backups restore $BACKUP_ID \ --restore-instance=TARGET_INSTANCE_NAME \ --backup-instance=SOURCE_INSTANCE_NAME`

IDSP Deployment Steps (DR Setup for Failover Cluster)

Ensure the Cloud SQL MySQL read-replica are functioning properly and can access the cluster. Then, deploy IDSP in the failover cluster.

Step	Steps Description	Detailed Steps
1	Retrieve the following secrets from the primary cluster: • sample-authhub-ssp-prd-ssp-east-ssp-keys-mek • sample-authhub-ssp-prd-ssp-east-ssp-keys-isk-ca • sample-authhub-ssp-prd-ssp-east-ssp-keys-isk • hazelcast-ssp-keys-isk With IDSP v 3.2, the IDSPchart supports DR deployment mode which automates all Kubernetes provisioning steps. It including generating local ISK secrets, while skipping data loading since the DR DB is assumed to be in read-only mode. Therefore, no ISK secrets must be copied. When deploying the chart, ensure you set the `drSite` parameter: `ssp.global.drSite=true` and `ssp.db.dataseed.enabled=true` . When "drSite=true", the database operates in a read-only mode, preventing any schema or data updates. However, dataseed must still be run to manage (includes create and update) Kubernetes secrets (isk/ssl secrets).	Retrieve the secrets into a file using the following command: `kubectl get $SECRET_NAME -n ssp -o yaml > $FILE_NAME`
2	Apply the secrets retrieved from step 1.	Update the values for the following secrets to align with the current environment: `ssp-isk-ca` ,`ssp-keys-isk` , and `ssp-keys-mek` . The `hazelcast-ssp-keys-isk` must have the same name as in the primary cluster. `kubectl apply -f $FILE_NAME -n ssp`

3	Add the following values to values.yaml and deploy IDSP helm chart in the failover cluster.	`global:`
		`  helmOfflineMode: true`
		`ssp:`
		`  global:`
		`  drSite: true`
		`  keys:`
		`  mek:`
		`  existingSecret: <sample-authhub-ssp-prd-ssp-west-ssp-keys-mek>`
		`  db:`
		`  dataseed:`
		`  enabled: true`
		
		`  featureFlags:`
		`  scheduler:`
		`  enabled: false`

Troubleshooting Steps

Error	Steps Taken to Resolve
failed exit status 1: Error: execution error at (ssp/templates/mek-secret.yaml:84:3): ***** Existing Master Encryption Key (MEK) secret specified (ssp-keys-mek) but does not exist\n\nUse --debug flag to render out invalid YAML" kubectl logs POD_NAME	Set dataseed to false and recreate the secrets from primary to failover (see section 2 step 2).
{"timestamp":"2024-04-25T17:36:11.349651Z","type":"log","level":"error","thread":"main","msg":"(conn=103) The MySQL server is running with the --super-read-only option so it cannot execute this statement","tid":"e202f42f-95fc-48dd-83b5-e72ec36d5c71","tname":"system"}	Make sure that the primary cluster is not pointed to the read/write db d.

Cache Considerations

Learn about Hazelcast configurations required during Symantec Identity Security Platform deployment.

IDSP uses Hazelcast (Cache) Enterprise version 5.2.1 which supports configuring the role-binding and service account during the IDSP deployment.

By default, the IDSP does not require Role-based access control (RBAC) for discovering Hazelcast cache instances for cache replication ("dns" discovery mode). Hazelcast maintains a backup for each replica in another instance. When running in a multi-zone cluster, "kubeapi" discovery mode is recommended. "kubeapi" discovery mode supports two modes, using a Kubernetes cluster role or a Kubernetes local role. It is recommended to use the Kubernetes cluster role. Granting the ClusterRole permissions to the Hazelcast service account improves data loss prevention by allowing Hazelcast to get information about the NODE and ZONE of each cache instance. Hazelcast uses this information to make a better decision about where to store replicas' backups (trying to avoid storing a replica's backup on the same node or the same zone as the replica itself).

Switching to "kubeapi" discovery mode can be done by setting the following parameters when deploying the IDSP helm chart to create the Hazelcast service account and grant permissions to it:

```
--set ssp.global.cache.discoveryMode=kubeapi
--set hazelcast-enterprise.rbac.create=true
--set hazelcast-enterprise.rbac.useClusterRole=true
```

Switching to "kubeapi" discovery mode can also be done ***manually*** by creating the proper Kubernetes RBAC Role or ClusterRole, binding it to the service account, and specifying that service account by adding the following parameters to the IDSP helm chart:

```
--set hazelcast-enterprise.serviceAccount.create=false \
--set hazelcast-enterprise.serviceAccount.name=<serviceaccount-name> \
--set "ssp.global.cache.discoveryMode"="kubeapi"
```

See HAZELCAST documentation for more information.

The following table shows the <parameter_name> and describes the <value> that can be passed into the IDSP helm chart using `--set <parameter_name>="<value>"` helm command.

Parameter Name	Default	Description
hazelcast-enterprise.rbac.create	false	Possible Values: • true - Creates a cluster role, and cluster role binding (or role and role binding depending on the value of hazelcast-enterprise.rbac.useClusterRole). • false - Does not create cluster role, and cluster role binding (or role and role binding)
hazelcast-enterprise.rbac.useClusterRole	false	Possible Values: • true - Set this parameter to "true" to create a cluster role. • false - Set this parameter to "false" to create a Kubernetes Role instead of a Kubernetes Cluster Role. When this parameter is set to false, some of the discovery features are unavailable.
hazelcast-enterprise.serviceAccount.create	true	Possible Values: • true: Set this parameter to true to enable creation of the Service Account. • false: Set this parameter to false to disable creation of the Service Account. In this case, you must specify the service account name to be used by specifying it in the parameter: hazelcast-enterprise.serviceAccount.name=<serviceaccount-n

Deploying Enclave Services

Enclave (observability) services refer to a group of external services (such as Elasticsearch, Kibana, Prometheus, Grafana) that are plugged into Symantec Identity Security Platform to collect information that is related to search, logging, and monitoring from services that are running in the IDSP Kubernetes cluster.

> **NOTE**
> Support for cloud-native "**Elastic" / "Kibana"** and "**Prometheus" / "Grafana"** in the cloud-native IDSP is provided on a best-effort basis. While customers typically have their own enterprise observability stacks, we aim to deliver an out-of-the-box observability experience using robust open-source alternatives, allowing them the flexibility to choose the appropriate tool for their specific environment.
> We certify major releases of both these platforms for compatibility with the IDSP, but the responsibility for upgrading these platforms to address vulnerabilities or use new features lies with our customers, following the general guidelines based on our certifications.

This page includes the following topics:

- Prerequisites
- Read Me First
- Configuring Environment Variables

Prerequisites

Before setting up the enclave services, ensure that you have gone through the Prerequisites and installed all the required software.

Read Me First

- Constructing the DNS domain name:
  - Setup steps assume a root domain, <DOMAIN> that is set by you in the section-Configuring Environment Variable and again while configuring */etc/hosts or the DNS entries.* By default, the deployment steps assume that services

are hosted under one DNS domain. For example, the domain value can be "myiam.example.com" and the services Fully Qualified Domain Name can be *kibana.myiam.example.com* or *grafana.myiam.example.com.*

- It is recommended that you acquire a TLS certificate signed by a well-known and trusted Certificate Authority (CA), for example; Let us Encrypt, Digicert. The following commands use a wildcard certificate for *. <DOMAIN> (for example, "*.myiam.example.com"), but you could also use host-specific certs.

  NOTE
  If you do not provide a certificate, the deployment uses a self-signed certificate for all ingress objects. However not all clients are able to work with self-signed certificates. For example, some clients reject self-signed SSL connections unless those clients have a way to enable the "ignoreSSL" capability.
- The key file and the certificate that is used in the deployment steps refer to the key file (.key) and the cert file (.crt) as provided from the Certificate Authority (CA). If you also receive an issuer certificate package containing the CA public certificate series, copy the contents of the package to the bottom of the issued .crt file (domain certificate) and save it as a separate .crt file.

Configuring Environment Variables

Set up your system environment variables so that you can use them in the commands that are used in deploying the enclave services.

These commands can be executed from your on-premise workstation or in the case of GCP, you could also use a cloud shell:

Follow these steps:

1. Configure environment variables to point to your DNS domain.
   ```
   export DOMAIN="<DOMAIN>"
   ```
   NOTE
   Example of <DOMAIN> name: myiam.example.com, iam.example.com
2. If you have a wildcard certificate from a trusted Certificate Authority (CA), configure the environment variables to point to the key and cert files.
   ```
   export KEYFILE="<keyfile>"
   export CERTFILE="<certfile>"
   ```

See the following topics:

- Deploying Elasticsearch and Kibana for Logging
- Deploying Prometheus for monitoring

Deploying Elastic Operator and Kibana

Learn to setup a basic Elasticsearch deployment (non-production) to index and analyze all the Symantec Identity Security Platform logs, view them in the Kibana interface, and configure a data retention policy in Elastic Search.

This topic includes the following sections:

- Deploying Elasticsearch and Kibana for Logging
- Publishing the Kibana Endpoint

Deploying Elastic Search and Kibana

NOTE

- Elasticsearch requires a persistent volume for storing logs and configurations. ***Refer to the official documentation and best practices provided by*** elastic.co for guidance on persistent volume requirements.
- If you already have a log storage system, skip this step.
- The default sizing used in this topic is for a minimal deployment. If the observability stack is used for production, then the replica counts must be 3 for Elastic.

Follow these steps to deploy Elastic Operator to use Elastic Search capabilities of indexing and analyzing all the IDSP logs and view them in the Kibana interface:

1. Deploy Elastic Cloud on Kubernetes (ECK) operator globally or for managing objects under the **logging** namespace. Follow the instructions at https://www.elastic.co/guide/en/cloud-on-k8s/master/k8s-install-helm.html.
   For example:
   a. Load the elastic chart repository:
   ```
   helm repo add elastic https://helm.elastic.co
   helm repo update
   ```
   b. To deploy Elastic Operator without specifying image registry credentials, use this command. This is a sample helm command for a default global deployment of an Elastic operator under the logging namespace:
   NOTE
   The following command can be used by all the IDSP supported cloud providers (including OpenShift) to install the elastic operator.
   ```
   kubectl create ns logging
   helm install elastic-operator elastic/eck-operator -n logging --create-namespace --version=3.2.0
   ```
   c. To deploy Elastic Operator by creating a secret to authenticate to your container-registry to pull images, create a Kubernetes secret of type docker-registry by running the following command:
   ```
   kubectl create secret docker-registry <registry-secret-name> -n logging \
   --docker-server="https://index.docker.io/v2/" \
   --docker-username="<dockerhub-username>" \
   --docker-password="<dockerhub-password>" \
   --docker-email="<your-email-address>"
   ```
   d. Add the preceding created secret to the helm install command that is used to install the Elastic operator by adding the parameter, `--set imagePullSecrets[0].name="<registry-secret-name>"`.
   For example:
   ```
   helm install elastic-operator elastic/eck-operator -n logging \
   --set imagePullSecrets[0].name="<registry-secret-name>" \
   --version=3.2.0
   ```
2. Using the Elastic operator, deploy Elastic Search by creating an Elasticsearch instance.
   a. To set your own password for the elastic default user, create a secret (for example, kibana-user)
   ```
   kubectl create secret generic kibana-user  -n logging \
   --from-literal roles=superuser \
   --from-literal username=kibana \
   --from-literal password=changeme
   ```
   Next, pass the secret (kibana-user) in the Elastic Operator configuration object by uncommenting the line "#-secretName: kibana-user".
   b. To set the registry, pull secret in the Elastic Operator configuration object, uncomment the following lines:
   ```
   #imagePullSecrets:
   #- name: <registry-secret-name>
   ```

An ***example*** of the Elastic Operator configuration object. This example object is not configured for HA and resilience.

```
cat <<EOF | kubectl apply -f -
apiVersion: elasticsearch.k8s.elastic.co/v1
kind: Elasticsearch
metadata:
name: elasticsearch
labels:
app: elasticsearch-master
namespace: logging
spec:
version: 9.2.1
#image: docker.elastic.co/elasticsearch/elasticsearch:9.2.1
http:
service:
 metadata:
nodeSets:
- config:
   node.roles:
     - master
     - data
   node.store.allow_mmap: false
 podTemplate:
   metadata:
     labels:
       app: elasticsearch-master
   spec:
     #imagePullSecrets:
     #- name: <registry-secret-name>
     containers:
     - name: elasticsearch
       resources:
         requests:
           memory: 4Gi
           cpu: 1
         limits:
           memory: 4Gi
           cpu: 1
 volumeClaimTemplates:
 - metadata:
     name: elasticsearch-data
   spec:
     #storageClassName: fast
     accessModes:
     - ReadWriteOnce
     resources:
       requests:
         storage: 20Gi
 name: default
 count: 1
volumeClaimDeletePolicy: DeleteOnScaledownOnly
auth:
fileRealm:
```

```
#To create new elastic users, see: https://www.elastic.co/guide/en/cloud-on-k8s/master/k8s-users-and-roles.html
#    - secretName: kibana-user ##uncomment if you want to create users/set password in elastic.
EOF
```

c. Verify that all persistent volume claims in the logging namespace are showing the *STATUS* as *Bound,* by running this command:

```
kubectl get pvc -n logging
```

The following image is a sample displaying a PVC with the "*STATUS"* reported as "*Bound"*:

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS
elasticsearch-data-elasticsearch-es-default-0	Bound	pvc-d81aabb1-f3fc-435a-a074-e20b3a2f1874	20Gi	RWO	standard

d. Verify that the pods are Ready (1/1) and in Running State by running this command:

```
kubectl get pods -n logging
```

NOTE
If you want to start with an empty data set after uninstalling elastic, then delete all the Elastic's PVC using this command: `kubectl delete pvc <elastic-pvc-name> -n logging`

3. To set a data retention policy for the log and audit messages in Elastic Search, see the steps that are described in the Setting up data retention policy for Elastic Search section.
4. Deploy Kibana by creating a Kibana cluster instance using the deployed Elastic operator.
   a. To add your own trusted certificate signed by the Certificate Authority (CA), run this command:

```
kubectl create secret tls logging-general-tls \
--cert "${CERTFILE}" \
--key "${KEYFILE}" -n logging
```

If you do not have a certificate, you can skip creating the "logging-general-tls" secret. The ingress-nginx ingress controller uses its built-in self-signed certificate if the "logging-general-tls" secret is not provided.

   b. To set the registry, pull secret in the Elasticsearch configuration object, uncomment the following lines:

```
#imagePullSecrets:
#- name: <registry-secret-name>
```

An ***example*** of the Kibana configuration object. The following example requires that the environment variable DOMAIN is set.

```
cat <<EOF | kubectl apply -n logging -f -
apiVersion: kibana.k8s.elastic.co/v1
kind: Kibana
metadata:
name: kibana
namespace: logging
spec:
version: 9.2.1
#image: docker.elastic.co/kibana/kibana:9.2.1
elasticsearchRef:
name: elasticsearch
namespace: logging
serviceName: elasticsearch-es-http
http:
service:
 spec:
   type: ClusterIP
tls:
 selfSignedCertificate:
   subjectAltNames:
   - dns: kibana.${DOMAIN}
podTemplate:
```

```
    metadata:
     labels:
       app: kibana
    spec:
     #imagePullSecrets:
     #- name: <registry-secret-name>
     containers:
       - name: kibana
         resources:
           requests:
             memory: 1Gi
             cpu: 0.5
           limits:
             memory: 2Gi
             cpu: 1
    count: 1
    EOF
```

c. Verify that the pods are Ready (1/1) and in Running State by running this command:

```
kubectl get pods -n logging
```

The following image is a sample displaying the logging pods with the STATUS-Running:

```
NAME                          READY   STATUS    RESTARTS
elastic-operator-0            1/1     Running   4 (12h ago)
elasticsearch-es-default-0    1/1     Running   0
kibana-kb-868977bf88-skvjw    1/1     Running   0
```

5. Expose the Kibana service. The default deployment does not create an Ingress object to expose the Kibana service. Use the following command to create an Ingress object. The following example requires that the environment variable "DOMAIN" is configured.

```
cat <<EOF | kubectl apply -n logging -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
name: kibana
namespace: logging
annotations:
nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
spec:
ingressClassName: nginx
rules:
- host: kibana.${DOMAIN}
 http:
   paths:
     - path: /
       pathType: Prefix
       backend:
         service:
           name: kibana-kb-http
           port:
             number: 5601
tls:
- hosts:
 - kibana.${DOMAIN}
```

```
  secretName: logging-general-tls
EOF
```

6. Connect to Kibana.
   a. Ensure that the Kibana FQDN defined in the ingress object is registered along with the ingress IP in your DNS.
   b. From your browser, go to the Kibana URL. For example, https://kibana.${DOMAIN}/
   c. If you created a Kibana user as specified above, use it to log in to Kibana.
   d. Otherwise, log in using the default user: ***elastic*** with the password that can be obtained with the following command:

```
kubectl get secret -n logging elasticsearch-es-elastic-user -o=jsonpath='{.data.elastic}' | base64 --
decode; echo
```

Publishing Kibana Service IP

Perform these steps to ensure that the FQDN of the Kibana service is resolvable to the Ingress IP Address. This could be done by adding its FQDN and its Ingress IP Address to your Cloud DNS or your local DNS.

1. The FQDNs of the Kibana service are:
   – Kibana.<DOMAIN>

   NOTE
   Replace the <DOMAIN>, with the name of the domain as mentioned in the environment section.
2. Use the following command to get the Ingress IP Address:

```
kubectl get ingress -n <NAMESPACE>
```

   NOTE
   If you are using a BareMetal cluster that is fronted by HAPROXY, then this IP Address would be of the machine running HAPROXY. Otherwise, the IP Address is the external IP Address of your Ingress service as returned by the following command:

```
kubectl get svc -n ingress ingress-nginx-controller
```

3. Update the DNS with the preceding host name and IP Address. If you are unable to update the DNS, and want to access the services from your local machine, add the following line to your "**/etc/host**" file:

```
<ingress-ip> kibana.<DOMAIN>
```

   NOTE

   - Replace the <DOMAIN>, with the name of the domain as mentioned in the environment section.
   - Replace the <ingress-ip> with the IP Address that you retrieved in the previous step.

Next Step

Deploying Prometheus

Reference Links

Setting up Data Retention Policy in Elastic Search

Setting-Up Data Retention Policy in Elastic Search

This topic outlines the process of setting up a data retention policy in Elasticsearch to manage log indexes and prevent disk space issues. It covers creating lifecycle policies and index templates for different log types (ssp-audit, ssp-log, ssp-tp-log) to ensure automated cleanup and efficient storage.

The IDSP enclave deployment instructions and the IDSP infra deployment steps provide examples of configuring fluent-bit to use elastic search as a log storage repository. Without proper maintenance of these indexes, the system can run out of disk space which would cause it to fail and not start. To prevent this, a lifecycle policy has to be defined in elastic

which would take care of data retention and would clean up these indexes. The log retention policy that is defined in this section is configured to leave logs in a "hot" state (readily available and indexed for fast search) for two days before being deleted.

Defining log retention in Elastic Search requires the creation of two objects (_ilm/policy and _index_template) per log type. Notice that the following example creates a retention policy for 3 types of logs, ssp-audit (with a retention policy of 30 days), ssp-log (with a retention policy of 10 days), and ssp-tp-log (third-party log - with a retention policy of 7 days). These values can be adjusted as needed.

1. Create a Lifecycle Policy:

   NOTE
   The lifecycle policy that is created in this example is shown for all the three types of logs; ssp_audit, ssp_log, and ssp_tp_log.

```
kubectl exec -n logging elasticsearch-es-default-0 -c elasticsearch -- curl --insecure -u "elastic:
$(kubectl get secret -n logging elasticsearch-es-elastic-user -o=jsonpath='{.data.elastic}' | base64 --
decode; echo)" -s -XPUT \
 "https://elasticsearch-es-http.logging.svc:9200/_ilm/policy/ssp_audit_lifecycle_policy" \
 -H 'Content-Type: application/json' \
 -d '{
"policy": {
"phases": {
 "hot": {
   "min_age": "0ms",
   "actions": {
     "set_priority": {
       "priority": 100
     },
     "rollover": {
       "max_age": "1d",
       "max_primary_shard_size": "50gb"
     }
   }
 },
 "delete": {
   "min_age": "30d",
   "actions": {
     "delete": {
       "delete_searchable_snapshot": true
     }
   }
 }
}
}
}'

kubectl exec -n logging elasticsearch-es-default-0 -c elasticsearch -- curl --insecure -u "elastic:
$(kubectl get secret -n logging elasticsearch-es-elastic-user -o=jsonpath='{.data.elastic}' | base64 --
decode; echo)" -s -XPUT \
 "https://elasticsearch-es-http.logging.svc:9200/_ilm/policy/ssp_log_lifecycle_policy" \
 -H 'Content-Type: application/json' \
 -d '{
"policy": {
"phases": {
 "hot": {
```

```
    "min_age": "0ms",
    "actions": {
      "set_priority": {
        "priority": 100
      },
      "rollover": {
        "max_age": "1d",
        "max_primary_shard_size": "50gb"
      }
    }
  },
  "delete": {
    "min_age": "10d",
    "actions": {
      "delete": {
        "delete_searchable_snapshot": true
      }
    }
  }
}
}
}'

kubectl exec -n logging elasticsearch-es-default-0 -c elasticsearch -- curl --insecure -u "elastic:
$(kubectl get secret -n logging elasticsearch-es-elastic-user -o=jsonpath='{.data.elastic}' | base64 --
decode; echo)" -s -XPUT \
 "https://elasticsearch-es-http.logging.svc:9200/_ilm/policy/ssp_tp_log_lifecycle_policy" \
 -H 'Content-Type: application/json' \
 -d '{
"policy": {
"phases": {
 "hot": {
   "min_age": "0ms",
   "actions": {
     "set_priority": {
       "priority": 100
     },
     "rollover": {
       "max_age": "1d",
       "max_primary_shard_size": "50gb"
     }
   }
 },
 "delete": {
   "min_age": "7d",
   "actions": {
     "delete": {
       "delete_searchable_snapshot": true
     }
   }
 }
}
}
```

```
}'
```

2. Create an Index Template that associates the Lifecycle Policy with the index:

   NOTE
   The Index Template created in this step is shown for all the three types of logs; ssp_audit, ssp_log, and ssp_tp_log.

```
kubectl exec -n logging elasticsearch-es-default-0 -c elasticsearch -- curl --insecure -u"elastic:
$(kubectl get secret -n logging elasticsearch-es-elastic-user -o=jsonpath='{.data.elastic}' | base64 --
decode; echo)" \
-s -XPUT "https://elasticsearch-es-http.logging.svc:9200/_index_template/ssp_audit" \
-H 'Content-Type: application/json' \
-d '{
"template": {
"settings": {
  "index": {
    "lifecycle": {
      "name": "ssp_audit_lifecycle_policy",
      "rollover_alias": "ssp_audit_alias"
    }
  }
},
"aliases": {
  "ssp_audit_alias": {
    "is_write_index": true
  }
}
},
"index_patterns": [
"ssp_audit*"
],
"composed_of": []
}'

kubectl exec -n logging elasticsearch-es-default-0 -c elasticsearch -- curl --insecure -u"elastic:
$(kubectl get secret -n logging elasticsearch-es-elastic-user -o=jsonpath='{.data.elastic}' | base64 --
decode; echo)" \
-s -XPUT "https://elasticsearch-es-http.logging.svc:9200/_index_template/ssp_log" \
-H 'Content-Type: application/json' \
-d '{
"template": {
 "settings": {
   "index": {
     "lifecycle": {
       "name": "ssp_log_lifecycle_policy",
       "rollover_alias": "ssp_log_alias"
     }
   }
 },
 "aliases": {
   "ssp_log_alias": {
     "is_write_index": true
   }
 }
```

```
},
"index_patterns": [
 "ssp_log*"
],
"composed_of": []
}'

kubectl exec -n logging elasticsearch-es-default-0 -c elasticsearch -- curl --insecure -u"elastic:
$(kubectl get secret -n logging elasticsearch-es-elastic-user -o=jsonpath='{.data.elastic}' | base64 --
decode; echo)" \
-s -XPUT "https://elasticsearch-es-http.logging.svc:9200/_index_template/ssp_tp_log" \
-H 'Content-Type: application/json' \
-d '{
"template": {
  "settings": {
    "index": {
      "lifecycle": {
        "name": "ssp_tp_log_lifecycle_policy",
        "rollover_alias": "ssp_tp_log_alias"
      }
    }
  },
  "aliases": {
    "ssp_tp_log_alias": {
      "is_write_index": true
    }
  }
},
"index_patterns": [
  "ssp_tp_log*"
],
"composed_of": []
}'
```

3. Create an empty initial (writable) index with the pattern: <ssp_audit-messaegs-{now/d}-000001>

```
kubectl exec -n logging elasticsearch-es-default-0 -c elasticsearch -- curl --insecure -u"elastic:
$(kubectl get secret -n logging elasticsearch-es-elastic-user -o=jsonpath='{.data.elastic}' | base64 --
decode; echo)" \
-s -XPUT "https://elasticsearch-es-http.logging.svc:9200/%3Cssp_audit-messaegs-%7Bnow%2Fd%7D-000001%3E" \
-H 'Content-Type: application/json' \
-d '{}'

kubectl exec -n logging elasticsearch-es-default-0 -c elasticsearch -- curl --insecure -u"elastic:
$(kubectl get secret -n logging elasticsearch-es-elastic-user -o=jsonpath='{.data.elastic}' | base64 --
decode; echo)" \
-s -XPUT "https://elasticsearch-es-http.logging.svc:9200/%3Cssp_log-messaegs-%7Bnow%2Fd%7D-000001%3E" \
-H 'Content-Type: application/json' \
-d '{}'

kubectl exec -n logging elasticsearch-es-default-0 -c elasticsearch -- curl --insecure -u"elastic:
$(kubectl get secret -n logging elasticsearch-es-elastic-user -o=jsonpath='{.data.elastic}' | base64 --
decode; echo)" \
```

```
-s -XPUT "https://elasticsearch-es-http.logging.svc:9200/%3Cssp_tp_log-messaegs-%7Bnow%2Fd%7D-000001%3E" \
-H 'Content-Type: application/json' \
-d '{}'
```

4. Elastic Search requires the creation of a new index to pick the new Index-Template. To retain the old logs, configure Fluent Bit to append the date to the index name. If you have created Kibana's dataview using the <index_name>* pattern, it picks both old and new indexes for log viewing. After about three weeks, manual deletion of the old indexes is required using the Kibana Index Management section. The following example shows the fluent-bit configuration's output:

```
[OUTPUT]
Name  es
Match *
Host  elasticsearch_host
Port  9200
Index ssp_audit_alias
```

Deploying Prometheus

Includes links to deploy Prometheus and Grafana in OpenShift and in Non-Openshift Clusters.

Symantec Identity Security Platform services expose performance monitoring data in a Prometheus metrics format. The endpoint exposing the Prometheus metrics is indicated using the IDSP service's label. If you have a Prometheus Operator deployed in your cluster, IDSP creates the proper serviceMonitor object to communicate this information to Prometheus.

If you plan to use another tool to scrape the Prometheus endpoints exposed by IDSP services, skip the deployment steps, and refer to the following information:

- Every service with the label `ssp-prometheus-metrics=actuator` exposes the endpoint `/actuator/prometheus` over TLS (port-443, HTTPS).
- Every service with the label `ssp-prometheus-metrics=metrics` exposes the endpoint `/metrics` over TLS (port-443, HTTPS).

Otherwise, follow the steps to deploy/configure Prometheus and Grafana outlined based on your cluster type.

See the following links based on your cluster requirement.

- Deploying Prometheus in Non-Openshift Cluster
- Deploying Prometheus in Openshift Cluster

Deploying Prometheus and Grafana in a Non-Openshift Cluster

Learn the steps to deploy Prometheus, AlertManager, and Grafana in Symantec Identity Security Platform using a Non-OpenShift Cluster to view performance data in Prometheus metrics format.

This topic includes the following sections:

- Deploying Prometheus
- Deploying Grafana
- Publishing the Grafana Service IP

> **NOTE**
> Prometheus requires a persistent volume to store its time series database containing metrics data. ***Refer to the official documentation and best practices provided by*** **Prometheus** for guidance on persistent volume usage.

Deploying Prometheus

Follow these steps to deploy Prometheus in a Non-OpenShift Cluster:

1. Create a monitoring namespace.
   ```
   kubectl create ns monitoring
   ```
2. Load the bitnami charts repository.

   NOTE

   - Please be aware that as of August 28th, 2025, Bitnami is archiving its OCI registry of charts and images on Docker Hub. Most older or versioned tags and non-hardened images are being migrated to the Bitnami Legacy repository (docker.io/bitnamilegacy) and will no longer receive updates. Refer to Bitnami official documentation for more information.
   - Based on these changes, the deployment command in step 3 (iii) has been updated.

   ```
   helm repo add bitnami https://charts.bitnami.com/bitnami
   helm repo update
   ```
3. Deploy Prometheus Operator to set up Prometheus, AlertManager, and Grafana.
   a. Deploy Prometheus under the monitoring namespace.
      a. If you are required to authenticate to your container-registry to pull images, then create a Kubernetes secret of type docker-registry as follows:
         ```
         kubectl create secret docker-registry <registry-secret-name> -n monitoring \
         --docker-server="https://index.docker.io/v2/" \
         --docker-username="<dockerhub-username>" \
         --docker-password="<dockerhub-password>" \
         --docker-email="<your-email-address>"
         ```
         Next, add this secret by adding the following parameters to the helm install command that is used to install the Prometheus operator.
         ```
         --set global.imagePullSecrets[0].name="<registry-secret-name>"
         --set grafana.image.pullSecrets[0]="<registry-secret-name>"
         ```
      b. When enabling TLS, alert manager uses a TLS secret by the name <alertmanager.ingress.hostname>**-tls,** if this secret is not found, the ingress controller uses its own self-signed certificate. You can also have the chart create this secret by passing the certificates using ***alertmanager.ingress.secrets.certificate*** and ***alertmanager.ingress.secrets.key***, or use ***alertmanager.ingress.selfSigned=true*** to have helm generate new self-signed certificates. For example,
         ```
         kubectl create secret tls alertmanager.${DOMAIN}-tls --cert ${CERTFILE} --key ${KEYFILE} -n
          monitoring
         ```
      c. Deploy the prometheus-operator under the monitoring namespace.

         ```
         helm install prometheus-operator bitnami/kube-prometheus -n monitoring \
         --set alertmanager.ingress.enabled=true \
         --set alertmanager.ingress.ingressClassName=nginx \
         --set alertmanager.ingress.hostname=alertmanager.${DOMAIN} \
         --set alertmanager.ingress.tls=true \
         --set prometheus.persistence.enabled=true \
         --set alertmanager.persistence.enabled=true \
         --set prometheus.scrapeInterval=1m \
         --set prometheus.evaluationInterval=1m \
         --set blackboxExporter.image.repository=bitnamilegacy/blackbox-exporter \
         --set kube-state-metrics.image.registry=bitnamilegacy \
         --set kube-state-metrics.image.repository=kube-state-metrics \
         --set operator.image.repository=bitnamilegacy/prometheus-operator \
         --set prometheus.image.repository=bitnamilegacy/prometheus \
         ```

```
--set alertmanager.image.repository=bitnamilegacy/alertmanager \
--set node-exporter.image.registry=bitnamilegacy \
--set node-exporter.image.repository=node-exporter \
--set global.security.allowInsecureImages=true \
--version=11.3.10
```

d. Verify that all persistent volume claims are bound for this namespace and that the pods are ready.

```
kubectl get pvc -n monitoring
kubectl get pods -n monitoring
```

e. Post the deployment of the Prometheus Operator, configure the resource allocation for your Prometheus Operator deployment by selecting an environment size (Small, Medium, Large, or Custom), based on which the CPU and memory limits are applied using a kubectl patch..

```
"Select your environment size:"
 "  1) Small    (< 100 targets)   - 1Gi/2Gi memory"
 "  2) Medium   (100-500 targets) - 2Gi/4Gi memory  [RECOMMENDED]"
 "  3) Large    (500+ targets)    - 4Gi/8Gi memory"
 "  4) Custom   (specify your own values)"
 ""
SIZE_CHOICE=2

case $SIZE_CHOICE in
    1)
        CPU_REQ="250m"
        CPU_LIMIT="1000m"
        MEM_REQ="1Gi"
        MEM_LIMIT="2Gi"
        ;;
    2)
        CPU_REQ="500m"
        CPU_LIMIT="2000m"
        MEM_REQ="2Gi"
        MEM_LIMIT="4Gi"
        ;;
    3)
        CPU_REQ="1000m"
        CPU_LIMIT="4000m"
        MEM_REQ="4Gi"
        MEM_LIMIT="8Gi"
        ;;

    4) Choose your own CPU and MEM REQ and LIMIT.
```

Apply this Kubectl patch:

```
kubectl patch prometheus $PROMETHEUS_NAME -n $NAMESPACE --type=merge -p "
{
  \"spec\": {
    \"resources\": {
      \"requests\": {
        \"cpu\": \"$CPU_REQ\",
        \"ephemeral-storage\": \"8Gi\",
        \"memory\": \"$MEM_REQ\"
      },
      \"limits\": {
        \"cpu\": \"$CPU_LIMIT\",
```

```
            \"ephemeral-storage\": \"8Gi\",
            \"memory\": \"$MEM_LIMIT\"
          }
        }
      }
    }"
```

Deploying Grafana

1. Deploy Grafana.
   a. If you want to add your own trusted certificate signed by the Certificate Authority (CA), run this command:
   ```
   kubectl create secret tls monitoring-general-tls \
   --cert "${CERTFILE}" \
   --key "${KEYFILE}" -n monitoring
   ```
   If you do not have a certificate, you can skip this step as the Monitoring ingress uses the self-signed certificate.
   b. To change the default password, update the value of the ***grafana.config.security.admin_password*** parameter in the grafana-operator helm install command.
   c. Deploy the Grafana operator using the following helm command. The Grafana Operator chart creates and deploys a Grafana instance.
   ```
   helm install grafana-operator bitnami/grafana-operator -n monitoring \
   --set operator.containerSecurityContext.readOnlyRootFilesystem=true \
   --set operator.image.repository=bitnamilegacy/grafana-operator \
   --set grafana.image.repository=bitnamilegacy/grafana \
   --set grafana.config.security.admin_password=prom-operator \
   --set grafana.ingress.enabled=true \
   --set grafana.ingress.ingressClassName=nginx \
   --set grafana.ingress.host=grafana.${DOMAIN} \
   --set grafana.ingress.tls=true \
   --set grafana.ingress.tlsSecret=monitoring-general-tls \
   --set-string grafana.labels.dashboards="ssp-grafana" \
   --version=4.9.37
   ```

2. Configure Grafana to use the local Prometheus by creating a Grafana DataSource object.
   Run this command to create a Grafana datasource object on the native cloud providers.
   ```
   cat <<EOF | kubectl apply -n monitoring -f -
   apiVersion: grafana.integreatly.org/v1beta1
   kind: GrafanaDatasource
   metadata:
     name: prometheus-datasource
   spec:
     datasource:
       name: Prometheus
       type: prometheus
       access: proxy
       editable: true
       isDefault: true
       url: 'http://prometheus-operator-kube-p-prometheus.monitoring.svc:9090'
       jsonData:
         timeInterval: 5s
     instanceSelector:
       matchLabels:
         dashboards: "ssp-grafana"
   ```

```
EOF
```

a. To install the IDSP dashboard through the Grafana operator, run this command:

```
cat <<EOF | kubectl apply -n monitoring -f -
apiVersion: grafana.integreatly.org/v1beta1
kind: GrafanaDashboard
metadata:
  name: ssp-monitoring-dashboard
spec:
  datasources:
    inputName: "DS_PROMETHEUS"
    datasourceName: "Prometheus"
  instanceSelector:
    matchLabels:
      dashboards: "ssp-grafana"
  grafanaCom:
    id: 25306
EOF
```

3. Delete the existing Grafana Pod so that the monitoring namespace configuration is read in.

```
kubectl delete pod -l app.kubernetes.io/name=grafana-operator -n monitoring
```

4. Verify your Prometheus/Grafana deployment:
   From your browser, you can now access Grafana using the Ingress URL that you have configured. For example: https://grafana.openshift.clusters.demo.dev-ssp.com
   – Default user: admin
   – Default Password: prom-operator

Publishing Grafana Service IP

Perform these steps to ensure that the FQDN of the Grafana service is resolvable to the Ingress IP Address. This could be done by adding its FQDN and its Ingress IP Address to your Cloud DNS or your local DNS.

1. The FQDNs of the Grafana service are:
   – Grafana.<DOMAIN>

   NOTE
   Replace the <DOMAIN>, with the name of the domain as mentioned in the environment section.
2. The Ingress IP Address can be retrieved from the "Address" column of the Grafana Ingress object when using the following command:

```
kubectl get ingress -n monitoring
```

3. Update the DNS with the preceding host names and IP Address. If you are unable to update the DNS, and want to access the services from your local machine, add the following line to your "**/etc/host**" file:

```
<ingress-ip> grafana.<DOMAIN>
```

   NOTE

   - Replace the <DOMAIN>, with the name of the domain as mentioned in the environment section.
   - Replace the <ingress-ip> with the IP Address that you retrieved in the previous step.

Next Step

Deploying IDSP

Deploying Prometheus and Grafana in an OpenShift Cluster

Learn the steps to deploy Prometheus and Grafana in Symantec Identity Security Platform using an OpenShift Cluster to view performance data in Prometheus metrics format.

This topic includes the following sections:

- Deploying Prometheus
- Deploying Grafana and AlertManager Service IP
- Publishing the Grafana Service IP

> **NOTE**
> In an OpenShift cluster, both Prometheus Operator and Grafana may require persistent volumes, depending on the specific deployment and usage requirements. While both Prometheus Operator and Grafana can function without persistent volumes, using them can enhance data reliability and availability. ***Refer to the official documentation and best practices provided by OpenShift and the Prometheus, and Grafana tools*** for guidance on persistent volume usage in your specific deployment scenario.

Deploying Prometheus

Follow these steps to deploy Prometheus in an OpenShift Cluster:

1. Deploy Prometheus Operator to set up Prometheus (including AlertManager), and Grafana on OpenShift.
   a. Given that OpenShift already includes a Prometheus operator deployment, there is no requirement to perform an additional deployment. However, it is necessary to grant the required permissions to Prometheus to enable it to scrape metrics from the IDSP (SSP) namespace.
   b. Enable Prometheus to Scrape IDSP (SSP) Services' Information:

      > **NOTE**
      > Ensure that the environment variable, NAMESPACE is set to the name of the Kubernetes namespace where IDSP (SSP) was deployed. If IDSP is not deployed yet, you can perform this step anytime after the IDSP (SSP) namespace is created.

```
kubectl label ns ${NAMESPACE} openshift.io/cluster-monitoring="true"

cat <<EOF | kubectl apply -n ${NAMESPACE} -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: prometheus-k8s
rules:
- apiGroups:
  - ""
  resources:
  - services
  - endpoints
  - pods
  verbs:
  - get
  - list
  - watch

---

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
```

```
  metadata:
    name: prometheus-k8s
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: Role
    name: prometheus-k8s
  subjects:
  - kind: ServiceAccount
    name: prometheus-k8s
    namespace: openshift-monitoring
  EOF
```

c. Perform the following steps to ensure that OpenShift's "monitoring for user-defined projects" is enabled:
   a. Edit the cluster-monitoring-config configmap.
      ```
      kubectl -n openshift-monitoring edit configmap cluster-monitoring-config
      ```
   b. Enable enableUserWorkload by adding enableUserWorkload: true to the config.yaml. For example:
      ```
      data:
       config.yaml: |
          enableUserWorkload: true
      ```
      NOTE
      If you get an error "Error from server (NotFound): configmaps "cluster-monitoring-config" not found" (configmap not found), create ConfigMap using the following command:
      ```
      cat <<EOF | kubectl create -f -
        apiVersion: v1
        kind: ConfigMap
        metadata:
          name: cluster-monitoring-config
          namespace: openshift-monitoring
        data:
          config.yaml: |
            enableUserWorkload: true
      EOF
      ```
   Next step is to deploy Grafana.

Deploying Grafana and AlertManager Service IP

1. Create a monitoring namespace.
   ```
   kubectl create ns monitoring
   ```
2. Load the bitnami charts repository.
   ```
   helm repo add bitnami https://charts.bitnami.com/bitnami
   helm repo update
   ```
3. Deploy Grafana.
   a. If you want to add your own trusted certificate signed by the Certificate Authority (CA), run this command:
      ```
      kubectl create secret tls monitoring-general-tls \
      --cert "${CERTFILE}" \
      --key "${KEYFILE}" -n monitoring
      ```
      If you do not have a certificate, you can skip this step as the Monitoring ingress uses the self-signed certificate.
   b. To change the default password, update the value of the ***grafana.config.security.admin_password*** parameter in the grafana-operator helm install command.
   c. Deploy the Grafana operator. Note that the Grafana Operator chart will also create and deploy a grafana instance.
      a. Set the following environment variables.

```
read -d '/' -r GRAFANA_UID < <(kubectl get ns monitoring -o
 jsonpath='{.metadata.annotations.openshift\.io/sa\.scc\.uid-range}') || GRAFANA_UID=1000650000
read -d '/' -r GRAFANA_GID < <(kubectl get ns monitoring -o
 jsonpath='{.metadata.annotations.openshift\.io/sa\.scc\.supplemental-groups}') ||
 GRAFANA_GID=1000650000

export GRAFANA_FSGROUP=${GRAFANA_GID}
```

b. Using the following command, deploy Grafana:

```
helm install grafana-operator bitnami/grafana-operator -n monitoring \
--set operator.containerSecurityContext.readOnlyRootFilesystem=true \
--set grafana.image.repository=bitnami/grafana \
--set grafana.config.security.admin_password=prom-operator \
--set grafana.ingress.enabled=true \
--set grafana.ingress.ingressClassName=nginx \
--set grafana.ingress.host=grafana.${DOMAIN} \
--set grafana.ingress.tls=true \
--set grafana.ingress.tlsSecret=monitoring-general-tls \
--set operator.containerSecurityContext.runAsUser=${GRAFANA_UID} \
--set operator.containerSecurityContext.runAsGroup=${GRAFANA_GID} \
--set operator.podSecurityContext.fsGroup=${GRAFANA_FSGROUP} \
--set grafana.containerSecurityContext.runAsUser=${GRAFANA_UID} \
--set grafana.podSecurityContext.runAsUser=${GRAFANA_UID} \
--set grafana.podSecurityContext.fsGroup=${GRAFANA_FSGROUP} \
--version=2.7.3
```

4. Configuring Grafana to use the local Prometheus by creating a Grafana DataSource object.
   Follow these instructions to create a Grafana datasource object on OpenShift:
   a. **Use this command for OpenShift version 4.16**
      Get the token from the openshift-user-workload-monitoring secret (see obtain the OpenShift's "prometheus-user-workload-token" bearer token from the "openshift-user-workload-monitoring" namespace secret)

```
SECRET_NAME=`kubectl get secret -n openshift-user-workload-monitoring | grep  prometheus-user-workload-
token | head -n 1 | awk '{print $1 }'`
TOKEN=`echo $(kubectl get secret $SECRET_NAME -n openshift-user-workload-monitoring -o json | jq -r
 '.data.token') | base64 -d`
```

      Use this command for OpenShift version 4.17 and higher as the secret will no longer be available
      You need a user (or a service account) in your cluster which has the "cluster-monitoring-view" cluster role granted. Acquire a bearer token for that user. Login as the user and use the following command to generate a bearer token for the user. (This is a short-lived token which needs to be refreshed periodically) :

```
TOKEN=$(oc whoami -t)
```

   b. Create the Grafana datasource object.

```
cat <<EOF | kubectl apply -n monitoring -f -
  apiVersion: integreatly.org/v1alpha1
  kind: GrafanaDataSource
  metadata:
    name: grafana-datasource
  spec:
    name: grafana-datasource.yaml
    datasources:
      - name: Prometheus
        type: prometheus
        version: 1
        access: proxy
        editable: true
```

```
            isDefault: true
            url: 'https://prometheus-k8s.openshift-monitoring.svc:9091'
            jsonData:
              timeInterval: 5s
              tlsSkipVerify: true
              httpHeaderName1: 'Authorization'
            secureJsonData:
              httpHeaderValue1: "Bearer ${TOKEN}"
    EOF
```

c. Delete the existing Grafana Pod so that the monitoring namespace configuration is read in.

```
kubectl delete pod -l app=grafana -n monitoring
```

5. Expose the Grafana service.
   When running on OpenShift, the default deployment creates a Route object. If you want to expose Grafana through your nginx ingress controller, create an ingress object use the following command:

```
cat <<EOF | kubectl apply -n monitoring -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: grafana
  namespace: monitoring
spec:
  ingressClassName: nginx
  rules:
    - host: grafana.${DOMAIN}
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: grafana-service
                port:
                  number: 3000
  tls:
    - hosts:
      - grafana.${DOMAIN}
      secretName:  monitoring-general-tls
EOF
```

6. Verify your Prometheus/Grafana deployment:
   From your browser, you can now access Grafana using the Ingress URL that you have configured. For example: https://grafana.openshift.clusters.demo.dev-ssp.com
   – Default user: admin
   – Default Password: prom-operator

Publishing the Grafana Service IP

Perform these steps to ensure that the FQDN of the Grafana service is resolvable to the Ingress IP Address. This could be done by adding its FQDN and its Ingress IP Address to your Cloud DNS or your local DNS.

1. The FQDNs of the Grafana service is:
   – grafana.<DOMAIN>

NOTE
Replace the <DOMAIN>, with the name of the domain as mentioned in the environment section.

2. Exposing the Grafana Service:
   – If you choose to expose the Grafana Service using an ***NGINX Ingress Controller***, the Ingress IP Address can be retrieved from the "Address" column of the Grafana Ingress object when using the following command:
   ```
   kubectl get ingress -n monitoring
   ```
   – If you choose to expose the Grafana Service using ***OpenShift Ingress Controller (using route)***, the IP Address is displayed under the "EXTERNAL IP" column of your OpenShift Ingress service as returned by the following command:
   ```
   kubectl get svc -n openshift-ingress
   ```
3. Update the DNS with the above host names and IP Address. If you are unable to update the DNS, and want to access the services from your local machine, add the following line to your "**/etc/host**" file:
   ```
   <openshift-ingress-ip> grafana.<DOMAIN>
   ```

   NOTE
   - Replace the <DOMAIN>, with the name of the domain as mentioned in the environment section.
   - Replace the <ingress-ip> with the IP Address that you retrieved in the previous step.

Next Step

Deploying IDSP

Deploying Symantec Identity Security Platform

Learn how to deploy Symantec Identity Security Platform solution on a Kubernetes environment.

This topic provides a streamlined deployment guide with the following sections:

- Understanding Deployment Modes
- Configuring the Environment Variables
- Configuring the Helm Repo
- Quick Deployment Summary
- Verifying the Deployment
- Retrieving Initial Security Credentials From Kubernetes Secrets
- Next Step - Configuring Administrative Access

Understanding the Deployment Modes

The IDSP offers two distinct modes of deployment to cater to different operational needs:

- **"custom" Mode (Default mode)**:
  – Provides flexibility to configure each service's pod count and allocate CPU and memory resources to suite your production environment's requirements. To learn about the IDSP deployment guidelines, see Sizing the Deployment.
  – You can override the default "values.yaml" file with your own specifications. If this file is provided, the deployment uses the custom values from this file.
  – If no override file is provided, the deployment uses the default values from the standard "values.yaml" file.
- **"demo" Mode**:

- Designed for Proof-of-Concept (POC) scenarios, allowing customers to test functionalities before committing to full-scale development environment deployments.
- In this mode, the pod count is fixed to 1 for each service.
- CPU and memory resources per pod are minimized, and database sizes are optimized, specifically for Proof-of-Concept (POC) environments.
  The following table shows how the resources are utilized optimally in this mode.

Table 32: Resources Table

Resources	CPU	Memory
limits	600m	2Gi
requests	400m	1Gi

- This mode should not be used for production environment.
- To deploy IDSP in demo mode, the following parameters must be passed along with the IDSP Helm deployment command:

```
--set ssp.deployment.size="demo" \
--set hazelcast-enterprise.cluster.memberCount=1
```

Configuring the Environment Variables

Once you installed the software that has been mentioned in the earlier section, you must set up your system environment from where you are managing your cluster.

These commands can be executed from your on-premise work station or cloud shell using bash shell:

1. Configure environment variables to point to your DNS domain.

```
export PREFIX="<PREFIX>"
export DOMAIN="<DOMAIN>"
export SSP_FQDN="${PREFIX}.${DOMAIN}"
```

   For **Ingress Nginx** deployment, configure the following environment variable:

```
export SSP_MGMT_FQDN="mgmt-${SSP_FQDN}"
```

   NOTE
   The mgmt-${SSP_FQDN} is the default FQDN. This could be overridden by specifying the management hostname in the parameter, "ssp.ingress.mgmt.host".

   For **OpenShift Routes** deployment, configure the following environment variable:

```
export SSP_MGMT_FQDN="system.${SSP_FQDN}"
```

   NOTE
   For example,

   - <PREFIX> can be replaced by "myssp" and,
   - <DOMAIN> can be replaced by "myiam.example.com"

   Resulting in FQDN for IDSP as - "myssp.myiam.example.com". This FQDN is used to access the IDSP service.

2. Configure the environment variables to point to the HELM release name and Kubernetes namespace. These names enable decoupling between deployments.

```
export RELEASENAME="<RELEASENAME>"
export NAMESPACE="<NAMESPACE>"
```

   Where,

- The ***Release Name*** is a Lifecycle Management (LCM) construct, representing a versionable instance of a deployment with specific configuration that can be upgraded, rolled back, and so on.
- The ***Namespace*** is a Kubernetes construct representing an operational partition aggregating specific permissions, seclists, network rules, quotas, and so on.
- Both names can be given the same value such as "ssp-dev", "ssp-prod".

3. Set the environment variables to indicate the HELM REPO and HELM REPO URL for IDSP helm charts deployment. For example,

```
export HELM_REPO='ssp_helm_charts'
export HELM_REPO_URL=https://ssp_helm_charts.storage.googleapis.com
```

4. If you have a wildcard certificate from a trusted Certificate Authority (CA), configure the environment variables to point to the key and cert files.

```
export KEYFILE="<keyfile>"
export CERTFILE="<certfile>"
```

Configuring Helm Chart Repo

These set of commands use the environment variable values created in the previous section.

1. Load the IDSP pods with the given release name and namespace (this may take several minutes).
   a. Create the IDSP namespace.
      ```
      kubectl create ns ${NAMESPACE}
      ```
   b. Configure the IDSP Helm chart repo.
      ```
      helm repo add "${HELM_REPO}" "${HELM_REPO_URL}"
      ```
   c. Get the latest Helm chart updates.
      ```
      helm repo update
      ```
   d. Check for all the available versions of IDSP Helm charts.
      ```
      helm search repo ssp_helm_charts --versions
      ```
      NOTE
      This command displays all the charts available in the repository. The first (latest) version of the chart is deployed by default unless a particular version is specified using --version=<*chart-version*>

Quick Deployment Summary

For a basic deployment with default settings, follow these essential steps:

1. **Deploy Infra Chart:**
   ```
   helm install "infra-${RELEASENAME}" "${HELM_REPO}/ssp-infra" -n "${NAMESPACE}" --set
    sspReleaseName="${RELEASENAME}"
   ```
2. **Wait for Database Creation:**
   ```
   kubectl wait jobs.batch --namespace "${NAMESPACE}" --selector "app.kubernetes.io/name=ssp-infra-create-db-
   job" --for 'condition=complete' --timeout '5m'
   ```
3. **Deploy SSP Chart:**
   ```
   helm install "${RELEASENAME}" -n "${NAMESPACE}" "${HELM_REPO}/ssp" --set ssp.ingress.host="${SSP_FQDN}" --
   timeout=120m
   ```
4. **Verify Deployment:**
   ```
   kubectl get pods -n "${NAMESPACE}"
   ```

For advanced configurations, refer to the detailed configuration guides below.

Additional Configuration Topics

For advanced deployment scenarios and customizations, refer to these detailed configuration topics:

- **IDSP Infra Chart Configuration** - Detailed configuration options for the Infra chart including database, fluent-bit, and OpenShift settings
- **IDSP Main Chart Configuration** - Comprehensive configuration options for the main IDSP chart including database connectivity, ingress, and feature flags
- **IDSP OpenShift Configuration** - OpenShift-specific configurations and deployment examples
- **IDSP SSP-Data Chart Deployment** - Deploying the SSP-Data chart to load network risk data

Verifying the deployment

To verify that your deployment has been successful, check that all the IDSP pods are in the running and ready state. This may take a few minutes.

- Use the following command to check the status of the pods:
  ```
  kubectl get pods -n "${NAMESPACE}"
  ```
- Use the following command to verify the status of the pods and services:
  ```
  kubectl get pods,svc,ing -n "${NAMESPACE}" -o wide
  ```
- Use the following command to view events of the deployment:
  ```
  kubectl get events -n "${NAMESPACE}"
  ```

NOTE

- The output shows a column named READY with values such as 0/1, 1/1, 1/2, 2/2. The number on the right shows the total number of containers in the pod, while the number on the left shows the number of containers in the pod that are considered ready. The pod is considered ready when the number of ready containers equals to the total number of containers in the pod.
- Unlike the pods, it is expected that upon completion, the *jobs* have their status changed to Completed and not show as ready.

The following image shows how a successful deployment would look when all the pods are *Ready* (1/1) and in *Running* state:

```
NAME                                                              READY   STATUS    RESTARTS   AGE
ssp-hazelcast-0                                                   1/1     Running   0          39h
ssp428-infra-fluent-bit-dcmdj                                     1/1     Running   0          40h
ssp428-infra-fluent-bit-t74mw                                     1/1     Running   0          40h
ssp428-infra-fluent-bit-x58qw                                     1/1     Running   0          40h
ssp428-infra-fluent-bit-xvnck                                     1/1     Running   0          40h
ssp428-infra-ssp-infra-percona-xtradb-cluster-operator-5f7tjsxk   1/1     Running   0          40h
ssp428-ssp-admin-f9bb6bc9d-mldkv                                  1/1     Running   0          40h
ssp428-ssp-adminconsole-66477f6cfd-qbh2s                          1/1     Running   0          40h
ssp428-ssp-auth-mgr-586d76c9b4-99vws                              1/1     Running   0          40h
ssp428-ssp-azserver-84bc44589-z6qvn                               1/1     Running   0          40h
ssp428-ssp-db-metrics-exporter-5d6d85b466-hvm74                   1/1     Running   0          40h
ssp428-ssp-db-pxc-0                                               1/1     Running   0          39h
ssp428-ssp-factor-79ccb56865-s67kp                                1/1     Running   0          40h
ssp428-ssp-iarisk-6bd69c47c8-q7j4t                                2/2     Running   0          40h
ssp428-ssp-opa-6c4f6d8869-lg867                                   1/1     Running   0          40h
ssp428-ssp-signin-79598f4ddf-blq4q                                1/1     Running   0          40h
ssp428-ssp-swagger-69b97f978c-cr5tt                               1/1     Running   0          40h
```

To validate the deployment, run the following command which invokes the openId discovery API call and in addition requests IDSP to return the IP Address as identified by it:

```
curl --insecure https://"${SSP_FQDN}"/default/.well-known/openid-configuration?sspinfo=true
```

Ensure that the IDSP detects the real client IP. If the client is accessing IDSP through the Internet, this IP would be the IP from which the client accesses the Internet.
In the following example, the "xxx.xxx.xxx.xxx" represents the IP Address of the client as identified by IDSP.

```
curl -k https://${SSP_FQDN}/default/.well-known/openid-configuration?sspinfo=true
```

```
{
.
.
.
"clientIp" : "xxx.xxx.xxx.xxx",
"userIp" : "xxx.xxx.xxx.xxx"
}
```

Compare the IP address listed as clientIP in the output above with the IP address of the machine from which you ran the curl command. Notice that when connecting to IDSP through the Internet, the IP address should be the Public IP address that your machine is identified with while connecting to the Internet.

To find out what this IP is, use the following command:

```
$ curl https://ifconfig.me
```

NOTE
If the validation fails, make sure to restart the ingress pods. Due to specific SSL configurations in the Ingress objects, sometimes these settings may require a restart of the ingress pods.

```
kubectl rollout restart deployment ingress-nginx-controller -n ingress
```

NOTE

- Now you can deploy and integrate the Demo Application. Click here to navigate to the demo section.
- To know more about logs generated by the solution and enclave services, click here.

Retrieving Initial Security Credentials from Kubernetes secrets

The solution by default provisions a few system and tenant-level clients and an admin user to help you to get started with the solution's capabilities. To know more about these seeded default credentials, see Administrative Model.

Next Step

See Configuring Administrative Access

IDSP Infra Helm Chart Configuration

Configuring IDSP Infra helm chart parameters and options.

This topic provides detailed information about all configuration options available for the IDSP Infra helm chart deployment.

Overview

The IDSP Infra chart serves to deploy a demo database and deploy fluent-bit for streaming IDSP logs and audit messages. If your preference is to use an external database and forgo the deployment of fluent-bit for log streaming, you have the option to skip the infra chart deployment and proceed directly to the IDSP deployment.

Demo Database Configuration

The demo database is deployed by default. If you do not intend to use the demo database, use the following command:

```
--set db.enabled=false
```

Fluent-bit Configuration

The fluent-bit is deployed by default. If you do not intend to use the fluent-bit to stream IDSP logs into the log storage, use the following parameter:

```
--set fluent-bit.enabled=false
```

You must provide a new customConfig parameters section under the fluent-bit section in the ***IDSP infra chart values yaml*** file to configure where to stream the IDSP log and audit messages. To know more about it, see Logs and Security Audit Data section.

Using Private Image Registry

To deploy IDSP infra chart using images stored in your private image registry:

```
--set fluent-bit.image.repository=<PRIVATE_IMAGE_REGISTRY>/fluent/fluent-bit
```

For more information, see the Using Private Image Registry section.

If you are using the internal demo database with images stored in your private image registry, add these values to your values.yaml:

```
db:
  image:
    pxc:
      repository: <PRIVATE_IMAGE_REGISTRY>percona/percona-xtradb-cluster
      tag: 8.0.29-21.1
    operator:
      repository: <PRIVATE_IMAGE_REGISTRY>percona/percona-xtradb-cluster-operator
      tag: 1.12.0
    haproxy:
      repository: <PRIVATE_IMAGE_REGISTRY>percona/percona-xtradb-cluster-operator
      tag: 1.12.0-haproxy
    proxysql:
      repository: <PRIVATE_IMAGE_REGISTRY>percona/percona-xtradb-cluster-operator
      tag: 1.12.0-proxysql
    pmm:
      repository: <PRIVATE_IMAGE_REGISTRY>percona/pmm-client
      tag: 2.32.0
    backup:
      repository: <PRIVATE_IMAGE_REGISTRY>percona/percona-xtradb-cluster-operator
      tag: 1.12.0-pxc8.0-backup
    mysqld_exporter:
      repository: <PRIVATE_IMAGE_REGISTRY>prom/mysqld-exporter
      tag: v0.12.1
```

Using External Database

If you are using a customer database (external db mode), pass this parameter in the IDSP infra helm install command:

```
--set db.enabled=false
```

Container Runtime Configuration

If the container runtime is "containerd" or "cri-o", add this parameter in the IDSP infra helm install command. If the parser parameter is not specified, it defaults to "cri".

Possible values include "cri" or "docker".

To find out which container runtime is used in your cluster nodes, run:

```
kubectl get nodes -o wide
```

Then add the appropriate parameter:

```
--set sspfbService.parser=cri
```

Docker Registry Authentication

By default, IDSP Infra helm chart pulls images from the Docker Hub registry which requires authentication. Create a docker-registry secret to store the registry credentials:

```
kubectl create secret docker-registry ssp-infra-registrypullsecret -n ${NAMESPACE} \
  --docker-server="https://index.docker.io/v2/" \
  --docker-username="<dockerhub-username>" \
  --docker-password="<dockerhub-password>" \
  --docker-email="<your-email-address>"
```

Once the secret is created, specify the secret by adding these parameters to the IDSP infra helm chart deployment command:

```
--set global.registry.existingSecrets[0].name='ssp-infra-registrypullsecret' \
--set fluent-bit.imagePullSecrets[0].name="ssp-infra-registrypullsecret"
```

> **NOTE**
> If you decide to use your own private registry (recommended) and your registry requires authentication as well, you can use a similar command using your own docker server URL and credentials.

Ingress Nginx Logs Streaming

To stream the Ingress Nginx logs in third-party dashboards, such as Kibana dashboard, pass the following parameter:

```
--set sspfbService.ingressLogsIncluded=true
```

For more information, see the Observing Nginx Ingress Traffic section.

Custom Log Configuration

To stream the IDSP log and audit messages (ssp_log, ssp_audit, ssp_tp_log) to log storage systems available in your environment, pass this parameter:

```
-f customConfig_sspinfra.yaml
```

For more information, see the Logs and Security Audit Data section.

Infra Helm Chart Deployment Command

Deploy IDSP Infra helm chart using the following command, adding to it any additional parameters that you may need as indicated in the previous sections:

```
helm install "infra-${RELEASENAME}" "${HELM_REPO}/ssp-infra" -n "${NAMESPACE}" --set
 sspReleaseName="${RELEASENAME}"
```

> **NOTE**
> When you are using the out-of-the-box demo database, ensure that the database is completely created by **waiting** until the database creation job shows "**completed**" status before proceeding with the IDSP deployment. You can check the status of the database creation job by running this command:
>
> ```
> kubectl wait jobs.batch \
> --namespace "${NAMESPACE}" \
> --selector "app.kubernetes.io/name=ssp-infra-create-db-job" \
> --for 'condition=complete' \
> ```

```
--timeout '5m'
```

IDSP OpenShift Configuration

Learn how to deploying IDSP on OpenShift with specific configurations and requirements.

This topic provides detailed information about OpenShift-specific configurations required for IDSP deployment.

Overview

Deploying IDSP on OpenShift requires specific security context configurations and ingress settings. This topic covers all OpenShift-specific parameters and configurations.

OpenShift Environment Setup

Before deploying on OpenShift, set the following environment variables:

```
read -d '/' -r SSP_UID < <(kubectl get ns ${NAMESPACE} -o jsonpath='{.metadata.annotations.openshift\.io/sa
\.scc\.uid-range}')
read -d '/' -r SSP_GID < <(kubectl get ns ${NAMESPACE} -o jsonpath='{.metadata.annotations.openshift\.io/sa
\.scc\.supplemental-groups}')
export SSP_FSGROUP=${SSP_GID}
```

Infra Chart OpenShift Configuration

Security Context for Infra Chart:

Pass the following additional parameters in the IDSP infra helm install command:

```
--set global.securityContext.runAsUser=${SSP_UID} \
--set global.securityContext.runAsGroup=${SSP_GID} \
--set global.securityContext.fsGroup=${SSP_FSGROUP} \
--set fluent-bit.serviceAccount.create=true \
--set fluent-bit.openShift.enabled=true \
--set fluent-bit.openShift.securityContextConstraints.create=true \
#Note that --set fluent-bit.securityContext.privileged=true is required if using tail log db
```

See Additional Parameters to consider in OpenShift.

Prometheus Configuration:

Enable Prometheus to Scrape IDSP Service's Information. For more information, see Enabling Promethues in OpenShift.

```
kubectl label ns ${NAMESPACE} openshift.io/cluster-monitoring="true"

cat <<EOF | kubectl apply -n ${NAMESPACE} -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: prometheus-k8s
rules:
- apiGroups:
  - ""
  resources:
  - services
  - endpoints
  - pods
```

```
  verbs:
  - get
  - list
  - watch

---

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: prometheus-k8s
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: prometheus-k8s
subjects:
- kind: ServiceAccount
  name: prometheus-k8s
  namespace: openshift-monitoring
EOF
```

IDSP Chart OpenShift Configuration

Security Context for SSP Chart:

To deploy in OpenShift, pass the additional parameters in the helm chart install command:

```
--set ssp.global.securityContext.fsGroup=${SSP_FSGROUP} \
--set ssp.global.securityContext.runAsGroup=${SSP_GID} \
--set ssp.global.securityContext.runAsUser=${SSP_UID} \
--set hazelcast-enterprise.securityContext.fsGroup=${SSP_FSGROUP} \
--set hazelcast-enterprise.securityContext.runAsGroup=${SSP_GID} \
--set hazelcast-enterprise.securityContext.runAsUser=${SSP_UID}
```

See Ingress in OpenShift for more information.

Ingress Type Configuration

OpenShift Routes:

If you intend to use OpenShift routes when deploying IDSP, pass these additional parameters in the IDSP helm deployment command. Note that OpenShift Route requires OpenShift version 4.11 and above.

```
--set ssp.ingress.type="openshiftroute" \
--set ssp.ingress.ingressClassName="openshift-default"
```

Default Nginx Ingress:

If you intend to use Nginx Ingress which is the default ingress used when deploying IDSP, pass these additional parameters in the IDSP helm deployment command:

```
--set ssp.ingress.type="nginx" \
--set ssp.ingress.ingressClassName="nginx"
```

See Ingress in OpenShift for more information.

SSP Data Chart OpenShift Configuration

For OpenShift deployment, the following additional parameters must be passed in the "ssp-data" helm install command:

```
--set ssp.global.securityContext.runAsUser=${SSP_UID} \
--set ssp.global.securityContext.runAsGroup=${SSP_GID} \
--set ssp.global.securityContext.fsGroup=${SSP_FSGROUP}
```

For information on deploying IDSP Data chart, see Deploying IDSP Data Chart .

Complete OpenShift Deployment Example

1. Set Environment Variables:

```
export PREFIX="myssp"
export DOMAIN="myiam.example.com"
export SSP_FQDN="${PREFIX}.${DOMAIN}"
export SSP_MGMT_FQDN="system.${SSP_FQDN}"
export RELEASENAME="ssp-prod"
export NAMESPACE="ssp-prod"
export HELM_REPO='ssp_helm_charts'
export HELM_REPO_URL=https://ssp_helm_charts.storage.googleapis.com

read -d '/' -r SSP_UID < <(kubectl get ns ${NAMESPACE} -o jsonpath='{.metadata.annotations.openshift\.io/sa
\.scc\.uid-range}')
read -d '/' -r SSP_GID < <(kubectl get ns ${NAMESPACE} -o jsonpath='{.metadata.annotations.openshift\.io/sa
\.scc\.supplemental-groups}')
export SSP_FSGROUP=${SSP_GID}
```

2. Deploy Infra Chart:

```
helm install "infra-${RELEASENAME}" "${HELM_REPO}/ssp-infra" -n "${NAMESPACE}" \
  --set sspReleaseName="${RELEASENAME}" \
  --set global.securityContext.runAsUser=${SSP_UID} \
  --set global.securityContext.runAsGroup=${SSP_GID} \
  --set global.securityContext.fsGroup=${SSP_FSGROUP} \
  --set fluent-bit.serviceAccount.create=true \
  --set fluent-bit.openShift.enabled=true \
  --set fluent-bit.openShift.securityContextConstraints.create=true
```

3. Deploy SSP Chart:

```
helm install "${RELEASENAME}" -n "${NAMESPACE}" "${HELM_REPO}/ssp" \
  --set ssp.ingress.host="${SSP_FQDN}" \
  --set ssp.ingress.type="openshiftroute" \
  --set ssp.ingress.ingressClassName="openshift-default" \
  --set ssp.global.securityContext.fsGroup=${SSP_FSGROUP} \
  --set ssp.global.securityContext.runAsGroup=${SSP_GID} \
  --set ssp.global.securityContext.runAsUser=${SSP_UID} \
  --set hazelcast-enterprise.securityContext.fsGroup=${SSP_FSGROUP} \
  --set hazelcast-enterprise.securityContext.runAsGroup=${SSP_GID} \
  --set hazelcast-enterprise.securityContext.runAsUser=${SSP_UID} \
  --timeout=120m
```

Deploying IDSP Data Chart

Learn to deploy the IDSP-Data chart to load network watchlist and fraud IP data into the IDSPdatabase.

Overview

NOTE

- You can install the "ssp-data" chart, only after installing the IDSP.
- The "ssp-data" chart is updated once a week, therefore to get the most recent data, upgrade to the latest helm chart. The ssp-data charts recent version is 1.0.2023.<week_number>.
- The Geo Location service is used to managing Network Location data which is deployed by default. Search for Geolocation Service in this link.

The solution provides a separate helm chart called, "ssp-data" to load the network watchlist, and fraud IP data into the IDSP database. The solution uses this data to access risk. When this helm chart is deployed, it loads approximately 250K records of data.

Data Sources

The data that is loaded by the "ssp-data" chart comes from the following sources:

Data Type	Source
Unallocated Network	Bogons
Anonymizer/Fraud IP	Neustar

Database Requirement

The database should have 100 MB (megabytes) of space to hold the network data information.

Deploying the latest "ssp-data" Chart

To deploy the latest "ssp-data" chart into the solution's database, use the following set of commands:

1. Get the latest Helm chart updates:
   ```
   helm repo add ssp_helm_charts https://ssp_helm_charts.storage.googleapis.com
   helm repo update
   ```
2. Perform the following steps to deploy the ssp-data chart:
   a. The "ssp-data" helm chart uses SSL mode set to "REQUIRED" as a default.
   When using Customer Provided database, add the attribute, **--set ssp.db.sslMode=<ssl mode>**, and change <ssl mode> to one of the supported values: DISABLED, PREFFERED, REQUIRED, VERIFY_CA, VERIFY_IDENTITY. When using VERIFY_CA and VERIFY_IDENTIFY, you are required to specify a CA certificate(s) that will be used to verify the DB server certificate by passing the SSL certificate(s) secret name, **"--set ssp.db.existingSslSecret=<secret name>".** See DB in SSL mode for more information.
   b. If you want IDSP to use the ***customer provided database***, pass the following database parameters to the "ssp-data" helm chart:
   ```
   --set sspReleaseName=${RELEASENAME} \
   --set ssp.db.serviceHost=<db host name, e.g. sspdb.mysql.database.azure.com> \
   --set ssp.db.servicePort=3306 \
   --set ssp.db.user="<db user name>" \
   --set ssp.db.existingSecret=<secret name>"
   ```
   The existing secret should point to the existing database secret used by IDSP Helm chart.
   c. Following is the helm install command to deploy the "ssp-data" helm chart:

```
helm install "${RELEASENAME}-data" -n "${NAMESPACE}" "${HELM_REPO}/ssp-data" \
--set sspReleaseName="${RELEASENAME}" \
--set ssp.global.ssp.registry.existingSecrets[0].name=ssp-registrypullsecret
```

NOTE
For OpenShift deployment, the following additional parameters must be passed in the "ssp-data" helm install command.

```
--set ssp.global.securityContext.runAsUser=${SSP_UID} \
--set ssp.global.securityContext.runAsGroup=${SSP_GID} \
--set ssp.global.securityContext.fsGroup=${SSP_FSGROUP}
```

This downloads the latest ssp-data helm chart. Once downloaded, the "ssp-data-networkdataload-job" will start automatically and upload the data into the solution's database.

Verifying SSP Data Chart Deployment

To verify that the SSP-Data chart has been deployed successfully:

1. Check the status of the data loading job:
   ```
   kubectl get jobs -n "${NAMESPACE}" | grep ssp-data-networkdataload
   ```
2. Monitor the job logs:
   ```
   kubectl logs -n "${NAMESPACE}" -l job-name=ssp-data-networkdataload-job
   ```
3. Verify the job completion:
   ```
   kubectl wait jobs.batch \
   --namespace "${NAMESPACE}" \
   --selector "app.kubernetes.io/name=ssp-data-networkdataload-job" \
   --for 'condition=complete' \
   --timeout '10m'
   ```

Once the job shows "Completed" status, the network data has been successfully loaded into your IDSP database.

Reference Link:

Additional parameters to pass in OpenShift environment, see here.

Post-Installation Checklist

Consider the tasks in the checklist after deploying the Symantec Identity Security Platform (SSP) helm chart.

After you have installed IDSP, review the following information and perform the post-installation tasks.

#	Post-installations Tasks
1.	Verify your IDSP installation, check that all the Kubernetes Pods are up and running, and ensure that your network components are See Verifying the deployment section.
2.	The solution by default provisions a few system and tenant-level clients and an admin user to help you to get started with the solutio credentials, see Administrative Model section.
3.	Take a backup of all the user-specified values that were used in the deployment of IDSP charts for recovery installation purposes. S
4.	Use a curl command to access the open id discovery url to obtain information needed to interact with the OAuth Authorization Serve section.
5.	Take a backup of the Master Encryption Key for future retrievals as this is the only key that can be used to decrypt the solution's enc there is no other way to recover the encrypted data. See Master Encryption Key.
6.	Retrieve the IDSP client secrets to administer the solution. To know more about the seeded default credentials, see Configuring Adm

7.	If you have deployed the enclave services, (such as Elasticsearch, Kibana, Prometheus, Grafana) into IDSP to monitor logs, configu Configuring Enclave Services section.
8.	You can optionally deploy the Sample Web Application provided by IDSP to test multiple scenarios and use cases in the solution. Se
9.	Broadcom provides a postman collection that can be downloaded and configured to test the demo environment. If you are using pos IDSP solution: 1. Set the environment variables. 2. Acquire the access token. 3. Update the CORS setting to update your "allowedOrigins". 4. Configure your ID Store. 5. Update the Tenant Admin AuthZ policy to add an admin user and a group. 6. Generate Tenant Admin access token / User access token to get the appropriate privileges to access the IDSP APIs. • IDSP **v4.0** Postman Collection is available for download from the Support Portal > My Downloads. • For the basic steps to get started with the IDSP Postman collection, see Using IDSP postman collection. • The Custom Claims SPI yaml can also be downloaded from the Support Portal > My Downloads

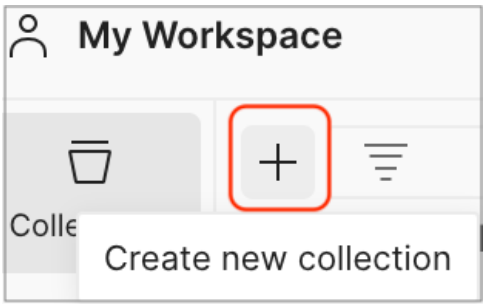
Using the Symantec Identity Security Platform postman collection

When you are setting up the Symantec Identity Security Platform Postman collection to manage your environments, it is a good practice to follow these steps to streamline the process and make things easier for the future.

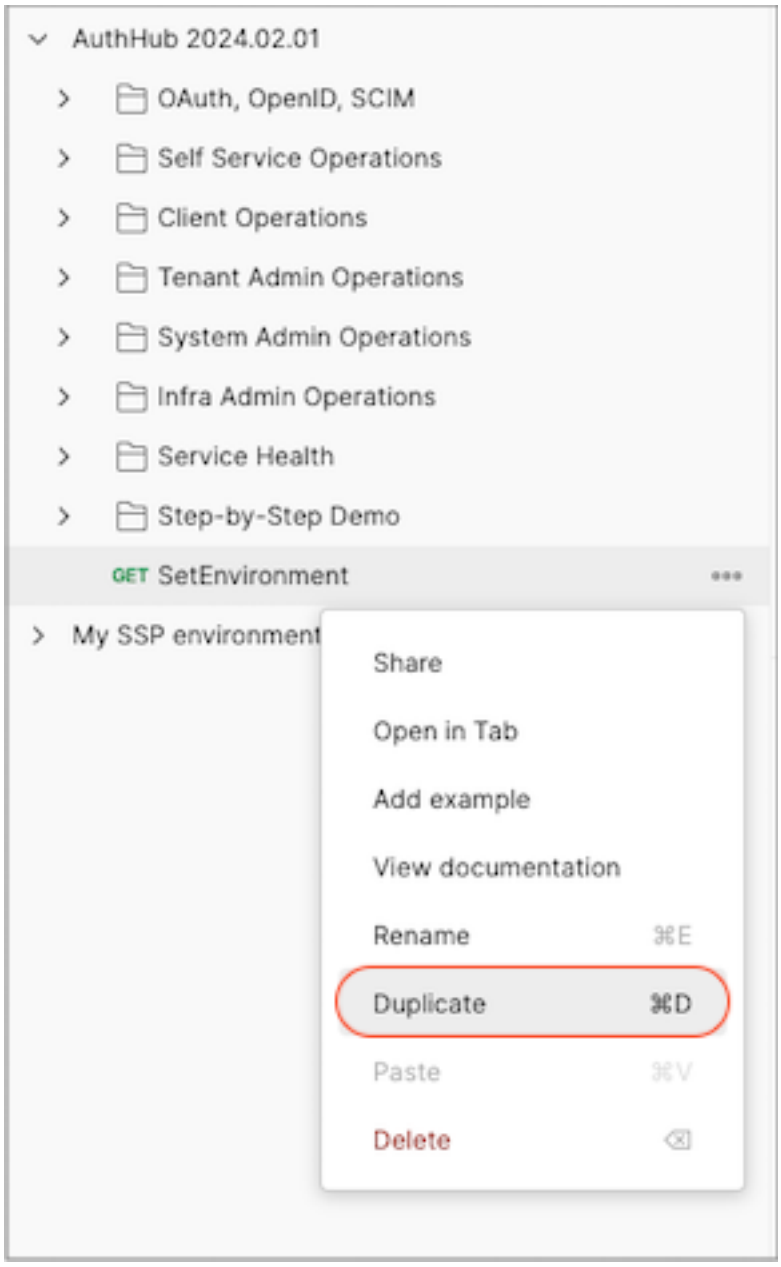
Follow these steps:

1. Import the IDSP API Collection into Postman.

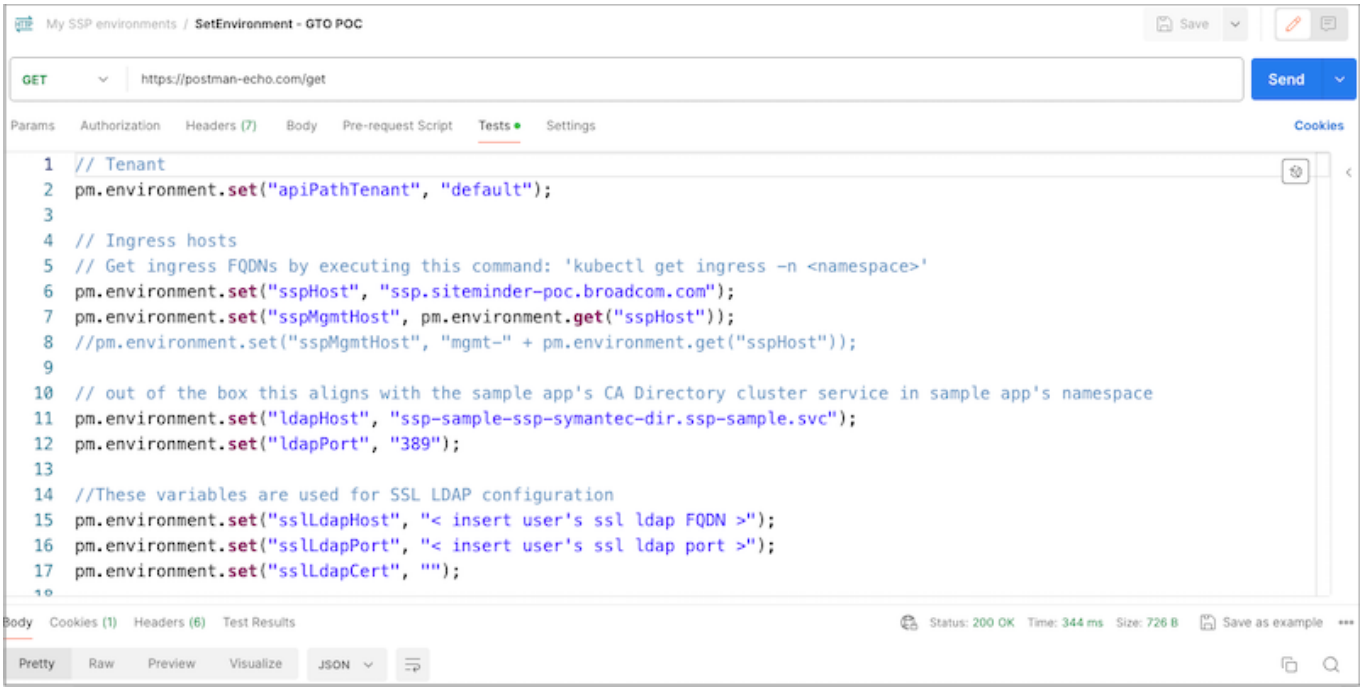
2. Create a new Blank collection by clicking the + sign.



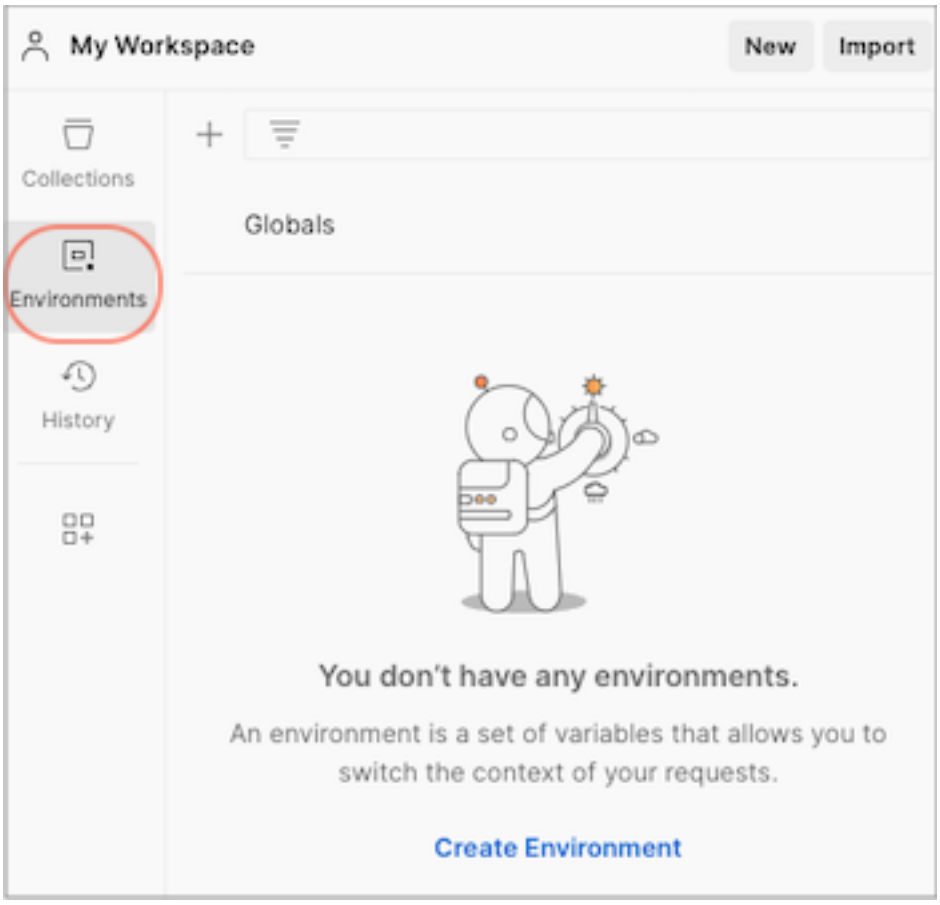
3. Rename the new collection. For instance, you might name the folder as "My SSP environments".
4. Duplicate the SetEnvironment API request (item) from the IDSP collection.



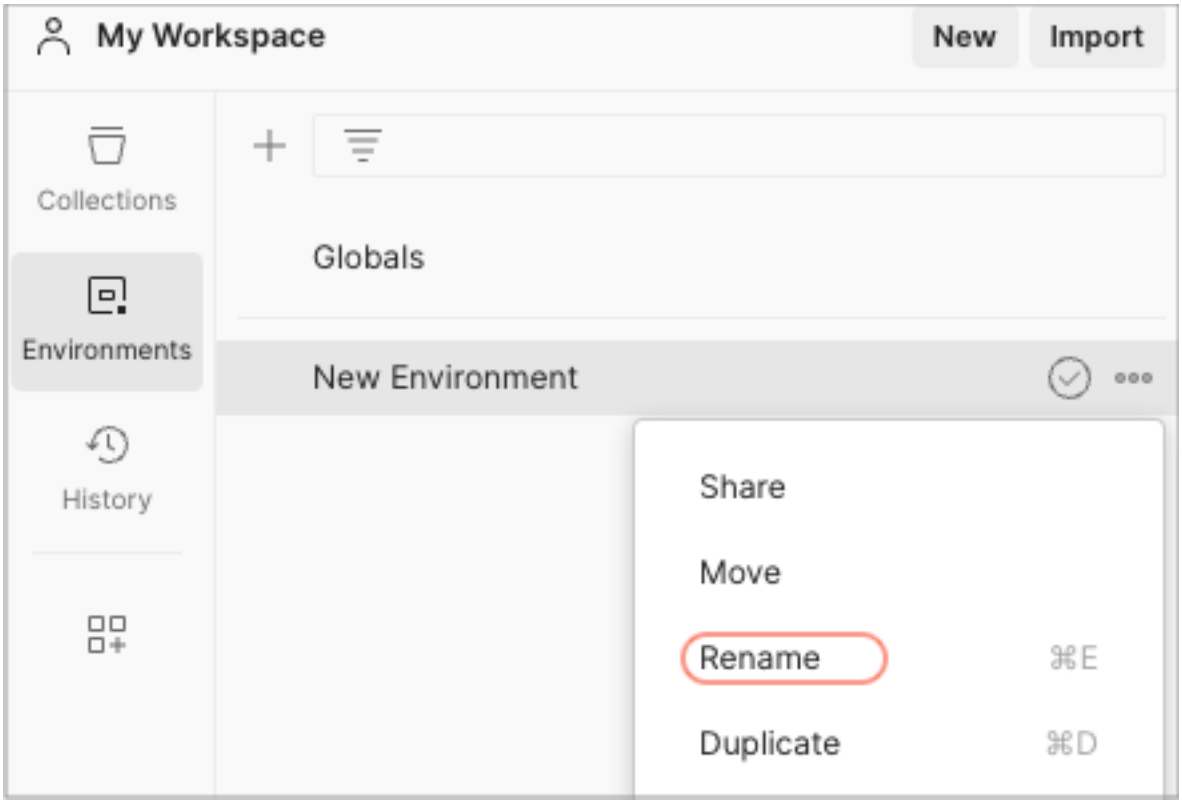
5. Drag the duplicate SetEnvironment item (SetEnvironment Copy) to the new collection you created ("My SSP environments").
6. Rename the "SetEnvironment Copy" to indicate the ssp environment name that you are planning to manage to use the API. For instance, "SetEnvironment - ah-admin-example.com"
7. Edit the Tests/Scripts section of the duplicated "SetEnvironment" item to update the environment information to include the following details:
   a. The environment's sspHost (For instance, ah-admin-example.com).
   b. Your admin client defaultTenantClientID and defaultTenantClientSecret.



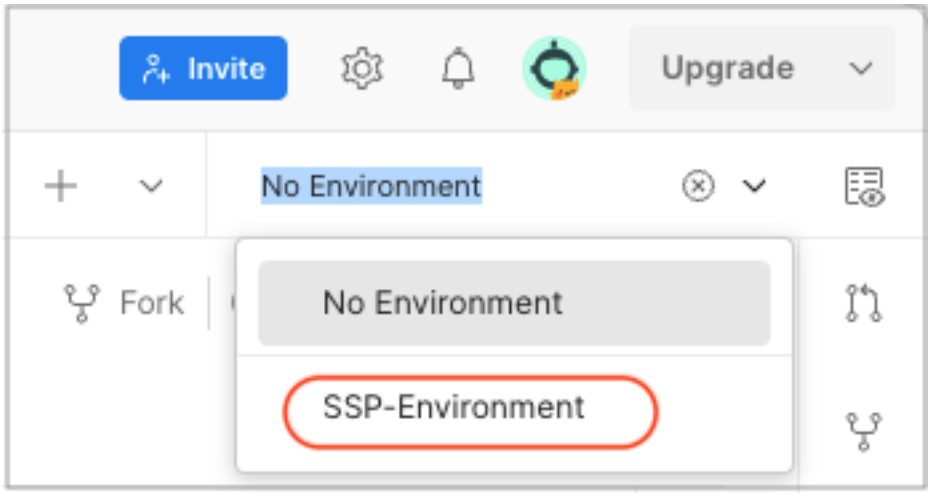
8. If you do not have an environment already created, follow these steps to create it:
   a. Go to the environment section (on the left navigation).



b. Create an environment by clicking the + sign.
c. Once created, click the ellipsis (...) and choose "Rename" to rename the environment (For instance, "SSP environment")



d. On the top left, select the environment that you created.



9. Navigate back to the ssp environment request you edited previously in step7 and invoke the request by clicking `Send`.
10. As you use the collection, you notice that most of the requests within the collection are set to carry context from one command to another by setting additional variables as needed.
For instance, acquiring an access token with one request, and using it with other requests of the same folder.

Configuring Enclave Services to Show Symantec Identity Security Platform Content

Learn how to configure Enclave Services to show Symantec Identity Security Platform content.

This page includes the following topics:

- Configuring Data Views in Kibana to Monitor Logs - 9.1.3
- Configuring Grafana to show the IDSP Dashboard

NOTE

- This configuration procedure and the reference screenshots are of Kibana 9.1.3. For information about the procedure on later versions of Kibana, see the Elasticsearch documentation.
- Ensure that the IDSP is installed before configuring the enclave services.

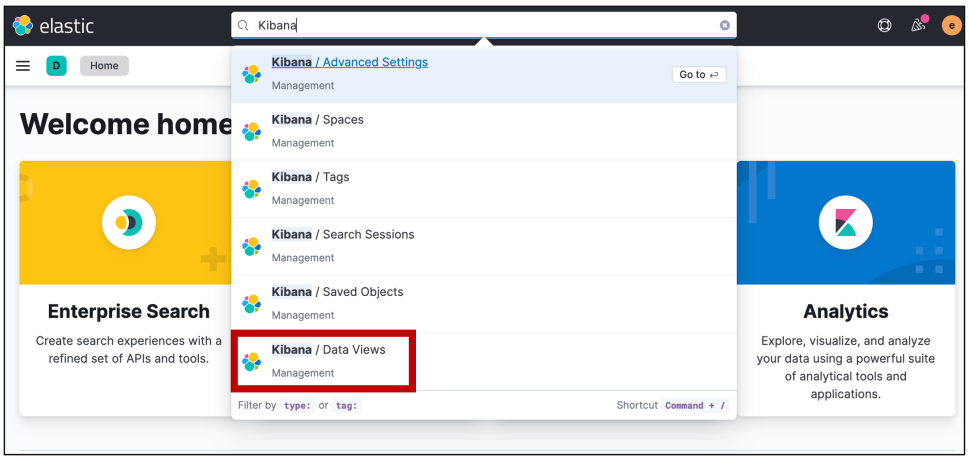
Configuring Data Views in Kibana to Monitor Logs - 9.1.3

To view IDSP logs, use the Kibana monitoring dashboard.

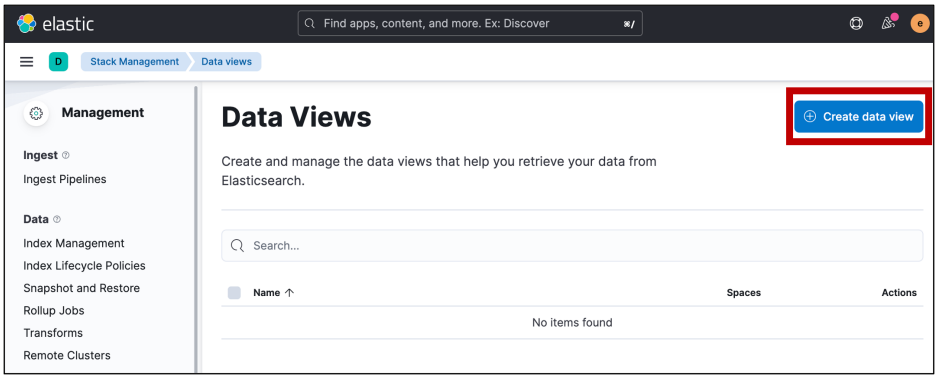
1. Access the Kibana service at the following location. Replace <DOMAIN> with the value that you have set in your environment during deployment, before running this command:

   ```
   http://kibana.<DOMAIN>
   ```

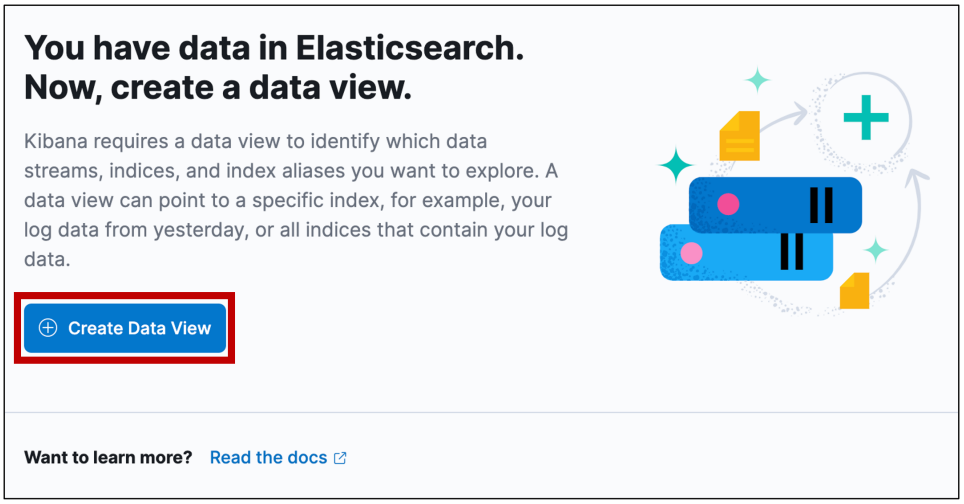
   This displays the Kibana home page.
2. In the home page, search for and click **Kibana / Data Views.**



3. In the **Data Views** page, click **Create data view.**



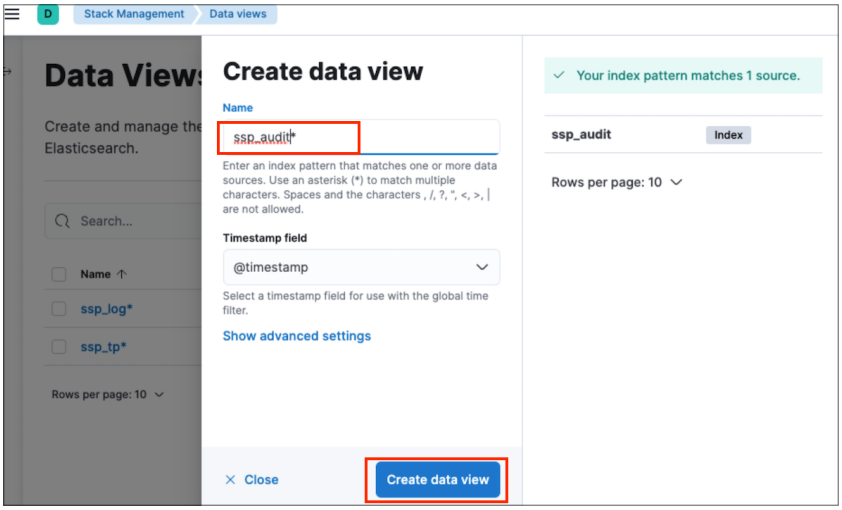
4. In the page that appears next, click **Create Data View.**



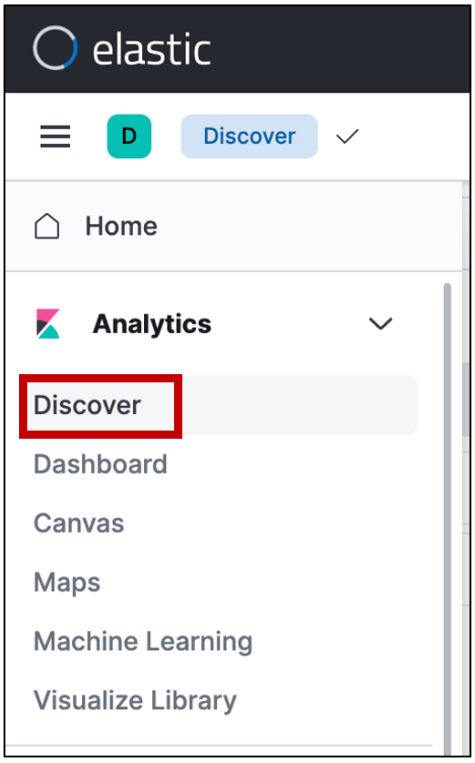
5. In the **Create data view** page, enter an Index pattern name (for example, ssp_log*, ssp_audit*, ssp_tp_log*) in the **Name** field, select the **Timestamp field** as **@timestamp**, and click **Create data view.**

   NOTE

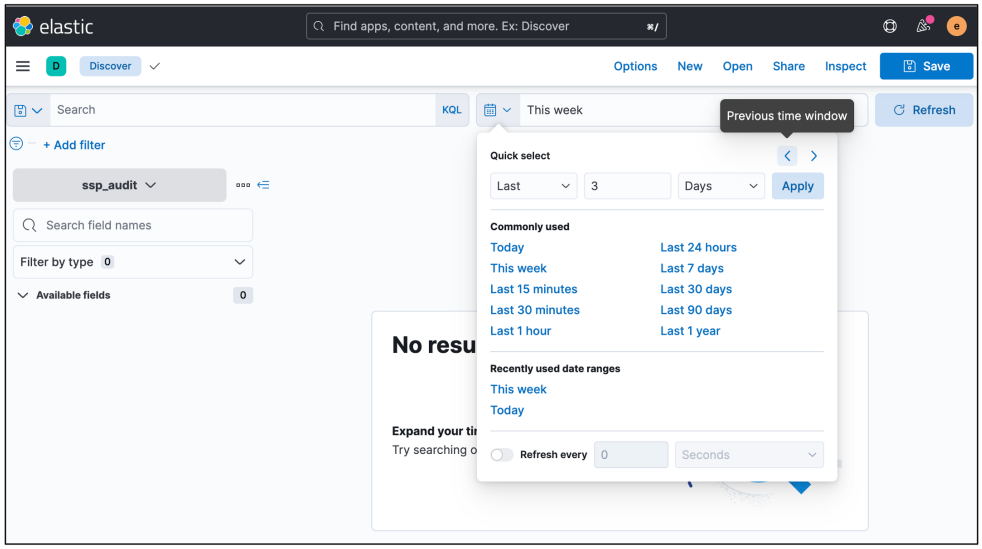
   Repeat step from 2 through 5 for each of the Index pattern names that you want to create.



6. In the Hamburger menu icon on the top-left of the page, click **Discover.**



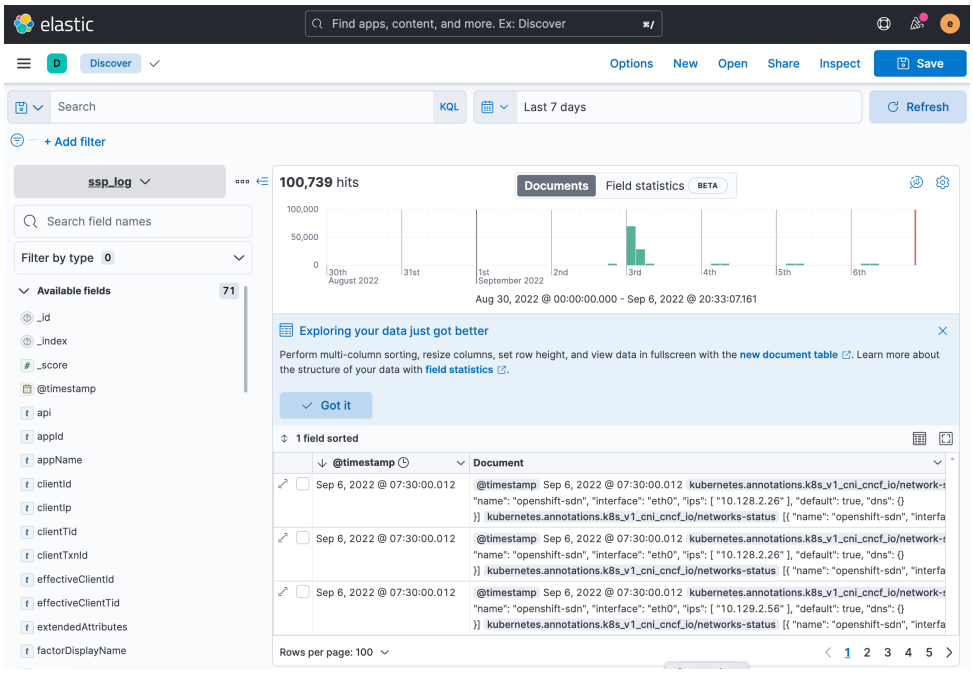
7. You can use the date filter on the top right to retrieve logs from a specific time.



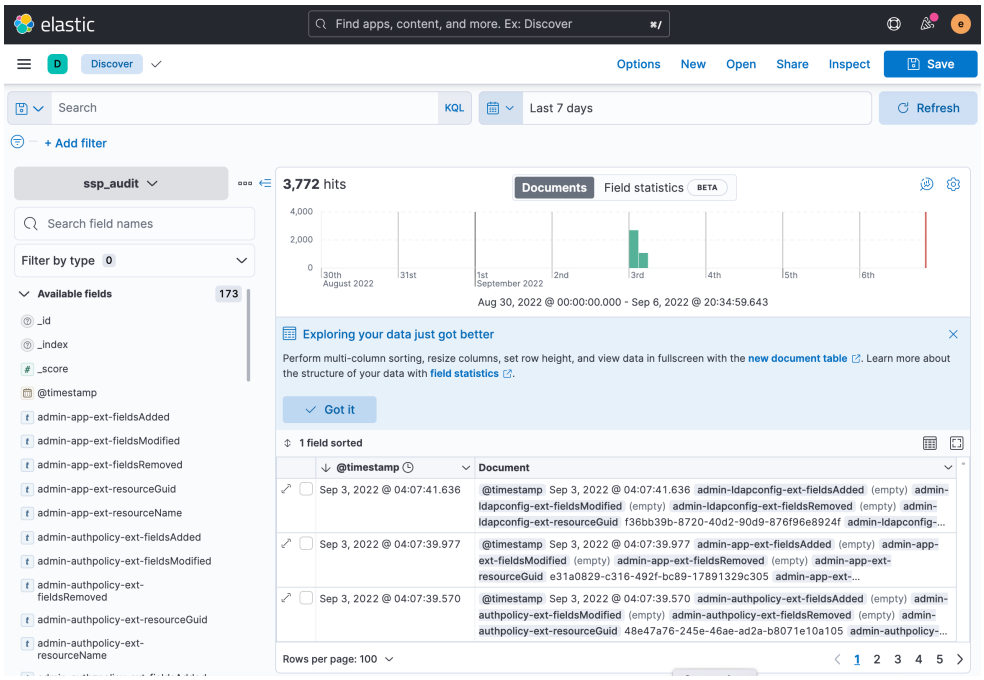
NOTE
Initially, the logs may display empty data. However, after you select the right filter option the data will be populated.

See sample images of the IDSP logs.

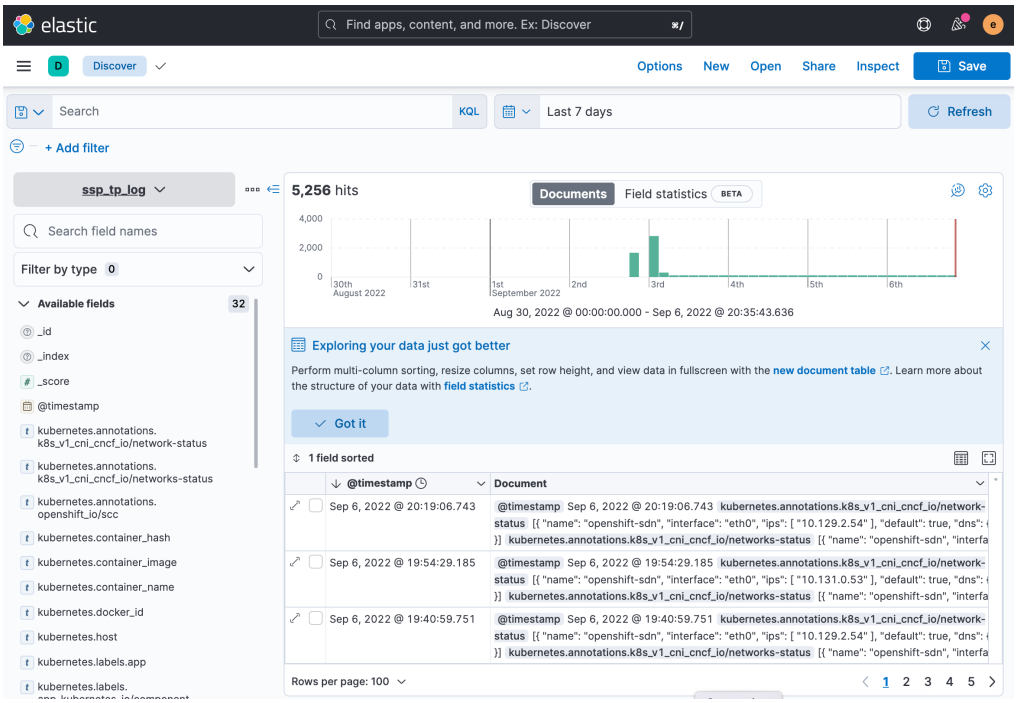
ssp_log



ssp_audit



ssp_tp_log



Configuring Grafana to show the IDSP Dashboard

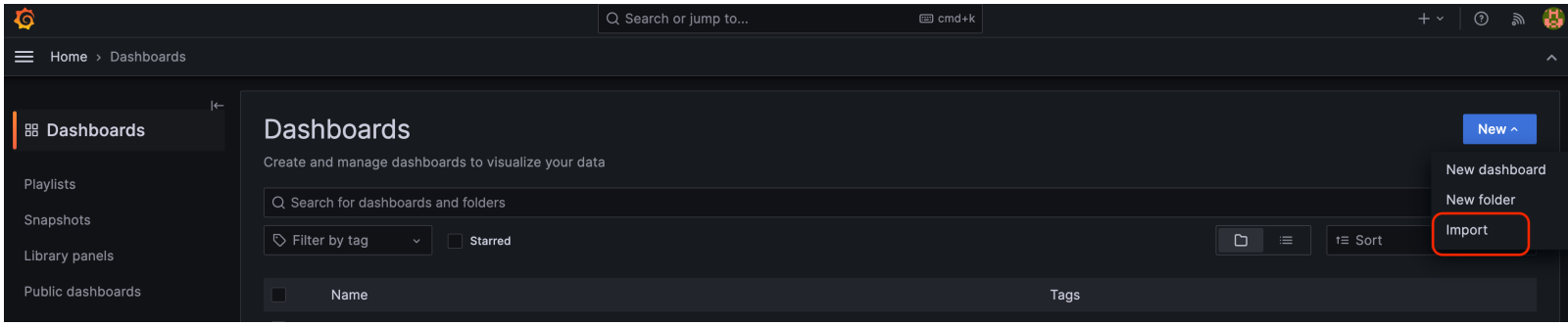
1. Access the Grafana UI at the following location.

   `http://grafana.<DOMAIN>`

   NOTE

   - In the command above, replace <DOMAIN> with the value that you have set in your environment during deployment, before running this command.
   - If you are unable to access Grafana, ensure that the Grafana FQDN is resolvable as described in the section, Publishing the Enclave Endpoints.
   - The default Username and Password to access Grafana are:
     Username: admin
     Password: prom-operator

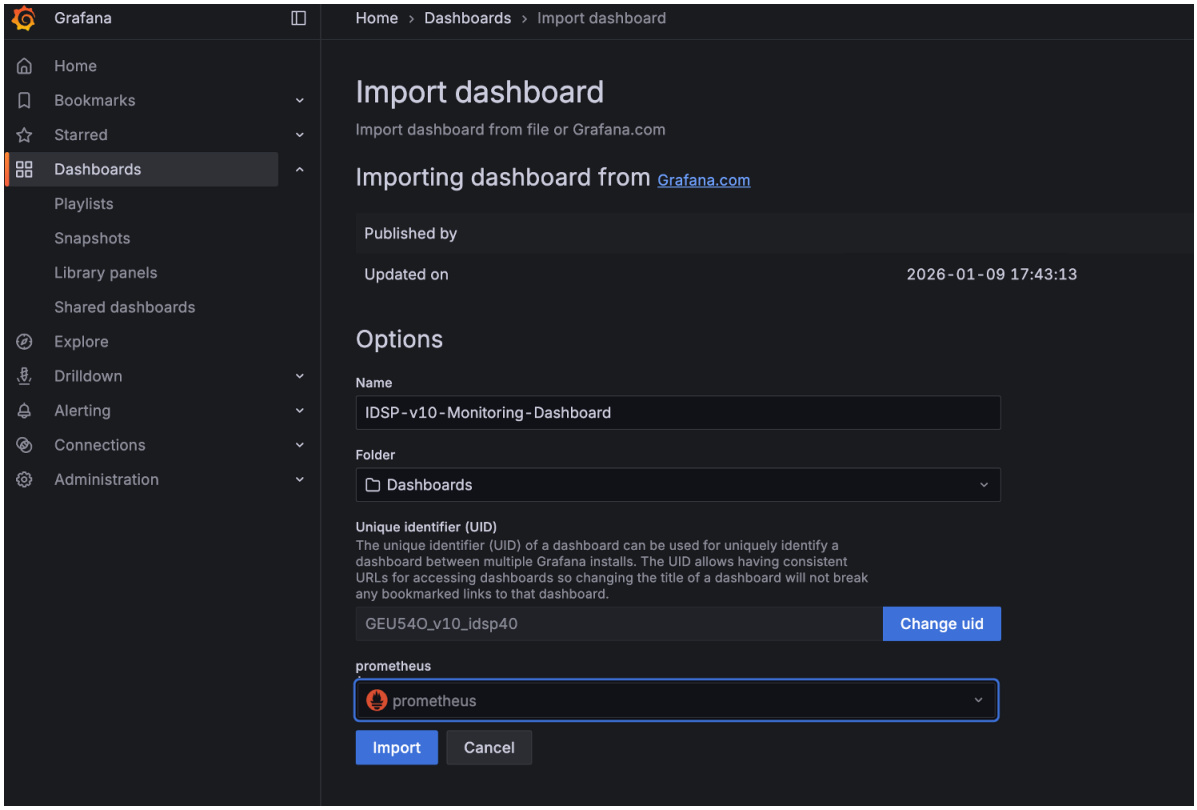
2. Navigate to Grafana dashboard > Import.



3. Under **Import via grafana.com** selector, enter the IDSP Monitoring Dashboard template Id-25306.
   Use the following Dashboard template Id to import the dashboard:
   – ***IDSP Monitoring Dashboard*** - 25306
4. Click **Load.**

The *SSP-v9-Monitoring-Dashboard* gets loaded. Similarly, other dashboards also get loaded if you import them in the same manner.

5. From the *Prometheus* drop-down, select **Prometheus** and click **Import.**

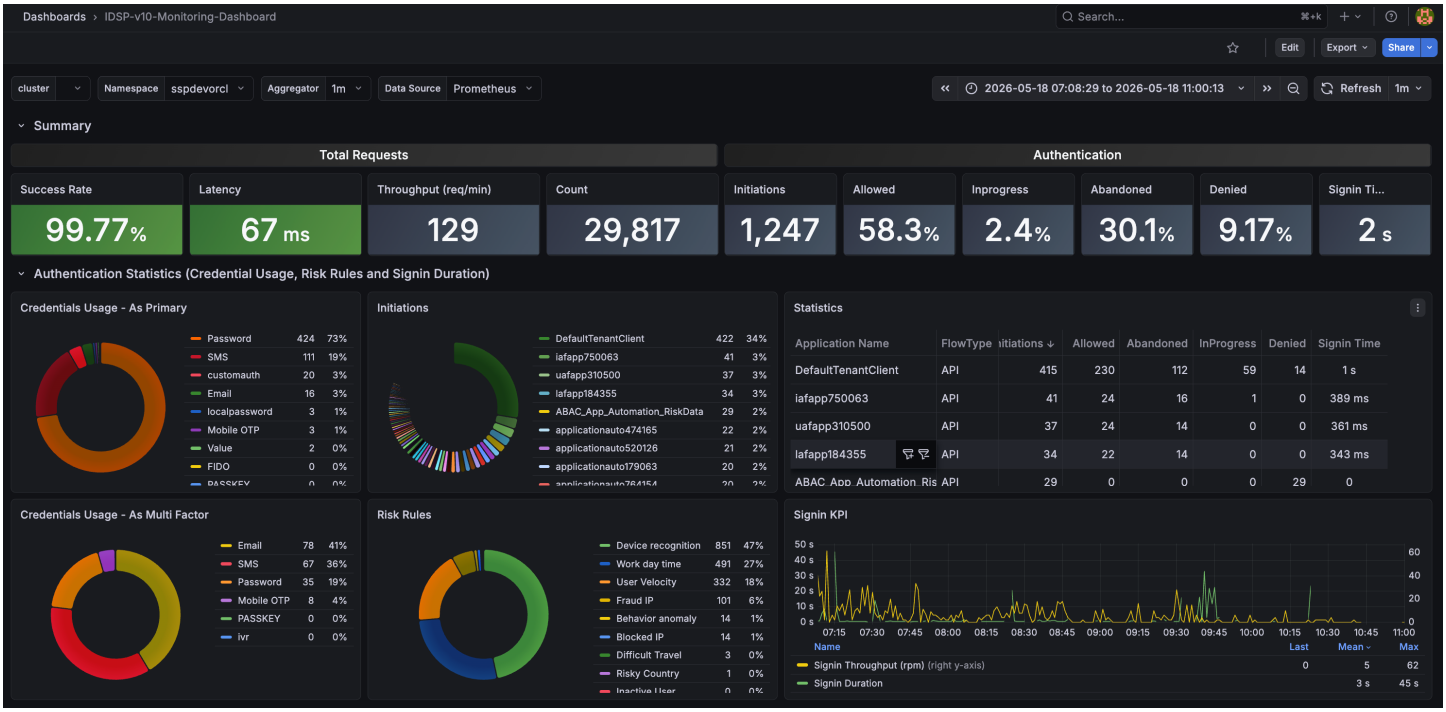


NOTE

Ensure that you verify the "Namespace" field at the top left of the dashboard page to have the name of the namespace where you have deployed the IDSP chart.

This is a sample image showing the IDSP Monitoring Dashboard:

Figure 7: Sample monitoring dashboard"

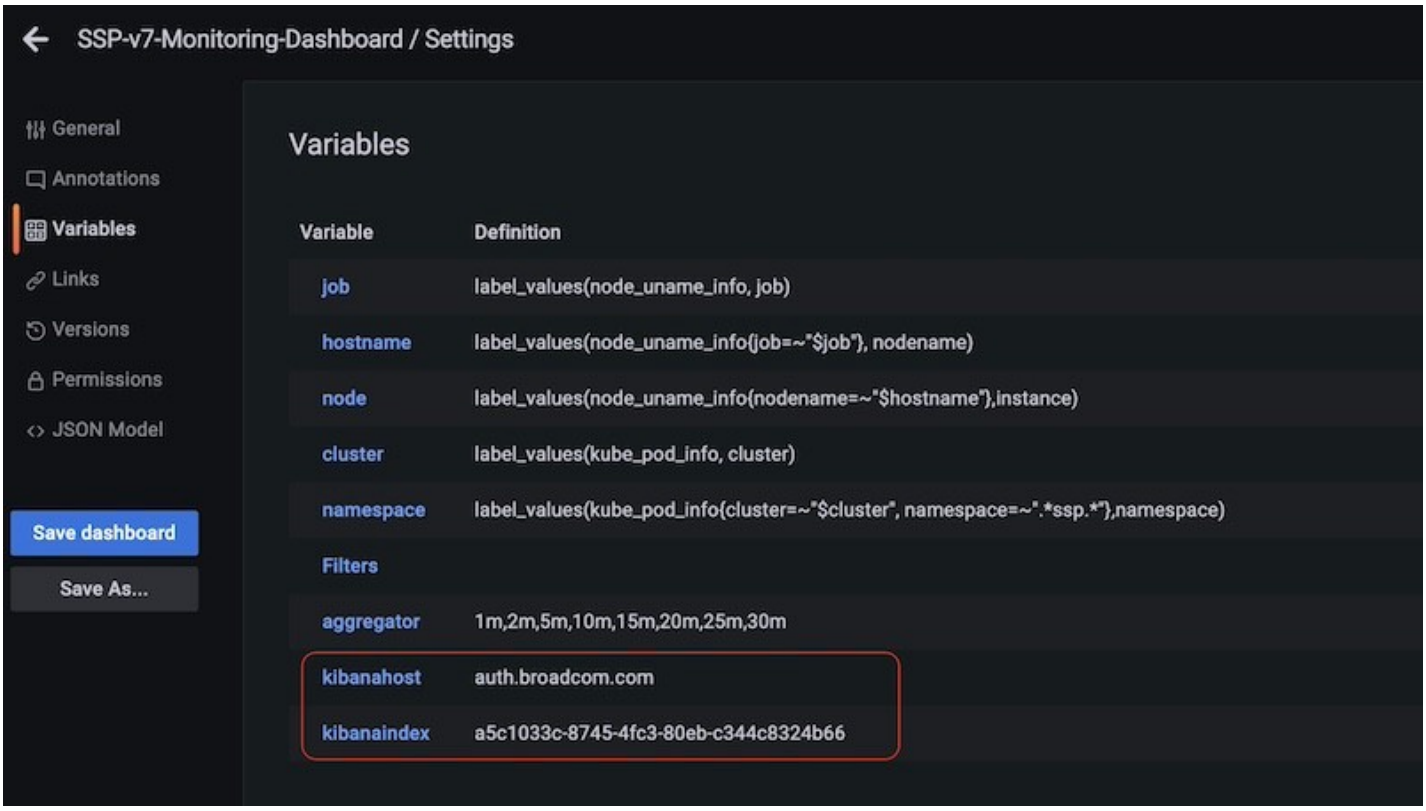


NOTE
Grafana could also be used to view generic cluster dashboards like Kubernetes Cluster Metrics (11074), JVM Metrics (9568), NGINX Ingress Controller Metrics (9614).

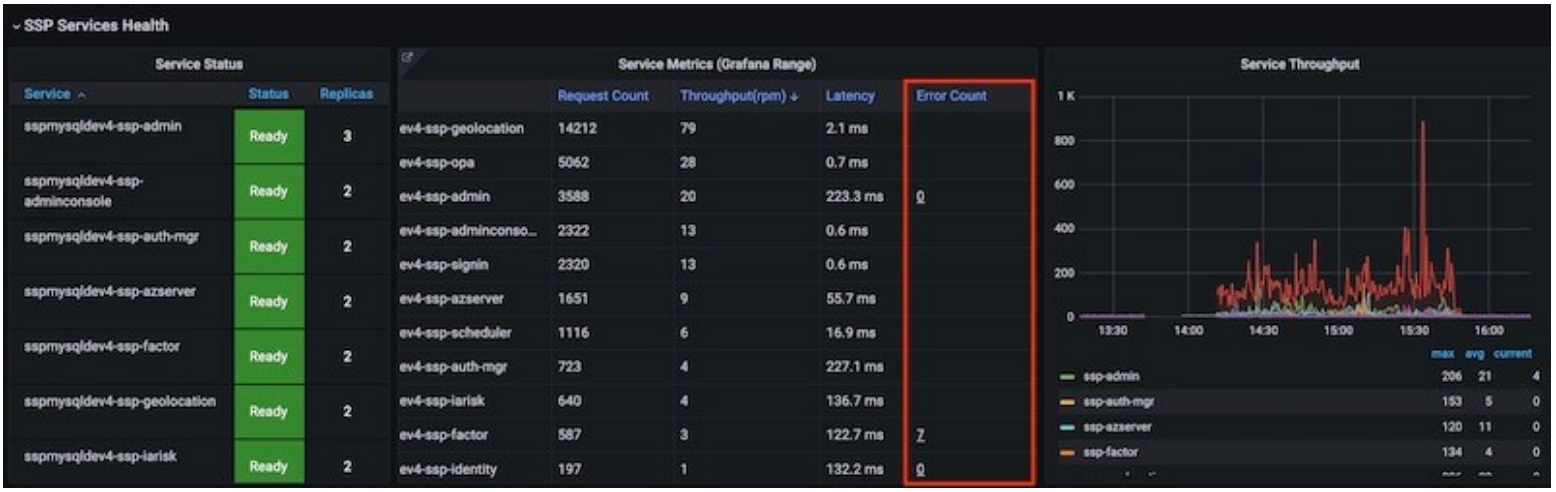
To view the IDSP (SSP) Service's health, and access it's error logs in Kibana, add the "kibanahost" and "kibanaindex" variables to the Grafana Dashboard.

Perform these steps in Grafana Dashboard page:

1. Click the ***Dashboard Settings*** (gear icon) on top right of the Monitoring Dashboard page.
2. On the Dashboard Settings, click ***Variables*** and add the values for the two new parameters, "***kibanahost***" and "***kibanaindex***" as highlighted in the following image:



3. Navigate back to the Monitoring Dashboard page, and expand the IDSP Services Health drop-down.
4. To view IDSP errors listed in the "Error Count" column, click a number to see detailed error information in the Kibana logs as highlighted in the following image.



For troubleshooting tips on monitoring, click here.

Next Step -

Integrate the Demo Application

Configuring Administrative Access

Covers the initial security credentials for the admin clients (Infra client, System client, Default Tenant client and defaultTenantAdminUser), recovery of these credentials after 48 hours, setting up an admin identity and granting permission to the admin role. All of these are provided out-of-the-box by Symantec Identity Security Platform during the initial deployment.

This topic includes the following sections:

- Retrieving Initial Credentials for the clients
- Recovering Initial Credentials after 48 hours (Break-Glass Recovery)
- Setting Up Your Admin Identity
- Granting administrative privileges to the Tenant Admin Role

Retrieving Initial Credentials for the clients'

NOTE
The commands to retrieve the initial credentials use environment variables that are expected to be set in your environment. Before running those commands, ensure that these environment variables are set for your terminal session. You can do so by running the following command:

```
echo RELEASENAME=\'${RELEASENAME}\', NAMEPSPACE=\'${NAMESPACE}\' ; if [ -z "${RELEASENAME}" ] || [ -z
 "${NAMESPACE}" ]; then echo "***** ERROR: RELEASENAME or NAMESPACE environment variables are not set
 *****" ;fi
```

The following initial credentials can be retrieved within 48 hours from the time of deployment by using the following commands:

Client	Role (AZ Policy)	Tenant Name	Steps to acquire Client ID
InfraClient	Infra Admin	System	kubectl get secret ${RELEASENAME}-ssp-secre jsonpath="{.data.clientId}" | base64 --decode; ech kubectl get secret ${RELEASENAME}-ssp-secre jsonpath="{.data.clientSecret}" | base64 --decode
SystemClient	System Admin	System	kubectl get secret ${RELEASENAME}-ssp-secre jsonpath="{.data.clientId}" | base64 --decode; ech kubectl get secret ${RELEASENAME}-ssp-secre jsonpath="{.data.clientSecret}" | base64 --decode
DefaultTenantClient	Tenant Admin	Default	kubectl get secret ${RELEASENAME}-ssp-secre jsonpath="{.data.clientId}" | base64 --decode; ech kubectl get secret ${RELEASENAME}-ssp-secre jsonpath="{.data.clientSecret}" | base64 --decode
DefaultTenantAdminUser You can use this user to login to the admin console and configure the default tenant. The validity of this admin user is limited to 48 hours by default, starting from the deployment time.	Tenant Admin User	Default	echo TENANT_ADMIN_USER=$(kubectl get sec defaulttenantadminuser -n ${NAMESPACE} -o js echo TENANT_ADMIN_USER_PASSWORD=$(k secret-defaulttenantadminuser -n ${NAMESPACE decode)

NOTE
To know more about these bootstrap clients, see Administrative Model.

Recovery of Credentials after 48 hours (Break-Glass Recovery)

If the credentials are not retrieved or an alternate administrative identity was not created and stored within the 48-hour validity period, an administrators can perform a break-glass recovery procedure to enable the default tenant admin user, reset its password and recreate its Kubernetes secret (which is again valid for the next 48 hours only).

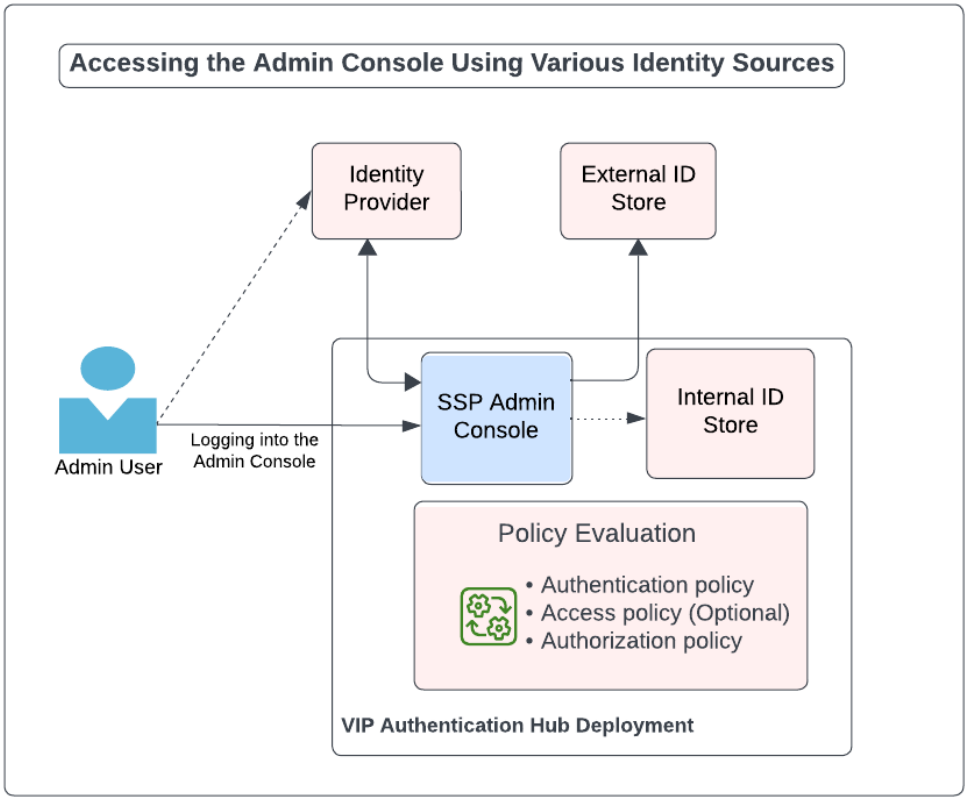
See Recovering Initial Admin Credentials section to perform a break-glass procedure.

Now, access the IDSP Admin Console by authenticating with the admin user and password retrieved in the previous step. After accessing the admin console, you will be required to ***change the admin user's password*** and ***setup the new admin identity.***

To know how to ***change the admin user's password*** and ***setup the new admin identity***, see Managing Tenant Admin Access to the Admin Console with a new user.

Setting-Up Your Admin User Identity

Admin user identities can originate from various sources to access the IDSP Admin Console. This has been explained in the following illustration along with its flow description.



Flow Description:

1. An admin user identity can come from the following user identity sources to log in to the Admin Console:
   - **Internal Identity store** - The internal identity store is deployed automatically as part of the IDSP deployment. The incoming user identity (users or groups) is stored in the IDSP database.
   - **External Identity Provider (BYOI)** - The user identity can come from an external SAML or OIDC Identity provider. The user could be JIT-created in the internal ID store, or configured to use a Zero Footprint (ZFP) mode.
   - **External Identity store (LDAP/SCIM)** - The user Identity can come from an LDAP or a SCIM configured Identity store.

2. Once the user identity is identified, it is evaluated against the Authentication policy, Access policy (if enabled), and Authorization policy.

   NOTE
   Before the policy evaluation takes place, ensure that the user is created in one of the previously stated identity sources and linked to the Authentication policy, Access policy (if enabled), and Authorization policy. Only when the user details are added to these policies, the evaluation is successful.
3. Upon evaluation, the user is provided with a list of authentication factors. Once the user authenticates successfully, access is granted to the admin console.

Use one of the following ways to set up the new admin identity:

- ***Use an existing Identity Store***: Use the Admin Console UI to define an Identity Store pointing to your existing Identity Store. The identity store can be accessed using LDAP or SCIM protocols.
- ***Use IDSP's Internal Identity Store:*** To create a user in the Internal Identity Store- see the Creating Users topic or you can view Create a new user and group in IDSP Internal Identity Store topic to check how to perform these steps from the Admin Console UI.
- ***Use Federated Identity***: Use the Admin Console UI to define an Identity Provider (OpenID or SAML) to be used as the Identity Source in a ZeroFootPrint configuration. See Using Identity Federation to Access the Admin Console.

Granting administrative privileges to the Tenant Admin Role

Based on the identity source type that you have chosen above to set up your administrative identity, use the Admin Console UI or the IDSP APIs to perform the following:

1. Identify the "***user***" or a "***group"*** from the ID store or IDP configured in the previous step to which you want to grant administrative privileges.
2. Grant administrative privileges (Tenant Admin Role) to the user or group. See an example of Granting Admin Privileges (with API usage) here, and to know more about the Authorization APIs, see Authorization Policy APIs. For Admin Console UI usage, see Managing Tenant Admin Access Using Admin Console.
3. Update the Authentication Policy (AdminLoginPolicy) for the Admin Console. See an example of Updating the Authentication Policy for Admin Console using the IDSP API here, and to know more about the Authentication APIs, see Authentication Policy APIs. For Admin Console UI usage, see here .

   NOTE
   Ensure that the identity store configured as your admin identity is correctly mapped to the "Admin Console Identity Store Assignment" tenant setting in the admin console. In the API context, this setting is called "***adminConsoleIdStoreToBeUsed***" and is detailed on the Managing Settings page.

Managing Tenant Administrator Access Using Admin Console

Following these steps ensures seamless administration by transitioning from the "DefaultTenantAdminUser" to a newly created Admin User/Group with assigned TenantAdministrator privileges.

The "**DefaultTenantAdminUser**" is an administrative user with Tenant Administrator privileges, created during Symantec Identity Security Platform deployment. For security reasons, this user is automatically disabled after 48 hours, and its credentials stored in Kubernetes secrets, are also deleted.

To ensure uninterrupted operations, a new user with the same privileges should be created before the "**DefaultTenantAdminUser**" is disabled.

This document outlines the steps to:

- Create a new user and group in IDSP Internal Identity Store
- Assign TenantAdministrator permissions to the new group
- Update the Authentication Policy to recognize the new administrator group
- Log in using the newly created administrator user

Creating a User and Group in IDSP Internal Identity Store

Log in to the IDSP Admin Console.

1. Log in to the IDSP Admin Console (https://${SSP_FQDN}/default/ui/v1/adminconsole/) using the default credentials for the DefaultTenantAdminUser.
   - Username: DefaultTenantAdminUser
   - Password: (Stored in Kubernetes secret)
2. To get the DefaultTenantAdminUser's password, use the following kubectl command:
   ```
   kubectl get secret ${RELEASENAME}-ssp-secret-defaulttenantadminuser -n ${NAMESPACE} -o
   jsonpath="{.data.userPassword}" | base64 --decode
   ```
3. Click Sign In.
   The Change Password dialog is displayed.
4. Enter a New Password, confirm it, and click Change Password.
   You are now logged into the IDSP Admin Console Dashboard.

Creating an Admin Group

1. Navigate to Group from the left panel.
2. Select Groups and click Add Group.
3. Enter a Group Name (for example, Admins).
4. Click Save.

Creating a New User

1. Navigate to Users from the left panel.
2. Select Users and click Add User.
3. Provide the required details:
   - First Name: Your first name.
   - Last Name: Your last name
   - Username: This is the name used to login to the Admin Console. By default the "Use Primary Email as the Username" option is selected.
   - Email: Your email address.
   - Phone Number: This is required if your SMTP provider is not configured yet. To receive the password reset link, specify a phone number that is capable of receiving text messages.
4. Click Save.
   The newly created user appears in the Users page.

Adding the Newly Created User to "Admins" Group

1. Navigate to Groups from the left panel.
2. In the search bar, type Admins and click Submit.
   The newly created "Admins" group gets displayed.

   NOTE
   You can search for the user by specifying the initial letters of the username ("starts with" filter), or choose "Show Filters" to add additional filters.
3. Click "Admins" and then click Add.
4. On the Add Users to Admin Group page, search for the newly added user by its name or first few letters of the name.
5. Select newly added user and click Add.
   The user now becomes the member of "Admins" group.

Assigning the Tenant Administrator Role to "Admins" Group

Steps to assign Tenant Administrator Role

1. Navigate to Settings > Administrators Roles > Roles.
2. In the search bar, search for the Tenant Admin Role.
   The Tenant Admin Role is displayed.
3. Select User Group.
4. Choose internal (Internal Identity Store) as the Identity Provider.
5. Enter "Admins" or click Select Group to search for it.
6. Click Grant.
   The "Admins" group is now listed as a Tenant Administrator.

Updating the Authentication Policy to Assign "Admins" group as the Tenant Administrator

Steps to update the Authentication Policy with the new administrator user/group so that it is recognized for future logins

1. Navigate to Authentication Policies in the left panel.
2. Select the "AdminLoginPolicy".
3. Click Add Rule.
4. Add a description and select authentication rule type, click Next.
5. Click Add Conditions, and select Groups.
6. Select the Identity Store as Internal.
7. Add the group by clicking on Select Group option.
8. Select "Admins", click Select, and click Next.
9. On the Result page, click Next.
10. In Authentication Factors, select the required methods, and click Next.
11. Review the Summary for Authentication Rule and click Done.
12. Click Save Policy.

Logging into the Admin Console with the New User

Steps to Log in as the newly created user in the Admin Console.

1. Wait 30 seconds for the changes to refresh.
2. Log in to the Admin Console using the user's username that you selected during the user creation.
3. The user is locked by default. Click Unlock Account.
   The Unlock Your Account and Reset Password page appears.
4. Select a password reset method:
   - Email: The password reset link would be sent to your email address. This option would only work if you already configured your SMTP provider. (SMTP Configuration)
   - SMS: The password reset link would be sent to your phone number that was provided during the user creation.
5. Follow the reset link from the SMS to update the password.
6. Once reset, return to the Admin Console Sign-In page.
7. Log in with the updated password.
   You are logged into the Admin Console and the dashboard page is displayed.

Upgrading Symantec Identity Security Platform

Learn how to upgrade and rollback Symantec Identity Security Platform deployment.

The In-place IDSP upgrade enables you to upgrade the solution's deployment and handles database schema updates automatically.

To know more about database upgrade, see the Database Considerations page.

Prerequisite for upgrading IDSP charts

- Ensure to upgrade your Ingress-Nginx to version 4.14.3 by using the following command:

```
helm get values ingress-nginx -n ingress >ingress-override.yaml
helm upgrade ingress-nginx -n ingress ingress-nginx/ingress-nginx --version=4.14.3 -f ingress-override.yaml
 \
--set controller.allowSnippetAnnotations=true \
--set controller.config.annotations-risk-level=Critical
```

  For information on why "--set controller.config.annotations-risk-level=Critical" should be added, see Deploying Ingress Controller.
- **Prior to upgrading Symantec Identity Security Platform, you must do the following**
  - Get the user-supplied current IDSP Infra Helm deployment values as instructed in step 7.
  - Get the user-supplied current IDSP Helm deployment values as instructed in step 11.
  - If you intend to upgrade the IDSP Data (ssp-data) helm chart, get the user-supplied current IDSP Helm deployment values as instructed in step 15.

Upgrade Considerations

IMPORTANT

- **IDSP Supports Incremental Upgrades**:
  IDSP facilitates incremental upgrade paths. If you plan to upgrade to the latest release, IDSP v4.0.2, and you are currently using an earlier version, it is advisable to first upgrade to the latest patch of the subsequent minor release before moving on to IDSP 4.0.2.
  IDSP uses the following format for naming release versions: MAJOR.MINOR.PATCH. For example, in the release version 4.0.x, "4" is the MAJOR version, "0" is the MINOR version, and "x" is the PATCH version.
  Upgrade Use Cases:
  - ***Upgrade Scenario 1***: Upgrade from IDSP v3.3 to IDSP v4.0
    Recommended Upgrade Path: You should follow this upgrade path- **3.3.x > 3.4.x > 4.0.2**
    Example: 3.3.6 > 3.4.8 > 4.0.2
  - ***Upgrade Scenario 2***: Upgrade from IDSP v3.4.x to IDSP v4.0.x
    Recommended Upgrade Path: You can directly proceed with the upgrade to v4.0.2 without any intermediate steps.
    Example: 3.4.8 > 4.0.2
- The dataseed job may take longer than usual during the upgrade due to the `ALTER TABLE` statement on the `T_USER` table. The duration depends on the number of records in the table. Plan the maintenance window accordingly.
- **Oracle SQL file Updates**: This release includes Oracle database schema changes that may require manual SQL file updates for environments with separate database administration. For detailed upgrade procedures and SQL file application order, see the Oracle SQL File Updates.

See Upgrading Symantec Identity Security Platform.

Retrieving Environment Variable values required for managing IDSP deployment

- To retrieve the values for Release Name and Namespace that were used during the deployment, use the following command:

```
helm ls --all-namespaces
```

In the returned result, look for a deployment that uses the IDSP(IDSP) chart to find the Release Name (NAME Column), and the Namespace (NAMESPACE Column).

After retrieving these values, set them as environment variables using these commands:

```
export RELEASENAME="<RELEASENAME>"
export NAMESPACE="<NAMESPACE>"
```

- To retrieve the value for IDSP(IDSP) Version, use the following command:

```
#use this command to extract and save the current version of the IDSP chart
export SSP_VERSION=$(helm list --filter="${RELEASENAME}$" -n ${NAMESPACE} -o yaml |grep chart:| cut -d'-' -f2-)&&echo version=${IDSP_VERSION}
```

Perform a version upgrade of the IDSP

Follow these steps:

1. As a precaution, it is always recommended to take a backup of the mysql operation db before the upgrade. If you are using a DB service at your cloud provider, refer to the cloud provider's documentation to find out how to involve an On-demand backup. It is also a good practice to periodically perform a restore-from-backup into a non-production environment to ensure your backups are good and that you are familiar with the restore procedure. For example:
   - GCP - https://cloud.google.com/sql/docs/mysql/backup-recovery/backing-up#console
   - AWS - https://aws.amazon.com/getting-started/hands-on/amazon-rds-backup-restore-using-aws-backup/ or https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/USER_CreateSnapshotCluster.html
   - Azure - https://learn.microsoft.com/en-us/azure/mysql/flexible-server/how-to-trigger-on-demand-backup
   - Percona MySQL DB - https://docs.percona.com/percona-operator-for-mysql/pxc/backups-ondemand.html
2. Set the Release Name and Namespace of the IDSP service that was used during the deployment as environment variables:

   ```
   export RELEASENAME="<RELEASENAME>"
   export NAMESPACE="<NAMESPACE>"
   ```

   where,

   The ***Release Name*** and ***Namespace*** values match the ones that were used during the solution's deployment.

   See Configuring the Environment Variables for more details.

   NOTE

   In case you do not remember the values for Release name and Namespace that were used during the deployment, use the following command to get those values:

   helm ls --all-namespaces
3. To get the versions of the charts that are currently deployed, run this command:

   ```
   helm ls -n ${NAMESPACE}
   ```

```
# helm ls -n ssp1
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART                   APP VERSION
db-ssp          ssp1            1               2021-11-15 04:43:21.932623178 -0500 EST deployed        mysql-1.6.7             5.7.30
ssp634          ssp1            1               2021-11-15 04:45:01.433356477 -0500 EST deployed        ssp-1.0.2430            1.0.2430
ssp634-infra    ssp1            1               2021-11-15 04:44:30.726934371 -0500 EST deployed        ssp-infra-1.0.2430      1.0.2430
```

4. Ensure that the helm is pointing to the correct IDSP helm repository where the charts can be found:

   ```
   helm repo add ssp_helm_charts https://ssp_helm_charts.storage.googleapis.com
   ```

   NOTE

   It is safe to run the preceding command more than once even if the repo was already added before.
5. Get the latest information about the IDSP charts from the chart repositories using this command:

   ```
   helm repo update
   ```
6. To get the list of all available helm chart versions in the IDSP helm chachart repo, run this command:

   ```
   helm search repo ssp_helm_charts --versions
   ```

```
# helm search repo ssp helm charts --versions
NAME                                    CHART VERSION   APP VERSION     DESCRIPTION
ssp_helm_charts/ssp                     1.0.2430        1.0.2430        2021.Nov.01 - Helm chart to deploy Security Ser...
ssp_helm_charts/ssp                     1.0.2287        1.0.2287        2021.Jul.01 - Helm chart to deploy Security Ser...
ssp_helm_charts/ssp                     1.0.2002        1.0.2002        2021.Apr.01 - Helm chart to deploy Security Ser...
ssp_helm_charts/ssp                     1.0.1925        1.0.1925        2021.Feb.01 - Helm chart to deploy Security Ser...
ssp_helm_charts/ssp-data                1.0.1017        1.0.1017        2021.Nov.01 - Helm chart to deploy Security Ser...
ssp_helm_charts/ssp-data                1.0.1015        1.0.1015        2021.Jul.01 - Helm chart to deploy Security Ser...
ssp_helm_charts/ssp-data                1.0.1013        1.0.1013        2021.Mar.01 - Helm chart to deploy Security Ser...
ssp_helm_charts/ssp-data                1.0.1012        1.0.1012        2021.Feb.01 - Helm chart to deploy Security Ser...
ssp_helm_charts/ssp-infra               1.0.2430        1.0.2430        2021.Nov.01 - Helm chart to deploy Fluent Bit, ...
ssp_helm_charts/ssp-infra               1.0.2287        1.0.2287        2021.Jul.01 - Helm chart to deploy Fluent Bit, ...
ssp_helm_charts/ssp-infra               1.0.2002        1.0.2002        2021.Apr.01 - Helm chart to deploy Fluent Bit, ...
ssp_helm_charts/ssp-infra               1.0.1925        1.0.1925        2021.Feb.01 - Helm chart to deploy Fluent Bit, ...
ssp_helm_charts/ssp-sample-app          1.0.2430        1.0.2430        2021.Nov.01 - Helm chart to deploy a sample app...
ssp_helm_charts/ssp-sample-app          1.0.2287        1.0.2287        2021.Jul.01 - Helm chart to deploy a sample app...
ssp_helm_charts/ssp-sample-app          1.0.2002        1.0.2002        2021.Apr.01 - Helm chart to deploy a sample app...
ssp_helm_charts/ssp-sample-app          1.0.1925        1.0.1925        2021.Feb.01 - Helm chart to deploy a sample app...
ssp_helm_charts/ssp-symantec-dir        1.0.2430        1.0.2430        2021.Nov.01 - Helm chart to deploy a Demo ID St...
ssp_helm_charts/ssp-symantec-dir        1.0.2287        1.0.2287        2021.Jul.01 - Helm chart to deploy a Demo ID St...
ssp_helm_charts/ssp-symantec-dir        1.0.2002        1.0.2002        2021.Apr.01 - Helm chart to deploy a Demo ID St...
ssp_helm_charts/ssp-symantec-dir        1.0.1925        1.0.1925        2021.Feb.01 - Helm chart to deploy a Demo ID St...
```

7. To get the user-supplied deployment values that were used during the deployment of the IDSP infra chart and save it in the *"infra-override.yaml" file, run this command:*

   ```
   helm get values infra-${RELEASENAME} -n ${NAMESPACE} >infra-override.yaml
   ```

   NOTE
   The infra-${RELEASENAME} is the helm release name of the ssp-infra deployment.

8. To switch to using the new log outputs configuration parameters, perform these steps:

   NOTE
   The old log output configuration parameters are deprecated and will not be supported in future releases. Although not required during the upgrade, ensure you switch to using the new configuration parameters before upgrading to the next release.

   a. Back up the existing config map using this command:

      ```
      kubectl get configmap ssp-infra-config -n ${NAMESPACE} -o yaml >fluent-bit-config.yaml
      ```

   b. Edit the ***infra-override.yaml*** file created in the previous step to include the new customConfig configuration parameters under the fluent-bit section. See Logs and Security Audit Data.

   c. Add the annotation, `configVersion: 1.0` to the existing fluent-bit config map (deployed by the IDSP Infra helm chart).
      This annotation indicates to the chart to start using the new customConfig configuration parameters.

      ```
      kubectl annotate configmap ssp-infra-config -n ${NAMESPACE} configVersion=1.0
      ```

9. To upgrade to the latest version of IDSP infra chart, run this command:

   ```
   helm upgrade infra-${RELEASENAME} ssp_helm_charts/ssp-infra -n ${NAMESPACE}  -f infra-override.yaml
   ```

   To upgrade to the latest infra chart using new docker credentials, run this command:

   ```
   helm upgrade infra-${RELEASENAME} ssp_helm_charts/ssp-infra -n ${NAMESPACE}  -f infra-override.yaml \ --
   set
    ssp.global.ssp.registry.credentials.username=<DockerHubUserName>,ssp.global.ssp.registry.credentials.password=<Doc
   ```

   To upgrade to a specific version of IDSP infra chart, run this command:

   ```
   helm upgrade infra-${RELEASENAME} ssp_helm_charts/ssp-infra -n ${NAMESPACE}  -f infra-override.yaml --
   version <CHART VERSION>
   ```

   <CHART VERSION> -This is the value that appears in the output of the helm search (helm search repo ssp_helm_charts --versions) command.

10. Check if the Prometheus adapter is deployed in your cluster, if yes, uninstall it from the cluster.

    a. Assuming that the namespace used to deploy Prometheus adapter is "prometheus-adapter", run this command:

       ```
       helm ls -n prometheus-adapter
       ```

NOTE
If you are not sure about the namespace, run "helm ls -A" to get a list of all the helm charts deployed in your cluster.

b. If the Prometheus adapter is deployed, uninstall it by running this command:

```
helm uninstall <prometheus-adapter-releasename> -n <proemtheus-adapter-namespace>
```

NOTE
The default behavior when using helm without specifying any "--set" options in the upgrade command is to reuse the previously specified values (--reuse-values).

11. To get the user-supplied deployment values that were used during the deployment of the IDSP chart and save it in the *"ssp-override.yaml" file, run this command:*

```
helm get values ${RELEASENAME} -n ${NAMESPACE} >ssp-override.yaml
```

12. If you are upgrading from an older version to IDSP 4.0.2 or newer, you must migrate your legacy direct backend routes (`oauth2` , `identitysvc` , `adminsvc` , etc.) to the new `ssp-router` architecture to avoid downtime.
The `ssp-router` architecture is highly recommended as it simplifies configuration into a single route, centralizes cluster routing, ensures consistent path rewriting, and is required for Gateway API support.
Step by step migration:
    a. **Deploy with parallel routes**: Run your initial upgrade with `keepLegacyRoutes: true` . This safely creates both routing sets,(Both legacy + ssp-router routes are active).

```
ssp:
  ingress:
    type: nginx
    ingressClassName: nginx
    keepLegacyRoutes: true
```

    b. **Validate**: Test the environment to ensure the new `ssp-router` routes are handling traffic properly.
    c. **Clean up legacy routes**: Run a final upgrade with `keepLegacyRoutes: false` to remove the old direct backend routes (Only ssp-router routes remain active).

```
ssp:
  ingress:
    type: nginx
    ingressClassName: nginx
    keepLegacyRoutes: false
```

13. **Choose one of the following upgrade types:**
    – (**Latest version Upgrade**) - To upgrade to the latest version of IDSP chart, run this command:

```
helm upgrade ${RELEASENAME} ssp_helm_charts/ssp -n ${NAMESPACE}  -f ssp-override.yaml --set
 ssp.featureFlags.dataseed.enabled=true --timeout=120m
```

    – (**Specific Version Upgrade**) To upgrade to a specific version of IDSP chart, run this command:

```
helm upgrade ${RELEASENAME} ssp_helm_charts/ssp -n ${NAMESPACE}  -f ssp-override.yaml --version <CHART
 VERSION> --set ssp.featureFlags.dataseed.enabled=true --timeout=120m
```

    Where,<CHART VERSION> -This is the value that appears in the output of the helm search (helm search repo ssp_helm_charts --versions) command.
    – (**Oracle DB SQL Manual Updates**) - The database schema is typically upgraded automatically as part of the dataseed job. In some instances, the **Oracle DB instance must be updated manually** outside of the regular process because the database administration is separate from the IDSP deployment.
    If manual updates of the schema are required, ensure to run the DB schema files (as discussed Oracle SQL File Updates) before upgrading IDSP. Once the DB files have run successfully, now you can upgrade the IDSP chart by enabling the property, "ssp.featureFlags.dataseed.skipSchemaInit" as shown here:

```
helm upgrade ${RELEASENAME} ssp_helm_charts/ssp -n ${NAMESPACE}  -f ssp-override.yaml --set
 ssp.featureFlags.dataseed.enabled=true --set ssp.featureFlags.dataseed.skipSchemaInit=true --
timeout=120m
```

NOTE

- When running the "helm upgrade" to upgrade to a new version of IDSP, the command must have the "ssp.featureFlags.dataseed.enabled" set to true. The "dataseed" job takes care of updating the database schema and the IDSP configuration data.
- The "--timeout=120m" parameter increases the helm timeout for running the "dataseed" job. This is done to ensure that the helm upgrade does not time out prematurely if the database upgrade steps take longer due to high volumes of data in database tables or slowness of the database. In case the helm command times out, it is important to check the status of the dataseed job and wait until it completes, before running the upgrade command again.
- When the IDSP deployment is in progress, **refrain from interrupting it**. You can open another terminal to watch the status of the dataseed job by running the command given in this section. If the deployment timeout expires before the dataseed job pod completes, let the job's pod continue to run, and you could watch the status of the dataseed job by using this command:

  ```
  kubectl get pods -n ${NAMESPACE} -l job-name=${RELEASENAME}-ssp-dataseed-job
  ```

  To get the Pod details, use this command:

  ```
  kubectl describe pods -n ${NAMESPACE} -l job-name=${RELEASENAME}-ssp-dataseed-job
  ```

  To get the log of the schema update container, use this command:

  ```
  kubectl logs -n ${NAMESPACE} -l job-name=${RELEASENAME}-ssp-dataseed-job -c ssp-db-init --
  tail=-1
  ```

  To get the log of the data update container, use this command:

  ```
  kubectl logs -n ${NAMESPACE} -l job-name=${RELEASENAME}-ssp-dataseed-job -c ssp-db-dataseed --
  tail=-1
  ```

- If the dataseed encounters an error, capture the logs and share them with support. The Job ID numbers in the logs confirm that the deployer checked the logs for details on the error. If the dataseed fails to start (for example, due to an incorrect registry secret), it is not treated as an error since no logs get generated. Once the initial issue is identified and resolved, delete the dataseed pod using

  ```
  kubectl delete pods -n ${NAMESPACE} -l job-name=${RELEASENAME}-ssp-dataseed-job
  ```

  and redeploy the IDSP helm chart.

14. To verify that the IDSP chart was upgraded to the desired version, run this command:

    ```
    helm ls -n ${NAMESPACE}
    ```

15. To validate the deployment, run the following command:

    ```
    curl --insecure https://"${SSP_FQDN}"/default/.well-known/openid-configuration
    ```

    NOTE

    If the validation fails, make sure to restart the ingress pods. Due to specific SSL configurations in the Ingress objects, sometimes these settings may require a restart of the ingress pods. Use the command:

    ```
    kubectl rollout restart deployment ingress-nginx-controller -n ingress
    ```

16. To get the user-supplied deployment values that were used during the deployment of the IDSP data chart and save it in the *"ssp-data-override.yaml" file, run this command:*

    ```
    helm get values ${RELEASENAME}-data -n ${NAMESPACE} >ssp-data-override.yaml
    ```

17. To upgrade the "ssp-data" chart to the latest version, run this command:

    NOTE

    - The "ssp-data" chart is updated once a week, therefore to receive the most recent data, follow the procedures that are outlined here and upgrade to the most recent chart.
    - Ensure that the database has at least 20GB of space to hold the network data information.

```
helm upgrade ${RELEASENAME}-data ssp_helm_charts/ssp-data -n ${NAMESPACE}  -f ssp-data-override.yaml
```

18. To verify that the "ssp-data" chart was upgraded to the desired version, run this command:

```
helm ls -n ${NAMESPACE}
```

The steps to perform a version upgrade of the IDSP chart are complete.

Perform a configuration update of the IDSP helm chart

To know about performing a Configuration update of the IDSP helm chart, see Managing Configuration Updates to the Symantec Identity Security Platform Helm Chart.

Steps to rollback IDSP deployment

Helm maintains a full history of the different revisions of each deployment. A revision can be a ***version-upgrade,*** or just a deployment ***configuration update*** (upgrade to the same version with a config change). Helm allows you to revert to any previous revision of the deployment. Helm does not distinguish between helm upgrade revisions that were used to update the deployment configuration and those that performed a version upgrade.

IDSP supports a rolling back for a single version upgrade, which means that even if the database has already been migrated to the new version, you can roll back the chart to the previous version. This limitation is only referring to an IDSP version rollback. You can revert (rollback) as many "configuration update" revisions as you need, but you should ensure that you are not reverting more than **one "version upgrade"**.

Follow these steps:

1. If you choose to revert to the previous version of the IDSP at any point following the upgrade, you must roll back the Helm version by running these commands:

```
#get history of revisions deployed
helm history <SSP_RELEASE_NAME> -n <SSP_NAMESPACE>

#rollback to the previous (or a specific revision)
helm rollback <SSP_RELEASE_NAME> [REVISION] -n <SSP_NAMESPACE>
```

   Where,

   - the ***<SSP_RELEASE_NAME>*** and ***<SSP_NAMESPACE>*** values should match with the ones that were used during the solution's deployment.

     NOTE
     In case you do not remember the values for Release name and Namespace that were used during the deployment, use the following command to get those values:
     helm ls --all-namespaces
   - ***[REVISION]*** should be replaced with the specific number of the IDSP deployment in the Kubernetes environment you want to roll back to.

2. To verify that you have rolled back to the intented version of IDSP and that the rollback was completed successfully, run the following commands:

```
#check the revision list again
helm history ${RELEASENAME} -n ${NAMESPACE}

#check the current version deployed
helm ls -n -n ${NAMESPACE}

#ensure that all the pods are in ready state
kubectl get pods -n ${NAMESPACE}
```

   The following command lists all the pods in the namespace with their current version:

```
kubectl get pods -n ssp -l app.kubernetes.io/part-of=ssp --template='{{range .items}}{{.metadata.name}}
      {{"\t"}}{{index .metadata.labels "app.kubernetes.io/version"}} {{"\t"}}{{.status.phase}}{{"\n"}}
{{end}}'| column -t -s $'\t'
```

Reference Links:

- Upgrading Symantec Identity Security Platform Sample Web App

Uninstalling the Solution and Enclave Services

Learn how to uninstall Symantec Identity Security Platform Services.

This topic describes the procedure to uninstall IDSP and enclave services from your Kubernetes environment.

Includes the following sections:

- Uninstalling SSP Data Chart
- Uninstalling IDSP
- Uninstalling Logging Service
- Uninstalling Monitoring Service
- Uninstalling Ingress Service

Uninstalling SSP Data Chart

The ssp-data chart depends on the IDSP chart. If you want to uninstall both charts, delete the ssp-data chart first, then the IDSP chart. If you simply want to uninstall the IDSP chart, see the "Uninstalling IDSP" section and follow the instructions.

To uninstall ssp-data chart from your environment, follow this step:

1. Run this command to uninstall ssp-data (network data loader) chart:
   ```
   helm uninstall ${RELEASENAME}-data -n ${NAMESPACE}
   ```

Uninstalling IDSP

To uninstall IDSP from your environment, follow these steps:

1. Run this command to uninstall IDSP release:
   ```
   helm uninstall <RELEASENAME> -n <NAMESPACE>
   ```
2. Run this command to uninstall the Infra helm chart:
   ```
   helm uninstall infra-<RELEASENAME> -n <NAMESPACE>
   ```
   NOTE
   Ensure that you uninstall the infra helm chart before deleting the namespace as mentioned in the steps.
3. Run this command to get the list of all namespaces deployed in the VIP cluster:
   ```
   helm ls --all-namespaces
   ```
4. After getting the list of namespaces, select the namespace that you want to delete. In this case, select a IDSP namespace and run this command to delete:
   ```
   kubectl delete ns <NAMESPACE>
   ```

Uninstalling Logging Service

To uninstall Kibana and Elastic Search from your environment, follow these steps:

1. Run this command to uninstall the Kibana service:
   ```
   helm uninstall kibana -n logging
   ```
2. Run this command to uninstall Elastic search:
   ```
   helm uninstall elasticsearch -n logging
   ```
3. Run this command to delete the logging namespace:
   ```
   kubectl delete ns logging
   ```

NOTE
If you want to start with an empty data set after uninstalling elastic, then delete all the Elastic's PVC using this command: `kubectl delete pvc <elastic-pvc-name> -n logging`

Uninstalling Monitoring Service

To uninstall Prometheus services from your environment, follow these steps:

1. Run this command to uninstall the Prometheus service:
   ```
   helm uninstall prometheus-operator -n monitoring
   ```
2. Run this command to delete the monitoring namespace:
   ```
   kubectl delete ns monitoring
   ```

Uninstalling Ingress Service

To uninstall Ingress services from your environment, follow these steps:

1. Run this command to uninstall the NGINX -Ingress service:
   ```
   helm uninstall nginx-ingress -n ingress
   ```
2. Run this command to delete the Ingress namespace:
   ```
   kubectl delete ns ingress
   ```

NOTE
When using a cloud provider (such as GKE, AKS, EKS) with the external load balancer, the next ingress installation will generate a new external IP address.

Operating

The Operating section guides you to perform your functions correctly and efficiently.

Operating the Symantec Identity Security Platform deployment involves a number of processes.

This topic includes the following sub-topics:

- Hardening the environment - see the Security Considerations section for guidelines to be applied on day one of the deployments.
- To integrate third-party tools with IDSP to capture and visualize operational and security audit data - see Managing Observability section.
- To choose the topology strategy that best suits your deployment environment, see Managing Deployments section.
- Troubleshooting the environment - see Troubleshooting section.

Security Considerations

Checklist for Securing and Hardening the Symantec Identity Security Platform Environment.

This is the checklist for Securing and Hardening IDSP environment.

Area	Consideration	Overview	Recommendation
Infrastructure	Kubernetes	Kubernetes security enables protecting a cluster from accidental or malicious access. Applicable Docs: • Overview of Cloud-Native Security • Securing a Cluster	• Understand & follow Kubernetes security best practices • Partner with Kubernetes security SME to develop and follow security guidelines
	External Ingress Access	An ingress controller is deployed to expose solution endpoints. By default, a single ingress exposes both the business services and management services. A separate ingress controller can be deployed to expose management services Applicable Docs: Deploying Ingress	• Obtain and deploy an external ingress certificate for your public-facing host. • If you intend to create air gap between the public-facing external ingresses and internal admin and/or management ingress, then obtain and deploy external ingress certificate for internal ingresses.
	Communications to external Operational Database (MySQL)	SSL-based connection with the following options: • DISABLED • PREFERRED • REQUIRED (**default**) • VERIFY_CA • VERIFY_IDENTITY Applicable Docs: Using Database in SSL Mode.	• Ensure SSL is enabled using the highest security setting supported by the database server. • Mutual SSL for MySQL to validate client certificate is not supported.
	Native AWS Authentication to AWS Aurora MySQL	Replaces DB password by AWS IAM as per IAM Database Authentication Applicable Docs: Configuring Connectivity to AWS MySQL DB Without Using a DB Password	• Use this AWS feature to eliminate the use of static DB passwords.

Use of Instance Metadata Service Version 2 (IMDSv2)	IDSP supports both IMDSv1 and IMDSv2 instances in an AWS cluster. However, it is recommended to use IMDSv2 for its enhanced security.	• Use IMDSv2 instances in an AWS cluster for its enhanced security.
Master Data Encryption Key (MEK	MEK is used to encrypt the Tenant Encryption keys which in turn encrypt tenant-specific data, including the system tenant. By default, MEK is auto-generated during the initial deployment. A custom MEK can be specified during the initial deployment. It is not possible for the solution to use the Operational Database without the matching MEK. Applicable Docs: Masterkey Encryption	• If using an automatic MEK created during initial deployment, must have it backed up and stored in a secure location immediately after the deployment. • If using a custom MEK, ensure it is being backed up in a secure location. • If restoring the Operational Database to another location, the original MEK must also be deployed in that location. MEK rollover is not supported.
Internal cluster communications	All internal communications traffic between micro-services is authenticated using an Access Token signed by the internal signing key (ISK). All internal communications traffic between internal microservices is SSL/VERIFY_CA using Internal Signing Key & Certificate. All internal communications traffic between ingress and micro-services is SSL/VERIFY_CA with additional server namespace verification by ingress. Internal services leverage the TLS certificate signed by the internal, self-signed root CA. A custom TLS key/certificate and CA certificate can be specified. Applicable Docs: Intra Cluster Communication	• If an SSL cluster communication uses your own CA, during the SSP helm chart deployment, specify the name of the Kubernetes secret that contains the following files, which represent the ISK: – tls.key file that holds your private key in PKCS8 format – tls.crt file that holds one or more server certificates in PKCS8 format – ca.crt file that holds one or more trusted CA certificates in PKCS8 format **Note**: The PKCS8 format uses the "BEGIN PRIVATE KEY / END PRIVATE KEY" and "BEGIN CERTIFICATE / END CERTIFICATE" header/footer separator lines.
External rate limiting	As an experimental feature, SSP cluster supports rate limiting to protect against cluster overloading from malicious actors. Contact your IDSP team for instructions configuring the NGINX ingress controller.	Access to the IDSP cluster should be rate limited to protect against DDOS attacks on the cluster.

Infra Admin Access	The initial deployment seeds the "Infra admin" client (root) principal with the least amount of privileges required in "break glass" situations when a system admin account is inoperative due to lockout, mistaken policy changes, etc. The client id/secret is exposed to the deployer through the Kubernetes secret for a limited period of time after the initial deployment, and then removed automatically. This Kubernetes secret may be re-created for a limited period of time by re-running the deployment chart with specific values requesting this secret to be re-recreated. Applicable Docs: Administrative Model	• Back up and save the Infra client id/ secret in a vault after the deployment, then delete the Kubernetes secret.
System Admin Access	The initial deployment seeds the "system admin" client principal with Cross Tenant and Tenant Admin functional roles. This client is used to execute system APIs to configure the system, configure system admins, etc. Additionally, this client is used to invoke tenant-level APIs when troubleshooting/fixing tenant-level configuration issues. The client id/secret is exposed to the deployer through the Kubernetes secret for a limited period of time after the initial deployment, and then removed automatically. This Kubernetes secret may be re-created for a limited period of time by re-running the deployment chart with specific values requesting this secret to be re-recreated. Applicable Docs: • Administrative Model • Signing Keys • Deploying Ingress Controller	• Disallow invoking of system-level APIs through the external ingress (see security settings table). • Configure a dedicated ingress network interface for system management operations and use firewall rules to constrain this ingress to internal IPs. Use distinct ingress controller class to set up the management ingress, and deploy the IDSP helm chart with "--set ssp.ingress.mgmt.ingressClassName=<managemen ingress-class-name>"

	System Settings	Various security settings are available to configure the behavior of the system, such as allowSystemOpsOnExternalIngress, accessTokenExpiryMaxMins, idTokenExpiryMaxMins, etc. See table below. Applicable Docs: Settings Configurations	• Configure allowSystemOpsOnExternalIngress=false system setting to disallow system operations on the business ingress object created by the SSP helm chart. • Configure the Issuer Signing Key for the system tenant by importing the Key and Certs signed by a trusted CA. This deletes the self-signed system signing key. • Configure a CORS system setting in the system tenant to accept API requests from specific hosts. • Review and configure various security settings as applicable in your environment for system-level operations and to provide defaults to tenant-level operations.
	Functional	Tenant Admin Access	During the creation of a tenant, the process seeds the "tenant admin" client principal with a Tenant Admin functional role. This client is used to configure the tenant such as to define protected apps, manage authentication/authorization policies, etc. Access by end users can be configured. Applicable Docs: Configuring the Solution • Set up MFA access by end-user admins to ensure user-specific accountability and security. Such admins are configured in the tenant admin signin policy and are granted the Tenant Admin role. • Allow admins to federate against corporate IDP (setup SAML IDP via UI/ API or OIDC OP via API). Restricting the Tenant Admin console to the internal-only network is not supported.
	Tenant Admin Console MFA Warning	Audit event "admin.authpolicy.mfa.warning" is issued when a single rule in an authentication policy protecting the Admin Console does not contain multiple factors.	Multifactor Authentication may be achieved by having multiple policiesor rules, but as a best practice a single rule should be used to secure the Admin Console.
	Tenant Data Encryption Key (TDEK)	Currently hardcoded for the auto-created "default" tenant. A custom TDEK can be provided when creating a tenant.	TDEK rollover is not supported.

	Signin UI Frame Hardening	Harden the Signin UI application by using the deployment-time feature flag "ssp.featureFlags.uiframe.option" specified through the helm chart: • --set ssp.featureFlags.uiframe.option=UNSET (signin UI is rendered with without x-frame-options header) • --set ssp.featureFlags.uiframe.option=DENY (signin UI is rendered with X-Frame-Options: DENY header) • --ssp.featureFlags.uiframe.option: SAMEORIGIN (signin UI is rendered with X-Frame-Options: SAMEORIGIN header) Applicable Docs: • Deployment Time Property	Assess based on security requirements
	Tenant Issuer Signing Key	Each tenant has a signing key used for non-repudiation (sign issued tokens) and confidentiality (decrypt incoming tokens). A newly created tenant is initialized with a self signed key. Applicable Docs: • Signing Keys	Whenever creating a tenant, or after initial deployment, make sure to configure the Issuer Signing Key for the business tenant by importing the Key and Cert signed by a trusted CA. This deletes the previously self-signed tenant signing key.
	Anti snooping In-Transit data confidentiality (also helps in anti replay MITM)	Use in-transit data encryption to ensure end2end confidentiality of the user data collected by the signin app during the user's signin ceremony, **without** depending on SSL. Tenant Setting "fieldsToBeEncryptedInAuthFlow" (csv field containing multiple values) controls whether certain security credentials transmitted by the signin app have additional encryption when the source of SSL termination cannot be trusted. • password - user password collected by sign-in app • devicesignature - device signature collected by sign-in app Applicable Docs: • Encypting Data in Transit	Enabled additional confidentiality capability.

	Tenant Settings for Security Artifacts	Various security settings are available to configure the behavior of the tenant. Tenant Settings overwrite matching System Settings, if any.overwrit, • azCodeExpirySecs • trackAccessToken • trackIdentityToken • etc Applicable Docs: • Settings Configurations	• Configure a CORS system setting in the system tenant to accept API requests from specific hosts. • Review tenant settings to ensure alignment with applicable operational policies. Update settings at the tenant level or at the system level to align with desired settings. • Pay special attention to settings marked in the doc with "Security" tag.
	OAuth Client Authentication	As a best practice, require connecting clients to use PKI for authentication. Applicable Docs: • Client Assertion	• Use JWT Assertion as a means to authenticate OAuth clients, instead of relying on Client ID and Secret. Applies to "/token" and "/introspect" endpoints only.
	Password Storage	User Secrets to store passwords Applicable Docs: • Managing Secrets	• Instead of storing passwords as part of the resource (such as /LDAPconfigs), store the password in a Secret pointed to by an Alias
	End-User Disclosure	Prevent attackers from identifying user accounts whether valid or invalid. Applicable to user-visible signin flows, error messages, API calls made by clients executing in user browsers. Applicable Docs: • End-User Disclosure of Authentication Failures	• Enable an anti-disclosure mode to prevent attackers from identifying user accounts
	KMS-Enabled Password Encryption	Prevent accidental/targeted disclosure of credentials via password encryption done by an external KMS service. Applicable Docs: • Password Encryption	• Take advantage of the Trusted Credential Feature enabling end2end transport security and credential encryption using an external KMS

	Anti Replay Man-in-the-middle (MITM) Attack Prevention	Prevent the MITM attack on artifacts used in various flows. If mitigation is enabled, unexpected changes to such artifacts will be detected. • User Browser's "user-agent" value is made available to the server on every request via an http header. • User's IP Address as is seen by the server. This protection applies to the sign-in flow. • User's Admin Console session as is maintained by the server. This ensures that unexpected changes in user browser's User-Agent, or user's IP Address, or the user's Admin Console session will cause the server to reject incoming requests and terminate the current Admin Console session. Applicable Docs: • Protection against MITM	• Enable MITM protection to guard against replay attacks by the routing tier. • Enable MITM protection to guard against reuse of artifacts copied from another end-user device. • Enable MITM protection to guard against use of bearer tokens copied from the Admin Console session.

<table>
<tr>
<td></td>
<td>Initial Security Credentials</td>
<td>Working with bootstrap credentials:
Due to security concerns with exposing sensitive AuthHub credentials for prolonged periods of time, it is a best practice to remove K8S containing such credentials after 48 hours after the deployment. If required, such bootstrap credentials can be republished into K8S secrets by re-deploying the chart with specific flags to cause K8S secrets to be re-created for another 48-hour period.
To recreate secrets, rerun the helm chart with the CLI values as necessary to re-create the K8S secrets. While doing so, the dataseed does not need to be executed as this would not be a data upgrade, but rather just a configuration update (--set ssp.featureFlags.dataseed.enabled=false).
Helm flags controlling publishing of bootstrap credentials:
<pre>ssp:
  global:
    seedData:
      disclosureExpiryHours: "48"
      defaultTenant:
        seededOnInstall: true
        adminUser:
          seededOnInstall: true
          seededOnUpgrade: false
        adminClient:
          seededOnInstall: true
          disclosedOnInstall:
true
          disclosedOnUpgrade:
false
        demoClient:
          seededOnInstall: true
          disclosedOnInstall:
true
          disclosedOnUpgrade:
false
      systemClient:
        disclosedOnInstall: true
        disclosedOnUpgrade: false
      infraClient:
        disclosedOnInstall: true
        disclosedOnUpgrade: false</pre>
Applicable Docs:
• Configuring Administrative Access</td>
<td>• Immediately after initial deployment, save the infra and system client's credentials in a secure location.
• Make a note of the admin client (defaultTenantClient) and adminUser (defaultTenantAdminUser) for the "default" tenant bootstrapped by the initial deployment. This admin client and admin user have access to the "default" tenant seeded during deployment. If the default admin user credential is requested to be re-published later, the user account will be reset to require a password change at the initial signin.</td>
</tr>
<tr>
<td>Integration</td>
<td>Outbound SMTP provider</td>
<td>Can be configured with or without SSL</td>
<td>• Use SSL.
VERIFY_CA SSL mode is not supported.</td>
</tr>
</table>

	Outbound SMS Provider (Twilio)	SSL is enabled by default	VERIFY_CA SSL mode is not supported.
	LDAP Directory	Can be configured with or without SSL/ VERIFY_CA Applicable Docs: LDAP Configuration	• Use SSL.
	SiteMinder-AuthHub Integration	Integration between SiteMinder and IDSP is over SSL and involves the exchange of JWT-based Identity Tokens in either direction. By default, these tokens are not encrypted, but the use of JWE to encrypt these JWTs is supported. Applicable Docs: SiteMinder Integration with IDSP	• Configure JWE to encrypt Identity Token JWTs exchanged between SiteMinder and AuthHub.
	Web Hooks - SPI hooks, SCIM ID Store	External integration using HTTP APIs should be secured using strong client authentication + transport security methods, when communicating with external Business API Resource Servers and Token STS endpoints. • SSL with at least CA Verification for transport security • For APi authentication, preference should be given to OAuth2-based Client Security using PKI-based Client Assertion, rather than using client id/ secret, API key, or Basic Auth methods, whenever possible. Applicable Docs: • SCIM ID Store • User Change Event Hook	• Use Client Assertion authentication for STS integration • Use SSL in VerifyCA/VerifyIdentity Modes for end2end verification
	Database Password	When connecting to the operational database whose password is being rotated on a regular basis, there is a way for IDSP to read the database password from the Kubernetes secret when it opens a database connection. Applicable Docs: Database Considerations/Vault Handling Database Password Rotation	• Enable database password rotation to harden database connectivity.
	IP-based and Country-based policies	Verify your IDSP installation, and ensure that your network components are configured to preserve the client IP address. This is required for IP-based and Country-based policies to function correctly. Applicable Docs: Use the procedure mentioned in Verifying the deployment section to verify that the client IP address has been preserved.	

Operations/Governance	Operational Log	The operational log receives operational messages indicating ongoing operations. The log can be routed using Fluent Bit configuration to one or more upstream collector/analyzer such as Elastic, Splunk, database, etc. using the "log" data stream By default, the log is "info" with a granularity of {error, warning, info, debug, trace} supported via system-level setting. Applicable Docs: Configuring Enclave Services to Show IDSP content	• Enable operational logging by routing log data into Elastic, Splunk, etc. • Keep the log level = "info" for normal operations. Out of the box, SSL connectivity to an upstream operational log collector is not supported but is field configurable.
	Security Audit Log	The security audit log receives audit messages corresponding to ongoing operations. The log can be routed using Fluent Bit configuration to one or more upstream collector/analyzer such as Elastic, Splunk, database, etc. using the "audit" data stream. By default, the log is "info" with a granularity of {error, warning, info, debug, trace} supported via system-level setting. Applicable Docs: Configuring Enclave Services to Show IDSP content	• Enable security audit by routing audit data into Elastic, Splunk, etc. Out of the box, SSL connectivity to an upstream audit log collector is not supported but is field configurable.
	Observability / Metrics	The operational and performance metrics are emitted for collection by Prometheus Server (or compatible). The Grafana dashboard is provided to visualize such metrics. Applicable Docs: Configuring Enclave Services to Show IDSP content	• Enable observability by routing metrics data into Prometheus. Out of the box, SSL connectivity to an upstream Prometheus Server instance (or compatible) is not supported but is field configurable.

Non Disclosure of User Authentication Failures

Disclosing a reason for authentication failure is considered to be a security vulnerability. IDSP has added the following tenant-level configuration to prevent attackers from determining if a user account is legitimate or invalid:

"hideAuthenticationFailureReason" = "true"/"false"

The Administrator can decide whether to disclose complete error messages regarding authentication failures by setting the above tenant configuration.

When the "***hideAuthenticationFailureReason***" configuration is set to the following:

- if ***true*** - The reason for the failure of the authentication is not disclosed to the user.
- if ***false*** - The user is informed of the reasons for the authentication failure.

This tenant configuration is covered in Global/Tenant Settings.

The following table shows the scenarios and changes in the display of information to the user based on the tenant configuration, "hideAuthenticationFailureReason"= true/false.

Scenario	hideAuthenticationFailureReason=false	hideAuthenticationFailureReason=true

User disambiguation fails (Unique user not found)	{
	"errorCode": "8800004",
	"errorMessage": "User disambiguation failed. [8800012] [Failed to assert user login id for user input value 'malbail1']"
	}
		{
		"flowState": "RGs4eHp1d2tVMFRPUzd2RmVhQjZFTmx6NnMwZzZLWjRKb2J6bU
		"nextaction": "PASSWORD_AUTH",
		"currentFactors": [
		"PASSWORD"
		],
		"additional": {
		"userName": "malbail1",
		"currentFactorLevel": 1
		}
		}
		The user is displayed with the password authentication screen for the number of times specified in the "maxStrikeCount" configuration. When the user has reached this limit, the app returns "AUTH_DENIED".

User disambiguation fails (User Inactive)	{
	"errorCode": "8800015",
	"errorMessage": "The user in the request is not active. [8800041] [value=512:,encoding=siteminder]"
	}
		{
		"flowState": "RVE1b05CN3pTR205amVSN0E5QmpYL0hIeHVEVlI4QXpKO.....
		"nextaction": "PASSWORD_AUTH",
		"currentFactors": [
		"PASSWORD"
		],
		"additional": {
		"userName": "malbail",
		"currentFactorLevel": 1
		}
		}
		The user is displayed with the password authentication screen for the number of times specified in the "maxStrikeCount" configuration. When the user has reached this limit, the app returns "AUTH_DENIED".

Password authentication fails N times for an active user	<pre>{
	"flowState":
	"aTU0ck5QenBNRlVyS1NFQitTRDdNUmt4S3d2RU1aZkxENhaDZ4ZWtsdjI4V3RNVldLOUx3RC9zcXUxWVp2WG1vM3V...
	"userName": "MalBAIL",
	"nextaction": "AUTH_DENIED",
	"currentFactors": [
	"PASSWORD"
	],
	"data": {
	"errorCode": "0000050",
	"errorMessage": "Number of
	authentication attempts exceeded for
	User"
	},
	"additional": {
	"currentFactorLevel": 1
	}
	}</pre>
		<pre>{
		"flowState":
		"userName": "MalBAIL",
		"nextaction": "AUTH_DENIED",
		"currentFactors": [
		"PASSWORD"
		],
		"data": {
		"errorCode": "0000050",
		"errorMessage": "Number of
		authentication attempts exceeded for
		User"
		},
		"additional": {}
		}</pre>

Password authentication fails for an active user	<pre>{
	"flowState":
	"aTU0ck5QenBNRlVyS1NFQitTRDdNUmt4S3d2RU1aZ"cENhaDZ4ZWtsdjI4V3RNVldLOUx3RC9zcXUxWVp2......",
	"userName": "MalBAIL",
	"nextaction": "PASSWORD_AUTH",
	"currentFactors": [
	"PASSWORD"
	],
	"data": {
	"errorCode": "1004003",
	"message": "Password authentication
	failed: invalid credentials for user
	(MalBAIL)"
	},
	"additional": {
	"currentFactorLevel": 1
	}
	}</pre>
		<pre>{
		"flowState":
		"userName": "MalBAIL",
		"nextaction": "PASSWORD_AUTH",
		"currentFactors": [
		"PASSWORD"
		],
		"data": {
		"errorCode": "0000061",
		"errorMessage": "Invalid credentials"
		},
		"additional": {}
		}</pre>

Initiate Authentication - Success	<pre>"data": {
	"message": "Password authentication is
	successful for user mhunter"
	},</pre>
		<pre>"data": {
		"message": "Success"
		},</pre>

Managing Observability

This topic helps us in understanding how to integrate third-party tools with Symantec Identity Security Platform to capture and visualize operational and security audit data.

This topic includes the following sub-topics:

- Logs and Security Audit Data
  - Observing Nginx Ingress Traffic
- Operational Metrics Data

Logs and Security Audit Data

Managing Symantec Identity Security Platform logs and audit events

Gathering of logs on a regular basis is crucial to understand the nature of security incidents, establishing benchmarks, and identifying operational trends. IDSP uses Fluent Bit as the open source tool to collect logs and audit events, and forward them to any of the supported fluentbit log outputs, for example, logs can be streams to elastic search and views in Kibana. However, if you are already using a solution to stream logs from the Kubernetes nodes, you can use it instead of configuring fluent-bit in IDSP Infra chart. To disable the deployment of fluent-bit in the IDSP Infra chart, add "--set fluent-bit.enabled=**false**" to the Infra chart install command.

Configuring Target Log and Audit Messages

To configure the IDSP log outputs to stream log and audit messages to your log repository, create a helm values file as described below and use it when deploying the IDSP Infra Helm chart. The following example uses Kibana to display logs and audit data and the type of logs displayed are: ***ssp_log***, ***ssp_tp_log***, and ***ssp_audit***.

> **NOTE**
> For each log type certain parameters must be configured. The details of these parameters are provided here.

New customConfig configuration parameters

The new customConfig configuration parameters under the fluent-bit section in the ***IDSP infra chart values yaml*** file uses the following format:

```
fluent-bit:
  customConfig:
    outputs:
    - tags:
      <list of log tags to apply the following output  (ssp_log, ssp_audit, ssp_tp_log). >
      template: |
          Name <fluent-bit output plug-in name>
          <configuration parameters as taken from fluent-bit's outputs configuration documentation https://
docs.fluentbit.io/manual/pipeline/outputs>


   - tags:
      <list of log tags to apply the following output (ssp_log, ssp_audit, ssp_tp_log)>
      template: |
          Name <fluent-bit output plug-in name>
          <configuration parameters as taken from fluent-bit's outputs configuration documentation https://
docs.fluentbit.io/manual/pipeline/outputs>
```

Example of the customConfig configuration parameters in the *IDSP infra chart values yaml* file

Based on the new customConfig configuration parameters stated in the previous section, the following example shows the log types: ***ssp_log***, ***ssp_tp_log***, and ***ssp_audit*** that are sent to elastic search to render respective outputs in Kibana. In addition, the ***ssp_audit*** log is also sent to splunk.
Note that the Host name (elasticsearch-es-http.logging.svc) in the following example is same as the Service name and Namespace of the elastic search service.

```
sspfbService:
  ingressLogsIncluded: false
fluent-bit:
  customConfig:
```

```
    outputs:
    - tags:
      - ssp_log
      - ssp_tp_log
      - ssp_audit
      template: |
          Name es
          Host  elasticsearch-es-http.logging.svc
          Port  9200
          tls On
          tls.verify Off
          Suppress_Type_Name On
          Replace_Dots On
          Index ${tag}-%Y.%m.%d
          HTTP_User elastic
          HTTP_Passwd $(kubectl get secret -n logging elasticsearch-es-elastic-user -
o=jsonpath='{.data.elastic}' | base64 --decode; echo)
   - tags:
     - ssp_audit
     template: |
       Name splunk
       Host mysplunkhost.local
       Port  8088
       TLS   on
       TLS.Verify   off
       Splunk_Token a6c401cd-e521-413e-a690-58dba6418a06
       Splunk_Send_Raw off
       event_source http:ssp_logs
       event_sourcetype _json
```

NOTE
When creating the values yaml file from a Linux shell using a construct like 'cat << EOF', you must escape the $ sign by prepending it with a '\' to avoid **'$' sign from being interpreted as an indicator of a shell variable**, as shown in the following example:

```
cat << EOF >  logging-infra-values.yaml
sspfbService:
  parser: cri
fluent-bit:
  enabled: true
  customConfig:
    outputs:
    - tags:
      - ssp_log
      - ssp_audit
      - ssp_tp_log
      template: |
        Name es
        Host  elasticsearch-es-http.logging.svc
        Port  9200
        tls   on
        tls.verify   off
        Suppress_Type_Name On
        Replace_Dots On
```

```
                Index \${tag}-%Y.%m.%d
                HTTP_User elastic
                HTTP_Passwd $(kubectl get secret -n logging elasticsearch-es-elastic-user -
        o=jsonpath='{.data.elastic}' | base64 --decode; echo)
        EOF
```

The IDSP Infra helm chart deployment requires that you override the customConfig section with your own output configuration parameters. Once the values yaml file is created, use the "-f " parameter followed by the name of the values yaml file in the IDSP Infra Helm chart deployment command, for example:

```
-f customConfig_sspinfra.yaml
```

See Additional Parameters to add to the IDSP Infra Chart Deployment section.

Setting Environment Variables in Fluent-Bit

Some output configuration requires an environment variable to be set (see AWS credentials required by the S3 output plug-in). The following example shows how an environment variable can be set (existing feature in fluent-bit chart).

```
  fluent-bit:
   env: []
   #  - name: FOO
   #    value: "bar"

   # The envWithTpl array below has the same usage as "env", but is using the tpl function to support
  templatable string.
   # This can be useful when you want to pass dynamic values to the Chart using the helm argument "--set
  <variable>=<value>"
   # https://helm.sh/docs/howto/charts_tips_and_tricks/#using-the-tpl-function
   envWithTpl: []
   #  - name: FOO_2
   #    value: "{{ .Values.foo2 }}"
   #
   # foo2: bar2

   envFrom: []
```

See the following Example:

```
fluent-bit:
      - env:
        - name: AWS_REGION
          value: ap-southeast-2
        - name: AWS_ACCESS_KEY_ID
          valueFrom:
            secretKeyRef:
              name: fluentbit-s3-key
              key: AWS_ACCESS_KEY_ID
        - name: AWS_SECRET_ACCESS_KEY
          valueFrom:
            secretKeyRef:
              name: fluentbit-s3-key
              key: AWS_SECRET_ACCESS_KEY
```

Mounting Certificates in Fluent-Bit

To enable "tls.verify" in IDSP, you must configure Elasticsearch's CA certificate. This can be accomplished by adding "tls.ca_file /etc/certs/ca.crt" to the output parameters mentioned above, and then using "extraVolumes" and "extraVolumeMounts" to mount the configmap or secret containing the certificate.

Example of the value yaml configuration parameters required to mount a certificate from a secret:

```
fluent-bit:
    extraVolumes:
      - name: tlsconfigs
        secret:
          name: fluent-general-tls
    extraVolumeMounts:
      - name: tlsconfigs
        mountPath: /etc/certs/ca.crt
        subPath: ca.crt
```

Observing Nginx Ingress Traffic

It is critical to monitor the Kubernetes Ingress traffic to detect and manage potential issues that may occur in real time. Using third-party tools like Kibana to display log data on the dashboard helps in tracking the status of services and delving further into any analysis.

Externalizing Ingress log outputs

In Symantec Identity Security Platform, if you intent to externalize the ingress log outputs, you should create a custom config map with the required parameters.

The following example shows a sample of how to create the Ingress config map:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: ingress-nginx-controller
  namespace: ingress
data:
  error-log-level: "warn"
  log-format-upstream:
  '{"level":"info","type":"log","timestamp": "$time_iso8601", "remoteAddr": "$proxy_protocol_addr",
    "userIP": "$proxy_add_x_forwarded_for", "requestId": "$req_id", "remoteUser":
    "$remote_user", "bytes_sent": "$bytes_sent", "requestTime": "$request_time", "status":
    "$status", "vhost": "$host", "protocol": "$server_protocol", "path": "$uri","api": "$uri",
    "reqLength": "$request_length", "method": "$request_method", "httpReferrer": "$http_referer",
 "userAgent": "$http_user_agent", "contentType": "$http_content_type",
    "clientTxnId": "$upstream_http_x_client_transaction_id", "txnId": "$upstream_http_x_transaction_id",
 "msg": "$server_protocol $request_method $uri processed"
    }'
```

Once the Ingress config map is created, this map should be updated in the Ingress by using this command:

> **NOTE**
> The following command assumes that the config file is named as "**ingress-nginx-controller.yaml**". Once the config map update is complete, the Ingress pod is reloaded with the new configurations.

```
kubectl apply -f ingress-nginx-controller.yaml -n ingress
```

The following example shows a sample output with the modified logging format:

```
{"level":"info","type":"log","timestamp": "2021-09-22T06:21:05+00:00", "remoteAddr": "-", "userIp":
 "192.0.2.1", "requestId": "06d79d53dc2cc220f777ba0ab4577d2a", "remoteUser": "55f71d8a-88d1-4a69-
b70e-c14ec9a8ee9e", "bytesSent": 1737, "requestDuration": 0.057, "status": 200, "vhost": "nbruce-63-
ssp.example.com", "protocol": "HTTP/1.1", "path": "/oauth2/v1/token","api": "/oauth2/v1/token", "reqLength":
 529, "method": "POST", "httpReferrer": "-",  "userAgent": "PostmanRuntime/7.26.8", "contentType":
 "application/x-www-form-urlencoded", "clientTxnId": "test", "txnId": "3de955d3-003d-4b73-b422-51167d5e235c",
 "msg": "HTTP/1.1 POST /oauth2/v1/token processed" }
{"level":"info","type":"log","timestamp": "2021-09-22T06:21:13+00:00", "remoteAddr": "-", "userIp":
 "192.0.2.1", "requestId": "8418e31de8f554d1c2ac7a4a7399a6ca", "remoteUser": "-", "bytesSent": 618,
 "requestDuration": 0.120, "status": 200, "vhost": "nbruce-2986-kibana.example.com", "protocol": "HTTP/2.0",
 "path": "/internal/search/es","api": "/internal/search/es", "reqLength": 1877, "method": "POST",
 "httpReferrer": "https://nbruce-2986-kibana.example.com/app/discover",  "userAgent": "Mozilla/5.0 (Macintosh;
 Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.82 Safari/537.36",
 "contentType": "application/json", "clientTxnId": "-", "txnId": "-", "msg": "HTTP/2.0 POST /internal/search/
es processed" }
{"level":"info","type":"log","timestamp": "2021-09-22T06:21:14+00:00", "remoteAddr": "-", "userIp":
 "192.0.2.1", "requestId": "e0d0eb8531c5e517495a0276cb36a88a", "remoteUser": "-", "bytesSent": 2659,
 "requestDuration": 0.020, "status": 200, "vhost": "nbruce-2986-kibana.example.com", "protocol":
 "HTTP/2.0", "path": "/internal/search/es","api": "/internal/search/es", "reqLength": 124, "method": "POST",
 "httpReferrer": "https://nbruce-2986-kibana.example.com/app/discover",  "userAgent": "Mozilla/5.0 (Macintosh;
 Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.82 Safari/537.36",
 "contentType": "application/json", "clientTxnId": "-", "txnId": "-", "msg": "HTTP/2.0 POST /internal/search/
es processed" }
{"level":"info","type":"log","timestamp": "2021-09-22T06:21:22+00:00", "remoteAddr": "-", "userIp":
 "192.0.2.1", "requestId": "a7215290b7453e3ad57709a54bfd0918", "remoteUser": "-", "bytesSent": 2842,
 "requestDuration": 0.025, "status": 200, "vhost": "nbruce-2986-kibana.example.com", "protocol":
 "HTTP/2.0", "path": "/internal/search/es","api": "/internal/search/es", "reqLength": 979, "method": "POST",
 "httpReferrer": "https://nbruce-2986-kibana.example.com/app/discover",  "userAgent": "Mozilla/5.0 (Macintosh;
 Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.82 Safari/537.36",
 "contentType": "application/json", "clientTxnId": "-", "txnId": "-", "msg": "HTTP/2.0 POST /internal/search/
es processed" }
{"level":"info","type":"log","timestamp": "2021-09-22T06:38:52+00:00", "remoteAddr": "-", "userIp":
 "192.0.2.1", "requestId": "1e1d46586df3cdde3cc46d00a035c7a2", "remoteUser": "-", "bytesSent": 296,
 "requestDuration": 0.355, "status": 200, "vhost": "nbruce-2986-kibana.example.com", "protocol": "HTTP/2.0",
 "path": "/api/ui_metric/report","api": "/api/ui_metric/report", "reqLength": 1060, "method": "POST",
 "httpReferrer": "https://nbruce-2986-kibana.example.com/app/discover",  "userAgent": "Mozilla/5.0 (Macintosh;
 Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.82 Safari/537.36",
 "contentType": "application/json", "clientTxnId": "-", "txnId": "-", "msg": "HTTP/2.0 POST /api/ui_metric/
report processed" }
```


Accessing the Ingress Logs

To access the Ingress logs in the third-party dashboard, for example, the Kibana dashboard, perform the following steps.

1. Include the following parameter in your IDSP "ssp-infra" helm chart install/upgrade command:

   ```
   --set sspfbService.ingressLogsIncluded=true
   ```

   For reference to deploy and upgrade commands, see here, Infra Helm deployment command and Infra Helm upgrade command (see step 9-Infra Upgrade).

   NOTE
   If the names of your ingress pods do not use the pattern "ingress-nginx-controller-*", use the following **customConfig > additionalInputs** section in your values yaml file, and replace "ingress-nginx-controller-*" with the pattern that matches your ingress-nginx-controller name and set the ingressLogsIncluded to false, as shown in this example:

```
sspfbService:
    ingressLogsIncluded=false
fluent-bit:
  customConfig:
    additionalInputs:
    - logFileFilter: ingress-nginx-controller-*.log
      tagWith: ssp_msg
```

See Additional Parameters to add to the IDSP Infra Chart Deployment section.

Operational Metrics Data

The operational metrics data helps in measuring efficiency, productivity, and quality of the processes running within the Symantec Identity Security Platform solution. External monitoring tools like Prometheus Operator can be integrated with the solution to monitor the health and get key performance indicators of the deployment.

> **NOTE**
> If you are using out-of-the-box enclave services provided by IDSP, you need not perform the steps mentioned in this topic. The following information is only required if you intend to connect your existing Prometheus solution with IDSP.

IDSP supports exposing metrics information to an external collector. Each of IDSP's micro services exposes a metrics endpoint in a Prometheus format. The information about these endpoints is provided using Prometheus Operator objects called ***ServiceMonitor***.

When integrating the IDSP solution with Prometheus, you should examine the configuration of the monitoring tool to see how this monitoring service is configured to identify **ServiceMonitor** objects that it must serve. This can be done by using a specific label on the serviceMonitor object or on a namespace.

To see the labels that are used by Prometheus for the serviceMonitor object, use the following command:

```
kubectl get prometheus -o yaml
```

The following example shows the labels that are used as a selector for the serviceMonitor object:

```
serviceMonitorSelector:
 matchLabels:
   team: frontend
```

To ensure that such a similar label exists on the IDSP created **serviceMonitor** object, add the following command to the IDSP (ssp) helm chart deployment command:

```
--set-string ssp.global.monitoring.prometheusOperator.serviceMonitor.additionalLabels.<label-name>="<label-value>"
```

- where, the <label-name> and <label-value> should be replaced with appropriate label and value.

After replacing the <label-name> and <label-value>, the configuration would look similar to this example:

```
--set-string ssp.global.monitoring.prometheusOperator.serviceMonitor.additionalLabels.team=frontend
```

Reference Links:

- Custom Metrics Supported by IDSP

Custom Metrics Exposed by Symantec Identity Security Platform

Symantec Identity Security Platform exposes custom metrics that provide insight into the health of the business services.

The following table summarizes the custom metrics (counters/timers) exposed by IDSP services.

Table 33: Custom Metrics exposed by IDSP

Metrics	Type	Description	Events	Tags
`ssp_auth_total`	Counter	Metrics published by the authentication orchestrator service (auth-mgr) for various authentication events	• auth-mgr.auth.initiate • auth-mgr.auth.success • auth-mgr.auth.failure	• eventId • tenant • appName
`ssp_azserver_total`	Counter	Metrics published by authorization server (azserver) for OIDC/OAuth2 events	• azserver.token.success • azserver.token.failure • azserver.token.revoke.success	• eventId • tenant • **grantType**
`ssp_factor_total`	Counter	Metrics published by factors service (factor) for various authentication events	• factor.me.register.initiate.success • actor.me.register.initiate.failure • factor.me.register.complete.success • factor.me.register.complete.failure • factor.me.auth.initiate.success • factor.me.auth.initiate.failure • factor.me.auth.complete.success • factor.me.auth.complete.failure • factor.me.auth.lock.success • factor.me.auth.lock.failure	• eventId • tenant • **factorCount** • **factorType**
`ssp_risk_total`	Counter	Metrics published by risk service (factor) for risk evaluation operations	• risk.eval.success • risk.eval.failure	• eventId • tenant • **riskScore** • **riskReason**
`ssp_idstore`	Timer	Timer for idstore events generated by IDSP	• operation= USER_GET • operation=PASSWORD_AUTH	• tenant • idstore • type • operation • status

Metrics	Type	Description	Events	Tags
`ssp_identity`	Counter	Metrics published by Identity Service and SCIM	**User metrics** • identity.scim.user.create.success • identity.scim.user.create.failure • identity.scim.user.update.success • identity.scim.user.update.failure • identity.scim.user.delete.success • identity.scim.user.delete.failure **Group metrics** • identity.scim.group.create.success • identity.scim.group.create.failure • identity.scim.group.update.success • identity.scim.group.update.failure • identity.scim.group.delete.success • identity.scim.group.delete.failure • identity.scim.group.members.update.success -extended attrs contain users being added or removed • identity.scim.user.groups.members.update.failure -extended attrs contain users being added or removed **Password and unlock metrics** • identity.scim.user.password.update.success • identity.scim.user.password.update.failure • identity.me.password.reset.initiate.success • identity.me.password.reset.initiate.failure • identity.password.reset.initiate.success • identity.password.reset.initiate.failure • identity.me.unlock.success • identity.me.unlock.failure	• eventId • tenant • type • idstore="internal"

`ssp_message_delivery_seconds_count`
`ssp_message_delivery_seconds_max`
`ssp_message_delivery_seconds_sum`
	Timer	Metrics published by message delivery service	• Message delivery events where channel is - EMAIL/SMS/PUSH/IVR	• tenant
				• deliveryChannel
				• providerName
				• **status**

Metrics	Type	Description	Events	Tags
ssp_admin_total	Counter	Metrics published by admin service for various administration events	• admin.me.cred.update.success • admin.me.cred.delete.success • admin.me.cred.update.failure • admin.me.cred.delete.failure • • admin.user.cred.update.success • admin.user.cred.delete.success • admin.user.cred.update.failure • admin.user.cred.delete.failure • • admin.user.device.create.success • admin.user.device.update.success • admin.user.device.delete.success • admin.user.device.create.failure • admin.user.device.update.failure • admin.user.device.delete.failure • • admin.me.device.create.success • admin.me.device.update.success • admin.me.device.delete.success • admin.me.device.create.failure • admin.me.device.update.failure • admin.me.device.delete.failure • • admin.cert.create.success • admin.cert.update.success • admin.cert.delete.success • admin.cert.create.failure • admin.cert.update.failure • admin.cert.delete.failure • • admin.app.create.success • admin.app.update.success • admin.app.delete.success • admin.app.create.failure • admin.app.update.failure • admin.app.delete.failure • • admin.app.activate.success • admin.app.activate.failure • admin.app.deactivate.success • admin.app.deactivate.failure • • admin.authpolicy.create.success • admin.authpolicy.update.success • admin.authpolicy.delete.success • admin.authpolicy.create.failure • admin.authpolicy.update.failure • admin.authpolicy.delete.failure • • admin.authzpolicy.create.success • admin.authzpolicy.update.success	• tenant • eventId
			• admin.authzpolicy.delete.success • admin.authzpolicy.create.failure • admin.authzpolicy.update.failure • admin.authzpolicy.delete.failure •	

Metrics	Type	Description	Events	Tags
ssp_identity_stats	Gauge	Static metrics published by the scheduler service	N/A	• type • tenant • provider="internal"

The following table displays the custom metrics and associated samples.

Table 34: Custom Metrics and Samples

Metrics	Sample
ssp_auth_total	ssp_auth_total{ container="ssp-auth-mgr", endpoint="https", eventId="auth-mgr.auth.initiate", instance="10.40.0.195:8086", job="ssp-1737-ssp-auth-mgr", namespace="ssp1", pod="ssp-1737-ssp-auth-mgr-9f684cfff-kq65p", service="ssp-1737-ssp-auth-mgr", tenant="default", appName="defaultClient"} codeph>
ssp_azserver_total	ssp_azserver_total{container="ssp-azserver", endpoint="https", eventId="azserver.token.failure", grantType="authorization_code", instance="10.40.1.54:8085", job="ssp2799-ssp-azserver", namespace="ssp1", pod="ssp2799-ssp-azserver-9bf94488c-7xmvt", service="ssp2799-ssp-azserver", tenant="tenant536114"}
ssp_factor_total	ssp_factor_total{container="ssp-factor", endpoint="https", eventId="factor.me.auth.complete.success", factorCount="1", factorType="password", instance="10.40.4.143:8080", job="ssp2799-ssp-factor", namespace="ssp1", pod="ssp2799-ssp-factor-84cdbdc77f-2cqvk", service="ssp2799-ssp-factor", tenant="tenant20413"}
ssp_risk_total	ssp_risk_total{container="ssp-iarisk", endpoint="https", eventId="risk.eval.success", instance="10.40.1.16:8088", job="ssp-1737-ssp-iarisk", namespace="ssp1", pod="ssp-1737-ssp-iarisk-64cdd95544-tpssm", riskReason="Allowed IP, Device recognition, Work day time", riskScore="100", service="ssp-1737-ssp-iarisk", tenant="tenant371010"}
ssp_idstore	ssp_idstore_seconds_count{container="ssp-auth-mgr", endpoint="https", idstore="example-ad", instance="10.112.12.39:8086", job="ssp89-ssp-auth-mgr", namespace="ssp1", operation="USER_GET", pod="ssp89-ssp-auth-mgr-6675747fc5-s9kzb", service="ssp89-ssp-auth-mgr", status="success", tenant="default", type="ldap"} ssp_idstore_seconds_count{container="ssp-factor", endpoint="https", idstore="example-idstore", instance="10.112.12.35:8080", job="ssp89-ssp-factor", namespace="ssp1", operation="PASSWORD_AUTH", pod="ssp89-ssp-factor-d78c48f-nmfp4", service="ssp89-ssp-factor", status="success", tenant="default", type="ldap"}

Metrics	Sample
ssp_message_delivery_seconds_count ssp_message_delivery_seconds_max ssp_message_delivery_seconds_sum	ssp_message_delivery_seconds_count{channel="EMAIL", container="ssp-factor", endpoint="https", exception="MdsException", instance="10.40.0.146:8080", job="ssp-1698-ssp-factor", namespace="ssp1", pod="ssp-1698-ssp-factor-69b79f6874-x8zdv", provider="smtp", service="ssp-1698-ssp-factor", status="failure", tenant="tenant198410"}
ssp_admin_total	ssp_admin_total{container="ssp-admin", endpoint="https", eventId="admin.app.success", instance="10.40.0.214:8083", job="ssp2799-ssp-admin", namespace="ssp1", pod="ssp2799-ssp-admin-588c55bbf5-mcspd", service="ssp2799-ssp-admin", tenant="tenant126419"}
ssp_identity_stats	ssp_identity_stats{container="ssp-scheduler", endpoint="https", instance="10.44.28.11:8083", job="ssp89-ssp-scheduler", namespace="ssp1", pod="ssp89-ssp-scheduler-68ff89849b-wtpk9", provider="internal", service="ssp89-ssp-scheduler", tenant="default", type="groups"}

Symantec Identity Security Platform Monitoring Information in Grafana Dashboard

The Symantec Identity Security Platform panels that are represented in Grafana dashboard provides insights into the performance and health of IDSP and its integrated components.

This document provides links to all the IDSP panels, providing a centralized view of key metrics for Authentication Statistics, Integrations, Resource Statistics, Policy Usage Statistics, and overall Services Health. It also provides the operational status of your environment with dedicated panels for Insights, Cache Health, Database Health, Policy Engine (OPA) Health, and KPI, alongside Enclave Services Health.

Top Bar Controls

`Cluster / Namespace / Aggregator Filters` : These controls allow filtering of data by cluster, namespace, and time interval aggregation (1m)

`Time Range Selector` : This controls the time window for the data displayed. Set the time selector to include metrics information from the various options available. For example, select "Last 24 hours".

The following IDSP panels are displayed in Grafana Dashboard:

- Summary
- Authentication Statistics
- Integrations
- IDSP Resource Statistics
- IDSP Policy Usage Statistics
- IDSP Services Health
- IDSP Insights
- IDSP Cache Health
- IDSP Database Health
- IDSP Policy Engine (OPA) Health
- IDSP KPI
- Enclave Services Health
- Kubernetes Cluster

Summary

Symantec Identity Security Platform Summary panels - Overview and Metrics

The Summary panels serve as high-level KPIs for monitoring the performance and behavior of authentication events.

- Total Requests
- Authentication

Total Requests

This panel summarizes the overall system-level performance metrics.

Table 35: Total Requests Panel

Metric	Description
Success Rate	API success percentage
Latency	The average response time (in milliseconds) for each API request
Throughput	The average API requests per minute
Count	Total number of API requests

Authentication

This panel summarizes the metrics of the authentication flows.

Table 36: Authentication Panel

Metric	Description
Initiations	Number of initiated authentication flows
Allowed	Percentage of successful authentications
In Progress	Percentage of authentications currently in progress
Abandoned	Percentage of authentications abandoned
Denied	Percentage of authentications denied

Metric	Description
Sign-in Duration	Average sign-in time

Authentication Statistics

The Authentication Statistics panels provide an overview of authentication behavior, analyzing credential usage (primary and multi-factor), risk rule triggers, and authentication outcomes.

The Authentication Statistics dashboard contains panels to analyze authentication behavior of Symantec Identity Security Platform.

The following panels are included:

- Credentials Usage - As Primary
- Credentials Usage - As Multi Factor
- Initiations
- Risk Rules
- Statistics
- Sign-In KPI

Credentials Usage - As Primary

This panel displays the metrics of credential types that are used as the primary authentication factor, measured in value and percentage.

- `Value` - The count of how many times a specific credential type was used during the selected time range. For example, the Password or SMS OTP was used for "x" number of times.
- `Percentage` - The percentage of each credential type's usage compared to the total count of factors for that panel.

Credentials Usage - As Multi Factor

This panel displays the metrics of credential types that are used in multi factor authentication, measured in value and percentage.

- `Value` - The count of how many times a specific credential type was used during the selected time range. For example, the Email and SMS OTP has been used for "x" number of times.
- `Percentage` - The percentage of each credential type's usage compared to the total count of factors for that panel.

Initiations

This panel displays metrics on distribution of authentication initiations across different applications.

- `Value` - The count of how many times a specific application has been initiated during the selected time range.

Risk Rules

This panel displays metrics on distribution of the triggered risk rules. It indicates the rules that are triggered the most and the least.

- `Value` - The count of how many times a specific risk rule type was triggered during the selected time range.
- `Percentage` - The percentage of each risk rule type's that were triggered compared to the total count of risk rules triggered for that panel.

Statistics

This panel displays the metrics to help correlate authentication outcomes with specific applications.

Table 37: Statistics Panel

Attributes	Description
Application Name	Lists each application involved in authentication.
Initiations	Number of initiated authentication attempts for the application.
Allowed	Number of successful authentications
Abandoned	Number of authentications that were abandoned before completion.
In Progress	Number of authentications currently ongoing.
Denied	Number of failed or denied authentications.
Sign-in Duration	Time taken (in seconds or ms) to complete the sign-in process per application.

Sign-In KPI (Key Performance Indicator)

This panel provides key performance indicators (KPIs) for the signin process. It helps assess both the duration and throughput of user signins over time.

The panel includes:

- A **time-series line graph** visualizing the KPIs.
- A **table** summarizing metrics on sign-in duration and sign-in throughput.

Graph Type: Time-series line graph

X-Axis Unit: Represents time (chronological series)

Y-Axis Unit: Represents the number of sign-in requests per minute

Table 38: Sign-In KPI Panel

Metric	Description
Sign-in Duration	Average time it takes for a user to complete the sign-in process
Sign-in Throughput	Rate at which sign-in requests are processed over time

Integrations

Symantec Identity Security Platform Integrations - provides insight into the performance of core backend integration services.

The "Integrations" panels provides real-time monitoring of the core backend integration services that support authentication flows in IDSP.

This topic includes the following panels:

- Message Delivery, Risk, Decryption Providers
- Identity Stores

Message Delivery, Risk, Decryption Providers

This panel provides insight into the performance of various backend integration services.
It captures the following key metrics for each service:

- **Success Rate** - The percentage of successful responses from the service.
- **Request Count** - The total number of requests made to the service.
- **Latency** - The average response time (in milliseconds) for each service.

Table 39: Service Descriptions

Service	Description
SMS	SMS provider used to deliver OTPs
Email	Email provider used to deliver OTPs
Push & TOTP (VIP)	Push notification or time-based OTP delivery via VIP
SPI (SMS, Email, IVR, Risk)	Custom service provider interface supporting multiple channels
Risk Engine	Evaluates risk based on behavior, device, etc
Decryption	Service responsible for data decryption
Geolocation	Service to fetch the geographic location based on IP address
Policy Engine (OPA)	Policy Engine of IDSP

Identity Stores

This panel displays metrics of requests made to the configured identity providers during authentication (like LDAP, SCIM, AD, or INTERNAL).
It captures the following key metrics for the identity stores:

- **Request Count** - The total number of requests made to the service.
- **Latency** - The average response time (in milliseconds) for each service.

Resource Statistics

Symantec Identity Security Platform Resource Statistics - these panels display metrics related to users and groups, password activities, applications , policies etc.

The Resource Statistics dashboard provides a holistic view of resources and policies that are configured and used in the IDSP (SSP) environment, useful for admins to monitor usage patterns.

This topic includes the following panels:

- Users and Groups
- Password Activities
- Application, Certificates and Identity Providers
- Policies

Users and Groups

These are separate panels, each displaying metrics of total and newly added users and groups in IDSP.

Table 40: User and Group Panels

Panel	Description
User Count	Total number of users currently registered in the system.
New Users	Number of users newly created in the system over the selected time range.
Group Count	Total number of user groups.
New Groups	Number of new groups created in the system over the selected time range.

Password Activities

These are separate panels, each displaying metrics related to password-related actions such as resets and updates by users and admins in IDSP.

Table 41: Password Activity Panels

Panel	Description
Password Reset (By Admin)	Number of password resets performed by administrators.
Password Reset (By User/Self)	Password resets initiated by users themselves (self-service).
Password Reset Completed (By User)	Successful completion of user-initiated password resets.
Password Update (By User)	Instances where users updated their password (not necessarily due to reset).

Application, Certificates and Identity Providers

These are separate panels, each displaying the metrics related to IDSP resource usage activities, including the number of registered applications, federated identity providers, and issued certificates.

Table 42: Application and Integration Stats Panel

Panel	Description
Applications	Number of registered applications in IDSP.
Identity Providers	Number of configured identity providers (for example - LDAP, SAML, OIDC).
Certificates	Number of active certificates

Policies

These are separate panels, each displaying metrics related to the number of configured authentication, authorization, and IDP discovery policies within IDSP.

Table 43: Policy Panel

Panel	Description
Authentication Policies	Total number of policies governing user authentication rules.
Authorization Policies	Total number of policies defining access control/permissions.
IDP Discovery Policies	Policies that determine how identity providers are selected during login.

Policy Usage Statistics

Know about Symantec Identity Security Platform Policy Usage Statistics.

Policy Usage Statistics panels monitors how often authentication, IDP discovery policies, and authorization policies are matched, helping assess policy usage trends.

This topic includes the following panels:

- Matched Authentication Policies
- Matched Authentication Policies Statistics
- Matched IDP Discovery Policies
- Matched IDP Discovery Policies Statistics
- Matched Authorization Policies
- Matched Authorization Policies Statistics

Matched Authentication Policies

This panel displays metrics of how frequently different authentication policies were matched during authentication flows.

Table 44: Matched Authentication Policies Panel

Metric	Description
Policy Names	Displays the matched authentication policy names.
Value	The count of how many times a specific authentication policy was matched during the selected time range.
Percentage	The percentage share of each authentication policy match found compared to the total count of policies for that panel.

Matched Authentication Policies Statistics

This panel displays metrics on how often each policy and the policy rule is matched during authentication flows.

Table 45: Matched Authentication Policies Statistics Panel

Metric	Description
Policy Name	Displays the matched authentication policy names.
Rule Name	Displays the matched authentication policy rule name.
Count	Number of times the identified policies were matched during authentication attempt.

Matched IDP Discovery Policies

This panel displays metrics of how frequently different IDP Discovery policies were matched during authentication flows.

Table 46: Matched IDP Discovery Policies Panel

Metric	Description
Policy Names	Displays the matched IDP Discovery policy names.
Value	The count of how many times a specific IDP Discovery policy was matched during the selected time range. For example, the "Corporate IDP Policy" was matched "x" number of times.
Percentage	The percentage share of each IDP Discovery policy matches found compared to the total count of policies for that panel.

Matched IDP Discovery Policies Statistics

This panel displays a metric to help correlate IDP Discovery policies with how often each policy is matched during authentication flows.

Table 47: Matched IDP Discovery Policies Statistics Panel

Metric	Description
Policy Name	Displays the matched IDP Discovery policy names.
Rule Name	Displays the matched IDP Discovery policy rule.
Count	Number of times the identified IDP Discovery policies were matched during authentication attempt.

Matched Authorization Policies

This panel displays metrics on matched authorization policies along with their usage frequency and descriptions to help assess policy effectiveness and access patterns.

Table 48: Matched Authorization Policies Panel

Metric	Description
Policy Names	Displays the matched authorization policy names.
Value	The count of how many times a specific authorization policy was matched during the selected time range. For example, the "TenantAdminPolicy" was matched "x" number of times.
Percentage	The percentage share of each authorization policy matches found compared to the total count of policies for that panel.

Matched Authorization Policies Statistics

This panel displays metrics that helps to correlate matched authorization policies with their access pattern.

Table 49: Matched Authorization Policies Statistics Panel

Metric	Description
Policy Name	Displays the matched authorization policy names.
Rule Name	Displays the matched authorization policy rule.

Metric	Description
Count	Number of times the identified authorization policy was matched.

Services Health

Know about Symantec Identity Security Platform Services Health

The Service Health panels provide real-time insights into service and pod level resource usage and database connection pool metrics, helping monitor system performance and identify bottlenecks.

This topic includes the following panels:

- Service Status
- Service Metrics
- Service Throughput
- Pod CPU, Pod Memory, JVM Threads, DB Pool Usage

Service Status

This panel displays the metrics on the status of all the IDSP services running in the Kubernetes cluster.

Table 50: Service Status Panel

Metric	Description
Service	Displays the IDSP services running in the Kubernetes cluster.
Status	Displays the status of the services. There are many possible service status - Ready, Initializing, Terminating etc.
Replicas	Displays the number of pods or instances of the IDSP services that are running at any given time.

Service Metrics

This panel represent the metrics related to key performance indicators, such as throughput, latency, and resource usage for monitoring the health and efficiency of each IDSP service.

Table 51: Service Metrics Panel

Metric	Description
Service	Displays the IDSP services running in the Kubernetes cluster.
Request Count	Displays the total number of requests received by each of these IDSP services.
Throughput (rpm)	Indicates the rate at which API requests are served by the IDSP services, measured in request per minute (rpm).
Latency	Indicates the time taken by IDSP services to respond to requests. This is a critical metric for understanding service performance and identifying potential bottlenecks or issues.
Error Count	Indicates the number of errors or failures associated with that specific IDSP service over a given time period.

Service Throughput

This panel displays the metrics on the number of requests processed by each IDSP service, indicating overall service load and performance. This information is represented through a graph.

Graph Type: Time-series line graph with a table of service throughputs.

Y-Axis Unit: Shows throughput value

Table 52: Service Throughput Panel

Metric	Description
Max	The maximum throughput value indicates the highest API requests handled by each of the IDSP service within a specified time frame. This can be useful for identifying peak loads and capacity planning.
Average	The average throughput shows the mean throughput over a specific period, providing an overall understanding of the service's performance. It can be useful for identifying trends and comparing the service's performance over time.
Current	The current throughput reflects the real-time throughput of the IDSP service at a specific moment. It provides a snapshot of the service's current activity and can be used for monitoring and alerting based on real-time performance.

Pod CPU, Pod Memory, JVM Threads, DB Pool Usage (1m)

This panel displays metrics related to DB connection pooling, availability, performance, and resource limits.

Table 53: Pod CPU, Pod Memory, JVM Threads, DB Pool Usage Panel

Metric	Description
Pod Name	The name of the Kubernetes pod being monitored.
Node IP	The IP address of the node hosting the pod.
Request Count	Total number of requests received by the pod in the observed interval.
Throughput (rps)	The number of requests processed per second by the pod (rps = requests per second).
Latency (ms)	Average time taken to respond to requests, measured in milliseconds.
CPU Limit	The maximum CPU resource allocated to the pod.
Memory Limit	The percentage of CPU used from the allocated limit.
JVM Threads (Live)	The number of active/live Java Virtual Machine (JVM) threads running in the pod.
DB Timeouts	The count of database timeout events experienced by the pod.
DB Conn Time	The average time (in milliseconds) taken to establish a database connection. Lower values indicate better performance; high values may suggest connection delays or DB contention.
DB Pool Active	The number of currently active (in-use) database connections in the connection pool. Reflects live demand on the pool.
DB Pool Idle	The number of idle (available and ready) connections in the pool. If this number is constantly low, it may indicate the pool is under-provisioned.
DB Pool Min	The minimum number of connections maintained in the connection pool, even during low traffic. This ensures a baseline readiness.

Metric	Description
DB Pool Max	The maximum number of connections allowed in the pool. Once this limit is reached, new requests must wait for a connection to be freed. It helps in controlling DB resource usage.

Insights

Know about Symantec Identity Security Platform Insights

The Insight panels collectively provide visibility into authentication risks, geographical threat patterns, and API performance bottlenecks.

This topic includes the following panels:

- Triggered Risk Rule Vs Authentications
- Location Vs Authentication Statistics
- API with High Response Time (>100ms)

Triggered Risk Rule Vs Authentications

This panel displays metrics on different risk rules that are triggered during authentication attempts, and how those triggered risk rules affect the outcome of each authentication.

Table 54: Triggered Risk Rule Vs Authentications Panel

Metric	Description
Risk Rule	Displays the risk rule that was triggered during the authentication attempt. For example, rules could detect suspicious IP addresses, risky countries, behavior anomalies, etc.
Auth Allowed	The number of authentication attempts where the risk rule was triggered but the authentication was allowed to proceed. This means the system flagged a risk, but the user was still permitted access.
Auth Denied	The number of authentication attempts where the risk rule was triggered and the authentication was denied. This means the risk rule caused a failure in the authentication.
Auth Abandoned	The number (or rate) of authentication attempts where the risk rule was triggered but the authentication process was abandoned by the user.

Location Vs Authentication Statistics

This panel displays metrics on how authentication attempts are distributed across different geographical locations, and how those attempts succeed, fail, or get abandoned based on risky countries. It helps identify patterns of suspicious behavior linked to specific regions.

Table 55: Location Vs Authentication Panel

Metric	Description
Country	Displays the country from which the authentication attempt was made during the authentication attempt.
Auth Allowed	Number of authentications from the country that were successful.
Auth Denied	Number of authentications from the country that were rejected.
Auth Abandoned	Number of authentications that were started but not completed.

API with High Response Time (>100ms)

This panel displays metrics on the APIs that consistently take longer than 100 milliseconds to respond. It helps identify performance bottlenecks and latency issues in IDSP services.

Table 56: API with High Response Time Panel

Metric	Description
Name	Displays the list of IDSP APIs that take longer than 100 ms to respond.
Mean	The average response time of the IDSP API over the observed time range.
Last	The most recent response time recorded for the IDSP API.
Max	The maximum response time of the IDSP API observed during the time range.
Min	The minimum response time of the IDSP API observed during the time range.

Cache Health

Know about the health of Symantec Identity Security Platform Cache

The Cache Health panels gives insights into the health of Cache service (Hazelcast) in IDSP. It provides details of functional metrics (map statistics) and operational metrics (memory, heap, CPU usage) of Hazelcast cluster members.

This topic includes the following panels:

- Connection Latency
- Availability
- Status
- Member Count
- Client Count
- Member Heap Usage
- Member Memory Usage
- Member CPU Usage
- Map Statistics

Connection Latency

This panel displays metrics on latency between Hazelcast members and clients.

Availability

This panel displays the percentage availability of cache service.

Status

This panel displays the status of the Hazelcast members.

Member Count

This panel displays the number of active Hazelcast members in the cluster.

Client Count

This panel displays the number of clients connected to Hazelcast cluster.

Member Heap Usage

This panel displays the metrics on the JVM heap memory in use by each Hazelcast member.

Table 57: Member Heap Usage Panel

Metric	Description
Name	The name of the Hazelcast member (pod) in the Kubernetes cluster.
Mean	The average heap memory used by that member over the selected time range, measured in MiB (Mebibytes). This shows general memory consumption trends.
Max	The maximum heap memory consumed by that member during the selected time period. It represents the peak memory usage, helping identify memory spikes or potential overloads.

Member Memory Usage

This panel displays the metrics on the total memory used by each Hazelcast member.

Table 58: Member Memory Usage Panel

Metric	Description
Name	The name of the Hazelcast member (pod) in the Kubernetes cluster.
Mean	The average total memory (not just heap) used by the member over the selected time range, measured in MiB (Mebibytes).
Max	The maximum total memory (not just heap) consumed by that member during the selected time period.

Member CPU Usage

This panel displays metrics on the CPU usage percentage for each Hazelcast member (pod) in the Kubernetes cluster over the selected time range.

Table 59: Member CPU Usage Panel

Metric	Description
Name	The name of the Hazelcast member (pod) in the Kubernetes cluster.
Mean	The average CPU usage of the member over the selected time range, measured in percentage.
Max	The maximum CPU usage of the member over the selected time range, measured in percentage.

Map Statistics

This panel provides metrics on key operational insights about Hazelcast distributed maps within a Kubernetes cluster. Each row corresponds to a specific map held by a Hazelcast member (pod), and shows various stats related to memory, and read/write behavior.

Table 60: Member CPU Usage Panel

Metric	Description
Map Name	The name of the distributed map.
Member	The name of the Hazelcast member (pod) that holds the data for the map.
Count	The number of key-value pairs (entries) in the map on that member.
Backup Count	The number of entries held as backups on this member (in case the primary owner fails).
Memory	Approximate memory used by the map on that member.
Cache Reads	The number of times the map was read (GET operations).
Cache Updates	The number of times entries in the map were updated (PUT operations).

Database Health

Know about the health of Symantec Identity Security Platform Databases

The Database Health panels provides a detailed view of database connection activity, usage duration, timeout events, and replication latency across all the IDSP services to ensure efficient database health.

This topic includes the following panels:

- Active Connections
- Connections Usage Time
- Connection Timeouts
- Replication Latency (Populated when system setting PersistentStoreForTransactionalData=DB)

Active Connections

This panel displays metrics showing all the IDSP active database connections from various IDSP services in a selected time range.

The panel includes:

- A **time-series line graph** visualizing the active connections.
- A **table** summarizing active db connections for each IDSP services.

Graph Type: Time-series line graph

X-Axis Unit: Represents time (chronological series)

Y-Axis Unit: Represents number of active connections.

Table 61: Active Connections Panel

Metric	Description
Name	Lists the names of the IDSP services that are being tracked for active connections.
Last	The most recent active connection recorded for each IDSP services.
Mean	The average active connections that were recorded for each IDSP services during the selected time range.
Max	The highest observed active connection recorded for each IDSP services during the selected time range.

Connections Usage Time

This panel displays metrics showing how long the IDSP database connections were used during the selected time range. It helps identify potential bottlenecks or performance issues related to connection handling.

The panel includes:

- A **time-series line graph** visualizing the connection duration trends.
- A **table** summarizing usage statistics for each IDSP services

Graph Type: Time-series line graph

X-Axis Unit: Represents time (chronological series)

Y-Axis Unit: Represents connection usage time, in milliseconds (ms).

Table 62: Connections Usage Time Panel

Metric	Description
Name	Lists the names of the IDSP services.
Mean	The average duration for which the connections were used by each IDSP services during the selected time range.
Last	The most recent connection usage time recorded for each IDSP services.
Max	The highest observed connection usage time recorded for each IDSPservices.

Connection Timeouts

This panel displays metrics showing the number of connection timeouts for each Symantec Identity Security Platform service over time. This metric could indicate connectivity issues.

The panel includes:

- A **time-series line graph** visualizing the connection timeout trends.
- A **table** summarizing connection timeout for each service.

Graph Type: Time-series line graph

X-Axis Unit: Represents time (chronological series)

Y-Axis Unit: Represents number of DB connection timeouts.

Table 63: Connection Timeouts Panel

Metric	Description
Name	Lists the IDSP services that are being tracked for connection timeouts.
Mean	The average number of connection timeouts recorded for each IDSP services during the selected time range.
Last	The most recent connection timeouts recorded for each IDSP services.
Max	The highest observed connection timeouts recorded for each IDSP services.

Replication Latency (Populated when system setting PersistentStoreForTransactionalData=DB)

This panel displays metrics showing the delay in IDSP database replication when the system setting, "PersistentStoreForTransactionalData=DB" is configured.

The panel includes:

- A **time-series line graph** visualizing the replication latency trends.
- A **table** measuring average and max replication latency for each IDSP services.

Graph Type: Time-series line graph

X-Axis Unit: Represents time (chronological series)

Y-Axis Unit: Represents DB replication latency time in minutes.

Table 64: Replication Latency Panel

Metric	Description
Name The "Name" categorizes the latency metric being tracked, either "Average Latency" or "Max Latency", helping users understand what kind of latency data is being displayed in the graph.	Indicates the type of latency being measured. • Average Latency - represents the mean time delay between the moment a transaction is written to the primary database and when it is replicated to the secondary database. • Max Latency - represents the maximum delay of database replication observed during the selected time range.
Mean	The average database replication delay recorded during the selected time range. This is displayed in "µs" (microseconds).
Last	The most recent IDSP database replication delay recorded.
Max	The highest observed IDSP database replication delay recorded during the selected time range.

Policy Engine Health

Know about Symantec Identity Security PlatformPolicy Engine.

The Policy Engine Health panels show how well OPA is working, including bundle status, load/download times, and policy evaluation performance.

This topic includes the following panels:

- Policy Engine Health Metric
- OPA Policy Loading Statistics
- OPA Policy Download API Statistics
- OPA Policy Evaluation API Statistics

Policy Engine Health Panels

Table 65: Policy Engine Health Panels

Panels	Description
OPA Policy Bundle State	Shows whether the OPA policy bundle is ready (means available) for use.
OPA Policy Load Status	Indicates if the policy loading was successful.
Policy Bundle Download Latency (ms)	Time taken to download the policy bundle.
Policy Evaluation Request Latency (ms)	Time taken to evaluate policy requests.

Panels	Description
Policy Evaluation Success Rate	Percentage of successful policy evaluations.
Panels available	• Pod - refers to the pod name of the OPA service. • Last bundle request - Timestamp of the latest attempt made to request the policy bundle. • Last successful bundle request - Timestamp marking the last time the policy bundle was successfully requested and received without errors. • Last bundle download - The timestamp when the policy bundle was last fully downloaded to the OPA instance. • Last bundle activation - Indicates when the downloaded policy bundle was last activated and made available for evaluations. • Success Load Count - Displays how many times policy bundles have been successfully loaded into OPA. • Failure Load Count - Displays how many times policy bundles have failed to load into OPA.

OPA Policy Loading Statistics

This panel displays metrics showing the number of successful and failed IDSP policy loads over time.

The panel includes:

- A **time-series line graph**
- A **table** summarizing OPA policy load failure or success events.

Graph Type: Time-series line graph

X-Axis Unit: Represents time (chronological series)

Y-Axis Unit: Represents number of policy load events, success or failure.

Table 66: OPA Policy Loading Statistics Panel

Metric	Description
Name The "Name" categorizes the OPA policy loading metric being tracked, either "Policyload Failure" or "Policyload Success", helping users understand what kind of Policyload data is being displayed in the graph.	Indicates the number of policy load failure or success. • Policyload Failure - represents the number of failed policy loads. • Policyload Success - represents the number of successful policy loads.
Last	The most recent value recorded for both API Request Count and API Latency during the selected time range.
Min	The minimum value observed for each metric within the selected time range.
Mean	The average value for each metric within the selected time range.
Max	The highest observed value for each metric within the selected time range.

OPA Policy Download API Statistics

This panel tracks the number of API requests made to download OPA policies and the latency (response time) of those requests over time. It helps monitor the efficiency and responsiveness of the policy download service.

The panel includes:

- A dual-axis-time-series line **graph**
- A **table** summarizing OPA policy downloads.

Graph Type: Dual-axis-time-series line graph

X-Axis Unit: Represents time (chronological series)

Y-Axis Unit: Represents number of policy download events.

Table 67: OPA Policy Download API Statistics Panel

Metric	Description
Name The "Name" categorizes the OPA policy download API Statistics metric being tracked, either "API Request Count" or "API Latency (ms)", helping users understand what kind of API data is being displayed in the graph.	Indicates the API metric being tracked, such as API Request Count or API Latency (ms). • API Request Count - represents the number of times the policy download API was called. • API Latency (ms) - represents the time taken to respond to policy download API calls (in milliseconds).
Last	The most recent value recorded for both API Request Count and API Latency during the selected time range.
Min	The minimum value observed for each metric within the selected time range.
Mean	The average value for each metric within the selected time range.
Max	The highest observed value for each metric within the selected time range.

OPA Policy Evaluation API Statistics

This panel tracks the number of API requests made to evaluate OPA policies and the latency (response time) of those requests over time. It helps monitor the efficiency and responsiveness of the policy evaluation service.

The panel includes:

- A dual-axis-**time-series line graph**
- A **table** summarizing OPA policy evaluation stats.

Graph Type: Dual-axis-time-series line graph

X-Axis Unit: Represents time (chronological series)

Y-Axis Unit: Represents number of OPA policy evaluation events.

Table 68: OPA Policy Evaluation API Statistics Panel

Metric	Description
Name The "Name" categorizes the OPA policy download API Statistics metric with either "API Request Count" or "API Latency (ms)". This helps users understand what kind of API data is being displayed in the graph.	Indicates the API metric being tracked, such as API Request Count or API Latency (ms). • API Request Count - represents the number of times the policy evaluation API was invoked. • API Latency (ms) - represents the time taken to evaluate the policy request (in milliseconds).
Last	The most recent value recorded for both API Request Count and API Latency during the selected time range.
Min	The minimum value observed for each metric within the selected time range.

Metric	Description
Mean	The average value for each metric within the selected time range.
Max	The highest observed value for each metric within the selected time range.

Key Performance Indicator (KPI)

Know about Symantec Identity Security Platform Key Performance Indicators (KPI)

The Key Performance Indicator panels monitor IDSP API performance by tracking throughput (operations per second), latency (response time in ms), and total call count across various endpoints to identify usage trends and potential bottlenecks.

This topic includes the following panels:

- API Throughput
- API Latency
- API Total Count
- API Error Count

API Throughput

This panel monitors the IDSPAPI performance by measuring the throughput (operations per second) of various API endpoints over time.

The panel includes:

- A **time-series line graph**
- A **table** summarizing throughput statistics for each API endpoint

Graph Type: Time-series line graph

X-Axis Unit: Represents time (chronological series)

Y-Axis Unit: Represents Throughput, measured in operations per second (ops/s)

Table 69: API Throughput Panel

Metric	Description
Name	Lists the IDSP API endpoints that are being tracked.
Mean	The average throughput (ops/s) for each IDSP API during the selected time range.
Last	The most recent throughput reading of each IDSP API during the selected time range.
Max	The highest observed throughput per each IDSP API.
Min	The lowest recorded throughput per each IDSP API.

API Latency

This panel monitors and measures the response time (latency) taken by each IDSPAPI call to respond to request.

The panel includes:

- A **time-series line graph**
- A **table** summarizing latency statistics for each API endpoint

Graph Type: Time-series line graph

X-Axis Unit: Represents time (chronological series)

Y-Axis Unit: Represents Latency, measured in milliseconds

Table 70: API Latency Panel

Metric	Description
Name	Lists the IDSP API endpoints that are being tracked.
Mean	The average response time (ops/s) for each IDSP API during the selected time range.
Last	The most recent response reading of each IDSP API during the selected time range.
Max	The highest observed response time taken by each IDSP API.
Min	The lowest recorded response time taken by each IDSP API.

API Total Count

This panel displays the total number of calls made to each IDSP API endpoints over time.

The panel includes:

- A **time-series line graph**
- A **table** summarizing total API count statistics for each API endpoint

Graph Type: Time-series line graph

X-Axis Unit: Represents time (chronological series)

Y-Axis Unit: Represents Total API Call Count in the selected time range

Table 71: API Total Count Panel

Metric	Description
Name	Lists the IDSP API endpoints that are being tracked.
Mean	The average count of calls made to each IDSP API during the selected time range.
Last	The most recent count of calls made to each IDSP API during the selected time range.
Max	The highest observed count of calls made to each IDSP API.
Min	The lowest count of calls made to each IDSP API.

API Error Count

This panel tracks the number of IDSPAPI errors that occurred over a given time period.

The panel includes:

- A **time-series line graph**
- A **table** summarizing total API error count for each API endpoint

Graph Type: Time-series line graph

X-Axis Unit: Represents time (chronological series)

Y-Axis Unit: Represents the count of API errors

Table 72: API Error Count Panel

Metric	Description
Name	Lists the IDSP API endpoints that are being tracked.
Mean	The average count of errors occurred for each IDSP API during the selected time range.
Last	The most recent count of errors occurred for each IDSP API during the selected time range.
Max	The highest observed count of occurred for each IDSP API.
Min	The lowest count of occurred for each IDSP API.

Enclave Services Health

Know about Symantec Identity Security Platform Enclave Services Health

The Enclave Services Health dashboard monitors the health of all the enclave services at pod level that are supported by IDSP. The different panels in this dashboard focuses on storage and resource usage in a Kubernetes environment.

This topic includes the following panels:

- Persistent Volume Claim
- CPU and Memory Usage per pod
- Pods CPU Usage
- Pods Memory Usage

Persistent Volume Claim

This panel displays the current status and usage of Persistent Volume Claims (PVCs).

Table 73: Persistent Volume Claim Panel

Metric	Description
Persistent Volume Claim (PVC)	Lists the PVCs that are created at the pod level
Namespace	Kubernetes namespace where the PVC is defined
StorageClass	Storage class type of the PVC (here it is specified as standard-rwo)
PhysicalVolume	ID of the associated persistent volume
Capacity (GiB)	Total storage capacity allocated to each pod
Used (GiB)	Current used storage
Available (GiB)	Free space available
Status	State of the volume (example, Bound)
Used (%)	Percentage of storage used

CPU and Memory Usage per pod

This panel provides visibility into how much CPU and RAM each pod in a Kubernetes namespace is using relative to its requested and limit values.

Table 74: CPU and Memory Usage Per Pod Panel

Metric	Description
Pod	Lists the IDSP API endpoints that are being tracked.
Namespace	Kubernetes namespace to which the pod belongs
RAM Used	Actual memory currently used by the pod
RAM Requests	Memory (RAM) the pod has requested to reserve.
RAM Limits	The maximum amount of memory the pod is allowed to use.
CPU Used	Actual CPU being consumed by the pod.
CPU Requests	Minimum CPU resources reserved for the pod.
CPU Limits	Maximum CPU the pod is allowed to use.

Pods CPU Usage

This panel monitors CPU usage by different pods in cores and displays the information through a graphical representation.

The panel includes:

- A **time-series line graph**
- A **table** summarizing CPU usage by different pods.

Graph Type: Time-series line graph

X-Axis Unit: Represents time (chronological series)

Y-Axis Unit: Represents CPU cores

Table 75: Pods CPU Usage Panel

Metric	Description
Mean	Average CPU usage during the time window
Last	Latest CPU usage value reported

Pods Memory Usage

This panel displays the memory consumption by each pods.

The panel includes:

- A **time-series line graph**
- A **table** summarizing memory usage by different pods.

Graph Type: Time-series line graph

X-Axis Unit: Represents time (chronological series)

Y-Axis Unit: Represents memory usage in MiB or GiB unit

Table 76: Pods Memory Usage Panel

Metric	Description
Mean	Average memory usage during the time window
Last	Latest memory usage value

Kubernetes Cluster

Know about Symantec Identity Security Platform Kubernetes Cluster panels

The Kubernetes Cluster panels monitor the health and performance of the Kubernetes cluster.

This topic includes the following panels:

- System Time/CPU Cores/Total RAM
- CPU Usage
- Memory Usage

System Time/CPU Cores/Total RAM

These provide a snapshot of the overall resource availability and health:

Table 77: System Time/CPU Cores/Total RAM - Left Panels

Panels	Description
System Uptime	Indicates how long the system has been continuously running (for example - 2.4 weeks).
CPU Cores	Total number of CPU cores available in the cluster (for example - 24).
Total Ram	Total physical memory available across the cluster nodes (for example - 94.05 GiB).
CPU IOWait	Percentage of time CPU is waiting for I/O (input/output) operations to complete (for example - 0.35%).

CPU Usage

This panel monitors the CPU utilization over time for each node which is useful to identify CPU bottlenecks or load spikes on specific nodes.

The panel includes:

- A **time-series line graph**
- A **table** summarizing the CPU usage

Graph Type: Time-series line graph

X-Axis Unit: Represents time, showing when the CPU usage was measured.

Y-Axis Unit: Represents CPU usage percentage, how much of the CPU resources are being used.

Table 78: CPU Usage - Middle Panel

Metric	Description
Name	IP and port of the node exporter on each node (for example - 10.26.2.42:9100).
Mean	Average CPU usage percentage.
Max	Peak CPU usage recorded.

Memory Usage

This panel monitors memory usage trends over time.

The panel includes:

- A **time-series line graph**
- A **table** summarizing the Memory usage

Graph Type: Time-series line graph

X-Axis Unit: Represents time, showing when the Memory usage was measured.

Y-Axis Unit: Represents Memory usage percentage, how much of the memory resources are being used.

Table 79: Memory Usage - Right Panel

Metric	Description
Name	Shows memory consumption for each node in the cluster.
Mean	Average memory usage percentage per node.

Managing Deployment

The deployment topology governs how pods are distributed across your cluster, including regions, zones, nodes, and other user-defined topological domains. This can aid in both high availability and efficient resource utilization.

This topic includes the following sub-topics:

- Meeting Topology Requirements
- Managing Configuration Updates
- Recovering Initial Admin Credentials
- Handling DB Password Rotation
- Scheduler Job
- Populating the Internal Email Domain Column

Meeting Topology Requirements

The content in this page explains the capabilities of the Symantec Identity Security Platform helm chart to manage pods to be align with your specific topology requirements.

In Kubernetes, a Pod represents a set of containers running in a cluster. Pods can be constrained to run on specific node(s), or to prefer to run on specific node(s).

For instructions on updating the configuration of a deployed IDSP helm Chart, see Managing Configuration Changes.

This topic includes the following methods to choose from where Kubernetes schedules specific Pods:

- Node Selector matching against node labels
- Affinity and Anti-Affinity
- Tolerations
- Setting Additional Labels
- Setting Annotations
- Setting Ingress Timeouts
- AWS STS Regional Endpoints
- Helm Offline Mode
- DNS related configuration for IDSP Deployment
- Enabling Availability Metrics
- Seccomp Profile
- HTTP Proxy

Node Selector matching against node labels

The "nodeSelector" parameter is used in IDSP to instruct Kubernetes to schedule the IDSP pods only to a specific set of nodes that have a specific label.

Example:
You can use the following command to add the label "disktype=ssd" to your node:

```
kubectl label nodes <your-node-name> disktype=ssd
```

Where, the <your-node-name> is the name of the node to which you want to add the label, disktype=ssd.

Use the following helm command-line parameter to describe a pod that has a node selector, disktype=ssd.

```
--set nodeSelector.disktype=ssd
```

The same command can be added to the values yaml in this format:

```
nodeSelector:
disktype: ssd
```

Where, the IDSP pods are scheduled on node(s) with the label, disktype=ssd.

> **NOTE**
> For more information, refer to Create a pod that gets scheduled to your chosen node.

Affinity and Anti-affinity

The "affinity" parameter is the advanced version of the "nodeSelector" parameter that provides more flexibility like the ability to use the "in" operator. This parameter can be used in IDSP to instruct Kubernetes to schedule the IDSP pods only to a specific set of nodes with a specific label.

Example:
You can use the following command to add the label "disktype=ssd" to your node:

```
kubectl label nodes <your-node-name> disktype=ssd
```

Where, the <your-node-name> is the name of the node to which you want to add the label, disktype=ssd.

Use the following helm command-line parameter to schedule IDSP pods on nodes that have a label, `disktype=ssd`, using the operator value as "in".

```
--set
 affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms[0].matchExpressions[0].key=disk
 \
```

```
--set
 affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms[0].matchExpressions[0].operator
 \

--set
 affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms[0].matchExpressions[0].values[0
 \
```

The same command can be added to the values yaml in this format:

```
affinity:
nodeAffinity:
requiredDuringSchedulingIgnoredDuringExecution:
  nodeSelectorTerms:
  - matchExpressions:
    - key: disktype
      operator: In
      values:
      - ssd
```

Where, the IDSP pods are scheduled on node(s) that have a label, `disktype=ssd`, using the operator value as "in".

NOTE
For more information, see Schedule a pod using required node affinity.

Tolerations

The "tolerations" parameter is used in IDSP to instruct Kubernetes to allow IDSP pods to be scheduled to a specific set of "tainted" nodes. A "tainted" node rejects all pods that do not have a toleration to their taint.

Example:
Use the following command to set a node, such as node1 with a taint key1=value1:NoSchedule

```
kubectl taint nodes node1 key1=value1:NoSchedule
```

Where, none of the IDSP pods are able to schedule onto node1 unless it has a matching toleration.

You can add tolerations to IDSP using the following helm parameter in the command line:

```
--set tolerations[0].key="key1" --set tolerations[0].operator="Equal" --set tolerations[0].value="value1" --
set tolerations[0].effect="NoSchedule"
```

The same command can be added to the values yaml in this format:

```
tolerations:
- key: "key1"
operator: "Equal"
value: "value1"
effect: "NoSchedule"
```

Where, the IDSP pods are scheduled on node(s) that have the key1=value1:NoSchedule taint. The node here is node1 as mentioned in the example.

NOTE
For more information, refer to taint and tolerations.

Setting Labels

The "additionalLabels" parameter is used to add additional labels to all the IDSP (ssp) pods, Deployments, Jobs, Services, HorizontalPodAutoscaler objects, Hazelcast ConfigMaps, Secrets, and Pod Disruption Budget (PDBs).

For example, the following helm command-line parameter can be used to add the label, env=test to all the IDSP (ssp) pods:

```
--set additionalLabels.env=test
```

The same command can be added to the values yaml in this format:

Example:

```
additionalLabels:
env: test
```

Where, the label "env=test" is added to all the IDSP pods.

> **NOTE**
> The "additionalLabels" parameter does not apply to all the objects created by the IDSP (ssp) helm chart. Therefore, to apply labels to all the objects within the helm chart, use the following parameters:
>
> ```
> hazelcast-enterprise:
> labels: {}
> podLabels: {}
> service:
>     labels: {}
> global:
> additionalLabels: {}
> ```

Setting Annotations

The "additionalAnnotations" parameter is used to add additional annotations to all the IDSP (ssp) pods, Deployments, Jobs, Services, HorizontalPodAutoscaler objects, Hazelcast ConfigMaps, Secrets, and Pod Disruption Budget (PDBs).

Example: The following helm command-line parameter can be used to add the annotation, "env=test" to all the IDSP (ssp) pods:

```
--set additionalAnnotations.env=test
```

The same command can be added to the values yaml in this format:

```
additionalAnnotations:
env: test
```

Where, the annotation "env=test" is added to all the IDSP pods.

> **NOTE**
>
> The "additionalAnnotations" parameter does not apply to all the objects created by the IDSP (ssp) helm chart. Therefore, to apply labels to all the objects within the helm chart, use the following parameters:
>
> ```
> hazelcast-enterprise:
> annotations: {}
> service:
>     annotations: {}
> global:
> additionalAnnotations: {}
> ```

The following example shows the values in yaml format for setting both, ***annotations*** and Labels.

```
hazelcast-enterprise:
annotations:
 costCenter: "B05431"
labels:
 owner: "JohnD"
podLabels:
owner: "JohnD"
service:
annotations:
  costCenter: "B05431"
```

```
labels:
  owner: "JohnD"

global:
additionalAnnotations:
costCenter: "B05431"
additionalLabels:
owner: "JohnD"
```

Setting Ingress Timeouts

You can set the ingress communication timeout using the following parameters:

- For NGINX:
  - `ssp.ingress.timeoutSecs` : Defines the connection timeout for an ingress connection within the SSP (the default is 60 seconds).
- For OpenShiftRoute:
  - `ssp.ingress.openShiftRoute.timeout` :Defines the connection timeout for an OpenShift route within the SSP (the default is 60 seconds).
  - `ssp.ingress.openShiftRoute.tunnelTimeout` : Defines how long a tunnel connection in the OpenShift Ingress Controller (including websockets) remains open while idle (the default is 1 hour).

Set these parameters using an override values yaml command or using the `--set <param>=<values>` Helm command. For example in a Helm installation, use:

```
--set ssp.ingress.timeoutSecs=60
```

AWS AWS_STS_REGIONAL_ENDPOINTS

The parameter "aws_sts_regional_endpoints" allows setting the AWS AWS_STS_REGIONAL_ENDPOINTS environment variable in all the IDSP (ssp) pods.

```
ssp.global.env.aws_sts_regional_endpoints
```

If this parameter value is set, an environment variable called, AWS_STS_REGIONAL_ENDPOINTS containing that specified value is added to all the pods in the IDSP (ssp) chart.

Helm Offline Mode

The parameter "global.helmOfflineMode" allows to generate Kubernetes yaml files from ssp_helm chart using helm's --dry-run option, or helm template. This option is useful to generate a yaml manifest without actually deploying.

```
helm template <helm-repo>/<chart-name> <deployment parameters> --set global.helmOfflineMode=true
```

DNS-related configuration for IDSP Deployment

IDSP chart supports setting dnsPolicy and dnsConfig for IDSP Kubernetes pods. Update the dnsPolicy and dnsConfig settings in the IDSP values override yaml file.

```
global:

  #If dnsPolicy is not explicitly specified, then "ClusterFirst" is used.
  dnsPolicy: ""

  #When a Pod's dnsPolicy is set to "None", the dnsConfig field has to be specified
  dnsConfig: {}
```

Enabling Availability Metrics

This feature provides information on the availability and latency of the IDSP cache system. Enable the availabilityMetrics by specifying the following command-line parameter in the IDSP helm deployment command:

```
--set ssp.featureFlags.availabilityMetrics.enabled=true
```

Or use the following setting in the IDSP values override yaml file:

```
ssp:
featureFlags:
    availabilityMetrics:
        enabled: true
```

Seccomp Profile

This feature allows you to specify the "seccompProfile" value for your IDSP Kubernetes deployment. For more information, see seccomp and Kubernetes. The default value is: ***type: RuntimeDefault***.

Use the following setting in the IDSP values override yaml file:

```
global:
securityContext:
seccompProfile:
  type: RuntimeDefault
```

HTTP Proxy

This feature enables the use of an HTTP proxy for the '/token' request. This is necessary when IDSP, acting as a Service Provider, needs to access external endpoints but the cluster's direct communication with external OIDC providers is restricted. You can configure this setting during the deployment of either the IDSP chart or the Sample app chart. For IDSP, the use of "HTTP Proxy" is controlled by an IDP metadata setting called, "Use HTTP Proxy" (urn:iam:idp:useHttpProxy).

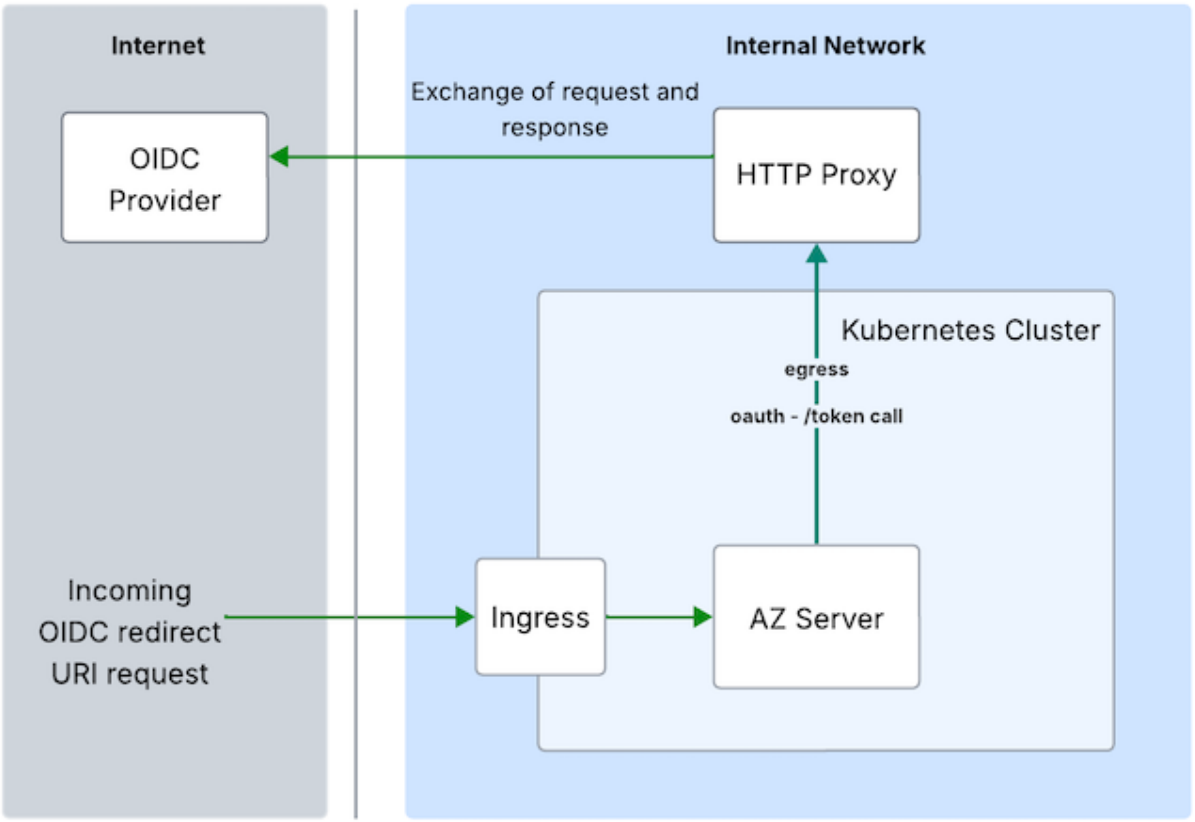
Use the following setting in the IDSP values override yaml file:

```
global:
proxy:
enabled: false
url: "https://httpproxy.example.com:443"
# if the proxy has authentication enabled, create a secret with username/password: kubectl create secret
 generic proxy-credentials --from-literal=username='<username>' --from-literal=password='<password>'
authEnabled: false
```

If your HTTP Proxy requires authentication, enable proxy authentication (`authEnabled: true`) in the previously shown values.yaml override file. Enabling proxy authentication requires to have a kubernetes secret named, "proxy-credentials" with username/password, as shown in the following command:

```
kubectl create secret generic proxy-credentials --from-literal=username='<username>' --from-
literal=password='<password>'
```

The HTTP Proxy scenario is explained with the help of the following illustration:



Steps:

1. Incoming OIDC redirect URI request with the AZ code coming to IDSP AZ server.
2. The Ingress routes the incoming request to the AZ server pod.
3. Traffic is routed to the OIDC provider's "/token" endpoint through the configured HTTP Proxy (requires IDP metadata setting called, "Use HTTP Proxy")
4. The HTTP Proxy forwards the request to the OIDC provider.
5. After receiving the token, the AZ Server completes the authentication process.

Managing Configuration Updates to the Symantec Identity Security Platform Helm Chart

Know how to make configuration updates to the deployed Symantec Identity Security Platform Helm chart by backing up current values, modifying them, and applying changes with a version-controlled upgrade.

Perform a configuration update of the IDSP helm chart

You can use the IDSP helm chart to perform deployment configuration updates. Whenever you must make a configuration change to an IDSP deployment, take a backup of the existing configuration by calling helm to get values, make the necessary config change in the values file, and run the 'helm upgrade' command.

Follow these steps:

1. Helm always uses the latest version of the IDSP chart, to guarantee we are just performing a configuration change and not an upgrade, check which version is deployed, and use it to force Helm to use this version. Save the version of the currently deployed chart by setting it as an environment variable:

```
#use the following command to get the list of all deployments in the namespace with their version.
#note that the CHART column shows the version in the format <chart-name>-<chart-version>, so, for IDSP it
 will show 'ssp-<CHART VERSION>'.

helm ls -n ${NAMESPACE}
#use this command to extract and save the current version of theIDSPchart:
export IDSP_VERSION=$(helm list --filter="${RELEASENAME}$" -n ${NAMESPACE} -o yaml |grep chart:| cut -d'-'
 -f2-)&&echo version=${IDSP_VERSION}
```

2. Save the existing chart configuration by getting the user-supplied values for the currently deployed IDSP release name and saving it into a values file:

```
helm get values ${RELEASENAME} -n ${NAMESPACE} -o yaml >exisiting_${RELEASENAME}_values.yaml
```

3. Save a copy of the resulting values file and make the desired configuration changes by updating it. Run the helm upgrade command using the updated values file:

```
helm upgrade ${RELEASENAME} -n ${NAMESPACE} ${HELM_REPO}/ssp --version=${IDSP_VERSION} --timeout=120m -f
 exisiting_${RELEASENAME}_values.yaml --set ssp.featureFlags.dataseed.enabled=false
```

   NOTE
   The configuration-update command has the "dataseed.enabled" set to "false".

4. Once you have performed the updates, save the updated value.

Reference Links

- Upgrading IDSP charts

Recovering Initial Admin Credentials

Learn about recovering the initial deployment credentials for "default-clients" and "defaultTenantadminUser" using the break-glass procedure after the 48 hours window lapses.

Before performing the Break-Glass procedure for the "default clients" and "DefaultTenantAdminUser", perform the following:

NOTE

- The commands to Break-Glass procedure use environment variables that are expected to be set in your environment. Before running those commands, ensure that these environment variables are set for your terminal session. See Configuring the Environment Variables.
- If required, such bootstrap credentials can be republished into Kubernetes secrets by re-deploying the chart with specific flags causing Kubernetes secrets to be re-created for another 48-hour period. To recreate secrets, you are required to rerun the helm chart with the CLI values as necessary to re-create the Kubernetes secrets. While doing so the dataseed needs to be executed. Therefore, ensure to set this parameter in the helm chart upgrade command - `--set ssp.featureFlags.dataseed.enabled=false`

Break-Glass procedure for the "default clients" (Infra, System, Tenant -Admin and Demo clients):

In case you missed saving the deployment Client IDs and their secrets within the 48 hours window, use the following break-glass procedure (valid for the next 48 hours):

```
helm get values ${RELEASENAME} -n ${NAMESPACE} -o yaml >exisiting_${RELEASENAME}_values.yaml
```

```
#use this command to extract and save the current version of the IDSP chart
export IDSP_VERSION=$(helm list --filter="${RELEASENAME}$" -n ${NAMESPACE} -o yaml |grep chart:| cut -d'-' -
f2-)&&echo version=${IDSP_VERSION}

helm upgrade ${RELEASENAME} -n ${NAMESPACE} ${HELM_REPO}/ssp --version=${IDSP_VERSION} --timeout=20m -f
 exisiting_${RELEASENAME}_values.yaml \
  --set ssp.featureFlags.dataseed.enabled=true \
  --set ssp.global.seedData.defaultTenant.adminClient.disclosedOnUpgrade=true \
  --set ssp.global.seedData.defaultTenant.demoClient.disclosedOnUpgrade=true \
  --set ssp.global.seedData.systemClient.disclosedOnUpgrade=true \
  --set ssp.global.seedData.infraClient.disclosedOnUpgrade=true
```

Once the Kubernetes secret is created again, use the command mentioned in the Retrieving Initial Credentials for the clients' section to retrieve the "default clients" secrets.

Break-Glass procedure for the "DefaultTenantAdminUser":

In case you missed the 48 hours window and did not create an alternate administrative identity, use the following break-glass procedure to enable the default tenant admin user, reset its password and recreate its Kubernetes secret (valid for the next 4

```
helm get values ${RELEASENAME} -n ${NAMESPACE} -o yaml >exisiting_${RELEASENAME}_values.yaml

helm upgrade ${RELEASENAME} -n ${NAMESPACE} ${HELM_REPO}/ssp --version=${IDSP_VERSION} --timeout=20m
 -f exisiting_${RELEASENAME}_values.yaml --set ssp.featureFlags.dataseed.enabled=true --set
 ssp.global.seedData.defaultTenant.adminUser.seededOnUpgrade=true
```

Once the Kubernetes secret is created again, use the command mentioned in the Retrieving Admin Users Initial Credentials section above to retrieve the "DefaultTenantAdminUser" initial password from the secret.

Handling Database Password Rotation

Learn about database password rotation.

Symantec Identity Security Platform Database (DB) password is stored in a Kubernetes Secret and mounted as a folder on the containers that require access to the DB. When mounted, Kubernetes copies the Secret's data into tmpfs (so that the confidential data is not written to durable storage), and keeps it updated with the secret data, using an eventually consistent approach. It is recommended that you create a secret containing a file called 'mysql-password' where the db password is stored, and specify its name during IDSP Database deployment using the "ssp.db.existingSecert" parameter. IDSP reads the password file from the mounted folder. The DB password is reloaded from this file when changed.

If you are using a vault to inject the DB password file into the container, you can use "ssp.db.existingVaultFolder" to indicate an alternate folder name. When this parameter is specified, IDSP will not mount the db secret and expect the folder to contain a file named 'mysql-password' with the current clear-text db password.

Scheduler Job

Learn about the **Scheduler Job that automatically does the clean-up of expired artifacts** and also the manual way of cleaning up the artifacts using the SQL commands.

Automatic Cleanup of Expired Artifacts

This document outlines the process for cleaning up expired database artifacts, primarily focusing on the automated scheduler job. The scheduler is responsible for cleaning up the following artifacts:

- **Expired Token Clean-up** -

  - **Authorization code** - An hourly process cleans up the OIDC Authorization codes that were abandoned and have never been redeemed.
  - **Device code** - An hourly process cleans up the device codes that were abandoned and have never been redeemed.
  - **Refresh token** - An hourly process cleans up the expired refresh tokens.
  - **Revoked token**- An hourly process cleans up expired revoked tokens. Revoked tokens are created to signify that a token has been cancelled, identified by its session ID (sid) and JWT ID (jti). More specifically:
    - If either the "trackAccessToken" or "trackIdentityToken" setting (at the tenant or global level) is enabled, Authentication Hub records revoked tokens by their jti. These "RevokedToken" objects are removed once the corresponding access token (AT) or identity token (IT) expires.
    - When a refresh token is revoked, a "RevokedToken" object is created for its session ID (sid) and is removed once the refresh token expires.
    - If the "ssoMode" tenant setting is disabled, revoking a session creates a "RevokedToken" object, which is deleted when the session expires.
  - **Credential Reset Token Service** - An hourly process cleans up the credential(password) reset tokens which are issued during various flows('Forgot Password'/'Account unlock'/'Account Activation' and are not redeemed and expired.
  - **Security Token** - An hourly process cleans up the credential(email/phone) verification tokens which are never redeemed and expired.

  To run the scheduler at a custom time, configure the following setting along with the IDSP installation or upgrade command:

```
--set ssp.featureFlags.scheduler.expiredTokensCleanup.fixedDelayInMilliseconds="3600000"
--set ssp.featureFlags.scheduler.expiredTokensCleanup.batchSize="5000"
```

- **FlowState** - this process is repeated every ten minutes to clean up the expired flowstate objects.

  To run the scheduler at a custom time, configure the following setting along with the IDSP installation or upgrade command:

```
--set ssp.featureFlags.scheduler.expiredFlowStateCleanup.fixedDelayInMilliseconds="600000"
--set ssp.featureFlags.scheduler.expiredFlowStateCleanup.batchSize="5000"
```

- **Kubernetes Secrets older than a certain time** (as configured in the Authentication Hub Helm chart parameter- **ssp.global.seedData.disclosureExpiryHours**) are removed from the Kubernetes cluster.
- **Purging Known IA risk tags for expired and deleted devices** - Known IA risk tags for devices that have expired or been deleted are purged daily (every 24 hours), only when risk is configured as true in the Authentication Hub Helm chart parameter- **ssp.featureFlags.iarisk.enabled**. See Known IA Risk Tags Purging Scheduler
- **Risk Active User Event Data Clean-up** - This scheduler job runs every 30 minutes to purge old risk events associated with active users. By default, events older than 730 days are removed. This retention period can be adjusted using the global setting, "riskRetentionDays", and the cleanup job itself is enabled by configuring the global setting, "isRiskCleanupEnabled" to true. (It is disabled by default). See Risk Active User Event Data Clean-Up Scheduler
- **Risk Inactive User Event Data Clean-up** - This scheduler job runs weekly, every Sunday, to purge old risk events associated with inactive users. The default retention period is 730 days, configurable through the global setting, "riskRetentionDays". This cleanup job is disabled by default and can be enabled by configuring the global setting, "isRiskInactiveUsersEventsCleanupEnabled" to true. See Risk Inactive User Event Data Clean-Up Scheduler
- **Inactive Device Purging Scheduler** - This is an automated maintenance job designed to manage device data retention in the system. This scheduler helps maintain optimal database performance by removing devices that are marked as inactive (deleted=1) and have not been accessed for a specified number of days, preventing device tables from growing indefinitely. See Inactive Device Purging Scheduler

Automatic Clean-up of IDSP objects that refers to orphan records

- **resourcesCleanupScheduler** - This scheduler job runs every 30 minutes to clean up any dangling references to the deleted resource in other operational objects.
  Example- If an AuthN policy contains an application under "apps" and that app is forcefully deleted, in this scenario, AuthN policy maintains a reference to the deleted app in the IDSP database, until scheduler job cleans up this policy to remove the dangling reference.
  To run the scheduler at a custom time, configure the following setting along with the IDSP installation or upgrade command:
  ```
  --set ssp.featureFlags.scheduler.resourcesCleanup.fixedDelayInMilliseconds="1800000"
  ```

Enabling Scheduler

This scheduler job is **enabled by default during** Symantec Identity Security Platform **deployment**. You can verify its enabled status through the following feature flag:

```
ssp:
  featureFlags:
    scheduler:
      enabled: true
```

While the scheduler handles most of the cleanup, it is important to note that if the scheduler job does not run by default for any reason, you can manually perform the clean-up using the manual commands mentioned in the next section.

Manual Cleanup of Expired Artifacts

To effectively manage your database and ensure optimal performance, it is recommended to periodically remove expired artifacts. This cleanup activity helps free up space and maintain data integrity.

To view and delete these expired entries for each artifact type, you can periodically execute the following SQL commands in the database:

- **Authorization Code (gets cleaned up every one hour)**
  View expired authorization codes:
  ```
  select * from T_AUTHORIZATION_CODE where EXPIRES_AT < sysdate();
  ```
  Delete expired authorization codes:
  ```
  delete from T_AUTHENTICATION_HOLDER where EXPIRES_AT < sysdate();
  delete from T_AUTHORIZATION_CODE where EXPIRES_AT < sysdate();
  ```
- **Device Code (gets cleaned up every one hour)**
  View expired device codes:
  ```
  select * from T_DEVICE_CODE where EXPIRES_AT < sysdate();
  ```
  Delete expired device codes:
  ```
  delete from T_DEVICE_CODE where EXPIRES_AT < sysdate();
  ```
- **Refresh Token (gets cleaned up every one hour)**
  View expired refresh tokens:
  ```
  select * from T_REFRESH_TOKEN where EXPIRES_AT < sysdate();
  ```
  Delete expired refresh tokens:
  ```
  delete from T_AUTHENTICATION_HOLDER where EXPIRES_AT < sysdate();
  delete from T_REFRESH_TOKEN where EXPIRES_AT < sysdate();
  ```
- **Revoked Token (gets cleaned up every one hour)**
  View expired revoked tokens:
  ```
  select * from T_REVOKED_TOKEN where EXPIRES_AT < sysdate();
  ```
  Delete expired revoked tokens:

```
delete from T_REVOKED_TOKEN where EXPIRES_AT < sysdate();
```

- **Flow State (gets cleaned up every ten minutes)**
  View expired flow state entries:
  ```
  select * from T_FLOW_STATE where EXPIRES_AT < sysdate();
  ```
  Delete expired flow state entries:
  ```
  delete from T_FLOW_STATE where EXPIRES_AT < sysdate()
  ```
- **Security Token (gets cleaned up every one hour)**
  View expired security token entries:
  ```
  select * from T_IDS_SECURITY_TOKEN where EXPIRATION_DATETIME < sysdate()
  ```
  Delete expired security token entries:
  ```
  delete from T_IDS_SECURITY_TOKEN where EXPIRATION_DATETIME < sysdate()
  ```
- **Credential Reset Token Service (gets cleaned up every one hour)**
  View expired Credential Reset Token entries:
  ```
  select  * from T_IDS_CREDENTIAL_RESET_TOKEN  where EXPIRATION_DATETIME < sysdate()
  ```
  Delete expired Credential Reset Token entries:
  ```
  delete from T_IDS_CREDENTIAL_RESET_TOKEN  where EXPIRATION_DATETIME < sysdate()
  ```
- **Expired Kubernetes Secrets (gets cleaned up every one hour)**
  Delete Kubernetes Secrets older than a certain time (as configured in the Authentication Hub Helm chart parameter- **ssp.global.seedData.disclosureExpiryHours**):
  ```
  kubectl delete secret <secret-name> ns <namespace>
  ```
- **Purging Known IA risk devices tags (gets cleaned up every 24 hours)**
  View tags for expired and deleted known IA risk devices:
  ```
  select * from IA_ISSUED_TAGS WHERE (sysdate() > EXPIRE_TIMESTAMP) OR DELETED = 1;
  ```
  Delete tags for expired and deleted known IA risk devices:
  ```
  DELETE FROM IA_ISSUED_TAGS WHERE (sysdate()> EXPIRE_TIMESTAMP) OR DELETED = 1;
  ```

Risk Active User Event Data Clean-up Scheduler

Learn about Risk active user event data clean-up scheduler.

The Risk Active User Event Data Clean-up Scheduler is an automated maintenance job designed to manage data retention for risk assessment events in the system. This scheduler helps maintain optimal database performance by removing outdated risk event data while preserving recent activity for security analysis.

This scheduler job runs every 30 minutes to purge old risk events associated with active users. By default, events older than 730 days are removed. The retention period can be adjusted using the global setting, "riskRetentionDays", and the cleanup job is enabled by configuring the global setting, "isRiskCleanupEnabled" to true. (Disabled by default).

NOTE

- Before running the **Risk Active User Event Data Clean-up** scheduler, note that the initial data volume may be large. It is recommended to perform a manual data clean-up beforehand to help ensure the scheduler runs smoothly and avoids longer execution times or added load on the database.
- The database scripts for cleaning up the initial data can be downloaded from the Support Portal > My Downloads (Search the Identity Security Platform in the portal, select the appropriate version number, and download the "Initial_Cleanup_db_scripts"). Ensure these scripts are first executed in the staging environment and reviewed by your DBAs before running them in production.

Database Impact

The cleanup process affects the following database tables:

- IA_EVENT
- IA_ENTITY
- IA_SUSPICIOUS_EVENTS
- IA_EVENT_CLUSTER
- IA_CASES_EVENTS
- IA_CLUSTER_DATA

Scheduler Configuration

The following table outlines the key scheduler parameter that controls when and how often the cleanup process runs:

Job Name	Value	Description
job.purge.risk.eventcleanup.fixedDelay	1800000	Execution interval in minutes (30 minutes)

Global Settings

The following settings control the behavior and scope of the risk event cleanup process:

Setting Name	Default Value	Supported Values
isRiskCleanupEnabled	false	true/false
riskRetentionDays	730	1-9999

Customizing Scheduler Settings

To run the scheduler operations at a custom time, configure the following settings along with the Symantec Identity Security Platform installation or upgrade command:

```
--set ssp.featureFlags.scheduler.riskEventCleanup.fixedDelayInMilliseconds="1800000"
--set ssp.featureFlags.scheduler.riskEventCleanup.userBatchSize="250"
--set ssp.featureFlags.scheduler.riskEventCleanup.eventBatchSize="2000"
--set ssp.featureFlags.scheduler.riskEventCleanup.deleteBatchSize="1000"
```

Risk Inactive User Event Data Clean-up Scheduler

Learn about Risk inactive user event data clean-up scheduler.

The Risk Active User Event Data Clean-up Scheduler is an automated maintenance job designed to manage data retention for risk assessment events in the system. his scheduler helps maintain optimal database performance by removing outdated risk event data while preserving recent activity for security analysis.

This scheduler job runs weekly, every Sunday, to purge old risk events associated with inactive users. The default retention period is 730 days, configurable through the global setting, "riskRetentionDays". This cleanup job is disabled by default and can be enabled by configuring the global setting, "isRiskInactiveUsersEventsCleanupEnabled" to true.

NOTE

- Before running the **Risk Inactive User Event Data Clean-up** scheduler, note that the initial data volume may be large. It is recommended to perform a manual data clean-up beforehand to help ensure the scheduler runs smoothly and avoids longer execution times or added load on the database.
- The database scripts for cleaning up the initial data can be downloaded from the Support Portal > My Downloads (Search the Identity Security Platform in the portal, select the appropriate version number, and download the "Initial_Cleanup_db_scripts"). Ensure these scripts are first executed in the staging environment and reviewed by your DBAs before running them in production.

Database Impact

The cleanup process affects the following database tables:

- IA_EVENT
- IA_ENTITY
- IA_SUSPICIOUS_EVENTS
- IA_EVENT_CLUSTER
- IA_CASES_EVENTS
- IA_CLUSTER_DATA

Scheduler Configuration

The following table outlines the key scheduler parameter that controls when and how often the cleanup process runs:

Job Name	Value	Description
job.purge.risk.inactive.usercleanup.cron	0 0 0 * * SUN	Execution interval weekly (Every Sun

Global Settings

The following settings control the behavior and scope of the risk event cleanup process:

Setting Name	Default Value	Supported Values
isRiskInactiveUsersEventsCleanupEnabled	false	true/false
riskRetentionDays	730	1-9999

Customizing Scheduler Settings

To run the scheduler operations at a custom time, configure the following settings along with the Symantec Identity Security Platform installation or upgrade command:

```
--set ssp.featureFlags.scheduler.riskInactiveUserEventsCleanup.cron="0 0 0 * * SUN"
--set ssp.featureFlags.scheduler.riskInactiveUserEventsCleanup.userBatchSize="250"
--set ssp.featureFlags.scheduler.riskInactiveUserEventsCleanup.eventBatchSize="2000"
--set ssp.featureFlags.scheduler.riskInactiveUserEventsCleanup.deleteBatchSize="1000"
```

Known IA Risk Tags Purging Scheduler

Learn about the Known IA risk tags purging scheduler for managing expired and deleted device risk tags.

The Known IA Risk Tags Purging Scheduler is an automated maintenance job designed to manage risk tag data retention in the system. This scheduler helps maintain optimal database performance by removing known IA risk tags for devices that have expired or been deleted, preventing risk tag tables from growing indefinitely.

This scheduler job runs daily (every 24 hours) to purge known IA risk tags for devices that satisfy the following criteria:

- Devices that have expired or been deleted from the system
- The purging functionality is only active when risk is configured as true in the Authentication Hub Helm chart parameter ssp.featureFlags.iarisk.enabled.

NOTE

- Before enabling the **Known IA Risk Tags Purging** Scheduler, consider the initial data volume that may need to be processed. It is recommended to review the current number of inactive devices to ensure the scheduler runs efficiently.
- Monitor the scheduler execution times and database performance during initial runs to optimize the purging schedule if needed.

Database Impact

The cleanup process affects the following database table:

- IA_ISSUED_TAGS

Scheduler Configuration

The risk tag purging process runs daily (every 24 hours). The following table outlines the scheduler configuration:

Job Name	Value	Description
job.purge.issuedtag.cron	0 0 2 * * ?	Daily execution at 2:00 AM

Customizing Scheduler Settings

To run the scheduler operations at a custom time, configure the following settings along with the Symantec Identity Security Platform installation or upgrade command:

```
--set ssp.featureFlags.scheduler.issuedTagPurge.cron="0 0 20 * * ?"
--set ssp.featureFlags.scheduler.issuedTagPurge.batchSize="5000"
```

Inactive Device Purging Scheduler

Learn about the inactive device purging scheduler for managing deleted devices.

The Inactive Device Purging Scheduler is an automated maintenance job designed to manage device data retention in the system. This scheduler helps maintain optimal database performance by removing devices that are marked as inactive (deleted=1) and have not been accessed for a specified number of days, preventing device tables from growing indefinitely.

This scheduler job runs on a configurable schedule to purge devices that satisfy the following criteria: devices not accessed in the last "N" days (where "N" is defined by global settings) and devices marked as deleted in the database. The retention period and cleanup functionality can be controlled through global settings.

NOTE

- Before enabling the **Inactive Device Purging** scheduler, consider the initial data volume that may need to be processed. It is recommended to review the current number of inactive devices to ensure the scheduler runs efficiently.
- Monitor the scheduler execution times and database performance during initial runs to optimize the purging schedule if needed.

Database Impact

The cleanup process affects the following database table:

- IA_DEVICES

Scheduler Configuration

The following table outlines the key scheduler parameter that controls when and how often the cleanup process runs.

Job Name	Value	Description
job.purge.device.cron	0 0 2 * * ?	Daily execution at 2:00 AM

Global Settings

The following settings control the behavior and scope of the inactive device cleanup process:

Setting Name	Default Value	Supported Values
isRiskDeviceCleanupEnabled	false	true/false
riskDeviceRetentionDays	730	1-9999

Customizing Scheduler Settings

To run the scheduler operations at a custom time, configure the following settings along with the Symantec Identity Security Platform installation or upgrade command:

```
--set ssp.featureFlags.scheduler.devicePurge.cron="0 0 20 * * ?"
--set ssp.featureFlags.scheduler.devicePurge.batchSize="5000"
```

Populating the Internal Email Domain Column

You can search for users by email domain by using the domain column in the `T_USER_EMAIL` table. The Identity Service SCIM API supports an indexed based search using the `T_USER_EMAIL.domain` column.

Creating or updating a user's email address automatically populates the domain column.

> **NOTE**
> For email addresses populated prior to AuthHub 3.1, the newly added domain column is empty.

To populate empty domain columns in email records, deployments can execute the following SQL statement:

```
UPDATE T_USER_EMAIL SET DOMAIN = SUBSTRING_INDEX(VALUE,'@',-1);
```

Symantec Identity Security Platform Upgrade Procedure with No Active Database Transactions

Step-by-step procedure to perform an IDSP Helm upgrade in a Kubernetes environment when there are no active transactions on the database.

Overview

This topic outlines the procedure to perform an IDSP (Identity Service Provider) Helm upgrade in a Kubernetes environment when there are no active transactions on the database. The process involves backing up autoscaler and deployment configurations, gracefully scaling all IDSP services down to zero replicas, and then executing the Helm upgrade with the preserved override values so that pods scale back to their defined counts post-upgrade.

Steps to Upgrade with No Transactions on Database

Follow the steps to perform a safe IDSP upgrade:

1. Take a Backup of Existing HorizontalPodAutoscaler (HPA) Details
   Capture the current HPA configuration for the target namespace:
   ```
   kubectl get hpa -n <namespace> > idsp_hpa_<namespace>
   ```
2. Get the IDSP Deployment Values to a Backup File
   Export the current Helm release values to an override file:
   ```
   helm get values ${RELEASENAME} -n ${NAMESPACE} > idsp-override.yaml
   ```
3. Resize the Deployments of IDSP Services to ZERO
   Scale all IDSP service deployments down to zero replicas before proceeding with the upgrade.
   There are two approaches to scale the IDSP service deployments down to zero replicas:

   - Manual Resize (Per Deployment)
   - Scripted Resize

   Manual Resize (Per Deployment)

   This is the recommended approach.

   Fetch the deployments in the IDSP namespace:

   ```
   kubectl get deployments -o name -n <namespace>
   ```

   Resize the replica count to ZERO for each deployment:

   ```
   kubectl scale deployment/<deploymentName> --replicas=0 -n <namespace>
   ```

   Scripted Resize

   To resize all deployments to zero pod count at once, use the following script as a reference. The script does the following:

   1. Fetches all deployment names into a temporary file (`/tmp/deployments`).
   2. Reads the file line by line and scales each deployment's replica count to ZERO in a loop until every deployment has been processed.

      NOTE
      Replace the `${NAMESPACE}` value with only the IDSP namespace. Providing a non-IDSP namespace scales all deployments in that namespace to zero, including those unrelated to IDSP.

   Script: ScaleIDSPDeploymentsToZero.sh

   ```
   export NAMESPACE=$1
   echo "NAMESPACE ToBe Modified is ${NAMESPACE}"

   rm -f /tmp/deployments

   kubectl get deploy -n ${NAMESPACE} -o name | cut -d'/' -f2 > /tmp/deployments

   while IFS= read -r line
   do
       echo $line
       kubectl scale deployment/$line --replicas=0 -n ${NAMESPACE}
   done < /tmp/deployments
   ```

   Run the script with the namespace as the argument:

   ```
   ScaleIDSPDeploymentsToZero.sh <namespace>
   ```

4. **Perform the Helm Upgrade**

   Run the Helm upgrade using the same build version that was previously deployed and the previously saved override file (which contains scaling details such as replicaCount, HorizontalPodAutoscaler settings, and similar configurations). After the upgrade completes, the pods scale back to the replica counts defined in the override values.

   Include the following flag to disable data seeding during the upgrade:

   ```
   --set ssp.featureFlags.dataseed.enabled=false
   ```

   To know about the IDSP Upgrade commands, see Upgrading Key definition for "pname" not found in the DITA map.

Troubleshooting

Use this section to identify and troubleshoot problems associated with Symantec Identity Security Platform.

- Using Logs to Troubleshoot Operational Errors
- General Issues
- Enclave Service Issues
- GCP Related Issues
- Database Connection Issues
- How to trace transactions between distributed services

Using Logs to Troubleshoot Operational Errors

Learn how to use logs to troubleshoot operational errors in Symantec Identity Security Platform.

By default, IDSP maintains operational and audit logs that contain data pertaining to each request processed. For example, this data can be seen in Kibana or in some other observability tools, when integrated.

Among other request-specific artifacts, each log entry contains transaction-identifying elements that can be used to observe, correlate, and troubleshoot ongoing operations. These identifiers are prefixed with the specified region when configured with `-- set ssp.deployment.region="<regionName>"` parameter in the IDSP deployment. Information about **Region Identifier** is also covered in Additional Parameters to consider adding before deploying the IDSP Helm Chart. Search for the term, **Region Identifier** in this page.

1. Use of "txnId" and "clientTxnId" in logs enables clients to correlate their logs and IDSP logs:
   - txnId - represents a unique identifier that is assigned to each incoming request. Each request that is processed is guaranteed to have a unique id that is maintained and is included in every log message throughout the processing of the request. The transaction id is included in the response returned to a client making the request in the response header X-TRANSACTION-ID.
   - clientTxnId - represents an identifier that is sent by a client making the request (via the X-CLIENT-TRANSACTION-ID header). The client would use the same client transaction Id for all the requests that should be associated together. Each message being logged by IDSP as part of these requests contain this client transaction id. For example, the Admin Console app sends the same clientTxnId for all the requests that are sent as part of the user's working session when using the admin console.
2. Use of "flowStateId" to track a user authentication ceremony across different requests. This id is associated with all the messages that are part of the authentication flow. Searching for messages with a specific flowstateId results in getting all the log messages of the end-to-end authentication flow that covers all requests from the moment authentication service is invoked until the authentication flow completes. For example, Factor Selection and OTP generation/ verification, and any other request in the same flow will have the same flowStateId.
3. Use of "clientId", "effectiveClientId", "clientIp", "appId", "appName" to track an application client making the request.
4. Use of "userGuid", "userLoginId", "userIp", "userUniversalId", and so on, to track a user on whose behalf a request is made.
5. Use of "api" to track the request being made.

General Issues

This section discusses how to troubleshoot general issues that you may encounter during the usage of the Symantec Identity Security Platform.

This topic includes the following sections:

- Slowness in IDSP Services- Check Entropy Regeneration Levels

Slowness in IDSP Services- Check Entropy Regeneration Levels

If you are experiencing slowness in your IDSP services, run this command to check the rate at which the entropy gets regenerated:

NOTE
Before running this command, ensure that the environment variable is set to your IDSP namespace.

```
kubectl exec -n ${NAMESPACE} $(kubectl get pods -n ${NAMESPACE} -l app=ssp-azserver -o name | head -1) -- time
 dd if=/dev/random of=/dev/null bs=1 count=1M
```

The expected time for generating 1mb of random data should be about 2 seconds. If you get a higher number, refer to the Entropy section.

Enclave Service Issues

This section discusses how to overcome issues with Prometheus, Elastic Search, etc.

This topic includes the following:

Unable to create an API-specific metrics map

Follow these steps to validate:

1. Validate a service generates metrics by running the following commands in the master node:
   ```
   kubectl get svc -n $(NAMESPACE)
   ```
   ```
   curl http://<cluster-ip>:<port>/actuator/prometheus -i -X GET
   ```
2. Debug Prometheus UI to check if metrics are getting scraped. Run this command in your work environment:
   ```
   kubectl port-forward -n monitoring prometheus-prometheus-operator-prometheus-0 9090
   ```
   <not sure what is meant here> you must Control-C out of it after validating next step.
3. In your work environment, open the browser and then open the "Prometheus UI". Look for http_server_... metrics.
   ```
   http://localhost:9090/graph
   ```
4. Check if monitoring dashboards reflects the current status. You can ensure this by checking the value set for the refresh interval,(default value-5secs) which should not be too long. The dashboard may take a minute or two to display the current status or the latest metrics.

Deleting Existing Index in Elastic Search

Run this command from one of the worker nodes:

```
curl -X DELETE <es-pod-ip>:9200/ssp_log
```

Where, the ip address is one of the nodes where ElasticSearch is running and ssp_log is the name of the index.

Deleting Index Pattern from Kibana

1. Run this command to find the GUID of the pattern:
   ```
   curl -XGET http://<kibana-pod-ip>:5601/api/saved_objects/index-pattern/ 165
   ```
2. Delete the index pattern by running this command:
   ```
   curl -XDELETE http://kibana-pod-ip:5601/api/saved_objects/index-pattern/<pattern-guid> 31 -H "kbn-xsrf:
    true"
   ```

Kibana Dashboard Does Not Display Any Output for Index Pattern

Take a look at the pod logs for Elastic and Kibana. It is possible that your Elastic Search or Kibana needs a restart. Delete the Elastic pod, and then the Kibana pod. The pods will auto restart.

Wait a little, then go to the kibana console to see if the logs are reappearing.

GCP Related Issues

This section discusses how to troubleshoot issues that you may encounter while setting up Symantec Identity Security Platform on Google Cloud Platform (GCP).

Issue

Error from server (Forbidden): If you are unable to run "**kubectl create clusterrolebinding cluster-admin-binding --clusterrole cluster-admin --user <USER_ACCOUNT>**" command to grant, cluster admin permission to your gcp user account, and you are getting the following error:

```
Error from server (Forbidden): clusterrolebindings.rbac.authorization.k8s.io is forbidden: User
 "john.doe@broadcom.com" cannot create resource "clusterrolebindings" in API group "rbac.authorization.k8s.io"
 at the cluster scope: Required "container.clusterRoleBindings.create" permission.
```

Solution:

1. Navigate to the *Google Cloud IAM site and click the "Go to Console" tab.*
2. Ensure your GPC user is assigned the role of "Kubernetes Engine Admin."

Database Connection Issues

This topic includes the following sections:

- DB SSL Connection Issue
- MySQL "RSA public key is not available client side (option serverRsaPublicKeyFile not set)" error
- An error occurs when connecting to a MySQL database if the password is set using a different character set

DB SSL Connection Issue

This section discusses how to overcome db ssl connection issue that you may encounter during its usage in Symantec Identity Security Platform.

1. Try connecting to the MySQL Server from a host with openssl tool and point to the db host or use localhost as shown in the following example, when running on the machine or pod of the db:
   ```
   openssl s_client -connect localhost:3306 -showcerts </dev/null
   ```
2. To validate MySQL's SSL configuration, access mysql as root from mysql client:
   - show variables like '%ssl%';
   - show status like '%ssl%';
3. To validate IDSP processes' connection to MySQL is over SSL, access mysql as root from mysql client. The IP addresses should correspond to the ip addresses of the Kubernetes pods running in IDSP micro services:
   ```
   SELECT PROCESSLIST_USER,PROCESSLIST_HOST,PROCESSLIST_INFO,CONNECTION_TYPE FROM performance_schema.threads
   where PROCESSLIST_USER="iamauth";
   ```

MySQL "RSA public key is not available client side (option serverRsaPublicKeyFile not set)" error

IDSP supports connecting to MySQL DB using a "`mysql_native_password` " user.

Error:

When connecting to a MySQL DB using unencrypted connection (`ssp.db.sslMode=DISABLED`), the "`RSA public key is not available client side (option serverRsaPublicKeyFile not set)` " error gets displayed.

Cause:

This indicates that the database user (`ssp.db.user`) is stored on the mysql database server-side using "`caching_sha2_password` " **instead of** "`mysql_native_password` ".

Solution:

Solution 1: To rectify this issue, set the "ssp.db.user" to specify a MySQL "mysql_native_password" user (see Customer Database Mode).

Solution 2: Set the parameter "ssp.db.jdbcOpts" to "allowPublicKeyRetrieval=true&useSSL=false" in the deployment yaml file.

An error occurs when connecting to a MySQL database if the password is set using a different character set.

Error:

When connecting to a MySQL DB using "mysql_native_password", you may encounter this error: "`41 more \nCaused by: java.sql.SQLInvalidAuthorizationSpecException: Could not connect to address=(host=34.118.228.175)(port=3306)(type=master) : (conn=12) Access denied for user 'root1'@'10.60.3.44' (using password: YES)\nCurrent charset is UTF-8. If password has been set using other charset, consider using option 'passwordCharacterEncoding'\n\tat org.mariadb.jdbc.internal.util.exceptions.ExceptionFactory.createException(ExceptionFactory.java \tat org.mariadb.jdbc.internal.util.e` "

Cause: You may encounter a password charset UTF issue if there are compatibility problems between the operating system and the Docker environment.

Solution:

To rectify this issue, set the following command in the IDSP deployment yaml file or pass it along with IDSP deployment command, "`--set ssp.db.jdbcOpts='useUnicode=yes&characterEncoding=UTF-8` ".

Tracing

Jaeger is an open-source tool used for tracing transactions between distributed services. It enables monitoring and troubleshooting of complex micro-services.

Symantec Identity Security Platform is developed with OpenTracing compliant tracing capabilities. Jaeger client library is used for tracing purposes. The profiling data from services can be fed to OpenTracing compliant Jaeger Agent for profiling.

Suggested Namespace	tracing
Chart Name	jaeger-operator
Chart Helm Repo	https://jaegertracing.github.io/helm-charts
Service URL	https://jaeger.<DOMAIN>
Tested Chart version	2.57.0

> **NOTE**
> The procedures discussed in this topic to deploy, enable, and configure Jaeger are applicable to Cloud Service Providers, OpenShift, and On-Premise deployments.

Deploying Jaeger for Tracing

1. Deploy Jaeger by the following these steps:
   a. Create a tracing namespace.
      ```
      kubectl create ns tracing
      ```
   b. Execute this command to apply Kubernetes Custom Resource Definitions (CRDs) for cert-manager:

```
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.13.3/cert-manager.yaml
```

c. Locate the Jaeger tracing helm chart here:

```
helm repo add jaegertracing https://jaegertracing.github.io/helm-charts
```

d. Get the latest helm chart updates by running this command:

```
helm repo update
```

2. If you want to add your own trusted certificate signed by the Certificate Authority (CA), run this command:

NOTE

- If you do not have a certificate, you can skip this as the Tracing ingress uses a self-signed certificate.
- Before running this command, ensure that the variables, ${CERTFILE} and ${KEYFILE} are set.

```
kubectl create secret tls tracing-general-tls --cert "${CERTFILE}" \
--key "${KEYFILE}" -n tracing
```

3. Deploy Jaeger.

NOTE

- Before running this command, ensure that the variable, ${DOMAIN} is set.
- If you are required to authenticate to your container-registry to pull images, do the following:
  - Create a Kubernetes secret of type docker-registry. The following command creates a secret for the Docker Hub container registry:

```
kubectl create secret docker-registry <registry-secret-name> -n tracing \
  --docker-server="https://index.docker.io/v2/" \
  --docker-username="<dockerhub-username>" \
  --docker-password="<dockerhub-password>" \
  --docker-email="<your-email-address>"
```

  - Add the following parameter to the helm installation Jaeger command:

```
--set image.imagePullSecrets[0].name="<registry-secret-name>"
```

```
helm install jaeger-operator -n tracing jaegertracing/jaeger-operator \
--set jaeger.create=true \
--set jaeger.spec.ingress.security=none \
--set rbac.clusterRole=true \
--set jaeger.spec.ingress.enabled=true \
--set jaeger.spec.ingress.ingressClassName=nginx \
--set jaeger.spec.ingress.hosts[0]=jaeger.${DOMAIN} \
--set jaeger.spec.ingress.tls[0].secretName=tracing-general-tls \
--set jaeger.spec.ingress.tls[0].hosts[0]=jaeger.${DOMAIN} \
--set jaeger.spec.storage.type=memory \
--version '2.57.0'
```

4. Verify that the pods are in Running State and Ready (1/1), by running this command:

```
kubectl get pods -n tracing
```

The following image is a sample displaying the tracing pods with the *STATUS-Running*:

```
$ kubectl get pods -n tracing
NAME                                      READY   STATUS    RESTARTS   AGE
jaeger-operator-848bcb46b-sgc9w           1/1     Running   0          34d
jaeger-operator-jaeger-695ff64774-8dwhc   1/1     Running   0          34d
```

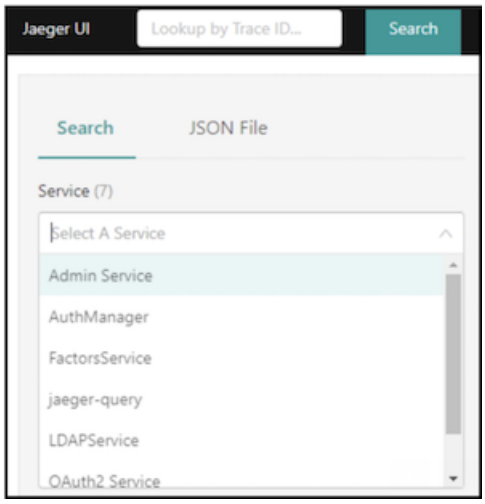
Enabling Jaeger Tracing

Jaeger Tracing is disabled by default in the solution. However, if you want to enable tracing for the services in the solution, you can either enable it during the deployment of the solution or post-deployment.

- To enable tracing globally for IDSP microservices, add the attribute, `--set ssp.global.tracing.enabled=true` to the jaeger-operator helm Install command. For example: helm install<helmRepo>/ssp --set ssp.global.tracing.enabled=true.
  This enables tracing for all the services running within the solution.

Configuring Jaeger to Trace Requests

1. Access the Jaeger service at the following location. Replace <DOMAIN > with the value that you have set in your environment during deployment, before running this command:
   ```
   http://jaeger.<DOMAIN>
   ```
2. Navigate to the Services drop-down (which can be found in the upper-left corner under Search) and select the service for which you want to view the instrumentation details.
   NOTE
   The services will be available in the drop-down only after REST API requests have been sent to the services.
   This is a sample image showing the "Service" drop-down in the Jaeger UI:



3. Click Find Traces.
   The results are displayed on the right-hand side of the application page.

Uninstalling Tracing Service

To uninstall Jaeger services from your environment, follow these steps:

1. Run this command to uninstall the Jaeger service:
   ```
   helm uninstall jaeger-operator -n tracing
   ```
2. Run this command to delete the tracing namespace:
   ```
   kubectl delete ns tracing
   ```

Understanding Resource Authorization Results using Explain

Resource Authorization results for Symantec Identity Security Platform Resource APIs (such as User, Group, applications, and so on) can often return complex outcomes. The `explain` parameter can provide detailed analysis and explanations of these outcomes.

Any IDSP Resource API that is subject to Resource Authorization supports the `explain` parameter. The `explain` parameter takes the following format:

```
{{sspHost}}/{{apiPathTenant}}/admin/v1/<ResourceType>?explain=true
```

For example:

- `{{sspHost}}/{{apiPathTenant}}/scim2/v1/Users?`**`explain=true`**
- `{{sspHost}}/{{apiPathTenant}}/scim2/v1/Groups?`**`explain=true`**

The response of a Resource API operation (POST/GET/PUT/PATCH) is a list of one or more resource elements. When `explain=true` is used, the first element in the response list is an `explain` element with the following format:

```
[
  {
     "explain": {
         // marker indicating this is the explain object (not a resource)
            "isExplainObject": "true",

         // final analysis of the constraint rules to be applied
              "adminScopeAnalysis": {},

          // the constraints the principal has
              "appliedConstraints": {},


          // which of those constraints apply to THIS request
              "constraintsApplicabilityDetermination" {},

          // the request context
              "request": {}
        },
        {
        // normal operation-specific resource elements
        }
]
```

Refer to the appropriate section of details about each element returned:

- Request Details
- Constraints Applicability Determination Details
- Applied Constraints Details
- Admin Scope Analysis Details
- Unconstrained Field Details

See also, Case-specific Exceptions to the Explain Results

Request Details

The `request` element contains the request details for the principal, the access token scopes, and the list of scoping constraints obtained from the set of Admin Roles assigned to the principal. This element provides the basis for the evaluation of the applied scope for the principal in the request.

The following sample shows an example request element:

```
"request": {
      "principal": {
  "sub_type": "USER",
      "sub": "f46d9d30-0058-4bb2-b9ed-5d12bfda0761",
      "user_loginid": "ljones"
  },
      "token": {
```

```
  "scope": "urn:iam:m.me urn:iam:t.session urn:iam:t.groups urn:iam:t.users",
      "scoping_constraints": "[
          GroupConstraint(5a74481a-f28d-4900-875c-4e02ab0384c5),
          UserConstraint(8529d589-c6d0-4262-a7cd-9b9b993702b2),
              Default Me Constraint(e51dd7ae-e8a5-4ada-a50d-cfcce23855e7)]"
  }
}
```

Constraints Applicability Determination Details

The `constraintsApplicabilityDetermination` shows the applicability of the constraints for each principal. The applicability determination is specific to the request type. For example, if the request is for an Application, a User, or a Group, the constraint would not be relevant. The `skipConstraint` field is true or false depending upon applicability. If `skipConstraint == true` a `SkipReason` field is provided.

The following example shows the constraint applicability when a User-based request has been made. The output shows that out of the three constraints that are applied to the principal, only the User-based constraint applies.

```
"constraintsApplicabilityDetermination": [
       {
                "id": "8529d589-c6d0-4262-a7cd-9b9b993702b2",
                "name": "UserConstraint",
                "resource": "urn:iam:res:user",
                "skipConstraint": false
},
    {
                "id": "5a74481a-f28d-4900-875c-4e02ab0384c5",
                "name": "GroupConstraint",
                "resource": "urn:iam:res:group",
                "skipConstraint": true,
                "skipReason": "Resource URI does not match request"
},
       {
                "id": "e51dd7ae-e8a5-4ada-a50d-cfcce23855e7",
                "name": "Default Me Constraint",
                "resource": "urn:iam:res:me",
                "skipConstraint": true,
                "skipReason": "Resource URI does not match request"
       }
]
```

Applied Constraints Details

Once it has been determined which of the constraints apply, further evaluation is made to determine which of the applicable constraint rules apply for the given resource request. The determination is shown in the `appliedConstraints` element. This element provides a summary section that shows the specific details of each applicable constraint. The applied Scope Rules section shows which of the rules are to be applied for this resource request.

The following User GET request example shows that the applied constraint has a rule that states the set of Users in scope are members of the Groups where the principal is a manager (from the instance `managedBy` rule). In this example, the principal is a manager of the *RADemo-Employees* group. This rule is the only applied scope rule for this User request.

```
"appliedConstraints": {
  "summary": {
      "constraints": "[UserConstraint(8529d589-c6d0-4262-a7cd-9b9b993702b2)]",
          "abacOps": "CURD",
          "gbacOps": "CURD",
          "disableOwner": false,
                  "instance": {
                      "managedBy": {
                                          "groups": "[RADemo-Employees]"
                      }
                   }
  },
  "appliedScopeRules": [
      {
          "constraint": "UserConstraint[8529d589-c6d0-4262-a7cd-9b9b993702b2]",
          "rule": "[ 'groups' eq 'urn:iam:adminscope:megroups' ]"
      }
  ]
}
```

Admin Scope Analysis Details

This `adminScopeAnalysis` element provides the final analysis of the rules to be applied to the request.

For resource searches (GET) the analysis contains the full set of queries to be applied to the search. In the example below, the principal (a user named *ljones*) is a manager of the *RADemo-Employees* group, so *ljones* has scope over any user who is a member of that group. Also, Resource Authorization implicitly allows access to any resource where the principal is an owner (created by the principal), so the combined query is the owner and group rule query.

```
"Admin Scope Analysis": {
  "enforcementQueries": {
      "ownerQuery": "createdBy eq ljones",
      "managedBy": {
          "groupsQuery": "groups eq RADemo-Employees"
      }
  },
  "finalSearchQuery": "ownerQuery or groupsQuery"
}
```

When the request is for an update or delete operation (POST/PUT/PATCH), the results are simplified. Rather than a query evaluation, the operation returns a determination of whether the resource is a match of the applied rule. For example, the following output is shown when a request is made to update a member of the *RADemo-Employees* group for which *ljones* is a manager:

```
"Admin Scope Analysis": {
  "groups": "match"
}
```

Unconstrained Field Details

Usually, the `"isExplainObject"` field is displayed for the `explain` object. However, there are cases where a Resource Authorization is *unconstrained*. There are multiple unconstrained conditions. If any of these conditions exist,

an `unconstrained` field is shown, along with the reason. No further explain content is provided when these conditions exist.

Condition	Field Value
`AdminScopeEnforced=false`	*Global AdminScope Enforcement is disabled*
Constraints are all inactive	*The principal doesn't have active scoping constraints*
Access Token has the scope `t.all` or `t.adminroles`	One of the following values: • *Principal has super privilege t.all* • *Principal has super privilege urn:aim:t.adminroles*
Scope is set to API scope enforcement only using the `adminScopeApi=true` setting in any of the constraints for the principal.	*The value of adminScopeApi is set as true in at least one scoping constraint*
The request is a User Self operation which is subject to administrative scope enforcement	*Self operation is allowed*

The following examples show two of the conditions and values:

```
"explain": {
      "isExplainObject": true,
  "unconstrained": "Global AdminScope Enforcement is disabled"
}

"explain": {
      "isExplainObject": true,
  "unconstrained": ""Principal has super privilege urn:iam:t.all"
}
```

Case-specific Exceptions to the Explain Results

In some use cases, the `explain` results differ slightly from the expected results.

The following examples show specific scenarios where the results of the `explain` parameter can differ from the results that are described in Understanding Resource Authorization Results using Explain:

- Delete
- Create
- Group Add/Remove Member
- ABAC/GBAC Ops (deny)
- Disable Owner

Delete

Resource DELETE provides just the status code of the operation. The operation does not provide response content or `explain` results.

Create

If attribute-based constraints are applied in create cases, the resources that are created must provide the appropriate value for these attributes.

In the following example, a User-based constraint is applied for the principal, with a rule that states *userType* = *employee*. If that value is not present in the create resource request, the operation fails. The Analysis section shows which attribute did not match the constraint.


```
"explain": {
      "isExplainObject": true,
      "Admin Scope Analysis": {
          "owner": "disabled",
          "attribute": {
              "userType": "notmatch"
          }
      },
      "appliedConstraints": {
          "summary": {
              "constraints": "[User Scope(8529d589-c6d0-4262-a7cd-9b9b993702b2),
                      HR Employee Policy(e8c80064-225b-47d5 9814-0b886de4ad0b)]",
              "abacOps": "CURD",
              "gbacOps": "URCD",
              "disableOwner": false,
              "instance": {
                  "attribute": "(userType eq employee)",
                  "managedBy": {
                      "groups": "[RADemo-Employees]"
                  }
              }
          },
          "appliedScopeRules": [
              {
                  "constraint": "HR Employee Policy[e8c80064-225b-47d5-9814-0b886de4ad0b]",
                  "rule": "[ 'userType' eq 'employee' ]"
              },
              {
                  "constraint": "User Scope[8529d589-c6d0-4262-a7cd-9b9b993702b2]",
                  "rule": "[ 'groups' eq 'urn:iam:adminscope:megroups' ]"
              }
          ]
      },
      "constraintsApplicabilityDetermination": [
          {
              "id": "8529d589-c6d0-4262-a7cd-9b9b993702b2",
              "name": "User Scope",
              "resource": "urn:iam:res:user",
              "skipConstraint": false,
              "gbacOps": "CD+RU"
          },
          {
              "id": "e8c80064-225b-47d5-9814-0b886de4ad0b",
              "name": "HR Employee Policy",
              "resource": "urn:iam:res:user",
              "skipConstraint": false
          },
          {
              "id": "5a74481a-f28d-4900-875c-4e02ab0384c5",
```

```
                "name": "Group Scope",
                "resource": "urn:iam:res:group",
                "skipConstraint": true,
                "skipReason": "Resource URI does not match request"
            },
            {
                "id": "e51dd7ae-e8a5-4ada-a50d-cfcce23855e7",
                "name": "Default Me Constraint",
                "resource": "urn:iam:res:me",
                "skipConstraint": true,
                "skipReason": "Resource URI does not match request"
            }
        ],
        "request": {
            "principal": {
                "sub_type": "USER",
                "sub": "f46d9d30-0058-4bb2-b9ed-5d12bfda0761",
                "user_loginid": "ljones"
            },
            "token": {
                "scope": "urn:iam:m.me urn:iam:t.session urn:iam:t.groups urn:iam:t.users",
                "scoping_constraints": "[Group Scope(5a74481a-f28d-4900-875c-4e02ab0384c5), User
Scope(8529d589-c6d0-4262-a7cd-9b9b993702b2), Default Me Constraint(e51dd7ae-e8a5-4ada-a50d-cfcce23855e7), HR
Employee Policy(e8c80064-225b-47d5-9814-0b886de4ad0b)]"
            }
        }
    }
```


Group Add/Remove Member

When a User is added or removed from a group, two conditions must be in place:

- The principal has to have scope over the Group (is a manager or matches a group attribute rule)
- The users to be added must be in scope for the principal. If an applicable user scope rule in not in place for the principal, then this second condition is not required.

In the following example, *ljones* is a manager of RADemo-Employees. Attempts to add the user *jholden* to the group fail because the user does not meet the applied scope rule `userType=employee` .


```
"explain": {
        "isExplainObject": true,
        "Admin Scope Analysis": {
            "managedBy": "match",
            "Analysis on Members": {
                "memberEnforcementQueries": {
                    "ownerQuery": "createdBy eq ljones",
                    "abacQuery": "userType eq employee",
                    "managedBy": {
                        "groupsQuery": "groups eq RADemo-Employees"
                    }
                },
                "memberEnforcementQueries": "notmatch"
```

```
            }

    },
    "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:Error"
    ],
    "detail": "Unprivileged request to add or remove the memberships of group RADemo-Employees with users
 [jholden] not in admin's scope",
    "status": "401",
    "errorCode": "89400006"
}
```

ABAC/GBAC Ops (deny)

A Scoping Constraint can define restrictions on allowed operations by using an operation qualifier. For example, when a '-' is used on any of the Create, Read, Update, or Delete ("C", "R", "U", "D") operators, the operation is denied.

As a further example, a constraint can define the following restriction to allow read and update operations for a User, but to disallow create and delete operation for the User:

```
"abacOpts": "-CRU-D"
```

When an applied constraint has this ABAC operations restriction in place for Users, a create user request fails with the following `explain` output:

```
"explain": {
        "isExplainObject": true,
        "Admin Scope Analysis": "Operation Denied. A constraint has an explicit CREATE deny setting",
        "appliedConstraints": {
            "summary": {
                "constraints": "[User Employee Policy(d5b099f4-67a7-432b-8f9e-17c15ddf3445)]",
                "abacOps": "U-C-DR",
                "gbacOps": "CURD",
                "disableOwner": false,
                "instance": {
                    "attribute": "(userType eq employee)"
                }
            },
            "appliedScopeRules": [
                {
                    "constraint": "User Employee Policy[d5b099f4-67a7-432b-8f9e-17c15ddf3445]",
                    "rule": "[ 'userType' eq 'employee' ]"
                }
            ]
        },
```

Disable Owner

By default, Resource operations implicitly include an owner rule. Owner rules are not defined in a scoping constraint but are always included when any constraint is applied. Owner rules can be disabled by defining the `"disableOwner": true` flag in a constraint.

The following `explain` output shows this condition. The `finalSearchQuery` element excludes the typical `ownerQuery` :

```
"Admin Scope Analysis": {
        "enforcementQueries": {
            "ownerQuery": "disabled",
        "abacQuery": "userType eq employee"
    },
    "finalSearchQuery": "abacQuery"
}
```

Configuring

Describes how to configure different features in the Symantec Identity Security Platform solution.

On this page, find links to various subpages that provide detailed instructions on configuring different components and features within your IDSP environment.

Click these individual links to know more about each topic:

- Configuring the IDSP Solution
- Configuring User Life Cycle Features
- Configuring the ID Store
- Configuring the Internal ID Store
- Configuring the Zero Footprint Identity
- Configuring the User Lockout
- Configuring the User Auto-Enrollment
- Configuring Inline Enrollment
- Configuring Trusted Devices
- Configuring SoftMFA
- Advanced Configurations
  - Managing Custom Vanity Ingresses
  - Using Custom CA for Intra Cluster SSL Communications
  - Operating with Multiple Data Centers

Configuring the Solution

Contains steps to configure the Solution.

After installing the Symantec Identity Security Platform, perform these steps to configure the solution.

This topic includes the following sections:

- About retrieving the Initial Security Credentials
- Granting Admin Privileges
- Updating the Authentication Policy for Admin Console (API) Usage

Retrieving Initial Credentials for the clients'

NOTE
The commands to retrieve the initial credentials use environment variables that are expected to be set in your environment. Before running those commands, ensure that these environment variables are set for your terminal session. You can do so by running the following command:

```
echo RELEASENAME=\'${RELEASENAME}\', NAMEPSPACE=\'${NAMESPACE}\' ; if [ -z "${RELEASENAME}" ] || [ -z
 "${NAMESPACE}" ]; then echo "***** ERROR: RELEASENAME or NAMESPACE environment variables are not set
 *****" ;fi
```

The following initial credentials can be retrieved within 48 hours from the time of deployment by using the following commands:

Client	Role (AZ Policy)	Tenant Name	Steps to acquire Client ID
InfraClient	Infra Admin	System	kubectl get secret ${RELEASENAME}-ssp-secre jsonpath="{.data.clientId}" | base64 --decode; ecl kubectl get secret ${RELEASENAME}-ssp-secre jsonpath="{.data.clientSecret}" | base64 --decode
SystemClient	System Admin	System	kubectl get secret ${RELEASENAME}-ssp-secre jsonpath="{.data.clientId}" | base64 --decode; ecl kubectl get secret ${RELEASENAME}-ssp-secre jsonpath="{.data.clientSecret}" | base64 --decode
DefaultTenantClient	Tenant Admin	Default	kubectl get secret ${RELEASENAME}-ssp-secre jsonpath="{.data.clientId}" | base64 --decode; ecl kubectl get secret ${RELEASENAME}-ssp-secre jsonpath="{.data.clientSecret}" | base64 --decode
DefaultTenantAdminUser You can use this user to login to the admin console and configure the default tenant. The validity of this admin user is limited to 48 hours by default, starting from the deployment time.	Tenant Admin User	Default	echo TENANT_ADMIN_USER=$(kubectl get sec defaulttenantadminuser -n ${NAMESPACE} -o js echo TENANT_ADMIN_USER_PASSWORD=$(secret-defaulttenantadminuser -n ${NAMESPACE decode)

NOTE
To know more about these bootstrap clients, see Administrative Model.

Invoke Token Endpoint

1. Invoke the Token Endpoint to get the Access Token.

```
POST https://{{IDSPHost}}/{{apiPathTenant}}/oauth2/v1/token
```

Example of Granting Admin Privileges

Perform this step as an administrator to grant admin role to a group or users in your ID store that will act as tenant administrators. To grant the admin role to a group or user, perform a "GET" of ".../admin/v1/AdminRolesDelegator" API to retrieve all the authorization policies in the solution, this helps you to find the policy ID of the Tenant Admin Authorization policy (TenantAdminPolicy) and perform a POST operation on it.

1. Perform a GET operation of the Authorization Policy using the following API:

```
GET https://{host}}/{{tenantName}}/admin/v1/AdminRolesDelegator
```

   The GET call response retrieves all the seeded IDSP's authorization policies. Find the "TenantAdminPolicy" in the list and note its corresponding ID. The following codeblock showcases the response specifically for the "TenantAdminPolicy."

```
{
        "id": "42a1faba-654a-4774-95af-5ee0331cd53f",
        "name": "TenantAdminPolicy",
        "displayName": "Tenant Administrator",
        "description": "Manage everything in a tenant",
        "principal": {
            "clientApp": {
                "operator": "in",
                "value": [
                    "baff7ce2-5dc9-4eec-b8f9-13a82b5ed93b"
```

```
                ]
            },
            "user": {
                "operator": "in",
                "value": [
                    "internal/DefaultTenantAdminUser"
                ]
            }
        },
        "isGrantableClientOnly": false
}
```

2. Use the POST operation to grant admin access to a new "*group*" principal condition in the TenantAdminPolicy. Replace the "<REPLACE-BY-POLICY-ID>" with the id of the TenantAdminPolicy id and update the ["<Group-Value>"]:

```
POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/AdminRolesDelegator
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/AdminRolesDelegator`
  should look similar to:
  `https://example.com/default/admin/v1/AdminRolesDelegator`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Request:

```
{
  "id": "<REPLACE-BY-POLICY-ID>",
  "principal": {
    "group": {
      "operator": "in",
      "value": ["<GROUP-VALUE>"]
    }
  }
}
```

Sample Request:

```
{
  "id": "42a1faba-654a-4774-95af-5ee0331cd53f",
  "principal": {
    "group": {
      "operator": "in",
      "value": ["admin"]
    }
  }
}
```

Response:

```
{
            "id": "42a1faba-654a-4774-95af-5ee0331cd53f",
            "name": "TenantAdminPolicy",
            "displayName": "Tenant Administrator",
```

```
        "description": "Manage everything in a tenant",
        "principal": {
            "group": {
                "operator": "in",
                "value": [
                    "Admin"
                ]
            },
            "clientApp": {
                "operator": "in",
                "value": [
                    "baff7ce2-5dc9-4eec-b8f9-13a82b5ed93b"
                ]
            },
            "user": {
                "operator": "in",
                "value": [
                    "internal/DefaultTenantAdminUser"
                ]
            }
        },
        "isGrantableClientOnly": false
    }
```

Example of Updating the Authentication Policy (AdminLoginPolicy) for Admin Console Usage

If you plan to use the Admin Console, you can perform any one of the following options:

- ***Using the default tenant*** - The initial deployment of IDSP creates a tenant named "default." This is primarily to facilitate the setup of the initial tenant. Within the default tenant creation process, a "defaulttenantclient" (assigned the tenant admin role) and a "DefaultTenantAdminUser" (part of the internal ID Store) with complete administrative privileges are provisioned by the seeded "TenantAdminPolicy" of Authorization policy (mentioned in the previouse step) and the user will be given authentication rights provisioned by the seeded "AdminLoginPolicy" Authentication Policy. You can use these credentials to access the admin console and configure the default tenant. For more details on the initial credentials seeded by IDSP deployment, see Initial Security Credentials.
  Example of the "AdminLoginPolicy" Authentication Policy where the "DefaultTenantAdminUser" is already provisioned:

```
{
    "policyId": "2067529c-f71e-44e5-b562-66f769e7fd80",
    "policyName": "AdminLoginPolicy",
    "description": "Default policy for Tenant Admin Console",
    "status": "active",
    "matchAnyApp": false,
    "apps": [
        {
            "name": "DefaultTenantClient",
            "id": "baff7ce2-5dc9-4eec-b8f9-13a82b5ed93b"
        },
        {
            "name": "AdminConsole",
            "id": "$TENANT_ADMINCONSOLE"
        }
    ],
    "rules": [
```

```
        {
            "id": "b59789db-bfe8-4fa9-add2-d18aac87f6c3",
            "conditions": {
                "principal": {
                    "user": {
                        "operator": "in",
                        "value": [
                            "internal/DefaultTenantAdminUser"
                        ]
                    }
                },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD:1"
                ],
                "msg": "This policy is for admin login"
            }
        }
    ]
```

- ***Creating your own tenant*** - If you opt for creating your own tenant, you need to configure the seeded "AdminLoginPolicy" Authentication policy with suitable rule conditions or create your own policy with appropriate rule conditions. See Managing Tenants .

Next Steps

After you complete performing the above steps, you can now use the DEMO environment to test the functionalities of the solution or use the REST APIs to perform the authentication flows.

Configuring the User Lifecycle Features

Know about User Life Cycle Management (LCM) and ID LCM WebHooks.

The User Lifecycle Management (Identity LCM) in Symantec Identity Security Platform allows notifications to be sent to an authoritative component regarding the state of an identity. This can be done through ***ID Store updates*** or ***ID LCM WebHooks***.

ID Store Updates

Configuration parameters, such as `isAuthoritativeForIDLCM` and `isAuthoritativeForUserCredsLCM` , control whether these updates are made in the Identity Store. These parameters are defined in the ID Store configuration.

- ***isAuthoritativeForIDLCM*** - If set to true, this parameter ensures that the user's lockout state in the IDSP is also updated in the Identity Store after a user lock.
- ***isAuthoritativeForUserCredsLCM*** - If set to true, this parameter ensures that the user credential registered in the IDSP during inline enrollment is also updated in the Identity Store.

Notifications include for the following User Credential LCM:

- User's lockout state
- Inline enrollment of user credentials (including email and phone number)

ID LCM WebHooks

SCIM-compliant webhooks can be set up to deliver these notifications to other authoritative systems.

The IDSP enables you to define a SCIM-compliant webhook indicating the events to be hooked and the SCIMconfigs resource to use as the target.

To configure the ID LCM WebHook, use the following API:

```
PATCH https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/Configs
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/Configs`
  should look similar to:
  `https://example.com/default/admin/v1/Configs`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

The ID LCM Web Hook is one of the properties under Service Configuration, called "idEventHookConfig".
Update the Global/Tenant parameter, "idEventHookConfig":

```
{
"name": "idEventHookConfig",
"value": "<SCIMconfigs-id-or-name>, event1, event2,..."
}
```

Where the "idEventHookConfig" parameter value contains:

- A value that starts with "#" implies a commented out / disabled configuration.
- A SCIMconfigs resource id or name, followed by one or more of the supported events:

  NOTE
  To know more about these auditing events, see Auditing in IDSP.

  The following events result in a SCIM request sent to the endpoint indicated by the SCIMconfig object:
  - ***factor.me.auth.lock.success*** - (Fires when a user gets locked out during authentication or password change due to exceeding max strike count) - sends a SCIM user update operation to lock the user (set user's "active" state to "false")
  - ***factor.me.auth.unlock.success*** - [Fires when a user gets unlocked from IDStore in authentication flow (requires IDStore, "isAuthoritativeForIDLCM=true" and "userAutoUnlockPeriodMinutes" defined)] - sends a SCIM user update operation to unlock the user (set user's "active" state to "true")
  - ***admin.me.user.lock.success*** - [Fires when a internal user (stored in IDSP internal ID Store) gets locked out during authentication or password change due to exceeding max strike count] - sends a SCIM user update operation to lock the user (set user's "active" state to "true")
  - ***admin.me.user.unlock.success*** - (Fires when a user gets unlocked during self-service password reset) - sends a SCIM user update operation to unlock the user (set user's "active" state to "true")
  - ***factor.me.register.complete.success*** - sends a SCIM user PATCH request whenever a new credential is created for a user (email or phone number).
  - ***internal.user.sync.initiate*** - sends a SCIM user PATCH request whenever a user in the internal ID store is created or updated. For more information, see Configuring User Change Event Hook.
- The following events result in a CAEP event sent to the endpoint indicated by the SCIMconfig object:

- ***session.revoke.success*** - sends a Continuous Access Evaluation Protocol (CAEP) message indicating that the user session is revoked, and includes a list of applications to be notified. For more information, see IDSP as CAEP Transmitter.
- ***identity.scim.user.account.disable.success*** - when an admin disables a user in the internal ID store when initiated using SCIM API.
- ***identity.scim.user.account.enable.success*** - when an admin enables a user in the internal ID store when initiated using SCIM API.
- ***identity.scim.user.account.lock.success*** - when an admin locks a user in the internal ID store when initiated using SCIM API.
- ***identity.scim.user.account.unlock.success*** - when an admin unlocks a user in the internal ID store when initiated using SCIM API.
- ***internal.cred.sync.initiate*** - when the passwords get updated for a user in the internal ID store when initiated using SCIM API.

NOTE
While the SCIMconfig resource is used as the identity store defining what ID LCM operations should be executed against the identity store, the same configuration is used to define a SCIM endpoint to deliver supported notifications. For example, in the case of the ZFP App where no ID Store is present, ID LCM notifications can be delivered using the ID LCM WebHook feature.

The appropriate ID LCM configuration depends on the deployment requirements and the authoritative identity source, which could be the ID Store or another system initiating necessary provisioning actions in response to updates initiated by IDSP.

The following table shows supported ID LCM options:

***"Yes" implies execute this Operation* Configuration Resource**	**Current ID Store**	
	LDAP	**SCIM**
ID Store LCM Operations • isAuthoritativeForIDLCM (true/false) • isAuthoritativeForUserCredsLCM (true/false)	Yes if the ID Store is authoritative for operation.	Yes if the ID Store is authoritative for operation.
ID WebHook Operations • factor.me.auth.lock.success • factor.me.auth.unlock.success • admin.me.user.unlock.success • factor.me.register.complete.success • internal.user.sync.initiate	Yes - if the webhook is configured	• Yes - if the webhook is configured. • No - – If ID Store is NOT authoritative – If the ID store is authoritative and is the same as the one configured in the ID webhook.

User Change Event Hook

Learn about the user change even hook.

The User Sync webhook triggers when an internal user is created or updated due to various actions.

The actions could include the following:

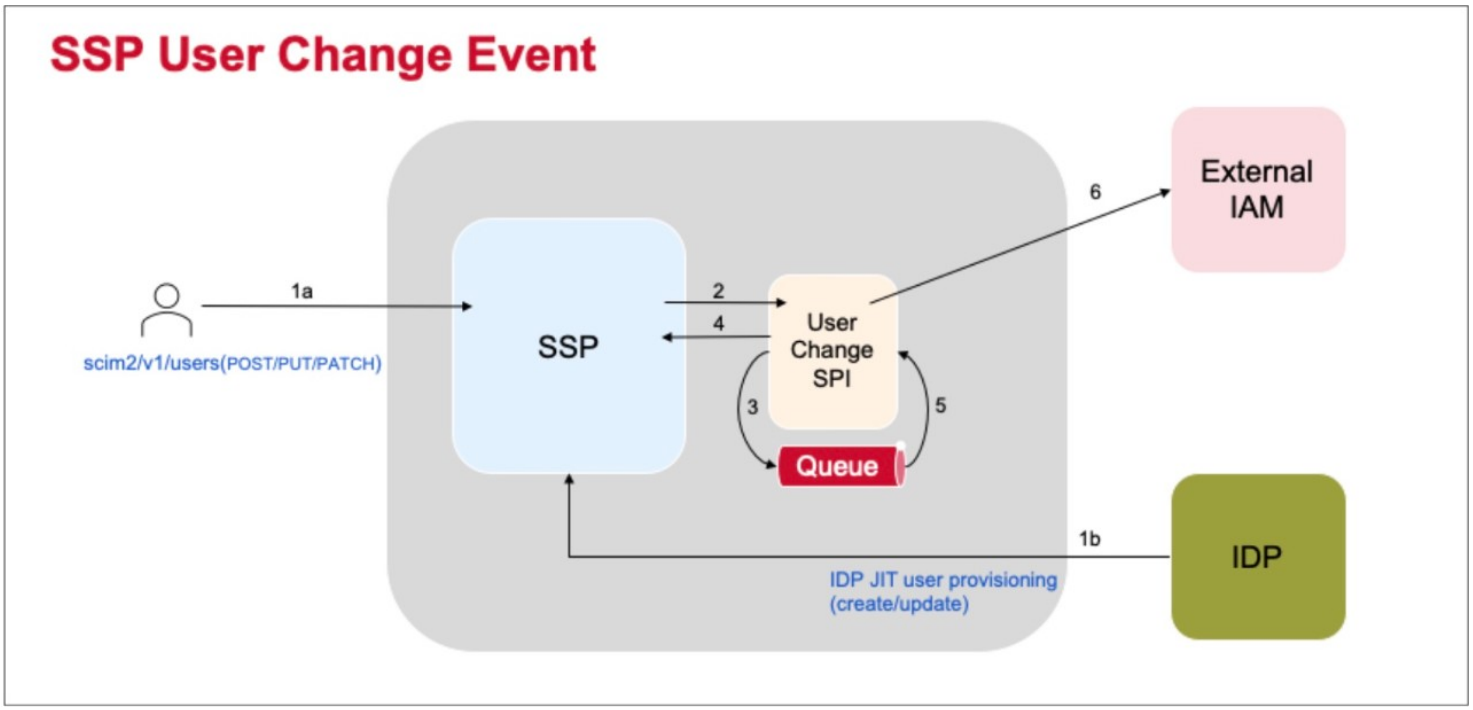
- federating to an external Identity Provider (IDP) configured for Just-In-Time (JIT) User Provisioning
- performing SCIM operations (POST, PUT, PATCH) on scim2/v1/Users/<userId-or-unique-loginid>

This webhook is represented by a SCIM Identity Store that is modeled in the Symantec Identity Security Platform, and mapped to the "internal.user.sync.initiate" event through an EventHook.

Upon firing, the webhook should process requests promptly by stashing them for asynchronous handling and returning a status code 204 without any response.

The SCIM Identity Store provides options for configuring connection and timeout settings, SSL validation settings, and client authentication security settings.

User profile attributes correspond to both native and custom logical user attributes, as shown on the Internal Identity Store configuration panel.



- Change events are triggered by either direct user changes through SCIM user API (1a) or JIT user provisioning during user authentication by the IDP (1b).
- Whenever a user change event occurs, the IDSP (Single Sign-On Platform) invokes the registered user change event hook Service Provider Interface (SPI) via webhook, sending a REST request containing a SCIM patch payload (2).
- The SPI must handle the request asynchronously, saving the information to a queue and responding with an HTTP 203 status code to acknowledge receipt. At this point, IDSP continues processing its request, reporting successful hook event delivery.
- The queue requests processor handles queue requests by creating or updating users in the external Identity and Access Management (IAM) system.

For simplicity, an in-memory queue can be used, with a separate thread managing the provisioning of users in the external IAM system.

Configuring Event Hook in IDSP

To configure the IDSP to invoke the User-Change SPI, follow these steps:

1. Create an ID Store object of type SCIM representing the "/user" SPI endpoint.
   a. Indicate the SCIM base (service hostname and base path) as scimURL.
   b. Configure proper authentication to the service.
   c. Configure SSL settings.
   d. Configure the attributeMappings property with the attributes that you must communicate to the external IAM system on each user change event.

e. The following example shows a sample of configuring a SCIM ID store:


```
{
    "configName": "SCIM Test",
    "configDescription": "",
    "securityMethod": "basic",
    "clientId": "test",
    "clientSecretAlias": "SCIM Test ID Store Password",
    "oauthResourceInAzRequest": "none",
    "scimURL": "https://example.com",
    "ignoreSslValidation": true,
    "sslMode": "verify_ca",
    "userLoginIdAttributeMappingName": "user_loginid",
    "groupNameAttributeMappingName": "name",
    "attributeMapping":
"user_loginid=userName,family_name=name.familyName,given_name=name.givenName,phone_number=phoneNumbers[primary]
    "groupAttributeMapping": "name=displayName,members=members.display",
    "isAuthoritativeForIDLCM": false,
    "isAuthoritativeForUserCredsLCM": false,
    "isAuthoritativeForDetectingLockedState": false,
    "statusEncoding": "scim",
    "accountRegisterURL": null,
    "privacyPolicyURL": null,
    "accountUnlockURL": null,
    "lookupQuery": "userName Eq \"${loginid}\"",
    "useUserLoginIdAsScimId": false,
    "useGroupNameAsScimId": false,
    "endpointSuffixOverride": {},
    "requestCtxBase64":
"W3sia2V5Ijoic2tpcHJlc3BvbnNlIiwidmFsdWUiOiJ0cnVlIiwidHlwZSI6Imh0dHAtaGVhZGVyIn1d",
    "readTimeoutMillis": "5000",
    "connectTimeoutMillis": "5000",
    "oauthTokenIgnoreSslValidation": true,
    "oauthTokenSslMode": "verify_ca",
    "oauthTokenReadTimeoutMillis": "10000",
    "oauthTokenConnectTimeoutMillis": "10000",
    "testConnectionURL": "https://example.com",
    "skewTimeSecs": "0",
    "idStoreMappingStrategy": "",
    "isPasswordAuthenticatorCapable": false,
    "createdDateTime": "2024-03-11 10:54 PM UTC",
    "updatedDateTime": "2024-03-12 06:42 AM UTC",
    "createdBy": {
        "principalType": "USER",
        "principalId": "4ed30eec-3c1f-4eeb-b4e8-cdb52c2ee01f",
        "principalName": "admin"
    },
    "updatedBy": {
        "principalType": "USER",
        "principalId": "4ed30eec-3c1f-4eeb-b4e8-cdb52c2ee01f",
        "principalName": "admin"
    },
    "scimConfigId": "97b89fa5-b674-4fa0-8133-57c2b7ba7ca2"
```

```
    }
```

2. Update Tenant Settings, ***idEventHookConfig*** to indicate the ID Store ID or the ID store name followed by a comma-separated list of events that should trigger a call to this ID Store. The user change event is "internal.user.sync.initiate". For example: Use the following API to update the configuration setting:

```
POST https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/Configs
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/Configs`
  should look similar to:
  `https://example.com/default/admin/v1/Configs`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

API Request:

```
{
"name": "idEventHookConfig",
"value": "97b89fa5-b674-4fa0-8133-57c2b7ba7ca2, internal.user.sync.initiate"
            }
```

This results in sending a scim patch call to this specified IDStore on every user update.

For more information about the various ID Store options, see Configuring the ID Store.

The following is an example of such an event-triggered scim patch call:

```
PATCH https://example.com/Users/2a5f7092-fb27-49d4-ad58-4384f424cf7d

{
    "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:PatchOp"
    ],
    "Operations": [
        {
            "op": "replace",
            "path": "emails[primary eq true].value",
            "value": "malbail@gmail.com"
        },
        {
            "op": "add",
            "path": "name.familyName"
        },
        {
            "op": "add",
            "path": "name.givenName"
        },
        {
            "op": "add",
            "path": "idp_first_name",
            "value": "Rohit"
        },
        {
```

```
            "op": "add",
            "path": "idp_last_name",
            "value": "Soni"
        },
        {
            "op": "add",
            "path": "is_federated_user",
            "value": "true"
        },
        {
            "op": "add",
            "path": "active",
            "value": false
        },
        {
            "op": "remove",
            "path": "phoneNumbers[primary eq true].value"
        },
        {
            "op": "add",
            "path": "userName",
            "value": "malbail@gmail.com"
        }
    ]
}
```

Configuring the ID Store

Contains information on ID Store overview, topology diagram, ID store attributes, and ID store APIs.

This topic includes the following sub-topics:

- ID Store Overview
- ID Store Capabilities
- ID Store Reserved User Attributes
- ID Store Reserved Group Attributes
- ID Store Other Attributes
- Configuring Tenant Settings
- ID Store Mapping in Password Authentication
- ID Store APIs
- ID Store Updates

ID Store Overview

The Identity Store represents an authoritative source of Identity for sourcing identity data that is used by the Symantec Identity Security Platform to manage authentication for non-Zero-Identity-Footprint applications. Access to such applications requires an identity to have an account in the Identity Store containing information that is used by IDSP to perform a user disambiguation, obtain, and update user's credentials such as email, phone number, validate user/group memberships, obtain/update disabled status.

IDSP supports any number of identity stores that can be accessed using LDAP or SCIM protocols.
Support for multiple identity stores introduces a number of concepts:

Identity Source: It represents an identity namespace such as LDAP or SCIM user directory, Federated Identity Provider, or a custom identity namespace used in ZFP-based token interactions.

- A ***User ID*** representing a user who undergoes the authentication flow in IDSP belongs to an identity namespace identified by the Id Store or Identity Provider ID. This results in a combination of identity namespace id and user login id, when taken together must be unique. Such uniqueness is enforced by IDSP's data schema constraints on the user account table in the database.
- The ***ID of the Identity Store and Identity Provider resources*** represents the identity source identifier. This identifier is used in ***policy bindings***, ***Id Store search order*** (tenant and application properties), and ***other use cases*** where the source of the identity must be maintained deterministically.

Tenant-level setting: This tenant-level setting helps maintain the uniqueness of user accounts and ensures that the correct identity is always used for operations. This is necessary to prevent users with the same login id across multiple identity stores from inadvertently being linked to the incorrect account in the IDSP account store.

- The "***identitySourceIdRequired***" tenant setting ensures tight compliance with the unique user requirement. There are few IDSP APIs that currently do not require use of the identity source id when invoking the API. However, this tenant setting mandates the use of such APIs to comply with the requirement to have a unique identity source id when invoking those APIs.
  - ***"identitySourceIdRequired"="false"(default)*** - implies user login ids are unique across identity namespaces. Therefore, use of a user login id without the identity source id is sufficient to locate the user account in the IDSP database. With multiple identity stores in action, user authentication may fail for a user with a login id that is not unique across identity stores, even if such login id is unique within a single identity store.
  - ***"identitySourceIdRequired"="true"*** implies user login ids are not unique across identity namespaces.
- "***principalConditionRequiredInPolicy***" tenant/global setting has been introduced as part of this feature to control whether an empty ***principal*** condition can be allowed in a policy.
  - ***"principalConditionRequiredInPolicy"="false"***- The existing policies and new policies without the principal condition continues to exist as-is.

    NOTE
    When "***principalConditionRequiredInPolicy***"="***false***", it is essential to have a principal condition ***ONLY*** if "***Admin Console***" is listed as one of the applications in the "apps" section within the authentication policy.
  - ***"principalConditionRequiredInPolicy"="true"(default for new deployments)*** - The POST and PUT calls of Authentication Policy API ("...admin/v1/AuthPolicies/{{policyId}}") should have the principal condition (user or group) specified in the new policies. See Settings and see Principal conditions to check for the "user" and "group" values supported.

    NOTE

    - The ***"principalConditionRequiredInPolicy"*** is set to "***false"*** for ***upgrade*** to allow existing policies that are created without principal conditions to exist as is, and allows editing the policies without specifying a principal condition. However, it is recommended to add principal conditions, as policies without the principal condition may not be fully operational when working with multiple identity stores.
    - When "***principalConditionRequiredInPolicy***" is set to"***false***", it is essential to have a principal condition ***ONLY*** if "***Admin Console***" is listed as one of the applications in the "apps" section within the authentication policy.

NOTE
"***idStoreToUse***" application setting should be set to ***"true"***. The "idStoreToUse" application setting accepts a comma-separated list of identity store IDs, whose order is significant. This enables the use of multiple identity stores that are implemented by LDAP or SCIM user directories. If this property is not configured, the system defaults to the "idStoreToBeUsed" setting in the tenant configuration.

ID Store Capabilities

The following table describes capabilities for each of the id store types supported.

Capability	LDAP ID Store	SCIM ID Store
Configuration Resource	`/LDAPconfigs` Only one resource can be configured per tenant representing the active identity store	`/SCIMconfigs` Only one resource can be configured per tenant representing the active identity store
Validate User Password	Yes	Yes, using a custom endpoint that is exposed by a SCIM service
Search User using a filter (during a user disambiguation)	Yes	Yes
Get/Update User Attributes	Yes	Yes
Get User Group Memberships	Yes	Yes
Update User Group Memberships	No	No
Lock/Unlock User Account (get & set a user locked status)	Yes	Yes
Get & Set User Credentials (phone number, email address)	Yes	Yes
Get Groups	Yes	Yes
Connection - Authentication	Bind DN & Password	Access Token via OAuth2 client id/secret Access Token via OAuth2 client assertion Basic Auth
Connection - SSL	Yes	Yes

ID Store - Reserved User Attributes

When working with the ID Store, the IDSP uses logical attributes that are mapped to physical id store schema attributes. The full set of attribute mappings is composed of mandatory attributes that are required to operate the solution, and optional attributes the solution must be aware of to support certain use cases.

The following is the list of reserved logical attributes that are used to operate the solution. Each of these logical attributes must be mapped to a physical schema attribute.

Reserved Attributes (Logical Attribute name)	Description	LDAP Attribute-Mapping value Example (AD/CA Dir)	SCIM Attribute Mapping value Example
user_loginid This logical attribute name is defined by the value of "userLoginIdAttributeMappingName". The mapping of the above logical attribute to the physical attribute is defined in the "attributeMapping" field.	This attribute represents the unique identifier of the user in the ID store. This unique identifier is used to associate the user with the IDSP account. This account stores additional information about the user, such as user credentials, device information. The physical attribute used for storing the user's user_loginid must be unique across the ID store and immutable (its value should never change). If you intend to allow the user login name (the username used on the login screen) to change, that can be achieved by using a disambiguation search query that uses a different physical attribute (not the one that is mapped to `user_loginid`). This disambiguation search query must be unique across the ID store, but it does not need to be immutable.	AD: sAMAccountName/ userPrincipalName CA Dir: cn	userName
name	End-User's full name in displayable form including all name parts, possibly including titles and suffixes, ordered according to the End-User's locale and preferences.	displayName	displayName
family_name	Surnames or last names of the End-User.	sn	name.familyName
given_name	Given names or first names of the End-User.	givenName	name.givenName
email	Preferred e-mail address for an OTP delivery. See the OTP section for more information.	mail	emails[primary]
phone_number	Preferred telephone number for an OTP delivery.	telephoneNumber	phoneNumbers.[mobile]
user_universalid	This attribute represents the Universal Id concept as used by SiteMinder.	<custom>	<custom>
status	Define the name of the attribute in the ID Store that contains the "Disabled Flag" value that is used by SiteMinder. See SiteMinder Documentation. This mapping is needed only when the statusEncoding is set to "siteminder" to indicate the same semantics for setting & discovering disabled status should be used.	<custom>	<custom>
memberof	This attribute is used to determine if 'memberOf' is enabled in the ID Store or not. • If the value is **True**, then the list of groups of the user is read from the user's attribute that is mapped to the "memberof" attribute. • If the value is **False**, then the group membership is determined by querying for the groups where the user's DN is listed in the group attribute that is defined by the **"groupUserMappingAttribute"** LDAPconfig/ SCIMconfig attribute.	memberOf	groups.display

ID Store - Reserved Group Attributes

The following is the list of reserved logical attributes that are used to operate the solution. Each of these logical attributes must be mapped to a physical schema attribute.

Reserved Attributes (Logical Attribute name)	Description	LDAP Attribute-Mapping value Example (AD/CA Dir)	SCIM Attribute-Mapping value Example
name This logical attribute name is defined by the value of "groupNameAttributeMappingName".	This attribute represents the unique identifier of the group in the ID store.	cn	displayName
members	This attribute represents the list of groups' members.	member	members.display

ID Store - Other Attributes Table

Using ID Store attributes in the context of OpenID Connect requires clients requesting identity data via "/UserInfo" request to have the necessary privilege. The following table describes the mapping between the client's OpenID Connect privileges as configured by the "allowedOpenIDScopes" property and OpenID Connect's attributes represented by logical attributes. Each of these logical attributes must be mapped to a physical schema attribute. Refer to Standard OpenID Connect spec for more details.

OAuth Scope	AH Logical Attributes
me	family_name, given_name, middle_name, zoneinfo, local
profile	name, family_name, given_name, middle_name, nickname, preferred_username, profile, picture, website, gender, birthdate, zoneinfo, locale, and updated_at.
email	email, email_verified
phone	phone_number, phone_number_verified
address	address
groups	groups

ID Store to be Used

There are two kinds of identity stores supported - LDAP-based and SCIM-based. Each tenant may have only one active Identity Store at a time. Each identity store is described by a configuration object, to configure this you need to set the idStoreToBeUsed tenant setting that controls whether a tenant uses the GUID of an LDAP configuration or the GUID of an SCIM configuration. For information, see Global/Tenant Settings. The default value is GUID of the configured ID store.

Use the ***PATCH https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/Configs*** API to update the ID store in IDSP. The following is an example of the tenant setting:

```
[
    {
        "name": "idStoreToBeUsed",
        "value": "cfbfe9d2-0749-487b-a5c0-c471d4f9516f"
    }
]
```

ID Store Mapping in Password Authentication

The ID Store property, "***idStoreMappingStrategy***" is required when the ID store where the user was disambiguated is different from the ID store where the user password authentication is carried out. This property should be configured on the ID Store where the user password authentication is performed.

The following table shows examples of the ID Store Strategy mapping value:

Format of the "idStoreMappingStrategy" value	Examples of a "idStoreMappingStrategy" value	Description
"userdn"	userdn	*"userdn"* is a keyword that indicates that the users in both ID stores have the same DN (find by dn).
"universalid"	universalid	*"universalid"* is a keyword that indicates that the users in both the ID stores have the same UniversalID value (find by the UniversalID field)
<disambiguated user value>	${idprofile.user_loginid} See Dynamic Environment Variable Substitution	Find the user using the lookup query (disambiguation query) that is configured in the target user store (using the value that is specified as the login id string)
=<disambiguated user value>	=${idprofile.user_loginId}	Search in the authentication ID store using the query, <user loginId attribute name>=<specified value> (find by user_loginid)
<local logical attr>=<disambiguated user value>	• email=${idProfile.email} • phone_number= ${idprofile.phone_number	Search for the user using the provided filter

ID Store APIs

The following table displays the list of ID Store APIs that can be used by an administrator to perform CRUD operations in the solution:

Operation Description	Operation Type	Endpoint URLs
Search for a user by using the user login id This operation performs disambiguation by applying ID store search order to locate the user by lookup filter configured on the id store.	GET	https://${SSP_FQDN}/{{tenantName}}/idstore/v1/Users/{{userLoginName}}?appid=<appid>&groups=<groups> • appId: <appid> • configid: <idstore-config-id> • groups: <true> or <false>
Search a user by filter This operation performs disambiguation by applying ID store search order to locate the user by the given filter.	GET	https://${SSP_FQDN}/{{tenantName}}/idstore/v1/Users?predicate=user_loginid&op=eq&value=admin&groups=true • appId: <appid> • configid: <idstore-config-id> • groups: <true> or <false> • predicate: user_loginid (any user logical attribute from id store attribute mapping) • op: eq (value = exact match), dis (apply search order to disambiguate using id store lookup query), dn (value = specific DN) • value: value to search for

Update a specific user detail in the ID Store	PATCH	https://${SSP_FQDN}/{{tenantName}}/idstore/v1/Users/{{userLoginName}}
		• configid: <idstore-config-id>
		• groups: <true> or <false>
		{
		"operations" : [
		
		{
		"action" : "REPLACE",
		"attrName": "email",
		"attrValues": ["example@gmail.com"]
		},
		
		{
		"action" : "REPLACE",
		"attrName": "phone_number",
		"attrValues": ["123-456-7980"]
		}
		
		]
		}

Update a specific user detail in the ID Store by using the filter	PATCH	https://${SSP_FQDN}/{{tenantName}}/idstore/v1/Users?predicate=user_loginid&op=eq&value=admin&groups=true • configid: <idstore-config-id> • groups: <true> or <false> • predicate: user_loginid (any user logical attribute from id store attribute mapping) • op: eq (value = exact match), dis (apply a search order to disambiguate using id store lookup query), dn (value = specific DN) • value: value to search for

Update a user's locked state by specifying the user login id	POST	https://${SSP_FQDN}/{{tenantName}}/idstore/v1/UserLocker/{{userLoginName}}
		• configid: <idstore-config-id>
		{
		"locked": true
		}

Retrieve a user's locked state by specifying the user login name	GET	https://${SSP_FQDN}/{{tenantName}}/idstore/v1/UserLocker/{{userLoginName}} • configid: <idstore-config-id>
Retrieve all ID Store groups	GET	https://${SSP_FQDN}/{{tenantName}}/idstore/v1/Groups • configid: <idstore-config-id> • members: <true> or <false>
Search a group by specifying the group name	GET	https://${SSP_FQDN}/{{tenantName}}/idstore/v1/Groups/{{groupName}} • configid: <idstore-config-id> • members: <true> or <false>

Search a group by using filter	GET	https://${SSP_FQDN}/{{tenantName}}/idstore/v1/Groups?predicate=<predicate>&op=<op>&value=<value> • configid: <idstore-config-id> • members: <true> or <false> • predicate: name (any group logical attribute) • op: eq (return one) or meq (return many) or dn (value=group DN) • value: * or x* or *x or x*y

Update a group by specifying the group name	PATCH	https://${SSP_FQDN}/{{tenantName}}/idstore/v1/Groups/{{groupName}}
		• configid: <idstore-config-id>
		• members: <true> or <false>
		{
		"operations" : [
		{
		"action" : "ADD",
		"attrName": "members",
		"attrValues": ["cn=nbruce,ou=Employees,o=democorp,c=us"]
		},
		{
		"action" : "REMOVE",
		"attrName": "members",
		"attrValues": ["cn=edunn,ou=HR,o=democorp,c=us"]
		}
		]
		}

Update a specific group by using filter	PATCH	https://${SSP_FQDN}/{{tenantName}}/idstore/v1/Groups/{{groupName}} • configid: <idstore-config-id> • members: <true> or <false> • predicate: name (any group logical attribute) • op: eq (return one) or dn (value=group DN) • value: * or x* or *x or x*y

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/idstore/v1/Groups`
  should look similar to:
  `https://example.com/default/idstore/v1/Groups`
- To know more about the scopes that are required to secure these APIs, refer to the "Scopes" section here.

ID Store Updates

The Identity Store, when in non-ZFP mode, is always authoritative for determining the user's initial lockout state and, if so configured, the user's initial credentials to be enrolled. In some cases, the Identity Store must be updated when such state

changes in the IDSP. Such behavior can be configured in IDSP by specifying appropriate feature flags on the Identity Store configuration resource:

- ***isAuthoritativeForIDLCM*** - If set to true, this parameter ensures that the user's lockout state in the IDSP is also updated in the Identity Store after a user lock.
- ***isAuthoritativeForUserCredsLCM*** - If set to true, this parameter ensures that the user credential registered in the IDSP during inline enrollment is also updated in the Identity Store.

Links to LDAP and SCIM Configurations:

- To know more about LDAP Configuration, see LDAP Configuration
- To know more about SCIM Configuration, see SCIM Configuration

Link to ID Store Search:

- To know more about ID Store Search, see ID Store Search

Configuring the Internal ID Store

For non-Zero-Identity-Footprint applications, in addition to support for LDAP or SCIM ID Stores, Symantec Identity Security Platform supports an Internal ID Store.

The Internal ID Store is part of the IDSP ID Store architecture. The Internal ID Store supports all the same concepts and capabilities as the current LDAP or SCIM user directories in addition to the Identity Service capabilities described. The internal ID store is automatically deployed and ready for use in IDSP.

The IDSP Admin Console > Identity Stores page lists the internal ID Store by default.

> **NOTE**
> For more information, see Configuring an Identity Store using the Admin Console in the Internal Identity Store topic.

You can configure the internal ID store through the Admin Console or through the Identity Store APIs. To configure the internal ID store through the Admin Console, see Internal Identity Store Configuration.

Internal ID Store Configuration Properties

The Internal ID store configuration properties appear in the following table.

Property Name	Description	Property Value
`accountActivationRequired`	When creating SCIM users, the account must be activated. If set to false, the account is automatically activated at creation.	false – an account is automatically activated at creation true - the account must be activated (default)
`accountActivationTokenExpiryDays`	The number of days within which the account must be activated before the token that is created by the activation request expires.	
`accountRegisterUrl`	An optional value that, when defined, provides a **Register Account** link on the **Sign-in App** login page.	This value is empty by default.

accountRegisterUrl	An optional value that, when defined, provides a **Register Account** link on the **Sign-in App** login page.	This value is empty by default.
activationNotificationDestinations	Identifies which credentials (Email or phoneNumber) are returned by the Verified Credentials API for the user, which is then used to deliver the activation token (see Identity Self-Service Operations API.). **Writer:** Change xref to new self service account activation topic. This flag is used to determine where the activation token is delivered.	Comma-separated list of email addresses or SMS-enabled phone numbers. The value is "email, sms" by default.
adminCanGeneratePassword	This setting determines if an Administrator can change a user password.	true/false. The default is false. If you change the value, you must issue new access tokens to reflect the permission change.
allowAdminSelfModifyOperation	By default, administrators cannot modify their own profile information. If set to true, administrators can modify their own profile information	false – do not allow administrators to modify their own profile information (default) true - allow administrators to modify their own profile information
allowSelfServiceAccountActivation	Allow a user to activate an account the first time that the user authenticates to the account. See Self Service Account Activation in the Identity Service	false – do not allow self-service account activation (default) true - allow self-service activation
allowStartIndexSearchIndex	Start Index is used in the SCIM pagination model to indicate the next page set to retrieve. With large query results, specifying a high start Index value (for example, page number 10k out of a 20K user query) can have a significant performance effect. Start index is turned off by default, both as a precaution and to bring awareness of the impact of its use.	true/false. The default is false
allowLocalPasswordForFederatedUsers	Allows federated users to have a local password.	true/false. The default is false.
disambiguationQuery	The query is used to disambiguate the user. **Note**: Make sure to consider minimal latency when defining this query.	`user_loginid eq ${loginid} or email eq ${loginid}`
displayNameTemplate	Defines how the user display name is generated from the user attribute values.	`"${idprofile.given_name} ${idprofile.family_name}"`

emailDomainForSimulatedAuthenticationFailure	When idle authentication failure is true and a user requests to reset a password, the Forgot Password flow presents a randomly generated and obfuscated email address or phone number to the user. Use this property to define the email domains that appear on this page for the user. Set whether the **Forgot Password** page presents an email address or phone number in the selfServiceAccountActivationStrategy property.	Comma-separated list of email domains. The default value is **null**.
emailsMustBeUnique	Determines if user email values must be unique across the tenant.	true/false. The default is true.
idStoreMappingStrategy	This property is used to determine how to look up the user for Password Authentication.	user_loginid=${idprofile.user_loginid}
loginIdRegexEnabled	This regular expression formats the user login ID to match the corresponding attribute value in the identity store.	true/false. The default is false.
loginIdRegexFrom	A regular expression to be applied to the user login ID.	
loginIdRegexTo	A regular expression that is used to format the user login ID to match the corresponding attribute value in the identity store.	***Example***: The "**loginIdRegexFrom**" and "**loginIdRegexTo**" parameter has been explained through the following example. In this example, when the following values are considered, • loginIdRegexFrom = ^(.*)\s(.*)$ • loginIdRegexTo = 0{2} • Input value = +32 495551320 The regular expression transforms the input value +32 495551320 into 0495551320 (result) by capturing the part after the space and adding 0 in front of it.
passwordHashAlgo	The Admin Console password is encrypted using the SHA-256 encryption algorithm. Select how to apply the SHA-256 algorithm.	• **SHA256**: the password is encrypted using one iteration with the SHA256 algorithm. • **SHA256_600**: the password is encrypted using 600,000 iterations with the SHA256 algorithm. The SHA256-600 option provides an additional level of security. The higher iteration count significantly increases the time that is required to compute the hash, making it more difficult for brute-force attacks to succeed.
privacyPolicyUrl	An optional value that, when defined, provides a **View Privacy Policy** link on the **Sign-in App** login page.	This value is empty by default.

scimBulkMaxOperations	The maximum number of SCIM operations that can be handled by the SCIM bulk API.	Default: 1000
scimBulkMaxPayloadSize	The maximum size of the payload request for the SCIM bulk API.	Default: 1 MB
scimSearchLimit	The search limit for SCIM User and Group search requests.	Default: 500
searchTimeoutSecs	The number of seconds after which the search fails if no values are returned.	Default: 0 Maximum: 1000
selfServiceAccountRecoveryCredentialStrategy	Identifies which credentials (Email or phoneNumber) are returned by the Verified Credentials API for the user (see User Self Service API and Identity Self-Service Operations API.). These credentials can be used in subsequent API calls to deliver the token for Password Reset or Account Unlock operations.	Comma-separated list of email addresses or SMS-enabled phone numbers. The value is "email, sms" by default
skipEmailValidation	When creating users, email addresses are validated for correctness. If set to false email addresses are not validated.	**false** – do not validate email addresses during user creation **true** - validate email addresses during user creation (default)
skipPhoneValidation	When creating users, phone numbers are validated for correctness. If set to false phone numbers are not validated.	false – do not validate phone numbers during user creation true - validate phone numbers during user creation (default)
strictValidationEnabled	Controls whether to adhere to Strict SCIM protocol in SCIM calls.	false – do not adhere to Strict SCIM protocol (default) true - adhere to Strict SCIM protocol
unlockAccountAfterPasswordReset	Support for Self-Service Account unlock. When initiating Account unlock a Password Reset flow is invoked, which, upon completion unlocks the user account.	false – Self-service account unlock is disabled (default) true – Self-service unlock using Password Reset is enabled
useMaskedCredentialValues	When a user profile email or phone number is used as a credential, a value of true for `displayName` indicates that the email or phone number contains a masked value and that value should be used when displaying the credential.	False – let the service provide the masked email and phone value to be displayed. True – assumes the persisted email/phone.displayName field on the user profile contains the masked value to be displayed

usePriorityToDetermineUserEmail	Some User multi-value user attributes support the designation of a primary value (for example, email.primary). In the absence of a primary designation, you can use a priority value on the attribute. The lowest priority value is deemed primary.	false – do not use priority for designating a primary (default) true - use priority for designating a primary
userNameGenerationRule	The rule that determines the structure of system-generated usernames. The username generation rule can be formed with the combination of one or more string-literals, place-holder-for-number-of-digits, and variables which can be separated by '+' character. See Username Generation Rules	"${idprofile.given_name} + '.' + ${idprofile.family_name}".
validatePolicyPrincipals	When saving an authorization or authentication policy, we validate the existence of the principal of the policy.	false – do not validate that the principal exists when saving an authentication or authorization policy (default) true - validate that the principal exists when saving an authentication or authorization policy
customUserAttributes	Defines the set of User custom attribute definitions.	See the following User Schema section for details.

Internal ID Store Configuration APIs

The following table displays the list of Internal ID Store Configuration APIs that an administrator can use to get the configuration properties for the store.

Operation Description	Operation Type	Endpoint URLs
Retrieve the current configuration	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/InternalIdStoreConfig

Update the current configuration	PUT	https://${SSP_FQDN}/{{tenantName}}/admin/v1/InternalIdStoreConfig
		
		{
		"idStoreMappingStrategy": "user_loginid=${idprofile.user_loginid}",
		"disambiguationQuery": "user_loginid eq ${loginid} or email eq ${loginid}",
		"unlockAccountAfterPasswordReset": false,
		"useMaskedCredentialValues": false,
		"passwordResetInitiatorUrl": null,
		"passwordResetChangerUrl": null,
		"accountUnlockInitiatorUrl": null,
		"customUserAttributes": null
		}

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring the Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values in all the endpoints that are shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/InternalIdStoreConfig`
  should look similar to:
  `https://example.com/default/admin/v1/InternalIdStoreConfig`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Configuring the Zero Footprint Identity

Learn about configuring Zero Footprint (ZFP) in Symantec Identity Security Platform.

This topic includes the following sections:

- About Zero Footprint (ZFP)
- Use Case: Using ZFP with IDSP and SiteMinder
- ID Token Hint Details
- ZFP Configuration in IDSP
  - Configuring Application Entity
  - Settings Configuration
- Display Name Generation in ZFP

About Zero Footprint (ZFP)

In some scenarios, it may be necessary to use ZFP instead of relying on direct communication routes to identity stores. For example, a deployment where the IDSP (solution) does not have direct access to a user store or where such a user store requires an expensive process of buildout or provisioning, the ZFP approach provides the solution with a simple way to extend its services.

Following are the benefits of ZFP, when applicable:

- Simplifies integration by not requiring ID Store connection between IDSP and user directories
- Minimizes dependency on having multiple external identity stores
- Maximizes resiliency by not requiring a physical connection

Zero Footprint (ZFP) - Best Practices

Table 80: ZFP Best Practices Table

ZFP Benefits (When to use ZFP)	ID Store Benefits (When not to use ZFP)
This approach becomes the only viable option when there is no connection to the ID Store.	When the user has a substantial number of groups and profile attributes that are essential for Authentication and Access policies requiring clients to issue Identity Token with that information, causing tokens to exceed header size limits supported by infrastructure.
ZFP provides greater resilience in situations where the connection to the ID Store is unreliable, offering more fault tolerance.	In cases where specific applications require app-specific tokens, authentication, or access policies that rely on real-time access to user profile attributes and group memberships.
By eliminating at least one trip to the ID Store, it improves performance, particularly in SiteMinder Web Flows that delegate authentication to IDSP.	ID store does not rely on the requesting client or a third-party Identity Provider to provide user profile and group membership data.
ZFP reduces dependency on the ID Store when not essential, making the system more robust.	When multiple clients are requesting IDSP services, and these clients do not share a common view of the user identity, this approach can be used.

Use Case: Using ZFP with IDSP and SiteMinder

ZFP enables SiteMinder to propagate the primary authentication context (for example- establish through Username and Password) to IDSP, which in turn evaluates MFA policy for additional step-up requirements.

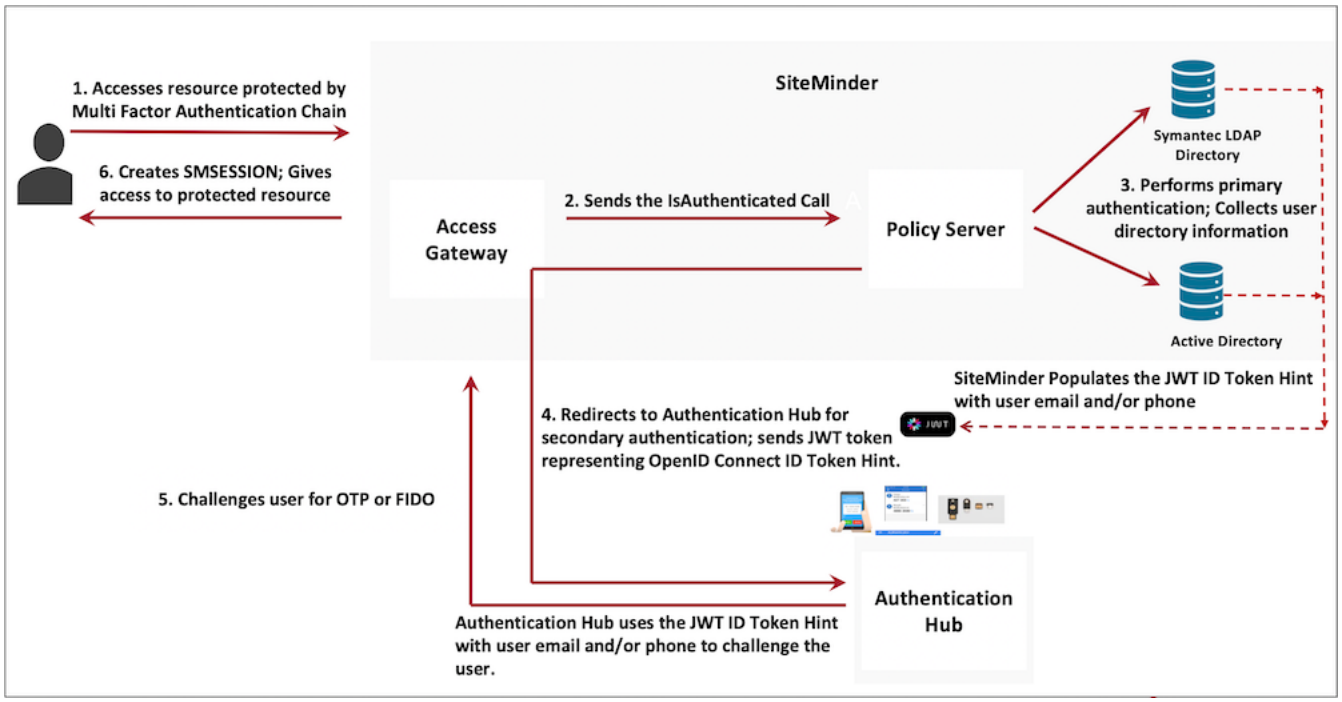
The SiteMinder integration with IDSP supports various use cases to support identity propagation. For the complete list, see the Supported Use Cases section in Identity Propagation in SiteMinder.

The following use case describes how IDSP uses an authentication context from SiteMinder when a user accesses a resource and IDSP is required to perform additional authentication depending on the user identity namespace represented by the SiteMinder User Directory.

Assumptions:

- SiteMinder has the following two user directories configured:
  - User Directory 1 - Symantec LDAP Directory
  - User Directory 2 - Active Directory
- SiteMinder has two user directories that are configured, Dir1-(CA Directory), and Dir2-(Active Directory)
- A Multi-Factor Authentication Chain authentication scheme is configured in SiteMinder with the primary authentication through the HTML Form authentication scheme, and secondary authentication being delegated to IDSP.
- In IDSP, an authentication policy defines the use of PWD + OTP for Symantec Directory users and PWD + FIDO for Active Directory users.

The following high-level diagram describes the process flow when the use case is executed:



The following process describes the execution flow of the use case:

- A user without an existing session accesses a resource that is protected by the Multi-Factor Authentication Chain authentication scheme.
- Access Gateway intercepts the request and forwards it to Policy Server.
- Policy Server authenticates the user using HTML Form authentication and creates ID Token Hint with a user ID and user directory name.
- Policy Server redirects the user to IDSP through Access Gateway for secondary authentication.
- IDSP performs the following steps:
  - Evaluates the ID Token Hint to determine how the user must be further challenged.
  - The user is not challenged PWD as SiteMinder has already authenticated the user for PWD.
  - IDSP evaluates the claim in the received Identity Token Hint to decide if the specified user email or phone number that is conveyed by the token should be auto-enrolled as authentication credentials for MFA purposes.
  - If the user belongs to Symantec LDAP Directory, IDSP challenges the user for OTP. If the user belongs to Active Directory, IDSP challenges the user for FIDO.
  - IDSP returns ID Token with its user authentication information to SiteMinder.
- SiteMinder creates an SMSESSION cookie for the user and grants the user access to the requested protected resource.

ID Token Hint Details

When SiteMinder (requesting client) integrates with IDSP and enables the "zeroFootPrint" property to true, (example-"zeroFootPrint": true) then the solution does not validate the user identity against the local LDAP directory, rather uses the ID Token Hint submitted by the requesting client through the (/auth/v1/authenticate) REST API or OpenID Connect "login_hint" request parameter (/oauth/v1/authorize). The client is assumed to have the authority to issue such hint to represent the user first factor, and must be declared as "Trusted" in VIP IDSP.

The ID Token Hint passed by the requesting client through the REST API (/authenticate) or OpenID Connect /Authorize grant, must include the following mandatory claims representing the Identity:

- email: The email ID configured in the requesting client end.
- phone_number: The phone number of the user who is configured in the client.
- user_universalID: The login ID of the requesting client.
- auto_inline_enrollment:
  - When this claim is set to true, the solution auto enrolls the credentials based on the value in the ID Token Hint
  - When this claim is set to false, the solution ignores the credentials that are sent in the ID Token Hint

NOTE

- If the ID Token Hint (JWT token) did not include the email and phone number of the requesting client, then inline enrollment of the client phone and email is allowed without challenging for a step-up. However, for inline enrollment to work, the JWT should contain "auto-enrolment" in ID Token Hint.
- If the ID Token Hint did include the email and phone number of the client, and auto-enrolment" in ID Token Hint is set to 'true', then client credentials are enrolled automatically in the solution.

The following table shows the ID Token Hint Claims as expected by IDSP and the attributes that are passed in the ID Token Hint:

Attribute	Description
Regular JWT Attributes as per IT spec	
user_loginId	User loginId used to identify the user. This value must uniquely identify the user and must be immutable.
aud	Client ID of Requesting Client (for example:SiteMinder) application in AuthenticationHub
auth_time	Time when the user was authenticated by the Requesting Client (SiteMinder)
iss	Issuer by which Requesting Client (SiteMinder) is identified
exp	Expiry time of the Identity Token Hint
nonce	nonce
iat	Time when the Identity Token Hint was issued
jti	Unique *identifier* for the JWT
nbf	Time when the user was authenticated by the Requesting Client (SiteMinder)
Required for user lookup	
email	Based on Requesting Client's (SiteMinder) User Directory (UD) attribute
phone_number	Based on Requesting Client's (SiteMinder) Authentication Scheme config. Leave the value empty, if it is not mapped to the UD attribute
user_universalID	Based on Requesting Client's (SiteMinder) UD attribute config
auto_inline_enrollment	Indicates if the credentials sent in the ID Token should be automatically enrolled (Based on Requesting Client's (SiteMinder) config. When this field is set to false then IDSP, ignores the creds that are sent in IT When this field is set to true then IDSP automatically enrolls the creds in the solution based on the values in the ID Token Hint.
Required for policy eval	

idp_name	A logical name representing the user identity namespace Note: In case of SiteMinder integrating with IDSP using ZFP,`idp_name` is the name of the User Directory where the user identity exists. SiteMinder's "IDSP" Auth Scheme sets this claim automatically.
Required for identity propagation	
amr	Based on the list of AMRs supported by IDSP. Note: In case of SiteMinder integrating with IDSP using ZFP,`amr` is the AMR value that is associated with the authentication method that is applied by SiteMinder during user authentication prior to requesting MFA from IDSP. SiteMinder's "IDSP" Auth Scheme sets this claim automatically.

ID Token Hint Sample

The following sample illustrates an ID Token Hint sent by the requesting client:

```
{
"sub": "nbruce",
"auto_inline_enrollment": true,
"idp_name": "Employees",
"user_universalid": "nbruce",
"amr": [
"PWD"
],
"user_loginid": "nbruce",
"iss": "https://security.com/iam/example",
"nonce": "382443df-9734f4d9-21553cf7-08425087-941d212a-cc",
"aud": "e4df1123-5579-474e-bbae-a23518d2e3a3",
"nbf": 1612428071,
"auth_time": 1612428071,
"phone_number": "+9190012530000",
"exp": 1612428371,
"iat": 1612428071,
"email": "nbruce@example.com",
"jti": "382443df-9734f4d9-21553cf7-08425087-941d212a-cc"
}
```

ZFP Configuration in IDSP

Staying with the SiteMinder example, upon successfully authenticating an identity using username/ pwd, SiteMinder sends the ID Token Hint with user ID, name of the user directory and other details to IDSP for secondary authentication. The ID token Hint contains the attribute "amr" that represents a factor that has already been applied by SiteMinder. Therefore, IDSP evaluates the Identity token hint to determine the next obligations that the user should be challenged with. However, the obligation that is conveyed in 'amr' is skipped, since this factor has already been used to authenticate the user in SiteMinder. Thus, the user is challenged with the remaining factors.

To know about configuring "IDSP" Authentication Scheme in SiteMinder, refer to SiteMinder-IDSP integration doc.

Configuring Application Entity

In IDSP, the Zero Footprint property is added to the "Application" configuration as a boolean value (default is set to "false"). To make use of this property, set `"zeroFootPrint": true` . The sample code shows the "zeroFootPrint" property configuration in the "Application" API:

```
{
   "status": "active",
```

```
    "name": "democlient",
    "description": "Demo client to perform tenant level operations for demo purpose.",
    "clientType": "TRUSTED",
    "allowedOpenIDScopes": [
        "openid",
        "profile",
        "email"
    ],
    "redirectURIs": [
        "https://testapp.vlsampleapp.bm.com/oauth2/callback"
    ],
    "allowedGrantTypes": [
        "client_credentials",
        "authorization_code",
        "urn:ietf:params:oauth:grant-type:jwt-bearer"
    ],
    "userInfoEndpointResponseFormat": "PLAIN_JSON",
    "skipIssuerAudienceForIT": false,
    "skipEmailForIT": false,
    "zeroFootPrint": false,
    "allowedOperations": [
        "introspect"
    ],
    "enableTrustedDevice": false,
    "appId": "df11af0f-e872-44c1-bafe-1fc310734e96",
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "7073a00c-cbad-4fa3-a58e-a3f02dc00793",
        "principalName": "defaulttenantclient"
    },
    "createdDateTime": "2020-10-27 23:27 PM UTC",
    "updatedDateTime": "2020-11-10 23:08 PM UTC",
    "secret": "0524011e-4172-4e96-abaa-2a30e4521792",
    "clientId": "3cb1361e-2f22-4d0e-b679-693db7e3c1ae"
}
```


See Managing Applications

Settings Configuration

(Optional) To use the User Auto-Enrollment feature, enable the "isAutoEnrollmentAllowed" setting in IDSP. The settings description is detailed in Settings. To know about User Auto-Enrollment, see here.

Configuring Authentication Policy

The IDSP Authentication Policy includes "idpName" condition for "zeroFootPrint". This condition indicates the name of the logical identity namespace in a policy condition. Optionally, add this condition to the policy by using a POST/PATCH operation.

For example: With SiteMinder using ZFP to integrate with IDSP, the identity namespace is represented by the name of the SiteMinder User Directory containing the user.


```
{
     "policyName": "Feature Policy Demo - siteminder integration",
     "description": "Feature Policy Demo - siteminder integration",
     "status": "active",
```

```
    "matchAnyApp": false,
    "apps": [
        {
            "name": "siteminder",
            "id": "7f27b5cd-5e22-43b7-a7a5-942472581ffe"
        }
    ],
    "rules": [
        {
            "conditions": {
                    "idpName": {
                           "operator": "in",
                           "value": ["Employees"]
                       },
                "acr": {
                    "operator": "in",
                    "value": [
                        "test"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD:1"
                ],
                "msg": "Rule1: Password auth for users in 3rd party directory"
            }
        },
        {
            "conditions": {
                "idpName": {
                    "operator": "in",
                    "value": ["Consumers"]
                },
                "acr": {
                    "operator": "in",
                    "value": ["test"]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD:1", "OTP:2", "MOBILEOTP:2"
                ],
                "msg": "Rule1: Password auth for users in LDAP store"
            }
        }
    ]
}
```

Display Name Generation in ZFP

When a ZFP account is created, Symantec Identity Security Platform ensures that the display name of the account is always set. Symantec Identity Security Platform generates the `displayName` value from one of the following sources (in order):

1. metadata of the identity provider; or,
2. metadata of the application; or,
3. `displayNameTemplate` configured in the internal identity store configuration (see Internal Identity Store Configuration); or,
4. `givenName` and `familyName` ; or,
5. user `loginId` .

Configuring the User Lockout

A setting that locks a user after the maximum number of failed login attempts has been reached.

This topic includes the following sections:

- User Account Lockout Behavior
- Updating the ID Store Config Request
- Using the User Locker APIs
- Resetting the Locked User Account
- User Lock Behavior Scenarios and Error Messages Table
- Lock User On Next Auth Failure Warning in the authentication API response

User Account Lockout Behavior

When a user attempts to log into a protected resource several times without success, the individual may be suspected of being fraudulent or malicious, resulting in a security vulnerability. As a result, a feature known as User Locker has been added to the solution. This feature prevents a user from performing unsuccessful logins beyond a certain number of attempts and locks the user in the authoritative source. There are a few tenant-level settings added to customize this option in the solution. These settings control the number of times a user can try unsuccessfully to log in before being locked out by the administrator for a predetermined length of time. The admin can manually choose to unlock the user by invoking the UserLocker API.

The required settings to be configured for user account lockout are explained in the following section.

Configuring the settings

The following table lists the tenant-level settings that should be configured in the solution:

Property Name	Property Scope	Property Description	Default Value
userMaxStrikeCount	Tenant	This property allows administrators to specify how many failed authentication attempts are allowed before the user is locked out. • If this value is set to '0', then this property will not be updated and the user will not be locked out. • If this value is set to more than '0', then the user will get locked in the authoritative source after this value is reached. The authoritative source is defined by the attribute "isAuthoritativeForIDLCM" in the ID Store configuration.	0

userAutoUnlockPeriodMinutes	Tenant	This property allows administrators to specify the number of minutes a locked-out account should remain locked before it is automatically unlocked. The locked user will be unlocked after this time period expires. • The default value configured for this setting is "0" minutes, which implies that autoUnlock functionality is disabled. Any positive value for this setting indicates that when the specified number of minutes have elapsed, autoUnlock will be enabled. • If the ID Store attribute,"isAuthoritativeForIDLCM" is set to "true", the "userAutoUnlockPeriodMinutes" property unlocks the user in the ID Store as well.	0
trackLastLogin	Tenant	This property allows administrators to track whether the last authentication was successful or unsuccessful.	false

These configuration settings are also covered in Global/Tenant Settings.

Updating the ID Store Config Request

The following attributes should be configured in the ID Store configuration request payload for UserLocker behavior:

- ***statusEncoding*** - This attribute checks the ID Store type being used. It helps to understand how the status attribute is encoded.
  If this value is not specified, then the ID Store service falls back to the backward compatibility mode where the status field is used with AD semantics.
- ***isAuthoritativeForIDLCM*** - This attribute checks if the ID Store that is being configured, is also the authoritative source for user locking. This is a boolean string with a value of true/false. If this value is set to "false", then Symantec Identity Security Platform becomes the authoritative locking source and the ID Store that has been configured will not be the locking source. The user locking is done when the user strike count reaches the maximum allowed value. If this value is set to "true", the user lock is done in the user store.

The descriptions for these attributes have also been covered in the LDAP ID Store Attributes table and SCIM ID Store Configs table.

Using the User Locker APIs

The UserLocker API allows an administrator to check and change the status of a user from unlock to lock and vise versa. To determine how to handle the user status for a certain directory, this API uses an ID Store config called "statusEncoding." The API also checks if the ID Store defined in the solution is the authoritative source for locking the user using an ID Store config called "isAuthoritativeForIDLCM". These ID Store attributes have been explained above.

Users who use the **ZFP flow** are not looked up in the ID Store User Directory; hence, they cannot be locked in ID Store but can be locked in the IDSP database. The locked-out user will be automatically unlocked later based on the tenant setting, "userAutoUnlockPeriodMinutes."

The administrator can use the following ID Store APIs to update the users status and also to retrieve the current status of the user from the ID Store directories:

Operation Description	Operation Type	Endpoint URLs
This API is used to update the user's account status as "Locked" or "Unlocked" in the ID Store by sending the request parameter called, "locked". When "locked" = "true", the user gets locked, and when "locked" = "false", the user gets unlocked.	POST	https://${SSP_FQDN}/{{tenantName}}/idstore/v1/UserLocker/{{userLoginid}}
This API is used to check the user's "Status". The API response contains information on whether the user is locked. The user's status is returned as "locked" = "true/false" in the payload.	GET	https://${SSP_FQDN}/{{tenantName}}/idstore/v1/UserLocker/{{userLoginid}}

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/idstore/v1/UserLocker/{{userLoginid}}`
  should look similar to:
  `https://example.com/default/idstore/v1/UserLocker/{{userLoginid}}`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Example of the POST UserLocker API

```
POST /idstore/v1/UserLocker/{UserLoginId}

200 OK
{
"locked": false
}

400 BAD REQUEST
{
    "errorCode": "1003022",
    "errorMessage": "Unable to update user's lockout state: Missing status encoding value in ID Store
 definition"
}

400 BAD REQUEST
{
    "errorCode": "1003019",
    "errorMessage": "Unable to update user's lockout state"
}
```

Example of the GET UserLocker API:

```
GET /idstore/v1/UserLocker/{UserLoginId}
```

```
200 OK
{
"locked": false
}

400 BAD REQUEST
{
    "errorCode": "1003022",
    "errorMessage": "Unable to detect user's lockout state: Missing status encoding value in ID Store
 definition"
}
```

Resetting the Locked User Account

The locked user account can be reset in two ways:

- An administrator can manually reset a user's account by invoking the "/UserLocker" API with the request, "locked" = "false/true". If set to true, the user account is locked and vice versa.
- A locked user account that uses the ZFP flow gets automatically unlocked after the timeframe specified in the attribute "userAutoUnlockPeriodMinutes" ends. See Global/Tenant Settings for more details.

User Lock Behavior Scenarios and Error Messages Table

The following table describes the user lock behavior based on the configuration of the following two settings:

- hideAuthenticationFailureReason
- skipHideAuthenticationFailureReasonForLockedUser

Scenario	**Pattern A** • hideAuthenticationFailureReason = false • skipHideAuthenticationFailureReasonForLockedUser = false	**Pattern B** • hideAuthenticationFailureReason = true • skipHideAuthenticationFailureReasonForLockedUser = false	**Pattern C** • hideAuthenticationFailureReason = true • skipHideAuthenticationFailureReasonForLockedUser = true
When the user is not found	"errorMessage": "User disambiguation failed. [USER_NOT_FOUND] []	The user is displayed with the password authentication screen a set number of times as determined by the configuration of the "maxStrikeCount" property. Once this threshold is reached, the user receives an "AUTH_DENIED" response.	The user is displayed with the password authentication screen a set number of times as determined by the configuration of the "maxStrikeCount" property. Once this threshold is reached, the user receives an "AUTH_DENIED" response.
When the user is locked	"errorMessage": "The user in the request is not active. [8800041] [SmDisabledReason=2,SmAuthReason=7,Encoding is missing]	The user is displayed with the password authentication screen a set number of times as determined by the configuration of the "maxStrikeCount" property. Once this threshold is reached, the user receives an "AUTH_DENIED" response.	"errorMessage": "The user in the request is not active. [8800041] [SmDisabledReason=2,SmAuthReason=7,Encoding

When the user is disabled by the admin	"errorMessage": "The user in the request is not active. [8800041] [SmDisabledReason=1,SmAuthReason=7,Encoding=is as def"	The user is displayed with the password authentication screen a number of times as determined by the configuration of the "maxStrikeCount" property. Once this threshold is reached, the user receives an "AUTH_DENIED" response.	"errorMessage": "The user in the request is not active. [8800041] [SmDisabledReason=1,SmAuthReason=7,Encodin
When the user is disabled due to inactivity	"errorMessage": "The user in the request is not active. [8800041] [SmDisabledReason=4,SmAuthReason=25,Encoding=is as def"	The user is displayed with the password authentication screen a number of times as determined by the configuration of the "maxStrikeCount" property. Once this threshold is reached, the user receives an "AUTH_DENIED" response.	"errorMessage": "The user in the request is not active. [8800041] [SmDisabledReason=4,SmAuthReason=25,Encodi

Lock User On Next Auth Failure Warning in the Authentication Response

The "lockUserOnNextAuthFailure" parameter is a warning message that is returned in the authentication response of any obligation (for example- Password, SMS OTP), when there is only one user strike left before the system locks the user's account. Therefore, if the user strike count left in the authentication flow is more than one, this parameter is not returned.

The following example shows the "userLockOnNextAuthFailure"="true" warning in the password obligation authentication response:


```
{
    "flowState":
 "SkFhcUhSQUQram1NbjBtL1hhSHE0eERTUHBtekxKNmszMUdXN0Uzb1VIS3N1WkovampBa1VYaWRBakFuejdybXU2M1VFS3dpanJxMmFjaWJjVCtGenJUN
    "nextaction": "PASSWORD_AUTH",
    "currentFactors": [
        "PASSWORD"
    ],
    "data": {
        "errorMessage": "Incorrect credentials",
        "errorCode": "3001033"
    },
    "userLockOnNextAuthFailure": "true",
    "additional": {
        "idpName": "TestConfig2",
        "idpGuid": "01f6dac1-08ce-4f4b-b777-b7f474ac0620",
        "idpType": "ldap",
        "currentFactorLevel": 1,
        "predicatesForCurrentFactors": [
            {
                "factor": "PASSWORD",
                "predicate": "TestConfig2",
                "providerAmr": "PWD"
            }
        ],
```


This parameter (lockUserOnNextAuthFailure) is also dependent on two tenant setting configurations:

- ***hideAuthenticationFailureReason*** -

  - If "hideAuthenticationFailureReason"="false", the authentication response returns the "lockUserOnNextAuthFailure" warning.
  - If hideAuthenticationFailureReason=false, and the next action is 'AUTH_DENIED', then the following error message is displayed in the response:

```
"data":
        {
            "errorCode": "00000125",
            "errorMessage": "There are no registed credentials and inline enrollment is disabled"
        }
```

  If "hideAuthenticationFailureReason"="true", the "lockUserOnNextAuthFailure"="true" warning is not returned in the authentication response of an obligation.
- ***skipHideAuthenticationFailureReasonForLockedUser*** - This parameter is dependent on the "hideAuthenticationFailureReason" parameter. Therefore, consider the following use cases:
  - When "hideAuthenticationFailureReason"="true" and "skipHideAuthenticationFailureReasonForLockedUser"="true", the authentication response of an obligation returns the "lockUserOnNextAuthFailure"="true" warning. This is because the "skipHideAuthenticationFailureReasonForLockedUse" parameter setting takes precedence over the "hideAuthenticationFailureReason" setting.
  - When "hideAuthenticationFailureReason"="true" and "skipHideAuthenticationFailureReasonForLockedUser"="false", the authentication response of an obligation does not return the "lockUserOnNextAuthFailure"="true" warning.
  - When "hideAuthenticationFailureReason"="false" and "skipHideAuthenticationFailureReasonForLockedUser"="false", the warning "lockUserOnNextAuthFailure"="true" is returned in the authentication response of an obligation.

These tenant settings, "hideAuthenticationFailureReason", and "skipHideAuthenticationFailureReasonForLockedUser" are coved in Global/Tenant Settings.

Configuring the User Auto Enrollment and Credential Synchronization

This topic includes the following sections:

- Auto Enrollment and Credential Synchronization
- Multiple Credential Synchronization Flows
- Table Showing Credential Synchronization Scenarios

Auto Enrollment and Credential Synchronization

The Auto enrollment functionality allows automatic synchronization of user credentials from an ID Store to the Symantec Identity Security Platform database when the "isAutoEnrollmentAllowed" tenant setting is set to **true**. This setting ensures that the IDSP's user credentials always reflect the latest state of the ID Store as the synchronization happens between user data in the ID Store and its credentials in IDSP.

The solution supports multiple credentials of a given type for a user during the auto enrollment or credential synchronization process. For example, multiple phone numbers or email IDs of a user can be enrolled. Apart from synchronizing user credentials from ID Store, it also synchronizes users' credentials that come from external systems acting as a primary authenticator, such as SiteMinder, during Zero Footprint or OIDC flow.

The solution maintains the origin and origin type of the auto enrolled credentials in the following ways:

Source of the credentials	ORIGIN	ORIGIN TYPE
Using the ID Store (LDAP/SCIM/Internal/SiteMinder)	<guid> of the id store	IDSTORE
Using an ID Token	<token-issuer>	IDTOKEN

Using the self-service flow	USER	USER
Using the import tool	IMPORT	IMPORT

User Credential Synchronization

The synchronization of the credentials from the source to the IDSP's database can be done by following the steps that are mentioned in this section. However, there are two points to consider before synchronizing the user credentials:

- The validation of user credentials coming from ID store or ID Token Hint will depend on the status of the "***isAuthoritativeForUserCredsLCM***" ID Store configuration setting. If it is set to true, IDSP will not validate user credentials from the ID Store or ID Token Hint assuming that these credentials are coming from a reliable source. If it is set to false, then IDSP verifies the status of the "skipValidationInAutoEnrollment" tenant setting.
- If the "***skipValidationInAutoEnrollment***" tenant setting is set to true, then IDSP skips the validation of user credentials during auto enrollment and credential synchronization. However, if set to false, then IDSP continues to validate the user credentials from ID Store or ID Token Hint.

Additional Notes on User Credential Synchronization Scenarios

- The user credentials are updated in the IDSP database when the credentials in the database differ from that in the ID Store.
- The user credentials are created in the IDSP database if the credentials do not exist. The creation of user credentials depends on whether the SMS and Email Id are added correctly in the ID Store. For example, if the phone number or the email id is specified in the wrong format, then the sync fails.
- In ZFP flow, ensure to enable the "auto_inline_enrollment" setting in SiteMinder along with "isAutoEnrollmentAllowed" setting in IDSP for the auto enrollment or credential synchronization to work correctly. See ZFP.
- Make sure that the Identity Store Config from which the credentials should be synced has logical attributes defined for "email" and "phone_number" mapping it to the physical id store attributes. The physical attribute representing the phone number is expected to comply with the international format (+1 ...).
- In addition to email and phone number, a credential of type TOTP can be auto enrolled by syncing it from the id store using the logical attribute, "credentials:devices:totp". For example, credentials:device:totp=carLicense. This logical attribute maps to a physical attribute containing TOTP seeded in one of the following formats -
  a. ***Symmetric Encryption using a {<Secret's alias>}*** - prefix indicating the alias of the secret containing the encryption key.
     AES-based data encryption using customer-supplied secret pointed by <secrets-alias> (Key must be a multiplier of 16, IV must be equal to the first 16 characters of the key, with offset 0)
  b. ***PKI (RSA) Encryption using a {<Key's alias>}*** - prefix indicating the alias of an RSA Key containing the private key to be used for decryption.
  c. ***PKI encryption using {x5c} prefix*** - similar to the previous PKI encryption, but using the tenant signing key indicating that the PKI encryption was done using the tenant's signing cert (using the public key of the tenant signing key which can be retrieved from the jwks endpoint)
  d. ***Plain text*** - No prefix. TOTP information is considered sensitive, hence it is not recommended to store it in plain text.
- Supports only SMSOTP, EmailOTP, IVROTP, and TOTP credentials.

To auto enroll the user credentials, perform the following steps:

1. Invoke the following API to update the tent/global settings:

```
PATCH https://${SSP_FQDN}/{{tenantName}}/admin/v1/Configs
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.

For example:
`https://{{host}}/{{tenantName}}/admin/v1/Configs`
should look similar to:
`https://example.com/default/admin/v1/Configs`

- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

2. Configure the following tenant/global settings:

NOTE

- The following configuration settings are also covered in Global/Tenant Settings.
- The tenant administrator can override the global settings using tenant settings.

a. **isAutoEnrollmentAllowed**
Auto enrollment of IDSP credentials happen only if this property is set to true. For ZFP flow, along with this property, auto enrollment should be enabled in the ID token.

NOTE

- If **isAutoEnrollmentAllowed** and **isInlineEnrollmentAllowed** are set to **false**, a user whose credentials are not registered in IDSP is denied further authentication.
- If **isAutoEnrollmentAllowed** is set to **false** and **isInlineEnrollmentAllowed** is set to **true**, a user whose credentials are not registered in IDSP is prompted for registration.

Default value: true
Possible values: true or false

b. **preserveOriginInAutoEnrollment**
This property ignores or considers the origin and origin type in the Credential Synchronization flow depending on the following values:

- If **preserveOriginInAutoEnrollment** is set to **false**: Ignores the origin and origin type in the Credential Synchronization flow and replaces an existing credential with a new credential from the authoritative source. **For example**: If there is an existing credential in the IDSP database with the cred type SMS OTP and the Origin and Origin Type as User (registered via the self-service portal) and the "preserveOriginInAutoEnrollment" setting is set to false, the existing credential is replaced with a new credential when that new credential comes from another authoritative source, such as the ID store (Origin). Note that the replacement takes place only when the cred type is the same but the cred value is different.
- If **preserveOriginInAutoEnrollment** is set to **true**: Considers the Origin and Origin Type in a Credential Synchronization flow and replaces an existing credential with the new credential from the authoritative source only when the origin and credential type match (ignores case). Note that the existing credential is replaced only when the credential value is changed.

Default value: false
Possible values: true or false

c. **autoEnrollCredentialSyncPolicy**
This property controls how the credentials coming from an endpoint are synced into the IDSP database.

NOTE
For this property to work, the "**preserveOriginInAutoEnrollment**" setting should be set to **true**.

Default value: REPLACE
Possible values: REPLACE or MERGE
REPLACE and MERGE have the same behavior in replacing the existing credentials of a user in the following scenarios:

- **When the credentials from a specific origin and origin type are already stored in the IDSP database, and an authentication flow contains new credentials that are from the same origin and origin type**: In this case, the existing credentials are replaced with the new credentials in the database.

Example: Assume that an email ID with the IDSTORE origin and origin type is already stored in the database. If the new credential is an email ID with the same origin and origin type in the authentication flow, the existing email ID is replaced with the new email ID.

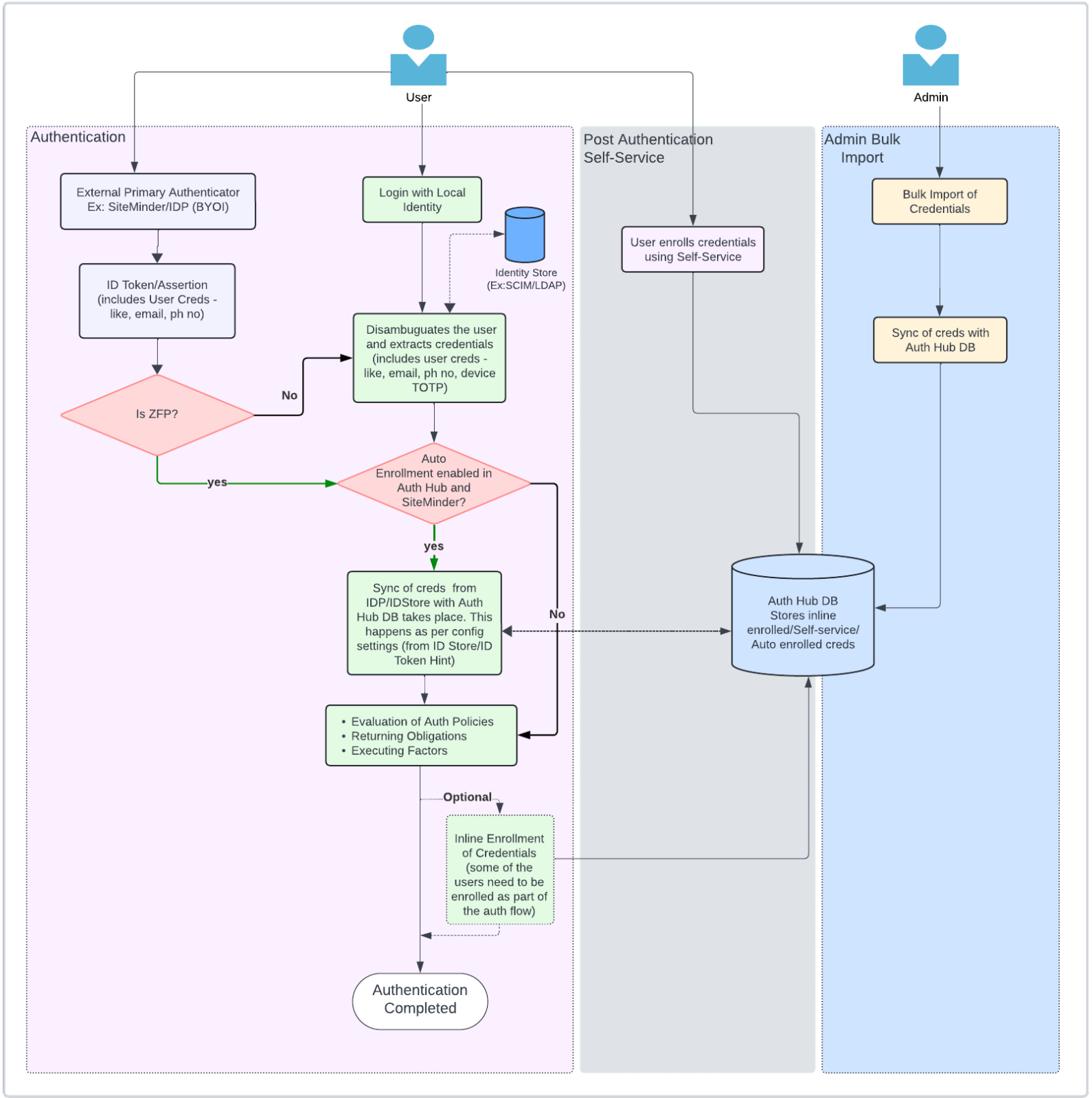
- **When an authentication flow contains new credentials that are not available in the IDSP database**: In this case, the new credentials and its origin and origin type are stored in the database.
  Example: Assume that an email ID with the USER origin and origin type is already stored in the database. If the authentication flow contains a mobile number with the IDSTORE origin and origin type, the mobile number with the IDSTORE origin is stored in the database. There is no change to the existing email ID with the USER origin and origin type.

However, REPLACE and MERGE differ in replacing the existing credentials of a user in the following scenario:

- **When the credentials with a specific origin and origin type are already stored in the IDSP database, and the authentication flow contains the same credentials but from a different origin**: If REPLACE is selected, the origin and origin type of the existing credential is replaced with the origin and origin type of the new credential in the authentication flow. If MERGE is selected, the existing origin and origin type is *not* replaced.
  Example: Assume that an email ID with the USER origin and origin type is already stored in the database. If an authentication flow contains the same email ID but from the IDSTORE origin and origin type, the behavior of the setting is as follows:
  - If REPLACE is selected, the existing origin and origin type is replaced with IDSTORE.
  - If MERGE is selected, the existing origin and origin type is not replaced and it remains USER.

Multiple Credential Synchronization Flows

The following diagram shows the different ways in which the multi credentials can sync with the IDSP database:


User
Admin
Authentication
Post Authentication Self-Service
Admin Bulk Import
External Primary Authenticator Ex: SiteMinder/IDP (BYOI)
Login with Local Identity
Identity Store (Ex:SCIM/LDAP)
User enrolls credentials using Self-Service
Bulk Import of Credentials
ID Token/Assertion (includes User Creds - like, email, ph no)
Disambuguates the user and extracts credentials (includes user creds - like, email, ph no, device TOTP)
Sync of creds with Auth Hub DB
Is ZFP?
No
yes
Auto Enrollment enabled in Auth Hub and SiteMinder?
yes
Sync of creds from IDP/IDStore with Auth Hub DB takes place. This happens as per config settings (from ID Store/ID Token Hint)
No
Auth Hub DB Stores inline enrolled/Self-service/ Auto enrolled creds
• Evaluation of Auth Policies
• Returning Obligations
• Executing Factors
Optional
Inline Enrollment of Credentials (some of the users need to be enrolled as part of the auth flow)
Authentication Completed

External Primary Authentication Flow

Flow Description:

1. The user initiates authentication into IDSP using an external system acting as a Primary Authenticator, such as SiteMinder, or using BYOI from a SAML IDP/OpenID Provider. The primary authenticator generates an ID Token or a SAML assertion and returns it to the IDSP server during the authentication flow. The ID Token (sent as id_token_hint) or the SAML assertion (sent as part of SAML authentication response) may contain information about the user credentials such as Email or Phone number. To enable auto enrollment during the authentication flow in IDSP, the "**auto_inline_enrollment**" property must be configured as **true** in the ID Token/ID Token Hint or SAML Assertion.
   - When the "**auto_inline_enrollment**" property is set to **false**: the email and phone number details that are available with the external authenticator are ignored.
   - When the "**auto_inline_enrollment**" property is set to **true**: the email and phone number details that are available with the external authenticator are considered for auto enrollment.
2. Depending on the ZFP configuration controlled by the application configuration or the IDP configuration in IDSP, the user may or may not be required to exist in the IDSP identity store.
3. In a ***non-ZFP (Zero FootPrint Identity) mode***, the user must be found (disambiguated) in the designated ID store, which is a source for additional user credentials such as email, phone no, and device (totp) information. ***For ZFP mode this step can be skipped***.
4. In IDSP, the "**isAutoEnrollmentAllowed**" tenant setting must be set to **true**, to enable auto enrollment during the authentication flow. When this setting is configured as true, the user credential information is taken from the ID Token/ SAML assertion, and/or ID Store and synchronized with the existing credentials that are already registered in the IDSP DB according to the "**autoEnrollCredentialSyncPolicy**" setting.
5. The IDSP server checks the status of the "**autoEnrollCredentialSyncPolicy**" tenant property as follows:
   - If "**autoEnrollCredentialSyncPolicy**" is set to **REPLACE**: IDSP replaces all the existing credentials of a user that matches the origin of the ID Token's issuer, with the credentials in the ID Token Hint. If a duplicate credential with an origin that differs from the ID Token's issuer already exists in the IDSP database, the origin of the existing credential is replaced with the origin of the ID Token's issuer.
   - If "**autoEnrollCredentialSyncPolicy**" is set to **MERGE**: IDSP replaces all the existing credentials of a user that match the origin of the ID Token's issuer, with the credentials in the ID Token Hint. If a duplicate credential with an origin that differs from the ID Token's issuer already exists in the IDSP database, the origin of the existing credential is not replaced.

     NOTE
     If the tenant property, "**preserveOriginInAutoEnrollment**" is set to **false**, the credentials' origin is ignored during the sync operation. This setting is also covered in Managing Settings.
6. If "**auto_inline_enrollment**" property is not set in the ID Token or the tenant property "**isAutoEnrollmentAllowed**" is set to **false** in IDSP, then the IDSP server skips auto enrollment or credential synchronization of the existing IDSP credentials with the credentials that are coming from the ID Token/IDToken Hint/SAML Assertion.
7. The Authentication Policies are evaluated and obligations are returned. The user executes the authentication factors to get authenticated.
8. (Optional)-In case, inline enrollment is required for any of the credentials, the user is prompted to enroll their credentials inline.

ID Store Flow

In this flow, the user initiates the login using the Identity store (LDAP/SCIM).

Flow Description:

1. The user initiates authentication into IDSP using the ID Store flow.
2. In this flow, the user credentials are also fetched from the ID Store like LDAP or SCIM during user disambiguation.
3. If the "**isAutoEnrollmentAllowed**" tenant property is set to **true** in IDSP, then the IDSP server checks the status of the "**autoEnrollCredentialSyncPolicy**" tenant property.

- If "**autoEnrollCredentialSyncPolicy**" is set to **REPLACE**: IDSP replaces all the existing credentials of a user that have the origin as IDSTORE, with the user credentials from the ID Store. If a duplicate credential with an origin other than IDSTORE already exists in the IDSP database, the origin of the existing credential is replaced with IDSTORE.
- If "**autoEnrollCredentialSyncPolicy**" is set to **MERGE**: IDSP replaces all the existing credentials of a user that have the origin as IDSTORE, with the user credentials from the ID Store. If a duplicate credential with an origin other than IDSTORE already exists in the IDSP database, the origin of the existing credential is not replaced.

4. If the tenant property "**isAutoEnrollmentAllowed**" is set as **false** in IDSP, then the IDSP server skips auto enrollment or credential sync of the existing IDSP credentials with the credentials that are coming from the ID Store.
5. The Authentication Policies are evaluated and obligations are returned. The user executes the authentication factors to get authenticated.
6. (Optional)-In case, inline enrollment is required for any of the credentials, the user is prompted to enroll their credentials inline.

Self-Service Flow

Flow Description:

1. The user initiates authentication into IDSP using the Self-Service flow to enroll their credentials.
2. The user registers their phone number or email id in the IDSP.
3. The enrolled credentials are automatically created in the IDSP database.

Import Flow

Flow Description:

1. Tenant Admin invokes the Bulk API. For more information, see Bulk Import Credentials and Bulk API.
2. The credentials get auto synced with the IDSP database.
   The credential auto-sync may include POST, PUT, and DELETE operations.

Table Showing Credential Synchronization Scenarios

The following table displays the various scenarios of syncing multiple credentials of users with the IDSP database:

- Scenarios of synchronizing multiple credentials in IDSP database

Configuring Inline Enrollment

The Inline Enrollment feature enables users to swiftly enroll their unregistered credentials within the authentication process at any stage, without interrupting the authentication flow.

When user credentials are not registered in the user directory prior to authentication, the Inline Enrollment feature allows users to enroll their credentials on the fly during the Sign In flow. If users are entitled to Multi-factor Authentication (MFA), they are prompted to select factors for the inline enrollment at all authentication levels. When the inline enrollment process completes, user credentials are updated in the user directory. The next time users log in, they will be presented with the authentication factors for which they have enrolled their credentials.

This topic includes the following sections:

- Configuration Settings Governing Inline Enrollment
- Authentication Flow for Inline Enrollment
- Sample Scenarios

Configuring Tenant Settings Governing Inline Enrollment

The following settings govern the inline enrollment of user credentials in the Symantec Identity Security Platform authentication flow.

- **inlineEnrollmentAllowed (AuthN Policy Property)**: This **authentication policy property** controls whether inline enrollment for new authentication factors is explicitly permitted or prevented. It allows you to tailor enrollment based on the security context, for instance, enabling it for on-network access and disabling it for off-network access to prevent unauthorized additions. When this policy property is explicitly set to true or false, it overrides and effectively **ignores the tenant-level isInlineEnrollmentAllowed** setting. For more information, see AuthN Policy Property.
  Therefore, it is important to check whether the inlineEnrollmentAllowed authN policy property is configured, if yes then inlineEnrollmentAllowed tenant setting will be disregarded.
- **isInlineEnrollmentAllowed**: Inline registration of user credentials is allowed only when this configuration setting is set to **true.**
- **inlineEnrollmentFactorsRequired**: Specifies comma-separated mandatory authentication factors with or without the corresponding authentication levels at which these factors should be executed before enrolling the user credentials inline. Users can proceed with the inline registration process only after completing the mandatory authentication factors specified in this setting. The default authentication factor set for this setting is PASSWORD.
  - The mandatory authentication factor configured as part of the **InlineEnrollmentFactorsRequired** setting is considered only during the inline registration of the **FIRST** unregistered factor, irrespective of the authentication levels. In the subsequent registration flow, users proceed with the inline enrollment of unregistered credentials only after authenticating themselves with an already registered credential at the same authentication level. If a registered credential does not exist at the same authentication level, then the user is allowed to proceed with the registration of an unregistered credential.
  - If a user is not registered for any of the mandatory factors that are specified in the "inlineEnrollmentFactorsRequired" setting, except for "Password," the authentication process results in AUTH_DENIED. For example, if SMSOTP is the configured as the mandatory inline factor, but the factor is not already registered then the user is denied to proceed with the inline enrollment flow. Upon denied authentication, the appropriate error message is displayed to the user only when the setting, ***"hideAuthenticationFailureReason"*** is configured as "***false***." To know more about this setting, see here.
  - You can add multiple mandatory factors and can determine their respective authentication levels in the "inlineEnrollmentFactorsRequired" setting. The sequence of authenticating these factors depend upon the levels specified in the setting's value. For instance, in the following example the user has to complete the "PASSWORD:1 or SMSOTP:1 as these factors are specified as level (1) before moving to the second level or authentication specified as (2), that is FIDO:2".

    ```
    {
      "name": "inlineEnrollmentFactorsRequired",
      "value": "PASSWORD:1,SMSOTP:1,FIDO:2",
      "origin": "tenant"
    }
    ```

  - In case the authentication levels for the mandatory factors are not specified within the "inlineEnrollmentFactorsRequired" setting, all these factors are at assumed to be at level (1) one. For instance, if you define the factors in the following manner, both PASSWORD OR SMSOTP factors are automatically considered to be at level one.

    ```
    {
     "name": "inlineEnrollmentFactorsRequired",
     "value": "PASSWORD,SMSOTP",
     "origin": "tenant"
    }
    ```

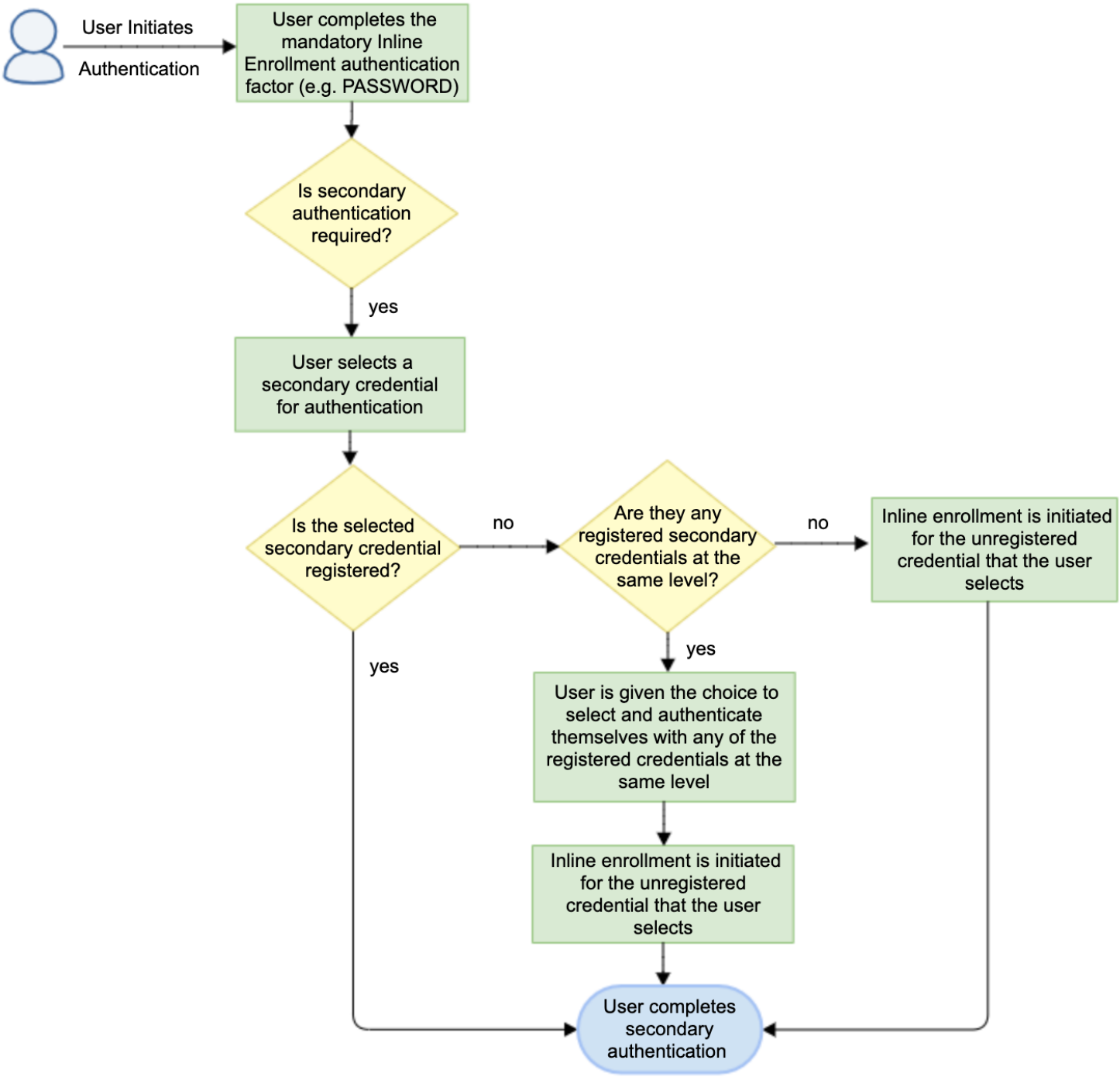
- **isInlineOTPEnrollmentToBePersisted:** User credentials that are enrolled using inline enrollment persist in the database when this property is set to **true**. When this configuration setting is set to **false**, all the user credentials that were enrolled using inline enrollment gets deleted.

These tenant configurations are also covered in Managing Settings topic.

Authentication Flow for Inline Enrollment

The following diagram is a pictorial representation of how the inline enrollment of user credentials works in an authentication flow.

Figure 8: Inline Enrollment - Authentication Flow



Sample Scenarios

Given are a few sample scenarios to help you understand how the authentication flow for inline enrollment works.

Scenario 1:

Assumption: Nina Bruce is obligated to sign into an application by satisfying the authentication factors - PASSWORD:1, SMSOTP:2, EMAILOTP: 2. She has not registered her phone number and email ID with the user directory prior to authentication. Assume that an administrator has configured "PASSWORD" as the mandatory factor for Inline Enrollment.

User Flow: Nina Bruce completes primary and mandatory authentication by providing PASSWORD in the SignIn screen. On successful authentication, she is given the choice to select the secondary factor as either SMSOTP or EMAILOTP. Assume that she selects SMSOTP as the secondary factor. Since SMSOTP is not registered, she is prompted to register or enroll her phone with the user directory. Once registered, she can use SMSOTP as the factor for the subsequent authentications.

Scenario 2:

Assumption: Emily Russell is obligated to sign into an application by satisfying the authentication factors - PASSWORD:1, MOBILEOTP: 2, EMAILOTP:2, SMSOTP:2 She has registered EMAILOTP and SMSOTP with the user directory prior to authentication. Assume that an administrator has configured "PASSWORD" as the mandatory factor for Inline Enrollment.

User Flow: Emily Russell completes primary and mandatory authentication by providing PASSWORD in the SignIn screen. On successful authentication, she is given the choice to select the secondary factor as either MOBILEOTP, EMAILOTP, or SMSOTP. Assume that she selects MOBILEOTP as the secondary factor. Since MOBILEOTP is not registered, she is forced to authenticate herself for the registered credential at the same level before allowing inline enrollment of the unregistered credential. In this case, she is prompted to select from one of the existing registered factors (EMAILOTP, SMSOTP). After successfully completing authentication for the registered credential at the same level, she is prompted to register or enroll her MOBILEOTP with the user directory. Once registered, she can use MOBILEOTP as the factor for the subsequent authentications.

Scenario 3:

Assumption: Brenda Steer is obligated to sign into an application by satisfying the authentication factors - MOBILEOTP:1, PASSWORD:2, SMSOTP:2, EMAILOTP:2, FIDO:2. She has registered MOBILEOTP, EMAILOTP, and FIDO with the user directory prior to authentication. Assume that an administrator has configured "PASSWORD" as the mandatory factor for Inline Enrollment.

User Flow: In the first level of authentication, Brenda Steer authenticates herself with MOBILEOTP. After completing the first level of authentication, she is given the choice to select the secondary factor as either PASSWORD, SMSOTP, EMAILOTP, or FIDO. Assume that she selects SMSOTP as the secondary factor. Since SMSOTP is not registered, she is prompted to authenticate with PASSWORD as it is the mandatory factor for Inline Enrollment. Next, she is prompted to register or enroll SMSOTP with the user directory.

Scenario 4:

Assumption: Enid Dunn is required to sign into an application using FIDO:1 authentication. However, she has not enrolled her credentials for FIDO yet. She intends to carry out inline registration for her FIDO credentials. Before proceeding with the authentication, she has already enrolled SMSOTP and EMAILOTP in the user directory. It is important to note that an administrator has established PASSWORD:1, SMSOTP:2, EMAILOTP:3 as the mandatory factors for Inline Enrollment.

User Flow: In the first level of authentication, Enid Dunn verifies herself using PASSWORD:1. Once this step is completed, she proceeds to authenticate using SMSOTP:2, and subsequently EMAILOTP:3. Upon successfully completing all three mandatory authentication levels, she is prompted to enroll for FIDO (biometrics). After successfully enrolling, she can use FIDO as the factor for the subsequent authentications.

Scenario 5:

Assumption: Dean Roth must access an application using FIDO:1 authentication. However, she has not enrolled her credentials for FIDO yet. She intends to carry out inline registration for her FIDO credentials. Additionally, Dean has not added her phone number to the user directory to receive SMSOTP. Note that the administrator has configured PASSWORD:1 and SMSOTP:2 as the required factors for Inline Enrollment.

User Flow: In the first level of authentication, Dean Roth verifies herself using PASSWORD:1. Once this step is completed, she is denied authentication (AUTH_DENIED) since she did not register her phone number to receive SMSOTP.

Scenario 6:

Assumption: Christopher Moran is required to sign into an application using FIDO:1 authentication. However, he has not enrolled her credentials for FIDO yet. He intends to carry out inline registration for his FIDO credentials. Before proceeding with the authentication, he has already registered his phone number in the user directory to receive SMSOTP. It is important to note that an administrator has established PASSWORD, SMSOTP as the mandatory factors for Inline Enrollment.

User Flow: In this use case, the factors are specified without the levels at which these should be authenticated. Therefore, both these factors (PASSWORD or SMSOTP) are assumed to be at level one. During the inline enrollment, Christopher is prompted to select either PASSWORD or SMSOTP as the authentication factor. He verifies using PASSWORD and upon successfully completing password authentication, he is prompted to enroll for FIDO (biometrics). After successfully enrolling the FIDO credentials, Christopher can use FIDO for the subsequent authentications.

Configuring Trusted Devices

Learn how to reduce the frequency of multi-factor authentication prompts for recognized devices when accessing a secured resource by setting up your device as a trusted device.

As a tenant admin, when you perform activities that deal with sensitive information such as editing user PII data or performing banking transactions, you are required to perform multi-factor authentications to ensure that this sensitive information is only accessed by you, and is secured. Most often, to perform these activities, you use the same set of devices like your personal mobile phone, tablet, or home computer. Therefore, you can use the trusted devices functionality to designate the most commonly used device or a handful of devices as trusted, which will allow you to log in to the protected resource by performing only the first factor of authentication and skipping other factors in all subsequent logins.

NOTE

- Trusted Device functionality is supported in SignIn UI, Sample app, and Mobile (Hedmoral -Android & iOS) applications.
- To skip complete authentication on known secure endpoints, configure the "**urn:iam:app:skipAuthenticationForTrustedDevice**" metadata attribute (see Application Metadata).
- "TrustedDevices" only applies to Multi-Factor Authentication scenarios. You are advised not to use trusted devices for sensitive applications.
- If there are multiple policies that are assigned to an application that has both “mfaFrequency”: “OnceForTrustedDevice” and “mfaFrequency”: “EveryTime” configured, then the latter takes precedence over the former and the user is obliged to authenticate all the factors.
- The "Remember This Device" option gets cleared off from the UI and the login device is treated as a new device and not a trusted device,
  when:

  - the device has not been used for the period that has been configured in the "trustedDeviceExpiryDays" parameter.
  - you clear the device-tag cookie from the browser that you have used to authenticate.

  In both these cases, you must select the "Remember This Device" checkbox in the UI once again.
- If multiple authentication factors are configured for your authentication flow, then the "Remember This Device" option is displayed only before authenticating the last factor. For example, if you have factors in this order, Password>SMSOTP, then the "Remember This Device" checkbox gets displayed only before the SMSOTP authentication.
- When risk is enabled for a trusted device, it is essential that the device recognition risk rule is also active. See Device Recognition risk rule.

Trusted Device Flows

There are two scenarios to explain regarding the trusted device functionality:

- ***Login from a New Device***: In this scenario, we will walk through the steps to set up your new device as a trusted one for future operations.
- ***Login from a Trusted Device***: In this scenario, your device is already established as a trusted device, which streamlines the multi-factor authentication process and grants you access to secured resources immediately after fulfilling the first authentication factor.

These scenarios have been explained in the following sections.

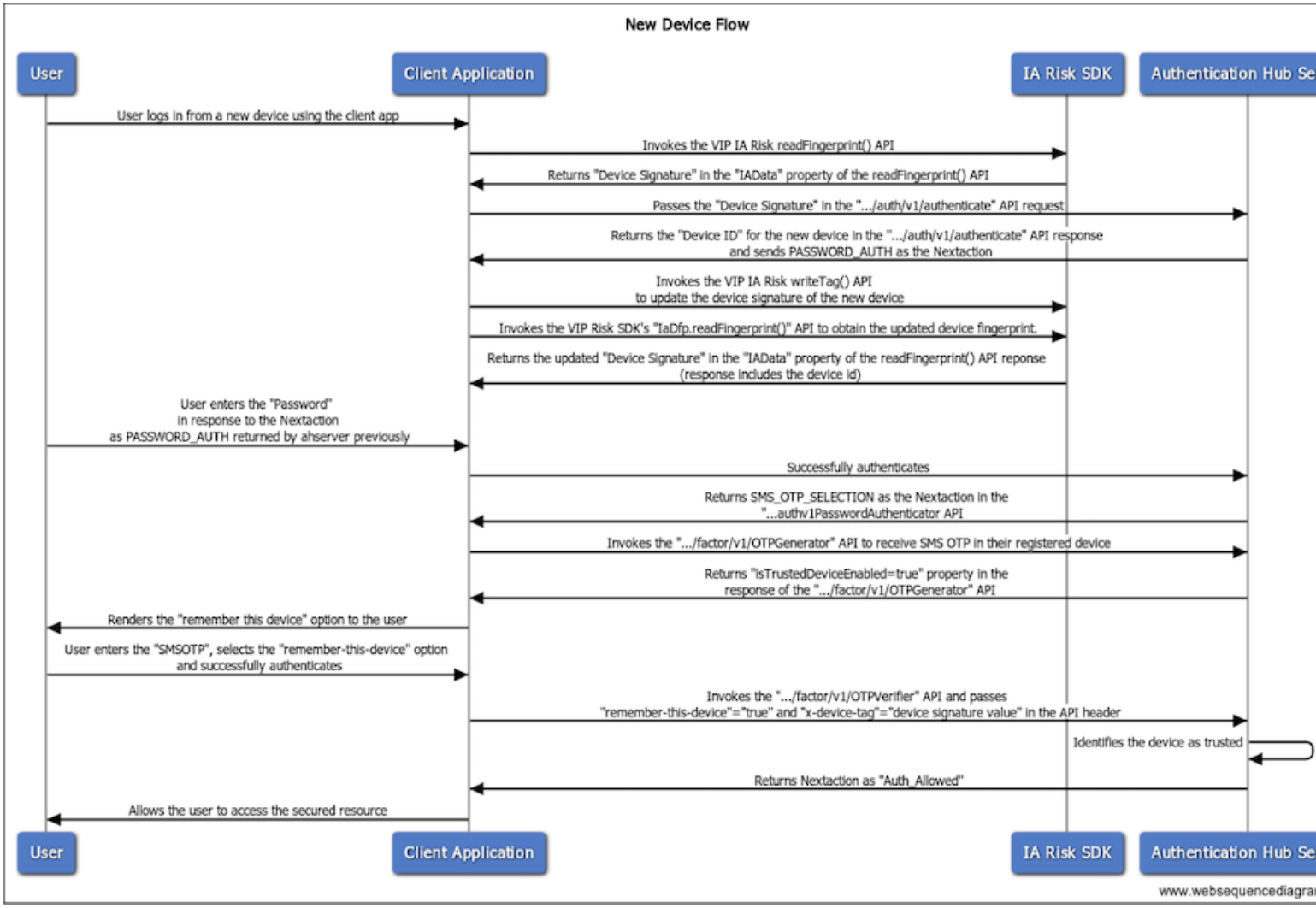
New Device Flow

In this scenario, we will walk through the steps required to set up your new device as a trusted one for future authentications.

List of participants and what they represent:

- ***User*** : One who initiates the authentication flow to configure a new device as a trusted device in Symantec Identity Security Platform.
- ***Client Application (Mobile/Web App)***: Represents the application that calls the IDSP REST APIs and VIP IA Risk SDK to obtain the device fingerprint to complete the authentication flow.
- ***IA SDK***: VIP IA RISK SDK that is called by the client application to get and update the device fingerprint.
- ***IDSP Server***: Server that is used to evaluate and authenticate the trusted device flow.

The following sequence diagram outlines the authentication flow that occurs when a user logs in from a new device and sets it up as a trusted device.



Assumption: In this flow we have assumed that the primary authentication factor is "***Password"***, while the secondary authentication factor is "***SMSOTP"***.

User experience when you login from a new device:

1. The user logs in to IDSP using a new device from the client app.
2. The client app invokes the VIP Risk SDK's "`IaDfp.readFingerprint()` " API to retrieve the device fingerprint value.
3. The VIP Risk SDK's "`IaDfp.readFingerprint()` " API returns the device fingerprint as part of the "iadata" property in the response.
4. The client app invokes the IDSP's "**/auth/v1/authenticate"** API and passes the device fingerprint value in the API request.
5. The IDSP server returns the "DeviceID" of the new device in the "**/auth/v1/authenticate"** API response and also sends "PASSWORD_AUTH" as the Nextaction to the client app.
6. The client app invokes the VIP Risk SDK's "writeTag()" API to update the device fingerprint for the new device.
7. The client app invokes the VIP Risk SDK's "`IaDfp.readFingerprint()` " API once again to obtain the updated device fingerprint.
8. The VIP Risk SDK's "`IaDfp.readFingerprint()` " API returns the updated device fingerprint along with the device id as part of the "iadata" property in the response.

9. User enters the "Password" in response to the Nextaction as PASSWORD_AUTH returned by the IDSP server previously (in step 5) and authenticates successfully.
10. IDSP server returns SMS_OTP_SELECTION as the Nextaction in the "**...\auth\v1\PasswordAuthenticator**" API.
11. The client app invokes the "**.../factor/v1/OTPGenerator**" API to receive OTP in their registered device.
12. Assuming that ***SMSOTP*** is the ***last authentication factor***, IDSP server returns the ***"isTrustedDeviceEnabled=true"*** property in the ***".../factor/v1/OTPGenerator"*** API to the client app.

    NOTE

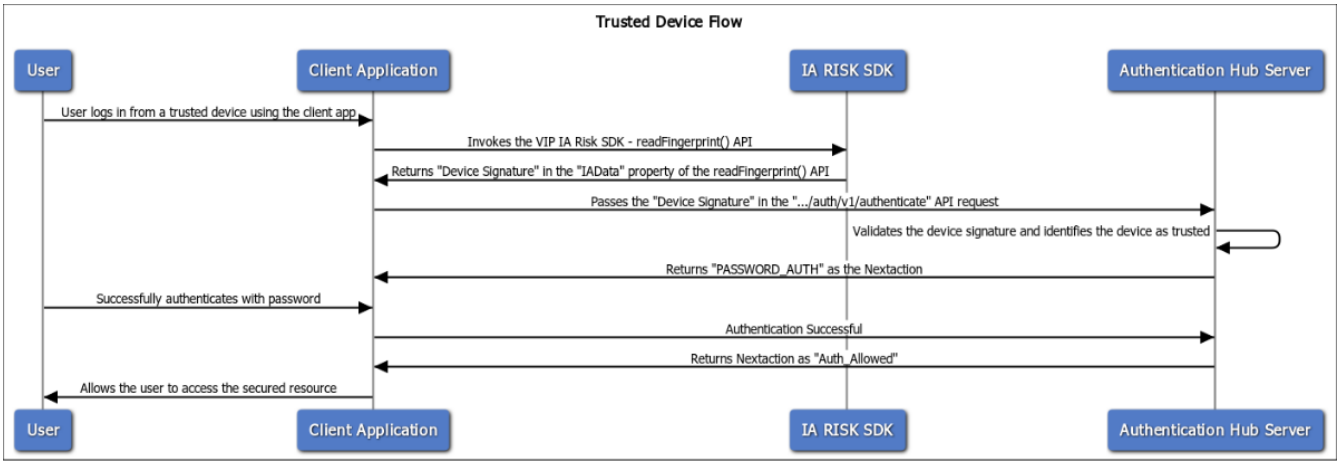
    If multiple authentication factors are configured for your authentication flow, then the "Remember This Device" option is displayed only before authenticating the last factor. For example, if you have factors in this order, Password>SMSOTP, then the "Remember This Device" checkbox gets displayed only before the SMSOTP authentication.
13. The client app renders the ***"Remember This Device"*** checkbox to the user.
14. The user receives OTP on email or phone and then enters the "OTP" in response to the Nextaction as SMSOTP returned by IDSP server previously (in step 10), and selects the ***"Remember This Device"*** checkbox.
15. Upon successful authentication by the user, the client app invokes the ***".../factor/v1/OTPVerifier"*** API and passes ***"remember-this-device"="true"*** and ***"x-device-tag"="device signature value"*** in the API header.
16. The IDSP server identifies the new device as a trusted device, returns nextaction as "Auth_Allowed" and allows the user to access the secured resource.

Trusted Device Flow

In this scenario, your device is already established as a trusted device, which streamlines the multi-factor authentication process and grants you access to secured resources immediately after fulfilling the first authentication factor.

List of participants and what they represent:

- ***User*** : One who initiates the authentication flow with a trusted device in IDSP.
- ***Client Application (Mobile/Web App)***: Represents the application that calls the VIP IA RISK SDK to obtain the device fingerprint to complete the authentication flow.
- ***IA SDK***: VIP IA RISK SDK that is called by the client application to update and get the device fingerprint.
- ***IDSP Server***: Server that is used to evaluate and authenticate the trusted device flow.



Assumption: In this flow we have assumed that the primary authentication factor is "***Password"***, while the secondary authentication factor is "***SMSOTP"***.

User Experience when you login from a Trusted Device

1. The user logs in to IDSP using a trusted device from the client app.

2. The client app invokes the VIP Risk SDK's "`IaDfp.readFingerprint()` " API to return the device fingerprint value in the response.
3. The VIP Risk SDK's "`IaDfp.readFingerprint()` " API returns the device fingerprint as part of the "iadata" property in the response.
4. The client app invokes the IDSP's ".../**auth/v1/authenticate"** API and passes the device fingerprint value in the API request.
5. IDSP server validates the device fingerprint and identifies it as a trusted device and returns "PASSWORD_AUTH" as the Nextaction in the ".../**auth/v1/authenticate"** API to the client app.
6. Using the client app, the user successfully authenticates with password.
7. IDSP server returns Nextaction as "Auth_Allowed" and allows the user to access the secured resource without prompting any other multi-factor-authentication.

> **NOTE**
> IDSP supports the transmission of "Device Signature" and "X-Device-Tag" values in an encrypted format. To learn how to encrypt these device signature and x-device-tag values, see Device Signature Encryption .

Steps to enable "TrustedDevices"

To enable "TrustedDevices", you must update configurations in the Tenant Config, update the Authentication Policy, and Invoking the Initiate Authenticate Call.

Updating Tenant Configuration

1. Use the following PATCH call to enable "TrustedDevices":
   `https://${SSP_FQDN}/{{tenantName}}/admin/v1/Configs`
2. Enable the following parameters in the PATCH request:
   - ***isTrustedDeviceEnabled***- This is a boolean value and the default is false.
   - ***maxTrustedDevicesPerUser***- This is an integer and the default value is 10. Possible values - 1 to 50.
   - ***trustedDeviceExpiryDays***- This is an integer and the default value is 365 days. Possible values - Valid values range from 0.0 upward with no restrictions on upper limits.

   The following is a patch request sample:

```
[
    {
     "name": "isTrustedDeviceEnabled",
     "value": "true"
    },
    {
      "name": "maxTrustedDevicesPerUser",
      "value": "10"
    },
   {
     "name": "trustedDeviceExpiryDays",
     "value": "365"
   }
]
```

   The parameter gets updated in the Tenant Configs.

These tenant configurations are also covered in Managing Tenant Settings.

Updating Authentication Policy

1. Use the following PUT/PATCH call to enable "TrustedDevices":
   `https://${SSP_FQDN}/{{tenantName}}/admin/v1/AuthPolicies/{{policyId}}`
2. Update the following parameter in the policy ***results***:

– ***mfaFrequency:OnceForTrustedDevice***

This enables the "**Remember This Device**" checkbox in the UI.

To know how to patch a policy, click here and, to know about the Authentication policy, click here.

Invoking the Initiate Authenticate Call

As discussed earlier in the notes section of this topic, if multiple authentication factors are configured for your authentication flow, then the "Remember This Device" option is displayed only before authenticating the last factor.

For example, if you have factors in this order, **Password > SMSOTP**, then the "Remember This Device" checkbox gets displayed only before the SMSOTP authentication.

Taking the same example into account, the steps that you must follow once you authenticate your last factor (SMSOTP) have been explained here.

Steps to update "**Remember This Device**":

1. Invoke the following OTPGenerator call:

   ```
   https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPGenerator
   ```

   The response fetches **"isTrustedDeviceEnabled": true** under the "**additional**" section as shown in the following sample:

   ```
   {
       "flowState": "eyJya....",
       "nextaction": "EMAIL_OTP_AUTH",
       "currentFactors": [
           "EMAILOTP"
       ],
       "data": {
           "credId": "206ed729-b9a2-4c91-a40a-81f2f6bbe782",
           "message": "030844"
       },
       "additional": {
           "currentFactorLevel": 2,
           "isTrustedDeviceEnabled": true
       }
   }
   ```

2. Use the following OTPVerifier call, and add the **"remember-this-device"** and **"x-device-tag"** parameters in the API header:

   ```
   https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPVerifier
   ```

   The response generates the "nextaction: AUTH_ALLOWED" as shown in the following sample:

   ```
   {
       "flowState": "eyJya....",
       "nextaction": "AUTH_ALLOWED",
       "data": {
           "status": "OK",
           "message": "One-Time Password (OTP) is verified successfully"
       },
       "id_token": "eyJ4......",
       "additional": {
           "authLevelsData": {
               "0988e396-089f-4c70-936f-d0ba27467a7b+craiglink+2": "SMSOTP",
               "0988e396-089f-4c70-936f-d0ba27467a7b+craiglink+1": "PASSWORD"
           },
           "currentFactorLevel": 2
   ```

```
        }
    }
```

To know more about Initiate Authenticate call, click here.

APIs to Manage Trusted Devices

Trusted Devices can be managed by Admins and Users.

Admin APIs to Manage Trusted Devices

The following table displays the list of Trusted Device APIs that an admin can invoke to manage user devices. For sample requests and response payloads of all these URLs, refer to the reference link provided at the end of the page for Trusted Devices API Swagger documentation.

Operation Description	Operation Type	Endpoint URLs
Retrieve all trusted devices of a user	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/TrustedDevices/{userId}
Retrieve the user's trusted device details	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/TrustedDevices/{userId}/{deviceId}
Delete trusted devices of a user	DELETE	https://${SSP_FQDN}/{{tenantName}}/admin/v1/TrustedDevices/{userId}/{{deviceId}}

User APIs to Manage Trusted Devices

The following table displays the list of Trusted Device APIs that a user can invoke to manage their own devices.

Operation Description	Operation Type	Endpoint URLs
Retrieve all my trusted devices	GET	https://${SSP_FQDN}/{{tenantName}}/auth/v1/MeTrustedDevices
Retrieve my trusted device details	GET	https://${SSP_FQDN}/{{tenantName}}/auth/v1/MeTrustedDevices/{deviceId}
Delete my trusted device	DELETE	https://${SSP_FQDN}/{{tenantName}}/auth/v1/MeTrustedDevices/{deviceId}

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/MeTrustedDevices`
  should look similar to:
  `https://example.com/default/admin/v1/MeTrustedDevices`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Examples

Getting Trusted Devices of a user

This section explains how to fetch all trusted devices for a specific user using their User ID.

Invoke this API to get all known devices for a user:

Request

```
GET https://${SSP_FQDN}/{{tenantName}}/auth/v1/MeTrustedDevices
```

Response

```
[
    {
        "userTrustDeviceId": "4641e562-93dd-4527-b6fb-a3fc541aacfd",
        "name": "Device:Mac, Browser:Chrome 144.0.0.0, OS:macOS 10.15.7",
        "status": "Active",
        "createdDatetime": "2026-02-20 06:45 AM UTC",
        "lastUsed": "2026-02-20 06:45 AM UTC",
        "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/144.0.0.0 Safari/537.36"
    }
]
```

Reference Links to:

- Trusted Devices API Error Codes
- Encryption of Device Signature and X-Device-Tag
- Swagger Documentation for Trusted Devices and Branded Settings.
- About SignIn User Interface

Encrypting the Device Signature

In the Symantec Identity Security Platform, Device Signature can be encrypted. The steps to encrypt the device signature are discussed in the following section.

Device signature encryption involves the utilization of a distinctive cryptographic signature, derived from device-specific characteristics, to safeguard customer device signatures during transmission, ensuring their confidentiality between the customer's mobile application or browser and the IT system where signature verification takes place. Encryption of Device Signature is implemented in IDSP as part of the ***Trusted Devices*** functionality.

To secure the device signature with encryption during the authentication flow, perform the following steps:

1. Configure the "***fieldsToBeEncryptedInAuthFlow***" branding settings value in the IDSP by invoking the following API:
   `hhttps://${SSP_FQDN}/<tenantName>/admin/v1/Configs`

   NOTE

   - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
   - The default value for `<tenantName>` is "default".
   - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
     For example:
     `https://{{host}}/{{tenantName}}/admin/v1/Configs`
     should look similar to:

```
https://example.com/default/admin/v1/Configs
```

- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

NOTE

1. The relevant configuration settings are covered in Global/Tenant Settings.
2. The tenant administrator can override the global settings using tenant settings.

Property Name	Property Scope	Property Description	Default Value	Comments
fieldsToBeEncryptedInAuthFlow	Global/Tenant/Branding Settings	• This property lists the fields that must be encrypted in the authentication flow. • As part of the branding setting, it helps the SignIn UI to decide if the device signature must be encrypted.	None	Possible values: devicesignature

2. Launch the application and generate the client Access Token.

   NOTE

   If you use the JavaScript (Web) SDK, Android SDK, and iOS SDK, which is provided out-of-the-box by the solution, then the SDK performs steps 3, and 4 automatically. If you do not use the SDK, then follow steps 3, and 4 as explained below.

3. Invoke the POST `https://${SSP_FQDN}/<tenantName>/auth/v1/NonceGenerator` API to get the server nonce by passing the encrypted client_nonce as input:

   NOTE

   If you have downloaded the JavaScript SDK, Android SDK, or iOS SDK, in step 1, then the SDK generates a client_nonce and stores it in the client app (browser or mobile) for the session duration. The client_nonce is encrypted using the IDSP public key. See SDK for more details.

```
Authorization: <client Access Token>

Payload:
{
    "clientNonce": "<client_nonce encrypted with AH public key>"
}
```

   The API response generates a server_nonce and server_nonce_id and stores them in the cache. server_nonce is encrypted (AES256) with client_nonce as shown here:

```
{
    "serverNonceId": "<server_nonce>"
    "serverNonce": "<server_nonce encrypted with client_nonce>"
}
```

4. ***Initiate the Authentication Flow***

   a. Initiate the `https://${SSP_FQDN}/<tenantName>/auth/v1/authenticate` API by passing the userLogin and authContext information:

   - userLogin
   - userContext (Device Signature)

```
POST /auth/v1/authenticate
Headers
    Authorization: <client Access Token>
```

```
Payload:
{
    "subject": "<userLoginid>",
    "channel": "web",
    "ipAddress": "10.10.123.124",
    "action":"authenticate",
    "rememberMe" : true,
    "device": {
        "signature": {
            "iaAuthData": "<Encrypted device signature>"
        }
    },
    "authContext": [
        {
            "key" : "nonceId",
            "value" : "<serverNonceId>"
        }
    ]
}
```

b. Internally, the authentication process is initiated and a flow state is generated for the authentication flow. The server_nonce is stored as part of the flow state and is used for validating the encrypted device signature that is sent by the client. The Flow State Id returns the authentication obligations to the client app (browser or mobile app).

5. Invoke the "***encryptWithSymmetricKey***" API from the IDSP SDK (iOS/Android/Web) to encrypt the ***x-device tag*** value with the symmetric key. However, this API is used to encrypt any sensitive data using symmetric encryption based upon server nonce

Configuring SoftMFA

SoftMFA is a setting available on an application resource that makes the multi-factor authentication (MFA) process more flexible. SoftMFA allows users without complete MFA registrations to still authenticate successfully. It skips unregistered factors instead of failing the login attempt.

When the **SoftMFAEnabled** setting is set to true, the authentication flow skips any required authentication factor if the user does not have that factor registered with the Symantec Identity Security Platform.

If the user successfully completes the available registered factors at each authentication level, the system returns an AUTH_SUCCESS response, even if some unregistered factors were originally required.

Additionally, when SoftMFA is enabled, the application's **isInlineEnrollmentEnabled** setting is ignored — meaning users are not prompted to enroll missing authentication methods during login.

> **NOTE**
> It applies across all authentication levels.

See Managing Applications for application configurations and Settings for global and tenant-level setting descriptions.

Examples of SoftMFA scenarios

Example 1: Skipping first-factor authentication

- The Authentication policy returned the following obligations:`FIDO:1, SMSOTP:1, EMAILOTP:2` where,

- FIDO (Biometric) - Level 1
- SMSOTP (SMS One-Time Password) - Level 1
- EMAILOTP (Email One-Time Password) - Level 2

User's situation:
The user has Biometric registration but has no phone or email registered for OTP.
With **SoftMFAEnabled** set to **true**: The user completes the FIDO authentication. Since they do not have registered SMSOTP and EMAILOTP methods, the second level is skipped. The authentication flow returns AUTH_SUCCESS after the FIDO authentication.

Example 2: Skipping second-factor authentication

- The Authentication policy returned the following obligations:`PWD:1, SMSOTP:2, EMAILOTP:2`
  where,
  - PWD (Password) - Level 1
  - SMSOTP (SMS One-Time Password) - Level 2
  - EMAILOTP (Email One-Time Password) - Level 2

  User's situation:
  The user has a password but has no phone or email registered for OTP.
  With **SoftMFAEnabled** set to **true**: The user completes the password (PWD) authentication. Since they do not have registered SMSOTP and EMAILOTP methods, the second level is skipped. The authentication flow returns AUTH_SUCCESS after the password authentication.

Table shows the SoftMFA Use Cases

The following table shows how enabling SoftMFA in the solution influence the Authentication Flow:

NOTE
In the following table,

- **Complete Delegation** refers to a scenario where all obligations are carried out in IDSP.
- **Partial Delegation** refers to a scenario where the first obligation is carried out in Siteminder, while the other obligations are carried out in the IDSP.
- **"Yes" or "No"** in the table columns indicates that the **"Final Obligation"** and "**Behavior"** remain unchanged, regardless of the selection.

	Delegation of Authentication in IDSP	Obligations returned by Policy	SoftMFA Enabled	Second Factor Registered	Trusted Device Enabled	Auth initiated from a Trusted Device	Final Obligations	Behavior
1	Complete Delegation	PWD:1, SMSOTP: 2	Yes	Yes	Yes	Yes	PWD	SMSOTP is ignored because the authentication has been initiated from a trusted device.
2	Complete Delegation	PWD:1, SMSOTP:2	Yes	Yes	Yes/No	No	PWD, SMSOTP	As the authentication request is initiated from a non trusted device, all factors returned by the policy will be considered for authentication.

	Delegation of Authentication in IDSP	Obligations returned by Policy	SoftMFA Enabled	Second Factor Registered	Trusted Device Enabled	Auth initiated from a Trusted Device	Final Obligations	Behavior
3	Complete Delegation	PWD:1, SMSOTP:2	Yes	No	Yes/No	Yes/No	PWD	SMSOTP:2 is ignored because it is not registered in the solution
4	Complete Delegation	PWD:1, SMSOTP:2	No	Yes/No	Yes	No	PWD, SMSOTP	When SoftMFA is disabled and the authentication request is initiated from a non trusted device, all factors returned by the policy will be considered for authentication.
5	Complete Delegation	PWD:1, SMSOTP:2	No	Yes/No	Yes	Yes	PWD	SMSOTP is ignored because the authentication has been initiated from a trusted device.
6	Complete Delegation	PWD:1, SMSOTP:2	No	Yes/No	No	NA	PWD, SMSOTP	When SoftMFA is disabled, all factors returned by the policy will be considered for authentication.
7	Partial Delegation	SMSOTP:2	Yes	Yes	Yes	Yes	NONE	AUTH_SUCCESS is returned by IDSP without any authentication as the authentication has been initiated from a trusted device.
8	Partial Delegation	SMSOTP:2	Yes	Yes	Yes/No	No	SMSOTP	SMSOTP is performed as the authentication has not been initiated from a trusted device.
9	Partial Delegation	SMSOTP:2	Yes	No	Yes/No	Yes/No	NONE	AUTH_SUCCESS is returned by IDSP without any authentication as SoftMFA is enabled and the user does not have any second factor registered
10	Partial Delegation	SMSOTP:2	No	Yes/No	Yes	No	SMSOTP	When SoftMFA is disabled and auth request is from a non trusted device, all factors returned by the policy will be considered for authentication.

	Delegation of Authentication in IDSP	Obligations returned by Policy	SoftMFA Enabled	Second Factor Registered	Trusted Device Enabled	Auth initiated from a Trusted Device	Final Obligations	Behavior
11	Partial Delegation	SMSOTP:2	No	Yes/No	Yes	Yes	NONE	AUTH_SUCCESS is returned by IDSP without any authentication as the authentication has been initiated from a trusted device.
12	Partial Delegation	SMSOTP:2	No	Yes/No	No	NA	SMSOTP	When SoftMFA is disabled, all factors returned by the policy will be considered for authentication when the Trusted device is disabled.
13	Complete Delegation	SMSOTP:1, PUSH:2	Yes	Both SMSOTP and PUSH are not registered for the user	Yes/No	Yes/No	-	AUTH_DENIED is returned by IDSP as there are no registered factors and softMFA is enabled.
14	Partial Delegation	SMSOTP:1, PUSH:2	Yes	Both SMSOTP and PUSH are not registered for the user	Yes/No	Yes/No	-	AUTH_SUCCESS is returned by IDSP without any authentication as primary was done at SSO and softMFA is enabled.
15	Complete Delegation	PWD:1, SMSOTP:2, PUSH:3	Yes	SMSOTP cred is not registered. PUSH cred may or may not exist.	Yes/No	Yes/No	PWD:1	SMSOTP:2 is ignored because it is not registered in the solution. Only PWD is performed.
16	Partial Delegation	SMSOTP:2	No	Yes/No	Yes/No	Yes/No	-	SMSOTP:2 → if SMSOTP is registered. AUTH_DENIED is returned by IDSP if SMSOTP is not registered.
17	Complete/Partial Delegation	PWD:1, SMSOTP:2, MOBILEOTP:3, PUSH:4	Yes	-	Yes/No	No	-	PWD:1 → if SMSOTP is not registered PWD:1, SMSOTP:2 → if MOBILEOTP is not registered PWD:1, SMSOTP:2, MOBILEOTP:3 → if PUSH is not register

Advanced Configurations

This topic includes the following advanced configurations that can be applied in Symantec Identity Security Platform:

- Managing Custom Vanity Ingresses
- Using Custom CA for Intra Cluster SSL Communications
- Operating with Multiple Data Centers
- Dynamic Environment Variable Substitution

Creating Vanity URLs

Vanity URLs are fully qualified domain names (FQDN) that are uniquely associated with an Symantec Identity Security Platform tenant. Learn how to configure vanity URLs and assign them to tenants.

A tenant can be configured to have one or more vanity URLs. To support vanity host names, you must update the IDSP ingress routing rules. The IDSP (ssp) chart manages the list of vanity hosts to ensure the ingress definition properly routes traffic.

Prerequisites

- You have deployed the solution in your Kubernetes cluster and exposed the main FQDN and System FQDN through an ingress controller.
- You have access to a system admin client to invoke the `.../system/admin/v1/Tenants` API.
- The vanity host cannot be a subdomain of the main FQDN specified during deployment. For example, if the FQDN is `authub.example.com` , the vanity host cannot be `api.authub.example.com` .

Types of Vanity Hosts

IDSP shows two types of vanity hosts that can be created:

Profile	Description	Default Hostname	Example Vanity
Full (main)	Exposes all IDSP paths, including admin, end-user, and API and routes them to the corresponding services.	`ssp.example.com`	`iam.acme.com`
End-User Vanity	Restricts access to end-user login portals only (sign-in, self-service, OIDC). Keeps admin APIs inaccessible.	*(none by default)*	`login.acme.com`

Tenant Configuration

The following step assumes that you have deployed IDSP (ssp) in your Kubernetes cluster and exposed the IDSP (ssp) FQDN and the IDSP (ssp) System FQDN through an ingress controller.

> **NOTE**
> The vanity hostname cannot be a subdomain of the IDSP FQDN that was specified during deployment. For example, if the FQDN of IDSP is "authub.example.com", the vanity host cannot be "api.authub.example.com".

To enable a vanity URL, you must register the vanity hostname with the tenant using the System Admin API so that the system can recognize and route requests for that hostname.

> **NOTE**
> If you omit this step, the router rejects requests arriving on the vanity hostname because the hostname is not associated with an active tenant.

Note that you must also configure the Helm chart to provision the underlying infrastructure (covered in the Helm Chart Configuration section).

For Versions 4.0 and 4.0.1

Using a system admin client, invoke the `.../system/admin/v1/Tenants` API to create or patch a tenant to add a vanity FQDN in the `vanityHosts` attribute.

When creating a new tenant (4.0 and 4.0.1): Include `vanityHosts` in the POST request:

```
curl -X POST "https://<MGMT_HOST>/system/admin/v1/Tenants" \
  -H "Authorization: Bearer <SYSTEM_ADMIN_TOKEN>" \
  -H "Content-Type: application/json" \
  -d '{
    "tenantName": "<name>",
    "masterKey": "<encryption-key>",
    "demoSetup": false,
    "defaultUserSetup": true,
    "defaultClientSetup": true,
    "vanityHosts": [
        "login.example.com",
        "iam.example.com"
    ]
}'
```

When updating an existing tenant (4.0 and 4.0.1): Use a PATCH request to add vanity hosts:

```
curl -X PATCH "https://<MGMT_HOST>/system/admin/v1/Tenants/<TENANT_ID>" \
  -H "Authorization: Bearer <SYSTEM_ADMIN_TOKEN>" \
  -H "Content-Type: application/json" \
  -d '{
    "vanityHosts": [
        "login.example.com",
        "iam.example.com"
    ]
}'
```

For Version 4.0.2

For 4.0.2, use the `vanityHostsInfo` array instead of `vanityHosts` . You must provide the host FQDN and its type (`ENDUSER` or `FULL`) within the `vanityHostsInfo` attribute.

When creating a new tenant (4.0.2): Include `vanityHostsInfo` in the POST request:

```
curl -X POST "https://<MGMT_HOST>/system/admin/v1/Tenants" \
  -H "Authorization: Bearer <SYSTEM_ADMIN_TOKEN>" \
  -H "Content-Type: application/json" \
  -d '{
    "tenantName": "<name>",
    "masterKey": "<encryption-key>",
    "demoSetup": false,
    "defaultUserSetup": true,
    "defaultClientSetup": true,
    "vanityHostsInfo": [
        {
            "hostFqdn": "login.acme.com",
            "type": "ENDUSER"
        }
```

```
    ]
}'
```

When updating an existing tenant (4.0.2): Use a PATCH request to add vanity hosts:

```
curl -X PATCH "https://<MGMT_HOST>/system/admin/v1/Tenants/<TENANT_ID>" \
  -H "Authorization: Bearer <SYSTEM_ADMIN_TOKEN>" \
  -H "Content-Type: application/json" \
  -d '{
    "vanityHostsInfo": [
        {
            "hostFqdn": "login.acme.com",
            "type": "ENDUSER",
            "allowScim": true
        },
        {
            "hostFqdn": "iam.acme.com",
            "type": "FULL"
        }
    ]
}'
```

The `type` field corresponds to the chart configuration: `ENDUSER` maps to end-user vanity hosts, and `FULL` maps to full vanity hosts.

Helm Chart Configuration

> **NOTE**
> **TLS Certificate Requirements:** Each vanity host requires a valid TLS certificate. You can configure TLS using one of the following methods:
>
> - **Per-host certificate:** Specify the `tlsSecretName` for each vanity host entry.
> - **Shared wildcard certificate:** Use a single wildcard certificate if multiple vanity hosts share a common domain (e.g., `*.acme.com`).
> - **Main certificate:** Set `tlsSecretName: null` to reuse the main ingress TLS certificate, provided it covers the vanity domain via Subject Alternative Name (SAN).

To configure vanity hosts, update the corresponding settings in your `values.yaml` (or `ssp-override.yaml`) file and upgrade your Helm release.

For Versions 4.0 and 4.0.1

Follow these steps to create routing rules for the vanity ingress:

1. Add the new vanity FQDN to the deployment ingress configuration. Get your current `values.yaml` by running the following command:

   ```
   helm get values ${RELEASENAME} -n ${NAMESPACE} >ssp-override.yaml
   ```

   > **NOTE**
   > Ensure to set the `RELEASENAME` and `NAMESPACE` environment variables prior to running this command. To get those values, use: `helm ls --all-namespaces`

2. Edit the `ssp-override.yaml` file, and modify the `ssp.ingress` section to add `fullVanity` and/or `endUsersVanity` URLs as shown in this example:

   ```
   ssp:
     ingress:
       fullVanity:
         enabled: true
         hosts:
   ```

```
    - host: iam.democorp.com
      tlsSecretName: iam-cert
    - host: iam.acme.com
      tlsSecretName: null
  endUsersVanity:
    enabled: true
    ingressClassName:
    additionalLabels: {}
    additionalAnnotations: {}
    allowScim: false
    hosts:
    - host: abc.example.com
      tlsSecretName: abc-cert
    - host: login.example.com
      tlsSecretName: null
```

In this example:

- Two full vanity FQDNs are created: `iam.democorp.com` uses its own TLS certificate from the Kubernetes secret `iam-cert` , and `iam.acme.com` sets `tlsSecretName: null` to reuse the main ingress TLS certificate (provided it covers the vanity domain via SAN). If you omit `tlsSecretName` , the chart uses the `ssp.ingress.tls.secretName` TLS secret.
- Two end-user vanity FQDNs are created: `abc.example.com` uses its own TLS certificate from the Kubernetes secret `abc-cert` , and `login.example.com` reuses the main ingress TLS certificate.
- `additionalLabels` and `additionalAnnotations` allow you to attach metadata to the end-user vanity ingress objects. `allowScim` controls whether SCIM provisioning paths are exposed on end-user vanity hosts.
- If you do not specify the `ingressClassName` , the chart uses the default `ssp.ingress.ingressClassName` value.
- If you use OpenShift Routes as the ingress type, depending on your ingress controller configuration, you may need to use labels to distinguish between different ingress controllers. See Using Additional Labels.

For OpenShift Routes, you can also specify `additionalLabels` under the ingress sections to target specific internal or external routers.

3. To upgrade the helm chart after adding the new vanity FQDN, run this command:

```
helm upgrade ${RELEASENAME} ssp_helm_charts/ssp -n ${NAMESPACE}  -f ssp-override.yaml \
--set ssp.featureFlags.dataseed.enabled=false \
--version=<current-ssp-version>
```

For Version 4.0.2 (Gateway API Support)

Follow these steps to create routing rules for the vanity ingress:

1. Add the new vanity FQDN to the deployment ingress configuration. Get your current `values.yaml` by running the following command:

```
helm get values ${RELEASENAME} -n ${NAMESPACE} >ssp-override.yaml
```

NOTE
Ensure to set the `RELEASENAME` and `NAMESPACE` environment variables prior to running this command. To get those values, use: `helm ls --all-namespaces`

2. Edit the `ssp-override.yaml` file, and modify the `ssp.ingress` section to add `fullVanity` and/or `endUsersVanity` URLs as shown in this example:

```
ssp:
  ingress:
    type: gatewayapi
    # Top-level Gateway API configuration (shared defaults for all endpoints)
    gatewayApi:
```

```
  gatewayType: auto                    # auto, envoy, gke-l7, azure-agc, aws-alb, avi, custom
  createGateway: true
  existingGateway: ""
  existingGatewayNamespace: ""
  gatewayClassName: ""
  infrastructureRef: {}
  additionalListeners: []
  httpToHttpsRedirect: "auto"          # "auto", true, false
  skipWildcardListener: false
  allowCrossNamespaceCreate: false
  createReferenceGrant: true
  createBackendTLSPolicy: true
  backendRef: {}
  additionalLabels: {}
  envoy:
    connectionBufferLimitBytes: 524288
    maxRequestBodySize: "50Mi"
  avi:
    sslProfile: {}
  gatewayAnnotations: {}
  addresses: []
  httpRouteAnnotations: {}

fullVanity:
  enabled: true
  hosts:
  - host: iam.acme.com
    tlsSecretName: acme-cert

endUsersVanity:
  enabled: true
  ingressClassName:
  gatewayApi:
    createGateway: false               # false = reuse main Gateway (default)
    existingGateway: ""                # name of a pre-existing Gateway
    existingGatewayNamespace: ""       # defaults to Release namespace
    gatewayClassName: ""               # defaults to main gatewayApi value
    createTrafficPolicies: false       # manage policies for an existingGateway
    gatewayAnnotations: {}
    addresses: []                      # static IPs (Envoy Gateway, Istio, Contour, Cilium)
    httpRouteAnnotations: {}
    additionalLabels: {}
  additionalLabels: {}
  allowScim: false
  hosts:
  - host: login.acme.com
    tlsSecretName: acme-cert

mgmt:
  enabled: true
  ingressClassName:
  gatewayApi:
    createGateway: false
```

```
    existingGateway: ""
    existingGatewayNamespace: ""
    gatewayClassName: ""
    createTrafficPolicies: false
    gatewayAnnotations: {}
    addresses: []
    httpRouteAnnotations: {}
    additionalLabels: {}
  additionalLabels: {}
  host: mgmt-ssp.example.com
  tls:
    enabled: true
    secretName:
    host:
    generateSelfSignedCert: false
```

What the Chart Creates:

Based on your `values.yaml` settings, the chart provisions the following resources:

Configuration	Gateway	HTTPRoute	Listeners
`endUsersVanity.enabled` + `createGateway: false` + no `existingGateway`	Uses main Gateway	Separate HTTPRoute (`*-router-enduser`)	Adds listeners to main Gateway
`endUsersVanity.enabled` + `createGateway: false` + `existingGateway: xyz`	Uses `xyz` Gateway	Separate HTTPRoute	Adds listeners to `xyz` Gateway
`endUsersVanity.enabled` + `createGateway: true`	Creates `*-gateway-enduser`	Separate HTTPRoute	Own listeners on separate LB
`mgmt.enabled` + `createGateway: false` + no `existingGateway`	Uses main Gateway	Separate HTTPRoute (`*-router-mgmt`)	Adds listener to main Gateway
`mgmt.enabled` + `createGateway: false` + `existingGateway: xyz`	Uses `xyz` Gateway	Separate HTTPRoute	Adds listeners to `xyz` Gateway
`mgmt.enabled` + `createGateway: true`	Creates `*-gateway-mgmt`	Separate HTTPRoute	Own listeners on separate LB
`fullVanity.enabled`	Uses main Gateway	Uses main HTTPRoute (`*-router-full`)	Adds listeners to main Gateway

Summary of Configuration Differences Between Versions 4.0/4.0.1 and 4.0.2:

- Version 4.0.2 introduces `gatewayapi` as new values for `ssp.ingress.type`, alongside the existing `nginx` and `openshiftroute` types.
- A new top-level `ssp.ingress.gatewayApi` section defines shared Gateway API configuration, including gateway type detection (`gatewayType`), Gateway resource creation (`createGateway`), `gatewayClassName`, backend TLS policies (`createBackendTLSPolicy`), backend references (`backendRef`), and controller-specific settings for Envoy (`connectionBufferLimitBytes`, `maxRequestBodySize`) and AVI (`sslProfile`).
- `endUsersVanity` gains a `gatewayApi` sub-section that supports network isolation through separate Gateways (`createGateway`), pre-existing Gateways (`existingGateway`, `existingGatewayNamespace`), traffic policies (`createTrafficPolicies`), static addressing (`addresses`), and HTTPRoute annotations

(`httpRouteAnnotations`). The `additionalAnnotations` parameter at the `endUsersVanity` level is removed in 4.0.2.

- `mgmt` gains the same `gatewayApi` sub-section as `endUsersVanity` , and a new `tls.generateSelfSignedCert` parameter.

Validating the Vanity Ingress Routing Rules

To verify the vanity ingress configuration, access the OpenID configuration discovery endpoint. A successful response confirms that the ingress is routing traffic correctly.

Sample Request

```
GET https://my.vanity.fqdn/common/.well-known/openid-configuration
{
    "issuer": "https://my.vanity.fqdn/common/",
    "authorization_endpoint": "https://my.vanity.fqdn/common/oauth2/v1/authorize",
    "device_authorization_endpoint": "https://my.vanity.fqdn/common/oauth2/v1/devicecode",
    "token_endpoint": "https://my.vanity.fqdn/common/oauth2/v1/token",
    "userinfo_endpoint": "https://my.vanity.fqdn/common/oauth2/v1/userinfo",
    "revocation_endpoint": "https://my.vanity.fqdn/common/oauth2/v1/revoke",
    "jwks_uri": "https://my.vanity.fqdn/common/oauth2/v1/jwks",
    "response_types_supported": [
        "code",
        "token",
        "id_token",
        "id_token token"
    ],
    "subject_types_supported": [
        "public"
    ],
    "id_token_signing_alg_values_supported": [
        "RS256"
    ],
    "scopes_supported": [
        "openid",
        "profile",
        "email",
        "phone",
        "address",
        "groups"
    ],
    "token_endpoint_auth_methods_supported": [
        "client_secret_post",
        "client_secret_basic",
        "private_key_jwt"
    ],
    "claims_supported": [
        "aud",
        "email",
        "email_verified",
        "phone_number",
        "phone_number_verified",
        "family_name",
        "middle_name",
```

```
        "given_name",
        "iss",
        "iat",
        "name",
        "picture",
        "sub",
        "locale",
        "nickname",
        "preferred_username",
        "profile",
        "website",
        "gender",
        "zoneinfo",
        "birthdate",
        "updated_at",
        "address",
        "sub",
        "groups"
    ],
    "response_modes_supported": [
        "fragment",
        "query"
    ],
    "grant_types_supported": [
        "authorization_code",
        "implicit",
        "urn:ietf:params:oauth:grant-type:jwt-bearer",
        "client_credentials",
        "refresh_token",
        "urn:ietf:params:oauth:grant-type:device_code"
    ],
    "id_token_encryption_alg_values_supported": [
        "RSA-OAEP",
        "RSA-OAEP-256"
    ],
    "id_token_encryption_enc_values_supported": [
        "A128GCM",
        "A192GCM",
        "A256GCM",
        "A192CBC-HS384",
        "A128CBC-HS256"
    ],
    "userinfo_signing_alg_values_supported": [
        "RS256"
    ],
    "userinfo_encryption_alg_values_supported": [
        "RSA-OAEP",
        "RSA-OAEP-256"
    ],
    "userinfo_encryption_enc_values_supported": [
        "A128GCM",
        "A192GCM",
        "A256GCM",
```

```
        "A192CBC-HS384",
        "A128CBC-HS256"
    ],
    "code_challenge_methods_supported": [
        "plain",
        "S256"
    ],
    "claims_parameter_supported": false,
    "request_parameter_supported": false,
    "request_uri_parameter_supported": false,
    "require_request_uri_registration": false
}
```

Using Custom CA for Intra Cluster SSL Communications

Use of Internal Signing Key (ISK) for Internal Cluster Communications

All internal cluster communications are authenticated using an Access Token signed by the Internal Signing Key (ISK). The ISK is also used for intra-cluster SSL communication.

Symantec Identity Security Platform automatically generates an ISK during its deployment. If you want to use your own certificate as the ISK, follow these steps to create a secret and pass it in the IDSP deployment:

Before creating a secret, ensure that;

Your certificate has the Subject Alternative Name (SAN) that looks like the following:

```
*.<k8s-namespace>.svc.
```

For example, assuming that the Kubernetes namespace = 'dev', the SAN should look like the following:

```
X509v3 Subject Alternative Name: critical
    DNS:*.dev.svc
```

NOTE

- The SAN in the certificate should match the proxy-ssl-name specified in the IDSP Ingress objects annotations:

  ```
  "http://nginx.ingress.kubernetes.io/proxy-ssl-name">nginx.ingress.kubernetes.io/proxy-ssl-name:
   any.{{ .Release.Namespace }}.svc
  ```

- The wildcarded SAN in the certificate is expected to match the "proxy-ssl-name" used by the IDSP Ingress:

  ```
  nginx.ingress.kubernetes.io/proxy-ssl-name: any.dev.svc
  ```

- Creating a secret requires the server certificate (tls.crt), server key (tls.key), and CA certificate (CA.crt).

1. Create a secret using the following command (this command assumes that the file names are: ca certificate is caroot.pem, isk certificate is isk-certificate.pem, and isk certificate key is isk-key.pem):

   ```
   kubectl create secret generic -n <namespace>  <isk-secretname> \
     --from-file=ca.crt=caroot.pem \
     --from-file=tls.crt=isk-certificate.pem \
     --from-file=tls.key=isk-key.pem
   ```

   NOTE

   - The isk-certificate.pem file may contain multiple intermediate certificates in PEM. It must have the server certificate first followed by intermediate certificates.
   - The isk-key.pem file must be in PKCS8 PEM format (starts with "-----BEGIN PRIVATE KEY-----" and ends with "-----END PRIVATE KEY-----")

2. Pass the secret that was created in the previous step when installing the IDSP helm chart deployment by adding the following syntax in the deployment command:

```
--set ssp.keys.isk.existingSecret=<k8s-tls-secret-name>
```

Reference Links to:

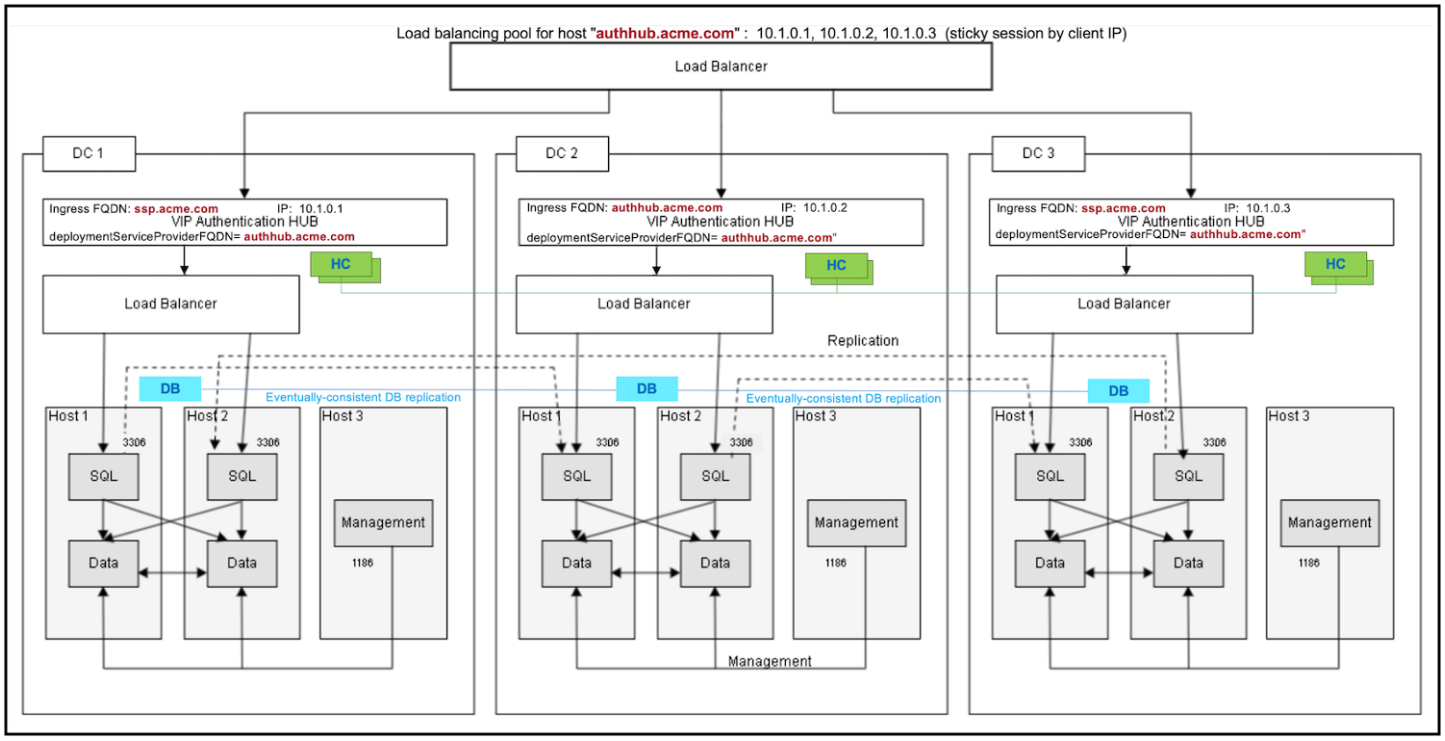
- Security Considerations See Infrastructure **>** Internal Cluster Communication

Operating with Multiple Data Centers

Advanced settings covering transactional artifacts

The following diagram illustrates the use of the Symantec Identity Security Platform with three data centers operating in Active/Active mode. Each data center has its own IDSP deployed with a dedicated MySQL cluster with InnoDB Storage Engine, integrated through eventually-consistent data replication.

Figure 9: Multi Data Center Image



NOTE
In this diagram, **HC** refers to Hazelcast Cache.

While operating with multiple data centers, the following advanced global settings have been added to IDSP to support configurable retries when reading transactional artifacts (such as Flow State, Az Code, Device Code, and User Code) that persist in eventually-consistent storage.

The following settings have to be used in the Multi Data Center mode:

- ***persistentStoreForTransactionalData*** - Note that this setting must be set to "db" for IDSP to be operational with multiple data centers. The default is "***cache***".
- ***transactionalDataReadsRetryCount***
- ***transactionalDataReadsRetryWaitPeriodMillis***

These configuration settings are covered in Global/Tenant Settings.

Dynamic Substitutions

This topic provides information on the supported dynamic environment variables in Symantec Identity Security Platform.

IDSP supports dynamic environment variables in various places. These environment variables are substituted with the appropriate value coming from different contexts.

This topic includes the following sections:

- Supported Dynamic Substitution Notations
- Supported Dynamic Variable Namespaces
- Supported Dynamic Substitution Functions
  - stringUtils Table
  - arrayUtils Table
- Usage of Dynamic Environment Substitutions

Supported Dynamic Substitution Notations

The dynamic substitution uses the following three types of notations:

- ${<namespace>.<param>} notation returns a single string, in case of multi-value attribute, it returns a csv string
- $(<namespace>.<param>) notation returns a single string, containing a csv wrapped with [] where every value in the csv is wrapped with double quotes (""), and a double quote inside the value is escaped by a backslash ("\").
- $[<namespace>.<param>] notation returns a list of strings.

NOTE

When a value is expected to return a list, you can choose the format in which the list is represented using either curly braces, square brackets, or parenthesis:
- $[...] returns a native list
- ${...} returns a CSV in a single element
- $(...) returns a CSV in a single element, wrapped with [ ] where:
  - every value in the csv is wrapped with double quotes (" ")
  - a double quote is escaped by a backslash ("\")

Examples: SAML Assertion:

Using $[...]

```
<saml2:Attribute Name="$[idProfile.email]" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
<saml2:AttributeValue xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:type="xsd:string">malbail17222@example.com</saml2:AttributeValue>
<saml2:AttributeValue xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:type="xsd:string">malbail17222@example.com</saml2:AttributeValue>
</saml2:Attribute>
```

Using ${...}

```
<saml2:Attribute Name="${idProfile.email}" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
<saml2:AttributeValue xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:type="xsd:string">malbail17222@example.com,malbail17222@example.com</saml2:AttributeValue>
</saml2:Attribute>
```

Using $(...)

```
<saml2:Attribute Name="$(idProfile.email)" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
```

```
<saml2:AttributeValue xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xsi:type="xsd:string">
["malbail17222@example.com","malbail17222@example.com"]</saml2:AttributeValue>
</saml2:Attribute>
```

Examples: JWT:

Using $[...]

```
"$[idProfile.email]": [
"malbail17222@example.com",
"malbail17222@example.com"
],
```

Using ${...}

```
"${idProfile.email}": "malbail17222@example.com,malbail17222@example.com",
```

Using $(...)

```
"$(idProfile.email)": "[\"malbail17222@example.com\",\"malbail17222@example.com\"]",
```

Supported Dynamic Variable Namespaces

The following table lists the supported variable namespaces that will be substituted with their corresponding values:

Namespace Identifier	Description	E
system.<*propertyName*>	Available property names: • issuerUrl - returns the issuer Url of the current request. • host - returns the host header variable of the current request. • newUuid - returns newly generated UUID. • thisResourceId - returns the ID of the subject-object of the request.	$ $
idProfile.<*logical-attribute*>	Returns the user profile attribute that is mapped with the indicated user profile logical attribute. If the attribute is not present in the ID Store or ZFP ID Token, the keyword is replaced with NODATA. If ZFP flow is used, the attribute is presumed to have been provided in the incoming ID Token.	$
rawAssertion.<*claim-name*> (Available from IDSP version-3.3)	Returns the value of a named claim from the Identity Assertion Token returned by the Identity Provider in a BYOI. If the ID Token is present in the flow but the claim is not present in the ID Token, the keyword is replaced with NODATA.	$
idToken.<*claim-name*>	Returns the value of a named claim from the original ID Token, if present in the flow. Otherwise, the key-value pair is skipped. If the ID Token is present in the flow but the claim is not present in the ID Token, the keyword is replaced with NODATA.	{
appMetadata.<*property-name*>	Returns the metadata property of the application for the specified property name. If the metadata property is not present, the keyword is replaced with NODATA.	$ v
idpMetadata.<*property-name*>	Returns the metadata property of the identityProvider for the specified property name. If the metadata property is not present, the keyword is replaced with NODATA.	$ v

clientContext.*<propertyName>*

Returns the clientContext value for the specified property name. If the client context property is not present, then the keyword is replaced with NODATA.

The `clientContext` can be provided through various methods:

- **Request Payload in Authenticate API**: The clientContext property can be passed into the ".../auth/v1/authenticate" call in the request payload. For example -

```
{
"subject": "nbruce",
"channel": "web",
"action": "authenticate",
"rememberMe": true,
"urn:iam:authn:clientContext": [
{
 "key": "email",
 "value": "user@example.com"
}
]

}
```

- **As a claim in the "id_token_hint" Parameter**: The clientContext property can be passed as a query parameter to ".../oauth2/v1/authorize" or can be passed into ".../auth/v1/authenticate" API call through the "existingIDToken" or "id_token_hint" payload parameters. The following example represents the decoded id_token_hint with clientContext claim:

```
{
"sub": "dd433ba1-b9ed-46bf-9d3e-15759d6bf89b",
"idp_name": "aa_idp",
"user_dn": "cn=user1,ou=Employees,o=democorp,c=us",
"amr": [
"PWD"
],
"user_loginid": "user1",
"idp_guid": "bcd084d4-6eb0-48bb-8d8b-25c19af9ec2e",
"iss": "https://www.example.com/default/",
"user_guid": "dd433ba1-b9ed-46bf-9d3e-15759d6bf89b",
"aal": "AAL1",
"tid": "3239b450-8c89-446f-81c3-70890815d2c9",
"transactionid": "dea78ba8-63a8-4454-8745-007b53dd0bd9",
"sid": "b8695bf1-ebd5-4c2f-807b-ac27bdd0e578",
"azp": "72316d69-9749-4ef7-a9bb-d92e433efe28",
"user_loginidattrmappingname": "user_loginid",
"auth_time": 1711336816,
"session_ttl_secs": 21600,
"exp": 1811358416,
"iat": 1711336816,
"client_name": "app1",
"jti": "c9fd1cb1-e5c3-4cd8-99ff-b66592799610",
"email": "user1@example.com",
"idp_type": "siteminder",
"client_tname": "default",
"client_tid": "3239b450-8c89-446f-81c3-70890815d2c9",
"ver": "1.0",
"user_universalid": "user1",
"user_riskscore": 100,
"auto_inline_enrollment": true,
"aud": [
"ceee9ad0-9414-422d-a264-179a85cbf4db",
"ad179147-fce0-4e09-a1cf-7830cfe3b4a3",
"72316d69-9749-4ef7-a9bb-d92e433efe28"
],
```

${requestParams.xxx}	This placeholder would be replaced with the value of a request parameter named xxx.	$
userGroups.all_groups	It returns the entire list of groups that the user is a member of. Returns a list or a csv depending on how it is invoked: ${userGroups.all_groups} returns a csv $[userGroups.all_groups] returns a list $(xxx.yyy) returns a csv wrapped with [] where: • every value in the csv is wrapped with double quotes ("") • a double quote is escaped by a backslash ("\")	$
userGroups.regex_groups	It is replaced by a filtered list of user's groups by applying a regex expression specified in application metadata. For OIDC (JWT): urn:iam:app:jwtGroupApplicabilityRegex For SAML: urn:iam:app:samlGroupApplicabilityRegex	$
userGroups.<*group_name*>	returns a boolean indicating if the user is a member of the specified group ${userGroups.<*groupname*>} returns true or false depending on whether the user is a member of the specified group.	$
• evaluateIsTrue.< expression *resulting in "true" / "false"*> • evaluateIsNotTrue.< expression *resulting in "true" / "false"*>	used to convert a string value to a boolean value "false" or "true" depending on the string value. A string value of "true", ignoring the case, causes "true" to be returned, otherwise "false" is returned.	"

Supported Dynamic Substitution Functions

The following table lists the supported functions (stringUtils and arrayUtils) that can be used with the substituted values mentioned in the previous table:

stringUtils Table

NOTE
String compares are case insensitive.

Functions	Description	Exa
${stringUtils.empty("string")	returns "true" if "string" is null or empty, otherwise returns "false"	${st
${stringUtils.notEmpty("string")}	returns "true" if "string" is not null and not empty, otherwise returns "false"	${st
${stringUtils.contains("string", "substring")	returns "true" if "string" contains "substring", otherwise returns "false"	${st
${stringUtils.notContains("string", "substring")}	returns "true" if "string" does not contain "substring", otherwise returns "false"	${st
${stringUtils.startsWith("string", "prefix-string")}	returns "true" if "string" starts with "prefix-string", otherwise returns "false"	${st
${stringUtils.notStartsWith("string", "prefix-string")}	returns "true" if "string" does not start with "prefix-string", otherwise returns "false"	${st
${stringUtils.endsWith("string", "suffix-string")}	returns "true" if "string" ends with "suffix-string", otherwise returns "false"	${st
${stringUtils.notEndsWith("string", "suffix-string")}	returns "true" if "string" does not end with "suffix-string", otherwise returns "false"	${st
${stringUtils.compare("string1", "string2")}	returns "true"/"false" if "string1" equals "string2"	${st

${stringUtils.if(“boolean-string”, “string-true”, “string-false”	returns “string-true” if “boolean-string” is “true”, otherwise returns “string-false”	${st “Er
${stringUtils.prefix(“string”, “substring”)}	returns the part of “string” that is before “substring” (not including “substring”)	${st
${stringUtils.suffix(“string”, “substring”)}	returns the part of “string” that is after “substring” (not including “substring”)	${st
${stringUtils.lower(“string”)}	returns “string” in lower case	${st
${stringUtils.upper(“string”)}	returns “string” in the upper case	${st
${stringUtils.toList(“csv-string”)}	parses a comma-separated value (CSV) string into a list of strings	
The following stringUtils functions are supported from IDSP v3.3.1 only.		
${stringUtils.maskEmail(“string”)}	returns masked email id	${st
${stringUtils.maskPhoneNumber(“string”)}	returns a masked phone number	${st
${stringUtils.maskString(“string”)}	returns masked string	${st
${stringUtils.guid("string")}	returns "true" if "string" is a valid UUID/GUID, otherwise returns "false"	${st
${stringUtils.notGuid("string")}	returns "true" if "string" is not a valid UUID/GUID, otherwise returns "false"	${st
${stringUtils.greaterThan("stringINT1", "stringINT2")}	returns "true" if the integer value of "stringINT1" > "stringINT2", otherwise "false"	${st
${stringUtils.lessThan("stringINT1", "stringINT2")}	returns "true" if the integer value of "stringINT1" < "stringINT2", otherwise "false"	${st
${stringUtils.or("bool1", "bool2")}	returns "true" if at least one of "bool1" or "bool2" is "true", otherwise "false"	${st ${st
${stringUtils.and("bool1", "bool2")}	returns "true" if both "bool1" and "bool2" are "true", otherwise "false"	${st ${st
${stringUtils.substring("string", startPos, endPos)}	returns substring from startPos (0-based, inclusive) to endPos (exclusive). Use negative startPos to count from end. Use endPos=-1 for rest of string. If endPos > length, returns to end.	${st
${stringUtils.quote("string")}	wraps string in quotes and escapes internal quotes. If already quoted, returns as-is.	${st
${stringUtils.unquote("string")}	removes surrounding quotes and unescapes. If not quoted, returns as-is.	${st

arrayUtils Table

NOTE
String compares are case insensitive.

Functions	Description
$(arrayUtils.filterStartsWith(“string-array”, “string”))	returns a csv with entries that do start with “string”
$(arrayUtils.filterNotStartsWith(“string-array”, “string”))	returns a csv with entries that do not start with “string”
$(arrayUtils.filterEndsWith(“string-array”, “string”))	returns a csv with entries that end with “string”
$(arrayUtils.filterNotEndsWith(“string-array”, “string”))	returns a csv with entries that do not end with “string”
$(arrayUtils.filterContainsSubString(“string-array”, “string”))	returns a csv with entries that contain the “string” as a substring of the

$(arrayUtils.filterNotContainsSubString(“string-array”, “string”))	returns a csv with entries that do not contain “string” as a substring of
${arrayUtils.at(“string-array”, <index>)}	returns the value in location <index> in the array (index=0 for first item range return “”
${arrayUtils.size(“string-array”)}	returns the number of elements in the array as a string
$(arrayUtils.filterByRegEx(“string-array”, “regex”))	returns a csv with entries that contain only the strings that match the r
$[arrayUtils.rewriteByRegEx(“string-array”, “regex”, "template")]	For each string in the array, apply the regex and rewrite the string usir In the template string, use $1 and $2 to indicate the regex group id to $(arrayUtils.rewriteByRegEx($[IdProfile.groups], "ad-(\w+)-(\d+)", "arn $2:role/$1"))"; or $(arrayUtils.rewriteByRegEx($[IdProfile.groups], "group_aws_(\d+)_(\ "arn:aws:iam::$1:role/$2"))
${arrayUtils.empty("string-array")}	returns "true" if array is null, empty, or "[]", otherwise "false"
${arrayUtils.notEmpty("string-array")}	returns "true" if array has at least one element, otherwise "false"
${arrayUtils.contains("string-array", "string")}	returns "true" if array contains exact match of "string" (case insensitive "false"
${arrayUtils.notContains("string-array", "string")}	returns "true" if array does not contain "string" (case insensitive), othe
${arrayUtils.filterByRegEx("string-array", "regex")}	returns csv of entries where the entire string matches the regex (case Java Pattern.matches().
$[arrayUtils.rewriteByRegEx("string-array", "regex", "replacement")]	applies String.replaceAll(regex, replacement) to each element. Suppo backreferences ($1, $2, etc).

NOTE

- filterByRegEx requires the entire string to match the pattern (anchored)
- rewriteByRegEx replaces all occurrences of the pattern in each string
- Use $[...] syntax for functions returning arrays (MVA), ${...} for scalar values

Where can Dynamic Environment Substitution be used?

Functional Area	Description
LDAPconfig	ldapQueryStart, ldapQueryEnd
SCIMconfig	lookupQuery
	requestCtxBase64, oauthTokenRequestCtxBase64 (the *value* field of requestCtxBase64 (or oauthTokenRequestCtxBase64) entries of *http-header*)
Both LDAPconfig and SCIMconfig	IdStoreMappingStrategy <value> will be "disambiguated" against the authentication id store (using idstore lo start/end search query) or <authn-id-store-attr>=<value> query against authentication idStore (on both the ids and the value)

provider config (Custom SPI)	providerHeaders and providerProperties
App	FlowURL This URL can use variables of the namespaces appMetadata, and idToken.
	errorUrl
	logoutUrl
	Application custom claim See Custom Token Claims.
Tenant Settings	InternalIdStoreDisambiguationQuery
Identity Provider	IDP Configuration

	IDP claim mapping Dynamic substitution using standard namespaces (and expressions as covered by Substitution Functions) is applicable to IDP's Subject Claim and Claim Mapping configuration.

Identity Service

Contains information about Managing Identities in the Internal Id Store using the Identity Service.

The section contains information on the Identity Service, the service capabilities, schemas, and the Identity Service API.

> **NOTE**
> Before you can manage identities, you must configure the Identity Service. For more information, see Internal Identity Store Configuration.

The Identity Service is a Business Service running in the Security Services Platform that manages identities and related capabilities. The Identity Service exposes REST APIs, including a SCIM interface that conforms to the SCIM v2 standard, for User and Group Management. The service operates on User and Groups maintained in the Internal ID Store. The service includes the following capabilities:

- Internal ID Store Configuration
- Full lifecycle management of Users and Groups
- Extensible User Schema
- Self Service Password Recovery
- Self Service Account Unlock
- User State Management
- Password Management including Password Policy support

User and Group Schemas

Describes the User and Group Schemas supported by Symantec Identity Security Platform.

The Identity Service supports the SCIM v2 User and Group core schema, the User Enterprise schema extension, the SSP-based Symantec Identity Security Platform User schema extension, and User custom schema extension that can be defined by deployments. You can also use the `UserSchemaHelper` parameter that supports filtering by schema fields.

The following table lists the defined schemas.

SCIM schema URN	Attribute Name	>Description	Value
`urn:ietf:params:scim:schemas:core:2.0:User`	active	Indicates the locked state of the user account	**True** – account is unlocked **False** – account is locked

	`addresses {`
	`country`
	`formatted`
	`locality`
	`postalCode`
	`primary`
	`region`
	`streetAddress`
	`type`
	`verified`
	`}`
		One or more address sets comprised of a composition of single value attributes representing the various address parts.
		country – users country
		formatted – The full mailing address, formatted for display or use with a mailing label.
		locality – users city/town
		postalCode – the postal code
		primary - a boolean indicating the primary address. Only one address can be marked as primary
		region – state or province
		streetAddress -= physical address
		type -a user-defined type (e.g. work or home)
		verified - a boolean SSP extension that is used to indicate the address has been verified	

	`emails {`
	`display`
	`primary`
	`priority`
	`type`
	`value`
	`verified`
	`}`
		One or more email sets comprised of a composition of single value attributes representing the various email parts.
		display – Value to use when displaying the email.
		primary – a Boolean indicating the primary email. Only one email can be marked as primary
		priority – an optional value indicating the priority. Used when the primary designation is not defined. The lowest priority value indicates primary.
		type – a user-defined type (e.g. work or home)
		value – the email value
		verified – a boolean SSP extension that is used to indicate the email has been verified
		Note: Only verified emails can be used as an Email Credential.	

SCIM schema URN	Attribute Name	>Description	Value
	displayName	The displayName of the user.	This value is also used in the list of Group members On a create User request, the Identity Service generates the `displayName` value from the `displayNameTemplate` configured in the internal identity store configuration (see Internal Identity Store Configuration). By default, the Display Name Template is set to `givenName` and `familyName` . If some or all of the values for the specified attributes are present, the display name is generated from the available values. If none of the values are present, the display name is generated from the user `Login` value.
	id	The unique SCIM resource id for the User	id is an immutable value that is generated by the Identity Service when a user is created (POST User) and which is returned in the POST response. Optionally, you can pass a unique value for the id into the Create User request. The value must be a unique GUID.
	groups	The list of groups the user is a member of	
	locale	Used to indicate the default location of the user for purposes of localizing such items as currency, date time format, or numerical representations.	

	`name {`
	`formatted`
	`familyName`
	`givenName`
	`middleName`
	`honorificPrefix`
	`honorificSuffix`
	`}`
		Name a composition of single value attributes representing the various name parts
		formatted - The full name, including all middle names, titles, and suffixes as appropriate, formatted for display
		familyName – family name or last name
		givenName – given name or first name
		middleName – middle name
		honorificPrefix- prefix or title
		honorificSufffix- suffix	

	nickName	The nickname of the user	

SCIM schema URN	Attribute Name	>Description	Value
	password	The password of the user	The request containing the user password can be clear text or in an in-transit encrypted format. See the section Password Management section for details.

	phoneNumbers {
	capabilityType
	display
	primary
	priority
	type
	value
	verified
	
	}
		One or more phoneNumber sets comprised of a composition of single value attributes representing the various phoneNumber parts.
		capabilityType– an SSP extension used to describe the capabilities that the phone has when used as a phone Credential.
		display – Value to use when displaying the phone.
		primary – a Boolean indicating the primary phone. Only one phone can be marked as primary
		priority – an optional value indicating the priority. Used when the primary designation is not defined. The lowest priority value indicates primary.
		type – a user-defined type (e.g. work or home)
		value – the phone number
		verified – a boolean SSP extension that is used to indicate the phone has been verified
		Note: Only verified phones can be used as a Phone Credential.
			capabilityType supported values: “sms” and “ivr”

	preferredLanguage	Indicates the preferred written or spoken languages of the user and is used for selecting a localized user interface.	
	profileUrl	The profile URL of the user	A URI that is a uniform resource locator (as defined in Section 1.1.3 of [RFC3986]) and that points to a location representing the online profile of the user (e.g., a web page).
	timezone	The time zone of the user, in IANA Time Zone database format	
	title	The title of the user	
	userName	The name of the user. Used for user login	The user name must be unique

SCIM schema URN	Attribute Name	>Description	Value
	userType	The user type. Typical values used might be "Contractor", "Employee", ”Partner” but any value may be used.	
urn:ietf:params:scim:schemas:extension:enterprise:2.0:User			
	costCenter	a value representing the cost center of the user	
	department	the department of the user	
	division	the division of the user	
	employeeNumber	the employeeNumber of the user	
	organization	the organization of the user	

	manager {
	value
	$ref
	displayName
	}
		the manager of the user
		A user cannot set himself or herself as their own manager. Additionally, a client must have scoping privileges on a manager.
		value – the id of the SCIM user representing the manager of the user
		$ref - The URI of the SCIM resource representing the manager of the user
		displayName – the display name of the manager of the user	

urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User			
	birthdate	The birthdate of the user	The format is in ISO 8601 Date time format
	disabled	The disabled state of the user	false – the user is not disabled true – the user is disabled
	gender	The gender of the user	
	passwordMustChange	A Boolean flag indicating if the user must change their password during the next authentication	false – password is not required to change true – password must be changed after authentication succeeds
	updateable	A flag indicating whether the user is updateable.	Typically set to true for Internal ID Store users. For users created using IDP JIT this flag could be set to false.
	birthdate	The birthdate of the user	The format is in ISO 8601 Date time format
urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User:credentials			

SCIM schema URN	Attribute Name	>Description	Value
	Credentials	Credentials are the set of user credentials based on verified user	
`urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User:custom`			
	The attributes within this URN are deployment-specific. See the User Schema extensions section for details.		
`urn:ietf:params:scim:schemas:core:2.0:Group`			
	displayName	The unique name of the group	
	id	The unique SCIM resource id for the Group	id is an immutable value that is generated by the Identity Service when a Group is created (POST Group) and is returned in the POST response
	members	A list of members of the Group.	

User Schema Extensions for the Identity Service

The Identity Service supports extending the User schema for deployment-specific needs.

You can define custom attributes using the Additional User Attributes section on either the Internal ID Store Configuration page, or the Internal ID Store API. The following additional attribute definitions are supported:

Property	Description	Value
name	A unique name for the attribute. The name must be unique across all of the SCIM attribute URNs defined in the Internal Identity Store Configuration.	
type	The data type of the attribute: • STRING • INTEGER • BOOLEAN • DATE	one of the supported types
searchable	An indicator if the attribute can be used in a search operation. **Note:** This enforcement is only honored using the ID Store API, SCIM protocol does not define searchable.	**false** – attribute cannot be used in a search expression **true** - attribute cannot be used in a search expression
enabled	Defines the enabled state. Attributes whose enabled state = false cannot be used in requests, searches, and are not returned in responses.	**false** – the attribute is disabled and cannot be used in user operations **true** - the attribute is enabled and can be used in user operations
multivalued	Sets whether attributes can have more than one value. Applies only to attributes of type string, integer, or date. Once set to multivalue, an attribute cannot be changed to single value.	**false** – the attribute can have only one value **true** - the attribute can have multiple values. If this property is not set, the attribute supports only a single value.

Defining Additional User Attributes for the Identity Service

The Symantec Identity Security Platform Admin Console supports managing ID Stores through the ID Store Management section.

When selecting the Internal ID Store, the main page provides an Additional User Attributes control where you can add/remove user attributes using the properties that are described in the User Schema Extensions for the Identity Service topic.

Also, the Internal ID Store Configuration API supports the **customUserAttributes** property. The following example shows the definition of five additional user attributes:

```
"customUserAttributes": [
     {
         "name": "officeLocation",
         "type": "STRING",
         "searchable": true,
         "enabled": true
         "multivalued": true
     },
     {
         "name": "jobCode",
         "type": "INTEGER",
         "searchable": true,
         "enabled": true
         "multivalued": false
     },
     {
         "name": "isUsCitizen",
         "type": "BOOLEAN",
         "searchable": true,
         "enabled": true
         "multivalued": false
     },
     {
         "name": "hireDate",
         "type": "DATE",
         "searchable": true,
         "enabled": true
         "multivalued": false
     },
     {
         "name": "adminType",
         "type": "STRING",
         "searchable": true,
         "enabled": true
         "multivalued": false
     }
 ]
```

Manage Users and Groups in the Identity Service

User and Group management is provided via a standard SCIM v2 interface, along with schema extensions that you can use as required by the business service. The Identity Service and the supported SCIM v2 interface operate on Users and Groups in the Internal Id Store. You can create Users and Groups using the SCIM interface, or optionally created via from an Identity Provider configured to support Just-In-Time (JIT) provisioning. For more information, see Identity Provider Discovery - Just-In-Time (JIT) User Provisioning.

This section provides an overview of the User and Group management capabilities and considerations. The Management APIs are fully documented in the Identity User API and Identity Group API topics of the Identity REST APIs section.

Manage Users in the Identity Service

User Management is provided through a standard SCIM v2 interface along with User schema extensions that you can use as required by the business service.

The Identity Service and the supported SCIM v2 interface operate on Users in the Internal Id Store. You can create Users with either the SCIM interface, or optionally from an Identity Provider that is configured to support Just-In-Time (JIT) provisioning.

For more information, see Identity Provider Discovery - Just-In-Time (JIT) User Provisioning.

This section provides an overview of the User Management capabilities and considerations. The User Management APIs are fully documented in the Identity User and Identity Group sections of the REST APIs.

Learn how to manage users with the Identity Service with the following capabilities:

- Creating Users
- Deleting Users
- Retrieving Users
- Updating Users
- User Self Service

Creating Users

You create users through the Identity Service SCIM2 Users API (`{tenantName}/scim2/v1/Users`) using a POST operation. Consider the following factors during User creation.

Identity Uniqueness

Identities must be unique. The **id** and **username** attributes are used to guarantee uniqueness. The **id** attribute (also known as a resource Id) identifies the unique SCIM User resource. The **id** attribute is automatically generated by the Identity Service on a create user request. Optionally, on a create user request, a client can pass in a unique GUID value for the **id**.

The **userName** must also be unique. **UserName** is an attribute that is typically used for user disambiguation as part of an Authentication Flow. You must provide the **userName** attribute as part of the create User request and the service checks for uniqueness. The request fails if the provided **userName** is not unique.

Emails and Phone Numbers

Emails and phone numbers are typical profile attributes that have a unique characteristic in Symantec Identity Security Platform, in that they *can be used* as a Credential for authentication.

To support using email and phone numbers for authentication, an extension to the standard SCIM User Emails/Phone Numbers adds a `verified` attribute: If the Email/Phone Number has a `verified=true` value, these entities populate as email and SMS credentials. Phone numbers support a `capabilityType` attribute, with a value of either `sms` or `ivr` . The IDSP Credential APIs returns these verified emails and phone numbers in the list of User credentials.

Subsequent updates to emails and phone numbers update the appropriate credential value. Changing the verified value of an email or phone number to false removes the associated credential.

The following table describes these attributes:

Attribute	Verified	Capability Type	Populated as a Credential
emails	true/false		When `verified = true`
phoneNumbers	true/false	values: `sms` , `ivr`	When`verified = true` The credential type matches the capability type

Verified emails and phone numbers are also used as authentication factors for Self Service Password Recovery and Self Service Account Unlock. See the Self Service sections for details.

Password Attribute

The password is an optional attribute that can be set on the user depending upon the business solution requirements. Composed passwords must conform to the associated Password Policy rules. AuthHub supports one more applicable Password Policies. A submitted password in a User request that does not conform to the applicable password policy fails the request.

> **NOTE**
> Administrators may or may not be allowed to set the User password. For more information, see the Administrative Password Management section of the Password Management in the Identity Service topic.

In-transit Password Encryption

The User APIs support the ability to encrypt the password in transit. A client of the API can first encrypt the password before making a User request containing the password. This ability is also true for the Password Authenticator. For example: `{tenantName}/scim2/v1/PasswordAuthenticator`

Create User Post-Processing

When a Create User request completes and that request did not have the User password that is defined in the request, the post-processing handler performs the following actions:

- Retrieves the primary email address for the user. If the created user has no defined primary email address, the process aborts.
- Initiates a `PasswordResetInitiator` request.
- Sends a welcome email to the primary email for the user containing a link allowing the user to set their password.

Applications willing to own the Create User Post-processing notifications can turn off the default behavior of sending welcome emails. To do so, set the query parameter `dontSendActivationNotice` to **true** to the create User API. For more information, see User API.

Retrieving Users

The SCIM v2 interface supports retrieving users by resource Id, with a GET operation or searching for users through filters using GET or POST.

These operations appear as follows:

```
GET {tenantName}/scim2/v1/Users/<user_id>
```

```
GET {tenantName}/scim2/v1/Users/?filter={attribute}{op}
{value}&sortBy={attributeName}&sortOrder={ascending|descending}
```

```
POST {tenantName}/scim2/v1/Users/.search
```

```
 {
    "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:SearchRequest"
```

```
    ],
    "filter": "(userType eq Employee) and (jobCode eq 10)"
}
```

Supported Attribute Operators

Attribute Operators	Description
eq	equals
ne	not equals
co	contains
sw	starts with
ew	ends with
pr	present (has a value)
gt	greater than
ge	greater than or equal
lt	less than
le	less than or equal

Examples:

```
?filter=title contains manager
```

```
?filter=department contains operations
```

```
?filter=addresses.country eq usa
```

Supported Logical Operators

You can combine attribute operators with the following logical operators:

Operators	Description
and	logical AND of attribute operators
or	logical OR of attribute operators
not	logical OR of attribute operators

Examples:

```
?filter=userType eq employee and addresses.country eq usa
```

```
?filter=(userType eq employee or userType eq contractor) and addresses.country eq usa
```

Paging Support

Paging is supported for /Users and /Group queries.

Operators	Description
count	Specifies the number of entries to return in each page.
startIndex	The index from the result is set to use for the next page.

Examples:

Returns the first 50 users from the query:

`?`**`count=50&startIndex=1`**`&filter=userType eq employee and addresses.country eq usa`

Returns the next 50 users from the query:

`?`**`count=50&startIndex=51`**`&filter=userType eq employee and addresses.country eq usa`

Attributes to Retrieve

SCIM supports the ability to specify the set of attributes to retrieve with the attributes parameter. The set of attributes that are persisted with the user are returned by default in a user request response. SCIM supports two attribute parameters to control this default behavior, as shown in the following table:

Parameter	Description
attributes	Defines the set of attributes to be returned in a User response.
excludedAttributes	Defines attributes that should not be returned in User response.

NOTE
The **User.groups** attribute is dynamically calculated. Like other attributes, it can be excluded if desired.

Updating Users

You can update Users with either a PATCH or PUT operation. PATCH replaces just the attributes that are specified in the request. PUT replaces the entire user object with the specified attributes in the request, except for attributes that are not writeable using SCIM. See the Internal Id Store Configuration page in the Admin Console for the list of attributes that are not SCIM writeable.

Deleting Users

You can delete users with the DELETE operation. The DELETE operation takes the user ID (mandatory) and ReplaceManager (optional) as input. User deletion is permanent; you can only restore deleted users from the Internal ID Store database backups.

Use the ReplaceManager parameter when deleting a manager to assign the provided managerID to all accounts previously assigned to the deleted manager.

User Self Service

Per SCIM standard in addition to the `/Users` endpoint a `/Me` endpoint (***`{tenantName}/scim2/v1/Me`***) is supported by the Identity Service.

The following additional self service operations are supported using the Identity Service:

- Initiate My Email Verification
- Verify My Email
- Initiate My Phone Number Verification
- Verify My Phone Number
- Get My Verified Credentials

See the complete details in the Identity Self Service section of the Using the REST APIs sections.

To access the Self Service endpoints, you require a User Access token as generated from an authentication flow (`/auth/v1/authenticate`) and a successful authentication that is based on the configured authentication flow.

Simple Attribute Names in API Parameters

Typically, attributes are uniquely identified by prefixing the attribute name with its schema URN separated by a colon (:). For example, the `birthDate` attribute is identified as `attributes=urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User.birthdate` .

When listing attributes for use with `filter` , `sort` , `attributes` , and `excludedAttributes` when managing users, it is not necessary to specify the complete attribute name (including URN); the simple attribute name is sufficient. For example: `filter=birthDate` .

Manage Groups in the Identity Service

This section provides an overview of the Group Management capabilities and considerations.

Group Management is provided using a standard SCIM v2 interface. The Identity Service and the supported SCIM v2 interface operate on Groups in the Internal Id Store. You can also fully manage Groups using the SCIM interface.

The Group Management APIs are fully documented in the Identity UserUser API and Identity GroupGroup API management sections of the Identity REST APIs sections.

Learn how to manage groups with the Identity Service with the following capabilities:

- Creating Groups with the Identity Service
- Adding and Removing Members with the Identity Service
- Retrieving Groups with the Identity Service

Creating Groups with the Identity Service

You create groups using the Identity Service SCIM2 Group API (***{tenantName}/scim2/v1/Groups***) using a POST operation. Consider the following factors during Group creation.

- ***Group Uniqueness***: Groups must be unique. The **id** and **displayName** attributes are used to guarantee uniqueness. The **id** attribute (also known as the resource Id) identifies the unique SCIM Group resource. The Identity Service automatically generates the **id** attribute on a create group request.
- The **displayName** must also be unique. You must provide the **displayName** attribute as part of the create Group request, and the service checks for uniqueness. The request fails if the provided **displayName** is not unique.

Making a Group Read Only

You set whether groups are read-only using the Identity Service GroupOperator API (`https://${SSP_FQDN}/{{tenantName}}/admin/v1/GroupOperator/{{groupID}}`) using a **readOnly** operational attribute in a POST operation. When the **readOnly** attribute is set to **true** for the group, the principal can read the group details but cannot perform any update or delete operations on the group, including updating group memberships. If set to `false`, principals with the appropriate permissions can read the details of the group and can perform update or delete operations on the group.

You can use the Identity Service GroupOperator API as a GET operation to view the read-only status of a group.

Adding and Removing Members with the Identity Service

The SCIM Group interface includes support for adding Users as members of a given group using the PATCH operation.

> **NOTE**
> Currently the Identity Service limits the maximum number of members to 5,000. Attempts to add members beyond this count cause the Add member operation to fail.

The following example shows add and remove member operations:

```
PATCH /scim2/v1/Groups/<group_id>
{
"schemas": [
    "urn:ietf:params:scim:api:messages:2.0:PatchOp"
],
"Operations": [
    {
        "op": "add",
        "path": "members",
        "value": [
            {
                "value": "89ac5e06-208e-4799-8c06-0ab981aa9848"
            }
        ]
    }
]
}

{
"schemas": [
    "urn:ietf:params:scim:api:messages:2.0:PatchOp"
],
"Operations": [
    {
        "op": "remove",
        "path": "members",
        "value": [
            {
                "value": "86ac5e06-208e-4799-8c06-0ab981aa2212"
            }
        ]
    }
]
}
```

Retrieving Groups with the Identity Service

The SCIM v2 interface supports retrieving groups either by resource Id with a GET operation, or searching for groups with filters using GET or POST.

The following examples show these operations:

```
GET {tenantName}/scim2/v1/Groups/<user_id>
GET {tenantName}/scim2/v1/Groups/?filter={attribute}{op}{value}&sortBy={attributeName}&sortOrder={ascending|
descending}
POST {tenantName}/scim2/v1/Groups/.search
{
"schemas": [
    "urn:ietf:params:scim:api:messages:2.0:SearchRequest"
],
"filter": "displayName eq EmployeeGroup"
}
```

Attributes to Retrieve

SCIM supports the ability to specify the set of attributes to retrieve with the attributes parameter. The set of attributes that are persisted with the group are returned by default in a group request response. SCIM supports two attribute parameters to control this default behavior, as shown in the following table:

Parameter	Description
attributes	defines the set of attributes to be returned in a User response
excludedAttributes	defines attributes that should not be returned in User response

NOTE
`Group.members` is NOT returned by default. This functionality is by design, due to the large number of values that can be returned for groups with many members. Adding the parameter `?attributes=members` returns the members set.

Constraining User and Group Administration by Origin ID

User and Group Administration can be constrained by the Identity Provider (IDP) from which these resources originated.

In Federated Authentication scenarios using Just In Time (JIT) provisioning, authenticated users and their associated groups can be provisioned into the Symantec Identity Security Platform internal identity store. When this JIT provisioning occurs, the originating IDP Identifier is maintained in the originId field of the JITprovisioned resources. Similarly, Users and Groups can be created using the IDSP SCIM API where the originId is specified in the request.

The SCIM User and Group APIs include an option to restrict administration of these users and groups based on a matching origin ID in the access token of the authenticated principal.

Enabling Origin Id in the access token

To enable the presence of the origin ID in the access token, you must configure the client application to invoke the /SCIM2/v1 User or Group API to contain the following metadata property:

Application Metadata Property	Value
urn:iam:app:assertIdpOriginIdOnScimOperations	`<idpID>`

This metadata property can be set using the Application API:

```
https://{{sspHost}}/{{apiPathTenant}}/admin/v1/Apps
```

The following example shows a sample JSON output which includes the required metadata property:

```
metadata": [
      {
          "name": "urn:iam:app:assertIdpOriginIdOnScimOperations",
          "value": "e5f61dec-93e2-4a57-bf6e-0fa27e4592d0"
          // Note: value could be the IDP Name (e.g. “bankingIDP”)
      }
]
```

When this property is set on the client application that is used to obtain an access token for the SCIM API request, the access token contains the following field matching the metadata property value for the application:

```
"idp_origin_id": ["e5f61dec-93e2-4a57-bf6e-0fa27e4592d0"]
```

SCIM Enforcement of origin ID

If the access token associated with the SCIM User or Group request contains the origin ID field as shown in the example above, the SCIM API relies on the value to operate on the matching User and Group resources.

The following table shows the User and Group origin Id attributes:

Resource	Origin ID Attribute
User	idpOriginId
Group	originId

SCIM Operational Semantics when access token has an origin ID field

The following table shows the operational enforcement semantics when the access token that is presented to a SCIM request contains an origin ID field:

Resource	SCIM Operation	Origin ID in Request	Result
User/Group	POST (create)	No	A User or Group is created where <originId> is set to `AT.idp_origin_id`.
		Yes	If <originId> = `AT.idp_origin_id`, a user or group is created containing this <originId> value. Otherwise, returns a not authorized error.
	GET (by ID)	N/A	If User/Group <originId>= `AT.idp_origin_id`, the User or Group is returned. Otherwise, returns a not authorized error.
	GET (search with filters)	No	User/Group <originId> = `AT.idp_origin_id` is added as a default filter.
		Yes	If User/Group <originId> = `AT.idp_origin_id` results are returned. Otherwise, no results are returned.
	PUT/PATCH (update)	No	If User/Group <originId> = `AT.idp_origin_id`, allows the operation. Otherwise, returns a not authorized error.
		Yes	If User/Group <originId> = `AT.idp_origin_id`, allows the operation. Otherwise, returns a not authorized error.

Resource	SCIM Operation	Origin ID in Request	Result
	DELETE	N/A	If User/Group <originId>= `AT.idp_origin_id` the User or Group is deleted. Otherwise, returns a not authorized error.
Group (members)	PUT/PATCH (with members)		For each member to be added or removed, the User.<originId> must match the `AT.idp_origin_id` and the `Group.<originId>`.

Managing the Identity State

There is a set of attributes for the User that control the operational state. These Boolean attributes and their values and operational semantics appear in the following table.

Attribute	Description	Operational Semantics	Default value if not designated in the create request
active	Determines the locked state	true: user is unlocked false: user is locked	false
disabled	Designates the enabled/ disabled state	true: user is disabled false: user is enabled	false
passwordMustChange	Determines if the user must change the password post-authentication	true: user must change password after login false: not required to change password after login	false
updateable	determines if the user can be modified	true: allow user changes false: prevent user changes	true

Password Management in the Identity Service

Identity Service provides Password Management for Users in the Internal ID Store.

The Password Management Operations described in this section only pertain to the Internal ID Store Users.

Learn how to manage passwords with the Identity Service with the following capabilities:

- Defining Password Policies
- The Default Password Policy
- Administrative Password Management
- Modifying Credentials or Profile Attributes for Another Administrator
- Administrative Password Reset and Email Templates
- Setting the Password Must Change Condition for a User
- Sign-in Application Force Change Password
- Federated User Lifecycle Management
- Managing the Federated Lifecycle Using a Privileged Application
- Authentication APIs
- Privileged Application Security Tokens Generation for Password-based Account Activation and Credential Verification

Defining Password Policies

Identity Service lets you define one or more Password Policies. Password policies define the following items:

- Composition rules
- History rule (number of previous passwords to maintain)
- Password age rules
- Optional dictionary of disallowed words in passwords
- Applicable user criteria (a directory search filter of ALL or the set of Groups where members apply)
- A priority value if more than one policy matches a user.
- Options to re-evaluate policies during Authentication
  - Evaluate Composition Rules On Authentication = true. (false by default)
    - Handles cases where a password composition rule has changed and during authentication of a password you want the new rules to take effect immediately. This policy forces the validation of the current password against the updated composition rules. If the current password does not meet the new rules, the user is forced to update their password.
  - Evaluate Expiration Rules On Authentication = true. (false by default)
    - Handles cases where a password expiration rule has changed and during authentication of a password you want the new rules to take effect immediately. This policy requires the validation of the current password age against the updated password expiration rule. If the current password is older than the updated expiration rule, the user is forced to update their password.

The Default Password Policy

A default Password Policy deploys when the Identity Service first deploys in Symantec Identity Security Platform. This password policy is named **DefaultPasswordPolicy.** *This policy cannot be disabled or deleted; it can only be updated.*

Password Policies are consulted in the following operations:

- All Password updates (administrative or self-changes)
- Explicit Invocation of the password validation API (/identity/v1/UserPasswordValidator)
- Invocation of the Generate Temporary Password API (/identity/v1/UserPasswordGenerator)

You can update the default password policy in the Admin Console (see Setting a Password Policy) or the Password Management API (see Password Management API).

Administrative Password Management

An Administrator performs password management on Users either through SCIM or dedicated Password Changer APIs.

Administrative Password Management is a privileged operation that is governed by the following items:

- The **Administrators Can Generate Password** setting (true/false) determines if Administrators can set a user password, and as a result, proves that the administrator has knowledge of that password. For more information, see Configuring the Internal ID Store.
- The Administrator having been granted the User Password Administrator Role that provides the `urn:iam:t.userpasswordadmin` privilege in an Access Token (AT).

The following table describes the Administrator Password semantics based on the setting, and an Administrator granted the User Password Administrator Role.

Setting	Value	`t.userpasswordadmin` privilege	Password Change Behavior
AdministratorCanGeneratePassword	false	Not present in AT	APIs containing a password are not allowed. Requests fail with 403
AdministratorCanGeneratePassword	true	Present in AT	APIs containing a password are allowed

An Administrator performs password management on Users either through SCIM or a dedicated Password Changer API, as shown in the following examples:

`PATCH scim2/v1/User/<userId>`

```
{
    "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:PatchOp"
    ],
    "Operations": [
        {
            "op": "replace",
            "path": "password",
            "value": "Sample@1234"
        }
    ]
}
```

`/identity/v1/UserPasswordChanger`

```
{
    "subject": "defaulttenantadminuser",
    "password": "Sample@1234"
}
```

Modifying Credentials or Profile Attributes for Another Administrator

An AuthHub Administrative User is a user that is granted any administrative privilege using an Auth Hub Administrative Role. Modifying the credentials or profile attributes of another or deleting an Admin User is restricted to a set of highly privileged Administrators.

A subject (authenticated principal) is limited to what changes they can make to an Administrative User target. Unless the following conditions are true for the subject, such subjects require new privileges to modify the credentials of an Administrative User, modify profile attributes, or delete an Administrative User.

1. The subject admin is the target admin (self-modify only, not deletion)
   - **Access Token conditions enforced**:
     `AT.user_guid eq target user_guid`
2. The subject is the Tenant Admin and for the email/phone/password changes the Internal Identity Store Configuration property `adminCanGeneratePassword` is set to true.
   - **Access Token conditions enforced**:
     `AT.scope co (urn:iam:t.all and urn:iam:t.userpasswordadmin)`
3. The subject admin is "System Admin", has been granted either the `urn:iam:t.adminroles` or `urn:iam:t.all` privilege, and for the email/phone/password changes the Internal Identity Store Configuration property `adminCanGeneratePassword` is set to true.

- **Access Token conditions enforced**:

```
AT.tenant eq "system" and AT.scope co (urn:iam:t.adminroles or urn:iam:t.all) and AT.scope co
 urn:iam:t.userpasswordadmin
```

Administrative Password Reset and Email Templates

An Identity Service administrator can set a password for a new user, and initiate a password recovery for a user. The Password Recovery feature relies on either a verified email or a verified phone number to authenticate and deliver a Password Reset Token. This Password Reset Token allows the user to reset their password using a Password Reset Changer Endpoint. Like the token, the Password Reset Changer endpoint message is sent to the primary or selected verified email or verified phone number of the user during the Password Recovery Process.

Administrative Password Recovery Process

The following procedure describes the Administrative Password Recovery Process. The client can be either a deployment-specific client or the Sign-in Application.

1. The Client invokes the API for the User, and returns one or more verified email or phone numbers.
   - **Endpoint URL**: `/identity/v1/VerifiedCredentials`
   - **Operation**: `GET`

   When invoked with the `primaryOnly=true` parameter, the API returns only the verified email or phone number that is set as a primary.
   - **Endpoint URL**: `/identity/v1/VerifiedCredentials?primaryOnly=true`
   - **Operation**: `GET`
2. The user selects a verified email or phone to use to deliver a Password Reset link
3. The client invokes the Password Reset Initiator API that provides the verified email or phone.
   - **Endpoint URL**: `/identity/v1/PasswordResetInitiator`
   - **Operation**: `POST`

   A Password Reset Token is generated. An email or SMS message delivers the password reset link. See the following sub-procedure: “Email and Phone Message Templates for Administrative Password Reset and Create New User”.

   Invoking the `PasswordResetInitiator` with the `newUser=true` parameter delivers a welcome message by email or SMS to the end user that contains the link to reset the password. See the following Message Templates `NEW_USER_WELCOME` actions for more information.
   - **Endpoint URL**: `/identity/v1/PasswordResetInitiator?newUser=true`
   - **Operation**: `POST`

   To activate a new user or reset a password using a vanity URL include the `passwordResetVanityURL` parameter:
   - For new user activation: `/identity/v1/PasswordResetInitiator?newUser=true&passwordResetVanityURL=<vanityIssuerUrl>`
   - For password reset: `/identity/v1/PasswordResetInitiator?passwordResetVanityURL=<vanityIssuerUrl>`
4. The users receive the email or SMS, and then the user selects the link to invoke the Password Reset Changer API with the Password Reset Token.
5. The client invokes the Password Reset Changer API.
   - **Endpoint URL**: `/identity/v1/PasswordResetChanger?token=<reset-token>`
   - **Operation**: `POST`
6. The API verifies the token and, if verified, allows the user to reset their password.

Email and Phone Message Templates for Administrative Password Reset and Create New User

The Administrative password reset email and SMS messages that are delivered are based on the following template actions and message types. The email message requires separate templates for the email subject and the email body.

Message template for SMS messages:

- **Message Action**: `ADMIN_PASSWORD_RESET`
- **Message Type**: `ADMIN.PASSWORD.RESET.SMS`

Message template for email subjects:

- **Message Action**: `ADMIN_PASSWORD_RESET`
- **Message Type**: `PASSWORD.RESET.EMAIL.SUBJECT`

Message template for the email body:

- **Message Action**: `ADMIN_PASSWORD_RESET`
- **Message Type**: `PASSWORD.RESET.EMAIL.BODY`

The welcome messages delivered to a newly created user by the Administrator are based on the following message template actions and message types.

Message template for SMS messages:

- **Message Action**: `NEW_USER_WELCOME`
- **Message Type**: `NEW.USER.WELCOME.SMS`

Message template for email subjects:

- **Message Action**: `NEW_USER_WELCOME`
- **Message Type**: `NEW.USER.WELCOME.EMAIL.SUBJECT`

Message template for the email body:

- **Message Action**: `NEW_USER_WELCOME`
- **Message Type**: `NEW.USER.WELCOME.EMAIL.BODY`

The `Get Messages` API lists the messages in the system.

- **Endpoint URL**: `/admin/v1/Messages`
- **Operation**: `GET`

The `Update Messages` API supports updating the message content.

- **Endpoint URL**: `/admin/v1/Messages`
- **Operation**: `PUT`

Setting the *Password Must Change* Condition for a User

Administrators can force users to change their password the next time that they log in if the `passwordCannotBeChanged` attribute for the user is **false**. An Administrator can do this change by setting the User attribute `passwordMustChange` to **true**. The default is **false**.

Sign-in Application Force Change Password

When authenticating through the Sign-In Application, after the user has completed authentication (which typically includes password as a factor), the user is then redirected to a **Password Must Change** page, if the `passwordCannotBeChanged` attribute for the user is **false**.

Federated User Lifecycle Management

Users can be created in two ways:

- Locally, typically using the SCIM Service API.
- Externally when a user accesses your system from an external identity provider (IdP) and **Just In Time Provisioning** is enabled.

The first time that a user authenticates from an external location, the user is added as a new account. The federated attribute for a user account that is created with Just in Time Provisioning is automatically set as true.

The following topics describe how to create and manage federated users with the SCIM Service API. Users which are service accounts cannot be federated.

NOTE
Federated users have a password only when the `allowLocalPasswordForFederatedUser` flag is set to **true** in the internalIdStoreConfig settings. See Internal ID Store Configuration API.

Creating a federated user

By default, the `federated` flag set to **false** for a user created locally using the SCIM Service API. However, you can manually create a federated user:

1. Use a POST operation to create a user and set them as federated.
2. Set the user attributes as required. Specifically, set the `federated` flag to **true** and provide a valid external IdP or external identity source for the `idpOriginId` flag.

For example:

```
{
    "schemas": [
      "urn:ietf:params: scim:schemas: core:2.0:User",
            "urn:ietf:params: scim:schemas: extension: iam:ssp:1.0:User",
            "urn:ietf: params: scim:schemas: extension:iam:ssp:1.0:User: custom"
    ],
    "userName":
    "displayName":
    "active": true,
    "name": {
            "formatted": "<value>",
            "familyName": "<value>",
            "givenName": "<value>"
    },
    "title": "HR Manager",
    "userType": "Employee",
    ...
    ...
    "urn:ietf: params: scim:schemas: extension: iam:ssp:1.0:User": {
            "gender": "male",
            "birthDate": "1990-01-01T00:00Z",
            "disabled": false,
            "federated": true
    },
    "urn:ietf: params: scim:schemas: extension: iam:ssp:1.0:User:custom": {
            "string1": "value",
            "integer1": "5",
            "boolean1": "true
}
```

Changing a local user to a federated user

To change a user from local to federated:

1. Use a PATCH operation to add `federated` to the schema.
2. Set the `federated` flag to **true** and provide a valid external IdP or external identity source for the `idpOriginId` flag.

3. Optionally, use the `forceReset` flag to require the user to reset the account.

For example:

```
{
   "schemas": [
      "urn:ietf:params:scim:api:messages:2.0:PatchOp"
   ],
   "Operations": [
      {
         "op": "replace",
         "path": "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User:federated",
         "value": "true"
      }
   ]
}
```

Changing a federated user to a local user

To change a user from federated to local:

1. Use a PATCH operation to add `local` to the schema. For example:
2. Set the `federated` flag to **false** and, optionally, set the `idpOriginId` flag to **internal**.

For example:

```
{
   "schemas": [
      "urn:ietf:params:scim:api:messages:2.0:PatchOp"
   ],
   "Operations": [
      {
         "op": "replace",
         "path": "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User:federated",
         "value": "false"
      }
   ]
}
```

The account is set to an inactive state, and the `passwordMustChange` flag is set to **true**. The user receives a security token in an email or SMS message (depending upon your Managing Message Provider Configuration). The user must reset the password and the account by following the instructions in the email or SMS message.

Using bulk operations to modify the federated flag

The procedures for using a bulk API operation to change a group of users from local to federated, or from federated to local, are nearly the same as when performing the action for single users. However, note the following differences:

- You cannot mix conversion types. That is, you cannot change some users to federated and some to local in the same operation. Use a separate operation for each type of change.
- Bulk conversions can only be performed on users.
- When converting from local to federated, all users in the bulk request must belong to the same idpOriginId.

For more information about bulk API operations, see Bulk API.

User Attribute Status based on token usage and operation

Attribute status per function illustrates the status of User attributes when a token is generated and after it is redeemed, as a result of the following operations:

NOTE
Only one token of any type can exist at one time. Each time a token is generated, the new token overwrites any existing token of that type.

- Resetting a password
- Activating a new user
- Changing a user from federated to local
- Changing a user from local to federated.

Table 81: Attribute status per function

Function	API	
Password Reset	Privileged Client: `/identity/v1/PasswordResetInitiator?target=api`	CRED_RES (Token is ge

New User Activation	Privileged Client:
	`/identity/v1/PasswordResetInitiator?target=api&newUser=true`
	SCIM:
	POST
	`/scim2/v1/Users`
		CRED_RES
		(Token is ge

Federated to Local	Privileged Client: `/identity/v1/PasswordResetInitiator?target=api&federated=false` SCIM: • PATCH `/scim2/v1/Users/` (patching federated as false) • PATCH (Bulk) `/scim2/v1/Bulk`	CRED_RES (Token is ge

Local to Federated	Privileged Client:
	`/identity/v1/PasswordResetInitiator?target=api&federated=true&idpOriginId=`
	SCIM PATCH:
	`/scim2/v1/Users/&?idpOriginId= (patching federated as true) PATCH (Bulk) /scim2/v1/Bulk/?idpOriginId=`
		No token ge

Managing the Federated Lifecycle Using a Privileged Application

The procedures for using a privileged application to change a user from local to federated, or from federated to local, are nearly the same as when using the SCIM Service API. However, note the following differences:

- You must send the following flags:
  - Set `federated` to **true** when converting a local account to federated, or to **false** when converting a federated account to local.
  - Set `idpOriginId` to a valid external IdP or external identity source when converting a local account to federated. Optionally, set `idpOriginId` to **internal** when converting a federated account to local.
- You may optionally set `forceReset` to **true** to require the user to reset the account (the default is **false**).
- You may optionally set the `newUser` to **true** (the default is **false**).

  - If sent, the response acts as a new user accountActivation token. The user must activate the account and reset the password.
  - If not sent, the response includes a password reset URL.
- If the user is changed from local to federated, no security token is generated. If the user is changed from federated to local, a new security token is generated and returned in the API response. The privileged application must provide this security token to the user to reset the account and password.

For more information about privileged applications, see Privileged Application Security Tokens Generation for Password-based Account Activation and Credential Verification.

Authentication APIs

When using the Authentication APIs, after successful authentication the **nextAction** is set to **CHANGE_PASSWORD**. The following code shows examples:

```
/auth/v1/authenticate
{
    "channel": "web",
    "subject": "lawrence_haley",
    "ipAddress": "10.10.123.124",
    "action":"authenticate",
    "acrValues": [],
    "authContext": [
        {
            "key": "useTenantKeyForITEncryption",
            "value": "false"
        }
    ]
}
```

Response (partial):

```
{
    "flowState":
 "NWsvSVZPQUN1Zko5enlWT0JCYUxnUHlxSXlCVVZHK1J0M1l5VnpRR25PeUcyenkrekx1SmJPOUVPUkV6U0czN0tiVDFlSFU3K1RKb3pUQ3R4R2s0OWg3e
    "userName": "lawrence_haley",
    "userId": "89ac5e06-208e-4799-8c06-0ab981aa9848",
    "nextaction": "PASSWORD_AUTH",
    "currentFactors": [
        "PASSWORD"
    ],

}
 /factor/v1/PasswordAuthenticator
{"password":"Sample@1234"}
```

Response (partial):

```
{
    "flowState":
 "NWsvSVZPQUN1Zko5enlWT0JCYUxnUHlxSXlCVVZHK1J0M1l5VnpRR25PeUcyenkrekx1SmJPOUVPUkV6U0czN0tiVDFlSFU3K1RKb3pUQ3R4R2s0OWg3e
    "userName": "lawrence_haley",
    "userId": "89ac5e06-208e-4799-8c06-0ab981aa9848",
    "nextaction": "CHANGE_PASSWORD",
    "currentFactors": [
```

```
        "CHANGEPASSWORD"
    ],
    "data": {
        "errorCode": "1003033",
        "message": "Password authentication success, however password must be changed",
        "passwordResetURL": "https://xxxx/default/ui/v1/signin/initiatePasswordReset?x-flow-
state=NWsvSVZPQUN1Zko5enlWT0JCYUxnUHlxSXlCVVZHK1J0M1l5VnpRR25PeUcyenkrekx1SmJPOUVPUkV6U0czN0tiVDFlSFU3K1RKb3pUQ3R4R2s0C
}
```


Privileged Application Security Tokens Generation for Password-based Account Activation and Credential Verification

Application teams handling their own user onboarding can generate one-time, expiring security tokens to use when notifying users of account activation (password reset), email verification, and phone verification.

Application teams intending to send their own notification can access appropriate APIs like `/identity/v1/PasswordResetInitiator` , `/identity/v1/EmailVerificationInitiator` , and `/identity/v1/PhoneVerificationInitiator` to request security tokens.

Access to these initiator APIs is protected by the `SecurityTokenAdminPolicy` Authorization policy in the `system` tenancy. Clients requiring such capability must request a privileged application in the `system` tenancy and must be granted access to the `SecurityTokenAdminPolicy` Authorization policy.

Applications use the registered ClientId and Client Secret of the privileged application to get an access token by invoking the `/oauth2/v1/token` API using the OAuth2 Client Confidential grant with the `X-Client-Tenant-Name=system` header to assert the tenancy.

Applications would then access the APIs with the query param `target=api` in the Admin mode using the previously generated access token.

```
https://${SSP_FQDN}/{{apiPathTenant}}/identity/v1/PasswordResetInitiator?target=api
https://${SSP_FQDN}/{{apiPathTenant}}/identity/v1/EmailVerificationInitiator?target=api
https://${SSP_FQDN}/{{apiPathTenant}}/identity/v1/PhoneVerificationInitiator?target=api
```

The requesting applications can also send the following optional flags:

Flag	Description
`federated`	Send `federated` to identify the user as accessing your system from an external IdP or external identity source. You must also send the `idpOriginId` flag with this flag. Do not send this flag if you are sending the `local` flag.
`local`	Send `local` to identify that the user is stored in your internal identity store. You must also send the `idpOriginId` flag with this flag. Do not send this flag if you are sending the `federated` flag.
`redirectUri`	Enter the URL where the user is redirected after the user performs an action through the requesting application. The `redirectUri` that is saved against the generated security token, which is then returned during the token redemption. If the `redirectUri` is not provided, then the Application HomeUrl from the AccessToken is saved against the security token.
`idpOriginId`	Send the `idpOriginId` flag to identify the identity source for the user: • Enter a valid external IdP or external identity source when the `federated` flag is sent. • Enter **internal** when the `local` flag is sent.

Flag	Description
`forceReset`	Send the `forceReset` flag to require the user to activate the account and reset the password: • If sent with the `federated` flag, the response includes an activation URL. • If sent with the `local` flag, the response acts as an accountActivation token.

Request Payload

```
{
   "subject": "<sampleUserId>",
   "redirect_uri": "<application-specific-redirect-uri>"
}
```

Example

```
{
   "subject": "ff4eb70e-9c99-4d7d-bacb-b01707b94a1e",
   "redirect_uri": "https://www.example.com"
}
```

Response

```
{
  "activationUrl": "<passwordResetUrl/EmailVerifier/PhoneVerifier>?token=<security-token>"
  "token": "<security-token>"
}
```

Example

```
{
"activationUrl":https://localhost/default/identity/v1/EmailVerifier?token=dpPfHolEoymH6k05HzUayBDRZDK"
  "token": "dpPfHolEoymH6k05HzUayBDRZDK"
}
```

Redeeming the Security token using the activationUrl returns the response, the saved redirectUri if present.

Response

```
{
   "redirect_uri": "<application-specific-redirect-uri>"
}
```

Example

```
{
   "redirect_uri": "https://www.example.com"
}
```

Self Service Account Activation in the Identity Service

The Identity Service provides an option for Users to activate their Account the first time that the user authenticates to the account.

During the User disambiguation phase of authentication the User's activation is checked. If the User account is not activated, they cannot proceed to the authentication phase. In the Sign-in Application, if the identified

user account is disabled and inactive, **accountActivated** is false, **passwordMustChange** is true, and **allowSelfServiceAccountActivation** is set to true, the User is prompted to activate their account. After the user successfully activates their account, they can proceed to the authentication phase.

Enabling Self Service Account Activation

Getting and setting the **allowSelfServiceAccountActivation** property.

The **Get Internal Identity Store Configuration** API is used to retrieve the value.

```
Endpoint URL: /admin/v1/InternalIdStoreConfig
Operation: GET
{
    "idStoreMappingStrategy": "user_loginid=${idprofile.user_loginid}",
    "disambiguationQuery": "user_loginid eq ${loginid} or email eq ${loginid}",
    "unlockAccountAfterPasswordReset": false,
    "useMaskedCredentialValues": false,
    "passwordResetInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiatePasswordReset",
    "passwordResetChangerUrl": "${system.issuerUrl}/ui/v1/signin/passwordReset",
    "accountUnlockInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiateAccountUnlock",
    "usePriorityToDeterminePrimary": false,
    "allowSelfServiceAccountActivation" : false,
    "customUserAttributes": []
}
```

The **Set Internal Identity Store Configuration** API is used to set all the values including the **allowSelfServiceAccountActivation** property.

```
Endpoint URL: /admin/v1/InternalIdStoreConfig
Operation: PUT
{
    "idStoreMappingStrategy": "user_loginid=${idprofile.user_loginid}",
    "disambiguationQuery": "user_loginid eq ${loginid} or email eq ${loginid}",
    "unlockAccountAfterPasswordReset": true,
    "useMaskedCredentialValues": false,
    "passwordResetInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiatePasswordReset",
    "passwordResetChangerUrl": "${system.issuerUrl}/ui/v1/signin/passwordReset",
    "accountUnlockInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiateAccountUnlock",
    "usePriorityToDeterminePrimary": false,
    "allowSelfServiceAccountActivation" : true,
    "customUserAttributes": []
}
```

Configuring the Activation Token Delivery

Self service account activation supports only SMS and Email for where the account activation token is delivered. You may set this configuration to SMS, email, or both using the `"accountNotificationStrategyOrder": "email, sms"` property.

```
Endpoint URL: /admin/v1/InternalIdStoreConfig
Operation: GET
{
    "idStoreMappingStrategy": "user_loginid=${idprofile.user_loginid}",
    "disambiguationQuery": "user_loginid eq ${loginid} or email eq ${loginid}",
```

```
    "unlockAccountAfterPasswordReset": false,
    "useMaskedCredentialValues": false,
    "passwordResetInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiatePasswordReset",
    "passwordResetChangerUrl": "${system.issuerUrl}/ui/v1/signin/passwordReset",
    "accountUnlockInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiateAccountUnlock",
    "allowLocalPasswordForFederatedUser": false,
    "usePriorityToDeterminePrimary": false,
    "accountNotificationStrategyOrder": "email, sms",
    "customUserAttributes": []
}
```

Self Service Password Recovery in the Identity Service

The Identity Service supports the ability for a user to recover their password (aka forgot my password use case). The Password Recovery feature relies on a verified email or verified phone as the means of authentication and delivery of a Password Reset token. This Password Reset Token allows the User to reset their password via a Password Reset Changer Endpoint. The delivery of the token and the Password Changer endpoint is via a message sent to the User selected verified email or verified phone number during the Password Recovery Process.

Learn about passwords recovery with the Identity Service with the following capabilities:

- Password Recovery Process
- Password Recovery via the Sign-in Application
- Email and Phone Password Reset Message Templates
- Email and Phone Number Verification

Password Recovery Process

The Password Recovery Process is shown below. The client can be a deployment specific client or the Sign-in Application.

- The client invokes the API for the User (returns one or more verified email or phone numbers)
  - Endpoint URL: /identity/v1/VerifiedCredentials
  - Operation: GET
- The user selects a verified email or phone to be used to deliver a Password Reset link using the Password Token
- The client invokes the **Initiate Password Recovery** API providing the verified email or phone
  - Endpoint URL: /identity/v1/PasswordResetInitiator
  - Operation: POST
  - A password reset token is generated
  - An email or SMS message is delivered with the password reset token and link (see Password Reset Message Templates, below)

    NOTE
    Only one active password recovery token can exist at a time. Each time the User invokes the **Initiate Password Recovery** API, the operation overwrites any existing password reset token with a new token.
- Users receive the email or SMS and invokes the Password Reset Changer Endpoint with the password reset token
- The client invokes the **Password Reset Changer API**
  - Endpoint URL: /identity/v1/PasswordResetChanger?token=<reset-token>
  - The API Verifies the token and if verified allows the user to reset their password

Password Recovery via the Sign-in Application

When using the Sign-in Application to authenticate to an Application, if a Password Factor is configured for authentication flow when presented with the enter a password screen, a “Forgot Password?” The link appears to initiate the Password Recovery Process.

If the user does not have a verified email or phone number, a **No verified credentials** message appears, and the process halts. Otherwise, a list of verified email and phone numbers appears to the user to use to deliver the password reset token.

If the user's attribute `passwordCannotBeChanged` is set as **true**, then the request is forbidden. If not set to **true**, a confirmation page then appears, instructing the user to check their email or phone for a link to return to sign-in after recovering the password. To complete the password recovery process, the user selects the link in the delivered email or SMS message to display a page where they can then reset their password.

Email and Phone Password Reset Message Templates

The Password Reset email and SMS message delivered are based on the following template actions and message types.

Message Action: PASSWORD_RESETMessage

Type: PASSWORD.RESET.SMS

OR

Message Type: PASSWORD.RESET.EMAIL.BODY

The **Get Messages** API lists the messages in the system.

Endpoint URL: /admin/v1/Messages

Operation: GET

The **Update Messages** API supports updating the message content.

Endpoint URL: /admin/v1/Messages

Operation: PUT

Email and Phone Number Verification

Administrators typically mark User emails and phone numbers form trusted sources as verified when creating or updating users in the Internal Id Store. The following example shows a verified email change:

```
Endpoint URL: /scim2/v1/Users/<user_id>
Operation: PATCH
{
    "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:PatchOp"
    ],
    "Operations": [
        {
            "op": "replace",
            "path": "emails",
            "value": [
                {
                    "value": "some-user@acme.com",
                    "display": "***user@**me.com",
                    "type": "work",
                    "primary": true,
```

```
                    "verified": true
                }
            ]
        }
    ]
}
```

In cases where users do not have a verified email or phone to use in the Password Recovery process, the Identity Service provides Self Service APIs for users to verify their email or phone numbers.

This process appears as follows:

- The User authenticates and generates a User Access Token
  - **Endpoint URL**: oauth2/v1/token
  - **Operation**: POST
- The User invokes the **Initiate My Email Verification** or **Initiate My Phone Number Verification** API with the User Access Token
  - **Endpoint URL**: /identity/v1/MeEmailVerificationInitiator
  - **Operation**: POST
    OR
  - **Endpoint URL**: /identity/v1/MePhoneVerificationInitiator
  - **Operation**: POST
- An email or SMS message is delivered with a verification token (see Verification Message Templates, below)

  NOTE
  Only one type of identity verification token can exist at a time; either an active email verification token or an active phone verification token. Each time the User invokes the **Initiate My Email Verification** or **Initiate My Phone Number Verification API** API, the operation overwrites any existing verification token with a new token of the requested type.
- The user invokes the **Verify My Email** or **Verify My Phone Number** API using the verification token
  - **Endpoint URL**: /identity/v1/MeEmailVerifier?token=<verification token>
  - **Operation**: POST OR
  - **Endpoint URL**: /identity/v1/MePhoneVerifier?token=<verification token>
  - **Operation**: POST
- The user can then confirm the process was successful using the **Get My Verified Credentials** API
  - **Endpoint URL**: /identity/v1/MeVerifiedCredentials
  - **Operation**: GET

Email and Phone Verification Message Templates

The Verification email and SMS message delivered are based on the following template actions and message types.

Message Action: VERIFY_PHONE

Message Type: VERIFY.PHONE.SMS

Message Action: VERIFY_EMAIL

Message Type: VERIFY.EMAIL.BODY

The **Get Messages** API lists the messages in the system.

Endpoint URL: /admin/v1/Messages

Operation: GET

The **Update Messages** API supports updating the message content.

Endpoint URL: /admin/v1/Messages

Operation: PUT

Self Service Account Unlock in the Identity Service

The Identity Service provides an option for Users to unlock their Account. An Account can be locked for a number of reasons including reaching the configured strike limit during authentication. The **User.active** attribute indicates the lock status. A value of false indicates the account is locked.

During the User disambiguation phase of authentication the User's lock state is checked. If the User is locked, they cannot proceed to the authentication phase. In the Sign-in Application if the identified user is locked and Self Service Account Unlock is enabled, an option is provided to allow the User to unlock their account. After the user successfully unlocks their account, they can proceed to the authentication phase.

The Identity Service relies on the Internal Identity Store Configuration's **unlockAccountAfterPasswordReset** property to determine if the self-service account unlock feature can be used. Self Service Account unlock is disabled by default. The Self Service Account Unlock Flow is implemented via an invocation of a successful Password Recovery (Reset) Process.

When **unlockAccountAfterPasswordReset=true**, the user's account unlocks when the Password Reset process completes.

Enabling Self Service Account Unlock

Getting and setting the **unlockAccountAfterPasswordReset** property.

The **Get Internal Identity Store Configuration** API is used to retrieve the value.

```
Endpoint URL: /admin/v1/InternalIdStoreConfig
Operation: GET
{
    "idStoreMappingStrategy": "user_loginid=${idprofile.user_loginid}",
    "disambiguationQuery": "user_loginid eq ${loginid} or email eq ${loginid}",
    "unlockAccountAfterPasswordReset": false,
    "useMaskedCredentialValues": false,
    "passwordResetInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiatePasswordReset",
    "passwordResetChangerUrl": "${system.issuerUrl}/ui/v1/signin/passwordReset",
    "accountUnlockInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiateAccountUnlock",
    "usePriorityToDeterminePrimary": false,
    "customUserAttributes": []
}
```

The **Set Internal Identity Store Configuration** API is used to set all the values including the **unlockAccountAfterPasswordReset** property.

```
Endpoint URL: /admin/v1/InternalIdStoreConfig
Operation: PUT
{
    "idStoreMappingStrategy": "user_loginid=${idprofile.user_loginid}",
    "disambiguationQuery": "user_loginid eq ${loginid} or email eq ${loginid}",
    "unlockAccountAfterPasswordReset": true,
    "useMaskedCredentialValues": false,
    "passwordResetInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiatePasswordReset",
```

```
    "passwordResetChangerUrl": "${system.issuerUrl}/ui/v1/signin/passwordReset",
    "accountUnlockInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiateAccountUnlock",
    "usePriorityToDeterminePrimary": false,
    "customUserAttributes": []
}
```

Self Service Password Reset for a Federated User

The Identity Service provides an option for Federated users to reset their Account Password. In the Sign-in Application, if the identified user forgets their password, an option allows the user to reset their account password. After the user successfully resets their account password, they can authenticate.

The Identity Service relies on the Internal Identity Store Configuration's `allowLocalPasswordForFederatedUser` property to determine if federated users can use the self-service account password reset feature. The self service account password reset is disabled by default. Setting `allowLocalPasswordForFederatedUser=true` allows the user's account password reset during the Sign-In flow.

Enabling Self Service Account Password Reset for Federated Users

Use the following code to get and setting the `allowLocalPasswordForFederatedUser` property.

The **Get Internal Identity Store Configuration API** is used to retrieve the value.

```
Endpoint URL: /admin/v1/InternalIdStoreConfig
Operation: GET
{
    "idStoreMappingStrategy": "user_loginid=${idprofile.user_loginid}",
    "disambiguationQuery": "user_loginid eq ${loginid} or email eq ${loginid}",
    "unlockAccountAfterPasswordReset": false,
    "useMaskedCredentialValues": false,
    "passwordResetInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiatePasswordReset",
    "passwordResetChangerUrl": "${system.issuerUrl}/ui/v1/signin/passwordReset",
    "accountUnlockInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiateAccountUnlock",
    "allowLocalPasswordForFederatedUser": false,
    "usePriorityToDeterminePrimary": false,
    "customUserAttributes": []
}
```

The **Set Internal Identity Store Configuration API** is used to set all the values, including the `allowLocalPasswordForFederatedUser` property.

```
Endpoint URL: /admin/v1/InternalIdStoreConfig
Operation: PUT
{
    "idStoreMappingStrategy": "user_loginid=${idprofile.user_loginid}",
    "disambiguationQuery": "user_loginid eq ${loginid} or email eq ${loginid}",
    "unlockAccountAfterPasswordReset": false,
    "useMaskedCredentialValues": false,
    "passwordResetInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiatePasswordReset",
    "passwordResetChangerUrl": "${system.issuerUrl}/ui/v1/signin/passwordReset",
    "accountUnlockInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiateAccountUnlock",
    "allowLocalPasswordForFederatedUser": true,
    "usePriorityToDeterminePrimary": false,
    "customUserAttributes": []
}
```

NOTE
An Identity Administrator can always reset a federated User password without any dependency with the `allowLocalPasswordForFederatedUser` Internal Store Configuration property.

Self Service Password Recovery and Account Unlock API

The Identity Service supports APIs allowing users to recover their password (aka forgotten password). The APIs rely on a User's verified email or verified phone as supported credentials to allow recovery.

The Account Unlock is a configurable capability. If enabled (using the **unlockAccountAfterPasswordReset** property on the Internal ID Store), when the **PasswordResetChanger** completes, the account unlocks.

Operation Description	Operation Types	Endpoint URLs
Self Service Password Recovery		
Get Verified Credentials	POST	`https://${SSP_FQDN}/{{tenantName}}/identity/v1/VerifiedCredentials`
Password Reset Initiator	POST	`https://${SSP_FQDN}/{{tenantName}}/identity/v1/PasswordResetInitiator`
Password Reset Validator	POST	`https://${SSP_FQDN}/{{tenantName}}/identity/v1/PasswordResetValidator`
Password Reset Changer	POST	`https://${SSP_FQDN}/{{tenantName}}/identity/v1/PasswordResetChanger`
Self Service Account Unlock		**Note**: If Self Service Account unlock is enabled, Account Unlock is executed once the Password Reset API sequence completes.
Verified Credentials return only primary, verified credentials	GET	`/identity/v1/VerifiedCredentials?primaryOnly=true/false`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/identity/v1/PasswordResetChanger`
  should look similar to:
  `https://example.com/default/identity/v1/PasswordResetChanger`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Sample Get Verified Credentials Request:

```
{
    "subject": "lhaley",
    "type": "email,sms"
}
```

Sample Get Verified Credentials Response:

```
[
```

```
    {
        "displayValue": "la*******@example.com",
        "deliveryType": "email",
        "credId": "5d69daad-ee81-4504-9daf-20228226f600"
      },
    {
        "displayValue": "xxxxxx5555",
        "deliveryType": "sms",
        "credId": "e8db1ce9-0a2a-4e1b-b617-1be09968d95f
    }
]
```

Sample Password Reset Initiator:

```
{
    "subject": "lhaley",
    "deliveryType": "email",
    "credId" : "e8db1ce9-0a2a-4e1b-b617-1be09968d95f",
    "language": "en_US"
}
```

Sample Password Reset Response:

```
{
    "token": "7cjbMij1IRcTcFD8hkq9lkVQzssrlgVJ"
}
```

NOTE
A token is normally not returned in the response. This sample response shown was returned when the cluster is deployed in test mode. The token is normally delivered to the verified email or phone only.

Sample Password Reset Changer request:

```
POST https://${SSP_FQDN}/{{apiPathTenant}}/identity/v1/PasswordResetChanger?
token=7cjbMij1IRcTcFD8hkq9lkVQzssrlgVJ
```

Configuring the Token Delivery for Password Reset and Account Unlock

Password Reset and Account Unlock token delivery support only SMS and Email. You may set this configuration to SMS, email, or both using the `"selfServiceAccountNotificationStrategy": "email, sms"` property.

```
Endpoint URL: /admin/v1/InternalIdStoreConfig
Operation: GET
{
    "idStoreMappingStrategy": "user_loginid=${idprofile.user_loginid}",
    "disambiguationQuery": "user_loginid eq ${loginid} or email eq ${loginid}",
    "unlockAccountAfterPasswordReset": false,
    "useMaskedCredentialValues": false,
    "passwordResetInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiatePasswordReset",
    "passwordResetChangerUrl": "${system.issuerUrl}/ui/v1/signin/passwordReset",
    "accountUnlockInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiateAccountUnlock",
    "allowLocalPasswordForFederatedUser": false,
    "usePriorityToDeterminePrimary": false,
    "selfServiceAccountNotificationStrategy": "email, sms",
    "customUserAttributes": []
}
```

Setting-Up the Demo Environment

Learn how to set up and run the Symantec Identity Security Platform demo environment.

This section outlines the sample demo environments that are offered by the solution, and the steps to deploy and configure these environments for testing.

Prior to deploying and configuring the demo environment for multi-factor authentication, it is crucial to familiarize yourself with key concepts. This includes understanding the embedded sample directory within the environment, reviewing sample policies, and understanding the various use cases available. You can find detailed information regarding these aspects in the following links:

- IDSP Concepts (includes use of Internal Identity Store, and Sample policy)
- Understanding the Demo Use Cases
- Deploying Demo Web Application
- Deploying Demo Mobile Application (Android and iOS)

Sample Applications

IDSP provides sample Web and Mobile applications to try out the authentication features.

Web Application

The Sample Web demo application discusses how to set up a demo environment and test the various multi-factor authentications flows offered by the solution. The demo environment can be set up by deploying the Sample application. To set up your web demo environment, see Deploying Demo Web Application.

Mobile Application

The Sample Mobile application aims to demonstrate an easy and complete framework for testing multi-factor authentication flows by using a mobile app. The Sample Mobile application is called "*Hedmoral"* application and is available in both, Android and iOS platform. See Deploying the Demo Mobile Application for more details.

Out of the Box - DemoClient

The IDSP provides a DemoClient app that is seeded with a client id and secret out of the box. This app performs the following tasks for you and immediately prepares the Sample application for use.

1. Creates an application
2. Creates a policy and associates it to the application
3. Updates the Authorization (AuthZ) policy with the required privileges

The DemoClient credentials are stored as a secret in the IDSP cluster. We have incorporated the step to retrieve the DemoClient client id and secrets from Kubernetes, in the web and mobile application topics respectively. In web app, see Deploying Sample App in IDSP Namespace, and under Deploying the Sample Application and in Mobile app, see section, Obtaining Client Id and Secret.

Symantec Identity Security Platform Concepts

Learn about Symantec Identity Security Platform concepts, Sample DIT, and Sample Policy.

This topic includes the following:

- IDSP Concept
- Internal Identity Store
- Sample Policy Used for Demo

IDSP Concepts

Before trying out the use cases in the demo environment, we recommend you to go through these concepts:

- Risk
- Policy
- Factors
- Step-Up functionality

Use of Internal Identity Store in the Demo Environment

The demo environment comes with an embedded Internal Identity store. Users and Groups along with demo-client app gets created in this environment. You can make use of these sample users to test the solution's multi-factor authentication capabilities. To know more about the sample users, see Demo Users.

Sample Policy Used for Demo

The sample policy comprises of a set of ***conditions*** defined in ***rules*** that govern the workflows that users must comply with, to access a protected resource. This sample policy is provided to users who are interested in implementing the use cases that are mentioned here.


```
{

        "policyId": "ead57040-d5bc-42be-ac41-91d7a5e8f50f",

        "policyName": "Feature Policy for Demo client",

        "description": "Feature Policy Demo client",

        "status": "active",

        "matchAnyApp": false,

        "apps": [

            {

                "name": "DemoClient",

                "id": "52caabed-c3ea-4e6a-bb0f-afda7029dded"

            }

        ],

        "rules": [

            {

                "id": "5e83badc-1e0e-45f8-ab80-0da55a050a6a",

                "conditions": {

                    "principal": {
```

```
            "group": {
                "operator": "in",
                "value": [
                    "Human Resources"
                ]
            }
        },
        "action": {
            "operator": "in",
            "value": [
                "authenticate"
            ]
        },
        "channel": {
            "operator": "in",
            "value": [
                "web",
                "mobile"
            ]
        }
    },
    "result": {
        "effect": "allow",
        "reAuthenticate": "false",
        "obligations": [
            "PASSWORD:1",
```

```
                        "EMAILOTP:2",
                        "SMSOTP:2",
                        "FIDO:2",
                        "SECURITYKEY:2"
                    ],
                    "msg": "This condition is for the Human Resources Group with PW+OTP or PW+FIDO or PW
+SECURITYKEY on the Web and Mobile Channels. Login with edunn for autoenrol and erussell for inline enrol."
                }
            },
            {
                "id": "a8eca05b-2e0b-4165-ad2a-e9985160dbf4",
                "conditions": {
                    "principal": {
                        "group": {
                            "operator": "in",
                            "value": [
                                "Employees",
                                "Security Administrators"
                            ]
                        }
                    },
                    "action": {
                        "operator": "in",
                        "value": [
                            "authenticate"
                        ]
                    },
```

```
                    "channel": {
                        "operator": "in",
                        "value": [
                            "web"
                        ]
                    }
                },
                "result": {
                    "effect": "allow",
                    "reAuthenticate": "false",
                    "obligations": [
                        "PASSWORD:1"
                    ],
                    "msg": "This condition is for the Employees Group with Password Only - Login with nbruce. Also for the Security Administrators Group when using OAuth/OIDC Postman flows - Login with mhunter, rchase, or vhickey."
                }
            },
            {
                "id": "faac6d75-a045-49a3-938a-aa5bd626e8bc",
                "conditions": {
                    "principal": {
                        "group": {
                            "operator": "in",
                            "value": [
                                "Employees"
                            ]
                        }
```

```
                    },
                    "action": {
                        "operator": "in",
                        "value": [
                            "commercial"
                        ]
                    },
                    "channel": {
                        "operator": "in",
                        "value": [
                            "web",
                            "mobile"
                        ]
                    }
                },
                "result": {
                    "effect": "allow",
                    "reAuthenticate": "false",
                    "obligations": [
                        "FIDO:2",
                        "SECURITYKEY:2"
                    ],
                    "msg": "This condition is for testing Continuous authentication on the Web Channel. Login
with nbruce and Select Commercial Banking to invoke this Condition."
                }
            },
            {
```

```
"id": "7e47fa2b-0437-439b-a62c-1ddc2f69aaa5",
"conditions": {
    "principal": {
        "group": {
            "operator": "in",
            "value": [
                "Customers"
            ]
        }
    },
    "action": {
        "operator": "in",
        "value": [
            "authenticate"
        ]
    },
    "channel": {
        "operator": "in",
        "value": [
            "web"
        ]
    }
},
"result": {
    "effect": "allow",
    "reAuthenticate": "false",
```

```
                    "obligations": [
                        "PASSWORD:1",
                        "FIDO:1",
                        "SECURITYKEY:1",
                        "SMSOTP:2",
                        "EMAILOTP:2",
                        "MOBILEOTP:2",
                        "PUSH:2"
                    ],
                    "msg": "This condition is PW + OTP/MOBILEOTP/PUSH or FIDO + OTP/MOBILEOTP/PUSH or
SECURITYKEY + OTP/MOBILEOTP/PUSH with choices for the Customers Group on the Web Channel. Login with droth."
                }
            },
            {
                "id": "24388338-069d-4bde-9704-d27281238a61",
                "conditions": {
                    "principal": {
                        "group": {
                            "operator": "in",
                            "value": [
                                "Development"
                            ]
                        }
                    },
                    "action": {
                        "operator": "in",
                        "value": [
```

```
                    "authenticate"
                ]
            },
            "channel": {
                "operator": "in",
                "value": [
                    "web"
                ]
            }
        },
        "result": {
            "effect": "allow",
            "reAuthenticate": "false",
            "obligations": [
                "FIDO:1",
                "SECURITYKEY:1"
            ],
            "msg": "This condition is Passwordless for the Development Group. Login with cmoran."
        }
    },
    {
        "id": "a176822b-f3da-4370-9670-312439a7a579",
        "conditions": {
            "principal": {
                "group": {
                    "operator": "in",
                    "value": [
```

```
                    "Finance"
                ]
            }
        },
        "action": {
            "operator": "in",
            "value": [
                "authenticate"
            ]
        },
        "channel": {
            "operator": "in",
            "value": [
                "web"
            ]
        }
    },
    "result": {
        "effect": "allow",
        "reAuthenticate": "false",
        "obligations": [
            "PASSWORD:1",
            "FIDO:2",
            "SECURITYKEY:2"
        ],
        "msg": "This condition is for the Finance Group PW+FIDO or PW+SECURITYKEY on the Web
Channels. Login with bsteer."
```

```
        }
    },
    {
        "id": "61a34bdd-2a98-48fe-a561-ea24cc2986f5",
        "conditions": {
            "principal": {
                "group": {
                    "operator": "in",
                    "value": [
                        "Employees",
                        "Human Resources",
                        "Finance",
                        "Development",
                        "Customers"
                    ]
                }
            },
            "action": {
                "operator": "in",
                "value": [
                    "authenticate"
                ]
            },
            "riskThreshold": {
                "operator": "ge",
                "value": 60
```

```
                    }
                },
                "result": {
                    "effect": "allow",
                    "reAuthenticate": "false",
                    "obligations": [
                        "PASSWORD:1",
                        "EMAILOTP:2",
                        "SMSOTP:2"
                    ],
                    "msg": "This condition is for Risk > 60 on mobile and web channels. Login with nbruce,
swise, or vburn with untrusted IP."
                }
            },
            {
                "id": "106bbde7-c67c-4a09-bb40-02d4ae446c8a",
                "conditions": {
                    "principal": {
                        "group": {
                            "operator": "in",
                            "value": [
                                "Employees"
                            ]
                        }
                    },
                    "action": {
                        "operator": "in",
```

```
                        "value": [
                            "investment",
                            "business"
                        ]
                    },
                    "channel": {
                        "operator": "in",
                        "value": [
                            "web",
                            "mobile"
                        ]
                    }
                },
                "result": {
                    "effect": "allow",
                    "reAuthenticate": "false",
                    "obligations": [
                        "SMSOTP:2",
                        "EMAILOTP:2"
                    ],
                    "msg": "This condition is for testing Continuous authentication on the Web Channel. Login with nbruce and Select Business Banking or Details to invoke this Condition."
                }
            },
            {
                "id": "4fc77a45-9543-47ed-8c8f-5cb0b0b2f253",
                "conditions": {
```

```
        "principal": {
            "group": {
                "operator": "in",
                "value": [
                    "Employees"
                ]
            }
        },
        "action": {
            "operator": "in",
            "value": [
                "authenticate"
            ]
        },
        "channel": {
            "operator": "in",
            "value": [
                "mobile"
            ]
        }
    },
    "result": {
        "effect": "allow",
        "reAuthenticate": "false",
        "obligations": [
            "PASSWORD:1",
            "FIDO:2",
```

```
                    "SECURITYKEY:2"

                ],

                "msg": "This condition is for the Employees Group PW+FIDO+SECURITYKEY on the Mobile
Channel. Login with nbruce"

            }

        },

        {

            "id": "bcc2b812-e170-4c6d-8305-232b42932b22",

            "conditions": {

                "principal": {

                    "group": {

                        "operator": "in",

                        "value": [

                            "Employees"

                        ]

                    }

                },

                "action": {

                    "operator": "in",

                    "value": [

                        "wealth"

                    ]

                },

                "channel": {

                    "operator": "in",

                    "value": [

                        "web"
```

```
                        ]

                    }

                },

                "result": {

                    "effect": "allow",

                    "reAuthenticate": "false",

                    "obligations": [

                        "MOBILEOTP:2"

                    ],

                    "msg": "This condition is for testing Continuous authentication on the Web Channel. Login
with nbruce and Select Wealth Management to invoke this Condition."

                }

            }

        ],

        "createdDate": "2024-03-14 11:57 AM UTC"

    }
```


Understanding the Demo Use Cases

This section outlines the different multi-factor authentication use cases that are supported by Symantec Identity Security Platform Sample Web Application.

> **NOTE**
> Sample Web application is used to perform these use cases. To know more about it, see Deploying Sample Web application.

This page includes the following topics:

Common Terminologies

This section helps you understand certain terminologies that have been used in the use cases.

1. Multi-factor authentication (MFA)

Multi-factor authentication is an authentication method where a user is given access to a protected resource only after two or more factors have been authenticated. The authentication mechanism is categorized into the following three factors:

Authentication Mechanism	Description	Examples
Something you KNOW (Knowledge)	Things that you know	Password, Pin
Something you HAVE (Possession)	Things that you have. Security Key based authentication	Yubikey, OTP
Something you ARE (Inherence)	Some form of a physical characteristic of you (Biometrics)	Fingerprint

2. Two-factor authentication (2FA)

This is a way of validating the claimed identities of the users by combining two different factors. These factors may include a combination of any two types of the following authentication:

1) something they know, 2) something they have, or 3) something they are.

3. User Profile

When users attempt to log into a secured application having access to sensitive applications, belonging to groups such as HR that perform sensitive tasks, untrusted IP's, or have higher risk scores based on the risk factor evaluation, all such users are identified with varying levels of risks and therefore, classified into three different categories:

- High-Risk User Profile- A user profile with a higher risk score is considered a high-risk user profile. This type of user profile leads to higher challenges and most definitely requires multi-factor authentication.
- Medium Risk User Profile - A user profile with a medium risk score is considered a medium-risk user profile. This type of user profile is needed to undergo a two-factor authentication.
- Low-Risk User Profile- A user profile with a low-risk score is considered a low-risk user profile. This type of user profile is mostly requested for single-factor authentication.

4. Auto Enrolment

Auto Enrolment is a process wherein users are automatically enrolled in the IDSP when their Personally Identifiable Information (PII), such as phone number and/or email ID, is updated in the User Directory prior to performing any form of authentication. Only the OTP factor can be auto-enrolled.

5. Inline Enrolment

This is a process whereby users can immediately register their credentials in IDSP, if they have not done so before performing any form of authentication. Users can complete this process by following the prompts in their mobile or web application, whichever device is used to authenticate. If the users are entitled to multiple authentication factors, then they are prompted to select which factor they would like to enrol. When the inline enrolment process completes, user credentials are updated in the solution. The next time users log in, they will be presented with the authentication factors for which they have enrolled their credentials. For more information, see Configuring Inline Enrolment.

Demo Users

The following table details the demo users, and the demo use cases that can be executed for testing the environment.

Full Name	Username	Groups	Notes	Authentica
Nina Bruce	nbruce ("vburn", "swise" These are alternate users in the Employee group.)	Employees	Web Channel, Low risk	password
			Mobile Channel	Password uses FIDO
			Untrusted IP address and thereby has been identified as a high-risk user profile	Two-factor • Passwo by • SMS or
Enid Dunn	edunn	Human Resources	Medium risk	Two-factor • Passwo by • Email C
Emily Russell	erussell	Human Resources	Medium risk Both Web and Mobile Channels	Two-factor • Passwo • OTP or
Brenda Steer	bsteer	Finance	High risk	Two-factor • Passwo • Biomet
Christopher Moran	cmoran ("rrex"- This is an alternate user in the Development group)	Development	Frictionless Authentication	Password- scanner or
Dean Roth	droth	Customers	Migrate to using password-less authentication	• Passwo • Biomet • Securit • Push A

Web Use Cases

Business Owners with Low-Risk User Profiles require a Low Level of Assurance to access Application

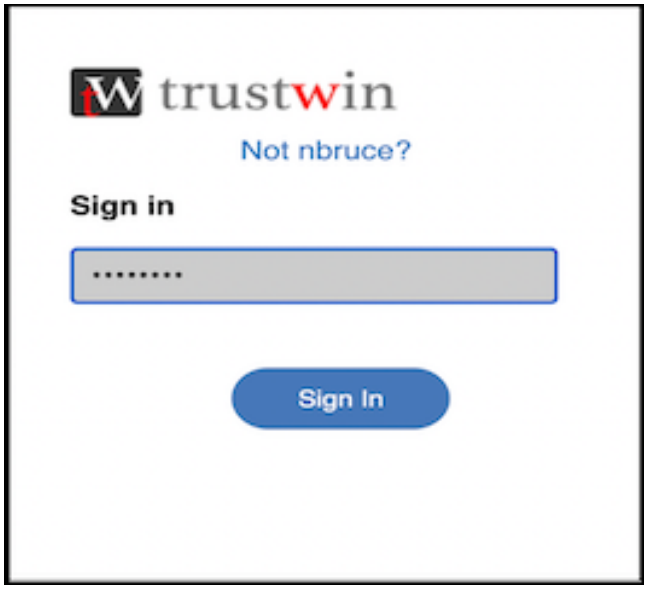
Assumption:

Nina Bruce (nbruce) is identified as a low-risk user by the IDSP. Therefore, she is asked to authenticate with minimum factor, 'Password' to access the Sample application. Her email Id and phone number are updated in the user directory prior to the authentication.

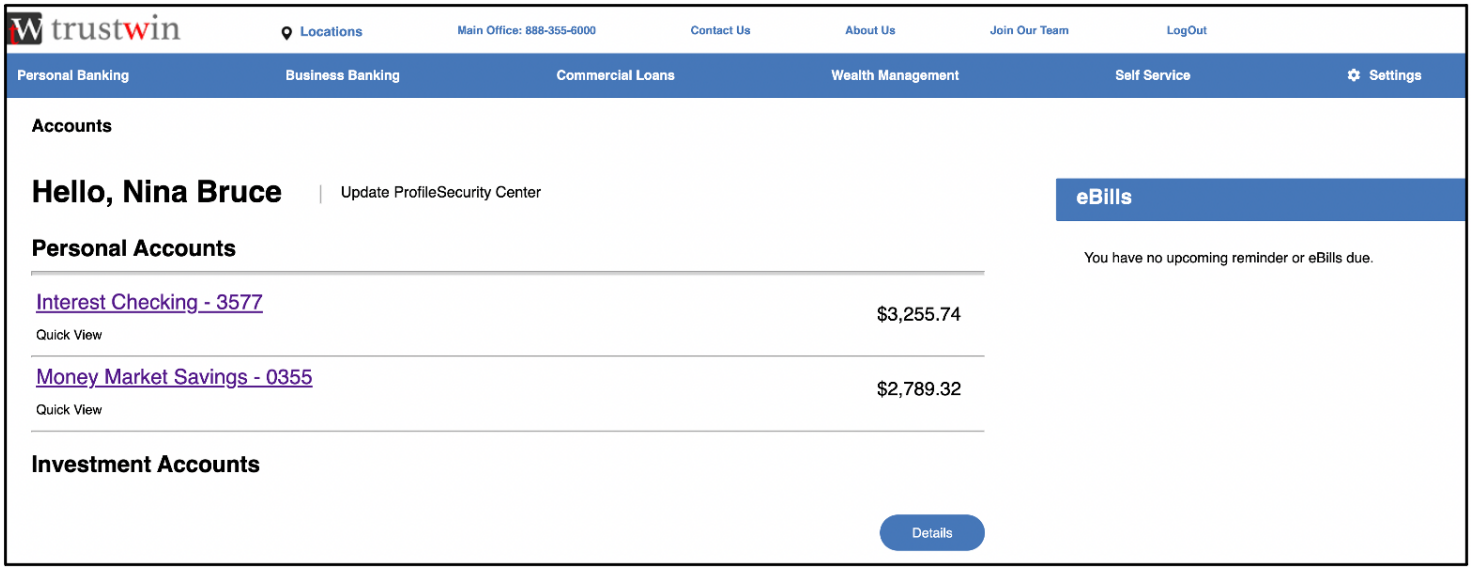
User Flow:

Nina Bruce logs into the Sample application with ‘Password’ (Something she knows) and gets access. She is not prompted to any other form of authentication. The following figure shows the Sign In page where she enters her username.

She enters her password to authenticate.



Nina gets access to the login page after successfully authenticating with her password.



Security Admins want to protect access to Personally Identifiable Information (PII) and therefore requires users to undergo Two-Factor Authentication

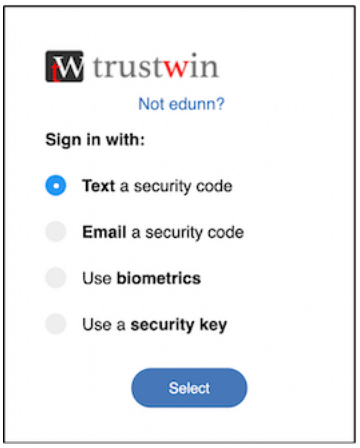
Assumption:

Enid Dunn (edunn) is a member of the HR Team and has access to Personally Identifiable Information (PII) information of customers. Therefore, she has been requested to undergo two-factor authentication, performing Password authentication as the first factor, followed by Email OTP or FIDO as the second factor. She has registered her phone number in the user directory prior to the authentication, which was used by the solution to create her SMS OTP credential. She goes ahead with SMSOTP as the second factor.

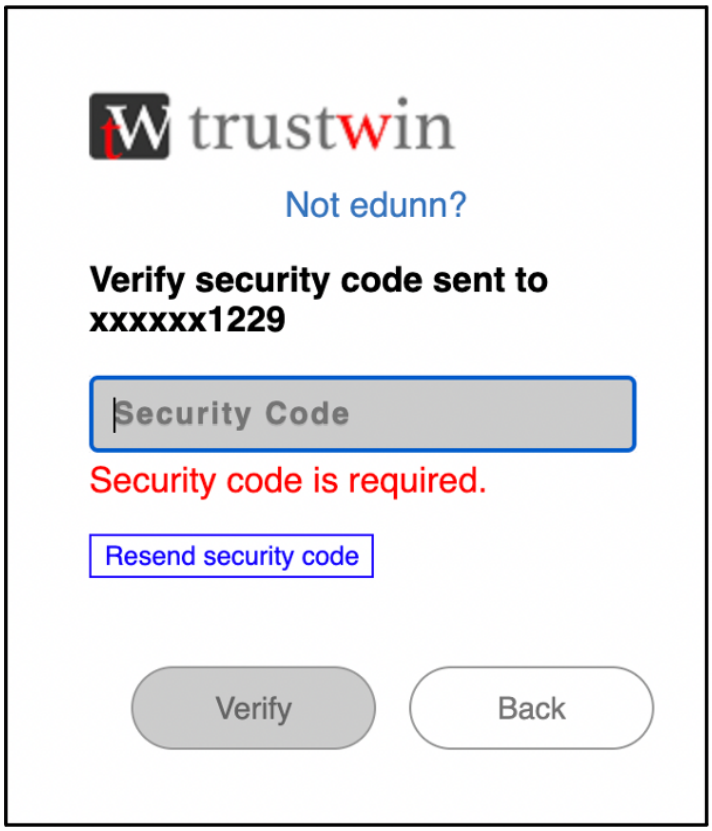
User Flow:

Enid Dunn logs into the sample application using a password (something she knows). Upon successfully authenticating with the password, she is presented with the following screen, where she is required to select a second factor. She opts for SMSOTP authentication (something she has).

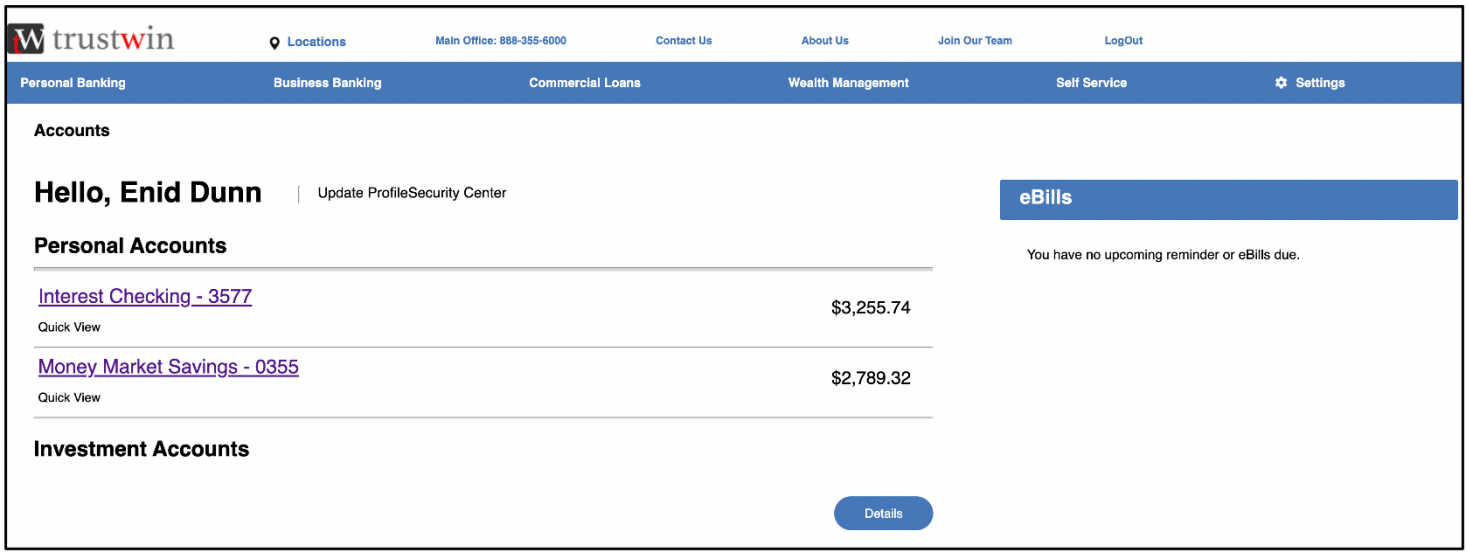
In the following figure, Enid selects email as the credential type where she gets the OTP delivered



Here, she selects "Text a security code" as the credential type and the OTP gets delivered to her registered number.



She enters the code, successfully authenticates the second factor, and gains access to the Sample application. The following figure shows the Sample app page to which Enid Dunn gets access after authenticating with SMSOTP.



Business Owners want to provide the option of inline enrolment to the users who have been requested with Two-Factor Authentication

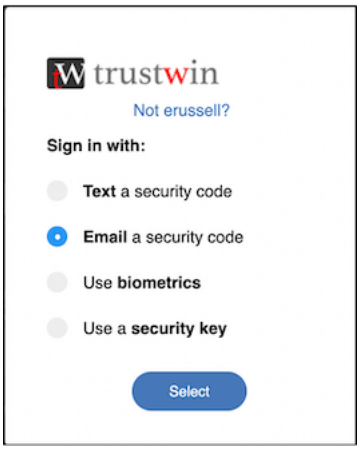
Assumption:

Emily Russell (erussell) is a member of the HR team and has access to Personally Identifiable Information (PII) of customers. Therefore, she has been requested to undergo two-factor authentication, performing Password authentication,

followed by OTP or FIDO. Her OTP credentials, such as phone number or email ID have not been registered in the user directory so she is prompted to perform an inline enrolment.

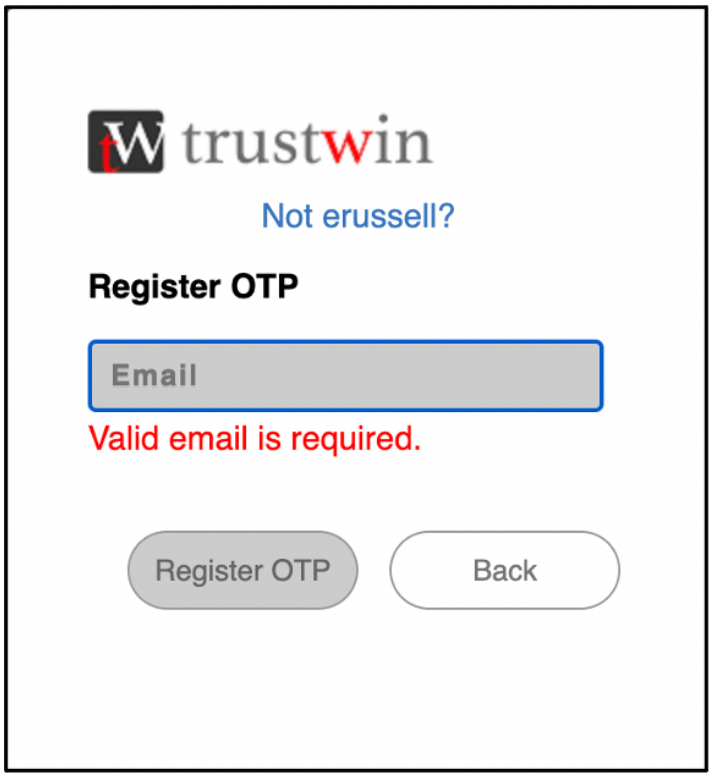
User Flow:

Emily Russell logs into the Sample application using a password (something she knows). Upon successfully authenticating with the password, she is presented with the following screen, where she is required to select a second factor.



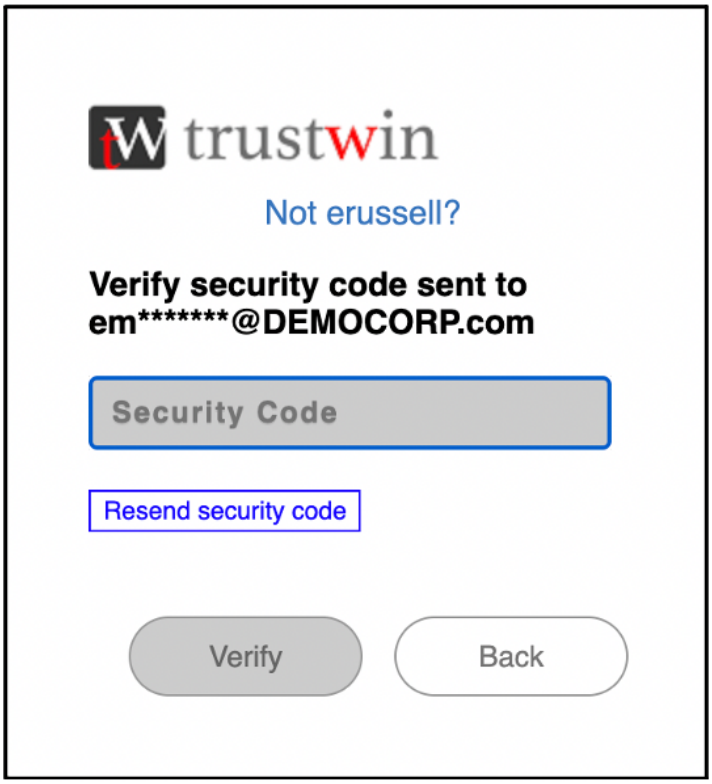
She opts for Email OTP authentication (something she has), therefore, she is prompted to register her Email OTP credentials through inline enrolment.

The following figure shows the inline enrolment page presented to Emily Russell where she must register her EMAIL to get the OTP delivered.

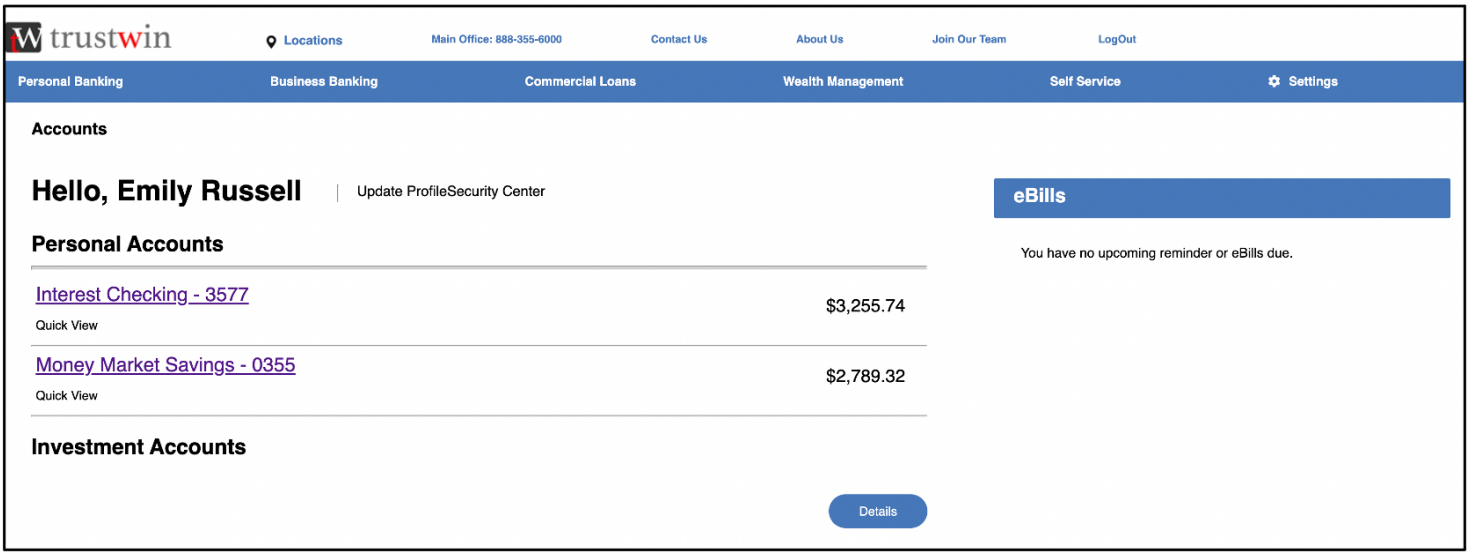


She registers her email Id where the OTP gets delivered.

In the following figure she by entering the email OTP that she received in her email.



The following figure shows the Sample app page to which Emily Russell gets access after authenticating with Email OTP.



Security Admins require Multi-Factor Authentication for High-Risk Users

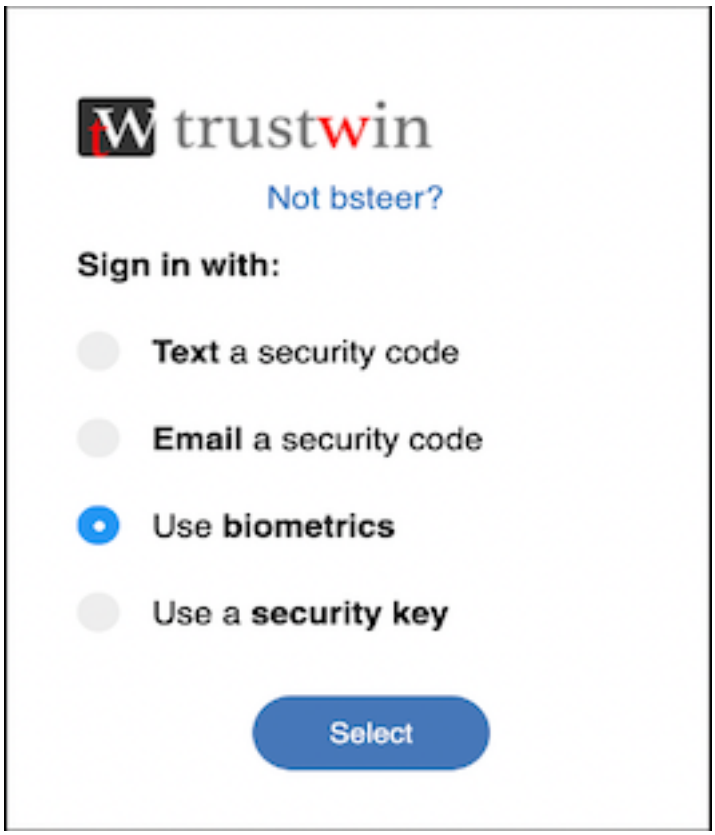
Assumption:

Brenda Steer (bsteer) is a member of the Finance Team and she has been identified as a high-risk user. She has been requested to undergo two-factor authentication, performing Password authentication, followed by

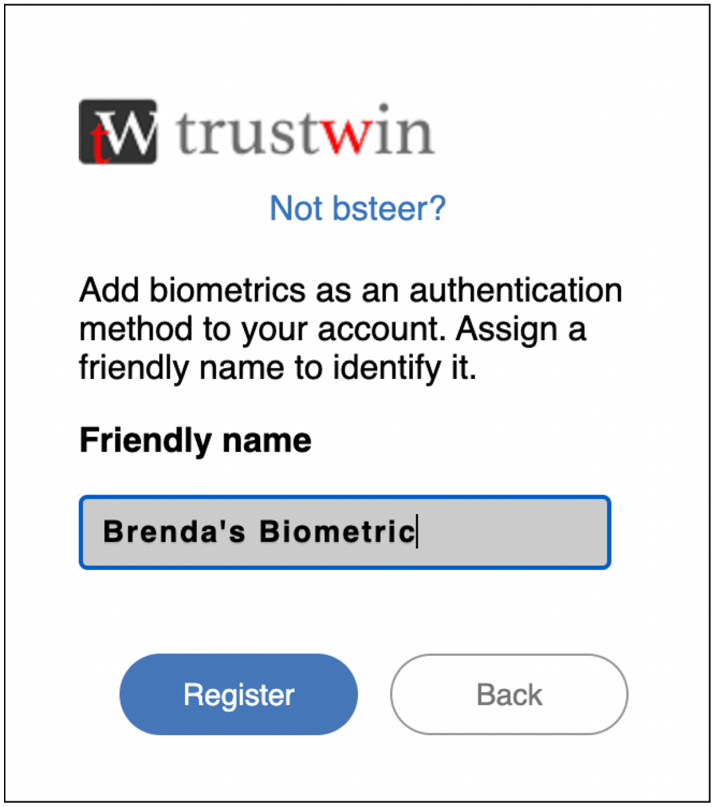
Biometrics authentication using her fingerprint. She is a new user and has not registered her Biometric credential in IDSP, so she is prompted to enroll inline.

User Flow:

Brenda Steer logs into the application using her password. Upon successfully authenticating with her password, she is prompted to select a second factor to register inline. In the following figure, she opts to register her Biometrics.



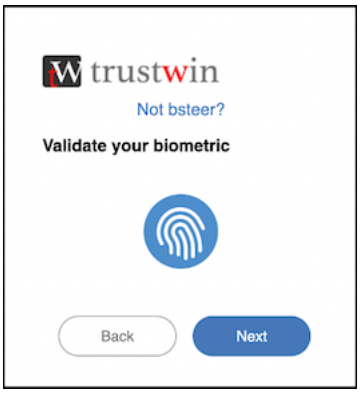
In the following figure, Brenda performs inline enrolment to register her biometrics.



She is displayed with the following screen to validate her fingerprint.

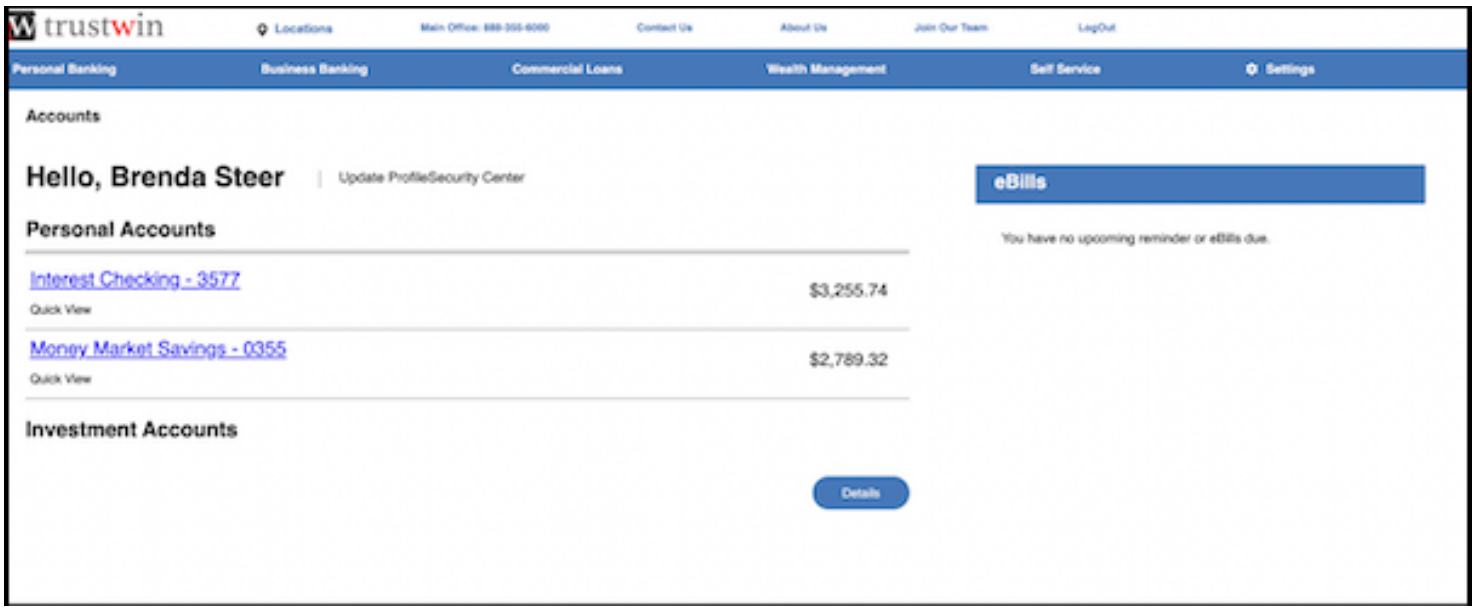
She validates her fingerprint by tapping on the fingerprint scanner that is embedded in her laptop. At this point, her biometric registration is complete.

In the following screen, Brenda clicks Next.



She is displayed with the following validation screen again, but this time it is for second-factor authentication.

Upon successfully authenticating with her fingerprint, she gains access to the protected application.



The next time that she attempts to log in, she will be prompted to enter her password and authenticate with her biometric.

Business Owners require Frictionless Authentication reducing the burden on customers while maintaining strong security

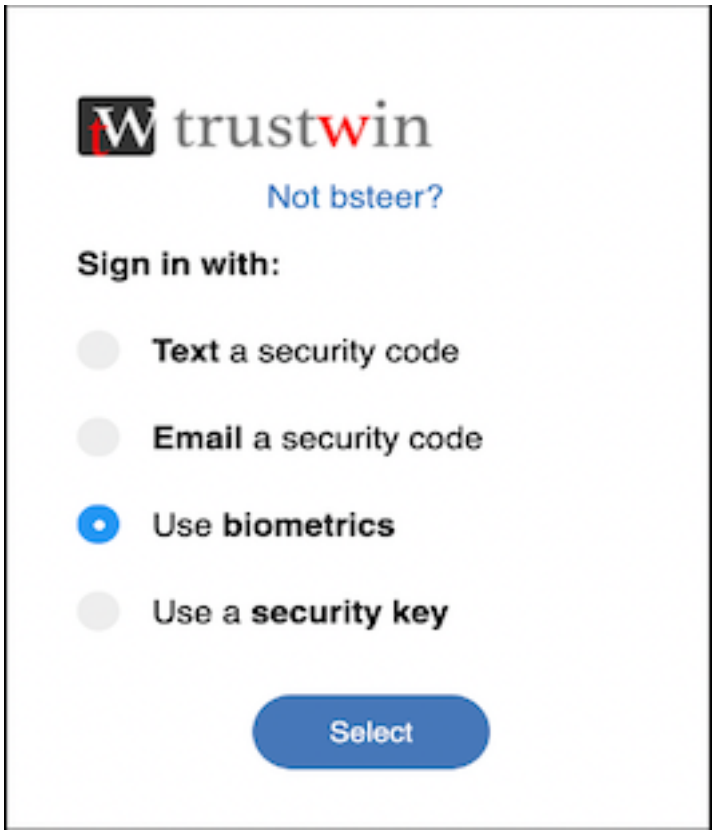
Assumption:

Christopher Moran (cmoran) is a member of the Development Team. He wants to be able to use password-less authentication by using a fingerprint scanner on his Mac computer (FIDO protocol). He is a new user and has not registered his biometric credentials in IDSP prior to the authentication, so he is prompted to enroll inline.

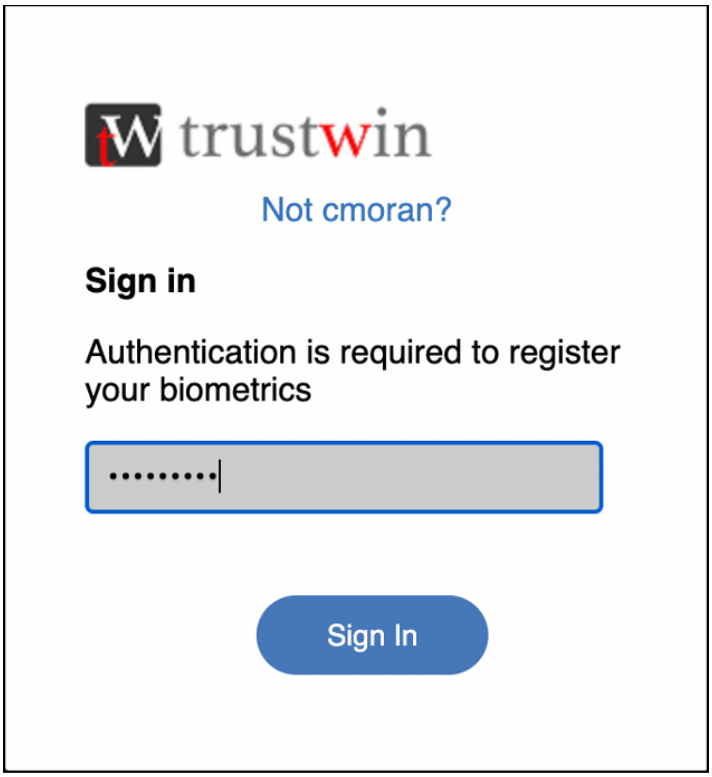
User Flow:

Christopher Moran logs into the Sample application by entering his username and password. After authenticating with the password, he is prompted to select one of the two factors to continue the authentication flow.

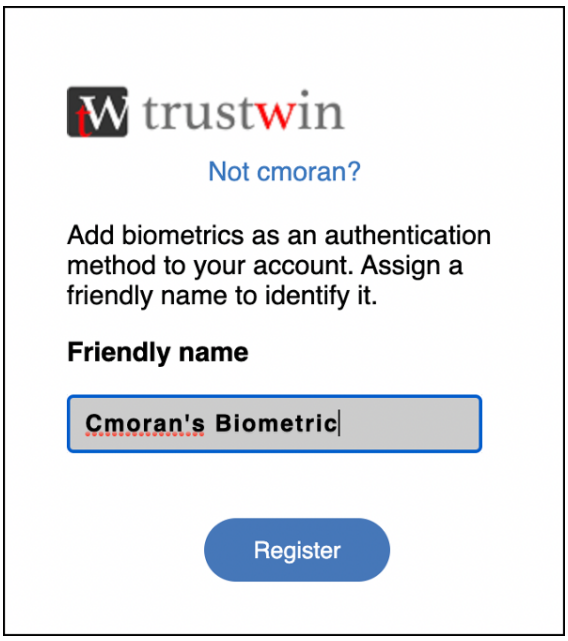
In the following figure, Christopher selects the biometric authentication factor.



Since Christopher has not registered for biometrics, he is required to carry out the mandatory Password authentication as part of the biometrics inline enrolment. The following figure shows the mandatory Password screen where Christopher authenticates using his password, before moving on to biometrics registration.

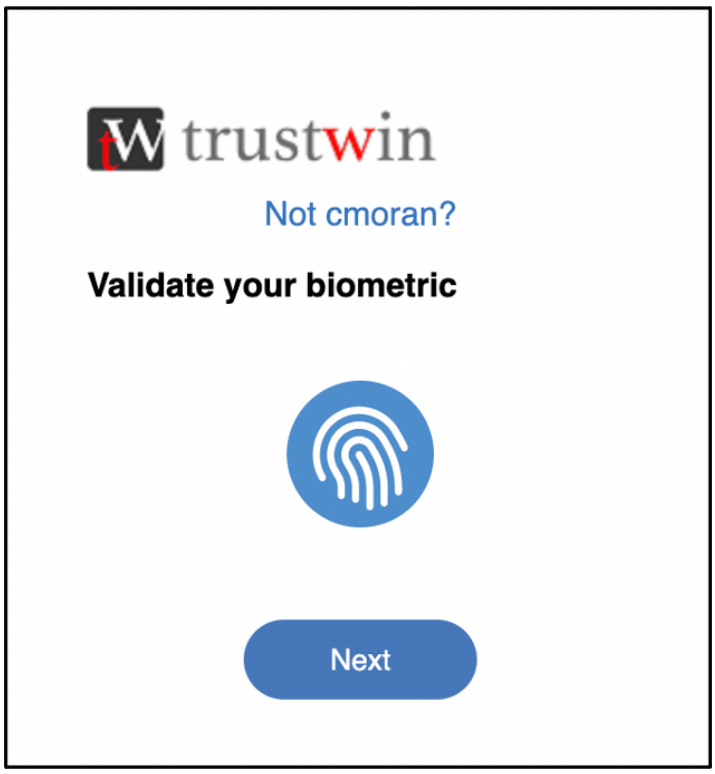


In the following figure, Christopher registers his biometrics with a friendly name to uniquely identify the factor.



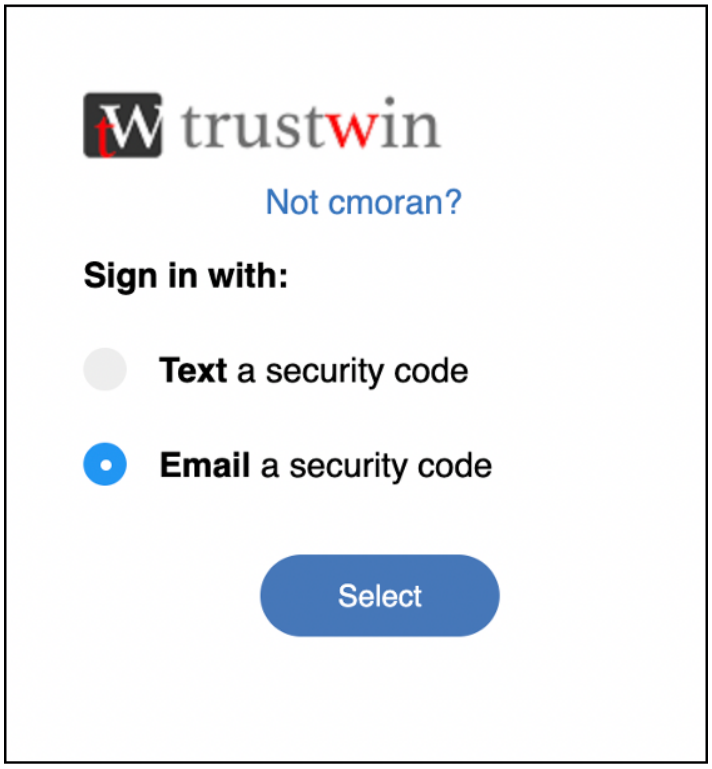
After registering his device for biometrics, he is displayed with the validation screen.

He authenticates by tapping the fingerprint sensor on his Mac computer. At this point, the security key registration is complete. In the following screen, Christopher clicks Next.

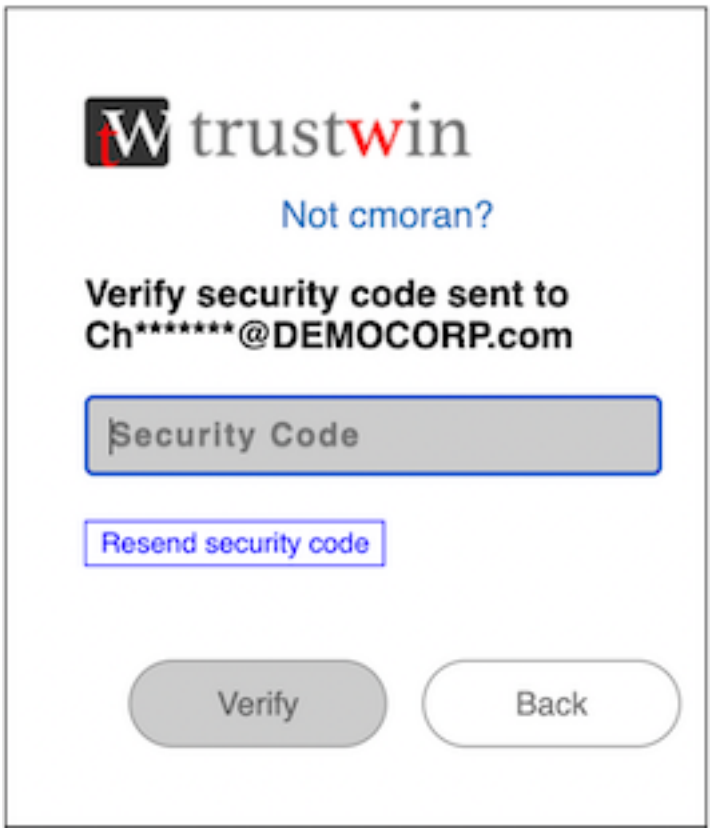


Christopher is displayed with the following validation screen again, but this time it is for second-factor authentication.

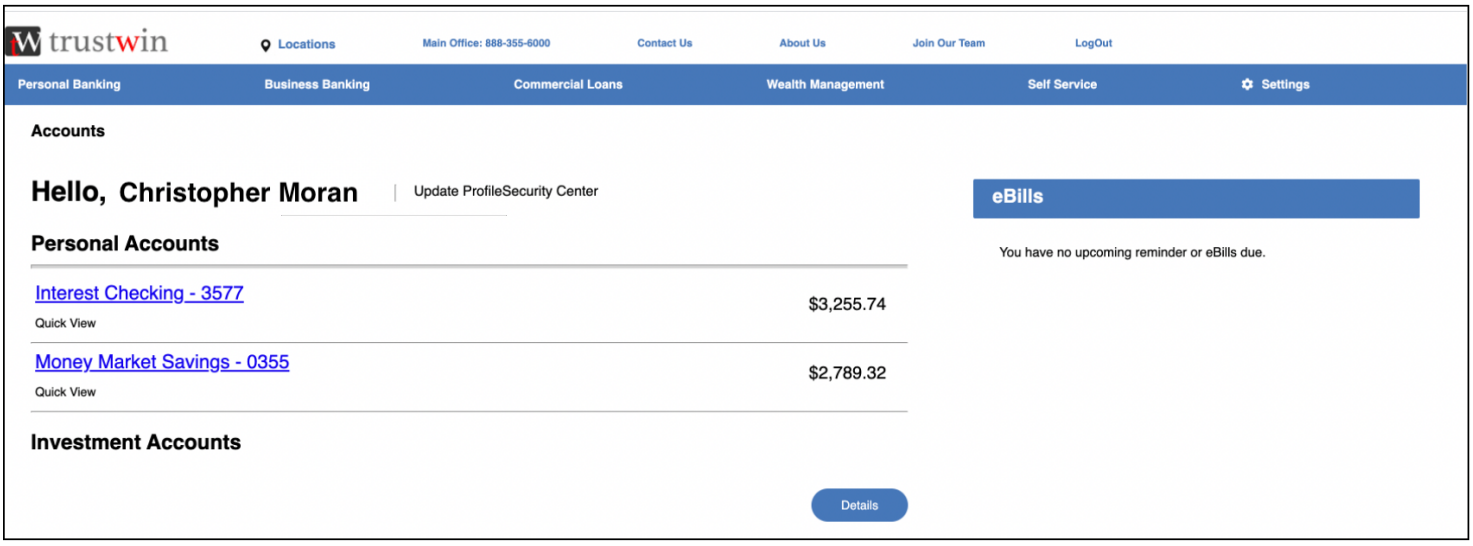
Now, as part of the IDSP risk evaluation process, since Christopher is accessing the Sample app for the first time from this device, he is also asked to authenticate using another factor. He chooses Email OTP as his second factor.



The following figure displays the screen where Christopher has to enter the Email OTP that has been delivered to his email. In the following figure, he enters the OTP received through his email and gains access to the Sample application.



The following figure shows the Sample app page to which Christopher Moran gets access after authenticating with Email OTP.



The next time Christopher attempts to log in, he is prompted to authenticate with his fingerprint alone and not prompted to any other form of authentication.

Step-Up Use Case

Security Admins want to provide increased security for users accessing sensitive assets within an application by initiating a step-up authentication.

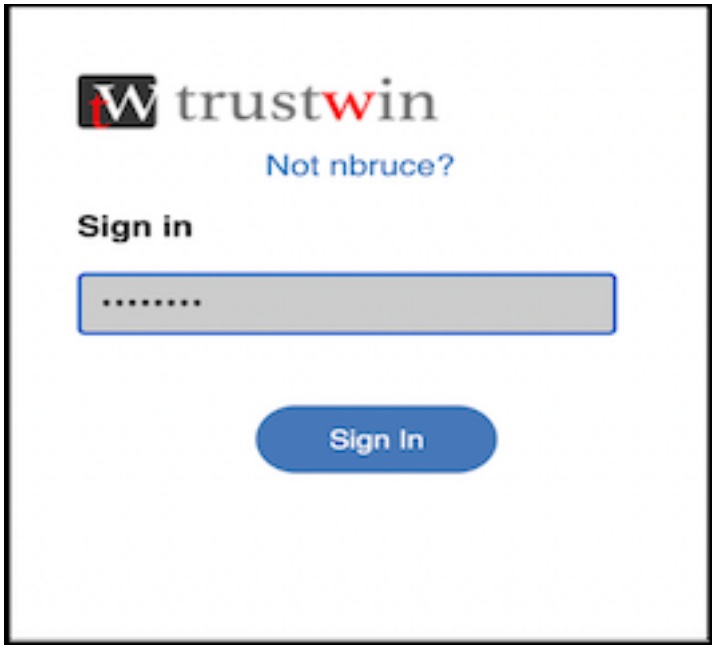
Assumption:

Nina Bruce (nbruce) is identified as a low-risk user by the IDSP. Therefore, she is asked to authenticate with the minimum factor, 'Password' when she initially signs in to the Sample application. However, when she accesses sensitive information on the application, IDSP prompts her for additional authentication. She uses the VIP Access app to authenticate further, which she downloads from the play store.

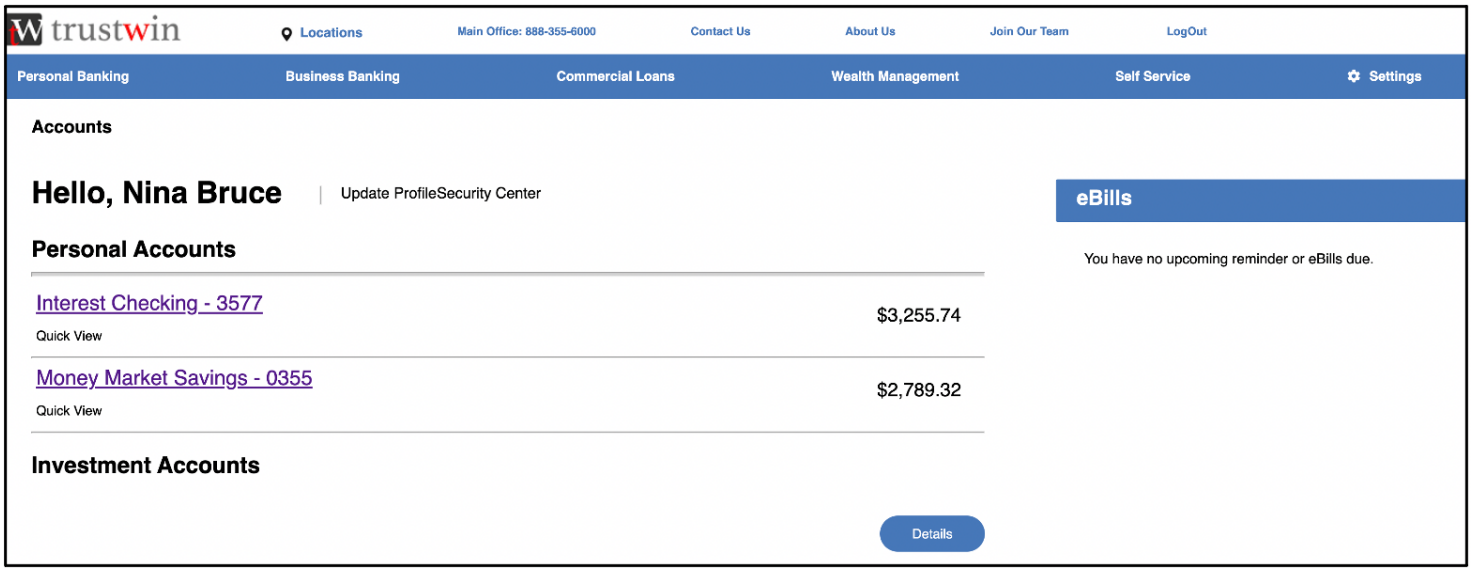
User Flow:

Nina Bruce logs in to the Sample application with only ‘Password’ (Something she knows) and gets access to perform simple tasks. The following figure shows the Sign In page where she enters her username.

She enters her password to authenticate.

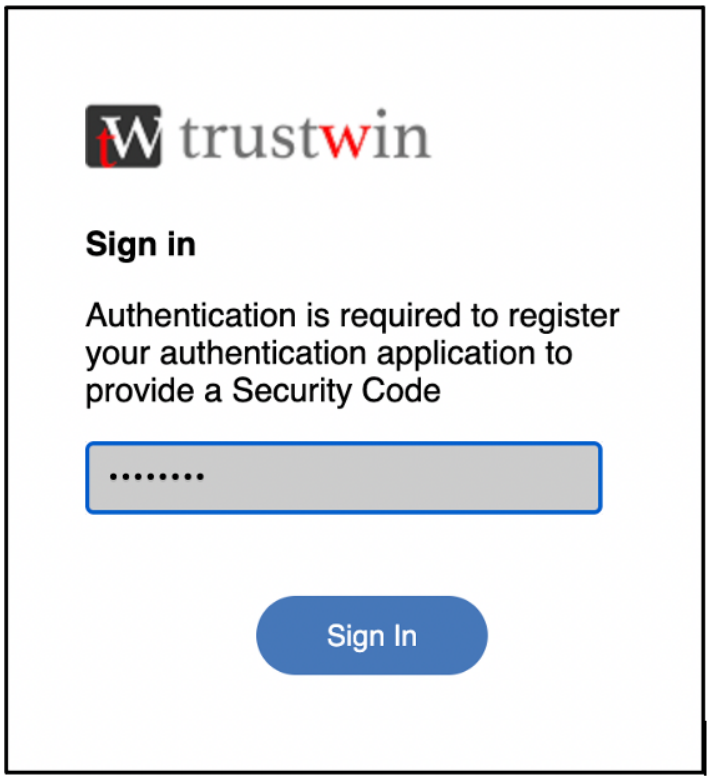


The following figure shows the Sample app page to which Nina Bruce gets access after authenticating with Password.



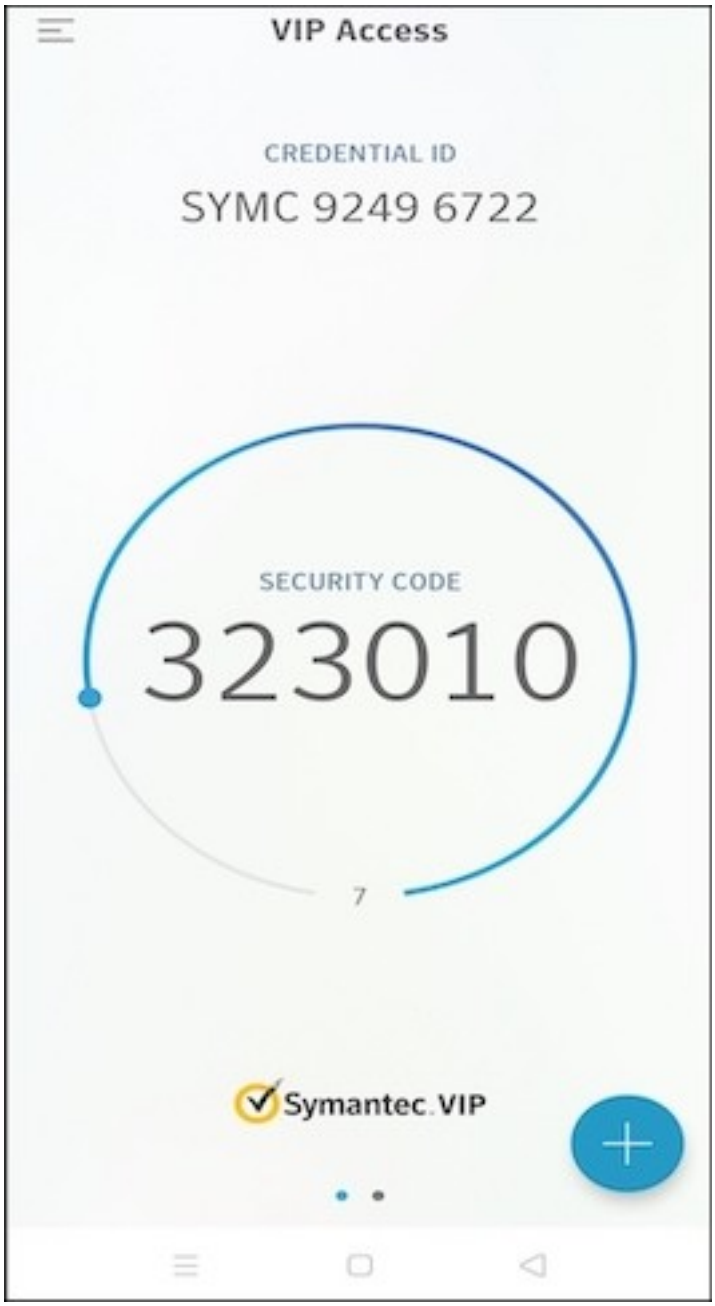
Later, Nina accesses the Wealth Management feature of the Sample application. IDSP recognizes the increased security requirements of her access to sensitive data and prompts her to register inline with MOTP and perform a step-up. However, before performing the step-up, she is asked to perform password authentication as a mandatory step.

The following figure shows the mandatory password screen where Nina authenticates using her password before she moves on to the step-up authentication. But before she does that, she must download the VIP Access app to perform mobile OTP authentication with IDSP.



Screen From VIP Access Mobile App

Nina downloads the Symantec VIP Access App from the play store. Upon installation, the app creates a credential ID for that particular installation and generates the otp. The following is the UI screen of the VIP Access App which generates the credential ID and OTP.

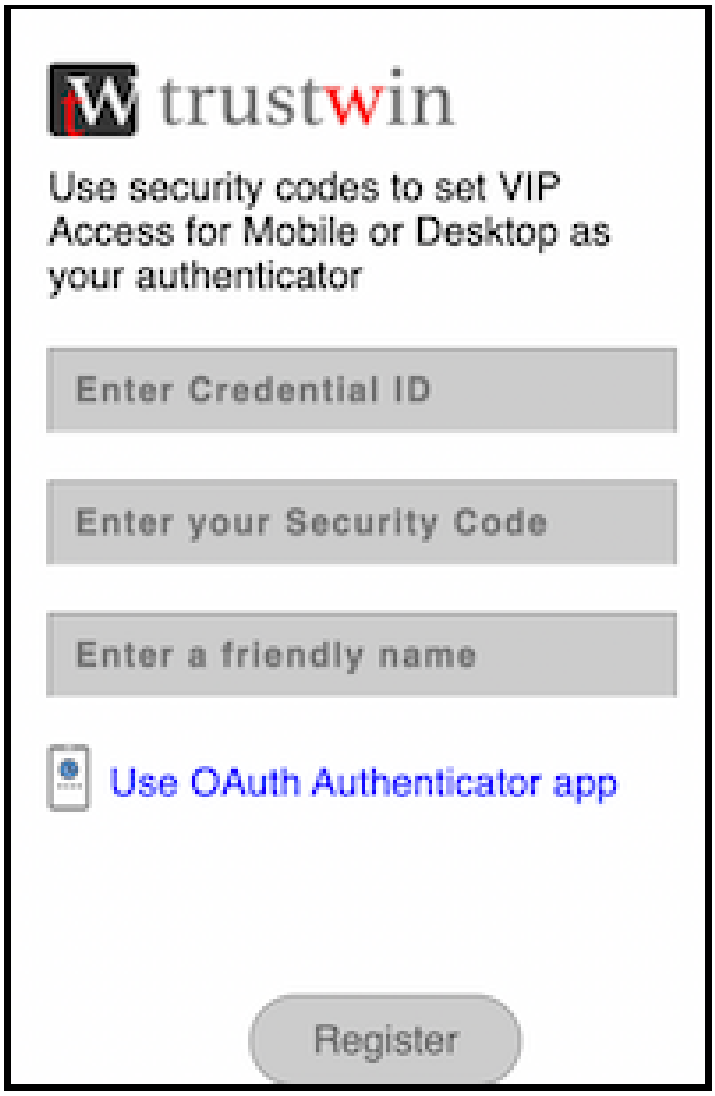


4

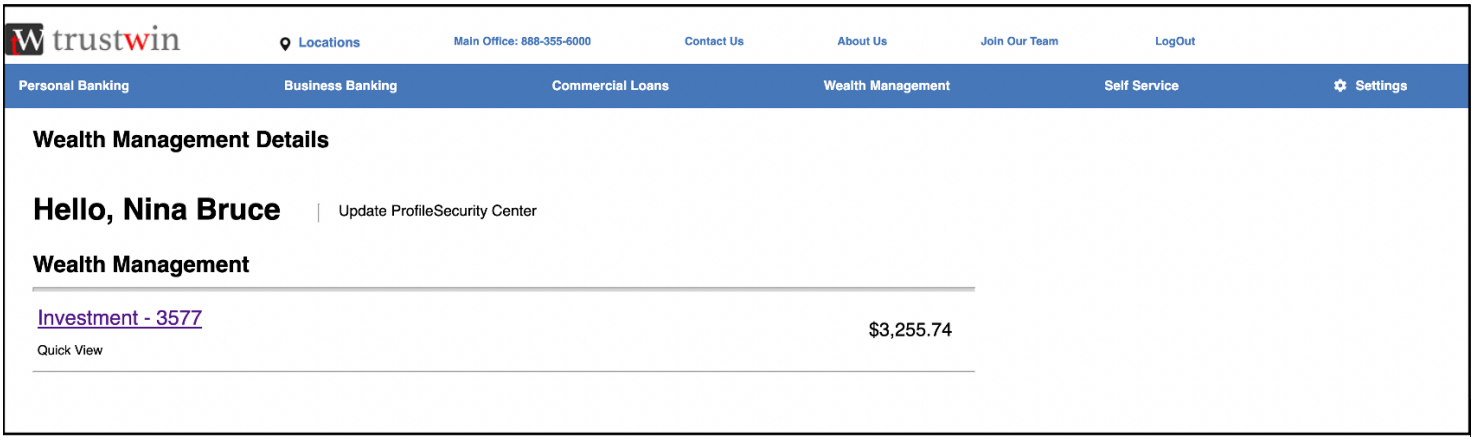
Now coming back to the demo use case, Nina is displayed with the Mobile OTP registration screen where she must enter the following details:

- VIP Credential ID
- Security Code
- Friendly Name

In this figure, Nina enters the details that she obtained from the VIP Access App.



The VIP Access App authenticates the details that are entered by Nina and gives her access to Wealth Management.



Using OAuth Authenticator App (Google Authenticator)

Nina can also use any other OAuth Authenticator app which uses QR code to generate otp and authenticate a request. For example, She can download Google Authenticator to scan the QR code using her mobile.

The following figure shows the QR code screen from the Google Authenticator app. Nina scans the QR code, enters the otp that is received after scanning the QR code, and provides a name to uniquely identify the mobile device. The app verifies the details that are entered by Nina. Upon successful verification, she is allowed to access the Wealth Management page in the Sample app.

Mobile Use Cases

Business Owners require Two-Factor Authentication when using Mobile Applications

Assumption:

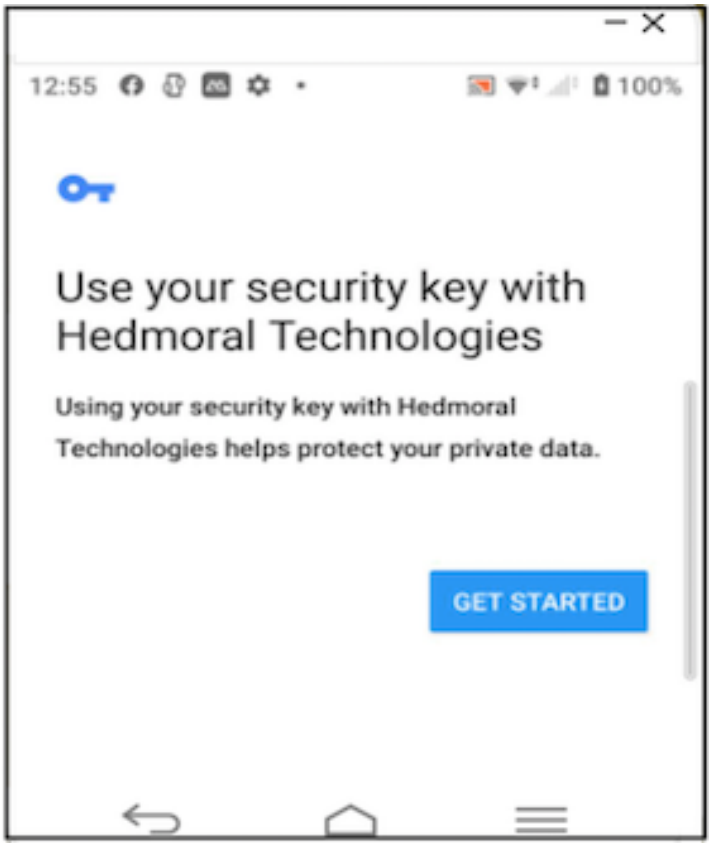
Nina Bruce (nbruce) logs in to the Sample application using her mobile device. She is requested to perform a (step-up) two-factor authentication, password followed by Security Key authentication (which uses FIDO standards). Earlier when she logged in to the Sample application through the web channel, she was only asked to perform a password authentication. Her security key credentials are not registered with the IDSP, so she is prompted to enroll inline.

User Flow:

Nina Bruce logs in to the application using a password, (something she knows). Upon successfully authenticating with a password, she is prompted to inline register her security device for second factor Security Key authentication (something she has).

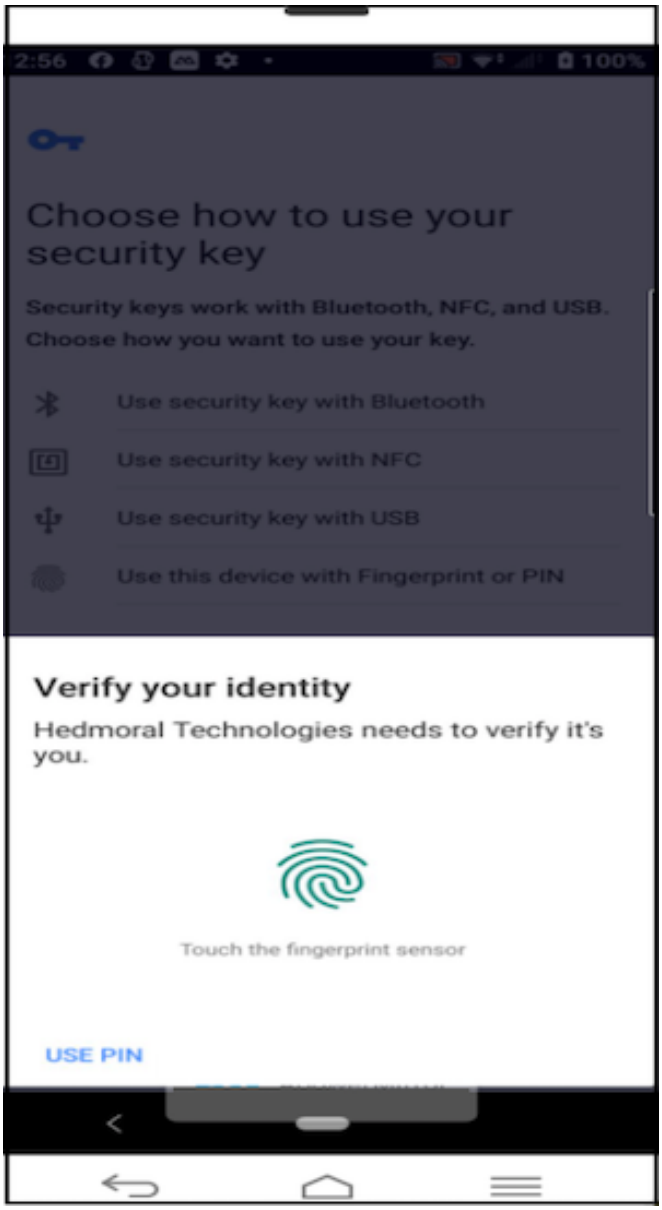
In this image, Nina is registering her security device:

After submitting her device name, she is prompted to provide her security key to complete the registration process. The following screen is displayed in front of her:

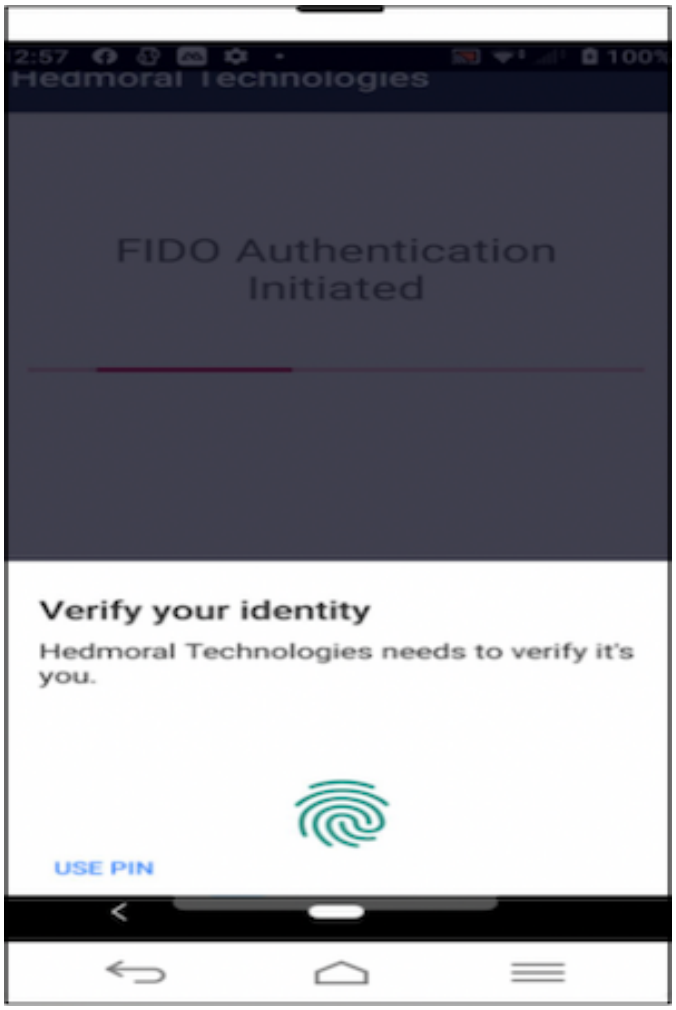


Another screen is displayed to her, where she must select a way in which she uses the security key:

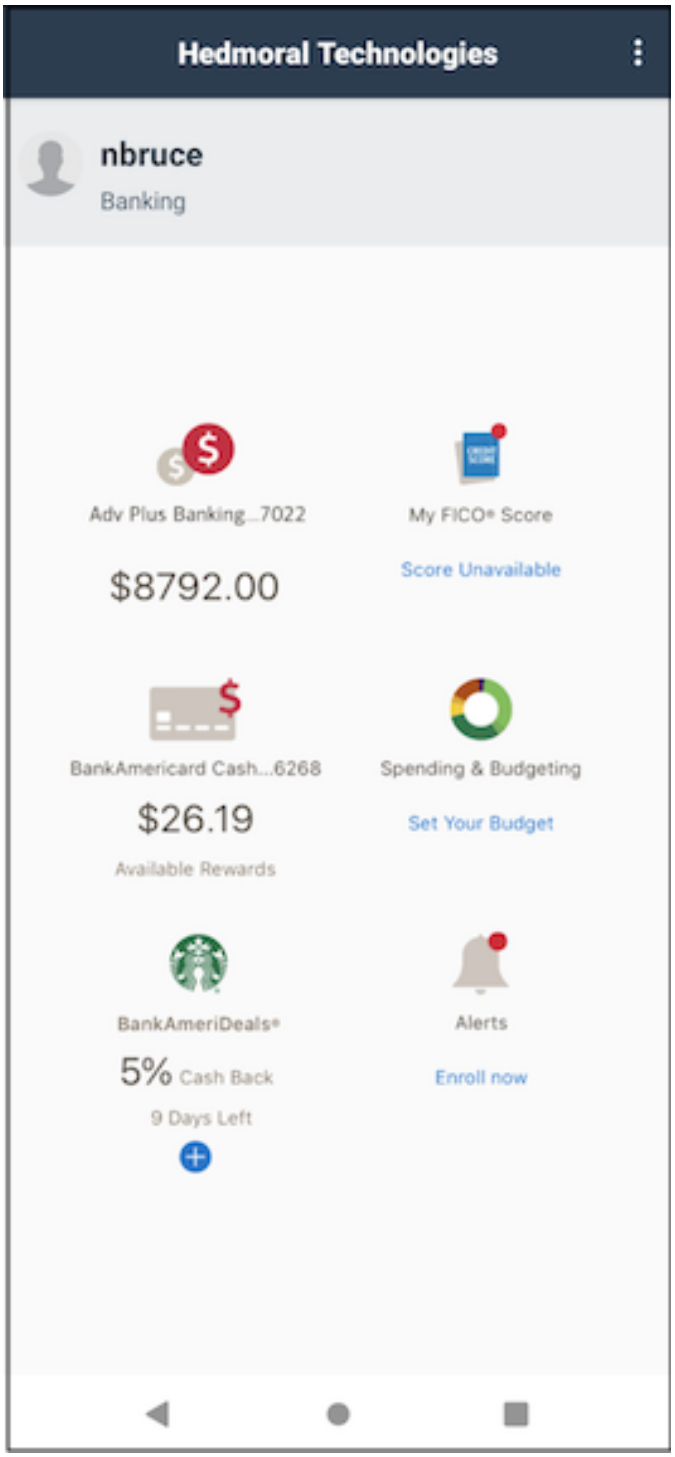
She selects "Use this device with Fingerprint or PIN". Now, she must verify her fingerprint by touching the fingerprint sensor on her device. She is displayed with the following screen, where she verifies her fingerprint:



Now, the IDSP's Hedmoral technology app verifies Nina's fingerprint. She is displayed with the following screen:



Upon successfully verifying her fingerprint, she gains access to the protected application.



The next time that she attempts to log in with the mobile device, she is prompted to enter her password and authenticate with her fingerprint. She is not prompted to any other form of authentication.

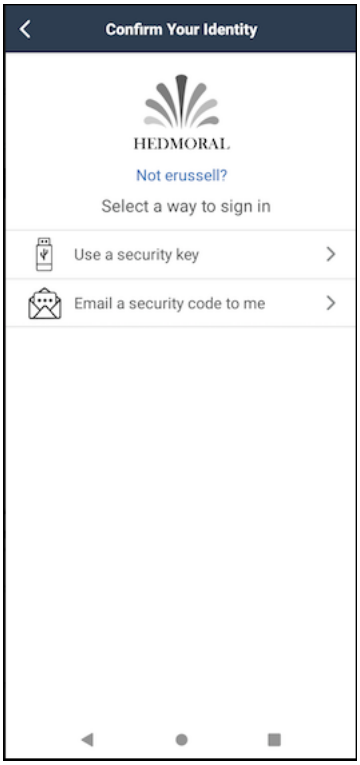
Security Admins want to easily build Authentication Policy that works across both Web and Mobile channels, providing a cohesive customer experience

Assumption:

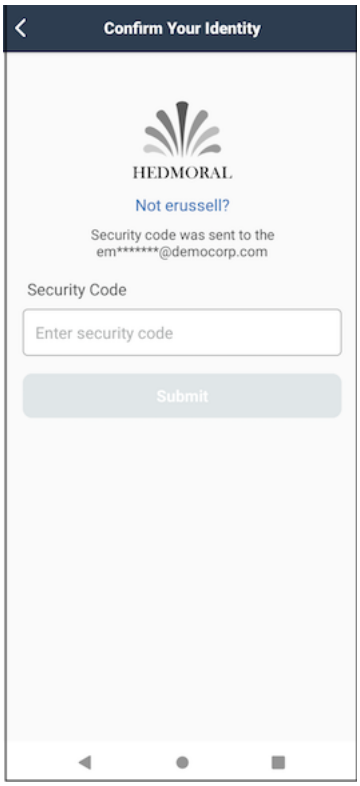
Emily Russell (erussell) is a member of the HR Team. She logs in to the Sample application using Password, followed by OTP factor authentication. When she logs into the Sample application next time using her Mobile device, she is able to leverage the authentication factors that she had registered with the web app.

User Flow:

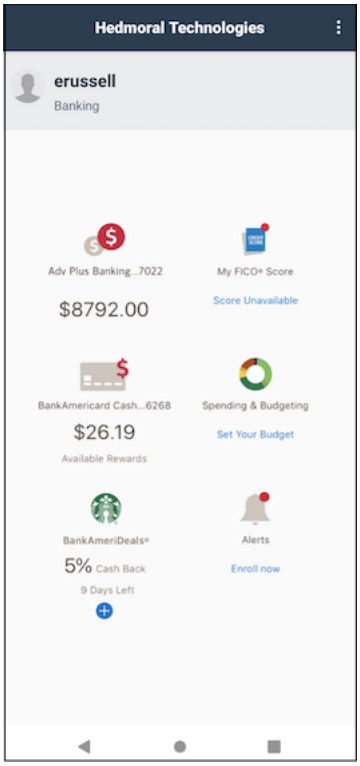
Emily Russell logs into the mobile application using a password. Upon successfully authenticating with the password, she is presented with the following screen where she must select one of the two second-factor authentication options. She selects Email OTP flow.



After she selects Email OTP, she is displayed with the following screen, where she is required to select Email Id and click Submit to get the OTP delivered.



She selects her email id where the OTP gets delivered. After successfully authenticating the second factor, she gains access to the Sample application.



Migration to Password-less

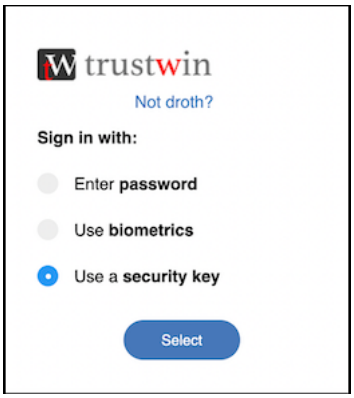
Business Owners would like to have their customers migrate to using password-less authentication but provide the ability in the interim to select either password or password-less factors.

Assumption:

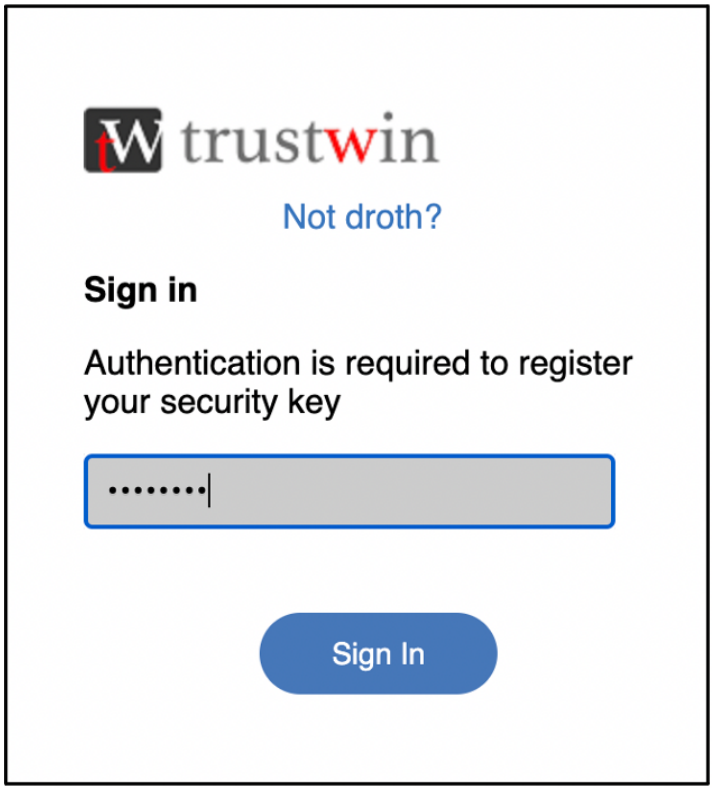
Dean Roth logs into the Sample application and is provided with these authentication factors, password, biometrics, and security key, as his business owner will soon switch away from the password authentication factor. Therefore, he selects the biometric authentication factor and does an inline enrolment to keep himself ready to switch his authentication factor to a password-less factor whenever his business requires him to do so.

User Flow:

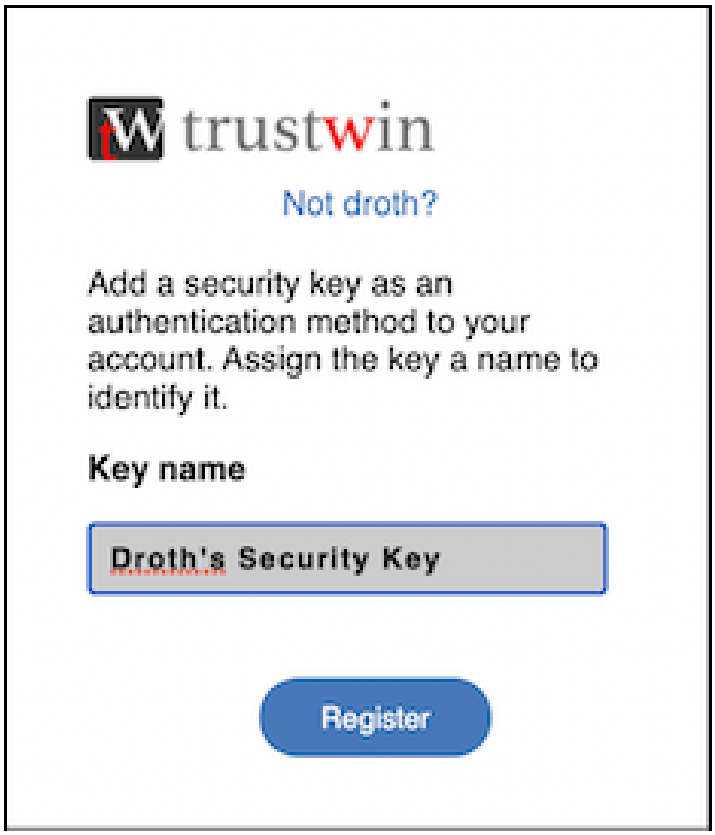
Dean Roth logs into the Sample application and opts to register for the Security Key (password-less). He goes ahead with inline registration from the factor selection list that is displayed to him.



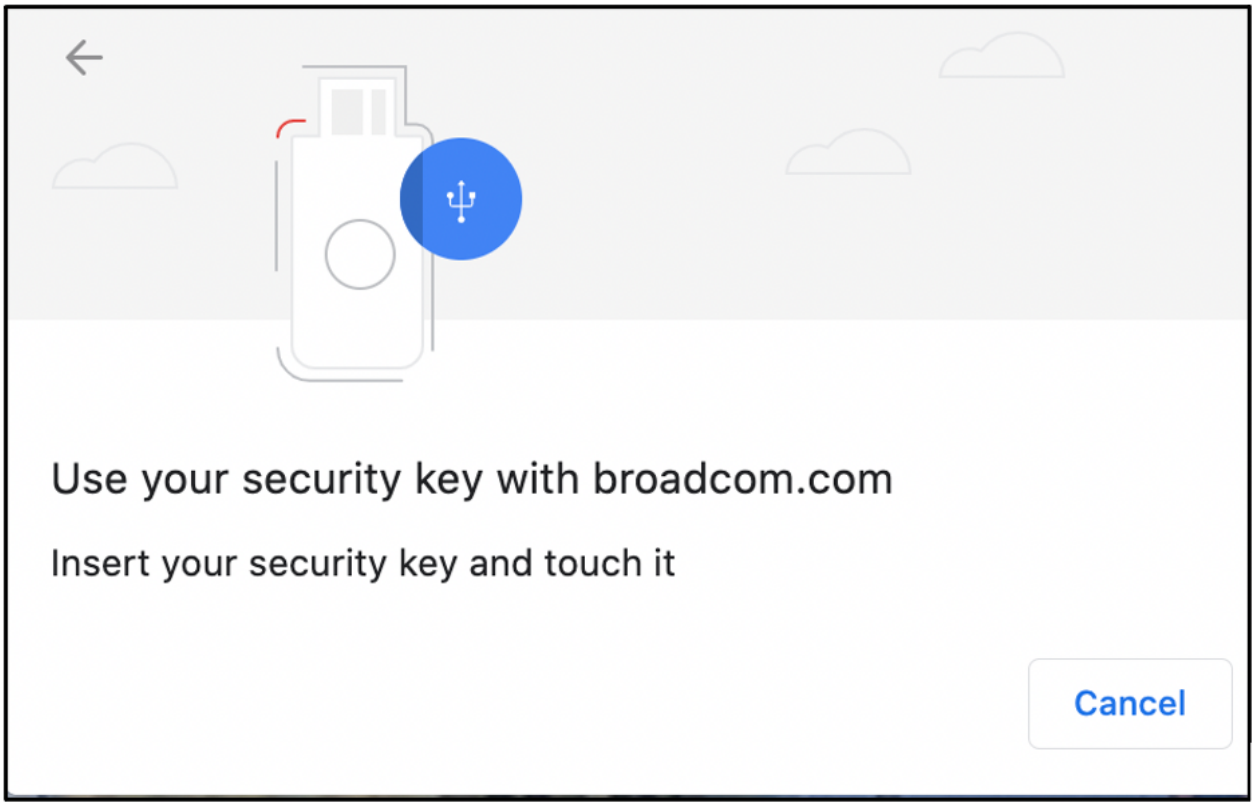
Since Dean has not registered for the security key, he is required to carry out the mandatory Password authentication as part of the security key inline enrolment. The following figure shows the mandatory password screen where Dean authenticates using his password before he moves on to security key registration.



In the following figure, Dean registers his security key on his Mac computer.



After registering his device for security key verification, he is displayed with the validation screen.



He authenticates by tapping the yubikey on his Mac computer. He successfully registers and gets the password-less factor ready for use, whenever needed.

Risk Use Case add images here

Security Admins want to increase security (step-up) when users access applications from untrusted networks (such as hotels or coffee shops)

Assumption:

Nina Bruce (nbruce) logs into the application from an untrusted IP address and thereby has been identified as a high risk user profile. She is asked to perform additional factor authentication by performing Password authentication, followed by OTP. Usually, she is only asked to perform password authentication. Her email ID and phone number have been updated in the user directory prior to the authentication.

User Flow:

Nina Bruce logs in to the application with a password (something she knows). Upon successfully authenticating with the password, she is prompted to perform an OTP authentication (something she has). She is required to choose either, SMS or EMAIL to get the OTP delivered. She chooses SMS, where the OTP gets delivered. She successfully authenticates with the second factor and gains access to the protected application.

Push Use Case

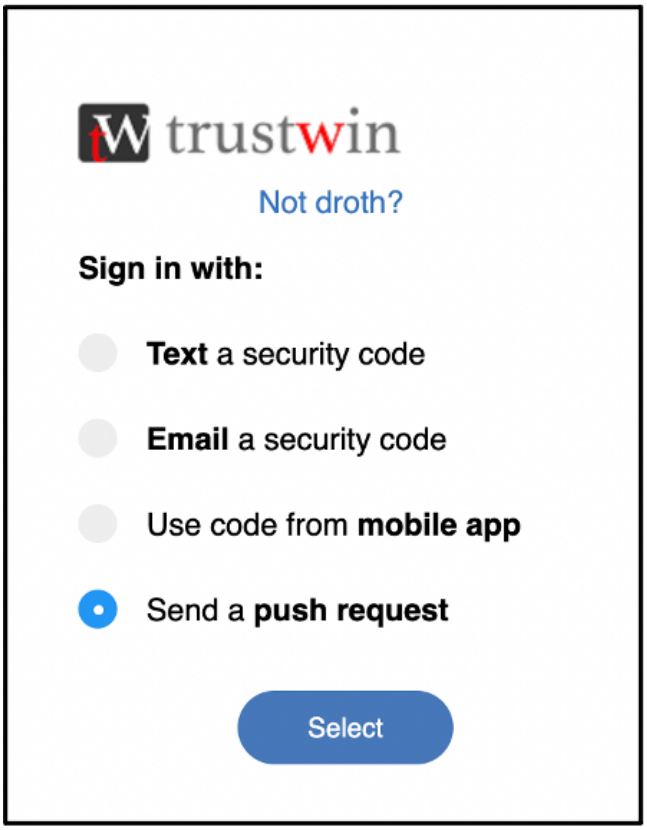
Business Owners want to provide the option of inline push registration to the users who must use two-factor authentication

Assumption:

Dean Roth (droth) belongs to the Customer group and has access to Personally Identifiable Information (PII) of customers. Therefore, he has been requested to enroll for two-factor authentication. He has not registered his credential for two-factor authentication, so he is prompted to perform inline registration of a device to receive push notifications as his second factor.

User Flow:

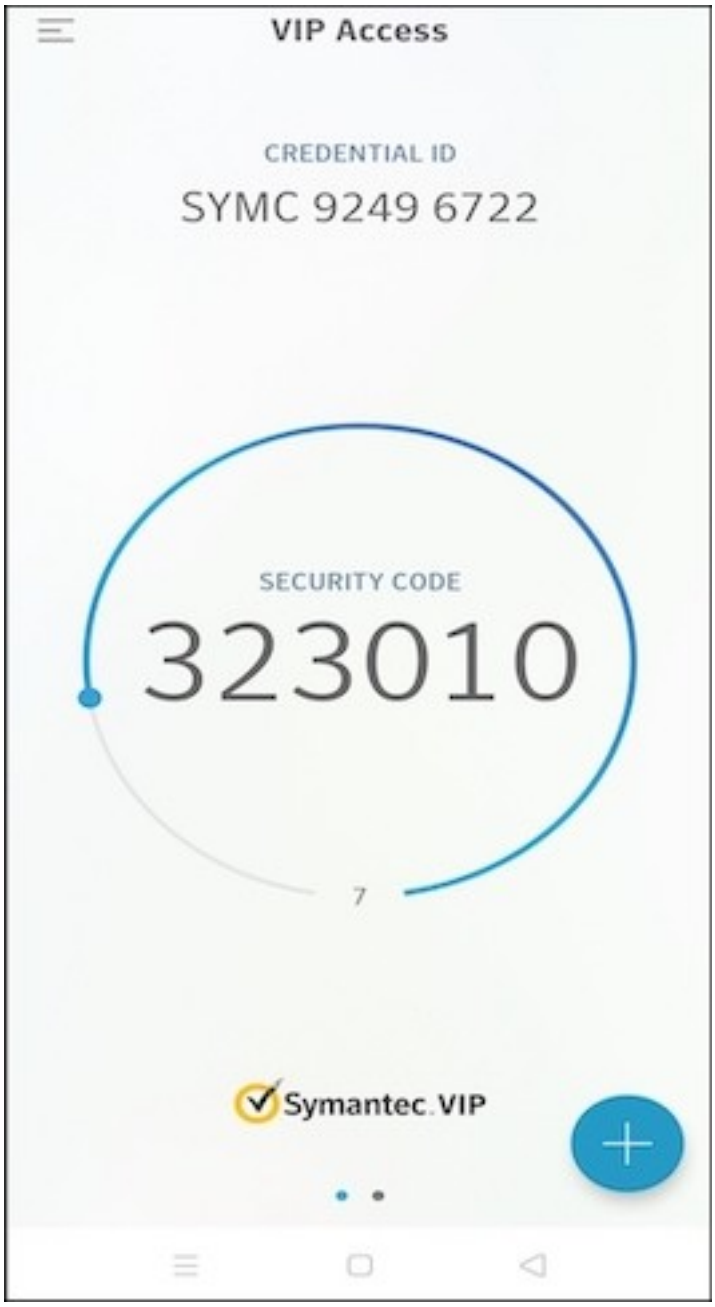
Dean Roth logs into the Sample application with username and password. Upon completing the password authentication, he is prompted to perform a second-factor authentication. He is displayed with the following factors for authentication from which he selects push as his second authentication factor.



Upon successfully authenticating with the password, he is prompted to perform an inline registration of his mobile device to receive push notifications. But before he does that, he must download the VIP Access app to perform push authentication with .

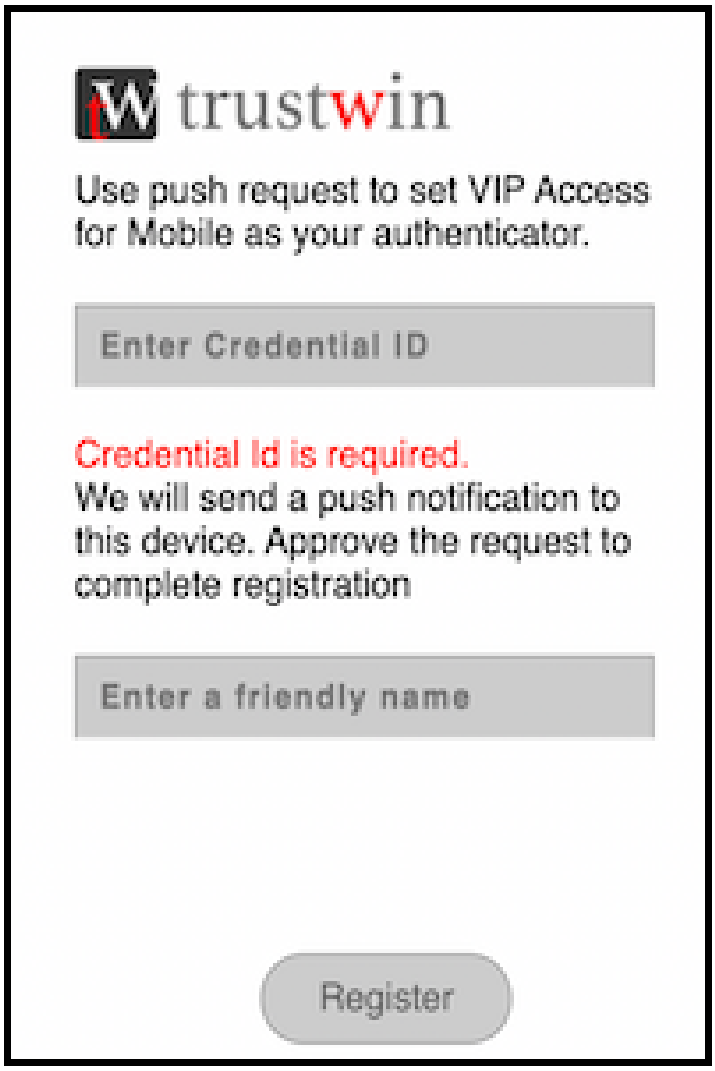
Screen From VIP Access Mobile/Push App

Dean downloads the VIP Access App from the play store. Upon installation, the app creates a credential ID for that particular installation and generates the OTP. The following is the UI screen from VIP Access App which generates the credential ID and OTP.



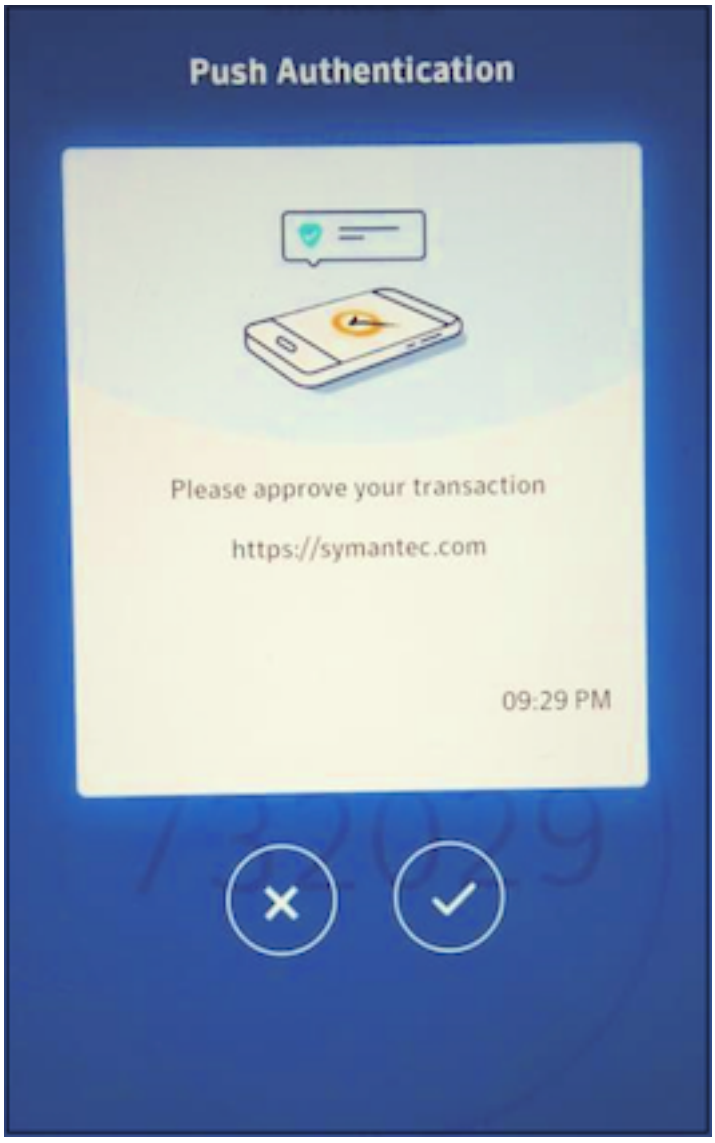
Now coming back to the demo use case, Dean is displayed with the IDSP registration screen where he must enter the following details that he obtained from the VIP Access App:

- Credential ID
- Friendly Name



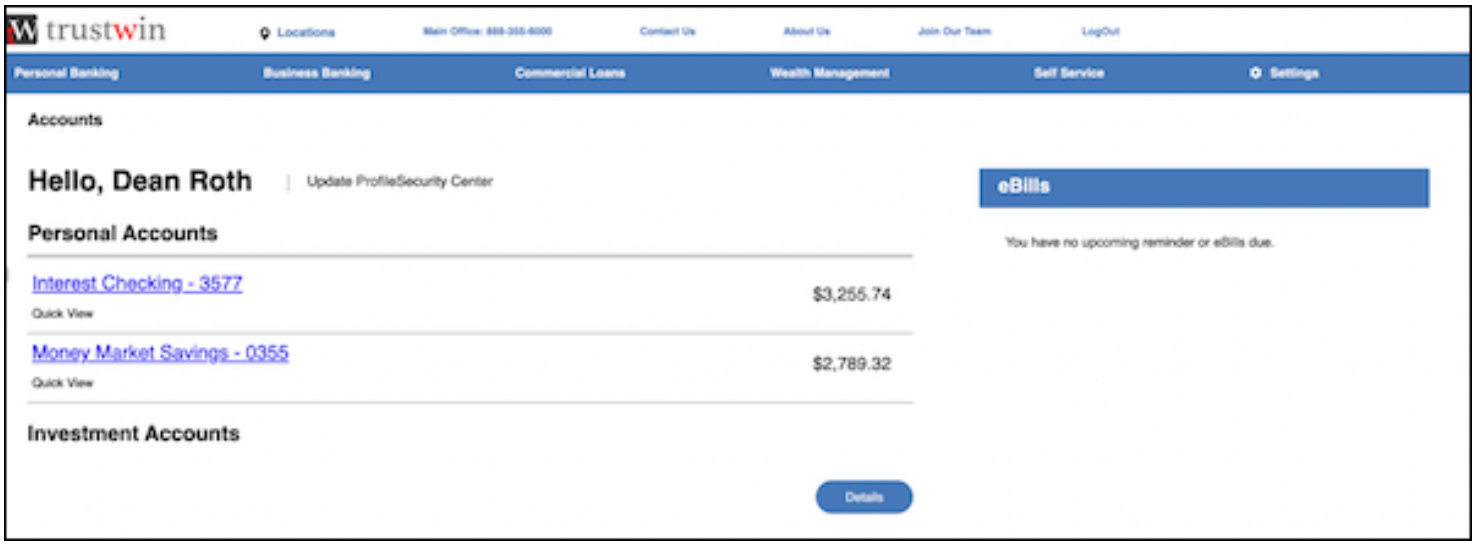
Upon entering the credential Id and name of the device, he is displayed with the following screen. In the meantime, a push notification is sent to Dean's mobile device.

He receives the Push notification on his mobile device which displays the following screen.



He confirms the push message by clicking, ✓.

After confirming the push notification, IDSP registers his mobile device and he gains access to the Sample application. The following screen is displayed to him.



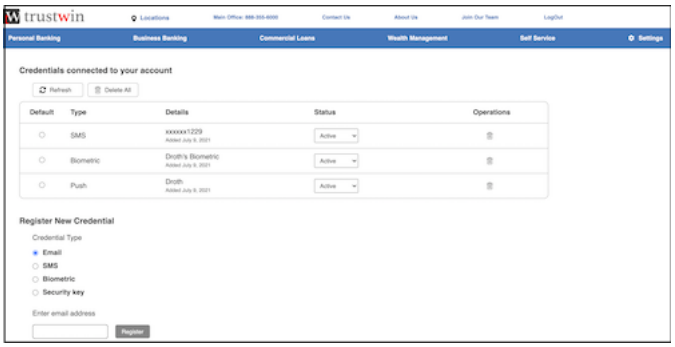
The next time Dean Roth signs in, he enters the password and selects push from the factor selection list. sends push notification to his mobile device which he approves and gains access to the Sample application.

SiteMinder and Use Cases

For SiteMinder and IDSP use cases, see Integration of SiteMinder with IDSP.

Self Service

The Self Service tab in the Sample Web application allows you to create and manage credentials that are linked to your user account.



You can perform the following activities in Self Service:

Checking Credential Status

Use the Self Service tab to check the status of the credentials that are registered with your account. A list of credentials will be displayed in a tabular format with their `Type` , `Details` , and `Status` .

Modify Credential Status

To modify the status of your credential into one of these types; `Active/Inactive` , set the desired choice from the `Status` drop-down in the table.

Registering New Credentials

To add a new credential, select a credential type from the list.

For example, to add an Email Id, do the following:

1. Enter the Id in the box that is displayed and click Register.
2. You are prompted to validate the credential by entering the OTP that has been sent to your Email.
3. Validate successfully with the OTP.
   The newly registered Email Id reflects in the Credential table. The status of the credential will change from `Initial` to `Active` after the validation. The format of the email Id should adhere to the standard protocol.

Deleting Credentials

To delete a specific credential, select the radio button next to the credential and clicking the delete icon under the “Operations” column in the table.

Use the `Delete All` option to delete all your registered credentials at once.

> **NOTE**
> The Refresh button can be used to fetch the latest list of credentials that are connected to your account.

Deploying Sample App

Know about different options to deploy the Symantec Identity Security Platform Sample App.

When you plan to deploy the Sample App for testing the IDSP's capabilities, you must first decide in which mode you would like to deploy the IDSP (SSP). Based on the deployment mode selected for IDSP, you can decide how Sample app can be deployed.

IDSP can be deployed in two modes:

- ***Demo Mode*** - In this mode you can only test the solutions capabilities using sample users and groups. This is a dedicated deployment for demo purposes.
- ***Non-Demo Mode*** - In this mode of deployment, you can perform activities in production. This is the standard IDSP deployment that can be used in a production environment. This environment does not auto create demo artifacts on the default tenant.

The IDSP Sample app can be deployed in both these modes, but to test the solutions capabilities, we recommend to deploy Sample application when the IDSP is deployed in the demo mode.

The demo environment comes with an embedded Internal Identity store. To know more about it, see here.

If you want the ootb demo artifacts (users and applications) to be auto created for you, you can either deploy IDSP in demo mode or create a tenant with demo setup enabled.

Deploy the Sample App in the following ways:

- Deploying Sample App in the IDSP namespace in Demo Mode (auto configured demo deployment).
- Deploying Sample App in the IDSP namespace in a Non-Demo mode by creating a separate tenant to be used only for demo purposes.
- Deploying Sample App in a different namespace in a Non-Demo mode (creating the Sample application, specifying custom URLs etc.).

Deploying Sample App in Symantec Identity Security Platform (IDSP) Namespace

Learn the steps required to deploy a Sample App in the same namespace where Symantec Identity Security Platform is deployed.

Prerequisites

IDSP should be deployed in the demo mode.

Ensure you have the following environment variables set before running the commands in this guide:

Variable	Description
RELEASENAME	The Helm release name of your IDSP deployment.
NAMESPACE	The Kubernetes namespace where IDSP is deployed.
SA_RELEASENAME	The Helm release name for the Sample App.
SA_NAMESPACE	The Kubernetes namespace for the Sample App which is same as the IDSP namespace.
HELM_REPO	The URI of your Helm repository which is same as the IDSP Helm repository.

NOTE
To check the environment variables used during the IDSP deployment, see Configuring the Environment Variables.

Deploying IDSP

- To deploy IDSP in the demo mode, add the following command line parameter to the IDSP helm install command:

```
--set ssp.deployment.size="demo"
```

This information is also covered in this topic: Additional Parameters to consider adding before deploying the IDSP Helm Chart.

When IDSP is deployed in the demo mode, it creates a set of users and groups, and a demo-client app in the default tenant. The demo client's client ID and secret credentials are stored in a Kubernetes secret in this format: `<SSP-RELEASENAME>-ssp-secret-democlient`.

Retrieve and set the client id and secret of the out-of-the-box DemoClient as environment variables, by running these commands on the same Kubernetes cluster where Symantec Identity Security Platform is deployed:

```
export SA_CLIENTID=$(kubectl get secret ${RELEASENAME}-ssp-secret-democlient \
-n "${NAMESPACE}" -o jsonpath="{.data.clientId}" | base64 --decode)

export SA_CLIENTSECRET=$(kubectl get secret ${RELEASENAME}-ssp-secret-democlient \
-n "${NAMESPACE}" -o jsonpath="{.data.clientSecret}" | base64 --decode)
```

Unlike the non-demo IDSP deployment, the Kubernetes secrets containing the default tenant's client Ids and secrets do not get deleted after 48 hours. When deploying the Sample app in the same namespace as IDSP, the demo client secret can be used as a default for the Sample app's client Ids and secrets.

NOTE
If you are deploying the sample-app into the same namespace as IDSP and encountering issues with Kubernetes clusters that return CNAMEs instead of service EXTERNAL-IP (such as Amazon's EKS), and the IDSP fqdn is not registered in DNS but defined locally in the workstation's /etc/host file, the IDSP service IP will not be set automatically. To set it, use "--set ssp.serviceIp=xxx.xxx.xxx.xxx" or use the helm parameter, "--enable-dns" to enable the chart to discover the IP automatically.

Deploying Sample App

When IDSP is deployed in demo mode, deploying the IDSP's SampleAPP chart (ssp-sample-app) can be streamlined.

In this mode, you can deploy the Sample app by specifying the `sspReleaseName: <value>` value in the **Helm values YAML file** or alternatively, you can use the `--set sspReleaseName=<value>` Helm command line parameter in the Sample app helm install command.

Deploying the Sample app chart into the IDSP namespace using the "sspReleaseName" option uses the following default values:

Parameter / Component	Default Value / Path
ClientId and **ClientSecret**	Taken from the IDSP demo client secret.
SSP service host	Taken from the IDSP ingress (the service URL uses the "default" tenant).
ServiceIP	Taken from IDSP ingress.
IDSP Web Sample	sample app host: sampleapp-***ssp-fqdn****/sample-app*
SPI Swagger Docs host	sampleapp-***ssp-fqdn***
Accessing SPI host	sampleapp-***ssp-fqdn***
IDSP Sample RP	• sampleapp-***ssp-fqdn***/sample-rp/home • sampleapp-***ssp-fqdn***/sample-rp/oidc/login • sampleapp-***ssp-fqdn***/sample-rp/saml/login

Information about the demo users and the demo use cases that you can try out in the demo environment are referenced in this page, see Demo Users and Demo Use Cases

(Optional Step) Configuring an Ingress SSL certificate for the Sample App

1. If you do not have a certificate, you can skip this step, as a self-signed certificate will be used when accessing the Sample Web Application.
   If you have a certificate from a trusted Certificate Authority (CA), configure the following environment variables to point to the key and cert files.
   ```
   export SA_KEYFILE="<keyfile>"
   export SA_CERTFILE="<certfile>"
   ```
   Where,
   - <keyfile> - The path to the certificate key file. (private)
   - <certfile> - The path to the certificate file. (public)

   NOTE
   The certificate ***Subject*** or ***Subject Alternative Name (SAN)*** should match the FQDN name of the Sample Web Application. To check that the SAN matches the FQDN name, use the following command:
   ```
   openssl x509 -in ${SA_CERTFILE} -noout -text
   ```
   Once the certificate details are displayed, look for the value of "CN=" in the ***Subject***, or ***Subject Alternate Name*** .
2. If you have a certificate from a trusted CA to use with the Sample Web Application for SSL, then create the secret in the same namespace where the Sample Web Application will be deployed.

   NOTE
   If you do not have a certificate, you can skip this step as a self-signed certificate will be used when accessing the Sample Web Application.

   Run this command to create the secret:
   ```
   kubectl create secret tls sampleapp-general-tls --cert "${SA_CERTFILE}" \
   --key "${SA_KEYFILE}" -n "${SA_NAMESPACE}"
   ```
3. (**Optional step**) If you must authenticate to connect to your container registry, add this command line parameter to the Sample app helm command:
   ```
   --set global.registry.existingSecrets[0].name="ssp-registrypullsecret"
   ```

Deploying the Sample App Helm Chart

Deploy the Sample app helm chart using the following command. If you configured a custom SSL certificate or a registry pull secret in the previous steps, ensure you include those parameters.

For example, the Sample app helm command would look like the following after adding the optional commands:

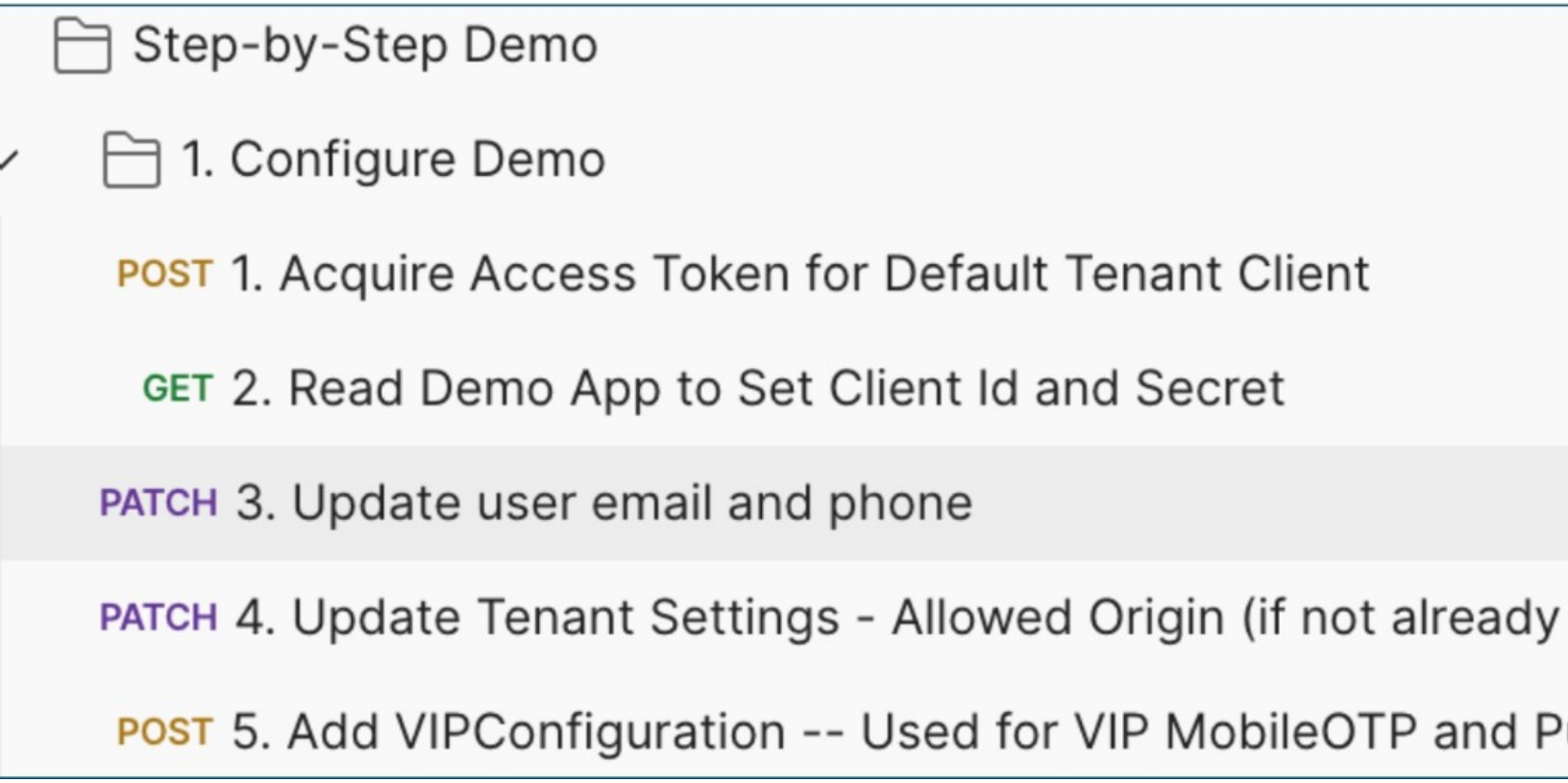
```
helm install "${SA_RELEASENAME}" -n "${SA_NAMESPACE}" ${HELM_REPO}/ssp-sample-app \
--set sspReleaseName=<value> \
--set ingress.tls.secretName=sampleapp-general-tls \
--set global.registry.existingSecrets[0].name="ssp-registrypullsecret"
```

NOTE

- The SMS OTP is preconfigured to use Twilio to send text messages for SMS OTP. Ensure to allow egress traffic to the Twilio service on the internet.
- If you would like to use the Email OTP use cases, ensure to configure an SMTP service provider in IDSP. See Managing Message Provider Configurations.
- The demo users created are part of the demo setup, have fictitious emails, and phone numbers. These emails, and phone numbers are not verified and therefore will not be used to send OTP messages. The real email or phone numbers can be configured using the Admin console, or the "/scim2/v1/Users" API, or using inline enrollment.
- To run the demo use cases, you must do the following:
  a. Set valid email and phone numbers to **nbruce** and **edunn.**
  b. Leave the user "**erussell"** without verified email and phone values, to display SMS/Email inline enrollment.

Using Postman to update the email Id's of the demo users

1. Update the SetEnvironment request script (in the *Tests* tab) to include the username, the demo default password, and the email and mobile phone numbers that you intend to set.
2. Aquire the Tenant Access Token.
3. Use "**Update user email and phone**" request to update the user's email and phone number, and verify them so that they can be used for OTP authentication.
   See the following image to reference the steps that you must perform in the Postman:



NOTE
Repeat these steps for multiple users.

Deploying Sample app in the IDSP namespace, and creating a new tenant for demo purposes

When the IDSP is deployed in non-demo mode, you can create a dedicated tenant enabling the demo setup, and by specifying the following parameter in the tenant payload:

```
demoSetup: true
```

For more information on Creating a Tenant, see Managing Tenants section.

- Creating the demo tenant, automatically sets the tenant setting, ***Allowed Origin*** (https://sampleapp-<sspfqdn>), creates the demo client app with the following details.
  - ***SAML EntityID (samlEntityId)***: https://sampleapp-<sspfqdn>
  - ***SAML Assertion Consumer Service URL (samlAcsUrl)***: https://sampleapp-<sspfqdn>/sample-rp/saml/acs
  - ***OpenID redirectURIs***: https://sampleapp-<sspfqdn>/sample-rp/oidc/callback
  - ***App Home URL***: https://sampleapp-<sspfqdn>/sample-rp/hom
- Save the demo client ID and secret returned by the create tenant API to be used during the Sample app deployment.

Once the demo tenant is created, deploy the sample app on the same Kubernetes namespace where the IDSP chart is deployed by using the following commands:

```
--set sspReleaseName=<value> \
--set demoTenant=<name of the demo tenant> \
--set ssp.clientId=<SA_CLIENTID> \
--set ssp.clientSecret=<SA_CLIENTSECRET}>
```

Where,

- <SA_CLIENTID> and <SA_CLIENTSECRET}> - these are the client Id and secret of the demo application.

Custom Deployment of the Sample App

Learn the steps to deploy the Sample application in a namespace that is different from the namespace where Symantec Identity Security Platform is deployed.

When a Sample app is deployed in the same namespace, it extracts some of its default values from the existing IDSP deployment. When deploying a Sample app on a different namespace or even a different cluster, you are required to specify additional parameters.

Purpose

The IDSP Sample Web application lets you set up a demo environment and test the multi-factor authentication flows offered by the solution. The sample web application is available and can be easily deployed on native cloud environment, or OpenShift.

Prerequisite

To be able to run the demo environment, ensure that you have deployed the following application:

- IDSP

  NOTE

  - If you are deploying Sample Web Application in a different cluster, ensure the prerequisites are met
  - The Sample Web Application supports RememberMe and Trusted Devices functionality. See RememberMe/ Trusted Devices.

This topic includes the following sections:

- Configuring Environment Variables
- Deploying Ingress for Sample Web Application
- Accessing the Sample Web Application
- Updating CORS Setting to Reflect Sample Web Application's FQDN
- Using the Sample Web Application

Configuring Environment Variables

Set up your system environment variables so that you can use them in the commands that are used in deploying the Sample Web Application. These commands can be executed in your native cloud providers, or OpenShift; you could also use a cloud shell.
Follow these steps:

1. Configure the following environment variables:

   NOTE
   Note that variables, RELEASENAME, and NAMESPACE refer to the IDSP helm release-name and namespace respectively, whereas the SA_RELEASENAME and SA_NAMESPACE refer to the Sample Web Application helm release-name and namespace.

   ```
   export SA_RELEASENAME="<sampleapp-release-name>"
   export SA_NAMESPACE="<sampleapp-namespace>
   export SA_FQDN="<sampleapp-fqdn>"
   ```

   Where,

- <sampleapp-release-name> - This is a name that you chose to identify your Sample Web Application deployment. In helm, a Release is an instance of a chart-deployment running in a Kubernetes cluster. The release name helps identify this particular deployment instance.
- <sampleapp-namespace> - This is the namespace for the Sample Web Application.
- <sampleapp-fqdn> - This is the Fully Qualified Domain Name that is used to access the Sample Web Application. If you are using your own certificate for the Sample Web Application, the certificate subject or alternate name should match this fqdn.

2. If you have a certificate from a trusted Certificate Authority (CA), configure the following environment variables to point to the key and cert files.

```
export SA_KEYFILE="<keyfile>"
export SA_CERTFILE="<certfile>"
```

Where,

- <keyfile> - The path to the certificate key file. (private)
- <certfile> - The path to the certificate file. (public)

NOTE

- The certificate ***Subject*** or ***Subject Alternative Name (SAN)*** should match the FQDN name of the Sample Web Application. To check that the SAN matches the FQDN name, use the following command:

```
openssl x509 -in ${SA_CERTFILE} -noout -text
```

Once the certificate details are displayed, look for the value of "CN=" in the ***Subject***, or ***Subject Alternate Name*** .

- If you do not have a certificate, you can skip this step, as a self-signed certificate will be used when accessing the Sample Web Application.

Deploying Ingress for Sample Web Application

Before deploying the Sample Web Application, set up the Ingress for the Sample Web Application.

NOTE

If you already have an Ingress setup in the cluster where you plan to deploy the Sample Web Application, then you can use the same Ingress.

For Ingress deployment steps, see Deploying Ingress.

Deploying the Sample Web Application

1. Create a namespace for the Sample Web Application.

```
kubectl create ns "${SA_NAMESPACE}"
```

2. If you have a certificate from a trusted CA to use with the Sample Web Application for SSL, then create the secret in the same namespace where the Sample Web Application will be deployed.

NOTE

If you do not have a certificate, you can skip this step as a self-signed certificate will be used when accessing the Sample Web Application.

Run this command to create the secret:

```
kubectl create secret tls sampleapp-general-tls --cert "${SA_CERTFILE}" \
--key "${SA_KEYFILE}" -n "${SA_NAMESPACE}"
```

3. To retrieve the IDSP "serviceUrl", use the following command:

NOTE

- Export the variables, ${NAMESPACE}, ${RELEASENAME}, and ${SSP_FQDN}. These are the IDSP (ssp) variables which you may have used during IDSP Helm chart deployment. See Setting up the Environment Variables .
- Execute this command on the IDSP Kubernetes cluster. If you are deploying the Sample app into a different cluster, you can run the command below on the ssp cluster and use these values when deploying Sample app.
- The command assumes that the IDSPtenant name is "default".
- "serviceUrl" (SSP_URL) is the url that the Sample Web Application uses in order to access the IDSP service. This url is composed of the service base url (https://${SSP_FQDN}/), followed by the tenant name with a trailing slash (https://${SSP_FQDN}/<tenatname>/). For example, https://ssp.iam.example.com/default/.

```
export SSP_FQDN=$(kubectl get ingress -n ${NAMESPACE} ${RELEASENAME}-ssp-ingress -o
 jsonpath={.spec.rules[].host})&&echo SSP_FQDN=${SSP_FQDN}
export SSP_URL=https://${SSP_FQDN}/default/&&echo SSP_URL=${SSP_URL}
```

4. Retrieve the client id and secret of the DemoClient App by navigating to the applications section in the Admin Console, identifying the app, and copying these values and setting them as environment valiables, SA_CLIENTID and SA_CLIENTSECRET respectively.
5. ***Registering a Pull Secret:***
   If you are required to use registry credentials to pull images from your container registry:
   – By default, the Sample Web Application deployment pulls images from the Broadcom container registry which requires authentication. The following command creates a docker-registry secret named, "ssp-reg-pullsecret" to store the registry credentials that would be used by Kubernetes to pull the images:

```
kubectl create secret docker-registry ssp-registrypullsecret -n ${SA_NAMESPACE} \
--docker-server="https://securityservices.packages.broadcom.com/" \
--docker-username="<username>" \
--docker-password="<password>" \
--docker-email="<your-email-address>"
```

NOTE

- Replace <username> and <password> which are the credentials that are required to log in to the Docker registry where the IDSP images are stored. The credentials for the Broadcom Docker registry are provided by Broadcom.
- The Sample App chart images are stored in the same registry as the IDSP (ssp) images. As a result, the same credentials are used to retrieve the images. However, because these two charts are deployed in different namespaces, the secret containing the credentials must be created in both IDSP (ssp) and Sample App namespaces.
- If you decide to use your own private registry and your registry requires authentication as well, you can use a similar command using your own docker server URL and credentials.

6. Get the latest Sample App helm chart updates by running this command:

```
helm repo add ssp_helm_charts https://ssp_helm_charts.storage.googleapis.com
helm repo update
```

7. Deploy the Sample Web Application.

```
helm install "${SA_RELEASENAME}" -n "${SA_NAMESPACE}" ${HELM_REPO}/ssp-sample-app \
--set ssp.serviceUrl="${SSP_URL}" \
--set ssp.clientId="${SA_CLIENTID}" \
--set ssp.clientSecret="${SA_CLIENTSECRET}" \
--set ingress.host="${SA_FQDN}" \
--set ingress.tls.host="${SA_FQDN}" \
--set ingress.tls.secretName=sampleapp-general-tls \
--set global.registry.existingSecrets[0].name="ssp-registrypullsecret"
```

NOTE

- The following command line parameters are optional:

```
--set samplerp.clientId="449b325c-96ee-4994-b0c2-827edf1b088c" \
--set samplerp.clientSecret="dc734a17-9ed6-4b90-a702-6b8853f78626" \
--set ssp.serviceIp="kubectl get svc -n ingress |awk 'NR==2 {print $4}'"
```

  - If you do not specify the "samplerp" clientId and clientSecret, the Sample app chart defaults to use the same clientId and secret that was provided for the sampleapp.
  - Specify the IDSP's service IP (ssp.serviceIp) when the IDSP FQDN is not resolvable from within the cluster. For instance, when the IDSP FQDN is only resolvable by the client machine using a local "/etc/hosts name" resolution.
- If you do not specify the ${SA_CLIENTID} and ${SA_CLIENTSECRET} here while installing the Sample app, then you must set it after the installation is complete, by clicking the Settings gear at the top left of the Sample Web Application's login page.

NOTE

For deploying Sample app in the OpenShift platform, carry out these additional steps:

- Set the following environment variables.

```
read -d '/' -r SA_UID < <(kubectl get ns ${SA_NAMESPACE} -o
 jsonpath='{.metadata.annotations.openshift\.io/sa\.scc\.uid-range}')
read -d '/' -r SA_GID < <(kubectl get ns ${SA_NAMESPACE} -o
 jsonpath='{.metadata.annotations.openshift\.io/sa\.scc\.supplemental-groups}')
export SA_FSGROUP=${SA_GID}
```

- Pass the following additional parameters in the Sample Web Application helm install command.

```
--set global.securityContext.runAsUser=${SA_UID} \
--set global.securityContext.runAsGroup=${SA_GID} \
--set global.securityContext.fsGroup=${SA_FSGROUP}
```

- To use OpenShift Routes in the Sample App, use the following parameter when deploying the sample app chart:

```
--set ingress.type=openshiftroute --set ingress.ingressClassName=openshift-default
```

8. Post deployment, Helm provides details regarding the deployment. If you wish to view this deployment information, use the following command:

```
helm get notes -n <namespace>  <sampleapp release name>
```

Accessing the Sample Web Application

Accessing the Sample Web Application is done through the Kubernetes cluster Ingress which routes requests to the Sample Web Application service based on the request's FQDN. This FQDN should match the Sample Web Application's FQDN that was specified during the deployment of the App.

To make the Sample Web Application accessible through a web browser, add the FQDN of the Sample Web Application and its IP Address to your Cloud DNS or your local DNS (In case, you are unable to update your DNS, you can update the "/etc/hosts" file on the machine where you are run your browser to allow local name resolution). The Sample Web Application FQDN should be resolved to the Ingress IP Address that can be retrieved by performing the following step:

1. To get the Ingress IP address, run the following command:

```
kubectl get svc -n ingress ingress-nginx-controller
```

   Where, ingress-*nginx-controller* is the name of the ingress controller Kubernetes service. The ingress IP address is listed under the "EXTERNAL_IP" column. When running this command on a cloud provider cluster, the "EXTERNAL_IP" may take a few minutes to be displayed since the service load balancer must be provisioned.

After you complete the deployment and update the FQDN and IP Address in your cloud DNS or your local DNS "/etc/hosts" file, the Sample Web Application can be accessed.

There are two URLS available for you to access the Sample App.

- ***Sample App URL***: This URL can be used to test the demo use cases in a programmatic way.
  `https://<sampleapp-fqdn>/sample-app/`
- ***Sample RP URL***: This URL can be used by developers to test the demo capabilities by invoking authentication using the protocol flows, SAML, and OIDC.
  `https://<sampleapp-fqdn>/sample-rp/home`

NOTE
The Sample Web Application is accessible at this point but requires performing the following steps to operate correctly.

Updating CORS Setting to Reflect Sample Web Application's FQDN

Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served. The list of "allowedOrigin" identifies the permitted origins to call IDSP APIs.

You are required to edit the CORS setting to add the Fully Qualified Domain Name (FQDN) of the Sample Web Application. This step is required if you are installing the Sample Web Application into an FQDN different from IDSP.

1. To set the CORS setting, update the "allowedOrigins" property (accepts a comma-separated list of urls) available in "tenant" settings with the required and valid http/https URLs.
2. Use the following "PATCH" call to update the payload with the Sample Web Application's FQDN URL.

   NOTE

   - Before running the following command, ensure that the variables, ${SSP_FQDN} in Configuring the Environment Variables and ${SSP_AT} as mentioned in the Obtaining Access Token for System Admin Tasks section.
   - Replace the "https://sampleapp.example:4200" with the actual URL as specified in the variable, ${SA_FQDN} provided to the *ingress.host* parameter during Sample App deployment.
   - The following cURL command uses *"--insecure"* parameter to ignore SSL verification. If your IDSP deployment uses trusted certificates, you can remove this parameter.

   Example of a cURL Request displaying the Tenant Settings PATCH payload

```
curl --insecure -L -X PATCH "https://"${SSP_FQDN}"/default/admin/v1/Configs" \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer ${SSP_AT}" \
-d '[
       {
           "name": "allowedOrigins",
           "value": "https://sampleapp.example.com:4200"
       }
 ]'
```

   Example of cURL Response

```
[
 {
     "name": "persistEnrolledFactorsinIDP",
     "value": "false",
     "origin": "global"
 },
 {
     "name": "azCodeExpirySecs",
     "value": "120",
     "origin": "global"
```

```
  },
  {
      "name": "isInlineOTPEnrollmentToBePersisted",
      "value": "true",
      "origin": "global"
  },
  {
      "name": "refreshTokenExpiryDays",
      "value": "7",
      "origin": "global"
  },
  {
      "name": "pushRegTimeoutSecs",
      "value": "300",
      "origin": "global"
  },
  {
  "name": "allowedOrigins",
      "value": "https://sampleapp.example.com:4200",
      "origin": "tenant"
  },
  {
      "name": "isInlineEnrollmentAllowed",
      "value": "true",
      "origin": "global"
  },
  {
      "name": "idTokenExpirySecs",
      "value": "21600",
      "origin": "global"
  },
  {
      "name": "pushAuthTimeoutSecs",
      "value": "300",
      "origin": "global"
  },
  {
      "name": "accessTokenExpirySecs",
      "value": "3600",
      "origin": "tenant"
  },
  {
      "name": "useOriginalIdentityTokenExpiry",
      "value": "false",
      "origin": "global"
  },
  {
      "name": "inlineEnrollmentFactorsRequired",
      "value": "PASSWORD",
      "origin": "global"
  }
]
```

Using the Sample Web Application

NOTE

The IDSP deployment automatically provisions a *DemoClient* app and a demo authentication policy. If you want to create your own application and authentication policy, perform the following steps. To continue with the pre-provisioned DemoClient app, **skip steps 1 and 2**. To know more about DemoClient, click here.

1. Create the Sample Web Application in the IDSP using the following cURL request:

   NOTE
   Before running the following command, ensure that the variables, ${SSP_FQDN} in Configuring the Environment Variables and ${SSP_AT} as mentioned in the Obtaining Access Token for System Admin Tasks section.

   Example of cURL Request

```
curl --insecure -L -X POST "https://"${SSP_FQDN}"/default/admin/v1/Apps" \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer ${SSP_AT}" \
-d '{
 "status": "active",
 "name": "app",
 "description": "Default Application for Security",
 "clientType": "CONFIDENTIAL",
 "allowedOpenIDScopes": [
     "openid",
     "email",
     "profile"
 ],
 "redirectURIs": [
     "https://www.example.com"
 ],
 "allowedGrantTypes": [
     "implicit",
     "authorization_code",
     "urn:ietf:params:oauth:grant-type:jwt-bearer",
     "client_credentials"
 ],
 "allowedOperations": [
     "introspect", "onbehalfof"
 ],
 "skipIssuerAudienceForIT": true,
 "userInfoEndpointResponseFormat": "PLAIN_JSON"
       }'
```

   The response payload fetches the Client ID and Client Secret values. Note these values, as these values will be used when you sign in to the Sample Web Application. For more information on the creation of the application, click here.

   Example of cURL Response

```
{
 "status": "active",
 "name": "app",
 "description": "Default Application for Security",
 "clientType": "CONFIDENTIAL",
 "allowedOpenIDScopes": [
     "openid",
     "profile",
     "email"
```

```
    ],
    "redirectURIs": [
        "https://www.example.com"
    ],
    "allowedGrantTypes": [
        "implicit",
        "client_credentials",
        "authorization_code",
        "urn:ietf:params:oauth:grant-type:jwt-bearer"
    ],
    "userInfoEndpointResponseFormat": "PLAIN_JSON",
    "skipIssuerAudienceForIT": true,
    "allowedOperations": [
        "introspect",
        "onbehalfof"
    ],
    "appId": "06c3cd12-20d2-4c89-8cc8-df429e3945e4",
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "006eddf2-ea8d-46b6-8c9b-2c355bf81aa2",
        "principalName": "defaulttenantclient"
    },
    "createdDateTime": "2020-06-23 04:28 AM UTC",
    "updatedDateTime": "2020-06-23 04:28 AM UTC",
    "secret": "c380f534-144d-44ac-b9e2-47a1dcfac5e1",
    "clientId": "5770ce5b-7252-4b9b-a7bd-da624c891fca"
}
```

2. Use the following cURL request to create a policy and associate it to the application created in the previous step:

NOTE

- This request creates a basic policy that contains only one rule. You should make use of the demo policy that is embedded in the Sample Web Application to be able to use all the use cases mentioned here. To see the detailed embedded policy, click Sample Policy Used for Demo
- Before running the following command, ensure that the variables, ${SSP_FQDN} in Configuring the Environment Variablesand ${SSP_AT} are set as mentioned in the Obtaining Access Token for System Admin Tasks section.

Example of cURL Request

```
cURL --insecure -L -X POST "https://"${SSP_FQDN}"/default/admin/v1/AuthPolicies" \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer ${SSP_AT}" \
-d '{
    "description": "Sample policy record",
    "status": "active",
    "policyName": "Sample policy",
    "apps": [
        {
            "id": "06c3cd12-20d2-4c89-8cc8-df429e3945e4"
        }
    ],
    "rules": [
        {
            "conditions": {
                "action": {
```

```
                "operator": "in",
                "value": [
                    "login"
                ]
            },
            "principal": {
                "group": {
                    "operator": "in",
                    "value": [
                        "HR"
                    ]
                }
            },
            "ipAddress": {
                "operator": "in",
                "value": [
                    "10.0.0.0/24"
                ]
            },
            "channel": {
                "operator": "in",
                "value": [
                    "mobile"
                ]
            },
            "riskThreshold": {
                "operator": "ge",
                "value": 60
            }
        },
        "result": {
            "effect": "allow",
            "obligations": [
                "PASSWORD",
                "OTP"
            ],
            "msg": "This policy is for login in web channel"
        }
    }
]
      }'
```

Example of cURL Response

```
{
 "policyId": "3b6dd3f3-307e-492c-aaac-2606a885d66f",
 "policyName": "Sample policy",
 "description": "Sample policy record",
 "status": "active",
 "matchAnyApp": false,
 "apps": [
     {
         "name": "app",
         "id": "06c3cd12-20d2-4c89-8cc8-df429e3945e4"
```

```
        }
    ],
    "rules": [
        {
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "HR"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "login"
                    ]
                },
                "ipAddress": {
                    "operator": "in",
                    "value": [
                        "10.0.0.0/24"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "mobile"
                    ]
                },
                "riskThreshold": {
                    "operator": "ge",
                    "value": 60
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD",
                    "OTP"
                ],
                "msg": "This policy is for login in web channel"
            }
        }
    ],
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "006eddf2-ea8d-46b6-8c9b-2c355bf81aa2",
        "principalName": "defaulttenantclient"
    },
    "createdDate": "2020-06-23 04:31:42"
```

```
}
```

3. Use the following URL to access the Sample Web Application in the browser:
   `https://<sampleapp-fqdn>/sample-app`

   NOTE
   If you have not installed the SSP server with an SSL certificate, the Sample Web Application has a certificate error accessing the SSP service. Before proceeding with the Sample Web Application, use the browser to access the SSP service' tenant URL ${SSP_URL} and accept the certificate exception. This lets the browser remember this exception.

After you have successfully completed all the steps that are mentioned here, you can test the web use cases that are described here.

Steps to Modify Client Secret and Client Id

To change the Client Secret and Id, follow these steps:

1. Use the following URL to access the Sample Web Application in the browser:
   `https://<sampleapp-fqdn>/sample-app`
2. Click the Settings gear and perform the following steps:
   a. Enter your Client Id, Client Secret, and the base URL.

      NOTE

      - The Client Id and the Client Secret details can be fetched from step 1. For more information, see Managing Applications.
      - The base URL is the default IDSP URL.

   b. (Optional step) Enter the client IP Address of the Sample Web Application. Use this option to test risk roles. For example- Testing negative country scenarios.
   c. Clicking the Generate button is optional as the application automatically generates the access token upon page load. You can now use the generated access token to perform the authentication flows.

Reference Links:

- Upgrading IDSP Sample App.

Deploying Demo Mobile Application (Android and iOS)

Learn how to set up the demo Mobile client app in your work environment and use it to receive notifications and perform authentication in the solution.

This topic includes the following sections:

- Hedmoral Mobile Android Application
- Hedmoral Mobile iOS Application

NOTE

- Passkey authentication is unavailable in Hedmoral application (iOS and Android).
- If you have already registered a FIDO credential over a web channel, and now you are attempting to register another FIDO credential through the Hedmoral app, in such a case, the authenticator returns a "credential not found" error. Therefore, before attempting to register a new FIDO credential on the Hedmoral app, you must first erase all currently registered FIDO credentials from Symantec Identity Security Platform. However, if you already registered a FIDO credential through the Hedmoral app and are attempting to add a new FIDO credential through the web app, it does not return any error and the flow is handled successfully.

Hedmoral Mobile Android Application

You can use the Android Mobile application (also known as Hedmoral Technologies app) to perform multi-factor authentication flows in IDSP. The supported authentication flows are Password, FIDO security key and biometric, Mobile OTP (includes VIP Access and Hedmoral), Server OTP (includes Email/SMS) and Push. The application also supports risk-based scenarios such as step-up (reauthentication), RememberMe, Trusted Devices, Public Client, and the Self-Service feature. To know more about these features, see the links that are provided in the reference section at the end of the page.

Prerequisites

To be able to run the Hedmoral Mobile Android App, ensure that you have met the following prerequisites:

- Obtained an Android device with Android 7.0 (Marshmallow) and above installed on it
- Deployed IDSP in your specific backend server (local or cloud environment)

Obtain Client Id and Secret

1. Retrieve the client id and secret for the out of the box, DemoClient (search for Demo Client) by running this command:

   ```
   kubectl get secret ${RELEASENAME}-ssp-secret-democlient -n ${NAMESPACE} -o jsonpath="{.data.clientId}" |
    base64 --decode; echo
   kubectl get secret ${RELEASENAME}-ssp-secret-democlient -n ${NAMESPACE} -o jsonpath="{.data.clientSecret}"
    | base64 --decode; echo
   ```

   The Client Id and Secret values that you have obtained should be entered in the Mobile Configurations Screen after you complete your installation. However, if you are using your mobile application as a ***Public Client***, you can ignore generating the secret. To know more about the public client, see Client Types.

Mobile Application Download Location

You can download the latest Mobile application from the Support Portal > My Downloads.

How to Install Mobile Application Using the APK File

Follow these steps to install the Mobile app:

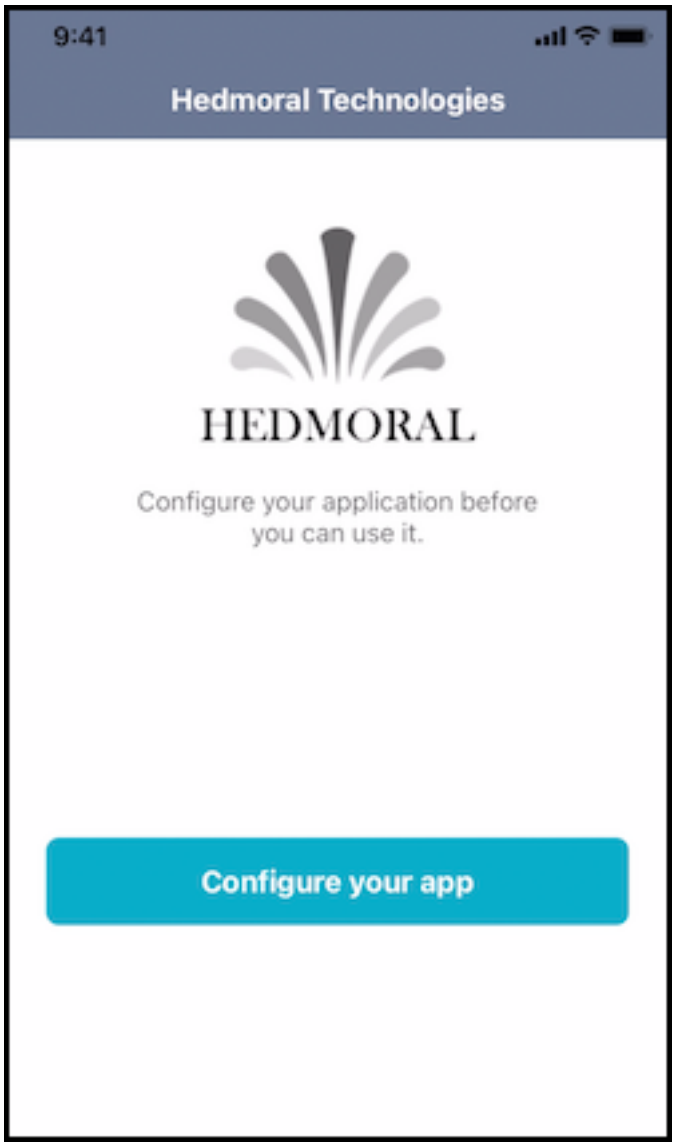
1. Navigate to the Validation site.
2. Copy the Android Package Kit, "APK" file from the "Clients" folder to your mobile device.
3. On your device, go to Menu > Settings > Security and check the **"Allow installation of apps from unknown sources"** option.
4. Tap the "APK" file and then click "INSTALL". The app begins installing on your device.
5. Look for the "Hedmoral Technologies" app in the Menu.

How to update the Hedmoral App with Configuration Details

After successfully installing the Hedmoral Mobile application on your phone, follow these steps:

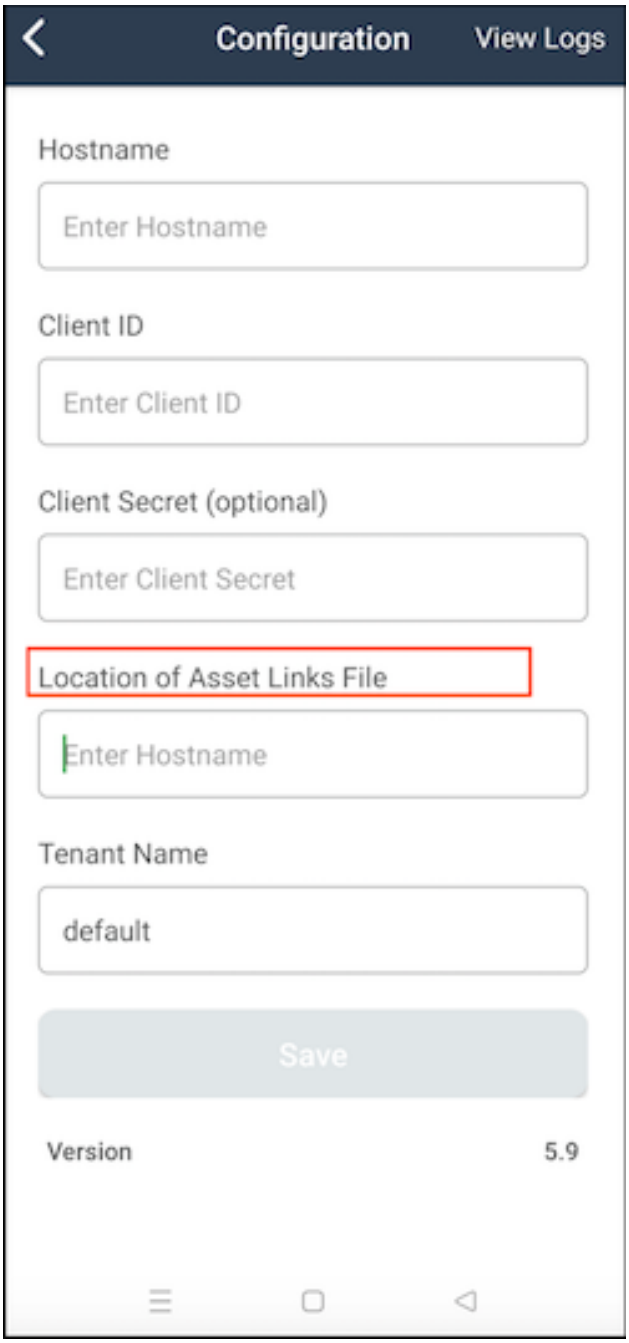
1. Open the application.
   The "Configure your app" screen is displayed.

Figure 10: Hedmoral Mobile App-Configure your app screen



2. Click the "Configure your app" field. The Configuration screen is displayed.
3. In the Configuration screen as shown here, enter the following details:

Figure 11: Hedmoral Mobile App-Configuration screen



a. Hostname: This is the IDSP server hostname.

NOTE

The supported hostname formats are as follows:

- ***Format 1: https://<Proxy Server URL>***
  Example: https://myProxy/Auth
  Where the host and tenant names are resolved by the Proxy server.
- ***Format 2: https://<hostname>***
  Example: https://myhost.com

When tenant is not part of the hostname, a tenant must be entered in the **Tenant Name** field.

- ***Format 3: <hostname>***
  Example: myhost.com
  In this case, the URI is optional. Since the tenant is not part of the hostname, a tenant must be entered in the **Tenant Name** field.

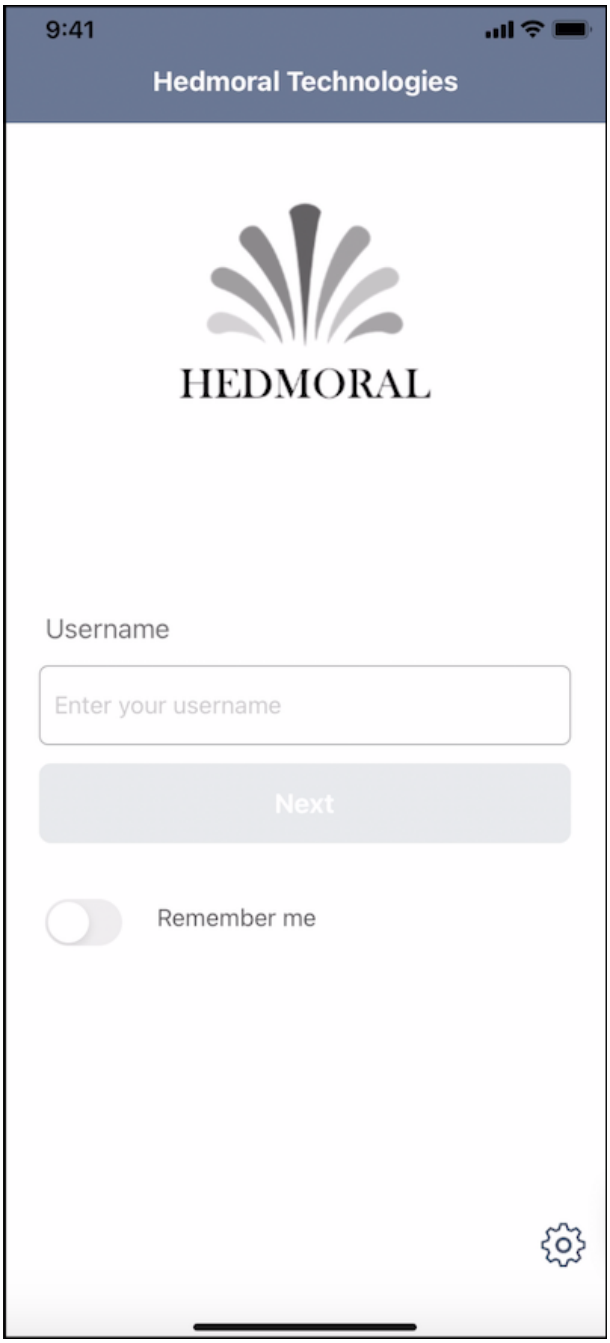
b. Client ID: This is obtained in the prerequisite section.
c. Client Secret: This field is optional if your mobile application is using the public client.
d. Location of Asset Links File: Provide the location where you have deployed the assetlinkjson.

   NOTE
   This field is only displayed for Hedmoral Android application.

e. Tenant Name: This is populated automatically as default. However, if you are using another tenant name, then update the same in this field.
f. Click Save. Upon saving, the Username screen is displayed.

4. Now, enter your credentials and log in to the application. <replace the username screen here>

Figure 12: Hedmoral Mobile App-Username screen



5. (Optional step) To check and update the configuration details, you can click the settings icon on the login screen.
6. (Optional step) To see the logs, click the **View Logs** in the configuration screen. The log information is helpful in troubleshooting issues. You can choose to share the logs in several ways, such as Gmail, WhatsApp, Mail, Message, etc.

Hedmoral Mobile iOS Application

As a user you can use the iOS Mobile application to execute, multi-factor authentication flows enabled by the solution. The supported authentication flows are Password, Platform Authenticators - (FIDO Face and Touch ID with a PIN as a fallback option), Mobile OTP (includes VIP Access and Hedmoral), Server OTP (includes Email/SMS) and Push. The application also supports risk-based scenarios such as step-up (reauthentication), RememberMe, Trusted Devices, Public

Client, and the Self-Service feature. To know more about these features, see the links that are provided in the reference section at the end of the page.

Prerequisites

To be able to run the Hedmoral Mobile iOS App for multi-factor authentication, ensure that you have met the following prerequisites:

- You have been invited to participate in the Apple TestFlight program and have downloaded the TestFight app from the Apple store to your phone. This allows you to download the Hedmoral Mobile app for iOS.
- Obtained an iOS device with iOS operating system 13 and above installed on it
- Deployed IDSP in your specific backend server (local or cloud environment)

Obtain Client Id and Secret

Follow the steps provided in the Android section.

How to update the iOS Hedmoral App with Configuration Details

Follow the steps provided in the Android section.

Now you can start testing the Mobile use cases described here using the IDSP app.

Reference Links to:

- Mobile Service endpoints
- RememberMe/Trusted Devices
- Reauthentication
- Push
- Public Clients
- Risk Evaluation
- Self-Service

Upgrading Symantec Identity Security Platform Sample Web App

Learn the steps to upgrade Symantec Identity Security Platform Sample Web application.

Follow these steps to upgrade the sample chart:

The version of IDSP has to be equal or greater than the version of the Sample App. Therefore, make sure you upgrade the IDSP deployment before upgrading the Sample App.

1. Set the Release Name and Namespace of the IDSP Sample App service that was used during the deployment as environment variables:

   ```
   export SA_RELEASENAME="<sampleapp-release-name>"
   export SA_NAMESPACE="<sampleapp-namespace>"
   ```

   where,
   - <sampleapp-release-name> -The ***Sample App Release Name*** that was used during the solution's deployment.
   - <sampleapp-namespace> - The ***Sample App Namespace*** used during the solution's deployment. See Deploying Sample Web App for more details.

     NOTE
     In case you do not remember the values for Sample App Release name and Namespace that were used during the deployment, use the following command to get those values:
     helm ls --all-namespaces
2. To get the versions of the Sample App charts that are currently deployed, run this command:

   ```
   helm ls -n ${SA_NAMESPACE}
   ```

```
# helm ls -n samplapp
NAME            NAMESPACE   REVISION   UPDATED                                   STATUS     CHART                    APP VERSION
ssp-sample      samplapp    1          2021-11-15 04:57:29.273625026 -0500 EST deployed   ssp-sample-app-1.0.2430 1.0.2430
```

3. Ensure that the helm is pointing to the correct IDSP helm repository where the charts can be found:

   ```
   helm repo add ssp_helm_charts https://ssp_helm_charts.storage.googleapis.com
   ```

   NOTE
   It is safe to run the above command more than once even if the repo was already added before.

4. To get the latest information about the IDSP charts available in the chart repositories, use this command:

   ```
   helm repo update
   ```

5. To get the list of all available sample app helm chart versions in the ssp_helm_charts repo, run this command:

   ```
   helm search repo ssp_helm_charts/ssp-sample-app --versions
   ```

```
# helm search repo ssp_helm_charts --versions | grep "sample-app"
ssp_helm_charts/ssp-sample-app          1.0.2430        1.0.2430        2021.Nov.01 - Helm chart to deploy a sample app...
ssp_helm_charts/ssp-sample-app          1.0.2287        1.0.2287        2021.Jul.01 - Helm chart to deploy a sample app...
ssp_helm_charts/ssp-sample-app          1.0.2002        1.0.2002        2021.Apr.01 - Helm chart to deploy a sample app...
ssp_helm_charts/ssp-sample-app          1.0.1925        1.0.1925        2021.Feb.01 - Helm chart to deploy a sample app...
```

6. To get the user-supplied deployment values that were used during the deployment of the IDSP sample chart and save it in the *"sample-override.yaml" file, run this command:*

   ```
   helm get values ${SA_RELEASENAME} -n ${SA_NAMESPACE} >sampleapp-override.yaml
   ```

7. To upgrade to the latest version of IDSP Sample app chart, run this command:

   ```
   helm upgrade ${SA_RELEASENAME} ssp_helm_charts/ssp-sample-app -n ${SA_NAMESPACE}  -f sampleapp-
   override.yaml
   ```

```
$ helm upgrade ${SA_RELEASENAME} ssp_helm_charts/ssp-sample-app -n ${SA_NAMESPACE} -f sampleapp-override.yaml
Release "sampleapp" has been upgraded. Happy Helming!
NAME: sampleapp
LAST DEPLOYED: Fri Oct 16 09:42:23 2020
NAMESPACE: dev-sample
STATUS: deployed
REVISION: 4
TEST SUITE: None
NOTES:
Thank you for installing ssp-sample-app.
  Access sample app here: https://sampleapp.aws.ssp.security.broadcom.com/sample-app
```

   To upgrade to a specific version of IDSPchart, run this command:

   ```
   helm upgrade ${SA_RELEASENAME} ssp_helm_charts/ssp-sample-app -n ${SA_NAMESPACE}  -f sampleapp-
   override.yaml --version <CHART VERSION>
   ```

   <CHART VERSION> -This is the value that appears in the output of the helm search (helm search repo ssp_helm_charts/ssp-sample-app --versions) command.

8. To verify that the IDSP Sample App chart was upgraded to the desired version, run this command:

   ```
   helm ls -n ${SA_NAMESPACE}
   ```

Developer Resources

Learn how to make your first API call using Symantec Identity Security Platform, and integrate it with sample mobile/web/custom web/custom service providers, custom authenticators and about SDKs that the solution provides.

Click these links to know more about each topic:

- About API Security Using OAuth2
- Making the First API call
- Integrating Sample Web App with IDSP
- Integrating Sample Mobile App with IDSP
- Implementing Registration of Multiple FIDO Credentials
- Integrating Custom Web Applications with IDSP
- Integrating with Custom Service Provider Interface (SPI)
- Integrating with Custom Authentication Factor
- IDSPas CAEP Transmitter/Receiver
- Auditing in Symantec Identity Security Platform
- Custom Metrics Exposed by Symantec Identity Security Platform
- IDSP SDKs

API Security Using OAuth2

Learn how to secure the Symantec Identity Security Platform REST APIs based on OAuth2 and OpenID industry standards, relying on JWT-based Access Token to control access to REST APIs. Learn about tokens, supported client types and grant types, and endpoints exposed by IDSP.

This lets developers authenticate users using Open ID Connect or other means to securely assert the identity, and then acquire access tokens to authorize access to the APIs.

- As the OAuth Resource Server, IDSP supports a rich set of administrative APIs to manage its resources and configuration.
- As the OAuth Authorization Server, IDSP supports client-only-based token issuance, as well as user-based token issuance to support identity propagation requirements.

> **NOTE**
> The use of OpenID Connect flows can be used independently of REST API access to handle identity assertion requirements.

This topic includes the following sections:

- General Notes for building or integrating client apps
- Prerequisites

General Notes for building or integrating client apps

As a developer, you are building applications using IDSP APIs or integrating your client apps using OpenID Connect standard. To perform these tasks, you should remember to do the following:

- Enabling CORS:
  - If your application's Fully Qualified Domain Name (FQDN)/Host is different from IDSP's, you must add your client app's FQDN by updating the "allowedOrigins" setting at the tenant level to enable the CORS standard to function properly.

    - To configure your client app's AllowedOrigins, use the PATCH "/admin/v1/Configs" API. See Settings page to know about allowedOrigins setting and see Applications to check the steps to update allowedOrigins setting.
- If your client app uses OpenID Connect to integrate with IDSP, make sure the app resource has the following configured:
  – redirectURI, allowedOpenIDScopes, allowedGrantTypes, and allowedOperations. These topics have been discussed in the subsequent sections in this page.
- If your client app uses the IDSP REST API to orchestrate authentication flow, the application has to be authorized with required privileges. Therefore, ensure that the client app is granted required privileges by adding it to the "AuthenticationClient" authorization policy. This policy gives privilege to invoke the authentication APIs. See Authentication and Authorization REST API (Scopes/Privileges) for more details.

Prerequisite

Install cURL to execute the samples provided in the sections below.

> **NOTE**
> A recommended way to verify the service URL is by running the service discovery curl command.

Reference Links to:

- Rest APIs
- Tokens
- Client Types
- Grant Types
- Authorization Server Endpoints
- JSON Web Encryption (JWE)
- Client Assertion
- Secure Communication for Public Clients USing PKCE
- Detailed Examples of Authorization Endpoints
- OAuth Swagger documentation
- OAuth Error Codes

Introduction to Symantec Identity Security Platform Rest APIs and Scopes

Learn about Symantec Identity Security Platform Rest APIs and Privileges.

REST APIs allow applications to interact with IDSP resources using standard HTTP methods. Access to these APIs is controlled by OAuth scopes, which define the specific permissions required to perform operations.

This page includes the following sections:

- Rest API Endpoints
- Authentication and Authorization (Scopes/Privileges)
- Scope Management and Usage
- Scope Notation
- API Response Code
- Rest APIs and Associated Privileges

REST API Endpoints

You access the IDSP REST APIs using a URL endpoint based on the following syntax:

```
https://${SSP_FQDN}/<tenant>/<function>/v1/<operation>
```

- `<host>` : The FQDN of the ingress endpoint exposed by the deployed cluster.
- `<tenant>` : Is set to "default".
- `<function>` : Is one of the deployed micro-services listed in the API table.
- `<operation>` is the one of the business operations listed in the API table

For example, the API endpoint to create an Application resource is -
`https://example.com/default/admin/v1/Apps`

To use the REST APIs, an application requests an Access Token with the desired privileges (OAuth scopes) and then invokes the REST API. The API requests supplies the Access Token in the Authorization header as `Bearer <token>`.

> **NOTE**
> **Requirements for OpenShift Environment**
>
> In the OpenShift environment, when invoking tenant-specific APIs, it is mandatory to provide the "X-TENANT-NAME" header in the API request, else the API request fails with "400 BAD REQUEST" status.
>
> **Key Points**:
>
> - The tenant identified by "X-TENANT-NAME" must exist in the system
> - SSL certificate must be issued for the SSP_FQDN host
> - All tenant-specific API clients must include the "X-TENANT-NAME" header in their requests
>
> For more information, see Deploying Ingress Controller on Openshift

Authentication and Authorization (Scopes/Privileges)

REST APIs are protected by privileges ("scopes") exposed by IDSP. Application clients specify scopes as required in order to invoke REST API when initiating flows to request an access token. Once the request has been validated, the authorization server returns an access token containing the requested and authorized scopes.

Scope Management and Usage

Scopes are a list of case-sensitive strings that are defined by the Authorization server to specify the access privileges requested by the client/user.

The following section describes how these scopes are managed and used:

- **If the requested scope is not entitled**: If the requested scope is not entitled to the client/user, the client/user will be issued an access token with the list of all relevant scopes.
- **No scope passed in request**: If the client/user does not pass any scope in the request, the issued access token contains "urn:iam:m.me" scope and all APIs which are mapped to this scope will be authorized.
- **Subset of scopes**: If the client/user wants to make use of only a subset of the scope that is assigned to them, they can specify the list of scope (separated by space) in the body of the request. If the "scope" body contains the value "urn:iam:myscopes", then the client/user is issued the access token containing all the scopes which are associated with them.
- **Best Practice**: As a best practice, you should only request access for what you absolutely need.

Scope Notation

- **urn:iam:g.* scope**: Represent system-level privileges required to execute APIs that manage the operational state of the system. These scopes are granted by the System Admin and Infra Admin Authorization Policies.
- **urn:iam:t.* scope**: Represent tenant-level privileges required to execute APIs that implement the business of the system. These scopes are granted by the Tenant Admin Authorization Policy.
- **urn:iam:m.me scope**: Represent tenant-level privileges required to execute APIs that implement self-service operations on an end user. All such operations use the identity from the access token's "sub" claim. There is no

specific Admin Authorization Policy for these scopes. Instead, all ID propagation access token flows automatically add the "urn:iam:m.me" scope to any access token issued when the token's "sub_type" is "user".

- **urn:iam:myscopes (special convenience scope)**: Unlike requesting specific privileges to ensure access tokens get generated using the "least privilege" model, requesting the special convenience scope `urn:iam:myscopes` instructs the authorization server to return all authorized privileges.
  Greater care should be taken with this type of token. For example, these tokens should have a limited lifespan.

API Response Code

REST APIs are available for controllers and resources. The error codes returned by the REST API are as follows:

- **Controllers -** Examples of controllers: `/authenticate`. For controllers, the supported HTTP response codes are:
  - Success - 200
  - Failure - 401, 404, 429, 500
- **Resources -** Examples of resources: `/Users`, `/Groups`, `/policies`. For resources, the supported HTTP response codes are:
  - Success:
    - GET, PUT - 200
    - POST - 201
    - DELETE - 204
  - Failure - 401, 404, 409, 429, 500

For specific API details, see Swagger UI API References.

Rest APIs and Associated Privileges

To grant access to REST APIs, administrators can authorize different types of entities:

- Clients (applications or services)
- User groups (collections of users with similar roles)
- Individual users (through identity propagation)

This is done by adding or removing these entities from authorization policies that correspond to specific Admin Roles.

Built-in Privilege Mapping:

The system comes with a predefined mapping that connects each Admin Role to the specific privileges needed to call particular REST APIs. This mapping is configured out-of-the-box and determines which API endpoints each role can access.

The following topics show REST API mappings with their function-based privileges:

- ACCESS API Privileges
- ADMIN API Privileges
- AUTH API Privileges
- CAEP API Privileges
- FACTOR API Privileges
- IARISK API Privileges
- IASERVICES API Privileges
- IDENTITY API Privileges
- IDSTORE API Privileges
- LDAP API Privileges
- ME API Privileges
- OAUTH2 API Privileges
- RISK API Privileges
- SCIM 2.0 API Privileges
- UI API Privileges

ACCESS APIs

Learn about Symantec Identity Security Platform API endpoints and their associated informations like privileges, supported HTTP methods, descriptions etc.

ACCESS Function

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Get User's Applications	/access/v1/Apps/{userLoginName}	T	GET	urn:iam:t.authenticate
Get My Applications	/access/v1/MeApps	M	GET	urn:iam:m.me
Get My App Roles	/access/v1/MeAppRoles	M	GET	urn:iam:m.me urn:iam:m.meclient
Get My App Privileges	/access/v1/MeAppPrivileges	M	GET	urn:iam:m.me urn:iam:m.meclient

ADMIN API Privileges

This section covers the **ADMIN REST API** endpoints and the specific authorization privileges required to access them.

The following table maps **ADMIN APIs** to their required scopes and supported HTTP methods:

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Get All Applications	/admin/v1/Apps	T	GET	urn:iam:t.apps
Create Application	/admin/v1/Apps	T	POST	urn:iam:t.apps
Get Application by ID	/admin/v1/Apps/{appId}	T	GET	urn:iam:t.apps
Update Application	/admin/v1/Apps/{appId}	T	PUT	urn:iam:t.apps
Delete Application	/admin/v1/Apps/{appId}	T	DELETE	urn:iam:t.apps

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Check Application Existence	/admin/v1/Apps/{appName}	T	HEAD	urn:iam:t.apps
Get Authentication Policies	/admin/v1/AuthPolicies	T	GET	urn:iam:t.authpolicies
Create Authentication Policy	/admin/v1/AuthPolicies	T	POST	urn:iam:t.authpolicies
Get Policy by ID	/admin/v1/AuthPolicies/{policyId}	T	GET	urn:iam:t.authpolicies
Update Policy	/admin/v1/AuthPolicies/{policyId}	T	PUT	urn:iam:t.authpolicies
Patch Policy	/admin/v1/AuthPolicies/{policyId}	T	PATCH	urn:iam:t.authpolicies
Delete Policy	/admin/v1/AuthPolicies/{policyId}	T	DELETE	urn:iam:t.authpolicies
Check Policy Existence	/admin/v1/AuthPolicies/{policyName}	T	HEAD	urn:iam:t.authpolicies
Evaluate Authentication Policy	/admin/v1/AuthPoliciesEvaluator	T	POST	urn:iam:t.authpolicies
Get App Authentication Policies	/admin/v1/AppsAuthPolicies	T	GET	urn:iam:t.apps
Get App Auth Policies	/admin/v1/AppsAuthPolicies/{appId}	T	GET	urn:iam:t.apps
Assign Auth Policy to App	/admin/v1/AppsAuthPolicies/{appId}	T	PATCH	urn:iam:t.apps
Remove Auth Policy from App	/admin/v1/AppsAuthPolicies/{appId}	T	DELETE	urn:iam:t.apps
Get Authorization Policies	/admin/v1/AuthZPolicies	T	GET	urn:iam:t.authzpolicies urn:iam:g.authzpolicywriter
Create Authorization Policy	/admin/v1/AuthZPolicies	T	POST	urn:iam:t.authzpolicies urn:iam:g.authzpolicywriter
Get AuthZ Policy by ID	/admin/v1/AuthZPolicies/{policyId}	T	GET	urn:iam:t.authzpolicies urn:iam:g.authzpolicywriter
Update AuthZ Policy	/admin/v1/AuthZPolicies/{policyId}	T	PUT	urn:iam:t.authzpolicies urn:iam:g.authzpolicywriter
Patch AuthZ Policy	/admin/v1/AuthZPolicies/{policyId}	T	PATCH	urn:iam:t.authzpolicies urn:iam:g.authzpolicywriter
Delete AuthZ Policy	/admin/v1/AuthZPolicies/{policyId}	T	DELETE	urn:iam:t.authzpolicies urn:iam:g.authzpolicywriter
Check AuthZ Policy Existence	/admin/v1/AuthZPolicies/{policyId}	T	HEAD	urn:iam:t.authzpolicies urn:iam:g.authzpolicywriter
Search Users	/admin/v1/Users	T	GET	urn:iam:t.accountadmin urn:iam:t.users urn:iam:t.userdata
Get User by ID	/admin/v1/Users/{userId}	T	GET	urn:iam:t.accountadmin urn:iam:t.users urn:iam:t.userdata
Delete User	/admin/v1/Users/{userId}	T	DELETE	urn:iam:t.accountadmin urn:iam:t.users urn:iam:t.userdata
Get User Lock Status	/admin/v1/UserLocker/{userId}	T	GET	urn:iam:t.users urn:iam:t.accountadmin
Set User Lock Status	/admin/v1/UserLocker/{userId}	T	POST	urn:iam:t.users urn:iam:t.accountadmin
Get User Disable Status	/admin/v1/UserDisabler/{userId}	T	GET	urn:iam:t.users urn:iam:t.accountadmin

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Set User Disable Status	/admin/v1/UserDisabler/{userId}	T	POST	urn:iam:t.users urn:iam:t.accountadmin
Update User Password	/admin/v1/PasswordUpdater	T	PUT	urn:iam:t.users
Get Admin Roles	/admin/v1/AdminRoles	T	POST	urn:iam:t.users urn:iam:t.groups urn:iam:t.apps urn:iam:t.adminroles
Get All Admin Roles	/admin/v1/AdminRolesDelegator	T	GET	urn:iam:t.users urn:iam:t.groups urn:iam:t.apps urn:iam:t.adminroles
Grant Admin Role	/admin/v1/AdminRolesDelegator	T	POST	urn:iam:t.users urn:iam:t.groups urn:iam:t.apps urn:iam:t.adminroles
Revoke Admin Role	/admin/v1/AdminRolesDelegator	T	DELETE	urn:iam:t.users urn:iam:t.groups urn:iam:t.apps urn:iam:t.adminroles
Get My Admin Roles	/admin/v1/MyAdminRoles	M	GET	urn:iam:m.me urn:iam:m.meclient
Get App Roles	/admin/v1/AppRoles	T	POST	urn:iam:t.users urn:iam:t.groups urn:iam:t.apps urn:iam:t.adminroles
Get App Privileges	/admin/v1/AppPrivileges	T	POST	urn:iam:t.users urn:iam:t.groups urn:iam:t.apps urn:iam:t.adminroles
Grant App Role	/admin/v1/AppRolesDelegator	T	POST	urn:iam:t.users urn:iam:t.groups urn:iam:t.apps urn:iam:t.adminroles
Revoke App Role	/admin/v1/AppRolesDelegator	T	DELETE	urn:iam:t.users urn:iam:t.groups urn:iam:t.apps urn:iam:t.adminroles
Get All Certificates	/admin/v1/Certs	T	GET	urn:iam:t.certs
Upload Certificate	/admin/v1/Certs	T	POST	urn:iam:t.certs
Get Certificate by ID	/admin/v1/Certs/{certId}	T	GET	urn:iam:t.certs
Update Certificate	/admin/v1/Certs/{certId}	T	PUT	urn:iam:t.certs
Delete Certificate	/admin/v1/Certs/{certId}	T	DELETE	urn:iam:t.certs
Get All Secrets	/admin/v1/Secrets	T	GET	urn:iam:t.secrets
Add Secret	/admin/v1/Secrets	T	POST	urn:iam:t.secrets
Get Secret by ID	/admin/v1/Secrets/{secretId}	T	GET	urn:iam:t.secrets

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Update Secret	/admin/v1/Secrets/{secretId}	T	PUT	urn:iam:t.secrets
Delete Secret	/admin/v1/Secrets/{secretId}	T	DELETE	urn:iam:t.secrets
Get Secret in Base64	/admin/v1/SecretsGetter/{secretId}	T	GET	urn:iam:t.secretsgetter
Get All Keys	/admin/v1/Keys	T	GET	urn:iam:t.secrets
Update Key	/admin/v1/Keys/{keyId}	T	PUT	urn:iam:t.secrets
Enable/Disable Key	/admin/v1/Keys/{keyId}	T	PATCH	urn:iam:t.secrets
Update Tenant Settings	/admin/v1/Configs	T	PATCH	urn:iam:t.configs
Get Tenant Settings	/admin/v1/Configs	T	GET	urn:iam:t.configs
Update Global Settings	/admin/v1/Settings	G	PATCH	urn:iam:g.settings
Get Global Settings	/admin/v1/Settings	G	GET	urn:iam:g.settings
Get All Tenants	/admin/v1/Tenants	G	GET	urn:iam:g.tenant
Create Tenant	/admin/v1/Tenants	G	POST	urn:iam:g.tenant
Update Tenant	/admin/v1/Tenants/{tenantId}	G	PATCH	urn:iam:g.tenant
Delete Tenant	/admin/v1/Tenants/{tenantId}	G	DELETE	urn:iam:g.tenant
Get Tenant Lock Status	/admin/v1/TenantLocker	T	GET	urn:iam:t.tenantlocker
Lock/Unlock Tenant	/admin/v1/TenantLocker	T	PUT	urn:iam:t.tenantlocker
Get Signing Certificate	/admin/v1/SigningCert	T	GET	urn:iam:t.configs
Update Signing Key	/admin/v1/SigningKey	T	PUT	urn:iam:t.signingkey
Update Next Signing Key	/admin/v1/SigningKey	T	PATCH	urn:iam:t.signingkey
Revoke Session	/admin/v1/Sessions/{sessionId}	T	DELETE	urn:iam:t.session
Revoke Refresh Tokens	/admin/v1/RefreshTokens/{sessionId}	T	DELETE	urn:iam:t.session
Revoke Client Refresh Tokens	/admin/v1/ClientsRefreshTokens/{clientId}	T	DELETE	urn:iam:t.session
Revoke User Sessions	/admin/v1/UsersSessions/{userId}	T	DELETE	urn:iam:t.accountadmin urn:iam:t.users urn:iam:t.userdata
Revoke User Refresh Tokens	/admin/v1/UsersRefreshTokens/{userId}	T	DELETE	urn:iam:t.accountadmin urn:iam:t.users urn:iam:t.userdata
Revoke User Sessions and Tokens	/admin/v1/UsersSessionsAndRefreshTokens/{userId}	T	DELETE	urn:iam:t.accountadmin urn:iam:t.users urn:iam:t.userdata
Revoke Session	/admin/v1/revoke	T	PATCH	urn:iam:t.session
Get Messages	/admin/v1/Messages	T	GET	urn:iam:t.configs
Update Messages	/admin/v1/Messages	T	POST	urn:iam:t.configs
Get Message by ID	/admin/v1/Messages/{messageId}	T	GET	urn:iam:t.configs
Update Message by ID	/admin/v1/Messages/{messageId}	T	PUT	urn:iam:t.configs
Delete Message by ID	/admin/v1/Messages/{messageId}	T	DELETE	urn:iam:t.configs
Get Message Locale	/admin/v1/Messages/locale	T	GET	urn:iam:t.configs
Get Message Providers	/admin/v1/MessageProviders	T	GET	urn:iam:g.settings urn:iam:t.configs

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Update Message Provider	/admin/v1/MessageProviders/{providerId}	T	PUT	urn:iam:g.settings urn:iam:t.configs
Flush Provider Cache	/admin/v1/FlushProviderCache	T	POST	urn:iam:g.settings urn:iam:t.configs
Get Custom Providers	/admin/v1/CustomProviders	T	GET	urn:iam:g.settings urn:iam:t.configs
Create Custom Provider	/admin/v1/CustomProviders	T	POST	urn:iam:g.settings urn:iam:t.configs
Get Custom Provider by ID	/admin/v1/CustomProviders/{providerId}	T	GET	urn:iam:g.settings urn:iam:t.configs
Update Custom Provider	/admin/v1/CustomProviders/{providerId}	T	PUT	urn:iam:g.settings urn:iam:t.configs
Delete Custom Provider	/admin/v1/CustomProviders/{providerId}	T	DELETE	urn:iam:g.settings urn:iam:t.configs
Get Auth Policies Apps	/admin/v1/AuthPoliciesApps	T	GET	urn:iam:t.authpolicies urn:iam:t.idpdiscoverypolicies urn:iam:t.authzpolicies
Get Auth Policies App by ID	/admin/v1/AuthPoliciesApps/{policyId}	T	GET	urn:iam:t.authpolicies urn:iam:t.idpdiscoverypolicies urn:iam:t.authzpolicies
Apps Provider Helper	/admin/v1/AppsProviderHelper	T	POST	urn:iam:t.apps urn:iam:t.authpolicies urn:iam:t.idpdiscoverypolicies urn:iam:t.authzpolicies urn:iam:t.adminroles
AMR Provider Helper	/admin/v1/AmrProviderHelper	T	POST	urn:iam:t.idpconfigs urn:iam:t.authpolicies urn:iam:t.configs
App Groups Provider Helper	/admin/v1/AppGroupsProviderHelper	T	POST	urn:iam:t.apps urn:iam:t.authpolicies urn:iam:t.idpdiscoverypolicies urn:iam:t.authzpolicies urn:iam:t.adminroles urn:iam:t.resourcegroups
Get Apps Auth Policies	/admin/v1/AppsAuthPolicies	T	GET	urn:iam:t.apps
Health Check - Tenant	/admin/v1/hc	G	GET	urn:iam:g.tenant
Health Check - Settings	/admin/v1/hc	G	GET	urn:iam:g.settings

AUTH API Privileges

This section covers the **AUTH REST API** endpoints and the specific authorization privileges required to access them.

The following table maps **AUTH REST APIs** to their required scopes and supported HTTP methods:

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Authentication Initiation	/auth/v1/authenticate	T	POST	urn:iam:t.authenticate
Authentication Simulator	/auth/v1/AuthenticateSimulator	T	POST	urn:iam:t.authenticate
User Authenticator	/auth/v1/UserAuthenticator	T	POST	urn:iam:t.authenticate
Multi-Factor Authentication	/auth/v1/SelectedFactor	T	POST	urn:iam:t.authenticate
UI Locale Content	/auth/v1/UILocaleContent	T	POST	urn:iam:t.authenticate
Tenant Branding Settings	/auth/v1/BrandingSettings/idp	T	GET	urn:iam:t.authenticate
Branding Settings	/auth/v1/BrandingSettings	T	GET	urn:iam:t.authenticate
Get My Trusted Devices	/auth/v1/MeTrustedDevices	M	GET	urn:iam:m.me
Get My Trusted Device	/auth/v1/MeTrustedDevices/{deviceId}	M	GET	urn:iam:m.me
Delete My Trusted Device	/auth/v1/MeTrustedDevices/{deviceId}	M	DELETE	urn:iam:m.me
Data Decryption	/auth/v1/DecryptData	T	POST	urn:iam:t.authenticate
Client Nonce Helper	/auth/v1/ClientNonceGenerator	T	POST	urn:iam:t.configs
Server Nonce Helper	/auth/v1/NonceGenerator	T	POST	urn:iam:t.authenticate
Get My Sessions	/auth/v1/MeSessions	M	GET	urn:iam:m.me
Revoke My Sessions	/auth/v1/MeSessions	M	DELETE	urn:iam:m.me
Get My Session by ID	/auth/v1/MeSessions/{sessionId}	M	GET	urn:iam:m.me
Revoke My Session	/auth/v1/MeSessions/{sessionId}	M	DELETE	urn:iam:m.me
Get My Refresh Tokens	/auth/v1/MeRefreshTokens	M	GET	urn:iam:m.me
Revoke My Refresh Tokens	/auth/v1/MeRefreshTokens	M	DELETE	urn:iam:m.me
Get My Refresh Token	/auth/v1/MeRefreshTokens/{sessionId}	M	GET	urn:iam:m.me
Revoke My Refresh Token	/auth/v1/MeRefreshTokens/{sessionId}	M	DELETE	urn:iam:m.me
Revoke My Sessions and Tokens	/auth/v1/MeSessionsAndRefreshTokens	M	DELETE	urn:iam:m.me
My Refresh Tokens Helper	/auth/v1/MeRefreshTokensHelper	M	GET	urn:iam:m.me

CAEP API Privileges

This section covers the **CAEP REST API** endpoints and the specific authorization privileges required to access them.

The following table maps **CAEP REST APIs** to their required scopes and supported HTTP methods:

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
CAEP Security Events	/caep/v1/Events	T	POST	urn:iam:t.securityevents

FACTOR API Privileges

This section covers the **FACTOR REST API** endpoints and the specific authorization privileges required to access them.

The following table maps **FACTOR REST API** to their required scopes and supported HTTP methods:

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Password Authentication	/factor/v1/PasswordAuthenticator	T	POST	urn:iam:t.authenticate
Custom Authentication	/factor/v1/CustomAuthenticator	T	POST	urn:iam:t.authenticate
Attribute Based Factor	/factor/v1/Authenticator	T	POST	urn:iam:t.authenticate
OTP Registration	/factor/v1/OTPRegistrar	M	POST	urn:iam:m.me urn:iam:t.authenticate
OTP Generation	/factor/v1/OTPGenerator	M	POST	urn:iam:m.me urn:iam:t.authenticate
OTP Verification	/factor/v1/OTPVerifier	M	POST	urn:iam:m.me urn:iam:t.authenticate
Update My Password	/factor/v1/MePasswordUpdater	M	PUT	urn:iam:m.me urn:iam:t.authenticate
Mobile OTP Registration	/factor/v1/MobileOTPRegistrar	M	POST	urn:iam:m.me urn:iam:t.authenticate
VIP Credential Registration	/factor/v1/VIPCredentialRegistrar	M	POST	urn:iam:m.me urn:iam:t.authenticate
Mobile OTP Verification	/factor/v1/MobileOTPVerifier	M	POST	urn:iam:m.me urn:iam:t.authenticate
OCRA Transaction Verification	/factor/v1/OCRATransactionVerifier	M	POST	urn:iam:m.me
Initiate Push Registration	/factor/v1/PushRegistrarInitiator	M	POST	urn:iam:m.me urn:iam:t.authenticate
Get Push Registration Status	/factor/v1/PushRegistrarStatus	M	GET	urn:iam:m.me urn:iam:t.authenticate
Register Device for Push Notification	/factor/v1/PushRegistrar	M	POST	urn:iam:m.me urn:iam:t.authenticate
Push Notification Generation	/factor/v1/PushGenerator	M	POST	urn:iam:m.me urn:iam:t.authenticate
Verify Push Notification	/factor/v1/PushVerifier	M	POST	urn:iam:m.me urn:iam:t.authenticate
Push Status Check	/factor/v1/PushStatus	M	POST	urn:iam:m.me urn:iam:t.authenticate
FIDO Authentication Challenge	/factor/v1/ FIDOAuthChallengeGenerator	M	POST	urn:iam:m.me urn:iam:t.authenticate
FIDO Authentication Verification	/factor/v1/FIDOAuthChallengeVerifier	M	POST	urn:iam:m.me urn:iam:t.authenticate
FIDO Registration Challenge	/factor/v1/ FIDORegChallengeGenerator	M	POST	urn:iam:m.me urn:iam:t.authenticate
FIDO Registration Verification	/factor/v1/FIDORegChallengeVerifier	M	POST	urn:iam:m.me urn:iam:t.authenticate
Get My Credentials	/factor/v1/MeCreds	M	GET	urn:iam:m.me

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Get My Credential	/factor/v1/MeCreds/{credId}	M	GET	urn:iam:m.me
Update My Credential	/factor/v1/MeCreds/{credId}	M	PUT	urn:iam:m.me
Delete My Credential	/factor/v1/MeCreds/{credId}	M	DELETE	urn:iam:m.me
Generate FIDO Transaction Signing Challenge	/factor/v1/FIDOTransactionSigningChallengeGenerator	T	POST	urn:iam:t.authenticate
Verify FIDO Transaction Signing Challenge	/factor/v1/FIDOTransactionSigningChallengeVerifier	T	POST	urn:iam:t.authenticate

IARISK API Privileges

This section covers the **IARISK REST API** endpoints and the specific authorization privileges required to access them.

The following table maps **IARISK REST API** to their required scopes and supported HTTP methods:

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Get My Known Devices	/iarisk/v1/MeKnownDevices	M	GET	urn:iam:m.me
Delete My Known Devices	/iarisk/v1/MeKnownDevices	M	DELETE	urn:iam:m.me
Get My Known Device	/iarisk/v1/MeKnownDevices/{deviceId}	M	GET	urn:iam:m.me
Delete My Known Device	/iarisk/v1/MeKnownDevices/{deviceId}	M	DELETE	urn:iam:m.me
Get User Known Devices	/iarisk/v1/KnownDevices/{userId}	T	GET	urn:iam:t.users urn:iam:t.userdata
Delete User Known Devices	/iarisk/v1/KnownDevices/{userId}	T	DELETE	urn:iam:t.users urn:iam:t.userdata
Get User Known Device	/iarisk/v1/KnownDevices/{userId}/{deviceId}	T	GET	urn:iam:t.users urn:iam:t.userdata
Delete User Known Device	/iarisk/v1/KnownDevices/{userId}/{deviceId}	T	DELETE	urn:iam:t.users urn:iam:t.userdata
IA Risk User Evaluation	/iarisk/v1/UserRiskScoreEvaluator	T	POST	urn:iam:t.risk
Post User Risk Evaluation	/iarisk/v1/PostUserRiskScoreEvaluator	T	POST	urn:iam:t.risk
Client Risk Evaluation	/iarisk/v1/ClientRiskScoreEvaluator	T	POST	urn:iam:t.risk
Post Client Risk Evaluation	/iarisk/v1/PostClientRiskScoreEvaluator	T	POST	urn:iam:t.risk
Evaluate Machine Risk Score	/iarisk/v1/MachineRiskScoreEvaluator	T	POST	urn:iam:t.risk
Post Machine Risk Score Evaluation	/iarisk/v1/PostMachineRiskScoreEvaluator	T	POST	urn:iam:t.risk
Set Risk Threshold	/iarisk/v1/RiskThreshold	T	PUT	urn:iam:t.configs
Update Risky Countries	/iarisk/v1/RiskyCountries	T	PUT	urn:iam:t.configs
Manage IP List	/iarisk/v1/IPList	T	GET	urn:iam:t.configs

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Set Allowed IP List	/iarisk/v1/IPList/ALLOWED	T	PUT	urn:iam:t.configs
Set Blocked IP List	/iarisk/v1/IPList/BLOCKED	T	PUT	urn:iam:t.configs
Manage Risk Policies	/iarisk/v1/Policies	T	GET	urn:iam:t.configs
Manage Login Risk Policies	/iarisk/v1/Policies/login	T	GET	urn:iam:t.configs
Update Risk Rules	/iarisk/v1/RiskRules	T	PUT	urn:iam:t.configs
Get User Risk Score	/iarisk/v1/UserRiskScore	T	GET	urn:iam:t.risk

IASERVICES API Privileges

This section covers the **IASERVICES REST API** endpoints and the specific authorization privileges required to access them.

The following table maps **SCIM 2.0 APIs** to their required scopes and supported HTTP methods:

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Confirm Risk Assessment	/iaservices/risk/v1/confirm	T	POST	urn:iam:t.risk

IDENTITY API Privileges

This section covers the **IDENTITY REST API** endpoints and the specific authorization privileges required to access them.

The following table maps **IDENTITY REST APIs** to their required scopes and supported HTTP methods:

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Validate User Password	/identity/v1/UserPasswordValidator	T	POST	urn:iam:t.identityclient urn:iam:t.authenticate
Change User Password	/identity/v1/UserPasswordChanger	T	POST	urn:iam:t.userpasswordadmin urn:iam:t.authenticate
Generate Temporary Password	/identity/v1/UserPasswordGenerator	T	POST	urn:iam:t.identityclient urn:iam:t.authenticate
Initiate Email Verification	/identity/v1/EmailVerificationInitiator	T	POST	urn:iam:t.identityclient urn:iam:t.authenticate urn:iam:g.securitytokens
Verify Email	/identity/v1/EmailVerifier	T	POST	urn:iam:t.identityclient urn:iam:t.authenticate
Initiate Phone Verification	/identity/v1/PhoneVerificationInitiator	T	POST	urn:iam:t.identityclient urn:iam:t.authenticate urn:iam:g.securitytokens
Verify Phone Number	/identity/v1/PhoneVerifier	T	POST	urn:iam:t.identityclient urn:iam:t.authenticate

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Get Verified Credentials	/identity/v1/VerifiedCredentials	T	GET	urn:iam:t.identityclient urn:iam:t.authenticate urn:iam:t.users
Update My Password	/identity/v1/MePasswordUpdater	M	PUT	urn:iam:m.me
My Registration	/identity/v1/MeRegistration	M	POST	urn:iam:m.me
Initiate My Email Verification	/identity/v1/ MeEmailVerificationInitiator	M	POST	urn:iam:m.me
Verify My Email	/identity/v1/MeEmailVerifier	M	POST	urn:iam:m.me
Initiate My Phone Verification	/identity/v1/ MePhoneVerificationInitiator	M	POST	urn:iam:m.me
Verify My Phone	/identity/v1/MePhoneVerifier	M	POST	urn:iam:m.me
Get My Verified Credentials	/identity/v1/MeVerifiedCredentials	M	GET	urn:iam:m.me
Validate My Password	/identity/v1/MePasswordValidator	M	POST	urn:iam:m.me
Change My Password	/identity/v1/MePasswordChanger	M	POST	urn:iam:m.me
Initiate Password Reset	/identity/v1/PasswordResetInitiator	T	POST	urn:iam:t.users urn:iam:t.identityclient urn:iam:t.authenticate urn:iam:g.securitytokens
Validate Password Reset	/identity/v1/PasswordResetValidator	T	POST	urn:iam:t.identityclient urn:iam:t.authenticate
Change Password via Reset	/identity/v1/PasswordResetChanger	T	POST	urn:iam:t.identityclient urn:iam:t.authenticate
Identity Entity Management	/identity/v1/entity	T	GET	urn:iam:t.identity
Identity Authorization	/identity/v1/authorization	T	POST	urn:iam:t.identity
Identity Query	/identity/v1/query	T	POST	urn:iam:t.identity
Identity Relationship Management	/identity/v1/relationship	T	GET	urn:iam:t.identity
Identity Schema Management	/identity/v1/schema	T	GET	urn:iam:t.identity
My Schema Helper	/identity/v1/MeSchemaHelper	M	GET	urn:iam:m.me
My Subdelegations	/identity/v1/MeSubdelegations	M	GET	urn:iam:m.me urn:iam:m.meclient
Subdelegations Management	/identity/v1/Subdelegations	M	GET	urn:iam:m.me urn:iam:m.meclient
Admin Roles Subdelegator	/identity/v1/AdminRolesSubdelegator	M	POST	urn:iam:m.me urn:iam:m.meclient
Resource Subdelegator	/identity/v1/ResourceSubdelegator	M	POST	urn:iam:m.me urn:iam:m.meclient
Subdelegations Helper	/identity/v1/SubdelegationsHelper	T	GET	urn:iam:t.diagnosticsadmin
Manage Service Accounts	/identity/v1/ServiceAccounts	T	GET	urn:iam:t.identityclient
My Service Accounts	/identity/v1/MeServiceAccounts	M	GET	urn:iam:m.me
My Service Account Locker	/identity/v1/MeServiceAccountLocker	M	POST	urn:iam:m.me

IDSTORE API Privileges

This section covers the **IDSTORE REST API** endpoints and the specific authorization privileges required to access them.

The following table maps **IDSTORE REST APIs** to their required scopes and supported HTTP methods:

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Search for Users	/idstore/v1/Users	T	GET	urn:iam:t.idstoreadmin urn:iam:t.idstorebrowser
Search for Groups	/idstore/v1/Groups	T	GET	urn:iam:t.idstoreadmin urn:iam:t.idstorebrowser
User Locker	/idstore/v1/UserLocker	T	POST	urn:iam:t.idstoreadmin

LDAP API Privileges

This section covers the **LDAP REST API** endpoints and the specific authorization privileges required to access them.

The following table maps **LDAP APIs** to their required scopes and supported HTTP methods:

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Authenticates user against ldap	/ldap/v1/authenticate	T	POST	urn:iam:t.authenticate

ME API Privileges

This section covers the **ME REST API** endpoints and the specific authorization privileges required to access them.

The following table maps **ME REST APIs** to their required scopes and supported HTTP methods:

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Get My Settings	/me/v1/Settings	M	GET	urn:iam:m.me
Update My Settings	/me/v1/Settings	M	PUT	urn:iam:m.me
Get My Preferences	/me/v1/Preferences	M	GET	urn:iam:m.me
Update My Preferences	/me/v1/Preferences	M	PUT	urn:iam:m.me
Get My App Tiles	/me/v1/AppTiles	M	GET	urn:iam:m.me
Create My App Tile	/me/v1/AppTiles	M	POST	urn:iam:m.me
Update My App Tile	/me/v1/AppTiles/{tileId}	M	PUT	urn:iam:m.me
Delete My App Tile Group	/me/v1/AppTiles/Group	M	DELETE	urn:iam:m.me
Service Account Schema Helper	/me/v1/ ServiceAccountSchemaHelper	M	GET	urn:iam:m.me
My Expression Validator	/me/v1/ExpressionValidator	M	POST	urn:iam:m.me

OAUTH2 API Privileges

This section covers the **OAUTH REST API** endpoints and the specific authorization privileges required to access them.

The following table maps **OAUTH 2.0 APIs** to their required scopes and supported HTTP methods:

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
OAuth Authorization Server Discovery	/.well-known/oauth-authorization-server		GET	No scope required
OpenID Connect Discovery	/.well-known/openid-configuration		GET	No scope required
JSON Web Key Set	/oauth2/v1/jwks		GET	No scope required
OAuth Authorization (non-PKCE)	/oauth2/v1/authorize		GET	No scope required
OAuth Authorization (PKCE)	/oauth2/v1/authorize		POST	No scope required
Token Request GET	/oauth2/v1/token		GET	No scope required
OAuth Token Endpoint	/oauth2/v1/token		POST	No scope required
UserInfo Endpoint	/oauth2/v1/userinfo	M	GET	urn:iam:m.me openid
UserInfo Endpoint	/oauth2/v1/userinfo	M	POST	urn:iam:m.me openid
OAuth Logout	/oauth2/v1/logout		GET	No scope required
Token Introspection	/oauth2/v1/introspect		POST	No scope required
Token Revocation	/oauth2/v1/revoke		POST	No scope required
Dynamic Client Registration	/oauth2/v1/register	T	POST	urn:iam:t.dcr
Delete Client	/oauth2/v1/register/{clientId}	T	DELETE	urn:iam:t.dcr
Client Secret Rotation	/oauth2/v1/rotatesecret		POST	No scope required
Device Code User Confirmation	/oauth2/v1/devicecode		GET	No scope required
Device Code Flow	/oauth2/v1/devicecode		POST	No scope required
Personal Token	/oauth2/v1/personaltoken		POST	No scope required
Identity Provider	/oauth2/v1/idp	T	POST	urn:iam:t.authenticate
Bring Your Own Identity	/oauth2/v1/byoi	T	POST	urn:iam:t.authenticate

RISK API Privileges

This section covers the **RISK REST API** endpoints and the specific authorization privileges required to access them.

The following table maps **RISK APIs** to their required scopes and supported HTTP methods:

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Evaluate Risk	/risk/v1/UserRiskScoreEvaluator	T	POST	urn:iam:t.risk
Update Risk	/risk/v1/UserRiskScoreUpdater	T	POST	urn:iam:t.risk
Get users risk score	/risk/v1/UserRiskScoreGetter	T	GET	urn:iam:t.risk

SCIM 2.0 API Privileges

This section covers the **SCIM 2.0 REST API** endpoints and the specific authorization privileges required to access them.

The following table maps **SCIM 2.0 APIs** to their required scopes and supported HTTP methods.

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
SCIM Get Users	/scim2/v1/Users	T	GET	urn:iam:t.users
SCIM Create User	/scim2/v1/Users	T	POST	urn:iam:t.users
SCIM Get User by ID	/scim2/v1/Users/{userId}	T	GET	urn:iam:t.users
SCIM Update User	/scim2/v1/Users/{userId}	T	PUT	urn:iam:t.users
SCIM Patch User	/scim2/v1/Users/{userId}	T	PATCH	urn:iam:t.users
SCIM Delete User	/scim2/v1/Users/{userId}	T	DELETE	urn:iam:t.users
SCIM Get Groups	/scim2/v1/Groups	T	GET	urn:iam:t.groups
SCIM Create Group	/scim2/v1/Groups	T	POST	urn:iam:t.groups
SCIM Get Group by ID	/scim2/v1/Groups/{groupId}	T	GET	urn:iam:t.groups
SCIM Update Group	/scim2/v1/Groups/{groupId}	T	PUT	urn:iam:t.groups
SCIM Patch Group	/scim2/v1/Groups/{groupId}	T	PATCH	urn:iam:t.groups
SCIM Delete Group	/scim2/v1/Groups/{groupId}	T	DELETE	urn:iam:t.groups
SCIM Update My Profile	/scim2/v1/Me	M	PUT	urn:iam:m.me
SCIM Search Users	/scim2/v1/Users/.search	T	POST	urn:iam:t.users urn:iam:t.identityclient
SCIM Search Groups	/scim2/v1/Groups/.search	T	POST	urn:iam:t.groups urn:iam:t.identityclient
SCIM Bulk Operations	/scim2/v1/Bulk	T	POST	urn:iam:t.users urn:iam:t.groups
SCIM Get Schemas	/scim2/v1/Schemas	T	GET	urn:iam:t.users urn:iam:t.groups urn:iam:t.identityclient
SCIM Service Provider Config	/scim2/v1/ServiceProviderConfig	T	GET	urn:iam:t.users urn:iam:t.groups urn:iam:t.identityclient
SCIM Resource Types	/scim2/v1/ResourceTypes	T	GET	urn:iam:t.users urn:iam:t.groups urn:iam:t.identityclient
SCIM2 CAEP Events	/scim2/v1/Caep	T	POST	urn:iam:t.securityevents
SCIM2 Me Resource	/scim2/v1/Me	M	GET	urn:iam:m.me
SCIM2 Users Resource	/scim2/v1/Users	T	GET	urn:iam:t.users
SCIM2 Strict Users Resource	/scim2/v1/strict/Users	T	GET	urn:iam:t.users
SCIM2 Groups Resource	/scim2/v1/Groups	T	GET	urn:iam:t.groups
SCIM2 Strict Groups Resource	/scim2/v1/strict/Groups	T	GET	urn:iam:t.groups
SCIM2 Users Search	/scim2/v1/Users/.search	T	POST	urn:iam:t.users urn:iam:t.identityclient
SCIM2 Strict Users Search	/scim2/v1/strict/Users/.search	T	POST	urn:iam:t.users urn:iam:t.identityclient
SCIM2 Groups Search	/scim2/v1/Groups/.search	T	POST	urn:iam:t.groups urn:iam:t.identityclient

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
SCIM2 Strict Groups Search	/scim2/v1/strict/Groups/.search	T	POST	urn:iam:t.groups urn:iam:t.identityclient
SCIM2 Bulk Operations	/scim2/v1/Bulk	T	POST	urn:iam:t.users urn:iam:t.groups
SCIM2 Strict Bulk Operations	/scim2/v1/strict/Bulk	T	POST	urn:iam:t.users urn:iam:t.groups
SCIM2 Schemas	/scim2/v1/Schemas	T	GET	urn:iam:t.users urn:iam:t.groups urn:iam:t.identityclient
SCIM2 Service Provider Config	/scim2/v1/ServiceProviderConfig	T	GET	urn:iam:t.users urn:iam:t.groups urn:iam:t.identityclient
SCIM2 Resource Types	/scim2/v1/ResourceTypes	T	GET	urn:iam:t.users urn:iam:t.groups urn:iam:t.identityclient
SCIM2 Strict Me Resource	/scim2/v1/strict/Me	M	GET	urn:iam:m.me
SCIM2 Strict Users Resource	/scim2/v1/strict/Users	T	GET	urn:iam:t.users
SCIM2 Strict Groups Resource	/scim2/v1/strict/Groups	T	GET	urn:iam:t.groups
SCIM2 Strict Users Search	/scim2/v1/strict/Users/.search	T	POST	urn:iam:t.users urn:iam:t.identityclient
SCIM2 Strict Groups Search	/scim2/v1/strict/Groups/.search	T	POST	urn:iam:t.groups urn:iam:t.identityclient
SCIM2 Strict Bulk Operations	/scim2/v1/strict/Bulk	T	POST	urn:iam:t.users urn:iam:t.groups
SCIM2 Strict Schemas	/scim2/v1/strict/Schemas	T	GET	urn:iam:t.users urn:iam:t.groups urn:iam:t.identityclient
SCIM2 Strict Service Provider Config	/scim2/v1/strict/ ServiceProviderConfig	T	GET	urn:iam:t.users urn:iam:t.groups urn:iam:t.identityclient
SCIM2 Strict Resource Types	/scim2/v1/strict/ResourceTypes	T	GET	urn:iam:t.users urn:iam:t.groups urn:iam:t.identityclient
SCIM2 Password Authenticator	/scim2/v1/PasswordAuthenticator	T	POST	urn:iam:t.users urn:iam:t.identityclient

UI API Privileges

This section covers the **UI REST API** endpoints and the specific authorization privileges required to access them.

The following table maps **UI REST API** to their required scopes and supported HTTP methods:

API Description	API URI	Global / Tenant / Me	HTTP Method	API Scope
Export DER Certificate	/ui/v1/adminconsole/export-der	T	GET	urn:iam:t.any
Download Admin Console Content	/ui/v1/adminconsole/download-content	T	GET	urn:iam:t.any

Tokens

Learn about Tokens in Symantec Identity Security Platform.

This topic includes the links to the following topics:

- Access Token
- Identity Token
- Refresh Token
- Custom Claims
- Revoke Token
- Using Tokens to Drive Generic Authorization

Access Token

Learn about Symantec Identity Security Platform Access Tokens.

This topic includes the following sections:

- Access Token
- Access Token Examples
- Custom Access Token Expiry and Associated Examples

Access Tokens are bearer security instruments used by REST API clients to gain access to protected APIs. An Access Token is a signed JWT, authorized and issued by IDSP. It represents a client's right to use specific privileges (called OAuth scopes) for the lifetime of the Access Token. Access Tokens have a limited lifetime, controlled by application property (**metadata**) or tenant configuration settings (**accessTokenExpirySecs**) to meet the requirements of corporate security policies. See Managing Applications for "**urn:iam:app:oauthAccessTokenExpirySecs**" app metadata attribute or see Managing Settings for ***"accessTokenExpirySecs"*** setting details. Once a token expires, the client must acquire another Access Token. The following table lists the contents of the Access Token.

Access Token Claims

Claim	Description
tok_type	Claim indicating the type of the token. Token type: AT
sub_type	This claim indicates the type of the subject for which this access token is issued. Token sub type: • USER • CLIENT
ver	Token schema version

Claim	Description
sub	Subject Identifier for which the access token is generated • <user_guid> or app-specific value, if sub_type is USER. If no app-specific token subject mapping attribute is available, then this is the user's GUID (user_guid). Otherwise, this is the value of the physical id store attribute mapped to the logical id store attribute specified by the app's token sub mapping attribute (userTokenSubAttributeMappingName). • Client GUID if sub_type is CLIENT • based on the configuration of "urn:iam:app:oauthCustomSubject" app metadata, this claim can supports Dynamic Substitutions such as ${clientContext.xxx}, ${idProfile.IDprop}
azp	Authorized party - the party to which the ID Token was issued. If present, it MUST contain the OAuth 2.0 Client ID of this party. This Claim is only needed when the ID Token has a single audience value and that audience is different than the authorized party. It MAY be included even when the authorized party is the same as the sole audience. The azp value is a case sensitive string containing a StringOrURI value.
iss	Issuer Identifier for the Issuer of the response. The iss value is a case sensitive URL using the https scheme that contains scheme, host, and optionally, port number and path components and no query or fragment components. Issuer URL representing the signing authority, in the format: https://<platform-base-url>>/<tenant-name>/
nbf	Time until when JWT cannot be accepted. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.
iat	The time at which the JWT was issued. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.
exp	Expiration time on or after which the ID Token MUST NOT be accepted for processing. The processing of this parameter requires that the current date/time MUST be before the expiration date/time listed in the value. Implementers MAY provide for some small leeway, usually no more than a few minutes, to account for clock skew. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time. See RFC 3339 [RFC3339] for details regarding date/times in general and UTC in particular.
jti	A unique identifier for the token, which can be used to prevent reuse of the token. These tokens MUST only be used once, unless conditions for reuse were negotiated between the parties; any such negotiation is beyond the scope of this specification.
client_name	Name of the client
client_tid	Client's tenant name
tid	Resource Tenant Name at the time request for Identity Token was made
aud	Audience(s) that this Access Token is intended for.
scope	The set of scopes exposed by your application for which the application has requested (and received) consent. This claim is space delimited string containing scopes.
user_loginid	User Identifier (only when `sub="user"`)
user_universalid	User's universal id identifier used to map across different identity namespaces (only when `sub="user"`)
azpacr	How client is authenticated at AZ Server: • 0 =public clients • 1 = for confidential clients
user_riskscore	User risk score evaluated (only when `sub="user"`)
idp_type	Identity provider type using which user is authenticated (only when `sub="user"`)
idp_name	Identity provider name using which user is authenticated (only when `sub="user"`)
idp_guid	Identity provider id using which user is authenticated (only when `sub="user"`)

Claim	Description
user_acr	Obtained from IT.acr Example: "user_acr": "0"
user_amr	Obtained from IT.amr Example: "user_amr": ["PWD"]
user_aal	Obtained from IT.aal Example: "user_all": "AAL1"
clientTransactionId	The client transaction received in the /authorize and/or /token endpoint. If no client transaction is received from RP, the SSP's transaction id with a prefix 'ssp:' gets populated
user_authlevel	The authLevel configured in the Policy result gets populated in the Access Token only when the claim, sub_type="user".
user_guid	GUID representing user's internal identifier in IDSP.
scoping_constraints	The scope constraints configured in the Authorization policy using the ".../admin/v1/AdminScopeConstrainer API gets populated in the Access Token. For more information on Scoping Constraints, see the Managing Scoping Constraints section.

Example of a Client Access Token obtained from the Client Credentials flow

Header Parameters in Access Token:

```
{
"x5t#S256": "d-TBloYpys_k122k8gXrQnmfeXb0FxiwKo_6Svw5fTE",
"x5t": "SuWQHOUnyRz3CtYEs3axMkcLFx0",
"kid": "SigningKey",
"alg": "RS256"
}
```

NOTE

The headers x5t and x5t#S256 are added in the header based on the "supportedJoseHeaderParams" configured in the Application. The supportedJoseHeaderParams option supports comma separated - x5t, x5t#s256, x5c Jose header parameters. If this option is left empty, then x5t and x5t#S256 are added in the JOSE Header of JWS tokens by default.

Body of the Access Token

```
{
 "client_tid": "default",
 "sub": "2db6e0b1-8f69-4f1c-b308-6ddacdca76f1",
 "ver": "1.0",
 "iss": "https://example.com/default/",
 "azpacr": "1",
 "tok_type": "AT",
 "tid": "default",
 "aud": "https://example.com/default/",
 "nbf": 1592160941,
 "sub_type": "CLIENT",
 "azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
 "scope": "urn:iam:t.apps urn:iam:t.userdata urn:iam:t.ldapconfigs urn:iam:t.configs urn:iam:t.authpolicies
 urn:iam:t.certs urn:iam:t.risk urn:iam:t.authzpolicies urn:iam:t.groups urn:iam:t.users
 urn:iam:t.authenticate urn:iam:t.signingkey",
 "exp": 1592164541,
```

```
 "iat": 1592160941,
 "client_name": "testapp",
 "jti": "f41468ad-ec8d-4a9f-814d-6f5fe07377fe"
}
```


Example of a User Access Token obtained from the Authorize flow for requested REST API scopes


```
{
 "client_tid": "default",
 "sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
 "ver": "1.0",
 "idp_name": "SecureLDAP",
 "user_universalid": "KevinLUCAS",
 "user_loginid": "kevinlucas",
 "idp_guid": "f77d69c3-718b-4b64-be5c-767e42a1213c",
 "iss": "https://example.com/default/",
 "azpacr": "1",
 "tok_type": "AT",
 "tid": "default",
 "aud": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
 "nbf": 1592174525,
 "sub_type": "USER",
 "azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
 "scope": "urn:iam:t.authenticate urn:iam:t.configs",
 "exp": 1592178125,
 "iat": 1592174525,
 "client_name": "testapp",
 "jti": "bd9b9eb9-5053-4073-a4c8-26d9170afe21",
 "idp_type": "ldap"
 "scoping_constraints":["33086569-1bdf-4c10-8eec-7edb7ab88121", "44123183-8caf-2c10-7eec-3edb1ab28130"],

}
```


Example of a User Access Token obtained from OpenID Connect's Authorize flow


```
{
 "client_tid": "default",
 "sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
 "ver": "1.0",
 "idp_name": "SecureLDAP",
 "user_universalid": "KevinLUCAS",
 "user_loginid": "kevinlucas",
 "idp_guid": "f77d69c3-718b-4b64-be5c-767e42a1213c",
 "iss": "https://example.com/default/",
 "azpacr": "1",
 "tok_type": "AT",
 "tid": "default",
 "aud": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
 "nbf": 1592175932,
 "sub_type": "USER",
 "azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
 "scope": "openid email",
 "exp": 1592179532,
 "iat": 1592175932,
```

```
  "client_name": "testapp",
  "jti": "488a7f51-7ec8-4aae-ac46-a166fa844213",
  "idp_type": "ldap"
}
```

Custom Access Token Expiry

You can define a custom expiration value for access tokens when making a request to IDSP. By default, access tokens expire after one hour, but this can be adjusted either in the Application's metadata attribute or in the Tenant Settings.

In the application, the metadata attribute is labeled as "**urn:iam:app:oauthAccessTokenExpirySecs**," while in the Tenant Settings, it is labeled as "**accessTokenExpirySecs**." For more details, see Managing Applications or see Managing Settings for setting details.

When including a custom expiration value in the access token request, the following points should be considered:

- If only the Tenant setting is configured and the Application's metadata attribute is not, the custom expiration value overrides the Tenant setting if its value is shorter than the tenant setting value. If the custom expiration exceeds the Tenant setting, the Tenant setting takes precedence.
- If both the Application's metadata attribute and Tenant setting are configured, the app metadata value takes precedence. In this case, if the custom expiration value is shorter than the app metadata value, it overrides. If the custom expiration exceeds the application metadata attribute value, the application metadata configuration takes precedence.

  However, the custom expiration is still bound by the overall access token expiration configured in the settings. This parameter can be used in any of the supported flows.

Example of custom access token expiry

The following sample displays the custom expiry timeframe added to a token request.

```
curl -k --request POST \--url 'https://example.com/default/oauth2/v1/token' \--header 'Content-Type:
 application/x-www-form-urlencoded' \-u "6cb446b9-33bf-4b97-a6e1-e7ad389974d2:2ef66db2-1e51-4868-8c72-
bd57ce7f69dc" \--data-urlencode 'grant_type=client_credentials' \--data-urlencode 'scope=urn:iam:myscopes'--
data-urlencode'expirySeconds=120'
```

This is the response to the requested token.

```
{
 "access_token": "eyJ4NXQjUzI1NiI6ImNJ......",
 "token_type": "Bearer",
 "expires_in": 120
}
```

Identity Token

Learn about Symantec Identity Security Platform Identity Tokens.

This topic includes the following sections:

- Identity Token
- Identity Tokens Examples
- Handling Identity Tokens
- Short-Lived Identity Token

Identity Tokens

Identity Tokens represent the user whose identity has been authenticated by IDSP, either through the OpenID Connect flow or programmatically through the REST API. This token is a signed JWT, authenticated and issued by IDSP Identity Tokens have a limited lifetime, controlled by application property (**metadata**), or tenant configuration setting ((**idTokenExpirySecs**) to meet the requirements of corporate security policies. See Managing Applications for "**urn:iam:app:oauthIdentityTokenExpirySecs**" application's metadata attribute or see Managing Settings for ***"idTokenExpirySecs"*** setting details. Once a token expires, the client has the responsibility to acquire another identity token by re-authenticating the user.

When validating the signature of the Identity Token Hint provided into "/authorize" or "/token" (JWT-BEARER grant), the certificate used for validating the token signature is looked up in this order: x5t+sha256, x5t, kid+alias, and finally the issuer claim.

The following table lists the contents of the Identity Token:

Identity Token contents

Name	Claim Description (openid.net)
tok_type	Claim indicating the type of the token Token type: `IT`
iss	Issuer Identifier for the Issuer of the response. The iss value is a case sensitive URL using the https scheme that contains scheme, host, and optionally, port number and path components and no query or fragment components Issuer URL representing the signing authority, in the format: https://<SSP_HOST>/<tenant-name>/
sub	Subject Identifier. <user_guid> or app-specific value. If no app-specific token subject mapping attribute is available, then this is the user's GUID (user_guid). Otherwise, this is the value of the physical id store attribute mapped to the logical id store attribute specified by the app's token sub mapping attribute (userTokenSubAttributeMappingName).
user_loginid	User’s id used during authentication
user_universalid	User's universal id identifier, used to map across different identity namespaces
tid	Resource Tenant Name at the time request for Identity Token was made
at_hash	Access Token hash value if Access Token was issued as part of the OpenID Connect request.
client_name	Name of the client requesting the OpenID Connect flow
azp	Authorized party - the party to which the ID Token was issued. If present, it MUST contain the OAuth 2.0 Client ID of this party. This Claim is only needed when the ID Token has a single audience value and that audience is different than the authorized party. It MAY be included even when the authorized party is the same as the sole audience. The azp value is a case sensitive string containing a StringOrURI value.

Name	Claim Description (openid.net)
aud	Audience(s) that this ID Token is intended for. It MUST contain the OAuth 2.0 client_id of the Relying Party as an audience value. It MAY also contain identifiers for other audiences. In the general case, the aud value is an array of case sensitive strings. In the common special case when there is one audience, the aud value MAY be a single case sensitive string.
iat	Time at which the JWT was issued. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.
nbf	Time until when JWT cannot be accepted. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.
auth_time	Time when the End-User authentication occurred. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time. When a max_age request is made or when auth_time is requested as an Essential Claim, then this Claim is REQUIRED; otherwise, its inclusion is OPTIONAL. (The auth_time Claim semantically corresponds to the OpenID 2.0 PAPE [OpenID.PAPE] auth_time response parameter.)
sid*	Session ID assigned by the Login service. Login/AuthN issuing IT always returns the SID associated with the user's session
session_exp	Session ID assigned by the Login service. Login/AuthN issuing IT always returns the SID associated with the user's session
idp_type	Type of IDP. Possible values are `ldap`, `local`, `fed`
idp_name	Name of the IDP
idp_guid	GUID of the IDP
nonce	String value used to associate a Client session with an ID Token, and to mitigate replay attacks. The value is passed through unmodified from the Authentication Request to the ID Token. If present in the ID Token, Clients MUST verify that the nonce Claim Value is equal to the value of the nonce parameter sent in the Authentication Request. If present in the Authentication Request, Authorization Servers MUST include a nonce Claim in the ID Token with the Claim Value being the nonce value sent in the Authentication Request. Authorization Servers SHOULD perform no other processing on nonce values used. The nonce value is a case sensitive string. Not present when IT is generated by /authenticatePresent when IT is returned by /Authorize from OIDC
exp	Expiration time on or after which the ID Token MUST NOT be accepted for processing. The processing of this parameter requires that the current date/time MUST be before the expiration date/time listed in the value. Implementers MAY provide for some small leeway, usually no more than a few minutes, to account for clock skew. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time. See RFC 3339 [RFC3339] for details regarding date/times in general and UTC in particular.
jti	A unique identifier for the token, which can be used to prevent reuse of the token. These tokens MUST only be used once, unless conditions for reuse were negotiated between the parties; any such negotiation is beyond the scope of this specification.

<table>
<tr><th>Name</th><th>Claim Description (openid.net)</th></tr>
<tr><td>aal*</td><td>Authentication Assurance assurance</td></tr>
<tr><td>acr</td><td>Authentication Context Class Reference value requested and authenticated at.</td></tr>
<tr><td>amr</td><td>Authentication Methods References. JSON array of strings that are identifiers for authentication methods used in the authentication. For instance, values might indicate that both password and OTP authentication methods were used. The definition of particular values to be used in the amr Claim is beyond the scope of this specification. Parties using this claim needs to agree upon the meanings of the values used, which may be context-specific. The amr value is an array of case sensitive strings.Standard amr values are captured from https://tools.ietf.org/html/draft-ietf-oauth-amr-values-02
The following table contains the authentication factors and their mappings with the AMR values in IDSP:
<table><tr><th><u>Authentication Factors</u></th><th>AMR Values</th></tr><tr><td>Password</td><td>PWD</td></tr><tr><td>SMSOTP, EmailOTP</td><td>OTP</td></tr><tr><td>Push</td><td>PUSH</td></tr><tr><td>MobileOTP</td><td>MOTP</td></tr><tr><td>Biometric</td><td>FIDO</td></tr><tr><td>Security Key</td><td>HWK</td></tr></table></td></tr>
<tr><td>user_riskscore</td><td>Risk score of the user computed during the authentication flow</td></tr>
<tr><td>ver</td><td>Token schema version number</td></tr>
<tr><td>email</td><td>User's email Id
Note:
• LDAP Config should have email mapping for the id store attribute.
• App configuration must have 'skipEmailForIT' as 'false'.</td></tr>
<tr><td>user_authlevel</td><td>The authLevel configured in the Policy result gets populated in the Identity Token.</td></tr>
<tr><td>user_guid</td><td>GUID representing user's internal identifier in IDSP.</td></tr>
</table>

Example of an Identity Token obtained from OpenID Connect's Authorize flow

Header Parameters in Identity Token

```
{
 "x5t#S256": "d-TBloYpys_k122k8gXrQnmfeXb0FxiwKo_6Svw5fTE",
 "x5t": "SuWQHOUnyRz3CtYEs3axMkcLFx0",
 "kid": "SigningKey",
 "alg": "RS256"
}
```

NOTE
The headers x5t and x5t#S256 are added in the header based on the "supportedJoseHeaderParams" configured in the Application. The supportedJoseHeaderParams option supports comma separated - x5t,

x5t#s256, x5c Jose header parameters. If this option is left empty, then x5t and x5t#S256 are added in the JOSE Header of JWS tokens by default.

Body of Identity Token


```
{
  "at_hash": "QBaiXONRKD-ev6popw-VoQ",
  "sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
  "ver": "1.0",
 "idp_name": "SecureLDAP",
 "user_universalid": "KevinLUCAS",
 "amr": [
    "PWD"
    ],
 "user_loginid": "kevinlucas",
 "idp_guid": "f77d69c3-718b-4b64-be5c-767e42a1213c",
 "iss": "https://example.com/default/",
 "tok_type": "IT",
 "nonce": "mynonce",
 "aal": "AAL1",
 "tid": "default",
 "sid": "fadc0d87-bbd2-4e5f-b37e-f2adf8e91160",
 "aud": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
 "acr": "urn:iam:acr:aal1",
 "nbf": 1592175932,
 "azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
 "auth_time": 1592175920,
 "exp": 1592197532,
 "iat": 1592175932,
 "client_name": "testapp",
 "jti": "3f21900d-e843-4a20-a846-60007394ef1f",
 "idp_type": "ldap",
 "user_authlevel": "125"
}
```


Handling Identity Tokens

The following section explains how an Identity Token JWT is processed in the following flows:

- JWT-BEARER grant (programmatic flow)
- /authorize (web flow)
- /authenticate REST API (programmatic flow)

The Identity Token JWT should conform to the OpenID Identity Token specification as outlined in the OpenID Connect Core specification. The JWT may be either signed, encrypted, or both. The flow validates the token's signature, expiration, audience, and then establishes the user identity according to the process stated in the following section.

Steps to handle Identity Tokens:

If the Identity Token was issued by IDSP (issuer claim indicates the IDSP's tenant issuer), the claims already contain the necessary information to establish the identity. No special processing is required. Otherwise, for identity tokens issued by any other issuer, the processing is as follows:

1. **If the IDP is Not Found**:

- **ZFP flow**: The `idp_id` and `user_loginid` are derived from the Identity Token (IT), with the sub claim used as a fallback for the `user_loginid` .
- **Non-ZFP flow**: The `idp_id` and `user_loginid` are taken from the Identity Token. Currently, there is no fallback to the sub claim for `user_loginid` . During the JWT-BEARER /token flow, the `user_loginid` must be successfully disambiguated against an Identity Store, otherwise the JWT-BEARER request fails.

2. **If the IDP is Found**:
   - **ZFP flow**: The `idp_id` is taken from the IT unless overridden by the IDP's `idp_id` if the IDP has the `"zfpUsesIdpAsIdentitySource=true"` metadata (this is set to false by default). The `user_loginid` is derived from the IT's claim that corresponds to the IDP's Token Subject Claim, with a fallback to sub. This is the current behavior for Siteminder or IDSP ZFP integration.
   - **Non-ZFP flow**: The `idp_id` and `user_loginid` are determined through a lookup in the ID Store using IT's claim that corresponds to the IDP's Account Mapping Attribute, with a fallback to user_loginid claim. The search looks for the value of the IT's claim that matches the IDP's Token Subject Claim, falling back to sub.

Short-Lived Identity Tokens

A short-lived identity token is a regular identity token issued by IDSP with shorter expiration time that is controlled by a tenant setting, and is not subjected to tracking. This identity token type is primarily used for ID propagation and to address security issues of replay attacks. Short-lived identity tokens have a default expiry of 30 seconds, which may be changed using the tenant setting, '***ssoIdPropTokenExpirySeconds***'. The property value can be modified to a range between 10 and 60 seconds.

> **NOTE**
> With the 'openid' and 'urn:iam:sso:idprop' scope, the refresh token grant flow can generate short-lived identity tokens in a manner similar to the jwt-bearer flow shown in the following example.

To generate short-lived identity tokens, set the open id scope in the application to "urn:iam:sso:idprop," where 'openid' is required to generate the "id_token" and the scope - 'urn:iam:sso:idprop' specifies that the identity token should be short-lived.

Configurations required in the application to generate a short lived identity token for jwt-bearer grant

- To generate short-lived identity token using the "oauth2/v1/token" API, configure the following in the application:
  - **allowed OpenID scope** - 'openid'
  - **allowed OpenID custom scope** - 'urn:iam:sso:idprop'
  - **clientType** - 'Trusted', 'Confidential', or 'Public'
  - **allowed operations** - 'offlineit' if the client is configured as Confidential or Public

Sample request of short-lived identity token in the jwt-bearer flow

Request

```
curl -k --request POST \
--url 'https://example.com/default/oauth2/v1/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
-u "6cb446b9-33bf-4b97-a6e1-e7ad389974d2:2ef66db2-1e51-4868-8c72-bd57ce7f69dc" \
--data-urlencode 'grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer' \
--data-urlencode 'scope=openid urn:iam:sso:idprop' \
--data-urlencode 'assertion=eyJ4NXQjUzI1.........5y3fw'
```

Response

```
{
"access_token": "eyJ4NXQjUzI1NiI6IjZqZlh5bHV6QnZmX0NOSkJnY...",
"id_token": "IJ4NeyQjUzX1NiI6IjZqZlh5bHV6QnOSkJnYZmX0N..."
}
```

Refresh Token

Learn about Symantec Identity Security Platform Refresh Token

This topic includes the following sections:

- Refresh Token
- Steps to obtain a Refresh Token
- Step to obtain a new Access Token using the Refresh Token grant
- Step to obtain both, new Access Token and new Refresh Token using the Refresh Token grants

Refresh Tokens are credentials used to re-generate an Access Token without re-authenticating, when the current access token becomes invalid or expires. When using the Refresh Token, you may request for the same scopes that were requested during the initial authentication or narrow down the scopes depending on your requirement. Refresh Tokens have a limited lifetime, controlled by application property (**metadata**), or tenant configuration setting (**refreshTokenExpiryDays**) to meet the requirements of corporate security policies. See Managing Applications for "**urn:iam:app:oauthRefreshTokenExpiryDays**" application's metadata attribute or see Managing Settings for ***"refreshTokenExpiryDays"*** setting details. Refresh Tokens are secrets and not JWT. To make use of the Refresh Token functionality, you should update the following endpoints with the scope, "**offline_access**":

- "/authorize"
- "/token"

> **NOTE**
> To know about additional settings related to the refresh token, look for the following settings in the Managing Settings page.
>
> - keepOldRefreshTokenOnRefreshTokenFlowWithOfflineAccess
> - keepOldSessionIdOnNewRefreshTokenFlowWithOfflineAccess

> **NOTE**
> To ensure accurate and reliable IP address usage in complex network environments (which may alter user IP information due to firewalls or load balancers), leverage the IDSP application metadata. To know more, see HTTP Header and End User IP Address Usage.

Steps to obtain a Refresh Token

The following step shows an example of how to send a Refresh Token request using "/token" and "/authorize" endpoint.

Perform the following step:

1. In the "/token" endpoint or "/authorize" request body, enter the scope as "**offline_access**" and invoke the API. The Refresh Token will be obtained.

   > **NOTE**
   > The "**offline_access**" scope only works with '**authorization_code**', '**jwt-bearer**' or '**device_code**' grant_types.

   Example of a Refresh Token request format in the "/token" endpoint:

   ```
   curl -k --request POST --url 'https://example.com/default/oauth2/v1/token' --header 'Content-Type:
    application/x-www-form-urlencoded' -u "6cb446b9-33bf-4b97-a6e1-e7ad389974d2:2ef66db2-1e51-4868-8c72-
   bd57ce7f69dc" --data-urlencode 'grant_type=authorization_code' --data-urlencode
    'code=sodgnomNDVB723a3dPRwMBGHTYHB' --data-urlencode 'scope=offline_access'
   ```

   Example of a Refresh Token request format in the "/authorize" endpoint:

   ```
   https://example.com/default/oauth2/v1/authorize?response_type=code&client_id=6cb446b9-33bf-4b97-a6e1-
   e7ad389974d2&redirect_uri=https://
   www.example.com&scope=urn:iam:myscopesopenidemailoffline_access&state=mystate
   ```

   Example of a Refresh Token response format:

   ```
   {
   ```

```
"access_token" : "eyJ4NXQjUzI1NiI6ImNJbUNYN1FyRy1BUUpkWWZ2Wlctd1...X6HEbM-Hdh-hzXqHA"

"token_type" : "Bearer",

"expires_in" : 3600,

"id_token" : "eyJ4NXQjUzI1NiI6ImNJbUNYN1FyRy1BUUpkWWZ2W....3qliSzy_Khn2wgqQ2bg2t8Keg",

"refresh_token": "m5370ROmNDVB7a3dPRwMs5FDTZjfFc"

}
```

Step to obtain a new Access Token using the Refresh Token grant

Perform the following step:

1. In the "/token" endpoint request body, enter the value of the Refresh Token that you obtained in the previous section and invoke the "/token" API. The new Access Token is obtained.
   Example of a request body that includes the refresh_token which is used to obtain the new Access Token:

   ```
   curl -k --request POST --url 'https://example.com/default/oauth2/v1/token' --header
    'Content-Type: application/x-www-form-urlencoded' -u "6cb446b9-33bf-4b97-a6e1-
   e7ad389974d2:2ef66db2-1e51-4868-8c72-bd57ce7f69dc" --data-urlencode 'grant_type=refresh_token' --data-
   urlencode 'refresh_token=m5370ROmNDVB7a3dPRwMs5FDTZjfFc' --data-urlencode 'scope=<subset of scopes>'
   ```

   The following table shows the request parameters required for getting the new Access Token:

Name	Description
client_id	Client Id of the client that was registered with IDSP.
client_secret	Client secret that is associated with the client_id.
grant_type	The grant type supported for generating a new Access Token using the Refresh Token is "**refresh_token**". This grant type can be used only with 'authorization_code', 'jwt-bearer' or 'device_code' grant type.
refresh_token	Value of the Refresh Token.
scope	Subset of scopes.

   Example of a response body which grants the new Access Token:

   ```
   {

   "access_token" : "eyJ4NXQjUzI1NiI6ImNJbUNYN1FyRy1BUUpkWWZ2Wlctd1...X6HEbM-Hdh-hzXqHA",

   "token_type" : "Bearer",

   "expires_in" : 3600

   }
   ```

Step to obtain both, new Access Token and new Refresh Token using the Refresh Token grants

Perform the following step:

1. In the "/token" endpoint request body, enter the grant type as "refresh_token" and the value of the "refresh_token" that you have obtained from the "Step to obtain Refresh Token" section.

```
curl -k --request POST --url 'https://example.com/default/oauth2/v1/token' --header
 'Content-Type: application/x-www-form-urlencoded' -u "6cb446b9-33bf-4b97-a6e1-
e7ad389974d2:2ef66db2-1e51-4868-8c72-bd57ce7f69dc" --data-urlencode 'grant_type=refresh_token ' --data-
urlencode 'refresh_token=m5370ROmNDVB7a3dPRwMs5FDTZjfFc' --data-urlencode 'scope=<subset of scopes>'
```

   NOTE
   To obtain a new Refresh Token, passing ***scope= "offline_access"*** in the "/token" endpoint is mandatory.

   Example of a response body that grants the new Access Token and the Refresh Token:

```
{

"access_token" : "eyJ4NXQjUzI1NiI6ImNJbUNYN1FyRy1BUUpkWWZ2Wlctd1...X6HEbM-Hdh-hzXqHA",

"token_type" : "Bearer",

"expires_in" : 3600,

"refresh_token": "tGzv3JOkF0XG5Qx2TlKWIA"

}
```

Reference Links

- Personal Refresh Tokens (PRT)

Personal Refresh Token (PRT)

Personal Refresh Tokens (PRTs) enable users to request long-lived tokens that allow client applications to generate Access Tokens with custom data.

Personal Refresh Tokens (PRTs) are long-lived tokens tied to a specific user and client application, primarily used for offline access. It allows an user to request a long-lived token enabling an app client in possession of such Refresh Token (RT) to then generate Access Tokens with custom claims containing app's custom data (Client Context) stored in RT.

This topic includes the following sections:

- Define custom claims
- Configure app metadata to set the expiry of the Personal Refresh Token
- Pass scopes in ".../oauth2/v1/token" endpoint (using Authorization Code Flow)
- Pass dynamic values to the Client Context in ".../oauth2/v1/token" in Authorization Code Flow
- Introspect of resulting PRT
- Redeem Personal Refresh Token (PRT)
- Manage Personal Refresh Token (PRT)

Define custom claims

Navigation path to the Apps in the Admin Console:
Select the Application > OAuth/OpenID > Custom Claims

In the Custom Claims section, define your additional, non-standard claims which may be included in the Access Token and/or Identity Token based on your configuration. These custom claims can also use the RT's Client Context provided by the application during RT initialization or while calling the ".../oauth2/v1/token" endpoint.

When a client application is authorized to use the "/token" endpoint, with the OpenID Connect "offline_access" scope used to receive an RT, it can include custom "Client Context" information. This "Client Context" information is then securely stored by the Authorization Server and associated with the Refresh Token (RT) making it available when issuing new AT's. When that RT is later used to get a new Access Token, the Authorization Server can retrieve and use the associated Client Context.

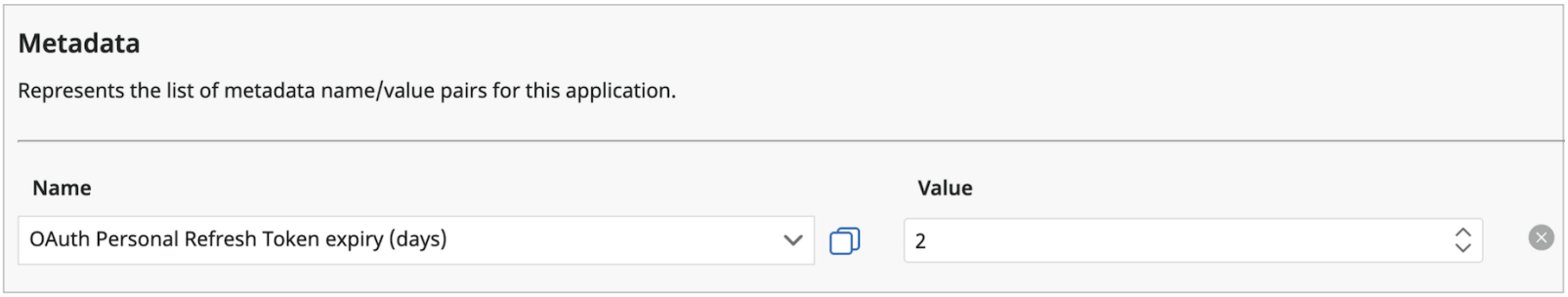
Configure app metadata to set the expiry of the Personal Refresh Token

By default, the PRT is set to have an expiry of 7 days. The application can be configured to allow different RT expiry by configuring the "**OAuth Personal Refresh Token expiry (days)**" metadata. The application client can ask for an RT with an expiration equal or less than this value using the "**expiryDays**" request param.

Navigation path to the app metadata in the **Admin Console**:
Select the Application > General > Metadata

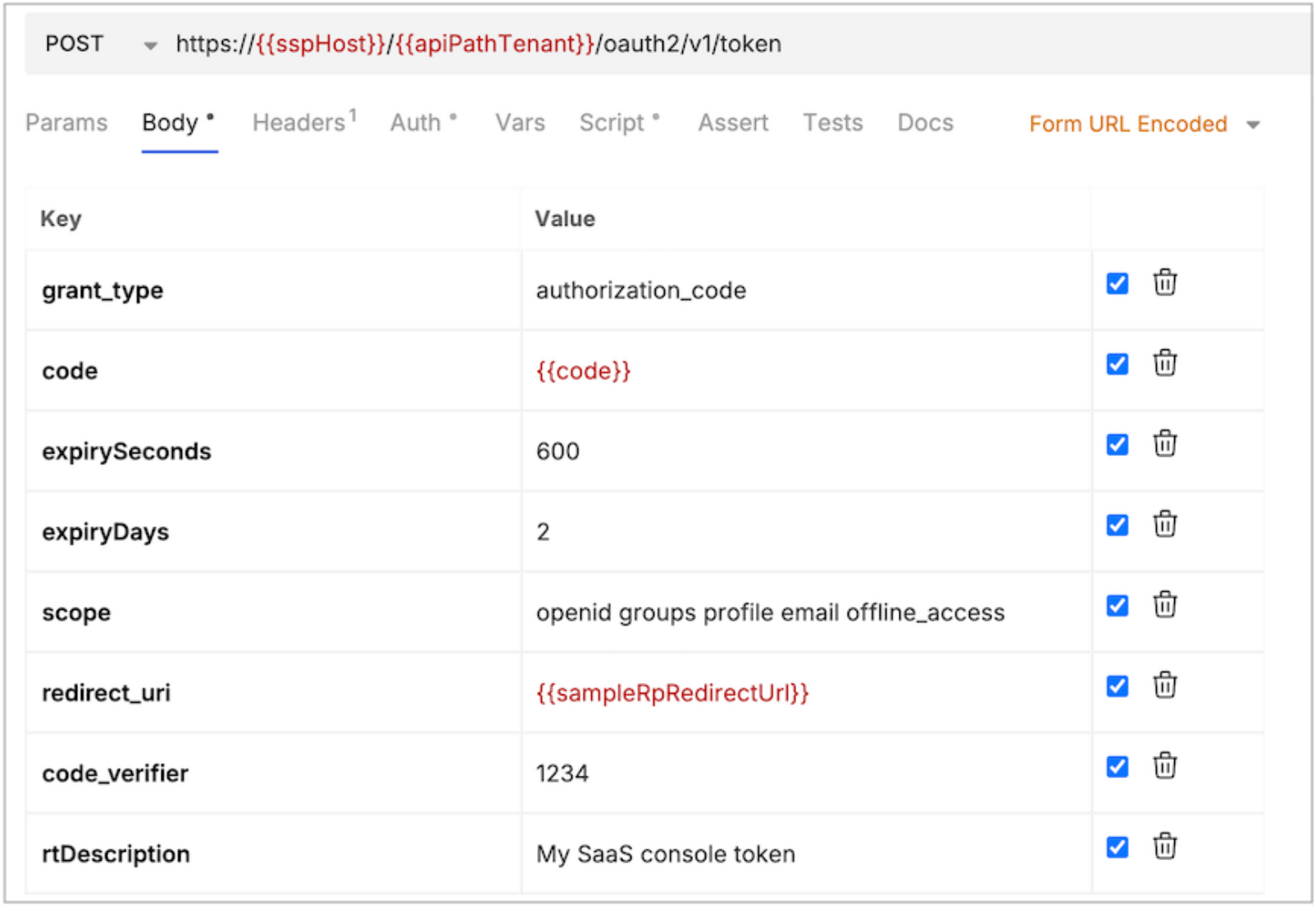
The following illustration displays how you can define the app metadata:

Figure 13: app metadata config in PRT



To configure the expiry **using the API**, the app client needs to invoke "/token" with "**expiryDays**" request param whose value must be equal to or less than the value in effect for the app client.

Figure 14: Passing "expiryDays" in header params



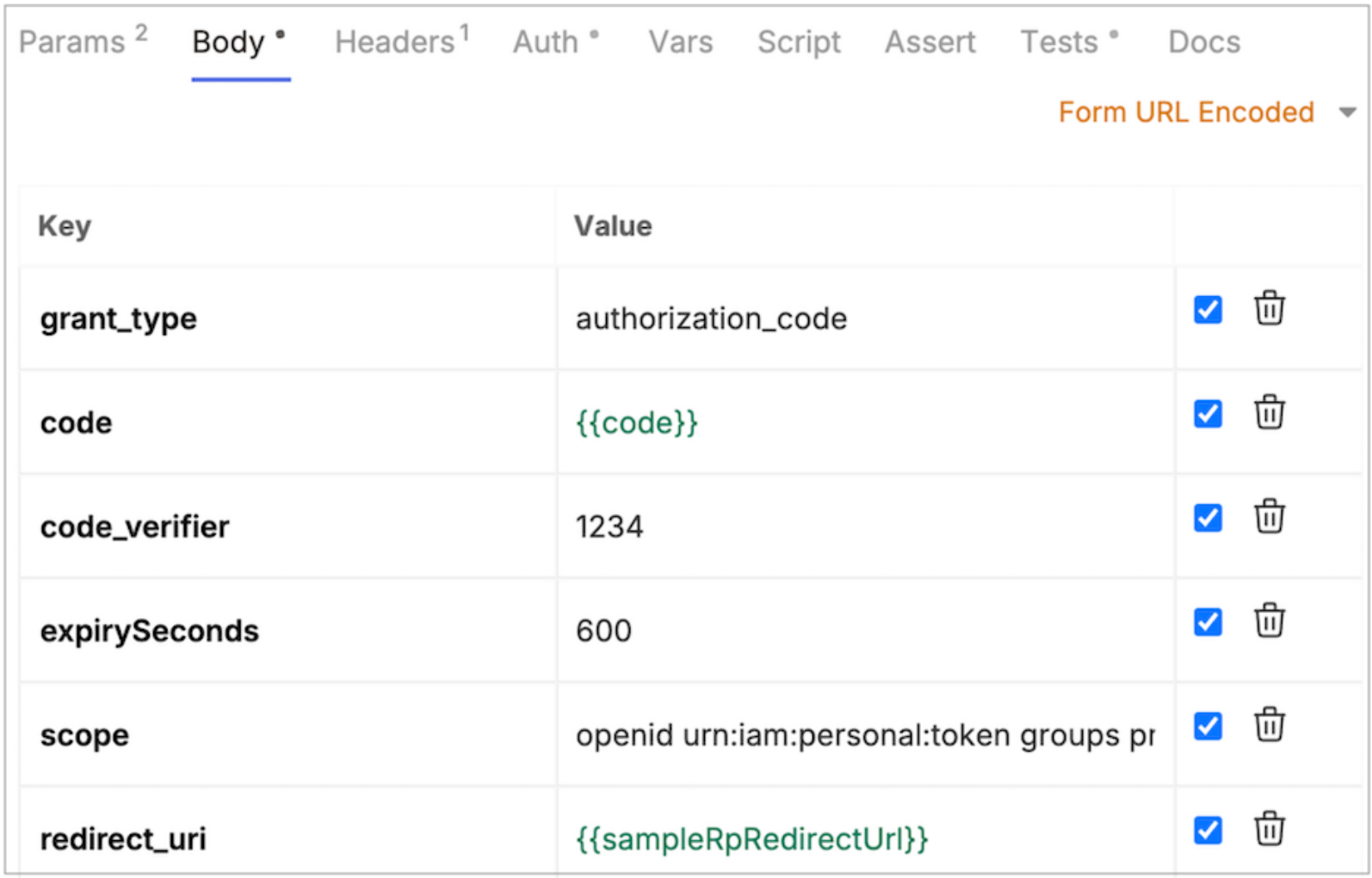
The PRTs expiry is determined by the application's metadata configuration, if not set then it falls back to the "**refreshTokenExpiryDays**" tenant settings. This setting is covered here.

Pass scopes in ".../oauth2/v1/token" endpoint (using Authorization Code Flow)

The client must include two specific "scopes" in the "/token" endpoint to request for a PRT:

- **"offline_access"**: This scope is required to obtain a Refresh Token, which allows the client to get new Access Tokens without the user having to re-authorize.
- "**urn:iam:personal:token**": This is a custom scope that indicates the client is requesting a Personal Refresh Token (PRT), which has specific characteristics for end-user management.

Figure 15: Defining Scopes in PRT

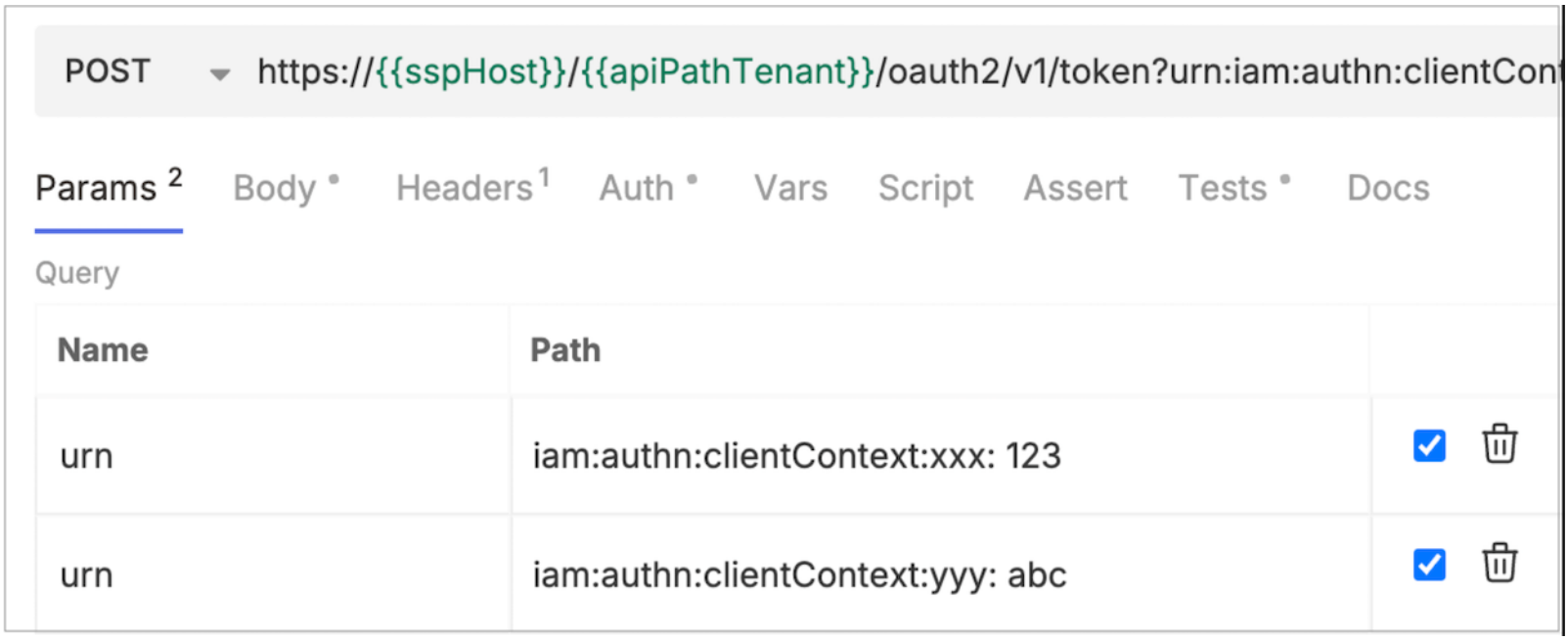


Pass dynamic values to the client context in ".../oauth2/v1/token" in authorization code clow

The client can include one or more URL-encoded query string parameters in the format "urn:iam:authn:clientContext:xxx=123" (this is an example value). These parameters are used to embed custom data (the "Client Context") into the PRT being created. In the example, "xxx" represents the key for the custom data, and "123" is the corresponding value. Similarly, there is another example, "urn:iam:authn:clientContext:xxx=abc" where "yyy" represents the key for the custom data, and "abc" is the corresponding value.

The following illustration displays how you can define the client context:

Figure 16: PRTClientContextConfig



The client can include multiple such parameters to store a list of key-value pairs in the PRT's Client Context.

Introspect of resulting PRT

After the refresh token is generated using the ".../oauth2/v1/token", this token can be used in the ".../oauth2/v1/introspect" endpoint with PRT o check if the client context values have passed correctly to the token. The following example shows the response containing PRT's Client Context from which custom claims are obtained when generating Access Token for this PRT.

```
/Introspect of resulting PRT
{

  "active": true,

 "personalRefreshToken": true,
  "urn:iam:authn:clientContext": [
     {
         "key": "myXXX",
        "value": "123"
     },
      {
          "key": "myYYY",
          "value": "abc"
      }
  ],

  "scope": "urn:iam:m.me openid profile groups email",

  "sub": "admin",

  "sid": "b3d7d5e1-0d93-4700-af60-71fba29cda8f",
```

```
"amrs": [

    "PWD"

],

"acr": "urn:iam:acr:aal1",

"aal": "AAL1",

"iss": "https://example.com/default",

"aud": [

    "929be3b0-54f3-4e31-a8b4-7518c1dedf99"

],

"exp": 1747128141,

"iat": 1746523341,

"nbf": 1746523341,

"client_id": "929be3b0-54f3-4e31-a8b4-7518c1dedf99",

"client_tid": "7f0ebdeb-0f54-43e0-b059-2238f59010f2",

"client_tname": "default",

"tid": "7f0ebdeb-0f54-43e0-b059-2238f59010f2",

"tname": "default",

"user_loginid": "admin",

"user_dn": "0129b61a-1d22-4a45-bade-1dee4e86938c",

"email": "example@01.com",

"user_guid": "0129b61a-1d22-4a45-bade-1dee4e86938c",

"idp_guid": "internal",

"idp_name": "internal",

"idp_type": "internal",

"user_zfp": "false",

"user_subattrmappingname": "user_loginid",
```

```
  "user_loginidattrmappingname": "user_loginid",

  "hasgroups": false,

  "token_type": "Bearer",

  "tok_type": "RT",

  "session_ttl_secs": 0,

  "session_idle_ttl_secs": 0

}
```


Redeem Personal Refresh Token (PRT)

To redeem a PRT for access token containing PRT's custom claims, the app client needs to invokes ".../oauth2/v1/token" with the regular Refresh Token grant. The claim mapping as configured on the app causes custom claim "**myXXX**" to be given value "123" as represented by PRT's Client Context containing key "**xxx"** with the value of "123".


```
{

"sub": "admin",

"idp_name": "internal",

"user_dn": "0129b61a-1d22-4a45-bade-1dee4e86938c",

"user_loginid": "admin",

"idp_guid": "internal",

"iss": "https://example.com/default/",

"user_guid": "0129b61a-1d22-4a45-bade-1dee4e86938c",

"tid": "7f0ebdeb-0f54-43e0-b059-2238f59010f2",

"transactionid": "11ca235d-84f9-4612-921c-a55bd349c20f",

"sid": "ed0aa229-657b-42c2-b241-c97e33f3ee9c",

"user_subattrmappingname": "user_loginid",

"myXXX": "123",
"myYYY": "abc",

"sub_type": "USER",

"azp": "929be3b0-54f3-4e31-a8b4-7518c1dedf99",

"user_loginidattrmappingname": "user_loginid",
```

```
"scope": "urn:iam:m.me profile groups email",

"session_ttl_secs": 627,

"hasgroupids": false,

"exp": 1762096427,

"iat": 1746544427,

"client_name": "DemoClient",

"jti": "090ea961-db13-4c36-a5c3-53c858a759a8",

"idp_type": "internal",

"scoping_constraints": [

  "ac3a567a-c6dc-426d-8ccc-f3c3129d2cd3"

],

"client_tname": "default",

"client_tid": "7f0ebdeb-0f54-43e0-b059-2238f59010f2",

"ver": "1.0",

"hasgroups": false,

"azpacr": "2",

"tname": "default",

"tok_type": "AT",

"user_riskscore": 0,

"user_amr": [

  "PWD"

],

"aud": "https://example.com/default/",

"nbf": 1746544427,

"user_zfp": false,

"client_transactionid": "ssp:11ca235d-84f9-4612-921c-a55bd349c20f",

"user_aal": "AAL1",
```

```
"tok_source": "RFR",

"user_acr": "urn:iam:acr:aal1"

}
```

Manage Personal Refresh Token (PRT)

There are two ways to manage the generated PRT, these are :

- Retrieve the PRTs of an user
- Revoke PRT of an User

Retrieve the PRTs of an user

To retrieve only the PRT of an user, invoke the following endpoint with "**revokepersonaltokens=true**" query string parameter:

```
GET https://{{sspHost}}/{{apiPathTenant}}/admin/v1/UsersRefreshTokens/{{REPLACE-BY-USERID-OR-USERLOGINID}}
```

This endpoint response returns the PRTs.

Revoke PRT of an User

To revoke the PRT of an user, invoke the following endpoint with "**showpersonaltokens=true**" query string parameter:

```
DELETE https://{{sspHost}}/{{apiPathTenant}}/admin/v1/UsersRefreshTokens/{{REPLACE-BY-USERID-OR-USERLOGINID}}
```

This endpoint response returns 204.

Custom Token Claims

Learn about Symantec Identity Security Platform Custom Token Claims

Custom Token Claims

IDSP provides the ability to define custom claims for a SAML/OIDC application using the "claims" property (for OIDC App) and "samlAssertionClaims" (for SAML app) on ".../auth/v1/authenticate" resource. These custom claims are included in the Identity and/or Access tokens that are generated by IDSP on behalf of an application as a relying party.

"claims" Property Syntax (for OIDC app):

```
"claims": [
            {
              "name": "<claim-name>",
              "target": "<comma-separated-token-types>",
              "source": "${<namespace>.<property>}"
            }
          ]
```

"samlAssertionClaims" Property Syntax (for SAML app):

```
"samlAssertionClaims": [
            {
              "name": "<claim-name>",
              "source": "${<namespace>.<property>}"
            }
          ]
```

- **name:** Specifies the name of the custom claim to be included in the token. As a good practice, prefix the custom claim name with a special character such as "_".
- **target:** Specifies the target token type for the custom claim. The input values for the token types are "it" for Identity Token and "at" for Access Token. This property is applicable for OIDC apps only.
- **source:** The supported namespaces and their flows for the custom claims are as follows:
  - **idProfile:** The idProfile namespace represents user profile attributes and it is available in both non-ZFP and ZFP flows.
    - **Non-ZFP Flow:** In a non-ZFP flow, idProfile includes all logical user attributes as configured in an ID store object for LDAP/SCIM ID Store.
    - **ZFP Flow:** In a ZFP flow, the idProfile includes user attributes that are a subset of claims as present in the incoming Identity Token.

      The list includes the following claims, subject to being present in the token:
      - **OpenID Claims:** NAME, GIVEN_NAME, MIDDLE_NAME, FAMILY_NAME, NICK_NAME, PREFERRED_USERNAME, PROFILE, PICTURE, WEBSITE, GENDER, BIRTHDATE, ZONEINFO, LOCALE, UPDATED_AT, EMAIL, EMAIL_VERIFIED, ADDRESS, PHONE_NUMBER, PHONE_NUMBER_VERIFIED
      - **Custom Claims:** USER_UNIVERSALID, USER_LOGINID, USER_GUID, USER_DN, IDP_NAME, IDP_TYPE, IDP_GUID, IDP_USER_GUID
      - **Standard Claims:** SUB
  - **idToken:** The idToken namespace is available in every flow where a request for a token has been made with an Identity Token (or Identity Token Hint), depending on the request. This namespace exposes the entire claim set present in the token, for top-level claims only.
  - **appMetadata:** The appMetadata namespace represents application metadata attributes, and it is available in every flow.
  - **clientContext:** The clientContext namespace represents the custom context passed in using one of the following flows:
    - **Request Payload in Authenticate API**: The clientContext property can be passed into the ".../auth/v1/authenticate" call in the request payload. For example -

```
{
        "subject": "nbruce",
        "channel": "web",
        "action": "authenticate",
        "rememberMe": true,
        "urn:iam:authn:clientContext": [
          {
            "key": "email",
            "value": "user1@example.com"
          }
        ]


    }
```

    - **As a claim in the "id_token_hint" Parameter**: The clientContext property can be passed as a query parameter to ".../oauth2/v1/authorize" or can be passed into ".../auth/v1/authenticate" API call through the "existingIDToken" or "id_token_hint" payload parameters. The following example represents the decoded id_token_hint with clientContext claim:

```
{
 "sub": "dd433ba1-b9ed-46bf-9d3e-15759d6bf89b",
 "idp_name": "idp_aa",
 "user_dn": "cn=user1,ou=Employees,o=democorp,c=us",
 "amr": [
   "PWD"
```

```
  ],
  "user_loginid": "user1",
  "idp_guid": "bcd084d4-6eb0-48bb-8d8b-25c19af9ec2e",
  "iss": "https://www.example.com/default/",
  "user_guid": "dd433ba1-b9ed-46bf-9d3e-15759d6bf89b",
  "aal": "AAL1",
  "tid": "3239b450-8c89-446f-81c3-70890815d2c9",
  "transactionid": "dea78ba8-63a8-4454-8745-007b53dd0bd9",
  "sid": "b8695bf1-ebd5-4c2f-807b-ac27bdd0e578",
  "azp": "72316d69-9749-4ef7-a9bb-d92e433efe28",
  "user_loginidattrmappingname": "user_loginid",
  "auth_time": 1711336816,
  "session_ttl_secs": 21600,
  "exp": 1811358416,
  "iat": 1711336816,
  "client_name": "app1",
  "jti": "c9fd1cb1-e5c3-4cd8-99ff-b66592799610",
  "email": "user1.1@example.com",
  "idp_type": "siteminder",
  "client_tname": "default",
  "client_tid": "3239b450-8c89-446f-81c3-70890815d2c9",
  "ver": "1.0",
  "user_universalid": "user1",
  "user_riskscore": 100,
  "auto_inline_enrollment": true,
  "aud": [
    "ceee9ad0-9414-422d-a264-179a85cbf4db",
    "ad179147-fce0-4e09-a1cf-7830cfe3b4a3",
    "72316d69-9749-4ef7-a9bb-d92e433efe28"
  ],
  "nbf": 1711336816,
  "urn:iam:authn:clientContext": [
    {
      "key": "email",
      "value": "user1@example.com"
    }
  ],
  "urn:iam:authn:userData": [
    {
      "key": "KeyID",
      "value": [
        "500076885"
      ]
    }
  ]
}
```


- **As a "`urn:iam:authn:clientContext`" Query Parameter**: The clientContext property can be passed into the following endpoints:
  - ".../oauth2/v1/authorize"
  - ".../oauth2/v1/token"
  - ".../saml/v1/idp/login"
  - ".../launch/v1/Apps/{appid}"

The clientContext is passed as a URL-encoded value. Optionally, it can be encrypted before encoding using one of the following methods:

- {x5c}: Encrypted using the tenant's public key. The encrypted value is obtained by calling ".../admin/v1/ClientNonceGenerator(value)".
- {secret-alias>}: AES-based data encryption using a customer-supplied secret identified by <secret-alias>.
  - Key: Must be a multiple of 16 bytes in length (for example, 16, 32, or 48 bytes).
  - IV (Initialization Vector): Derived from the first 16 bytes of the key, starting at offset 0.

The following example represents the url-encoded form of {x5c}, indicating that the subsequent value is encrypted using the {x5c} format:

```
https://{{sspHost}}/{{apiPathTenant}}/oauth2/v1/token?urn:iam:authn:clientContext:encOrgToken=
%7Bx5c%7DYzRTM6jAilpQmm1CREo8f%2FBKPxYswfRTqsO796bfkmSmlpvwCVA1ylVinZdU
%2BbhrIhriE6bZkON%2Bb9CF3If9lLc7L3BovLsqSqfKASzskiWBqZU9m38AKMPmyhw%2BKxuGEfIGKX
%2FL5d4FEz2yMnDiCQFiCAaczV1%2FaKscEiQpgL7hqBoej44Gf4wYP6uPt4O7ul3VECV4bOv9Btp7J1d0eG6tL7FmGZ6bYo7wqJFBQSqVeBji
%2BoEwz53inh%2BlwMrZGbybRoIFkAZKz4Z5DPl1VgDGgDOpaT4%2B6BqkggT4ErpAJMQA8P7lWqaVqcX65QnBJmw4ZR88P5KA%3D
%3D%0A
```

This table maps various authentication and authorization flows to the types of data (user profile, application metadata, and dynamic client context) that are available and can be used to populate **custom claims** within issued Access Tokens and ID Tokens. It also specifies the conditions under which this data is included or utilized in each flow.

Flow			
	idProfile	idToken	appMetadata
/authenticate REST API (IT)	yes	yes (if Identity Token Hint is passed into /authenticate)	yes
OAuth Client Credentials (AT)	n/a	n/a	yes
OAuth AZ Code (IT, AT)	yes	yes (if Identity Token Hint is passed into /authorize)	yes
OAuth Device Code (IT, AT)	yes	n/a	yes
OAuth Refresh Token (AT)	yes	n/a	yes
OAuth JWT-BEARER (AT)	yes	yes (if Identity Token Hint is passed into JWT-BEARER)	yes
OAuth Implicit (IT, AT)	yes	yes (if Identity Token Hint is passed into /authorize)	yes

Clients requesting admin-approved user profile attributes

Clients, such as applications, can request administratively permitted user profile attributes (or "scopes") in their tokens using custom token claims.

Administrators - Administrators set up the ability for clients to request specific user profile attributes through the definition of a custom claim.

The Application administrator defines a custom claim with the name `urn:iam:idprofile:requested` . The value of this custom claim can be any character except for a blank value. This custom claim indicates that the client (app/service) is allowed to request specific attributes (user profile scopes) and have them returned in the tokens.

Client Request: Clients use predefined syntax or fields to request the user profile attributes, which are included as custom claims in the tokens issued by the IDSP.

- **OIDC Client** - For OpenID Connect (OIDC) clients (apps), they request specific user profile attributes by specifying the following syntax: "`urn:iam:idprofile:<id-profile-attribute>`" . Here, id-profile-attribute is the actual profile attribute (scope) the client wants.
- **Programmatic Client (API)** -
  - For clients that authenticate programmatically (For example, API clients using /authenticate), the request is made differently.
  - The client uses the request's authContext parameter to pass the requested attributes. The field's key is `requestIdProfileAttributes` , and the value is a comma-separated list (CSV) of logical ID profile attributes that the client wants to receive.
- **Token Issuance** -
  - The IDSP (authentication system) issues tokens—whether they are ID tokens (IT) or Access tokens (AT)—that contain the requested identity profile attributes.
  - These tokens include custom claims that represent the client-requested attributes, allowing the client to receive and use this information securely.

Example of custom claims in SCIM Config Resource

The following example demonstrates the use of two custom claims.

- "branchName" (IT/AT) whose data source is represented by the "branchName" ID Store attribute. Notice the "branchName" logical attribute being part of the attribute mapping in the SCIM ID Store configuration.
- "productEntitlement" (AT) whose data source is represented by the "productEntitlement" claim in the Identity Token (when such IT is used to generate AT)


```
{
   "configName": "TestConfig",
   "configDescription": "",
   "securityMethod": "oauthassertion",
   "clientId": "70d8bd85645c428ea4704a8e1823c5e3",
   "clientSecretAlias": "client_assertion_signer",
   "oauthScopes": "urn:opc:idm:__myscopes__",
   "oauthTokenUrl": "https://example.com/oauth2/v1/token",
   "oauthDiscoveryUrl": "",
   "oauthResourceInAzRequest": null,
   "oauthAssertionKidType": "x5t",
   "oauthAssertionAud": "https://example.com/",
   "scimURL": "https://example.com/admin/v1",
   "certificateAlias": "Cert",
   "ignoreSslValidation": false,
   "sslMode": "verify_ca",
   "userLoginIdAttributeMappingName": "user_loginid",
   "isAuthoritativeForIDLCM": true,
   "isAuthoritativeForUserCredsLCM": false,
   "statusEncoding": "SCIM",
   "passwordResetURL": null,
   "lookupQuery": "userName Eq \"${loginid}\"",
   "useUserLoginIdAsScimId": false,
   "endpointSuffixOverride": {},
   "ctxMapBase64": "",
   "readTimeoutMillis": "10000",
   "connectTimeoutMillis": "10000",
```

```
  "attributeMapping": "title=title, user_loginid=userName, family_name=name.familyName,
given_name=name.givenName, phone_number=phoneNumbers[work], email=emails[primary], user_universalid=userName,
memberOf=groups.display",
  "oauthTokenIgnoreSslValidation": false,
  "oauthTokenSslMode": "verify_ca",
  "oauthTokenCertificateAlias": "Cert",
  "oauthTokenCtxMapBase64": null,
  "oauthTokenReadTimeoutMillis": "10000",
  "oauthTokenConnectTimeoutMillis": "10000",
  "createdDateTime": "2021-10-26 10:46 AM UTC",
  "updatedDateTime": "2021-10-26 22:03 PM UTC",
  "createdBy": {
      "principalType": "CLIENT",
      "principalId": "c96f6992-5385-423c-82eb-d2173fd39c42",
      "principalName": "defaulttenantclient"
  },
  "updatedBy": {
      "principalType": "CLIENT",
      "principalId": "c96f6992-5385-423c-82eb-d2173fd39c42",
      "principalName": "defaulttenantclient"
  },
  "scimConfigId": "bba696c9-92ec-4c5d-a6a4-1b2494f4e3fd"
        }
```


Example of custom claims in Application Resource

The following example shows the "claims" property as part of the demo app configuration:


```
{
   "status": "active",
   "name": "democlient",
   "description": "Demo client to perform tenant level operations for demo purpose.",
   "clientType": "TRUSTED",
   "allowedOpenIDScopes": [
       "address",
       "phone",
       "openid",
       "profile",
       "groups",
       "email"
   ],
   "redirectURIs": [
       "https://www.example.com"
   ],
   "allowedGrantTypes": [
       "refresh_token",
       "implicit",
       "client_credentials",
       "authorization_code",
       "urn:ietf:params:oauth:grant-type:jwt-bearer"
   ],
   "flowURL": null,
   "deviceCodeFlowURL": null,
   "userInfoEndpointResponseFormat": "PLAIN_JSON",
```

```
    "skipIssuerAudienceForIT": false,
    "skipEmailForIT": false,
    "zeroFootPrint": false,
    "softMFAEnabled": false,
    "userTokenSubAttributeMappingName": null,
    "supportedJoseHeaderParams": null,
    "claims": [
        {
            "name": "_title",
            "target": "at,it",
            "source": "${idProfile.title}"
        },
        {
            "name": "_idp_name",
            "target": "at,it",
            "source": "${idToken.idp_name}"
        },
        {
            "name": "_env",
            "target": "at,it",
            "source": "${appMetadata}.env}"
        }
        {
            "name": "_acct_num",
            "target": "at,it",
            "source": "12345"
        }
    ],
    "allowedOperations": [
        "itgroups",
        "atgroups",
        "introspect"
    ],
    "secondaryAudiences": [
        "env=blue",
        "https://example.com/myapi"
    ],
    "assertionVerificationCertAlias": null,
    "appIcon": null,
    "secret": "b62989d7-b992-4a77-88c8-5f6595eeedf3",
    "clientId": "b72ca551-59ec-4e07-ac5b-d09f19c663a8",
    "itEncryptionTarget": "NONE",
    "itEncryptionCertAlias": null,
    "userInfoEncryptionCertAlias": null,
    "appId": "4c19e3a6-e793-40b5-9c07-2f687d8ccc50",
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "c96f6992-5385-423c-82eb-d2173fd39c42",
        "principalName": "defaulttenantclient"
    },
    "createdDateTime": "2021-06-30 10:29 PM UTC",
    "updatedDateTime": "2021-10-08 01:40 PM UTC"
}
```

Example of custom claims in Identity Token

The following example depicts the "claims" property as part of the demo app configuration.


```
{
 "at_hash": "zjrD0VV0LnDbCIAmr5gwkg",
 "sub": "00ec5422-6625-4205-bbeb-21dcc9f9b4d2",
 "idp_name": "TestConfig",
 "amr": [
   "PWD"
 ],
 "user_loginid": "nbruce",
 "idp_guid": "bba696c9-92ec-4c5d-a6a4-1b2494f4e3fd",
 "iss": "https://example.com/default/",
 "user_guid": "00ec5422-6625-4205-bbeb-21dcc9f9b4d2",
 "aal": "AAL1",
 "tid": "d250d650-46e0-4519-8c89-6d4cec63bb05",
 "sid": "463930ad-45cd-4c5a-a1cb-22258e79f48b",
 "acr": "urn:iam:acr:aal1",
 "azp": "b72ca551-59ec-4e07-ac5b-d09f19c663a8",
 "user_loginidattrmappingname": "user_loginid",
 "auth_time": 1635874080,
 "user_url": "https://example.com/default/idstore/v1/Users/nbruce",
 "exp": 1635895707,
 "iat": 1635874107,
 "client_name": "democlient",
 "jti": "e04832ce-ff88-4c61-a555-0e6b3145d2d9",
 "email": "nbruce@gmail.com",
 "idp_type": "ldap",
 "client_tname": "default",
 "client_tid": "d250d650-46e0-4519-8c89-6d4cec63bb05",
 "ver": "1.0",
 "user_universalid": "nbruce",
 "__branchName": "LXM",
 "tname": "default",
 "tok_type": "IT",
 "user_riskscore": 100,
 "aud": [
   "env=blue",
   "https://example.com/myapi",
   "b72ca551-59ec-4e07-ac5b-d09f19c663a8",
   "https://example.com/default/"
 ],
 "nbf": 1635874107,
 "client_transactionid": "ssp:fbc14efa-a856-4ad6-ad05-0128026f9d7f"
}
```


Example of custom claims in Access Token

The following example depicts the "branchName" and "productEntitlement" custom claims in the Access Token:


```
{
 "sub": "Kevin.LUCAS@DEMOCORP.com",
 "idp_name": "TestConfig",
 "user_loginid": "kevinlucas",
```

```
 "idp_guid": "69f47924-a0f0-490a-b011-92078c0b3b36",
 "iss": "https://example.com/default/",
 "user_guid": "2b379d4f-6ab3-4722-b34b-b8fce6f06de0",
 "tid": "d250d650-46e0-4519-8c89-6d4cec63bb05",
 "sid": "94a154f3-9ff2-41ab-9ae4-4ecbfdb0f482",
 "user_subattrmappingname": "email",
 "sub_type": "USER",
 "azp": "b72ca551-59ec-4e07-ac5b-d09f19c663a8",
 "user_loginidattrmappingname": "user_loginid",
 "scope": "urn:iam:m.me openid",
 "exp": 1645834791,
 "iat": 1645831191,
 "client_name": "democlient",
 "jti": "4fed91f9-f56d-45c9-b7c1-c5331c79bede",
 "idp_type": "ldap",
 "client_tname": "default",
 "client_tid": "d250d650-46e0-4519-8c89-6d4cec63bb05",
 "ver": "1.0",
 "user_universalid": "kevinlucas",
 "hasgroups": true,
 "azpacr": "2",
 "mobile": "1 555 694 3977",
 "tname": "default",
 "__branchName": "LXM",
 "__productEntitlement": "6Z04",
 "groups": [
   "specialgroup",
   "newgroup"
 ],
 "tok_type": "AT",
 "user_riskscore": 100,
 "user_amr": [
   "PWD"
 ],
 "aud": "https://example.com/default/",
 "nbf": 1645831191,
 "user_zfp": false,
 "client_transactionid": "ssp:530c3d0c-e796-47b7-8b2a-eec4dea7d715",
 "user_aal": "AAL1",
 "tok_source": "AZC",
 "user_acr": "urn:iam:acr:aal1"
}
```


Example of custom claims in "/Authenticate API" and "Identity Token Hint" (ITH)

This example demonstrates the use of clientContext in two modes:

- programmatic authentication (/authenticate accepting clientContext)
- Identity Token Hint that could be passed into OpenID Connect's /authorize as id_token_hint parameter

```
https://${SSP_FQDN}/{{apiPathTenant}}/auth/v1/authenticate
{
   "subject": "mhunter",
   "action": "authenticate",
```

```
   "ipAddress": "1.1.1.2",
   "channel": "web",
   "acrValues" : ["urn:iam:acr:aal:aal1"],
   "existingIDToken": "<ith>",
   "clientContext": [
       {
           "key": "clientCustomData2",
           "value": "gga323a24"
       }
   ],
   "authContext": [
       {
           "key": "useTenantKeyForITEncryption",
           "value": "false"
       },
       {
           "key": "requestGroupsInIT",
           "value": "true"
       },
       {
           "key": "requestClientContextInIT",
           "value": "true"
       },
       {
           "key": "requestUserDataInIT",
           "value": "true"
       }
   ]
}
```

Identity Token Hint example

```
{
 "sub": "malbail",
 "auto_inline_enrollment": true,
 "idp_name": "ldap_ud",
 "idp_type": "siteminder",
 "user_universalid": "MalBAIL",
 "amr": [
   "PWD"
 ],
 "user_loginid": "malbail",
 "user_dn": "cn=malbail,dc=ca,dc=com",
 "iss": "https://example.com/xyz/siteminder",
 "given_name": "Mal",
 "middle_name": "",
 "family_name": "Bail",
 "nonce": "1f82c059-661a86d2-9028adcb-0fe29cbb-db3c6bea-383",
 "aud": "8324d892-f2d1-4f1a-95ab-fa5c398f4ed6",
 "nbf": 1622039058,
 "auth_time": 1622039058,
 "exp": 1622039358,
```

```
"iat": 1622039058,

"phone_number": "3344444",
"email": "malbal@test.com",

"idp_id": "AD_NAME",
"idp_user_guid": "user1024_idunique",

"urn:iam:authn:clientContext": [
  {
    "value": "567890",
    "key": "TxnId"
  },
  {
     "key": "clientCustomData2",
     "value": "gga323a24"
  }
],

"urn:iam:authn:userData": [
  {
    "key": "email",
    "value": [
      "malbal@test.com",
      "test@ca.com"
    ]
  },
  {
    "key": "phone_number",
    "value": [
      "3344444",
      "3344445"
    ]
  }
]
```


Revoke Token

Learn about Symantec Identity Security Platform Token Lifecycle Management and how Revoke Tokens work.

This topic includes the following sections:

- Token Lifecycle Management
- Revoke Token

Token Lifecycle Management

Revocation is an important aspect of Token Lifecycle Management. When a token is revoked, the authorization server is intimated that the token is no longer needed. This helps the authorization server to clean up any security credentials that are associated with the token and also prevents unknown users from reusing the token.

Token Validation Semantics

The following token validation semantics is used to revoke a token in IDSP:

- Check for Token Validity (TV)
  - Signature
  - Expiry
- Check for Token Scope (TS)
  - Audience
  - Different validation semantics per token type
- Check for Token Revocation (TR)
  - Revocation Status (AT/IT)
    - valid jti claim
  - Session status (AT/IT)
    - valid sid claim
  - Presence in DB (RT/AZCode)

The following table depicts the token validation semantics (TV/TS/TR) for different kinds of tokens:

Token Type	Supported Behavior				
	Revoking Tokens	Generating Tokens			
	/revoke	/introspect	/userinfo	/token JWT-Bearer grant	/toke Cod
Access Token (AT)	revokes at.jti Do not revoke at.sid if present	y TV TS is N/A TR (check for non revoked jti, and sid if present)	y TV TS (oidc scope as per allowedOidcScopes) TR on AT (check for non revoked jti, and sid if present)	TS/TV n/a add sid (guid) to RT at create time when offline_access is requested TR on IT (check for non revoked jti, and sid if present)	TS/T add (guic at cr time offlin is re TR c Cod
Identity Token (IT)	Revokes it.jti Revokes it.sid	y TV TS is N/A TR (check for non revoked jti and sid)	n/a	y TV TS (issuer) TR (check for non revoked jti and sid)	n/a
Refresh Token (RT)	deletes RT from DB Revokes rt.sid	y TV TS is N/A TR	n/a	n/a	n/a
AZ Code	n/a	n/a	n/a	n/a	y TV TS (TR

Revoke Token

Revoke Token is an authorization endpoint added to the OAuth Authorization server which lets the client indicate to the authorization server that a previously generated access token or refresh token is no longer needed. Therefore, the

token allows the authorization server to invalidate the token and clean up any security credentials associated with the authorization.

NOTE

- The revocation of the encrypted ID Token is not yet supported.
- For admin-based Token Revocation API, see Managing Token Revocation.

In order to request for revoke token in the APIs, you must first configure ***"trackAccessToken"*** and ***"trackIdentityToken"*** settings in IDSP. See Managing Settings for these setting details. The Revoke Token can be passed either in the authorization header or POST body. The following table shows the parameters that should be passed in the revoke request:

Name	Description
token	Specifies the Access Token or Refresh Token that must be revoked.
token_type_hint	Specifies the type of the token that must be revoked. Supported tokens are: **access_token, id_token** or **refresh_token**
Supported HTTP Methods	POST method

Example displaying revocation of Refresh Token

Example: Displays the revocation of a refresh token through the Revoke Token endpoint:

```
curl --location --request POST 'https://example.com/default/oauth2/v1/revoke' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'x-tenant-name: default' \
--header 'Authorization: Basic
 PHBhc3RlIGNsaWVudElkIGZyb20gYWJvdmUgY29tbWFuZD46PHBhc3RlIGNsaWVudFNlY3JldCBmcm9tIGFib3ZlIGNvbW1hbmQ+' \
--data-urlencode 'token=ZMgip58RpaWEVB5B9sCUjHrRkxw1xlPoYbLc' \
--data-urlencode 'token_type_hint=refresh_token'
```

Example of Revoke Token response:

```
HTTP 200 OK
```

Example of Revoke Token error response:

In the Tenant /Golbal Settings, when trackAccessToken or trackIdentityToken is set to false and you try to revoke an access token or identity token, then
the Authorization server responds with a 400 error code with the error message below.

```
{
error: “unsupported_token_type”,
error_description:”The authorization server does not support revocation of the presented token type”
}
```

Example of Revoke Token with Client Assertion as Client Credentials

The following sample shows a request payload for "/revoke" with client assertion as client credentials:

```
curl -k --request POST \
--url 'https://example.com/default/oauth2/v1/revoke' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'x-tenant-name: default' \
--data-urlencode 'token=eyJ4NXQjUzI1NiI6ImNJbUNYN1FyRy1BUUpkWWZ......'
```

```
--data-urlencode 'client_id=7b447d36-cac4-4a9e-b093-42f0d553d9f4' \
--data-urlencode 'client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer' \
--data-urlencode 'client_assertion=ewogICJ4NXQjUzI1NiI6ICI4bkRSWURaUF9xcENwUz.....'
```

Using Tokens to Drive Generic Authorization

This content gives a comprehensive overview of different token types used in Symantec Identity Security Platform and how they can be used for generic authorization.

This topic includes the following sections:

- Using Tokens to Drive Generic Authorization
- Use Case 1- Acquiring Identity Token
- Use Case 2 - Acquiring Access Token
- Use Case 3 - Acquire Access Token with OpenID privileges to access authorized User Data

The following use cases are applicable to leverage basic tokenization constructs to drive entitlement (authorization) capabilities for an application with the following sample profile, developed to work with identities as shown below by a sample user identity profile in a sample store: The following demo client sample profile shows the added groups in "allowedOpenIDScopes" and the "itgroups" and "atgroups" in "allowedOperations".

> **NOTE**
> The maxGroupsInJWT configuration controls the maximum number of groups that can be returned in tokens. If the number of groups to which a user belongs exceeds the maxGroupInJWT configuration, then the token returns an empty “groups” claim along with the claim "hasgroups": true. This indicates that groups cannot be injected into the token even though the user belongs to those groups.

"democlient" Application Profile

```
{
     "status": "active",
     "name": "democlient",
     "description": "Demo client to perform tenant level operations for demo purpose.",
     "clientType": "TRUSTED",
     "allowedOpenIDScopes": [
         "openid",
         "profile",
         "address",
         "groups",
         "email",
         "phone"

     ],
     "redirectURIs": [
         "https://www.example.com"
     ],
     "allowedGrantTypes": [
         "client_credentials",
         "authorization_code",
         "urn:ietf:params:oauth:grant-type:jwt-bearer"
     ],
     "userInfoEndpointResponseFormat": "PLAIN_JSON",
     "skipIssuerAudienceForIT": false,
     "skipEmailForIT": false,
     "zeroFootPrint": false,
```

```
    "softMFAEnabled": false,
    "allowedOperations": [
        "introspect",
        "itgroups",
        "atgroups"
    ],
    "secondaryAudiences": [
        "https://www.example.com/api",
        "tag/env=blue"
    ],
    "secret": "efadbe82-a78b-488a-b3cd-b8f4ccfdafda",
    "clientId": "281182a5-0e03-4ec3-bc5f-79c78a964911",
    "itEncryptionTarget": "NONE",
    "appId": "3fd0a9a4-4179-4eee-83fa-32cbd66a35e8",
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "bd0a5a5c-58b9-4728-8276-f813ef8a17fa",
        "principalName": "defaulttenantclient"
    },
    "createdDateTime": "2021-06-18 12:13 AM UTC",
    "updatedDateTime": "2021-06-28 09:32 AM UTC"
}
```


The following example shows the detailed User Profile of "malbail" in the ID Store, showing userAttributes and userStatus.


```
{
 "ldapConfigId": "d946e268-3ecf-4e44-8f14-fa192e8f0027",
 "configName": "TestConfig",
 "isAuthoritativeForIDLCM": "false",
 "userId": "963fdc2f-0eee-4ce2-b803-9b1bc9a66457",
 "userDN": "cn=MalBAIL,ou=Administration,ou=Corporate,o=democorp,c=us",
 "userStatus": "UNKNOWN",
 "passwordResetURL": "http://userPasswordResetURL",
 "userGroups": [
     "IT",
     "Admin"
 ],
 "userAttributes": {
     "user_universalid": "MalBAIL",
     "profile": "Regional Manager",
     "name": "MalBAIL",
     "nickname": "",
     "phone_number": "",
     "preferred_username": "",
     "given_name": "",
     "family_name": "BAIL",
     "email": ""
 }
}
```


"TestConfig" ID Store Profile


```
{
    "configName": "TestConfig",
```

```
    "configDescription": "CA DIR k8s with default data",
    "sslEnabled": "false",
    "isAuthoritativeForIDLCM": "false",
    "certificate": null,
    "certificateAlias": null,
    "statusEncoding": "",
    "ldapQueryStart": "(cn=",
    "ldapQueryEnd": ")",
    "userObjectClasses": "inetorgperson",
    "groupObjectClasses": "groupOfNames",
    "hostName": "examplecompany.example.net",
    "portNumber": "31888",
    "baseDN": "o=democorp,c=us",
    "groupBaseDN": "o=democorp,c=us",
    "bindDN": "cn=DominicMAJOR,ou=Administration,ou=Corporate,o=democorp,c=US",
    "bindPassword": "***",
    "attributeMapping":
"name=cn,family_name=sn,given_name=givenName,nickname=displayName,preferred_username=displayName,profile=title,email=m
    "groupUserMappingAttribute": "member",
    "useMemberOf": "true",
    "passwordResetURL": "http://userPasswordResetURL",
    "createdDateTime": "2021-06-23 21:53 PM UTC",
    "updatedDateTime": "2021-06-23 21:53 PM UTC",
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "bd0a5a5c-58b9-4728-8276-f813ef8a17fa",
        "principalName": "defaulttenantclient"
    },
    "ldapConfigId": "d946e268-3ecf-4e44-8f14-fa192e8f0027"
}
```


Use Case 1- Acquiring Identity Token

Identity Token can be acquired by:

- performing a fully programmatic flow (/authenticate flow)
- performing Web-based flow initiated by /authorize (3-legged OAuth / OpenID Connect)

It is possible for the resulting Identity Token to be requested and authorized to contain entitlements in the form of user groups. This is made possible by requesting special scope "**urn:iam:it:groups" - initially with the Web-flow and then also with the programmatic flow. For example:**

https://${SSP_FQDN}/{{apiPathTenant}}/oauth2/v1/authorize?scope=openid%20groups
*%20**urn:iam:it:groups**&client_id={{clientId}}&response_type=code&redirect_uri=https://www.example.com*

By requesting this scope in this flow, for ex PKCE-based AZ Code flow, the resulting Identity Token will be given a list of user groups as present in the Identity Store. For example, notice the list of user groups in this IT, and also the use of "hasgroups" to indicate the token does contain user groups. Once the Identity Token has been received by the application, access control decisions can be made using the group data.

- Authorization of the client to request such scope is controlled by the app's AllowedOperatons "itgroups"
- Groups are removed, and "hasgroups" claims are false, if the list of groups exceeds a certain size (typically to prevent Identity Token from exceeding 4 KB) even if the client is authorized to request user groups

Example of Identity Token


```
{
```

```
"at_hash": "Xh8yy6IBXIbJh_Hte5LqFQ",
"sub": "f5c6e32f-e4af-4030-bf69-26eaeb936b3f",
"idp_name": "TestConfig",
"amr": [
 "PWD"
],
"user_loginid": "nbruce",
"idp_guid": "d946e268-3ecf-4e44-8f14-fa192e8f0027",
"iss": "https://user.nbruce.id1.com/default/",
"clientTransactionId": " ",
"aal": "AAL1",
"tid": "1c2b4a58-b652-47b1-b48d-0ed441e8baea",
"sid": "f60b1f02-4e19-4291-b582-e9e034bc5e10",
"hasgroups": true,
"acr": "urn:iam:acr:aal1",
"azp": "281182a5-0e03-4ec3-bc5f-79c78a964911",
"auth_time": 1625048414,
"exp": 1625070023,
"iat": 1625048423,
"client_name": "democlient",
"jti": "5eb0bde9-7fbb-4bab-9fd8-a8d87aca2595",
"email": "nbruce@example.com",
"idp_type": "ldap",
"client_tname": "default",
"client_tid": "1c2b4a58-b652-47b1-b48d-0ed441e8baea",
"ver": "1.0",
"user_universalid": "nbruce",
"tname": "default",
"tok_type": "IT",
"user_riskscore": 100,
"aud": [
 "https://www.example.com/api",
 "tag/env=blue",
 "281182a5-0e03-4ec3-bc5f-79c78a964911",
 "https://user.nbruce.id1.com/default/"
],
"nbf": 1625048423,
"groups": [
 "Employees",
 "Admin"
]
}
```


Use Case 2 - Acquiring Access Token

The Access Token can be acquired for the purpose of redeeming such token against IDSP or against a 3rd party Resource Server. While Admin-specific permissions are beyond the scope of this overview, of special interest is acquiring an Access Token against 3rd party Resource Server or with which to request user data or entitlements via /userinfo. Acquiring an Access Token for user data/entitlements can be achieved by:

- requesting Access Token to be returned together with the Identity Token by /authorize. This works well when the Access Token must be used right away. When such an Access Token must be refreshed during the user's session, another /authorize call is made to seamlessly refresh the Access Token
- exchanging Identity Token for the Access Token using /token with JWT-BEARER grant, with an optional refresh token after which the Access Token can continue to get refreshed without the user needing to be "in session".

The Access Token can be requested to contain user groups. By requesting this scope in any flow using the "urn:iam:at:groups" claim, the resulting Access Token is given a list of user groups as present in the Identity Store. For example, notice the list of user groups in this AT, and also the use of "hasgroups" to indicate the token does contain user's groups. Once the Access Token has been received by the application, access control decisions can be made using the group data.

- Authorization of the client to request such scope is controlled by the app's AllowedOperations "atgroups"
- Groups are removed, and "hasgroups" claims are false, if the list of groups exceeds a certain size (typically to prevent Access Token from exceeding 4 KB) even if the client is authorized to request user groups

Example of Resulting Access Token


```
{
"sub": "f5c6e32f-e4af-4030-bf69-26eaeb936b3f",
"idp_name": "TestConfig",
"user_loginid": "nbruce",
"idp_guid": "d946e268-3ecf-4e44-8f14-fa192e8f0027",
"iss": "https://user.nbruce.id1.com/default/",
"clientTransactionId": " ",
"tid": "1c2b4a58-b652-47b1-b48d-0ed441e8baea",
"sid": "e8369b97-6064-4b58-9864-c9ed061c7e4c",
"hasgroups": true,
"sub_type": "USER",
"azp": "281182a5-0e03-4ec3-bc5f-79c78a964911",
"scope": "urn:iam:m.me openid groups",
"exp": 1625052023,
"iat": 1625048423,
"client_name": "democlient",
"jti": "c65f9ac9-98c1-4a8f-a75c-9b5d9973f165",
"idp_type": "ldap",
"client_tname": "default",
"client_tid": "1c2b4a58-b652-47b1-b48d-0ed441e8baea",
"ver": "1.0",
"user_universalid": "nbruce",
"azpacr": "2",
"tname": "default",
"tok_type": "AT",
"user_riskscore": 100,
"user_amr": [
 "PWD"
],
"aud": "https://user.nbruce.id1.com/default/",
"nbf": 1625048423,
"user_aal": "AAL1",
"groups": [
 "Employees",
 "Admin"
],
"user_acr": "urn:iam:acr:aal1"
```

```
}
```

The audience claim "aud" of the Access Token is critical since it determines where the Access Token may be redeemed. When exchanging Access Token with 3rd party resource servers with whom the app is designed to work, the IDSP can be configured, and the client can request that the Access Token be given the audience associated with 3rd party resource servers that the client is administratively configured to invoke APIs, where the Access Token plays the role of the authorization credential. While this is not a full-blown 3rd party Resource Server support, this feature can be used to manage the trust scope of the Access Token requested by the client. For example, notice the audience claim of the resulting Access Token where the client requests that only application-specific (aka secondary) audiences be used in the resulting Access Token. Correlate this audience with the "secondaryAudiences" property of the application client given in the previous example. Also, note that such "3rd party" Access Token too can have user group information encoded in it. It is the responsibility of the 3rd party resource server (or its Policy Enforcement Point) to enforce such audience restrictions.

```
https://${SSP_FQDN}/{{apiPathTenant}}/oauth2/v1/authorize?scope=openid%20urn:iam:at:appaud
%20urn:iam:at:groups&client_id={{clientId}}&response_type=code&redirect_uri=https://www.example.com
```

Example of resulting Access Token with Custom Audience

```
{
"sub": "f5c6e32f-e4af-4030-bf69-26eaeb936b3f",
"idp_name": "TestConfig",
"user_loginid": "nbruce",
"idp_guid": "d946e268-3ecf-4e44-8f14-fa192e8f0027",
"iss": "https://user.nbruce.id1.com/default/",
"clientTransactionId": " ",
"tid": "1c2b4a58-b652-47b1-b48d-0ed441e8baea",
"sid": "c612812b-3126-4158-829d-1267ef35b983",
"hasgroups": true,
"sub_type": "USER",
"azp": "281182a5-0e03-4ec3-bc5f-79c78a964911",
"scope": "urn:iam:at:appaud",
"exp": 1625056262,
"iat": 1625052662,
"client_name": "democlient",
"jti": "4ff75835-d95e-431d-a068-907a159b5b35",
"idp_type": "ldap",
"client_tname": "default",
"client_tid": "1c2b4a58-b652-47b1-b48d-0ed441e8baea",
"ver": "1.0",
"user_universalid": "nbruce",
"azpacr": "2",
"tname": "default",
"tok_type": "AT",
"user_riskscore": 100,
"user_amr": [
  "PWD"
],
"aud": [
  "https://www.example.com/api",
  "tag/env=blue"
],
"nbf": 1625052662,
"user_aal": "AAL1",
"groups": [
```

```
  "Employees",
  "Admin"
],
"user_acr": "urn:iam:acr:aal1"
}
```

Use Case 3 - Acquire Access Token with OpenID privileges to access authorized User Data

The Access Token with which to access authorized user data via /userinfo can be acquired by requesting openid + other OpenID Connect and Custom scopes. A client is authorized for these operations by having properly configured "allowedOpenIDScopes" property as shown above.

- openid - request OpenID Connect authentication flow and return user identity in the Identity Token's "sub" claim
- email - allow access to user's email and verified_email data
- profile - allow access to user's profile attributes
- phone - allow access to user's phone and verified_phone data
- address - allow access to user's address
- groups - allow access to user's id store group data

For example, requesting these scopes via AZ Code flow allows the application to acquire the Identity Token (with user groups), and also Access Token with which it can invoke /userinfo to get user data to get user groups (or just get user groups via Access Token), while also requesting access to user's profile, address, email, or phone data.

```
https://${SSP_FQDN}/{{apiPathTenant}}/oauth2/v1/authorize?scope=openid%20email%20phone%20profile
%20groups%20urn:iam:it:groups
%20urn:iam:at:groups&client_id={{clientId}}&response_type=code&redirect_uri=https://www.example.com
```

Example of resulting Access Token

```
{
"at_hash": "ox-D-r_jjYYUE0QvL6T39A",
"sub": "f5c6e32f-e4af-4030-bf69-26eaeb936b3f",
"idp_name": "TestConfig",
"amr": [
 "PWD"
],
"user_loginid": "nbruce",
"idp_guid": "d946e268-3ecf-4e44-8f14-fa192e8f0027",
"iss": "https://user.nbruce.id1.com/default/",
"clientTransactionId": " ",
"aal": "AAL1",
"tid": "1c2b4a58-b652-47b1-b48d-0ed441e8baea",
"sid": "2d47b9e9-848f-40ec-b7b0-e1b96e39dc6e",
"hasgroups": true,
"acr": "urn:iam:acr:aal1",
"azp": "281182a5-0e03-4ec3-bc5f-79c78a964911",
"auth_time": 1625053809,
"exp": 1625075416,
"iat": 1625053816,
"client_name": "democlient",
"jti": "880a54d8-991a-4fe5-aaa4-ee492e68049d",
"email": "nbruce@example.com",
"idp_type": "ldap",
"client_tname": "default",
"client_tid": "1c2b4a58-b652-47b1-b48d-0ed441e8baea",
"ver": "1.0",
```

```
"user_universalid": "nbruce",
"tname": "default",
"tok_type": "IT",
"user_riskscore": 100,
"aud": [
 "https://www.example.com/api",
 "tag/env=blue",
 "281182a5-0e03-4ec3-bc5f-79c78a964911",
 "https://user.nbruce.id1.com/default/"
],
"nbf": 1625053816,
"groups": [
 "Employees",
 "Admin"
]
}
```


Example of resulting Access Token


```
{
"sub": "f5c6e32f-e4af-4030-bf69-26eaeb936b3f",
"idp_name": "TestConfig",
"user_loginid": "nbruce",
"idp_guid": "d946e268-3ecf-4e44-8f14-fa192e8f0027",
"iss": "https://user.nbruce.id1.com/default/",
"clientTransactionId": " ",
"tid": "1c2b4a58-b652-47b1-b48d-0ed441e8baea",
"sid": "c4368c6a-9a96-4d2a-97e4-d166dea5b41e",
"sub_type": "USER",
"azp": "281182a5-0e03-4ec3-bc5f-79c78a964911",
"scope": "urn:iam:m.me phone openid profile groups email",
"exp": 1625057416,
"iat": 1625053816,
"client_name": "democlient",
"jti": "a302ebed-5810-487b-b8a8-23110ebfb24d",
"idp_type": "ldap",
"client_tname": "default",
"client_tid": "1c2b4a58-b652-47b1-b48d-0ed441e8baea",
"ver": "1.0",
"user_universalid": "nbruce",
"azpacr": "2",
"tname": "default",
"tok_type": "AT",
"user_riskscore": 100,
"user_amr": [
 "PWD"
],
"aud": "https://user.nbruce.id1.com/default/",
"nbf": 1625053816,
"user_aal": "AAL1",
"user_acr": "urn:iam:acr:aal1"
}
```

Invoking /userinfo with the above Access Token allows the application to acquire user data from /userinfo (in this case as JSON) on behalf of the authenticated identity.

Example of resulting User Data


```
{
 "sub": "f5c6e32f-e4af-4030-bf69-26eaeb936b3f",
 "profile": "Regional Administrator",
 "iss": "https://user.nbruce.id1.com/default/",
 "preferred_username": "",
 "given_name": "Nina",
 "aud": "281182a5-0e03-4ec3-bc5f-79c78a964911",
 "name": "nbruce",
 "nickname": "",
 "phone_number": "nbruce@example.com",
 "groups": [
     "Employees",
     "Admin"
 ],
 "family_name": "BRUCE",
 "email": "nbruce@example.com"
}
```


Client Types

Learn about the client types supported by Symantec Identity Security Platform.

IDSP supports the following Client Types:

- ***Public***
  - A Public Client is an application that initiates the authentication flow by passing only the client ID to process the authentication request. Public Client does not make use of the client secret and ignores the value, if provided. Examples of Public Client - Javascript based applications, Single Page Applications (SPA), or mobile clients.
  - The mobile apps can use a Public Client profile to initiate an authentication ceremony. This is consistent with the best practice covered by the native apps guideline as per https://tools.ietf.org/html/draft-ietf-oauth-native-apps-12
  - In IDSP, you can use the following grant types and request for specific scopes:
    - "ClientCreds" grant- With this grant type the application using the Public Client can obtain an access token with "t.authenticate" scope. This use of the Public Client is restricted to initiating authentication purpose only.
    - "JWT-BEARER" grant - With this grant type the application using the Public Client can obtain an access token with "m.me." scope. This use of the Public Client is restricted to self-service use cases, and not for performing any admin related tasks.
  - Public clients can submit their credentials using Basic Auth Authorization or POST body
- ***Confidential***
  - Clients that can be trusted to securely store their credentials such as client secrets. For example, web apps running on Web servers or App servers.

    NOTE
    A mobile app running on a user's mobile device can often be considered to be confidential as it belongs to a single user and has security controls to protect the device before the device can be used.
  - Confidential clients can submit their credentials using Basic Auth Authorization or POST body.
- ***Trusted***

Clients that are trusted to issue user assertions such as an ID Token Hint when initiating OpenID Connect interactions, or when submitting additional information about the user in certain API calls (such as when specifying a User's IP Address in user authentication API calls).

NOTE
Trusted clients are also considered confidential clients.

Refer to Managing Applications for more information on the Application resource containing the client definition.

Grant Types

Learn about the standard OAuth2 grant types supported by Symantec Identity Security Platform.

The following table shows the industry-recommended use of various OAuth grant types. The (✓) shows the recommended grand type that is suitable for the indicated apps. The grant types are linked to their respective sections, click each grant type for more information.

Grant Type	Protocols	Client Type	Single Page Apps	Native Apps	Microservices	Web App	Typical IoT Device
Client credentials	OAuth 2	public, confidential, trusted		✓ (Public Clients)	✓	✓	
Authorization code	OAuth 2.0 and OIDC	public, confidential, trusted				✓	
Authorization code With PKCE	OAuth 2.0 and OIDC	public, confidential, trusted	✓	✓			
Implicit (not recommended)	OAuth 2.0 and OIDC	public	✓				
Refresh Token	OAuth 2.0 and OIDC	public, confidential, trusted		✓	✓	✓	
JWT-Bearer	OAuth 2.0	public, confidential, trusted		✓	✓	✓	
STS-Confidential	OAuth 2.0	public, confidential, trusted					
Device Code	OAuth 2.0	public, confidential, trusted					✓

NOTE
Due to the inherent risks of returning access tokens in an HTTP redirect, it is not recommended to use the implicit flow. Public clients should be using the authorization code flow with the PKCE extension instead.

Allowed Operations

When configuring application clients, the following can be specified under *Allowed Operations* to control the client behavior:

- ***introspect*** - OAuth '**introspect'** endpoint works only when the client is allowed to perform this operation by having this configuration parameter enabled. Otherwise, the introspect endpoint returns an error response.
- ***onbehalfof*** - when the client is configured with '`onbehalfof`', the authorized scopes are computed by the authorization server on the basis of user's privileges only, instead of the intersection of user and client privileges.
- ***offlineit***: When '**offlineit**' is configured, the token endpoint returns the ID Token in the response of Refresh token and JWT-Bearer flows, if the openid scope is requested.
- ***itgroups*** - when the client is configured with '**itgroups'**, the application is allowed to include user groups in the identity token.
- ***atgroups*** - when the client is configured with '**atgroups'**, the application is allowed to include user groups in the access token.

Allowed OpenID Scopes

OpenID Connect (OIDC) scopes are a set of user attributes like email, phone that are used by an application during authentication to authorize access to the user. Each scope returns a set of user attributes, which are called claims. The scopes an application should request depend on which user attributes the application needs. Once the user authorizes the requested scopes, the claims are returned in an ID Token. IDSP supports the following OpenID scopes:

- openid - Indicates that the application intends to use OIDC to verify the user's identity.
- email - Configured email id where all the required mail notifications can be sent.
- profile - Option to personalize the email with the user's name, or picture.
- phone - Configured phone number where all the required SMS notifications can be sent.
- address - Configured postal address in the ID store.
- groups - Configured groups to which the user belongs.

Redirect URI

This is a list of allowed callback URLs where the authorization server sends the user once the app has been successfully authorized and granted an authorization code or access token. For example, https://www.example.com.

Client Credential Grant

The Client Credentials grant is used to obtain an Access Token on behalf of the client itself with no user present.

1. The client sends Client Credentials to the `/token` endpoint with `grant_type=client_credentials`, either in the Authorization header or in the POST body, depending on the client authentication method configured in the client definition
2. Symantec Identity Security Platform authenticates the client credentials and authorizes the client's requested scopes against the client's authorized privileges.
3. IDSP generates and returns the Access Token.

Request for all authorized scopes with client credentials passed in through the Authorization header:

```
curl -k --request POST \
--url 'https://example.com/default/oauth2/v1/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
-u "6cb446b9-33bf-4b97-a6e1-e7ad389974d2:2ef66db2-1e51-4868-8c72-bd57ce7f69dc" \
--data-urlencode 'grant_type=client_credentials' \
--data-urlencode 'scope=urn:iam:myscopes'
```

Response

```
{
  "access_token": "eyJ4NXQjUzI1NiI6Im......",
  "token_type": "Bearer",
  "expires_in": 3600
```

```
}

Decoded Access Token:
{
  "client_tid": "default",
  "sub": "2db6e0b1-8f69-4f1c-b308-6ddacdca76f1",
  "ver": "1.0",
  "iss": "https://example.com/default/",
  "azpacr": "1",
  "tok_type": "AT",
  "tid": "default",
  "aud": "https://example.com/default/",
  "nbf": 1592160941,
  "sub_type": "CLIENT",
  "azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
  "scope": "urn:iam:t.apps urn:iam:t.userdata urn:iam:t.ldapconfigs urn:iam:t.configs urn:iam:t.authpolicies
 urn:iam:t.certs urn:iam:t.risk urn:iam:t.authzpolicies urn:iam:t.groups urn:iam:t.users
 urn:iam:t.authenticate urn:iam:t.signingkey",
  "exp": 1592164541,
  "iat": 1592160941,
  "client_name": "testapp",
  "jti": "f41468ad-ec8d-4a9f-814d-6f5fe07377fe"
}
```

Request for specific scopes with client credentials passed in through the POST body:

```
curl -k --request POST \
--url 'https://example.com/default/oauth2/v1/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=client_credentials' \
--data-urlencode 'scope=urn:iam:t.authenticate urn:iam:t.configs' \
--data-urlencode 'client_id=6cb446b9-33bf-4b97-a6e1-e7ad389974d2' \
--data-urlencode 'client_secret=2ef66db2-1e51-4868-8c72-bd57ce7f69dc'
```

Response

```
{
  "access_token": "eyJ4NXQjUzI1NiI6ImNJ......",
  "token_type": "Bearer",
  "expires_in": 3600
}

Decoded Access Token
{
  "client_tid": "default",
  "sub": "2db6e0b1-8f69-4f1c-b308-6ddacdca76f1",
  "ver": "1.0",
  "iss": "https://example.com/default/",
  "azpacr": "1",
  "tok_type": "AT",
  "tid": "default",
  "aud": "https://example.com/default/",
  "nbf": 1592161069,
  "sub_type": "CLIENT",
  "azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
  "scope": "urn:iam:t.configs urn:iam:t.authenticate",
```

```
  "exp": 1592164669,
  "iat": 1592161069,
  "client_name": "testapp",
  "jti": "96b9c0a7-8e52-487c-b261-fc1e7f4bb7c2"
```

Request for all authorized scopes with client assertion as client credentials passed in through the Authorization header:

```
curl -k --request POST 'https://${SSP_FQDN}/default/oauth2/v1/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=client_credentials' \
--data-urlencode 'scope=urn:iam:myscopes' \
--data-urlencode 'client_id=7b447d36-cac4-4a9e-b093-42f0d553d9f4' \
--data-urlencode 'client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer' \
--data-urlencode 'client_assertion=ewogICJ4NXQjUzI1NiI6ICI4bkRSWURaU......'
```

Response

```
{
  "access_token": "eyJ4NXQjUzI1NiI6ImNJ......",
  "token_type": "Bearer",
  "expires_in": 3600
}

Decoded Access Token
{
  "client_tid": "default",
  "sub": "2db6e0b1-8f69-4f1c-b308-6ddacdca76f1",
  "ver": "1.0",
  "iss": "https://example.com/default/",
  "azpacr": "1",
  "tok_type": "AT",
  "tid": "default",
  "aud": "https://example.com/default/",
  "nbf": 1592161069,
  "sub_type": "CLIENT",
  "azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
  "scope": "urn:iam:t.configs urn:iam:t.authenticate",
  "exp": 1592164669,
  "iat": 1592161069,
  "client_name": "testapp",
  "jti": "96b9c0a7-8e52-487c-b261-fc1e7f4bb7c2"
```

Authorization Code Grant

Learn about /authorize endpoint.

The Authorization Code grant is used to obtain an Access Token (and optionally an Identity Token in the case of the OpenID Connect flow) on behalf of the user identity identified by the Authorization Code obtained by previously invoking the `/authorize` endpoint. Optionally, to obtain a Refresh Token, pass "offline_access" as one of the scopes.

NOTE

- Authorization Code grant supports encryption of Identity Token. See JSON Web Encryption (JWE).
- To exchange Identity Token for Access Token at a third party Authorization Server, additional audience values may be required to be validated by that AZ Server. See **Secondary Audience** configuration in Managing Application.

1. The client sends Client Credentials to `/token` endpoint with `grant_type=authorization_code`, either in the Authorization header or in the POST body, depending on client authentication method configured in the client definition. The POST body also includes a JWT token with an assertion parameter.

   NOTE
   When the client exchanges an Authorization Code acquired using PKCE for an Access Token, it has to includes the Code Verifier.
2. Symantec Identity Security Platform authenticates the client credentials, verifies the signature of the JWT, and authorizes the client's requested scopes against the authorized privileges.

   NOTE
   If the client is acting On-Behalf-Of, the authorization check uses (and the resulting Access Token includes) the user's privileges only. Otherwise, the authorization check uses (and the resulting Access Token includes) the intersection of the client's and the user's privileges.
3. IDSP generates and returns the Access Token.

Request with client credentials passed in through the Authorization header and Authorization Code obtained previously from an `/authorize` request for `scope=openid` (see Client Credentials grant for an example of client credentials passed in through the POST body).

NOTE
With PKCE, the value of `code_verifier` is computed and added to the request (`--data-urlencode 'code_verifier=dGVzdA=='`)

Request

```
curl --location '"https://example.com/default/oauth2/v1/token">https://example.com/default/oauth2/v1/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Authorization: Basic PHBhc3RlIGNsaWVudElkIGZyb20gYWJvdmUgY29tbWFuZD46d3Jvbmc=' \
--data-urlencode 'grant_type=authorization_code' \
--data-urlencode 'code={{code}}' \
--data-urlencode 'code_verifier=1234' \
--data-urlencode 'redirect_uri="https://example.com/sample-rp/oidc/callback">https://example.com/sample-rp/
oidc/callback' \
--data-urlencode 'scope=openid groups profile email'
```

Response

```
{
"access_token" : "eyJ4NXQjUzI1NiI6ImNJ......",
"token_type" : "Bearer",
"expires_in" : 3600,
"id_token" : "eyJ4NXQjUzI1NiI6Im....."
}

Decoded Access Token
{
"client_tid": "default",
"sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
"ver": "1.0",
"idp_name": "SecureLDAP",
"user_universalid": "KevinLUCAS",
```

```
"user_loginid": "kevinlucas",
"idp_guid": "f77d69c3-718b-4b64-be5c-767e42a1213c",
"iss": "https://example.com/default/",
"azpacr": "1",
"tok_type": "AT",
"tid": "default",
"aud": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"nbf": 1592175229,
"sub_type": "USER",
"azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"scope": "openid email",
"exp": 1592178829,
"iat": 1592175229,
"client_name": "testapp",
"jti": "ced094c6-f38f-4eae-8e03-b1b7ecbd5faf",
"idp_type": "ldap"
}

Decoded Identity Token
{
"at_hash": "b7Bbqf3vxFzzXBnV1CS9mA",
"sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
"ver": "1.0",
"idp_name": "SecureLDAP",
"user_universalid": "KevinLUCAS",
"amr": [
 "PWD"
],
"user_loginid": "kevinlucas",
"idp_guid": "f77d69c3-718b-4b64-be5c-767e42a1213c",
"iss": "https://example.com/default/",
"tok_type": "IT",
"nonce": "mynonce",
"aal": "AAL1",
"tid": "default",
"sid": "3645d037-7f12-4a47-9fc8-e8665d0d76b1",
"aud": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"acr": "urn:iam:acr:aal1",
"nbf": 1592175229,
"azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"auth_time": 1592175218,
"exp": 1592196829,
"iat": 1592175229,
"client_name": "testapp",
"jti": "b57a5295-4f36-47a8-a81e-b29d28d523cc",
"idp_type": "ldap"
}
```


To know more about the "/authorize" API endpoint, see Authorize Endpoint .

JWT Bearer Grant

Learn about JWT Bearer grant.

The JWT-Bearer grant is used to obtain the Access Token on behalf of the user identity identified by the JWT assertion. As a result, the grant performs *identity propagation*. Optionally, to obtain a Refresh Token, pass "offline_access" as one of the scopes.

NOTE

- JWT-Bearer grant supports encryption of Identity Token. See JSON Web Encryption (JWE).
- To exchange Identity Token for Access Token at a third party Authorization Server, additional audience values may be required to be validated by that AZ Server. See **Secondary Audience** configuration in Managing Application.
- Only an assertion encrypted with Tenant's key is accepted. If the assertion is encrypted with any other public key, decryption fails as Symantec Identity Security Platform is not the holder of the key.

1. The client sends the Client Credentials to `/token` endpoint either in the Authorization header or in the POST body with `grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer`, depending on the client authentication method configured in the client definition. The POST body also includes a JWT token with an `assertion` parameter.
2. IDSP authenticates the client credentials, verifies the signature of the JWT, and authorizes the client's requested scopes against the authorized privileges.

   NOTE
   If the client is acting On-Behalf-Of, the authorization check uses (and the resulting Access Token includes) the user's privileges only. Otherwise, the authorization check uses (and the resulting Access Token includes) the intersection of the client's and the user's privileges.
3. IDSP generates and returns the Access Token with the `sub` claim representing the propagated user identity.

Request for all authorized scopes with client credentials passed in via the Authorization header and user assertion representing `defaultadmin` user (see Client Credentials grant for an example of client credentials passed in through the POST body).
Example

```
curl -k --request POST \
--url 'https://example.com/default/oauth2/v1/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
-u "6cb446b9-33bf-4b97-a6e1-e7ad389974d2:2ef66db2-1e51-4868-8c72-bd57ce7f69dc" \
--data-urlencode 'grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer' \
--data-urlencode 'scope=urn:iam:myscopes' \
--data-urlencode 'assertion=eyJ4NXQjUzI1..........5y3fw'

Decoded User Assertion (Identity Token)
{
 "sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
 "ver": "1.0",
 "idp_name": "SecureLDAP",
 "user_universalid": "KevinLUCAS",
 "amr": [
   "PWD"
 ],
 "user_loginid": "kevinlucas",
 "idp_guid": "f77d69c3-718b-4b64-be5c-767e42a1213c",
 "iss": "https://example.com/default/",
 "tok_type": "IT",
 "user_riskscore": 65,
 "aal": "AAL1",
 "tid": "default",
 "sid": "4b0f09b2-8006-4e10-9a84-099a991f27bf",
 "aud": [
```

```
    "6f0f804b-38ec-4e91-aac9-f112ffc849ce",
    "https://example.com/default/"
  ],
  "acr": "urn:iam:acr:aal1",
  "nbf": 1592228201,
  "azp": "6f0f804b-38ec-4e91-aac9-f112ffc849ce",
  "auth_time": 1592228201,
  "exp": 1592249801,
  "iat": 1592228201,
  "client_name": "DefaultTenantClient",
  "jti": "0843475a-5bf7-47db-a752-bb9d6e21faa9",
  "idp_type": "ldap"
 }
```


Response


```
 {
  "access_token" : "eyJ4NXQMUoSRyg.....",
  "token_type" : "Bearer",
  "expires_in" : 3600
 }

 Decoded Access Token
 {
  "sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
  "idp_name": "SecureLDAP",
  "user_loginid": "kevinlucas",
  "idp_guid": "f77d69c3-718b-4b64-be5c-767e42a1213c",
  "iss": "https://example.com/default/",
  "tid": "default",
  "sid": "4b0f09b2-8006-4e10-9a84-099a991f27bf",
  "sub_type": "USER",
  "azp": "6f0f804b-38ec-4e91-aac9-f112ffc849ce",
  "scope": "urn:iam:t.apps urn:iam:t.userdata urn:iam:t.signingkey urn:iam:t.ldapconfigs urn:iam:t.configs
  urn:iam:t.authpolicies urn:iam:t.certs urn:iam:m.me urn:iam:t.risk urn:iam:t.authzpolicies urn:iam:t.groups
  urn:iam:t.users urn:iam:t.authenticate",
  "exp": 1592231808,
  "iat": 1592228208,
  "client_name": "DefaultTenantClient",
  "jti": "fc40133d-9bda-4216-81e1-90c0f2ed0b87",
  "idp_type": "ldap",
  "client_tid": "default",
  "ver": "1.0",
  "user_universalid": "KevinLUCAS",
  "azpacr": "1",
  "tok_type": "AT",
  "user_riskscore": 65,
  "user_amr": [
    "PWD"
  ],
  "aud": "https://example.com/default/",
  "nbf": 1592228208,
  "user_aal": "AAL1",
  "user_acr": "urn:iam:acr:aal1"
 }
```

Device Code Grant

Learn about Device Code Grant.

The Device Code grant type allows OAuth clients to request user authorization from apps on devices with restricted input capabilities or a lack of a suitable browser. Smart TVs, media consoles, picture frames, and printers are examples of gadgets that lack a convenient input mechanism or a compatible browser for typical OAuth interactions. The device code authorization flow allows the user to review the authorization request in a secondary device, such as a smartphone that has the required input and browser capabilities to complete the user interaction.

Follow these steps to retrieve the access token using Device User Code Grant:

1. Client (Device App) issues a backchannel Authorize request with the required parameters which includes its client identifier. The Authorization server responds with the device_code, user_code, verification_uri, verification_uri_complete, expires_in, and interval. The following is the response of Device Code request API:

   NOTE

   - The ‘verification_uri’ is the 'deviceCodeflow' url configured in the app.
   - The client can generate a QR code using the value of ‘verification_uri_complete’.

```
{
 "device_code": "lhBX4HHZlaeZpBmbuBIoY81jrsHEOK",
 "user_code": "WDJBMJHT",
 "verification_uri": "https://${SSP_FQDN}/tenant/ui/v1/device",
 "verification_uri_complete": "https://${SSP_FQDN}/tenant/ui/v1/device?user_code=WDJBMJHT",
 "expires_in": 300,
 "interval": 5,
}
```

2. The client app asks the user to activate using their smartphone or computer.
   – The user requires to enter the user_code in the verification_uri that was generated in the 1st step. The user code (for example- "WDJBMJHT") is also generated in the above step.
   – The above step can also be achieved by, either accessing the verification_uri in the browser and entering the user code manually or accessing the verification_uri_complete (which auto-populates the user_code).
3. The UI invokes the backend Device User Code Verification URI. Upon success, the Authorization server redirects to the authentication flow to perform end-user authentication.

   NOTE
   Depending on the configuration, this authentication flow may redirect to your own login app if one was configured and deployed on the application entity representing the client, or else, the authentication flow is redirected to the default out-of-the-box login page.
4. On successful user authentication, the user is redirected to a page showing the "Success" message.
5. The client invokes the backchannel Token endpoint with ***grant_type=urn:ietf:params:oauth:grant-type:device_code*** and includes the device_code value that was generated in the first step.
6. The Token server generates and returns the Access Token. It also includes the Identity Token or the Refresh Token if the client has initially requested the "**openid**" scope or the "**offline_access**" scope respectively as part of invoking the Device Authorize endpoint.

Device Code APIs

The following APIs are used to retrieve the access token:

1. Device User Code Authorization
2. Device User Code Verification
3. Device User Code Access Token

These APIs have been explained in detail in the following sections.

Device User Code Authorization API

Device User Code Authorization API request: This API allows the user to create a device authorization request. The API generates the device code and the user code.

```
POST https://${SSP_FQDN}/{{tenantName}}/oauth2/v1/devicecode
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/oauth2/v1/devicecode`
  should look similar to:
  `https://example.com/default/oauth2/v1/devicecode`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

```
POST /oauth2/v1/devicecode HTTP/1.1
   Host: example.com
   Content-Type: application/x-www-form-urlencoded
   Authorization: Basic Auth
   client_id=1406020730&scope=openid
```

Where,

- client_id - The Client ID of the application. This client id is used to look up the allowed operations and to have the Device Code issued.
- client_secret - This indicates the client secret of the application. This is applicable in non-public clients only.
- scope - The device adds the required scopes to be granted in the Access token. Example values can be "openid" and "offline_access".

Device User Code Authorization API response:

```
{
 "device_code": "lhBX4HHZlaeZpBmbuBIoY81jrsHEOK",
 "user_code": "AQL9EIBH",
 "verification_uri": "https://${SSP_FQDN}/tenant/ui/v1/device",
 "verification_uri_complete": "https://${SSP_FQDN}/tenant/ui/v1/device?user_code=AQL9EIBH",
 "expires_in ": "300",
 "interval ": "60"
}
```

Where,

- device_code - The long string that the device uses to exchange for an access token. This is the user code received from the POST /devicecode API.
- user_code - The text that you enter at the URL that is listed as the value for verification_uri. This is the user code received from the POST /devicecode API.
- verification_uri - The end-user verification URI on the authorization server. This is the URL that the user must access from their device to start the sign-in process.
- verification_uri_complete - The device verification URL along with the embedded user_code on the Authorization server.
- expires_in - This indicates the number of seconds before the device_code and user_code expire.
- interval - This indicates the number of seconds the client should wait between polling the requests.

Device User Code Verification API

This is the device verification URL on the Authorization server which is invoked by the UI (front end "verification_uri"). This API validates the user code and initiates the user authentication flow.

```
GET https://${SSP_FQDN}/{{tenantName}}/oauth2/v1/devicecode?user_code={{user_code}}
```

Example of the request payload for GET /devicecode API:

```
GET /oauth2/v1/devicecode HTTP/1.1
   Host: example.com
   Content-Type: application/x-www-form-urlencoded
```

Device User Code Access Token API

This API generates the access:

```
POST https://${SSP_FQDN}/{{tenantName}}/oauth2/v1/token
curl -k --request POST \
--url 'https://example.com/default/oauth2/v1/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
-u "6cb446b9-33bf-4b97-a6e1-e7ad389974d2:2ef66db2-1e51-4868-8c72-bd57ce7f69dc" \
--data-urlencode 'grant_type=urn:ietf:params:oauth:grant-type:device_code' \
--data-urlencode 'device_code=uknSklnJPKenRZkRhMsZxJVdEnzEwF'\
--data-urlencode 'scope=urn:iam:myscopes'
```

Where,

- grant type- this is the device code grant type
- device code- this is the device code received from the POST /devicecode API
- client id- this is the Client ID of the application. This client id is used to look up the allowed operations and to have the Device Code issued.

  NOTE

  - In the case of a non-public client, enter the client_id and client_secret. In the case of a public client, enter only client_id.
  - You can pass the client_id and client_secret in the Authorization Header or pass the client _id in the request body.

The sample response of the /token API is shown below:

```
{
"access_token":"2YotnFZFEjr1zCsicMWpAA........",
"token_type":"Bearer",
"expires_in":3600,
"id_token":"tnFZFEjr1zCsicMWpAA.......",
"refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
}
```

NOTE
If you provide the "offline_access" scope in the device authorization, then the token endpoint API generates the refresh_token.

Application Configuration

The following application property should be configured while creating an application in the solution:

Property Name	Description
allowedGrantTypes	This parameter allows the device code grant type- ***urn:ietf:params:oauth:grant-type:device_c***

deviceCodeFlowURL	(Optional) This is the customized UI URL required to enter the user code in OAUTH2.0 Device Au If the customized URL is not configured, then the default Symantec Identity Security Platform ***ver*** ***${SSP_FQDN}/tenant/ui/v1/device*** is used.

The relevant application properties are covered in Managing Applications .

Using Custom Device Code Flow URL

If you are using a custom device code flow URL, for example- "deviceCodeFlowURL":

"https://www.test.com/custom/device" then the Device Authorization response would look like the following:

```
{
 "device_code": "lhBX4HHZlaeZpBmbuBIoY81jrsHEOK",
 "user_code": "AQL9EIBH",
 "verification_uri": "https://www.test.com/custom/device",
 "verification_uri_complete": "https://www.test.com/custom/device?user_code=AQL9EIBH",
 "expires_in ": "300",
 "interval ": "60"
}
```

NOTE
For the "verification_uri_complete" - The user_code gets appended as a query parameter.

Device Code with Client Assertion

The following sample shows a request payload for "/devicecode" with client assertion as client credentials:

```
curl -k --request POST \
--url 'https://example.com/default/oauth2/v1/devicecode' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'client_id=7b447d36-cac4-4a9e-b093-42f0d553d9f4' \
--data-urlencode 'client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer' \
--data-urlencode 'client_assertion=ewogICJ4NXQjUzI1NiI6ICI4bkRSWURaUF9xcENwUz.....'
```

Custom UI Actions

While using custom device code flow, the UI should be able to handle these scenarios:

1. Your custom UI should display a field to enter the user_code.
   NOTE
   If the user_code is already present in the query parameter, then pre-populate the user_code in the UI control.
2. The following IDSP API should be invoked:
   `https://${SSP_FQDN}/<tenant>/oauth2/v1/devicecode?user_code=<Entered User Code>`
   - If the user code verification is a success, the API returns 302 with the location as a server redirect to the SignIn UI (Either default or configured flowURL in the App)
   - If the user code verification is a failure, the API returns 400 with an error code and message as a JSON response.
3. Upon user authentication:
   - If the authentication is successful, the UI redirects the user to the device code flow URL with "action"="success" query parameter.
     `Example: https://www.test.com/custom/device?action=success`
   - If the authentication fails, the UI redirects the user to the device code flow URL with "action"="failed" query parameter.
     `Example: https://www.test.com/custom/device?action=failure`

Global and Tenant Settings

Configure the following tenant and global level properties to enable Device Code grant.

NOTE
NOTE: Tenant Administrators can override the global settings using tenant settings.

Property Name	Property Scope	Property Description	Default Value	Comments
deviceCodeExpirySecs	Global (Tenant can override the setting)	This property represents the "expires_in" parameter which is generated as part of the response of POST / oauth2/v1/devicecode. It indicates the number of seconds before the device_code and user_code expire.	300 secs	Possible values: The range is 60secs-{deviceCodeExpiryMaxMins} (in secs)
devicePollingIntervalSecs	Global (Tenant can override the setting)	This property indicates the number of seconds the client should wait between polling requests.	5 secs	Possible values: The range is 5 sec - {deviceCodeExpiryMaxMins} (in secs)
deviceCodeExpiryMaxMins	Global	This property governs the maximum value that can be set for "deviceCodeExpirySecs" and "deviceCodePollingIntervalSecs" . The system internally translates the given mins to secs and checks the max value for deviceCodeExpirySecs / deviceCodePollingIntervalSecs at the time of property update.	5 mins	Possible values: 1 min to 60 mins

The relevant configuration settings are covered in Global/Tenant Settings.

PKCE Support in Device User Code Flow

In the case of PKCE Device Authorization request API, the authorization server performs the following tasks:

1. The client generates a code verifier, computes the code challenge, and sends the code challenge and code challenge method with the authorization request to the Device Authorization Endpoint.
2. Authorization Server at Device Authorization Endpoint records code challenge and code challenge method, and generates authorization code.
3. The client sends authorization grant_type, device_code, and code_verifier to Access Token Endpoint.
4. Authorization Server at Access Token Endpoint verifies that the request contains the code verifier parameter. Then, it validates the code verifier value with the code challenge and code challenge method that was sent in the authentication request at the authorization endpoint.

Example of Device User Code grant using PKCE

```
POST /oauth2/v1/devicecode HTTP/1.1
```

```
Host: example.com
Content-Type: application/x-www-form-urlencoded
client_id=1406020730&scope=openid
code_challenge=computedvalueofcodeverifier
code_challenge_method=plain|S256
```

Device User Code Error Codes

8100060	Invalid user code
8100061	Invalid request
8100062	Failed to create flowstate
8100063	User code expired
8100064	Missing/Invalid device code
8100065	The authorization for this token is pending
8100066	Polling too quickly
8100067	Forbidden request
8100068	Missing user code
8100074	Device code expired

The following error codes are specified for use with the device authorization grant in token endpoint responses:

authorization_pending	The authorization request is still pending as the end-user has not yet completed the user interaction. The client should repeat the access token request to the token endpoint.
slow_down	This error indicates that the polling is too fast. The polling in seconds is happening faster than the time configured in the setting- '**deviceCodePollingIntervalSecs**'.
access_denied	The authorization request was denied.
invalid_client	The client Id provided in the request is invalid.
expired_token	The "device_code" has expired, and the device authorization session has concluded. The client may commence a new device authorization request but should wait for user interaction before restarting to avoid unnecessary polling.
unsupported_grant_type	This error is displayed when the grant_type parameter is not specified, or the given value is incorrect.
invalid_grant	The grant type provided in the request is invalid. The grant type does not exist in IDSP.

Refresh Token Grant

Learn about the Refresh Token Grant.

This topic includes the following sections:

- Refresh Token
- Steps to obtain a Refresh Token
- Step to obtain a new Access Token using the Refresh Token grant
- Step to obtain both, new Access Token and new Refresh Token using the Refresh Token grants

Refresh Tokens are credentials used to re-generate an Access Token without re-authenticating, when the current access token becomes invalid or expires. When using the Refresh Token, you may request for the same scopes that were requested during the initial authentication or narrow down the scopes depending on your requirement. Refresh Tokens have a limited lifetime, controlled by application property (**metadata**), or tenant configuration setting (**refreshTokenExpiryDays**) to meet the requirements of corporate security policies. See Managing Applications for "**urn:iam:app:oauthRefreshTokenExpiryDays**" application's metadata attribute or see Managing Settings for ***"refreshTokenExpiryDays"*** setting details. Refresh Tokens are secrets and not JWT. To make use of the Refresh Token functionality, you should update the following endpoints with the scope, "**offline_access**":

- "/authorize"
- "/token"

NOTE
To know about additional settings related to the refresh token, look for the following settings in the Managing Settings page.

- keepOldRefreshTokenOnRefreshTokenFlowWithOfflineAccess
- keepOldSessionIdOnNewRefreshTokenFlowWithOfflineAccess

NOTE
To ensure accurate and reliable IP address usage in complex network environments (which may alter user IP information due to firewalls or load balancers), leverage the Symantec Identity Security Platform application metadata. To know more, see HTTP Header and End User IP Address Usage.

Steps to obtain a Refresh Token

The following step shows an example of how to send a Refresh Token request using "/token" and "/authorize" endpoint.

Perform the following step:

1. In the "/token" endpoint or "/authorize" request body, enter the scope as "**offline_access**" and invoke the API. The Refresh Token will be obtained.

   NOTE
   The "**offline_access**" scope only works with '**authorization_code**', '**jwt-bearer**' or '**device_code**' grant_types.

   Example of a Refresh Token request format in the "/token" endpoint:

   ```
   curl -k --request POST --url 'https://example.com/default/oauth2/v1/token' --header 'Content-Type:
    application/x-www-form-urlencoded' -u "6cb446b9-33bf-4b97-a6e1-e7ad389974d2:2ef66db2-1e51-4868-8c72-
   bd57ce7f69dc" --data-urlencode 'grant_type=authorization_code' --data-urlencode
    'code=sodgnomNDVB723a3dPRwMBGHTYHB' --data-urlencode 'scope=offline_access'
   ```

   Example of a Refresh Token request format in the "/authorize" endpoint:

   ```
   https://example.com/default/oauth2/v1/authorize?response_type=code&client_id=6cb446b9-33bf-4b97-a6e1-
   e7ad389974d2&redirect_uri=https://
   www.example.com&scope=urn:iam:myscopesopenidemailoffline_access&state=mystate
   ```

   Example of a Refresh Token response format:

   ```
   {

   "access_token" : "eyJ4NXQjUzI1NiI6ImNJbUNYN1FyRy1BUUpkWWZ2Wlctd1...X6HEbM-Hdh-hzXqHA"

   "token_type" : "Bearer",

   "expires_in" : 3600,

   "id_token" : "eyJ4NXQjUzI1NiI6ImNJbUNYN1FyRy1BUUpkWWZ2W....3qliSzy_Khn2wgqQ2bg2t8Keg",
   ```

```
"refresh_token": "m5370ROmNDVB7a3dPRwMs5FDTZjfFc"

}
```

Step to obtain a new Access Token using the Refresh Token grant

Perform the following step:

1. In the "/token" endpoint request body, enter the value of the Refresh Token that you obtained in the previous section and invoke the "/token" API. The new Access Token is obtained.
   Example of a request body that includes the refresh_token which is used to obtain the new Access Token:
   ```
   curl -k --request POST --url 'https://example.com/default/oauth2/v1/token' --header
    'Content-Type: application/x-www-form-urlencoded' -u "6cb446b9-33bf-4b97-a6e1-
   e7ad389974d2:2ef66db2-1e51-4868-8c72-bd57ce7f69dc" --data-urlencode 'grant_type=refresh_token' --data-
   urlencode 'refresh_token=m5370ROmNDVB7a3dPRwMs5FDTZjfFc' --data-urlencode 'scope=<subset of scopes>'
   ```
   The following table shows the request parameters required for getting the new Access Token:

Name	Description
client_id	Client Id of the client that was registered with IDSP
client_secret	Client secret that is associated with the client_id
grant_type	The grant type supported for generating a new Access Token using the Refresh Token is "**refresh_token**". This grant type can be used only with 'authorization_code', 'jwt-bearer' or 'device_code' grant type.
refresh_token	Value of the Refresh Token
scope	Subset of scopes.

   Example of a response body which grants the new Access Token:
   ```
   {

   "access_token" : "eyJ4NXQjUzI1NiI6ImNJbUNYN1FyRy1BUUpkWWZ2Wlctd1...X6HEbM-Hdh-hzXqHA",

   "token_type" : "Bearer",

   "expires_in" : 3600

   }
   ```

Step to obtain both, new Access Token and new Refresh Token using the Refresh Token grants

Perform the following step:

1. In the "/token" endpoint request body, enter the grant type as "refresh_token" and the value of the "refresh_token" that you have obtained from the "Step to obtain Refresh Token" section.
   ```
   curl -k --request POST --url 'https://example.com/default/oauth2/v1/token' --header
    'Content-Type: application/x-www-form-urlencoded' -u "6cb446b9-33bf-4b97-a6e1-
   e7ad389974d2:2ef66db2-1e51-4868-8c72-bd57ce7f69dc" --data-urlencode 'grant_type=refresh_token ' --data-
   urlencode 'refresh_token=m5370ROmNDVB7a3dPRwMs5FDTZjfFc' --data-urlencode 'scope=<subset of scopes>'
   ```
   NOTE
   To obtain a new Refresh Token, passing ***scope= "offline_access"*** in the "/token" endpoint is mandatory.
   Example of a response body that grants the new Access Token and the Refresh Token:

```
{

"access_token" : "eyJ4NXQjUzI1NiI6ImNJbUNYN1FyRy1BUUpkWWZ2Wlctd1...X6HEbM-Hdh-hzXqHA",

"token_type" : "Bearer",

"expires_in" : 3600,

"refresh_token": "tGzv3JOkF0XG5Qx2TlKWIA"

}
```

Resource Owner

Legacy OAuth 2.0 flow for direct credential validation

The Resource Owner grant type is a legacy OAuth 2.0 flow where the application directly receives an access token after the user's credentials (username and password) are validated by the Symantec Identity Security Platform.

> **NOTE**
> This is explicitly noted as a legacy grant type that should only be used when a legacy component specifically requires it.

Implementation

Token Endpoint

```
POST https://{{sspHost}}/{{apiPathTenant}}/oauth2/v1/token
```

Required Headers:

- **Authorization Header**
  Pass the application Client ID and secret in the request in the authorization header.
- **Content Type Header**
  ```
  content-Type: application/x-www-form-urlencoded
  ```

Sample Request

```
curl --request POST \
 --url https://<ssp-hostname>/<tenantName>/oauth2/v1/token \
 --header 'authorization: Basic <Base64 encode of Client ID:Client Secret>' \
 --header 'content-type: application/x-www-form-urlencoded' \
 --data grant_type=password \
 --data username=<userName> \
 --data scope=urn:iam:myscopes \
 --data 'password=<replace by user password>'
```

Where,

- grant_type: This value must be set to "password"
- username: The resource owner username
- password: The resource owner password
- scope: The scope of the access request

Sample Response

```
HTTP/1.1 200 OK
    Content-Type: application/json;charset=UTF-8
    Cache-Control: no-store
    Pragma: no-cache

    {
      "access_token":"2YotnFZFEjr1zCsicMWpAA",
      "id_token":"xetotnFZFEjr1zCsicMWpAA",
      "scope":"urn:iam:t.all",
      "token_type":"Bearer",
      "expires_in":3600,
      "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA"
    }
```

Authorization Server Endpoints

Learn about Authorization server endpoints, such as /jwks, /authorize, /token, /revoke, /introspect, /devicecode, /openid-configuration, /logout.

The following endpoints are the OAuth Authorization Server endpoints that are exposed by Symantec Identity Security Platform.

- Prerequisites
- Discovery Endpoint
- JSON Web Key Set (JWKS) Endpoint
- Authorize Endpoint
  - Authorize Code Flow
- Token Endpoint
- Introspect Endpoint
- UserInfo Endpoint

NOTE
Replace the text within the placeholder < > with the appropriate values in all the endpoints shown in the following table.

Endpoint	Description
https://<host>/<tenant>/.well-known/openid-configuration	Provides metadata about the authorization server
https://<host>/<tenant>/oauth2/v1/jwks	Provides applications with public keys as required to verify the signature of a JWT issued by the IDSP Authorization Service, or to encrypt a JWT sent to the Authorization server (such as encrypting Identity Token Hint).
https://<host>/<tenant>/oauth2/v1/authorize	Authenticates user identity, enabling users to authorize clients requesting access to protected resources
https://<host>/<tenant>/oauth2/v1/token	Generates access tokens and refresh tokens for a client or user
https://<host>/<tenant>/oauth2/v1/revoke	Revokes an access token or a refresh token
https://<host>/<tenant>/oauth2/v1/userinfo	Returns claims or assertions about the user authenticated using the `/authorize` flow
https://<host>/<tenant>/oauth2/v1/introspect	Returns information about a token

Endpoint	Description
https://<host>/<tenant>/oauth2/v1/devicecode	• The POST /devicecode API allows the user to create a device authorization request. The API generates the device code and the user code. • The GET /devicecode API is the device verification URL on the Authorization server which is invoked by the UI (front end "verification_uri"). This API validates the user code and initiates the user authentication flow.
https://<host>/<tenant>/oauth2/v1/logout	Terminates the user session as specified by either of the following options: • specify the logout session in the "session_state" query string parameter • if not specified in the "session_state" query string parameter, IDSP logs out the session indicated by the "sspsession" cookie. Once the session is logged out, IDSP redirects the client to a Post Logout URL if specified in the "post_logout_redirect_uri" query string paramter. If this is not specified, IDSP redirects to a default Post logout URI which is searched for in the following order: 1. App-level Post logout URL 2. Tenant Setting's postLogoutUrl 3. Application's Home URL 4. Tenant's Home URL If the client specifies the Post Logout URL query string parameter, then the URL must be authorized. This URL must be defined in one of the following places: It should either match the default Post logoutUrl, or default logoutUrl (which is configured in the application logoutUrl, or in the tenant setting logoutUrl), or any of the application's redirect URL.

Prerequisite

Install cURL to execute the samples provided in the sections below.

> **NOTE**
> A recommended way to verify the service URL is by running the service discovery curl command.

Discovery Endpoint

The discovery endpoint lets clients obtain information needed to interact with the OAuth Authorization Server, including its OAuth 2.0 endpoints.

Request

```
https://<host>/<tenant>/.well-known/openid-configuration
```

Response

```
{
"issuer": "https://<ssphost>/default/",
"authorization_endpoint": "https://<ssphost>/default/oauth2/v1/authorize",
"device_authorization_endpoint" : "https://<ssphost>/default/oauth2/v1/devicecode",
"token_endpoint": "https://<ssphost>/default/oauth2/v1/token",
"userinfo_endpoint": "https://<ssphost>/default/oauth2/v1/userinfo",
"revocation_endpoint": "https://<ssphost>/default/oauth2/v1/revoke",
"jwks_uri": "https://<ssphost>/default/oauth2/v1/jwks",
"response_types_supported": [
   "code",
```

```
    "token",
    "id_token",
    "id_token token"
],
"subject_types_supported": [
    "public"
],
"id_token_signing_alg_values_supported": [
    "RS256"
],
"scopes_supported": [
    "openid",
    "profile",
    "email",
    "phone",
    "address",
    "groups"
],
"token_endpoint_auth_methods_supported": [
    "client_secret_post",
    "client_secret_basic",
    "private_key_jwt"
],
"claims_supported": [
    "aud",
    "email",
    "email_verified",
    "phone_number",
    "phone_number_verified",
    "family_name",
    "middle_name",
    "given_name",
    "iss",
    "iat",
    "name",
    "picture",
    "sub",
    "locale",
    "nickname",
    "preferred_username",
    "profile",
    "website",
    "gender",
    "zoneinfo",
    "birthdate",
    "updated_at",
    "sub",
    "address",
    "groups",
],
"response_modes_supported": [
    "fragment",
    "query"
```

```
],
"grant_types_supported": [
   "authorization_code",
   "implicit",
   "urn:ietf:params:oauth:grant-type:jwt-bearer",
   "client_credentials",
   "refresh_token",
   "urn:ietf:params:oauth:grant-type:device_code"
],
"id_token_encryption_alg_values_supported": [
   "RSA-OAEP",
   "RSA-OAEP-256"
],
"id_token_encryption_enc_values_supported": [
   "A128GCM",
   "A192GCM",
   "A256GCM",
   "A192CBC-HS384",
   "A128CBC-HS256"
],
"userinfo_signing_alg_values_supported": [
   "RS256"
],
"userinfo_encryption_alg_values_supported": [
   "RSA-OAEP",
   "RSA-OAEP-256"
],
"userinfo_encryption_enc_values_supported": [
   "A128GCM",
   "A192GCM",
   "A256GCM",
   "A192CBC-HS384",
   "A128CBC-HS256"
],
"code_challenge_methods_supported": [
   "plain",
   "S256"
],
"claims_parameter_supported": false,
"request_parameter_supported": false,
"request_uri_parameter_supported": false,
"require_request_uri_registration": false
}
```


NOTE
For detailed examples of Discovery Endpoint, see here.

JSON Web Key Set (JWKS) Endpoint

A read-only JWKS service is for use by someone who develops PKI-based code to perform one of two activities:

- validate signature of received artifacts
- encrypt artifacts to be sent

The main purpose of a JWKS endpoint is to provide applications with public keys as required to verify the signature of a JWT issued by the IDSP Authorization Service, or to encrypt a JWT sent to the Authorization server (such as encrypting Identity Token Hint).

Request

```
https://<host>/<tenant>/oauth2/v1/jwks
```

Response


```
{
"keys": [
{
"kty": "RSA",
"x5t#S256": "btn1jH3EjLwP6jcWWMPK9-t0rnFMM0eEIUhYym7xRnE",
"e": "AQAB",
"use": "sig",
"x5t": "NmopuEBSpLIdTSlmZbhDP_TLUsY",
"kid": "SSP-SigningKey",
"x5c": [
"MIIDjjCCAnagAw"
],
"key_ops": [
"verify",
"sign"
],
"alg": "RS256",
"n": "32A4qjqY3M_UKxD3GR0rQuCH927_6-Z7"
},
{
"kty": "RSA",
"x5t#S256": "btn1jH3EjLwP6jcWWMPK9-t0rnFMM0eEIUhYym7xRnE",
"e": "AQAB",
"use": "enc",
"x5t": "NmopuEBSpLIdTSlmZbhDP_TLUsY",
"kid": "EncryptionKey",
"x5c": [
"MIIDjjCCAnagAwIBAgIBAD",
"MIIDeTCCAmGgAwIBAgIJALViYpj"
],
"key_ops": [
"decrypt",
"encrypt"
],
"alg": "RS256",
"n": "32A4qjqY3M_UKxD3GR0rQuCH927_6-Z7qszjEGesEKq2J_zLdv"
}
]
}
```


> **NOTE**
> For detailed examples of JWKS Endpoint, see here.

Authorize Endpoint

The main purpose of this endpoint is to support the Authorization Code flow. The authorization endpoint is used by the authorization server to initiate the user authentication ceremony to assert the user identity. Then, the client exchanges the

Authorization Code for the Access Token (and Identity Token if the client initially requested openid scope as part of the "/authorize" request) by invoking authorization's server Token endpoint. The client also exchanges the Authorization code for the Refresh Token, if the client initially requested "offline_access" scope as part of the "/authorize" request or as part of "/token" request.

```
https://<host>/<tenant>/oauth2/v1/authorize
```

Security consideration for clients The `/authorize` endpoint also supports *Implicit* flow for use by native apps, such as single-page apps (see Implicit Flow). However, use of the Implicit flow is not recommended. Instead, single-page apps should use Authorization Code flow with Proof Key for Code Exchange (PKCE). When public clients such as native and single-page applications request Access Tokens, there are additional security concerns since such clients cannot securely store a Client Secret or indicate a trusted callback URL. The PKCE specification (OAuth 2.0 RFC 7636) addresses this issue. Adding PKCE parameters to the Authorization Code flow enables the client to introduce a secret (Code Verifier) that can be validated by the authorization server. The client creates and sends the Code Challenge as part of the `/authorize` request to get the Authorization Code. When the client exchanges the Authorization Code for an Access Token, it includes the Code Verifier. As a result, even if attackers intercept the `/authorize` request, they will not be able to exchange it for the access token without the Code Verifier. See PKCEfor more information.

"/authorize" parameters

Parameter	Optionality	Description
client_id	Required	The Client ID for this application. This client id is used to look up allowed grants and operations, and to have the Authorization Code.
response_type	Required	• or the Authorization Code flow, specify `code`. • For the Implicit flow, specify one of the following: – `token` – `id_token` – `id_token token`
scope	Required	This is a space-delimited string containing scopes (privileges) for which you want to request authorization. You can request scopes supported by the target API (for example, `urn:iam:myscopes` or `urn:iam:t_authenticate`) as indicated in the REST API section, or any of the OpenID Connect scopes supported by the Allowed OpenID Scopes configuration parameter of the application whose client is requesting access (for example, `profile` or `email`). If initiating the OpenID flow, add `openid` to the request issuance of the Identity Token. Indicate additional OpenID scopes (client's Allowed OpenID Scopes) to authorize the client to invoke the Userinfo endpoint to request user data represented by those scopes. If initiating the Oauth or OIDC flow, add "offline_access" as part of the "/token" request to get the Refresh Token.
redirect_uri	Required	This is the URL-encoded endpoint to which the client will be sent back once the Authorization Code has been issued by the authorization server. The Authorization Code will be available in the `code` URL parameter. This URL must be specified exactly as provided in one of the values during client registration.
state	Optional	The value supplied by the client will be returned by the authorization server along with the Authorization Code.
nonce	Optional (for use with OpenID only)	This is an optional security mechanism to protect against replay attacks. This should be a strong random value associated with the internal application session. If asking for an Identity Token (in the case of OpenID Connect requests only), this value is included in the Identity Token.

Parameter	Optionality	Description
code_challenge_method	Optional (for use with PKCE only)	The method used to generate the **PKCE** challenge. The PKCE specification includes `S256` and `plain`. Use of `plain` is not recommended.
code_challenge	Optional (for use with PKCE only)	Challenge generated from the PKCE Code Verifier
state	Optional (for use with PKCE only)	This is an optional security parameter for clients to protect against certain attacks such as Cross-Site Request Forgery (CSRF, XSRF). Usually, this value is generated by cryptographically binding to some browser state such as an application cookie to detect a possible request hijacking attempt.
login_hint	Optional	This is the hint to pre-populate the user name field in the out of the box signin application.
prompt	Optional	The "prompt" property only accepts the value "login". When the "prompt" parameter is set to "login" in the "/authorize" request, users are prompted to complete the authentication process, regardless of whether they already have a valid session in the IDSP browser. This requires users to re-enter any applicable credentials to proceed with the authentication flow.
id_token_hint	Optional	The ID Token being passed as a hint about the End-User's current or past authenticated session. If the End-User identified by the ID Token is already logged in or is logged in as a result of the request (with the OP possibly evaluating other information beyond the ID Token in this decision), then the Authorization Server returns a positive response; otherwise, it MUST return an error, such as login_required. When possible, an id_token_hint SHOULD be present when prompt=none is used and an invalid_request error MAY be returned if it is not; however, the server SHOULD respond successfully when possible, even if it is not present. When validating the signature of the Identity Token Hint provided into "/authorize" API, the certificate used for validating the token signature is looked up in this order: x5t+sha256, x5t, kid+alias, and finally the issuer claim.
response_mode	Optional	This property specifies whether the authentication response to redirect_uri must be sent in the form of query parameters or encoded as HTML form values that are sent in the POST method. By default, Authorization Code Flow uses query as the response mode. To send the response in query format, you need not send this parameter in the authentication request. To send the response in encoded as HTML form values, specify form_post as the value in the authentication request.

Authorize Code Flow

The Authorization endpoint is used to obtain an Access Token on behalf of the user whose identity will be authenticated by the authorization server. Additionally, this endpoint is used by the OpenID Connect flow to acquire an Identity Token for the authenticated user by specifying `openid` as one of (or the only one) the requested scopes. The same endpoint is also used to acquire a refresh Token for the authenticated user by specifying `"offline_access"` as one of the requested scopes.

1. The client issues a front channel Authorize request with the required parameters.
2. The Authorization server redirects to the authentication flow to perform end-user authentication.

   NOTE
   Depending on the configuration, this authentication flow may redirect to an out-of-the-box *login* page or your own login app if one was configured and deployed on the application entity representing the client.

3. On successful user authentication:
   a. The Authorization server authorizes the client's requested scopes against the authorized scopes, and if valid, generates and returns the Authorization Code.

      NOTE
      If the Client is acting On-Behalf-Of, the authorization check uses the user's privileges as authorized privileges. Otherwise, the authorization check uses the intersection of client's and user's privileges as authorized privileges.

   b. The Authorization server executes a 302 Redirect to the client's registered callback URL with the Authorization Code.
4. The client invokes the backchannel Token endpoint with `grant_type=authorization_code` and includes the Authorization Code value.
5. The Token server generates and returns the Access Token. It also includes the Identity Token or the Refresh Token if the client has initially requested the `openid` scope or the "**offline_access**" scope respectively as part of invoking the Authorize endpoint.

Authorize Code Flow Examples Browser request to generate Authorization Code to be exchanged for an access token with REST API scopes only:

```
https://example.com/default/oauth2/v1/authorize?response_type=code&client_id=6cb446b9-33bf-4b97-a6e1-
e7ad389974d2&redirect_uri=&scope=uhttps://www.example.com&scope=urn:iam:myscopes&state=mystate
```

NOTE
With PKCE, the value of `code_CHALLENGE` is computed and added to the query string params (`...&code_challenge_method=S256&code_challenge=9f86d081884c7d6#39;`)

After user authentication for user "kevinlucas" is complete, the Authorization server uses a 302 Redirect to the client's redirect_uri with the Authorization Code (and State if originally provided when invoking the `/authorize` endpoint).

```
https://www.example.com/?code=66fRVqzzFdTAjmMlcirB2r1P2D0MBd&X-TRANSACTION-ID=7fd70067-d5cb-4ea6-8ea6-
c3ca4551b5cc&state=mystate
```

The client then exchanges the Authorization code for the Access Token:
Example

```
curl -k --request POST --url 'https://example.com/default/oauth2/v1/token' --header 'Content-Type:
 application/x-www-form-urlencoded' -u "6cb446b9-33bf-4b97-a6e1-e7ad389974d2:2ef66db2-1e51-4868-8c72-
bd57ce7f69dc" --data-urlencode 'grant_type=authorization_code' --data-urlencode
 'code=66fRVqzzFdTAjmMlcirB2r1P2D0MBd'
{
"access_token" : "eyJ4NXQjUzI1NiI6ImN.......",
"token_type" : "Bearer",
"expires_in" : 3600
}

Decoded Access Token
{
"client_tid": "default",
"sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
"ver": "1.0",
"idp_name": "SecureLDAP",
"user_universalid": "KevinLUCAS",
"user_loginid": "kevinlucas",
"idp_guid": "f77d69c3-718b-4b64-be5c-767e42a1213c",
"iss": "https://example.com/default/",
"azpacr": "1",
"tok_type": "AT",
"tid": "default",
```

```
"aud": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"nbf": 1592229699,
"sub_type": "USER",
"azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"scope": "urn:iam:t.apps urn:iam:t.userdata urn:iam:t.signingkey urn:iam:t.ldapconfigs urn:iam:t.configs
 urn:iam:t.authpolicies urn:iam:t.certs urn:iam:t.risk urn:iam:t.authzpolicies urn:iam:t.groups
 urn:iam:t.users urn:iam:t.authenticate",
"exp": 1592233299,
"iat": 1592229699,
"client_name": "testapp",
"jti": "2eaf8974-ac4e-46bc-8827-126a74b40744",
"idp_type": "ldap"
}
```

Browser request to generate an Authorization Code to be exchanged for the Identity Token (OpenID Connect) and Access Token with OpenID Connect scopes only:

```
https://example.com/default/oauth2/v1/authorize?response_type=code&client_id=6cb446b9-33bf-4b97-a6e1-
e7ad389974d2&redirect_uri=https://www.example.com&scope=openid%20email&state=mystate&nonce=mynonce
```

After user authentication for user "kevinlucas" is complete, the authorization server uses a 302 Redirect to the client's redirect_uri with Authorization Code (and State if originally provided when invoking the `/authorize` endpoint).

```
https://www.example.com/?code=m5370ROmNDVB7a3dPRwMs5FDTZjfFc&X-TRANSACTION-ID=0e4649c1-1c85-4fca-
b446-0bd02e3675b9&state=mystate
```

The client then exchanges the Authorization code for the Access Token and the Identity Token. Note that the Identity Token has the `nonce` claim:

Example

```
curl -k --request POST --url 'https://example.com/default/oauth2/v1/token' --header 'Content-Type:
 application/x-www-form-urlencoded' -u "6cb446b9-33bf-4b97-a6e1-e7ad389974d2:2ef66db2-1e51-4868-8c72-
bd57ce7f69dc" --data-urlencode 'grant_type=authorization_code' --data-urlencode
 'code=m5370ROmNDVB7a3dPRwMs5FDTZjfFc'
{
"access_token" : "eyJ4NXQjUzI1N......",
"token_type" : "Bearer",
"expires_in" : 3600,
"id_token" : "eyJ4NXQjUzI1NiI6ImN......"
}

Decoded Access Token
{
"client_tid": "default",
"sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
"ver": "1.0",
"idp_name": "SecureLDAP",
"user_universalid": "KevinLUCAS",
"user_loginid": "kevinlucas",
"idp_guid": "f77d69c3-718b-4b64-be5c-767e42a1213c",
"iss": "https://example.com/default/",
"azpacr": "1",
"tok_type": "AT",
"tid": "default",
"aud": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"nbf": 1592230192,
"sub_type": "USER",
"azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
```

```
"scope": "openid email",
"exp": 1592233792,
"iat": 1592230192,
"client_name": "testapp",
"jti": "619c78ea-d5fc-4fed-88e8-dfbd2a7bb31e",
"idp_type": "ldap"
}

Decoded Identity Token
{
"at_hash": "O2eVIySLj6oDVd8WF9aLrg",
"sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
"ver": "1.0",
"idp_name": "SecureLDAP",
"user_universalid": "KevinLUCAS",
"amr": [
"PWD"
],
"user_loginid": "kevinlucas",
"idp_guid": "f77d69c3-718b-4b64-be5c-767e42a1213c",
"iss": "https://example.com/default/",
"tok_type": "IT",
"nonce": "mynonce",
"aal": "AAL1",
"tid": "default",
"sid": "fc665a4e-56d6-4aca-9900-d783f3548e6d",
"aud": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"acr": "urn:iam:acr:aal1",
"nbf": 1592230192,
"azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"auth_time": 1592230174,
"exp": 1592251792,
"iat": 1592230192,
"client_name": "testapp",
"jti": "6a2837a2-bcb0-43b0-b61a-6ba85c3195c1",
"idp_type": "ldap"
}
```


Browser request to generate Authorization Code to be exchanged for access token (REST API scopes and OpenID Connect scopes) and identity token (OpenID Connect flow):

```
https://example.com/default/oauth2/v1/authorize?response_type=code&client_id=6cb446b9-33bf-4b97-a6e1-
e7ad389974d2&redirect_uri=https://www.example.com&scope=urn:iam:myscopes%20openid%20email&state=mystate
```

After user authentication for user "kevinlucas" is complete, the Authorization server uses a 302 Redirect to the client's redirect_uri with the Authorization Code (and State if originally provided when invoking the `/authorize` endpoint).

```
https://www.example.com/?code=1jk4lfwJYrtvoIlgWP66WaUIHBhND8&X-TRANSACTION-ID=3df091cd-c918-4e7c-84ed-
c5ce5bc8123e&state=mystate
```

The client then exchanges the Authorization code for the Access Token and the Identity Token. Note that the Identity Token has the `nonce` claim:

Example

```
curl -k --request POST --url 'https://example.com/default/oauth2/v1/token' --header 'Content-Type:
 application/x-www-form-urlencoded' -u "6cb446b9-33bf-4b97-a6e1-e7ad389974d2:2ef66db2-1e51-4868-8c72-
bd57ce7f69dc" --data-urlencode 'grant_type=authorization_code' --data-urlencode
 'code=1jk4lfwJYrtvoIlgWP66WaUIHBhND8'
```

```
{
"access_token" : "eyJ4NXQjUzI1NiI6ImNJ....",
"token_type" : "Bearer",
"expires_in" : 3600,
"id_token" : "eyJ4NXQjUzI1NiI6ImNJbUN......"
}
```


Decoded Access Token


```
{
"client_tid": "default",
"sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
"ver": "1.0",
"idp_name": "SecureLDAP",
"user_universalid": "KevinLUCAS",
"user_loginid": "kevinlucas",
"idp_guid": "f77d69c3-718b-4b64-be5c-767e42a1213c",
"iss": "https://example.com/default/",
"azpacr": "1",
"tok_type": "AT",
"tid": "default",
"aud": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"nbf": 1592230692,
"sub_type": "USER",
"azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"scope": "openid email",
"exp": 1592234292,
"iat": 1592230692,
"client_name": "testapp",
"jti": "3d07dd7c-dd36-4146-adea-d1fc24f7c689",
"idp_type": "ldap"
}
```


Decoded Identity Token


```
{
"at_hash": "s3cWtDuhkRua9KQjZn2EyQ",
"sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
"ver": "1.0",
"idp_name": "SecureLDAP",
"user_universalid": "KevinLUCAS",
"amr": [
"PWD"
],
"user_loginid": "kevinlucas",
"idp_guid": "f77d69c3-718b-4b64-be5c-767e42a1213c",
"iss": "https://example.com/default/",
"tok_type": "IT",
"aal": "AAL1",
"tid": "default",
"sid": "966e4eec-cae5-4a27-8c4e-d3bdc02733b2",
"aud": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"acr": "urn:iam:acr:aal1",
"nbf": 1592230692,
"azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
```

```
"auth_time": 1592230673,
"exp": 1592252292,
"iat": 1592230692,
"client_name": "testapp",
"jti": "9bdbefb7-8618-4c03-8461-20becaafab31",
"idp_type": "ldap"
}
```


NOTE
For more details on Authorization Code Flow, see here.

Token Endpoints

For details on Token endpoint, see OAuth2 Token Endpoint.

Revoke Token

For details on Revoke token, see here

Introspect Endpoint

The Introspect Endpoint lets clients verify tokens (Access Token or Refresh Token). Note that these remote calls are considered to be expensive.

```
https://<host>/<tenant>/oauth2/v1/introspect
```

NOTE
The introspection of the encrypted ID Token is not yet supported.

For maximum performance, applications should perform token validation on the client side using token validation best practices that includes at least the following:

- Validate token signature
- Validate token expiration
- Validate token audience

Click here to see Request and Response of this endpoint

```
curl -k --request POST \
--url 'https://example.com/default/oauth2/v1/introspect' \
--header 'Content-Type: application/x-www-form-urlencoded' \
-u "6cb446b9-33bf-4b97-a6e1-e7ad389974d2:2ef66db2-1e51-4868-8c72-bd57ce7f69dc" \
--data-urlencode 'token=eyJ4NXQjUzI1NiI6ImNJbUNYN1FyRy1BUUpkWWZ2Wlctd1B....'

Decoded Token
{
"client_tid": "default",
"sub": "2db6e0b1-8f69-4f1c-b308-6ddacdca76f1",
"ver": "1.0",
"iss": "https://example.com/default/",
"azpacr": "1",
"tok_type": "AT",
"tid": "default",
"aud": "https://example.com/default/",
"nbf": 1592163258,
"sub_type": "CLIENT",
"azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
```

```
"scope": "urn:iam:t.configs urn:iam:t.authenticate",
"exp": 1592166858,
"iat": 1592163258,
"client_name": "testapp",
"jti": "bbb543b2-3e4b-4be1-bcd7-5dafc3170186"
}
```

Response

```
{
"active" : true,
"scope" : "urn:iam:t.configs urn:iam:t.authenticate",
"sub" : "2db6e0b1-8f69-4f1c-b308-6ddacdca76f1",
"iss" : "https://example.com/default/",
"jti" : "bbb543b2-3e4b-4be1-bcd7-5dafc3170186",
"aud" : [ "https://example.com/default/" ],
"exp" : 1592166858000,
"iat" : 1592163258000,
"nbf" : 1592163258000,
"client_id" : "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"token_type" : "Bearer"
}
```

Example: Request for Access token introspection with client assertion as client credentials:

```
curl -k --request POST \
--url 'https://example.com/default/oauth2/v1/introspect' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'token=eyJ4NXQjUzI1NiI6ImNJbUNYN1FyRy1BUUpkWWZ......'
--data-urlencode 'client_id=7b447d36-cac4-4a9e-b093-42f0d553d9f4' \
--data-urlencode 'client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer' \
--data-urlencode 'client_assertion=ewogICJ4NXQjUzI1NiI6ICI4bkRSWURaUF9xcENwUz.....'

Decoded Token
{
"client_tid": "default",
"sub": "2db6e0b1-8f69-4f1c-b308-6ddacdca76f1",
"ver": "1.0",
"iss": "https://example.com/default/",
"azpacr": "1",
"tok_type": "AT",
"tid": "default",
"aud": "https://example.com/default/",
"nbf": 1592163258,
"sub_type": "CLIENT",
"azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"scope": "urn:iam:t.configs urn:iam:t.authenticate",
"exp": 1592166858,
"iat": 1592163258,
"client_name": "testapp",
"jti": "bbb543b2-3e4b-4be1-bcd7-5dafc3170186"
}
```

Response

```
{
```

```
"active" : "true",
"scope" : "urn:iam:t.configs urn:iam:t.authenticate",
"sub" : "2db6e0b1-8f69-4f1c-b308-6ddacdca76f1",
"iss" : "https://example.com/default/",
"jti" : "bbb543b2-3e4b-4be1-bcd7-5dafc3170186",
"aud" : [ "https://example.com/default/" ],
"exp" : 1592166858000,
"iat" : 1592163258000,
"nbf" : 1592163258000,
"client_id" : "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"token_type" : "Bearer"
}
```


NOTE

For more details on Introspect endpoint, see here.

UserInfo Endpoint

The UserInfo Endpoint gets identity information about the user. The supplied Access Token must have been generated as part of the OpenID Connect authorization flow.

```
https://<host>/<tenant>/oauth2/v1/userinfo
```

Example: Request with Access Token in the authorization header

```
curl -k --request GET \
--url 'https://example.com/default/oauth2/v1/userinfo' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Authorization: Bearer eyJ4NXQjUzI1Niuk5VYTsg'

Decoded Access Token
{
"client_tid": "default",
"sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
"ver": "1.0",
"idp_name": "SecureLDAP",
"user_universalid": "KevinLUCAS",
"user_loginid": "kevinlucas",
"idp_guid": "f77d69c3-718b-4b64-be5c-767e42a1213c",
"iss": "https://example.com/default/",
"azpacr": "1",
"tok_type": "AT",
"tid": "default",
"aud": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"nbf": 1592181818,
"sub_type": "USER",
"azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"scope": "openid email",
"exp": 1592185418,
"iat": 1592181818,
"client_name": "testapp",
"jti": "7eae9886-ce9d-4ea4-abac-e377240362e9",
"idp_type": "ldap"
}
```

Response

```
{
"aud":"6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"sub":"5aee7c64-a5e1-4899-aad3-51e1f569c292",
"iss":"https://example.com/default/",
"email":"Kevin.LUCAS@DEMOCORP.com"
}
```

Example: Request with Access Token in the POST body

```
curl -k --request POST \
--url 'https://example.com/default/oauth2/v1/userinfo' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'access_token=eyJ4NXQjUzI1NiI6ImNJbUNY3KUNQ7_UrdYuk5VYTsg'
'


Decoded Access Token
{
"client_tid": "default",
"sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
"ver": "1.0",
"idp_name": "SecureLDAP",
"user_universalid": "KevinLUCAS",
"user_loginid": "kevinlucas",
"idp_guid": "f77d69c3-718b-4b64-be5c-767e42a1213c",
"iss": "https://example.com/default/",
"azpacr": "1",
"tok_type": "AT",
"tid": "default",
"aud": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"nbf": 1592181818,
"sub_type": "USER",
"azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"scope": "openid email",
"exp": 1592185418,
"iat": 1592181818,
"client_name": "testapp",
"jti": "7eae9886-ce9d-4ea4-abac-e377240362e9",
"idp_type": "ldap"
}
```

Response

```
{
"aud": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
"sub": "5aee7c64-a5e1-4899-aad3-51e1f569c292",
"iss": "https://example.com/default/",
"email": "Kevin.LUCAS@DEMOCORP.com"
}
```

NOTE

For more details on UserInfo Endpoint, see here.

OAuth2 Token Endpoint

The OAuth2 Token endpoint enables client applications to obtain access tokens using multiple OAuth2 grant types including client credentials, authorization code, JWT bearer, refresh token, device code, and resource owner password credentials for API authentication and authorization.

API Overview

The OAuth2 Token endpoint implements multiple OAuth 2.0 grant flows, allowing client applications to authenticate with the authorization server to obtain access tokens. This endpoint supports Client Credentials Grant, Authorization Code Grant, JWT Bearer Grant, Refresh Token Grant, Device Code Grant, and Resource Owner Password Credentials Grant flows for various authentication scenarios.

Key Functions:

- **Token Generation:** Issues JWT access tokens for authenticated client applications
- **Client Authentication:** Validates client credentials using Basic Authentication
- **Scope Authorization:** Grants specific permissions based on requested scopes and client privileges
- **Session Management:** Provides token expiration and lifecycle management

Use Cases:

- **Client Credentials:** API-to-API communication where client applications need to access protected resources
- **Authorization Code:** Web applications requiring user authorization with redirect-based flows
- **JWT Bearer:** Token exchange scenarios using existing JWT assertions (e.g., OIDC ID tokens)
- **Refresh Token:** Obtaining new access tokens using previously issued refresh tokens without user re-authentication
- **Device Code:** Input-constrained devices (smart TVs, IoT devices) that require user authorization on a separate device
- **Resource Owner Password:** Trusted applications that can securely handle user credentials (legacy systems, first-party applications)
- Backend services requiring authentication for system-level operations
- Automated processes and scheduled jobs that need API access
- Microservices authentication within a trusted environment

The returned access token is a JWT that contains client identity, granted scopes, and metadata required for subsequent API calls. This token must be included in the Authorization header of requests to protected endpoints.

Endpoint Overview

Operation Description	HTTP Method	Endpoint URL
Obtain OAuth2 access token using client credentials grant	POST	https://{{sspHost}}/{{apiPathTenant}}/oauth2

Request Parameters

Headers

Header Name	Required	Value
Content-Type	Yes	application/x-www-form-urlencoded

Authentication

There are four ways to authenticate the token endpoint:

- **Basic Authentication using client credentials in Header**:

Username: Client ID (The Client ID of the application. This client id is used to look up the allowed operations and to have the AZ Code issued.)

– **Password:** Client Secret (Required for confidential and trusted clients.)

- **Client credentials in Request Body**:
  The same client credentials can be passed in the request body instead of the header.

  NOTE

  Important: Client credential information can only be passed in one place. If credentials are duplicated in both header and body, an error is displayed.
- **Client Assertion**:
  Authentication using client assertion method. For more information, see Client Assertion.
- **MTLS (Mutual TLS)**:
  Authentication using mutual TLS certificates. For more information, see MTLS.

Request Body Parameters

Parameter	Required	Type	Grant Type	Description
grant_type	Yes	String	• client_credentials • authorization_code • jwt-bearer • refresh_token • device_code • password (Resource Owner)	These are the OA to invoke the "... /
scope	Optional	String	• client_credentials • authorization_code • jwt-bearer • refresh_token • password (Resource Owner)	This is a space-de scopes (privileges authorization. You by the target API or urn:iam:t_authe API section. Requ authorization_cod You can request f obtaining a Refres when the grant-ty or "jwt-bearer".
code	Yes	String	authorization_code	Authorization cod endpoint
code_verifier	Optional	String	• authorization_code • device_code	PKCE Code Verif Optional for grant was used with the
redirect_uri	Yes	String	authorization_code	Redirect URI that authorization requ
assertion	Yes	String	jwt-bearer	JWT assertion tok used for authentic When validating t provided into the the certificate use looked up in this c and finally the issu
refresh_token	Yes	String	refresh_token	Refresh token obt request

device_code	Yes	String	device_code	Device code obta endpoint
username	Yes	String	password (Resource Owner)	Resource owner's
password	Yes	String	password (Resource Owner)	Resource owner's

Response Format

Success Response (HTTP 200)

Field	Type	Description
access_token	String	JWT access token for API authentication
token_type	String	Token type, "Bearer"
expires_in	Integer	Token lifetime in seconds
scope	String	Space-separated list of granted scopes
id_token	String	JWT ID token (returned when openid scope
refresh_token	String	Refresh token for obtaining new access toke

Example Response:

```
{
  "access_token": "eyJ4NXQjUzI1NiI6Im1RNXJCd2tPSjF5X2toZU9Vc...",
  "id_token": "jUzyJ4NXQI1NiI6Im1RNXJCd2tPSjF5X2toZU9Vc...",
  "refresh_token": "iI6Im1RNXJCd2tPSjeyJ4NXQjUzI1NF5X2toZU9Vc...",
  "token_type": "Bearer",
  "expires_in": 3600,
  "scope": "urn:iam:t.all urn:iam:m.meclient urn:iam:t.userpasswordadmin"
}
```

Request Examples

cURL Example for client_credentials grant

```
curl -X POST "https://{{sspHost}}/<tenantName>/oauth2/v1/token" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -u "username:password" \
  -d "grant_type=client_credentials&scope=urn:iam:t.all"
```

cURL Example for authorization_code grant

```
curl --request POST \
  --url "https://{{sspHost}}/<tenantName>/oauth2/v1/token" \
  --header "authorization: Basic e3tjbGllbnRJZH19Ont7Y2xpZW50U2VjcmV0fX0=" \
  --header "content-type: application/x-www-form-urlencoded" \
  --data "grant_type=authorization_code" \
  --data "code=<code>" \
  --data "code_verifier=<enter the code verifier value>" \
  --data "redirect_uri=https://sampleapp-undefined/sample-rp/oidc/callback" \
  --data "scope=openid groups profile email"
```

cURL Example for jwt-bearer grant

```
curl --request POST \
  --url "https://{{sspHost}}/<tenantName>/oauth2/v1/token" \
  --header "authorization: Basic e3tjbGllbnRJZH19Ont7Y2xpZW50U2VjcmV0fX0=" \
  --header "content-type: application/x-www-form-urlencoded" \
  --data "grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer" \
  --data "assertion=<oidcIdToken>" \
  --data "scope=openid groups profile email"
```

cURL Example for refresh_token grant

```
curl --request POST \
  --url "https://{{sspHost}}/<tenantName>/oauth2/v1/token" \
  --header "authorization: Basic e3tjbGllbnRJZH19Ont7Y2xpZW50U2VjcmV0fX0=" \
  --header "content-type: application/x-www-form-urlencoded" \
  --data "grant_type=refresh_token" \
  --data "refresh_token=<refreshToken>" \
  --data "scope=openid groups profile email offline_access urn:iam:sso:idprop"
```

cURL Example for device_code grant

```
curl --request POST \
  --url "https://{{sspHost}}/<tenantName>/oauth2/v1/token" \
  --header "authorization: Basic e3tjbGllbnRJZH19Ont7Y2xpZW50U2VjcmV0fX0=" \
  --header "content-type: application/x-www-form-urlencoded" \
  --data "grant_type=urn:ietf:params:oauth:grant-type:device_code" \
  --data "device_code=<device_code>" \
  --data "code_verifier=<enter the code verifier value>"
```

cURL Example for Resource Owner (password) grant

```
curl --request POST \
  --url "https://{{sspHost}}/<tenantName>/oauth2/v1/token" \
  --header "authorization: Basic e3tjbGllbnRJZH19Ont7Y2xpZW50U2VjcmV0fX0=" \
  --header "content-type: application/x-www-form-urlencoded" \
  --data "grant_type=password" \
  --data "scope=urn:iam:myscopes" \
  --data "username=<replace by username>" \
  --data "password=<replace by user password>"
```

Implementation Notes

NOTE

- Replace {{sspHost}} and <tenantName> with actual values in your environment
- The access token is a JWT that contains client and scope information
- Token expiration is typically 3600 seconds (1 hour)
- Use the access token in the Authorization header as "Bearer {access_token}" for subsequent API calls
- Granted scopes may differ from requested scopes based on client permissions

Client Assertion

Learn about the mandatory client validation claims in the headers and in the body of the JWT token.

Client assertion is a piece of data that verifies the client. It offers an alternate client authentication technique, one that does not need to send client secrets. Therefore, the client authentication can be done without passwords, by creating a JWT Token and signing it with the client's private key or a secret. The JWT is called client assertion in the context of client authentication. The client assertion is included in a token request as the value of the client_assertion request parameter.

Client Assertion is accepted by the "/token" and "/introspect" endpoints in Symantec Identity Security Platform. These endpoints must add the following parameters in the POST body of the "/token" and "/introspect" endpoint:

- **client_assertion = <jwt-assertion>**
- **client_assertion_type = urn:ietf:params:oauth:client-assertion-type:jwt-bearer**

The following table shows the mandatory client validation claims in the headers and in the body of the JWT token.

Mandatory JWT Headers/Claims

Header Claims	Validations
kid	Key Identifier. Used to identify the trusted third-party certificate to validate the assertion signature. The Key ID in a client assertion = Cert Alias in IDSP (/ Certs)
x5t	A unique certificate entry should exist in IDSP matching the x5t stored on the Cert resource. Lookup is based on the normal JWT validation order already in place.
alg	Identifies the specific type of JWT signing algorithm. The algorithm supported is RS256.
typ	Identifies the type of assertion. This must be JWT.

Mandatory JWT Body/Claims

Body Claims	Validations
aud	Audience(s) that this JWT Token is intended for. It is an array of strings or string values that must contain, "<token-issuer-url>" as one of the AUD values. For example, https://ssp.vlssp.bm.com:443/default/
iss	This is the Client id for the JWT assertion issued by the client.
sub	A valid client_id. **Note**: An active app should exist at the current tenant level in IDSP. This value must be the same as the client_id request param. This client id is the one in the client assertion, if the client id in params is different then the request fails.
jti	A unique identifier for the token, which can be used to prevent reuse of the token. These tokens MUST only be used once unless conditions for reuse were negotiated between the parties; any such negotiation is beyond the scope of this specification.
exp	Expiration time on or after which the JWT Token must not be accepted for processing.
nbf	Time until when JWT cannot be accepted. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.
iat	Time at which the JWT was issued. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.

Example of a JTW Client Assertion Payload

```
{
"kid": "AppSigningKeyAlias"
"x5t": "SuWQHOUnyRz3CtYEs3axMkcLFx0",
```

```
"x5t#S256": "d-TBloYpys_k122k8gXrQnmfeXb0FxiwKo_6Svw5fTE",
 "alg": "RS256",
 "typ": "JWT"
}


{
"aud": ["https://example.com:443/default/", "https://${SSP_FQDN}/default/"]
"iss": "<client-id>",
"sub": "<client-id>",
"jti": "a2b0bd31-f81d-4f6c-8656-e9dca7355892"
"exp": 1535810505,
"nbf": 1535806905,
"iat": 1535806905
}
```

Configuring OAuth mTLS between Symantec Identity Security Platform Clients and Cluster

Learn how to enable and enforce OAuth mutual Transport Layer Security(mTLS) within an Symantec Identity Security Platform environment.

mTLS enhances security by authenticating clients using their client certificates when interacting with critical OAuth endpoints, **Token**, **Revoke**, and **Introspect**. It details both the deployment-level configuration using Helm charts and the application-specific enforcement through metadata attributes.

This topic includes the following sections:

- Enabling mTLS in the IDSP deployment
- Enforcing mTLS at the app level
- Sample cURL command

Enabling mTLS in the IDSP deployment

To enable Mutual TLS (mTLS) for IDSP deployment, you need to configure the Helm chart with specific flags. It is important to note that mTLS is an optional feature at the deployment level, and its enforcement can be controlled at the application level. Therefore, enabling mTLS in theIDSP deployment does not automatically mandate all clients to present a client certificate. However, if these mTLS parameters are omitted during the initial deployment and the mTLS feature is required later, a redeployment of IDSP with the necessary parameters becomes mandatory. Following redeployment, you need to configure the application metadata values to fully enable mTLS.

Helm Chart Configuration:

To enable mTLS, pass these parameters along with IDSP helm install command:

```
--set ssp.ingress.mtls.enabled=true \
--set ssp.ingress.mtls.caCertSecretName=<secret-name>
```

NOTE

- **caCertSecretName** - This is a Kubernetes secret containing a list of trusted CA certificates.

Enforcing mTLS at the app level

To mandate mTLS for a specific OAuth Application, you need to configure the following metadata attributes for that application. These attributes allow you to specify how mTLS is used for client authentication and token binding.

Metadata Property	Description	Default Value	Possible Va
urn:iam:app:**oauthAuthMethod**	To enable mTLS for the app, configure "tls_client_auth" metadata.	null/blank	• tls_clier • client_s • client_s • private_
(Optional) urn:iam:app:**oauthUseMtlsCertTokenBinding**	Indicates whether to populate the cnf (confirmation) claim in the access token or not.	false	true/false
urn:iam:app:**tlsClientAuthnMethod**	To configure the client authentication TLS method, consider one of the following values from the client certificate received: • tls_client_auth_subject_dn - Complete subject DN • tls_client_auth_san_dns - Subject Alternate Name - DNS value • tls_client_auth_san_ip - Subject Alternate Name - IP value • tls_client_auth_san_email - Subject Alternate Name - EMAIL value • tls_client_auth_san_uri - Subject Alternate Name - URI value	tls_client_auth_subject_dn	• tls_clier • tls_clier • tls_clier • tls_clier • tls_clier
urn:iam:app:**tlsClientAuthnValue**	The admin should configure the expected values for comparing with the client certificate values. At runtime, when a client attempts to authenticate through mTLS, the value based on the tlsClientAuthnMethod (for example, the full Subject DN) from the client certificate is extracted. Then, compare that extracted value to the configured "tlsClientAuthnValue" value. If they match, the client certificate is considered valid for authentication.		<expected
(Optional) urn:iam:app:**oauthRequireTlsAuth**	Configure this metadata to mandate mTLS authentication along with the other supported authentications like client secret basic/post, client assertion.	false	true/false

These metadata properties are also covered in Application Metadata Properties.

Example: The following example shows how to set the metadata attributes for an OAuth Application to enforce mTLS authentication and token binding:

```
"metadata": [
{
"name": "urn:iam:app:oauthAuthMethod",
"value": "tls_client_auth"
},
{
"name": "urn:iam:app:oauthUseMtlsCertTokenBinding",
"value": "true"
},
```

```
{
"name": "urn:iam:app:tlsClientAuthnMethod",
"value": "tls_client_auth_subject_dn"
},
{
"name": "urn:iam:app:tlsClientAuthnValue",
"value": "CN=ClientApp,OU=FN,O=XYZ,ST=AP,C=IN"
},
{
"name": "urn:iam:app:oauthRequireTlsAuth",
"value": "true"
}
]
```

Sample cURL command

The following cURL command performs an OAuth 2.0 Client Credentials Grant flow with Mutual TLS (mTLS) client authentication to obtain an access token from an endpoint:

```
curl --cert <PATH_TO_CLIENT_CERT> \
--key <PATH_TO_PRIVATEKEY> \
--request POST \
--url https://<SSP_FQDN>/default/oauth2/v1/token \
--header 'content-type: application/x-www-form-urlencoded' \
--data grant_type=client_credentials \
--data client_id=c36df799-m72f-0000-5476b-n88990h3b810
```

Where,

- **--cert <PATH_TO_CLIENT_CERT>**: This specifies the path to the client's certificate (.crt file). In an mTLS handshake, the client presents this certificate to the server for authentication.
- **--key <PATH_TO_PRIVATEKEY>:** This specifies the path to the client's private key (.key file) that corresponds to the provided public certificate.
- **--request POST**: This sets the HTTP request method to POST.
- **--url https://<SSP_FQDN>/default/oauth2/v1/token**: This is the URL of the OAuth 2.0 token endpoint to which the request is being sent. It is a secure HTTPS endpoint.
- **--header 'content-type: application/x-www-form-urlencoded'**: This sets the Content-Type HTTP header, indicating that the request body is URL-encoded form data.
- **--data grant_type=client_credentials**: This sends data in the request body, specifying the grant_type as client_credentials.
- **--data client_id=c36df799-m72f-0000-5476b-n88990h3b810**: This sends the client_id as part of the request body. This identifies the OAuth client making the request.

JSON Web Encryption (JWE)

The content outlines a secure identity/authorization system that leverages JWT encryption with configurable public keys, supporting OAuth2/OpenID Connect flows. It enables clients to encrypt tokens using specific aliases, manage certificates, and through grants such as `sts-confidential` , ensuring flexible and secure token exchange.

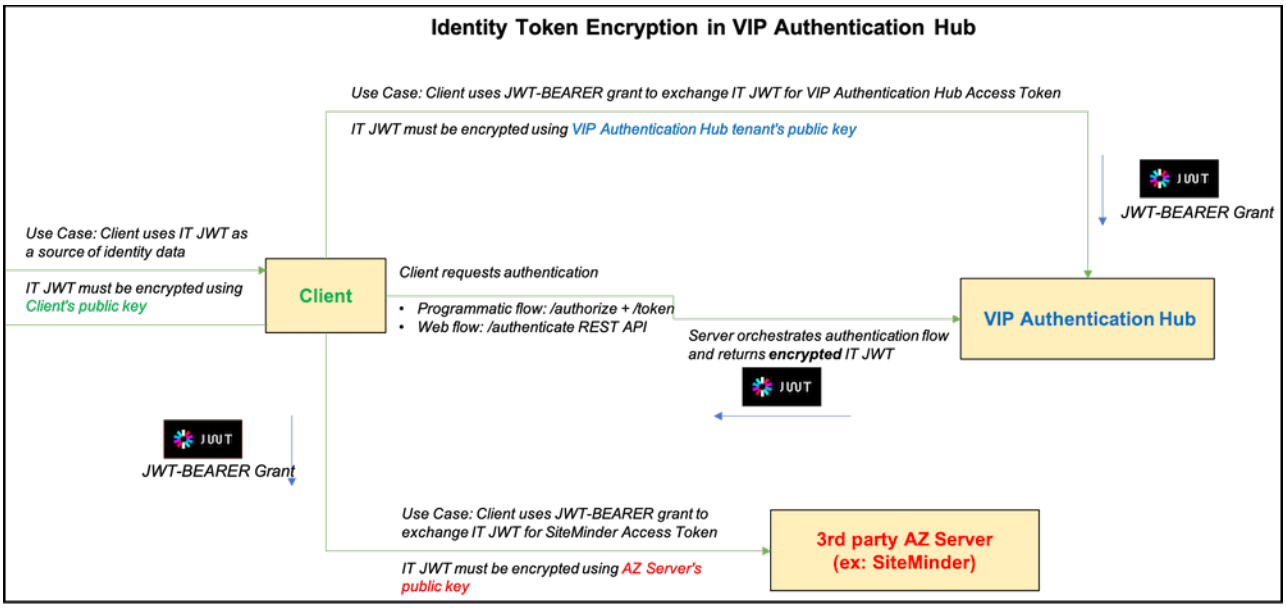
Symantec Identity Security Platform issues Identity Token (IT) for consumption by clients. Clients can use IT for a few different reasons:

- As proof of identity for client's own purposes
- Exchange IT for solution's Access Token (AT) in order to invoke solution's REST APIs
- Exchange IT for an AT from another authorization server (example: SiteMinder) that is configured to trust the issuer of the client's IT

By default IT generated by the solution is a signed JSON Web Token (JWT). JSON Web Encryption (JWE) feature can be used to encrypt IT to maintain confidentiality of IT for storage or during transmission.

Additionally, the contents of the UserInfo response to a client may be encrypted. The target of such a request is the client itself. OpenID Discovery API and JWKS API indicates availability of JWE.

Overview of JWE Support



Refer to the following JWE related topics:

- Supported Use Cases
- Global and Tenant Settings
- Keys and Certs Required
- Requesting Encrypted JWT in Authentication, Authorization Code Grant and "/userinfo" request

Supported JWE Use Cases

IT encryption is supported using the standard JWE approach to generate and/or accept encrypted Identity Token JWTs. Encryption of UserInfo JWT response is supported as well. When JWE encryption is enabled for ITs, the solution encrypts the IT using the public key of the **target** intended to receive the encrypted data (IT or userinfo response). Only the intended target meant to receive the encrypted data can decrypt it using its own private key. The configuration of the client represented by "**itEncryptionTarget"** property indicates the intended target to receive the encrypted data.

NOTE

If the client intends to use encrypted IT with multiple targets as mentioned above, then the client MUST be configured with "**itEncryptionTarget=TENANT"** in order to be able to use **sts-confidential token grant** to exchange IT encrypted by solution tenant's public key for an IT encrypted by another target's public key depending on the use case (such as client's public key or 3rd party Authorization Server public key

The following table covers supported use cases depicting the intended target of the encrypted data.

Target of the encrypted data (IT or Userinfo)	Use Case (Issue or Exchange)
Symantec Identity Security Platform	JWT-BEARER grant to exchange IT encrypted by solution tenant's public key for the solution Access Token (See JWT-Bearer grant use case)
	AZ Code grant or "/authenticate" sign-in ceremony to issue IT encrypted by solution tenant's public key
	AZ Code flow or "/authenticate" sign-in ceremony to accept IT Hint encrypted by solution tenant's public key
Client	AZ Code flow or "/authenticate" sign-in ceremony to issue IT encrypted by a client's public key (see AZ code flow grant use case)
	"/userinfo" returns JWT encrypted by client's public key (see userinfo encryption use case) **Note the following:** • Encryption of Userinfo response is independent of the **itEncryptionTarget** client property. UserInfo's JWT response is encrypted if the client is configured to request userinfo through the JWT format • The client has the option to request IT to be encrypted for a specific target by overriding application configuration during the /Authenticate request. The possible values are: – useTenantKeyForITEncryption – useUserInfoKeyForITencryption – encryptionAlias See the sections, Requesting Encrypted JWT in the Authentication flow and Requesting Encrypted JWT in AZ Code grant flow on how to pass encrypted data in the /Authenticate and /Authorize APIs.
Third-Party Authorization Server configured to trust IDSP	JWT-BEARER grant to exchange IT encrypted by a third party public key for an AT issued by that third party's Authorization Server
	AZ Code flow or "/authenticate" sign-in ceremony to issue IT encrypted by a third party's public key
Client or Third-Party Authorization Server configured to trust IDSP	**sts-confidential** grant to exchange IT encrypted by solution tenant's public key for an IT encrypted by a third party public key associated with either the client itself or another authorization server configured to accept the IT issued by the VIP Authorization Server (see STS grant use case)

Settings and Application Configurations for JWE

Learn about the tenant or global settings and application configurations that are required to enable JSON Web Encryption.

Global and Tenant Settings

Use the tenant-level properties to set the encryption algorithm and method. The following settings should be configured in the solution to enable encryption.

NOTE
Tenant Administrator can override the global settings using tenant settings.

Property Name	Property Scope	Property Description	Default Value	Comments
encryptionAlgorithm	Global (Tenant can override the setting)	This property displays the standard encryption algorithms supported by the solution. • RSA-OAEP • RSA-OAEP-256	RSA-OAEP-256	Possible values: RSA-OAEP, RSA-OAEP-256
encryptionMethod	Global (Tenant can override the setting)	This property displays the encryption methods supported by the solution. • A128CBC-HS256 • A192CBC-HS384 • A128GCM • A192GCM • A256GCM	A256GCM	Possible values: A128CBC-HS256, A192CBC-HS384, A128GCM, A192GCM, A256GCM

The relevant configuration settings are covered in Global/Tenant Settings.

Application Configuration

The following application property should be configured while creating an application in the solution:

Property Name	Description
itEncryptionTarget	Use this property to manage IT encryption. Supported values are (Case-insensitive) • NONE - IT will not be encrypted • APP - IT will be encrypted by the public key referenced by 'itEncryptionCertAlias' value. • TENANT - IT will be encrypted by solution Tenant's public key The itEncryptionTarget values that are configured through the Application can be overridden by the client. This can be done by adding additional request parameters in the following APIs: /authenticate, /authorize, /userinfo, and /token. See details below.
itEncryptionCertAlias	This property represents the alias of the encryption cert to encrypt the IT. This must be specified when 'itEncryptionTarget' is set to 'APP'.
userInfoEncryptionCertAlias	This property represents the alias of the public key to encrypt the response of the /userinfo REST API. This option is used in conjunction with the application property 'userInfoResponseFormat' and is used only when 'userInfoResponseFormat' is JWT. If the value is provided, and 'userInfoResponseFormat' is JWT - the userInfo response is signed and then encrypted, otherwise only signed.
supportedJoseHeaderParams	(Optional) This option supports comma-separated - x5t, x5t#s256, x5c Jose header parameters. If this option is left empty, then x5t and x5t#S256 are added in the JOSE Header of JWE tokens by default.

The relevant application properties are covered in Managing Applications .

Keys and Certs required for JWE

This topic describes the process of securely encrypting **Identity Tokens (ITs)** using **JWE (JSON Web Encryption).**

As JWE works using a public key of the target expected to receive the encrypted JWT, certificates have to be loaded into the solution using /Certs API prior to requesting or exchanging ITs intended to be decrypted by an entity other than the solution itself. That public key to encrypt the IT should be imported using the solution's "/Certs" REST API. Also, the certificate alias should be set up when the application is created. To encrypt the IT Token Hint received by the solution during "/authorize" flow, the third party should use the Symantec Identity Security Platform's public key. The IDSP tenant's public key is returned by the JWKS endpoint and by the "/GETSigningKey" REST API.

Requesting Encrypted JWE in various AuthN and AuthZ flows

This topic explains how to request for encrypted JWT in authentication flow, authorization code grant flow, and in "/userinfo" request. Also learn how can "sts-confidential" Token grant type be used.

Requesting Encrypted JWT in the Authentication flow

> **NOTE**
> The properties under "authContext" that is, "useUserInfoKeyForITencryption" and "useTenantKeyForITencryption" are mutually exclusive. A request with multiple properties at once results in an error response.

- When performing an authentication ceremony using programmatic REST API ("/authenticate") to acquire the IT, the client can request the IT to be encrypted by the solution tenant's public key by overriding application configuration (Application's **itEncryptionTarget** property) by passing authN context indicating use of solution tenant's public key.

```
POST /authenticate
...
"authContext": [
{
"key": "useTenantKeyForITEncryption",
"value": "true"
}
]
```

- When using authentication ceremony using programmatic REST API (/authenticate) to acquire the IT, you can force the IT to be encrypted by the certificate alias configured as 'userInfoEncryptionCertAlias' by overriding application configuration (Application's **itEncryptionTarget** property) by passing authN context indicating the use of userinfo encryption cert alias configured.

```
POST /authenticate
...
"authContext": [
{
"key": "useUserInfoKeyForITencryption",
"value": "true"
}]
```

- When using an authentication ceremony using programmatic REST API ("/authenticate") to acquire the IT, the client can request the IT to be encrypted by using a cert represented by a specific alias. This can be done by adding the "encryptionAlias" parameter into the "/authenticate" request.

```
POST /authenticate
...
"authContext": [
{
"key": "encryptionAlias",
```

```
"value":"<cert alias>"
]
```

NOTE
Pass these parameters ("useTenantKeyForITEncryption", "useUserInfoKeyForITencryption", "encryptionAlias") in the "/authenticate" call, will override the app configuration (itEncryptionTarget property) mentioned in this table.

The relevant authentication attribute details are covered in Initiate Authentication.

Requesting Encrypted JWT in AZ Code grant

- When using the AZ Code grant to obtain the solution's IT, the client can request the IT to be encrypted by the solution tenant's public key. This can be done by overwriting application configuration (**itEncryptionTarget** property) and passing the following scope in addition to other required scopes into the "/authorize" request.
  ```
  scope=urn:iam:useTenantKeyForITencryption
  ```
- When using the AZ Code grant to obtain the solution's IT, you can force the IT to be encrypted by the certificate alias configured as 'userInfoEncryptionCertAlias' by overwriting application configuration (**itEncryptionTarget** property) by passing the following scope into the /authorize request (in addition to other required scopes).
  ```
  scope=urn:iam:useUserInfoKeyForITencryption
  ```
  NOTE
  - The scopes *urn:iam:useUserInfoKeyForITencryption* and *urn:iam:useTenantKeyForITencryption* are mutually exclusive. The request with both the scopes at-a-time results in an error response.
  - Pass these parameters ("urn:iam:useTenantKeyForITencryption", "urn:iam:useUserInfoKeyForITencryption") in the "/authenticate" call to override the app configuration (Application's **itEncryptionTarget** property) mentioned in this table.
- When using the AZ Code grant to obtain the solution's IT, the client can request the IT to be encrypted by using a cert represented by a specific alias. This can be done by adding the "encryptionAlias" parameter into the "/authorize" request.
  ```
  GET /authorize?client_id=<>&scope=<>&response_type=code&redirect_uri=<value>&encryptionAlias=<value>
  ```

Requesting Encrypted JWT in /userinfo request

Configure the "userInfoEncryptionCertAlias" application property to contain the alias to a client's certificate in the certs store. Userinfo JWT response is encrypted if the alias had been provided and the application's "userInfoEndpointResponseFormat" is "jwt". Otherwise, the userinfo JWT will not be encrypted.

- When using the userinfo request to acquire the Userinfo JWT response, the client can request the userinfo to be encrypted by using a cert represented by a specific alias. This can be done by adding the "encryptionAlias" parameter into the "/userinfo" request.
  ```
  POST /userinfo
  access_token=<value>&encryptionAlias=<cert-alias-value>
  ```
  NOTE
  Pass these parameters ("encryptionAlias") in the "/userinfo" call to override the app configuration (Application's **itEncryptionTarget** property) mentioned in this table.

Using "sts-confidential" Token grant type

- Whenever a client requires an encrypted IT to be sent to multiple targets, the generated IT must be encrypted using the solution tenant's public key. After that, the sts-confidential token grant can be used to exchange that IT for an IT encrypted by the intended targets' public key. Therefore, a client can use the sts-confidential token grant to swap the IT encrypted with the solution tenant's public key for the IT encrypted by the public key of the intended target as configure by application's **itEncryptionTarget** property.

Sample Request showing sts_confidential grant type in the "/token" request:

```
POST https://${SSP_FQDN}/<tenant-name>/oauth2/v1/token

grant_type=sts_confidential&

assertion=<IT Encrypted using AH tenants key>
```

Sample Response:

```
{
"id_token": "<IT encrypted with Client's public key configured in the App>"
}
```

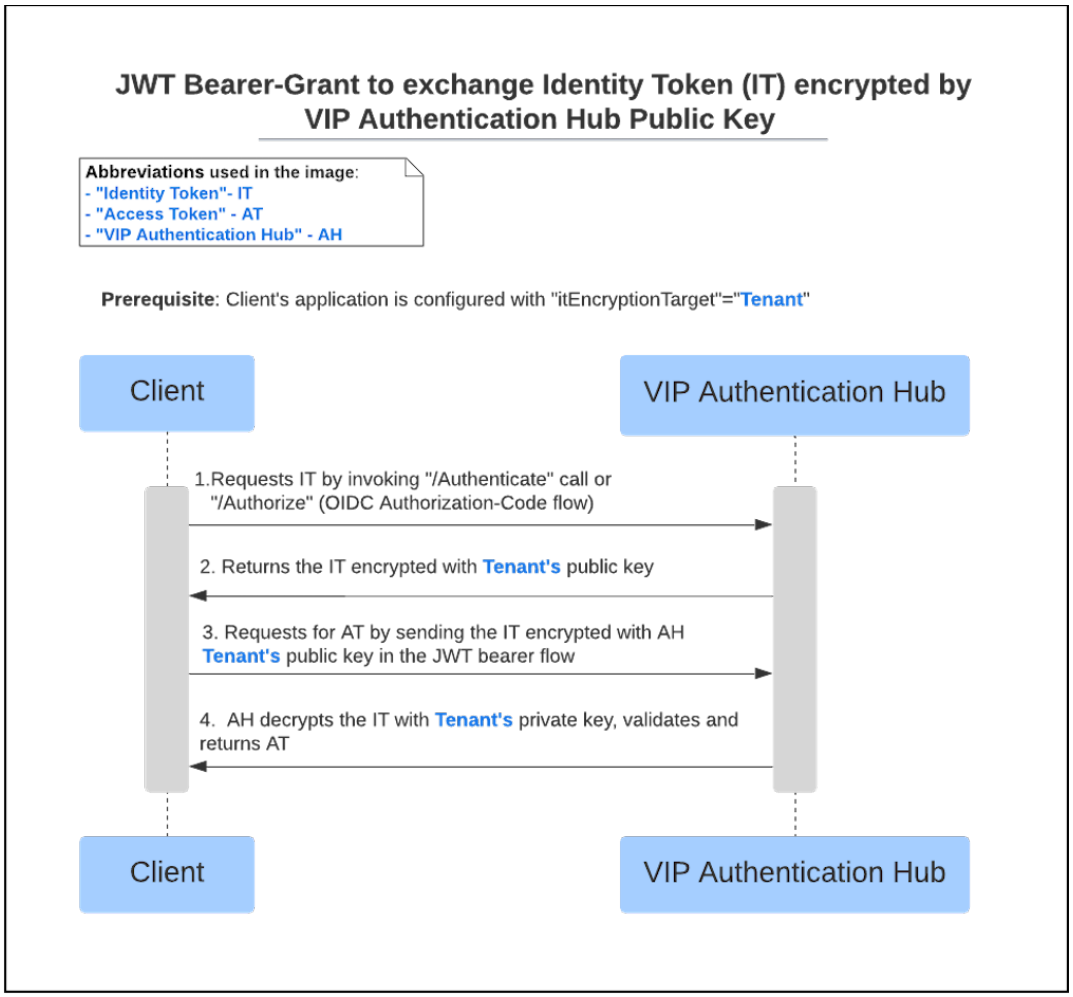
- When using the "sts-confidential" Token grant type request to encrypt the IT in the response, the client can request the token to be encrypted by using a cert represented by a specific alias. This can be done by adding the "encryptionAlias" parameter into the "/token" request.

  ```
  POST /token?grant_type=sts-confidential&assertion=<IDToken>&encryptionAlias=<cert-alias-value>
  ```

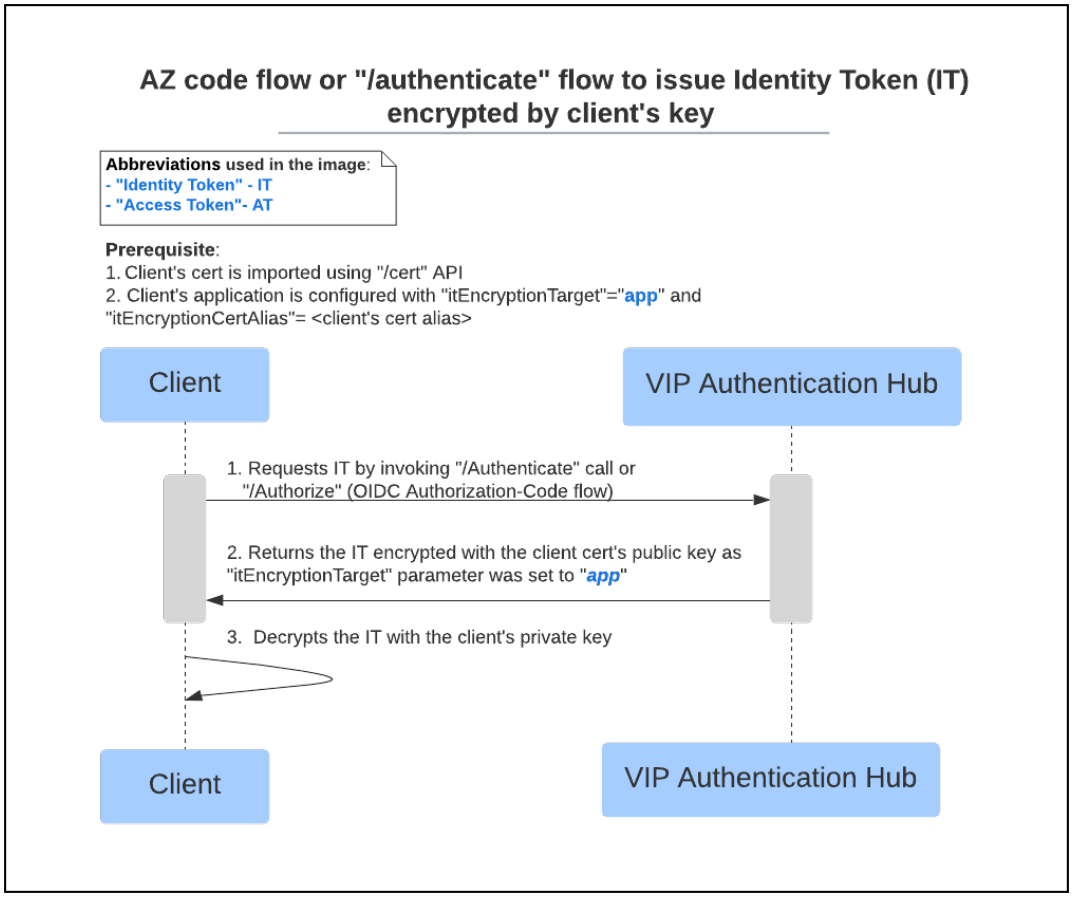
  NOTE
  Pass these parameters ("encryptionAlias") in the "/token" call to override the app configuration (Application's **itEncryptionTarget** property) mentioned in this table.

Use Cases Detailed

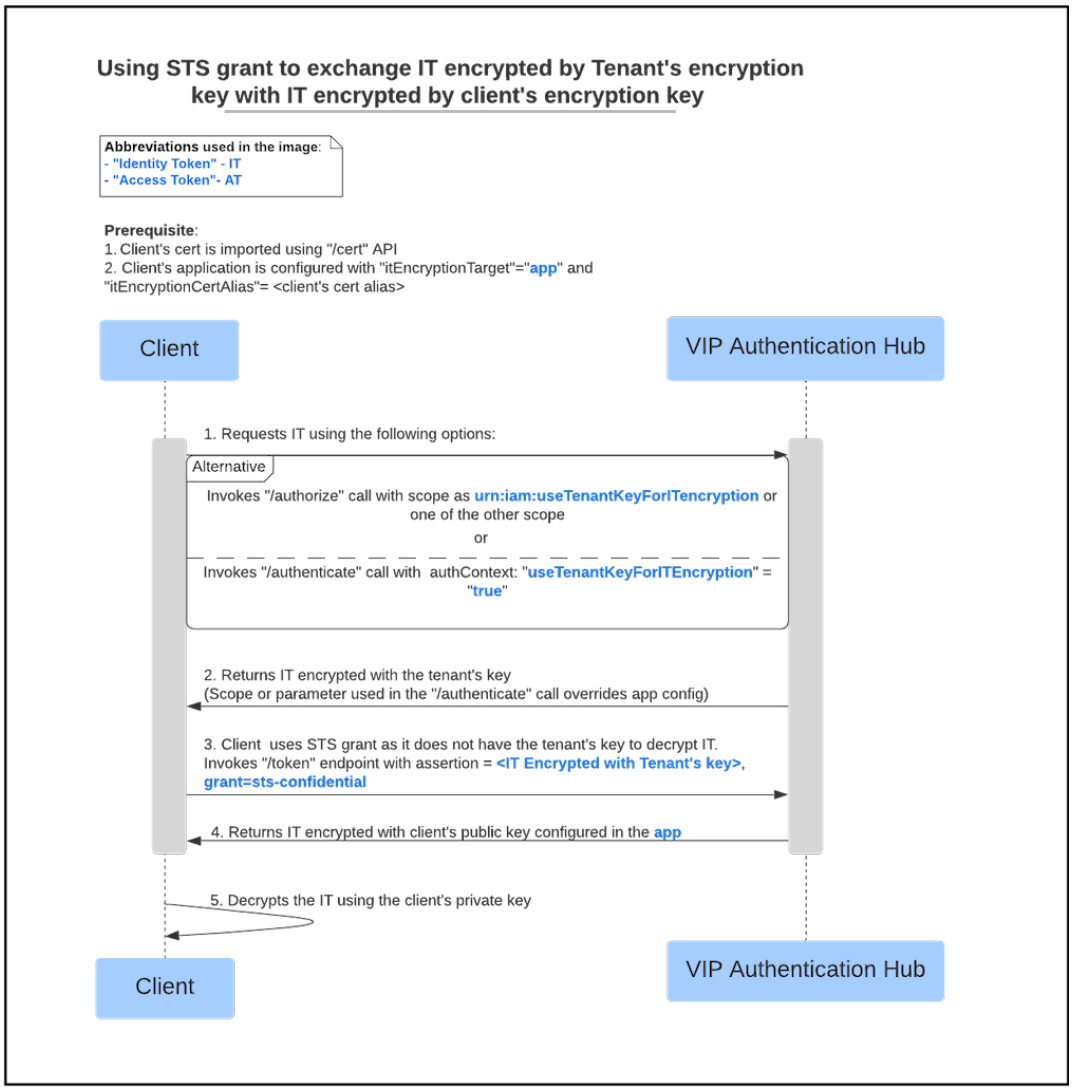
JWT-BEARER grant to exchange IT encrypted by Symantec Identity Security Platform's encryption key for the IDSP Access Token



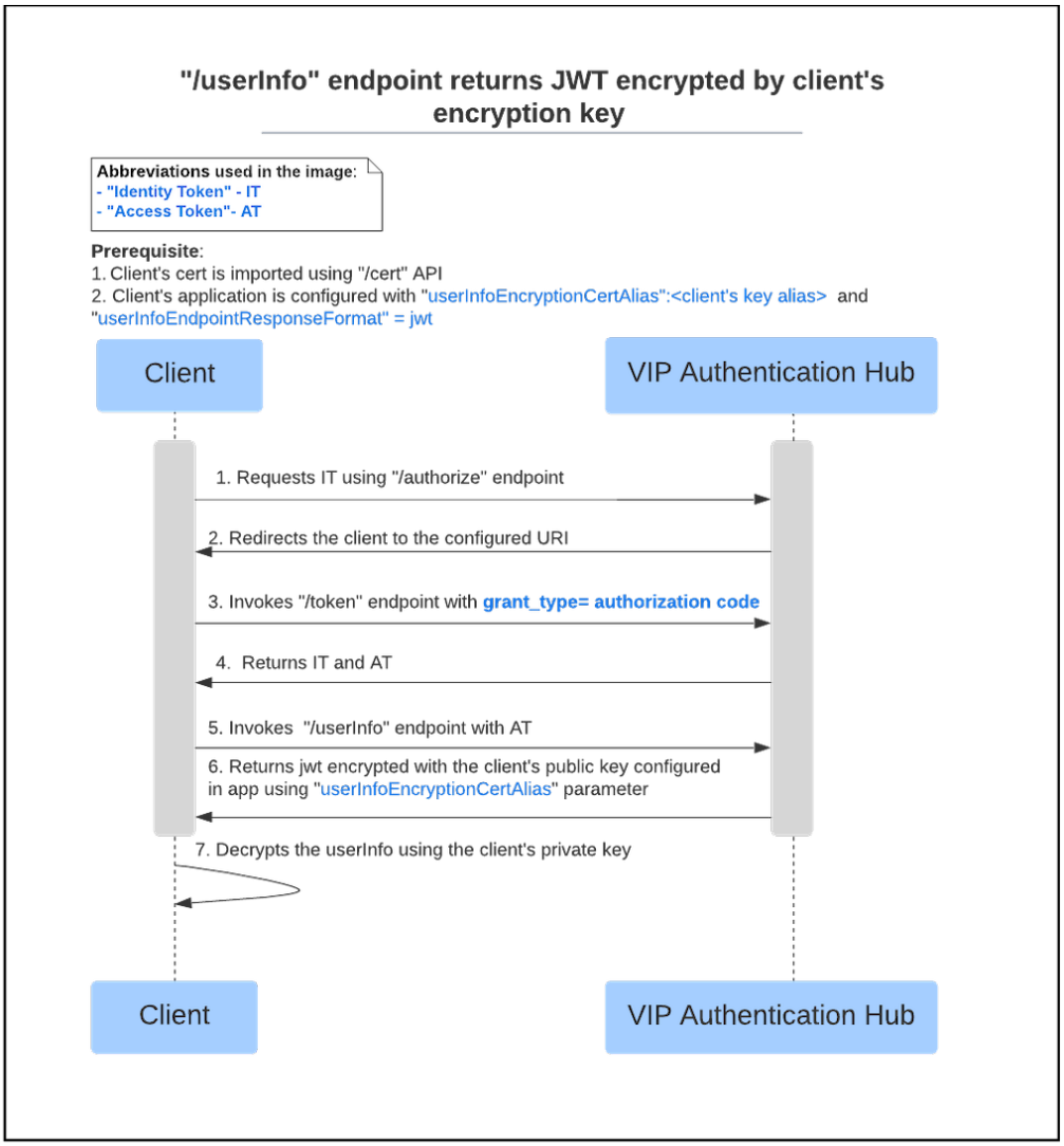
AZ Code flow or "/authenticate" sign-in ceremony to issue IT encrypted by a client's key



STS grant to exchange IT encrypted by IDSP's encryption key for an IT encrypted by a third party key associated with either the client itself or another authorization server configured to accept the IT issued by the VIP Authorization Server



"/userinfo" returns JWT encrypted by the client's encryption key (provided client is configured to request JWT)



Secure Communication for Clients Using PKCE

The communication between Authorization Endpoint and Access Token Endpoint might be compromised for public, confidential, and trusted clients. An interceptor can use the generated Authorization Code to request the tokens. To mitigate the interception attack on Authorization Code Flow for the mentioned clients, you can configure Proof Key for Code Exchange (PKCE) and secure the communication between Authorization Endpoint and Access Token Endpoint.

PKCE uses the following parameters:

- code verifier
  Defines the cryptographical random key that is generated dynamically to bind the authorization request to the token request.
- code challenge
  Defines the plain or hash value of the code verifier based on the code challenge method. The authorization server compares this value with the code verifier before generating the token.

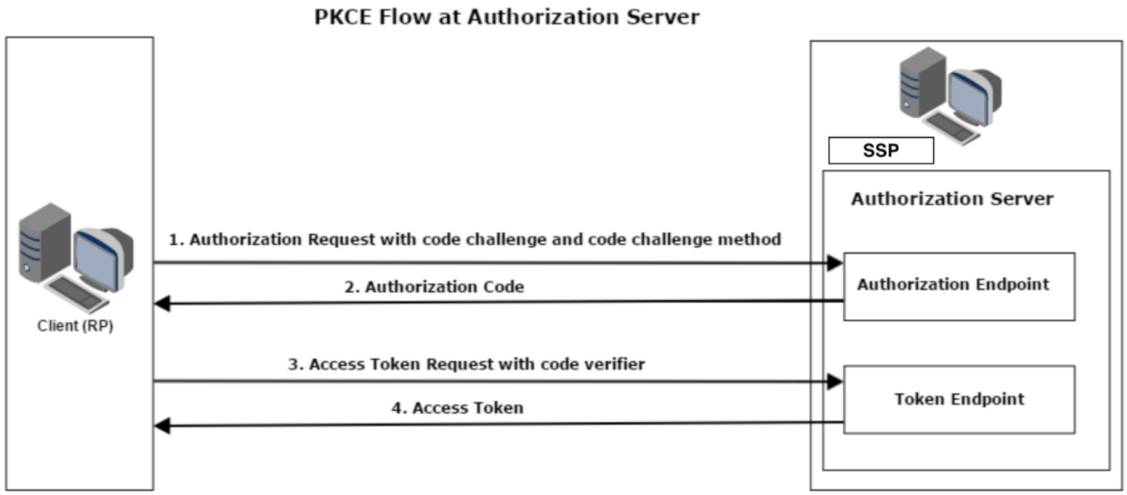
NOTE
When a code challenge is generated, characters such as '/' or '+' may appear in the challenge. For example, the code challenge may look like this-

- /iWStCpyfpd/BVlHOFtwnMgrFrmof4jGq/OQDWXQzcM
- "73l8gRjwLftklgfdXT+MdiMEjJwGPVMsyVxe16iYpk8"

In such a case, you should replace the characters '/' and '+' with '_' and '-' respectively. See PCKE RFC link for more information.

- code challenge method
  Defines how the code challenge is generated. The value is either Plain or S256. If only a code challenge is sent in the authentication request, the code challenge method defaults to Plain.

The following diagram explains the PKCE process flow:



When PKCE is enabled, the authorization server performs the following tasks:

1. The client generates a code verifier, computes the code challenge, and sends the code challenge and code challenge method with the authorization request to the Authorization Endpoint.
   ```
   https://authorization_endpoint_URL?
   response_type=code&scope=openid&client_id=clientID&state=value&redirect_uri=redirect_URL&nonce=value&code_challenge=
   S256
   ```
2. Authorization Server at Authorization Endpoint records code challenge and code challenge methods, and generates authorization code.
3. The client sends the authorization code and code verifier to the Access Token Endpoint.
4. Authorization Server at Access Token Endpoint verifies that the request contains the code verifier parameter. Then, it validates the code verifier value with the code challenge and code challenge method that was sent in the authentication request at the authorization endpoint.

Note: Example of a code

Detailed Examples of Authorization Endpoints

This section includes detailed examples of the OAuth2 Authorization Endpoints.

See the following topics:

- Supported Authorization Flows
- Discovery Profile of Provider metadata

Authorization Flows Supported

You can configure Symantec Identity Security Platform with OAuth to authenticate users and generate token for native and web applications using Authorization Code flow:

- Authorization Code Flow - returns the tokens from Token Endpoint. Use Authorization Code Flow for clients that can secure their communication with Authorization Server.

The following table differentiates how IDSP works in Authorization Code Flow and Implicit Flow:

	Authorization Code Flow	Implicit Flow
Authorization Server		
Returns Authorization Code at Authorization Endpoint	Yes	No
Returns ID token and Access Token at Authorization Endpoint	No	Yes
Returns ID token and Access Token at Access Token Endpoint	Yes	No
Authenticates Client	Yes	No
Supports PKCE with Public Clients	Yes	No
Authentication Request Parameters		
client_id	Mandatory	Mandatory
redirect_uri	Mandatory	Mandatory

response_type	Mandatory
	Must be
	`code`
		Mandatory
		Can be
		`id_token`
		or
		`id_token token`
		or
		token

scope	Mandatory One value must be `openid - to return id_token`	Mandatory One value must be openid `- to return id_token`
nonce	Optional	Mandatory
state	Optional	Optional
code_challenge & code_challenge_method	Optional	Not Applicable
Response Code		
Authorization response contains code	Yes	No
Authorization response contains access_token	No	Yes for id_token token, token No for id_token
Authorization response contains ID_token	No	Yes
ID_token contains `at_hash`	No	Yes
ID_token contains `nonce`	Yes if `nonce` was sent in the authentication request	Yes

Authentication using Authorization Code Flow

VIP Authentication Service authenticates users using Authorization Code Flow that uses *code* as the response type. Authorization Code Flow returns the tokens from Token Endpoint.

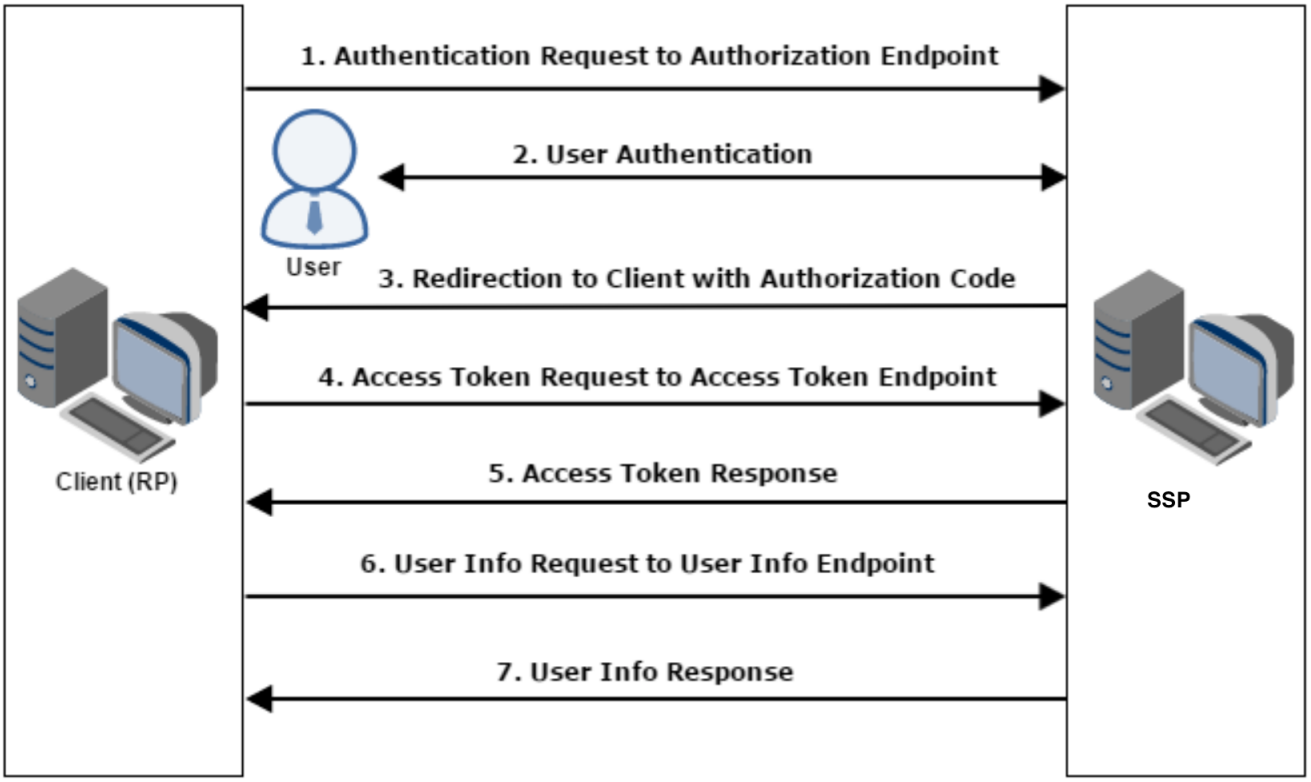
This topic includes the following sections:

- Authorization Endpoint
- Token Endpoint
- Refresh Token
- Introspection Endpoint
- UserInfo Endpoint
- JWKS Endpoint

How does Authorization Code Flow work?

The following diagram details the flow:

Figure 17: Authentication using Authorization Code Flow



The Authorization Code Flow works as follows:

1. Client sends an authentication request to Authorization Endpoint.
2. a. Authorization service redirects the user browser to a signin application indicated by the "flowUrl" application property, to perform the user authentication ceremony. If the "flowUrl" property is not specified, then a default, out-of-the-box signin application is used to orchestrate the signin flow.
   b. After the signin app has successfully completed the signin, the signin app redirects the user browser (using /AuthComplete request) back to the Authorization service, which in turn redirects to the callback URL of the application.
3. Authorization Endpoint redirects the user to the Client along with an authorization code.

4. Client sends the authorization code to Token Endpoint and requests for Access Token.
5. Token Endpoint authorizes the request and generates Access Token and ID Token. It also generates a Refresh Token, if you pass "offline_access" as one of the scopes in the "/token" request endpoint.
6. Client sends Access Token to UserInfo Endpoint.
7. UserInfo Endpoint authenticates the request and returns claims about the user

Authorization Endpoint

Authorization Endpoint handles the OAuth2/OpenId protocol part of the application's request for user authentication. It redirects to the signin application to initiate and complete the user signin ceremony, and constructs the necessary identity artifacts (Identity token) to be returned back to the requesting application.

The application is registered in Symantec Identity Security Platform as an Application with an OAuth client and other OAuth-specific artifacts required to execute an authentication request.

Supported Parameters

The authentication request supports the following parameters:

- **response_type**
  Defines the authorization processing flow that must be used. This value must be **code**.
- **scope**
  Defines the scope of the accessing request. You can specify multiple values that are separated by space. **openid** must be one of the values.
  Example: openid email phone
- **client_id**
  Defines the Client ID of the client that was registered with VIP Authentication Service.
- **state**
  (Optional) Defines an opaque value that the client uses to maintain state between the request and callback. The authorization server includes this value when redirecting the request back to the client.
- **redirect_uri**
  Specifies the URI to which the authentication response must be sent. This value must be the URI that was specified during client registration.
  The URI can begin with one of the following schemes:
  – HTTP
  – HTTPS
  – Custom URI in the *scheme* `://` format
    where *scheme* = alphabet *(alphabet|digit|+|-|.)
    Examples:

```
https://sample.ca.com/example/redirect_uri
OR
oidc.example.sso://sample.com/example/redirect_uri
```

    For Public Clients, use the HTTPS or *scheme*:// format.
- **nonce**
  (Optional) Specifies a value that associates a client session with an ID Token. The value must contain sufficient entropy to prevent attacks. VIP Authentication Service stores nonce from the authorization request and sends the same value as a standard nonce claim in ID Token.
- **code_challenge**

Defines the plain or hash value of the code verifier based on the code challenge method. Authorization Server compares this value with the code verifier before generating the token.

- **code_challenge_method**
  Defines how the code challenged is generated. The value is either **Plain** or **S256**. If only a code challenge is sent in the authentication request, the code challenge method defaults to Plain.

Supported HTTP Methods

Authorization Endpoint supports GET and POST methods.

GET

For GET, the authorization request contains an authorization endpoint URL and required query parameters in the following format:

```
 https://authorization_endpoint_URL?
response_type=code&scope=scope_value&client_id=clientID_value&state=state_value&redirect_uri=redirect_URL_value&nonce=n
 or S256 value
```

Example Request and Responses Request

Request: GET request.

```
 https://iam.example.com/<tenant-name>/oauth2/v1/authorize?response_type=code&scope=openid
%20address&client_id=c633b3f5-378d-4a22-bfa1-a0aaaa3cd799&state=UoS53-qFk0oI43icRjURWpk0OiM&redirect_uri=https
%3A%2F%2Fsample.ca.com%2Fexample%2Fredirect_uri&nonce=IkfJuXy8Sseq_8L2Fq5aH2MVUYCUV7Khm1ApnLysLmU
```

Responses

Response 1: Successful response

```
Successful response

HTTP/1.1 302 Found
Date: Thu, 08 Dec 2016 04:31:17 GMT
Server: Apache/2.4.18 (Win64) OpenSSL/1.0.2g-fips mod_jk/1.2.41
Location: https://sample.ca.com/example/redirect_uri?
code=MmU5Nzk5ZDEtNjUxMC00NjZmLTk5YzAtMWJmY2FlZDQzYjFkLUxkT1N6NldUUCtmdlZ3cmIyeHFqRHl4c3FPVT0=&state=UoS53-
qFk0oI43icRjURWpk0OiM&nonce=IkfJuXy8Sseq_8L2Fq5aH2MVUYCUV7Khm1ApnLysLmU
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive
Transfer-Encoding: chunked
Response 2
```

Response 2: Invalid response:

```
 HTTP/1.1 302 Found
Location: https://sample.ca.com/example/redirect_uri?
error=invalid_request
&error_description= error
Unsupported%20response_type%20value
&state=af0ifjsldkj
```

Error Codes and Messages

Authorization Endpoint returns an error in response to an authentication request if the authentication fails or the user denies the request. When an error occurs, an error response with error, error description, and state (if included in the original request) is appended to the query string and sent to Client in redirect_uri. Authorization Endpoint displays the following errors and error descriptions:

Note:

- HTTP Status Code 302 indicates that the error code and description are sent to redirect_uri.
- HTTP Status Code 400 and HTTP Status Code 500 indicate that the error message is displayed in the web page after failing at , and that the Administrator must verify the VIP Authentication Service log and trace files.

Scenario	Error Code	Error Description	HTTP Status Code
The scope parameter is missing, or its value is empty in the authentication request	invalid_request	Missing or invalid scope	302
The value of scope is not *openid*	invalid_request	Missing or invalid scope	302
The response_type parameter is missing, or its value is empty in the authentication request	invalid_request	Missing or invalid response_type	302
The value of response_type is not *code*	unsupported_response_type	Missing or invalid response_type	302
The client_id of a disabled client is specified in the authentication request	unauthorized_client	Invalid client_id	400
The client_id parameter is missing, or its value is empty or invalid in the authentication request	invalid_request	client_id is invalid or missing	400
The redirect_uri parameter is missing or its value is empty in the authentication request, or its value is not matching with the value configured in the client configuration	invalid_request	redirect_uri is invalid or missing	400
Authorization Server configuration error	server_error	Internal Server Error	500
Authorization Code generation error	access_denied	Unauthorized user	302

Token Endpoint

Authorization Server uses the authorization code to generate access and identity tokens. . Client sends Authorization Code in a token request to Token Endpoint using **authorization_code** as the grant_type. If the client type is confidential, client must authenticate at Token Endpoint using the authentication method specified during client registration.

Token Endpoint Process

Authorization Server performs the following tasks at Token Endpoint:

1. Validates that Client uses the registered client credentials.
2. Verifies that the Authorization Code is valid and is unused.
3. Verifies that the Authorization Code was issued in response to an OpenID Connect authentication request.
4. Verifies that the **redirect_uri** parameter value is identical to the **redirect_uri** parameter value that was sent in the initial authorization request. It returns an error if the redirect_uri parameter value is not present when there is only one registered redirect_uri value.
5. If the validation is successful, it generate a successful response that includes an Access Token and ID Token . It also generates a Refresh Token if you pass "offline_access" as one of the scopes in the "/token" request endpoint.
6. If you selected the **Send User Information in ID Token** option during client registration, sends the user claims information in the ID Token.

Supported Parameters

The token request supports the following parameters:

- **grant_type**
  Specifies the code flow. This value must be **authorization_code**.
- **code**
  Specifies Authorization Code.
- **assertion**
  Specifies id_token.
- **client_assertion**
  Specifies the assertion being used to authenticate the client. Specific serialization of the assertion is defined by profile documents.
- **client_assertion_type**
  Specifies the format of the assertion as defined by the authorization server. The value is an absolute URI.
- **redirect_uri**
  (Optional) Verifies if an Authorization Code was generated for the redirect_uri defined in the initial authorization request.

Supported HTTP Methods

Token Endpoint supports the POST method.

The following is the token request format for an application that is Public:

```
POST
token_endpoint
HTTP/1.1
Host:
host_name
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code&code=
authorization_code
```

The following is the token request format for a Confidential application that uses the Basic authentication type.

```
POST
token_endpoint
HTTP/1.1
Host:
host_name
```

```
Content-Type: application/x-www-form-urlencoded
Authorization: Basic
encoded(client_id:client_secret)
grant_type=authorization_code&code=
authorization_code
```

The Client ID and Client Secret values can be specified in one of the following formats in the authorization header for a Basic client authentication:

- Base64 encoded (Client ID:Client Secret)
- Base64 encoded (URL encoded Client ID:URL encoded Client Secret)

The following is the token request format for a Confidential application that uses the POST authentication type:

```
POST
token_endpoint
HTTP/1.1
Host:
host_name
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code&code=
authorization_code
&client_id=
client_id
&client_secret=
client_secret
```

Example Requests and Responses

Request 1: Token request format for a Public Client

```
POST <tenant-name>/oauth2/v1/token HTTP/1.1
Host: iam.example.com
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA&redirect_uri=https%3A%2F%2Fsample.ca.com%2Fexample
%2Fredirect_uri
```

Request 2: Token request format for a Confidential application that uses the Basic authentication type.

```
POST <tenant-name>/oauth2/v1/token HTTP/1.1
Host: wa.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA&redirect_uri=https%3A%2F%2Fsample.ca.com%2Fexample
%2Fredirect_uri
```

Request 3: Token request format for a Confidential application that uses the POST authentication type.

```
POST <tenant-name>/oauth2/v1/token HTTP/1.1
Host: wa.example.com
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA&redirect_uri=https%3A%2F%2Fsample.ca.com%2Fexample
%2Fredirect_uri
&client_id=c633b3f5-378d-4a22-bfa1-a0aaaa3cd799
&client_secret=SplxASdfcdslOBeZQQYbYS6WxSbIA
```

Responses Response 1: Successful response

```
200 OK
Cache-Control: no-store
Content-Type: application/json; charset=UTF-8
```

```
Pragma: no-store
{
"access_token": "TkE+qlbu7dcHibOutW7QEcESdpE=",
"id_token": "yI6ICJodHRwOi8vc2Vyd....",
"expires_in": 120,
"token_type": "Bearer"
}
```

Response 2: Invalid response

```
400 Bad Request
Cache-Control: no-store
Content-Type: application/json; charset=UTF-8
Pragma: no-store
{
"error": "invalid_grant",
"error_description": "Invalid authorization code"
}
```

Error Codes and Messages Token Endpoint uses the following error codes and descriptions:

Scenario	Error Code	Error Description	HTTP Status Code
Token Endpoint received a GET request	invalid_request	GET request is not allowed	400
Token Endpoint received a POST request with the HTTP protocol	invalid_request	The request is not secure	400
The *code* request parameter is missing in the request body	invalid_request	Specify a value for Code	400
The grant_type request parameter is missing or its value is not *authorization_code*	invalid_request	Specify authorization_code as grant_type	400
Duplicate client credentials are sent with both the authorization header, and client_id and/or client_secret in the POST body	invalid_request	The request body contains both the authorization header and client_id/ client_secret	400
Authorization Code is invalid	invalid_grant	Invalid code	400
Authorization Code expired	invalid_grant	Code expired	400
The client_id configuration is missing	invalid_client	Failed to retrieve client info	401
The client authentication failed because of invalid inputs	invalid_client	Client authentication failed - (with invalid inputs)	401
The client authentication failed because authorization header was expected but received client_id and client_secret in POST body	invalid_request	Invalid client credentials	400
The redirect_uri value in the request parameter is different from the one sent in the initial request Or The redirect_uri value is modified or deleted in the client configuration during flows	invalid_grant	Invalid redirect_uri	400

Any exceptions such as failure to decrypt client_secret stored in Policy Server or failure to perform CRUD operations on session store due to connection issues	invalid_request	Internal server error	400

Refresh Token

For information, see Refresh Token.

Introspection Token Endpoint

Use the improved Introspection Endpoint using POST method in the following format to validate the received Access Token or Refresh Token, and view the scopes associated with the token:

```
POST introspection_endpoint HTTP/1.1
Host: ah_hostname
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Basic base64_encoded(client_id and client_secret)
token=token&token_type_hint=token_type
```

token

Defines the token. Enter the value of the access_token or refresh_token that you received from Token Endpoint.

(Optional) token_hint_type

Defines a hint about the token type that you want to verify. Authorization Server uses this hint in simplifying the search for the token. Enter **access_token** or **refresh_token** depending on the token value.

Client ID and Client Secret values must be sent in the authorization header:

- Base64 encoded (Client ID:Client Secret)
- Base64 encoded (URL encoded Client ID:URL encoded Client Secret)

Example Requests and Response

Request 1: Request with token and token_hint_type

```
POST /<tenant-name>/oauth2/v1/introspect  HTTP/1.1
Host: example.iam.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
token=2YotnFZFEjr1zCsicMWpAA&token_type_hint=access_token
```

Request 2: Request with only token

```
POST /<tenant-name>/oauth2/v1/introspect  HTTP/1.1
Host: example.iam.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
token=2YotnFZFEjr1zCsicMWpAA
```

Responses Response 1: Valid token response

```
HTTP/1.1 200 OK
Content-Type: application/json
{
"active": true,
"client_id": "bb8293f6-ceef-4e7a-90c8-1492e97df19f",
```

```
"token_type": "refresh_token",
"scope": "openid profile",
"sub": "cn=odicuser,dc=example,dc=com",
"exp": "86400",
"iat": "1506513918",
"iss": "https://example.iam.com"
}
```

active

Specifies the status of the token. The value true indicates that the token is valid and has not expired. The value false indicates that the token is invalid or has expired.

client_id

Identifies the client that requested the token.

token_type

Identifies the token type.

scope

Identifies the scopes that are associated with the token.

sub

Identifies the user who authorized the request.

exp

Identifies the time period in seconds in which the token expires.

iat

Identifies the time in seconds when the token was originally issued.

iss

Identifies the authorization provider that issued the token.

Response 2: Response for invalid or expired token

```
HTTP/1.1 200 OK
Content-Type: application/json
{
"active": false
}
```

Response 3: Error response

```
HTTP/1.1 400 Bad Request Content-Type:application/json;
charset=UTF-8 Cache-Control:
no-store Pragma: no-cache
{
"error":"invalid_request"
"error_description": "token is missing in the request"
}
```

Request for Access token introspection with client assertion as client credentials:

```
curl -k --request POST \
--url 'https://example.com/default/oauth2/v1/introspect' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'token=eyJ4NXQjUzI1NiI6ImNJbUNYN1FyRy1BUUpkWWZ......'
--data-urlencode 'client_id=7b447d36-cac4-4a9e-b093-42f0d553d9f4' \
```

```
--data-urlencode 'client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer' \
--data-urlencode 'client_assertion=ewogICJ4NXQjUzI1NiI6ICI4bkRSWURaUF9xcENwUz.....'

Decoded Token
{
  "client_tid": "default",
  "sub": "2db6e0b1-8f69-4f1c-b308-6ddacdca76f1",
  "ver": "1.0",
  "iss": "https://example.com/default/",
  "azpacr": "1",
  "tok_type": "AT",
  "tid": "default",
  "aud": "https://example.com/default/",
  "nbf": 1592163258,
  "sub_type": "CLIENT",
  "azp": "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
  "scope": "urn:iam:t.configs urn:iam:t.authenticate",
  "exp": 1592166858,
  "iat": 1592163258,
  "client_name": "testapp",
  "jti": "bbb543b2-3e4b-4be1-bcd7-5dafc3170186"
}
```

Responses

```
{
  "active" : true,
  "scope" : "urn:iam:t.configs urn:iam:t.authenticate",
  "sub" : "2db6e0b1-8f69-4f1c-b308-6ddacdca76f1",
  "iss" : "https://example.com/default/",
  "jti" : "bbb543b2-3e4b-4be1-bcd7-5dafc3170186",
  "aud" : [ "https://example.com/default/" ],
  "exp" : 1592166858000,
  "iat" : 1592163258000,
  "nbf" : 1592163258000,
  "client_id" : "6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
  "token_type" : "Bearer"
}
```

See Client Assertion for more information. **Error Codes and Messages** The following table lists the error codes displayed:

Scenario	error_code	HTTP Status Code
• Required inputs are missing in the request • Authorization header, and Client ID and Client Secret exist in the request • Unable to get client ID from the authorization header or form parameters of the request	invalid_request	400
Client credentials are invalid	invalid_client	401
The GET request is received	invalid_request	405
Other server issues	Internal Server Error	500

UserInfo Endpoint

UserInfo Endpoint returns claims related to the authenticated user. Client sends a request to UserInfo Endpoint along with Access Token, and UserInfo Endpoint returns the claims in the Signed JWT format. VIP Authentication Service sends claims that are scoped by the token to the Client. **Supported HTTP Methods** UserInfo Endpoint supports GET and POST methods. The GET request can be of the header or query format. The POST request is of the following format:

```
POST
userinfo_endpoint
 HTTP/1.1
Host:
host_name
Content-Type: application/x-www-form-urlencoded
access_token=
access_token
```

Examples Requests and Responses

Requests

Request 1: GET request in header format.

```
GET /<tenant-name>/oauth2/v1/userinfo HTTP/1.1
Host: example.iam.com
Authorization: Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6IjF
```

Request 2: GET request in query format.

```
/<tenant-name>/oauth2/v1/userinfo?access_token=eyJhbGciOiJSUzI1NiIsImtpZCI6IjF
```

Request 3: POST request

```
POST /<tenant-name>/oauth2/v1/userinfo HTTP/1.1

Host: example.iam.com
Content-Type: application/x-www-form-urlencoded
access_token=sfthwYNeo%2BvOOJses6KlKwShZF0=
```

Responses

Response 1: Successful response

```
HTTP/1.1 200 OK
Content-Type: application/jwt
eyJ4NXQjUzI1NiI6ImNJbU.........
```

Response 2: Successful response with encrypted user info

```
HTTP/1.1 200 OK
Content-Type: application/jwt
NqeGpe-gccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7Tpd
```

Response 3: Invalid response

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: error="invalid_token",
error_description="The Access Token expired"
```

Error Codes and Messages

Error codes are displayed in the following scenarios:

Scenario	Error Code	HTTP Status Code
• Access Token is not found in either form parameter or authorization header of the request • Request content type is not application/x-www-form-urlencoded • Request have more than one Access Token • Request has both authorization bearer header and access_token in the form parameters • Request contains both header and query.	invalid_request	400
The access_token value has expired, is revoked, malformed, or invalid	invalid_token	401
Other server issues	Internal Server Error	500

JSON Web Key Set Endpoint

A read-only JWKS service is for use by someone who develops PKI-based code to perform one of two activities:

- validate signature of received artifacts
- encrypt artifacts to be sent

The main purpose of a JWKS endpoint is to provide applications with public keys as required to verify the signature of a JWT issued by the IDSP Authorization Service, or to encrypt a JWT sent to the Authorization server (such as encrypting Identity Token Hint).

The following is the format of the endpoint:

When you export the metadata of a Client, this endpoint information is available in the *jwks_uri* attribute.

```
https://<tenant-name>/default/oauth2/v1/jwks
```

Example Responses

Response 1: Successful response


```
{"keys":[{"kty":"RSA","e":"AQAB","kid":"5e35223a-58b8-4f78-b8df-f92a8bae3c0b","key_ops":
["encrypt","wrapKey","verify"],"alg":"SHA256WITHRSA","n":"vemHcoWegeZvXBAurW2Im3b-
v4r_AFwlIOTG8WfXS9qiHxc5b9p-_ol9j8oncwsM1qLv-9n2UlIqZ5UqwkSZYPyCeJf1QeXrRAJjJDhzqx485GLu69It-
avhqVVWxCKMOxE2VStuLIn4D-1VSiqhB4uGI3Gz9Hq2ar02IdG4qaCOe9IJwYB0zS_kpijEyGuCOe_VDJSGXpTCPaap-
aa7BCMUQtLUR_GUDxy5rZuYCBqgeb4YAq4Z5hXsv7VKPmpWV_ZywRDypGlZA5CVmK54x5TVHCRzBo2x-
A1ExXMhxXBxHdv3xNfxLlsIfHGmuiiWjf8K5KmzBo-n1lBu1keTIQ"}]}
```


Response 2: Invalid response


```
{"error":"invalid_request","error_description":"Failed to retrieve Configuration information."}
```


Error Codes and Messages

The following table lists the error messages that this endpoint displays:

Scenario	Error Code	HTTP Status Code
Invalid POST request	invalid_request	405
Authorization Provider is missing in the request or is invalid, or is disabled	invalid_request	400
Other server issues	Internal Server Error	500

Discovery Profile of Provider Metadata

Use the Well-known Endpoint to dynamically retrieve OAuth provider metadata including endpoints, supported scopes, claims, and authentication methods.

You can use a Well-known Endpoint to dynamically retrieve the metadata of Symantec Identity Security Platform. A Relying Party can use this endpoint to dynamically retrieve the metadata of an Authorization Provider that is configured with a Client. The following scenarios describe how a Relying Party can use the metadata:

- OAuth Clients that display multiple authentication providers to end-users send requests to OAuth provider based on the end-user choice. Such clients can use the metadata to construct requests to specific endpoints as required.
- If a Client or mobile OAuth SDK does not have the provision to configure OAuth Client information, they can use the metadata information of endpoints, claims, and scopes to build dynamic requests to receive authorization code and tokens from Symantec Authentication Service OAuth Provider.
- If a Client does not store public certificates of Symantec Authentication Service OAuth Provider, Client can use the JWKS endpoint in the metadata to retrieve the public certificates during runtime for signature verification of ID Token.

Request Format

To retrieve the metadata, send an HTTPS GET request to the following endpoint:

```
/<tenant-name>/.well-known/openid-configuration
```

Defines the name of the client whose associated Authorization Provider metadata must be retrieved along with the following client information:

- grant_types
- response_types_supported
- response_modes_supported
- scopes_supported
- claims_supported
- code_challenge_methods_supported
- token_endpoint_auth_methods_supported

Example:

```
GET /<tenant-name>/.well-known/openid-configuration HTTP/1.1
Host: ${SSP_FQDN}
```

Response Format

The metadata response from a well-known endpoint returns a JSON document in the **application/json** content type.

The following metadata is returned for a specific Client:

Parameter	Description	Values
Issuer	Defines the issuer that generate Claims.	https://${SSP_FQDN}/<tenant-name>/
authorization_endpoint	Defines the URL of the Authorization Endpoint.	https://${SSP_FQDN}/<tenant-name>/oauth2/
token_endpoint	Defines the URL of Token Endpoint. This value is returned only if the Authorization Code Flow is selected.	https://${SSP_FQDN}/<tenant-name>/oauth2/
device_authorization_endpoint	Defines the URL of the Device Code grant. This value is returned only if the Device Used Code grant flow is selected.	https://${SSP_FQDN}/<tenant-name>//oauth2
revoke_endpoint	Defines the URL of Revoke Token Endpoint	https://${SSP_FQDN}/<tenant-name>/oauth2/

userinfo_endpoint	Defines the URL of the User Info Endpoint.	https://${SSP_FQDN}/<tenant-name>/oauth2/
introspection_endpoint	Defines the URL of Introspect Endpoint.	https://${SSP_FQDN}/<tenant-name>/oauth2/
jwks_uri	Defines the URI of JSON Web Key Set Endpoint.	https://${SSP_FQDN}/<tenant-name>/oauth2/
scopes_supported	Specifies the list of scopes that are selected for OAuth Client during client configuration in Administrative UI.	"openid", "profile", "email", "phone", "address", "groups"
claims_supported	Specifies how the Client can obtain Claims about the End-User and the Authentication event. It also defines a standard set of basic profile Claims. Pre-defined sets of Claims can be requested using specific scope values or individual Claims can be requested using the `claims` request parameter. The Claims can come directly from the OpenID Provider or from distributed sources as well.	"aud","email","email_verified","phone_number","phone_number_verified","family_name","middle_name","given_name","iat","name", "sub","address","groups","picture","nickname","preferred_username","profile","website","zoneinfo","birthdate","updated_at","sub"
response_types_supported	Specifies the list of response types that are selected for OAuth Client during client configuration in Administrative UI.	code, token, id_token, and id_token token
response_modes_supported	Specifies the list of response modes that are supported by IDSP.	Value for Authorization Code Flow, if configure Value for Implicit Flow, if configured: fragment
grant_types_supported	Specifies the list of grant types that are supported by IDSP for the specified client.	authorization_code, implicit, urn:ietf:params:oa type:jwt-bearer, client_credentials, refresh_token, urn:ietf:params:oauth:grant-typ
subject_types_supported	Specifies the list of the Subject Identifier types that are supported by IDSP	public
id_token_signing_alg_values_supported	Specifies the list of the JWS algorithms (alg values) that are supported by IDSP to sign the claims of an ID Token.	RS256
id_token_encryption_alg_values_supported	(Optional) This is a JSON array that contains a list of JWE encryption algorithms (alg values) supported by the OP for the ID Token to encode the Claims in a JWT.	"RSA-OAEP", "RSA-OAEP-256"
id_token_encryption_enc_values_supported	(Optional) This is a JSON array that contains a list of JWE encryption algorithms (enc values) supported by the OIDC for the ID Token to encode the Claims in a JWT.	"A128GCM", "A192GCM", "A256GCM", "A HS384", "A128CBC-HS256"
userinfo_signing_alg_values_supported	Specifies the list of the JWS algorithms (alg values) that are supported by IDSP to sign the claims of a UserInfo response.	RS256
userinfo_encryption_alg_values_supported	(Optional) This is a JSON array that contains a list of JWE encryption algorithms (alg values) supported by the UserInfo Endpoint to encode the Claims in a JWT.	"RSA-OAEP", "RSA-OAEP-256"
userinfo_encryption_enc_values_supported	(Optional) This is a JSON array that contains a list of JWE encryption algorithms (enc values) supported by the UserInfo Endpoint to encode the Claims in a JWT.	"A128GCM", "A192GCM", "A256GCM", "A HS384", "A128CBC-HS256"
token_endpoint_auth_methods_supported	This parameter is returned only when client type is confidential.	client_secret_basic, client_secret_post, "privat
display_values_supported	Specifies the display parameter that is supported by OP for this client.	page
code_challenge_methods_supported	Defines the supported code challenge methods.	plain, S256

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/oauth2/v1/authorize`
  should look similar to:
  `https://example.com/default/oauth2/v1/authorize`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Sample Request and Response

Request

```
GET /<tenant-name>/.well-known/openid-configuration HTTP/1.1
Host: ${SSP_FQDN}
```

Response

```
HTTP/1.1 200 OKContent-Type: application/json
```


```
{
"keys": [
{
"kty": "RSA",
"x5t#S256": "btn1jH3EjLwP6jcWWMPK9-t0rnFMM0eEIUhYym7xRnE",
"e": "AQAB",
"use": "sig",
"x5t": "NmopuEBSpLIdTSlmZbhDP_TLUsY",
"kid": "SSP-SigningKey",
"x5c": [
"MIIDjjCCAnagAwIBAgIBADANBgkqhkiG9w0BAQsFADBTMQswCQYDVQQGEwJJTj.....0nq+BwPNNY5raGVTY0F1vMaJwn2wFKUi6fag=",
"MIIDeTCCAmGgAwIBAgIJALViYpjBt+/QMA0GCSqGSIb3DQEBCwUAMFMxCzAJBg.....M+sQArQh9P+mdmCZASr+DNdewXnBLGr483Hc="
],
"key_ops": [
"verify",
"sign"
],
"alg": "RS256",
"n": "32A4qjqY3M_UKxD3GR0rQuCH927_6-Z7qszjEGesEKq2J_zLdv_zOxx.......DQfv4_5Q"
},
{
"kty": "RSA",
"x5t#S256": "btn1jH3EjLwP6jcWWMPK9-t0rnFMM0eEIUhYym7xRnE",
"e": "AQAB",
"use": "enc",
"x5t": "NmopuEBSpLIdTSlmZbhDP_TLUsY",
"kid": "EncryptionKey",
"x5c": [
"MIIDjjCCAnagAwIBAgIBADANBgkqhkiG9w0BAQsFADBTMQswCQYDVQQGEwJJTj.....0nq+BwPNNY5raGVTY0F1vMaJwn2wFKUi6fag=",
"MIIDeTCCAmGgAwIBAgIJALViYpjBt+/QMA0GCSqGSIb3DQEBCwUAMFMxCzAJBg.....M+sQArQh9P+mdmCZASr+DNdewXnBLGr483Hc="
```

```
],
"key_ops": [
"decrypt",
"encrypt"
],
"alg": "RS256",
"n": "32A4qjqY3M_UKxD3GR0rQuCH927_6-Z7qszjEGesE......"
}
]
}
```

Error Codes

The following table lists the error codes that Symantec Authentication Service returns:

Condition	HTTP Status Code
Request URL method is not GET	405

Making the First REST API call

Learn how to make your first API call by creating the application entity to begin modeling the application integration with Symantec Identity Security Platform.

Overview

REST APIs provide a way to integrate IDSP with custom applications that support REST.

This enables you to:

- Delegate authentication management to IDSP
- Develop custom login pages for your own applications
- Fully configure IDSP to behave exactly the way your application deployment and functionality requires

NOTE
We recommend you to get familiar with the REST Architecture Style.

NOTE
Requirements for OpenShift Environment

In the OpenShift environment, when invoking tenant-specific APIs, it is mandatory to provide the "X-TENANT-NAME" header in the API request, else the API request fails with "400 BAD REQUEST" status.

Key Points:

- The tenant identified by "X-TENANT-NAME" must exist in the system
- SSL certificate must be issued for the SSP_FQDN host
- All tenant-specific API clients must include the "X-TENANT-NAME" header in their requests

For more information, see Deploying Ingress Controller on Openshift

Application Resources

Any application invoking a IDSP REST API must be represented in the solution by a dedicated \"application" resource. This resource represents the real application you are integrating with the solution. With this Application entity created, you are expected to:

- Know what REST API calls your application must invoke in order to implement the desired functionality
- Obtain security credentials (Client Id and Secret) for your application to acquire an Access Token through the OAuth2 protocol
- Use acquired Access Token to invoke the desired REST API

Once this application resource has been defined, the developer would use IDSP REST APIs to accommodate two kinds of use cases:

- **Administration** - Developers integrate with IDSP for the purpose of performing administrative tasks. For example, to manage authentication policies, configuring configuration settings, etc. These are often called IDSP Control Plane APIs.
- **Security** - Developers integrate with IDSP to request security services. For example, to have the solution perform user authentication through browser-based OpenID Connect API or programmatic /authenticate REST API. These are often called IDSP Data Plane APIs.

In this tutorial, we assume the Administration use case, and using IDSP REST API to register an application capable of performing administrative tasks such as managing Authentication Policies. The next tutorial will showcase the security use case of using IDSP APIs to authenticate your application's users.

Get Access Token to use IDSP REST APIs

Until the Admin UI is available, we use the pre-seeded Default Client to obtain an Access Token using OAuth's "Client Credential" flow.

1. Make a note of the Client's Id and Secret obtained from the Kubernetes secret representing the Default Tenant Client.
   - Client Id:
     ```
     kubectl get secret ${RELEASENAME}-ssp-secret-defaulttenantclient -n ${NAMESPACE} -o
      jsonpath="{.data.clientId}" | base64 --decode; echo
     ```
   - Client Secret:
     ```
     kubectl get secret ${RELEASENAME}-ssp-secret-defaulttenantclient -n ${NAMESPACE} -o
      jsonpath="{.data.clientSecret}" | base64 --decode; echo
     ```
2. Make a note of the ingress FQDN hosting the IDSP Service URL:
   - Ingress Host FQDN:
     ```
     kubectl get ingress -n <namespace>
     ```
3. Open a command prompt and enter the following cURL command to acquire the admin access token, substituting the ServiceIngressFQDN and clientId/clientSecret parameters:

   Token Request:
   ```
   curl -k -X POST https://<ServiceIngressFQDN>/default/oauth2/v1/token -u '<client-id>:<client-secret>' -d
    'grant_type=client_credentials&scope=urn:iam:myscopes'
   ```
   Token Response:
   ```
   {
     "access_token" : "eyJ4NXQjUzI1NiI6ImNJbUNYN1FyRy1BU......",
     "token_type" : "Bearer",
     "expires_in" : 3600
   }
   ```
   The response contains the Access Token, token_type, and expires_in. The Access Token represents your application client's requested level of access for the REST API calls it is making. This token is encoded following the JSON Web

Token (JWT) standard. To check the JWT token, copy the "access_token" and introspect its content using the jwt debugger.

Make a IDSP REST API call to create an Application Resource

Using the just acquired admin access token, the next step is to create an Application resource representing your application.

1. Copy the following example Application resource definition into "myfirstapp.json".
   Application Resource JSON

```
{
    "status": "active",
    "name": "MyFirstApp",
    "description": "My first app",
    "clientType": "CONFIDENTIAL",
    "allowedOpenIDScopes": [
        "openid",
        "email",
        "profile"
    ],
    "redirectURIs": [
        "https://www.myfirstapp.com"
    ],
    "allowedGrantTypes": [
        "client_credentials",
        "authorization_code",
        "urn:ietf:params:oauth:grant-type:jwt-bearer"
    ],
    "allowedOperations": [
        "introspect"
    ],
    "userInfoEndpointResponseFormat": "PLAIN_JSON"
}
```

2. Open a command prompt and enter the following `cURL` command to create the Application resource, substituting the ServiceIngressFQDN and AccessToken parameters:
   Application Create Request

```
curl -k -X POST https://<ServiceIngressFQDN>/default/admin/v1/Apps  -H 'Content-Type: application/json'  -
d '@myfirstapp.json'  -H 'Authorization: Bearer <AccessToken>'
```

   Application Create Response

```
{
    "status": "active",
    "name": "myfirstapp",
    "description": "My First App",
    "clientType": "CONFIDENTIAL",
    "allowedOpenIDScopes": [
        "openid",
        "profile",
        "email"
    ],
    "redirectURIs": [
        "https://www.myfirstapp.com"
    ],
    "allowedGrantTypes": [
```

```
            "authorization_code",
            "urn:ietf:params:oauth:grant-type:jwt-bearer"
        ],
        "userInfoEndpointResponseFormat": "PLAIN_JSON",
        "allowedOperations": [
            "introspect"
        ],
        "appId": "9047e9e8-aa9c-46eb-8950-24e2b6223fbe",
        "tenantId": "4764021a-9048-11ea-bb37-0242ac130002",
        "createdBy": {
            "principalType": "CLIENT",
            "principalId": "e719078f-3aa5-4433-a621-6fecd9a75ef5",
            "principalName": "defaulttenantclient"
        },
        "createdDateTime": "2020-06-01 07:37 AM UTC",
        "updatedDateTime": "2020-06-01 07:37 AM UTC",
        "secret": "393476f5-5722-42fb-a784-6251651e831e",
        "clientId": "8b109251-70d9-48e1-921c-fb35d3d91f0e"
    }
```


3. At a command prompt, enter the curl command to get all application resources, substituting the ServiceIngressFQDN and AccessToken parameters. The newly created applications should be one of them.

```
curl -k -X POST https://<ServiceIngressFQDN>/default/admin/v1/Apps -H 'Authorization: Bearer
 <AccessToken>'
```

You have successfully executed your first REST API call by creating the Application resource to begin integrating your application with IDSP. The next tutorial will demonstrate this application's integration with SSP to authenticate application users using the OpenID Connect protocol.

Integrating Sample Web App with Symantec Identity Security Platform

Learn how to integrate your web application with IDSP using REST APIs or OIDC flows to implement multi-factor authentication (MFA).

Purpose

The main aim of this topic is to help you make use of the Symantec Identity Security Platform APIs and integrate your web application with the IDSP to perform multi-factor authentication flows. You can integrate with the solution either by using the REST APIs or OIDC flow.

Includes the following sub-topics:

Sequence of Steps

This section helps you obtain a high level understanding of the sequence of steps you must take in order to use the IDSP APIs.

The solution comes with a default tenant client which can be used to perform the following tasks:

1. Obtain the default tenant client id and secret from the IDSP Kubernetes cluster.
2. Generate a default Tenant Access Token- This step helps you to generate a default tenant access token which is a prerequisite for creating an application in the solution.
3. Create an Application- This step helps you create a client in the solution.

4. Create a Policy - This step lets you perform the following:
   a. Create the Authentication policy: This step helps you create an authentication policy for the application that you have created in the previous step. The authentication policy is evaluated, when the AuthN endpoint is accessed, and on successful policy evaluation, the ID Token is presented to you.
   b. Update the Authorization policy to provide the required privileges for the newly created application. This policy is evaluated to get the Access Token.
5. Generate the client access token. This helps you initiate the authentication flow to get the user Id token.
6. Upon successfully completing the authentication flow, you obtain the User ID token.
7. Using the User ID token, generate the access token. Now, that you have obtained the User Access Token, you can invoke the appropriate IDSP APIs to implement desired functionality.

These steps have been explained in the context of the Sample Web App with the help of REST APIs in the following section.

> **NOTE**
> The APIs that you access to perform any task depends upon the privileges defined in the access token. To perform specific tasks, you should have the corresponding scopes added to the access token. See Rest APIs and Associated Privileges section for specific authorization privileges required by each API.

Exploring the REST APIs through the Sample Web App

The Sample Web App is available to help you start creating your first app. The steps are carried out using the IDSP REST APIs, which are protected with an OAuth2-based security model.

Artifact Location

You can locate the demo artifacts for the Sample Web App from the Support Portal > My Downloads (Search for IDSP in the portal and select the appropriate version number).

The Sample App works out of the box, follow the deployment instructions provided here.

Steps to Deploy the Sample App

Perform these steps to deploy the Sample App in your environment:

> **NOTE**
> In the following steps, the curl commands have used <iam.example.com> as the ingress hostname which is a dummy value. Therefore, replace <iam.example.com> with your deployment's ingress hostname. You can get this value by using this command:
> *"kubectl gets ingress -n <namespace>"*

1. Use the following command to get the default tenant client id and secret from the IDSP Kubernetes cluster:
   - `kubectl get secret ${RELEASENAME}-ssp-secret-defaulttenantclient -n ${NAMESPACE} -o jsonpath="{.data.clientId}" | base64 --decode; echo`
   - `kubectl get secret ${RELEASENAME}-ssp-secret-defaulttenantclient -n ${NAMESPACE} -o jsonpath="{.data.clientSecret}" | base64 --decode; echo`
2. Use the following API to generate the default tenant access token by passing the client Id and secret in the authorization header:
   - `https://${SSP_FQDN}/{{tenantName}}/oauth2/v1/token`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/oauth2/v1/token`
  should look similar to:
  `https://example.com/default/oauth2/v1/token`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Generate Access Token Request

```
curl -L -X POST 'https://iam.example.com/default/oauth2/v1/token' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H 'Authorization: Basic
 OTc1MDljMjUtNjU0Ni00NmM0LTllNGQtMTJmYmJjYTYwZjNhOjgzNGI4ZjUxLWlMC1iYmFkLTg1NDdmMzYyM2ViNA==' \
--data-urlencode 'grant_type=client_credentials' \
--data-urlencode 'scope=urn:iam:myscopes'
```

Generate Access Token Response

```
{
 "access_token": "XQjUzI1NiI6ImNJbUNYN1FyRy",
 "token_type": "Bearer",
 "expires_in": 360
}
```

Now, you have the default Tenant Client Access Token with which you can create an application.

3. Create an application using the following API:

– `https://${SSP_FQDN}/{{tenantName}}/admin/v1/Apps`

Create Application Request

```
curl -L -X POST 'https://iam.example.com/default/admin/v1/Apps' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer Ry1BUUpkWWZ2Wlctd1BDcF' \
--data-raw '{
   "status": "active",
   "name": "Test Application",
   "description": "Default Application for Security",
   "clientType": "CONFIDENTIAL",
   "allowedOpenIDScopes": [
       "openid",
       "email",
       "profile"
   ],
   "redirectURIs": [
       "https://www.example.com"

   ],
   "allowedGrantTypes": [
       "implicit",
       "authorization_code",
       "urn:ietf:params:oauth:grant-type:jwt-bearer",
```

```
            "client_credentials"
        ],
        "allowedOperations": [
            "introspect", "onbehalfof"
        ],
        "skipIssuerAudienceForIT": true,
        "userInfoEndpointResponseFormat": "PLAIN_JSON"
    }'
```

Create Application Response

```
{
    "status": "active",
    "name": "test application",
    "description": "Default Application for Security",
    "clientType": "CONFIDENTIAL",
    "allowedOpenIDScopes": [
        "openid",
        "profile",
        "email"
    ],
    "redirectURIs": [
        "https://www.example.com",

    ]
    "allowedGrantTypes": [
        "implicit",
        "client_credentials",
        "authorization_code",
        "urn:ietf:params:oauth:grant-type:jwt-bearer"
    ],
    "userInfoEndpointResponseFormat": "PLAIN_JSON",
    "skipIssuerAudienceForIT": true,
    "allowedOperations": [
        "introspect",
        "onbehalfof"
    ],
    "appId": "a45247ea-1652-4b3b-9178-970a2caf89e5",
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "97509c25-6546-46c4-9e4d-12fbbca60f3a",
        "principalName": "defaulttenantclient"
    },
    "createdDateTime": "2020-06-11 10:37 AM UTC",
    "updatedDateTime": "2020-06-11 10:37 AM UTC",
    "secret": "b8fc0a31-2c0c-4e1f-ac3e-da298cd12819",
    "clientId": "947b2632-3283-4766-9cc9-f00a3fce6aee"
}
```

NOTE
The response payload fetches you the values for client Id, secret, and application Id for the application. Make a note of these values, as these will be used to sign in to the sample application.

4. Use the following API to create an authentication policy:
   - `https://${SSP_FQDN}/{{tenantName}}/admin/v1/AuthPolicies`

   Create Authentication Policy Request

```
curl -L -X POST 'https://iam.example.com/default/admin/v1/AuthPolicies' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer 6ImNJbUNYN1FyRy1BUUpkWWZ2Wlctd' \
--data-raw '{
   "description": "Sample policy Record ",
   "status": "active",
   "policyName": "Sample policy with channel web",
   "apps": [
       {
           "id": "a45247ea-1652-4b3b-9178-970a2caf89e5"
       }
   ],
   "rules": [
       {
           "conditions": {
               "channel": {
                   "operator": "in",
                   "value": [
                       "web"
                   ]
               }
           },
           "result": {
               "effect": "allow",
               "obligations": [
                   "PASSWORD",
                   "OTP"
               ],
               "msg": "This policy is for login in web channel"
           }
       }
   ]
}'
```

Create Authentication Policy Response

```
{
   "policyId": "8e9de6e1-8105-40da-942e-7841332d6b5b",
   "policyName": "Sample policy with channel web",
   "description": "Sample policy Record ",
   "status": "active",
   "matchAnyApp": false,
   "apps": [
       {
           "name": "Test Application",
           "id": "a45247ea-1652-4b3b-9178-970a2caf89e5"
       }
   ],
   "rules": [
       {
           "conditions": {
               "channel": {
                   "operator": "in",
                   "value": [
```

```
                        "web"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD",
                    "OTP"
                ],
                "msg": "This policy is for login in web channel"
            }
        }
    ],
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "97509c25-6546-46c4-9e4d-12fbbca60f3a",
        "principalName": "defaulttenantclient"
    },
    "createdDate": "2020-06-11 10:41:39"
}
```

5. Update the "AuthClient" Authorization policy with the application Id. To perform the update, we must first get all the authorization policies, and then update the "AuthClient" Authorization policy.
   a. Run the following endpoint to get the Authorization policies:

```
https://${SSP_FQDN}/{{tenantName}}/admin/v1/AuthZPolicies/{{authClientAuthorizationPolicyId}}
```

   Get all tenant Authorization Policy Request

```
curl -L -X GET 'https://iam.example.com/default/admin/v1/AuthZPolicies' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer XQjUzI1NiI6ImNJbUNYN1FyRy1BUUpkWWZ2Wlctd1BDcFQ0WT' \
--data-raw ''
```

   Get all tenant Authorization Policy Response

```
[
   {
       "policyId": "46010fd8-93c3-4adc-9632-b0d0726cf761",
       "policyName": "TenantAdminPolicy",
       "description": "Policy for Tenant Admin",
       "status": "active",
       "matchAnyApp": false,
       "apps": [
           {
               "name": "ssp",
               "id": "3690ac36-c805-43ef-8318-627023b16d64"
           }
       ],
       "rules": [
           {
               "conditions": {
                   "principal": {
                       "clientApp": {
                           "operator": "in",
                           "value": [
```

```
                            "7328b5c9-0ece-46d7-8758-6d08a618fb9a",
                            "059bcbbc-c8b6-467d-85d0-6d15f5a5c0f7"
                        ]
                    }
                }
            },
            "result": {
                "msg": "This is Tenant Admin authorization policy",
                "privileges": [
                    "urn:iam:t.apps",
                    "urn:iam:t.authenticate",
                    "urn:iam:t.configs",
                    "urn:iam:t.ldapconfigs",
                    "urn:iam:t.users",
                    "urn:iam:t.signingkey",
                    "urn:iam:t.certs",
                    "urn:iam:t.authzpolicies",
                    "urn:iam:t.authpolicies",
                    "urn:iam:t.groups",
                    "urn:iam:t.risk",
                    "urn:iam:t.userdata"
                ]
            }
        }
    ],
    "createdDate": "2020-06-10 05:36:21",
    "updatedDate": "2020-06-10 05:36:21"
},
{
    "policyId": "dbb932c7-c926-43df-a35c-f93a71c43f4e",
    "policyName": "AuthClientPolicy",
    "description": "Policy for Authentication Client",
    "status": "active",
    "matchAnyApp": false,
    "apps": [
        {
            "name": "ssp",
            "id": "3690ac36-c805-43ef-8318-627023b16d64"
        }
    ],
    "rules": [
        {
            "conditions": {
                "principal": {
                    "clientApp": {
                        "operator": "in",
                        "value": [
                            "2d93ffc1-ac22-48f7-8f80-206535f40afb"
                        ]
                    }
                }
            },
            "result": {
```

```
                    "msg": "This policy allows clients to call authentication APIs",
                    "privileges": [
                        "urn:iam:t.authenticate"
                    ]
                }
            }
        ],
        "createdDate": "2020-06-10 05:36:21",
        "updatedDate": "2020-06-10 05:36:21"
    }
                ]
```

b. Run the following API to grant "clientApp" to the "AuthClientPolicy":

```
https://${SSP_FQDN}/{{tenantName}}/admin/v1/AdminRolesDelegator
```

Update Authorization Policy Request

```
curl --location 'example.com/default/admin/v1/AdminRolesDelegator' \
 --header 'Content-Type: application/json' \
 --header 'Authorization: Bearer eyJ4NXQ....' \
 --data '{
     "id": "3805d28b-f980-46de-a497-fad560433623",
     "principal": {
         "clientApp": {
             "operator": "in",
             "value": [
                 "a45247ea-1652-4b3b-9178-970a2caf89e5"
             ]
         }
     }
 }'
```

The following is the Authorization Policy Response:

```
{
   "id": "dbb932c7-c926-43df-a35c-f93a71c43f4e",
   "name": "AuthClientPolicy",
   "displayName": "Authentication Client",
   "description": "Perform authentication-related operations as required to implement a custom signin
 app.",
   "principal": {
       "clientApp": {
           "operator": "in",
           "value": [
               "a45247ea-1652-4b3b-9178-970a2caf89e5",
           ]
       }
   },
   "isGrantableClientOnly": true
   }
```

6. (Optional) Perform this step to configure an LDAP User Store, if you have not configured it earlier. Run the following endpoint to configure an LDAP User Store:

```
https://${SSP_FQDN}/tenantName/admin/v1/LDAPconfigs
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/LDAPconfigs`
  should look similar to:
  `https://example.com/default/admin/v1/LDAPconfigs`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Configure LDAP User Store Request

```
curl -L -X GET 'https://iam.example.com/default/admin/v1/LDAPconfigs' \
-H 'Authorization: Bearer <token>'
```

Configure LDAP User Store Response


```
[
    {
        "tenantId": "4764021a-9048-11ea-bb37-0242ac130002",
        "configName": "TestConfig",
        "configDescription": "CA DIR k8s with default data",
        "sslEnabled": "NO",
        "certificate": null,
        "certificateAlias": null,
        "ldapQueryStart": "(&(cn=",
        "ldapQueryEnd": ")(objectClass=*))",
        "attributes": "uid,telephoneNumber,employeeType,mail,displayName,givenName,cn,sn,memberof,title",
        "objectClasses": "string",
        "groupObjectClasses": "groupOfNames",
        "hostName": "iam.example.com",
        "portNumber": "31568",
        "baseDN": "o=democorp,c=us",
        "groupBaseDN": "o=democorp,c=us",
        "userDN": "cn=DominicMAJOR,ou=Administration,ou=Corporate,o=democorp,c=US",
        "bindPassword": "/OTqCj2yPWVvhl6AjpJ/DQ==",
        "attributeMapping":
 "name=cn,family_name=sn,given_name=givenName,middle_name=givenName,nickname=displayName,preferred_username=displayN
 status=employeeType,memberof=memberof",
        "groupSearchMappingAttribute": "member",
        "useMemberOf": "yes",
        "namingAttribute": "cn",
        "passwordResetURL": "http://userPasswordResetURL",
        "createdDateTime": "2020-06-30 15:40 PM UTC",
        "updatedDateTime": "2020-06-30 15:40 PM UTC",
        "createdBy": {
            "principalType": "CLIENT",
            "principalId": "f4bcb936-769f-4377-9e76-276083d1958e",
            "principalName": "defaulttenantclient"
        },
        "ldapConfigId": "3ec2ecfa-d88b-49cd-b50b-e4dba6094a9d"
    }
]
```

7. (Optional) Perform this step to create an LDAP User Store.

   NOTE

   Check if a user store already exists. If yes, then skip this step.

Configure LDAP User Store Request

```
curl -L -X POST 'https://iam.example.com/default/admin/v1/LDAPconfigs' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <token>' \
--data-raw '{
    "configName": "TestConfig",
    "configDescription": "CA DIR k8s with default data",
    "sslEnabled": "NO",
    "ldapQueryStart": "(&(cn=",
    "ldapQueryEnd": ")(objectClass=*))",
    "attributes": "uid,displayName,telephoneNumber,mail,memberof",
    "objectClasses": "string",
    "hostName": "iam.example.com",
    "portNumber": "31888",
    "baseDN": "o=democorp,c=us",
    "groupBaseDN": "o=democorp,c=us",
    "groupObjectClasses": "groupOfNames",
    "groupSearchMappingAttribute": "member",
    "useMemberOf": "yes",
    "namingAttribute": "cn",
    "userDN": "cn=DominicMAJOR,ou=Administration,ou=Corporate,o=democorp,c=US",
    "bindPassword": "password",
    "attributeMapping":
 "name=cn,family_name=sn,given_name=givenName,middle_name=givenName,nickname=displayName,preferred_username=displayN
 status=employeeType,memberof=memberof",
    "passwordResetURL": "http://userPasswordResetURL"
}'
```

Configure LDAP User Store Response

```
{
    "tenantId": "4764021a-9048-11ea-bb37-0242ac130002",
    "configName": "TestConfig",
    "configDescription": "CA DIR k8s with default data",
    "sslEnabled": "NO",
    "certificate": null,
    "certificateAlias": null,
    "ldapQueryStart": "(&(cn=",
    "ldapQueryEnd": ")(objectClass=*))",
    "attributes": "uid,telephoneNumber,employeeType,mail,displayName,givenName,cn,sn,memberof,title",
    "objectClasses": "string",
    "groupObjectClasses": "groupOfNames",
    "hostName": "iam.example.com",
    "portNumber": "31888",
    "baseDN": "o=democorp,c=us",
    "groupBaseDN": "o=democorp,c=us",
    "userDN": "cn=DominicMAJOR,ou=Administration,ou=Corporate,o=democorp,c=US",
    "bindPassword": "/OTqCj2yPWVvhl6AjpJ/DQ==",
    "attributeMapping":
 "name=cn,family_name=sn,given_name=givenName,middle_name=givenName,nickname=displayName,preferred_username=displayN
 status=employeeType,memberof=memberof",
```

```
        "groupSearchMappingAttribute": "member",
        "useMemberOf": "yes",
        "namingAttribute": "cn",
        "passwordResetURL": "http://userPasswordResetURL",
        "createdDateTime": "2020-06-23 04:24 AM UTC",
        "updatedDateTime": "2020-06-23 04:24 AM UTC",
        "createdBy": {
            "principalType": "CLIENT",
            "principalId": "006eddf2-ea8d-46b6-8c9b-2c355bf81aa2",
            "principalName": "defaulttenantclient"
        },
        "ldapConfigId": "c4a82c0d-b3c1-4e9c-b492-cb046ea364f4"
    }
```

Running the Sample App Locally

To run the Sample App locally, you must perform the following steps.

1. Run the following command to obtain the out of the box default tenant client id and secret from the IDSP Kubernetes cluster:
   - `kubectl get secret ${RELEASENAME}-ssp-secret-defaulttenantclient -n ${NAMESPACE} -o jsonpath="{.data.clientId}" | base64 --decode; echo`
   - `kubectl get secret ${RELEASENAME}-ssp-secret-defaulttenantclient -n ${NAMESPACE} -o jsonpath="{.data.clientSecret}" | base64 --decode; echo`
2. Navigate to the Sample Web App project folder in the Support Portal > My Downloads (Search for IDSP in the portal and select the appropriate version number) or to your locally downloaded location, access the 'env.js' file and perform the following tasks:
   a. Provide the following details of the newly created application that you have obtained in Step 1:
      a. clientId
      b. clientSecret
      c. baseServiceUrl
3. In the Sample Web App project folder update the *package.json* file as follows:
   - *"start": "ng serve -ssl -host <HOSTNAME>",*
   - For example, "ng serve --ssl --host sampleapp.example.com",
4. Add an entry in the hosts file to match the <HOSTNAME> specified in step 3 pointing to 127.0.0.1.
5. Update the IDSP allowed origins by using the tenant settings PATCH call. In this call, update the hostname that you have specified in step 3 by appending the port at the end of the code, as shown in the following request:

```
curl -L -X PATCH 'https://iam.example.com/default/admin/v1/Configs' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <token>' \
--data-raw '[

    {
        "name": "allowedOrigins",
        "value": "https://sampleapp.example.com:4200"
    }
]'
```

Response

```
[
    {
        "name": "persistEnrolledFactorsinIDP",
```

```
        "value": "false",
        "origin": "global"
    },
    {
        "name": "azCodeExpirySecs",
        "value": "120",
        "origin": "global"
    },
    {
        "name": "isInlineOTPEnrollmentToBePersisted",
        "value": "true",
        "origin": "global"
    },
    {
        "name": "refreshTokenExpiryDays",
        "value": "7",
        "origin": "global"
    },
    {
        "name": "pushRegTimeoutSecs",
        "value": "300",
        "origin": "global"
    },
    {
        "name": "allowedOrigins",
        "value": "https://sampleapp.example.com:4200",
        "origin": "tenant"
    },
    {
        "name": "isInlineEnrollmentAllowed",
        "value": "true",
        "origin": "global"
    },
    {
        "name": "idTokenExpirySecs",
        "value": "21600",
        "origin": "global"
    },
    {
        "name": "pushAuthTimeoutSecs",
        "value": "300",
        "origin": "global"
    },
    {
        "name": "accessTokenExpirySecs",
        "value": "3600",
        "origin": "tenant"
    },
    {
        "name": "useOriginalIdentityTokenExpiry",
        "value": "false",
        "origin": "global"
    },
```

```
    {
        "name": "inlineEnrollmentFactorsRequired",
        "value": "PASSWORD",
        "origin": "global"
    }
]
```

NOTE
Ensure that the defaulttenant client access token that you have used is valid and has not crossed the expiry time.

6. Open a command prompt, navigate to the 'sampleapp' folder and enter "npm install". This step installs all the required modules for the sample app.
7. Run "**npm run build – --prod"** to build the project.
8. Run "**npm start**".
9. Open a browser and navigate to **"https://sampleapp.example.com:4200"**. The login page of the sample app is displayed.
10. Enter your username.
11. (Optional) Click the Settings gear at the top left of the login page to provide inputs for additional parameters, such as ip address or acr values.

Building the Docker Image

Open a command prompt in the Sample App folder where the "Dockerfile" is located. Run the following commands to generate the docker image:

- docker build --no-cache -t <IMAGE NAME>
- docker tag <IMAGE NAME> < ARTIFACTORY LOCATION>/<IMAGE NAME>
- docker push <ARTIFACTORY LOCATION>/<IMAGE NAME>

Installing Sample App Helm Chart

To install the Sample App Helm Chart, run the following commands:

- `helm repo add ssp_helm_charts https://ssp_helm_charts.storage.googleapis.com`
- `helm repo update`
- `helm install <helm_release_name> -n <k8s_namespace> --set image.sampleWebapp.repository=<docker_repository_name> --set image.sampleWebapp.tag=<image_tag_version> --set featureFlags.sampleIdStore.enabled=false --set ssp.serviceUrl="https://<authHub-fqdn-hostname>/<tenant>/" --set ssp.clientId="<clientId>" --set ssp.clientSecret="<clientSecret>" --set ingress.host=<sample-app-fqdn-hostname> --set ingress.tls.host=<sample-app-fqdn-hostname> --set ingress.tls.secretName=<k8s_secretName> ssp_helm_charts/ssp-sample-app`

Internals of the Sample Application

This section discusses, for example, how the access token is created, how the transactions flow between multiple factors and the reauthentication flows, the internal workings of the sample applications.

Generating Access Token

- Access token is generated on initial page load in the browser and is stored in local storage, so the end-user does not need to generate the token multiple times.
- Access token is generated using "ngOnInit()" (line#47) method in "init.component.ts" . This method checks if there are any pre-populated values as part of "env.js" and decides which ones to use whether the environment values or the locally stored values.

Transaction Flows Between the Multi-Factors

- After successful response from each api's, a call is made to "**displayservices.ts"** which has "**routeActions()"**(line#69). Based on the "nextaction" which is part of api response the flow is routed to the respective component. See examples below:
- Example explaining the flow to id store authentication:
  - {"flowState":<FLOWSTATE>,"nextaction":"PASSWORD_AUTH","currentFactors":["PASSWORD"]} – As the next action in this example is pointing to "PASSWORD_AUTH" we route this transaction to "ldap.component.ts" (line#88).
- Example explaining the flow to otp authentication:
  - {"flowState":"<FLOWSTATE>","credentials":[{"credId":"e6240c61-c0d0-4933-8566-f31cab4becaa","tenantId":"100","userId":"8c49025d-4a60-458a-9335-f94486687029","credType":"email","credValue":"su*******@broadcom.com ["OTP","FIDO"]} –As the next action in this example is pointing to "OTP_SELECTION" we route this transaction to "sotp.component.ts"(line#108)

Generating Device Signature for Device Risk Analysis

- All of this happens when the login page is loaded for the first time in the browser. An asynchronous call is sent to the "**fingerprint2.getpromise()"**(line#93) from **"ngOnInit()"**(line#47) in **"init.component.ts"** and will store this device signature in one of our local variables called **"fingerPrintDna"** which is will passed in the initial authenticate api call.

Re-authentication Flow

- Re-authentication happens in two services calls as to demo the usecases in multiple scenarios. One deals with **"StepUp"** which happens in **"welcome.component.ts"** and other scenarios in **"header.component.ts".**
- For more in-depth information we can go to individual methods like **"forceReauthWealth"** (line#21)**,"forceReauthCommercial"** (line#78),**"forceReauthBiz"**(line#52)

Token Display on the Welcome Page

- After successful authentication at the end of transaction **"id token"** is generated as part of successful response.
- Example: {"flowState": "<FLOWSTATE>","nextaction": "AUTH_ALLOWED","data": {"message": "Password authentication is successful for user example"},"deviceID": "<DEVICEID>","id_token": "<IDTOKEN>"}
- With this id token a "**user access token"** is generated via **"getUserAccessToken"**(line#46) in **"init.service.ts"**

FIDO Self-Registration

- With the above generated user access token fido self-registration is progressed via **"onRegister ()"**(line#80) in **"fidoreg.component.ts".**

Integrating Sample Mobile App with Symantec Identity Security Platform

Integrating Sample Mobile App with Symantec Identity Security Platform. Learn how to integrate the sample mobile application with IDSP APIs for multi-factor authentication, reauthentication, and risk flows.

The Sample Mobile application discusses how to set up a demo environment and test the various multi-factor authentication flows offered by the solution.

> **NOTE**
> The Sample Mobile application is not intended to be published to production and as such does not concern itself with the best practices, such as data modeling, design patterns, and so on. The primary aim of the Sample Mobile application is to address the recommended approach to integrate with IDSP

Sequence of Steps

This section lets you obtain a high-level understanding of the sequence of steps you must take to integrate the SampleMobile application with the IDSP APIs.

To set up your sample mobile application, perform the following steps:

1. Obtain the default tenant client id and secret from the IDSP Kubernetes cluster.
2. Generate a default Tenant Access Token- This step helps you to generate a default tenant access token which is a prerequisite for creating an application in the solution.
3. Create an Application- This step helps you create a client in the solution.
4. Create a Policy - This step lets you perform the following:
   a. Create the Authentication policy: This step helps you create an authentication policy for the application that you have created in the previous step. The authentication policy is evaluated, when the AuthN endpoint is accessed, and on successful policy evaluation, the ID Token is presented to you.
   b. Update the Authorization policy to provide the required privileges for the newly created application. This policy is evaluated to get the Access Token.
5. Generate the client access token. This helps you initiate the authentication flow to get the user Id token.
6. Upon successfully completing the authentication flow, you obtain the User ID token.
7. Using the User ID token, generate the access token. Now, that you have obtained the User Access Token, you can invoke the appropriate IDSP APIs to implement desired functionality.

These steps have been explained in the context of REST APIs in the following section.

Configuration Steps

> **NOTE**
> In the following steps, the curl commands have used <iam.example.com> as the ingress hostname which is a dummy value. Therefore, replace <iam.example.com> with your deployment's ingress hostname. You can get this value by using this command:
> *"kubectl gets ingress -n <namespace>"*

1. Use the following command to get the default tenant client id and secret from the IDSP Kubernetes cluster:
   - `kubectl get secret ${RELEASENAME}-ssp-secret-defaulttenantclient -n ${NAMESPACE} -o jsonpath="{.data.clientId}" | base64 --decode; echo`
   - `kubectl get secret ${RELEASENAME}-ssp-secret-defaulttenantclient -n ${NAMESPACE} -o jsonpath="{.data.clientSecret}" | base64 --decode; echo`
2. Use the following API to generate the default tenant access token by passing the client Id and secret in the authorization header:
   - `https://${SSP_FQDN}/{{tenantName}}/oauth2/v1/token`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/oauth2/v1/token`
  should look similar to:
  `https://example.com/default/oauth2/v1/token`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Generate Access Token Request

```
curl -L -X POST 'https://iam.example.com/default/oauth2/v1/token' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H 'Authorization: Basic
 OTc1MDljMjUtNjU0Ni00NmM0LTllNGQtMTJmYmJjYTYwZjNhOjgzNGI4ZjUxLWlMC1iYmFkLTg1NDdmMzYyM2ViNA==' \
--data-urlencode 'grant_type=client_credentials' \
--data-urlencode 'scope=urn:iam:myscopes'
```

Generate Access Token Response

```
{
 "access_token": "XQjUzI1NiI6ImNJbUNYN1FyRy",
 "token_type": "Bearer",
 "expires_in": 360
}
```

Now, you have the default Tenant Client Access Token with which you can create an application.

3. Create an application using the following API:

– `https://${SSP_FQDN}/{{tenantName}}/admin/v1/Apps`

Create Application Request

```
curl -L -X POST 'https://iam.example.com/default/admin/v1/Apps' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer Ry1BUUpkWWZ2Wlctd1BDcF' \
--data-raw '{
   "status": "active",
   "name": "Test Application",
   "description": "Default Application for Security",
   "clientType": "CONFIDENTIAL",
   "allowedOpenIDScopes": [
       "openid",
       "email",
       "profile"
   ],
   "redirectURIs": [
       "https://www.example.com"

   ],
   "allowedGrantTypes": [
       "implicit",
       "authorization_code",
       "urn:ietf:params:oauth:grant-type:jwt-bearer",
```

```
            "client_credentials"
        ],
        "allowedOperations": [
            "introspect", "onbehalfof"
        ],
        "skipIssuerAudienceForIT": true,
        "userInfoEndpointResponseFormat": "PLAIN_JSON"
    }'
```

Create Application Response

```
{
    "status": "active",
    "name": "test application",
    "description": "Default Application for Security",
    "clientType": "CONFIDENTIAL",
    "allowedOpenIDScopes": [
        "openid",
        "profile",
        "email"
    ],
    "redirectURIs": [
        "https://www.example.com",

    ]
    "allowedGrantTypes": [
        "implicit",
        "client_credentials",
        "authorization_code",
        "urn:ietf:params:oauth:grant-type:jwt-bearer"
    ],
    "userInfoEndpointResponseFormat": "PLAIN_JSON",
    "skipIssuerAudienceForIT": true,
    "allowedOperations": [
        "introspect",
        "onbehalfof"
    ],
    "appId": "a45247ea-1652-4b3b-9178-970a2caf89e5",
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "97509c25-6546-46c4-9e4d-12fbbca60f3a",
        "principalName": "defaulttenantclient"
    },
    "createdDateTime": "2020-06-11 10:37 AM UTC",
    "updatedDateTime": "2020-06-11 10:37 AM UTC",
    "secret": "b8fc0a31-2c0c-4e1f-ac3e-da298cd12819",
    "clientId": "947b2632-3283-4766-9cc9-f00a3fce6aee"
}
```

NOTE
The response payload fetches you the values for client Id, secret, and application Id for the application. Make a note of these values, as these will be used to sign in to the sample application.

4. Use the following API to create an authentication policy:
   - `https://${SSP_FQDN}/{{tenantName}}/admin/v1/AuthPolicies`

   Create Authentication Policy Request

```
curl -L -X POST 'https://iam.example.com/default/admin/v1/AuthPolicies' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer 6ImNJbUNYN1FyRy1BUUpkWWZ2Wlctd' \
--data-raw '{
   "description": "Sample policy Record ",
   "status": "active",
   "policyName": "Sample policy with channel web",
   "apps": [
       {
           "id": "a45247ea-1652-4b3b-9178-970a2caf89e5"
       }
   ],
   "rules": [
       {
           "conditions": {
               "channel": {
                   "operator": "in",
                   "value": [
                       "web"
                   ]
               }
           },
           "result": {
               "effect": "allow",
               "obligations": [
                   "PASSWORD",
                   "OTP"
               ],
               "msg": "This policy is for login in web channel"
           }
       }
   ]
}'
```

Create Authentication Policy Response

```
{
   "policyId": "8e9de6e1-8105-40da-942e-7841332d6b5b",
   "policyName": "Sample policy with channel web",
   "description": "Sample policy Record ",
   "status": "active",
   "matchAnyApp": false,
   "apps": [
       {
           "name": "Test Application",
           "id": "a45247ea-1652-4b3b-9178-970a2caf89e5"
       }
   ],
   "rules": [
       {
           "conditions": {
               "channel": {
                   "operator": "in",
                   "value": [
```

```
                        "web"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD",
                    "OTP"
                ],
                "msg": "This policy is for login in web channel"
            }
        }
    ],
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "97509c25-6546-46c4-9e4d-12fbbca60f3a",
        "principalName": "defaulttenantclient"
    },
    "createdDate": "2020-06-11 10:41:39"
}
```

5. Update the "AuthClient" Authorization policy with the application Id. To perform the update, we must first get all the authorization policies, and then update the "AuthClient" Authorization policy.
   a. Run the following endpoint to get the Authorization policies:

```
https://${SSP_FQDN}/{{tenantName}}/admin/v1/AuthZPolicies/{{authClientAuthorizationPolicyId}}
```

   Get all tenant Authorization Policy Request

```
curl -L -X GET 'https://iam.example.com/default/admin/v1/AuthZPolicies' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer XQjUzI1NiI6ImNJbUNYN1FyRy1BUUpkWWZ2Wlctd1BDcFQ0WT' \
--data-raw ''
```

   Get all tenant Authorization Policy Response

```
[
   {
       "policyId": "46010fd8-93c3-4adc-9632-b0d0726cf761",
       "policyName": "TenantAdminPolicy",
       "description": "Policy for Tenant Admin",
       "status": "active",
       "matchAnyApp": false,
       "apps": [
           {
               "name": "ssp",
               "id": "3690ac36-c805-43ef-8318-627023b16d64"
           }
       ],
       "rules": [
           {
               "conditions": {
                   "principal": {
                       "clientApp": {
                           "operator": "in",
                           "value": [
```

```
                            "7328b5c9-0ece-46d7-8758-6d08a618fb9a",
                            "059bcbbc-c8b6-467d-85d0-6d15f5a5c0f7"
                        ]
                    }
                }
            },
            "result": {
                "msg": "This is Tenant Admin authorization policy",
                "privileges": [
                    "urn:iam:t.apps",
                    "urn:iam:t.authenticate",
                    "urn:iam:t.configs",
                    "urn:iam:t.ldapconfigs",
                    "urn:iam:t.users",
                    "urn:iam:t.signingkey",
                    "urn:iam:t.certs",
                    "urn:iam:t.authzpolicies",
                    "urn:iam:t.authpolicies",
                    "urn:iam:t.groups",
                    "urn:iam:t.risk",
                    "urn:iam:t.userdata"
                ]
            }
        }
    ],
    "createdDate": "2020-06-10 05:36:21",
    "updatedDate": "2020-06-10 05:36:21"
},
{
    "policyId": "dbb932c7-c926-43df-a35c-f93a71c43f4e",
    "policyName": "AuthClientPolicy",
    "description": "Policy for Authentication Client",
    "status": "active",
    "matchAnyApp": false,
    "apps": [
        {
            "name": "ssp",
            "id": "3690ac36-c805-43ef-8318-627023b16d64"
        }
    ],
    "rules": [
        {
            "conditions": {
                "principal": {
                    "clientApp": {
                        "operator": "in",
                        "value": [
                            "2d93ffc1-ac22-48f7-8f80-206535f40afb"
                        ]
                    }
                }
            },
            "result": {
```

```
                    "msg": "This policy allows clients to call authentication APIs",
                    "privileges": [
                        "urn:iam:t.authenticate"
                    ]
                }
            }
        ],
        "createdDate": "2020-06-10 05:36:21",
        "updatedDate": "2020-06-10 05:36:21"
    }
                ]
```

b. Run the following API to grant "clientApp" to the "AuthClientPolicy":

```
https://${SSP_FQDN}/{{tenantName}}/admin/v1/AdminRolesDelegator
```

Update Authorization Policy Request

```
curl --location 'example.com/default/admin/v1/AdminRolesDelegator' \
 --header 'Content-Type: application/json' \
 --header 'Authorization: Bearer eyJ4NXQ....' \
 --data '{
     "id": "3805d28b-f980-46de-a497-fad560433623",
     "principal": {
         "clientApp": {
             "operator": "in",
             "value": [
                 "a45247ea-1652-4b3b-9178-970a2caf89e5"
             ]
         }
     }
 }'
```

The following is the Authorization Policy Response:

```
{
   "id": "dbb932c7-c926-43df-a35c-f93a71c43f4e",
   "name": "AuthClientPolicy",
   "displayName": "Authentication Client",
   "description": "Perform authentication-related operations as required to implement a custom signin
 app.",
   "principal": {
       "clientApp": {
           "operator": "in",
           "value": [
               "a45247ea-1652-4b3b-9178-970a2caf89e5",
           ]
       }
   },
   "isGrantableClientOnly": true
   }
```

6. (Optional) Perform this step to configure an LDAP User Store, if you have not configured it earlier. Run the following endpoint to configure an LDAP User Store:

```
https://${SSP_FQDN}/tenantName/admin/v1/LDAPconfigs
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/LDAPconfigs`
  should look similar to:
  `https://example.com/default/admin/v1/LDAPconfigs`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Configure LDAP User Store Request

```
curl -L -X GET 'https://iam.example.com/default/admin/v1/LDAPconfigs' \
-H 'Authorization: Bearer <token>'
```

Configure LDAP User Store Response


```
[
    {
        "tenantId": "4764021a-9048-11ea-bb37-0242ac130002",
        "configName": "TestConfig",
        "configDescription": "CA DIR k8s with default data",
        "sslEnabled": "NO",
        "certificate": null,
        "certificateAlias": null,
        "ldapQueryStart": "(&(cn=",
        "ldapQueryEnd": ")(objectClass=*))",
        "attributes": "uid,telephoneNumber,employeeType,mail,displayName,givenName,cn,sn,memberof,title",
        "objectClasses": "string",
        "groupObjectClasses": "groupOfNames",
        "hostName": "iam.example.com",
        "portNumber": "31568",
        "baseDN": "o=democorp,c=us",
        "groupBaseDN": "o=democorp,c=us",
        "userDN": "cn=DominicMAJOR,ou=Administration,ou=Corporate,o=democorp,c=US",
        "bindPassword": "/OTqCj2yPWVvhl6AjpJ/DQ==",
        "attributeMapping":
 "name=cn,family_name=sn,given_name=givenName,middle_name=givenName,nickname=displayName,preferred_username=displayN
 status=employeeType,memberof=memberof",
        "groupSearchMappingAttribute": "member",
        "useMemberOf": "yes",
        "namingAttribute": "cn",
        "passwordResetURL": "http://userPasswordResetURL",
        "createdDateTime": "2020-06-30 15:40 PM UTC",
        "updatedDateTime": "2020-06-30 15:40 PM UTC",
        "createdBy": {
            "principalType": "CLIENT",
            "principalId": "f4bcb936-769f-4377-9e76-276083d1958e",
            "principalName": "defaulttenantclient"
        },
        "ldapConfigId": "3ec2ecfa-d88b-49cd-b50b-e4dba6094a9d"
    }
]
```

7. (Optional) Perform this step to create an LDAP User Store.

   NOTE

   Check if a user store already exists. If yes, then skip this step.

Configure LDAP User Store Request

```
curl -L -X POST 'https://iam.example.com/default/admin/v1/LDAPconfigs' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <token>' \
--data-raw '{
    "configName": "TestConfig",
    "configDescription": "CA DIR k8s with default data",
    "sslEnabled": "NO",
    "ldapQueryStart": "(&(cn=",
    "ldapQueryEnd": ")(objectClass=*))",
    "attributes": "uid,displayName,telephoneNumber,mail,memberof",
    "objectClasses": "string",
    "hostName": "iam.example.com",
    "portNumber": "31888",
    "baseDN": "o=democorp,c=us",
    "groupBaseDN": "o=democorp,c=us",
    "groupObjectClasses": "groupOfNames",
    "groupSearchMappingAttribute": "member",
    "useMemberOf": "yes",
    "namingAttribute": "cn",
    "userDN": "cn=DominicMAJOR,ou=Administration,ou=Corporate,o=democorp,c=US",
    "bindPassword": "password",
    "attributeMapping":
 "name=cn,family_name=sn,given_name=givenName,middle_name=givenName,nickname=displayName,preferred_username=displayN
 status=employeeType,memberof=memberof",
    "passwordResetURL": "http://userPasswordResetURL"
}'
```

Configure LDAP User Store Response

```
{
    "tenantId": "4764021a-9048-11ea-bb37-0242ac130002",
    "configName": "TestConfig",
    "configDescription": "CA DIR k8s with default data",
    "sslEnabled": "NO",
    "certificate": null,
    "certificateAlias": null,
    "ldapQueryStart": "(&(cn=",
    "ldapQueryEnd": ")(objectClass=*))",
    "attributes": "uid,telephoneNumber,employeeType,mail,displayName,givenName,cn,sn,memberof,title",
    "objectClasses": "string",
    "groupObjectClasses": "groupOfNames",
    "hostName": "iam.example.com",
    "portNumber": "31888",
    "baseDN": "o=democorp,c=us",
    "groupBaseDN": "o=democorp,c=us",
    "userDN": "cn=DominicMAJOR,ou=Administration,ou=Corporate,o=democorp,c=US",
    "bindPassword": "/OTqCj2yPWVvhl6AjpJ/DQ==",
    "attributeMapping":
 "name=cn,family_name=sn,given_name=givenName,middle_name=givenName,nickname=displayName,preferred_username=displayN
 status=employeeType,memberof=memberof",
```

```
        "groupSearchMappingAttribute": "member",
        "useMemberOf": "yes",
        "namingAttribute": "cn",
        "passwordResetURL": "http://userPasswordResetURL",
        "createdDateTime": "2020-06-23 04:24 AM UTC",
        "updatedDateTime": "2020-06-23 04:24 AM UTC",
        "createdBy": {
            "principalType": "CLIENT",
            "principalId": "006eddf2-ea8d-46b6-8c9b-2c355bf81aa2",
            "principalName": "defaulttenantclient"
        },
        "ldapConfigId": "c4a82c0d-b3c1-4e9c-b492-cb046ea364f4"
    }
```


Steps to Build and Install Sample Mobile Application

Learn about the steps to install the Sample Mobile application (also known as Hedmoral Technologies app) to perform authentication flows in Symantec Identity Security Platform.

The Mobile client application is available for users to perform EMAILOTP, SMSOTP, FIDO, and Password authentication flows.

> **NOTE**
> The Mobile application does not support Push and Mobile OTP flows.

This topic includes the following:

- Prerequisites
- Latest Builds
- Installing the Mobile Application Using APK File

Prerequisites

To be able to run the latest Hedmoral Mobile Application (version 12.0), ensure that you have met the following prerequisites:

- Android Studio "2025.1.4" or later
- Android Gradle Plugin 8.13.2

Downloading and Building

Follow these steps to download the latest code and build the Mobile Authentication application.

1. Navigate to the Support Portal > My Downloads.
2. In the package, look for **IDSP_4.0-Android-Release-Package-v3.0.3.zip** and unzip it in a local directory in your machine.
3. Unzip the **"hedmoral-sample-app.zip"** source code in the same directory as the previous step.
4. Use the following commands to build the application. Ensure to set the ANDROID_HOME environment before building the sample app:

   ```
   $ cd hedmoral-sample-app
   $ ./gradlew build
   ```

   The output displays a message similar to the following:

   ```
   BUILD SUCCESSFUL in 14s
   59 actionable tasks: 58 executed, 1 up-to-date
   ```
5. After the build is successfully deployed, you can find the APK debug & release file in the following location:

- The "hedmoraltechnologies-debug.apk" output file is available in the following location:
  ```
  $ cd hedmoraltechnologies/build/outputs/apk/debug
  ```
- The "hedmoraltechnologies-release.apk" output file is available in the following location:
  ```
  $ cd hedmoraltechnologies/build/outputs/apk/release
  ```

Installing the Mobile Application Using APK File

Follow these steps:

1. Copy the Android Package Kit, "APK" file from the "hedmoral-sample-app-apk-files" folder to your mobile device.
2. On your device, go to Menu Settings Security and check the **Menu** > **Settings** > **Security "Allow installation of apps from unknown sources"** option.
3. Tap the "APK" file and then click "INSTALL". The app begins installing on your device.
4. Look for the "Hedmoral Technologies" app in the Menu.

Now you can start testing the Mobile use cases described here using the IDSP app.

Using the Sample Mobile Application

This topic includes the following:

Using the Sample Mobile application

Follow these steps to use the Sample Mobile application:

1. Open the Installed Application.
2. Click the gear icon available on the bottom right corner.
   This opens the Application Configuration UI.
3. Enter the following inn the UI:
   `Endpoint Host Name, Client ID & Client Secret`
4. Save the Application Configuration.
   You are automatically navigated to the MainActivity where you must enter your Username to initiate the Authentication.

Flow Controller in Mobile Application

The following flow states take care of user factors registration and authentication based on a configured Authentication Policy.

PolicyController.java

State	Description
INIT0	Generates Access Token to invoke the "/authenticate" API. For this API Endpoint Host, Client ID & Client Secret is picked from Preferences which the user enters by launching the **ApplicationSettings** Activity.
INIT	Calls "/authenticate" API to evaluate.
FACTOR_SELECTION	Opens UI for Factor Selection where user can select (PASSWORD, EMAILOTP, SMSOTP, FIDO, etc) factors for authentication (**FactorSelectionActivity**)
FACTOR_SELECTED_BY_USER	Factor selected by the user
PASSWORD_AUTH	Opens UI for User to enter password (**LDAPPassword**)

LDAP_AUTHENTICATION	Initiates the LDAP Authentication
EMAIL_OTP_REGISTER	This activity opens only when the user does not have EMAIL configured and EMAIL_OTP Factor is not available yet. Opens UI for user to register EMAIL_OTP Factor (EMAIL_OTP_REGISTER).
EMAIL_OTP_REGISTER_2	Initiates the EMAIL_OTP Factor registration API
EMAIL_OTP_SELECTION	Multiple registered email addresses are shown as masked text, where the user can select one of them to receive EMAIL_OTP (**EMAILOTPSelection**)
EMAIL_OTP_GENERATE	Generates EMAIL_OTP and sends EMAIL_OTP to EMAIL
EMAIL_OTP_AUTH	Opens UI to enter EMAIL_OTP received by User (EMAIL_OTP Collection)
EMAIL_OTP_VERIFY	Initiates EMAIL_OTP Authentication
SMS_OTP_REGISTER	This activity opens only when the user does not have SMS configured and SMS_OTP Factor is not available yet. Opens UI for user to register SMS_OTP Factor (SMS_OTP_Register).
SMS_OTP_REGISTER_2	Initiates the SMS_OTP Factor registration API
SMS_OTP_SELECTION	Multiple registered mobile numbers are shown as masked text, where the user can select one of them to receive SMS_OTP (SMSOTPSelection).
SMS_OTP_GENERATE	Generates SMS_OTP and sends SMS_OTP as SMS
SMS_OTP_AUTH	Opens UI to enter SMS_OTP received by user (SMS_OTP Collection)
SMS_OTP_VERIFY	Initiates SMS_OTP Authentication
FIDO_REGISTER_GENERATE_CHALLENGE	Opens UI to enter Device Identifier in case the user does not have FIDO Factor available (**FIDODeviceNameCollection**).
FIDO_REGISTER_GENERATE_CHALLENGE_2	Initiates FIDO Registration SDK API which in turn initiates the Android OS FIDO Registration Activity.
FIDO_AUTH_GENERATE_CHALLENGE	Opens FIDO Authentication Waiting screen and initiates the Android OS FIDO Authentication (**FIDOAuthenticationActivity**).
AUTH_FAILURE	Indicates that User Authentication was not successful and then takes the user back to MainActivity or ProtectedActivity based on whether a user wants to perform authentication or reauthentication respectively.
AUTH_ALLOWED	Opens ProtectedActivity which is the User Protected Dashboard and stores the user ID Token in Preferences from where the user can try initiating the reauthentication flow by selecting any of the following action: 1. Investment Banking 2. Business Banking 3. Wealth Banking 4. Commercial Banking

Implementing Registration of Multiple FIDO Credentials

This topic describes how to handle the registration of multiple FIDO credentials in different devices in Symantec Identity Security Platform.

Use Case

If you already have a FIDO credential registered for a device (such as Chrome on Windows), and you attempt to register a new credential for a different device (such as an Android phone, an iOS phone, or a MAC), a credential registration error is returned by the authenticator.

For both, iOS and Android FIDO SDKs, the mobile SDK returns the error, **FAILURE_FIDO_CREDENTIAL_NOT_FOUND: 909**, and the web client /browser returns a generic error message, for example- Google Chrome returns, **"DOMException: The operation either timed out or was not allowed. See: https://www.w3.org/TR/webauthn-2/#sctn-privacy-considerations-client."**, when the FIDO credential that was passed to the authenticator (device/browser) is not present on the device. This error code signifies that the FIDO Authenticator was unable to locate the credential.

In this case, the client application can be designed to provide a choice to the user to register the new FIDO credential on the new device.

Follow this step to implement such a flow at the mobile/web application side:

When the error message is displayed, the client app should check for the **'isInlineEnrollmentAllowed'** attribute configuration that is set in the IDSP's ***".../factor/v1/FIDOAuthChallengeGenerator"*** server API response. This may result in the following:

- if the **"isInlineEnrollmentAllowed"**="**true**", the client application can render the option to register a new FIDO credential. Therefore, if the user decides to register a new credential, the client app must invoke the IDSP's ***".../auth/v1/SelectedFactor"*** server API with the following payload:

  ```
  { "action": "restorelastaction",  "registerFido" : true }
  ```

  This initiates the FIDO credential registration flow. This step is followed by next actions, where the client application must invoke IDSP's ***".../factor/v1/FIDORegChallengeGenerator"*** API and ***".../factor/v1/FIDORegChallengeVerifier"*** server APIs to register the new credential.
- if the **"isInlineEnrollmentAllowed"="false",** the client app cannot register a new FIDO credential and must display the proper option to proceed.

NOTE

- Every client type returns different error messages and codes.
- For more details about error messages from mobile SDK, refer to the FIDO SDK (iOS) Error Codes.

Reference Link:

- Security Key and Biometric (FIDO) Authenticator.

Integrating Custom Web Applications with Symantec Identity Security Platform

This section describes the process of using Symantec Identity Security Platform to provide authentication for a custom web application, using the OpenID Connect protocol. It covers the required artifacts and sequence of calls between the application and IDSP in order to demonstrate the OpenID Connect flow.

The OpenID Connect (OIDC) protocol is a lightweight REST/JSON based identity protocol built on top of the OAuth 2.0 protocol. It is important to understand that OpenID Connect is mainly a user authentication mechanism enabling use of identity to be decoupled from the management of identity, while OAuth2 itself is a delegated authorization mechanism. Since user authentication is necessary to support the delegated authorization, the OAuth2 specification was overloaded to satisfy user authentication use cases. Using OpenID Connect is a modern and straightforward way for an application to delegate user authentication to a common authentication service that also doubles up as a lightweight identity profile service completing its OpenID Connect responsibilities.

NOTE
Access Tokens and Identity Tokens generated by flows mentioned in this document can be examined using the jwt debugger.

Overview

The application is expected to integrate with OpenID Connect using the OpenID Connect Authorization Code flow. Note that to perform the actual OpenID Connect flow, numerous libraries exist for different platforms/languages to assist developers with the uptake.

The Authorize endpoint is used to initiate OpenID Connect flow to authenticate the user identity and acquire Identity and Access tokens.

1. Application makes a browser call to IDSP's Authorize endpoint with the required parameters, specifying "openid" as one of the requested scopes (or as the only scope), and providing the "callback URL" for IDSP to invoke after the authentication flow completes. Presence of the "openid" scope instructs IDSP to begin the OpenID Connect flow.
2. IDSP authorizes application's request for requested scopes and begins the user authentication ceremony
3. IDSP redirects user's browser to the sign in page. This can be IDSP's built-in sign in page, or this can be the application's own custom login page. For this example we use the built-in sign in page
4. The sign in page orchestrates the user authentication ceremony to obtain from the user all the credentials required by applicable authentication policies in effect for the requesting application, specific user identity, and other contextual information.
5. Upon successful authentication, IDSP generates an authorization code and redirect user's browser to the callback URL specified in step 1, appending the code to the redirect URL.
6. Application makes a REST call to IDSP's Token endpoint with grant_type=authorization_code and Authorization Code value
7. IDSP authenticates application's client credentials, supplied authorization code, and returns a JSON document containing the JWT-based Identity Token and Access Token
8. Application uses IDSP's public signing certificate to validate, among other validations, the Identity Token containing claims about the authenticated identity. The exact validation flow is specified by the OpenID Connect specification.
9. If required, application makes a REST call to IDSP's Userinfo endpoint to request user profile data authorized by the Access Token.
10. This completes the OpenID Connect flow. The application now has the user's identity and profile information and can map the identity to, or create a profile for, local application identity.

Configuring IDSP

To get the OpenID Connect flow working, we use IDSP REST APIs to pre-create required artifacts to ensure the example works end to end. It is assumed these steps have not been done in the IDSP environment.

Step1: Generate access token to make REST API calls

1. Use the following command to get the default tenant client id and secret from IDSP's K8S cluster. Alternatively use the id/secret of the OpenID Connect capable client as provided to you. *Obtain Admin Client's ID and Secret*

```
kubectl get secret ${RELEASENAME}-ssp-secret-defaulttenantclient -n ${NAMESPACE} -o
 jsonpath="{.data.clientId}" | base64 --decode; echo
kubectl get secret ${RELEASENAME}-ssp-secret-defaulttenantclient -n ${NAMESPACE} -o
 jsonpath="{.data.clientSecret}" | base64 --decode; echo
```

2. Generate an access token by calling IDSP's Token endpoint using <clientId> and <clientSecret> from the previous step.
   Request payload

```
curl -k -u '<clientId>:<clientSecret>' \
   --request POST --url 'https://iam.example.com/default/oauth2/v1/token' \
   --header 'Content-Type: application/x-www-form-urlencoded' \
   --data-urlencode 'grant_type=client_credentials' \
          --data-urlencode 'scope=urn:iam:myscopes'
```

   Response payload

```
"access_token": "eyJ4NXQjUzI1NiI6ImNJbUNYN1........",
```

```
        "token_type": "Bearer",
        "expires_in": 3600
      }
```

Step 2 - Create LDAP ID Store Configuration for Sample ID Store

In the sample OpenID Connect flow, users being authenticated are expected to be located in an identity store. Invoke a IDSP REST API to create the ID Store configuration representing IDSP's Sample ID Store, replacing <AccessToken> with the value of the generated access token. Make sure to replace the hostName and PortNumber with your sample directory's FQDN host and port.

Create LDAP Configuration Request

```
curl -k --request POST --url 'https://iam.example.com/default/admin/v1/LDAPconfigs' \
   --header 'Content-Type: application/json' \
   --header 'Authorization: Bearer eyJ4NXQjUzI1NiI6ImNJbUNYN1........' \
   --data-raw '{
       "configName": "SampleIDStore",
       "configDescription": "SampleIDStore CA Dir",
       "sslEnabled": "no",
       "ldapQueryStart": "(&(cn=",
       "ldapQueryEnd": ")(objectClass=*))",
       "attributes":
 "employeeType,telephoneNumber,mail,displayName,givenName,name,company,cn,sn,memberof,title,l",
       "objectClasses": "inetorgperson",
       "hostName": "iam.example.com",
       "portNumber": "31888",
       "baseDN": "o=democorp,c=us",
       "groupBaseDN": "o=democorp,c=us",
       "groupObjectClasses": "groupOfNames",
       "groupSearchMappingAttribute": "member",
       "useMemberOf": "yes",
       "namingAttribute": "cn",
       "userDN": "cn=DominicMAJOR,ou=Administration,ou=Corporate,o=democorp,c=US",
       "bindPassword": "password",
       "attributeMapping": "status=employeeType, name=cn, family_name=sn, given_name=givenName,
 middle_name=givenName, nickname=displayName, preferred_username=displayName,
 profile=title,email=mail,phone_number=telephoneNumber, user_universalid=cn,memberof=memberof",
       "passwordResetURL": "http://userPasswordResetURL"
        }
```

Create LDAP Configuration Response

```
{
     "tenantId": "4764021a-9048-11ea-bb37-0242ac130002",
     "configName": "SampleIDStore",
     "configDescription": "SampleIDStore CA Dir",
     "sslEnabled": "no",
     "certificate": null,
     "certificateAlias": null,
     "ldapQueryStart": "(&(cn=",
     "ldapQueryEnd": ")(objectClass=*))",
     "attributes":
 "employeeType,telephoneNumber,mail,displayName,givenName,name,company,cn,sn,memberof,title,l",
     "objectClasses": "inetorgperson",
     "groupObjectClasses": "groupOfNames",
     "hostName": "iam.example.com",
```

```
    "portNumber": "31888",
    "baseDN": "o=democorp,c=us",
    "groupBaseDN": "o=democorp,c=us",
    "userDN": "cn=DominicMAJOR,ou=Administration,ou=Corporate,o=democorp,c=US",
    "bindPassword": "/OTqCj2yPWVvhl6AjpJ/DQ==",
    "attributeMapping": "status=employeeType, name=cn, family_name=sn, given_name=givenName,
 middle_name=givenName, nickname=displayName, preferred_username=displayName,
 profile=title,email=mail,phone_number=telephoneNumber, user_universalid=cn,memberof=memberof",
    "groupSearchMappingAttribute": "member",
    "useMemberOf": "yes",
    "namingAttribute": "cn",
    "passwordResetURL": "http://userPasswordResetURL",
    "createdDateTime": "2020-06-23 18:52 PM UTC",
    "updatedDateTime": "2020-06-23 18:52 PM UTC",
    "createdBy": {
      "principalType": "CLIENT",
      "principalId": "6f0f804b-38ec-4e91-aac9-f112ffc849ce",
      "principalName": "defaulttenantclient"
    },
    "ldapConfigId": "cdd13ae4-b9ab-4296-ae2f-4a17e7c2b7e5"
  }
```


Step 3 - Create Application Resource

This application requests OpenID Connect authentication for users residing in the sample id store. Invoke the IDSP REST API to create the Application Resource, replacing <AccessToken> in the Authorization header with the access token thats has beed generated in Step 1. Make sure to replace the hostName and PortNumber with your sample directory's FQDN host and port.

The response payload brings back the appId, clientId, and clientSecret for the newly created application.

- Make a note of the returned AppID as it will be used in the Authentication Policy to "secure the app" by Password-based authentication.
- Make a note of returned clientID and ClientSecret as these will be used to execute the OpenID Connect flow.

Create Application Request

```
curl -k --request POST --url 'https://iam.example.com/default/admin/v1/Apps' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer eyJ4NXQjUzI1N........' \
--data-raw '{
"status": "active",
"name": "OIDC App",
"description": "App using OpenID Connect AZ Code flow",
"clientType": "CONFIDENTIAL",
"allowedOpenIDScopes": [
"openid",
"email",
"profile"
],
"redirectURIs": [
"https://www.example.com/azcode"
],
"allowedGrantTypes": [
"implicit",
"authorization_code"
```

```
],
"allowedOperations": [
"introspect",
"onbehalfof"
],
"skipIssuerAudienceForIT": false,
"userInfoEndpointResponseFormat": "PLAIN_JSON"
}'
```

Create Application Response


```
{
  "status": "active",
  "name": "oidc app",
  "description": "App using OpenID Connect AZ Code flow",
  "clientType": "CONFIDENTIAL",
  "allowedOpenIDScopes": [
    "openid",
    "profile",
    "email"
  ],
  "redirectURIs": [
    "https://www.example.com/azcode"
  ],
  "allowedGrantTypes": [
    "implicit",
    "authorization_code"
  ],
  "userInfoEndpointResponseFormat": "PLAIN_JSON",
  "skipIssuerAudienceForIT": false,
  "allowedOperations": [
    "introspect",
    "onbehalfof"
  ],
  "appId": "5211485d-4013-4136-9f4e-4abc4b928d81",
  "createdBy": {
    "principalType": "CLIENT",
    "principalId": "6f0f804b-38ec-4e91-aac9-f112ffc849ce",
    "principalName": "defaulttenantclient"
  },
  "createdDateTime": "2020-06-23 21:26 PM UTC",
  "updatedDateTime": "2020-06-23 21:26 PM UTC",
  "secret": "0989bcba-fba9-477b-9841-194b0f7147e4",
  "clientId": "a85a9a21-7d37-4afe-9c9c-ee19174fb66b"
}
```


Step 4 - Create Authentication Policy

Applicable authentication policies drives the specific credentials a user has to present in order to get authenticated.

Invoke the IDSP REST API to create the Authentication Policy, replacing <AccessToken> in the Authorization header with the access token generated in Step 1. Also, replace the "id" in the "apps" section with the "appId" of the Application Resource created in Step 3.

Create Authentication Policy Request

```
curl -k --request POST --url 'https://iam.example.com/default/admin/v1/AuthPolicies' \
```

```
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer eyJ4NXQjUzI1NiI6I......' \
--data-raw '{
  "description": "Test Authentication Policy for OpenID Connect App",
  "status": "active",
  "policyName": "Test Authentication policy for OIDC",
  "apps": [
    {
      "id": "5211485d-4013-4136-9f4e-4abc4b928d81"
    }
  ],
  "rules": [
    {
      "conditions": {
        "principal": {
          "group": {
            "operator": "in",
            "value": [
              "Admin"
            ]
          },
          "user": {
            "operator": "in",
            "value": [
              "Malbail"
            ]
          }
        },
        "action": {
          "operator": "in",
          "value": [
            "authenticate"
          ]
        },
        "channel": {
          "operator": "in",
          "value": [
            "web"
          ]
        }
      },
      "result": {
        "effect": "allow",
        "obligations": [
          "PASSWORD"
        ],
        "msg": "This authentication policy is for signin via web channel"
      }
    }
  ]
}'
```

Create Authentication Policy Response

```
{
```

```
"policyId": "8306992b-54fe-44a3-9118-465eb5e0bc93",
"policyName": "Test Authentication policy for OIDC",
"description": "Test Authentication Policy for OpenID Connect App",
"status": "active",
"matchAnyApp": false,
"apps": [
  {
    "name": "OIDC App",
    "id": "5211485d-4013-4136-9f4e-4abc4b928d81"
  }
],
"rules": [
  {
    "conditions": {
      "principal": {
        "group": {
          "operator": "in",
          "value": [
            "Admin"
          ]
        },
        "user": {
          "operator": "in",
          "value": [
            "Malbail"
          ]
        }
      },
      "action": {
        "operator": "in",
        "value": [
          "authenticate"
        ]
      },
      "channel": {
        "operator": "in",
        "value": [
          "web"
        ]
      }
    },
    "result": {
      "effect": "allow",
      "reAuthenticate": "false",
      "obligations": [
        "PASSWORD"
      ],
      "msg": "This authentication policy is for signin via web channel"
    }
  }
],
"createdBy": {
  "principalType": "CLIENT",
```

```
    "principalId": "6f0f804b-38ec-4e91-aac9-f112ffc849ce",
    "principalName": "defaulttenantclient"
  },
  "createdDate": "2020-06-23 21:49:58"
}
```


Authenticating the User

Now that we have the setup completed, we can look at the details of the OpenID Connect flow. There are three main steps involved and the first is to send the user to the OpenID Connect Provider for authentication. This is done by redirecting the client browser to IDSP OpenID Connect's Authorize endpoint and passing parameters, as defined below. Note that this redirect can be accomplished in a number of different ways: the user can explicitly click a link on the app homepage to start the login flow, or the app could use an intercepting session filter to generate an automatic redirect when it detects that the browser does not have a valid application session.

The Authorize endpoint is used by IDSP to initiate the user authentication ceremony to assert the user identity. The main purpose of this endpoint is to support the Authorization Code flow. After that the client exchanges the Authorization Code for the Identity Token and Access Token by invoking the authorization's server Token endpoint.

Security Considerations for Public Clients

When public clients such as native and single-page applications request user authentication, there are additional security concerns related to the fact that such clients cannot securely hold a secret or indicate a trusted callback URL. Proof Key for Code Exchange (PKCE) as OAuth 2.0 RFC 7636 addresses this issue.

Adding PKCE parameters to the Authorization Code flow enables the client to introduce a secret (Code Verifier) that can be validated by the authorization server. The client creates and sends Code Challenge as part of the /authorize request to get the Authorization Code. When the client exchanges the Authorization Code for an Access Token, it includes the Code Verifier. This ensures that even if the attacker intercepts the /authorize request, they will not be able to exchange is for the access token without the Code Verifier.

OpenID Connect Parameters

The following table displays the OIDC parameters.

Parameter	Optionality	Description
client_id	Required	The Client ID for this application. This client id is used to lookup allowed grants and operations, and to have the Authorization Code.
response_type	Required	For Authorization Code flow, this must be specified as "code".
scope	Required	For OpenID Connect flow, add "openid" to request issuance of Identity Token. Indicate additional OpenID scopes (Client's Allowed OpenID Scopes such as *profile* or *email*) to authorize the client to invoke the Userinfo endpoint to request user data represented by those scopes.
redirect_uri	Required	This is the URL-encoded endpoint to which the client will be sent back once the Authorization Code has been issued by the authorization server. The Authorization Code is available in the *code* URL parameter. This URL must be specified exactly as provided in one of the values during client registration.

Parameter	Optionality	Description
state	Optional	This is an optional security parameter for the client to protect against certain attacks such as Cross-Site Request Forgery (CSRF, XSRF). Usually this value is generated by cryptographically binding to some browser state such as an application cookie to detect a possible request hijacking attempt. The value supplied by the client will be returned by the authorization server along with the Authorization Code.
nonce	Optional (OpenID only)	This is an optional security mechanism to protect against replay attacks. This should be a strong **random** value associated with the internal application session. For OpenID Connect flows, this value is included in the Identity Token.
code_challenge_method	Optional (For use with PKCE only)	Method used to generate the **PKCE** challenge The PKCE spec includes `S256` and `plain`. Use of *plain* is not recommended.
code_challenge	Optional (For use with PKCE only)	Generated challenge from the **PKCE** Code Verifier

Step 1 - Obtaining Authorization Code

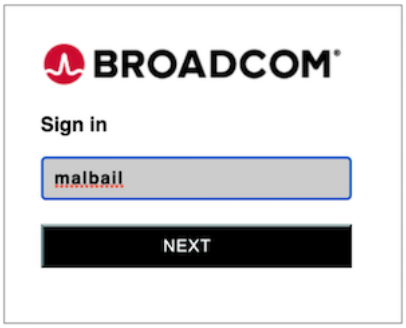
Application executes a browser request to generate an Authorization Code to be exchanged for OpenID Connect's Identity Token and Access Token in the next step. Here, the application uses its clientID that was established during creation of the Application Resource.

Create Authentication Policy Response

```
https://iam.example.com/default/oauth2/v1/authorize?response_type=code&client_id=a85a9a21-7d37-4afe-9c9c-
ee19174fb66b&redirect_uri=https://www.example.com/azcode&scope=openid%20email&state=mystate&nonce=mynonce
```

> **NOTE**
> *If PKCE was used, code_challenge would have been computed and added to the query string params: ...&code_challenge_method=S256&code_challenge=9f86d081884c7d659a2feaa0c55ad.....'*

User browser is redirected to the built-in signin page. We use user "malbail" expected to be a member of "admin" group as maintained in IDSP's sample id store.



After the authentication flow for user "malbail" completes, IDSP will redirect to client's redirect_uri with Authorization Code, and State if originally provided when invoking the Authorize endpoint.

```
https://www.myapp.com/azcode/?code=7riOHCJohHZK5Fs1C6jNHf3FuQdZWf&state=mystate&X-TRANSACTION-
ID=66d19fce-60fe-4ba7-a4a8-b48dfbf53814
```

Step 2 - Exchanging Authorization Code for Tokens

The client then exchanges the Authorization Code for Identity Token and Identity Token. Before the application does that, it should examine the received "state" parameter value redirect to make sure it matches the one specified in the previous steps when redirecting to the Authorize endpoint. Then, application extracts the received code parameter and passes it to IDSP through the REST API call to IDSP's Token endpoint. This back-channel call guards against the potential "man in the middle" threat that exists in browser redirects. Here, the application uses its clientID and clientSecret that was established during creation of the Application resource.

Token Request

```
curl -k --request POST --url 'https://iam.example.com/default/oauth2/v1/token' --header 'Content-
Type: application/x-www-form-urlencoded' -u "a85a9a21-7d37-4afe-9c9c-ee19174fb66b:0989bcba-
fba9-477b-9841-194b0f7147e4" --data-urlencode 'grant_type=authorization_code' --data-urlencode
 'code=7riOHCJohHZK5Fs1C6jNHf3FuQdZWf'
```

Token Response

```
{
  "access_token" : "eyJ4NXQjUzI1NiI6ImNJb......",
  "token_type" : "Bearer",
  "expires_in" : 3600,
  "id_token" : "eyJ4NXQjUzI1NiI6ImNJbUNYN1......"
}

Decoded Access Token
{
  "client_tid": "default",
  "sub": "b0ee6444-2c9c-4f67-84fb-9fd1e6040397",
  "ver": "1.0",
  "idp_name": "SampleIDStore",
  "user_universalid": "MalBAIL",
  "user_loginid": "malbail",
  "idp_guid": "cdd13ae4-b9ab-4296-ae2f-4a17e7c2b7e5",
  "iss": "https://iam.example.com/default/",
  "azpacr": "1",
  "tok_type": "AT",
  "user_riskscore": 0,
  "tid": "default",
  "aud": "https://iam.example.com/default/",
  "nbf": 1592949250,
  "sub_type": "USER",
  "azp": "a85a9a21-7d37-4afe-9c9c-ee19174fb66b",
  "scope": "urn:iam:m.me openid email",
  "exp": 1592952850,
  "iat": 1592949250,
  "client_name": "oidc app",
  "jti": "a9edbe83-efb0-492a-b113-b7b0976f5c88",
  "idp_type": "ldap"
}

Decoded Identity Token
{
  "at_hash": "aka9_MfKGSvmJkVoQ67E2A",
  "sub": "b0ee6444-2c9c-4f67-84fb-9fd1e6040397",
  "idp_name": "SampleIDStore",
```

```
  "amr": [
    "PWD"
  ],
  "user_loginid": "malbail",
  "idp_guid": "cdd13ae4-b9ab-4296-ae2f-4a17e7c2b7e5",
  "iss": "https://iam.example.com/default/",
  "aal": "AAL1",
  "tid": "default",
  "sid": "eb005d6f-865a-489d-93d0-74b9f16efe1e",
  "acr": "urn:iam:acr:aal1",
  "azp": "a85a9a21-7d37-4afe-9c9c-ee19174fb66b",
  "auth_time": 1592949198,
  "exp": 1592970850,
  "iat": 1592949250,
  "client_name": "oidc app",
  "jti": "b355d5f1-e69d-4276-9791-d330de550c49",
  "idp_type": "ldap",
  "ver": "1.0",
  "user_universalid": "MalBAIL",
  "tok_type": "IT",
  "user_riskscore": 0,
  "nonce": "mynonce",
  "aud": [
    "a85a9a21-7d37-4afe-9c9c-ee19174fb66b",
    "https://iam.example.com/default/"
  ],
  "nbf": 1592949250
}
```


Step 3 - Validating Identity Token

Before the Identity Token can be used, it must be validated as per the OpenID Connect specification, validating the signature, expiration, and audience.

To support signature validation, your application needs to acquire IDSP's signing certificate from IDSP and pass that into the library you will be using to perform JWT validation. There are different ways of using such libraries, but almost every approach requires getting the signer's certificate. In IDSP, this can be done using the JWKS endpoint (curl -k https://iam.example.com/default/oauth2/v1/jwks). There are best practices for obtaining, caching, and using public keys to accommodate expected and unexpected server-side certificate rollovers. The actual validation of the Identity Token is beyond the scope of this document.

> **NOTE**
> If the "nonce" value was used, the application must validate that the nonce in the Identity Token matches the nonce passed into IDSP's Authorize flow.

Step 4 - Acquiring User Info

Finally, the application invokes IDSP's Userinfo endpoint to request information about the user, using the Access Token generated in the previous step.

> **NOTE**
> Another way to specify the access token is via form variable instead of the authorization header: --data-urlencode 'access_token=<token>'

The application can use this data to create a local user profile. It can also perform "just in time" creation of the user if so required.

UserInfo Request

```
curl -k --request GET \
--url 'https://iam.example.com/default/oauth2/v1/userinfo' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Authorization: Bearer eyJ4NXQjUzI1NiI6ImNJbUNYN1.......'
```

Response below assumes the application has been configured to receive JSON from the Userinfo endpoint.

> **NOTE**
> *To see the email value get returned for user 'malbail', first update the sample id store's entry for 'malbail' with the email value.*

UserInfo Response

```
 {
   "aud":"6cb446b9-33bf-4b97-a6e1-e7ad389974d2",
   "sub":"5aee7c64-a5e1-4899-aad3-51e1f569c292",
   "iss":"https://iam.example.com/default/",
   "email":"mal.bail@DEMOCORP.com"
}
```

Troubleshooting API Errors

To identify log messages related to your application's API call processing, include the "X-CLIENT-TRANSACTION-ID" request header in your API calls. This header's value should be unique to your application and will appear in all log messages generated by the API calls. By using the same value in your application log messages, you can correlate them with the IDSP logs. See also Using Logs to Troubleshoot Operational Errors .

Working with ACR

Working with ACR

The Authentication Context Class Reference (ACR) is a parameter used during authentication to indicate the context in which the authentication takes place. During the creation of an authentication policy in Symantec Identity Security Platform, authentication policy rules can be associated with ACR values to trigger required authentication flows.

This topic includes the following sections:

Using ACR in Authentication Policy

The ACR parameter can be used as a condition in the Authentication policy. The parameter 'acr' is a list of strings, for example:

```
"acr": {
    "operator": "in",
    "value": ["urn:iam:acr:aal:aal2", "urn:iam:acr:aal:aal1"]
}
```

The policy containing the 'acr' rule condition only gets triggered when an incoming authentication request with 'acr' is sent by the client. The incoming 'acr' is then evaluated with the 'acr's' that are specified in the policy. If multiple 'acr's' are specified in the policy rule, then the first 'acr' from the list will be picked for evaluation, and if these values match, then the user is presented with the obligations that are defined in the authentication policy rule. The rest of the 'acr's" will not be evaluated, as a match was already found. However, during evaluation, if the first 'acr' did not match, then the next value in the list will be picked for evaluation. The evaluation process continues until an "acr" match is found.

If the policy evaluation results in a mismatch of 'acr' values, authentication is denied to the user.

Example of a rule containing "ACR" condition

```
{
  "policyName": "ACR Based Policy",
  "description": "Policy with rules based on the ACR condition",
  "status": "active",
  "apps": [
    {
      "id": "059bcbbc-c8b6-467d-85d0-6d15f5a5c0f7"
    }
  ],
  "rules": [
    {
      "conditions": {
        "principal": {
          "user": {
            "operator": "in",
            "value": [
              "user1",
              "user2"
            ]
          }
        },
        "action": {
          "operator": "in",
          "value": [
            "authenticate"
          ]
        },
        "channel": {
           "operator": "in",
           "value": [
                 "web"
           ]
        },
        "acr": {
          "operator": "in",
          "value": [
            "urn:iam:acr:aal:aal1"
          ]
        }
      },
      "result": {
        "effect": "allow",
        "obligations": [
          "PWD:1"
        ],
        "msg": "AAL1 level policy rule"
      }
    },
    {
      "conditions": {
        "principal": {
          "group": {
```

```
          "operator": "in",
          "value": [
            "Administrators"
          ]
        }
      },
      "action": {
        "operator": "in",
        "value": [
          "authenticate"
        ]
      },
      "acr": {
        "operator": "in",
        "value": [
          "urn:iam:acr:aal:aal1"
        ]
      }
    },
    "result": {
      "effect": "allow",
      "obligations": [
        "PWD:1", "FIDO:2"
      ],
      "msg": "AAL1 level policy rule with an administrator user."
    }
  },
  {
    "conditions": {
      "principal": {
        "user": {
          "operator": "in",
          "value": [
            "user1",
            "user2"
          ]
        }
      },
      "action": {
        "operator": "in",
        "value": [
          "authenticate"
        ]
      },
      "acr": {
        "operator": "in",
        "value": [
          "urn:iam:acr:aal:aal3"
        ]
      }
    },
    "result": {
      "effect": "allow",
```

```
        "obligations": [
          "PWD:1", "FIDO:2"
        ],
        "msg": "AAL3 level policy rule"
      }
    }
  ]
}
```

NOTE
To define multiple conditions in an ACR policy, ensure that all those conditions such as 'action' or 'channel' are included in the ACR rule.

When ACR is introduced in a policy, the key-value pair must be passed in the "/authenticate" and "/authorize" calls.

Indicating desired ACR in the '/authentication' call

The 'acrValues' value should be passed in the POST /authenticate call. You can pass multiple 'acrValues' .

This is a sample 'acr' request payload for a /authenticate call:

```
{
  "subject": "malbail",
  "ipaddress": "172.##.#.##",
  "action": "authenticate",
  "acrValues": ["urn:iam:acr:aal:aal2", "urn:iam:acr:aal:aal1"]
}
```

NOTE
When you must add multiple 'acrValues', specify the list of strings in the /authenticate call.

Indicating desired ACR in the OIDC '/authorize' call

The 'acr_values' must be passed along with other query parameters, in the GET /authorize call.

This is a sample 'acr' request payload for an /authorize call:

```
https://test.example.net/default/oauth2/v1/authorize?scope=openid&
                                                client_id=322b2331-e336-46eb-885a-4beaa78784a7&
                                                response_type=id_token token&
                                                redirect_uri=https://www.example.com&
                                                acr_values=urn:iam:acr:aal:aal2 urn:iam:acr:aal:aal1
```

NOTE
If you must specify multiple 'acr values,' the /authorize call only accepts space-separated values.

Results in the ID Token

Once the authentication is successful and you have generated the ID token, the specific "acr" value used during authentication will be contained in the ID token. However, if you did not specify any "acr" values during authentication, then the "acr" value in the ID token will be one of the following based on the authentication factor performed by the user:

- urn:iam:acr:aal3 (If you validated using FIDO or SECURITYKEY as one of the factors)
- urn:iam:acr:aal2 (If you validated using PUSH or OTP as one of the factors)
- urn:iam:acr:aal1 otherwise

Custom Web App with Identity Token Hint

Configure Symantec Identity Security Platform to delegate authentication to a custom web application using a custom flow URL.

IDSP supports different user Multi-factor Authentication (MFA) mechanisms. You can use the MFA mechanisms that are provided by IDSP or if you have your own ways to perform MFA, IDSP can delegate authentication to your configured custom web app using OIDC protocol. If you intend to use your own custom web app to perform authentication, you must design your own authentication policy. IDSP policies will not be applicable to determine the authentication factors.

Follow these steps to integrate your custom web app with IDSP to use its Open ID Connect capabilities:

1. Configure the Application:
   a. Use this API to configure your application:
      `https://${SSP_FQDN}/{{tenantName}}/admin/v1/App`

      NOTE

      - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
      - The default value for `<tenantName>` is "default".
      - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
        For example:
        `https://{{host}}/{{tenantName}}/admin/v1/App`
        should look similar to:
        `https://example.com/default/admin/v1/App`
      - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

   b. Set the "flowURL": The flowURL is your custom web app sign-in URL.
   c. Set the following application parameters:
      - "delegatedAuthentication":"true"- This property specifies whether the configured flow URL application is built with IDSP APIs or not. The default value is false.
      - "autoPostToFlowURL":"true"- This property specifies whether the data should be redirected or posted into the flowURL. The default value is false.

        NOTE
        Configuring this parameter is optional as it depends on whether the custom sign-in URL accepts a POST or a GET request.

      The relevant configuration settings are covered in Application.
2. Configure CORS Origin:
   a. Configure the AllowedOrigins of your custom web app, using the `PATCH "/admin/v1/Configs"` API. See Settings page to know about allowedOrigins setting and see Applications to check the steps to update allowedOrigins setting.
3. Import Certificate:
   a. Add the public certification for the JWT signature verification. The JWT is obtained by IDSP following a successful user authentication at the third party. See Managing Certs for more info on adding certificates.

Integration Between IDSP and Custom Web App

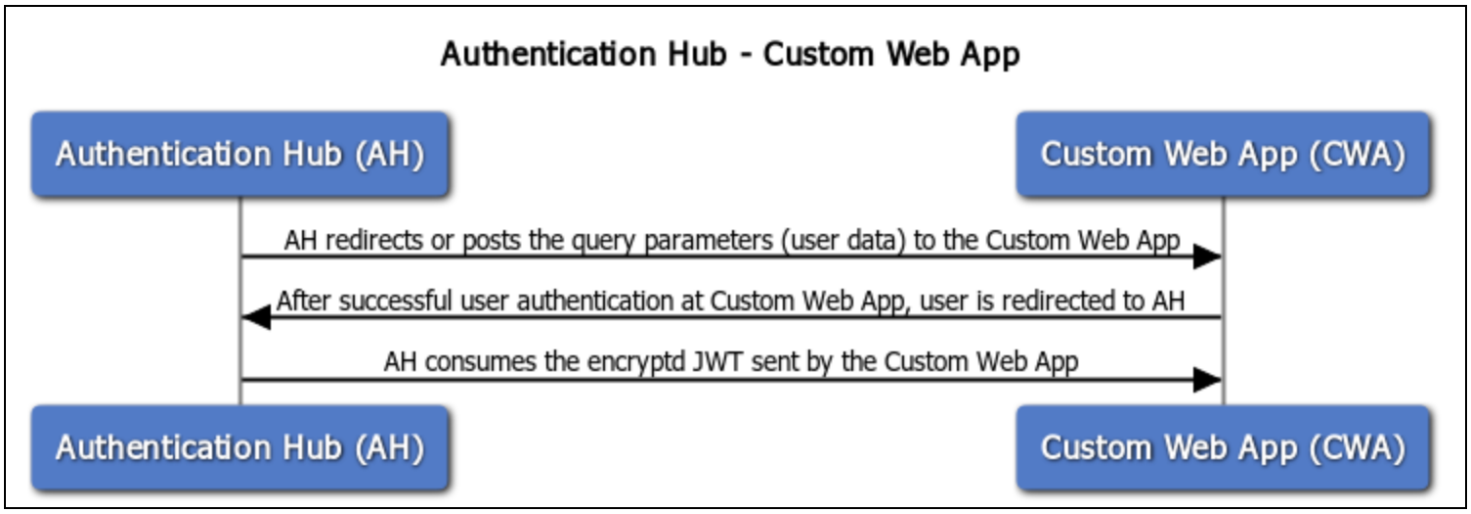
When the Custom Web App interacts with the IDSP to use the open ID connect feature of IDSP, information is exchanged between the two entities. IDSP sends.

The following use case describes the integration between Custom SMS Provider and IDSP.

List of participants and what they represent:

- IDSP- This is the solution which delegates authentication to the third party.
- Custom web App- This is the third party where the authentication takes place.

Figure 18: Custom Web App with ID Token Hint



Flow Description:

1. IDSP redirects or posts the query parameters (user data) to the Custom Web App endpoint.

   NOTE
   IDSP generates a new 'x-flow-state' and 'id_token_hint' (optional). ID Token Hint is generated only when IDSP receives request for a partial delegation from the relying party. ID Token Hint is signed by the private key of IDSP.
2. After user authentication, the user is redirected to IDSP.

   NOTE
   A custom Web App displays the required SignIn actions in the UI. After successful user authentication, the Custom Web App generates required information in the JWT, encrypts it with the public key of IDSP.
3. IDSP consumes the encrypted JWT sent by the Custom Web App.

   NOTE
   At the backend, the encrypted JWT is decrypted using the private key of IDSP. The JWT information is retrieved, flow state details are updated in the cache, and it is used as needed.

   Following successful authentication, the flow of obtaining tokens can be accomplished by invoking the token endpoints and user information endpoints. See Authorization Server Endpoints.

Data sent to the Custom Web App

IDSP sends the following query parameters to the Custom Web Application in the form of an ID token:

Parameter Name	Value	Mandatory Fields
x-flow-state		Yes
x-service-baseurl	<IDSP Issuer URL>	Yes
x-callback-url	<IDSP Issuer URL>**/oauth2/v1/authcomplete**	Yes
id_token_hint		Optional. Only if received from the relying party.

X-CLIENT-TRANSACTION-ID		Optional. Only if received from the relying party.
x-app-id	IDSP application Id	Yes
ui_locales	UI Locale information that is received from the relying party.	Optional. Only if received from the relying party.
signinExpMins	The number of minutes before the active sign-in flow expires.	Optional.

Data sent to the IDSP

The Custom Web Application sends the following parameters to the IDSP using the POST /oauth2/vi/authcomplete API:

Parameter Name	Value	Mandatory Fields
x-flow-state	Received from IDSP	Yes
userContext	Encrypted JWT	Yes
X-CLIENT-TRANSACTION-ID	Received from IDSP	Optional. Only if received from the relying party.

User Context Claims sent to IDSP

IDSP expects the following userContext claims in the JWT sent by the Custom Web App:

Claim Name	Claim Description	Data Type	Full Delegation Mandatory/ Optional	Partial Delegation
user_loginid	User login identifier	String	Yes	**Not available in VIPLogin**
user_loginidattrmappingname	User attribute name in the user store that is mapped to the user login id.	String	Yes	
user_universalid	User universal id mapped in the store	String	Yes	
user_riskscore	The user risk score, if any	Number	Optional	
user_dn	User DN	String	Yes	
amr	Authentication methods used for the user authentication Standard AMR	Array	Yes	Yes
sid	User Session id, if any	String	Optional	Yes
email	User Email	String	Yes	
phone_number	User Phone Number	String	Yes	
auth_time	The time at which the user authentication is complete.	Epoch Time	Yes	Yes
idp_name	User Store Name	String	Yes	
idp_guid	User Store GUID	String	Yes	
idp_type	User Store Type	String	Yes	
x-flow-state	x-flow-state received for the check	String	Yes	Yes

Standard attributes exp, nbf, iss, azp, etc.			Yes	Yes
error	Error code as per OIDC spec		Yes, if there is a failure scenario.	Yes, if there is a failure scenario.
error_description	Error Message		Yes, if there is a failure scenario.	Yes, if there is a failure scenario.

Integrating with Custom Service Provider Interface (SPI)

Overview of authentication delegation using custom flow URLs in Symantec Identity Security Platform.

By delegating authentication to an external web app, you can implement bespoke identity verification steps while maintaining Symantec Identity Security Platform as the primary orchestrator.

This topic includes the following:

- Custom Provider Support
- Custom Provider SPI
- Steps to Integrate SPI
- Open API Spec
- Custom Provider APIs
- Examples of Custom Provider APIs
- List of attributes and their descriptions used in the Custom Provider API request
- Security Method Types
- Custom Provider API Error Codes
- Sharing IDSP context with SPI

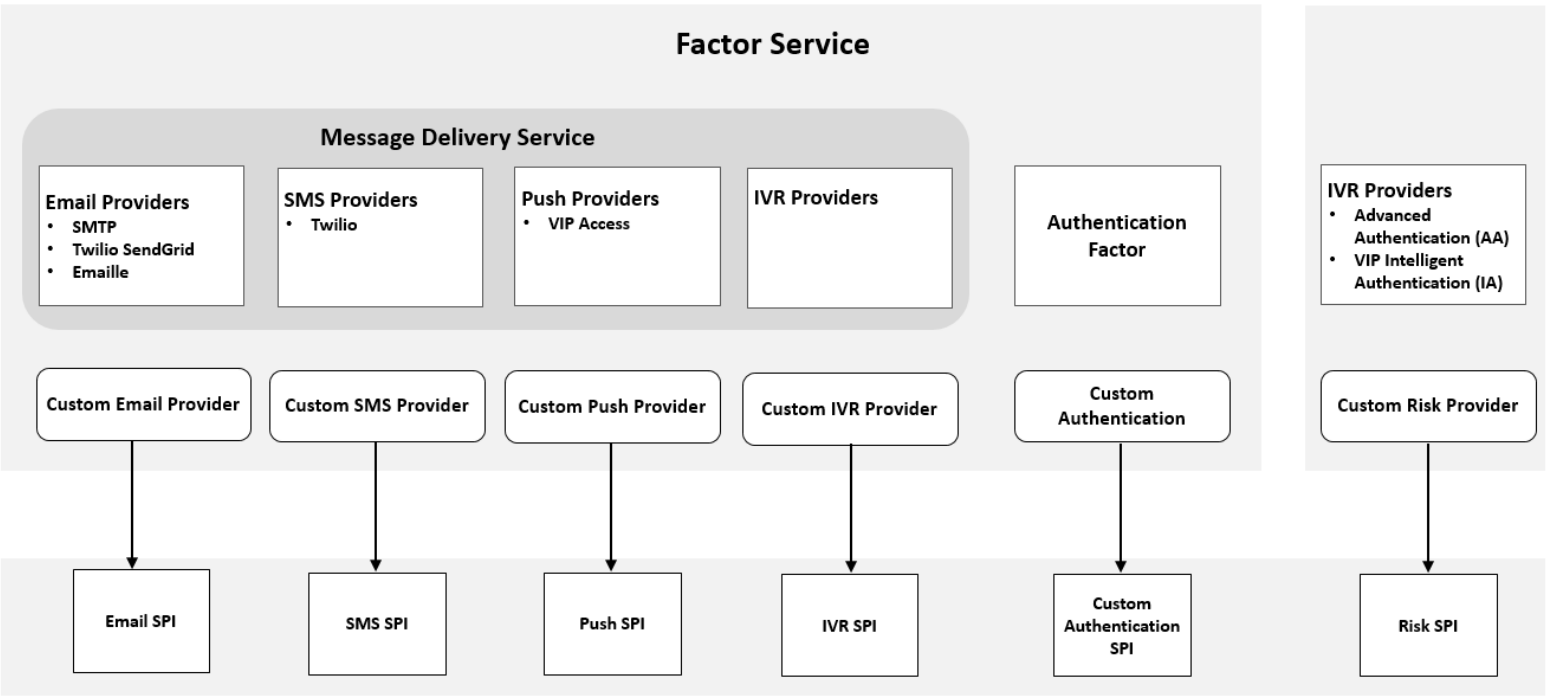
Custom Provider Support

Symantec Identity Security Platform extends Custom Provider support using Service Provider Interface (SPI) to its customers. This feature can be used by the customers to leverage their existing investment in message delivery infrastructure and risk engines with our solution. The Custom SPI support is extended to message delivery channels that include SMS, EMAIL, PUSH, and IVR.

IDSP exposes an SPI for each of the supported custom operations in the solution. The SPI uses an Open API spec and expects the customers to implement it as a separate web service that runs in their environment. The customer should implement the APIs that are defined by the solution for each service to enable the solution's custom SPI to communicate with the customer's service provider.

Custom Provider Support Using SPI for all supported services

The solution exposes a separate Service Provider Interface for each operation. Each interface can be implemented, deployed, and configured with the solution separately as shown in the following diagram:



Each message delivery service and the risk service has its own SPI that is exposed to the customer by the IDSP. Using this Open API specification, the customers can implement their custom provider as a REST service in their own environment.

Steps to Integrate with Custom SPI

The three steps that the Customers have to perform to integrate with the SPI are:

1. Develop their custom service based on the Open API Spec provided by IDSP.
2. Create a custom provider in IDSP using the Provider endpoint- https://<SSP_FQDN>/<tenantName>/admin/v1/CustomProviders
3. Specify the configuration settings in IDSP.

Common Open API Spec for all Custom Providers (Yaml File)

Click IDSP SPI Spec to view the consolidated Open API specification YAML file for Custom SPI.

Custom Provider APIs

The following table lists the Custom Provider endpoint URLs that are available for the admin to invoke and perform CRUD (create, retrieve, update, and delete) operations on all the custom providers that are added to the solution. These operations can be performed by both, Tenant and System admins.

For sample request and response payloads of all these URLs, refer to Custom Provider Swagger content.

Operation Description	Operation Type	Endpoint URLs
Creating a new Custom Provider	POST	https://${SSP_FQDN}/<tenantName>/admin/v1/CustomProviders
Retrieving Custom Provider based on Provider Id	GET	https://${SSP_FQDN}/<tenantName>/admin/v1/CustomProviders/<providerId>

Updating Custom Providers based on Provider Id	PUT	https://${SSP_FQDN}/<tenantName>/admin/v1/CustomProviders/<providerId>
Retrieving all Custom Providers	GET	https://${SSP_FQDN}/<tenantName>/admin/v1/CustomProviders
Deleting a specific Custom Provider based on Provider Id	DELETE	https://${SSP_FQDN}/<tenantName>/admin/v1/CustomProviders/<providerId> To forcefully delete a custom provider, including those currently in use, see Resource Deletion in IDSP.

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/CustomProviders`
  should look similar to:
  `https://example.com/default/admin/v1/CustomProviders`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Samples of POST and PUT call for Custom Providers

Sample of Custom Provider POST call with "securityMethod" as apikey

The following is a sample that shows the request payload of Custom Provider POST call with **securityMethod-apiKey**.

```
{
"providerName": "customRiskProvider",
"version": "1.0",
"apiKey": "6fd06fd82aceb78b0bbbe3990b865f96ea",
"url": "https://localhost:8091",
"securityMethod": "apikey",
"certAlias": "<certificateAlias>",
"ignoreSSLValidation": false,
"providerHeaders": [
   {
       "name": "clientId",
       "value": "8289-b1e8817a3fec"
   }
],
"providerProperties": [
   {
       "name": "app",
       "value": "test"
   },
   {
       "name": "region",
       "value": "us-west"
   }
]
}
```

The value in the "providerHeaders" and "providerProperties" attributes of the Custom Providers config can contain dynamic environment variables. When used, these variables are substituted with their corresponding value in runtime. See Dynamic Environment Variable Substitution.

Sample of Custom Provider POST call with "securityMethod" as basic

The following is a sample that shows the request payload of Custom Provider POST call with **securityMethod-basic**.

```
{
"providerName": "customRiskProvider",
"version": "1.0",
"clientId": "<clientId>",
"clientSecretAlias": "<clientSecretAlias>",
"url": "https://localhost:8091",
"securityMethod": "basic",
"certAlias": "<certificateAlias>",
"ignoreSSLValidation": false,

"providerHeaders": [
   {
       "name": "clientId",
       "value": "8289-b1e8817a3fec"
   }
],
"providerProperties": [
   {
       "name": "app",
       "value": "test"
   },
   {
       "name": "region",
       "value": "us-west"
   }
]
}
```

NOTE
In the case of "securityMethod" as “basic”, the userid is passed through the clientId, and the password is passed through the clientSecretAlias.

Sample of Custom Provider POST call with "securityMethod" as oauth

The following is a sample that shows the request payload of Custom Provider POST call with **securityMethod-oauth**.

```
{
"providerName": "customRiskProvider",
"version": "1.0",
"url": "https://localhost:8091",
"certAlias": "<certificateAlias>",
"ignoreSSLValidation": false,
"clientId": "clientId",
"clientSecretAlias": "clientSecretAlias",
"oauthScopes": "scopes",
"securityMethod": "oauth",
"sslMode": "VERIFY_CA",
"oauthTokenUrl": "http://<hostname>/<tenantName>/oauth2/v1/token",
"oauthTokenIgnoreSslValidation": true,
```

```
"providerHeaders": [
   {
       "name": "clientId",
       "value": "8289-b1e8817a3fec"
   }
],
"providerProperties": [
   {
       "name": "app",
       "value": "test"
   },
   {
       "name": "region",
       "value": "us-west"
   }
],
"oauthResourceInAzRequest": "none",
"oauthTokenCertificateAlias": "",
"oauthTokenRequestCtxBase64": "",
"oauthTokenSslMode": "VERIFY_CA",
"oauthTokenReadTimeoutMillis": "10000",
"oauthTokenConnectTimeoutMillis": "10000",
"skewTimeSecs": "0"
}
```

Sample of Custom Provider POST call with "securityMethod" as oauthassertion

The following is a sample that shows the request payload of Custom Provider POST call with **securityMethod-oauthassertion**.

```
{
"providerName": "customRiskProvider",
"version": "1.0",
"url": "https://localhost:8091",
"certAlias": "<certificateAlias>",
"ignoreSSLValidation": false,
"skipUserStrikeCountIncrement": false,
"clientId": "<clientId>",
"clientSecretAlias": "<clientSecretAlias>",
"oauthScopes": "<scopes>",
"sslMode": "VERIFY_CA",
"securityMethod": "oauthassertion",
"oauthTokenUrl": "http://<hostname>/<tenantName>/oauth2/v1/token",
"oauthAssertionAud": "http://<hostname>/<tenantName>/",
"oauthResourceInAzRequest": "none",
"oauthTokenIgnoreSslValidation": true,
"oauthAssertionKidType": "x5t",
"providerHeaders": [
   {
       "name": "clientId",
       "value": "8289-b1e8817a3fec"
   }
],
"providerProperties": [
   {
```

```
        "name": "app",
        "value": "test"
    },
    {
        "name": "region",
        "value": "us-west"
    }
],
"oauthTokenCertificateAlias": "",
"oauthTokenRequestCtxBase64": "",
"oauthTokenSslMode": "VERIFY_CA",
"oauthTokenReadTimeoutMillis": "10000",
"oauthTokenConnectTimeoutMillis": "10000",
"skewTimeSecs": "0"
}
```

List of attributes and their descriptions used in the Custom Provider API request

Table 82: Custom Provider attributes and their descriptions

Attribute Name	Mandatory Fields	Description
providerName	Yes	Specify the name of the custom provider to be used.
version	Yes	Specify if any particular version of the provider must be picked.
apiKey	No	This is a unique passphrase or key that is used by Custom Provid identify the calling application. The value of the "securityMethod" for this model is - "securityMethod":"apikey" In the case of the "securityMethod" property as "apikey", the IDSP checks for the value of the "apiKey" property. If the property value empty, then the apiKey value should be added using the Secrets (/admin/v1/Secrets/) and the name of the created secret should b provided in the clientSecretAlias parameter.
url	Yes	The location where the provider is hosted.
certAlias	No	This is the name of the certificate alias pointing to IDSP's certifica The certificate should be added using the https://<SSP_F <tenantName>/admin/v1/Certs API. This attribute should be provided if "ignoreSSLValidation" is set to
sslMode	No	This indicates whether SSL validation should be performed or ign the "sslMode" is enabled, it should validate the server's identity. Possible Values are - "VERIFY_CA", "VERIFY_IDENTITY". If the "sslMode" is not indicated, it defaults to ""VERIFY_CA".
ignoreSSLValidation	No	This boolean indicates if the SSL validation is to be ignored or no The default is set to "false".

spiReadTimeoutMillis	No	This is the value of time in milliseconds that will be used on waitin read data. However, the value of spiReadTimeoutMillis property will be contr the configuration set at the ingress controller when the value of "c timeout" set in the ingress is less than the value of "spiReadTime set in the SPI. For example: If the "spiReadTimeoutMillis" is set to 120000 millis (2mins) in the SPI configuration and the read-timeout is set to 60 in the ingress controller, then the ingress controller timeout takes precedence over the spiReadTimeoutMillis value. The default value is 60000 and it allows a value greater than 0.
spiConnectTimeoutMillis	No	This is the value of time in milliseconds that will be used in makin initial connection. However, the value of spiConnectTimeoutMillis will be controlled by the configuration set at the ingress controller the value of "connect-timeout" set in the ingress is less than the v "spiConnectTimeoutMillis" set in the SPI. For example: If the "spiConnectTimeoutMillis" is set to 120000 milliseconds (2 mins) in the SPI configuration and the connect-tim is set to 60 seconds in the ingress controller, then the ingress con timeout takes precedence over the spiConnectTimeoutMillis value The default value is 60000.
providerHeaders	No	These are key-value pairs set as part of the header while invoking endpoint. Whenever the securityMethod="apikey", the header value passed the **providerHeaders** is considered. However, in the case of the 3 security methods (basic, OAuth, oauthassertion) the value of th **requestCtxBase64** is considered and the value of the providerHe ignored.
providerProperties		These are key-value pairs that contain additional information that required by the provider to perform its functions. This needs to be while invoking the SPI endpoint.
clientId	No	Client Id is used to connect to the SPI host. This is mandatory, if "*securityMethod*" is "OAuth" or "oauthassertion" or "basic". • For securityMethod="basic" - clientId corresponds to the User • For securityMethod="OAuth" or "oauthassertion" - clientId corresponds to the Oauth2 clientId

clientSecretAlias	No	Alias to a secret of the type as required by "*securityMethod*" prop The secret is expected to contain client authentication details req connect to the SPI - apiKey, basic, OAuth, oauthassertion. • In the case of the "securityMethod" property as "apikey", the I checks for the value of the "apiKey" property. If the property v empty, then the apiKey value should be added using the Secr (/admin/v1/Secrets/) and the name of the created secret shou provided in the clientSecretAlias parameter. See Managing Se • In the case of the "securityMethod" property as "basic", the pa should be added using the Secrets API (/admin/v1/Secrets/) a name of the created secret should be provided in the clientSe parameter. • In the case of "securityMethod" property as "OAuth", the clien should be added using the Secrets API(/admin/v1/Secrets/) a name of the created secret should be provided in the clientSe parameter. • In the case of "securityMethod" property as "oauthassertion", secret of type RSAKEY must be created containing the RSA private key to sign the assertion using the keys API(/admin/v1 Keys/) and the name of the created RSAKEY should be provi the clientSecretAlias parameter. • In the case of "securityMethod" as "oauthassertion", the "clientSecretAlias" property is required when "oauthAssertionKidType" is "secret" or "skid". Refer to the "oauthAssertionKidType" property mentioned in this table for on when "clientSecretAlias" is required.
requestCtxBase64	No	This is a 64 Bit encoded string of the Context map in key-value pa used to execute when invoking SPI endpoints - such as sending s headers. This parameter can be used as a substitute property for providerH
oauthScopes	No	Space delimited, optional OAuth scopes to request the access tol
securityMethod	Yes	Type of security method to be used to connect. Values are "basic "OAuth", "oauthassertion", and "apikey". The default value is "apil For more information about the securityMethods and its associate mandatory properties that are required while creating a Custom P see securityMethod Required Properties.
oauthTokenUrl	No	URL for the Token generation. This is mandatory if "*securityMeth* "OAuth" or "oauthassertion".

oauthResourceInAzRequestCsv	No	Represents the RFC8707 resource indicator for OAuth STS requ comma-separated values (csv) format. These are the request par that enable a client to explicitly signal to an authorization server a the identity of the protected resource(s) to which it is requesting a based on oauthResourceInAzRequest parameter value. Whenev oauthResourceInAzRequest is configured as "all", the values are in the csv format as given in the example. Possible values are "none", "all" and "base".
oauthResourceInAzRequest	No	Represents whether to add RFC8707 resource indicator for OAut request. When oauthResourceInAzRequest is configured as "all", resource values given in oauthResourceInAzRequestCsv are pas during the OAuth STS request. Possible values are "none", "all" and "base"
oauthAssertionKidType	No	Used when the "securityMethod" is set to "oauthassertion". It indi how to establish the kid of the signed assertion. Possible Values: skid, x5t, secret If "oauthAssertionKidType" is "secret", then the "kid" value of the assertion is populated from the "kid" value of the rsakey secret id by “clientSecretAlias". If "oauthAssertionKidType" is "skid", then the "kid" value of the as is populated from the "skid" value of the certificate stored in the rs secret identified by clientSecretAlias. If clientSecretAlias is not pr the signing key of the tenant is used. If "oauthAssertionKidType" is "x5t", then the "kid" value of the ass populated from the x5t value of the certificate stored in the rsakey identified by clientSecretAlias. If clientSecretAlias is not provided, signing key of the tenant is used.
oauthAssertionAud		When using "*securityMethod*" as "oauthassertion", this attribute in the audience to inject into the assertion. If this attribute is not spe the value of the oauthTokenUrl will be injected.
oauthTokenIgnoreSslValidation	Yes	This attribute indicates if the SSL validation should be ignored or default is set to "false".
oauthTokenSslMode	No	This attribute indicates if the SSL validation should be enabled. Possible Values are - "VERIFY_CA", "VERIFY_IDENTITY". If the "sslMode" is not indicated, it defaults to "VERIFY_CA".
oauthTokenCertificateAlias	No	This is the name of the certificate alias pointing to IDSP's "/Certs" This attribute should be provided if "ignoreSSLValidation" is set to false. The certificate or certificate chain pointed to by the alias sh added using the "/Certs" API.
oauthTokenRequestCtxBase64	No	This is a base64 encoded string of the Context map in key-value
oauthTokenReadTimeoutMillis	No	This is the value of time in milliseconds that will be used on waitin read the data.
oauthTokenConnectTimeoutMillis	No	This is the value of time in milliseconds that will be used in makin initial connection.

skewTimeSecs	No	When using securityMethod="oauthassertion", this property is hel cases when the target STS has no capability to adjust time comp for skew time. In that case, an integer "skewTimeSecs" (could be or negative) is used to offset the "exp" and "nbf" claims. In case o securityMethod as "oauthassertion", the clientassertion expiry = 6 1000 miliseconds (10 mins) + skewTimeSecs x1000 miliseconds. Here,10 minutes is the constant value.
skipUserStrikeCountIncrement	No	This boolean setting determines whether the strikeCount update s be skipped during a custom authentication failure for the user. If s true, a failed login attempt does not increase the strikeCount in th record, which means the user does not get closer to being locked The default is set to "false".

Sample Response of a Custom Provider POST Call

The following sample shows the **response of a Custom Provider POST call**. The response returns a ***providerId*** that must be used in the subsequent API calls to perform GET/PUT or DELETE operations.


```
{
"providerName": "customRiskProvider",
"version": "1.0",
"apiKey": "gJuDj15N8iLeQa08kuX20gD0nID4jsXo/XEDdCyjieHBLd+R7SE+ZpPgwBXl1CWw",
"certAlias": "",
"url": "http://localhost:8091",
"providerHeaders": [
   {
       "name": "clientId",
       "value": "8289-b1e8817a3fec"
   }
],
"providerProperties": [
   {
       "name": "app",
       "value": "test"
   },
   {
       "name": "region",
       "value": "us-west"
   }
],
"updatedBy": {
   "principalType": "CLIENT",
   "principalId": "1fc0e655-b5a1-4cb5-9251-5828d9769029",
   "principalName": "defaulttenantclient"
},
"updatedDateTime": "2021-12-01 05:46 AM UTC",
"origin": "tenant",
"ignoreSSLValidation": true,
"providerId": "8b59cf35-61db-45fc-8744-6eef9f02876b",
"sslMode": "VERIFY_CA",
"requestCtxBase64": "",
"clientId": "c801e8ee-b581-4365-aa1a-9c61a8bc7012",
"clientSecretAlias": "samplealias",
```

```
"oauthScopes": "test",
"securityMethod": "apiKey",
"oauthTokenUrl": "http://oauth:8091",
"oauthResourceInAzRequest": "none",
"oauthAssertionKidType": "",
"oauthAssertionAud": "",
"oauthTokenIgnoreSslValidation": true,
"oauthTokenSslMode": "VERIFY_CA",
"oauthTokenCertificateAlias": "",
"oauthTokenRequestCtxBase64": "d06fd82a90b865f96eadfdsf",
"oauthTokenReadTimeoutMillis": "10000",
"oauthTokenConnectTimeoutMillis": "10000",
"skewTimeSecs": "10"
 }
```


Sample request of a Custom Provider PUT call

The following sample shows the **request of a Custom Provider PUT call**:


```
{
"version": "1.0",
"apiKey": "6fd06fd82aceb78b0bbbe3990b865f96ea",
"url": "https://localhost:8091",
"securityMethod": "apikey",
"certAlias": "<certificateAlias>",
"ignoreSSLValidation": false,

"providerHeaders": [
   {
       "name": "clientId",
       "value": "8289-b1e8817a3fec"
   }
],
"providerProperties": [
   {
       "name": "app",
       "value": "test"
   },
   {
       "name": "region",
       "value": "us-west"
   }
]
}
```


Sample response of a Custom Provider PUT call

The following sample shows the **response of a Custom Provider PUT call**:


```
{
   "providerName": "customRiskProvider",
   "version": "1.0",
   "apiKey": "9nlsomQ4CZkz4jOOHT/3cQ==",
   "certAlias": "spiip",
   "url": "https://localhost:8094/",
```

```
    "providerHeaders": [
        {
            "name": "clientId",
            "value": "8289-b1e8817a3fec"
        }
    ],
    "providerProperties": [
        {
            "name": "app",
            "value": "test"
        },
        {
            "name": "region",
            "value": "us-west"
        }
    ],
    "origin": "tenant",
    "ignoreSSLValidation": true,
    "providerId": "bdac5a70-b145-46f9-9e8c-22969269e70e",
    "sslMode": "VERIFY_CA",
    "spiReadTimeoutMillis": "60001",
    "spiConnectTimeoutMillis": "60002",
    "requestCtxBase64": "",
    "clientId": "5798b507-4afa-4a19-8ec5-c7c05adebad0",
    "clientSecretAlias": "sspkeyoauth",
    "oauthScopes": "urn:iam:myscopes",
    "securityMethod": "oauthassertion",
    "oauthTokenUrl": "https://shiny-3424-ssp.sspdev.dev.broadcom.com/default/oauth2/v1/token",
    "oauthResourceInAzRequest": "none",
    "oauthAssertionKidType": "x5t",
    "oauthAssertionAud": "https://Nbruce-3424-ssp.sspdev.dev.broadcom.com/default/",
    "oauthTokenIgnoreSslValidation": true,
    "oauthTokenSslMode": "VERIFY_CA",
    "oauthTokenCertificateAlias": "",
    "oauthTokenRequestCtxBase64": "",
    "oauthTokenReadTimeoutMillis": "10000",
    "oauthTokenConnectTimeoutMillis": "10000",
    "skewTimeSecs": "0"
}
```


securityMethod Types and Associated Mandatory Properties

The following table lists the securityMethods and their associated mandatory properties that are required while creating a Custom Provider:

Table 83: securityMethod Types and its Associated Mandatory Properties

securityMethod Type	Required Properties for the securityMethod Type	Notes
apikey	apikey or clientSecretAlias	The apiKey value must be added using the Secrets API (/admin/v1/Secrets/) and the name of the created secret must be provided in the clientSecretAlias property. See Managing Secrets.
basic	clientId (username), clientSecretAlias (Password)	The password must be added using the Secrets API(/admin/v1/Secrets/) and the name of the created secret must be provided in the clientSecretAlias property.
OAuth	clientId (clientId), clientSecretAlias (secret), oauthTokenUrl, oauthScopes	The client secret must be added using the Secrets API(/admin/v1/Secrets/) and the name of the created secret must be provided in the clientSecretAlias property. The header key-value pair that is associated with the OAuth TokenServer can be passed through the oauthTokenRequestCtxBase64 property.
oauthassertion	clientId (clientId), clientSecretAlias (key), oauthTokenUrl, oauthScopes, oauthAssertionAud, oauthAssertionKidType(secret)	In the case of oauthAssertionKidType as secret, the signature details must be added using the keys API(/admin/v1/Keys/) and the name of the created key must be provided in the clientSecretAlias parameter. The clientSecretAlias property is required when oauthAssertionKidType is secret or skid. The header key-value pair that is associated with the OAuth TokenServer can be passed through the oauthTokenRequestCtxBase64 property.

Custom Provider API Error Codes

The following table lists the error codes that the Custom Provider API returns:

Table 84: Error Codes for Custom Provider API

Error Codes	Error Messages
8800049	Invalid configuration ({0})
8800050	Length of parameter ({0}) more than expected
8800051	Invalid configuration - missing required configuration parameter(s) ({0})
8800052	Provided certificate Alias is not valid
8800053	Provided the secret alias is not valid
8800054	Provided secret Alias is not having the required type 'rsakey' in the secrets store
8800055	Internal Server Error

Sharing IDSP Context with SPI

IDSP sends the current operational state as the context in each SPI call. The following table lists the supported key-value pairs of the IDSP context that is shared in the **ahContext** request parameter in a SPI call:

Reserved Key	Description	Example
x-ssp-base-uri	Specifies the base URL of the IDSP deployment.	https://ssp.ssp10.bm.com
x-ssp-issuer-url	Specifies the logical tenant-specific issuer that signs the IDSP tokens.	https://ssp.ssp10.bm.com/default
x-SSP_FQDN	Specifies the FQDN of the host that is assigned to the IDSP service.	ssp.ssp10.bm.com
x-ssp-path	Specifies the tenant-specific component of the IDSP URL.	/default
clientTxnId	Specifies the request ID as indicated by a client that is making a request to IDSP.	myClientTxnID
txnId	Specifies the unique transaction ID that is asserted by IDSP for every received request.	<guid>
tname	Specifies the name of the tenant that receives a request.	default
tid	Specifies the ID of the tenant that receives a request.	<guid>
clientIp	Specifies the IP address of the client that is making a request.	10.76.13.107
clientTid	Specifies the ID of the tenant that owns the client, which is making a request.	<guid>
clientId	Specifies the ID of the client that is making a request.	<guid>
userIp	Specifies the IP address of the user on whose behalf a client is making a request.	1.1.1.2
reauthIsRequest	Specifies that re-authentication was requested by a client that is making a request.	false
reauthIsPolicy	Specifies that re-authentication was requested by an evaluated policy.	false
reauthAmrNames	In case of re-authentication, specifies the AMR values from an ID Token.	"OTP","PWD","FIDO","MYAUTH"
completedAmrNames	Specifies the AMR values of the completed authentication factors.	"OTP","MYAUTH"
locale	Specifies the current locale as indicated by the server.	en-US
userId	Specifies the internal ID of the user being authenticated.	<guid>
userDN	Specifies the DN of the user being authenticated.	cn=bob, ou=org, o=acme, dc=com

userLoginId	Specifies the login ID of the user being authenticated.	bob
userUniversalId	Specifies the universal ID of the user being authenticated.	12345
userStatusCode	In case of ID store status encoding = "Siteminder", specifies the user's "disabled state" as encoded per SiteMinder's convention.	"Disabled flag" bitmask. For information, see this SiteMinder Knowledge Base article.
userStatusEncoding	Specifies the type of the ID store status encoding, as configured for the ID store that contains the user who isbeing authenticated.	Siteminder, oud, cadir, etc.
userIdTokenPresent	Specifies whether an ID Token was provided during the initiation of the authentication flow.	true
userIdToken	Specifies the JSON string that contains the ID Token as provided during the initiation of the authentication flow.	false
hideAuthFailureReason	Specifies whether an authentication failure reason must be returned.	true
idpUserGuid	Specifies the unique ID of the user as maintained in the ID store or provided in an ID Token.	Usually a <guid>
idpType	Specifies the type of the identity provider that is used to assert the identity being disambiguated. In ZFP mode with SiteMinder integration, this may contain different types such as "siteminder".	LDAP or SCIM
idpName	Specifies the name of the identity provider that is used to assert the identity being disambiguated.	Name of the ID store or as provided via ZFP token
idpGuid	Specifies the ID of the identity provider that is used to assert the identity being disambiguated.	ID of the ID store or as provided via a ZFP token
providerName	Specifies the name of the custom provider that is implementing the current SPI.	MyCustomProvider
providerId	Specifies the ID of the custom provider that is implementing the current SPI.	<guid>
appId	Specifies the ID of the application on whose behalf the user is being authenticated.	MyApp

Integrating with External Risk Provider

Learn how the Symantec Identity Security Platform Custom Service Provider Interface (SPI) integrates with a custom risk provider for risk-based authentication.

This topic includes the following:

- Custom Provider Overview
- Implementing the APIs
- API Specification
- Open API Schema - (Swagger Content)
- Supported Use Cases

Custom Provider Overview

The IDSP Custom Service Provider Interface (SPI) allows the solution to be integrated with a custom risk provider for risk-based authentication. The SPI uses an Open API spec and expects customers to implement it as a separate web service that runs in their environment. The customer should implement a set of APIs that are defined by the solution to enable the solution's custom SPI to communicate with the customer's risk provider. These APIs have been explained in the subsequent section.

Implementing the APIs

The following APIs should be implemented by the customer to integrate their risk provider with the solution's custom SPI:

- **UserRiskScoreEvaluator** - This API request is used to evaluate the user risk and return a risk profile (which includes risk score and risk advice) apart from additional risk profile details which may be required for post user evaluation. For example, a device tag. The use of risk profile differs based on the configuration that is set for the parameters, **riskProviderPrePrimaryAuth** and **riskProviderPrimaryAuthSuccess** in the global/tenant settings. See Settings for more information.
- **PostUserRiskScoreEvaluator** - This API request is used to update the risk context for a user whose risk has been evaluated as part of the /UserRiskScoreEvaluator API. If the UserRiskScoreEvaluator returns risk advice of stepup or allow, this API is invoked.

The API request and response templates are based on the standard Open API spec that IDSP expects the customers to provide during the integration process. These specifications have been explained in the subsequent sections.

API Specification

The API request and response templates are explained below.

UserRiskScoreEvaluator API Specification

UserRiskScoreEvaluator API Specification	
API Name	UserRiskScoreEvaluator
API URL	*https://<CustomProviderHost>/UserRiskScoreEvaluator*
API OPERATION	*POST*
AUTHORIZATION	*<API Key shared by the custom risk provider>*
API RESPONSE CODE	*200 Success, 400 Failures, 500 Server Errors*

Request Payload includes the following parameters:

Parameter	Type	Mandatory (Yes/No)	Description
X-API-KEY	Header	No	API key used to invoke the custom risk provider.
subject	string	Yes	Unique identifier of the user.

Parameter	Type	Mandatory (Yes/No)	Description
clientContext	array of key-value pairs	Yes	Client context parameters received as part of the "/auth/v1/authenticate" call will be passed here by the client. Example: transactionIdentifier, appid `{ "subject": "<userLoginid>", "channel": "web", "ipAddress": "10.10.123.124", "action":"authenticate", "rememberMe" : true, "device": { "signature": { "iaAuthData": "_v02MyUgNGgYOi88OTk0....." } }, "clientContext": [{ "key": "transactionIdentifier", "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40" }] }`
ahContext	array of key-value pairs	No	Contains the IDSP context parameters. For information, see the *Sharing IDSP Context with SPI* section in Integrating with Custom SPI.
providerContext	array of key-value pairs	No	Parameters that are configured as part of the custom risk provider.

Sample Request of "/UserRiskScoreEvaluator" API:

```
{
"subject": "nbruce",
"clientContext": [
{
"key": "transactionIdentifier",
"value": "gga323a24"
}
],
"ahContext": [
{
"key": "clientTxnId",
"value": "cd2f1e6e-c356-49ae-a5b8-c10bf571a9a0"
},
{
"key": "txnId",
"value": "txn-id-hd26se23"
}
],
"providerContext": [
{
```

```
"key": "instanceId",
"value": "brazil-instance-1"
}
]
}
```

Response Payload should include the following parameters:

Parameter	Type	Mandatory (Yes/No)	Description
subject	string	Yes	Unique identifier of the user.
riskScore	int	Yes	Risk score returned by the risk service.
riskReason	string	No	Reason for the risk score and risk advice.
riskFactors	array of strings	No	An array of risk factors that resulted in the risk score by the risk service.
riskAdvice	string	Yes	This attribute provides the risk advice which results in one of the following: • ALLOW - Ignores the remaining obligations and allows the user to continue the transaction. • DENY - User is denied from performing any further transactions. • STEPUP - Continues the authentication flow with the remaining obligations that are defined in the policy.
msg	string	Yes	If the status of the transaction is successful, returns "Risk evaluation success". If the status of the transaction is a failure, return the reason for the failure.
riskEvalContext	array of key-value pairs	No	All the parameters that must be returned as part of the post evaluate risk call should be sent in this field (example: transactionid, deviceid)

Sample Response of "/UserRiskScoreEvaluator" API:

```
{
"subject": "nbruce",
"riskScore": 55,
"riskReason": "Registered device not detected. Possibly shared remembered device",
"riskFactors": [
"Device Recognition"
],
"riskAdvice": "deny",
"msg": "Risk evaluation success",
"riskEvalContext": [
{
"key": "transactionId",
"value": "txnId-6hdds733"
},
{
"key": "deviceId",
"value": "device-hdyn235sd"
```

```
}
]
}
```

PostUserRiskScoreEvaluator API Specification

PostUserRiskScoreEvaluator API Specification	
API Name	PostUserRiskScoreEvaluator
API URL	*https://<CustomProviderHost>/PostUserRiskScoreEvaluator*
API OPERATION	*POST*
AUTHORIZATION	*<API Key shared by the custom risk provider>*
API RESPONSE CODE	*200 Success, 400 Failures, 500 Server Errors*

Request Payload includes the following parameters:

Parameter	Type	Mandatory (Yes/No)	Description
X-API-KEY	Header	No	API key used to invoke the custom risk provider.
subject	string	Yes	Unique identifier of the user.
authStatus	string	Yes	Status of the authentication (possible values - success/ failure).
riskEvalContext	array of key-value pairs	No	Add the parameters that are returned as part of the *UserRiskScoreEvaluator* call in this field (example: transactionid, deviceid). Add only the parameters and not the entire payload as part of this field.
ahContext	array of key-value pairs	No	Contains the IDSP context parameters. For information, see the *Sharing IDSP Context with SPI* section in Integrating with Custom SPI.
providerContext	array of key-value pairs	No	All the providerContext received in the "/ UserRiskScoreEvaluator" call will be returned back in the post risk eval

Sample Request of "/PostUserRiskScoreEvaluator" API:

```
{
"subject": "nbruce",
"authStatus": "success",
"riskEvalContext": [
{
"key": "transactionId",
"value": "txnId-6hdds733"
},
{
"key": "deviceId",
"value": "device-hdyn235sd"
}
],
"ahContext": [
{
"key": "clientTxnId",
```

```
"value": "cd2f1e6e-c356-49ae-a5b8-c10bf571a9a0"
},
{
"key": "txnId",
"value": "txn-id-hd26se23"
},
{
"key": "isTrustedDevice",
"value": "true"
}
],
"providerContext": [
{
"key": "instanceId",
"value": "brazil-instance-1"
}
]
}
```


Response Payload should include the following parameters:

Parameter	Type	Mandatory (Yes/No)	Description
subject	string	Yes	Unique identifier of the user.
msg	string	No	If the status is successful, returns ”Post Risk evaluation successful”. If the status is a failure, return the reason for the failure.
clientRespContext	array of key-value pairs	No	All the client response context that must be returned to the client as part of the successful authentication response should be returned in this field.

Sample Response of "/PostUserRiskScoreEvaluator" API:


```
{
"subject": "nbruce",
"msg": "Post risk evaluation success",
"clientRespContext": [
{
"key": "deviceId",
"value": "device-hdyn235sd"
}
]
}
```


API Schema - (Swagger Content)

To view the sample representation of the API schema, see the **Common Open API Spec for all Custom Providers - (Swagger Content)** section in Integrating with Custom SPI.

Supported Use Cases

The following diagrams show different scenarios of integration between Custom Risk Provider and IDSP.

Use Case: 1

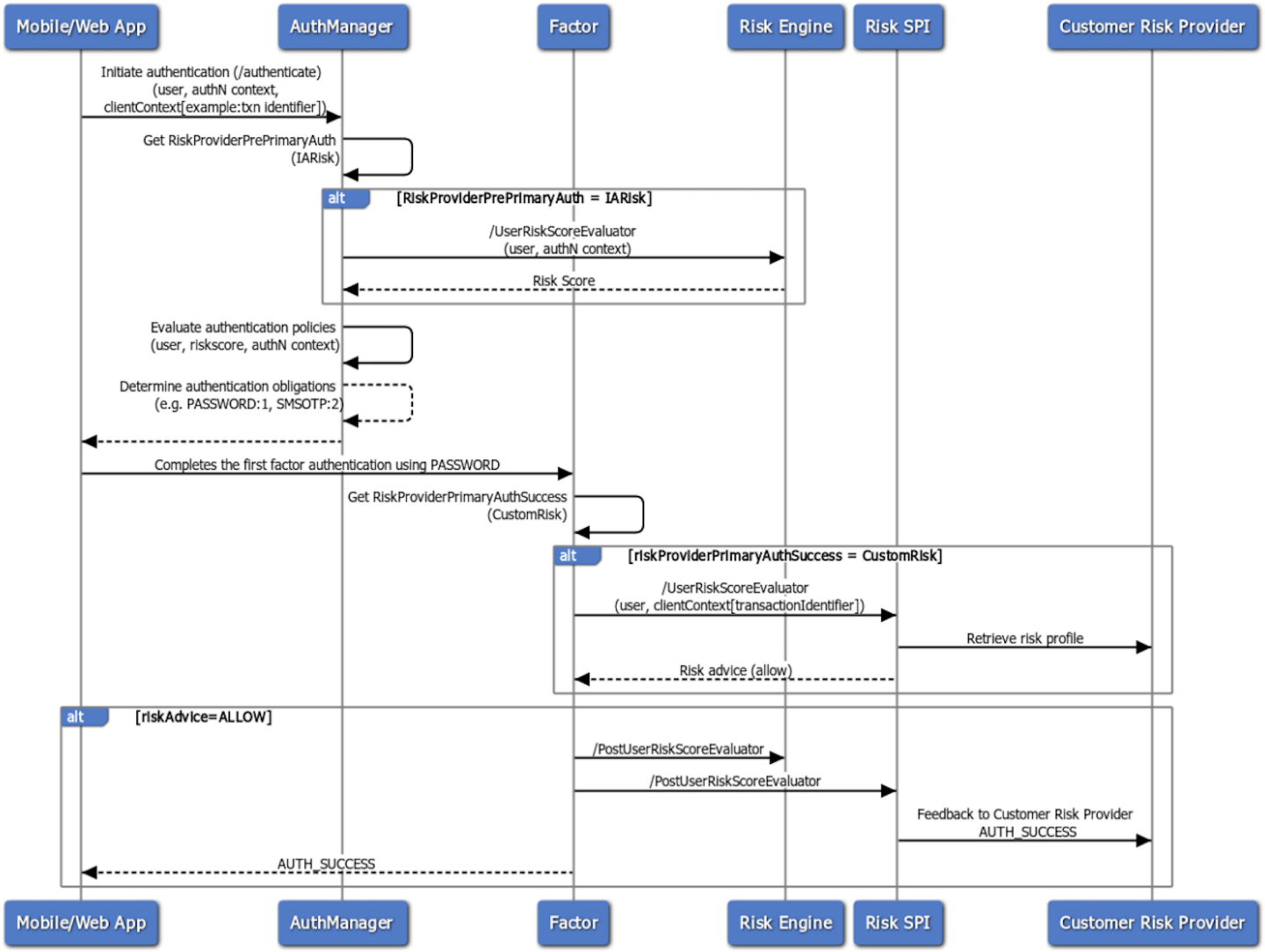
This use case assumes that the customer has configured the global setting parameters for risk-based authentication as follows:

- riskProviderPrePrimaryAuth = IARISK
- riskProviderPrimaryAuthSuccess = CustomRisk

> **NOTE**
>
> "IARISK", is the name of an internal risk provider.
>
> "CustomRisk" is the name of the customer risk provider.
>
> To know more about the global setting parameters, see Settings.

List of participants and what they represent:

- **Application (Mobile/Web App):** Represents the application that calls the solution APIs to acquire the identity token and complete the authentication flow.
- **IDSP Microservices:**
  - **AuthManager:** Initiates the authentication flow when a client invokes the **/auth/v1/authenticate** API.
  - **Factor:** Exposes the APIs to perform user authentication using different factors/obligations (PASSWORD, SMSOTP, EMAILOTP, Biometrics, PUSH, and so on) identified by the policy engine.
  - **Risk Engine:** Represents an internal risk engine that exposes the APIs to evaluate the risk score for a user transaction.
  - **Risk SPI:** Represents a wrapper (implemented by the customer) that the solution uses to call the customer risk provider.
  - **Customer Risk Provider:** Represents the customer risk provider API that is used to evaluate the risk advice for a user transaction.



Flow Description:

1. The client app invokes the **/auth/v1/authenticate** API to initiate the authentication flow.

   NOTE
   In certain cases, a client app may have to send additional parameters that may be mandatorily required by the custom risk to function. These parameters (example, transaction Id, and so on) can be passed through the **clientContext** in the **/auth/v1/authenticate** API call. See the following reference sample:

```
{
    "subject": "<userLoginid>",
    "channel": "web",
    "ipAddress": "10.10.123.124",
    "action":"authenticate",
    "rememberMe" : true,
    "device": {
        "signature": {
            "iaAuthData": "_v02MyUgNGgYOi88OTk0........."
        }
    },
```

```
            "clientContext": [
                {
                    "key": "transactionIdentifier",
                    "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40"
                }
            ]
        }
```

2. Internally, the **/auth/v1/authenticate** API performs the following actions:
   a. Retrieves the configured risk provider from the global setting - **riskProviderPrePrimaryAuth.** The supported values are **IARISK, AARisk, CustomRisk** or **None.** In this flow, **IARISK** is assumed to be configured.
   b. The AuthManager invokes the **/iarisk/v1/UserRiskScoreEvaluator** API to evaluate the risk score.
   c. Upon getting the risk score, AuthManager internally communicates with the policy engine to evaluate the configured policy rules for the client app and determines the authentication obligations. In the flow, it is assumed that the obligations returned are PASSWORD:1 and SMSOTP:2. The AuthManager communicates the authentication challenges to the client.
3. The client invokes the **/factor/v1/PasswordAuthenticator** API to successfully complete authentication for the first factor, PASSWORD:1.
4. After the authentication is successful, and before authenticating the second factor, the PasswordAuthenticator API checks the configured value of the global setting - **riskProviderPrimaryAuthSuccess.** This property defines the risk provider that has been configured to perform risk evaluation post the primary authentication is successful. The supported values are **CustomRisk** or **None.** In this flow, **CustomRisk** is assumed to be configured.

   NOTE
   If the **riskProviderPrimaryAuthSuccess (secondary risk)** parameter is set to custom, the solution analyzes the risk and uses the ***risk advice*** to assess the next steps in the flow. The risk score returned by the custom risk engine is ignored.
5. The **PasswordAuthenticator** API invokes the **<CustomProviderHost>/UserRiskScoreEvaluator** API to evaluate the risk advice. Based on the evaluation, the Risk SPI returns either of the following advice:
   – **ALLOW:** The solution skips the remaining authentication factors, completes the authentication flow, and marks it as AUTH_ALLOWED.
   – **DENY:** The solution skips the remaining authentication factors, completes the authentication flow, and marks it as AUTH_DENIED.
   – **STEPUP:** The solution continues to challenge the user with the remaining authentication factors.
6. Assume that the **<CustomProviderHost>/UserRiskScoreEvaluator** API returns the risk advice as ALLOW in the response payload.

```
{
   "subject": "nbruce",
   "riskScore": 5,
   "riskReason": "Registered device detected",
   "riskFactors": [
   "Device Recognition"
       ],
   "riskAdvice": "allow",
   "msg": "Risk evaluation success",
   "riskEvalContext": [
     {
       "key": "transactionId",
       "value": "txnId-6hdds733"
     },
     {
       "key": "deviceId",
       "value": "device-hdyn235sd"
```

```
        }
      ]
  }
```

7. The PasswordAuthenticator API invokes the **/PostUserRiskScoreEvaluator** API individually to the Risk Engine and Risk SPI to inform about the authentication status. The Risk SPI in return sends the feedback about the authentication status to the Customer Risk Provider.
8. The PasswordAuthenticator API returns AUTH_SUCCESS to the client, thereby completing the authentication flow.

Use Case: 2

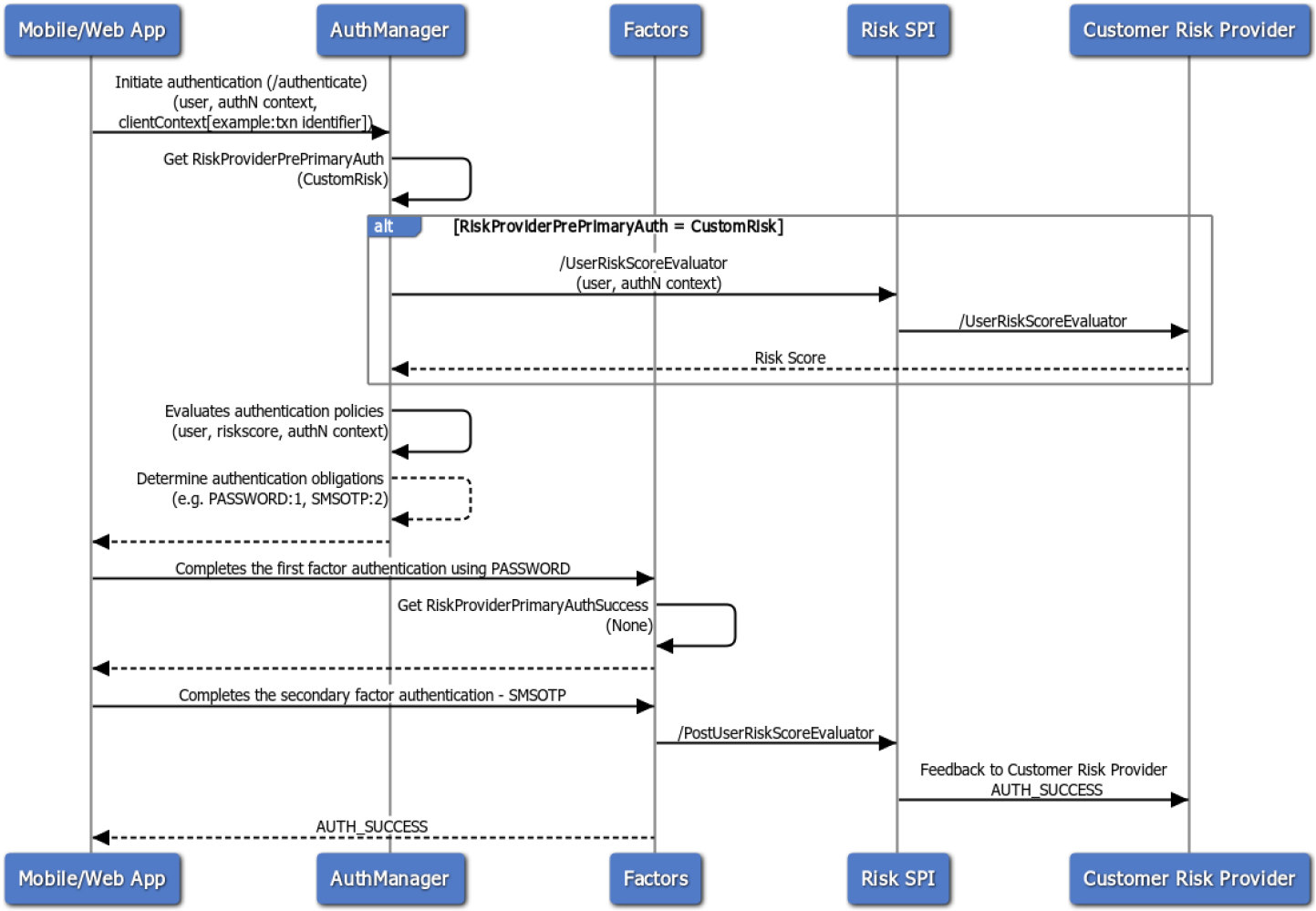
This use case assumes that the customer has configured the global setting parameters for risk-based authentication as follows:

- riskProviderPrePrimaryAuth = CustomRisk
- riskProviderPrimaryAuthSuccess = None

> **NOTE**
> "CustomRisk" is the name of the customer risk provider. To know more about the global setting parameters, see Settings.

List of participants and what they represent:

- **Application (Mobile/Web App):** Represents the application that calls the solution APIs to acquire the identity token and complete the authentication flow.
- **IDSP Microservices:**
  - **AuthManager:** Initiates the authentication flow when a client invokes the **/auth/v1/authenticate** API.
  - **Factor:** Exposes the APIs to perform user authentication using different factors/obligations (PASSWORD, SMSOTP, EMAILOTP, Biometrics, PUSH, and so on) identified by the policy engine.
  - **Risk SPI:** Represents a wrapper (implemented by the customer) that the solution uses to call the customer risk provider.
  - **Customer Risk Provider:** Represents the customer risk provider API that is used to evaluate the risk advice for a user transaction.



Flow Description:

1. The client app invokes the **/auth/v1/authenticate** API to initiate the authentication flow.

   NOTE
   In certain cases, a client app may have to send additional parameters that may be mandatorily required by the custom risk to function. These parameters (example, transaction Id, and so on) can be passed through the **clientContext** in the **/auth/v1/authenticate** API call. See the following reference sample:

```
{
    "subject": "<userLoginid>",
    "channel": "web",
    "ipAddress": "10.10.123.124",
    "action":"authenticate",
    "rememberMe" : true,
    "device": {
        "signature": {
            "iaAuthData": "_v02MyUgNGgYOi88OTk0........."
        }
    },
     "clientContext": [
        {
            "key": "transactionIdentifier",
```

```
                    "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40"
                }
            ]
        }
```

2. Internally, the **/auth/v1/authenticate** API performs the following actions:
   a. Retrieves the configured risk provider from the global setting - **riskProviderPrePrimaryAuth.** The supported values are **IARISK, AARisk, CustomRisk** or **None.** In this flow, **CustomRisk** is assumed to be configured.
   b. The AuthManager invokes the **<CustomProviderHost>/UserRiskScoreEvaluation** API to evaluate the risk score. The API call passes through the Risk SPI to the Custom Risk Provider. Risk evaluation results in the risk score and risk advice.

      NOTE
      In a scenario where **riskProviderPrePrimaryAuth** is set to **CustomRisk,** only the risk score is considered for the policy evaluation and the risk advice is ignored.

   c. Upon getting the risk score, AuthManager internally communicates with the policy engine to evaluate the configured policy rules for the client app and determines the authentication obligations. In the flow, it is assumed that the obligations returned are PASSWORD:1 and SMSOTP:2. The AuthManager communicates the authentication challenges to the client.
3. The client invokes the **/factor/v1/PasswordAuthenticator** API to successfully complete authentication for the first factor, PASSWORD:1.
4. After the authentication is successful, and before authenticating the second factor, the PasswordAuthenticator API checks the configured value of the global setting - **riskProviderPrimaryAuthSuccess.** This property defines the risk provider that has been configured to perform risk evaluation post the primary authentication is successful. The supported values are **CustomRisk** or **None.** In this flow, **None** is assumed to be configured.
5. The client is challenged with the remaining obligation - SMSOTP:2. The client successfully completes the Multi-factor authentication (MFA).
6. On MFA completion, the PasswordAuthenticator API invokes the **/PostUserRiskScoreEvaluator** API to the Risk SPI to inform about the authentication status. The Risk SPI in return sends the feedback about the authentication status to the Customer Risk Provider.
7. The PasswordAuthenticator API returns AUTH_SUCCESS to the client, thereby completing the authentication flow.

Use Case: 3

This use case assumes that the customer has configured the global setting parameters for risk-based authentication as follows:

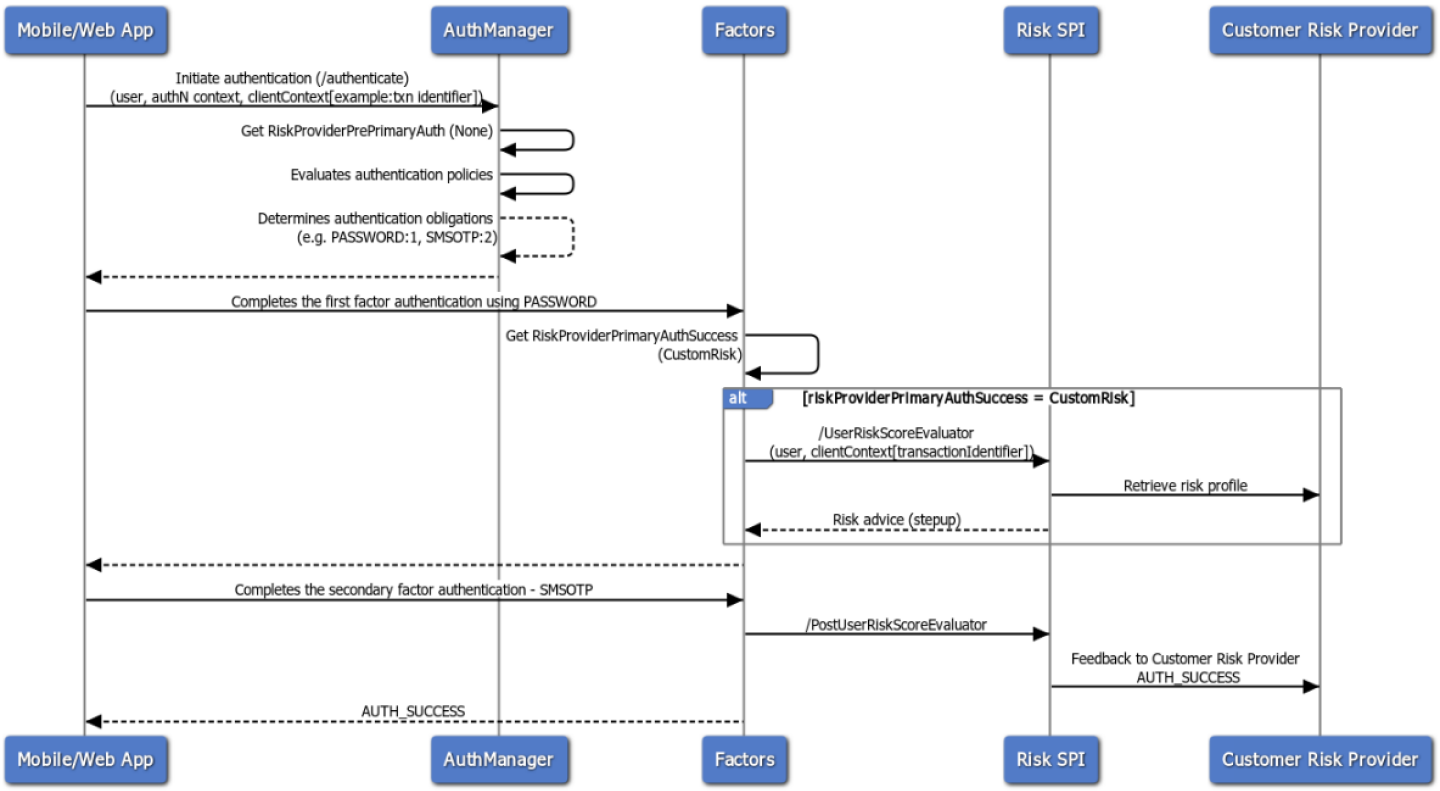
- riskProviderPrePrimaryAuth = None
- riskProviderPrimaryAuthSuccess = CustomRisk

NOTE
"CustomRisk" is the name of the customer risk provider. To know more about the global setting parameters, see Settings.

List of participants and what they represent:

- **Application (Mobile/Web App):** Represents the application that calls the solution APIs to acquire the identity token and complete the authentication flow.
- **IDSP Microservices:**

- **AuthManager:** Initiates the authentication flow when a client invokes the **/auth/v1/authenticate** API.
- **Factor:** Exposes the APIs to perform user authentication using different factors/obligations (PASSWORD, SMSOTP, EMAILOTP, Biometrics, PUSH, and so on) identified by the policy engine.
- **Risk SPI:** Represents a wrapper (implemented by the customer) that the solution uses to call the customer risk provider.
- **Customer Risk Provider:** Represents the customer risk provider API that is used to evaluate the risk advice for a user transaction.



Flow Description:

1. The client app invokes the **/auth/v1/authenticate** API to initiate the authentication flow.

   NOTE
   In certain cases, a client app may have to send additional parameters that may be mandatorily required by the custom risk to function. These parameters (example, transaction Id, and so on) can be passed through the **clientContext** in the **/auth/v1/authenticate** API call. See the following reference sample:

```
{
    "subject": "<userLoginid>",
    "channel": "web",
    "ipAddress": "10.10.123.124",
    "action":"authenticate",
    "rememberMe" : true,
    "device": {
        "signature": {
            "iaAuthData": "_v02MyUgNGgYOi88OTk0........."
        }
    },
     "clientContext": [
```

```
                {
                    "key": "transactionIdentifier",
                    "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40"
                }
            ]
        }
```

2. Internally, the **/auth/v1/authenticate** API performs the following actions:
   a. Retrieves the configured risk provider from the global setting - **riskProviderPrePrimaryAuth.** The supported values are **IARISK, AARisk, CustomRisk** or **None.** In this flow, **None** is assumed to be configured.
   b. AuthManager invokes the Policy Engine to evaluate the configured policy rules, without any riskscore for the client app, and determines the authentication obligations. In this flow, it is assumed that the obligations returned are PASSWORD:1, and SMSOTP:2.
3. The client invokes the **/factor/v1/PasswordAuthenticator** API to successfully complete authentication for the first factor, PASSWORD:1.
4. After the authentication is successful, and before authenticating the second factor, the PasswordAuthenticator API checks the configured value of the global setting - **riskProviderPrimaryAuthSuccess.** This property defines the risk provider that has been configured to perform risk evaluation post the primary authentication is successful. The supported values are **CustomRisk** or **None.** In this flow, **CustomRisk** is assumed to be configured.

   NOTE
   If the **riskProviderPrimaryAuthSuccess (secondary risk)** parameter is set to custom, the solution analyzes the risk and uses the ***risk advice*** to assess the next steps in the flow. The risk score returned by the custom risk engine is ignored.
5. The **PasswordAuthenticator** API invokes the **<CustomProviderHost>/UserRiskScoreEvaluator** API to evaluate the risk advice. Based on the evaluation, the Risk SPI returns either of the following advice:
   - **ALLOW:** The solution skips the remaining authentication factors, completes the authentication flow, and marks it as AUTH_ALLOWED.
   - **DENY:** The solution skips the remaining authentication factors, completes the authentication flow, and marks it as AUTH_DENIED.
   - **STEPUP:** The solution continues to challenge the user with the remaining authentication factors.
6. Assume that the **<CustomProviderHost>/UserRiskScoreEvaluator** API returns the risk advice as STEPUP in the response payload.

```
{
  "subject": "nbruce",
  "riskScore": 65,
  "riskReason": "Unknown device",
  "riskFactors": [
  "Device Recognition"
    ],
  "riskAdvice": "stepup",
  "msg": "Risk evaluation success",
  "riskEvalContext": [
    {
      "key": "transactionId",
      "value": "txnId-6hdds733"
    },
    {
      "key": "deviceId",
      "value": "device-hdyn235sd"
    }
  ]
}
```

7. The **PasswordAuthenticator** API invokes the **/factor/v1/OTPGenerator** API to initiate MFA using SMSOTP.
8. The client is challenged with the obligation - SMSOTP. The client successfully completes the Multi-factor authentication (MFA).
9. On completing the authentication, the Factor API (/factor/v1/OTPVerifier) invokes the **<CustomProviderHost>/PostUserRiskScoreEvaluator** API to the Risk SPI. The Risk SPI internally invokes the Customer Risk Provider to inform about the AUTH_SUCCESS status. The Custom Risk Provider sends the response details to Risk SPI > Factor API, for example, device Id.
10. The Factor API returns AUTH_SUCCESS to the client app, thereby completing the authentication flow.

Reference Link to:

- Custom Risk Provider APIs

Integrating with External SMS Provider

Integrate Symantec Identity Security Platform with external SMS providers for multi-factor authentication using the Custom Service Provider Interface (SPI).

The Custom SPI utilizes an OpenAPI specification, requiring you to host a compliant web service within your own environment. To enable communication between Symantec Identity Security Platform and your SMS provider, you must implement the specific API endpoints defined in the following sections.

This topic includes the following:

- Custom Provider Overview
- Steps to Integrate with Custom SMS SPI
- Implementing the API
- API Schema - (Swagger Content)
- Creating a SMS Provider
- Configuring Settings in IDSP
- SMS Use Case

Custom Provider Overview

The Symantec Identity Security Platform Custom Service Provider Interface (SPI) enables integration with third-party SMS providers for multi-factor authentication. This SPI is based on an OpenAPI specification and requires a custom web service hosted within your environment. To establish communication, you must implement the API endpoints and configure the system settings as defined in this documentation.

Steps to Integrate with Custom SMS SPI

The three steps that the Customers have to perform to integrate with the SMS SPI are:

1. Develop the custom SMS service based on the SMS-Open API Spec provided by IDSP.
2. Create an SMS Custom Provider in IDSP using the Provider endpoint- https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders
3. Specify the configuration settings in IDSP.

These steps have been described in the subsequent sections.

Implementing the API

The following API should be implemented by the customer to integrate their SMS provider with the solution's custom SPI:

- "**SMSMessageSender**" - This API request is used to send the SMS to the customers.

The API request and response templates are based on the standard Open API spec that IDSP expects the customers to provide during the integration process. These specifications have been explained in the subsequent sections.

SMSMessageSender API Specification

SMSMessageSender API Specification	
API Name	SMSMessageSender
API URL	*https://<CustomProviderHost>/**SMSMessageSender***
API OPERATION	*POST*
AUTHORIZATION	*<<API Key shared by the custom SMS provider>>*
API RESPONSE CODE	*200 Success, 400 Failures, 500 Server Errors*

Request Payload includes the following parameters:

Parameter	Type	Mandatory (Yes/No)	Description
X-API-KEY	Header	No	API key used to invoke the custom SMS provider
subject	string	Yes	Unique identifier of the user
phoneNumber	string	Yes	Phone number to which SMS must be delivered
otp	string	Yes	OTP value that must be delivered
otpValidityInSecs	Integer	Yes	Validity of the OTP in seconds
message	string	Yes	The status message that must be delivered through the SMS

clientContext	an array of key-value pairs	Yes	Client context parameters received as part of the "/auth/v1/authenticate" call will be passed here by the client. Example: transactionIdentifier, appid, etc.
			`{`
			`"subject": "{{userLoginid}}",`
			`"channel": "web",`
			`"ipAddress": "10.10.123.124",`
			`"action":"authenticate",`
			`"rememberMe" : true,`
			`"device": {`
			`"signature": {`
			`"iaAuthData": "_v02MyUgNGgYOi88OTk0....."`
			`}`
			`},`
			`"clientContext": [`
			`{`
			`"key": "transactionIdentifier",`
			`"value": "540715af-96f2-4b86-b89f-ca0e8ae93f40"`
			`}`
			`]`
			`}`

ahContext	an array of key-value pairs	No	Contains the IDSP context parameters. For information, see the *Sharing IDSP Context with SPI* section in Integrating with Custom SPI.

Parameter	Type	Mandatory (Yes/No)	Description
providerContext	array of key-value pairs	No	Parameters configured as part of the custom sms provider

Sample Request of "/SMSMessageSender" API:

```
{
  "subject": "nbruce",
  "phoneNumber": 13242342341,
  "otp": 783242,
  "otpValidityInSecs": 300,
  "message": "783242 is your One-Time Password to login. The OTP is valid upto 10 minutes.",
  "clientContext": [
    {
      "key": "transactionIdentifier",
      "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40"
    }
  ],
  "ahContext": [
    {
      "key": "clientTxnId",
      "value": "cd2f1e6e-c356-49ae-a5b8-c10bf571a9a0"
    },
    {
      "key": "txnId",
      "value": "txn-id-hd26se23"
    }
  ],
  "providerContext": [
    {
      "key": "instanceId",
      "value": "brazil-instance-1"
    }
  ]
}
```

Response Payload should include the following parameters:

Parameter	Type	Mandatory (Yes/No)	Description
status	string	Yes	This parameter returns the message delivery status as either a "success" or a "failure".
msg	string	Yes	This parameter returns a status message reflecting the delivery outcome. It returns specific messages based on success or failure instances. For example: • In case of failure, it may return a message such as "INTERNAL ERROR". • In case of success, it may return a message like "Message delivered successfully via SMS".

Parameter	Type	Mandatory (Yes/No)	Description
clientRespContext	an array of key-value pairs	No	This parameter returns all the client response context that must be returned by the SPI client as part of the successful authentication response.

Sample Response of "/SMSMessageSender" API:

```
{
   "status": "success",
   "msg": "Message delivered successfully via SMS",
   "clientRespContext": [
    {
      "key": "transactionId",
      "value": "txnId-6hdds733"
    },
    {
      "key": "deviceId",
      "value": "device-hdyn235sd"
    }
  ]
}
```

API Schema - (Swagger Content)

To view the sample representation of the API schema, see the **Common Open API Spec for all Custom Providers - (Swagger Content)** section in Integrating with Custom SPI.

Creating an SMS Custom Provider

- Create an SMS custom provider in IDSP using the Provider endpoint-
  `https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders`

  NOTE

  - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
  - The default value for `<tenantName>` is "default".
  - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
    For example:
    `https://{{host}}/{{tenantName}}/admin/v1/CustomProviders`
    should look similar to:
    `https://example.com/default/admin/v1/CustomProviders`
  - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

  NOTE
  To refer to the Custom Provider APIs which can be used to perform CRUD operations in IDSP, see here.

Configuring the Global/Tenant Setting attribute

- Configure the global/tenant setting, ***smsProviderToBeUsed*** in IDSP to set a custom SPI provider as the SMS provider. The default value is "Twilio". See Settings for more information.

SMS Use Case

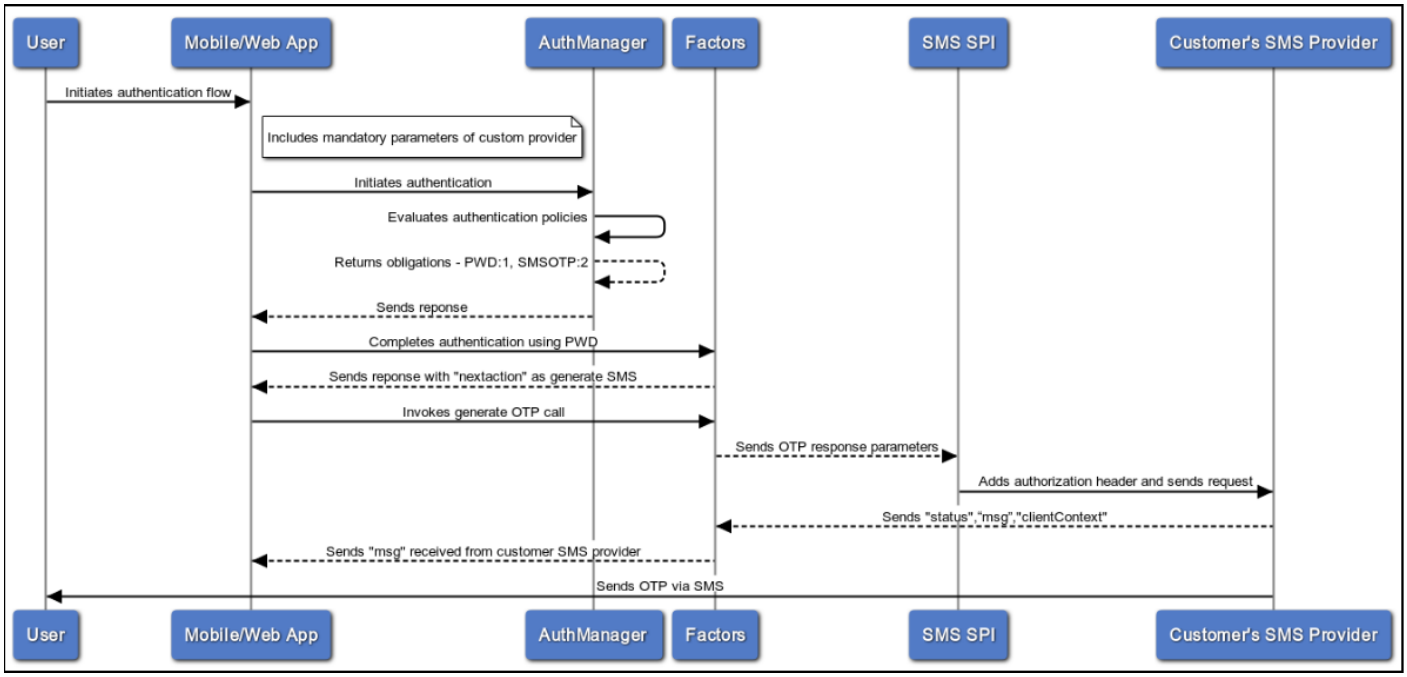
The following use case describes the integration between Custom SMS Provider and IDSP.

It is assumed that the customer has configured the following global/tenant setting parameter:

- "smsProviderToBeUsed"

List of participants and what they represent:

- User- User who initiates the authentication flow.
- Application (Mobile/Web App): This is the application that calls the solution's APIs to acquire the identity token and complete the authentication flow.
- IDSP microservices:
  - AuthManager- This is the ***"/auth/v1/authenticate"*** API used to initiate the authentication flow.
  - Factors- This is the ***"/factor/v1/OTPGenerator"*** API used to generate the otp.
  - SMS SPI- This is the SMS module in the IDSP used to call the custom SMS provider.
  - Customer's SMS Provider- This is the customer's implementation of SMS Provider API which is used to deliver SMS to the end user's phone number.



Flow Description:

1. The User initiates the Authentication Flow.
2. The client app invokes the "auth/v1/authenticate" API to initiate the Authentication flow. The request includes parameters such as user, authN context, clientContext [example:txn identifier].

   NOTE
   In certain cases, a client app may have to send additional parameters that may be mandatorily required by the custom SMS provider to function. These parameters, for example, transaction Id, etc. can be passed through the "***clientContext***" in the "/auth/v1/authenticate" call. See the following reference sample:

```
{
    "subject": "{{userLoginid}}",
    "channel": "web",
    "ipAddress": "10.10.123.124",
```

```
            "action":"authenticate",
            "rememberMe" : true,
            "device": {
                "signature": {
                    "iaAuthData": "_v02MyUgNGgYOi88OTk0........."
                }
            },
             "clientContext": [
                {
                    "key": "transactionIdentifier",
                    "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40"
                }
            ]
        }
```

3. Internally, the "/auth/v1/authenticate" API does the following:
   - Checks whether the global/tenant settings parameter, "smsProviderToBeUsed" is configured in the IDSP. In this flow, we have assumed that the setting is configured to a custom SMS provider.
   - AuthManager communicates with the Policy Engine to evaluate the configured policy rules for the client app and determines the authentication obligations. Also, sends a response to the client app. In the flow, it is assumed that the obligations returned are PWD:1 and SMSOTP:2.
4. The client successfully completes the first-factor authentication, PWD:1.
5. The client is challenged with the remaining obligation, that is SMSOTP:2.
6. The Factor API sends response to the client app with "nextaction" as generate SMSOTP.
7. The client app invokes the "/factor/v1/OTPGenerator" API to generate the SMSOTP.
8. The Factor API sends SMSOTP details, including otp, otpValidity, message, clientContext, providerContext and so on, to the SMS SPI.
9. The SMS SPI adds an Authorization header (API Key) and sends the SMSOTP details to the Customer's SPI provider.
10. The Customer's SMS Provider in return sends the SMS delivery response to the Factors service. The response includes transaction status and message. It also sends the OTP through SMS to the end customer’s mobile number.
11. The Factor API sends "msg" received from the customer's SMS provider to the client app.

Reference Links to:

- SMS OTP Authentication

Integrating with External Email Provider

Integrate Symantec Identity Security Platform with a custom email provider for email-based authentication using the Custom Service Provider Interface (SPI).

The Custom SPI is based on an OpenAPI specification and requires a separate web service hosted within your infrastructure. To enable communication between Symantec Identity Security Platform and your email provider, you must implement the required API endpoints and configure the settings defined in this section.

This topic includes the following:

- Custom Provider Overview
- Steps to Integrate with Custom Email SPI
- Implementing the APIs
- API Schema - (Swagger Content)
- Creating an Email Provider
- Configuring Settings in IDSP
- Email Use Case

Custom Provider Overview

The Symantec Identity Security Platform Custom Service Provider Interface (SPI) allows the solution to be integrated with a custom Email provider for Email-based authentication. The SPI uses an Open API spec and expects the customers to implement it as a separate web service that runs in their environment. The customer should implement the API, and should configure the required settings that are defined by the solution, to enable the solution's custom SPI to communicate with the customer's email provider. The API and Configuration settings have been explained in the subsequent sections.

Steps to Integrate with Custom Email SPI

The three steps that the Customers have to perform to integrate with the Email SPI are:

1. Develop the custom Email service based on the Email-Open API Spec provided by IDSP.
2. Create an Email Custom Provider in IDSP using the Provider endpoint- https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders
3. Specify the configuration settings in IDSP.

These steps have been described in the subsequent sections.

Implementing the API

The following API should be implemented by the customer to integrate their Email provider with the solution's custom SPI:

- "**EmailMessageSender**" - This API request is used to send the Email to the customers.

The API request and response templates are based on the standard Open API spec that IDSP expects the customers to provide during the integration process. These specifications have been explained in the subsequent sections.

EmailMessageSender API Specification

EmailMessageSender API Specification	
API Name	EmailMessageSender
API URL	*https://<CustomProviderHost>/**EmailMessageSender***
API OPERATION	*POST*
AUTHORIZATION	*<<API Key shared by the custom Email provider>>*
API RESPONSE CODE	*200 Success, 400 Failures, 500 Server Errors*

Request Payload includes the following parameters:

Parameter	Type	Mandatory (Yes/No)	Description
X-API-KEY	Header	No	API key used to invoke the custom Email provider
subject	string	Yes	Unique identifier of the user
email	string	Yes	Email Address to which email must be delivered
otp	string	Yes	OTP value that must be delivered
emailSubject	string	Yes	Subject of the email to be delivered
otpValidityInSecs	Integer	Yes	Validity of the OTP in seconds
message	string	Yes	The status message that must be delivered through the email

Parameter	Type	Mandatory (Yes/No)	Description
clientContext	an array of key-value pairs	No	Client context parameters received as part of the "/auth/v1/authenticate" call will be passed here by the client. Example: transactionIdentifier, appid, etc. `{ "subject": "{{userLoginid}}", "channel": "web", "ipAddress": "10.10.123.124", "action":"authenticate", "rememberMe" : true, "device": { "signature": { "iaAuthData": "_v02MyUgNGgYOi88OTk0........." } }, "clientContext": [{ "key": "transactionIdentifier", "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40" }] }`
ahContext	an array of key-value pairs	No	Contains the IDSP context parameters. For information, see the *Sharing IDSP Context with SPI* section in Integrating with Custom SPI.
providerContext	array of key-value pairs	No	Parameters configured as part of the custom Email provider

Sample Request of "/EmailMessageSender" API:

```
{
  "subject": "nbruce",
  "email": "nina.bruce@example.com",
  "emailSubject": "One-Time Password",
  "otp" : "657841"
  "otpValidityInSecs": 300,
  "message": "657841 is your One-Time Password to login. The OTP is valid upto 10 minutes.",
  "clientContext": [
    {
      "key": "transactionIdentifier",
      "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40"
    }
  ],
  "ahContext": [
    {
      "key": "clientTxnId",
      "value": "cd2f1e6e-c356-49ae-a5b8-c10bf571a9a0"
    },
    {
      "key": "txnId",
```

```
      "value": "txn-id-hd26se23"
    }
  ],
  "providerContext": [
    {
      "key": "instanceId",
      "value": "brazil-instance-1"
    }
  ]
}
```

Response Payload should include the following parameters:

Parameter	Type	Mandatory (Yes/No)	Description
status	string	Yes	This parameter returns the message delivery status as either a "success" or a "failure".
msg	string	Yes	This parameter returns a status message reflecting the delivery outcome. It returns specific messages based on success or failure instances.
clientRespContext	an array of key-value pairs	No	All the client response context that must be returned to the client as part of the successful authentication response should be returned in this field.

Sample Response of "/EmailMessageSender" API:

```
{
   "status": "success",
   "msg": "Message delivered successfully via Email",
   "clientRespContext": [
    {
      "key": "transactionId",
      "value": "txnId-6hdds733"
    },
    {
      "key": "deviceId",
      "value": "device-hdyn235sd"
    }
  ]
}
```

API Schema - (Swagger Content)

To view the sample representation of the API schema, see the **Common Open API Spec for all Custom Providers - (Swagger Content)** section in Integrating with Custom SPI.

Creating an Email Custom Provider

- Create an Email custom provider in IDSP using the Provider endpoint-
  `https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/CustomProviders`
  should look similar to:
  `https://example.com/default/admin/v1/CustomProviders`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

NOTE
To refer to the Custom Provider APIs which can be used to perform CRUD operations in IDSP, see here.

Configuring the Global/Tenant Setting attribute

- Configure the global/tenant setting, ***emailProviderToBeUsed*** in IDSP to set a custom SPI Provider as the Email provider. The default value is "SMTP". See Settings for more information.

Email Use Case

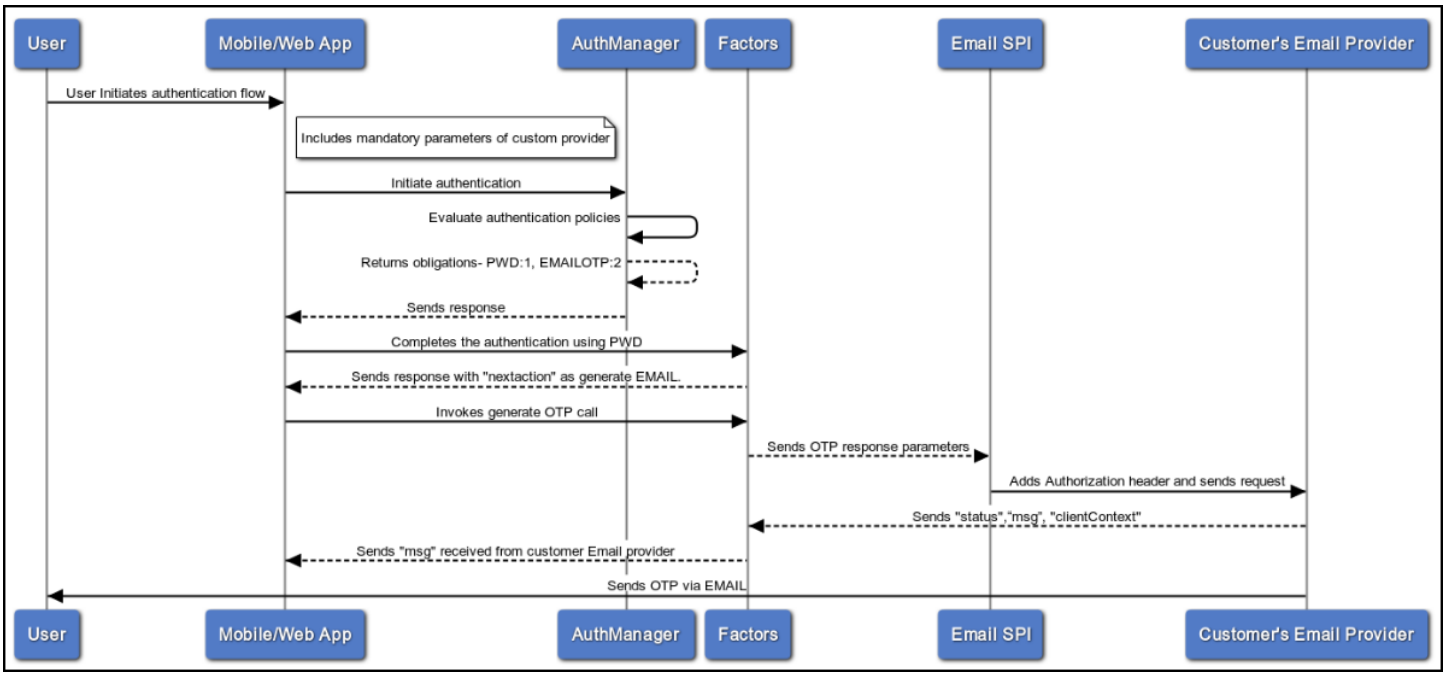
The following use case describes the integration between Custom Email Provider and IDSP.

It is assumed that the customer has configured the following global/tenant setting parameter:

- "emailProviderToBeUsed"

List of participants and what they represent:

- User- User who initiates the authentication flow.
- Application (Mobile/Web App): This is the application that calls the solution's APIs to acquire the identity token and complete the authentication flow.
- IDSP microservices:
  - AuthManager- This is the ***"/auth/v1/authenticate"*** API used to initiate the authentication flow.
  - Factor- This is the ***"/factor/v1/OTPGenerator"*** API used to generate the EMAIL.
  - Email SPI- This is the email wrapper used by the solution to call the custom email provider.
  - Customer's Email Provider- This is the customer's implementation of Email Provider API which is used to deliver email to the end user's email Id.



Flow Description:

1. The User initiates the Authentication Flow.
2. The client app invokes the "auth/v1/authenticate" API to initiate the Authentication flow. The request includes parameters such as user, authN context, clientContext [example:txn identifier].

   NOTE

   In certain cases, a client app may have to send additional parameters that may be mandatorily required by the custom Email provider to function. These parameters, for example, transaction Id, etc can be passed through the "***clientContext***" in the "/auth/v1/authenticate" call. See the following reference sample:

```
{
    "subject": "{{userLoginid}}",
    "channel": "web",
    "ipAddress": "10.10.123.124",
    "action":"authenticate",
    "rememberMe" : true,
    "device": {
        "signature": {
            "iaAuthData": "_v02MyUgNGgYOi88OTk0........."
        }
    },
     "clientContext": [
        {
            "key": "transactionIdentifier",
            "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40"
        }
    ]
}
```

3. Internally, the "/auth/v1/authenticate" API does the following:

  - Checks whether the global/tenant settings parameter, "emailProviderToBeUsed" is configured in the IDSP. In this flow, we have assumed that the setting is configured to a custom email provider.
  - AuthManager communicates with the Policy Engine to evaluate the configured policy rules for the client app and determines the authentication obligations. Also, sends a response to the client app. In the flow, it is assumed that the obligations returned are PWD:1, and EMAILOTP:2.
4. The client successfully completes the first-factor authentication, PWD:1.
5. The client is challenged with the remaining obligation, that is EMAILOTP:2.
6. The Factors API sends a response to the app with "nextaction" as generate EMAILOTP.
7. The Client app invokes the "/factor/v1/OTPGenerator" API to generate the EMAILOTP.
8. The Factor API sends EMAILOTP details, including email, emailSubject, otp, otpValidity, clientContext, providerContext, and so on, to the EMAIL SPI.
9. The Email SPI adds an Authorization header (API Key) and sends the EMAILOTP details to the Customer's SPI provider.
10. The Customer's Email Provider in return sends the Email delivery response to the Factors service. The response includes transaction status and message. It also sends the OTP through email to the end customer’s email Id.
11. The Factor API sends "msg" received from customer's Email provider to the client app.

Reference Links to:

- Email OTP Authentication

Integrating with External Push Provider

Integrate Symantec Identity Security Platform with a custom push provider to send authentication notifications to mobile devices using the Custom Service Provider Interface (SPI).

This topic includes the following:

- Custom Provider Overview
- Implementing the SPIs
- Steps to Integrate with Custom Push SPI
- API Schema - (Swagger Content)
- Creating a Push Provider
- Configuring Settings in IDSP
- Push Use Case

Custom Provider Overview

The Custom SPI utilizes an OpenAPI specification and requires a separate web service hosted within your infrastructure. To enable communication between Symantec Identity Security Platform and your push provider, you must implement the SPI endpoints and configure the system settings detailed in the following sections.

Steps to Integrate with Custom Push SPI

The three steps that the Customers have to perform to integrate with the Push SPI are:

1. Develop the Custom Push Service based on the Push-Open API Spec provided by IDSP.
2. Create a Push Custom Provider in IDSP using the Provider endpoint-

```
https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/CustomProviders`
  should look similar to:
  `https://example.com/default/admin/v1/CustomProviders`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

3. Specify the configuration settings in IDSP.

These steps have been described in the subsequent sections.

Implementing the SPIs

The following SPI should be implemented by the customer to integrate their Push provider with the solution's custom SPI:

- "**PushMessageSender**" - This SPI request is used to send push notifications to the customers.
- "**PushStatus**" - This SPI request is used to check the status of push authentication at the Push SPI provider service.

The SPI request and response templates are based on the standard Open API spec that IDSP expects the customers to provide during the integration process. The SPI specifications have been explained in the subsequent sections.

PushMessageSender SPI Specification

PushMessageSender SPI Specification	
API Name	PushMessageSender
API URL	*https://<CustomProviderHost>/**PushMessageSender***
API OPERATION	*POST*
AUTHORIZATION	*<<SPI Key shared by the custom Push provider>>*
API RESPONSE CODE	*200 Success, 400 Failures, 500 Server Errors*

Request Payload includes the following parameters:

Parameter	Type	Mandatory (Yes/No)	Description
X-API-KEY	Header	No	API key used to invoke the custom Push provider
subject	string	Yes	Unique identifier of the user
deviceIdentifier	string	Yes	Device Identifier of the device to which a Push notification must be sent. For example, FCM registration token for a device.
title	string	Yes	Title of Push notification
msg	string	Yes	The message that must be delivered through the Push notification

Parameter	Type	Mandatory (Yes/No)	Description
clientContext	an array of key-value pairs	Yes	Client context parameters received as part of the "/auth/v1/authenticate" call will be passed here by the client. Example: transactionIdentifier, appid, etc. `{ "subject": "{{userLoginid}}", "channel": "web", "ipAddress": "10.10.123.124", "action":"authenticate", "rememberMe" : true, "device": { "signature": { "iaAuthData": "_v02MyUgNGgYOi88OTk0........." } }, "clientContext": [{ "key": "transactionIdentifier", "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40" }] }`
ahContext	an array of key-value pairs	No	Contains the IDSP context parameters. For information, see the *Sharing IDSP Context with SPI* section in Integrating with Custom SPI.
providerContext	array of key-value pairs	No	Parameters configured as part of the custom Push provider

Sample Request of "/PushMessageSender" SPI:

```
{
  "subject": "nbruce",
  "deviceIdentifier": "DEVID-01-002",
  "title": "Push Authentication",
  "msg": "Please approve your transaction",
  "clientContext": [
    {
      "key": "transactionIdentifier",
      "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40"
    }
  ],
  "ahContext": [
    {
      "key": "clientTxnId",
      "value": "cd2f1e6e-c356-49ae-a5b8-c10bf571a9a0"
    },
    {
      "key": "txnId",
      "value": "txn-id-hd26se23"
    }
```

```
  ],
  "providerContext": [
    {
      "key": "instanceId",
      "value": "brazil-instance-1"
    }
  ]
}
```

Response Payload should include the following parameters:

Parameter	Type	Mandatory (Yes/No)	Description
status	string	Yes	This parameter returns the message delivery status as either of the followig: "INITIATED", "AUTHENTICATED", "REJECTED", "IN_PROGRESS", "TIMED_OUT"
msg	string	Yes	This parameter returns a status message reflecting the delivery outcome. For example: In case of success, it may return a message like "Message delivered successfully via Push".
pushContext	an array of key-value pairs	No	The keys in the 'pushContext' element are returned by your PushMessageSender SPI implementation and consumed by your PushStatus SPI implementation. It is upto your implementation to construct and name these keys as you see fit. Note that whatever lookup you have implemented in your PushStatus custom SPI to track previously sent PUSH request, that lookup uses the data from and only from, 'pushContext' that is being passed into your custom PushStatus SPI. Your custom PushStatus SPI must not rely on any other identifiers. All the push context key-value pairs are returned by your SPI client as part of the Push authentication response. For example, pushRequestId, customDeviceId

Sample Response of "/PushMessageSender" SPI:

```
{
   "status": "success",
   "msg": "Message delivered successfully via Push",
   "pushContext": [
    {
      "key": "pushRequestId",
      "value": "12345"
    },
    {
      "key": "customDeviceId",
      "value": "device-hdyn235sd"
    }
  ]
}
```

PushStatus SPI Specification

This SPI request is used to check the status of Push authentication at the Push SPI provider service.

PushStatus SPI Specification	
API Name	PushStatus
API URL	*https://<CustomProviderHost>/**PushStatus***
API OPERATION	*POST*
AUTHORIZATION	*<<SPI Key shared by the custom Push provider>>*
API RESPONSE CODE	*200 Success, 400 Failures, 500 Server Errors*

Request Payload includes the following parameters:

Parameter	Type	Mandatory (Yes/No)	Description
X-API-KEY	Header	No	API key used to invoke the custom Push provider
subject	string	Yes	Unique identifier of the user
pushContext	an array of key-value pairs	Yes	PushContext is returned by the /PushMessageSender SPI that contains push-specific context attributes. Example: pushRequestId, customDeviceId
ahContext	an array of key-value pairs	No	Contains the IDSP context parameters. For information, see the *Sharing IDSP Context with SPI* section in Integrating with Custom SPI.
providerContext	array of key-value pairs	No	Parameters configured as part of the custom push provider SPI.

Sample Request of "/PushStatus" SPI:

```
{
  "subject": "nbruce",
  "pushContext": [
    {
      "key": "pushRequestId",
      "value": "12345"
    },
    {
      "key": "customDeviceId",
      "value": "device-hdyn235sd"
    }
  ],
  "ahContext": [
    {
      "key": "clientTxnId",
      "value": "cd2f1e6e-c356-49ae-a5b8-c10bf571a9a0"
    },
    {
      "key": "txnId",
      "value": "txn-id-hd26se23"
    }
  ],
  "providerContext": [
```

```
    {
      "key": "instanceId",
      "value": "brazil-instance-1"
    }
  ]
}
```

Response Payload should include the following parameters:

Parameter	Type	Mandatory (Yes/No)	Description
status	string	Yes	This parameter returns the message delivery status as either of the followig: "INITIATED", "AUTHENTICATED", "REJECTED", "IN_PROGRESS", "TIMED_OUT"
msg	string	Yes	This parameter returns a status message reflecting the delivery outcome. For example: 'Approved'
clientRespContext	an array of key-value pairs	No	All the push context key-value pairs must be returned by the SPI client as part of the Push authentication response. For example, transactionId.

Sample Response of "/PushStatus" SPI:

```
{
  "status": "success",
  "msg": "Message delivered successfully via Push",
  "clientRespContext": [
    {
      "key": "pushRequestId",
      "value": "12345"
    },
    {
      "key": "customDeviceId",
      "value": "device-hdyn235sd"
    }
  ]
}
```

Possible PushStatus responses

Message	Status	Description
Approved	success	If push is approved by end-used, this particular message and status will be returned by Push SPI end-point.
Denied	success	If push is denied by end-used, this particular message and status will be returned by Push SPI end-point.
Inprogress	success	If the end-used has not taken any action on the Push notification, this particular message and status will be returned.
Timeout	success	If the Push notification got timed out, this particular message and status will be returned.

Message	Status	Description
Any error message	failure	In case, any other error takes place at the Push SPI end-point, the error message and status as 'failure' will be returned.

API Schema - (Swagger Content)

To view the sample representation of the API schema, see the **Common Open API Spec for all Custom Providers - (Swagger Content)** section in Integrating with Custom SPI.

Creating a Push Custom Provider

- Create a Push custom provider in IDSP using the Provider endpoint-
  `https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders`

  NOTE
  To refer to the Custom Provider SPIs that can be used to perform CRUD operations in IDSP, see here.

Configuring the Global/Tenant Setting attribute

- Configure the global/tenant setting, ***pushProviderToBeUsed*** i n IDSP to set a custom SPI Provider as the Push provider. The default value is "VIP". See Settings for more information.

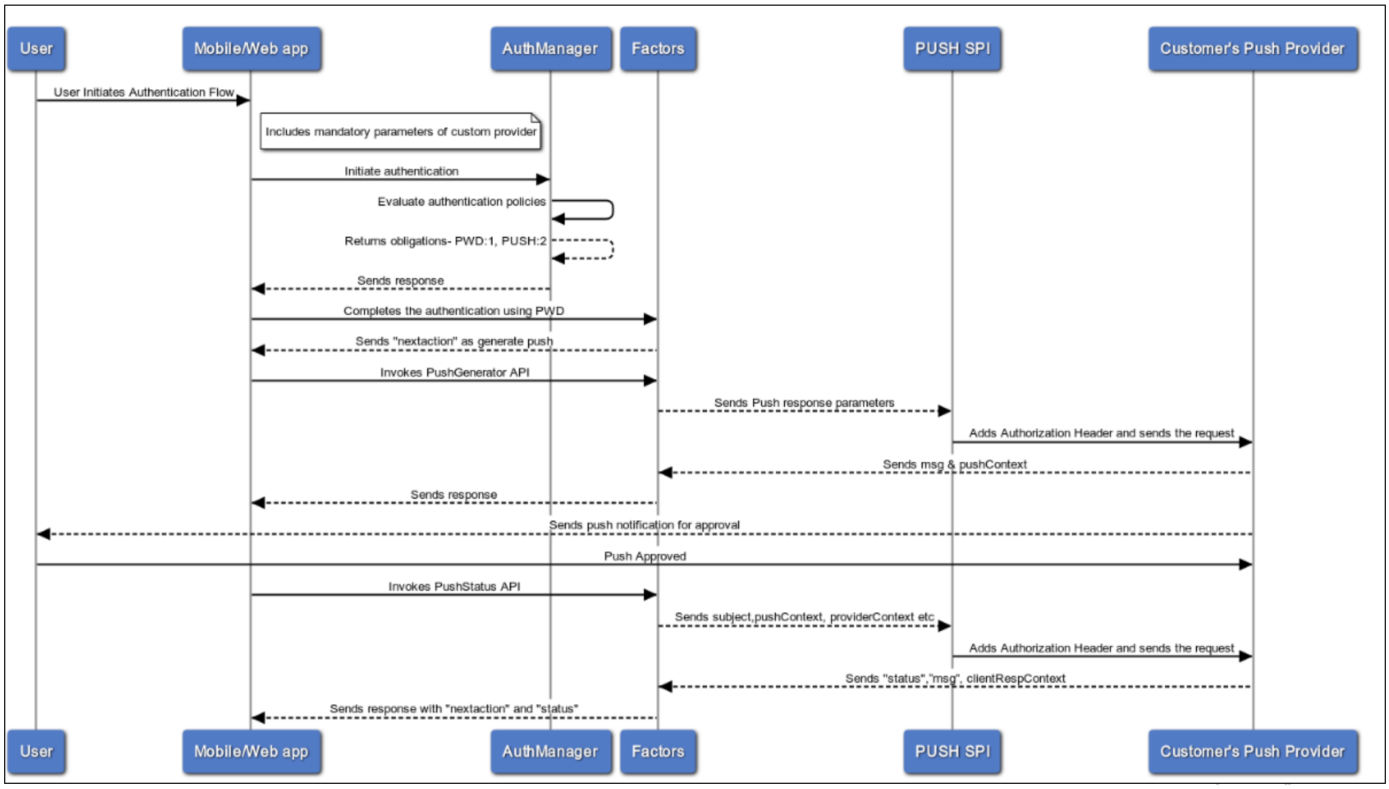
Push Use Case

The following use case describes the integration between Custom Push Provider and IDSP. It is assumed that the customer has configured the following global/tenant setting parameter:

- "pushProviderToBeUsed"

List of participants and what they represent:

- User- One who initiates the authentication flow.
- Application (Mobile/Web App): This is the application that calls the solution's APIs to acquire the identity token and complete the authentication flow.
- IDSP microservices:
  - AuthManager- This is the ***"/auth/v1/authenticate"*** API used to initiate the authentication flow.
  - Factor- This is the ***"/factor/v1/PushGenerator"*** API used to generate the PUSH.
  - PUSH SPI- This is the push wrapper used by the solution to call the custom push provider.
  - Customer's PUSH Provider- This is the customer's implementation of PUSH Provider SPI which is used to deliver push to the end user's phone number.



Flow Description:

1. The User initiates the Authentication Flow.
2. The client app invokes the "auth/v1/authenticate" API to initiate the Authentication flow. The request includes parameters such as user, authN context, clientContext [example:txn identifier].

   NOTE

   In certain cases, a client app may have to send additional parameters that may be mandatorily required by the custom Push provider to function. These parameters, for example, transaction Id, etc can be passed through the "***clientContext***" in the "/auth/v1/authenticate" call. See the following reference sample:

```
{
    "subject": "{{userLoginid}}",
    "channel": "web",
    "ipAddress": "10.10.123.124",
    "action":"authenticate",
    "rememberMe" : true,
    "device": {
        "signature": {
            "iaAuthData": "_v02MyUgNGgYOi88OTk0........."
        }
    },
     "clientContext": [
        {
            "key": "transactionIdentifier",
            "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40"
        }
    ]
}
```

3. Internally, the "/auth/v1/authenticate" API does the following:
   – Checks whether the global/tenant settings parameter, "pushProviderToBeUsed" is configured in the IDSP. In this flow, we have assumed that the setting is configured to a custom push provider.
   – AuthManager communicates with the Policy Engine to evaluate the configured policy rules for the client app and determines the authentication obligations and sends the response to the client app. In the flow, it is assumed that the obligations returned are PWD:1, and PUSH:2.
4. The client successfully completes the first-factor authentication, PWD:1.
5. The Factor API challenges the client with the remaining obligation, that is PUSH:2. Sends the response to the client app with "nextaction" as generate PUSH.
6. The client app invokes the "/factor/v1/PushGenerator" API to generate the PUSH.
7. The Factors API sends the subject, title, pushMsg, deviceIdentifier, providerContext, and so on, to the Push SPI.
8. The PUSH SPI adds an Authorization header (API Key) and sends the PUSH details to the Customer's SPI provider.
9. The Customer's PUSH Provider in return sends success, Message delivery successful, pushContext to the factors and also sends the push notification for approval to user.
10. The user approves the push and sends the response back to the custom provider.
11. The client app invokes the "/factor/v1/PushStatus" SPI to check the status of the push authentication. The Factors API sends subject, pushContext, providerContext, and so on, to the PUSH SPI.
12. The PUSH SPI adds an Authorization header (API Key) and sends the pushContext details to the Customer's SPI provider.
13. The custom push provider sends "status"="success",“msg”: "Approved”, clientRespContext [example:txn identifier] back to the factors.
14. The Factors API sends back the response with "nextaction" as "AUTH_ALLOWED" and "status" as "AUTHENTICATED" to the client app.

Reference Links to:

- VIP Push Authentication

Integrating with External IVR Provider

Integrate Symantec Identity Security Platform with a custom IVR provider to deliver voice-based authentication to mobile devices using the Custom Service Provider Interface (SPI).

This topic includes the following:

- Custom Provider Overview
- Steps to Integrate with Custom IVR SPI
- Implementing the APIs
- API Schema - (Swagger Content)
- Creating a IVR Provider
- Configuring Settings in IDSP
- IVR Use Case

Custom Provider Overview

The Custom SPI is based on an OpenAPI specification and requires a separate web service hosted within your infrastructure. To enable communication between Symantec Identity Security Platform and your IVR provider, you must implement the required API endpoints and configure the system settings detailed in the following sections.

NOTE

- IDSP does not have a built-in provider for IVR.
- The Sample Applications (Mobile & Web) and the Signin UI have not been updated to support IVR.
- If a user has already registered a phone number with IDSP for SMS OTP, the IVR service will utilize that phone number to deliver IVR OTP. The user is requested for inline registration if there is no phone number registered.

Steps to Integrate with Custom IVR SPI

The three steps that the Customers have to perform to integrate with the IVR SPI are:

1. Develop the custom IVR service based on the IVR-Open API Spec provided by IDSP.
2. Create an IVR Custom Provider in IDSP using the Provider endpoint-
   `https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders`

   NOTE

   - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
   - The default value for `<tenantName>` is "default".
   - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
     For example:
     `https://{{host}}/{{tenantName}}/admin/v1/CustomProviders`
     should look similar to:
     `https://example.com/default/admin/v1/CustomProviders`
   - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.
3. Specify the configuration settings in IDSP.

These steps have been described in the subsequent sections.

Implementing the API

The following APIs should be implemented by the customer to integrate their IVR provider with the solution's custom SPI:

- "**IVRMessageSender**" - This API request is used to send the SMS to the custom IVR provider that sends a voice message to the customer's phone.

The API request and response templates are based on the standard Open API spec that IDSP expects the customers to provide during the integration process. The API specifications have been explained in the subsequent sections.

IVRMessageSender API Specification

IVRMessageSender API Specification	
API Name	IVRMessageSender
API URL	*https://<CustomProviderHost>/**IVRMessageSender***
API OPERATION	*POST*
AUTHORIZATION	*<<API Key shared by the custom IVR provider>>*
API RESPONSE CODE	*200 Success, 400 Failures, 500 Server Errors*

Request Payload includes the following parameters:

Parameter	Type	Mandatory (Yes/No)	Description
X-API-KEY	Header	No	API key used to invoke the custom IVR provider
subject	string	Yes	Unique identifier of the user
phoneNumber	string	Yes	Phone number to which the voice SMS must be delivered
otp	string	Yes	otp value that must be delivered
otpValidityInSecs	Integer	Yes	Validity of the OTP in seconds
message	string	Yes	Status message that must be delivered via voice SMS

clientContext	an array of key-value pairs	Yes	Client context parameters received as part of the "/auth/v1/authenticate" call will be passed here by the client. Example: transactionIdentifier, appid, etc.
			`{`
			`"subject": "{{userLoginid}}",`
			`"channel": "web",`
			`"ipAddress": "10.10.123.124",`
			`"action":"authenticate",`
			`"rememberMe" : true,`
			`"device": {`
			`"signature": {`
			`"iaAuthData": "_v02MyUgNGgYOi88OTk0....."`
			`}`
			`},`
			`"clientContext": [`
			`{`
			`"key": "transactionIdentifier",`
			`"value": "540715af-96f2-4b86-b89f-ca0e8ae93f40"`
			`}`
			`]`
			`}`

ahContext	an array of key-value pairs	No	Contains the IDSP context parameters. For information, see the *Sharing IDSP Context with SPI* section in Integrating with Custom SPI.
providerContext	array of key-value pairs	No	Parameters configured as part of the custom IVR provider

Sample Request of "/IVRMessageSender" API:

```
{
  "subject": "nbruce",
  "phoneNumber": 13242342341,
  "otp": 783242,
  "otpValidityInSecs": 300,
  "message": "783242 is your One-Time Password to login. The OTP is valid upto 10 minutes.",
  "clientContext": [
    {
      "key": "transactionIdentifier",
```

```
      "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40"
    }
  ],
  "ahContext": [
    {
      "key": "clientTxnId",
      "value": "cd2f1e6e-c356-49ae-a5b8-c10bf571a9a0"
    },
    {
      "key": "txnId",
      "value": "txn-id-hd26se23"
    }
  ],
  "providerContext": [
    {
      "key": "instanceId",
      "value": "brazil-instance-1"
    }
  ]
}
```

Response Payload should include the following parameters:

Parameter	Type	Mandatory (Yes/No)	Description
status	string	Yes	This parameter returns the voice message delivery status as either a "success" or a "failure".
msg	string	Yes	This parameter returns a status message reflecting the delivery outcome. It returns specific messages based on success or failure instances.
clientRespContext	an array of key-value pairs	No	All the IVR context key-value pairs must be returned by the SPI client as part of the IVR authentication response. For example, transactionId.

Sample Response of "/IVRMessageSender" API:

```
{
  "status": "success",
  "msg": "Message delivered successfully via IVR",
  "clientRespContext": [
    {
      "key": "transactionId",
      "value": "txnId-6hdds733"
    },
    {
      "key": "deviceId",
      "value": "device-hdyn235sd"
    }
  ]
}
```

API Schema - (Swagger Content)

To view the sample representation of the API schema, see the **Common Open API Spec for all Custom Providers - (Swagger Content)** section in Integrating with Custom SPI.

Creating an IVR Custom Provider

- Create an IVR custom provider in IDSP using the Provider endpoint-

  ```
  https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders
  ```

  NOTE
  To refer to the Custom Provider APIs that can be used to perform CRUD operations in IDSP, see here.

Configuring the Global/Tenant Setting attribute

- Configure the global/tenant setting, ***ivrProviderToBeUsed*** in IDSP to set a custom SPI Provider as the IVR provider. See Settings for more information. Default value is "**none**".

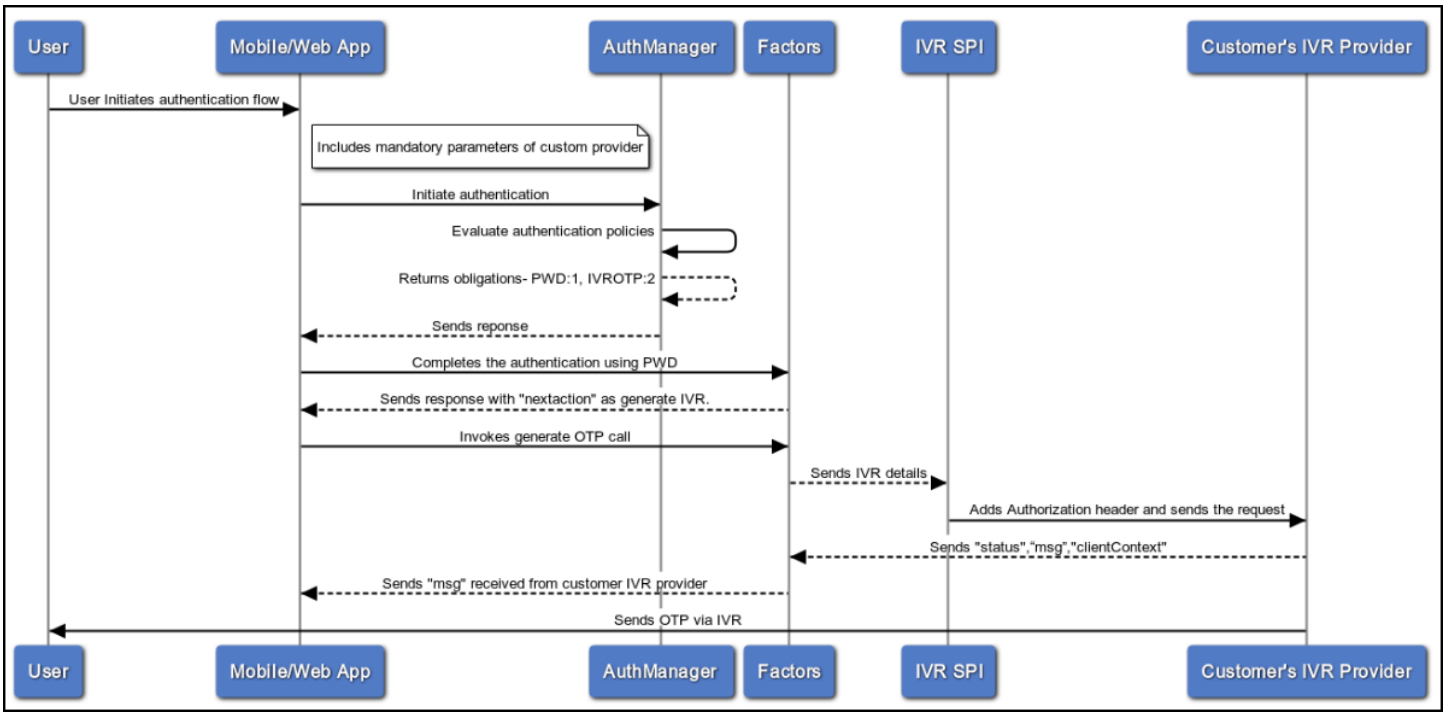
IVR Use Case

The following use case describes the integration between Custom IVR Provider and IDSP. It is assumed that the customer has configured the following global/tenant setting parameter:

- "ivrProviderToBeUsed"

List of participants and what they represent:

- User- User who initiates the authentication flow.
- Application (Mobile/Web App): This is the application that calls the solution's APIs to acquire the identity token and complete the authentication flow.
- IDSP microservices:
  - AuthManager- This is the ***"/auth/v1/authenticate"*** API used to initiate the authentication flow.
  - Factor- This is the ***"/factor/v1/OTPGenerator"*** API used to generate the IVR.
  - IVR SPI- This is the IVR module in IDSP used to call the custom ivr provider in customer's environment.
  - Customer's IVR Provider- This is the customer's implementation of IVR Provider API which is used to deliver ivr to the end user's phone number.



Flow Description:

1. The User initiates the Authentication Flow.
2. The client app invokes the "auth/v1/authenticate" API to initiate the Authentication flow. The request includes parameters such as user, authN context, clientContext [example:txn identifier].

   NOTE

   In certain cases, a client app may have to send additional parameters that may be mandatorily required by the custom IVR provider to function. These parameters, for example, transaction Id, etc. can be passed through the "***clientContext***" in the "/auth/v1/authenticate" call. See the following reference sample:

```
{
    "subject": "{{userLoginid}}",
    "channel": "web",
    "ipAddress": "10.10.123.124",
    "action":"authenticate",
    "rememberMe" : true,
    "device": {
        "signature": {
            "iaAuthData": "_v02MyUgNGgYOi88OTk0........."
        }
    },
     "clientContext": [
        {
            "key": "transactionIdentifier",
            "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40"
        }
    ]
}
```

3. Internally, the "/auth/v1/authenticate" API does the following:

   - Checks whether the global/tenant settings parameter, "ivrProviderToBeUsed" is configured in the IDSP. In this flow, we have assumed that the setting is configured to a custom ivr provider.
   - AuthManager communicates with the Policy Engine to evaluate the configured policy rules for the client app and determines the authentication obligations. Also, send a response to the client app. In the flow, it is assumed that the obligations returned are PWD:1, and IVROTP:2.
4. The client successfully completes the first-factor authentication, PWD:1.
5. The client is challenged with the remaining obligation, that is IVROTP:2.
6. The Factor API sends the response to the client app with "nextaction" as generate IVROTP.
7. The Client app invokes the "/factor/v1/OTPGenerator" API to generate the IVROTP. IDSP sends the IVROTP to the IVR SPI.
8. The Factor API sends IVROTP details, including subject, phoneNumber, otp, message, otpValidity, clientContext, providerContext, and so on, to the IVR SPI.
9. The IVR SPI adds an Authorization header (API Key) and sends the IVROTP details to the Customer's SPI provider.
10. The Customer's IVR Provider in return sends the ivr delivery response to the Factors service. The response includes transaction status and message. It also sends the IVR OTP through phone to the end customer’s phone number.
11. The Factor API sends "msg" received from customer's IVR provider to the client app.

Reference Links to:

- IVR OTP Authentication

Integrating with Custom Authentication Factor

Configure and implement custom authentication factors within Symantec Identity Security Platform policies to meet specialized security and business requirements.

Consider using a custom authentication factor in the following scenarios:

- When a password is hashed in a specific way.
- When implementing a custom challenge/response mechanism with a custom SignIn Client that uses a unique data set.
- To address any other specialized business requirements.

To implement this, develop a Custom SPI that integrates with the standard SignIn UI via the Custom Authenticator API. IDSP also supports fully custom SignIn clients, allowing the custom factor to function as a standalone requirement or as part of a multi-factor authentication policy.

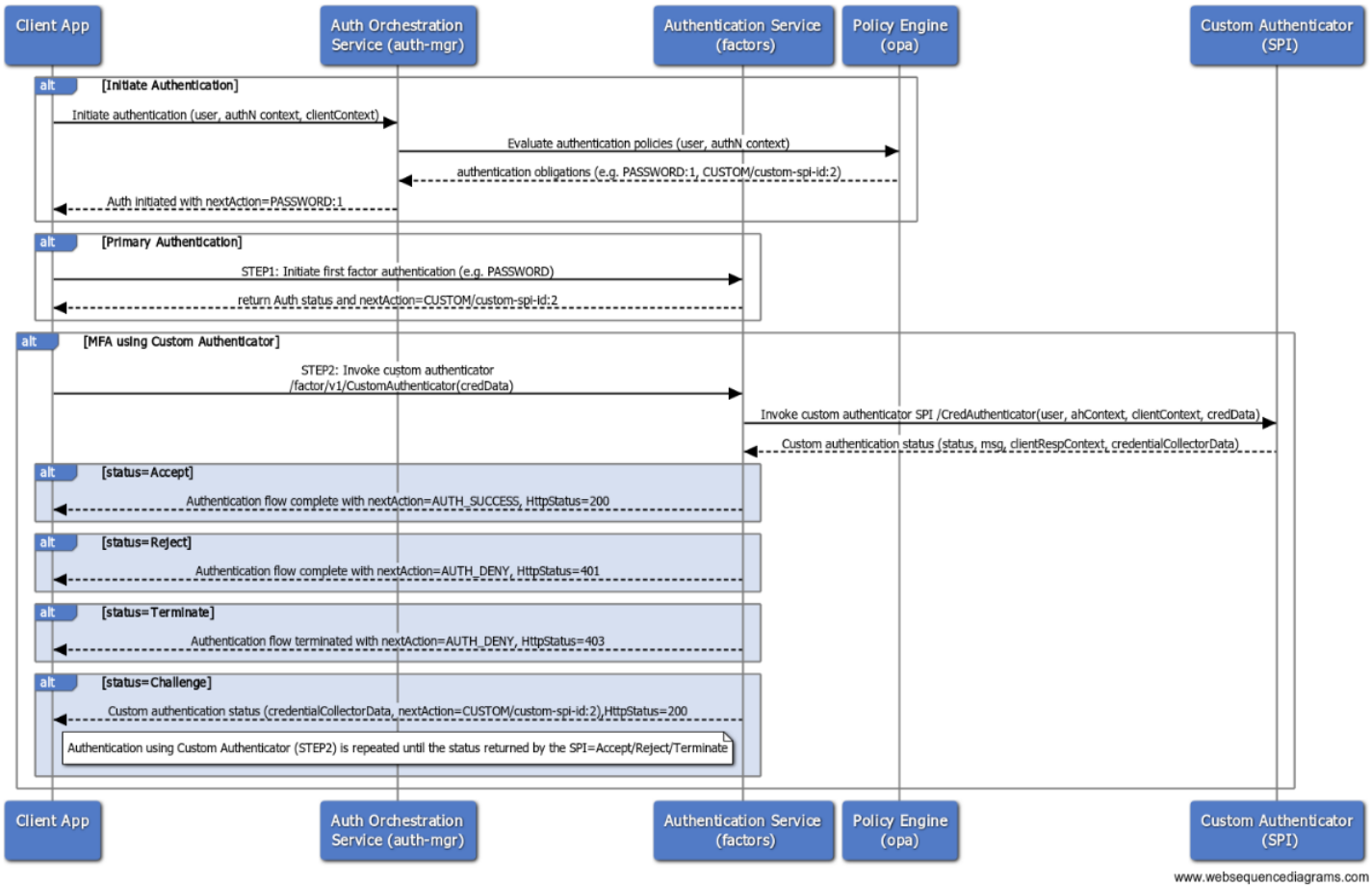
- For information about developing a custom SPI, see Integrating with Custom SPI.
- For information about a SignIn Client, see SignIn User Interface (UI).

This topic covers the following sections:

- Authentication Flow of a Custom Authentication Factor
- Configuration to Support a Custom Authentication Factor
- API Specification
- SPI Specification

Authentication Flow of a Custom Authentication Factor

The following sequence diagram explains the authentication flow between a Client and the Custom Authentication Factor API:



The following process explains the workflow between the Client and the Custom Authentication Factor API when a custom authentication factor is used:

1. The Client invokes `https://${SSP_FQDN}/{{tenantName}}/auth/v1/authenticate`.

   NOTE

   - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
   - The default value for `<tenantName>` is "default".
   - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.

     For example:

     `https://{{host}}/{{tenantName}}/auth/v1/authenticate`

     should look similar to:

     `https://example.com/default/auth/v1/authenticate`
   - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.
2. IDSP evaluates the authentication policy with an optional risk SPI.
3. IDSP constructs a flow state with specific factors to execute, such as F1, F2, and F3, where F2 is a custom authentication factor and CUSTOM_AUTH is its obligation.
4. The Client invokes the /CustomAuthenticator API with the "first time" directive in OperationalContext, an optional parameter in the request.

This step begins the interaction with the custom SPI that is represented by the custom provider, which is associated with the custom authentication factor. The interaction continues until the API indicates AUTH_ALLOW or AUTH_DENY.

5. The custom SPI returns challenge directive for the Client to display.
6. The Client invokes the /CustomAuthenticator API with the response directive in the CredData context of the request.
7. The custom SPI returns Accept/Reject/Challenge/Terminate as the status as per the custom SPI implementation.
   - **Accept**: Indicates that the custom SPI has successfully authenticated the user. IDSP advances the authentication flow as per authentication policy.
   - **Challenge**: Indicates that the custom SPI requests a challenge. IDSP does not proceed to the next authentication factor or complete the flow. The Client challenges again with "credData" as returned by the Custom SPI. This "challenge" does not impact the flow state as asserted by IDSP and does not impact the next authentication factor if there is one. Any state to be passed to the custom SPI is done by the client using dedicated Credential Data.
   - **Reject**: Indicates that the custom SPI has rejected credentials and expects the Client to redo the operation. IDSP increments the user lock strike count, and does not advance the authentication flow.
   - **Terminate**: Indicates that the custom SPI has failed to authenticate the user. IDSP fails the authentication flow.

For more information about the flow, see Examples of Using a Custom Authentication Factor.

> **NOTE**
> The OperationalContext of the custom SPI may include the cookie/header directives to be processed by IDSP when returning the API response.

Configuration to Support a Custom Authentication Factor

The following configurations are required to use a custom authentication factor:

Authentication Policy

Configure the authentication policy to use the `CUSTOM/<custom-provider-guid>:<factor-level>` obligation.

Example: CUSTOM/MyCustomAuthProvider:2

A custom provider is resolved in the resource tenancy first. If it is not found, then it is resolved in the system tenancy.

Custom Provider - providerProperties

A custom provider uses the providerProperties key-value pairs to define a provider-specific behavior. There are two property types:

- **Custom**: All custom properties are published into the providerContext of the SPI request and are consumed by the custom SPI.
- **Well-known**: All well-known properties begin with the **spi.** prefix, representing the directives to be performed by IDSP when sending requests to and receiving responses from the custom SPI.

The following table lists the providerProperties that must be configured to support the use of a custom authentication factor:

Type	providerContext	Description
Custom	**key**: <custom-key> **value**: <custom-value>	Publishes a custom key-value pair into the ProviderContext of the custom SPI. The value in a key-value pair supports dynamic substitution using the ${keyword} notation. For information, see Dynamic Environment Substitution.

Well-known	**key**: spi.discover.capabilities **value**: CSV list of capabilities	Specifies the capabilities that are supported by the custom provider using multiple, comma-delimited placeholders. This enables IDSP to recognize the configured Custom Provider as having custom authentication capabilities, for example: • **customauthenticator**: Specifies that the custom SPI supports a custom authentication factor that is accessible through the "/CustomAuthenticator" runtime flow.
Well-known	**key**: spi.auth.factor.flowid **value**: N/A	Identifies the flow to be performed by the SignIn Client. If this property is defined, this flowid is communicated to the SignIn Client when a custom authentication factor is to be invoked the first time. • Prior to the first invocation of the custom SPI, if no such property exists, the flowid comes from ProviderContext to be passed into the custom SPI. The SignIn Client is expected to pass this value into the custom SPI via OperationalContext of the request from the custom SPI. • On subsequent flows within the same custom authentication factor, the value is derived from OperationalContext that is returned by the custom SPI.

Well-known	**key**: spi.auth.factor.amr **value**: <amr-value>	Represents one or more AMR values (in CSV format) of the custom authentication factor. These AMR values are to be associated with the custom provider. You can enter multiple comma-delimited values from the following list: • PWD: Password style authentication • OTP: One time code (SMS, EMAIL, IVR) • FIDO: Passwordless • HWK: Hardware Key (Yubikey) • MOTP: Mobile one time code • PUSH: Mobile push • POP: SiteMinder X509 integration use case • SWK: N/A • WIA: SiteMinder IWA/Kerberos integration use case • <custom>: A custom AMR value If multiple instances of this key are provided, the last one takes precedence. If this key is specified by ProviderContext and also returned by the OperationalContext of the custom SPI, then the value from the OperationalContext takes precedence.
Well-known	**key**: spi.request.header.proxy **value**: <*header-name*>	Identifies the header name in an API request that must be included as an HTTP header in the request of the custom SPI.
Well-known	**key**: spi.request.cookie.proxy **value**: <*cookie-name*>	Identifies the cookie name in an API request that must be included as an HTTP cookie in the request of the custom SPI.
Well-known	**key**: spi.response.header.proxy **value**: <*header-name*>	Identifies the header name in the response of the custom SPI that must be included as an HTTP header in an API request.
Well-known	**key**: spi.response.cookie.proxy **value**: <*cookie-name*>	Identifies the cookie name in the response of the custom SPI that must be included as an HTTP cookie in an API request.

Custom Provider - providerHeaders

A custom provider uses providerHeaders in a key-value pair format to define provider-specific headers to be sent to the custom SPI. Similar to "providerProperties", the headers definitions also supports dynamic substitution using the ${keyword} notation. For information, see Dynamic Environment Variable Substitution.

API Specification

API Name	**CustomAuthenticator**
API URL	https://${SSP_FQDN}/<tenant-name>/factor/v1/CustomAuthenticator

API OPERATION	POST
AUTHORIZATION	Client-Access-Token
API RESPONSE CODE	• 200 Success (Allow or Challenge) • 401 Unauthorized (Reject) • 403 Forbidden (Terminate) • 400 Bad Request (API/SPI error) • 500 Internal Server Error (API/SPI error)

Note: A custom SPI that expects to plug and play with IDSP's out-of-the-box SignIn UI SignIn Client must support the challenge/response directives as documented.

API Request

The API request to a custom provider can contain cookies, headers, standard headers, and regular x-flow-state header along with the following properties:

Property	Type	Mandatory?	Description		
operationalContext	Array of key-value pairs	Yes	Specifies the optional directives as hints to a custom SPI.		
			Key	**Value**	**Description**
			(Optional) spi.auth.factor.flowid	<string>	Client submits a flowid for the custom SPI to execute. Custom provider uses the same property to let the Client know the flowid to start with. Subsequent invocations of the custom SPI return next flowid for the Client to request.
			spi.auth.factor.firsttime	true/false	Client submits this flag for the custom SPI to know that it is being invoked for the first time. Custom SPI supporting such behavior would respond back with the challenge (in credData) for the Client to challenge the user with. **Note:** • Custom SPI that expects to plug and play with the out-of-the-box SignIn UI/ custom SignIn Client must support this.

			<custom-key>	<string>	Any custom key and related value the Client considers to be as an operational input to the custom SPI.
credData	Array of key-value pairs	Yes	Specifies the credential data to be validated by the custom authenticator SPI using the challenge/response approach.		
			Key	**Value**	**Description**
			<custom-key>	<string>	Specifies the custom data to be exchanged between the custom SPI and out-of-the-box SignIn UI/custom SignIn Client.
			spi.auth.factor.request	<string>	Indicates the credential exchange between the default IDSP SignIn Client and the custom SPI in the future, IDSP SignIn Client has a support for the Custom Authenticator API.

API Response

The API response from a custom provider can contain cookies, headers, regular flow state response that includes the following "data" and "additional" properties:

Property under "data"	**Type**	**Description**		
status	String	Indicates the status of the custom authentication as given by the SPI. Supported values are Accept, Reject, Challenge, or Terminate.		
clientRespContext	Array of key-value pairs	Specifies the additional context data that must be returned to the Client by the SPI, at the end of the authentication by the custom authentication factor.		
credData	Array of key-value pairs	Specifies the credential data to be validated by the Custom Authenticator API.		
		Key	**Value**	**Description**
		<custom-key>	<string>	Represents the custom data to be exchanged between the custom SPI and out-of-the-box SignIn UI/custom SignIn Client.

		spi.auth.factor.challenge	<string>	Represents the credential exchange (Challenge, Reject) between the default custom SPI and the IDSP SignIn Client in the future, when IDSP SignIn Client supports the Custom Authenticator API.
		spi.auth.factor.response	<string>	Represents the credential exchange (Accept, Terminate) between the default custom SPI and the IDSP SignIn Client in the future, when IDSP SignIn Client supports the Custom Authenticator API.

Property under "additional"	Type	Description
isCustom	String	If set to **true**, indicates that the current factor is **CUSTOM**.
customProvider	String	Specifies the name of the custom provider.
(Optional) customFlowId	String	Specifies the flowid returned by the custom SPI.

For more information about these properties, see Examples of Using a Custom Authentication Factor.

SPI Specification

SPI Name	CustomAuthenticator
SPI URL	https://<CustomProviderHost>/CustomAuthenticator
SPI OPERATION	POST
AUTHORIZATION	As per the SPI security model: • OAuth Client Assertion • OAuth ClientId & Secret • ApiKey • Basic Auth
SPI RESPONSE CODE	200 Success, 400 Failure, 500 Server Error

When the request comes to the IDSP endpoint that is hosted on the "signin domain", IDSP proxies cookies and/or headers to and from the custom SPI as per the configuration of the custom provider. If the custom authentication factor specifies that a cookie named "mycookie" must be acquired from the request and passed into the Custom Authenticator SPI, then the actual invocation of the custom SPI includes the cookie and/or header in the custom SPI request's **ahContext**. If the response from the SPI indicates that a cookie and/or header must be set on the way back, the response from the factors API includes the native cookie and/or header as indicated. For the details, see the sample response payload of the SPI.

SPI Request

The SPI request to a custom provider can contain cookies and headers that are requested by SPI, along with the following parameters:

Parameter	Type	Mandatory	Description
subject	String	Yes	Specifies the unique identifier of the user representing **user_loginid** semantics as configured in the ID store config resource.
clientContext	Array of key-value pairs	Yes	Specifies that the Client context parameters that are received as part of the programmatic/authenticate or web/authorized (IT) flows are passed here as key-value pairs, which are understood by client. Example: transactionIdentifier, appid.
ahContext	Array of key-value pairs	Yes	Contains the IDSP context parameters. For information, see the *Sharing IDSP Context with SPI* section in Integrating with Custom SPI.
providerContext	Array of key-value pairs	Yes	Specifies the custom provider-specific configuration. For information, see Integrating with Custom SPI.
operationalContext	Array of key-value pairs	Yes	Specifies the optional directives to be processed by SPI. For information, see OperationalContext in the API Request section. In addition to the directives passed by the Client into the Custom Authenticator API, the following directive is added by IDSP:
			Key / **Value**
			spi.auth.factor.amr / Represents the AMR values as specified by the custom authenticator provider.
metadataContext	Array of key-value pairs	Yes	Specifies the metadata of the application that is requesting authentication.
credData	Array of key-value pairs	Yes	Specifies the credential data that is required by the custom SPI to process authentication.

SPI Response

The SPI response from a custom provider can contain cookies and headers that are requested by SPI's response, along with the following parameters:

Parameter	Type	Description
msg	String	Specifies the message to be logged.
status	String	Specifies the status of the custom authentication. The supported values are Accept, Reject, Challenge, or Terminate.
credData	Array of key-value pairs	Specifies the data to be collected by SignIn Client, which is required by the Credential Authenticator SPI, as key-value pairs.
clientRespContext	Array of key-value pairs	Specifies the additional context data to be returned to SignIn Client by the SPI at the end of the authentication by the custom authentication factor.
operationalContext	Array of key-value pairs	Specifies the optional directives that are indicated by the SPI to be processed by IDSP. The following directives are processed:
		Key / **Value**

		spi.response.header.proxy	Represents the name of the header present in the SPI response to proxy to the API response as an http header.
		spi.response.cookie.proxy	Represents the name of the cookie present in the SPI response to proxy to the API response as an http header.
		spi.auth.factor.flowid	Represents the "flow id" as a hint to the signin client to know what specific flow to perform.
		spi.auth.factor.amr	Represents the "amr" value of the custom authentication, returned after the user has successfully completed authentication. If this value is provided by the SPI, it overwrites the value that is configured on the custom provider.

For more information about the flow, see Examples of Using a Custom Authentication Factor.

Examples of Using a Custom Authentication Factor

Explore detailed implementation examples of Custom Service Provider Interfaces (SPI) within Symantec Identity Security Platform, including runtime interaction models, state management, and multi-factor authentication workflows.

The following examples illustrate use of custom SPI under few different conditions and configurations. The example uses an out of the box **Sample Demo** custom SPI that is deployed by the **Sample App** chart in the same cluster where Symantec Identity Security Platform is deployed. IDSP accesses this custom SPI using an internal service name as configured in the custom provider representing the SPI, using no functional security. In a real deployment, a custom SPI is accessed on its published URL that is protected by one of the supported security methods.

The examples demonstrate:

- Runtime interaction model depicting SignIn Client → IDSP → SPI workflow model and the use of runtime semantics, such as:
  - "First Time" and "FlowID" state management
  - Out of the box Challenge/Response interaction
- Propagation of headers and cookies as per directives that are configured in the custom provider or as instructed by SPI's OperationalContext.
- Propagation of providerContext and providerHeaders as configured in the custom provider.
- Dynamic substitution of custom providerHeader and providerContext directives using data from Identity Profile, ID Token, and Application Metadata.
- Propagation of ClientContext and CredData Context.

This topic covers the following sections:

- Using Provider Helper API to get Authentication Providers Information
- Example 1: Admin client using custom SPI to obtain access token for an administrator
- Example 2: Multi-factor flow with different custom SPIs to complete OpenID Connect authentication flow

Using Provider Helper API to get Authentication Providers Information

To view the different custom providers, use the Provider Helper API.

Request and response formats of the ProviderHelper API in the following format:

Request

```
GET https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/ProviderHelper
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/ProviderHelper`
  should look similar to:
  `https://example.com/default/admin/v1/ProviderHelper`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Response


```
[
    {
        "id": "91d14d97-d82d-4d7c-90b7-eddb18bb4f7c",
        "name": "Test",
        "type": "openid",
        "purpose": "IDP",
        "predicate": "Test"
    },
    {
        "id": "{idstore}74929623-e0c8-461b-ad1b-6fc97abc3719",
        "name": "{idstore}TestConfig1",
        "type": "LDAP",
        "purpose": "PASSWORD",
        "predicate": "TestConfig1"
    },
    {
        "id": "{idstore}d15e9718-404d-4345-9198-1625d093c079",
        "name": "{idstore}TestConfig",
        "type": "LDAP",
        "purpose": "PASSWORD",
        "predicate": "TestConfig"
    },
    {
        "id": "ea7ae57b-104c-45fb-91fe-51c29ff56596",
        "name": "customSampleProvider3",
        "type": "CUSTOM",
        "purpose": "CUSTOM",
        "predicate": "customSampleProvider3"
    },
    {
        "id": "2539cd04-02f1-453e-ab20-8b1b22f84f28",
```

```
        "name": "customSampleProvider2",
        "type": "CUSTOM",
        "purpose": "CUSTOM",
        "predicate": "customSampleProvider2"
    },
    {
        "id": "29fdfd06-e5ad-425f-b108-f681ec108b8d",
        "name": "customSampleProvider",
        "type": "CUSTOM",
        "purpose": "CUSTOM",
        "predicate": "customSampleProvider"
    },
    {
        "id": "{custom}29fdfd06-e5ad-425f-b108-f681ec108b8d",
        "name": "{custom}customSampleProvider",
        "type": "CUSTOM",
        "purpose": "PASSWORD",
        "predicate": "customSampleProvider"
    },
    {
        "id": "{custom}2db88c50-93a9-4618-ad64-23b9291811cb",
        "name": "{custom}MyCustomAuthProvider",
        "type": "CUSTOM",
        "purpose": "PASSWORD",
        "predicate": "MyCustomAuthProvider"
    }
]
```


Example 1: Admin client using custom SPI to obtain an access token for an administrator

This example demonstrates a simple custom SPI setup that authenticates an administrator using an out of the box **Sample Demo** custom SPI. It extends the "Acquire Admin Token" example in IDSP's out of the box postman collection under "Tenant Admin Operations" to obtain an Access Token. Normally, the authentication step requires a single ID store password, however, for this example, the authentication flow is updated to require an ID store password or a custom factor that is implemented by out of the box Sample SPI Demo provider.

Step 1: Create a custom provider definition for "MyCustomAuthProvider"

This provider uses the out of the box Sample Demo custom provider that is deployed by Sample App chart.

Request


```
curl --location --request POST 'https://example.com/default/admin/v1/CustomProviders' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer
 eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0UG
xbG9Fq6IWxYUEVkPVXKWhKbusXyhyqLWyPfgsRQ_Adx3I2NFNAHM9P3YZpX4JLHICwrjKUQ2cIkPDrtZI_8pl9UERdy7gQasvnd8kk2R8AKrCEvPrUP7aAK
 \
--data-raw '    {
       "providerName": "MyCustomAuthProvider",
       "version": "1.0",
       "apiKey":
 "0BX9mtir0+W8C3r4UqSL39POSo1BO35LYdCVf7Dp9WCH7m9mu4abrpnJs5A8AknrYwejLHcq9S7DNYj5Z8FKlpxFkHONl3e2wEjUiJ8Pf3UyJwrTXv5Cc
       "certAlias": "",
```

```
        "url": "https://ssp-sample-ssp-sample-app-spi-demo.dev-sample.svc:443",
        "providerHeaders": [
            {
                "name": "titleFromIdProfile",
                "value": "${idProfile.title}"
            },
            {
                "name": "userRiskScoreFromIdToken",
                "value": "${idToken.user_riskscore}"
            },
            {
                "name": "customHeader",
                "value": "customValue"
            }
        ],
        "providerProperties": [
            {
                "name": "spi.discover.capabilities",
                "value": "customauthenticator"
            },
            {
                "name": "orgFromAppMetadata",
                "value": "${appMetadata.org}"
            },
            {
                "name": "customProp",
                "value": "12345"
            },
            {
                "name": "spi.auth.factor.amr",
                "value": "PWD"
            }
        ],
        "ignoreSSLValidation": true,
        "sslMode": "VERIFY_CA",
        "spiReadTimeoutMillis": "60000",
        "spiConnectTimeoutMillis": "60000",
        "securityMethod": "apikey",
        "oauthResourceInAzRequest": "none",
        "oauthAssertionAud": "",
        "oauthTokenIgnoreSslValidation": false,
        "oauthTokenSslMode": "VERIFY_CA",
        "oauthTokenReadTimeoutMillis": "10000",
        "oauthTokenConnectTimeoutMillis": "10000",
        "skewTimeSecs": "0"
    }'
```

Response

```
{
   "providerName": "MyCustomAuthProvider",
   "version": "1.0",
```

```
   "apiKey": "uSx8ixLc7OFxmWDfzSgJppKTwz+HckRC8AM+ei2hnAr2JyRHjLdrXPfA2/
RCAGGBbyFykMWkR0Uqf6KJk8b9IXvqTGa2Gecn4xlBnpiqpsevCE4X1q7pG
+obyPlHNlrlbfamMrP/1PD5eBcwGZI5jazihze39z4HuooAKMzMapi0Jg7t/Uv4YxTlIstX9EfKNixLT8ZNpzzA0T9jLjonfQ==",
   "certAlias": "",
   "url": "https://ssp-sample-ssp-sample-app-spi-demo.dev-sample.svc:443",
   "providerHeaders": [
       {
           "name": "titleFromIdProfile",
           "value": "${idProfile.title}"
       },
       {
           "name": "userRiskScoreFromIdToken",
           "value": "${idToken.user_riskscore}"
       },
       {
           "name": "customHeader",
           "value": "customValue"
       }
   ],
   "providerProperties": [
       {
           "name": "spi.discover.capabilities",
           "value": "customauthenticator"
       },
       {
           "name": "orgFromAppMetadata",
           "value": "${appMetadata.org}"
       },
       {
           "name": "customProp",
           "value": "12345"
       },
       {
           "name": "spi.auth.factor.amr",
           "value": "PWD"
       }
   ],
   "createdBy": {
       "principalType": "USER",
       "principalId": "fc3108d9-86ce-4829-a034-553a7375dd6d",
       "principalName": "malbail"
   },
   "createdDateTime": "2022-04-16 08:34 PM UTC",
   "origin": "tenant",
   "ignoreSSLValidation": true,
   "providerId": "2db88c50-93a9-4618-ad64-23b9291811cb",
   "sslMode": "VERIFY_CA",
   "spiReadTimeoutMillis": "60000",
   "spiConnectTimeoutMillis": "60000",
   "clientId": "",
   "clientSecretAlias": "",
   "oauthScopes": "",
   "securityMethod": "apikey",
```

```
    "oauthTokenUrl": "",
    "oauthDiscoveryUrl": "",
    "oauthResourceInAzRequest": "none",
    "oauthAssertionKidType": "",
    "oauthAssertionAud": "",
    "oauthTokenIgnoreSslValidation": false,
    "oauthTokenSslMode": "VERIFY_CA",
    "oauthTokenCertificateAlias": "",
    "oauthTokenRequestCtxBase64": "",
    "oauthTokenReadTimeoutMillis": "10000",
    "oauthTokenConnectTimeoutMillis": "10000",
    "skewTimeSecs": "0"
}
```

Step 2: Specify ID for "MyCustomAuthProvider" custom provider as a factor in the Authentication Policy "AdminLoginPolicy"

Request

```
curl --location --request PATCH 'https://example.com/default/admin/v1/AuthPolicies/ba8a5b86-
e28c-4cfc-9380-381ad749fcfb' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer
 eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0UG
xbG9Fq6IWxYUEVkPVXKWhKbusXyhyqLWyPfgsRQ_Adx3I2NFNAHM9P3YZpX4JLHICwrjKUQ2cIkPDrtZI_8pl9UERdy7gQasvnd8kk2R8AKrCEvPrUP7aAK
 \
--data-raw '{
   "Operations": [
       {
           "op": "add",
           "path": "/rules/[id eq \"1c722537-aaa2-4d62-a476-147e693c097a\"]/result/obligations",
           "value": "CUSTOM/2db88c50-93a9-4618-ad64-23b9291811cb"
       }
   ]
}'
```

Response

```
{
   "policyId": "ba8a5b86-e28c-4cfc-9380-381ad749fcfb",
   "policyName": "AdminLoginPolicy",
   "description": "Default policy for Tenant Admin Console",
   "status": "active",
   "matchAnyApp": false,
   "apps": [
       {
           "name": "AdminConsole",
           "id": "$TENANT_ADMINCONSOLE"
       },
       {
           "name": "DefaultTenantClient",
           "id": "7bd13001-f024-45cf-92c2-d986381694a9"
       }
   ],
   "rules": [
```

```
        {
            "id": "1c722537-aaa2-4d62-a476-147e693c097a",
            "conditions": {
                "channel": {
                    "operator": "in",
                    "value": [
                        "web",
                        "mobile"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "CUSTOM/2db88c50-93a9-4618-ad64-23b9291811cb",
                    "PASSWORD"
                ],
                "msg": "This policy is for admin login"
            }
        }
    ],
    "createdDate": "2022-03-06 09:05 AM UTC",
    "updatedBy": {
        "principalType": "USER",
        "principalId": "fc3108d9-86ce-4829-a034-553a7375dd6d",
        "principalName": "malbail"
    },
    "updatedDate": "2022-04-16 08:37 PM UTC",
    "preventedOperations": "Delete"
}
```


Step 3: Initiate authentication for user malbail (using same client Access Token as before)

Request


```
curl --location --request POST 'https://example.com/default/auth/v1/authenticate' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer
 eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0UG
xbG9Fq6IWxYUEVkPVXKWhKbusXyhyqLWyPfgsRQ_Adx3I2NFNAHM9P3YZpX4JLHICwrjKUQ2cIkPDrtZI_8pl9UERdy7gQasvnd8kk2R8AKrCEvPrUP7aAK
 \
--data-raw '{
    "channel": "web",
    "subject": "malbail",
    "ipAddress": "10.10.123.124",
    "authContext": [
        {
            "key": "useTenantKeyForITEncryption",
            "value": "false"
        }
    ]
}'
```


Response

```
{
   "flowState":
 "TWJZZzkzNW12ZGwwOHRhK3BFL2dDTXR5dDY4MTFVZmxqTWNZU0hCblpkamcxUzFPT21lSkRnQzRhQWpMaWg4S2pFNFNzSHVnOWxrVCtMV3RGY2t2UDBle
   "nextaction": "FACTOR_SELECTION",
   "currentFactors": [
       "PASSWORD",
       "CUSTOM/MyCustomAuthProvider"
   ],
   "additional": {
       "userName": "malbail",
       "currentFactorLevel": 1
   }
}
```


Step 4: Select custom authentication as the factor to use

Request


```
curl --location --request POST 'https://example.com/default/auth/v1/SelectedFactor' \
--header 'X-Flow-State:
 TWJZZzkzNW12ZGwwOHRhK3BFL2dDTXR5dDY4MTFVZmxqTWNZU0hCblpkamcxUzFPT21lSkRnQzRhQWpMaWg4S2pFNFNzSHVnOWxrVCtMV3RGY2t2UDBleC
 \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer
 eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0UG
xbG9Fq6IWxYUEVkPVXKWhKbusXyhyqLWyPfgsRQ_Adx3I2NFNAHM9P3YZpX4JLHICwrjKUQ2cIkPDrtZI_8pl9UERdy7gQasvnd8kk2R8AKrCEvPrUP7aAK
 \
--data-raw '{
   "factor": "CUSTOM/2db88c50-93a9-4618-ad64-23b9291811cb"
}'
```


Response


```
{
   "flowState":
 "TWJZZzkzNW12ZGwwOHRhK3BFL2dDTXR5dDY4MTFVZmxqTWNZU0hCblpkamcxUzFPT21lSkRnQzRhQWpMaWg4S2pFNFNzSHVnOWxrVCtMV3RGY2t2UDBle
   "nextaction": "CUSTOM_AUTH",
   "currentFactors": [
       "PASSWORD",
       "CUSTOM/2db88c50-93a9-4618-ad64-23b9291811cb"
   ],
   "additional": {
       "currentFactorLevel": 1,
       "isCustom": true,
       "isCustomFirstTime": true,
       "customProvider": "2db88c50-93a9-4618-ad64-23b9291811cb"
   }
}
```


Step 5: Authenticate using custom SPI

Note that this example does not use the Challenge/Response workflow. The Client invokes the custom SPI (as per Sample Demo SPI semantics), instructing it to "accept" without actually challenging for credentials. A real custom SPI would request and authenticate credentials using the client/SPI workflow as covered above.

Request

```
curl --location --request POST 'https://example.com/default/factor/v1/CustomAuthenticator' \
--header 'Content-Type: application/json' \
--header 'X-Flow-State:
 TWJZZzkzNW12ZGwwOHRhK3BFL2dDTXR5dDY4MTFVZmxqTWNZU0hCblpkamcxUzFPT21lSkRnQzRhQWpMaWg4S2pFNFNzSHVnOWxrVCtMV3RGY2t2UDBleC
 \
--header 'Authorization: Bearer
 eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0UG
xbG9Fq6IWxYUEVkPVXKWhKbusXyhyqLWyPfgsRQ_Adx3I2NFNAHM9P3YZpX4JLHICwrjKUQ2cIkPDrtZI_8pl9UERdy7gQasvnd8kk2R8AKrCEvPrUP7aAK
 \
--header 'Cookie: cookie-by-spi=sample-cookie-by-spi' \
--data-raw '{
   "operationalContext": [
   ],
   "credData": [
       {
           "key": "spi.auth.factor.response",
           "value": "accept"
       }
   ]
}'
```

SPI Request to the Custom SPI

Output from the Sample Demo Custom SPI provider "MyCustomAuthProvider" showing the SPI request.

Notice substitutions that are made by IDSP as per providerContext and header configuration in the custom provider "MyCustomAuthProvider". As this authentication request did not include an ID Token, the "${idToken.user_riskscore}" got removed from the headers sent to the SPI. If the authentication request did include '"id_token_hint" : "<existingIdToken>" (as defined in provider configuration) in the payload, the header would have been sent with the proper value.

```
2022-04-16 20:41:54.141  INFO 1 --- [nio-8092-exec-3] i.s.a.CustomAuthenticatorApiController   :
----- Custom Auth Sample Provider Request v1.0.0000 -----   ************************************** REQUEST
subject=MalBAIL
----- clientContext -----
----- ahContext -----
tid=51d8c90f-bd95-41eb-a73d-fb725b84cbb6
tname=default
clientIp=10.230.9.183
userIp=10.230.9.183
txnId=dd7be0a1-8b46-431b-b3d7-e6c1277a20ba
clientTxnId=
locale=en-US
userId=fc3108d9-86ce-4829-a034-553a7375dd6d
userUniversalId=MalBAIL
idpUserGuid=MalBAIL
idpGuid=d15e9718-404d-4345-9198-1625d093c079
idpName=TestConfig
idpType=ldap
completedAmrNames=[]
clientSslCert=-----BEGIN%20CERTIFICATE-----%0AMIIDXDCCAkSgAwIBAgIBATANBgkqhkiG9w0BAQsFADBMMRUwEwYDVQQDDAxWYWRp
%0AbSBMYW5kZXIxCzAJBgNVBAYTAlVTMSYwJAYJKoZIhvcNAQkBFhd2YWRpbS5sYW5k
%0AZXJAb3JhY2xlLmNvbTAeFw0yMjA0MDYyMjI1MDJaFw0yMzA0MDYyMjI1MDJaMEwx
%0AFTATBgNVBAMMDFZhZGltIExhbmRlcjELMAkGA1UEBhMCVVMxJjAkBgkqhkiG9w0B
%0ACQEWF3ZhZGltLmxhbmRlckBvcmFjbGUuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOC%0AAQ8AMIIBCgKCAQEAokwo6Wcf2yy
```

%2FqKEjK1zEycs1RnY2ZyXmJmGLXlG79KuSuXFz%0AC%2B96yApNcUu4P9H4bTWFvqbeEd0eJLCFp0aOM
%2BTe6caJ4Q6x5DDQ3LE1bIDLsyeJ%0ATlHYWN%2FZ6%2BoHRdJ5fjgPgvx9clBG1JWO9NTuSPW
%2BUnCJu3FJLfgZ6BbPDV6dizB%2F%0Aatjgvdx%2FDk6aeVkcQ8kVbSTAuHMtkOh%2FJEcaTRUPtYb42FWlPMMZ0yHPZkq4NovW
%0Awukwn4nteXFWaSqORW%2BJK%2BBSK605DLRtI9U14hCsWH8YjcF0bSemRNPil%2FcutpUX
%0A2jkvZ7a3PF8s7SKkaDBlwtf9bqDYcprdlTrjYwIDAQABo0kwRzAOBgNVHQ8BAf8E%0ABAMCB4AwFgYDVR0lAQH
%2FBAwwCgYIKwYBBQUHAwIwHQYDVR0OBBYEFDMtfLKBYjcb
%0AnFx46bWVW3TOZ5nVMA0GCSqGSIb3DQEBCwUAA4IBAQBedJG7hGaITf7dMNQr2%2BCQ%0A%2Fcd
%2BSdp83aW9%2Fdi5ZYHMCxIR0yLFg2kQN6xMgkQUXLP3AqoOpiNC6ZZ9YldX8h9J%0A1qLhxlIgCt
%2FmrkAzfPAWxOdD0qhlH7jyYO692c0aYSZDUAVF
%2BtsYkiBJtMzZQ1d4%0AUGdtyMc9IEu5TNy0zZfCODBkKmXX18acep3xYD7S8ky2nfqc2ZyThx1j3EDeladv
%0AEZ%2B30jmOdOth5ycG0kBgbBZR9zLem3lU78yZPkzG%2BP7tHMA5vFzp0p9762A%2Fc5ZF
%0AUsqFOx3AluNr8XtYXiYxywFB3a1SVQW738548mAtELzAeP1kfjMfX0q6pASUXEy%2B%0A-----END%20CERTIFICATE-----%0A
x-ssp-base-uri=https://example.com
x-ssp-issuer-url=https://example.com/default
x-ssp-host=example.com
x-ssp-path=/default
providerName=MyCustomAuthProvider
providerId=2db88c50-93a9-4618-ad64-23b9291811cb
----- providerContext -----
orgFromAppMetadata=123
customProp=12345
----- metadataContext -----
env=test
org=123
region=us-east-1
----- operationalContext -----
----- credData -----
spi.auth.factor.response=accept
----- headers -----
accept=[application/json]
content-type=[application/json]
x-client-transaction-id=[]
x-transaction-id=[dd7be0a1-8b46-431b-b3d7-e6c1277a20ba]
x-forwarded-for=[]
x-tenant-id=[51d8c90f-bd95-41eb-a73d-fb725b84cbb6]
x-tenant-name=[default]
titlefromidprofile=[Regional Manager]
customheader=[customValue]
cookie=[]
x-api-
key=[0BX9mtir0+W8C3r4UqSL39POSo1BO35LYdCVf7Dp9WCH7m9mu4abrpnJs5A8AknrYwejLHcq9S7DNYj5Z8FKlpxFkHONl3e2wEjUiJ8Pf3UyJwrTXv
content-length=[2665]
host=[ssp-sample-ssp-sample-app-spi-demo.dev-sample.svc:443]
connection=[Keep-Alive]
user-agent=[Apache-HttpClient/4.5.13 (Java/11.0.11)]
accept-encoding=[gzip,deflate]

SPI Response from the custom SPI

Notice that the Sample Demo custom SPI sends back a few headers and cookies instructing IDSP to proxy those back to the client.

2022-04-16 21:03:18.774 INFO 1 --- [nio-8092-exec-3] i.s.a.CustomAuthenticatorApiController   :
----- Custom Auth Sample Provider Response v1.0.0000 ----- ************************************** RESPONSE

```
status=accept
msg=Received response: 'accept' for Flow Id: 'initial'
----- clientRespContext -----
----- operationalContext -----
spi.response.header.proxy=header-by-spi-expect2parts
spi.response.cookie.proxy=cookie-by-spi
spi.auth.factor.flowid=initial.accept
----- credData -----
spi.auth.factor.response=accept
----- headers -----
header-by-spi-expect2parts=[sample-header-by-spi-part1, sample-header-by-spi-part2]
CustomResponseHeader=[sample-header-by-provider]
RandomResponseHeader=[sample-header-by-spi-not-requested]
----- cookies -----
cookie-by-spi=sample-cookie-by-spi; Domain=example.com; Path=/default; Secure; HttpOnly
CustomResponseCookie=sample-cookie-by-provider; Domain=example.com; Path=/default; Secure; HttpOnly
RandomResponseCookie=sample-cookie-by-spi-not-requested; Domain=example.com; Path=/default; Secure; HttpOnly
```

API Response


```
{
   "flowState":
 "TWJZZzkzNW12ZGwwOHRhK3BFL2dDTXR5dDY4MTFVZmxqTWNZU0hCblpkamcxUzFPT21lSkRnQzRhQWpMaWg4S2pFNFNzSHVnOWxrVCtMV3RGY2t2UDBle
   "userName": "MalBAIL",
   "nextaction": "AUTH_ALLOWED",
   "data": {
       "status": "accept",
       "clientRespContext": [],
       "credData": [
           {
               "key": "spi.auth.factor.response",
               "value": "Sample Custom Auth SPI: credentials validated successfully"
           }
       ]
   },
   "id_token":
 "eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0U
eT4y_Zr8L-shklIlsy7dpGgkVxHucBWeBxSqnb-
IMLpHUnBDYkOqBiw7AgWX208tP0VwfvJvp6t9QdcBNZlMOwMZba8xQf1AAvZXjPQD5MjhHgG4I4VELfqlWy5_Hp0RrQ-
XD44kjNuMSVUxum_hWmOPUNTF6cMN1GG9OZT5UHndfqDbODiYaIJQdbDvRnhSl34Ouk6aEEnNIMVEbV4RfbSHPHKe1f1ZTw6bbermFkgAxg",
   "additional": {
       "authLevelsData": {
           "7bd13001-f024-45cf-92c2-d986381694a9+MalBAIL+1": "CUSTOM/2db88c50-93a9-4618-ad64-23b9291811cb"
       },
       "currentFactorLevel": 1,
       "clientRespContext": [
           {
               "key": "deviceId",
               "value": "post risk success"
           }
       ]
   }
}
```

As per instructions from the SPI, the API is given a header and cookie:

```
header-by-spi-expect2parts : [sample-header-by-spi-part1, sample-header-by-spi-part2]
Set-Cookie : cookie-by-spi=sample-cookie-by-spi; Domain=example.com; Path=/default; Secure; HttpOnly
```

ID Token


```
{
 "sub": "fc3108d9-86ce-4829-a034-553a7375dd6d",
 "idp_name": "TestConfig",
 "user_dn": "cn=MalBAIL,ou=Administration,ou=Corporate,o=democorp,c=us",
 "amr": [
   "PWD"
 ],
 "user_loginid": "malbail",
 "idp_guid": "d15e9718-404d-4345-9198-1625d093c079",
 "iss": "https://example.com/default/",
 "user_guid": "fc3108d9-86ce-4829-a034-553a7375dd6d",
 "aal": "AAL1",
 "tid": "51d8c90f-bd95-41eb-a73d-fb725b84cbb6",
 "sid": "ce24cb09-80d6-4270-bcda-b385a56631dd",
 "acr": "urn:iam:acr:aal1",
 "azp": "1bcea78c-9f11-467a-9dba-ccb0a660092b",
 "user_loginidattrmappingname": "user_loginid",
 "auth_time": 1650142998,
 "idp_user_guid": "MalBAIL",
 "exp": 1650164598,
 "iat": 1650142998,
 "client_name": "DefaultTenantClient",
 "jti": "09c49f2a-2903-4dd8-8fee-6e9bbe285a4a",
 "email": "",
 "idp_type": "ldap",
 "client_tname": "default",
 "client_tid": "51d8c90f-bd95-41eb-a73d-fb725b84cbb6",
 "ver": "1.0",
 "user_universalid": "malbail",
 "tname": "default",
 "tok_type": "IT",
 "user_riskscore": 60,
 "aud": [
   "1bcea78c-9f11-467a-9dba-ccb0a660092b",
   "https://example.com/default/"
 ],
 "nbf": 1650142998,
 "user_zfp": false,
 "client_transactionid": "ssp:28b77af8-40b0-4656-8cd3-9fa3231bb138",
 "tok_source": "FLS"
}
```


Step 6: Exchange Identity Token for Access Token

Request

```
curl --location --request POST 'https://example.com/default/oauth2/v1/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
```

```
--header 'Authorization: Basic
 MWJjZWE3OGMtOWYxMS00NjdhLTlkYmEtY2NiMGE2NjAwOTJiOjI4NDFlZDY4LTZjMTEtNGY0NC1hMDk4LTUwN2ZhZGNmZWIzMQ==' \
--data-urlencode 'grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer' \
--data-urlencode
 'assertion=eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVg
eT4y_Zr8L-shklIlsy7dpGgkVxHucBWeBxSqnb-
IMLpHUnBDYkOqBiw7AgWX208tP0VwfvJvp6t9QdcBNZlMOwMZba8xQf1AAvZXjPQD5MjhHgG4I4VELfqlWy5_Hp0RrQ-
XD44kjNuMSVUxum_hWmOPUNTF6cMN1GG9OZT5UHndfqDbODiYaIJQdbDvRnhSl34Ouk6aEEnNIMVEbV4RfbSHPHKe1f1ZTw6bbermFkgAxg' \
--data-urlencode 'scope=urn:iam:myscopes'
```

Response

```
{
   "access_token":
 "eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0U
keX96AoX5FwneZ0wlZzOhFEq_v6b0mMaXZhk9dFAdGMuShFJyYdnMVh26atpCnAYyi5BXGGk_MJrT-
jvfdDwa2Wubzej5Y6BQJVo2S3_KmcNV27aNI9P5TuLPC_MZkoPU4qH391wl8do2zfizCYNcfxWNSHGh9SlyofDYOK6ZNM9lqVOEJl4CIA",
   "token_type": "Bearer",
   "expires_in": 3600
}
```

Access Token

```
{
 "sub": "fc3108d9-86ce-4829-a034-553a7375dd6d",
 "idp_name": "TestConfig",
 "user_loginid": "malbail",
 "idp_guid": "d15e9718-404d-4345-9198-1625d093c079",
 "iss": "https://example.com/default/",
 "user_guid": "fc3108d9-86ce-4829-a034-553a7375dd6d",
 "tid": "51d8c90f-bd95-41eb-a73d-fb725b84cbb6",
 "sid": "2a250782-f195-4865-84b6-96f9c9bf6898",
 "sub_type": "USER",
 "azp": "1bcea78c-9f11-467a-9dba-ccb0a660092b",
 "user_loginidattrmappingname": "user_loginid",
 "scope": "urn:iam:m.me urn:iam:t.all",
 "exp": 1650145593,
 "iat": 1650141993,
 "client_name": "DefaultTenantClient",
 "jti": "6b5ebcc4-a458-476b-81a9-e8f763988964",
 "idp_type": "ldap",
 "client_tname": "default",
 "client_tid": "51d8c90f-bd95-41eb-a73d-fb725b84cbb6",
 "ver": "1.0",
 "user_universalid": "malbail",
 "azpacr": "1",
 "tname": "default",
 "tok_type": "AT",
 "user_riskscore": 91,
 "user_amr": [
   "PWD"
 ],
 "aud": "https://example.com/default/",
 "nbf": 1650141993,
```

```
 "user_zfp": false,
 "client_transactionid": "ssp:e621154a-f2bd-4877-9ab9-98697393329b",
 "user_aal": "AAL1",
 "tok_source": "JBR",
 "user_acr": "urn:iam:acr:aal1"
}
```

Example 2: Multi-factor flow with different custom SPIs to complete the OpenID Connect authentication flow

This example demonstrates use of custom SPI to implement multi-step authentication flow using an out of the box Sample Demo custom SPI deployed by the Sample App chart to the same cluster running IDSP. This example extends AZ Code flow under "OAuth and OpenId Connect" postman section to leverage custom SPI authentication using a more complete flow. Unlike the previous example, this example does not show the creation of entities that are required to support the runtime flow, but rather shows those entities as already existing.

Step 1: Create an application

Notice the use of custom claims and metadata.

```
{
   "status": "active",
   "name": "DemoClient",
   "description": "Demo client to perform tenant level operations for demo purpose.",
   "clientType": "TRUSTED",
   "allowedOpenIDScopes": [
       "address",
       "phone",
       "openid",
       "profile",
       "offline_access",
       "groups",
       "email"
   ],
   "redirectURIs": [
       "https://www.example.com"
   ],
   "allowedGrantTypes": [
       "refresh_token",
       "implicit",
       "client_credentials",
       "urn:ietf:params:oauth:grant-type:device_code",
       "authorization_code",
       "sts_confidential",
       "urn:ietf:params:oauth:grant-type:jwt-bearer"
   ],
   "deviceCodeFlowURL": null,
   "userInfoEndpointResponseFormat": "PLAIN_JSON",
   "skipIssuerAudienceForIT": false,
   "skipEmailForIT": false,
   "zeroFootPrint": false,
   "softMFAEnabled": false,
   "userTokenSubAttributeMappingName": "email",
   "supportedJoseHeaderParams": null,
   "claims": [
```

```
        {
            "name": "_titleIdStore",
            "target": "at,it",
            "value": "${idProfile.title}"
        },
        {
            "name": "_acctNumStatic",
            "target": "at,it",
            "value": "12345"
        },
        {
            "name": "_orgAppMd",
            "target": "at,it",
            "value": "${appMetadata.org}"
        },
        {
            "name": "_tenantIdToken",
            "target": "at,it",
            "value": "${idToken.tid}"
        },
        {
            "name": "_tags",
            "target": "at,it",
            "value": "tenantIdToken=${idToken.tid}, titleIdStore=${idProfile.title}, orgAppMd=
${appMetadata.org}"
        }
    ],
    "metadata": [
        {
            "name": "env",
            "value": "test"
        },
        {
            "name": "org",
            "value": "123"
        },
        {
            "name": "region",
            "value": "us-east-1"
        }
    ],
    "allowedOperations": [
        "itgroups",
        "introspect",
        "atgroups"
    ],
    "secondaryAudiences": null,
    "assertionVerificationCertAlias": null,
    "appIcon": null,
    "secret": "cd73aa53-6d3a-460a-9518-62169cf6f065",
    "clientId": "a940539a-dd10-4e21-b765-e9b54dc2037d",
    "itEncryptionTarget": "NONE",
    "itEncryptionCertAlias": null,
```

```
    "userInfoEncryptionCertAlias": null,
    "samlEntityId": null,
    "samlAcsUrl": null,
    "samlEnableSingleLogout": false,
    "samlNameIdFormat": null,
    "samlVerifyRequestSignature": false,
    "samlVerifyCertAlias": null,
    "samlEncryptSamlResponse": false,
    "samlEncryptCertAlias": null,
    "samlIdpInitiatedRelaystateMapping": {},
    "skewTimeSecs": 0,
    "passwordAuthoritativeSource": "remote",
    "mitmProtectionLevel": null,
    "appId": "55434a3c-f1b3-4b51-9696-6db7556aaf11",
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "1bcea78c-9f11-467a-9dba-ccb0a660092b",
        "principalName": "DefaultTenantClient"
    },
    "createdDateTime": "2022-03-06 09:05 AM UTC",
    "updatedDateTime": "2022-03-31 09:32 PM UTC"
}
```


Step 2: Create an authentication policy

Notice use of custom factors in obligations for the policy condition with the id "995bb16a-de68-42e9-b18e-b6d0116e80e8" : "obligations": ["CUSTOM/customSampleProvider3:3", "CUSTOM/customSampleProvider2:2", "CUSTOM/customSampleProvider:2", "PASSWORD:1"]


```
{
    "policyId": "dbdfb24e-b432-48bd-8a34-597eb76ac1d4",
    "policyName": "Feature Policy for Demo client",
    "description": "Feature Policy Demo client",
    "status": "active",
    "matchAnyApp": false,
    "apps": [
        {
            "name": "DemoClient",
            "id": "55434a3c-f1b3-4b51-9696-6db7556aaf11"
        }
    ],
    "rules": [
        {
            "id": "493e98ed-ea4c-41ad-ad68-6654b72c2826",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "Finance"
                        ]
                    }
                },
                "action": {
```

```
                    "operator": "in",
                    "value": [
                        "authenticate"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD:1",
                    "FIDO:2",
                    "SECURITYKEY:2"
                ],
                "msg": "This condition is for the Finance Group PW+FIDO or PW+SECURITYKEY on the Web Channels.
Login with bsteer."
            }
        },
        {
            "id": "51c3d6d3-2dd7-4646-907c-3183bd68787c",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "Employees",
                            "Human Resources",
                            "Finance",
                            "Development",
                            "Customers"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "authenticate"
                    ]
                },
                "riskThreshold": {
                    "operator": "ge",
                    "value": 60
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
```

```
                "obligations": [
                    "PASSWORD:1",
                    "EMAILOTP:2",
                    "SMSOTP:2"
                ],
                "msg": "This condition is for Risk > 60 on mobile and web channels. Login with nbruce, swise,
 or vburn with untrusted IP."
            }
        },
        {
            "id": "2a866359-3308-449e-96b8-73b730698566",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "Human Resources"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "authenticate"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web",
                        "mobile"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD:1",
                    "EMAILOTP:2",
                    "SMSOTP:2",
                    "FIDO:2",
                    "SECURITYKEY:2"
                ],
                "msg": "This condition is for the Human Resources Group with PW+OTP or PW+FIDO or PW
+SECURITYKEY on the Web and Mobile Channels. Login with edunn for autoenrol and erussell for inline enrol."
            }
        },
        {
            "id": "4826238a-f819-48c4-bfb1-8b2b5875e5ad",
            "conditions": {
                "action": {
                    "operator": "in",
```

```
                    "value": [
                        "wealth"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "MOBILEOTP:2"
                ],
                "msg": "This condition is for testing Continuous authentication on the Web Channel. Login with
nbruce and Select Wealth Management to invoke this Condition."
            }
        },
        {
            "id": "995bb16a-de68-42e9-b18e-b6d0116e80e8",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "Security Administrators",
                            "Employees"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "authenticate"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web"
                    ]
                },
                "acr": {
                    "operator": "in",
                    "value": [
                        "urn:iam:acr:aal:aal1"
                    ]
                }
            },
            "result": {
```

```
            "effect": "allow",
            "reAuthenticate": "false",
            "obligations": [
                "CUSTOM/ea7ae57b-104c-45fb-91fe-51c29ff56596:3",
                "CUSTOM/2539cd04-02f1-453e-ab20-8b1b22f84f28:2",
                "CUSTOM/29fdfd06-e5ad-425f-b108-f681ec108b8d:2",
                "PASSWORD:1"
            ],
            "msg": "This condition is for the Employees Group with Password Only. Login with nbruce"
        }
    },
    {
        "id": "db633d60-e792-46b2-a9db-ec8df2b875f0",
        "conditions": {
            "principal": {
                "group": {
                    "operator": "in",
                    "value": [
                        "Development"
                    ]
                }
            },
            "action": {
                "operator": "in",
                "value": [
                    "authenticate"
                ]
            },
            "channel": {
                "operator": "in",
                "value": [
                    "web"
                ]
            }
        },
        "result": {
            "effect": "allow",
            "reAuthenticate": "false",
            "obligations": [
                "FIDO:1",
                "SECURITYKEY:1"
            ],
            "msg": "This condition is Passwordless for the Development Group. Login with cmoran."
        }
    },
    {
        "id": "659b82ed-1ce4-4ec4-86b7-ebd9e0acc046",
        "conditions": {
            "principal": {
                "group": {
                    "operator": "in",
                    "value": [
                        "Customers"
```

```
                    ]
                }
            },
            "action": {
                "operator": "in",
                "value": [
                    "authenticate"
                ]
            },
            "channel": {
                "operator": "in",
                "value": [
                    "web"
                ]
            }
        },
        "result": {
            "effect": "allow",
            "reAuthenticate": "false",
            "obligations": [
                "PASSWORD:1",
                "FIDO:1",
                "SECURITYKEY:1",
                "SMSOTP:2",
                "EMAILOTP:2",
                "MOBILEOTP:2",
                "PUSH:2"
            ],
            "msg": "This condition is PW + OTP/MOBILEOTP/PUSH or FIDO + OTP/MOBILEOTP/PUSH or SECURITYKEY +
OTP/MOBILEOTP/PUSHwith Choices for the Customers Group on the Web Channel. Login with droth."
        }
    },
    {
        "id": "a6afbd6f-f251-4c2b-84a4-50c21f22ed83",
        "conditions": {
            "action": {
                "operator": "in",
                "value": [
                    "investment",
                    "business"
                ]
            },
            "channel": {
                "operator": "in",
                "value": [
                    "web",
                    "mobile"
                ]
            }
        },
        "result": {
            "effect": "allow",
            "reAuthenticate": "false",
```

```
            "obligations": [
                "SMSOTP:2",
                "EMAILOTP:2"
            ],
            "msg": "This condition is for testing Continuous authentication on the Web Channel. Login with
nbruce and Select Business Banking or Details to invoke this Condition."
        }
    },
    {
        "id": "196377ea-7e01-4183-bfd6-ba4dd8606da3",
        "conditions": {
            "principal": {
                "group": {
                    "operator": "in",
                    "value": [
                        "Employees"
                    ]
                }
            },
            "action": {
                "operator": "in",
                "value": [
                    "authenticate"
                ]
            },
            "channel": {
                "operator": "in",
                "value": [
                    "mobile"
                ]
            }
        },
        "result": {
            "effect": "allow",
            "reAuthenticate": "false",
            "obligations": [
                "PASSWORD:1",
                "FIDO:2",
                "SECURITYKEY:2"
            ],
            "msg": "This condition is for the Employees Group PW+FIDO+SECURITYKEY on the Mobile Channel.
Login with nbruce"
        }
    },
    {
        "id": "ead8794e-2e23-4cdf-a955-a5e746d18b68",
        "conditions": {
            "action": {
                "operator": "in",
                "value": [
                    "commercial"
                ]
            },
```

```
                "channel": {
                    "operator": "in",
                    "value": [
                        "web",
                        "mobile"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "FIDO:2",
                    "SECURITYKEY:2"
                ],
                "msg": "This condition is for testing Continuous authentication on the Web Channel. Login with
 nbruce and Select Commercial Banking to invoke this Condition."
            }
        }
    ],
    "createdDate": "2022-03-06 09:05 AM UTC",
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "1bcea78c-9f11-467a-9dba-ccb0a660092b",
        "principalName": "DefaultTenantClient"
    },
    "updatedDate": "2022-04-08 04:32 PM UTC",
    "preventedOperations": "Delete"
}
```


Step 3: Create a custom SPI

Notice the use of directives in ProviderProperties in configurations for both "customSampleProvider2" and "customSampleProvider3", and correlate the use of those directives to runtime flows.


```
{
    "providerName": "customSampleProvider",
    "version": "1.0",
    "apiKey": "uSx8ixLc7OFxmWDfzSgJppKTwz+HckRC8AM+ei2hnAr2JyRHjLdrXPfA2/
RCAGGBbyFykMWkR0Uqf6KJk8b9IXvqTGa2Gecn4xlBnpiqpsevCE4X1q7pG
+obyPlHNlrlbfamMrP/1PD5eBcwGZI5jazihze39z4HuooAKMzMapi0Jg7t/Uv4YxTlIstX9EfKNixLT8ZNpzzA0T9jLjonfQ==",
    "certAlias": "",
    "url": "https://ssp-sample-ssp-sample-app-spi-demo.dev-sample.svc:443",
    "providerHeaders": [
        {
            "name": "tagsExampleHeader",
            "value": "titleIdProfile=${idProfile.title}, userRiskScoreIdToken=${idToken.user_riskscore},
 orgAppMd=${appMetadata.org}, numStatic=12345"
        },
        {
            "name": "CustomProvider",
            "value": "CustomSampleProvider"
        }
    ],
```

```
    "providerProperties": [
        {
            "name": "spi.discover.capabilities",
            "value": "customauthenticator"
        },
        {
            "name": "tagsExampleContext",
            "value": "${idProfile.groups}, titleIdProfile=${idProfile.title}, userRiskScoreIdToken=
${idToken.user_riskscore}, orgAppMd=${appMetadata.org}, numStatic=12345"
        },
        {
            "name": "spi.auth.factor.amr",
            "value": "FIDO, OTP"
        },
        {
            "name": "spi.auth.factor.flowid",
            "value": "flowId2-0"
        },
        {
            "name": "spi.request.header.proxy",
            "value": "user-agent"
        },
        {
            "name": "spi.request.cookie.proxy",
            "value": "CustomRequestCookie"
        },
        {
            "name": "spi.response.header.proxy",
            "value": "CustomResponseHeader"
        },
        {
            "name": "spi.response.cookie.proxy",
            "value": "CustomResponseCookie"
        }
    ],
    "origin": "tenant",
    "ignoreSSLValidation": true,
    "providerId": "29fdfd06-e5ad-425f-b108-f681ec108b8d",
    "sslMode": "VERIFY_CA",
    "spiReadTimeoutMillis": "60000",
    "spiConnectTimeoutMillis": "60000",
    "requestCtxBase64": "",
    "clientId": "",
    "clientSecretAlias": "",
    "oauthScopes": "",
    "securityMethod": "apikey",
    "oauthTokenUrl": "",
    "oauthDiscoveryUrl": "",
    "oauthResourceInAzRequest": "none",
    "oauthAssertionKidType": "",
    "oauthAssertionAud": "",
    "oauthTokenIgnoreSslValidation": false,
    "oauthTokenSslMode": "VERIFY_CA",
```

```
   "oauthTokenCertificateAlias": "",
   "oauthTokenRequestCtxBase64": "",
   "oauthTokenReadTimeoutMillis": "10000",
   "oauthTokenConnectTimeoutMillis": "10000",
   "skewTimeSecs": "0"
},
{
   "providerName": "customSampleProvider2",
   "version": "1.0",
   "apiKey": "uSx8ixLc7OFxmWDfzSgJppKTwz+HckRC8AM+ei2hnAr2JyRHjLdrXPfA2/
RCAGGBbyFykMWkR0Uqf6KJk8b9IXvqTGa2Gecn4xlBnpiqpsevCE4X1q7pG
+obyPlHNlrlbfamMrP/1PD5eBcwGZI5jazihze39z4HuooAKMzMapi0Jg7t/Uv4YxTlIstX9EfKNixLT8ZNpzzA0T9jLjonfQ==",
   "certAlias": "",
   "url": "https://ssp-sample-ssp-sample-app-spi-demo.dev-sample.svc:443",
   "providerHeaders": [
       {
           "name": "tagsExampleHeader",
           "value": "titleIdProfile=${idProfile.title}, userRiskScoreIdToken=${idToken.user_riskscore},
 orgAppMd=${appMetadata.org}, numStatic=12345"
       },
       {
           "name": "CustomProvider",
           "value": "CustomSampleProvider2"
       }
   ],
   "providerProperties": [
       {
           "name": "spi.discover.capabilities",
           "value": "customauthenticator"
       },
       {
           "name": "tagsExampleContext",
           "value": "${idProfile.groups}, titleIdProfile=${idProfile.title}, userRiskScoreIdToken=
${idToken.user_riskscore}, orgAppMd=${appMetadata.org}, numStatic=12345"
       },
       {
           "name": "spi.auth.factor.amr",
           "value": "FIDO, OTP"
       },
       {
           "name": "spi.auth.factor.flowid",
           "value": "flowId2-0"
       },
       {
           "name": "spi.request.header.proxy",
           "value": "user-agent"
       },
       {
           "name": "spi.request.cookie.proxy",
           "value": "CustomRequestCookie"
       },
       {
           "name": "spi.response.header.proxy",
```

```
            "value": "CustomResponseHeader"
        },
        {
            "name": "spi.response.cookie.proxy",
            "value": "CustomResponseCookie"
        }
    ],
    "origin": "tenant",
    "ignoreSSLValidation": true,
    "providerId": "2539cd04-02f1-453e-ab20-8b1b22f84f28",
    "sslMode": "VERIFY_CA",
    "spiReadTimeoutMillis": "60000",
    "spiConnectTimeoutMillis": "60000",
    "requestCtxBase64": "",
    "clientId": "",
    "clientSecretAlias": "",
    "oauthScopes": "",
    "securityMethod": "apikey",
    "oauthTokenUrl": "",
    "oauthDiscoveryUrl": "",
    "oauthResourceInAzRequest": "none",
    "oauthAssertionKidType": "",
    "oauthAssertionAud": "",
    "oauthTokenIgnoreSslValidation": false,
    "oauthTokenSslMode": "VERIFY_CA",
    "oauthTokenCertificateAlias": "",
    "oauthTokenRequestCtxBase64": "",
    "oauthTokenReadTimeoutMillis": "10000",
    "oauthTokenConnectTimeoutMillis": "10000",
    "skewTimeSecs": "0"
},
{
    "providerName": "customSampleProvider3",
    "version": "1.0",
    "apiKey": "uSx8ixLc7OFxmWDfzSgJppKTwz+HckRC8AM+ei2hnAr2JyRHjLdrXPfA2/
RCAGGBbyFykMWkR0Uqf6KJk8b9IXvqTGa2Gecn4xlBnpiqpsevCE4X1q7pG
+obyPlHNlrlbfamMrP/1PD5eBcwGZI5jazihze39z4HuooAKMzMapi0Jg7t/Uv4YxTlIstX9EfKNixLT8ZNpzzA0T9jLjonfQ==",
    "certAlias": "",
    "url": "https://ssp-sample-ssp-sample-app-spi-demo.dev-sample.svc:443",
    "providerHeaders": [
        {
            "name": "tagsExampleHeader",
            "value": "titleIdProfile=${idProfile.title}, userRiskScoreIdToken=${idToken.user_riskscore},
 orgAppMd=${appMetadata.org}, numStatic=12345"
        },
        {
            "name": "CustomProvider",
            "value": "CustomSampleProvider3"
        }
    ],
    "providerProperties": [
        {
            "name": "spi.discover.capabilities",
```

```
            "value": "customauthenticator"
        },
        {
            "name": "tagsExampleContext",
            "value": "${idProfile.groups}, titleIdProfile=${idProfile.title}, userRiskScoreIdToken=
${idToken.user_riskscore}, orgAppMd=${appMetadata.org}, numStatic=12345"
        },
        {
            "name": "spi.auth.factor.amr",
            "value": "FIDO, OTP"
        },
        {
            "name": "spi.auth.factor.flowid",
            "value": "flowId3-0"
        },
        {
            "name": "spi.request.header.proxy",
            "value": "user-agent"
        },
        {
            "name": "spi.request.cookie.proxy",
            "value": "CustomRequestCookie"
        },
        {
            "name": "spi.response.header.proxy",
            "value": "CustomResponseHeader"
        },
        {
            "name": "spi.response.cookie.proxy",
            "value": "CustomResponseCookie"
        }
    ],
    "origin": "tenant",
    "ignoreSSLValidation": true,
    "providerId": "ea7ae57b-104c-45fb-91fe-51c29ff56596",
    "sslMode": "VERIFY_CA",
    "spiReadTimeoutMillis": "60000",
    "spiConnectTimeoutMillis": "60000",
    "requestCtxBase64": "",
    "clientId": "",
    "clientSecretAlias": "",
    "oauthScopes": "",
    "securityMethod": "apikey",
    "oauthTokenUrl": "",
    "oauthDiscoveryUrl": "",
    "oauthResourceInAzRequest": "none",
    "oauthAssertionKidType": "",
    "oauthAssertionAud": "",
    "oauthTokenIgnoreSslValidation": false,
    "oauthTokenSslMode": "VERIFY_CA",
    "oauthTokenCertificateAlias": "",
    "oauthTokenRequestCtxBase64": "",
    "oauthTokenReadTimeoutMillis": "10000",
```

```
   "oauthTokenConnectTimeoutMillis": "10000",
   "skewTimeSecs": "0"
}
```

Step 4: Relying Party initiates OpenID Connect flow

Request

```
curl --location --request GET 'https://example.com/default/oauth2/v1/authorize?scope=openid
 offline_access&client_id=a940539a-dd10-4e21-b765-e9b54dc2037d&response_type=code&redirect_uri=https://
www.example.com' \
--header 'X-CLIENT-TRANSACTION-ID: MyClientTxnId' \
--header 'Cookie: cookie-by-spi=sample-cookie-by-spi'
```

Response

```
Redirect to Location: https://example.com/default/ui/v1/signin/?x-flow-
state=T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3M
service-baseurl=https://example.com/default
```

Step 5: SignIn Client acquires Access Token

Request

```
curl --location --request POST 'https://example.com/default/oauth2/v1/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Authorization: Basic
 YTk0MDUzOWEtZGQxMC00ZTIxLWI3NjUtZTliNTRkYzIwMzdkOmNkNzNhYTUzLTZkM2EtNDYwYS05NTE4LTYyMTY5Y2Y2ZjA2NQ==' \
--header 'Cookie: cookie-by-spi=sample-cookie-by-spi' \
--data-urlencode 'grant_type=client_credentials' \
--data-urlencode 'scope=urn:iam:myscopes'
```

Response

```
{
   "access_token":
 "eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0U
wqHxEO4--DHhXmUpa-
H5ptkOnjGdw5Z8uQeHnlosG_SdaqoKSNAruHdIhwWFqMB2Yie73RNfo96L13NGAJQq9kFA24awzKei7yp01LFD0z086HtYoMKP3dDVF-
bNEBZRZZC4uE8grMUfmeqqw3uGq1QIFxcGtjeL3z7B_RaniLpVBIORGY6YZfpJz0XWMr_w1t41XaRb9-
PMxg4xwbNZ106SnamyOkDPyoCl_x_OxtCzuj_gII-KfqzmNyH2HE_yRwThaw",
   "token_type": "Bearer",
   "expires_in": 3600
}
```

Step 6: SignIn Client initiates authentication

Request

```
curl --location --request POST 'https://example.com/default/auth/v1/authenticate' \
--header 'Content-Type: application/json' \
--header 'x-flow-state:
 T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80NH
 \
--header 'Authorization: Bearer
 eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0UG
wqHxEO4--DHhXmUpa-
```

```
H5ptkOnjGdw5Z8uQeHnlosG_SdaqoKSNAruHdIhwWFqMB2Yie73RNfo96L13NGAJQq9kFA24awzKei7yp01LFD0z086HtYoMKP3dDVF-
bNEBZRZZC4uE8grMUfmeqqw3uGq1QIFxcGtjeL3z7B_RaniLpVBIORGY6YZfpJz0XWMr_w1t41XaRb9-
PMxg4xwbNZ106SnamyOkDPyoCl_x_OxtCzuj_gII-KfqzmNyH2HE_yRwThaw' \
--header 'Cookie: cookie-by-spi=sample-cookie-by-spi' \
--data-raw '{
   "subject": "mhunter",
   "action": "authenticate",
   "ipAddress": "1.1.1.2",
   "channel": "web",
   "acrValues" : ["urn:iam:acr:aal:aal1"],
   "clientContext": [
       {
           "key": "clientCustomData1",
           "value": "gga323a24"
       }
]
}'
```

Response

```
{
   "flowState":
 "T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80N
   "nextaction": "PASSWORD_AUTH",
   "currentFactors": [
       "PASSWORD"
   ],
   "additional": {
       "userName": "mhunter",
       "currentFactorLevel": 1
   }
}
```

Step 7: SignIn Client authenticates using ID store password

Request

```
curl --location --request POST 'https://example.com/default/factor/v1/PasswordAuthenticator' \
--header 'Content-Type: application/json' \
--header 'x-flow-state:
 T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80NH
 \
--header 'Authorization: Bearer
 eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0UG
wqHxEO4--DHhXmUpa-
H5ptkOnjGdw5Z8uQeHnlosG_SdaqoKSNAruHdIhwWFqMB2Yie73RNfo96L13NGAJQq9kFA24awzKei7yp01LFD0z086HtYoMKP3dDVF-
bNEBZRZZC4uE8grMUfmeqqw3uGq1QIFxcGtjeL3z7B_RaniLpVBIORGY6YZfpJz0XWMr_w1t41XaRb9-
PMxg4xwbNZ106SnamyOkDPyoCl_x_OxtCzuj_gII-KfqzmNyH2HE_yRwThaw' \
--header 'Cookie: cookie-by-spi=sample-cookie-by-spi' \
--data-raw '{"password":"password"}'
```

Response

```
{
"flowState":
 "T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80N
```

```
"userName": "mhunter",
"nextaction": "FACTOR_SELECTION",
"currentFactors": [
"CUSTOM/customSample2539cd04-02f1-453e-ab20-8b1b22f84f28Provider2",
"CUSTOM/29fdfd06-e5ad-425f-b108-f681ec108b8d"
],
"data": {
"message": "Password authentication is successful for user mhunter"
},
"additional": {
"currentFactorLevel": 2
}
}
```


Step 8: SignIn Client selects Next Factor

Request


```
curl --location --request POST 'https://example.com/default/auth/v1/SelectedFactor' \
--header 'X-Flow-State:
 T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80NH
 \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer
 eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0UG
wqHxEO4--DHhXmUpa-
H5ptkOnjGdw5Z8uQeHnlosG_SdaqoKSNAruHdIhwWFqMB2Yie73RNfo96L13NGAJQq9kFA24awzKei7yp01LFD0z086HtYoMKP3dDVF-
bNEBZRZZC4uE8grMUfmeqqw3uGq1QIFxcGtjeL3z7B_RaniLpVBIORGY6YZfpJz0XWMr_w1t41XaRb9-
PMxg4xwbNZ106SnamyOkDPyoCl_x_OxtCzuj_gII-KfqzmNyH2HE_yRwThaw' \
--header 'Cookie: cookie-by-spi=sample-cookie-by-spi' \
--data-raw '{
"factor": "CUSTOM/2539cd04-02f1-453e-ab20-8b1b22f84f28"
}'
```


Response


```
{
"flowState":
 "T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80N
"nextaction": "CUSTOM_AUTH",
"currentFactors": [
"CUSTOM/2539cd04-02f1-453e-ab20-8b1b22f84f28",
"CUSTOM/29fdfd06-e5ad-425f-b108-f681ec108b8d"
],
"additional": {
"currentFactorLevel": 2,
"isCustom": true,
"isCustomFirstTime": true,
"customProvider": "2539cd04-02f1-453e-ab20-8b1b22f84f28",
"customFlowId": "flowId2-0"
}
}
```


Step 9: SignIn Client orchestrates interaction with "CustomSampleProvider2"

9.1 "First Time" API Request

```
curl --location --request POST 'https://example.com/default/factor/v1/CustomAuthenticator' \
--header 'Content-Type: application/json' \
--header 'X-Flow-State:
 T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80NH
 \
--header 'x-forwarded-for: 1.2.3.4' \
--header 'Authorization: Bearer
 eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0UG
wqHxEO4--DHhXmUpa-
H5ptkOnjGdw5Z8uQeHnlosG_SdaqoKSNAruHdIhwWFqMB2Yie73RNfo96L13NGAJQq9kFA24awzKei7yp01LFD0z086HtYoMKP3dDVF-
bNEBZRZZC4uE8grMUfmeqqw3uGq1QIFxcGtjeL3z7B_RaniLpVBIORGY6YZfpJz0XWMr_w1t41XaRb9-
PMxg4xwbNZ106SnamyOkDPyoCl_x_OxtCzuj_gII-KfqzmNyH2HE_yRwThaw' \
--header 'Cookie: cookie-by-spi=sample-cookie-by-spi' \
--data-raw '{
"operationalContext": [
{
"key": "spi.auth.factor.flowid",
"value": "flowId2-0.FirstTime"
},
{
"key": "spi.auth.factor.firsttime",
"value": "true"
}
],
"credData": [
]
}'
```

9.1.1 SPI Request

```
2022-04-16 22:08:06.586 INFO 1 --- [nio-8092-exec-5] i.s.a.CustomAuthenticatorApiController :
----- Custom Auth Sample Provider Request v1.0.0000 ----- ****************************************** REQUEST
subject=mhunter
----- clientContext -----
clientCustomData1=gga323a24
----- ahContext -----
tid=51d8c90f-bd95-41eb-a73d-fb725b84cbb6
tname=default
clientIp=10.230.9.183
userIp=1.1.1.2
txnId=76e72ee8-a659-49a6-af96-046220ede29c
clientTxnId=MyClientTxnId
locale=null
userId=5fa64f0b-e6e0-4479-9c5e-3d1f2be93834
userUniversalId=mhunter
idpUserGuid=mhunter
idpGuid=d15e9718-404d-4345-9198-1625d093c079
idpName=TestConfig
idpType=ldap
completedAmrNames=["PWD"]
clientSslCert=
x-ssp-base-uri=https://example.com
x-ssp-issuer-url=https://example.com/default
x-ssp-host=example.com
```

```
x-ssp-path=/default
providerName=customSampleProvider2
providerId=2539cd04-02f1-453e-ab20-8b1b22f84f28
----- providerContext -----
----- metadataContext -----
env=test
org=123
region=us-east-1
----- operationalContext -----
spi.auth.factor.flowid=flowId2-0
spi.auth.factor.firsttime=true
----- credData -----
----- headers -----
accept=[application/json]
content-type=[application/json]
x-client-transaction-id=[MyClientTxnId]
x-transaction-id=[76e72ee8-a659-49a6-af96-046220ede29c]
x-forwarded-for=[]
x-tenant-id=[51d8c90f-bd95-41eb-a73d-fb725b84cbb6]
x-tenant-name=[default]
customprovider=[CustomSampleProvider2]
user-agent=[PostmanRuntime/7.29.0]
cookie=[]
x-api-
key=[0BX9mtir0+W8C3r4UqSL39POSo1BO35LYdCVf7Dp9WCH7m9mu4abrpnJs5A8AknrYwejLHcq9S7DNYj5Z8FKlpxFkHONl3e2wEjUiJ8Pf3UyJwrTXv
content-length=[2717]
host=[ssp-sample-ssp-sample-app-spi-demo.dev-sample.svc:443]
connection=[Keep-Alive]
accept-encoding=[gzip,deflate]
```

9.1.1 SPI Response

```
2022-04-16 22:08:06.587 INFO 1 --- [nio-8092-exec-5] i.s.a.CustomAuthenticatorApiController :
----- Custom Auth Sample Provider Response v1.0.0000 ----- ************************************** RESPONSE
status=Challenge
msg=First time for Flow Id: flowId2-0
----- clientRespContext -----
clientCustomData1=gga323a24
----- operationalContext -----
spi.auth.factor.flowid=flowId2-0.FirstTime
----- credData -----
----- headers -----
----- cookies -----
```

9.1 "First Time" API Response

```
{
"flowState":
 "T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80N
"userName": "mhunter",
"nextaction": "CUSTOM_AUTH",
"currentFactors": [
"CUSTOM/2539cd04-02f1-453e-ab20-8b1b22f84f28",
"CUSTOM/customSampleProvider"
```

```
],
"data": {
"status": "Challenge",
"clientRespContext": [
{
"key": "clientCustomData1",
"value": "gga323a24"
}
],
"credData": [
{
"key": "spi.auth.factor.challenge",
"value": "Sample Custom Auth SPI: enter authentication status [Accept, Challenge, Reject, Terminate,
 Exception]:"
}
]
},
"additional": {
"currentFactorLevel": 2,
"isCustom": true,
"customProvider": "2539cd04-02f1-453e-ab20-8b1b22f84f28",
"customFlowId": "flowId2-0.FirstTime"
}
}
```


9.2 "Challenge" API Request


```
curl --location --request POST 'https://example.com/default/factor/v1/CustomAuthenticator' \
--header 'Content-Type: application/json' \
--header 'X-Flow-State:
 T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80NH
 \
--header 'x-forwarded-for: 1.2.3.4' \
--header 'Authorization: Bearer
 eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0UG
wqHxEO4--DHhXmUpa-
H5ptkOnjGdw5Z8uQeHnlosG_SdaqoKSNAruHdIhwWFqMB2Yie73RNfo96L13NGAJQq9kFA24awzKei7yp01LFD0z086HtYoMKP3dDVF-
bNEBZRZZC4uE8grMUfmeqqw3uGq1QIFxcGtjeL3z7B_RaniLpVBIORGY6YZfpJz0XWMr_w1t41XaRb9-
PMxg4xwbNZ106SnamyOkDPyoCl_x_OxtCzuj_gII-KfqzmNyH2HE_yRwThaw' \
--header 'Cookie: CustomResponseCookie=sample-cookie-by-provider; cookie-by-spi=sample-cookie-by-spi' \
--data-raw '{
"operationalContext": [
{
"key": "spi.auth.factor.flowid",
"value": "flowId2-0.FirstTime.challenge"
}
],
"credData": [
{
"key": "spi.auth.factor.response",
"value": "challenge"
}
]
}'
```

9.2.1 SPI Request

```
2022-04-16 22:11:37.965 INFO 1 --- [nio-8092-exec-1] i.s.a.CustomAuthenticatorApiController :
----- Custom Auth Sample Provider Request v1.0.0000 ----- ************************************** REQUEST
subject=mhunter
----- clientContext -----
clientCustomData1=gga323a24
----- ahContext -----
tid=51d8c90f-bd95-41eb-a73d-fb725b84cbb6
tname=default
clientIp=10.230.9.183
userIp=1.1.1.2
txnId=76e72ee8-a659-49a6-af96-046220ede29c
clientTxnId=MyClientTxnId
locale=null
userId=5fa64f0b-e6e0-4479-9c5e-3d1f2be93834
userUniversalId=mhunter
idpUserGuid=mhunter
idpGuid=d15e9718-404d-4345-9198-1625d093c079
idpName=TestConfig
idpType=ldap
completedAmrNames=["PWD"]
clientSslCert=
x-ssp-base-uri=https://example.com
x-ssp-issuer-url=https://example.com/default
x-ssp-host=example.com
x-ssp-path=/default
providerName=customSampleProvider2
providerId=2539cd04-02f1-453e-ab20-8b1b22f84f28
----- providerContext -----
----- metadataContext -----
env=test
org=123
region=us-east-1
----- operationalContext -----
spi.auth.factor.flowid=flowId2-0.FirstTime
----- credData -----
spi.auth.factor.response=challenge
----- headers -----
accept=[application/json]
content-type=[application/json]
x-client-transaction-id=[MyClientTxnId]
x-transaction-id=[76e72ee8-a659-49a6-af96-046220ede29c]
x-forwarded-for=[]
x-tenant-id=[51d8c90f-bd95-41eb-a73d-fb725b84cbb6]
x-tenant-name=[default]
customprovider=[CustomSampleProvider2]
user-agent=[PostmanRuntime/7.29.0]
cookie=[]
x-api-
key=[0BX9mtir0+W8C3r4UqSL39POSo1BO35LYdCVf7Dp9WCH7m9mu4abrpnJs5A8AknrYwejLHcq9S7DNYj5Z8FKlpxFkHONl3e2wEjUiJ8Pf3UyJwrTXv
content-length=[2730]
host=[ssp-sample-ssp-sample-app-spi-demo.dev-sample.svc:443]
```

```
connection=[Keep-Alive]
accept-encoding=[gzip,deflate]
```

9.2.1 SPI Response

```
2022-04-16 22:11:37.969 INFO 1 --- [nio-8092-exec-1] i.s.a.CustomAuthenticatorApiController :
----- Custom Auth Sample Provider Response v1.0.0000 ----- ************************************** RESPONSE
status=challenge
msg=Received response: 'challenge' for Flow Id: 'flowId2-0.FirstTime'
----- clientRespContext -----
clientCustomData1=gga323a24
----- operationalContext -----
spi.response.header.proxy=header-by-spi-expect2parts
spi.response.cookie.proxy=cookie-by-spi
spi.auth.factor.flowid=flowId2-0.FirstTime.challenge
----- credData -----
spi.auth.factor.response=challenge
----- headers -----
header-by-spi-expect2parts=[sample-header-by-spi-part1, sample-header-by-spi-part2]
CustomResponseHeader=[sample-header-by-provider]
RandomResponseHeader=[sample-header-by-spi-not-requested]
----- cookies -----
cookie-by-spi=sample-cookie-by-spi; Domain=example.com; Path=/default; Secure; HttpOnly
CustomResponseCookie=sample-cookie-by-provider; Domain=example.com; Path=/default; Secure; HttpOnly
RandomResponseCookie=sample-cookie-by-spi-not-requested; Domain=example.com; Path=/default; Secure; HttpOnly
```

9.2 "Challenge" API Response


```
{
"flowState":
 "T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80N
"userName": "mhunter",
"nextaction": "CUSTOM_AUTH",
"currentFactors": [
"CUSTOM/2539cd04-02f1-453e-ab20-8b1b22f84f28",
"CUSTOM/29fdfd06-e5ad-425f-b108-f681ec108b8d"
],
"data": {
"status": "challenge",
"clientRespContext": [
{
"key": "clientCustomData1",
"value": "gga323a24"
}
],
"credData": [
{
"key": "spi.auth.factor.challenge",
"value": "Sample Custom Auth SPI: enter authentication status [Accept, Challenge, Reject, Terminate,
 Exception] "
}
]
},
"additional": {
```

```
"currentFactorLevel": 2,
"isCustom": true,
"customProvider": "2539cd04-02f1-453e-ab20-8b1b22f84f28",
"customFlowId": "flowId2-0.FirstTime.challenge"
}
}
```

9.3 "Accept Response" API Request

```
curl --location --request POST 'https://example.com/default/factor/v1/CustomAuthenticator' \
--header 'Content-Type: application/json' \
--header 'X-Flow-State:
 T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80NH
 \
--header 'Authorization: Bearer
 eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0UG
wqHxEO4--DHhXmUpa-
H5ptkOnjGdw5Z8uQeHnlosG_SdaqoKSNAruHdIhwWFqMB2Yie73RNfo96L13NGAJQq9kFA24awzKei7yp01LFD0z086HtYoMKP3dDVF-
bNEBZRZZC4uE8grMUfmeqqw3uGq1QIFxcGtjeL3z7B_RaniLpVBIORGY6YZfpJz0XWMr_w1t41XaRb9-
PMxg4xwbNZ106SnamyOkDPyoCl_x_OxtCzuj_gII-KfqzmNyH2HE_yRwThaw' \
--header 'Cookie: CustomResponseCookie=sample-cookie-by-provider; cookie-by-spi=sample-cookie-by-spi' \
--data-raw '{
"operationalContext": [
{
"key": "spi.auth.factor.flowid",
"value": "flowId3-0"
}
],
"credData": [
{
"key": "spi.auth.factor.response",
"value": "accept"
}
]
}'
```

9.3.1 SPI Request

```
2022-04-16 22:13:05.874 INFO 1 --- [nio-8092-exec-8] i.s.a.CustomAuthenticatorApiController :
----- Custom Auth Sample Provider Request v1.0.0000 ----- ****************************************** REQUEST
subject=mhunter
----- clientContext -----
clientCustomData1=gga323a24
----- ahContext -----
tid=51d8c90f-bd95-41eb-a73d-fb725b84cbb6
tname=default
clientIp=10.230.9.183
userIp=1.1.1.2
txnId=76e72ee8-a659-49a6-af96-046220ede29c
clientTxnId=MyClientTxnId
locale=null
userId=5fa64f0b-e6e0-4479-9c5e-3d1f2be93834
userUniversalId=mhunter
idpUserGuid=mhunter
```

```
idpGuid=d15e9718-404d-4345-9198-1625d093c079
idpName=TestConfig
idpType=ldap
completedAmrNames=["PWD"]
clientSslCert=
x-ssp-base-uri=https://example.com
x-ssp-issuer-url=https://example.com/default
x-ssp-host=example.com
x-ssp-path=/default
providerName=customSampleProvider2
providerId=2539cd04-02f1-453e-ab20-8b1b22f84f28
----- providerContext -----
----- metadataContext -----
env=test
org=123
region=us-east-1
----- operationalContext -----
spi.auth.factor.flowid=flowId2-0.FirstTime.challenge
----- credData -----
spi.auth.factor.response=accept
----- headers -----
accept=[application/json]
content-type=[application/json]
x-client-transaction-id=[MyClientTxnId]
x-transaction-id=[76e72ee8-a659-49a6-af96-046220ede29c]
x-forwarded-for=[]
x-tenant-id=[51d8c90f-bd95-41eb-a73d-fb725b84cbb6]
x-tenant-name=[default]
customprovider=[CustomSampleProvider2]
user-agent=[PostmanRuntime/7.29.0]
cookie=[]
x-api-
key=[0BX9mtir0+W8C3r4UqSL39POSo1BO35LYdCVf7Dp9WCH7m9mu4abrpnJs5A8AknrYwejLHcq9S7DNYj5Z8FKlpxFkHONl3e2wEjUiJ8Pf3UyJwrTXv
content-length=[2737]
host=[ssp-sample-ssp-sample-app-spi-demo.dev-sample.svc:443]
connection=[Keep-Alive]
accept-encoding=[gzip,deflate]
```

9.3.1 SPI Response

```
2022-04-16 22:13:05.874 INFO 1 --- [nio-8092-exec-8] i.s.a.CustomAuthenticatorApiController :
----- Custom Auth Sample Provider Response v1.0.0000 ----- ************************************** RESPONSE
status=accept
msg=Received response: 'accept' for Flow Id: 'flowId2-0.FirstTime.challenge'
----- clientRespContext -----
clientCustomData1=gga323a24
----- operationalContext -----
spi.response.header.proxy=header-by-spi-expect2parts
spi.response.cookie.proxy=cookie-by-spi
spi.auth.factor.flowid=flowId2-0.FirstTime.challenge.accept
----- credData -----
spi.auth.factor.response=accept
----- headers -----
header-by-spi-expect2parts=[sample-header-by-spi-part1, sample-header-by-spi-part2]
```

```
CustomResponseHeader=[sample-header-by-provider]
RandomResponseHeader=[sample-header-by-spi-not-requested]
----- cookies -----
cookie-by-spi=sample-cookie-by-spi; Domain=example.com; Path=/default; Secure; HttpOnly
CustomResponseCookie=sample-cookie-by-provider; Domain=example.com; Path=/default; Secure; HttpOnly
```

9.3 "Accept Response" API Response


```
{
"flowState":
 "T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80N
"userName": "mhunter",
"nextaction": "CUSTOM_AUTH",
"currentFactors": [
"CUSTOM/ea7ae57b-104c-45fb-91fe-51c29ff56596"
],
"data": {
"status": "accept",
"clientRespContext": [
{
"key": "clientCustomData1",
"value": "gga323a24"
}
],
"credData": [
{
"key": "spi.auth.factor.response",
"value": "Sample Custom Auth SPI: credentials validated successfully"
}
]
},
"additional": {
"currentFactorLevel": 3,
"isCustom": true,
"isCustomFirstTime": true,
"customProvider": "ea7ae57b-104c-45fb-91fe-51c29ff56596",
"customFlowId": "flowId3-0"
}
}
```


Step 10: SignIn Client orchestrates interaction with "CustomSampleProvider3"

10.1 "First Time" API Request

```
curl --location --request POST 'https://example.com/default/factor/v1/CustomAuthenticator' \
--header 'Content-Type: application/json' \
--header 'X-Flow-State:
 T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80NH
 \
--header 'Authorization: Bearer
 eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0UG
wqHxEO4--DHhXmUpa-
H5ptkOnjGdw5Z8uQeHnlosG_SdaqoKSNAruHdIhwWFqMB2Yie73RNfo96L13NGAJQq9kFA24awzKei7yp01LFD0z086HtYoMKP3dDVF-
bNEBZRZZC4uE8grMUfmeqqw3uGq1QIFxcGtjeL3z7B_RaniLpVBIORGY6YZfpJz0XWMr_w1t41XaRb9-
PMxg4xwbNZ106SnamyOkDPyoCl_x_OxtCzuj_gII-KfqzmNyH2HE_yRwThaw' \
```

```
--header 'Cookie: CustomResponseCookie=sample-cookie-by-provider; cookie-by-spi=sample-cookie-by-spi' \
--data-raw '{
"operationalContext": [
{
"key": "spi.auth.factor.flowid",
"value": "flowId3-0.FirstTime"
},
{
"key": "spi.auth.factor.firsttime",
"value": "true"
}
],
"credData": [
]
}'
```

10.1.1 SPI Request

```
2022-04-16 22:20:19.172 INFO 1 --- [nio-8092-exec-5] i.s.a.CustomAuthenticatorApiController :
----- Custom Auth Sample Provider Request v1.0.0000 ----- ****************************************** REQUEST
subject=mhunter
----- clientContext -----
clientCustomData1=gga323a24
----- ahContext -----
tid=51d8c90f-bd95-41eb-a73d-fb725b84cbb6
tname=default
clientIp=10.230.9.183
userIp=1.1.1.2
txnId=76e72ee8-a659-49a6-af96-046220ede29c
clientTxnId=MyClientTxnId
locale=null
userId=5fa64f0b-e6e0-4479-9c5e-3d1f2be93834
userUniversalId=mhunter
idpUserGuid=mhunter
idpGuid=d15e9718-404d-4345-9198-1625d093c079
idpName=TestConfig
idpType=ldap
completedAmrNames=["OTP","PWD","FIDO"]
clientSslCert=
x-ssp-base-uri=https://example.com
x-ssp-issuer-url=https://example.com/default
x-ssp-host=example.com
x-ssp-path=/default
providerName=customSampleProvider3
providerId=ea7ae57b-104c-45fb-91fe-51c29ff56596
----- providerContext -----
----- metadataContext -----
env=test
org=123
region=us-east-1
----- operationalContext -----
spi.auth.factor.flowid=flowId3-0
spi.auth.factor.firsttime=true
----- credData -----
```

```
----- headers -----
accept=[application/json]
content-type=[application/json]
x-client-transaction-id=[MyClientTxnId]
x-transaction-id=[76e72ee8-a659-49a6-af96-046220ede29c]
x-forwarded-for=[]
x-tenant-id=[51d8c90f-bd95-41eb-a73d-fb725b84cbb6]
x-tenant-name=[default]
customprovider=[CustomSampleProvider3]
user-agent=[PostmanRuntime/7.29.0]
cookie=[]
x-api-
key=[0BX9mtir0+W8C3r4UqSL39POSo1BO35LYdCVf7Dp9WCH7m9mu4abrpnJs5A8AknrYwejLHcq9S7DNYj5Z8FKlpxFkHONl3e2wEjUiJ8Pf3UyJwrTXv
content-length=[2734]
host=[ssp-sample-ssp-sample-app-spi-demo.dev-sample.svc:443]
connection=[Keep-Alive]
accept-encoding=[gzip,deflate]
```

10.1.1 SPI Response

```
2022-04-16 22:20:19.173 INFO 1 --- [nio-8092-exec-5] i.s.a.CustomAuthenticatorApiController :
----- Custom Auth Sample Provider Response v1.0.0000 ----- ************************************** RESPONSE
status=Challenge
msg=First time for Flow Id: flowId3-0
----- clientRespContext -----
clientCustomData1=gga323a24
----- operationalContext -----
spi.auth.factor.flowid=flowId3-0.FirstTime
----- credData -----
----- headers -----
----- cookies -----
```

10.1 "First Time" API Response

```
{
"flowState":
 "T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80N
"userName": "mhunter",
"nextaction": "CUSTOM_AUTH",
"currentFactors": [
"CUSTOM/ea7ae57b-104c-45fb-91fe-51c29ff56596"
],
"data": {
"status": "Challenge",
"clientRespContext": [
{
"key": "clientCustomData1",
"value": "gga323a24"
}
],
"credData": [
{
"key": "spi.auth.factor.challenge",
```

```
"value": "Sample Custom Auth SPI: enter authentication status [Accept, Challenge, Reject, Terminate,
 Exception]:"
}
]
},
"additional": {
"currentFactorLevel": 3,
"isCustom": true,
"customProvider": "ea7ae57b-104c-45fb-91fe-51c29ff56596",
"customFlowId": "flowId3-0.FirstTime"
}
}
```

10.2 "Challenge" API Request

```
curl --location --request POST 'https://example.com/default/factor/v1/CustomAuthenticator' \
--header 'Content-Type: application/json' \
--header 'X-Flow-State:
 T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80NH
 \
--header 'Authorization: Bearer
 eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0UG
wqHxEO4--DHhXmUpa-
H5ptkOnjGdw5Z8uQeHnlosG_SdaqoKSNAruHdIhwWFqMB2Yie73RNfo96L13NGAJQq9kFA24awzKei7yp01LFD0z086HtYoMKP3dDVF-
bNEBZRZZC4uE8grMUfmeqqw3uGq1QIFxcGtjeL3z7B_RaniLpVBIORGY6YZfpJz0XWMr_w1t41XaRb9-
PMxg4xwbNZ106SnamyOkDPyoCl_x_OxtCzuj_gII-KfqzmNyH2HE_yRwThaw' \
--header 'Cookie: CustomResponseCookie=sample-cookie-by-provider; cookie-by-spi=sample-cookie-by-spi' \
--data-raw '{
"operationalContext": [
{
"key": "spi.auth.factor.flowid",
"value": "flowId3-0.FirstTime.challenge"
}
],
"credData": [
{
"key": "spi.auth.factor.response",
"value": "challenge"
}
]
}'
```

10.2.1 SPI Request

```
2022-04-16 22:24:56.162 INFO 1 --- [nio-8092-exec-1] i.s.a.CustomAuthenticatorApiController :
----- Custom Auth Sample Provider Request v1.0.0000 ----- ****************************************** REQUEST
subject=mhunter
----- clientContext -----
clientCustomData1=gga323a24
----- ahContext -----
tid=51d8c90f-bd95-41eb-a73d-fb725b84cbb6
tname=default
clientIp=10.230.9.183
userIp=1.1.1.2
```

```
txnId=76e72ee8-a659-49a6-af96-046220ede29c
clientTxnId=MyClientTxnId
locale=null
userId=5fa64f0b-e6e0-4479-9c5e-3d1f2be93834
userUniversalId=mhunter
idpUserGuid=mhunter
idpGuid=d15e9718-404d-4345-9198-1625d093c079
idpName=TestConfig
idpType=ldap
completedAmrNames=["OTP","PWD","FIDO"]
clientSslCert=
x-ssp-base-uri=https://example.com
x-ssp-issuer-url=https://example.com/default
x-ssp-host=example.com
x-ssp-path=/default
providerName=customSampleProvider3
providerId=ea7ae57b-104c-45fb-91fe-51c29ff56596
----- providerContext -----
----- metadataContext -----
env=test
org=123
region=us-east-1
----- operationalContext -----
spi.auth.factor.flowid=flowId3-0.FirstTime
----- credData -----
spi.auth.factor.response=challenge
----- headers -----
accept=[application/json]
content-type=[application/json]
x-client-transaction-id=[MyClientTxnId]
x-transaction-id=[76e72ee8-a659-49a6-af96-046220ede29c]
x-forwarded-for=[]
x-tenant-id=[51d8c90f-bd95-41eb-a73d-fb725b84cbb6]
x-tenant-name=[default]
customprovider=[CustomSampleProvider3]
user-agent=[PostmanRuntime/7.29.0]
cookie=[]
x-api-
key=[0BX9mtir0+W8C3r4UqSL39POSo1BO35LYdCVf7Dp9WCH7m9mu4abrpnJs5A8AknrYwejLHcq9S7DNYj5Z8FKlpxFkHONl3e2wEjUiJ8Pf3UyJwrTXv
content-length=[2747]
host=[ssp-sample-ssp-sample-app-spi-demo.dev-sample.svc:443]
connection=[Keep-Alive]
accept-encoding=[gzip,deflate]
```

10.2.1 SPI Response

```
2022-04-16 22:24:56.162 INFO 1 --- [nio-8092-exec-1] i.s.a.CustomAuthenticatorApiController :
----- Custom Auth Sample Provider Response v1.0.0000 ----- ************************************** RESPONSE
status=challenge
msg=Received response: 'challenge' for Flow Id: 'flowId3-0.FirstTime'
----- clientRespContext -----
clientCustomData1=gga323a24
----- operationalContext -----
spi.response.header.proxy=header-by-spi-expect2parts
```

```
spi.response.cookie.proxy=cookie-by-spi
spi.auth.factor.flowid=flowId3-0.FirstTime.challenge
----- credData -----
spi.auth.factor.response=challenge
----- headers -----
header-by-spi-expect2parts=[sample-header-by-spi-part1, sample-header-by-spi-part2]
CustomResponseHeader=[sample-header-by-provider]
RandomResponseHeader=[sample-header-by-spi-not-requested]
----- cookies -----
cookie-by-spi=sample-cookie-by-spi; Domain=example.com; Path=/default; Secure; HttpOnly
CustomResponseCookie=sample-cookie-by-provider; Domain=example.com; Path=/default; Secure; HttpOnly
RandomResponseCookie=sample-cookie-by-spi-not-requested; Domain=example.com; Path=/default; Secure; HttpOnly
```

10.2 "Challenge" API Response


```
{
"flowState":
 "T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80N
"userName": "mhunter",
"nextaction": "CUSTOM_AUTH",
"currentFactors": [
"CUSTOM/ea7ae57b-104c-45fb-91fe-51c29ff56596"
],
"data": {
"status": "challenge",
"clientRespContext": [
{
"key": "clientCustomData1",
"value": "gga323a24"
}
],
"credData": [
{
"key": "spi.auth.factor.challenge",
"value": "Sample Custom Auth SPI: enter authentication status [Accept, Challenge, Reject, Terminate,
 Exception] "
}
]
},
"additional": {
"currentFactorLevel": 3,
"isCustom": true,
"customProvider": "ea7ae57b-104c-45fb-91fe-51c29ff56596",
"customFlowId": "flowId3-0.FirstTime.challenge"
}
}
```


10.3 "Accept Response" API Request

```
curl --location --request POST 'https://example.com/default/factor/v1/CustomAuthenticator' \
--header 'Content-Type: application/json' \
--header 'X-Flow-State:
 T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80NH
 \
```

```
--header 'Authorization: Bearer
 eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0UG
wqHxEO4--DHhXmUpa-
H5ptkOnjGdw5Z8uQeHnlosG_SdaqoKSNAruHdIhwWFqMB2Yie73RNfo96L13NGAJQq9kFA24awzKei7yp01LFD0z086HtYoMKP3dDVF-
bNEBZRZZC4uE8grMUfmeqqw3uGq1QIFxcGtjeL3z7B_RaniLpVBIORGY6YZfpJz0XWMr_w1t41XaRb9-
PMxg4xwbNZ106SnamyOkDPyoCl_x_OxtCzuj_gII-KfqzmNyH2HE_yRwThaw' \
--header 'Cookie: CustomResponseCookie=sample-cookie-by-provider; cookie-by-spi=sample-cookie-by-spi' \
--data-raw '{
"operationalContext": [
{
"key": "spi.auth.factor.flowid",
"value": "flowId3-0.FirstTime.challenge"
}
],
"credData": [
{
"key": "spi.auth.factor.response",
"value": "accept"
}
]
}'
```

10.3.1 SPI Request

```
2022-04-16 22:26:49.666 INFO 1 --- [nio-8092-exec-8] i.s.a.CustomAuthenticatorApiController :
----- Custom Auth Sample Provider Request v1.0.0000 ----- ****************************************** REQUEST
subject=mhunter
----- clientContext -----
clientCustomData1=gga323a24
----- ahContext -----
tid=51d8c90f-bd95-41eb-a73d-fb725b84cbb6
tname=default
clientIp=10.230.9.183
userIp=1.1.1.2
txnId=76e72ee8-a659-49a6-af96-046220ede29c
clientTxnId=MyClientTxnId
locale=null
userId=5fa64f0b-e6e0-4479-9c5e-3d1f2be93834
userUniversalId=mhunter
idpUserGuid=mhunter
idpGuid=d15e9718-404d-4345-9198-1625d093c079
idpName=TestConfig
idpType=ldap
completedAmrNames=["OTP","PWD","FIDO"]
clientSslCert=
x-ssp-base-uri=https://example.com
x-ssp-issuer-url=https://example.com/default
x-ssp-host=example.com
x-ssp-path=/default
providerName=customSampleProvider3
providerId=ea7ae57b-104c-45fb-91fe-51c29ff56596
----- providerContext -----
----- metadataContext -----
env=test
```

```
org=123
region=us-east-1
----- operationalContext -----
spi.auth.factor.flowid=flowId3-0.FirstTime.challenge
----- credData -----
spi.auth.factor.response=accept
----- headers -----
accept=[application/json]
content-type=[application/json]
x-client-transaction-id=[MyClientTxnId]
x-transaction-id=[76e72ee8-a659-49a6-af96-046220ede29c]
x-forwarded-for=[]
x-tenant-id=[51d8c90f-bd95-41eb-a73d-fb725b84cbb6]
x-tenant-name=[default]
customprovider=[CustomSampleProvider3]
user-agent=[PostmanRuntime/7.29.0]
cookie=[]
x-api-
key=[0BX9mtir0+W8C3r4UqSL39POSo1BO35LYdCVf7Dp9WCH7m9mu4abrpnJs5A8AknrYwejLHcq9S7DNYj5Z8FKlpxFkHONl3e2wEjUiJ8Pf3UyJwrTXv
content-length=[2754]
host=[ssp-sample-ssp-sample-app-spi-demo.dev-sample.svc:443]
connection=[Keep-Alive]
accept-encoding=[gzip,deflate]
```

10.3.1 SPI Response

```
2022-04-16 22:26:49.667 INFO 1 --- [nio-8092-exec-8] i.s.a.CustomAuthenticatorApiController :
----- Custom Auth Sample Provider Response v1.0.0000 ----- ************************************** RESPONSE
status=accept
msg=Received response: 'accept' for Flow Id: 'flowId3-0.FirstTime.challenge'
----- clientRespContext -----
clientCustomData1=gga323a24
----- operationalContext -----
spi.response.header.proxy=header-by-spi-expect2parts
spi.response.cookie.proxy=cookie-by-spi
spi.auth.factor.flowid=flowId3-0.FirstTime.challenge.accept
----- credData -----
spi.auth.factor.response=accept
----- headers -----
header-by-spi-expect2parts=[sample-header-by-spi-part1, sample-header-by-spi-part2]
CustomResponseHeader=[sample-header-by-provider]
RandomResponseHeader=[sample-header-by-spi-not-requested]
----- cookies -----
cookie-by-spi=sample-cookie-by-spi; Domain=example.com; Path=/default; Secure; HttpOnly
CustomResponseCookie=sample-cookie-by-provider; Domain=example.com; Path=/default; Secure; HttpOnly
RandomResponseCookie=sample-cookie-by-spi-not-requested; Domain=example.com; Path=/default; Secure; HttpOnly
```

10.3 "Accept Response" API Response

```
{
"flowState":
 "T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3MW80N
"userName": "mhunter",
"nextaction": "AUTH_ALLOWED",
"data": {
```

```
"status": "accept",
"clientRespContext": [
{
"key": "clientCustomData1",
"value": "gga323a24"
}
],
"credData": [
{
"key": "spi.auth.factor.response",
"value": "Sample Custom Auth SPI: credentials validated successfully"
}
]
},
"authCompleteUrl": "https://example.com/default/oauth2/v1/authcomplete?x-flow-
state=T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3M
"additional": {
"authLevelsData": {
"55434a3c-f1b3-4b51-9696-6db7556aaf11+mhunter+1": "PASSWORD",
"55434a3c-f1b3-4b51-9696-6db7556aaf11+mhunter+3": "CUSTOM/ea7ae57b-104c-45fb-91fe-51c29ff56596",
"55434a3c-f1b3-4b51-9696-6db7556aaf11+mhunter+2": "CUSTOM/2539cd04-02f1-453e-ab20-8b1b22f84f28"
},
"currentFactorLevel": 3,
"clientRespContext": [
{
"key": "deviceId",
"value": "post risk success"
}
]
}
}
```


Step 11: SignIn Client executes redirect to Relying Party's Callback URL

Request

```
curl --location --request GET 'https://example.com/default/oauth2/v1/authcomplete?x-flow-
state=T3ZkbFJicXYxVENMRFJtdmEwaWNRTkQ1bkNJeGs0RHBxRlQ4cWhJZFlLemIrVjZGTEFjV0I5V2JjejBsZFhnQVh0R0JydjIyOWg0S2wzN3pIYUw3M
 \
--header 'Cookie: CustomResponseCookie=sample-cookie-by-provider; cookie-by-spi=sample-cookie-by-spi'
```

Response

```
Redirect to location: https://www.example.com?code=4byYV2LChpyHKj5Dsk6jHWTPDqQXjg&X-TRANSACTION-ID=76e72ee8-
a659-49a6-af96-046220ede29c
```

Step 12: Relying Party acquires Identity Token and Access Token

Request

```
curl --location --request POST 'https://example.com/default/oauth2/v1/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'x-tenant-name: default' \
--header 'Authorization: Basic
 YTk0MDUzOWEtZGQxMC00ZTIxLWI3NjUtZTliNTRkYzIwMzdkOmNkNzNhYTUzLTZkM2EtNDYwYS05NTE4LTYyMTY5Y2Y2ZjA2NQ==' \
--header 'Cookie: CustomResponseCookie=sample-cookie-by-provider; cookie-by-spi=sample-cookie-by-spi' \
```

```
--data-urlencode 'grant_type=authorization_code' \
--data-urlencode 'code=4byYV2LChpyHKj5Dsk6jHWTPDqQXjg' \
--data-urlencode 'expiry=600' \
--data-urlencode 'scope=offline_access'
```

Response


```
{
"access_token":
 "eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0U
PC9KABspczOV4gQeVljcS1sHBA",
"token_type": "Bearer",
"refresh_token": "9Hv5AEQB4geKHYfZWXpkyh1Jz63ASkj8aG7o",
"expires_in": 3600,
"id_token":
 "eyJ4NXQjUzI1NiI6InY1ckluZElocS1peXdRdTMxcTF3Vm9idEQ1YVRTckFyNmloa2d3WmJQeU0iLCJ4NXQiOiJZbExTMDdlWG5ZRVU1TVgxQlpMTDd0U
zVxKKI-
yvFjbTDEbah01n_dgPvNb32R2vZLER_rFFOn4vQKnNbtkDgMFPQP3AdZAFk2d6sOCHnmPl0yRD5p0BlQigriWJZa6AF5oNokxlrt_cojZ5cEPAkDLzAkTOn
eCIjniJAPoSQCULjYsw"
}
```


Identity Token


```
{
"at_hash": "WAMtY2iJlbkdbNJUVSm0PQ",
"sub": "Maree.HUNTER@DEMOCORP.com",
"idp_name": "TestConfig",
"user_dn": "cn=mhunter,ou=IT,o=democorp,c=us",
"amr": [
"OTP",
"PWD",
"FIDO"
],
"user_loginid": "mhunter",
"idp_guid": "d15e9718-404d-4345-9198-1625d093c079",
"iss": "https://example.com/default/",
"_tags": "tenantIdToken=NODATA, titleIdStore=Services Accountant, orgAppMd=123",
"user_guid": "5fa64f0b-e6e0-4479-9c5e-3d1f2be93834",
"aal": "AAL3",
"tid": "51d8c90f-bd95-41eb-a73d-fb725b84cbb6",
"sid": "23d537de-8242-4311-b2e2-1200bf9e653b",
"user_subattrmappingname": "email",
"acr": "urn:iam:acr:aal:aal1",
"_titleIdStore": "Services Accountant",
"azp": "a940539a-dd10-4e21-b765-e9b54dc2037d",
"user_loginidattrmappingname": "user_loginid",
"auth_time": 1650148432,
"_acctNumStatic": "12345",
"idp_user_guid": "mhunter",
"exp": 1650170041,
"iat": 1650148441,
"client_name": "DemoClient",
"jti": "ffd98a99-5979-4279-9c66-d18d57c96f8f",
"email": "Maree.HUNTER@DEMOCORP.com",
```

```
"idp_type": "ldap",
"client_tname": "default",
"client_tid": "51d8c90f-bd95-41eb-a73d-fb725b84cbb6",
"ver": "1.0",
"user_universalid": "mhunter",
"tname": "default",
"tok_type": "IT",
"_orgAppMd": "123",
"user_riskscore": 43,
"aud": [
"a940539a-dd10-4e21-b765-e9b54dc2037d",
"https://example.com/default/"
],
"nbf": 1650148441,
"user_zfp": false,
"client_transactionid": "MyClientTxnId",
"tok_source": "AZC",
"_tenantIdToken": "NODATA"
}
```


Access Token


```
{
"sub": "Maree.HUNTER@DEMOCORP.com",
"idp_name": "TestConfig",
"user_loginid": "mhunter",
"idp_guid": "d15e9718-404d-4345-9198-1625d093c079",
"iss": "https://example.com/default/",
"_tags": "tenantIdToken=NODATA, titleIdStore=Services Accountant, orgAppMd=123",
"user_guid": "5fa64f0b-e6e0-4479-9c5e-3d1f2be93834",
"tid": "51d8c90f-bd95-41eb-a73d-fb725b84cbb6",
"sid": "23d537de-8242-4311-b2e2-1200bf9e653b",
"user_subattrmappingname": "email",
"_titleIdStore": "Services Accountant",
"sub_type": "USER",
"azp": "a940539a-dd10-4e21-b765-e9b54dc2037d",
"user_loginidattrmappingname": "user_loginid",
"scope": "urn:iam:m.me openid",
"_acctNumStatic": "12345",
"idp_user_guid": "mhunter",
"exp": 1650152041,
"iat": 1650148441,
"client_name": "DemoClient",
"jti": "523dfa3d-1c7d-4f77-add5-057fb3828d5b",
"idp_type": "ldap",
"client_tname": "default",
"client_tid": "51d8c90f-bd95-41eb-a73d-fb725b84cbb6",
"ver": "1.0",
"user_universalid": "mhunter",
"azpacr": "2",
"tname": "default",
"tok_type": "AT",
"_orgAppMd": "123",
"user_riskscore": 43,
```

```
"user_amr": [
"OTP",
"PWD",
"FIDO"
],
"aud": "https://example.com/default/",
"nbf": 1650148441,
"user_zfp": false,
"client_transactionid": "MyClientTxnId",
"user_aal": "AAL3",
"tok_source": "AZC",
"user_acr": "urn:iam:acr:aal:aal1",
"_tenantIdToken": "NODATA"
}
```


Integrating with Custom Password Authenticator

Implement the Custom Password Authenticator SPI to integrate your existing password authentication infrastructure with IDSP as a hosted web service.

This topic covers the following sections:

- Custom Password Authentication Overview
- Steps to Integrate with Custom Password Authenticator SPI
- API Schema - (Swagger Content)
- Configuration to Support Custom Password Authentication
  - Configuring the Custom Provider - providerProperties
  - Updating Authentication (AuthN)Policy
- IDSP Password Authentication API Specification
- SPI Specification to be Used by the Customer
- Use Case: Custom Password Authentication Flow

Custom Password Authentication Overview

The IDSP Custom Password Authenticator enables integration with your existing password authentication infrastructure through a Service Provider Interface (SPI). To use this functionality, you must implement the SPI as a web service within your environment. Your implementation must adhere to the IDSP API specification to ensure seamless communication between the solution and your custom authentication service.

Steps to Integrate with Custom Password Authenticator SPI

You have to perform the following steps to integrate with the Custom Password Authenticator SPI:

1. Develop the Custom Password Authentication service based on the Password Authentication-Open API Schema - (Swagger Content) provided by IDSP.
2. Create a Custom Password Authentication Custom Provider in IDSP using the Provider endpoint- `https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders`
3. Update the Authentication Policy with the custom provider Id in IDSP.

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/CustomProviders`
  should look similar to:
  `https://example.com/default/admin/v1/CustomProviders`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

These steps have been described in the subsequent sections.

API Schema - (Swagger Content)

To view the sample representation of the API schema, see the **Common Open API Spec for all Custom Providers - (Swagger Content)** section in Integrating with Custom SPI.

Configurations to support Custom Password Authentication in IDSP

Creating a Custom Password Authentication Custom Provider

- Create a custom password provider in IDSP to use your current password authentication infrastructure using the following API:

```
POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders
```

NOTE
To refer to the Custom Provider APIs which can be used to perform CRUD operations in IDSP, see here.

Configuring the Custom Provider - providerProperties

Custom Provider - providerProperties

A custom provider uses the providerProperties key-value pairs to define a provider-specific behavior. There are two property types:

- **Custom**: All custom properties are published into the providerContext of the SPI request and are consumed by the custom SPI.
- **Well-known**: All well-known properties begin with the **spi.** prefix, representing the directives to be performed by IDSP when sending requests to and receiving responses from the custom SPI.

The following table lists the providerProperties that must be configured to support the use of a custom authentication factor:

Type	providerContext	Description
Custom	**key**: <custom-key> **value**: <custom-value>	Publishes a custom key-value pair into the ProviderContext of the custom SPI. The value in a key-value pair supports dynamic substitution using the ${keyword} notation. For information, see Dynamic Environment Substitution.

Well-known	**key**: spi.discover.capabilities **value**: CSV list of capabilities	Specifies the capabilities that are supported by the custom provider using multiple, comma-delimited placeholders. This enables IDSP to recognize the configured Custom Provider as having custom authentication capabilities, for example: • **custompasswordauthenticator**: Specifies that the custom SPI supports a custom password authentication factor that is accessible via "/CustomPasswordAuthenticator" runtime flow.
Well-known	**key**: spi.auth.factor.amr **value**: <amr-value>	Represents one or more AMR values (in CSV format) of the custom authentication factor. These AMR values are to be associated with the custom provider. You can enter multiple comma-delimited values from the following list: • PWD: Password style authentication • OTP: One time code (SMS, EMAIL, IVR) • FIDO: Passwordless • HWK: Hardware Key (Yubikey) • MOTP: Mobile one time code • PUSH: Mobile push • POP: SiteMinder X509 integration use case • SWK: N/A • WIA: SiteMinder IWA/Kerberos integration use case • <custom>: A custom AMR value If multiple instances of this key are provided, the last one takes precedence. If this key is specified by ProviderContext, and also returned by the OperationalContext of the custom SPI, then the value from the OperationalContext takes precedence.

Custom Provider - providerHeaders

A custom provider uses providerHeaders in a key-value pair format to define provider-specific headers to be sent to the custom SPI. Similar to "providerProperties", the headers definitions also supports dynamic substitution using the ${keyword} notation. For information, see Dynamic Environment Variable Substitution.

Updating the Authentication Policy

To support the implementation of Custom Password Authenticator, you must update the IDSP Authentication Policy.

Perform the following PATCH call to update the Authentication Policy to add a custom provider:

```
PATCH https://${SSP_FQDN}/{{tenantName}}/admin/v1/AuthPolicies/{{policyId}}
```

Sample Patch payload:

```
{
```

```
    "Operations": [
        {
            "op": "replace",
            "path": "/rules/[id eq \"1c722537-aaa2-4d62-a476-147e693c097a\"]/result/obligations/0",
            "value": "PASSWORD/{custom}2db88c50-93a9-4618-ad64-23b9291811cb"
        }
    ]
}
```

Where, the "**value**" field holds the PASSWORD/{custom} custom provider id which was returned when creating a Custom Provider in IDSP.

IDSP Password Authentication API Specification

For Password Authentication API specification, see Password Authentication API Specification .

SPI Specification to be Used by the Customer

The following SPI specification should be used:

SPI Name	CustomPasswordAuthenticator
SPI URL	https://<CustomProviderHost>/CustomPasswordAuthenticator
SPI OPERATION	POST
AUTHORIZATION	As per the SPI security model: • OAuth Client Assertion • OAuth ClientID and Secret • ApiKey • Basic Auth
SPI RESPONSE CODE	200 Success, 400 Failure, 500 Server Error

SPI Request:
The SPI request to a custom provider can contain cookies and headers that are requested by SPI, along with the following parameters:

Parameter	Type	Mandatory	Description
subject	String	Yes	Specifies the unique identifier of the user representing **user_loginid** semantics as configured in the ID store config resource.

clientContext	Array of key-value pairs	Yes	Specifies that the Client context parameters that are received as part of the ".../auth/v1/ authenticate" or web/authorized (IT) flows are passed here as key-value pairs, which are understood by client. Example: transactionIdentifier, appid. `{ "subject": "<userLoginid>", "channel": "web", "ipAddress": "10.10.123.124", "action":"authenticate", "rememberMe" : true, "device": { "signature": { "iaAuthData": "_v02MyUgNGgYOi88OTk0..... } }, "clientContext": [{ "key": "transactionIdentifier", "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40" }] }`
ahContext	Array of key-value pairs	Yes	Contains the IDSP context parameters. For information, see the *Sharing IDSP Context with SPI* section in Integrating with Custom SPI.

providerContext	Array of key-value pairs	Yes	Specifies the custom provider-specific configuration. For information, see Integrating with Custom SPI. • Provider should be configured with "spi.discover.capabilities" key containing "custompasswordd value. This enables Admin UI to recognize configured custom provider as supporting a PASSWORD policy obligation. • Custom Password Provider can be configured with "spi.auth.factor.amr" key containing Authentication Method Reference (AMR) values (in CSV format) to be associated with the custom authentication provider. – Multiple comma-delimited values can be provided. – If multiple instances of this key are provided, the last one takes precedent. – If this key is specified by both ProviderContext, and returned by SPI's OperationalContext, then the value from the OperationalContext takes precedent.
operationalContext	Array of key-value pairs	Yes	Specifies the optional directives to be processed by SPI: **Key** spi.auth.factor.amr
metadataContext	Array of key-value pairs	Yes	Specifies the metadata of the application that is requesting authentication.

credData	Array of key-value pairs	Yes	Specifies the credential data that is required by the custom SPI to process authentication. "spi.auth.factor.password" key carries the password value.

SPI Response

The SPI response from a custom provider can contain cookies and headers that are requested by SPI's response, along with the following parameters:

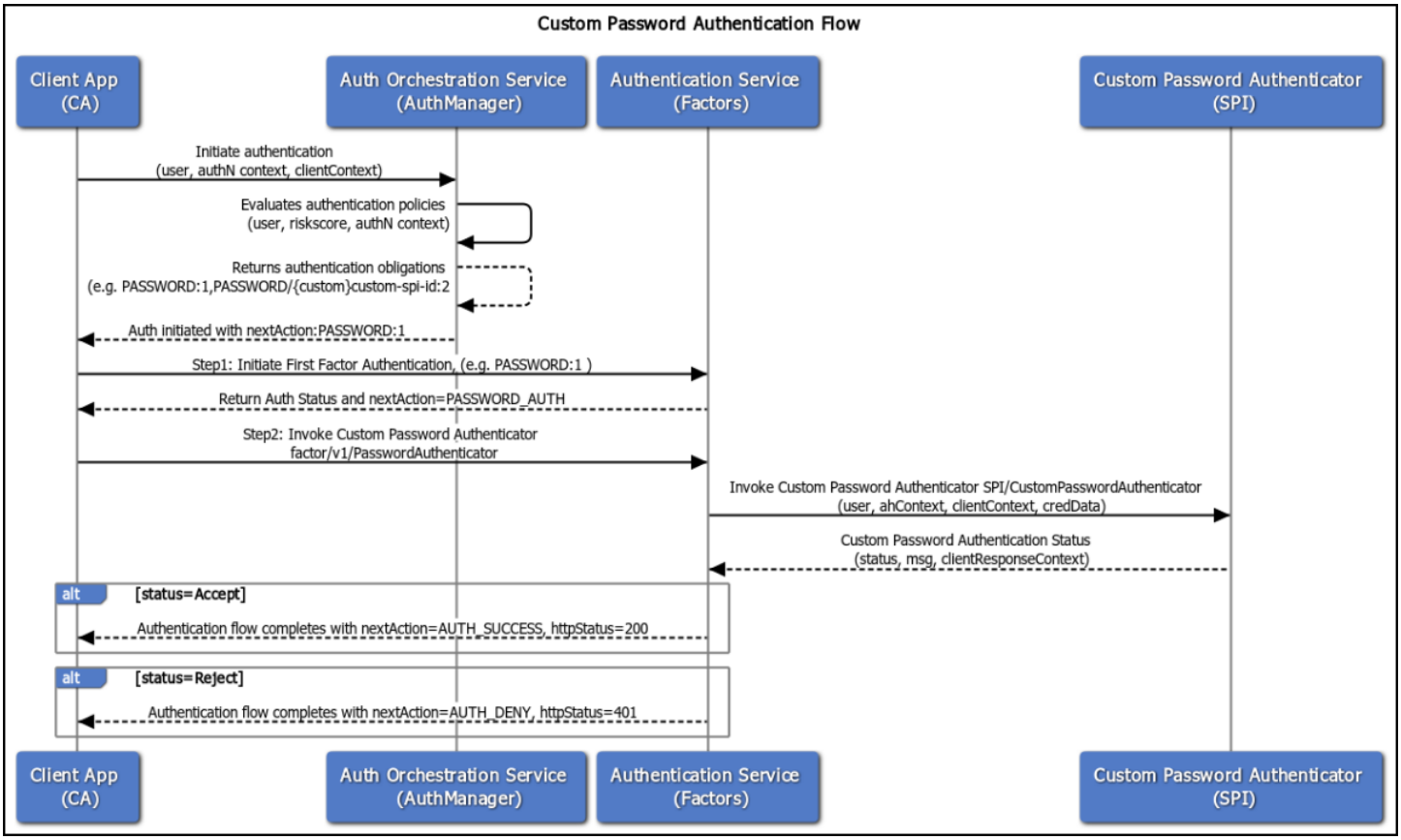
Parameter	Type	Description	
msg	String	Specifies the message to be sent back to IDSP to be logged in the operational log.	
status	String	Specifies the status of the custom password authentication. The supported values are Accept| Reject. All other values are treated as Reject. In case of Reject, additional data should be provided via directives in clientRespContext.	
clientRespContext	Array of key-value pairs	Specifies the response data to be returned by the SPI to satisfy the contract expected by ".../factors/v1/PasswordAuthenticator" API:	
		Key	**Value**
		spi.auth.factor.password.message	Contains the message IDSP to return to the signin client in case of successful and failed authentication.
		spi.auth.factor.password.errorcode	Contains the error code IDSP to return to the signin client when authentication fails.
		spi.auth.factor.password.reseturl	Contains the password reset URL IDSP to return to the signin client.
operationalContext	Array of key-value pairs	Specifies the optional directives that are indicated by the SPI to be processed by IDSP. The following directives are processed:	
		Key	**Value**
		spi.auth.factor.amr	Represents the "amr" value of the custom password authentication, returned after the user has successfully completed authentication. If this value is provided by the SPI, it overwrites the value that is configured on the custom provider.

Use Case: Custom Password Authentication Flow

List of participants and what they represent:

- Client App (CA) (Web App): This is the application that calls the solution's APIs to acquire the identity token and complete the authentication flow.
- IDSP microservices:
  - Auth Orchestration Service (AuthManager)- This is the ***"/auth/v1/authenticate"*** API used to initiate the authentication flow.
  - Authentication Service (Factors)- This is the ***"/factor/v1/PasswordAuthenticator"*** API used to authenticate an identity using a password.
  - Customer's Password Authentication Service Provider Interface (SPI)- This is the customer's implementation of Password Authentication Service Provider Interface (SPI) which will be used to perform the Password Authentication flow.

The following sequence diagram explains the authentication workflow between a Client and a Customers Password Authenticator SPI:



Flow Description:

1. The Client invokes `https://${SSP_FQDN}/{{tenantName}}/auth/v1/authenticate` API to initiate the Authentication flow. The request includes parameters such as user, authN context, clientContext [example:txn identifier].
2. Internally, the ".../auth/v1/authenticate" API communicates with the Policy Engine to evaluate the configured policy rules for the client app and determines the authentication obligations. Also, sends a response to the client app. In the flow, it is assumed that the obligations returned are PASSWORD: 1, PASSWORD/{custom} -custom_spi_Id: 2.
3. The client successfully completes the first-factor authentication, PASSWORD: 1.
4. The client is challenged with the remaining obligation that is PASSWORD/{custom} -custom_spi_Id: 2 this is the custom password authenticator.
5. The ".../factor/v1/PasswordAuthenticator sends a response to the client app with "nextaction" as PASSWORD_AUTH.

6. The Client invokes the "...factor/v1/PasswordAuthenticator" API. The Factor API sends details including user, ahContext, clientContext, credData to the Custom Password Authenticator SPI.
   This step begins the interaction with the custom SPI that is represented by the custom provider, which is associated with the custom password authentication factor. The interaction continues until the API indicates AUTH_ALLOW or AUTH_DENY.
7. The custom SPI returns authentication status that includes, status, msg, clientResponseContext to the Factor API.
8. The factor API sends "msg" received from customer's password authenticator to the client app.
9. The custom Password authenticator SPI returns Accept/Reject as the status as per the custom SPI implementation.
   - **Accept**: Indicates that the custom SPI has successfully authenticated the user. IDSP advances the authentication flow as per authentication policy.
   - **Reject**: Indicates that the custom SPI has rejected credentials and expects the Client to redo the operation. IDSP increments the user lock strike count, and does not advance the authentication flow.

Integrating with Custom Claims Provider

Populate custom claims in SAML assertions and OIDC tokens by implementing and configuring a Custom Claim Service Provider Interface (SPI).

This topic covers the following sections:

- Custom Claims Provider Overview
- Steps to Integrate with Custom Claims SPI
- SPI specification to be used by the customer
- Creating a Custom Claims Provider
  - Configuring - providerProperties
  - Configuring - headerProperties
- Configurations to Support Custom Claims
  - Application Configuration
  - Tenant Settings Configuration
- Sample Custom Claims Provider
- Custom Claims SPI Examples

Custom Claims Provider Overview

To deliver custom identity data, you can build a Custom Claim SPI provider and host it as a web service in your environment. These providers can be applied at the tenant level for global use or at the application level for specific requirements. Note that while tenant-level providers serve as a default, an application-specific configuration takes precedence; only one provider can be active for a single application at any time.

Steps to Integrate with Custom Claims SPI

Perform the following steps to integrate the Custom Claims SPI with IDSP:

1. Develop the Custom Claims service based on the Open API specification that is made available to you in YAML format and can be downloaded. Click this link to download the IDSP SPI Spec.yml.
2. Create a Custom Claims Provider in IDSP using the Provider endpoint- `https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders`

   NOTE

   - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
   - The default value for `<tenantName>` is "default".
   - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.

For example:
`https://{{host}}/{{tenantName}}/admin/v1/CustomProviders`
should look similar to:
`https://example.com/default/admin/v1/CustomProviders`

- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

3. Update the tenant or the application level configuration in IDSP.

These steps have been described in detail in the subsequent sections.

SPI specification to be implemented by customer

While building the Custom Claim SPI, the implementation should follow the specified SPI specification:

SPI Name	**CustomClaimGetter**
SPI URL	https://<CustomProviderHost>:<port>/CustomClaimGetter
SPI OPERATION	POST
AUTHORIZATION	As per the SPI security model: • OAuth Client Assertion • OAuth ClientID and Secret • ApiKey • Basic Auth
SPI RESPONSE CODE	200 Success, 400 Failure, 500 Server Error

SPI Request:
The SPI request to a custom provider can contain cookies and headers that are requested by SPI, along with the following parameters:

Parameter	**Type**	**Mandatory**	**Description**
subject	String	Yes	Specifies the unique identifier of the user representing **user_loginid** semantics as configured in the ID store config resource.
clientContext	Array of key-value pairs	No	Not Used.
ahContext	Array of key-value pairs	Yes	Contains the IDSP context parameters. For information, see the *Sharing IDSP Context with SPI* section in Integrating with Custom SPI.

providerContext	Array of key-value pairs	Yes	Specifies the custom provider-specific configuration. For information, see Integrating with Custom SPI. • Provider should be configured with "spi.discover.capabilities" key containing "customclaim value. This enables Admin UI to recognize configured custom provider as supporting custom claims. • Custom Claims Provider can be configured with "spi.customclaim.token.types " key containing a combination of these values: ***it***, ***userat***, ***clientat***, ***userinfo***, samlassertion values (in CSV format) to be associated with the custom claims provider.
metadataContext	Array of key-value pairs	Yes	Specifies the metadata of the application that is requesting authentication.
tokenType	String	Yes	The type of token being issued corresponds to the values defined for "spi.customclaim.token.types".
rawType	String	Yes	Indicates that the raw data can be represented either in XML or JSON format.

SPI Response

The SPI response from a custom provider can contain cookies and headers that are requested by SPI's response, along with the following parameters:

Parameter	Type	Description
msg	String	Specifies the message to be sent back to IDSP to be logged in the operational log.
status	String	Specifies the status of the custom claims. The supported values are: success| failure.
clientRespContext	Array of key-value pairs	Not Used
claimsListMva	Array of key-value pairs	MVA (Multiple-Valued Attribute) claims. Each claim in the array can have multiple values.
claimsListSva	Array of key-value pairs	SVA (Single-Valued Attribute) claims.
claimsListRaw	List of values	Raw XML or JSON claims, depending on the value of request's rawType.
claimsListRawNs	Array of key-value pairs	For XML claims, specifies additional XML namespaces to be added to the SAML response.

Creating a Custom Claims Provider

After you have implemented the SPI, create a Custom Claim Provider in IDSP.

Creating a Custom Claims Provider

- Create a custom claims provider in IDSP using the following API:

```
POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders
```

Custom Provider- Sample Request Payload

```
{
    "providerName": "customRiskProvider",
    "version": "1.0",
    "apiKey": "6fd06fd82aceb78b0bbbe3990b865f96ea",
    "securityMethod": "apikey",
    "url": "https://example:xxxx",
   "providerHeaders": [
        {
            "name": "clientId",
            "value": "8289-b1e8817a3fec"
        }
    ],
    "providerProperties": [
        {
            "name": "app",
            "value": "test"
        },
        {
            "name": "region",
            "value": "us-west"
        }
    ]
}
```

Custom Provider - Sample Response Payload

The response generates a providerId for the custom Claim.


```
{
        "providerName": "customRiskProvider",
        "version": "1.0",
        "apiKey": "McbVn0nLba2e1vPFSw0hDtkvR0cGSLj3h1RgDNwQHTEL6gA8s8Xy4MKJ5lpP6iQU",
        "certAlias": "",
        "url": "http://example:xxxx",
        "providerHeaders": [
            {
                "name": "clientId",
                "value": "8289-b1e8817a3fec"
            }
        ],
        "providerProperties": [
            {
                "name": "app",
                "value": "test"
            },
            {
                "name": "region",
                "value": "us-west"
            }
        ],
        "createdBy": {
            "principalType": "CLIENT",
            "principalId": "c2d28f2c-4f4a-4557-9ba7-0248052e9a60",
            "principalName": "DefaultTenantClient"
        },
        "createdDateTime": "2024-04-11 10:26 AM UTC",
        "origin": "tenant",
        "ignoreSSLValidation": false,
        "providerId": "2cebd036-8ebe-4fe9-8752-5d6ca7ec5c12",
        "sslMode": "VERIFY_CA",
        "spiReadTimeoutMillis": "60000",
        "spiConnectTimeoutMillis": "60000",
        "requestCtxBase64": "",
        "clientId": "",
        "clientSecretAlias": "",
        "oauthScopes": "",
        "securityMethod": "apikey",
        "oauthTokenUrl": "",
        "oauthDiscoveryUrl": "",
        "oauthResourceInAzRequest": "none",
        "oauthAssertionKidType": "",
        "oauthAssertionAud": "",
        "oauthTokenIgnoreSslValidation": false,
        "oauthTokenSslMode": "VERIFY_CA",
        "oauthTokenCertificateAlias": "",
        "oauthTokenRequestCtxBase64": "",
        "oauthTokenReadTimeoutMillis": "30000",
        "oauthTokenConnectTimeoutMillis": "30000",
        "skewTimeSecs": "0"
```

```
}
```

NOTE
To refer to the Custom Provider APIs which can be used to perform CRUD operations in IDSP, see here.

Configuring Custom Provider - providerProperties

A custom provider uses the providerProperties key-value pairs to define a provider-specific behavior. There are two property types:

- **Custom**: All custom properties are published into the providerContext of the SPI request and are consumed by the custom SPI.
- **Well-known**: All well-known properties begin with the **spi.** prefix, representing the directives to be performed by IDSP when sending requests to and receiving responses from the custom SPI.

The following table lists the providerProperties that must be configured to support the use of custom claims:

<table>
<tr><th>Type</th><th>providerContext</th><th>Description</th></tr>
<tr><td>Custom</td><td>key: <custom-key>
value: <custom-value></td><td>Publishes a custom key-value pair into the ProviderContext of the custom SPI. The value in a key-value pair supports dynamic substitution using the ${keyword} notation. For information, see Dynamic Environment Substitution.</td></tr>
<tr><td>Well-known</td><td>key: spi.description
value: provider description</td><td>This value contains the description of the provider.</td></tr>
<tr><td>Well-known</td><td>key: spi.discover.capabilities
value: CSV list of capabilities</td><td>Specifies the capabilities that are supported by the custom provider using multiple, comma-delimited placeholders.
This enables Admin UI to recognize configured custom provider as supporting custom claims.
<table>
<tr><th>Capability</th><th>Description</th></tr>
<tr><td>customclaim</td><td>SPI supports a custom claim getter exposed through the "/CustomClaimGetter" API.</td></tr>
</table>
</td></tr>
<tr><td>Well-known</td><td>key: spi.customclaim.token.types
value: CSV list of token types</td><td>Informs the token generator of which token types should retrieve claims from the custom claims SPI provider.
The following CSV lists indicate the supported token types
• IT (OIDC id token)
• USERAT (OIDC user access token)
• CLIENTAT (OIDC client-only access token)
• USERINFO (OIDC user info)
• SAMLASSERTION (SAML User Assertion)</td></tr>
</table>

Well-known	**key**: spi.customclaim.cache.enabled **value**: true | false	Cache is disabled if this property is not defined or contains a value other than "true." Caching results from a custom claim SPI provider that is done based on <AppId>-<TokenType> leverages distributed cluster-wide Hazelcast cache.

Configuring Custom Provider - headerProperties

A custom provider uses providerHeaders in a key-value pair format to define provider-specific headers to be sent to the custom SPI. Similar to "providerProperties", the headers definitions also supports dynamic substitution using the ${keyword} notation. For information, see Dynamic Environment Variable Substitution.

Configuring Application/Tenant settings to support Custom Claims

After creating the CustomProvider in IDSP, you are required to configure custom claim related property either at the tenant level or at the application level.

- When configured at the ***tenant level*** (property name- **customClaimProviderToBeUsed**), this update is applied universally to all applications within the specified tenant.
- When configured at the ***application level*** (property name- **customClaimProviderToUse**) overrides the configuration at the tenant level for that particular application, when the token gets generated.

Application Configuration to support Custom Claims

You can configure the application property in IDSP to set a custom claim SPI Provider either using the Admin Console or by invoking the Application API.

- ***Configuring the "custom claim provider" application property using the Admin Console***:
  Follow these steps:
  a. Login to the Admin Console.
  b. Navigate to **Applications** > **General**.
  c. Under the "***Additional Configuration"*** section, configure the "***Custom Claim Provider"*** property.
  The **Claim Provider** configuration is also covered in the Admin Console - Claim Provider setting.
- ***Configuring the "customClaimProviderToUse" application property by invoking the Application API endpoint***.
  Follow this step:
  a. Invoke the following API:

```
POST https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/Apps
```

Sample Application Request payload

The value of the "***customClaimProviderToUse***" property is a GUID of the "providerId" that can be found in the response of the ".../admin/v1/CustomProviders" API. In this sample, look for the application property, "customClaimProviderToUse": "2cebd036-8ebe-4fe9-8752-5d6ca7ec5c12" in the request.

```
{
        "status": "active",
        "name": "DemoClient",
        "description": "Demo client to perform tenant level operations for demo purpose.",
        "clientType": "TRUSTED",
        "allowedOpenIDScopes": [
            "urn:iam:at:groups",
            "address",
            "urn:iam:it:groups",
            "openid",
```

```
        "profile",
        "urn:iam:scim:core",
        "urn:iam:scim:ssp",
        "groups",
        "urn:iam:scim:enterprise",
        "urn:iam:scim:credentials",
        "urn:iam:scim:custom",
        "phone",
        "offline_access",
        "email"
    ],
    "redirectURIs": [
        "https://example.com/sample-rp/oidc/callback",
        "https://example.com/sample-rp/saml",
        "https://example.com/sample-rp/oidc"
    ],
    "allowedGrantTypes": [
        "implicit",
        "refresh_token",
        "client_credentials",
        "urn:ietf:params:oauth:grant-type:device_code",
        "authorization_code",
        "sts_confidential",
        "urn:ietf:params:oauth:grant-type:jwt-bearer"
    ],
    "flowURL": null,
    "deviceCodeFlowURL": null,
    "userInfoEndpointResponseFormat": "PLAIN_JSON",
    "skipIssuerAudienceForIT": false,
    "skipEmailForIT": false,
    "idpDiscoveryEnabled": "true",
    "zeroFootPrint": false,
    "softMFAEnabled": false,
    "delegatedAuthentication": false,
    "autoPostToFlowURL": false,
    "userTokenSubAttributeMappingName": "user_loginid",
    "supportedJoseHeaderParams": null,
    "claims": [
        {
            "name": "$[idProfile.groups]",
            "target": "it,at",
            "value": "$[idProfile.groups]"
        },
        {
            "name": "${idProfile.groups}",
            "target": "it,at",
            "value": "${idProfile.groups}"
        },
        {
            "name": "$[userGroups.all_groups]",
            "target": "it,at",
            "value": "$[userGroups.all_groups]"
        },
```

```
    {
        "name": "${isMemberofGroups_XXX}",
        "target": "it,at",
        "value": "${userGroups.XXX}"
    },
    {
        "name": "${isMemberofGroups_admin}",
        "target": "it,at",
        "value": "${userGroups.admin}"
    },
    {
        "name": "userData",
        "target": "it,at",
        "value": "urn:iam:authn:userData"
    },
    {
        "name": "${userGroups.all_groups}",
        "target": "it,at",
        "value": "${userGroups.all_groups}"
    }
],
"customClaimProviderToUse": "2cebd036-8ebe-4fe9-8752-5d6ca7ec5c12",
"metadata": [
    {
        "name": "org",
        "value": "demoOrg"
    },
    {
        "name": "urn:iam:app:allowLoginHint",
        "value": "true"
    }
],
"allowedOperations": [
    "itgroups",
    "atgroups",
    "introspect"
],
"secondaryAudiences": null,
"assertionVerificationCertAlias": null,
"appIcon": null,
"secret": "187a27ad-eb94-40a5-8593-559c5523c7f7",
"clientId": "f35fa6ae-2fa8-4d67-8d45-a4218563f82a",
"itEncryptionTarget": "NONE",
"itEncryptionCertAlias": null,
"userInfoEncryptionCertAlias": null,
"samlEntityId": "example.com",
"samlAcsUrl": "example.com/sample-rp/saml/acs",
"samlEnableSingleLogout": false,
"samlNameIdFormat": "unspecified",
"samlVerifyRequestSignature": false,
"samlVerifyCertAlias": null,
"samlEncryptSamlResponse": false,
"samlEncryptCertAlias": null,
```

```
            "samlSignResponse": true,
            "samlSignAssertion": true,
            "samlSignatureWithKeyInfo": false,
            "samlIdpInitiatedRelaystateMapping": {},
            "samlAssertionClaims": [],
            "samlUserTokenSubAttributeMappingName": null,
            "skewTimeSecs": 0,
            "passwordAuthoritativeSource": "remote",
            "mitmProtectionLevel": null,
            "idStoreToUse": "internal,9d82dd53-9fcc-49f0-a163-568a1b1265f3",
            "samlEffectiveIdpInitiatedUrlWithEntityId": "https://example.com",
            "samlEffectiveIdpInitiatedUrlWithAppId": "https://example.com/default/saml/v1/idp/login?
    appId=d8eb7dcd-2b69-4e0b-ad99-0af22797342c",
            "samlEffectiveSpAcsUrlWithEntityId": "https://example.com",
            "samlEffectiveSpAcsUrlWithAppId": "https://example.com/default/saml/v1/sp/acs?
    sp=d8eb7dcd-2b69-4e0b-ad99-0af22797342c",
            "samlEffectiveSpAcsUrlWithAppName": "https://example.com/default/saml/v1/sp/acs?sp=DemoClient",
            "homeUrl": "https://democlient/homeUrl?p1=n1&p2=n2&p3=n3",
            "errorUrl": "",
            "logoutUrl": "https://democlient/logouturl",
            "postLogoutUrl": "https://democlient/postlogouturl",
            "isSamlApp": true,
            "isOidcApp": true,
            "isLauncherApp": true,
            "requireAccessControl": "false"
        }
```


The **customClaimProviderToUse** configuration is also covered in the Applications API.

Tenant Configuration to support Custom Claims

Configure the tenant setting in IDSP to set a custom claim SPI Provider using the IDSP ".../admin/v1/Configs" API, or using the Admin Console.

- ***Configuring the "custom claim provider"setting using the Admin Console***:
  Follow this step:
  a. Login to the Admin Console.
  b. Navigate to the **Settings** > **Service Configuration**.
  c. Under the "***Risk and Custom Service Provider Settings"*** section, configure the "***Claim Provider"*** property.
  The **Claim Provider** tenant setting information is also covered in the Admin Console - Claim Provider setting.
- ***Configuring the "customClaimProviderToBeUsed" tenant setting by invoking the Config API endpoint***.
  Follow this step:
  a. Invoke the following API:
  ```
  PATCH https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/Configs
  ```
  Sample Apps Request payload: The value of this property is a GUID of the "providerId" that can be found in the response of the ".../admin/v1/CustomProviders" API.
  ```
  {
     "name": "customClaimProviderToBeUsed",
     "value": "<GUID_of_the_providerId>"
  }
  ```
  The **customClaimProviderToBeUsed** configuration is also covered in Settings.

Creating a Sample Custom Claim Provider

The following IDSP Custom Provider API is used to create the Sample Provider:

```
POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders
```

NOTE

- The custom claims SPI is invoked for various token types, including it, userat, clientat, userinfo, and samlassertion. This is specified by the "***spi.customclaim.token.types***" property under the "***providerProperties***," which defines the token types for which custom claims are generated.
- The presence of the "***spi.discover.capabilities***" property under the "***providerProperties***" signifies that this is a custom claim SPI.
- The URL specified by the ***"url": "https://<custom-provider-URL>"*** property in the example must be accessible from the IDSP (SSP) pod.
- Provider's Header and Property configurations show examples of dynamically substituted expressions using appMetadata, idProfile, and userGroups contexts.

```
{
        "providerName": "DemoClientCustomClaimProvider",
        "version": "1.0",
        "apiKey": "mscTWk5ZchgoMN0+hHKL20/5e16Ar5sPR2K3rDuQX53+Z02i/6MiUFtDRyynBqLI9YOBS7lqIuPSXT/
uBHrLZSmgpDwSmf8HWamLWUL3Wf5D9ylRI9WgHzNnq2EOwzUOLbehv02ZXODxxjITqXXT8Yz10ioIk9DazsfSjXMfHcX0vXLrVlX4jW7HeCuqqV0q9S83Vv
+VT24OAVysFJjcE/xNib4n6IgAWQSQSzcA5Q33SbPmJXfQm2TglqL/
ZiF24UmdoO9i0qrrUcxGUO2f2nKf06J4sdPiSebKja0dDTPZfWdigwTF82tl3WwSr0iSnrBi2yYWQuWdC3T
+BqyLYwDbQJehfRu7mlYfbZPDqC3w0NaZdnfAzKw==",
        "certAlias": "",
        "url": "https://<custom-provider-URL>",
        "providerHeaders": [
            {
                "name": "titleFromIdProfileHeader",
                "value": "${idProfile.title}"
            },
            {
                "name": "groupsHeader",
                "value": "${userGroups.all_groups}"
            },
            {
                "name": "customHeader",
                "value": "custom1"
            }
        ],
        "providerProperties": [
            {
                "name": "spi.discover.capabilities",
                "value": "customclaim"
            },
            {
                "name": "spi.customclaim.token.types",
                "value": "it,userat,clientat,userinfo,samlassertion"
            },
            {
                "name": "groupsProviderProperty",
                "value": "${userGroups.all_groups}"
            },
            {
                "name": "orgFromAppMetadataProviderProperty",
```

```
                "value": "${appMetadata.org}"
            },
            {
                "name": "customProviderProperty",
                "value": "custom1"
            }
        ],
        "origin": "tenant",
        "ignoreSSLValidation": true,
        "providerId": "e82c5379-4437-4e53-b461-a35fd34dab89",
        "sslMode": "VERIFY_CA",
        "spiReadTimeoutMillis": "60000",
        "spiConnectTimeoutMillis": "60000",
        "requestCtxBase64": "",
        "clientId": "",
        "clientSecretAlias": "",
        "oauthScopes": "",
        "securityMethod": "apikey",
        "oauthTokenUrl": "",
        "oauthDiscoveryUrl": "",
        "oauthResourceInAzRequest": "none",
        "oauthResourceInAzRequestCsv": "",
        "oauthAssertionKidType": "",
        "oauthAssertionAud": "",
        "oauthTokenIgnoreSslValidation": false,
        "oauthTokenSslMode": "VERIFY_CA",
        "oauthTokenCertificateAlias": "",
        "oauthTokenRequestCtxBase64": "",
        "oauthTokenReadTimeoutMillis": "10000",
        "oauthTokenConnectTimeoutMillis": "10000",
        "skewTimeSecs": "0"
    }
```

Custom Claims SPI Execution Payload Examples

Custom Claim Provider SPI for SAML Identity Assertion

Custom Claim Provider SPI request for SAML Identity Assertion:

```
{
    "subject": "edunn",
    "rawType": "XML",
    "tokenType": "SAMLASSERTION",
    "clientContext ": [
        {
            "key ": "transactionIdentifier ",
            "value ": "abcd-tid-gga323a24 "
        }
    ],
    "ahContext ": [
        {
            "key ": "tid ",
            "value ": "b6f84cf6-1abf-495c-8e57-d62b704e1f0f "
        },
```

```
        {
            "key ": "tname ",
            "value ": "tenant120939 "
        },
        {
            "key ": "clientIp ",
            "value ": "10.0.0.0 "
        },
        {
            "key ": "userIp ",
            "value ": "10.0.0.0 "
        },
        {
            "key ": "txnId ",
            "value ": "fdc341ab-f65f-430f-90ca-5ddc9b1ffa22 "
        }
        {
            "key ": "locale ",
            "value ": null
        }
    ],
    "providerContext ": [
        {
            "key ": "app ",
            "value ": "test "
        },
        {
            "key ": "region ",
            "value ": "us-west "
        }
    ]
}
```


Custom Claim Provider SPI response for SAML Identity Assertion:


```
{
    "msg": "Sample Custom Claim SPI: get claims for user edunn",
    "status": "success",
    "clientRespContext": [],
    "claimListMva": {
        "x-mva-allowed-operations": [
            "buy",
            "sell"
        ],
        "x-mva-roles": [
            "Purchaser",
            "Seller",
            "Approver"
        ],
        "x-mva-authority": [
            "level500"
        ]
    },
    "claimListSva": {
```

```
        "x-sva-authority": "level500",
        "x-sva-allowed-operations": "buy, sell",
        "x-sva-roles": "Purchaser, Seller, Approver"
    },
    "claimListRaw": [
        "<custom1:Attribute Name=\"custom1\" NameFormat=\"urn:oasis:names:tc:SAML:2.0:attrname-format:basic\">
 <custom1:AttributeValue xmlns:xsi=\""http://www.w3.org/2001/XMLSchema-instance%5C" scope="external">http://
www.w3.org/2001/XMLSchema-instance\" xsi:type=\"xsd:string\">custom-value-1</custom1:AttributeValue> </
custom1:Attribute>",
        "<custom2:Attribute Name=\"custom2\" NameFormat=\"urn:oasis:names:tc:SAML:2.0:attrname-format:basic\">
 <custom2:AttributeValue xmlns:xsi=\""http://www.w3.org/2001/XMLSchema-instance%5C" scope="external">http://
www.w3.org/2001/XMLSchema-instance\" xsi:type=\"xsd:string\">custom-value-2</custom2:AttributeValue> </
custom2:Attribute>",
        "<custom3:Attribute xmlns:custom3=\""http://custom3/" scope="external">http://custom3\" Name=
\"custom3\" NameFormat=\"urn:oasis:names:tc:SAML:2.0:attrname-format:basic\"> <custom3:AttributeValue
 xmlns:xsi=\""http://www.w3.org/2001/XMLSchema-instance%5C" scope="external">http://www.w3.org/2001/XMLSchema-
instance\" xsi:type=\"xsd:string\">custom-value-3</custom3:AttributeValue> </custom3:Attribute>",
        "<custom4:Attribute Name=\"custom4\" NameFormat=\"urn:oasis:names:tc:SAML:2.0:attrname-format:basic\">
 <custom4:AttributeValue xmlns:xsi=\""http://www.w3.org/2001/XMLSchema-instance%5C" scope="external">http://
www.w3.org/2001/XMLSchema-instance\" xsi:type=\"xsd:string\">custom-value-4</custom4:AttributeValue> </
custom4:Attribute>",
        "<ns2:Attribute Name=\"habilitations\" NameFormat=\"urn:oasis:names:tc:SAML:2.0:attrname-
format:basic\"><ns2Attribute Code=\"MyPassword\" Libelle=\"MYPASSWORD\" Name=\"myPassword\" NameFormat=
\"urn:oasis:names:tc:SAML:2.0:attrname-format:basic\"><ns2:AttributeValue>Null^Null</ns2:AttributeValue>\n</
ns2Attribute>\n<ns2Attribute Code=\"A-000009999\" Libelle=\"A-000009999\" Name=\"A-000009999\" NameFormat=
\"urn:oasis:names:tc:SAML:2.0:attrname-format:basic\"><ns2:AttributeValue>D-000009999^D-000009999</
ns2:AttributeValue></ns2Attribute>\n<ns2Attribute Code=\"A-0000001312\" Libelle=\"BULK_APPROVALS
\" Name=\"A-0000001312\" NameFormat=\"urn:oasis:names:tc:SAML:2.0:attrname-format:basic
\"><ns2:AttributeValue>Null^Null</ns2:AttributeValue></ns2Attribute></ns2:Attribute>"
    ],
    "claimListRawNs": {
        "custom1": ""https://custom1/">https://custom1",
        "custom2": ""https://custom2/">https://custom2",
        "ns2": ""https://ns2/">https://ns2"
    }
}
```

Sample of Resulting SAML Identity Assertion

```
#<?xml version="1.0" encoding="UTF-8"?>
<saml2p:Response Destination="https://example.com/sample-rp/saml/acs"
 ID="_d8c05c10-5a44-4a5e-951d-14cc57dfd158" InResponseTo="_756b9b81a2a794b14fd9972e141ad188"
 IssueInstant="2024-02-14T18:48:57.203Z" Version="2.0" xmlns:custom1="https://custom1" xmlns:custom2="https://
custom2" xmlns:ns2="https://ns2" xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
    <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity"
 xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">example.com/default</saml2:Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
            <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
            <ds:Reference URI="#_d8c05c10-5a44-4a5e-951d-14cc57dfd158">
                <ds:Transforms>
                    <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
```

```
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces PrefixList="xsd" xmlns:ec="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
                    </ds:Transform>
                </ds:Transforms>
                <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
                <ds:DigestValue>R/2wHcpUmpW8+ODz1Rx81EHTgP1u+1ej/Ona13i0syY=</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>KLU3d591P79usJsQp7hKqCkwQt49iSCb3+8tJN8PHSIbKIdxuJrZ6UD3tdY6lIHXOCVaagT01Mje&#xd;
YHp5J+aKxhxGviiLjfsoZSXzZ6OJyQVzAb5+0cyWWwTV6I5qWG+19DHk4O7KRKxk/PR83skQRxhr&#xd;
UfbRBJ5g/kSfWAgzqmOWOSXjMG2tLkQsEM/ODfJf7CNHEwIA1P2iP7b3Qgtd2dY9AXY/lGOkJuc2&#xd;
vBMSxX0jnbdMXbt5M7yqTAftZQglxcGvQz/OnUH1U0uJWo8VZ9uY/tlAOc3O7etrDaUkdBM57y8i&#xd;
+5nMEdVE9OFibNblLABkW7Yr9Ymm1cw54UOKTA==</ds:SignatureValue>
    </ds:Signature>
    <saml2p:Status>
        <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
    </saml2p:Status>
    <saml2:Assertion ID="_5b6c39e1-4de8-43cd-80e6-016fb89ce9d7" IssueInstant="2024-02-14T18:48:57.234Z"
 Version="2.0" xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsd="http://www.w3.org/2001/
XMLSchema">
        <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">https://example.com/default</
saml2:Issuer>
        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:SignedInfo>
                <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
                <ds:Reference URI="#_5b6c39e1-4de8-43cd-80e6-016fb89ce9d7">
                    <ds:Transforms>
                        <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                            <ec:InclusiveNamespaces PrefixList="xsd" xmlns:ec="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
                        </ds:Transform>
                    </ds:Transforms>
                    <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
                    <ds:DigestValue>Wm9n9OYakVrgUoQ1Tk9pAOlwnXnq2Wuu1yF5B68jYuE=</ds:DigestValue>
                </ds:Reference>
            </ds:SignedInfo>

 <ds:SignatureValue>ECv7SwTX96Fo7/2nZJI5FB5Vtxz9kGLykEw945QrnvOKTVMiNcvGKT9h0XlzZNfVGxgEv27AYlFg&#xd;
xgt+twEWmQhaURxVxh+Bo3DcQAp2GKt/rZyjnCeDBI6yz1l5KRjZVvbEqxrV/4oS9QSs9qtpAfth&#xd;
IRWreaUM2wSXctUC/mERGdG2Ey50eLO/Y8dIPO1gKxjqBgzMEIMxRMZ+G2r2UN+No/fTt0T9pWu5&#xd;
fy/DyT2SzDk8GxbGKIL4CS2hLonWT7qvr/xJ775+Se/VH3vhm7sO5nV30uyRU7MTl6irpejL1pqI&#xd;
ft4aIUjVRFi4kcIU8ZfoWdhAovNB7kvOiXSx1A==</ds:SignatureValue>
        </ds:Signature>
        <saml2:Subject>
            <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">admin</saml2:NameID>
            <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
                <saml2:SubjectConfirmationData InResponseTo="_756b9b81a2a794b14fd9972e141ad188"
 NotOnOrAfter="2024-02-15T02:48:57.207Z" Recipient="https://example.com/sample-rp/saml/acs"/>
            </saml2:SubjectConfirmation>
        </saml2:Subject>
```

```
        <saml2:Conditions NotBefore="2024-02-14T18:45:57.208Z" NotOnOrAfter="2024-02-14T18:51:57.208Z">
            <saml2:AudienceRestriction>
                <saml2:Audience>https://example.com</saml2:Audience>
            </saml2:AudienceRestriction>
        </saml2:Conditions>
        <saml2:AuthnStatement AuthnInstant="2024-02-14T18:48:57.208Z">
            <saml2:AuthnContext>
                <saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:Password</
saml2:AuthnContextClassRef>
                <saml2:AuthenticatingAuthority>https://example.com/default</saml2:AuthenticatingAuthority>
            </saml2:AuthnContext>
        </saml2:AuthnStatement>
        <saml2:AttributeStatement>
            <saml2:Attribute Name="user_guid" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">4ed30eec-3c1f-4eeb-b4e8-cdb52c2ee01f</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="user_loginid" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">admin</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="user_riskscore" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">0</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="sid" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">uuid-bc170935-f9d7-4513-8354-f57432153e9c</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="${isMemberofGroups_admin}"
 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">true</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="${userGroups.all_groups}" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">Security Administrators1,MyGroup,Admin</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="${isMemberofGroups_XXX}" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">false</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="$[idProfile.groups]" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">Security Administrators1</saml2:AttributeValue>
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">MyGroup</saml2:AttributeValue>
```

```
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">Admin</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="$[userGroups.all_groups]" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">Security Administrators1</saml2:AttributeValue>
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">MyGroup</saml2:AttributeValue>
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">Admin</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="${idProfile.groups}" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">Security Administrators1,MyGroup,Admin</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="userData" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">userData</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="x-mva-allowed-operations" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">buy</saml2:AttributeValue>
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">sell</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="x-mva-roles" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">Purchaser</saml2:AttributeValue>
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">Seller</saml2:AttributeValue>
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">Approver</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="x-mva-authority" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">level500</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="x-sva-authority" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">level500</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute Name="x-sva-allowed-operations" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">buy, sell</saml2:AttributeValue>
            </saml2:Attribute>
```

```
            <saml2:Attribute Name="x-sva-roles" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
 format:basic">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="xsd:string">Purchaser, Seller, Approver</saml2:AttributeValue>
            </saml2:Attribute>
            <custom1:Attribute Name="custom1" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
                <custom1:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="xsd:string">custom-value-1</custom1:AttributeValue>
            </custom1:Attribute>
            <custom2:Attribute Name="custom2" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
                <custom2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="xsd:string">custom-value-2</custom2:AttributeValue>
            </custom2:Attribute>
            <custom3:Attribute Name="custom3" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
  xmlns:custom3="http://custom3">
                <custom3:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="xsd:string">custom-value-3</custom3:AttributeValue>
            </custom3:Attribute>
            <ns2:Attribute Name="habilitations" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
 format:basic">
                <ns2Attribute Code="MyPassword" Libelle="MYPASSWORD" Name="myPassword"
  NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
                    <ns2:AttributeValue>Null^Null</ns2:AttributeValue>
                </ns2Attribute>
                <ns2Attribute Code="A-000009999" Libelle="A-000009999" Name="A-000009999"
  NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
                    <ns2:AttributeValue>D-000009999^D-000009999</ns2:AttributeValue>
                </ns2Attribute>
                <ns2Attribute Code="A-0000001312" Libelle="BULK_APPROVALS" Name="A-0000001312"
  NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
                    <ns2:AttributeValue>Null^Null</ns2:AttributeValue>
                </ns2Attribute>
            </ns2:Attribute>
        </saml2:AttributeStatement>
    </saml2:Assertion>
</saml2p:Response>
```

Custom Claim Provider SPI for OIDC Identity Token

Custom Claim Provider SPI Request for OIDC Identity Token:

```
{
    "subject": "edunn",
    "rawType": "JSON",
    "tokenType":"IT",
    "clientContext ": [
        {
            "key ": "transactionIdentifier ",
            "value ": "abcd-tid-gga323a24 "
        }
    ],
    "ahContext ": [
        {
            "key ": "tid ",
```

```
            "value ": "b6f84cf6-1abf-495c-8e57-d62b704e1f0f "
        },
        {
            "key ": "tname ",
            "value ": "tenant120939 "
        },
        {
            "key ": "clientIp ",
            "value ": "10.0.0.0 "
        },
        {
            "key ": "userIp ",
            "value ": "10.0.0.0 "
        },
        {
            "key ": "txnId ",
            "value ": "fdc341ab-f65f-430f-90ca-5ddc9b1ffa22 "
        },
        {
            "key ": "clientTxnId ",
            "value ": "TSRCAAK12_ldapAuthenticateAllowUser "
        },
        {
            "key ": "locale ",
            "value ": null
        }
    ],
    "providerContext ": [
        {
            "key ": "app ",
            "value ": "test "
        },
        {
            "key ": "region ",
            "value ": "us-west "
        }
    ]
}
```


Custom Claim Provider SPI Response for OIDC Identity Token:


```
{
    "msg": "Sample Custom Claim SPI: get claims for user edunn",
    "status": "success",
    "clientRespContext": [],
    "claimListMva": {
        "x-mva-allowed-operations": [
            "buy",
            "sell"
        ],
        "x-mva-roles": [
            "Purchaser",
            "Seller",
            "Approver"
```

```
        ],
        "x-mva-authority": [
            "level500"
        ]
    },
    "claimListSva": {
        "x-sva-authority": "level500",
        "x-sva-allowed-operations": "buy, sell",
        "x-sva-roles": "Purchaser, Seller, Approver"
    },
    "claimListRaw": [
        "\"customSva\": \"custom1\"",
        "\"customMva\": [\"custom1\",\"custom2\"]",
        "\"habilitations\": [    {      \"ou\": \"A-0000001312\",      \"libelle\": \"BULK_APPROVALS\",
 \"roles\": []    },    {      \"ou\": \"A-0000001170\",      \"libelle\": \"ITG_MyPassword\",      \"roles\":
 [        {          \"cn\": \"D-0000002524\",          \"libelle\": \"Support\"        }      ]    },    {
     \"ou\": \"A-000009999\",      \"libelle\": \"A-000009999\",      \"roles\": [        {          \"cn\":
 \"D-000009999\",          \"libelle\": \"D-000009999\"        }      ]    }  ]"
    ],
    "claimListRawNs": {}
}
```


Sample of resulting OIDC Identity Token:


```
{
    "at_hash": "DjT1pxJgcVWV9JWsQTvc9Q",
    "sub": "admin",
    "idp_name": "internal",
    "user_dn": "4ed30eec-3c1f-4eeb-b4e8-cdb52c2ee01f",
    "amr": [
        "PWD"
    ],
    "user_loginid": "admin",
    "idp_guid": "internal",
    "iss": <issuer URL>,
    "user_guid": "4ed30eec-3c1f-4eeb-b4e8-cdb52c2ee01f",
    "aal": "AAL1",
    "tid": "19657db7-9efc-4d7e-9c53-19e9895e8ab7",
    "transactionid": "c628b1b4-9c3a-4e1c-beec-f7e54b9d5fa3",
    "sid": "uuid-67ce101a-d1bb-4525-bce6-553f3bb1c218",
    "user_subattrmappingname": "user_loginid",
    "acr": "urn:iam:acr:aal1",
    "x-mva-allowed-operations": [
        "buy",
        "sell"
    ],
    "azp": "f35fa6ae-2fa8-4d67-8d45-a4218563f82a",
    "user_loginidattrmappingname": "user_loginid",
    "auth_time": 1708035999,
    "idp_user_guid": "4ed30eec-3c1f-4eeb-b4e8-cdb52c2ee01f",
    "exp": 1708039599,
    "iat": 1708035999,
    "client_name": "DemoClient",
    "jti": "2c00d5fc-da8e-4c67-9878-6a6d2e8111fe",
    "idp_type": "internal",
```

```
"client_tname": "default",
"x-sva-roles": "Purchaser, Seller, Approver",
"client_tid": "19657db7-9efc-4d7e-9c53-19e9895e8ab7",
"ver": "1.0",
"customSva": "custom1",
"user_universalid": "abcd",
"x-sva-authority": "level500",
"habilitations": [
    {
        "ou": "A-0000001312",
        "libelle": "BULK_APPROVALS",
        "roles": []
    },
    {
        "ou": "A-0000001170",
        "libelle": "ITG_MyPassword",
        "roles": [
            {
                "libelle": "Support",
                "cn": "D-0000002524"
            }
        ]
    },
    {
        "ou": "A-000009999",
        "libelle": "A-000009999",
        "roles": [
            {
                "libelle": "D-000009999",
                "cn": "D-000009999"
            }
        ]
    }
],
"tname": "default",
"customMva": [
    "custom1",
    "custom2"
],
"tok_type": "IT",
"user_riskscore": 66,
"aud": [
    "f35fa6ae-2fa8-4d67-8d45-a4218563f82a",
    "https://example.com/default/"
],
"nbf": 1708035999,
"user_zfp": false,
"client_transactionid": "ssp:c628b1b4-9c3a-4e1c-beec-f7e54b9d5fa3",
"x-mva-roles": [
    "Purchaser",
    "Seller",
    "Approver"
],
```

```
    "tok_source": "AZC",
    "x-sva-allowed-operations": "buy, sell",
    "x-mva-authority": [
        "level500"
    ]
}
```

SPI Request and Response with OIDC Access Token

Custom Claim Provider SPI Request for OIDC Access Token Request:

```
{
    "subject": "edunn",
    "rawType": "JSON",
    "tokenType":"AT",
    "clientContext ": [
        {
            "key ": "transactionIdentifier ",
            "value ": "abcd-tid-gga323a24 "
        }
    ],
    "ahContext ": [
        {
            "key ": "tid ",
            "value ": "b6f84cf6-1abf-495c-8e57-d62b704e1f0f "
        },
        {
            "key ": "tname ",
            "value ": "tenant120939 "
        },
        {
            "key ": "clientIp ",
            "value ": "10.0.0.0 "
        },
        {
            "key ": "userIp ",
            "value ": "10.0.0.0 "
        },
        {
            "key ": "txnId ",
            "value ": "fdc341ab-f65f-430f-90ca-5ddc9b1ffa22 "
        },
        {
            "key ": "clientTxnId ",
            "value ": "TSRCAAK12_ldapAuthenticateAllowUser "
        },
        {
            "key ": "locale ",
            "value ": null
        }
    ],
    "providerContext ": [
        {
            "key ": "app ",
            "value ": "test "
```

```
        },
        {
            "key ": "region ",
            "value ": "us-west "
        }
    ]
}
```

Custom Claim Provider SPI Access Token Response:

```
{
    "msg": "Sample Custom Claim SPI: get claims for user edunn",
    "status": "success",
    "clientRespContext": [],
    "claimListMva": {
        "x-mva-allowed-operations": [
            "buy",
            "sell"
        ],
        "x-mva-roles": [
            "Purchaser",
            "Seller",
            "Approver"
        ],
        "x-mva-authority": [
            "level500"
        ]
    },
    "claimListSva": {
        "x-sva-authority": "level500",
        "x-sva-allowed-operations": "buy, sell",
        "x-sva-roles": "Purchaser, Seller, Approver"
    },
    "claimListRaw": [
        "\"customSva\": \"custom1\"",
        "\"customMva\": [\"custom1\",\"custom2\"]",
        "\"habilitations\": [    {      \"ou\": \"A-0000001312\",      \"libelle\": \"BULK_APPROVALS\",
 \"roles\": []    },    {      \"ou\": \"A-0000001170\",      \"libelle\": \"ITG_MyPassword\",      \"roles\":
 [        {          \"cn\": \"D-0000002524\",          \"libelle\": \"Support\"        }      ]    },    {
     \"ou\": \"A-000009999\",      \"libelle\": \"A-000009999\",      \"roles\": [        {          \"cn\":
 \"D-000009999\",          \"libelle\": \"D-000009999\"        }      ]    }  ]"
    ],
    "claimListRawNs": {}
}
```

Sample of resulting OIDC Access Token:

```
{
    "sub": "admin",
    "idp_name": "internal",
    "user_loginid": "admin",
    "idp_guid": "internal",
    "iss": "https://example.com/default/",
    "user_guid": "4ed30eec-3c1f-4eeb-b4e8-cdb52c2ee01f",
    "tid": "19657db7-9efc-4d7e-9c53-19e9895e8ab7",
```

```
"transactionid": "c628b1b4-9c3a-4e1c-beec-f7e54b9d5fa3",
"sid": "uuid-67ce101a-d1bb-4525-bce6-553f3bb1c218",
"user_subattrmappingname": "user_loginid",
"x-mva-allowed-operations": [
    "buy",
    "sell"
],
"urn:iam:device:sig": "6e8215641c46b50804b89b1c4f479de2d24f12c3d35349cfc699c6ebbc728d5a",
"sub_type": "USER",
"azp": "f35fa6ae-2fa8-4d67-8d45-a4218563f82a",
"user_loginidattrmappingname": "user_loginid",
"scope": "urn:iam:m.me openid",
"idp_user_guid": "4ed30eec-3c1f-4eeb-b4e8-cdb52c2ee01f",
"exp": 1708039599,
"iat": 1708035999,
"client_name": "DemoClient",
"jti": "87ebe08f-2cf4-46e3-92ec-b5888718f600",
"idp_type": "internal",
"client_tname": "default",
"x-sva-roles": "Purchaser, Seller, Approver",
"client_tid": "19657db7-9efc-4d7e-9c53-19e9895e8ab7",
"ver": "1.0",
"customSva": "custom1",
"user_universalid": "abcd",
"x-sva-authority": "level500",
"habilitations": [
    {
        "ou": "A-0000001312",
        "libelle": "BULK_APPROVALS",
        "roles": []
    },
    {
        "ou": "A-0000001170",
        "libelle": "ITG_MyPassword",
        "roles": [
            {
                "libelle": "Support",
                "cn": "D-0000002524"
            }
        ]
    },
    {
        "ou": "A-000009999",
        "libelle": "A-000009999",
        "roles": [
            {
                "libelle": "D-000009999",
                "cn": "D-000009999"
            }
        ]
    }
],
"azpacr": "2",
```

```
    "tname": "default",
    "customMva": [
        "custom1",
        "custom2"
    ],
    "tok_type": "AT",
    "user_riskscore": 66,
    "user_amr": [
        "PWD"
    ],
    "aud": "https://example.com/default/",
    "nbf": 1708035999,
    "user_zfp": false,
    "client_transactionid": "ssp:c628b1b4-9c3a-4e1c-beec-f7e54b9d5fa3",
    "x-mva-roles": [
        "Purchaser",
        "Seller",
        "Approver"
    ],
    "user_aal": "AAL1",
    "tok_source": "AZC",
    "x-sva-allowed-operations": "buy, sell",
    "user_acr": "urn:iam:acr:aal1",
    "x-mva-authority": [
        "level500"
    ]
}
```


SPI Request and Response with OIDC Userinfo

Custom Claim Provider SPI Request for OIDC Userinfo:


```
{
    "subject": "edunn",
    "rawType": "JSON",
    "tokenType":"USERINFO",
    "clientContext ": [
        {
            "key ": "transactionIdentifier ",
            "value ": "abcd-tid-gga323a24 "
        }
    ],
    "ahContext ": [
        {
            "key ": "tid ",
            "value ": "b6f84cf6-1abf-495c-8e57-d62b704e1f0f "
        },
        {
            "key ": "tname ",
            "value ": "tenant120939 "
        },
        {
            "key ": "clientIp ",
            "value ": "10.0.0.0 "
        },
```

```
        {
            "key ": "userIp ",
            "value ": "10.0.0.0 "
        },
        {
            "key ": "txnId ",
            "value ": "fdc341ab-f65f-430f-90ca-5ddc9b1ffa22 "
        },
        {
            "key ": "clientTxnId ",
            "value ": "TSRCAAK12_ldapAuthenticateAllowUser "
        },
        {
            "key ": "locale ",
            "value ": null
        }
    ],
    "providerContext ": [
        {
            "key ": "app ",
            "value ": "test "
        },
        {
            "key ": "region ",
            "value ": "us-west "
        }
    ]
}
```


Custom Claim Provider SPI Response for OIDC Userinfo:


```
{
    "msg": "Sample Custom Claim SPI: get claims for user edunn",
    "status": "success",
    "clientRespContext": [],
    "claimListMva": {
        "x-mva-allowed-operations": [
            "buy",
            "sell"
        ],
        "x-mva-roles": [
            "Purchaser",
            "Seller",
            "Approver"
        ],
        "x-mva-authority": [
            "level500"
        ]
    },
    "claimListSva": {
        "x-sva-authority": "level500",
        "x-sva-allowed-operations": "buy, sell",
        "x-sva-roles": "Purchaser, Seller, Approver"
    },
```

```
    "claimListRaw": [
        "\"customSva\": \"custom1\"",
        "\"customMva\": [\"custom1\",\"custom2\"]",
        "\"habilitations\": [    {      \"ou\": \"A-0000001312\",      \"libelle\": \"BULK_APPROVALS\",
  \"roles\": []    },    {      \"ou\": \"A-0000001170\",      \"libelle\": \"ITG_MyPassword\",      \"roles\":
  [        {          \"cn\": \"D-0000002524\",          \"libelle\": \"Support\"        }      ]    },    {
     \"ou\": \"A-000009999\",      \"libelle\": \"A-000009999\",      \"roles\": [        {          \"cn\":
  \"D-000009999\",          \"libelle\": \"D-000009999\"        }      ]    }  ]"
    ],
    "claimListRawNs": {}
}
```


OIDC Userinfo Sample

Sample of resulting OIDC Userinfo


```
{
    "sub": "admin",
    "idp_name": "internal",
    "user_loginid": "admin",
    "idp_guid": "internal",
    "iss": "https://example.com/default",
    "user_guid": "4ed30eec-3c1f-4eeb-b4e8-cdb52c2ee01f",
    "user_subattrmappingname": "user_loginid",
    "x-mva-allowed-operations": [
        "buy",
        "sell"
    ],
    "user_loginidattrmappingname": "user_loginid",
    "session_ttl_secs": 4203,
    "idp_type": "internal",
    "x-sva-roles": "Purchaser, Seller, Approver",
    "customSva": "custom1",
    "x-sva-authority": "level500",
    "habilitations": [
        {
            "ou": "A-0000001312",
            "libelle": "BULK_APPROVALS",
            "roles": []
        },
        {
            "ou": "A-0000001170",
            "libelle": "ITG_MyPassword",
            "roles": [
                {
                    "cn": "D-0000002524",
                    "libelle": "Support"
                }
            ]
        },
        {
            "ou": "A-000009999",
            "libelle": "A-000009999",
            "roles": [
                {
```

```
                    "cn": "D-000009999",
                    "libelle": "D-000009999"
                }
            ]
        }
    ],
    "customMva": [
        "custom1",
        "custom2"
    ],
    "given_name": "Admin",
    "aud": "f35fa6ae-2fa8-4d67-8d45-a4218563f82a",
    "user_zfp": "false",
    "session_last_access": 1708031692,
    "x-mva-roles": [
        "Purchaser",
        "Seller",
        "Approver"
    ],
    "family_name": "Admin",
    "x-sva-allowed-operations": "buy, sell",
    "x-mva-authority": [
        "level500"
    ]
}
```


Discovery of User Groups by Applications

Learn how applications can discover information about user groups using different approaches.

The following approaches can be used to search for User Groups information by an application:

- Obtaining user groups information by application in the "/oauth/v1/userinfo" API response (OIDC flow)
- Obtaining user groups information by application in Access Token or Identity Token
- Obtaining user groups information by application by configuring the "Custom Claims" property in the application
- Obtaining user groups information by application using the Admin API or SCIM API

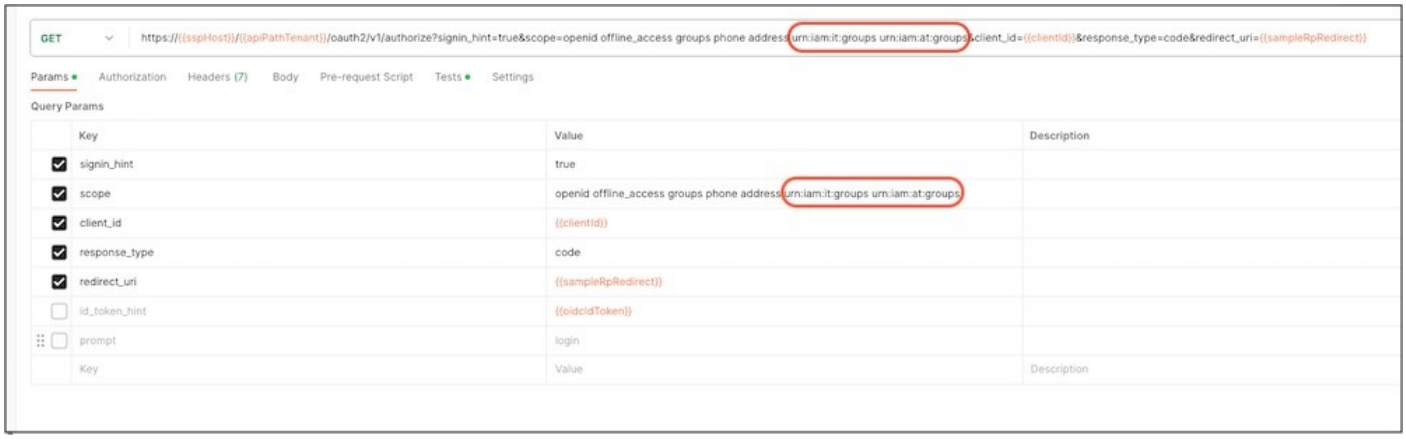
Obtaining user groups information in the "/oauth/v1/userinfo" API response using OIDC flow

This section describes the steps that are required to perform when an application wants to obtain user groups information using the "/oauth/v1/userinfo" API.

You can authorize the application to access a user's groups through the "/oauth2/v1/userinfo" API by modifying the application configuration to incorporate the "groups" scope within the "Allowed Open Id scopes" section. Once this scope is granted, the application can request the "groups" scope, and using the Access Token with that scope, invoke the "/userinfo" API. The response from the "/userinfo" API will include the user's group information. See Userinfo API.

1. From the Admin Console UI, if you want the application to be able to request the "groups" scope, you must perform the following steps:
   a. Navigate to **Applications**, select the app to which you want to configure to obtain user's groups information.
   b. Select the **User Groups (groups)** scope under the **Allowed Open ID Scopes** section as shown in the following image:

2. The following image shows how you can request "groups" scope in the Authorization Code Request using the Postman.



The application invokes the "/oauth2/v1/userinfo" API using an access token containing the "groups" scope, and receives the following userinfo response:


```
{
    "sub": "admin",
    "idp_name": "internal",
    "address": "this is formatted address",
    "user_loginid": "admin",
    "idp_guid": "internal",
```

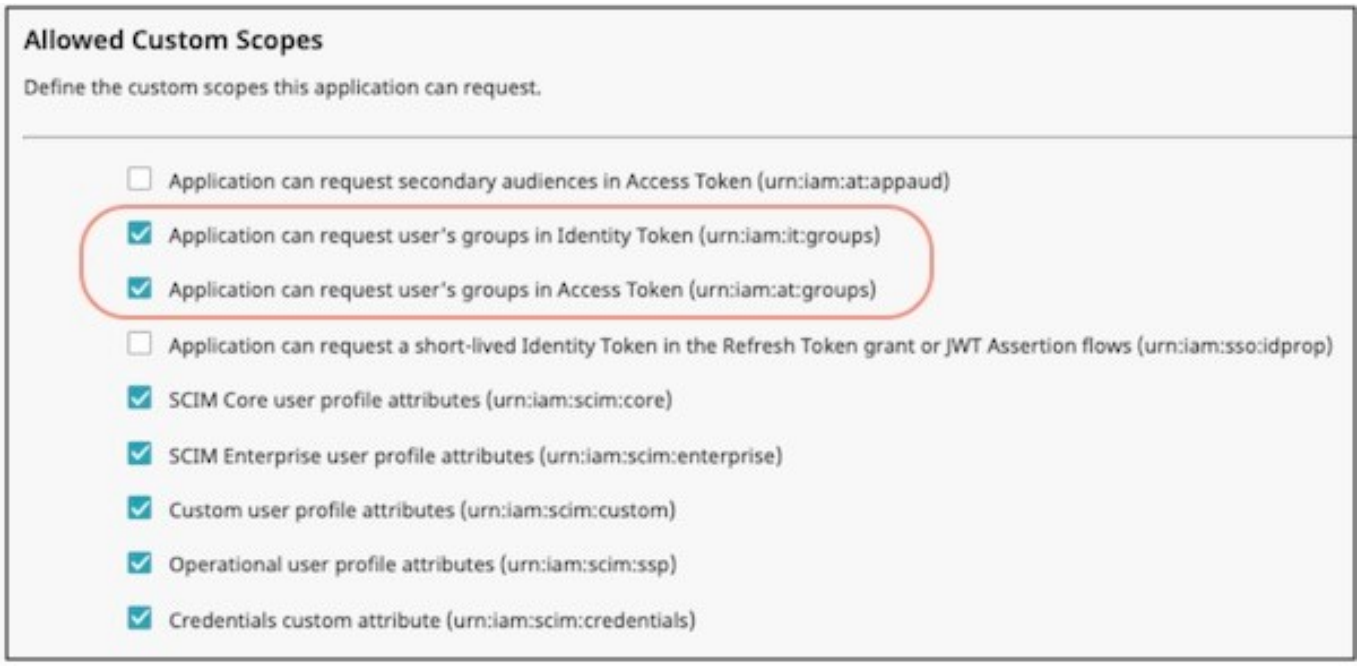
```
    "iss": "https://myssp.example.com/default",
    "groups": [
        "Admin",
        "MyGroup"
    ],
    "phone_number_verified": "true",
    "user_guid": "4ed30eec-3c1f-4eeb-b4e8-cdb52c2ee01f",
    "given_name": "Admin",
    "user_subattrmappingname": "user_loginid",
    "aud": "f35fa6ae-2fa8-4d67-8d45-a4218563f82a",
    "user_zfp": "false",
    "user_loginidattrmappingname": "user_loginid",
    "address_verified": "false",
    "phone_number": "1-617-584-9860",
    "family_name": "Admin",
    "idp_type": "internal"
}
```


Obtaining user groups information by application tokens

This section outlines the necessary steps for an application to obtain user groups information through the application tokens, Access and/or Identity token.

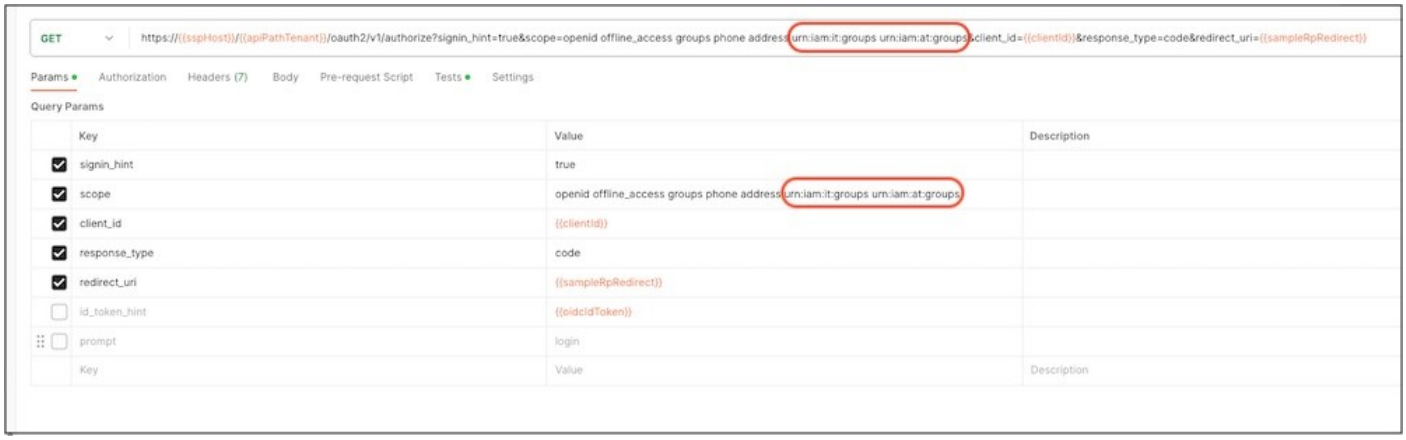
You can authorize the application to obtain user groups information by specifying the "urn:iam:it:groups" and/or "urn:iam:at:groups" scopes in the Identity token and/or Access Token accordingly.

1. To allow these scopes to be used by the application, in the Admin Console UI, perform the following steps:
   a. Navigate to **Applications**, select the app to which you want to configure to obtain user's groups information.
   b. Open the application, navigate to the OAuth/OpenID tab, and under the **Allowed Custom Scopes** section, select the "urn:iam:it:groups" and/or "urn:iam:at:groups" scopes as highlighted in the following image:



2. The application invokes the "/authorize" API requesting for "urn:iam:it:groups" and/or "urn:iam:at:groups" scopes. The user groups are returned in the Identity Token or Access Token respectively.

The following image shows how you can request "groups" scope in the Authorization Code Request using the Postman.



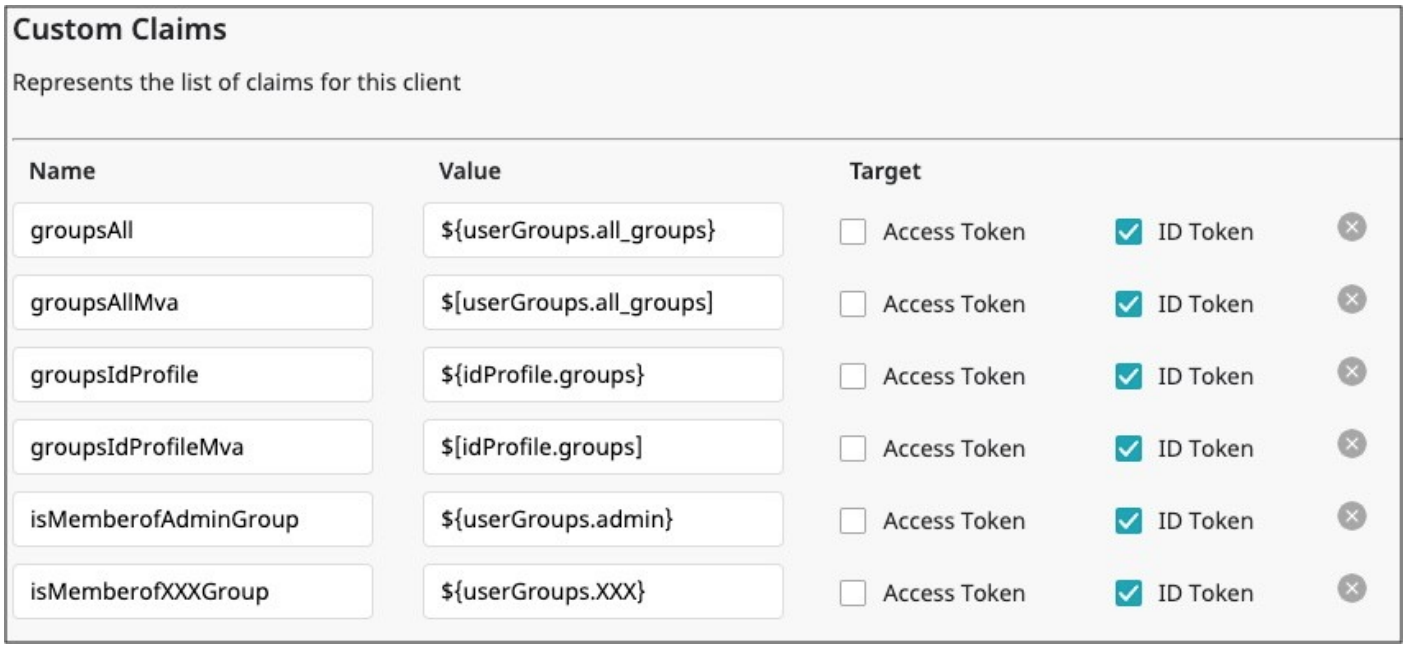
The following example displays the additional user's group information that gets added to the token.

```
"hasgroups": true,
"groups": [
  "MyGroup",
  "Admin"
]
```

Obtaining user's groups information by defining "Custom Claims" in the application

This section outlines the different options in which "Custom Claims" can be configured to contain user's groups information in the application's Access and/or Identity token.

1. In **Applications**, select the app to which you want to configure to obtain user's groups information.
2. In the "Custom Claims" section, create a custom claim with any of the following values as shown in the following image:
   - $[userGroups.all_groups] or $[userGroups.all_groups] - This will be replaced with a list of user's groups.
   - ${userProfile.groups} or $[userProfile.groups] - This will be replaced with a comma separated value (csv) containing the user's groups.



NOTE
Alternatively, you can create a custom claim that indicates whether the user is a member of a specific group by using ${userGroups.<groupname>} value which sets the claim value to true or false accordingly.

Obtaining user's groups information by using the Admin API or SCIM API

This section outlines the APIs that can be used by an application to obtain user's groups information.

- Obtain the Authorized adminAccessToken from the "/authorize" API and use the following API to request user's profile that will include user groups.

  `https://${SSP_FQDN}/<apiPathTenant>/idstore/v1/Users/<userLoginName>`

  NOTE

  - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
  - The default value for `<tenantName>` is "default".
  - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
    For example:
    `https://{{host}}/{{tenantName}}/idstore/v1/Users/<userLoginName>`
    should look similar to:
    `https://example.com/default/idstore/v1/Users/<userLoginName>`
  - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.
- Use the Authorized adminAccessToken obtained from the following APIs to discover user groups by apps.

  `https://${SSP_FQDN}/<apiPathTenant>/scim2/v1/Users?filter=userName=<userLoginid>?attributes=groups (az=adminAccessToken)`

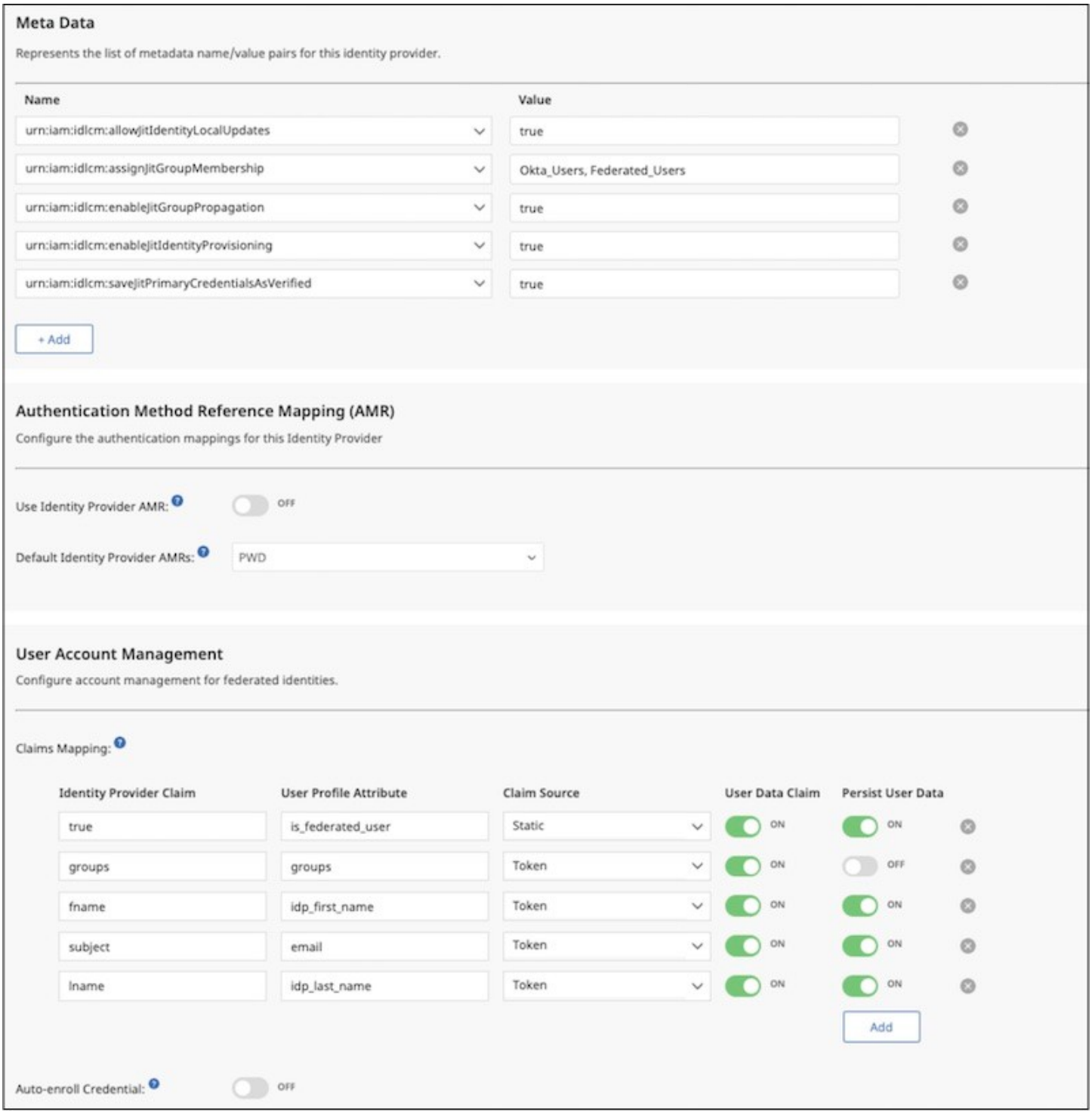
  `https://${SSP_FQDN}/<apiPathTenant>/scim2/v1/Users/?<userId>?attributes=groups (az=adminAccessToken)`

  For more information on these APIs, see ID Store APIs and Managing Users in the Identity Service.

IDP Flow

If a third-party Identity Provider (IDP) is set up to include groups in the user assertion, the Symantec Identity Security Platform IDP configuration allows mapping these groups from the incoming user assertion, referred to as the Identity Provider Claim "groups" in the following image to the universally recognized User Profile Attribute "groups." This attribute is populated by every IDP interaction during the signin ceremony.

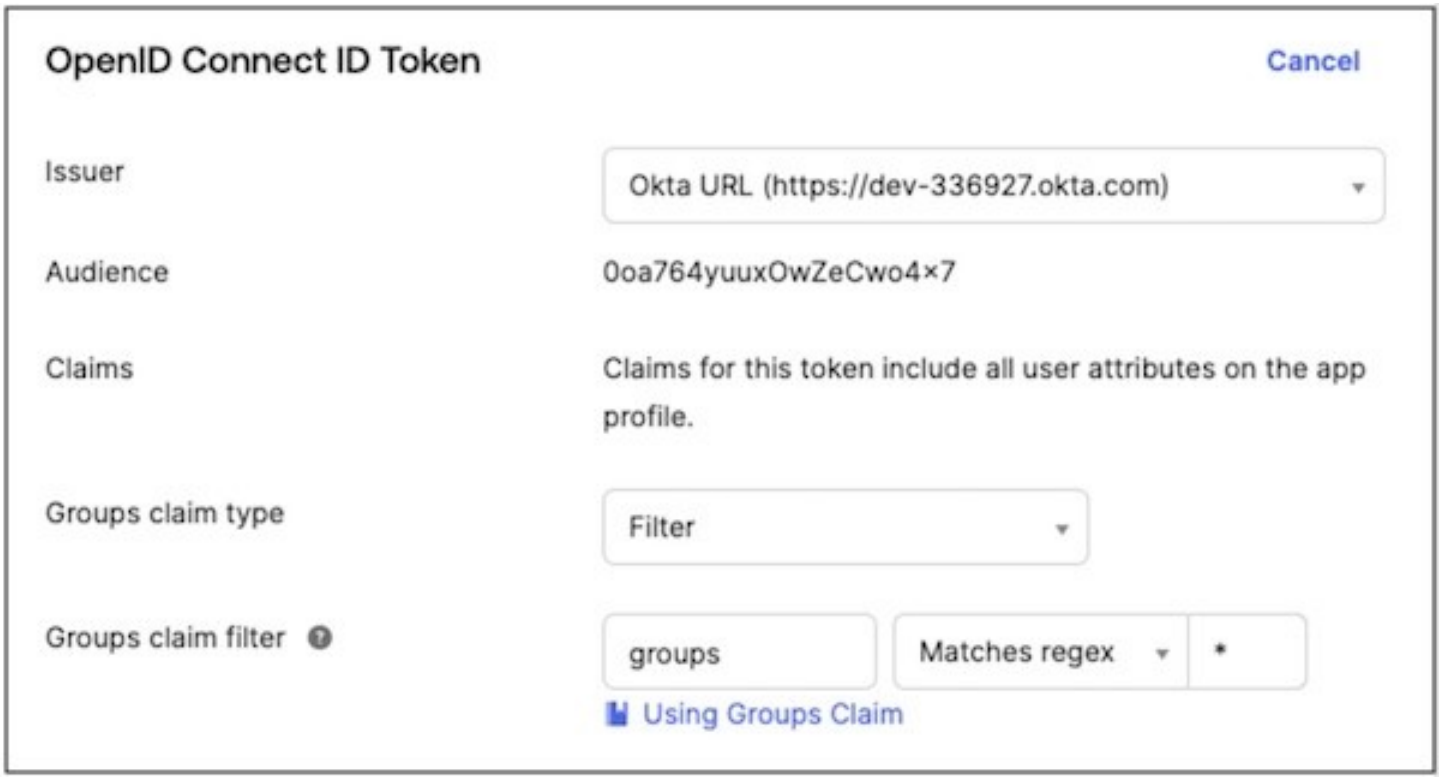
In non Zero-Footprint (ZFP) flows, user groups from the identity assertion are appended to the list of user groups retrieved from the identity store. Refer to each third-party IDP to know how to configure groups to return user groups in the identity assertion.



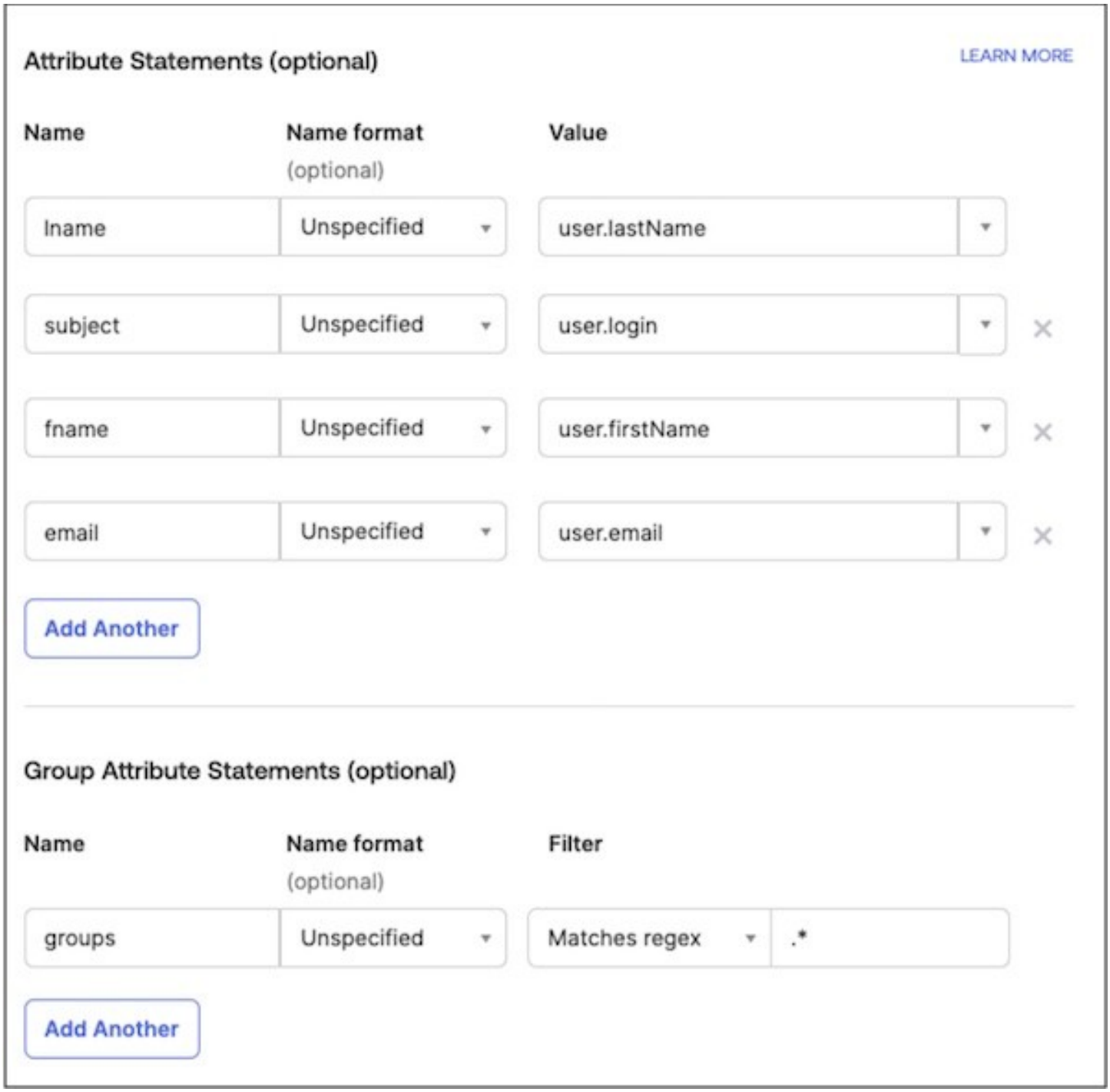
Configuring Third-Party IDP's to include user's groups in the user assertion

The following examples show how different IDP configurations include user's groups information in the user assertion.

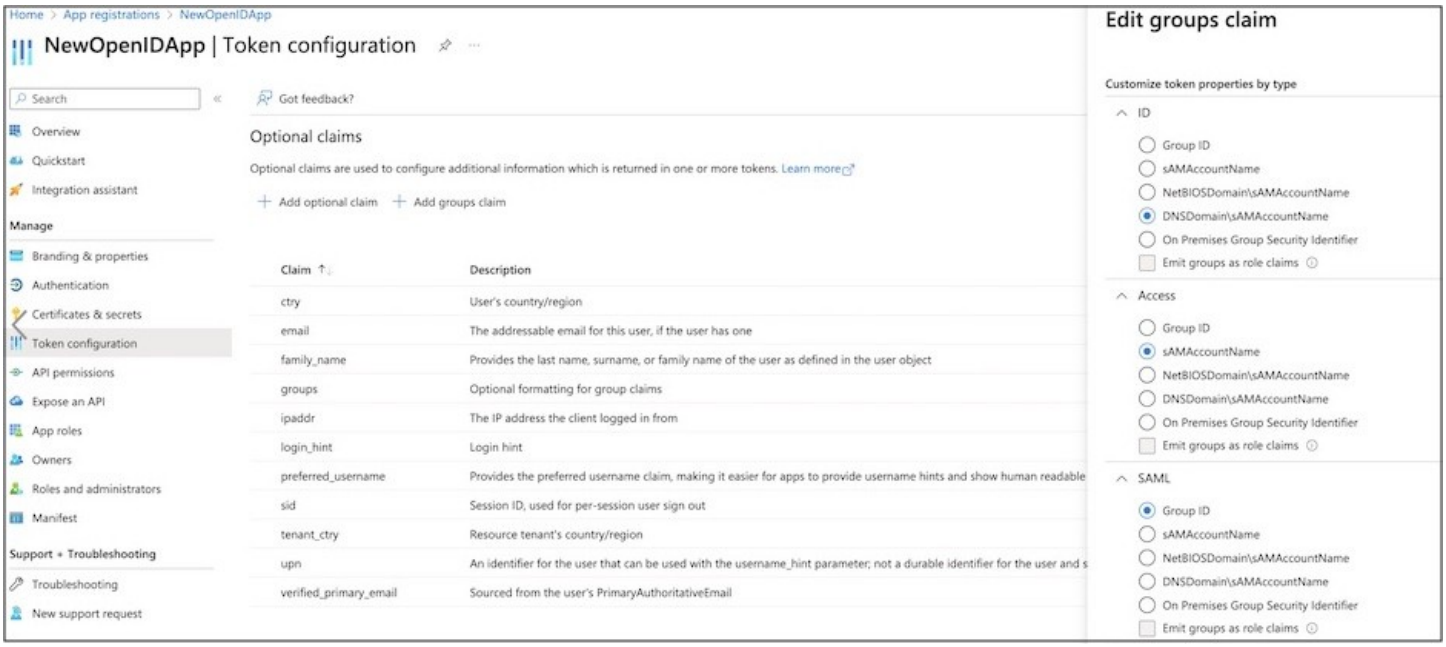
Example of OKTA IDP OpenID Configuration UI that shows how user's groups claims can be configured.



Example of OKTA IDP SAML Configuration UI that shows how user's groups claims can be configured.



Example of Azure Configuration UI that shows how user's groups claims can be configured.



Resource Server Application Flow

Resource Servers host protected resources. This topic covers the provisioning of client applications, the setup of resource server applications, and the definition of associated authentication and authorization policies, roles, and scopes, including the OIDC (OpenID Connect) flow.

Resource Servers are essential components in an OAuth 2.0 and OpenID Connect ecosystem, responsible for hosting protected resources and responding to authenticated requests from client applications. Proper management of Resource Server Applications involves defining their scopes, roles, and access policies to ensure secure and controlled access to sensitive data and functionalities. This topic provides a detailed walkthrough of this functionality.

This topic includes the following sections to secure a Resource Server using Symantec Identity Security Platform :

- Aquire Access Token
- Client App Provisioning
- Resource Server Provisioning
- Client App Flow Using Resource Server
- OIDC Flow using Resource Server

Tenant Setting Configuration

Configure this tenant setting, "**persistUserConsentGivenForResourceApps**" to true, if the consent given by the user during the authorization flow for the resource apps needs to be persisted. By default, this setting is set to **false**. This setting is also covered in the See Managing Settings page.

Get Access Token

- Acquire Client-only Access Token by using the "/token" API:

  ```
  POST https://{{sspHost}}/{{apiPathTenant}}/oauth2/v1/token
  ```

  Acquire Client-only Access Token- Response

  Response:

  ```
  {
  ```

```
"access_token": "eyJ4NXQjUzI1NiI6I.....",
"token_type": "Bearer",
"expires_in": 3600,
"scope": "urn:iam:t.authpolicies urn:iam:t.authzpolicies urn:iam:t.adminroles"
}
```

Client App Provisioning

1. **Create a client app named "acmehr"**:
   This step involves registering a new client application named "acmehr" with the authorization server. Client applications are entities that request access to protected resources on behalf of a user or themselves.

   ```
   POST https://{{sspHost}}/{{apiPathTenant}}/admin/v1/Apps
   ```

   Create a client app- Response

   Request:

   ```
   {
     "status": "active",
     "name": "acmehr",
     "description": "Client Application to access HR Resource server",
     "clientType": "CONFIDENTIAL",
     "allowedOpenIDScopes": [
       "openid",
       "email",
       "profile",
       "urn:iam:sso:idprop"
     ],
     "redirectURIs": [
       "https://www.example.com"
     ],
     "allowedGrantTypes": [
       "implicit",
       "authorization_code",
       "urn:ietf:params:oauth:grant-type:jwt-bearer",
       "client_credentials",
       "urn:ietf:params:oauth:grant-type:device_code"
     ],
     "allowedOperations": [
       "introspect",
       "onbehalfof",
       "offlineit"
     ],
     "skewTimeSecs": 0,
     "skipIssuerAudienceForIT": false,
     "userInfoEndpointResponseFormat": "PLAIN_JSON",
     "zeroFootPrint": false,
     "skipEmailForIT": false,
     "supportedJoseHeaderParams": "x5t,x5t#s256"
   }
   ```

   Response:

   ```
   {
     "appId": "c95ae9c1-eca0-42e3-afa7-bba019aa82c0",
     "status": "active",
   ```

```
"name": "acmehr",
"resScope": null,
"description": "Client Application to access HR Resource server",
"logoId": null,
"idpDiscoveryEnabled": null,
"skewTimeSecs": 0,
"passwordAuthoritativeSource": "remote",
"mitmProtectionLevel": null,
"zeroFootPrint": false,
"idStoreToUse": null,
"softMFAEnabled": false,
"delegatedAuthentication": false,
"flowURL": null,
"deviceCodeFlowURL": null,
"autoPostToFlowURL": false,
"clientType": "CONFIDENTIAL",
"clientId": "f8378200-bc7a-4e93-9e21-76ed2d0e81aa",
"regenerateClientId": false,
"secret": "a3a51551-03ff-4167-8e79-03093c2bf6d1",
"regenerateSecret": false,
"assertionVerificationCertAlias": null,
"allowedOpenIDScopes": [
  "urn:iam:sso:idprop",
  "openid",
  "email",
  "profile"
],
"redirectURIs": [
  "https://www.example.com"
],
"allowedGrantTypes": [
  "implicit",
  "client_credentials",
  "urn:ietf:params:oauth:grant-type:device_code",
  "authorization_code",
  "urn:ietf:params:oauth:grant-type:jwt-bearer"
],
"allowedOperations": [
  "offlineit",
  "introspect",
  "onbehalfof"
],
"supportedJoseHeaderParams": "x5t,x5t#s256",
"secondaryAudiences": null,
"claims": [],
"userInfoEndpointResponseFormat": "PLAIN_JSON",
"skipIssuerAudienceForIT": false,
"skipEmailForIT": false,
"userTokenSubAttributeMappingName": null,
"customClaimProviderToUse": null,
"itEncryptionTarget": "NONE",
"itEncryptionCertAlias": null,
"userInfoEncryptionCertAlias": null,
```

```
    "samlEntityId": null,
    "samlEntityIdStripe": "c95ae9c1-eca0-42e3-afa7-bba019aa82c0",
    "samlAcsUrl": null,
    "samlEnableSingleLogout": false,
    "samlNameIdFormat": null,
    "samlVerifyRequestSignature": false,
    "samlVerifyCertAlias": null,
    "samlEncryptSamlResponse": false,
    "samlEncryptCertAlias": null,
    "samlEncryptAlgo": null,
    "samlKeyTransportAlgo": null,
    "samlSignResponseKeyAlias": null,
    "samlSignAssertionKeyAlias": null,
    "samlSignResponse": false,
    "samlSignAssertion": false,
    "samlSignResponseAlgo": null,
    "samlSignResponseCanonicalizationAlgo": null,
    "samlSignAssertionAlgo": null,
    "samlSignAssertionCanonicalizationAlgo": null,
    "samlSignatureWithKeyInfo": false,
    "samlIdpInitiatedRelaystateMapping": null,
    "samlAssertionClaims": [],
    "samlUserTokenSubAttributeMappingName": null,
    "homeUrl": null,
    "launcherUrl": null,
    "errorUrl": null,
    "logoutUrl": null,
    "postLogoutUrl": null,
    "metadata": [],
    "label": null,
    "sortOrder": null,
    "hidden": false,
    "primaryAudience": null,
    "resourceServerScopes": [],
    "consentUsagePolicyUrl": null,
    "createdBy": {
      "principalType": "CLIENT",
      "principalId": "ae9964b4-44fa-4976-8557-4097bbfbb9b5",
      "principalName": "DefaultTenantClient"
    },
    "createdDateTime": "2025-11-03 09:21 AM UTC",
    "updatedDateTime": "2025-11-03 09:21 AM UTC",
    "adminscope.criteria": [],
    "adminscope.managedby": [],
    "isOidcApp": true,
    "isSamlApp": false,
    "isLauncherApp": false,
    "requireAccessControl": null,
    "isResourceServerApp": false
  }
```


2. **Perform a GET of the "AuthClientPolicy" AuthZ Policy details**:

This step retrieves the details of a specific Authorization Policy named "AuthClientPolicy". This particular policy is designed to enable client applications to perform authentication-related operations.

```
GET https://{{sspHost}}/{{apiPathTenant}}/admin/v1/AuthZPolicies/f5a82a07-b87b-4557-9967-87cccaefeec2
```

Perform a GET of the "AuthClientPolicy" AuthZ Policy- Response

Response:


```
{
  "policyId": "f5a82a07-b87b-4557-9967-87cccaefeec2",
  "policyName": "AuthClientPolicy",
  "description": "Perform authentication-related operations as required to implement a custom signin app",
  "status": "active",
  "isGrantableClientOnly": true,
  "isSubdelegatable": false,
  "displayName": "Authentication Client",
  "matchAnyApp": false,
  "apps": [
    {
      "name": "SSP",
      "id": "$SSP_APP"
    }
  ],
  "rules": [
    {
      "id": "24fe2029-9490-4d3b-bc92-f086a9b92592",
      "conditions": {
        "principal": {
          "clientApp": {
            "operator": "in",
            "value": [
              "$DEMO_CLIENT_APP_ID"
            ]
          }
        }
      },
      "result": {
        "effect": "grant",
        "msg": "This policy allows clients to call authentication APIs",
        "privileges": [
          "$AUTHCLIENT_PRIVILEGES"
        ]
      }
    }
  ],
  "adminscope.criteria": [],
  "adminscope.managedby": [],
  "createdDate": "2025-10-15 06:05 AM UTC",
  "updatedBy": {
    "principalType": "USER",
    "principalId": "ff337dec-d588-4564-b147-d1cccb3e4a73",
    "principalName": "internaluser"
  },
  "updatedDate": "2025-11-03 07:22 AM UTC",
  "preventedOperations": "Delete,Update"
```

```
}
```

3. **Grant the "AuthClientPolicy" role to the already created client app** (acmehr - id:c95ae9c1-eca0-42e3-afa7-bba019aa82c0).
   The "AuthClientPolicy" is granted to the "acmehr" client application. This action effectively assigns the privileges defined within the "AuthClientPolicy" to the "acmehr" app, allowing it to execute authentication-related operations. The request uses the "authZPolicyID" to identify the policy and "ClientAppId" to specify the recipient client application.

   ```
   POST https://{{sspHost}}/{{apiPathTenant}}/admin/v1/AdminRolesDelegator
   ```

   Grant the "AuthClientPolicy" role- Request and Response

   Request:

   ```
   {
     "id": "f5a82a07-b87b-4557-9967-87cccaefeec2",
     "principal": {
       "clientApp": {
         "operator": "in",
         "value": [
           "c95ae9c1-eca0-42e3-afa7-bba019aa82c0"
         ]
       }
     }
   }
   ```

   Where,

   - id = This is the "AuthClientPolicy" authorization PolicyID
   - ClientApp value = This is the client app (acmehr app id) already created in step1.

   Response:

   ```
   {
     "id": "f5a82a07-b87b-4557-9967-87cccaefeec2",
     "name": "AuthClientPolicy",
     "displayName": "Authentication Client",
     "description": "Perform authentication-related operations as required to implement a custom signin app",
     "principal": {
       "clientApp": {
         "operator": "in",
         "value": [
           "c95ae9c1-eca0-42e3-afa7-bba019aa82c0",
           "$DEMO_CLIENT_APP_ID"
         ]
       }
     },
     "isGrantableClientOnly": true,
     "isSubdelegatable": false,
     "privileges": [
       "$AUTHCLIENT_PRIVILEGES"
     ]
   }
   ```

4. **Create a authentication Policy for the Client App**:
   This step creates an Authentication Policy specifically for the "acmehr" client application. This policy defines rules that, under certain conditions (for example, a specific user attempting to "authenticate"), will result in a granted effect with required obligations (for example, "PASSWORD:1" for single-factor password authentication).

   ```
   POST https://{{sspHost}}/{{apiPathTenant}}/admin/v1/AuthPolicies
   ```

Create a authentication Policy for the Client App- Request and Response

Request:

```
{
  "description": "Authentication policy for acmehr",
  "status": "active",
  "policyName": "AuthNPolicy for acmehr",
  "apps": [
    {
      "id": "c95ae9c1-eca0-42e3-afa7-bba019aa82c0"
    }
  ],
  "rules": [
    {
      "conditions": {
        "principal": {
          "user": {
            "operator": "in",
            "value": [
              "malbail"
            ]
          }
        },
        "action": {
          "operator": "in",
          "value": [
            "authenticate"
          ]
        }
      },
      "result": {
        "effect": "allow",
        "obligations": [
          "PASSWORD:1"
        ],
        "msg": "AuthNPolicy rule for acmehr"
      }
    }
  ]
}
```

Where,

- "description" and "policyName" has the client app name, which is "acmehr" in this example.
- "apps" id = client app Id (acmehr app id)

Response:

```
{
  "policyId": "f543eac3-d0f3-4a7d-b965-f47599fcebd6",
  "policyName": "AuthNPolicy for acmehr",
  "description": "Authentication policy for acmehr",
  "status": "active",
  "matchAnyApp": false,
  "apps": [
    {
```

```
        "name": "acmehr",
        "id": "c95ae9c1-eca0-42e3-afa7-bba019aa82c0"
      }
    ],
    "rules": [
      {
        "id": "d732df16-f896-482f-8b9a-ed0ad9c061ce",
        "conditions": {
          "principal": {
            "user": {
              "operator": "in",
              "value": [
                "malbail"
              ]
            }
          },
          "action": {
            "operator": "in",
            "value": [
              "authenticate"
            ]
          }
        },
        "result": {
          "effect": "allow",
          "reAuthenticate": "false",
          "obligations": [
            "PASSWORD:1"
          ],
          "msg": "AuthNPolicy rule for acmehr"
        }
      }
    ],
    "adminscope.criteria": [],
    "adminscope.managedby": [],
    "createdBy": {
      "principalType": "CLIENT",
      "principalId": "ae9964b4-44fa-4976-8557-4097bbfbb9b5",
      "principalName": "DefaultTenantClient"
    },
    "createdDate": "2025-11-03 09:47 AM UTC",
    "policySubType": "userDefined"
  }
```


Client Provisioning steps end here. Next section describes Resource Server App provisioning.

Resource Server Provisioning

NOTE

- An authorization policy (AppRole) can include only single resource server app. It is not recommended to have multiple resource server apps associated to a single authorization policy.

  Example:

  ```
  "apps": [
  ```

```
 {
  "id": "{{rsAppId}}"
 }
]
```

- When configuring an application role for a resource server, verify that the resource server application is the sole entry under 'apps'. The out-of-the-box SSP application should not be included along with the resource server application.
- Only resource server applications can be assigned to application roles; normal applications are not supported.
- Application group is supported in Authentication policy, Risk policy and IDP Discovery policy. It is not supported in Authorization policy.

1. **Create a Resource Server App** (HR)
   This step involves registering the "HR" application as a Resource Server. A Resource Server hosts protected resources and accepts and responds to protected resource requests using access tokens.
   Key attributes for a Resource Server include the following:
   For all other attribute descriptions, see Description of Application Attributes (OIDC and SAML attributes).

   - **name**: The name of the Resource Server application (example, "HR").
   - **resScope**: A unique URI that identifies the Resource Server within the authorization server's domain (e.g., "urn:rs:hr"). This scope is used to request access to resources protected by this server.
   - **description**: A clear description of the Resource Server's function.
   - **isResourceServerApp**: Set to `true` to designate this application as a Resource Server. This enables specific behaviors and configurations related to resource protection.
   - **primaryAudience**: The primary audience for access tokens issued for this Resource Server (e.g., "https://hr.com"). Access tokens with this audience are intended for this Resource Server.
   - **resourceServerScopes**: A list of scopes defined by the Resource Server, along with their descriptions and whether consent is required. These scopes represent specific permissions that client applications can request to access the Resource Server's protected resources.
     - **promote**: A scope to manage promotions within a department.
     - **hire**: A scope to manage hiring processes within a department.
     - **transfer**: A scope to manage transfers in and out of a department.
   - **consentTermsAndConditionsUrl**: The URL for the terms and conditions that users must agree to before granting consent to the client application.
   - **consentPrivacyPolicyUrl**: The URL for the privacy policy that users can review before granting consent to the client application.

   ```
   POST https://{{sspHost}}/{{apiPathTenant}}/admin/v1/Apps
   ```

   Create a a Resource Server App- Request and Response

   Request:

   ```
   {
     "status": "active",
     "name": "HR",
     "resScope": "urn:rs:hr",
     "description": "HR Resource Server Application",
     "idpDiscoveryEnabled": null,
     "skewTimeSecs": 0,
     "passwordAuthoritativeSource": "remote",
     "mitmProtectionLevel": null,
     "zeroFootPrint": false,
     "idStoreToUse": null,
     "softMFAEnabled": false,
   ```

```
"delegatedAuthentication": false,
"flowURL": null,
"deviceCodeFlowURL": null,
"autoPostToFlowURL": false,
"clientType": "CONFIDENTIAL",
"regenerateClientId": false,
"regenerateSecret": false,
"assertionVerificationCertAlias": null,
"allowedOpenIDScopes": [
  "openid",
  "profile",
  "groups",
  "address",
  "phone",
  "offline_access",
  "urn:iam:scim:core",
  "urn:iam:scim:ssp",
  "urn:iam:scim:credentials",
  "urn:iam:scim:enterprise",
  "urn:iam:scim:custom",
  "email"
],
"redirectURIs": [
  "https://www.example.com"
],
"allowedGrantTypes": [
  "implicit",
  "client_credentials",
  "urn:ietf:params:oauth:grant-type:device_code",
  "authorization_code",
  "urn:ietf:params:oauth:grant-type:jwt-bearer"
],
"allowedOperations": [
  "offlineit",
  "introspect",
  "onbehalfof"
],
"supportedJoseHeaderParams": "x5t,x5t#s256",
"secondaryAudiences": [],
"claims": [],
"userInfoEndpointResponseFormat": "PLAIN_JSON",
"skipIssuerAudienceForIT": false,
"skipEmailForIT": false,
"userTokenSubAttributeMappingName": null,
"customClaimProviderToUse": null,
"itEncryptionTarget": "NONE",
"itEncryptionCertAlias": null,
"userInfoEncryptionCertAlias": null,
"samlEntityId": null,
"samlAcsUrl": null,
"samlEnableSingleLogout": false,
"samlNameIdFormat": null,
"samlVerifyRequestSignature": false,
```

```
"samlVerifyCertAlias": null,
"samlEncryptSamlResponse": false,
"samlEncryptCertAlias": null,
"samlEncryptAlgo": null,
"samlKeyTransportAlgo": null,
"samlSignResponseKeyAlias": null,
"samlSignAssertionKeyAlias": null,
"samlSignResponse": false,
"samlSignAssertion": false,
"samlSignResponseAlgo": null,
"samlSignResponseCanonicalizationAlgo": null,
"samlSignAssertionAlgo": null,
"samlSignAssertionCanonicalizationAlgo": null,
"samlSignatureWithKeyInfo": false,
"samlIdpInitiatedRelaystateMapping": [],
"samlAssertionClaims": [],
"samlUserTokenSubAttributeMappingName": null,
"homeUrl": null,
"launcherUrl": null,
"errorUrl": null,
"logoutUrl": null,
"postLogoutUrl": null,
"metadata": [
  {
    "name": "urn:iam:app:allowNonSslRedirectUri",
    "value": "true"
  },
  {
    "name": "urn:iam:app:allowNonPKCEConnections",
    "value": "true"
  },
  {
    "name": "urn:iam:app:allowAppNotToSendRedirectUri",
    "value": "true"
  }
],
"label": null,
"sortOrder": null,
"hidden": false,
"isResourceServerApp": true,
"consentUsagePolicyUrl": "https://policy.hr.com",
"primaryAudience": "https://hr.com",
"resourceServerScopes": [
  {
    "scope": "promote",
    "description": "scope to promote",
    "consentDescription": "Review, approve, and manage employee promotions within your department",
    "label": "View and manage promotions in your department",
    "consentRequired": false
  },
  {
    "scope": "hire",
    "description": "scope to hire",
```

```
      "consentDescription": "Review, approve, and manage hiring of new employees within your department",
      "label": "View and manage hiring in your department",
      "consentRequired": false
    },
    {
      "scope": "transfer",
      "description": "scope to transfer",
      "consentDescription": "Review, approve, and manage transfer of new employees across departments",
      "label": "View and manage transfers in and out of your department",
      "consentRequired": false
    }
  ],
  "isOidcApp": true,
  "isSamlApp": false,
  "isLauncherApp": false,
  "requireAccessControl": null
}
```


Response:


```
{
  "appId": "25d25465-db50-44ec-b9ea-5c9b2ef2f8ff",
  "status": "active",
  "name": "HR",
  "resScope": "urn:rs:hr",
  "description": "HR Resource Server Application",
  "logoId": null,
  "idpDiscoveryEnabled": null,
  "skewTimeSecs": 0,
  "passwordAuthoritativeSource": "remote",
  "mitmProtectionLevel": null,
  "zeroFootPrint": false,
  "idStoreToUse": null,
  "softMFAEnabled": false,
  "delegatedAuthentication": false,
  "flowURL": null,
  "deviceCodeFlowURL": null,
  "autoPostToFlowURL": false,
  "clientType": "CONFIDENTIAL",
  "clientId": "d546089d-ea21-4caf-9e3e-87aa231c16c5",
  "regenerateClientId": false,
  "secret": "bbeee44e-c63c-4932-ade7-0433aebc1724",
  "regenerateSecret": false,
  "assertionVerificationCertAlias": null,
  "allowedOpenIDScopes": [
    "address",
    "phone",
    "openid",
    "urn:iam:scim:core",
    "urn:iam:scim:ssp",
    "offline_access",
    "profile",
    "groups",
    "urn:iam:scim:enterprise",
```

```
    "urn:iam:scim:credentials",
    "urn:iam:scim:custom",
    "email"
  ],
  "redirectURIs": [
    "https://www.example.com"
  ],
  "allowedGrantTypes": [
    "implicit",
    "client_credentials",
    "urn:ietf:params:oauth:grant-type:device_code",
    "authorization_code",
    "urn:ietf:params:oauth:grant-type:jwt-bearer"
  ],
  "allowedOperations": [
    "offlineit",
    "introspect",
    "onbehalfof"
  ],
  "supportedJoseHeaderParams": "x5t,x5t#s256",
  "secondaryAudiences": [],
  "claims": [],
  "userInfoEndpointResponseFormat": "PLAIN_JSON",
  "skipIssuerAudienceForIT": false,
  "skipEmailForIT": false,
  "userTokenSubAttributeMappingName": null,
  "customClaimProviderToUse": null,
  "itEncryptionTarget": "NONE",
  "itEncryptionCertAlias": null,
  "userInfoEncryptionCertAlias": null,
  "samlEntityId": null,
  "samlEntityIdStripe": "25d25465-db50-44ec-b9ea-5c9b2ef2f8ff",
  "samlAcsUrl": null,
  "samlEnableSingleLogout": false,
  "samlNameIdFormat": null,
  "samlVerifyRequestSignature": false,
  "samlVerifyCertAlias": null,
  "samlEncryptSamlResponse": false,
  "samlEncryptCertAlias": null,
  "samlEncryptAlgo": null,
  "samlKeyTransportAlgo": null,
  "samlSignResponseKeyAlias": null,
  "samlSignAssertionKeyAlias": null,
  "samlSignResponse": false,
  "samlSignAssertion": false,
  "samlSignResponseAlgo": null,
  "samlSignResponseCanonicalizationAlgo": null,
  "samlSignAssertionAlgo": null,
  "samlSignAssertionCanonicalizationAlgo": null,
  "samlSignatureWithKeyInfo": false,
  "samlIdpInitiatedRelaystateMapping": [],
  "samlAssertionClaims": [],
  "samlUserTokenSubAttributeMappingName": null,
```

```
"homeUrl": null,
"launcherUrl": null,
"errorUrl": null,
"logoutUrl": null,
"postLogoutUrl": null,
"metadata": [
  {
    "name": "urn:iam:app:allowNonPKCEConnections",
    "value": "true"
  },
  {
    "name": "urn:iam:app:allowNonSslRedirectUri",
    "value": "true"
  },
  {
    "name": "urn:iam:app:allowAppNotToSendRedirectUri",
    "value": "true"
  }
],
"label": null,
"sortOrder": null,
"hidden": false,
"primaryAudience": "https://hr.com",
"resourceServerScopes": [
  {
    "scope": "promote",
    "description": "scope to promote",
    "consentRequired": false,
    "label": "View and manage promotions in your department",
    "consentDescription": "Review, approve, and manage employee promotions within your department"
  },
  {
    "scope": "hire",
    "description": "scope to hire",
    "consentRequired": false,
    "label": "View and manage hiring in your department",
    "consentDescription": "Review, approve, and manage hiring of new employees within your department"
  },
  {
    "scope": "transfer",
    "description": "scope to transfer",
    "consentRequired": false,
    "label": "View and manage transfers in and out of your department",
    "consentDescription": "Review, approve, and manage transfer of new employees across departments"
  }
],
"consentUsagePolicyUrl": "https://policy.hr.com",
"createdBy": {
  "principalType": "CLIENT",
  "principalId": "ae9964b4-44fa-4976-8557-4097bbfbb9b5",
  "principalName": "DefaultTenantClient"
},
"createdDateTime": "2025-11-03 09:51 AM UTC",
```

```
  "updatedDateTime": "2025-11-03 09:51 AM UTC",
  "adminscope.criteria": [],
  "adminscope.managedby": [],
  "isOidcApp": true,
  "isSamlApp": false,
  "isLauncherApp": false,
  "requireAccessControl": null,
  "isResourceServerApp": true
}
```

2. **Create a Role (Hire and Transfer) for the Resource Server App**

   This step creates an Authorization Policy named "HRHireAndTransfer". In the context of a Resource Server, an Authorization Policy acts as a role that defines a set of privileges that can be granted to a principal (user, client app, or group). This specific policy grants the "hire" and "transfer" privileges, allowing entities assigned this role to perform hiring and transfer-related operations on the HR Resource Server.

   ```
   POST https://{{sspHost}}/{{apiPathTenant}}/admin/v1/AuthZPolicies
   ```

   Create a Role (Hire and Transfer) for the Resource Server App- Request and Response

   Request:

   ```
   {
     "policyName": "HRHireAndTransfer",
     "description": "HR role for hire and transfer privilege",
     "displayName": "HR role for hire and transfer privilege",
     "status": "active",
     "isGrantableClientOnly": false,
     "isSubdelegatable": true,
     "matchAnyApp": false,
     "apps": [
       {
         "id": "25d25465-db50-44ec-b9ea-5c9b2ef2f8ff"
       }
     ],
     "rules": [
       {
         "conditions": {
           "principal": {
             "clientApp": {
               "operator": "in",
               "value": [
                 "c95ae9c1-eca0-42e3-afa7-bba019aa82c0"
               ]
             },
             "user": {
               "operator": "in",
               "value": [
                 "malbail"
               ]
             }
           }
         },
         "result": {
           "effect": "grant",
           "msg": "HR role for hire and transfer privilege",
           "privileges": [
   ```

```
          "hire",
          "transfer"
        ]
      }
    }
  ]
}
```

Where,

- apps Id = This is the value of the resource app (HR) created in step 1 (25d25465-db50-44ec-b9ea-5c9b2ef2f8ff)
- Under rules, the client app value = acmehr client app id, and user =malbail.

Response:


```
{
  "policyId": "6b53393e-d906-45ca-a531-2e7c0daa0c90",
  "policyName": "HRHireAndTransfer",
  "description": "HR role for hire and transfer privilege",
  "status": "active",
  "isGrantableClientOnly": false,
  "isSubdelegatable": true,
  "displayName": "HR role for hire and transfer privilege",
  "matchAnyApp": false,
  "apps": [
    {
      "name": "HR",
      "id": "25d25465-db50-44ec-b9ea-5c9b2ef2f8ff"
    }
  ],
  "rules": [
    {
      "id": "eda41878-65cd-4a1a-a37c-05ad96085c54",
      "conditions": {
        "principal": {
          "user": {
            "operator": "in",
            "value": [
              "malbail"
            ]
          },
          "clientApp": {
            "operator": "in",
            "value": [
              "c95ae9c1-eca0-42e3-afa7-bba019aa82c0"
            ]
          }
        }
      },
      "result": {
        "effect": "grant",
        "msg": "HR role for hire and transfer privilege",
        "privileges": [
          "hire",
          "transfer"
```

```
          ]
        }
      }
    ],
    "adminscope.criteria": [],
    "adminscope.managedby": [],
    "createdBy": {
      "principalType": "CLIENT",
      "principalId": "ae9964b4-44fa-4976-8557-4097bbfbb9b5",
      "principalName": "DefaultTenantClient"
    },
    "createdDate": "2025-11-03 09:59 AM UTC"
  }
```

3. **Create Promote role for the Resource Server App**:

   Similar to the previous step, this creates another Authorization Policy named "HRPromote". This policy defines the "promote" privilege, enabling principals with this role to manage promotions within the HR department. These roles are critical for implementing fine-grained access control to the Resource Server's functionalities.

   ```
   POST https://{{sspHost}}/{{apiPathTenant}}/admin/v1/AuthZPolicies
   ```

   Create Promote role for the Resource Server App- Request and Response

   Request:

   ```
   {
     "policyName": "HRPromote",
     "description": "HR promote role",
     "displayName": "HR promote role",
     "status": "active",
     "isGrantableClientOnly": false,
     "isSubdelegatable": true,
     "matchAnyApp": false,
     "apps": [
       {
         "id": "25d25465-db50-44ec-b9ea-5c9b2ef2f8ff"
       }
     ],
     "rules": [
       {
         "conditions": {
           "principal": {
             "clientApp": {
               "operator": "in",
               "value": [
                 "c95ae9c1-eca0-42e3-afa7-bba019aa82c0"
               ]
             },
             "user": {
               "operator": "in",
               "value": [
                 "malbail"
               ]
             }
           }
         },
         "result": {
   ```

```
        "effect": "grant",
        "msg": "HR promote role",
        "privileges": [
          "promote"
        ]
      }
    }
  ]
}
```

Where,

- apps Id = This is the value of the resource app (HR) created in step 1 (25d25465-db50-44ec-b9ea-5c9b2ef2f8ff)
- Under rules, the client app value = acmehr client app id, and user = malbail.

Response:

```
{
  "policyId": "0d701ed6-eab4-44bf-8132-5cf1c1042d40",
  "policyName": "HRPromote",
  "description": "HR promote role",
  "status": "active",
  "isGrantableClientOnly": false,
  "isSubdelegatable": true,
  "displayName": "HR promote role",
  "matchAnyApp": false,
  "apps": [
    {
      "name": "HR",
      "id": "25d25465-db50-44ec-b9ea-5c9b2ef2f8ff"
    }
  ],
  "rules": [
    {
      "id": "bda17ffe-eec9-454b-9cf1-79031a15c8b8",
      "conditions": {
        "principal": {
          "user": {
            "operator": "in",
            "value": [
              "malbail"
            ]
          },
          "clientApp": {
            "operator": "in",
            "value": [
              "c95ae9c1-eca0-42e3-afa7-bba019aa82c0"
            ]
          }
        }
      },
      "result": {
        "effect": "grant",
        "msg": "HR promote role",
        "privileges": [
```

```
            "promote"
          ]
        }
      }
    ],
    "adminscope.criteria": [],
    "adminscope.managedby": [],
    "createdBy": {
      "principalType": "CLIENT",
      "principalId": "ae9964b4-44fa-4976-8557-4097bbfbb9b5",
      "principalName": "DefaultTenantClient"
    },
    "createdDate": "2025-11-03 10:08 AM UTC"
  }
```

4. **Grant (Hire and Transfer) role to the Client App**

   This action delegates the "HRHireAndTransfer" role to the client application (acmehr) and/or specific users. By granting this role, the client application (and/or user) is authorized to request and receive access tokens with the "hire" and "transfer" scopes, allowing them to interact with the HR Resource Server's protected resources that correspond to these privileges.

   ```
   POST https://{{sspHost}}/{{apiPathTenant}}/admin/v1/AppRolesDelegator
   ```

   Grant (Hire and Transfer) role to the Client App- Request and Response

   Request:

   ```
   {
     "id": "6b53393e-d906-45ca-a531-2e7c0daa0c90",
     "principal": {
       "clientApp": {
         "operator": "in",
         "value": [
           "c95ae9c1-eca0-42e3-afa7-bba019aa82c0"
         ]
       },
       "user": {
         "operator": "in",
         "value": [
           "malbail"
         ]
       }
     }
   }
   ```

   Where,

   - id= This is the "hireandtransfer" PolicyId
   - Under rules, the client app value = acmehr client app id, and user = malbail.

   Response:

   ```
   {
    "id": "6b53393e-d906-45ca-a531-2e7c0daa0c90",
    "name": "HRHireAndTransfer",
    "displayName": "HR role for hire and transfer privilege",
    "description": "HR role for hire and transfer privilege",
    "principal": {
   ```

```
    "user": {
      "operator": "in",
      "value": [
        "malbail"
      ]
    },
    "clientApp": {
      "operator": "in",
      "value": [
        "c95ae9c1-eca0-42e3-afa7-bba019aa82c0"
      ]
    }
  },
  "isGrantableClientOnly": false,
  "isSubdelegatable": true,
  "privileges": [
    "hire",
    "transfer"
  ]
}
```

5. **Grant (Promote) role to the Client App**

   This step grants the "HRPromote" role to the client application (acmehr) and/or specific users. This authorizes the client application (and/or user) to request and use access tokens that include the "promote" scope, thereby enabling access to promotion-related functionalities on the HR Resource Server.

   ```
   POST https://{{sspHost}}/{{apiPathTenant}}/admin/v1/AppRolesDelegator
   ```

   Grant (Promote) role to the Client App- Request and Response

   Request:

   ```
   {
     "id": "0d701ed6-eab4-44bf-8132-5cf1c1042d40",
     "principal": {
       "clientApp": {
         "operator": "in",
         "value": [
           "c95ae9c1-eca0-42e3-afa7-bba019aa82c0"
         ]
       },
       "user": {
         "operator": "in",
         "value": [
           "malbail"
         ]
       }
     }
   }
   ```

   Where,

   - id= This is the "Promote" PolicyId
   - Under rules, the client app value = acmehr client app id, and user = malbail.

   Response:

   ```
   {
   ```

```
  "id": "0d701ed6-eab4-44bf-8132-5cf1c1042d40",
  "name": "HRPromote",
  "displayName": "HR promote role",
  "description": "HR promote role",
  "principal": {
    "user": {
      "operator": "in",
      "value": [
        "malbail"
      ]
    },
    "clientApp": {
      "operator": "in",
      "value": [
        "c95ae9c1-eca0-42e3-afa7-bba019aa82c0"
      ]
    }
  },
  "isGrantableClientOnly": false,
  "isSubdelegatable": true,
  "privileges": [
    "promote"
  ]
}
```

6. **(Optional) Create an Access Policy for the Resource App**:
   This is the Authentication Policy created for the HR Resource Server, named "Access Policy for HR". This policy defines the access rules for the Resource Server. It specifies that any principal possessing either the "hireandtransferPolicyId" or "promotePolicyId" app roles is allowed to "access" the HR Resource Server. This policy acts as a gatekeeper, ensuring that only clients with the appropriate roles can interact with the HR resources.

   ```
   POST https://{{sspHost}}/{{apiPathTenant}}/admin/v1/AuthPolicies
   ```

   Create an Access Policy for the Resource App- Request and Response

   Request:

   ```
   {
     "description": "Access policy for HR ",
     "status": "active",
     "policyName": "Access Policy for HR",
     "apps": [
       {
         "id": "{{rsAppId}}"
       }
     ],
     "rules": [
       {
         "conditions": {
           "principal": {
             "appRole": {
               "operator": "in",
               "value": [
                 "6b53393e-d906-45ca-a531-2e7c0daa0c90",
                 "0d701ed6-eab4-44bf-8132-5cf1c1042d40"
               ]
   ```

```
          }
        },
        "action": {
          "operator": "in",
          "value": [
            "access"
          ]
        }
      },
      "result": {
        "effect": "allow",
        "msg": "Access rule for HR"
      }
    }
  ]
}
```

Where,

- "description" and "policyName"= This is the value of the resource app (HR in this example)
- "id"= This is the resource app id.
- Under rules, app role value= "hireandtransfer" policy Id and promote policy Id.

Response:

```
{
  "policyId": "fadbc766-642c-4f5e-ad28-f4f40614db2f",
  "policyName": "Access Policy for HR",
  "description": "Access policy for HR ",
  "status": "active",
  "matchAnyApp": false,
  "apps": [
    {
      "name": "HR",
      "id": "25d25465-db50-44ec-b9ea-5c9b2ef2f8ff"
    }
  ],
  "rules": [
    {
      "id": "c5b755a1-477e-4a3d-ae1c-ed19df025597",
      "conditions": {
        "principal": {
          "appRole": {
            "operator": "in",
            "value": [
              "6b53393e-d906-45ca-a531-2e7c0daa0c90",
              "0d701ed6-eab4-44bf-8132-5cf1c1042d40"
            ]
          }
        },
        "action": {
          "operator": "in",
          "value": [
            "access"
          ]
```

```
            }
          },
          "result": {
            "effect": "allow",
            "reAuthenticate": "false",
            "msg": "Access rule for HR"
          }
        }
      ],
      "adminscope.criteria": [],
      "adminscope.managedby": [],
      "createdBy": {
        "principalType": "CLIENT",
        "principalId": "ae9964b4-44fa-4976-8557-4097bbfbb9b5",
        "principalName": "DefaultTenantClient"
      },
      "createdDate": "2025-11-03 10:26 AM UTC",
      "policySubType": "userDefined"
    }
```


Client App Flow Using Resource Server

This section describes the client application's interaction with the Resource Server after all provisioning and policy configurations are in place. The client application acquires an access token with the necessary scopes to access the protected resources.

```
POST https://{{sspHost}}/{{apiPathTenant}}/oauth2/v1/token
```

This is a request to the OAuth 2.0 token endpoint to obtain an access token. The client application typically sends its credentials and requests specific scopes (for example, "promote", "transfer", "hire") that it needs to access the Resource Server's functionalities.

Client App Flow Using Resource Server- Request and Response

Response:


```
{
"access_token": "eyJ4NXQjUzI1NiI6IjdvcXZva2tnSkZ0dWd0....",
"token_type": "Bearer",
"expires_in": 3600,
"scope": "promote transfer hire"
}
```


Where,

- access_token - "eyJ4NXQjUzI1NiI6IjdvcXZva2tnSkZ0dWd0...." - The JWT (JSON Web Token) that grants access to protected resources on the Resource Server.
- token_type: "Bearer" - Indicates the type of token issued. "Bearer" means the token should be included in the Authorization header as "Bearer <access_token>" when making requests to protected resources.
- expires_in: 3600 - The lifetime of the access token in seconds. In this case, 3600 seconds (1 hour) indicates when the token will expire and need to be renewed.
- Scope: "promote transfer hire" - These scopes indicate that the issued access token grants the client application permissions for the "promote", "transfer", and "hire" operations on the Resource Server. These are the specific privileges that the client application can exercise.

Decoded Response:


```
{
```

```
"client_tid": "181b8f45-e1dc-45cf-8d5b-7811b6c4f19b",
"sub": "f8378200-bc7a-4e93-9e21-76ed2d0e81aa",
"ver": "1.0",
"iss": "https://<sspHost>/default/",
"azpacr": "1",
"tname": "default",
"tok_type": "AT",
"user_ipaddress": "192.19.203.252",
"tid": "181b8f45-e1dc-45cf-8d5b-7811b6c4f19b",
"transactionid": "7b753052-524f-4ad5-9d47-bf85006c7944",
"aud": [
"https://hr.com",
"https://<sspHost>/default/"
],
"nbf": 1762246429,
"client_transactionid": "ssp:7b753052-524f-4ad5-9d47-bf85006c7944",
"sub_type": "CLIENT",
"azp": "f8378200-bc7a-4e93-9e21-76ed2d0e81aa",
"scope": "promote transfer hire",
"tok_source": "CCR",
"exp": 1762250029,
"iat": 1762246429,
"client_name": "acmehr",
"jti": "f2d4b00c-651e-421a-9ca1-c2d21b1ffe3c",
"client_tname": "default"
}
```


OIDC Flow using Resource Server

These steps outlines the OpenID Connect (OIDC) flow, detailing the steps required to acquire an Access Token (AT) and Identity Token (IT) when interacting with a resource server.

1. **Initiate Authorization Request**
   This is the initial request made to the authorization server to begin the OIDC flow. The client application requests authorization from the user, specifying its identity and the resource server it intends to access.
   ```
   POST https://{{sspHost}}/{{apiPathTenant}}/oauth2/v1/authorize?signin_hint=true
   ```
   Where, pass the query parameter, `signin_hint=true` in the authorization URL. This parameter instructs the authorization server to immediately present the sign-in interface to the user.
   Initiate Authorization - Request and Response
   In the request, specify the following request body parameters:

Key	Value
client_id	Identifies the client application provisioned in the system (For example, acmehr).
resource	Specifies the identifier of the resource server - urn:rs:type:appid:25d25465-db50-44ec-b9ea-5c9b2ef2f8ff
response_type	Specifies the OAuth 2.0 flow type being used. Must be set to code for Authorization Code flow. For example: "response_type=code"
redirect_URI	URL where the authorization server will redirect the user after authentication. This URL should match the redirect URIs registered for the client application. For example: "redirect_URI=https://www.example.com"

code_challenge	Base64-URL encoded SHA256 hash of the code verifier for Proof Key for Code Exchange (PKCE). Example: "code_challenge=A6xnQhbz4Vx2HuGl4lXwZ5U2I8iziLRFnhP5eNfIRvQ"
code_challenge_method	Specifies the method used to generate the code challenge. It should be S256 (SHA256 hashing method) when using PKCE. Example: code_challenge_method=S256
scope	List of permissions being requested from the resource server. These scopes must match scopes defined in the resource server application. Example: "scope=promote hire transfer"

Response:

```
https://<sspHost>/default/ui/v1/signin/?x-flow-state=UHh4OTdkQ3BWUmNySkxub2FKbHc0dHl6WklyUytBRFZm....
```

The authorization server responds with a redirect URI that includes the following parameters, crucial for continuing the authentication flow:

- redirect Uri - The URL to which the user's browser should be redirected.
- x-flow-state - A unique identifier representing the current state of the authentication flow.
- x-service-baseurl - The base URL for subsequent service interactions.

2. **Initial Token Request (Client Credentials Flow)**

This step involves invoking the token endpoint. This initial call is made to the token endpoint to obtain an Access Token using client credentials.

```
POST https://{{sspHost}}/{{apiPathTenant}}/oauth2/v1/token
```

Initial Token Request - Request and Response

Request:

- Under "Auth" headers > Basic Auth - provide the client app id and password.
- Under "Body" parameters > - provide grant_type=client_credentials, and scope=urn:iam:myscopes

Response:

```
{
"access_token": "eyJ4NXQjUzI1NiI6IjdvcXZva2tnSkZ0dWd0TEhyem....",
"token_type": "Bearer",
"expires_in": 3600,
"scope": "urn:iam:t.authenticate urn:iam:m.meclient"
}
```

Where,

- access_token: The JWT (JSON Web Token) that grants access to protected resources.
- token_type: "Bearer" - Indicates the type of token issued. "Bearer" means the token should be included in the Authorization header as "Bearer <access_token>" when making requests to protected resources.
- expires_in: 3600 - The lifetime of the access token in seconds. In this case, 3600 seconds (1 hour) indicates when the token will expire and need to be renewed.
- scope - The permissions granted by the Access Token (for example, urn:iam:t.authenticate urn:iam:m.meclient).

3. **User Authentication - Provide Username**

This step initiates the user authentication process. The client sends the user's login name to the authentication endpoint, along with the x-flow-state received in the initial authorization request. This informs the system about the user attempting to authenticate and the current state of their session.

```
POST https://{{sspHost}}/{{apiPathTenant}}/auth/v1/authenticate
```

User Authentication (Provide Username) - Request and Response

Request:

```
{
```

```
"subject": "{{userLoginName}}",
"action": "authenticate",
"ipAddress": "1.1.1.2",
"channel": "web",
"acrValues": []
}
```

Where,

- subject: The user's login name who is attempting to authenticate.
- action: Specifies the action being performed, which is authentication in this case.
- ipAddress: The IP address of the client making the authentication request.
- channel: The channel or medium through which the authentication request is being made.
- acrValues: Authentication Context Class Reference values.

Response:

```
{
"flowState":
 "UHh4OTdkQ3BWUmNySkxub2FKbHc0dHl6WklyUytBRFZmRGlBWmxNNThJRitvNzRUOWUwYkRZVDh0WTRmaHA0OGJPM1hac0dLYm12U3JxVzYxWGkwVm
"userName": "malbail",
"nextaction": "PASSWORD_AUTH",
"currentFactors": [
"PASSWORD"
],
"additional": {
"userName": "malbail",
"idpName": "internal",
"idpGuid": "internal",
"idpType": "internal",
"passwordResetURL": "https://<sspHost>/default/ui/v1/signin/initiatePasswordReset?x-flow-
state=UHh4OTdkQ3BWUmNySkxub2FKbHc0dHl6WklyUytBRFZmRGlBWmxNNThJRitvNzRUOWUwYkRZVDh0WTRmaHA0OGJPM1hac0dLYm12U3JxVzYxWG
"currentFactorLevel": 1,
"predicatesForCurrentFactors": [
{
"factor": "PASSWORD",
"predicate": "internal",
"providerAmr": "PWD"
}
],
"signinExpMins": 60,
"identityProviders": []
}
}
```

Where,

- flowState: The current x-flow-state to maintain the authentication session.
- userName: The user's login name (malbail in this example).
- nextaction: Indicates the next expected action in the authentication flow, such as PASSWORD_AUTH.
- currentFactors: An array listing the authentication factors currently being used (for example, PASSWORD).
- additional: Contains supplementary authentication information:

- userName: "malbail" - The user's login name.
- idpName: "internal" - The identity provider name.
- idpGuid: "internal" - The identity provider GUID.
- idpType: "internal" - The type of identity provider.
- passwordResetURL: URL for password reset functionality.
- currentFactorLevel: 1 - The current authentication factor level.
- predicatesForCurrentFactors: Array containing factor details including factor type, predicate, and provider AMR.
- signinExpMins: 60 - Session expiration time in minutes.
- identityProviders: Array of available identity providers.

4. **User Authentication - Provide Password**

Following the username submission, this step authenticates the user by verifying their password. The client sends the user's credentials (or an indicator that the password has been provided) along with the flowState to the designated password authenticator endpoint.

```
POST https://{{sspHost}}/{{apiPathTenant}}/factor/v1/PasswordAuthenticator
```

User Authentication (Provide Password) - Request and Response

Request:

```
{
// user password in plain text or encrypted using tenant's public signing certificate
"password": "{{defaultPassword}}"
}
```

Where,

- Enter the password for the user.

Response

```
{
"flowState":
 "UHh4OTdkQ3BWUmNySkxub2FKbHc0dHl6WklyUytBRFZmRGlBWmxNNThJRitvNzRUOWUwYkRZVDh0WTRmaHA0OGJPM1hac0dLYm12U3JxVzYxWGkwVm
"userName": "malbail",
"nextaction": "AUTH_ALLOWED",
"data": {
"message": "Success",
"passwordResetURL": "https://<sspHost>/default/ui/v1/signin/initiatePasswordReset?x-flow-
state=UHh4OTdkQ3BWUmNySkxub2FKbHc0dHl6WklyUytBRFZmRGlBWmxNNThJRitvNzRUOWUwYkRZVDh0WTRmaHA0OGJPM1hac0dLYm12U3JxVzYxWG
},
"authCompleteUrl": "https://<sspHost>/default/oauth2/v1/authcomplete?x-flow-
state=UHh4OTdkQ3BWUmNySkxub2FKbHc0dHl6WklyUytBRFZmRGlBWmxNNThJRitvNzRUOWUwYkRZVDh0WTRmaHA0OGJPM1hac0dLYm12U3JxVzYxWG
"additional": {
"idpName": "internal",
"idpGuid": "internal",
"idpType": "internal",
"passwordResetURL": "",
"authLevelsData": {
"c95ae9c1-eca0-42e3-afa7-bba019aa82c0+malbail+1": "PASSWORD"
},
"currentFactorLevel": 1,
"amrForCompletedFactors": [
"PWD"
],
"lastObligation": "PASSWORD"
}
}
```

Where,

- flowState: The current x-flow-state to maintain the authentication session.
- userName: The user's login name.
- nextaction: Indicates that authentication is now allowed
  - If the next action is (AUTH_ALLOWED), **skip step 5 and go to step 6**.
  - If the next action is (CONSENT_AUTH), **go to step 5**.
- data: Contains additional information about the authentication result:
  - message: "Success" - Indicates the authentication status.
  - passwordResetURL: URL for password reset functionality if needed.
- authCompleteUrl: The URL to redirect to once the authentication flow is fully completed.
- additional: Contains supplementary authentication details:
  - idpName: "internal" - The identity provider name used for authentication.
  - idpGuid: "internal" - The identity provider GUID.
  - idpType: "internal" - The type of identity provider.
  - passwordResetURL: URL for password reset.
  - authLevelsData: Authentication level data mapping.
  - currentFactorLevel: 1 - The current authentication factor level achieved.
  - amrForCompletedFactors: ["PWD"] - Authentication Method Reference for completed factors.
  - lastObligation: "PASSWORD" - The last authentication obligation that was fulfilled.

5. **Consent Updater**: (Optional step) This step is performed if there are consents added in the resource server application privileges.

```
POST https://{{sspHost}}/{{apiPathTenant}}/factor/v1/ConsentUpdater
```

Consent Updater - Request and Response

Request:

```
{
    // consentGiven of true indicates that the user has given consent to the requested scopes
    "consentGiven": true
}
```

Where,

- Provide the consent as true or false for the given privileges.

Response

```
{
    "flowState": "VG5sWUlFMUFtWWlTOUNFZkhlQlZuS0tKakVsZ00vc...",
    "userName": "internaluser",
    "nextaction": "AUTH_ALLOWED",
    "deviceID": "798f053b8bc4453b8a832c2685926ae2",
    "data": {
        "consentUpdated": true
    },
    "authCompleteUrl": "https://example.com/default/oauth2/v1/authcomplete?x-flow-
state=VG5sWUlFMUFtWWl...",
    "additional": {
        "idpName": "internal",
        "idpGuid": "internal",
        "idpType": "internal",
        "passwordResetURL": "",
        "authLevelsData": {
            "da2a6ee2-c549-4265-97a8-6ff0ab47ef86+internaluser+1": "PASSWORD"
```

```
            },
            "currentFactorLevel": 1,
            "amrForCompletedFactors": [
                "PWD"
            ]
        }
    }
```

Where,

- flowState: The current x-flow-state to maintain the authentication session after consent processing.
- userName: The user's login name who provided the consent.
- nextaction: "AUTH_ALLOWED" - Indicates that authentication is now allowed after consent has been processed.
- deviceID: Unique identifier for the device used in the authentication flow.
- data: Contains information about the consent operation:
  - consentUpdated: true - Indicates whether the user's consent for the requested scopes has been successfully updated.
- authCompleteUrl: The URL to redirect to once the authentication flow is fully completed after consent processing.
- additional: Contains supplementary authentication details:
  - idpName: "internal" - The identity provider name used for authentication.
  - idpGuid: "internal" - The identity provider GUID.
  - idpType: "internal" - The type of identity provider.
  - passwordResetURL: URL for password reset.
  - authLevelsData: Authentication level data mapping.
  - currentFactorLevel: 1 - The current authentication factor level achieved.
  - amrForCompletedFactors: ["PWD"] - Authentication Method Reference for completed factors.

6. **Authentication Flow Completion and Redirect**:
   Once the user successfully authenticates, the authorization server redirects the user's browser to the pre-configured redirect URL (client application's callback URL). This redirect includes crucial information for the client to proceed with obtaining tokens.

   ```
   GET {{authCompleteUrl}}
   ```

   Authentication Flow Completion and Redirect - Request and Response
   Where,

   - authCompleteUrl - this is the configured redirect URL where the user is redirected to after the authentication completion.

   Response:The response retrieves the authorization code and the redirect URL.

   ```
   https://www.example.com?X-TRANSACTION-ID=28b8eb92-a9e3-4e93-831c-
   ed79abb69816&code=86QjvgQ3XAzcMQMNDZnOKiYB3iQ03o&session_state=4758a76a6f8490858abbb1334173d6ede2f1ecfff3ef57d0e9921
   ```

   Where,

   - X-TRANSACTION-ID: A unique identifier for the entire transaction.
   - code: This is the authorization code, a temporary, single-use code that the client exchanges for an Access Token and optionally an ID Token (example value-86QjvgQ3XAzcMQMNDZnOKiYB3iQ03o)
   - session_state: A value that represents the current session state, used to detect changes in the authentication session and provide additional security for the OAuth 2.0 flow.

7. **Exchange Authorization Code for Access Token**:
   The client application, having received the authorization code in the previous step, now makes a final request to the token endpoint. This request exchanges the authorization code for the actual Access Token (AT) and potentially an ID Token (IT). This Access Token is then used to securely access protected resources on the resource server.

   ```
   POST https://{{sspHost}}/{{apiPathTenant}}/oauth2/v1/token
   ```

Exchange Authorization Code for Access Token - Request and Response
Key Parameters (in Request Body or as Form Data)

Key	Value
code	The authorization code obtained in Step 5 (for example, 86QjvgQ3XAzcMQMNDZnOKiYB3iQ03o).

Response: A JSON object containing the final Access Token, its type, expiration, and granted scopes.


```
{
"access_token":
 "eyJ4NXQjUzI1NiI6IjdvcXZva2tnSkZ0dWd0TEhyem5QSGE0NkNIbGNGVGZmam1wSHRCY2E1T3MiLCJ4NXQiOiJJbUhCcWRMTDRnWWpiUERJOW9qWk
NnoFizuPJYgDIJohStaTQW3eI56nMvbfGsBpeUxuSokmydbD5dyWLfQS0cmbNc8omTNhPj8hauRTrpQ_0yzXJT31wo9V4IN1lqpPAlbgWE1MJTiDyGG0
ertuzm-z-QL2WM2YUDNnL1wNpOph41MtSWeg",
"token_type": "Bearer",
"expires_in": 3600,
"scope": "promote hire"
}
```


Where,

- access_token: The JWT (JSON Web Token) that grants access to protected resources.
- token_type: "Bearer" - Indicates the type of token issued. "Bearer" means the token should be included in the Authorization header as "Bearer <access_token>" when making requests to protected resources.
- expires_in: 3600 - The lifetime of the access token in seconds. In this case, 3600 seconds (1 hour) indicates when the token will expire and need to be renewed.
- scope - The permissions granted by the Access Token (for example, promote and hire).

Decoded AT:


```
{
"sub": "90390e76-4076-4885-bf3c-762f4fe88557",
"idp_name": "internal",
"user_dn": "90390e76-4076-4885-bf3c-762f4fe88557",
"user_loginid": "malbail",
"idp_guid": "internal",
"iss": "https://<sspHost>/default/",
"user_guid": "90390e76-4076-4885-bf3c-762f4fe88557",
"tid": "181b8f45-e1dc-45cf-8d5b-7811b6c4f19b",
"transactionid": "3a9379e5-0cd9-4d6d-be4e-5a7cbf8a5047",
"sid": "0789ac1d-958d-4429-adf8-57f29c6c77bc",
"sub_type": "USER",
"azp": "f8378200-bc7a-4e93-9e21-76ed2d0e81aa",
"user_loginidattrmappingname": "user_loginid",
"scope": "promote hire",
"session_ttl_secs": 21600,
"hasgroupids": false,
"exp": 1762250896,
"iat": 1762247296,
"client_name": "acmehr",
"jti": "e8744141-a66c-41b0-9678-ea43d841ced9",
"idp_type": "internal",
"client_tname": "default",
"client_tid": "181b8f45-e1dc-45cf-8d5b-7811b6c4f19b",
"ver": "1.0",
"hasgroups": false,
```

```
"azpacr": "1",
"tname": "default",
"tok_type": "AT",
"user_riskscore": 100,
"user_ipaddress": "198.51.100.255",
"user_amr": [
"PWD"
],
"aud": [
"https://hr.com"
],
"nbf": 1762247296,
"user_zfp": false,
"client_transactionid": "ssp:3a9379e5-0cd9-4d6d-be4e-5a7cbf8a5047",
"user_aal": "AAL1",
"tok_source": "AZC",
"user_acr": "urn:iam:acr:aal1"
}
```


This completes the OIDC flow, resulting in the client application possessing an Access Token that can be used to interact with the protected resource server on behalf of the authenticated user.

Reference Links

- Creating a Resource Server using the Admin Console UI
- Creating a Resource Server using REST API
- Signin UI Consent for Resource Server Applications

Securing MCP Servers with Symantec Identity Security Platform

Learn how to secure MCP servers using IDSP

Overview

This document provides comprehensive technical instructions on securing Model Context Protocol (MCP) servers using Identity Security Platform (IDSP). The process involves onboarding MCP servers as resource server applications and configuring MCP clients to access them securely through OAuth 2.0 and OpenID Connect protocols.

Model Context Protocol (MCP)

Model Context Protocol (MCP) is an emerging standard that enables Artificial Intelligence (AI) applications to interact with external systems through standardized tool interfaces. MCP servers expose executable tools, accessible resources, and prompts that AI applications can consume to perform operations, access data, and integrate with various backend systems.

Key Components of MCP:

- **MCP Server**: A service that exposes executable tools, accessible resources, and prompts through standardized HTTP/HTTPS endpoints.
  - **Tools**: Discrete functions that perform specific operations (for example, data retrieval, file operations, API calls)
  - **Resources**: Data or content accessible to authenticated clients (for example, documents, databases, external APIs)
  - **Prompts**: Reusable prompt templates that can be invoked by clients to generate consistent outputs
- **MCP Client**: AI applications or other clients that consume server capabilities by invoking tools, accessing resources, and using prompts.

MCP servers typically implement OAuth 2.0 resource server patterns, making them compatible with standard identity and access management solutions. The protocol includes a discovery mechanism through the "`https://{{IDSPHost}}/{{apiPathTenant}}/.well-known/oauth-protected-resource/mcp`" endpoint, which provides security metadata, exposed scopes, and application information.

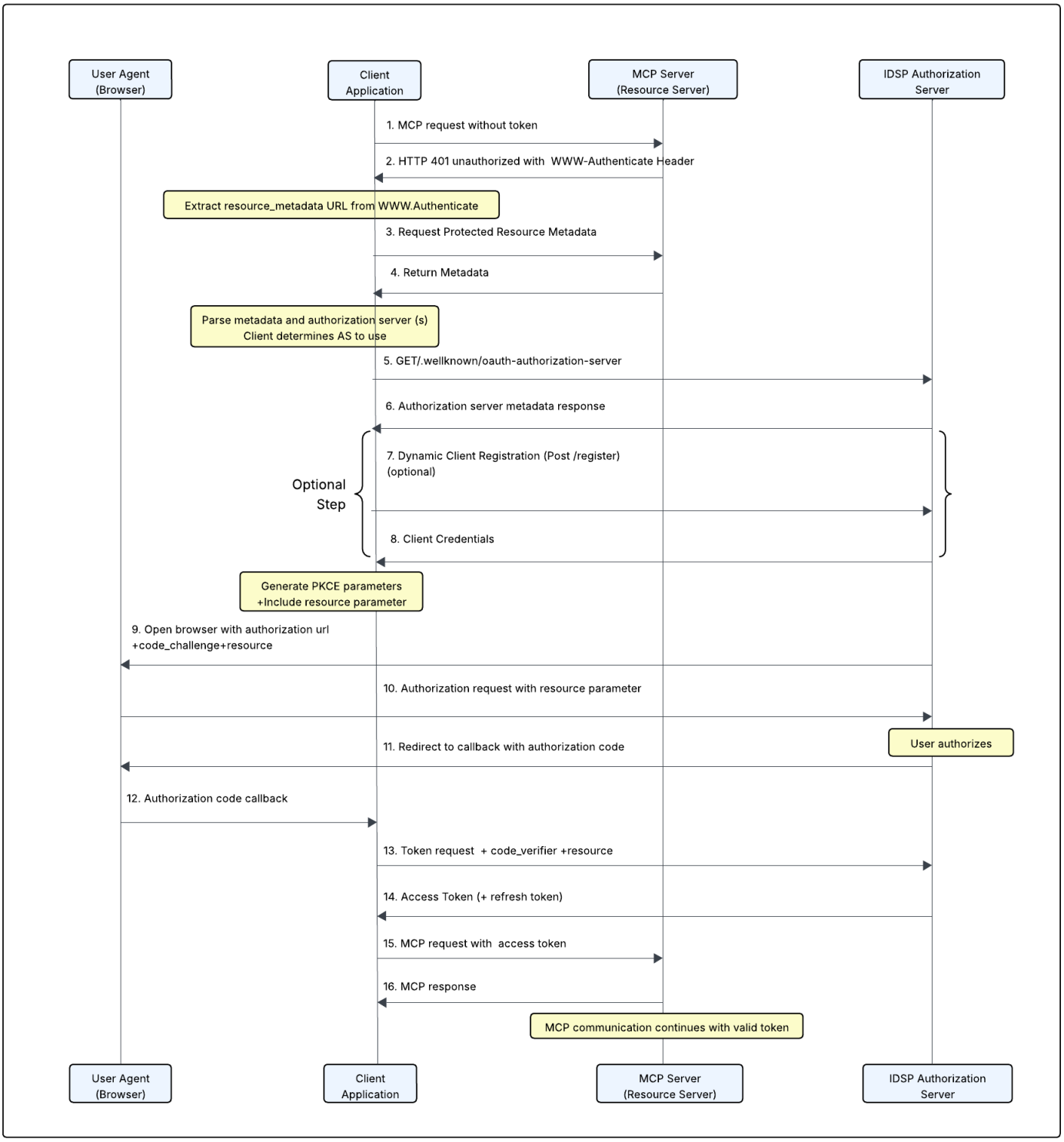
Integration Architecture

The integration of MCP servers with IDSP follows the standard OAuth 2.0 resource server pattern:

1. **MCP servers** register with IDSP as resource server applications, defining their scopes and access requirements
2. **MCP clients** authenticate with IDSP and obtain access tokens with appropriate scopes
3. **Token validation** occurs at the MCP server, which validates bearer tokens against IDSP
4. **Authorization enforcement** ensures that only authorized principals (users, groups, or client applications) can access specific tools and resources

This architecture provides enterprise-grade security while maintaining the flexibility and extensibility required for AI-driven tool execution contexts. The standardized approach ensures compatibility with existing OAuth 2.0 and OpenID Connect ecosystems.

Figure 19: Integration Architecture of MCP servers with IDSP



Gathering MCP Server Properties

Before onboarding an MCP server as a resource server application in IDSP, you need to collect the following information:

- **Resource** URL: The base URL of the MCP server (for example, https://hr.com/mcp)
- **Scopes**: List of scopes exposed by the resource (for example, hire, promote, transfer)
- **Resource Name**: Display name for the application (for example, HR)

Reference Links

- Onboarding MCP Server as a Resource Server Application
- Onboarding MCP Clients

Onboarding MCP Server as a Resource Server Application

Learn about the steps to onboard MCP server as a Resource Server application in IDSP.

Step 1. Create Resource Server Application

The first step is to provision the Model Context Protocol (MCP) server as a resource server application in IDSP using the following API and adhearing to the guidelines mentioned in the subsequent sections:

```
POST https://{{IDSPHost}}/{{apiPathTenant}}/admin/v1/Apps
```

Key Requirements:

When creating the resource server, adhere to the following requirements:

1. The resource URL from metadata should be mapped to "primaryAudience" and it must be unique across all apps.
2. The scopes in the metadata should be provisioned as "resourceServerScopes" and mark them as default. If the scopes have standard OAuth claims (like openid, profile, email, etc.), they should be ignored and should not be added as resource server scopes.
3. Mark the application as resource server app (that is, "isResourceServerApp: true")
4. Set the name of the application collected in the gathering phase.

Resource Server Application Payload:

```
{
    "status": "active",
    "name": "HR",
    "resScope": "urn:rs:hr",
    "description": "HR Resource Server Application",
    "skewTimeSecs": 0,
    "passwordAuthoritativeSource": "remote",
    "clientType": "CONFIDENTIAL",
    "allowedOpenIDScopes": [
        "openid",
        "profile",
        "groups",
        "address",
        "offline_access",
        "email"
    ],
    "redirectURIs": [
        "https://www.example.com"
    ],
    "allowedGrantTypes": [
```

```
        "implicit",
        "client_credentials",
        "urn:ietf:params:oauth:grant-type:device_code",
        "authorization_code",
        "urn:ietf:params:oauth:grant-type:jwt-bearer"     ],
    "allowedOperations": [
        "offlineit",
        "introspect",
        "onbehalfof"
    ],
    "supportedJoseHeaderParams": "x5t,x5t#s256",
    "userInfoEndpointResponseFormat": "PLAIN_JSON",
    "itEncryptionTarget": "NONE",
    "isResourceServerApp": true,
    "consentUsagePolicyUrl": "https://policy.hr.com",
    "primaryAudience": "https://hr.com/mcp",
    "resourceServerScopes": [
        {
            "scope": "promote",
            "description": "scope to promote",
            "consentDescription": "Review, approve, and manage employee promo tions within your department",
            "label": "View and manage promotions in your department",
            "consentRequired": true,
            "isDefaultScope": true
        },
        {
            "scope": "hire",
            "description": "scope to hire",
            "consentDescription": "Review, approve, and manage hiring of new employees within your
 department",
            "label": "View and manage hiring in your department",
            "consentRequired": true,
            "isDefaultScope": true
        },
        {
            "scope": "transfer",
            "description": "scope to transfer",
            "consentDescription": "Review, approve, and manage transfer of ne w employees across
 departments",
            "label": "View and manage transfers in and out of your department
",
            "consentRequired": true,
            "isDefaultScope": true
        }
    ],
    "isOidcApp": true
}
```

Step 2: Create Authorization roles

The second step is to create the roles (authorization policies) for the provisioned resource server application using the `POST "https://{{IDSPHost}}/{{apiPathTenant}}/admin/v1/AuthZPolicies"` API. Each role defines who can access which privileges.

Create a separate authorization policy for each scope or group of scopes.

Authorization Policy Payload:

```
{
   "policyName": "HRPromote",
   "description": "HR promote role",
   "displayName": "HR promote role",
   "status": "active",
   "apps": [
       {
           "id": "{{rsAppId}}"
       }
   ],
   "rules": [
       {
           "conditions": {
               "principal": {
                   "clientApp": {
                       "operator": "in",
                       "value": [
                           "<CHANGE-ME>"
                       ]
                   },
                   "user": {
                       "operator": "in",
                       "value": [
                           "<CHANGE-ME>"
                       ]
                   },
                   "group": {
                       "operator": "in",
                       "value": [
                           "<CHANGE-ME>"
                       ]
                   }
               }
           },
           "result": {
               "effect": "grant",
               "msg": "HR promote role",
               "privileges": [
                   "promote"
               ]
           }
       }
   ]
}
```

Where,

- Replace the {{rsAppId}} with the actual resource server application ID returned from Step 1.
- Configure the principal conditions (clientApp, user, or group) based on your access control requirements.

Step 3: Grant Roles to Principals

Grant the roles to principals (users or client applications) using the `POST "https://{{IDSPHost}}/{{apiPathTenant}}/admin/v1/AppRolesDelegator"` API. Prompt the user to provide the username or client application ID for which the roles needs to be granted.

Sample Grant Payload for User

```
{
    "id": "{{promotePolicyId}}",
    "principal": {
        "user": {
            "operator": "in",
            "value": [
                "{{userLoginName}}"
            ]
        }
    }
}
```

Sample Grant Payload for Client Application

```
{
    "id": "{{promotePolicyId}}",
    "principal": {
        "clientApp": {
            "operator": "in",
            "value": [
                "{{clientAppId}}"
            ]
        }
    }
}
```

Where,

- Replace {{promotePolicyId}} with the actual authorization policy ID returned from Step 2, and {{userLoginName}} or {{clientAppId}} with the appropriate principal identifier.

Step 4: Access Instructions

After onboarding, ensure that the newly secured MCP server can be accessed from an MCP client. The URL should match with the resource in the OAuth protected resource endpoint `"https://{{mcpserver-host}}/{{apiPathTenant}}/.well-known/oauthprotected-resource/mcp"` . After onboarding, ensure that the newly secured MCP server can be accessed from an MCP client. The URL should match with the resource in the OAuth protected resource endpoint

Example of Client Configuration:

```
{
    "mcpServers": {
        "appname": {
            "url": "https://hr.com/mcp"
        }
    }
}
```

Reference Links

- Onboarding MCP Clients
- Securing MCP Servers with Symantec Identity Security Platform

Onboarding MCP Clients

Learn how to onboard the MCP clients in Symantec Identity Security Platform

Model Context Protocol (MCP) clients are applications that connect to MCP servers to fetch the list of tools, resources, and prompts exposed by the server.

Clients can be of two types:

- Trusted client
- On behalf of (OBO) client

Trusted Clients

Trusted clients integrate with the MCP server directly. These are typically created or registered in Symantec Identity Security Platform offline with necessary privileges.

Characterisitcs of Trusted Clients:

- Clients integrate with the MCP server directly
- Typically created or registered in IDSP offline with necessary privileges
- Access to these clients are secured by various conditions like IP address, country, risk, etc.
- Grant access to the client application for the MCP server roles
- Use client_credentials or JWT bearer grant types

API Endpoint:

```
POST https://{{IDSPHost}}/{{apiPathTenant}}/admin/v1/Apps
```

Trusted Client Application Payload:

```
{
   "status": "active",
   "name": "acmehr",
   "description": "Client Application to access HR Resource server",
   "clientType": "CONFIDENTIAL",
   "allowedOpenIDScopes": [
       "openid",
       "email",
       "profile",
       "urn:iam:sso:idprop"
   ],
   "redirectURIs": [
       "https://www.example.com"
   ],
   "allowedGrantTypes": [
       "urn:ietf:params:oauth:grant-type:jwt-bearer",
       "client_credentials"
   ],
   "allowedOperations": [
       "introspect",
       "offlineit"
```

```
    ],
    "skewTimeSecs": 0,
    "userInfoEndpointResponseFormat": "PLAIN_JSON",
    "supportedJoseHeaderParams": "x5t,x5t#s256"
}
```

Grant Role to Client Applications:

After creating the trusted client, grant access to the client application for the MCP server roles using the `POST "https://{{IDSPHost}}/{{apiPathTenant}}/admin/v1/AppRolesDelegator"` API with the client application grant payload shown in Step 3.

Public or On-Behalf-Of (OBO) Clients

These clients are used on behalf of users to connect to MCP servers. Ensure that the users are granted access to the MCP server or resource server application roles. OBO clients are typically registered by Dynamic Client Registration (DCR).

DCR can handle the following scenarios:

- Single Client Registration
- Template Client for Multiple Registration

Single Client Registration : If only one client needs to be created to access the MCP server, create the client application with the following metadata: `urn:iam:app:useAppAsRegisteredClientInDcr=true`

API Endpoint

```
POST https://{{IDSPHost}}/{{apiPathTenant}}/admin/v1/Apps
```

OBO Client Payload for Single Registration Payload

```
{
    "status": "active",
    "name": "Cursor",
    "description": " ",
    "skewTimeSecs": 0,
    "passwordAuthoritativeSource": "remote",
    "clientType": "PUBLIC",
    "allowedOpenIDScopes": [
        "openid"
    ],
    "redirectURIs": [
        "https://www.example.com",
        "cursor://anysphere.cursor-mcp/oauth/user-isp/callback",
        "cursor://anysphere.cursor-mcp/oauth/user-acme/callback",
        "cursor://anysphere.cursor-mcp/oauth/user-idsp/callback",
        "https://www.cursor.com",
        "cursor://anysphere.cursor-retrieval/oauth/user-sisp/callback"
    ],
    "allowedGrantTypes": [
        "refresh_token",
        "authorization_code"
    ],
    "allowedOperations": [
        "onbehalfof"
    ],
    "userInfoEndpointResponseFormat": "JWT",
```

```
    "itEncryptionTarget": "NONE",
    "samlEntityIdStripe": "1f7731a0-c9ed-45e3-91f6-8f1cf97ee04b",
    "metadata": [
        {
            "name": "agent",
            "value": "cursor"
        },
        {
            "name": "urn:iam:app:useAppAsRegisteredClientInDcr",
            "value": "true"
        }
    ],
    "isOidcApp": true
}
```

Template Client for Multiple Registrations : If the client always invokes the DCR with the same name and IDSP needs to create the client with a unique name, create a client application before enabling the MCP clients with the following metadata: `urn:iam:app:useAppAsTemplateClientinDcr=true`

API Endpoint:

```
POST https://{{IDSPHost}}/{{apiPathTenant}}/admin/v1/Apps
```

OBO Client Payload for Template Client :

```
{
    "status": "active",
    "name": "MCP Inspector",
    "description": " ",
    "skewTimeSecs": 0,
    "passwordAuthoritativeSource": "remote",
    "clientType": "PUBLIC",
    "allowedOpenIDScopes": [
        "address",
        "phone",
        "openid",
        "offline_access",
        "profile",
        "groups",
        "email"
    ],
    "redirectURIs": [
        "http://localhost:6274/oauth/callback",
        "http://localhost:6274/oauth/callback/debug"
    ],
    "allowedGrantTypes": [
        "refresh_token",
        "authorization_code"
    ],
    "allowedOperations": [
        "introspect",
        "onbehalfof"
    ],
    "userInfoEndpointResponseFormat": "JWT",
    "itEncryptionTarget": "NONE",
```

```
    "samlEntityIdStripe": "3c45d411-b26b-4ed9-90c9-a0c0dc35d2d0",
    "metadata": [
        {
            "name": "urn:iam:app:useAppAsTemplateClientinDcr",
            "value": "true"
        },
        {
            "name": "agent",
            "value": "inspector"
        }
    ],
    "label": "MCP Inspector",     "isOidcApp": true
}
```

NOTE
Creating the authentication policy with the app metadata that is configured in the template applications ensures that the policy is applied to all newly created clients with the same template application.

Create Authentication Policy

Create the authentication policy for the client application based on the application metadata. If fine-grained authentication control is necessary, include other conditions supported by authentication policies.

API Endpoint:

```
POST https://{{IDSPHost}}/{{apiPathTenant}}/admin/v1/AuthPolicies
```

Authentication Policy Payload:

```
 {
    "policyName": "Cursor AuthN Policy",
    "description": " ",     "status": "active",
    "matchAnyApp": true,
    "rules": [
        {
            "conditions": {
                "principal": {
                    "user": {
                        "operator": "in",
                        "value": [
                            "*"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "authenticate"
                    ]
                },
                "context": {
                    "operator": "expression",
                    "value": "(${appmetadata.agent} eq cursor)"
                }
            },
            "result": {
```

```
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD:1"
                ],
                "msg": "Cursor AuthN Rule",
                "mfaFrequency": "OnceForTrustedDevice"
            }
        }
    ]
}
```

Grant Roles to OBO Client Users

For OBO clients, ensure users are granted access to the MCP server roles using the `POST "https://{{IDSPHost}}/{{apiPathTenant}}/admin/v1/AppRolesDelegator"` API . Use the user grant payload from Step 3 to grant roles to users who uses the OBO client.

Reference Links

- Onboarding MCP Server as Resource Server Application
- Securing MCP Servers with IDSP

Symantec Identity Security Platform as CAEP Transmitter/Receiver

Configure the ID LCM WebHook to deliver Continuous Access Evaluation Protocol (CAEP) events to a SCIM endpoint ensuring the endpoint can receive events and is protected by either Basic Auth or OAuth2 security models.

IDSP as CAEP Transmitter

Symantec Identity Security Platform delivers Continuous Access Evaluation Protocol (CAEP) events to the ID LCM WebHook, which is modeled in IDSP as a SCIM endpoint represented by "SCIM Identity Store" resource.

The ID LCM Web Hook is one of the properties under Service Configuration, called "idEventHookConfig".

To configure the ID LCM WebHook, use the following API:

```
PATCH https://{{sspHost}}/{{apiPathTenant}}/admin/v1/Configs
```

Update the Global/Tenant parameter, "idEventHookConfig":

```
{
"name": "idEventHookConfig",
"value": "<SCIMconfigs-id-or-name>, event1, event2,..."
}
```

Where,

- A value that starts with "#" implies a commented out / disabled configuration.
- A SCIMconfigs resource matching given id or name is validated against this event:
  - session.revoke.success - sends a Continuous Access Evaluation Protocol (CAEP) message indicating that the user session is revoked, and includes a list of applications to be notified.

The ID LCM Web Hook is expected to expose "**/Caep"** endpoint to receive CAEP events containing the prescribed payload.

This endpoint is assumed to be protected by one of the following security models:

- Basic Auth security (ID/Secret)
- OAuth2 security with Bearer Token obtained via Client Credentials OAuth grant using Client ID/Secret or Client Assertion

IDSP as CAEP Receiver

The IDSP functions as a CAEP (Continuous Access Evaluation Protocol) listener by exposing the CAEP listener at the following URL:

```
https://{{sspHost}}/{{apiPathTenant}}/identity/v1/Caep
```

This endpoint allows IDSP to receive and process CAEP events.

Auditing in Symantec Identity Security Platform

Learn about the different types of authentication and administration auditing events logged in Symantec Identity Security Platform.

This topic contains the following sections:

- Authentication Events
- Administration Events

Authentication Events

The following table lists different types of authentication auditing events along with their corresponding event IDs that are logged in IDSP.

Table 85: Authentication Events and Event Ids in IDSP

Authentication Event	eventId (Log messages with type=audit)
Initiate Authentication	auth-mgr.auth.initiate
Authentication Successful	auth-mgr.auth.success
Authentication Denied	auth-mgr.auth.failure
Authentication Abandoned	auth-mgr.auth.abandoned
Factor Authentication Successful	factor.me.auth.complete.success
Factor Authentication Failed	factor.me.auth.complete.failure
Factor Auth Initialization Successful	factor.me.auth.initiate.success
Factor Auth Initialization Failed	factor.me.auth.initiate.failure
Factor Registration Successful	factor.me.register.complete.success
Factor Registration Failed	factor.me.register.complete.failure
Factor Register Initialization Successful	factor.me.register.initiate.success
Factor Register Initialization Failed	factor.me.register.initiate.failure
Token issue successful	azserver.token.success
Token issue failed	azserver.token.failure
Token revocation successful	azserver.token.revoke.success
Initiate Authorization flow	azserver.authorize.initiate

Authentication Event	eventId (Log messages with type=audit)
Successful Authorization flow	azserver.authorize.complete.success
Failure Authorization flow	azserver.authorize.complete.failure
Risk evaluation success	risk.eval.succes
Risk evaluation failed	risk.eval.failure
Post Risk evaluation success	risk.posteval.success
Post Risk evaluation failed	risk.posteval.failure
User lock success	factor.me.auth.lock.success
User lock failure	factor.me.auth.lock.failure
User unlock success	factor.me.auth.unlock.success

Administration Events

The following table lists different types of administration audit events along with their corresponding event IDs that are logged in IDSP.

For each successful update operation (admin.<resource>.update.success), the operation also logs two additional entries:

- The object before the update (`-ext-before` extension)
- The object after the update (`-ext-after` extension)

Table 86: Administration Events

Administration Event	eventId (Log messages with type=audit)
Application Events	admin.app.create.success
	admin.app.create.failure
	admin.app.activate.success
	admin.app.activate.failure
	admin.app.deactivate.success
	admin.app.deactivate.failure
	admin.app.delete.success
	admin.app.delete.failure
	admin.app.update.success
	admin.app.update.failure
Authentication Policy Events	admin.authpolicy.create.success
	admin.authpolicy.create.failure
	admin.authpolicy.delete.failure
	admin.authpolicy.delete.success
	admin.authpolicy.update.failure
	admin.authpolicy.update.success
Authorization Policy Events	admin.authzpolicy.create.success
	admin.authzpolicy.update.success
	admin.authzpolicy.delete.success

Administration Event	eventId (Log messages with type=audit)
	admin.authzpolicy.create.failure
	admin.authzpolicy.update.failure
	admin.authzpolicy.delete.failure
Certificate Mgmt Events	admin.cert.create.failure
	admin.cert.create.success
	admin.cert.delete.failure
	admin.cert.delete.success
	admin.cert.update.failure
	admin.cert.update.success
Identity Provider Configuration Events	admin.identityprovider.create.failure
	admin.identityprovider.create.success
	admin.identityprovider.delete.failure
	admin.identityprovider.delete.success
	admin.identityprovider.update.failure
	admin.identityprovider.update.success
	admin.identityprovidermetadata.import.failure
	admin.identityprovidermetadata.import.success
	admin.identityprovidermetadata.update.failure
	admin.identityprovidermetadata.update.success
LDAP Configuration Events	admin.ldapconfig.create.failure
	admin.ldapconfig.create.success
	admin.ldapconfig.delete.failure
	admin.ldapconfig.delete.success
	admin.ldapconfig.update.failure
	admin.ldapconfig.update.success
Credential Mgmt Events triggered by User Self-Service Operations	admin.me.cred.delete.failure
	admin.me.cred.delete.success
	admin.me.cred.update.failure
	admin.me.cred.update.success
	admin.me.cred.regsiter.successs
	admin.me.cred.register.failure
Credential Mgmt Events triggered by Administrator	admin.user.cred.delete.failure
	admin.user.cred.delete.success
	admin.user.cred.update.failure
	admin.user.cred.update.success
Message Events	admin.message.create.failure
	admin.message.create.success
	admin.message.delete.failure

Administration Event	eventId (Log messages with type=audit)
	admin.message.delete.success
	admin.message.update.failure
	admin.message.update.success
Message Provider Events	admin.messageprovider.create.failure
	admin.messageprovider.create.success
	admin.messageprovider.delete.failure
	admin.messageprovider.delete.success
	admin.messageprovider.update.failure
	admin.messageprovider.update.success
Password Encryption Events	admin.password.encrypt.failure
	admin.password.encrypt.success
Token revocation by admin	admin.revoke.success
Locale Events	admin.locale.create.success
	admin.locale.update.success
	admin.locale.delete.success
	admin.locale.create.failure
	admin.locale.update.failure
	admin.locale.delete.failure
Factor transaction signing Events	factor.me.transactionsigning.initiate.success
	factor.me.transactionsigning.initiate.failure
	factor.me.transactionsigning.complete.success
	factor.me.transactionsigning.complete.failure
Opa Events	opa.policyload.success
Identity scim User Events	identity.scim.user.create.success
	identity.scim.user.create.failure
	identity.scim.user.update.success
	identity.scim.user.update.failure
	identity.scim.user.delete.success
	identity.scim.user.delete.failure
Identity scim group Events	identity.scim.group.members.update.success
	identity.scim.group.create.success
	identity.scim.group.create.failure
	identity.scim.group.update.success
	identity.scim.group.update.failure
	identity.scim.group.delete.success
	identity.scim.group.delete.failure
Identity scim me update Events	identity.scim.me.update.success
	identity.scim.me.update.failure

Administration Event	eventId (Log messages with type=audit)
Identity scim bulk update Events	identity.scim.bulk.success
	identity.scim.bulk.failure
Identity scim user account lock/enable Events	identity.scim.user.account.lock.success
	identity.scim.user.account.unlock.success
	identity.scim.user.account.enable.success
	identity.scim.user.account.disable.success
Identity me user Events	identity.me.unlock.success
	identity.me.lock.success
Identity scim password change mandatory Events	identity.scim.user.account.passwordmustchange.set.success
	identity.scim.user.account.passwordmustchange.clear.success
Identity scim password update Events	identity.scim.user.password.update.success
	identity.scim.user.password.update.failure
Identity scim update Events	identity.scim.group.members.update.success
	identity.scim.user.groups.update.success
Identity password reset Events	identity.me.password.reset.initiate.success
	identity.me.password.reset.initiate.failure
	identity.me.password.reset.complete.success
	identity.me.password.reset.complete.failure
Identity password update Events	identity.me.password.update.success
	identity.me.password.update.failure
Secret Events	admin.secret.update
User Events	admin.user.lock.success
	admin.me.user.unlock.success
	admin.user.unlock.success

Resource Deletion in Symantec Identity Security Platform

This document outlines the "Complete resource deletion process" focusing on the force delete functionality using a scheduler for cleanup.

This feature introduces a "force delete" option for specific Symantec Identity Security Platform resources, allowing administrators to immediately delete in-use objects. Instead of preventing deletion, a record of the forcefully deleted resource is created in the database. The "resourcesCleanupScheduler" job cleans up any dangling references to the deleted resource in other operational resources.

Existing API Changes

The existing DELETE APIs for the supported resources are enhanced to accept an optional "**force=true**" query parameter.

- /admin/v1/Apps/{{id}} (DELETE)
- /admin/v1/IdentityProviders/{{id}} (DELETE)
- /admin/v1/CustomProviders/{{id}} (DELETE)
- /admin/v1/ResourceGroups/{{ApplicationGroupId}} (DELETE)
- /admin/v1/AuthZPolicies/{{azPolicyId}} (DELETE)

Changes in the Referencing Objects Helper API

The ReferencingObjectsHelper API (/admin/v1/ReferencingObjectsHelper) has been enhanced to fetch the complete list of Symantec Identity Security Platform resources referring to a given object, which can be used by the Admin or the UI to render the list of dependent objects on the UI to the admin user choosing to force the delete, or the client can choose to force the delete right away.

The "/ReferencingObjectsHelper" API response includes a new parameter, "**allowResourceToBeForcefullyDeletable**", to indicate whether a resource's deletion might violate any system constraints. If this parameter is false, then forceful deletion of that resource is not permitted. The response from "/ReferencingObjectsHelper" API looks as follows -

```
[
{
"id": "304b6c70-e752-4c73-afaa-24d5ba4126e1",
"name": "Sample app group policy",
"type": "urn:iam:res:authnpolicy",
"allowResourceToBeForcefullyDeletable": true
}
]
```

See Referencing Objects Helper API.

Handling Constraints During Force Delete

Force deletion of Apps, AppGroups, or principals may result in policy constraint violations. During force delete operations, multiple existing constraints can be affected. The table below details each constraint type and its corresponding handling mechanism during force deletion.

Table 87: Handling Constraints during force delete in IDSP

Scenario	Handling during force delete	Policy Types applicable	Impact after forceful delete
Single App or AppGroup in a Policy with Force Delete, when "matchAnyApp=false" When a poli cy has only one App or AppGroup assigned to it and the "matchAnyApp" attribute is set to false, the policy requires at least one application to be configured for proper evaluation. Normally, deleting this single app would violate this constraint since it would leave the policy with zero apps while "matchAnyApp=false" demands at least one.	**Force delete is permitted despite the constraint violation.** The system allows the force deletion to proceed despite this constraint violation. However, as a consequence, any policies that are left with zero apps and have "matchAnyApp=false" attribute become non-functional, hence do not get evaluated.	• AuthN Policy • IDP Policy • Risk Policy	Policies with "matchAnyApp=false" and zero apps after force delete do not get evaluated.
Single IDP in Policy with Force Delete When an IDP policy has only one Identity Provider (IDP) configured and that IDP is forcefully deleted, it creates a constraint violation. IDP policies require at least one identity provider to function properly, and removing the sole IDP leaves the policy without any valid identity providers for user authentication routing.	**Force delete is permitted despite the constraint violation.** The system allows the IDP deletion to proceed but renders the affected IDP policies non-functional, as they cannot evaluate authentication requests without any configured identity providers.	• IDP Policy	IDP policies with zero identity providers after force delete do not get evaluated.

Scenario	Handling during force delete	Policy Types applicable	Impact after forceful delete
Single Principal in AuthN Policy with Required Principal Setting When the tenant setting "principalConditionRequiredInPolicy" is enabled (set to 'true'), AuthN or Access policies must maintain at least one principal for proper evaluation. Attempting to force delete the sole principal configured in an AuthN policy violates this mandatory constraint, as it would leave the policy without any principals while the tenant requires principal conditions to be present.	Force delete is **not allowed** and the operation is blocked. The system prevents the deletion and returns a descriptive error message to inform administrators of the constraint violation and affected policy. **Error Message**: Resource can't be deleted as it is the only Principal in AuthN policy 'policy-name' (error-code - 8000145)	• AuthN Policy	

Immediate Deletion of Resources and Scheduler Record Creation

When a DELETE API is invoked with "force=true":

- **Immediate Resource Deletion**: The target resource (e.g., Application, IDP) are immediately deleted.
- **Record Creation in Database**: A new record is created in the database.

Upon immediate deletion, the core operations, such as policy fetches, no longer return the deleted resource object. This ensures that the deleted resource is effectively hidden from operational flows immediately, even while dangling references may temporarily persist until the scheduler performs its cleanup.

The scheduler performs the following cleanup operations for each resource type:

- **Application**: Remove outdated application references in AuthN Policies, AuthZ Policies (clientApps as Principals, Apps), and IDP Policies.
- **IDP**: Remove outdated IDP references in AuthN Policies (IDPs in policy obligations, IDPs in policy principals), AuthZ Policies (User/Group in policy principals), and IDP Policies (IDPs in policy results).
- **Custom Provider**: Remove outdated provider config references in AuthN Policies (custom authenticator, custom password, SM password as obligations) and AuthZ Policies (SM provider as roles).
- **Application Group**: Remove unresolved app group references in AuthN Policies (app groups as subjects), IDP Policies (app groups as subjects), Risk Policies (app groups as subjects), and AuthZ Policies (app groups as Principals).
- **AppRole**: Remove unresolved appRole references in AuthN Policies (Access) (appRole as Principal).

Reference Links

- Scheduler Job

Symantec Identity Security Platform SDKs

Symantec Identity Security Platformprovides a variety of Software Development Kits (SDKs) to promote the development of custom solutions. These SDKs help in creating and deploying applications that integrate with IDSP. Each SDK offers the resources you must create an app, including code samples.

SDK Location

The iOS SDK package can be downloaded from the Support Portal > My Downloads (Search for IDSP in the portal and select the appropriate version number).

This module includes the following topics:

- IDSP Client Release Notes
- IDSP iOS SDK Framework
- IDSP Android SDK
- IDSP Web SDKs
- VIP SDKs

Client Release Notes

View information about the Symantec Identity Security Platform client releases and release packages.

Release information about IDSP iOS and Android clients are covered in this section.

This page includes Release Notes for IDSP iOS and Android SDKs:

- iOS Client Release Notes
- Android Client Release Notes

IDSP iOS Client Release Notes

Table 88: IDSP iOS Client Release Notes

Client	Version	Release Notes	Reference Document
iOS SDK	3.1.5	3.1.5 iOS SDK Release Notes	This release includes bug fixes, and enhancements.
iOS SDK	3.1.4	3.1.4 iOS SDK Release Notes	This release includes bug fixes, and enhancements. **Note: SDK Version Compatibility Requirement**: **Note:** Client applications must upgrade to SDK version 3.1.3 or higher before upgrading the IDSP server to version 4.0 to ensure FIDO registration functionality remains intact and prevent compatibility issues with older SDK versions.
iOS SDK	3.1.3	3.1.3 iOS SDK Release Notes	This release includes bug fixes, and enhancements.
iOS SDK	3.1.2	3.1.2 iOS SDK Release Notes	This release includes bug fixes, and enhancements.
iOS SDK	3.1.1	3.1.1 iOS SDK Release Notes	This release has incremental updates to streamline the SDK functionality. No new functional changes were introduced in this release.
iOS SDK	3.1	3.1 iOS SDK Release Notes	This release has incremental updates to streamline the SDK functionality. No new functional changes were introduced in this release.
iOS SDK	3.0	3.0 iOS SDK Release Notes	This release introduces PASSKEY as an authentication factor, similar to the "SECURITYKEY" and "FIDO" (Biometric) factors.
iOS SDK	2.3.6	2.3.6 iOS SDK Release Notes	This release includes bug fixes, enhancements, and new error codes (809-822) for FIDO WebAuthN functionality. See iOS SDK Error Codes.
iOS SDK	2.3.5	2.3.5 iOS SDK Release Notes	This release includes new error codes for FIDO WebAuthN functionality and SDK initialization. See iOS SDK Error Codes.
iOS SDK	2.3.4	2.3.4 iOS SDK Release Notes	This release includes the necessary privacy manifest file as mandated by Apple.com. See IDSP Manifest File .

Client	Version	Release Notes	Reference Document
iOS SDK	2.3.3	2.3.3 iOS SDK Release Notes	In IDSP version 3.1 and later, when data is encrypted in transit, the server expects the following encryption formats: • PKI encrypted field with the {x5c} prefix (For example, - in-transit encrypted password data) • Symmetric encrypted field with the {enc} prefix (For example, - encrypted device signature data) These changes mean that IDSP iOS SDK v2.3.3 will not function with a IDSP server version lower than 3.1. Therefore, it is necessary to upgrade to IDSP version 3.1 before using IDSP iOS SDK v2.3.3.
iOS SDK	2.3.2	2.3.2 iOS SDK Release Notes	IDSP iOS Core SDK • Setting Up the Network Request Object • Initiate Authentication • Getting the list of fields to be encrypted in the authentication flow – Support for proxy URL configuration to get the list of fields that are required for encryption

For package download details, see Client SDK Packaged Links: Download Locations

IDSP Android Client Release Notes

Table 89: IDSP Android Client Release Notes

Client	Version	Release Notes	Reference Document
Android SDK	3.0.3	Android SDK 3.0.3	In this release, bug fixes and enhancements have been made.
Android SDK	3.0.2	Android SDK 3.0.2	In this release, bug fixes and enhancements have been made.
Android SDK	3.0.1	Android SDK 3.0.1	In this release, bug fixes and enhancements have been made.
Android SDK	3.0	Android SDK 3.0	This release introduces PASSKEY as an authentication factor, similar to the "SECURITYKEY" and "FIDO" (Biometric) factors.
Android SDK	2.3.3	Android SDK 2.3.3	IDSP Android SDK v2.3.3 addresses some potential vulnerabilities.

Client	Version	Release Notes	Reference Document
Android SDK	2.3.2	2.3.2 Android SDK	In IDSP version 3.1 and later, when data is encrypted in transit, the server expects the following encryption formats: • PKI encrypted field with {x5c} prefix - (For example- in transit encrypted password data) • Symmetric encrypted field with {enc} prefix - (For example- encrypted device signature data) These changes mean that IDSP Android SDK v2.3.2 will not work with function with an IDSP server version lower than 3.1. Therefore, it is necessary to upgrade to IDSP version 3.1 before using IDSP Android SDK v2.3.2.
Android SDK	2.3.1	2.3.1 Android SDK	IDSP Android Core SDK • Initiate Authentication • Getting the list of fields to be encrypted in the authentication flow – Support for Proxy URL Configuration

For package download details, see Client SDK Packaged Links: Download Locations

iOS SDK 3.1.5

View information about the iOS Symantec Identity Security Platform 3.1.5 release.

This topic includes the following sections:

- Release Information
- New Features and Improvements
- IDSP iOS SDK Kit Details

Release Information

- Release Title: IDSP iOS SDK
- Release Version: 3.1.5
- Preceding Version: 3.1.4
- Minimum OS Requirement: iOS v15.0 and above
- Release Date: February, 2026

> **NOTE**
> **SDK Version Compatibility Requirement**: Client applications must upgrade to SDK version 3.1.3 or higher before upgrading the IDSP server to version 4.0 to ensure FIDO registration functionality remains intact and prevent compatibility issues with older SDK versions.

New Features and Improvements

- Addressed bug fixes and enhancements.

IDSP iOS SDK Package Details

The latest iOS SDK package can be downloaded from the Support Portal > My Downloads (Search for Symantec Identity Security Platform in the portal and select the appropriate version number).

iOS SDK 3.1.4

View information about the iOS Symantec Identity Security Platform 3.1.4 release.

This topic includes the following sections:

- Release Information
- New Features and Improvements
- IDSP iOS SDK Kit Details

Release Information

- Release Title: IDSP iOS SDK
- Release Version: 3.1.4
- Preceding Version: 3.1.3
- Minimum OS Requirement: iOS v15.0 and above
- Release Date: December, 2025

> **NOTE**
> **SDK Version Compatibility Requirement**: Client applications must upgrade to SDK version 3.1.3 or higher before upgrading the IDSP server to version 4.0 to ensure FIDO registration functionality remains intact and prevent compatibility issues with older SDK versions.

New Features and Improvements

- Introduced FIDO SDK configurations to skip the passcode fallback option when biometric authentication fails. When these configurations are not specified or used with default settings, the SDK maintains existing behavior with no impact on the current FIDO registration and authentication user experience.
- Enhanced error handling for passkey registration and passkey authentication.
- Addressed few other bug fixes and enhancements.

IDSP iOS SDK Package Details

The latest iOS SDK package can be downloaded from the Support Portal > My Downloads (Search for VIP Authentication Hub in the portal and select the appropriate version number).

iOS SDK 3.1.3

View information about the iOS Symantec Identity Security Platform 3.1.3 release.

This topic includes the following sections:

- Release Information
- New Features and Improvements
- IDSP iOS SDK Kit Details

Release Information

- Release Title: IDSP iOS SDK
- Release Version: 3.1.3
- Preceding Version: 3.1.2
- Minimum OS Requirement: iOS v15.0 and above
- Release Date: November, 2025

NOTE
SDK Version Compatibility Requirement: Client applications must upgrade to SDK version 3.1.3 or higher before upgrading the IDSP server to version 4.0 to ensure FIDO registration functionality remains intact and prevent compatibility issues with older SDK versions.

New Features and Improvements

- When a user attempts to register a passkey that is already present on the same device and platform, both the enrollment and authentication flows return an identical error code. This behavior has been addressed by enhancing the error handling for passkey registration and authentication. (DE648771)
- Addressed few other bug fixes and enhancements.

IDSP iOS SDK Package Details

The latest iOS SDK package can be downloaded from the Support Portal > My Downloads (Search for VIP Authentication Hub in the portal and select the appropriate version number).

iOS SDK 3.1.2

View information about the iOS Symantec Identity Security Platform 3.1.2 release.

This topic includes the following sections:

- Release Information
- New Features and Improvements
- IDSP iOS SDK Kit Details

Release Information

- Release Title: IDSP iOS SDK
- Release Version: 3.1.2
- Preceding Version: 3.1.1
- Minimum OS Requirement: iOS v15.0 and above
- Release Date: September, 2025

New Features and Improvements

- Duplicate passkey registrations occured for an application on a given device when a passkey for the same user and domain is already registered, particularly when iCloud Keychain is used for passkey synchronization. This leads to an undesirable state where multiple identical passkeys are registered, impacting user experience. To resolve this duplication issue, "**exclude credential Id list**" has been added.
  This resolution works on devices that run on iOS 17.0 and above.(DE644244)
- Addressed few other bug fixes and enhancements.

IDSP iOS SDK Package Details

The latest iOS SDK package can be downloaded from the Support Portal > My Downloads (Search for VIP Authentication Hub in the portal and select the appropriate version number).

iOS SDK 3.1.1

View information about the iOS Symantec Identity Security Platform 3.1.1 release.

This topic includes the following sections:

- Release Information
- New Features and Improvements
- IDSP iOS SDK Kit Details

Release Information

- Release Title: IDSP iOS SDK
- Release Version: 3.1.1
- Preceding Version: 3.1
- Minimum OS Requirement: iOS v15.0 and above
- Release Date: July, 2025

New Features and Improvements

Incremental updates were made to streamline the SDK functionality. No new functional changes were introduced in this release.

IDSP iOS SDK Package Details

The latest iOS SDK package can be downloaded from the Support Portal > My Downloads (Search for IDSP in the portal and select the appropriate version number).

iOS SDK 3.1

View information about the iOS Symantec Identity Security Platform 3.1 release.

This topic includes the following sections:

- Release Information
- New Features and Improvements
- IDSP iOS SDK Kit Details

Release Information

- Release Title: IDSP iOS SDK
- Release Version: 3.1
- Preceding Version: 3.0
- Minimum OS Requirement: iOS v15.0 and above
- Release Date: February, 2025

New Features and Improvements

Incremental updates were made to streamline the SDK functionality. No new functional changes were introduced in this release.

IDSP iOS SDK Package Details

The latest iOS SDK package can be downloaded from the Support Portal > My Downloads (Search for IDSP in the portal and select the appropriate version number).

iOS SDK 3.0

View information about the iOS Symantec Identity Security Platform 3.0 release.

This topic includes the following sections:

- Release Information
- New Features and Improvements
- IDSP iOS SDK Kit Details

Release Information

- Release Title: IDSP iOS SDK
- Release Version: 3.0
- Preceding Version: 2.3.6
- Minimum OS Requirement: iOS v15.0 and above
- Release Date: November, 2024

New Features and Improvements

In this release, we have introduced Passkey as an authentication factor, similar to the "Security Key" and "FIDO" (Biometric) factors. The "fidoCredType" parameter in the FIDO Registration (".../factor/v1/FIDORegChallengeGenerator") API differentiates the type of FIDO factor being used during registration. The "fidoCredType" property now supports the following values: `PASSKEY` , `FIDO (Biometric Authentication)` , and `SECURITYKEY` .

NOTE

- In this release, to register a new passkey, you must update or create an Authentication policy which contains the passkey as a factor.
- The "`useFQDN`" setting which was introduced in IDSP v2.2, is now used solely for backward compatibility. Therefore, it is only applicable during authentication (using the ".../factor/v1/FIDOAuthChallengeGenerator" and ".../factor/v1/FIDOAuthChallengeVerifier" APIs) for devices that have already registered passkey.

See Enabling Passkey.

IDSP iOS SDK Package Details

The latest iOS SDK package can be downloaded from the Support Portal > My Downloads (Search for IDSP in the portal and select the appropriate version number).

iOS SDK 2.3.6

View information about the iOS Symantec Identity Security Platform 2.3.6 release.

This topic includes the following sections:

- Release Information
- New Features and Improvements
- IDSP iOS SDK Kit Details

Release Information

- Release Title: IDSP iOS SDK
- Release Version: 2.3.6
- Preceding Version: 2.3.5
- Minimum OS Requirement: iOS v15.0 and above
- Release Date: October, 2024

New Features and Improvements

In this release, bug fixes and enhancements have been made. New error codes (809-822) have also been added for FIDO WebAuthN functionality. To view the error codes, see iOS SDK Error Codes.

IDSP iOS SDK Package Details

The latest iOS SDK package can be downloaded from the Support Portal > My Downloads

iOS SDK 2.3.5

View information about the iOS Symantec Identity Security Platform 2.3.5 release.

This topic includes the following sections:

- Release Information
- New Features and Improvements
- IDSP iOS SDK Kit Details

Release Information

- Release Title: IDSP iOS SDK
- Release Version: 2.3.5
- Preceding Version: 2.3.4
- Minimum OS Requirement: iOS v15.0 and above
- Release Date: July, 2024

New Features and Improvements

In this release, new error codes have been added for FIDO WebAuthN functionality and SDK initialization. To view the error codes, see iOS SDK Error Codes.

IDSP iOS SDK Package Details

The latest iOS SDK package can be downloaded from the Support Portal > My Downloads

iOS SDK - 2.3.4

View information about the iOS Symantec Identity Security Platform 2.3.4 release.

This topic includes the following sections:

- Release Information
- New Features and Improvements
- IDSP iOS SDK Kit Details

Release Information

- Release Title: IDSP iOS SDK
- Release Version: 2.3.4
- Preceding Version: 2.3.3
- Minimum OS Requirement: iOS v15.0 and above
- Release Date: May, 2024

New Features and Improvements

In this release, we have included the necessary privacy manifest file as mandated by Apple.com. To access the IDSP manifest file, click here.

IDSP iOS SDK Package Details

The latest iOS SDK package can be downloaded from the Support Portal > My Downloads

iOS SDK 2.3.3

View information about the iOS Symantec Identity Security Platform 2.3.3 release.

This topic includes the following sections:

- Release Information
- Changes
- IDSP iOS SDK Kit Details

Release Information

- Release Title: IDSP iOS SDK
- Release Version: 2.3.3
- Preceding Version: 2.3.2
- Minimum OS Requirement: iOS v15.0 and above
- Release Date: April 2, 2024

Changes

In IDSP version 3.1 and later, when data is encrypted in transit, the server expects the following encryption formats:

- PKI encrypted field with the {x5c} prefix (For example, - in-transit encrypted password data)
- Symmetric encrypted field with the {enc} prefix (For example, - encrypted device signature data)

These changes mean that IDSP iOS SDK v2.3.3 will not function with a IDSP server version lower than 3.1. Therefore, it is necessary to upgrade to IDSP version 3.1 before using IDSP iOS SDK v2.3.3.

IDSP Android SDK Package Details

The latest Android SDK package can be downloaded from the Support Portal > My Downloads .

iOS SDK 2.3.2

View information about the iOS Symantec Identity Security Platform 2.3.2 release.

This topic includes the following sections:

- Release Information
- New Features and Improvements
- Deprecations
- IDSP Android SDK Kit Details

Release Information

- Release Title: IDSP iOS SDK
- Release Version: 2.3.2
- Preceding Version: 2.3.1
- Minimum OS Requirement: iOS v15.0 and above
- Release Date: 17 February, 2024

New Features and Improvements

IDSP includes the following new features and improvements in iOS SDK 2.3.2 release:

- Introduced a new SDK API, "getFieldsToBeEncryptedInAuthFlow()", which returns the list of the fields that require encryption by the SDK during communication with the IDSP server. For more information, see Getting the list of fields to be encrypted in the authentication flow, and Support for proxy URL configuration to get the list of fields that are required for encryption .
- The Branding Settings related SDK API calls and Network request object initialization SDK API call has been refactored. See Initiate Authentication .
- Following this release, the network request object can be set to the AHCore instance object using the `setAuthHubRequestObject()` SDK API. See Setting Up the Network Request Object .

  NOTE
  If you previously used the `initAuth` SDK API to set the request object for logging network request data, you now must set the IDSP request object by invoking the setAuthHubRequestObject() SDK API on the AHCore instance.

Deprecations

- The following existing initAuth() SDK API which by default calls BrandingSettings SDK API has been deprecated.

```
public func initAuth(serverHost: String,
                     tenant: String,
                     requestHeaders:[String:Any],
                     authRequestParams: [String:Any],
                     authhubRequest: AuthHubBasicRequest?,
                     completionBlock: @escaping (( _ response: AHCoreResult) -> Void)) throws
```

  It has been replaced by the following SDK API:

```
public func initAuth(serverHost: String,
                     tenant: String,
                     requestHeaders:[String:Any],
                     authRequestParams: [String:Any],
                     fieldsTobeEncryptedList:[String]?,
                     completionBlock: @escaping (( _ response: AHCoreResult) -> Void)) throws
```

  NOTE
  It is recommended to use the new initAuth() SDK API that accepts the list of fields using the `fieldsTobeEncryptedList` function argument.

IDSP iOS SDK Package Details

The latest iOS SDK package can be downloaded from the Support Portal > My Downloads .

Android SDK - 3.0.3

View information about the Symantec Identity Security Platform Android 3.0.3 release.

This topic includes the following sections:

- Release Information
- Changes
- IDSP Android SDK Kit Details

Release Information

- Release Title: IDSP Android SDK
- Release Version: 3.0.3
- Preceding Version: 3.0.2
- Minimum OS Requirement: Android v24.0 and above
- Release Date: Dec, 2025

Changes

In this release, bug fixes and enhancements have been made.

- Updated Android SDK dependencies to latest version.
- Addressed few other bug fixes and enhancements.

IDSP Android SDK Package Details

The latest Android SDK package can be downloaded from the Support Portal > My Downloads.

Android SDK - 3.0.2

View information about the Symantec Identity Security Platform Android 3.0.2 release.

This topic includes the following sections:

- Release Information
- Changes
- IDSP Android SDK Kit Details

Release Information

- Release Title: IDSP Android SDK
- Release Version: 3.0.2
- Preceding Version: 3.0.1
- Minimum OS Requirement: Android v24.0 and above
- Release Date: Dec, 2025

Changes

In this release, bug fixes and enhancements have been made.

- Updated Android SDK dependencies to latest version.
- Addressed few other bug fixes and enhancements.

IDSP Android SDK Package Details

The latest Android SDK package can be downloaded from the Support Portal > My Downloads.

Android SDK 3.0.1

View information about the Symantec Identity Security Platform Android 3.0.1 release.

This topic includes the following sections:

- Release Information
- Changes
- IDSP Android SDK Kit Details

Release Information

- Release Title: IDSP Android SDK
- Release Version: 3.0.1
- Preceding Version: 3.0
- Minimum OS Requirement: Android v24.0 and above
- Release Date: July, 2025

Changes

In this release, bug fixes and enhancements have been made.

- Upgraded Android platform SDK version to 36

IDSP Android SDK Package Details

The latest Android SDK package can be downloaded from the Support Portal > My Downloads.

Android SDK 3.0

View information about the Symantec Identity Security Platform Android 3.0 release.

This topic includes the following sections:

- Release Information
- New Features and Improvements
- IDSP Android SDK Kit Details

Release Information

- Release Title: IDSP Android SDK
- Release Version: 3.0
- Preceding Version: 2.3.3
- Minimum Android SDK version: Android v24.0 and above
- Release Date: November, 2024

New Features and Improvements

In this release, we have introduced Passkey as an authentication factor, similar to the "Security Key" and "FIDO" (Biometric) factors. The "fidoCredType" parameter in the FIDO Registration (".../factor/v1/FIDORegChallengeGenerator") API differentiates the type of FIDO factor being used during registration. The "fidoCredType" property now supports the following values: `PASSKEY` , `FIDO(Biometric Authentication)` , and `SECURITYKEY` .

NOTE

- In this release, to register a new passkey, you must update or create an Authentication policy which contains the passkey as a factor.
- The "`useFQDN`" setting which was introduced in IDSP v2.2, is now used solely for backward compatibility. Therefore, it is only applicable during authentication (using the ".../factor/v1/FIDOAuthChallengeGenerator" and ".../factor/v1/FIDOAuthChallengeVerifier" APIs) for devices that have already registered passkey.

See Using FIDO Passkey.

IDSP Android SDK Package Details

The latest Android SDK package can be downloaded from the Support Portal > My Downloads .

Android SDK 2.3.3

View information about the Symantec Identity Security Platform Android 2.3.3 release.

This topic includes the following sections:

- Release Information
- Changes
- IDSP Android SDK Kit Details

Release Information

- Release Title: IDSP Android SDK
- Release Version: 2.3.3
- Preceding Version: 2.3.2
- Minimum OS Requirement: Android v15.0 and above
- Release Date: 6th September, 2024

Changes

IDSP Android SDK v2.3.3 addresses some potential vulnerabilities.

IDSP Android SDK Package Details

The latest Android SDK package can be downloaded from the Support Portal > My Downloads .

Android SDK 2.3.2

View information about the Symantec Identity Security Platform Android 2.3.2 release.

This topic includes the following sections:

- Release Information
- Changes
- IDSP iOS SDK Kit Details

Release Information

- Release Title: IDSP Android SDK
- Release Version: 2.3.2
- Preceding Version: 2.3.1
- Minimum OS Requirement: Android v15.0 and above
- Release Date: 2nd April, 2024

Changes

In IDSP version 3.1 and later, when data is encrypted in transit, the server expects the following encryption formats:

- PKI encrypted field with {x5c} prefix - (For example- in transit encrypted password data)
- Symmetric encrypted field with {enc} prefix - (For example- encrypted device signature data)

These changes mean that IDSP Android SDK v2.3.2 will not function with an IDSP server version lower than 3.1. Therefore, it is necessary to upgrade to IDSP version 3.1 before using IDSP Android SDK v2.3.2.

IDSP Android SDK Package Details

The latest Android SDK package can be downloaded from the Support Portal > My Downloads .

Android SDK 2.3.1

View information about the Symantec Identity Security Platform Android 2.3.1 release.

This topic includes the following sections:

- Release Information
- New Features and Improvements
- Deprecations
- IDSP iOS SDK Kit Details

Release Information

- Release Title: IDSP Android SDK
- Release Version: 2.3.1
- Preceding Version: 2.3.0
- Android SDK Version: Android SDK Version 24 and above
- Minimum OS Requirement: Android 7 and above
- Release Date: 2nd April, 2024

New Features and Improvements

IDSP includes the following new features and improvements in the Android SDK 2.3.1 release:

- Introduced a new SDK API, "getFieldsToBeEncryptedInAuthFlow()", which returns the list of the fields that require encryption by the SDK during communication with the IDSP server. For more information, see Getting the list of fields to be encrypted in the authentication flow, and Support for proxy URL configuration to get the list of fields that are required for encryption .
- The Branding Settings related SDK API calls have been refactored. See Initiate Authentication.

Deprecations

- The following existing initAuth() SDK API which by default calls BrandingSettings SDK API has been deprecated.

```
public void initAuth(String hostName,
                     String tenant,
                     JSONObject body,
                     Map<String, String> headers,
                     VolleyCallback callback) throws AuthError
```

It has been replaced by the following SDK API:

```
public void initAuth(String hostName,
                     String tenant,
                     JSONObject body,
                     Map<String, String> headers,
                     String[] fieldsToBeEncrypted,
                     VolleyCallback callback) throws AuthError
```

- Support for **proxy URL** configuration: The following existing initAuth() SDK API by default calls BrandingSettings SDK API has been deprecated.

```
public void initAuth(String serverUrl,
                     JSONObject body,
                     Map<String, String> headers,
                     VolleyCallback callback) throws AuthError
```

It has been replaced by the following SDK API:

```
public void initAuth(String serverUrl,
                     JSONObject body,
                     Map<String, String> headers,
                     String[] fieldsToBeEncrypted,
                     VolleyCallback callback) throws AuthError
```

NOTE
It is recommended to use the new initAuth() SDK API that accepts the list of fields using the `fieldsTobeEncryptedList` function argument.

IDSP Android SDK Package Details

The latest Android SDK package can be downloaded from the Support Portal > My Downloads .

Client-Server Compatibility Matrix

Explore the compatibility between Symantec Identity Security Platform Client SDKs (Android and iOS) and the IDSP Server component.

This topic includes the following sections:

- Compatibility Matrix: IDSP iOS Client SDK and Server Versions
- Compatibility Matrix: IDSP Android Client SDK and Server Versions
- Client SDK Capabilities
- Client SDK Packaged Links: Download Locations

iOS Client SDK and Server Version Compatibility Matrix

The following table indicates the versions of the **IDSP iOS clients** that are compatible with **IDSP Server versions**.

Check the Client Release Notes to know what is new in each release.

IDSP iOS Client SDK Version	IDSP Server Version	Server Compatibility
• SDK v3.1.5 • SDK v3.1.4 • SDK v3.1.3 • SDK v3.1.2 • SDK v3.1.1 • SDK v3.1 • SDK v3.0 • SDK v2.3.6 • SDK v2.3.5 • SDK v2.3.4	IDSP v4.0 release	**SDK Version Compat** Client applications mus IDSP server to version prevent compatibility is
• SDK v3.1.3 • SDK v3.1.2 • SDK v3.1.1 • SDK v3.1 • SDK v3.0 • SDK v2.3.6 • SDK v2.3.5 • SDK v2.3.4 • SDK v2.3.3	• VIP Authentication Hub v3.4.x release	
• SDK v3.1 • SDK v3.0 • SDK v2.3.6 • SDK v2.3.5 • SDK v2.3.4 • SDK v2.3.3	• VIP Authentication Hub v3.3.x release	
• SDK v3.1 • SDK v3.0 • SDK v2.3.6 • SDK v2.3.5 • SDK v2.3.4 • SDK v2.3.3	• VIP Authentication Hub v3.2.x release	
• SDK v3.1 • SDK v3.0 • SDK v2.3.6 • SDK v2.3.5 • SDK v2.3.4 • SDK v2.3.3 • SDK v2.3.2	• VIP Authentication Hub v3.1.x releases	In VIP Authentication H encryption formats for • PKI encrypted field data) • Symmetric encrypte signature data) These changes mean t 2.3.4 are incompatible versions. Therefore, it i 3.1 before upgrading to

Android Client SDK and Server Version Compatibility Matrix

The following table indicates the versions of the **IDSP Android clients** that are compatible with **IDSP Server versions**.

Check the Client Release Notes to know what is new in each release.

IDSP Android Client SDK Version	IDSP Server Version	SDK and Server Co
• SDK v3.0.3 • SDK v3.0.2 • SDK v3.0.1 • SDK v3.0 • SDK v2.3.3 • DK v2.3.2	• IDSP v4.0 release	
• SDK v3.0.1 • SDK v3.0 • SDK v2.3.3 • DK v2.3.2	• VIP Authentication Hub v3.4 release	
• SDK v3.0 • SDK v2.3.3 • SDK v2.3.2	• VIP Authentication Hub v3.3 release	
• SDK v3.0 • SDK v2.3.3 • SDK v2.3.2	• VIP Authentication Hub v3.2.x release	
• SDK v3.0 • SDK v2.3.3 • SDK v2.3.2 • SDK v2.3.1	• VIP Authentication Hub v3.1.x release	In IDSP version 3.1 data encrypted in tra • PKI encrypted fie data) • Symmetric encry signature data) These changes mea incompatible with VI Therefore, it is neces before upgrading to

IDSP Client SDK Capabilities

The following table outlines the capabilities of the IDSP Client SDKs (Android, iOS, Web).

Client SDK	Proxy URL	Password Encryption	Encrypt Data With Public Key	FIDO (Biometric and Security Key)	PASSKEY	FIDO T
iOS SDK	YES	YES	YES	YES	YES	YES
Android SDK	YES	YES	YES	YES	YES	YES
Web SDK (JS SDK)	**NO**	YES	YES	**NO**	**NO**	YES

Client SDK Packaged Links: Download Locations

The following table includes the location of the IDSP client (Android and iOS) SDK links from where these can be downloaded.

iOS Download Package Link	Android Download Package Link

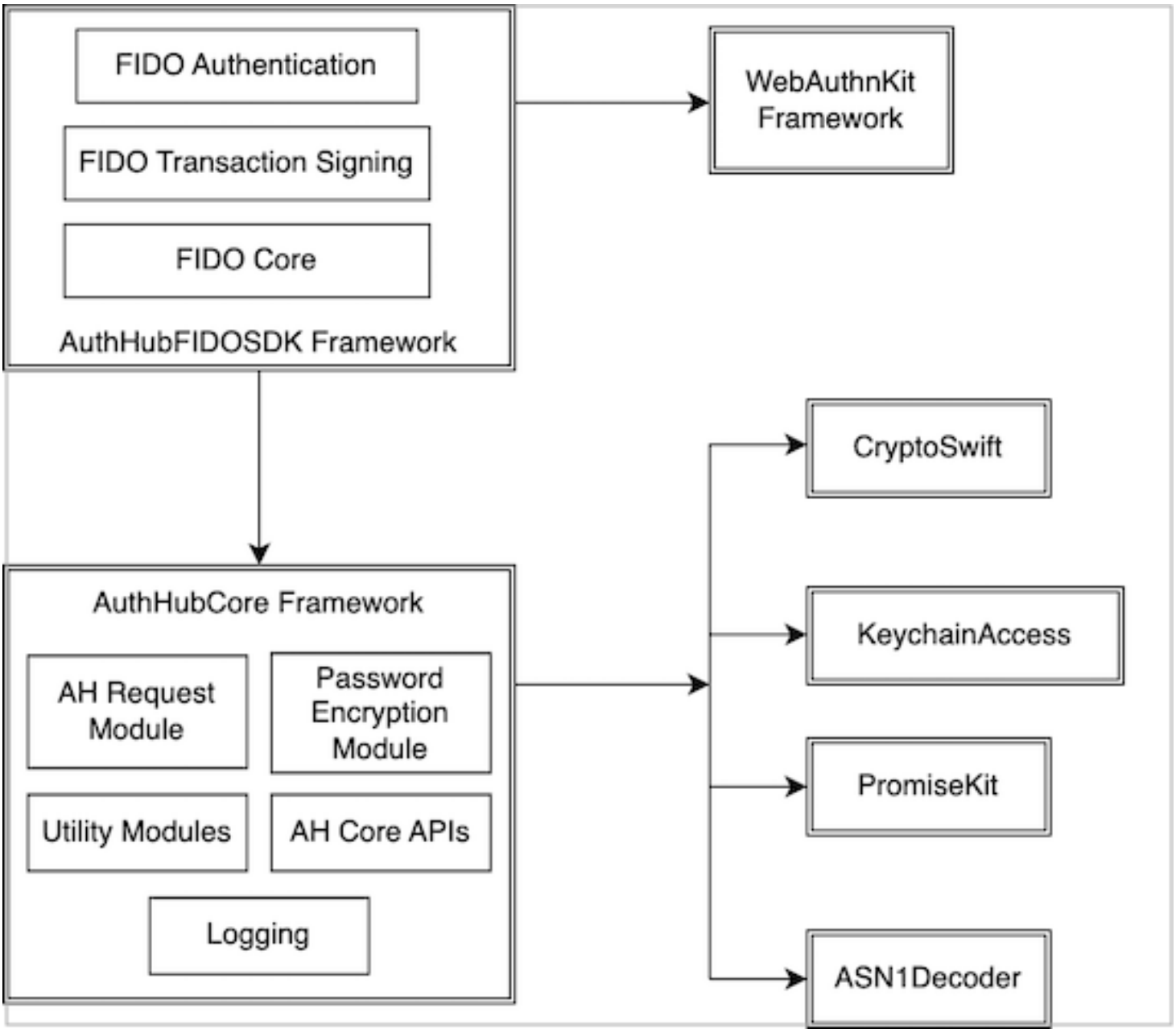
AuthHub-iOS-3.1.4.zip - Navigate to Support Portal > My Downloads (Search for VIP Authentication Hub in the portal and select the VIP Authentication Hub 3.4 version number).	AuthHub-Android SDK-3.0.3.zip - Navigate to Support for VIP Authentication Hub in the portal and select the V version number).
AuthHub-iOS-3.1.3.zip - Navigate to Support Portal > My Downloads (Search for VIP Authentication Hub in the portal and select the VIP Authentication Hub 3.4 version number).	AuthHub-Android SDK-3.0.2.zip - Navigate to Support for VIP Authentication Hub in the portal and select the V version number).
AuthHub-iOS-3.1.2.zip - Navigate to Support Portal > My Downloads (Search for VIP Authentication Hub in the portal and select the VIP Authentication Hub 3.4 version number).	AuthHub-Android SDK-3.0.1.zip - Navigate to Support for VIP Authentication Hub in the portal and select the V version number).
AuthHub-iOS-3.1.1.zip - Navigate to Support Portal > My Downloads (Search for VIP Authentication Hub in the portal and select the VIP Authentication Hub 3.4 version number).	AuthHub-Android SDK-3.0.zip - Navigate to Support P for VIP Authentication Hub in the portal and select the V version number).
AuthHub-iOS-3.1.zip - Navigate to Support Portal > My Downloads (Search for VIP Authentication Hub in the portal and select the VIP Authentication Hub 3.3/3.4 version number).	AuthHub-Android-SDK_v2.3.3.zip - Navigate to Suppo (Search for VIP Authentication Hub in the portal and se 3.2.1 version number).
AuthHub-iOS-3.0.zip - Navigate to Support Portal > My Downloads (Search for VIP Authentication Hub in the portal and select the VIP Authentication Hub 3.3 version number).	AuthHub-Android-SDK_v2.3.2.zip - Navigate to Suppor (Search for VIP Authentication Hub in the portal and se 3.1/3.2 version number).
AuthHub-iOS-2.3.6.zip - - Navigate to Support Portal > My Downloads (Search for VIP Authentication Hub in the portal and select the VIP Authentication Hub 3.2 version number).	AuthHub-Android-SDK_v2.3.1.zip - Navigate to Suppor (Search for VIP Authentication Hub in the portal and se 3.1 version number).
AuthHub-iOS-2.3.5.zip - Navigate to Support Portal > My Downloads (Search for VIP Authentication Hub in the portal and select the VIP Authentication Hub 3.2 version number).	
AuthHub-iOS-2.3.4.zip - Navigate to Support Portal > My Downloads (Search for VIP Authentication Hub in the portal and select the VIP Authentication Hub 3.1 version number).	
AuthHub-iOS-SDK-v2.3.3.zip - Navigate to Support Portal > My Downloads (Search for VIP Authentication Hub in the portal and select the VIP Authentication Hub 3.1 version number).	
AuthHub-iOS-SDK_v2.3.2.zip - Navigate to Support Portal > My Downloads (Search for VIP Authentication Hub in the portal and select the VIP Authentication Hub 3.1 version number).	

Symantec Identity Security Platform iOS SDK Framework

Integrating the Symantec Identity Security Platform iOS SDK with other customer iOS applications to use the IDSP Core, Password Encryption, and other functionalities.

The IDSP iOS client SDK framework has been explained with the help of the following diagram:

Figure 20: IDSP Client SDK Framework



The IDSP iOS client comprises two frameworks:

- AuthHubCore and AuthHubFIDOSDK

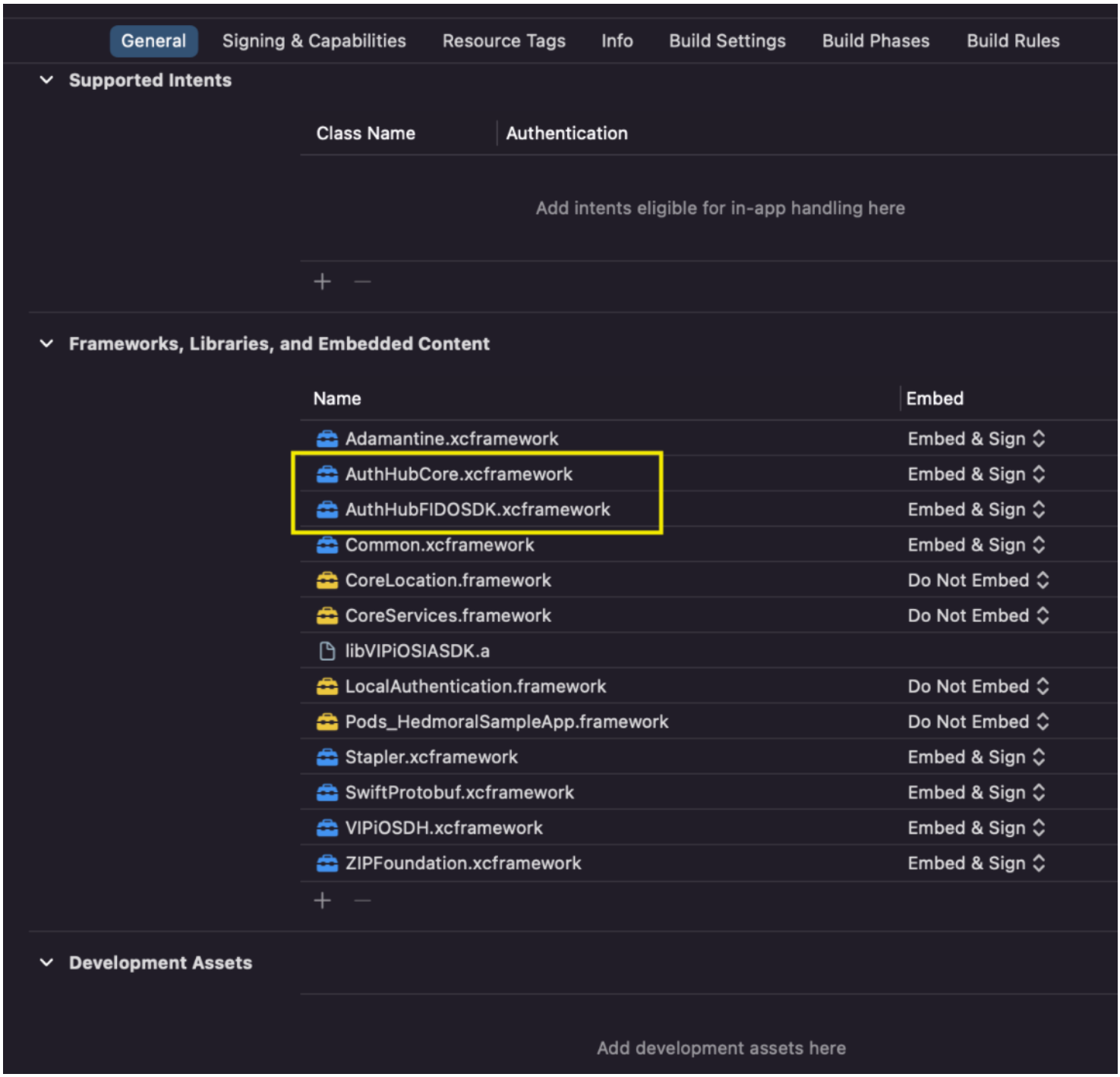
- AuthHubCore SDK includes AH Request Modules, Password Encryption Modules, Network Modules, Logging, and Utility APIs.
- AuthHubFIDOSDK supports APIs related to the FIDO authentication flow, and FIDO Transaction Signing.

Integration with iOS App

Learn about the Symantec Identity Security Platform iOS SDK, facilitating app developers in seamlessly integrating Password Encryption, FIDO (biometric support), and Transaction Signing functionalities into their iOS applications. Additionally, explore the IDSP Privacy Manifest file within this topic.

1. To integrate IDSP iOS SDKs with an iOS application, import and add the **AuthHubCore.xcframework** and **AuthHubFIDOSDK.xcframework** to the (XCode) application project.
   The following image shows the SDK frameworks that are imported into the XCode project:

   Figure 21: SDK Frameworks that are imported to the app project



2. The IDSP iOS SDK has third party pod dependencies, therefore uses the following Cocoa Pods which must be imported using the podfile:
   a. PromiseKit (https://cocoapods.org/pods/PromiseKit)
   b. KeychainAccess (https://cocoapods.org/pods/KeychainAccess)
   c. CryptoSwift (https://cocoapods.org/pods/CryptoSwift)
   d. ASN1Decoder (https://cocoapods.org/pods/ASN1Decoder)
3. Navigate to the XCode project root directory where the `.xcodeproj` file resides.

4. Type the following command to create a pod file in the root directory:
   ```
   $ pod init
   ```
5. Add the following dependencies to the pod file for the previously mentioned pods:
   ```
   # Uncomment the next line to define a global platform for your project
   platform :ios, '13.0'

   target 'iOS_APP_TARGET' do
        use_frameworks!
        inherit! :search_paths
        pod "PromiseKit", "~> 8.2.0"
        pod "KeychainAccess", "~> 4.2.2"
        pod "CryptoSwift", "~> 1.8.4"
        pod "ASN1Decoder", "~> 1.10.0"
   end
   ```
6. Run the following command to install these pods:
   ```
   $ pod install
   ```
   The following pods get installed.

Pod Name	Version
PromiseKit	8.2.0
KeychainAccess	4.2.2
CryptoSwift	1.8.4
ASN1Decoder	1.10.0

After the successful installation of the pods, open `.xcworkspace` and build it.

IDSP Manifest File

The following privacy manifest file has been added in the IDSP iOS SDK 2.3.4 as mandated by Apple.com. Refer to describing data use in privacy manifests link for more information on manifest file mandated by Apple.com.

```
<!-- Privacy manifest file for AuthHub iOS SDK -->
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>NSPrivacyTracking</key>
    <false/>
    <key>NSPrivacyTrackingDomains</key>
    <array/>
    <key>NSPrivacyCollectedDataTypes</key>
    <array>
        <dict>
            <key>NSPrivacyCollectedDataType</key>
            <string>NSPrivacyCollectedDataTypeUserID</string>
            <key>NSPrivacyCollectedDataTypeLinked</key>
            <false/>
            <key>NSPrivacyCollectedDataTypeTracking</key>
            <false/>
            <key>NSPrivacyCollectedDataTypePurposes</key>
            <array>
```

```
                <string>NSPrivacyCollectedDataTypePurposeAppFunctionality</string>
            </array>
        </dict>
    </array>
    <key>NSPrivacyAccessedAPITypes</key>
    <array/>
</dict>
</plist>
```

Integration with iOS App with dependent .xcframeworks (without running pod commands)

With Symantec Identity Security Platform iOS SDK v3.0, we are providing `.xcframework` files for the Pod dependencies of AuthHubCore.xcframework and AuthHubFIDOSDK.xcframework. If you are unable to run Pod commands in your development environment due to organizational restrictions, you can directly include these `.xcframework` dependencies in your application project. This way, there is no need to use the pod command to download, install, or build the dependencies.

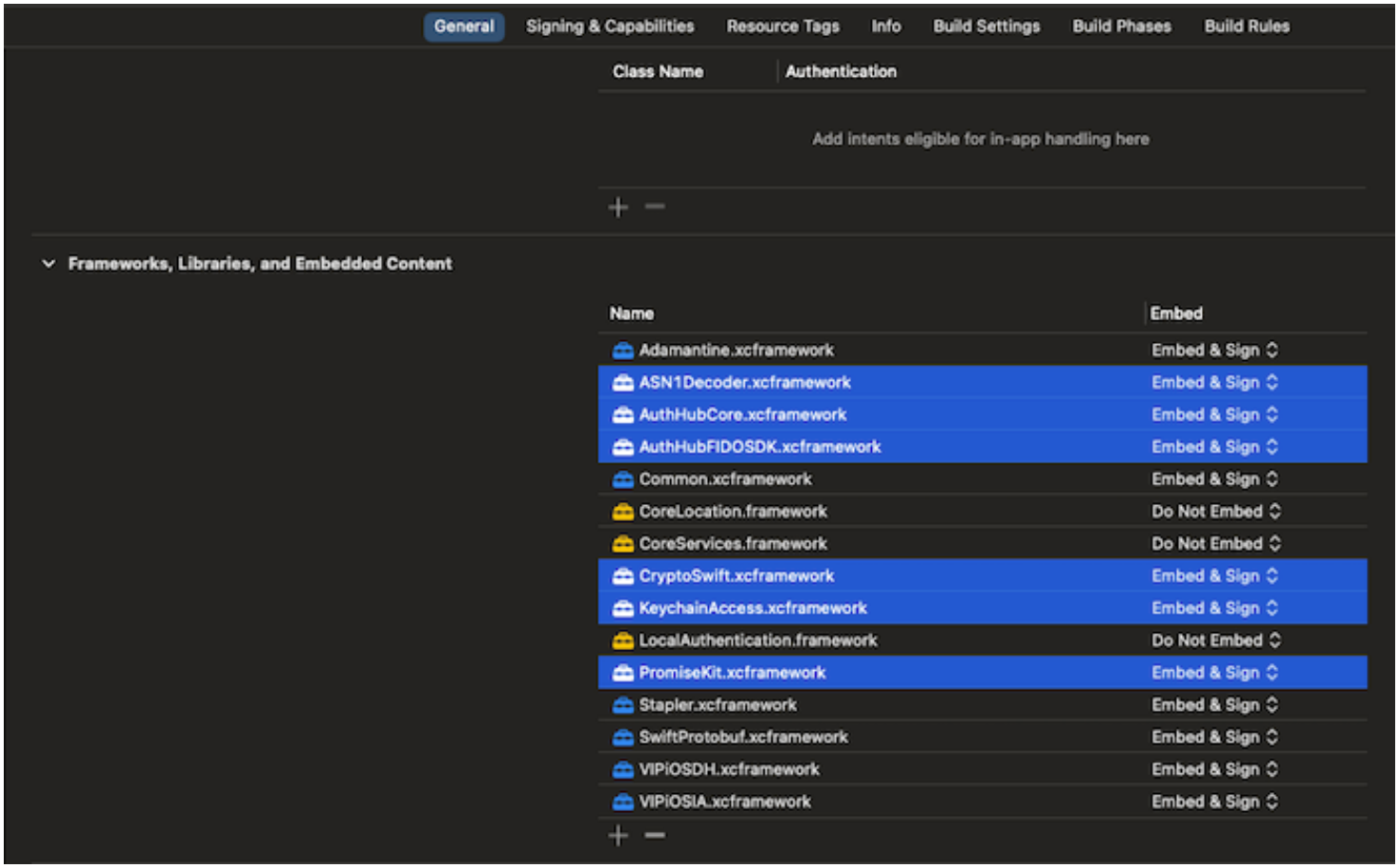
To integrate IDSP iOS SDKs with an iOS application, you must import **AuthHubCore.xcframework** and **AuthHubFIDOSDK.xcframework** to the application project along with dependent `.xcframework` libraries as the following:

- ASN1Decoder.xcframework
- CryptoSwift.xcframework
- KeychainAccess.xcframework
- PromiseKit.xcframework

These `.xcframework` libraries are available to download from the IDSPvalidation site as "**AH-Pods-Dependencies.zip**" file.

The following image shows the `.xcframeworks` that are imported to the app project:

Figure 22: SDK Frameworks that are imported to the app project



Symantec Identity Security Platform iOS Core SDK

Symantec Identity Security Platform Core SDK includes IDSP Request Module, Password Encryption Module, Network Module, and Utility APIs.

AuthHubCore.xcframework APIs and Integration Flow

This section includes the following sub-sections:

- Network Request Module
- IDSP Core APIs
  - Create an instance of the AuthHubCore.framework AHCore class
  - Setting Up the Network Request Object
  - Initiate Authentication
  - Data Encryption ("Password" and "Device Signature")
    - Get the list of fields to be encrypted in the authentication flow
      - Support for proxy URL configuration to get the list of fields that are required for encryption
    - Encryption Check
      - Support for proxy URL configuration for encryption check
    - Encrypt "Password" and "Device Signature"

IDSP Network Request Module

IDSP Network Request Module is a part of the AuthHubCore framework. The IDSP Client SDK APIs are designed to handle the network request and response for all the IDSP APIs, such as, "/authentication", "/BrandingSettings" and have complete control over them. Apart from handling essential API request payload/headers/query params, if you intend to customize the network layer to use it for tasks like adding custom headers or supporting SSL pinning, you need to implement three interfaces in your application to handle the network request customization. These interfaces are provided by IDSP and are explained in the following sections:

- ***AuthHubEndPoint***: This interface provides the set of APIs that mobile applications can implement to customize the network request. Using these API implementation, the mobile application can override the headers, add custom headers, or add new parameters to the request payload as required.
  This interface defines the following APIs -

```
func setRequestPayloadParameters(_ params: [String: Any]!)
func setRequestHeaders(_ headers: [String: Any]!)
func build() throws -> URLRequest
```

  The following table explains the APIs of this interface:

API Name	Description
setRequestPayloadParameters()	Accepts the map of key-value pairs which is sent in the JSON request body of the network request.
setRequestHeaders()	Accepts the map of key-value pairs which is sent as a request header of the network request.
build()	Builds an actual network request object which is sent to the IDSP server.

- ***AuthHubRequestLogger***: This interface provides the APIs required for logging the network level data and logs the time that is required for the network calls. The AuthHubRequestLogger provides the ability to log network request level information, such as request headers, request body, request response body, and response time.

  NOTE
  The AuthHubRequestLogger interface must be implemented if you want to add custom logging to the network request object.

  This interface defines the following APIs -

```
func enableLog(enable: Bool)
func setLoggingType(type: RequestLoggingType)
func logRequestData(responseTime: Int64, endPoint: AuthHubEndpoint, respDict: [String : AnyObject]?)
```

  The following table explains the APIs of this interface:

API Name	Description
enableLog()	Enables logging for a network request. By default it is ***disabled***.
setLoggingType()	Sets the logging type of the request logger. The RequestLoggingType is an ***enum*** that can have these values: ***resp_time***, ***data***, and ***none***. • resp_time - This type of logger logs only the response time • data - This type of logger logs the request data, such as, headers, request body, response body and response time • none - This type of logger does not log any data
logRequestData()	This API is used for logging the request data as per the RequestLoggingType.

  – ***AHRequestLogger*** :This class is the default implementation of the AuthHubRequestLogger interface. By default, this class logs all request data to the console.

NOTE
You must create an instance of this class and must initialize it with the name of the log file, if you want the logger to write logs to the file.

- ***AHAPIRequest***: This interface provides the APIs required to handle network level functionality, such as, sending requests, logging network request. This interface defines the following APIs -

```
func sendRequest(endpoint: AuthHubEndpoint) async throws -> AHRequestResult
func setRequestLogger(requestLogger: AuthHubRequestLogger)
```

The following table explains the APIs of this interface:

API Name	Description
sendRequest()	This API accepts the AuthHubEndpoint instance object.
setRequestLogger()	This API sets the AuthHubRequestLogger object for the network request object.

  - ***AuthHubBasicRequest***: This class is the default implementation of the ***AHAPIRequest*** interface.

The following example shows a sample implementation of the network request interface that is defined previously.

```
import Foundation
import AuthHubCore


class SampleAuthHubRequest : AuthHubBasicRequest {
  func addRequestHeaders(endpoint: AuthHubEndpoint) -> [String:Any] {
      let urlPath = endpoint.path
      var newHeaders = endpoint.headers
      switch(urlPath) {
      case .brandingSettings:
            newHeaders["HEDMORAL_BRANDING_SETTINGS"] = "HEDMORAL_BRANDING_VALUE";
            break;
      case .authHubPublicKey:
            newHeaders["HEDMORAL_PUBLIC_KEY"] = "HEDMORAL_PUBLIC_KEY_VALUE";
            break;
      case .serverNonceGenerator:
            newHeaders["HEDMORAL_SERVER_NONCE"] = "HEDMORAL_SERVER_NONCE_VALUE";
            break;
      case .authEndpoint:
            newHeaders["HEDMORAL_INIT_AUTH"] = "HEDMORAL_INIT_AUTH_VALUE";
            break;
      case .registerChallengeGenerator:
            newHeaders["HEDMORAL_REG_CHAL_GEN"] = "HEDMORAL_REG_CHAL_GEN_VALUE";
            break;
      case .registerChallengeVerifier:
            newHeaders["HEDMORAL_REG_CHAL_VERIFY"] = "HEDMORAL_REG_CHAL_VERIFY_VALUE";
            break;
      case .authChallengeGenerator:
            newHeaders["HEDMORAL_AUTH_CHAL_GEN"] = "HEDMORAL_AUTH_CHAL_GEN_VALUE";
            break;
      case .authChallengeVerifier:
            newHeaders["HEDMORAL_AUTH_CHAL_VERIFY"] = "HEDMORAL_AUTH_CHAL_VERIFY_VALUE";
            break;
      case .txnSignChallengeGenerator:
            newHeaders["HEDMORAL_TXN_CHAL_GEN"] = "HEDMORAL_TXN_CHAL_GEN_VALUE";
```

```
            break;
      case .txnSignChallengeVerifier:
            newHeaders["HEDMORAL_TXN_CHAL_VERIFY"] = "HEDMORAL_AUTH_CHAL_VERIFY_VALUE";
            break;
      @unknown default:
            newHeaders["HEDMORAL_DEFAULT_HEADER"] = "HEDMORAL_DEFAULT_HEADER_VALUE";
            break;
      }
      return newHeaders;
 }
  override func sendRequest(endpoint: AuthHubEndpoint) async throws -> AHRequestResult {
      let newHeaders = addRequestHeaders(endpoint: endpoint)
      endpoint.setRequestHeaders(newHeaders)
      return try await super.sendRequest(endpoint: endpoint)
 }
}
```

The following example shows how to use the sample request that was created in the previous example to initialize the IDSP FIDO SDK instance:

```
let ahfidosdk = AHFIDOSDK.getInstance()

/*Create instance of SampleAuthHubRequest*/
let hedmoralRequestObject = SampleAuthHubRequest()


/*Create instance of AHRequestLogger and enable it*/
let ahReqLogger = AuthHubCore.AHRequestLogger(fileName: nil)
ahReqLogger.enableLog(enable: true)

/*Set request logger for created request object*/
hedmoralRequestObject.setRequestLogger(requestLogger: ahReqLogger)


/*Create instance of AHRequestLogger and enable it*/
ahfidosdk.initFIDO(serverHost: "example.broadcom.com", tenant: "default", authhubRequest:
 hedmoralRequestObject)
```

IDSP Core Framework APIs

This section includes the IDSP Core Framework APIs:

Create an instance of the AuthHubCore.framework AHCore class

- To create an instance of the **AuthHubCore.framework AHCore** class, invoke the **getInstance()** API. The API returns a singleton instance of the AHCore class.

  ```
  public static func getInstance() -> AHCore
  ```

- To get the version of the **AuthHubCore.framework,** invoke the **getVersion() API.** The currently supported version of the **AuthHubCore.framework** is 2.3.2.

  ```
  public static func getVersion() -> String
  ```

Setting Up the Network Request Object

To set the network request object, call the AuthHubCore API, ***setAuthHubRequestObject()***

```
public func setAuthHubRequestObject(authhubRequest: AuthHubBasicRequest)
```

The API parameters have been described in the following table:

Name	Type	Description
authhubRequest	AuthHubBasicRequest	This is an optional parameter and can be set to nil value. As a developer, if you intend to customise the network request wh request body param), then implement AHAPIRequest interface. T of the AHAPIRequest interface. When the initAuth() is invoked wit implementation of AHAPIRequest interface that is; AuthHubBasicl

Initiate Authentication

- Use the **initAuth** function to initiate the authentication flow by invoking the POST `https://${SSP_FQDN}/<tenantName>/auth/v1/authenticate` API. Use this function at all existing places where you are invoking the */* **authenticate** API.

  NOTE

  - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
  - The default value for `<tenantName>` is "default".
  - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.

    For example:

    `https://{{host}}/{{tenantName}}/auth/v1/authenticate`

    should look similar to:

    `https://example.com/default/auth/v1/authenticate`
  - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

API Signature

```
public func initAuth(serverHost: String,
                     tenant: String,
                     requestHeaders:[String:Any],
                     authRequestParams: [String:Any],
                     fieldsTobeEncryptedList:[String]?,
                     completionBlock: @escaping (( _ response: AHCoreResult) -> Void)) throws
```

The API parameters are described in the following table:

Name	Type	Description
serverHost	string	This is the IDSP server host.
tenant	string	This is the tenant name.
requestHeaders	dictionary	The requestHeaders is the collection of IDSP FIDO Server specific headers that are required for REST API calls. The following parameters should be passed with proper values: • Authorization

authRequestParams	dictionary	This is a collection of parameters that are required for '/authenticate' REST API calls. The following sample parameters can be passed as a dictionary object:
		`{`
		`"subject": "{{userLoginid}}",`
		`"channel": "mobile",`
		`"ipAddress": "10.10.123.124",`
		`"action":"authenticate",`
		`"rememberMe" : true,`
		`"device": {`
		`"signature": {`
		`"iaAuthData":`
		`"_v02MyUgNGgYOi88OTk0emB7ZXV9GDQ2PDshOiY9bnUcOyEwOXUYNDZ1GgZ1DXVkZQpk`
		`NzclNDA6PzozOj06MDMyPDA9PzQ8KTw7ITAnOzQ5eDs0Njl4JTkgMjw7czMlJTxoczMlIS`
		`}`
		`}`
		`}`

fieldsTobeEncryptedList	string array	• This is optional parameter and can be set to nil value. • The list of string array contains the list of the fields which needs encryption. This list of fields can be obtained from getFieldsToBeEncryptedInAuthFlow() API. The list of fields that are sent to initAuth() decides whether server nonce is required to encrypt field in the transit.
completionBlock	closure block	Represents the closure block that the SDK passes to the calling API. The closure block returns the AHCoreResult object as a response. AHCoreResult is a struct object containing data (dictionary object), code (API response), and message (string) as members.

The completionBlock closure returns a AHCoreResult struct object as defined here:

```
public struct AHCoreResult {
 public init(code: Int, message: String, data: [String:Any]) {
     self.code = code
     self.message = message
     self.data = data
 }
 public var code : Int


 public var message : String


 public var data : [String:Any]

}
```

- ***Example: API call with Server Host and Tenant***

```
do
{
 //set network request object in AuthCore
 ahCore.setAuthHubRequestObject(authhubRequest:hedmoralRequestObject)


 //get access token from server
```

```
 var jsonResponse = getAccessTokenFromAHServer()
 var accessToken = "Bearer \(jsonResponse["access_token"] as! String)"
 var reqHeaders = [String:Any]()
 reqHeaders["Authorization"] = accessToken


 /*get list of the fields from BrandingSettings API*/
 let brandingSettingsHeaders:[String:String] = [AHConstants.HttpHeader.KeyAz:accessToken]
 try ahCore.getFieldsToBeEncryptedInAuthFlow(serverHost: serverHost,
                                             tenant: tenant,
                                             requestHeaders:brandingSettingsHeaders)
{ result in
     if(result.code == AHResult.Code.SUCCESS) {
         //On success, fetch the list of the fields which needs encryption
         let fieldsToBeEncrypted = result.data[AHConstants.GenericKeys.FieldsToBeEncrypted] as? [String] ??
 []
         //pass on the list to initAuth()
         try ahCore.initAuth(serverHost: serverHost,
                             tenant: tenant,
                             requestHeaders: reqHeaders,
                             authRequestParams: paramBodyRequest,
                             fieldsTobeEncryptedList:fieldsToBeEncrypted) { response in
             if response.data["errorCode"] != nil {
                 let errorCode = response.data["errorCode"] as! String
                 let code = Int(errorCode)!
                 var errorMsg = response.data["errorMessage"] as? String ?? ""
                 if(!errorMsg.isEmpty) {
                     print("Error: message - \(errorMsg) and code: \(errorCode)")
                 }
             }
             else {
                 if let flowState = response.data["flowState"] as? String {
                    print(flowState)
             }
             guard let nextAction = response.data["nextaction"] else {
                   Logger().logMessage(message: "Next Action not set!!")
                   return
             }
             //handle nevigation to next page based on response from AH server
             navigateToNextPageBasedOnNextAction()
         }
      }
 }
}
```

This function is used for initiating the authentication flow by invoking the POST `https://<proxyUrl>/auth/v1/authenticate` API. This API also supports the **proxy URL** configuration for the IDSP instance where the tenant name is not required as part of the URL. This function should be used at all the existing places where you are invoking the **/authenticate** API.

API Signature

```
public func initAuth(serverUrl: String,
                 requestHeaders:[String:Any],
                 authRequestParams: [String:Any],
                 fieldsTobeEncryptedList:[String]?,
```

```
completionBlock: @escaping (( _ response: AHCoreResult) -> Void)) throws
```

The API parameters have been described here:

Name	Type	Description
serverURL	string	The endpoint url that can be a proxy url for an IDSP instance.
requestHeaders	dictionary	The requestHeaders is the collection of IDSP FIDO Server specific headers that are required for REST API calls. The following parameters should be passed with proper values: • Authorization
authRequestParams	dictionary	This is a collection of parameters that are required for '/authenticate' REST API calls. The following sample parameters can be passed as a dictionary object: `{ "subject": "{{userLoginid}}", "channel": "mobile", "ipAddress": "10.10.123.124", "action":"authenticate", "rememberMe" : true, "device": { "signature": { "iaAuthData": "_v02MyUgNGgYOi88OTk0emB7ZXV9GDQ2PDshOiY9bnUcOyEwOXUYNDZ1GgZ1DXVkZQpk NzclNDA6PzozOj06MDMyPDA9PzQ8KTw7ITAnOzQ5eDs0Nj14JTkgMjw7czMlJTxoczMlIS } } }`
fieldsTobeEncryptedList	string array	• This is an optional parameter and can be set to nil. • The list of string array contains the list of the fields which needs encryption. This list of fields can be obtained from getFieldsToBeEncryptedInAuthFlow() API. The list of fields sent to initAuth() decides whether server nonce is required to encrypt the fields in the transit.
completionBlock	closure block	Represents the closure block that the SDK passes to the calling API. The closure block returns the AHCoreResult object as a response. AHCoreResult is a struct object containing data (dictionary object), code (API response), and message (string) as members.

The completionBlock closure returns a AHCoreResult struct object as defined here:

```
public struct AHCoreResult {
 public init(code: Int, message: String, data: [String:Any]) {
     self.code = code
     self.message = message
     self.data = data
 }
 public

var code : Int
 public
var message : String
 public
var data : [String:Any]
```

```
}
```

- ***Example: API call with Proxy URL***

```
do
{
 //set network request object in AuthCore
 ahCore.setAuthHubRequestObject(authhubRequest:hedmoralRequestObject)


 //get access token from server
 var jsonResponse = getAccessTokenFromAHServer()
 var accessToken = "Bearer \(jsonResponse["access_token"] as! String)"
 var reqHeaders = [String:Any]()
 reqHeaders["Authorization"] = accessToken


 /*get list of the fields from BrandingSettings API*/
 let brandingSettingsHeaders:[String:String] = [AHConstants.HttpHeader.KeyAz:accessToken]
 try ahCore.getFieldsToBeEncryptedInAuthFlow(serverUrl: proxyUrl,
                                             requestHeaders:brandingSettingsHeaders)
{ result in
     if(result.code == AHResult.Code.SUCCESS) {
         //On success, fetch the list of the fields which needs encryption
         let fieldsToBeEncrypted = result.data[AHConstants.GenericKeys.FieldsToBeEncrypted] as? [String] ??
 []
         //pass on the list to initAuth()
         try ahCore.initAuth(serverUrl: proxyUrl,
                             requestHeaders: reqHeaders,
                             authRequestParams: paramBodyRequest,
                             fieldsTobeEncryptedList:fieldsToBeEncrypted) { response in
             if response.data["errorCode"] != nil {
                 let errorCode = response.data["errorCode"] as! String
                 let code = Int(errorCode)!
                 var errorMsg = response.data["errorMessage"] as? String ?? ""
                 if(!errorMsg.isEmpty) {
                     print("Error: message - \(errorMsg) and code: \(errorCode)")
                 }
             }
             else {
                 if let flowState = response.data["flowState"] as? String {
                    print(flowState)
             }
             guard let nextAction = response.data["nextaction"] else {
                   Logger().logMessage(message: "Next Action not set!!")
                   return
             }
             //handle nevigation to next page based on response from AH server
             navigateToNextPageBasedOnNextAction()
         }
      }
 }
}
```

Data Encryption

IDSP supports "Password" (In-Transit and At-Rest) encryption and "Device Signature" (In-Transit) encryption.

Prerequisite

To use the iOS client SDK for ***"Password"*** and ***"Device Signature"*** encryption, the following prerequisites must be met.

1. Change the value of the tenant setting, **fieldsToBeEncryptedInAuthFlow** to **"Password", and/or "Device Signature".** The default value is **none,** which indicates that "Password" and "Device Signature" encryption is disabled by default.

```
[
 {
     "name": "fieldsToBeEncryptedInAuthFlow",
     "value": "password, devicesignature"
 }
]
```

2. Integrate the **AuthHubCore.framework** SDK with the iOS application. The SDK includes the APIs that can be called from the iOS application context to achieve "Password" and "Device Signature" encryption.

   NOTE
   The **AuthHubCore.framework** SDK file can be download from the Support Portal > My Downloads along with other SDKs.

Get the list of fields to be encrypted in the authentication flow

1. To get a list of the fields that are enabled for encryption, invoke **getFieldsToBeEncryptedInAuthFlow()** function before calling initAuth().

```
public func getFieldsToBeEncryptedInAuthFlow(serverHost: String,
                                             tenant: String,
                                             requestHeaders:[String:Any],
                                             completionBlock: @escaping ( _ response: AHCoreResult) -> Void)
 throws
```

   The API parameters have been described here:

Name	Type	Description
serverHost	string	This is the IDSP server host.
tenant	string	This is the tenant name.
requestHeaders	dictionary	• The requestHeaders is the collection of IDSP FIDO Server specific headers that are required for for "/auth/v1/BrandingSettings" IDSP REST API calls. • In this case, you must pass authorization request header value as reqHeaders["Authorization"] = accessToken as mandatory request header parameter. • Other optional custom headers can be passed as per customer requirement.
completionBlock	closure block	• Represents the closure block that the SDK passes to the calling API. The closure block returns the AHCoreResult object as a response. AHCoreResult is a struct object containing data (dictionary object), code (API response), and message (string) as members. • The data is a dictionary object, message is a string, and the code is the API response code. • The result.data[AHConstants.GenericKeys.FieldsToBeEncrypted] contains list of fields(string array) [String] which can be passed on to the initAuth() API

The completionBlock closure returns an AHCoreResult object, containing a dictionary collection named "data." The result.data dictionary, comprises a key identified as AHConstants.GenericKeys.FieldsToBeEncrypted (specified in the

fieldName argument of the API) and a list of fields in string array format. [String], denoting the fields requiring encryption. Currently, it may include "password," "devicesignature," or both.

```
do
{
 //set network request object in AuthCore
 ahCore.setAuthHubRequestObject(authhubRequest:hedmoralRequestObject)


 //get access token from server
 var jsonResponse = getAccessTokenFromAHServer()
 var accessToken = "Bearer \(jsonResponse["access_token"] as! String)"
 var reqHeaders = [String:Any]()
 reqHeaders["Authorization"] = accessToken


 /*get list of the fields from BrandingSettings API*/
 let brandingSettingsHeaders:[String:String] = [AHConstants.HttpHeader.KeyAz:accessToken]
 try ahCore.getFieldsToBeEncryptedInAuthFlow(serverHost: serverHost,
                                             tenant: tenant,
                                             requestHeaders:brandingSettingsHeaders)
{ result in
     if(result.code == AHResult.Code.SUCCESS) {
         //On success, fetch the list of the fields which needs encryption
         let fieldsToBeEncrypted = result.data[AHConstants.GenericKeys.FieldsToBeEncrypted] as? [String] ?? []
         //pass on the list to initAuth()
         try ahCore.initAuth(serverHost: serverHost,
                             tenant: tenant,
                             requestHeaders: reqHeaders,
                             authRequestParams: paramBodyRequest,
                             fieldsTobeEncryptedList:fieldsToBeEncrypted) { response in
             if response.data["errorCode"] != nil {
                 let errorCode = response.data["errorCode"] as! String
                 let code = Int(errorCode)!
                 var errorMsg = response.data["errorMessage"] as? String ?? ""
                 if(!errorMsg.isEmpty) {
                     print("Error: message - \(errorMsg) and code: \(errorCode)")
                 }
             }
             else {
                 if let flowState = response.data["flowState"] as? String {
                    print(flowState)
             }
             guard let nextAction = response.data["nextaction"] else {
                   Logger().logMessage(message: "Next Action not set!!")
                   return
             }
             //handle nevigation to next page based on response from AH server
             navigateToNextPageBasedOnNextAction()
         }
       }
 }
}
```

Support for proxy URL configuration to get the list of fields that are required for encryption

The following API supports the proxy Url configuration for IDSP instance where tenant name is not required as part of the URL.

```
public func getFieldsToBeEncryptedInAuthFlow(serverUrl: String,
                                             requestHeaders:[String:Any],
                                             completionBlock: @escaping ( _ response: AHCoreResult) -> Void)
 throws
```

The API parameters have been described here:

Name	Type	Description
serverURL	string	The endpoint url that can be a proxy url for an IDSP instance.
requestHeaders	dictionary	• requestHeaders is the collection of AuthHub FIDO Server specific parameters required for "/auth/v1/BrandingSettings" AH REST call. • In this case, developer must pass authorization request header value as reqHeaders["Authorization"] = accessToken as mandatory request header parameter. • Other optional custom headers can be passed as per customer requirement.
completionBlock	closure block	• Represents the closure block that the SDK passes to the calling API. The closure block returns the AHCoreResult object as a response. • AHCoreResult is a struct object containing data (dictionary object), code (API response), and message (string) as members. • The data is a dictionary object, message is a string, and the code is the API response code.

The completionBlock closure returns an AHCoreResult object, containing a dictionary collection named "data." The result.data dictionary, comprises a key identified as AHConstants.GenericKeys.FieldsToBeEncrypted (specified in the fieldName argument of the API) and a list of fields in string array format. [String], denoting the fields requiring encryption. Currently, it may include "password," "devicesignature," or both, as shown in the following example.

```
do
{
 //set network request object in AuthCore
 ahCore.setAuthHubRequestObject(authhubRequest:hedmoralRequestObject)


 //get access token from server
 var jsonResponse = getAccessTokenFromAHServer()
 var accessToken = "Bearer \(jsonResponse["access_token"] as! String)"
 var reqHeaders = [String:Any]()
 reqHeaders["Authorization"] = accessToken


 /*get list of the fields from BrandingSettings API*/
 let brandingSettingsHeaders:[String:String] = [AHConstants.HttpHeader.KeyAz:accessToken]
 try ahCore.getFieldsToBeEncryptedInAuthFlow(serverUrl: proxyUrl,
                                             requestHeaders:brandingSettingsHeaders)
{ result in
     if(result.code == AHResult.Code.SUCCESS) {
         //On success, fetch the list of the fields which needs encryption
         let fieldsToBeEncrypted = result.data[AHConstants.GenericKeys.FieldsToBeEncrypted] as? [String] ?? []
         //pass on the list to initAuth()
```

```
        try ahCore.initAuth(serverUrl: proxyUrl,
                            requestHeaders: reqHeaders,
                            authRequestParams: paramBodyRequest,
                            fieldsTobeEncryptedList:fieldsToBeEncrypted) { response in
            if response.data["errorCode"] != nil {
                let errorCode = response.data["errorCode"] as! String
                let code = Int(errorCode)!
                var errorMsg = response.data["errorMessage"] as? String ?? ""
                if(!errorMsg.isEmpty) {
                    print("Error: message - \(errorMsg) and code: \(errorCode)")
                }
            }
            else {
                if let flowState = response.data["flowState"] as? String {
                   print(flowState)
            }
            guard let nextAction = response.data["nextaction"] else {
                  Logger().logMessage(message: "Next Action not set!!")
                  return
            }
            //handle nevigation to next page based on response from AH server
            navigateToNextPageBasedOnNextAction()
        }
    }
  }
 }
```

Encryption Confirmation

Encryption confirmation for "Password" and "Device Signature" is described in this section.

- ***"Password" Encryption Confirmation***
  Call the **isEncryptionRequiredForField** function to check if the password field is configured for "Password" encryption.

  API Signature

```
 public func isEncryptionRequiredForField(serverHost: String,
                                          tenant: String,
                                          requestHeaders:[String:Any],
                                          fieldName:String,
                                          completionBlock: @escaping (( _ response: AHCoreResult) -> Void))
  throws -> Bool
```

  Sample Code for "Password" Encryption

```
 //get access token from server
 var jsonResponse = getAccessTokenFromAHServer()
 var accessToken = "Bearer \(jsonResponse["access_token"] as! String)"
 var reqHeaders = [String:Any]()
 reqHeaders["Authorization"] = accessToken

 let ahCore = AHCore.getInstance()do {

  //get user password
  try ahCore.isEncryptionRequiredForField(serverHost: serverHost,
                                          tenant: tenant,
```

```
                                        requestHeaders:reqHeaders,
                                        fieldName: "password", completionBlock: {
        result in
        if result.data["errorCode"] != nil {
            let errorCode = result.data["errorCode"] as! String
            let code = Int(errorCode)!
            var errorMsg = result.data["errorMessage"] as? String ?? ""
            if(!errorMsg.isEmpty) {
                print("Error: message - \(errorMsg) and code: \(errorCode)")
            }
            handleError(errorCode, errorMsg)
        }
        do {
            let code = result.code
            let isPswdEncRequired = result.data[AHConstants.GenericKeys.PasswordField] as? Bool ?? false
            if(code == AHResult.Code.SUCCESS && isPswdEncRequired) {
                let encryptedPassword = try ahCore.encryptdData(plainText: password)
                if(!encryptedPassword.isEmpty) {
                    password = encryptedPassword
                }
            }
            //send password authentication request to the IDSP server
            performPasswordAuthentication(password)

        } catch {
            print(error)
            return
        }
    })
   }
   catch {
    let message = error.localizedDescription
    print("Error: message - \(message)")
   }
```

- ***"Device Signature" Encryption Confirmation***

  Call the **isEncryptionRequiredForField** function to check if the device signature field is configured for encryption.

  API Signature

```
public func isEncryptionRequiredForField(serverHost: String,
                                         tenant: String,
                                         requestHeaders:[String:Any],
                                         fieldName:String,
                                         completionBlock: @escaping (( _ response: AHCoreResult) -> Void))
 throws -> Bool
```

  Sample Code for "Device Signature" Encryption

```
//get access token from server
var jsonResponse = getAccessTokenFromAHServer()
var accessToken = "Bearer \(jsonResponse["access_token"] as! String)"
var reqHeaders = [String:Any]()
reqHeaders["Authorization"] = accessToken

let ahCore = AHCore.getInstance()do {
```

```
    //get device signature from VIPIA framework
    try ahCore.isEncryptionRequiredForField(serverHost: serverHost,
                                            tenant: tenant,
                                            requestHeaders:reqHeaders,
                                            fieldName: "devicesignature", completionBlock: {
        result in
        if result.data["errorCode"] != nil {
            let errorCode = result.data["errorCode"] as! String
            let code = Int(errorCode)!
            var errorMsg = result.data["errorMessage"] as? String ?? ""
            if(!errorMsg.isEmpty) {
                print("Error: message - \(errorMsg) and code: \(errorCode)")
            }
            handleError(errorCode, errorMsg)
        }
        do {
            let code = result.code
            let isEncRequired = result.data["devicesignature"] as? Bool ?? false
            if(code == AHResult.Code.SUCCESS && isEncRequired) {
                let encryptedSignature = try ahCore.encryptdData(plainText: deviceSignature)
                if(!encryptedSignature.isEmpty) {
                    deviceSignature = encryptedSignature
                }
            }
            //send encrypted device signature as "x-device-tag"  header to the IDSP server
            setEcryptedDeviceTagHeader(deviceSignature)

        } catch {
            print(error)
            return
        }
     })
    }
    catch {
     let message = error.localizedDescription
     print("Error: message - \(message)")
    }
```

Support for proxy URL configuration

The Encryption API also supports the **proxy URL** configuration for the AuthHub instance where the tenant name is not required as part of the URL.

The following is an example for "Password" encryption. You can use the same API signature for "Device Signature".

API Signature

```
 public func isEncryptionRequiredForField(serverUrl: String,
                                          requestHeaders:[String:Any]
                                          fieldName:String,
                                          completionBlock: @escaping (( _ response: AHCoreResult) -> Void))
  throws -> Bool
```

Sample Code: This example shows a "Password" encryption sample using a proxyURL. Similar code can be used for "Device Signature".

```
//get access token from server
var jsonResponse = getAccessTokenFromAHServer()
var accessToken = "Bearer \(jsonResponse["access_token"] as! String)"
var reqHeaders = [String:Any]()
reqHeaders["Authorization"] = accessToken


let ahCore = AHCore.getInstance()
do {

 //get user password
 try ahCore.isEncryptionRequiredForField(serverUrl: proxyUrl,
                                         requestHeaders:reqHeaders,
                                         fieldName: "password", completionBlock: {
     result in
     if result.data["errorCode"] != nil {
         let errorCode = result.data["errorCode"] as! String
         let code = Int(errorCode)!
         var errorMsg = result.data["errorMessage"] as? String ?? ""
         if(!errorMsg.isEmpty) {
             print("Error: message - \(errorMsg) and code: \(errorCode)")
         }
         handleError(errorCode, errorMsg)
     }
     do {
         let code = result.code
         let isPswdEncRequired = result.data[AHConstants.GenericKeys.PasswordField] as? Bool ?? false
         if(code == AHResult.Code.SUCCESS && isPswdEncRequired) {
             let encryptedPassword = try ahCore.encryptdData(plainText: password)
             if(!encryptedPassword.isEmpty) {
                 password = encryptedPassword
             }
         }
         //send password authentication request to AH server
         performPasswordAuthentication(password)

     } catch {
         print(error)
         return
     }
 })
}
catch {
 let message = error.localizedDescription
 print("Error: message - \(message)")
}
```

Parameter Description

- **hostName:** Specifies the SSP (IDSP) server hostname. Data Type - **string.**
- **tenant:** Specifies the tenant name. Example: default. Data Type - **string.**
- **serverUrl:** Specifies the endpoint URL that can be a proxy URL for an IDSP instance. Data Type - **string.** Ensure that the proxy resolves the IDSP instance URL that points to the intended IDSP server host and tenant.
- **requestHeaders:** Represents a collection of IDSP server-specific headers that are required for the REST API calls. The mandatory parameter to be passed is the Authorization header containing **"Bearer <auth_token>".** Data Type - **dictionary.**
- **fieldName:** Specifies the field name that is marked for password encryption. The value of this parameter must be **password.**
- **completionBlock:** Represents the closure block that the SDK passes to the calling API. The closure block returns the AHCoreResult object as a response. AHCoreResult is a struct object containing data (dictionary object), code (API response), and message (string) as members. The result.data dictionary contains the key as the field name (password) and boolean data (true, false) as a value. The value **true** indicates that the password field must be encrypted.

Encrypt "Password" and "Device Signature"

- ***Encrypt Password***
  To encrypt the plain password with the tenant-level public key `without` server nonce, call the **encryptDataWithPublicKey** function:
  API Signature
  ```
  public func encryptDataWithPublicKey(plainText: String) throws -> String
  ```
  To encrypt the plain password with the tenant-level public key `along with` server nonce, call the **encryptData** function:
  API Signature
  ```
  public func encryptData(plainText: String) throws -> String
  ```
  Parameter Description
  - **plainText:** Specifies the password string that must be encrypted in the authentication flow.

  API Response
  The API encrypts the plain text and returns the encrypted value as a base64 encoded string. The encrypted password can be sent to the IDSP where password encryption is enabled for authentication. For server-side information, see Password Encryption.
- ***Encrypt "Device Signature"***
  To encrypt the plain "Device Signature" with symmetric key, call the **encryptDataWithSymmetricKey** function:
  API Signature
  ```
  public func encryptDataWithSymmetricKey(plainText: String) throws -> String
  ```
  Parameter Description
  – **plainText:** Specifies the "Device Signature", "X-Device-Tag" string that must be encrypted in the authentication flow.
  API Response
  The API encrypts the plain text and returns the encrypted value in a base64 encoded string. The encrypted device signature and x-device-tag can be sent to the IDSP as an encrypted data. For server-side information, see "Device Signature" Encryption.

FIDO iOS SDK

The FIDO iOS Software Development Kit (SDK) provides the APIs and other configuration details that are required to build your own mobile device application on iOS for FIDO/SECURITYKEY/PASSKEY registration, FIDO/SECURITYKEY/PASSKEY authentication, and transaction signing.

This topic includes the following sections:

- Creating an instance For FIDO class
- Initializing SDK
- FIDO Device Registration
- FIDO Device Authentication
- Enabling Passkey
  - Hosting the "Apple-App-Site-Association" File
  - Passkey Device Registration
  - Passkey Device Authentication
- Transaction Signing
- Localizing the FIDO Registration and Authentication Biometric Messages

The FIDO APIs perform the following functions:

- Register FIDO and Passkey onto the device storage
- Authenticate FIDO and Passkey credentials
- Callback API after successful registration and authentication of FIDO
- Sign transactions in the Relying Party-initiated transaction flow

Creating an instance for FIDO class

To create an instance of the **AuthHubFIDOSDK.framework AHFIDOSDK** class, invoke the **getInstance()** API. The API returns a singleton instance of the AHFIDOSDK class.

```
public static func getInstance() -> AHFIDOSDK
```

To know the version of the **AuthHubFIDOSDK.framework,** invoke the **getVersion() API.** The currently supported version of the **AuthHubFIDOSDK.framework** is 2.1.1.

```
public static func getVersion() -> String
```

Initializing FIDO SDK

This method initializes the FIDO SDK.

API Signature

```
public func initFIDO(serverHost: String, tenant: String, authhubRequest: AuthHubBasicRequest?)
```

This API takes Symantec Identity Security Platform server name as input to initialize the IDSP FIDO SDK.

The API parameters are described in the following table:

Name	Type	Description
serverHost	string	This is the IDSP server host.
tenant	string	This is the tenant name.
authhubRequest	AuthHubBasicRequest	This is an optional parameter and can be set to nil value. If you intent to customize the network request that is being sent to IDSP (like adding a custom header or additional request body param), then you must implement an **AHAPIRequest** interface. The **AuthHubBasicRequest** class is the default implementation of the **AHAPIRequest** interface. When initAuth() is invoked with authhubRequest as nil object, then SDK makes use of the default implementation of the **AHAPIRequest** interface, **AuthHubBasicRequest**.

Example: API call with IDSP Server Host and Tenant

```
let ahfidosdk = AHFIDOSDK.getInstance()
```

```
ahfidosdk.initFIDO(serverHost: "example.com", tenant: "default")
```

API Signature

The API supports the **proxy URL** configuration for the IDSP instance where the tenant name is not required as part of the URL. Ensure that the proxy resolves IDSP instance URL that points to the intended IDSP server host and tenant.

```
public func initFIDO(serverUrl: String, authhubRequest: AuthHubBasicRequest?)
```

This API takes a proxy URL pointing to an IDSP instance as an input to initialize the IDSP FIDO SDK.

The API parameters are described in the following table:

Name	Type	Description
serverURL	string	This is the Proxy URL string pointing to IDSP instance
authhubRequest	AuthHubBasicRequest	This is an optional parameter and can be set to nil value. If you intent to customize the network request that is being sent to IDSP (like adding a custom header or additional request body param), then you must implement an **AHAPIRequest** interface. The **AuthHubBasicRequest** class is the default implementation of the **AHAPIRequest** interface. When initAuth() is invoked with authhubRequest as nil object, then SDK makes use of the default implementation of the **AHAPIRequest** interface, **AuthHubBasicRequest**.

Example: API call with Proxy URL

```
let ahfidosdk = AHFIDOSDK.getInstance()
ahfidosdk.initFIDO(serverUrl: "https://example.com/myproxy", authhubRequest: nil)
```

FIDO Device Registration

This API method is used to register FIDO onto a mobile device.

API Signature

```
public func register(userId: String,
                     domainName: String,
                     deviceName: String,
                     ahConnParams:[String:Any],
                     completionBlock: @escaping (( _ response: AHFIDOResult) -> Void))
```

Parameter Description

- UserId: Enter the user Id with which you are going to register for an FIDO credential. Example - nbruce
- domainName: Enter the domain name of the relying party from where the user is going to perform the transaction. Example - www.example.com
- deviceName: This is the credential name that the user wants to set as the name for the new credential. Example - nbruce's iPhone.
- ahConnParams: This is the collection of IDSP FIDO server-specific parameters that are required for REST API calls. Following are the parameters that are supposed to be passed with proper values:
  a. flowState
  b. bearerToken
  c. deviceTag - (optional) This is the IAData that is returned by IA Risk mobile SDK API (getIAData())
- completionBlock: This is the closure block that is passed from the SDK to the Registration API. AHFIDOResult object is returned back as a response.

NOTE

- With IDSP iOS SDK v3.0, this API **ONLY** registers a FIDO credential by default.
- To register a **PASSKEY** credential, use the new register() API, which accepts credType as an input parameter. See Passkey Device Registration.

Sample Code

```
//Registration of FIDO Credential
      var ahParams = [String:Any]()
        ahParams["flowState"] = xFlowState
        ahParams["bearerToken"] = accessToken
        self.ahfidosdk?.register(userId: userId,
                                 domainName: documentDomain,
                                 deviceName: deviceName,
                                 ahConnParams: ahParams)
         { (regResult) in
              if(regResult.code == RC.Code.SUCCESS_REGISTRATION) {
                  print("message :: \(regResult.message)")
                  print("flowState :: \(String(describing: regResult.data["flowState"]!))")
              }
         }
```

NOTE
The SDK returns an error, FAILURE_FIDO_CREDENTIAL_NOT_FOUND: 909, when the passed FIDO credential is not present on the device. This might occur if a user has already registered a FIDO credential for one device (like Chrome on Windows) and tries to register a new credential for another device (like an Android phone or an iOS phone). See Implementing Registration of Multiple FIDO Credentials and FIDO iOS error codes.

FIDO Device Authentication

This API method is used to authenticate with **FIDO Credentials**.

API Signature

```
public func authenticate(userId: String,
                         domainName: String,
                         ahConnParams:[String:Any],
                         completionBlock: @escaping (( _ response: AHFIDOResult) -> Void))
```

Parameter Description

- UserId: This is the user Id with which you are going to authenticate the FIDO credential. For example-nbruce
- domainName: This is the domain name of the replying party from where the user is performing the transaction. For example- www.example.com
- ahConnParams: This is the collection of IDSP FIDO server-specific parameters that are required for REST API calls. Pass appropriate values for the following parameters:
  a. flowState - The flowState must be passed to the IDSP Server.
  b. bearerToken - The header that denotes the bearerToken that must be passed to the IDSP Server.

     NOTE
     During authentication "flowState" and "bearerToken" are mandatory. "bearerToken" is the ClientAT. After authentication is complete and if there is a business scenario where FIDO authentication must be done then "flowState" does not exist in that scenario and only userAT will be there. In that case, UserAT is passed as the "bearerToken".
  c. deviceTag - (optional) The IAData that is returned by IA Risk mobile SDK API (getIAData()).

d. TrustedDeviceEnabled: This parameter has a boolean value. It denotes whether the IDSP server response indicates that "isTrustedDevice" is enabled or not.

- completionBlock: This is the closure block that is passed from the SDK to the Authenticate API. AHFIDOResult object is returned back as a response.

NOTE
With IDSPiOS SDK v3.0, this API by default authenticates **ONLY FIDO credential**. To **authenticate using already registered passkey credential**, use the new authenticate() API which accepts credType as an input parameter. See Passkey Device Authentication.

Example

```
//Authentication of FIDO Credential
 var ahParams = [String:Any]()
 ahParams["flowState"] = xFlowState
 ahParams["bearerToken"] = accessToken
 ahParams["TrustedDeviceEnabled"] = false
 self.ahfidosdk?.authenticate(userId: userId,
                              domainName: documentDomain,
                              ahConnParams: ahParams)
          { (authResult) in
                  if(authResult.code == RC.Code.SUCCESS_AUTHENTICATION) {
                      print("message :: \(authResult.message)")
                      print("flowState :: \(String(describing: authResult.data["flowState"]!))")
                  }
          }
```

API Response

IDSP FIDO SDK returns the AHFIDOResult object as the response.

```
struct AHFIDOResult {
   public var code : Int
   public var message : String
   public var data : [String:Any]
}
```

IDSP FIDO Result returns code, message, and data as part of the API response. In addition to this, response object returns one more parameter as 'data'.

The data contains values for flowState, nextAction (for example, 'FIDO_AUTH_GENERATE_CHALLENGE') and status (for example, 'ok') and data. 'data' denotes the complete response from the IDSP server in json format.

Enabling FIDO Passkey in IDSP

The FIDO Passkey implementation leverages the existing IDSP FIDO SDK for mobile clients (iOS and Android). This implementation introduces new API methods specifically designed for the FIDO Passkey registration and authentication processes. It is important to note that the implementation of passkey is not interoperable with existing FIDO implementation. Therefore, passkey authentication and FIDO authentication are mutually exclusive and once you choose to use Passkey as a factor to authenticate, you should register and authenticate ONLY using the FIDO Passkey.

NOTE

- To know more about Passkey as an authenticator in IDSP, see Using Passkey.
- Transaction signing using a passkey credential is not supported in IDSP.

To enable passkey, you must first host the "***Apple-App-Site-Association"*** file in your site's ***.well-known*** directory.

Hosting the "*Apple-App-Site-Association*" File to Sync Passkeys between Web and Mobile Clients (Native Apps)

The "***Apple-App-Site-Association***" is a JSON file that allows iOS apps to establish secure communication with their associated websites and provide a secure method to verify domain ownership.

- To establish a connection between your mobile application and the website using Passkey, you must have the corresponding domain file on your website and the necessary entitlement within your app. The applications listed in the "apple-app-site-association" file on your website must correspond to the Associated Domains Entitlement in your app to ensure a seamless connection. For implementing passkey authentication within the native app, refer to the hosting details of the "apple-app-site-association" file at Supporting Associated Domains.
- The following example shows a sample, "apple-app-site-association" file for passkey authentication in the Hedmoral Sample App with IDSP FIDO server:

```
{
  "applinks": {
      "apps": [],
      "details": [
          {
              "appIDs": ["Y2CCP3S9W7.HedmoralSampleApplication"],
              "paths": [
                  "*"
              ],
              "components": [
                  {"/": "/*"}
              ]
          }
      ]
  },
  "webcredentials": {
      "apps": [
          "Y2CCP3S9W7.HedmoralSampleApplication"
      ]
  }
}
```

  where,
  - appIDs and apps Keys - specify the application identifiers for the apps associated with this website, along with their corresponding service types. The format for these values should be "<Application Identifier Prefix>.<Bundle Identifier>.
  - "details" - specific to the "applinks" service type; other service types do not utilize it.
  - "components" - This key is an array of dictionaries used for pattern matching of various components within the URL.
- Once you have constructed the association file, place it in your site's ***.well-known*** directory. The URL for the file should adhere to the following format:

  ```
  https://<fully qualified domain>/.well-known/apple-app-site-association
  ```

  Importantly, you must host the file using HTTPS, with a valid certificate, and without any redirects. This ensures the proper functioning of the association between your website and associated apps.

IDSP Passkey Device Registration

To register Passkey credential, as an application developer you can call the overloaded "registerFactor" API. The passkey supported version of registerFactor is described here.

This API method is used to register Passkey onto a mobile device.

API Signature

```
public func register(userId: String,
                     domainName: String,
                     deviceName: String,
                     credType:String = AHConstants.CRED_TYPE_PASSKEY,
                     anchor: ASPresentationAnchor?,
                     ahConnParams:[String:Any],
                     completionBlock: @escaping (( _ response: AHFIDOResult) -> Void))
```

Parameter Description

- UserId: Enter the user Id with which you are going to register for the PASSKEY credential. Example - nbruce
- domainName: Enter the domain name of the relying party from where the user is going to perform the transaction. Example - www.example.com
- deviceName: This is the credential name that the user wants to set as the name for the new credential. Example - myIphoneXR
- credType: The type of the credential which the user wants to create. Currently, `AHConstants.CRED_TYPE_FIDO` and `AHConstants.CRED_TYPE_PASSKEY` credential types are supported. **The default value for credType is AHConstants.CRED_TYPE_FIDO.** To create a passkey credential, the credType should be AHConstants.CRED_TYPE_PASSKEY.
- ahConnParams: This is the collection of IDSP FIDO/PASSKEY server-specific parameters that are required for REST API calls.

  Following are the parameters that are supposed to be passed with proper values:
  a. flowState - The flowState must be passed to the IDSP Server.
  b. bearerToken - The header that denotes the bearerToken that must be passed to the IDSP Server.
  c. deviceTag - (optional) This is the IAData that is returned by IA Risk mobile SDK API (getIAData())
- anchor - This is the handle to the UI screen on which consent is presented to the user.
- completionBlock: This is the closure block that is passed from the SDK to the Registration API. AHFIDOResult object is returned back as a response.

Sample Code for Registration of Passkey Credential

```
//Registration of Passkey Credential
 var ahParams = [String:Any]()
 var domainName = ""
 ahParams["flowState"] = xFlowState
 ahParams["bearerToken"] = accessToken
 ahParams["TrustedDeviceEnabled"] = false
 ahParams["deviceTag"] = strDeviceTag
 //Get the handle of the viewcontroller window

 guard let window = controller.view.window else { fatalError("The view was not in the app's view
 hierarchy!") }
 //check if passkey is enabled
 var registrationAnchor:UIWindow? = nil
 var domainName = domName
 var deviceName = deviceName
 if(nextAction.caseInsensitiveCompare("PASSKEY_CRED_CREATION") == .orderedSame) {
      registrationAnchor = window
      if let textPasskeyDomain = UserDefaults.standard.string(forKey: AppConstants.PASSKEY_DOMAIN), !
textPasskeyDomain.isEmpty {
          domainName = textPasskeyDomain
          //deviceName = textPasskeyDomain
          var fidoCredType = AHConstants.CRED_TYPE_PASSKEY
          //register passkey
```

```
            self.ahfidosdk?.register(userId: userId,
                           domainName: documentDomain,
                           deviceName: deviceName,
                           FIDOContants.CRED_TYPE_PASSKEY,
                           ahConnParams: ahParams,
                           anchor: registrationAnchor)
            { (authResult) in
                if(authResult.code == RC.Code.SUCCESS_REGISTRATION) {
                    print("message :: \(authResult.message)")
                    print("flowState :: \(String(describing: authResult.data["flowState"]!))"
                }
            }
        } else {
            Logger().logMessage(message: "Invalid passkey domain. Please provide valid passkey domain in
application Settings screen")
        }
}
```

NOTE

- The SDK returns an error, FAILURE_FIDO_CREDENTIAL_NOT_FOUND: 909, when the passed FIDO credential is not present on the device. This might occur if a user has already registered a FIDO credential for one device (like Chrome on Windows) and tries to register a new credential for another device (like an Android phone or an iOS phone). See Implementing Registration of Multiple FIDO Credentials and FIDO iOS error codes.
- Transaction signing using a passkey credential is not supported.

IDSP Passkey Device Authentication

This API method is used to authenticate IDSP Passkey Credentials.

API Signature

```
public func authenticate(userId: String,
                         domainName: String,
                         ahConnParams:[String:Any],
                         credType:String = AHConstants.CRED_TYPE_PASSKEY,
                         anchor: ASPresentationAnchor?,
                         completionBlock: @escaping (( _ response: AHFIDOResult) -> Void))
```

Parameter Description

- UserId: This is the user Id with which you are going to authenticate the PASSKEY credential. For example-nbruce
- domainName: This is the domain name of the replying party from where the user is performing the transaction. For example- www.example.com
- ahConnParams: This is the collection of IDSP FIDO server-specific parameters that are required for REST API calls. Pass appropriate values for the following parameters:
  a. flowState - The flowState must be passed to the IDSP Server.
  b. bearerToken - The header that denotes the bearerToken that must be passed to the IDSP Server.

     NOTE
     During authentication "flowState" and "bearerToken" are mandatory. "bearerToken" is the ClientAT. After authentication is complete and if there is a business scenario where FIDO authentication must be done then "flowState" does not exist in that scenario and only userAT will be there. In that case, UserAT is passed as the "bearerToken".
  c. deviceTag - (optional) The IAData that is returned by IA Risk mobile SDK API (getIAData()).

d. TrustedDeviceEnabled: This parameter has a boolean value. It denotes whether the IDSP server response indicates that "isTrustedDevice" is enabled or not.

- credType: The type of the credential which the user wants to create. Currently, AHConstants.CRED_TYPE_FIDO and AHConstants.CRED_TYPE_PASSKEY credential types are supported. The default value for credType is AHConstants.CRED_TYPE_FIDO. To create a passkey credential, the credType should be AHConstants.CRED_TYPE_PASSKEY.
- anchor: This is the handle to the UI screen on which consent is presented to the user.
- completionBlock: This is the closure block that is passed from the SDK to the Authenticate API. AHFIDOResult object is returned back as a response.

Example

```
//Authentication of Passkey Credential
 var ahParams = [String:Any]()
 var domainName = ""
 ahParams["flowState"] = xFlowState
 ahParams["bearerToken"] = accessToken
 ahParams["TrustedDeviceEnabled"] = false
 var authAnchor:UIWindow? = nil
 //check if we want to create passkey credential
 if(nextAction.caseInsensitiveCompare("PASSKEY_AUTH_GENERATE_CHALLENGE") == .orderedSame) {
      //Get the handle of the viewcontroller window
      guard let window = controller.view.window else { fatalError("The view was not in the app's view
 hierarchy!") }
      authAnchor = window
      if let textPasskeyDomain = UserDefaults.standard.string(forKey: AppConstants.PASSKEY_DOMAIN), !
textPasskeyDomain.isEmpty {
         domainName = textPasskeyDomain
      } else {
          Logger().logMessage(message: "Invalid passkey domain. Please provide valid passkey domain in
 application Settings screen")
      }
      credType = AHConstants.CRED_TYPE_PASSKEY

      self.ahfidosdk?.authenticate(userId: userLogInId,
                             domainName: domainName,
                             ahConnParams: ahParams,
                             credType: credType,
                             anchor: authAnchor)
         { (authResult) in
              if(authResult.code == RC.Code.SUCCESS_AUTHENTICATION) {
              print("message :: \(authResult.message)")
              print("flowState :: \(String(describing: authResult.data["flowState"]!))")
         }
      }
 }
```

API Response

IDSP FIDO SDK returns the AHFIDOResult object as the response.

```
struct AHFIDOResult {
```

```
    public var code : Int
    public var message : String
    public var data : [String:Any]
}
```

IDSP FIDO Result returns code, message, and data as part of the API response. In addition to this, response object returns one more parameter as 'data'.

The data contains values for flowState, nextAction (for example, 'FIDO_AUTH_GENERATE_CHALLENGE') and status (for example, 'ok') and data. 'data' denotes the complete response from the IDSP server in json format.

Transaction Signing using FIDO

IDSP protects authenticated user session attacks (for example, MitB attacks) by supporting FIDO-based transaction data signing. The FIDO iOS SDK verifies and signs the transaction challenge request that the Relying Party initiates for the sensitive user transactions.

> **NOTE**
> Transaction signing using a passkey credential is not supported in IDSP.

Transaction Signing Flow:

1. When a user initiates a sensitive transaction on a mobile device, the mobile application invokes the business API of the Relying Party.
2. The Relying Party initiates the transaction signing flow by invoking the **FIDOTransactionSigningChallengeGenerator** API of the IDSP server for a challenge. For information about the server API, see Transaction Signing.
3. The IDSP server generates and returns the challenge (SVR_CHL) to the Relying Party.
4. On receiving the transaction data and challenge details from the Relying Party by the calling API, the mobile application prompts the user to confirm the transaction data (TXN_DATA). For example, "Do you want to transfer $1000 to John?". If the user denies the transaction, the transaction signing flow ends with a proper error message that is displayed to the user.
5. If the user approves the transaction, the mobile application invokes the FIDO SDK **signTransaction** API.

   API Signature

   ```
   signTransaction(
     domainName: String,
     transactionData: String,
     assertionOptionsResponse:[String:Any],
     completionBlock: @escaping (( _ response: AHFIDOResult) -> Void))
   ```

   Sample Code

   ```
   var response : [String : Any] = [:]
   var apiResponse = [String:Any]()
   // apiResponse is theIDSP server response for the server API FIDOTransactionSigningChallengeGenerator.
   // JSON response from the server should be converted into a dictionary of String and Any pairs.

   let transactionMessage = "Do you want to transfer $1000 to John?”

   //Base64 encode the message string.
   let transactionData = Common.encodeBase64(str:transactionMessage)

   //domain name of the relying party from where the user is going to perform the transaction.
   let domainName = “www.example.com”
   self.ahfidosdk?.signTransaction(domainName: domainName,
                                   transactionData: transactionData,
                                   assertionOptionsResponse: apiResponse)
   ```

```
{ (authResult) in

if(authResult.code == RC.Code.SUCCESS_TRANSACTION_SIGNING) {

print("message :: \(authResult.message)")

response = authResult.data

//transactionData has to be passed to the Relying Party API where it adds transactionData and invokes
 FIDOTransactionSigningChallengeVerifier IDSP server API to complete the transaction flow.
}
```

Parameter Description

- domainName: Specifies the domain name of the Relying Party. For example, www.example.com
- transactionData: Specifies a string that the Relying Party denotes as the transaction data.
- assertionOptionsResponse: Represents the challenge response that the IDSP server returns in response to the **FIDOTransactionSigningChallengeGenerator** API.
- completionBlock: Represents the closure block that the SDK returns to the calling mobile application API. AHFIDOResult object is returned back as a response.

6. The **signTransaction** API generates a new transaction challenge (TXN_CHL) by concatenating transaction data, server challenge (passed in the assertionOptionsResponse attribute), and tilde '~'.
   TXN_CHL = SVR_CHL + '~' + TXN_DATA
7. The SDK challenges the user for Biometrics (FaceID, TouchID) or Security Key depending on the authentication factor that is configured on the mobile device. Once the identity of the user is verified, the SDK unlocks the private key that is stored on the mobile device and uses it to sign the transaction challenge (TXN_CHL).
8. The **signTransaction** API returns the signed signature to the calling mobile application as a AHFIDOResult object.
   Sample Code for authResult.data

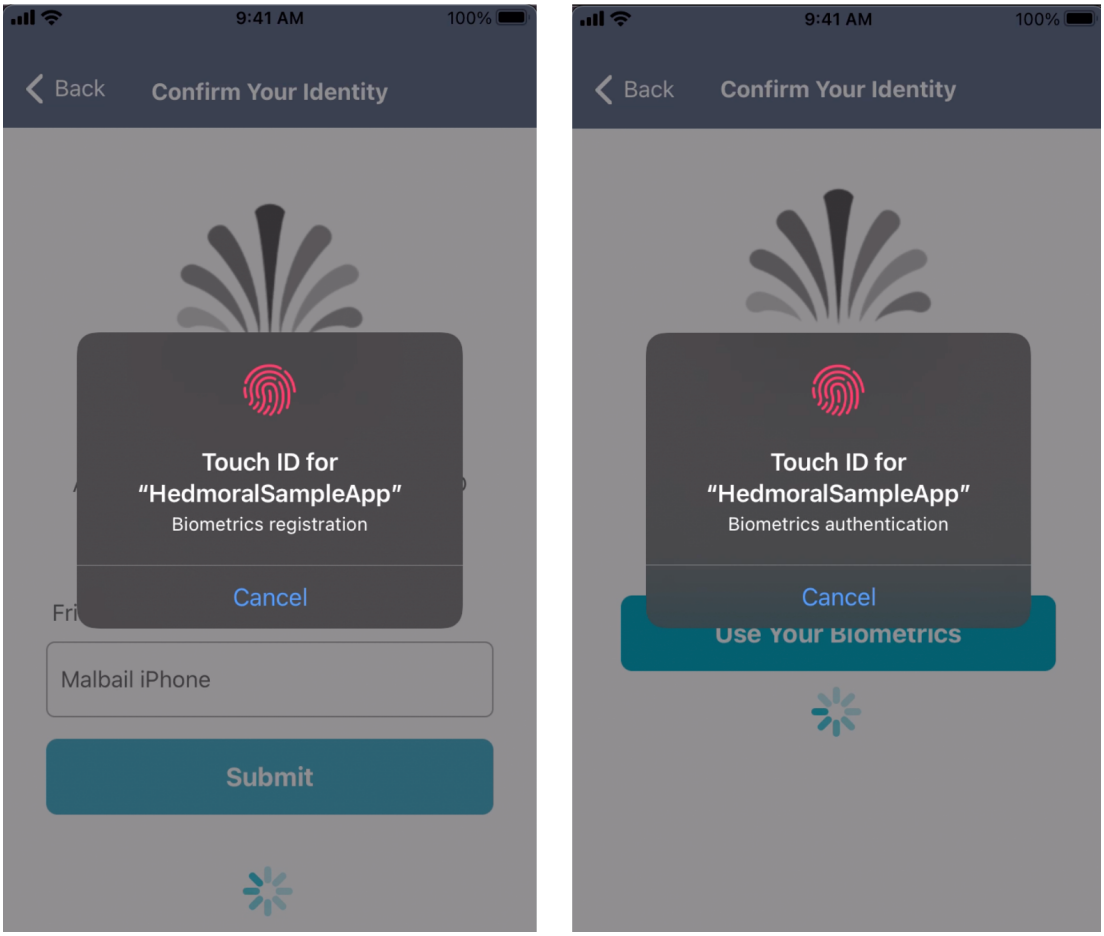
```
{
"rawId": "2BiknB-5xPd8uWtaJ59V3nfbb4HUQicd7V82AthLxOs",
"response": {
  "authenticatorData": "jl48hdEAN0kIRcYwrC7TwCI27jApz656ec7rgfPWJXMFAAAAAg",
  "signature": "MEUCIHo_k7NMIh3CWS10yKr8bBqiqIEANdnTvPpvWuCaB1kKAiEAvwNMfqdjslKIFOFJAOjmU0d8JFzoe-
ruW22TzLj3XlM",
  "userHandle": "MUlVcFdYQVZySkpUa0ptdjV2UTBadEp3WHZQdXFfVHNxSmRlYk1NUFczOA",
  "clientDataJSON":
 "eyJ0eXBlIjoid2ViYXV0aG4uZ2V0IiwiY2hhbGxlbmdlIjoiWkRGRVFtdHBNbFJLWTNOMWMxbzRkMmwyYTNsSlpYWmFWWEZmVWpOVVdUTmpTbEJsVl
},
  "getClientExtensionResults": {},
  "id": "CKCHRI1DRYqlj3KG3PjHXg",
  "type": "public-key"
}
```

9. The mobile application returns the signed transaction challenge as a response to the Relying Party.
10. The Relying Party invokes the **FIDOTransactionSigningChallengeVerifier** API of the IDSP server. The API signature includes the transaction data and the signed transaction challenge for verification. For information about the server API, see Transaction Signing.
11. The Challenge Verifier API returns the validation status to the Relying Party as a response.
12. Depending on the validation status, the mobile application displays the transaction summary or an error message to the user.

Localizing the FIDO Registration and Authentication Biometric Messages

The AuthHubFIDO iOS SDK provides localization support for the biometric messages that are displayed on the mobile device at the time of registering and authenticating FIDO credentials.

Figure 23: FIDO Registration and Authentication Biometric Messages



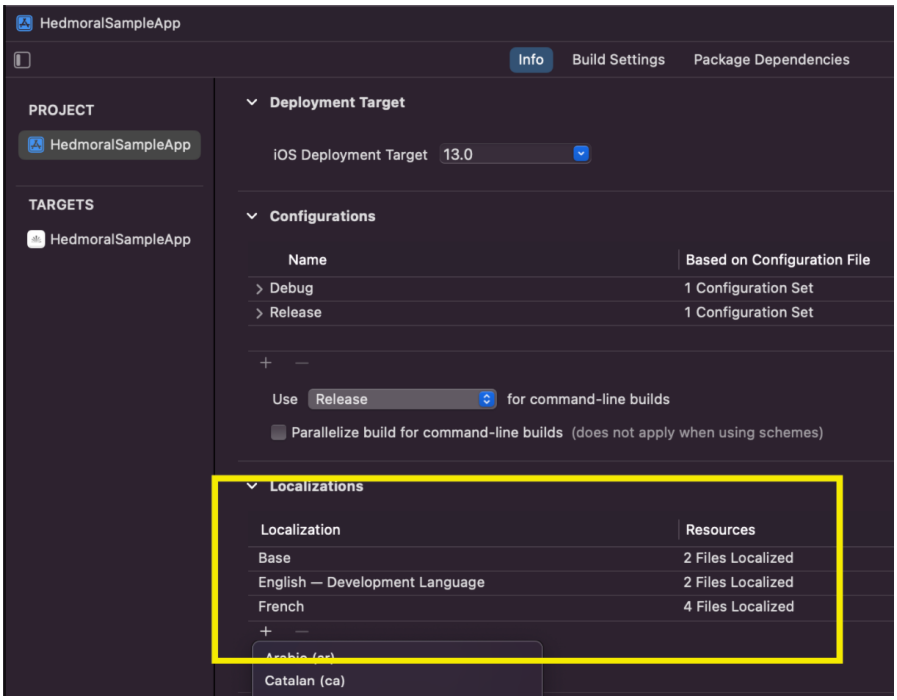
Follow these steps to add localization support for the FIDO registration and authentication biometric messages:

1. Download the **AuthHubLocalizable.strings** file from the release package, which is available on the Support Portal > My Downloads (Search for IDSP in the portal and select the appropriate version number)
2. The **AuthHubLocalizable.strings** file contains the following default FIDO registration and authentication biometric messages:

```
/*
AuthHubLocalizable.strings
AuthHubFIDOSDK
*/
"AH_BIOMETRIC_REGISTRATION_MSG"="Biometrics registration";
"AH_BIOMETRIC_AUTHENTICATION_MSG"="Biometrics authentication";
```

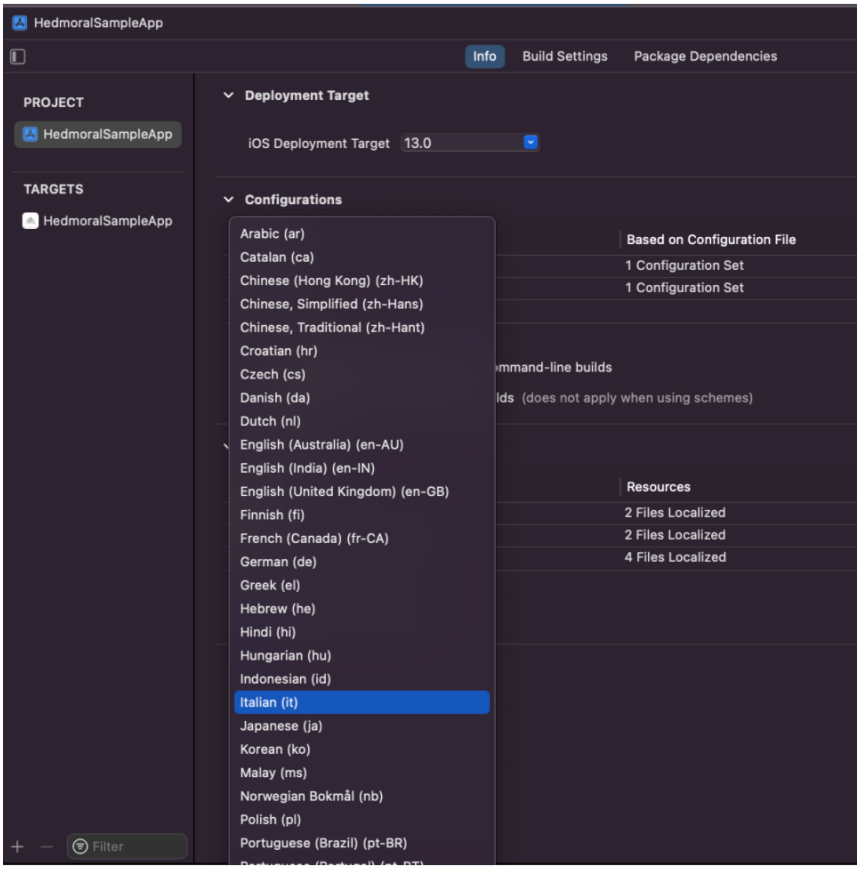
   You can customize the biometric messages as per your mobile application requirements.
3. Copy the **AuthHubLocalizable.strings** file into the XCode project base localization directory (en.lproj, where English is the default application development language).
4. Create localization folders to support multiple locales as per the application requirements.
   a. In the XCode project, click the "+" icon in the **Localizations** section.

Figure 24: Create Localization Folders for Locales



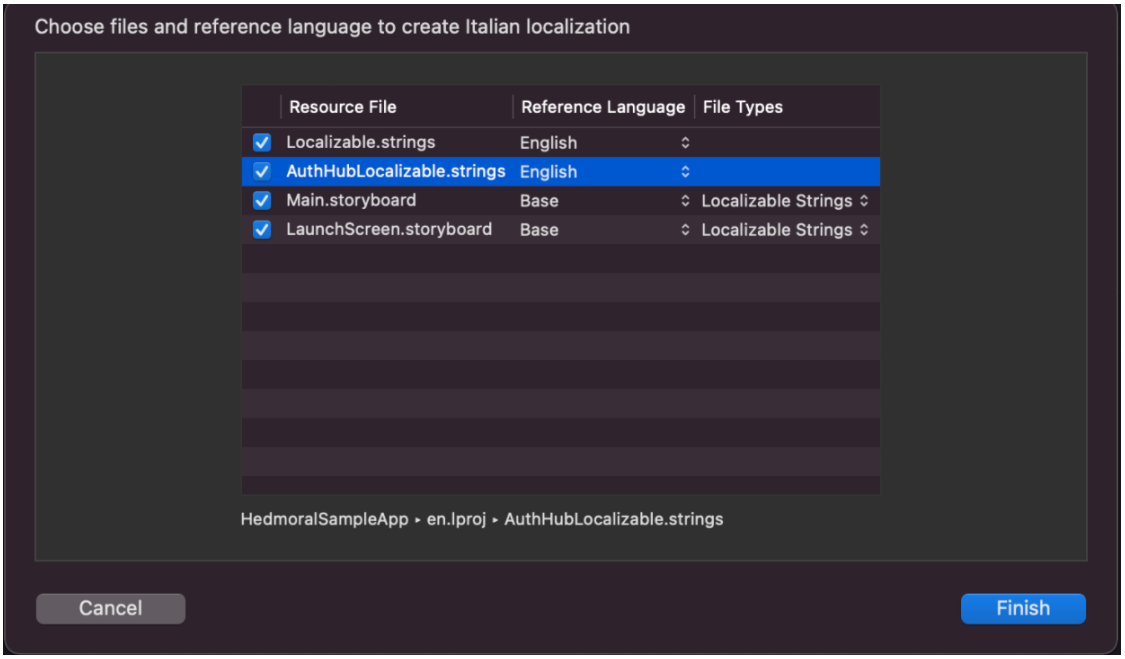
b. Select a locale. For example, Italian (it).

Figure 25: Select a Locale



c. Select **AuthHubLocalizable.strings** as the resource file and **English** as the reference language, and click **Finish.**

Figure 26: Select a Resource File and a Reference Language



d. The XCode project creates a localization folder for the selected locale (example, **it.lproj** folder is created for Italian) and adds the **AuthHubLocalizable.strings** file to it.

Figure 27: Example of a Localization Folder Created for the Italian Locale

e. Open the **AuthHubLocalizable.strings** file from the Localization folder (example, it.lproj) of a locale and translate the biometric messages into the corresponding language (example, Italian). XCode project now includes localized biometric messages.

f. The AuthHub FIDO SDK displays the biometric messages in the language that is set on the mobile device. If localization support for a locale is not included in the XCode project, then the SDK displays the biometric messages in English.

Reference Link:

- iOS SDK Error messages and codes
- AuthHubCore SDK Framework

iOS SDK Error Codes

View the error codes and messages related to the iOS SDK.

This topic includes iOS SDK error messages and codes.

Table 90: Error Codes for FIDO iOS SDK

Error Strings	Error Codes
SUCCESS	0
SUCCESS_REGISTRATION	0
SUCCESS_AUTHENTICATION	0
SUCCESS_TRANSACTION_SIGNING	0
FAILURE_REGISTRATION	10
FAILURE_AUTHENTICATION	11
FAILURE_TRANSACTION_SIGNING	12
FAILURE_SDK_INITIALIZATION_ERR	13

Table 91: HTTP Error Codes for FIDO iOS SDK

Error Strings	Error Codes
FAILURE_NW_ERROR	100
FAILURE_BAD_REQUEST	101
FAILURE_UNAUTHORIZED	102

FAILURE_FORBIDDEN	103
FAILURE_NOT_FOUND	104
FAILURE_METHOD_NOT_ALLOWED	105
FAILURE_PROXY_AUTH_REQUIRED	106
FAILURE_SERVER_ERROR	107
FAILURE_BAD_GATEWAY	108
FAILURE_SERVICE_UNAVAILABLE	109
FAILURE_GATEWAY_TIMEOUT	110
FAILURE_URL_BUILDER_ERR	111

Table 92: Response Error Codes for FIDO iOS SDK

Error Strings	Error Codes
FAILURE_WEBAUTHN_BAD_DATA	800
FAILURE_WEBAUTHN_BAD_OPERATION	801
FAILURE_WEBAUTHN_INVALID_STATE	802
FAILURE_WEBAUTHN_USER_VERIFICATION_ERR	803
FAILURE_WEBAUTHN_OPERATION_CANCELLED	804
FAILURE_WEBAUTHN_OPERATION_TIMEOUT	805
FAILURE_WEBAUTHN_CREDENTIAL_NOT_FOUND	806
FAILURE_WEBAUTHN_UNSUPPORTED_AUTHENTICATOR	807
FAILURE_WEBAUTHN_UNKNOWN_REASON	808
FAILURE_WEBAUTHN_CREDENTIAL_SAVE_ERROR	809
FAILURE_WEBAUTHN_SIGN_ERROR	810
FAILURE_WEBAUTHN_USER_FALLABACK	811
FAILURE_WEBAUTHN_USER_CANCEL	812
FAILURE_WEBAUTHN_AUTH_ERROR	813
FAILURE_WEBAUTHN_PASSCODE_NOT_SET	814
FAILURE_WEBAUTHN_SYSTEM_CANCEL	815
FAILURE_WEBAUTHN_BIOMETRY_LOCKOUT	816
FAILURE_WEBAUTHN_BIOMETRY_NOT_AVAILABLE	817
FAILURE_WEBAUTHN_BIOMETRY_NOT_ENROLLED	818
FAILURE_WEBAUTHN_APP_CANCEL	819
FAILURE_WEBAUTHN_AUTH_POLICY_EVAL_ERROR	820
FAILURE_WEBAUTHN_TRANSPORT_MISMATCH	821
FAILURE_WEBAUTHN_AUTHENTICATOR_UNAVAILABLE	822
FAILURE_INVALID_URL	900
FAILURE_INVALID_CLIENT_ID	901
FAILURE_INVALID_CLIENT_SECRET	902
FAILURE_INVALID_CLIENT_NONCE	903
FAILURE_PUBLIC_KEY_ENC	904
FAILURE_SERVER_NONCE_DECRYPTION	905

FAILURE_PASSWORD_ENCRYPTION	906
FAILURE_FIDO_DOMAIN_NAME	907
FAILURE_INIT_AUTH_ERROR	908
FAILURE_FIDO_CREDENTIAL_NOT_FOUND	909
FAILURE_FIDO_AUTH_ERROR	910
FAILURE_FIDO_REG_ERROR	912
FAILURE_FIDO_REG_EXCLUDE_CRED_ERROR	913
FAILURE_UNKNOWN_ERR_MSG	998
FAILURE_MISC_ERROR	999
FAILURE_RESPONSE_DATA_ERROR	1000
FAILURE_RESPONSE_NOT_OK	1001
FAILURE_RESPONSE_CHALLENGE_MISSING	1002
FAILURE_RESPONSE_USER_MISSING	1003
FAILURE_RESPONSE_RP_MISSING	1004
FAILURE_RESPONSE_AUTH_SELECT_MISSING	1005
FAILURE_RESPONSE_ATTESTATION_MISSING	1006
FAILURE_RESPONSE_PUBLICKEYCREDS_MISSING	1007
FAILURE_RESPONSE_RPID_MISSING	1008
FAILURE_RESPONSE_UV_MISSING	1009
FAILURE_RESPONSE_ALLOWCREDS_MISSING	1010
FAILURE_RESPONSE_BRTOKEN_MISSING	1011
FAILURE_RESPONSE_TENANT_MISSING	1012
FAILURE_RESPONSE_TXN_DATA_MISSING	1013
FAILURE_RESPONSE_AH_RESPONSE_MISSING	1014
FAILURE_RESPONSE_TXN_CONFIRMATION_DENIED	1015
FAILURE_RESPONSE_SERVER_RESPONSE_EMPTY	1016
FAILURE_RESPONSE_PASSKEY_REGISTRATION_ERROR	1021
FAILURE_RESPONSE_PASSKEY_AUTH_ERROR	1022
FAILURE_RESPONSE_PASSKEY_REGISTRATION_CANCELED	1023
FAILURE_RESPONSE_PASSKEY_AUTH_CANCELED	1024
FAILURE_RESPONSE_PASSKEY_REGISTRATION_DUPLICATE_PASSKEY	1025
FAILURE_RESPONSE_PASSKEY_REGISTRATION_SYSTEM_CANCELED_OR_TIMEOUT	1026
FAILURE_RESPONSE_PASSKEY_AUTH_SYSTEM_CANCELED_OR_TIMEOUT	1027

Table 93: Error Codes for Auth Core iOS SDK

Error Messages	Error Codes
FAILURE_DATA_ENCRYPTION	911
FAILURE_RESPONSE_SERVER_NONCE	1017
FAILURE_RESPONSE_INITIATE_AUTH	1018
FAILURE_RESPONSE_SERVER_RESPONSE_EMPTY	1019

Table 94: Messages for FIDO iOS SDK

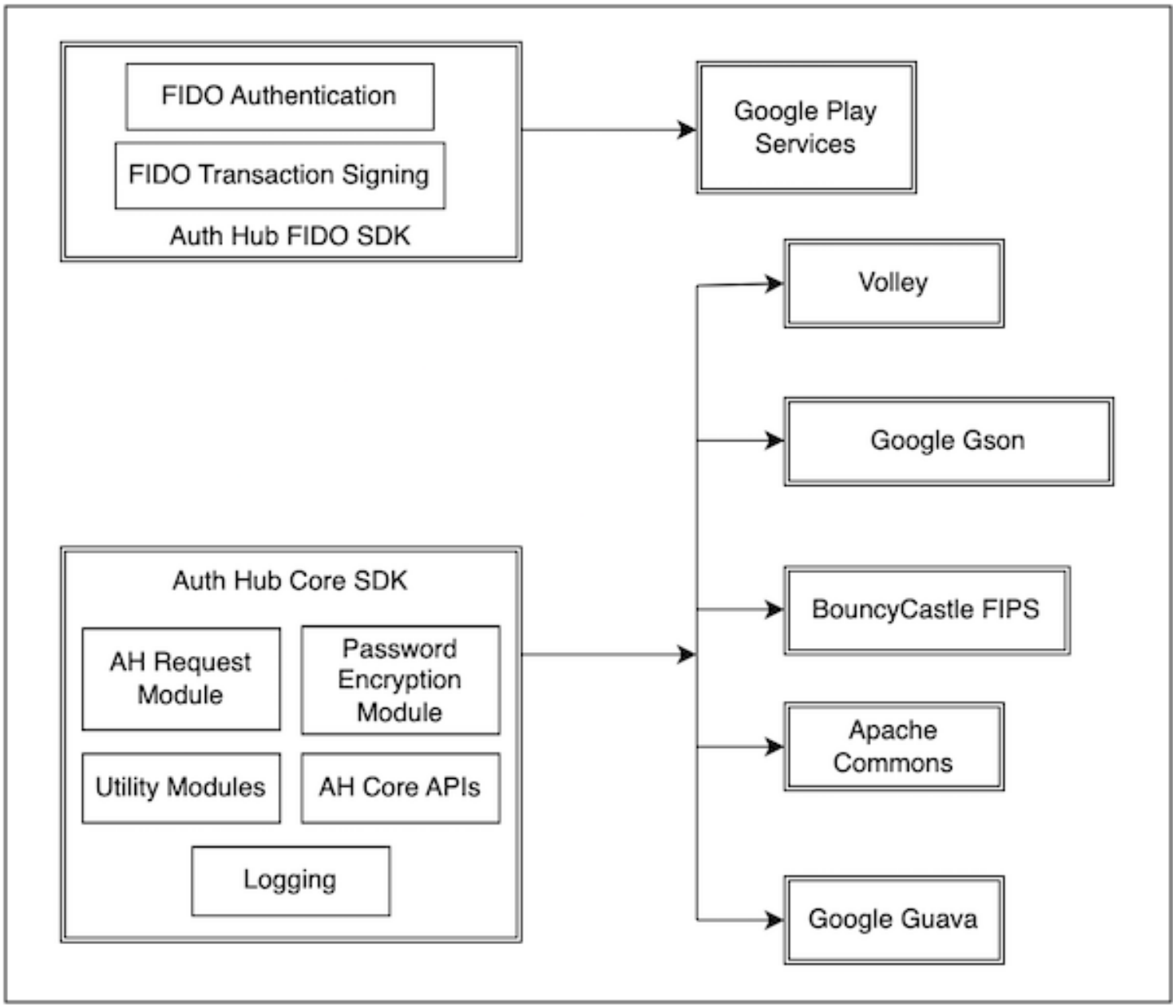
Error Strings	Error Messages
SUCCESS_REGISTRATION_MSG	"FIDO registration successful"
SUCCESS_AUTHENTICATION_MSG	"FIDO authentication successful"
SUCCESS_TRANSACTION_SIGNING_MSG	"FIDO transaction signing successful"
SUCCESS_VALIDATION_MSG	"Validation successful"
FAILURE_REGISTRATION_MSG	"FIDO registration failed"
FAILURE_AUTHENTICATION_MSG	"FIDO authentication failed"
FAILURE_TRANSACTION_SIGNING_MSG	"FIDO transaction signing failed"
FAILURE_NW_ERROR_MSG	"Network error"
FAILURE_RESPONSE_DATA_ERROR_MSG	"Data missing in Server API response"
FAILURE_RESPONSE_NOT_OK_MSG	"Server API response not OK"
FAILURE_RESPONSE_CHALLENGE_MISSING_MSG	"Challenge missing in Server API response"
FAILURE_RESPONSE_USER_MISSING_MSG	"User missing in Server API response"
FAILURE_RESPONSE_RP_MISSING_MSG	"RP missing in Server API response"
FAILURE_RESPONSE_AUTH_SELECT_MISSING_MSG	"Authenticator Selection missing in Server API
FAILURE_RESPONSE_ATTESTATION_MISSING_MSG	"Attestation missing in Server API response"
FAILURE_RESPONSE_PUBLICKEYCREDS_MISSING_MSG	"PublicKeyCreds missing in Server API respon
FAILURE_RESPONSE_RPID_MISSING_MSG	"RPId missing in Server API response"
FAILURE_RESPONSE_UV_MISSING_MSG	"UserVerification missing in Server API respon
FAILURE_RESPONSE_ALLOWCREDS_MISSING_MSG	"AllowCredentials missing in Server API respo
FAILURE_RESPONSE_BRTOKEN_MISSING_MSG	"Bearer Token not passed"
FAILURE_RESPONSE_TENANT_MISSING_MSG	"Tenant not passed"
FAILURE_RESPONSE_TXN_DATA_MISSING_MSG	"Transaction-Data not passed"
FAILURE_RESPONSE_AH_RESPONSE_MISSING_MSG	"AuthHub response not in proper format"
FAILURE_RESPONSE_TXN_CONFIRMATION_DENIED_MSG	"FIDO Transaction confirmation denied by the
FAILURE_RESPONSE_SERVER_RESPONSE_EMPTY_MSG	"AuthHub Server response is empty"
FAILURE_RESPONSE_PASSKEY_REGISTRATION_ERROR	"Passkey registration failed"
FAILURE_RESPONSE_PASSKEY_AUTH_ERROR	"Passkey authentication failed"
FAILURE_RESPONSE_PASSKEY_REGISTRATION_CANCELED	"Passkey registration cancelled"
FAILURE_RESPONSE_PASSKEY_AUTH_CANCELED	"Passkey authentication cancelled"
FAILURE_RESPONSE_PASSKEY_REGISTRATION_DUPLICATE_PASSKEY	"Passkey registration failed. A passkey with th
FAILURE_RESPONSE_PASSKEY_REGISTRATION_SYSTEM_CANCELED_OR_TIMEOUT	"Passkey registration operation timed out or ca
FAILURE_RESPONSE_PASSKEY_AUTH_SYSTEM_CANCELED_OR_TIMEOUT	"Passkey authentication operation timed out o

Symantec Identity Security Platform Android SDK

Understand about the Android Client SDK and it is two components, Auth Hub Core and Auth Hub FIDO SDK.

The Symantec Identity Security Platform Android SDK has been explained with the help of the following diagram:

Figure 28: IDSP Android Client SDK



The IDSP Android SDK comprises of two components: - Auth Hub Core and Auth Hub FIDO SDK

- Auth Hub Core SDK includes IDSP Request Module (Network Layer), Password Encryption Module, Logging Module, and Utility APIs.
- Auth Hub FIDO SDK supports existing APIs related to the FIDO authentication flow, as well as APIs related to FIDO Transaction Signing.

Integration with Android App

Symantec Identity Security Platform provides an Android SDK that enables app developers to integrate with their Android app to perform password encryption, FIDO (supports biometric and Security key), and Transaction Signing capabilities.

To integrate IDSPAndroid SDKs with an Android application, perform the following steps:

- Configure the Package and Key-Store Settings
- Configure the dependencies in the "Gradle" build tool
- Add permissions in the "AndroidManifest.xml" file
- Create an "assetlinks.json" file (used only for integrating FIDO capability with Android apps)

Configuring Package and Key-Store settings

To use the IDSP Android SDK and its functionality, you must configure the following settings in the application:

1. Create an Android application with a Package Name and Key-Store dependency as shown in the following example. in the assetlinks.json file. To create a "assetlinks.json" file, see "here.
   If you are using your own "assetlinks.json" file, you can set the name of the package and the information of the key-store to your discretion. However, if you plan to use the "assetlinks.json" file provided by IDSP, then create your sample application using the following defaults:
   1. Package name: com.entersekt.fido2
   2. Keystore Name: keystore.jks (The keystore.jks file is available in the HEDMORAL Android Sample app)
   3. storePassword: fido2password
   4. KeyAlias: fido2

      NOTE
      The IDSP "assetlinks.json" should only be used for demo purposes and should not be used in the production environment.

Steps to create a Keystore file in Android

To create a keystore file, follow these steps:

1. On the menu bar of the Android studio, click Build > Generate Signed APK.
2. On the Generate Signed APK Wizard window, click Create new to create a keystore.
3. On the New Key Store window, provide the required information.
4. On the Generate Signed APK Wizard window, select a keystore, a private key, and enter the passwords for both. Then click Next.
5. On the next window, select a destination for the signed APK and click Finish.

Configuring dependencies in the "Gradle" build tool

You must configure the following "App Level" dependencies in the "dependencies" section of the "***build.gradle"*** file.

NOTE

- Ensure to replace the text including the braces, <latest version> with the actual value of the latest version in the following APIs.
- The following steps are not sequential, so they can be performed in any sequence; we have shown it in a sequence for your convenience.

1. Configure the following "**Gson**" dependency in the file:
   ```
   implementation 'com.google.code.gson:gson:<latest version>'
   ```
2. Configure the following "**Volley** " dependency in the file:
   ```
   implementation 'dev.dworks.libs:volleyplus:<latest version>'
   ```
3. Configure the following "**Firebase**" dependencies in the file:
   Ensure to replace the text including the braces, <latest version> with the actual value of the latest version in the following APIs.
   a. "Google Actions, Base Client Library" API:

```
implementation 'com.google.android.gms:play-services-base:<latest version>'
```

b. "Google Account Login" API:

```
implementation 'com.google.android.gms:play-services-auth:<latest version>'
```

c. "Google Services FIDO" API:

```
implementation 'com.google.android.gms:play-services-fido:<latest version>'
```

4. Add the following "**Fido and AuthCore** dependencies in the file:

NOTE
Obtain the "authfido-release" and the "authcore-release" files from this <location TBD>. These files have the "aar" extension.

a. `implementation(name:'authfido-release', ext:'aar')`

b. `implementation(name:'authcore-release', ext:'aar')`

Adding permissions in the "AndroidManifest.xml" file

This file is stored in the root directory of the Project Hierarchy.

Add the following permissions in the "AndroidManifest" file:

NOTE
The following permissions can be applied anywhere in the manifest file other than the < application > tag.

- `<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />`
- `<uses-permission android:name="android.permission.INTERNET" />`

Creating an "assetlinks.json" file

NOTE
Perform this step only if you intend to integrate and use the FIDO capability offered by IDSP in your Android app.

Create a assetlinks.json file.

The following is a sample of an assetlinks.json file content: Update the "package_name" and "sha256_cert_fingerprints" in this file from the application and keystore that you have created in ....section respectively.

```
[ {
  "relation" : [ "delegate_permission/common.handle_all_urls" ],
  "target" : {
    "namespace" : "android_app",
    "package_name" : "com.entersekt.fido2test",
    "sha256_cert_fingerprints" :
 [ "9E:63:92:4E:DD:33:E9:35:C6:78:42:E8:AA:97:6C:C7:76:AD:1E:97:07:79:07:68:5D:F7:9B:D5:E3:4C:2F:F7" ]
  }
} ]
```

NOTE

IDSP defines the "assetlinksJson" as one of the attributes in the tenant settings and also provides an out-of-the-box embedded assetlinks.json file with seeded data for your convenience. You can use the embedded file to update the following attributes:

- package_name
- sha256_cert_fingerprints

To update these attributes in this file, perform a patch operation using the following API:

```
PATCH https://${SSP_FQDN}/{{tenantName}}/admin/v1/Configs
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/Configs`
  should look similar to:
  `https://example.com/default/admin/v1/Configs`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

The Patch sample would look like this:

```
[
    {
        "name": "assetlinksJson",
        "value": "[{ \"relation\": [\"delegate_permission/common.handle_all_urls
\"], \"target\": {  \"namespace\": \"android_app\",            \"package_name\":
 \"com.entersekt.fido2\",            \"sha256_cert_fingerprints\": [
  \"C5:8B:E3:9B:36:B3:67:12:D7:0C:DA:C5:9D:65:2A:FC:43:9B:AE:1B:76:C9:7D:A1:7E:69:2B:7A:15:AB:27:96\" ]  } }]"
    }
]
```

After you complete your updates to the file, use the following API to retrieve the contents of the file with your updates:

```
GET https://mgmt-<ssp-fqdn>/<tenant-name>/.well-known/assetlinks.json
```

To know more about FIDO2 standards on how to create and host the assetlinks.json file, refer to these links:

- https://developers.google.com/identity/smartlock-passwords/android/associate-apps-and-sites
- https://developers.google.com/identity/fido/android/native-apps

Symantec Identity Security Platform Android Core SDK

The Android Software Development Kit (SDK) includes Symantec Identity Security Platform Request Module, Password Encryption Module, Network Module, Logging, and Utility APIs.

This page includes the following topics:

- Network Request Module
- Prerequisite for "Password" and "Device Signature" Encryption
- Initiate Authentication
- Data Encryption ("Password" and "Device Signature")
  - Encryption Check
  - Support for Proxy URL Configuration
  - Get the list of fields to be Encrypted
  - Encrypt "Password" and "Device Signature"

IDSP Network Request Module (Optional)

IDSP Network Request Module is a part of the AuthHubCore framework. The IDSP Client SDK APIs are designed to handle the network request and response for all the IDSP APIs, such as, "/authentication", "/BrandingSettings" and have complete control over them. Apart from handling essential API request payload/headers/query params, if you intend to customize the network layer to use it for tasks like adding custom headers or supporting SSL pinning, you must implement

three interfaces in your application to handle the network request customization. These interfaces are provided by IDSP and are explained in the following sections:

- ***AuthHubEndPoint***: This interface provides the set of APIs that mobile applications can implement to customize the network request. Using these API implementation, the mobile application can override the headers, add custom headers, or add new parameters to the request payload as required.
  This interface defines the following APIs -

```
public void setRequestPayloadParameters(JSONObject params);
public void setRequestHeaders(Map<String, String> headers);
public Request build(Response.Listener<JSONObject> successListener, Response.ErrorListener errorListener);
```

  The following table explains the APIs of this interface:

API Name	Description
setRequestPayloadParameters()	Accepts the JSONObject that is sent in the body of the network request.
setRequestHeaders()	Accepts the map of key-value pairs which is sent as a request header of the network request.
build()	Builds an actual network request object which is sent to the IDSP server and accepts Volley Error and Success Listeners.

- ***AuthHubRequestLogger***: This interface provides the APIs required for logging the network level data and logs the time that is required for the network calls. The AuthHubRequestLogger provides the ability to log network request level information, such as request headers, request body, request response body, and response time.

  NOTE
  The AuthHubRequestLogger interface must be implemented if you want to add custom logging to the network request object.

  This interface defines the following APIs -

```
public void enableLog(boolean flag);
public void setLoggingType(RequestLoggingType type);
public void logRequestData(double responseTime, AHEndPoint ahEndPoint, String responseBody);
```

  The following table explains the APIs of this interface:

API Name	Description
enableLog()	Enables logging for a network request. By default it is ***disabled***.
setLoggingType()	Sets the logging type of the request logger. The RequestLoggingType is an ***enum*** that can have these values: ***resp_time***, ***data***, and ***none***. • resp_time - This type of logger logs only the response time • data - This type of logger logs the request data, such as, headers, request body, response body, and response time • none - This type of logger does not log any data
logRequestData()	This API is used for logging the request data as per the RequestLoggingType.

  – ***AHRequestLogger*** : This class is the default implementation of the AuthHubRequestLogger interface. By default, this class logs all request data to the console.

    NOTE
    You must create an instance of this class and must initialize it with the name of the log file, if you want the logger to write logs to the file.

- ***AHAPIRequest***: This interface provides the APIs required to handle network level functionality, such as, sending requests, logging network request. This interface defines the following APIs -

```
public void sendRequest(Context context, AHEndPoint AHEndPoint, Response.Listener<JSONObject>
 successListener, Response.ErrorListener errorListener);
public void setRequestLogger(AuthHubRequestLogger authHubRequestLogger);
```

The following table explains the APIs of this interface:

API Name	Description
sendRequest()	This API accepts the Android Context, AuthHubEndpoint instance object, Volley Success, and Error Listener.
setRequestLogger()	This API sets the AuthHubRequestLogger object for the network request object.

– ***AHBasicAPIRequest***: This class is the default implementation of the ***AHAPIRequest*** interface.

The following example shows a sample implementation of the IDSP network request implementation that is defined previously.

```
package com.entersekt.fido2;


import android.content.Context;

import com.android.volley.AuthFailureError;
import com.android.volley.Response;
import com.android.volley.toolbox.Volley;
import com.broadcom.layer7authentication.authcore.network.AHAPIRequest;
import com.broadcom.layer7authentication.authcore.network.AHEndPoint;
import com.broadcom.layer7authentication.authcore.network.AuthHubRequestLogger;
import com.broadcom.layer7authentication.authcore.network.CustomJSONObjectRequest;

import org.json.JSONObject;

import java.util.HashMap;
import java.util.Map;

public class SampleAuthHubRequest implements AHAPIRequest {

AuthHubRequestLogger authHubRequestLogger;

@Override
public void sendRequest(Context context, AHEndPoint ahEndPoint, Response.Listener<JSONObject> successListener,
 Response.ErrorListener errorListener) {

// Always get existing Request Headers first if not done it will override existing headers with newer header
 objects
Map<String, String> requestHeader = ahEndPoint.getRequestHeaders();

// Add any new header
requestHeader.put("TestKey", "TestValue");

// Set updated headers object to AHEndPoint headers
ahEndPoint.setRequestHeaders(requestHeader);

// Build Custom JSON Object Request Object
```

```
CustomJSONObjectRequest jsonObjectRequest = (CustomJSONObjectRequest) ahEndPoint.build(successListener,
 errorListener);
jsonObjectRequest.setRequestLogger(this.authHubRequestLogger, ahEndPoint);
// Add to volly request queue for API execution
Volley.newRequestQueue(context).add(jsonObjectRequest);
}

@Override
public void setRequestLogger(AuthHubRequestLogger authHubRequestLogger) {
this.authHubRequestLogger = authHubRequestLogger;
}
}
```

The following example shows how to use the sample request that was created in the previous example to initialize the IDSP FIDO SDK instance:

```
/*Initialization of SampleAuthHubRequest*/
SampleAuthHubRequest sampleAuthHubRequest;


/*Create instance of SampleAuthHubRequest*/
sampleAuthHubRequest = new SampleAuthHubRequest();



/*Create instance of AHRequestLogger and enable it*/

AHRequestLogger ahRequestLogger = new AHRequestLogger();
ahRequestLogger.enableLog(true);
ahRequestLogger.setLoggingType(RequestLoggingType.data);



/*Set request logger for created Sample Request object*/
sampleAuthHubRequest.setRequestLogger(ahRequestLogger);



/*Set Sample Request Object in AuthHub Fido Object*/
authFido.setApiRequest(sampleAuthHubRequest);
```

Setting Custom Implemented Network Object

This API is required to perform the custom implementation of the network request call.

```
public void setApiRequest(AHAPIRequest apiRequest)
```

API Description:

API Name	Type	Description
apiRequest	AHAPIRequest	Custom Implementation of AHAPIRequest to perform sendRequest Network call.

Prerequisite for "Password" and "Device Signature" Encryption

To use the Android SDK for "Password" and "Device Signature" encryption, the following prerequisites must be met.

1. Change the tenant setting of **fieldsToBeEncryptedInAuthFlow** to ***"Password"*** and/or ***"Device Signature"***. The default value is **none,** which indicates that encryption is disabled by default.

```
[
  {
      "name": "fieldsToBeEncryptedInAuthFlow",
      "value": "password", "devicesignature"
  }
]
```

2. Integrate the **authcore.aar** SDK with the Android application. The SDK includes the APIs that can be called from the Android application context to achieve "Password" and "Device Signature" encryption.

   NOTE
   The **authcore.aar** SDK file can be found in the Support Portal > My Downloads along with other SDKs.

Initiate Authentication

To encrypt the "Password" and/or "Device Signature" in the authentication flow, call the Android SDK APIs from the application context in the following sequence.

Step 1: Initiate Authentication API

Call the **initAuth** function to initiate the authentication flow by invoking the POST `https://${SSP_FQDN}/<tenantName>/auth/v1/authenticate` API. Use this function at all existing places where you are invoking the */* **authenticate** API.

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/auth/v1/authenticate`
  should look similar to:
  `https://example.com/default/auth/v1/authenticate`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

API Signature

```
public void initAuth(String hostName,
                  String tenant,
                  JSONObject body,
                  Map<String, String> headers,
                  String[] fieldsToBeEncrypted,
                  VolleyCallback callback) throws AuthError
```

Sample Code

```
String[] fieldsToBeEncrypted = {AuthCoreConstants.PASSWORD_FIELD, AuthCoreConstants.DEVICESIGNATURE_FIELD};


AuthCore.getInstance(context).initAuth(Util.getServerHost(context, prefs), Util.getTenantName(context, prefs),
 body, headers, fieldsToBeEncrypted, new VolleyCallback() {
```

```
  @Override
  public void onSuccess(JSONObject response) throws JSONException {
      Log.d("", "Response from initAuth() " + response);
      callback.onSuccess(response);
  }

  @Override
  public void onError(String error) {
      Log.d("", "Got Error in initAuth() API " + error);
      callback.onError(error);
  }
})
```

The API also supports the **proxy URL** configuration for the AuthHub instance where the tenant name is not required as part of the URL. This function is used for initiating the authentication flow by invoking the POST `https://<proxyUrl>/auth/v1/authenticate` API. This function should be used at all the existing places where you are invoking the **/authenticate** API.

API Signature

```
public void initAuth(String serverUrl,
                  JSONObject body,
                  Map<String, String> headers,
                  String[] fieldsToBeEncrypted,
                  VolleyCallback callback) throws AuthError
```

Sample Code

```
String[] fieldsToBeEncrypted = {AuthCoreConstants.PASSWORD_FIELD, AuthCoreConstants.DEVICESIGNATURE_FIELD};

AuthCore.getInstance(context).initAuth(Util.getServerProxyUrl(context, prefs), body, headers,
 fieldsToBeEncrypted, new VolleyCallback() {
  @Override
  public void onSuccess(JSONObject response) throws JSONException {
      Log.d("", "Response from initAuth() " + response);
      callback.onSuccess(response);
  }

  @Override
  public void onError(String error) {
      Log.d("", "Got Error in initAuth() API " + error);
      callback.onError(error);
  }
})
```

Parameter Description

- **hostName:** Specifies the SSP (IDSP) server hostname.
- **tenant:** Specifies the tenant name. Example: **default.**
- **serverUrl:** Specifies the endpoint URL that can be a proxy URL for an IDSP instance. Data Type - **string.** Ensure that the proxy resolves an IDSP instance URL that points to the intended IDSP server host and tenant.
- **body:** Specifies the JSONObject of the **/authenticate** API request. An example of the request body is as follows:

```
{
  "subject": "{{userLoginid}}",
  "channel": "mobile",
  "ipAddress": "10.10.123.124",
  "action":"authenticate",
  "rememberMe" : true,
  "device": {
      "signature": {
          "iaAuthData":
 "_v02MyUgNGgYOi88OTk0emB7ZXV9GDQ2PDshOiY9bnUcOyEwOXUYNDZ1GgZ1DXVkZQpkYApjfHUUJSU5MAIwNx48IXpgZmJ7ZmN1fR4dARgZeXU5PD
NzclNDA6PzozOj06MDMyPDA9PzQ8KTw7ITAnOzQ5eDs0Njl4JTkgMjw7czMlJTxoczMlIS9oYHtgczMlOTtoMDt4EhcpKSlzMyU2JWhlZGRkZWRkZGRk
      }
  }
}
```

- **headers:** Specifies the headers to pass in the **/authenticate** API.
- **fieldsToBeEncrypted**: This is the list of fields of type String [] which must be encrypted. This list can contain constant values defined as AuthCoreConstants.PASSWORD_FIELD or AuthCoreConstants.DEVICESIGNATURE_FIELD or both fields. A developer can fetch this list dynamically by calling getFieldsToBeEncryptedInAuthFlow() on AuthCore instance.
- **onSuccess(JSONObject response):** Specifies the JSONObject success response that the **initAuth** function returns when the **/authenticate** API call succeeds.
- **onError(String error):** Specifies an error string that the **initAuth** function returns when the **/authenticate** API call fails.

Data Encryption

IDSP supports In-Transit and At-Rest "***Password***" encryption and In-Transit "***Device Signature***" encryption.

"Password" and "Device Signature" Encryption Confirmation API

Call the **isEncryptionRequiredForField** function to check if the "Password" and/or the "Device Signature" field is configured for encryption. The API returns a JSONObject with the value of the key **FIELDS_TO_BE_ENCRYPTED_IN_AUTH_FLOW** as **true** or **false.** The value, "**true"** is a confirmation that the "Password" and/or the "Device Signature" in the authentication flow should be encrypted.

API Signature for "Password" and "Device Signature"

```
public void isEncryptionRequiredForField(String hostName,
                                         String tenant,
                                         String field,
                                         Map<String, String> headers,
                                         VolleyCallback callback)
```

The value of the **field** input parameter must be "Password" and/or "Device Signature".

Sample Code for "Password"

```
AuthCore.getInstance(getApplicationContext()).isEncryptionRequiredForField(getApplicationContext(),
 serverHost, tenant, password, headers, new VolleyCallback() {
       @Override
       public void onSuccess(JSONObject response) throws JSONException {

 if(response.getBoolean(com.broadcom.layer7authentication.authcore.Util.FIELDS_TO_BE_ENCRYPTED_IN_AUTH_FLOW))
 {
              // Call encryptData() function to encrypt User Password
          }else{
```

```
                // Password Encryption feature is disabled for Teanant, call /PasswordAuthenticator API with
 user entered password
            }
        }
        @Override
        public void onError(String error) {
            // Any error while executing isPasswordEncryptionRequired() gets logged here
        }
    });
```

Sample Code for "Device Signature"

```
AuthCore.getInstance(getApplicationContext()).isEncryptionRequiredForField(getApplicationContext(),
 serverHost, tenant, "devicesignature", headers, new VolleyCallback() {
  @Override
  public void onSuccess(JSONObject response) throws JSONException {

 if(response.getBoolean(com.broadcom.layer7authentication.authcore.Util.FIELDS_TO_BE_ENCRYPTED_IN_AUTH_FLOW))
 {
          // Call respective encryption function
      }else{
          // Device Signature encryption is disabled for Tenant
      }
  }
  @Override
  public void onError(String error) {
      // Any other error gets logged here
  }
      });
```

Support for proxy URL configuration

The API also supports the **proxy URL** configuration for the IDSP instance where the tenant name is not required as part of the URL.

The following is an example for "Password" Encryption. You can use the same API signature for "Device Signature".

API Signature

```
public void isEncryptionRequiredForField(String serverUrl,
                                         String field,
                                         Map<String, String> headers,
                                         VolleyCallback callback)
```

The value of the **field** input parameter must be "Password" and/ or "Device Signature".

Sample Code: This example shows a "Password" encryption sample using a proxyURL. Similar code can be used for "Device Signature".

```
AuthCore.getInstance(getApplicationContext()).isEncryptionRequiredForField(getApplicationContext(), proxyUrl,
 password, headers, new VolleyCallback() {
       @Override
       public void onSuccess(JSONObject response) throws JSONException {

 if(response.getBoolean(com.broadcom.layer7authentication.authcore.Util.FIELDS_TO_BE_ENCRYPTED_IN_AUTH_FLOW))
 {
                // Call encryptData() function to encrypt User Password
```

```
            }else{
                // Password Encryption feature is disabled for Tenant, call /PasswordAuthenticator API with
  user entered password
            }
        }
        @Override
        public void onError(String error) {
            // Any error while executing isPasswordEncryptionRequired() gets logged here
        }
    });
```

Get the list of fields to be encrypted

To fetch the list of the fields which needs encryption dynamically, the developer must invoke the getFieldsToBeEncryptedInAuthFlow() API. This API is supported with both the options - one with server host and tenant details and another with proxy url details. On successful response from the server, the JSON object contains list of the fields which needs encryption. This API provides the dynamic way of getting a list of the fields to be encrypted form AH server tenant configuration.

```
public void getFieldsToBeEncryptedInAuthFlow(String hostName,
                                             String tenant,
                                             Map<String, String> headers,
                                             VolleyCallback callback)
```

On successful response, the JSONObject response contains AuthCoreConstants.FIELDS_TO_BE_ENCRYPTED field with value as an array of the fields which must be encrypted. The value of this array object can be passed to the initAuth() for handling in-transit encryption of password, devicesignature, etc.

Example:

```
AAuthCore.getInstance(context).getFieldsToBeEncryptedInAuthFlow(Util.getServerHost(context, prefs),
 Util.getTenantName(context, prefs), headers, new VolleyCallback() {
          @Override
          public void onSuccess(JSONObject response) throws JSONException {
              //
              try {
                  JSONArray arrJson = response.getJSONArray(AuthCoreConstants.FIELDS_TO_BE_ENCRYPTED);
                  String[] fieldsToBeEncrypted = new String[arrJson.length()];
                  for (int i = 0; i < arrJson.length(); i++)
                      fieldsToBeEncrypted[i] = arrJson.getString(i);
                  AuthCore.getInstance(context).initAuth(Util.getServerHost(context, prefs),
 Util.getTenantName(context, prefs), body, headers, fieldsToBeEncrypted, new VolleyCallback() {
                      @Override
                      public void onSuccess(JSONObject response) throws JSONException {
                          Log.d("", "Response from initAuth() " + response);
                          callback.onSuccess(response);
                      }
                      @Override
                      public void onError(String error) {
                          Log.d("", "Got Error in initAuth() API " + error);
                          callback.onError(error);
                      }
                  });
              } catch (AuthError authError) {
                  // TODO: Handle error scenario here
```

```
                authError.printStackTrace();
            }
        }
```

Support for Proxy URL configuration:

```
public void getFieldsToBeEncryptedInAuthFlow(String serverUrl,
                                        Map<String, String> headers,
                                         VolleyCallback callback)
```

On successful response, the JSONObject response contains AuthCoreConstants.FIELDS_TO_BE_ENCRYPTED field with value as an array of the fields which must be encrypted. The value of this array object can be passed to the initAuth() for handling in-transit encryption of password, devicesignature, etc.

Example:

```
AAuthCore.getInstance(context).getFieldsToBeEncryptedInAuthFlow(Util.getServerUrl(context, prefs), headers,
 new VolleyCallback() {
          @Override
          public void onSuccess(JSONObject response) throws JSONException {
              //
              try {
                  JSONArray arrJson = response.getJSONArray(AuthCoreConstants.FIELDS_TO_BE_ENCRYPTED);
                  String[] fieldsToBeEncrypted = new String[arrJson.length()];
                  for (int i = 0; i < arrJson.length(); i++)
                      fieldsToBeEncrypted[i] = arrJson.getString(i);
                  AuthCore.getInstance(context).initAuth(Util.getServerHost(context, prefs),
 Util.getTenantName(context, prefs), body, headers, fieldsToBeEncrypted, new VolleyCallback() {
                      @Override
                      public void onSuccess(JSONObject response) throws JSONException {
                          Log.d("", "Response from initAuth() " + response);
                          callback.onSuccess(response);
                      }
                      @Override
                      public void onError(String error) {
                          Log.d("", "Got Error in initAuth() API " + error);
                          callback.onError(error);
                      }
                  });
              } catch (AuthError authError) {
                  // TODO: Handle error scenario here
                  authError.printStackTrace();
              }
          }
```

Parameter Description

- **hostName:** Specifies the SSP (IDSP) server hostname.
- **tenant:** Specifies the tenant name. Example: **default.**
- **serverUrl:** Specifies the endpoint URL that can be a proxy URL for an IDSP instance. Data Type - **string.** Ensure that the proxy resolves an IDSP instance URL that points to the intended IDSP server host and tenant.
- **headers:** Specifies the headers to pass in the **/authenticate** API.
- **onSuccess(JSONObject response):** Specifies the JSONObject success response that contains AuthCoreConstants.FIELDS_TO_BE_ENCRYPTED field with value as an array of the fields which must be

encrypted. The value of this array object can be passed to the initAuth() for handling in-transit encryption of password, devicesignature, etc.

- **onError(String error):** Specifies an error string that the **initAuth** function returns when the **/authenticate** API call fails.

Encrypt "Password" and "Device Signature"

- ***Encrypt "Password"***

  Call the **encryptDataWithPublicKey** function to encrypt the plain "Password" with the tenant level public key without server nonce.

  API Signature

  ```
  public void encryptDataWithPublicKey(String plainText, VolleyCallback callback)
  ```

  Call the **encryptData** function to encrypt the plain "Password" with the tenant level public key with server nonce.

  API Signature

  ```
  public void encryptData(String plainText,VolleyCallback callback)
  ```

  The value of the **plainText** input parameter must be the "Password" string that is entered by the user.

  Sample Code

  ```
  AuthCore.getInstance(getApplicationContext()).encryptData(serverHost, tenant,
   ldapPassword.getText().toString(), new VolleyCallback() {
          @Override
          public void onSuccess(JSONObject response) throws JSONException {
              // Encrypted Password is returned from JSONObject with Key
   com.broadcom.layer7authentication.authcore.Util.CIPHER_TEXT and its value is CipherText Password which
   then can be sent for /PasswordAuthenticator API
          }
          @Override
          public void onError(String error) {
              // Password Encryption failed error will be logged here
          }
      });
  ```

- ***Encrypt "Device Signature"***

  To encrypt the "Device Signature" with symmetric key, call the **encryptWithSymmetricKey** function.

  API Signature

  ```
  public String encryptWithSymmetricKey(String plainText) throws Exception
  ```

  The value of the **plainText** input parameter must be the "Device Signature" string that is entered by the user.

  Sample Code

  ```
  AuthCore.getInstance(context).encryptWithSymmetricKey(deviceID);
  ```

Reference Link:

- Password Encryption
- Device Signature Encryption

FIDO Android SDK

The FIDO Android Software Development Kit (SDK) provides the APIs and other configuration details that are required to build your own mobile device application on Android for FIDO/FIDO Passkey registration, FIDO/FIDO Passkey authentication, and transaction signing.

This topic includes the following sections:

- Creating Object For FIDO Class
- Implementing IDSP Network Request Module (Optional)
- Using IDSP APIs
  - Hosting the "***assetlinks.json***" file for FIDO Authentication
  - FIDO Device Registration
  - FIDO Device Authentication
  - Handle On Activity Result
- Using FIDO Passkey
  - Hosting the "***assetlinks.json***" file for FIDO Passkey Authentication
  - FIDO Passkey Device Registration
  - FIDO Passkey Device Authentication
- Transaction Signing Using FIDO
- Error Codes and Messages

Creating Objects for FIDO class

In the client app, create the following object for the AuthFido class. This enables the client app to access SDK APIs which are already built in the AuthFido SDK.

```
public AuthFido(String serverHost, String tenantName, String fidoAuthenticationType)
```

or

```
public AuthFido(String serverUrl, String fidoAuthenticationType)
```

Parameter Description

- **`serverHost`** - Host name of the server where you have deployed the backend code. For example: sspHost191-example.com
- **`tenantName`** - Tenant name of the server.
- **`serverUrl`** - Endpoint URL that can be a proxy URL for an Symantec Identity Security Platform instance. Ensure that the proxy resolves IDSP instance URL that points to the intended IDSP server host and tenant.
- **`fidoAuthenticationType`** - This parameter can consist of either of the following value:
  - FIDO for Platform authenticators. For example, Fingerprint, Pin, Pattern
  - SECURITYKEY for Cross-Platform authenticators. For example, Yubikey, BLE, NFC etc.

Once the object creation is complete, pass the following xFlowState and bearerToken to the FIDO class object:

- **`authFido.XFlowState = <xflowState>`** - Upon invoking the "/auth/v1/authenticate" API with the tag name "flowState", the <xflowState> is obtained in the API response.
- **`authFido.bearerToken = <apiKey value>`** - Upon invoking the "/oauth2/v1/token" API with the tag name "access_token", the <apiKey value> is obtained in the API response.
- **`authFido.deviceTag = <deviceTag value>`** - deviceTag value is retrieved from IDSP Risk, default value can be an empty string.

IDSP Network Request Module (Optional)

If you intend to implement the IDSP Network Request Module, perform the following step.

Setting Custom Implemented Network Object

If you have implemented the Network Request Module, then you must call the following API.

This API is required to perform the custom implementation of the network request call.

```
public void setApiRequest(AHAPIRequest apiRequest)
```

API Description:

API Name	Type	Description
apiRequest	AHAPIRequest	Custom Implementation of AHAPIRequest to perform sendRequest Network call.

See Network Request Module for more information.

Using IDSP FIDO APIs

The FIDO functionality is considered a powerful and strong authentication method for securing sensitive information online from unauthorized access. Through the IDSP FIDO APIs, you can perform FIDO registration, FIDO authentication, FIDO Passkey registration, FIDO Passkey Authentication, and transaction signing using FIDO. There are many different forms of FIDO authenticators, some of which are embedded within the client device (such as fingerprint sensor) and some are connected to the client device through USB ports, NFC, or Bluetooth. IDSP FIDO SDK supports in-built fingerprint sensor, USB, NFC, and Bluetooth authenticators.

The APIs that are used to build your own FIDO/FIDO Passkey enabled mobile device application are described in the following sections.

Hosting the "assetlinks.json" file for FIDO Authentication

To establish the association between your website and Android apps while verifying the app's URL intents, you must publish a Digital Asset Links JSON file on your website. This JSON file utilizes the "package_name" field, which corresponds to the application ID declared in the app's build, to identify the associated apps.

Perform the following step to map the "assetlinks.json" file to the FIDO SDK.

Assign the following variables in the Sample Application before calling the registerFactor() API:

- "AuthFido.USE_CUSTOM_ASSET_LINK"
- "AuthFido.CUSTOM_RP_DOMAIN"

When you set the AuthFido.USE_CUSTOM_ASSET_LINK variable to true, (example-"AuthFido.USE_CUSTOM_ASSET_LINK = true"), then pass the value of the hostname in the "AuthFido.CUSTOM_RP_DOMAIN" variable, from where the assetlinks.json will be picked.

Example of hostname: `abc.example.com <domain name without a protocol>`

NOTE

- If you set the AuthFido.USE_CUSTOM_ASSET_LINK variable to false which is the default, (example-"AuthFido.USE_CUSTOM_ASSET_LINK = false"), then the assetlinks.json is picked automatically from the IDSP Hostname.
- To know more about FIDO Authentication in IDSP, see FIDO Authentication Flow.

FIDO Device Registration

This API method is used to register FIDO onto the device.

API Signature

```
public void registerFactor(Activity context, String userName, String displayName, String deviceName,
        Response.Listener<JSONObject> successListener,
        Response.ErrorListener errorListener)
```

Parameter Description

- UserName: Enter the user name with which you log in to the application.
- displayName: Enter the display name which must be displayed after you log in to the application.
- successListener<JSONObject>: This is the Volley new Response.Listener class Object.
- ErrorListener: This is the Volley new Response. ErrorListener class Object.

NOTE

- The SDK returns an error, FAILURE_FIDO_CREDENTIAL_NOT_FOUND: 909, when the passed FIDO credential is not present on the device. This might occur if a user has already registered a FIDO credential for one device (like Chrome on Windows) and tries to register a new credential for another device (like an Android phone or an iOS phone). See Implementing Registration of Multiple FIDO Credentials and FIDO Android error codes.

FIDO Device Authentication

This API method is used to Authenticate FIDO Credentials.

API Signature

```
public void authenticate (Activity context, String username,
                          Response.Listener successListener
                          Response.ErrorListener errorListener)
```

Parameter Description

- UserName: Enter the user name with which you log in to the application.
- successListener<JSONObject>: This is the Volley new Response.Listener class Object.
- ErrorListener: This is the Volley new Response. ErrorListener class Object.

Handle On Activity Result

This Callback API is used to retrieve control over the application after FIDO Authentication is successful.

Callback Method used:

```
public void handleOnActivityResult(Activity context, int requestCode,
                                   int resultCode, Intent data,
                                   Response.Listener<JSONObject>
successListener, Response.ErrorListener errorListener)
```

Using FIDO Passkey in Mobile Apps

The FIDO Passkey implementation leverages the existing IDSP FIDO SDK for mobile clients (iOS and Android). This implementation introduces new API methods specifically designed for the FIDO Passkey registration and authentication processes. It is important to note that the implementation of passkey is not interoperable with existing FIDO credentials. Therefore, Passkey authentication and FIDO authentication are mutually exclusive and once you choose to use Passkey as a factor to authenticate, you should register and authenticate ONLY using FIDO Passkey.

NOTE

- To know more about Passkey as an authenticator in IDSP, see Using Passkey.
- Transaction signing using a passkey credential is not supported in IDSP.

To enable passkey, you must first host the "***assetlink.json"*** file in your site's ***.well-known*** directory. The steps to host the file are described in the following section.

Hosting "*assetlink.json"* File to Sync Passkeys Between Web and Mobile Clients (Native Apps)

The "assetlink.json" file required for passkey synchronization between web and mobile clients is the same file hosted for FIDO Authentication. However, for passkey functionality, additional properties must be included in the file, and it must be hosted on the authenticating domain. This ensures that the passkey synchronization works effectively during the FIDO Passkey Authentication process.

To enable passkeys to operate correctly during passkey registration and authentication with IDSP FIDO Server, you are required to create an "***assetlink.json"*** file. This file lets your website declare an association with your Android apps.

- Create the "***assetlink.json"*** json file, deep link the android app with the authenticating domain. To know more about the creation of the "assetlink.json" file, see Android Documentation.
- The "assetlink.json" file should contain the "***delegate permission***" property to delegate permissions to the login credentials at "Web" and "Android" application level.
  Sample Content of the assetlink.json file: In the following example, the "delegate permission" property is added to the assetlink.json file.

```
[ {
 "relation" : [ "delegate_permission/common.get_login_creds" ],
 "target" : {
   "namespace" : "web",
   "site" : "https://mydoamin.example.com"
 }
}, {
 "relation" : [ "delegate_permission/common.handle_all_urls", "delegate_permission/
common.get_login_creds" ],
 "target" : {
   "namespace" : "android_app",
   "package_name" : "com.entersekt.fido2",
   "sha256_cert_fingerprints" :
 [ "C5:8B:E3:9B:36:B3:67:12:D7:0C:DA:C5:9D:65:2A:FC:43:9B:AE:1B:76:C9:7D:A1:7E:69:2B:7A:15:AB:27:96" ]
 }
} ]
```

- Host the file in the following location:

```
"https://<Authenticating_Domain>/.well-known/assetlinks.json
```

FIDO Passkey Device Registration

This API method is used to register FIDO Passkey onto the device.

API Signature

```
public void registerFactor(Activity context,
                           String userName,
                           String displayName,
                           String deviceName,
                           boolean isPasskey,
                           String domainName,
                           Response.ListenerJSONObject successListener,
                           Response.ErrorListener errorListener)
```

Parameter Description

- UserName: Enter the user name with which you log in to the application.
- displayName: Enter the display name which must be displayed after you log in to the application.
- deviceName: Enter the display name which must be displayed after you log in to the application.
- isPasskey: This is a boolean value. When set to "true," it indicates that users are required to register the device using a passkey, and FIDO registration cannot be utilized.
- successListener<JSONObject>: This is the Volley new Response.Listener class Object.
- ErrorListener: This is the Volley new Response. ErrorListener class Object.

NOTE

- The SDK returns an error, FAILURE_FIDO_CREDENTIAL_NOT_FOUND: 909, when the passed FIDO/FIDO Passkey credential is not present on the device. This might occur if a user has already registered a FIDO/ FIDO Passkey credential for one device (like Chrome on Windows) and tries to register a new credential for another device (like an Android phone or an iOS phone). See Implementing Registration of Multiple FIDO Credentials and FIDO Android error codes.
- Transaction signing using a passkey credential is not supported in IDSP.

FIDO Passkey Device Authentication

This API method is used to Authenticate FIDO Passkey Credentials.

API Signature

```
public void authenticate(Activity context,
                         String username,
                         boolean isPasskey,
                         String domainName,
                         Response.Listener successListener,
                         Response.ErrorListener errorListener)
```

Parameter Description

- UserName: Enter the user name with which you log in to the application.
- isPasskey: This is a boolean value. When set to "true," it indicates that users are required to authenticate the device using a passkey, and FIDO authentication cannot be utilized.
- successListener<JSONObject>: This is the Volley new Response.Listener class Object.
- ErrorListener: This is the Volley new Response. ErrorListener class Object.

Transaction Signing using FIDO

IDSP protects authenticated user session attacks (for example, MitB attacks) by supporting FIDO-based transaction data signing. The Transaction Signing Android SDK verifies and signs the transaction challenge request that the Relying Party initiates for the sensitive user transactions.

Transaction Signing Flow:

1. When a user initiates a sensitive transaction on a mobile device, the mobile application invokes the business API of the Relying Party.
2. The Relying Party initiates the transaction signing flow by invoking the **FIDOTransactionSigningChallengeGenerator** API of the IDSP server for a challenge. For information about the server API, see Transaction Signing.
3. The IDSP server generates and returns the challenge (SVR_CHL) to the Relying Party.
4. On receiving the transaction data and challenge details from the Relying Party by the calling API, the mobile application prompts the user to confirm the transaction data (TXN_DATA). For example, "Do you want to transfer $1000 to John?". If the user denies the transaction, the transaction signing flow ends with a proper error message that is displayed to the user.

5. If the user approves the transaction, the mobile application invokes the FIDO SDK **signTransaction** API.

   API Signature

   ```
   public void signTransaction(Activity context,
                               AssertionOptionsResponse optionsFromServer,
                               String transactionData)
   ```

   Parameter Description

   - context: Specifies the activity context of an Android application.
   - optionsFromServer: Represents the challenge response that the IDSP server returns in response to the **FIDOTransactionSigningChallengeGenerator** API.
   - transactionData: Specifies a string that the Relying Party denotes as the transaction data.

   Sample Code

   ```
   // Mobile application converts the server response (challenge) received from the Relying Party into the
    AssertionOptionsResponse object.
   Gson gson = new Gson();
   AssertionOptionsResponse challResponse = null;
   challResponse = gson.fromJson(response.toString(), AssertionOptionsResponse.class);
   String base64TransactionData = Base64.encodeToString(transactionData.getBytes(StandardCharsets.UTF_8),
    Base64.NO_WRAP);
   authFido.signTransaction(context, challResponse, base64TransactionData);
   ```

6. The **signTransaction** API generates a new transaction challenge (TXN_CHL) by concatenating transaction data, server challenge (passed in the assertionOptionsResponse attribute), and tilde '~'.

   TXN_CHL = SVR_CHL + '~' + TXN_DATA

7. The SDK challenges the user for Biometrics (TouchID) or Security Key depending on the authentication factor that is configured on the mobile device. Once the identity of the user is verified, the SDK unlocks the private key that is stored on the mobile device and uses it to sign the transaction challenge (TXN_CHL).
8. After successful FIDO authentication, the Android Operating System calls the **onActivityResult(int requestCode, int resultCode, @Nullable Intent data)** API to pass control to the mobile application. The mobile application in turn calls the **handleOnActivityResult()** SDK API to receive the assertion data.

   API Signature

   ```
   public AssertionVerifyRequestWithTransactionData handleOnActivityResult(Activity context, int requestCode,
    int resultCode, Intent data, String transactionData)
   ```

   Parameter Description

   - context: Specifies the activity context of an Android application.
   - requestCode: Identifies FIDO authentication request.
   - ResultCode: Indicates whether the FIDO authentication succeeded or failed.
   - data: Contains the signed signature or an error response.
   - transactionData: Specifies a string that the Relying Party denotes as the transaction data.

   Sample Code

   ```
   @Override
     protected void onActivityResult( int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(resultCode == 0) {
           finish();
        } else{
            String base64TransactionData =
    Base64.encodeToString(transactionData.getBytes(StandardCharsets.UTF_8), Base64.NO_WRAP);
            AssertionVerifyRequestWithTransactionData assertVerifyRequest =
    authFido.handleOnActivityResult(context, requestCode, resultCode, data, base64TransactionData);
   ```

```
        }
    }
```

9. The mobile application returns assertion (assertVerifyRequest) data to the Relying Party.
10. The Relying Party invokes the **FIDOTransactionSigningChallengeVerifier** API of the IDSP server. The API signature includes the transaction data and the signed transaction challenge for verification. For information about the server API, see Transaction Signing.
11. The Challenge Verifier API returns the validation status to the Relying Party by invoking its business API.
12. Depending on the validation status, the mobile application displays the transaction summary or an error message to the user.

Error Code and Message

The following error code is returned by the FIDO Android SDK.

Error Message	Error Code
FAILURE_FIDO_CREDENTIAL_NOT_FOUND	909

WebAuthN SDK for Android

The WebAuthN SDK can be used by customers who would not like to support hosting of the assetlinks.json file, have restrictions on using Google Play services, and do not want any dependency on Android SDK APIs for FIDO authentication.

Currently, Symantec Identity Security Platform FIDO SDK makes use of native Android SDK FIDO APIs which are dependent on Google Play services. This implementation also mandates the hosting of assetlinks.json file. To authenticate using FIDO, our customer needs to host the assetlinks.json file and mobile application needs details of the hosted domain. To remove dependency on the Google Play service and mandatory hosting of assetlinks.json file, we can make use of the third party open source WebAuthN SDK which supports FIDO functionality. This WebAuthN SDK supports FIDO functionality as a independent library and does not depend on any of the Google's components on the Android platform. But being a third party library, it has its own maintenance.

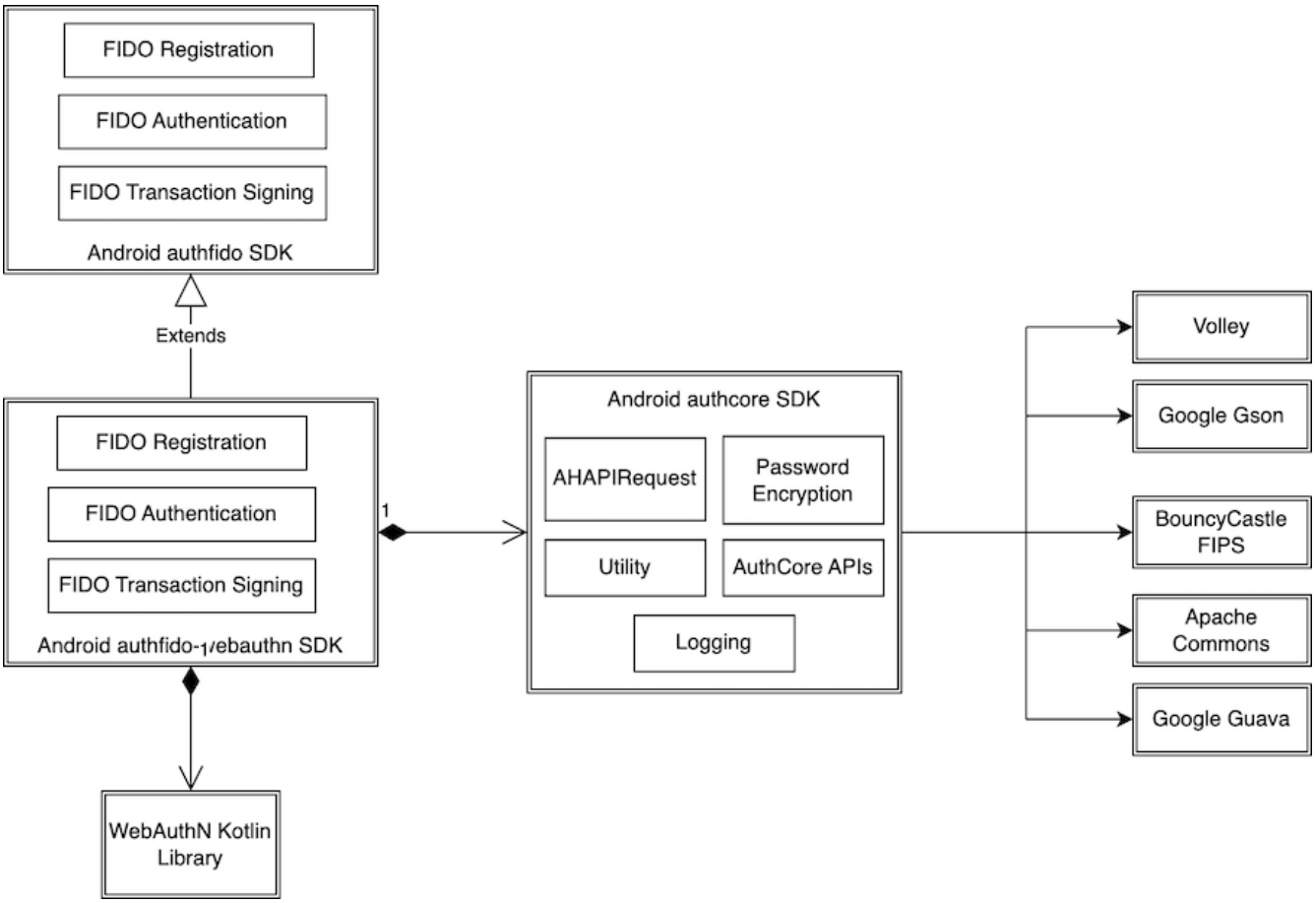
Table 95: Usage of IDSP SDK

IDSP SDK	Usage
auth-fido	IDSP Android FIDO SDK can be used where customer can make use of Google Play services and host assetlinks.json file. Google provides integrity checks for the application consuming FIDO APIs by using Deep Linking mechanism (hosting of assetlinks.json file). It provides secure authentication mechanism on the Android device. It is recommended to use this form of SDK whenever possible.
auth-webautnfido	The authfido-webauthn module can be used by the customers who cannot support hosting of the assetlinks.json and have restriction to use Google Play service (countries like China have restrictions to use Google Play services on the Android phones etc.). It is also provided to the customers who do not want any dependency on Android SDK APIs for FIDO authentication.

Integration of WebAuthN SDK with IDSP FIDO SDK

The existing IDSP Android SDK takes care of all the flow level complexity in executing the IDSP FIDO flows on native Android platform. It already exposes a set of APIs which are used by Android Application developer for FIDO credential registration and authentication. To support WebAuthN FIDO registration/authentication, we need to integrate the WebAuthN library with IDSP Android SDK to have consistent developer experience.

To support consistent developer experience for FIDO authentication flows using WebAuthN FIDO, we have extended IDSP Android SDK and created authfido-webauthn SDK module.



The WebAuthN FIDO SDK supports same set of APIs which are supported by IDSP authfido SDK. Through the webauthn-authfido SDK APIs, you can perform FIDO registration, FIDO authentication, and transaction signing using FIDO, based on WebAuthN library which is not dependent on Google Play service. The following APIs are overriden by the authfido-webauthn module which makes use of WebAuthN library to provide IDSP FIDO auth functionality.

Device Registration

This API method is used to register FIDO onto the device.

API Signature

```
public void registerFactor(Activity context, String userName, String displayName, String deviceName,
  Response.Listener<JSONObject> successListener, Response.ErrorListener errorListener)
```

API parameter descriptions

Table 96: Device Signature API Parameter Description

Name	Type	Description
context	Activity	Specifies the activity context of an Android application
userName	String	Enter the user name with which you want to log in to the application

Name	Type	Description
displayName	String	Enter the display name which must be displayed after you log in to the application.
successListener<JSONObject>	Response.Listener object	This is the Volley new Response.Listener class object which is invoked when success response is received.
ErrorListener	Response.ErrorListener	This is the Volley new Response.ErrorListener class object which is invoked when error response is received.

Device Authentication

This API method is used to authenticate FIDO credentials.

API Signature

```
public void authenticate(Activity context, String username, Response.Listener successListener
  Response.ErrorListener errorListener)
```

API parameter descriptions

Table 97: Device Signature API Parameter Description

Name	Type	Description
context	Activity	Specifies the activity context of an Android application
userName	String	Enter the user name with which you want to log in to the application
successListener<JSONObject>	Response.Listener object	This is the Volley new Response.Listener class object which is invoked when success response is received.
ErrorListener	Response.ErrorListener	This is the Volley new Response.ErrorListener class object which is invoked when error response is received.

Handle on Activity Result

This callback API is used to retrieve control over the application after FIDO Authentication is successful. The parameter values received in OnActivityResult() API should be passed to the respective parameters of the handleOnActivityResult() API.

```
publicvoid handleOnActivityResult(Activity context, int requestCode, int resultCode, Intent data,
  Response.Listener<JSONObject> successListener, Response.ErrorListener errorListener)
```

Transaction Signing using FIDO (signTransaction API)

This API method is used to digitally sign or authenticate a transaction.

API Signature

```
public void signTransaction(Activity context, AssertionOptionsResponse optionsFromServer, String
  transactionData)
```

API parameter descriptions

Table 98: Device Signature API Parameter Description

Name	Type	Description
context	Activity	Specifies the activity context of an Android application
optionsFromServer	AssertionOptionsResponse	Represents the challenge response that the IDSP server returns in response to the FIDOTransactionSigningChallengeGenerator API.
transactionData	String	Specifies a string that the Relying Party denotes as the transaction data.
successListener<JSONObject>	Response.Listener object	This is the Volley new Response.Listener class object which is invoked when success response is received.
ErrorListener	Response.ErrorListener	This is the Volley new Response.ErrorListener class object which is invoked when error response is received.

Symantec Identity Security Platform Web SDK

Symantec Identity Security Platform Web sdks include Transaction Signing SDK and Password Encryption SDK.

This topic includes the following:

- Transaction Signing Web SDK
- Password Encryption Web SDK
- Device Signature Web SDK

SDK for Transaction Signing (Web)

Learn how to use the JavaScript SDK for Transaction Signing.

Purpose of the SDK

This JavaScript SDK signs transactions in the Relying Party-initiated FIDO authentication web flows.

Prerequisite

To use the SDK for signing transactions, include the IDSP-WEBSDK.js SDK script file in the HTML code of your webpage to make it available as a namespaced object, **AHUI**.

Example:

```
<!DOCTYPE html>
<html>
<head>
    . . .
    . . .
    <script src="IDSP-WEBSDK.js"></script>
</head>
<body>
    <!-- body content goes here -->
</body>
</html>
```

NOTE
The IDSP-WEBSDK.js SDK script file is available for download from the Support Portal > My Downloads along with other SDKs.

Integrating the SDK

The **AHUI** object contains a JavaScript function named **signTransaction** for signing Relying Party-initiated FIDO authentication transactions. The **AHUI.signTransaction** function has the following syntax:

```
AHUI.signTransaction(params, successCallback, errorCallback);
```

NOTE
If you are using TypeScript, then you may need to use window["AHUI"] for accessing the AHUI object.

The following process describes how the SDK works:

1. Relying Party initiates a transaction signing flow by sending a POST request to the `https://${SSP_FQDN}/{{tenantName}}/auth/v1/FIDOTransactionSigningChallengeGenerator` endpoint.

   NOTE

   - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
   - The default value for `<tenantName>` is "default".
   - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
     For example:
     `https://{{host}}/{{tenantName}}/auth/v1/FIDOTransactionSigningChallengeGenerator`
     should look similar to:
     `https://example.com/default/auth/v1/FIDOTransactionSigningChallengeGenerator`
   - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

2. The response from the endpoint is passed to the SDK along with the transaction data in the following format.

```
AHUI.signTransaction({
  transactionData: "<transactionData of type String>" /* See (1) Below */,
  challengeData: Object /* See (2) Below */
}, function successCallback(signedChallengeData) {
   console.log("Signed Challenge Data:", signedChallengeData);
   /*
   Pass the signedChallengeData and transactionData to Relying Party
   so that it can invoke factor/v1/FIDOTransactionSigningChallengeVerifier
   */
}, function errorCallback(err) {
     console.log(err);
});

/*
(1): The transactionData that will be used for signing purpose.
(2): The response obtained by invoking factor/v1/FIDOTransactionSigningChallengeGenerator from the Relying
 Party.
*/
```

As explained in the above sample, the **signTransaction** function must be passed in **3 arguments**:

1. params object: This object contains all the necessary information for signing transactions:

   - **transactionData**: Defines a string that is used by FIDO for signing transactions.
   - **challengeData**: Indicates the challenge that is obtained from a FIDO challenge generator.
2. successCallback: Indicates the action to be done when the call is successful. The callback is passed with the signed challenge.
3. errorCallback: Indicates the action to be done when an error occurs. The callback is passed with an error.

3. The SDK signs the challenge and passes the signed challenge in the successCallback function as a parameter.
4. The business application passes the transaction data and the signed challenge to the Relying Party.
5. Relying Party invokes the challenge verifier by sending a POST request to the `https://${SSP_FQDN}/{{tenantName}}/auth/v1/FIDOTransactionSigningChallengeVerifier` endpoint with the transaction data and the signed challenge in the payload.
6. The challenge verifier validates the transaction data and the signed challenge, and returns HTTP Status 200 upon a successful verification.

Reference Link:

- Transaction Signing

SDK for Password Encryption (Web)

The Web Software Development Kit (SDK) for Password Encryption facilitates password security by supporting end-to-end encryption of passwords in the authentication flow.

This topic includes the following sections:

Purpose of the SDK

This JavaScript SDK is intended to perform encryption-related operations during the authentication flow. Currently, it supports the encryption of the “Password” field.

Prerequisites

To enable encryption of a password, perform these steps:

1. Configure the “**fieldsToBeEncryptedInAuthFlow**” tenant-level setting to “**password**”. The default value of this setting is “none” which indicates that encryption is disabled.

```
[
    {
        "name": "fieldsToBeEncryptedInAuthFlow",
        "value": "password"
    }
]
```

2. Add the Encryption Provider by invoking a POST call to `https://${SSP_FQDN}/{{Tenant}}/admin/v1/EncryptionProviders`.

   NOTE

   - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
   - The default value for `<tenantName>` is "default".
   - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.

     For example:

     `https://{{host}}/{{tenantName}}/admin/v1/EncryptionProviders`

should look similar to:

`https://example.com/default/admin/v1/EncryptionProviders`

- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

The following is a sample showing how to add an encryption provider:

```
{
        "providerName": "aws-kms-encryption-provider",
        "providerProperties": [
            {
                "name": "cmkAliasName",
                "value": "arn:aws:kms:us-west-1:688201620683:alias/e2e-pwd-encryption-cmk"
            },
            {
                "name": "type",
                "value": "aws-kms"
            }
        ]
    }
```

3. Include the SDK script file (IDSP-WEBSDK.js) in your webpage's HTML code.
   Example:

```
<!DOCTYPE html>
<html>
<head>
    . . .
    . . .
    <script src="IDSP-WEBSDK.js"></script>
</head>
<body>
    <!-- body content goes here -->
</body>
</html>
```

This SDK file contains the following:

- JavaScript Cryptography algorithms that are related to RSA and AES
- IDSP specific logic for initiating the authentication and encrypting the password.

> **NOTE**
> The SDK file (IDSP-WEBSDK.js) is available for download from the Support Portal > My Downloads along with other SDKs.

Integrating the SDK

After the above prerequisite steps are met, the SDK will be available in the webpage's context as a namespaced object - **AHUI**. This **AHUI** object contains the JavaScript functions for performing the steps necessary for initiating the authentication flow with encryption setup and encrypting the password field payload.

> **NOTE**
> If you are using TypeScript, then you may need to use window["AHUI"] for accessing the AHUI object.

There are two functions included with the SDK: **initAuth** and **encrypt**. These functions have been explained in the following sections.

AHUI.initAuth function

This function is used for initiating the authentication flow by invoking the POST `https://${SSP_FQDN}/{{tenantName}}/auth/v1/authenticate` API. This function should be used at all the existing places where you are invoking the **/authenticate** API.

```
AHUI.initAuth({
        sspServiceURL: "https://${SSP_FQDN}/<tenant>/",
        authToken: "Bearer . . . .",
        trustedDeviceID: "<Trusted Device ID if available - OPTIONAL>",
        fidoID: "<FIDO Identifier if available - OPTIONAL>",
        withCredentials: false, /* OPTIONAL, default value is true if unspecified */
        payload: {
            param1: value1,
            param2: value2,
            /* additional parameters */
        },
        headers: {
            header1: headervalue1,
            header2: headervalue2,
            /* additional headers */
        }
    },
        function successCallback(respBody, respHeaders) { },
        function errorCallback(err) { }
    );
```

As explained in the sample above, the **initAuth** function should be passed in **3 arguments**:

1. config object: This object contains all the necessary information for initiating the authentication flow:
   - **sspServiceURL**: The SSP (Symantec Identity Security Platform) URL.
   - **authToken**: The authentication Bearer token.
   - **trustedDeviceID**: [OPTIONAL] If trustedDeviceID is available, then it should be passed here.
   - **fidoID**: [OPTIONAL] If the FIDO Identifier is available, then it should be passed here.
   - **withCredentials**: [OPTIONAL] Indicate whether the AJAX "withCredentials" mode is true or false for Cross Domain Requests. Default is true if not specified.
   - **payload**: The POST payload for the /authenticate API call.
   - **headers**: The headers for /authenticate API call. A map of key-value pairs.
2. successCallback: The action to be done when the /authenticate call is successful. The callback is passed the response body as well as the response headers.
3. errorCallback: The action is to be done when an error occurs. The callback is passed the error.

An example of **AHUI.initAuth** usage

```
AHUI.initAuth({
    "sspServiceURL": "https://auth.ssp.com/default/",
    "authToken": "Bearer eyJ4NXQjUzI1NiI6IlA3.....BJQ-vomRf913Aa1NZDQDDivcpLSziJ48bY-AZp46yUQ",
    "trustedDeviceID": "jn9Zz9gA3s52RumLSxRhu.....yDGa5w5mbFgFUC/2faQvnkzV/NvL",
    "fidoID": "lI6jotihH.....YPAp9iFo6omQCpJ8NOSuHud9Ah5",
    "payload": {
        "subject": "nbruce",
        "channel": "web",
        "device": {
            "signature": {
```

```
                    "iaAuthData":
"_v02MyUgNGgYOi88OTk.....Z2BzMyUhMmhtNDRjNGJnNmdlbTBhY20zbDEzZTZgMzFhZGZmM2JmMA=="
                }
            },
            "action": "authenticate",
            "rememberMe": true
        },
        "headers": {
            "x-flow-state": "R3lzV0JVNXZXM1ByNWJtd.....2Y5ejMrV0J0bFVOc2JidlJPR2lEVjRVUUhuUlhjdz09",
            "x-device-tag": "_v02MyUgNGgYOi88OTk0e.....htNDRjNGJnNmdlbTBhY20zbDEzZTZgMzFhZGZmM2JmMA=="
        }
    }, function successCallback(respBody, respHeaders) {
        console.log("initAuth is successful.");
        console.log("Response Body:", respBody);
        console.log("Response Headers:", respHeaders);
    }, function errorCallback(err) {
        console.error("There was an error: ", err);
    });
```

The above example describes the possible values that can be passed to the config object.

> **NOTE**
> The existing code invoking the /authenticate call **needs to be removed after integrating the AHUI.initAuth function**, since initAuth will invoke the /authenticate API.

AHUI.encrypt function

The **AHUI.encrypt** function can be used to encrypt the password payload while invoking the POST `https://${SSP_FQDN}/{{tenantName}}/factor/v1/PasswordAuthenticator` API. This function requires the password value that must be encrypted as an argument.

```
AHUI.encrypt(passwordString);
```

An example of **AHUI.encrypt** usage

```
// Previous password authentication code:
doPasswordAuth({
     password: "mysecretpassword"
});

// Password authentication with encryption:
doPasswordAuth({
   password: AHUI.encrypt("mysecretpassword")
});
```

In the above example, assume that the doPasswordAuth is a function that invokes the /PasswordAuthenticator API. Here, you must encrypt the password using the **AHUI.encrypt(. . . .)** function, before passing it to the /PasswordAuthenticator API.

> **NOTE**
> The **AHUI.encrypt** function skips the encryption logic and return the original password value if **fieldsToBeEncryptedInAuthFlow** tenant-level setting is set to “none”.

AHUI.encryptDataWithPublicKey function

The **AHUI.encryptDataWithPublicKey** function can be used to encrypt a data string with the public key of the IDSP. This function does not salt the data with the nonce before encrypting the data. This function requires the data string that must be encrypted as an argument.

```
AHUI.encryptDataWithPublicKey(dataString);
```

NOTE
The **AHUI.encryptDataWithPublicKey** function skips the encryption logic and throws an error in the following scenarios:

- The **fieldsToBeEncryptedInAuthFlow** tenant setting is set to **none.**
- The **encryptDataWithPublicKey** function is invoked before invoking the **initAuth** function.

Reference Links to:

- Password Encryption

SDK for Device Signature Encryption

The Web Software Development Kit (SDK) for Device Signature Encryption provides a secure means of uniquely identifying and verifying devices for authentication purposes.

This section includes the following sub-sections:

- Purpose
- Prerequisites
- Integrating the SDK
  - AHUI.encryptWithSymmetricKey Function

Purpose of the SDK

This JavaScript SDK is intended to perform encryption-related operations during the authentication flow. Currently, it supports the encryption of the “Device Signature” and "X-Device-Tag" fields.

Prerequisites

To enable encryption of Device Signature, perform these steps:

1. Configure the “**fieldsToBeEncryptedInAuthFlow**” tenant-level setting to “**devicesignature**”. The default value of this setting is “none” which indicates that encryption is disabled.

```
[
    {
        "name": "fieldsToBeEncryptedInAuthFlow",
        "value": "devicesignature"
    }
]
```

2. Include the SDK script file (IDSP-WEBSDK.js) in your webpage’s HTML code.
   Example:

```
<!DOCTYPE html>
<html>
<head>
    . . .
    . . .
    <script src="IDSP-WEBSDK.js"></script>
</head>
```

```
<body>
    <!-- body content goes here -->
</body>
</html>
```

This SDK file contains the following:

- JavaScript Cryptography algorithms that are related to RSA and AES
- IDSP specific logic for initiating the authentication and encrypting the device signature.

> **NOTE**
> The SDK file (IDSP-WEBSDK.js) is available for download from the Support Portal > My Downloads along with other SDKs.

Integrating the SDK

After the above prerequisite steps are met, the SDK will be available in the webpage's context as a namespaced object - **AHUI**. This **AHUI** object contains the JavaScript functions to perform the steps necessary for initiating the authentication flow with encryption setup and encrypting the device signature field.

> **NOTE**
> If you are using TypeScript, then you must use window["AHUI"] for accessing the AHUI object.

The ***AHUI.encryptWithSymmetricKey*** function is added to the Symantec Identity Security Platform Web SDK to perform encryption on any sensitiveData using a symmetricKey.This function has been explained in the following section.

AHUI.encryptWithSymmetricKey function

```
AHUI.encryptWithSymmetricKey(symmetricKey, sensitiveData);
```

The **AHUI.encryptWithSymmetricKey** function accepts two arguments:

- ***symmetricKey***: This is a cryptographic key, specifically a symmetric key. Symmetric keys are used in symmetric encryption algorithms, where the same key is used for both encryption and decryption. In this context, `symmetricKey` is the key that is used to encrypt the data.
- ***sensitiveData***: This is the data that you want to encrypt, and it is referred to as "sensitive" because it contains information that must be kept confidential.

The function encryptWithSymmetricKey performs encryption on the sensitiveData using the provided symmetricKey.

Reference Links to:

- Encryption of Device Signature

VIP SDKs

Learn about VIP SDKs.

The SDKs referred in this topic are from VIP documentation. Refer VIP SDKs for latest information.

This module includes the following VIP SDK documentation:

- SDK for Risk Android
- SDK for Risk iOS
- SDK for Risk Web
- SDK for Mobile OTP Android
- SDK for Mobile OTP iOS
- SDK for Password Encryption

SDK for Risk Android

The Symantec Identity Security Platform SDK for Android provides the sample code and describes the APIs needed to build your own mobile device application for the Android platform. The APIs perform the following functions:

- Returns the device fingerprint data and device credential.
- Stores the provisioned device credential in the mobile storage.

This topic includes the following:

SDK Location

The IDSPRISK Android SDK is available for download from the Support Portal > My Downloads.

Prerequisite

Before working with the SDK documentation, ensure to obtain the VIP certificate by performing the steps given here.

Sample Code

The sample code illustrates possible ways to write your Android application using the APIs. The sample code shows the functions your application must perform using the IDSP APIs to create and manage your credential object.

The following sample code initiates the evaluate risk flow, stores the credential tag or device tag that is received from the server into the local repository, and sends the challenge to the server to validate the device along with the user.

```
String IAData = mvipIAHandler.getIAData(context, null);

String deviceTag = <value of key "device.tag" in UserRiskScoreEvaluator REST API response>

boolean status = mvipIAHandler.setIAData(context, deviceTag);
```

This sample code performs the following functions:

1. Creates the MVIPIntelligentAuth object.
2. Calls the getIAData IDSP API to retrieve fingerprint data of the mobile device.
3. Sends fingerprint data to IDSP and initiates a risk-based authentication flow.
4. Stores the device credential that is received in the previous step into mobile secure storage.
5. Calls the getIAData IDSP API to retrieve a fresh set of fingerprint data of the mobile device.
6. Completes the risk-based authentication flow and on success, registers the device to VIP Services.

IDSP APIs

MVIPIntelligentAuth class

Provides a functional interface to retrieve and store the fingerprint data of the mobile device.

Tasks

- ***getIAData*** - Retrieve fingerprint data of the mobile device.
- ***setIAData*** - Store the fingerprint of the mobile device in the mobile local storage.
- ***getVersion*** - Retrieve the version number of the IDSP library.

Class methods

- ***getIAData:***
  - Retrieves fingerprint data of the mobile device.
  - String getIAData(Context context, String code) throws MVIPException.
- ***setIAData:***

  - Stores fingerprint of the mobile device in the mobile local storage.
  - boolean setIAData (Context context, String deviceTag);
- ***getVersion:***
  - Returns the version of the IDSP library.
  - String getVersion()

Error Codes

The following table lists the error codes that you may encounter while using the IDSP Android APIs:

Error Code	Error Message
1001 - 1005	Internal Error
2001 - 2011	Certificate Error
3001 - 3006	I/O Error
4001 - 4012	Crypto Error
5001 - 5007	Network Error

Sample Fingerprint of Mobile

Following is the sample fingerprint of mobile IAdata with Encrypted and Decrypted value:

IAdata Encrypted:


```
_v02MyUxIWhnczgzOiZoFDsxJzo8MXM4MzojaGJ7ZHtkczgzIjRoZWdvZWVvZWVvZWVvZWVvZWVzODMmNmhkZG1hKWJnZXMzJTk7aDA7CgAGczgzMjZoZGJ
+MCwmczgzNAoxI2g7PDY
+OTQgJgozczgzNAoxJWgbGBRnY3thZ3hkY2xzODM0CjgzaDg6ITonOjk0czgzNAo4MWgYOiE6dRB1fWF8dQU5ICZzODM0CiE0MiZoJzA5MDQmMHg
+MCwmczgzNAo3OWggOz47OiI7czgzNAo2NGdoNCc4MDQ3PHM4MzQKPSJoOCFjYmZgczgzNAoxPDFoZmBkbWxhZWxlYmJtYWxlczgzNAomJjtobWxsZG1gYW
+BSETASMeBwU9O2U0Zw8NJAF6Ag1kfgAEBn5iMH4MFBYsYBITZxc/ES0fNmcGOxptF3o9bT8/ITQdATE
+OAEDMDdhPzoDBjIgPDkTAzgAEAMPIRACAzgDGWwHIxZtDT1iPi85HhoYJxY4LQMNZGAHLD8sej0QZiI
+PzICOCwXBxwRZ2M5GRNsMDMFNh8mAzA+AmUAZCxiAH5kIWYMMBBsDBsDHgw8NGYXBi8RZi0nHSwnHjQnEDk/
AGFlFCQXYy0xDGAFbRBlNmVmAxE2NDYaBhtsPRc/
Ni9jOWdhMywDBRwPGmQ3NBsaFgEWJx8sGhtkPxkfHgINJBE0BW0WbRIMAxYzIGcMFiQmIxkTMzcjP2VhPRolORgcegMZIGUNeiIzO2MRMzQ5JhI5MRMHISA
```


IAdata Decrypted:


```
fpdt=2&mfos=Android&mfov=7.1.1&mfwa=02:00:00:00:00:00&mfsc=1184|720&fpln=en_US&mfgc=17.518705|
78.31355&mfpv=2&fpts=1591685907667&mfappid=com.verisign.mvip.iaapp.activity&mfa_isrooted=false&mfa_id=12fe289ea3501f9b&
v7a&mfa_fp=motorola/nicklaus_f/nicklaus_f:7.1.1/NMA26.42-169/300:user/release-
keys&mfa_dv=nicklaus_f&mfa_dp=NMA26.42-169&mfa_mf=motorola&mfa_md=Moto E (4) Plus&mfa_tags=release-
keys&mfa_bl=unknown&mfa_ca2=armeabi&mfa_hw=mt6735&mfa_did=351894090778490&mfa_ssn=8991854040028344073 9&mfa_sid=40585407
 4G&mfa_so=405854&mfa_cgc=2065684|115&mfec=J2iGw5OztbxiZI5ViHeXmmUhlKKW8KgSV9ejgdHwX9/ctiytJrip/
cDVYxRlNfguctEJh8b1JQlYK02QPol/qwfmaA9Pp6GoFF5Q1lcwYSyYgR4HiwAFqEoyeEgCIiHMnA0XtK7Rb3RXUuW/
JBvr1Qnu31vtZ4hZe0tONEZmrPqXMv7FwUeBgJ0x9VwHskNI8r9NsF64IB8Q59rd9G27Xvj4KfmWAfmWvG0GnX3z31ECn58C8gBaIlZ2Z2p2Tv9FJOKG3Lq
+Py6B+ktglkQ==
```


Table displays the attribute details used in the Sample fingerprint:

Attribute Name	Example Value	Attribute Information
fpdt	2	DeviceType(Android device = 2)
mfos	Android	OS (Android OS = android)

mfov	7.1.1	Device version
mfwa	02:00:00:00:00:00	Mac address through wifi Manager
mfsc	1184|720	Mobile Display
fpln	en_US	Locale information
mfgc	17.518705|78.31355	GPRS provides latitude and longitude values
mfpv	2	CURRENT_MOB_SDK_VER(VIP SDK)
fpts	1591685907667	System Current time in milliseconds
mfappid	com.verisign.mvip.iaapp.activity	Package name
mfa_isrooted	true/false	Device is rooted
mfa_id	12fe289ea3501f9b	ANDROID_ID
mfa_bd	bravo	Board
mfa_ca2	armeabi	CPU architecture
mfa_hw	mt6735	HARDWARE
mfa_sci	in	Sim Country
mfa_spn	Jio 4G	Sim Operator Name
mfa_so	405854	MCC+MNC (mobile country code + mobile network code)
mfa_cgc	2065684|115	Gsm cell id|gsm location area code
mfec	J2iGw5OztbxiZI5ViHeXmmUhlKKW 8KgSV9ejgdHwX9/ctiytJrip/cDVYxR lNfguctEJh8b1JQlYK02QPol/qwfma A9Pp6GoFF5Q1lcwYSyYgR4HiwAF qEoyeEgCliHMnA0XtK7Rb3RXUuW /JBvr1Qnu31vtZ4hZe0tONEZmrPqX Mv7FwUeBgJ0x9VwHskNI8r9NsF64 IB8Q59rd9G27Xvj4KfmWAfmWvG0 GnX3z31ECn58C8gBaIlZ2Z2p2Tv9 FJOKG3LqWNbLXkwSPCzhLudxm CBxCvq3CbjWS4Xc96tsffaQVE8Mm wObmHhR3Ldr9cWeZImpq7G+P y6B+ktglkQ==	Encryption Key

NOTE
The following attributes are not retrieved from Android version-10 and above:

Attribute Name	Example Value	Attribute Information
mfa_did	351894090778490	Device-Id
mfa_ssn	89918540400283440739	Sim Serial Number
mfa_sid	405854071389373	Subscriber ID

Required Permissions to get IAData attribute values:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

SDK for Risk iOS

The VIP SDK APIs with IA for iOS provides the sample code and describes the APIs needed to build your own mobile device application using VIP's Intelligent Authentication (IA) for the iOS platform. The APIs perform the following functions:

- Returns the device fingerprint data and device credential.
- Stores the provisioned device credential in the mobile storage.

This topic includes the following:

SDK Location

The Symantec Identity Security PlatformRISK iOS SDK is available for download from the Support Portal > My Downloads.

Prerequisite

Before working with the SDK documentation, ensure to obtain the VIP certificate by performing the steps given here.

Sample Code

The sample code illustrates possible ways to write your iOS application using the Intelligent Authentication APIs. The sample code shows the functions your application must perform to use the VIP iOS Intelligent Authentication APIs to create and manage your credential object.

> **NOTE**
> To develop an application with embedded VIP Intelligent Authentication SDK, you must add "CoreLocation.framework", "Security.framework", and "MobileCoreServices.framework" to your application.

The following sample code retrieves the fingerprint data of the mobile device.

```
MVIPIntelligentAuth *mvipIAHandler = [[MVIPIntelligentAuth alloc] init];
NSError *error = nil;
NSString *code = nil;
NSString *IAData = [mvipIAHandler getIAData:code error:&error];
if(IAData != null){
//Invoke evaluateRisk() webservice API passing the IAData in the request object.
//Parse evaluateRisk() response to get EventId, Status and KeyValuePair.
//If status is "RISKY" GOTO step 4
//if status is "NOTRISKY" and value of key "device.registered" is "true" return
//success to the user.
NSString *deviceTag = <value of key "device.tag" in evaluateRisk() webservice
response> BOOL status = [mvipIAHandler setIAData:deviceTag error:&error];
If(status){
NSString *IADataUpd = nil;
IADataUpd = [mvipIAHandler getIAData:code error:&error];
if(IADataUpd != nil){
6 //Invoke denyRisk() webservice API passing the IADataUpd in the request object.
//Parse denyRisk() response to get KeyValuePair.
```

```
//If value of key "device.registered" is "true" return success to the user else
return failure.
}
}
}
```

This sample code performs the following functions:

- Creates the MVIPIntellingentAuth object.
- Calls getIAData VIP Intelligent Authentication API to retrieve fingerprint data of the mobile device.
- Sends fingerprint data to VIP Services and initiates a risk-based authentication flow.
- Stores the device credential, received in the previous step, into mobile secure storage.
- Calls getIAData VIP Intelligent Authentication API to retrieve fresh set of fingerprint data of the mobile device.
- Completes the risk-based authentication flow and on success, registers the device to VIP Services.

IDSP APIs

MVIPIntelligentAuth class

Provides a functional interface to retrieve and store the fingerprint data of the mobile device.

Tasks

- ***getIAData*** - Retrieve fingerprint data of the mobile device.
- ***setIAData*** - Store the fingerprint of the mobile device in the mobile local storage.
- ***getVersion*** - Retrieve the version number of the IDSP library.

Class methods

- ***getIAData:***
  - Retrieves fingerprint data of the mobile device.
  - `(NSString *) getIAData: (NSString *)code error:(NSError**)error;`
- ***setIAData:***
  - Stores fingerprint of the mobile device in the mobile local storage.
  - `(BOOL)setIAData: (NSString *)IAData error: (NSError **)error;`
- ***getVersion:***
  - Returns the version of the IDSP library.
  - `(NSString *)getVersion`

Error Codes

The following table lists the error codes that you may encounter while using the IDSP iOS APIs:

Error Codes	Error Messages
1001	Internal Error
2001 - 2012	Certificate Error
3001 - 3022	I/O Error
4001 - 4006	Crypto Error
5001	Network Error

Sample Fingerprint of Mobile

Following is the sample fingerprint of mobile IAdata with Encrypted and Decrypted value:

IAdata Encrypted:


```
_v02ODM0JSU8MWg2Ojh7Nyc6NDE2Ojh7HTAxODonNDkGNDglOTAUJSVzMyUxIWhnczgzOiZoPBoGczgzOiNoZGZ7YnM4MyI0aGVnb2Vlb2Vlb2Vlb2Vlb2V
MjY4LTcQBxsGN2VhBWxkAg08ZGUyJBMMMQIgGAYUBRQ3N2EeOBgSZhMQMjJsJQA0GyQjHBgQZBssEQAUL2QHZhs6ECQbegI2IyYHOh8YNDEFEBwGJR4PImU
+bAx+GHoNbGUMPBgRBzQFFCwbOR8UGBwXJTgwHToTODIYPAAWZzY
+DwANGGElFgIXIzwMPgxtYAFjBjY8IxcgGGcNZTxjASY4DTsmJT0PGjgAIWI7fh4ZYA9+OxMdbRJiBi0lNDkcZjMUNGITPAQxLyNgNjpsNjAFDAEUA2QkEg
```


IAdata Decrypted:


```
mfappid=com.broadcom.HedmoralSampleApp&
 fpdt=2&mfos=iOS&mfov=13.7&mfwa=02:00:00:00:00:00&mfpv=1&fpts=1613028239335&mfi_name=Yashpal
 Singh Slathias iPhone&mfi_md=iPhone&mfi_lmd=iPhone&fpln=en-IN&mfsc=828|
1792&fptg=ad42f750e2964e4b957efc920d63de76&mfec=QE63Ue71sxpR97cpGS8wg8x
+cF8vLob7s5tLCx3H6VvPJLjgcmxbERNSb04P91WXi10gqFYdWuMSAPAbb4KmMG3FEgg9pUaNqvIME1NyDUAz1R3NoEqN/
WcvsRoJMadPEISpKZw0iWpCSnCvba9yURwg4xHP+Fcs5UMCTXF7AXUvnpiQhNnQOk9Y+M/
X90YiMDRaPAyNlJAMIBpmeHoFmgMiUC2ckZUXM4pCWBviYkY85T6ScivBuM2X0i6TsmXnsphZOmUt7n+KL5Z
+nFH8G7SxpalI3fAa7FiQdzv5co9cePYTAV1qGQx7a6SvodHLhaKMieshOBP7eR7pmmiRfg==&mfnc=6999666&mfmd=MEfsgqv6clB1k6rDKhGRnGRm6iC
```


Table displays the attribute details used in the Sample fingerprint:

Attribute Name	Example Value	Attribute Information
mfappid	com.broadcom.HedmoralSampleApp&	Application Id
fpdt	2&mfos=iOS	DeviceType Field
mfov	13.7	OS Version Field
mfwa	02:00:00:00:00:00	Mac address through wifi Manager
mfpv	1	CURRENT_MOB_SDK_VER(VIP SDK)
fpts	1613028239335	TimeStamp Field
mfi_name	Mal Bail	Device Name Field
mfi_md	iPhone	Device Model Field
mfi_lmd	iPhone	Device Localized Model Field
fpln	en-IN	Device language Field
mfsc	8281792	Mobile Screen Dimensions Field
fptg	ad42f750e2964e4b957efc920d63de76	
mfec	QE63Ue71sxpR97cpGS8wg8x +cF8vLob7s5tLCx3H6VvPJLjgcmxbERNSb04P91WXi10gqFYdWuMSAPAbb4KmMG3FEgg9pUaN +Fcs5UMCTXF7AXUvnpiQhNnQOk9Y +MX90YiMDRaPAyNlJAMIBpmeHoFmgMiUC2ckZUXM4pCWBviYkY85T6ScivBuM2X0i6TsmXnsph +KL5Z +nFH8G7SxpalI3fAa7FiQdzv5co9cePYTAV1qGQx7a6SvodHLhaKMieshOBP7eR7pmmiRfg==	Encryption Key
mfnc	6999666	Nonce field
mfmd	MEfsgqv6clB1k6rDKhGRnGRm6iC4yt/4FJAymEsbaqeBTW1u8D	Digest

Enabling Location Service Permissions

- Add "NSLocationWhenInUseUsageDescription" in the application Info.plist file and prompt the user for location service permissions through the application. In case you do not define the location service in the application, the IA iOS SDK does not consider location service attributes as part of the IA fingerprint data.

Reference Link: Preparing the IA based sample app in an iOS simulator or device

SDK for Risk Web

Integrate the Risk Web SDK with your web applications to enable risk-based transaction analysis and security features offered by IDSP.

Integrate the Symantec Identity Security Platform Risk Web SDK with your web applications to assess and perform risk-based transactions. This integration allows your application to leverage the security and threat detection capabilities of IDSP.

> **NOTE**
> This documentation only explains how the RISK Web SDK can be integrated with an existing Web app. For more information on the risk, see Managing Risk.

SDK Location

The IDSP RISK Web SDK is available for download from the Support Portal > My Downloads.

Integration Steps

Follow these steps to integrate the IDSP RISK Web SDK with your web application:

1. Add the IA Risk JavaScript library from the location mentioned in the previous section.
2. Call the JavaScript function, `IaDfp.readFingerprint()` to return the device fingerprint value. The device fingerprint value should be submitted as part of the request parameters while calling either the "authenticate" API or "UserRiskScoreEvaluator" API directly.
   Use either of the following API URLs:
   – ***Authenticate API*** :
   `hhttps://${SSP_FQDN}/{{tenantName}}/auth/v1/authenticate`

   > **NOTE**
   > - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
   > - The default value for `<tenantName>` is "default".
   > - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
   >   For example:
   >   `https://{{host}}/{{tenantName}}/auth/v1/authenticate`
   >   should look similar to:
   >   `https://example.com/default/auth/v1/authenticate`
   > - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

   – ***Evaluate Risk API***:
   `https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/UserRiskScoreEvaluator`

   Example of Device Fingerprint:
   `_v02MyUgNGgYOi88OTk0emB7ZXV9GDQ2PDshOiY9bnUcOyEwOXUYNDZ1GgZ1DXVkZQpkYApjfHUUJSU5MAIwNx48IXpgZmJ7ZmN1fR4dARgZeXU5PD4v`

   Sample Code:

```
this.iaFingerPrint = window["IaDfp"].readFingerprint();
```

3. Assuming that the user is logging in for the first time and the device from which the user signs in is unrecognized (different from the device pattern that is established within the user's profile), you can prompt the user to decide whether the device should be recognized ("remembered") for future access events.
   For example, if you use a check box as your prompt and when the user selects the check box, the device stores a browser tag `(writeTag())` in the user's browser.
4. If the user selects to be remembered on the particular device so that the risk score is not triggered every time a user logs in, invoke `IaDfp.writeTag` (tag.value, remember.checked) with the device tag that is part of evaluate risk API response and the remember value. This JavaScript function writes the tag into the user's browser.
   However, if the device is already registered, then the device tag is displayed empty and there will be no need to invoke the (writeTag()).
   Sample Code:
   ```
   if (deviceTag != null) {
   window["IaDfp"].writeTag(deviceTag, true);
   }
   ```
5. Now, call the JavaScript function `IaDfp.readFingerprint()` once again to get the updated finger print with the device tag and send this new updated signature in the post risk evaluator API using the following API:
   ```
   https://${SSP_FQDN}/default/iarisk/v1/PostUserRiskScoreEvaluator
   ```
   Sample Code:
   ```
   this.fingerPrintDna = window["IaDfp"].readFingerprint();
   ```

SDK for Mobile OTP Android

This topic includes the following sections:

- SDK Location
- Prerequisite
- Provisioning the credential
- Saving the credential
- Restoring the credential and generating a security code
- Synchronizing the time of creation

SDK Location

The Symantec Identity Security PlatformMobile OTP Android SDK is available for download from the Support Portal > My Downloads.

Prerequisite

Before working with the SDK documentation, ensure to obtain the VIP certificate by performing the steps given here.

Provisioning the credential

The following sample code provisions an authentication-based credential. After successful provisioning, the authentication credential object is returned to the mobile application, which can be used for generating the security code.

The Credential prefix (QAMT) provided in this sample are trial production account-specific values. Contact your Symantec representative for production values when you are ready to move to production implementation.

```
//Pass activationCode and the credential prefix to createTimeBasedCredential()
//factory method
CredentialFactory.createTimeBasedCredential(getApplicationContext(), testListener,
12345678, QAMT);
```

```
//testListener refers to instance of NetworkListener.

NetworkListener testListener = new NetworkListener(){
     public void onReceiveResponse(MVIPException vipExp, CredentialInterface
     credIRef){
          if(vipExp == null)
               Credential credObject = credIRef;
          }
}
```

This sample code performs the following functions:

- Asynchronously initiates the Authentication Credential provisioning through VIP Services with an activation code and the credential prefix.
- Returns the credential object (credObject) as part of the provisioning response to the NetworkListener callback method onReceiveResponse ().

Saving the credential

The following sample saves the credential object into the mobile device in encrypted form:

```
// Get the credential object and adds the credential object to an ArrayList of
//credential interface.
// Saves the data on the mobile device
ArrayList<CredentialInterface> credentialArray=new ArrayList<CredentialInterface>();
if (credObject != null){
     Vault repository = Vault.getInstance(activityContext);
          try {
               credentialArray.add((Credential) credObject);
               status = repository.storeCredential(credentialArray);
          } catch (MVIPException exception){
               // Catch and handle exception
          }
}
```

This sample code performs the following functions:

- Gets the credential object that is specified when the credential is provisioned, and adds it to an ArrayList of the credential interface.
  See Provisioning the credential.
- Saves the data on the mobile device vault.
  If you do not use the Vault interface that Symantec provides, you must first convert the Credential object to the JSON format using the toJSON API and then store it in the mobile device vault.

Restoring the credential and generating a security code

The following sample code retrieves the data from the mobile device, regenerates the credential object, and generates a security code.

```
// restore values from storage.
Vault vault = Vault.getInstance(activityContext);
String credentialId = null;
ArrayList<CredentialInterface> mCredentials = null;
try {
     mCredentials = vault.retrieveCredential(credentialId);
```

```
} catch (MVIPException exception){
     // Catch and handle exception
}
```

This sample code performs the following functions:

- Retrieves the serialized encrypted credential object that was saved earlier.
  See Saving the credential.
- Calls getSecurityCode on the first element in the ArrayList returned from the retrieveCredentialmethod. This call obtains the next sequential security code from the credential.
  The call to retrieveCredential with a credentialId returns an ArrayList that contains the credential with the given credentialId. In the case that a credentialId is passed as null to the retrieveCredential method, then an ArrayList of all credentials that are stored in the vault is returned.
  If you do not use the Vault interface that Symantec provides, you must reconstruct the credential object from the JSON stored in the vault (non-Symantec) using the credential's constructor and then call the getSecurityCode method on the credential object.

Synchronizing the time of creation

The following code re-synchronizes the time of creation of the credential with the VIP server time. This functionality must be used when the credential goes out-of-sync with the VIP server time and starts generating invalid security codes. The credential may go out-of-sync with the VIP server time if the time of the mobile device is changed or the VIP server time is changed.

```
//Asynchronously initiate the call to VIP Services to fetch the server time.
credObj.resetServerTime(testListener, "https://services.vip.symantec.com/prov");
NetworkListener testListener = new NetworkListener(){
     public void onReceiveResponse(MVIPException vipExp, CredentialInterface
     credIRef){
          if(vipExp == null){
               Credential credObject = credIRef;
               credObj = null;
//Store the new credObject
     .....
          }
     }
}
```

Re-generate the security code.

This sample code performs the following functions:

- Asynchronously initiates the call to VIP Services to fetch the server time.
- Returns the credential object (credObject) with the updated server time as part of the resetServerTime response to the NetworkListener callback method onReceiveResponse ().
- Stores the credential object on the mobile device.
  See Saving the credential.
- Re-generates the security code using the new credential object.
  See Restoring the credential and generating a security code.

SDK for Mobile OTP iOS

This topic includes the following:

SDK Location

The Symantec Identity Security PlatformMobile OTP iOS SDK is available for download from the Support Portal > My Downloads.

Prerequisite

Before working with the SDK documentation, ensure to obtain the VIP certificate by performing the steps given here.

Provisioning the credential

The following sample code provisions an authentication-based credential. After successful provisioning, the authentication credential object is returned to the mobile application, which can be used for generating the security code.

The Credential prefix (QAMT) provided in this sample are trial production account-specific values. Contact your Symantec representative for production values when you are ready to move to production implementation.

```
//Pass activationCode and the credential prefix to createTimeBasedCredential()
//factory method
1 [CredentialFactory createTimeBasedCredential:@"QAMT" activationCode:@"12345678"
success:^(id<CredentialProtocol> iRefCredential)
{
2 Credential * securityCredential = (Credential *) iRefCredential;
} failure:^(NSError *error) {
// Handle Error
}];
```

This sample code performs the following functions:

- Asynchronously initiates the time-based credential provisioning through VIP Services with an activation code and the credential prefix.
- Credential object (securityCredential) is returned as part of the provisioning response in the success callback.

Saving the credential

The following sample code gets the credential object and saves it:

```
NSError *error = [[NSError alloc] init]
  if(securityCredential != nil){
  NSArray *credentialArray = [NSArray arrayWithObjects:securityCredential, nil];
  // store the values in secure storage.
       BOOL isSucess = [self.vault storeCredential:credentialArray error:&error];
  NSString *CredentialId = [securityCredential getCredentialId];
       // Save credentialIds
  }
```

This sample code performs the following functions:

- Puts the credential object into an NSArray.
- Saves the array on the mobile device.
- Saves the credential ID of the credential.

If you do not use the Vault interface that Symantec provides, you must use the following code snippet to the credential object as NSData. Save this data in the encrypted form on the mobile device (for example, in the Keychain).

```
Credential *securityCredential = nil;
NSData *myData = nil;
NSMutableDictionary *dict = nil;
if(securityCredential!= nil){
     NSString *credentialId = [securityCredential getCredentialId];
```

```
    [dict setObject: securityCredential forKey:credentialId];
    myData = [NSKeyedArchiver archivedDataWithRootObject:dict];
    // Store myData (NSData) in the key chain storage.
}
```

Restoring the credential and generating a security code

The following sample code retrieves the data from the mobile device, regenerates the credential object, and generates a security code.

```
Credential *securityCredential = nil;
NSMutableArray *credentialArray = [[NSMutableArray alloc] init]; NSError *error =
[[NSError alloc] init];
// Restore values from storage
1 BOOL success = [self.vault retrieveCredential:credentialId
credentialArray:credentialArray error:&error]; if(success)
{
2 securityCredential = [credentialArray objectAtIndex:0];
NSString *code = nil;
3 code = [securityCredential getSecurityCode:&error];
}
```

This sample code performs the following functions:

- Retrieves the NSArray that was saved using the previous sample code by specifying the saved credentialID.
- Gets the first element in the array, which is the credential object.
- Calls getSecurityCode. This call obtains the next sequential security code from the credential.

If you do not use the Vault interface that Symantec provides, you must retrieve a CredentialTimeBased object that was saved in the mobile storage using the previous sample code.

```
NSData *credData = nil; NSMutableDictionary *query = nil; NSString *credId = nil;
// Get NSData object of credential from the keychain
*dict = [NSKeyedUnarchiver unarchiveObjectWithData:credData];
securityCredential = [dict objectForKey: credId];
```

Synchronizing the time of creation

The following code re-synchronizes the time of creation of the credential with the VIP server time. This functionality must be used when the credential goes out-of-sync with the VIP server time and starts generating invalid security codes. The credential may go out-of-sync with the VIP server time if the time of the mobile device is changed or the VIP server time is changed.

```
//Asynchronously initiate the call to VIP Services to fetch the server time.
[securityCredential resetServerTime:https://services.vip.symantec.com/prov
     success:^(id<CredentialProtocol> iRefCredential) {
Credential * newSecurityCredential = (Credential *) iRefCredential;
// Store the credential object newSecurityCredential.
// Re-generate security code.
} failure:^(NSError *error) {
// Handle Error
}];
```

This sample code performs the following functions:

- Asynchronously initiates the call to VIP Services to fetch the server time.
- Credential object (newSecurityCredential) with the updated server time is returned as part of the *resetServerTime* response in the success callback.
- Stores the credential object on the mobile device.
  See Saving the credential.
- Re-generates the security code using the new credential object.
  See Restoring the credential and generating a security code.

Getting the activation code

As part of the provisioning process, you must get an activation code from VIP Services. To obtain an activation code, send a getActivationCode API call to VIP Services.

VIP Services verifies the status of the issuer's VIP certificate and then sends a response containing an activation code. VIP Services also stores the activation code along with the issuer and the creation timestamp.

Activation Code Input Fields

The following table lists the getActivationCode input fields.

Table 99: getActivationCode input fields

Input Fields	Required	Type	Purpose
Version	Yes	VersionType	API version - either 1.0 or 2.0
Id	No	String	A pseudo-random number that is used for request-response matching. The Id is a maximum 40-char alphanumeric string.
ACProfile	Yes	ACProfileType	Kind of credential required. In this case, MOBILEPHONE.

Example of getActivationCode SOAP request

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns3="http://www.w3.org/2000/09/xmldsig#"
xmlns:ns1="https://schemas.vip.symantec.com/2006/08/vipservice">
     <SOAP-ENV:Body>
          <ns1:GetActivationCode Version="2.0" Id="EJHJ3351">
          <ns1:ACProfile>MOBILEPHONE</ns1:ACProfile>
          <ns1:OrderProfileId>1</ns1:OrderProfileId>
          </ns1:GetActivationCode>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Activation Code Output Fields

The following table lists the getActivationCode output fields. The response (if successful) contains the activation code along with the expiration time.

Table 100: getActivationCode output fields

Output Fields	Required	Type	Purpose
Version	Yes	VersionType	API version - either 1.0 or 2.0
Id	No	String	A pseudo-random number that is used for request-response matching. The Id is a maximum 40-char alphanumeric string.
Status	Yes	StatusType	Incorporates ReasonCode and StatusMessage
ReasonCode	Yes	hexBinary	If a getActivationCode request is unsuccessful, the ReasonCode provides the reason.
StatusMessage	No	String	States whether the getActivationCode request was successful.
ActivationCode	Yes	ActivationCodeType	Activation Code
ACExpiry	Yes	dateTime	Expiration date time of the activation code.

Example of getActivationCode SOAP response

```
<?xml version="1.0" encoding="UTF-8" ?>
     <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
     <Body>
     <GetActivationCodeResponse RequestId="DGID6448" Version="2.0"
https://schemas.vip.symantec.com/2006/08/vipservice
     <Status>
<ReasonCode>0000</ReasonCode>
<StatusMessage>Success</StatusMessage>
     </Status>
<ActivationCode>79287884</ActivationCode>
<ACExpiry>2008-08-06T15:03:25.174-07:00</ACExpiry>
     </GetActivationCodeResponse>
     </Body>
     </Envelope>
```

Password Encryption iOS SDK

Secure the authentication flow in iOS applications by implementing end-to-end password encryption using the IDSP iOS SDK.

Prerequisites

To use the iOS SDK for password encryption, the following prerequisites must be met.

1. Change the tenant setting of **fieldsToBeEncryptedInAuthFlow** to **password.** The default value is **none,** which indicates that password encryption is disabled by default.

```
[
   {
       "name": "fieldsToBeEncryptedInAuthFlow",
```

```
            "value": "password"
        }
    ]
```

2. Integrate the **AuthHubCore.framework** SDK with the iOS application. The SDK includes the APIs that can be called from the iOS application context to achieve end-to-end password encryption.

   NOTE
   The **AuthHubCore.framework** SDK file is available for download from the Support Portal > My Downloads along with other SDKs.

Integrating the SDK

To encrypt the password in the authentication flow, call the iOS SDK APIs from the mobile application context in the following sequence.

Step 1: Create an Instance of the AuthHubCore.framework AHCore Class

To create an instance of the **AuthHubCore.framework AHCore** class, invoke the **getInstance()** API. The API returns a singleton instance of the AHCore class.

```
public static func getInstance() -> AHCore
```

To know the version of the **AuthHubCore.framework,** invoke the **getVersion() API.** The currently supported version of the **AuthHubCore.framework** is 2.1.1.

```
public static func getVersion() -> String
```

Step 2: Initiate Authentication API

Call the **initAuth** function to initiate the authentication flow by invoking the POST `https://${SSP_FQDN}/<tenantName>/auth/v1/authenticate` API. Use this function at all existing places where you are invoking the */* **authenticate** API.

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/auth/v1/authenticate`
  should look similar to:
  `https://example.com/default/auth/v1/authenticate`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

API Signature

```
public func initAuth(serverHost: String,
                     tenant: String,
                     requestHeaders:[String:Any],
                     authRequestParams: [String:Any],
                     completionBlock: @escaping (( _ response: AHCoreResult) -> Void)) throws
```

Sample Code

```
do
{
   //get access token from server
   var jsonResponse = getAccessTokenFromAHServer()
```

```
    var accessToken = "Bearer \(jsonResponse["access_token"] as! String)"
    var reqHeaders = [String:Any]()
    reqHeaders["Authorization"] = accessToken
    try ahCore.initAuth(serverHost: serverHost, tenant: tenant, requestHeaders: reqHeaders, authRequestParams:
  paramBodyRequest)
    {
        response in
        if response.data["errorCode"] != nil {
            let errorCode = response.data["errorCode"] as! String
            let code = Int(errorCode)!
            var errorMsg = response.data["errorMessage"] as? String ?? ""
            if(!errorMsg.isEmpty) {
                print("Error: message - \(errorMsg) and code: \(errorCode)")
            }
        }
        else {
            if let flowState = response.data["flowState"] as? String {
                print(flowState)
            }
            guard let nextAction = response.data["nextaction"] else {
                Logger().logMessage(message: "Next Action not set!!")
                return
            }
            //handle nevigation to next page based on response from AH server
            navigateToNextPageBasedOnNextAction()
        }
    }

 } catch {
    let message = error.localizedDescription
    print("Error: message - \(message)")
 }
```

The API also supports the **proxy URL** configuration for the Symantec Identity Security Platform instance where the tenant name is not required as part of the URL. This function is used for initiating the authentication flow by invoking the POST `https://<proxyUrl>/auth/v1/authenticate` API. This function should be used at all the existing places where you are invoking the **/authenticate** API.

API Signature

```
public func initAuth(serverUrl: String,
                     requestHeaders:[String:Any],
                     authRequestParams: [String:Any],
                     completionBlock: @escaping (( _ response: AHCoreResult) -> Void)) throws
```

Sample Code

```
do
{
    //get access token from server
    var jsonResponse = getAccessTokenFromAHServer()
    var accessToken = "Bearer \(jsonResponse["access_token"] as! String)"
    var reqHeaders = [String:Any]()
    reqHeaders["Authorization"] = accessToken
    try ahCore.initAuth(serverUrl: proxyUrl, requestHeaders: reqHeaders, authRequestParams: paramBodyRequest)
```

```
    {
        response in
        if response.data["errorCode"] != nil {
            let errorCode = response.data["errorCode"] as! String
            let code = Int(errorCode)!
            var errorMsg = response.data["errorMessage"] as? String ?? ""
            if(!errorMsg.isEmpty) {
                print("Error: message - \(errorMsg) and code: \(errorCode)")
            }
        }
        else {
            if let flowState = response.data["flowState"] as? String {
                print(flowState)
            }
            guard let nextAction = response.data["nextaction"] else {
                Logger().logMessage(message: "Next Action not set!!")
                return
            }
            //handle nevigation to next page based on response from AH server
            navigateToNextPageBasedOnNextAction()
        }
    }

 } catch {
    let message = error.localizedDescription
    print("Error: message - \(message)")
 }
```

Parameter Description

- **hostName:** Specifies the SSP (IDSP) server hostname. Data Type - **string.**
- **tenant:** Specifies the tenant name. Example: default. Data Type - **string.**
- **serverUrl:** Specifies the endpoint URL that can be a proxy URL for an IDSP instance. Data Type - **string.** Ensure that the proxy resolves IDSP instance URL that points to the intended IDSP server host and tenant.
- **requestHeaders:** Represents a collection of IDSP server specific headers that are required for the REST API calls. The mandatory parameter to be passed is Authorization header containing **"Bearer <auth_token>".** Data Type - **dictionary.**
- **authRequestParams:** Represents the collection of parameters that are required for the /authenticate REST API calls. Data Type - **dictionary.** Sample parameters that can be passed are as follows:

```
    {
        "subject": "{{userLoginid}}",
        "channel": "mobile",
        "ipAddress": "10.10.123.124",
        "action":"authenticate",
        "rememberMe": true,
        "device": {
             "signature": {
                  "iaAuthData":
     "_v02MyUgNGgYOi88OTk0emB7ZXV9GDQ2PDshOiY9bnUcOyEwOXUYNDZ1GgZ1DXVkZQpkYApjfHUUJSU5MAIwNx48IXpgZmJ7ZmN1fR4dARgZeXU5PD
    NzclNDA6PzozOj06MDMyPDA9PzQ8KTw7ITAnOzQ5eDs0Njl4JTkgMjw7czMlJTxoczMlIS9oYHtgczMlOTtoMDt4EhcpKSlzMyU2JWhlZGRkZWRkZGRk
        }
      }
```

```
}
```

- **completionBlock:** Represents the closure block that the SDK passes to the calling API. The closure block returns the AHCoreResult object as a response. AHCoreResult is a struct object containing data (dictionary object), code (API response), and message (string) as members.

```
public struct AHCoreResult {
   public init(code: Int, message: String, data: [String:Any]) {
       self.code = code
       self.message = message
       self.data = data
   }
   public

var code : Int
   public
var message : String
   public
var data : [String:Any]

}
```

Step 2: Password Encryption Confirmation API

Call the **isEncryptionRequiredForField** function to check if the password field is configured for password encryption.

API Signature

```
public func isEncryptionRequiredForField(serverHost: String,
                                         tenant: String,
                                         requestHeaders:[String:Any],
                                         fieldName:String,
                                         completionBlock: @escaping (( _ response: AHCoreResult) -> Void))
  throws -> Bool
```

The API also supports the **proxy URL** configuration for the IDSP instance where the tenant name is not required as part of the URL.

API Signature

```
public func isEncryptionRequiredForField(serverUrl: String,
                                         requestHeaders:[String:Any]
                                         fieldName:String,
                                         completionBlock: @escaping (( _ response: AHCoreResult) -> Void))
  throws -> Bool
```

Parameter Description

- **hostName:** Specifies the SSP (IDSP) server hostname. Data Type - **string.**
- **tenant:** Specifies the tenant name. Example: default. Data Type - **string.**
- **serverUrl:** Specifies the endpoint URL that can be a proxy URL for an IDSP instance. Data Type - **string.** Ensure that the proxy resolves IDSP instance URL that points to the intended IDSP server host and tenant.
- **requestHeaders:** Represents a collection of IDSP server specific headers that are required for the REST API calls. The mandatory parameter to be passed is Authorization header containing **"Bearer <auth_token>".** Data Type - **dictionary.**
- **fieldName:** Specifies the field name that is marked for password encryption. The value of this parameter must be **password.**
- **completionBlock:** Represents the closure block that the SDK passes to the calling API. The closure block returns the AHCoreResult object as a response. AHCoreResult is a struct object containing data (dictionary object), code (API response), and message (string) as members. The result.data dictionary contains key as the field name (password) and boolean data (true, false) as a value. The value **true** indicates that the password field must be encrypted.

```
//get access token from server
var jsonResponse = getAccessTokenFromAHServer()
var accessToken = "Bearer \(jsonResponse["access_token"] as! String)"
var reqHeaders = [String:Any]()
reqHeaders["Authorization"] = accessToken

let ahCore = AHCore.getInstance()do {

   //get user password
   password = getUserPassword()
   try ahCore.isEncryptionRequiredForField(serverHost: serverHost,
                                           tenant: tenant,
                                           requestHeaders:reqHeaders,
                                           fieldName: "password", completionBlock: {
       result in
       if result.data["errorCode"] != nil {
           let errorCode = result.data["errorCode"] as! String
           let code = Int(errorCode)!
           var errorMsg = result.data["errorMessage"] as? String ?? ""
           if(!errorMsg.isEmpty) {
               print("Error: message - \(errorMsg) and code: \(errorCode)")
           }
           handleError(errorCode, errorMsg)
       }
       do {
           let code = result.code
           let isPswdEncRequired = result.data[AHConstants.GenericKeys.PasswordField] as? Bool ?? false
           if(code == AHResult.Code.SUCCESS && isPswdEncRequired) {
               let encryptedPassword = try ahCore.encryptdData(plainText: password)
               if(!encryptedPassword.isEmpty) {
                   password = encryptedPassword
               }
           }
           //send password authentication request to the IDSP server
           performPasswordAuthentication(password)

       } catch {
           print(error)
           return
```

```
        }
    })
}
catch {
    let message = error.localizedDescription
    print("Error: message - \(message)")
}
//get access token from server
var jsonResponse = getAccessTokenFromAHServer()
var accessToken = "Bearer \(jsonResponse["access_token"] as! String)"
var reqHeaders = [String:Any]()
reqHeaders["Authorization"] = accessToken


let ahCore = AHCore.getInstance()
do {

    //get user password
    password = getUserPassword()


    try ahCore.isEncryptionRequiredForField(serverUrl: proxyUrl,
                                            requestHeaders:reqHeaders,
                                            fieldName: "password", completionBlock: {
        result in
        if result.data["errorCode"] != nil {
            let errorCode = result.data["errorCode"] as! String
            let code = Int(errorCode)!
            var errorMsg = result.data["errorMessage"] as? String ?? ""
            if(!errorMsg.isEmpty) {
                print("Error: message - \(errorMsg) and code: \(errorCode)")
            }
            handleError(errorCode, errorMsg)
        }
        do {
            let code = result.code
            let isPswdEncRequired = result.data[AHConstants.GenericKeys.PasswordField] as? Bool ?? false
            if(code == AHResult.Code.SUCCESS && isPswdEncRequired) {
                let encryptedPassword = try ahCore.encryptdData(plainText: password)
                if(!encryptedPassword.isEmpty) {
                    password = encryptedPassword
                }
            }
            //send password authentication request to AH server
            performPasswordAuthentication(password)

        } catch {
            print(error)
            return
        }
    })
}
catch {
```

```
        let message = error.localizedDescription
        print("Error: message - \(message)")
    }
```

Step 3: Encrypt Password API

Call the **encryptDataWithPublicKey** function to encrypt the plain password with the tenant level public key without server nonce.

API Signature

```
public func encryptDataWithPublicKey(plainText: String) throws -> String
```

Call the **encryptData** function to encrypt the plain password with the tenant level public key along with server nonce.

API Signature

```
public func encryptData(plainText: String) throws -> String
```

Parameter Description

- **plainText:** Specifies the password string that must be encrypted in the authentication flow.

API Response

The API encrypts the plain text and returns the encrypted value as a base64 encoded string. The encrypted password can be sent to the IDSP where password encryption is enabled for authentication.

Reference Link:

- Password Encryption

Using the REST APIs

Explore how to leverage Symantec Identity Security Platform's comprehensive REST APIs for both end-user authentication flows and administrative management of the solution's infrastructure and configurations.

The Symantec Identity Security Platform provides a robust set of REST APIs designed to facilitate multi-factor authentication, OAuth 2.0-based authorization, and seamless integration with OpenID Connect compliant applications. These APIs cater to both end-users and administrators, offering powerful capabilities to manage identity and security within your environment.

> **NOTE**
> The 'Content-Type' HTTP request header is mandatory for HTTP POST, PUT, and PATCH REST requests across the Authentication, Management, and Self-Service APIs.

This topic includes the following sections:

- Requirements for OpenShift Environment
- API Validation
- Reference Links

Requirements for OpenShift Environment

In the OpenShift environment, when invoking tenant-specific APIs, it is mandatory to provide the "X-TENANT-NAME" header in the API request, else the API request fails with "400 BAD REQUEST" status.

Key Points:

- The tenant identified by "X-TENANT-NAME" must exist in the system
- SSL certificate must be issued for the SSP_FQDN host
- All tenant-specific API clients must include the "X-TENANT-NAME" header in their requests

For more information, see Deploying Ingress Controller on Openshift

API Input Validation

The Symantec Identity Security Platform includes comprehensive input validation for all REST API, operations including POST, PUT, and PATCH. This security enhancement protects against malicious input, including Cross-Site Scripting (XSS) attacks, script injection, and other security vulnerabilities. The validation system leverages the OWASP Java HTML Sanitizer library, a robust and well-tested security framework to detect bad characters or scripts in the request payload for RESTAPI.

> **NOTE**
> These API endpoints are excluded from the validation:
>
> - ".../admin/v1/LocaleContent"
> - ".../admin/v1/Messages"
> - ".../admin/v1/Notifications"

Sample Payload that shows bad character input in the request:

```
{
    "status": "active",
    "name": "{{appName}}<SCRIPT/SRC=\"http://xss.rocks/xss.js\"></SCRIPT>",
    "description": "Default Application for Security",
    "clientType": "CONFIDENTIAL",
```

```
    "allowedOpenIDScopes": [
        "openid",
        "email",
        "profile",
        "urn:iam:sso:idprop"
    ],
    "redirectURIs": [
        "https://www.example.com"
    ],
    "allowedGrantTypes": [
        "implicit",
        "authorization_code",
        "urn:ietf:params:oauth:grant-type:jwt-bearer",
        "client_credentials",
        "urn:ietf:params:oauth:grant-type:device_code"
    ],
    "allowedOperations": [
        "introspect",
        "onbehalfof",
        "offlineit"
    ],
    "metadata": [
        {
            "name": "myOrg",
            "value": "org123"
        }
    ],
    "skewTimeSecs": 0,
    "skipIssuerAudienceForIT": false,
    "userInfoEndpointResponseFormat": "PLAIN_JSON",
    "zeroFootPrint": false,
    "skipEmailForIT": false,
    "supportedJoseHeaderParams": "x5t,x5t#s256"
}
```

Sample Response: Displays the error after validation

```
{
    "errorCode": "0000137",
    "errorMessage": "Invalid characters '[<SCRIPT/SRC=\\\"http://xss.rocks/xss.js\\\">, </SCRIPT>]' found in
 the request"
}
```

Reference Links

This section includes the links to the categories of APIs, helping you effectively utilize the IDSP:

- Authentication APIs - Understand the REST APIs that enable end-user multi-factor authentication flows and identity verification.
- Self Service APIs - Discover the APIs dedicated to self-service functionalities, empowering users to manage their own credentials and account settings.
- Self Service Console APIs - Learn how administrators and end users can manage the behavior of groups and notifications in the Self Service Console.
- Management APIs - Learn how administrators can use APIs to access and control the solution's infrastructure, including organizational policies, message templates, tenant settings, and risk factors.
  - REST API Filter - Explore how to construct conditions and queries to refine and narrow down large datasets when retrieving information via the APIs.
- Identity REST APIs - Learn about the RESTAPIs that allow you to configure and manage the identities in the Internal Identity Store.
- Helper APIs - Understand the Helper APIs that allows clients to both retrieve information and execute capabilities ancillary to the core APIs of the Identity Service.

Authentication APIs

Symantec Identity Security Platform Authentication APIs enable secure user verification through various factors such as passwords, OTPs, push notifications, security keys and many more.

The IDSP Authentication APIs help certify the identity of users who are attempting to access protected resources. IDSP supports the authentication APIs that enables to initiate the initial authentication flow and also authenticate users with multi-factors, such as Security Key (FIDO), Password, OTPs, Push notifications.

This page includes the following links to sub topics:

- Initiate Authentication
- Authentication Factors

Initiate Authentication

Learn how to initiate authentication using the authenticate API, select authentication factors with the SelectedFactor API, and decrypt redirection URLs using the Decrypt Data API in Symantec Identity Security Platform.

This topic Includes the following sections:

- Initiate Authenticate API
  - Description of Request Headers in the Authenticate API
  - Invoking the ".../authenticate" call without a subject
  - Description of Request attributes in the Authenticate API
  - Sample of an Initiate Authenticate API
  - Description of Response attributes returned in the Initiate Authenticate API
- Factor Selection API
  - Configuring Default Authentication Factors
  - Restore Last Action
- Decrypt Data API

Initiate Authenticate API

Use the ".../authenticate" API to initiate an authentication flow in IDSP.

The following table displays the Initiate Authentication API URL and the operation type that is supported:

Operation Description	Operation Type	Endpoint URLs
The ".../authenticate" API allows you to Initiate an authentication flow	POST	https://${SSP_FQDN}/{{tenantName}}/auth/v1/authentica

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/auth/v1/authenticate`
  should look similar to:
  `https://example.com/default/auth/v1/authenticate`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Request Headers in the Authenticate API

The list of request headers that are used in the Authenticate call and their descriptions are given in the following table:

Header Name	Description
Authorization	Includes a Bearer Access Token that contains the **urn:iam:t.authenticate** scope.
(Optional) X-Flow-State	Contains the value of the **x-flow-state** query parameter that is received by the custom flow URL of a Client. Include this header in the Authenticate call *only* if the Client receives the **x-flow-state** query parameter through the custom flow URL redirect, initiated by a protocol such as OpenID Connect. If this header is included, do not send the **subject** attribute in the request.
(Optional) X-Reset-Flow	Set this request header to 'true' to implement the "Not User?" functionality. **For example**: When you click "Not Kevin Lucas?" on the password page that is displayed after you have initially signed in to the SignIn UI as Kevin Lucas, a new authenticate call is triggered. To remove the username, Kevin Lucas (subject) from the flow-state and add the new user to the authentication flow, you must include the request header "X-Reset-Flow=true" in this ".../authenticate" call.

Invoking the ".../authenticate" call without a subject

NOTE
When developing a custom sign-in application, the initial ".../authenticate" call without a subject (username) is typically needed when the sign-in flow is initiated by a protocol-driven app—such as a SAML app, OpenID/OAuth app, or a URL launcher app. However, for standalone native sign-in flows (mobile flows), such as those on iOS or Android apps, where the login process starts by directly prompting for the username, the initial ".../authenticate" call without a subject is not required.
For more information, see the "NextAction" attribute description.

Description of Request Attributes in the Authenticate API

The list of attributes that are used in the Authenticate call and their descriptions are given in the following table:

Attribute Name	Descripti
subject	Login id of the user involved in the authentication process.
ipAddress	IP Address from where the authentication is initiated. To ensure accurate and reliable IP address IP information, see Custom Headers for End User IP Address Usage.
action	The type of action through which the authentication is initiated. For example, - authenticate, login.
appName	Name of the application initiating the authenticate call.
channel	Type of channel from which the authenticate call is initiated. For example, - web, mobile.
acrValues	Provide the acr values. Example values: urn:iam:acr:aal:aal1, urn:iam:acr:aal:aal2 To know more, click here.
isReauthenticate	Indicates if this is a reauthenticate call. If this is set to true, then "existingIDToken" should contain here.
existingIDToken	ID Token previously issued by IDSP or third party token service. When using a third party token s issuer's certificate into IDSP. The token must contain the following claims: • must contain "amr" claim with values corresponding to IDSP's AMR values
	Authentication Factors
	Password
	SMSOTP, EmailOTP
	Push
	MobileOTP
	Biometric/Passkey
	Security Key
	Custom AMR
	• must contain the "user_loginid" claim to identify the user as this claim is used as the subject. • must have a certificate imported into IDSP matching the token's issuer.
state	Passed in on subsequent invocations to represent the ongoing state.
device.id	ID assigned to the device by IDSP in the previous calls. Not a mandatory parameter.
device.signature	Specifies the device signature of the device used by the user for authentication.
device.isTrusted	Indicates if this is a trusted device or not.

Attribute Name	Descripti

authContext	• As a client, when you create an application in the solution, if you have configured, '**itEncrypti** from the certificate configured in the attribute '**itEncryptionCertAlias**'. See Managing Applica However, if the Application expects the ID token to be encrypted with the Tenant's key instead configuration in the application, use the '**useTenantKeyForITEncryption**' attribute in the auth
	`"authContext": [ { "key": "useTenantKeyForITEncryption", "value": "true" } ]`
	• As a client, if you have registered to the FIDO-Biometric factor authenticator using either a bro called, '**fidoIdentifier**'. Configure this value in the "authContext" attribute, which will check for device. If you do not pass any value in the "authContext", a new registration is initiated. This a
	`"authContext": [ { "key": "fidoIdentifier", "value": "<encrypted-cred-id>" } ]`
	• When a client wants to obtain encrypted IT for itself, as well as obtain IT for the target the clie authentication request in this format:
	`"authContext": [ { "key": "useUserInfoKeyForITencryption", "value": "true" } ]`
	• When a client wants to request for an encrypted IT using a cert represented by a specific alias
	`"authContext": [ { "key": "encryptionAlias", "value": "<cert alias>" } ]`
	• When an Identity Token is acquired by performing a programmatic call (.../auth/v1/authenticate resulting Identity Token.
	`"authContext": [ { "key": "requestGroupsInIT", "value": "true" } ]`
	• When a client wants to encrypt the ***password*** or ***device signature*** end to end during the tran call. The value of the nonceid this will change with every transaction. For more information, se
	`"authContext": [ { "key" : "nonceId", "value" : "<serverNonceId>" } ]`

Attribute Name	Descripti

clientContext	• Sends additional context to a custom provider API, if a custom provider is configured in IDSP
	Example: To send "transactionIdentifier" as an additional context through the "/auth/v1/auther
	`"clientContext": [ { "key": "transactionIdentifier", "value": "540715af-96f2-4b86-b89f-ca0e8ae93f40" } ]`
	For information about using a custom provider, see Integrating with Custom SPI.
	• Sends dynamic context as key- value pairs within the "clientContext" parameter.
	Example: Pass the dynamic context through the "/auth/v1/authenticate" call as follows:
	`"clientContext": [ { "key": "department", "value": "ims" }, { "key": "env", "value": "prod" } ]`
	This example indicates that the client belongs to the "ims" department and is operating in the
	Such contextual information can be used in the context condition of the rules in an authenticat
	authentication policy.

domainName	Specifies the domain name of the FIDO (Biometric)/SecurityKey. **Example**: https://www.authhub.broadcom.com

Sample of an Initiate Authenticate API

Sample of Initiate **Authenticate Request**:

```
{
   "subject": "nbruce",
   "channel": "web",
   "ipAddress": "10.10.123.124",
   "action": "authenticate",
   "rememberMe": true,
   "device": {
       "signature": {
           "iaAuthData": "_v02MyUgNGgYOi88OTk0emB7ZXV9GDQ......"
       }
   }
}
```

Sample of Initiate **Authenticate Response**:

```
{
   "flowState": "RU9VMHlPWUZFVGdickNUWVFtVj.....",
   "userName": "nbruce",
   "nextaction": "AUTH_ALLOWED",
   "data": {
       "status": "OK",
       "message": "One-Time Password (OTP) is verified successfully"
```

```
    },
    "id_token": "eyJ4NXQjUzI1NiI6IkN..........",
    "additional": {
        "userName": "nbruce",
        "idpName": "TestConfig",
        "idpGuid": "452300a6-6777-4953-b486-6e38f62db340",
        "idpType": "ldap",
        "authLevelsData": {
            "60321052-3d2b-4f44-8ddc-a61b769344be+kp+1": "EMAILOTP"
        },
        "currentFactorLevel": 2,
        "amrForCompletedFactors": [
            "OTP"
        ],
        "isRememberMe": true,
        "lastObligation": "EMAILOTP"
    }
}
```


Description of response attributes returned in the Initiate Authenticate API

The list of attributes that are returned in the response of the Initiate Authenticate call and their descriptions are given in the following table:

Attribute Name	Description
flowState	This is the identifier at the server for the end-user authentication flow in the given instance.
userName	Name of the user who initiated the authentication flow.
nextaction	Provides information to the client about the next user authentication mechanism that must be performed to get access to the protected resource. • **In a Full Delegation flow** when the ".../authenticate" API is called without including the subject (username) in the request payload, the response returns "nextaction": null. In this case, the UI should prompt to enter username and make a second call to the "/authenticate" API with the subject in the request. • **In a Partial Delegation flow** when the ".../authenticate" API is called without the subject, the response includes a valid "nextaction" that specifies the next required factor (for example, OTP, push, etc.). The UI logic should be designed to implement this "nextaction" accordingly to guide the user through the authentication flow.
id_token	This is the JWT that contains claims about the authentication of the end-user.
currentFactors	Authentication factors that are assigned to the user.
additional • userName • idpName • idpGuid • idpType • authLevelsData • currentFactorLevel • isRememberMe • lastObligation	These depict the number of MFA levels that the user has completed.

Factor Selection API

The ".../SelectedFactor" API enables you to select one of the factors from the multiple options provided to you based on the Policy configuration to continue the authentication.

The following table displays the Factor Selection API operation type and endpoint URL:

Operation Description	Operation Type	Endpoint URLs
Select the factor with which you continue the authentication flow.	POST	https://${SSP_FQDN}/{{tenantName}}/auth/v1/SelectedFactor

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/auth/v1/SelectedFactor`
  should look similar to:
  `https://example.com/default/auth/v1/SelectedFactor`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Sample of Factor Selection Request:

The following example shows "PASSWORD" as the factor that a user has to authenticate with. However, the "factor" could include any of the factors that the policy evaluation has returned to the user.

```
{
   "factor": "PASSWORD"
}
```

The response for the factor selection API depends upon the factor that the user has selected.

For example, if you select **Attribute Based Authentication Factor**, then the factor should be passed in the following manner:

```
{
   "factor": "FACTOR/7d182f97-8351-4a5e-99a8-710a0a13d00"
}
```

For request and response payload of these URLs, refer to Authenticate API Swagger documentation.

Default Authentication Factors

In IDSP, you have the capability to set a default factor for a specific authentication level using the "***defaultAuthFactors***" tenant setting. This allows you to control the factor displayed in the SignIn UI, as opposed to it automatically showcasing the first factor returned by the authentication policy, typically sorted alphabetically. You can configure the default factors for all the four supported levels of authentications, based on your requirement.

Tenant Setting Configuration:

The "***defaultAuthFactors***" tenant setting is available through which you can configure the default factor that should be displayed at one or multiple levels.

Property Name	Property Scope	Property Description
defaultAuthFactors	Tenant	***When a user initiates the initial authentication flow or when no cookies or preferences are present in the agent (mobile app and web browser):*** • if the "defaultAuthFactors" property is ***not configured***, the SignUI displays the factors list as per the authentication policy evaluation. • if the "defaultAuthFactors" ***is configured***, the API response includes the default factor configured at the specified level, and the SignIn UI presents the associated action.

This setting is also covered in the Managing Settings page.

Sample Use Cases:

"defaultAuthFactors" Tenant Setting	Factors evaluated from the AuthnPolicies	SignIn UI behavior
"PASSWORD:1, SMSOTP:2"	PASSWORD:1, SMSOTP:2, EMAILOTP:2, PUSH:3, FIDO:3	***level 1*** - In this case, the Authentication Policy is configured with only one factor, "PASSWORD:1," Therefore, the "defaultAuthFactors" logic is not evaluated, and the Signin UI displays 'PASSWORD'. ***level 2*** - Defaults to 'SMSOTP' ***level 3*** - As there is no default configured at this level in the tenant setting, the Signin UI displays the list of factors that are returned from the Authentication Policy.
"PASSWORD:1, SMSOTP:2"	SMSOTP:1, EMAILOTP:1, PUSH:2, FIDO:2	***level 1*** - In this scenario, there is a discrepancy between the default factor configuration and the factors returned by the authentication policy, causing the Signin UI to display the list of factors that are returned from the Authentication Policy. ***level 2*** - In this scenario, there is a discrepancy between the default factor configuration and the factors returned by the authentication policy, causing the Signin UI to display the list of factors that are returned from the Authentication Policy.
"PASSWORD:1, SMSOTP:2, PUSH:3, FIDO:4"	PASSWORD:1, SMSOTP:2, EMAILOTP:2, PUSH:3, MOBILEOTP:3, FIDO:4	***level 1*** - In this case, the Authentication Policy is configured with only one factor, "PASSWORD:1," Therefore, the "defaultAuthFactors" logic is not evaluated, and the Signin UI displays 'PASSWORD'. ***level 2*** - Defaults to 'SMSOTP' ***level 3*** - Defaults to 'PUSH' ***level 4*** - In this case, the Authentication Policy is configured with only one factor, "FIDO:4". Therefore, the

"defaultAuthFactors" Tenant Setting	Factors evaluated from the AuthnPolicies	SignIn UI behavior
		"defaultAuthFactors" logic is not evaluated, and the Signin UI displays 'FIDO'.

Sample response when the next action is 'FACTOR_SELECTION':

```
"nextaction": "FACTOR_SELECTION",

"additional": {

"defaultFactor": "SMSOTP",

"currentFactorLevel": 2

}
```

Restore Last Action

The "restore last action" property allows you to cancel the factor that you have selected from the list provided on the factor selection page for authentication. You can then return to the original factor selection page where you have the option to choose another factor to proceed with the authentication flow.

To initiate a cancellation and return to the original factor selection page, you should modify the factor payload within the ".../SelectedFactor" API. Use "action" instead of "factor" and specify the value of "action" as "restorelastaction," as shown in the following example:

```
{
"action": "restorelastaction"
}
```

Decrypt Data API

Use the ".../DecryptData" API to decrypt redirection URLs managed by the IDSP UI. These URLs are encrypted using the server's private key. IDSP UI client must invoke this API to decrypt and retrieve the actual redirect URLs (such as homeUrl and errorUrl) configured in the application. These URLs are sent as part of the URL parameters in encrypted form.

The following table displays the DecryptData API URL and the operation type that is supported:

Operation Description	Operation Type	Endpoint URLs
The ".../DecryptData" API allows you to decrypt the encrypted redirection urls that are configured in the Application and sent as part of URL parameter in the Apps API.	POST	https://${SSP_FQDN}/{{tenantName}}/auth/v1/DecryptData

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/auth/v1/DecryptData`
  should look similar to:

```
https://example.com/default/auth/v1/DecryptData
```

- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Sample of Decrypt Data API request and response

Decrypt Data Request:

```
{
   "encryptedData":"{x5k}<encrypted data>"
}
```

Where,

- encryptedData - represents the encrypted redirect URL value.

Decrypt Data Response:

```
{
   "plainData":"<decrypted data>"
}
```

Where,

- plainData - represents the decrypted redirect URL value.

Reference Links to:

- Understanding the User Flow
- Next Actions Table
- Step-up/Reauthentication Concept
- Auth Manager Error Codes

Understanding the Authentication Flow

Discover the step-by-step authentication flow, highlighting how an ID token is obtained, and how this process securely enables legitimate users to access protected resources.

Multi-Factor Authentication (MFA) is a critical security process that requires users to provide multiple forms of verification to prove their identity, significantly enhancing protection against unauthorized access by combining different credential types like passwords and one-time passcodes.

In today's digital landscape, robust security is paramount. Multi-Factor Authentication (MFA) stands as a cornerstone of modern security, moving beyond simple passwords to offer an enhanced layer of protection. This document outlines a comprehensive MFA flow, detailing how users, applications, and a sophisticated Security Services Platform collaborate to verify identities.

This topic delves into the detailed authentication process, from initial user verification to the issuance of an ID token, guaranteeing secure access to protected resources for authorized users.

Example Scenario

Assumption: The following example assumes that you have updated your user record in the ID Store (LDAP/SCIM) with your mobile phone number to receive text messages.

> **NOTE**
> The phone number has to include the country code prefix. For example, 1 555 555 5555

Here is a list of objects and what they perform in this flow

- **User**: Person who accesses the application and has to be authenticated.
- **Application**: The application calls the solution's APIs to authenticate the user.
- **Security Services Platform**: A set of microservices running under .
  - AuthManager ("/auth")- Provides the authenticate API to initiate the authentication flow.
  - Risk Engine- (Internal) Risk API evaluates the risk score.
  - Policy Engine- (Internal) Policy API evaluates the authentication and authorization policies. Determines the authentication obligations.
  - Authorization Server ("/oauth2")- This is the "/oauth2" API that returns the access token.
  - ID Store- ("/idstore") This is the ID store service in charge of communicating with the LDAP/SCIM directory to authenticate and get user information.
  - Factors- This is the "/factor" API that generates and verifies the SMSOTP.

How does Multi-Factor Authentication Flow work?

1. The User invokes the application. The application acquires an access token by calling the "/oauth2/v1/token" API. In response, the API returns the "client access token".
2. The application calls the "auth/v1/authenticate" API to initiate the Authentication flow. Internally, the "/authenticate" API performs the following:
   a. Gets user information from the ID store.
   b. Communicates with Symantec Identity Security Platform-Risk Engine or Advanced Authentication risk engine (whichever risk engine is configured for the user) to evaluate the risk score.
   c. Communicates with the Policy Engine to evaluate the configured policy rules and determine authentication obligations.
   d. Returns the response with the authentication flow state and next action as "PASSWORD_AUTH" to the application.

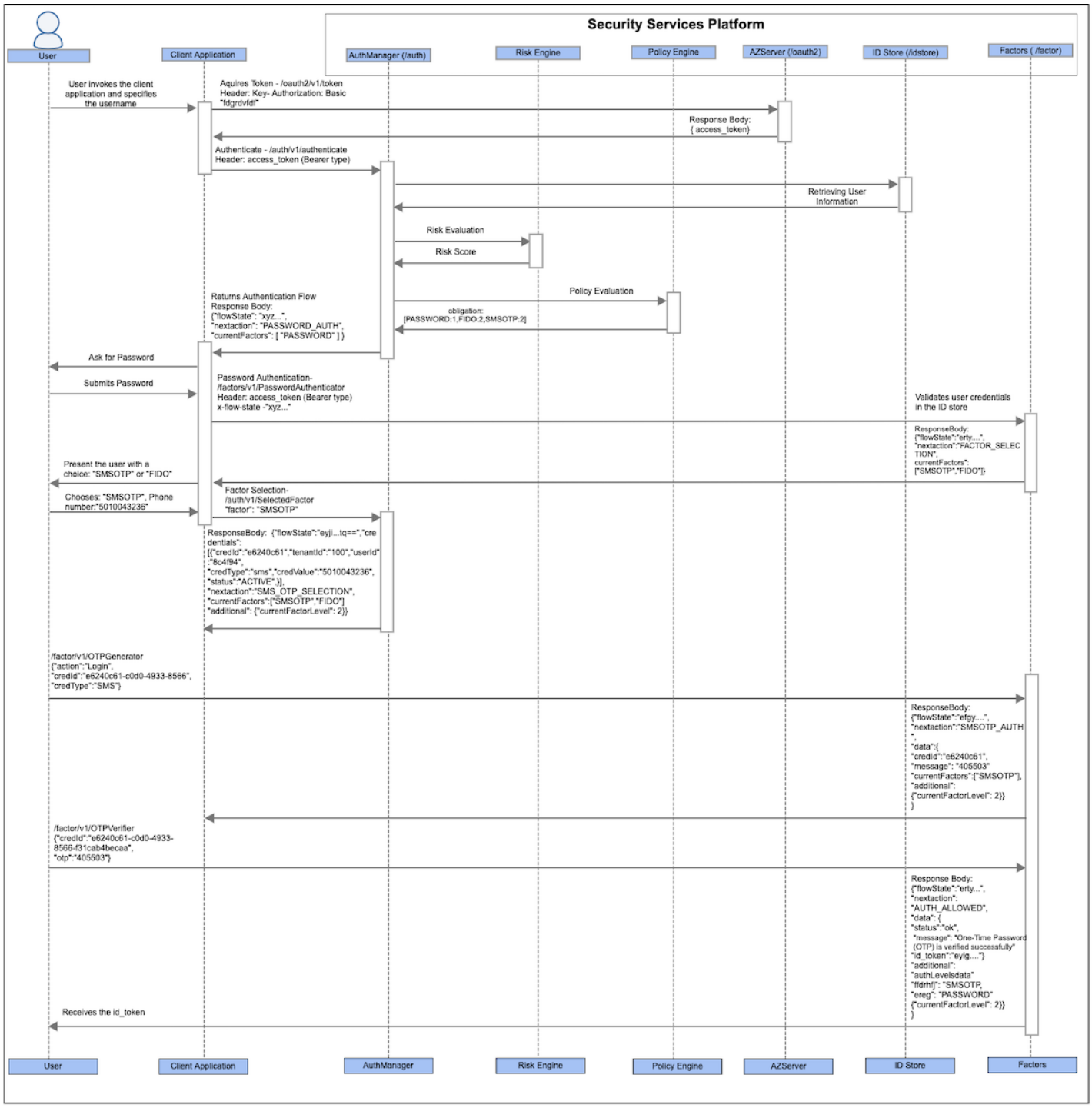
      NOTE
      In this example, we have assumed that the obligation generated by the policy engine is (Password:1, FIDO:2; SMSOTP:2).

      where, Password is the primary authentication factor, and FIDO and SMSOTP are the Secondary authentication factor.
3. The application requests the user to enter their password. The user submits the password.
4. The application calls the "/factors/v1/PasswordAuthenticator" API to validate the user credentials. The factors server returns the response with the Factor Selection flow state for the second factor ("SMSOTP" and "FIDO"). At this point, the primary factor authentication completes. Now, the user is supposed to select the secondary authentication factor from the two factors returned by the flow state, that is; SMSOTP and FIDO.
5. The user calls "/factor/v1/SelectedFactor" API to select the SMSOTP credential as the second-factor authentication. The response returns the authentication flow state and next action as "SMS_OTP_SELECTION".
6. The user invokes the "/factor/v1/OTPGenerator" API to generate the SMSOTP. The response returns the authentication flow state, next action as "SMS_OTP_AUTH", and the OTP along with other details.
7. On receiving the SMSOTP, the user invokes the "/factor/v1/OTPVerify" API to authenticate self.
8. The user is authenticated by the Factor server. The response returns next action as "AUTH_ALLOWED" with the ID token and other details.

   This completes the authentication flow. The user can now use the ID token to login to the protected page and perform user-related tasks.

The following sequence diagram shows the interaction between the user, application, and the solution's microservices.

User Authentication Flow



Next Actions Table

In the Next Actions table, you can view a snapshot of all the probable next actions that you may have to perform in a multi-factor authentication flow and also the use of the API URL's and the flowState that will be returned to the user.

In the Next Actions table, you can view a snapshot of all the probable next actions that you may have to perform in a multi-factor authentication flow and also the use of the API URL's and the flowState that will be returned to the user.

Next Action	API URL to be used	Request Body	Response Body
AUTH_ALLOWED	N/A		Redirects back to the clients home page
AUTH_DENIED	N/A		Displays an error, "Auth Denied"

PASSWORD_AUTH	/factor/v1/PasswordAuthenticator	<pre>{
		Password: password12
		}</pre>
			<pre>{
			"flowState":
			"eyJy...fQ==",
			"userName": "nbruce",
			"nextaction":
			"AUTH_ALLOWED",
			"data": {
			"message":
			"Password authentication
			is successful for user
			nbruce"
			},
			"id_token":
			"eyJ4...l5Hg",
			"additional": {
			"userName":
			"nbruce",
			"authLevelsData": {
			"450db18b-
			beb6-44cb-a1a8-c7e36bcc14ec
			+nbruce+1": "PASSWORD"
			},
			
			"currentFactorLevel": 1,
			"isRememberMe":
			true
			}
			}</pre>

FACTOR_SELECTION	/auth/v1/SelectedFactor	<pre>{
		"factor":"FIDO"
		}</pre> This can be any factor. FIDO is just an example.
			Next action will depend upon the factor you select

Next Action	API URL to be used	Request Body	Response Body

FIDO_REGISTER_GENERATE_CHALLENGE	/factor/v1/FIDORegChallengeGenerator	{
		"userName":"malbail",
		"deviceName":"susan_d1",
		"documentDomain":"https://
		host.domain.net:4200"
		}
			{"flowState":"",
			"nextaction":"FIDO_REGISTER_VERIFY_CHALLENGE",
			"data":{
			"challenge":"Yi1AQHaw_1qP0KgM30ci94sZsN6VXH7P-
			LL5ZCPeiR8",
			rp = { id = "broadcom.com";
			name = "broadcom.com"; },
			"id":"abcd"http://
			broadcom.net/">.net"},
			"user":{
			"id":"0ESaV",
			"name":"malbail",
			"displayName":"malbail"
			},
			"attestation":"none",
			"pubKeyCredParams":
			[{"type":"public-
			key","alg":-7}],
			"authenticatorSelection":
			{"userVerification":"preferred",
			"requireResidentKey":false
			},
			"excludeCredentials":[],
			"status":"ok",
			"errorMessage":""},
			"currentFactors":
			["SMSOTP","FIDO"]
			}

FIDO_REGISTER_VERIFY_CHALLENGE	/factor/v1/FIDORegChallengeVerifier	{"rawId": "eDiJ....oX6g",
		"response": {
		"attestationObject":
		"o2Nmb...Z1-4IH",
		"getTransports": {},
		"clientDataJSON":
		"eyJjaGFsb...0ZSJ9"
		},
		"getClientExtensionResults":
		{},
		"id": "eDiJ....poX6g",
		"type": "public-key"
		}
			{
			"flowState":"ewqs.....u09t",
			"nextaction":"FIDO_AUTH_GENERATE_CHALLENGE",
			"data":{"status":"ok"},
			"currentFactors":
			["SMSOTP","FIDO"]
			}

Next Action	API URL to be used	Request Body	Response Body

FIDO_AUTH_GENERATE_CHALLENGE	/factor/v1/FIDOAuthChallengeGenerator	{
		"documentDomain":"https://
		host.domain.net:3200",
		"userName":"NadiaKITE"
		}
			{
			"flowState":"ewqs.....u09t",
			"nextaction":"FIDO_AUTH_GENERATE_CHALLENGE",
			"data":{"status":"ok"},
			"currentFactors":
			["SMSOTP","FIDO"]
			}
			
			{
			"flowState":"qwDiJ....woX6",
			"nextaction":"FIDO_AUTH_VERIFY_CHALLENGE",
			"data":
			{
			
			"challenge":"ac569J9nGFVlCOqtl_3LN0yYn5TH1voHk
			"rpId" = "broadcom.com",
			"allowCredentials":
			[
			{"type":"public-
			key","transports":[],
			"id":"1GhAC....fcQ"}
			],
			
			"userVerification":"preferred"
			},
			"currentFactors":
			["SMSOTP","FIDO"]
			}

FIDO_AUTH_VERIFY_CHALLENGE	/factor/v1/FIDOAuthChallengeVerifier	{ "rawId": "1s5Q.....LLfcQ",
		"response": {
		"attestationObject":
		"czgT.....AEOg",
		"signature":
		MEYCIQCQEdbg,
		"userHandle": "null"
		},
		"getClientExtensionResults":
		{},
		"clientDataJSON":"eyJjn0"}
			{
			"flowState":"awqs.....09ht",
			"nextaction":"AUTH_ALLOWED",
			"data":{"status":"ok"},
			"id_token":"eyJ4....LqWA"
			}

Next Action	API URL to be used	Request Body	Response Body
EMAILOTP_REGISTER	/factor/v1/OTPRegistrar	{ "credType": "EMAIL", "credValue": "nbruce@broadcom.com" }	{ "flowState": "eyJy...Q==", "nextaction": "EMAIL_OTP_AUTH", "currentFactors": ["EMAILOTP"], "data": { "credId": "61c52b16-8121-48b3-92de-2458671fac8b", "tenantId": "85564e75-f799-48e1-ab51-78c43086de3b", "userId": "639f6488-209c-4a8b-8130-2029b0ab72c5", "credType": "email", "credValue": "840703 is your One-Time Password to login. The OTP is valid upto 5 minutes.", "status": "INITIAL", "createdDatetime": "2020-12-21 05:55 AM UTC", "modifiedDatetime": "2020-12-21 05:55 AM UTC", "default": false }, "additional": {} }
EMAILOTP_SELECTION	/factor/v1/OTPGenerator	{ "action": "Login", "credId": "61c52b16-8121-48b3-92de-2458671fac8b", "credType": "EMAIL" }	{ "flowState": "eyJy...TCJ9", "nextaction": "EMAIL_OTP_AUTH", "currentFactors": ["EMAILOTP"], "data": { "credId": "61c52b16-8121-48b3-92de-2458671fac8b", "message": "078250" }, "additional": { "currentFactorLevel": 1 } }

Next Action	API URL to be used	Request Body	Response Body

EMAILOTP_AUTH	/factor/v1/OTPVerifier	{
		"credId":"e6240c61-c0d0-4933-8566-f31cab4a"
		
		"otp":"403613"
		}
			{
			"flowState":
			"eyJy...n0=",
			"nextaction":
			"AUTH_ALLOWED",
			"data": {
			"status": "OK",
			"message": "One-Time Password (OTP) is
			verified successfully"
			},
			"id_token":
			"eyJ4...EL_g",
			"additional": {
			"authLevelsData": {
			"2ab486b8-b19f-4f13-9c9c-1a3c1f939908+malbail+1": "PASSWORD",
			"2ab486b8-b19f-4f13-9c9c-1a3c1f939908+malbail+2": "EMAILOTP"
			},
			
			"currentFactorLevel": 1
			}
			}

Next Action	API URL to be used	Request Body	Response Body
SMSOTP_REGISTER	/factor/v1/OTPRegistrar	{ "credType": "SMS", "credValue": "nbruce@broadcom.com" }	{ "flowState": "eyJy...Q==", "nextaction": "SMS_OTP_AUTH", "currentFactors": ["SMSOTP"], "data": { "credId": "61c52b16-8121-48b3-92de-2458671fac8b", "tenantId": "85564e75-f799-48e1-ab51-78c43086de3b", "userId": "639f6488-209c-4a8b-8130-2029b0ab72c5", "credType": "sms", "credValue": "840703 is your One-Time Password to login. The OTP is valid upto 5 minutes.", "status": "INITIAL", "createdDatetime": "2020-12-21 05:55 AM UTC", "modifiedDatetime": "2020-12-21 05:55 AM UTC", "default": false }, "additional": {} }
SMSOTP_SELECTION	/factor/v1/OTPGenerator	{ "action": "Login", "credId": "61c52b16-8121-48b3-92de-2458671fac8b", "credType": "SMS" }	{ "flowState": "eyJy...TCJ9", "nextaction": "SMS_OTP_AUTH", "currentFactors": ["SMSOTP"], "data": { "credId": "61c52b16-8121-48b3-92de-2458671fac8b", "message": "078250" }, "additional": { "currentFactorLevel": 1 } }

Next Action	API URL to be used	Request Body	Response Body

SMSOTP_AUTH	/factor/v1/OTPVerifier	{
		"credId":"e6240c61-c0d0-4933-8566-f31cab4ba"
		"otp":"403613"
		}
			{
			"flowState": "eyJy...n0=",
			"nextaction":
			"AUTH_ALLOWED",
			"data": {
			"status": "OK",
			"message": "One-Time Password (OTP) is verified successfully"
			},
			"id_token":
			"eyJ4...EL_g",
			"additional": {
			"authLevelsData": {
			"2ab486b8-b19f-4f13-9c9c-1a3c1f939908+malbail+1": "PASSWORD",
			"2ab486b8-b19f-4f13-9c9c-1a3c1f939908+malbail+2": "SMSOTP"
			},
			
			"currentFactorLevel": 1
			}
			}

Next Action	API URL to be used	Request Body	Response Body

MOBILEOTP_REGISTER	/factor/v1/VIPCredentialRegistrar	<pre>{
		"credType": "totp",
		"credValue": "Android Phone",
		"vipCredentialID":
		"SYMC59197959"
		"otp": "058691"
		}</pre>
			<pre>{
			"flowState":
			"ezZG....ZX0=",
			"nextaction":
			"AUTH_ALLOWED",
			"data": {
			"credId":
			"afa0e0bc-9350-4a9f-9fe2-
			b42f82bdbacf",
			"tenantId":
			"4764021a-9048-11ea-
			bb37-0242ac130002",
			"userId":
			"bab073a2-8201-4037-9b60-24adff61d3dc",
			"credType": "TOTP",
			"credValue":
			"Android Phone",
			"status": "ACTIVE",
			"createdDatetime":
			"2020-09-30 05:14 AM UTC",
			"modifiedDatetime":
			"2020-09-30 05:14 AM UTC",
			"default": false,
			"vipCredentialID":
			"VSST74448879",
			"userDisplayName":
			"nbruce",
			"tenantName":
			"default"
			},
			"deviceID":
			"45eaee7e-0cda-4b24-
			bd0a-3e8bd13ba123",
			"id_token":
			"eyJ4...lkqQ"
			}</pre>

Next Action	API URL to be used	Request Body	Response Body

MOBILEOTP_AUTH	/factor/v1/MobileOTPVerifier	{
		"credId":
		"18929a44-47fe-4efd-902d-46c6274fa631",
		"otp": "482501",
		"credValue":"Android Device"
		}
			{
			"flowState":
			"ezZG....ZX0=",
			"nextaction":
			"AUTH_ALLOWED",
			"data": {
			"credId":
			"afa0e0bc-9350-4a9f-9fe2-b42f82bdbacf",
			"tenantId":
			"4764021a-9048-11ea-bb37-0242ac130002",
			"userId":
			"bab073a2-8201-4037-9b60-24adff61d3dc",
			"credType": "TOTP",
			"credValue":
			"Android Phone",
			"status": "ACTIVE",
			"createdDatetime":
			"2020-09-30 05:14 AM UTC",
			"modifiedDatetime":
			"2020-09-30 05:14 AM UTC",
			"default": false,
			"vipCredentialID":
			"VSST74448879",
			"userDisplayName":
			"nbruce",
			"tenantName":
			"default"
			},
			"deviceID":
			"45eaee7e-0cda-4b24-bd0a-3e8bd13ba123",
			"id_token":
			"eyJ4...lkqQ"
			}

Next Action	API URL to be used	Request Body	Response Body

PUSH_REGISTER	/factor/v1/VIPCredentialRegistrar	<pre>{
		"credType": "totp_push",
		"credValue":"AndroidPhone",
		"vipCredentialID":
		"SYMC59197959"
		}</pre>
			<pre>{
			"flowState":
			"eyJhY...xzZX0=",
			"nextaction":
			"PUSH_AUTH",
			"data": {
			"credId":
			"44aef37f-e263-4241-
			a510-2d615b00a3a8",
			"tenantId":
			"4764021a-9048-11ea-
			bb37-0242ac130002",
			"userId":
			"eef7746e-175e-4a4f-8ce5-0978bd8d2ff0",
			"credType":
			"TOTP_PUSH",
			"credValue":
			"Android Phone",
			"status":
			"INITIAL",
			"createdDatetime":
			"2020-09-30 07:25 AM UTC",
			"modifiedDatetime":
			"2020-09-30 07:25 AM UTC",
			"pushStatusMsg":
			"Mobile push request
			sent",
			"default": false,
			"vipCredentialID":
			"SYMC59197959",
			"userDisplayName":
			"bsteer",
			"tenantName":
			"default"
			},
			"currentFactors": [
			"PUSH"
			]
			}</pre>

Next Action	API URL to be used	Request Body	Response Body

PUSH_REGISTRATION_STATUS	/factor/v1/PushStatus	<pre>{
		"credId": "44aef37f-e263-4241-a510-2d615b00a3a8"
		}</pre>
			<pre>{
			"flowState": "eQ1M...fQ==",
			"nextaction": "AUTH_ALLOWED",
			"data": {
			"status": "AUTHENTICATED",
			"message": "Mobile push request approved by user"
			},
			"deviceID": "93514d51-3284-487d-95c2-d6337afe226d",
			"id_token": "eyJ4....Efww"
			}</pre>

PUSH_SELECTION	/factor/v1/PushGenerator	<pre>{
		"credId": "66c4cac6-594d-4785-8eda-2b186facd1a5",
		"action": "LOGIN"
		}</pre>
			<pre>{
			"flowState": "eyJy....CJ9",
			"nextaction": "PUSH_AUTH",
			"currentFactors": [
			"PUSH"
			],
			"data": {
			"credId": "cd9a2b28-5d16-4a01-873d-514bbd006225",
			"tenantId": "85564e75-f799-48e1-ab51-78c43086de3b",
			"message": "Mobile push request sent"
			},
			"additional": {
			
			"currentFactorLevel": 1
			}
			}</pre>

Reauthentication/Step-Up

In Symantec Identity Security Platform, reauthentication is a security process requiring users to verify their identity again to ensure ongoing access control and protect sensitive information.

Reauthentication or Step-Up enhances security when accessing sensitive data. It verifies that the user accessing a protected resource during a valid session is the same user who initially authenticated. This validation occurs until the expiration of the ID token. If a user attempts to access a protected resource within this timeframe, the

"Reauthenticate" feature, which is governed by configurations in the ".../auth/v1/authenticate" API and ".../admin/v1/AuthPolicies" API, is triggered.

This topic includes the following sections:

- Reauthentication Parameter Dependencies and Authentication Outcome
- Reauthenticate Examples
- Returning the Updated ID Token
- Reauthentication in OIDC Authorise Calls

Reauthentication Parameter Dependencies and Authentication Outcome

Important:

The concept of Reauthentication depends upon the following parameter configurations:

- "isReauthenticate" - present in the ".../auth/v1/authenticate" call request body
- "existingIDToken"- present in the ".../auth/v1/authenticate" call request body
- "reAuthenticate"- present in the ".../admin/v1/AuthPolicies" API response body

The following table shows the use cases, dependencies between the parameters, and the outcome of authentication obligation:

Is the application sending ID_TOKEN? (Parameter name-" existingID Token")	Application requesting re-authentication (Parameter name -" isReauthenticate")	AuthN Policy configured to force re-authentication (Parameter name-" reAuthenticate")	Authentication flow	Final ID_TOKEN
False	False	True/False	All obligations are to be performed as per the evaluation of the AuthN policy	IDSP issues a new id_token
True	False	False	Step-Up The obligations present in the AuthN policy are compared with the obligations derived from the "amr" claim in the id_token. Only the missing obligations from AuthN policy are used for the step-up flow.	IDSP issues a new id_token with the following claims derived from the id_token in the request: • sub/ user_login • sid • amr - amr from the id_token in the request and the amr derived from the obligations are used for step up acr • aal - derived from the new amr • exp - new expiration time if the tenant setting-'useOriginalIdentityTokenExpiry' is set to false. Otherwise, the expiration time is taken from the id_token in the request.

<table>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>All obligations are to be performed as per the evaluation of the AuthN policy</td>
<td>IDSP issues a new id_token with the following claims derived from the input id_token in the request:
• sub/ user_login
• sid
• amr - amr from input id_token and the amr derived from the obligations used for step up acr. The following table contains the authentication factors and their mappings with the AMR values in IDSP:
<table><tr><th><u>Authentication Factors</u></th><th>AMR Values</th></tr><tr><td>Password</td><td>PWD</td></tr><tr><td>SMSOTP, EmailOTP</td><td>OTP</td></tr><tr><td>Push</td><td>PUSH</td></tr><tr><td>MobileOTP</td><td>MOTP</td></tr><tr><td>Biometric</td><td>FIDO</td></tr><tr><td>Security Key</td><td>HWK</td></tr></table>• aal - derived from the new amr
• exp - new expiration time if the tenant setting- 'useOriginalIdentityTokenExpiry' is set to false. Otherwise, the expiration time is taken from the id_token in the request.</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>All obligations are to be performed as per the evaluation of the AuthN policy</td>
<td>IDSP issues a new id_token</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True/False</td>
<td>All obligations are to be performed as per the evaluation of the AuthN policy</td>
<td>IDSP issues a new id_token with the following claims:
• amr - derived from the obligations used for step-up acr
• aal - derived from the amr but with the following claims derived from the id_token in the request:
— sub/ user_login
— sid
— exp - new expiration time if the tenant setting- 'useOriginalIdentityTokenExpiry' is set to false. Otherwise, the expiration time is taken from the id_token in the request.</td>
</tr>
</table>

True	False	False	A new tenant setting, **useGranularAMRForOTP**, has been introduced to provide more precise control over how One-Time Passwords (OTPs) are handled in your authentication policies. For more information on how this setting works and interacts with "skipFactorLevelForMatchingAmr" tenant setting, see Enhanced OTP AMR.	For final AMR results, see Enhanced OTP AMR.

NOTE
If the ID token is found invalid, a log is written to indicate that the 'existingIDToken' sent in the request was invalid. The system further processes the authenticate request as a new request and disregards the ID Token that was sent in the request.

Reauthenticate Examples

Sample of ".../auth/v1/authenticate" API with "isReauthenticate" parameter configured as 'true' in the request body.

```
{
    "subject": "{{userName}}",
    "channel": "web",
    "ipAddress": "10.10.123.124",
    "existingIDToken": "{{User Id Token}}",
    "isReauthenticate": "true"
}
```

Sample of a ".../auth/v1/authenticate" API response body where the 'reAuthenticate' parameter is 'true' in the 'result'.

```
{
  "description": "Sample reauthentication policy",
  "status": "active",
  "apps": [
    {
      "name": "default application",
      "id": "059bcbbc-c8b6-467d-85d0-6d15f5a5c0f7"
    }
  ],
  "rules": [
    {
      "conditions": [
        {
          "action": {
            "operator": "eq",
            "value": [
              "authenticate"
            ]
          }
        },
        {
```

```
          "ipAddress": {
            "operator": "in",
            "value": [
              "10.0.0.0/24"
            ]
          }
        }
      ],
      "result": {
        "effect": "allow",
        "reAuthenticate": "true",
        "obligations": [
          "Password:1",
          "OTP:1",
          "FIDO:2"
        ],
        "msg": "Reauthentication policy for default application"
      }
    }
  ]
}
```

NOTE
When you create a policy without the "reAuthenticate" parameter, then the default value is treated as false.

Example 1:

As per the following policy (Policy 1), the user logs into the sample application to perform a business action for which the policy returns Password:1, as the obligation. The user completes the "PASSWORD:1" obligation. As the "reAuthenitcate" option is set to 'false' in the policy, it implies that the user does not have to reauthenticate.

Sample AuthN Policy 1 (click here to expand)

```
{
    "policyName": "Sample Policy 1",
    "description": "Sample policy Record ",
    "status": "active",
    "apps": [
        {
            "id": "{{ReauthclientAppId}}"
        }
    ],
   "rules": [
        {
            "conditions": {
                "action": {
                    "operator": "in",
                    "value": [
                        "business"
                    ]
                },
```

```
                "channel": {
                    "operator": "in",
                    "value": [
                        "web"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD:1"
                ],
                "msg": "This policy is for Continuous authentication - Business"
            }
        },

    ]
}
```

Example 2:

As per the following policy (Policy 2), when the same user logs into the sample application for the second time, while the user's Id token is still valid, the user attempts to perform authentication for which the policy returns "Password:1", and "FIDO:2" as obligations. As the "reAuthenitcate" option is set to 'false' in the policy, it implies that the user has to perform a step-up. As per initial authentication, the user has already performed "Password:1" so the final obligation that the user has to perform is- FIDO:2.

The flow state is created with Password already completed and FIDO:2 is the remaining authentication.

Sample AuthN Policy 2 (click here to expand)

```
{
    "policyName": "Sample Policy 2",
    "description": "Sample policy Record ",
    "status": "active",
    "apps": [
        {
            "id": "{{ReauthclientAppId}}"
        }
    ],
   "rules": [
        {
            "conditions": {
                "action": {
                    "operator": "in",
                    "value": [
                        "authenticate"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web"
```

```
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD:1",
                    "FIDO:2"
                ],
                "msg": "This policy is for Continuous authentication - Business"
            }
        },

    ]
}
```

NOTE
IDSP does not challenge the user for any authentication in the following cases, and instead returns a modified identity token:

- When the obligations returned during reauthentication are the same as the authentication
- When no obligations are returned by the policy engine during the reauthentication as none of the policies match the criteria

Returning the Updated ID Token

After authentication is complete based on the auth obligations from reauth policies, IDSP returns the updated ID Token based on both, the auth obligations performed in initial authentication and reauthentication.

NOTE
When a new Identity Token is generated, the expiry of the token is based on the current timestamp + configured expiry duration. However, if a tenant wants to retain the expiry from the original ID Token then the user has to set the flag *"useOriginalIdentityTokenExpiry" = "true"* in the tenant settings. By default, this property is set to "false".

Reauthentication in OIDC Authorize Calls

If the authentication flow is initiated through the OIDC ".../authorize" API call, passing the "existingIDToken" is done by specifying it in the "id_token_hint" query string parameter. Passing "isReauthenticate" is done by specifying the query string parameter, "prompt=login".

Perform the following steps to enable reauthentication in OIDC Authorization calls:

Using Prompt and id_token_hint

During Reauthentication, the client is required to invoke the "/authorize" call by setting the following:

- prompt=login
- id_token_hint=<IDToken issued by IDSP (SSP)>

Example of an "/authorize" call passing the prompt and id_token_hint parameters

```
https://iam.example.com/default/oauth2/v1/authorize?client_id=4ceeda4d-5b6a-4680-92db-
abb4177e4a0a&scope=openid&response_type=id_token&redirect_uri=http://iam.example.com:88/siteminderagent/forms/
authhubcb.fcc&flowId=null&nonce=1584629668331&prompt=login&id_token_hint=eyJraWQiOiI0Y2VlZGE0ZC01Yj&state=http
%3A%2F%2Fauthiam.example.com
```

Custom Headers for End-User IP Address Usage

This document explains how Symantec Identity Security Platform utilizes application metadata to define custom headers with valid user or client IP addresses to be used across all authentication flows.

Standard IP address usage in environments with firewalls or load balancers can alter user/client IP addresses or make multiple requests appear to originate from a single source. IDSP's application metadata can be configured to explicitly define the IP addresses to be used across all authentication flows, even in such challenging network setups.

Defining Custom Header in IDSP Application Metadata

Application Metadata Attribute Name used in Admin Console	Application Metadata Attribute Name used in API	Application Metadata D
User IpAddress Headers to use (csv)	urn:iam:app:userIpAddressHeaderToUse	This application metadata is used to read th used to replace the user IP/client IP through **Note the following**: • When the metadata value is populated w TRUE-IP", these headers are used as pa provided they contain valid IP addresses are present in all the relevant IDSP APIs invalid values, IDSP falls back to using t header instead. • When the metadata value is populated w TRUE-IP", and the first header does not checks if the next coma separated value IDSP uses that value. • If multiple header names like "X-TRUE-I addresses, IDSP uses the first one in the • If a header name like"X-TRUE-IP" is pop IDSP falls back to using the XFF header

This application metadata is also covered in Table describing Application Metadata list.

Examples of Custom Header Usage

In this table we have used "**X-TRUE-IP-1**" as an example custom header name, but customers can customize this value to suit their preferences.

API Headers	Scenario	Final Value Usa
"urn:iam:app:userIpAddressHeaderToUse" : "X-TRUE-IP-1"	"X-TRUE-IP-1" : ""	Uses the XFF header value as the populate
"urn:iam:app:userIpAddressHeaderToUse" : "X-TRUE-IP-1"	"X-TRUE-IP-1" : "198.51.100.1.2"	Uses the XFF header value as the populate
"urn:iam:app:userIpAddressHeaderToUse" : "X-TRUE-IP-1"	"X-TRUE-IP-1" : "198.51.100.1"	Uses the "X-TRUE-IP-1" value as the popul
"urn:iam:app:userIpAddressHeaderToUse" : "X-TRUE-IP-1, X-TRUE-IP-2"	"X-TRUE-IP-1" : "198.51.100.1" "X-TRUE-IP-2" : "198.51.100.2"	In this case, both the IP values are valid. Ho TRUE-IP-1" header value for usage as this sequence of IP addresses.
"urn:iam:app:userIpAddressHeaderToUse" : "X-TRUE-IP-1, X-TRUE-IP-2"	"X-TRUE-IP-1" : "198.51.100.1.2" "X-TRUE-IP-2" : "198.51.100.1"	In this case, the "X-TRUE-IP-1" value is inva the "X-TRUE-IP-2" header value for usage a the list.

API Headers	Scenario	Final Value Usa
"urn:iam:app:userIpAddressHeaderToUse" : "X-TRUE-IP-1, X-TRUE-IP-2"	"X-TRUE-IP-1" : "198.51.100.1.2" "X-TRUE-IP-2" : "198.51.100.1.3"	In this case, both the header values are inva the XFF header value for usage.

Reference Links

- Initiate Authentication - (the "IP Address" attribute has a note on Custom Headers)
- Refresh Token - (This section has a note on Custom Headers)

Enhanced OTP Authentication Method References (AMR)

Learn about enhanced OTP AMR.

To enforce more precise control over how One-Time Passwords (OTPs) operate in your authentication policies, configure the **`useGranularAMRForOTP`** tenant setting. This parameter explicitly defines how the system identifies and validates incoming OTP requests:

- When ***useGranularAMRForOTP*** is ***true***:
  This property enhances the OTP Authentication Method Reference (AMR) by adding EMAIL and SMS as specific AMR values. This offers more granular control on the OTP AMR by providing a specific AMR match for either SMS or EMAIL. Upon successful completion, the final Authentication Method Reference (AMR) includes the specific method (EMAIL or SMS) along with OTP (for example, ["EMAIL", "OTP"] or ["SMS", "OTP"]).
- When ***useGranularAMRForOTP*** is ***false***: The OTP AMR remains a single, generic value (OTP), covering both EMAIL and SMS OTPs. In this scenario, the system checks for a generic OTP AMR match to confirm completion of an obligation. Upon successful completion, the final Authentication Method Reference (AMR) shows OTP.

  NOTE

  - **IMPORTANT**
    Starting with Symantec Identity Security Platform v4.0.2, Identity/Access Tokens include granular AMRs by default, regardless of the `useGranularAMRForOTP` tenant setting. This enhancement ensures that existing Refresh Tokens (created before enabling the setting) generate Identity/Access Tokens with proper granular AMRs without requiring user re-authentication.
  - This enhanced granularity is not supported for IVR (Interactive Voice Response) based OTPs.

How "useGranularAMRForOTP" Interacts with "skipFactorLevelForMatchingAmr"

The behavior of **useGranularAMRForOTP** is also influenced by the tenant setting, **skipFactorLevelForMatchingAmr**. This setting determines whether the system can bypass authentication levels if certain factors within that level have already been satisfied.

- When the tenant setting "***skipFactorLevelForMatchingAmr"*** is set to ***true***:
  Consider a scenario where your Authentication Policy (Policy 1) requires PASSWORD:1, SMSOTP:2, and EMAILOTP:2. If a initial login journey of a user included PASSWORD and EMAILOTP, then upon re-login with an existing ID Token, the system recognizes these fulfilled obligations and simply prompt AUTH_ALLOWED. This is because the true setting allows the system to skip further authentication steps when corresponding AMRs have already been satisfied.
  In this case, the final AMR's will include: PWD, OTP, EMAIL.
- When the tenant setting ***skipFactorLevelForMatchingAmr"*** is set to ***false***:
  Using the same Authentication Policy (Policy 1: PASSWORD:1, SMSOTP:2, EMAILOTP:2) and the initial journey of a user is PASSWORD and EMAILOTP, the behavior changes. When that user re-logs in with an existing ID Token, the system now prompts to complete SMSOTP. This happens because the false setting prevents the system from skipping authentication levels, even if other factors at that level have been met. It requires all defined factors at that level to be confirmed.

In this case, the final AMRs include: PWD, OTP, EMAIL, SMS.

Reference Links

- Managing Settings - ("useGranularAMRForOTP" tenant setting is also covered in this link)
- Initiate Authentication - (the "existingIDToken" attribute has a AMR table which references OTP AMR)
- Re-authentication - ("Reauthentication Parameter Dependencies and Authentication Outcome" table has a OTP AMR link)

Authentication Factors

An authentication factor is a credential used to validate the identity of a user who is trying to access a protected resource. There are different categories of authentication factors, such as single factor, second factor, multi-factor, etc.

An authentication factor is a credential used to validate the identity of a user who is trying to access a protected resource. There are different categories of authentication factors, such as single factor, second factor, multi-factor, etc.

Symantec Identity Security Platform uses multi-factor authentication mechanism to verify the authenticity of a user where the user is required to provide multiple means of identification from separate categories of factors. The user is given access only after the obligations that are required by these factors are met. Every additional factor added to the multi-factor authentication increases the certainty of the person seeking access to a resource is who, or what they say they are and decreases the possibility of any ambiguous attacker trying to obtain access.

The multi-factor authentication mechanism is categorized into the following three most common factors:

Authentication Mechanism	Description	Examples
Something you KNOW (Knowledge)	Things that you know	Password, Pin
Something you HAVE (Possession)	Things that you have. Security Key based authentication	Yubikey, OTP
Something you ARE (Inherence)	Some form of physical characteristic of you (Biometrics)	Fingerprint

IDSP supports advanced policies that make it possible to configure the exact combination of factors that are necessary or sufficient to validate user identity using such methodologies.

The following factors are supported by the solution:

- Password
- Security Key, Biometric (FIDO), Passkey
- OTP (SMS, IVR and EMAIL)
- Push
- Identity Provider
- Custom Authentication

Password Authentication

Learn about Password Authentication Flow in Symantec Identity Security Platform.

The most widely used authentication form is Password-Based authentication. Many multi-factor authentication mechanisms depend on passwords as one of the authentication criteria. This factor requires the user to provide a user name and password before they can access the protected resource.

- IDSP facilitates the use of external user directories through LDAP / SCIM protocol and/or an Internal Identity Store Configuration (using the SCIM protocol), for the storage of user information. This user information is validated during the password authentication.
- To use an encrypted password for authentication flow, see Password Encryption.

This topic includes the following sections:

- Using Internal Identity Store
- Password Authentication API
  - Password Authentication Flow

Using Internal Identity Store for Password Authentication

> **NOTE**
> If you have configured an external Identity Store during IDSP deployment, skip this step and move to the Password Authentication flow section.

If you plan to use the internal identity store for user authentication through the password flow, the administrator overseeing the Internal Identity Store can enforce password changes for users during their next login. Therefore, when authenticating through the Signin UI application, after the user has completed authentication (which typically includes password as a factor), the user is then redirected to a **Password Must Change** page.

> **NOTE**
>
> - In this scenario, the Signin UI exhibits a `Http status code 428` upon activation of the "/factor/v1/PasswordAuthenticator" API. The error message accompanying this code indicates: "1003033=Password must be changed".
> - The "`Password Must Change`" option is exclusively displayed when using the internal Identity Store, and is not applicable to external identity stores.

The following example describes the password authentication flow using the IDSP API and the step where password change is returned as the Next Action.

1. User initiates the Authentication flow using the following API:

   ```
   POST https://${SSP_FQDN}/{{tenantName}}/auth/v1/authenticate
   ```

   The API request includes the following attributes:

   ```
   {
       "channel": "web",
       "subject": "lawrence_haley",
       "ipAddress": "10.10.1X.X.1XX",
       "action":"authenticate",
       "acrValues": [],
       "authContext": [
           {
               "key": "useTenantKeyForITEncryption",
               "value": "false"
           }
       ]
   }
   ```

   The API returns the following (partial) response:

   ```
   {
   ```

```
    "flowState":
 "NWsvSVZPQUN1Zko5enlWT0JCYUxnUHlxSXlCVVZHK1J0M1l5VnpRR25PeUcyenkrekx1SmJPOUVPUkV6U0czN0tiVDFlSFU3K1RKb3pUQ3R4R2s0OW
    "userName": "lawrence_haley",
    "userId": "89ac5e06-208e-4799-8c06-0ab981aa9848",
    "nextaction": "PASSWORD_AUTH",
    "currentFactors": [
        "PASSWORD"
    ],

}
```


2. User invokes the following API to authenticate using a password:

   ```
   POST https://${SSP_FQDN}/{{tenantName}}/factor/v1/PasswordAuthenticator
   ```

   Enters password, for example:

   ```
   {"password":"Sample@1234"}
   ```

   The API returns the following partial response:


```
{
    "flowState":
 "NWsvSVZPQUN1Zko5enlWT0JCYUxnUHlxSXlCVVZHK1J0M1l5VnpRR25PeUcyenkrekx1SmJPOUVPUkV6U0czN0tiVDFlSFU3K1RKb3pUQ3R4R2s0OW
    "userName": "lawrence_haley",
    "userId": "89ac5e06-208e-4799-8c06-0ab981aa9848",
    "nextaction": "CHANGE_PASSWORD",
    "currentFactors": [
        "CHANGEPASSWORD"
    ],
    "data": {
        "errorCode": "1003033",
        "message": "Password must be changed",
        "passwordResetURL": "https://xxxx/default/ui/v1/signin/initiatePasswordReset?x-flow-
state=NWsvSVZPQUN1Zko5enlWT0JCYUxnUHlxSXlCVVZHK1J0M1l5VnpRR25PeUcyenkrekx1SmJPOUVPUkV6U0czN0tiVDFlSFU3K1RKb3pUQ3R4R2
}
```


After successfully changing the password in the flow, the user is successfully signed in.

Password Authentication API

The following table displays the IDSP supported Password Authenticator API, it is operation type, and the endpoint URL:

Operation Description	Operation Type	Endpoint URLs
This API allows you to authenticate yourself using your password in the authentication flow	POST	https://${SSP_FQDN}/{{tenantName}}/factor/v1/PasswordAuthenticator

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/factor/v1/PasswordAuthenticator`
  should look similar to:
  `https://example.com/default/factor/v1/PasswordAuthenticator`
- To know more about the scopes that are required to secure these APIs, refer to the "Scopes" section here.

NOTE
Example of "/PasswordAuthenticator" API - Requests

```
{"password":"{{defaultPassword}}"}
```

Use Case- Password-Based Authentication Flow

The following use case explains the password authentication flow in IDSP.

Assumption

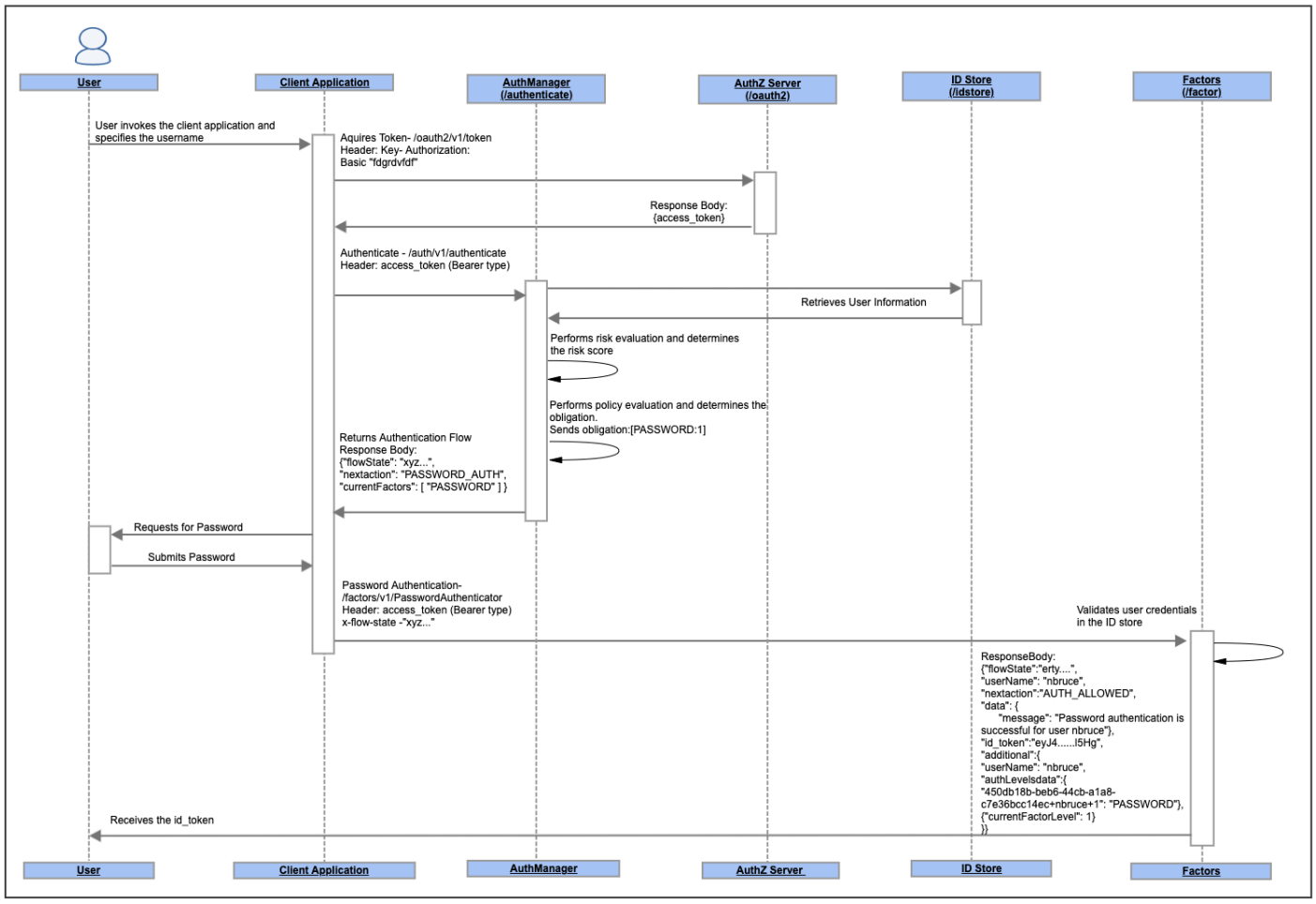
- Password is the only factor configured in the policy
- IDSP Risk has been enabled

Flow Description

1. The User invokes the application.
2. The application sends the user information using the "/token" API to the Authorization server.
3. The Authorization server determines whether the password sent from the client matches the password stored in the user store. If the server finds a match, it authenticates the user identity and sends the access token (AT) to the application.
4. Next, the application calls the "auth/v1/authenticate" API to initiate the Authentication flow. Internally, the "/authenticate" API performs the following
   a. Gets user information from the ID store.
   b. Communicates with Risk Engine or Advanced Authentication risk engine (whichever risk engine is configured for the user) to evaluate the risk score.
   c. Communicates with the Policy Engine to evaluate the configured policy rules and determine authentication obligations.
   d. Returns the response with the authentication flow state and next action as "PASSWORD_AUTH" to the application. In this example, we have assumed that the obligation generated by the policy engine is (Password:1) and that is the only authentication factor.
5. The application requests the user to enter the password. The user submits the password.
6. The application calls the "/factor/v1/PasswordAuthenticator" API to validate the user credentials. Upon authenticating, the factor server returns the response with the flow state, the next action as "AUTH_ALLOWED", ID token, and other details.

This completes the authentication flow. The user can now use the ID token to log into the protected page and perform user-related tasks.

The following sequence diagram shows the interaction between the user, application, and the solution's microservices to perform password authentication.



For information about the error codes, see Factor Error Codes, and ID Store Error Codes.

Password Encryption

Secure user credentials in Symantec Identity Security Platform by implementing password encryption both in-transit and at-rest.

Encrypting Password in Transit

IDSP utilizes end-to-end encryption (E2EE) to secure passwords during transmission. This ensures that sensitive credentials remain protected and are never exposed at intermediate nodes between the client—such as a mobile application or browser—and the backend system where verification occurs.

To secure the password with encryption during the authentication flow, perform the following steps:

1. Configure the "***fieldsToBeEncryptedInAuthFlow***" branding settings in the IDSP by invoking the following API: `https://${SSP_FQDN}/<tenantName>/admin/v1/Configs`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/Configs`
  should look similar to:
  `https://example.com/default/admin/v1/Configs`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

NOTE

1. The relevant configuration settings are covered in Global/Tenant Settings.
2. The tenant administrator can override the global settings using tenant settings.

Property Name	Property Scope	Property Description	Default Value	Comm
fieldsToBeEncryptedInAuthFlow	Global/Tenant/Branding Settings	• This property lists the fields that must be encrypted in the authentication flow. • As part of the branding setting, it helps the SignIn UI to decide if the password must be encrypted.	None	Possib passw

2. Launch the application and generate the client Access Token.

   NOTE
   If you use the JavaScript SDK, Android SDK, and iOS SDK, which is provided out-of-the-box by the solution, then the SDK performs steps- 2, 3, and 4 automatically. If you do not use the SDK, then follow steps 2, 3, and 4 as explained below.

3. Invoke the POST `https://${SSP_FQDN}/<tenantName>/auth/v1/NonceGenerator` API to get the server nonce by passing the encrypted client_nonce as input:

   NOTE
   If you have downloaded the JavaScript SDK, Android SDK, and iOS SDK, in step 1, then the SDK generates a client_nonce and store it in the client app (browser or mobile) for the session duration. The client_nonce is encrypted using the IDSP public key. See SDK for more details.

```
Authorization: <client Access Token>

Payload:
{
   "clientNonce": "<client_nonce encrypted with AH public key>"
}
```

   The API response generates a server_nonce and server_nonce_id and stores them in the cache. server_nonce is encrypted (AES256) with client_nonce as shown here:

```
{
   "serverNonceId": "<server_nonce>"
   "serverNonce": "<server_nonce encrypted with client_nonce>"
}
```

4. ***Initiate the Authentication Flow***
   a. Initiate the `https://${SSP_FQDN}/<tenantName>/auth/v1/authenticate` API by passing the userLogin and authContext information:
      - userLogin
      - userContext (Device Signature)

```
POST /auth/v1/authenticate
Headers
   Authorization: <client Access Token>
   SSP-TDI: <Encrypted TDI>

Payload:
{
   "subject": "<userLoginid>",
   "channel": "web",
   "ipAddress": "10.10.123.124",
   "action":"authenticate",
   "rememberMe" : true,
   "device": {
       "signature": {
           "iaAuthData": "<Encrypted device signature>"
       }
   },
   "authContext": [
       {
           "key" : "nonceId",
           "value" : "<serverNonceId>"
       }
   ]
}
```

b. Internally, the authentication process is initiated and a flow state is generated for the authentication flow. The server_nonce is stored as part of the flow state and is used for validating the encrypted password that is sent by the client. The Flow State Id returns the authentication obligations to the client app (browser or mobile app). Assuming that the obligation returned is the password, now initiate the password authentication.

5. ***Initiate Password Authentication***

a. Initiate the `https://${SSP_FQDN}/<tenantName>/factor/v1/PasswordAuthenticator` API by entering the encrypted password.

```
POST /factor/v1/PasswordAuthenticator
Headers
   Authorization: <client Access Token>
   X-Flow-State: <Encrypted Flow State Id>

Payload:
{"password":"{x5c}<IDSP public key encrypted (server_nonce+"~"+password)>"}
```

b. As discussed in step 2, IDSP uses the tenant configuration parameter, "**fieldsToBeEncryptedInAuthFlow**" to determine whether the password must be decrypted before it is processed in the authentication. If the "**fieldsToBeEncryptedInAuthFlow**" parameter has the value as "password", then the encrypted password is decrypted using the IDSP private key and the password is extracted using server_nonce stored in the Flow State.

c. Password is sent for authentication to the respective ID store (LDAP).

d. Password authentication status is returned to the client app (browser or mobile).
Based on the authentication policy configured in the solution, the customer may either get other obligations to perform, may get authenticated, or may get an auth denied.

Encrypting Password at Rest

To safeguard the confidentiality of the customer passwords, the passwords should only be verified in a hardened or temper resistant system. IDSP uses a Hardware Security Module (HSM), AWS Key Management Service (KMS) to

encrypt the password before storing it in the IDSP database. It uses the HSM to decrypt the password for comparison during the authentication flow. "AWS KMS" is the external encryption provider presently supported by IDSP.

Follow these steps to encrypt the password before storing it in the IDSP database:

1. Configure the external encryption provider in IDSP by using the EncryptionProviders API. The following example shows how to configure the external encryption provider in IDSP:

   NOTE
   The relevant Encryption Provider APIs are covered in here.

```
POST /admin/v1/EncryptionProviders
Headers:
   Authorization: <client Access Token or User Access Token>

Payload:
{
   "providerName": "aws-kms-encryption-provider",
   "providerProperties": [
       {
           "name": "cmkAliasName",
           "value": "arn:aws:kms:us-west-1:788212320683:alias/e2e-pwd-encryption-cmk"
       },
       {
           "name": "type",
           "value": "aws-kms"
       }
   ]
}
```

   where,
   – providerName: Specify the name of the external encryption provider to be used. Currently, AWS-KMS provider is supported.
   – providerProperties: These are key-value pairs that contain additional information required by the provider to perform its functions.
2. Configure the following application parameter to determine the authoritative source of the password by invoking the POST `https://${SSP_FQDN}/<tenantName>/admin/v1/Apps` API:

   NOTE
   For the Password Encryption feature to work, the "passwordAuthoritativeSource" property should be set to local.

Application Property Name	Property Description	Default Value	Comments
passwordAuthoritativeSource	This property identifies the authoritative source for password verification. • local - indicates password is stored in the IDSP account store (T_USER) and password verification happens against the IDSP account store • remote - indicates the password is verified against the remote identity store (ldap)	Remote	Possible values: Remote/Local

3. Configure the global/tenant-level setting to set the "***passwordProviderToBeUsed***" parameter:

NOTE
The tenant administrator can override the global settings using tenant settings.

Property Name	Property Scope	Property Description	Default Value	Comments
passwordProviderToBeUsed	Global/Tenant Settings	This property determines which provider to be used for encryption.	None	Possible values: None/Encryption provider name

4. Invoke the PUT `/<tenantName>/admin/v1/PasswordUpdater` API to update your password in the encrypted format in the IDSP database. The following sample shows how to invoke this API:

```
PUT /admin/v1/PasswordUpdater
Headers:
   Authorization: <client Access Token>

Payload:
{
   "userLoginId": "nbruce",
   "password”: "{x5c}<IDSP Public Key Encrypted user password>"
}
```

   where the user enters the password. The password is encrypted using the IDSP public key. This public key can be downloaded using the JWKS endpoint. **For example: https://<example.com>/<tenantName>/oauth2/v1/jwks**

   – The "/PasswordUpdater" API is invoked to check if the user exists in the IDSP database. If the user does not exist, then the user is disambiguated using the id store and a new user account is created in the IDSP database. However, if the user is not found in the id store, then IDSP displays an error message, "User Not Found".
   – User password update is performed based on the configuration of the tenant property - passwordCountToBeStoredInHistory.

   NOTE
   IDSP decrypts the password using the Tenant Private Key. The password is then hashed and encrypted with AWS KMS before being saved in the IDSP database. The SHA256 algorithm is used for hashing. The encrypted password hash is prepended with the encryption provider name before it is stored in the IDSP account store. For example, "aws-kms-example-wm"+ ":" + <hashed and encrypted password>.
   See Configuring Connectivity to AWS KMS for Password Encryption.

5. ***Initiate the Authentication Flow:***
   a. Invoke the `https://${SSP_FQDN}/<tenantName>/auth/v1/authenticate` API by passing the userLogin:

```
POST /auth/v1/authenticate
Headers
   Authorization: <client Access Token>
   SSP-TDI: <Encrypted TDI>

Payload:
{
   "subject": "<userLoginid>",
   "channel": "web",
   "ipAddress": "10.10.123.124",
   "action":"authenticate",
   "rememberMe" : true,
   "device": {
       "signature": {
           "iaAuthData": "<Encrypted device signature>"
       }
   }
}
```

b. Internally, the authentication process is initiated and a flow state is generated for the authentication flow. The Flow State Id returns the authentication obligations to the client app (browser or mobile app).
Assuming that the obligation returned is the password, now initiate the password authentication.

6. ***Initiate the Password Authentication***:
   a. Initiate the POST `https://${SSP_FQDN}/<tenantName>/factor/v1/PasswordAuthenticator` API by entering the password.

```
POST /factor/v1/PasswordAuthenticator
Headers
   Authorization: <client Access Token>
   X-Flow-State: <Encrypted Flow State Id>

Payload:
{"password":"<password>"}
```

   The password received from the client is hashed using SHA256. The password stored within IDSP, on the other hand, is decrypted with AWS KMS to get a hash of the password. The two password hashes are compared and if they match then the authentication is successful.

7. Optionally, after the first successful authentication to an application, a user can change its own password that is stored in an encrypted format in the IDSP database. To change the password, a user must invoke the PUT `https://${SSP_FQDN}/<tenantName>/factor/v1/MePasswordUpdater` API. The following sample shows how to invoke this API:

```
PUT /factor/v1/MePasswordUpdater
Headers:
   Authorization: <User Access Token>

Payload:
{
   "currentPassword": "{x5c}<Existing user password encrypted using IDSP public key>",
   "newPassword”: “{x5c}<New user password encrypted using IDSP public key>”
}
```

   In the API, the user must enter the current and new password in an encrypted format. The password can be encrypted using the IDSP public key which can be downloaded using the JWKS endpoint. For example: https://<example.com>/<tenantName>/oauth2/v1/jwks
   The API checks if the currentPassword matches the password that is stored in the IDSP database for that user. If the passwords match, the API validates if the newPassword meets the configured password policy criteria. If the password policy is met, the API successfully updates the new password in the IDSP database in an encrypted format.

Encryption Provider APIs

The following table displays the External Encryption Provider API operation type and endpoint URLs:

Operation Description	Operation Type	Endpoint URLs
This API allows you to create an external encryption provider in IDSP.	POST	https://<ssp-host>/<tenantName>/admin/v1/EncryptionProviders
This API returns the external encryption provider configured in the solution based on the specified provider Id.	GET	https://<ssp-host>/<tenantName>/admin/v1/EncryptionProviders/providerId
This API returns the list of all the external encryption provider's configured in the solution.	GET	https://<ssp-host>/<tenantName>/admin/v1/EncryptionProviders/

This API allows to update the external encryption provider by specifying the Provider Id.	PUT	https://<ssp-host>/<tenantName>/admin/v1/EncryptionProviders/providerId
This API allows to delete the external encryption provider by specifying the Provider Id. • Encryption Provider can be deleted only if there are no records that are encrypted with the encryption provider. • The name of the provider is prepended to the encrypted data before it is stored in the IDSP database.	DELETE	https://<ssp-host>/<tenantName>/admin/v1/EncryptionProviders/providerId

NOTE
Replace <text> with the appropriate values in all the endpoints shown in the following table.
For example:

```
https://<ssp-host>/<tenantName>/admin/v1/EncryptionProviders
```

Should look similar to:

```
https://auth.example.com/default/admin/v1/EncryptionProviders
```

Reference Links to:

- Configuring Connectivity to AWS KMS for Password Encryption

Learning Password Encryption Using the Encryption CLI Tool
The Encryption CLI Tool simplifies learning and implementing end-to-end password encryption, which operates at both in-transit and at-rest levels.

As part of the end-to-end password encryption implementation, Symantec Identity Security Platform supports the encryption of a password at two levels, in-transit and at-rest. The process of password encryption at each level involves multiple steps that include configuring branding settings, generation of a client nonce and a server nonce, authentication of a user and password, and so on.

Before implementing the end-to-end password encryption in your environment, familiarize yourself with the all the steps that are involved in the process. To aid in your learning, IDSP provides the Encryption CLI tool that simplifies steps within the password encryption process at the in-transit level. Use this hands-on experience to implement the end-to-end password encryption process as required in your environment.

The Encryption CLI tool performs the following tasks in the password encryption process at the in-transit level:

1. Generates a client nonce that is encrypted using the public key of IDSP.
   Send the generated client nonce in the request to the /NonceGenerator API call to generate a server nonce.
2. Encrypts a user password and server nonce using the public key of IDSP.
   Use the encrypted user password for a user authentication by the /PasswordAuthenticator API.

Additionally, the tool can also be used to encrypt a user password using the public key of IDSP. The encrypted user password can be updated in the IDSP database using the /PasswordUpdater API.

Run the Encryption CLI Tool

Follow these steps:

1. Ensure that AdoptOpenJDK 11 is installed.
2. Navigate to the Support Portal > My Downloads (Search for IDSP in the portal and select the IDSP 4.0 version number) from where you can downloaded the IDSP Encryption CLI Tool.

3. Open the command prompt in the same location and run the following command from the command prompt:
   ```
   java -jar EncryptionCLITool.jar
   ```
4. Enter the hostname of IDSP when prompted.
5. Perform one of the following steps as required:
   To generate a client nonce
   a. Type **1** and press Enter.
      An encrypted client nonce value is displayed, along with the corresponding non-encrypted client nonce value.
      Example:
      ```
      Encrypted Client Nonce: AHeHJp2F2aeDKqqCjP/fgBUK/ Zjb5bBSUwwleZjZLN dwF290T0mYAbQUutYMKN3yWpwof
       E9RH6M724wPpIGT18yzkhCv+ Gug==
      Non-encrypted Client Nonce: N6bezkrbakMkntacw7WhKDfMClijFC2a
      ```
   To encrypt a user password and server nonce using the public key of IDSP
   a. Type **2** and press Enter.
   b. Type the value of server nonce when prompted and press Enter.
   c. Type the user password that must be encrypted and press Enter.
      An encrypted user password is displayed.
      Example:
      ```
      Encrypted Password: iQ3iE9yE7EKHaKEwIlMoG+PMj18mULSKa9Mrj2mWvP
       JmeH23qHqYppWMO3A0gkLnCZgwUIKjTT7vlcyisk3ibUoiQC3exMhRUmtBCw
      ```
   To encrypt a user password using the public key of IDSP
   a. Type **2** and press Enter.
   b. Press Enter when prompted for a server nonce, without entering any value.
   c. Type the user password that must be encrypted.
      An encrypted user password is displayed.
      Example:
      ```
      Encrypted Password: QEmmlB62WcCiLW6OO4qRRsIYxT6iPBgHb zHPZTa5paZ3UUqP1l7O+MkF13xGsr9Z
      ```

Troubleshooting

Symptom:

The tool fails to run after entering a hostname and displays the following error:

```
java.lang.SecurityException: JCE cannot authenticate the provider BC
```

Solution:

As a workaround, perform the following steps:

1. Navigate to **jre\lib\security** and open the **java.security** file.
2. Append the following line of code at the end of the existing security.providers list:
   ```
   security.provider.<number-of-the-last-line>=org.bouncycastle.jce.provider.BouncyCastleProvider
   ```
   For example, if there are nine security.providers listed in the java.security file, add the following line as the tenth security.provider:
   ```
   line:security.provider.10=org.bouncycastle.jce.provider.BouncyCastleProvider
   ```
3. Save the change.
4. Download the latest Bouncy Castle provider JAR from MVN Repository, and copy it in the **jre\lib\ext** location.

Configuring Connectivity to AWS KMS for Encryption

Configuring Connectivity to AWS KMS for end-to-end Password Encryption

Symantec Identity Security Platform uses an external encryption provider, AWS KMS for encrypting and decrypting sensitive data, such as passwords. The instructions to connect the solution with the external encryption provider and using its services are outlined here.

This topic includes the following sections:

- Creating an IAM Policy, Role, and a Service Account in AWS
- Deploying IDSP using AWS KMS for end-to-end password encryption
- Verifying Access to the AWS KMS

Creating an IAM Policy, Role, and a Service Account in AWS

Follow these steps to create IAM Roles for Service Accounts (IRSA):

1. **Create an IAM policy that allows connection to AWS KMS for password encryption**.
   a. **Create a policy.json file** with the following content:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:ListKeys",
        "kms:Encrypt",
        "kms:Decrypt"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

> **NOTE**
> If you want to use an AWS MySQL DB along with the AWS KMS, then add the following action in the above policy.json file.
>
> `"Action": [ "rds-db:connect" ]`
>
> To know how to connect an AWS MySQL DB with IDSP, see Configuring Connectivity to AWS MySQL DB

   b. **Create the IAM Policy** using this command:

```
aws iam create-policy --policy-name ssp-aws-policy --policy-document file://policy.json
```

   where, "ssp-aws-policy" is the name of the policy.
2. **Create an OIDC provider for your Kubernetes EKS cluster**.
   a. Get the OIDC provider issuer name for your EKS cluster:

```
aws eks describe-cluster --name <cluster-name> --query "cluster.identity.oidc.issuer" --output text
```

   This command returns the OIDC issuer URL in the following format:
   `https://oidc.eks.us-west-1.amazonaws.com/id/<OIDC Issuer Id>`
   b. Check that the above OIDC issuer exists in the list of your existing OIDC providers by looking for its ID:

```
aws iam list-open-id-connect-providers | grep <OIDC Issuer Id>
```

   c. If the OIDC provider does not exist, create one (see AWS documentation for more information):

```
eksctl utils associate-iam-oidc-provider --cluster <cluster-name> --approve
```

   d. Use the OIDC Issuer Id from the previous command to validate that the new OIDC provider got created:

```
aws iam list-open-id-connect-providers | grep <This is the New OIDC Issuer Id>
```

3. **Create a Kubernetes IAM Service Account along with the role and map it to the Namespace, Cluster, and Policy**:
   This step associates the Kubernetes service account with the IAM Role. After performing this step, any pod that is running as this service account would be able to request an access token to connect to the database.
   This step does the following:
   - creates a Kubernetes Service account
   - creates an IAM Role and assigns the IAM Policy to the role
   - links the Kubernetes service account with the IAM role

     NOTE
     Before running the commands in this step,
     - Make sure to define the NAMESPACE environment variable containing the name of the Kubernetes namespace where your IDSP is going to be deployed. See Configuring the Environment Variables section.
     - Replace <cluster-name> with the name of your EKS cluster.

```
eksctl create iamserviceaccount \
    --name ssp-eks-iam-sa \
    --namespace ${NAMESPACE} \
    --cluster <cluster-name> \
    --attach-policy-arn arn:aws:iam::688201620683:policy/ssp-aws-policy \
    --approve \
    --role-name cluster-<cluster-name>-ssp-eks-iam-sa
```

4. **Check the Kubernetes Service Account creation for the Cluster**:
```
eksctl get iamserviceaccounts --cluster <cluster-name>
```

Deploying IDSP using AWS KMS for end-to-end password encryption

1. Deploy the IDSP chart following the steps mentioned in theConfiguring the Environment Variables section.
   When deploying the IDSP helm chart, use the following additional parameter:
   – Specify the existingServiceAccount to instruct IDSP pods to run as the Kubernetes Service Account created in step 3:
```
--set ssp.global.existingServiceAccount="ssp-eks-iam-sa"
```

Verifying Access to the AWS KMS

Validate that the AWS Pod Identity is properly deployed and configured after deploying the IDSP helm chart.

Follow these steps:

1. Log in to the AWS Pod identity to access the AWS Environment Variables:
```
kubectl describe pod/<any pod within the SSP1 namespace> -n ssp1
```
2. Access the AWS KMS related resources:
```
aws kms list-keys
```

Reference Links to:

- Configuring Connectivity to AWS MySQL DB

SiteMinder as a Password Authenticator

This topic describes the steps to configure SiteMinder as a password factor authenticator in the Symantec Identity Security Platform.

When SiteMinder is configured as a password authenticator in IDSP, and the latter is given the task of managing the authentication flow, the password policies that are set up in SiteMinder are used by IDSP. The control over password policies, such as the rules and limits governing the expiration, composition, and usage of a password, remains with SiteMinder even though the authentication has been handed off to IDSP. See Support for SiteMinder Password Policy in IDSP.

NOTE

- IDSP uses the Siteminder authentication and authorization web services (AuthAz web services) for user disambiguation, user password authentication, and change password. Therefore, ensure that your deployment allows the IDSP Kubernetes PODs to communicate with the Access Gateway URLs.
- The out-of-the-box SignIn UI does not support the "**password change**" flow when SiteMinder is configured for Password Authentication.

You should perform a few configuration updates in IDSP as described in the following section to include SiteMinder as a password factor authenticator in the authentication flow.

Configurations in IDSP to Set SiteMinder as a Password Authenticator

Perform these steps:

1. ***Import*** the SiteMinder Access Gateway Server certificate using the ".../admin/v1/Certs" API.
   Add this certificate to enable the SSL communication between IDSP and SiteMinder. To know more about the IDSP Certs API, see Certificate API.
2. Create a ***Custom Provider*** using the ".../admin/v1/CustomProviders" API.
   This setting creates the Service Provider Interface (SPI) implemented by IDSP for user disambiguation, password authentication, and update password operations using the SiteMinder authentication and authorization web services (AuthAz).
   The following example shows how the values should be specified in the Custom Provider API:

```
{
"providerName": "SiteMinderConfig",
"version": "1.0",
"securityMethod": "jwtbearer",
"certAlias": "AGServerSSLCert",
"url": "https://<<AGServerHostName>>",
"providerProperties": [
    {
        "name": "spi.description",
        "value": "Siteminder AuthAz WebServices for User Disambiguation and Password authentication"
    },
    {
        "name": "spi.discover.capabilities",
        "value": "smpasswordauthenticator"
    },
    {
        "name": "appId",
        "value": "app1"
    },
    {
        "name": "subject",
        "value": "<<admin user>>" // when security method is jwtbearer
    },
    {
        "name": "passwordProtectedResourceFilter",
        "value": "/loginservice"
```

```
        },
        {
            "name": "jwtProtectedResourceFilter",
            "value": "/jwt"
        },
        {
            "name": "jwtResetProtectedResourceFilter",
            "value": "/jwttoken"
        },
        {
            "name": "effectiveIdentitySourceId",
            "value": "<LDAP Config id>"
        },
        {
            "name": "attributeMapping",
            "value": "user_loginid=SM_UNIVERSALID, user_dn=SM_USERDN, email=mail,
    phone_number=telephoneNumber, status=SM_AUTHREASON, SM_USERLOGINFAILURESCOUNT=SM_USERLOGINFAILURESCOUNT,
    effectiveIdentitySourceId=effectiveIdentitySourceId"
        },
        {
            "name": "userLoginIdAttributeMappingName",
            "value": "user_loginid"
        },
        {
            "name": "groupAttributeMapping",
            "value": "name=SM_USERDN, members=SM_USERGROUPS"
        },
        {
            "name": "contentType",
            "value": "application/json; charset=utf-8"
        }
    ],
    "ignoreSSLValidation": true,
    "spiReadTimeoutMillis": "60000",
    "spiConnectTimeoutMillis": "60000"
    }
```

The CustomProvider Properties that are used in the API request are described here:

Property Name	Description	Example
url	This property is the Access Gateway Base URL (with the virtual host that is created for AuthAz web services). This property should be an HTTPS url.	https://aghostauthazws.broadcom.com Trailing slash '/' is not required.
certAlias	This property is the certificate alias of the certificate that is used for enabling SSL to the Access Gateway.	<based on IDSP configuration>
securityMethod	In SiteMinder, the base AuthAz web services URL (http://hostname:port/authazws/) can be protected with Basic/X.509Cert/JWT authentication schemes. IDSP supports only the **JWT** auth scheme.	Always 'jwtBearer'

subject	This property is always a required property as the securityMethod should always be jwtBearer. The value that is configured in this property is used as 'sub' claim in the JWT token that is passed as the authorization bearer token. The value is used for the base AuthAz url protection.	Any user name from the User Directory used in the SiteMinder Domain/Policy for the AuthAz web services base URL protection. <based on SiteMinder configuration>
passwordProtectedResourceFilter	This property is the resource filter which is protected with the Basic/Forms authentication scheme that is used for the SiteMinder AuthAz web services '/login' API call. Should be prefixed with a slash - '/'	<based on SiteMinder configuration>
jwtProtectedResourceFilter	This property is the resource filter which is protected with the JOSE web token authentication scheme that is used for the SiteMinder AuthAz web services user disambiguation or sync API call. Should be prefixed with a slash - '/'	<based on SiteMinder configuration>
jwtResetProtectedResourceFilter	This property is the resource filter which is protected with the JOSE web token authentication scheme that is used for the SiteMinder AuthAz web services for password reset API call. Should be prefixed with a slash - '/'	<based on SiteMinder configuration>
effectiveIdentitySourceId	This optional property is used when SiteMinder acts as an ID Store. This property is a valid LDAP or SCIM config id in the IDSP. When this property is configured, SiteMinder acts as a logical idstore and refers to the physical store configured in the LDAP or SCIM.	
appId	This property is the App ID that is used in the 'AgentName' property in the SiteMinder Agent Configuration Object that is created for AuthAz web services.	<based on SiteMinder configuration>
attributeMapping	This mapping is a list of coma-separated key-value pairs. • Keys are standard IDSP ID Store names. • These values are response attributes from the SiteMinder AuthAz web services When the 'LegacyVariables' parameter in the SiteMinder Agent Configuration Object is set to "no" or commented out, the response headers returned by SiteMinder Authentication and Authorization web services (AuthAZ) do not include underscores. Therefore, the value in the attributeMapping should refer appropriately.	
userLoginIdAttributeMappingName	This property is the key from the list of keys that are configured in 'attributeMapping' used to disambiguate user.	

groupAttributeMapping	This mapping is the list of coma-separated key-value pairs. • Keys are standard IDSP ID Store group attributes. • These values are response attributes from the SiteMinder AuthAz web services that represent the user group information.	
contentType This property is available from IDSP 2.2.5 release onwards.	This property is optional. The "contentType" property supports the following values: • application/json (default) • application/json; charset=utf-8	
spi.description	Description of the Service Provider Interface (SPI)	
spi.discover.capabilities	This value should be - 'smpasswordauthenticator'	smpasswordauthenticator
loginIdRegexEnabled	This attribute enables regular expression on the login ID before looking up in the ID store.	true/false
loginIdRegexFrom	This property is a regular expression to be applied to the user login ID.	"^(+.*)\\s(.*)$"
loginIdRegexTo	This property is a regular expression that is used to format the user login ID to match the corresponding attribute value in the identity store.	"0{2}"

3. Create an ***application*** using the ".../admin/v1/Apps" API, considering the following options:
   - ***Option A: Configure to use SiteMinder as the user disambiguator.***
     Update the "idStoreToUse" property in the app to use SiteMinder with <custom-provider-id>.
     For example, "idStoreToUse": "{sm}da548073-0dbc-49a5-94c3-386a8fac7566", where, the value-da548073-0dbc-49a5-94c3-386a8fac7566 represents the custom provider identifier that is generated in the previous step.
   - ***Option B: Configure to use any ID Store when multiple ID Stores are configured.***
     Update the "idStoreToUse" property in the app to be set to the configuration ID of the desired ID Store.
     For example, "idStoreToUse": "ab548073-0dbc-49a5-94c3-386a8fac1266," where the value "ab548073-0dbc-49a5-94c3-386a8fac1266" represents the LDAP configuration identifier generated.
   - ***Option C: Configure to use the default ID Store configured at the tenant level.***
     Leave the "idStoreToUse" property as empty in the app.
   - ***Option D: Configure LDAP as a User Disambiguation ID Store and SiteMinder as a Password Authenticator***.
     In this scenario, IDSP uses LDAP as the user disambiguation ID Store while SiteMinder serves as the password authenticator. Both IDSP and SiteMinder are connected to the same LDAP Directory server.
     a. Configure the LDAP configuration using the following attributes:
        - ***bindControlsCsv***: One of the value in this attribute should comprise the customer provider id that was configured in the previous step (prefixed with "{sm}"). This attribute supports coma-separated values.
          Example:"bindControlsCsv": "{sm}8997e500-bdbd-4fa7-87db-0db789b21ae8",
        - ***attributeMapping*** : Configure this ldap attribute with "***SM_USERLOGINFAILURESCOUNT=SM_USERLOGINFAILURESCOUNT***" attribute to sync with the user strike count.
          Example:

```
"attributeMapping": "user_loginid=cn, family_name=sn, given_name=givenName, email=mail,
 phone_number=telephoneNumber, user_universalid=cn, status=employeeType, memberof=memberof,
 title=cn, SM_USERLOGINFAILURESCOUNT=SM_USERLOGINFAILURESCOUNT"
```

NOTE

- Ensure that both IDSP and SiteMinder maps the "status" attribute to the same LDAP attribute. For example, if the status attribute is mapped to "employeeType" (status=employeeType), in IDSP, the same LDAP config should be set in the Siteminder User Directory.
  For example:
  – In IDSP, map "attributeMapping": "....., **status=employeeType**, ..."
  – In SiteMinder, perform the following actions:
    a. Navigate to the **Directory** > **User Directories**
    b. Select a particular user directory to modify.
    c. Navigate to **User Attributes** > **Disabled Flag (RW):** "employeeType"
- The LDAP attribute, "***SM_USERLOGINFAILURESCOUNT=SM_USERLOGINFAILURESCOUNT***" should be configured at SiteMinder.
  – In SiteMinder, the response that is attached to the rules protecting the realms (such as the resources that are protected with the JWT and Base/Forms authentication schemes) should return WebAgent-HTTP-Header-Variable SM_USERLOGINFAILURESCOUNT=< %userattr="SM_USERLOGINFAILURESCOUNT" %>

To know more about LDAP configurations in IDSP, see Managing LDAP Configuration.
To know more about how to create an application in IDSP, see Managing Applications.

4. Create an Authentication Policy using the ".../admin/v1/AuthPolicies" API
   Specify the following values:
   – in the principal condition, specify the ***user*** /***group*** value based on the option you chose during the creation of the app in the previous step.
     - if you have chosen ***Option A*** in the previous step, specify the ***user*** /***group*** value with the SiteMinder prefix in the principal condition. For example:

```
principal": {
                    "user": {
                        "operator": "in",
                        "value": [
                            "{sm}da548073-0dbc-49a5-94c3-386a8fac7566/*"
                ]
                    }
                }
```

     - if you have chosen ***Option B or Option C*** in the previous step, specify the ***user*** /***group*** value with the idStore config id in the principal condition. For example:

```
principal": {
                    "user": {
                        "operator": "in",
                        "value": [
                            "ab548073-0dbc-49a5-94c3-386a8fac1266/*"
                ]
                    }
                }
```

   – in the password ***obligation*** specify the value with the SiteMinder prefix

   The following example shows how the values can be specified in the Authentication Policy:

```
            "conditions": {
               "principal": {
                   "user": {
                       "operator": "in",
                       "value": [
```

```
                        "{sm}da548073-0dbc-49a5-94c3-386a8fac7566/*"
            ]
                  }
              },
              "channel": {
                  "operator": "in",
                  "value": [
                      "web",
                      "mobile"
                  ]
              }
          },
      "result": {
            "effect": "allow",
            "reAuthenticate": "false",
            "obligations": [
                  "PASSWORD/{sm}da548073-0dbc-49a5-94c3-386a8fac7566:1"
             ],
             "msg": "This policy is for admin login"
       }
```

To know more about the accepted ***user*** and ***group*** formats in the authentication policy, See Authentication Policy.

Changing Password during the Authentication Flow

During password authentication, if the response from SiteMinder indicates that the user has to change the password, IDSP returns the nextaction as "CHANGE_PASSWORD" before completing the authentication flow.

"

To change the password, invoke the ".../factor/v1/MePasswordUpdater" API. See Password Updater API.

NOTE

- The new password must meet the password criteria as configured in the SiteMinder password policy. See Password Services and Policies.
- If incorrect password or OTP is provided during authentication, the "userMaxStrikeCount"setting that is specified in IDSP synchronizes with the SiteMinder password policies so that both have the same userMaxStrikeCount. Upon successful authentication, the "userMaxStrikeCount" value resets to zero at both ends, IDSP and SiteMinder.

Changing Password Using Forgot Password or Self-initialized Reset Password Policy-based Flows

Your end users can initiate a password change using the Forgot Password link on the login screen, or in a self-initiated Reset Password flow in your client app. To configure these password reset flows in your client app, complete the following procedures:

Configure your SiteMinder implementation

Complete these general steps in your SiteMinder implementation. Refer to the SiteMinder documentation for specific procedures.

1. In the Agent Configuration object that is configured for integration with the Self Service Portal, enable the `enablepwselfchange` property.
2. If session store is enabled in your SiteMinder implementation, select **Prevent token replay (JTI validation)** in the Authentication Scheme.

Skip this step if session store is not configured on your SiteMinder implementation as the JTI is not saved in the SiteMinder database.

Prepare your client app

Complete the following steps to customize your client app to perform Forgot Password or self-initiated password reset operations for an end user:

> **NOTE**
> Steps 1 through 3 are only required if using a tenant-level client SignIn app. Skip these three steps if using the out-of-the-box system SignIn client app.

1. **(Tenant-level client SignIn app only):** On the **Authentication and Access Policies** page of the Admin Console:
   - Add the client app to the **ResetPasswordPolicy**
   - In the Auto-provision rule of the **ResetPasswordPolicy**, edit the authentication rules to identify to whom the policy applies.

   See Reset Password Policy.
2. **(Tenant-level client SignIn app only):** Configure the following settings in the **OAuth Client** tab of the **Applications** page of the Admin Console for the client app. See OAuth/OpenID.
   - Add the **Redirect URI**. The redirect URI here should match the redirect URI configured in the client app.
   - Select the **Authorization Code** grant.
   - Set the **Allowed Custom Scope** to `urn:iam:lcm:resetpassword`.
3. **(Tenant-level client SignIn app only):** Do not set the *X-CLIENT TENANT-NAME* to `system` in the Header mappings for the calls.
4. Set the client app to call one of the following APIs after the user is authenticated (after "CHANGE_PASSWORD" as the nextaction):
   - For the Forgot Password flow: Invoke the MePasswordReset API call as part of the flow state. This call only requires the new password.
     See Password Reset API.
   - For the Self-password reset password flow: Invoke the MePasswordUpdater API as part of the flow state. This call requires the old password and a new password.
     See #unique_1069.

Identifying SiteMinder Configuration in "/authenticate" API

To identify SiteMinder as a custom provider in the IDSP's ".../auth/v1/authenticate" call, check that the '***Application***' property, "***idStoreToUse***" accepts the SiteMinder Provider ID in this format - "***{sm}<custom-provider-id>***" as shown in the following example:

```
"idStoreToUse": "{sm}da548073-0dbc-49a5-94c3-386a8fac7566"
```

See Settings and Authenticate API.

Identifying SiteMinder Configuration in "/PasswordAuthenticator" API

To identify SiteMinder as a password authenticator in the IDSP's .../factor/v1/PasswordAuthenticator" call, check that the authentication policy has the authentication factor configured as "***PASSWORD/{sm}<custom-provider-id>***".

Artifacts Signature Verification between SiteMinder and IDSP

The following table shows the different artifacts, signing key required, and token signature verification approach.

Artifact	Signing Key	Verification Approach
JWT (for User Disambiguation) JWT (for /authazws security)	IDSP's Tenant Signing Key	These configurations are set in SiteMinder. In the JWT auth scheme, the verification approach considers either the value of the cert alias or the values that are specified in the JOSE headers.
Authorization Bearer Header	IDSP's Tenant Signing Key	These configurations are set in SiteMinder. In the JWT auth scheme, the verification approach considers either the value of the cert alias or the values that are specified in the JOSE headers.

Auto Enrollment of Credentials

The IDSP ".../auth/v1/authenticate" API internally calls the SiteMinder AuthAz web services, and if the response contains "email" and/or "telephoneNumber", these details are auto-enrolled as user Credentials.

Platform and Cross-Platform Authenticators

Learn about Security Key, Biometric and Passkey authenticators in Symantec Identity Security Platform, and related Browser Support Matrix.

The Security Key, Biometric (FIDO) and Passkey authenticators define the user authentication mechanism that reduces the dependability on passwords. Passwordless authentication is considered more secure for organizations as it swaps the use of a traditional password with more secure factors, improves user experience, greater convenience, and stronger security.

IDSP (solution) uses FIDO2 compliant authenticator to perform multi-factor authentication flows. To know more about the FIDO2 protocol, click here.

The solution has designed a passwordless authentication mechanism that supports both, Platform and Roaming Authenticators.

- ***Platform Authenticators*** are typically embedded into a user's computing device. The use of this authenticator is growing as part of the FIDO Alliance's FIDO2 authentication protocol. Platform Authenticators are referred to as ***BIOMETRIC*** authenticators in the solution's UI and in Hedmoral Mobile application. In the authentication policy, this authenticator is referred as **FIDO**. Supported platform authenticator types include Fingerprint and FaceID.
- ***Roaming Authenticators*** aka Cross-Platform Authenticators are not bound to any device. Users can use them with any number of devices. Roaming authenticators are referred as ***SECURITYKEY*** authenticators in the solution. These types of authenticators include Universal Serial Bus (USB), Bluetooth Low Energy (BLE), or Near Field Communications (NFC), and so on.
- **Passkey Authenticator**: This type of authenticator is not bound to the device but bound to the platform (Google account with Google Password Manager or any iOS, iPad, MacOS device, with iCould account). Passkey Authenticator is referred as PASSKEY authenticators in the solution's UI and in Hedmoral Mobile application. In the authentication policy, this authenticator is referred as **PASSKEY**.

  NOTE
  Passkey can behave as both, platform and roaming authenticator based on the way the passkey credential has been registered.

- If the passkey credential is registered using QR Code scan (where passkey stored in the account linked to the user, for example, Google account or iCloud account), then it works as a roaming authenticator.
- If the passkey credential is registered on a device which is linked to the Google account or iCloud account, then it works as a platform authenticator.

This topic includes the following sub-topics:

- Working with Security Key and Biometric Authentication
- How Security Key and Biometric Registration Works?
- How Security Key, Biometric and Passkey Authentication Works?
- Configuring Tenant Settings
- Platform and Cross-Platform Authenticator Authentication APIs
- Security Key/Biometric Platform and Browser Support Matrix

Working with Security Key, Biometric and Passkey Authentication

The solution exposes APIs to authenticate identities using factors Security Key, FIDO (refers to Biometric authentication factor), and Passkey. There are two main use cases to consider when using these authentication factors, ***Registration*** and ***Authentication***. The steps to register and authenticate identities have been explained in the subsequent sections.

Description of FIDO components

The following components and their description assist in understanding the FIDO registration and authentication flows.

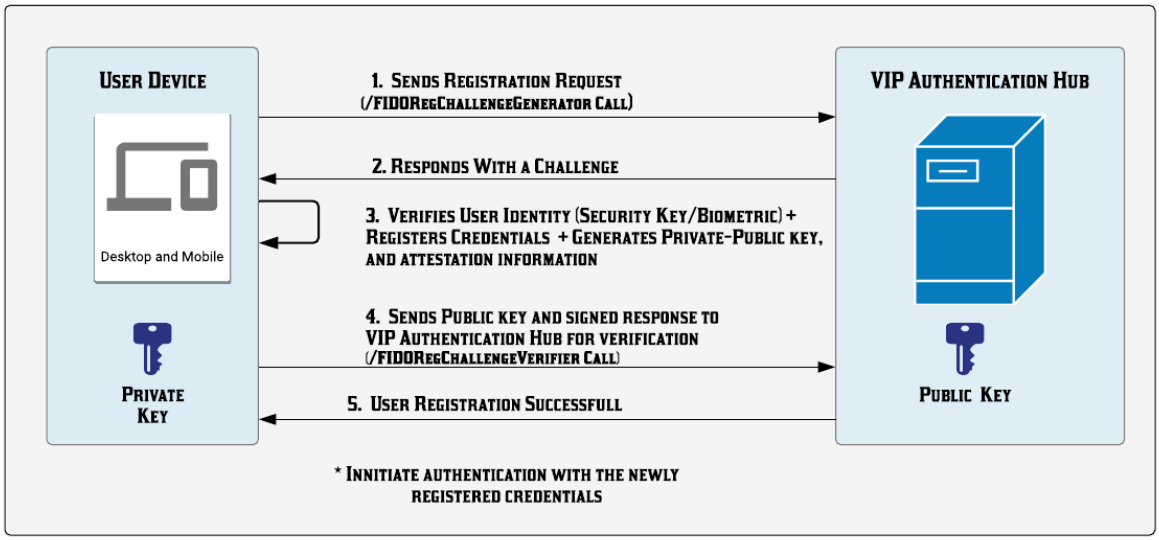
- **Fido Authenticators** - Facilitates login attempts to a user-enrolled application against the authenticator.
- **User Device** - Device from where the FIDO client operates, and from where the user initiates an action.
- **Application** - A software application that works on the FIDO2 protocol. It provides an interface on the FIDO user device to select an authenticator from the available list. Supported FIDO clients are Web, Android, and iOS.
- **IDSP** - Entity that uses the FIDO2 protocol to directly authenticate users. It manages the public keys and verifies the signatures that are created on the device.

How Security Key, Biometric and Passkey *Registration* Works?

Follow these steps to register your device with Security Key and Biometric authentication factor:

1. The user initiates registration of their account with IDSP by invoking the "/FIDORegChallengeGenerator" API.
2. IDSP generates a registration challenge and sends it back to the user's device to initiate the registration.
3. The user is prompted to select an available FIDO/Security key authenticator.
4. The user provides the user verification input based on the selected authenticator.
5. The user device verifies the user identity, registers credentials, and generates a unique public-private key pair and attestation information.
6. The application sends the public key and signed challenge generated by the FIDO (Biometric/Security Key/Passkey) authenticator to the IDSP by invoking the "/FIDORegChallengeVerifier" API. These user details are saved in the database.
7. IDSP registers the user successfully with the FIDO (Biometric/Security Key/Passkey) authenticator.

The steps to register a user with a FIDO (Biometric/Security Key/Passkey) credential completes here.

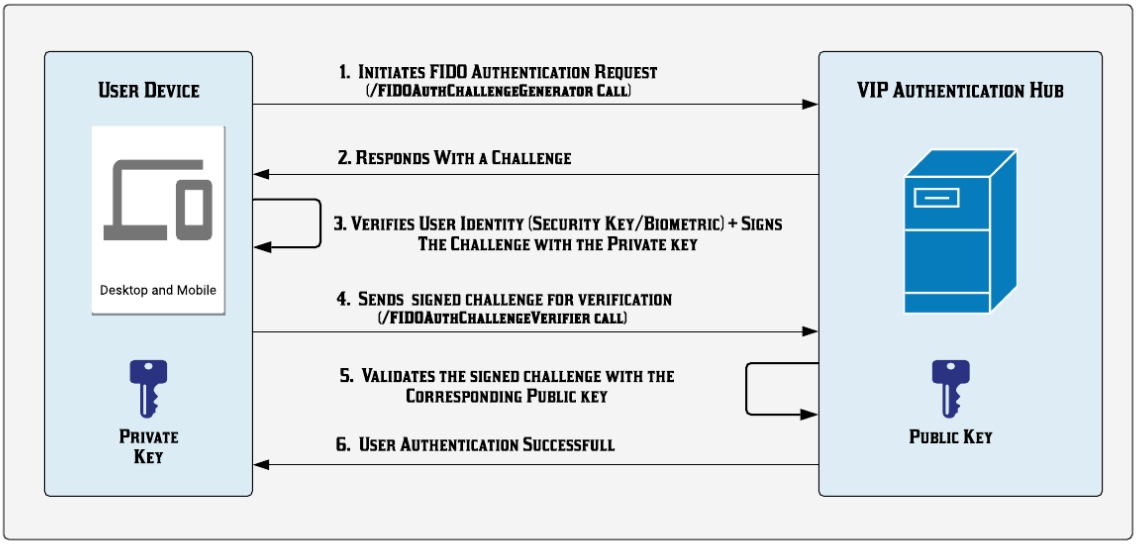


How Security Key, Biometric and Passkey Authentication Works?

Follow these steps to authenticate your account:

1. The user initiates an account authentication request with by invoking the "FIDOAuthChallengeGenerator" API.
2. IDSP responds back with a challenge.
3. User is prompted to select an authenticator and provide their verification input.
4. The user device verifies the user identity, and signs the challenge with the private key sends it back to the IDSP by invoking the "/FIDOAuthChallengeVerifier" API.
5. IDSP validates the signed challenge with the corresponding public key that was received during Security Key/ Biometric/Passkey registration.
6. Upon successful validation of the signed challenge, the user is authenticated.

The steps to authenticate a user with a FIDO (Biometric/Security Key/Passkey) credential completes here.



NOTE

- As a client, if you have registered to the FIDO-Biometric/Passkey factor authenticator using either a browser or a mobile device, the "/FIDORegChallengeVerifier" API returns a key-value pair called, "fidoIdentifier". When you pass the "fidoidentifier" value in the "authContext" attribute of the "/authenticate" call, it checks for any prior FIDO registration or authentication records made by the browser or mobile device. The fidoIdentifier attribute supports only a single key-value pair. Pass the value in this format:

```
"authContext": [
        {
            "key": "fidoIdentifier",
            "value": "<encrypted-cred-id>"
        }
    ]
```

  Refer to the IDSP cookies page for information on the Fido Identifier cookie, **__Secure-fid-XXXX**.
- The FIDO-Biometric/Passkey factor authentication functions with or without the "fidoIdentifier", therefore, it is not mandatory to pass the "fidoIdentifier" in the "/authenticate" request. However, if a user's device is not already registered with the factor, and the "isInlineEnrollmentAllowed" setting is configured as true, than a new registration will be prompted. See Registering multiple FIDO credentials.
- Users are allowed to register multiple devices with the FIDO-Biometric/Passkey authenticator but they are not allowed to reregister the same device multiple times.
- FIDO-Biometric Authentication is supported in Incognito/Private mode.

Configuring Tenant Settings for Platform and Cross-Platform Authenticators

To know about the various **tenant settings that must be configured to enable Security Key and Biometric authentication factors**, and to know about the **Recommended Passkey Configurations for Enhanced User Experience**, see here.

Security Key, Biometric, and Passkey Authentication APIs

To know about APIs that can be invoked by a user to register and authenticate user identities using Security Key, Biometric, and Passkey factors and API examples, see here.

Security Key/Biometric Platform and Browser Support Matrix

The following tables show the supported FIDO (platform and roaming) authenticator types on **Web and Mobile browsers**.

Web Browsers supported by Authenticator Types

The following table shows the supported FIDO (platform and roaming) authenticator types on different **Web browsers in Windows and MAC**:

FIDO Authenticator Type (Includes Biometric and Roaming Authenticators)	Windows (Chrome & Firefox)	MAC (Safari, Chrome, & Firefox)
Touch ID/Fingerprint/Face ID	Windows Hello Fingerprint • Chrome Version 137.0.7151.122 (Official Build) (64-bit) • Firefox 119.0.1 (64 bit) Windows Hello Face • Chrome Version 135.0.7049.85 (Official Build) • Firefox 119.0.1 (64 bit)	Touch ID/Fingerprint • Safari Version 18.3 (20620.2.4.11.5) • MAC Google Chrome Version 135.0.7049.85 (arm64) • Not supported in Firefox Face ID - Not Applicable
USB Security Key	Chrome Version 137.0.7151.122 (Official Build) (64-bit) Firefox 119.0.1 (64 bit)	MAC Google Chrome Version 135.0.7049.85 (arm64) Firefox 112.0.2 (64-bit)
Bluetooth Security Key	Chrome Version 137.0.7151.122 (Official Build) (64-bit) Firefox 119.0.1 (64 bit)	Not Certified

Reference Links to:

- Factor Swagger documentation
- Factor Error Code
- Tenant Settings
- Using Passkey

Configuring Tenant Settings for Platform and Cross-Platform Authenticators

Manage tenant settings for Security Key & Biometric authentication, covering device limits, FIDO expiry, QR code skipping, and recommended Passkey configurations for optimal user experience.

This topic includes the following sections:

- Configuring Tenant Settings for Security Key, or Biometric authentication factors
- Recommended Passkey Configurations for Enhanced User Experience

Configuring Tenant Settings for Security Key, or FIDO Biometric authentication factors

You must configure these tenant-level settings in the solution for the **Security Key** or **FIDO Biometric authentication** factors to work:

Property Name	Property Scope	Property Description
maxBiometricDevicesPerUser	Tenant	This property represents the maximum n can be maintained for biometric authenti Symantec Identity Security Platform data

fidoIdentifierExpiryDays	Tenant	This property governs the number of day valid in the browser cache or mobile data
skipQRCodeForSecurityKeyAuthentication • This property applies exclusively to **Security Keys**. • To enforce **Security Key authentication** over **QR code scanning**, set this property to **true**.	Tenant	• In case of **Registration**, if this proper users can register a **security key** as while the server still displays the **QR** option. However, if you select **QR co** during registration, the process comp errors, but authentication ultimately fa • In case of **Authentication**, if this pro skips the QR code scan and waits for plugged in. If the operation is cancele

These settings are also covered in the Managing Settings page.

Recommended Passkey Configurations for Enhanced User Experience

For streamlined Passkey registration and authentication that prioritizes a specific authenticator and avoids errors from invalid tenant setting combinations, refer to the recommended settings in the following table:

Use Case 1	**Passkey Registration Tenant Settings**	**Recommended Value**
Let user use synced passkeys on the platform	passkeyRegistrationAuthenticatorAttachment	"platform "
	passkeyRegistrationHints	" "
	Passkey Authentication Tenant Settings	**Recommended Value**
	passkeyAuthenticationHints	" "
	passkeyAuthenticationTransports	"internal,hybrid"

Use Case 2	**Passkey Registration Tenant Settings**	**Recommended Value**
Enforce Security Key for Authentication	passkeyRegistrationAuthenticatorAttachment	"cross-platform"
	passkeyRegistrationHints	"security-key"
	Passkey Authentication Tenant Settings	**Recommended Value**
	passkeyAuthenticationHints	"security-key"
	passkeyAuthenticationTransports	"usb,ble,nfc"

Use Case 3	**Passkey Registration Tenant Settings**	**Recommended Value**
Enforce passkey QR code for Authentication (roaming passkey using user mobile)	passkeyRegistrationAuthenticatorAttachment	"cross-platform"
	passkeyRegistrationHints	"client-device"
	Passkey Authentication Tenant Settings	**Recommended Value**
	passkeyAuthenticationHints	"client-device"

	passkeyAuthenticationTransports	"hybrid"

Use Case 4	**Passkey Registration Tenant Settings**	**Recommended Value**
Let user choose as per their convenience	passkeyRegistrationAuthenticatorAttachment	" "
	passkeyRegistrationHints	" "
	Passkey Authentication Tenant Settings	**Recommended Value**
	passkeyAuthenticationHints	" "
	passkeyAuthenticationTransports	"internal,usb,ble,nfc,hybrid"

These settings are also covered in the Managing Settings page.

IMPORTANT

Hints Prioritization in WebAuthn

When dealing with WebAuthn in Google Chrome, understanding how authentication hints are prioritized is key.

- Currently, in Chrome, the "authenticatorAttachment" parameter is respected as the decisive factor, regardless of other hints. This means it primarily dictates which authenticator is suggested. However, this behavior is expected to evolve; future Chrome versions are likely to make public key credential hints the preferred and sole approach for authenticator selection.
- It is also important to note that these hints might not be honored on all platforms, particularly on Windows 11 when using Chrome or Edge. This is because the authentication user interface (such as Windows Hello or Windows Security) is controlled by the operating system itself, not the browser.
- Hints can contradict information contained in "authenticatorAttachment" and credential "transports". When this occurs, hints take precedence.

Platform and Cross-Platform Authenticator Authentication APIs

Explore the APIs that are available for Security Key, Biometric, and Passkey Authentication, with examples for both user registration and authentication.

This topic includes the following sections:

- Authentication APIs
- Example of User Registration using ".../factor/v1//FIDORegChallengeGenerator" API
- Example of User Registration Verification using ".../factor/v1/FIDORegChallengeVerifier" API
- Example of User Authentication Challenge Generator using ".../factor/v1/FIDOAuthChallengeGenerator" API
- Example of User Authentication Verification using ".../factor/v1/FIDOAuthChallengeVerifier" API

Authentication APIs

The following table lists the APIs that can be used by a user to register and authenticate user identities using Security Key, Biometric, and Passkey factors.

Operation Description	Operation Type	Endpoint URLs
This API generates a challenge for user registration.	POST	https://${SSP_FQDN}/<tenantName>/factor/v1/FIDORegChall
This API generates a challenge for user registration verification.	POST	http://${SSP_FQDN}/<tenantName>/factor/v1/FIDORegChalle

This API generates a challenge for user authentication.	POST	https://${SSP_FQDN}/<tenantName>/factor/v1/FIDOAuthChal
This API generates a challenge for user authentication verification.	POST	http://${SSP_FQDN}/<tenantName>/factor/v1/FIDOAuthChalle

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring Symantec Identity Security Platform Environment Variables.
- The default value for `<tenantName>` is "default".
- The notation <text> in the above endpoint implies the text between and including < > characters must be replaced with appropriate values. For example - In this endpoint - **https://${SSP_FQDN}/<tenantName>/factor/v1/FIDORegChallengeGenerator,** if you replace the <text> with values, it would look like this- https://iam.example.com/default/factor/v1/FIDORegChallengeGenerator
- To know more about the scopes that are supported by these endpoints, refer to the Rest APIs and Associated Privileges section.

Example of User Registration using ".../factor/v1//FIDORegChallengeGenerator" API

Request Payload

The following is a sample of a user registration request payload that uses "FIDO" as the "fidoCredType":

```
{
"documentDomain":"https://www.example.com",
"userName": "nbruce",
"deviceName": "test1",
"fidoCredType" : "FIDO"
}
```

Where,

- documentDomain- The domain portion of the current document's origin. For example, "documentDomain"="https://www.voonair.com"
- userName- The name of the user to whom this credential is assigned in IDSP.
- deviceName- The name assigned to the device in IDSP during credential registration. For example- "deviceName": "nbruce's Security Key"
- fidoCredType- This is the value of the authenticator type used. The supported values are "**FIDO**", "**SECURITYKEY**" or "**PASSKEY".**

  NOTE
  If fidoCredType is not passed, then "FIDO" will be treated as the default.

Response Payload

The following is a sample of a ".../factor/v1/FIDORegChallengeGenerator" API response payload that uses "FIDO" as the cred type:

```
{
 "flowState": "eyJraWQiOiJFbmNyeXB0aW9uS2V5IiwiZW5jIjoiQTI1NkdDTSIs.....",
 "nextaction": "FIDO_REGISTER_VERIFY_CHALLENGE",
 "currentFactors": [
     "FIDO"
 ],
 "data": {
     "challenge": "opGSdUjuivSV-PD3vuD7f9iMbjPR4otlk_Q0ju7-L9s",
     "rp": {
```

```
            "name": " broadcom.com ",
            "id": " broadcom.com"
        },
        "user": {
            "id": "hxz5awwjHEWQriM9plxgD9pmSLjix4u3PHFoJb63y4w",
            "name": "nbruce",
            "displayName": "nbruce"
        },
        "attestation": "none",
        "pubKeyCredParams": [
            {
                "type": "public-key",
                "alg": -7
            },
            {
                "type": "public-key",
                "alg": -257
            }
        ],
        "timeout": 120000,
        "authenticatorSelection": {
            "authenticatorAttachment": "platform",
            "userVerification": "preferred",
            "requireResidentKey": false
        },
        "excludeCredentials": [],
        "status": "ok",
        "errorMessage": ""
    },
    "additional": {
        "currentFactorLevel": 1
    }
```


Where,

- If you choose "**FIDO**" as the "fidoCredType" in the request payload, the value of the property "**authenticatorAttachment**" will be displayed as "**platform**," and if you choose "**SECURITYKEY"**, it is presented as "**cross-platform**."
- The **timeout** attribute specifies the time (in milliseconds) that the caller (application) is willing to wait for the call to complete. The default timeout value is 120000 milliseconds (2 minutes). Ensure that you provide the user gesture for the biometric or security key within the timeout value.
- In case of "**PASSKEY**", the "**authenticatorAttachment"** attribute controls the user experience on the device on which passkey is being registered based on a tenant setting. For more information on this tenant setting, see Recommended Passkey Configurations for Enhanced User Experience.

Example of User Registration Verification using ".../factor/v1/FIDORegChallengeVerifier" API

Request Payload

The following is a sample of a user registration verification request payload:


```
{
"rawId": "eDiJuQwnvzPWDEfmVGg1IlRmVG7ckqwOxdouTOIieu0j7v1NanQ1ZLDT6b607WoJKpKv7asNt3PWV05F2poX6g",
"response": {
          "attestationObject": "o2NmbXRmcGFja2VkZ2F0dFN0bXSjY2FsZyZjc2lnWEgwRgIhAKTCp.......",
```

```
          "transports": [
                      "hybrid",
                      "internal"
                ],
          "clientDataJSON": "eyJjaGFsbGVuZ2UiOiJ0Sm1WM18xVGVQcVQtQU5JVmxCM1JxWk1LWmJfOV9J......."
            },
 "getClientExtensionResults": {},
 "id": "eDiJuQwnvzPWDEfmVGg1IlRmVG7ckqwOxdouTOIieu0j7v1NanQ1ZLDT6b607WoJKpKv7asNt3PWV05F2poX6g",
 "type": "public-key"
 "fidoCredType":"FIDO"
 }
```

Where,

- rawId- Returns the ArrayBuffer contained in the id's internal slot (the credential identifier).
- response- Contains the credential public key and metadata which can be used by the Relying Party to assess the characteristics of the credential:
  - transports: a method of the AuthenticatorAttestationResponse interface that returns an array containing strings describing the different transports which may be used by the credential. Such transports may be USB, NFC, BLE, or internal (the credential cannot be removed from the device).
  - clientDataJSON: This attribute contains a JSON serialization of the client data passed to the credential by the client in its call to either create() or get().
  - attestationObject: This attribute contains an encrypted attestation object that is unreadable by the client. The attestation object contains authenticator data (the AAGUID, a unique credential ID, and the credential public key) and an attestation statement. The contents of the attestation statement are determined by the attestation statement format used by the credential.

    The attestation statement also contains additional information that the Relying Party's server requires to validate. The Relying Party also decodes and validates the credential data, along with the JSON-serialized client data.
- getClientExtensionResults: If any extensions were requested, this method returns the results of processing those extensions.
- id: The unique identifier assigned to the credential at manufacture.
- type: The algorithm type supported by the Relying Party(origin).
- fidoCredType: This is the value of the authenticator type used. The supported values are "**FIDO**", "**SECURITYKEY**" and "**PASSKEY**".

Response Payload

The following is a sample of a ".../factor/v1/FIDORegChallengeVerifier" API response payload:

```
{
 "flowState": "eyJraWQiOiJFbmNyeXB0aW9uS2V5Iiwi.....",
 "nextaction": "FIDO_AUTH_GENERATE_CHALLENGE",
 "currentFactors": [
     "FIDO"
 ],
 "data": {
     "status": "ok",
     "fidoIdentifier": "jFJPrmp4fBi+dG9AaS1QZzMc9z2MBHagHv/TPvBU3jpabxABcWsUWYnigfx8J9xa",
     "fidoIdentifierExpiryDays": "365",
     "authenticatorAaguid": "994073e3-d50b-4cd1-9aea-63b300252893"
 },
 "additional": {
     "currentFactorLevel": 1
```

```
  }
 }
```

Where,

- The data parameter includes the following attributes:
  - status- This indicates whether the registration or authentication process was successful or unsuccessful. "OK" indicates success, while an error message indicates failure.
  - fidoIdentifier- This is the identifier that indicates if this device was used for FIDO biometric authentication previously.
  - fidoIdentifierExpiryDays- This is a tenant-level setting that governs the number of days a fido Identifier is valid in the browser cache or mobile DB.
  - authenticatorAaguid- This attribute value is returned when the tenant setting, "**enableFIDOMetadataAttestation**" is enabled during IDSP deployment.

    NOTE
    The attributes "fidoIdentifier" and "fidoIdentifierExpiryDays" appear when you select "FIDO" as the fidoCredType. These attributes do not appear for "SECURITYKEY".

Example of User Authentication Challenge Generator using ".../factor/v1/FIDOAuthChallengeGenerator" API

Request Payload

The following is a sample of a user authentication challenge generator request payload that uses "FIDO" as the "fidoCredType":

```
{
"documentDomain":"https://www.example.com",
"userName": "nbruce",
"fidoCredType" : "FIDO"
}
```

Where,

- documentDomain- The domain portion of the current document's origin. For example, "documentDomain"="https://www.voonair.com"
- userName- The name of the user to whom this credential is assigned in IDSP.
- fidoCredType- This is the value of the authenticator type used. The supported values are "**FIDO**", "**SECURITYKEY**" or **PASSKEY**.

  NOTE
  If fidoCredType is not passed, then the default will be treated as "FIDO".

Response Payload

The following is a sample of a ".../factor/v1/FIDOAuthChallengeGenerator" API response payload:

```
{
 "flowState": "N2ZZYU1pdnFYOC8zS1dDK2NxYlR1Kzc3M.........",
 "nextaction": "FIDO_AUTH_VERIFY_CHALLENGE",
 "currentFactors": [
     "FIDO"
 ],
 "data": {
     "challenge": "K6dYTQDNGRJ1OzzyZdPsO6P25kUqqjtaj0llpaBKdJU",
     "timeout": 120000,
     "rpId": "broadcom.com",
     "allowCredentials": [
         {
             "type": "public-key",
```

```
            "transports": [
                "internal"
            ],
            "id": "TsxuQRLyRzalzpDFzVkrGw"
        }
    ],
    "userVerification": "preferred",
    "isInlineEnrollmentAllowed": "true"
},
"additional": {
    "idpName": "TestConfig",
    "idpGuid": "1946ffb1-9881-4f9e-b8dc-5295c9714117",
    "idpType": "ldap",
    "currentFactorLevel": 1,
    "lastObligation": "FIDO",
    "nextObligation": "FIDO"
}
}
```


Where,

- if you select "**FIDO**" as the **fidoCredType**, the value for "transports" is returned as ["internal"], if you select "SECURITYKEY" as the fidoCredType, the value is returned as ["usb","nfc","ble"] and if you select "PASSKEY" as the fidoCredType, the value for "transports" is returned as ["internal","hybrid"]
- The **timeout** attribute specifies the time (in milliseconds) that the caller (application) is willing to wait for the call to complete. The default timeout value is 120000 milliseconds (2 minutes). Ensure that you provide the user gesture for the biometric or security key within the timeout value.
- When "**isInlineEnrollmentAllowed**" setting is configured as true, then the user is provided with the register option in case their device is new.
- In case of **Passkey**, the "**transports**" attribute specifies how an authenticator communicates with a client. In "allowCredentials", the "**transports**" attribute values determine passkey authentication user experience. Supported values for "**transports**" are:
  - **"[internal, hybrid]"**: Allows passkeys from platform authenticators (for example, Face ID, Windows Hello) or cross-platform authentication.
  - **"[internal]"**: Restricts passkeys to platform authenticators only, (for example, Face ID, Windows Hello).
  - **No "transports" property**: Default behavior with no restrictions.

  To give users the choice between platform-based or roaming passkeys, "transports" attribute should be set to ["internal","hybrid"]. This allows authentication with a passkey on the device, or displays a QR code for authentication if no passkey is present.
  The "**transports**" object controls the user experience on the device on which passkey is being authenticated based on few tenant settings. For more information on this tenant setting, see Recommended Passkey Configurations for Enhanced User Experience.

Example of User Authentication Verification using ".../factor/v1/FIDOAuthChallengeVerifier" API

Request Payload

The following is a sample of a user authentication verification request payload:


```
{
"rawId": "eDiJuQwnvzPWDEfmVGg1IlRmVG7ckqwOxdouTOIieu0j7v1NanQ1ZLDT6b607WoJKpKv7asNt3PWV05F2poX6g",
"response": {
   "authenticatorData": "SZYN5YgOjGh0NBcPZHZgW4_krrmihjLHmVzzuoMdl2MBAAAAVw",
```

```
    "signature": "MEUCIQC-_Yx6ADeqNVl7bTJ9LVO4RLol54Gk-JLateVL5YjR8gIgJxohXtPCNRE7d61cDVCEColck.....",
    "userHandle": "",
    "clientDataJSON": "eyJjaGFsbGVuZ2UiOiI4VnlJUEp3Yy1ablljMF9LaTZNeUJicjBDNmVrZ1pGT2JtcGY2czB0R2JFIiwib3....."
      },
 "getClientExtensionResults": {},
 "id": "eDiJuQwnvzPWDEfmVGg1IlRmVG7ckqwOxdouTOIieu0j7v1NanQ1ZLDT6b607WoJKpKv7asNt3PWV05F2......",
 "type": "public-key"
 }
```


Where,

- rawId- Returns the ArrayBuffer contained in the id's internal slot (the credential identifier).
- response- Contains the credential public key and metadata which can be used by the Relying Party to assess the characteristics of the credential:
  - getTransports: a method of the AuthenticatorAttestationResponse interface that returns an array containing strings describing the different transports which may be used by the credential. Such transports may be USB, NFC, BLE, or internal, hybrid (the credential cannot be removed from the device).
  - clientDataJSON: This attribute contains a JSON serialization of the client data passed to the credential by the client in its call to either create() or get().
  - attestationObject: This attribute contains an encrypted attestation object that is unreadable by the client. The attestation object contains authenticator data (the AAGUID, a unique credential ID, and the credential public key) and an attestation statement. The contents of the attestation statement are determined by the attestation statement format used by the credential.

    The attestation statement also contains additional information that the Relying Party's server requires to validate. The Relying Party also decodes and validates the credential data, along with the JSON-serialized client data.
- getClientExtensionResults: If any extensions were requested, this method returns the results of processing those extensions.
- id: The unique identifier assigned to the credential at manufacture.
- type: The algorithm type supported by the Relying Party.

Response Payload

The following is a sample of a user authentication verification response payload:


```
{
 "flowState": "eyJraWQiOiJFbmNyeXB0aW9uS2V5IiwiZW5jIjoiQTI1Nkd.....",
 "nextaction": "AUTH_ALLOWED",
 "data": {
     "status": "ok"
 },
 "id_token": "eyJ4NXQjUzI1NiI6Ikx2Wk93dGJCOW9ydGMwazdtYTN......",
 "additional": {
     "authLevelsData": {
         "67b5cea4-54ba-4341-b491-f14b4ca3144a+nbruce+1": "PASSWORD",
         "67b5cea4-54ba-4341-b491-f14b4ca3144a+nbruce+2": "FIDO"
     },
     "currentFactorLevel": 1
 }
         }
```


Using Passkey

Passkey is a type of authentication technique that does away with passwords and makes it quicker, simpler, and safer for users to access secured services.

Passkey is a passwordless authentication method that offers a faster, simpler, and more secure user experience. It uses FIDO (Fast Identity Online) standards and relies on public key cryptography and biometric authentication to verify user identity.

In the Symantec Identity Security Platform, passkeys can serve as an authentication factor. Standardized by the FIDO Alliance, a passkey comprises a public-private key pair:

- The public key is registered with the application.
- The private key is securely stored on the user's device (Apple or Android).

On Apple devices, passkeys are stored in iCloud Keychain, and on Android, in the Google Password Manager—both in encrypted form. This enables seamless sign-in across multiple devices without re-registering each device on every account.

Passkey is saved with the FQDN, which it is tied to the origin of the domain for which it was registered. This means that authentication credentials cannot be used across different domains, reducing the risk of phishing and ensuring that credentials remain confined to their intended context.

This page has the following sections:

- Prerequisites
- Registering a Passkey in IDSP Signin UI
- Authenticating Using Passkey in IDSP Signin UI
- Operating System and Browser Support
- Deleting Passkey
- Hosting Files to Sync Passkey in iOS and Android Apps

Prerequisites

The following prerequisites have to be met for using passkey in Symantec Identity Security Platform:

- iOS version 16 or above is installed.
- iOS devices should have "***iCloud Passwords and Keychain***" option enabled.
  If iCloud Keychain is not enabled, you will be prompted to activate it during registration.
- Two-Factor Authentication is enabled for the Apple ID account.

Registering a Passkey in IDSP Signin UI

Follow these steps:

1. Log in to the IDSP SiginIn UI.
2. Select the ***Passkey*** option, provide a ***Passkey name*** to identify your security key, and click ***Next***.
   "Create a passkey" option is displayed where you should choose the way you want to create a passkey.
3. Choose a method to save your passkey for the displayed FQDN, for example, nbruce-xx8-jhe1-utg3-example.com.
   – iCloud Keychain (Available on your Apple Devices).
   – iPhone, iPad, or Android device. (Save a passkey on a device with a camera)
   – Security Key (Use an external security key)
4. Assuming you chose the "iCloud Keychain" option, now scan the QR code with the device where you want to create a passkey. (This can either be done using an Apple or an Android device)
   "Save a passkey" text is displayed.
5. Click "***Save a passkey***", that initiates the creation of a passkey.
6. Use an authentication method to save the passkey, for example, FaceID or Touch ID.
   The passkey is created and saved in the app.

NOTE

- The passkey that is created in this step can be viewed in your apple device by navigating to ***Settings > Passwords***. You can see an entry with the user passkey.
- If you register a passkey through a web app and later attempt to authenticate using a mobile app, you will not be able to authenticate with the mobile/native app using the registered passkey if the "assetlink.json" (for Android) or Apple-App-Site-Association (for iOS) files are hosted on different domains for the web and mobile applications.

Authenticating Using Passkey in IDSP Signin UI

Follow these steps:

1. Log in to the IDSP Signin UI.
2. Select the ***Passkey*** option.
   Use your passkey option is displayed.
3. You are displayed with this message, "Use Touch ID to sign in". You are signed in to your FQDN with your passkey.
4. Give your consent, tap "***Continue***".
5. Authenticate using a chosen method, for example, FaceID or Touch ID.
   Upon the successful passkey authentication, you are signed in to the app.

Operating System and Browser Support

The following table shows the supported operating system and browsers for passkey authentication:

FIDO Authenticator Type	Android-version 10 or later	iOS version 15.0 or later
Passkey	Chrome Version 113	Safari- 17.3 Chrome - 129

Deleting a Passkey

If you delete a passkey from iCloud Keychain or Google Password Manager but do not remove the corresponding credential from the IDSP, you may encounter errors: "No matching passkeys found."

As a result, there is no prompt to save or register a passkey. Therefore, to re-register, or save the passkey again, you should delete the credential associated with the passkey using the Self-Service API or inform the admin to delete the credential.

Hosting Files to Sync Passkey in iOS and Android Apps

- See Hosting files for iOS Apps
- See Hosting files for Android Apps

Reference Links

- Recommended Passkey Configurations for Enhanced User Experience
- FIDO Android SDK
- FIDO iOS SDK

Hosting Files for Syncing Passkey in iOS Apps

This topic helps you to host the "***Apple-App-Site-Association"*** file which is used to sync passkey with iOS applications.

The "***Apple-App-Site-Association"*** file lets you link a website with your iOS apps.

- To link a website with your application, you must possess the corresponding domain file on your website, along with the requisite entitlement within your app. The applications listed in the "apple-app-site-association" file on your website must align with a matching Associated Domains Entitlement. For implementing passkey authentication within the native app, please refer to the hosting details of the "apple-app-site-association" file at https://developer.apple.com/documentation/xcode/supporting-associated-domains.
- The following example shows a sample, "apple-app-site-association" file for passkey authentication in the Hedmoral Sample App with Symantec Identity Security Platform FIDO server:

```
{
  "applinks": {
      "apps": [],
      "details": [
          {
              "appIDs": ["Y2CCP3S9W7.com.example.HedmoralSampleApplication"],
              "paths": [
                  "*"
              ],
              "components": [
                  {"/": "/*"}
              ]
          }
      ]
  },
  "webcredentials": {
      "apps": [
          "Y2CCP3S9W7.com.example.HedmoralSampleApplication"
      ]
  }
}
```

  where, the "appIDs" and "appKeys" within the "apple-app-site-association" file specify the application identifiers for the apps associated with this website, along with their corresponding service types.
- Use the following format for the values in these keys:

  ```
  <Application Identifier Prefix>.<Bundle Identifier>
  ```

  It is important to note that the "details" dictionary is specific to the "applinks" service type; other service types do not utilize it. Additionally, the "components" key is an array of dictionaries used for pattern matching of various components within the URL.
- Once you have constructed the association file, it should be placed in your site's ***.well-known*** directory. The URL for the file should adhere to the following format:

  ```
  https://<fully qualified domain>/.well-known/apple-app-site-association
  ```

  Importantly, you must host the file using HTTPS, with a valid certificate, and without any redirects. This ensures the proper functioning of the association between your website and associated apps.

Hosting Files for Syncing Passkey in Android Apps

This topic helps you to host the "***assetlink.json"*** file which is used to sync passkey with Android applications.

To enable passkeys to operate correctly during passkey registration and authentication with Symantec Identity Security Platform FIDO Server, host the asseltlink.json file as described in the following section.

Hosting the "*assetlink.json"* File for Android Application

The "***assetlink.json"*** file lets your website declare an association with your Android apps.

- To enable passkey support for your Android app, associate the android app with the authenticating domain, that is; the IDSP (ssp) by hosting the "***assetlink.json***" file in the following location:
  ```
  "https://<Authenticating_Domain>/.well-known/assetlinks.json
  ```
  To know more about the creation of the "assetlink.json" file, see Android Documentation.
- The "assetlink.json" file should contain the "***delegate permission***" property to delegate permissions to the login credentials at "Web" and "Android" application level.
  Sample Content of the assetlink.json file: In the following example, the "delegate permission" property is added to the assetlink.json file.

```
[ {
"relation" : [ "delegate_permission/common.get_login_creds" ],
"target" : {
  "namespace" : "web",
  "site" : "https://mgmt-eniddunn-1157-ssp.sspdev.dev.example.com"
}
}, {
"relation" : [ "delegate_permission/common.handle_all_urls", "delegate_permission/
common.get_login_creds" ],
"target" : {
  "namespace" : "android_app",
  "package_name" : "com.entersekt.fido2",
  "sha256_cert_fingerprints" :
 [ "C5:8B:E3:9B:36:B3:67:12:D7:0C:DA:C5:9D:65:2A:FC:43:9B:AE:1B:76:C9:7D:A1:7E:69:2B:7A:15:AB:27:96" ]
}
} ]
```

Fido Attestation

This topic provides an overview of FIDO Attestation, explaining its purpose in verifying authenticator authenticity, the necessity of metadata attestation for enhanced security and compliance, the structure of FIDO metadata, and the detailed process for acquiring, validating, and utilizing this metadata during user registration.

Introduction to FIDO Attestation

FIDO Attestation is a fundamental security mechanism within the WebAuthn protocol, designed to verify the genuineness and trustworthiness of an authenticator (such as a security key or biometric device) during its initial registration. It provides relying parties with verifiable information about an authenticator's origin and characteristics, ensuring that only trusted devices are registered for authentication.

At its core, attestation aims to:

- **Proof of Authenticity**: It offers evidence that a genuine and trusted device was used during the registration process.
- **Device Information**: It provides details about the device’s origin, model, and other pertinent information, allowing for policy enforcement based on authenticator capabilities.
- **Enhanced Security**: It strengthens security, especially in high-risk scenarios where the trustworthiness of the client device is crucial.

With attestation object, authenticator returns information about the newly created credential. The attestation provides assurance that the authentication process is not being tricked by a malicious device or software. Attestation gives you provable knowledge that you can then use to asses the level of assurance that device can claim, and from there on make informed decisions based on that.

FIDO Attestation is required to establish a high level of trust in the authentication process. It allows relying parties to verify that authenticators are legitimate, secure, and meet specific industry standards. This verification is often done by validating the authenticator's information against metadata from the FIDO Alliance's Metadata Service (MDS).

Thus metadata attestation provide the capability to identify authenticators and validate it against the metadata available from https://mds3.fidoalliance.org/.

Attestation helps establish trust in software by providing verifiable evidence of its origin and build process, making it harder to introduce malicious code or vulnerabilities.

Metadata Acquisition and Validation Process

1. **Download Metadata BLOB (JWT)**: The process begins by making an HTTP GET request to the FIDO Alliance Metadata Service (MDS) endpoint, which is currently https://mds3.fidoalliance.org/. The server responds with a single file, often named blob.jwt, which is a JSON Web Token.
2. **Extract Certificate Chain (x5c)**: The JWT is cryptographically signed. The first step in validation is to look inside the JWT's header for the x5c parameter. This parameter contains an array of Base64-encoded certificates that represents the signing certificate's chain.
3. **Validate Chain against FIDO Root Certificate**: Your server must have a hardcoded copy of the official FIDO MDS Root Certificate. This certificate is the ultimate "trust anchor." The server constructs a certificate path from the x5c chain and validates that it chains up to this trusted root. If it doesn't, the JWT cannot be trusted and must be rejected.
4. **Verify JWT Signature**: If the certificate chain is trusted, the server now knows the leaf certificate (the first one in the x5c array) is legitimate. It uses the public key from this leaf certificate to verify the digital signature of the JWT. This step proves two things:•The JWT was signed by the holder of the private key corresponding to the certificate.•The content of the JWT (the payload) has not been altered since it was signed.
5. **Decode and Store**: Once the signature is verified, the server can fully trust the JWT's payload. It decodes the Base64Url-encoded payload, which contains a large JSON object with a list of entries. The server parses these entries and stores the critical metadataStatement for each authenticator in a fast-access cache (like an in-memory Map), using the authenticator's AAGUID as the key.

Using Metadata for User Registration

Every time a new user or new authenticator is registered with your service, the following process takes place:

1. **Receive Attestation Object**: During a WebAuthn registration ceremony, the user's browser and authenticator creates an attestationObject, which is sent to your server. This object contains proof of the authenticator's identity.
2. **Extract AAGUID**: Your server parses the attestationObject to find the Authenticator Attestation Globally Unique Identifier (AAGUID). This ID uniquely identifies the model of the authenticator (for example, a "YubiKey 5C NFC").
3. **Look up Metadata**: The server takes the AAGUID and looks it up in the metadata cache it built during Phase 1.
4. **Validate Attestation**: If a metadata statement is found, the server uses the information within it to validate the user's attestation. For example, the metadata statement contains the specific attestation root certificates for that authenticator model. The server checks if the certificate chain in the user's attestationObject chains up to one of these trusted roots.
5. This entire flow ensures a strong chain of trust, from the FIDO Alliance all the way down to an individual user's registration, preventing the use of unvetted or fraudulent authenticators.

The FIDO metadata attestation process involves the authenticator providing attestation information (like a certificate) to the relying party (RP) during registration, which the RP then verifies against the FIDO Metadata Service. This process ensures the authenticator is trustworthy and meets the RP's security requirements.

The metadata attestation verification process involves following steps:

1. **Authenticator Registration**:
   - The user initiates registration with a relying party (that is, a website) using a FIDO authenticator.
   - The authenticator generates a new key pair (public and private).
   - The authenticator creates an attestation object, which includes:
     - The newly generated public key.
     - An attestation signature, signed by the authenticator's private key (attestation private key).
     - Optionally, a certificate or similar data to verify the public key's provenance.

2. **Attestation Statement**: The authenticator sends the attestation object to the relying party.
3. **Metadata Verification**: A FIDO server, when registering a new authenticator, will:
   – Receive the authenticator's attestation statement.
   – Look up the corresponding metadata statement from the FIDO Metadata Service. The relying party retrieves the FIDO metadata statement for the specific authenticator model from the FIDO Metadata Service (MDS).
   – The RP verifies the attestation signature against the attestation certificate(s) listed in the metadata statement. The RP may also verify that the attestation certificate chains up to a trusted issuer.
   – Check if the authenticator meets the relying party's registration policy (for example, specific AAGUID, protocol version).

Enabling Tenant Settings

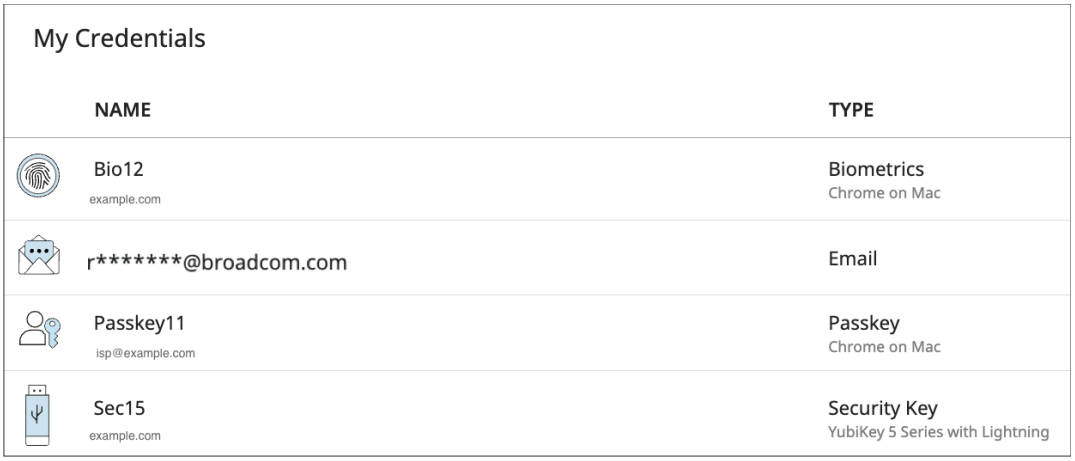
enableFIDOMetadataAttestation Setting: The enableFIDOMetadataAttestation setting controls the behavior of FIDO attestation during registration.

- Default Behavior (Value: false): By default, enableFIDOMetadataAttestation is set to false. This maintains the previous behavior for the ".../factor/v1/FIDORegChallengeGenerator" API, where the attestation type is specified as "none". In this mode, no detailed authenticator attestation data is collected or verified by the IDSP FIDO server.
- Enabled Behavior (Value: true): When enableFIDOMetadataAttestation is changed to true, the attestation type within the /factor/v1/FIDORegChallengeGenerator API is updated to "direct". This change allows the client to transmit comprehensive authenticator attestation data to the FIDO server. Consequently, the IDSP FIDO server gains the capability to perform verification of this attestation data, enhancing the trustworthiness and security of registered authenticators.

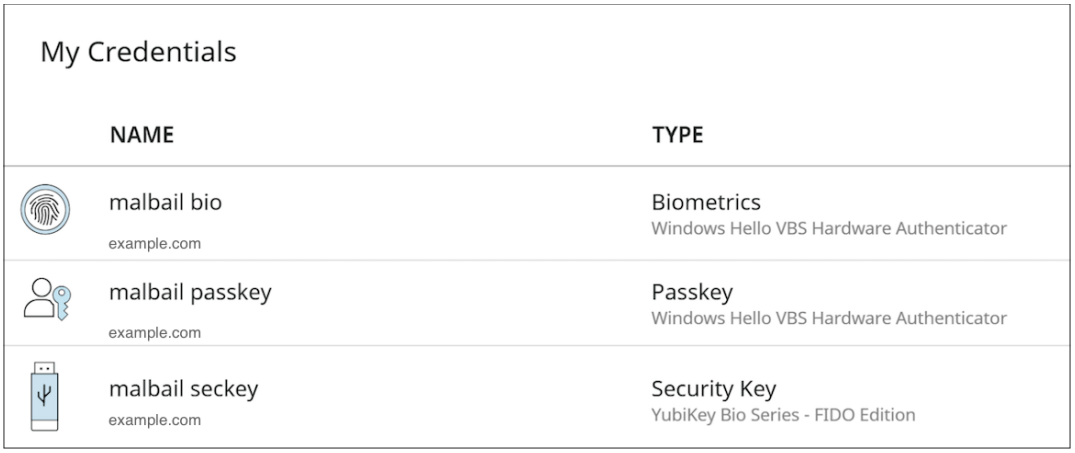
Example values shown during FIDO registration

During credential registration, when the "enableMetadataAttestation" setting is enabled, then the credentials show their identification such as the authenticator platform, browser, etc.

- FIDO metadata attestation information in Mac -



- FIDO metadata attestation information in Windows -



Reference Links

- FIDO Attestation Metadata upload API

VIP Access Push Authenticator

Learn about VIP Access Push authentication application.

VIP Access Push app sends a push notification to your mobile device when configured with Symantec Identity Security Platform. Your mobile device must be push-enabled to receive push requests. You can either click the **tick** or **cross** button to approve or deny the notification request to access your online accounts.

Push notifications are small pop-ups that alert users to information opted-in from apps and services. Symantec Identity Security Platform supports push notification as one of the two-factor authentication mechanism, where you can accept or deny a transaction by approving or rejecting a push request on a registered device.

The VIP Access Push app sends a push notification to your mobile device when configured with IDSP. Your mobile device must be push-enabled, to receive push requests. You can either click the **tick** or **cross** button to approve or deny the notification request to access your online accounts.

To enable the **Push Number Challenge** during push authentication, set the tenant-level property `"pushNumberChallengeEnabled"` to **"true"**.

This and other push related settings are covered in the Configuring Tenant Settings section, and is also covered in Manage Settings page.

This topic includes the following sections:

- Prerequisite to Register a User Device with VIP Access Push
- Configuring Tenant Settings
- Registering a User Device with VIP Access Push Credential
- Authenticating a User Device with VIP Access Push Credential

Prerequisite to Register a User Device with VIP Access Push

To get started with the VIP Access Push registration, ensure that you have installed the VIP Access Push app on to your mobile device from the play store. Once you install the app, you are issued with a credID and OTP.

Configuring Tenant Settings

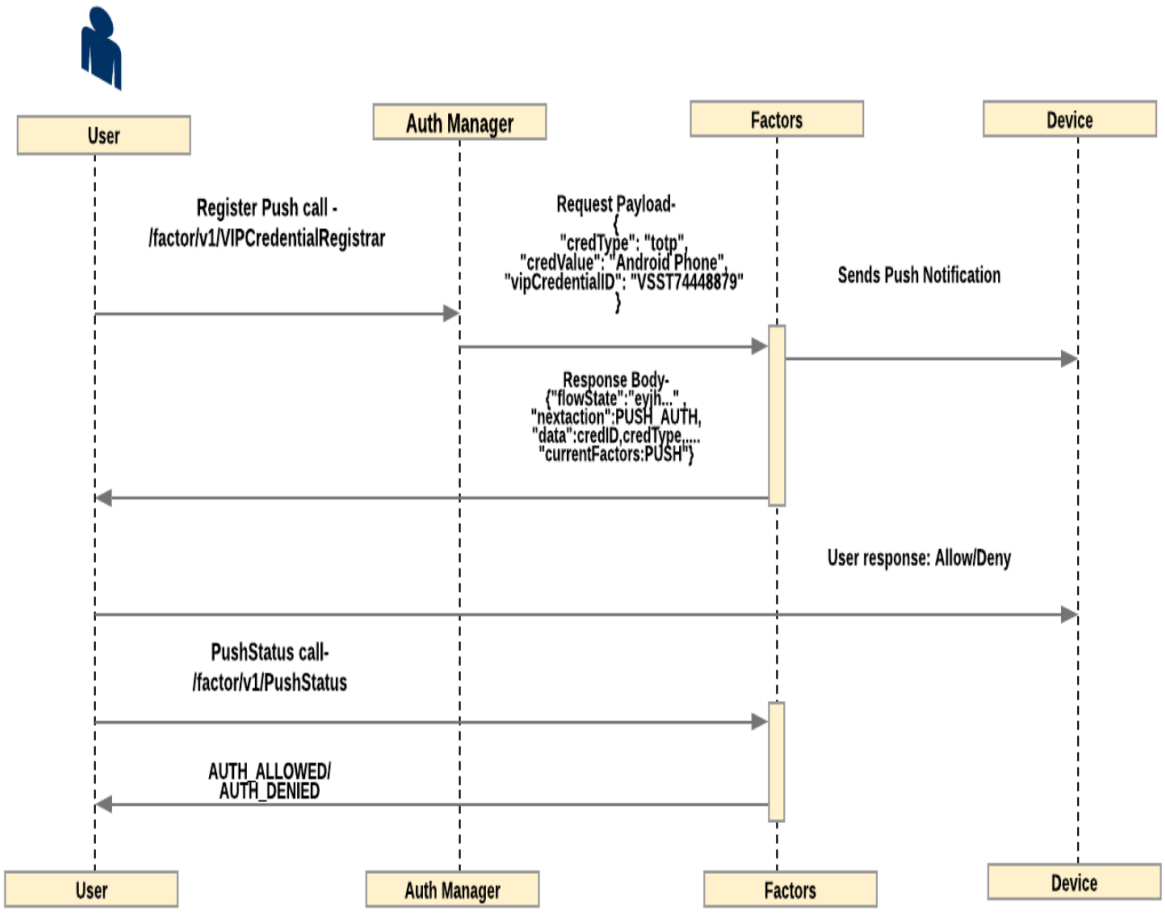
The following are the **VIP Access Push authentication** related tenant-level settings in the solution that you may want to enable or disable based on your requirements:

Property Name	Property Scope	Property Description
pushNumberChallengeEnabled	Tenant	This property manages the "Push Number "VIP Policy" through IDSP: • When this property is set to **true** , the **V** and expects a push number challenge Authentication. • When this property is set to **false** , the **V** allowing users to simply **Approve/Deny** challenge during Push Registration & A
pushIncludeDeviceInfo	Global, Tenant	This property is used to include device info app.
pushProviderToBeUsed	Global, Tenant	This property determines which push provi to the end user's device.At the tenant level, values from a list of customer providers at t push providers at the global level.At the glo possible values from a list of customer prov Push providers available in AuthHub (like v
pushRegTimeoutSecs	Global, Tenant	This property governs the registration timeo
pushEnforceLocalAuth	Global, Tenant	This property governs the timeout value for expected to respond to push notifications w
pushAuthTimeoutSecs	Global, Tenant	This property governs the timeout value for expected to respond to push notifications w

These settings are also covered in the Managing Settings page.

Registering a User Device with VIP Access Push Credential

When you are tied to a policy in (referred to as solution) that uses push as one of the two-factor obligation (authentication mode), the user is required to register a device with the solution to receive push notifications and authenticate them in order to determine the device's proof of possession.



Follow these steps to register the VIP Access Push app in a user device:

1. Initiate a "/VIPCredentialRegistrar" call to register VIP Access Push onto a device. Enter the "credType","credValue" and "vipCredentialID" in the request payload.
   where,
   - credType- Specify the type of push OTP algorithm supported by the app. Supported type is Time-based-One-Time-Password (totp_push) which means push is enabled.
   - credValue- Specify a value of the device that helps the user to easily recognize the device where the VIP Access Push software is installed.
   - vipCredentialID- Specify the credential ID that has been generated by the VIP Access app after its installation on your device.

     NOTE
     Please note that the vipCredentialID must be entered without any spaces. For example-the credID should be entered like this "VSST74448879" and not like "VSST 7444 8879".

   In response to the registration request, the solution returns the "flowState" as "IN_PROGRESS" and "nextaction" as PUSH_AUTH.
2. A push notification is sent to your device to authenticate the proof of possession of the device. Perform either of the following:
   - **Allow**- This option approves the registration and the next action returned is "PUSH_AUTH".
   - **Deny**- This option rejects the registration initiation and the next action returned is "AUTH_DENIED".

   However, in this step we have assumed that you have clicked "Allow" and have approved the registration.

NOTE
A periodic polling of the registration status (PushStatus) is done by IDSP and until the device is registered, solution returns the response as "IN_PROGRESS".

3. IDSP registers the user device with push two-factor authentication mode.
4. The next time you perform a "/PushStatus" call, the solution will return the response as "AUTHENTICATED". This indicates that the registration process has successfully completed

NOTE
If you do not complete the registration process within the timeout period, you have the option to restart the push flow again.

Steps to Register VIP Access Push Using APIs

1. Initiate the following "/VIPCredentialRegistrar" call to register push in your device.

```
https://<sspHost>/<tenantName>/factor/v1/VIPCredentialRegistrar
```

NOTE

- The notation {{text}} implies the text between and including {{}} characters must be replaced with appropriate values. For example-In this endpoint-**https://{{host}}/{{tenantName}}/factor/v1/VIPCredentialRegistrar**, if you replace the {{text}} with values, it would look like this- https://example.com/default/factor/v1/VIPCredentialRegistrar
- To know about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Request Payload

In the request payload, enter the following details:

```
{

   "credType": "totp_push",

   "credValue": "Android Phone",

   "vipCredentialID": "VSST74448879"

}
```

where,

- credType- Specify the type of push OTP algorithm supported by the app, which is Time-based-One-Time-Password (totp_push).
- credValue- Specify a name of the device that helps you to easily recognize the device the software is installed on.
- vipCredentialID- Specify the credential ID that has been generated by the VIP Access app after its installation on your device.

NOTE
Please note that the vipCredentialID must be entered without any spaces. For example-the credID should be entered like this "VSST74448879" and not like "VSST 7444 8879".

Response Payload

Upon successfully verifying the values in the request payload, the response generates the "flowState" as "IN_PROGRESS" and "nextaction" as PUSH_AUTH. By this time you must have received a push notification on your mobile for approval.

```
{

   "flowState": "eyJh.........WxzZX0=",

   "nextaction": "PUSH_AUTH",
```

```
    "data": {

        "credId": "44aef37f-e263-4241-a510-2d615b00a3a8",

        "tenantId": "4764021a-9048-11ea-bb37-0242ac130002",

        "userId": "eef7746e-175e-4a4f-8ce5-0978bd8d2ff0",

        "credType": "TOTP_PUSH",

        "credValue": "Android Phone",

        "status": "IN_PROGRESS",
        "createdDatetime": "2020-09-30 07:25 AM UTC",

        "modifiedDatetime": "2020-09-30 07:25 AM UTC",

        "pushStatusMsg": "Mobile push request sent",

        "default": false,

        "vipCredentialID": "SYMC59197959",

        "userDisplayName": "bsteer",

        "tenantName": "default"

    },

    "currentFactors": [

        "PUSH"

    ]

}
```


2. Once you approve the push request, initiate the following "/PushStatus" API to check the status of the push device registration with VIP Access App:

   ```
   https://{{sspHost}}/{{tenantName}}/factor/v1/PushStatus
   ```

   The status can be of three types: In-Progress, Authenticated or Denied.

   - **IN-PROGRESS** status is returned by the API until the device is registered. The solution returns the "nextaction" as "PUSH_AUTH".
   - **AUTHENTICATED** status is returned by the API when the you approve (selects "Allow") the push request received on the mobile device. The solution returns the response as "AUTH_ALLOWED".
   - **REJECTED** status is returned by the API when you reject (selects "Deny") the request. The solution returns the response as "AUTH_DENIED".

     NOTE
     If you do not respond to the push request before the timeout period, then the solution returns the state as IN-PROGRESS.

   PushStatus Request Payload where you enter the credId:

```
{

   "credId": "44aef37f-e263-4241-a510-2d615b00a3a8"

}
```

where,

– credId- Specify the credId that you received from the "/VIPCredentialRegistrar" response.

In this example, the response payload displays the "nextaction" as "AUTH_ALLOWED" and "status" as "AUTHENTICATED".

```
{

   "flowState": "eyJh.........fQ==",

   "nextaction": "AUTH_ALLOWED",

   "data": {

       "status": "AUTHENTICATED",

       "message": "Mobile push request approved by user"

   },

   "deviceID": "93514d51-3284-487d-95c2-d6337afe226d",

   "id_token": "eyJ4.......Efww"

}
```

The registration process is a one-time operation which is completed once the push status API returns the result as "AUTHENTICATED". Next time when a push authentication is requested to you, the push notification will be sent directly to your registered mobile device for approval.

Authenticating a User Device with Push Credential

When you receive Push_Auth as a next action for authentication and have already registered a mobile device with VIP Access app, the solution directly sends push notification to the registered device for approval.

Follow these steps to authenticate the VIP Access Push notification in a registered device:

1. Initiate a "/PushGenerator" call to receive push notification from IDSP (referred to as solution) onto the registered device. Enter the "credId" and "action" in the API request.
   In response to the authentication request, the solution returns the "flowState" as "IN_PROGRESS" and "nextaction" as PUSH_AUTH. By this time, a push notification is sent to the registered mobile for approval.
2. Approve the push notification and initiate a status call using "/PushStatus" API to display the status of push auth. Assuming that you have approved the push notification, the status returned will be "AUTHENTICATED" and "nextaction" will be "AUTH_ALLOWED".

Steps to Authenticate VIP Access Push Using APIs

1. Initiate the following "/PushGenerator" call to begin the Push authentication.

```
https://<sspHost>/<tenantName>/factor/v1/PushGenerator
```

NOTE

- The notation {{text}} implies the text between and including {{}} characters must be replaced with appropriate values. For example-In this endpoint-**https://{{host}}/{{tenantName}}/factor/v1/PushGenerator**, if you replace the {{text}} with values, it would look like this- https://example.com/default/factor/v1/PushGenerator
- To know about the scopes that are required to secure these API's, refer to the Scopes section.

PushGenerator Request Payload

In the request payload, enter the following details:

```
{
            "credId": "44aef37f-e263-4241-a510-2d615b00a3a8",
            "action": "LOGIN"
        }
```

where,

- credId- Specify the credId that has been generated by the response from the registration call.
- action- Specify the action for which the authentication is initiated. For example, - authenticate, login.

PushGenerator Response Payload

After the solution successfully verifying the values in the request payload, the response generates the "flowState" as "IN_PROGRESS" and "nextaction" as PUSH_AUTH. By this time, the push notification is sent to the registered mobile for approval.


```
{

    "flowState": "eyJh........ZX0=",

    "nextaction": "PUSH_AUTH",

    "data": {

        "credId": "44aef37f-e263-4241-a510-2d615b00a3a8",

        "tenantId": "4764021a-9048-11ea-bb37-0242ac130002",

        "userId": "eef7746e-175e-4a4f-8ce5-0978bd8d2ff0",

        "credType": "TOTP_PUSH",

        "credValue": "Android Phone",

        "status": "IN_PROGRESS",

       "createdDatetime": "2020-09-30 07:25 AM UTC",

        "modifiedDatetime": "2020-09-30 07:25 AM UTC",

        "pushStatusMsg": "Mobile push request sent",

        "default": false,

        "vipCredentialID": "SYMC59197959",

        "userDisplayName": "bsteer",
```

```
        "tenantName": "default"

    },

    "currentFactors": [

        "PUSH"

    ]

}
```

2. Once you approve the request, initiate the following "/PushStatus" API to check the status of the push registration:

```
https://{{sspHost}}/{{tenantName}}/factor/v1/PushStatus
```

The status can be of three types: In-Progress, Authenticated or Denied.

- **IN-PROGRESS** status is returned by the API until the device is registered. The solution returns the "nextaction" as "PUSH_AUTH".
- **AUTHENTICATED** status is returned by the API when you approve (selects "Allow") the push request received on the mobile. The solution returns the response as "AUTH_ALLOWED".
- **REJECTED** status is returned by the API when you reject (selects "Deny") the request. The solution returns the response as "AUTH_DENIED".

  NOTE
  If you do not respond to the push request before the timeout period, then the solution returns the state as IN-PROGRESS.

PushStatus Request Payload where you enter the credId:

```
{

    "credId": "44aef37f-e263-4241-a510-2d615b00a3a8"

}
```

where,

– credID- Specify the credId that has been returned by the "/VIPCredentialRegistrar" API.

The response payload displays the "nextaction" as "AUTH_ALLOWED" and "status" as AUTHENTICATED.

```
{

    "flowState": "eyJh.......fQ==",

    "nextaction": "AUTH_ALLOWED",

    "data": {

        "status": "AUTHENTICATED",

        "message": "Mobile push request approved by user"

    },

    "deviceID": "93514d51-3284-487d-95c2-d6337afe226d",

    "id_token": "eyJ4.....Efww"
```

```
}
```

The push authentication completes, once the "/PushStatus" API returns the result as "AUTHENTICATED".

Reference Links to:

- Factor Swagger documentation
- Factor Error Code
- Tenant Settings

OTP Authenticator

One-Time Password (OTP) is a mechanism through which a single-use password is generated and sent to your registered mobile or Email Id to access a protected resource, such as, a website. The OTP authenticator adds an additional layer of security, which otherwise could be vulnerable to phishing and other such activities.

The Symantec Identity Security Platform OTP authenticator consists of the following types:

- Mobile OTP
- VIP Access App
- Server OTP

How the IDSP OTP Authentication Works?

You must perform the following steps to get the OTP authenticator up and running:

1. ***Register*** - Using the *Register OTP API*, you can register your email Id and phone number where you wish to receive your OTP.

   NOTE
   If your Personally Identifiable Information (PII), such as phone number and/or email ID is updated in the User directory prior to performing OTP authentication, then the authentication flow skips the registration process and jump start you with generating and verifying OTP flows. This process is also known as **Auto-Enrolment**. However, if your details do not exist in the User directory, then the authentication flow directs you to follow the OTP registration process.
2. ***Generate*** - Using the *Generate OTP API,* you can generate OTP and get it delivered to your registered email or phone. The generated OTP is valid for a single login attempt.
3. ***Authenticate*** - Using the *Verify OTP API*, you can authenticate the transaction request with the OTP.

   NOTE

   The OTP authenticator is dependent on the LDAP service to operate consistently. It is therefore necessary to verify that the LDAP factor runs correctly, so that OTP authenticator works.

Reference Links to:

- Factor Swagger documentation
- Factor Error Code
- Mobile Client App (also known as Hedmoral Technologies App)

Mobile OTP Authenticator

Learn about Symantec Identity Security Platform Mobile OTP

Integrate Mobile OTP as an authentication factor for transactions. The Mobile OTP APIs facilitate the registration, generation, and verification of these one-time passwords. Symantec Identity Security Platform supports OTP generation

through the VIP Access app and other OATH-compliant authenticator apps like Google Authenticator, which are available on mobile app stores. After installation, users must register their chosen app with IDSP to enable OTP generation and complete the authentication process.

NOTE
To enhance tenant customization of the Mobile Authenticator enrollment experience, IDSP now offers a setting to configure the default enrollment mode displayed in the Signin UI. The enrollment mode now dynamically aligns with the value specified in tenant setting, "**motpDefaultEnrollmentMode**". This setting accepts two modes, "vipaccess" (default value) and "qrcode".
This setting is covered as part of tenant settings documentation.

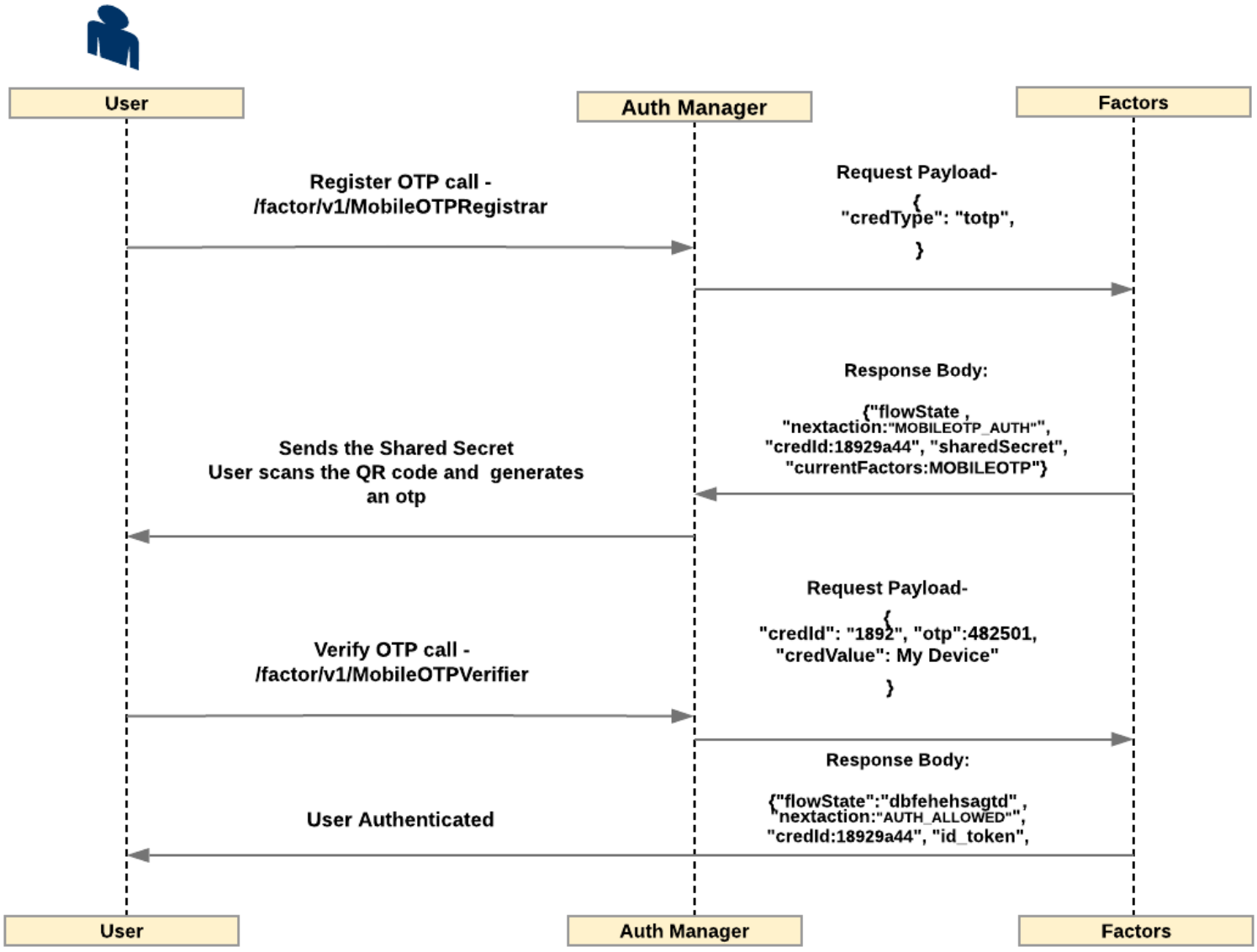
How Mobile OTP Authentication Works?

The Mobile OTP authentication process can be broken down into two flows. The steps to be performed in each of these flows are discussed here:

- Registration Flow
- Verification flow

Registration Flow

The registration of mobile OTP credentials is a one time activity. If you are using OTP authentication for the first time, you must complete the OTP registration process by providing your credentials. Once you are through the registration process, your OTP credentials are stored in the database. This allows you to generate and validate OTPs using the verification flow. The next time you login, you will be immediately prompted to generate and verify transactions using OTP.



Follow these steps to register mobile OTP:

1. Perform a "Register Mobile OTP" call using "/factor/vi/MobileOTPRegistrar" API to the Factor server. The request payload consists of "credType". For example- {"credType":"totp"}.
2. The Factor server returns the "flowState", "nextaction" as "MobileOTP_Auth", "credID" (along with other necessary details) and shared secret (includes QR Code) in its response body.
3. Download a mobile authenticator application from a third-party onto your registered mobile device.

   NOTE
   VIP Access and Google Authenticator apps are supported and these apps can be download from the play store. To know more about configuring the VIP Access app with Symantec Identity Security Platform, see VIP Configuration Settings.

4. Scan the QR code received in step 2 using the authenticator app, which in turn creates an account in the app to enable two-factor authentication.
5. Perform a "Verify Mobile OTP" call using "/factor/vi/MobileOTPVerifier" API to authenticate.

Step to Register a User Device with Mobile OTP Through REST APIs

1. Use the following MobileOTPRegistrar API to register your device with the Mobile OTP:

```
https://<sspHost>/<tenantName>/factor/v1/MobileOTPRegistrar
```

NOTE

- The notation {{text}} implies the text between and including {{}} characters must be replaced with appropriate values. For example-In this endpoint-**https://{{host}}/{{tenantName}}/factor/v1/MobileOTPRegistrar**, if you replace the {{text}} with values, it would look like this- https://iam.acme.com/default/factor/v1/MobileOTPRegistrar
- To know about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.
- The solution provides a Sample Mobile application that you can use to perform authentication operations. Click here to know how to setup the application for demo usage.

In the **request payload**, enter the following details:

```
{
"credType": "totp"
}
```

Where,

- credType- Specify the type of mobile OTP algorithm supported by the app, which is Time-based-One-Time-Password (TOTP).

Response Payload

The response generates the "flowState", "nextaction" as "MobileOTP_Auth", "credID" (along with other necessary details) and the shared secret (includes QR Code). The "credID" indicates that the device is active to receive MOTP's and indicates that the registration process has successfully completed. Although, in this case the "nextaction" is shown as "MobileOTP_Auth", it may differ based on the policy.

```
{
"flowState": "eyJh......c2V9",
"nextaction": "MOBILEOTP_AUTH",
"data": {
"credId": "18929a44-47fe-4efd-902d-46c6274fa631",
"tenantId": "830de181-89b7-4849-a3e3-6021f66660fd",
"userId": "d06974be-d944-4728-b8ab-7a88e2e61b30",
"credType": "TOTP",
"credValue": "MY MOBILE AUTH DEVICE",
"status": "INITIAL",
"createdDatetime": "2020-10-06 09:14 AM UTC",
"modifiedDatetime": "2020-10-06 09:14 AM UTC",
"default": false,
"sharedSecret": "JNBFMSKXKNCFCV2CKJCDIS2RGRMVKWKW",
"userDisplayName": "erussell",
"tenantName": "default"
},
"currentFactors": [
"MOBILEOTP"
]
}
```

2. The Factor server returns the "flowState", "nextaction" as "MobileOTP_Auth", "credID" (along with other necessary details), "status": "INITIAL", and shared secret (includes QR Code) in its response body. Since the status is still showing "initial", you must perform a verification call to complete the registration.
3. Download the authenticator app and scan the QR code received in step 2 using the authenticator app. This creates an account in the app to enable two-factor authentication.
4. Initiate the following MobileOTPVerifier API endpoint to authenticate the MOTP flow:

```
https://<sspHost>/<tenantName>/factor/v1/MobileOTPVerifier
```

NOTE

- The notation {{text}} implies the text between and including {{}} characters must be replaced with appropriate values. For example-In this endpoint-**https://{{host}}/{{tenantName}}/factor/v1/MobileOTPVerifier,** if you replace the {{text}} with values, it would look like this- https://iam.acme.com/default/factor/v1/MobileOTPVerifier
- To know about the scopes that are required to secure these API's, refer to the Scopes section.

MobileOTPVerifier Request Payload

In the **request payload**, enter the following details:

```
{
"credId": "18929a44-47fe-4efd-902d-46c6274fa631",
"otp": "482501",
"credValue" :"My Device"
}
```

Where,

- credId- Specify the credential ID that has been generated by Symantec Identity Security Platform during the registration flow.
- otp- Specify the TOTP generated by the authenticator app.
- credValue- Specify the credential value that has been generated by Symantec Identity Security Platform during the registration flow.

NOTE
Entering the credValue in the MobileOTPVerifier call is only required until the status of the credential is returned as "INITIAL". Once the status changes to "OK", it becomes optional to specify the credValue in the call.

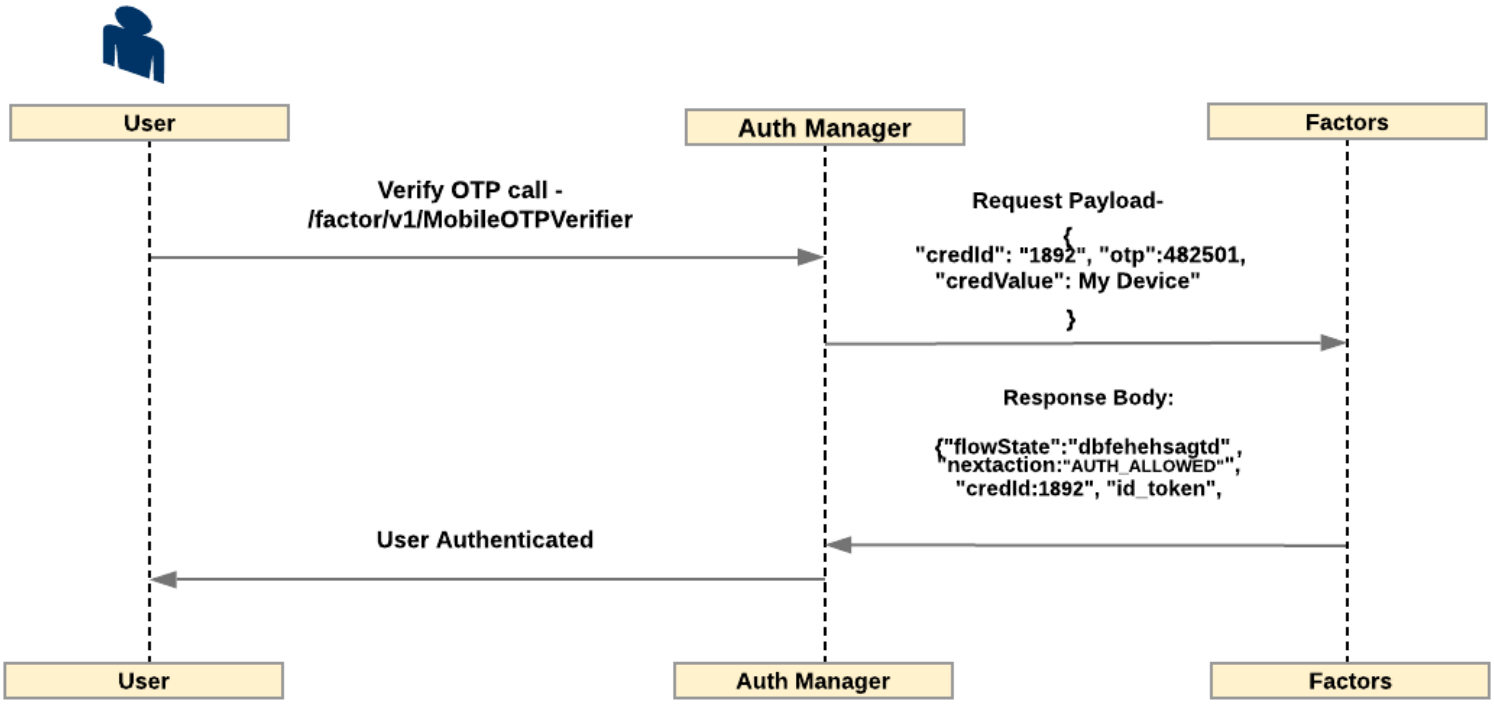
MobileOTPVerifier Response Payload

In **response** to the authentication request, the solution returns the "flowState", "nextaction" as "AUTH_ALOWED", id_token and other necessary details. However, the "nextaction" will depend upon the policy and may not always result in "Auth_Allowed" as you may have to authenticate with other factors. After you obtain the ID token, the Mobile OTP authentication flow completes successfully.

```
{
"flowState": "eyJh.....c2V9",
"nextaction": "AUTH_ALLOWED",
"data": {
"status": "OK",
"message": "One-Time Password (OTP) is verified successfully"
},
"deviceID": "529eb6fc-8694-4041-91ef-7aec07cd8310",
"id_token": "eyJ4....IVjA"
}
```

Verification Flow

If you have already registered your OTP credentials, you can skip the registration flow and start using this flow where you will verify the OTP to complete the authentication flow.



Follow these steps to verify the MOTP:

1. Perform a "Verify Mobile OTP" call, using "/factor/vi/MobileOTPVerifier" API to the Factor server. Send the "credId" which was generated in the registration flow along with the OTP that is generated by the authenticator app, and the credValue (A unique name to identify the device) in the request body. For example- {"credId: 18929a44-47fe-4efd-902d-46c6274fa631", "OTP":"23434", "credValue" :"My Device"}.
2. The Factor server verifies the "credId", and "OTP" and sends the "flowState", "nextaction" as "AUTH_ALLOWED", ID token and other necessary details to you. However, the "nextaction" will depend upon the policy and may not always result in "Auth_Allowed" as you may have to go through other factors of authentication. After you obtain the ID token, the Mobile OTP authentication flow completes successfully.

Step to Verify MOTP flow Through REST APIs

1. Initiate the following MobileOTPVerifier API endpoint to authenticate the MOTP flow:

```
https://<sspHost>/<tenantName>/factor/v1/MobileOTPVerifier
```

NOTE

- The notation {{text}} implies the text between and including {{}} characters must be replaced with appropriate values. For example-In this endpoint-**https://{{host}}/{{tenantName}}/factor/v1/MobileOTPVerifier,** if you replace the {{text}} with values, it would look like this- https://iam.acme.com/default/factor/v1/MobileOTPVerifier
- To know about the scopes that are required to secure these API's, refer to the Scopes section.

MobileOTPVerifier Request Payload

In the request payload, enter the following details:

```
{
"credId": "18929a44-47fe-4efd-902d-46c6274fa631",
"otp": "482501",
"credValue" :"My Device"
```

```
}
```

Where,

- credId- Specify the credential ID that has been generated by Symantec Identity Security Platform during the registration flow.
- otp- Specify the TOTP generated by the authenticator app.
- credValue- Specify the credential value that has been generated by Symantec Identity Security Platform during the registration flow.

> **NOTE**
> Entering the credValue in the MobileOTPVerifier call is only required until the status of the credential is returned as "INITIAL". Once the status changes to "OK", it becomes optional to specify the credValue in the call.

MobileOTPVerifier Response Payload

In response to the authentication request, the solution returns the "flowState", "nextaction" as "AUTH_ALOWED", id_token and other necessary details. However, the "nextaction" will depend upon the policy and may not always result in "Auth_Allowed" as you may have to authenticate with other factors. After you obtain the ID token, the Mobile OTP authentication flow completes successfully.

```
{
"flowState": "eyJh.....sc2V9",
"nextaction": "AUTH_ALLOWED",
"data": {
"status": "OK",
"message": "One-Time Password (OTP) is verified successfully"
},
"deviceID": "529eb6fc-8694-4041-91ef-7aec07cd8310",
"id_token": "eyJ4.....VjA"
}
```

To view the request and response payload for these URLs, refer to Factor Swagger documentation

Reference Links to:

- Factor Error Codes

VIP Access Authenticator (For Mobile and Desktop)

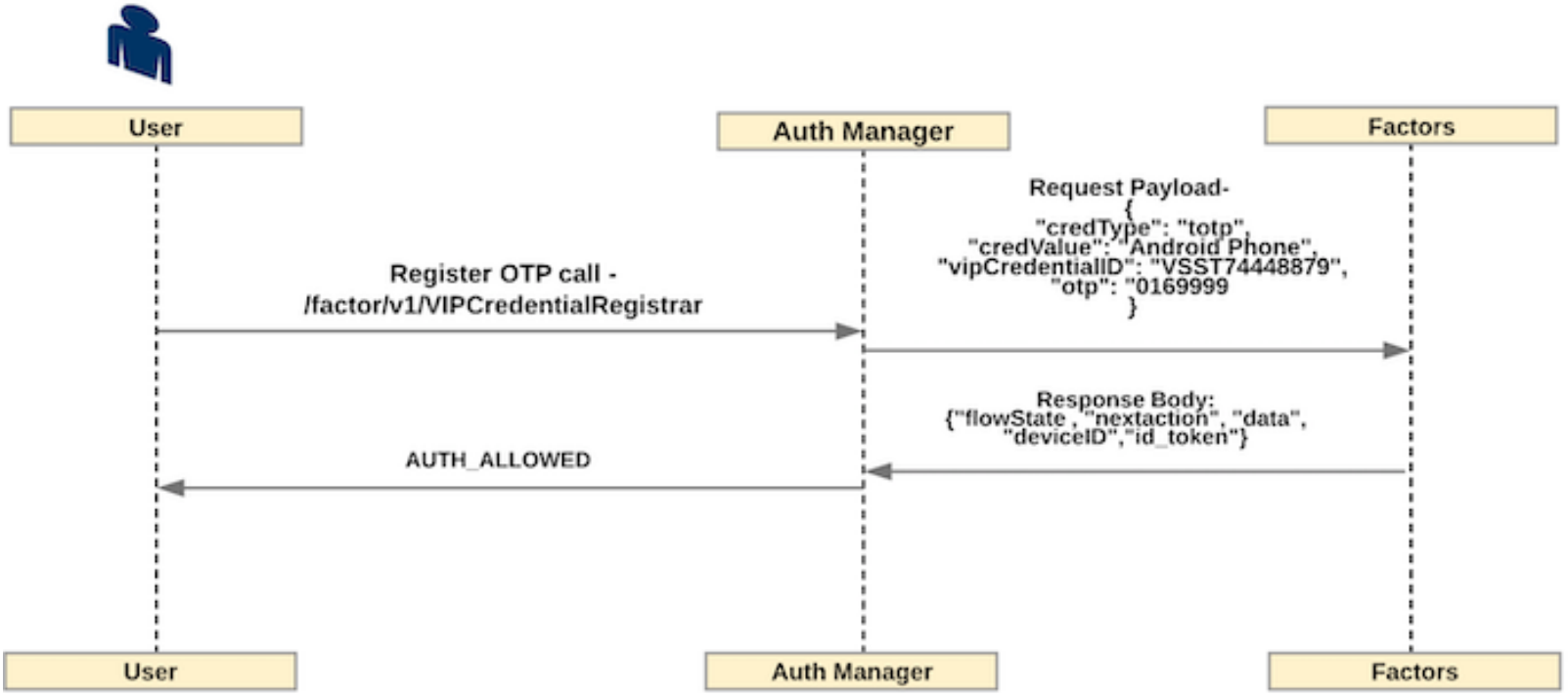
VIP Access application helps protect your online accounts and transactions by using a strong authentication process when you sign in to your VIP-enabled accounts.

VIP Access App is used to authenticate online transactions through TOTP and is available in the play store for installation. The app has a mobile and a desktop version and upon installation, it creates a credential ID for that specific installation and also generates OTP. The OTP's keep refreshing based on the timeout period and new ones get generated.

> **NOTE**
> When you install the VIP Access App in your mobile, the Mobile OTP and Push capabilities are activated for that installation by default. However, if you install the VIP Access desktop app, it will only allow you to execute Mobile OTP capabilities and not allow you to execute Push. To use Push flows, you must install and register with the VIP Access Mobile App only.

Registration Flow in VIP Access App

Registration of mobile OTP credentials in VIP Access is a one-time operation. If you are using OTP authentication for the first time, you must complete the OTP registration process by providing your credentials. Once you are through the registration process, your OTP credentials are stored in the user directory. This allows you to generate and validate OTPs using the verification flow. The next time you login, you will be immediately prompted to generate and verify transactions using OTP.



Follow this step to register VIP Access app in your device:

1. Initiate a "/VIPCredentialRegistrar" call to register VIP Access App onto your device. Enter the "credType","credValue", "vipCredentialID" and "otp" in the request payload.
   where,
   - credType- Specify the type of push OTP algorithm supported by the app. Supported type is Time-based-One-Time-Password (TOTP) and TOTP_PUSH.
   - credValue- Specify a value/name of the device to help you easily recognize the device where the VIP Access Push software is installed.
   - vipCredentialID- Specify the credential ID that has been generated by the VIP Access app after its installation on your device.

     NOTE
     Note that the vipCredentialID must be entered without any spaces. For example-the credID should be entered like this "VSST74448879" and not like "VSST 7444 8879".
   - otp- Specify the otp generated by the VIP Access app.

   In response to the registration request, the solution returns the "flowState", "nextaction" as AUTH_ALLOWED and the "Status" as "ACTIVE" with other required details. The status indicates that the device is active to receive Mobile OTP (MOTP) and indicates that the registration process has successfully completed. However, the "nextaction" will depend upon the policy and may not always result in "Auth_Allowed". Next time onward, if you must use MOTP authentication, the MOTP will be sent directly to your registered mobile device for approval.

Step to Register Your Device with VIP Access App Using REST APIs

1. Use the following VIPCredentialRegistrar API to register your device with the VIP Access app:

   ```
   https://<sspHost>/<tenantName>/factor/v1/VIPCredentialRegistrar
   ```

   NOTE

   - The notation {{text}} implies the text between and including {{}} characters must be replaced with appropriate values. For example-In this endpoint-**https://{{host}}/{{tenantName}}/factor/v1/**

VIPCredentialRegistrar, if you replace the {{text}} with values, it would look like this- https://example.com/default/factor/v1/VIPCredentialRegistrar

- To know about the scopes that are required to secure these API's, refer to the Rest APIs and Associated Privileges section.
- The solution provides a Sample Mobile application that you can use to perform authentication operations. Click here to know how to setup the application for demo usage.

In the request payload, enter the following details:

```
{

    "credType": "totp",

    "credValue": "Android Phone",

    "vipCredentialID": "VSST74448879",

    "otp": "016999"

        }
```

where,

- credType- Specify the type of mobile OTP algorithm supported by the app, which is Time-based-One-Time-Password (TOTP) and TOTP_PUSH.
- credValue- Specify a value/name of the device to help you easily recognize the device where the VIP Access Push software is installed.
- vipCredentialID- Specify the credential ID that has been generated by the VIP Access app after its installation on your device.

  NOTE
  Note that the vipCredentialID must be entered without any spaces. For example-the credID should be entered like this "VSST74448879" and not like "VSST 7444 8879".
- otp- Specify the TOTP generated by the VIP Access app.

Response Payload

Upon successfully verifying the values in the request payload, the response generates the "flowState", "nextaction" as "AUTH_ALLOWED", "status" as "ACTIVE" with other useful details. The status of the credential indicates that the device is active to receive MOTP's and indicates that the registration process has successfully completed. In this flow, both registration and verification of the device is completed.

```
{

    "flowState": "eyJh.....ZX0=",

    "nextaction": "AUTH_ALLOWED",

    "data": {

        "credId": "afa0e0bc-9350-4a9f-9fe2-b42f82bdbacf",

        "tenantId": "4764021a-9048-11ea-bb37-0242ac130002",

        "userId": "bab073a2-8201-4037-9b60-24adff61d3dc",

        "credType": "TOTP",
```

```
            "credValue": "Android Phone",

            "status": "ACTIVE",

            "createdDatetime": "2020-09-30 05:14 AM UTC",

            "modifiedDatetime": "2020-09-30 05:14 AM UTC",

            "default": false,

            "vipCredentialID": "VSST74448879",

            "userDisplayName": "nbruce",

            "tenantName": "default"

        },

        "deviceID": "45eaee7e-0cda-4b24-bd0a-3e8bd13ba123",

        "id_token": "eyJ4.....kqQ"

    }
```


Verification Flow in VIP Access App

Once you have completed the registration flow, you can thereafter use only the "/MobileOTPVerifier" call to verify the MOTP flows.

Follow this step to authenticate VIP Access MOTP flow in your device:

1. Initiate a "/MobileOTPVerifier" call to receive MOTP from Symantec Identity Security Platform (referred to as solution) onto your registered device. Enter the "credID", "credValue" and "otp" details in the API request.
   where,
   - credID- Specify the credId that has been generated by the response from the registration call.
   - credValue- Specify the device name.
   - otp- Specify the TOTP generated by the VIP Access app.

   In response to the authentication request, the solution returns the "flowState", "nextaction" as "AUTH_ALOWED" and status of the device as "ACTIVE". AUTH_ALOWED indicates that you have been authenticated with MOTP and can now access the restricted resource. However, the "nextaction" will depend upon the policy and may not always result in "Auth_Allowed".

Step to Verify MOTP flow with VIP Access App Using REST APIs

1. Initiate the following MobileOTPVerifier API endpoint to authenticate the MOTP flow:

   ```
   https://<sspHost>/<tenantName>/factor/v1/MobileOTPVerifier
   ```

   NOTE

   - The notation {{text}} implies the text between and including {{}} characters must be replaced with appropriate values. For example-In this endpoint-**https://{{host}}/{{tenantName}}/factor/v1/MobileOTPVerifier,** if you replace the {{text}} with values, it would look like this- https://example.com/default/factor/v1/MobileOTPVerifier

   MobileOTPVerifier Request Payload

   In the request payload, enter the following details:

   ```
   {
   ```

```
  "credId": "afa0e0bc-9350-4a9f-9fe2-b42f82bdbacf",
  "credValue": "Android Phone"
  "otp": "712479"

}
```

where,

- credId- Specify the credId that has been generated by the response from the registration call.
- "credValue": Specify the device name.
- otp- Specify the TOTP generated by the VIP Access app.

MobileOTPVerifier Response Payload

In response to the authentication request, the solution returns the "flowState", "nextaction" as "AUTH_ALOWED" and status of the device as "ACTIVE". AUTH_ALOWED indicates that the you have been authenticated with MOTP and can now access the restricted resource. However, the "nextaction" will depend upon the policy and may not always result in "Auth_Allowed".

```
{

    "flowState": "eyJh.....zZX0=",

    "nextaction": "AUTH_ALLOWED",

    "data": {

        "credId": "afa0e0bc-9350-4a9f-9fe2-b42f82bdbacf",

        "tenantId": "4764021a-9048-11ea-bb37-0242ac130002",

        "userId": "bab073a2-8201-4037-9b60-24adff61d3dc",

        "credType": "TOTP",

        "credValue": "Android Phone",

        "status": "ACTIVE",

        "createdDatetime": "2020-09-30 05:14 AM UTC",

        "modifiedDatetime": "2020-09-30 05:14 AM UTC",

        "default": false,

        "vipCredentialID": "VSST74448879",

        "userDisplayName": "nbruce",

        "tenantName": "default"

    },

    "deviceID": "45eaee7e-0cda-4b24-bd0a-3e8bd13ba123",

    "id_token": "eyJ4....kqQ"
```

```
}
```

Server OTP Authenticator

Learn about Server OTP, covering the registration, verification, and generation of OTP flows for SMS, Email, and IVR types.

Server OTPs are generated on the server-side and sent to your registered mobile or Email Id with which you can authenticate transactions. To start using the Server OTP capability, you must perform the steps that are mentioned in the Registration Flow and Verification flow subsequently.

Auto-enrollment of SMS, IVR, or Email OTP are possible only when the user credentials are correctly mapped in the ID Store, aligning the logical attribute with the physical attribute.

Server OTP Type	*logical Attributes - Physical Attributes*
SMS	phone_number=telephoneNumber
IVR	ivr_phone_number=telephoneNumber
Email	email=mail

NOTE

- In this release, IVR OTP as a factor can be leveraged only through the Symantec Identity Security Platform API and is not available through the Admin Console.
- The **OTP Authentication Method Reference (AMR)** now offers enhanced detail: with the **`useGranularAMRForOTP`** tenant setting enabled, the system can differentiate if an OTP was delivered via **EMAIL** or **SMS**. For more information on it, see Enhanced OTP AMR.

How Server OTP Authentication Works in the solution?

Server OTP authentication works by requiring first-time users to complete a registration and verification process. Once registered, users can generate OTPs for subsequent logins, which they must verify to access protected resources. These steps have been explained in the following sections:

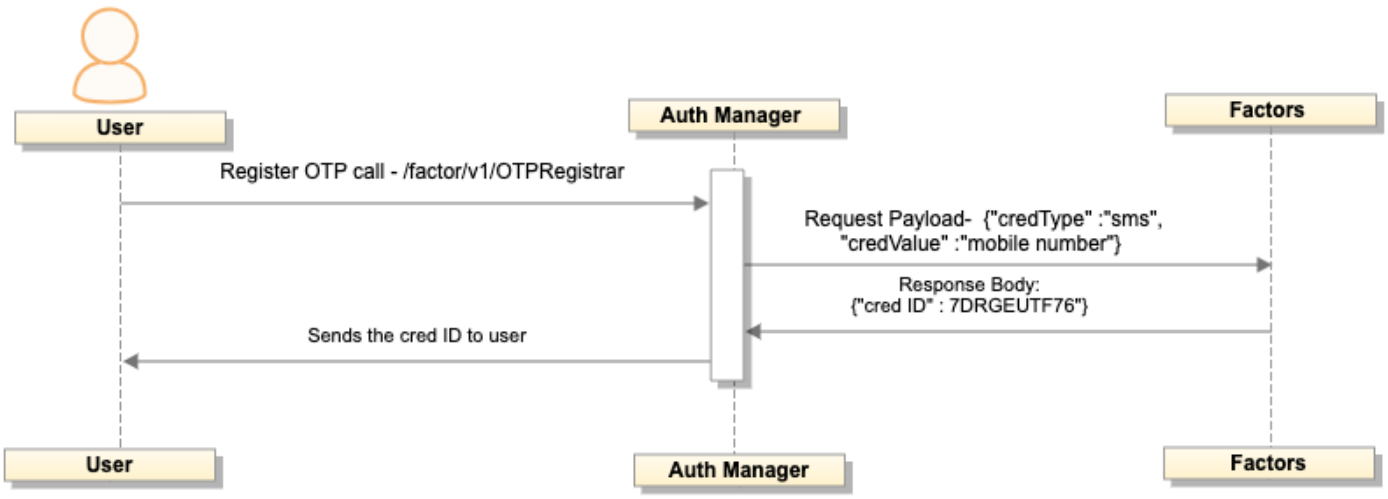
- Registration Flow
- Verification flow
- OTP Generation

Registration Flow

The registration of server OTP credentials is a one-time activity. If you are using server OTP authentication for the first time, you must complete the registration process by providing your credentials. Once you are through the registration process, your server OTP credentials are stored in the database. This allows you to generate and validate OTPs using the verification flow. The next time you login, you are immediately prompted to generate and verify transactions using server OTP.

Supported Server OTP types are SMS, EMAIL, and IVR.

The following image shows an example of credType as SMS and the steps that you must perform to register Server OTP in your device.



Follow this step to register server OTP:

> **NOTE**
> The following example describes the flow for registering "SMS" OTP onto your mobile. You can follow the same steps for "EMAIL"/ "IVR" OTP by specifying the credType as "EMAIL"/ "IVR" and entering the Email Id/ mobile number in the "credValue".

1. Acquire the User Access Token by initiating the User Authentication in the IDSP using the following API:
   ```
   POST https://${SSP_FQDN}/{{apiPathTenant}}/auth/v1/authenticate
   ```
2. Upon successful authentication, initiate a "Register OTP" call using the following "OTPRegistrar" API to register your mobile number with server OTP:
   ```
   https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPRegistrar
   ```

   NOTE

   - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
   - The default value for `<tenantName>` is "default".
   - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
     For example:
     `https://{{host}}/{{tenantName}}/factor/v1/OTPRegistrar`
     should look similar to:
     `https://example.com/default/factor/v1/OTPRegistrar`
   - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Request Payload:

```
{
  "credType": "SMS",
  "credValue": "xxxxxx3697"
  "language": "en_US"
}
```

The following information is required in the Request payload based on your selection of Credential type:

Request Attributes	*Description*	*Supported Values*
credType	Specify the type of server OTP.	***"SMS"***, ***"EMAIL"*** and "***IVR***".
credValue	• Specify the mobile number for the credTypes, ***SMS*** and ***IVR*** where the server OTP must be delivered. • specify Email ID for the credType, ***EMAIL*** where the server OTP must be delivered.	Phone Number and Email ID
language	The language code specifying the language in which the OTP message should be delivered.	

Response payload:


```
{
        "flowState": "amZMMXdFenh1MDIwNlFVaXhhVGV5Y0...",
        "userName": "CraigLINK",
        "nextaction": "SMS_OTP_AUTH",
        "data": {
            "credId": "3510b60a-f498-496f-a7b0-1686a1d32fc4",
            "tenantId": "2f0b7ce5-f78f-4a32-b248-9fd7bee0eb25",
            "userId": "16999592-0e46-404d-a38c-f028f81e169b",
            "credType": "sms",
            "credValue": "003178 is your One-Time Password to login. The OTP is valid up to 5 minutes.",
            "status": "INITIAL",
            "createdDatetime": "2024-06-28 09:46 AM UTC",
            "modifiedDatetime": "2024-06-28 09:46 AM UTC",
            "default": false
        },
        "additional": {
            "idpName": "TestConfig1",
            "idpGuid": "fa7c2fe6-2a30-436e-8d0d-d4ddc6326981",
            "idpType": "ldap",
            "passwordResetURL": "",
            "lastObligation": "IVROTP",
            "nextObligation": "SMSOTP"
        }
```


The following information is returned based on your selection of Credential type:

Response Attributes	*Description*	*Supported Values*
flowState	Refers to the current status and context of an ongoing authentication process. It acts as a container that holds all necessary information to manage and track the progression through various stages of the authentication procedure.	
userName	The username of the individual undergoing authentication. In this case, it is "CraigLINK".	
nextaction	Displays the value based on your specification of "credType". The "nextaction" depends upon the policy that is defined in the solution.	***"SMS_OTP_AUTH"*** ***"EMAIL_OTP_AUTH"*** ***"IVR_OTP_AUTH"***

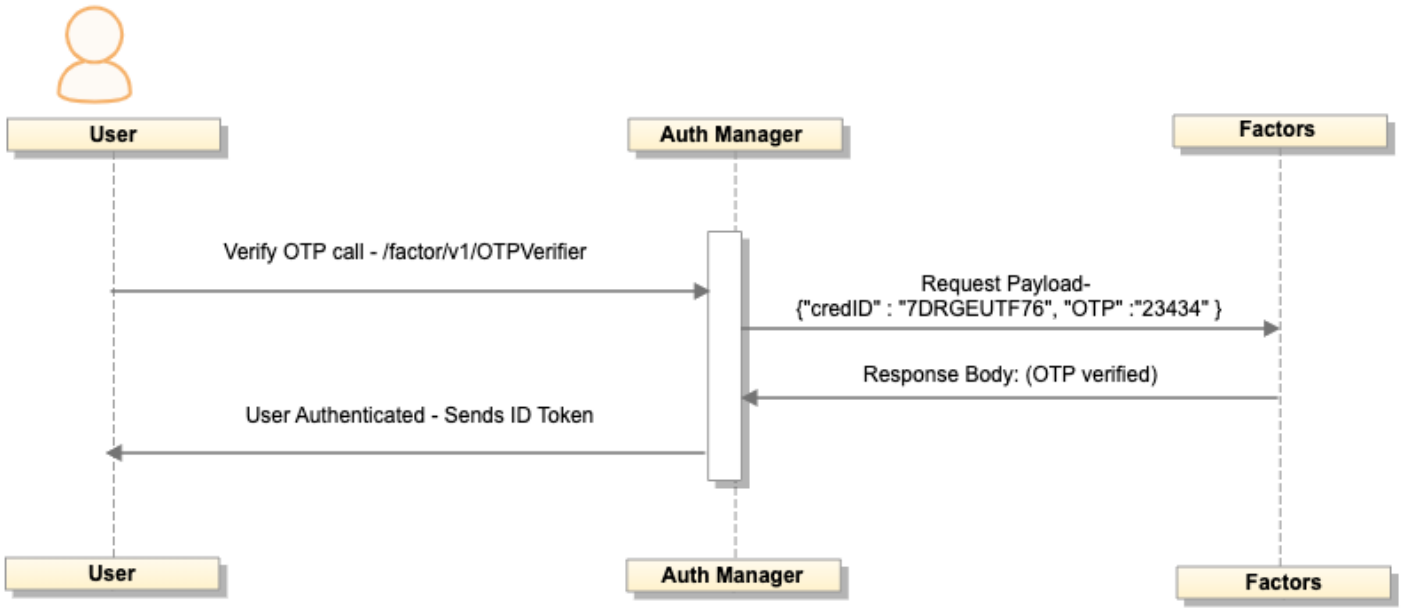
Response Attributes	*Description*	*Supported Values*
data	Includes the following information: • credId - Uniquely identifies the mobile number or email id where you want to register the server OTP. This value is used in the subsequent flow (verification flow) to verify the OTP. • tenantId - Uniquely identifies the specific tenant to which the user and the authentication request belong. • userId - A unique identifier for the user associated with the credential. • credType - Returns the type of server OTP used in the authentication flow. • credValue - Returns the OTP to your phone number. This value is used in the subsequent flow (verification flow) to verify the OTP. • status - During the registration process, the status is displayed as ***"INITIAL"*** indicating that the server OTP registration process is in progress. • createdDatetime - The date and time when the credential was created. • modifiedDatetime - The date and time when the credential was last modified. • default - A boolean indicating if this is the default credential for the user.	
additional	Displays additional information related to the authentication context. For example: idpName, idpType, lastObligation, etc.	

The server OTP registration step concludes here. The next step is to verify the OTP request. Refer to the "Verify OTP" section further down this page for details.

Verification Flow

NOTE
If you have already registered your server OTP credentials, you can skip the registration flow and start using this (verification) flow where you can straightaway generate the OTP and verify it to complete the authentication flow.

The figure shows the steps that you must perform to verify Server OTP on your mobile or email.



Follow these steps to verify the server OTP:

NOTE
The following example describes the flow for verifying "SMS" OTP in your mobile. You can follow the same steps for "EMAIL" and "IVR" OTP by specifying the "credId" and "otp" in the request payload.

1. Use the following "OTPVerifier" API to verify your mobile with server OTP:
   `https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPVerifier`

   NOTE

   - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
   - The default value for `<tenantName>` is "default".
   - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
     For example:
     `https://{{host}}/{{tenantName}}/factor/v1/OTPVerifier`
     should look similar to:
     `https://example.com/default/factor/v1/OTPVerifier`
   - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

2. **Request payload**:

```
{
 "credId": "3510b60a-f498-496f-a7b0-1686a1d32fc4",

 "otp": "256280"
}
```

The following information is required in the Request payload based on your selection of Credential type:

Request Attributes	*Description*
credId	Specify the "credId" obtained from the registration flow.
otp	Specify the "otp" received in your registered phone during the registration flow.

Response payload:


```
{
    "flowState": "amZMMXdFenh1MDIwNlFVaXh...",
    "userName": "CraigLINK",
    "nextaction": "AUTH_ALLOWED",
    "data": {
        "status": "OK",
        "message": "One-Time Password (OTP) is verified successfully"
    },
    "id_token": "eyJ4NXQjUzI1NiI6Imw5...",
    "additional": {
        "idpName": "TestConfig1",
        "idpGuid": "fa7c2fe6-2a30-436e-8d0d-d4ddc6326981",
        "idpType": "ldap",
        "passwordResetURL": "",
        "authLevelsData": {
            "0a505804-ed46-4e81-bfb6-f6e30b17bb41+CraigLINK+1": "SMSOTP"
        },
        "currentFactorLevel": 1,
        "clientRespContext": [
            {
                "key": "deviceId",
                "value": "post risk success"
            }
        ],
        "amrForCompletedFactors": [
            "OTP"
        ],
        "lastObligation": "SMSOTP"
    }
}
```


3. The following table describes the response attributes:

Response Attributes	*Description*	*Supported Values*
flowState	Refers to the current status and context of an ongoing authentication process. It acts as a container that holds all necessary information to manage and track the progression through various stages of the authentication procedure.	
userName	The username of the individual undergoing authentication. In this case, it is "CraigLINK".	

Response Attributes	*Description*	*Supported Values*
nextaction	Displays the value based on your specification of "credType". The "nextaction" depends upon the policy defined in the solution. In this sample, the "***nextaction***" is assumed to be "AUTH_ALLOWED". However, based on the policy that has been defined in IDSP, it may challenge you with other factors before allowing you to access the protected resource.	***"SMS_OTP_AUTH", "EMAIL_OTP_AUTH" ,"IVR_OTP_AUTH"***
data	Includes the following information: • status - The "***OK***" status indicates that the process of server OTP verification is complete and the mobile is ready to receive server OTPs for authentication. After completing the registration and verification process as a first-time user, for all subsequent verification response, the status would turn to "***ACTIVE***" indicating that the mobile is active to receive server OTPs for authentication. • message - Displays message that says, One-Time Password (OTP) is verified successfully	
id_token	After the user has successfully verified the One-Time Password (OTP), the ID token is provided. The token helps the client application understand that the user has been authenticated and can proceed with authorized actions.	
additional	Displays additional information related to the authentication context. For example: idpName, idpType, authLevelsData, etc.	

NOTE
When the "***preserveOriginInAutoEnrollment***" configuration setting is set to ***false*** and "***isAutoEnrollmentAllowed***" is set to ***true***, SMS or EMAIL credentials that are registered by the user using inline enrollment are **removed** the next time that the user logs in for authentication. The Credentials are removed only when the existing credentials do not match the updated credentials in the authoritative source. The relevant configuration settings are covered in Global/Tenant Settings and Credential Synchronization in Auto Enrollment.

Generate OTP API

After completing the registration and verification process as a first-time user, you can generate OTPs directly to your email or mobile for all subsequent requests using the OTPGenerator API. To complete the authentication flow and access-protected resources, use the ".../factor/v1/OTPVerifier" API to verify the OTP, as detailed in the preceding section.

Use the following "OTPGenerator" API to generate OTP to receive in your registered mobile number:

```
https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPGenerator
```

The following is the request payload for the "Generate OTP" call:

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/factor/v1/OTPGenerator`
  should look similar to:
  `https://example.com/default/factor/v1/OTPGenerator`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

```
Request : {
    "action": "Login",
    "credId": "3510b60a-f498-496f-a7b0-1686a1d32fc4",
    "credType": "SMS",
    "language": "en_US"
}
```

The following information is required in the Request payload based on your selection of Credential type:

Request Attributes	***Description***	***Supported Values***
action	The specific action or operation for which the OTP is being requested.	login, authenticate
credId	Specify the "credId" obtained from the registration flow.	
credType	Returns the type of OTP server used in the authentication flow.	***"SMS"***, ***"EMAIL"*** and "***IVR***".
language	The language code specifying the language in which the OTP message should be delivered.	

The following is the response payload:

```
"flowState": "amZMMXdFenh1MDIwNlFVaX...",
    "userName": "CraigLINK",
    "nextaction": "SMS_OTP_AUTH",
    "currentFactors": [
        "SMSOTP"
    ],
    "data": {
        "credId": "3510b60a-f498-496f-a7b0-1686a1d32fc4",
        "message": "Your one-time OTP has been sent to your registered mobile number"
    },
    "additional": {
        "idpName": "TestConfig1",
        "idpGuid": "fa7c2fe6-2a30-436e-8d0d-d4ddc6326981",
        "idpType": "ldap",
        "passwordResetURL": "",
        "currentFactorLevel": 1,
        "lastObligation": "SMSOTP",
        "nextObligation": "SMSOTP"
    }
```

```
}
```

The following information is returned based on your selection of Credential type:

Response Attributes	*Description*	*Supported Values*
flowState	Refers to the current status and context of an ongoing authentication process. It acts as a container that holds all necessary information to manage and track the progression through various stages of the authentication procedure.	
userName	The username of the individual undergoing authentication. In this case, it is "CraigLINK".	
nextaction	Displays the value based on your specification of "credType". The "nextaction" depends upon the policy defined in the solution.	***"SMS_OTP_AUTH"*** ***"EMAIL_OTP_AUTH"*** ***"IVR_OTP_AUTH"***
currentFactors	A list of current authentication factors being used.	
data	Includes the following information: • credId - Uniquely identifies the mobile number or email id where you want to register the server OTP. • message - This is the OTP value that is sent to your registered phone number.	
additional	Displays additional information related to the authentication context. For example: idpName, idpType, currentFactorLevel, lastObligation, etc.	

After the OTP gets generated to your mobile or email, use the "/OTPVerifier" API to verify and get access to the protected page.

Server OTP endpoint URLS

The following table lists the Server OTP Authenticator APIs that you can use to register, generate, and validate transactions through OTPs.

Operation Description	Operation Type	Endpoint URLs
Register OTP - SMS	POST	https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPRegistrar
Generate OTP - SMS	POST	https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPGenerator
Verify OTP - SMS	POST	https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPVerifier
Register OTP - EMAIL	POST	https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPRegistrar
Generate OTP - EMAIL	POST	https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPGenerator

Verify OTP - EMAIL	POST	https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPVerifier
Register OTP - IVR	POST	https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPRegistrar
Generate OTP - IVR	POST	https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPGenerator
Verify OTP - IVR	POST	https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPVerifier

To view the request and response payload for these URLs, refer to Factor Swagger documentation.

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/factor/v1/OTPRegistrar`
  should look similar to:
  `https://example.com/default/factor/v1/OTPRegistrar`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Reference Links to:

- Integrating with Custom SPI
- Factor Error Codes

Identity Provider (IDP) as an Authentication Factor

Identity Provider acting as an authentication factor.

An Identity Provider (IDP) is an entity that provides claims about identity. Such claims are expressed through an entity called Assertion. For example, such Assertions may state claims covering work-related information, personal information, business context-specific information, etc. Such claims are assumed to be trusted at a certain level of assurance as defined by those responsible for establishing the relationship with the IDP. Once an IDP is defined, it is presumed to be authoritative for certain claims about the identity, and such claims become available to business logic involved in the process of Authentication, Authorization, Administration, and Governance, meeting the business needs of each individual deployment.

IDP as an Authentication Factor

As a factor credential, it is assumed that the identity established by the IDP (or a derivative of that identity as communicated by the IDP) is used to assert the Identity in Symantec Identity Security Platformin order to drive additional authentication flows. The final identity is established by IDSP on the basis of its own identity currently required to exist in IDSP's LDAP Identity Store (since AuthN/AuthZ policies are mapped to LDAP subjects). IDSP supports the use of SAML and OIDC standard protocols to establish the IDP relationships. It accepts third-party IDP identity as a factor in the authentication flow.

NOTE
In the IDSP, IDP as an authentication factor is supported only by the Signin UI.

Configuring Identity Provider in IDSP Policy

The following sample shows the configuration of IDP as one of the authentication factors in theIDSP policy similar to other factors, such as PASSWORD, SMSOTP, etc.

Prerequisite: The IdentityProvider with a valid identityProviderId should already be configured in the system.
The identityProviderId can be retrieved from the response of the POST admin/v1/IdentityProviders API or POST /admin/v1/SAMLIdpMetadata API. To create an Identity Provider in IDSP, see Managing Identity Provider .

Sample Policy which shows IDP configured as a factor in the policy.

```
{
            "policyName": "Sample policy with IDP as a factor",
            "description": "Default policy for Tenant Admin Console",
            "status": "active",
            "matchAnyApp": false,
            "apps": [
                {
                    "name": "testApp",
                    "id": "ad96404e-a808-4586-880c-e1312bfd3210"
                }
            ],
            "rules": [
                {
                    "conditions": {
                        "principal": {
                            "user": {
                                "operator": "in",
                                "value": [
                                    "testUser"
                                ]
                            }
                        },
                        "channel": {
                            "operator": "in",
                            "value": [
                                "web",
                                "mobile"
                            ]
                        }
                    },
                    "result": {
                        "effect": "allow",
                        "reAuthenticate": "false",
                        "obligations": [
                            "PASSWORD:1",
                            "IDP/<identityProviderID>:2"
                        ],
                        "msg": "This policy is for testing IDP",
                        "mfaFrequency": "OnceForTrustedDevice"
                    }
                }
            ]
        }
```

Here, IDP has been configured as the 2nd factor.

The following format is used to configure the IDP factor in the policy:

```
"IDP/{{identityProviderId}}:2"
```

Where,

- identityProviderId is the <idp-guid> which is the SAML IDP id that is generated by IDSP. The identityProviderId can be retrieved from the response of the POST admin/v1/IdentityProviders API or POST /admin/v1/SAMLIdpMetadata API.
- "2" denotes the level at which the factor should be authenticated.

Reference Links to:

- Managing Identity Provider
- Integrating Identity Provider with Custom SignIn UI
- Factor Swagger documentation
- Factor Error Code

Custom Authentication Factor

Learn how to implement and integrate custom authentication factors using the Custom Authenticator API, which enables seamless interaction between customer-developed SPIs and the out-of-the-box SignIn UI for enhanced authentication flexibility.

Symantec Identity Security Platform supports custom authentication factors through a Custom SPI, which integrates with the default SignIn UI or a custom-developed SignIn client. The Custom Authenticator API facilitates communication between the SPI and the UI, allowing the custom factor to be used as a standalone requirement or as part of a multi-factor authentication policy.

The following table displays the Custom Authenticator API operation type and endpoint URL:

Operation Description	Operation Type	Endpoint URL
This API allows you to authenticate yourself using a custom authentication factor in the authentication flow	POST	https://${SSP_MGMT_FQDN}/system/factor/v1/CustomAuthenticator

NOTE

- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://${SSP_MGMT_FQDN}/<systemtenantName>/factor/v1/CustomAuthenticator`
  should look similar to:
  `https://mgmt-example.com/factor/v1/CustomAuthenticator`
- To know the value of ${SSP_MGMT_FQDN} and ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<systemTenantName>` is "system".
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

For details on how the Custom Authenticator API interacts with the custom SPI and the out-of-the-box SignIn UI to support a custom authentication factor, see Integrating with Custom Authentication Factor.

Attribute-Based Authentication Factors

Explore how to implement attribute-based authentication using the ".../AuthenticationFactor" API.

Attribute-based authentication factors enable administrators to configure authentication challenges that verify users based on one or more of their personal attributes, such as Date of Birth, Social Security Number, or other identity data. By leveraging these attributes, administrators can define authentication factors tailored to individual users, allowing for more dynamic and personalized authentication workflows that enforce the appropriate level of security for each user. User attributes can be retrieved from connected Identity Stores, such as LDAP, SCIM, or the Internal ID Store.

This topic includes the following sections:

- API for managing attribute-based authentication factors
- Examples of configuring attribute-based authentication factors
- API usage examples
- Adding user attribute-based authentication factors to authentication policies
- Displaying attribute-based authentication factors as the next action in the authentication flow
- API for authenticating using AuthenticationFactor in the authentication flow

API for managing attribute-based authentication factors

The following table displays the attribute-based authentication factor API, and the operation type, scope, and endpoint URL that this API supports:

NOTE
The scope of ".../admin/v1/AuthenticationFactor" API is `urn:iam:t.configs` .

Operation Description	Operation Type	Endpoint URL
This API allows you to create an authentication factor based on one or more user attributes for a specific tenant.	POST	https://${SSP_FQDN}/<tena
This API allows you to update the user attributes in the authentication factor based on the specified authentication factor Id.	PUT	https://${SSP_FQDN}/<tena
This API returns a list of all the authentication factors that are created for a specified tenant in Symantec Identity Security Platform.	GET	https://${SSP_FQDN}/<tena
This API returns the authentication factor that was created for a tenant, based on the specified authentication factor Id.	GET	https://${SSP_FQDN}/<tena
This API allows you to delete an authentication factor for a specific tenant based on the specified "AuthenticationFactor" Id.	DELETE	https://${SSP_FQDN}/<tena

NOTE

- To know the value of the ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {text}/<text> including {}/<> characters with appropriate values in all the endpoints.
  For example:`https://${SSP_FQDN}/<tenantName>/admin/v1/AuthenticationFactor` should look similar to:`https://example.com/default/admin/v1/AuthenticationFactor`

Examples of configuring attribute-based authentication factors

Authenticator Name (Customer Defined)	**Attribute Details**			
DOB and DOJ	**Source**	**Attribute Name**	**Data Type**	**Display Name in**

	idstore	dateOfBirth	date	DOB
	idstore	dateOfJoining	date	DOJ
DOB and SSN	**Source**	**Attribute Name**	**Data Type**	**Display Name in**
	idstore	dateOfBirth	date	DOB
	idstore	ssn	string	Last 4 digits of So
MAIDEN NAME	**Source**	**Attribute Name**	**Data Type**	**Display Name in**
	idstore	mothersMaidenName	string	Mothers Maiden N
SSN	**Source**	**Attribute Name**	**Data Type**	**Display Name in**
	idstore	ssn	string	Last 4 digits of So
PHONE	**Source**	**Attribute Name**	**Data Type**	**Display Name in**
	idstore	phone	number	Phone Number

API usage examples

POST operation: Sample request to create an authentication factor based on one or more user attributes

```
POST https://${SSP_FQDN}/<tenantName>/admin/v1/AuthenticationFactor
```

Request

```
{
  "name": "FACTOR_DOB_SSN",
  "description": "Authenticate using DOB and SSN",
  "status": "active",
  "displayText": [
    {
      "locale": "en_us",
      "label": "Verify Identity with Date of Birth and SSN"
    }
  ],
  "amr": "kba",
  "attributes": [
    {
      "name": "dateOfBirth",
      "uiFieldName": "dob",
      "source": "idstore",
      "displayText": [
        {
          "locale": "en_us",
          "label": "Date of Birth",
          "helpText": "MM/DD/YYYY"
        }
      ],
      "operator": "eq",
      "dataType": "date"
    },
    {
      "name": "socialSecurityNumber",
      "uiFieldName": "ssn",
      "attributeSource": "idstore",
      "displayText": [
        {
```

```
            "locale": "en_us",
            "label": "Last 4 digits of SSN",
            "helpText": "Last 4 digits"
          }
        ],
        "operator": "regex(([a-zA-Z0-9]{4})$)",
        "dataType": "string"
      }
    ]
  }
```

Where,

- **name** - Unique name of the authentication factor.
- **description** - Description of the Authentication Factor
- **status** - Status of the factor (active/inactive)
- **displayText** - List of labels for the factor based on the locale-
  - **locale** - Specify the locale in which the attributes must be displayed.
  - **label** - Specify the label of the factor in the specified locale.
- **attributeSource** - Specify the ID of the ID Store or the IDTOKEN (JWT) if using ZFP flow, to disambiguate the user.
- **amr** - Authentication Method Reference (amr) associated with the factor.
- **attributes** - List of attributes based on which the factor is defined. To see a few samples of how to configure attribute based authentication factors, see Table showing samples of attribute-based authentication factors configuration.
  - **name** - Name of the logical attribute defined in the identity store mapping. If using ZFP flow, this value is the name of the claim (standard or custom claim).
  - **uiFieldName** - Name of the attribute to be displayed by the application.
  - **source** - This is an optional parameter. If this value is not specified, then it uses the "attributeSource" value as the identity source for this attribute.
    Specify the ID of the identity store (Internal or LDAP Identity Store Config ID or SCIM Identity Store Config ID). If using ZFP flow, the value is "IDTOKEN".
  - **dataType** - Specify the data type of the attribute that helps to provide additional context about it. Supported data types are: number, string, and date.
  - **isSensitive** - Default value is false.
    If the value is set to true, then the UI does not display the users input value (similar to the password field), and the value is encrypted before it is sent to the backend for validation.
  - **displayText** - List of labels for the factor based on the locale-
    - **locale** - Specify the locale in which the attributes must be displayed.
    - **label** - Specify the label of the factor in the specified locale.
    - **helpText** - Specify the help text to be returned back to the UI.
  - **operator** - Specify the operator that should be used when comparing the users input with the attribute data in the identity store.
    The following are the supported operator values:

- **eq** - Checks whether the user's input value is equal to the attribute value stored in the identity store. Both values are converted to lower case before comparison. This operator can be used with all data types (string, number, and date).
- **eq (<date format>)** - Specifies the date format used to store the date in the idStore/IDTOKEN, for example; (MM/DD/YYYY). If no format is specified, then the default IDSP supported format (yyyy-MM-dd HH:mm aaa z) is used to read the date from idStore/IDTOKEN.
- **ne** - Checks whether the user's input value is not equal to the attribute value stored in the identity store. This operator can be used with all data types (string, number, and date).
- **gt** - Checks whether the user's input value is greater than the attribute value stored in the identity store. This operator can only be used with the number data type.
- **lt** - Checks whether the user's input value is less than the attribute value stored in the identity store. This operator can only be used with the number data type.
- **substring(<startindex>,<endindex>)** - Checks whether the user's input value matches the result of attribute_value.substring(<startindex>,<endindex>). Both the start index and end index are inclusive, and indexing is zero-based (that is, the first character is at index 0). This operator can only be used with the string data type.
  Examples:
  - substring(0,3) — Matches the first 4 characters of the attribute value (indices 0, 1, 2, 3).
  - substring(1,4) — Matches a substring from the 2nd character to the 5th character (indices 1, 2, 3, 4).
- **regex(<regex string>)** - Checks whether the user's input value matches the result of applying the specified regular expression to the attribute value, i.e., attribute_value.regex(<regex expression>). This operator can be used only for string data type.
  For example,
  - regex(**^[a-zA-Z0-9]{4}**) - match first 4 characters.
  - regex(**([a-zA-Z0-9]{4})$**) - match last 4 characters.

Response

The response returns the **"authFactorId"** and other details.


```
{
  "authFactorId": "7d182f97-8351-4a5e-99a8-710a0a13d00",
  "name": "FACTOR_DOB_SSN",
  "description": "Authenticate using DOB and SSN",
  "status": "active",
  "displayText": [
    {
      "locale": "en_us",
      "label": "Verify Identity with Date of Birth and SSN"
    }
  ],
  "amr": "kba",
  "attributes": [
    {
      "name": "dateOfBirth",
      "uiFieldName": "dob",
      "source": "idstore",
      "displayText": [
        {
          "locale": "en_us",
          "label": "Date of Birth",
          "helpText": "MM/DD/YYYY"         }
      ],
```

```
      "operator": "eq",
      "dataType": "date"
    },
    {
      "name": "socialSecurityNumber",
      "uiFieldName": "ssn",
      "attributeSource": "idstore",
      "displayText": [
        {
          "locale": "en_us",
          "label": "Last 4 digits of SSN",
          "helpText": "Last 4 digits"
        }
      ],
      "operator": "regex(([a-zA-Z0-9]{})$)",
      "dataType": "string"
    }
  ],
  "createdBy": {
    "principalType": "CLIENT",
    "principalId": "84484476-e246-4b03-aa83-a6d0bd16e9c2",
    "principalName": "DefaultTenantClient"
  },
  "createdDate": "2024-01-04 06:38 AM UTC"
}
```

PUT operation: Sample request to update the user attributes in the authentication factor based on the specified authentication factor Id

```
PUT https://${SSP_FQDN}/<tenantName>/admin/v1/AuthenticationFactor{{id}}
```

Where,

- id - This is the "**authFactorId**" that is returned in the response of the POST "...admin/v1/AuthenticationFactor" API after the successful creation of the authentication factor.

Request

```
{
  "name": "FACTOR_DOB_SSN",
  "description": "Authenticate using DOB and SSN",
  "status": "active",
  "displayText": [
    {
      "locale": "en_us",
      "label": "Verify Identity with Date of Birth and SSN"
    }
  ],
  "amr": "kba",
  "attributes": [
    {
      "name": "dateOfBirth",
      "uiFieldName": "dob",
      "source": "idstore",
      "displayText": [
        {
          "locale": "en_us",
```

```
          "label": "Date of Birth",
          "helpText": "MM/DD/YYYY"
        }
      ],
      "operator": "eq",
      "dataType": "date"
    },
    {
      "name": "socialSecurityNumber",
      "uiFieldName": "ssn",
      "attributeSource": "idstore",
      "displayText": [
        {
          "locale": "en_us",
          "label": "Last 4 digits of SSN",
          "helpText": "Last 4 digits"
        }
      ],
      "operator": "regex(([a-zA-Z0-9]{})$)",
      "dataType": "string"
    }
  ]
}
```

Response

```
{
  "name": "FACTOR_DOB_SSN",
  "description": "Authenticate using DOB and SSN",
  "status": "active",
  "displayText": [
    {
      "locale": "en_us",
      "label": "Verify Identity with Date of Birth and SSN"
    }
  ],
  "amr": "kba",
  "attributes": [
    {
      "name": "dateOfBirth",
      "uiFieldName": "dob",
      "source": "idstore",
      "displayText": [
        {
          "locale": "en_us",
          "label": "Date of Birth",
          "helpText": "MM/DD/YYYY"
        }
      ],
      "operator": "eq",
      "dataType": "date"
    },
    {
       "name": "socialSecurityNumber",
```

```
        "uiFieldName": "ssn",
        "attributeSource": "idstore",
       "displayText": [
         {
           "locale": "en_us",
           "label": "Last 4 digits of SSN",
           "helpText": "Last 4 digits"           }
       ],
       "operator": "regex(([a-zA-Z0-9]{4})$)",
       "dataType": "string"
     }
  ],
  "createdBy": {
    "principalType": "CLIENT",
    "principalId": "84484476-e246-4b03-aa83-a6d0bd16e9c2",
    "principalName": "DefaultTenantClient"
  },
  "createdDate": "2024-01-04 06:38 AM UTC",
  "updatedBy": {
    "principalType": "USER",
    "principalId": "e786f772-43da-48cb-bb85-5879c1c5dcfa",
    "principalName": "kevinlucas"
  },
  "updatedDate": "2024-01-04 11:28 AM UTC"
}
```

GET operation: Sample request to retrieve the list of all user attributes based Authentication Factors based on the specified AuthenticationFactor Id (authFactorId).

```
GET https://${SSP_FQDN}/<tenantName>/admin/v1/AuthenticationFactor{{id}}
```

Response

```
[
    {
        "authFactorId": "c27c5bcf-a12d-4303-be59-1b9f0d2ca758",
        "name": "FACTOR_DOB_SSN",
        "description": "Authenticate using DOB and SSN",
        "status": "ACTIVE",
        "displayText": [
            {
                "locale": "en-US",
                "label": "Verify Identity with Date of Birth and SSN (en-US)"
            }
        ],
        "attributeSource": "internal",
        "amr": "kba",
        "attributes": [
            {
                "name": "birthdate",
                "uiFieldName": "DOB",
                "source": "",
                "displayText": [
                    {
                        "locale": "en-US",
                        "label": "Enter date of birth",
```

```
                        "helpText": "MM/DD/YYYY"
                    },
                    {
                        "locale": "en-GB",
                        "label": "Enter date of birth",
                        "helpText": "MM/DD/YYYY"
                    }
                ],
                "operator": "eq(yyyy-MM-dd HH:mm aaa z)",
                "dataType": "date",
                "sensitive": false
            },
            {
                "name": "nickname",
                "uiFieldName": "nickname",
                "source": "",
                "displayText": [
                    {
                        "locale": "en-US",
                        "label": "Enter your nick name",
                        "helpText": "Nick name"
                    },
                    {
                        "locale": "en-GB",
                        "label": "Enter your nick name",
                        "helpText": "Nick name"
                    }
                ],
                "operator": "eq",
                "dataType": "String",
                "sensitive": false
            }
        ],
        "createdBy": {
            "principalType": "CLIENT",
            "principalId": "636c4aae-4543-4f9c-b570-cf205257aa26",
            "principalName": "DefaultTenantClient"
        },
        "createdDate": "2024-10-17 03:55 AM UTC",
        "updatedDate": "2024-10-17 03:55 AM UTC"
    }
]
```


Adding user attribute-based authentication factors to authentication policies

After the user attributes authenticator (factor) has been created using the POST "https://${SSP_FQDN}/<tenantName>/admin/v1/AuthenticationFactor" API, the "authFactorId" generated from this POST operation should be added as one of the obligations in the required authentication policies to have this factor in the authentication flow.

The following sample shows the inclusion of an attributes authenticator as an obligation in the Authentication policy:

API URL

```
POST https://${SSP_FQDN}/<tenantName>/admin/v1/AuthPolicies
```

Request

```
{
    "description": "Policy using user attributes authenticator (FACTOR)",
    "status": "active",
    "policyName": "FACTOR POLICY",
    "apps": [
        {
            "id": "710a0a13d00-8351-4a5e-99a8-7d182f97"
        }
    ],
    "rules": [
        {
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "HR"
                        ]
                    }
                }
            },
            "result": {
                "effect": "allow",
                "obligations": [
                    "FACTOR/7d182f97-8351-4a5e-99a8-710a0a13d00:1",
                    "EMAILOTP:2",
                    "SMSOTP:2"
                 ],
                "msg": "MFA Rule with user attributes authenticator as primary factor"
        }
}
```

Where the "**authFactorId**" GUID, "**FACTOR/7d182f97-8351-4a5e-99a8-710a0a13d00:1**" is added in the "**result**" section of the Authentication Policy.

Displaying attribute-based authentication factors as the next action in the authentication flow

The following sample displays the attribute-based authentication factor as a next action in the authentication flow. The "nextaction" is added in the auth flows as - **ATTR_AUTH.**

> **NOTE**
>
> In this authentication flow response, the values that are displayed under the "**data**" field are used by the application as input to render the user attributes for authentication in the Signin UI.

```
{
  "flowState": "<flowstateid>",
  "userName": "malbail",
  "nextaction": "ATTR_AUTH",
  "currentFactors": [
    "FACTOR/749ccbbd-7d7c-4f40-bbe4-2a0b6d64bbcb"
  ],
  "data": {
    "credData": [
```

```
      {
        "key": "auth.factor.challenge",
        "value": "Validate Identity with Date of Birth and SSN"
      }
    ],
    "credDataInputFields": [
      {
        "key": "dob",
        "label": "Date of Birth",
        "helpText": "MM/DD/YYYY",
        "dataType": "date"
      },
      {
        "key": "ssn",
        "label": "Last 4 digits of SSN",
        "helpText": "Last 4 digits",
        "dataType": "number"
      }
    ]
  },
  "additional": {
    "currentFactorLevel": 1,
    "predicatesForCurrentFactors": [
      {
        "factor": "FACTOR/749ccbbd-7d7c-4f40-bbe4-2a0b6d64bbcb",
        "predicate": "DateofbirthAndSSN",
        "amr": "kba"
      }
    ]
  }
}
```

API for authenticating using AuthenticationFactor in the authentication flow

This API validates the user response for the authentication factor attributes during authentication flow.

API URL

```
POST https://${SSP_FQDN}/<tenantName>/factor/v1/Authenticator
```

Request

```
{
  "credData": [
    {
      "key": "dob",
      "value": "01/31/1980"
    },
    {
      "key": "ssn",
      "value": "1234"
    }
  ]
}
```

The "date" type attribute value should be in the mm/dd/yyyy format. The response for this API is either an authSuccess message or displays the next action with the obligations. If there is a Next Action, and there are multiple options, the Next Action is initiated based on the selected factor. See Factor Selection API.

Performing ID Store Search

The IdStore search allows you to search users and authenticate users based on username and password.

The ID Store search helps in searching and authenticating users and groups based on username and password in both SCIM and LDAP directories based on the configuration setting in the Symantec Identity Security Platform. The search performs read-only operations for user look-up on the configured LDAP V3 compliant directories and SCIM based ID Store directory service. Many other management APIs make use of this search for user attribute look-up.

To use this API, you must first configure LDAP/SCIM in the solution. See the following links to know how to configure LDAP/SCIM using REST API.

- To know about LDAP Configuration, see LDAP Configuration
- To know about SCIM Configuration, see SCIM Configuration

ID Store API

To perform ID Store user authentication, configure the following attributes in the request payload of the Factor Authenticate API. **Factor Authenticate API Request Payload**

```
{
  "userId": "NadiaKite",
  "password": "password"
}
```

List of attributes and descriptions used in the ID Store Search API

Attribute Name	Description
userId	User who must be authenticated in the LDAP/SCIM Id store.
password	Password of the user specified in the LDAP/SCIM username field.

The following table displays the list of Id Store APIs that can be used to perform user look-up:

Operation Description	Operation Type	Endpoint URLs
Search Users	GET	https://${SSP_FQDN}/{{tenantName}}/idStore/v1/Users/{{userN e}}
Retrieve a list of all the groups in an ID Store	GET	https://${SSP_FQDN}/{{tenantName}}/idStore/v1/Groups
Search the user by specifying an attribute	GET	https://${SSP_FQDN}/{{tenantName}}/idStore/v1/Users?<logica tribute-name>=value
Authenticate Users	POST	https://${SSP_FQDN}/{{tenantName}}/factors/v1/PasswordAuth icator

To view the request and response payload for these URLs, refer to ID Store Operations User Search Swagger documentation.

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/idStore/v1/Users`
  should look similar to:
  `https://example.com/default/idStore/v1/Users`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Transaction Signing using FIDO

Learn about Transaction signing using FIDO2 authentication flow.

The process of transaction signing detects any tampering of a transaction message during its route to the end device display and back to the Relying Party. Once a user approves a transaction, the user cannot deny approving it, thus giving the transaction signing another alternative name, transaction non-repudiation. The use cases of transaction signing include a funds transfer, a context specific privileged action, the confirmation of an email ID or an address, and so on.

This topic includes the following sections:

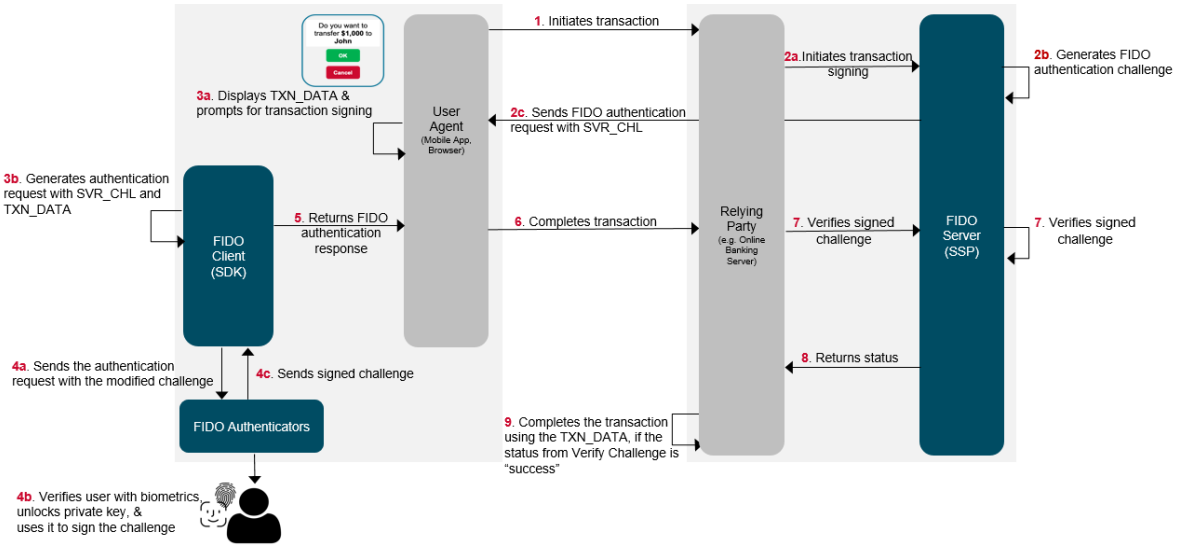
1. Transaction Signing Flow
2. List of Transaction Signing APIs
3. Example of API Request and Response
4. Usage of Transaction Signing APIs

Transaction Signing Flow

Symantec Identity Security Platform supports transaction signing using the FIDO2 authentication flow.

The following diagram describes how the enhanced FIDO2 authentication flow supports transaction signing in IDSP:

Figure 29: Transaction Signing Using FIDO2 Authentication Flow



1. User initiates a sensitive transaction in the application.
2. Relying Party initiates the transaction signing flow using the FIDOTransactionSigningChallengeGenerator API. As part of this, FIDO Server generates an authentication request with a server challenge (SVR_CHL) and sends it to the user.
3. Application displays a user verification prompt with the transaction data (TXN_DATA). When the user accepts the prompt, the application initiates transaction signing by passing the FIDO authentication request, including the server challenge (SVR_CHL) and the transaction data (TXN_DATA), to FIDO Client (SDK).
   FIDO Client (SDK) appends the transaction data (TXN_DATA) to the server challenge (SVR_CHL) in the authentication request.
4. FIDO Client (SDK) sends the authentication request with the modified challenge to FIDO Authenticators. FIDO Authenticators verify the user to unlock the private key, use it to sign the challenge hash, and send the signed challenge to FIDO Client (SDK).
5. FIDO Client (SDK) returns the FIDO authentication response to the application.
6. Application initiates transaction completion by passing the original transaction data (TXN_DATA) and the signed challenge to the Relying Party.
7. Relying Party invokes the FIDOTransactionSigningChallengeVerifier API to verify the signed challenge, and to confirm that the transaction data (TXN_DATA) is not tampered and that the FIDO authentication is successful.
   FIDO Server verifies the transaction data (TXN_DATA), server challenge (SVR_CHL), and the signature. It also extracts the original SVR_CHL and TXN_DATA from the FIDO challenge response.
   - The extracted FIDO challenge must match the original challenge that was created by FIDO Server.
   - The extracted TXN_DATA must match the TXN_DATA in the verify challenge request payload.
8. FIDO server returns the status of the transaction verification. If the verification of challenge, transaction, and signature is successful, “success” is returned. If the verification fails, “failure” is returned.
9. Relying Party completes the transaction if the transaction verification is “success”.

Once a transaction is complete and the data is signed, the Relying Party completes the transaction with confidence that the transaction is approved by the user.

List of Transaction Signing APIs

The following table lists the APIs that are used in the transaction signing process:

Operation Description	Operation Type	Endpoint URL
Generate a FIDO challenge to sign transactions	POST	https://${SSP_FQDN}/{{tenantName}}/factor/v1/FIDOTransactionSigningChallengeGenerator
Validate transaction data and signed challenge	POST	https://${SSP_FQDN}/{{tenantName}}/factor/v1/FIDOTransactionSigningChallengeVerifier

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/factor/v1/FIDOTransactionSigningChallengeGenerator`
  should look similar to:
  `https://example.com/default/factor/v1/FIDOTransactionSigningChallengeGenerator`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Example Request and Response Formats

Generate a FIDO challenge to sign transactions using the FIDOTransactionSigningChallengeGenerator API

POST Request

```
{
 "documentDomain": "https://www.example.com",
 "userName": "nbruce",
 "fidoCredType": "FIDO"
}
```

Response

```
{
    "challenge": "pqfCyzIjldPAqyD2NUzkWfxJ1Z3hJF5H1R3puJQdYcA",
    "rpId": "example.com",
    "allowCredentials": [
        {
            "type": "public-key",
            "transports": [
                "internal"
            ],
            "id":
 "AaNJCHXpuOaq0IIy838KNfi3rNXzgF__svqm3qRBSKaj-7aPEQfYeMEDrSc79ztWVJTnXLICmpUSYxoH23Fl8XY7AhPjUEm3NGhKS0RSyg"
        }
    ],
    "userVerification": "preferred"
}
```

Validate transaction data and signed challenge using the FIDOTransactionSigningChallengeVerifier API

POST Request


```
{
  "rawId": "ASxiwVt-32DWw5pOk79fzaQQH4vpJkMJcGMfbmu4HdtNsV7-
 n9yoZtjVOhHR6MbONDtm-6wiM7aY48v6CDhV0i98iblM0OljSiaJKc1qRiWHqmM",
  "response": {
    "authenticatorData": "jl48hdEAN0kIRcYwrC7TwCI27jApz656ec7rgfPWJXMFYKSc-w",
    "signature": "MEUCIQDoj9s6eE94lE3gv_1WY_c05rSzFl-fs-
AihSEqT4ZPZwIgbHRvekNZK2NY8KSQRbY2qiMDJ1ikQJ9ShS07BB5AHmY",
    "userHandle": "Q2rQCUcSK6C6rebHBQ2-JOnrGUxU6AQkBPSDhHG_PVU",
    "clientDataJSON": "eyJ0eXBlIjoid2ViYXV0aG4uZ2V0IiwiY2hhbGxlbmdlIjoiSXBGU
 Uc2UlAxVXI2RW9aRVpKYU04Z2ZxOTZoeUhGRVhPaDRxMVFjZElzayIsIm9yaWdpbiI6Imh0dHBz
 Oi8vdGhpcnVtYWwtMjctc2FtcGxlYXBwLnNzcGRldi5kZXYuYnJvYWRjb20uY29tIiwiY3Jvc3NPcmlnaW4iOmZhbHNlfQ"
  },
  "getClientExtensionResults": {},
  "id": "ASxiwVt-32DWw5pOk79fzaQQH4vpJkMJcGMfbmu4HdtNsV7-
n9yoZtjVOhHR6MbONDtm-6wiM7aY48v6CDhV0i98iblM0OljSiaJKc1qRiWHqmM",
  "type": "public-key"
"transactionData": "asul48hdYwrC7TwCI27jApz656ec7rgXMFYKSc"
}
```


Successful Response


```
{
    "status": "ok"
}
HTTP Status: 200
```


Failure Response


```
{
    "errorCode": "2003022"
    "errorMessage": "Transaction verification failed"
}
HTTP Status: 401
```


Usage of Transaction Signing APIs

These transaction signing APIs can be used on a device running Android or iOS, or in a web browser.

- For information about transaction signing on an Android device, see SDK for FIDO Android.
- For information about transaction signing on an iOS device, see SDK for FIDO iOS.
- For information about transaction signing in a web browser, see SDK for Transaction Signing (Web).

Transaction Signing using OCRA

Describes how Transaction Signing can be used using OCRA (OATH Challenge-Response Algorithm)

OCRA (OATH Challenge-Response Algorithm) is a cryptographic mechanism used for secure transaction signing, authentication, and validation. It is part of the OATH (Initiative for Open Authentication) standard and is designed to provide strong security against fraud, especially in online banking and financial transactions.

Symantec Identity Security Platform supports OATH credential as a authentication factor (MobileOTP). The existing MobileOTP credential in IDSP can be extended to support challenge based authentication, OCRA: OATH Challenge-Response Algorithm.

To support OCRA, the MobileOTP verification flow needs to be extended to accept the "ChallengeData" along with OTP value at the time of OTP verification. The OTP generated from the device is validated along with the challengeData for the provided credential. As per specification, challenge data can be numeric, alphanumeric or hexadecimal value of 64 bytes.

How OCRA Works for Transaction Signing

1. Assuming that a user already has a MOTP credential registered with IDSP.
2. The user initiates Transaction Signing using OCRA verify challenge flow.
3. The user generates an OTP using a challenge provided by the relying party.
4. The generated OTP is sent for verification along with the challenge value which was used for generating OTP.
5. OCRA transaction verification returns either success or failure message based on verification of the OTP value and challenge.

OCRA Transaction Verifier API

Admin Scope	Operation Type	Endpoint URLs
urn:iam:m.me	POST	https://${SSP_FQDN}/<tenantName>/factor/v1/OCRATransactionVerifier

NOTE

- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://${SSP_FQDN}/<tenantName>/factor/v1/OCRAVerifier`
  should look similar to:
  `https://example.com/default/factor/v1/OCRAVerifier`
- To know the value of ${SSP_FQDN} environment variable, see Configuring the Environment Variables.
- The default value for `<tenantName>` is "default".

Request and Response Samples

- Perform a POST request using the following admin API

```
POST https://${SSP_FQDN}/factor/v1/OCRATransactionVerifier
```

API Request

This request shows .....

```
{
    "credId": "{{mobileOTPCredId}}",
    "otp":"784715",
    "challengeData": "1111"
}
```

Where,

– **credId** - Specify the credential ID that has been generated by Symantec Identity Security Platform during the registration flow of MobileOTP credential.
– **otp** - Specify the TOTP generated by the authenticator app.
– **challengeData** - Specify the challenge value provided by the server that must be signed by the user's authentication device.

API Response:

– This is an example of a successful transaction verification.

```
{
"status": "OK",
"message": "Transaction verified successfully."
```

```
}
```

Where,

- **status** - Displays the status as "OK" if the transaction was verified succefully.
- **message** - Description of this message is only populated when API is successful.

– This is an example of an unsuccessful transaction verification.

```
{
"errorCode": "3001033",
"errorMessage": "Transaction verification failed."
}
```

Where,

- **errorCode** - Specifies the error code when an error occurs.
- **errorMessage** - Description of the error is only populated when an error occurs.

Self Service APIs

Learn about Self Service APIs in Symantec Identity Security Platform.

As an end user, you can manage your own credentials using the Self Service APIs. Tasks such as retrieving your passwords, updating your credentials, etc. can be performed. You can also manage the registration and authentication of SMSOTP, Push, and other supported factors.

The solution provides a range of Self Service APIs to make your experience smoother and quicker without the support of an administrator.

The following table lists out all the Self Service APIs that are made available to you to manage your own credentials. The type of operations that you are authorized to perform are based on the scope, “urn:iam:m.me" and all the relevant APIs are mapped to this scope. You are authorized to use only those specific APIs.

Operation Description	Operation Type	Endpoint URLs
Credential Management APIs		
Retrieving all credentials of the user	GET	https://${IDSP_FQDN}/{{tenantName}}/fa
Retrieving all email credentials of the user	GET	https://${IDSP_FQDN}/{{tenantName}}/fa
Retrieving all sms credentials of the user	GET	https://${IDSP_FQDN}/{{tenantName}}/fa
Retrieving all totp credentials of the user	GET	https://${IDSP_FQDN}/{{tenantName}}/fa
Updating the User credential by specifying the cred Id	PUT	https://${IDSP_FQDN}/{{tenantName}}/fa
Deleting user email credential by specifying the cred Id	DELETE	https://${IDSP_FQDN}/{{tenantName}}/fa
Deleting user sms credential by specifying the cred Id	DELETE	https://${IDSP_FQDN}/{{tenantName}}/fa
Deleting user push credential by specifying the cred Id	DELETE	https://${IDSP_FQDN}/{{tenantName}}/fa
Deleting user totp credential by specifying the cred Id	DELETE	https://${IDSP_FQDN}/{{tenantName}}/fa
Deleting user fido credential by specifying the cred Id	DELETE	https://${IDSP_FQDN}/{{tenantName}}/fa
My Trusted Devices		
Retrieving all the user trusted devices	GET	https://${IDSP_FQDN}/{{tenantName}}/a
Retrieving user trusted device by specifying the device Id	GET	https://${IDSP_FQDN}/{{tenantName}}/a
Updating user trusted device by specifying the device Id	PUT	https://${IDSP_FQDN}/{{tenantName}}/a
Deleting user trusted device by specifying the device Id	DELETE	https://${IDSP_FQDN}/{{tenantName}}/a
Managing Push Credentials		

Registering Push on the user device	POST	https://${IDSP_FQDN}/{{tenantName}}/fa
Generating Push notification onto the user device	POST	https://${IDSP_FQDN}/{{tenantName}}/fa
Check the status of the push registration	POST	https://${IDSP_FQDN}/{{tenantName}}//f
Managing SMSOTP Credentials		
Registering user mobile number with server OTP	POST	https://${IDSP_FQDN}/{{tenantName}}/fa
Generating OTP onto user mobile	POST	https://${IDSP_FQDN}/{{tenantName}}/fa
Verifying user mobile with server OTP	POST	https://${IDSP_FQDN}/{{tenantName}}/fa
Managing EMAILOTP Credentials		
Registering user email Id with server OTP	POST	https://${IDSP_FQDN}/{{tenantName}}/fa
Generating OTP to your email Id	POST	https://${IDSP_FQDN}/{{tenantName}}/fa
Verifying server OTP received in the user email Id	POST	https://${IDSP_FQDN}/{{tenantName}}/fa
Managing TOTP (OAuth) Credentials		
Registering mobile OTP onto user's mobile device	POST	https://${IDSP_FQDN}/{{tenantName}}/fa
Verifying mobile OTP received on the user's registered mobile device	POST	https://${IDSP_FQDN}/{{tenantName}}/fa
Managing TOTP (VIP) Credentials		
Registering user device with the VIP Access app	POST	https://${IDSP_FQDN}/{{tenantName}}/fa
Verifying Mobile OTP received in user's registered device	POST	https://${IDSP_FQDN}/{{apiPathTenant}}
Managing IA Risk		
Deleting all the devices belonging to the user	DELETE	https://${IDSP_FQDN}/{{tenantName}}/ia
Managing User Passwords		
Updating user passwords	PUT	https://${IDSP_FQDN}/{{apiPathTenant}}
Session Management API		
Revoking my Sessions and Refresh Tokens	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}
Get my Sessions	GET	https://${IDSP_FQDN}/{{apiPathTenant}}
Revoking my Sessions	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}
Get My Session by Session ID	GET	https://${IDSP_FQDN}/{{apiPathTenant}}
Revoke My Session by Session ID	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}
Get my Refresh Tokens	GET	https://${IDSP_FQDN}/{{apiPathTenant}}
Revoke my Refresh Tokens	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}
Get My Session by Session ID	GET	https://${IDSP_FQDN}/{{apiPathTenant}}
Revoke My Session by Session ID	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}
My Admin Roles API		
Returns my list of admin capabilities.	GET	https://${IDSP_FQDN}/{{apiPathTenant}}

NOTE

- To know the value of ${IDSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:

`https://{{host}}/{{tenantName}}/auth/v1/MeCreds`
should look similar to:
`https://example.com/default/auth/v1/MeCreds`

- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Self Service Console APIs

You can use the Self Service Console APIs to manage how the Self Service Console sends notifications, displays apps, and organizes the apps into different groups (also called sections).

The Self Service Console allows you to manage all of the apps to which you or your end users are authorized or granted access. As an administrator, you can use the Self Service Console APIs to set how the Self Service Console displays apps and groups and sends notifications for all end users. As an end user, you can set how your Self Service Console displays apps and groups and sends notifications. See the appropriate resource:

Table 101: Self Service Console APIs

Operation	As an end user, see:	As an administrator, see:
Manage how the apps that are displayed (display order, tile size and layout, and whether to display recently used apps)	Self Service Console Preferences API (for End Users)	Self Service Console UserPreferences API (for Administrators)
Manage how the apps are grouped (create, rename, or remove groups of apps and add or remove apps from the groups)	Self Service Console AppTiles API (for End Users)	Self Service Console UserAppTiles API (for Administrators)
View notifications sent to me as an end user	Self Service Console MyNotifications API	N/A
Manage Self Service Center notifications for all end users	N/A	Self Service Console Notifications API

Self Service Console Preferences API (for End Users)

Use the Self Service Console Preferences API to set how your apps appear in the Self Service Console.

By default, all the applications that are authorized or granted to you appear (as app tiles) on the Self Service Console dashboard. As an end user, you can manage how these apps appear on the Self Service Console dashboard using the Preferences API. The Preferences API lets you set:

- whether the apps appear in ascending or descending alphabetical order
- the size of the app tiles (small, medium, or large)
- whether the apps are laid out in a grid or a list

If you have used the AppTiles API to create groups of apps, the Preferences API affects the display of the group tiles in the same way that they affect the display of the app tiles. See Self Service Console AppTiles API (End Users).

User Preferences Management	GET PATCH	`https://${SSP_FQDN}/{{tenantName}}/admin/v1/UserPreferences`
	GET	`GET /admin/v1/UserPreferences` Returns the default display preferences for end users.

	PATCH	`PATCH /admin/v1/UserPreferences` Adds or updates user display preferences. This operation requires at least one of the following optional display parameters. You can include one, some, or all display preferences in a single operation. • `appTileDisplay` • `appTileSize` • `appTileLayout`

		Optional PATCH parameter: `appTileDisplay` : Sets whether the apps are set in ascending or descending alphabetical order.
		• `ascending` : apps are displayed in ascending alphabetical order (a to z)
		• `descending` : apps are displayed in descending alphabetical order (z to a)
		`{`
		`"name": "urn:iam:userpreferences:appTileDisplay",`
		`"value": "ascending"`
		`}`

		Optional PATCH parameter: `appTileSize` : Sets the size of the app tiles:
		• `small` : Displays small app tiles
		• `medium` : Displays medium app tiles
		• `large` : Displays large app tiles
		`{`
		`"name": "urn:iam:userpreferences:appTileSize",`
		`"value": "medium"`
		`}`

		Optional PATCH parameter: `appTileLayout` : Sets whether the apps appear in a grid or in a list.
		• grid: Apps (or groups) appear in a grid
		• list: Apps (or groups) appear in a list
		`{`
		`"name": "urn:iam:userpreferences:appTileLayout",`
		`"value": "grid"`
		`}`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values in all the endpoints shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/scim2/v1/Users`
  should look similar to:
  `https://example.com/default/scim2/v1/Users`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Self Service Console UserPreferences API (for Administrators)

Use the Self Service Console UserPreferences API to set how apps appear in the Self Service Console for all users.

By default, all the applications that are authorized or granted to an end user appear (as app tiles) on the Self Service Console dashboard for that end user. As an administrator, you can manage how these apps appear to end users on the Self Service Console dashboard using the Preferences API. The Preferences API lets you set the following display preferences as the default for all users:

- whether the apps appear in ascending or descending alphabetical order
- the size of the app tiles (small, medium, or large)
- whether the apps are laid out in a grid or a list

If an end user sets their own display preferences using the Self Service Console Preferences API (for End Users), the Self Service Console honors those preferences over the default preferences that you set here.

If you have used the AppTiles API to create default groups of apps for end users, the Preferences API affects the display of the group tiles in the same way that they affect the display of the app tiles. See Self Service Console AppTiles API (for End Users).

User Preferences Management	GET PATCH	`https://${SSP_FQDN}/{{tenantName}}/admin/v1/UserPreferences`
	GET	`GET /admin/v1/UserPreferences` Returns the default display preferences for end users.
	PATCH	`PATCH /admin/v1/UserPreferences` Adds or updates user display preferences. This operation requires at least one of the following optional display parameters. You can include one, some, or all display preferences in a single operation. • `appTileDisplay` • `appTileSize` • `appTileLayout`

		Optional PATCH parameter: `appTileDisplay` : Sets whether the apps are set in ascending or descending alphabetical order.
		• `ascending` : apps are displayed in ascending alphabetical order (a to z)
		• `descending` : apps are displayed in descending alphabetical order (z to a)
		`{`
		`"name": "urn:iam:userpreferences:appTileDisplay",`
		`"value": "ascending"`
		`}`

		Optional PATCH parameter: `appTileSize` : Sets the size of the app tiles:
		• `small` : Displays small app tiles
		• `medium` : Displays medium app tiles
		• `large` : Displays large app tiles
		`{`
		`"name": "urn:iam:userpreferences:appTileSize",`
		`"value": "medium"`
		`}`

		Optional PATCH parameter: `appTileLayout` : Sets whether the apps appear in a grid or in a list.
		• grid: Apps (or groups) appear in a grid
		• list: Apps (or groups) appear in a list
		`{`
		`"name": "urn:iam:userpreferences:appTileLayout",`
		`"value": "grid"`
		`}`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values in all the endpoints shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/scim2/v1/Users`
  should look similar to:
  `https://example.com/default/scim2/v1/Users`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Self Service Console AppTiles API (for End Users)

Use the Self Service Console AppTiles API to display groups of apps on your Self Service Console.

By default, all the applications that are authorized or granted to you appear (as app tiles) on the Self Service Console dashboard. As an end user, you can manage how these apps are grouped on the Self Service Console dashboard using the AppTiles API. The AppTiles API lets you create and name groups of application tiles, delete groups of application tiles, and rename groups of application tiles. The AppTiles API also lets you add or remove apps from the groups that you create.

App Tiles Management	GET PATCH DELETE	`https://${SSP_FQDN}/{{tenantName}}/me/v1/AppTiles`
	GET	`GET /me/v1/AppTiles` Returns all authorized applications for the user.

	PATCH	`PATCH /me/v1/AppTiles`
		Creates or updates a group. This operation requires the following parameters:
		• `name` : a unique name for the group
		• `order` : the order that it appears on the dashboard
		If adding apps to this group (either at creation or in a later operation), also include the `app` parameter. The `app` parameter requires the following parameters:
		• `id` : the app ID
		• `name` : a unique name for the app
		• `order` : The order that it appears in the group
		For example, to create a group named*Favorites* which includes one app that is called *Mobile Banking*:
		`[`
		`{`
		`"name": "Favorites",`
		`"order": 3,`
		`"apps": [`
		`{`
		`"id": "e9b39fef-c91d-4b19-8c37-50ee4c0b8919",`
		`"name": "Mobile Banking",`
		`"order": 1`
		`}`
		`]`
		`}`
		`]`

		`PATCH /me/v1/AppTiles/Group?groupName={{existingGroupName}}` Renames the existing group. This operation requires the following parameters: • In the request query: `existingGroupName` . The current name of the group to be renamed • In the request body: `name` . The new name for the group.
	DELETE	`DELETE /me/v1/AppTiles/Group?groupName={{groupName}}` Deletes the named group. The group can only be deleted if it is empty (without applications).

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values in all the endpoints shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/scim2/v1/Users`
  should look similar to:
  `https://example.com/default/scim2/v1/Users`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

A sample AppTiles POST request (adding the group *Favorites* which includes one app that is called *Mobile Banking*)**:**

```
[
{
"name": "Favorites",
"order": 3,
"apps": [
{
"id": "e9b39fef-c91d-4b19-8c37-50ee4c0b8919",
"name": "Mobile Banking",
"order": 4
}
]
}
]
```

A sample AppTiles POST response:

```
{
"groups": [
{
"name": "Favorites",
"order": 3,
"apps": [
{
"id": "e9b39fef-c91d-4b19-8c37-50ee4c0b8919",
"name": "Mobile Banking",
"order": 4
}
]
},
{
```

Self Service Console UserAppTiles API (for Administrators)

Use the Self Service Console UserAppTiles for Administrators API to display groups of apps on your Self Service Console.

By default, all the applications that are authorized or granted to you appear (as app tiles) on the Self Service Console dashboard. As an administrator, you can use the UserAppTiles API to manage how these apps are grouped for your end users on the Self Service Console dashboard. The UserAppTiles API lets you create and name groups of application tiles, delete groups of application tiles, and rename groups of application tiles. The UserAppTiles API also lets you add or remove apps from the groups that you create.

If an end user sets their own display preferences using the Self Service Console AppTiles API (for End Users), the Self Service Console honors those preferences over the default preferences that you set here.

App Tiles Management	GET PATCH DELETE	`https://${SSP_FQDN}/{{tenantName}}/admin/v1/UserAppTiles?userId=<userId>`
	GET	`GET /admin/v1/UserAppTiles?userId={{userId}}` Returns all authorized applications for the specified end user.

	PATCH	`PATCH /admin/v1/UserAppTiles?userId={{userId}}`
		Creates or updates a group for the specified user. This operation requires the following parameters:
		• `name` : a unique name for the group
		• `order` : the order that it appears on the dashboard
		If adding apps to this group (either at creation or in a later operation), also include the `app` parameter. The `app` parameter requires the following parameters:
		• `id` : the app ID
		• `name` : a unique name for the app
		• `order` : The order that it appears in the group
		For example, to create a group named*Favorites* which includes one app that is called *Mobile Banking*:
		`[`
		`{`
		`"name": "Favorites",`
		`"order": 3,`
		`"apps": [`
		`{`
		`"id": "e9b39fef-c91d-4b19-8c37-50ee4c0b8919",`
		`"name": "Mobile Banking",`
		`"order": 1`
		`}`
		`]`
		`}`
		`]`

		`PATCH /admin/v1/UserAppTiles/Group?groupName={{existingGroupName}}&userId={{userId}}` Renames the existing group for the specified user. This operation requires the following parameters: • In the request query: – `existingGroupName` . The current name of the group to be renamed. – `userId` . The ID of the user whose group you are renaming. • In the request body: `name` . The new name for the group.
	DELETE	`DELETE /admin/v1/UserAppTiles/Group?groupName={{existingGroupName}}&userId={{userId}}` Deletes the named group for the specified user. The group can only be deleted if it is empty (without applications). This operation requires the following parameters: • In the request query: – `existingGroupName` . The current name of the group to be renamed – `userId` . The ID of the user whose group you are renaming. • In the request body: `name` . The new name for the group.

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values in all the endpoints shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/scim2/v1/Users`
  should look similar to:
  `https://example.com/default/scim2/v1/Users`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

A sample AppTiles POST request (adding the group *Favorites* which includes one app that is called *Mobile Banking*)**:**

```
[
{
 "name": "Favorites",
 "order": 3,
 "apps": [
   {
     "id": "e9b39fef-c91d-4b19-8c37-50ee4c0b8919",
     "name": "Mobile Banking",
     "order": 4
   }
  ]
}
]
```

A sample AppTiles POST response:

```
{
"groups": [
{
  "name": "Favorites",
  "order": 3,
  "apps": [
    {
       "id": "e9b39fef-c91d-4b19-8c37-50ee4c0b8919",
       "name": "Mobile Banking",
       "order": 4
    }
   ]
},
]
```

Self Service Console MyNotifications API

Use the Self Service Console MyNotifications API to view all Self Service Console notifications sent to you as an end user.

The Self Service MyNotifications API returns Self Service Console notifications sent to the end user from one of the apps that are assigned or granted to the end user and which are not expired.

App Tiles Management	GET	`https://${SSP_FQDN}/{{tenantName}}/me/v1/MyNotifications`
	GET	`GET /me/v1/MyNotifications` Returns all unexpired Self Service Console notifications for the end user.

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values in all the endpoints shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/scim2/v1/Users`
  should look similar to:
  `https://example.com/default/scim2/v1/Users`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

A sample MyNotifications GET response:

```
[
{
  "id": "45f6da0e-7535-4815-b6af-353ef2866b0c",
  "subject": "Maintenance Message",
  "priority": "low",
  "message": "Sales Tools will be unavailable until further notice",
  "audience": "urn:iam:aud:type:app=SalesTools",
  "startDate": "2025-09-25 00:00:00",
  "expiryDate": "2025-09-26 17:00:00"
},
{
  "id": "aab56c6f-1130-48f4-ba3f-7f7c84b08c0d",
  "subject": "Maintenance Message",
  "priority": "low",
  "message": "Human Resources website will be unavailable until further notice.",
  "audience": "urn:iam:aud:type:app=HRWebsite",
  "startDate": "2025-09-25 00:00:00",
  "expiryDate": "2025-10-02 00:00:00"
}
]
```

Self Service Console Notifications API

Use the Self Service Console Notifications API as an administrator to manage all Self Service Console notifications.

The Self Service Console provides three types of notifications:

- App addition or removal: Notification that an app is added or removed from the user. This notification is sent only to the affected user.
- App-specific notification: Notification specific to an application. This notification is sent only to users that are authorized or are granted access to the application.
- Generic notification: Notification sent to all users.

Notifications of app addition or removal are generated by the Self Service Console, and cannot be managed by an administrator. However, as an administrator, you can use the Self Service Notifications API to manage the other Self Service Console notifications. With this API you can:

- View all non-expired notifications (including app addition or app removal notifications), and filter notifications by a specific user or users, or by a specific application or applications.
- Create notifications to send to all users or applications (generic notifications), or to specific users or applications (app-specific notifications).
- Delete notifications (except app addition or removal notifications).

App Tiles Management	GET PATCH DELETE	`https://${SSP_FQDN}/{{tenantName}}/admin/v1/Notifications`
	GET	`GET /admin/v1/Notifications` Returns all unexpired notifications.
		Optional search parameter: `id` . Provide the notification ID (created in the response to the POST request) to return specific notifications. `GET /admin/v1/Notifications?id=e9b39fef-c91d-4b19-8c37-50ee4c0b8919`
	POST	`POST/admin/v1/Notifications` Creates a notification. This operation requires the following parameters: • `audience` : who should receive the notification: – `urn:iam:aud:type:user=*`: Send to notification to all users. – `urn:iam:aud:type:app=<app-id-or-name>`: Send to users assigned or granted access to the app specified by the appId or app name. • `subject` . Sets the subject line of the notification. • `message` . Sets the content of the body of the notification. The message can contain up to 1000 characters and can include HTML tags. The character limit can be overridden by the maxAllowedNotificationSize tenant setting. (see Managing Settings). • `startDate` : the date and time that the notification should be sent, in the following format: YYYY-MM-DD hh:mm:ss. The time is in 24-hour format. The `startDate` can be in the future. The response to the POST request includes a unique notification ID. Use the ID to view or delete specific notifications.

		Optional POST parameter: `expiryDate` . Sets when the notification is removed, in the following format: YYYY-MM-DD hh:mm:ss. The time is in 24-hour format. The `expiryDate` cannot be earlier than the `startDate` or the current date. If this parameter is not provided, the API uses the tenant default expiration set in appSpecificNotificationExpiryDaysForSSC or toAllUsersNotificationExpiryDaysForSSC, depending on the audience value set for the notification (see Managing Settings).
		Optional POST parameter: `priority` . Sets the priority of the message. Priorities can be one of the following options: • `low` (the default) • `medium` • `high`
		Optional POST parameter: `notificationType` . Sets the notification as a banner on the Self Service Console. Only include this parameter if the notification should be a banner. This parameter takes the value, `urn:iam:notification:type:banner` .
	DELETE	`DELETE /admin/v1/Notifications/{{id}}` Deletes the specified notification.

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values in all the endpoints shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/scim2/v1/Users`
  should look similar to:
  `https://example.com/default/scim2/v1/Users`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

A sample Notifications POST request. This notification is a high priority banner notification.

```
{
"subject": "Maintenance Message",
"priority": "high",
"audience": "urn:iam:aud:type:user=*",
"notificationType": "urn:iam:notification:type:banner",
"message": "The Sales Tool is undergoing maintenance and will not be available until further notice.",
"startDate": "2025-10-08 05:00:00"
"expiryDate": "2025-10-15 05:00:00"
}
```

A sample Notifications POST response:

```
{
"id": "dd6ac104-0d39-4bcd-967f-e5e2a298d7b1",
"subject": "Maintenance Message",
"priority": "high",
"message": "The Sales Tool is undergoing maintenance and will not be available until further notice.",
"audience": "urn:iam:aud:type:user=*",
```

```
"notificationType": "urn:iam:notification:type:banner",
"startDate": "2025-10-08 05:00:00",
"expiryDate": "2025-10-15 05:00:00"
}
```

Management APIs

Know about Symantec Identity Security Platform management APIs.

Symantec Identity Security Platform offers a managed approach to the solution's administrative access with the option of enabling multi-factor authentication for additional security. With IDSP, you can designate limited administrators to manage the authorization and authentication flows. Admins can be assigned for purposes such as configuring LDAP, adding applications, creating policies, assigning tenant privileges to the users, managing RP domain names, etc.

As an administrator, you learn how to administer and maintain the daily tasks to keep the IDSP environment up-to-date and stable.

The following links take you to the respective documentation of the resources that you are configuring and administering:

- RESTAPI Filter
- LDAP Configuration
- SCIM Configuration
- Managing Applications
- Managing Policies
- Managing Signing Keys
- Managing Certificates
- Managing Secrets
- Managing IDSP Risk Engine
- Managing Credentials
- Monitoring Services
- Managing Settings
- Managing User Operations
- VIP Configuration Settings
- Managing Message Templates
- Managing Provider Configuration
- Managing Tenants
- Managing Service Accounts
- Managing Token Revocation
- Managing Scoping Constraints
- Managing Sessions
- Auth Policies Evaluator

REST API Filter

Use REST API filters to query resource collections and retrieve specific subsets of data based on defined conditions.

Filtering enables you to narrow down large result sets by applying conditional queries to a resource collection. This allows for more efficient data management by ensuring the API returns only the records that meet your specific criteria.

This topic includes the following sections:

- Table displaying supported filter operators
- Table displaying supported logical operators
- REST API - Filter Attributes
- Filter Limitations
- Filter Examples

Table displaying supported filter operators

The following table lists the filter operators that can be used in Symantec Identity Security Platform REST APIs:

Operator	Description	Behavior
eq	equal	The attribute and operator values must be identical for a match.
ne	not equal	The attribute and operator values are not identical.
co	contains	The entire operator value must be a substring of the attribute value
sw	starts with	The entire operator value must be a substring of the attribute value, the two strings are identical.
ew	ends with	The entire operator value must be a substring of the attribute value, two strings are identical.
pr	present (has value)	If the attribute has a non-empty or (has value) non-null value, or if it
gt	greater than	If the attribute value is greater than the operator value, there is a ma
ge	greater than or equal to	If the attribute value is greater than or equal to the operator value, t
lt	less than	If the attribute value is less than the operator value, there is a match
le	less than or equal to	If the attribute value is less than or equal to the operator value, ther

Table displaying supported logical operators

The following table lists the logical operators supported:

Operator	Description	Behavior
and	Logical "and"	The filter is only a match
or	Logical "or"	The filter is a match if ei

REST API Attributes - Applying Filters

List of API attributes where filter can be applied:

API name and resource	Filter Attributes
Get All Application https://{{sspHost}}/{{tenantName}}/admin/v1/Apps	1. name 2. description 3. status 4. clientType 5. appId 6. clientId
Get All Certificates https://{{sspHost}}/{{tenantName}}/admin/v1/Certs	1. tokenIssuer 2. certificateId 3. alias 4. issuerDN 5. subjectDN 6. fipsCompliant 7. keyAlgorithm 8. serialNumber
Get All AuthN Policies https://{{sspHost}}/{{tenantName}}/admin/v1/AuthPolicies	1. policyId 2. policyName 3. description
Get All AuthZ Policies https://{{sspHost}}/{{tenantName}}/admin/v1/AuthZPolicies	1. policyId 2. policyName 3. description 4. status
Get all Credentials of User https://{{sspHost}}/{{tenantName}}/admin/v1/Creds/{{userId}}	1. credId 2. userId 3. status 4. credType 5. default
Get RP List https://{{sspHost}}/{{tenantName}}/admin/v1/FIDORelyingParties	1. rpName 2. rpId 3. rpDomainName
getLdapConfig https://{{sspHost}}/{{tenantName}}/admin/v1/LDAPconfigs	1. configName 2. configDescription 3. sslEnabled 4. hostName 5. portNumber 6. baseDN 7. groupBaseDN 8. userDN 9. ldapConfigId
getUsers https://{{sspHost}}/{{tenantName}}/admin/v1/Users	1. userId 2. userLoginId 3. userUniversalId 4. idpUserGuid 5. status 6. identitySourceId 7. identitySourceType 8. identitySourceName
getMessages https://{{sspHost}}/{{tenantName}}/admin/v1/Messages	1. messageId 2. action 3. messageType 4. language 5. message
getCredentials	1. credId

Filter Limitations

- Filter is not supported in nested objects.
- Filter based on data type is not supported. For example- Filtering based on date will not fetch any results.

Filter Examples

This section includes examples of filter from different resource categories.

Filter by Policies

This example displays the filter query that you can use to filter results for an Authentication Policy API:

```
https://${SSP_FQDN}/{{tenantName}}/admin/v1/AuthPolicies?filter=policyId co 3 and policyName co a and
 policyName eq AdminLoginPolicy
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/AuthPolicies?filter=policyId co 3 and policyName co a and policyName eq AdminLoginPolicy`
  should look similar to:
  `https://example.com/default/admin/v1/AuthPolicies?filter=policyId co 3 and policyName co a and policyName eq AdminLoginPolicy`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Filter by Messages

This example displays the filter query that you can use to filter results for the Messages API:

```
https://${SSP_FQDN}/{{tenantName}}/admin/v1/Messages?filter=( action eq LOGIN ) and messageType co EMAIL
https://${SSP_FQDN}/{{tenantName}}/admin/v1/Messages?filter=( action eq LOGIN and messageType ne IVR ) and
 messageId co 2d0
https://${SSP_FQDN}/{{tenantName}}/admin/v1/Messages?filter=messageId ew 78a
```

Managing LDAP Directory Configuration

Learn how to configure LDAP Directory in Symantec Identity Security Platform.

This topic includes the following sections:

- LDAP Configuration APIs
- LDAP Attributes Descriptions
- Attribute Mapping Table
- User Status Table
- LDAP Connection Pool
- LDAP lookup Query
- Optional Parameters

IDSP uses an LDAP v3 compliant directory store to look up and validate users and groups. The LDAP configuration contains the information that allows the LDAP service to connect to the LDAP directory and perform search operations.

LDAP Configuration APIs

The following table displays the list of APIs that can be used by an administrator to configure an LDAP v3 compliant directory in the solution:

Operation Description	Operation Type	Endpoint URLs
Create LDAP Configuration	POST	https://${SSP_FQDN}/{{tenantName}}/admin/v1/L
Retrieve all LDAP configurations	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/L
Retrieve LDAP configuration by specifying the LDAP Id	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/L
Update the LDAP configuration by specifying the LDAP Id	PUT	https://${SSP_FQDN}/{{tenantName}}/admin/v1/L
Delete an LDAP configuration by specifying the LDAP Id	DELETE	https://${SSP_FQDN}/{{tenantName}}/admin/v1/L

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/LDAPconfigs`
  should look similar to:
  `https://example.com/default/admin/v1/LDAPconfigs`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

A sample LDAP config request payload looks like the following:

```
{
    "configName": "TestConfig",
    "configDescription": "CA DIR k8s with default data",
    "sslEnabled": "false",
    "certificateAlias": "mysslcert.pem",
    "ldapQueryStart": "(|(cn=${loginid})(mail=${loginid})(telephoneNumber=${loginid}))",
    "ldapQueryEnd": "",
    "userObjectClasses": "inetorgperson",
    "hostName": "cadir-ssp-symantec-dir.idstore.svc.cluster.local",
    "portNumber": "1389",
    "baseDN": "o=democorp,c=us",
    "groupBaseDN": "o=democorp,c=us",
    "groupNameAttributeMappingName": "name",
    "groupObjectClasses": "groupOfNames",
    "useMemberOf": "true",
    "statusEncoding": "SITEMINDER",
    "isAuthoritativeForIDLCM": "true",
    "bindDN": "cn=DominicMAJOR,ou=Administration,ou=Corporate,o=democorp,c=US",
    "bindPassword": "password",
    "bindPasswordSecretAlias": "secretAlias",
    "userLoginIdAttributeMappingName": "user_loginid",
    "attributeMapping": "user_loginid=cn, family_name=sn, given_name=givenName, email=mail,
 phone_number=telephoneNumber, user_universalid=cn, status=employeeType, memberof=memberof",
    "readTimeoutMillis": "10000",
    "connectTimeoutMillis": "10000",
```

```
    "poolEnabled": "true",
    "groupAttributeMapping": "name=cn, members=member",
    "isAuthoritativeForIDLCM": "false",
    "isAuthoritativeForUserCredsLCM": "false",
    "passwordResetURL": "http://userPasswordResetURL"
}
```

For sample request and response payloads of all these URLs, refer to LDAP Configuration API- Swagger documentation

List of attributes and their descriptions used in LDAP configuration

Table 102: LDAP configuration attributes and their descriptions

Attribute Name	Mandatory Fields	Description	Example values
configName	No	Name of the LDAP configuration to be created	"TestConfig"
configDescription	No	Short description of the LDAP configuration so that it is easily identifiable	"Test LDAP configu
sslEnabled	Yes	This parameter enables SSL, which secures the communication between the LDAP server and LDAP authenticator. Default Value is set as "**False**".	"true"
certificate This attribute has been deprecated and is only supported for backward compatibility. Use certificateAlias instead.	No	This is the CA signed certificate for SSL. (This attribute should be "**Yes**", if **SSL** is enabled) (The certificate should be in Base64 encoded DER format)	
certificateAlias	No	Name of the entry in the certificate store containing the SSL cert. See Certs API.	"mysslcert.pem"
ldapQueryStart	Yes	This is the beginning of a query to locate the user entry in LDAP. The end of the LDAP query will be appended to it. If the final query does not contain the substitution marker, $loginid then one is added between the beginning and the end part. The names of the attributes used in the query are physical attribute names. See examples here. This value can contain dynamic environment variables which will be substituted with their corresponding variable value, when the query is used. See Dynamic Environment Variable Substitution.	"(cn=${loginid}"
ldapQueryEnd	Yes	This is the end of a query. The names of the attributes used in the query are physical attribute names. This value can contain dynamic environment variables which will be substituted with their corresponding variable value, when the query is used. See Dynamic Environment Variable Substitution.	")"
userObjectClasses	Yes	The class name that holds attributes about users in the LDAP server.	"inetorgperson"

hostName	Yes	Name of the host where the LDAP server is deployed. In case, both IDSP and the directory service is running on the same Kubernetes cluster, use the FQDN of the Kubernetes service, in this format: **dir-servicename.dir-namespace.svc**	"sampleapp-ssp-sy
portNumber	Yes	Port number of the LDAP server	"12345"
baseDN	Yes	This is the distinguished name of the container where all the users are stored.	"o=democorp,c=us'
groupBaseDN	No	This is the base DN that is used for group search. If the groupBaseDN is not specified, the value of baseDN will be used instead.	"o=democorp,c=us'
groupNameAttributeMappingName	No	This attribute contains the name of the logical mapping attribute representing the group name. This value should have a mapping from the logical to the physical attribute.	"name"
groupObjectClasses	Yes	The class name that holds the list of comma-separated values that are used to search the group names.	"groupOfNames"
groupAttributeMapping	No	This is a comma-separated list of key-value pairs representing the mapping between logical attributes and physical LDAP attributes for a group.	"name=cn, member
useMemberOf	No	This attribute is used to determine if 'memberOf' is enabled in the LDAP or not. • If the value is **true**, then the 'memberof' attribute of users who are defined in LDAP is used to get the group information. You can also use this behavior to obtain a list of values for custom attributes. • If the value is **false**, then the group membership cannot be determined.	
statusEncoding	No	This attribute specifies the type of the LDAP Directory being configured. This attribute is used by IDSP to perform native LDAP operations such as User Lockout. If the value is not specified, User Lockout operations fail unless the logical "status" attribute mapping is defined by LDAPconfig's "attributeMapping" property. • By default, this attribute is set to "SITEMINDER". • See Supported User Status behavior table. • See User Locker	Possible Values: AD INTERNAL • For SiteMinder, must be defined property, mappi representing "S • "INTERNAL" is implementations disabled user st – For INTERN mappings, "**i** within the "**a** mappings m the user lock Upon user d determined b mapped field "is_locked=l

idStoreMappingStrategy	No	The value of this attribute contains the mapping strategy for mapping the disambiguation Id Store to the authentication Id Store.
		See ID Store Mapping in Password Authentication.
		This attribute value can contain dynamic environment variables which will be substituted with their corresponding variable value, when this attribute is used.
		See Dynamic Environment Variable Substitution.
			`"attribute_ma`
			`"user_loginid`
			`family_name=s`
			`middle_name=g`
			`nickname=disp`
			`preferred_use`
			`profile=title`
			`phone_number=`
			`user_universa`
			`status=employ`

isAuthoritativeForIDLCM	No	This attribute specifies if the LDAP that is being configured, is also the authoritative source for user locking. This is a boolean string with a value of true/false. If this value is set to "false", then IDSP becomes the authoritative locking source and the LDAP that has been configured will not be the locking source. The user locking is done when the user strike count reaches the maximum allowed value. If this value is set to "true", the user lock is done in the user store.	
isAuthoritativeForUserCredsLCM	No	This attribute specifies if the LDAP ID Store that is being configured, is also the authoritative source for user's credentials. If this value is true, the user's OTP creds that is established during inline registration would be synced back to the id store.	
isAuthoritativeForDetectingLockedState	No	This boolean attribute specifies if the LDAP ID Store that is configured, is also the authoritative source for detecting the locked state of the user or not. If this value is true, the user's locked state in the LDAP ID Store will be used, otherwise the user's internal locked state is used.	
bindDN	No	This is the distinguished name of the user that is used to connect to the LDAP directory server.	"cn=admin,o=demo
bindPassword This attribute has been deprecated and is only supported for backward compatibility. Use bindPasswordSecretAlias instead.	No	This is the password of the user whose dn is passed in the "bindDN" attribute. The "bindDN" and "bindPassword" are used to connect to the LDAP directory server.	"password"
bindPasswordSecretAlias	No	If a user creates a secret with the "bindPassword" and wants to use this secret, then the user can define an alias in the "bindPasswordSecretAlias" attribute. • **Precedence:** When both "bindPasswordSecretAlias" and "bindPassword" are provided for LDAP configuration, "bindPassword" takes precedence and is used. If "bindPassword" is not provided, "bindPasswordSecretAlias" is used. • **Visibility**: In cases where both "bindPassword" and "bindPasswordSecretAlias" are supplied during LDAP configuration, only the "bindPasswordSecretAlias" value is displayed in responses from GET, POST, and PUT endpoints; the "bindPassword" value is not shown.	"secretAlias"

bindControlsCsv		This attribute contains comma-separated list of LDAP connection controls that are applied when establishing an LDAP connection. It serves two main purposes: 1. LDAP Connection Controls: When connecting to an LDAP server, this field allows you to specify custom LDAP controls that should be sent during the bind operation. These controls are used to modify the behavior of the LDAP connection. 2. SiteMinder Integration: This field also supports a special syntax for integrating with SiteMinder (CA Single Sign-On) for password validation. If a control starts with {sm} (the SiteMinder provider prefix), it is treated as a SiteMinder configuration ID.	" \"bindControlsCsv +
userLoginIdAttributeMappingName	Yes	This attribute contains the name of the logical mapping attribute representing the user's unique id. This value should have a mapping from the logical to physical attribute.	"true", "false"
attributeMapping See the table for more info.	No	This is a comma-separated string value listing all the mappings of logical attribute names to their physical LDAP attribute. This applies to Active Directory only. The "status" attribute is used to determine whether the user is marked INACTIVE. The attribute follows the standard "UserAccountControl" flag values. If the value in the user status field is set to 514, the user is deemed as INACTIVE. This value is evaluated as a bitmask and if the bits representing 514 are not set then the status will be deemed as ACTIVE. If you leave the attributeMapping value empty, the user's First Name and Last Name will not appear and the status will appear as 'UNKNOWN'.	"user_loginid=cn,na

mvAttributeMapping	No	This attribute contains a formatted JSON string that defines the mapping of logical attribute (like email and phone) to multiple single-valued LDAP attributes, each with an assigned tag (for example, 'business', 'personal'), to differentiate credentials.
		When using MVA (Multi-Valued Attributes), you need to remove "mail" and "telephoneNumber" from the attributeMapping configuration to take effect. Otherwise, attributeMapping values are considered.
		For example, if LDAP has two single-valued attributes, "mail" for business email and "personalmail" for personal email, they can be mapped to the logical "email" attribute, configured in the following manner:
		`"email":[`
		`{"attribute": "mail", "tag":"business"},`
		`{"attribute": "personalmail",`
		`"tag":"personal"}]`
		To use tags in the authentication flow, configure these metadata in the application -
		• urn:iam:app:allowSmsCredsOfTag
		• urn:iam:app:allowEmailCredsOfTag
		• urn:iam:app:allowIvrCredsOfTag
		• excludeCredsNotUsedForXDays
		See Application Metadata Properties.
			`"mvAttributeMap`
			`\": \"mail\", \`
			`{\"attribute\":`
			`\"personal\"}],`
			`\": \"telephone`
			`\"},{\"attribut`
			`\"tag\":\"perso`
			`\":[{\"attribut`
			`\"tag\":\"busi`
			`\"departmentNu`

readTimeoutMillis	No	This is the value of time in milliseconds that will be used on waiting to read data from the LDAP server.	"10000"
connectTimeoutMillis	No	This is the value of time in milliseconds that will be used in making the initial connection to the LDAP server.	"10000"
poolEnabled See the table for more info.	No	When connection pooling is enabled, the LDAP directory server maintains a pool of connections and assigns them as needed. When a connection is closed, the directory server returns the connection to the pool for future use.	"true", "false"
passwordResetURL	No	A link to this URL is presented to the user during password authentication by the Signin UI to allow users to reset their password or to provide instructions on how to proceed.	"http://userPasswor
loginIdRegexEnabled	No	This attribute enables regular expression on the user login ID before looking up in the ID store. Default Value is set as "**False**".	"true", "false"
loginIdRegexFrom	No	This is a regular expression to be applied to the user login ID.	

loginIdRegexTo	No	This is a regular expression used to format the user login ID to match the corresponding attribute value in the identity store.	***Example***: The "**loginIdRegex** parameter has beer example. In this example, wh • loginIdRegexFr • loginIdRegexTo • Input value = +3 The regular expres 495551320 into 049 after the space and

Attribute-Mapping table

The attributes-mapping table here shows the attribute mappings between the physical LDAP attributes (CA Directory) to the logical attributes in IDSP. Refer to the Standard OpenID Connect spec for more details.

Supported User Status behavior table

The following table shows the supported user status (ACTIVE/INACTIVE/UNKNOWN) based on User Disabled and User Lockout behaviors when a specific ID Store is authoritative:

Table 103: Supported User Disabled and User Lockout behaviors with the underlying state of the physical attributes in the ID Store

ID Store Type	Supported Behaviors in ID STore to determine User's ACTIVE and INACTIVE status in IDSP	Required State of LDAPconfig		Required State of ID Store Attributes to determine the user's ACTIVE/INACTIVE status in IDSP			
		statusEncoding (case insensitive)	Attribute Mapping	AuthHub thinks User is Enabled	AuthHub thinks User is Disabled	AuthHub thinks User is Unlocked	AuthHub thinks User is locked
None (Backward compatible post-upgrade)	Enabled/ Disabled (read-only)	Not specified or empty	status=physical-attribute>	physical-attribute> == 0x0200 (512)	physical-attribute> == 0x0202 (514)	N/A	N/A
Active Directory (AD) See AD link for more information	• Enabled/ Disabled read-only • Unlocked/ Locked read/write	"AD"	N/A "status" not used if present	UserAccountControl & 0x0202 != 0x0202	UserAccountControl & 0x0202 == 0x0202	UserAccountControl & 0x0210 ! = 0x0210	UserAccountControl & 0x0210 == 0x0210
DSEE	• Unlocked/ Locked read/write	"DS"	N/A "status" not used if present	N/A	N/A	nsAccountLock == false	nsAccountLock == true
Oracle Universal Directory See the following links for more info: • OUD info 1 • OUD Info 2	• Unlocked/ Locked read/write	"OUD"	N/A "status" not used if present	N/A	N/A	ds-pwp-account-disabled == false	ds-pwp-account-disabled == true

CA Directory See the CA Dir link for more information.	• Unlocked/ Locked read/write	"CADIR"	N/A "status" not used if present	N/A	N/A	dxPwdLocked == false	dxPwdLocked == true
SiteMinder (any ID Store in SiteMinder-only ACTIVE/ INACTIVE behavior) See the SiteMinder link for more information. • DISABLED = 0x01 MAX_LOGINS = 0x02 INACTIVITY = 0x04 • PWEXPIRED = 0x08 • PWMUSTCHANGE = 0x1000000	• Enabled/ Disabled read-only • Unlocked/ Locked read/write	"SITEMINDER"	status=<physical-attribute>	<physical-attribute> & (0x01 | 0x04) == 0	<physical-attribute> & (0x01 | 0x04)!= 0)	<physical-attribute> & 0x02 != 0	<physical-attribute> & 0x02 != 0
INTERNAL	• Enabled/ Disabled (read-only) • Unlocked/ Locked read/write	"INTERNAL"	is_locked=locked, is_disabled=disabled	disabled= false	disabled=true	locked=false	locked - locked=true

LDAP Connection Pool

When using LDAP Identity Store, IDSP supports the use of LDAP Connection Pool (except when validating user credentials using ldap bind) if the pool is enabled in the LDAPconfig resource.

The operating parameters of the LDAP pool are global and must be configured at the time of deploying the IDSP chart. The defaults are maintained in IDSP helm chart's values.yaml.

The LDAP pool parameters that you can pass while deploying IDSP (ssp) helm chart using ***--set <name>="<value>"*** helm parameters, are as follows:

LDAP Pool Config Value	Default	Description
ssp.global.ldap.poolTimeoutMillis	60000	Time to wait until an idle socket gets removed fror the pool provided the current pool size is greater t the preferred pool size time.
ssp.global.ldap.poolInitSize	1	Number of initial connections
ssp.global.ldap.poolPrefSize	0	Number of connections to keep around = number active connections + number of idle connections
ssp.global.ldap.poolMaxSize	0	Maximum number of connections
ssp.global.ldap.poolDebug	"none"	Possible values are "none", "fine", "all". The log shows up in Kibana under the "Third-Party Logs" index.

ssp.global.ldap.poolKeepAlive	"off"	This parameter specifies whether to have the LDA DB connection kept alive. When set to 'on', this parameter enables LDAP connections' to be kept alive. This option prevents the connection from be terminated for being idle (idle termination).

Constructing a user lookup LDAP query

LDAP has strong search capabilities with which you can create search queries both simple and complex to narrow down your users or groups. These are a few optional ldapStart/ldapEnd syntax to locate the user identity:

Table 104: Different ways of constructing the lookup query to locate the user DN

Use Case	User Objectclass	ldapStartQuery	ldapEndQuery	Final LDAP Query to be executed to find the user DN
Locate the user by searching for cn	inetorgperson, person	(cn= ${loginid}	)	(&(cn=${loginid})(| (objectclass=inetorgperson) (objectclass=person)))
Locate user by searching for either cn, email or phone	inetorgperson, person	(|(cn=${loginid})(email= ${loginid})(phone= ${loginid})	)	(&(|(cn=${loginid})(email= ${loginid})(phone=${loginid})) (objectclass=inetorgperson))
Locate the user by searching for uid	person	(uid=${loginid})		(&(uid=${loginid}) (objectclass=person))
Locate directly by constructing DN w/o a search	inetorgperson, person	cn=${loginid}, ou=developers, ou=example, dc=com		Note: No query constructed, DN is looked up directly: cn=${loginid}, ou=developers, ou=example, dc=com
Locate the user by searching for either cn, email, phone, and either idToken, application metadata	inetorgperson, person	(&((|(cn=${loginid}) (email=${loginid}) (phone=${loginid})) (|(ssobuid= ${idToken.ssobuid}) (postalCode= ${appMetadata.postalcode}))))		(&((|(cn=${loginid})(email=${loginid}) (phone=${loginid}))(|(ssobuid= ${idToken.ssobuid})(postalCode= ${appMetadata.ssobuid}))(| (objectclass=inetorgperson) (objectclass=person))))

Optional request parameters

The following LDAP queries can be used optionally to perform the search:

- ***"?ignoreConnectionError=true"*** - Use this query in the POST/PUT API to allow the creation or update of LDAPconfig even when the id store is down.
  Sample Query:
  ```
  https://${SSP_FQDN}/{{tenantName}}/admin/v1/LDAPconfigs/669e9c79-161b-4bec-9d95-d9b76198b7bb?
  ignoreConnectionError=true
  ```
- ***"?templates=true"*** - Use this query in the GET/POST/PUT API to visualize effective search templates.
  Sample Query:
  ```
  https://${SSP_FQDN}/{{tenantName}}/admin/v1/LDAPconfigs/669e9c79-161b-4bec-9d95-d9b76198b7bb?
  templates=true
  ```
  The payload returns additional information about the following:
  – effectiveUserLookupUnderBaseDN
  – effectiveGroupLookupUnderGroupBaseDN
  – effectiveGroupUserMembershipLookupUnderGroupBaseDN

Sample Payload:


```
{
    "configName": "TestConfig",
    "configDescription": "CA DIR k8s with default data",
    "sslEnabled": "false",
    "certificate": null,
    "certificateAlias": null,
    "ldapQueryStart": "(|(cn=${loginid})(mail=${loginid})(telephoneNumber=${loginid}",
    "ldapQueryEnd": ")",
    "userObjectClasses": "inetorgperson",
    "groupObjectClasses": "groupOfNames",
    "hostName": "cadir-ssp-symantec-dir.idstore.svc.cluster.local",
    "portNumber": "1389",
    "baseDN": "o=democorp,c=US",
    "groupBaseDN": "o=democorp,c=us",
    "bindDN": "cn=DominicMAJOR,ou=Administration,ou=Corporate,o=democorp,c=US",
    "bindPassword": "***",
    "attributeMapping":
 "user_loginid=cn,name=cn,family_name=sn,given_name=givenName,middle_name=givenName,nickname=displayName,preferred_u
    "groupUserMappingAttribute": "member",
    "useMemberOf": "false",
    "passwordResetURL": "http://userPasswordResetURL",
    "createdDateTime": "2021-02-22 07:42 AM UTC",
    "updatedDateTime": "2021-02-22 07:49 AM UTC",
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "86414094-beff-4058-8837-0fc7278aab0a",
        "principalName": "defaulttenantclient"
    },
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "86414094-beff-4058-8837-0fc7278aab0a",
        "principalName": "defaulttenantclient"
    },
    "ldapConfigId": "669e9c79-161b-4bec-9d95-d9b76198b7bb",
    "_effectiveUserLookupUnderBaseDN": "(&(|(cn=${loginid})(mail=${loginid})(telephoneNumber=${loginid})
(objectClass=inetorgperson))",
    "_effectiveGroupLookupUnderGroupBaseDN": "(objectClass=groupOfNames)",
    "_effectiveGroupUserMembershipLookupUnderGroupBaseDN": "(&(objectClass=groupOfNames)(member=
${userdn}))"
}
```


Reference Links to:

- ID Store Configuration API Error Codes

Managing SCIM Configuration

Learn how to configure SCIM in Symantec Identity Security Platform.

This topic includes the following sections:

- Overview
- SCIM Configuration APIs
- List of attributes and their descriptions used in SCIM configuration
- securityMethod Types and its Associated Mandatory Properties
- Constructing a user lookup SCIM query

Overview

The System for Cross-domain Identity Management (SCIM) specification is designed to make managing user identities in cloud-based applications and services easier. The specification builds upon experience with existing schemas and deployments, placing specific emphasis on simplicity of development and integration while applying existing authentication, authorization, and privacy models. Its intent is to reduce the cost and complexity of user management operations by providing a common user schema and extension model, as well as binding documents to provide patterns for exchanging this schema using standard protocols. In essence: make it fast, cheap, and easy to move users in to, out of, and around the cloud.

While there are existing standards for describing and exchanging user information, many of these standards can be difficult to implement and/or use. For example, their wire protocols do not easily traverse firewalls and/or are not easily layered onto existing web protocols. As a result, many cloud providers implement non-standardized protocols for managing users within their services. This increases both the cost and complexity associated with organizations adopting products and services from multiple cloud providers, as they must perform redundant integration development. Similarly, cloud service providers seeking to interoperate with multiple application marketplaces or cloud identity providers would require pairwise integration. SCIM seeks to simplify this problem through an easily implemented specification suite that provides a common user schema and extension model, as well as a SCIM protocol document that defines exchanging this schema via an HTTP-based protocol.

NOTE
The solution supports SCIM 2.0 and 1.1 protocols.

SCIM Configuration APIs

The following table displays the list of APIs that can be used by an administrator to configure a SCIM compliant directory in the solution:

Operation Description	Operation Type	Endpoint URLs
Create SCIM Configuration	POST	https://${SSP_FQDN}/{{tenantName}}/admin/v1/SCIMconfigs
Retrieve all SCIM configurations	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/SCIMconfigs
Retrieve SCIM configuration by specifying the SCIM config Id	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/SCIMconfigs/{{scimConfigId}}
Update SCIM configuration by specifying the SCIM config Id	PUT	https://${SSP_FQDN}/{{tenantName}}/admin/v1/SCIMconfigs/{{scimConfigId}}
Delete a SCIM configuration by specifying the SCIM config Id	DELETE	https://${SSP_FQDN}/{{tenantName}}/admin/v1/SCIMconfigs/{{scimConfigId}}

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/SCIMconfigs`

should look similar to:
`https://example.com/default/admin/v1/SCIMconfigs`

- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Click here to expand the Sample SCIM config request payload

A sample POST SCIM config request payload looks like the following:

```
POST https://{{sspHost}}/{{apiPathTenant}}/admin/v1/SCIMconfigs
{
    "configName": "IDCS SCIM",
    "configDescription": "",
    "clientId": "70d8bd85645c428ea4704a8e1823c5e3",
    "clientSecretAlias": "idcsclientsecret",  --> points to base64("8921bd74-16b0-4b94-b1e3-22054fe5b0f7")
    "oauthScopes": "urn:opc:idm:__myscopes__",
    "securityMethod": "oauth",
    "oauthTokenUrl": "https://idcs-e976c92e06d64eeab87e14cb1ef8d206.identity.oraclecloud.com/oauth2/v1/token",
    "oauthResourceInAzRequest": "none",
    "oauthAssertionKidType": "x5t",
    "oauthAssertionAud": "",
    "groupAttributeMapping": "name=displayName, members=members.display",
    "groupNameAttributeMappingName": "name",
    "scimURL": "https://idcs-e976c92e06d64eeab87e14cb1ef8d206.identity.oraclecloud.com/admin/v1",
    "certificateAlias": "",
    "ignoreSslValidation": true,
    "isAuthoritativeForIDLCM": true,
    "isAuthoritativeForUserCredsLCM": false,
    "statusEncoding": "SCIM",
    "lookupQuery": "userName Eq \"${loginid}\"",
    "useUserLoginIdAsScimId": false,
    "useGroupNameAsScimId": false,
    "endpointSuffixOverride": {},
    "userLoginIdAttributeMappingName":"user_loginid",
    "attributeMapping": "name=userName, family_name=name.familyName, given_name=name.givenName,
 phone_number=phoneNumbers[primary], email=emails[primary], user_universalid=userName,
 memberOf=groups.display",
    "readTimeoutMillis": "10000",
    "connectTimeoutMillis": "10000",
    "oauthTokenCertificateAlias": "",
    "oauthTokenIgnoreSslValidation": true,
    "oauthTokenSslMode": "VERIFY_CA",
    "oauthTokenReadTimeoutMillis": "10000",
    "oauthTokenConnectTimeoutMillis": "10000"
}

IDCS Client App
pm.environment.set("app_name", "MyNewTrustApp");
pm.environment.set("CLIENT_ID", "70d8bd85645c428ea4704a8e1823c5e3");
pm.environment.set("CLIENT_SECRET", "8921bd74-16b0-4b94-b1e3-22054fe5b0f7")
```

For sample request and response payloads of all these URLs, refer to SCIM Configuration API- Swagger documentation

List of attributes and their descriptions used in SCIM configuration

Table 105: SCIM configuration attributes and their descriptions

Attribute Name	Mandatory Fields	Description	Example values
configName	Yes	Name of the SCIM configuration to be created	"TestSCIMConfig"
configDescription	No	Short description of the SCIM configuration so that it is easily identifiable	"Test SCIM configuration"
clientId	Yes	• For securityMethod="basic" - the clientId corresponds to the Username. • For securityMethod="OAuth" or "oauthassertion"- the clientId corresponds to the Oauth2 clientId obtained when IDSP is registered as an oauth2 application with the SCIM service.	"abc1234"
clientSecretAlias	No	This is the alias secret created using the admin /Secrets API. This alias is used to connect to the SCIM server. **Note:** In the case of "securityMethod" as "oauthassertion", the "clientSecretAlias" property is required when "oauthAssertionKidType" is "secret" or "skid". Refer to the "oauthAssertionKidType" property for details on when "clientSecretAlias" is required.	"MyScimConnectionClientSecret"
oauthScopes	Yes	Scope of the token that will be generated.	"<as-required-by-scim-service>"
securityMethod	Yes	Type of security method to be used to connect to the SCIM server. Possible values are basic, OAuth, oauthassertion. For more information about the securityMethods and its associated mandatory properties that are required while creating the SCIM configuration, see Security Method Required Properties.	"OAuth"
oauthTokenUrl	Yes	URL for the Token endpoint where Access token request will be sent.	"https://scimservice.com/oauth"

oauthResourceInAzRequest	No	RFC8707 resource indicator for OAuth STS request. Possible values are "none", "base", and "all".	"none"
oauthAssertionKidType	No	Used when the "securityMethod" is set to "oauthassertion". It indicates how to establish the kid of the signed assertion. Possible Values: skid, x5t, secret If "oauthAssertionKidType" is "secret", then the "kid" value of the assertion is populated from the "kid" value of the rsakey secret identified by “clientSecretAlias". If "oauthAssertionKidType" is "skid", then the "kid" value of the assertion is populated from the "skid" value of the certificate stored in the rsakey secret identified by clientSecretAlias. If clientSecretAlias is not provided, the signing key of the tenant is used. If "oauthAssertionKidType" is "x5t", then the "kid" value of the assertion is populated from the x5t value of the certificate stored in the rsakey secret identified by clientSecretAlias. If clientSecretAlias is not provided, the signing key of the tenant is used.	"x5t"
oauthAssertionAud	No	Only used when using securityMethod="oauthassertion". It indicates the audience to inject into the assertion. If not specified, the value of the oauthTokenUrl will be injected.	" "
scimURL	Yes	URL to connect to the SCIM server. Unless overwritten by "endpointSuffixOverride", it is expected that the "/Users" and "/Groups" will be appended to the SCIM service URL. It is mandatory to prefix the SCIM URL with the HTTP/HTTPS protocol.	"https://scimservice.com"
testConnectionURL	No	Used to validate SCIM service connection during create/update of the configuration. If not specified, no validation is done when creating/saving configuration	"https://scimservice.com/testconnection"
ignoreSSLValidation	Yes	This attribute indicates if the SSL validation should be ignored or not. The default is set to "true".	"true"

sslMode	No	This indicates whether SSL validation should be performed or ignored. If the "sslMode" is enabled, it should validate the server's identity. Possible Values are - "VERIFY_CA", "VERIFY_IDENTITY". If the "sslMode" is not indicated, it defaults to "VERIFY_CA".	"VERIFY_CA"
certificateAlias	No	Name of the certificate alias pointing to IDSP's "/cert" entry. The certificate should be added using the "/Certs" API. This attribute should be provided if the "ignoreSSLValidation" attribute is set to "false".	"MyScimCaCert"
lookupQuery	No	This is the SCIM filter to locate the user entry in SCIM, the syntax for the SCIM filter can be found in RFC7644. The names of the attributes used in this query are physical attribute names. This value can contain dynamic environment variables which will be substituted with their corresponding variable value, when the query is used. See Dynamic Environment Variable Substitution.	"userName Eq $user-name"
useUserLoginIdAsScimId	Yes	This attribute determines if "user_loginId" can be used as SCIM Id.	"true", "false" depending on how the Scim service is setup.
useGroupNameAsScimId	Yes	This property determines if the group name can be used as the SCIM Reference ID.	true/false depending on how the Scim service is setup.

endpointSuffixOverride	No	URL map in key-value pair representing custom SCIM endpoints.	Only used in case a SCIM service does not support standard SCIM endpoints (/Users, /Groups) relative to the base URL identified by "scimURL"
			`{`
			`"GetUsers": "https://scimservice.com/getUsers",`
			`"PatchUsers":"https://scimservice.com/patchUsers"`
			`}`

requestCtxBase64	No	This is a 64-bit encoded string of the Context map in key-value pair, representing instructions to execute when invoking **SCIM** endpoints - such as sending specific headers.
		In this attribute, the "***value***" field can contain dynamic environment variables which will be substituted with their corresponding variable value when this attribute is used.
		See Dynamic Environment Variable Substitution.
			<pre>"requestCtxBase64": [
			{"type" : "http-header",
			"key" : "userdn",
			"value" : "${IdToken.user_dn}"
			},
			{"type" : "http-header",
			"key" : "myheader",
			"value" : "myvalue"
			}
			]</pre>

statusEncoding	No	This attribute specifies the type of encoding used to represent the disabled status managed by the user lockout feature. The value can be "scim" to signal that user's disabled status should be determined using SCIM's standard user attribute "active", or it can be "siteminder" to manage the user's disabled state using a bit-mask notation where a custom ID store attribute represented by the 'status' logical attribute holds the physical bitmask representing various user states. For information about Sm_Api_Disabled_xxx values, see Exported Enumerations. Without the specific value, user lockout operations fail and the user's status will be assumed to be "unknown". This parameter is optional and when used should be set to the actual type of the Identity store. • See Supported User Status behavior table. • See User Locker	"scim" or "siteminder" The only value for SCIM config is "scim" (implying the use of SCIM's active attribute) or "siteminder" (implying the use of any xxx attribute indicated by status=xxx mapping)
idStoreMappingStrategy	No	The value of this property contains the mapping strategy for mapping the disambiguation Id Store to the authentication Id Store. See ID Store Mapping in Password Authentication. This attribute value can contain dynamic environment variables which will be substituted with their corresponding variable value, when this attribute is used. See Dynamic Environment Variable Substitution.	<pre>"attribute_mapping" : "user_loginid=cn, name=cn, family_name=sn, given_name=givenName, middle_name=givenName, nickname=displayName, preferred_username=displayName, profile=title,email=mail, phone_number=telephoneNumber, user_universalid=cn, status=employeeType, memberof=memberof"</pre>

isAuthoritativeForIDLCM	No	This attribute specifies if the SCIM ID Store that is being configured, is also the authoritative source for user locking. This is a String boolean with a value of true/false. If this value is set to "false", then IDSP becomes the authoritative locking source and the SCIM that has been configured will not be the locking source. The user locking is done when the user strike count reaches the maximum allowed value. If this value is set to "true", the user lock will be done in the user store.	"true", "false"
isAuthoritativeForUserCredsLCM	No	This attribute specifies if the SCIM ID Store that is being configured, is also the authoritative source for the user's credentials. If this value is true, the user's OTP creds established during inline registration would be synced back to the id store.	"true", "false"
isAuthoritativeForDetectingLockedState	No	This boolean attribute specifies if the SCIM ID Store that is being configured, is also the authoritative source for detecting the locked state of the user or not. If this value is true, the user's locked state in the SCIM IDStore will be used, otherwise the user's internal locked state is used.	"true", "false"
readTimeoutMillis	No	This is the value of time in milliseconds that will be used on waiting to read data.	"10000"
connectTimeoutMillis	No	This is the value of time in milliseconds that will be used in making the initial connection.	"10000"
userLoginIdAttributeMappingName	No	Name of the logical attribute representing the unique userid in the id store	"user_loginid" This aligns with the "name=userName" portion of the config for attributeMapping attribute. If omitted, it defaults to "name"
groupNameAttributeMappingName	No	This attribute contains the name of the logical mapping attribute representing the group name. This value should have a mapping from the logical to the physical attribute.	"name" If omitted, it defaults to "name"

groupAttributeMapping	No	This is a comma separated list of key-value pairs representing the mapping between logical attributes and physical SCIM attributes for a group	"name=displayName, members=members.display"
attributeMapping	No	This is a list of mappings that map logical attribute names to their physical SCIM attribute.	"**user_loginid**=userName, **family_name**=name.familyName, **given_name**=name.givenName, **phone_number**=phoneNumbers[primary], **email**=emails[primary], **user_universalid**=userName, **memberOf**=groups.display"
passwordResetURL	No	A link to this URL is presented to the user during password authentication by the Signin UI to allow users to reset their password or to provide instructions on how to proceed.	"http://userPasswordResetURL" to be replaced by customer-specific value.
oauthTokenIgnoreSslValidation	Yes	This attribute indicates if the SSL validation should be enabled. The default value is set as "true".	"true"
oauthTokenSslMode	No	This indicates whether SSL validation is enabled. If enabled, it should validate the server's identity. Valid values are "VERIFY_CA", "VERIFY_IDENTITY". If not indicated, defaults to "VERIFY_CA".	"VERIFY_CA"
oauthTokenCertificateAlias	No	This is the name of the certificate alias pointing to IDSP's "/Certs" entry. This attribute should be provided if "ignoreSSLValidation" is set to false. The certificate or certificate chain pointed to by the alias should be added using the "/Certs" API.	"MyScimOauthCaCert"
oauthTokenRequestCtxBase64	No	This is a base64 encoded string of the Context map in key-value pair, indicating hints to SSP to execute when invoking **STS** endpoint - such as sending specific headers. In this attribute, the "***value***" field can contain dynamic environment variables which will be substituted with their corresponding variable value when this attribute is used. See Dynamic Environment Variable Substitution.	`BASE64 <[ {"type" : "http-header", "key" :"myheader", "value": "myvalue" }] >`
oauthTokenReadTimeoutMillis	No	This is the value of time in milliseconds that will be used on waiting to read data.	"10000"

oauthTokenConnectTimeoutMillis	No	This is the value of time in milliseconds that will be used in making the initial connection.	"10000"
skewTimeSecs	No	When using securityMethod="oauthassertion", this property is helpful in cases when the target STS has no capability to adjust time comparisons for skew time. In that case, an integer "skewTimeSecs" (could be positive or negative) is used to offset the "exp" and "nbf" claims.	"0"
loginIdRegexEnabled	No	This attribute enables regular expression on the login ID before looking up in the ID store.	"true", "false"
loginIdRegexFrom	No	This is a regular expression to be applied to the user login ID.	
loginIdRegexTo	No	This is a regular expression used to format the user login ID to match the corresponding attribute value in the identity store.	***Example***: The "**loginIdRegexFrom**" and "**loginIdRegexTo**" parameter has been explained through the following example. In this example, when the following values are considered, • loginIdRegexFrom = ^(.*)\s(.*)$ • loginIdRegexTo = 0{2} • Input value = +32 495551320 The regular expression transforms the input value +32 495551320 into 0495551320 (result) by capturing the part after the space and adding 0 in front of it.
passwordResetURL		A link to this URL is presented to the user during password authentication by the Signin UI to allow users to reset their password or to provide instructions on how to proceed.	
Account Unlock URL			

securityMethod Types and its Associated Mandatory Properties

The following table lists the securityMethods and its associated mandatory properties that are required while creating the SCIM configuration:

Table 106: securityMethod Types and its Associated Mandatory Properties

securityMethod Type	Required Properties for the securityMethod Type	Notes
OAuth	clientId (clientId), clientSecretAlias (secret), oauthTokenUrl, oauthScopes	The client secret must be added using the Secrets API(/admin/v1/Secrets/) and the name of the created secret must be provided in the clientSecretAlias property. The header key-value pair that is associated with the OAuth TokenServer can be passed through the oauthTokenRequestCtxBase64 property.
oauthassertion	clientId (clientId), clientSecretAlias (key), oauthTokenUrl, oauthScopes, oauthAssertionAud, oauthAssertionKidType(secret)	In the case of oauthAssertionKidType as secret, the signature details must be added using the keys API(/admin/v1/Keys/) and the name of the created key must be provided in the clientSecretAlias parameter. The clientSecretAlias property is required when oauthAssertionKidType is secret or skid. The header key-value pair that is associated with the OAuth TokenServer can be passed through the oauthTokenRequestCtxBase64 property.

Constructing a user lookup SCIM query

SCIM lookup query can be used to search users in the SCIM ID Store using user login identity, identity token (ZFP), and application metadata. The supported substitution strings in the SCIM lookup query are as follows:

- ${loginid}: Represents the user login id.
- ${idToken}: Represents an Identity Token, which is available only when Identity Token Hint is present in the flow.
- ${appMetadata}: Represents attributes from the application metadata.

Examples of the SCIM lookup query:

```
"lookupQuery": "userName eq \"${loginid}\""
"lookupQuery": "(userName eq \"${loginid}\") or (emails[type eq \"work\" and value eq \"${loginid}\"])"
"lookupQuery": "userType eq \"Employee\" and emails[type eq \"work\" and value eq \"${loginid}\"]"
"lookupQuery": "(userName eq \"${loginid}\") and (ssobuid eq \"${idToken.ssobuid}\" or reqion eq
 \"${appMetadata.region}\")"
```

Reference Links to:

- ID Store Configuration API Error Codes

Managing Applications

Manage applications in Symantec Identity Security Platform using the Application REST API to perform CRUD operations and integrate services via standard OpenID Connect (OIDC) and SAML protocols.

The Application API provides a centralized interface for adding, updating, and overseeing the lifecycle of applications within your environment. By leveraging industry-standard OIDC and SAML protocols, IDSP ensures interoperability with a wide range of third-party platforms. This allows you to configure authentication flows, manage client credentials, and define application-specific security policies to facilitate seamless integration across your infrastructure.

This topic includes the following sections:

- Application API and Supported Operations
- Creating an Application
  - Creating an OIDC Application - Request and Response sample
  - Creating a SAML Application - Request and Response sample
- Description of Application Attributes (OIDC and SAML attributes)
- Managing the Client ID and Secret
- Updating CORS Setting to Reflect Your Application's FQDN
- Deleting an Application Referenced by Other IDSP Objects

Application API and Supported Operations

The following table displays the list of Application APIs that you can use to manage applications that have been added to the solution.

Operation Description	Operation Type	Endpoint URLs
Create Application	POST	https://${SSP_FQDN}/{{tenantName}}/admin/v1/App
Search for Applications	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Apps
Search for Application by ID	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Apps/{{appId}}
Update Application by ID	PUT	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Apps/{{appId}}
Delete Application by ID	DELETE	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Apps/{{appId}} To forcefully delete an application group, including those currently in use, s Resource Deletion in IDSP.

To view the request and response payload for these URLs, refer to Application API Swagger documentation.

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/Apps`
  should look similar to:
  `https://example.com/default/admin/v1/Apps`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Creating an Application in IDSP

Use the following Application API to create a new application in IDSP:

```
POST https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/Apps
```

Creating an OIDC Application - Request and Response Sample

The following example shows the request payload of the ".../admin/v1/Apps" API using the OIDC Protocol. In addition to the obligatory attributes displayed in the sample, there are several optional attributes that can be included in the payload by performing a PATCH call.

Request Payload of a Confidential Client Type

```
{
"status": "active",
```

```
"name": "Apple Application",
"description": "Default Application for Security",
"clientType": "CONFIDENTIAL",
"allowedOpenIDScopes": [
"openid",
"email",
"profile",
"urn:iam:sso:idprop"
],
"redirectURIs": [
"https://www.example.com"
],
"allowedGrantTypes": [
"implicit",
"authorization_code",
"urn:ietf:params:oauth:grant-type:jwt-bearer",
"client_credentials",
"urn:ietf:params:oauth:grant-type:device_code"
],
"claims": [
{
"name": "_title",
"target": "at,it",
"value": "${idProfile.title}"
},
{
"name": "_idp_name",
"target": "at,it",
"value": "${idToken.idp_name}"
},
{
"name": "_env",
"target": "at,it",
"value": "${appMetadata}.env}"
},
{
"name": "_acct_num",
"target": "at,it",
"value": "12345"
},
{
"name": "_transactionIdentifier",
"target": "at,it",
"value": "${clientContext.transactionIdentifier}"
}
],
"allowedOperations": [
"introspect",
"onbehalfof",
"offlineit"
],
"skewTimeSecs": 0,
"skipIssuerAudienceForIT": false,
```

```
"userInfoEndpointResponseFormat": "PLAIN_JSON",
"zeroFootPrint": false,
"skipEmailForIT": false,
"supportedJoseHeaderParams": "x5t,x5t#s256",
"appIcon": "..."
}
```


Response of the Confidential Client Type

The response displays both the mandatory and optional parameters for OIDC, and SAML protocol flows.


```
{
"status": "active",
"name": "Apple Application",
"description": "Default Application for Security",
"clientType": "CONFIDENTIAL",
"allowedOpenIDScopes": [
"openid",
"profile",
"urn:iam:sso:idprop",
"email"
],
"redirectURIs": [
"https://www.example.com"
],
"allowedGrantTypes": [
"implicit",
"client_credentials",
"urn:ietf:params:oauth:grant-type:device_code",
"authorization_code",
"urn:ietf:params:oauth:grant-type:jwt-bearer"
],
"flowURL": null,
"deviceCodeFlowURL": null,
"userInfoEndpointResponseFormat": "PLAIN_JSON",
"skipIssuerAudienceForIT": false,
"skipEmailForIT": false,
"idpDiscoveryEnabled": null,
"zeroFootPrint": false,
"softMFAEnabled": false,
"delegatedAuthentication": false,
"autoPostToFlowURL": false,
"userTokenSubAttributeMappingName": null,
"supportedJoseHeaderParams": "x5t,x5t#s256",
"claims":[
{
"name": "_title",
"target": "at,it",
"value": "${idProfile.title}"
},
{
"name": "_idp_name",
"target": "at,it",
```

```
"value": "${idToken.idp_name}"
},
{
"name": "_env",
"target": "at,it",
"value": "${appMetadata}.env}"
},
{
"name": "_acct_num",
"target": "at,it",
"value": "12345"
},
{
"name": "_transactionIdentifier",
"target": "at,it",
"value": "${clientContext.transactionIdentifier}"
}
],
"customClaimProviderToUse": null,
"allowedOperations": [
"onbehalfof",
"introspect",
"offlineit"
],
"secondaryAudiences": null,
"assertionVerificationCertAlias": null,
"appIcon": "...",
"secret": "7edc39c7-fa15-47a1-8473-8204e4772790",
"clientId": "5ee5687a-eb66-4025-8f6c-a2eae14256ea",
"itEncryptionTarget": "NONE",
"itEncryptionCertAlias": null,
"userInfoEncryptionCertAlias": null,
"samlEntityId": null,
"samlAcsUrl": null,
"samlEnableSingleLogout": false,
"samlNameIdFormat": null,
"samlVerifyRequestSignature": false,
"samlVerifyCertAlias": null,
"samlEncryptSamlResponse": false,
"samlEncryptCertAlias": null,
"samlSignResponse": false,
"samlSignAssertion": false,
"samlSignatureWithKeyInfo": false,
"samlIdpInitiatedRelaystateMapping": {},
"samlAssertionClaims": [],
"samlUserTokenSubAttributeMappingName": null,
"skewTimeSecs": 0,
"passwordAuthoritativeSource": "remote",
"mitmProtectionLevel": null,
"idStoreToUse": null,
"homeUrl": null,
"launcherUrl": null,
"errorUrl": null,
```

```
"logoutUrl": null,
"postLogoutUrl": null,
"appId": "3191a834-faeb-42ed-a3cf-b10953ca54ee",
"createdBy": {
"principalType": "CLIENT",
"principalId": "c2d28f2c-4f4a-4557-9ba7-0248052e9a60",
"principalName": "DefaultTenantClient"
},
"createdDateTime": "2024-04-11 04:49 AM UTC",
"updatedDateTime": "2024-04-11 04:49 AM UTC",
"isSamlApp": false,
"isOidcApp": true,
"isLauncherApp": false,
"requireAccessControl": null
}
```


Creating a SAML Application - Request and Response Sample

The following example shows the request payload of the ".../admin/v1/Apps" API using the SAML Protocol. In addition to the obligatory attributes demonstrated in the sample, there are several optional attributes that can be included in the payload by performing a PATCH operation.

SAML App Request Payload


```
{
"status": "active",
"name": "Test App",
"description": "Default Application for Security",
"clientType": "CONFIDENTIAL",
"allowedOpenIDScopes": [
"openid",
"email",
"profile"
],
"redirectURIs": [
"https://www.example.com"
],
"allowedGrantTypes": [
"implicit",
"authorization_code",
"urn:ietf:params:oauth:grant-type:jwt-bearer",
"client_credentials",
"urn:ietf:params:oauth:grant-type:device_code"
],
"allowedOperations": [
"introspect",
"onbehalfof"
],
"skewTimeSecs": 0,
"skipIssuerAudienceForIT": false,
"userInfoEndpointResponseFormat": "PLAIN_JSON",
"zeroFootPrint": false,
"skipEmailForIT": false,
"supportedJoseHeaderParams": "x5t,x5t#s256",
"appIcon": "...",
```

```
"samlAssertionClaims": [
{
"name": "ctry",
"value": "${idProfile.ctry}"
},
{
"name": "fname",
"value": "${idProfile.fname}"
},
{
"name": "lname",
"value": "${idProfile.lname}"
}
],
"samlEntityId": "https://example.com",
"samlAcsUrl": "https://www.example.com",
"samlEnableSingleLogout": false,
"samlVerifyRequestSignature": false,
"samlEncryptSamlResponse": false,
"samlNameIdFormat": "unspecified",
"samlIdpInitiatedRelaystateMapping": {
"PORTAL": "test/test"
}
}
```


SAML App Response Payload

The response displays both the mandatory and optional parameters for OIDC, and SAML protocol flows.


```
{
 "appId": "1b564f7e-65d5-4f31-bb9d-abc135940e4d",
 "status": "active",
 "name": "SAML for App Launcher testing",
 "resScope": null,
 "description": "Sample SAML Application",
 "logoId": null,
 "idpDiscoveryEnabled": null,
 "skewTimeSecs": 0,
 "passwordAuthoritativeSource": "remote",
 "mitmProtectionLevel": null,
 "zeroFootPrint": false,
 "idStoreToUse": null,
 "softMFAEnabled": false,
 "delegatedAuthentication": false,
 "flowURL": null,
 "deviceCodeFlowURL": null,
 "autoPostToFlowURL": false,
 "clientType": "CONFIDENTIAL",
 "clientId": null,
 "regenerateClientId": false,
 "regenerateSecret": false,
 "assertionVerificationCertAlias": null,
 "allowedOpenIDScopes": [],
 "allowedGrantTypes": [
```

```
"implicit",
"client_credentials",
"urn:ietf:params:oauth:grant-type:device_code",
"authorization_code",
"urn:ietf:params:oauth:grant-type:jwt-bearer"
],
  "allowedOperations": [
"onbehalfof",
"introspect"
],
  "supportedJoseHeaderParams": "x5t,x5t#s256",
  "secondaryAudiences": null,
  "claims": [],
  "userInfoEndpointResponseFormat": null,
  "skipIssuerAudienceForIT": false,
  "skipEmailForIT": false,
  "userTokenSubAttributeMappingName": PLAIN_JSON,
  "customClaimProviderToUse": null,
  "itEncryptionTarget": null,
  "itEncryptionCertAlias": null,
  "userInfoEncryptionCertAlias": null,
  "samlEntityId": "https://example.com",
  "samlEntityIdStripe": "1b564f7e-65d5-4f31-bb9d-abc135940e4d",
  "samlAcsUrl": "https://example.com",
  "samlEnableSingleLogout": false,
  "samlNameIdFormat": "unspecified",
  "samlVerifyRequestSignature": false,
  "samlVerifyCertAlias": null,
  "samlEncryptSamlResponse": false,
  "samlEncryptCertAlias": null,
  "samlEncryptAlgo": "http://www.w3.org/2001/04/xmlenc#aes256-cbc",
  "samlKeyTransportAlgo": "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p",
  "samlSignResponseKeyAlias": null,
  "samlSignAssertionKeyAlias": null,
  "samlSignResponse": false,
  "samlSignAssertion": false,
  "samlSignResponseAlgo": null,
  "samlSignResponseCanonicalizationAlgo": null,
  "samlSignAssertionAlgo": null,
  "samlSignAssertionCanonicalizationAlgo": null,
  "samlSignatureWithKeyInfo": false,
  "samlIdpInitiatedRelaystateMapping": [
    {
      "key": "test",
      "value": "test"
    }
  ],
  "samlAssertionClaims": [
 {
   "name": "ctry",
   "value": "${idProfile.ctry}"
 },
 {
```

```
    "name": "fname",
    "value": "${idProfile.fname}"
  },
  {
    "name": "lname",
    "value": "${idProfile.lname}"
  }
  ]
   "samlUserTokenSubAttributeMappingName": null,
   "samlEffectiveIdpInitiatedUrlPreferred": "https://example.com/default/saml/v1/idp/login/app/
MWI1NjRmN2UtNjVkNS00ZjMxLWJiOWQtYWJjMTM1OTQwZTRk?isIdpInit=true",
   "samlEffectiveIdpInitiatedUrlWithEntityId": "https://example.com/default/saml/v1/idp/login?entityId=https://
example.com",
   "samlEffectiveIdpInitiatedUrlWithAppId": "https://example.com/default/saml/v1/idp/login?
appId=1b564f7e-65d5-4f31-bb9d-abc135940e4d",
   "samlEffectiveSpAcsUrlWithEntityId": "https://example.com/default/saml/v1/sp/acs?sp=https://example.com",
   "samlEffectiveSpAcsUrlWithAppId": "https://example.com/default/saml/v1/sp/acs?sp=1b564f7e-65d5-4f31-bb9d-
abc135940e4d",
   "samlEffectiveSpAcsUrlWithAppName": "https://example.com/default/saml/v1/sp/acs?sp=SAML for App Launcher
 testing",
   "homeUrl": null,
   "launcherUrl": null,
   "errorUrl": null,
   "logoutUrl": null,
   "postLogoutUrl": null,
   "metadata": [
     {
       "name": "myOrg",
       "value": "org123"
     }
   ],
   "label": null,
   "sortOrder": null,
   "hidden": false,
   "primaryAudience": null,
   "resourceServerScopes": [],
   "consentUsagePolicyUrl": null,
   "createdBy": {
     "principalType": "CLIENT",
     "principalId": "63771ac5-b70b-4e4b-8c17-1ec8f303a260",
     "principalName": "DefaultTenantClient"
   },
   "createdDateTime": "2026-01-08 05:38 AM UTC",
   "updatedDateTime": "2026-01-08 05:38 AM UTC",
   "adminscope.criteria": [],
   "adminscope.managedby": [],
   "isOidcApp": false,
   "isSamlApp": true,
   "isLauncherApp": false,
   "requireAccessControl": null,
   "isResourceServerApp": false
}
```

NOTE

SAML applications support the ${saml_idp_init} placeholder for the launcherUrl field, which automatically populates the URL with the "app.samlEffectiveIdpInitiatedUrlPreferred" value during launcherUrl invocation, eliminating the need for manual URL entry.

The "app.samlEffectiveIdpInitiatedUrlPreferred" attribute value can be found in the response of the SAML application.

To enable launcher app, see Launcher App.

Description of Application Attributes (OIDC and SAML attributes)

The following table provides descriptions of both OIDC and SAML related (obligatory and optional) attributes that are used in the ".../admin/v1/Apps" API:

Attribute Name	Description
status	Validates the status of the application. Supported values- active (Default) &
name	Specify a name to the application.
description	Short description of the application so that it is easily identifiable.
clientType	A string representing the client type. OAuth flows work based on client type Supported values- • **Confidential**- – Clients that can be trusted to securely store their credentials such as App servers. – Clients that are trusted to issue user assertions such as an ID Token submitting additional information about the user in certain API calls (API calls). – Confidential clients can submit their credentials using Basic Auth Au • **Trusted**- – Clients that are trusted to issue user assertions such as an ID Token submitting additional information about the user in certain API calls (API calls). • **Public**- – Clients that cannot securely store their credentials. For example, Jav mobile clients. – Public clients can submit their credentials using Basic Auth Authoriz To know more about Client Types, see Client Types.
allowedOpenIDScopes	Allowed scopes are ***openid***, ***profile***, ***phone***, ***email***, ***address***, ***offline_acce*** Custom Scopes such as: • urn:iam:useTenantKeyForITencryption • urn:iam:useUserInfoKeyForITencryption • urn:iam:at:appaud • urn:iam:it:groups • urn:iam:at:groups • urn:iam:sso:idprop • urn:iam:lcm:resetpassword
redirectURIs	Lists out the allowed callback URLs to which authorization code or tokens a

allowedGrantTypes	User restricts the client to allow only specified flows. Supported grant types are - • authorization_code • device_code • refresh_token • jwt_bearer • client_credentials • sts_confidential • implicit To know more about these grant types, see Grant Types . The Refresh Tok '**device_code**' grant_types.
allowedOperations	• ***introspect***: The Introspect Endpoint is used to validate the received Ac the token. OAuth introspect endpoint only works when 'introspect' is co returns an error. • ***onbehalfof***: When 'onbehalfof' is configured, then the client fetches the are only privileges assigned to the user, and not to the combination of i • ***offlineit***: When 'offlineit' is configured, the token endpoint returns the ID the openid scope is requested. • **itgroups**: When the client is configured with itgroups, the application is • **atgroups**: When the client is configured with atgroups, the application i
flowURL	(Optional) This attribute specifies the Symantec Identity Security Platform U Security Platform API for login controls. If this attribute value is empty, no v For SignIn UI and Custom SignIn UI related information, see SignIn User In This URL can use environment variables of the namespaces, appMetadata Environment Variable Substitution.
deviceCodeFlowURL	(Optional) This is the customized UI URL required to enter the user code in is not configured, then the default IDSP***verification_uri https://$SSP_FQ***
skipIssuerAudienceForIT	By default 'aud' claim in the IT will have two values: 1. client guid and 2. issuer url. Enabling the flag 'skipIssuerAudienceForIT' adds only clientid in the 'aud' c Supported values are - True and False (Default - False)
skipEmailForIT	By default the ID token issued by the Symantec Identity Security Platform c flag 'skipEmailForIT' does not add email claim in the ID token.
userInfoEndpointResponseFormat	The userInfoEndpointResponseFormat supported are PLAIN_JSON, JWT,
idpDiscoveryEnabled	This attribute specifies whether the Identity Discovery policy should underg policy, this specific configuration takes precedence over the same setting c with three possible values: ***true***, **false** or ***tenant***. • If configured as "***true***" then IDP Discovery policy evaluation will take pla • If configured as "***false***" then IDP Discovery policy evaluation will not tak • If configured as "***tenant***" then IDP Discovery policy evaluation will take either true or false. See IDP Discovery Policy and Managing Settings.

zeroFootPrint (ZFP)	(Optional) When an application is configured with "zeroFootPrint": true, the party identity against the local LDAP configuration, rather mandates its vali example-SiteMinder). The ID Token Hint must include the following mandatory attributes along wi • email- This is the email ID configured in the third party. • phone_number- This is the phone number of the user configured in the • user_universalID- This is the login ID of the third-party user. • auto_inline_enrollment- – When set to true, IDSP auto enrolls the credentials based on the val – When set to false, IDSP ignores the credentials sent in the ID Token To know more about ZFP, see here.
SoftMFAEnabled	When an application is configured with "SoftMFAEnabled": true, then the ID are two obligations returned and out of the two, the1st obligation is register challenged with the 2nd factor, and the authentication flow concludes with t the 2nd factor is registered, the authentication challenge is extended to the To know more about SoftMFA, see here.
delegatedAuthentication	This attribute specifies whether the configured flow URL application is built
autoPostToFlowURL	This attribute specifies whether the user claims data should be redirected
userTokenSubAttributeMappingName	This is the name of the logical id store attribute whose physical id store attr issued to this application. This logical attribute has to appear as one of the resource.
supportedJoseHeaderParams	(Optional) This attribute supports comma separated - x5t, x5t#s256, x5c J x5t#S256 are added in the JOSE Header of JWS and JWE tokens by defau

claims	The "claims" attribute lets you configure custom claims for an application. in the Identity and/or Access tokens on behalf of this application as a Relyi Connect relying parties only, requesting Identity and/or Access tokens. Cus request authentication through SAML login requests.
	Syntax:
	`"claims": [`
	`{`
	`"name": "<claim-name>",`
	`"target": "<comma-separated-token-types>",`
	`"value": "${<namespace>.<property>}"`
	`}`
	`]`
	• **name:** Specifies the name of the custom claim to be included in the tok
	• **target:** Specifies the target token type for the custom claim. The input v Access Token.
	• **value**: Specifies the claim value. This can contain dynamic environmen variable value, when this attribute is used. See Dynamic Environment V
	Example:
	`"claims": [`
	`{`
	`"name": "_branchname",`
	`"target": "at,it",`
	`"value": "${idProfile.branch}"`
	`},`
	`{`
	`"name": "_region",`
	`"target": "at,it",`
	`"value": "${appMetadata.region}"`
	`},`
	`{`
	`"name": "_accountnumber",`
	`"target": "at,it",`
	`"value": "${idToken.accountnumber}"`
	`}`
	`{`
	`"name": "_transactionIdentifier",`
	`"target": "at,it",`
	`"value": "${clientContext.transactionIdentifier}"`
	`}`
	`]`
	For more information, see Custom Token Claims.

customClaimProviderToUse	This property determines which custom claims provider should be used to r "providerId" that can be found in the response of the ".../admin/v1/CustomP level (tenant property name-"customClaimProviderToBeUsed"). Applying th tenant level for that particular application. For information on Custom Claim
secondaryAudiences	(Optional) Use this option, when the consuming application needs to valida tokens audience (aud), in addition to the default audience values, (client ID Substitution. (This support is added from IDSP v3.4.4).
assertionVerificationCertAlias	(Optional) This is the certificate alias name for the Trusted client for signatu
appIcon	The image data to upload the logo (of any format) for the application. In cas application name on the login page. The image data is represented in base

secret	Specify the key value for the secret. If this value is not provided, a random
clientId	Specify the key value for the clientId. If this value is not provided, a random
itEncryptionTarget	(Optional) Use this option to enable/disable ID Token encryption. Supported values are: • NONE- ID Token encryption is disabled. • APP- ID Token encryption is enabled and the encryption key is used fro • TENANT- (Case in-sensitive) ID Token encryption is enabled and the T
itEncryptionCertAlias	(Optional) This option represents the Id token encryption cert alias using w mandatory, only when 'itEncryptionTarget' is set to 'APP'.
userInfoEncryptionCertAlias	(Optional) This option represents the user info encryption cert alias using w conjunction with the 'userInfoResponseFormat' and is used only when 'use 'userInfoResponseFormat' is JWT - the userinfo response is signed and the
samlEntityId	This attribute corresponds to the entityID provided by the Service Provider. • In the Service Provider Initiated flow, the "samlEntityId" should match th • In the IDP initiated flow, if the entityId is provided in the request parame configured application.
samlAcsUrl	This is the "AssertionConsumerService" URL obtained from the service pro
samlEnableSingleLogout	This attribute indicates that if a user logs out of IDSP, they will be logged o
samlNameIdFormat	This is the "NameID" format that is supported for the Subject. Supported formats are: emailAddress, unspecified , X509SubjectName
samlVerifyRequestSignature	This attriute indicates whether the signature of the received SAML Authenti
samlVerifyCertAlias	If "samlVerifyRequestSignature"="true", this attribute is mandatory, and refe can be imported in either ways: • Imported automatically from the Service Provider (SP) metadata, if SP • Imported manually using the ".../admin/v1/Certs" API.
samlEncryptSamlResponse	This is a boolean attribute that indicates, if SAML assertion sent by IDSP s
samlEncryptCertAlias	If "samlEncryptSamlResponse"="true", this attribute is mandatory and refer certificate can be imported in either ways: • Imported automatically from the Service Provider (SP) metadata, if SP • Imported manually using the ".../admin/v1/Certs" API.
samlEncryptAlgo	This parameter specifies the encryption algorithm used to encrypt the SAM The following algorithms are supported: • http://www.w3org/2001/04/xmlenc#aes256-cbc • http://www.w3.org/2001/04/xmlenc#aes128-cbc • http://www.w3.org/2001/04/xmlenc#aes192-cbc • http://www.w3.org/2009/xmlenc11#aes128-gcm • http://www.w3.org/2009/xmlenc11#aes192-gcm • http://www.w3.org/2009/xmlenc11#aes256-gcm
samlKeyTransportAlgo	This parameter defines the algorithm used to encrypt the symmetric key tha with the SAML message and is later used by the recipient to decrypt the ac • http://www.w3.org/2001/04/xmlenc#rsa-1_5 • http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p
samlSignResponse	This is a boolean attribute that indicates, if SAML response sent by IDSP s
samlSignResponseKeyAlias	This parameter specifies the alias of the cryptographic key used to digitally

samlSignResponseAlgo	This parameter defines the digital signature algorithm used to sign the SAM
	The following algorithms are supported:
	• http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
	• http://www.w3.org/2000/09/xmldsig#rsa-sha1
	• http://www.w3.org/2001/04/xmldsig-more#rsa-sha512
	• http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256 When using th ensure to use the key type as "EC Key" and this key must be generated
	  • prime256v1
	  • P-256
	  • P-384
	  • P-521
	  • secp256r1
	  • secp384r1
	  • secp521r1

samlSignResponseCanonicalizationAlgo	This parameter defines the canonicalization algorithm used when signing th The following algorithms are supported: • http://www.w3.org/2001/10/xml-exc-c14n# • http://www.w3.org/TR/2001/REC-xml-c14n-20010315 • http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments • http://www.w3.org/2001/10/xml-exc-c14n#WithComments • http://www.w3.org/2006/12/xml-c14n11 • http://www.w3.org/2006/12/xml-c14n11#WithComments
samlSignAssertion	This is a boolean attribute that indicates, if SAML Assertion sent by IDSP a
samlSignAssertionAlgo	This parameter specifies the digital signature algorithm used to sign the SA The following algorithms are supported: • http://www.w3.org/2001/04/xmldsig-more#rsa-sha256 • http://www.w3.org/2000/09/xmldsig#rsa-sha1 • http://www.w3.org/2001/04/xmldsig-more#rsa-sha512 • http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256
samlSignAssertionKeyAlias	This parameter specifies the alias of the private key used to digitally sign th
samlSignAssertionCanonicalizationAlgo	This parameter defines the canonicalization algorithm used when digitally s The following algorithms are supported: • http://www.w3.org/2001/10/xml-exc-c14n# • http://www.w3.org/TR/2001/REC-xml-c14n-20010315 • http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments • http://www.w3.org/2001/10/xml-exc-c14n#WithComments • http://www.w3.org/2006/12/xml-c14n11 • http://www.w3.org/2006/12/xml-c14n11#WithComments
samlSignatureWithKeyInfo	This is a boolean attribute that indicates, if SAML signature should include

samlIdpInitiatedRelaystateMapping

This attribute defines the mapping between relayState provided by the IDP
The relayState value specifies either of the following:

- If IDP init is initiated by external IDP, the relay state value specifies a va
- If the IDP init is initiated by IDSP as the IDP, the relay state value is ser

Note that mapped relayState values are defined using REGEX, necessitati
The following example shows how the relayStateMapping can be defined:

```
"samlIdpInitiatedRelaystateMapping": {

"default": "https://example.com",

"test" : "https://example1.com",

"https://a\.b\.c/.*" : "https://homepage",

"https://myapp.co.com/customers/(.*$)": "https://www.google.c
 match , actual value param is replaced {1}

}
```

The following two IDP init flow scenarios are considered here:

- In the case of IDP init from an external IDP to SSP (as Service Provide relayState value in the IDP init request. If there's an invalid URL in the r following sequence:
  - **Application errorUrl**, if configured
  - **Tenant errorUrl**, if the application errorURL is not configured
  - ***Default Error Page***, if neither the application nor the tenant errorUR

  Following table lists how relayState values are resolved based on

RelayState from External IDP	
No relayState	Redirect to the to define the "d
relayState=test	Redirect to http
relayState=abcd	If no mapping i • ***Applicatio*** • ***Tenant err*** • ***Default Er*** configured.
relayState=https://a.b.c/can-be-anything	Redirects to htt
relayState=https://myapp.co.com/customers/test123	Redirects to htt

If the goal is to redirect to the SAML Assertion Consumer URL (AC include either "samlEffectiveIdpInitiatedUrlWithAppId" or "samlEff

- In the case of IDP init with SSP (where SSP is IDP), SSP sends the rel relayState in the IDP init request.

No relayState	RelayState="" i
relayState=test	RelayState="ht
relayState=abcd	RelayState="ab

This attribute should propagate regex parameter from mapped-from URL to portion of the relayState parameter on the mapped-from URL can be propa ~~Support {N} as substitution marker in relay state's mapped-to URL~~. Each nu be present in the mapped-from URL. If the capturing group is not present, t string.
For example: `https://myapp.acme.com/customers/(.*$) =`
To learn about scenarios of Identity Provider Initiated flows and the behavio to utilize the idp-init flow with relayState mapping capability, ensure to use

samlAssertionClaims	The "samlAssertionClaims" attribute lets you configure custom SAML claim
	these custom SAML claims in the Identity and/or Access tokens on behalf
	Syntax:
	`"samlAssertionClaims": [`
	`{`
	`"name": "<claim-name>",`
	
	`"value": "${<namespace>.<property>}"`
	`}`
	`]`
	• **name:** Specifies the name of the custom SAML claim to be included in
	• **value**: Specifies the SAML claim value. This can contain dynamic envir
	variable value, when this attribute is used. See Dynamic Environment V
	Example:
	`"samlAssertionClaims": [`
	`{`
	`"name": "_branchname",`
	`"value": "${idProfile.branch}"`
	`},`
	`{`
	`"name": "_region",`
	`"value": "${appMetadata.region}"`
	`},`
	`{`
	`"name": "_accountnumber",`
	`"value": "${idToken.accountnumber}"`
	`}`
	`{`
	`"name": "_transactionIdentifier",`
	`"value": "${clientContext.transactionIdentifier}"`
	`}`
	`]`
	For more information, see Custom Token Claims.

samlUserTokenSubAttributeMappingName	This is the name of the logical id store attribute whose physical id store attr issued to this application. This logical attribute has to appear as one of the resource.
skewTimeSecs (number)	Specifies the number of seconds added to the current time to account for th assertion/ ID Token hint expiration.
passwordAuthoritativeSource	This property identifies the authoritative source for password verification. Possible values: • local - indicates that password is stored in IDSP account store (T_USE store • remote (default) - indicates that the password is verified against the rem

mitmProtectionLevel	Specifies whether a factor request is validated at an application level using Device Identifier (TDI) in an HTTP cookie, to mitigate a man-in-the-middle (This field supports one of the following values: • null Specifies that the feature is disabled. This is the default value. • empty Specifies that the feature is disabled. • user-ip Stores the IP address of the authenticated user in the flowstate attribute received, the IP address of the user in the factor request is compared to is processed only if the values match. • user-agent **Using flowstate**: Stores the user agent of the authenticated user in the request is received, the user agent in the factor request is compared to is processed only if the values match. **Using TDI**: If TDI is present as an HTTP cookie in the initial authenticat agent in the TDI. If the value in TDI does not match the value in the aut MFA. • user-ip, user-agent **Using flowstate**: Stores the IP address and user agent of the authentic When a factor request is received, the IP address and user agent of the values that are stored in the flowstate attribute. The request is processe **Using TDI (for user-agent)**: If TDI is present as an HTTP cookie in the compared to the user agent in the TDI. If the value in TDI does not mat user is challenged for MFA. • tenant Specifies that the value of mitmProtectionLevel in an application must b tenant settings. If mitmProtectionLevel is set to "tenant" in an application but the value i about tenant settings, Managing Settings.
idStoreToUse	(Optional) Specifies a single ID store or a comma-separated list of multiple disambiguation and, by default, for user authentication. If this property is no in the tenant configuration.
metadata	This attribute specifies a list of metadata name/value pairs that an administ list, see metadata table.
requireAccessControl	Enables the enforcement of access controls after successful user authentic or controls are applied to user access following the authentication process application. Supported Values are - true/false/tenant (default). • tenant- Uses the configuration that is applied in the tenant setting. This • The default value for this setting has changed from IDSP v4.0.1 onwar • This property can also be configured at the tenant level (tenant property application level overrides the configuration at the tenant level for that p
isSamlApp	Specifies if the application is build using the SAML protocol.
isOidcApp	Specifies if the application is build using the OIDC protocol.
isLauncherApp	Specifies if the application is a Launcher app.

errorURL	This attribute allows specifying a url to an error page, whenever there is an but if the value is not specified, it is taken from the tenant settings. The following redirection behavior occurs for error handling within an applic 1. If an app-level Error URL is not provided, or it is provided as "protocol" 2. If the application intends to redirect to a specific Error URL, one of the f a) The application explicitly specifies its own Error URL. b) If you do not want to specify an app specific URL, you can instruct th info in the application metadata, `urn:app:useGlobalErrorUrl=true`. In this case: — If a tenant-level Error URL is provided, the redirection occurs to that — If no tenant-level Error URL is provided, the redirection takes place — If the application intends to redirect to a built-in Error URL, specify C
homeUrl	When an Identity Provider initiates a global logout, it uses this attribute as t process.
launcherUrl	The launcher app functions by directing users to the URL specified in its co access shortcut for all applications, whether they use protocol flows like SA
logoutUrl	An application maintains a logout URL within its configuration, using during the application does not maintain its own logout URL, this attribute serves a
postLogoutUrl	An application initiating a logout process can either specify a URL for redire

The application creation process collects and assigns the following values to your app. You need to use the following attribute values in the consequent steps:

Attributes	Usage
Application ID	Uniquely identifies your App. This ID is used during the creation of policy.
Client ID	Used during the generation of the Client Access Token in the Authorization header.
Client Secret	Used during the generation of the Client Access Token in the Authorization header.

Managing the Client ID and Secret

The Client ID and Secret are generated by the POST and PUT operations.

When creating an application or updating an existing one, note the following:

- If you do not provide the key value for the Client ID and Secret, then the value is auto-generated.
- If you provide a value for the Client ID and Secret, then those values are used during creation or update.
- If the key value is provided (example- "clientId") then the corresponding data value must not be empty.

The PUT operation allows you to regenerate the Client ID and Client Secret.

```
PUT https://${SSP_FQDN}/{{tenantName}}/admin/v1/Apps/{{appId}}
```

If you set the "regenerateClientId" or "regenerateSecret" to true, (example-"regenerateClientId": true) then the corresponding value is regenerated.

Sample PUT request Payload:

```
{
"status": "active",
"name": "TestApp",
"description": "Default Application",
"clientType": "PUBLIC",
"allowedOpenIDScopes": [
```

```
"openid",
"email",
"profile"
],
"redirectURIs": [
"https://www.example.com"
],
"allowedGrantTypes": [
"implicit",
"authorization_code",
"client_credentials",
"urn:ietf:params:oauth:grant-type:jwt-bearer"
],
"allowedOperations": [
"introspect",
"onbehalfof"
],
"skipIssuerAudienceForIT": "false",
"userInfoEndpointResponseFormat": "PLAIN_JSON",
"secret": "BroadcomSecret",
"clientId": "BroadcomClient",
"regenerateClientId": "true"
"regenerateSecret": "true"
}
```

Updating CORS Setting to Reflect Your Application's FQDN

Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served. The list of "allowedOrigin" identifies the permitted origins to call IDSP APIs.

You are required to edit the CORS setting to add the Fully Qualified Domain Name (FQDN) of the Application that you created. This step is required if you are installing your application into an FQDN different from Symantec Identity Security Platform.

1. To set the CORS setting, update the "allowedOrigins" property (which accepts a comma-separated list of URLs) available in "tenant" settings with the required and valid http/https URLs.
2. Use the following "PATCH" call to update the payload with your Application's FQDN URL.

   NOTE

   - Before running the following command, ensure that you update <Your Application's FQDN> with the actual FQDN and obtain the ${SSP_AT} as mentioned in the ""Obtaining the SSP FQDN and the Access Token to perform administrative tasks"" section.
   - Replace the "https://myapp.example:1200" with the actual URL provided to the *ingress.host* parameter during your app's deployment.

   Example of a cURL Request displaying the Tenant Settings PATCH payload

```
curl --insecure -L -X PATCH "https://"<SSP's FQDN>"/default/admin/v1/Configs" \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer ${SSP_AT}" \
-d '[
{
 "name": "allowedOrigins",
 "value": "https://myapp.example.com:1200"
}
```

```
]'
```

Example of cURL Response

```
[
{
"name": "persistEnrolledFactorsinIDP",
"value": "false",
"origin": "global"
},
{
"name": "azCodeExpirySecs",
"value": "120",
"origin": "global"
},
{
"name": "isInlineOTPEnrollmentToBePersisted",
"value": "true",
"origin": "global"
},
{
"name": "refreshTokenExpiryDays",
"value": "7",
"origin": "global"
},
{
"name": "pushRegTimeoutSecs",
"value": "300",
"origin": "global"
},
{
 "name": "allowedOrigins",
"value": "https://myapp.example.com:1200",
"origin": "tenant"
},
{
"name": "isInlineEnrollmentAllowed",
"value": "true",
"origin": "global"
},
{
"name": "idTokenExpirySecs",
"value": "21600",
"origin": "global"
},
{
"name": "pushAuthTimeoutSecs",
"value": "300",
"origin": "global"
},
{
"name": "accessTokenExpirySecs",
"value": "3600",
"origin": "tenant"
```

```
},
{
"name": "useOriginalIdentityTokenExpiry",
"value": "false",
"origin": "global"
},
{
"name": "inlineEnrollmentFactorsRequired",
"value": "PASSWORD",
"origin": "global"
}
]
```

Deleting an Application Referenced by Other IDSP Objects

If a tenant's application is being deleted and it is currently referenced by other IDSP objects, the `DELETE` API performs a dependency check. If a conflict is detected, and other objects are using the same application, the API prevents the delete operation, and returns error code 8000037.

This allows the administrator to review where the application is being used and manually remove the reference from each object. For example, the application might be used in configurations such as Authentication (AuthN) policy, Authorization (AuthZ) Policy or Discovery Policy. **The delete operation is only allowed once all such references are removed.**

NOTE

- The HTTP response returned in case of a conflict (resource being reused by other IDSP objects) during deletion of an application is **409**.
- When attempting to delete an application that is currently referenced by other IDSP objects, the system returns error code 8000037 to prevent the deletion. See Error Codes here.
- The DELETE operation must contain the "**force=true**" query parameter to delete references of the IDSP objects.
- A Helper API is available to retrieve the list of dependent objects in case of a conflict, see ReferencingObjectsHelper API.

The following DELETE operation is where the reference list gets populated in cases where there are conflicts (resource being reused by other IDSP objects):

```
DELETE https://${SSP_FQDN}/{{tenantName}}/admin/v1/Apps/{{appId}}
```

The following sample response illustrates how this list is displayed.

Sample Response:

```
{
"errorCode": "8000037",

"errorMessage": "Application associated with one or more policies cannot be deleted"
}
```

Reference Links to:

- For more information on Creating Application using the Admin Console, see here.
- Application API Error Codes

Creating a Resource Server Application

Learn about creating and configuring a Resource Server application, covering essential attributes like name, resource scope, audience, and defining specific access scopes (promote, hire, transfer). Includes API request/response example.

This topic provides a detailed walkthrough for creating a Resource Server application. It covers the registration process of a Resource Server application, including the definition of key attributes that ensure secure and controlled access to protected resources. The focus is on setting up the application's core properties and understanding the structure of API requests and responses involved in its creation.

1. **Create a Resource Server App** (HR)
   This step involves registering an "HR" application as a Resource Server. A Resource Server hosts protected resources and accepts and responds to protected resource requests using access tokens.

   ```
   POST https://{{sspHost}}/{{apiPathTenant}}/admin/v1/Apps
   ```

   Request:

   ```
   {
   "status": "active",
   "name": "HR",
   "resScope": "urn:rs:hr",
   "description": "HR Resource Server Application",
   "idpDiscoveryEnabled": null,
   "skewTimeSecs": 0,
   "passwordAuthoritativeSource": "remote",
   "mitmProtectionLevel": null,
   "zeroFootPrint": false,
   "idStoreToUse": null,
   "softMFAEnabled": false,
   "delegatedAuthentication": false,
   "flowURL": null,
   "deviceCodeFlowURL": null,
   "autoPostToFlowURL": false,
   "clientType": "CONFIDENTIAL",
   "regenerateClientId": false,
   "regenerateSecret": false,
   "assertionVerificationCertAlias": null,
   "allowedOpenIDScopes": [
   "openid",
   "profile",
   "groups",
   "address",
   "phone",
   "offline_access",
   "urn:iam:scim:core",
   "urn:iam:scim:ssp",
   "urn:iam:scim:credentials",
   "urn:iam:scim:enterprise",
   "urn:iam:scim:custom",
   "email"
   ],
   "redirectURIs": [
   "https://www.example.com"
   ],
   "allowedGrantTypes": [
   "implicit",
   "client_credentials",
   "urn:ietf:params:oauth:grant-type:device_code",
   "authorization_code",
   "urn:ietf:params:oauth:grant-type:jwt-bearer"
   ```

```
],
"allowedOperations": [
"offlineit",
"introspect",
"onbehalfof"
],
"supportedJoseHeaderParams": "x5t,x5t#s256",
"secondaryAudiences": [],
"claims": [],
"userInfoEndpointResponseFormat": "PLAIN_JSON",
"skipIssuerAudienceForIT": false,
"skipEmailForIT": false,
"userTokenSubAttributeMappingName": null,
"customClaimProviderToUse": null,
"itEncryptionTarget": "NONE",
"itEncryptionCertAlias": null,
"userInfoEncryptionCertAlias": null,
"samlEntityId": null,
"samlAcsUrl": null,
"samlEnableSingleLogout": false,
"samlNameIdFormat": null,
"samlVerifyRequestSignature": false,
"samlVerifyCertAlias": null,
"samlEncryptSamlResponse": false,
"samlEncryptCertAlias": null,
"samlEncryptAlgo": null,
"samlKeyTransportAlgo": null,
"samlSignResponseKeyAlias": null,
"samlSignAssertionKeyAlias": null,
"samlSignResponse": false,
"samlSignAssertion": false,
"samlSignResponseAlgo": null,
"samlSignResponseCanonicalizationAlgo": null,
"samlSignAssertionAlgo": null,
"samlSignAssertionCanonicalizationAlgo": null,
"samlSignatureWithKeyInfo": false,
"samlIdpInitiatedRelaystateMapping": [],
"samlAssertionClaims": [],
"samlUserTokenSubAttributeMappingName": null,
"homeUrl": null,
"launcherUrl": null,
"errorUrl": null,
"logoutUrl": null,
"postLogoutUrl": null,
"metadata": [
{
"name": "urn:iam:app:allowNonSslRedirectUri",
"value": "true"
},
{
"name": "urn:iam:app:allowNonPKCEConnections",
"value": "true"
},
```

```
{
"name": "urn:iam:app:allowAppNotToSendRedirectUri",
"value": "true"
}
],
"label": null,
"sortOrder": null,
"hidden": false,
"isResourceServerApp": true,
"consentUsagePolicyUrl": "https://policy.hr.com",
"primaryAudience": "https://hr.com",
"resourceServerScopes": [
{
"scope": "promote",
"description": "scope to promote",
"consentDescription": "Review, approve, and manage employee promotions within your department",
"label": "View and manage promotions in your department",
"consentRequired": false
},
{
"scope": "hire",
"description": "scope to hire",
"consentDescription": "Review, approve, and manage hiring of new employees within your department",
"label": "View and manage hiring in your department",
"consentRequired": false
},
{
"scope": "transfer",
"description": "scope to transfer",
"consentDescription": "Review, approve, and manage transfer of new employees across departments",
"label": "View and manage transfers in and out of your department",
"consentRequired": false
}
],
"isOidcApp": true,
"isSamlApp": false,
"isLauncherApp": false,
"requireAccessControl": null
}
```


Key attributes for a Resource Server include the following:

NOTE

For all other attribute descriptions, see Description of Application Attributes (OIDC and SAML attributes)

- **name**: The name of the Resource Server application (example, "HR").
- **resScope**: A unique URI that identifies the Resource Server within the authorization server's domain (e.g., "urn:rs:hr"). This scope is used to request access to resources protected by this server.
- **description**: A clear description of the Resource Server's function.
- **isResourceServerApp**: Set to "true" to designate this application as a Resource Server. This enables specific behaviors and configurations related to resource protection.
- **primaryAudience**: The primary audience for access tokens issued for this Resource Server (e.g., "https://hr.com"). Access tokens with this audience are intended for this Resource Server.
- **resourceServerScopes**: A list of scopes defined by the Resource Server, along with their descriptions and whether consent is required. These scopes represent specific permissions that client applications can request to access the Resource Server's protected resources.
  - **promote**: A scope to manage promotions within a department.
  - **hire**: A scope to manage hiring processes within a department.
  - **transfer**: A scope to manage transfers in and out of a department.
- **consentTermsAndConditionsUrl**: The URL for the terms and conditions that users must agree to before granting consent to the client application.
- **consentPrivacyPolicyUrl**: The URL for the privacy policy that users can review before granting consent to the client application.
- **Oidc App**: This property should be set to true to create a resource server app.

Response:


```
{
"appId": "25d25465-db50-44ec-b9ea-5c9b2ef2f8ff",
"status": "active",
"name": "HR",
"resScope": "urn:rs:hr",
"description": "HR Resource Server Application",
"logoId": null,
"idpDiscoveryEnabled": null,
"skewTimeSecs": 0,
"passwordAuthoritativeSource": "remote",
"mitmProtectionLevel": null,
"zeroFootPrint": false,
"idStoreToUse": null,
"softMFAEnabled": false,
"delegatedAuthentication": false,
"flowURL": null,
"deviceCodeFlowURL": null,
"autoPostToFlowURL": false,
"clientType": "CONFIDENTIAL",
"clientId": "d546089d-ea21-4caf-9e3e-87aa231c16c5",
"regenerateClientId": false,
"secret": "bbeee44e-c63c-4932-ade7-0433aebc1724",
"regenerateSecret": false,
"assertionVerificationCertAlias": null,
"allowedOpenIDScopes": [
"address",
"phone",
"openid",
"urn:iam:scim:core",
"urn:iam:scim:ssp",
"offline_access",
```

```
"profile",
"groups",
"urn:iam:scim:enterprise",
"urn:iam:scim:credentials",
"urn:iam:scim:custom",
"email"
],
"redirectURIs": [
"https://www.example.com"
],
"allowedGrantTypes": [
"implicit",
"client_credentials",
"urn:ietf:params:oauth:grant-type:device_code",
"authorization_code",
"urn:ietf:params:oauth:grant-type:jwt-bearer"
],
"allowedOperations": [
"offlineit",
"introspect",
"onbehalfof"
],
"supportedJoseHeaderParams": "x5t,x5t#s256",
"secondaryAudiences": [],
"claims": [],
"userInfoEndpointResponseFormat": "PLAIN_JSON",
"skipIssuerAudienceForIT": false,
"skipEmailForIT": false,
"userTokenSubAttributeMappingName": null,
"customClaimProviderToUse": null,
"itEncryptionTarget": "NONE",
"itEncryptionCertAlias": null,
"userInfoEncryptionCertAlias": null,
"samlEntityId": null,
"samlEntityIdStripe": "25d25465-db50-44ec-b9ea-5c9b2ef2f8ff",
"samlAcsUrl": null,
"samlEnableSingleLogout": false,
"samlNameIdFormat": null,
"samlVerifyRequestSignature": false,
"samlVerifyCertAlias": null,
"samlEncryptSamlResponse": false,
"samlEncryptCertAlias": null,
"samlEncryptAlgo": null,
"samlKeyTransportAlgo": null,
"samlSignResponseKeyAlias": null,
"samlSignAssertionKeyAlias": null,
"samlSignResponse": false,
"samlSignAssertion": false,
"samlSignResponseAlgo": null,
"samlSignResponseCanonicalizationAlgo": null,
"samlSignAssertionAlgo": null,
"samlSignAssertionCanonicalizationAlgo": null,
"samlSignatureWithKeyInfo": false,
```

```
"samlIdpInitiatedRelaystateMapping": [],
"samlAssertionClaims": [],
"samlUserTokenSubAttributeMappingName": null,
"homeUrl": null,
"launcherUrl": null,
"errorUrl": null,
"logoutUrl": null,
"postLogoutUrl": null,
"metadata": [
{
"name": "urn:iam:app:allowNonPKCEConnections",
"value": "true"
},
{
"name": "urn:iam:app:allowNonSslRedirectUri",
"value": "true"
},
{
"name": "urn:iam:app:allowAppNotToSendRedirectUri",
"value": "true"
}
],
"label": null,
"sortOrder": null,
"hidden": false,
"primaryAudience": "https://hr.com",
"resourceServerScopes": [
{
"scope": "promote",
"description": "scope to promote",
"consentRequired": false,
"label": "View and manage promotions in your department",
"consentDescription": "Review, approve, and manage employee promotions within your department"
},
{
"scope": "hire",
"description": "scope to hire",
"consentRequired": false,
"label": "View and manage hiring in your department",
"consentDescription": "Review, approve, and manage hiring of new employees within your department"
},
{
"scope": "transfer",
"description": "scope to transfer",
"consentRequired": false,
"label": "View and manage transfers in and out of your department",
"consentDescription": "Review, approve, and manage transfer of new employees across departments"
}
],
"consentUsagePolicyUrl": "https://policy.hr.com",
"createdBy": {
"principalType": "CLIENT",
"principalId": "ae9964b4-44fa-4976-8557-4097bbfbb9b5",
```

```
"principalName": "DefaultTenantClient"
},
"createdDateTime": "2025-11-03 09:51 AM UTC",
"updatedDateTime": "2025-11-03 09:51 AM UTC",
"adminscope.criteria": [],
"adminscope.managedby": [],
"isOidcApp": true,
"isSamlApp": false,
"isLauncherApp": false,
"requireAccessControl": null,
"isResourceServerApp": true
}
```


Reference Links

- Creating a Resource Server using the Admin Console UI
- Resource Server Application Flow (Includes complete OIDC Flow)
- Signin UI Consent for Resource Server Applications

Application Metadata properties

The following table outlines the metadata attributes used when configuring an application in Symantec Identity Security Platform.

Application Metadata Attribute Name used in Admin Console	Application Metadata Attribute Name used in REST API	Application Metadata
Allow login hint (true/false)	***urn:iam:app:allowLoginHint***	This attribute serves regarding the login id logging in. • When set to "***true*** and accepts a "lo this application du • When set to "***fals*** the client is disreg • Default is set to fa

Claims to be logged (csv)	***urn:iam:app:clientContextClaimsToBeLogged***	This attribute displays
		names received in th
		"urn:iam:authn:client
		For example: If the ID
		the following manner
		"urn:iam:authn:c
		{
		"key": "sm_agent
		"value": "agent"
		},
		{
		"key": "sm_domai
		"value": "testdo
		}
		]
		then, the App metada
		"metadata": [
		{
		"name":
		"urn:iam:app:cl
		"value": "sm_age
		}
		]
		Possible Value: A C
		claims in an Identity T

Http headers to be logged (csv)	***urn:iam:app:httpHeadersToBeLogged***	This attribute allows I names. Use a comma names. You must ma feature.
Use global Error URL (true/false)	***urn:app:useGlobalErrorUrl***	This attribute instruct URL for redirecting u during a sign-in flow. the Error URL configu level Error URL is co built-in /signin/v1/erro not configured, the er • Default is set to fa

OAuth Access Token expiry (seconds)	***urn:iam:app:oauthAccessTokenExpirySecs***	This attribute governs
		number of seconds a
		example shows how
		application:
		{
		"name":
		"urn:iam:app:oa
		"value": "1200"
		}
		Application-level conf
		settings, if both are p

OAuth Identity Token expiry (seconds)	***urn:iam:app:oauthIdentityTokenExpirySecs***	This attribute governs number of seconds a example shows how application:
		{
		"name":
		"urn:iam:app:oa
		"value": "1200"
		}
		Application-level conf settings, if both are p

OAuth Refresh Token expiry (days)	***urn:iam:app:oauthRefreshTokenExpiryDays***	This attribute governs number of days a refr example shows how application:
		{
		"name":
		"urn:iam:app:oa
		"value": "365"
		}
		Application-level conf settings, if both are p

Custom SAML destination URL	***urn:iam:app:samlCustomDestinationUrl***	This attribute ensures contain **destination/** ACS URL.
Allow non SSL URLs (true/false)	***urn:iam:app:allowNonSslRedirectUri***	Set this attribute to '***t*** logout, postlogout UR default, the create/up home, logout, postlog If this attribute is not indicates that non-htt • Default is set to fa
Allow /token request without redirect_uri (true/false)	***urn:iam:app:allowAppNotToSendRedirectUri***	Determines whether is allowed to skip ser standard, the redirect should match the sar call. • Default setting is
Allow non PKCE connections (true/false)	***urn:iam:app:allowNonPKCEConnections***	This attribute ensures the app for authorize attribute to '***true***' to d specified or the value enabled for the app. • Default is set to fa

Allow idp hint (true/false)	***urn:iam:app:allowIdpHint***	When this attribute is specify an idp_hint=< (any app) whose purp In addition to using a IDPs, any app (OIDC specific IDP have a w specific IDP. This is h "urn:iam:app:allowIdp whether the app can specific IDP. The idp_ the IDP cannot be res proceeds as if the hir • idp_hint=urn:iam: • idp_hint=urn:iam: • idp_hint=urn:iam: • idp_hint=urn:iam: discovery process • By default, this at
Successive Client Secret Rotation Interval (Days)	***urn:iam:app:successiveClientSecretRotationIntervalDays***	This attribute sets the rotations. • The default value • If you set this valu multiple secrets. The value of this attri more than the value s attribute. { "name": "urn:iam:app:su "value": "7" } See Client Secret Ro
Old Client Secret Validity Interval (Days)	***urn:iam:app:oldClientSecretValidityDays***	This attribute sets the default value is 7 day { "name": "urn:iam:app:ol "value": "7" } See Client Secret Ro
Do not publish token claims with no data (true/false)	***urn:iam:app:doNotPublishNullTokenClaims***	This attribute specifie should be written out "**false**" (default).
Allow skip redirecting to login app (true/false)	***urn:iam:app:allowSkipRedirectLogin***	Determines whether skiploginredirect" to b valid user session. • If allowed, the ser and if no re-authe authentication cer • Default is set to fa

Session Token Expiry (seconds)	***urn:iam:app:sessionTokenExpirySecs***	Determines SSO ses "3600" implies that th
Ignore SSO Session (true/false)	***urn:iam:app:ignoreSsoSession***	Determines whether (For example: When that the app does not • Default is set to fa
SSO Zone	***urn:iam:app:ssoZone***	Determines the suffix example, app-specifi global cookie name "
Allow skip validating session state (true/false)	***urn:iam:app:allowSkipSessionStateValidation***	Determines whether the `session_stat` `authorize` endpoi parameter is required metadata and is conf cookie policy. Suppor "**false**" (default).
Regex for filtering applicable user groups in Identity and Access Tokens	***iam:urn:app:jwtGroupApplicabilityRegex***	Determines whether the regex should be c Access Token. Suppo
Maximum number of groups allowed in tokens	***iam:urn:app:jwtGroupMaxCount***	Determines the maxi placed into Identity o setting. Supported va
IDP Hint strategy to use when disambiguating 3rd party identity tokens. Either provide a specific Identity Provider to use by ID or Name, or indicate a strategy to locate an applicable Identity Provider. No explicitly provided strategy	***urn:iam:app:idpHintStrategy***	Determines the IDP c processing an incomi authorize", ".../auther flows. • If this metadata it metadata value to following strategie • "none" • "byResourceS • "byIssuer" • "byResourceS • "byToke • **If the strategy va above**: – The system at metadata valu • **If "urn:iam:app"** • The system d ITH's issuer • **ITH Processing**: or not a matching link: Handling Ide **ITH Processing**: or not a matching link:

Construct group context from groups in identity store and identity assertion (true/false)	***urn:iam:app:enableIdentityAssertionGroupPropagation***	If this metadata tag is context from groups i Otherwise the group groups only. This context is made the context rule such user principal conditi This context is also a custom claims: • To get all groups: – ${userProfile.g – $[userProfile.g according to th • To discover wheth group: ${userProf false. • Default is set to fa
SameSite value for session cookie	***urn:iam:app:sameSiteValue***	Determines the Same ***Possible Values***: • None • Lax • Strict
Skip Auto Redirect on Single IDP	***urn:iam:idp:skipAutoRedirectOnSingleIdp***	Determines whether IDP if only one IDP w • Default is set to fa
SAML Subject Confirmation expiry (seconds) This application metadata is available for use only from Symantec Identity Security Platform v3.3.3 release onwards.	***urn:iam:app:samlSubjectConfirmationExpirySecs***	Defines the expiration Confirmation. For exa subject asserted in th after one hour. • Defaults to 60 mi
SAML Conditions expiry (seconds) This application metadata is available for use only from Symantec Identity Security Platform v3.3.3 release onwards.	***urn:iam:app:samlConditionsExpirySecs***	Determines the expir • Default is set to 3 provided.
OAuth AZ Code expiry (seconds) This application metadata is available for use only from Symantec Identity Security Platform v3.3.3 release onwards.	***urn:iam:app:oauthAzCodeExpirySecs***	Determines the expir • Default is set to 2 provided.
Support username and password on initial screen (true/false)	***urn:iam:app:supportUsernameAndPasswordOnInitialScreen***	Determines if the Sig password on the initi • false (**default**): Si password from di • true: Signin UI co the initial login sc
Use application as a template client in Dynamic Client Registration (true/false)	***urn:iam:app:useAppAsTemplateClientinDcr***	To enable the Dynam application metadata • When set to true: client for new clie • Newly registered this template app

Use application as registered client in Dynamic Client Registration	***urn:iam:app:useAppAsRegisteredClientInDcr***	When a DCR registra name that correspon with **urn:iam:app:us** metadata, the system application. Additiona not already present, i redirectURIs list.
SAML Subject Confirmation expiry (seconds)	***urn:iam:app:samlSubjectConfirmationExpirySecs***	Determines SAML As example: value of "36 SAML assertion shou
SAML Conditions expiry (seconds)	***urn:iam:app:samlConditionsExpirySecs***	Determines SAML As • Default is set to 3
OAuth AZ Code expiry (seconds)	***urn:iam:app:oauthAzCodeExpirySecs***	Determines OAuth AZ • Default is set to 2
Prioritize IDP redirect over session	***urn:iam:app:prioritizeIdpRedirectOverSession***	Prioritizes IDP redire • Default is set to fa
CAEP endpoint (URL)	***urn:iam:app:caepUri***	Specifies a URL repr on which CAEP even • Metadata value m
CAEP API key secret alias	***urn:iam:app:caepApiKeySecretAlias***	Specifies the alias of key to use as Authori endpoint. • Metadata value m Secret-Object>
Show IDP chooser on multiple matches by username hint or IDP hint (true/false)	***urn:iam:app:showIdpChooserOnMultipleMatchesByHint***	Determines whether IDPs to choose from, username hint or idp
Custom Token Issue	***urn:iam:app:customIssuer***	Specifies custom issu "iss" claim or SAML's instead of the standa url".
Assert this IDP Origin ID on SCIM operations	***urn:iam:app:assertIdpOriginIdOnScimOperations***	Used to constrain SC originId as published asserted from app m • Multiple values re Provider can be p
Maximum number of user sessions	***urn:iam:app:ssoMaxSession***	Max number of end u specific application. T overrides tenant setti • Default is set to " • Possible Values:
User IpAddress Headers to use (csv)	***urn:iam:app:userIpAddressHeaderToUse***	Defines the custom h
OAuth Personal Refresh Token expiry (days)	***urn:iam:app:oauthPersonalRefreshTokenExpiryDays***	Determines Personal • Default is set to 7

Custom Token Subject for OAuth Access Token with Secondary Audiences (supports dynamic substitution syntax)	***urn:iam:app:oauthCustomSubject***	Defines a custom sub which commonly use like "${clientContext.x "${idToken.subject}". The Access Token's " this dynamic express and been authorized "urn:iam:at:appaud" s
Custom Subject for Identity Token/Assertion (supports dynamic substitution syntax) (**available from** IDSP **v3.4.3)**	***urn:iam:app:customSubject***	This metadata field a subject value for an i uses dynamic placeh substitution expressio `${idProfile.<a` data.
Reset SSO Session start and end on reauthentication (true/false) (**available from** IDSP **v3.4.3**)	***urn:iam:app:resetSsoSessionOnReauthentication***	This metadata field in end of a session afte
Account lookup strategy to use when disambiguating 3rd party identity tokens (**available from** IDSP **v3.4.3)**	***urn:iam:app:itHintLookupStrategy***	This metadata field d account in IDSP DB f Metadata values sup • byUserLoginId (d • byIdpUserGuidFa • byIdpUserGuid • byUniversalId
Minimize size of OAuth Access Token generated for secondary audience (true/false) (**available from** IDSP **v3.4.4)**	***urn:iam:app:oauthAppAudAccessTokenMinimalSize***	This metadata field d to the access token w resulting in a smaller • Default is set to fa
Display name template	***urn:iam:app:displayNameTemplate***	Defines how the user user attribute values.
Allow client to control whether presence of user password in SCIM requests should be ignored	***urn:iam:app:ignoreUserPasswordOnScimOperations***	Allows IDSP to ignore client app if the SCIM has scope "urn:iam:t. "allowLocalPassword
Allow SMS credentials of tag	***urn:iam:app:allowSmsCredsOfTag***	Comma-separated lis number credentials to
Allow EMAIL credentials of tag	***urn:iam:app:allowEmailCredsOfTag***	Comma-separated lis credentials to be use
Exclude credentials which are not used for X days	***urn:iam:app:excludeCredsNotUsedForXDays***	Specifies the number on its lastUsed times excluded from use in
VPN required to launch (true/false)	**urn:iam:app:*vpnRequiredToLaunch***	Specifies whether VP application. If set to **t** the **VPN Required** di
Application specific VPN required warning message	**urn:iam:app:*vpnRequiredWarningMessage***	If **urn:iam:app:vpnR** an application, define displayed in the **VPN** accessed.

Application specific knowledge base URL	**urn:iam:app:*knowledgeBaseUrl***	Defines the URL of a provides additional in URL of a resource is to the application and tile. To view the URL, **Knowledge Base** op • **Default**: https://ex
Default SSP Scopes to be used if no scopes are passed in authorize request	**urn:iam:app:*defaultSSPScopes***	Lists the deafult scop in an authorization re values.
Require Risk Control (true/false)	**urn:iam:app:*requireRiskControl***	Specifies the behavio any of the existing ris • If the value is set matching risk poli • If the value is set executed when a • **Default**: false
Generate OAuth Refresh Token on Refresh Token Grant (true/false)	**urn:iam:app:*generateRefreshTokenOnRefreshTokenGrant***	Specifies whether a r • If the attribute is issued if **offline_a** • If the attribute is s issued. • If the attribute is s issued.
Regex for filtering applicable user groups in JWT-based Identity and Access Tokens	**urn:iam:app:*jwtGroupApplicabilityRegex***	Determines whether will be omitted from th Multiple values are pr
Regex for filtering applicable user groups in SAML Assertions	**urn:iam:app:*samlGroupApplicabilityRegex***	Determines whether will be omitted from th Multiple values are pr
Enforces strict compliance with the SAML specification.	**urn:iam:app:*samlStrictCompliance***	When set to true, the of the SAML assertio false. • app_id • session_state • user_guid • idp_guid • user_loginid • user_riskscore • sid

Table 107: v4.0.2 Application Metadata Properties

Application Metadata Attribute Name used in Admin Console	Application Metadata Attribute Name used in REST API	Application Metadata
Allow IVR credentials of tag	***urn:iam:app:allowIvrCredsOfTag***	Comma-separated lis number credentials to

Delay evaluation of IDP Discovery policy	***urn:iam:app:delayProtocolLevelIdpDiscoveryUntilSignin***	Configure this metad Discovery policy utill false). • Default is false.
Enhanced Launch Context	***urn:iam:app:includeAdditionalContextInLaunchUrl***	Applications can rece (logout URL, session launch URL after suc
Skip entire authentication for trusted device	***urn:iam:app:skipAuthenticationForTrustedDevice***	When enabled, this fe to safely and seamles flow. It treats the reco factor, removing the r known, secure device • Default is set to fa

Table 108: mTLS Metadata Properties

Application Metadata Attribute Name used in Admin Console	Application Metadata Attribute Name used in REST API	Application Metadata
OAuth client authentication method	*urn:iam:app:**oauthAuthMethod***	To enable mTLS for t metadata. • Default is set to n • Possible values: – tls_client_auth – client_secret_ – client_secret_ – private_key_jw
Use MTLS cerificate for token binding (true/false)	*(Optional) urn:iam:app:**oauthUseMtlsCertTokenBinding***	Indicates whether to the access token or n • Default is set to fa
OAuth TLS client Authentication method	*urn:iam:app:**tlsClientAuthnMethod***	To configure the clien one of the following v • tls_client_auth_su • tls_client_auth_sa value • tls_client_auth_sa value • tls_client_auth_sa EMAIL value • tls_client_auth_sa value • Default is set to "t
OAuth TLS client authentication value	*urn:iam:app:**tlsClientAuthnValue***	The admin should co comparing with the cl At runtime, when a cl mTLS, the value base example, the full Sub is extracted. Then, co configured "tlsClientA client certificate is co

Require TLS Authentication (true/false)	*(Optional) urn:iam:app:**oauthRequireTlsAuth***	Configure this metad along with the other s secret basic/post, clie • Default is set to fa

Client Secret Rotation

Learn how client secret rotation can be done in Symantec Identity Security Platform.

Client Secret Rotation is a security feature that allows for the safe and periodic update (rotation) of the client secret used in OAuth 2.0 or other authentication frameworks. Client secrets act like passwords that authenticate an application when requesting tokens from an identity provider or authorization server.

IDSP has introduced the Client Secret Rotation feature to improve security by reducing the impact of a compromised client secret and also by complying with security policies that require periodic secret rotations.

Client Secret rotation can be performed for both your own clients and on behalf of other clients (applications) if the admin has been granted the following scope: **"urn:iam:t.clientseccretrotator"**. This scope must be included in the Authorization Header of the Bearer Access Token.

This topic includes the following sections:

- Rotating Client Secrets For Other Clients/Applications
- Application Metadata Configuration
- Rotating Client Secrets For Self Clients/Applications
- Rotating Client Secrets For OOTB Clients/Applications

Rotating client secret for other applications/clients

The following API allows a client to rotate the client secret for other applications/clients:

```
POST https://${SSP_FQDN}/<tenantName>/admin/v1/ClientSecretRotator
```

Request

```
{
"clientId": "8a9d4e31-9d8a-4a0e-98f2-c2f268d55fb8",
"secret": "test",
"newClientSecret":"test2"
}
```

Where,

- clientId - Specifies the client ID (GUID) for which the secret must be rotated. If this value is left empty, the system generates a new secret for the client and returns it in the response.
- secret - The current client secret that is replaced.
- newClientSecret - The new client secret to be set.

Response

On success, a **200 OK** response is returned with the following payload.

```
{
"clientId":"8a9d4e31-9d8a-4a0e-98f2-c2f268d55fb8",
"secret":"test2",
"oldClientSecret":"test"
}
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace {{text}} with the appropriate values in all the endpoints shown in the following table. For example:
  `https://${SSP_FQDN}/<tenantName>/admin/v1/ClientSecretRotator`
  should look similar to:
  `https://example.com/default/admin/v1/ClientSecretRotator`

Application metadata configuration

As part of the client secret rotation feature, two new application metadata attributes have been added. These are "**urn:iam:app:successiveClientSecretRotationIntervalDays**" and "**urn:iam:app:oldClientSecretValidityDays**". See Application metadata section.

Rotating client secret for self applications/clients

NOTE
Before invoking this API, ensure that the "Allowed Operations" attribute is configured with "**client_secret_rotation**" in the POST ".../admin/v1/Apps" API. See Application Attributes (OIDC and SAML attributes).

The following API allows a client to rotate the client secret of their own applications/clients using the client Access Token (for whichever application you intend to rotate the secret):

```
POST https://${SSP_FQDN}/<tenantName>/oauth2/v1/rotatesecret
```

Request

```
{
"newClientSecret":"test2"
}
```

Where,

- newClientSecret - The new client secret to be set.

Response
On success, a **200 OK** response is returned with the following payload.

```
{
"clientId":"8a9d4e31-9d8a-4a0e-98f2-c2f268d55fb8",
"secret":"test2",
"oldClientSecret":"test"
}
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace {{text}} with the appropriate values in all the endpoints shown in the following table. For example:
  `https://${SSP_FQDN}/<tenantName>/oauth2/v1/rotatesecret`
  should look similar to:
  `https://example.com/default/oauth2/v1/rotatesecret`

Rotating client secret for OOTB applications/clients

(**Available in IDSP version 3.4.2 onwards**.) This section explains how to update the secrets for the out-of-the-box (OOTB) IDSP clients, such as, the Admin Console, Sign-In App, and DefaultTenant client.

Follow these steps:

This example focuses on rotating the Admin Console client app's secret.

1. Get the system access token for the system client by invoking the following API:
   ```
   POST https://${SSP_MGMT_FQDN}/system/oauth2/v1/token
   ```
2. Get the app id of the system client by invoking the following API:
   ```
   GET https://${SSP_MGMT_FQDN}/system/admin/v1/Apps
   ```
   This API retrieves all the OOTB clients for a system tenant. From the response of the API, copy the system client app id, which is a GUID. Assuming that the System client app ID is - 45cd3fa4-9590-4504-a61b-4ed6ae4df792.

   NOTE

   - To know how to pass the value of ${SSP_MGMT_FQDN} environment variable, see Configuring IDSP Environment Variables.
   - "system" is the default value for <systemTenantName>.
   - Replace the text {text}/<text> including {}/<> characters with appropriate values in all the endpoints. For example: `https://${SSP_MGMT_FQDN}/<tenantName>/admin/v1/Apps` should look similar to: `https://example.com/system/admin/v1/Apps`
3. Get all the Authorization policies to find the policy ID of the **Integration Admin Authorization Policy**. This policy includes the `urn:iam:t.rotateclientsecret` scope, which is essential for rotating the system client secret using the API.
   ```
   GET https://${SSP_MGMT_FQDN}/system/admin/v1/AuthZPolicies
   ```
   Assuming that the policy ID for the Integration Admin Policy is - 22acde94-13b2-4650-9679-f2447a21e894, this id is used in the subsequent steps.
4. Update the Integration Admin Policy to include the system client app as the condition in the rules. The following example shows how to add the condition:
   ```
   PUT https://{{sspMgmtHost}}/system/admin/v1/AuthZPolicies/22acde94-13b2-4650-9679-f2447a21e894
   {

      "policyName": "IntegrationAdminPolicy",
      "description": "Manage Applications, IDPs, IDP Discovery policies. Also manage Identity Stores,
    Certificates, and Secrets.",
      "status": "active",
      "isGrantableClientOnly": false,
      "isSubdelegatable": true,
      "displayName": "Integration Administrator",
      "apps": [
        {
          "name": "SSP",
          "id": "$SSP_APP"
        }
      ],
      "rules": [
        {
         "conditions": {
          "principal": {
            "clientApp": {
              "operator": "in",
              "value": [
                "45cd3fa4-9590-4504-a61b-4ed6ae4df792"
              ]
            }
          }
   ```

```
    },

      "result": {
        "msg": "This is Integration Admin authorization policy",
        "privileges": [
          "urn:iam:t.integration",
          "urn:iam:t.apps",
          "urn:iam:t.idpconfigs",
          "urn:iam:t.idstoreconfigs",
          "urn:iam:t.certs",
          "urn:iam:t.secrets",
          "urn:iam:t.idpdiscoverypolicies",
          "urn:iam:t.idstorebrowser",
          "urn:iam:t.rotateclientsecret",
          "urn:iam:t.dcr"
        ],
        "constraints": [
          "07472cbd-5a5a-4b8c-9beb-79b13750bb84"
        ]
      }
    }
  ]
}
```

5. Regenerate the system access token for the system client by invoking the following API:

   ```
   POST https://${SSP_MGMT_FQDN}/system/oauth2/v1/token
   ```

   Verify that the scope, **urn:iam:t.rotateclientsecret** is returned in the response of the API.
6. The following API allows the system client to rotate the client secret of the OOTB applications/clients such as Admin Console client:

   ```
   POST https://${SSP_MGMT_FQDN}/system/oauth2/v1/ClientSecretRotator
   ```

   Request:

   ```
   {
   "clientId": "56a9d4e31-9d8a-4a0e-98f2-c2f268d55fb6",
   "secret": "test",
   "newClientSecret":"test2"
   }
   ```

   Where,

   - clientId - "56a9d4e31-9d8a-4a0e-98f2-c2f268d55fb6" - this is the client id of the Admin Console client.
   - secret - test (this is the old secret of the Admin Console client)
   - newClientSecret - The new client secret to be set for Admin Console client.

   Response

   On success, a **200 OK** response is returned with the following payload.

   ```
   {
   "clientId":"56a9d4e31-9d8a-4a0e-98f2-c2f268d55fb6",
   "secret":"test2",
   "oldClientSecret":"test"
   }
   ```

Launcher App

The launcher app provides streamlined access to various applications through a centralized interface, simplifying navigation and enhancing user efficiency.

The launcher app functions by directing users to the URL specified in its configuration, similar to a browser bookmark. It serves as a quick access shortcut for all applications, whether they use protocol flows like SAML/OIDC or are simple web apps without any protocol integration.

Use the following Application API to create a launcher application URL in the Symantec Identity Security Platform:

```
POST https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/Apps
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/Apps`
  should look similar to:
  `https://example.com/default/admin/v1/Apps`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

The following example shows the request payload of the ".../admin/v1/Apps" API where you are required to add the launcher App URL.

Request

Sample where the launcher URL is added:

```
{
        "status": "active",
        "name": "MyNewLaunchApp",
        "description": "MyNewLaunchApp",
        "launcherUrl":"https://www.example.com",
        "isLauncherApp":"true"
    }
```

Where,

- status - Specifies the status of the launcher application. Set this to "active" to make the application accessible to users.
- name - The name of the launcher application, as it appears to users. Ensure this is descriptive and easily recognizable.
- description - A brief description of the launcher application, providing additional context or details about its purpose.
- launcherURL - The URL to which users will be redirected when they select this launcher application. This can point to a web application or a resource, regardless of whether it uses protocol flows (e.g., SAML/OIDC) or is a standalone web app without protocol integration.
- isLauncherApp - Indicates that this is a launcher application.

Response

Sample where the launcher URL gets populated:

```
{
    "status": "active",
    "name": "MyNewLaunchAppNew",
    "description": "MyNewLaunchAppNew",
    "clientType": null,
    "allowedOpenIDScopes": null,
```

```
"allowedGrantTypes": null,
"flowURL": null,
"deviceCodeFlowURL": null,
"userInfoEndpointResponseFormat": null,
"skipIssuerAudienceForIT": false,
"skipEmailForIT": false,
"zeroFootPrint": false,
"softMFAEnabled": false,
"delegatedAuthentication": false,
"autoPostToFlowURL": false,
"userTokenSubAttributeMappingName": null,
"supportedJoseHeaderParams": null,
"claims": null,
"allowedOperations": null,
"secondaryAudiences": null,
"assertionVerificationCertAlias": null,
"appIcon": null,
"secret": "099d3ed2-aa90-4453-a5de-cedad67ea710",
"clientId": "65b1cee9-0c37-45ef-b9ad-0b37fcc5c4ca",
"itEncryptionTarget": null,
"itEncryptionCertAlias": null,
"userInfoEncryptionCertAlias": null,
"samlEntityId": null,
"samlAcsUrl": null,
"samlEnableSingleLogout": false,
"samlNameIdFormat": null,
"samlVerifyRequestSignature": false,
"samlVerifyCertAlias": null,
"samlEncryptSamlResponse": false,
"samlEncryptCertAlias": null,
"samlSignResponse": false,
"samlSignAssertion": false,
"samlSignatureWithKeyInfo": false,
"samlIdpInitiatedRelaystateMapping": null,
"skewTimeSecs": 0,
"passwordAuthoritativeSource": "remote",
"mitmProtectionLevel": null,
"idStoreToUse": null,
"launcherUrl": "https://www.example.com",
"errorUrl": null,
"logoutUrl": null,
"postLogoutUrl": null,
"isSamlApp": false,
"isOidcApp": false,
"isLauncherApp": true,
"appId": "1e1cf0e9-4fa7-4b1a-8eaf-36766678e4e1",
"createdBy": {
    "principalType": "CLIENT",
    "principalId": "aff219b8-e866-4297-bcdb-160693c30134",
    "principalName": "DefaultTenantClient"
},
"createdDateTime": "2023-06-12 12:22 PM UTC",
"updatedDateTime": "2023-06-12 12:22 PM UTC"
```

```
}
```

Create an Authentication Policy to assign it to the launcher App

Request

Sample Launcher app policy.

> **NOTE**
> This example uses the "appId": "1e1cf0e9-4fa7-4b1a-8eaf-36766678e4e1" created for the launcher app in the previous step.

```
{
   "description": "Launch App Policy",
   "status": "active",
   "policyName": "Launch App Policy",
   "apps": [
       {
           "id": "1e1cf0e9-4fa7-4b1a-8eaf-36766678e4e1"
       }
   ],
   "rules": [
       {
           "conditions": {
               "action": {
                   "operator": "in",
                   "value": [
                       "login"
                   ]
               },
               "principal": {
                   "user": {
                       "operator": "in",
                       "value": [
                           "lawrence_haley"
                       ]
                   }
               },
               "channel": {
                   "operator": "in",
                   "value": [
                       "web"
                   ]
               }
           },
           "result": {
               "effect": "allow",
               "obligations": [
                   "PASSWORD"
               ],
               "msg": "This policy is for login in web channel and Launch App",
               "authLevel": 100
           }
       }
   ]
```

```
}
```

Response

Sample where the launcher app details get populated with its "name" and "id":


```
{
    "policyId": "708e658f-2061-47f4-85bd-b01f7b9bc7cd",
    "policyName": "Launch App Policy",
    "description": "Launch App Policy",
    "status": "active",
    "matchAnyApp": false,
    "apps": [
        {
            "name": "MyNewLaunchAppNew",
            "id": "1e1cf0e9-4fa7-4b1a-8eaf-36766678e4e1"
        }
    ],
    "rules": [
        {
            "id": "d5eca988-98f9-4a13-9db3-43ee657d0b41",
            "conditions": {
                "principal": {
                    "user": {
                        "operator": "in",
                        "value": [
                            "lawrence_haley"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "login"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD"
                ],
                "msg": "This policy is for login in web channel and Launch App",
                "authLevel": 100
            }
        }
    ],
    "createdBy": {
```

```
        "principalType": "CLIENT",
        "principalId": "aff219b8-e866-4297-bcdb-160693c30134",
        "principalName": "DefaultTenantClient"
    },
    "createdDate": "2023-06-12 12:24 PM UTC"
}
```

Accessing the launcher API

You can access the launcher App by using the following URL:

```
https://${SSP_FQDN}/<tenant-name>/launch/v1/Apps/<appId>
```

Examples of Launch App Configurations

- Configure the following request to enable Launcher App URL to access an application using the SAML Identity Provider (IDP) initiated flow:
  Request:

```
{
   "status": "active",
   "name": "LaunchAppIdpInitSSP",
   "description": "LaunchAppIdpInitSSP",
   "launcherUrl":"https://ssp.siteminder-poc.mydomain.com/default/saml/v1/idp/login?entityId=https://
sampleapp.siteminder-poc.mydomain.com",
   "isLauncherApp": "true"
}
```

  Response:
  The response in this sample is showing only the "launcherUrl", ignoring the other response attributes.

```
"launcherUrl": "https://ssp.siteminder-poc.mydomain.com/default/saml/v1/idp/login?entityId=https://
sampleapp.siteminder-poc.mydomain.com"
```

- Configure the following request to enable Launcher App URL to access an application using the SAML Service Provider (SP) initiated flow:
  Request:

```
{
    "status": "active",
    "name": "LaunchAppSPInit",
    "description": "LaunchAppSPInit",
    "launcherUrl":"https://sampleapp.siteminder-poc.mydomain.com/sample-rp/saml/login",
    "isLauncherApp": "true"
}
```

  Response:
  The response in this sample is showing only the "launcherUrl", ignoring the other response attributes.

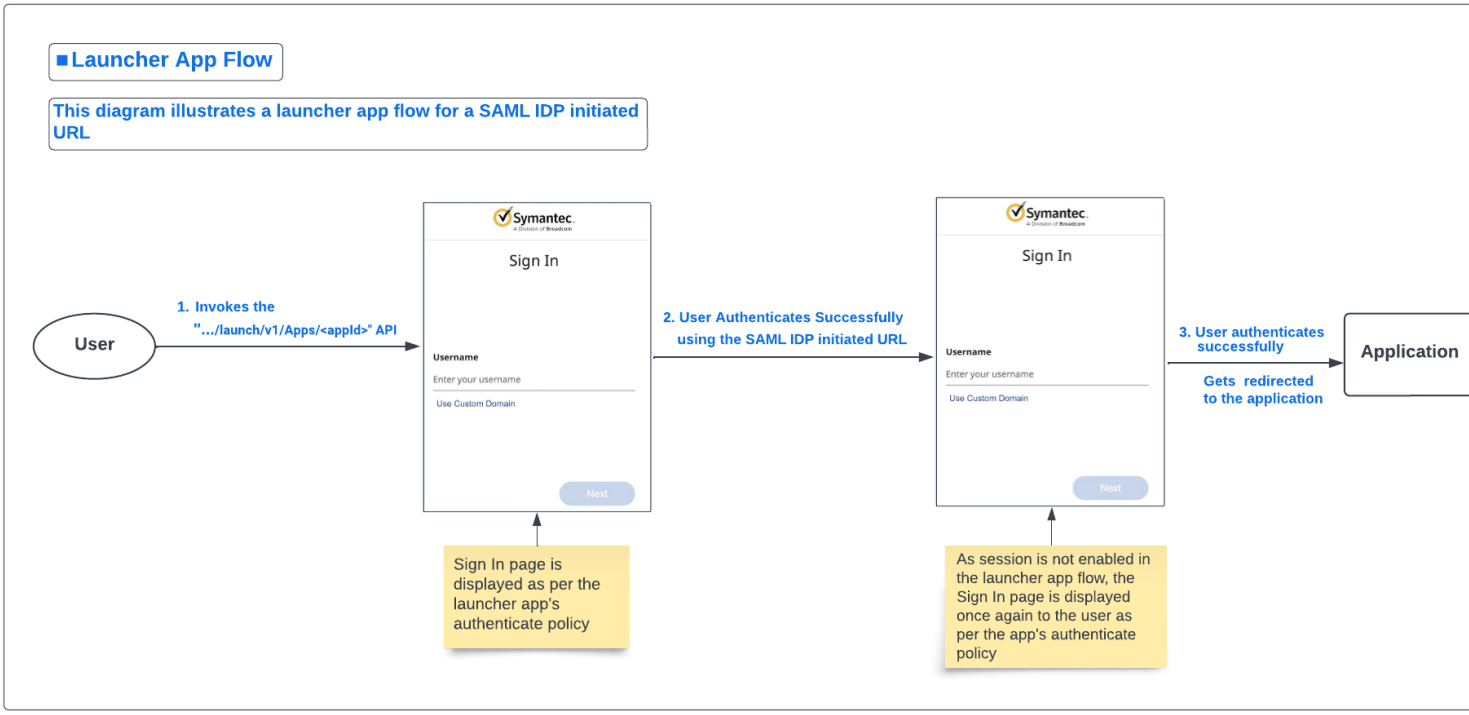
```
"launcherUrl": "https://sampleapp.siteminder-poc.mydomain.com/sample-rp/saml/login"
```

- Configure the following request to enable Launcher App URL without Session Flow:
  Request:

```
{
    "status": "active",
    "name": "LaunchAppIdpInitSSP",
    "description": "LaunchAppIdpInitSSP",
```

```
    "launcherUrl":"https://ssp.siteminder-poc.mydomain.com/default/saml/v1/idp/login?entityId=https://
sampleapp.siteminder-poc.mydomain.com",
    "isLauncherApp": "true"
}
```

Response:
A launcher app flow without a session involves accessing an application using a shortcut link, where each access initiates a new authentication process without maintaining a persistent session within the launcher app.
In this example, the launcher app flow operates without an enabled session, causing the Sign In UI to appear twice. This occurs because policy evaluation takes place at each access point resulting in a new authentication flow at each of these access points. In this case, the first authentication takes place at IDSP, and the next authentication takes place at the location where the app is hosted. Upon successful authentication, the user is redirected to the application.



Where,

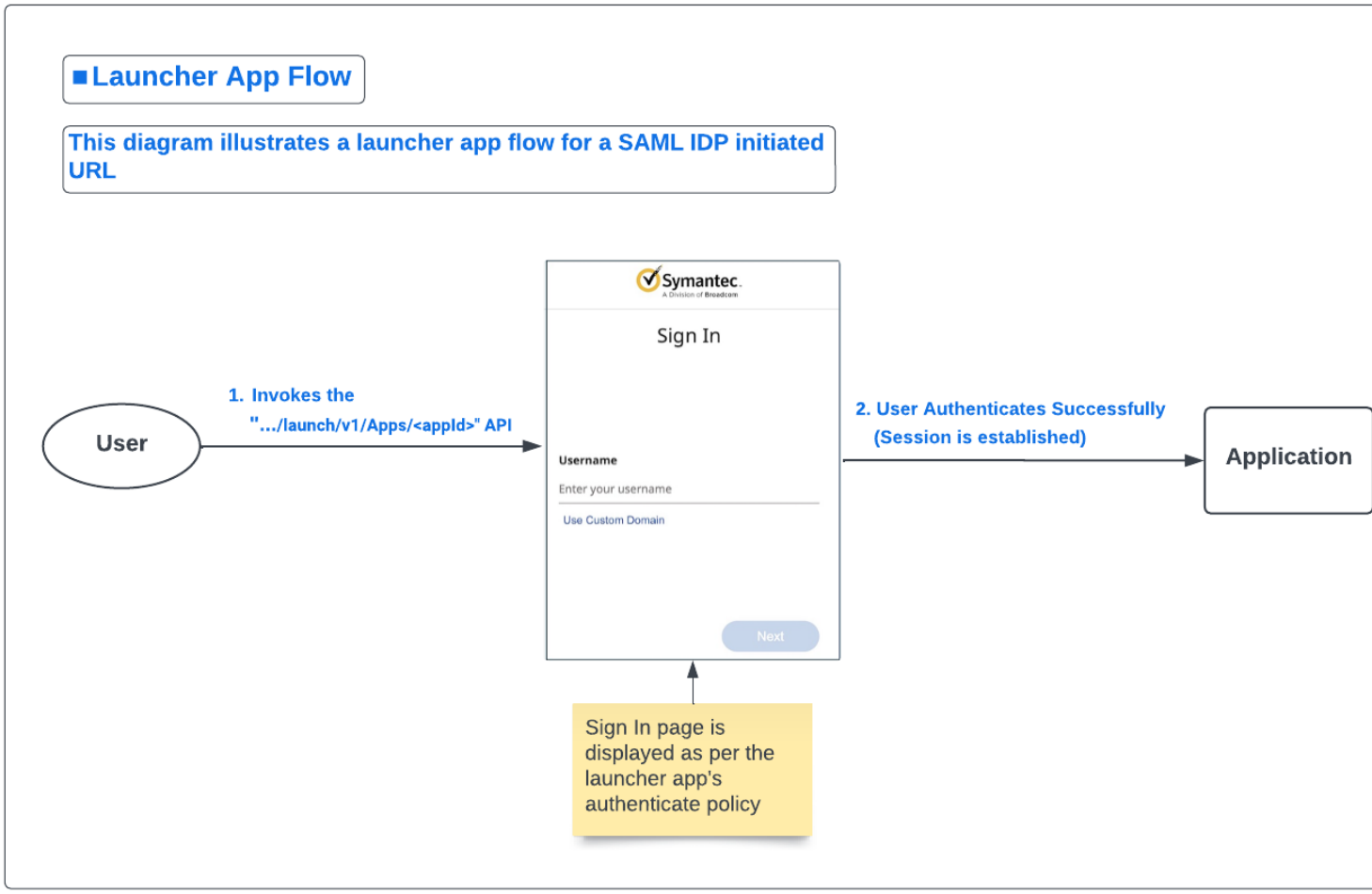
- App Launcher URL - https://${SSP_FQDN}/<tenant-name>/launch/v1/Apps/<appId>
- Home URL - https://${SSP_FQDN}/saml/v1/idp/login?appid=<value>

- Configure the following request to enable Launcher App URL with Session Flow:
  To enable session flow, you are required to configure the "ssoMode" attribute using the following API:

```
PATCH https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/Configs
[
  {
    "name": "ssoMode",
    "value": "enabledCookie"
  }
]
```

Response:
A launcher app flow with a session offers efficient access to applications, with the session providing continuity and convenience across multiple accesses without the need for repeated authentication.

In this example, the user invokes the launcher app URL, containing the shortcut to the web application. This triggers a redirection to the application's authentication endpoint. If a session is established in the launcher app, the authentication process may automatically log in the user or prompt for credentials if necessary. After successful authentication, the user gains access to the application.



Where,

- App Launcher URL -https://${SSP_FQDN}/<tenant-name>/launch/v1/Apps/<appId>
- Home URL - https://${SSP_FQDN}/saml/v1/idp/login?appid=<value>

For details on creating a Launcher App using the Admin Console, see here.

Symantec Identity Security Platform Operating as an Identity Provider

In this scenario, the IDSP provides authentication services to a Service Provider (SP), which could be another application or a service.

The interaction between the IDSP functioning as an Identity Provider (IDP) with the Service Provider (SP) typically perform the following steps for user authentication:

1. The SP relies on the IDSP to authenticate its users.
2. Users attempting to access the SP are redirected to the IDSP for authentication.

3. After successful authentication, the IDSP sends an assertion or token back to the SP.
4. The SP trusts this assertion and grants the user access to the SP's resource.

To accomplish the above steps, configurations need to be implemented on both ends, both at the IDSP (IDP) and the Service Provider's (SP) end. These configurations enable the connection between the two parties and initiate the federated authentication flow.

To extend authentication services to an external service provider, create an application within IDSP through the Admin Console or IDSP APIs. This involves configuring the necessary details obtained from the Service Provider's metadata to set up the application within IDSP.

OIDC Flow

IDSP Operating as an Identity Provider

In the following steps, IDSP is operating as an Identity Provider. Follow these steps to configure an external Service Provider in IDSP:

1. Download the Service Provider's metadata.
   The service provider creates an IDP at their end. Once the Service Provider creates the IDP with the details from IDSP and saves it, this information in the form of metadata is downloadable by IDSP from the Service Provider's site.
   The following is a sample of IDP metadata that can be downloaded from a service provider:

```
<?xml version="1.0" encoding="UTF-8"?>md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
 entityID="https://de-template-may-7-15-dev-ed.my.salesforce.com" validUntil="2033-10-14T09:07:20.522Z"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<md:IDPSSODescriptor WantAuthnRequestsSigned="false"
 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
<md:KeyDescriptor use="signing">
         <ds:KeyInfo>
            <ds:X509Data>
<ds:X509Certificate>MIIEhDCCA2ygAwIBAgIOAYs83+C3AAAAAFVR
+ucwDQYJKoZIhvcNAQELBQAwgYMxGzAZBgNVBAMMElNTUF9DZXJ0X09jdDE3MjAyMzEYMBYGA1UECwwPMDBENWcwMDAwMEw4Wjd6MRcwFQYDVQQKDA5T
+cP/vc5HWb5d3zP2cFNCcbrD4cy8tXOQaehAAg2bVUT5pFgm9IE+hLSfJLLqz2Q8miTSKGgPLa2tu
+D8KCYcv2uRMt8OpCf2BzZ66xlit0y9c4aGD6XbHuu4iCNUCtVP8V7a5xz5c1zk3OXXg7YHqzp3VjfG9Mcy9WjQcW/
bupOsJECVyAo/5rZSLcM12+hJks7y7ieHkelEgNrLysDnIotKh0mN4TWOBzmkn/
uNZunxv7PAgMBAAGjgfMwgfAwHQYDVR0OBBYEFGoiSOhQET3675k7AVRI/mHuD9K7MA8GA1UdEwEB/
wQFMAMBAf8wgb0GA1UdIwSBtTCBsoAUaiJI6FARPfrvmTsBVEj
+Ye4P0ruhgYmkgYYwgYMxGzAZBgNVBAMMElNTUF9DZXJ0X09jdDE3MjAyMzEYMBYGA1UECwwPMDBENWcwMDAwMEw4Wjd6MRcwFQYDVQQKDA5TYWxlc2
+ucwDQYJKoZIhvcNAQELBQADggEBAB0YEyKx8KEwgqFYp9OumcvPLe+GjIscQDwAYlPtkWXOvLuLvzFtLoVik3e/
Et8R6kqjjbYmAFsNaCHbJQZ+1+SQJD9qtWIl4Od5y1Fie/b0aWGed0vjEVwelGoSI4SM1T6L0fVnQuhidL+gicc8m56a4pU
+/7yA3vxW31KBIuVa3nwbFrOdgsWQrJENnIfY3M0CPtHfhXcvRvxV6CC878UrNZROsmm9XkMluHdd/PuTlP5vHCMbY/
M6t0vr6fBgQ2A3tUWRbYY3uV/81o5u6s6LnPwUb2fJ0QMbYGMO+gnIopkyLJzVLSy081azhXxSy6Cy2NpJ/WQsNOuSBMDaxHo=</
ds:X509Certificate>
            </ds:X509Data>
         </ds:KeyInfo>
      </md:KeyDescriptor>
      <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameIDFormat>
      <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://
broadcominc-dev-ed.develop.my.salesforce.com/idp/endpoint/HttpPost"/>
      <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
 Location="https://broadcominc-dev-ed.develop.my.salesforce.com/idp/endpoint/HttpRedirect"/>
   </md:IDPSSODescriptor>
</md:EntityDescriptor>
```

2. Create an OIDC application in IDSP for the service provider.
   Use the following Application API to create a new application in IDSP:

```
https://{{sspHost}}/{{apiPathTenant}}/admin/v1/Apps
```

Creating Application Using OIDC Protocol:

The following example shows the request payload of the ".../admin/v1/Apps" API using the OIDC Protocol. In addition to the obligatory attributes displayed in the sample, there are several optional attributes that can be included in the payload by performing a PATCH call.

```
{
    "status": "active",
    "name": "democlient",
    "description": "Demo client to perform tenant level operations for demo purpose.",
    "clientType": "TRUSTED",
    "allowedOpenIDScopes": [
        "address",
        "phone",
        "openid",
        "profile",
        "groups",
        "email"
    ],
    "redirectURIs": [
        "https://www.example.com"
    ],
    "allowedGrantTypes": [
        "refresh_token",
        "implicit",
        "client_credentials",
        "authorization_code",
        "urn:ietf:params:oauth:grant-type:jwt-bearer"
    ],
    "flowURL": null,
    "deviceCodeFlowURL": null,
    "userInfoEndpointResponseFormat": "PLAIN_JSON",
    "skipIssuerAudienceForIT": false,
    "skipEmailForIT": false,
    "zeroFootPrint": false,
    "softMFAEnabled": false,
    "userTokenSubAttributeMappingName": null,
    "supportedJoseHeaderParams": null,
    "claims": [
        {
            "name": "_title",
            "target": "at,it",
            "value": "${idProfile.title}"
        },
        {
            "name": "_idp_name",
            "target": "at,it",
            "value": "${idToken.idp_name}"
        },
        {
            "name": "_env",
            "target": "at,it",
            "value": "${appMetadata}.env}"
        }
```

```
            {
                "name": "_acct_num",
                "target": "at,it",
                "value": "12345"
            }
            {
                "name": "_transactionIdentifier",
                "target": "at,it",
                "value": "${clientContext.transactionIdentifier}"
            }
        ],
        "allowedOperations": [
            "itgroups",
            "atgroups",
            "introspect"
        ],
        "secondaryAudiences": [
            "env=blue",
            "https://consumer.com/myapi"
        ],
        "assertionVerificationCertAlias": null,
        "appIcon": null,
        "secret": "b62989d7-b992-4a77-88c8-5f6595eeedf3",
        "clientId": "b72ca551-59ec-4e07-ac5b-d09f19c663a8",
        "itEncryptionTarget": "NONE",
        "itEncryptionCertAlias": null,
        "userInfoEncryptionCertAlias": null,
    }
```


For more information on how to create an app and to know about the description of the attributes in IDSP, see Managing Application.

Example: Integrating a SAML Service Provider with IDSP

In the following scenario, Symantec Identity Security Platform operates as an Identity Provider, and Salesforce functions as the Service Provider. Configurations must be applied on both ends to establish the connection between the two parties and initiate the federated authentication flow. Therefore the integration setup between both the parties is described though the following steps.

Prerequisites: Ensure that you have met the following prerequisites before progressing with the SalesForce integration.

- A Public IP Machine or a Cloud Provider that has a Kubernetes cluster that is provisioned and IDSP deployed on it.
- An active SalesForce account to perform the integration.

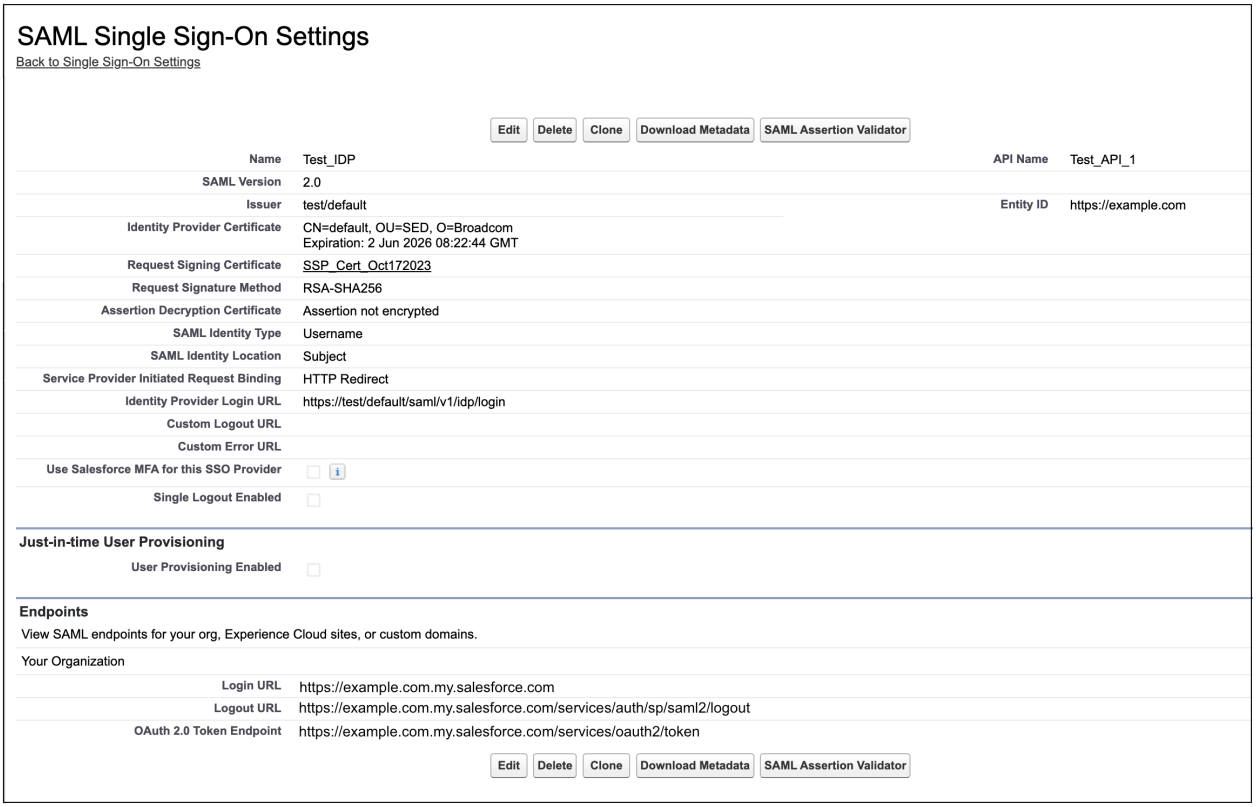
> **NOTE**
>
> Ensure that you use a CA Signed certificate to enable SSL to the ingress controller. This implies that all IDSP urls with https should use proper CA signed certificates. This is important as the *https://${SSP_FQDN}/default/oauth2/v1/token endpoint* gets called from the backend SalesForce server to trust the certificate chain.

IDSP Operating as an Identity Provider

In the following steps, IDSP is operating as a SAML Identity Provider and assuming SalesForce as the Service Provider.

Follow these steps:

1. Register IDSP as an Identity Provider with the Service Provider.
   a. Log into the SalesForce site.
   b. Navigate to ***Setup > Security Controls > Single Sign-On Settings.***
   c. Click "**New From Metadata**" file to add IDSP SAML information.
   d. Select the metadata file from your system and click **Create.**
   e. Configure the following IDSP SAML attributes:
      a. Name
      b. Issuer
      c. Identity Provider Certificate
      d. SAML Identity Type - Set to the default - **Assertion contains the User's Salesforce username**
      e. SAML Identity Location - Set to the default - **Identity is in the NameIdentifier element of the Subject statement**
      f. Service Provider Initiated Request Binding - Set to the default - **HTTP Redirect**
      g. Identity Provider Login URL
   f. **Save** the configuration.
      The following URLs are generated at this point:
      - Login URL
      - Logout URL



2. Download the Service Provider's metadata.
   The following is a sample of IDP metadata that can be downloaded from a service provider:

```
<?xml version="1.0" encoding="UTF-8"?>md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
 entityID="https://de-template-may-7-15-dev-ed.my.salesforce.com" validUntil="2033-10-14T09:07:20.522Z"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
```

```
<md:IDPSSODescriptor WantAuthnRequestsSigned="false"
 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
<md:KeyDescriptor use="signing">
         <ds:KeyInfo>
            <ds:X509Data>
<ds:X509Certificate>MIIEhDCCA2ygAwIBAgIOAYs83+C3AAAAAFVR
+ucwDQYJKoZIhvcNAQELBQAwgYMxGzAZBgNVBAMMElNTUF9DZXJ0X09jdDE3MjAyMzEYMBYGA1UECwwPMDBENWcwMDAwMEw4Wjd6MRcwFQYDVQQKDA5T
+cP/vc5HWb5d3zP2cFNCcbrD4cy8tXOQaehAAg2bVUT5pFgm9IE+hLSfJLLqz2Q8miTSKGgPLa2tu
+D8KCYcv2uRMt8OpCf2BzZ66xlit0y9c4aGD6XbHuu4iCNUCtVP8V7a5xz5c1zk3OXXg7YHqzp3VjfG9Mcy9WjQcW/
bupOsJECVyAo/5rZSLcM12+hJks7y7ieHkelEgNrLysDnIotKh0mN4TWOBzmkn/
uNZunxv7PAgMBAAGjgfMwgfAwHQYDVR0OBBYEFGoiSOhQET3675k7AVRI/mHuD9K7MA8GA1UdEwEB/
wQFMAMBAf8wgb0GA1UdIwSBtTCBsoAUaiJI6FARPfrvmTsBVEj
+Ye4P0ruhgYmkgYYwgYMxGzAZBgNVBAMMElNTUF9DZXJ0X09jdDE3MjAyMzEYMBYGA1UECwwPMDBENWcwMDAwMEw4Wjd6MRcwFQYDVQQKDA5TYWxlc2
+ucwDQYJKoZIhvcNAQELBQADggEBAB0YEyKx8KEwgqFYp9OumcvPLe+GjIscQDwAYlPtkWXOvLuLvzFtLoVik3e/
Et8R6kqjjbYmAFsNaCHbJQZ+1+SQJD9qtWIl4Od5y1Fie/b0aWGed0vjEVwelGoSI4SM1T6L0fVnQuhidL+gicc8m56a4pU
+/7yA3vxW31KBIuVa3nwbFrOdgsWQrJENnIfY3M0CPtHfhXcvRvxV6CC878UrNZROsmm9XkMluHdd/PuTlP5vHCMbY/
M6t0vr6fBgQ2A3tUWRbYY3uV/81o5u6s6LnPwUb2fJ0QMbYGMO+gnIopkyLJzVLSy081azhXxSy6Cy2NpJ/WQsNOuSBMDaxHo=</
ds:X509Certificate>
            </ds:X509Data>
         </ds:KeyInfo>
      </md:KeyDescriptor>
      <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameIDFormat>
      <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://
broadcominc-dev-ed.develop.my.salesforce.com/idp/endpoint/HttpPost"/>
      <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
 Location="https://broadcominc-dev-ed.develop.my.salesforce.com/idp/endpoint/HttpRedirect"/>
   </md:IDPSSODescriptor>
</md:EntityDescriptor>
```

3. Register a SAML application on behalf of the service provider in IDSP. For more information on how to create an app in IDSP, see Managing Application. Configure the application using the information provided in the Service Provider's metadata that you have downloaded in the previous step.
   a. Use the following endpoint to register an application in IDSP:

      `POST https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/Apps`

      NOTE

      - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
      - The default value for `<tenantName>` is "default".
      - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.

        For example:

        `https://{{host}}/{{tenantName}}/admin/v1/Apps`

        should look similar to:

        `https://example.com/default/admin/v1/Apps`
      - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

b. The following table describes the SAML attributes that must be configured in the application to register it as a SAML service provider:

Attribute Name	Description
samlEntityId	This attribute corresponds to the entityID provided by the Service Provider. • In the Service Provider Initiated flow, the "samlEntityId" should match the Issuer in the authentication request. • In the IDP initiated flow, if the entityId is provided in the request parameter, it should match the configured "samlEntityId" to point to the configured application.
samlAcsUrl	This is the "AssertionConsumerService" URL obtained from the service provider, where the SAML response is sent to.
samlEnableSingleLogout	This attribute indicates that if a user logs out of IDSP, they are logged out of the Service Provider.
samlNameIdFormat	This is the "NameID" format that is supported for the Subject. Supported formats are: emailAddress, unspecified, X509SubjectName
samlVerifyRequestSignature	This attriute indicates whether the signature of the received SAML Authentication request should be verified or not.
samlVerifyCertAlias	If "samlVerifyRequestSignature"="true", this attribute is mandatory, and refers to the alias of the certificate used for verification. The certificate can be imported in either ways: • Imported automatically from the Service Provider (SP) metadata, if SP metadata xml is available. • Imported manually using the ".../admin/v1/Certs" API.
samlEncryptSAMLResponse	This is a boolean attribute that indicates, if SAML Response sent by IDSP should be encrypted or not.
samlEncryptCertAlias	If "samlEncryptSamlResponse"="true", this attribute is mandatory and refers to the alias of the certificate used for verification. The certificate can be imported in either ways: • Imported automatically from the Service Provider (SP) metadata, if SP metadata xml is available. • Imported manually using the ".../admin/v1/Certs" API.
samlIdpInitiatedRelaystateMapping	This attribute defines the mapping b/w relayState provided during the IDP Initiated flow and actual relative URL of the app, where the flow ends. "samlIdpInitiatedRelaystateMapping": { "PORTAL": "test/Users/home" }

<table>
<tr><td>samlAssertionClaims</td><td>The "samlAssertionClaims" attribute lets you configure custom SAML claims for an application. IDSP includes these custom SAML claims in the Identity and/or Access tokens on behalf of this application as a Relying Party.
Syntax:
<pre>"samlAssertionClaims": [
{
"name": "<claim-name>",

"value": "${<namespace>.<property>}"
}
]</pre>
• name: Specifies the name of the custom SAML claim to be included in the token.
• value: Specifies the SAML claim value. This can contain dynamic environment variables which will be substituted with their corresponding variable value, when this attribute is used. See Dynamic Environment Variable Substitution.
Example:
<pre>"samlAssertionClaims": [
{
"name": "_branchname",
"value": "${idProfile.branch}"
},
{
"name": "_region",
"value": "${appMetadata.region}"
},
{
"name": "_accountnumber",
"value": "${idToken.accountnumber}"
}
{
"name": "_transactionIdentifier",
"value": "${clientContext.transactionIdentifier}"
}
]</pre>
For more information, see Custom Token Claims.</td></tr>
<tr><td>samlUserTokenSubAttributeMappingName</td><td>This is the name of the logical id store attribute whose physical id store attribute's value is to be placed in the "sub" token claims for IT and AT issued to this application. This logical attribute has to appear as one of the mappings in the "attributeMapping" property of the id store config resource.</td></tr>
<tr><td>samlSignatureWithKeyInfo</td><td>This is a boolean attribute that indicates, if SAML signature should include the KeyInfo or not.</td></tr>
<tr><td>samlSignResponse</td><td>This is a boolean attribute that indicates, if the SAML response sent by IDSP should be signed or not.</td></tr>
<tr><td>samlSignAssertion</td><td>This is a boolean attribute that indicates, if SAML Assertion sent by IDSP as part of the SAML response should be signed or not.</td></tr>
</table>

Sample SAML App API request:

```
{
        "status": "active",
        "name": "SP-saml",
        "description": "Application for SP integration",
```

```
        "samlAssertionClaims": [
          {
            "name": "ctry",
            "value": "${idProfile.ctry}"
          },
          {
            "name": "fname",
            "value": "${idProfile.fname}"
          },
          {
            "name": "lname",
            "value": "${idProfile.lname}"
          }
        ],
        "samlUserTokenSubAttributeMappingName": "user_loginid",
        "samlEntityId": "https://www.sp.com",
        "samlAcsUrl": "https://www.sp.com/saml/acs",
        "samlEnableSingleLogout": false,
        "samlNameIdFormat": "unspecified",
        "samlVerifyRequestSignature": false,
        "samlVerifyCertAlias": null,
        "samlEncryptSamlResponse": false,
        "samlEncryptCertAlias": null,
        "samlIdpInitiatedRelaystateMapping": {}
}
```


Sample SAML App API response:


```
{
        "status": "active",
        "name": "SP-saml",
        "description": "Application for SP integration",
        "clientType": "CONFIDENTIAL",
        "allowedOpenIDScopes": [
            "openid",
            "profile",
            "email"
        ],
        "redirectURIs": [
            "www.example.com"
        ],
        "allowedGrantTypes": [
            "implicit",
            "authorization_code"
        ],
        "flowURL": null,
        "deviceCodeFlowURL": null,
        "userInfoEndpointResponseFormat": "PLAIN_JSON",
        "skipIssuerAudienceForIT": true,
        "skipEmailForIT": false,
        "idpDiscoveryEnabled": null,
        "zeroFootPrint": false,
        "softMFAEnabled": false,
        "delegatedAuthentication": false,
```

```
        "autoPostToFlowURL": false,
        "userTokenSubAttributeMappingName": null,
        "supportedJoseHeaderParams": null,
        "claims": [],
        "allowedOperations": null,
        "secondaryAudiences": null,
        "assertionVerificationCertAlias": null,
        "appIcon": null,
        "secret": "388a1f0d-7dd1-45e9-915f-341346970ded",
        "clientId": "cac65437-3588-4b61-989b-45a56bdc18df",
        "itEncryptionTarget": "NONE",
        "itEncryptionCertAlias": null,
        "userInfoEncryptionCertAlias": null,
        "samlEntityId": "https://www.sp.com/saml2/service-provider/splylemeeypkgaksryyt",
        "samlAcsUrl": "https://sp.com/sso/saml2/0oadhf7dftXhv66I85d7",
        "samlEnableSingleLogout": false,
        "samlNameIdFormat": "unspecified",
        "samlVerifyRequestSignature": false,
        "samlVerifyCertAlias": null,
        "samlEncryptSamlResponse": false,
        "samlEncryptCertAlias": null,
        "samlSignResponse": false,
        "samlSignAssertion": true,
        "samlSignatureWithKeyInfo": true,
        "samlIdpInitiatedRelaystateMapping": {
            "PORTAL": "home"
        },
        "samlAssertionClaims": [],
        "samlUserTokenSubAttributeMappingName": null,
        "skewTimeSecs": 0,
        "passwordAuthoritativeSource": "remote",
        "mitmProtectionLevel": null,
        "idStoreToUse": null,
        "samlEffectiveIdpInitiatedUrlWithEntityId": "https://idp.com/default/saml/v1/idp/login?
entityId=https://www.sp.com/saml2/service-provider/splylemeeypkgaksryyt",
        "samlEffectiveIdpInitiatedUrlWithAppId": "https://idp.com/default/saml/v1/idp/login?
appId=eaeb66b0-30b5-4852-8dc6-e79664b2d7ac",
        "samlEffectiveSpAcsUrlWithEntityId": "https://idp.com/default/saml/v1/sp/acs?sp=https://
www.okta.com/saml2/service-provider/splylemeeypkgaksryyt",
        "samlEffectiveSpAcsUrlWithAppId": "https://idp.com/default/saml/v1/sp/acs?
sp=eaeb66b0-30b5-4852-8dc6-e79664b2d7ac",
        "samlEffectiveSpAcsUrlWithAppName": "https://idp.com/default/saml/v1/sp/acs?sp=sp-saml",
        "errorUrl": null,
        "appId": "eaeb66b0-30b5-4852-8dc6-e79664b2d7ac",
        "createdBy": {
            "principalType": "CLIENT",
            "principalId": "17784384-e37b-4c7a-8d09-3d8b1a7a72c7",
            "principalName": "DefaultTenantClient"
        },
        "createdDateTime": "2023-11-25 04:10 PM UTC",
        "updatedDateTime": "2023-11-25 04:10 PM UTC",
                  }
```


Where,

- samlEffectiveIdpInitiatedUrlWithEntityId: - This is the IDP initiated URL for the application (when IDSP acts as the SAML IDP), with the configured SAML Entity Id for the application.
- samlEffectiveIdpInitiatedUrlWithAppId: This is the IDP initiated URL to be used for the application (when IDSP acts as the SAML IDP) with the App Id for the application.
- samlEffectiveSpAcsUrlWithEntityId: This is the AssertionConsumerService URL to be configured with External IDP, when IDP Initiated flow is invoked by the External Identity Provider (when IDSP acts as the Service Provider) and the underlying application is identified by the Entity Id.
- samlEffectiveSpAcsUrlWithAppId: This is the AssertionConsumerService URL to be configured with External IDP, when IDP Initiated flow is invoked by the External Identity Provider (when IDSP acts as the Service Provider) and the underlying application is identified by the App Id.
- samlEffectiveSpAcsUrlWithAppName: This is the AssertionConsumerService URL to be configured with External IDP, when IDP Initiated flow is invoked by the External Identity Provider (when IDSP acts as the Service Provider) and the underlying application is identified by the App Name.

c. After creating an application in IDSP on behalf of the Service Provider, the IDSP's IDP metadata can be downloaded using the following endpoint:

```
POST  https://${SSP_FQDN}/default/saml/v1/metadata
```

If the Service Provider supports metadata import, the XML file can be directly provided. Otherwise, the necessary information must be manually provided.

d. Define the Authentication Policy in IDSP to authenticate the user. See Authentication Policy

4. Initiate the authentication flow. There are two ways to invoke the authentication flow:

– ***Identity Provider Initiated Flow*** : As an Identity Provider, you can initiate the authentication flow to access the application by using one of the following endpoints:

- `https://<SSP-Ingress-Host>/<Tenant-Name>/saml/v1/idp/login?entityId=<samlEntityId of the SP application registered>`
- `https://<SSP-Ingress-Host>/<Tenant-Name>/saml/v1/idp/login?appId=<appId of the SP application registered>`
- `https://<SSP-Ingress-Host>/<Tenant-Name>/saml/v1/idp/login?name=<name of the SP application registered>`

– ***Service Provider Initiated Flow*** : Perform the following step:

a. Navigate to **Domain Management** > **My Domain in SalesForce**
b. Edit the **Authentication Configuration** section and select the IDSP **Authentication Service**.
c. Click **Save**.
This step enables the IDSP login page when you inititate the authentication flow as a service Provider.

– Now, you can initiate the authentication flow as a service Provider to access the application by using the following login endpoint:

```
https://<SSP-Ingress-Host>/<Tenant-Name>/saml/v1/idp/login
```

. This step results in the following outcomes:

- The user is authenticated based on the factors defined in the policy.
- SAML response is sent back to the service provider.

Example: Integrating an OIDC Service Provider with IDSP

In the following scenario, Symantec Identity Security Platform operates as an Identity Provider, and Salesforce functions as the Service Provider. Configurations must be applied on both ends to establish the connection between the two parties and initiate the federated authentication flow. Therefore the integration setup between both the parties is described though the following steps.

Prerequisites: Ensure that you have met the following prerequisites before progressing with the SalesForce integration.

- A Public IP Machine or a Cloud Provider that has a Kubernetes cluster that is provisioned and IDSP deployed on it.
- An active SalesForce account to perform the integration.

NOTE

- The Standard OpenID Connect claims are mapped in the IDSP LDAPconfig using attributeMapping. Ensure to use the latest postman collection for creating LDAPconfig. Also, note that the user that has been used for SalesForce use case must have values for all the attributes in the LDAP ID Store as per mappings in the IDSP ldapconfig.
- Ensure that you use a CA Signed certificate to enable SSL to the ingress controller. This implies that all IDSP urls with https should use proper CA signed certificates. This is important as the *https://${SSP_FQDN}/default/oauth2/v1/token endpoint* gets called from the backend SalesForce server to trust the certificate chain.
- When using OpenID Connect protocol to integrate with IDSP, ensure that your app has the redirectURI, allowedOpenIDScopes, allowedGrantTypes, and allowedOperations configured. See API Security Using OAuth2 for more information.

This topic includes the following sections:

- Configuration a Service Provider in IDSP
- Configuration at SalesForce
- Updating the Client Config at IDSP
- Linking Accounts
- Logging into SalesForce Using IDSP Account

NOTE
The following example is indicative of SalesForce as a Service Provider in IDSP.

The steps outlined in this section detail the procedure of configuring SalesForce as an external service provider in IDSP.

Perform the following steps:

1. Download the Service Provider's metadata. This step is covered in this page, Downloading Service Provider's metadata step.
2. Create an application to register SalesForce in IDSP using the following API:

   `https://${SSP_FQDN}/default/admin/v1/Apps`

   NOTE

   - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
   - The default value for `<tenantName>` is "default".
   - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
     For example:
     `https://{{host}}/{{tenantName}}/admin/v1/Apps`
     should look similar to:
     `https://example.com/default/admin/v1/Apps`
   - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

   NOTE
   Ensure that the request payload has the following attributes configured:

   - "userInfoEndpointResponseFormat": "PLAIN_JSON"
   - "skipIssuerAudienceForIT": true

   The examples in this section describe the creation of an application using the OIDC protocol.
   Sample of an OIDC application

Request Payload:
The following example shows the request payload while creating a SalesForce application in IDSP using the ***OIDC protocol***.

```
{
    "status": "active",
    "name": "salesforce-app",
    "description": "Application for Salesforce integration",
    "clientType": "CONFIDENTIAL",
    "allowedOpenIDScopes": [
        "openid",
        "profile",
        "email"
    ],
    "redirectURIs": [
        "https://www.example.com"
    ],
    "allowedGrantTypes": [
        "authorization_code"
    ],
    "userInfoEndpointResponseFormat": "PLAIN_JSON",
    "skipIssuerAudienceForIT": true
}
```

Response Payload:
The following example shows the response of the SalesForce application created in IDSP using the ***OIDC protocol***.

```
{
    "status": "active",
    "name": "salesforce-app",
    "description": "Application for Salesforce integration",
    "clientType": "CONFIDENTIAL",
    "allowedOpenIDScopes": [
        "openid",
        "profile",
        "email"
    ],
    "redirectURIs": [
        "https://www.example.com"
    ],
    "allowedGrantTypes": [
        "authorization_code"
    ],
    "userInfoEndpointResponseFormat": "PLAIN_JSON",
    "skipIssuerAudienceForIT": true,
    "appId": "06c3cd12-20d2-4c89-8cc8-df429e3945e4",
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "006eddf2-ea8d-46b6-8c9b-2c355bf81aa2",
        "principalName": "defaulttenantclient"
    },
    "createdDateTime": "2020-06-23 04:28 AM UTC",
    "secret": "c380f534-144d-44ac-b9e2-47a1dcfac5e1",
    "clientId": "5770ce5b-7252-4b9b-a7bd-da624c891fca"
}
```

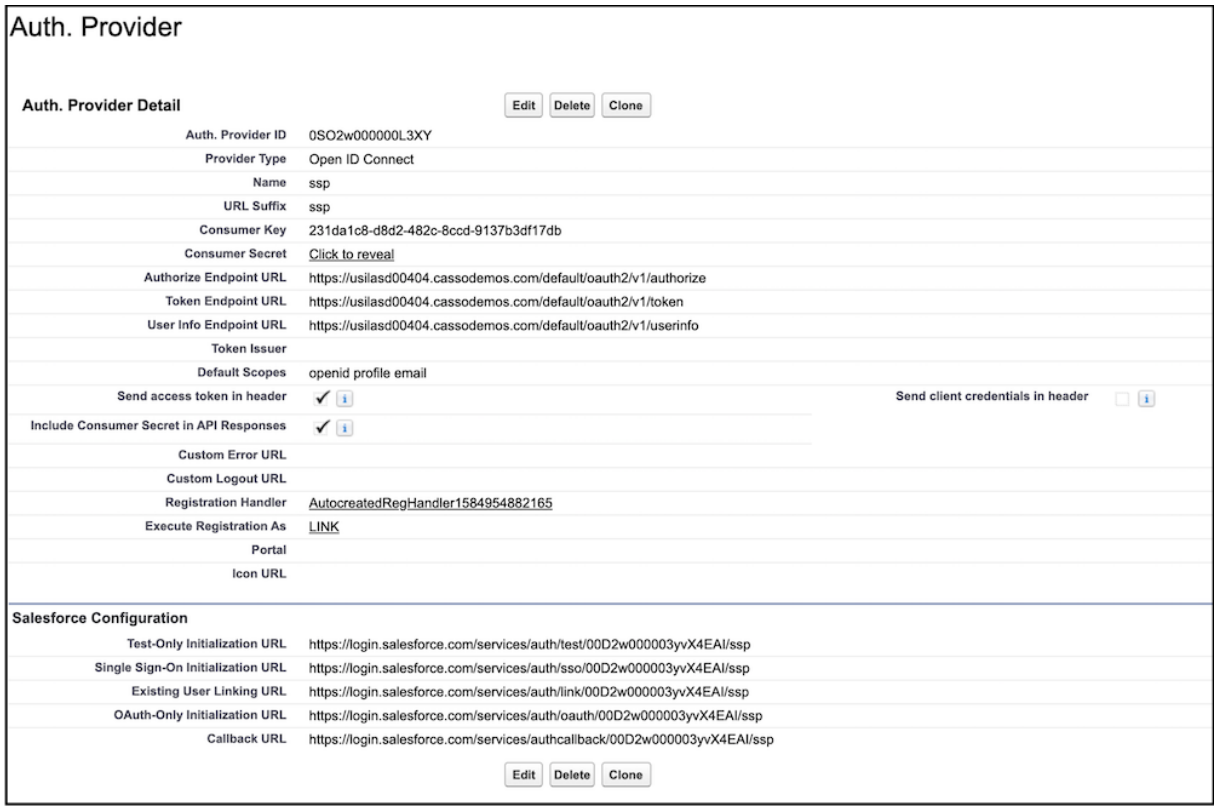
NOTE
For OIDC application

- Make a note of the *clientId* and *secret* attributes as these will be used to update *Consumer Key* and *Consumer Secret in the next section.*
- At this stage, you do not have a SalesForce callback url. You must update the redirect uri once SalesForce establishes an Auth Provider.

Configuration at SalesForce

After creating the application in IDSP, perform the following configurations at SalesForce:

1. Log into the SalesForce site.
2. Navigate to **Setup > Security Controls > Auth. Providers.**
3. Click **New** to add IDSP OIDC Client information and endpoints.
4. Configure the Consumer Key and the Consumer secret.
   NOTE
   Add the value of *clientId* in Consumer Key and the value of *secret in Consumer Secret* that you have obtained from IDSP.
5. Configure the following IDSP OIDC endpoints:
   NOTE
   Replace the ${SSP_FQDN} with appropriate hostname.
   a. *Authorize Endpoint URL*: https://${SSP_FQDN}/default/oauth2/v1/authorize
   b. *Token Endpoint URL*: https://${SSP_FQDN}/default/oauth2/v1/token
   c. *UserInfo Endpoint URL*: https://${SSP_FQDN}/default/oauth2/v1/userinfo
   d. (OPTIONAL) *Token Issuer*: https://${SSP_FQDN}/default/
6. Set the *Default Scopes* as *openid profile email.*
7. Click the **Auto Generate Registration handler link in the Registration Handler field.** This is required for SalesForce to generate the SSO initialization URL.
8. In the ***Run as User*** field, select ***Search*** and then select ***LoggedIn*** *user.*
9. Save the configuration.
   The following URLs are generated at this point:
   - Test-Only Initialization URL
   - Single Sign-On Initialization URL
   - Existing User Linking URL
   - Oauth-Only Initialization URL
   - Callback URL



10. Copy the *Callback, Existing User Linking* and *Single Sign-On* URLs.
11. Next step is to perform the SalesForce configuration at IDSP. See Updating the Client Config at IDSP section.

Updating the Client Config at IDSP

At this stage, you have obtained the SalesForce callback URL.

1. Perform a Put call to update the SalesForce application with the callback URL by using the following endpoint:

   ```
   https://${SSP_FQDN}/{{tenantName}}/admin/v1/Apps/{{clientAppId}}
   ```

   Example of Put a call to update the SalesForce callback URL

   ```
   {
       "status": "active",
       "name": "salesforce-app",
       "description": "Application for Salesforce integration",
       "clientType": "CONFIDENTIAL",
       "allowedOpenIDScopes": [
           "openid",
           "profile",
           "email"
       ],
       "redirectURIs": [
           "https://www.example.com",
           "https://<salesforce callback URL>"
       ],
       "allowedGrantTypes": [
           "authorization_code"
       ],
       "userInfoEndpointResponseFormat": "PLAIN_JSON",
   ```

```
        "skipIssuerAudienceForIT": true
    }
```

Linking Accounts

In this step, you are linking the IDSP User Id with the SalesForce account.

1. Open the *Existing User Linking* URL in a browser. This URL is provided to you when you save the Auth.Providers configuration at SalesForce.
   You are redirected to the IDSP Auth UI for authentication.
2. After a successful authentication, you will be displayed on the SalesForce login page.
3. Login with SalesForce User details.
   You may be prompted to authenticate with OTP, before proceeding with the *Link Accounts*. If you are prompted, authenticate with OTP.
4. Once you have authenticated successfully, click *Link Accounts*.
   You are redirected to the SalesForce site.
5. Navigate to *Setup* > *Manage Users* > *Users* > and then click the *Logged in user link.*
6. Navigate to the *Third-Party* Account.
   This displays the IDSP user id as one of the linked accounts along with the *Auth Provider Name.*

Logging into SalesForce Using IDSPAccount

1. Once your account linking is successful, open the *Single Sign-On* URL and the SalesForce Auth. Provider URL in a browser.
   You are redirected to the IDSP Auth UI login page.
2. Once you have authenticated successfully, you are redirected to the secure site.

Example: Integrating VCF as a SAML Service Provider with IDSP

Configure Symantec Identity Security Platform as a SAML Identity Provider for VMware Cloud Foundation (VCF).

In the following scenario, Symantec Identity Security Platform operates as the SAML Identity Provider (IDP), and VCF functions as the Service Provider (SP). Configurations must be applied on both ends to establish trust and initiate the federated authentication flow.

The integration setup involves the following high-level steps:

1. Gather VCF Service Provider metadata.
2. Create a SAML application in the IDSP Admin Console on behalf of VCF.
3. Download IDSP IDP metadata and provide it to VCF.
4. Define an Authentication Policy in IDSP.
5. Register IDSP as an Identity Provider in VCF, configure user and group provisioning, and test login.

Prerequisites

Ensure that you have met the following prerequisites before progressing with the VCF integration.

- A deployment of IDSP that is accessible to the VCF infrastructure.
- VCF 9.1 or later is installed or upgraded.

Step 1: Gather VCF Service Provider Metadata

Before creating the SAML application in IDSP, you need to obtain the Service Provider metadata from VCF. This metadata contains the **Entity ID** and the **Assertion Consumer Service (ACS) URL** that are required to configure the SAML application.

Log in to the VCF console and follow the steps under the **Choose an identity provider** and **Configure the identity provider** sections in the VCF documentation: Configure Broadcom IDSP as an Identity Provider using SAML.

In the **Service Provider Registration** screen, note down or download the following values. You will use these values when creating the SAML application in IDSP:

- **Entity ID/Metadata URL** -- This value will be used as the **Entity ID** in the IDSP SAML application.
- **Assertion Consumer Service URL** -- This value will be used as the **ACS URL** in the IDSP SAML application.

Step 2: Create a SAML Application in the IDSP Admin Console

In this step, you create a SAML application in the IDSP Admin Console to register VCF as a Service Provider.

1. Log in to the IDSP Admin Console.
2. Navigate to **Applications**.
3. Click **Add Application** to create a new application.
4. Provide a **Name** for the application (for example, "VCF SAML Application") and an optional **Description**.
5. Navigate to the **SAML** tab and select **Enabled**. The SAML attributes are displayed.
6. Configure the following SAML attributes using the VCF Service Provider metadata obtained in Step 1:

Attribute	Description / Value
Entity ID	Enter the Entity ID obtained from the VCF Service Provider Registration screen. • In the Service Provider Initiated flow, the Entity ID should match the Issuer in t • In the IDP Initiated flow, if the Entity ID is provided in the request parameter, it point to the configured application.
ACS URL	Enter the Assertion Consumer Service URL obtained from the VCF Service Provi where IDSP sends the SAML response.
NameID Format	Enter the NameID format that is supported for the Subject. Select a format that ma VCF. Supported formats are: **emailAddress**, **unspecified**, **X509SubjectName**.
Verify Request Signature	(Optional) Enable this attribute to verify the signature of the SAML Authentication If enabled, select a **Verification Certificate**: • Click **Select Certificate** to search for a certificate using alias. • Use filters (such as Attributes, Operator, Value) to narrow your search. You ca • If a certificate is not available, click **Import** to upload the VCF signing certificat
Encrypt Assertion	(Optional) Enable this to encrypt the SAML assertion sent by IDSP to VCF. If enabled, configure the following: • **Encryption Certificate**: Select or import the VCF encryption certificate. – Click **Select Certificate** to search for a certificate by its alias. – If no certificate is available, click **Import** to upload the VCF encryption certi • **Encryption Algorithm**: Select the algorithm used to encrypt the SAML assert • **Key Transport Algorithm**: Select the algorithm for securely delivering the enc
Sign Response	Enable this attribute to sign the SAML response sent by IDSP. This verifies the int If enabled, configure the following: • **Sign Response Key Alias**: Click **Select Key** to select the signing key. If no k one. • **Sign Response Algorithm**: Select the algorithm to generate the signature. • **Sign Response Canonicalization Algorithm**: Select the canonicalization alg

Sign Assertion	(Optional) Enable this attribute to sign only the SAML assertion within the SAML r If enabled, configure the following: • **Sign Assertion Key Alias**: Click **Select Key** to select the signing key. If no ke one. • **Sign Assertion Algorithm**: Select the algorithm type to sign the assertion: • **Sign Assertion Canonicalization Algorithm**: Select the canonicalization alg
Sign with Key Information	(Optional) Enable this attribute to include the KeyInfo element in the SAML signat
Relay State Mapping	(Optional) Define the mapping between the relay state that is provided during the relative URL of the application, where the flow concludes. 1. Click **+Add**. 2. Enter the **Key (regex)** using regular expressions, and the associated **Value** th or action that should be triggered based on the matched state. Mapped relay state values are defined using regex, necessitating the escaping of
Custom Claims	The **Custom Claims** attribute allows you to configure custom SAML claims for an SAML claims in the Identity and/or Access tokens on behalf of this application as 1. Click **+Add**. The **Name** and **Value** fields appear. 2. Enter custom claims in the Name and Value field.
User Assertion Subject Attribute Mapping Name	(Optional) Enter the name of the logical ID store attribute whose physical ID store Subject of the SAML assertion. This logical attribute must appear as one of the m of the ID store config resource.

7. Click **Save Application**. The application is saved to the Application dashboard.

Step 3: Download IDSP IDP Metadata

After creating the SAML application in IDSP, download the IDP metadata to provide to VCF.

1. In the IDSP Admin Console, navigate to the SAML application you created for VCF.
2. In the **SAML** tab, locate the **Integration Setup** section.
3. Click **View Setup**. The Integration Setup page is displayed with the IDP metadata details.
4. Download or copy the IDP metadata XML. This metadata contains the following information required by VCF:
   - IDP Entity ID
   - Single Sign-On Service URL
   - Signing certificate

These necessary information must be captured and provided to VCF.

Step 4: Define an Authentication Policy in IDSP

Configure an authentication policy in IDSP to define how users are authenticated when accessing VCF through the SAML flow.

For more information, see Authentication Policy.

Step 5: Register IDSP as an Identity Provider in VCF and Configure Provisioning

In this step, you register IDSP as the SAML Identity Provider in the VCF console using the IDP metadata obtained in Step 3. You also configure user and group provisioning, and optionally test the login.

Follow the complete procedure documented in the VCF documentation: Configure Broadcom IDSP as an Identity Provider using SAML.

Testing the Authentication Flow

After completing the integration, you can test the flow based on the instructions shared in the Test Login section in the VCF documentation.

Important Notes on Backup and Role Assignment

For important information on backing up your VCF SSO configuration and assigning VCF roles, see the **Important Notes on Backup** and **Important Notes on Role Assignment** sections in the VCF documentation: Configure Broadcom IDSP as an Identity Provider using SAML.

Example-Integrating VCF-OIDC Service Provider with IDSP

Configure Symantec Identity Security Platform as an OIDC Identity Provider for VMware Cloud Foundation (VCF).

In the following scenario, Symantec Identity Security Platform operates as the OIDC Identity Provider (IDP), and VCF functions as the Service Provider (SP). Configurations must be applied on both ends to establish trust and initiate the federated authentication flow.

The integration setup involves the following high-level steps:

- Obtain VCF Service Provider OIDC configuration details.
- Create an OIDC application in the IDSP Admin Console on behalf of VCF.
- Configure OIDC claims and scopes.
- Define an Authentication Policy in IDSP.
- Register IDSP as an OIDC Identity Provider in VCF.

Prerequisites

Ensure that you have met the following prerequisites before progressing with the VCF integration.

- A deployment of IDSP that is accessible to the VCF infrastructure.
- VCF 9.1 or later is installed or upgraded.

> **NOTE**
> Ensure that you use a CA-signed certificate to enable SSL in IDSP see TLS Certificate Configuration in IDSP.

Step 1: Obtain VCF Service Provider OIDC Configuration Details

Before creating the OIDC application in IDSP, you need to obtain the OIDC configuration details from VCF. These details will be used to configure the OIDC application in IDSP.

Refer to the VCF documentation for detailed steps to obtain the required configuration details.

The following values are typically required from VCF:

- **Callback URL / Redirect URI** -- This is the URL to which IDSPredirects users after authentication. This value is used as the redirectURI in the IDSP OIDC application.
- **Client Type** -- Verify that VCF requires a CONFIDENTIAL or PUBLIC client type. For most enterprise deployments, CONFIDENTIAL is recommended.
- **Required Scopes** -- Document any custom or specific scopes that VCF expects from the OIDC provider. Standard scopes typically include: **openid**, **profile**, **email**.

TIP: Save this information in a secure location for reference during the IDSP configuration step.

Step 2: Create an OIDC Application in the IDSP Admin Console

In this step, you create an OIDC application in the IDSP Admin Console to register VCF as a Service Provider.

1. Log in to the IDSP Admin Console.
2. Navigate to **Applications**, select Type as **OAuth/OpenID Web App**.
3. Click **Add Application** to create a new application.
4. Provide a Name for the application (for example, "VCF OIDC Application") and an optional Description.
5. Select the **OIDC** tab and configure the following OIDC attributes using the VCF configuration details obtained in Step 1.
   The following table describes the key OIDC attributes to configure:

Attribute	Description / Value
Client Type	Select the appropriate client type based on VCF requirements: CONFIDENTIAL or PUBLIC. For most enterprise deployments, select CONFIDENTIAL, as the client credentials will be used for secure communication with VCF.
Redirect URIs	Enter the callback URL / Redirect URI obtained from VCF in Step 1. NOTE: You can update this value later if VCF provides additional redirect URIs. Multiple redirect URIs are supported.
Allowed OpenID Scopes	Select the scopes that VCF requests. Standard OIDC scopes include: • openid (required for OIDC) • profile (provides user profile information) • email (provides user email address) NOTE: Coordinate with the VCF administrator to confirm which scopes are required for the specific VCF deployment.
Allowed Grant Types	Select "authorization_code" as the grant type for the authorization code flow (recommended for web applications like VCF).
User Info Endpoint Response Format	Select "PLAIN_JSON" to return user information in plain JSON format. This is the recommended setting for compatibility with most OIDC clients.

6. Click **Save Application**. The application is saved to the Application dashboard.

After saving, note the following values that are generated by IDSP. These values are needed to configure VCF:

- Client ID
- Client Secret
- IDSP OIDC Endpoints:
  - Authorize Endpoint: https://${SSP_FQDN}/default/oauth2/v1/authorize
  - Token Endpoint: https://${SSP_FQDN}/default/oauth2/v1/token
  - UserInfo Endpoint: https://${SSP_FQDN}/default/oauth2/v1/userinfo

Step 3: Configure OpenID Connect Claims and Scopes

To ensure VCF receives the appropriate user attributes during authentication, configure the claims mapping in IDSP.

1. In the IDSP Admin Console, open the OIDC application created for VCF in Step 2.
2. Navigate to the **Custom Claims** section.

3. Map the required user attributes. The following are common claims that VCF typically requires:

Claim Name	Value (using IDSP Dynamic Variables)
sub (subject)	Auto-populated with user's unique identifier
email	${idProfile.email}
name	${idProfile.displayName}
family_name	${idProfile.lastname}
given_name	${idProfile.firstname}
phone_number	${idProfile.telephone} (if applicable)
groups (custom)	${idProfile.groups} (if VCF uses group-based authorization)

NOTE
The exact claim names and values depend on your LDAP schema and VCF attribute requirements. Coordinate with the VCF administrator to ensure the claim mappings align with their expected attributes.

4. After configuring claims, save the application.

Step 4: Define an Authentication Policy in IDSP

Configure an authentication policy in IDSP to define how users are authenticated when accessing VCF through the OIDC flow.

For detailed information on creating and managing authentication policies, see Authentication Policy documentation.

Step 5: Register IDSP as an OIDC Identity Provider in VCF

After creating the OIDC application in IDSP, you need to register IDSP as the OIDC Identity Provider in VCF.

Refer to the VCF documentation for detailed steps to configure IDSP as an OIDC identity provider:

When configuring VCF, provide the following IDSP OIDC endpoints and credentials obtained in Step 2:

- Authorize Endpoint URL: https://${SSP_FQDN}/default/oauth2/v1/authorize
- Token Endpoint URL: https://${SSP_FQDN}/default/oauth2/v1/token
- UserInfo Endpoint URL: https://${SSP_FQDN}/default/oauth2/v1/userinfo
- Client ID: [Use the Client ID generated in Step 2]
- Client Secret: [Use the Client Secret generated in Step 2]
- Scopes: openid profile email (or scopes specified in Step 2)

Ensure that the claim mappings configured in Step 3 align with VCF's expected user attributes.

Testing the OIDC Authentication Flow

After completing the integration, you can test the flow based on the instructions shared in the Test Login section in the VCF documentation.

Important Notes on Backup and Role Assignment

For important information on backing up your VCF SSO configuration and assigning VCF roles, see the "Important Notes on Backup" and "Important Notes on Role Assignment" sections in the VCF documentation.

Managing Application Groups

The Application Group APIs provide administrative functionalities for managing logical groupings of applications. Using the urn:iam:t.resourcegroups scope, administrators can perform various operations via RESTful endpoints.

Application Group APIs enable administrators to efficiently create, retrieve, update, and delete application groups, which are crucial for organizing resources and enforcing authentication policies within a secure environment. The APIs facilitate streamlined management of application access by defining clear group memberships.

NOTE
Application group is supported in Authentication policy, Risk policy, and IDP Discovery policy. It is not supported in Authorization policy.

Application Group API scopes

- "urn:iam:t.resourcegroups" - This scope grants administrative privileges to manage application groups.

Application Group API

API Description	Operation Type	Endpoint URLs
Check existence of the application group	HEAD	https://${SSP_FQDN}/<tenantName>/admin/
Create an application group	POST	https://${SSP_FQDN}/<tenantName>/admin/
Update an application group	PUT	https://${SSP_FQDN}/<tenantName>/admin/
Adding or removing an application from an existing group	PATCH	https://${SSP_FQDN}/<tenantName>/admin/
Get all the application groups	GET	https://${SSP_FQDN}/<tenantName>/admin/
Get application group based on the application Id	GET	https://{{sspHost}}/{{apiPathTenant}}/admin/v
Delete an application group	DELETE	https://${SSP_FQDN}/<tenantName>/admin/ To forcefully delete an application, including t

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace {text} with the appropriate values in all the endpoints shown in the following table.
  For example:
  `https://${SSP_FQDN}/<tenantName>/admin/v1/ResourceGroups`
  should look similar to:
  `https://example.com/default/admin/v1/ResourceGroups`
- To know more about the Symantec Identity Security Platform Rest API syntax and scopes overview, refer to the Rest APIs and Associated Privileges section.

Request and Response Samples

POST operation to create an Application Group

Using the following POST operation you can create an application group under the Resource Group in IDSP:

```
POST https://${SSP_FQDN}/<tenantName>/admin/v1/ResourceGroups
```

API Request:

```
{
```

```
"name": "Blue Group",
"description": "Includes employees with blue tags",
"resources": [
"urn:iam:res:app"
]
}
```

Where,

- **name** - Specify the name of the application group
- **description** - Specify a description to identify the application group
- **resources** - This is the type of the resource that the resource group contains. In this example the resource type is **app**.

API Response: The response creates an app group and gives a group Id to it.

```
{
"adminscope.criteria": [],
"adminscope.managedby": [],
"groupId": "26acc07c-d4fb-45fd-ae4d-0720f911ae9a",
"name": "Blue Group",
"description": "Includes employees with blue tags",
"resources": [
"urn:iam:res:app"
],
"createdDateTime": "2025-09-02 10:05 AM UTC",
"updatedDateTime": "2025-09-02 10:05 AM UTC",
"createdBy": {
"principalType": "CLIENT",
"principalId": "3f3b8c8d-c9dc-4684-9228-865beeb81674",
"principalName": "DefaultTenantClient"
},
"updatedBy": {
"principalType": "CLIENT",
"principalId": "3f3b8c8d-c9dc-4684-9228-865beeb81674",
"principalName": "DefaultTenantClient"
}
}
```

PUT operation to update an Application Group details

Using the following PUT operation you can update an application group under the Resource Group in IDSP:

```
PUT https://${SSP_FQDN}/<tenantName>/admin/v1/ResourceGroups/{{ApplicationGroupId}}
```

API Request:

```
{
"name":
"GroupCreatedByClient",

"description":
"Group created by client",

"adminscope.criteria": [

"organization=banking",
```

```
"service=sports"

]

}
```

Where,

- **name** - Change the name of the application group
- **description** - Change the description of the application group
- **adminscope.criteria** - This is an array of tags defined for admin to manage resource groups and group members.

API Response:

```
{
"adminscope.criteria": [

"organization=banking",

"service=sports"

],

"adminscope.managedby": [],

"groupId":
"4b812cf8-93d4-4ade-91a0-d9763a7a28f1",

"name":
"GroupCreatedByClient",

"description":
"Group created by client",

"resources": [

"urn:iam:res:idppolicy",

"urn:iam:res:app"

],

"membersResults": {

"memberCount": {

  "urn:iam:res:app":
1

}

},

"createdDateTime":
```

```
"2025-06-25 03:39 PM UTC",

"updatedDateTime":
"2025-09-05 02:52 PM UTC",

"createdBy": {

"principalType":
"CLIENT",

"principalId":
"894cd809-4982-4968-b4cc-cdfeefba5ca5",

"principalName":
"RISTestApp"

},

"updatedBy": {

"principalType":
"CLIENT",

"principalId":
"cbfa35be-594c-49ac-852d-c5aa1c45f26d",

"principalName":
"DefaultTenantClient"

}

}
```

PATCH operation to either add or remove an application in the Application Group

Using the following PATCH operation you can either add an application in the application group or remove an application from the application group:

```
PATCH https://${SSP_FQDN}/<tenantName>/admin/v1/ResourceGroups/{{ApplicationGroupId}}
```

API Request: This example shows an add operation. To add an application to the existing app group, you can either specify the application Id or the application name in the payload as shown in the example.

```
{
"operations": [
{
//Valid action is ADD or REMOVE
"action": "ADD",
"members": [
//Add new resource by id
{
  "type": "urn:iam:res:app",
  "id": "78a9853b-4668-47f0-bd1e-5589a76c4fb4"
},
```

```
//Add new resource by name
{
  "type": "urn:iam:res:app",
  "name": "App2"
}
]
}
]
}
```

Where, you can update either the application Id or the application name:

- "type": "urn:iam:res:app", "id": "78a9853b-4668-47f0-bd1e-5589a76c4fb4" - Add or remove an application by specifying the application Id.
- "type": "urn:iam:res:app", "name": "test" - Add or remove an application by specifying the name of the application.

> **NOTE**
> This API's response confirms the total memberCount after adding a new application to the group, but it does not list individual applications. To retrieve details of applications within the group, use the GET https://${SSP_FQDN}/<tenantName>/admin/v1/Apps API and include the application group ID as a **filter query** parameter. This API is covered in Managing Applications.

API Response: The response shows the increase in the memberCount "urn:iam:res:app" field.


```
{
"adminscope.criteria": [],
"adminscope.managedby": [],
"groupId": "21e9cc82-8367-4bb7-8dc6-fc55c9d2a798",
"name": "Apple Group",
"description": "",
"resources": [
"urn:iam:res:app"
],
"membersResults": {
"memberCount": {
"urn:iam:res:app": 2
}
},
"createdDateTime": "2025-08-21 05:09 AM UTC",
"updatedDateTime": "2025-09-02 09:18 AM UTC",
"createdBy": {
"principalType": "USER",
"principalId": "940b4226-e553-48c9-804f-dc7fc589c99e",
"principalName": "malbail"
},
"updatedBy": {
"principalType": "USER",
"principalId": "940b4226-e553-48c9-804f-dc7fc589c99e",
"principalName": "malbail"
}
}
```

GET operation to retrieve an Application Group by its Id

Using the following GET operation you can retrieve an application group by specifying its Id under the Resource Group:

```
GET https://${SSP_FQDN}/<tenantName>/admin/v1/ResourceGroups/{{ApplicationGroupId}}
```

API Response: This is a sample response of an application group, assuming the group Id in the request was "21e9cc82-8367-4bb7-8dc6-fc55c9d2a798".


```
{
"adminscope.criteria": [],
"adminscope.managedby": [],
"groupId": "21e9cc82-8367-4bb7-8dc6-fc55c9d2a798",
"name": "Apple Group",
"description": "",
"resources": [
"urn:iam:res:app"
],
"membersResults": {
"memberCount": {
"urn:iam:res:app": 1
}
},
"createdDateTime": "2025-08-21 05:09 AM UTC",
"updatedDateTime": "2025-08-21 05:33 AM UTC",
"createdBy": {
"principalType": "USER",
"principalId": "940b4226-e553-48c9-804f-dc7fc589c99e",
"principalName": "malbail"
},
"updatedBy": {
"principalType": "USER",
"principalId": "940b4226-e553-48c9-804f-dc7fc589c99e",
"principalName": "malbail"
}
}
```


GET operation to retrieve all the Application Groups that are created within the resource group - "urn:iam:res:app"

Using the following GET operation you can retrieve all the application groups that are created under the Resource Group - "urn:iam:res:app":

```
GET https://${SSP_FQDN}/<tenantName>/admin/v1/ResourceGroups?filter=resources eq urn:iam:res:app
```

API Response: This is a sample response of all the application group that are created under the Resource Group - "urn:iam:res:app".


```
[
{
"adminscope.criteria": [],
"adminscope.managedby": [],
"groupId": "21e9cc82-8367-4bb7-8dc6-fc55c9d2a798",
"name": "Apple Group",
"description": "",
"resources": [
```

```
"urn:iam:res:app"
],
"membersResults": {
"memberCount": {
"urn:iam:res:app": 2
}
},
"createdDateTime": "2025-08-21 05:09 AM UTC",
"updatedDateTime": "2025-09-02 09:18 AM UTC",
"createdBy": {
"principalType": "USER",
"principalId": "940b4226-e553-48c9-804f-dc7fc589c99e",
"principalName": "malbail"
},
"updatedBy": {
"principalType": "USER",
"principalId": "940b4226-e553-48c9-804f-dc7fc589c99e",
"principalName": "malbail"
}
},
{
"adminscope.criteria": [],
"adminscope.managedby": [],
"groupId": "26acc07c-d4fb-45fd-ae4d-0720f911ae9a",
"name": "Blue Group",
"description": "Includes employees with blue tags",
"resources": [
"urn:iam:res:app"
],
"membersResults": {
"memberCount": {}
},
"createdDateTime": "2025-09-02 10:05 AM UTC",
"updatedDateTime": "2025-09-02 10:05 AM UTC",
"createdBy": {
"principalType": "CLIENT",
"principalId": "3f3b8c8d-c9dc-4684-9228-865beeb81674",
"principalName": "DefaultTenantClient"
},
"updatedBy": {
"principalType": "CLIENT",
"principalId": "3f3b8c8d-c9dc-4684-9228-865beeb81674",
"principalName": "DefaultTenantClient"
}
},
{
"adminscope.criteria": [],
"adminscope.managedby": [],
"groupId": "bafa3063-f59d-417d-9ac0-ee8bce8dab30",
"name": "orange group",
"description": "",
"resources": [
"urn:iam:res:app"
```

```
],
"membersResults": {
"memberCount": {
"urn:iam:res:app": 1
}
},
"createdDateTime": "2025-08-25 08:45 AM UTC",
"updatedDateTime": "2025-08-25 08:45 AM UTC",
"createdBy": {
"principalType": "USER",
"principalId": "940b4226-e553-48c9-804f-dc7fc589c99e",
"principalName": "malbail"
},
"updatedBy": {
"principalType": "USER",
"principalId": "940b4226-e553-48c9-804f-dc7fc589c99e",
"principalName": "malbail"
}
},
{
"adminscope.criteria": [],
"adminscope.managedby": [],
"groupId": "c4c5ee74-bdb3-44b9-92d1-de28cfecc45b",
"name": "Red Group",
"description": "Includes employees with red tags",
"resources": [
"urn:iam:res:app"
],
"membersResults": {
"memberCount": {}
},
"createdDateTime": "2025-09-01 05:53 AM UTC",
"updatedDateTime": "2025-09-01 05:53 AM UTC",
"createdBy": {
"principalType": "CLIENT",
"principalId": "3f3b8c8d-c9dc-4684-9228-865beeb81674",
"principalName": "DefaultTenantClient"
},
"updatedBy": {
"principalType": "CLIENT",
"principalId": "3f3b8c8d-c9dc-4684-9228-865beeb81674",
"principalName": "DefaultTenantClient"
}
}
]
```


DELETE operation to remove an Application Group by specifying its Id

Using the following DELETE operation you can remove an application group by specifying the Id in the API path:

```
DELETE https://${SSP_FQDN}/<tenantName>/admin/v1/ResourceGroups/{{ApplicationGroupId}}
```

API Response:

```
Shows a "204 No Content" for a successful DELETE operation.
```

For sample request and response payloads of this URL, refer to App Groups API Swagger documentation.

Deleting Application Groups Referenced by Other IDSP Objects

If a tenant's application group is being deleted and it is currently referenced by other IDSP objects, the `DELETE` API performs a dependency check. If a conflict is detected, and other objects are using the same application group, the API prevents the delete operation, and returns error code 8000143.

This allows the administrator to review where the application group is being used and manually remove the reference from each object. For example, the application group might be used in configurations such as Authentication (AuthN) policy, Authorization (AuthZ) Policy or Discovery Policy. **The delete operation is only allowed once all such references are removed.**

NOTE

- The HTTP response returned in case of a conflict (resource being reused by other IDSP objects) during deletion of an application is **409**.
- When attempting to delete an identity provider that is currently referenced by other IDSP objects, the system returns error code 8000143 to prevent the deletion.
- The DELETE operation must contain the "**force=true**" query parameter to delete references of the IDSP objects.
- A Helper API is available to retrieve the list of dependent objects in case of a conflict, see ReferencingObjectsHelper API.

The following DELETE operation is where the reference list gets populated in cases where there are conflicts (resource being reused by other IDSP objects):

```
DELETE https://${SSP_FQDN}/{{tenantName}}/admin/v1/Apps/{{appId}}
```

The following sample response illustrates how this list is displayed.

Sample Response:

```
{
"errorCode": "8000037",

"errorMessage": "Application associated with one or more policies cannot be deleted"
}
```

Reference Links to:

- For more information on Creating Application using the Admin Console, see here.
- Application API Error Codes

Managing Policies

Policy management within Symantec Identity Security Platform encompasses the management of Authentication (AuthN) Policies, Authorization Policies, Discovery Policies, Risk Polices, and Account Activation, Unlock, and Password Reset Policies.

Authentication Policy

The Authentication Policy acts as a gatekeeper, defining how users are authenticated and under what conditions access is granted. The policy ensures that only authorized users with the required authentication factors can access specific resources within the system. The policy can be fine-tuned to match the security and access control needs of the organization.

To know more about creating Authentication Policies, including essential attributes like rules and conditions, and using the Authentication Policy APIs to perform CRUD operations on them, see Authentication Policy.

Authorization Policy and Roles

Authorization Policies govern access to resources, specifying who can do what, while Admin Roles determine which users or entities have specific responsibilities and permissions within IDSP. Together, they establish a robust system of access control, ensuring that users can only access the resources and perform the actions to which they are authorized. In IDSP, the process of authorization is managed through administrative roles which are implemented through policies.

To know more about Authorization Admin Roles, including the scopes that are associated with each role and Authorization APIs, see Managing Authorization.

Discovery Policy

The IDP Discovery Policy streamlines the authentication process by ensuring that users are directed to the correct Identity Provider (IDP) based on the defined rules and conditions, ultimately enhancing security and user convenience.

To know more about creating an IDP Discovery Policy, including essential attributes like rules and conditions, and using the Discovery Policy APIs to perform CRUD operations on them, see Identity Provider (IDP) Discovery Policy.

Risk Policy

The Risk Policy evaluates the risk profile for a user at an application level. The Risk Policy assesses user risk in an application based on rules and contextual evaluation, without affecting user experience.

To know more about creating a Risk Policy, including essential attributes like rules and conditions, and using the Risk Policy APIs to perform CRUD operations on them, see Risk Policy.

Account Activation Policy

The Account Activation policy allows you to set which credential types users can use to authenticate when activating an account.

To know more about creating a Risk Policy, including essential attributes like rules and conditions, and using the Risk Policy APIs to perform CRUD operations on them, see Activate Account Policy.

Account Unlock Policy

The Account Unlock policy allows you to set which credential types users can use to authenticate when unlocking an account.

To know more about creating a Risk Policy, including essential attributes like rules and conditions, and using the Risk Policy APIs to perform CRUD operations on them, see Unlock Account Policy.

Reset Password Policy

The Reset Password policy allows you to set which credential types users can use to authenticate when resetting a password.

To know more about the Reset Password Policy, including essential attributes like rules and conditions, and using the Risk Policy APIs to perform CRUD operations on them, see Reset Password Policy.

Authentication Policy

Symantec Identity Security Platform Authentication (AuthN) Policy Management

The Authentication Policy, also known as the AuthN policy in IDSP operates by defining a set of rules and conditions that determine how user authentication is handled within the solution.

This topic includes the following sections:

- Working of the Authentication Policy in IDSP
- Authentication Policy Overview
- Policy Attributes
- Rule Conditions
  - action (includes "authenticate" and "access"actions)
  - principal
    - User
    - User Expressions
    - Groups
  - ipAddress
    - Example of IPv6 format support in IP Address condition
  - channel
  - riskThreshold
  - idpName
  - acr
  - triggeredRiskRule
  - country
  - context

How Authentication Policy Works in IDSP

How it typically works:

1. **Policy Configuration**: An administrator configures the Authentication Policy by defining various parameters and conditions. These parameters can include the authentication methods allowed, authentication factors required, and any additional security settings.
2. **User Authentication**: When a user attempts to log in or access a protected resource, IDSP evaluates the Authentication Policy associated with that user or resource.
3. **Policy Evaluation**: The Authentication Policy is evaluated based on the user's identity, the requested resource, and other contextual factors. It checks whether the user meets the specified conditions for authentication.
4. **Factor Authentication**: If the policy requires multiple authentication factors (e.g., password, SMSOTP, fingerprint), the user is prompted to provide them accordingly.
5. **Authentication Verification**: IDSP verifies the provided credentials against the configured factors and security mechanisms.
6. **Access Grant**: If the user successfully satisfies the Authentication Policy's conditions, they are granted access to the requested resource or are logged in.
7. **Access Denial**: If the user fails to meet the conditions specified in the Authentication Policy, access is denied, and the user may need to retry authentication or follow alternative procedures.
8. **Logging and Auditing**: IDSP often logs authentication events and policy evaluations for security and auditing purposes.

Authentication Policy Overview

An administrator can configure an application's authentication policies and can perform CRUD operations on it. Each policy comprises of rules, and each rule comprises a set of "**conditions**" and "**result**", which helps in determining how the authentication flow should proceed when a user attempts to authenticate. Based on the inputs provided by the user, for example; ipAddress, subject, and others, the policy decides whether single or multi-factor authentication should

be imposed upon the user. Eventually, the policy lays out an authentication flow for the user which is governed by the conditions that are specified within the rules of the policy. Different conditions can be configured in the rules such as action, principal, ipAddress, channel, and riskThreshold.

NOTE
To know more about Authorization Policy, see Managing Authorization.

Let us look at the details of a Policy API attributes such as rules, conditions, and result.

Authentication Policy Attributes

The details of the policy attributes are described here:

- **policy name** – The name displays the title of the policy
- **priority** - (Optional) Set the priority of the policy ranging from 1 (highest) to 1,000,000 (lowest). Decimals can be used for finer control. For more information on how priority influences an Authentication Policy result, see Setting priority in Authentication Policy and Rule.
- **description** – The description details a brief about the policy
- **status** – The status displays whether a policy is **active** or **inactive**
- **matchAnyApp** - This is a boolean attribute.
  - When ***matchAnyApp*** is set to "***true***", the policy is applied to all the applications.
  - When ***matchAnyApp*** is set to "***false***", the policy is applied to specific apps that are mentioned under the "apps" section in this policy.
- **apps** – The property supports the following values:
  - "**id**" - Id of the application.
  - "**name**" - Name of the application.
  - "**type**" - (optional)- supports two values, "app" (default) and "appgroup"

  It is mandatory to specify either the application id ("id") or the application name ("name"). The rules defined in the policy gets applied to the configured application(s).

  NOTE
  The application group is only supported in Authentication Policy, Risk Policy and IDP Discovery Policy. App group is not supported in Authorization Policy.
- **rules** - Rules include **conditions** and **result**. You can specify a single condition or multiple conditions to a rule. If the specified conditions in the rule are met, then the corresponding result is applied. The supported condition categories are: **action**, **principal** (group or user), **ipAddress**, **channel, riskThreshold**, **idpName**, **acr**, **triggeredRiskRule**, **country**, and, **context**.
  - **priority** - (Optional) This property allows you to set the priority of the rule ranging from 1 (highest) to 1,000,000 (lowest). Decimals can be used for finer control. For more information on how priority influences an Authentication Policy result, see Setting priority in Authentication Policy and Rule.
  - **conditions**: This property allows you to specify various conditions that must be evaluated for an authentication request. Conditions express the relationship of an attribute to a value. For example, if you specify "action" as one of the attributes, then the values could be "login".
    - Support for "**NOT**" operator- The "NOT" operator is supported in the rule conditions; ipAddress, actions, channel, principal (group and users), and idpName. This operator is not supported in rule condition; ACR and Riskthreshold.
      Example showing the request and response payload of "NOT" operator used in an Authentication Policy.
      Request payload:

```
{
    "policyName": "AdminLoginPolicy",
    "priority": "1",
    "description": "Allow with password obligation",
    "status": "active",
```

```
    "matchAnyApp": false,
    "apps": [
        {
            "name": "AdminConsole",
            "id": "$TENANT_ADMINCONSOLE"
        },
        {
            "name": "DefaultTenantClient",
            "id": "fd43c762-cde8-447f-ad45-cb4e68b46b99"
        }
    ],
    "rules": [
        {
            "priority": "1",
            "conditions": {
                "principal": {
                    "user": {
                        "operator": "not",
                        "value": [
                            "edunn"
                        ]
                    }
                },
                "ipAddress": {
                    "operator": "not",
                    "value": [
                        "10.0.0.0/24"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "FIDO:2",
                    "PASSWORD:1",
                    "SMSOTP:2",
                    "SECURITYKEY:2",
                    "EMAILOTP:2"
                ],
                "msg": "Sample authentication policy"
            }
        }
    ]
}
```

Response payload:

```
{
    "policyId": "d15646a5-8b30-408f-b2e9-a29fe28dda04",
    "policyName": "AdminLoginPolicy",
    "description": "Allow with password obligation",
    "status": "active",
    "matchAnyApp": false,
    "apps": [
```

```
            {
                "name": "DefaultTenantClient",
                "id": "fd43c762-cde8-447f-ad45-cb4e68b46b99"
            },
            {
                "name": "AdminConsole",
                "id": "$TENANT_ADMINCONSOLE"
            }
        ],
        "rules": [
            {
                "id": "92934894-e9fc-47e2-86b9-ff640e2a7afb",
                "conditions": {
                    "principal": {
                        "user": {
                            "operator": "not",
                            "value": [
                                "edunn"
                            ]
                        }
                    },
                    "ipAddress": {
                        "operator": "not",
                        "value": [
                            "10.0.0.0/24"
                        ]
                    }
                },
                "result": {
                    "effect": "allow",
                    "reAuthenticate": "false",
                    "obligations": [
                        "FIDO:2",
                        "PASSWORD:1",
                        "SMSOTP:2",
                        "SECURITYKEY:2",
                        "EMAILOTP:2"
                    ],
                    "msg": "Sample authentication policy"
                }
            }
        ],
        "createdDate": "2021-07-22 10:21 AM UTC",
        "updatedBy": {
            "principalType": "CLIENT",
            "principalId": "1fc0e655-b5a1-4cb5-9251-5828d9769029",
            "principalName": "defaulttenantclient"
        },
        "updatedDate": "2022-02-24 10:22 AM UTC",
        "preventedOperations": "Delete"
    }
```


– **result** - This property indicates the actions that a user must perform before getting authenticated. The result has three sub-properties, effect, reAuthenticate, and obligations.

- **effect**- This property specifies if the authentication request should be allowed or denied. Supported values of "effect" are "allow" and "deny". If the result of "effect" is denied, then the authentication process is ended there. However, if the result of "effect" is allow then the user is presented with an obligation that governs the authentication flow.
- **reAuthenticate**- This property is used to confirm that the user who has requested access to a secured resource is the same user who has authenticated at session initialization. By default this property is disabled. To know more, click here.
- **obligations**- This property specifies the user workflow for authentication. Here, multiple obligations can be returned by the solution, based on the policy evaluation. The authentication factors supported by the solution are FIDO (Biometrics), SECURITYKEY, SMSOTP, EMAILOTP, PUSH, PASSWORD, IVROTP, MOBILEOTP, PASSKEY, IDP (Identity Provider), and CUSTOM (Custom Authenticator). As a user, you can start the authentication flow from here based on the obligation returned. The obligation can have primary authentication and secondary authentication.

  The following table shows the format in which the primary and secondary authentication appears in a policy JSON:

Examples of Primary and Secondary Authentication	
Primary Authentication	"PASSWORD: 1"
Secondary Authentication	"SMSOTP: 2"
Multiple Primary Authentications	"EMAILOTP: 1", "PASSWORD: 1"
Multiple Secondary Authentications	"PASSWORD: 2", "FIDO: 2", "SECURITYKEY: 2", "PASSKEY:2", "IVROTP: 3", IDP:3

  The obligation can have multiple primary and/or secondary authentication factors. In such cases, you must call the factor selection API to select any one factor each from the primary and/or secondary authentication factors and perform the authentication. See Authentication flow for end-user to find more about factor selection. To know more about these factors, see Authentication Factors documentation.
- **inlineEnrollmentAllowed**- This optional property controls whether users can enroll new authentication factors inline. It enables you to explicitly permit or prevent enrollment based on the security context (for example allowing it on-network but disabling it off-network to prevent unauthorized additions). When explicitly set to true or false, this property overrides the tenant-level **isInlineEnrollmentAllowed** setting.
  - If a policy explicitly sets inlineEnrollmentAllowed = **false**, inline enrollment is blocked. (This property takes precedence over the tenant-level **isInlineEnrollmentAllowed** setting.)
  - If a policy explicitly sets inlineEnrollmentAllowed = **true**, inline enrollment is allowed.(This property takes precedence over the tenant-level **isInlineEnrollmentAllowed** setting.)
  - If the policy does not explicitly set inlineEnrollmentAllowed (or sets it to "**tenant**"), the system default's to using the Tenant setting, "**isInlineEnrollmentAllowed**".
- **mfaFrequency**-

  The "mfaFrequency" policy setting controls how often users are challenged for Multi-Factor Authentication. It determines whether MFA is required on every login or can be reduced to once per trusted device when the trusted device feature is already enabled at the tenant level.

  "mfaFrequency" can be either set to "OnceForTrustedDevice" or "EveryTime" (default).

  When "mfaFrequency" is set to "OnceForTrustedDevice":

  - The user is challenged for MFA only once per device
  - Subsequent logins require only primary authentication
  - Device is marked as trusted for the duration specified in the tenant setting, "trustedDeviceExpiryDays"

  When "mfaFrequency" is set to "EveryTime":

- Trusted device feature is disabled - User is challenged for MFA on every login
- "Remember This Device" checkbox is not displayed in the UI

Dependencies and Requirements when "mfaFrequency" is set to "OnceForTrustedDevice":

a. Tenant-Level Configuration "isTrustedDeviceEnabled": Must be enabled at tenant level
b. Policy-Level Configuration "mfaFrequency": Set to "OnceForTrustedDevice"
c. User Action Required User must check the "Remember This Device" checkbox in the UI

The feature only works when ALL three levels are properly configured (tenant settings, policy settings, and user action).

NOTE
If there are multiple policies assigned to an application that has both "mfaFrequency": "OnceForTrustedDevice" and "EveryTime" are configured, then "EveryTime" takes precedence over "OnceForTrustedDevice". The user is obliged to authenticate all the factors and the trusted device feature is effectively disabled.

- **AuthLevel**-This is an optional attribute that may be used to specify the authLevel in the policy results. When the authLevel is set in the policy, the ID token is populated with the claim, ***user_authlevel*** and the Access Token is populated with the claim as, ***subType='user'***.
  - When there are many rules to assess for the authentication flow, and every rule has authLevel defined, the highest authentication level is used, and the Identity token is populated with that value in the claim, "user_authlevel".
  - The "user authLevel" claim does not generate in the Identity Token if authLevel is not set in the policy result, or if the outcome of authLevel is Zero after considering numerous rules.

Sample Policy Payload

```
{

   "description": "Sample policy Record ",

   "status": "active",

   "policyName": "Sample policy",

   "apps": [

       {

           "id": "{{clientAppId}}"

       }

   ],

   "rules": [

       {

           "conditions": {

               "action": {

                   "operator": "in",
```

```
        "value": [
            "login"
        ]
    },
    "principal": {
        "group": {
            "operator": "in",
            "value": [
                "hr",
                "dev",
                "admin"
            ]
        },
        "user": {
            "operator": "in",
            "value": [
                "user1",
                " user2",
                " user3"
            ]
        }
    },
    "ipAddress": {
        "operator": "in",
        "value": [
            "10.0.0.0/24"
        ]
```

```
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "mobile"
                    ]
                },
                "riskThreshold": {
                    "operator": "ge",
                    "value": 60
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "true",
                "obligations": [
                    "PASSWORD:1",
                    "SMSOTP:2"
                ],
                "msg": "This policy is for login in mobile channel"
            }
        }
    ]
}
```

Rule Conditions

As you have already seen the format of a policy in the sample JSON, we now go on to explain the various rule **conditions** that the IDSP supports. These conditions are the most essential part of any policy and they govern the policy evaluation.

Based on the outcome of the policy evaluation, obligations are imposed on the user. When all the obligations are met, then the user is authenticated.

NOTE

- Policy rules may include multiple conditions.
- Characters "{", "}", "[", "]", or "?" are reserved for internal processing. Due to this none of the "Rule Conditions""Rule Conditions" should include these reserved characters in their values except "context" condition, where the "operator" is set to "expression".

Rule Conditions supported in IDSP
action (includes "authenticate" and "access"actions)
principal-user & group
ipAddress
channel
riskThreshold
idpName
acr
triggeredRiskRule
country
context

These conditions have been explained with examples in the subsequent topics. Click the condition to navigate to the respective condition descriptions.

Description of "action" condition

You can configure "authenticate" and, "access" values in the "action" condition.

- When the "action" condition is configured as "authenticate," it validates the incoming user's identity.
- When the "action" condition is configured as "access," it governs the user's permissions and privileges after authentication.

Both these "action" values have been described with examples in this section.

Setting the "action" condition to "authenticate"

The following example illustrates 'authenticate' condition.

```
{
   "description": "Sample policy Record ",
   "status": "active",
   "policyName": "Sample policy",
   "apps": [
       {
```

```
            "id": "{{clientAppId}}"
        }
    ],
    "rules": [
        {
            "conditions": {
               "action": {
                    "operator": "in",
                    "value": [
                        "authenticate"
                    ]
                },
            "result": {
                "effect": "allow",
                "obligations": [
                    "PASSWORD:1",
                    "SMSOTP:1"
                ],
                "msg": "This policy uses "action" condition"
            }
        }
    ]
}
```

User Flow:

In this example, the "action" condition depicts that the policy is applicable if the action being performed is "authenticate." The "in" operator ensures that the action under consideration matches the one specified in the value. If the action aligns with the defined value (authenticate), the policy condition is met, and a result is presented to the user. If the "effect" is "allow", then the user is displayed with an obligation that defines the authentication flow. If the "effect" is denied, the user is denied authentication.

In this example, the "effect" is "allow," which means the user is allowed to perform the authentication by successfully completing the primary authentication, "Password:1" or "SMSOTP: 1".

Setting the "action" condition to "Access"

The "action" condition when set to "access," gets triggered post authentication of the user.

NOTE

- This condition gets triggered only when the property "**requireAccessControl**" is either enabled in the **Application property** or in the **Tenant settings**. This property information is covered in Managing Applications , and Managing Settings.
- If you choose "AdminConsole" as the application in the AuthN policy with condition as "Access", it is mandatory to define the "principal" condition in the policy. In case the "principal" condition is undefined, that particular rule is ignored."

The following example illustrates an "access" condition.

```
{
"policyName": "Sample policy",
"description": "description of the policy",
"status": "active",
"apps": [
   {
      "id": "059bcbbc-c8b6-467d-85d0-6d15f5a5c0f7"
   }
 ],
"rules": [
  {
    "conditions": {
       "principal": {
          "user": {
             "operator": "in",
             "value": ["( (email ne *@broadcom.com) and (user_loginid ne *@broadcom.net) and
(group ne 00g1g3dsq9x59eZ641d8) )"]
          }
       },
       "action": {
         "operator": "in",
         "value": [ "access"]
       }

       }
    },

     "result": {
     "effect": "allow",
     "msg": "This policy is for authenticate in mobile channel"
     }
   }
  ]
 }
```

User Flow:

In this example, the "action" condition depicts that the policy is applicable if the action being performed is "access." The "in" operator ensures that the action under consideration matches the one specified in the array. If the action aligns with

the defined value (access), the policy condition is met, and the subsequent rules or decisions in the policy is applied. If the policy evaluation results in a match between the user input and the action value that has been configured in the condition, then this condition is triggered and a result is presented to the user. If "effect" is "allow", then the user is allowed to access the resource. If the "effect" is deny, then the user is denied access to the resource.

Description of "principal" condition

Example of a rule that contains "principal" as a condition.


```
{
        "policyId": "dec6e47c-0b9a-462e-b4e8-800a7433592d",
        "policyName": "Test Policy",
        "description": " ",
        "status": "active",
        "matchAnyApp": true,
        "rules": [
            {
                "id": "b0c756c5-1559-4f3d-8699-6c6b1cb731c5",
                "conditions": {
                    "principal": {
                        "group": {
                            "operator": "in",
                            "value": [
                                "Employees",
                                "SSOUD/Finance",
                                "e89a1aaa-2fcc-4e54-9081-d6646ac83030/Development"
                            ]
                        },
                        "user": {
                            "operator": "in",
                            "value": [
                                "nbruce",
                                "*",
                                "SSOUD/nbruce",
                                "SSOUD/*",
                                "81dbf453-3b72-4855-a6c9-782156470ad3/nbruce",
                                "81dbf453-3b72-4855-a6c9-782156470ad3/*"
                            ]
                        }
                    },
                    "action": {
                        "operator": "in",
                        "value": [
                            "authenticate"
                        ]
                    }
                },
                "result": {
                    "effect": "allow",
                    "reAuthenticate": "false",
                    "obligations": [
                        "PASSWORD:1"
                        "SMSOTP:2"
```

```
                    ],
                    "msg": "Sample Authentication Rule",
                    "mfaFrequency": "OnceForTrustedDevice"
          }

        }

    ]

}
```

The principal condition contains two attributes, the group, and the user. This condition uses "or" as the operator value. You can define a user condition, a group condition, or both group and user in a condition.

> **NOTE**
> The "principalConditionRequiredInPolicy" tenant/global setting determines whether an empty principal condition is allowed in a policy. When this property is set to true in the IDSP settings, the POST/PUT calls of Authentication policy for the new policies requires the presence of "**principal**" condition. Whereas the existing policies which were created without the principal condition continues to exist as-is. See Global/Tenant Settings.

Principal conditions are explained here.

Only "**User**" condition:
When a policy has a principal condition that is configured with the "**user**" attribute, then this policy is triggered when an incoming authentication request containing "subject" contains the value of the user who is mentioned in the policy attribute. If the policy evaluation results in a match between the user input and the user value that has been configured in the condition, then this condition is met and a result is presented to the user. If "effect" is "allow", then the user is displayed with an obligation that defines the authentication flow. If the "effect" is denied, then the flow ends there. In this example, five users are configured under the "user" condition, if the user input matches with any one of the users, then the policy is triggered.

The following example shows the type of values that are supported in the "**user**" condition:

```
principal:{
           "user": {
             "operator": "in",
             "value": [
               "nbruce",
               "*",
               "SSOUD/nbruce",
               "SSOUD/*",
               "81dbf453-3b72-4855-a6c9-782156470ad3/nbruce",
               "81dbf453-3b72-4855-a6c9-782156470ad3/*"
            ]
          }
       }
```

The following table explains the values that are supported in the "**user"** condition:

"nbruce"	The disambiguated user, "nbruce" can belongs to one of the following: • any IDSP ID Store identifier configured in the application setting, "idStoreToUse" or in the tenant setting "idStoreToBeUsed" If the application setting "idStoreToUse" is not configured, the policy engine looks for the configuration set in the tenant setting "idStoreToBeUsed". • any IDSP IdentityProvider identity that is found in the Identity Token (IT) or Identity Token Hint (ITH) See Managing Applications and Global/Tenant Settings.
"*"	The disambiguated user "*" can be any user who belongs to one of the following: • any IDSP ID Store identifier that is configured in the application setting "idStoreToUse" or in the tenant setting "idStoreToBeUsed" If the application setting "idStoreToUse" is not configured, the policy engine looks for the configuration set in the tenant setting "idStoreToBeUsed". See Managing Applications and Global/Tenant Settings. • any IDSP IdentityProvider identity that is found in the Identity Token (IT) or Identity Token Hint (ITH) • "*" value is supported only in the Authentication Policy, and not supported in the Authorization policy.
"SSOUD/nbruce"	The value "SSOUD/nbruce" in this user condition indicates that the disambiguated user "nbruce" is a member of the SSOUD Identity Provider.
"SSOUD/*"	The value "SSOUD/*" in this user condition indicates that the disambiguated user can be anybody from the SSOUD Identity Provider.
"81dbf453-3b72-4855-a6c9-782156470ad3/nbruce"	The value "81dbf453-3b72-4855-a6c9-782156470ad3/nbruce" in this user condition indicates that the disambiguated user, "nbruce" has the IDSP ID Store identifier or IDSP IdentityProvider identifier as 81dbf453-3b72-4855-a6c9-782156470ad3.
"81dbf453-3b72-4855-a6c9-782156470ad3/*"	The value "81dbf453-3b72-4855-a6c9-782156470ad3/*" in this user condition indicates that the disambiguated user can be any user who has the IDSP ID Store identifier or IDSP IdentityProvider identifier as 81dbf453-3b72-4855-a6c9-782156470ad3.

Use of Expressions in "User" condition:

IDSP supports Attribute Based Access Control (ABAC) which is an authorization model that evaluates the user attributes in the authentication policy conditions to determine the level of user access.

This is a policy condition that allows the identification of users based on attributes found in their user record from external (LDAP/SCIM) or Internal Identity Store. You can configure this condition by specifying an array of attribute values in the "principal.user" condition.

The following example shows how user attributes can be defined using expressions in the principal.user condition of the authentication policy.

```
 "rules": [
         {
             "conditions": {
                 "principal": {
                     "user": {
                         "operator": "in",
```

```
                        "value": ["( (email ne *@broadcom.com) and (user_loginid ne *@broadcom.net) and (group
ne 00g1g3dsq9x59eZ641d8) )"]
                    }
                },
```

Where,

- (email ne *@example.com): User's email should not be *@example.com.
- (user_loginid ne *@example.net): User's login ID should not be *@example.net.
- (group ne 00g1g3dsq9x59eZ641d8): User should not belong to the group with ID 00g1g3dsq9x59eZ641d8.

In summary, the Authentication Policy example sets conditions for access based on the attributes of the user. If the user's email, login ID, and group membership meet the specified criteria, the rule defined in the policy will be applied. To know more about expressions supported in Authentication Policies, see Expressions Supported.

Only "**Group**" condition:
When a policy has a principal condition that is configured with the "**group**" attribute, then this policy is triggered when an incoming authentication request containing "subject" contains the user who belongs to the group mentioned in the policy attribute. If the policy evaluation results in a match between the user input and the group value that has been configured in the condition, then this condition is met and a result is presented to the user. If the "effect" is "allow", then the user is set forth with an obligation that defines the authentication flow. If the "effect" is denied, the flow ends there. In this example, there are multiple groups that are configured under the "group" condition, if the user input matches with any one of these groups, then the policy is triggered.

The following example shows the type of values that are supported in the "**group**" condition:

```
principal: {
            "group":{
             "operator":"in",
             "value":[
                     "Employees",
                     "SSOUD/Finance",
                     "e89a1aaa-2fcc-4e54-9081-d6646ac83030/Development",
                ]
              }
           }
```

The following table explains the values that are supported in the "**group**" condition:

"Employees"	In this group condition, the disambiguated user should be a member of the "Employees" group in any of the ID Stores (LDAP/SCIM) or IDP (Identity Provider in ZFP mode).
"SSOUD/Finance"	In this group condition, the disambiguated user should be a member of the "Finance" group. The disambiguated user comes through the ZFP flow where the Identity Provider is assumed to be SSO User Directory (SSOUD).
"e89a1aaa-2fcc-4e54-9081-d6646ac83030/Development"	In this group condition, the disambiguated user should have the IDSP ID Store identifier or IDSP IdentityProvider identifier Id as e89a1aaa-2fcc-4e54-9081-d6646ac83030. The user should also be a member of the "Development" group.

NOTE
The "group" condition does not support "*" value.

Both **User** and **Group**:

When a policy has a principal condition that is configured with both "user" and "group" attribute, then this policy is triggered when an incoming authentication request containing "subject", contains the value of the user, who belongs to any of the groups. For example, **user: "nbruce", group {"Employees","SSOUD/Finance"}** - in this instance, "nbruce" must belong either to the "Employees" or "Finance" group. For a detailed description of group-SSOUD/Finance refer to the previous table. If the policy evaluation results in a match between the user input (which may contain either user or group attribute) with the values that have been configured in the condition, then this condition is met and a result is presented to the user. If the "effect" is "allow", then the user is displayed with an obligation that defines the authentication flow. If the "effect" is "deny", then the flow ends there.

In the example policy stated previously, the "effect" is "allow," which means the user is allowed to perform the authentication by successfully completing the primary authentication, "PASSWORD: 1" and secondary authentication "SMSOTP: 2".

Description of "ipAddress" condition

Example of a rule that contains "ipAddress" as a condition.

```
{
 "description": "Sample policy Record ",
 "status": "active",
 "policyName": "Sample policy",
 "apps": [
     {
         "id": "{{clientAppId}}"
     }
 ],
 "rules": [
     {
         "conditions": {
             "ipAddress": {
                 "operator": "in",
                 "value": [
                     "10.0.0.0/24"
                 ]
             },
         "result": {
```

```
                "effect": "allow",

                "obligations": [

                    "PASSWORD:1",

                    "SMSOTP:2"

                ],

                "msg": "This policy uses "ipAddress" condition"

            }

        }

    ]

}
```

The "ipAddress" condition specifies the connectivity to the IDSP solution from a specific ipAddress or a range of ipAddresses. You can configure the "ipAddress" range in the condition.

When a user provides inputs that contain the "ipAddress" attribute, then this policy is triggered and the user inputs are evaluated. If the policy evaluation results in a match between the user input and the "ipAddress" range that has been configured in the condition, then the condition is met and a result is presented to the user. If the "effect" results in "allow", then the user is displayed with an obligation that defines the authentication flow. If the "effect" results in a deny, the flow ends there. The supported value of ipAddress is - ipAddress range.

You can specify multiple values, which are separated by commas, or can use quotation marks around each individual value. The ipAddress condition uses "in" as the operator value.

In this example, the "effect" is "allow," which means the user is allowed to perform the authentication by successfully completing the primary authentication, "Password:1" and secondary authentication "SMSOTP: 2".

".../admin/v1/AuthPolicies" API Supports IPv6 IP Addresses in Policy Rules

The following example shows IPv6 format and IPv6 range included in an Authentication Policy Rule:

```
"rules": [
            {
                "conditions": {
                    "action": {
                        "operator": "in",
                        "value": [
                            "authenticate"
                        ]
                    },
                    "ipAddress": {
                        "operator": "in",
                        "value": [
                            "2A0D:5600:40:4000:A38D:9321:8819:8A8B/32"
                        ]
                    }
                },
                "result": {
```

```
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "MOBILEOTP:1"
                ],
                "msg": "ipv6 range japan",
                "mfaFrequency": "OnceForTrustedDevice"
            }
        },
        {
            "conditions": {
                "action": {
                    "operator": "in",
                    "value": [
                        "authenticate"
                    ]
                },
                "ipAddress": {
                    "operator": "in",
                    "value": [
                        "2405:205:2106:e8f6:40a4:59cd:5ac9:49fa"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PUSH:1"
                ],
                "msg": "ipv6 india",
                "mfaFrequency": "EveryTime"
            }
        }

    ]
```

Description of "channel" condition

Example of a rule that contains "channel" as a condition.

```
{

  "description": "Sample policy Record ",

  "status": "active",

  "policyName": "Sample policy",

  "apps": [

      {

          "id": "{{clientAppId}}"
```

```
        }
    ],
    "rules": [
        {
            "conditions": {
                "channel": {
                    "operator": "in",
                    "value": [
                        "mobile"
                    ]
                },
            "result": {
                "effect": "allow",
                "obligations": [
                    "PASSWORD:1",
                    "SMSOTP:2"
                ],
                "msg": "This policy is for login in mobile channel"
            }
        }
    ]
}
```

In the channel condition, you can configure the medium through which a user can connect to the IDSP. The channel condition uses "in" as the operator value. Examples of channel: Web and Mobile

When a user provides an input that contains the "channel" attribute, then this policy is triggered and the user inputs are evaluated. If the policy evaluation results in a match between the user input and the "channel" that has been configured in the condition, then the condition is met and a result is presented to the user. If the "effect" results in "allow", then the user is displayed with an obligation that defines the authentication flow. If the "effect" results in deny, the flow ends there.

In this example, the "effect" is "allow," which means the user is allowed to perform the authentication by successfully completing the primary authentication, "Password:1" and secondary authentication "SMSOTP: 2".

Description of "riskThreshold" condition

Example of a rule that contains "riskThreshold" as a condition.

```
{
  "description": "Sample policy Record ",
  "status": "active",
  "policyName": "Sample policy",
  "apps": [
      {
          "id": "{{clientAppId}}"
      }
  ],
  "rules": [
      {
          "conditions": {
              "riskThreshold": {
                  "operator": "ge",
                  "value": 60
              }
          },
          "result": {
              "effect": "allow",
              "obligations": [
                  "PASSWORD:1",
                  "SMSOTP:1"
              ],
              "msg": "This policy is for risk over 60"
```

```
            }

        }

    ]

}
```

The riskThreshold condition enables you to determine the likelihood of the authentication attempt being malicious or fraudulent. This condition uses the following operator values; eq (equal to), ge (greater than equal to), le (Less than equal to), gt (Greater than), and lt (Less than).

When a user provides input that contains the "riskScore" attribute, then this policy is triggered.

In this example, the "riskThreshold" is configured with value - gt 60. When a user makes an authentication call, it sends additional details like device information, device signature, and ipaddress to IDSP, which enables the risk server to evaluate the risk score of the user. If the risk score that is returned by the server is greater than 60, then this condition is met and a result is presented to the user. If the risk score that is returned by the server is less than 60, then this policy does not get triggered.

In this example, the "effect" is "allow," which means the user is allowed to perform the authentication by successfully completing the primary authentication, "Password:1" or "SMSOTP: 1".

Description of "idpName" condition

Example of a rule that contains "idpName" as a condition.

```
{
        "policyName": "Feature Policy Demo – siteminder integration",
        "description": "Feature Policy Demo - siteminder integration",
        "status": "active",
        "matchAnyApp": false,
        "apps": [
            {
                "name": "siteminder",
                "id": "7f27b5cd-5e22-43b7-a7a5-942472581ffe"
            }
        ],
        "rules": [
            {
                "conditions": {
                        "idpName": {
                               "operator": "in",
                               "value": ["Employees"]
                           },
                    "acr": {
                        "operator": "in",
                        "value": [
                            "test"
                        ]
                    }
                },
                "result": {
                    "effect": "allow",
```

```
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD:1"
                ],
                "msg": "Rule1: Password auth for users in 3rd party directory"
            }
        },
        {
            "conditions": {
                "idpName": {
                    "operator": "in",
                    "value": ["Consumers"]
                },
                "acr": {
                    "operator": "in",
                    "value": ["test"]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD:1", "OTP:2", "MOBILEOTP:2"
                ],
                "msg": "Rule1: Password auth for users in LDAP store"
            }
        }
    ]
}
```

The idpName condition indicates the logical name representing the identity namespace of the user in a policy condition. This condition is optional, and can be added to the policy by using a POST/PATCH operation. idpName condition is used in scenarios where the client uses Zero Footprint Identity. See Zero Footprint Identity for more information.

NOTE
In cases where SiteMinder integrates with IDSP using ZFP, this is the name of the User Directory where the user identity exists. SiteMinder's "IDSP" Auto Scheme sets this claim automatically.

Description of "acr" condition

The Authentication Context Class Reference (ACR) is a parameter that is used during the creation of a policy inIDSP. It specifies the ACR Values within the authentication request that the Authorization server is requested to use for the processing of requests from the Client. The values are processed based on the preferred order in which they are added to the policy.

When a user provides input that contains the "acrValues" attribute, then this policy is triggered.

In the following example, the "acr" is configured with value - "urn:iam:acr:aal:aal1". When a user makes an authentication call with 'acr' values in the request, the policy evaluates the incoming 'acrValues' with the 'acr' values that are specified in the policy. After evaluation, if these values match, then the user will be presented with the obligations that are defined in the policy.

NOTE
If multiple 'acr' values are specified in the policy rule, then the first 'acr' value from the list is considered for evaluation. If the first 'acr' value does not match, then the next value in the list is evaluated, the evaluation process continues until an "acr" match has been found. Once the match is found, the remaining 'acr' values are not evaluated.

In case, the policy evaluation results in a mismatch of 'acr' values then the authentication will be denied to the user.

In this example, the "effect" is "allow," which means the user is allowed to perform the authentication by successfully completing the primary authentication, "Password:1".

Let us see an example of policy comprising of "acr" condition:

Example of a rule which contains an "acr" condition.

```
{
  "policyName": "ACR Based Policy",
  "description": "Policy with rules based on the ACR condition",
  "status": "active",
  "apps": [
    {
      "id": "059bcbbc-c8b6-467d-85d0-6d15f5a5c0f7"
    }
  ],
  "rules": [
    {
      "conditions": {
        "principal": {
          "user": {
            "operator": "in",
            "value": [
              "user1",
              "user2"
            ]
          }
        },
        "action": {
          "operator": "in",
          "value": [
            "authenticate"
          ]
        },
        "acr": {
          "operator": "in",
          "value": [
            "urn:iam:acr:aal:aal1"
          ]
        }
      },
      "result": {
        "effect": "allow",
        "obligations": [
          "PWD:1"
        ],
        "msg": "AAL1 level policy rule"
      }
```

```
    }
  ]
}
```

NOTE
When you must define multiple conditions in an acr policy, ensure that all those defined conditions such as 'action' or 'channel' are included in the acr rule.

To know more about acr, click here.

Description of "triggeredRiskRule" condition

Example of a rule that contains "triggeredRiskRule" as a condition.

```
{
    "policyName": "Feature Policy for Demo client",
    "description": "Feature Policy Demo client",
    "status": "active",
    "matchAnyApp": false,
    "apps": [
        {
            "name": "DemoClient",
            "id": "95845b14-3044-4a7a-ab88-0972079f12b3"
        }
    ],
    "rules": [
        {
            "conditions": {
                "triggeredRiskRule": {
                    "operator": "in",
                    "value": [
                        "User Velocity",
                        "Risky Country"
                    ]
                }
            },
            "result": {
                "effect": "deny",
                "reAuthenticate": "false",
                "msg": "TriggeredRiskRule - Deny authentication if the user is attempting to login multiple
 times, or authenticating from risky countries.",
                "mfaFrequency": "EveryTime",
                "authLevel": 100
            }
        }
    ]
}
```

This policy condition lets you add multiple triggered risk rules that are supported by IDSP, such as "User Velocity", "Risky Country". During authentication, when a user attempts a number of times to log into an application within a span of time, or if a user authenticates from a IP address that is considered risky, this condition gets triggered.

If the "effect" is "allow", then the user is displayed with an obligation that defines the authentication flow. If the "effect" is deny, the user is denied authentication. In this example, the "effect" is "allow," which means the user is allowed to perform the authentication by successfully completing the primary authentication, "Push:1".

To know about the various risk rules that are supported in IDSP, see Understanding Risk Rules .

Description of "country" condition

Example of a rule that contains "country" as a condition.

```
{
    "policyName": "Feature Policy for Demo client",
    "description": "Feature Policy Demo client",
    "status": "active",
    "matchAnyApp": false,
    "apps": [
        {
            "name": "DemoClient",
            "id": "95845b14-3044-4a7a-ab88-0972079f12b3"
        }
    ],
    "rules": [
        {
            "conditions": {
                "country": {
                    "operator": "not",
                    "value": [
                        "us",
                        "in"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "FIDO:2"
                ],
                "msg": "Country - Stepup authentication for countries other than USA and India",
                "mfaFrequency": "EveryTime",
                "authLevel": 100
            }
        }
    ]

}
```

This policy condition lets you add multiple country names that will be considered during the policy evaluation. When a user authenticates using an IP address from a country that is specified in the condition, then this policy is triggered. This condition uses the following operator values; "not", "eq" and "in".

In this sample JSON, we have taken an example of the "country" condition where the configured values are "us" and "in". At runtime, when a user authenticates using an IP address that belongs to either of these countries, "us" or "in" this policy gets triggered and the user inputs are evaluated. Upon evaluation, if the value of the user action matches the value that has been configured in the condition, then the condition is met and a result is presented to the user.

If the "effect" is "allow", then the user is displayed with an obligation that defines the authentication flow. If the "effect" is deny, the user is denied authentication. In this example, the "effect" is "allow," which means the user is allowed to perform the authentication by successfully completing the primary authentication, "SMSOTP: 1".

Description of "Context" condition

Example of a sample policy that defines the "context" condition.

```
{
    "policyName": "Sample policy",
    "description": "description of the policy",
    "status": "active",
    "apps": [
        {
            "id": "059bcbbc-c8b6-467d-85d0-6d15f5a5c0f7"
        }
    ],
    "rules": [
        {
            "conditions": {
                "principal": {
                    "user": {
                        "operator": "in",
                        "value": ["( (email ne *@broadcom.com) and (user_loginid ne *@broadcom.net) and (group
 ne 00g1g3dsq9x59eZ641d8) )"]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": ["authenticate"]
                },
               "context": {
                    "operator": "expression",
                    "value": "( ${clientcontext.env} eq ${appmetadata.env} ) and ( ${appmetadata.department}
 eq ${clientcontext.department} ) and ((${idprofile.user_loginid} eq kevinlucas) or (${idtoken.sub} eq
 kevinlucas))"
                }
            },
            "result": {
                "effect": "allow",
                "obligations": ["PASSWORD:1", "SMSOTP:2"],
                "msg": "This policy is for authenticate in mobile channel"
            }
        }
    ]
}
```

In this example, under the "context" condition, the "operator" is set to "expression," indicating that a complex expression is used for evaluating access. The value consists of three main conditions separated by the operator "and".

1. Condition 1: `${clientcontext.env} eq ${appmetadata.env}` - Checks if the environment in the client context is equal to the environment in the application metadata.
2. Condition 2: `${appmetadata.department} eq ${clientcontext.department}` - Ensures that the department in the application metadata matches the department in the client context.

3. Condition 3: (${idprofile.user_loginid} eq kevinlucas) or (${idtoken.sub} eq kevinlucas) - Checks if either the user login ID in the identity profile is "kevinlucas" or the subject in the identity token is "kevinlucas."

This context expression specifies that access is allowed if the environment, department, and user login ID or subject meet the defined criteria. The "and" logical operators require all conditions to be satisfied for the expression to evaluate to true and grant access. To know more about different contexts used as expression in the "context" policy condition, see Expressions Supported.

Reference links:

- Authentication Policy APIs
- Authentication Policy Patch Operations
- Policy API Error Codes
- Authentication Factors

Authentication Policy APIs

Learn about the Symantec Identity Security Platform Authentication APIs, its operations, and examples.

As an administrator, you have the capability to utilize the Authentication (AuthN) Policy APIs for various purposes, including creating new policies, retrieving, editing, and deleting existing policies. Additionally, you can add, remove, or replace "apps" within an authentication policy using a defined set of APIs.

The AuthN Policy APIs control all authentication operations of a user as per the rules set in the policy and routes users to perform the necessary actions to authenticate themselves.

> **NOTE**
> IDSP provides a default authentication policy. If you make any updates to this policy, it overrides the default and the changes are irreversible.

The following table lists out the Policy APIs available to manage IDSP Policies. For sample request and response payloads of all these URLs, refer to Policy API Swagger documentation.

Operation Description	Operation Type	Endpoint URLs
This API allows creating an authentication policy	POST	https://${IDSP_FQDN}/{{tenantName}}/admin/v1/AuthPolicies
This API retrieves all authentication policies.	GET	https://${IDSP_FQDN}/{{tenantName}}/admin/v1/AuthPolicies
This API retrieves an authentication policy that is specified by the policy Id.	GET	https://${IDSP_FQDN}/{{tenantName}}/admin/v1/AuthPolicies/{{policyId}}
This API allows updating an authentication policy by specifying the policy Id.	PUT / PATCH	https://${IDSP_FQDN}/{{tenantName}}/admin/v1/AuthPolicies/{{policyId}}
This API deletes an authentication policy by specifying the policy Id. Note that seeded/embedded authentication policies within IDSP cannot be deleted.	DELETE	https://${IDSP_FQDN}/{{tenantName}}/admin/v1/AuthPolicies/{{policyId}}
The following APIs are used to perform actions such as ***adding apps*** to, ***removing apps*** from, or ***replacing apps*** in a specified authentication policy. Note that these API operations are exclusively applicable to existing authentication policies and do not apply to new ones. These operations are performed by the individuals with the ***Integration Admin Role*** , who are granted the ***"urn:iam:t.apps"*** scope in addition to other scopes. See Integration Administrator.		
This API substitutes the entire set of applications in the authentication policy with the specified set of applications.	PUT	https://${IDSP_FQDN}/{{tenantName}}/admin/v1/AuthPoliciesApps/<authpolicy-id>

This API allows the addition of specified applications to an authentication policy.	PATCH	https://${IDSP_FQDN}/{{tenantName}}/admin/v1/AuthPoliciesApps/<authpolicy-id>
Deletes given apps from the authentication policy.	DELETE	https://${IDSP_FQDN}/{{tenantName}}/admin/v1/AuthPoliciesApps/<authpolicy-id>
This API returns authentication policies that include the specified application.	GET	https://${IDSP_FQDN}/{{tenantName}}/admin/v1/AppsAuthPolicies/<app-id>

NOTE

- To know the value of ${IDSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/AuthPolicies`
  should look similar to:
  `https://example.com/default/admin/v1/AuthPolicies`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Examples of Authentication Policy Operations

POST https://${IDSP_FQDN}/{{tenantName}}/admin/v1/AuthPoliciesApps/<authpolicy-id>

Request:

```
{
 "policyName": "AdminLoginPolicy",
 "description": "Default policy for Tenant Admin Console",
 "status": "active",
 "matchAnyApp": false,
 "apps": [
 {
 "name": "DefaultTenantClient",
 "id": "5960f72b-aeee-4f1f-8841-1d8cd3c8b76b"
 }
 ],
 "rules": [
 {
 "conditions": {
 "principal": {
 "group": {
 "operator": "in",
 "value": [
 "Admin"
 ]
 }
 },
 "channel": {
 "operator": "in",
 "value": [
 "web",
 "mobile"
 ]
```

```
  }
  },
  "result": {
  "effect": "allow",
  "reAuthenticate": "false",
  "obligations": [

  "SECURITYKEY:1"
  ],
  "msg": "This policy is for admin login"
  }
  }
  ]
 }
```

Response:

```
  {
  "policyId": "3e8ec344-5c15-430c-8919-18dc806bb0c7",
  "policyName": "AdminLoginPolicy",
  "description": "Default policy for Tenant Admin Console",
  "status": "active",
  "matchAnyApp": false,
  "apps": [
  {
  "name": "DefaultTenantClient",
  "id": "5960f72b-aeee-4f1f-8841-1d8cd3c8b76b"
  },
  {
  "name": "AdminConsole",
  "id": "$TENANT_ADMINCONSOLE"
  }
  ],
  "rules": [
  {
  "id": "5ff23478-6d75-4fb7-98a3-25d71cbbc01f",
  "conditions": {
  "principal": {
  "group": {
  "operator": "in",
  "value": [
  "Admin"
  ]
  }
  },
  "channel": {
  "operator": "in",
  "value": [
  "web",
  "mobile"
  ]
  }
  },
  "result": {
  "effect": "allow",
```

```
 "reAuthenticate": "false",
 "obligations": [
 "PASSWORD:1"
 ],
 "msg": "This policy is for admin login"
 }
 }
 ],
 "createdDate": "2023-10-18 07:44 AM UTC",
 "updatedBy": {
 "principalType": "CLIENT",
 "principalId": "f7a6500b-c26d-452c-9b34-18d8c41b1225",
 "principalName": "DefaultTenantClient"
 },
 "updatedDate": "2023-10-26 01:36 PM UTC",
 "preventedOperations": "Delete"
}
```

PUT https://${IDSP_FQDN}/{{tenantName}}/admin/v1/AuthPoliciesApps/<authpolicy-id>

```
[
     {
         "id": "d809829d-c004-4d28-8787-58bbe6b55a21"
     },
     {
         "id": "d8094444-1111-abcd-0123-58bbe6bcccc"
     }
]
```

PATCH https://${IDSP_FQDN}/{{tenantName}}/admin/v1/AuthPoliciesApps/<authpolicy-id>

```
[
     {
         "id": "d8094444-1111-abcd-0123-58bbe6bcccc"
     }
]
```

DELETE https://${IDSP_FQDN}/{{tenantName}}/admin/v1/AuthPoliciesApps/<authpolicy-id>

```
[
     {
         "id": "d8094444-1111-abcd-0123-58bbe6bcccc"
     }
]
```

GET https://${IDSP_FQDN}/{{tenantName}}/admin/v1/AppsAuthPolicies/<app-id>

Response: The response of the GET operation contains authentication policies that contain given app directly or indirectly through

```
200 OK
[
   {
        "policyName": "Default Policy",
        "description": "Default policy for Tenant Admin Console",
        "status": "active",
        "matchAnyApp": false
   },
   {
        "policyName": "Another Policy",
```

```
        "description": "Sample Policy1",
        "status": "active",
        "matchAnyApp": true
    }
]
```

Using Policy Patch Operation

Use the Patch operation in Symantec Identity Security Platform policy REST API to partially update an existing policy. The Patch operation can only be performed on user updatable fields for Authentication (AuthN) policies.

All patch syntax uses JSON Patch specification that is described in RFC 6902 (http://tools.ietf.org/html/rfc6902).

This topic includes the following:

Sample Patch Operation Payload

This is an example displaying one of the patch formats that is supported by IDSP to perform a "***replace***" operation on Authentication or Authorization Policy. However, there are multiple formats supported for the patch operation which are discussed in the sub-topics.

```
{
"Operations":
    [{
        "op": "replace",
        "path": "/rules/[id eq ruleId]",
        "value": [{
        }]
    }]
}
```

Where,

- op = This is the operation type. Allowed operation types are "add", "remove" and "replace".
- path = This is a string which contains an attribute path that describes the operation's target. The "**id**" is an attribute of rules. In the above example, replace the value of "path" ("path": "/rules/[id eq ruleId]") with the new value. Sample Syntax:
  "path":"/rules/[id eq \"f28b2cd4-f793-4e22-af68-29249bd3ca9d\"]"
- value = Contains the new value to be patched.

  NOTE
  A "*remove*" operation does not require a value.

Patch operations and their supported formats in AuthN policy have been described in the following topic:

- Patch operations and formats supported in Authentication (AuthN) Policy

Patch Operations Supported in Authentication (AuthN) Policy

You can perform *"add, "remove",* and *"replace"* patch operations on Authentication policy. These operations can be performed on the following attributes:`rules, description, and status.`

Policy Patch Endpoint URL

```
https://${SSP_FQDN}/{{tenantName}}/admin/v1/AuthPolicies/{{policyId}}
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/AuthPolicies/{{policyId}}`
  should look similar to:
  `https://example.com/default/admin/v1/AuthPolicies/{{policyId}}`
- To know more about the scopes that are required to secure these APIs, refer to the "Scopes" section here.

Patch formats supported in Authentication (AuthN) Policy

There are several formats that can be used in Authentication policy to perform patch operations based on the format of the path.

The following table displays supported AuthN policy patch formats:

Patch Format	Examples
Format 1- "path": "/rules	"path": "/rules"
Format 2 - "path": "/rules/[id eq \"<RULEID>\"]"	"path": "/rules/[id eq \"d28b2cd4-f793-4e22-af68-29249bd3ca9d\"]"
Format 3 - "path": "/rules/[id eq \"<RULEID>\"]/<ABSOLUTE PATH OF RULES ATTRIBUTE> "	"path": "/rules/[id eq \"d28b2cd4-f793-4e22-af68-29249bd3ca9d\"]/conditions/principal/group" Note that in this format you can patch the following attributes: group (Type-String), group (Type-Array), user (Type-String & Array), action (Type- String & Array), channel (Type- String & Array), idpName (Type- String & Array), Ipaddress (Type- String & Array), riskthreshold, Result - obligations, effect, msg, reauthenticate, and mfaFrequency.
Other PATCH Formats	policyName, description, Effect, reAuthenticate, channel, user, status.

Steps to perform patch operation on Authentication Policy

The following example shows the steps to update the authenticate policy with a "replace" patch operation.

1. Perform a GET operation to retrieve the Authentication Policy. The following code snippet displays the retrieved Authentication Policy which has multiple rules.

```
{
    "description": "Sample policy Record ",
    "policyName": "Sample policy1",
    "status": "active",
    "matchAnyApp": false,
    "rules": [
        {
```

```
    "id": "d28b2cd4-f793-4e22-af68-29249bd3ca9d",
    "conditions": {
        "action": {
            "operator": "in",
            "value": [
                "login"
            ]
        },
        "principal": {
            "group": {
                "operator": "in",
                "value": [
                    "Customers",
                    "HR"
                ]
            }
        }
    },
    "result": {
        "effect": "allow",
        "obligations": [
            "PASSWORD",
            "OTP"
        ],
        "msg": "This policy is for login in mobile channel"
    }
},
{
                "id": "f28b2cd4-f793-4e22-af68-29249bd3ca9d",
    "conditions": {
        "ipAddress": {
            "operator": "in",
            "value": [
                "10.0.0.0/24"
            ]
        },
        "channel": {
            "operator": "in",
            "value": [
                "web"
            ]
        },
        "riskThreshold": {
            "operator": "ge",
            "value": 60
        }
    },
    "result": {
        "effect": "allow",
        "obligations": [
            "PASSWORD",
            "OTP"
        ],
```

```
                "msg": "This policy is for login in mobile channel",
                "mfaFrequency;: "OnceForTrustedDevice"  }
        }
    ]
        }
```

2. Perform a "*replace*" operation on the Authentication Policy by replacing the content within the rule id-"f28b2cd4-f793-4e22-af68-29249bd3ca9d". The patch payload is in the JSON format and would look like the following sample.

```
curl \
-X PATCH
-H "Content-Type:application/json" \
-H "Authorization: Bearer <Access Token Value>" \
https:/<SERVICE_URL>/default/admin/v1/AuthPolicies/{id} \
{
"Operations":
[
  {
    "op": "replace",
    "path": "/rules/[id eq \"f28b2cd4-f793-4e22-af68-29249bd3ca9d\"]",
    "value": {
        "conditions": {
            "channel": {
                "operator": "eq",
                "value": [
                    "web"
                ]
            }
        },
        "result": {
            "effect": "allow",
            "obligations": [
                "PASSWORD"
            ],
            "msg": "Perform PATCH OPERATION"
        }
    }
  }
]
```

3. The following is the result of the "**replace**" patch operation. In the following code snippet, the "replace" operation has replaced all the current conditions within the rule with a single condition of "channel".

```
{
    "description": "Sample policy Record ",
    "policyName": "Sample policy1",
    "status": "active",
    "matchAnyApp": false,
    "rules": [
        {
            "id": "d28b2cd4-f793-4e22-af68-29249bd3ca9d",
            "conditions": {
                "action": {
                    "operator": "in",
                    "value": [
```

```
                        "login"
                    ]
                },
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "Customers",
                            "HR"
                        ]
                    }
                }
            },
            "result": {
                "effect": "allow",
                "obligations": [
                    "PASSWORD",
                    "OTP"
                ],
                "msg": "This policy is for login in mobile channel"
            }
        },
        {
            "id": "f28b2cd4-f793-4e22-af68-29249bd3ca9d",
            "conditions": {
                "channel": {
                    "operator": "eq",
                    "value": [
                        "web"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "obligations": [
                    "PASSWORD"
                ],
                "msg": "Perform PATCH OPERATION"
            }
        }
```

Format 1-("path": "/rules")

Example 1: "add" patch operation

1. In this format ("path": "/rules"), you can perform "add", "replace" and "remove" operations. The following code snippet displays the patch payload to be used for an "add" operation in Authentication policy.

   NOTE
   Perform a GET operation to retrieve the Authentication Policy before executing the "add" operation. In this example, we have assumed that the retrieved policy is the same as the response payload of the previous policy.

```
{
    "Operations": [
        {
```

```
            "op": "add",
            "path": "/rules",
            "value": [
                {
                    "conditions": {
                        "principal": {
                            "group": {
                                "operator": "in",
                                "value": [
                                    "lib"
                                ]
                            },
                            "user": {
                                "operator": "in",
                                "value": [
                                    "admin"
                                ]
                            }
                        },
                        "action": {
                            "operator": "in",
                            "value": [
                                "login"
                            ]
                        },
                        "channel": {
                            "operator": "in",
                            "value": [
                                "web"
                            ]
                        }
                    },
                    "result": {
                        "effect": "allow",
                        "reAuthenticate": "true",
                        "obligations": [
                            "password"
                        ],
                        "msg": "This policy is for login in mobile channel"
                    }
                }
            ]
        }
    ]
}
```

2. The following is the response body of the "add" operation where a new rule "1430025f-28bd-4a32-bb4d-2cde75e53b07" has been added to the existing policy.

```
{
    "policyId": "9981a795-eebf-4cb7-b9a7-629becd45d28",
    "policyName": "TestingPatch123456",
    "description": "Feature Policy Demo client",
    "status": "inactive",
```

```
    "matchAnyApp": true,
    "rules": [
        {
            "id": "83402db8-6fa3-4c1f-afd3-5906103c4258",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "lib",
                            "hr"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "adam"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "adam",
                        "hr",
                        "web",
                        "mobile"
                    ]
                },
                "idpName": {
                    "operator": "in",
                    "value": [
                        "adam"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "FIDO:2"
                ],
                "msg": "This condition is for testing Continuous authentication on the Web Channel. Login
with nbruce and Select Commercial Banking to invoke this Condition."
            }
        },
        {
            "id": "1430025f-28bd-4a32-bb4d-2cde75e53b07",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
```

```
                            "lib"
                        ]
                    },
                    "user": {
                        "operator": "in",
                        "value": [
                            "admin"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "login"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "true",
                "obligations": [
                    "password"
                ],
                "msg": "This policy is for login in mobile channel"
            }
        }
    ],
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
        "principalName": "defaulttenantclient"
    },
    "createdDate": "2020-10-07 13:11:18",
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
        "principalName": "defaulttenantclient"
    },
    "updatedDate": "2020-10-09 07:25:09"
}
```


Example 2: "replace" patch operation

1. The following code snippet displays the patch payload to be used for a "replace" operation in Authentication policy.

   NOTE
   Perform a GET operation to retrieve the Authentication Policy before executing the "replace" operation. In this example, we have assumed that the retrieved policy is the same as the response payload of the previous policy.

```
{
    "Operations": [
        {
            "op": "replace",
            "path": "/rules",
            "value": [
                {
                    "conditions": {
                        "principal": {
                            "group": {
                                "operator": "in",
                                "value": [
                                    "lib"
                                ]
                            },
                            "user": {
                                "operator": "in",
                                "value": [
                                    "admin"
                                ]
                            }
                        },
                        "action": {
                            "operator": "in",
                            "value": [
                                "login"
                            ]
                        },
                        "channel": {
                            "operator": "in",
                            "value": [
                                "web"
                            ]
                        }
                    },
                    "result": {
                        "effect": "allow",
                        "reAuthenticate": "true",
                        "obligations": [
                            "password"
                        ],
                        "msg": "This policy is for login in mobile channel"
                    }
                }
            ]
        }
    ]
}
```

2. The following is the response body of the "replace" operation where the "rules" which previously had multiple rule id's have been replaced with a single rule.

```
{
    "policyId": "9981a795-eebf-4cb7-b9a7-629becd45d28",
```

```
"policyName": "TestingPatch123456",
"description": "Feature Policy Demo client",
"status": "inactive",
"matchAnyApp": true,
"rules": [
    {
        "id": "f29ae5cb-3709-4f47-aa78-cde8a2217b97",
        "conditions": {
            "principal": {
                "group": {
                    "operator": "in",
                    "value": [
                        "lib"
                    ]
                },
                "user": {
                    "operator": "in",
                    "value": [
                        "admin"
                    ]
                }
            },
            "action": {
                "operator": "in",
                "value": [
                    "login"
                ]
            },
            "channel": {
                "operator": "in",
                "value": [
                    "web"
                ]
            }
        },
        "result": {
            "effect": "allow",
            "reAuthenticate": "true",
            "obligations": [
                "password"
            ],
            "msg": "This policy is for login in mobile channel"
        }
    }
],
"createdBy": {
    "principalType": "CLIENT",
    "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
    "principalName": "defaulttenantclient"
},
"createdDate": "2020-10-07 13:11:18",
"updatedBy": {
    "principalType": "CLIENT",
```

```
            "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
            "principalName": "defaulttenantclient"
        },
        "updatedDate": "2020-10-09 07:28:33"
    }
```

Example 3: "remove" patch operation

NOTE
Perform a GET operation to retrieve the Authentication Policy. In this example, the GET however, retrieves the AuthN policy with a single rule. Therefore, it restricts you from performing a "remove" operation. Note that it is mandatory to have at least one rule in a policy.

Format 2-("/rules/[id eq \"<RULEID>\"]")

Example 1: "replace" patch operation

1. In this format ("path": "/rules/[id eq \"<RULEID>\"]"), you can perform "replace" and "remove" operations but you are not allowed to perform "add" operation. The following code snippet displays the patch payload to be used for a "replace" operation in Authentication policy.

   NOTE
   Perform a GET operation to retrieve the Authentication Policy before executing the "replace" operation. In this example, we have assumed that the retrieved policy is the same as the response payload of the previous policy.

```
{
    "Operations": [
        {
            "op": "replace",
            "path": "/rules/[id eq \"f29ae5cb-3709-4f47-aa78-cde8a2217b97\"]",
            "value": {
                "conditions": {
                    "principal": {
                        "group": {
                            "operator": "in",
                            "value": [
                                "lib",
                                "hr"
                            ]
                        },
                        "user": {
                            "operator": "in",
                            "value": [
                                "admin"
                            ]
                        }
                    },
                    "action": {
                        "operator": "in",
                        "value": [
                            "login",
                            "otp"
                        ]
                    },
                    "channel": {
                        "operator": "in",
```

```
                        "value": [
                            "web",
                            "mobile"
                        ]
                    }
                },
                "result": {
                    "effect": "allow",
                    "reAuthenticate": "true",
                    "obligations": [
                        "password"
                    ],
                    "msg": "This policy is replaced"
                }
            }
        }
    ]
}
```

2. The following is the response body of the "replace" operation where the value of the rule "f29ae5cb-3709-4f47-aa78-cde8a2217b97" has been replaced with the following ruleId.

```
{
    "policyId": "9981a795-eebf-4cb7-b9a7-629becd45d28",
    "policyName": "TestingPatch123456",
    "description": "Feature Policy Demo client",
    "status": "inactive",
    "matchAnyApp": true,
    "rules": [
        {
            "id": "f29ae5cb-3709-4f47-aa78-cde8a2217b97",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "lib",
                            "hr"
                        ]
                    },
                    "user": {
                        "operator": "in",
                        "value": [
                            "admin"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "login"
                    ]
                },
                "channel": {
```

```
                    "operator": "in",
                    "value": [
                        "web",
                        "mobile"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "true",
                "obligations": [
                    "password",
                    "otp"
                ],
                "msg": "This policy is replaced"
            }
        }
    ],
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
        "principalName": "defaulttenantclient"
    },
    "createdDate": "2020-10-07 13:11:18",
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
        "principalName": "defaulttenantclient"
    },
    "updatedDate": "2020-10-09 07:39:53"
}
```

Example 2: "remove" patch operation

1. The following code snippet displays the patch payload to be used for a "remove" operation in Authentication policy.

   NOTE

   - Perform a GET operation to retrieve the Authentication Policy before executing the "remove" operation. In this example, we have assumed that the retrieved policy is the same as the response payload of the previous policy.
   - "remove" operation is not allowed if there is a single rule in a policy.

```
{
    "Operations": [
        {
            "op": "remove",
            "path": "/rules/[id eq \"f29ae5cb-3709-4f47-aa78-cde8a2217b97\"]"

        }
    ]
}
```

2. The following is the response body of the "remove" operation where the "/rules/[id eq \"f29ae5cb-3709-4f47-aa78-cde8a2217b97\"]" has been removed.

```
{
    "policyId": "9981a795-eebf-4cb7-b9a7-629becd45d28",
```

```
    "policyName": "TestingPatch123456",
    "description": "Feature Policy Demo client",
    "status": "inactive",
    "matchAnyApp": true,
    "rules": [
        {
            "id": "83402db8-6fa3-4c1f-afd3-5906103c4258",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "lib",
                            "hr"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "shiny"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "shiny",
                        "hr",
                        "web",
                        "mobile"
                    ]
                },
                "idpName": {
                    "operator": "in",
                    "value": [
                        "shiny"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "FIDO:2"
                ],
                "msg": "This condition is for testing Continuous authentication on the Web Channel. Login
with nbruce and Select Commercial Banking to invoke this Condition."
            }
        },
        {
            "id": "1430025f-28bd-4a32-bb4d-2cde75e53b07",
            "conditions": {
                "principal": {
```

```
                    "group": {
                        "operator": "in",
                        "value": [
                            "lib"
                        ]
                    },
                    "user": {
                        "operator": "in",
                        "value": [
                            "admin"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "login"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "true",
                "obligations": [
                    "password"
                ],
                "msg": "This policy is for login in mobile channel"
            }
        }
    ],
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
        "principalName": "defaulttenantclient"
    },
    "createdDate": "2020-10-07 13:11:18",
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
        "principalName": "defaulttenantclient"
    },
    "updatedDate": "2020-10-09 07:25:09"
}
```


Format 3 - "path": "/rules/[id eq \"<RULEID>\"]/<ABSOLUTE PATH OF RULES ATTRIBUTE>"

In this format you can patch the following attributes:

group (Type- String & Array), user (Type- String & Array), action (Type- String & Array), channel (Type- String & Array), idpName (Type- String & Array), Ipaddress (Type- String & Array), riskThreshold, riskThreshold operator, Result - obligations, effect , msg, Reauthenticate, mfaFrequency, apps, policyName, description, status

Example 1: patch operation to "add" a riskThreshold condition in "rules"

1. The following code snippet displays the patch payload to "add" riskThreshold as a condition in the rules.

   NOTE
   Perform a GET operation to retrieve the Authentication Policy before executing the "add" operation. In this example, we have assumed that the retrieved policy is the same as the response payload of the previous policy.

```
{
    "Operations": [
        {
            "op": "add",
            "path": "/rules/[id eq \"1e60fbfa-9161-4ec9-b70d-db6bbc84b4c7\"]/conditions/riskThreshold",
            "value": 65
        }
    ]
}
```

2. The following is the response body of the "add" operation where the riskThreshold condition has been added. By default, the "ge" operator gets added, however, you can modify the operator by performing a patch on it. See the next example to know more.

```
{
    "policyId": "9981a795-eebf-4cb7-b9a7-629becd45d28",
    "policyName": "TestingPatch123456",
    "description": "Feature Policy Demo client",
    "status": "inactive",
    "matchAnyApp": true,
    "rules": [
        {
            "id": "1e60fbfa-9161-4ec9-b70d-db6bbc84b4c7",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "lib"
                        ]
                    },
                    "user": {
                        "operator": "in",
                        "value": [
                            "admin"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "login"
                    ]
                },
```

```
                    "channel": {
                        "operator": "in",
                        "value": [
                            "web"
                        ]
                    },
                    "riskThreshold": {
                        "operator": "ge",
                        "value": 65
                    }
                },
                "result": {
                    "effect": "allow",
                    "reAuthenticate": "true",
                    "obligations": [
                        "password"
                    ],
                    "msg": "This policy is for login in mobile channel"
                }
            }
        ],
        "createdBy": {
            "principalType": "CLIENT",
            "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
            "principalName": "defaulttenantclient"
        },
        "createdDate": "2020-10-07 13:11:18",
        "updatedBy": {
            "principalType": "CLIENT",
            "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
            "principalName": "defaulttenantclient"
        },
        "updatedDate": "2020-10-09 09:22:38"
    }
```


Example 2: patch operation to "replace" an "operator"

1. The following code snippet displays the patch payload to "replace" an operator

   NOTE
   Perform a GET operation to retrieve the Authentication Policy before executing the "replace" operation. In this example, we have assumed that the retrieved policy is the same as the response payload of the previous policy.


```
{
    "Operations": [
        {
            "op": "replace",
            "path": "/rules/[id eq \"1e60fbfa-9161-4ec9-b70d-db6bbc84b4c7\"]/conditions/riskThreshold/
operator",
            "value": "gt"
        }
    ]
}
```

2. The following is the response body of the "replace" operation where the operator value has been replaced from "ge" to "gt".


```
{
    "policyId": "9981a795-eebf-4cb7-b9a7-629becd45d28",
    "policyName": "TestingPatch123456",
    "description": "Feature Policy Demo client",
    "status": "inactive",
    "matchAnyApp": true,
    "rules": [
        {
            "id": "1e60fbfa-9161-4ec9-b70d-db6bbc84b4c7",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "lib"
                        ]
                    },
                    "user": {
                        "operator": "in",
                        "value": [
                            "admin"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "login"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web"
                    ]
                },
                "riskThreshold": {
                    "operator": "gt",
                    "value": 65
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "true",
                "obligations": [
                    "password"
                ],
                "msg": "This policy is for login in mobile channel"
            }
        }
    ],
```

```
        "createdBy": {
            "principalType": "CLIENT",
            "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
            "principalName": "defaulttenantclient"
        },
        "createdDate": "2020-10-07 13:11:18",
        "updatedBy": {
            "principalType": "CLIENT",
            "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
            "principalName": "defaulttenantclient"
        },
        "updatedDate": "2020-10-09 09:25:06"
    }
```

Example 3: patch operation to "add" an "app"

1. The following code snippet displays the patch payload to "add" an "app".

   NOTE
   Perform a GET operation to retrieve the Authentication Policy before executing the "add" operation.

```
{
    "Operations": [
        {
            "op": "add",
            "path": "/apps",
            "value": [
                {
                    "id": "be7ab129-aa46-4b3d-96f9-fed48602793b"
                }
            ]
        }
    ]
}
```

2. The following is the response body of the "add" operation where a new application "id": "be7ab129-aa46-4b3d-96f9-fed48602793b" has been added.

```
{
    "policyId": "9981a795-eebf-4cb7-b9a7-629becd45d28",
    "policyName": "TestingPatch123456",
    "description": "Feature Policy Demo client",
    "status": "inactive",
    "matchAnyApp": false,
    "apps": [
        {
            "name": "DemoClient",
            "id": "be7ab129-aa46-4b3d-96f9-fed48602793b"
        }
    ],
    "rules": [
        {
            "id": "1e60fbfa-9161-4ec9-b70d-db6bbc84b4c7",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
```

```
                        "value": [
                            "lib"
                        ]
                    },
                    "user": {
                        "operator": "in",
                        "value": [
                            "admin"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "login"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web"
                    ]
                },
                "riskThreshold": {
                    "operator": "gt",
                    "value": 65
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "true",
                "obligations": [
                    "password"
                ],
                "msg": "This policy is for login in mobile channel"
            }
        }
    ],
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
        "principalName": "defaulttenantclient"
    },
    "createdDate": "2020-10-07 13:11:18",
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
        "principalName": "defaulttenantclient"
    },
    "updatedDate": "2020-10-09 09:27:44"
}
```

Example 4: patch operation to "replace" an "app"

1. The following code snippet displays the patch payload to "replace" an "app".

   NOTE
   Perform a GET operation to retrieve the Authentication Policy before executing the "replace" operation. In this example, we have assumed that the retrieved policy is the same as the response payload of the previous policy.

```
{
    "Operations": [
        {
            "op": "replace",
            "path": "/apps/[id eq \"be7ab129-aa46-4b3d-96f9-fed48602793b\"]",
            "value": "b4e0b7e4-36c6-43e6-97e2-df144a0a64d6"
        }
    ]
}
```

2. The following is the response body of the "replace" operation where "/apps/[id eq \"be7ab129-aa46-4b3d-96f9-fed48602793b\"]" has been replaced with a new value.

```
{
    "policyId": "9981a795-eebf-4cb7-b9a7-629becd45d28",
    "policyName": "TestingPatch123456",
    "description": "Feature Policy Demo client",
    "status": "inactive",
    "matchAnyApp": false,
    "apps": [
        {
            "name": "DefaultTenantClient",
            "id": "b4e0b7e4-36c6-43e6-97e2-df144a0a64d6"
        }
    ],
    "rules": [
        {
            "id": "1e60fbfa-9161-4ec9-b70d-db6bbc84b4c7",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "lib"
                        ]
                    },
                    "user": {
                        "operator": "in",
                        "value": [
                            "admin"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "login"
                    ]
```

```
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web"
                    ]
                },
                "riskThreshold": {
                    "operator": "gt",
                    "value": 65
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "true",
                "obligations": [
                    "password"
                ],
                "msg": "This policy is for login in mobile channel"
            }
        }
    ],
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
        "principalName": "defaulttenantclient"
    },
    "createdDate": "2020-10-07 13:11:18",
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
        "principalName": "defaulttenantclient"
    },
    "updatedDate": "2020-10-09 09:36:00"
}
```

Example 5: patch operation to "remove" an "app"

1. The following code snippet displays the patch payload to remove "/apps/[id eq \"b4e0b7e4-36c6-43e6-97e2-df144a0a64d6\"]".

   NOTE
   Perform a GET operation to retrieve the Authentication Policy before executing the "remove" operation. In this example, we have assumed that the retrieved policy is the same as the response payload of the previous policy.

```
{
    "Operations": [
        {
            "op": "remove",
            "path": "/apps/[id eq \"b4e0b7e4-36c6-43e6-97e2-df144a0a64d6\"]"
        }
    ]
}
```

2. The following is the response body of the "remove" operation where the "/apps/[id eq \"b4e0b7e4-36c6-43e6-97e2-df144a0a64d6\"]" has been removed from the policy.


```
{
    "policyId": "9981a795-eebf-4cb7-b9a7-629becd45d28",
    "policyName": "TestingPatch123456",
    "description": "Feature Policy Demo client",
    "status": "inactive",
    "matchAnyApp": true,
    "rules": [
        {
            "id": "1e60fbfa-9161-4ec9-b70d-db6bbc84b4c7",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "lib"
                        ]
                    },
                    "user": {
                        "operator": "in",
                        "value": [
                            "admin"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "login"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web"
                    ]
                },
                "riskThreshold": {
                    "operator": "gt",
                    "value": 65
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "true",
                "obligations": [
                    "password"
                ],
                "msg": "This policy is for login in mobile channel"
            }
        }
    ],
```

```
        "createdBy": {
            "principalType": "CLIENT",
            "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
            "principalName": "defaulttenantclient"
        },
        "createdDate": "2020-10-07 13:11:18",
        "updatedBy": {
            "principalType": "CLIENT",
            "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
            "principalName": "defaulttenantclient"
        },
        "updatedDate": "2020-10-09 09:37:08"
    }
```


Example 6: patch operation to "replace" a "description"

1. The following code snippet displays the patch payload to update "/description" of as existing policy.

   NOTE
   Perform a GET operation to retrieve the Authentication Policy before executing the "remove" operation. In this example, we have assumed that the retrieved policy is the same as the response payload of the previous policy.


```
{
    "Operations": [
        {
            "op": "replace",
            "path": "/description",
            "value": "testing policy"
        }
    ]
}
```


2. The following is the response body of the "replace" operation where the "/description" of the policy has changed to -"testing policy".


```
{
    "policyId": "9981a795-eebf-4cb7-b9a7-629becd45d28",
    "policyName": "testing policy",
    "description": "testing policy",
    "status": "inactive",
    "matchAnyApp": true,
    "rules": [
        {
            "id": "1e60fbfa-9161-4ec9-b70d-db6bbc84b4c7",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "lib"
                        ]
                    },
                    "user": {
                        "operator": "in",
                        "value": [
                            "admin"
```

```
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "login"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web"
                    ]
                },
                "riskThreshold": {
                    "operator": "gt",
                    "value": 65
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "true",
                "obligations": [
                    "password"
                ],
                "msg": "This policy is for login in mobile channel"
            }
        }
    ],
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
        "principalName": "defaulttenantclient"
    },
    "createdDate": "2020-10-07 13:11:18",
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
        "principalName": "defaulttenantclient"
    },
    "updatedDate": "2020-10-09 09:37:08"
}
```


Example 7: patch operation to "replace" an "obligation"

1. The following code snippet displays the patch payload to replace the value of existing "obligations"- "/rules/[id eq \"1e60fbfa-9161-4ec9-b70d-db6bbc84b4c7\"]/result/obligations" with a new value.

   NOTE
   Perform a GET operation to retrieve the Authentication Policy before executing the "remove" operation. In this example, we have assumed that the retrieved policy is the same as the response payload of the previous policy.


```
{
```

```
    "Operations": [
        {
            "op": "replace",
            "path": "/rules/[id eq \"1e60fbfa-9161-4ec9-b70d-db6bbc84b4c7\"]/result/obligations"
            "value": "SMSOTP:1"
        }
    ]
}
```

2. The following is the response body of the "replace" operation where the "obligations"- "/rules/[id eq \"1e60fbfa-9161-4ec9-b70d-db6bbc84b4c7\"]/result/obligations" value has been replaced with "SMSOTP:1" in the policy.

```
{
    "policyId": "9981a795-eebf-4cb7-b9a7-629becd45d28",
    "policyName": "testing policy",
    "description": "Feature Policy Demo client",
    "status": "inactive",
    "matchAnyApp": true,
    "rules": [
        {
            "id": "1e60fbfa-9161-4ec9-b70d-db6bbc84b4c7",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "lib"
                        ]
                    },
                    "user": {
                        "operator": "in",
                        "value": [
                            "admin"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "login"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web"
                    ]
                },
                "riskThreshold": {
                    "operator": "gt",
                    "value": 65
                }
            },
```

```
            "result": {
                "effect": "allow",
                "reAuthenticate": "true",
                "obligations": [
                    "SMSOTP:1"
                ],
                "msg": "This policy is for login in mobile channel"
            }
        }
    ],
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
        "principalName": "defaulttenantclient"
    },
    "createdDate": "2020-10-07 13:11:18",
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "09ad28ca-b8a5-4688-89a8-8681a0b37ca2",
        "principalName": "defaulttenantclient"
    },
    "updatedDate": "2020-10-09 09:37:08"
}
```


Example 8: patch operation to "replace" "mfaFrequency"

1. The following code snippet displays the patch payload to replace the value of existing "mfaFrequency"- "/rules/[id eq \"cd219805-64b5-4a85-86a5-4c7a9f693589\"]/result/mfaFrequency" with a new value.

   NOTE
   Perform a GET operation to retrieve the Authentication Policy before executing the "remove" operation.


```
{
    "Operations": [
        {
            "op": "replace",
            "path": "/rules/[id eq \"cd219805-64b5-4a85-86a5-4c7a9f693589\"]/result/mfaFrequency",
            "value": "EveryTime"
        }
    ]
}
```


2. The following is the response body of the "replace" operation where the "mfaFrequency"- "/rules/[id eq \"cd219805-64b5-4a85-86a5-4c7a9f693589\"]/result/mfaFrequency" value has been replaced with "OnceForTrustedDevice" in the policy.


```
{
    "policyId": "0481ee2d-6e0c-47fe-8ec0-51dffd42d3f2",
    "policyName": "Sample policy",
    "description": "Sample policy Record mfaFrequency",
    "status": "active",
    "matchAnyApp": true,
    "rules": [
        {
            "id": "cd219805-64b5-4a85-86a5-4c7a9f693589",
            "conditions": {
                "principal": {
```

```
                    "group": {
                        "operator": "in",
                        "value": [
                            "HR"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "login"
                    ]
                },
                "ipAddress": {
                    "operator": "in",
                    "value": [
                        "10.0.0.0/24"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "mobile"
                    ]
                },
                "riskThreshold": {
                    "operator": "ge",
                    "value": 60
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD"
                ],
                "msg": "This policy is for login in mobile channel",
               "mfaFrequency": "OnceForTrustedDevice"
            }
        }
    ],
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "2231b8eb-94a0-4f6a-b1ba-3b6dcbcd205f",
        "principalName": "defaulttenantclient"
    },
    "createdDate": "2021-03-08 05:42:08"
}
```


You can perform "add" and "remove" operation on "mfaFrequency". The following are the samples of "add" and "remove" patch operation.

Format to add the attribute "mfaFrequency"


```
{
```

```
    "Operations": [
        {
            "op": "add",
            "path": "/rules/[id eq \"cd219805-64b5-4a85-86a5-4c7a9f693589\"]/result/mfaFrequency",
            "value": "EveryTime"
        }
    ]
}
```

Format to remove the attribute "mfaFrequency"

```
{
    "Operations": [
        {
            "op": "remove",
            "path": "/rules/[id eq \"cd219805-64b5-4a85-86a5-4c7a9f693589\"]/result/mfaFrequency"
        }
    ]
}
```

Example 9: "replace" patch operation performed on a given index of an attribute (Type-String)

In this sample, the "replace" operation is performed to replace the value of the "group" string which is positioned at index "0" with the value "accounts". The following code snippet displays the patch payload to be used for the "replace" operation.

> **NOTE**
> If the index is not mentioned at the end of the path, the "0"th index is taken by default.

```
{
    "Operations": [
        {
            "op": "replace",
            "path": "/rules/[id eq \"cd219805-64b5-4a85-86a5-4c7a9f693589\"]/conditions/principal/group/0",
            "value": "accounts"
        }
    ]
}
```

The following is the response body of the "replace" operation where the value of the "group" as mentioned in the path has been replaced with the value, "accounts".

```
    "rules": [
        {
            "id": "cd219805-64b5-4a85-86a5-4c7a9f693589",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                             "accounts",
                             "lib",
                            "edu",
                            "admin"
                        ]
```

```
                    },
                    "clientApp": {
                        "operator": "in",
                        "value": [
                            "fb12da11-e48b-4cb9-a03a-e455231b5a13"
                        ]
                    }
                }
            }
```

Example 10: "replace" patch operation performed on a given index of an attribute (Type Array)

In this sample, the "replace" operation is performed to replace the value of the "group" array which is positioned at index "1" with the value "accounts". The following code snippet displays the patch payload to be used for the "replace" operation.

NOTE
If the index is not mentioned at the end of the path, the "0"th index is taken by default.

```
{
    "Operations": [
        {
            "op": "replace",
            "path": "/rules/[id eq \"1b0b439d-270a-4471-975c-6ac99ae9b539\"]/conditions/principal/group",
            "value": [
                "fin",
                "dev"
            ]
        }
    ]
}
```

The following is the response body of the "replace" operation where the value of the entire "group" as mentioned in the path has been replaced with the values, "fin" and "dev".

```
    "rules": [
        {
            "id": "1b0b439d-270a-4471-975c-6ac99ae9b539",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "fin",
                            "dev"
                            ]
                    },
                    "clientApp": {
                        "operator": "in",
                        "value": [
                            "fb12da11-e48b-4cb9-a03a-e455231b5a13"
                        ]
                    }
```

```
                }
            }
```

Example 11: "remove" patch operation performed on a given index of an attribute

In this sample, the "remove" operation is performed to remove the value ("accounts") of the "group" which is positioned at index "1" of the array. The following code snippet displays the patch payload to be used for the "remove" operation.

```
{
    "Operations": [
        {
            "op": "remove",
            "path": "/rules/[id eq \"1b0b439d-270a-4471-975c-6ac99ae9b539\"]/conditions/principal/group/1"

        }
    ]
}
```

NOTE

- *In a "remove"* operation, you do not have to specify the value of the attribute.
- If the index is not mentioned at the end of the path, then the entire attribute is remove

The following is the response body of the "remove" operation where the value of the "group" as mentioned in the path has been removed.

```
    "rules": [
        {
            "id": "1b0b439d-270a-4471-975c-6ac99ae9b539",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "fin"
                            ]
                    },
                    "clientApp": {
                        "operator": "in",
                        "value": [
                            "fb12da11-e48b-4cb9-a03a-e455231b5a13"
                        ]
                    }
                }
            }
```

Other Patch Formats

The following examples show the replace operation for all the attributes.

Format to replace the attribute "PolicyName"

```
{
    "Operations": [
        {
            "op": "replace",
            "path": "/policyName",
```

```
            "value": "policy for testing"
        }
    ]
}
```

Format to replace the attribute "Effect"

```
{
    "Operations": [
        {
            "op": "replace",
            "path": "/rules/[id eq \"3932c847-27c7-40d2-af2a-5184ffe6b3cc\"]/result/effect",
            "value": "deny"
        }
    ]
}
```

Format to replace the attribute "reAuthenticate"

```
{
    "Operations": [
        {
            "op": "replace",
            "path": "/rules/[id eq \"3932c847-27c7-40d2-af2a-5184ffe6b3cc\"]/result/reAuthenticate",
            "value": "true"
        }
    ]
}
```

Format to replace the attribute "channel"

NOTE
The following example shows replace operation for channel. However, you can use the same format for "add" and "remove" operations.

```
{
    "Operations": [
        {
            "op": "replace",
            "path": "/rules/[id eq \"d1aeead8-10df-480a-b5c4-9f24bca46042\"]/conditions/principal/channel",
            "value": "mobile"
        }
    ]
}
```

Format to replace the attribute "action"

NOTE
The following example shows replace operation for action. However, you can use the same format for "add" and "remove" operations.

```
{
    "Operations": [
        {
            "op": "replace",
            "path": "/rules/[id eq \"d1aeead8-10df-480a-b5c4-9f24bca46042\"]/conditions/principal/action",
            "value": "authenticate"
```

```
        }
    ]
}
```

Format to replace the attribute "user"

NOTE
The following example shows replace operation for user. However, you can use the same format for "add" and "remove" operations.

```
{
    "Operations": [
        {
            "op": "replace",
            "path": "/rules/[id eq \"d1aeead8-10df-480a-b5c4-9f24bca46042\"]/conditions/principal/user",
            "value": "Nadia Kite"
        }
    ]
}
```

Expressions Supported in Authentication Policies

Learn the standard format of expressions supported in Symantec Identity Security Platform that can be integrated into Attribute Based Access Control (ABAC) policy definitions.

This section outlines the standardized format of expressions supported in Authentication Policies.

- Simple Expression
- Complex Expression
- Expressions Used in Context

Simple Expression

Expression format: `EXPRESSION = (LEFT-VALUE OPERATOR RIGHT-VALUE)`

Table 109: Expression Table

Expression Element	Possible Values	Examples
Left-Value	• USER ATTRIBUTE • CONTEXT ATTRIBUTES	• USER ATTRIBUTE examples: – user_loginid – email – custom-attribute "rallyId" etc. • CONTEXT ATTRIBUTES examples: – ${appMetadata.env} – ${idpMetadata.department} – ${idProfile.sub} – ${idToken.sub} – ${clientContext} To know more about the supported namespaces, such as idProfile, idToken, appMetadata, and clientContext, see Custom Token Claims.
Operator	• EQ/eq: Equals • NE/ne: Not equals • GT/gt: Greater than • LT/lt: Less than • GE/ge: Greater than or Equals • LE/le: Less than or equals	
Right-Value	• STATIC • DYNAMIC SUBSTITUTION • WILDCARDED VALUES	• STATIC examples: – us – IMS – 10 • DYNAMIC SUBSTITUTION examples: – ${appMetadata.env} – ${idtoken.region} • WILDCARDED VALUES examples: – *@EXAMPLE.com, – *@*.saepqa.xyz* "*" is a special character and treated as a wildcard matching 0+ characters.

Examples of Simple Expressions:

- (email ne *@example.com)
- (${clientcontext.env} eq ${appmetadata.env})

To know more about the supported namespaces, such as idProfile, idToken, appMetadata,and clientContext, see Custom Token Claims.

Complex Expression built from Simple Expressions

Expression format:

```
EXPRESSION = (EXPRESSION OPERATOR EXPRESSION)
```

Supported OPERATOR:

```
AND, OR
```

Examples of Complex Expressions:

- ((email ne *@broadcom.com) and (user_loginid ne *@broadcom.net) and (group ne 00g1g3dsq9x59eZ641d8))
- ((${clientcontext.env} eq ${appmetadata.env}) and (${appmetadata.department} eq ${clientcontext.department}))

Using different contexts as expression in “context” policy condition

Table 110: Using different contexts as expression in “context” policy condition

Scenario	Context	Expression used in context condition
Policy should match when the • "environment" attribute of application metadata is equal to "development"	• appmetadata	(${appmetadata.environment} eq development)
Policy should match when the • "environment" attribute of application metadata is equal to "development" AND • "department" attribute of application metadata is equal to "ims"	• appmetadata	(${appmetadata.environment} eq development) and (${appmetadata.department} eq ims)
Policy should match when the • "environment" attribute of application metadata is equal to "development" OR • "department" attribute of application metadata is equal to "ims"	• appmetadata	(${appmetadata.environment} eq development) or (${appmetadata.department} eq ims)
Policy should match when • "environment" attribute of application metadata is equal to clientcontext attribute "env"	• appmetadata • clientcontext	(${appmetadata.environment} eq ${clientcontext.env})
Policy should match when • "environment" attribute of application metadata is equal to "clientcontext attribute "env" AND • "region" attribute of application metadata is equal to idprofile attribute "location"	• appmetadata • clientcontext • idprofile	(${appmetadata.environment} eq ${clientcontext.env}) and (${appmetadata.region} eq ${idprofile.location})
Policy should match when • "environment" attribute of application metadata is equal to "clientcontext attribute "env" AND • "region" attribute of application metadata is equal to idtoken attribute "location"	• appmetadata • clientcontext • idtoken	(${appmetadata.environment} eq ${clientcontext.env}) and (${appmetadata.region} eq ${idtoken.location})
Policy should match when the • "environment" attribute of Identity Provider metadata is equal to clientcontext attribute "env"	• idpmetadata • clientcontext	(${idpmetadata.environment} eq ${clientcontext.env})
Policy should match when the • "vanity host" used to access is auth.example.com	• system	(${system.host} eq auth.example.com)

Scenario	Context	Expression used in context condition
Policy should match when the • "issuerurl" is https://auth.example.com/default	• system	(${system.issuerurl} eq https://auth.example.com/default)

Setting Priority in Authentication Policy and Rule

This topic explains how setting priority for authentication policies and rules ensures only prioritized ones influence the final result (Next Action).

This topic includes the following sections:

- Overview of Policy and Rule Priority
- How Priorities Work in Policy Evaluation
- Matching Logic: Priority vs. Non-Priority
- Handling Conflicts: Allow vs. Deny
- Understanding Priority with Decimals
- Policy API payload with Priority
- Common Use Cases

Overview of Policy and Rule Priority

Priority is a numerical value assigned to a policy to control its evaluation order. Lower numbers represent **higher priority** (for example, `1` is higher than `100`). Rule Priority is a numerical value assigned to individual rules within a policy. Like policies, lower numbers mean higher importance or precedence.

Priority can be set at both the policy and rule levels, ranging from 1 (highest) to 1,000,000 (lowest). Decimals can be used for finer control. Multiple policies and rules can share the same priority. If two or more matched rules have the same priority at both the policy and rule levels, the final result is a combination of all the matched rules.

How Priorities Work in Policy Evaluation

- During policy evaluation, all policies regardless of whether they have priorities or not are considered. However, if there are matching policies that have priorities defined, only those policies with defined priorities and matched policies are used to determine the final result (Next Action). In other words, when policies with priority and non-priority matches, only the policies with priority influences the final result.
- During policy evaluation, all policies are evaluated, but if no policies with priority match the defined conditions, then the matched policies without priority are used to determine the final result (Next Action). In other words, if no policies or rules with priorities are applicable, the final outcome includes all obligations from the matched policies and rules without priorities.
- A policy with its priority set to zero (0) is effectively deactivated and is not considered when determining the final result.

Refer the next section understand the same with examples.

Matching Logic: Priority vs. Non-Priority

When both prioritized and non-prioritized policies and rules are present, the evaluation logic first considers those with defined priorities. If no matching conditions are found among them, it then evaluates the policies and rules without priorities.

Refer to the following example for better understanding.

Policy Details	Policy Condition Match	Obligation
Policy 1 with "policyPriority": 1 and "rulePriority": 1	No policy conditions match	SMSOTP
Policy 2 with "policyPriority": none and "rulePriority": 1	No policy conditions match	PASSWORD
Policy 3 with "policyPriority": none and "rulePriority": none	Yes, policy conditions match	EMAILOTP

Policy Evaluation Outcome:

- Although Policies 1 and 2 include priority values, neither of them matches the evaluation conditions. Policy 3, which has no priorities defined, is the only one with a matching condition.
- As a result, Policy 3 takes precedence and its obligation (EMAILOTP) is applied in the final result.
- When none of the policies with priorities match, the policy evaluation uses the matching policies without priorities to determine the final outcome (Next Action). In this case, `EMAILOTP` is selected because it is the only obligation from a policy that successfully matched the conditions.

Handling Conflicts: Allow vs. Deny

Priority plays a key role in determining how policies and rules are evaluated, especially when they have conflicting effects (Allow vs. Deny). The following examples illustrate different scenarios based on priority levels defined at both the policy and rule levels.

> **NOTE**
> The Authentication Policy API defines policy and rule priorities using the attributes **"policyPriority"** and **"rulePriority,"** respectively.

Use Case 1: Same Priority Level, Conflicting Effects
When multiple policies and rules share the same priority level but have conflicting "effect" (Allow and Deny), the "Deny" effect takes precedence and determines the final result (Next Action).

Policy 1 with "policyPriority": 1	"rulePriority": 1	Effect: **Allow**
Policy 2 with "policyPriority": 1	"rulePriority": 1	Effect: **Deny**

Rule Summary - Since both policies have the same priority, **Policy 2 (Deny)** takes precedence and is applied in the final result.

Use Case 2: Different Priority Levels, Conflicting Effects
When policies and rules have **different priority levels**, the policy with the **higher priority (lower number)** takes precedence, regardless of its effect.

Policy 1 with "policyPriority": 1	"rulePriority": 2	Effect: **Allow**
Policy 2 with "policyPriority": 2	"rulePriority": 1	Effect: **Deny**

Rule Summary - Since Policy 1 has the highest priority (1 is higher than 2), **Policy 1 (Allow)** takes precedence and is applied in the final result.

Use Case 3: Same Policy Priority, Different Rule Priorities
When policies have the **same priority level**, but their rules have **different priorities**, the rule with the highest priority (lowest number) takes precedence.

Policy 1 with "policyPriority": 1	"rulePriority": 2	Effect: **Allow**
Policy 2 with "policyPriority": 1	"rulePriority": 1	Effect: **Deny**

Rule Summary - Since Policy 2 has a rule with a higher priority (1 vs. 2), **Policy 2 (Deny)** takes precedence and is applied in the final result.

Understanding Priority with Decimals

The priorities can include decimals allowing for finer control over policy execution result. Smaller numerical values indicate higher priority (where 1 is the highest priority and 1,000,000 is the lowest).

The following example shows how priority with decimals work in Symantec Identity Security Platform.

Consider the following priority values as an example: `1,1.1,1.23,1.4,2,3,4,4.1,4.5`

The policy evaluation results in the following order:

- 1 has the highest priority, followed by 1.1 then 1.23, and so on.
- 1.1 takes precedence over 1.23
- 1.4 has higher priority than 4
- 4 is prioritized over 4.5

This approach ensures more granular control within the same shared priority range, ensuring policies and rules are executed in the intended sequence

Authentication Policy API payload with "policyPriority" and "rulePriority"

> **NOTE**
> The Authentication Policy API defines policy and rule priorities using the attributes **"policyPriority"** and **"rulePriority,"** respectively.

Example of a Policy API payload with priority

```
{
  "policyId": "3ad197b7-5222-449b-84b3-9e40fd03b893",
  "policyName": "Test Policy",
  "description": "Default policy for Tenant Admin Console",
  "policyPriority": 1,
  "status": "active",
  "matchAnyApp": false,
  "apps": [
    {
      "name": "DefaultTenantClient",
      "id": "3d7c77cd-bf45-4d68-a6d2-cb9607c5c521"
    },
    {
      "name": "Sample Application",
      "id": "c47936b7-dbea-4d76-acf5-f84ca4abb8e3"
    },
    {
      "name": "AdminConsole",
      "id": "$TENANT_ADMINCONSOLE"
    }
  ],
  "rules": [
    {
      "id": "6031706a-300b-4fc7-a752-c78462ae6d51",
      "conditions": {
        "principal": {
          "user": {
            "operator": "in",
            "value": [
              "*"
```

```
            ]
          }
        }
      },
      "result": {
        "effect": "allow",
        "reAuthenticate": "false",
        "obligations": [
          "PASSWORD:1"
        ],
        "msg": "Rule 2",
        "rulePriority": 2
      }
    }
  ],
  "adminscope.criteria": [],
  "adminscope.managedby": [],
  "createdBy": {
    "principalType": "CLIENT",
    "principalId": "3d7c77cd-bf45-4d68-a6d2-cb9607c5c521",
    "principalName": "DefaultTenantClient"
  },
  "createdDate": "2025-01-29 12:38 pm UTC",
  "updatedBy": {
    "principalType": "CLIENT",
    "principalId": "3d7c77cd-bf45-4d68-a6d2-cb9607c5c521",
    "principalName": "DefaultTenantClient"
  },
  "updatedDate": "2025-02-13 12:14 pm UTC"
}
```


Common Use Cases : Authentication Policy with Priority Based Policy and Rule Selection

NOTE
The Authentication Policy API defines policy and rule priorities using the attributes **"policyPriority"** and **"rulePriority,"** respectively.

Authentication Policy Use Cases			
Sl.No	**No of Policies and their Priorities**	**Matched Rules with Priorities**	**Result**
1	Policy 1 (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 1 Rule 2 with "rulePriority": 2	Obligations of **Rule: 1** will be triggered.
2	Policy 1 (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 2 Rule 2 with "rulePriority": 1	Obligations of **Rule: 2** will be triggered.
3	Policy 1 (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 1 Rule 2 with "rulePriority": 1	Obligations of both the rules **(Rule:1 and Rule: 2)** will be triggered.
4	Policy 1 with "policyPriority": 1 (has two rules)	Rule 1 with "rulePriority": 1 Rule 2 with "rulePriority": 2	Obligations of **Policy 1, Rule: 1** will be triggered.
	Policy 2 with "policyPriority":2 (has two rules)	Rule 1 with "rulePriority": 1 Rule 2 with "rulePriority": 2	

5	Policy 1 with "policyPriority":1 (has two rules)	Rule 1 with "rulePriority": 2 Rule 2 with "rulePriority": 1	Obligations of **Policy 1, Rule: 2** will be triggered.
	Policy 2 with "policyPriority": 2 (has two rules)	Rule 1 with "rulePriority": 2 Rule 2 with "rulePriority": 3	
6	Policy 1 with "policyPriority": 1 (has two rule)	Rule 1 with "rulePriority": 2 Rule 2 with "rulePriority": 1	Obligations of **Policy 1, Rule: 2** will be triggered.
	Policy 2 with "policyPriority": 2 (has two rules)	Rule 1 with "rulePriority": 2 Rule 2 with "rulePriority": 3	
	Policy 3 with "policyPriority": 3 (has one rule)	Rule 1 with "rulePriority": 1	
7	Policy 1, (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 1 Rule 2 with "rulePriority": 1.1	Obligations of **Rule: 1** will be triggered.
8	Policy 1, (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 2 Rule 2 with "rulePriority": 3	Obligations of **Policy 2, Rule: 1** will be triggered.
	Policy 2, (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 1.1 Rule 2 with "rulePriority": 2	
9	Policy 1, (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 2 Rule 2 with "rulePriority": 3	Obligations of **Policy 2, Rule: 1** will be triggered.
	Policy 2, (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 1.4 Rule 2 with "rulePriority": 2	
10	Policy 1 with "policyPriority": 1 (has no rule priority)	Rule 1 with no "rulePriority"	Obligations of **Policy 1** will be triggered.
	Policy 2 with "policyPriority": 2 (has no rule priority)	Rule 2 with no "rulePriority"	
11	Policy 1 with "policyPriority": 2 (has one rule)	Rule 1 with "rulePriority": 3	Obligations of **Policy 2, Rule: 1** will be triggered.
	Policy 2 with "policyPriority": 1 (has two rules)	Rule 1 with "rulePriority": 1.1 Rule 2 with "rulePriority": 2	
12	Policy 1 with "policyPriority": 1 (has a single rule)	Rule 1 with "rulePriority": 2	Obligations of **Policy 1, Rule: 1** will be triggered.
	Policy 2, (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 1.1 Rule 2 with "rulePriority":	

Reference Links:

- Priority configuration in IDP Discovery Policy and Rule.

Identity Provider (IDP) Discovery Policy

Learn about creating an IDP Discovery Policy, including essential attributes like rules and conditions, and using the Discovery Policy APIs to perform CRUD operations on them.

IDP Discovery Policy Overview

An IDP Discovery Policy is a set of rules and conditions that determine how Identity Providers (IDPs) are discovered and used during the authentication process within Symantec Identity Security Platform. When IDP authentication occurs through a front-channel browser in real time, the typical integration involves the use of industry-standard "Federated Identity Providers" like OpenID Connect or SAML. As part of this authentication, the knowledge about

the principal's identity may be with, or may be delegated to, an external entity responsible for handling all or some of the assurance. Authentication involves validating the user's or client's identity at a certain level of assurance which is consistent with the requirement of the application that is being accessed by the user or client.

The IDP Discovery Policy typically includes:

- ***Rules***: These rules define conditions that must be met to determine which IDP should be used for authentication. IDP discovery uses a generic Discovery Policy that consists of multiple rules, mapping action-specific context conditions to the IDPs.
- ***Policy Evaluation***: Active policies are evaluated against given context resulting in a set of **unique** and **active** *applicable providers*.
- ***User Experience***: IDP Discovery Policies can also impact the user experience by determining whether users are automatically redirected to the appropriate IDP, or the user is required to manually select their identity provider.

Overall, the IDP Discovery Policy streamlines the authentication process by ensuring that users are directed to the correct IDP based on the defined rules and conditions, ultimately enhancing security and user convenience.

This topic includes the following sections:

- IDP Discovery Policy Attributes
- Sample IDP Discovery Policy
- Configuring Application or Tenant-level Setting
- IDP Discovery Policy APIs
  - IDP Discovery Policy API Examples

IDP Discovery Policy Attributes

The details of the IDP Discovery policy attributes are described here:

- ***policyName***: The name displays the title of the policy.
- ***description***: The description details a brief about the policy.
- ***status***: The status displays whether a policy is **active** or **inactive**.
- ***rules***: Rules include **conditions** and **result**. You can specify a single condition or multiple conditions to a rule. If the specified conditions in the rule are met, then the corresponding result is applied. The supported condition categories are: **action**, **app**, **ipAddress**, **channel,** and **loginName**.
  The supported conditions are described here:
  - ***action***: The "action" condition supports two values, "***Before user enters username"*** and ***"After user enters username"***.
    - ***Before user enters username*** - This is the protocol layer that decides if an auto redirect to IDP should take place immediately ***BEFORE*** redirecting to the Flow Url (Signin). This redirect depends on the App, IP Address, and Channel configurations. For more details about this policy, see "Before user enters username" policy flow.
    - ***After user enters username*** - This is the authentication controller that decides if an auto redirect to IDP should take place ***AFTER*** the user had provided the login name. This redirect depends on App, IP Address, Channel, and Login Name configurations. For more details about this policy, see "After user enters username" policy flow.

    For more details about this condition, see action condition .
  - ***matchAnyApp***: this is a boolean property.

    - ***true***: if set to "***true"***, the policy is applied to any or all applications.
    - ***false***: if set to "***false"***, specify the application name against which the IDP Discovery policy should be evaluated.
  - ***channel***: The "channel" condition specifies the medium through which a user can connect to the IDSP. The channel condition uses "in" as the operator value. Examples of channel: web and Mobile. For more details about this condition, see channel condition .
  - ***ipAddress***: The "ipAddress" condition specifies the connectivity to the IDSP solution from a specific ipAddress or a range of ipAddresses. You can configure the "ipAddress" range in the condition. For more details about this condition, see ipAddress condition.
  - ***loginName***: The "loginName" condition specifies the login name that is entered by the user which is used to validate against the Identity Provider's list to determine the appropriate IDP selection.
    Examples:

    - ```
      "loginName": {
         "operator": "in",
            "value": [
                "*@example.com",
                "*@example.net"
            ]
         }
      ```
    - ```
      "loginName": {
         "operator": "not",
            "value": [
                "*@example.com",
                "*@example.net"
            ]
         }
      ```
    - ```
      "loginName": {
         "operator": "regex",
            "value": [
                "(?i)^.*@example\\\\.(com|net)$"*@"
            ]
         }
      ```
    - ```
      "loginName": {
         "operator": "notregex",
            "value": [
                "(?i)^.*@example\\\\.(com|net)$"*@"
            ]
         }
      ```

- ***deriveIdpFromLoginName***: In the results section of the discovery policy, the "deriveIdpFromLoginName" property specifies if an IDP is derived from the login name that is entered by a user. This is a boolean property and is defaulted to ***false***. In the case of the ***"Before user enters username*** " flow, this property is always ***false***. This property applies exclusively to the "***After user enters username***" flow.
  - when "***deriveIdpFromLoginName***" is set to ***false***, the user is presented with a list of IDPs from where the user can select, along with an option to continue with the standard "AuthN" flow.
  - when "***deriveIdpFromLoginName***" is set to ***true***, the discovery policy performs the process of extracting the domain from the user name, locating the Identity Provider (IDP) associated with that domain, and finally, redirecting to the chosen IDP.
- ***idps***: The property includes a list of Identity Provider IDs (GUIDs) which will be used for policy evaluation. The response of the POST and PUT APIs have the Identity Provider IDs (GUIDs) along with their corresponding names.
- ***msg***: This property displays a message about the policy. For example: "This policy is for login in mobile channel."

Sample IDP Discovery Policy

The following example provides a sample of an IDP Discovery (After user enters username) policy, including the relevant rules and results, for your reference.

```
{
    "policyName": "Sample policy",
    "description": "Sample IDP Discovery policy",
    "status": "active",
    "rules": [
        {
            "conditions": {
                "action": {
                    "value": "idpAuthenticate"
                },
                "matchAnyApp": true,
                "app": {
                    "operator": "in",
                    "value": [
                        {
                            "id": "2833d52b-10aa-41a7-bf0d-06139e1b337a"
                        }
                    ]
                },
                "loginName": {
                    "operator": "in",
                    "value": [
                        "*"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "Corporate channel"
                    ]
                }
            },
            "result": {
                "idps": [
                    {
                        "id": "c3b2f037-4b3b-4b0f-b20c-60460556XXXX"
                    },
                    {
                        "id": "997349a0-9bfc-4024-82d5-e0d92ea3ZZZZ"
                    }
                ],
                "msg": "Sample policy",
                "deriveIdpFromLoginName": true
            }
        }
    ]
}
```

Configuring Application or Tenant-level Setting

To enable the identity provider discovery flow, you should configure the following property at either the application or tenant level within the solution.

Property Name	Property Scope	Property Description	Default Value	Comments
idpDiscoveryEnabled	Configured either in the Application, or in the Tenant settings.	This property determines whether the Identity Discovery policy should undergo evaluation or not. • If configured as "***true***" then IDP Discovery policy evaluation takes place. • If configured as "***false***" then IDP Discovery policy evaluation will not take place. • (This option is **ONLY** available in App Config) If configured as "***tenant***" in the application, then IDP Discovery policy evaluation will take precedence over the tenant setting configuration.	false	Possible values: true, false, or tenant
adminConsoleIdpDiscoveryEnabled	Configured ONLY in the IDSP Admin Console Application's settings.	This property determines whether the Identity Discovery policy should be assessed for the AdminConsole Application or not.	false	Possible values: true or false

The relevant configuration settings are covered in Global/Tenant Settings, Applications.

IDP Discovery Policy APIs

The following table displays the list of IDP Discovery Policy APIs that you can use to perform CRUD operations on the applicable IDPs.

Operation Description	Operation Type	Endpoint URLs
Create an IDP Discovery Policy.	POST	https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdpDiscoveryPolicies
Get all IDP Discovery Policies.	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdpDiscoveryPolicies
Retrieve an IDP Discovery Policy by specifying the ID.	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdpDiscoveryPolicies/{idpDiscoveryPolicyId}
Update an IDP Discovery Policy by specifying the ID.	PUT	https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdpDiscoveryPolicies/{idpDiscoveryPolicyId}
Update an IDP Discovery Policy Parameter.	PATCH (not yet implemented)	https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdpDiscoveryPolicies/{idpDiscoveryPolicyId}
Delete an IDP Discovery Policy by specifying the ID. The seeded policies that are preloaded into the IDSP cannot be deleted.	DELETE	https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdpDiscoveryPolicies/{idpDiscoveryPolicyId}

Evaluate IDP Discovery Policy API assists users in validating the inputs that they provide when creating or updating the IDP Discovery Policy against the displayed results in the response of the API.	POST	https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdpDiscoveryPolicies/evaluatePolicy

To view the request and response payload for these URLs, refer to IDP Discovery Policy API- Swagger documentation.

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/IdpDiscoveryPolicies`
  should look similar to:
  `https://example.com/default/admin/v1/IdpDiscoveryPolicies`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Examples of POST/PUT and Evaluate IDP Discovery Policy are given here. Click this link to expand the content.

Sample of Creating or Updating IDP Discovery Policy API:

Request: You can add the following IDP-related inputs in the request to create or update an IDP policy.

```
{
    "policyName": "Sample policy",
    "description": "Sample IDP Discovery policy",
    "status": "active",
    "rules": [
        {
            "conditions": {
                "action": {
                    "value": "idpAuthenticate"
                },
                "matchAnyApp": true,
                "app": {
                    "operator": "in",
                    "value": [
                        {
                            "id": "2833d52b-10aa-41a7-bf0d-06139e1b337a"
                        }
                    ]
                },
                "loginName": {
                    "operator": "in",
                    "value": [
                        "*"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
```

```
                        "Corporate channel"
                    ]
                }
            },
            "result": {
                "idps": [
                    {
                        "id": "c3b2f037-4b3b-4b0f-b20c-604605564501"
                    },
                    {
                        "id": "997349a0-9bfc-4024-82d5-e0d92ea3410c"
                    }
                ],
                "msg": "Sample policy",
                "deriveIdpFromLoginName": true
            }
        }
    ]
}
```

Response: The response returns the following details about the IDP policy.

```
{
    "policyId": "b9a4ef8a-0261-41ed-9aab-5d268d8b336b",
    "policyName": "Sample policy",
    "description": "Sample IDP Discovery policy",
    "status": "active",
    "rules": [
        {
            "id": "2358fa35-8f1f-4db1-a90c-1b24e4682130",
            "conditions": {
                "action": {
                    "value": "idpAuthenticate"
                },
                "matchAnyApp": true,
                "ipAddress": {
                    "operator": "in",
                    "value": [
                        "*"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "Corporate channel"
                    ]
                },
                "loginName": {
                    "operator": "in",
                    "value": [
                        "*"
                    ]
                }
            },
```

```
            "result": {
                "idps": [
                    {
                        "name": "Okta",
                        "id": "c3b2f037-4b3b-4b0f-b20c-604605564501"
                    },
                    {
                        "name": "Azure",
                        "id": "997349a0-9bfc-4024-82d5-e0d92ea3410c"
                    }
                ],
                "msg": "Sample policy",
                "deriveIdpFromLoginName": true
            }
        }
    ],
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "d5677e89-cedb-4715-a148-851e23b0b1c9",
        "principalName": "DefaultTenantClient"
    },
    "createdDate": "2023-09-06 05:03 AM UTC"
}
```


Sample of "Evaluate IDP Discovery Policy" API:

Request: You can add the following IDP-related inputs in the request to check the expected result in the response.

```
{
"action": "idpAuthenticate",
"app":["DemoClient"],
"loginName": "cmoran@azure",
"channel":"web"
}
```

Response: The response returns the following details about the IDP policy.

```
[
 {
 "policyName": "Sample idpProtocol policy",
 "msg": "Sample idpProtocol policy",
 "deriveIdpFromLoginName": false,
 "opaIdps": [
 "<idp-guid-given-in-policy-definition>"
 ],
 "effectiveIdps": [
 {
 "id": "<idp-guid-given-in-policy-definition>",
 "name": "<idp-name>",
 "domain": "<idp-domain>",
 "amr": "PWD"
 }
 ]
 }
]
```

Where,

- ***opaIdps***: This field lists the idp's that are mentioned in the Idp Policy definition and that have been returned from the OPA policy evaluation.
- ***effectiveIdps***: This field lists the Idp information for those Idp's that are active and will be used in further authentication flow.

Reference links:

- IDP Discovery Flows
- Creating IDPs

IDP Discovery Flows

Learn about the IDP Discovery Protocol (Before user enters username) and IDP Discovery Authenticate (After user enters username) Flows.

The IDP Discovery policy comprises of these flows, "***Before user enters username***", "***After user enters username***", "***When User is Found***" and "***When User is Not Found***" flows which are explained in the following sections.

This topic includes the following sections:

- IDP Discovery Protocol Flow
  - IDP Discovery Protocol (Before user enters username) Policy Structure
  - Sample of "Before user enters username" Policy
  - User experience based on the "Before user enters username" policy evaluation
- IDP Discovery Authenticate (After user enters username) Flow
  - IDP Discovery Authenticate (After user enters username) Policy Structure
  - Sample of "After user enters username" Policy
  - User experience based on the "After user enters username" policy evaluation
- When User is Found Flow
- When User is Not Found Flow

IDP Discovery Protocol Flow

This flow occurs before the user enters their username in the Signin process and involves the user authentication process using the IDP Protocol flow before the user's actual sign-in.

IDP Discovery Protocol (Before user enters username) Policy Structure

To use this flow, you must create an "Before user enters username" policy which will be evaluated during the Protocol flow. The "Before user enters username" policy should have the following defined in the policy:

- ***action***: specify the value of the "***action***" as "Before user enters username".
- ***matchAnyApp***: this is a boolean property.
  - ***true***: if you set this to "***true***", it matches any application id.
  - ***false***: if you set this to "***false***", specify the application name against which this policy should be evaluated.
- ***channel***: specify the medium through which a user can connect to the Symantec Identity Security Platform. Examples of channel: web and Mobile.
- ***ipAddress***: specify the "ipAddress" to check the connectivity to the IDSP solution from a specific ipAddress or a range of ipAddresses. You can configure the "ipAddress" range in the condition.
- ***result***: Specify the following in the results section:

- ***idps***: specify either a unique idp or a list of idps.
- ***msg***: specify a message about the policy. For example: "This policy is for Before user enters the username" flow.
- ***deriveIdpFromLoginName***: This is a boolean property and is defaulted to ***false***. In case of the "Before user enters username" flow, this property is not considered during evaluation.

Sample "Before user enters username" Policy

The following example provides a sample of an "Before user enters a username" policy, including the relevant rules and results, for your reference.

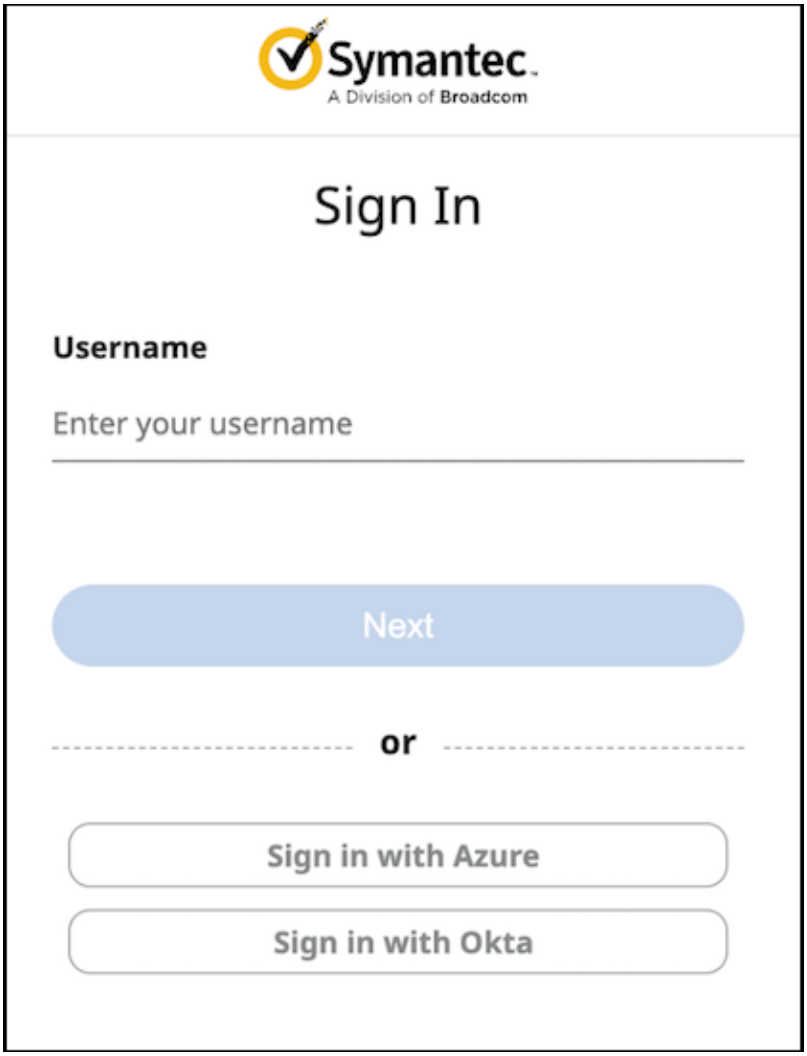
```
{
    "policyName": "Sample policy",
    "description": "Sample IDP Discovery policy",
    "status": "active",
    "rules": [
        {
            "conditions": {
                "action": {
                    "value": "idpProtocol"
                },
                "matchAnyApp": true,
                "app": {
                    "operator": "in",
                    "value": [
                        {
                            "id": "2833d52b-10aa-41a7-bf0d-06139e1b337a"
                        }
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "Corporate channel"
                    ]
                }
            },
            "result": {
                "idps": [
                    {
                        "id": "c3b2f037-4b3b-4b0f-b20c-604605564501"
                    },
                    {
                        "id": "997349a0-9bfc-4024-82d5-e0d92ea3410c"
                    }
                ],
                "msg": "Sample policy",
                "deriveIdpFromLoginName": false
            }
        }
    ]
}
```

User experience based on the policy evaluation

When using this flow, the evaluation of the discovery policy populates either a unique IDP or a list of Identity Provider IDs (GUIDs) based on the policy definition. The GUIDs for these Identity Providers are drawn from those already established within IDSP.

- ***unique IDP*** - When the policy evaluation results in a unique IDP, the user is redirected to the IDP Signin page.
- ***array of IDP's*** - If the policy evaluation results in an array of IDPs, the IDSP Signin page is presented, offering the user the option to either enter their login name or select from the available IDPs, as shown in the following image.

  Figure 30: IDP Discovery Policy-Protocol Flow Image



  - If the user opts to enter their "Username," they must click "***Next***" and should proceed with the standard (AuthN) authentication flow.
  - If the user selects one of the IDPs on the IDSP Signin page, they are redirected to the respective IDP Signin page.

IDP Discovery Authenticate Flow

This flow occurs when the username is entered during the Signin process, and it involves using the IDP Authenticate flow for signing in to an Identity Provider (IDP).

IDP Discovery Authenticate (After user enters username) Policy Structure

To use this flow, you must create an "After user enters username" policy which will be evaluated after a user enters their login name during the Signin flow. The "After user enters username" policy should have the following defined in the policy:

- ***action***: specify the value of the action as "After user enters username".
- ***matchAnyApp***: this is a boolean property.
  - ***true***: if you set this to true, it matches any application id.
  - ***false***: if you set this to false, specify the application name against which this policy should be evaluated.
- ***loginName***: specify the login name which should be validated against the Identity Provider's list to determine the appropriate IDP selection.
- ***channel***: specify the medium through which a user can connect to the IDSP. Examples of channel: web and Mobile.
- ***ipAddress***: specify the "ipAddress" to check the connectivity to the IDSP solution from a specific ipAddress or a range of ipAddresses. You can configure the "ipAddress" range in the condition.
- ***result***: Specify the following in the results section:
  - ***idps***: specify either a unique idp or a list of idps.
  - ***msg***: specify a message about the policy. For example: "This policy is for the "Before user enters username" flow.
  - ***deriveIdpFromLoginName***: In the results section of the discovery policy, the "deriveIdpFromLoginName" property specifies if an IDP is derived from the login name that is entered by a user. This is a boolean property and is defaulted to ***false***.

    NOTE
    When this property is set to ***true***, it takes precedence over the ***idps*** mentioned in the ***result*** section.
    - when "***deriveIdpFromLoginName***" is set to ***false***, the user is presented with a list of IDPs from where the user can select, along with an option to continue with the standard "AuthN" flow.
    - when "***deriveIdpFromLoginName***" is set to ***true***, the discovery policy performs the process of extracting the domain from the user name, locating the Identity Provider (IDP) associated with that domain, and finally, redirecting to the chosen IDP.

Sample "After user enters username" Policy

The following example provides a sample of an "After user enters username" policy, including the relevant rules and results, for your reference.

```
{
    "policyName": "Sample policy",
    "description": "Sample IDP Discovery policy",
    "status": "active",
    "rules": [
        {
            "conditions": {
                "action": {
                    "value": "idpAuthenticate"
                },
                "matchAnyApp": true,
                "app": {
                    "operator": "in",
                    "value": [
                        {
                            "id": "2833d52b-10aa-41a7-bf0d-06139e1b337a"
                        }
                    ]
                },
                "loginName": {
```

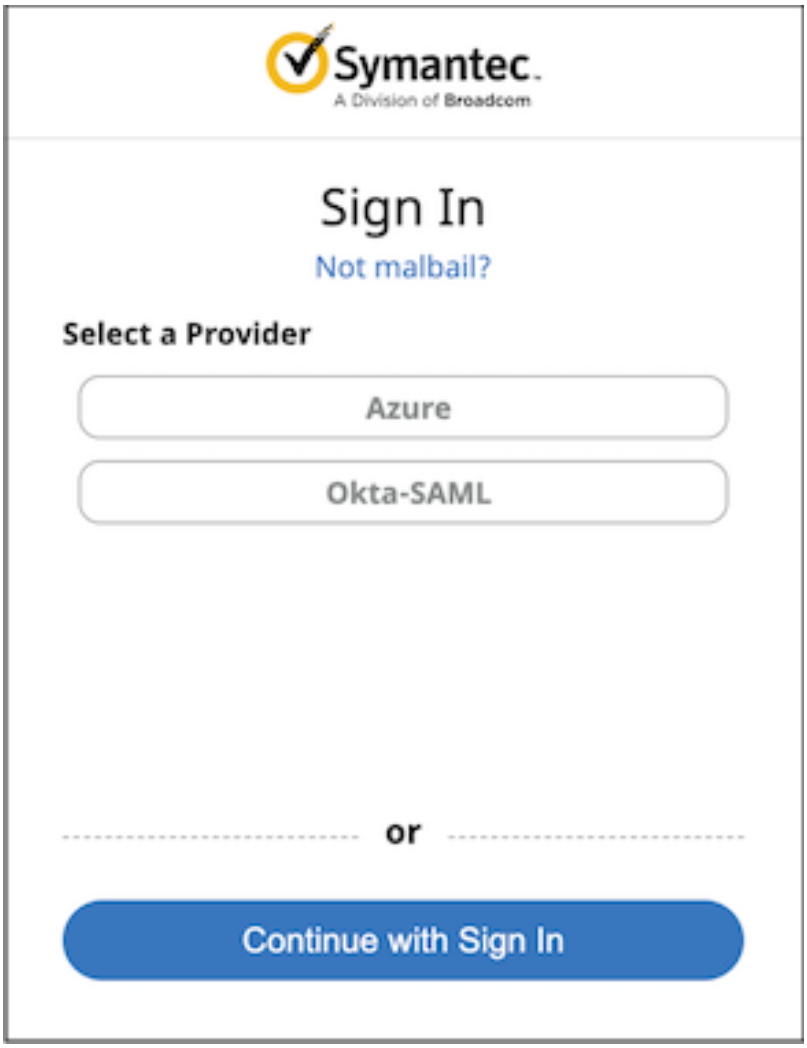
```
                    "operator": "in",
                    "value": [
                        "*"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "Corporate channel"
                    ]
                }
            },
            "result": {
                "idps": [
                    {
                        "id": "c3b2f037-4b3b-4b0f-b20c-604605564501"
                    },
                    {
                        "id": "997349a0-9bfc-4024-82d5-e0d92ea3410c"
                    }
                ],
                "msg": "Sample policy",
                "deriveIdpFromLoginName": true
            }
        }
    ]
}
```

User experience based on the policy evaluation

When using the "***After user enters username***" flow, the evaluation of the idp discovery policy will be based on the configurations of the ***deriveIdpFromLoginName*** property. If ***deriveIdpFromLoginName*** is set to ***true***, it takes precedence over the ***idps*** specified in the ***"result"*** section of the policy.

- If the ***deriveIdpFromLoginName*** property is set to "***true":*** the following scenarios can occur:
  - Regardless of whether ***idps*** are explicitly mentioned in the ***"result"*** section or not, the "***After user enters username***" policy will consistently consider the ***"loginName"*** input to search for associated user information and evaluate the policy accordingly.
    - ***Scenario 1***: If a user includes the domain name in their login name, the policy searches for that domain within the established IDPs in IDSP. If an idp is found matching to that specific domain, the Idp is validated to be present in the list given in Policy (if no list was provided in the Policy, the Idp will be used to redirect), and if the Idp is present in that list, then the user is directed to the corresponding IDP Signin page. However, if the domain is not found within the established idps, the policy searches for the user in the configured IDStore. If the user is found there, the standard "AuthN" policy evaluation continues, otherwise, an error message stating, ***User Not Found*** is displayed to the user.
    - ***Scenario 2***: If a user enters a username without a domain, the user must click **"Next"** and proceed with the standard "AuthN" flow.
- If ***deriveIdpFromLoginName*** property is set to "***false"***, the user is shown the option to choose between the list of available IDPs or continue with the standard "AuthN" flow as shown in the following image.

Figure 31: IDP Discovery Policy- Authenticate Flow Image



When User is Found

In this flow, the authentication controller decides if an auto redirect to IDP should take place after the user is disambiguated successfully. This is based on the following conditions defined in the IDSP AuthN (authentication) policy:

- App
- IP Address
- Channel
- User Context (in the Context condition)

If any one or all the conditions in the AuthN policy are matched, the user is redirected to the IDP.

When the User is Not Found

In this flow, the authentication controller decides if an auto redirect to IDP should take place after the user was not disambiguated. This is based on the following conditions defined in the IDSP AuthN (authentication) policy:

- App
- IP Address
- Channel

If any one or all the conditions in the AuthN policy are matched, the user is redirected to the IDP.

To know more about AuthN Policy conditions, see here.

Setting Priority in IDP Policy and Rule

This topic explains how setting priority for Identity Discovery policies and rules ensures only prioritized ones influence the final result (Next Action).

This topic includes the following sections:

- Overview of Policy and Rule Priority
- How Priorities Work in Policy Evaluation
- Matching Logic: Priority vs. Non-Priority
- Handling Conflicts: Allow vs. Deny
- Understanding Priority with Decimals
- Policy API payload with Priority
- Common Use Cases

Overview of Policy and Rule Priority

Priority is a numerical value assigned to a policy to control its evaluation order. Lower numbers represent **higher priority** (for example, `1` is higher than `100`). Rule Priority is a numerical value assigned to individual rules within a policy. Like policies, lower numbers mean higher importance or precedence.

Priority can be set at both the policy and rule levels, ranging from 1 (highest) to 1,000,000 (lowest). Decimals can be used for finer control. Multiple policies and rules can share the same priority. If two or more matched rules have the same priority at both the policy and rule levels, the final result is a combination of all the matched rules.

How Priorities Work in Policy Evaluation

- During policy evaluation, all policies regardless of whether they have priorities or not are considered. However, if there are matching policies that have priorities defined, only those policies with defined priorities and matched policies are used to determine the final result. In other words, when policies with priority and non-priority matches, only the policies with priority influences the final result.
- During policy evaluation, all policies are evaluated, but if no policies with priority match the defined conditions, then the matched policies without priority are used to determine the final result. In other words, if no policies or rules with priorities are applicable, the final outcome includes all obligations from the matched policies and rules without priorities.
- A policy with its priority set to zero (0) is effectively deactivated and is not considered when determining the final result.

Refer to the next section to understand the same with examples.

Matching Logic: Priority vs. Non-Priority

When both prioritized and non-prioritized policies and rules are present, the evaluation logic first considers those with defined priorities. If no matching conditions are found among them, it then evaluates the policies and rules without priorities.

Refer to the following example for better understanding.

Policy Details	Policy Condition Match	Result
Policy 1 with "policyPriority": 1 and "rulePriority": 1	No policy conditions match	No IDP
Policy 2 with "policyPriority": none and "rulePriority": 1	No policy conditions match	No IDP
Policy 3 with "policyPriority": none and "rulePriority": none	Yes, policy conditions match	IDPs from **Policy 3** will be sent as the result.

Policy Evaluation Outcome:

- Although Policies 1 and 2 include priority values, neither of them matches the evaluation conditions. Policy 3, which has no priorities defined, is the only one with a matching condition.
- As a result, Policy 3 takes precedence and its IDPs is applied in the final result.
- When none of the policies with priorities match, the policy evaluation uses the matching policies without priorities to determine the final result. In this case, `IDPs from` **`Policy 3`** are selected because it is the only result from a policy that successfully matched the conditions.

Handling Conflicts: Allow vs. Deny

Priority plays a key role in determining how policies and rules are evaluated, especially when they have different results (Result 1 with IDPs vs Result 2 with IDPs). The following examples illustrate different scenarios based on priority levels defined at both the policy and rule levels.

> **NOTE**
> The IDP Policy API defines policy and rule priorities using the attributes **"policyPriority"** and **"rulePriority,"** respectively.

Use Case 1: Same Priority Level, Different Results
When multiple policies and rules share the same priority level but have different "Result", the result is collated to determine the final result.

Policy 1 with "policyPriority": 1	"rulePriority": 1	Result: Result 1 with IDPs
Policy 2 with "policyPriority": 1	"rulePriority": 1	Result: Result 2 with IDPs
Rule Summary - Since both policies have the same priority, **Policy 1 and Policy 2 results are collated** and is applied in the final result.		

Use Case 2: Different Priority Levels, Different Results
When policies and rules have **different priority levels**, the policy with the **higher priority (lower number)** takes precedence, regardless of its result.

Policy 1 with "policyPriority": 1	"rulePriority": 2	Result: Result 1 with IDPs
Policy 2 with "policyPriority": 2	"rulePriority": 1	Result: Result 2 with IDPs
Rule Summary - Since Policy 1 has the highest priority (1 is higher than 2), **Policy 1 (Result 1 with IDPs)** takes precedence and is applied in the final result.		

Use Case 3: Same Policy Priority, Different Rule Priorities
When policies have the **same priority level**, but their rules have **different priorities**, the rule with the highest priority (lowest number) takes precedence.

Policy 1 with "policyPriority": 1	"rulePriority": 2	Result: Result 1 with IDPs
Policy 2 with "policyPriority": 1	"rulePriority": 1	Result: Result 2 with IDPs

Rule Summary - Since Policy 2 has a rule with a higher priority (1 vs. 2), **Policy 2 (Result 2 with IDPs)** takes precedence and is applied in the final result.

Understanding Priority with Decimals

The following example shows how priority with decimals work in Symantec Identity Security Platform.

Consider the following priority values as an example: `1,1.1,1.23,1.4,2,3,4,4.1,4.5`

The policy evaluation results in the following order:

- 1 has the highest priority, followed by 1.1 then 1.23, and so on.
- 1.1 takes precedence over 1.23
- 1.4 has higher priority than 4
- 4 is prioritized over 4.5

This approach ensures more granular control within the same shared priority range, ensuring policies and rules are executed in the intended sequence

IDP Policy API payload with "policyPriority" and "rulePriority"

> **NOTE**
> The IDP Policy API defines policy and rule priorities using the attributes **"policyPriority"** and **"rulePriority,"** respectively.

Example of an IDP Policy API payload with "policyPriority" and "rulePriority" priority

```
{
    "policyName": "Sample policy",
    "description": "Sample IDP Discovery policy",
    "status": "active",
    "policyPriority": "1",
    "rules": [
        {
            "conditions": {
                "action": {
                    "value": "idpAuthenticate"
                },
                "matchAnyApp": true,
                "app": {
                    "operator": "in",
                    "value": [
                        {
                            "id": "2833d52b-10aa-41a7-bf0d-06139e1b337a"
                        }
                    ]
                },
                "loginName": {
                    "operator": "in",
                    "value": [
                        "*"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
```

```
                    "Corporate channel"
                ]
            }
        },
        "result": {
            "idps": [
                {
                    "id": "c3b2f037-4b3b-4b0f-b20c-60460556XXXX"
                },
                {
                    "id": "997349a0-9bfc-4024-82d5-e0d92ea3ZZZZ"
                }
            ],
            "msg": "Sample policy",
            "deriveIdpFromLoginName": true,
            "rulePriority": 2
        }
    }
]
}
```

Common Use Cases - IDP Policy with Priority Based Policy and Rule Selection

NOTE
The IDP Policy API defines policy and rule priorities using the attributes **"policyPriority"** and **"rulePriority,"** respectively.

IDP Policy Use Cases			
Sl.No	**No of Policies and their Priorities**	**Matched Rules with Priorities**	**Result**
1	Policy 1 (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 1 Rule 2 with "rulePriority": 2	IDP of **Rule: 1** will be sent as the result.
2	Policy 1 (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 2 Rule 2 with "rulePriority": 1	IDP of **Rule: 2** will be sent as the result.
3	Policy 1 (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 1 Rule 2 with "rulePriority": 1	IDPs of both the rules **(Rule:1 and Rule: 2)** will be sent as the result.
4	Policy 1 with "policyPriority": 1 (has two rules)	Rule 1 with "rulePriority": 1 Rule 2 with "rulePriority": 2	IDP of **Policy 1, Rule: 1** will be sent as the result.
	Policy 2 with "policyPriority":2 (has two rules)	Rule 1 with "rulePriority": 1 Rule 2 with "rulePriority": 2	
5	Policy 1 with "policyPriority":1 (has two rules)	Rule 1 with "rulePriority": 2 Rule 2 with "rulePriority": 1	IDP of **Policy 1, Rule: 2** will be sent as the result.
	Policy 2 with "policyPriority": 2 (has two rules)	Rule 1 with "rulePriority": 2 Rule 2 with "rulePriority": 3	
6	Policy 1 with "policyPriority": 1 (has two rule)	Rule 1 with "rulePriority": 2 Rule 2 with "rulePriority": 1	IDP of **Policy 1, Rule: 2** will be sent as the result.
	Policy 2 with "policyPriority": 2 (has two rules)	Rule 1 with "rulePriority": 2 Rule 2 with "rulePriority": 3	

	Policy 3 with "policyPriority": 3 (has one rule)	Rule 1 with "rulePriority": 1	
7	Policy 1, (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 1 Rule 2 with "rulePriority": 1.1	IDP of **Rule: 1** will be sent as the result.
8	Policy 1, (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 2 Rule 2 with "rulePriority": 3	IDP of **Policy 2, Rule: 1** will be sent as the result.
	Policy 2, (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 1.1 Rule 2 with "rulePriority": 2	
9	Policy 1, (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 2 Rule 2 with "rulePriority": 3	IDP of **Policy 2, Rule: 1** will be sent as the result.
	Policy 2, (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 1.4 Rule 2 with "rulePriority": 2	
10	Policy 1 with "policyPriority": 1 (has no rule priority)	Rule 1 with no "rulePriority"	IDP of **Policy 1** will be sent as the result.
	Policy 2 with "policyPriority": 2 (has no rule priority)	Rule 2 with no "rulePriority"	
11	Policy 1 with "policyPriority": 2 (has one rule)	Rule 1 with "rulePriority": 3	IDP of **Policy 2, Rule: 1** will be sent as the result.
	Policy 2 with "policyPriority": 1 (has two rules)	Rule 1 with "rulePriority": 1.1 Rule 2 with "rulePriority": 2	
12	Policy 1 with "policyPriority": 1 (has a single rule)	Rule 1 with "rulePriority": 2	IDP of **Policy 1, Rule: 1** will be sent as the result.
	Policy 2, (has no explicitly defined policy priority but contains two rules)	Rule 1 with "rulePriority": 1.1 Rule 2 with "rulePriority": 2	

Reference Links:

- Priority configuration in AuthN Policy and Rule

Risk Policy

This topic describes a "Risk Policy" as a set of rules and conditions used to evaluate a user's risk profile at an application level. It outlines key policy attributes like policyName, description, status, and rules.

Risk Policy Overview

A Risk Policy is a set of rules and conditions that determine how to evaluate a user’s risk profile at an application-level.

The Risk Policy typically includes the following elements:

- **Rules**: These rules define conditions that must be met to determine which risk rules should be used for evaluation of a user’s risk profile. You can define multiple rules and map action-specific conditions to each risk rule.
- **Policy Evaluation**: Active policies are evaluated against given context resulting in a set of unique and active applicable risk rules.
- **User Experience**: There is no impact to the user experience with the use of a risk policy as the risk evaluation is a background process.

Overall, a Risk Policy assesses user risk in an application based on rules and contextual evaluation, without affecting user experience.

Application Metadata Configuration

To enable stricter risk evaluation for authentication flows within IDSP, configure the following application metadata:

- "urn:iam:app:requireRiskControl"="true"

This metadata indicates that a risk policy must be matched exactly. If set to "true", and no matching risk policies or rules are found for evaluation, an exception is thrown, and an error is displayed. In this scenario, the risk evaluation does not fall back to the Global or Tenant settings. See Application Metadata.

Risk Policy Attributes

The details of the Risk policy attributes are described here:

- **policyName**: Displays the title of the policy.
- **description**: Details a brief about the policy.
- **priority**- displays the priority of the policy, if configured. The concept of priority in risk policy and rule is same as the priority explained in the Authentication Policy. To know more about policy priority, see Setting Priority in Authentication Policy and Rule.
- **status**: Specifies whether a policy is active or inactive.
- **rules**: Includes conditions and result. You can specify a single condition or multiple conditions in a rule. If the specified conditions in the rule are met, then the corresponding result is applied. The supported condition categories are app and action.
  - **conditions**: Allows you to specify various conditions that must be evaluated for an authentication request. Conditions express the relationship of an attribute to a value. For example, if you specify "action" as one of the attributes, then the values could be "login". The "NOT" operator is supported in the rule conditions such as app and action.
  - A risk policy supports the following **rule conditions**:
    - **action**: Specifies the action of the context in which the risk is evaluated. It supports a string value which is case sensitive. So it is crucial that the "action" value defined in your risk policy precisely matches the one used in your risk API calls (for example, "action":"authenticate").
    - **matchAnyApp**: Specifies whether the policy is applicable to all the applications or to a specific set of applications that are defined in the apps section in the policy. It supports true/false values.
      - If set to true: The policy is applied to all the applications.
      - If set to false (default): The policy is applied to the applications that are defined in the app. It supports an array of application IDs to which a policy is applied.
    - **app**: Specifies the applications to which the policy is applied. This value supports an array of application IDs to which a policy is applied. This value is case sensitive, so it is crucial that the "app" value defined in your risk policy precisely matches the one used in your risk API calls (for example, "app":"DemoClientApp"). The operator supported is - “in”.
  - **result**: Indicates the risk rules that the risk engine evaluates as part of the riskEvaluation operation. The result has msg (a brief description of a rule) and riskRules (risk rule name and risk score). Note that the rules that are added as part of the risk policy must be enabled in the tenant.

Sample Risk Policy Payload

Request Payload

```
{
"policyName": "Sample risk policy ",
"description": "Sample policy for risk rules to be used for a given app, action",
"status": "active",
"rules": [
{
"conditions": {
```

```
"action": {
"operator": "in""value": [
"authenticate",
]
},
"matchAnyApp": false,
"app": {
"operator": "in",
"value": [
{
"name": "DemoClientApp",
"id": "{{appId}}"
}
]
}
},
"result": {
"riskRules": [
{
"riskRule": "behavior anomaly",
"riskScore": 100
},
{
"riskRule": "device recognition""riskScore": 75,
},
{
"riskRule": "inactive user""riskScore": 100,
},
{
"riskRule": "previous event is confirmed risk""riskScore": 75,
},
{
"riskRule": "user velocity""riskScore": 100,
},
{
"riskRule": "work day time""riskScore": 75,
},
{
"riskRule": "enhanced difficult travel""riskScore": 100,
},
{
"riskRule": "risky country",
"riskScore": 75
},
{
"riskRule": "fraud ip""riskScore": 75,
},
{
"riskRule": "blocked ip""riskScore": 100,
},
{
"riskRule": "allowed ip""riskScore": 0,
},
```

```
{
"riskRule": "rc result""riskScore": 75,
},
{
"riskRule": "country change",
"riskScore": 75
},
{
"riskRule": "ip change""riskScore": 75,
}
],
"msg": "sample policy with all risk rules for demo purpose"
}
}
]
}
```


Response Payload


```
{
"policyId": "6d049d5b-9235-4fb9-a933-6f8533cdf744",
"policyName": "Sample risk policy",
"description": "Sample policy for risk rules to be used for a given
app, action",
"status": "active",
"rules": [
{
"id": "4093aeca-2c8b-4abe-ac2f-3631b9d1d420",
"conditions": {
"action": {
"operator": "in",
"value": [
"authenticate"
]
},
"matchAnyApp": false,
"app": {
"operator": "in",
"value": [
{
"name": "DemoClientApp",
"id": "cfe1cacc-e38f-442a-8417-2911cee45dac"
}
]
}
},
"result": {
"riskRules": [
{
"riskRule": "behavior anomaly",
"riskScore": 100
},
{
"riskRule": "device recognition",
```

```
},
{},
{},
{},
{},
{},
{},
{},
{},
{
"riskScore": 75
"riskRule": "inactive user",
"riskScore": 100
"riskRule": "previous event is confirmed risk""riskScore": 75
"riskRule": "user velocity""riskScore": 100,
"riskRule": "work day time""riskScore": 75,
"riskRule": "enhanced difficult travel",
"riskScore": 100
"riskRule": "risky country",
"riskScore": 75
"riskRule": "fraud ip""riskScore": 75,
"riskRule": "blocked ip""riskScore": 100,
,
"riskRule": "allowed ip",
"riskScore": 0
},
{
"riskRule": "rc result""riskScore": 75,
},
{
"riskRule": "country change""riskScore": 75,
},
{
"riskRule": "ip change""riskScore": 75,
}
],
"msg": "sample policy with all risk rules for demo purpose"
}
}
],
"adminscope.criteria": [],
"adminscope.managedby": [],
"createdBy": {
"principalType": "CLIENT",
"principalId": "cfe1cacc-e38f-442a-8417-2911cee45dac""principalName": "DemoClientApp",
},
"createdDate": "2025-07-16 08:44 AM UTC"
}
```


Risk Policy API Scope

The scope of this API is "**urn:iam:t.configs**".

Risk Policy APIs

Operation Description	Operation Type	Endpoint URLs
This API creates a risk policy	POST	https://${SSP_FQDN}/{{TenantName}}/admin/v1/RiskPo
This API retrieves all the risk policies.	GET	https://${SSP_FQDN}/{{TenantName}}/admin/v1/RiskPo
This API retrieves a risk policy that is specified by the policy Id	GET	https://${SSP_FQDN}/{{TenantName}}/admin/v1/RiskPo {{policyId}}
This API updates a risk policy by specifying the policy Id	PUT	https://${SSP_FQDN}/{{TenantName}}/admin/v1/RiskPo {{policyId}}
This API deletes a risk policy by specifying the policy Id	DELETE	https://${SSP_FQDN}/{{TenantName}}/admin/v1/RiskPo {{policyId}}

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/RiskPolicies`
  should look similar to:
  `https://example.com/default/admin/v1/RiskPolicies`
-

GET All ".../admin/v1/RiskPolicies" sample

Perform a GET request using the following admin API:

API Request

```
GET https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/RiskPolicies
```

API Response

The following response retrieves all the risk policies that are created under Symantec Identity Security Platform.


```
[
{
"policyId": "1c8847db-4fdb-4414-8bbc-a90e540b1726",
"policyName": "risk policy 123",
"description": " ",
"status": "inactive",
"rules": [
{
"id": "6b9a55d7-26d1-417d-a299-b978eebd4d74",
"conditions": {
"action": {
"operator": "in",
"value": [
"authenticate"
]
},
"matchAnyApp": false,
"app": {
```

```
"operator": "in",
"value": [
{
 "name": "App2",
 "id": "78a9853b-4668-47f0-bd1e-5589a76c4fb4",
 "type": "app"
}
]
},
"channel": {
"operator": "in",
"value": [
"web"
]
}
},
"result": {
"riskRules": [
{
"riskRule": "behavior anomaly",
"riskScore": 50
}
],
"msg": "fgdg"
}
}
],
"adminscope.criteria": [],
"adminscope.managedby": [],
"createdBy": {
"principalType": "USER",
"principalId": "7d425a1c-e823-482d-9208-889f7dd85b4d",
"principalName": "malbail"
},
"createdDate": "2025-09-23 10:26 AM UTC",
"updatedBy": {
"principalType": "USER",
"principalId": "7d425a1c-e823-482d-9208-889f7dd85b4d",
"principalName": "malbail"
},
"updatedDate": "2025-09-23 10:40 AM UTC"
},
{
"policyId": "ed7c948e-342c-4ed6-a834-a6bf925ff491",
"policyName": "risk policy",
"description": " ",
"status": "active",
"rules": [
{
"id": "0febca6d-df70-4cc1-a482-84c128978d9e",
"conditions": {
"action": {
"operator": "in",
```

```
"value": [
"authenticate"
]
},
"matchAnyApp": false,
"app": {
"operator": "in",
"value": [
{
 "name": "App1",
 "id": "ff193aaf-c757-4d2d-b008-02150425d163",
 "type": "app"
}
]
},
"channel": {
"operator": "in",
"value": [
"web"
]
}
},
"result": {
"riskRules": [
{
"riskRule": "behavior anomaly",
"riskScore": 50
},
{
"riskRule": "device recognition",
"riskScore": 50
}
],
"msg": "testing"
}
}
],
"adminscope.criteria": [],
"adminscope.managedby": [],
"createdBy": {
"principalType": "USER",
"principalId": "940b4226-e553-48c9-804f-dc7fc589c99e",
"principalName": "malbail"
},
"createdDate": "2025-09-19 10:53 AM UTC",
"updatedBy": {
"principalType": "USER",
"principalId": "940b4226-e553-48c9-804f-dc7fc589c99e",
"principalName": "malbail"
},
"updatedDate": "2025-09-19 10:56 AM UTC"
}
```

POST ".../admin/v1/RiskPolicies" sample

Perform a POST request using the following admin API:

API Endpoint

```
POST https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/RiskPolicies
```

API Request

This request shows how a risk policy can be created.

```
{
"policyName": "Sample risk policy for web channel",
"description": "Sample policy for risk rules to be used for a given app, action and channel",
"status": "active",
"rules": [
  {
      "conditions": {
          "action": {
              "operator": "in",
              "value": [
                  "authenticate"
              ]
          },
          "channel": {
              "operator": "in",
              "value": [
                  "web"
              ]
          },
          "matchAnyApp": false,
          "app": {
              "operator": "in",
              "value": [
                  {
                      "id": "78a9853b-4668-47f0-bd1e-5589a76c4fb4"
                  }
              ]
          }
      },
      "result": {
          "riskRules": [
              {
                  "riskRule": "country change",
                  "riskScore": 100
              },
              {
                  "riskRule": "behavior anomaly",
                  "riskScore": 75
              }
          ],
          "msg": "Country change and BE Risk Rule"
      }
  }
]
```

```
}
```

API Response

The following response retrieves all the risk policies that are created under IDSP.


```
{
"policyId": "50c2ed70-8508-4cb5-8500-0c3ba407065b",
"policyName": "Sample risk policy for web channel",
"description": "Sample policy for risk rules to be used for a given app, action and channel",
"status": "active",
"rules": [
{
"id": "a9cb26b0-9ff4-4be0-a359-aa65214719da",
"conditions": {
"action": {
"operator": "in",
"value": [
 "authenticate"
]
},
"matchAnyApp": false,
"app": {
"operator": "in",
"value": [
 {
   "name": "App2",
   "id": "78a9853b-4668-47f0-bd1e-5589a76c4fb4"
 }
]
},
"channel": {
"operator": "in",
"value": [
 "web"
]
}
},
"result": {
"riskRules": [
{
 "riskRule": "country change",
 "riskScore": 100
},
{
 "riskRule": "behavior anomaly",
 "riskScore": 75
}
],
"msg": "Country change and BE Risk Rule"
}
}
],
"adminscope.criteria": [],
```

```
"adminscope.managedby": [],
"createdBy": {
"principalType": "CLIENT",
"principalId": "3f3b8c8d-c9dc-4684-9228-865beeb81674",
"principalName": "DefaultTenantClient"
},
"createdDate": "2025-09-24 09:16 AM UTC"
}
```


GET ".../admin/v1/RiskPolicies/specify ID" sample

Perform a GET request using the following admin API:

API Endpoint

```
GET https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/RiskPolicies/1c8847db-4fdb-4414-8bbc-a90e540b1726
```

API Response

The following response shows the retrieved risk policy based on the specified risk policy Id


```
{
"policyId": "1c8847db-4fdb-4414-8bbc-a90e540b1726",
"policyName": "risk policy 123",
"description": " ",
"status": "inactive",
"rules": [
{
"id": "6b9a55d7-26d1-417d-a299-b978eebd4d74",
"conditions": {
"action": {
"operator": "in",
"value": [
 "authenticate"
]
},
"matchAnyApp": false,
"app": {
"operator": "in",
"value": [
 {
   "name": "App2",
   "id": "78a9853b-4668-47f0-bd1e-5589a76c4fb4",
   "type": "app"
 }
]
},
"channel": {
"operator": "in",
"value": [
 "web"
]
}
},
"result": {
"riskRules": [
{
```

```
 "riskRule": "behavior anomaly",
 "riskScore": 50
}
],
"msg": "fgdg"
}
}
],
"adminscope.criteria": [],
"adminscope.managedby": [],
"createdBy": {
"principalType": "USER",
"principalId": "7d425a1c-e823-482d-9208-889f7dd85b4d",
"principalName": "malbail"
},
"createdDate": "2025-09-23 10:26 AM UTC",
"updatedBy": {
"principalType": "USER",
"principalId": "7d425a1c-e823-482d-9208-889f7dd85b4d",
"principalName": "malbail"
},
"updatedDate": "2025-09-23 10:40 AM UTC"
}
```


PUT ".../admin/v1/RiskPolicies" sample

Perform a PUT request using the following admin API:

API Endpoint

```
PUT https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/RiskPolicies/50c2ed70-8508-4cb5-8500-0c3ba407065b
```

API Request

This request shows how a risk policy can be created.

```
{
"policyName": "Sample risk policy for web channel",
"description": "Sample policy for risk rules to be used for a given app, action and channel",
"status": "active",
"rules": [
  {
      "conditions": {
          "action": {
              "operator": "in",
              "value": [
                  "authenticate"
              ]
          },
          "channel": {
              "operator": "in",
              "value": [
                  "web"
              ]
          },
          "matchAnyApp": false,
          "app": {
```

```
                "operator": "in",
                "value": [
                    {
                        "id": "78a9853b-4668-47f0-bd1e-5589a76c4fb4"
                    }
                ]
            }
        },
        "result": {
            "riskRules": [
                {
                    "riskRule": "country change",
                    "riskScore": 100
                },
                {
                    "riskRule": "behavior anomaly",
                    "riskScore": 75
                }
            ],
            "msg": "Country change and BE Risk Rule"
        }
    }
]
}
```

API Response

The following response retrieves all the risk policies that are created under IDSP.

```
{
"policyId": "50c2ed70-8508-4cb5-8500-0c3ba407065b",
"policyName": "Sample risk policy for web and mobile channel",
"description": "Sample policy for risk rules to be used for a given app, action and channel",
"status": "active",
"rules": [
{
"id": "dd252e18-8c44-4ff4-8c21-d091bc7c65d1",
"conditions": {
"action": {
"operator": "in",
"value": [
 "authenticate"
]
},
"matchAnyApp": false,
"app": {
"operator": "in",
"value": [
 {
   "name": "App2",
   "id": "78a9853b-4668-47f0-bd1e-5589a76c4fb4"
 }
]
},
```

```
"channel": {
"operator": "in",
"value": [
 "mobile"
]
}
},
"result": {
"riskRules": [
{
 "riskRule": "country change",
 "riskScore": 100
},
{
 "riskRule": "behavior anomaly",
 "riskScore": 75
}
],
"msg": "Country change and BE Risk Rule"
}
}
],
"adminscope.criteria": [],
"adminscope.managedby": [],
"createdBy": {
"principalType": "CLIENT",
"principalId": "3f3b8c8d-c9dc-4684-9228-865beeb81674",
"principalName": "DefaultTenantClient"
},
"createdDate": "2025-09-24 09:16 AM UTC",
"updatedBy": {
"principalType": "CLIENT",
"principalId": "3f3b8c8d-c9dc-4684-9228-865beeb81674",
"principalName": "DefaultTenantClient"
},
"updatedDate": "2025-09-24 09:23 AM UTC"
}
```


DELETE ".../admin/v1/RiskPolicies" sample

Perform a DELETE request using the following admin API:

API Request

```
DELETE https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/RiskPolicies/1c8847db-4fdb-4414-8bbc-a90e540b1726
```

API Response

The following response retrieves all the risk policies that are created under IDSP.

```
204 No Content
```

Admin Login Policy

Learn about OOTB AdminLoginPolicy

The "Admin Login Policy" is a pre-configured, out-of-the-box solution designed to govern how tenant administrators access the Admin Console. It primarily targets internal or designated administrator users (such as internal, defaulttenantadminuser), enabling their authentication with a password. Applicable across both web and mobile access channels, this policy supports applications like "DefaultTenantClient" and "AdminConsole." Furthermore, its protected status ensures it cannot be deleted.

This policy is primarily meant for tenant administrators who need to securely access the Admin Console and related default client applications within the platform.


```
{
"policyId": "c6df7028-3edb-4cec-a61f-9bdec8c83470",
"policyName": "AdminLoginPolicy",
"description": "Default policy for Tenant Admin Console",
"status": "active",
"matchAnyApp": false,
"apps": [
{
  "name": "DefaultTenantClient",
  "id": "3f3b8c8d-c9dc-4684-9228-865beeb81674"
},
{
  "name": "AdminConsole",
  "id": "$TENANT_ADMINCONSOLE"
}
],
"rules": [
{
  "id": "1e72aba9-79cb-4057-9387-9d722a06975a",
  "conditions": {
    "principal": {
      "user": {
        "operator": "in",
        "value": [
          "internal/*",
          "be4001e4-7324-4fb9-a569-7b3a4728bcdd/*",
          "internal/defaulttenantadminuser"
        ]
      }
    },
    "action": {
      "operator": "in",
      "value": [
        "authenticate"
      ]
    },
    "channel": {
      "operator": "in",
      "value": [
        "web",
        "mobile"
      ]
    }
  },
  "result": {
```

```
    "effect": "allow",
    "reAuthenticate": "false",
    "obligations": [
      "PASSWORD:1"
    ],
    "msg": "Auto provisioned login rule for DefaultTenantAdminUser",
    "mfaFrequency": "EveryTime"
  }
}
],
"adminscope.criteria": [],
"adminscope.managedby": [],
"createdDate": "2025-07-23 08:07 AM UTC",
"updatedBy": {
"principalType": "USER",
"principalId": "940b4226-e553-48c9-804f-dc7fc589c99e",
"principalName": "malbail"
},
"updatedDate": "2025-08-01 09:45 AM UTC",
"preventedOperations": "Delete"
}
```


Self-Service Login Policy

Learn about OOTB SelfServiceLoginPolicy

This is a pre-configured, out-of-the-box policy provided by Symantec Identity Security Platform for logging into the Self-Service Console. This policy has two pre-configured rules which are explained here:

- **Auto provisioned login rule for all users** - This rule manages the user login process for the console. By default, the policy uses PASSWORD as the only authentication factor.
- **Auto provisioned registration rule for all users** - This rule by default enables re-authentication when a user attempts to add a new authentication method.


```
{
  "policyName": "SelfServiceLoginPolicy",
  "description": "Default policy for Self Service Console",
  "status": "active",
  "matchAnyApp": false,
  "apps": [
      {
          "name": "SelfServiceConsole",
          "id": "$SELFSERVICE_CONSOLE"
      }
  ],
  "rules": [
      {
          "conditions": {
              "principal": {
                  "user": {
                      "operator": "in",
                      "value": [
                          "*"
```

```
                ]
            }
        },
        "channel": {
            "operator": "in",
            "value": [
                "web",
                "mobile"
            ]
        },
        "acr": {
            "operator": "in",
            "value": [
                "urn:iam:acr:aal:selfservicereauth"
            ]
        }
    },
    "result": {
        "effect": "allow",
        "reAuthenticate": "true",
        "obligations": [
            "PASSWORD:1"
        ],
        "msg": "Auto provisioned registration rule for all users"
    }
},
{
    "conditions": {
        "principal": {
            "user": {
                "operator": "in",
                "value": [

                    "*"
                ]
            }
        },
        "action": {
            "operator": "in",
            "value": [
                "authenticate"
            ]
        },
        "channel": {
            "operator": "in",
            "value": [
                "web",
                "mobile"
            ]
        }
    },
    "result": {
        "effect": "allow",
```

```
                "reAuthenticate": "false",
                "obligations": [
                    "PASSWORD:1"
                ],
                "msg": "Auto provisioned login rule for all users",
                "mfaFrequency": "EveryTime"
            }
        }
    ],
    "adminscope.criteria": [],
    "adminscope.managedby": []
}
```

Activate Account Policy

Learn about the out-of-the-box ActivateAccountPolicy.

The "Activate Account Policy" is a pre-configured, out-of-the-box solution that is designed to govern how users authenticate themselves when activating an account. This policy cannot be deleted.

This policy allows you to set which credential types users can use to authenticate when activating an account. By default, this policy is pre-configured to use SMS and email as the security code delivery method. This policy is also assigned to all users. However, you can modify these settings as required.

> **NOTE**
> The account activation flow does not allow inline credential registration. Make sure to set credential types that your users already have registered for their use.

This policy applies only to your sign-in client app. You cannot change the application that is assigned to this policy. Also, you cannot add any custom actions to this policy.

Set the **Account Activation Flow Type** option to **Policy** in the internal Identity Store to set this policy as the default. See Internal Identity Store Configuration.

```
{
    "policyId": "6f70144f-8d7b-4d18-9e35-e30a0d95b927",
    "policyName": "ActivateAccountPolicy",
    "description": "Default policy for Activate Account",
    "status": "active",
    "matchAnyApp": false,
    "apps": [
        {
            "name": "SigninClient",
            "id": "$SIGNINCLIENT"
        }
    ],
    "rules": [
        {
            "id": "9ad72abb-7518-413d-9ecb-131c28acb913",
            "conditions": {
                "principal": {
                    "user": {
                        "operator": "in",
                        "value": [
                            "internal/*"
```

```
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "activateaccount"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web",
                        "mobile"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "true",
                "obligations": [
                    "SMSOTP:1",
                    "EMAILOTP:1"
                ],
                "msg": "Auto provisioned login rule for all users",
                "inlineEnrollmentAllowed": "false"
            }
        }
    ],
    "adminscope.criteria": [],
    "adminscope.managedby": [],
    "createdDate": "2025-12-20 07:42 PM UTC",
    "updatedDate": "2025-12-20 07:42 PM UTC",
    "preventedOperations": "Delete",
    "policySubType": "systemAccountRecoveryActivate"
}
```


Unlock Account Policy

Learn about the out-of-the-box UnlockAccountPolicy.

The "Unlock Account Policy" is a pre-configured, out-of-the-box solution that is designed to govern how users authenticate themselves when unlocking an account. This policy cannot be deleted.

This policy allows you to set which credential types users can use to authenticate when unlocking an account. By default, this policy is pre-configured to use SMS and email as the security code delivery method. This policy is also assigned to all users. However, you can modify these settings as required.

> **NOTE**
> The account unlock flow does not allow inline credential registration. Make sure to set credential types that your users already have registered for their use.

This policy applies only to your sign-in client app. You cannot change the application that is assigned to this policy. Also, you cannot add any custom actions to this policy.

Set the **Account Unlock Flow Type** option to **Policy** in the internal Identity Store to set this policy as the default. See Internal Identity Store Configuration.

```
{
    "policyId": "70db1d84-42fa-4acb-bc25-489c449089e7",
    "policyName": "UnlockAccountPolicy",
    "description": "Default policy for Unlock Account",
    "status": "active",
    "matchAnyApp": false,
    "apps": [
        {
            "name": "SigninClient",
            "id": "$SIGNINCLIENT"
        }
    ],
    "rules": [
        {
            "id": "04b55813-1dcf-45bd-9eff-72e32650275a",
            "conditions": {
                "principal": {
                    "user": {
                        "operator": "in",
                        "value": [
                            "internal/*"
                        ]
                    }
                },
                "action": {
                    "operator": "in",
                    "value": [
                        "unlockaccount"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web",
                        "mobile"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "true",
                "obligations": [
                    "SMSOTP:1",
                    "EMAILOTP:1"
                ],
                "msg": "Auto provisioned login rule for all users",
                "inlineEnrollmentAllowed": "false"
            }
```

```
        }
    ],
    "adminscope.criteria": [],
    "adminscope.managedby": [],
    "createdDate": "2025-12-20 07:42 PM UTC",
    "updatedDate": "2025-12-20 07:42 PM UTC",
    "preventedOperations": "Delete",
    "policySubType": "systemAccountRecoveryActivate"
}
```

Reset Password Policy

Learn about the out-of-the-box ResetPasswordPolicy.

The "Reset Password Policy" is a pre-configured, out-of-the-box solution that is designed to govern how users authenticate themselves when resetting a password. This policy cannot be deleted.

This policy allows you to set which credential types users can use to authenticate when resetting a password. By default, this policy is pre-configured to use SMS and email as the security code delivery method. This policy is also assigned to all users. However, you can modify these settings, as required.

> **NOTE**
> The reset password flow does not allow inline credential registration. Make sure to set credential types that your users already have registered for their use.

You cannot add any custom actions to this policy.

Set the **Reset Password Flow Type** option to **Policy** in the internal Identity Store Reworded "default" setting to set this policy as the default. See Internal Identity Store Configuration.

```
{
    "policyId": "6b6ad821-7e6a-466b-b691-091698353fb1",
    "policyName": "ResetPasswordPolicy",
    "description": "Default policy for Reset Password",
    "status": "active",
    "matchAnyApp": false,
    "apps": [
        {
            "name": "SigninClient",
            "id": "$SIGNINCLIENT"
        }
    ],
    "rules": [
        {
            "id": "3f1941fb-2164-4ec2-be6a-b2c99ba32b4f",
            "conditions": {
                "principal": {
                    "user": {
                        "operator": "in",
                        "value": [
                            "internal/*"
                        ]
                    }
                },
                "action": {
```

```
                    "operator": "in",
                    "value": [
                        "resetpassword"
                    ]
                },
                "channel": {
                    "operator": "in",
                    "value": [
                        "web",
                        "mobile"
                    ]
                }
            },
            "result": {
                "effect": "allow",
                "reAuthenticate": "true",
                "obligations": [
                    "SMSOTP:1",
                    "EMAILOTP:1"
                ],
                "msg": "Auto provisioned login rule for all users",
                "inlineEnrollmentAllowed": "false"
            }
        }
    ],
    "adminscope.criteria": [],
    "adminscope.managedby": [],
    "createdDate": "2025-12-20 07:42 PM UTC",
    "updatedDate": "2025-12-20 07:42 PM UTC",
    "preventedOperations": "Delete",
    "policySubType": "systemAccountRecoveryActivate"
}
```


Managing Signing Keys

Learn about retrieving and updating a tenant's Signing Keys.

Signing Key management allows an administrator to manage the private key that is used to sign the tokens (Access Token and ID Token) issued by Symantec Identity Security Platform. With the Signing Key API's, the tenant administrator can update the private key, and export the corresponding public certificate. Each tenant has an associated single signing key. A tenant administrator can update a tenant's existing private and public keys. A pair of private and public signing key is seeded into the solution's database when you deploy IDSP. Separate signing keys are seeded for system and tenant administrators.

This topic includes the following sections:

- Signing Key APIs
- Signing Key Request and Response of GET and PUT Operation
- Retaining Previous Signing Keys
- Steps to update a Signing Key
- Steps to extract certificate and validate the certificate using openSSL
- Implementing Graceful Signing Key Updates

Signing Key APIs

Use the Signing Key APIs for the following operations:

- Update the existing private and public keys for the tenant.

The following table lists the SigningKey APIs:

Operation Description	Operation Type	Endpoint URLs
Invoke this API to retrieve the signing key public certificate	GET	https://${SSP_FQDN}/<tenantName>/admin/v1/Signing
Invoke this API to update the existing signing key of a tenant	PUT	https://${SSP_FQDN}/<tenantName>/admin/v1/Signing
Invoke this API to update the existing signing key of a tenant with a cert chain	PUT	https://${SSP_FQDN}/<tenantName>/admin/v1/Signing
Invoke this API to publish the 'next' signing certificate data, making it available through SAML metadata and JWKS endpoints for clients to acquire before the actual key update.	PATCH	https://${SSP_FQDN}/<tenantName>/admin/v1/Signing

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace {{text}} with the appropriate values in all the endpoints shown in the following table. For example:
  `https://${SSP_FQDN}/<tenantName>/admin/v1/SigningCert`
  should look similar to:
  `https://example.com/default/admin/v1/SigningCert`

Signing Key Request and Response

GET operation of Signing Cert

Using the GET operation you can retrieve the signing public certificate.

```
GET https://${SSP_FQDN}/<tenantName>/admin/v1/SigningCert
```

API Response:


```
{
"b64CertData": "MIIDMjCCAhqgAwIBA........",
"kid": "SigningKey",
"name": "SigningKey",
"signingKeyId": "635b4b7c-1c03-4868-a99d-26446d373d49",
"x5t": "Ly268cXsgyM2QqVwnTjh-C3_vRs",
"createdDateTime": "2024-03-18 01:20 AM UTC",
"issuerDN": "CN=default,OU=SED,O=Xcom",
"subjectDN": "CN=default,OU=SED,O=Xcom",
"serialNumber": "63A21A00331A19B7F9731B198FBCC9C2",
"validFrom": "2024-03-18 01:20 AM UTC",
"validTo": "2026-12-13 01:20 AM UTC",
"keyAlgorithm": "SHA256WITHRSA",
"x5t#S256": "v1S3HNugyxcczX-Y2cXflA1BZEm84uCmileanqdtSoc"
}
```


Where,

- b64CertData - Base64 encoded PEM formatted Public key/Certificate data.
- kid - key identifier of the certificate.
- name - name of the certificate
- signingKeyId - Unique identifier for the signing key.
- x5t - Thumbprint of the certificate (SHA-1).
- createdDateTime - Date and time when the key or certificate was created.
- issuerDN - Distinguished Name (DN) of the issuer of the certificate.
- subjectDN - Distinguished Name (DN) of the subject of the certificate.
- serialNumber - Serial number of the certificate.
- validFrom - Start date of the certificate's validity period.
- validTo - End date of the certificate's validity period.
- keyAlgorithm - Algorithm used by the key for signing (for example, - SHA256WITHRSA).
- x5t#S256 - Thumbprint of the certificate (SHA-256).

Put operation of Signing Key API

Using the PUT operation, you can update the existing signing key of a tenant.

```
PUT https://${SSP_FQDN}/<tenantName>/admin/v1/SigningKey
```

API Request:

```
{
"kid": "ssp-signingkey",
"b64CertData": "MIIDRjCCAi4CCQCxqZKNu8fplzANB........",
"b64KeyData": "MIIEvQIBADANBgkqhkiG9w0BAQEFAA........"
}
```

Where,

- kid - The key identifier parameter provides a unique identifier for the specific cryptographic key used for signing.
- b64CertData - This is the Base64 encoded PEM Cert. To know how to get this data, see here.
- b64KeyData - This is the Base64 encoded PEM Key. To know how to get this data, see here.

Retaining the Previous Signing Keys

This mechanism ensures a seamless transition during key rollovers by keeping the old signing key active for a defined "grace period." During this period, systems can continue to validate tokens issued with the older key while gradually transitioning to the new one. During JWT token validation, the system initially attempts verification or decryption with the current signing key pair. If this primary attempt is unsuccessful, it then gracefully falls back to utilizing the previous signing key pair for validation.

This approach enhances system resilience, prevents service interruptions, and allows dependent applications ample time to update to the new key without immediate impact, ultimately improving the overall reliability and user experience of services relying on these cryptographic operations.

Settings Configurations:

Previous signing key retention is managed through two global/tenant settings with default configurations. These default values are adjustable to accommodate different operational needs.

Settings	Description
successiveSigningKeyUpdateIntervalMins	This is a global setting that mitigates successive calls to the defined time interval. Default value is 30 minutes.

gracePeriodToRetainPreviousSigningKeyDays	This is a tenant setting that defines how many days previous verification after a key update, ensuring continuity during ke Default value is 3 days.

These settings are also covered in the Manage Settings topic.

Signing Key APIs:

- POST ".../admin/v1/SigningKey": Used for initial update with a new signing key pair.
- PATCH ".../admin/v1/SigningKey": Used to update the previous signing key pair if the signing key is already updated.

The following table depicts the lifecycle and rollover process of signing keys within IDSP, specifically highlighting how different key pairs are managed and used for token verification at various stages.

Scenario	Process	Token Verification/Decryption	JWK Entries
Initial Setup	Deployment or Create tenant flow generates a key pair. Current key pair • kid = "signingkey" • encryption kid = "EncryptionKey"	Verification is done with one key pair	2 entries • Current key pair
Adding next cert	Update NextCert with NBF before current key expiration. Current key pair, Next key pair Current key pair • kid = "signingkey" • encryption kid = "EncryptionKey" Next key pair • nextkid = "New-signingkey" • nextencryption kid = "NewEncryptionKey"	Verification is done only with one key (existing key pair)	4 entries • 2 Current key pair • 2 Next key pair
Updating signing key (Rollover)	When the current key is expired/compromised, update the signing key. New Private key and Certificate (which is the next cert from the previous stage) will become the current. NextCert will be blanked out. Existing keys saved as PrevPrivKey and PrevCertificate. • kid and encrkid are copied as prevkid and prevencrkid • nextkid, nextencrkid are copied to kid and encrkid Current key pair (new): • kid = "New-signingkey" • encryption kid = "New-EncryptionKey" Previous key pair: • prevkid = "signingkey" • prevencryption kid = "EncryptionKey"	Verify/Decrypt with the new keys and fall back to previous keys in case of failure.	4 entries • 2 New (Current) key pa • 2 Previous key pair

Updating next cert (before rollover)	This state occurs when a new NextCert is added while the PrevCert is still active during its grace period. Grace period to retain previous keys is configurable in the tenant setting. Current key pair: • kid = "New-signingkey" • encryption kid = "New-EncryptionKey" Previous key pair: • prevkid = "signingkey" • prevencryption kid = "EncryptionKey" Next key pair: • nextkid = "Next-signingkey" • nextencryption kid = "Next-EncryptionKey"	Verify/Decrypt with the new keys and fall back to previous keys in case of failure.	6 entries • 2 Current key pair • 2 Previous key pair • 2 Next key pair
Removing next cert	Remove the next certificate if the planned rollover is not going to take place. No previous keys will be present. Current key pair: • kid = "signingkey" • encryption kid = "EncryptionKey"	Verification is done with one key pair	2 entries • Current key pair
Updating next cert	Update the next certificate with NBF before the current key is rolled over	*no prev keys will be present*	

Steps to update a Signing Key

IDSP maintains a dedicated signing key for each tenant. During deployment, a self signed private - public key pair gets seeded into the system.
If you intend to update the Signing Key, you must obtain the "***b64CertData***," and the "***b64KeyData***" values. The next section lets you perform the steps needed to get these values.

Follow these steps:

1. Obtain the Tenant Admin Access Token.
2. To obtain the **b64KeyData** value that is required to be updated in the ".../admin/v1/SigningKey" API, perform the following steps:
   a. Obtain the un-encrypted private key in PEM format, as shown in the following example:

```
-----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAL5c8+80sSkWmBR3
iet0TSrugNuEb14AoH72aa9oSN90YmRDHX8VNeKfH5mafsKi638TkuKxmtBnqHmp
6JN2Ztgw9NLr/aBKuP5h5po5rdlpuvQnFUFiuZ60RRDMJUcI62j2KZ7pnLiiiJn2
ALu8aVzCqX0cU/ZqGrPnoYRWB7QjAgMBAAECgYEAgr9rWA+z4sm++MfF9e9ylRst
bBjBOSprEekDW9PfwYK5la23U7LoyNeMJ79Cc0LdRvHVwFhGQgAAalf2A1qedY2T
L0rCWF4SbqmxZIQHfYKTU7jY7Bb6yfFQqTxMAW54WRnkano8Uumjfrw2DboAS8kg
uRO0e+/1dcHHvGLJ03kCQQDhOhtwYXZRCGeZs5Xh6L/L1DxcCjyHjCrzq32abqil
DmQbJ7m1HM0tP+qdJL2teJVviGyXxqve1kIBS7xkyPrHAkEA2F9kRehMZGaWit4i
NbndXCy3Ad6/8zZo9oo9MSBObLmvwYC67ZYn8qmiID0C8YJIRmGS48ga41fuWMEE
z1p/xQJAV3Eb3hG4wdUlg8L5We/nJteLZ6CBx45fzqKx7RcxWoOv0esmam54Ki06
hTAShBrgyaCJqJN1ftpoXeBtkCOHWQJAEnBmqjJ2uCOp9qGhhrSVMbK50dX6w65C
WahEOLiJFmjomnPu+ipOszZTk62/X3d700ujaff2mqM2s6eXUlhjBQJANzPRz4d/
jH0fjjT//0dtfrkFHi/r2Jiq/1A9DVdInE7/SFVAVB63qVE5D28vkNaApZ9mkrcx
PXfQ26C1Y30O9Q==
```

```
-----END PRIVATE KEY-----
```

b. Change the following in the un-encrypted key:
   a. Remove the header, footer tags, and the line breaks.
   b. The value that you have obtained from the previous step would looks like the following sample:

      NOTE
      The following key that is obtained can be used with or without encoding it in base64 format.

```
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAL5c8+80sSkWmBR3iet0TSrugNuEb14AoH72aa9oSN90YmRDHX8VNeKfH5maf
aBKuP5h5po5rdlpuvQnFUFiuZ60RRDMJUcI62j2KZ7pnLiiiJn2ALu8aVzCqX0cU/ZqGrPnoYRWB7QjAgMBAAECgYEAgr9rWA
+z4sm+
+MfF9e9ylRstbBjBOSprEekDW9PfwYK5la23U7LoyNeMJ79Cc0LdRvHVwFhGQgAAalf2A1qedY2TL0rCWF4SbqmxZIQHfYKTU7jY7Bb6yfFQq
+/1dcHHvGLJ03kCQQDhOhtwYXZRCGeZs5Xh6L/L1DxcCjyHjCrzq32abqilDmQbJ7m1HM0tP
+qdJL2teJVviGyXxqve1kIBS7xkyPrHAkEA2F9kRehMZGaWit4iNbndXCy3Ad6/8zZo9oo9MSBObLmvwYC67ZYn8qmiID0C8YJIRmGS48ga41
xQJAV3Eb3hG4wdUlg8L5We/
nJteLZ6CBx45fzqKx7RcxWoOv0esmam54Ki06hTAShBrgyaCJqJN1ftpoXeBtkCOHWQJAEnBmqjJ2uCOp9qGhhrSVMbK50dX6w65CWahEOLiJ
+ipOszZTk62/X3d700ujaff2mqM2s6eXUlhjBQJANzPRz4d/jH0fjjT//0dtfrkFHi/r2Jiq/1A9DVdInE7/
SFVAVB63qVE5D28vkNaApZ9mkrcxPXfQ26C1Y30O9Q==
```

   c. Optionally, you can run this command to perform the steps mentioned in b(1) and (ii):
      - **Using OpenSSL**:

```
openssl x509 -in your_certificate.crt -outform DER | openssl base64 -A
```

      - **Linux and macOS**:

```
cat your_certificate.crt | grep -v "\-\-\-\-\-" | tr -d '\n'
```

      or

```
sed '/-----/d'  your_certificate.crt  | tr -d '\n'
```

      - **Windows (Using PowerShell)**:

```
(Get-Content -Path your_certificate.crt | Where-Object { $_ -notmatch '^-' }) -join ''
```

   d. Use this value as the b64KeyData in the ".../admin/v1/SigningKey" API request.

3. To obtain the **b64CertData** value that is required to be updated in the ".../admin/v1/SigningKey" API, perform the following steps:
   a. Concatenate the Signing Key public certificate, followed by the Root CA as shown in this sample and save it, for example, as "SigningCertificateChain.pem"

      NOTE
      While merging the Signing Key public certificate, followed by the Root CA, if there are any intermediate server certificates, the order of the chain starts with your server certificate at the beginning, then follow by the intermediate certificate(s) and end up with the root certificate.

```
-----BEGIN CERTIFICATE-----
MIIDizCCAnOgAwIBAgIUCQ5lXP6cPNl92QJNzjqo/Aa0em4wDQYJKoZIhvcNAQEL
BQAwVTELMAkGA1UEBhMCSU4xCzAJBgNVBAgMAkFQMQwwCgYDVQQHDANIWUQxDTAL
BgNVBAoMBHJvb3QxDTALBgNVBAsMBHJvb3QxDTALBgNVBAMMBHJvb3QwHhcNMjQw
MzEyMTA1NzQzWhcNMjYxMjMxMTA1NzQzWjBVMQswCQYDVQQGEwJJTjELMAkGA1UE
CAwCQVAxDDAKBgNVBAcMA0hZRDENMAsGA1UECgwEcm9vdDENMAsGA1UECwwEcm9v
dDENMAsGA1UEAwwEcm9vdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
AKNqdYbVCwqkBYrC0pyHH1OZ23pWeh0ZbE3iPmEJra8KNsVHGdR8zFPXG4iSgMDT
4d5Ks/xSDn7vmtow3IwJOMktrQAyv6AJbaeggQp0CmMge7fhpn0Mf10DIkmdfd+9
TgvbYqREqs51Uxoyv7axaUvF6GU4cYOaUo7ZKvoD4xGhQwuJtpR3oQRnN8QIrBda
3U+iR4XRicpy99l60/BMsQrs9mchYbKwhF8mqQVsXch0nEybjIEz5OGiQCZxVRd+
vHFAT1l3brGGMkxE6D2i5YnK8y047gvoJ1HLRejeUPMASH9gc5KoNn+DvTUERJfc
x54coc8ToikAQSEId579gtUCAwEAAaNTMFEwHQYDVR0OBBYEFMgdMlEPNDOy+z39
ei2xX0OaSm2ZMB8GA1UdIwQYMBaAFMgdMlEPNDOy+z39ei2xX0OaSm2ZMA8GA1Ud
EwEB/wQFMAMBAf8wDQYJKoZIhvcNAQELBQADggEBACCN9O7CQVSjbS0hL6/55aJI
5pfDw0PuDe37Cws3MPhZ+MPPYD6og7z9cvJE041C2qZbAfM4CmQv5/coInYZ2YZo
```

```
itAfXIpfUvtv72Hih5jxQLNg6Vi2HLYXBl12SxYSiJv2jAp361+F4Fgtjv8vLPP2
1A2CxRvS91OqB5K3UR3qqxzh+NkyF7wmp0jRfK63kWdHAdnm/dvXQiQDBcMhhnbh
OoJmUlbhG2/0/zedixEB9lpnQlF8JoolbwWl9QSjeiC+qEoKlSPcpKk2Iql5Gbbz
tPMYD+PMeBmJFfFsu0bnL3AU/SJoIWN0NigrrYBmYmOs8ymay5vQ2SRFmVhFB/U=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIC4TCCAcmgAwIBAgIBAjANBgkqhkiG9w0BAQsFADBVMQswCQYDVQQGEwJJTjEL
MAkGA1UECAwCQVAxDDAKBgNVBAcMA0hZRDENMAsGA1UECgwEcm9vdDENMAsGA1UE
CwwEcm9vdDENMAsGA1UEAwwEcm9vdDAeFw0yNDAzMTIxMDU4MjZaFw0yNDA2MjAx
MDU4MjZaMEgxCzAJBgNVBAYTAklOMQswCQYDVQQIDAJBUDENMAsGA1UECgwEcm9v
dDENMAsGA1UECwwEcm9vdDEOMAwGA1UEAwwFY2VydDEwgZ8wDQYJKoZIhvcNAQEB
BQADgY0AMIGJAoGBAL5c8+80sSkWmBR3iet0TSrugNuEb14AoH72aa9oSN90YmRD
HX8VNeKfH5mafsKi638TkuKxmtBnqHmp6JN2Ztgw9NLr/aBKuP5h5po5rdlpuvQn
FUFiuZ60RRDMJUcI62j2KZ7pnLiiiJn2ALu8aVzCqX0cU/ZqGrPnoYRWB7QjAgMB
AAGjTTBLMAkGA1UdEwQCMAAwHQYDVR0OBBYEFPF+0zUPB2Mgnfpe8GXhF31RGjcg
MB8GA1UdIwQYMBaAFMgdMlEPNDOy+z39ei2xX0OaSm2ZMA0GCSqGSIb3DQEBCwUA
A4IBAQBkixGiTva8YgZJLmrzTIWwK94WRW4gEtNdHKdLKh9ZgUq5Jf2gwmqISXOB
q4ZTfLN9bWn0bN5Rkrpn6XqmgG20iD6Z9GENl2D9AgiWV1QrWawGXKMtBf1zSmt
OtW5ef9zJRZ87t2FWCP3JCO8TSwekS0pIcCWMtJv001SnBf548klxDDtsSE8XjN5
sZwZGH1JXF7sG2ahc7L7D5KPd49hJAEK21BcUOc8bH9APviLRNVtezGLkk8h0yCp
xdGuAHxvWfRRcjnH0p76yg7GHDyxhimEGu6QuDbrMyafgWQDx3Y5xYJfIYHItFQ/
a83fEzxcqR3RccFGr/ILZ41/wlbB
-----END CERTIFICATE-----
```

b. Encode the result in Base64 format by using the following command:

```
openssl base64 -in SigningCertificateChain.pem -out SigningCertificateChain.base64 -A
```

NOTE
If you do not have openssl, you can use any one of the following commands based on the operating system you are using:

- For Linux:
  ```
  base64 -w 0 SigningCertificateChain.pem > SigningCertificateChain.base64
  ```
- For Mac:
  ```
  base64  SigningCertificateChain.pem > SigningCertificateChain.base64
  ```
- For Windows:
  - powershell:
    ```
    [System.Convert]::ToBase64String([IO.File]::ReadAllBytes("c:\path
    \SigningCertificateChain.pem")) > .\SigningCertificateChain.base64
    ```
  - certutil:
    ```
    certutil -encode SigningCertificateChain.pem tmp.b64 && findstr /v /c:- tmp.b64 >
     SigningCertificateChain.base64
    ```

c. The value that you have obtained from the previous step would looks like the following sample:

```
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM0VENDQWNtZ0F3SUJBZ0lCQWpBTkJna3Foa2lHOXcwQkFRc0ZBREJWTVFzd0NRWURWUVFH
```

d. This value should be used in the ".../admin/v1/SigningCert" API request.

4. Run the following PUT call to update the Signing key:

```
PUT https://${SSP_FQDN}/<tenantName>/admin/v1/SigningKey
```

API Request:

```
{
"kid": "ssp-signingkey",
"b64CertData": "LS0tLS1CRUdJTiBDRVJUSUZJQ0........",
"b64KeyData": "MIICdgIBADANBgkqhkiG9w0BAQA........"
}
```

Where, the values of "b64CertData" and "b64KeyData" are obtained from steps, 2 and 3.

NOTE
Signing key rollover is not supported, meaning that when the **SigningKey** is changed, all previously issued access and identity tokens become invalid. As a result, you must obtain a new access token to continue performing any operations after the signing key has been updated.

Reference Links to:

- For sample request and response payloads of all the URLs mentioned in the above table, refer to SigningKey API Swagger documentation
- SigningKey API Error Codes

Steps to extract Signing Certificate information, and print it in the console

The following steps enable you to extract certificate information and print it in the terminal/commandPrompt console.

1. Perform a GET of ".../admin/v1/SigningCert" API. The "b64CertData" attribute value obtained from this step would looks like the following sample:

```
MIIC4TCCAcmgAwIBAgIBAjANBgkqhkiG9w0BAQsFADBVMQswCQYDVQQGEwJJTjELMAkGA1UECAwCQVAxDDAKBgNVBAcMA0hZRDENMAsGA1UECgwEcm9v
aBKuP5h5po5rdlpuvQnFUFiuZ60RRDMJUcI62j2KZ7pnLiiiJn2ALu8aVzCqX0cU/
ZqGrPnoYRWB7QjAgMBAAGjTTBLMAkGA1UdEwQCMAAwHQYDVR0OBBYEFPF
+0zUPB2Mgnfpe8GXhF31RGjcgMB8GA1UdIwQYMBaAFMgdMlEPNDOy
+z39ei2xX0OaSm2ZMA0GCSqGSIb3DQEBCwUAA4IBAQBkixGiTva8YgZJLmrzTIWwK94WRW4gEtNdHKdLKh9ZgUq5Jf2gwmqISXOBq4ZTfLN9bWn0bN5F
a83fEzxcqR3RccFGr/ILZ41/wlbB
```

2. Copy the "b64CertData" attribute value obtained in the previous step as this value is utilized in the next step.
3. Create a file with the name "xyz.cert" with the header, and footer tags, and the b64CertData in between.

```
-----BEGIN CERTIFICATE-----
MIIC4TCCAcmgAwIBAgIBAjANBgkqhkiG9w0BAQsFADBVMQswCQYDVQQGEwJJTjELMAkGA1UECAwCQVAxDDAKBgNVBAcMA0hZRDENMAsGA1UECgwEcm9v
aBKuP5h5po5rdlpuvQnFUFiuZ60RRDMJUcI62j2KZ7pnLiiiJn2ALu8aVzCqX0cU/
ZqGrPnoYRWB7QjAgMBAAGjTTBLMAkGA1UdEwQCMAAwHQYDVR0OBBYEFPF
+0zUPB2Mgnfpe8GXhF31RGjcgMB8GA1UdIwQYMBaAFMgdMlEPNDOy
+z39ei2xX0OaSm2ZMA0GCSqGSIb3DQEBCwUAA4IBAQBkixGiTva8YgZJLmrzTIWwK94WRW4gEtNdHKdLKh9ZgUq5Jf2gwmqISXOBq4ZTfLN9bWn0bN5F
a83fEzxcqR3RccFGr/ILZ41/wlbB
-----END CERTIFICATE-----
```

4. Run the following openssl command to print the certificate information:

```
openssl x509 -in xyz.cert -text -noout
```

Response:

```
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 2 (0x2)
Signature Algorithm: sha256WithRSAEncryption
Issuer: C=IN, ST=AP, L=HYD, O=root, OU=root, CN=root
Validity
Not Before: Mar 12 10:58:26 2024 GMT
Not After : Jun 20 10:58:26 2024 GMT
Subject: C=IN, ST=AP, O=root, OU=root, CN=cert1
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (1024 bit)
Modulus:
00:be:5c:f3:ef:34:b1:29:16:98:14:77:89:eb:74:
4d:2a:ee:80:db:84:6f:5e:00:a0:7e:f6:69:af:68:
```

```
48:df:74:62:64:43:1d:7f:15:35:e2:9f:1f:99:9a:
7e:c2:a2:eb:7f:13:92:e2:b1:9a:d0:67:a8:79:a9:
e8:93:76:66:d8:30:f4:d2:eb:fd:a0:4a:b8:fe:61:
e6:9a:39:ad:d9:69:ba:f4:27:15:41:62:b9:9e:b4:
45:10:cc:25:47:08:eb:68:f6:29:9e:e9:9c:b8:a2:
88:99:f6:00:bb:bc:69:5c:c2:a9:7d:1c:53:f6:6a:
1a:b3:e7:a1:84:56:07:b4:23
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Basic Constraints:
CA:FALSE
X509v3 Subject Key Identifier:
F1:7E:D3:35:0F:07:63:20:9D:FA:5E:F0:65:E1:17:7D:51:1A:37:20
X509v3 Authority Key Identifier:
C8:1D:32:51:0F:34:33:B2:FB:3D:FD:7A:2D:B1:5F:43:9A:4A:6D:99
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
64:8b:11:a2:4e:f6:bc:62:06:49:2e:6a:f3:4c:85:b0:2b:de:
16:45:6e:20:12:d3:5d:1c:a7:4b:2a:1f:59:81:4a:b9:25:fd:
a0:c2:6a:88:49:73:81:ab:86:53:7c:b3:7d:6d:69:f4:6c:de:
51:92:ba:67:e9:7a:a6:80:6d:b4:3a:20:fa:67:d1:84:36:5d:
83:f4:08:22:59:5d:50:ad:66:b0:19:72:8c:b4:17:f5:cd:29:
ad:3a:d5:b9:79:ff:73:25:16:7c:ee:dd:85:58:23:f7:24:23:
bc:4d:2c:1e:91:2d:29:21:c0:96:32:d2:6f:d3:4d:52:9c:17:
f9:e3:c9:25:c4:30:ed:b1:21:3c:5e:33:79:b1:9c:19:18:7d:
49:5c:5e:ec:1b:66:a1:73:b2:fb:0f:92:8f:77:8f:61:24:01:
0a:db:50:5c:50:e7:3c:6c:7f:40:3e:f8:8b:44:d5:6d:7b:31:
8b:92:4f:21:d3:20:a9:c5:d1:ae:00:7c:6f:59:f4:51:72:39:
c7:d2:9e:fa:ca:0e:c6:1c:3c:b1:86:29:84:1a:ee:90:b8:36:
eb:33:26:9f:81:64:03:c7:76:39:c5:82:5f:21:81:c8:b4:54:
3f:6b:cd:df:13:3c:5c:a9:1d:d1:71:c1:46:af:f2:0b:67:8d:
7f:c2:56:c1
```

Implementing Graceful Signing Key Updates

The graceful signing key feature allows clients to acquire the "next" signing certificate before the actual signing key change is made. This ensures a smooth transition for clients validating artifacts signed by the IDSP Signing Key.

This process involves two main steps:

1. **Publishing the "Next" Certificate Data:**
   Use the PATCH operation of the Signing Key API to provide the data for the "next" certificate. This makes the "next" certificate available via SAML metadata and JWKS endpoints before the actual key update.

   ```
   PATCH https://${SSP_FQDN}/<tenantName>/admin/v1/SigningKey
   ```

   API Request:

   ```
   {
   "b64NextCertData": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURhVENDQWxHZ0F3....",
   "nextCertNbfDate": "2026-10-25 05:36 AM UTC"
   "nextCertKid": "New_SigningKey",
   "nextCertEncrKid": "New_EncryptionKey"
   }
   ```

   Where:

- `b64NextCertData` : Base64 encoded PEM formatted Public key/Certificate data of the next signing certificate. This can include an optional certificate chain.
- `nextCertNbfDate` : The 'not before' date for the next signing certificate, indicating when it becomes valid.
- `nextCertKid` : Specifies the Key ID (kid) for the new signing certificate that will be used after the key update. The kid serves as a unique identifier for a cryptographic key. In this example, "New_SigningKey" is the identifier chosen for the upcoming signing key.
- `nextCertEncrKid` : Specifies the Key ID (kid) for the new encryption certificate that will be used after the key update. It uniquely identifies the encryption key. In this example, "New_EncryptionKey" is the identifier chosen for the upcoming encryption key.

Once this PATCH operation is applied, SAML Metadata and JWKS endpoints will start publishing this "next" certificate data. For JWKS, "nbf" will be `nextCertNbfDate` and "exp" will be the cert's expiry. For SAML Metadata, "not before" will be `nextCertNbfDate` and "not after" will be the cert's expiry.

Example of a JWKSL Response


```
{
"keys": [
{
"kty": "RSA",
"x5t#S256": "XS3QC7O32mCFGqjuYz40aQjKXF47PMOA1cDhr5iLJAM",
"nbf": 1707285478,
"e": "AQAB",
"use": "sig",
"x5t": "gqnB7g1Cwiv8jWAU-JE6Tu1B3vE",
"kid": "SigningKey",
"x5c": [
"MIIDaTCCAlGgAwIBAgIBATANBgkqhkiG9w0BAQsFADBVMQswCQYDVQQGEwJJTjELMAkGA1UECAwCQVAxDDAKBgNVBAcMA0hZRDEMMAoGA1UECgwDQkN
E8u8bcx4vGNcrK6uaaR+Fe2XCOzs96oLyQxgjcPKEUePDcE6cG32kvCIInaUE6yGGq1bmIT4RIWPHJ8Ah3EDBkJkx4k
+xgqTSEXiEALJLObaeAAUPW6Rgp3+9p9LefYLnC/
FxHP28Ogh95Uc14L4CAJLH07emrdC5hu3blr2kOEkZda6dj5RvSLDkXQJutSPMvqYl56ITNr7WQb3EgwdhYrkiCSZ6ABJj3wxUZMz325NR/67AQ1/
T83q9zPRtQZ8MiAuWOa9BKvdg2IjrrfSx3Y77OhWPUJGNkAcKvR4oP42zjn3kfe14BAK59APSMJAf0xuP6MwZmUUHwIDAQABo00wSzAJBgNVHRMEAjAA
ZUIXHVUxRDP9CJNFwn0JjAfBgNVHSMEGDAWgBR
+GEiCqzw62+G06glBVoTi4yHx2zANBgkqhkiG9w0BAQsFAAOCAQEAIBsBINteAnsTFJcRMPsDgYuvNDm8+bxe82T8Z7DAqgEYZOYtyWdSu/
Bm2Wvc1AiNh46Fiplg0sQo6WZHO43TDOkMdezuFYvILrw87VmcZyAxneOpn+Rl6qq5ntsWWilmSr9MAWshBCgoT2ro2rS/
E5RCA4ffvXKxSr1gOtCl317aFdIOKi1m2M78VMz2Mh8RhY8gZDDAPV4yzL
+JwXfSFMEOnY56IMYcEmeCqLlotGYX4bYYoCqoxic7URA85sqE3r1rmIJxRkeNnPIs6hAtcNKDontMTgNlx9fU
+NgnoDt85ZQbXuRTKujyFl1sgfxMaC73eWnaLHGhaKP7wxs0Pw==",
"MIIDizCCAnOgAwIBAgIUNSWT61lmCu1FHxrAfXowi4r43sYwDQYJKoZIhvcNAQELBQAwVTELMAkGA1UEBhMCSU4xCzAJBgNVBAgMAkFQMQwwCgYDVQQ
+eSvtX/WbZ8qQiL9Ci5oT9o3m64Xd/olG+lcSwNr+ptgTTpooStIe2m7JAzjXJ9XLkRbE+P+4LdRiC+Z1G0oc8T+mlQMBpRaPTKh/Tzec/
V/
hz1aCDPQPZ4iOTySun01ZwpEeOZksimuT7qkCUACHUQS78OSqq73WrZVUwkIUxRLJEptrNjRGinVybQTURQDBTxzqpFOZzbtfvzBVHSzHMqR1SsggvmU
wQFMAMBAf8wDQYJKoZIhvcNAQELBQADggEBAFM/6dlf0WRQTnwjOtGbOXIygMh9UlBaILol3DiaAIBy88ZU714NN/n
+Zkpr3xTMAfKpQZA4VLjrU39W6/ltwlCIvov
+zSzfeWUxuUqkSkriK8KygzybCx0CyVH6K8AmWliEfPhYzBcjPLcZA7+hG8hTWoXMIz6nhhoUIvEGvQRYJY/
a2AZxqmxoh7iR00REFwaMNdXoJWY85qyk8+kNA4qjm5eRtKwiDwIY57zxitjr2uJQrnwd/
N6SaFcEBlcXNEsisZfXtmCTSUQ3z1WCeAXJDdXDWqW3BP8rspHEDgdsyc9gacEKodI/7ivZR98kDv+yxeP25BDAuhuTjcvC4Ss="
],
"key_ops": [
"verify"
],
"exp": 2022645478,
"alg": "RS256",
"n": "vu2mSLr_E8u8bcx4vGNcrK6uaaR-Fe2XCOzs96oLyQxgjcPKEUePDcE6cG32kvCIInaUE6yGGq1bmIT4RIWPHJ8Ah3EDBkJkx4k-
xgqTSEXiEALJLObaeAAUPW6Rgp3-9p9LefYLnC_FxHP28Ogh95Uc14L4CAJLH07emrdC5hu3blr2kOEkZda6dj5RvSLDkXQJutSPMvqYl56ITNr7WQb3
```

```
},
{
"kty": "RSA",
"x5t#S256": "XS3QC7O32mCFGqjuYz40aQjKXF47PMOA1cDhr5iLJAM",
"nbf": 1707285478,
"e": "AQAB",
"use": "enc",
"x5t": "gqnB7g1Cwiv8jWAU-JE6Tu1B3vE",
"kid": "EncryptionKey",
"x5c": [
"MIIDaTCCAlGgAwIBAgIBATANBgkqhkiG9w0BAQsFADBVMQswCQYDVQQGEwJJTjELMAkGA1UECAwCQVAxDDAKBgNVBAcMA0hZRDEMMAoGA1UECgwDQkN
E8u8bcx4vGNcrK6uaaR+Fe2XCOzs96oLyQxgjcPKEUePDcE6cG32kvCIInaUE6yGGq1bmIT4RIWPHJ8Ah3EDBkJkx4k
+xgqTSEXiEALJLObaeAAUPW6Rgp3+9p9LefYLnC/
FxHP28Ogh95Uc14L4CAJLH07emrdC5hu3blr2kOEkZda6dj5RvSLDkXQJutSPMvqYl56ITNr7WQb3EgwdhYrkiCSZ6ABJj3wxUZMz325NR/67AQl/
T83q9zPRtQZ8MiAuWOa9BKvdg2IjrrfSx3Y77OhWPUJGNkAcKvR4oP42zjn3kfel4BAK59APSMJAf0xuP6MwZmUUHwIDAQABo00wSzAJBgNVHRMEAjAA
ZUIXHVUxRDP9CJNFwn0JjAfBgNVHSMEGDAWgBR
+GEiCqzw62+G06glBVoTi4yHx2zANBgkqhkiG9w0BAQsFAAOCAQEAIBsBINteAnsTFJcRMPsDgYuvNDm8+bxe82T8Z7DAqgEYZOYtyWdSu/
Bm2Wvc1AiNh46Fiplg0sQo6WZHO43TDOkMdezuFYvILrw87VmcZyAxneOpn+Rl6qq5ntsWWilmSr9MAWshBCgoT2ro2rS/
E5RCA4ffvXKxSr1gOtCl317aFdIOKi1m2M78VMz2Mh8RhY8gZDDAPV4yzL
+JwXfSFMEOnY56IMYcEmeCqLlotGYX4bYYoCqoxic7URA85sqE3r1rmIJxRkeNnPIs6hAtcNKDontMTgNlx9fU
+NgnoDt85ZQbXuRTKujyFl1sgfxMaC73eWnaLHGhaKP7wxs0Pw==",
"MIIDizCCAnOgAwIBAgIUNSWT61lmCu1FHxrAfXowi4r43sYwDQYJKoZIhvcNAQELBQAwVTELMAkGA1UEBhMCSU4xCzAJBgNVBAgMAkFQMQwwCgYDVQQ
+eSvtX/WbZ8qQiL9Ci5oT9o3m64Xd/olG+lcSwNr+ptgTTpooStIe2m7JAzjXJ9XLkRbE+P+4LdRiC+Z1G0oc8T+mlQMBpRaPTKh/Tzec/
V/
hz1aCDPQPZ4iOTySun01ZwpEeOZksimuT7qkCUACHUQS78OSqq73WrZVUwkIUxRLJEptrNjRGinVybQTURQDBTxzqpFOZzbtfvzBVHSzHMqR1SsggvmU
wQFMAMBAf8wDQYJKoZIhvcNAQELBQADggEBAFM/6dlf0WRQTnwjOtGbOXIygMh9UlBaILol3DiaAIBy88ZU714NN/n
+Zkpr3xTMAfKpQZA4VLjrU39W6/ltwlCIvov
+zSzfeWUxuUqkSkriK8KygzybCx0CyVH6K8AmWliEfPhYzBcjPLcZA7+hG8hTWoXMIz6nhhoUIvEGvQRYJY/
a2AZxqmxoh7iR00REFwaMNdXoJWY85qyk8+kNA4qjm5eRtKwiDwIY57zxitjr2uJQrnwd/
N6SaFcEBlcXNEsisZfXtmCTSUQ3z1WCeAXJDdXDWqW3BP8rspHEDgdsyc9gacEKodI/7ivZR98kDv+yxeP25BDAuhuTjcvC4Ss="
],
"key_ops": [
"encrypt"
],
"exp": 2022645478,
"alg": "RSA-OAEP-256",
"n": "vu2mSLr_E8u8bcx4vGNcrK6uaaR-Fe2XCOzs96oLyQxgjcPKEUePDcE6cG32kvCIInaUE6yGGq1bmIT4RIWPHJ8Ah3EDBkJkx4k-
xgqTSEXiEALJLObaeAAUPW6Rgp3-9p9LefYLnC_FxHP28Ogh95Uc14L4CAJLH07emrdC5hu3blr2kOEkZda6dj5RvSLDkXQJutSPMvqYl56ITNr7WQb3
},
{
"kty": "RSA",
"x5t#S256": "XS3QC7O32mCFGqjuYz40aQjKXF47PMOA1cDhr5iLJAM",
"nbf": 1759210560,
"e": "AQAB",
"use": "sig",
"x5t": "gqnB7g1Cwiv8jWAU-JE6Tu1B3vE",
"kid": "SigningKey_NextCert",
"x5c": [
"MIIDaTCCAlGgAwIBAgIBATANBgkqhkiG9w0BAQsFADBVMQswCQYDVQQGEwJJTjELMAkGA1UECAwCQVAxDDAKBgNVBAcMA0hZRDEMMAoGA1UECgwDQkN
E8u8bcx4vGNcrK6uaaR+Fe2XCOzs96oLyQxgjcPKEUePDcE6cG32kvCIInaUE6yGGq1bmIT4RIWPHJ8Ah3EDBkJkx4k
+xgqTSEXiEALJLObaeAAUPW6Rgp3+9p9LefYLnC/
FxHP28Ogh95Uc14L4CAJLH07emrdC5hu3blr2kOEkZda6dj5RvSLDkXQJutSPMvqYl56ITNr7WQb3EgwdhYrkiCSZ6ABJj3wxUZMz325NR/67AQl/
T83q9zPRtQZ8MiAuWOa9BKvdg2IjrrfSx3Y77OhWPUJGNkAcKvR4oP42zjn3kfel4BAK59APSMJAf0xuP6MwZmUUHwIDAQABo00wSzAJBgNVHRMEAjAA
```

```
ZUIXHVUxRDP9CJNFwn0JjAfBgNVHSMEGDAWgBR
+GEiCqzw62+G06glBVoTi4yHx2zANBgkqhkiG9w0BAQsFAAOCAQEAIBsBINteAnsTFJcRMPsDgYuvNDm8+bxe82T8Z7DAqgEYZOYtyWdSu/
Bm2Wvc1AiNh46Fiplg0sQo6WZHO43TDOkMdezuFYvILrw87VmcZyAxneOpn+Rl6qq5ntsWWilmSr9MAWshBCgoT2ro2rS/
E5RCA4ffvXKxSr1gOtCl317aFdIOKi1m2M78VMz2Mh8RhY8gZDDAPV4yzL
+JwXfSFMEOnY56IMYcEmeCqLlotGYX4bYYoCqoxic7URA85sqE3r1rmIJxRkeNnPIs6hAtcNKDontMTgNlx9fU
+NgnoDt85ZQbXuRTKujyFl1sgfxMaC73eWnaLHGhaKP7wxs0Pw==
],
"key_ops": [
"verify"
],
"exp": 2022645478,
"alg": "RS256",
"n": "vu2mSLr_E8u8bcx4vGNcrK6uaaR-Fe2XCOzs96oLyQxgjcPKEUePDcE6cG32kvCIInaUE6yGGq1bmIT4RIWPHJ8Ah3EDBkJkx4k-
xgqTSEXiEALJLObaeAAUPW6Rgp3-9p9LefYLnC_FxHP28Ogh95Uc14L4CAJLH07emrdC5hu3blr2kOEkZda6dj5RvSLDkXQJutSPMvqYl56ITNr7WQb3
}
]
}
```


Example of a SAML Response


```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" cacheDuration="P29DT12H44M3.840S"
 entityID="https://<sspHost>/default" validUntil="2035-09-29T09:17:36.619Z">
<md:IDPSSODescriptor WantAuthnRequestsSigned="true"
 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
<md:KeyDescriptor use="signing">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
   <ds:X509Certificate>MIIDMjCCAhqgAwIBAgIQQCa3dG4iWjhTUv6+DCFExTANBgkqhkiG9w0BAQsFADAzMREwDwYDVQQK
DAhCcm9hZGNvbTEMMAoGA1UECwwDU0VEMRAwDgYDVQQDDAdkZWZhdWx0MB4XDTI1MDkyNTEyNTcx
MVoXDTI4MDYyMTEyNTcxMVowMzERMA8GA1UECgwIQnJvYWRjb20xDDAKBgNVBAsMA1NFRDEQMA4G
A1UEAwwHZGVmYXVsdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAJs21S+B7nir+jQj
mYX2orjyYegAwLh7INYRLWbOmDlvVcKg9dRrYZ7l2JgkPILJqekwHTxgNirsTy2P3lqvIrTSdA/T
dy2IQ3S6N6hzCG4a2GN6dfI7x5ruyv/tJTdE0Lla/X7+hPP0mHVnTQVf/nrxwtzvC5mCZ1LaAdck
oJ6MR3he3GG4QVbv0ZnOTYvCnAlj8pjh+KrbzAdKeFKe39Fc0DraqgHQIwzGrjV7uSR9He486Wo9
d8dCTT0uiF68/LbMseBtVZ12wPvQKIiS5ZeXvfHpg9hWrllNIyNsFg8Wf3mjY6fltarc034xKWMb
Yo7OJhBeErjSPpLaZQA+W6cCAwEAAaNCMEAwDAYDVR0TAQH/BAIwADAOBgNVHQ8BAf8EBAMCBaAw
IAYDVR0lAQH/BBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMA0GCSqGSIb3DQEBCwUAA4IBAQBsQW8q
oqkbsVbDltUx5pRI6HMoCX+hulVPG/NfMwIvhWzOofQ+jZ5avSuivsPf7V3BYk3lmh0Cka8Zrfsj
QK0KShPJyxCFZXKcADu6Zswg8sLz6rCmf8+2EVysMFhyuym7baAGvR5z3dIw3c+4zmCPcWlkCMUE
b4zCygzk2f6993n43EKW7Plu/rl3fy9mTmYTGpK/PSY6HuXgU3t1KbR/S4J1/D0047A3s9R7Xf00
cEoaG3/+hd2BARezKNz12ObWDBcfqa6w7ur20BrLj+DgALmb9m21/W7dSpjAQYyxCDklHD1Hp89O
8+5DcoaTwsVBbuq4WGws4BkVLKzmfdRn</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:KeyDescriptor use="encryption">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
   <ds:X509Certificate>MIIDMjCCAhqgAwIBAgIQQCa3dG4iWjhTUv6+DCFExTANBgkqhkiG9w0BAQsFADAzMREwDwYDVQQK
DAhCcm9hZGNvbTEMMAoGA1UECwwDU0VEMRAwDgYDVQQDDAdkZWZhdWx0MB4XDTI1MDkyNTEyNTcx
MVoXDTI4MDYyMTEyNTcxMVowMzERMA8GA1UECgwIQnJvYWRjb20xDDAKBgNVBAsMA1NFRDEQMA4G
A1UEAwwHZGVmYXVsdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAJs21S+B7nir+jQj
mYX2orjyYegAwLh7INYRLWbOmDlvVcKg9dRrYZ7l2JgkPILJqekwHTxgNirsTy2P3lqvIrTSdA/T
dy2IQ3S6N6hzCG4a2GN6dfI7x5ruyv/tJTdE0Lla/X7+hPP0mHVnTQVf/nrxwtzvC5mCZ1LaAdck
```

```
oJ6MR3he3GG4QVbv0ZnOTYvCnAlj8pjh+KrbzAdKeFKe39Fc0DraqgHQIwzGrjV7uSR9He486Wo9
d8dCTT0uiF68/LbMseBtVZ12wPvQKIiS5ZeXvfHpg9hWrllNIyNsFg8Wf3mjY6fltarc034xKWMb
Yo7OJhBeErjSPpLaZQA+W6cCAwEAAaNCMEAwDAYDVR0TAQH/BAIwADAOBgNVHQ8BAf8EBAMCBaAw
IAYDVR0lAQH/BBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMA0GCSqGSIb3DQEBCwUAA4IBAQBsQW8q
oqkbsVbDltUx5pRI6HMoCX+hulVPG/NfMwIvhWzOofQ+jZ5avSuivsPf7V3BYk3lmh0Cka8Zrfsj
QK0KShPJyxCFZXKcADu6Zswg8sLz6rCmf8+2EVysMFhyuym7baAGvR5z3dIw3c+4zmCPcWlkCMUE
b4zCygzk2f6993n43EKW7Plu/rl3fy9mTmYTGpK/PSY6HuXgU3t1KbR/S4J1/D0047A3s9R7Xf00
cEoaG3/+hd2BARezKNz12ObWDBcfqa6w7ur20BrLj+DgALmb9m21/W7dSpjAQYyxCDklHD1Hp89O
8+5DcoaTwsVBbuq4WGws4BkVLKzmfdRn</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:KeyDescriptor use="signing">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
   <ds:X509Certificate>MIID8TCCAtmgAwIBAgIUVQTIAc0qcU/0WfKbY+8tZVybjTowDQYJKoZIhvcNAQELBQAwgYcxCzAJ
BgNVBAYTAklOMRIwEAYDVQQIDAlUZWxhbmdhbmExEjAQBgNVBAcMCUh5ZGVyYWJhZDENMAsGA1UE
CgwEQlJDTTEMMAoGA1UECwwDU0JVMQswCQYDVQQDDAJWVjEmMCQGCSqGSIb3DQEJARYXYXRsdXJp
X3ZlbmthdEB5YWhvby5jb20wHhcNMjUwOTI0MDUxOTQ4WhcNMjgwNzE0MDUxOTQ4WjCBhzELMAkG
A1UEBhMCSU4xEjAQBgNVBAgMCVRlbGFuZ2FuYTESMBAGA1UEBwwJSHlkZXJhYmFkMQ0wCwYDVQQK
DARCUkNNMQwwCgYDVQQLDANTQlUxCzAJBgNVBAMMAlZWMSYwJAYJKoZIhvcNAQkBFhdhdGx1cmlf
dmVua2F0QHlhaG9vLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMGvz49yz3rX
9wE0DR0BeS2x1LF2SeYrxWDWcmtis5sWuk2fk7i4pfNpyZmu3W16pomWNIaUYWn8fBWvAmw7ZlsD
ep3+phlD+2wsG+dTAkc/nzztHZDKDaNPbipcrldyrFygjXxFsxYDMqikzwzO/rJpLj814zKrGqQ4
k0TAf/+Dy9Y++QVkt9jiJqqFa2HbV+n+v5L+/r2L5Z/lYl8eNLMCczOMRY1hPEFQzkoQHz2Yeccz
UX7w2Gumre/l4b6UawpOo6b2jk+pFXZdDzFoNvV8UkoJDs+w5pbMKmQakQuAZiy21852RFp5vsnt
sJYv4xlD8Xh+g9NuCc4Vxhur0uMCAwEAAaNTMFEwHQYDVR0OBBYEFNMcN492y8jrcvjPWKOusx6H
REErMB8GA1UdIwQYMBaAFNMcN492y8jrcvjPWKOusx6HREErMA8GA1UdEwEB/wQFMAMBAf8wDQYJ
KoZIhvcNAQELBQADggEBACam0y+HAF2XPINvwn8K7CbIj3cq0zTTOhzT3y19JJ3JHPWxLJgCAnsB
U1Tl/xoDIPqMetgk2p+o4uCM2c/MaP6QeLpgT2V+KirxPDssdxsh6ujz/Ld7JD83bcWK2gamq69l
9zK//IdvZWBxrz13FsZl8r8pY0aY/2rI8iCGT39icFN9SsxiFmhg/IclOq60roc5Ob9zgetl4PlZ
rwGePWm70G3bxmfJ85o9nefHvHxvWdpKgNkn8TLS0B6yRvga7AdAG1VruBqbYY7ap8T51jj/B+h6
G4u+jX1OfhA0fa4mKlUs0yTGbC9HcPOcDgMGhPQt3p3khRwZVKKg6QOdAWw=</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:KeyDescriptor use="encryption">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
   <ds:X509Certificate>MIID8TCCAtmgAwIBAgIUVQTIAc0qcU/0WfKbY+8tZVybjTowDQYJKoZIhvcNAQELBQAwgYcxCzAJ
BgNVBAYTAklOMRIwEAYDVQQIDAlUZWxhbmdhbmExEjAQBgNVBAcMCUh5ZGVyYWJhZDENMAsGA1UE
CgwEQlJDTTEMMAoGA1UECwwDU0JVMQswCQYDVQQDDAJWVjEmMCQGCSqGSIb3DQEJARYXYXRsdXJp
X3ZlbmthdEB5YWhvby5jb20wHhcNMjUwOTI0MDUxOTQ4WhcNMjgwNzE0MDUxOTQ4WjCBhzELMAkG
A1UEBhMCSU4xEjAQBgNVBAgMCVRlbGFuZ2FuYTESMBAGA1UEBwwJSHlkZXJhYmFkMQ0wCwYDVQQK
DARCUkNNMQwwCgYDVQQLDANTQlUxCzAJBgNVBAMMAlZWMSYwJAYJKoZIhvcNAQkBFhdhdGx1cmlf
dmVua2F0QHlhaG9vLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMGvz49yz3rX
9wE0DR0BeS2x1LF2SeYrxWDWcmtis5sWuk2fk7i4pfNpyZmu3W16pomWNIaUYWn8fBWvAmw7ZlsD
ep3+phlD+2wsG+dTAkc/nzztHZDKDaNPbipcrldyrFygjXxFsxYDMqikzwzO/rJpLj814zKrGqQ4
k0TAf/+Dy9Y++QVkt9jiJqqFa2HbV+n+v5L+/r2L5Z/lYl8eNLMCczOMRY1hPEFQzkoQHz2Yeccz
UX7w2Gumre/l4b6UawpOo6b2jk+pFXZdDzFoNvV8UkoJDs+w5pbMKmQakQuAZiy21852RFp5vsnt
sJYv4xlD8Xh+g9NuCc4Vxhur0uMCAwEAAaNTMFEwHQYDVR0OBBYEFNMcN492y8jrcvjPWKOusx6H
REErMB8GA1UdIwQYMBaAFNMcN492y8jrcvjPWKOusx6HREErMA8GA1UdEwEB/wQFMAMBAf8wDQYJ
KoZIhvcNAQELBQADggEBACam0y+HAF2XPINvwn8K7CbIj3cq0zTTOhzT3y19JJ3JHPWxLJgCAnsB
```

```
U1Tl/xoDIPqMetgk2p+o4uCM2c/MaP6QeLpgT2V+KirxPDssdxsh6ujz/Ld7JD83bcWK2gamq69l
9zK//IdvZWBxrz13FsZl8r8pY0aY/2rI8iCGT39icFN9SsxiFmhg/IclOq60roc5Ob9zgetl4PlZ
rwGePWm70G3bxmfJ85o9nefHvHxvWdpKgNkn8TLS0B6yRvga7AdAG1VruBqbYY7ap8T51jj/B+h6
G4u+jX1OfhA0fa4mKlUs0yTGbC9HcPOcDgMGhPQt3p3khRwZVKKg6QOdAWw=</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://
<sspHost>/default/saml/v1/idp/logout"/>
<md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="https://
<sspHost>/default/saml/v1/idp/logout"/>
<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameIDFormat>
<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</md:NameIDFormat>
<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName</md:NameIDFormat>
<md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://
<sspHost>/default/saml/v1/idp/login"/>
<md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="https://
<sspHost>/default/saml/v1/idp/login"/>
</md:IDPSSODescriptor>
<md:SPSSODescriptor AuthnRequestsSigned="true" WantAssertionsSigned="true"
 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
<md:KeyDescriptor use="signing">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
   <ds:X509Certificate>MIIDMjCCAhqgAwIBAgIQQCa3dG4iWjhTUv6+DCFExTANBgkqhkiG9w0BAQsFADAzMREwDwYDVQQK
DAhCcm9hZGNvbTEMMAoGA1UECwwDU0VEMRAwDgYDVQQDDAdkZWZhdWx0MB4XDTI1MDkyNTEyNTcx
MVoXDTI4MDYyMTEyNTcxMVowMzERMA8GA1UECgwIQnJvYWRjb20xDDAKBgNVBAsMA1NFRDEQMA4G
A1UEAwwHZGVmYXVsdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAJs21S+B7nir+jQj
mYX2orjyYegAwLh7INYRLWbOmDlvVcKg9dRrYZ7l2JgkPILJqekwHTxgNirsTy2P3lqvIrTSdA/T
dy2IQ3S6N6hzCG4a2GN6dfI7x5ruyv/tJTdE0Lla/X7+hPP0mHVnTQVf/nrxwtzvC5mCZ1LaAdck
oJ6MR3he3GG4QVbv0ZnOTYvCnAlj8pjh+KrbzAdKeFKe39Fc0DraqgHQIwzGrjV7uSR9He486Wo9
d8dCTT0uiF68/LbMseBtVZ12wPvQKIiS5ZeXvfHpg9hWrllNIyNsFg8Wf3mjY6fltarc034xKWMb
Yo7OJhBeErjSPpLaZQA+W6cCAwEAAaNCMEAwDAYDVR0TAQH/BAIwADAOBgNVHQ8BAf8EBAMCBaAw
IAYDVR0lAQH/BBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMA0GCSqGSIb3DQEBCwUAA4IBAQBsQW8q
oqkbsVbDltUx5pRI6HMoCX+hulVPG/NfMwIvhWzOofQ+jZ5avSuivsPf7V3BYk3lmh0Cka8Zrfsj
QK0KShPJyxCFZXKcADu6Zswg8sLz6rCmf8+2EVysMFhyuym7baAGvR5z3dIw3c+4zmCPcWlkCMUE
b4zCygzk2f6993n43EKW7Plu/rl3fy9mTmYTGpK/PSY6HuXgU3t1KbR/S4J1/D0047A3s9R7Xf00
cEoaG3/+hd2BARezKNz12ObWDBcfqa6w7ur20BrLj+DgALmb9m21/W7dSpjAQYyxCDklHD1Hp89O
8+5DcoaTwsVBbuq4WGws4BkVLKzmfdRn</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:KeyDescriptor use="encryption">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
   <ds:X509Certificate>MIIDMjCCAhqgAwIBAgIQQCa3dG4iWjhTUv6+DCFExTANBgkqhkiG9w0BAQsFADAzMREwDwYDVQQK
DAhCcm9hZGNvbTEMMAoGA1UECwwDU0VEMRAwDgYDVQQDDAdkZWZhdWx0MB4XDTI1MDkyNTEyNTcx
MVoXDTI4MDYyMTEyNTcxMVowMzERMA8GA1UECgwIQnJvYWRjb20xDDAKBgNVBAsMA1NFRDEQMA4G
A1UEAwwHZGVmYXVsdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAJs21S+B7nir+jQj
mYX2orjyYegAwLh7INYRLWbOmDlvVcKg9dRrYZ7l2JgkPILJqekwHTxgNirsTy2P3lqvIrTSdA/T
dy2IQ3S6N6hzCG4a2GN6dfI7x5ruyv/tJTdE0Lla/X7+hPP0mHVnTQVf/nrxwtzvC5mCZ1LaAdck
oJ6MR3he3GG4QVbv0ZnOTYvCnAlj8pjh+KrbzAdKeFKe39Fc0DraqgHQIwzGrjV7uSR9He486Wo9
d8dCTT0uiF68/LbMseBtVZ12wPvQKIiS5ZeXvfHpg9hWrllNIyNsFg8Wf3mjY6fltarc034xKWMb
```

Yo7OJhBeErjSPpLaZQA+W6cCAwEAAaNCMEAwDAYDVR0TAQH/BAIwADAOBgNVHQ8BAf8EBAMCBaAw
IAYDVR0lAQH/BBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMA0GCSqGSIb3DQEBCwUAA4IBAQBsQW8q
oqkbsVbDltUx5pRI6HMoCX+hulVPG/NfMwIvhWzOofQ+jZ5avSuivsPf7V3BYk3lmh0Cka8Zrfsj
QK0KShPJyxCFZXKcADu6Zswg8sLz6rCmf8+2EVysMFhyuym7baAGvR5z3dIw3c+4zmCPcWlkCMUE
b4zCygzk2f6993n43EKW7Plu/rl3fy9mTmYTGpK/PSY6HuXgU3t1KbR/S4J1/D0047A3s9R7Xf00
cEoaG3/+hd2BARezKNz12ObWDBcfqa6w7ur20BrLj+DgALmb9m21/W7dSpjAQYyxCDklHD1Hp89O
8+5DcoaTwsVBbuq4WGws4BkVLKzmfdRn</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:KeyDescriptor use="signing">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
<ds:X509Certificate>MIID8TCCAtmgAwIBAgIUVQTIAc0qcU/0WfKbY+8tZVybjTowDQYJKoZIhvcNAQELBQAwgYcxCzAJ
BgNVBAYTAklOMRIwEAYDVQQIDAlUZWxhbmdhbmExEjAQBgNVBAcMCUh5ZGVyYWJhZDENMAsGA1UE
CgwEQlJDTTEMMAoGA1UECwwDU0JVMQswCQYDVQQDDAJWVjEmMCQGCSqGSIb3DQEJARYXYXRsdXJp
X3ZlbmthdEB5YWhvby5jb20wHhcNMjUwOTI0MDUxOTQ4WhcNMjgwNzE0MDUxOTQ4WjCBhzELMAkG
A1UEBhMCSU4xEjAQBgNVBAgMCVRlbGFuZ2FuYTESMBAGA1UEBwwJSHlkZXJhYmFkMQ0wCwYDVQQK
DARCUkNNMQwwCgYDVQQLDANTQlUxCzAJBgNVBAMMAlZWMSYwJAYJKoZIhvcNAQkBFhdhdGx1cmlf
dmVua2F0QHlhaG9vLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMGvz49yz3rX
9wE0DR0BeS2x1LF2SeYrxWDWcmtis5sWuk2fk7i4pfNpyZmu3W16pomWNIaUYWn8fBWvAmw7ZlsD
ep3+phlD+2wsG+dTAkc/nzztHZDKDaNPbipcrldyrFygjXxFsxYDMqikzwzO/rJpLj814zKrGqQ4
k0TAf/+Dy9Y++QVkt9jiJqqFa2HbV+n+v5L+/r2L5Z/lYl8eNLMCczOMRY1hPEFQzkoQHz2Yeccz
UX7w2Gumre/l4b6UawpOo6b2jk+pFXZdDzFoNvV8UkoJDs+w5pbMKmQakQuAZiy2l852RFp5vsnt
sJYv4xlD8Xh+g9NuCc4Vxhur0uMCAwEAAaNTMFEwHQYDVR0OBBYEFNMcN492y8jrcvjPWKOusx6H
REErMB8GA1UdIwQYMBaAFNMcN492y8jrcvjPWKOusx6HREErMA8GA1UdEwEB/wQFMAMBAf8wDQYJ
KoZIhvcNAQELBQADggEBACam0y+HAF2XPINvwn8K7CbIj3cq0zTTOhzT3y19JJ3JHPWxLJgCAnsB
U1Tl/xoDIPqMetgk2p+o4uCM2c/MaP6QeLpgT2V+KirxPDssdxsh6ujz/Ld7JD83bcWK2gamq69l
9zK//IdvZWBxrz13FsZl8r8pY0aY/2rI8iCGT39icFN9SsxiFmhg/IclOq60roc5Ob9zgetl4PlZ
rwGePWm70G3bxmfJ85o9nefHvHxvWdpKgNkn8TLS0B6yRvga7AdAG1VruBqbYY7ap8T51jj/B+h6
G4u+jX1OfhA0fa4mKlUs0yTGbC9HcPOcDgMGhPQt3p3khRwZVKKg6QOdAWw=</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:KeyDescriptor use="encryption">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
<ds:X509Certificate>MIID8TCCAtmgAwIBAgIUVQTIAc0qcU/0WfKbY+8tZVybjTowDQYJKoZIhvcNAQELBQAwgYcxCzAJ
BgNVBAYTAklOMRIwEAYDVQQIDAlUZWxhbmdhbmExEjAQBgNVBAcMCUh5ZGVyYWJhZDENMAsGA1UE
CgwEQlJDTTEMMAoGA1UECwwDU0JVMQswCQYDVQQDDAJWVjEmMCQGCSqGSIb3DQEJARYXYXRsdXJp
X3ZlbmthdEB5YWhvby5jb20wHhcNMjUwOTI0MDUxOTQ4WhcNMjgwNzE0MDUxOTQ4WjCBhzELMAkG
A1UEBhMCSU4xEjAQBgNVBAgMCVRlbGFuZ2FuYTESMBAGA1UEBwwJSHlkZXJhYmFkMQ0wCwYDVQQK
DARCUkNNMQwwCgYDVQQLDANTQlUxCzAJBgNVBAMMAlZWMSYwJAYJKoZIhvcNAQkBFhdhdGx1cmlf
dmVua2F0QHlhaG9vLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMGvz49yz3rX
9wE0DR0BeS2x1LF2SeYrxWDWcmtis5sWuk2fk7i4pfNpyZmu3W16pomWNIaUYWn8fBWvAmw7ZlsD
ep3+phlD+2wsG+dTAkc/nzztHZDKDaNPbipcrldyrFygjXxFsxYDMqikzwzO/rJpLj814zKrGqQ4
k0TAf/+Dy9Y++QVkt9jiJqqFa2HbV+n+v5L+/r2L5Z/lYl8eNLMCczOMRY1hPEFQzkoQHz2Yeccz
UX7w2Gumre/l4b6UawpOo6b2jk+pFXZdDzFoNvV8UkoJDs+w5pbMKmQakQuAZiy2l852RFp5vsnt
sJYv4xlD8Xh+g9NuCc4Vxhur0uMCAwEAAaNTMFEwHQYDVR0OBBYEFNMcN492y8jrcvjPWKOusx6H
REErMB8GA1UdIwQYMBaAFNMcN492y8jrcvjPWKOusx6HREErMA8GA1UdEwEB/wQFMAMBAf8wDQYJ
KoZIhvcNAQELBQADggEBACam0y+HAF2XPINvwn8K7CbIj3cq0zTTOhzT3y19JJ3JHPWxLJgCAnsB
U1Tl/xoDIPqMetgk2p+o4uCM2c/MaP6QeLpgT2V+KirxPDssdxsh6ujz/Ld7JD83bcWK2gamq69l
9zK//IdvZWBxrz13FsZl8r8pY0aY/2rI8iCGT39icFN9SsxiFmhg/IclOq60roc5Ob9zgetl4PlZ

```
rwGePWm70G3bxmfJ85o9nefHvHxvWdpKgNkn8TLS0B6yRvga7AdAG1VruBqbYY7ap8T51jj/B+h6
G4u+jX1OfhA0fa4mKlUs0yTGbC9HcPOcDgMGhPQt3p3khRwZVKKg6QOdAWw=</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</md:NameIDFormat>
<md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://
<sspHost>/default/saml/v1/sp/acs" index="0" isDefault="true"/>
</md:SPSSODescriptor>
</md:EntityDescriptor>
```

2. **Updating the Signing Key:**
   After clients have had sufficient time to acquire the "next" certificate, the actual signing key can be updated using the PUT operation of the Signing Key API. Upon this update, the previously published "next" certificate data will be removed from SAML metadata and JWKS endpoints.
   PUT request payload:

```
{
"kid": "ssp-signingkey",
"encrKid": "EncryptionKey", //optional
"b64CertData": "MIIDqDCCApCgAwIBAgIBAjANBgkqhkiG9....",
"b64KeyData": "MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKY...."
}
```

   PUT response payload:

```
{
"b64CertData": "MIIDqDCCApCgAwIBAgIBAjANBgkqhkiG9w0BAQsFADB5MQ....",
"b64NextCertData": null,
"b64KeyData": "MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEA....",
"kid": "ssp-signingkey",
"encrKid": "EncryptionKey",
"name": "SigningKey",
"signingKeyId": "1120ac71-2856-4c35-a6f2-63a0143dd187",
"x5t": "GlEVv62d2qRPrccBqo-YtMWkgNI",
"updatedBy": {
"principalType": "CLIENT",
"principalId": "bedfc020-482c-43db-b570-817babc98645",
"principalName": "DefaultTenantClient"
},
"createdDateTime": "2025-10-13 05:07 AM UTC",
"updatedDateTime": "2025-10-13 12:31 PM UTC",
"x5t#S256": "IyjVm2eMQQkS9UILEHAFK3jxSVlNCNZe1kvKKW6a6sk"
}
```

   NOTE
   The choice of when to publish the "next" certificate and when to update the signing key rests with the operators of the service. There will be no time-based logic built into the manual update process. The "next" signing cert will be published until the signing key is updated.

Managing Certificates

Learn how to perform CRUD operations using the Certificate APIs.

Certificate Management allows an administrator to manage public keys or certificates of other parties for token validation, and ssl communication. With the Certificate APIs, the administrator can add, retrieve, update, and delete certificates as appropriate.

Use the Certificate Management APIs for the following operations:

- Adding a new certificate, and adding a certificate to an existing alias.
- Retrieving all certificates
- Retrieving a certificate by specifying the certificate Id
- Updating a certificate by specifying the certificate Id
- Deleting a certificate by specifying the certificate Id

This topic includes the following sections:

- Certificate APIs and CRUD Operations
- Examples of Certificate API CRUDs
  - Updating a Certificate referenced by other IDSP objects
- Converting DER and PEM certificate files
- Generating a sample self-signed "ssp-general-tls" certificate

Certificate APIs and CRUD Operations

The following table includes the list of Certificate APIs that can be used for the management of certificates:

Operation Description	Operation Type	Endpoint URLs
Invoke this API to add or import public certificates or public keys	POST	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Certs/
Invoke this API to retrieve all certificates or public keys	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Certs
Invoke this API to retrieve certificate or public key by specifying the certificate Id	GET	https://${SSP_FQDN}/default/admin/v1/Certs/{{certificateId}}
Invoke this API to update a certificate or public key by specifying the Id	PUT	https://${SSP_FQDN}/default/admin/v1/Certs/{{certificateId}}
Invoke this API to delete a certificate or public key by specifying the Id	DELETE	https://${SSP_FQDN}/default/admin/v1/Certs/{{certificateId}}

For information about the scopes that are supported by these endpoints, see "Scopes". The request and response payloads are samples.

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.

  For example:

  `https://{{host}}/{{tenantName}}/admin/v1/Certs`

  should look similar to:

  `https://example.com/default/admin/v1/Certs`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Examples of Certificate API CRUDs

Adding Certificate

Using the POST Operation you can either add a completely new certificate or add a new certificate to an existing alias.

```
POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/Certs/
```

API Request:

```
{
    "tokenIssuer": "https://security.com/iam/siteminder",
    "alias": "SSOCert",
    "kid": "ssocert-env1",
    "b64CertData": "MIIDcjCCAlqgAwIBA........."
}
```

Where,

- (optional) tokenIssuer - issuer of the token
- alias - The name of the certificate alias. You can import multiple certificates under the same alias. If there are multiple certificates, at least one certificate should be valid among all the imported certificates to continue the SSL communication. For more details, see **certificate rotation**.
- kid - key identifier of the certificate.
- b64CertData - Base64 encoded DER formatted Public key/Certificate data

Retrieving Certificate

```
GET https://${SSP_FQDN}/{{tenantName}}/admin/v1/Certs/
```

API Response:

```
 {
        "tokenIssuer": "",
        "certificateId": "ad0fdab9-cc4b-4718-a440-04ce031a2843",
        "b64CertData": "MIIEszCCA5ugAwIBAg...",
        "alias": "test1",
        "description": "testconfig",
        "createdBy": {
            "principalType": "USER",
            "principalId": "b3484610-2213-4015-bba6-a7ac5e36177a",
            "principalName": "malbail"
        },
        "updatedBy": {
            "principalType": "USER",
            "principalId": "b3484610-2213-4015-bba6-a7ac5e36177a",
            "principalName": "malbail"
        },
        "createdDateTime": "2024-03-05 05:43 AM UTC",
        "updatedDateTime": "2024-03-05 05:43 AM UTC",
        "issuerDN": "ST=California,L=San Francisco,OU=CloudFlare Origin SSL Certificate Authority,O=
\"CloudFlare\\, Inc.\",C=US",
        "subjectDN": "CN=CloudFlare Origin Certificate,OU=CloudFlare Origin CA,O=\"CloudFlare\\, Inc.",
        "serialNumber": "5D2F49759E692E5245102FEC366C4DA8FA0565BE",
        "fipsCompliant": "Yes",
        "validFrom": "2022-10-25 06:45 AM UTC",
        "validTo": "2037-10-21 06:45 AM UTC",
        "keyAlgorithm": "SHA256WITHRSA",
```

```
        "x5t": "pKgsQc0BCqPNbKErFBfJtQbBDFE",
        "x5t#S256": "RNefxfHWbd9QkRMsOlovTjyt58bdMAOfaIMAdFIhcFU"
    }
```

Updating Certs

```
PUT https://${SSP_FQDN}/default/admin/v1/Certs/{{certificateId}}
```

API Request:

```
{
    "tokenIssuer": "https://example.com/iam/siteminder",
    "alias": "keycert1",
    "b64CertData": "MIIDmzC.......Rd"
}
```

Where you can update the b64CertData only.

Deleting Certs

```
DELETE https://${SSP_FQDN}/default/admin/v1/Certs/{{certificateId}}
```

API Response:
204 No Content.

Updating a Certificate referenced by other IDSP objects

When updating a tenant's certificate that is referenced by other IDSP objects, the existing **PUT**/**DELETE** API returns a list of all dependent objects using the certificate. This allows you to review where the certificate is in use and decide whether to proceed with the update or deletion. If the certificate is already in use elsewhere, you have the option to retain it.

The PUT/DELETE operations is where the reference list gets populated:

NOTE

- If a certificate resource that is being used by other IDSP objects is deleted or its alias is updated, the existing update or delete API returns the list of objects that are using the resource being modified or removed.
- When attempting to update or delete an certificate that is currently referenced by other IDSP objects, the system returns error code 80000134 to prevent the deletion. See Error Codes here.
- The DELETE operation must contain the "**force=true**" query parameter to delete references of the IDSP objects.

```
PUT/DELETE https://${SSP_FQDN}/<tenantName>/admin/v1/Certs/{{certificateId}}
```

The following sample response illustrates how this list is displayed.

Sample Response:

```
{

    "errorCode": "0000118",

    "errorMessage": "Certificate alias can't be deleted as resource is in use",

    "referencedObjects": [

    {

      "id": "c042e581-5bef-4831-a52a-25ddcf4e4beb",
```

```
      "name": "SCIM1",
      "type": "urn:iam:res:scimconfig"
    },
    {
      "id": "e20bbc85-92a1-43ee-a37b-a48f8a9af047",
      "name": "SCIM",
      "type": "urn:iam:res:scimconfig"
    },
    {
      "id": "fd467d85-c1e3-4acb-ad47-596e529c1a0e",
      "name": "customRiskProvider1",
      "type": "urn:iam:res:providerconfig"
    }
    ]
}
```

Converting PEM to DER certificate files

The "/admin/v1/Certs" API requires a certificate format of DER encoded using base64 in one contiguous line without line breaks/new-line characters. A PEM certificate is a Base64 encoded DER certificate (with a -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- markers).

To convert a .pem file to base64 encoded DER, use the following command:

```
openssl x509 -in <cert file name>.pem -outform DER | openssl enc -base64 -A
```

Where, <cert file name>.pem is the name of the pem certificate file

To convert a .der file to a base64 encoded DER, use the following command:

```
openssl base64 -in <cert file name>.der -out <cert file name>.der.base64 -A
```

Where, <cert file name>.der is the name of the .der certificate file.

NOTE
When you import a certificate through the Symantec Identity Security Platform Admin Console, the required format of the certificate is a "***.der***" file.

To convert a .pem file to a DER file, use the following command:

```
openssl x509 -in <cert file name>.pem -outform DER -out <cert file name>.der
```

Where, <cert file name>.pem is the name of the pem certificate file and the <cert file name>.der is the name of the .der file to be created.

Generating a sample self-signed "ssp-general-tls" certificate

To generate a **sample** self-signed "ssp-general-tls" certificate for the ${SSP_FQDN}, use the following command:

```
openssl req -new -x509 -nodes -config $CERTFILE.cnf -extensions v3_ca -keyout $KEYFILE -out $CERTFILE -days
 3650
```

Where,

- KEYFILE=ssp-general-tls.key
- CERTFILE=ssp-general-tls.crt

See the following sample "ssp-general-tls.crt.cnf" openssl configuration file for a sample "**example.com**" domain.

> **NOTE**
> When creating a similar file for your environment, ensure to update the "ssp-general-tls.crt.cnf" file with appropriate details and replace **example.com** with your ${SSP_FQDN} domain.

Reference Links to:

- For sample request and response payloads of all the URLs mentioned in the above table, refer to Certificate API Swagger documentation
- Certificate API Error Codes

Certificate Rotation

Certificate Rotation allows seamless replacement of expiring certificates by adding new ones under an existing alias without service disruption.

Certificate rotation is handled by adding a new certificate under the same alias. This allows seamless replacement without downtime, as multiple certificates can coexist under the same alias.

Key Points:

- Multiple certificates can be associated with a single alias.
- At least one certificate must remain valid under a given alias to ensure uninterrupted SSL communication.
- When rotating a certificate:
  - Add the new certificate using a unique kid.
  - Ensure the existing valid certificate is still active until the new certificate is fully propagated and trusted by all consumers.
  - Once the new certificate is confirmed as active and in use, the old certificate can be removed if desired.

Example of Certificate Rotation

Assuming that you already have the following certificate added in Symantec Identity Security Platform:

```
{
  "tokenIssuer": "https://security.com/iam/siteminder",
  "alias": "SSOCert",
  "kid": "ssocert-env1",
  "b64CertData": "MIIDcjCCAlqgAwIBA........."
}
```

Where,

- (optional) tokenIssuer - The issuer of the token
- alias - The name of the certificate alias. You can import multiple certificates under the same alias. If there are multiple certificates, at least one certificate should be valid among all the imported certificates to continue the SSL communication.
- kid - The key identifier of the certificate.
- b64CertData - Base64 encoded DER formatted Public key or Certificate data.

To rotate a certificate under the same alias "SSOCert" that is specified in the previous example, you would add the new certificate using a distinct kid, as shown here:

```
{
"alias": "SSOCert",
"kid": "ssocert-env2",
"b64CertData": "NEW_CERTIFICATE_DATA_HERE"
}
```

This process results in smooth transition of certificates during the renewal cycles.

Reference Links

- Managing Certificates (API Flow)
- Certificates (Admin Console Flow)

Managing Secrets

Perform CRUD operations using the Secrets and Keys APIs to manage sensitive credentials independently from service configurations.

Decoupling the lifecycle management (LCM) of Identity Store configurations from the management of their associated secrets is essential for operational security. Currently, LDAP and SCIM configurations store connection secrets during creation or updates but do not return them during GET requests. Consequently, only clients with knowledge of the original secret can modify Identity Store objects—such as remapping attributes—which creates significant LCM challenges for standard operations.

In many organizational structures, the responsibility for managing external service integrations is separate from the privilege of accessing connection credentials. To accommodate this separation, a Secrets LCM pattern is utilized via the /Secrets endpoint. This supports requirements for /SCIMconfigs (client secrets and private keys) and /LDAPconfigs (bind passwords).

A Secret represents a sensitive object required for normal system operations. All Secrets are stored in an encrypted format and are only decrypted at the moment of execution.

Each Secret can be assigned a type to categorize the stored object. While the Secret type does not affect runtime execution, it helps associate the data with a well-known category. If the following specific types are not applicable, use custom or secret:

- cert
- custom
- eckey
- rsakey
- secret

This topic includes the following sections:

- List of Secret APIs
- Samples of POST and PUT "/Secrets" API
- Samples of POST, and PUT "/Keys" API
- Updating a Secret referenced by other IDSP objects

List of Secret APIs

The following table displays the list of Secret APIs that can be used to manage secrets of the ID store to manage ongoing changes such as adding or remapping attributes, etc. in the ID store.

The "/secrets" API is an admin API that provides admin-level access to secrets using the **urn:iam:t.secrets** scope.

Operation Description	Operation Type	Endpoint URLs
Add a secret	POST	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Secrets
Retrieve all secrets	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Secrets
Retrieve a secret by specifying the secret Id	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Secrets/{{secretId}}
Update a secret by specifying the Id	PUT	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Secrets/{{secretId}}
Delete a secret by specifying the Id	DELETE	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Secrets/{{secretId}}
Add a key to the secret	POST	https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/Keys
Update a key by specifying the Id	PUT	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Keys/{secretId}
Retrieve a key by specifying the secret Id	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Keys/{secretId}
Retrieve a complete secret value by specifying the secret Id. This operation requires Tenant admin privileges.	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/SecretsGetter/{secretId}

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/Secrets`
  should look similar to:
  `https://example.com/default/admin/v1/Secrets`
- The Delete operation deletes the message template Id for languages other than English (if other languages exist).
- During the PUT operation, only field types, "message_template" and "updated_datetime" can be updated, all other fields cannot be updated.
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Samples of POST and PUT "/Secrets" API

POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/Secrets

```
{
"alias": "samplealias",
"type":"secret",
"description":"Sample secret",
"b64Data": "cGFzc3dvcmQ="
}
```

Where,

- alias - This is the name of the secret.
- type - This is the type of secret. For example, secret, rsakey
- description - This is the description of the secret.
- b64Data - (Optional) This is the secret, password, or certificate that you want to save in this secret. The b64Data should be a Base64 encoded string.

PUT https://${SSP_FQDN}/{{tenantName}}/admin/v1/Secrets

```
{
"alias": "samplealias",
"type":"secret",
"description":"Sample secret",
"b64Data": "cGFzc3dvcmQ="
}
```

Samples of POST, and PUT "/Keys" API

POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/Keys

```
{
"alias": "sampleKeyalias",
"description":"Sample secret",
"data" : {"b64Key" : "cmVxdWlyZWQtYmFzZTY0S2V5RGF0YQ==", "b64Crt" : "cmVxdWlyZWQtYmFzZTY0Q2VydERhdGE=",
 "kid" : "kid-value"}
}
```

PUT https://${SSP_FQDN}/{{tenantName}}/admin/v1/Keys

```
{
"alias": "sampleUpdatedKeyalias",
"description":"Sample updated secret",
"data" : {"b64Key" : "cmVxdWlyZWQtYmFzZTY0S2V5RGF0YQ==", "b64Crt" : "cmVxdWlyZWQtYmFzZTY0Q2VydERhdGE=",
 "kid" : "kid-value"}
}
```

NOTE
b64key and b64Crt elements of the json structure accept Base64 encoded key and cert data, respectively. To convert from standard RSA private/public formats into Base64 format as required by /key payload, using the openssl command line tool, run the following commands:

- openssl rsa -in <key file name>.key -outform DER | openssl enc -base64 -A
- openssl x509 -in <cert file name>.pem -outform DER | openssl enc -base64 -A

For information about X509 encodings, see https://www.ssl.com/guide/pem-der-crt-and-cer-x-509-encodings-and-conversions/

Updating or Deleting a Secret or its Alias Referenced by other IDSP objects

When updating a Secret or Key Alias (Only applicable in API flow):

If a tenant's secret or key alias is being updated and it is currently referenced by other IDSP objects, the PUT API performs a dependency check. If a conflict is detected, and other objects are using the same secret or key alias, the API prevents the update and returns error code 80000134, along with a list of all objects that reference the alias.

This allows the administrator to review where the alias is being used and manually remove the reference from each object before proceeding with the update. For example, the alias might be used in configurations such as LDAP, SCIM, or Provider settings. **The update is only allowed once all such references are removed.**

When Deleting a Secret or Key:

If a tenant's secret or key is being deleted and it is referenced by other IDSP objects, the DELETE API performs a dependency check. If a conflict is detected, and other objects are using the same secret or key, the API prevents the deletion and returns error code 80000134, along with a list of all objects that reference the secret or key.

This allows the administrator to review where the secret or key is being used and manually remove the reference from each object before proceeding with the deletion. For example, the secret or key might be used in configurations such as LDAP, SCIM, or Provider settings. **Delete is only allowed once all such references are removed.**

NOTE

- The HTTP response returned in case of a conflict (resource being reused by other IDSP objects) during update or deletion of a secret/key alias is **409**.
- When attempting to update or delete a Secret that is currently referenced by other IDSP objects, the system returns error code 80000134 to prevent the deletion. See Error Codes here.
- The DELETE operation must contain the "**force=true**" query parameter to delete references of the IDSP objects.
- A Helper API is available to retrieve the list of dependent objects in case of a conflict, see ReferencingObjectsHelper API.

The following PUT operation for secret where the reference list gets populated in case of conflict (resource being reused by other IDSP objects):

```
PUT https://${SSP_FQDN}/{{tenantName}}/admin/v1/Secrets/{{secretId}}
```

The following PUT operation for key where the reference list gets populated in case of conflict (resource being reused by other IDSP objects):

```
PUT https://${SSP_FQDN}/{{tenantName}}/admin/v1/Keys/{{keyId}}
```

The following DELETE operation for secret/key where the reference list gets populated in case of conflict (resource being reused by other IDSP objects):

```
DELETE https://${SSP_FQDN}/{{tenantName}}/admin/v1/Secrets/{{secretId}}
```

The following sample response illustrates how this list is displayed.

Sample Response:

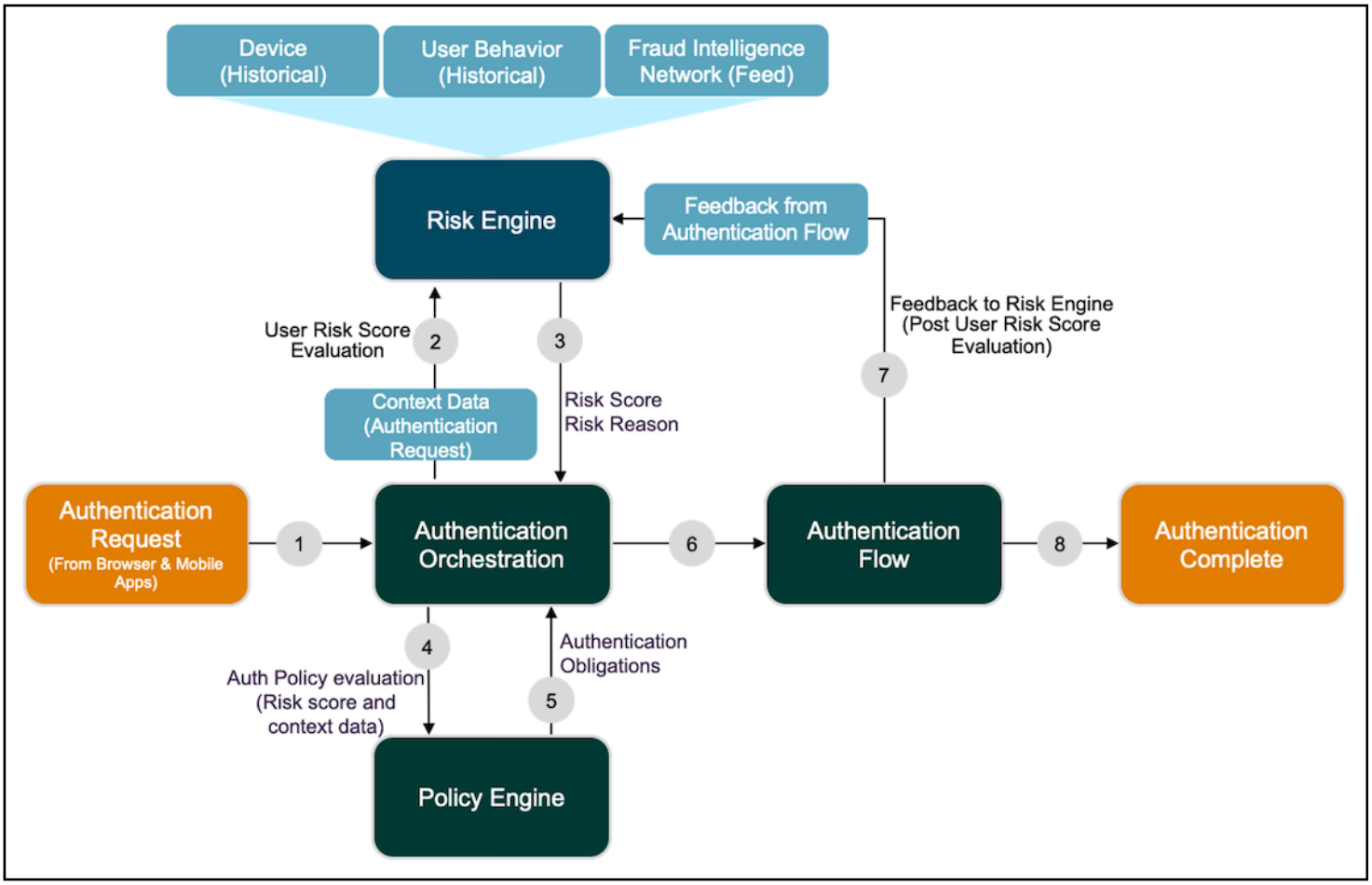
```
{

    "errorCode": "80000134",

    "errorMessage": "Secret can't be deleted as resource is in use",

    "referencedObjects": [

    {

      "id": "e20bbc85-92a1-43ee-a37b-a48f8a9af047",

      "name": "LDAP",

      "type": "urn:iam:res:ldapconfig"

    },

    {
```

```
        "id": "fd467d85-c1e3-4acb-ad47-596e529c1a0e",
        "name": "customRiskProvider1",
        "type": "urn:iam:res:providerconfig"
      }
      ]
  }
```


Symantec Identity Security Platform Risk Engine

The Symantec Identity Security Platform Risk Engine effectively analyzes and detects risks in authentication activities and prevents fraud with little or no impact on the user experience. The risk engine determines the risks by combining context data from the authentication request with device profiling, user behavior, and fraud intelligence network data.

The risk engine uses many contextual and historical parameters to assess the risk of an authentication activity in real time. It assigns a risk score from 0 to 100 to each authentication request, with 100 indicating the highest risk. It also plays an important role in the Zero Trust security architecture, as it constantly establishes and re-establishes the trust of the user.



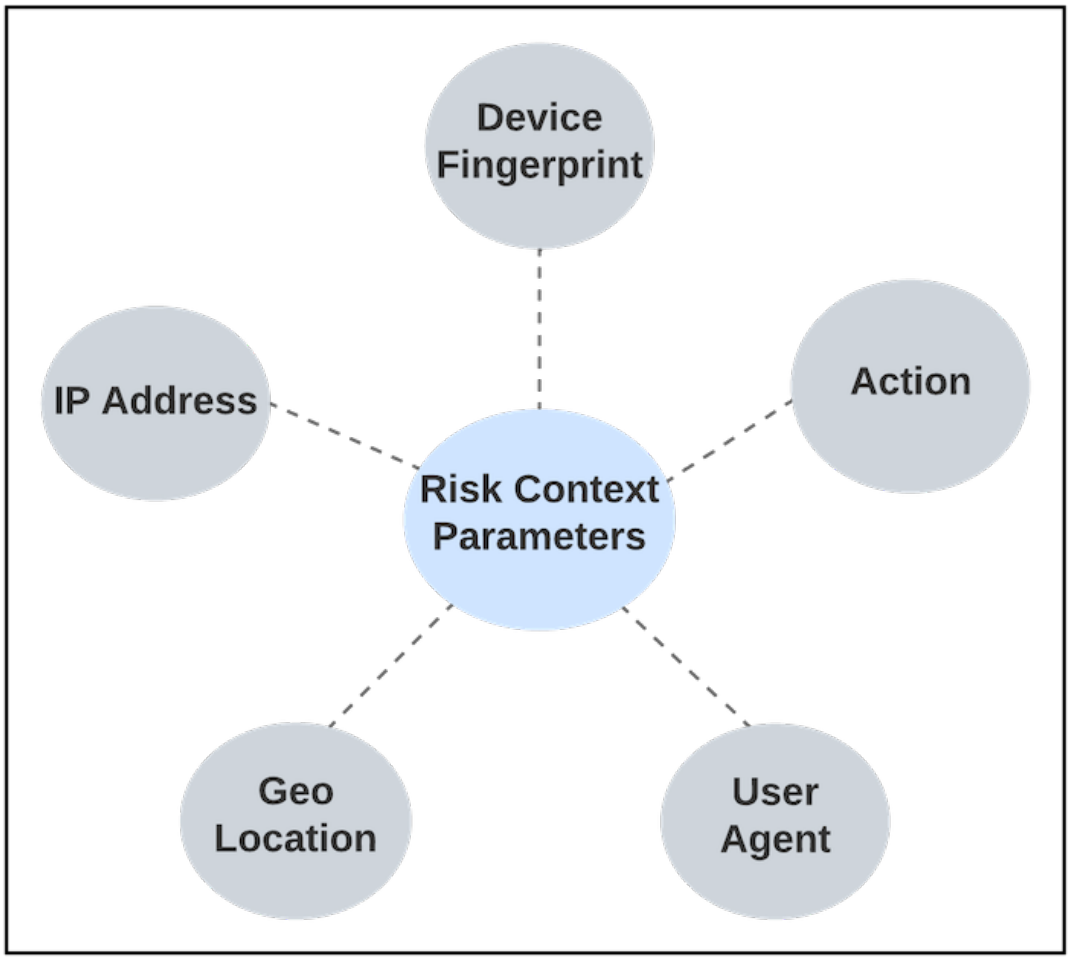
The context from the current request, user device profile, user behavior profile, and fraud intelligence network data all contribute to the risk score. The score of the risk engine, together with other authentication context data, is used by the IDSP policy engine to determine the authentication obligations. These obligations must be fulfilled by the user to complete the authentication activity. At the end of the authentication flow, feedback is delivered to the risk engine which indicates whether the risk that is recognized by the risk engine is dangerous.

Context Data

The risk engine utilizes context data from the authentication request along with device profile, user behavior, and fraud intelligence network data to identify the risk of the authentication activities.

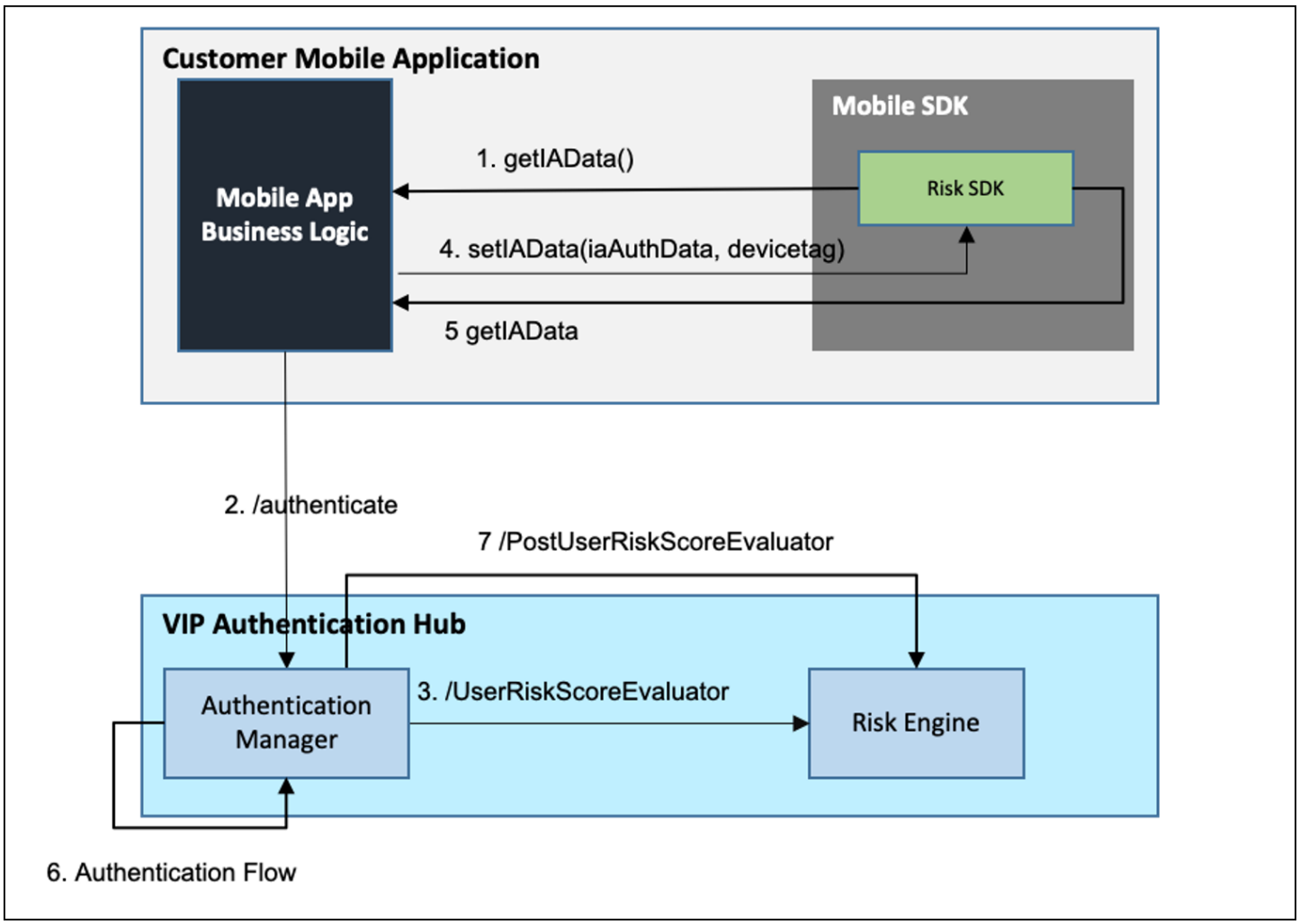
Context data, from the current authentication request, which is used by the risk engine includes:

- User Agent
  - Browser Type, Version, OS Version
- Device Fingerprint
  - Browser or Device Characteristics- Device Type, Screen resolution, locale, time zone, etc.
- Geo Location (for mobiles)
  - Latitude and Longitude of the device location
- IP Address
- Action
  - SIGININ for authentication



Collection of Device Information

Some of the context data (User Agent, Device Fingerprint) used by the risk engine is device specific and this data is collected using device-specific risk engine SDKs – JavaScript SDK for browsers and Android/iOS SDK for mobile devices. The following diagram demonstrates how device-specific data is collected using the SDK in a mobile.



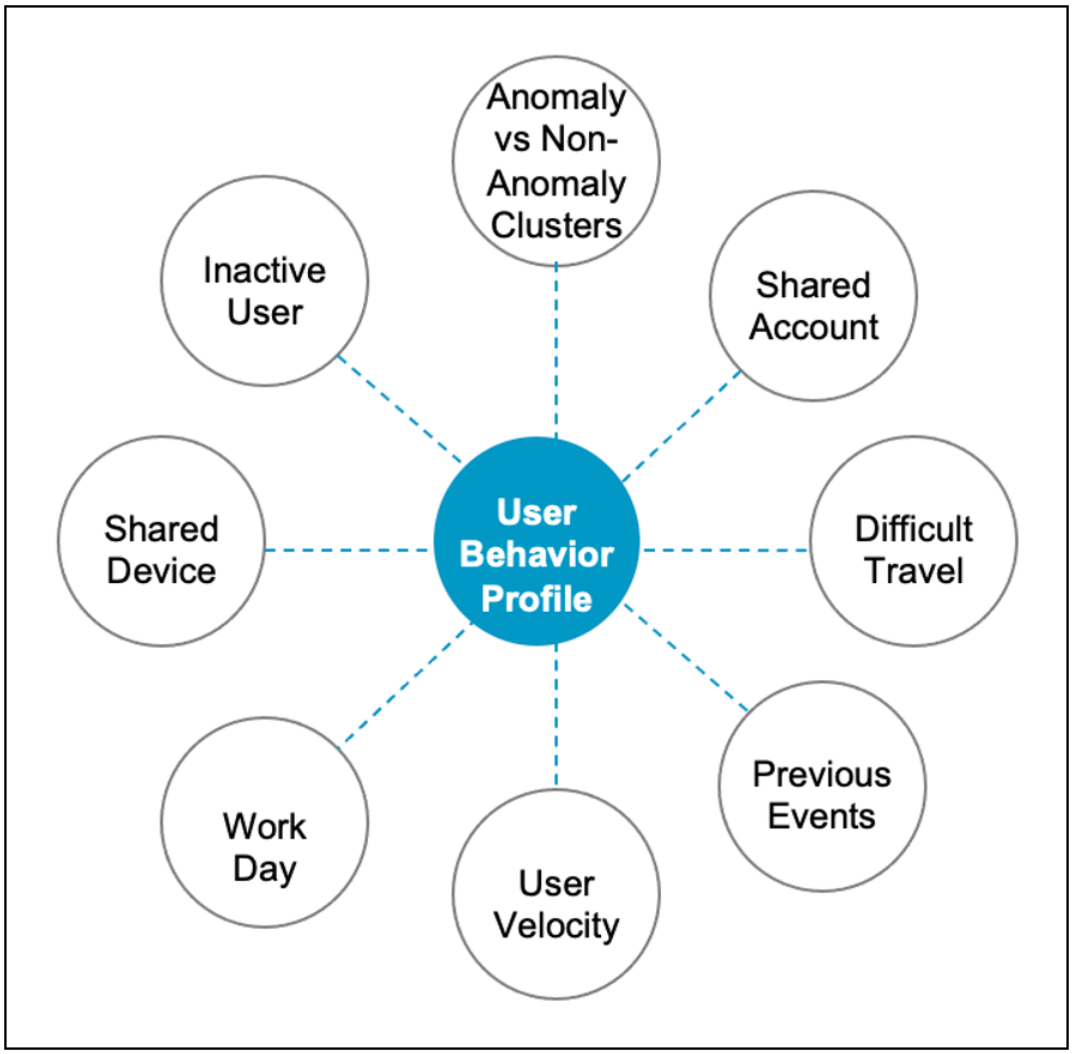
Based on the data collected by the SDKs, the risk engine recognizes the channel, web, or mobile, from which the activity is initiated. Risk evaluation from the mobile channel executes mobile-specific rules using mobile-specific device attributes. Similarly, for the web channel, web-optimized rules are executed using browser attributes.

Device Profile

The risk engine keeps track of all the devices from where the user has initiated the authentication request to a website or mobile application. Based on the device, signature each device is profiled and assigned a unique device tag. The device tag is stored by the device sent to the risk engine along with the device signature. The device profile is used to determine whether the current device is one of the devices from which the user typically requests access or whether the device is a new one. Parameters analyzed include operating system version, browser type, screen resolution, previously issued tags, and, other device settings. Device profiling is done differently for browser and mobile applications based on specific context information. If the device signature in the authentication request does not match with any of the user's registered devices, then the risk engine treats it as a new device and gives a high-risk score for the current authentication request.

User Behavior Profile

User behavior profiling is the history of typical activities for the user. The risk engine compares the current activity with the usual user behavior to determine the risk. The user profile determines if the various activities are typical for that user or if the behavior is indicative of known fraudulent patterns. Parameters examined include device, location, activity type, time of day and type of activity. User behavior is profiled based on the following categories:



- Anomaly vs non-anomaly clusters: The risk engine creates clusters based on location and device and a cluster is marked anomaly or not based on the history of activities from that cluster. The risk engine determines if the current activity falls into one of the clusters and increases the risk score if the current event is from a known anomaly cluster.
- Shared Device: The risk engine determines if the device is shared by multiple users and tracks the behavior accordingly
- Shared Account: The risk engine determines if the account is shared by multiple users and tracks the behavior accordingly
- Difficult Travel: The risk engine determines if an authentication attempt is made by a user account from two distant locations within a questionable time frame and raises the risk if the account is not a shared account
- Previous Events: The risk engine tracks the previous events of the user and increases the risk of the current activity if the last event is confirmed risky
- User Velocity: The risk engine tracks the number of activities that are performed by the user in a short period and increases the risk if the count exceeds a defined threshold
- Work Day: The risk engine tracks if the user is accessing the system outside of workday hours
- Inactive User Velocity: The risk engine tracks if the user is inactive before the current transaction is made

Fraud Intelligence Network (FIN)

Fraud intelligence network (FIN) data is loaded into the risk engine from multiple sources – Network location and anonymizer data from Neustar, Unallocated network data from Bogons. If a request is initiated from the network that is present in the FIN data, the risk engine increases the risk score for the request. Along with the FIN data, the risk engine allows enterprises to configure customs that are accepted or blocked IP addresses and countries. If the request is initiated from an accepted IP address, then the risk engine lowers the risk score. If there are requests from blocked IP addresses or countries, the risk engine increases the risk score and marks the request as risky.

Feedback from Authentication Flow

The risk engine uses context data, historical profile data, and FIN data to calculate the risk score of the current authentication request. If the risk score is greater than a configured threshold, then the current authentication request is marked risky. Based on the risk and other context data, the policy engine determines the authentication obligations that are performed by the user to complete the authentication. If the current authentication request is marked risky, then at the end of the authentication flow the status of the authentication is passed to the risk engine as feedback. If the user fails to complete authentication and the risk engine is notified, the account and the associated profiles (device and user behavior) are flagged as having a failed authentication. Any future authentication requests from the same user with the same device and user behavior will have higher risk scores. Alternatively, if the user completes the authentication successfully, the user and the associated profiles are flagged for a successful authentication. Any future authentication requests from the same user with the same device and user behavior will have lower risk scores.

Configurations supported by the Risk Engine

The IDSP risk engine supports the following configurations through REST APIs:

Configuration	Details
Configure Risk Rules	Configuration of risk rules - enable/disable, rule score, etc.
Enterprise custom-allowed IP list	List of IP Addresses and IP ranges that are allowed or whitelisted by an enterprise
Enterprise custom blocked IP list	List of IP Addresses and IP ranges that are blocked or blacklisted by an enterprise
Enterprise custom risky countries	List of countries that are marked risky by an enterprise
User velocity threshold	The number of times a user can authenticate in a given time frame.

Difficult travel threshold	Velocity configuration (300 mph) allowed by difficult travel rule

Risk Evaluation

The IDSP risk engine supports the following risk evaluation operations as REST APIs:

Operation	Details
User Risk Score Evaluation	Evaluate the risk of authentication using the context data, historical user profile (behavior and device), and FIN data.
Post User Risk Score Evaluation	Feedback to the risk engine at the end of the authentication flow to update the status of the authentication

Reference Links to:

- Managing Risks
- Uploading Network Risk Data into IDSP Database
- Integrating with External Risk Provider

Managing Risk

Contains information on the risk rules, evaluating the user's risk profiles, and custom risk provider details.

Symantec Identity Security Platform Risk service is a powerful risk-based, adaptive authentication solution that works in real time to evaluate the context and calculate a risk score. The risk server calculates the risk score for any given access attempt based on a predefined set of rules. The calculated risk score is fed into the policies to decide whether to authorize the current activity, request step-up authentication and/or send an alert or block the activity. Users are then presented with authentication options appropriate to that risk level. This approach provides the organization with a transparent layer of protection against identity theft, data breaches, and fraud.

The Risk Service provides adaptive or risk-based authentication to evaluate a set of contextual factors that are related to an access attempt or transaction to better estimate the risk involved.

Global/Tenant Setting for Risk Evaluation

The following global/tenant setting is configured by default in IDSP to evaluate risk in user profiles:

Table 111: Tenant Settings

Global/Tenant Setting	Description
isRiskServiceEnabled	This is a global/tenant setting used to control whether risk evaluation should be enabled or disabled during user authentication. It is enabled by default. It can be disabled at the tenant level individually. However, if it is globally disabled, tenants cannot override this setting to enable it. **Default Value**: "true" **Possible values**: "true" or "false"
To disable this setting use the PATCH `https://${SSP_FQDN}/{{tenantName}}/admin/v1/Configs` API to update the value. This information is also covered in the Managing Settings page.	

Reference Links

- Understanding Risk Rules
- Evaluating the Risk Score
- Risk Service APIs

Other related links

- Custom Risk Provider
- IDSP Risk Engine
- Integrating with External Risk Provider
- Risk API Error Codes
- Using Advanced Authentication Risk Service

Understanding Risk Rules

This topic describes the different types of risk rules that are supported in Symantec Identity Security Platform.

The Risk scores that are evaluated by the risk service are based on a number of risk rules that are related to user authentication.

The risk rules that are supported by the solution are shown here:

- Device Recognition
  - Device Change
- Fraud IP
- Blocked IP
- Allowed IP
- Risky Country
- Recognized Computer (RC result)
- Difficult Travel
- Behavioral Engine
- Previous Event Confirmed Risky
- User Velocity
- Work Day
- Inactive User
- Country Change
- IP Address Change

For sample request and response payloads of all risk APIs, refer to IDSP Risk API Swagger documentation

Risk Rules have been explained in detail in the following section.

Device Recognition

When a user logs in using a device for the first time, the device recognition rule treats the user transaction as risky as the device is not registered, and creates a device tag for it. However, if the user logs in for a subsequent time using the same device, then this device signature has a device tag associated with this user from the user's initial transaction. This device is then treated as a registered device, and the risk score that is calculated is much lesser.

Use this API to configure the "Device Recognition" risk rule:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/iarisk/v1/RiskRules`
  should look similar to:
  `https://example.com/default/iarisk/v1/RiskRules`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

NOTE
To know about the request and response payloads of "..."/iarisk/v1/RiskRules" API, see Setting Risk Rules.

Sample request payload:

```
{
        "name": "Device recognition",
        "enabled": true,
        "riskScore": 100.0,
        "settings": [
            {
                "key": "registeredPct",
                "value": "-25"
            },
            {
                "key": "unrecognizedPct",
                "value": "100"
            },
            {
                "key": "registeredmobile",
                "value": "-25"
            },
            {
                "key": "deviceFPExpiryInactivityTime",
                "value": "360"
            },
            {
                "key": "deviceFPExpiryMaxBindingLife",
                "value": "90"
            },
            {
                "key": "maxNumberOfDevicesPerUser",
                "value": "2"
            },
            {
                "key": "simThreshold",
                "value": "2"
            },
            {
                "key": "deviceIdPct",
                "value": "-25"
```

```
            },
            {
                "key": "lastNRiskEvaluations",
                "value": "-1"
            },
            {
                "key": "deviceLimitAction",
                "value": "deleteOldest"
            },
            {
                "key": "tagExpireInSeconds",
                "value": "1800"
            }
        ]
    }
```

The following table describes the "***Device Recognition"*** settings that can be updated using the PUT "... /iarisk/v1/RiskRules" API:

Settings Name	Description
• unrecognizedPct (API terminology) • Unrecognized percentage (UI terminology)	Indicates the percentage of risk score that is assign rule if the device is not recognized. **For example**: If the risk score is 50 and "unrecognizedPct" is 50% risk score for device recognition would be 25, which total risk score.
• deviceIdPct (API terminology) • Device Id percentage (UI terminology)	Indicates the percentage of risk score that is assign rule if the deviceId is valid and active.
• deviceLimitAction • Device Limit Action (UI terminology)	This setting defines the action to be taken when a u to register a new device, but they have already reac Number Of Devices Per User" limit. The available o are: • rejectNewDeviceRegistration: If selected, any at to register a new device beyond the set limit are • deleteOldest: If selected, when a user tries to re device and has reached the limit, the system au removes the oldest registered device to allow th be registered.
• maxNumberOfDevicesPerUser (API terminology) • Max Number Of Devices Per User (UI terminology)	• This refers to the maximum number of devices t is allowed to have registered with their account. indicates that there is no explicit limit enforced, a to register an unlimited number of devices. If a p were set (for example, 5), a user would only be up to that many devices.
• deviceFPExpiryInactivityTime (API terminology) • Device fingerprint expiry inactivity time (UI terminology)	Indicates the max number of days after which an ina (FP) is deleted. This is tied to the last access time. risk reason as - "DEVICE_EXPIRED_NO_AUTH"
• registeredPct (API terminology) • Registered percentage (UI terminology)	Indicates the percentage of risk score that is assign rule if the browser is registered.
• simThreshold (API terminology) • Similar threshold (UI terminology)	Indicates the similarity threshold.

• registeredmobile (API terminology) • Registered mobile (UI terminology)	Indicates the risk score that is assigned to this risk r device is registered.
• tagExpireInSeconds (API terminology) • Tag expire in seconds (UI terminology)	Indicates the expiry of a new tag issued during the r
• deviceFPExpiryMaxBindingLife (API terminology) • Device fingerprint expiry max binding life (UI terminology)	Indicates the max allowed days for user to device Ta is tied to the last created time. You can get the risk r "DEVICE_EXPIRED_MAX_LIFE"
• lastNRiskEvaluation (API terminology) • Last N risk evaluations (UI terminology)	Specifies the number of past authentications used f the current device from which the authentication is a found in these past **N** records, the rule is triggered. To know when to enable "Last N risk evaluations" se ***Device Change rule descirption in the next secti***

The default risk score for this risk rule is ***50***.

Device Change

The **Device Change** rule enhances the Device Recognition rule by introducing additional risk evaluation criteria. To enable this rule alongside **Device Recognition**, set **lastNRiskEvaluation** to **1 or higher**.

The assigned risk score is determined based on the configured threshold. This rule always triggers when a device is unknown or new. If the device is known, the rule is triggered when the latest authentication attempts originates from a device not found in the last N authentications (explained in the previous table). The assigned risk score is based on the configured threshold.

Example

The following table illustrates how the **"Device Change"** rule behaves for a system with `lastNRiskEvaluations = 3` that is, it checks the last **3** risk evaluations for the authentication attempts):

Table 112: Device Change Risk Rule Example Scenarios

Trans action	Device	Rule Triggered?	Reason
1	Device1	No	Not enough previous transactions (rule requires 3 for evaluation).
2	Device2	No	Still not enough data to evaluate the rule.
3	Device3	No	Now we have 3 transactions, but still not enough data to evaluate the rule.
4	Device4	Yes	The device `(Device4)` is **not present** in the last 3 transactions (`Device1` , `Device2` , `Device3`), so the rule is triggered.
5	Device3	No	The device (`Device3`) **exists** in the last 3 transactions (`Device2` , `Device3` , `Device4`), so the rule is not triggered.
6	Device3	No	The device (`Device3`) is still in the last 3 transactions (`Device3` , `Device4` , `Device3`), so the rule is not triggered.
7	Device2	Yes	The device (`Device2`) is **not present** in the last 3 transactions (`Device4` , `Device3` , `Device3`), so the rule is triggered.

Fraud IP

The rule compares the IP of current event with the fraud IP static list loaded using a network data loader job. If the event IP matches the IP ranges in the list, and that fraud ip range has not expired, it increments the final risk score based on the risk level set by that fraud ip range. This risk rule works only when the Network Data chart is deployed. To know more about network data chart, see Uploading Network Data Into IDSP Risk Database.

Use this API to update the "Fraud IP" risk rule:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

Sample request payload:

```
{
"name": "Fraud IP",
"enabled": true,
"riskScore": 100.0,
"settings": null
}
```

The default risk score for this risk rule is ***100***.

Blocked IP

The rule checks whether the user's IP address exists in the enterprise custom blocked IP list. If it exists, then the current event is marked risky.

Use this API to update the "Blocked IP" risk rule:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

Sample request payload:

```
{
"name": "Blocked IP",
"enabled": true,
"riskScore": 100.0,
"settings": null
}
```

The default risk score for this risk rule is ***100***.

NOTE

- The Blocked IP rule can be updated but the risk score will always reflect the value as "100".
- The Blocked IP and Blocked IP ranges can be configured using the ".../iarisk/v1/IPList/BLOCKED administration API. See Setting Blocked IP List.

Allowed IP

This rule uses the IP address of the user’s computer as one of the input parameters to assess the risk of each transaction. The rule checks whether the user's IP address exists in the enterprise custom accepted IP list. If it exists, then the risk score is set low even if the current location does not match any of the known clusters that are associated with the entity or the user. Allowed IP is configured using the IPList administration API.

Use this API to update the "Allowed IP" risk rule:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

Sample request payload:

```
{
"name": "Allowed IP",
"enabled": true,
"riskScore": 0.0,
"settings": null
}
```

The default risk score for this risk rule is ***0.0***.

NOTE

- The Allowed IP rule can be updated but the risk score will always reflect the value as "0".
- The Allowed IP and Allowed IP ranges can be configured using the ".../iarisk/v1/IPList/ALLOWED administration API. See Setting Allowed IP List.

Risky Country

This rule comprises a list of countries from which fraudulent or malicious transactions generally originate. Enterprises maintain this list in line with the country regulations. IDSP derives the country information based on the GPS coordinates captured by the SDK or the IP address, and uses this data to score the potential for online fraud transactions from such countries. It evaluates the incoming transactions and checks if these transactions originated from an IP address that belongs to a country marked as negative. Such transactions are marked risky. Risky countries are configured using the RiskyCountries administration API.

This risk rule works only when the Network Data chart is deployed. To know more about network data chart, see Uploading Network Data Into IDSP Risk Database.

NOTE
The country code that is used in the API should be compliant with ISO2 two letter country codes. (For example: "ie" is the two-letter country abbreviation for Ireland, "zw" is the two-letter country abbreviation for Zimbabwe).

Use this API to update the "Risky Country" risk rule:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

Sample request Payload:

```
{
"name": "Risky Country",
"enabled": true,
"riskScore": 100.0,
"settings": null
}
```

The default risk score for this risk rule is ***100***.

NOTE
The Risky Countries list can be configured using the ".../iarisk/v1/IPList/RiskyCountries administration API. See Setting Risky Country List.

Recognized Computer

This rule checks if the event is initiated from a recognized computer.

Use this API to update the "Recognized Computer" risk rule:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

Sample request payload:

```
{
"name": "RC result",
"enabled": true,
"riskScore": 100.0,
"settings": [
{
"key": "recognizedScore",
"value": "-25"
}
]
```

```
}
```

The following table describes the "***Recognized Computer"*** settings that can be updated using the PUT "... /iarisk/v1/RiskRules" API:

Settings Name	Description	Default Value
recognizedScore (API Terminology) Recognized Score (UI Terminology)	the risk score that is assigned to this risk rule if the request is from a recognized computer.	-25

The default risk score for this risk rule is 100.

Difficult Travel

The Difficult Travel rule detects when a user attempts to log in from geographically distant locations within an unreasonably short time period. This rule analyzes the IP address of the current authentication event and compares it with the IP address from the previous event for the same user entity. The rule calculates the geographical distance between these two IP locations and measures the time duration between the two events. It then determines the implied "travel speed" and compares this against a configurable threshold. If the calculated travel speed exceeds the difficult travel speed limit, the rule increments the risk score by the specified amount.

> **NOTE**
>
> The rule result is skipped if both the previous event and current event belong to non-anomaly clusters. This design reduces false positives in scenarios where an account is legitimately shared by multiple users from different geographical locations. This risk rule requires the Network Data chart to be deployed.
>
> For more information about network data charts, see Uploading Network Data Into IDSP Risk Database.

Use this API to update the "Difficult Travel" risk rule:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

Sample request payload:

```
{
  "name": "Enhanced Difficult Travel",
  "enabled": true,
  "riskScore": 50.0,
  "settings": [
    {
      "key": "maxDistanceTolerance",
      "value": "50"
    },
    {
      "key": "maxSpeedUserCanTravel",
      "value": "500"
    },
    {
      "key": "maxSharedSessionsPerUser",
      "value": "1"
    }
  ]
}
```

The following table describes the "***Difficult Travel"*** settings that can be configured using the PUT "... /iarisk/v1/RiskRules" API:

Settings Name	Description	Default Value
• maxSpeedUserCanTravel (API terminology) • Maximum speed user can travel (UI terminology)	Indicates the maximum reasonable travel speed tolerance allowed during distance calculation. Value is measured in miles per hour.	500
The following two settings are currently inactive and do not affect rule behavior. They are reserved for future functionality.		
• maxDistanceTolerance (API terminology) • Maximum Distance Tolerance (UI terminology)	Indicates the maximum threshold distance (in miles) a user can travel beyond the set "maxSpeedUserCanTravel" limit before triggering the risk rule.	50
• maxSharedSessionsPerUser (API terminology) • Maximum shared session per user (UI terminology)	Indicates the maximum number of concurrent sessions a user can maintain from different geographical locations.	1

The default risk score for this risk rule is ***50***.

Behavioral Engine

The risk engine keeps track of all the locations from where a user performs various events and those locations are grouped as clusters. A cluster is identified by its current location, OS, and browser. Based on the event status, a cluster is marked as either an anomaly or non-anomaly. The rule checks if the current event is from a known cluster and determines the risk score.

If the current event does not match with any of the existing clusters, then the risk engine checks if the current event is closest to any non-anomaly clusters for the last 10 events. A comparison score is derived between the current event and clusters based on IP, OS, and browser. If the comparison score is greater than (>) the threshold, then the current event is marked as belonging to that cluster.

Use this API to update the "Behavioral Engine" risk rule:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

Sample request payload:

```
{
"name": "Behavior anomaly",
"enabled": true,
"riskScore": 50,
"settings": [
{
"key": "enableStaticScore",
"value": "false"
}
]
}
```

The default risk score for this risk rule is ***50***.

The following table describes the "***Behavior Anomaly"*** settings that can be updated invoking the ".../RiskRules" API:

Settings Name	Description	Default Value
Enable Static Score	This toggle controls whether a static score is used for behavioral anomalies.	Disabled

Previous Event Confirmed Risky

The rule checks if the previous event of this entity is marked as suspicious. If the previous event is marked suspicious, then the current event is marked risky and the risk score is set to maximum.

Use this API to update the "Previous event is confirmed risk" rule:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

Sample request payload:

```
{
"name": "Previous event is confirmed risk",
"enabled": true,
"riskScore": 100.0,
"settings": null
}
```

The default risk score for this risk rule is ***100***.

User Velocity

User Velocity rule gets triggered based on the number of times a user attempts to log into an application within a span of time. The average number of logins that triggers the rule is five times in a span of 10 minutes. This rule has been enabled in the system by default.

Use this API to update the "User Velocity" risk rule:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

Sample request payload:

```
{
"name": "User Velocity",
"enabled": true,
"riskScore": 50.0,
"settings": [
{
"key": "timeIntervalInMinutes",
"value": "10"
},
{
"key": "numberOfRiskEvaluationsPerUser",
"value": "6"
}
]
}
```

The following table describes the "***User Velocity"*** settings that can be configured using the PUT "... /iarisk/v1/RiskRules" API:

Settings Name	Description	Default V
timeIntervalInMinutes (API Terminology) Time interval in minutes (UI Terminology)	Indicates the minutes considered for user velocity calculation.	10
numberOfRiskEvaluationsPerUser (API Terminology) Number of risk evaluations per user (UI Terminology)	Indicates the max number of risk evaluations allowed per user within the defined timeInterval.	6

The default risk score for this risk rule is ***50***.

Work Day

Work Day rule checks if the user is accessing the application during work hours and working days.

Use this API to update the "Work Day" risk rule:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

Sample request payload:

```
[
{
"name": "Work day time",
"enabled": true,
"riskScore": 50.0,
"settings": [
{
"key": "workDays",
"value": "MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY"
},
{
"key": "startHour",
"value": "8"
},
{
"key": "endHour",
"value": "20"
}
]
}
]
```

The following table describes the "***Work Day"*** settings that can be configured using the PUT "... /iarisk/v1/RiskRules" API:

Settings Name	Description	Default Value
startHour	Indicates the start hour of a working day.	8
endHour	Indicates the end hour of a working day.	17
workDays	Indicates the days of the week considered as work days.	MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY

The default risk score for this risk rule is ***50***.

Inactive User

Inactive User rule checks if the user is accessing the application after a long time.

Use this API to update the "Inactive User" risk rule:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

Sample request payload:

```
{
```

```
"name": "Inactive User",
"enabled": true,
"riskScore": 50.0,
"settings": [
{
"key": "numberOfDaysUserInactive",
"value": "90"
}
]
}
```

The following setting in the "***Inactive User"*** rule can be configured using the PUT "... /iarisk/v1/RiskRules" API:

Settings Name	Description	Default Value
numberOfDaysUserInactive (API Terminology) Number of days user inactive (UI Terminology)	Indicates the maximum number of days a user is inactive after which this rule gets triggered.	90

The default risk score for this risk rule is ***50***.

Country Change

The 'Country Change' risk rule gets triggered if the latest authentication originates from a country that differs from the countries of the last N authentication attempts, assigning a risk score based on the configured threshold.

Use this API to update the "Country Change" risk rule:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

Sample request payload:

```
[
{
"name": "Country Change",
"enabled": false,
"riskScore": 50.0,
"settings": [
{
"key": "lastNRiskEvaluation",
"value": "5"
}
]
}
]
```

Configurable Setting for the "Country Change" Rule

The following setting in the "***Country Change"*** rule can be configured using the PUT "... /iarisk/v1/RiskRules" API:

Settings Name	Description	Default Value
lastNRiskEvaluation	Specifies the number of past authentications used for evaluation. If the current country from which the authentication is attempted is not found in these past **N** records, the rule is triggered.	5

The default risk score for this risk rule is ***50***.

Example

The following table illustrates how the **"Country Change"** rule behaves for a system with `lastNRiskEvaluations = 3` that is, it checks the last **3** risk evaluations for the authentication attempts):

Table 113: Country Change Risk Rule Example Scenarios

Trans action	Country	Rule Triggered?	Reason
1	India	No	Not enough previous transactions (rule requires 3 for evaluation).
2	Japan	No	Still not enough data to evaluate the rule.
3	Korea	No	Now we have 3 transactions, but still not enough data to evaluate the rule.
4	China	Yes	The country `(China)` is **not present** in the last 3 transactions (`India` , `Japan` , `Korea`), so the rule is triggered.
5	Korea	No	The country (`Korea`) **exists** in the last 3 transactions (`Japan` , `Korea` , `China`), so the rule is not triggered.
6	Korea	No	The country (`Korea`) is still in the last 3 transactions (`Korea` , `China` , `Korea`), so the rule is not triggered.
7	Cambodia	Yes	The country (`Cambodia`) is **not present** in the last 3 transactions (`China` , `Korea` , `Korea`), so the rule is triggered.

IP Address Change

The "IP Address Change" risk rule gets triggered if the latest authentication originates from an IP address that differs from the IP addresses of the last N authentication attempts, assigning a risk score based on the configured threshold.

Use this API to update the "IP Address Change" risk rule:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

Sample request payload:

```
[
{
"name": "IP Change",
"enabled": false,
"riskScore": 50.0,
"settings": [
{
"key": "lastNRiskEvaluation",
"value": "5"
}
]
}
]
```

Configurable Setting for the "IP Address Change" Rule

The following setting in the "***IPAddress Change"*** rule can be configured using the PUT "... /iarisk/v1/RiskRules" API:

Settings Name	Description	Default Value	Possible Valu
lastNRiskEvaluations	Specifies the number of past authentications used for evaluation. If the current authentication IP is not found in these past **N** records, the rule is triggered.	5	Any positive

The default risk score for this risk rule is ***50***.

Example

The following table illustrates how the **"IP Address Change"** rule behaves for a system with `lastNRiskEvaluations = 3` that is, it checks the last **3** risk evaluations for the authentication attempts):

Table 114: IP Address Change Risk Rule Example Scenarios

Transaction	IP Address	Rule Triggered?	Reason
1	1.0.0.1	No	Not enough previous transactions (rule requires 3 for evaluation).
2	1.0.0.2	No	Still not enough data to evaluate the rule.
3	1.0.0.3	No	Now we have 3 transactions, but still not enough data to evaluate the rule.
4	1.0.0.4	Yes	The IP (`1.0.0.4`) is **not present** in the last 3 transactions (`1.0.0.1`, `1.0.0.2`, `1.0.0.3`), so the rule is triggered.
5	1.0.0.3	No	The IP (`1.0.0.3`) **exists** in the last 3 transactions (`1.0.0.2`, `1.0.0.3`, `1.0.0.4`), so the rule is not triggered.
6	1.0.0.3	No	The IP (`1.0.0.3`) is still in the last 3 transactions (`1.0.0.3`, `1.0.0.4`, `1.0.0.3`), so the rule is not triggered.
7	1.0.0.2	Yes	The IP (`1.0.0.2`) is **not present** in the last 3 transactions (`1.0.0.3`, `1.0.0.4`, `1.0.0.3`), so the rule is triggered.

Reference Links:

- Managing Risk
- Evaluating the Risk Score
- Risk Service APIs
- Configuring Risk Rules using Admin Console UI

Other related links

- Custom Risk Provider
- IDSP Risk Engine
- Integrating with External Risk Provider
- Risk API Error Codes
- Using Advanced Authentication Risk Service

Evaluating a User's Risk Score

This topic describes the best practices to follow after upgrading Symantec Identity Security Platform and also explains the steps to evaluate the risk scope in IDSP.

This topic includes the following sections:

- Best Practice: Passing the "identitySourceId" in the Risk APIs after Upgrading
- Evaluating a User's Risk Score

Best Practice: Passing the "identitySourceId" in the Risk APIs after Upgrading

This section provides essential steps for maintaining user risk profile continuity when working with multiple identity stores after a system upgrade, ensuring that existing user profiles are properly mapped and new identity stores are correctly integrated with the Risk API framework.

When using the Risk API, ensure you have an identity store (such as LDAP) configured and use that store's identifier as a parameter in the Risk API calls. Whether you're working with existing identity stores or creating new ones, this practice ensures a seamless transition if those same identities become subject to authentication flows at a later time. Additionally, this approach ensures proper account usage monitoring under Credentials and Devices.

After upgrading, it is recommended to pass the "***identitySourceId***" parameter in the "/UserRiskScoreEvaluator" API and "/PostUserRiskScoreEvaluator" API. However, if the "***identitySourceIdRequired***" parameter is set to "***true***", then it is mandatory to pass the "identitySourceId" parameter value in the Risk APIs.

Use Case:

If you already have an identity store created before upgrading and want to create another identity store with non-unique users after the upgrade, you must pass the "identitySourceId" parameter of the pre-upgrade identity store in the ".../iarisk/v1/UserRiskScoreEvaluator" API and ".../iarisk/v1/PostUserRiskScoreEvaluator" API. This is a one-time operation that maps the existing user profile to the respective identity store, helping to uniquely identify the user profile.

Once the new identity store is created, you can pass either the new or the old "**identitySourceId**" value in the ".../iarisk/v1/UserRiskScoreEvaluator" API and ".../iarisk/v1/PostUserRiskScoreEvaluator" API for all subsequent transactions, as the new user profile will be mapped to the respective identity store.

The following example describes the steps in details:

1. **Pre-upgrade state**: User "nbruce" already exists in "idstore-1" before upgrading.
   The risk profile of nbruce was evaluated in the previous IDSP release.
2. **Perform the upgrade**: Complete the system upgrade process..
3. **Map existing user profile**: Run the ".../iarisk/v1/UserRiskScoreEvaluator" API and ".../iarisk/v1/PostUserRiskScoreEvaluator" API to evaluate the risk profile of "nbruce" by passing the "***identitySourceId***" of the pre-upgrade identity store. This step allows mapping the user profile to the existing idstore.
4. **Create new identity store**: Create a new LDAP identity store and keep a copy of the identitySourceId for the next step. User "nbruce" now exists in "idstore-2" (assuming the newly created identity store is named idstore-2).
5. **Map user to new identity store**: Run the ".../iarisk/v1/UserRiskScoreEvaluator" API and ".../iarisk/v1/PostUserRiskScoreEvaluator" API to evaluate the risk profile of nbruce by passing the "***identitySourceId***" of the newly created identity store. This step maps the user profile of "nbruce" to the new identity store (idstore-2).
   The outcome of this operation is as follows:
   - A new risk profile is created for user "nbruce" from idstore -2.
   - The user search filter shows two records of "nbruce", one in the idstore-1 and the other in the idstore-2.

Example of a ".../UserRiskScoreEvaluator" API payload with "identitySourceId" in the payload:

```
{
 "userId": "nbruce",
 "identitySourceId": "7b026f11-338a-4efc-b17a-b1a2e8f390ff",
 "ipAddress": "10.40.30.31",
 "userAgent": "PostmanRuntime/7.26.5",
 "action": "SIGNIN",
 "keyValuePair": [
     {
         "key": "deviceSignature",
         "value": "_v02MyUgNGgYOi88OTk0emB7ZXV9GDQ2PDshOiY9bnUcOy..."
     }
 ]
}
```

Example of a ".../PostUserRiskScoreEvaluator" API payload where "identitySourceId" parameter is passed:

```
{
```

```
  "identitySourceId": "7b026f11-338a-4efc-b17a-b1a2e8f390ff",
  "userId": "{{userId}}",
  "eventId": "{{eventId}}",
  "riskConfirmed": false,
  "verifyMethod": "DENYRISK",
  "action": "SIGNIN",
  "rememberDevice": true,
  "keyValuePair": [
      {
          "key": "deviceSignature",
          "value": "_v02MyUgNGgYOi88OTk0emB7ZXV9..."
      }
  ]
}
```

Evaluating a User's Risk Score

Follow these steps to evaluate the risk score of a user transaction.

1. The "***userRiskScoreEvaluator"*** API is used to evaluate the risk of a transaction. This API returns a risk score based on risk evaluation. Higher the risk score, the more the risk associated with the transaction. It returns the name of all the risk rules that got triggered during risk evaluation. It also returns the device tag for the device that is used for risk evaluation. This is a POST call and the URL of the API is -

   ```
   https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/UserRiskScoreEvaluator
   ```

   NOTE

   - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
   - The default value for `<tenantName>` is "default".
   - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
     For example:
     `https://{{host}}/{{tenantName}}/iarisk/v1/UserRiskScoreEvaluator`
     should look similar to:
     `https://example.com/default/iarisk/v1/UserRiskScoreEvaluator`
   - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

   The request payload of "userRiskScoreEvaluator" is shown here with the example of IP Address as the configured risk rule:

   ```
   {
    "identitySourceId": "<identitySourceId-optional-payload>",
    "userId": "{{userId}}",
    "ipAddress": "10.40.30.31",
    "userAgent": "PostmanRuntime/7.26.5",
    "action":"SIGNIN",
    "keyValuePair": [
        {
            "key": "deviceSignature",
            "value": "_v02MyUgNGgYOi88OTk0emB7ZXV9GDQ2PDshOiY9bnUcOyEwOXUYNDZ1GgZ1DXVkZQpkYApjfHUUJSU5M....."
        }
    ]
   }
   ```

   Where,

- identitySourceId - specifies the identity source Id
- userId- specifies the user Id for whom the risk score has to be evaluated
- ipAddress- specifies the ipAddress of the user device
- userAgent- specifies the user agent of the device from where the transaction is initiated
- action- Indicates the action under which the users risk score is evaluated. Possible values are - SIGNIN, SESSION, GOVERNANCE, PRIVIEGEDACCESS. Default value is SIGNIN.
- keyValuePairs:
  - deviceSignature- specifies the device signature of the device that is used by the user for transaction
    Example of Device Signature
    Device Signature in Encoded form (The encoded form of device signature should be used by the user)

    ```
    _v02MyUgNGgYOi88OTk0emB7ZXV9GDQ2PDshOiY9bnUcOyEwOXUYNDZ1Gg......
    ```

    Device Signature in decoded form

    ```
    fpua=Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko)
     Chrome/87.0.4280.67 Safari/537.36|MacIntel&fpsc=24|1680|1050|1680|1050&fpsw=internal-pdf-
    viewer|mhjfbmdgcfjbbpaeojofohoefgiehjai|internal-nacl-plugin&fppi=&fptz=5.5&fpln=en-GB|||
    &fpcp=01110111111111111111111011111111111101111011111111111111111111111111010&fpvc=&fpvs=2&fpts=&fptg=
    ```

The response payload of "userRiskScoreEvaluator" is shown here:

```
{
 "risky": true,
 "riskScore": 100,
 "riskThreshold": 50,
 "riskReason": "Allowed IP, User Velocity, Blocked IP, Device recognition, Inactive User",
 "eventId": "54258",
 "userId": "d77e9b3a-e8fd-4e20-b2b6-6dbf8814c2ab",
 "keyValuePair": [
     {
         "key": "device.registered",
         "value": "false"
     },
     {
         "key": "device.registered.reason",
         "value": "Registered device not detected: invalid tag"
     },
     {
         "key": "device.shared",
         "value": "false"
     },
     {
         "key": "device.tag",
         "value": "e2ad454f36204a0cb4eafe0091cee497"
     },
     {
         "key": "location.ip.coordinates",
         "value": "-1.0,-1.0"
     },
     {
         "key": "location.ip.country.state.city",
         "value": "UNKNOWN/UNKNOWN/UNKNOWN"
     }
 ],
 "rulesTriggered": [
```

```
    {
        "ruleName": "Allowed IP",
        "reason": " Matched enterprise custom allowed IP for IP:170401311"
    },
    {
        "ruleName": "Device recognition",
        "reason": " invalid tag. ",
        "additionalDetails": [
            {
                "key": "deviceRegistered",
                "value": "false"
            },
            {
                "key": "deviceRegisteredReason",
                "value": "Registered device not detected: invalid tag"
            },
            {
                "key": "deviceShared",
                "value": "false"
            },
            {
                "key": "deviceTag",
                "value": "e2ad454f36204a0cb4eafe0091cee497"
            },
            {
                "key": "deviceMatchedTag",
                "value": "d5223700d51b8de831bffba76d83ff35646f8333"
            }
        ]
    },
    {
        "ruleName": "User Velocity",
        "reason": " User velocity (7) exceeded the threshold (3). ",
        "additionalDetails": [
            {
                "key": "currentValue",
                "value": "7"
            },
            {
                "key": "thresholdValue",
                "value": "3"
            }
        ]
    },
    {
        "ruleName": "Inactive User",
        "reason": " User is last active on 2023-04-28 06:04:31.0 and this is beyond the user inactive
threshold of 0 days. "
    },
    {
        "ruleName": "Blocked IP",
        "reason": " Matched enterprise custom blocked IP for IP:170401311. "
    }
```

```
  ]
 }
```

Where,

- risky- specifies whether the transaction carries risk or not.
- riskScore- specifies the score of risk associated with the risk transaction.
- riskThreshold- defines the threshold of risk associated with a transaction, helping in the assessment of its level of riskiness.
- riskReason- specifies all the reasons responsible for the risk score.
  Supported risk reasons are: Allowed IP', 'Behavior anomaly', 'Blocked IP', 'Device recognition', 'Enhanced Difficult Travel', 'Fraud IP', 'Inactive User', 'Previous event is confirmed risk', 'RC result', 'Risky Country', 'User Velocity', 'Work day time'.
- eventId- specifies the unique event Id linked to this transaction involving risk.
- userId- specifies the unique user Id for whom the risk score has been evaluated.
- keyValuePair- represents the pairs of keys and values used in assessing risk. These are utilized for internal purposes.
- rulesTriggered- represents the rules that have been activated or triggered for the user concerning that specific transaction.

Example of IpAddress risk rule supporting IPv6 format that is sent in as part of the "/UserRiskScoreEvaluator" API

```
 {
  "identitySourceId": "<identitySourceId-optional-payload>",
  "userId": "{{userId}}",
  "ipAddress": "3001:db8:3333:4444:5555:6666:7777:8888",
  "userAgent": "PostmanRuntime/7.26.5",
  "action":"SIGNIN",
  "keyValuePair": [
      {
          "key": "deviceSignature",
          "value": "_v02MyUgNGgYOi88OTk0emB7ZXV9GDQ2P....."
      }
  ]
 }
```

Example of Recognized Computer risk rule that is sent in as part of the "/UserRiskScoreEvaluator" API

```
 {
  "identitySourceId": "<identitySourceId-optional-payload>",
  "userId": "{{userId}}",
  "ipAddress": "10.20.30.40",
  "userAgent": "PostmanRuntime/7.26.5",
  "action":"SIGNIN",
  "keyValuePair": [
      {
             "key": "SECONDARY_AUTH_TYPE",
             "value": "RC"
         },
          {
             "key": "SECONDARY_AUTH_RESULT",
             "value": "True"
         }
  ]
 }
```

This risk rule checks if the event is initiated from a recognized computer. A device is marked as recognized or not, based on the "SECONDARY_AUTH_TYPE" and "SECONDARY_AUTH_RESULT" parameters sent in the "/UserRiskScoreEvaluator" API. If the SECONDARY_AUTH_RESULT is set to false, then the engine treats the device as not recognized and gives the maximum risk score. If not, it gives a negative score for this rule.

2. The ***"PostUserRiskScoreEvaluator"*** API is used to update the risk context for the user for which the risk evaluation is performed as part of UserRiskScoreEvaluator API. This API is invoked if the UserRiskScoreEvaluator returns a riskscore higher than the riskThreshold. If higher risk score is returned by UserRiskScoreEvaluator API, based on configured authentication policy, user may have to perform a step-up authentication. Result of step-up authentication (whether true or false) is indicated by the riskConfirmed parameter. If the result is ***true****, it* indicates that the authentication has failed and if the result is ***false****, it* indicates that the authentication was successful and the transaction is not risky.

   This is a POST call and the URL of the API is -

   ```
   https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/PostUserRiskScoreEvaluator
   ```

   The request payload of "PostUserRiskScoreEvaluator" is shown here:

   ```
   {
    "eventId": "{{eventId}}",
    "userId": "{{userId}}",
    "riskConfirmed": false,
    "verifyMethod": "DENYRISK",
    "action": "SIGNIN",
    "rememberDevice": true,
    "keyValuePair": [
        {
            "key": "deviceSignature",
            "value": "_v02MyUgNGgYOi88OTk0emB7ZXV9GDQ2PDshOiY9bnUcOyEwOXUYNDZ1GgZ1DXVk...."
        }
    ]
   }
   ```

   Where,

   - eventId- string which identifies the transaction in the risk server
   - userId- specifies the user Id for whom the risk score has to be evaluated
   - riskConfirmed- specifies the authentication status of step-up authentication. If the result is true, it indicates that the authentication has failed and risk is confirmed, if the result is false, it indicates that the authentication was successful and it is not a risky transaction.
   - verifyMethod - indicates the outcome of the risk mitigation process in relation to the detected risk. It has two possible values:
     - DENYRISK: This value signifies that the authentication attempt successfully confirmed the user's identity, thereby mitigating the initial risk. In this scenario, despite an initial high risk score, the user successfully passed a step-up authentication, leading to the risk being effectively "denied" or invalidated.
     - CONFIRMRISK: This value indicates that the authentication attempt, even after any potential step-up, still showed signs of risk, thereby confirming that the transaction was indeed risky.
   - action - Indicates the action under which the users risk score is evaluated. Possible values are - SIGNIN, SESSION, GOVERNANCE, PRIVIEGEDACCESS. Default value is SIGNIN
   - rememberDevice - indicates that the device is to be registered with risk engine
   - keyValuePairs
     - deviceSignature - specifies the device signature of the device that is used by the user for transaction

   The response payload of "PostUserRiskScoreEvaluator" is shown here:

   ```
   {
    "userId": "5f246fb9-f17d-4bd4-8eef-4f0523eae68b",
    "eventId": "54204",
   ```

```
"status": "success",
"keyValuePair": [
    {
        "key": "device.feedback",
        "value": "false"
    }
],
"internalKeyValuePair": [
    {
        "key": "be.feedback",
        "value": "true"
    },
    {
        "key": "input.userid",
        "value": "5f246fb9-f17d-4bd4-8eef-4f0523eae68b"
    },
    {
        "key": "input.vipaccount",
        "value": "58a273d2-8cac-484f-ac30-e32d25f424b0"
    },
    {
        "key": "re.feedback.case.casestatus",
        "value": "3"
    },
    {
        "key": "re.feedback.case.id",
        "value": "13084"
    },
    {
        "key": "re.feedback.case.modifieddate",
        "value": "Fri Feb 05 07:11:11 GMT 2021"
    },
    {
        "key": "re.feedback.case.verifymethod",
        "value": "DENYRISK"
    },
    {
        "key": "re.feedback.denyrisk",
        "value": "true"
    },
    {
        "key": "re.feedback.suspiciousevent.id",
        "value": "13284"
    },
    {
        "key": "re.feedback.suspiciousevent.newstatus",
        "value": "UNSUSPICIOUS"
    },
    {
        "key": "re.feedback.suspiciousevent.oldstatus",
        "value": "UNSUSPICIOUS"
    }
]
```

```
}
```

Where,

- userId- specifies the unique user Id for whom the risk score has been evaluated.
- eventId- specifies the unique event Id linked to this transaction involving risk.
- status-specifies if the risk score update was successful.
- keyValuePairs- represents the pairs of keys and values used in assessing risk. These are used for internal purposes.
- internalKeyValuePair- represents the pairs of keys and values used post evaluating risk. These are used for internal purposes.

3. Use the "***userRiskScoreGetter***" API to retrieve the risk score for a specific user ID.
   This is a GET call and the URL of the API is -

```
https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/UserRiskScore/{{userId}}
```

   Sample response payload of "UserRiskScore" is shown here:

```
{
 "userId": "5f246fb9-f17d-4bd4-8eef-4f0523eae68b",
 "riskScore": 100,
 "vectors": [
     {
         "action": "SIGNIN",
         "score": 100
     }
 ]
}
```

Risk Service APIs

This topic lists the Symantec Identity Security Platform Risk APIs and its examples.

The following table lists out the Risk endpoint URLs that are available to perform CRUD operations to evaluate risks for the incoming transactions. For sample request and response payloads of all these URLs, refer to Risk API Swagger documentation.

As an admin, you can use the following Risk APIs to create, retrieve, update, and delete risk rules.

Admin Risk APIs (scope - "urn:iam:t.configs")

Operation Description	Operation Type	Endpoint URLs
Setting the Risk Threshold	PUT	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/RiskThreshold
Retrieving Risk Threshold	GET	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/RiskThreshold See examples here .
Setting Risky Countries List	PUT	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/RiskyCountries
Retrieve Risky Countries List	GET	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/RiskyCountries See examples here.
Setting Allowed IP List	PUT	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/IPList/ALLOWED
Retrieving Allowed IP List	GET	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/IPList/ALLOWED
Setting Blocked IP List	PUT	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/IPList/BLOCKED

Retrieving Blocked IP List	GET	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/IPList/BLOCKED See examples here .
Uploading the list for allowed or blocked IPs	POST	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/EnterpriseIPListUploader
Retrieving the upload status of the allowed or blocked IP list	GET	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/EnterpriseIPListUploader/statu ipListType=ALLOWED/BLOCKED
Downloading a file containing Allowed or Blocked IP addresses	GET	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/EnterpriseIPListUploader
Deleting the Allowed or Blocked IP List	DELETE	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/EnterpriseIPListUploader/statu ipListType=ALLOWED/BLOCKED See examples here.
Retrieving all known devices of a user	GET	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/KnownDevices/{{userId}}
Retrieving a specific Known Device for a user ID	GET	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/KnownDevices/{{userId}}/{{dev eId}}
Deleting a specific Known Device for a user ID	DELETE	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/KnownDevices/<userID>/<dev eId>
Deleting all Known Devices for a specific user ID	DELETE	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/KnownDevices/<userID> See examples here .
Updating the risk rules for a specific tenant	PUT	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
Retrieving risk rules for a specific tenant	GET	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules See examples here .
Admin Risk Evaluation APIs (Scope -"urn:iam:t.risk")		
Evaluating the risk score of a user	POST	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/UserRiskScoreEvaluator See examples here .
Post User Risk Score Evaluator This API is invoked only if the UserRiskScoreEvaluator API returns the risky flag as "true".	POST	https://${IDSP_FQDN}/default/iarisk/v1/PostUserRiskScoreEvaluator See examples here .
Retrieve the risk score of a given user.	GET	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/UserRiskScore/{userId} See examples here .

User APIs to Manage Known Devices (Scope - "urn:iam:m.me")

As a end-user, you can use the following Risk APIs to retrieve and delete your known devices.

Operation Description	Operation Type	Endpoint URLs
Retrieving all known devices of a user	GET	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/MeKnownDevices
Deleting all Known Devices for a specific user	DELETE	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/MeKnownDevices
Retrieving a specific Known Device for a user ID	GET	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/MeKnownDevices/{deviceId}
Deleting a specific Known Device for a user ID	DELETE	https://${IDSP_FQDN}/{{tenantName}}/iarisk/v1/MeKnownDevices/{deviceId} See examples here .

NOTE

- To know the value of ${IDSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/RiskRules`
  should look similar to:
  `https://example.com/default/admin/v1/RiskRules`

Risk Rules API example

This topic outlines how to manage risk rules, which are crucial for assessing user transaction risk. It covers updating various rule types, including their enablement, risk scores, and specific settings.

Setting Risk Rules

This section explains how to configure and update various risk rules within the system. These rules, such as "Enhanced Difficult Travel" and "Inactive User," each have a configurable risk score and specific settings that influence how transactions are evaluated for risk.

Invoke this API to update the Risk Rules:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskRules
```

Request

```
[
   {
       "name": "Enhanced Difficult Travel",
       "enabled": true,
       "riskScore": 50.0,
       "settings": [
           {
               "key": "maxSharedSessionsPerUser",
               "value": "1"
           },
           {
               "key": "maxSpeedUserCanTravel",
               "value": "500"
           },
           {
               "key": "maxDistanceTolerance",
               "value": "50"
           }
       ]
   },
   {
       "name": "Inactive User",
       "enabled": true,
       "riskScore": 50.0,
       "settings": [
           {
               "key": "numberOfDaysUserInactive",
               "value": "90"
           }
```

```
        ]
    },
    {
        "name": "User Velocity",
        "enabled": true,
        "riskScore": 50.0,
        "settings": [
            {
                "key": "timeIntervalInMinutes",
                "value": "10"
            },
            {
                "key": "numberOfRiskEvaluationsPerUser",
                "value": "6"
            }
        ]
    },
    {
        "name": "Work day time",
        "enabled": true,
        "riskScore": 50.0,
        "settings": [
            {
                "key": "workDays",
                "value": "MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY"
            },
            {
                "key": "startHour",
                "value": "8"
            },
            {
                "key": "endHour",
                "value": "20"
            }
        ]
    }
]
```

Response

```
[
 {
   "name": "Allowed IP",
   "enabled": true,
   "riskScore": 0,
   "settings": null
 },
 {
   "name": "Behavior anomaly",
   "enabled": true,
   "riskScore": 50,
   "settings": [
     {
       "key": "enableStaticScore",
```

```
      "value": "false"
    }
  ]
},
{
  "name": "Blocked IP",
  "enabled": true,
  "riskScore": 100,
  "settings": null
},
{
  "name": "Client Velocity",
  "enabled": true,
  "riskScore": 50,
  "settings": [
    {
      "key": "timeIntervalInMinutes",
      "value": "10"
    },
    {
      "key": "numberOfRiskEvaluationsPerClient",
      "value": "6"
    }
  ]
},
{
  "name": "Country Change",
  "enabled": false,
  "riskScore": 50,
  "settings": [
    {
      "key": "lastNRiskEvaluations",
      "value": "5"
    }
  ]
},
{
  "name": "Device recognition",
  "enabled": true,
  "riskScore": 50,
  "settings": [
    {
      "key": "deviceFPExpiryMaxBindingLife",
      "value": "90"
    },
    {
      "key": "registeredmobile",
      "value": "-25"
    },
    {
      "key": "deviceIdPct",
      "value": "-25"
    },
```

```
      {
        "key": "simThreshold",
        "value": "2"
      },
      {
        "key": "deviceLimitAction",
        "value": "deleteOldest"
      },
      {
        "key": "registeredPct",
        "value": "-25"
      },
      {
        "key": "maxNumberOfDevicesPerUser",
        "value": "-1"
      },
      {
        "key": "lastNRiskEvaluations",
        "value": "-1"
      },
      {
        "key": "unrecognizedPct",
        "value": "100"
      },
      {
        "key": "tagExpireInSeconds",
        "value": "1800"
      },
      {
        "key": "deviceFPExpiryInactivityTime",
        "value": "360"
      }
    ]
  },
  {
    "name": "Enhanced Difficult Travel",
    "enabled": true,
    "riskScore": 50,
    "settings": [
      {
        "key": "maxSpeedUserCanTravel",
        "value": "500"
      },
      {
        "key": "maxSharedSessionsPerUser",
        "value": "1"
      },
      {
        "key": "maxDistanceTolerance",
        "value": "50"
      }
    ]
  },
```

```
{
  "name": "Fraud IP",
  "enabled": true,
  "riskScore": 100,
  "settings": null
},
{
  "name": "Inactive Client",
  "enabled": true,
  "riskScore": 50,
  "settings": [
    {
      "key": "numberOfDaysClientInactive",
      "value": "90"
    }
  ]
},
{
  "name": "Inactive User",
  "enabled": true,
  "riskScore": 50,
  "settings": [
    {
      "key": "numberOfDaysUserInactive",
      "value": "90"
    }
  ]
},
{
  "name": "IP Change",
  "enabled": false,
  "riskScore": 50,
  "settings": [
    {
      "key": "lastNRiskEvaluations",
      "value": "5"
    }
  ]
},
{
  "name": "Previous event is confirmed risk",
  "enabled": true,
  "riskScore": 100,
  "settings": null
},
{
  "name": "RC result",
  "enabled": true,
  "riskScore": 100,
  "settings": [
    {
      "key": "recognizedScore",
      "value": "-25"
```

```
      }
    ]
  },
  {
    "name": "Risky Country",
    "enabled": true,
    "riskScore": 100,
    "settings": null
  },
  {
    "name": "User Velocity",
    "enabled": true,
    "riskScore": 50,
    "settings": [
      {
        "key": "timeIntervalInMinutes",
        "value": "10"
      },
      {
        "key": "numberOfRiskEvaluationsPerUser",
        "value": "6"
      }
    ]
  },
  {
    "name": "Work day time",
    "enabled": true,
    "riskScore": 50,
    "settings": [
      {
        "key": "endHour",
        "value": "20"
      },
      {
        "key": "workDays",
        "value": "MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY"
      },
      {
        "key": "startHour",
        "value": "8"
      }
    ]
  }
]
```


Reference Links

- Understanding Risk Rules

Risky Country API Examples

This topic details API interactions for managing the Risky Countries List, allowing users to retrieve the current list and update it by adding or removing country codes.

This section provides examples for both retrieving and updating the list of countries designated as "risky." You can use the GET request to view the current list and the PUT request to modify it by providing new country codes.

Invoke this API to retrieve the Risky Countries List:

```
GET https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskyCountries
```

Response

```
{
"countryCodes": [
  "by",
  "cg",
  "ci",
  "cu",
  "iq",
  "ir",
  "kp",
  "lr",
  "sd",
  "sy",
  "zw"
]
}
```

Invoke this API to update the Risky Countries List:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskyCountries
```

Request

```
{
"countryCodes": [
  "by"
]
}
```

Response: 200 OK

```
{
"countryCodes": [
  "by"
]
}
```

Risk Threshold API Example

This topic explains the Risk Threshold API, which defines a limit to determine if a user transaction is high or low risk based on its calculated risk score.

The Risk Threshold specifies a limit that enables the risk engine to confirm by comparing the risk score with the risk threshold, whether a specific user transaction is at high risk or low risk. If a transaction returns a risk score higher than the risk threshold, it is considered to be a high risk transaction, in such a case the risk is set to true.

Invoke this API to update the Risk Threshold:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/RiskThreshold
```

Request

```
{
"riskThreshold": 50
}
```

Response

```
{
```

```
"riskThreshold": 50
}
```

IP List API Examples

This topic provides examples for managing IP lists (Allowed and Blocked) using various API endpoints, including setting, uploading, and checking the status of these lists.

Setting Allowed IP List

This section details how to update the Allowed IP List, providing the API endpoint, request, and response payloads.

Invoke the API to update the Allowed IP List:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/IPList/ALLOWED
```

Request

```
{
"ipAddressList": {
"ipAddressRange": [
{
"fromIP": "20.40.30.30",
"toIP": "20.40.30.80"
}
],
"ipAddress": [
"20.40.40.40"
]
}
}
```

Response

```
{
"count": 2,
"ipAddressList": {
"ipaddressRange": [
{
"fromIP": "20.40.30.30",
"toIP": "20.40.30.80"
}
],
"ipaddress": [
"20.40.40.40"
]
},
"iplistType": "ALLOWED"
}
```

IPv6 format supported in "/IPList/ALLOWED"

This section details how IPv6 addresses can be included in the request payload for the Allowed IP List.

```
{
"ipAddressList": {
```

```
"ipAddressRange": [
{
"fromIP": "2001:db8:3333:4444:5555:6666:7777:8888",
"toIP": "2001:db8:3333:4444:CCCC:DDDD:EEEE:FFFF"
}
],
"ipAddress": [
"2001:0000:130F:0000:0000:09C0:876A:130B"
]
}
}
```

Setting Blocked IP List

This section details how to update the Blocked IP List, including the API endpoint, request, and response examples.

Invoke this API to update the Blocked IP List:

```
PUT https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/IPList/BLOCKED
```

Request

```
{
"ipAddressList": {
"ipAddressRange": [
{
"fromIP": "10.40.30.30",
"toIP": "10.40.30.80"
}
],
"ipAddress": [
"10.40.40.40"
]
}
}
```

Response

```
{
"count": 2,
"ipAddressList": {
"ipaddressRange": [
{
"fromIP": "10.40.30.30",
"toIP": "10.40.30.80"
}
],
"ipaddress": [
"10.40.40.40"
]
},
"iplistType": "BLOCKED"
}
```

IPv6 format supported in "/IPList/BLOCKED"

This section details the use of IPv6 addresses when setting the Blocked IP List.

```
{
"ipAddressList": {
"ipAddressRange": [
{
"fromIP": "3001:db8:3333:4444:5555:6666:7777:8888",
"toIP": "3001:db8:3333:4444:CCCC:DDDD:EEEE:FFFF"
}
],
"ipAddress": [
"3001:0000:130F:0000:0000:09C0:876A:130B"
]
}
}
```

Upload Allowed or Blocked IP List

This section details how to upload a file containing Allowed or Blocked IP addresses, specifying supported file types and the request format.

File type support:

- CSV and TXT file

  NOTE

  - The input file should contain one IP entry per line. Each line can specify a single IP address, a comma-separated IP range (for example, 198.51.100.240,198.51.100.255), or an IP address in Classless Inter-Domain Routing (CIDR) format.
  - A single version of the allowed or blocked IP list is maintained. Any new upload for these lists will overwrite the prior version.

Invoke this API to upload a IP List. The following example shows the upload of Allowed IP List.

```
POST https://{{sspHost}}/{{apiPathTenant}}/iarisk/v1/EnterpriseIPListUploader
```

Request

Key	Value
ipListType	ALLOWED (To upload blocked IP List, specify the value as BLOCKED)

file	{
	"ipListType" : "Allowed",
	"filename": <file>
	}
	To upload a blocked IP list, specify the "ipListType" as "Blocked".

Response

```
{
"status": "Upload Initiated",
"message": "Upload initiated successfully. Processing in background."
}
```

Upload Status of Allowed or Blocked IP List

This section details how to retrieve the upload status of the Allowed or Blocked IP List using a GET request with a query parameter-"ipListType=ALLOWED" or "ipListType=BLOCKED".

Invoke the API to get the status of the uploaded IP List. The following example shows the upload of allowed IP list.

```
GET https://{{sspHost}}/{{apiPathTenant}}/iarisk/v1/EnterpriseIPListUploader/status?ipListType=ALLOWED
```

Response

```
{
"status": "Upload Completed",
"uploadStartTime": "11-07-2025 06:08:42",
"uploadEndTime": "11-07-2025 06:08:42"
}
```

Download Allowed or Blocked IP List

This section details the process of downloading a file containing Allowed or Blocked IP addresses, including the API endpoint and request format.

Invoke this API to download the IP List. The following example shows the download of allowed IP List.

```
GET https://{{sspHost}}/{{apiPathTenant}}/iarisk/v1/EnterpriseIPListUploader
```

Request

Key	Value
ipListType	Allowed (To download blocked IP List, specify the value as Blocked)

Response: If the file is small in size, the result is displayed directly in the postman. However, if the file size is large, the postman triggers a pop-up asking you to save the file in your system.

The following example shows downloading a small allowed IP list:

```
# Enterprise IP List Export
# Enterprise ID: 181b8f45-e1dc-45cf-8d5b-7811b6c4f19b
# IP List Type: ALLOWED
# Export Date: 2025-11-07T06:20:45.475063915
#
192.168.10.0
192.168.20.0
192.168.30.0

# Export completed
# Total records exported: 3
# Errors encountered: 0
```

Delete Allowed or Blocked IP List

This section describes how to delete the Allowed or Blocked IP List using a DELETE request with the query parameter -"ipListType=ALLOWED" or "ipListType=BLOCKED".

Invoke this API to delete the IP List. The following example shows the deletion of allowed IP list.

```
DELETE https://{{sspHost}}/{{apiPathTenant}}/iarisk/v1/EnterpriseIPListUploader/status?ipListType=ALLOWED
```

Response

```
204 No Content
```

Known Devices API Examples

This topic provides API examples for retrieving known device information for users, including all devices associated with a user, specific devices by ID, and personal known devices.

Getting Known Devices of a user

This section explains how to fetch all known devices for a specific user using their User ID.

Invoke this API to get all known devices for a user:

Request

```
GET https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/KnownDevices/{{userId}}
```

Response


```
[
{
"deviceDetails": "Device:Mac, Browser:Chrome 144.0.0.0, OS:macOS 10.15.7",
"lastUsed": "2025-06-11 05:54 AM UTC",
"userId": "104c07c5-420b-46d0-9f57-9c4650d27ea7",
"deviceId": "5e23703806f756fb40f8b990c7b6a356d0a65633",
"userAgent": "PostmanRuntime/7.26.5",
"country": "india",
"state": "maharashtra",
"city": "mumbai",
"ipAddress": "2001:df6:6f80:1:0:0:0:205"
},
{
"deviceDetails": "Device:PC, Browser:Chrome, OS:Macintosh",
"lastUsed": "2025-06-12 10:38 AM UTC",
"userId": "104c07c5-420b-46d0-9f57-9c4650d27ea7",
"deviceId": "7b27f1cc079ae9108868517e0dea243c041a6706"
"userAgent": "PostmanRuntime/7.26.5",
"country": "india",
"state": "maharashtra",
"city": "mumbai",
"ipAddress": "2001:df6:6f80:1:0:0:0:205"
}
]
```


Getting Known Device of a user based on the device Id

This section describes how to retrieve a specific known device for a user by providing both the User ID and Device ID.

Invoke this API to get a specific known device for a user:

Request

```
GET https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/KnownDevices/{{userId}}/{{deviceId}}
```

Response

```
[
{
"deviceDetails":"Device:Mac, Browser:Chrome 144.0.0.0, OS:macOS 10.15.7",
"lastUsed": "2025-06-12 10:38 AM UTC",
"userId": "104c07c5-420b-46d0-9f57-9c4650d27ea7"
"deviceId": "7b27f1cc079ae9108868517e0dea243c041a6706"
"userAgent": "PostmanRuntime/7.26.5",
"country": "india",
"state": "maharashtra",
"city": "mumbai",
"ipAddress": "2001:df6:6f80:1:0:0:0:205"
}
]
```


Getting Me Known Devices

This section details how to retrieve all known devices associated with the currently authenticated user.

Invoke this API to get all your known devices:

Request

```
GET https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/MeKnownDevices
```

Response


```
[
{
"deviceDetails": "Device:Mac, Browser:Chrome 144.0.0.0, OS:macOS 10.15.7",
"lastUsed": "2025-06-11 05:54 AM UTC",
"userId": "104c07c5-420b-46d0-9f57-9c4650d27ea7",
"deviceId": "5e23703806f756fb40f8b990c7b6a356d0a65633",
"userAgent": "PostmanRuntime/7.26.5",
"country": "india",
"state": "maharashtra",
"city": "mumbai",
"ipAddress": "2001:df6:6f80:1:0:0:0:205"
},
{
"deviceDetails": "Device:Mac, Browser:Chrome 144.0.0.0, OS:macOS 10.15.7",
"lastUsed": "2025-06-12 10:38 AM UTC",
"userId": "104c07c5-420b-46d0-9f57-9c4650d27ea7",
"deviceId": "7b27f1cc079ae9108868517e0dea243c041a6706"
"userAgent": "PostmanRuntime/7.26.5",
"country": "india",
"state": "maharashtra",
"city": "mumbai",
"ipAddress": "2001:df6:6f80:1:0:0:0:205"
}
]
```


Getting Me Known Devices based on the device Id

This section shows how to retrieve a specific known device for the currently authenticated user by providing its Device ID.

Invoke this API to get a specific known device for yourself:

Request

```
GET https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/MeKnownDevices/{{deviceId}}
```

Response

```
 [
 {
 "deviceDetails": "Device:Mac, Browser:Chrome 144.0.0.0, OS:macOS 10.15.7",
 "lastUsed": "2025-06-12 10:38 AM UTC",
 "userId": "104c07c5-420b-46d0-9f57-9c4650d27ea7"
 "deviceId": "7b27f1cc079ae9108868517e0dea243c041a6706"
 "userAgent": "PostmanRuntime/7.26.5",
 "country": "india",
 "state": "maharashtra",
 "city": "mumbai",
 "ipAddress": "2001:df6:6f80:1:0:0:0:205"
 }
 ]
```

Deleting Me Known Devices based on the device Id

This section shows how to delete a specific known device of a user by providing its Device Id.

Invoke this API to delete a specific known device for a user ID:

Request

```
DELETE https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/MeKnownDevices/{{deviceId}}
```

Response

```
 204 (NO_CONTENT)
```

Deleting all Known Devices for a specific user

This section shows how to delete all known devices of a specific user.

Invoke this API to delete all known devices of a user:

Request

```
DELETE https://${SSP_FQDN}/{{tenantName}}/iarisk/v1/MeKnownDevices
```

Response

```
 204 (NO_CONTENT)
```

Using Advanced Authentication Risk Service

In addition to the Symantec Identity Security Platform risk service provider, the solution also provides its users with an alternative risk service by integrating with the Advanced Authentication (AA) risk service.

To integrate AA risk service with IDSP, as a System Admin, you must perform the following configuration steps:

1. Use the "patchGlobalSettings" call to update the default value of "riskProviderPrePrimaryAuth" property from IDSP to Advanced Authentication. The default value is SSP. However, if you are using the AA risk service, then modify the value to "AARISK" as shown in the following example.

NOTE
To know about "riskProviderPrePrimaryAuth" property, see Settings page.

The following is the API URL of a PATCH call to update global settings:

```
https://${SSP_MGMT_FQDN}/system/admin/v1/Settings
```

The request payload of "patchGlobalSettings" is shown here:

```
[
    {
        "name": "riskProviderPrePrimaryAuth",
        "value": "AARISK"
    }
]
```

2. Use the following PATCH RiskProvider call to configure the Advanced Authentication endpoint:

```
https://${SSP_MGMT_FQDN}/system/admin/v1/RiskProviders/aarisk
```

The request payload of "updateAARisk" is shown here:

```
[
        {
            "name": "aaRiskCertAlias",
            "value": "aarisk"
        },
        {
            "name": "aaRiskAPIEndpoint",
            "value": "http://<AARISK FQDN>:<PORT>/aa-restapi/ca/advancedauth/v1"
        },
        {
            "name": "aaRiskOrg",
            "value": "<SSPOrgName>"
        },
        {
            "name": "aaAdmin",
            "value": "<globalAdminUser>"
        },
        {
            "name": "aaRiskEnabled",
            "value": "true"
        },
        {
            "name": "aaAdminOrg",
            "value": "<globalAdminOrgName>"
        },
        {
            "name": "aaAdminCred",
            "value": "<globaladminPassword>"
        }
]
```

Where,

– aaRiskCertAlias- denotes alias of trusted certificate for AARISK REST API service.
– aaRiskAPIEndpoint- denotes URL of AARISK REST API
  - **<AARISKFQDN>** - This is the application server's fully qualified domain name where *"aarisk-restapi.war"* got deployed pointing to the AARISK server.
  - **<PORT>** - This is the application server's configured port number.
– aaRiskOrg- denotes organization of the user

    - **<SSPOrgName>** - This is the organization which contains the IDSP User Store details.
  - aaAdmin- denotes a global admin of AARisk Service
    - **<globalAdminUser>** - This is the global admin user which is used to generate authtoken for rest calls.
  - aaRiskEnabled- denotes whether AA Risk Service is enabled or not
    - This property must be set to true to get the riskscore from AARISK. Ensure that the "isRiskServiceEnabled" property is enabled in the global settings.
  - aaAdminOrg- denotes organization of the global admin
    - **<globalAdminOrgName>** - This is the name of the organization where a global admin user is created. This user is utilized to generate authtoken for rest calls.
  - aaAdminCred- denotes password of the global admin
    - **<globaladminPassword>** - This is the password of the global admin user which is used to generate authtoken for rest calls.
3. Once configured, if you want to retrieve the Advanced Authentication endpoint details, use the following RiskProvider call:

   ```
   https://${SSP_MGMT_FQDN}/system/admin/v1/RiskProviders/aarisk
   ```

   The above steps complete the configuration between IDSP and Advanced Authentication risk server and the AARisk Engine will be ready to start providing its service.

The following table displays the list of APIs that an administrator can use to manage the Advanced Authentication risk provider configurations:

Operation Description	Operation Type	Endpoint URLs
System administrator retrieves all risk providers that are configured in the solution	GET	https://${SSP_MGMT_FQDN}/system/admin/v1/RiskProvider
System administrator retrieves the risk provider that is configured in the solution by specifying the risk provider name	GET	https://${SSP_MGMT_FQDN}/system/admin/v1/RiskProviders/aarisk
System administrator updates configurations of a risk provider	PATCH	https://${SSP_MGMT_FQDN}/system/admin/v1/RiskProviders/aarisk

NOTE

- Ensure that "AA" users are added to the IDSP User Store.
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://${SSP_MGMT_FQDN}/<systemtenantName>/admin/v1/RiskProviders/aarisk`
  should look similar to:
  `https://mgmt-example.com/system/admin/v1/RiskProviders/aarisk`
- To know the value of ${SSP_MGMT_FQDN} and ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<systemTenantName>` is "system".
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Reference Links to:

- Risk Provider API Swagger

Custom Risk Providers

Learn about Custom Risk Provider, its APIs, and examples in Symantec Identity Security Platform.

This topic includes the following sections:

- Custom Risk Provider
- Custom Risk APIs
- Samples of POST and PUT call for Custom Risk Provider

Custom Risk Provider

As an admin, you can use Custom Risk Provider APIs to integrate with two different risk engines at distinct levels for risk analysis. To integrate a risk engine at the initial transaction stage, configure the "***riskProviderPrePrimaryAuth***" attribute. For the second stage, you can integrate another risk engine by configuring the "***riskProviderPrimaryAuthSuccess***" attribute, which applies when the primary authentication obligation has been successful.

This attribute provides the risk advice which results in one of the following advice:

- ALLOW - The response sends the status as AUTH_ALLOWED, which ignores the remaining obligations and allows the user to continue the transaction.
- DENY - The response sends the status as AUTH_DENIED, which means the user is denied of any further transactions.
- STEPUP - Continues the authentication flow with the remaining obligations that are defined in the policy

Configure the following global setting in the solution to enable custom provider functionality:

- riskproviderprimaryauthsuccess

  NOTE
  If "riskProviderPrimaryAuthSuccess" is set to "none", the Custom Risk Engine evaluation is disabled.

To know more about these settings and the PATCH call to update these settings, see Managing Settings. To integrate your external risk system with the solution, see the instructions given in Integrating with External Risk Provider.

Sample payload of the settings that you must configure in IDSP:

```
[
  {
      "name": "riskProviderPrimaryAuthSuccess",
      "value": "none",
      "origin": "global"
  },

  {
      "name": "riskProviderPrePrimaryAuth",
      "value": "IARISK",
      "origin": "global"
  }
]
```

Custom Risk Provider APIs

The following table lists out the Custom Risk Provider endpoint URLs that are available for you to invoke and create, retrieve, update, or delete custom providers from the solution. You can perform CRUD operations on these APIs. For sample request and response payloads of all these URLs, refer to Risk API Swagger documentation.

As an admin, you can consume the Risk APIs to create, retrieve, update, and delete risk rules.

Operation Description	Operation Type	Endpoint URLs
Creating a Custom Provider	POST	https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders
Retrieving Custom Provider	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders/{{providerId}}
Updating Custom Providers	PUT	https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders/{{providerId}}
Retrieving all Custom Providers	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders
Deleting a specific Custom Provider	DELETE	https://${SSP_FQDN}/{{tenantName}}/admin/v1/CustomProviders/{{providerId}}

NOTE

- To know the value of ${IDSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/CustomProviders`
  should look similar to:
  `https://example.com/default/admin/v1/CustomProviders`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Samples of POST and PUT call for Custom Risk Provider are shown here

The following sample shows the request of a Custom Risk Provider POST call:

```
{
"providerName": "customRiskProvider",
"version": "1.0",
"apiKey": "6fd06fd82aceb78b0bbbe3990b865f96ea",
"url": "https://4afc-8289-b1e8817a3fec.mock.pstmn.io",
"providerHeaders": [
  {
    "name": "clientId",
    "value": "8289-b1e8817a3fec"
  }
],
"providerProperties": [
  {
    "name": "app",
    "value": "test"
  },
  {
    "name": "region",
    "value": "us-west"
  }
```

```
]
}
```

Where,

- providerName- Specify the name of the risk provider to be used.
- version- Specify if any particular version of the risk provider must be picked.
- apiKey- This is a unique passphrase or key that is used by the Custom Provider to identify the calling application.
- url- This is the endpoint that is called where the risk provider is being hosted.
- providerHeaders- These are key value pairs set as a part of the header in the endpoint call.
- providerProperties- These are key value pairs that contain additional information that is required by the provider to perform its functions.

The following sample shows the response of a Custom Risk Provider POST call. The response returns a providerId that must be used in the subsequent API calls to perform GET/PUT and DELETE operations.

```
{
  "providerName": "customRiskProvider",
  "version": "1.0",
  "apiKey": "2JYQ4uvEBk2ahpcvbnk7PyEfDr4ybWji06m2gg5TcBtJ0z3yJoRGMaAzjHVUUta8",
  "url": "https://4afc-8289-b1e8817a3fec.mock.pstmn.io",
  "providerHeaders": [
      {
          "name": "clientId",
          "value": "8289-b1e8817a3fec"
      }
  ],
  "providerProperties": [
      {
          "name": "app",
          "value": "test"
      },
      {
          "name": "region",
          "value": "us-west"
      }
  ],
  "createdBy": {
      "principalType": "CLIENT",
      "principalId": "c3331698-f8ec-4396-ae2d-a794755c297d",
      "principalName": "systemclient"
  },
  "createdDateTime": "2021-03-11 12:33 PM UTC",
  "origin": "global",
  "providerId": "26b8b2bb-2e61-4a26-b220-8b777455538b"
}
```

The following sample shows the request of a Custom Risk Provider PATCH call:

```
{
  "version": "1.01",
  "apiKey": "6fd06fd82aceb78b0bbbe3990b865f96ea",
  "url": "https://b1e8817a3fec.mock.pstmn.io",
  "providerHeaders": [
      {
```

```
            "name": "clientId1",
            "value": "b1e8817a3fec"
        }
    ],
    "providerProperties": [
        {
            "name": "app",
            "value": "test2"
        },
        {
            "name": "region",
            "value": "east"
        },
        {
            "name": "location",
            "value": "hyderabad"
        }
    ]
}
```


The following sample shows the response of a Custom Risk Provider PATCH call:


```
{
    "providerName": "customRiskProvider",
    "version": "1.01",
    "apiKey": "2JYQ4uvEBk2ahpcvbnk7PyEfDr4ybWji06m2gg5TcBtJ0z3yJoRGMaAzjHVUUta8",
    "url": "https://b1e8817a3fec.mock.pstmn.io",
    "providerHeaders": [
        {
            "name": "clientId1",
            "value": "b1e8817a3fec"
        }
    ],
    "providerProperties": [
        {
            "name": "app",
            "value": "test2"
        },
        {
            "name": "location",
            "value": "hyderabad"
        },
        {
            "name": "region",
            "value": "east"
        }
    ],
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "c3331698-f8ec-4396-ae2d-a794755c297d",
        "principalName": "systemclient"
    },
    "updatedDateTime": "2021-03-11 12:51 PM UTC",
    "origin": "global",
```

```
  "providerId": "26b8b2bb-2e61-4a26-b220-8b777455538b"
}
```

Managing Credentials

Learn about managing user and admin credentials using Symantec Identity Security Platform APIs.

The credential API is used to manage credentials of type, SMS, Email, Push, and FIDO (Security Key & Biometrics). The solution provides two different sets of APIs to manage credentials based on these roles; User and Administrator.

- User Credentials: Users can manage all the credentials that they have registered in the solution through the User Credential APIs.
- Administrator Credentials: The administrator can manage their own credentials and the credentials that have been registered by the users in the solution through the Admin Credential APIs. Admin can only manage the user's credentials but cannot create new ones for them.

Both these roles can only perform the update, retrieve, and delete operations through the Credential APIs. To register new credentials, use the Self-Service APIs.

User Credential APIs

The following table displays the list of Credential APIs that a user can invoke to manage their registered credentials.

Operation Description	Operation Type	Endpoint URLs
Retrieving all Credentials	GET	https://${SSP_FQDN}/{{tenantName}}/factor/v1/MeCreds
Retrieving Individual Credentials	GET	For ***Email***: https://${SSP_FQDN}/{{tenantName}}/factor/v1/MeCreds/{{credId-email}} For ***SMS***: https://${SSP_FQDN}/{{templateName}}/factor/v1/MeCreds/{{credId-sms}} For ***TOTP***: https://${SSP_FQDN}/{{templateName}}/factor/v1/MeCreds/{{credId-totp}} For ***FIDO***: https://${SSP_FQDN}/{{templateName}}/factor/v1/MeCreds/{{credId-fido}}
Updating Credential	PUT	https://${SSP_FQDN}/{{templateName}}/factor/v1/MeCreds/{{credId}}
Deleting Credential	DELETE	For ***Email***: https://${SSP_FQDN}/{{templateName}}/factor/v1/MeCreds/{{credId-email}} For ***SMS***: https://${SSP_FQDN}/{{templateName}}/factor/v1/MeCreds/{{credId-sms}} For ***Push***: https://${SSP_FQDN}/{{templateName}}/factor/v1/MeCreds/{{credId-push}} For ***TOTP***: https://${SSP_FQDN}/{{templateName}}/factor/v1/MeCreds/{{credId-totp}} For ***FIDO***: https://${SSP_FQDN}/{{templateName}}/factor/v1/MeCreds/{{credId-fido}}

Admin Credential APIs

The following table displays the list of Credential APIs that an administrator can invoke to manage the registered credentials of registered users and themselves.

Operation Description	Operation Type	Endpoint URLs
Retrieving the user Id of the user name	GET	https://${SSP_FQDN}/{{tenantName}}/idstore/v1/Users/{{userName}}
Retrieving all Credentials of the user	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Creds/{{userId}}
Retrieving user Credential by specifying the user Id	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Creds/{{userId}}/{{credId}}
Updating the User Credential by specifying the user Id	PUT	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Creds/{{userId}}/{{credId}}
Delete the user credential by specifying the user Id	DELETE	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Creds/{{userId}}/{{credId}}

Example of PUT request payload for Credential API

With the PUT call, a user or an admin can update only two attributes, status and default.

- A user can update the status to ***Inactive*** and ***Active***.
- An admin can update the status to ***Inactive***, ***Active*** and ***Disable***.

```
{
    "status":"ACTIVE",
    "default":true
}
```

Where,

- status- specifies if the credential is active, inactive, or disabled.
- default- specifies if the credential is the default credential.

Example of GET request payload for Credential API

```
https://${SSP_FQDN}/{{tenantName}}/admin/v1/Creds/{{userId}}/{{credId}}
```

Response:

```
[
    {
        "credId": "124aa1f3-8c49-41f4-82ae-5e6ffd4b5a1d",
        "tenantId": "e505513e-fb7d-4982-9ca5-6bfd674eab25",
        "userId": "f9e5f392-e372-4dd6-9eb6-bb4956d23a27",
        "credType": "sms",
```

```
        "credValue": "xxxxxx7256",
        "status": "ACTIVE",
        "createdDatetime": "2025-12-02 02:40 PM UTC",
        "modifiedDatetime": "2025-12-02 02:40 PM UTC",
        "tag": "business",
        "origin": "LDAPConfig",
        "originType": "IDSTORE",
        "lastUsedDatetime": "2025-12-03 05:05 PM UTC",
        "default": false
    },
    {
        "credId": "224c0411-182a-4071-8fee-b9928c80efcc",
        "tenantId": "e505513e-fb7d-4982-9ca5-6bfd674eab25",
        "userId": "f9e5f392-e372-4dd6-9eb6-bb4956d23a27",
        "credType": "sms",
        "credValue": "xxxxxx0718",
        "status": "ACTIVE",
        "createdDatetime": "2025-12-03 04:02 AM UTC",
        "modifiedDatetime": "2025-12-03 04:02 AM UTC",
        "tag": "personal",
        "origin": "LDAPConfig",
        "originType": "IDSTORE",
        "lastUsedDatetime": "2025-10-02 03:10 PM UTC",
        "default": false
    },
    {
        "credId": "cad6def9-385e-4018-8bb2-9417bf52bddf",
        "tenantId": "e505513e-fb7d-4982-9ca5-6bfd674eab25",
        "userId": "f9e5f392-e372-4dd6-9eb6-bb4956d23a27",
        "credType": "email",
        "credValue": "te*******@gmail.com",
        "status": "ACTIVE",
        "createdDatetime": "2025-12-02 02:40 PM UTC",
        "modifiedDatetime": "2025-12-02 02:40 PM UTC",
        "tag": "personal",
        "origin": "LDAPConfig",
        "originType": "IDSTORE",
        "lastUsedDatetime": "2025-11-05 10:00 PM UTC",
        "default": false
    }
]
```


Reference Links to:

- For sample request and response payloads of all the URLs mentioned in the above table, refer to Credential API Swagger documentation
- Credential API Error Codes

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.

For example:
`https://{{host}}/{{tenantName}}/admin/v1/creds/{{userID}}`
should look similar to:
`https://example.com/default/admin/v1/creds/{{userID}}`

- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Monitoring Services

Health check API endpoint that monitors the health of all the services running in the solution.

In Symantec Identity Security Platform, each service has a health check API endpoint that monitors the health of all the services and returns the status of the services. The API endpoint performs various checks such as: free memory, max memory, no of processors available, total memory, used memory, etc. If the service is healthy, it returns status as "healthy"=true, and if the service is unhealthy, it returns status as, "healthy"=false, in the response payload.

The list of Service Health API attributes and their descriptions are given in the following table:

Attribute Name	Description
totalFreeMemory	Represents the amount of free memory in the Java virtual machine
noOfProcessorsAvailable	Represents the number of processors available for the Java virtual machine
totalMemory	Represents the total memory available for use by the Java virtual machine
maxMemory	Represents the maximum memory that the Java virtual machine attempts to use
freeMemory	Represents the total free memory available for use by the Java virtual machine
usedMemory	Represents the used up memory by the Java virtual machine

The following table displays the list of Service Health APIs that can be used to check the status of the services running in IDSP:

Operation Description	Operation Type	Endpoint URLs
Factor Service Health	GET	https://${SSP_MGMT_FQDN}/system/factor/v1/health?detailed=true
Admin Service Health	GET	https://${SSP_MGMT_FQDN}/system/admin/v1/health?detailed=true
IDStore Service Health	GET	https://${SSP_MGMT_FQDN}/system/identity/v1/health?detailed=true
AuthZ Service Health	GET	https://${SSP_MGMT_FQDN}/system/oauth2/v1/health?detailed=true
Auth Manager Service Health	GET	https://${SSP_MGMT_FQDN}/system/auth/v1/health?detailed=true
Risk Service Health	GET	https://${SSP_MGMT_FQDN}/system/iarisk/v1/health?detailed=true

Reference Links to:

NOTE

- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://${SSP_MGMT_FQDN}/<systemtenantName>/factor/v1/health?detailed=true`
  should look similar to:
  `https://mgmt-example.com/system/factor/v1/health?detailed=true`
- To know the value of ${SSP_MGMT_FQDN} and ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<systemTenantName>` is "system".
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

- For sample request and response payloads of all the URLs mentioned in the above table, refer to Service Health Swagger documentation

Managing Settings

The Managing Settings page comprises global settings and tenant configurations, with the list of settings/configurations under each category mentioned in their respective tables.

This topic includes the following sections:

- Global Settings
- Tenant Settings
- Tenant and Global Settings
  - Self-Service Console Tenant Settings

Global Settings

The Global settings enable granular control over the Symantec Identity Security Platform features that are made available across all the tenants in the solution. Global system settings are applied identically to all the tenants. These settings help system admins to configure system-wide configuration using the properties defined in this table available at the end of the page.

The following table lists the operations that a global system admin can perform in IDSP.

Operation Description	Operation Type	Endpoint URLs
Retrieve global settings	GET	https://${SSP_FQDN}/system/admin/v1/Settings
Update global settings	PATCH	https://${SSP_FQDN}/system/admin/v1/Settings

Example of a Global Settings PATCH request payload

```
[
{
"name": "emailProviderToBeUsed",
"value": "smtp"
},
{
"name": "logLevel",
"value": "INFO"
}
]
```

In the preceding patch call, you are required to pass the attribute's key values pair to update the "cacheRefreshIntervalMins" settings. The Patch call does not require the complete request payload to be updated.

Where,

- emailProviderToBeUsed- This property determines which email provider should be used to send the emails with one-time password.
- logLevel- specifies Error, Warning, and Info level details of the solution's services.

Tenant Settings

The tenant settings enable granular control over the application features that are made available to your organization. As a tenant administrator, you can search for tenant settings and update. The tenant setting properties are defined in the table available at the end of the page.

The following table lists the operations that you can perform with the tenant settings in IDSP:

Operation Description	Operation Type	Endpoint URLs
Retrieve tenant settings	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Configs
Update tenant settings	PATCH	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Configs

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/Configs`
  should look similar to:
  `https://example.com/default/admin/v1/Configs`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Example of a Tenant Settings PATCH request payload

```
{
"name": "allowedOrigins",
"value": "http://example.com,https://authui.broadcom.net,http://authui.broadcom.net:4200"
}
```

For sample request and response payloads of all these URLs, refer to Settings- Swagger documentation

Tenant and Global Settings table

This table lists the global system settings and tenant settings with their descriptions, scope, default, and possible values:

Property Name	Property Scope	Security Related	Property Description
accessTokenExpiryMaxMins	Global		This property governs the maximum val accessTokenExpirySecs; the system int and checks the max value for accessTo update. • This is a security related property. S

accessTokenExpirySecs	Global, Tenant		The number of seconds an access toke "**urn:iam:app:oauthAccessTokenExpi** the application takes precedence over t See Managing Applications .
adminConsoleIdpDiscoveryEnabled	Tenant		This property determines whether the Id evaluated for the AdminConsole applica
adminConsoleIdStoreToBeUsed	Tenant		This property represents a list of ID Stor disambiguating administrative users goi signing into the Admin Console.
adminConsoleRequireAccessControl	Tenant		This property determines if the post-autl evaluated for the Admin Console Applic
adminConsoleSsoMaxSessions	Tenant		This property determines the max numb the Admin Console.
adminConsoleSsoExpirySecs	Tenant		This property indicates the max duration session, in seconds. Admin Console SS AdminConsoleUserSession MITM prote
adminConsoleSsoIdleExpirySecs	Tenant		This property refers to the duration of in Sign-On (SSO) session in the Admin Co out or prompted to reauthenticate.
allowSystemOpsOnExternalIngress	Global		This property specifies for backward cor system APIs that require no tenancy ma external (customer-visible) ingress using This setting can only be changed from t • This is a security related property.
anonymousDcrEnabled	Global, Tenant		This property indicates whether Dynami without the access token (AT), as per R required to invoke DCR API **"/oauth2/v**
allowedOrigins	Tenant		A comma-separated list of applications' Supports the use of '*' to indicate to the by any origin. Supported formats are: http://, https:// & • This is a security related property.
allowNullCors	Tenant		Enable or disable this setting to control IDSP APIs.
assetlinkjson	Global, Tenant		The value of the “assetlinks” parameter element should have a namespace of e • If the namespace is “web” then the JS mandatory element • If the namespace is “android_app” the JSON elements package_name and sh For more information on this file, see as

autoEnrollCredentialSyncPolicy	Tenant		This property controls how the user cred synced into the IDSP database. Enter R REPLACE and MERGE have the same credentials of a user in the following sce • **When the credentials from a spec database, and an authentication f from the same origin**: In this case, the new credentials in the database. • **When an authentication flow cont available in the IDSP database**: In origin are stored in the database. However, REPLACE and MERGE differ user in the following scenario: • **When the credentials with a speci database, and the authentication from a different origin**: If REPLACI credential is replaced with the origin flow. If MERGE is selected, the exist
azCodeExpirySecs	Global, Tenant		The number of seconds an authorizatio • This is a security related property.
customClaimProviderToBeUsed	Tenant		This property determines which custom the claims. This property can also be configured at • Setting this property at the tenant le tenant. • Applying this property at the applica customClaimProviderToUse) overrid that particular application, when the See Managing Applications
defaultAuthFactors	Tenant		***When a user initiates the initial authe preferences are present in the agent*** • if the "defaultAuthFactors" property i factors list as per the authentication • if the "defaultAuthFactors" ***is config*** default factor configured at the spec associated action.
deviceCodeExpiryMaxMins	Global		This property governs the maximum val for deviceCodeExpirySecs and deviceC internally translates the given mins to se for deviceCodeExpirySecs /deviceCode update. • This is a security related property.
deviceCodeExpirySecs	Global, Tenant		This property represents the "expires_in the response of POST /oauth2/v1/devic before the device_code and user_code • This is a security related property.

deviceCodePollingIntervalSecs	Global, Tenant		This property represents the value "inter of the response of POST /oauth2/v1/dev seconds the client should wait between • This is a security related property.
disableIdentityAssertionSignatureVerification	Tenant		This property is used to disable signatur Bearer flow and id_token_hint in the Au troubleshooting purposes and is not rec environment.
emailPhoneVerificationTokenExpiryMins	Tenant		The number of minutes an email/phone
emailprovidertobeused	Global		This property determines which email pr with a one-time password.
enableFIDOMetadataAttestation	Tenant	yes	In case of **FIDO Registration**, When "**enableMetadataAttestation**" pr **FIDORegChallengeGenerator** API resp "**direct**". When this setting is enabled, th sends attestation response to the serve platform. In case of hardware authentica attestation data sent to server while reg When "**enableMetadataAttestation**" pr **v1/FIDORegChallengeGenerator** API as"**none**".
enableReverseLookupInTDI	Global, Tenant		This property enables the reverse looku to identify the trusted device based on th where the device tag is not passed in th device-tag cookie from the end-user dev
encryptionAlgorithm	Global (Tenant can override the setting)		This property displays the standard enc the solution. • RSA-OAEP • RSA-OAEP-256
encryptionMethod	Global (Tenant can override the setting)		This property displays the encryption m • A128CBC-HS256 • A192CBC-HS384 • A128GCM • A192GCM • A256GCM
endUserNotificationDeliveryMode	Tenant		Sets how the Admin Console sends not conjunction with notifyEventsConfig pro

errorUrl	Tenant		This property represents a URL to redire The following redirection behavior occu 1. If an app-level Error URL is not prov SAML), the error redirection occurs 2. If the application intends to redirect following conditions must be met: a) The application explicitly specifie b) If you do not want to specify an a app to use a global error page. This application metadata, urn:app:useG In this case: – If a tenant-level Error URL is pro – If no tenant-level Error URL is pr built-in /signin/v1/error URL.
expireTransactionalDataEnabled	Global		When this settings is enabled, database byproduct of an authentication flow get data artifacts are removed by the sched away at the end of the authentication flo • This approach enables active/active multiple data centers. This helps in having a single authoritative source
fidoIdentifierExpiryDays	Tenant		This property represents the allowed ex saved as a cookie at the client side.
fieldsToBeEncryptedInAuthFlow	Global/Tenant/ Branding Settings		• This property lists the fields that mus • As part of the branding setting, it hel must be encrypted.
gracePeriodToRetainPreviousSigningKeyDays	Tenant		This property that defines how many da token verification after a key update, en
hideAuthenticationFailureReason	Tenant		This property decides whether to disclos invalid user/locked user tries to start the the error message will not provide any i simulate the PASSWORD obligation. In the case of Authorization Code flow, t allowing IDSP to redirect to a redirect U authentication failure.
identityVerificationTokenExpiryMins	Tenant		This property specifies the number of m
identitySourceIdRequired	Tenant		The initial out of the box assumption is t possible identity source namespaces. When using multiple identity sources (Id expecting that the userLoginId values m **'true'**. This mandates the client to provid that consume it (Risk Eval APIs, Passw passwords).

idEventHookConfig	Global, Tenant		This property defines audit event hooks The currently supported hooks are: • factor.me.auth.lock.success • factor.me.register.complete.success • internal.user.sync.initiate • session.revoke.success For information on auditing events logge
idpDiscoveryEnabled	Tenant		This property determines whether the Id evaluation or not. • If configured as "***true***" then IDP Disc • If configured as "***false***" then IDP Dis place. However, the "idpDiscoveryEnabled" co takes precedence over the configuration Applications
idStoreToBeUsed	Tenant		There are two kinds of identity stores su Each tenant may have any number of a is described by a configuration object. T list of ID store GUIDs that make up the identities during authentication. The app default disambiguation search order by maintained by the application property, '
idTokenExpiryMaxMins	Global		This property governs the maximum val the system internally translates the give for idTokenExpirySecs at the time of pro
idTokenExpirySecs	Global, Tenant		The number of seconds an ID token is v However, the "**urn:iam:app:oauthIdent** configuration made in the application tal the tenant settings. See Managing Appl
isInlineEnrollmentAllowed	Global, Tenant		Inline registration of user credentials is a to true. When the policy property, "inlineEnrollm it overrides this tenant-level "isInlineEnr Property.
inlineEnrollmentFactorsRequired	Global, Tenant		This property specifies the mandatory a Users can proceed with the inline regist themselves with the mandatory authenti property. If the inline mandatory authentication fa authentication is denied (AUTH_DENIE
isInlineOTPEnrollmentToBePersisted	Global, Tenant		User credentials that are enrolled using when this property is set to 'true'. When through inline enrollment are deleted.
isAutoEnrollmentAllowed	Global, Tenant		Auto-enrollment of IDSP credentials hap ZFP flow, along with this flag, auto-enro

isAdminScopeEnforced	Global, Tenant		• This property controls whether resou constrained versus unconstrained a resource authorization, admin has to Global level to apply across all the te • This is a security related property.
isIdStoreAuthoritativeForIDLCM	Tenant		This property is to check what authorita or IDSP. If this value is set to false, ther source if the strike count reaches the m
isRememberMeEnabled	Tenant		This property represents whether to ena for the tenant.
isRiskInactiveUsersEventsCleanupEnabled	Global		This property is used to enable or disab inactive users.
isRiskCleanupEnabled	Global		This property is used to enable or disab activity.
isRiskServiceEnabled	Global, Tenant		This property is used to disable risk eva be disabled at each tenant level. When this property is globally disabled, level.
isTrustedDeviceEnabled	Tenant		This property represents whether to ena for the tenant.
ivrProviderToBeUsed	Global, Tenant		This property determines which IVR pro with a one-time password. For tenant level, this property can have providers at tenant & all the custom pro At the global level, this property can hav providers at the global level.
keepOldRefreshTokenOnRefreshTokenFlowWithOfflineAccess	Tenant		When this setting is disabled, the refres the refresh token flow, and a new refres a behavior change from IDSP v2.1 relea token behavior, where the refresh token and the same refresh token is returned should be enabled.
keepOldSessionIdOnNewRefreshTokenFlowWithOfflineAccess	Tenant		When this setting is enabled, if a client (AT) and Refresh Token (RT) using the `scope='offline_access'`, the as the existing IT.
localeStrategy	Tenant		This property defines the strategy to det The possible values for the localeStrate If the localStrategy is set to browser, the locale for the SignIn UI. If the localStrategy is set to tenantLocal localeToBeUsed property is used for the
localeToBeUsed	Tenant		This property specifies the locale to be u following scenarios: • When the localeStrategy is set to ten configured in the localeToBeUsed pr • When the browser locale or the loca query (ui_locales) does not exist in t the SignIn UI display is the locale th property.

logLevel	Global		This property helps to set the log level f change is made to this property value, it interval. Configuring log levels of IDSP services • If set to INFO-Error, Warning, and In • If set to Debug-Detailed debug-relat • If set to Trace- Very detailed trace o diagnose problems.
logRequestPayloads	Global		This property enables to decide when to
maxAdminResourcesReturnedByGetAll	Tenant		This property limits the maximum results value is configured then a default value param "`countonly` " should be enabl the API.
maintainSelfServiceCreds	Tenant		This parameter controls if the origin of th or inline enrollment) can be changed or trusted source. It has 2 possible values - NEVER: the origin can be changed from ALWAYS: the origin of the self-service c
maxBiometricDevicesPerUser	Tenant		This property represents the maximum allowed per user.

maxGroupsInJWT	Global, Tenant		This property controls the maximum nur
			to a token in the following scenarios:
			• The client requests groups via useri
			scope.
			• Client requests groups in Access To
			groups" / "urn:iam:at:groups" scopes
			`"allowedOperations": [`
			`"itgroups",`
			`"atgroups"`
			`],`
			• When an Identity Token is acquired
			auth/v1/authenticate), an authorized
			request groups in the resulting Ident
			`"authContext": [`
			`...`
			`{`
			`"key": "requestGroupsInIT",`
			`"value": "true"`
			`},`
			`...`

maxTrustedDevicesPerUser	Tenant		This property represents the maximum in the IDSP database.

mitmProtectionLevel	Tenant		This property specifies whether a factor the flowstate attribute in an authenticati (TDI) in an HTTP cookie, to mitigate a n The following are the possible value: • none: Specifies that the feature is di • user-ip: Stores the IP address of the attribute within an authenticate respo the IP address of the user in the fact that is stored in the flowstate attribut values match. • user-agent: **Using flowstate**: Stores the user ag flowstate attribute within an authenti received, the user agent in the facto that is stored in the flowstate attribut values match. **Using TDI**: If TDI is present as an H request, the user agent in the reque TDI. If the value in TDI does not mat TDI is ignored and the user is challe • user-ip, user-agent: **Using flowstate**: Stores the IP addr user in the flowstate attribute within request is received, the IP address a request are compared to the corresp flowstate attribute. The request is pr **Using TDI (for user-agent)**: If TDI i initial authentication request, the use the user agent in the TDI. If the valu authentication request, TDI is ignore
motpDefaultEnrollmentMode	Tenant		To enhance tenant customization of the experience, IDSP now offers this setting displayed in the Signin UI. The enrollme value specified in tenant setting.
notifyEventsConfig	Tenant		Sets whether the Admin Service Consol actions, including: • Account activation on a new device • Account lock or unlock (AccountLoc • Password reset (passwordChange) • Credential activity, such as credentia (credentialActivity) • End user email changed (userEmail • End user account deleted (userAcco If this property is set, you should also co in the endUserNotificationDeliveryMode
otpLength	Global, Tenant		This property represents the length of th
otpLookUpCounterSeconds	Global, Tenant		This property represents the time period and after the current timestamp to creat for authentication is from this window, it
otpValidityPeriodSeconds	Global, Tenant		This property represents the validity per

passkeyAuthenticationHints	Tenant	Yes	In case of passkey authentication, if "**pa** • **client-device** : passkey should be a device platform (iCloud Keychain or • **security-key**: passkey should be au connected to authenticating device • **"" or null** : It may give user a choice authenticated. The final user experience of passkey au and browser combination (if given confi on the given platform). In case of combi matters.
passkeyAuthenticationTransports	Tenant	Yes	In case of passkey authentication, if "**pa** set to • **internal** : passkey should be authe device platform (iCloud Keychain or • **usb, ble, nfc**: passkey should be au security key connected to the authe NFC transport type • **hybrid, null or ""**: It may give user a be authenticated. The final user experience of passkey au and browser combination (if given confi on the given platform). In case of combi matters.
passkeyRegistrationAuthenticatorAttachment	Tenant	Yes	In case of passkey registration, if "**passkeyRegistrationAuthenticatorAt** • **platform** : passkey should be regis (iCloud Keychain or Google Passwo for authentication on other devices c • **cross-platform**: passkey registratio hardware security key (roaming dev • **"" or null** : It may give user a choice registered. The final user experience of the passke the other configuration setting (passkey experience of passkey registration may combination (if given configuration value platform.)
passkeyRegistrationHints	Tenant	Yes	In case of passkey registration, if "**pass** • **client-device** : passkey should be r platform (iCloud Keychain or Google platform for authentication on other • **security-key**: passkey registration hardware security key (roaming dev • **"" or null** : It may give user a choice registered. The final user experience of passkey re browser combination (if given configurat on the given platform). In case of combi matters.

passwordCountToBeStoredInHistory	Tenant		This property defines the password hist IDSP database. Password history defin must be associated with a user account password update succeeds only when t password history match.
passwordProviderToBeUsed	Global, Tenant		This property determines which provide
passwordResetTokenExpiryMins	Tenant		This property governs the expiry duratio password reset.
persistAdditionalContextInRiskEvent	Tenant	yes	This property controls whether additiona should be stored in the Symantec Ident this setting results in consumption of ad should be sized appropriately before en
persistEnrolledFactorsinIDP	Global, Tenant		This property governs the functionality t or phone number to IDP attributes.
persistentStoreForTransactionalData	Global		This property specifies the store where Authorization Code, Device Code, and t
persistUserConsentGivenForResourceApps	Tenant		This setting indicates if the consent give for the resource apps needs to be persi
preserveOriginInAutoEnrollment	Tenant		This property ignores or considers the o Synchronization flow depending on the • false: Ignores the origin in the Crede existing credential with the new cred • true: Considers the origin in a Crede an existing credential with the new c only when the origin and credential t existing credential is replaced only w
principalConditionRequiredInPolicy	Global, Tenant		This property determines whether an en an Authentication policy. • true: The POST/PUT requests for Au inclusion of the "principal" condition • false: Existing policies without the pr If an existing authentication policy in applications in the "apps" section, th authentication policy is mandatory.
PrivacyURL	Tenant		This property indicates the Privacy URL signin app. This URL redirects users to privacy practices, policies, and terms re company.
pushAuthTimeoutSecs	Global, Tenant		This property governs the timeout value expected to respond to push notification
pushEnforceLocalAuth	Global, Tenant		This property is used to enforce the use authentication (LocalAuth) which is enal VIP PUSH app. For example-fingerprint
pushIncludeDeviceInfo	Global, Tenant		This property is used to include device i app.

pushNumberChallengeEnabled	Tenant	No	This property manages the "Push Numb Policy" through IDSP: • When this property is set to **true**, the and expects a push number challeng Authentication. • When this property is set to **false**, th allowing users to simply **Approve/D** challenge during Push Registration To update the value of this setting, you IDSP.
pushProviderToBeUsed	Global, Tenant		This property determines which push pr to the end user's device. At the tenant level, this property can hav providers at the tenant & all the custom At the global level, this property can hav providers at the global level and default vip).
pushRegTimeoutSecs	Global, Tenant		This property governs the registration ti
refreshTokenExpiryDays	Global, Tenant		This property governs the refresh Token number of days a refresh token is valid. However, the "**urn:iam:app:oauthRefre** configuration made in the application tal the tenant settings. See Managing Appl
requireAccessControl	Tenant		Determines if the post-authentication ac This property can also be configured at name-"requireAccessControl"). Applying the configuration at the tenant level for t also covered in Managing Applications .
riskRetentionDays	Global		This property indicates how many days tables, before the clean-up activity takes
returnCredentialsInFactorSelection	Tenant		When multiple obligations are configure level, the **nextaction** attribute is set to ' By default, a user is prompted to select Based on the selected obligation, the lis for the user to select for authentication. To display the list of registered user cre obligations itself, set the **returnCredent** **true**.
riskProviderPrePrimaryAuth	Global		This property defines the risk provider in has been completed. The risk score tha authentication policy is used by the solu obligations.

riskProviderPrimaryAuthSuccess	Global		This property defines the risk provider ir successful. Based on the risk advice ret decides the following: • riskAdvice>Allow → The solution ski completes the auth flow, and marks • riskAdvice>Deny → The solution ski completes the auth flow, and mark it • riskAdvice>stepup → The solution c remaining authentication factors
serviceAccountSsoEnabled	Tenant		This property determines whether accou processes (non-human) can open SSO
serviceAccountSsoMaxSessions	Tenant		This property determines the max numb the Admin Console.
serviceAccountSsoExpirySecs	Tenant		This property indicates the max duration session, in seconds. Admin Console SS AdminConsoleUserSession MITM prote
serviceProviderFqdn	Global		This is the Host FQDN specified during "ssp.ingress.host" helm parameter.
signAlgo	Global		This property governs the signing algori issued.
SigninFlowExpiryMins	Tenant		This property defines the number of min In partial delegation, if a third party configures "***urn:iam:login_flow:expire*** Hint (ITH), it overrides the value of this s
skipQRCodeForSecurityKeyAuthentication	Tenant	Yes	• In case of **Registration**, if this prope a **security key** as their preferred me the **QR code scanning** option. How instead during registration, the proce authentication ultimately fails. • In case of **Authentication**, if this pro code scan and waits for the security canceled, authentication fails.
skipFactorLevelForMatchingAmr	Global, Tenant		This property handles obligation remova • When this property is set to true, pol level when at least one obligation wi a completed obligation (this is simila AMR in the incoming Identity Token) • When this property is set to false, po obligation from a higher level, but no For example, consider an authentication FIDO: 1, SMSOTP: 2, FIDO: 2. When th sufficient to satisfy the policy, else, PAS

skipHideAuthenticationFailureReasonForLockedUser	Tenant		This property determines whether to trig locked user when the tenant property “h set to true. When "skipHideAuthenticatio is set to true, the actual error message i user and the PASSWORD obligation is “hideAuthenticationFailureReason” is se
skipValidationInAutoEnrollment	Tenant		This property decides whether to validat phone, e-mail after user disambiguation auto-enrollment and credential sync.
smsProviderToBeUsed	Global, Tenant		This property determines which SMS pr with a one-time password. For the tenant level, this property can ha providers & all the custom and can SMS For the global level, this property can ha providers. The default SMS provider ava
ssoExpirySecs	Tenant		When Single Sign On (SSO) is enabled the SSO session, in seconds.
ssoIdPropTokenExpirySeconds	Tenant		This property determines how long an ID considered only when the client is config and the same scope is requested in the the jwt-bearer flows.
ssoMaxSessions	Tenant		This property determines the max numb applications.
ssoMode	Tenant		If this property is "enabled", applications Sign On (SSO). This is tracked by "sess are connected through the OpenID Prot API. If the property is set to "enabledCookie" the web protocol such as (OpenID and S cookie named, "sspsession". With SSO enabled, applications can log using the following approaches: • SAML app logging out via a redirect • OpenID app logging out via a redire • OpenID app invoking /oauth2/v1/rev carrying "sid" session id claim • Any app invoking /admin/v1/revoke
successiveSigningKeyUpdateIntervalMins	Global		This property mitigates successive calls defined time interval.
tenantBackgroundImage	Tenant		This is the Base 64 data format of the im the SignIn user interface.

tenantCustomBODYContent	Tenant	No	This property accepts base64 represent
			following format:
			`data:text/`
			`html; base64,SGVsbG8gV29ybGQhDQo8`
			It allows to embed custom HTML Conte

tenantCustomCSS	Tenant		This property enables the customization The CSS file that you upload must be B updating any of the css settings.

tenantCustomHEADContent	Tenant	No	This property accepts base64 represent
			following format:
			`data:text/`
			`html; base64,SGVsbG8gV29ybGQhDQo8`
			It allows to embed custom HTML Conte

tenantFavicon	Tenant		This property can be used to update the image can be either a PNG, GIF, or ICO
tenantLogo	Tenant		This property is used to update the tena data format.
trackAccessToken	Global, Tenant		This property is used to revoke the Acce the provided access token has been rev When enabled, IDSP tracks the revoked "RevokedToken" object containing this j
trackIdentityToken	Global, Tenant		This property is used to revoke Identity whether the provided Identity Token or I When enabled, IDSP tracks the revoked "RevokedToken" object containing this j
trackLastLogin	Tenant		This property is to track the last login su authentication attempt. When this setting is enabled, the systen for all those user credentials that are us only successful logins attempts to provi attribute is captured in the response of t
transactionalDataReadsRetryCount	Global		This property indicates the number of re
transactionalDataReadsRetryWaitPeriodMillis	Global		This property specifies the waiting perio attempts for reading the transactional da
trustedDeviceExpiryDays	Tenant		This property defines the maximum dura remains stored in the IDSP database. It day for finer control, such as specifying 0.33).
useGranularAMRForOTP	Tenant	No	This property enhances the OTP Authen adding EMAIL and SMS as specific AM This offers more granular control on the match for either SMS or EMAIL. Upon successful completion, the final A includes the specific method (EMAIL or ["EMAIL", "OTP"] or ["SMS", "OTP"]). For more information on how this setting with "skipFactorLevelForMatchingAmr"
useOriginalIdentityTokenExpiry	Global, Tenant		This property is used in re-authenticatio Identity Token expiry value.

userAutoUnlockPeriodMinutes	Tenant		This property allows administrators to se user. The locked user will be unlocked a If the ID Store attribute,"isAuthoritativeF "userAutoUnlockPeriodMinutes" propert
userMaxStrikeCount	Tenant		This property lets administrators define attempts are allowed before a user acco
vipURL	Global		This property is used to configure the U

Self-Service Console Tenant Settings table

This table lists the Self-Service Console related tenant settings with their descriptions, scope, default, and possible values:

Property Name	Property Scope	Security Related	Property Description
appSpecificNotificationExpiryDaysForSSC	Tenant		This property controls how many days a sent to users assigned or granted acces If the notification is generated by a Notif not provide an explicit `expiryDate` the `startDate` value.
endUserNotificationDeliveryMode	Tenant		This property indicates which delivery m
factorsAllowedForEnrollmentInSSC	Tenant		This property determines the factors tha Console.
landingPageInSSC	Tenant		This property determines the post-login based on the configured value. It contro standard Navigation Tile View (dashboa after signing in.
leftNavigationAllowedInSSC			This property determines the navigation set to **true**, navigation is displayed using navigation is presented through the top
maxAllowedAppTileGroupsForSSC	Tenant		This property controls the maximum nur displayed on the Self-Service Console a
maxAllowedNotificationSize	Tenant		This property controls the maximum nur of a Self-Service Console notification ca
privacyUrlForSSC	Tenant		This property is used by the Self-Servic URLs in Footer, and About Page, etc.

sectionToBeHiddenInSSC	Tenant		This property controls the visibility of sp Service Console based on the configure **Example**: If **TRUSTED_DEVICES** is co disabled in the Self Service Console alo
selfServiceConsoleIdStoreToBeUsed	Tenant		This property represents a list of ID Sto disambiguating self-Service Console us when signing into the Self-Service Cons
supportDocumentUrlForSSC	Tenant		This property is used by the Self-Servic Documentation URLs in Footer, About F
tenantCustomBODYContentForSSC	Tenant		
tenantCustomCSSForSSC	Tenant		This property can be used to update the user interface. The CSS must be Base6
tenantCustomHEADContentForSSC	Tenant		
tenantFaviconForSSC	Tenant		This property can be used to update the user interface. The image can be either Base64 encoded.
termsAndConditionUrlForSSC	Tenant		This property is used by the Self-Servic Conditions URLs in Footer, and About F
toAllUsersNotificationExpiryDaysForSSC	Tenant		This property controls how many days a sent to all users that it expires. If the notification is generated by a Notif not provide an explicit `expiryDate` the `startDate` value.

Reference Links:

- Settings- Swagger documentation
- Settings API Error Codes

Managing IDSP User Operations

Use the Symantec Identity Security Platform User Operations APIs to search for users, manage user lock status, and delete users within the internal database.

The following sections detail the specific APIs and their operations:

- User Search API
- User Locker API
- User Delete API

User Search API (GET)

Use the following admin API to perform a search for users in the IDSP database (Internal DB):

```
GET https://${SSP_FQDN}/{{tenantName}}/admin/v1/Users/{{userLoginid}}
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/Users/{{userLoginid}}`
  should look similar to:
  `https://example.com/default/admin/v1/Users/{{userLoginid}}`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

This API returns user details like userId, firstname, lastname, status for users existing in the IDSP database. Users that are existing in ID Stores or coming from external applications into IDSP for authentication using ZFP or OIDC are created into the IDSP database. Such users can be searched using their loginId.

User Search API Error Codes

The following table provides information about the error codes that may be returned by the IDSP User Search API.

Table 115: Error Codes for User Search APIs

Error Codes	Error Messages
0000048	Invalid UserLoginId
0000049	Missing UserLoginI

UserLocker API

Use the User Locker API to perform **POST** and **GET** operations in IDSP database (Internal DB).

User Locker GET Operation:

- Use the following admin API to retrieve the lock/unlock status of a user from the IDSP database:
  ```
  GET https://${SSP_FQDN}/{{tenantName}}/admin/v1/UserLocker/{{userLoginid}}
  ```
  GET response sample of a user who is unlocked:
  ```
  {
    "locked": false
  }
  ```

User Locker POST Operation:

- Use the following admin API to configure the lock/unlock status of a user in the IDSP database:
  ```
  POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/UserLocker/{{userLoginid}}
  ```
  Sample of a Locked User POST request:
  ```
  {
    "locked": true
  }
  ```
  Sample of an Unlocked User POST request:
  ```
  {
    "locked": false
  }
  ```

User Delete API (DELETE)

Use the following admin API to delete a user from the IDSP database (Internal DB):

```
DELETE https://${SSP_FQDN}/{{tenantName}}/admin/v1/Users/{{userId}}
```

This API deletes an existing user in the IDSP database using the User Id.

When a user is deleted, the API also deletes the corresponding user information from FIDO devices, access tokens, trusted devices, risk devices, risk device tags, risk events, and so on. Depending on the deletion status, the API displays an appropriate status message in the database.

The API invokes the following audit events:

- admin.user.delete.success
- admin.user.delete.failure

Reference Links:

- The Internal Identity Store Configuration API.

Managing VIP Configuration Settings

Configure VIP settings using the VIP Configuration APIs to integrate Symantec VIP with Symantec Identity Security Platform for Mobile OTP and Push authentication using the VIP Access app.

Tenant and system administrators use the VIP Configuration APIs to establish the necessary settings for integrating Symantec VIP with IDSP. These configurations are required to authorize individual users to perform Mobile OTP or Push authentication using the VIP Access application.

The following table lists the operations that a tenant or system admin can perform in IDSP.

Operation Description	Operation Type	Property Scope	Endpoint URLs
Add/Update VIP configurations	POST	Tenant/System	https://${SSP_FQDN}/{{tenantName}}/admin/v1/VIPConfiguration
Retrieve VIP configurations	GET	Tenant/System	https://${SSP_FQDN}/{{tenantName}}/admin/v1/VIPConfiguration
Delete VIP configurations	DELETE	Tenant	https://${SSP_FQDN}/{{tenantName}}/admin/v1/VIPConfiguration
Configuring VIP connection URL	POST	System	https://${SSP_FQDN}/{{tenantName}}/admin/v1/Settings

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/VIPConfiguration`
  should look similar to:
  `https://example.com/default/admin/v1/VIPConfiguration`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Obtaining your VIP certificate

You need a certificate for client authentication to secure communications and identify yourself to the Symantec VIP. In communications with the Symantec VIP, the VIP certificate is used as a TLS/SSL client certificate. You can obtain a VIP certificate from the VIP Manager.

Follow these steps to obtain a VIP certificate:

1. Sign in to VIP Manager (https://manager.vip.symantec.com). You need your credential.

   NOTE
   The credentials for the VIP Manager are provided by Broadcom.
2. From the dashboard, select Account in the navigation bar at the top of the page.
3. Select the Manage VIP Certificates link on the right side of the page.
4. Select Request a Certificate. Review the certificate instructions and select Continue.
5. Enter a name for your certificate that is easy to remember. Do not generate a CSR.
6. Select Submit Request.
   Select the certificate format as PKCS#12 (file extension as .p12).
7. Enter a password to protect access to this certificate. The password must be at least eight characters and must include one uppercase and one lowercase letter, plus one number.
   Do not lose this password. You need it to install the certificate.
8. Click Download Certificate. You are prompted to save the file to your local system. You can return to this page at any time to download this certificate again.
9. Encode the .p12 file as base64 by executing this command:

   ```
   openssl base64 -in vip_cert.p12 -out vip_cert.p12.base64 -A
   ```

   NOTE
   If you do not have openssl, you can use any one of the following commands based on the operating system you are using:

   - For Linux:

     ```
     base64 -w 0 vip_cert.p12 > vip_cert.p12.base64
     ```
   - For Mac:

     ```
     base64  vip_cert.p12 > vip_cert.p12.base64
     ```
   - For Windows:
     - powershell:

       ```
       [System.Convert]::ToBase64String([IO.File]::ReadAllBytes("c:\path\vip_cert.p12")) > .
       \vip_cert.p12.base64
       ```
     - certutil:

       ```
       certutil -encode vip_cert.p12 tmp.b64 && findstr /v /c:- tmp.b64 > vip_cert.p12.base64
       ```

Adding VIP Configuration setting

NOTE

- To perform patch operation on the VIP Configuration setting, see the patch example that is given in the Settings page.
- Replace the text {{text}} including {{}} characters with appropriate values in the following endpoint.

Follow this step to add VIP Configuration setting:

1. Perform a "POST" operation using the following VIP Configuration endpoint:

   ```
   https://${SSP_FQDN}/{{tenantName}}/admin/v1/VIPConfiguration
   ```

   This is a sample request payload to add the VIP Configuration setting:

   ```
   {
   ```

```
  "vipCertData": "<base64 encoded p12 certificate data>",
  "vipCertPwd" : "p12 certificate password"
}
```

where,

- vipCertData- This is a Base64 encoded p12 certificate that is obtained from the VIP Manager.
- vipCertPwd- The password of the certificate which is obtained from the VIP Manager.

VIP Config endpoint returns the following sample response payload:

```
{
  "vipAccountName": "SYMANTEC",
  "vipCertExpiryDate": "25 Oct 2022"
}
```

Retrieving VIP Configuration setting

Follow this step to retrieve VIP Configuration setting:

1. Perform a "GET" operation using the following VIP Configuration endpoint:

   NOTE
   Replace the text {{text}} including {{}} characters with appropriate values in all the endpoint.

```
https://${SSP_FQDN}/{{tenantName}}/admin/v1/VIPConfiguration
```

VIP Config endpoint returns the following sample response payload:

```
{
  "vipAccountName": "SYMANTEC",
  "vipCertExpiryDate": "25 Oct 2022"
}
```

Reference Links to:

- VIP Configuration Settings- Swagger documentation
- VIP Integration API Error Codes

Managing Message Templates

Learn about the Symantec Identity Security Platform Message Templates API.

Message Templates are structured emails, SMS, or Push messages that are sent to multiple recipients communicating about their status during different phases of authentication. Provides the flexibility to create common ready-to-use message templates that can be shared among multiple organizations and thereby applied to multiple users.

These templates can be customized for different organizational needs and in different languages. IDSP provides out-of-the-box message templates for the following languages:

- en_US → English (United States)
- pt_BR → Portuguese (Brazil)
- es_ES → Spanish (Castilian), Spanish (Spain)
- fr_FR → French (France)
- ja_JP → Japanese (Japan)

The templates are created based on user actions that are performed during the authentication process, such as, login, New User Welcome, Password reset, registration and verification of emails, phone number and many more. The supported message types are SMS, Email Subject, Email Body, Push, Push Message Title, Push Message Profile, and Push Message Text, and IVR.

This topic includes the following sections:

- Message APIs
- Message API Sample Payload
- Using Message Templates in the Factors (OTP and Push)
- End-user Notification Templates
- References

Message APIs

You can use the Message API endpoints to create and manage the templates.

The following table lists out the Message Template Endpoint URLs available to an administrator for the management of these templates:

Operation Description	Operation Type	Endpoint URLs
Adds a new message in the application	POST	https://${SSP_FQDN}/<tenantName>/admin/v1/Messages
Retrieves the particular message based on the specified message ID.	GET	https://${SSP_FQDN}/<tenantName>/admin/v1/Messages/<messageId>
Retrieves all the messages that are registered with the application.	GET	https://${SSP_FQDN}/<tenantName>/admin/v1/Messages
Allows to update only ONLY the "message" parameter in the message request payload by specifying the message ID. During the PUT operation, only the "message" field can be updated. In the response, the "message" and "updated_datetime" fields are updated, while all other fields remain unchanged.	PUT	https://${SSP_FQDN}/<tenantName>/admin/v1/Messages/<messageId>
Allows to delete the message entity by specifying the message ID. Deletion of OOTB messages is not allowed.	DELETE	https://${SSP_FQDN}/<tenantName>/admin/v1/Messages/<messageId>

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/Messages`
  should look similar to:
  `https://example.com/default/admin/v1/Messages`
- The Delete operation deletes the message template Id for languages other than English (if other languages exist).
- During the PUT operation, only field types, "message_template" and "updated_datetime" can be updated, all other fields cannot be updated.
- To know more about the scopes that are required to secure these APIs, refer to the "Scopes" section here.

Message API Sample Payload

Sample POST Request Payload for OTP

The following code snippet shows the request payload for the "Create Message" API.

```
{
```

```
  "action": "LOGIN",
  "messageType": "EMAIL.BODY",
  "message": "[OTP] is your One-Time Password to login. The OTP is valid upto [MINUTES] minutes.",
  "language": "en_US"
}
```

Where,

- **action** - specify the action for which the message is created. The supported actions are LOGIN, REGISTER, or custom action name.
- **messageType** - specify a message type based on the authentication factor. Supported message types are EMAIL.BODY/ EMAIL.SUBJECT/ SMS.
- **message** - add your message here. This field supports plain text and html formatted messages.

  NOTE

  - Include message validity information along with the text while adding messages for OTP and Email. In the above e time login password. The OTP is valid for up to 2 minutes.
  - Example of HTML formatted message for the "message" attribute:

    ```
    <html><body><div style='font-size: 14px'>Hi ${idProfile.user_loginid}, <span style='text-align:
     center; font-size: 16px; font-weight: 600;'>$$OTP$$</span> is your One-Time Password to login.
     The OTP is valid up to $$MINUTES$$ minutes. <br><br><b>Note: Do not share One Time Password
     with Anybody. For more information <a href='#'>click here</a></b></div><body></html>
    ```

    To know about the ${idProfile.user_loginid} substitutor, see Dynamic Environment Variable Substitution
- **language** - specify the language for the template in the correct format. The languages can be specified in the following format: for English - "en_US", for French - "fr_FR", for Japanese - "jp_JP" etc.

  NOTE

  - If you do not follow this format and send inputs like "en", you get an error message saying, "8800023=invalid language".
  - To create messages in multiple languages, create separate messages for each language.

Sample POST Request Payload for Push

The following code snippet shows the Push request payload for the "Create Message" API.

```
{
   "action": "REGISTER",
   "messageType": "PUSH.MESSAGE.TITLE",
   "message": "Push Authentication",
   "language": "jp_JP",
}
{
  "action": "REGISTER",
  "messageType": "PUSH.MESSAGE.PROFILE",
  "message": "https://symantec.com",
  "language": "en_US",
}
{
  "action": "REGISTER",
  "messageType": "PUSH.MESSAGE.TEXT",
  "message": "Please approve your transaction",
  "language": "fr_FR"
}
```

Where,

- **action** - specify the action for which the message is created. The supported actions are LOGIN, REGISTER, or provide a custom action name.
- **messageType** - specify a message type based on the authentication factor. Supported message types are, PUSH.MESSAGE.TITLE/ PUSH.MESSAGE.PROFILE/ PUSH.MESSAGE.TEXT
- **message** - add your message here.
- **language** - specify the language for the template in the correct format. The languages can be specified in the following format: for English - "en_US", for French - "fr_FR", for Japanese - "jp_JP" etc.

  NOTE

  - If this format is not followed, and you send inputs like "en", it results in an error and displays a message saying, "8800023=invalid language".
  - To create messages in multiple languages, create separate messages for each language.

Sample PUT Request Payload

The following code snippet shows the request payload for the "Update Message" API.

```
{
    "message": "[OTP] is your One-Time Password to login."
}
```

Where you can only update the message. In the above example, the [OTP] must be replaced with appropriate values.

NOTE

During the PUT operation, only the field type, "message" can be updated, other fields cannot be updated, therefore those fields do not get displayed in the payload.

Using Message Templates in the Factors (OTP and Push)

Once you create the message templates, these can be used in the following APIs:

- /OTPRegistrar for SMS and Email
- /OTPGenerator for SMS and Email
- /VIPCredentialRegistrar for Push
- /PushGenerator for Push

Example

The following example shows how to create a message template and use it in the OTPRegistrar API.

1. Assuming that you have created the following message using the Message API - POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/Messages:

   ```
   {
     "action": "LOGIN",
     "messageType": "SMS",
     "message": "[OTP] est votre mot de passe à usage unique pour vous connecter. L'OTP est valide jusqu'à
    [MINUTES] minutes.",
     "language": "fr_FR"
   }
   ```

2. Now, while you use the OTP Registrar call - POST https://${SSP_FQDN}/{{tenantName}}/factor/v1/OTPRegistrar, specify the "language": "fr_FR" in the request of the API, as shown in the following sample:

   ```
   {
     "credType": "SMS",
     "credValue": "{{userMobileNumber}}}}",
     "language": "fr_FR"
   }
   ```

NOTE
If you do not specify the language in the request payload of the /OTPRegistrar API, or you specify a language for which you have not created a message, the system defaults to "en_US".

3. The response sends the message to the recipient in the specified language, that is French (in the example).

Similarly, you can create and use the message templates in the other specified APIs to send language-specific messages.

For sample request and response payloads of all these URLs, refer to Message Template API Swagger documentation.

End-user Notification Templates

IDSP provides the following templates for end-user notifications. You can modify them using the Message API - PUT https://${SSP_FQDN}/{{tenantName}}/admin/v1/Messages. Only modify the content of the `message` parameter.

Register Messages

- Push Message Title

```
{
  "messageType": "PUSH.MESSAGE.TITLE",
  "action": "REGISTER",
  "message": "Push Authentication"
},
```

- Push Message Text

```
{
  "messageType": "PUSH.MESSAGE.TEXT",
  "action": "REGISTER",
  "message": "Please approve your transaction"
},
```

- Push Message Profile

```
{
  "messageType": "PUSH.MESSAGE.PROFILE",
  "action": "REGISTER",
  "message": "https://symantec.com"
},
```

New User Welcome Messages

- Email Message Subject

```
{
  "messageType": "NEW.USER.WELCOME.EMAIL.SUBJECT",
  "action": "NEW_USER_WELCOME",
  "message": "Welcome!"
},
```

- Email Message Body

```
{
  "messageType": "NEW.USER.WELCOME.EMAIL.BODY",
  "action": "NEW_USER_WELCOME",
  "message": "A new account has been created for you. Your login id for this account is:
 '${idProfile.user_loginid}'.\n\nYou can set the password for your account using the following
  link: $$PASSWORD_RESET_LINK$$.\n\nIf you believe this account create request was initiated in
  error, contact your system administrator at $$HELPDESK_EMAIL_DL$$.\n\nThis link will expire in $
$PASSWORD_RESET_TOKEN_EXPIRY_MINS$$ days."
```

```
},
```

- SMS Message Text

```
{
  "messageType": "NEW.USER.WELCOME.SMS",
  "action": "NEW_USER_WELCOME",
  "message": "A new account has been created for you. Your login id for this account is:
 '${idProfile.user_loginid}'.\n\nYou can set the password for your account using the following
 link: $$PASSWORD_RESET_LINK$$.\n\nIf you believe this account create request was initiated in
 error, contact your system administrator at $$HELPDESK_EMAIL_DL$$.\n\nThis link will expire in $
$PASSWORD_RESET_TOKEN_EXPIRY_MINS$$ days."
 },
```

Email and Phone Verification Messages

- Email Message Subject

```
{
  "messageType": "VERIFY.EMAIL.SUBJECT",
  "action": "VERIFY_EMAIL",
  "message": "Verify Email"
 },
```

- Email Message Body

```
{
  "messageType": "VERIFY.EMAIL.BODY",
  "action": "VERIFY_EMAIL",
  "message": "Email verification has been initiated for ${idProfile.user_loginid}.\n\nIf you have initiated
 this request, click the following link: $$EMAIL_VERIFICATION_LINK$$ to verify your email: $$ADDED_EMAIL$
$.\n\nIf you have not initiated this request, contact your system administrator at $$HELPDESK_EMAIL_DL$$.\n
\nThis link will expire in $$IDENTITY_VERIFICATION_TOKEN_EXPIRY_MINS$$ minutes."
},
```

- SMS Message Text

```
{
  "messageType": "VERIFY.PHONE.SMS",
  "action": "VERIFY_PHONE",
  "message": "Phone number verification has been initiated for ${idProfile.user_loginid}.\n\nIf you
 have initiated this request, click the following link: $$PHONE_VERIFICATION_LINK$$ to verify your phone
 number: $$ADDED_PHONE$$.\n\nIf you have not initiated this request, contact your system administrator at $
$HELPDESK_EMAIL_DL$$.\n\nThis link will expire in $$IDENTITY_VERIFICATION_TOKEN_EXPIRY_MINS$$ minutes."
},
```

Login Messages

- Email Message Subject

```
{
  "messageType": "EMAIL.SUBJECT",
  "action": "LOGIN",
  "message": "One-Time Password."
},
```

- Email Message Body

```
{
  "messageType": "EMAIL.BODY",
```

```
  "action": "LOGIN",
  "message": "$$OTP$$ is your One-Time Password to login. The OTP is valid up to $$MINUTES$$ minutes."
},
```

- SMS Message Text

```
{
  "messageType": "SMS",
  "action": "LOGIN",
  "message": "$$OTP$$ is your One-Time Password to login. The OTP is valid up to $$MINUTES$$ minutes."
 },
```

- Push Message Title

```
{
  "messageType": "PUSH.MESSAGE.TITLE",
  "action": "LOGIN",
  "message": "Push Authentication"
},
```

- Push Message Text (Approve)

```
{
  "messageType": "PUSH.MESSAGE.TEXT",
  "action": "LOGIN",
  "message": "Please approve your transaction"
},
```

- Push Message Text (Delivered)

```
{
  "messageType": "PUSH",
  "action": "LOGIN",
  "message": "Push Delivered Successfully"
},
```

- Push Message Profile

```
{
  "messageType": "PUSH.MESSAGE.PROFILE",
  "action": "LOGIN",
  "message": "https://symantec.com"
 },
```

- Interactive Voice Response (IVR) Message Text

```
{
  "messageType": "IVR",
  "action": "LOGIN",
  "message": "$$OTP$$ is your One-Time Password to login. The OTP is valid up to $$MINUTES$$ minutes."
},
```

Password Reset by End User Messages

- Email Message Subject

```
{
  "messageType": "PASSWORD.RESET.EMAIL.SUBJECT",
  "action": "PASSWORD_RESET",
  "message": "Reset Password"
},
```

- Email Message Body

```
{
  "messageType": "PASSWORD.RESET.EMAIL.BODY",
```

```
  "action": "PASSWORD_RESET",
  "message": "A password reset has been initiated for ${idProfile.user_loginid}.\n\nIf you have initiated
 this request, click the following link to reset your password: $$PASSWORD_RESET_LINK$$.\n\nIf you have
 not initiated this request, contact your system administrator at $$HELPDESK_EMAIL_DL$$.\n\nThis link will
 expire in $$PASSWORD_RESET_TOKEN_EXPIRY_MINS$$ minutes."
},
```

- SMS Message Text

```
{
  "messageType": "PASSWORD.RESET.SMS",
  "action": "PASSWORD_RESET",
  "message": "A password reset has been initiated for ${idProfile.user_loginid}. \nIf you have initiated
 this request, click the following link to reset your password: $$PASSWORD_RESET_LINK$$.\n\nImportant: If
 you have not initiated this request, contact your system administrator at $$HELPDESK_EMAIL_DL$$. \nThis
 link will expire in $$PASSWORD_RESET_TOKEN_EXPIRY_MINS$$ minutes."
},
```

Password Reset by Administrator Messages

- Email Message Subject

```
{
  "messageType": "ADMIN.PASSWORD.RESET.EMAIL.SUBJECT",
  "action": "ADMIN_PASSWORD_RESET",
  "message": "Reset Password"
},
```

- Email Message Body

```
{
  "messageType": "ADMIN.PASSWORD.RESET.EMAIL.BODY",
  "action": "ADMIN_PASSWORD_RESET",
  "message": "A password set or reset operation has been initiated for ${idProfile.user_loginid}.\n\nClick
 the following link to set your password: $$PASSWORD_RESET_LINK$$.\n\nIf you believe this request has been
 made in error, contact your system administrator at $$HELPDESK_EMAIL_DL$$.\n\nThis link will expire in $
$PASSWORD_RESET_TOKEN_EXPIRY_MINS$$ minutes."
},
```

- SMS Message Text

```
{
  "messageType": "ADMIN.PASSWORD.RESET.SMS",
  "action": "ADMIN_PASSWORD_RESET",
  "message": "A password set or reset operation has been initiated for ${idProfile.user_loginid}.\n\nClick
 the following link to set your password: $$PASSWORD_RESET_LINK$$.\n\nIf you believe this request has been
 made in error, contact your system administrator at $$HELPDESK_EMAIL_DL$$.\n\nThis link will expire in $
$PASSWORD_RESET_TOKEN_EXPIRY_MINS$$ minutes."
},
```

User Locked by Administrator Messages

- Email Message Subject

```
{
  "messageType": "ADMIN.USER.LOCKED.EMAIL.SUBJECT",
  "action": "ADMIN_USER_LOCK",
  "message": "Account Locked"
},
```

- Email Message Body

```
{
  "messageType": "ADMIN.USER.LOCKED.EMAIL.BODY",
  "action": "ADMIN_USER_LOCK",
  "message": "Admin has locked your user account : '${idProfile.user_loginid}'."
},
```

- SMS Message Text

```
{
  "messageType": "ADMIN.USER.LOCKED.SMS",
  "action": "ADMIN_USER_LOCK",
  "message": "Admin has locked your user account '${idProfile.user_loginid}'."
},
```

User Unlocked by Administrator Email Messages

- Email Message Subject

```
{
  "messageType": "ADMIN.USER.UNLOCKED.EMAIL.SUBJECT",
  "action": "ADMIN_USER_UNLOCK",
  "message": "Account Unlocked"
},
```

- Email Message Body

```
{
  "messageType": "ADMIN.USER.UNLOCKED.EMAIL.BODY",
  "action": "ADMIN_USER_UNLOCK",
  "message": "Admin has unlocked your user account : '${idProfile.user_loginid}'."
},
```

- SMS Message Text

```
{
  "messageType": "ADMIN.USER.UNLOCKED.SMS",
  "action": "ADMIN_USER_UNLOCK",
  "message": "Admin has unlocked your user account : '${idProfile.user_loginid}'."
},
```

User Locked by Self Messages

- Email Message Subject

```
{
  "messageType": "SELF.USER.LOCKED.EMAIL.SUBJECT",
  "action": "SELF_USER_LOCK",
  "message": "Account Locked"
},
```

- Email Message Body

```
{
  "messageType": "SELF.USER.LOCKED.EMAIL.BODY",
  "action": "SELF_USER_LOCK",
  "message": "Your user account : '${idProfile.user_loginid}' has been locked due to unsuccessful
 authentication attempts."
},
```

- SMS Message Text

```
{
  "messageType": "SELF.USER.LOCKED.SMS",
```

```
  "action": "SELF_USER_LOCK",
  "message": "Your user account : '${idProfile.user_loginid}' has been locked to unsuccessful
 authentication attempts."
},
```

User Unlocked by Self Messages

- Email Message Subject

```
{
  "messageType": "SELF.USER.UNLOCKED.EMAIL.SUBJECT",
  "action": "SELF_USER_UNLOCK",
  "message": "Account Unlocked"
},
```

- Email Message Body

```
{
  "messageType": "SELF.USER.UNLOCKED.EMAIL.BODY",
  "action": "SELF_USER_UNLOCK",
  "message": "Your user account : '${idProfile.user_loginid}' has been unlocked successfully."
}
```

- SMS Message Text

```
{
  "messageType": "SELF.USER.UNLOCKED.SMS",
  "action": "SELF_USER_UNLOCK",
  "message": "Your user account : '${idProfile.user_loginid}' has been unlocked successfully."
},
```

New Device Sign-in

- Email Message Subject

```
{
  "messageType": "NEW.DEVICE.SIGNIN.EMAIL.SUBJECT",
  "action": "NEW_DEVICE_SIGNIN",
  "message": "Signin from a new device"
 },
```

- Email Message Body

```
{
  "messageType": "NEW.DEVICE.SIGNIN.EMAIL.BODY",
  "action": "NEW_DEVICE_SIGNIN",
      "message": "A new device '$DEVICE_DETAILS' with os '$OS_DETAILS' and browser '$BROWSER_DETAILS' has
 been used to access your account identified by login id: ''${idProfile.user_loginid}'. If you have not
 initiated this request, contact your system administrator at $$HELPDESK_EMAIL_DL$$."
},
```

- SMS Message Text

```
{
  "messageType": "NEW.DEVICE.SIGNIN.SMS",
  "action": "NEW_DEVICE_SIGNIN",
  "message": "A new device '$DEVICE_DETAILS' with os '$OS_DETAILS' and browser '$BROWSER_DETAILS' has
 been used to access your account identified by login id: ''${idProfile.user_loginid}'. If you have not
 initiated this request, contact your system administrator at $$HELPDESK_EMAIL_DL$$."
    },
```

User Password Changed Messages

- Email Message Subject

```
{
  "messageType": "PASSWORD.CHANGE.COMPLETED.EMAIL.SUBJECT",
  "action": "USER_PASSWORD_CHANGE_COMPLETED",
  "message": "Password Change"
},
```

- Email Message Body

```
{
  "messageType": "PASSWORD.CHANGE.COMPLETED.EMAIL.BODY",
  "action": "USER_PASSWORD_CHANGE_COMPLETED",
  "message": "Password changed successfully for User account identified by login id:
 '${idProfile.user_loginid}'."
},
```

- SMS Message Text

```
{
  "messageType": "PASSWORD.CHANGE.COMPLETED.SMS",
  "action": "USER_PASSWORD_CHANGE_COMPLETED",
  "message": "Password changed successfully for User account identified by login id:
 '${idProfile.user_loginid}'."
},
```

Credential Created by Administrator

- Email Message Subject

```
{
  "messageType": "ADMIN.USER.CRED.CREATE.EMAIL.SUBJECT",
  "action": "USER_CREDENTIAL_CREATE_COMPLETED",
  "message": "Credential Created"
},
```

- Email Message Body

```
{
  "messageType": "ADMIN.USER.CRED.CREATE.EMAIL.BODY",
  "action": "USER_CREDENTIAL_CREATE_COMPLETED",
  "message": "$$USER_CRED_TYPE$$ credential: $$USER_CRED_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been registered by admin successfully."
},
```

- SMS Message Text

```
{
  "messageType": "ADMIN.USER.CRED.CREATE.SMS",
  "action": "USER_CREDENTIAL_CREATE_COMPLETED",
  "message": "$$USER_CRED_TYPE$$ credential: $$USER_CRED_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been registered by admin successfully."
},
```

Credential Updated by Administrator

- Email Message Subject

```
{
  "messageType": "ADMIN.USER.CRED.UPDATE.EMAIL.SUBJECT",
  "action": "USER_CREDENTIAL_UPDATE_COMPLETED",
```

```
    "message": "Credential Updated"
  },
```

- Email Message Body

```
  {
    "messageType": "ADMIN.USER.CRED.UPDATE.EMAIL.BODY",
    "action": "USER_CREDENTIAL_UPDATE_COMPLETED",
    "message": "$$USER_CRED_TYPE$$ credential: $$USER_CRED_VALUE$$ for User account identified by login id:
   '${idProfile.user_loginid}' has been updated by admin successfully."
  },
```

- SMS Message Text

```
  {
    "messageType": "ADMIN.USER.CRED.UPDATE.SMS",
    "action": "USER_CREDENTIAL_UPDATE_COMPLETED",
    "message": "$$USER_CRED_TYPE$$ credential: $$USER_CRED_VALUE$$ for User account identified by login id:
   '${idProfile.user_loginid}' has been updated by admin successfully."
  },
```

Credential Deleted by Administrator Messages

- Email Message Subject

```
  {
    "messageType": "ADMIN.USER.CRED.DELETE.EMAIL.SUBJECT",
    "action": "USER_CREDENTIAL_DELETE_COMPLETED",
    "message": "Credential Deleted"
  },
```

- Email Message Body

```
  {
    "messageType": "ADMIN.USER.CRED.DELETE.EMAIL.BODY",
    "action": "USER_CREDENTIAL_DELETE_COMPLETED",
    "message": "$$USER_CRED_TYPE$$ credential: $$USER_CRED_VALUE$$ for User account identified by login id:
   '${idProfile.user_loginid}' has been deleted by admin successfully."
  },
```

- SMS Message Text

```
  {
    "messageType": "ADMIN.USER.CRED.DELETE.SMS",
    "action": "USER_CREDENTIAL_DELETE_COMPLETED",
    "message": "$$USER_CRED_TYPE$$ credential: $$USER_CRED_VALUE$$ for User account identified by login id:
   '${idProfile.user_loginid}' has been deleted by admin successfully."
  },
```

Credential Created by End User Messages

- Email Message Subject

```
  {
    "messageType": "SELF.USER.CRED.CREATE.EMAIL.SUBJECT",
    "action": "SELF_USER_CREDENTIAL_CREATE_COMPLETED",
    "message": "Credential Created"
  },
```

- Email Message Body

```
  {
    "messageType": "SELF.USER.CRED.CREATE.EMAIL.BODY",
```

```
  "action": "SELF_USER_CREDENTIAL_CREATE_COMPLETED",
  "message": "$$USER_CRED_TYPE$$ credential: $$USER_CRED_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been registered successfully."
},
```

- SMS Message Text

```
{
  "messageType": "SELF.USER.CRED.CREATE.SMS",
  "action": "SELF_USER_CREDENTIAL_CREATE_COMPLETED",
  "message": "$$USER_CRED_TYPE$$ credential: $$USER_CRED_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been registered successfully."
},
```

Credential Updated by End User Messages

- Email Message Subject

```
{
  "messageType": "SELF.USER.CRED.UPDATE.EMAIL.SUBJECT",
  "action": "SELF_USER_CREDENTIAL_UPDATE_COMPLETED",
  "message": "Credential Updated"
},
```

- Email Message Body

```
{
  "messageType": "SELF.USER.CRED.UPDATE.EMAIL.BODY",
  "action": "SELF_USER_CREDENTIAL_UPDATE_COMPLETED",
  "message": "$$USER_CRED_TYPE$$ credential: $$USER_CRED_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been updated successfully."
},
```

- SMS Message Text

```
{
  "messageType": "SELF.USER.CRED.UPDATE.SMS",
  "action": "SELF_USER_CREDENTIAL_UPDATE_COMPLETED",
  "message": "$$USER_CRED_TYPE$$ credential: $$USER_CRED_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been updated successfully."
},
```

Credential Deleted by End User Messages

- Email Message Subject

```
{
  "messageType": "SELF.USER.CRED.DELETE.EMAIL.SUBJECT",
  "action": "SELF_USER_CREDENTIAL_DELETE_COMPLETED",
  "message": "Credential Deleted"
},
```

- Email Message Body

```
{
  "messageType": "SELF.USER.CRED.DELETE.EMAIL.BODY",
  "action": "SELF_USER_CREDENTIAL_DELETE_COMPLETED",
  "message": "$$USER_CRED_TYPE$$ credential: $$USER_CRED_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been deleted successfully."
},
```

- SMS Message Text

```
{
  "messageType": "SELF.USER.CRED.DELETE.SMS",
  "action": "SELF_USER_CREDENTIAL_DELETE_COMPLETED",
  "message": "$$USER_CRED_TYPE$$ credential: $$USER_CRED_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been deleted successfully."
},
```

Email Address Added by Administrator Messages

- Email Message Subject

```
{
  "messageType": "ADMIN.USER.EMAIL.ADD.EMAIL.SUBJECT",
  "action": "USER_EMAIL_ADD_COMPLETED",
  "message": "Email Added"
},
```

- Email Message Body

```
{
  "messageType": "ADMIN.USER.EMAIL.ADD.EMAIL.BODY",
  "action": "USER_EMAIL_ADD_COMPLETED",
  "message": "A new email $$USER_EMAIL_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been added by admin successfully."
},
```

- SMS Message Text

```
{
  "messageType": "ADMIN.USER.EMAIL.ADD.SMS",
  "action": "USER_EMAIL_ADD_COMPLETED",
  "message": "A new email $$USER_EMAIL_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been added by admin successfully."
},
```

Email Address Updated by Administrator Messages

- Email Message Subject

```
{
  "messageType": "ADMIN.USER.EMAIL.UPDATE.EMAIL.SUBJECT",
  "action": "USER_EMAIL_UPDATE_COMPLETED",
  "message": "Email Updated"
},
```

- Email Message Body

```
{
  "messageType": "ADMIN.USER.EMAIL.UPDATE.EMAIL.BODY",
  "action": "USER_EMAIL_UPDATE_COMPLETED",
  "message": "Email $$USER_EMAIL_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been updated by admin successfully."
},
```

- SMS Message Text

```
{
  "messageType": "ADMIN.USER.EMAIL.UPDATE.SMS",
  "action": "USER_EMAIL_UPDATE_COMPLETED",
  "message": "Email $$USER_EMAIL_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been updated by admin successfully."
},
```

Email Deleted by Administrator Messages

- Email Message Subject

```
{
  "messageType": "ADMIN.USER.EMAIL.DELETE.EMAIL.SUBJECT",
  "action": "USER_EMAIL_DELETE_COMPLETED",
  "message": "Email Removed"
},
```

- Email Message Body

```
{
  "messageType": "ADMIN.USER.EMAIL.DELETE.EMAIL.BODY",
  "action": "USER_EMAIL_DELETE_COMPLETED",
  "message": "Email $$USER_EMAIL_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been removed by admin successfully."
},
```

- SMS Message Text

```
{
  "messageType": "ADMIN.USER.EMAIL.DELETE.SMS",
  "action": "USER_EMAIL_DELETE_COMPLETED",
  "message": "Email $$USER_EMAIL_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been removed by admin successfully."
},
```

Email Address Added by End User Messages

- Email Message Subject

```
{
  "messageType": "ME.USER.EMAIL.ADD.EMAIL.SUBJECT",
  "action": "ME_USER_EMAIL_ADD_COMPLETED",
  "message": "Email Added"
},
```

- Email Message Body

```
{
  "messageType": "ME.USER.EMAIL.ADD.EMAIL.BODY",
  "action": "ME_USER_EMAIL_ADD_COMPLETED",
  "message": "A new email $$USER_EMAIL_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been added successfully."
},
```

- SMS Message Text

```
{
  "messageType": "ME.USER.EMAIL.ADD.SMS",
  "action": "ME_USER_EMAIL_ADD_COMPLETED",
  "message": "A new email $$USER_EMAIL_VALUE$$ for User account identified by login id:
 '${idProfile.user_loginid}' has been added successfully."
},
```

Email Address Updated by End User

- Email Message Subject

```
{
  "messageType": "ME.USER.EMAIL.UPDATE.EMAIL.SUBJECT",
  "action": "ME_USER_EMAIL_UPDATE_COMPLETED",
```

```
    "message": "Email Updated"
  },
```

- Email Message Body

```
  {
    "messageType": "ME.USER.EMAIL.UPDATE.EMAIL.BODY",
    "action": "ME_USER_EMAIL_UPDATE_COMPLETED",
    "message": "Email $$USER_EMAIL_VALUE$$ for User account identified by login id:
   '${idProfile.user_loginid}' has been updated successfully."
  },
```

- SMS Message Text

```
  {
    "messageType": "ME.USER.EMAIL.UPDATE.SMS",
    "action": "ME_USER_EMAIL_UPDATE_COMPLETED",
    "message": "Email $$USER_EMAIL_VALUE$$ for User account identified by login id:
   '${idProfile.user_loginid}' has been updated successfully."
  },
```

Email Deleted by End User

- Email Message Subject

```
  {
    "messageType": "ME.USER.EMAIL.DELETE.EMAIL.SUBJECT",
    "action": "ME_USER_EMAIL_DELETE_COMPLETED",
    "message": "Email Removed"
  },
```

- Email Message Body

```
  {
    "messageType": "ME.USER.EMAIL.DELETE.EMAIL.BODY",
    "action": "ME_USER_EMAIL_DELETE_COMPLETED",
    "message": "Email $$USER_EMAIL_VALUE$$ for User account identified by login id:
   '${idProfile.user_loginid}' has been removed successfully."
  },
```

- SMS Message Text

```
  {
    "messageType": "ME.USER.EMAIL.DELETE.SMS",
    "action": "ME_USER_EMAIL_DELETE_COMPLETED",
    "message": "Email $$USER_EMAIL_VALUE$$ for User account identified by login id:
   '${idProfile.user_loginid}' has been removed successfully."
  },
```

Account Deleted by Administrator Messages

- Email Message Subject

```
  {
    "messageType": "USER.ACCOUNT.DELETED.EMAIL.SUBJECT",
    "action": "USER_ACCOUNT_DELETED",
    "message": "User Account $$USER_LOGINID$$ Deleted"
  },
```

- Email Message Body

```
  {
    "messageType": "USER.ACCOUNT.DELETED.EMAIL.BODY",
```

```
  "action": "USER_ACCOUNT_DELETED",
  "message": "Your User account identified by login id: $$USER_LOGINID$$ has been deleted by Admin."
}
```

- SMS Message Text

```
{
  "messageType": "USER.ACCOUNT.DELETED.SMS",
  "action": "USER_ACCOUNT_DELETED",
  "message": "Your User account identified by login id: $$USER_LOGINID$$ has been deleted by Admin."
},
```

Reference Links

- Message Template API Error Code
- OTP Services
- VIP Access Push Service

Managing Message Provider Configuration

Configure and manage third-party message providers using the Message Provider API to deliver multi-channel authentication codes and notifications.

The Message Provider API enables the integration of external vendors—such as Twilio, SMTP, and Twilio SendGrid—with IDSP. As a system administrator, you can use this interface to manage service providers and deliver messages to registered mobile devices or email addresses via SMS, Email, and Push notifications.

This topic includes the following sections:

- Supported Message Providers
- Message Provider APIs
- Message Provider Configurations
  - SMS provider - Twilio Configuration
  - Email provider - SMTP Configuration
  - Email provider - Emailie Configuration
  - Email provider - Twilio SendGrid Configuration
  - Push provider - FCM Configuration

Message Providers Supported in IDSP

Existing message providers in the solution and their modes of delivery are listed here:

- SMS- Twilio (default provider)
- Email- SMTP (default provider), Emailie, Twilio SendGrid

  NOTE
  To configure the email provider, you must enable the global setting, "emailprovidertobeused" in the solution. See Settings for more info.
- Push- VIP Access Push (default provider)

  NOTE
  To configure the VIP Access Push provider, see VIP Access Push Authenticator documentation for more info.

APIs to Manage the Message Providers

The following table displays the list of Provider APIs that you can use to manage the message provider configurations:

Operation Description	Operation Type	Endpoint URLs
System Tenant admin retrieves the list of all message providers that are configured in the solution	GET	https://${SSP_MGMT_FQDN}/system/admin/v1
System Tenant admin retrieves the message provider configured in the solution by specifying the provider name	GET	https://${SSP_MGMT_FQDN}/system//admin/v ame}}
System admin updates configurations of a message provider	PATCH	https://${SSP_MGMT_FQDN}/system/admin/v1 {{providerName}}

NOTE

- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://${SSP_MGMT_FQDN}/<systemtenantName>/admin/v1/MessageProviders`
  should look similar to:
  `https://mgmt-example.com/system/admin/v1/MessageProviders`
- To know the value of ${SSP_MGMT_FQDN} and ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<systemTenantName>` is "system".
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges .

GET Sample

```
GET https://${SSP_MGMT_FQDN}/system/admin/v1/MessageProviders
```

The following is the response of the GET API call:

```
[
    {
        "providerName": "twilioSendGrid",
        "providerProperties": [
            {
                "name": "postURL",
                "value": "<<provide postURL>>"
            },
            {
                "name": "apiKey",
                "value": "<<provide apikey>>"
            },
            {
                "name": "twilioSendGrid_guid",
                "value": "300"
            },
            {
                "name": "retryCount",
                "value": "4"
            },
            {
                "name": "fromAddress",
```

```
                "value": "<<provide fromAddress>>"
            },
            {
                "name": "class",
                "value":
"com.broadcom.layer7authentication.core.service.mds.providers.email.TwilioSendGridEmailService"
            }
        ],
        "origin": "global"
    },
    {
        "providerName": "fcm",
        "providerProperties": [
            {
                "name": "trustStorePassword",
                "value": "<<provide trustStorePassword>>"
            },
            {
                "name": "fcm_guid",
                "value": "500"
            },
            {
                "name": "retryCount",
                "value": "4"
            },
            {
                "name": "serverKey",
                "value": "<<provide serverKey>>"
            },
            {
                "name": "notificationSound",
                "value": "default"
            },
            {
                "name": "serverURL",
                "value": "<<provide serverURL>>"
            },
            {
                "name": "notificationBody",
                "value": "<<provide notificationBody>>"
            },
            {
                "name": "trustStore",
                "value": "<<provide trustStore>>"
            },
            {
                "name": "iconURL",
                "value": "<<provide iconURL>>"
            },
            {
                "name": "title",
                "value": "<<provide title>>"
            },
```

```
            {
                "name": "class",
                "value": "com.broadcom.layer7authentication.core.service.mds.providers.push.PushFCMService"
            },
            {
                "name": "authInterval",
                "value": "<<provide authInterval>>"
            }
        ],
        "origin": "global"
    },
    {
        "providerName": "twillio",
        "providerProperties": [
            {
                "name": "accountID",
                "value": "<<provide accountID>>"
            },
            {
                "name": "twillio_guid",
                "value": "100"
            },
            {
                "name": "fromPhone",
                "value": "+1xxxxxxxxxx"
            },
            {
                "name": "retryCount",
                "value": "4"
            },
            {
                "name": "authToken",
                "value": "<<provide authToken>>"
            },
            {
                "name": "class",
                "value": "com.broadcom.layer7authentication.core.service.mds.providers.sms.TwilioSmsService"
            }
        ],
        "origin": "global"
    },
    {
        "providerName": "smtp",
        "providerProperties": [
            {
                "name": "smtpSSLEnabled",
                "value": "TRUE"
            },
            {
                "name": "smtpFromAddress",
                "value": "<<provide smtpFromAddress>>"
            },
            {
```

```
            "name": "smtpUserName",
            "value": "<<provide smtpUserName>>"
        },
        {
            "name": "retryCount",
            "value": "3"
        },
        {
            "name": "smtp_guid",
            "value": "800"
        },
        {
            "name": "smtpUserPassword",
            "value": "<<provide smtpUserPassword>>"
        },
        {
            "name": "smtpStartTLSEnabled",
            "value": "TRUE"
        },
        {
            "name": "class",
            "value": "com.broadcom.layer7authentication.core.service.mds.providers.email.SMTPEmailService"
        },
        {
            "name": "smtpPortNumber",
            "value": "<<provide smtpPortNumber>>"
        },
        {
            "name": "smtpAuthEnabled",
            "value": "TRUE"
        },
        {
            "name": "smtpServerHostName",
            "value": "<<provide smtpServerHostName>>"
        }
    ],
    "origin": "global"
},
{
    "providerName": "emailie",
    "providerProperties": [
        {
            "name": "postURL",
            "value": "<<provide postURL>>"
        },
        {
            "name": "apiKey",
            "value": "<<provide apikey>>"
        },
        {
            "name": "retryCount",
            "value": "4"
        },
```

```
            {
                "name": "fromAddress",
                "value": "<<provide smtpFromAddress>>"
            },
            {
                "name": "emailie_guid",
                "value": "400"
            },
            {
                "name": "class",
                "value": "com.broadcom.layer7authentication.core.service.mds.providers.email.EmailIEService"
            }
        ],
        "origin": "global"
    },
    {
        "providerName": "clickatell",
        "providerProperties": [
            {
                "name": "password",
                "value": "<<provide password>>"
            },
            {
                "name": "retryCount",
                "value": "4"
            },
            {
                "name": "clickatell_guid",
                "value": "200"
            },
            {
                "name": "class",
                "value":
"com.broadcom.layer7authentication.core.service.mds.service.providers.sms.ClickATellSMSService"
            },
            {
                "name": "apiID",
                "value": "<<provide apiID>>"
            },
            {
                "name": "url",
                "value": "<<provide url>>"
            },
            {
                "name": "username",
                "value": "<<provide username>>"
            }
        ],
        "origin": "global"
    },{
        "providerName": "smtp2",
        "providerProperties": [
            {
```

```
                "name": "smtpSSLEnabled",
                "value": "false"
            },
            {
                "name": "smtpFromAddress",
                "value": "<<provide smtpFromAddress>>"
            },
            {
                "name": "retryCount",
                "value": "3"
            },
            {
                "name": "smtpUserName",
                "value": "<<provide smtpUserName>>"
            },
            {
                "name": "smtpUserPassword",
                "value": "<<provide smtpAuthEnabled>>"
            },
            {
                "name": "smtpStartTLSEnabled",
                "value": "false"
            },
            {
                "name": "smtp2_guid",
                "value": "900"
            },
            {
                "name": "class",
                "value":
 "com.broadcom.layer7authentication.core.service.mds.service.providers.email.SMTPEmailService"
            },
            {
                "name": "smtpAuthEnabled",
                "value": "false"
            },
            {
                "name": "smtpPortNumber",
                "value": "<<provide smtpPortNumber>>"
            },
            {
                "name": "smtpServerHostName",
                "value": "<<provide smtpServerHostName>>"
            }
        ],
        "origin": "global"
    }
]
```

PATCH Sample

```
PATCH https://${SSP_MGMT_FQDN}/system/admin/v1/MessageProviders/{{providerName}}
```

The following code snippet shows a sample of a PATCH payload where you can update any property value by specifying the provider name:

```
[
    {
        "name": "retryCount",
        "value": "3"
    }
]
```

To view the request and response payload for these URLs, refer to the Provider API Swagger documentation.

List of Message Providers and their attributes that you must update while configuring each provider:

- SMS provider - Twilio Configuration
- Email provider - SMTP Configuration
- Email provider - Emailie Configuration
- Email provider - Twilio SendGrid Configuration
- Push provider - FCM Configuration

NOTE
The values that are provided for each of these attributes are samples. You must update them with appropriate values.

Twilio Attributes

NOTE

Twilio uses HTTPS on standard port 443. Therefore, firewalls and proxies must allow outbound HTTPS traffic on port 443 to connect to the Twilio REST API hosted in https://api.twilio.com/. For more details, see Requirements-for-Connecting-to-the-Twilio-REST-API.

These are the ***basic attributes*** in "twilio" that you must update when configuring with IDSP.

```
[
    {
        "name": "accountID",
        "value": "<<provide accountID>>"
    },
    {
        "name": "twillio_guid",
        "value": "100"
    },
    {
        "name": "fromPhone",
        "value": "+1xxxxxxxxxx"
    },
    {
        "name": "retryCount",
        "value": "4"
    },
    {
        "name": "authToken",
        "value": "<<provide authToken>>"
    },
    {
        "name": "class",
```

```
        "value": "com.broadcom.layer7authentication.core.service.mds.providers.sms.TwilioSmsService"
    }
]
```

TwilioSendGrid Attributes

These are the ***basic attributes*** in "twilioSendGrid" that you must update when configuring with IDSP.

```
[
    {
        "name": "postURL",
        "value": "<<provide postURL>>"
    },
    {
        "name": "apiKey",
        "value": "<<provide apikey>>"
    },
    {
        "name": "twilioSendGrid_guid",
        "value": "300"
    },
    {
        "name": "retryCount",
        "value": "4"
    },
    {
        "name": "fromAddress",
        "value": "<<provide fromAddress>>"
    },
    {
        "name": "class",
        "value":
 "com.broadcom.layer7authentication.core.service.mds.providers.email.TwilioSendGridEmailService"
    }
]
```

FCM Attributes

These are the ***basic attributes*** in "FCM" that you must update when configuring with IDSP.

```
[
    {
        "name": "trustStorePassword",
        "value": "<<provide trustStorePassword>>"
    },
    {
        "name": "fcm_guid",
        "value": "500"
    },
    {
        "name": "retryCount",
        "value": "4"
    },
    {
```

```
        "name": "serverKey",
        "value": "<<provide serverKey>>"
    },
    {
        "name": "notificationSound",
        "value": "default"
    },
    {
        "name": "serverURL",
        "value": "<<provide serverURL>>"
    },
    {
        "name": "notificationBody",
        "value": "<<provide notificationBody>>"
    },
    {
        "name": "trustStore",
        "value": "<<provide trustStore>>"
    },
    {
        "name": "iconURL",
        "value": "<<provide iconURL>>"
    },
    {
        "name": "title",
        "value": "<<provide title>>"
    },
    {
        "name": "class",
        "value": "com.broadcom.layer7authentication.core.service.mds.providers.push.PushFCMService""
    },
    {
        "name": "authInterval",
        "value": "<<provide authInterval>>"
    }
]
```

Emailie Attributes

These are the ***basic attributes*** in "emailie" that you must update when configuring with IDSP.

```
[
    {
        "name": "postURL",
        "value": "<<provide postURL>>"
    },
    {
        "name": "apiKey",
        "value": "<<provide apikey>>"
    },
    {
        "name": "retryCount",
        "value": "4"
```

```
    },
    {
        "name": "fromAddress",
        "value": "<<provide smtpFromAddress>>"
    },
    {
        "name": "emailie_guid",
        "value": "400"
    },
    {
        "name": "class",
        "value": "com.broadcom.layer7authentication.core.service.mds.providers.email.EmailIEService"
    }
]
```

SMTP Attributes

NOTE
Based on the TCP, port your organization uses for outgoing SMTP communication, your firewall policy should allow outbound traffic on one of the following ports: TCP 25, 587, or 465".

These are the ***basic attributes*** in "SMTP" that you must update when configuring with IDSP.

```
[
    {
        "name": "smtpSSLEnabled",
        "value": "false"
    },
    {
        "name": "smtpFromAddress",
        "value": "<<provide smtpFromAddress>>"
    },
    {
        "name": "retryCount",
        "value": "3"
    },
    {
        "name": "smtpUserName",
        "value": "<<provide smtpUserName>>"
    },
    {
        "name": "smtpUserPassword",
        "value": "<<provide smtpAuthEnabled>>"
    },
    {
        "name": "smtpStartTLSEnabled",
        "value": "false"
    },
    {
        "name": "smtp2_guid",
        "value": "900"
    },
    {
```

```
        "name": "class",
        "value": "com.broadcom.layer7authentication.core.service.mds.providers.email.SMTPEmailService"
    },
    {
        "name": "smtpAuthEnabled",
        "value": "false"
    },
    {
        "name": "smtpPortNumber",
        "value": "<<provide smtpPortNumber>>"
    },
    {
        "name": "smtpServerHostName",
        "value": "<<provide smtpServerHostName>>"
    }
]
```

Reference Links to:

- Push provider - VIP Access Push Authenticator Configuration
- Provider API Error Codes

Managing Identity Providers (IDP)

Learn how to effectively create and manage Identity Providers within the Symantec Identity Security Platform by leveraging the Identity Provider APIs, while also exploring the functionality of Just-In-Time user provisioning.

In the IDSP, you have the capability to establish a trusted third-party Identity Provider (IDP) entity that is responsible for storing and validating user identities on behalf of the IDSP. The external IDP service enables efficient management of user identities and work with service providers like IDSP to authenticate the users and grant them access to the protected resource.

It is imperative that the external IDPs are compatible with SAML 2.0 (Security Assertion Markup Language 2.0) or OpenID Connect (OAuth 2.0) protocol.

IDSP can act as both an Identity Provider (IDP) to another Service Provider (SP) and a Service Provider to an external IDP.

To know more about these authentication flows, see the following links:

- IDSP Operating as a Service Provider

This topic includes the following sections:

- Identity Provider APIs
  - Deleting an Identity Provider (IDP) Referenced by Other IDSP Objects
- Working with Identity Providers
- Identity Provider Metadata Table
- JIT User Provisioning and Metadata fields

Identity Provider APIs

The following table lists the APIs that can be used by an administrator to configure, update, retrieve, and delete Identity Providers in IDSP:

Operation Description	Operation Type	Endpoint URLs
API to configure an Identity Provider in the solution.	POST	https://${SSP_FQDN}/{{tenantName}}/a
API to retrieve all IDP configurations in the solution.	GET	https://${SSP_FQDN}/{{tenantName}}/a
API to retrieve IDP configuration based on the specified IDP Id.	GET	https://${SSP_FQDN}/{{tenantName}}/a
API to update an IDP configuration by specifying the IDP Id.	PUT	https://${SSP_FQDN}/{{tenantName}}/a
API to delete an IDP configuration by specifying the IDP Id.	DELETE	https://${SSP_FQDN}/{{tenantName}}/a To forcefully delete an identity provider
APIs to Import SAML IDP Metadata		
This API imports the SAML metadata from an IDP	POST	https://${SSP_FQDN}/{{tenantName}}/a
This API is used to modify the existing SAML metadata with new updates.	PUT	https://${SSP_FQDN}/{{tenantName}}/a

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/IdentityProviders`
  should look similar to:
  `https://example.com/default/admin/v1/IdentityProviders`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

For sample request and response payloads of these URLs, refer to Identity Provider API- Swagger documentation

Deleting an Identity Provider Referenced by Other IDSP Objects

If a tenant's Identity Provider is being deleted and it is currently referenced by other IDSP objects, the `DELETE` API performs a dependency check. If a conflict is detected, and other objects are using the same Identity Provider, the API prevents the delete operation, and returns error code 8000136.

This allows the administrator to review where the Identity Provider is being used and manually remove the reference from each object. For example, the Identity Provider might be used in configurations such as Authentication (AuthN) policy, Authorization (AuthZ) Policy or Discovery Policy. **The delete operation is only allowed once all such references are removed.**

NOTE

- The HTTP response returned in case of a conflict (resource being reused by other IDSP objects) during deletion of an Identity Provider is **409**.
- When attempting to delete an identity provider that is currently referenced by other IDSP objects, the system returns error code 8000136 to prevent the deletion.
- The DELETE operation must contain the "**force=true**" query parameter to delete references of the IDSP objects. See Error Codes here.
- A helper API is available to retrieve the list of dependent objects in case of a conflict, see ReferencingObjectsHelper API.

The following DELETE operation is where the reference list gets populated in cases where there are conflicts (resource being reused by other IDSP objects):

```
DELETE https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdentityProviders/{{identityProviderId}}
```

The following sample response illustrates how this list is displayed.

Sample Response:

```
{
"errorCode": "8000136",

"errorMessage": "IdentityProvider associated with one or more policies cannot be deleted"
}
```

Working with Identity Providers

IDPs represent configuration artifacts necessary for orchestrating Bring-Your-Own-Identity (BYOI) and IDP-as-a-Authentication-Factor (IDPaaF) identity federation flows.

IDP Flows:

- **OpenID IDP Flows**:
  - Always RP-initiated
  - Supports the "state" parameter
- **SAML IDP Flows**:
  - Can be either SP-initiated or IDP-initiated
  - SP-initiated flows support the relayState parameter
  - IDP-initiated flows support relay state mapping
  - IDP-initiated flows include:
    - IDSP as IDP
    - External IAM as IDP

    NOTE
    For more information on IDP-Initiated Flows, see Identity Provider Init Flows and applicable Metadata.

In addition to using a policy-based approach to discover IDPs, any app (OIDC/SAML/Launcher) that knows a specific IDP can request sign-in using that specific IDP. This is managed via the app metadata urn:iam:app:allowIdpHint (default: false), which controls whether the app can use the idp_hint parameter to specify the IDP.

`idp_hint` Parameter Syntax

If the IDP cannot be resolved, the hint is ignored, and the flow continues as if the hint was not submitted.

The syntax for the idp_hint parameter is as follows:

- idp_hint=urn:iam:idphint:id=<idp-id>
- idp_hint=urn:iam:idphint:name=<idp-name>
- idp_hint=urn:iam:idphint:domain=<idp-domain>
- idp_hint=urn:iam:idphint:none=true, provided IDP is configured with the "urn:iam:app:allowIdpHint=true" metadata (Use this metadata to skip the IDP discovery process)

Identity Provider Metadata

For information on **Identity Provider metadata**, see IDP Metadata table.

Identity Provider Discovery - Just-In-Time (JIT) User Provisioning

For information on JIT user provisioning, see Identity Provider Discovery - Just-In-Time (JIT) User Provisioning

Symantec Identity Security Platform Operating as a Service Provider

Learn how to configure an external IDP in Symantec Identity Security Platform

This topic includes the following sections:

- Working with Identity Providers in IDSP
- Configuring IDSP as a Service Provider (SP)
  - Identity Provider APIs
  - Certificate Information
  - Configuring an external IDP in IDSP

Working with Identity Providers in IDSP

In IDSP, you can set up a trusted third-party Identity Provider (IDP) entity that manages and verifies user identities on behalf of IDSP. This external IDP service simplifies user identity management and integrates with IDSP to authenticate users and provide access to protected resources.

To ensure compatibility, external IDPs must support either the SAML 2.0 (Security Assertion Markup Language 2.0) or OpenID Connect (OAuth 2.0) protocols.

IDSP has the flexibility to function as both an Identity Provider (IDP) for another Service Provider (SP) and as a Service Provider to an external IDP.

Identity Providers are essential configurations that enable users to bring their own identity (BYOI) or use an IDP as an additional authentication factor (IDPaaF) in identity federation setups.

Types of IDP Flows	
OpenID IDP Flows:	• Always initiated by the Relying Party (RP), meaning that the request to authenticate starts from th • Supports the "state" parameter to manage session details during the authentication flow.
SAML IDP Flows	• Can be initiated by either the Service Provider (SP) or the Identity Provider (IDP). • **SP-Initiated Flows**: Begin with the Service Provider and support the use of the **"relayState" par** • **IDP-Initiated Flows**: Begin with the Identity Provider and allow for **relay state mapping** to direct • **IDP-initiated Flows include**: – IDSP as IDP: IDSP acts as the Identity Provider. – External IAM as IDP: An external Identity and Access Management (IAM) system serves as th

Configuring IDSP as a Service Provider (SP)

When the IDSP acts as a Service Provider (SP), configurations must be set up on both the IDSP (SP) side and the external IDP side. These settings establish the connection between the two systems and initiate the federated authentication flow.

To integrate an external IDP, you must create an Identity Provider configuration in IDSP, either through the Admin Console or using IDSP APIs. This setup requires the external IDP's metadata information to configure the connection. Additionally, before configuring an IDP in IDSP, consider the required certificate information for secure communication.

Prerequisites:

- Obtain the IDP's metadata xml to configure the connection between IDSP and the External IDP.

Identity Provider APIs

The following table lists the IDSP APIs available for administrators to manage and interact with Identity Providers.

Operation Description	Operation Type	Endpoint URLs
API to configure an Identity Provider in the solution.	POST	https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdentityProviders
API to retrieve all IDP configurations in the solution.	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdentityProviders
API to retrieve IDP configuration based on the specified IDP Id.	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdentityProviders/{{identityProviderId}}
API to update an IDP configuration by specifying the IDP Id.	PUT	https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdentityProviders/{{identityProviderId}}
API to delete an IDP configuration by specifying the IDP Id.	DELETE	https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdentityProviders/{{identityProviderId}}
APIs to Import SAML IDP Metadata		
This API imports the SAML metadata from an IDP	POST	https://${SSP_FQDN}/{{tenantName}}/admin/v1/SAMLIdpMetadata
This API is used to modify the existing SAML metadata with new updates.	PUT	https://${SSP_FQDN}/{{tenantName}}/admin/v1/SAMLIdpMetadata/{{identityProviderId}}

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/IdentityProviders`
  should look similar to:
  `https://example.com/default/admin/v1/IdentityProviders`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

For sample request and response payloads of these URLs, refer to Identity Provider API- Swagger documentation

Certificate Information

You may be required to create a certificate before configuring the Identity Provider in the IDSP. The decision to create or not to create a certificate depends on the following points:

- If the IDP provides you with a metadata XML that includes a certificate that you can use to verify the SAML response from it, then you are required to create a certificate. In this scenario, follow all the steps that are given in this section.
- If the IDP does not provide you with a metadata XML that includes a certificate to validate its SAML response, then you can configure the IDP without creating a certificate and keeping the attribute, "verifyCertAlias" as null. In this scenario, navigate to step 4 directly, skipping all other steps.

Follow these steps

1. Use Add/Import certificate API to create a certificate to be used in the IDP configuration.
   ```
   "https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/Certs/"
   ```
   This can also be done using the IDSP Admin Console. See Certificate Management for more details.
2. Provide the information in the certs API from the IDP metadata XML downloaded from the third party.
3. Extract the alias from the certs API response and use this value against the attribute-"verifyCertAlias" when creating the Identity Provider.
4. Create an IDP using the ".../admin/v1/IdentityProviders" API or through the Admin Console. The steps to create an IDP is described in the next section.

   NOTE
   Upon IDP creation, an IDP ID is generated. Use this IDP ID value within the IDSP policy settings to configure the IDP as an authentication factor. To know how to configure an IDP in the Authentication Policy as a factor, see Identity Provider in IDSP Policy.

Configuring an external IDP in IDSP

The steps outlined in this section details the procedure of configuring an external Identity Provider (IDP) in the IDSP.

Prerequisite:

- IDSP (ssp) is deployed in your machine.
- You have obtained the metadata from the external IDP.

  NOTE
  For precise instructions on how to set up the integration in the IDP end, please consult the documentation of the external IDP. Our documentation ONLY covers IDSP related integration steps.

Choosing the correct IDSP API to create the IDP

- If the external IDP provides the metadata XML, create the IDP using the following API:
  ```
  POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/SAMLIdpMetadata
  ```
  (Recommended for SAML flows)
- If metadata XML is not available from the external IDP, use the following API to manually setup the IDP, ensuring proper integration for user authentication with IDSP:
  ```
  POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdentityProviders
  ```
  (Supports both SAML and OIDC flows)

Detailed instructions for creating an IDP using each option are provided in the following links:

- Creating an IDP using the metadata xml
- Creating an IDP using the Identity Provider API
  - Table describing IDP properties
  - Table describing IDP metadata properties
- Identity Provider as an Authentication Factor

Creating an IDP by Importing SAML IDP Metadata

Use the SAMLIdpMetadata API to create an Identity Provider (IDP) by importing and encoding external SAML metadata XML into Symantec Identity Security Platform.

The recommended method for creating an Identity Provider (IDP) in Symantec Identity Security Platform is to import the external provider's metadata using the .../admin/v1/SAMLIdpMetadata API. This process involves consuming a standard SAML metadata XML file, which must be Base64 encoded before submission to the API.

NOTE

- If you do not receive the metadata XML from certain third-party Identity Providers (IDPs), an alternative approach is to create an IDP manually using the ".../admin/v1/IdentityProviders" API.
- If you want to use the OIDC protocol to create an IDP, using the IDSP ".../admin/v1/IdentityProviders" API is the only way out.
- Some steps and details may vary depending on the IDP you are using. It is important to consult the documentation of the specific IDP for precise instructions on how to set up the integration at their end.

Follow these steps:

1. To configure an external IDP in IDSP, you should first get the following metadata from IDSP (SSP):
   – Entity ID
   – Assertion Consumer Service (ACS)
   – Certificate (To know more about certs, see Certificate Information)
2. Perform the following steps to get the metadata in IDSP:
   – Perform a GET https://${SSP_FQDN}/default/saml/v1/metadata
   – The response fetches your content that looks similar to this:

```
<?xml version="1.0" encoding="UTF-8"?>
    <md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
 cacheDuration="P0Y0M29DT12H44G*.840S" entityID="https://example.com/testtenant"
 validUntil="2031-06-08T20:18:17.963Z">
       <md:SPSSODescriptor AuthnRequestsSigned="true" WantAssertionsSigned="true"
 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
           <md:KeyDescriptor use="signing">
               <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                   <ds:X509Data>
                       <ds:X509Certificate>DFIIDEDC.......=</ds:X509Certificate>
                   </ds:X509Data>
               </ds:KeyInfo>
           </md:KeyDescriptor>
           <md:KeyDescriptor use="encryption">
               <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                   <ds:X509Data>
                       <ds:X509Certificate>DFIDE.........=</ds:X509Certificate>
                   </ds:X509Data>
               </ds:KeyInfo>
           </md:KeyDescriptor>
           <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</md:NameIDFormat>
           <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://example.com/testtenant/saml/v1/sp/acs"/>
       </md:SPSSODescriptor>
   </md:EntityDescriptor>
```

   – The API returns the SAML metadata representing IDSP (SSP). Extract the Entity ID and AssertionConsumerService (ACS) from the metadata.

3. Now, navigate to the external IDP configuration page, enter the Entity Id and the ACS values and save the configuration.
4. You are provided with a download link to the metadata XML. This metadata has all IDP-related information, along with the required certificates.
5. Encode this metadata using an online tool, for example; base64encode.org.
6. Navigate back to the IDSP postman collection and perform the following steps:
   – Invoke the SAMLIdpMetadata API:

```
POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/SAMLIdpMetadata
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/SAMLIdpMetadata`
  should look similar to:
  `https://example.com/default/admin/v1/SAMLIdpMetadata`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Request payload:
Enter the encoded metadata and other parameter details in the request and send the request. This is a sample IDP metadata request payload. For parameter details, check the table describing the IDP metadata attributes.

```
{
"idpName": "test",
"b64IdpMetadata": "94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0i....",
"idpTokenSubjectClaim": "NameID",
"spMappingAttribute": "email",
"verifyCertAlias": "test",
"issuer": "https://example.com",
"domainName": "example.com"
"validateIdpMappingRequired": "true"
}
```

Response payload:
The response provides you with the IdentityProvider ID.

```
{
    "id": "e35db4fd-d79c-4a47-a258-11daf688b45b",
    "idpName": "test",
    "verifyCertAlias": "test",
    "samlMDidpPostBindingURL": "https://example.com/idp/endpoint/HttpPost",
    "samlMDidpRedirectBindingURL": "https://example.com/idp/endpoint/HttpRedirect",
    "issuer": "https://example.com",
    "domainName": "example.com",
    "validateIdpMappingRequired": true,
    "spMappingAttribute": "email",
    "idpTokenSubjectClaim": "NameID"
}
```

7. Activate the IDP using the following API operation:

```
PATCH ".../IdentityProviders/{{identityProviderId}}"
```

Request payload:

```
{
        "status": "active"
}
```

8. Use the value (GUID) of the IdentityProvider ID in the IDSP authentication policy to configure IDP as a factor. See Identity Provider in IDSP Policy for details. Once the policy is configured with the IDP Id, invoke the policy request.
9. Navigate to the configured application's Signing UI page and start authenticating the flow.

Table describing the IDP attributes used in the ".../admin/v1/SAMLIdpMetadata" API request

The following table outlines the list of metadata attributes and their descriptions used in the API payload.

Attribute Name	Mandatory Fields (Y/N)	Description
idpName	Y	This attribute specifies the unique name of the Identity Provider.
b64IdpMetadata	Y	Base64 encoded IDP Metadata XML.
idpTokenSubjectClaim	Y	This is the Claim/Assertion in the response consisting of the Authenticated Subject from the IDP.
spMappingAttribute	Depends on the value of validateIdpMappingRequired	This is the attribute in the IDSP (SSP) identity store, which will be mapped to the value specified by "idpTokenSubjectClaim" to disambiguate the user in the IDSP identity store.
verifyCertAlias	N	This is the alias of the certificate that is used to verify the assertion/ token in the IDP response.
issuer	Y	This attribute specifies the issuer of the token in the response sent by the IDP.
domainName	Y	This attribute specifies the domain name for the Identity Provider.
validateIdpMappingRequired	N	This attribute specifies if the value of "idpTokenSubjectClaim" in the response token from IDP must be mapped to the value of the "spMappingAttribute" in the user profile. Default value- False. Note the Following: • if the "***validateIdpMappingRequired***" attribute is set to "***true***", the value of "spMappingAttribute" must be defined to establish a mapping between the value of "idpTokenSubjectClaim" in the response token from IDP and the "spMappingAttribute" in the user profile. • if the "***validateIdpMappingRequired***" attribute is set to "***false***", then defining the "spMappingAttribute" is optional as no mapping will be established.

Creating Identity Provider using "/IdentityProviders" API

If SAML metadata is unavailable, use the Symantec Identity Security Platform Identity Provider API to manually configure an external Identity Provider (IDP). This API supports both SAML and OIDC protocols. Successful integration requires gathering specific configuration details directly from the external IDP prior to execution.

> **NOTE**
> Ensure all connection parameters and protocol-specific details are obtained from the external IDP before attempting to create the configuration.

Follow these steps:

1. Invoke the following Identity Provider API:

```
POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/IdentityProviders
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/IdentityProviders`
  should look similar to:
  `https://example.com/default/admin/v1/IdentityProviders`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Request payload:

Specify all the attributes required for creating the IDP. The following example shows the request of a SAML based IDP Provider.

For parameter details, check the table describing the IDP attributes.

```
{
        "idpName": "myidp",
        "idpType": "saml",
        "domainName": "myidp.com",
        "domainNames": "myidp.com","mynewidp.com"
        "jitEnabled": false,
        "verifyCertAlias": "myidp-saml",
        "samlMDidpPostBindingURL": "https://example.com/saml2",
        "samlMDidpRedirectBindingURL": "https://example.com/saml2",
        "samlAuthnRequestBinding": "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST",
        "idpTokenSubjectClaim": "NameID",
        "spMappingAttribute": "email",
        "subjectAllowedInRequest": false,
        "forwardACR": false,
        "oidDiscoveryUrl": null,
        "oidAuthorizeEndpoint": null,
        "oidTokenEndpoint": null,
        "oidUserInfoEndpoint": null,
        "oidJwksEndpoint": null,
        "oidRequestedScope": null,
        "oidClientId": null,
        "oidClientSecret": null,
        "oidIgnoreSSL": false,
        "oidSignatureValidationStrategy": null,
        "verifySignature": false,
        "validateIdpMappingRequired": true,
        "useCustomAmrClaim": false,
        "customAmrClaim": null,
        "customAmrMappings": {},
        "idpAmrAuthoritativeForPolicyObligations": true,
        "defaultAmr": "PWD",
        "idpClaimsMapping": [
          {
            "idpClaimName": "lname",
            "profileLogicalAttrName": "lname",
```

```
            "userData": true,
            "persistUserData": true
          }
        ],
        "autoInlineEnrollment": false,
        "issuer": "https://example.com/",
        "issuers": "https://example.com/","https://example2.com/"
        "status": "active",
        "description": null,
        "zeroFootPrint": "true",
        "ignoreSSLValidation": true,
        "sslVerifyCertAlias": null,
        "readTimeoutMillis": "10000",
        "connectTimeoutMillis": "10000"
        "itDecryptionSource": "idp",
        "itDecryptionKeyAlias": "wildcard"
}
```


Response payload:

The following response returns the SAML based IDP information created in IDSP (SSP).


```
[
   {
        "identityProviderId": "92d92b71-1f12-43e9-a41c-808e61a4e778",
        "idpName": "myidp saml",
        "idpType": "saml",
        "domainName": "myidp.com",
        "domainNames": "myidp.com","mynewidp.com"
        "jitEnabled": false,
        "verifyCertAlias": null,
        "samlMDidpPostBindingURL": "https://myidp.com/idp/endpoint/HttpPost",
        "samlMDidpRedirectBindingURL": "https://myidp.com/idp/endpoint/HttpRedirect",
        "samlAuthnRequestBinding": "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST",
        "idpTokenSubjectClaim": "NameID",
        "idpSubjectClaimSource": "token",
        "spMappingAttribute": "email",
        "subjectAllowedInRequest": false,
        "forwardACR": false,
        "oidDiscoveryUrl": null,
        "oidAuthorizeEndpoint": null,
        "oidTokenEndpoint": null,
        "oidUserInfoEndpoint": null,
        "oidJwksEndpoint": null,
        "oidRequestedScope": null,
        "oidClientId": null,
        "oidClientSecret": null,
        "oidIgnoreSSL": false,
        "oidSignatureValidationStrategy": null,
        "verifySignature": false,
        "validateIdpMappingRequired": false,
        "useCustomAmrClaim": false,
        "customAmrClaim": null,
        "customAmrMappings": {},
        "idpAmrAuthoritativeForPolicyObligations": false,
```

```
        "defaultAmr": null,
        "acrAllowedInRequest": false,
        "defaultAcrMappingInRequest": null,
        "acrRequestMapping": {},
        "idpClaimsMapping": [],
        "autoInlineEnrollment": false,
        "issuer": "https://example.com",
        "issuers": "https://example.com","https://example2.com"
        "status": "active",
        "description": null,
        "zeroFootPrint": "none",
        "ignoreSSLValidation": true,
        "sslVerifyCertAlias": null,
        "readTimeoutMillis": "10000",
        "connectTimeoutMillis": "10000",
        "itDecryptionSource": null,
        "itDecryptionKeyAlias": null,
        "createdDateTime": "2023-10-17 09:10 AM UTC",
        "updatedDateTime": "2023-10-17 09:11 AM UTC",
        "createdBy": {
            "principalType": "USER",
            "principalId": "15515583-09cc-4800-aa01-1b4f86960349",
            "principalName": "malbail"
        },
        "updatedBy": {
            "principalType": "USER",
            "principalId": "15515583-09cc-4800-aa01-1b4f86960349",
            "principalName": "malbail"
        }
    }
```


2. [***Optional step***] If you have not set the "***status***" of the IDP to active in the request payload, then the IDP is in inactive state by default. In such a case, activate the IDP using the following API operation:

   ```
   PATCH ".../IdentityProviders/{{identityProviderId}}"
   ```

   Request Payload:

   ```
   {
        "status": "active"
   }
   ```

3. Use the value (GUID) of the IdentityProvider ID in the IDSP policy to configure IDP as a factor. See Identity Provider in IDSP Policy for details. Once the policy is configured with the IDP Id, invoke the policy request.
4. Navigate to the configured application's Signing UI page and start authenticating the flow.

For sample request and response payloads of all these URLs, refer to IDP Configuration API- Swagger Documentation.

Identity Provider Properties

Learn about Identity Provider properties.

The table below outlines the attributes used when configuring the IdentityProvider for SAML and OIDC flows. Attribute configuration may differ based on the type of IDP. SAML attributes are prefixed with "saml," OIDC attributes with "oid," and common attributes used in both protocols are also included.

Attribute Name	Mandatory Fields (Y/N)	Description
idpName	Y	This attribute specifies the unique name of the Identity Provider.

idpType	Y	This attribute specifies the type of the Identity Provider. ***Possible values*** -saml or openid
domainName	Y	This attribute specifies the domain name for the Identity Provider.
domainNames	N	This attribute specifies the additional domain names associated with th
jitEnabled	N	This attribute checks if "Just In Time Provisioning" (JIT) is enabled. ***Default value***- False.
verifyCertAlias	N	This is the alias of the certificate that is used to verify the assertion/tok
samlMDidpPostBindingURL	N	This is the HTTP-POST Binding URL for the SAML Authentication req
samlMDidpRedirectBindingURL	N	This is the HTTP-REDIRECT Binding URL for the SAML Authenticatio
samlAuthnRequestBinding	Yes if idpType=saml	This attribute specifies the type of request binding to be used to send Possible Values - • urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST • urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect
idpTokenSubjectClaim	Yes, if idpType=oidc	This is the name of the claim returned from the IDP containing the Aut the authenticated user to the Symantec Identity Security Platform iden
idpSubjectClaimSource	Yes, if idpType=oidc	This attribute specifies the source of the claim specified in the claim, " • For OpenID - specifies whether the "idpTokenSubjectClaim" is pres "userinfo" endpoint.
spMappingAttribute	Depends on the value of the "validateIdpMappingRequired" property	This is the attribute in the IDSP (SSP) identity store, which will be map disambiguate the user in the IDSP identity store.
subjectAllowedInRequest	N	This attribute specifies the Subject that is allowed to be sent in the req Note the Following: • If the "subjectAllowedInRequest" attribute is set to "true," the value the value of "spMappingAttribute" is searched up in the user profile • If the "subjectAllowedInRequest" attribute is set to "false, then defi be sent in the request.
forwardACR	N	Default value- False.
oidDiscoveryUrl	N	This attribute represents the OpenID discovery URL.
oidAuthorizeEndpoint	Yes, if idpType=oidc	This attribute represents the OpenID provider "/authorize" endpoint.
oidTokenEndpoint	Yes, if idpType=oidc	This attribute represents the OpenID provider "/token" endpoint.
oidUserInfoEndpoint	Yes, if idpType=oidc	This attribute represents the OpenID provider "/userinfo" endpoint.
oidJwksEndpoint	Yes, if idpType=oidc	This attribute represents the OpenID provider "/jwks" endpoint.
oidRequestedScope	Yes, if idpType=oidc	This attribute represents the requested scope for the OpenID provider
oidClientID	Yes, if idpType=oidc	This is the client ID that is obtained from the OpenID provider when th
oidClientSecret	Yes, if idpType=oidc	This is the client Secret obtained from OID provider when IDSP is regi
oidSignatureValidationStrategy	N	This attribute defines the signature validation strategy for Identity Toke Possible values: • jwks - The signature validation is performed based on the jwks key • cert - The signature validation is performed using the matching cer be added/imported in the IDSP (SSP) certificate store using the "/C • any - IDSP (SSP) will try to validate signature using "jwks" stratgey then "cert" strategy is used to validate the signature.

verifySignature	N	This attribute states whether IDSP should verify the signature of the to
validateIdpMappingRequired	N	This attribute specifies if the value of "idpTokenSubjectClaim" in the re the "spMappingAttribute" in the user profile. Default value- False. Note the Following: • if the "validateIdpMappingRequired" attribute is set to "true", the va a mapping between the value of "idpTokenSubjectClaim" in the res the user profile. • if the "validateIdpMappingRequired" attribute is set to "false", then mapping will be established.
useCustomAmrClaim	N	If "idpAmrAuthoritativeForPolicyObligations = true", this attribute speci response of the assertion or token.
customAmrClaim	N	If idpAmrAuthoritativeForPolicyObligations = true, this attribute specifi looked at in the response of the assertion or token.
customAmrMappings	N	This attribute defines the mapping between the custom Authentication Provider (IDP) and the corresponding AMR values used by the IDSP (1. The mapping is first checked for an exact match with the value of t is a direct match, the corresponding AMR value from the map is us property is not provided, the system checks the value of the "amr" AMR value from the map is applied. 2. If, after the above steps, there is still no match, the map is checke a "last-chance" AMR, providing a fallback option if no other mappi
idpAmrAuthoritativeForPolicyObligations		This attribute determines whether the Response or Assertion received identify the Authentication Method Reference (AMR) values. If this intr to the corresponding obligations associated with those AMR values be
defaultAmr		This attribute specifies the default AMR value(s) for IDP in CSV forma

idpClaimMapping	N	This attribute specifies the mapping between the attributes/claims retu
		attributes in the user profile in IDSP.
		Example:
		`[`
		`{`
		`"idpClaimName": "email",`
		`"idpClaimSource": "token",`
		`"profileLogicalAttrName": "user_loginid",`
		`"userData": true,`
		`"persistUserData": true`
		`},`
		`{`
		`"idpClaimName": "first_name",`
		`"idpClaimSource": "userinfo",`
		`"profileLogicalAttrName": "first_name",`
		`"userData": true,`
		`"persistUserData": true`
		`},`
		`...`
		`]`
		If this value is provided, it is in a json format.
		• idpClaimName: This is the name of the claim present in the token
		• idpClaimSource: This attribute is mandatory when idpType="openi
		following values:
		— "token": This indicates that the claim is present in the identity to
		— "userinfo": This specifies that the claim is present in the claims
		— "static": When set to "static," it signifies that the claim has a sta
		static value.
		• profileLogicalAttrName: This attribute represents the logical name
		IDP should be mapped.
		• userData: This boolean indicates whether the attribute is intended
		• persistUserData: This boolean is only applicable when "userData"=
		persisted.

autoInlineEnrollment	N	This attribute checks if auto inline enrollment of mandatory factors is e ***Default Value*** - false.

issuer	Y	This attribute specifies the issuer of the token in the response sent by
issuers	N	This attribute specifies the additional issuers associated with this Iden
status	-	The **status** attribute identifies if the status of an Identity Provider is ac **Possible Values:** active, inactive **Usage:** The status of an Identity Provider can be changed to active/ina IdentityProviders" API with the payload as follows: `{ "status": "active" }` **Usage Scenarios:** • If an Identity Provider is created using the ".../admin/v1/SAMLIdpM Provider is “inactive”. You can change the status to "active" using t • If an Identity Provider is created using the ".../admin/v1/IdentityPro determined based on the status attribute that is defined in its paylo with status attribute defined in the IdentityProviders API, see Creat If the status attribute is not defined in the IdentityProviders API pay
description		This attribute specifies the description of the Identity Provider.
zeroFootPrint		This attribute defines how the "zeroFootPrint" (ZFP) is interpreted in B virtualization of the application's ZFP behavior and can be configured • "true": If set to "true," it is assumed that there is no local footprint o • "false": If set to "false," it is assumed that there is a local footprint c • "none": When set to "none," the ZFP behavior is determined by the Additionally, this attribute can accept expressions that evaluate to "true For example: "${evaluateIsNotTrue.${idProfile.isPlanAdmin}}". This expression checks if the "idProfile" claim "isPlanAdmin" is not true evaluates to "false." Supported expression values include: • "evaluateIsTrue" • "evaluateIsNotTrue"
ignoreSSLValidation		This attribute checks if there is a requirement to bypass SSL validation flows in the context of back channel calls for tokens, userinfo, and the
sslVerifyCertAlias		This attribute specifies the alias for the certificate that should be used OpenID flows in the context of back channel calls for tokens, userinfo,
readTimeoutMillis		This attribute represents the time in seconds that will be used when w used for OpenID flows in the context of back channel calls for tokens, ***Default Value*** = "10000".
connectTimeoutMillis		This attribute represents the time in seconds that will be used when es IDSP (ssp). It is currently used exclusively for OpenID flows in the con the JSON Web Key Set (JWKS) URL. ***Default Value*** = "10000".
itDecryptionSource		This attribute specifies the source from which to obtain the encryption The possible values are: • "idp": This value specifies that the alias of the secret (of type rsake fetched and provided for encryption. • "tenant": This value indicates that the certificate corresponding to t encryption. • "none": When set to "none," no encryption certificate is provided fo

itDecryptionKeyAlias		This attribute is used when the "EncryptionTarget"="idp." It specifies the alias of the secret (of type rsakey), and the certificate c encryption.
acrAllowedInRequest		This attribute specifies whether the "acr_values" parameter should be type IdentityProvider. ***Default Value*** - false.
defaultAcrMappingInRequest		This attribute specifies the key, and the corresponding value associate parameter in the "/authorize" request made to the "openid" type Identit
acrRequestMapping		This attribute specifies the mapping of the key specified by the default sent as the "acr_values" parameter in the "/authorize" request to the "c
metadata		This attribute specifies a list of metadata name/value pairs that define See the IDP metadata properties table to know more about the suppor
signRequest	N	This attribute determines whether IDSP signs the SAML authentication • If set to `true` , the SAML authentication request is signed. • If set to `false` or not provided, the SAML authentication request i If set to true, set the `signingKeySource` and, optionally, the `sign`
signingKeySource	Yes, if signRequest=true	This attribute identifies from where the signing key originates. • If set to `tenant` , SAML authentication requests are signed using • If set to `idp` , SAML authentication requests are signed with the k Console (see Identity Providers). If set to `idp` , also set the `signingKeyAlias` .
signingKeyAlias	Yes, if signingKeySource=idp	This attribute defines the alias for the signing key.

IDP Metadata properties

Learn about Identity Provider metadata properties.

This table includes the **Identity Provider metadata** property descriptions:

IDP Metadata Attribute Name used in Admin Console	IDP Metadata Attribute Name used in API	Values
Custom query string parameters in login request (ex: login_hint=${loginid}&x=y	urn:iam:idp:queryparams:login	"<name>=<value>"
Do not show IDP on signin page (true/false)	urn:iam:idp:hideOnDiscovery	"true" / "false" (default is false)
Ignore URL parameters on AssertionConsumerService URL check (true/false)	urn:iam:idp:ignoreUrlParamsOnAcsCheck	"true" / "false"

Construct group context from groups in identity store and identity assertion (true/false)	urn:iam:enableIdentityAssertionGroupPropagation	"true" / "false" (default is false)
IDP supports forced reauthentication (prompt=login for OpenID or ForceAuthn="true" for SAML) (true/false)	urn:iam:idp:supportsPromptLogin	"true" / "false" (default is false)
IDP supports silent reauthentication (prompt=none for OpenID or IsPassive="true" for SAML) (true/false)	urn:iam:idp:supportsPromptNone	"true" / "false" (default is false)
Require login name resolved from IDP Account Mapping to match login name/hint (true/false)	urn:iam:idp:requireLoginNameMatch	"true" / "false" (default is false)
For BYOI and JWT-BEARER with ZFP, use IDP as Identity Source (true/false)	urn:iam:idp:zfpUsesIdpAsIdentitySource	"true" / "false" (default is false)
Skip the signing of SAML Authentication Request to be sent to the Identity Provider (true/false)	urn:iam:idp:skipSamlRequestSigning	"true" / "false" (default is false)
SAML Authentication Request NameID Format	urn:iam:idp:samlRequestNameidFormat	NameID format to use
Use cookie for state management (true/false)	urn:iam:idp:useCookieForState	"true" / "false" (default is false)

Security Method to be used when calling OpenID Provider /token endpoint	urn:iam:idp:oauth:securityMethod	• OAuth (this is the default) - IDP property "clientSecret" is a literal value. • oauthassertion - IDP property "clientSecret" is the alias for a Secret of type RSAKEY/ECKEY.
Integration Mode for this Identity Provider	urn:iam:idp:integrationMode	• shared (same as no value this is the default) • virtual-idp • virtual-app • specific-app-idp
Skip UI notice on redirect	urn:iam::idp:skipNoticeOnRedirect	"true" / "false" (default is false)
Require login name resolved from IDP Account Mapping to match IDP's domains (true/false)	urn:iam:idp:requireLoginNameDomainMatch	"true" / "false" (default is false)
Use regex such as '^(.*)@.*$' to extract the login name (resolved from IDP Account Mapping) to be matched against login name/hint. This is a regex containing a 'regex group' represented by subpattern enclosed within the parentheses (). The subpattern represents the login name to be compared.	urn:iam:idp:useRegexForLoginNameMatch	Regex containing a 'regex group' represented by sub-pattern enclosed within the parentheses ()
Time Skew toleration in seconds to be used when generating OAuth Client Assertion (Assertion lifetime will be 10 minutes +/- skew time	urn:iam:idp:oauth:assertion:skewTimeSecs	
Skip UI notice on redirect	urn:iam:idp:skipNoticeOnRedirect	"true" / "false" (default is false)

Construct group context from groups in identity store and identity assertion (true/false)	urn:iam:idp:enableIdentityAssertionGroupPropagation	"true" / "false" (default is false)
Allow Idp Not To Send InResponseTo	urn:iam:idp:allowIdpNotToSendInResponseTo	"true" / "false" (default is false)
Grace time period (+/-) to be used for validating time-instance claims in tokens received from IDP (seconds)	urn:iam:idp:oauthTokenGraceSecs	Integer representing the number of seconds
• Identity Source ID Override • Identity Source Name Override • Identity Source Type Override	• urn:iam:idp:identitySourceId • urn:iam:idp:identitySourceName • urn:iam:idp:identitySourceType	<string>
Skip auto unlock on successful authentication by the Identity Provider	urn:iam:idp:skipAutoUnlockOnAuthentication	"true" / "false" (default is false)
Strategy to use when merging multi-valued email, phone_number, address, and device claims from IDP with user profile attributes. No explicitly provided strategy defaults to User Profile being authoritative for the primary value	urn:iam:idp:idpClaimsMergeStrategy	• preferIDP • preferProfile
Use HTTP Proxy	urn:iam:idp:useHttpProxy	true/false (default)
Display name template	urn:iam:idp:displayNameTemplate	${idprofile.given_name} ${idprofile.family_name}

Skip signature in request body on HTTP redirect (true/false)	urn:iam:idp:skipSignatureInRequestBodyOnHttpRedirect	
Allow ZFP IDP to verify identity in IDP-as-a-Factor flows (true/false)	urn:iam:idp:allowZfpIdpToVerifyIdentity	true/false

Types of Integration Flows in Identity Provider

There are different partnership models based on which the Identity provider can be integrated with Symantec Identity Security Platform (Service Provider).

The partnership models supported by IDSP are:

- Shared/Default
- Virtual App
- Virtual IDP
- Specific App and IDP

Before exploring the Identity Provider flow, it is important to understand the following terms and their roles within the flow:

- **EntityID**: represents a unique and immutable ID of the relationship between a SAML SP and SAML IDP. This identifier is a string with the value being as generic as possible, free of special characters and assumptions known to cause various issues.
- **ACS Url**: The URL exposed by a SAML SP where SAML responses are received. While this URL can be managed by the SP, keeping it simple minimizes potential complications for SAML IDPs when storing and maintaining it.
- **Resource Scope**: Allows multiple IDPs to share the same Issuer and/or Domain. It enables Lines of Business (LoBs) to locate IDPs using a partitioning technique that separates the use of physical domains/issuers from the IDPs used by LoBs or applications.

In a typical Identity Provider flow, the following partners are involved:

- **Application** - This is the entity which initiates the Service Provider (SP) Init flow, and handles the response that comes back from the Identity Provider (IDP) (SP and IDP Init flows).
  - The Identity Provider is registered at the application side with the Issuer and login url provided by the Identity provider.
- **Identity Provider** - The identity provider is the entity that authenticates the user or delegates the authentication to an external Identity Provider.
  - The application is registered as a Service Provider application with the Identity Provider using the entity Id and assertion consumer service (ACS) URL provided by the application.
  - If the authentication is delegated to an external Identity provider, then external Identity provider is registered using the Issuer and login url provided
- **External Identity Provider** - The identity provider that is used to authenticate the users.
  - The Identity provider which delegates the authentication to the external authentication provider is registered as a Service Provider application with the external Identity Provider using the Entity Id and ACS Url that are provided by the delegating Identity Provider.

Based on the understanding the following Partnership models are identified when IDSP is used as an Identity Provider and delegating authentication to external Identity Providers.

Partnership Model	Supported Flows	Configuration Required
Shared/Default This is the default model.	Supports both SP and IDP init flows	**Configuration at IDSP's end** • One IDP configuration in IDSP corresponding to each ex **Configuration at External IDP's end** • One App definition representing each instance of IDSP, on Entity ID and ACS URL provided in IDSP's Service P metadata
Virtual App • Addresses "One IDP" use cases where the physical IDP such as Azure must be able to host properly differentiated app metadata	SP init flow	**Configuration at IDSP's end** • Multiple IDP definitions in IDSP corresponding to the sa IDP issuer, differentiated by unique resource scope. • One IDP configuration in IDSP corresponding to each ap the external IDP. • Each IDP configuration in IDSP has a unique Entity ID u IdentityProvider ID of the corresponding IDP definition. **Configuration at External IDP's end** • Multiple app definitions on external IDP corresponding t the same IDSP instance, differentiated based on the uni ID provided, representing unique IDP configuration on th side.
Virtual IDP • Addresses scenarios where a single IDP accommodates multiple apps	Supports both SP and IDP init flows	**Configuration at IDSP's end** • One IDP configuration in IDSP corresponding to an exte issuer. • The IDSP metadata depends on the app selected at the metadata generation, which provides a unique Entity ID URL corresponding to the selected IDSP application. **Configuration at External IDP's end** • Multiple App definitions on external IDP corresponding t the same IDSP instance, differentiated based on the uni ID provided, representing unique IDP configuration on th
Specific APP & IDP • Addresses "Service Provider" use cases with specific constraints such as all IDPs having the same issuer but different assertion validation certs	Supports both SP and IDP init flows	**Configuration at IDSP's end** • Multiple IDP definitions in IDSP corresponding to the sa IDP issuer, differentiated by unique resource scope. • One IDP configuration in IDSP corresponding to each ap the external IDP. • Each IDP configuration in IDSP has unique Entity ID an using the identityProvider ID of the corresponding IDP d selected IDSP application. **Configuration at External IDP's end** • Multiple App definitions on external IDP corresponding t the same IDSP instance, differentiated based on the uni ID provided, representing unique IDP configuration on th

Identity Provider Discovery - Just-In-Time (JIT) User Provisioning

When configuring Identity Provider Discovery for the purpose of setting up desired Bring-Your-Own-Identity (BYOI) flows, it is possible to use the Just-In-Time (JIT) User Provisioning feature to automatically onboard user identities into Symantec Identity Security Platform's internal identity store. This can be achieved by configuring JIT-specific properties on the Identity Provider resource.

The JIT process creates an "identity" footprint within IDSP's internal identity store, which is subjected to further administration by the admin, or through the self-service options, for example, to manage user profiles and group memberships. JIT identities can be configured as read-only, allowing updates solely from the Identity Provider as the authoritative source.

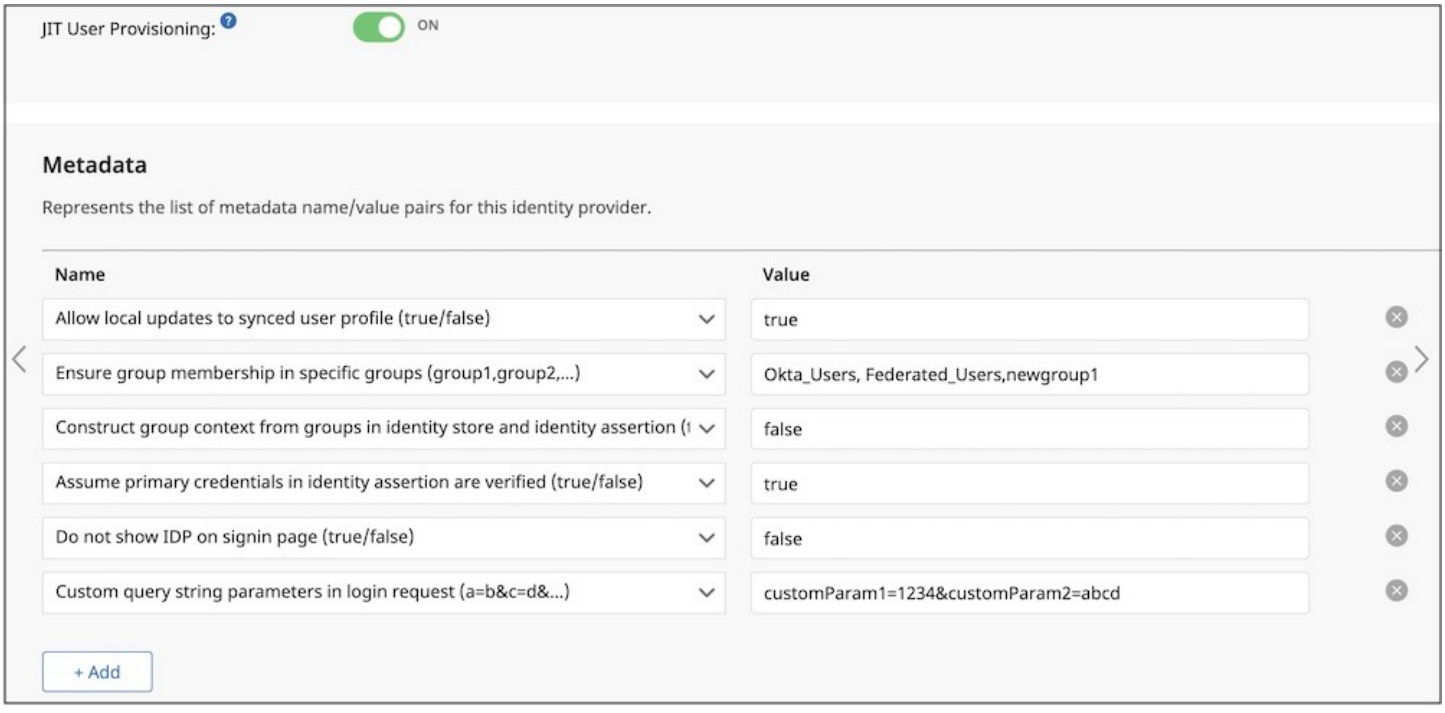
During JIT user sync, the user profile to be saved is constructed from the internal id store attributes mapped to by claims in the IDP identity assertion. Such claims must be defined as "***User Data***" and "***Persitable***".

User group memberships are synced depending on whether the groups are dynamic (present in the IDP identity assertion) or static (defined by the IDP urn:iam:idlcm:assignJitGroupMembership metadata). A group is created on the first membership grant, but not removed even if the last membership was revoked.

- Membership in Dynamic groups is synced if the IDP User Account Management has defined User Profile Attribute named "groups" marked as "Persistable User Data" claim sourced from a named Identity Provider claim. In this case, the user is added to groups that are present in the IDP assertion, and removed from groups associated with that IDP that were in the previous assertion, but not in the latest one. The named Identity Provider claim must be multi-valued - for OpenID IDPs this claim is a json array ([group1, group2]), and for SAML IDPs, this is a SAML Attribute with multiple AttributeValue elements) - see the following examples.
- Membership in Static groups is synced if the IDP metadata "urn:iam:idlcm:assignJitGroupMembership" defines a list of groups that the user should be a member of.

The following illustration shows the JIT Provisioning enablement with the list of metadata name-value pairs that are defined for an Identity Provider:

Figure 32: JIT User Provisioning - Metadata Name-Value Pairs for an Identity Provider



IDP metadata that is used to control JIT user provisioning

The following table includes the Identity Provider metadata that is used to control JIT provisioning behaviors:

Metadata Tag	Values	Requires JIT User Sync to enabled
urn:iam:idlcm:allowJitIdentityLocalUpdates	"true" / "false" (default is true)	yes
urn:iam:idlcm:assignJitGroupMembership	"group1, ..., groupN	yes
urn:iam:idlcm:saveJitPrimaryCredentialsAsVerified	"true" / "false" (default is true)	no

Display Name Generation in JIT Provisioning

While onboarding a user through JIT provisioning, Symantec Identity Security Platform ensures that the display name of the user is always set. Symantec Identity Security Platform generates the `displayName` value from one of the following sources (in order):

1. User Data claims mapping; or,
2. metadata of the identity provider; or,
3. metadata of the application; or,
4. `displayNameTemplate` configured in the internal identity store configuration (see Internal Identity Store Configuration); or,
5. `givenName` and `familyName` ; or,
6. user `loginId` .

Reference Links

For additional information, see:

- Identity Provider Discovery Policy
- Identity Provider as an authentication factor

Identity Provider SAML Initialization Flow

Learn how the Symantec Identity Security Platform handles different use cases involving RelayState mappings and redirections for service provider (SP) and identity provider (IDP) initiated flows

> **NOTE**
> Use of RelayState is optional.

This table outlines how the IDSP handles different use cases involving RelayState mappings and redirections for service provider (SP) and identity provider (IDP) initiated flows.

General Use Case	Endpoint	Action Taken by IDSP New RelayState (NRS)=
Use Case 1 Regular Service Provider (SP) Init	https://example.com/default/saml/v1/idp/login?[?RelayState=<rs-value>]	POST to App's ACS with **Example 1** - regular POS • App M/D: Not relevar • RelayStateMapping: • Request: .../saml/v1/i • Action: POST to App – The relay state pa
Use Case 2 Regular Identity Provider (IDP) Init	https://example.com/default/saml/v1/idp/login?appId=1234-4567-9087-1234[?RelayState=<rs-value>] AppId or EntityID can be used interchangeably as a value to denote the App.	Conditional Logic: 1. If NRS matches the p – Action: POST to A 2. Else: – Action: POST to A endif **Example 1 - Regular PO** • App M/D: Not relevar • App RelayStateMapp • Request: .../saml/v1/i • Action: POST to App **Example 2 - Regular PO** • App M/D: Not relevar • App RelayStateMapp • Request: .../saml/v1/i • Action: POST to App **Example 3 - Regular PO** • App M/D: Not relevar • App RelayStateMapp • Request: .../saml/v1/i • Action: POST to App

Use Case 3
Identity Provider (IDP) Init from External IDP
	https://example.com/default/saml/v1/sp/acs?sp=TestApp[&RelayState=<rs-value>]
	AppId, AppName, and EntityId can be used interchangeably as a value for the "sp" parameter to denote the App.
		For applications using re
		`saml/v1/sp/acs?sp`
		), where the mapped-to v
		`"urn:iam:redirec`
		`"urn:iam:app:rela`
		• To know the value of
		• The default value for
		• Replace the text {{tex
		For example:
		`https://{{host`
		`entityId>&Rela`
		should look similar to
		`https://exampl`
		`entityId>&Rela`
		Conditional Logic:
		1. If NRS matches the p
		– Action: 302 Redir
		2. Else if the (App's M/D
		– Action: 302 Redir
		3. Else:
		– Action: POST to A
		Example 1 - Behavior fo
		• App M/D: Not relevan
		• App RelayStateMapp
		• Request: .../saml/v1/s
		• Action: 302 Redirect
		Example 2- New Behav
		• App M/D: Not relevan
		• App RelayStateMapp
		• Request: .../saml/v1/s
		• Action: 302 Redirect
		Example 3- As-is Behav
		• App M/D: "urn:iam:ap
		• App RelayStateMapp
		• Request: .../saml/v1/s
		• Action: 302 Redirect
		Example 4- New Behav
		• App M/D: "urn:iam:ap
		• App RelayStateMapp
		• Request: .../saml/v1/s
		• Action: POST to App

Example of IDSP Application's IDP initiated RelayState Mappings

The following example demonstrates how the IDSP's Application uses IDP-initiated RelayState mappings to determine the final redirection URLs.

```
App.samlIdpInitiatedRelaystateMapping -
```

```
      {
 "default" : "urn:iam:redirectHint=https://example.com",
 "relaystate1": "urn:iam:redirectHint=https://example1.com",
 "relaystate2": "test123",
 "https://a\.b\.c/.*" : "urn:iam:redirectHint=https://homepage", //regex in key
 "https://myapp.co.com/customers/(.*$)": "urn:iam:redirectHint=https://www.google.com/search?q={1}" // regex in
  key and based on the match, actual value param is replaced {1}
 }
```

This table outlines various IDP scenarios, detailing the "From" and "To" values of RelayState mappings and the corresponding final redirections for each scenario, based on the RelayStateMapping shown in the previous example.

Scenario	**RelayState Mapped- From Value**	RelayState Mapped- To Value
IDP Init Flow from external IDP → IDSP <IDP-Init-URL-For-External-IDP>?**RelayState=abcd**	abcd	**app.samlIdpInitiatedRelaystateMappi** No mapping found
IDP Init Flow from external IDP → IDSP <IDP-Init-URL-For-External-IDP>?**RelayState=default**	default	app.samlIdpInitiatedRelaystateMapping. **urn:iam:redirectHint=** https://example.c searched for and flow is redirected to the
IDP init from IDSP to the App /saml/v1/idp/login?**RelayState=default**	default	app.samlIdpInitiatedRelaystateMapping. **urn:iam:redirectHint**=https://example.c
IDP Init Flow from external IDP → IDSP <IDP-Init-URL-For-External-IDP> **RelayState=https://a.b.c/xyz**	https://a.b.c/xyz	app.samlIdpInitiatedRelaystateMapping. **urn:iam:redirectHint**=https://homepage redirection takes the user to the → https://homepage
IDP init from IDSP to the App /saml/v1/idp/login?entityId=&**RelayState=https://example2.com**	https://example2.com	app.samlIdpInitiatedRelaystateMapping. No mapping found, then use the value o flow is redirected to app ACS url
IDP Init Flow from external IDP → IDSP <IDP-Init-URL-For-External-IDP> **RelayState=https://myapp.co.com/customers/test123**	https://myapp.co.com/customers/test123	app.samlIdpInitiatedRelaystateMapping. customers/test123") → urn:iam:redirectH search?q={1} - Regex matched
IDP init from IDSP to the App <IDSP_BASE_URL>/saml/v1/idp/login?appId=**1234**	No RelayState	No Relay state provided
IDP Init Flow from external IDP → IDSP <IDP-Init-URL-For-External-IDP>	No RelayState provided	app.samlIdpInitiatedRelaystateMapping. **urn:iam:redirectHint=https://example.** "default" is searched for and flow is redi
IDP Init Flow from external IDP → IDSP <IDP-Init-URL-For-External-IDP>?**RelayState=relaystate1**	relaystate1	app.samlIdpInitiatedRelaystateMapping. **urn:iam:redirectHint=** https://example1 searched for and flow is redirected to the
IDP init fromIDSP to the App /saml/v1/idp/login?entityId=abcd&**RelayState=relaystate1**	relaystate1	app.samlIdpInitiatedRelaystateMapping. **urn:iam:redirectHint=** https://example1
IDP Init Flow from external IDP → IDSP <IDP-Init-URL-For-External-IDP>?**RelayState=relaystate2**	relaystate2	app.samlIdpInitiatedRelaystateMapping. test123

Managing Tenants

Learn how to create, configure, and manage tenants in IDSP, including tenant routing, CRUD operations, and tenant locking capabilities.

This topic includes the following:

- Resolving Tenancy
- Tenant CRUD Operation
- Creating a Tenant
- Updating a Tenant
- Tenant Locker API
- Error Codes

Resolving Tenancy

An incoming request can indicate its tenancy is using one of the following supported ways:

- Specifying the tenant in the URL path
- Using a sub-domain of the main IDSP FQDN
- Using a tenant's vanity URL

The general process of resolving a tenancy is as follows:

1. If the host matches the <service-provider-fqdn> domain, then the tenant gets taken from the API Path. This can be tenant-name or tenant-identifier.
2. If the host is a subdomain of the <service-provider-fqdn> domain, then the tenant gets taken from the subdomain. This can be tenant-name or tenant-identifier.
3. If the host matched neither the domain nor subdomain, the host is assumed to be the vanity host, and the tenant is searched for by the vanity host.

You can follow four different ways to access a tenant-specific IDSP service as explained in the following sections.

Common Service Provider URL

In this approach, a single and common service provider URL is used for all tenants, and the tenant name is part of the URL path.

Format of the Common Service Provider URL:

```
https://<service-provider-fqdn>/<tenant-name-or-identifier>/<functional-area>/v1/<api>
```

Where,

- **service-provider-fqdn:** Specifies a common service provider URL.
- **tenant-name:** Specifies a unique tenant name.
- **functional-area:** Specifies a service that IDSP deploys.
- **api:** Specifies a resource in the deployed service.

Vanity URL

A vanity URL is typically used in consumer-centric deployments to ensure that you, as a customer, can see the URL namespace of the business entity that you interact with, rather than the URL namespace of the service provider running the IDSP service.

Format of the Vanity URL:

```
https://<vanity-fqdn>/common/<functional-area>/v1/<api>
```

Where,

- **vanity-fqdn:** The vanity-fqdn belongs to you (customer). For example- "login.example.com". A vanity host fqdn is configured in DNS to point to the actual tenant identifier using a friendly URL or permanent URL approach.

  NOTE
  Vanity host must not be a subdomain of the <service-provider-fqdn> domain.
- **common**: This is the tenant-name placeholder that is used with vanity hosts.
- **functional-area:** Specifies a service that IDSP deploys.
- **api:** Specifies a resource in the deployed service.

  NOTE

  - To expose specific operations on the vanity host, see Partitioning Ingress..
  - To know more about managing vanity URLs, see Configuring a Vanity URL for a Tenant in IDSP.

Friendly URL

A friendly URL is typically used by service providers to support multiple business tenants.

Format of the Friendly URL:

```
https://<tenant-name>.<service-provider-fqdn>/common/<functional-area>/v1/<api>
```

Where,

- **friendly-tenant-name:** Specifies a friendly tenant name such as "example" which is a unique tenant name in IDSP. You can change this name as appropriate.
- **service-provider-fqdn:** Specifies a friendly URL that is typically used by service providers to support multiple business tenants.
- **functional-area:** Specifies a service that IDSP deploys.
- **api:** Specifies a resource in the deployed service.

Permanent URL

A permanent URL is typically used by long-running back-channel programmatic integrations. This URL enables you to change the friendly tenant name without changing the permanent URL.

Format of the Permanent URL:

```
https://<tenant-identifier>.<service-provider-fqdn>/common/<functional-area>/vi/<api>
```

Where,

- **tenant-identifier:** Specifies a permanent tenant ID, which is a Globally Unique Identifier (GUID) that cannot be changed.
- **service-provider-fqdn:** Specifies a friendly URL that is typically used by service providers to support multiple business tenants.
- **functional-area:** Specifies a service that IDSP deploys.
- **api:** Specifies a resource in the deployed service.

APIs to Perform CRUD operations on Tenants

As tenant management is a system administration task, ensure that you acquire the ***System Admin token*** before performing tenant management operations.

The following table displays a list of APIs that can be used to perform CRUD operations on tenants in IDSP:

Operation Description	Operation Type	Endpoint URLs
Create a tenant	POST	https://${SSP_MGMT_FQDN}/system/admin/v1/Tenants
Retrieve all configured tenants	GET	https://${SSP_MGMT_FQDN}/system/admin/v1/Tenants

Retrieve a tenant by specifying the tenant Id	GET	https://${SSP_MGMT_FQDN}/system/admin/v1/Tenants/<REPLACE-TENANT-ID>
Delete a tenant by specifying the tenant Id	DELETE	https://${SSP_MGMT_FQDN}/system/admin/v1/Tenants/<REPLACE-TENANT-ID>
Update a tenant by specifying the tenant Id	PATCH	https://${SSP_MGMT_FQDN}/system/admin/v1/Tenants/<REPLACE-TENANT-ID>
Retrieve the status of the tenant, whether the tenant is locked or unlocked.	GET	https://${SSP_FQDN}/<tenantName>/admin/v1/TenantLocker
Allows the tenant admin to lock or unlock a tenant by setting the attribute "locker" to true or false.	POST	https://${SSP_FQDN}/<tenantName>/admin/v1/TenantLocker

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/TenantLocker`
  should look similar to:
  `https://example.com/default/admin/v1/TenantLocker`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Creating a Tenant in IDSP

Use the following POST API to create a tenant in IDSP.

```
https://${SSP_MGMT_FQDN}/system/admin/v1/Tenants
```

The following API request payload is used to create a tenant in IDSP.

```
{
      "tenantName":"<tenant name>",
      "masterKey":"<key>",
      "demoSetup": false,
      "defaultClientSetup" : true,
      "defaultUserSetup" : true,
      "vanityHosts": ["<vanity-fqdn>"]
}
```

Where,

- **tenantName:** Specifies a unique name of the tenant to be added to IDSP.
- **masterKey:** (Optional) Specifies a secret key that is used to encrypt tenant data. If the master key is not provided, the system generates a random key.

  NOTE
  The secret key that is used to encrypt tenant data can be specified only once, and cannot be changed after a tenant is created.
- **demoSetup:** Creates a demo client for the newly created tenant. This demo client is used by the Sample App chart and by the demo example in the Postman collection.

Possible Values: true, false
Default: false

- **defaultClientSetup:** Creates a default client in the newly created tenant.
  Possible Values: true, false
  Default: false
- **defaultUserSetup**: When this is configured as true, an admin user is created for the new tenant with a random password (returned in the response payload) that must change on the next login. This user gets disabled after 48 hours. Note that the admin user is auto-seeded for the default tenant. Before the 48 hours window expires, configure an alternate administrative access. For more information, see Setting Up Your Admin Identity and Granting administrative privileges to the Tenant Admin Role sections. To know more about demo client, see Administrative Model, (search for Demo Client).
  Possible Values: true, false
  Default: false
- **vanityHosts:** (Optional) Specifies multiple vanity hosts that are exposed by an ingress controller. For example, login.ssp.com

  NOTE
  When configuring a tenant with multiple vanity hosts, we recommend using the 'vanityHosts' attribute rather than the 'vanityHost' attribute. If both the 'vanityHosts' and 'vanityHost' attributes are set, the combined list of vanity hosts is used for tenant identification.

  For the vanity host support, see Partitioning Ingress.

The following payload shows a sample request to create a tenant called "broadcom".

```
{
  "tenantName":"broadcom",
  "masterKey":"firewall",
  "demoSetup": true,
  "defaultClientSetup" : true,
  "defaultUserSetup" : true,
  "vanityHosts": ["nbruce-1224-ssp1.broadcom.com",
                  "nbruce-1224-ssp2.broadcom.com"]
}
```

The following payload shows a sample response to the "broadcom" tenant creation request:

```
{
    "tenant": {
       "tenantName": "broadcom",
       "tenantId": "cdd750b4-96a0-4b80-abd6-3cd3481c7dc0",
       "masterKey":"firewall",
       "vanityHosts": ["nbruce-1224-ssp1.broadcom.com",
                       "nbruce-1224-ssp2.broadcom.com"]
    "clients": {
          "DefaultTenantAdminUser": {
           "UserLoginId": "DefaultTenantAdminUser",
           "userPassword": "Njgbq101#"
       },
          "DemoClient": {
           "clientId": "bbb9eff1-34e0-4c4d-a688-f6e09a38a4f1",
           "clientSecret": "0b318282-4a58-4d00-810f-e886c964d70e"
       },
       "DefaultTenantClient": {
           "clientId": "af9e09fd-32ac-40e6-9290-0eecb5fce173",
```

```
            "clientSecret": "a46ceeac-3310-4c2e-adcd-35b925261418"
        }
    }
}
```

Updating a Tenant in IDSP

Use the following PATCH API to create a tenant in IDSP.

```
https://${SSP_MGMT_FQDN}/system/admin/v1/Tenants
```

The following code snippet shows the request payload to update a tenant. Here, only the status, vanityHost, and vanityHosts can be updated.

```
{
   "status":"active"
   "vanityHosts": ["nbruce-1224-ssp3.broadcom.com",
                   "nbruce-1224-ssp4.broadcom.com"]
}
```

Where,

- **status:** (Optional) Specifies the status of the tenant.
  Possible Values: active, inactive, disable
- **vanityHosts:** Specifies multiple <vanity-fqdn> hosts of a tenant that are mapped to the ingress controller by exposing the vanity history.
  The following payload shows the updated tenant sample response:

```
    {
        "tenantName": "broadcom",
        "tenantId": "cdd750b4-96a0-4b80-abd6-3cd3481c7dc0",
        "masterKey": "firewall",
        "status": "active",
        "vanityHosts": ["nbruce-1224-ssp3.broadcom.com",
                        "nbruce-1224-ssp4.broadcom.com"]
    }
```

TenantLocker API

The tenant can be locked by the tenant admin by setting the attribute **locked** to **true.** Since a system admin can perform cross-tenant operations, the locked attribute can be modified by the system admin, if necessary.

When a tenant is in the locked state, no tenant-level operations can be performed and the tenant will not be accessible to other clients. If this attribute is set to **false,** then the tenant is in the active state and any clients can access the tenant and can perform allowed operations.

Example:

```
{
"locked" : true
}
```

> **NOTE**
> When the status of the tenant is ***"inactive"***:

- The "/Token" endpoint can still generate an Access Token for the tenant Admin with the scope "TenantLocker".
- The "/TokenLocker" endpoint can still be used by the tenant admin to unlock the tenant.
- All other tenant-level operations are blocked with the error "Tenant is disabled".
- System Admin can acquire a Cross Tenant Access Token and can perform all the allowed operations for the inactive tenant.

When the status of the tenant is ***"disable" or the "Delete" operation is in progress***:

- The Tenant Admin or any clients in the tenant are forbidden from performing any of the tenant-level operations.
- The System Admin can use the PATCH "/tenant" endpoint to update the status of the tenant.
- The System Admin is not allowed to acquire a cross-tenant Access Token.

Error Codes

0000053	Tenant with the given name exists
0000054	Invalid status
0000055	Tenant name cannot be changed.
0000056	Missing tenant name
0000057	Tenant name contains invalid characters. Only Alphanumeric and '-' are allowed.
0000058	Tenant name is a reserved one. Try with the different tenant name.
0000059	Tenant with the given vanity host exists.

Managing Service Accounts

You can use the Service Account APIs to manage service accounts.

Service accounts are specialized accounts for service level users such as trusted clients. Service accounts are not subject to the same limits that interactive users are for authentication session lifetime and credential lifetime.

As an end user, use the Service Account APIs to view, validate, lock, or unlock your service accounts. As an administrator, use the Service Account APIs to view and validate service accounts for your users who are marked as service accounts.

See the appropriate resource:

Table 116: Service Account APIs

Operation	API Resource
View and validate your service accounts as an end user	MeServiceAccounts API
Lock or unlock your service accounts as an end user	MeServiceAccountsLocker API
As an administrator, view and validate service accounts for your users who are marked as service accounts	ServiceAccounts API

MeServiceAccounts API

Use the MeServiceAccounts API to view and validate your service accounts.

As an end user, you can use the MeServiceAccounts API to view your accounts that are marked as service accounts. You can also use the MeServiceAccounts API to verify (validate) that specified service accounts are assigned to you.

Service Account Management (User)	GET POST	`https://${SSP_FQDN}/{{tenantName}}/identity/v1/MeServiceAccounts`
	GET	`GET /identity/v1/MeServiceAccounts` Returns details about all service accounts that are assigned to you. Use filters (such as attributes, operators, values) to narrow your search. See REST API Filter.
		Optional GET parameter: `countonly` Returns only the number of service accounts that are assigned to you (and that match the filter criteria, if sent)

	POST	`POST /identity/v1/MeServiceAccounts`
		`{`
		`"accounts": [`
		`"{{USER-ID-OR-NAME1}}”,`
		`"{{USER-ID-OR-NAME2}}”`
		`]`
		`}`
		Returns one of the following responses:
		• `"allServiceAccountsManaged": true` : All of the service accounts that are included in the request are assigned to you
		• `"allServiceAccountsManaged": false` : None of the service accounts that are included in the request are assigned to you

		Optional POST parameter: `showInvalidAccounts` Response includes the `accountsNotFound` entry indicating whether any of the service accounts that are included in the request are invalid (non-existent).

MeServiceAccountsLocker API

Use the MeServiceAccountsLocker API to view and manage the lock or unlock status of your service accounts.

Use the MeServiceAccountsLocker API as an end user to view the lock status of any service accounts assigned to you. You can also use the MeServiceAccountsLocker API to change the lock status.

Service Account Lock Management (User)	GET POST	`https://${SSP_FQDN}/{{tenantName}}/identity/v1/MeServiceAccountLocker`
	GET	`GET /identity/v1/MeServiceAccountLocker/{{SERVICE_ACCOUNT-USERID-OR-USERLOGINID}}` Identifies if the service account that is provided in the request is locked or unlocked: • `"locked": true` : The service account is locked • `"locked": false` : The service account is unlocked

	POST	`POST /identity/v1/MeServiceAccountLocker/{{SERVICE_ACCOUNT-USERID-OR-USERLOGINID}}`
		`{`
		`"locked":{{true or false}}`
		`}`
		Sets the requested lock state:
		• If set to `true` , the service account is locked
		• If set to `false` , the service account is set to unlocked

ServiceAccounts API

Use the ServiceAccounts API as an administrator to view and validate service accounts for users that you manage.

As an administrator, you can use the ServiceAccounts API to view details about accounts that are marked as service accounts for your users. You can also use the ServiceAccounts API to verify (validate) that specified service accounts are assigned to a specified user.

Service Account Management (Admin)	GET POST	`https://${SSP_FQDN}/{{tenantName}}/identity/v1/ServiceAccounts`
	GET	`GET /identity/v1/ServiceAccounts{{USERID-OR-NAME}}` Returns details about all service accounts that are assigned to the specified user. Use filters (such as attributes, operators, values) to narrow your search. See REST API Filter.
		Optional GET parameter: `countonly` Returns only the number of service accounts that are assigned to you (and that match the filter criteria, if sent)

	POST	`POST /identity/v1/ServiceAccounts`
		`{`
		`"user": "{{USER-ID-OR-NAME}}",`
		`"accounts": [`
		`"{{USER-ID-OR-NAME1}}”,`
		`"{{USER-ID-OR-NAME2}}”`
		`]`
		`}`
		Returns one of the following responses:
		• `"allServiceAccountsManaged": true` : All of the service accounts that are included in the request are assigned to the specified user
		• `"allServiceAccountsManaged": false` : None of the service accounts that are included in the request are assigned to the specified user

		Optional POST parameter: `showInvalidAccounts` Response includes the `accountsNotFound` entry indicating whether any of the service accounts that are included in the request are invalid (non-existent).

Managing Token Revocation

Learn how to revoke user sessions and security artifacts using the Token Revocation API, which allows tenant administrators to revoke access tokens, refresh tokens, and identity tokens by Session ID (SID).

When SSO Session Management is enabled, security artifacts issued in the context of a user session such as identity tokens, access tokens, refresh tokens, are associated with the Session ID. When a specific Session ID is revoked via admin-based or self service Session Management API (/Sessions, /RefreshTokens), all security artifacts associated with that Session ID become revoked immediately. When SSO Session Management is disabled, use the "`.../admin/v1/revoke`" API to revoke security artifacts by Session ID.

The Token Revocation API is used by the tenant admin to revoke user sessions by SID (session Id). Multiple user sessions can be revoked at the same time. Session identifier can be taken from the Access Token or Identity Token from the 'sid' claim. Currently, only SID type entities are allowed to revoke.

> **NOTE**
> This API requires an Access Token with the scope 't.session'. The 't.session' scope is granted to the 'TenantAdmin' role by default.

Revoke API endpoint:

```
POST https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/revoke
```

> **NOTE**
>
> - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
> - The default value for `<tenantName>` is "default".
> - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
>
> For example:
>
> `https://{{host}}/{{tenantName}}/admin/v1/revoke`
>
> should look similar to:
>
> `https://example.com/default/admin/v1/revoke`
> - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Sample of Token Revocation API request:

```
curl --location --request PATCH 'https://example.com/default/admin/v1/revoke' \

--header 'Content-Type: application/json' \

--header 'x-tenant-name: default' \

--header 'Authorization: Basic
 PHBhc3RlIGNsaWVudElkIGZyb20gYWJvdmUgY29tbWFuZD46PHBhc3RlIGNsaWVudFNlY3JldCBmcm9tIGFib3ZlIGNvbW1hbmQ+' \

--data-raw '[ { "idType": "SID", "id": "a" }'
          [
    {
        "idType": "SID",
        "id": "a"
    },
    {
        "idType": "SID",
        "id": ""
    },
```

```
    {
        "idType": "SIDs",
        "id": "d"
    },
    {
        "idType": "",
        "id": ""
    }
]
```

Sample of Token Revocation API response:

```
[
    {
        "idType": "SID",
        "id": "a",
        "message": "Revoked successfully"
    },
    {
        "idType": "SID",
        "id": "",
        "error": "Missing id"
    },
    {
        "idType": "SIDs",
        "id": "d",
        "error": "Invalid idType"
    },
    {
        "idType": "",
        "id": "",
        "error": "Missing idType"
    }
]
```

Managing Scoping Constraints

Learn about performing grant and revoke operations on Scoping Constraints.

About Scoping Constraints

Using the Scoping Constraints, you can effectively manage administrative permissions for granting access to specific attributes of a resource. The Scoping Constraint allows the definition of the permissions that are allowed on resource attributes. For example, you can define the set of attribute permissions that are allowed on a user. The resource attributes permissions are defined in one or more Scoping Constraints. For more details about scoping constraints, see Defining Scoping Constraints. These constraints can be assigned and revoked from Admin roles as needed.

Essentially, a role can be granted to a principal with zero or more security constraints, similar to providing access to a specific folder with Viewer, Contributor, or Editor permissions.

Using the POST ".../admin/v1/AdminScopeConstrainer" API constraints can be added to a specific Authorization Policy. These constraints are then published in the "result" section of the Authorization Policy. Similarly, a scope constraint can be revoked from an admin role using the DELETE ".../admin/v1/AdminScopeConstrainer" API

NOTE

- Administrators with the API scope "urn:iam:t.adminroles" are granted access to manage the scoping constraints resource.
- Granting access without scoping constraints indicates that no restrictions are placed on the operation.
- Scoping Constraints when added to an Authorization policy gets reflected in the User Access Token. See Access Token section.

The following section includes Scoping Constraints APIs, supported CRUD operations along with examples.

APIs to Grant and Revoke Scoping Constraints

The following table includes the APIs to grant and revoke Scoping Constraints on Authorization Policies in Symantec Identity Security Platform:

Operation Description	Admin Scope	Operation Type
Use this API to add scoping constraints to an Authorization Policy	urn:iam:t.adminroles	POST
Use this API to revoke constraints from an Authorization Policy	urn:iam:t.adminroles	DELETE

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/AdminScopeConstrainer`
  should look similar to:
  `https://example.com/default/admin/v1/AdminScopeConstrainer`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Examples of Using Scoping Constraints APIs in Authorization Policy

Adding Constraints to an Admin Role

The following example shows how to add a constraint to a ***Admin Role***.

Perform a POST request using the following admin API:

```
POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/AdminScopeConstrainer
```

API Request:

```
{
    "id": "ed696a27-9f2f-47d7-bb25-1d6d5677c985",
    "constraints": [
    "id596a27-1m3f-53d7-ab25-1a6e5677c261"
    ]
    }
```

Where,

- ***id*** - This is the Authorization Policy ID to which the scoping constraint must be added.
- ***constraints -*** This is the constraint ID that is returned by the GET https://{{sspHost}}/{{tenantName}}/admin/v1/ScopingConstraints API. See Defining Scoping Constraints.

Deleting Constraints from an Admin Role

The following example shows how to delete a constraint from an ***Admin Roles***.

Perform a DELETE request using the following Authorization API:

```
https://${SSP_FQDN}/{{tenantName}}/admin/v1/AdminScopeConstrainer
```

API Request:

```
{
"id": "ec9bac96-aee1-4848-b9d2-fee79dd0ba90",
"constraints": [
"id596a27-1m3f-53d7-ab25-1a6e5677c261"
]
}
```

Where,

- ***id*** - is the Authorization Policy ID from where the scoping constraint must be deleted.
- ***constraints*** - is the constraint ID that is returned by the GET https://{{sspHost}}/{{tenantName}}/admin/v1/ScopingConstraints API. See Defining Scoping Constraints.

Performing a GET operation of a specific Authorization Policy to view the newly added constraints as part of the "result" section in that policy.

The following example shows how the **scoping constraint gets added** to the specified Authorization policy in the "**result**" section of the **API response**.

Perform a GET request using the following Authorization API:

```
https://${SSP_FQDN}/{{tenantName}}/admin/v1/AuthZPolicies/{{authZPolicyID}}
```

API Response:

```
{
        "policyId": "b82e9902-15dc-4aeb-858d-a838129ed073",
        "policyName": "IdentityAdminPolicy",
        "description": "Manage Users",
        "status": "active",
        "isGrantableClientOnly": false,
        "displayName": "Identity Administrator",
        "apps": [
            {
                "name": "SSP",
                "id": "$SSP_APP"
            }
        ],
        "rules": [
            {
                "id": "f048579a-4da6-4593-a0b2-85814d5929d7",
                "conditions": {
                    "principal": {
                        "group": {
                            "operator": "in",
                            "value": [
                                "internal/Helpdesk Admins",
                                "internal/HR Admins
                            ]
                        }
                    }
                },
                "result": {
```

```
                    "msg": "This is Identity Admin authorization policy",
                    "privileges": [
                        "urn:iam:t.users",
                        "urn:iam:t.userdata"
                    ],
                     "constraints": [
                      "33086569-1bdf-4c10-8eec-7edb7ab88121",
                      "44123183-8caf-2c10-7eec-3edb1ab28130"
                   ]
                 }
             }
         ]
}
```

Managing Sessions

Learn about Session APIs that can be invoked to manage user sessions.

In Symantec Identity Security Platform, both administrators and users have the capability to manage user sessions. This functionality is facilitated through a set of admin and user APIs, detailed in the following section. Additionally, various settings have been introduced to enhance session management capabilities.

Session Settings

The following settings have been added as part of the Session management:

- ***SSO Mode***: Support for Cross Domain Single Sign-on (SSO).
- ***SSO Maximum Expiry***: Support for Max Session Expiry for applications integrating with the solution.
- ***SSO Idle Expiry***: Support for Idle Session Expiry for applications integrating with the solution.
- ***Service Account session enabled***: Enforce SSO session requirements for service accounts.
- ***SSO Maximum User Sessions for Service Accounts***: Support for Max User Sessions for service accounts.
- ***SSO Maximum Expiry for Service Accounts***: Support for Max Session Expiry for service accounts.
- ***Admin Console SSO Maximum Expiry***: Support for Max Session Expiry for the Admin Console.
- ***Admin Console SSO Idle Expiry***: Support for Idle Session Expiry for the Admin Console.

These settings are covered in the admin console section and in the settings section. See admin Console and settings.

Session Management Admin API Scope

The scope of the Session Management Admin API is "**urn:iam:t.session**".

Session Management Admin APIs

Operation Description	Operation Type	Endpoint URLs	
Revoke Sessions and Refresh Tokens by UserID or LoginId	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}/admin/v1/UsersSessionsAndRefreshTokens/{userIdOrLoginId}	
Get Sessions by UserID or LoginId	GET	https://${IDSP_FQDN}/{{apiPathTenant}}/admin/v1/UsersSessions/{userIdOrLoginId}	
Revoke Sessions by UserID or LoginId	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}/admin/v1/UsersSessions/{userIdOrLoginId}	

Operation Description	Operation Type	Endpoint URLs	
Get Session by Session ID	GET	https://${IDSP_FQDN}/{{apiPathTenant}}/admin/v1/Sessions/{sid}	
Revoke Sessions by Session ID	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}/admin/v1/Sessions/{sid}	
Get Refresh Tokens by UserID or LoginId	GET	https://${IDSP_FQDN}/{{apiPathTenant}}/admin/v1/UsersRefreshTokens/{userIdOrLoginId}	
Revoke Refresh Tokens by UserID or LoginId	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}/admin/v1/UsersRefreshTokens/{userIdOrLoginId}	
Get Refresh Tokens by Session ID	GET	https://${IDSP_FQDN}/{{apiPathTenant}}/admin/v1/RefreshTokens/{sid}	
Revoke Refresh Tokens by Session ID	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}/admin/v1/RefreshTokens/{sid}	

NOTE

- To know the value of ${IDSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/UsersSessionsAndRefreshTokens/{userIdOrLoginId}`
  should look similar to:
  `https://example.com/default/admin/v1/UsersSessionsAndRefreshTokens/{userIdOrLoginId}`
- To view Session Management API in Swagger format, see here.

Session Management Self-Service API Scope

The scope of the Session Management Self-Service API is "**urn:iam:m.me**".

Session Management Self-Service APIs

The following table lists the user APIs to manage their own sessions. These APIs have also been covered in the Self Service API section.

Operation Description	Operation Type	
Revoking my Sessions and Refresh Tokens	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}/auth/v1/MeSessionsAndRefreshTokens
Get my Sessions	GET	https://${IDSP_FQDN}/{{apiPathTenant}}/auth/v1/MeSessions
Revoking my Sessions	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}/auth/v1/MeSessions
Get My Session by Session ID	GET	https://${IDSP_FQDN}/{{apiPathTenant}}/auth/v1/MeSessions/{sid}
Revoke My Session by Session ID	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}/auth/v1/MeSessions/{sid}

Operation Description	Operation Type	
Get my Refresh Tokens	GET	https://${IDSP_FQDN}/{{apiPathTenant}}/auth/v1/MeRefreshTokens
Revoke my Refresh Tokens	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}/auth/v1/MeRefreshTokens
Get My Session by Session ID	GET	https://${IDSP_FQDN}/{{apiPathTenant}}/auth/v1/MeRefreshTokens/{sid}
Revoke My Session by Session ID	DELETE	https://${IDSP_FQDN}/{{apiPathTenant}}/auth/v1/MeRefreshTokens/{sid}

Authentication Policies Evaluator API

Learn about the policy evaluator API.

This API evaluates the authentication context (policy rules) from the request payload, and returns the details of all matching policies along with a consolidated result. The result of this evaluation can be either allow or deny. If allowed, it also specifies the obligations, the user must meet to successfully authenticate.

Authentication Policies Evaluator Admin API

Admin Scope	Operation Type	Endpoint URLs
urn:iam:t.authpolicies	POST	https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/AuthPoliciesEvaluator

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/AuthPoliciesEvaluator`
  should look similar to:
  `https://example.com/default/admin/v1/AuthPoliciesEvaluator`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Request and Response Samples

Perform a POST request using the following admin API:

```
POST https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/AuthPoliciesEvaluator
```

API Request

This request shows how an internal/{{userID}} can be used in this API as an input to evaluate the obligations that the user must execute.

```
{
    "action": "authenticate",
    "user":"internal/{{userID}}",
    "app": "DefaultTenantClient",
    "ipAddress":"1.64.0.0",
    "channel": "web"
}
```

Where,

- **action** - The type of action through which the authentication is initiated. For example, authenticate login.
- **user** - Log in id of the user involved in the authentication process.
- ***app* -** Name of the application initiating the authenticate call.
- **ipAddress** - IP Address from where the authentication is initiated.
- **channel** - Type of channel from which the authenticate call is initiated.

The following items can also be added as part of the request:

- **group** - List of user groups.
- **idpName** - The idpName indicates the logical name representing the identity namespace of the user in a policy condition.
- **acr** - Specifies the ACR Values within the authentication request that the Authorization server is requested to use for the processing of requests from the client.
- **country** - Country from where the authentication is initiated.
- **triggeredRiskRule** - Multiple risk rules could be triggered under this rule condition for a user that must be evaluated. For example, User Velocity and Risky Country.
- **type** - type of policy, Authentication, Authorization, or IDP discovery flow.
- **identitySourceId** - Id of the identity source for the user.
- **clientContext** - Additional context provided by the client to be used while evaluating the policy.
- **idTokenHint** - encrypted idToken Hint that is sent by the federated application.
- **allConditionsMustMatch** - Indicates if all the conditions must match.
- **skipUserDisambiguation** - Indicates whether to skip the user disambiguation while evaluating the policy.
- **riskThreshold** - Risk score of the user.

API Response

The following response displays the outcome of evaluating two matching policies.


```
{
    "effect": "allow",
    "reAuthenticate": "false",
    "mfaFrequency": "EveryTime",
    "authLevel": 0,
    "policyLoadTime": "2024-07-30 06:04:09 GMT",
    "sortedObligations": "[[PASSWORD:1, SMSOTP:2]]",
    "matchedPoliciesAndRules": [
        {
            "effect": "allow",
            "obligation": [
                "PASSWORD:1"
            ],
            "policyName": "AdminLoginPolicy",
            "reAuthenticate": "false",
            "msg": "Auto provisioned login rule for DefaultTenantAdminUser",
            "mfaFrequency": "EveryTime",
            "authLevel": 0,
            "policyLoadTime": "2024-07-30 06:04:09 GMT",
            "channelMatch": "true",
            "allConditionsMatched": "true"
        },
        {
            "effect": "allow",
            "obligation": [
```

```
                "SMSOTP:2"
            ],
            "policyName": "test policy eval api",
            "reAuthenticate": "false",
            "msg": "test internal user",
            "mfaFrequency": "OnceForTrustedDevice",
            "authLevel": 0,
            "policyLoadTime": "2024-07-30 06:04:09 GMT",
            "allConditionsMatched": "true"
        }
    ]
}
```

Where,

- **effect** - Shows the effect of policy whether it results in an allow or a deny.
- **reAuthenticate** -
  Confirms if the user who has requested access to a secured resource is the same user who has authenticated at session initiali
- **mfaFrequency** - Multi-
  Factor Authentication (MFA) frequency in the Policy "result" is used to enable the trusted device option.
- **authLevel** - An optional attribute that may be used to specify the authLevel in the policy results.
- **policyLoadTime** - Displays the load time of the last policy.
- **sortedObligations** - Specifies the user workflow for authentication in the sorted order.
- **matchedPoliciesAndRules** - Displays all the list of matched policies and rules.
- **channelMatch** - Indicates whether the channel has matched.
- **policyName** - Displays the title of the policy that has been evaluated.
- **obligations** - Displays the obligations of each policy.
- **msg** - Specifies the description of the policy rule.
- **allConditionsMatched** - Indicates whether all the conditions have matched. For
  example, acrMatch, triggeredRiskRuleMatch, ipAddressMatch.

To know more about the description of the policy rules attributes, see Policy Attributes

Samples of different conditions specified in the Authentication Policies Evaluator API request

Example of "**Access**" condition in the Authentication Policies Evaluator API

```
{
    "action": "access",
    "user":"{{ldapConfigId}}/{{userId}}",
    "app": "DefaultTenantClient",
    "ipAddress":"192.0.2.1",
    "channel": "web"
}
```

Example of "**Groups**" condition in the Authentication Policies Evaluator API

```
{
    "action": "authenticate",
    "app": "DefaultTenantClient",
     "user": "{{ldapConfigId}}/{{userId}}",
    "ipAddress": "192.0.2.1",
    "channel": "web",
    "group": [
        "{{ldapConfigId}}/Admin",
```

```
        "{{ldapConfigId}}/Human Resources"
    ]
}
```

Example of "**triggered risk rule**" condition in the Authentication Policies Evaluator API

```
{
    "action": "authenticate",
    "user": "{{ldapConfigId}}/{{userId}}",
    "app": "DefaultTenantClient",
    "ipAddress": "192.0.2.1",
    "channel": "web",
    "triggeredRiskRule": ["device recognition"]
}
```

Example of "**riskThreshold**" condition in the Authentication Policies Evaluator API

```
{
    "action": "authenticate",
    "user": "{{ldapConfigId}}/{{userId}}",
    "app": "DefaultTenantClient",
    "ipAddress": "192.0.2.1",
    "channel": "web",
    "riskThreshold": 90
}
```

The following example shows how multiple conditions can be added in the Authentication Policies Evaluator API.

```
 {
     "ipAddress": "192.0.2.1",
     "app": "{{appName}}",
     "channel": "web",
     "group": [
         "{{ldapConfigId}}/Admin"
     ],
     "idpName": "samlbroacom.com",
     "action": "authenticate",
     "user": "{{ldapConfigId}}/{{userId}}",
     "acr": "urn:iam:acr:aal:pwd",
     "country": "Australia",
     "triggeredRiskRule": [
         "device recognition"
     ],
     "type": "authetnicate",
     "identitySourceId": "{{ldapConfigId}}",
     "clientContext": [
         {
             "key": "instanceId",
             "value": "brazil-instance-1"
         }
     ],
     "idTokenHint": "{{idTokenHint}}",
     "allConditionsMustMatch": true,
     "skipUserDisambiguation": true,
     "riskThreshold": 80
```

```
}
```

Reference Links

- The Authentication policy can be tested using the Admin Console. For information on this, see Test Policy in Admin Console.

Downloading and Uploading Fraud Data

Learn how to download and upload fraud data from the Support site to customer production environments using the ".../FraudDataConfiguration" and ".../FraudDataUploader" APIs.

Overview

To ensure accurate risk evaluation, fraud data must be regularly downloaded and uploaded to your (customer's) environment. This process can be done manually or automatically using APIs. Only system administrators can perform this task.

APIs Involved:

- ".../FraudDataConfiguration" API - Establishes a connection to the FTP Broadcom Support site. (***Not required for manual downloads***.)
- ".../FraudDataUploader" API - Handles both downloading and uploading of fraud data.
- ".../FraudDataUploader/status" API - Checks the status of the upload process.

Manual Download and Upload of Fraud Data

You need to download fraud data weekly and upload it to your database.

For a manual approach, at a nutshell these are the steps to follow, details have been explained in the FraudDataUploader API section:

1. Access the **Broadcom Support site** and navigate to the Solutions page.
2. Download the latest fraud data manually from this location - ***in the Broadcom Support Site (https://support.broadcom.com/)*, navigate to *My Tools > Solutions > Search by apar no (Apar Id: 99112374)***.
3. Use the **".../FraudDataUploader" API** to upload the downloaded data to the **Symantec Identity Security Platform database**

Automatic Download and Upload of Fraud Data

For an automated approach, at a nutshell these are the steps to follow, details have been explained in the next sections:

1. **Configure the connection**: Use the ".../FraudDataConfiguration" API to set up the FTP Broadcom Support site connection.
2. **Download and upload data**: Use the ".../FraudDataUploader" API, specifying the required key-value pairs in the request body.
3. **Check Upload Status**: Use the ".../FraudDataUploader/status" API to monitor the upload progress.

The manual and automatic download and upload of Fraud data steps have been explained in the following sections:

- FraudDataConfiguration API
- FraudDataUploader API

FraudDataConfiguration API

The ".../**FraudDataConfiguration"** API allows to configure the details required to connect to the FTP Broadcom support site for automatic download and upload of fraud data.

If you are manually downloading the fraud data from the Broadcom Support site, then you can use the ".../FraudDataUploader" API directly to upload the fraud data without updating the fraud data configuration.

This file will be updated every week with the latest Fraud Data.

API details:

Admin Scope	Operation Type	Endpoint URLs
urn:iam:g.settings	PUT	https://${SSP_MGMT_FQDN}/system/admin/v1/FraudDataConfiguration
urn:iam:g.settings	GET	https://${SSP_MGMT_FQDN}/system/admin/v1/FraudDataConfiguration

NOTE

- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://${SSP_MGMT_FQDN}/<systemtenantName>/admin/v1/FraudDataConfiguration`
  should look similar to:
  `https://mgmt-example.com/system/admin/v1/FraudDataConfiguration`
- To know the value of ${SSP_MGMT_FQDN} environment variable, see Configuring the Environment Variables.
- The default value for `<systemTenantName>` is "system".

Request and Response Samples of ".../FraudDataConfiguration" API

NOTE
Your Kubernetes cluster should have ***ports 21/22*** open to establish the FTP connection.

- Perform a PUT request using the following admin API to establish the connection to the FTP Broadcom support site

```
PUT https://${SSP_MGMT_FQDN}/system/admin/v1/FraudDataConfiguration
```

API Request

```
{
"accountName": "B-HFilpDak4y",
"accountPassword": "GXjSv3jZiP&|",
"ftpDownloadFile": "/VIPAuthHub/vip_authhub_fraud_data_latest.zip"
"testConnection": false
}
```

  - Enter the "`accountName` ", "`accountPassword` " that are used to access the Broadcom Support Site. The password will be stored in the IDSP database in encypted format.
  - Enter the "`ftpDownloadFile` " value.
    There are following two ways to enter this value, select one of the ways:
    - **Automatic Download**: To automate the download process and avoid weekly configuration changes, set the ftpDownloadFile value to: "/VIPAuthHub/vip_authhub_fraud_data_latest.zip". Ensure to enter the Broadcom-provided Username and Password to access the support site.
    - **Manual Download**:

- Login to the **Broadcom Support Site** (https://support.broadcom.com/), **navigate to My Tools > Solutions > Search by apar no** (Apar Id: 99112374).
- Click on the "**Secure FTP Download**" icon for the file "**vip_authhub_fraud_data_latest.zip**". This file will be updated every week with the latest Fraud Data.
- On the "***Secure FTP Download Instructions***" window, copy the "***ftpDownloadFile***" value from the ***Order Location***. For example, if the order location is this - /***36945088***/vip_authhub_fraud_data_latest.zip, the "ftpDownloadFile" value will be ***/36945088/vip_authhub_fraud_data_latest.zip***.

– If the credentials are proper, the configuration is updated, but, in case of incorrect credentials, the following error is displayed:
```
{ "errorCode": "9800020", "errorMessage": "Fraud data Configuration credentials not proper. Please verify the credentials and ftp download file again." }
```
– Configure the "testConnection" to "true" to perform a test connection based on the credentials entered to connect to the ftp support site. **By default, this parameter is set to "false".**

API Response


```
{

"accountName": "B-HFilpDak4",
"accountPassword": "TpiPYjZiaJL2pE==",
"ftpDownloadUrl": "https://ftp.broadcom.com/VIPAuthHub/vip_authhub_fraud_data_latest.zip",
"testConnection": false
"createdBy": {
"principalType": "CLIENT",
"principalId": "7f103150-4838-4018-8142-3dc2094148cc",
"principalName": "SystemClient"
},
"updatedBy": {
"principalType": "CLIENT",
"principalId": "7f103150-4838-4018-8142-3dc2094148cc",
"principalName": "SystemClient"
},
"updatedDateTime": "2025-03-20 08:43 am UTC"

}
```


- Perform a GET request using the following admin API to get the fraud data configuration
```
GET https://${SSP_MGMT_FQDN}/system/admin/v1/FraudDataConfiguration
```

API Response


```
{

"accountName": "B-HFilpDak4",
"accountPassword": "TpiPYjZiaJL2pE==",
"ftpDownloadUrl": "https://ftp.broadcom.com/VIPAuthHub/vip_authhub_fraud_data_latest.zip",
"testConnection": false",
"createdBy": {
"principalType": "CLIENT",
"principalId": "7f103150-4838-4018-8142-3dc2094148cc",
"principalName": "SystemClient"
},
"updatedBy": {
"principalType": "CLIENT",
"principalId": "7f103150-4838-4018-8142-3dc2094148cc",
"principalName": "SystemClient"
```

```
},
"updatedDateTime": "2025-03-20 08:43 am UTC"

}
```

FraudDataUploader API

The ".../**FraudDataUploader"** API allows weekly download and upload of fraud data in ZIP format from Neustar to customers production environments. This API ensures the latest data is downloaded, uploaded and stored in the IDSP database. Only System administrators can perform this task.

You can use this API to both download and upload fraud data, or solely for uploading if you have manually downloaded the fraud data from the IDSP Support site.

Admin Scope	Operation Type	Endpoint URLs
urn:iam:g.settings	POST	https://${SSP_MGMT_FQDN}/system/admin/v1/FraudDataUploader
urn:iam:g.settings	GET	https://${SSP_MGMT_FQDN}/system/admin/v1/FraudDataUploader/status

NOTE

- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://${SSP_MGMT_FQDN}/<systemtenantName>/admin/v1/FraudDataUploader`
  should look similar to:
  `https://mgmt-example.com/system/admin/v1/FraudDataUploader`
- To know the value of ${SSP_MGMT_FQDN} environment variable, see Configuring the Environment Variables.
- The default value for `<systemTenantName>` is "system".

Request and Response Samples of ".../FraudDataUploader" API

- Perform a POST request using the following admin API

```
POST https://${SSP_MGMT_FQDN}/system/admin/v1/FraudDataUploader
```

API Request

This API request can be used in two ways:

Manual Upload (if the file was downloaded manually):

- **Key-Value Pair**:
  - **file**: Specify the fraud data file in ZIP format in the "form-data" section of the API request body, as shown here:

    ```
    file: fraud data in zip format (example;2025.12.zip)
    ```
  - Example, 2025.12.zip or vip_authhub_fraud_data_latest.zip.
    - Where,
      - "**2025.12.zip" (example)** - contains fraud data for the 12th week of 2025. The week number updates every weekend.
      - "**vip_authhub_fraud_data_latest.zip**" - Always contains the most recent fraud data.

Automatic Download & Upload:

If you want the fraud data to be downloaded and uploaded automatically, specify the mode in the "form-data" section of the API request body, as shown here:

- **Key-Value Pair**:

```
mode = auto
```

NOTE
If both **file** and **mode** key-values are provided, the mode value takes precedence.

API Response

```
{
"fileName": "data.zip",
"message": "Fraud data upload started successfully. Please check status of the process using the status
 API."
}
```

- Perform a GET request using the following admin API to check the status of the fraud data upload

```
GET https://${SSP_MGMT_FQDN}/system/admin/v1/FraudDataUploader/status
```

API Response

```
{
"status": "Upload Completed",
"fraudDataUploadStartDateTime": "2025-02-12 08:43 AM UTC",
"fraudDataUploadEndDateTime": "2025-02-12 08:47 AM UTC"
}
```

Where, the possible status options are: **Upload Initiated**, **Upload Completed**, and **Upload Failed**.

For sample request and response payloads of all these URLs, refer to Fraud Data Uploader API Swagger documentation.

Uploading Fido Attestation Metadata

This document details the API endpoint for securely uploading FIDO metadata attestation file to the Symantec Identity Security Platform database.

FIDO Attestation Metadata Overview

This API facilitates the storage of FIDO metadata file, namely blob.jwt (containing attestation metadata for FIDO certified authenticators), and combined_aaguid.json, into the Symantec Identity Security Platform database. This central storage is necessary for the FIDO server to perform verification of attestation data sent by authenticators.

NOTE

These files are typically managed and refreshed by the Symantec Identity Security Platform dataseed job. However, should you choose to upload these files using this API, you assume responsibility for their ongoing maintenance. Once uploaded using the API, the dataseed job no longer manages these specific files.

Authorization

The request must include an Authorization header with a valid client access token.

API Endpoint

The API expects a multipart/form-data payload containing the metadata files packaged within a zip file.

API details:

Admin Scope	Operation Type	Endpoint URLs
urn:iam:g.settings	POST	https://${SSP_MGMT_FQDN}/system/admin/v1/FIDOMetadataUploader

NOTE

- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.

For example:
`https://${SSP_MGMT_FQDN}/<systemtenantName>/admin/v1/FIDOMetadataUploader`
should look similar to:
`https://mgmt-example.com/system/admin/v1/FIDOMetadataUploader`

- To know the value of ${SSP_MGMT_FQDN} environment variable, see Configuring the Environment Variables.
- The default value for `<systemTenantName>` is "system".

Request Header

Header	Value	Description
Content -Type	multipart/form-data	Specifies the content type of the request.
Authorization	Bearer <client_acess_token>	Bearer token for authentication

Request Payload

The API expects a multipart/form-data payload containing the metadata files packaged within a zip file.

Field	Type	Description
mode	string	Specifies the upload mode.
file	file	The zip file (.zip) containing blob.jwt and combined_aaguid.json.

Once the file is uploaded, "Fido metadata uploaded successfully" message is displayed.

Dynamic Client Registration

Learn about Dynamic Client Registration API

The Dynamic Client Registration (DCR) API is used to allow applications clients to register themselves dynamically with Symantec Identity Security Platform (acting as the authorization server). This is commonly used in OAuth 2.0 and OpenID Connect environments to enable automated client onboarding. This has been explained in the Open ID spec and OAuth RFC.

Dynamic Client Registration API

Admin Scope	Operation Type	Endpoint URLs
urn:iam:t.dcr To know more about the admin role that will have access to this scope, see Registration Client Administrator.	POST	https://${SSP_FQDN

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace {text} with the appropriate values in all the endpoints shown in the following table.
  For example:
  `https://${SSP_FQDN}/<tenantName>/oauth2/v1/register`
  should look similar to:

```
https://example.com/default/oauth2/v1/register
```

- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Request and Response Samples

POST operation of Dynamic Client Registration

Using the following POST operation you can register a dynamic client application in IDSP:

```
POST https://${SSP_FQDN}/<tenantName>/oauth2/v1/register
```

API Request:

```
#example 1 - DCR request payload

{
  "redirect_uris": [
    "https://www.example.com"
  ],
  "client_name": "testDCRApp",
  "client_type": "CONFIDENTIAL",
  "grant_types": [
    "authorization_code"
  ],
  "token_endpoint_auth_method": "tls_client_auth",
  "tls_client_auth_subject_dn": "CN=example-client,OU=clients,O=Example Inc,L=City,C=US",
  "tls_client_auth_san_dns": "CN=example-client,OU=clients,O=Example Inc,L=City,C=US",
  "tls_client_auth_san_uri": "CN=example-client,OU=clients,O=Example Inc,L=City,C=US",
  "tls_client_auth_san_ip": "123.456.789.0",
  "tls_client_auth_san_email": "User@example.com",
  "scopes": ["openid", "email"]
}

#example 2 - DCR request payload with template client

{
  "template_client_name": "template-client-app",
  // template-client-app is an active OAuth/OIDC App with the app m/d urn:app:useAppAsTemplateClientinDcr:
 true
  "redirect_uris": [
      "https://www.example.com"
  ]
}
```

The following table describes the parameter and its description:

Parameter	Description
redirect_uris	(Mandatory) Specify the redirect URIs for this application.
client_name	(Optional) Specify the client name for this application. If not provided in the API request, defau
grant_type	(Optional) Specify the grant type for this application. If not provided in the API request, defaults
client_type	(Optional) Accepts only "CONFIDENTIAL" and "PUBLIC" clients. If not provided in the API req

template_client_name	(Optional) Specify the template name. For example: "oidc-app1" (sample application name) or application GUID). To accept a template client name, you should have the following configured: • An existing OIDC app in active state of type "**Public/Confidential**". • Metadata configured as - "***urn:iam:app:useAppAsTemplateClientinDcr***": "true" (To know table). • When copying the application properties, only the "general" and "oauth/oidc" app properties "redirect_uri's" are not copied. • Request payload attribute values take precedence over the template values, if provided.
scope	(Optional) Represents Allowed Open ID Scope for this client. If not provided in the API request
scopes	Represents the list of allowed scopes for this client (as per rfc). Example: ["openid", "email"]
token_endpoint_auth_method	Represents the authentication mechanism for the "/token" endpoint. Example: tls_client_auth
tls_client_auth_subject_dn	Represents the subject DN value for client authentication using MTLS. Example: "CN=example-client,OU=clients,O=Example Inc,L=City,C=US"
tls_client_auth_san_dns	Represents the SAN DNS value for client authentication using MTLS. Example: "CN=example-client,OU=clients,O=Example Inc,L=City,C=US"
tls_client_auth_san_uri	Represents the SAN URI value for client authentication using MTLS. Example: "CN=example-client,OU=clients,O=Example Inc,L=City,C=US"
tls_client_auth_san_ip	Represents the SAN IP value for client authentication using MTLS. Example: "123.456.789.0"
tls_client_auth_san_email	Represents the SAN email value for client authentication using MTLS. Example: "User@example.com"

API Response:


```
{
"client_name": "testDCRApp",
"grant_types": [
"authorization_code"
],
"redirect_uris": [
"https://www.example.com"
],
"client_id": "f19d0e7d-704f-4bdc-909b-e30f60241bbc",
"client_secret": "052bc8c1-0a99-422e-a9af-99152bf480da",
"client_id_issued_at": 1739943291842,
"client_secret_expires_at": 0,
"token_endpoint_auth_method": "tls_client_auth"
```

```
}
```

For sample request and response payloads of this URL, refer to DCR API Swagger documentation.

Identity REST APIs

Learn about the Identity REST APIs.

The Identity Service provides REST APIs for configuring and managing the identities in the Internal Identity Store. The Identity Service REST APIs provide capabilities for tasks such as Internal Identity Store Configuration, User Management, Group Management, Password Management, and Self Service.

For more information about these APIs, refer to the following topics:

- Invoking the Identity REST APIs
- The Internal Identity Store Configuration API
- SCIM Schema and Resource Types API
- User API
- Group API
- Group Operator API
- Bulk API
- Bulk Import Credentials
- User Self Service API
- Password Management API
- Scoping Constraints API
- Delegations APIs

Invoking the Identity REST APIs

The Identity REST APIs use the same security model used by all other Symantec Identity Security Platform REST APIs – a model based on OAuth2 and OpenID industry standards, relying on JWT-based Access Token to control access to REST APIs. See the API Security using OAuth2 and the Making the First REST API Call sections in the Developer Resources section for details.

Privileges Required to Invoke the Identity REST APIs

Identity REST APIs require “urn:iam:t.*” scopes representing tenant-level privileges granted by the *Tenant Admin Authorization Policy* as well as “urn:iam:m.me" scopes representing self service operations on an end user using the identity from the access token when the token’s sub_type is user. The following tenant level scopes are required for the Identity REST APIs:

- `urn:iam:t.users`
- `urn:iam:t.groups`
- `urn:iam:t.identityclient`

Refer to the Scopes and APIs Table and the Identity API Privileges for complete details.

The endpoint URLs defined in this section contain placeholders for deployment specific values. Replace the text `{{text}}` including `{{}}` characters with appropriate values, in all the endpoints shown in the table. For example:

```
https://${SSP_FQDN}/{{tenantName}}/scim2/v1/Users
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring the Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/scim2/v1/Users`
  should look similar to:
  `https://example.com/default/scim2/v1/Users`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Should look similar to:

```
https://example.com/default/scim2/v1/Users/
```

The Internal Identity Store Configuration API

The Internal Identity Store can be configured with various properties. The Configuration API supports the following endpoints.

Operation Description	Operation Types	Endpoint URLs
Internal Identity Store Configuration	GET PUT PATCH	`https://${SSP_FQDN}/{{tenantName}}/admin/v1/InternalIdStoreConfig`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/InternalIdStoreConfig`
  should look similar to:
  `https://example.com/default/admin/v1/InternalIdStoreConfig`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

GET API

```
GET https://example.com/default/admin/v1/InternalIdStoreConfig
```

GET API Response

```
{
"idStoreMappingStrategy": "user_loginid=${idprofile.user_loginid}",
"disambiguationQuery": "user_loginid eq ${loginid}",
"unlockAccountAfterPasswordReset": false,
"useMaskedCredentialValues": false,
"passwordResetInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiatePasswordReset",
"passwordResetChangerUrl": "${system.issuerUrl}/ui/v1/signin/passwordReset",
"accountUnlockInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiateAccountUnlock",
```

```
"usePriorityToDeterminePrimary": false,
"customUserAttributes": [
  {
      "name": "company",
      "type": "STRING",
      "searchable": true,
      "enabled": false
      "multivalued": true
  {
      "name": "hireDate",
      "type": "DATE",
      "searchable": true,
      "enabled": true
  },
  {
      "name": "jobCode",
      "type": "INTEGER",
      "searchable": true,
      "enabled": false
  },
  {
      "name": "isLocalCitizen",
      "type": "BOOLEAN",
      "searchable": true,
      "enabled": true
  }
],
"createdDateTime": "2023-11-21 07:35 PM UTC",
"updatedDateTime": "2023-11-21 07:35 PM UTC",
"updatedBy": {
  "principalType": "CLIENT",
  "principalId": "e9c2fb28-f90c-4f9c-9b63-ce6da98d28e7",
  "principalName": "DefaultTenantClient"
}
}
```

PUT API

The Internal Id Store supports a PUT operation to update the configuration. As with any PUT operation, the request overwrites the entire configuration. The following PUT semantics are enforced to prevent certain property overwrites.

- Custom attribute definitions are additive only. Custom attribute definitions cannot be removed, they can only be disabled/enabled. If a custom attribute is disabled, it cannot be referenced in a request or returned in a response – it is effectively does not exist to the Identity Service.
- Restoring the Password Recovery and Account Unlock URLs. The "passwordResetInitiatorUrl", “passwordResetChangerUrl”, accountUnlockInitiatorUrl" properties define the endpoints that facilitate Password Recovery and Account Unlock. The Identity Service provides default endpoints for these operations. If a PUT operation removed or overwrote these default URLs a templates=true parameter can restore these defaults as follows:

  ```
  PUT https://${SSP_FQDN}/{{tenantName}}/admin/v1/InternalIdStoreConfig?templates=true
  ```

```
PUT https://example.com/default/admin/v1/InternalIdStoreConfig
```

Sample PUT API Request

```
{
"idStoreMappingStrategy": "user_loginid=${idprofile.user_loginid}",
"disambiguationQuery": "user_loginid eq ${loginid}",
"unlockAccountAfterPasswordReset": true,
"useMaskedCredentialValues": false,
"passwordResetInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiatePasswordReset",
"passwordResetChangerUrl": "${system.issuerUrl}/ui/v1/signin/passwordReset",
"accountUnlockInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiateAccountUnlock",
"usePriorityToDeterminePrimary": false,
"customUserAttributes": [
[
  {
      "name": "officeLocation",
      "type": "STRING",
      "searchable": true,
      "enabled": true
  }
]
}
```

PATCH API

The Internal Id Store supports a PATCH operation to update the configuration. The PATCH operation performs a **REPLACE** operation on all attributes except **customUserAttributes**. Custom user attributes can be added, updated, and removed using the corresponding operation; **ADD**, **REPLACE** or **REMOVE**, respectively. The **REMOVE** operation on custom attributes do not remove the attribute definition. Instead, it sets **enabled** to **false**.

> **NOTE**
> The REPLACE operation does not update the **type** of an existing attribute. The REPLACE operation only updates the **enabled** and **searchable** attributes for existing attributes whose name matches the value that is provided for **name**.

```
PLACE https://example.com/default/admin/v1/InternalIdStoreConfig
```

Sample PATCH API Request

```
{
"disambiguationQuery": "user_loginid eq ${loginid}",
"unlockAccountAfterPasswordReset": false,
"adminCanGeneratePassword": false,
"userNameGenerationRule": "${idprofile.given_name} + '.' + ${idprofile.family_name}",
"emailsMustBeUnique": true,
"customUserAttributes": [
 {
     "operation": "ADD",
     "attribute": {
         "name": "<BOOLEAN-ATTR-NAME>",
         "type": "boolean",
         "searchable": false,
         "enabled": true
     }
 },
 {
     "operation": "ADD",
     "attribute": {
```

```
            "name": "<DATE-ATTR-NAME>",
            "type": "date",
            "searchable": false,
            "enabled": true
        }
    },
    {
        "operation": "ADD",
        "attribute": {
            "name": "<INTEGER-ATTR-NAME>",
            "type": "integer",
            "searchable": false,
            "enabled": true
        }
    },
    {
        "operation": "REPLACE",
        "attribute": {
            "name": "<STRING-ATTR-NAME-2>",
            "searchable": true,
            "enabled": true
        }
    },
    {
        "operation": "REMOVE",
        "attribute": {
            "name": "<ATTR-NAME>"
        }
    }
  ]
  }
```

SCIM Schema and Resource Types API

Learn about SCIM Schema and Resource Types API

The SCIM API defines the following endpoints for retrieving the schemas, resource types, and service provider configuration details. These are informational APIs that only support GET operations. The Resource Types API lists the supported namespaces on each Resource Type. The Schemas API reflects the namespaces and the set of attributes in those namespaces. See the Identity User and Group Schemas topic for details on the supported namespaces and the namespace attributes.

Operation Description	Operation Types	Endpoint URLs
SCIM Configuration		
get schemas	GET	`https://${SSP_FQDN}/{{tenantName}}/scim2/v1/Schemas`
get Service Provider Configuration	GET	`https://${SSP_FQDN}/{{tenantName}}/scim2/v1/ServiceProviderConfig`
get Resource Types	GET	`https://${SSP_FQDN}/{{tenantName}}/scim2/v1/ResourceTypes`

NOTE

- To know the value of ${IDSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/scim2/v1/Schemas`
  should look similar to:
  `https://example.com/default/scim2/v1/Schemas`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

SCIM Resource Types Request

```
GET https://example.com/default/scim2/v1/ResourceTypes
```

SCIM Resource Types Response

```
{
    "totalResults": 2,
    "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:ListResponse"
    ],
    "Resources": [
        {
            "schema": "urn:ietf:params:scim:schemas:core:2.0:User",
            "endpoint": "/Users",
            "meta": {
                "location": " https://${IDSP_FQDN}/{{tenantName}}/scim2/v1/ResourceTypes/User",
                "resourceType": "ResourceType"
            },
            "schemas": [
                "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
            ],
            "name": "User",
            "description": "User Account",
            "schemaExtensions": [
                {
                    "schema": "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User",
                    "required": false
                },
                {
                    "schema": "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User:credentials",
                    "required": false
                },
                {
                    "schema": "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
                    "required": false
                },
                {
                    "schema": "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User:custom",
                    "required": false
                }
```

```
            ],
            "id": "User"
        },
        {
            "schema": "urn:ietf:params:scim:schemas:core:2.0:Group",
            "endpoint": "/Groups",
            "meta": {
                "location": "https://${SSP_FQDN}/{{tenantName}}/scim2/v1/ResourceTypes/Group",
                "resourceType": "ResourceType"
            },
            "schemas": [
                "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
            ],
            "name": "Group",
            "description": "Group",
            "id": "Group"
        }
    ]
}
```

User API

You administratively manage Users through the SCIM Users endpoint. The following table shows the endpoint and supported operations.

User Management	GET PATCH POST PUT	`https://${IDSP_FQDN}/{{tenantName}}/scim2/v1/Users`

	GET	`GET /scim2/v1/Users`
		Adds meta support for `createdBy` and `updatedBy`.
		For example:
		`"meta": {`
		`"created": "2024-03-01T18:03:49Z",`
		`"createdBy": "DefaultTenantClient",`
		`"lastModified": "2024-03-05T16:28:12Z",`
		`"location": "https://ssp-bob-gke.sspdev.dev.broadcom.com/default/scim2/v1/Users/1500d7e7-e132-4d48-9bbc-f409fa66835c",`
		`"resourceType": "USER",`
		`"updatedBy": "defaulttenantadminuser"`
		`}`

	POST	Optional query parameter `accountActivated` : Mandates that the token generated is for `accountActivation` and not for `passwordReset` . For example, the email or SMS templates that are chosen for the delivery of the token are for `accountActivation` and not `passwordReset` . Default is **true**. When used with the optional query parameter `dontSendActivationNotice` and both are set to **true**, the `accountActivated` flag dictates the state of the attribute (`accountActivated`) on the user created. `https://${IDSP_FQDN}/{{tenantName}}/scim2/v1/Users?accountActivated=true`
		Optional query parameter `countonly` : If set to `true` , the SCIM search returns the number of users or groups present in the tenant. `https://${IDSP_FQDN}/{{tenantName}}/scim2/v1/Users?countonly=true`
		Optional query parameter `dontSendActivationNotice` : Applications willing to own the Create User Post-processing notifications can turn off the default behavior of sending welcome emails. To do so, set the `dontSendActivationNotice` parameter to **true** to the create User API. `https://${IDSP_FQDN}/{{tenantName}}/scim2/v1/Users?dontSendActivationNotice=true`
		Optional query parameter `idpOriginId` : Privileged Clients can create a federated user (for example, if `federated` is set to **true**) by setting this attribute to a valid external IdP or external identity source. `https://${IDSP_FQDN}/{{tenantName}}/scim2/v1/Users?idpOriginId=<valid external IdP or external identity source>`
		Optional query parameter `redirect_uri` : Set a value that Privileged Clients can store on the activation token during User creation and which is returned during the token redeem operation. `https://${IDSP_FQDN}/{{tenantName}}/scim2/v1/Users?redirect_uri=<redirect_url_on_token_redemption>`
		Optional query parameter `skipEmailValidation` : Allows Privileged Clients to override the `skipEmailValidation` setting on the Identity Store configuration. `https://${IDSP_FQDN}/{{tenantName}}/scim2/v1/Users?skipEmailValidation=<true/false>`

		Optional query parameter `skiplimit` : If set to `true` , and the principal has one of the following privileges: • g.settings • t.all • t.exceedmaxscimsearchlimit, The principal can retrieve all users and groups that match the filter without bounding to the value of the scimSearchLimit setting as defined in the Identity Store configuration. `https://${IDSP_FQDN}/{{tenantName}}/scim2/v1/Users?skiplimit=true`
		Optional query parameter `skipPhoneValidation` : Allows Privileged Clients to override the `skipPhoneValidation` setting on the Identity Store configuration. `https://${IDSP_FQDN}/{{tenantName}}/scim2/v1/Users?skipPhoneValidation=<true/false>`
	DELETE	Optional parameter `replacementManager` : If a user that is assigned as a manager to other accounts (users, service accounts, and so on) is deleted, the accounts are not automatically assigned a new manager. The `replacementManager` parameter assigns the manager that is included in the parameter as the new manager for all accounts previously assigned to the deleted manager. `https://${IDSP_FQDN}/{{tenantName}}/scim2/v1/Users?replacementManager=<managerID>` Assigning the provided manager fails if: • The provided manager does not exist • The API call cannot access the provided manager. For example: — The IdP of the new manager does not match the IdP of the calling client (that is, the calling client exists in an IdP, but the provided manager does not exist there). — The calling client does not have the appropriate admin scope on the provided manager to assign the manager to the user.

NOTE

- To know the value of ${IDSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values in all the endpoints shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/scim2/v1/Users`
  should look similar to:
  `https://example.com/default/scim2/v1/Users`
- To know more about the scopes that are required to secure these APIs, refer to the Scopes and APIs Table.

User attributes that are referenced in SCIM requests and responses are qualified by a defined namespace. This namespace appears in the following sample POST User request and response.

A sample User create (POST) request

```
{
  "schemas":[
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User"
  ],
  "userName":"amelia_haley",
  "password":"Sample@1234",
  "displayName":"Amelia Haley",
  "active":true,
  "name":{
    "formatted":"Amelia Haley",
    "familyName":"Haley",
    "givenName":"Amelia"
  },
  "title":"HR Specialist",
  "userType":"Employee",
  "emails":[
    {
      "value":"amelia.haley@acme.com",
      "type":"work",
      "primary":true,
      "verified":true
    }
  ],
  "phoneNumbers":[
    {
      "value":"555-555-5555",
      "type":"home",
      "primary":true,
      "verified":true
    }
  ],
  "addresses":[
    {
      "streetAddress":"333 Stader Drive",
      "locality":"Framingham",
      "region":"MA",
      "postalCode":"01702",
      "country":"USA",
      "formatted":"1000 Stader Drive, Framingham, MA-01702, USA",
      "type":"work",
      "primary":true
    }
  ],
  "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": {
      "costCenter": "US",
      "department": "HR"
    },
  "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User": {
      "gender": "female",
      "birthDate": "2000-01-24T00:00Z",
      "disabled": false
```

```
    },
  "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User:custom": {
      "officeLocation":"New York",
      "jobCode":"3000"
    }
  }
```

Sample create response:

```
{
  "schemas": [
     "urn:ietf:params:scim:schemas:core:2.0:User",
     "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
     "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User",
     "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User:credentials",
     "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User:custom"
  ],
  "active": true,
  "addresses": [
    {
       "country": "USA",
       "formatted": "1000 Stader Drive, Framingham, MA-01702, USA",
       "locality": "Framingham",
       "postalCode": "01702",
       "primary": true,
       "region": "MA",
       "streetAddress": "333 Stader Drive",
       "type": "work",
       "verified": false
     }
  ],
  "displayName": "Amelia Haley",
  "emails": [
     {
       "primary": true,
       "type": "work",
       "value": "amelia.haley@acme.com",
       "verified": true
     }
  ],
  "id": "fcf3a1cd-f3f7-463f-979e-e4edfff91996",
  "meta": {
     "created": "2023-11-26T20:38:39.704Z",
     "lastModified": "2023-11-26T20:38:39.704Z",
     "location": "https://{some_sp host}default/scim2/v1/Users/fcf3a1cd-f3f7-463f-979e-e4edfff91996",
     "resourceType": "User"
  },
  "name": {
     "familyName": "Haley",
     "formatted": "Amelia Haley",
     "givenName": "Amelia"
    },
  "phoneNumbers": [
     {
```

```
      "primary": true,
      "type": "home",
      "value": "617-555-5555",
      "verified": true
    }
  ],
  "title": "HR Specialist",
  "userName": "amelia_haley",
  "userType": "Employee",
  "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": {
      "costCenter": "US",
      "department": "HR"
    },
  "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User": {
      "birthDate": "2000-01-24T00:00:00Z",
      "disabled": false,
      "gender": "female",
      "identitySourceId": "internal",
      "identitySourceName": "internal",
      "identitySourceType": "internal",
      "idpOriginId": "internal",
      "passwordMustChange": false,
      "status": "Active",
      "strikeCount": 0,
      "updatable": true,
      "userPasswordChangeDateTime": "2023-11-26T20:38:39.709Z"
    },
  "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User:custom": {
      "jobCode": 3000,
      "officeLocation": "New York"
    }
  }
```


The full list of User schema namespaces and the namespace attributes are listed in the Identity Service User and Group Management section and in the Admin Console Identity Store section under the Internal Identity Store topic.

Group API

You can administratively manage Groups and Group members through the SCIM Groups endpoint.

The following table shows the endpoint and supported operations.

Operation Description	Operation Types	Endpoint URLs
Group Management	GET DELETE PATCH POST PUT	`https://${SSP_FQDN}/{{tenantName}}/scim2/v1/Groups`
	POST	Optional query parameter `countonly` : If set to `true` , the SCIM search returns the number of users or groups present in tenant. `https://${SSP_FQDN}/{{tenantName}}/scim2/v1/Users?countonly=true`
		Optional query parameter `skiplimit` : If set to `true` , and the principal has one of the following privileges: • g.settings • t.all • t.exceedmaxscimsearchlimit The principal can retrieve all users and groups that match the filter without bounding to the value of the `scimSearchLimit` setting as defined in the Identity Store configuration. `https://${SSP_FQDN}/{{tenantName}}/scim2/v1/Users?skiplimit=true`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see #unique_155.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values in all the endpoints shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/scim2/v1/Groups`
  should look similar to:
  `https://example.com/default/scim2/v1/Groups`
- To know more about the scopes that are required to secure these APIs, refer to the Scopes and APIs Table.

A sample Group create (POST) request:

```
{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:Group"
  ],
  "displayName": "North_America_Sales"
}
```

Sample Create response:

```
{
  "displayName": " North_America_Sales ",
  "meta": {
  "created": "2023-11-27T01:36:44.359Z",
```

```
  "location": "https://{some_ssp_host}/default/scim2/v1/Groups/be382041-228e-417c-8b74-a836898ae5e3"
},
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:Group"
    ],
  "id": "be382041-228e-417c-8b74-a836898ae5e3"
}
```

Sample Add/Remove Group Member Request:

```
PUT https://${SSP_FQDN}/{{apiPathTenant}}/scim2/v1/Groups/be382041-228e-417c-8b74-a836898ae5e3

{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:PatchOp"
    ],
  "Operations": [
    {
      "op": "add",
      "path": "members",
      "value": [
        {
          "value": "fcf3a1cd-f3f7-463f-979e-e4edfff91996"
        }
      ]
    },
    {
      "op": "remove",
      "path": "members",
      "value": [
        {
          "value": "0e471e9b-8865-4b0f-88b5-ddba5ccaeec2"
        }
      ]
    }
  ]
}
```

Allowing the API to Partially Succeed

When using the Group API to perform a PUT, POST, or PATCH operation to assign members to a group, set the `allowPartialSuccess` flag to continue processing if the API cannot complete the operation on all members. Without this flag, the Group API returns an error and makes no changes if it cannot correctly process all operations. Setting the `allowPartialSuccess` flag allows the API to capture invalid operations while still processing valid operations.

When this flag is set to `true` , the API returns a 200 code when only some operations succeed. Otherwise, standard success or failure codes are returned. The response also includes detail messages that describe any failures.

The following sample shows the response to a PATCH call using the Group API. In this example, the first member (Sally Smith) is successfully assigned to the group but the second member (John Johnson) is not assigned.

```
[
  {
    "membersResults": {
      "startIndex": 1,
```

```
      "totalResults": 1,
      "itemsPerPage": 1
      },
    "displayName": "FinanceGroup",
    "schemas": [
      "urn:ietf:params:scim:schemas:core:2.0:Group"
      ],
    "members": [
      {
        "display": "Sally Smith",
        "type": "User",
        "$ref": "https://auth.broadcom.com/default/scim2/v1/Users/b24je73-bd3f-44jf-869c-d98400bf645",
        "value": "b26db773-bd3f-45c7-869c-d98400bf4910"
      }
    ],
    "id": "bc8b865f-4775-4294-b2e4-cc5067e65208",
    "status": 200
  },
  {
    "schemas": [
      "urn:ietf:params:scim:api:messages:2.0:Error"
      ],
    "errorCode": "89400001",
    "scimType": "invalidValue",
    "detail": "Provided user(s) not found: "John Johnson",
    "status": "400"
  }
]
```


Group Description Field and Member Search

The following APIs represent a group description API and new member search parameters:

`startIndex` - The 1-based index of the query result for the first member;

`count` - Non-negative integer, specifies the desired maximum number of member query results per page. A value bigger than the `InternalIdStoreConfig.scimSearchLimit` is replaced by `InternalIdStoreConfig.scimSearchLimit;`

`sortBy` - The attribute whose value SHALL be used to order the returned responses

`sortOrder` - sort order descending/ascending

New result attribute:


```
"membersResults": {
"itemsPerPage": 10,
"startIndex": 1,
"totalResults": 44
}
```


Sub-attributes `startIndex` and `itemsPerPage` are present in the response only if the `members` attribute is requested in the response.

The API also supports search by `members.display` and `members.value` attributes, and filter `members pr`.

API	Request Change	Response Change

POST/PUT/PATCH / scim2/v1/Groups	POST/PUT
	{
	"schemas": [
	"urn:ietf:params:scim:schemas:core:2.0:Group"
	],
	"displayName":
	"SCIM_Sample_Group",
	"description": "demo group"
	}
	PATCH
	{
	"schemas":
	["urn:ietf:params:scim:api:messages:2.0:PatchOp"
	],
	"Operations": [
	{
	"op": "add",
	"path": "description",
	"value": "new description"
	}
	]
	}
		{
		"schemas": [
		"urn:ietf:params:scim:schemas:core:2.0:Group"
		],
		"description": "demo group",
		"displayName":
		"SCIM_Sample_GroupTTT",
		"id": "2d55d376-8147-408c-acb3-bd092da1c5e8",
		"membersResults": {
		"totalResults": 0
		},
		"meta": {
		"created":
		"2024-03-08T16:38:34.632Z",
		"createdBy":
		"DefaultTenantClient",
		"location": "https://c02g83ynmd6t.dhcp.broadcom.net/default/scim2/v1/Groups/2d55d376-8147-408c-acb3-bd092da1c5e8"
		}
		}

GET /scim2/v1/Groups	?attributes=members,membersResults&startIndex=1&count=10&sortBy=display&sortOrder=ascending	{
		"schemas":
		["urn:ietf:params:scim:schemas:core:2.0:Group"],
		"displayName": "group10000",
		"id": "f0c2faf3-d907-483d-bb03-730c122dbb77",
		"membersResults": {
		"itemsPerPage": 20,
		"startIndex": 1,
		"totalResults": 44
		},
		"members": [
		{
		…

GET /scim2/v1/Groups	&filter=members pr &filter=members.display sw ABC &filter=members.value eq XYZ	

Group Operator API

Use the Group Operator API to set a group to be read-only.

You can set whether a group is read-only using the Group Operator API. When the `readOnly` attribute for a group is set to **true**, the principal can only read details for the group but cannot perform any update or delete operations on the group,

including updating group memberships. If set to `false`, principals with the appropriate permissions can read the details for the group and can perform update or delete operations on the group.

You can also use the Group Operator to view whether the group is read-only.

Operation Description	Operation Types	Endpoint URLs
Group Operational Status Management	GET	`https://${SSP_FQDN}/{{tenantName}}/admin/v1/GroupOperator/{{groupID}}`
	POST	`https://${SSP_FQDN}/{{tenantName}}/admin/v1/GroupOperator/{{groupID}}`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Constructing the Service FQDN.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values in all the endpoints shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/GroupOperator`
  should look similar to:
  `https://example.com/default/admin/v1/GroupOperator`
- To know more about the scopes that are required to secure these APIs, refer to the Scopes and APIs Table.

A sample Group Operator POST request:

```
{
"readOnly":true
}
```

Bulk API

SCIM supports bulk operation on Users and Groups using a Bulk endpoint through the POST operation. Any write operation on a User or Group is supported through the bulk endpoint.

Operation Description	Operation Types	Endpoint URLs
Bulk Management	POST	https://${SSP_FQDN}/{{tenantName}}/scim2/v1/Bulk

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values in all the endpoints shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/scim2/v1/Bulk`
  should look similar to:
  `https://example.com/default/scim2/v1/Bulk`
- To know more about the scopes that are required to secure these APIs, refer to the Scopes and APIs Table.

The Bulk API supports lists of operations in the request. Each Operation contains the following fields:

- method – POST, PUT, PATCH, or DELETE
- path - /Users or /Groups
- bulkId – a unique bulk operation id
- data – the User or Group json request data for the method

The following example shows a sample bulk request that creates three users:

```
{
 "schemas": [
     "urn:ietf:params:scim:api:messages:2.0:BulkRequest"
 ],
 "failOnErrors": 20,
 "Operations": [
     {
         "method": "POST",
         "path": "/Users",
         "bulkId": "bulktest1",
         "data": {
             "schemas": [
                 "urn:ietf:params:scim:schemas:core:2.0:User",
                 "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User"
             ],
             "userName":"bjones",
             "password":"SAMPLe1234#",
             "displayName":"Billi Jones",
             "active":true,
             "name":{
                 "formatted":"Billi Jones",
                 "familyName":"Jones",
                 "givenName":"Billi"
             },
             "userType":"temporary",
             "emails":[
                 {
                     "value":"billi.jones@acme.com",
                     "type":"work",
                     "primary":true
                 }
             ],
             "phoneNumbers":[
                 {
                     "value":"1-555-555-5555",
                     "type":"work",
                     "primary":true
                 }
             ],
             "addresses":[
                 {
                     "streetAddress":"32 Riverside Lane",
                     "locality":"Marcy Island",
                     "region":"FL",
                     "postalCode":"92812",
```

```
                "country":"USA",
                "formatted":"32 Riverside Lane, Marcy Island, FL, 92812, USA",
                "type":"home",
                "primary":true
            }
        ],
        "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User": {
                    "gender": "female",
                    "birthDate": "2000-01-24T00:00Z",
                    "disabled": false
            }
    }
},
{
    "method": "POST",
    "path": "/Users",
    "bulkId": "bulktest2",
    "data": {
        "schemas": [
            "urn:ietf:params:scim:schemas:core:2.0:User",
            "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User"
        ],
        "userName":"mharris",
        "password":"SAMPLe1234#",
        "displayName":"Matt Harris",
        "active":true,
        "name":{
            "formatted":"Matt Harris",
            "familyName":"Harris",
            "givenName":"Matt"
        },
        "userType":"temporary",
        "emails":[
            {
                "value":"mharris@acme.com",
                "type":"work",
                "primary":true
            }
        ],
        "phoneNumbers":[
            {
                "value":"888-888-8888",
                "type":"work",
                "primary":true
            }
        ],
        "addresses":[
            {
                "streetAddress":"11 Glenn Ave",
                "locality":"Springfield",
                "region":"CT",
                "postalCode":"92811",
                "country":"USA",
```

```
                "formatted":"11 Glenn Ave, Springfield, CT,92811, USA",
                "type":"home",
                "primary":true
            }
        ],
        "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User": {
                    "gender": "male",
                    "birthDate": "2000-01-24T00:00Z",
                    "disabled": false
            }
    }
},
{
    "method": "POST",
    "path": "/Users",
    "bulkId": "bulktest3",
    "data": {
        "schemas": [
            "urn:ietf:params:scim:schemas:core:2.0:User",
            "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User"
        ],
        "userName":"dmarkus",
        "password":"SAMPLe1234#",
        "displayName":"Dharma Markus",
        "active":true,
        "name":{
            "formatted":"Dharma Markus",
            "familyName":"Markus",
            "givenName":"Dharma"
        },
        "userType":"temporary",
        "emails":[
            {
                "value":"dmarkus@acme.com",
                "type":"work",
                "primary":true
            }
        ],
        "phoneNumbers":[
            {
                "value":"999-999-9999",
                "type":"work",
                "primary":true
            }
        ],
        "addresses":[
            {
                "streetAddress":"555 Pine St",
                "locality":"Lowell",
                "region":"MA",
                "postalCode":"01822",
                "country":"USA",
                "formatted":"555 Pine St, Lowell, MA,01822, USA",
```

```
                    "type":"home",
                    "primary":true
                }
            ],
            "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User": {
                        "gender": "male",
                        "birthDate": "2000-01-24T00:00Z",
                        "disabled": false
                }
        }
    }
 ]
}
```

Bulk request response:

```
{
 "schemas": [
     "urn:ietf:params:scim:api:messages:2.0:BulkResponse"
 ],
 "Operations": [
     {
         "bulkId": "bulktest1",
         "method": "POST",
         "location": "https://{some_ssp_host}/default/scim2/v1/Users/c9658004-1bdc-4569-8d1a-fbc9c9e97935",
         "status": {
             "code": 201
         }
     },
     {
         "bulkId": "bulktest2",
         "method": "POST",
         "location": "https://{some_ssp_host}/default/scim2/v1/Users/1b5d08d2-692a-4132-bc53-04c51d658751",
         "status": {
             "code": 201
         }
     },
     {
         "bulkId": "bulktest3",
         "method": "POST",
         "location": "https://{some_ssp_host}/default/scim2/v1/Users/3e6a8c13-748a-4ae0-8507-eb01a5b18659",
         "status": {
             "code": 201
         }
     }
 ]
}
```

Allowing the API to Partially Succeed

When using the Bulk API to perform a PUT, POST, or PATCH operations and one or more of the operations adds members to a group, set the `allowPartialSuccess` flag to continue processing if the group membership add operation in the API cannot add one or more members.

Without this flag, the response returns an error and makes no changes if it cannot correctly process group membership add operations. Setting the `allowPartialSuccessflag` flag allows the API to capture invalid operations while still processing valid operations.

When this flag is set to `true` , the API returns a 207 code if one or more of the group membership modify operations in the API fail. Otherwise, standard success or failure codes are returned for each operation. The response also includes detail messages that describe any group membership add failures.

The following sample shows the response to a PATCH call using the Bulk API. In this example, the first member (Sally Smith) is successfully assigned to the group, but two other members (Bob Johnson and John Johnson) are not.


```
[
{
    "membersResults": {
        "startIndex": 1,
        "totalResults": 1,
        "itemsPerPage": 1
    },
    "displayName": "FinanceGroup",
    "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:BulkResponse"
    ],
    "members": [
        {
            "display": "Sally Smith",
            "type": "User",
            "$ref": "https://auth.broadcom.com/default/scim2/v1/Users/b26db773-bd3f-45c7-869c-d98400bf4910",
            "value": "b26db773-bd3f-45c7-869c-d98400bf4910"
        }
    ],
    "id": "bc8b865f-4775-4294-b2e4-cc5067e65208",
    "status": 200
},
{
            "path": "/Groups/FinanceGroup",
            "bulkId": "bulkPatch01",
            "method": "PATCH",
            "response": [
                "{\"schemas\":[\"urn:ietf:params:scim:api:messages:2.0:Error\"],\"errorCode\":\"89400001\",
\"scimType\":\"invalidValue\",\"detail\":\"Provided user(s) not found: BobJohnson\",\"status\":\"400\"}",
                "{\"schemas\":[\"urn:ietf:params:scim:api:messages:2.0:Error\"],\"errorCode\":\"89400001\",
\"scimType\":\"invalidValue\",\"detail\":\"Provided user(s) not found: JohnJohnson\",\"status\":\"400\"}"
            ],
            "location": "https://example.com/default/scim2/v1/Groups/9ebc5de9-52be-46a0-8c48-97547d2c8588",
            "status": {
                "code": 207
            }
        }
}

]
```

Bulk Import Credentials

Use SCIM Bulk Import to manage user authentication credentials (email addresses, phone numbers, and devices) for both internal ID store identities and external ID store accounts, with support for encrypted data transmission and various CRUD operations.

You can bulk import user credentials, such as email addresses, phone number, using SCIM Bulk import. This bulk import applies both to identities managed in the Internal ID store and accounts representing identities managed in external ID stores.

The SCIM Bulk Import procedure supports managing stated account attributes while accommodating synchronizing user credentials in the payload. The identity stack operates at both the Internal ID Store level and Account level. By using the SCIM protocol, the credentials sync converts given emails, phone numbers, and devices with a known `capabilityType` into authentication credentials. This process helps with accommodating any future credential type.

The outcome of the sync process changes the user authentication credentials. Changes may include adding a credential, deleting an existing credential, or replacing the origin or status of the existing credential.

By default, without providing an identity source id in the request URL, import sync operations occur as a regular SCIM service. This SCIM service operates on behalf of the internal id store users. In this mode, the user's authentication credentials align with the most-recent state of the user's emails, phoneNumbers, and devices.

For account sync operations representing either an external ID store users or internal ID store users, specify the ID store id in the URL. To sync new accounts, the URL must also specify the identity store name and type.

The SCIM Bulk Import procedure supports in-transit security of sensitive credential data using multiple methods:

- `{x5c}xxxxx` : In-transit, PKI-based data encryption using the public key of the tenant signing key (which can be retrieved from the JWKS endpoint).
  For testing purposes, `/admin/v1/ClientNonceGenerator(value)` exists to programmatically generate values that are encrypted using the tenant issuer public key. This process is equivalent to obtaining the public key from the public `/jwks` endpoint, and using a local SDK to encrypt the value using the public key.
- `{<secrets-alias>}xxxxx` : In-transit data encryption using the secret in the secrets store that is identified by `<secrets-alias>`
  - `rsakey` type: In-transit, PKI-based data encryption using the customer-supplied certificate of the RSA key referenced in `<secrets-alias>`.
  - `secret` type: In-transit, AES-based data encryption using the customer-supplied secret referenced in `<secrets-alias>`. The key must be a multiplier of 16, and IV must be equal to the first 16 characters of the key, with offset 0.
  - To generate a sample AES encryption payload, go to https://devtoolcafe.com/tools/aes.

Credentials being provided into a Bulk operation represent one of the following scenarios, operating on the credentials of the original `IMPORT` .

> **NOTE**
> More account-level attributes may be provided alongside the credentials.

- **POST**: Create an account with the given credentials.
- **PUT**: Create an account if it does not exist, or update the account if it exists, with the given credentials replacing existing ones.
- **PATCH**: Update the existing account according to the patch operations, including;
  - **ADD**: Add additional credentials.
  - **REPLACE**: Replace all existing credentials with the provided credentials, within the credential type.
  - **REMOVE**: Remove all existing credentials of the given type.

The following table describes the operations, their types, and the endpoint URLs:

Operation Description	Operation Types	Endpoint URLS
Bulk Import of User Credentials	POST	`https://${SSP_FQDN}/{{tenantName}}/scim2/vs/Bulk`
Bulk Import of Account Credentials (Existing identity sources, such as an ID Store (LDAP, SCIM, Internal, SiteMinder) or Identity Provider.)	POST	`https://${SSP_FQDN}/{{tenantName}}/scim2/vs/Bulk?identitySourceId=xxxx`
Bulk Import of Account Credentials (Logical identity source such as one represented by a ZFP identity)	POST	`https://${SSP_FQDN}/{{tenantName}}/scim2/vs/Bulk?identitySourceId=xxx1&identitySourceName=yy`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/scim2/vs/Bulk`
  should look similar to:
  `https://example.com/default/scim2/vs/Bulk`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

A sample for Add User Credentials:

Request:

```
POST https://${SSP_FQDN}/{{tenantName}}/scim2/vs/Bulk
{
    "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:BulkRequest"
    ],
    "Operations": [
        {
            "method": "PATCH",
            "path": "/Users/sofie_haley",
            "bulkId": "bulkPatch01",
            "data": {
                "schemas": [
                    "urn:ietf:params:scim:schemas:core:2.0:User",
                    "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User"
                ],
                "Operations": [
                    {
                        "op": "add",
                        "path": "emails",
                        "value": [
                            {
```

```
                                "value": "sofie_haley@work222.com",
                                "type": "work",
                                "verified": true,
                                "primary": true
                            },
                            {
                                "value": "sofie_haley@home222.com",
                                "display": "home-email-222",
                                "verified": true
                            }
                        ]
                    },
                    {
                        "op": "add",
                        "path": "phoneNumbers",
                        "value": [
                            {
                                "value": "000-000-2222",
                                "type": "work",
                                "verified": true,
                                "primary": false,
                                "capabilityType": "sms"
                            },
                            {
                                "value": "111-111-2222",
                                "display": "home-sms-2222",
                                "verified": true,
                                "capabilityType": "sms"
                            }
                        ]
                    },
                    {
                        "op": "add",
                        "path": "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User:devices",
                        "value": [
                            {
                                "value": "12345",
                                "display": "Work App - TOTP222",
                                "capabilityType": "totp",
                                "primary": true
                            },
                            {
                                "value": "98765",
                                "display": "Work App - PUSH222",
                                "capabilityType": "push"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```


Response:

```
{
    "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:BulkResponse"
    ],
    "Operations": [
        {
            "path": "/Users/sofie_haley",
            "bulkId": "bulkPatch01",
            "method": "PATCH",
            "location": "https://${SSP_FQDN}/{{tenantName}}//scim2/v1/Users/781bd6d0-
ef81-4304-9a76-544b8c60398a",
            "status": {
                "code": 200
            }
        }
    ]
}
```

A sample Replace/Remove User Credentials:

Request:

```
POST https://${SSP_FQDN}/{{tenantName}}/scim2/vs/Bulk
{
    "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:BulkRequest"
    ],
    "Operations": [
        {
            "method": "PATCH",
            "path": "/Users/sofie_haley",
            "bulkId": "bulkPatch01",
            "data": {
                "schemas": [
                    "urn:ietf:params:scim:schemas:core:2.0:User",
                    "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User"
                ],
                "Operations": [
                    {
                        "op": "replace",
                        "path": "emails",
                        "value": [
                            {
                                "value": "sofie_haley@personal333.com",
                                "type": "personal",
                                "verified": true,
                                "primary": true
                            }
                        ]
                    },
                    {
                        "op": "remove",
                        "path": "phoneNumbers"
                    },
                    {
```

```
                        "op": "replace",
                        "path": "urn:ietf:params:scim:schemas:extension:iam:ssp:1.0:User:devices",
                        "value": [
                            {
                                "value": "3333",
                                "display": "Personal App - TOTP333",
                                "capabilityType": "totp",
                                "primary": true
                            }
                        ]
                    }
                ]
            }
        }
    ]
}Response:
{
    "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:BulkResponse"
    ],
    "Operations": [
        {
            "path": "/Users/sofie_haley",
            "bulkId": "bulkPatch01",
            "method": "PATCH",
            "location": "https://${SSP_FQDN}/{{tenantName}}//scim2/v1/Users/781bd6d0-
ef81-4304-9a76-544b8c60398a",
            "status": {
                "code": 200
            }
        }
    ]
}
```

User Self Service API

User Self Service APIS include the SCIM /ME endpoint as well as additional Identity APIs for credential and email/phone verification.

The Self Service APIs expect the *"urn:iam:m.me"* scope, representing self service operations on an end user using the identity from the access token when the token's sub_type is user.

Operation Description	Operation Types	Endpoint URLs
User Self Service		
/Me	GET PUT PATCH	`https://${SSP_FQDN}/{{tenantName}}/scim2/v1/Me`
Get My Credentials	GET	`https://${SSP_FQDN}/{{tenantName}}/identity/v1/MeVerifiedCredentials`
Verify My Credentials		

Operation Description	Operation Types	Endpoint URLs
initiate email verification	POST	`https://${SSP_FQDN}/{{tenantName}}/identity/v1/MeEmailVerifier/identity/v1/MeEmailVerificationInitiator`
verify email	POST	`https://${SSP_FQDN}/{{tenantName}}/identity/v1/MeEmailVerifier/identity/v1/MeEmailVerifier`
initiate phone verification	POST	`https://${SSP_FQDN}/{{tenantName}}/identity/v1/MePhoneVerifier/identity/v1/MePhoneVerificationInitiator`
verify phone	POST	`https://${SSP_FQDN}/{{tenantName}}/identity/v1/MePhoneVerifier/identity/v1/MePhoneVerifier`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/scim2/v1/Me`
  should look similar to:
  `https://example.com/default/scim2/v1/Me`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

A Sample /ME Request:

```
Request:  GET https://${SSP_FQDN}/{{apiPathTenant}}/scimv2/v1/Me
```

Response:

```
{

    "displayName": "Brenda Steer",
    "userName": "bsteer”,
    "name": {
        "formatted": "Brenda Steer",
        "familyName": "Steer",
        "givenName": "Brenda"
    }
}
```

A sample Get My Credentials Request/Response:

```
Request:  GET https://${SSP_FQDN}/{{apiPathTenant}}/identity/v1/MeVerifiedCredentials
```

Response:

```
[
    {
        "displayValue": "xxxxxx5555",
        "deliveryType": "sms",
        "credId": "73c34afe-759f-4eab-8392-57f8327d5704",
```

```
        "type": "73c34afe-759f-4eab-8392-57f8327d5704"
    },
    {
        "displayValue": "edunn@example.com",
        "deliveryType": "email",
        "credId": "e927baba-9e8f-47c9-9438-fdb9ecd04bcd",
        "type": "e927baba-9e8f-47c9-9438-fdb9ecd04bcd"
    }
]
```

A sample Initiate Email Verification Request:

```
Request:  GET https://${SSP_FQDN}/{{apiPathTenant}}/identity/v1/MeEmailVerificationInitiator

{
{
    "email": "emily.russell@example.com",
    "language": "en_US"
}

}
```

Response:

```
{
    "token": "PZLgz6g3h7m9PJriTES4BM9M0rXpj53a"
}
```

Sample Verify Email Request/Response that uses the token from the Verification Initiator API:

```
Request: GET https://${SSP_FQDN}/{{apiPathTenant}}/identity/v1/MeEmailVerifier?
token=PZLgz6g3h7m9PJriTES4BM9M0rXpj53a
```

Response:

```
Status: 200 OK
```

Password Management API

The Password Management API provides lifecycle management of Password Policies, Password change operations, and SCIM-based Password Authentication.

Operation Description	Operation Types	Endpoint URLs
Password Management		
Password Policy	GET DELETE POST PUT	`https://${IDSP_FQDN}/{{tenantName}}/admin/v1/PasswordPolicies`
Password Dictionary	GET DELETE POST PUT	`https://${IDSP_FQDN}/{{tenantName}}/admin/v1/PasswordDictionary`
Validate Password	POST	`https://${IDSP_FQDN}/{{tenantName}}/identity/v1/UserPasswordValidator`

Operation Description	Operation Types	Endpoint URLs
Change Password	POST	`https://${IDSP_FQDN}/{{tenantName}}/identity/v1/UserPasswordChanger`
Generate Temporary Password	POST	`https://${IDSP_FQDN}/{{tenantName}}/identity/v1/UserPasswordGenerator`
Password Authenticator	POST	`https://${IDSP_FQDN}/{{tenantName}}/scim2/v1/PasswordAuthenticator`

NOTE

- To know the value of ${IDSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values in all the endpoints shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/PasswordPolicies`
  should look similar to:
  `https://example.com/default/admin/v1/PasswordPolicies`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

This topic contains the following subtopics:

- Determining the Password Policies that Apply
- Composition Rules
- Password Aging and History
- Default Password Policy
- Using Password Dictionary
- Administrative Password Management APIs

Determining the Password Policies that Apply

The Applicable policy to be applied for a given User is selected based on:

1) The directorySearchFilter match

2) The policy is in an "enabled" state

If multiple policies match the criteria, then the priority of the policy in Ascending order is applied. Only one policy is applied. The directorySearchFilter criteria currently support two values for searchObject(searchObj) that is, "ALL" (Acts like all User match) and "groups". The directorySearchFilter is a List of conditions that are evaluated to find the applicable policies.

The criteria for groups supports a list of groups where the equals operator is used to match the group name. The following sample criteria provides an example.

In this example, groups->"displayName eq Auditors" is **OR'ed** with groups->"displayName eq "Admin".

```
[
 {
   "criteria": [
     {
       "searchObj": "groups",
       "attribute": "displayName",
```

```
        "op": "eq",
        "value": "Auditors"
      },
      {
        "searchObj": "groups",
        "attribute": "displayName",
        "op": "eq",
        "value": "Admin"
      }
    ]
}
```

Composition Rules

The following composition-related fields comprise the password composition rules:

- Minimum password length (minLength)
  Specifies a minimum length for user passwords.
- Maximum password length (maxLength)
  Specifies the maximum length for user passwords.
- Maximum repeating characters (maxRepeatedChars)
  Determines the maximum number of identical characters that can appear consecutively in a password. For example, if this value is set to 3, then “aaaa” cannot appear anywhere in the password. However, “aaa” is acceptable within a password. Set this value to ensure that users cannot enter passwords of a single character.
- Minimum unique characters (minUniqueChars)
  Determines the minimum number of unique characters that must appear in a password.
- Upper case letters (minUpperCase)
  Specifies whether to allow upper case alphabetic characters and, if so, the minimum number a password must contain.
- Lower case letters (minLowerCase)
  Specifies whether to allow lower case alphabetic characters and, if so, the minimum number a password must contain.
- Non-alphanumeric/Alphanumeric (minNonAlphaNumerals/maxNonAlphaNumerals/minAlphaNumerals)
  Specifies whether to allow non-alphanumeric characters such as punctuation marks and other symbols (“@”, “$”, and “#”) and if so, the minimum/maximum number a password can contain.
- Allowed special characters (allowedSpecialChars)
  Specifies the allowed symbols (“@”, “$”, and “#”)
  Starting in 3.1.1, comma and double quotes must be escaped: Comma as \",\" and double quotes as \"\"\". These characters were erroneously allowed in previous releases.
  For example, (sample with comma):
  ```
  "allowedSpecialChars" : "@,$,\",\""
  ```
- Disallowed characters (disallowedChars)
  Specifies the disallowed symbols (“*”).
  Starting in 3.1.1, comma and double quotes must be escaped: Comma as \",\" and double quotes as \"\"\". These characters were erroneously allowed in previous releases.
  For example, (sample with double quote):
  ```
  "disallowedChars" : "#, \"\"\""
  ```
- Disallowed sub strings (disallowedSubstrings)
  Specifies the disallowed Sub strings that cannot be part of the password.
- Regex(policyRegex)
  Password regular expressions let you specify regular expression text patterns for matching or non-matching criteria. This test can be useful, for example, when you want to require that the first character is a digit, and the last character

is not. You configure multiple expressions for a single password policy. If you create multiple expressions, acceptable passwords must match all the expressions.

The following regular expression definition can be used to require that all passwords start with an upper or lower case letter:

```
Name: MustStartAlpha  Expression: [a-zA-Z].*
```

- Dictionary (dictionaryWordsDisallowed)
  Specifies whether the dictionary password verification is to be done as per the dictionary that is uploaded using the API /admin/v1/PasswordDictionary.
- Match Length (dictionaryMatchLength)
  Controls the length of strings that are compared against values in the dictionary file. The comparison ignores the case of the strings. You can leave the Match Length field blank or can set it to zero. In these cases, it only rejects passwords that match a string in the dictionary exactly. When the match length is greater than zero, rejects entries during the following conditions:
  – The password includes a substring which starts with the same series of characters as a dictionary entry.
  – The number of consecutive matching characters is greater than or equal to the number specified in the Match Length field.

  For example, consider a dictionary file that contains the following entries:
  – lion
  – tiger
  – bear

When the Match Length field is set to four results in the following actions:"TeddyBear", rejected because Bear matches the bear entry in the dictionary file."dictionary1", rejected because "tige" matches the first four characters of the tiger entry in the dictionary file."dictonary2", accepted since "iger" does not include the first letter of the tiger entry in the dictionary file.

Password Aging and History

The Password Policy supports a password age and previous password reuse rules.

- Password age in days (minPasswordAge)
  A value of 0 means that there is no password aging.
- Reuse of a previous password (numPasswordsInHistory).
  Specifies the previous number of passwords that cannot be re-used.
  For example, to allow password good for 30 days and you cannot re-use the last 5 passwords the following values would be

```
used.minPasswordAge=30numPasswordInHistory=5
```

Default Password Policy

A password policy deployed named “DefaultPasswordPolicy” is automatically deployed. The Policy defines the following defaults:

- Minimum Password length = 8
- 1 Uppercase letter
- 1 Lowercase letter
- 1 numeric character
- 1 special case character (NonAphaNumeral)

This policy cannot be disabled or deleted, it can only be updated.

Sample Create Password Policy API Request:

```
[
   {
       "name": "SamplePwdPolicy",
       "description": "sample password policy",
       "enabled": true,
       "priority": 100,
       "directorySearchFilter": [
           {
               "criteria": [
                   {
                       "searchObj": "ALL"
                   }
               ]
           }
       ],
       "minLength": 8,
       "maxLength": 20,
       "minAlphas": 0,
       "minNumerals": 1,
       "minUpperCase": 1,
       "minLowerCase": 1,
       "minUniqueChars": 2,
       "maxRepeatedChars": 0,
       "minAlphaNumerals": 0,
       "minNonAlphaNumerals": 1,
       "maxNonAlphaNumerals": 3,
       "allowedSpecialChars": "&,#,%,@,!,^",
       "numPasswordsInHistory": 3,
       "firstNameDisallowed": false,
       "userNameDisallowed": false,
       "lastNameDisallowed": false,
       "dictionaryWordsDisallowed": false,
       "dictionaryMatchLength": 0,
       "maxPasswordAge": 45
       "evaluatePasswordExpirationRulesOnAuthentication": false,
       "evaluatePasswordCompositionRulesOnAuthentication": false
   }
]
```

Using Password Dictionaries

An optional Dictionary file can be defined containing words that are not allowed to be used in passwords. The dictionary files are a text file with a single restricted word on each line. The following list is sample dictionary file content:

- password
- my
- me
- test
- cat
- dog

The following sample shows a Create Dictionary request. The Create dictionary API supports a form-data key=value pairs.

```
POST https://${IDSP_FQDN}/{{apiPathTenant}}/admin/v1/PasswordDictionary
```

```
DictionaryFiles=password_dict1.txt
Name=Sample Dictionary 1
Description=word list 1
```

Sample Create Dictionary response:


```
{
   "name": "Sample Dictionary 1",
   "description": "word list 1",
   "dictionaryFiles": [
       "password_dict1.txt"
   ],
   "policyDictionaryId": "d85b0833-8af3-42bf-8eed-ca4ed2272949",
   "createdBy": {
       "principalType": "CLIENT",
       "principalId": "2ad92f82-fdf3-4fdd-9205-9b81a037a2a3",
       "principalName": "DefaultTenantClient"
   },
   "createdDateTime": "2023-11-27 05:31 PM UTC"
}
```


Administrative Password Management APIs

The Administrative Password Management API support two parameters: Revoke Sessions and New-user.

The `revokeSessions` parameter can revoke a user's sessions and tokens:

API	Change Description	Request	Response
`UserPasswordChanger`	Revoke session support	`/identity/v1/UserPasswordChanger?revokeSessions=<true/false>`	Unchanged: the API now supports all the existing Sessions to be revoked for the user when it is invoked with the request Parameter `revokeSessions` set as true.
`UserPasswordGenerator`	Revoke session support	`/identity/v1/UserPasswordGenerator?revokeSessions=<true/false>`	Unchanged The API now supports all the existing Sessions to be revoked for the user when it is invoked with the request Parameter `revokeSessions` set as true.
`PasswordResetInitiator`	Revoke session support	`/identity/v1/PasswordResetInitiator?revokeSessions=<true/false>`	Unchanged The API now supports all the existing Sessions to be revoked for the user when it is invoked with the request Parameter `revokeSessions` set as true.
	New user support	`/identity/v1/PasswordResetInitiator?newUser=true`	Unchanged The API supports New User welcome message templates that are used for email and SMS delivery

API	Change Description	Request	Response
PasswordResetChanger	Revoke session support	/identity/v1/PasswordResetChanger?token=<reset-token>&revokeSessions=<true / false>	Unchanged The API now supports all the existing Sessions to be revoked for the user when it is invoked with the request Parameter revokeSessions set as true.

Scoping Constraints API

The Scoping Constraints API provides CRUD operations to define, manage, and control attribute-level access permissions for user resources, requiring the urn:iam:t.adminroles scope for all operations.

The following table shows the Scoping Constraints APIs endpoint and the supported operations. The Scoping Constraints CRUD APIs expect the urn:iam:t.adminroles scope for all operations.

Operation Description	Operation Types	Endpoint URLs
Define Scoping Constraints	POST	https://${SSP_FQDN}/{{tenantName}}/admin/v1/ScopingConstraints
Update the Scoping Constraints (replace all required attributes)	PUT	https://${SSP_FQDN}/{{tenantName}}/admin/v1/ScopingConstraints/{{constraintId}}
Update the Scoping Constraints (replace specific attributes)	PATCH	https://${SSP_FQDN}/{{tenantName}}/admin/v1/ScopingConstraints/{{constraintId}}
Retrieve the Scoping Constraints by ID	GET	https://${SSP_FQDN}/{{tenantName}}/admin/v1/ScopingConstraints/{{constraintId}}
Retrieve all the Scoping Constraint definitions	GET	/{{tenantName}}/admin/v1/ScopingConstraints
Delete the Scoping Constraints	DELETE	https://${SSP_FQDN}/{{tenantName}}/admin/v1/ScopingConstraints/{{constraintId}}
Scoping Constraints Map Helper	Deprecated	
Scoping Constraints List by ID Helper	Deprecated	

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for <tenantName> is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  https://{{host}}/{{tenantName}}/admin/v1/ScopingConstraints
  should look similar to:
  https://example.com/default/admin/v1/ScopingConstraints
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

A sample Create Scoping Constraints Request:

```
POST https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/ScopingConstraints

{
   "name": "UMSupport1",
   "description": "Internal Helpdesk",
   "resource": "urn:res:iam:user",
   "data": [
       {
           "attr": "*",
           "priv": "r"
       },
       {
           "attr": "user_loginid",
           "priv": "rw"
       },
       {
           "attr": "is_locked",
           "priv": "w"
       },
             {
           "attr": "family_name",
           "priv": "rw"
       }
   ]
}
```

Create Scoping Constraints Response:

```
{
   "constraintId": "09669f93-583b-4e02-bdd2-d3fc69e9d4c2",
   "name": "UMSupport1",
   "description": "Internal Helpdesk",
   "resource": "urn:res:iam:user",
   "data": [
       {
           "attr": "*",
           "priv": "r"
       },
       {
           "attr": "user_loginid",
           "priv": "rw"
       },
       {
           "attr": "is_locked",
           "priv": "w"
       },
       {
           "attr": "family_name",
           "priv": "rw"
       }
   ],
   "createdBy": {
       "principalType": "CLIENT",
       "principalId": "0987f113-dca9-4c70-b35d-cbe5a63d7ab7",
```

```
        "principalName": "DefaultTenantClient"
    },
    "createdDateTime": "2024-03-07 12:33 PM UTC"
}
```

A sample Update Scoping Constraints Request (PUT):

```
PUT https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/ScopingConstraints/{{constraintId}}

{
    "name": "UMSupport1",
    "description": "Internal Helpdesk",
    "resource": "urn:res:iam:user",
        "data": [
            {
            "attr": "*",
            "priv": "r"
            },
            {
                "attr": "given_name",
                "priv":"wr"
            },
            {
                "attr": "family_name",
                "priv": "rw"
            },
            {
                "attr": "is_locked",
                "priv": "wr"
            },
            {
                "attr": "email",
                "priv": "rw"
            },
            {
                "attr": "address",
                "priv": "rw"
            },
            {
                "attr": "birthdate",
                "priv": "r"
            },
            {
                "attr": "locale",
                "priv": "w"
            },
            {
                "attr": "address_verified",
                "priv": "rw"
            },
            {
                "attr": "phone_number",
```

```
                "priv": "wr"
            },
            {
                "attr": "nameFormatted",
                "priv": "rw"
            },
            {
                "attr": "nickname",
                "priv": "rw"
            }
        ]
  }
```

A sample Update Scoping Constraints Response (PUT):

```
{
   "constraintId": "09669f93-583b-4e02-bdd2-d3fc69e9d4c2",
   "name": "UMSupport1",
   "description": "Internal Helpdesk",
   "resource": "urn:res:iam:user",
   "data": [
        {
            "attr": "*",
            "priv": "r"
        },
        {
            "attr": "given_name",
            "priv": "wr"
        },
        {
            "attr": "family_name",
            "priv": "rw"
        },
        {
            "attr": "is_locked",
            "priv": "wr"
        },
        {
            "attr": "email",
            "priv": "rw"
        },
        {
            "attr": "address",
            "priv": "rw"
        },
        {
            "attr": "birthdate",
            "priv": "r"
        },
        {
            "attr": "locale",
            "priv": "w"
        },
        {
```

```
            "attr": "address_verified",
            "priv": "rw"
        },
        {
            "attr": "phone_number",
            "priv": "wr"
        },
        {
            "attr": "nameFormatted",
            "priv": "rw"
        },
        {
            "attr": "nickname",
            "priv": "rw"
        }
    ],
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "0987f113-dca9-4c70-b35d-cbe5a63d7ab7",
        "principalName": "DefaultTenantClient"
    },
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "0987f113-dca9-4c70-b35d-cbe5a63d7ab7",
        "principalName": "DefaultTenantClient"
    },
    "createdDateTime": "2024-03-07 12:33 PM UTC",
    "updatedDateTime": "2024-03-07 12:38 PM UTC"
}
```


A sample Update Scoping Constraints Request (PATCH):

```
PATCH https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/ScopingConstraints/{{constraintId}}
```


```
{
    "name": "UMSupport1",
    "description": "Internal Helpdesk",
    "resource": "urn:res:iam:user",
    "abacOps": "CRUD"
  }
```


A sample Update Scoping Constraints Response (PATCH):


```
{
    "constraintId": "f7abf23a-4fce-4cf3-80d9-1042814cb740",
    "name": "InternalAdmin",
    "description": "Internal Helpdesk",
    "resource": "urn:iam:res:user",
    "abacOps": "CDRU",
    "gbacOps": null,
    "status": "inactive",
    "adminScopeApi": false,
```

```
    "disableOwner": false,
    "data": [
        {
            "attr": "*",
            "priv": "rw"
        }
    ],
    "instance": [
        {
            "attr": "userType",
            "scope": [
                {
                    "op": "eq",
                    "value": "Employee"
                }
            ]
        }
    ]
   }
```

A sample GET Scoping Constraints Request:

```
GET https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/ScopingConstraints/{{constraintId}}

Response:
{
   "constraintId": "09669f93-583b-4e02-bdd2-d3fc69e9d4c2",
   "name": "UMSupport1",
   "description": "Internal Helpdesk",
   "resource": "urn:res:iam:user",
   "data": [
       {
           "attr": "*",
           "priv": "r"
       },
       {
           "attr": "given_name",
           "priv": "wr"
       },
       {
           "attr": "family_name",
           "priv": "rw"
       },
       {
           "attr": "is_locked",
           "priv": "wr"
       },
       {
           "attr": "email",
           "priv": "rw"
       },
       {
           "attr": "address",
```

```
            "priv": "rw"
        },
        {
            "attr": "birthdate",
            "priv": "r"
        },
        {
            "attr": "locale",
            "priv": "w"
        },
        {
            "attr": "address_verified",
            "priv": "rw"
        },
        {
            "attr": "phone_number",
            "priv": "wr"
        },
        {
            "attr": "nameFormatted",
            "priv": "rw"
        },
        {
            "attr": "nickname",
            "priv": "rw"
        }
    ],
    "createdBy": {
        "principalType": "CLIENT",
        "principalId": "0987f113-dca9-4c70-b35d-cbe5a63d7ab7",
        "principalName": "DefaultTenantClient"
    },
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "0987f113-dca9-4c70-b35d-cbe5a63d7ab7",
        "principalName": "DefaultTenantClient"
    },
    "createdDateTime": "2024-03-07 12:33 PM UTC",
    "updatedDateTime": "2024-03-07 12:38 PM UTC"
}
```


A sample GET ALL Scoping Constraints Request:

```
GET https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/ScopingConstraints

Response:

[
    {
        "constraintId": "09669f93-583b-4e02-bdd2-d3fc69e9d4c2",
        "name": "UMSupport1",
        "description": "Internal Helpdesk",
```

```
"resource": "urn:res:iam:user",
"data": [
    {
        "attr": "*",
        "priv": "r"
    },
    {
        "attr": "given_name",
        "priv": "wr"
    },
    {
        "attr": "family_name",
        "priv": "rw"
    },
    {
        "attr": "is_locked",
        "priv": "wr"
    },
    {
        "attr": "email",
        "priv": "rw"
    },
    {
        "attr": "address",
        "priv": "rw"
    },
    {
        "attr": "birthdate",
        "priv": "r"
    },
    {
        "attr": "locale",
        "priv": "w"
    },
    {
        "attr": "address_verified",
        "priv": "rw"
    },
    {
        "attr": "phone_number",
        "priv": "wr"
    },
    {
        "attr": "nameFormatted",
        "priv": "rw"
    },
    {
        "attr": "nickname",
        "priv": "rw"
    }
],
"createdBy": {
    "principalType": "CLIENT",
```

```
            "principalId": "0987f113-dca9-4c70-b35d-cbe5a63d7ab7",
            "principalName": "DefaultTenantClient"
        },
        "updatedBy": {
            "principalType": "CLIENT",
            "principalId": "0987f113-dca9-4c70-b35d-cbe5a63d7ab7",
            "principalName": "DefaultTenantClient"
        },
        "createdDateTime": "2024-03-07 12:33 PM UTC",
        "updatedDateTime": "2024-03-07 12:38 PM UTC"
    },
    {
        "constraintId": "14078fef-4a0b-4b21-aaaa-caee3ef58536",
        "name": "UMFinance",
        "description": "Finance Helpdesk",
        "resource": "urn:res:iam:user",
        "data": [
            {
                "attr": "*",
                "priv": "r"
            },
            {
                "attr": "user_loginid",
                "priv": "rw"
            },
            {
                "attr": "is_locked",
                "priv": "w"
            },
            {
                "attr": "family_name",
                "priv": "rw"
            }
        ],
        "createdBy": {
            "principalType": "CLIENT",
            "principalId": "0987f113-dca9-4c70-b35d-cbe5a63d7ab7",
            "principalName": "DefaultTenantClient"
        },
        "createdDateTime": "2024-03-07 12:45 PM UTC"
    }
]
```


A Sample Delete Scoping Constraints Request:

```
DELETE https://${SSP_FQDN}/{{apiPathTenant}}/admin/v1/ScopingConstraints/{{constraintId}}

Response:

Status: 204 No Content
```

Delegations APIs

The Delegation API allows Administrators to delegate their privileges and access to others. The API covers the ability to manage (add, remove) and view delegations. The APIs support delegation of Admin Roles and Resources supported by Resource Authorization capabilities.

Adding or removing a delegation is subject to administrative scope checks. When attempting to delegate a role or a resource to a subject, the administrator must have administrative scope over that subject or else the delegation is denied. However, if the administrator initiated the delegation attempts to remove a delegation, no administrative scope check is required; you can remove what you delegated.

Additionally, if an administrator has the following privileges, no administrative scope checks are enforced:

- urn:iam:t.all
- urn:iam:t.adminroles

You can manage (add or remove) and view delegations directly through the Delegation APIs:

> **NOTE**
> You can also view and remove the roles and resources assigned by you or to you from the Profiles page of the Admin Console. See Profiles.

Table 117: Delegation APIs

Operation	Operation Type	Endpoint URL	
Add an Admin Role delegation	POST	https://${SSP_FQDN}/{{tenantName}}/identity/AdminRolesSubdelegator	Adding an Admin Role Delegation (AdminRoleSubdelegator)
Remove an Admin Role delegation	DELETE	https://${SSP_FQDN}/{{tenantName}}/identity/AdminRolesSubdelegator	Removing an Admin Role Delegation (AdminRoleSubdelegator)
Add a resource delegation	POST	https://${SSP_FQDN}/{{tenantName}}/identity/ResourceSubdelegator	Adding a Resource Delegation (ResourceSubdelegator)
Remove a resource delegation	DELETE	https://${SSP_FQDN}/{{tenantName}}/identity/ResourceSubdelegator	Removing a Resource Delegation (ResourceSubdelegator)
View My Admin Role delegations	GET	https://${SSP_FQDN}/{{tenantName}}/identity/MeSubdelegations	Viewing Delegations
View My Resource delegations	GET	https://${SSP_FQDN}/{{tenantName}}/identity/MeSubdelegations	
iew delegations by user name or user group	GET	https://${SSP_FQDN}/{{tenantName}}/identity/Subdelegations	

Adding an Admin Role Delegation (AdminRoleSubdelegator)

Use the AdminRoleSubdelegator API to delegate administrator roles. The AdminRoleSubdelegator API supports the delegation of multiple roles to multiple subjects.

Sample Add AdminRoleSubdelegator Request

The following example is a sample AdminRoleSubdelegator request that delegates the Partner Administration Role to three subjects.

```
{
   "roles": [
         "Partner Administration Role"
    ],
    "subjects": {
           "users": [
                 "dmartin",
                 "mmasterson"
            ],
            "groups": [
                "partners"
            ],
            "clients": [
                "partnerClient"
            ]
        }
}
```

Sample Add AdminRoleSubdelegator Response

The following example is a sample response to the AdminRoleSubdelegator request. If the delegation succeeds, the `delegationId` for each subject provided in the request is returned in the response. The `delegationId` can be used to revoke the delegation.

```
{
    "users": [
        "140a974e-ada5-46ed-9cee-a99db74c25f9"
     ],
    "groups": [
        "ee64dd91-5c67-41c9-a415-56ed45fa921d"
    ],
    "clients": [
         "ea0e04e9-89ab-4ada-a828-c2ad8b62072d"
      ]
}
```

Removing an Admin Role Delegation (AdminRoleSubdelegator)

Use the AdminRoleSubdelegator API to remove delegated administrator roles. The AdminRoleSubdelegator API supports the removal of multiple roles from multiple subjects.

Remove AdminRoleSubdelegator Request

In the following example request, an administrator removes the Partner Administration Role previously delegated to three subjects.

```
{
    "subdelegations":[
        "140a974e-ada5-46ed-9cee-a99db74c25f9",
        "ee64dd91-5c67-41c9-a415-56ed45fa921d",
        "ea0e04e9-89ab-4ada-a828-c2ad8b62072d"
    ]
}
```

Remove AdminRoleSubdelegator Response

If the delegation removal request succeed, the response returns the code 204. All other responses indicate an error.

Adding a Resource Delegation (ResourceSubdelegator)

Use the ResourceSubdelegator API to delegate resources (such as an application or group) to a manager (administrator). The ResourceSubdelegator API supports the delegation of multiple resources to multiple subjects.

The following designated resource types are supported for Resource delegation:

- urn:iam:res:app
- urn:iam:res:authnpolicy
- urn:iam:res:authzpolicy
- urn:iam:res:cert
- urn:iam:res:group
- urn:iam:res:idp
- urn:iam:res:idppolicy
- urn:iam:res:secret

ResourceSubdelegator Request

In the following example request, an administrator delegates the PartnerGroup resource to three subjects.

```
{
   Resources: [
    {"urn:iam:res:group": "PartnerGroup"}
   ]
    "subjects": {
           "users": [
                 "dmartin",
                 "mmasterson"
            ],
            "groups": [
                "partners"
            ],
            "clients": [
                "partnerClient"
            ]
        }
}
```

ResourceSubdelegator Response

The following example is a sample response to the ResourceSubdelegator request. If the delegation succeeds, the `delegationId` for each subject provided in the request is returned in the response. The `delegationId` can be used to revoke the delegation.

```
{
    "users": [
        "ae9778b5-b1d1-4c0e-b83e-f709b825c977"
     ],
    "groups": [
        "cec58c74-08aa-45a9-8857-54973c0ea93a"
    ],
    "clients": [
        "fcd9e04e9-89ab-4ada-a828-c2ad8b62072d"
```

```
        ]
}
```

Removing a Resource Delegation (ResourceSubdelegator)

Use the ResourceSubdelegator API to remove delegated resources. The ResourceSubdelegator API supports the removal of multiple resources from multiple subjects.

Remove AdminRoleSubdelegator Request

In the following example request, an administrator removes the PartnerGroup resource previously delegated to three subjects.

```
{
    "subdelegations":[
        "ae9778b5-b1d1-4c0e-b83e-f709b825c977",
        "cec58c74-08aa-45a9-8857-54973c0ea93a",
        "fcd9e04e9-89ab-4ada-a828-c2ad8b62072d"
    ]
}
```

Remove AdminRoleSubdelegator Response

If the delegation removal request succeed, the response returns the code 204. All other responses indicate an error.

Viewing Delegations

You can view the following delegations:

- View Admin Roles delegated to or delegated by yourself using the MeSubdelegations API. See Viewing My Admin Role Delegations (MeSubdelegations).
- View resources delegated to or delegated by yourself using the MeSubdelegations API. See Viewing My Admin Resource Delegations (MeSubdelegations).
- View the delegations made by or to any user, group, or principal using the Subdelegations API. See Viewing Delegations (Subdelegations).

The following example is a summary Delegations request body:

```
{
    "adminScope": "adminrole | managedby",
    "principal":  "<user-id or app-id>"   // Note: not supported with Get My Delegations
    "subject": {
        "type": "user | group | client",
        "id": "<user-id or group-id or app-id>"
    },
    "resource": {
        "type": "<resource-URN>",
        "id": "<resource-id>"
    }
}
```

Viewing My Admin Role Delegations (MeSubdelegations)

Use the MeSubdelegations API to view the roles that you delegated to others (the grantor) as well as the roles that were delegated to you (the grantee).

```
{
    "adminScope" : "adminrole"
}
```

The following sample is an example response to the MeSubdelegations API for Admin Roles.


```
{
    "grantor": [
        {
            "id": "ea0e04e9-89ab-4ada-a828-c2ad8b62072d",
            "principalId": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
            "principalType": "USER",
            "principalName": "Tim Richards",
            "subject": {
                "type": "user",
                "id": "d3b9ccc8-2e73-4760-8343-f32f1afa98fc",
                "name": "mmasterson",
                "displayName": "Michelle Masterson"
            },
            "resourceId": "e4ad353a-312c-46d4-8893-93de85640e42",
            "resourceType": "urn:iam:res:adminrole",
            "resourceName": "Banking Resource Role"
        },
        {
            "id": "140a974e-ada5-46ed-9cee-a99db74c25f9",
            "principalId": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
            "principalType": "USER",
            "principalName": "Tim Richards",
            "subject": {
                "type": "user",
                "id": "7e677043-1bd0-45cf-b8e3-65fd2e0bf813",
                "name": "dmartin",
                "displayName": "Diane Martin"
            },
            "resourceId": "e4ad353a-312c-46d4-8893-93de85640e42",
            "resourceType": "urn:iam:res:adminrole",
            "resourceName": "Banking Resource Role"
        }
    ],
    "grantee": [
        {
            "id": "58920f84-de43-4747-b978-96b4e372de89",
            "principalId": "791cee1e-00af-48b1-8a47-78aa17d92f63",
            "principalType": "USER",
            "principalName": "Edward Morris",
            "subject": {
                "type": "user",
                "id": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
                "name": "trichards",
                "displayName": "Tim Richards"
            },
            "resourceId": "c5fdab41-8eeb-4861-86ae-a55452b1d14c",
            "resourceType": "urn:iam:res:adminrole",
            "resourceName": "Integration Administrator"
        }
    ]
}
```

Viewing My Admin Resource Delegations (MeSubdelegations)

Use the MeSubdelegations to view the resources that you delegated to others (the grantor) as well as the resources that were delegated to you (the grantee). The following is a sample query for the resource delegations for the principal user, *trichards*.

```
{
    "adminScope" : "adminrequest"
}
```

The following sample is an example response to the MeSubdelegations API for resources.

```
{
    "grantor": [
        {
            "id": "ae9778b5-b1d1-4c0e-b83e-f709b825c977",
            "principalId": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
            "principalType": "USER”
            "principalName": "Tim Richards",
            "subject": {
                "type": "user",
                "id": "c2e02e4f-40c4-4a3d-b0d5-299eff7a4686",
                "name": "jmarshall",
                "displayName": "James Marshall"
            },
            "resourceId": "f19f4b22-01e0-4f73-933c-d3474225e3cb",
            "resourceType": "urn:iam:res:group",
            "resourceName": "Partners"
        },
        {
            "id": "cec58c74-08aa-45a9-8857-54973c0ea93a",
            "principalId": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
            "principalType": "USER",
            "principalName": "Tim Richards",
            "subject": {
                "type": "user",
                "id": "f2bcd19a-9dd3-48dc-8023-8ce8a02f4120",
                "name": "lmasterson",
                "displayName": "Linda Masterson"
            },
            "resourceId": "f19f4b22-01e0-4f73-933c-d3474225e3cb",
            "resourceType": "urn:iam:res:group",
            "resourceName": "Partners"
        }
    ],
    "grantee": [
        {
            "id": "9ba279f9-74d0-4616-8ec9-b2494531192b",
            "principalId": "45dfe8f9-0b88-4640-864f-0565cd80aabe",
            "principalType": "CLIENT",
            "principalName": "InvestmentsClient",
            "subject": {
                "type": "user",
                "id": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
                "name": "trichards",
                "displayName": "Tim Richards"
```

```
            },
            "resourceId": "d9103143-a2cd-4e47-80ce-2dd3af4bee53",
            "resourceType": "urn:iam:res:idp",
            "resourceName": "InvestmentsIDP-D"
        },
         {
            "id": "bac983ce-f747-444d-abed-a674a5536f85",
            "principalId": "f1e1171a-9e2e-40d6-9d18-ac170c725a11",
            "principalType": "USER",
            "principalName": "James Holden",
            "subject": {
                "type": "user",
                "id": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
                "name": "trichards",
                "displayName": "Tim Richards"
            },
            "resourceId": "295dc751-25e7-4d97-8c2a-9acac51a39d6",
            "resourceType": "urn:iam:res:app",
            "resourceName": "InvestmentsAppB"
         }
     ]
 }
```

Viewing Delegations (Subdelegations)

Use the Subdelegations API to view the delegations made by any user, group, or principal, (the grantor) as well as the delegations made to any user, group, or principal (the grantee).

- **Viewing Delegations for a Group**
  The following is a sample query to view the delegations for a group.

```
 {
     "adminScope" : "managedby",
      "resource" : {
      "type": "urn:iam:res:group",
       "id": "BankingCustomerGroup"
     }
 }
```

- **Viewing Delegations for a User**
  The following is a sample query to view the delegations for a user.

```
 GET /identity/v1/MeSubdelegations
 Request:
 {
     "adminScope" : "adminrequest"
 }
```

- **Viewing Delegations for a Principal**
  The following is a sample query to view the delegations for a principal.

```
 GET /identity/v1/MeSubdelegations
 Request:
 {
     "adminScope" : "adminrequest"
 }
```

- **Sample Viewing Delegations for a User (Subdelegations) Response**
  The following sample is an example response to the Subdelegations request for a user, showing all of the delegations where *trichards* is the subject.


```
{
    "id": "3607084a-63d4-4b81-9df8-ff76e0d76bd0",
    "principalId": "45dfe8f9-0b88-4640-864f-0565cd80aabe",
    "principalType": "CLIENT",
    "principalName": "DefaultTenantClient",
    "subject": {
        "type": "user",
        "id": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
        "name": "trichards",
        "displayName": "Tim Richards"
    },
    "resourceId": "f19f4b22-01e0-4f73-933c-d3474225e3cb",
    "resourceType": "urn:iam:res:group",
    "resourceName": "RADemo-Partners"
},
{
    "id": "58920f84-de43-4747-b978-96b4e372de89",
    "principalId": "791cee1e-00af-48b1-8a47-78aa17d92f63",
    "principalType": "USER",
    "principalName": "Edward Morris",
    "subject": {
        "type": "user",
        "id": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
        "name": "trichards",
        "displayName": "Tim Richards"
    },
    "resourceId": "c5fdab41-8eeb-4861-86ae-a55452b1d14c",
    "resourceType": "urn:iam:res:adminrole",
    "resourceName": "Integration Administrator"
},
{
    "id": "9ba279f9-74d0-4616-8ec9-b2494531192b",
    "principalId": "45dfe8f9-0b88-4640-864f-0565cd80aabe",
    "principalType": "CLIENT",
    "principalName": "DefaultTenantClient",
    "subject": {
        "type": "user",
        "id": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
        "name": "trichards",
        "displayName": "Tim Richards"
    },
    "resourceId": "d9103143-a2cd-4e47-80ce-2dd3af4bee53",
    "resourceType": "urn:iam:res:idp",
    "resourceName": "RADemo-InvestmentsIDP-D"
},
{
    "id": "adc686eb-4661-4567-bb9a-84c53408c616",
```

```
        "principalId": "f1e1171a-9e2e-40d6-9d18-ac170c725a11",
        "principalType": "USER",
        "principalName": "James Holden",
        "subject": {
            "type": "user",
            "id": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
            "name": "trichards",
            "displayName": "Tim Richards"
        },
        "resourceId": "ab2134d7-73a3-4d59-8051-00548de98a25",
        "resourceType": "urn:iam:res:group",
        "resourceName": "RADemo-Employees"
    },
    {
        "id": "bac983ce-f747-444d-abed-a674a5536f85",
        "principalId": "f1e1171a-9e2e-40d6-9d18-ac170c725a11",
        "principalType": "USER",
        "principalName": "James Holden",
        "subject": {
            "type": "user",
            "id": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
            "name": "trichards",
            "displayName": "Tim Richards"
        },
        "resourceId": "295dc751-25e7-4d97-8c2a-9acac51a39d6",
        "resourceType": "urn:iam:res:app",
        "resourceName": "RADemo-InvestmentsAppB"
    },
    {
        "id": "e564c10d-a987-4f7c-b257-226decee671c",
        "principalId": "f1e1171a-9e2e-40d6-9d18-ac170c725a11",
        "principalType": "USER",
        "principalName": "James Holden",
        "subject": {
            "type": "user",
            "id": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
            "name": "trichards",
            "displayName": "Tim Richards"
        },
        "resourceId": "77db37c1-66bf-4c82-8356-c89f4c9fb690",
        "resourceType": "urn:iam:res:app",
        "resourceName": "RADemo-InvestmentsAppA"
    },
    {
        "id": "f3bb8a65-404e-4a17-a0a7-6301be1a1f3e",
        "principalId": "45dfe8f9-0b88-4640-864f-0565cd80aabe",
        "principalType": "CLIENT",
        "principalName": "DefaultTenantClient",
        "subject": {
            "type": "user",
            "id": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
            "name": "trichards",
            "displayName": "Tim Richards"
```

```
            },
            "resourceId": "9a59034b-3944-40a5-bb56-46db4b6df4c6",
            "resourceType": "urn:iam:res:app",
            "resourceName": "RADemo-BankingAppD"
        },
        {
            "id": "f642ae1f-85ea-46af-85a0-f82ec7086857",
            "principalId": "f1e1171a-9e2e-40d6-9d18-ac170c725a11",
            "principalType": "USER",
            "principalName": "James Holden",
            "subject": {
                "type": "user",
                "id": "7d4d3ab6-a8f3-4f3f-856c-24d6e205672c",
                "name": "trichards",
                "displayName": "Tim Richards"
            },
            "resourceId": "9251e9db-735a-45f4-a372-071bdfee19d1",
            "resourceType": "urn:iam:res:group",
            "resourceName": "RADemo-UserAdministrators"
        }
    ]
```


Helper APIs

The Admin Service supports Helper APIs that allow clients to both retrieve information and execute capabilities ancillary to the core APIs of the Identity Service.

This section describes the Identity Service Helper APIs:

- Me Schema Helper API
- User and Group Schema Helper APIs
- Resource Schema Helper API
- Username Generation Helper API
- Referencing Objects Helper API
- Revoke All App Roles Helper
- Access Helper API
- Resource Counter Helper API

Me Schema Helper API

The Me Schema Helper API allows clients to retrieve their User schema (the attributes that the client is allowed to access based on the constraints applied to the client).

Admin Scope	Operation Type	
urn:iam:m.me	GET	`https://${SSP_FQDN}/{{tenantName}}/identity/v1/Me`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.

For example:
`https://{{host}}/{{tenantName}}/admin/v1/MeSchemaHelper`
should look similar to:
`https://example.com/default/admin/v1/MeSchemaHelper`

- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

The Me Schema Helper API returns the full supported schemas of the User objects for the calling client. The schema information contains schema attributes describing the attribute names, URN, and various characteristics of the attributes, such as the read/write permissions. Parameters can be passed to the Schema Helper API to indicate the ID Store identifier, to allow for searches for specific schema attributes, and to provide a count of the objects returned. The following table displays these parameters:

NOTE
The '**values**' attribute returned in responses to the Me Schema Helper API is deprecated.

Parameter Name	Parameter Value	Usage
`configId`	`<idstore-configid>`	The identifier of the ID Store to be used. For the Internal Id Store, use the value `internal` .
`filter`	A filter of the form: `?filter=<schemaAttribute> eq <value>`	Apply a filter to return the attributes that match the `schemaAttribute` filter criteria. Example: `filter?searchable eq true and visible eq true`
`countonly`	`countonly`	**true** – `countonly` is enabled. **false** – `countonly` is disabled (default).

User and Group Schema Helper APIs

The User and Group Schema Helper APIs allow clients to retrieve the full schema of User and Group objects.

The User and Group Schema Helper APIs consist of two separate APIs:

- User Schema Helper API
- Group Schema Helper API

Operation Description	Admin Scope	Operation Type	
User Schema Helper	urn:iam:t.users urn:iam:t.identityclient	GET	`https://${SSP_FQDN}/{{tenantNam`
Group Schema Helper	urn:iam:t.groups urn:iam:t.identityclient	GET	`/{{tenantName}}/admin/v1/GroupS`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/UserSchemaHelper`
  should look similar to:
  `https://example.com/default/admin/v1/UserSchemaHelper`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

The Schema Helper APIs return the full supported schemas of the User and Group objects. The schema information contains schema attributes describing the attribute names, URN for SCIM, and various characteristics of the attributes. Parameters can be passed to the Schema Helper APIs to indicate the ID Store identifier, to allow for searches for specific schema attributes, and to provide the number of objects returned. The following table displays these parameters:

NOTE
The '**values**' attribute returned in responses to the User Schema Helper API and the Group Schema Helper API is deprecated.

Parameter Name	Parameter Value	Usage
`configId`	`<idstore-configid>`	The identifier of the ID Store to be used. For the Internal Id Store, use the value `internal` .
`filter`	A filter of the form: `?filter=<schemaAttribute> eq <value>` The `filter` parameter is only applicable for the User Schema Helper and is not applicable for the Group Schema Helper.	Apply a filter to return the attributes that match the `schemaAttribute` filter criteria. Example: `filter?searchable eq true and visible eq true`
`countonly`	`countonly`	true – `countonly` is enabled. false – `countonly` is disabled (default).

Resource Schema Helper API

Returns information about which attributes you can use in searches for a specified resource.

Admin Scope	Operation Type	
urn:iam:t.apps urn:iam:t.authzpolicies urn:iam:t.authpolicies urn:iam:t.idpconfigs urn:iam:t.secrets urn:iam:t.certs urn:iam:t.idpdiscoverypolicies	GET	`https://${SSP_FQDN}/{{tenantName}}/admi`

The query returns all searchable attributes for the specified resource type.

NOTE
The '**values**' attribute returned in responses to the Resource Schema Helper API is deprecated.

Parameter Name	Parameter Value	Resource Type
Resource	app	Application
	idp	Identity Provider
	cert	Certification
	secret	Secret
	authnpolicy	Authentication policy
	authzpolicy	Authorization policy
	idppolicy	Identity Provider Discovery policy
	ldapidstore	LDAP Identity Store
	scimidstore	SCIM Identity Store

For example, the following query returns a list of all applications whose descriptions include the string “Default”.

```
https://${SSP_FQDN}/{{tenantName}}/admin/v1/ResSchemaHelper/Apps?filter=description co Default
```

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/ResSchemaHelper`
  should look similar to:
  `https://example.com/default/admin/v1/ResSchemaHelper`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Username Generation Helper API

APIs for retrieving username generation rules, validating rule configurations, and generating usernames based on configured rules and user attributes.

The following table describes the Username Generation Helper APIs. All Username Generation Helper APIs use the same endpoint URL:

```
https://${SSP_FQDN}/{{tenantName}}/admin/v1/UserNameGenerationHelper
```

Operation Description	Admin Scope	Operation Type	
Username Generation Helper	urn:iam:t.users	GET	Getting configured username ge
Username Generation Helper (Validate)		POST	Validating username generation
Username Generation Helper (Generate)		POST	Generating usernames

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values in all the endpoints shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/UserNameGenerationHelper`
  should look similar to:
  `https://example.com/default/admin/v1/UserNameGenerationHelper`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Username Generation Rules

The username generation rule can be formed with the combination of one or more of the following components, separated by '+' character:

- STRING-LITERAL:
  Include any string that is provided within single or double quotes. If there are any special characters provided in `STRING-LITERAL` then the rule is validated as successful. However, the special characters **$, ( ) { } "** and **'** are removed when generating the user name.
- N
  Add a random number with specified number of digits. The character **N** denotes a digit. A two-digit number is represented as **NN**. Similarly, a four-digit number is represented as **NNNN**. All **N** characters must be consecutive, without spaces in between. A random number is generated which contains the number of **N** characters provided. For example, if the rule string is `user + NNN` , then the rule generates usernames such as *user234*, *user730*, *user649*, and so on.
- VARIABLE
  Dynamically substitute values from the user attributes. Variable can be represented in the following formats:
  - `${idprofile.users-logical-attr-name}`
  - `${idprofile.users-logical-attr-name} (start-index,length)`
  - `${idprofile.users-logical-attr-name} (start-index,)`

  Where:
  - `idprofile` is the namespace to indicate processing of the user attribute.
  - `users-logical-attr-name` is the logical attribute name for the user.
  - `(start-index, length)` syntax is used to extract a fixed-length substring from the value of the logical attribute of the user. The beginning character is `start-index` , and `length` is the number of characters to include starting with `start-index` .
  - `(start-index,)` syntax is used to extract the value of the logical attribute of the user without fixed a length. This syntax can be used to get the substring beginning from start-index until the end of the value.

Getting configured username generation rules and required attributes

Use the Username Generation Helper GET operation to return the username generation rule string that is configured in the internal ID store configuration, along with the attributes that are required to generate a username.

```
GET https://${SSP_FQDN}/{{tenantName}}/admin/v1/UserNameGenerationHelper
```

Sample Response

```
{
   "ruleString": "${idprofile.given_name} + '.' + ${idprofile.family_name}",
```

```
    "requiredAttributes": [
        "given_name",
        "family_name"
    ]
}
```

Validating username generation rules

Use the Username Generation Helper (Validate) POST operation to validate the username generation rule before you configure it in the internal ID store.

```
POST https://${SSP_FQDN}/{{tenantName}}/admin/v1/UserNameGenerationHelper
```

Sample request

```
{
   "operation": "validate",
   "value":  "${idprofile.given_name} + '.' + ${idprofile.family_name}"
}
```

Sample response

```
{
   "status": "User name generation rule is valid"
}
```

Generating usernames

Use the Username Generation Helper (Generate) POST operation to generate the provided username based on the already configured username generation rule.

The attribute details provided as part of `value` must include all required attributes in the username generation rule. For information on viewing the username generation rule and the required attributes for username generation, see Getting configured username generation rules and required attributes.

```
POST https://{{sspHost}}/{{apiPathTenant}}/admin/v1/UserNameGenerationHelper
```

Sample request

```
{
   "operation": "generate",
   "value": {
        "given_name": "Lawrence",
        "family_name": "Haley"
    }
}
```

Sample response

```
{
   "userName": "lawrence.haley"
}
```

Referencing Objects Helper API

Learn about "/ReferencingObjectsHelper" API

The "/**ReferencingObjectsHelper"** API is designed to assist administrators by providing visibility into all dependent Symantec Identity Security Platform objects when attempting to update or delete a resource. The API returns a list of all IDSP objects that are linked to a Certificate or its alias, Secret or its alias, Key or its alias, Application, or Identity Provider for administrator's guidance during updates or deletions.

During update or deletion of a resource by an admin, that has been referenced by other IDSPobjects, this information is also retrieved by the Admin Console App.

Referencing Objects Helper API

Admin Scope	Operation Type	Endpoint URLs
urn:iam:t.configs urn:iam:t.integration	POST	https://${SSP_FQDN}/<tenantName>/admin/v1/ReferencingObjectsHelper

NOTE

- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://${SSP_FQDN}/<tenantName>/admin/v1/ReferencingObjectsHelper`
  should look similar to:
  `https://example.com/default/admin/v1/ReferencingObjectsHelper`
- To know the value of ${SSP_MGMT_FQDN} environment variable, see Configuring the Environment Variables.
- The default value for `<tenantName>` is "default".

Request and Response Samples

- Perform a POST request using the following admin API

```
POST https://${SSP_FQDN}/<tenantName>/admin/v1/ReferencingObjectsHelper
```

API Request

```
{

"idOrName": "c5a2f0a1-29a4-4cd0-848f-c718f7671f3e",

"resType": "urn:iam:res:secret"

}
```

Where,

– **idOrName** - Specify resource id or alias for which referencing objects need to be listed.

– **resType** - Specify resource type. **Possible values** are - urn:iam:res:cert, urn:iam:res:secret, urn:iam:res:key, urn:iam:res:app or urn:iam:res:idp

API Response

The response returns a list of IDSP objects that are using the resource being modified or removed.

```
[
{
"id": "c042e581-5bef-4831-a52a-25ddcf4e4beb",
"name": "SCIM1",
"type": "urn:iam:res:scimconfig"
"allowResourceToBeForcefullyDeletable": "true"
},
```

```
{
"id": "e20bbc85-92a1-43ee-a37b-a48f8a9af047",
"name": "SCIM",
"type": "urn:iam:res:scimconfig"
"allowResourceToBeForcefullyDeletable": "true"
},

{
"id": "fd467d85-c1e3-4acb-ad47-596e529c1a0e",
"name": "customRiskProvider1",
"type": "urn:iam:res:providerconfig"
"allowResourceToBeForcefullyDeletable": "true"
}
]
```

Where,

- "**allowResourceToBeForcefullyDeletable**":
  - If the param is set as true, then forceful delete of the resource is allowed.
  - If the param is set as false, then forceful delete of the resource is not allowed.

Reference Links

- Updating or deleting a Certificate or its alias referenced by other IDSP objects
- Updating or deleting a Secret or its alias referenced by other IDSP objects
- Updating or deleting an Application referenced by other IDSP objects
- Updating or deleting an Identity Provider (IDP) referenced by other IDSP objects
- Resource Deletion in IDSP.

Revoke All App Roles Helper

This helper API performs a DELETE operation to revoke all application roles for a specified user or group from a given application.

About RevokeAllAppRoles Helper API

The **RevokeAllAppRoles Helper API** revokes all application roles for a specified principal (user or group) from a given application.

Admin Scope	Operation Type	Endpoint URLs
urn:iam:t.apps urn:iam:t.users urn:iam:t.groups urn:iam:t.approles urn:iam:t.adminroles	DELETE	https://${SSP_FQDN}/<tenantName>/access/v1/AccessHelper

NOTE

- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://${SSP_FQDN}/<tenantName>/access/v1/AccessHelper`
  should look similar to:

`https://example.com/default/access/v1/AccessHelper`

- To know the value of ${SSP_FQDN} environment variable, see Configuring the Environment Variables.
- The default value for `<tenantName>` is "default".

API Request

```
{
    "appId": "string",
    "id": "string",
    "type": "string",
    "identitySourceId": "string"
}
```

Where, all the following request parameters are required:

- **appId** - Application ID or Application Name
- **id** - User ID or Name or Group ID or Name
- **type** - Type of principal: "user" or "group"
- **identitySourceId** - Identity source identifier

API Response

204 No Content on successful revocation.

Error Responses

- 400 Bad Request - Invalid input parameters
- 401 Unauthorized - Insufficient permissions
- 404 Not Found - Application or principal not found

Access Helper API

The Access Helper API evaluates and returns application access permissions by analyzing both access policies and role assignments.

About Access Helper API

The **Access Helper API** retrieves comprehensive application access information for a user, combining data from both access policies and application roles.

Admin Scope	Operation Type	Endpoint URLs
urn:iam:t.apps urn:iam:t.users urn:iam:t.groups urn:iam:t.approles urn:iam:t.adminroles	POST	https://${SSP_FQDN}/<tenantName>/access/v1/AccessHelper

NOTE

- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://${SSP_FQDN}/<tenantName>/access/v1/AccessHelper`
  should look similar to:

`https://example.com/default/access/v1/AccessHelper`

- To know the value of ${SSP_FQDN} environment variable, see Configuring the Environment Variables.
- The default value for `<tenantName>` is "default".

API Request

```
{
    "tenantId": "string",
    "subject": {
        "id": "string",
        "type": "string",
        "identitySourceId": "string"
    }
}
```

Where, all the following request parameters are required:

- **tenantId** - Tenant identifier for the organization
- **subject** -
  - **id** - User ID or Group ID for access evaluation
  - **type** - Type of subject: "user" or "group"
  - **identitySourceId** - Identity source identifier where the subject exists

API Response

```
[
    {
        "appId": "string",
        "appName": "string",
        "description": "string",
        "isAccessPolicyApplicable": boolean,
        "isAppRoleApplicable": boolean,
        "appRole": {
            "appId": "string",
            "appName": "string",
            "roleId": "string",
            "roleName": "string",
            "roleDescription": "string",
            "displayName": "string",
            "privileges": ["string"],
            "effect": "string",
            "principal": {
                "user": {...},
                "group": {...}
            }
        }
    }
]
```

Resource Counter Helper API

The Resource Counter Helper API allows clients to retrieve a count of resources persisted in the tenant, by resource type.

Admin Scope	Operation Type	
<update>	GET	`https://${SSP_FQDN}/{{tenantName}}/admin/v1/Resou`

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values in all the endpoints shown in the preceding table.
  For example:
  `https://{{host}}/{{tenantName}}/admin/v1/ResourceCounter`
  should look similar to:
  `https://example.com/default/admin/v1/ResourceCounter`
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

The Resource Counter Helper API returns a count of resource in the tenant, by resource type.

```
{
"urn:iam:res:app": 4,
"urn:iam:res:approle": 24,
"urn:iam:res:authnfactor": 0,
"urn:iam:res:authnpolicy": 8,
"urn:iam:res:authzpolicy": 24,
"urn:iam:res:cert": 0,
"urn:iam:res:cred": 7,
"urn:iam:res:group": 83,
"urn:iam:res:idp": 0,
"urn:iam:res:idppolicy": 0,
"urn:iam:res:ldapconfig": 0,
"urn:iam:res:machine": 4,
"urn:iam:res:machine2machinepolicy": 0,
"urn:iam:res:message": 295,
"urn:iam:res:refreshtoken": 0,
"urn:iam:res:resourceattracl": 73,
"urn:iam:res:resourcegroup": 0,
"urn:iam:res:riskpolicy": 0,
"urn:iam:res:scimconfig": 0,
"urn:iam:res:secret": 0,
"urn:iam:res:subdelegation": 0,
"urn:iam:res:user": 6,
"urn:iam:res:usersession": 0
}
```

The Resource Counter Helper API can accept the following optional parameter:

Parameter Name	Parameter Value	Usage
resource	<resource_type>	The resource type for which to return a count. for example, urn:iam:res:approle

Finding the Count of System Tenants

You can use the Resource Counter Helper API to retrieve the count of tenants in the system by calling the Resource Counter API from the system tenant. This API call retrieves a count of all resources in the tenant by type, and retrieves the count of all tenants in the system.

Admin Scope	Operation Type	
<update>	GET	https://{{sspMgmtHost}}/system/admin/v1/ResourceC


```
{
"urn:iam:res:app": 4,
"urn:iam:res:approle": 24,
"urn:iam:res:authnfactor": 0,
"urn:iam:res:authnpolicy": 8,
"urn:iam:res:authzpolicy": 24,
"urn:iam:res:cert": 0,
"urn:iam:res:cred": 7,
"urn:iam:res:group": 83,
"urn:iam:res:idp": 0,
"urn:iam:res:idppolicy": 0,
"urn:iam:res:ldapconfig": 0,
"urn:iam:res:machine": 4,
"urn:iam:res:machine2machinepolicy": 0,
"urn:iam:res:message": 295,
"urn:iam:res:refreshtoken": 0,
"urn:iam:res:resourceattracl": 73,
"urn:iam:res:resourcegroup": 0,
"urn:iam:res:riskpolicy": 0,
"urn:iam:res:scimconfig": 0,
"urn:iam:res:secret": 0,
"urn:iam:res:subdelegation": 0,
"urn:iam:res:user": 6,
"urn:iam:res:usersession": 0
 "urn:iam:res:tenant": 2
}
```

SignIn User Interface (UI)

Learn the process of modifying the branding within the SignIn User Interface (UI).

This topic includes the following sections:

- SignIn UI App - Browser Support
- Branding Customization
  - Steps to update branding settings in SigIn UI
  - Step to update tenant logo and background image in SigIn UI
- RememberMe
  - Steps to enable "Remember Me"

SignIn User Interface (UI) is a software application that allows users to perform multi-factor authentication flows in Symantec Identity Security Platform, such as Password, OTP (includes SMS OTP, EMAIL OTP, and MOBILE OTP), Biometric, Security Key, Passkey, and Push. This is the default UI used by the solution. In addition, when any OIDC standard application such as Symantec SiteMinder, SalesForce, and others integrate with the solution to perform multi-factor authentication flows, the SignIn user interface is used.

SignIn UI App - Browser Support

The following table shows the web browsers supported by the SignIn UI application.

Supported Browser	Supported Browser Versions
Google Chrome	last two versions
Mozilla Firefox	last two versions
Safari	last two versions
Microsoft Edge	last two versions

Branding Customization

As a tenant administrator, you can customize the branding of the SignIn User Interface (UI) to suit the requirements of your organization. For example, you may want to change the logo, background image, or tenant name in the UI. To perform such modifications, you can use the branding APIs.

You can customize the branding of the following SignIn UI parameters:

Parameters	Description
tenantName	Name of the Tenant
tenantBackgroundImage	Background Image of the SignIn UI. This is Base 64 image dat
tenantCustomCSS	Custom Cascading Style Sheet to be used in the SignIn UI.
tenantFavicon	Favicon to be used in SignIn UI. This is Base 64 image data.
tenantCustomHEADContent	This property accepts base64 representation of third-party cod `data:text/html; base64,SGVsbG8gV29ybGQhDQo8c2NyaXB0Pg0KZnVuY3` It allows to embed custom HTML Content in the OOTB SignIn

tenantCustomBodyContent	This property accepts base64 representation of third-party cod `data:text/html; base64,SGVsbG8gV29ybGQhDQo8c2NyaXB0Pg0KZnVuY3` It allows to embed custom HTML Content in the OOTB SignIn
tenantLogo	Logo to be used in SignIn UI. This is Base 64 image data.
fieldsToBeEncryptedInAuthFlow	This property lists the fields that must be encrypted in the auth decide if the password must be encrypted. By default, it is set
Application Properties (Optional)	
appId	Unique identifier of the app
appName	Name of the application
appLogo	Logo of the application. This is Base 64 image data.
Feature Flags	See Managing Settings for more information on these flags.
isRememberMeEnabled	Indicates if the "Remember Me" feature must be enabled by th
riskProviderName	Indicates the Risk provider SDK to be used for the fingerprint c Supported Values: iarisk (default risk) and aarisk.
isCredentialChooserEnabled	By default, it is set to "false".
isTrustThisDeviceEnabled	Indicates whether to enable or disable the 'Trusted Device' fea
localeToBeUsed	Specifies the locale to be used for the SignIn UI display
localeStrategy	Defines the strategy to determine a locale for the SignIn UI dis tenantLocale.
x-mfa-username	When the x-flow-state is passed in the header of the /auth/v1/E of the response. This parameter indicates the username that is allows Signin UI to skip collecting the username.

Steps to update branding settings in SigIn UI

1. Use the following GET call to retrieve all tenant-related branding settings:

   `https://${SSP_FQDN}/<tenantName>/auth/v1/BrandingSettings?clientId=<Client ID>`

   NOTE

   - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
   - The default value for `<tenantName>` is "default".
   - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
     For example:
     `https://{{host}}/{{tenantName}}/auth/v1/BrandingSettings?clientId=<Client ID>`
     should look similar to:
     `https://example.com/default/auth/v1/BrandingSettings?clientId=<Client ID>`
   - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges sections.

2. where Client ID is an optional parameter. This is the Client ID of the application where the login has been performed. If you do not add the <Client ID> parameter to the URL, then the application details do not appear in the response.
   The response is as follows:

```
{
 "tenantName": "default",
 "tenantBackgroundImage": " . . . . . .",
 "tenantCustomCSS": " . . . . . .",
```

```
 "tenantFavicon": " . . . . . .",
 "tenantLogo": " . . . . . .",
 "fieldsToBeEncryptedInAuthFlow": "password",
 "featureFlags": {
     "riskProviderName": "IARISK",
     "isRememberMeEnabled": true,
     "isCredentialChooserEnabled": false,
     "isTrustThisDeviceEnabled": true,
     "isCustomDomainEnabled": false,
     "localeToBeUsed": "en-US",
     "localeStrategy": "tenantLocale"
 },
 "appDetails": {
     "appName": "AdminConsole",
     "appId": "8822a3fe-099a-49fd-9576-6caf07ac9be6"
 },
 "x-mfa-username": "mhunter"
}
```

3. Use the following PATCH call to modify the branding settings:

   ```
   https://${SSP_FQDN}/<tenantName>/admin/v1/Configs
   ```

   This is the request payload to update branding settings for a tenant:

   ```
   [
   {
   "name": "isRememberMeEnabled",
   "value": "true"
   },
   {
   "name": "isCredentialChooserEnabled",
   "value": "false"
   },
   {
   "name": "isTrustedDeviceEnabled",
   "value": "true"
   },
   {
   "name": "maxTrustedDevicesPerUser",
   "value": "10"
   },
   {
   "name": "trustedDeviceExpiryDays",
   "value": "365"
   }
   ]
   ```

   The response retrieves all the updated settings.

Step to update tenant logo and background image in SigIn UI

1. Use the following PATCH call to update the tenant logo and background image:

   ```
   https://${SSP_FQDN}/<tenantName>/admin/v1/Configs
   ```

   The following code snippet shows a sample request to update the tenant logo and background image. The image data is in the base64 format:

   ```
   [
   ```

```
  {
      "name": "tenantLogo",
      "value": ".........K5CYII="
  },
  {
      "name": "tenantBackgroundImage",
      "value": "........73A/q/df/9k="
  }
 ]
```

Remember Me

As a tenant administrator, you can enable the "Remember Me" feature in the SignIn UI, Sample App, and Mobile (Hedmoral -Android & iOS) Client apps. When this feature is enabled, a "Remember Me" checkbox is displayed in the UI.

As a user, when you log into any of the supported UI for the first time, you are prompted to enter the “Username”, and a “Remember Me” checkbox is displayed in the UI. After entering your username, if you select the "Remember Me" checkbox, your username gets stored as a cookie named, "SSP-USERNAME" in the browser. The password prompt is displayed thereafter.

For all your subsequent logins, this cookie is used to automatically authenticate you and log you in without having you enter your username.

NOTE

- For better usability, the application stores the authentication factor with which you logged into the UI for the first time and even the most commonly used authentication factor, so that when you log in the next time, you are immediately prompted for that same authentication factor, and you are not asked to choose the factor every time. If you want to change the factor, however, you can click on "**Choose another choice**" in the UI and pick a different factor.
- You can change your username by clicking the "Not <Username>" link. This link only appears when IDSP manages both primary and secondary authentication.
- If you clear the cookies from the browser that you have been using to authenticate, then the "Remember Me" functionality gets invalid. Therefore, you must select the "Remember Me" checkbox once again to activate it.

Steps to enable "Remember Me"

To enable "Remember Me", you have to update configurations in the Tenant Config and update the Initiate Authenticate Call.

Updating Tenant Configuration

Follow these steps:

1. Use the following PATCH call to enable the "isRememberMeEnabled" parameter in the tenant settings:

   ```
   https://${SSP_FQDN}/<tenantName>/admin/v1/Configs
   ```

   The following is the PATCH request:

   ```
   [
   {
    "name": "isRememberMeEnabled",
    "value": "true"
   }
   ]
   ```

   The parameter gets updated in the Tenant settings. To view the Tenant configurations, click here.

Updating the Initiate Authenticate Call

Follow this step:

1. Use the following POST call to enable the "rememberMe" parameter in the request payload:
   ```
   https://${SSP_FQDN}/<tenantName>/auth/v1/authenticate
   ```
   Update the parameter as follows:
   ```
   "rememberMe": "true"
   ```

To know more about Initiate Authenticate call, click here.

Reference Links to:

- Configuring Trusted Devices.
- Swagger Documentation for Trusted Devices and Branded Settings.
- Localization for the SignIn UI

Integrating Identity Provider with Custom SignIn UI

Enables your custom SignIn UI application to integrate with the Identity Provider service provided by Symantec Identity Security Platform.

The steps in this topic guide you to enable your custom SignIn UI application to integrate with the Identity Provider service provided by IDSP.

Prerequisite

- You should have an Identity Provider created in the IDSP.

  NOTE
  See Identity Provider for information on how to create an Identity Provider in the solution.
- Create an Authentication Policy for your application and add the Identity Provider that you have created as part of the prerequisite in the policy obligations. For example, the authentication policy should have your identity provider added as shown in the following example:
  ```
  "obligations": [
      "PASSWORD:1",
      "IDP/0747a4dd-84ba-49c0-accc-89a75e5eb6c3:2"
  ],
  ```
  Upon successfully adding the IDP to the policy obligation, the IDP gets added to the list of the factors in the UI.

Steps to be Performed in the Custom SignIn UI

1. Upon triggering the ".../auth/v1/SelectedFactor API, the response of this API contains the name of the IDP that can be used for display in the custom SignIn UI. The response includes the IDP details in the "***additional***" section, under the "***predicatesForCurrentFactors***" section, as shown in the following example:
   ```
   {
       "flowState": "bDJpVm5BQW........K1BDQT09",
       "nextaction": "FACTOR_SELECTION",
       "currentFactors": [
           "SMSOTP",
           "IDP/c856ffcc-d700-4b9e-94d2-da41fddbda50"
       ],
       "additional": {
           "currentFactorLevel": 2,
           "predicatesForCurrentFactors": [
               {
                   "factor": "IDP/c856ffcc-d700-4b9e-94d2-da41fddbda50",
                   "predicate": "Azure-OID
   ```

```
            }
        ],
        "amrForCompletedFactors": [
            "PWD"
        ],
        "defaultFactor": "SMSOTP"
    }
}
```

2. In the custom SignIn UI, click the IDP that internally invokes the following "SelectedFactor" API:

   `HTTP POST: https://${SSP_FQDN}/tenant/auth/v1/SelectedFactor`

   NOTE

   - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
   - The default value for `<tenantName>` is "default".
   - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
     For example:
     `https://{{host}}/{{tenantName}}/auth/v1/SelectedFactor`
     should look similar to:
     `https://example.com/default/auth/v1/SelectedFactor`
   - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

   As per the IDP example used in the prerequisite (2nd bullet), the payload would look like the following:

   `{"factor": "IDP/0747a4dd-84ba-49c0-accc-89a75e5eb6c3"}`

   The "SelectedFactor" API responds with the "nextaction" as "IDP_AUTH and the "flowState".

3. Invoke the following API, pass the "flowState" as a request header "x-flow-state" and add another request header "idp-relay-state" with the custom SignIn UI client's URL:

   `https://${SSP_FQDN}/tenant/oauth2/v1/idp`

   NOTE
   For example, the Custom SignIn UI client URL would look like this-
   `https://<your-custom-signin-client-url>`

   The API responds with either of the following:

   – an “X-samlUrl” response header
   – HTML markup containing a form.

   This depends on the configuration of the IDP, whether Redirect or HTTP POST. If the API responds with an ‘X-samlUrl’ response header, then the UI can directly navigate to that URL. In the absence of ‘X-samlUrl’, there will be markup in the API response containing an HTML form. This form must be programmatically submitted through JavaScript by appending it to the DOM and invoking the submit function over the form.
   Both these actions take the user to the external Identity Provider's website for further authentication. Once successfully authenticated, the user is redirected to the IDSP for further processing.

Steps to be performed in Custom SignIn UI (After redirection from external Identity Provider)

The following steps should be performed after the user is successfully authenticated in the external IDP's website and redirected to IDSP.

1. Create a file named, "idp.html" that is hosted at the "idp-relay-state" path that was passed as a request header in step 3.
   For example, if the "idp-relay-state" path is in the following location- **`https://<your-custom-signin-client-url>`** then the "idp.html" file should be available here - **`https://<your-custom-signin-client-url>/idp.html`**

The following should be the contents of the "idp.html" file:

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
</head>
<body>
    <script type="text/javascript">
        if (window.parent !== window) {
            window.parent.postMessage("RECEIVER_READY", "<SSP Host or Vanity Host>");
            window.addEventListener("message", function (e) {
                if (typeof e.data === "string") {
                    try {
                        var tmp = JSON.parse(e.data);
                        if ("flowState" in tmp) {
                            sessionStorage.setItem("idpData", e.data);
                            window.parent.postMessage("RECEIVED_DATA", "<SSP Host or Vanity Host>");
                        }
                    } catch (_ex) {
                        /* No Action */
                    }
                }
            }, false);
        }
    </script>
</body>
</html>
```

Once the user returns from the external Identity Provider's website, the above "idp.html" file will be invoked and the necessary response data are captured available in the sessionStorage as "idpData". IDSP will then redirect to the Custom Signin UI's URL, that is, the URL that was sent as part of the "idp-relay-state" request header in step 3.

2. The UI client can consume the "idpData" and do either of the following:
   – Redirect the user to the application if the "idpData" contains "nextaction"= "AUTH_ALLOWED", and includes the "authCompleteUrl" property.
   – If the "nextaction" property displays other factors for authentication, then the custom SignIn UI can display the factor authentication screens for further processing.

Use a Custom SignIn UI Application

Replace the default SignIn UI with a custom application built using Symantec Identity Security Platform APIs to provide tailored authentication experiences and branding.

This topic Includes the following sections:

- Configuration to Support a Custom SignIn UI Application
- Invoking the ".../authenticate" call without a subject
- Responsibilities of a Custom SignIn UI Application
- Locale Management APIs

Configuration to Support a Custom SignIn UI Application

To use a custom Signin UI application, configure the **flowURL** property in the required application to define the redirect URL to the custom SignIn UI application.

Example: "flowURL": "https://HOST/<*custom-signin-url*>?param1=xxx&param2=yyy&..."

If the **flowURL** property is configured, IDSP adds the following query parameters to the **flowURL** property for redirection to the defined custom SignIn UI URL:

- x-flow-state
- X-CLIENT-TRANSACTION-ID
- x-service-baseurl
- x-mfa-username-obfs
- ui_locales
- failureCallbackURLBase64

Example:

```
?x-flow-
state=RkFBUFdWZ2NzOHA2TlNncHo3aXVIMjNhQnlxWi9ac1JmZm1tcllwQlVwZStMU1JrL1pGNkV0cnVkOHNpOExTRzJFQVpNamtSQUZtai9hdkY5Uk5zM
service-baseurl=https:///&X-CLIENT-TRANSACTION-ID=abc12345678&x-mfa-username-
obfs=xxxx&ui_locales=xxx&failureCallbackURLBase64=aHR0cHM6Ly9rcmlzaG5hLTE4ODgtc3NwLnNzcGRldi5kZXYuYnJvYWRjb20uY29tL2RlZ
```

Invoking the ".../authenticate" call without a subject

> **NOTE**
> When developing a custom sign-in application, the initial ".../authenticate" call without a subject (username) is typically needed when the sign-in flow is initiated by a protocol-driven app—such as a SAML app, OpenID/OAuth app, or a URL launcher app. However, for standalone native sign-in flows (mobile flows), such as those on iOS or Android apps, where the login process starts by directly prompting for the username, the initial ".../authenticate" call without a subject is not required.
> For more information, see the information on initiating the ".../authenticate" call without a subject in this bullet- "Full/Partial Delegation".

Responsibilities of a Custom SignIn UI Application

A custom SignIn UI application must be developed to perform the following steps:

- Save the following query parameters that are added to flowURL and that are received through a query string, until the SignIn UI flow completes:
  - x-flow-state
  - X-CLIENT-TRANSACTION-ID
  - x-service-baseurl
  - x-mfa-username-obfs
  - ui_locales
  - failureCallbackURLBase64
- Use the saved query parameters in each API that is invoked during the authentication flow.
  - **x-service-baseurl**: Use the value of the x-service-base URL parameter as the base URL for making IDSP API calls.
  - **x-flow-state**: In the first call, send the value of the x-flow-state parameter as the request header. In the subsequent calls, send the x-flow-state parameter as the request header with the value that is populated from the previous response.
  - **X-CLIENT-TRANSACTION-ID**: Send the value of the X-CLIENT-TRANSACTION-ID parameter as the request header to ensure that the same client transaction ID is used in the operational log. This value might be empty. Alternatively, the custom SignIn UI application can choose to indicate its own transaction ID during the authentication flow.
  - **x-mfa-username-obfs**: Use this parameter when the primary factor authentication is done outside IDSP. For example, if the primary factor authentication is done in Siteminder, the obfuscated username is passed through this

query parameter to the Signin UI so that the trustedDeviceID of the user, and preferred factors can be picked from the cookies, etc.

- **ui_locales**: Siteminder uses this parameter to send a preferred locale through the "/authorize" endpoint to the Signin UI. Signin UI picks up the ui_locales in the following sequence:
  a. query parameter if exists
  b. If the query parameter does not exist, check the value that is configured in "localeStrategy" under tenant settings.
     If localeStrategy == tenantLocale, then the preferred locale is the locale that is configured in the tenant setting, "localeToBeUsed".
     If localeStrategy == browser, then the preferred locale is the browser locale.

     NOTE
     IDSP supports five locales- en-US, fr-FR, ja-JP, pt-BR, es-ES. These locales are maintained by IDSP during the upgrade. To use locales other than the supported ones, you should update them manually.
- **failureCallbackURLBase64**: If there is any failure in the custom SignIn UI flow, use this URL for redirection to complete the authentication flow. The failure call back URL will proxy these parameters, &X-CLIENT-TRANSACTION-ID=<id> and &X-TRANSACTION-ID=<id> to the RP's call back URL.

- Begin by sending the first authentication request to: `<x-service-baseurl>/auth/v1/authenticate` .
  - **In a Full Delegation flow** when the "/...authenticate" API is called without including the subject (username) in the request payload, the response returns "nextaction": null. In this case, the UI should prompt to enter username and make a second call to the "/authenticate" API with the subject in the request.
  - **In a Partial Delegation flow** when the ".../authenticate" API is called without the subject, the response includes a valid "nextaction" that specifies the next required factor (for example, OTP, push, etc.). The UI logic should be designed to implement this "nextaction" accordingly to guide the user through the authentication flow.

  For information about using this the "/authenticate" API, see Understanding the User Flow.
- If the API response contains authCompleteUrl, redirect to authCompleteUrl to complete the authentication flow.
  Example: authCompleteUrl

  ```
  <x-service-base>/oauth2/v1/authcomplete?x-flow-
  state=RkFBUFdWZ2NzOHA2TlNncHo3aXVIMjNhQnlxWi9ac1JmZm1tcllwQlVwZStMU1JrL1pGNkV0cnVkOHNpOExTRzJFQVpNamtSQUZtai9hdkY5Uk
  service-baseurl=https://${SSP_FQDN}/<tenant-name>&X-CLIENT-TRANSACTION-ID=abc1234567
  ```
- Set the "***X-Reset-Flow"*** ***request header*** in the ".../auth/v1/authenticate" API to implement the "Not User?" functionality.
  For example: When you click "Not Kevin Lucas?" on the password page that is displayed after you have initially signed in to the SignIn UI as Kevin Lucas, a new authenticate call will be triggered. To remove the username, Kevin Lucas (subject) from the flow-state and add the new user to the authentication flow, you must include the request header "X-Reset-Flow=true" in this authenticate call. See Authenticate API

Locale Management APIs

The following table displays the list of APIs that can be used by an administrator to configure the locales in the solution:

Operation Description	Operation Type	Endpoint URLs
Create localized content for a locale.	POST	https://${SSP_MGMT_FQDN}/system/admin/v1/LocaleContent
Update the localized content for a specific custom locale.	PUT	https://${SSP_MGMT_FQDN}/system/admin/v1/LocaleContent/{{locale-name}}
Update localized content for the OOTB supported locales.	PATCH	https://${SSP_MGMT_FQDN}/system/admin/v1/LocaleContent/{{locale-name}}
Delete localized content for a given locale.	DELETE	https://${SSP_MGMT_FQDN}/system/admin/v1/LocaleContent/{{locale-name}}

Get localized content of a locale.	GET	https://${SSP_MGMT_FQDN}/system/admin/v1/LocaleContent/{{locale-name}}
Get the list of locales that have localized content in the IDSP.	GET	https://${SSP_MGMT_FQDN}/system/admin/v1/LocaleContent/
Get Locale content (Invoked by SignIn UI).	POST	https://${SSP_FQDN}/<tenantName>/auth/v1/UILocaleContent/

NOTE

- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
  For example:
  `https://${SSP_MGMT_FQDN}/<systemtenantName>/admin/v1//LocaleContent/{{locale-name}}`
  should look similar to:
  `https://mgmt-example.com/system/admin/v1//LocaleContent/{{locale-name}}`
- To know the value of ${SSP_MGMT_FQDN} and ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<systemTenantName>` is "system".
- To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.

Reference Links:

- To know more about Localizing the IDSP SignIn User Interface using the IDSP APIs, see Localizing the Signin UI Application.

Custom Header and Footer

Learn how to update the header and footer content in the OOTB Sign In UI application.

You can customize the content, structure, and styling of the header and footer according to your specific requirements and design preferences in the Sign In UI application. The sections that are mentioned in this topic are applicable only for the OOTB Sign In application, and not applicable for custom Sign In application.

NOTE
Customization of Header and Footer are exclusively achievable through the POST ".../admin/v1/LocaleContent" API and cannot be performed using the Admin Console.

The ***Header*** and ***Advanced Footer*** customization supports the use of HTML tags for HTML rendering.

The following HTML tags are ***not supported*** in the customization of header and footer content:

- <script>
- <applet>
- <object>
- <embed>

This topic includes the following sections:

- Invoking the ".../admin/v1/LocaleContent" API to update custom header and footer
- Updating a Custom Header
- Updating a Custom Footer

Invoking the ".../admin/v1/LocaleContent" API to update custom header and footer

To update a header or footer in the Sign In application, use the following Symantec Identity Security Platform API endpoint:

```
POST https://{{sspMgmtHost}}/system/admin/v1/LocaleContent
```

To know more about the request payload of the ".../admin/v1/LocaleContent" API, see Managing localization through IDSP APIs.

Updating the Custom Header in the Sign In application

To update the header in the Sign In application, add your HTML Header code inside the following attribute in the ".../admin/v1/LocaleContent" API request:

```
"headerContent":""
```

- The <headerContent> can include elements such as website title, icons, navigation links, and the current year can also be dynamically integrated into the header. This is explained in the Advanced Footer section.

Updating Custom Footer in the Sign In application

IDSP supports two types of custom footer, Basic Footer and Advanced Footer.

Basic Footer:

- In this scenario, the footer content is displayed in a single line.
- The footer content can be inserted into the "***footerContent***" key-value pair which is an attribute available within the ".../admin/v1/LocaleContent" API. Footer content typically includes elements such as privacy policy, terms and conditions, and others.

Advanced Footer:

- Unlike the Basic Footer, the Advanced Footer allows for content to be rendered across multiple lines.
- This footer variant permits customized content, such as paragraphs, multiple links, icons, etc.
- To implement an advanced footer, enclose the complete footer markup within a tag and insert it into the "**footerContent**" key-value pair attribute available within the ".../admin/v1/LocaleContent" API.
  The following example shows the div tag within which the footer markup must be enclosed:
  ```
  <div class="custom-mega-footer">
  add the footer markup here
  </div>
  ```
- Current Year can be dynamically integrated into the ***header and footer***.
  – To update the current year dynamically into the header and footer, you can use a substituter in the HTML markup which gets automatically substituted with the correct value at runtime.
  Add the current year value in this format:
  ```
  {CURRENT_FULL_YEAR}
  ```

Editing CSS to Modify Elements in the Sign In UI application

Learn how to edit the Sign In UI application elements in a web page using the Cascading Style Sheet (CSS).

This topic includes the following sections:

- Inspecting a page to locate CSS elements for modification
- Using the "!important" rule for CSS overrides
- Limiting CSS Scope with [page=...] Attribute
- Examples

  - Modifying the text color of the "Sign In" term
    - Use of the "!important" rule
  - Modifying the shape of the "Sign In" button
  - Hiding an element in the Sign In application
  - Customizing Sign In UI for Specific Applications
- Save and Apply the CSS updates

Inspecting a page to locate CSS elements for modification

To inspect a page for CSS modification:

1. Log in to the application.
2. Select and right-click on the element you want to modify in the webpage, and click "Inspect".
   The browser's Developer Tools opens, and displays the HTML markup and CSS styles for the selected element.
3. Locate the CSS styles associated with the element and make the desired modifications.
4. Test the changes to ensure that the changes have been applied correctly.

Refer to the section titled, Modifying the text color of the "Sign In" term for instructions on inspecting an element and adjusting its CSS style.

Using the "!important" rule for CSS overrides

The "***!important***" rule in CSS is used to give a particular style declaration higher priority over others. When applied to a CSS property, "!important" ensures that the specified style overrides any other conflicting styles, even if they are more specific or come later in the CSS cascade. You can apply this rule when changes to the CSS do not reflect in the application due to existing CSS configurations.

Refer to the third step in the section titled, Modifying the text color of the "Sign In" term for instructions on using the "! Important" rule.

Limiting CSS Scope with [page=...] Attribute

Using the [page=...] attribute allows CSS rules to be applied selectively to specific pages within the sign-in UI, ensuring that the modifications only affect elements on the designated page rather than across the entire sign-in interface. This approach enables customization while maintaining consistency and avoiding unintended changes to other parts of the UI.

Refer to the section titled, Modifying the shape of the "Sign In" button for instructions on page-specific updates.

Examples

Modifying the text color of the "Sign In" term

The following example shows how to inspect a page to locate CSS elements for modification.

To change the text color of the "Sign In" term from black to red, follow these steps:

1. Open the Sign In application.
2. Navigate to the Sign In application header, then right-click and select **Inspect**.
   The **Elements** tab opens, highlighting the header element as shown in the following illustration:

```
<header _ngcontent-hhc-c52 class="title">Sign In</header> == $0
```

3. Create a CSS file if you have not already done, and add the "***!important***" css rule, to change the text color of the "Sign In" term from black to red, as shown in the following illustration:

```
header.title {
    color: red !important;
}
```

Modifying the shape of the "Sign In" button

The following example shows how to inspect a specific page to locate CSS elements for modification, avoiding unintended changes to other pages in the UI.

To modify the shape of the "Sign In" button from oval to rectangle in the "Enter your password" page, follow these steps:

1. Open the "Enter your Password" page where the "Sign In" button is located.
2. Right-click on the "Sign In" button and select **Inspect**.
   The **Elements** tab opens, highlighting the button element as shown in the following illustration:

   ```
   <button _ngcontent-hhc-c57 type="submit" class="primary-button"
   disabled>Sign In</button> == $0
   ```

   The Elements tab also displays the page name, which is crucial to make changes to that specific page. Take note of the page name displayed as shown in the following illustration:

   ```
   <html lang="en" page="/default/ui/v1/signin/verify/">
   ```

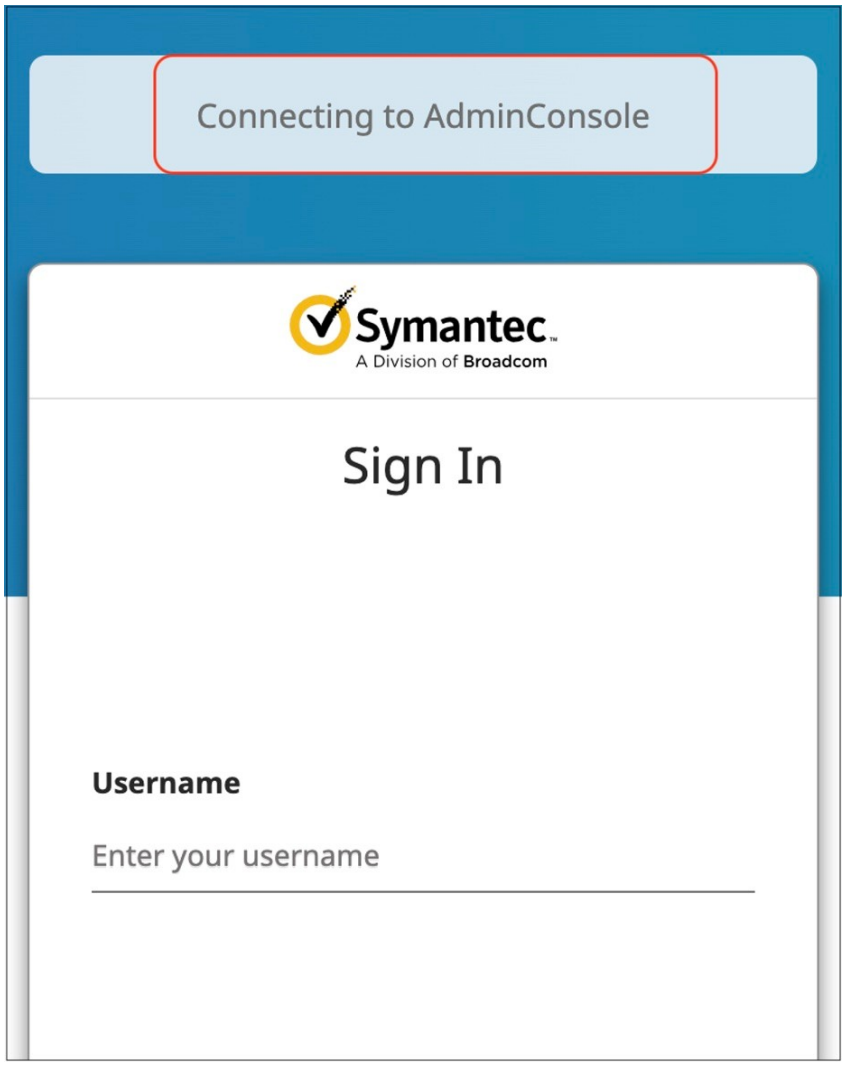
3. Create a CSS file if you have not already done, and add the following css rule to change the shape of the "Sign In" button from oval to rectangle:

   ```
   [page*="/signin/verify/"] {
       button.primary-button {
           border-radius: unset;
       }
   }
   ```

   Where,
   - **page*** - Indicates a 'contains' selector for matching part of the page name.
   - **button** - This is the button whose shape has to change from oval to rectangle
   - **border-radius:unset** - This rule unsets the oval border and changes the button to rectangular shape.

Hiding an element in the "Sign In" application

Assuming that you want to hide the "Connecting to Admin Console" Box from the SignIn application page as highlighted in the following image:



Follow these steps:

1. On the Sign In applications Username page, select the “Connecting to Admin Console” box, and click **Inspect**. The **Elements** tab opens, highlighting the button element as shown in the following illustration:

```
▶ <div _ngcontent-wma-c62 id="js-appInfo" class="appInfo">…</div> == $0
  <!---->
```

2. Create a CSS file if you have not already done, and add the following css rule to hide the “Connecting to Admin Console” box:

```
div.appInfo {
    display: none;
}
```

   Where,
   – **div.appInfo=display: none** - This is the css rule that should be applied to hide the “Connecting to Admin Console” box from the css configuration.

Customizing "Sign In" UI for Specific Applications

Assuming you need to modify the font color on the Username page within the Sign In application, such as changing it to red, when the Sign-In interface is rendered for a particular application, follow these steps:

Steps to customize font color based on application:

1. On the Sign In applications Username page, click **Inspect**.
2. There are two options to retrieve the App ID.
   - Locate and copy the value of the "**app-id**" attribute from the HTML element. This ID uniquely identifies the application for which you want to apply the customization. For example: "app-id"="93af69e4-0675-4d82-af51-0a8a6535a906" as shown in the following illustration:

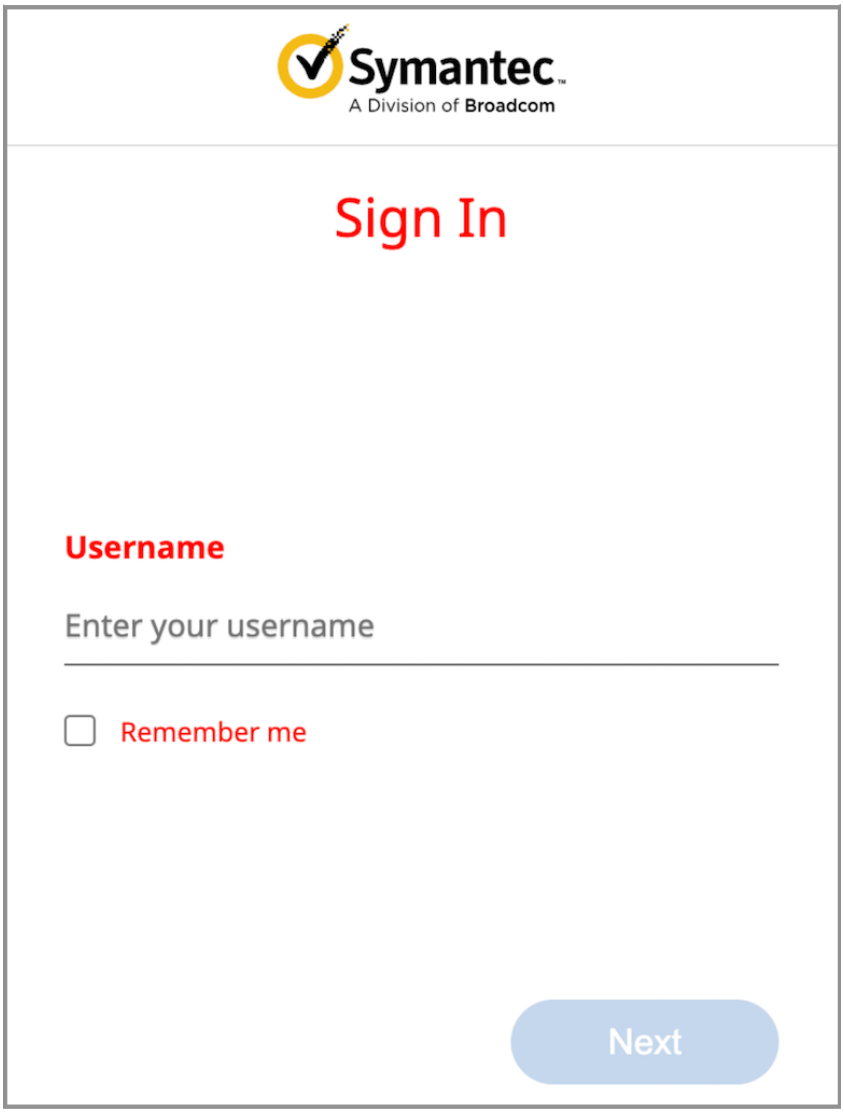
     ```
     <html lang="en" data-critters-container page="/default/ui/v1/signin/" a
     ```

   - Alternatively, retrieve the app-id from the Admin Console (This option is only applicable for tenant level applications):
     Navigate to Admin Console > Applications > [Select Application], then click the "Copy to Clipboard" icon next to the App ID.
     For example: "app-id"="0c0ee7d1-fcfd-4118-950a-82db1defa27e" as shown in the following illustration:

3. Create a CSS file if you have not already done, and add the following css rule to modify the font of the Username page for the specific app:

   ```
   [app-id="93af69e4-0675-4d82-af51-0a8a6535a906"] body{
       color: #f00;
   }
   ```

   Where,
   - **app-id="93af69e4-0675-4d82-af51-0a8a6535a906"** - This is the specific application ID used to target the Sign In UI modifications.
   - **color: #f00;**: This sets the text color to red. The modification is applied to the Sign In UI as shown in the following illustration:



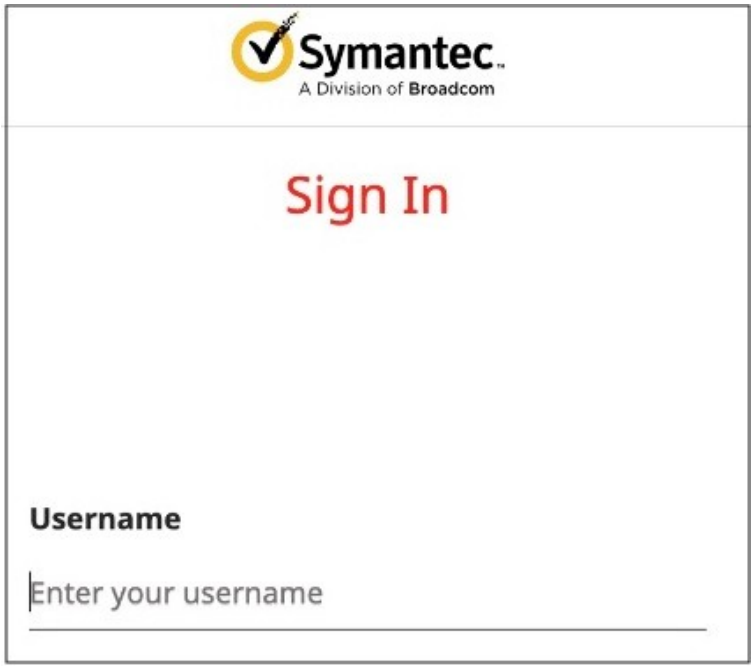
Save and Apply the CSS Updates

After making all the necessary changes in the css file, save the file with a name, for instance- custom.css.This is how the file content looks like after adding few examples that have been cited in the previous sections:

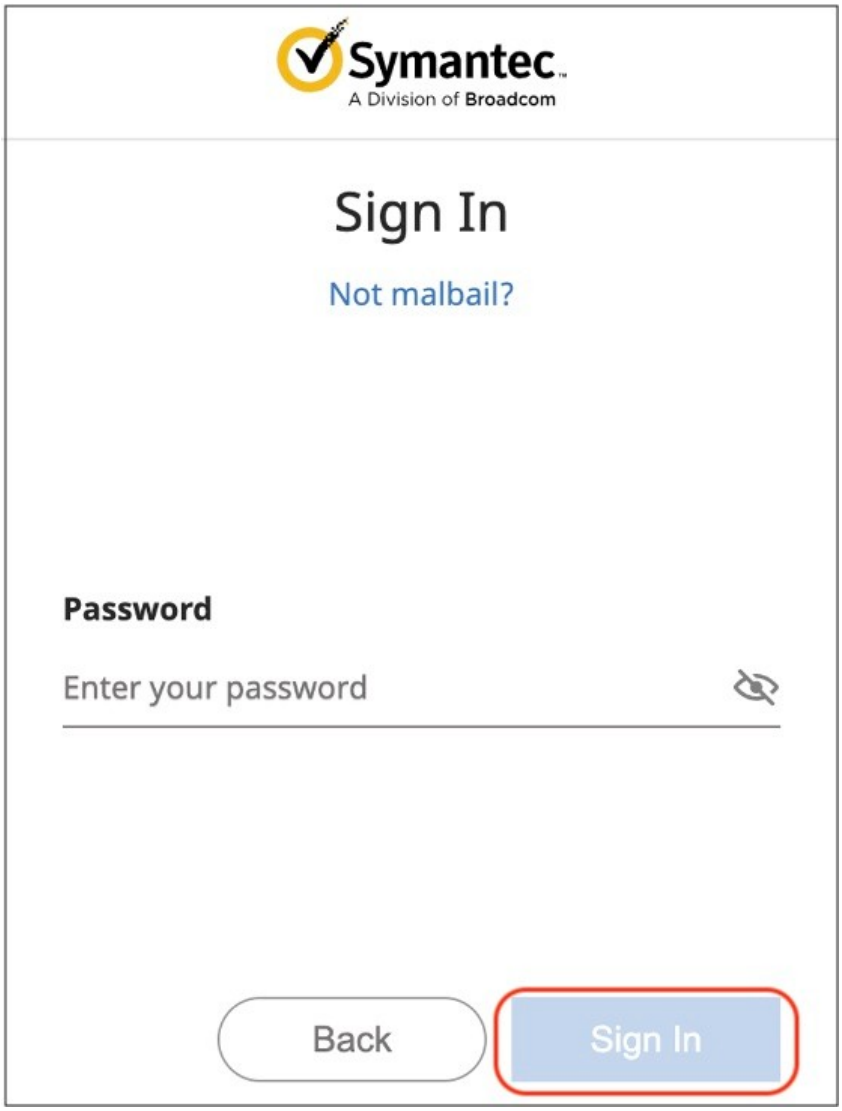
```
header.title {
    color: red !important;
}

[page*="/signin/verify/"] {
    button.primary-button {
        border-radius: unset;
    }
}

div.appInfo {
    display: none;
}
```

To apply these changes, perform these steps in the Admin Console:

1. Navigate to the Admin Console's **Settings** > **Branding** > **Custom Cascading Style Sheet (CSS)**
2. Select the **Use Style from an override css file** option, and upload the saved "custom.css" file.
3. Sign-out of the Admin Console application, and Sign-In again.
   You are displayed with the modified application:
   – The Header color has changed to red in the username page, and the "Connecting to Admin Console" box has disappeared from the Sign In application page:



   – The "Sign In" button in the "Enter your password" page has changed to a rectangle:



Supporting Legacy Browsers

Learn how you can use legacy browsers in the Symantec Identity Security Platform SignIn UI.

When enabled, the Sign-In UI supports `IE 11`, `EDGE 18.x`, and `modern browsers`. If disabled, only modern browsers are supported.

To enable legacy browser support, add the following setting during the IDSP Helm deployment:

`ssp.featureFlags.signinSupportsLegacyBrowser.enabled = true`.
By default this setting is configured as false.

Browser capabilities supported in Legacy Signin UI

The following table outlines the browsers and their capabilities supported in the Legacy Sign-In UI:

Features Supported	IE 11	EDGE 18.x (Embedded web view 3.0 inside Office 365 and Teams)	Modern Browsers
General Rendering	Yes	Yes	Yes
JavaScript Support	Yes	Yes	Yes
FIDO	No	Yes	Yes
IDP Flows	No	No	Yes

Support Username and Password On Initial Login Screen

Learn how the Symantec Identity Security Platform SignIn UI collects both username and password from the initial login screen and processes authentication based on AuthN policy (with password as the first factor) associated with the selected application.

The Sign-in UI is designed to collect both the username and password on the initial login screen and authenticate users based on the authentication (AuthN) policy associated with a specific application. The authentication flow relies on the password as the first factor, ensuring compliance with the defined AuthN policy.

The steps to configure this feature in the SignIn UI application have been described in the following sections.

Assumption

- To enable this feature, ensure that "Password" is configured as the **first and the only factor** for authentication in your authentication (AUTHN) policy for the specified application.

Application Metadata Configuration

The following application metadata must be set to "true":

```
{
     "name": "urn:iam:app:supportUsernameAndPasswordOnInitialScreen",
     "value": "true"
}
```

Where,

Application Metadata Value	Behavior in the SignIn UI
`"urn:iam:app:supportUsernameAndPasswordOnInitialScreen": true`	Enables display of both username and password on the same screen.
`"urn:iam:app:supportUsernameAndPasswordOnInitialScreen": false`	Enables display of username and password on separate screens.

Branding Settings Behavior

- **Based on the application metadata (urn:iam:app:supportUsernameAndPasswordOnInitialScreen)** configuration, the Branding Setting API response returns the following setting as true or false :

  ```
  "supportUsernameAndPasswordOnInitialScreen": true/false
  ```

Forgot Password URL Configuration

The Branding Setting API response returns the following setting as true or false, depending on whether the "**passwordResetURL**" URL is configured in the ID Store (property) associated with the application.

```
"forgotPasswordUrlExists": true/false
```

To determine whether the forgot password URL is configured, the Sign-in UI checks the ID Store defined in IDSP in this order -

1. Application property: "**idStoreToUse**"
2. Tenant configuration: "**idStoresToBeUsed**" (used if the application property is not defined)

Behavior of this setting, "forgotPasswordUrlExists":

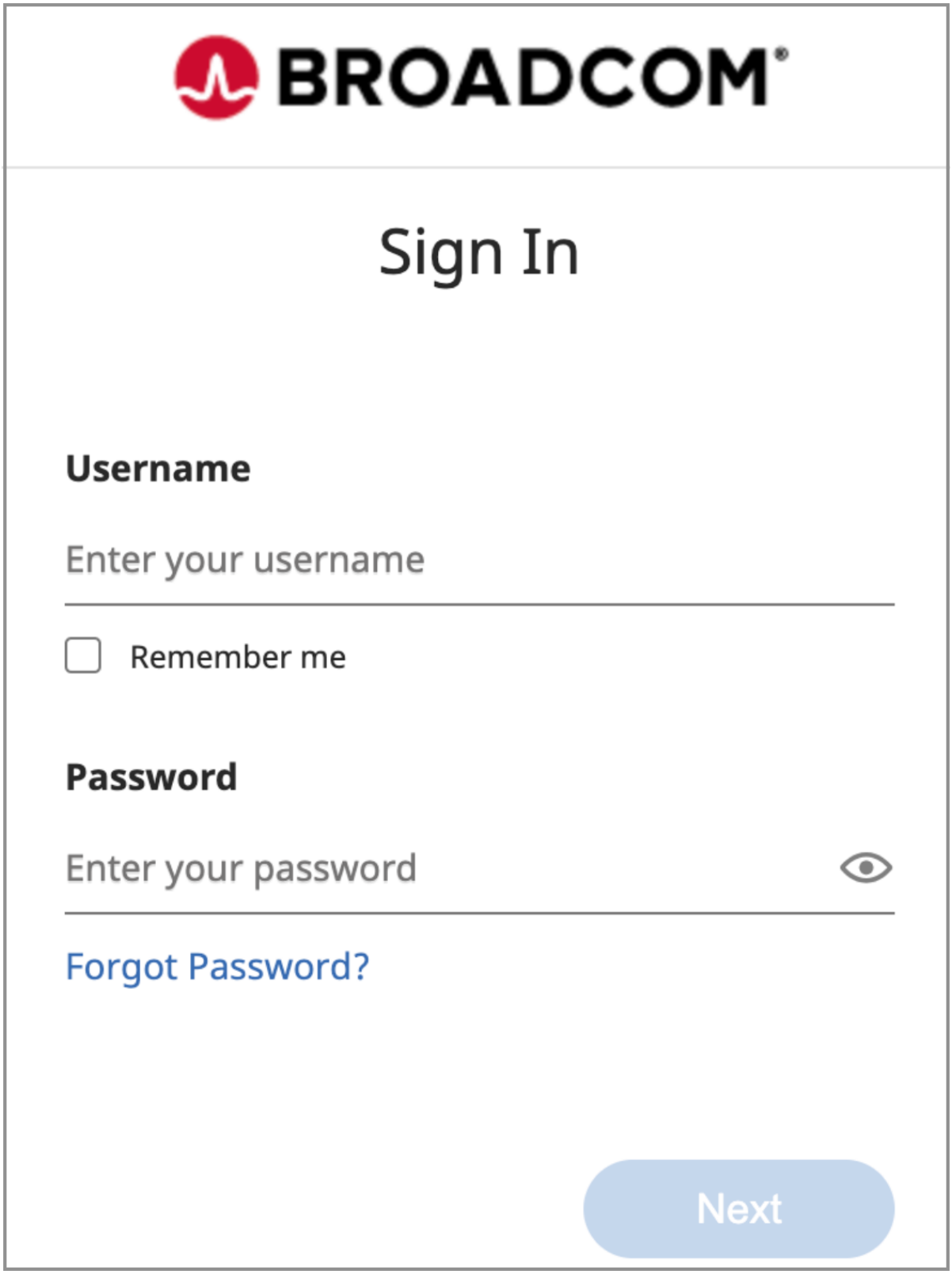
Based on the ID Store configuration, the behavior changes as the following:

ID Store Configuration	Branding Setting (true/fasle)	Authentication Flow Outcome
"passwordResetURL" value is provided	"supportUsernameAndPasswordOnInitialScreen"	The "Forgot Password" link is displayed on the username/password screen in the SignIn UI.
"passwordResetURL" value is not provided	"supportUsernameAndPasswordOnInitialScreen"	The "Forgot Password" link is not shown on the username/password screen in the SignIn UI.

Username and Password representation in the SignIn UI

The following illustration is an example of Username and Password displayed on the same screen, along with the "Forgot Password" link.

Figure 33: Username and Password displayed on the same screen



Enable Custom Content Embeds in SignIn UI

Allows administrators to embed custom HTML content in the out-of-the-box SignIn UI.

In Symantec Identity Security Platform, you can configure custom HTML content, providing greater flexibility for tasks like analytics, tagging, or injecting custom scripts into the Sign-In UI.

The steps to configure this feature in the SignIn UI application have been described in the following sections.

Configure Tenant / Branding Settings in IDSP

- **If you are using the IDSP API** to configure the required tenant setting as discussed in the following table, use the `https://${SSP_FQDN}/{{tenantName}}/admin/v1/Configs` API to update these settings. This allows to enable embedding custom HTML content in the OOTB SignIn UI page's <head>, <body> tag:
- **If you are using the Admin Console UI** to enable embedding custom HTML content in the OOTB SignIn UI page's <head>, <body> tag, see Branding settings.

> **NOTE**
> When configuring tenant settings through the IDSP API flow, use the Base64 DataURI format. However, if you are using the Admin Console UI, provide the value in plain text, as the UI automatically converts it to Base64 DataURI format for further processing.

Table 118: Tenant Settings

Tenant Setting	Description
tenantCustomHEADContent	This property accepts base64 representation of third-party code/script in the following format: `data:text/html; base64,PCFET0NUWVBFIGh0bWw+CjxodG1sPgoKPGhlYWQ+Cgo8L2hlYWQ+Cgo8Ym9keT4KCjwvYm9keT4KCjwvaHRtbD4=` It allows to embed custom HTML Content in the OOTB SignIn UI page's <head> tag. You can provide any valid HTML string such as JavaScript, HTML, CSS. **Default Behavior**: This field is empty by default.
tenantCustomBODYContent	This property accepts base64 representation of third-party code/script in the following format: `data:text/html; base64,PCFET0NUWVBFIGh0bWw+CjxodG1sPgoKPGhlYWQ+Cgo8L2hlYWQ+Cgo8Ym9keT4KCjwvYm9keT4KCjwvaHRtbD4=` It allows to embed custom HTML Content in the OOTB SignIn UI page's <body> tag. You can provide any valid HTML string such as JavaScript, HTML, CSS. **Default Behavior**: This field is empty by default.

These tenant settings are also covered in the Managing Settings page.

Branding Settings Behavior

- **Based on the tenant settings (in API flow) or branding settings configuration (in Admin Console flow)** , the Branding Setting API response returns the following settings with it's corresponding values:


```
{
   "tenantName": "default",
   "tenantBackgroundImage": "....jwvc3ZnPg==",
   "tenantLogo": "....SUVORK5CYII=",
   "tenantFavicon": "....SUVORK5CYII=",
```

```
    "tenantCustomCSS": "",
    "tenantCustomHEADContent": "Base64 HTML Code",
    "tenantCustomBODYContent": "Base64 HTML Code",
    "fieldsToBeEncryptedInAuthFlow": "password,devicesignature",
    "featureFlags": {
        "riskProviderName": "iarisk",
        "isRememberMeEnabled": true,
        "isCredentialChooserEnabled": false,
        "isTrustThisDeviceEnabled": true,
        "isCustomDomainEnabled": false,
        "localeToBeUsed": "en-US",
        "localeStrategy": "browser"
    },
    "appDetails": {
        "appName": "AdminConsole",
        "appId": "6a285fd7-4403-48f4-893f-962fa391115b"
    },
    "x-mfa-username": "devuser01"
}
```


Where,

- "tenantCustomHEADContent" - this value returns the Base64 HTML Code that was added in the tenant/branding setting.
- "tenantCustomBODYContent" - this value returns the Base64 HTML Code that was added in the tenant/branding setting.

SignIn UI Behavior

Based on the configurations specified in the tenant/branding settings, the changes are reflected in the SignIn UI.

Signin UI Consent for Resource Server Applications

This topic describes the Signin UI's behavior when an application requires user consent for specific privileges.

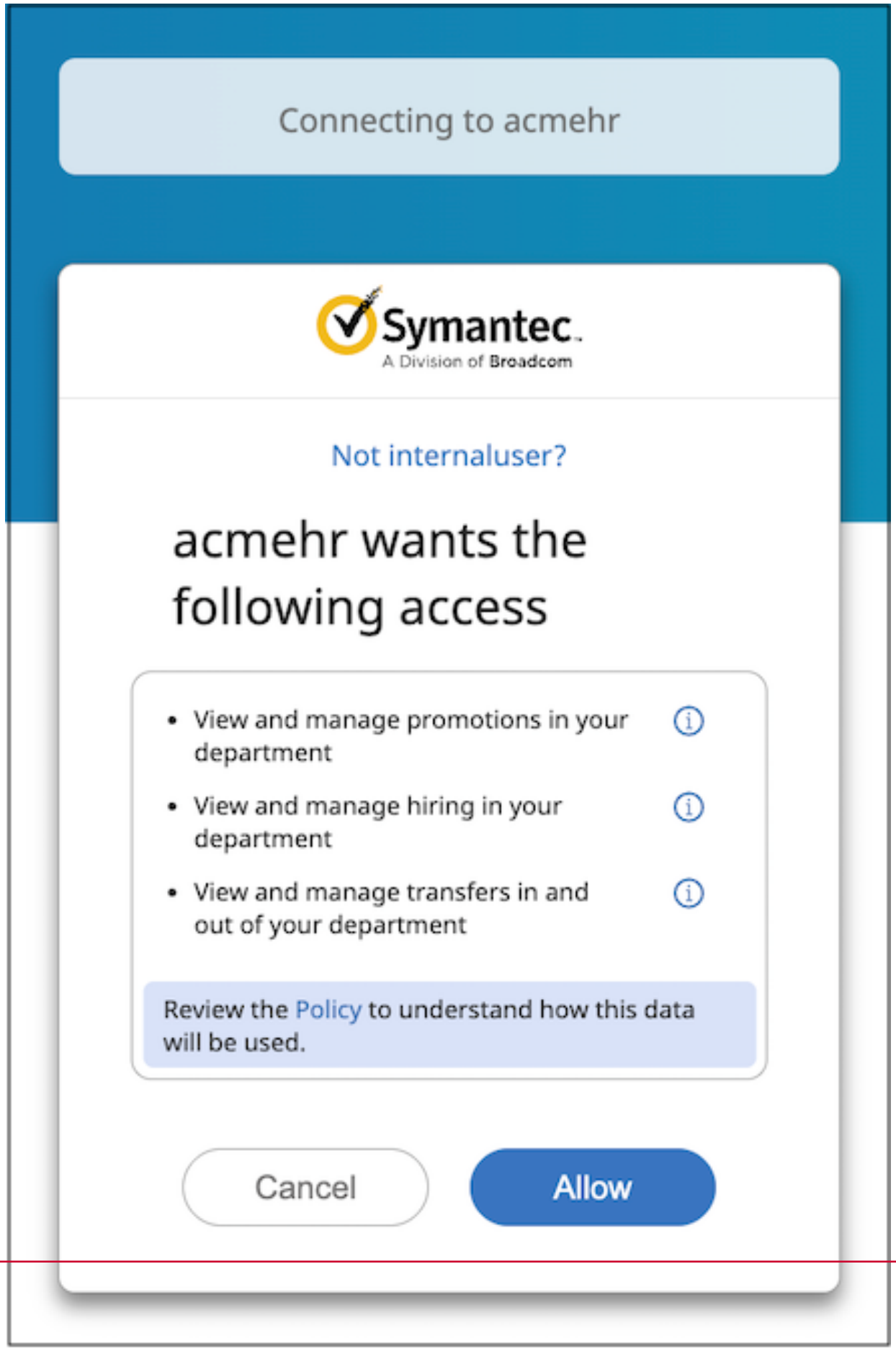
When an application is configured to require user consent for specific privileges (scopes), the Signin UI presents a consent screen to the user. This screen details the access the application is requesting and allows the user to either "Allow" or "Cancel" the request.

Admin Console Configuration

In the Admin Console, under the "OAuth Resource Server" tab for an application, administrators can define privileges (scopes) and their associated consent labels and descriptions. For instance, a "promote" scope might have a "Consent Label" of "Scope to promote employees" and a "Consent Description" of "View and manage promotions in your department." These values are displayed to the user on the Signin UI consent screen.

Runtime Behaviour

When a user attempts to access an application that requires consent for specific scopes, the Signin UI displays a screen similar to the following screen.


Connecting to acmehr
Symantec.
A Division of Broadcom
Not internaluser?
acmehr wants the following access
View and manage promotions in your department
View and manage hiring in your department
View and manage transfers in and out of your department
Review the Policy to understand how this data will be used.
Cancel
Allow

This screen dynamically populates the consent details based on the configuration in the Admin Console.

The user is displayed the following options:

- **Application Name**: The name of the application requesting access (for example, "acmehr").
- **Requested Access**: A list of privileges (scopes) the application is requesting, along with their descriptions (for example, "View and manage promotions in your department").
- **Usage Policy URL**: A link to the application's usage policy, if configured (for example, "https://policy.hr.com").
- **Action Buttons**: "Allow" and "Cancel" buttons, allowing the user to grant or deny the requested access. If the user clicks "Allow," the application is granted the requested privileges. If the user clicks "Cancel," the access is denied. This mechanism ensures that users have explicit control over the data and actions an application can perform on their behalf.

Reference Links

- Creating a Resource Server using REST API
- Creating a Resource Server using Admin Console
- Resource Server Application Flow (Including OIDC Flow)

Support for Factor Enrollment Status Indicators with Tenant Configuration and CSS Customization

Learn about MFA Factor Enrollment Status in the Signin UI screen - Tenant Settings and Styling.

The factor selection screen displays enrollment status for authentication factors, showing users which methods are ready to use versus those requiring enrollment. Users can see a clear distinction between factors they can immediately use and those requiring initial setup. Enrolled factors appear ready for selection, while unenrolled factors display an "(enrollment required)" indicator. When users select an unenrolled factor, they seamlessly enter an inline enrollment workflow that maintains their current authentication session, allowing them to complete setup and use their newly configured authentication method without interrupting their signin process.

Tenant Configuration to enable Factor Enrollment Status Indicators

Enable the following tenant setting to get the factor enrollment status text displayed in the factor screen:

```
returnCredentialsInFactorSelection=true
```

The default is set to **false**.

Customize Factor Enrollment Text

Use the Localization API to add the new locale key, "**factorselection.enrollmentRequired**" and change the content of the enrollment text.

The following is an example showing how the text can be modified:

API Endpoint

```
PATCH https://${SSP_MGMT_FQDN}/systemTenantName>/admin/v1/LocaleContent/{{locale-name}}
```

API Request

```
{
    "content": {
        "factorselection": {
            "enrollmentRequired": "<br/><span class=\"enrollmentRequiredText\"><b><i>(enrollment is needed)</
i></b></span>"
        }
```

```
    }
}
```

Style Factor Enrollment Text with CSS

To modify the style of the Enrollment text, use the Tenant Custom CSS and modify the following CSS class:

```
enrollmentRequiredText
```

The following is an example showing how the text can be styled:

```
.enrollmentRequiredText {
    color: blue;
    font-weight: bold;
    font-style: italic;
}
```

Localizing the Symantec Identity Security Platform UI Consoles

Symantec Identity Security Platform supports localization for the SignIn User Interface (UI), and Self-Service Console (UI) allowing end users to view the SignIn UI and Self-Service Console screens in their preferred language. To localize these UI's, two sets of APIs are available at the system and tenant levels.

All localization content described in this topic apply to the SignIn User Interface (UI) and Self-Service Console UI components of Symantec Identity Security Platform

This topic covers the following sections:

- Supported Locales
- OOTB Supported Locales
- Locale Selection Flow

Supported Locales

The supported locales for the UI are listed in the following link:

http://www.lingoes.net/en/translator/langcode.htm

OOTB Supported Locales

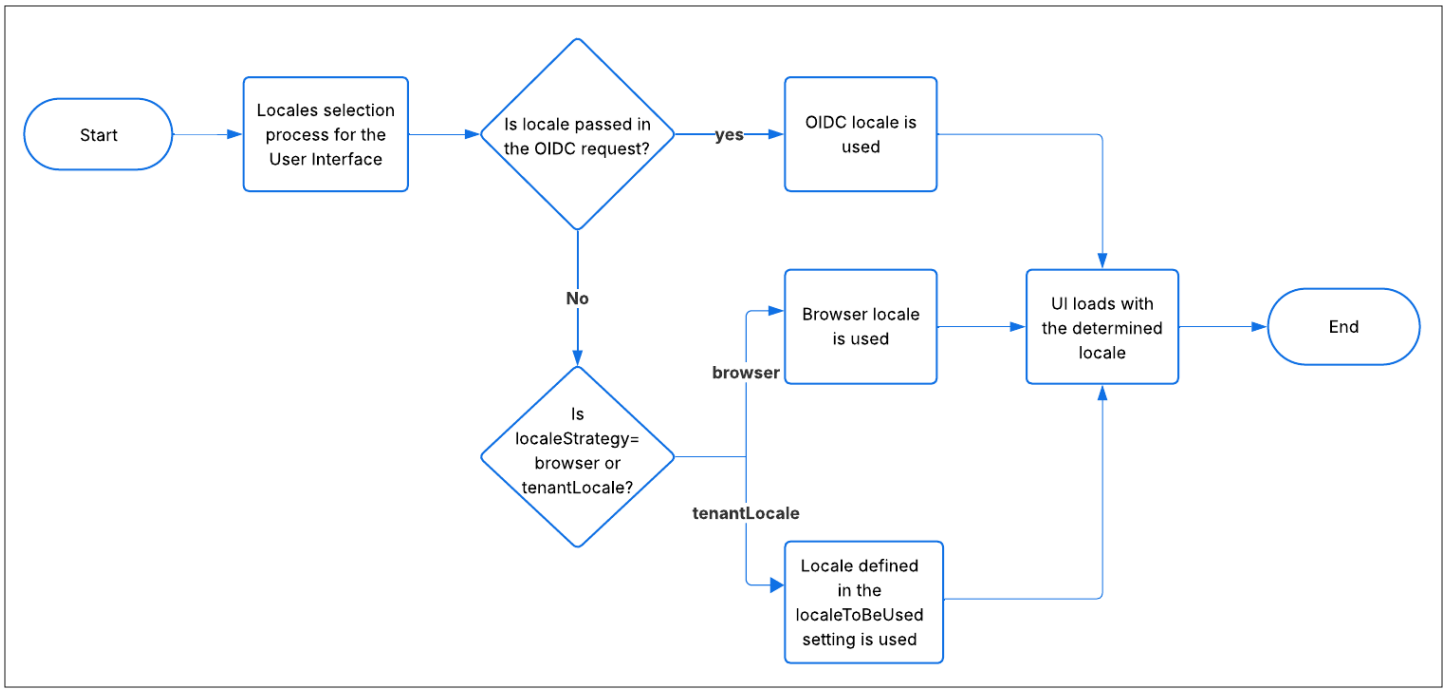
The OOTB supported locales for the UI are as follows:

- pt-BR → Portuguese (Brazil)
- es-ES → Spanish (Castilian), Spanish (Spain)
- fr-FR → French (France)
- ja-JP → Japanese (Japan)
- en-US → English (United States)

Locale Selection Flow for the UI

The following diagram describes how the locale selection flow for the UI works:

Figure 34: Locale Selection Flow for the UI



Use Case 1:

This use case takes precedence over the other cases. In this use case, the locale is passed in the OIDC request query parameter (ui_locales). When the UI loads, it invokes the following two APIs in the given sequence.

1. **BrandingSettings API ("…/auth/v1/BrandingSettings" API):** The UI ignores the locale flags **(localeToBeUsed, localeStrategy)** that are passed in this API as it considers the locale that is passed in the OIDC query parameter.
2. **UILocaleContent API ("…/auth/v1/UILocaleContent" API):** The OIDC locale is passed in the request payload of the **UILocaleContent** API. The API returns the localized content for the locale as a response and the UI is displayed in that locale.

Use Case 2:

In a non-OIDC request, the locale for the UI is determined using the tenant locale settings **(localeStrategy, localeToBeUsed).** When the UI loads, it invokes the following two APIs in the given sequence.

1. **BrandingSettings API ("…/auth/v1/BrandingSettings" API):** The UI checks if the localeStrategy is set to **browser** or **tenantLocale** in the Branding Settings API.
   - If the localeStrategy is set to browser, then the browser locale is used.
   - If the localeStrategy is set to tenantLocale, then the locale that is configured in the **localeToBeUsed** setting is considered.
2. **UILocaleContent API ("…/auth/v1/UILocaleContent" API):** Depending on the tenant locale settings passed in the BrandingSettings API, the determined locale is passed in the request payload of the UILocaleContent API. The API returns the localized content for the locale as a response and the UI is displayed in that locale.

Fallback Use Case:

If the locale determined in use cases 1 and 2 does not exist in IDSP, then the UI falls back to the following flow.

Flow Summary:

The process starts by specifying a locale in the "…/auth/v1/UILocaleContent" API, e.g., "fr-FR".

- **Tenant-Level Search**:
  IDSP first searches for the locale at the tenant level based on the request payload of ".../auth/v1/UILocaleContent" API.
  - If found, the tenant-level locale is returned in the response of the ".../auth/v1/UILocaleContent" API.
  - If not found, it proceeds to the next step.
- **System-Level Search**:
  IDSP searches for the locale at the system level.
  - If found at the system level, that locale is returned in the response of the ".../auth/v1/UILocaleContent" API.
  - If not found, it moves on to the next step.
- **"localeToBeUsed" Configuration Search**: IDSP checks the "localeToBeUsed" configuration in the tenant settings.
  - If the locale exists in the tenant level, it is returned in the response of the ".../auth/v1/UILocaleContent" API.
  - If not, the system level locale is returned in the response of the ".../auth/v1/UILocaleContent" API.

Reference Links

- To learn about the Localization APIs, see Localization APIs.

Localization through IDSP APIs

Learn how to manage UI localization content using system and tenant-level APIs to support multiple languages and locales in your application.

All localization API content described in this topic apply to the SignIn User Interface (UI) and Self-Service Console UI components of Symantec Identity Security Platform

This topic covers the following sections:

- Managing localization through IDSP APIs
  - System APIs
  - Tenant APIs
  - Examples
- Locale creation flow
- Updating the custom locale content manually when additions are made in the OOTB Locales
- Error codes

Manage Localized Content for a Locale using the System and Tenant Level APIs

A system or tenant administrator can localize the UI in the supported languages by adding localized content for the preferred locales to the IDSP server using the provided system and tenant-level APIs.

System Level API Operations

> **NOTE**
>
> Self-Service Console localization requires additional API parameters and content keys beyond the SignIn UI elements shown in the API examples. For Self-Service Console localization details, refer to Additional Information on Localizing Self-Service Console.

The following table lists the system-level APIs that a system administrator can use to create and manage localized content:

Table 119: System Level Locale Content APIs

Operation Description	Operation Type	Endpoint URLs
Creates localized content for a new locale	POST	https://${SSP_MGMT_FQDN}/<systemTena
Updates the localized content for a specific locale	PUT	https://${SSP_MGMT_FQDN}/<systemTena
Partially updates out-of-the-box supported locales only	PATCH	https://${SSP_MGMT_FQDN}/<systemTena
Deletes localized content for specified locale	DELETE	https://${SSP_MGMT_FQDN}/<systemTena
Retrieves content for a specific locale	GET	https://${SSP_MGMT_FQDN}/<systemTena
Lists all available locales in IDSP	GET	https://${SSP_MGMT_FQDN}/<systemTena

NOTE

- To know the value of ${SSP_MGMT_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<systemTenantName>` is "system".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints.
  For example:`https://${SSP_MGMT_FQDN}/<systemTenantName>/admin/v1/LocaleContent` should look similar to:`https://mgmt-example.com/system/admin/v1/LocaleContent`
- To view the requests and responses of these APIs, see the UI Localization API Swagger documentation.

Tenant Level API Operations

NOTE

- At the tenant level the OOTB languages are not preloaded during the IDSP deployment, unlike in the system level.
- If the same locales exist in both system and tenant level, then the tenant level locale takes precedence.
- Self-Service Console localization requires additional API parameters and content keys beyond the SignIn UI elements shown in the API examples. For Self-Service Console localization details, refer to Additional Information on localizing Self-Service Console.

The following table lists the tenant-level APIs that a tenant administrator can use to create and manage localized content in the UI for a preferred locale:

Table 120: Tenant Level Locale Content APIs

Operation Description	Operation Type	Endpoint URLs
Creates localized content for a new locale	POST	https://${SSP_FQDN}/<tenantName>/admir
Updates the localized content for a specific locale	PUT	https://${SSP_FQDN}/<tenantName>/admir
Deletes localized content of specific locale	DELETE	https://${SSP_FQDN}/<tenantName>/admir
Retrieves content for a specific locale	GET	https://${SSP_FQDN}/<tenantName>/admir
Lists all available locales in IDSP	GET	https://${SSP_FQDN}/<tenantName>/admir

".../UILocaleContent" API - used to retrieve the locale in which the UI should display content

Operation Description	Operation Type	Endpoint URLs
Depending on the tenant locale settings passed in the BrandingSettings API, the determined locale is passed in the request payload of the UILocaleContent API. This API returns the localized content for the locale as a response and the UI is displayed in that locale.	POST	https://${SSP_FQDN}/<ten

NOTE

- To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
- The default value for `<tenantName>` is "default".
- Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints.
  For example:`https://${SSP_FQDN}/<tenantName>/admin/v1/LocaleContent` should look similar to:`https://example.com/default/admin/v1/LocaleContent`
- To view the requests and responses of these APIs, see the UI Localization API Swagger documentation.
- The system reads the locale from the ui_locale parameter and converts any underscores ("_") to hyphens ("-"). For example, if ui_locale is provided as "en_US", it is converted to "en-US" before being sent to the UI.

Examples

NOTE
The Request and Response examples mentioned here are common to both Tenant and System Level APIs. The following examples are shown using the ".../<systemTenantName>/admin/v1/LocaleContent" API. The same operations can be performed using the ".../<tenantName>/admin/v1/LocaleContent" API.

POST operation: Sample request to create localized content for the locale (en-CB)

```
POST https://${SSP_MGMT_FQDN}/<systemTenantName>/admin/v1/LocaleContent
```

Request

```
{
"locale": "en-CB",
"content": {
"home": {
"welcomeMsg": "Welcome !!",
"appMsg": "{{appName}} app is running!"
},
"messages": {
"clientToken": "Client Token:"
},
"app": {
" Title": "Sign In",
"deviceCodeTitle": "Connect Device"
},
...
}
```

Response

```
{
"locale": "en-CB",
"content": {
"home": {
```

```
"welcomeMsg": "Welcome !!",
"appMsg": "ssoSPI app is running!"
},
"messages": {
"clientToken": "Client Token:"
},
"app": {
"signInTitle": "Sign In",
"deviceCodeTitle": "Connect Device"
},
...
}
```

PUT operation - Sample request to update localized content for the locale (en-CB)

```
PUT https://${SSP_MGMT_FQDN}/<systemTenantName>/admin/v1/LocaleContent
```

Request

```
{
"content": {
"home": {
"welcomeMsg": "Welcome !!",
"appMsg": "{{appName}} app is running!"
},
"messages": {
"clientToken": "Client Token:"
},
"app": {
"signInTitle": "Sign In",
"deviceCodeTitle": "Connect Device"
},
...
}
```

Response

```
{
"locale": "en-CB",
"content": {
"home": {
"welcomeMsg": "Welcome !!",
"appMsg": "ssoSPI app is running!"
},
"messages": {
"clientToken": "Client Token:"
},
"app": {
"signInTitle": "Sign In",
"deviceCodeTitle": "Connect Device"
},
...
}
```

GET operation - Sample request to retrieve localized content for the locale (en-CB)

```
GET https://${SSP_MGMT_FQDN}/<systemTenantName>/admin/v1/LocaleContent
```

Response


```
{
"locale": "en-CB",
"content": {
"home": {
"welcomeMsg": "Welcome !!",
"appMsg": "ssoSPI app is running!"
},
"messages": {
"clientToken": "Client Token:"
},
"app": {
"signInTitle": "Sign In",
"deviceCodeTitle": "Connect Device"
},
"labels": {
"or": "or",
"signInWith": "Sign in with",
"selectAProvider": "Select a Provider",
"seeAllProviders": "See all providers",
"searchAllProviders": "Search all providers",
"continueWithSignIn": "Continue with Sign In"
},
"login": {
"title": "Sign In",
"userName": "Username",
"placeholder": "Enter your username",
"errorMessage": "Username is required",
"byoiLinkText": "Use Custom Domain",
"registerLink": "Register",
"privacyLink": "Privacy Policy"
},
...
}
```


For sample request and response payloads of all these URLs, refer to UI Localization API - Swagger documentation.

Locale Creation Flow

In this section, we explain with an example the end-to-end steps that a system administrator must follow to create a locale (say Italian). The same steps can be executed by invoking the tenant level APIs as well.

1. Get the English locale content for the UI by invoking the GET payload - `https://${SSP_MGMT_FQDN}/<systemTenantName>/admin/v1/LocaleContent/en-US`.
2. Translate the English locale content into the Italian language.
3. Add the Italian localized content to IDSP by invoking the POST payload - `https://${SSP_MGMT_FQDN}/<systemTenantName>/admin/v1/LocaleContent`
4. Configure the following tenant settings to make Italian as the default locale:
   - localeStrategy=tenantLocale
   - localeToBeUsed=it-IT

NOTE
This step is applicable only when you have to set a locale as the default.

5. The localized content for the Italian locale has been added successfully to the IDSP and you can update it anytime by invoking the PUT payload - `https://${SSP_MGMT_FQDN}/<systemTenantName>/admin/v1/LocaleContent/it-IT`

Updating Custom Locale Content manually when new keys are added in the OOTB Locales

This section outlines the steps needed to update the custom locale content manually when new keys are added to the OOTB Locales content.

Assumptions:

- The customer has created a custom locale based on the "en-US" locale, which is one of the out-of-the-box (OOTB) locales.
- Customer is using IDSP out-of-the-box (OOTB) UI.

Use Case:

A customer using an older version of IDSP has created custom locale content based on the out-of-the-box "en-US" locale for the IDSP UI. After upgrading to the latest version of IDSP, which includes new keys in the OOTB "en-US" locale, the customer must manually synchronize these updates in their custom locale. This involves comparing the updated "en-US" locale content with their custom locale and manually adding any new keys found, along with their corresponding values, to the custom locale content. You can perform the following steps to synchronize the updates to your custom content.

Steps to Follow:

1. Retrieve the English locale "en-US" content for the UI by making a GET request: `https://${SSP_MGMT_FQDN}/system/admin/v1/LocaleContent/en-US`.

   NOTE
   You can replace the system-level API with the tenant-level API to make the GET request.
2. Identify any new keys in the "en-US" content from the GET response.
3. Manually add these new keys and their corresponding values in the custom locale content.
4. Perform a PUT operation to update the custom locale content with any changes to the existing "en-US" content.

   NOTE

   - If the PUT operation is performed without manually adding the new keys, an error may occur. This is because the PUT call only updates existing keys and does not add new keys, causing the API to return an error when it cannot find the new keys. Hence, step 3 is important.
   - If you are using a custom UI instead of the ***OOTB IDSP UI***, you must add any new keys found in step 2 from the OOTB locale. However, you can leave the values blank if those keys are not required for your ***custom*** UI.

Error Codes

9800001	Locale with the given name already exists
9800002	Locale with the given name does not exist
9800003	Locale cannot be deleted as the entry is referenced in tenant settings
9800004	Pre-loaded Locale bundles are not allowed to be modified
9800005	Given Locale is not supported

Reference Links

- To know how to use a custom SignIn UI application that is developed using the IDSP APIs, see Using Custom SignIn UI Application.

Additional Information on localizing the Self-Service Console

Symantec Identity Security Platform supports localization of the Self-Service Console, allowing end users to view the Self-Service Console screens in their preferred language.

The localization of Self-Service Console is executed by the System or Tenant admins.

Fresh Installation Scenario

When IDSP is freshly installed, out-of-the-box locales are set up automatically. These include "content" for the SignIn UI and "selfServiceContent" for the Self-Service Console.

Creating Custom Locales

You can create custom locales using the POST API (.../admin/v1/LocaleContent), depending on whether you plan to use the Self-Service Console:

- **Without Self-Service Console**: Create a custom locale without providing "selfServiceContent."
- **With Self-Service Console**: Create a custom locale that includes "selfServiceContent."
  Important: If you create a custom locale with "selfServiceContent," then any future updates using the PUT API must always include both "content" and "selfServiceContent" in the request.

This example shows a combined locale payload, including "**content**" for the Sign-In UI application and "**selfServiceContent**" for the Self-Service Console:

```
{
"locale": "{{ custom-locale }}",
"content": { /* ... content JSON ... */ },
"selfServiceContent": { /* ... self-service content JSON ... */ }
}
```

Upgrade Scenario

When IDSP is upgraded, out-of-the-box locales are automatically updated to include "selfServiceContent." These can later be modified using the PATCH API.

Creating Custom Locales:

- **Created Before Upgrade**: Custom locales that existed prior to the upgrade will continue to function normally. When using the PUT API to update these locales, "selfServiceContent" is not required in the request body unless explicitly added later.
- **Created After Upgrade**: Administrators have the option to include "selfServiceContent" when creating new custom locales using the POST API (along with "content"), depending on their use of the Self-Service Console.
  - If a custom locale was created **without** "selfServiceContent," subsequent PUT API updates does not expect "selfServiceContent" in the request.
  - If a custom locale was created **with** "selfServiceContent," then all future PUT API updates must include both "content" and "selfServiceContent" in the request body.

API Examples

For API examples, see Localization APIs.

Admin Console

This section explains how to use the Admin Console to configure your tenant, authentication, and integration settings and monitor tenant-level activities in the Symantec Identity Security Platform Admin Console.

The Admin console is a web-based user interface where tenant administrators can update default configuration properties, add applications, create authentication policies to govern the applications, manage certificates, and keys, etc. You must be a tenant administrator to make changes in the Admin Console.

To access the Admin Console in your IDSP environment, use the following URL format and replace the {SSP_FQDN} with the appropriate value:

```
https://${SSP_FQDN}/default/ui/v1/adminconsole
```

Where,

- ***${SSP_FQDN}*** is the fully-qualified-domain-name of the host where you installed IDSP. See Configuring the Environment Variables.
- ***default*** is the unique tenant name in IDSP

An audit log tracks all changes made in the Administrative Console. The username of the author and the date of the change is recorded in the log.

Logging Into the Admin Console

When you use the above URL to access the Admin Console, the Symantec Signs In page is displayed. On the Sign In page, enter your username followed by the password. The username and password is provided to you by the Broadcom tenant administrator. Once you log in successfully, the Dashboard page is displayed. There are other icons on the console's navigation panel which helps in navigating to the other modules, such as, Applications, Policies, Identity Providers, and more.

To access information on each of the listed topics, choose the corresponding module and click its respective link to navigate to the relevant page.

- Dashboard: Allows you to view and manage the data in your IDSP account.
- Users: Allows you to search users using username or other filter options.
- Applications: Allows you to add OAuth 2 complaint applications in the solution.
- Policies: Allows you to create two types of policies, Authentication Policy and IDP Discovery Policy.
- Identity Providers: Allows you to add Identity Providers in IDSP.
- Certificates: Allows you to manage public keys or certificates (certificates) of other parties for token validation, token encryption, and SSL communication.
- Accounts: Displays the details of all the user accounts that are synchronized with IDSP.
- Profiles: Displays information about your roles, the roles that you have delegated to others, and what resources have been delegated to you
- Settings: Allows you to manage admin roles, customize the Signin UI's branding, configure tenant level settings, securely upload, and manage secrets and keys used during secured connections, configure Identity Stores where IDSP can search for users.
- User and Group Management: Shows you how to use the User and Group icons to manage internal Identity Store Users and Groups. For more information, see Managing Users and Managing Groups.

  NOTE
  The term IDSP has been interchangeably used as "Solution" in the documentation.

Reference Link to:

- Using Identity Federation to Access IDSP Admin Console

User and Group Management

This topic shows you how to manage Internal Identity Store Users and Groups in Symantec Identity Security Platform using the Admin Console UI.

The User and Group icons on the left panel of the Admin Console let you manage internal Identity Store Users and Groups in the IDSP. These Users and Groups can be integrated with IDSP to leverage its modern authentication and authorization services.

The Admin Console experience reflects the current privileges of each administrator. Therefore, to access and modify information within the console, you must possess the necessary privileges. You can view the User and Group pages because you hold the following privileges:

- `urn:iam:t.users`
- `urn:iam:t.groups` urn:iam:t.groups
- `urn.iam.t.sessions`

Managing Users

Use the following procedures to manage users.

Managing users consists of the following options:

- Accessing the User UI
- Searching for Users
  - Using the Default User Search
  - Search Using Defined Filters
- Viewing a User
- Adding New Users
  - User Tabs and Tab Details
  - Entering Profile Screen User Information
  - Email and Phone Information
  - Setting a User's Password
  - Sending an Email to Establish a User's Password
  - Entering User Status Information
- Updating Users
  - Updating User Search Results
  - Updating a Selected User
  - Viewing and Managing a User's Groups
- Deleting Users
- Resetting a User Password
- Scoping Constraint Enforcement in the User Management Console

Accessing the User UI

To access the part of the Admin UI that allows you to manage users, follow these steps:

1. Log in to the Admin Console.

2. On the left panel of the Admin Console, hover over the

icon, and the select the **Users** link.
The initial page that appears allows you to search for existing users or add a user.

Searching for Users

The search page allows for a default search or a search using defined filters.

Using the Default User Search

To use the default search, enter a `userName` value in the provided search box, and then select the **Submit** button to search for that user. The default search operator attempts to match a `username` value using a `starts With` operator. Any users found appear in the search results.

Search Using Defined Filters

You can customize the search using defined search filters. The **Show Filters** button allows you to define specific search filters by attribute values and a specific search operator. The **Submit** button executes the search using any defined filters to display the search results.

To define search filters, follow these steps:

1. Select an attribute, an operator, and a value.
2. To add more filtering, select the **+** button.
3. If entering more than one filter, select the appropriate **Match** criteria as follows:
   a. **Any**: Combines the filter using an OR operator.
   b. **All**: Combines the filters using an AND operator.

Showing and Hiding Filters

You can show or hide the defined filters using the **Show/Hide Filters** button. Hiding the filters does not clear the filters, as they are still in place when you select the **Submit** button.

Clearing Search Results

When using the Defined Filters search model, select **Clear Filters** to clear the current search result and reset the defined filters.

Viewing a User

Once you have found the user that you searched for, click the name of the user to view the user details page. The user details page displays any of the following information about the user that you have collected:

- The **Status** tab displays the status of the user, including whether the user is locked or disabled, the status of the user password, if the user is a federated user, the last time that the user successfully authenticated, and more.
- The **Profile** tab lets you view the following details about the user:

  - **Basic Info** lists identifying information about the user such as name, birthdate, gender, time zone, and title, if any.
  - **Contact Info** lists email, phone, and address information for the user.
  - **Business** lists division, department, and employee information about the user.
  - **Additional** lists any further information that you have collected, including any custom-defined attributes that you have configured.
- The **Groups** tab displays any groups to which this user has been assigned. Click the name of the group to view details about the group. See Managing Groups.
- The **Authorization** tab lets you view and modify the resource authorizations that have been assigned to this user. This window provides the following sections:
  - **Tags**: Allows you to add tags to this user. The tag is a name-value pair that you use to assign an attribute to this resource. Once a resource is tagged with a criteria, you can include that criteria in a constraint statement for any policy that affects that resource.

    The format must be `name=value`. However, there are no restrictions on the name or value that you assign.
  - **Administrative Roles**: Displays a list of administrative roles that are assigned to this user.
  - **Resources Managed By <Application Name>**: Displays a list of the resources that this user can manage.
  - **Resources Delegated By <Application Name>**: Displays a list of the resources that this user has delegated to other entities.
  - **Modified By**: Displays the user name and date and time stamp for the most recent creation or modification of the user.
- The **Access** tab displays a list of applications that the user has access to. It shows a table of applications. For each application, the following details are provided:
  - **APPLICATION**: The name of the application.
  - **POLICY**: Indicates if a policy is associated with the application.
  - **ROLE**: If any roles have been assigned to this user, the number of roles granted appears. You can click on the role value to open the edit roles dialog. You can revoke access to this role by clicking the ellipses button.
  - **DESCRIPTION**: A brief description of the application.
  - **SEARCH**: A search bar, "Search by application name or description" is available to filter the list of applications.
  - **Grant Role Access**: Indicates that you can assign roles or permissions for applications to the user.

This page also allows you to perform the following actions:

- **Delete**: Use this option to delete the user. A reconfirmation is requested before deleting a user. Once confirmed, the user is deleted.
- **Reset Password**: Use this option to force the user to reset the password. The user is prompted to reset the password the next time that the user attempts to authenticate.

Adding New Users

Adding new users consists of the following general steps:

1. From the initial **User** search screen, select the **Add User** button. The **Profile** tab and **Status** tab appear.
2. Enter the information in each of the **Profile** tab and **Status** tab, as desired. Required fields are marked with an asterisk (*). The **Save** button is only enabled when each required field has a value.
3. Select **Save** to add the new **User**.

User Tabs and Tab Details

The default tabs for users creation appear in the following table.

Tab Name	Mode Available	Description
Profile	Add or View/Edit user	Basic Info, Contact Info, Business, and Additional sections. An additional section contains the custom-defined attributes.
Status	Add or View/Edit user	State attributes
Groups	View/Edit User	The groups assigned to the user.
Modified	View/Edit User	Create and Modify data

Entering Profile Screen User Information

User Name Information

A user name must be unique within the provided tenant. *A user name cannot be changed after the user is created*.

You have a number of options to specify a user name for the user. The user name includes the following options:

- Use the provided primary email address for the user name. By default, email addresses must be unique across the tenant.
- Create a user name using the combination of **First Name** + **Last Name**. If that combination is not unique, the system appends a unique value after the Last name portion to make the generated user name unique.
- Allow the Administrator to enter a unique user name. If the entered user name is not unique, the **Save** fails, indicating that the user name is already in use.

Email and Phone Information

- **Email**
  Email is a required attribute when creating a User. In Symantec Identity Security Platform, emails are treated as a credential. Emails are used for IDSP notifications, and can be used as an authentication factor. Multiple email addresses can be set on the User. One of the email addresses must be marked as the primary email address. The primary email address is the default email that is used to deliver IDSP notifications. These notifications include the delivery of User activation emails or emails containing Password Reset links. A validated email is required for Password Reset delivered emails. User emails added to the user in the User Management page are considered validated by default. A valid email address is expected for proper delivery of IDSP generated emails.
- **Phones**
  Phones that are defined for a user are treated as an SMS credential in IDSP. Adding a phone to a user in the **User Management** page defines the capability type as SMS. This capability type allows phones to receive IDSP generated messages for various operations, such as delivery of an OTP for authentication, or delivering messages containing password reset links. SMS delivery requires a validated phone. By default, phone numbers that are added in the User Management page are considered validated.
  To ensure the highest level of interoperability with various SMS providers, administrators should define phone values using the E.164 international phone number standard. E.164 number standard is as follows:
  ```
  [+][country code][area code][local phone number]
  ```
  So a phone number that uses the E.164 phone number format might look something like this:
  ```
  +14151231234
  ```

  NOTE
  E. 164 uses no spaces, no dashes, no parenthesis, and no periods.

Setting a User's Password

The available options for setting a user password depend on the **Administrator Generated Password Setting** defined on the **Internal Identity Store Configuration**. For more information, see the `PasswordResetInitiator` and Email Template section of the Password Management in the Identity Service topic.

In User Management, the **Administrator Generated Password Setting** determine the available **Administrator Password Options**:

Administrator Can Generate Password Setting	Available User password options
false	• Send an email to a User to establish the password
true	• Send an email to a User to establish the password • Create a Temporary Password

Sending an Email to Establish a User's Password

This option relies on the user having a verified primary email address. An email address that is entered by an administrator is considered verified by default. When choosing this option, an email is sent to the primary email for the user. This email contains a link to allow the user to establish their password.

If you have configured an **End User Password Reset URLs** in the internal Identity Store, those URLs are used for the link sent in the email sent to the end user. See Internal Identity Store Configuration.

The email is based on a template whose contents can be customized. For more information, see the "Email and Phone Template" section in the Password Management in the Identity Service topic.

Generating a Temporary Password

This option sets a user password to a temporary value, and sets the `passwordMustChange` option to true. When selecting Save, a dialog opens allowing the administrator to both copy the password and to communicate the temporary password to the user.

> **NOTE**
> IDSP does not provide a mechanism to transmit the temporary password to the User. The onus is on the Administrator to do so in a safe and secure manner.

Entering User Status Information

You can enter the new status for the user on the **Status** tab. The status values and their operational semantics are described in the Managing the Identity State topic.

Updating Users

Updating User Search Results

From the **User** search results screen, you can use the following options from the

icon at the end of each user row.

- **Reset Password**
- **Disable**
- **Delete**

Updating a Selected User

When selecting a user from the **User Search** list, the **Status**, **Profile**, **Groups**, and **Modified** tabs appear.

For the **Profile** tab, the default mode is **View**. To enter edit mode, select **Edit**.

> **NOTE**
> The **Edit** option is only available in the Profile tab.

Viewing and Managing a User's Groups

When selecting a user, you can use the **Groups**tab lets you view and manage the Groups to which the user is assigned. The current groups to which the user is assigned appear by default.

Adding a User to a Group

You can use the following options to add a user to other groups:

- Add the user to an existing group.
- Create a group, and then add the user to that new group.

Searching for Existing Groups

The search option for locating existing groups supports a single search edit field. The value in this field searches for the **Group by Name** or **Description** using a `starts with` search operator.

Removing a User from a Group

To remove a user from a group, select the

icon next to the user.

Deleting Users

The **User Search** results screen or the **User Selected** screen provide options to delete users.

From the **User Search** results screen, select the

icon at the end of each user row to delete the user.

From the **User Selected** screen, the **Delete** button at the top of the page deletes the selected User.

Resetting a User Password

An administrator can reset a user password from either the **User Search** results screen or the **User Selected** screen. These options are described in the Setting a User's Password section in this topic.

Scoping Constraint Enforcement in the User Management Console

The **User Management** console enforces scoping constraints applicable for the currently logged-in administrator. The following screen display semantics apply to attribute permissions:

1. For attributes on the **Status**, **Profile**, and **Modified** tabs
   a. Read-only attribute permissions set the screen control as disabled for input.
   b. No permission hides the attribute control on the screen.
   c. If all of the attributes of a **Profile Screen** section have no permission, the entire section is hidden.
2. If the `User.groups` (list of users groups) attribute has no permission, then the current set of Groups that are assigned to the user is not shown.
3. When using **User Search** filters, the list of attributes that are shown are only the attributes for which the user has `r-w` permission.

Managing Groups

Use the following procedures to manage groups.

Managing groups consists of the following options:

- Accessing the Group UI
- Searching for Groups
- Viewing a Group
- Adding New Groups
- Updating Groups
- Changing the Group Name or Description
- Viewing and Searching for Group Members
- Adding and Removing Group Users
- Deleting a Group

Accessing the Group UI

To access the part of the Admin UI that allows you to manage groups, follow these steps:

1. Log in to the Admin Console.
2. On the left panel of the Admin Console, hover over the

   icon, and the select the **Groups** link.
   The initial page that appears allows you to search for existing groups or add a group.

Searching for Groups

The search option for locating existing groups supports a single search edit field. The value in this field searches for the group by name or description using a `starts with` search operator.

The group search results display any matching groups. The results contain the **Group Name**, **Description**, and the **Total Member Count**.

Viewing a Group

Once you have found the group that you searched for, click the name of the group to view the group details page. The group details page displays the following information about the group:

- The **Users** tab displays the all of the users who are assigned to the group. Click the name of the group to view details about the group. See Managing Users.
- The **Managers** tab lets you view and modify the entities that can currently manage this group. From the vertical ellipses (**...**) next to each row, perform one of the following actions for that group:
  - Select **Assign** to assign entities that can manage this group.
  - Select **Revoke** to revoke the assignment of entities that can manage this group.
- **Authorization** - To view and modify the resource authorizations that have been assigned to this group, select this tab. This window provides the following sections:
  - **Tags**: Allows you to add tags to this group. The tag is a name-value pair that you use to assign an attribute to this resource. Once a resource is tagged with a criteria, you can include that criteria in a constraint statement for any policy that affects that resource.

The format must be `name=value`. However, there are no restrictions on the name or value that you assign.
  - **Administrative Roles**: Displays a list of administrative roles that are assigned to this group.
  - **Resources Managed By**: Displays a list of the resources that this group can manage.
  - **Modified By**: Displays the user name and date and time stamp for the most recent creation or modification of the group.
- The **Access** tab displays a list of applications that the group has access to. It shows a table of applications. For each application, the following details are provided:
  - **APPLICATION**: The name of the application.
  - **ROLE**: If any roles have been assigned to this group, the number of roles granted appears. You can click on the role value to open the edit roles dialog. You can revoke access to this role by clicking the ellipses button.
  - **DESCRIPTION**: A brief description of the application.
  - **SEARCH**: A search bar, "Search by application name or description" is available to filter the list of applications.
  - **Grant Role Access**: Indicates that you can assign roles or permissions for applications to the user.

This page also allows you to perform the following actions:

- **Delete Group**: Use this option to delete the group. A reconfirmation is requested before deleting a group. Once confirmed, the group is deleted.
- **Add Users**: Use this option to add an existing user to the group.
- **Assign Manager**: Use this option to assign an entity to manage this group.

Adding New Groups

The **Group Search** page provides an option to add a group. When adding a group, you enter the **Group Name** and **Description**. The **Group Name** must be unique within the tenant. The group name can be changed after creation.

Updating Groups

Selecting a group from the **Group** search results puts the group in edit mode.

Changing the Group Name or Description

To enter edit mode, either select the **Group Name** and **Description** fields, or hover over the field and select the

icon.

Either pressing **Enter** or selecting the

icon from either of these fields updates the **Name** or **Description**.

Viewing and Searching for Group Members

Default Group Member List

When selecting a group, the first 500 members appear on the **Users** tab. The page controls allow you to scroll through the user list. To find other or specific users of the group, use the **User Search** option.

Searching for Group Members

To find specific users, use the **User Search** option. The **User Search** follows the same search capabilities as described on the **User Search** page.

Adding and Removing Group Users

Selecting a group enables options to add new users to the group. A user search screen can locate users to add. You can select multiple users to add from the search results screen.

Each user who is shown in the members list displays an icon at the end of the row. To remove a user from a group, select the icon to remove that user.

Deleting a Group

Selecting a group enables a **Delete Group** option at the top of the screen. You can use this option to delete the group.

Applications and Application Groups

Applications tab within the Symantec Identity Security Platform Admin Console encompasses Applications, and Application Groups.

1. On the Admin Console navigation panel, click .
   The Applications menu is displayed.
2. Select the Applications or Application Group you want to navigate to, and click the respective page for detailed information.
   - Applications
   - Application Groups

Applications

Learn how to create an application in the Symantec Identity Security Platform using the Admin Console UI.

> **NOTE**
> The admin console experience is tailored to match the current privileges of each administrator. Therefore, to access and modify information within the console, you must possess the necessary privileges. You are currently viewing this page because you hold the `urn:iam:t.apps` privilege.

The icon on the left panel of the Admin Console lets you add and manage applications and application groups in the IDSP.

- ***Applications***- These applications can be integrated with IDSP to leverage its modern authentication services, such as multi-factor authentication flows in both mobile and web channels.
- ***Application Groups*** - Application groups offer enhanced scalability and streamlined management for collections of related applications that share common policies. This approach eliminates the need to define separate policies for individual applications, significantly reducing maintenance overhead. For example, rather than creating distinct policy rules for App_A, App_B, and App_C, a single rule can be applied to an App_Group_Finance that encompasses all three. This not only simplifies policy management but also enables more granular control over access across different application sets.

The Applications page displays the "DefaultTenantClient" which is the default application provided by IDSP.

- "***DefaultTenantClient***" -This client serves as the gateway for executing tenant-level operations within the solution (IDSP). To know more about this client, see Configuring Administrative Access.

This topic includes the following sections:

- Adding an Application
- Deleting an Application

Adding an Application

This section describes the steps that are required to add an application to the solution.

1. Log in to the Admin Console.
2. On the navigation panel, click **Applications**.
   The **Applications** page is displayed.
3. To search for any existing application, use the search bar on the top of the page. Use this option to locate a specific configured resource by name or description, or apply custom filters to refine your search. You can apply single or multiple filters and use the **"Matching Filters"** option to return results based on your criteria, choosing "All" to return resources matching every condition or "Any" to find those meeting at least one.
   - **Override max search results limit**: Tenant administrators can use the "Override max search results limit" checkbox to display all available Admin Resources without any restrictions. When this checkbox is selected, the system bypasses the default maximum search results limit, which normally limits the display of Admin Resources to a maximum of 500 results, controlled through the configurable tenant setting, "maxAdminResourcesReturnedByGetAll". See Managing Settings.
4. On the Add Application page, select an application type.
   The available application options are:
   - **OAuth/OpenID Web App** - Confidential web server application that runs on a secure server.
   - **Native/Mobile OAuth App** - Public app that runs on a native desktop or mobile device.
   - **OAuth Service App** - Service-provider or machine-to-machine application (no user interaction).
   - **OAuth/OpenID Single Page App** - JavaScript-based single-page app that runs entirely in the browser.
   - **SAML App** - Application that leverages the SAML 2.0 protocol to enable secure and standardized authentication, supporting single sign-on and seamless access to resources within a federated environment.
   - **URL Launcher App** - Application that is triggered or opened through a specific URL or web address. Clicking a particular URL initiates the launching or opening of the associated application.
5. Select the type of application that you want to create, and click **Next**.
   The **Add Web Application** page is displayed.
6. Enter the `name` of the application in the **Name** filed. (modifiable).
7. Optionally, provide a `description` for the application in the **Description** field (modifiable).
8. Click **Add**.
   The application is successfully created, and the application page is displayed with the following tabs:

   NOTE
   The information within each of these tags is covered in the links that are specified at the end of this page.

   - General - Update general information about the application in this tab which includes **Custom URIs**, **Metadata**, **Additional Configurations**, **Identity Stores to be used** etc. Detailed information on this tab is available in the General tab. Click the blue link that is associated with the General tab.
   - OAuth/OpenID - To create an OAuth/OpenID protocol-based application, update the required fields in this tab, such as **Allowed Grant Type**, **Allowed Operations**, **Allowed Scopes** etc. Detailed information on this tab is available in the OAUTH/OpenID tab. Click the blue link that is associated with the OAUTH/OpenID tab.
   - SAML - To create a SAML application, update the required fields in this tab. Detailed information on this tab is available in the SAML tab. Click the blue link that is associated with the SAML tab.
   - Launcher App -

The launcher app functions by directing users to the URL specified in its configuration, similar to a browser bookmark. The launcher app serves as a quick access shortcut for all applications, whether they use protocol flows like SAML/OIDC or whether they are simple web apps without any protocol integration.

– **Policies** - To assign a policy to your application, select this tab and click **Assign**.
The **Assign Policies** page is displayed.
  a. Click **Select a Policy**.
  All the existing policies are displayed.
  b. Select the desired policy from the list displayed, and click **Done**.
  The assigned policy is displayed for the application.

NOTE

- If a specific policy is assigned to all applications, it is automatically displayed in the Policies tab and cannot be deleted.
- To delete an assigned policy, click the **X** button that is located to the right of the policy.

– **Modified by** - To view the user who created the application along with the date and time of its creation, select this tab.
– **Managers** - To view and modify the entities that can currently manage this application, select this tab. From the vertical ellipses (**...**) next to each row, perform one of the following actions for that entity:
  - Select **Assign** to assign entities that can manage this application
  - Select **Revoke** to revoke the assignment of entities that can manage this application
– **Authorization** - To view and modify the resource authorizations that have been assigned to this application, select this tab. This window provides the following sections:
  - **Tags**: Allows you to add tags to this application. The tag is a name-value pair that you use to assign an attribute to this resource. Once a resource is tagged with a criteria, you can include that criteria in a constraint statement for any policy that affects that resource.
  The format must be `name=value`. However, there are no restrictions on the name or value that you assign.
  - **Administrative Roles**: Displays a list of administrative roles that are assigned to this application.
  - **Resources Managed By <Application Name>**: Displays a list of the resources that this application can manage.
  - **Resources Delegated By <Application Name>**: Displays a list of the resources that this application has delegated to other entities.
  Modified By: Displays the user name and the date and time stamp for the most recent creation or modification of the application.
– **Groups**: You have the option to either assign this application to a pre-existing app group, or create a completely new app group directly from this location.

Deleting an Application

Once you have configured the application, it appears on the **Applications** page in a tabular format. The ellipses (**...**) at the end of each row lets you delete, disable, copy client Id and copy client secret of the application.

There are two ways of deleting an application: a standard deletion that requires all references to be removed beforehand or forceful deletion. The administrator has the discretion to choose which option to use. In general, if a resource is not in-use for longer time, it gets cleaned up by the standard delete procedure.

Deleting an Application Referenced by Other IDSP Objects

You can delete an application directly if it is not being used by other resources. A reconfirmation is requested before deleting an application. Once confirmed, the application is deleted. However, if one or more resource types have referenced this application, a message is displayed stating that the application cannot be deleted because it is still in use. The message looks like the following:

```
Application associated with one or more policies cannot be deleted.
```

```
[ txnId : ebe3ef4d-1c50-42ef-8989-f52bd1c10354 ]
```

In this case, you need to invoke the helper API, "**/ReferencingObjectsHelper**" which retrieves the list of dependent objects in case of a conflict. You need to review and manually remove the reference from each object before deleting the application.

For more information, see ReferencingObjectsHelper API.

For example, the application might be used in configurations such as Authentication (AuthN) policy, Authorization (AuthZ) Policy or Discovery Policy. **The delete is only allowed once all such references are removed.**

Forceful Deletion of Application

When attempting to delete an application that is referenced by multiple policies, a warning dialog titled "Deleting this Application will remove all associations" appears. This dialog informs you about the potential impact of your action and displays a list of associated resources in a detailed table. You can then choose "Cancel" to abort the deletion or "OK" to proceed. Clicking "OK" deletes the application and automatically removes all its listed associations from policies. This process ensures full awareness of the consequences before finalizing the deletion. See Resource Deletion in IDSP using REST API.

Reference Links:

- For more information on Creating Application using the API flow, see here.

General

Allows you to update general information about the application which includes **Custom URIs**, **Metadata**, **Additional Configurations**, **Identity Stores to be used** etc.

In the **General** tab, update the following attributes:

- **Identity Provider Discovery Enabled**: Allows you to specify whether identity provider discovery should be enabled for the application. When enabled, the IDP Discovery policy is evaluated. This value overrides the tenant setting, "idpDiscoveryEnabled" configuration.
  Select one of the following options from the drop-down list:
  - **Yes**: Enables IDP discovery policy evaluation.
  - **No**: Disables IDP discovery policy evaluation.
  - **Default**: Uses the configuration set in the tenant setting.
- **Require Post Authentication Access Control**: Enables the enforcement of access controls after successful user authentication. When enabled, this setting ensures that additional checks or controls are applied to user access following the authentication process to further secure access to resources or functionalities within the application. You need to create an access rule in the authentication policy to evaluate this rule. For information on Access rule in Authentication Policy, see Post Authenticate Access.
  Select one of the following options from the drop-down list:
  - **Yes**: enables identity provider discovery.
  - **No**: disables identity provider discovery.
  - **Default**: uses the configuration set in the tenant setting
- **Skip Multi-Factor Authentication During Registration**: Controls whether multi-factor authentication (MFA) should be bypassed during the inline credential registration process for users. When enabled, users are not required to undergo MFA during Sign-in.
- **Zero Footprint**:
  - When enabled, Symantec Identity Security Platform does not validate the third-party identity against the local Identity Store configuration, rather mandates its validation on the ID Token Hint received from the third party (for example-SiteMinder)
    Ensure that the ***ID Token Hint*** contains the following mandatory attributes and their respective values:

- email- The email address associated with the third-party configuration.
- phone_number- The phone number of the user configured in the third party.
- user_universalID- The login ID of the third-party user.
- auto_inline_enrollment-
  - When set to true, IDSP auto enrolls the credentials based on the value in the ID Token Hint
  - When set to false, IDSP disregards the credentials that are sent in the ID Token Hint.

– When disabled, IDSP validates the third-party identity against the local Identity Store configuration.

To know more about ZFP, see here.

Configuring Custom URLs for the Application

- **Custom Login URL**: Define the URL for the IDSP UI or a custom login URL using the IDSP API for login controls. If the attribute value is empty, this value defaults to the tenant configuration.
- **Delegate Authentication**: Enable this attribute when the configured application is built using custom APIs and specifies a **Custom Login URL** value in the previous attribute. If IDSP APIs are used to build this application, keep the default configuration (which is disabled).
- **Auto Post to Custom Login URL**: Enable this option when the configured **Custom Login URL** application is designed to accept data from IDSP through an HTTP POST request.
- **Home URL**: When an Identity Provider initiates a global logout, it uses this attribute as the post-logout URL for redirection after completing the global logout process.
- **Logout URL**: An application maintains a logout URL within its configuration, using during the discovery of an application-specific logout URL. In cases where the application does not maintain its own logout URL, this attribute serves as an alternative for discovering the logout URL.
- **Post Logout URL**: An application initiating a logout process can either specify a URL for redirection or have an application-specific logout URL configured.
  In scenarios where the application initiating the logout process neither has a configured logout URL nor specifies one in the logout request, this attribute serves as the post-logout URL. If this attribute is not provided in such cases, the homeUrl setting is used instead.
- **Error URL**: Allows specifying a url for an error page to be displayed during authentication failures. This value overrides the tenant error URL; however, if no value is specified, it defaults to the URL set in the tenant settings.
  The following redirection behavior occurs for error handling within an application:
  a. If an app-level Error URL is not provided, or it is provided as "protocol" (OIDC/SAML), the error redirection occurs via the protocol.
  b. If the application intends to redirect to a specific Error URL, one of the following conditions must be met:
     a) The application explicitly specifies its own Error URL.
     b) If you do not want to specify an app specific URL, you can instruct the app to use a global error page. This is done by including the URL info in the application metadata, **urn:app:useGlobalErrorUrl=true**. See **Metadata attribute** in the next section.
     In this case:
     - If a tenant-level Error URL is provided, the redirection occurs to that URL.
     - If no tenant-level Error URL is provided, the redirection takes place to a built-in "/ui/v1/signin/error" URL.
     - If the application intends to redirect to a built-in Error URL, specify Custom URL as "${system.issuerUrl}/ui/v1/signin/error"

Metadata

Configure the metadata name/value pairs associated with this application, providing additional descriptive information and context.

1. Click **+Add**.
   The **Name** and **Value** fields appear.

2. Add custom metadata in the Name and Value field or select from the predefined metadata properties that IDSP provides. To know the predefined list of metadata values available, see Application Metadata properties.

Additional Configuration

Configure these additional attributes, if necessary.

- **Skew Time**: Specify the number of seconds that must be added to the current time to account for the difference in the system clocks when validating the client assertion/ ID Token hint expiration.
- **Password Authoritative Source**: This property identifies the authoritative source for password verification.
  Select one of the following options from the drop-down list:
  – local - indicates that password is stored in IDSP account store (T_USER) and password verification happens against the IDSP account store.
  – remote (default) - indicates that the password is verified against the remote identity store (ldap)
- **Custom Claim Provider**: The "claims" attribute lets you configure custom claims for an OIDC/SAML application. IDSP includes these custom claims into the Identity token, Access token, UserInfo, and SAML assertion on behalf of this application as a Relying Party.

Identity Store to be used

Configure the identity store to be used.

Select one of the following identity stores options:

- **Default Identity Stores**: Specifies the id store to use for user disambiguation, and by default, for user authentication. This is the idstore that you have configured in the tenant setting, "idStoreToBeUsed" after the deployment of IDSP.
- **Application-specific Identity Stores**: If you do not want to use the default Identity Store, and want to specify the identity store at the application level, you can select the identity stores from the **Available Identity Stores** field and click **>** to move it to the **Assigned Identity Stores**.

Protection against Man-in-the-Middle(MITM) attack

Specifies whether a factor request is validated at an application level using the flowstate attribute in an authentication response and Trusted Device Identifier (TDI) in an HTTP cookie, to mitigate a man-in-the-middle (MITM) attack.

- **None**: Defines that no level of mitmProtectionLevel is applied to the application.
- **Tenant level Settings**: Defines that the value of mitmProtectionLevel in an application must be determined by its corresponding values that are defined in the tenant settings.
  If mitmProtectionLevel is set to "tenant" in an application but the value is set to "none" in a tenant, the feature is disabled.
- **Application level settings**: Defines that the value of mitmProtectionLevel in an application must be configured in the application.
  Select one of the options:
  - **Consider User's device during authentication flow** - Considers the device of the user during the authentication process to detect and mitigate potential MITM attacks.
  - **Consider User's IP Address during authentication flow** - Considers the IP address of the user during the authentication process to detect and prevent potential MITM attacks.

Reference Links:

- For more information on Creating Application using the API flow, see here.

OAuth/OpenID

Configure the OAuth/OpenID application attributes in the Admin Console.

In the **OAuth/OpenID** tab, select **enabled**.
The OAUTH/OpenID attributes are displayed.

Integration Setup: If you intend to integrate Symantec Identity Security Platform with your service provider, the information in the **Integration Setup** tab is useful.

- Click **View Setup**.
  The Integration Setup page is displayed. See details here.

Configure the following OAUTH/OpenID attributes:

- **Client ID**: The client ID serves as a unique identifier assigned to the application by the IDSP. It is used to identify and authenticate the application when interacting with the authorization server during the authentication process.
  - Enter the client Id that was used during the generation of the Client Access Token in the Authorization header.
  - (Optional) To copy the Client ID, click **Copy**.
  - (Optional) To regenerate the Client ID, select the **Regenerate Client Secret** check box.
- **Client Secret**: Client Secret is a secure password provided by the IDSP. This is known only to the application and the authorization server. It serves as a means of authenticating the application to the authorization server during the authentication process.
  - Enter the client secret used during the generation of the Client Access Token in the Authorization header.
  - (Optional) To copy the Client ID, click **Copy**.
  - (Optional) To regenerate the Client ID, select the **Regenerate Client Secret** check box.
- **Redirect URI**: Specify the list of allowed callback URLs where the authorization server sends the user once the app has been successfully authorized and granted an authorization code or access token. For example, https://www.example.com.
  - To add multiple redirect URIs, click **Add**.
    An URI field is displayed where you can enter another redirect URI.
  - To delete a redirect URI, click **X** button located on the right of the attribute.
- **Device Code Flow URL** : (Optional) Specify the URL endpoint that facilitates the initiation and processing of the device authorization flow for this application. If this URL is not configured, then the default `IDSP verification_uri https://${SSP_FQDN}/tenant/ui/v1/device` is used.

  NOTE

  - To know the value of ${SSP_FQDN} environment variable, see Configuring IDSP Environment Variables.
  - The default value for `<tenantName>` is "default".
  - Replace the text {{text}} including {{}} characters with appropriate values, in all the endpoints shown in the above table.
    For example:
    `https://{{host}}/{{tenantName}}/ui/v1/device`
    should look similar to:
    `https://example.com/default/ui/v1/device`
  - To know more about the scopes that are required to secure these APIs, refer to the Rest APIs and Associated Privileges section.
- **Client Type**: Select the type of client application that is going to interact with the authorization server. This attribute helps the authorization server determine the appropriate security measures and authentication mechanisms to apply during the authentication process.
  - **Confidential**

- Clients that can be trusted to securely store their credentials such as client secrets. For example, web apps running on Web servers or App servers.
- Clients that are trusted to issue user assertions such as an ID Token Hint when initiating OpenID Connect interactions, or when submitting additional information about the user in certain API calls (such as when specifying a Users IP Address in user authentication API calls).

  NOTE
  A mobile app running on a user's mobile device can often be considered to be confidential as it belongs to a single user and has security controls to protect the device before the device can be used.
- Confidential clients can submit their credentials using Basic Auth Authorization or POST body.

– **Trusted**
- Clients that are trusted to issue user assertions such as an ID Token Hint when initiating OpenID Connect interactions, or when submitting additional information about the user in certain API calls (such as when specifying a User's IP Address in user authentication API calls).

  NOTE
  Trusted clients are also considered confidential clients.
- If you select **Trusted** as the client type, select the certificate alias from the **Client Aseertion Validation Certificate** drop-down list.
  - Click **Select Certificate** to search for a certificate using an alias.
  - You can use the filter to search for a certificate using filters, such as Attributes, Operator, Value, Match Filters with **All** or **Any**.
  - If a certificate is not available, click on **Import** to upload a new certificate.

– **Public**
- Clients that cannot securely store their credentials. For example, Javascript based applications, Single Page Applications (SPA), or mobile clients.
- Public clients can submit their credentials using Basic Auth Authorization or POST body.

To know more about Client Types, see Client Types.

Allowed Grant Types

Select the **Grant Type** for the application from the following options:

- **Client Credentials**: After the user authorizes an action on the application, the application sends the client ID and client secret to IDSP. The client ID and client secret are validated using the client ID and client secret registered with IDSP. Once the validation is successful, IDSP returns an Access Token that grants access for the application.
- **JWT Assertion**: The application uses a JSON Web Token (jwt) to authenticate. The token is validated against the token registered with IDSP. IDSP returns as Access Token that grants access to the application.
- **Authorization Code**: After the user authorizes an action on the application, the application receives an authorization code. The application presents the authorization code to IDSP, where it is exchanged for an Access Token. IDSP returns the token that grants access for the application. The application request for the token is protected by the client secret, and the Access Token is never seen by the user.
- **Implicit**: After the user authorizes an action on the application, the application receives an Access Token directly from IDSP. The Access Token then grants access to the application. The application request for the Access Token is protected by the client secret.
- **Refresh Token**: Refresh Tokens are credentials used to re-generate an Access Token without re-authenticating, when the current access token becomes invalid or expires. When using the Refresh Token, you may request for the

same scopes that were requested during the initial authentication or narrow down the scopes depending on your requirement.

- **Device Code**: The Device Code grant type allows devices with limited input capabilities to obtain an access token by displaying a code to the user, who then enters it into a separate device to complete authentication.
- **STS Confidential**: A client can use the sts-confidential token grant to swap the IT encrypted with the IDSP's tenant's public key for the IT encrypted by the public key of the intended target as configure by application's `itEncryptionTarget` property, ensuring secure transmission of sensitive information.

The following table shows the industry-recommended use of various OAuth grant types. The (✓) shows the recommended grand type that is suitable for the indicated apps. The grant types are linked to their respective sections, click each grant type for more information.

For detailed information on Grant Types, click the respective links on the table.

Grant Type	Protocols	Client Type	Single Page Apps	Native Apps	Microservices	Web App	Typical IoT Device
Client credentials	OAuth 2	public, confidential, trusted		✓ (Public Clients)	✓	✓	
Authorization code	OAuth 2.0 and OIDC	public, confidential, trusted				✓	
Authorization code With PKCE	OAuth 2.0 and OIDC	public, confidential, trusted	✓	✓			
Implicit (not recommended)	OAuth 2.0 and OIDC	public	✓				
Refresh Token	OAuth 2.0 and OIDC	public, confidential, trusted		✓	✓	✓	
JWT-Bearer	OAuth 2.0	public, confidential, trusted		✓	✓	✓	
STS-Confidential	OAuth 2.0	public, confidential, trusted					
Device Code	OAuth 2.0	public, confidential, trusted					✓

Allowed Operations

When configuring application clients, configure the operations that the application can perform.

Select the options relevant to the application from the following list to control the client behavior:

- **Client can inspect access tokens (Introspect)** - The Introspect Endpoint is used to validate the received Access Token or Refresh Token, and view the scopes associated with the token. OAuth introspect endpoint only works when 'introspect' is configured here, else the introspect endpoint returns an error.
- **Client can act onbehalfof a user** - When this option is configured, the authorized scopes are computed by the authorization server based on user's privileges only, instead of the intersection of user and client privileges.
- **Allow adding user groups in the Identity Token** - When this option is configured, the application is allowed to add user groups in the Identity Token when the custom scope, "urn:iam:it:groups" is requested.
- **Allow adding user groups in the Access Token** - When this option is configured, the application is allowed to add user groups in the Access Token when the custom scope, "urn:iam:it:groups" is requested.
- **Client can request Identity Token in the Refresh Token and/or JWT Assertion flows** - When this option is configured, the identity token is returned along with the Access Token in the Refresh Token grant or/and JWT assertion flows.
- **Allow client to rotate it's secret** - When this option is configured, client secret rotation can be performed for your own clients and also on behalf of other clients (applications) if the admin has been granted the following scope: **"urn:iam:t.clientseccretrotator"**. See Client Secret Rotation for more information.

Allowed OpenID Scopes

OpenID Connect (OIDC) scopes are a set of user attributes like email, phone, etc. that are used by an application during authentication to authorize access to the user. Each scope returns a set of user attributes, which are called claims. The scopes an application should request depend on which user attributes the application needs. Once the user authorizes the requested scopes, the claims are returned in the ID Token.

IDSPsupports the following OpenID scopes:

- **Profile (profile)** - This scope in OpenID Connect allows client applications to request access to basic profile information about the authenticated user, such as their name, preferred username, profile picture, etc. It enables the client application to personalize the user experience and tailor content based on the user's profile data.
- **Email (email)** - This scope allows the client application to access the user's email address associated with their account. It enables applications to communicate with users through the configured email to send all the required mail notifications.
- **Address (address)** - This scope allows the client application to access the user's postal address in the ID store, including street address, city, state, postal code, and country. It is useful for applications that require shipping or location-based services.
- **Phone (phone)** - This scope allows the client application to access the user's phone number associated with their account. It enables applications to communicate with users through the configured phone number to send all the required sms notifications or phone calls.
- **Offline Access (offline-access)** - This scope is provided to client applications, enabling them to request a Refresh Token. To utilize this, include the **"offline_access"** scope when making requests to the `/authorize` or `/token` endpoints to obtain a refresh token.
- **User Groups (groups)** - User groups are logical collections of users who share common characteristics, permissions, or roles within a system or application. They are used to simplify access control management by allowing administrators to grant or revoke permissions to multiple users simultaneously based on their group membership.

Allowed Custom Scopes

Custom scopes in OIDC (OpenID Connect) scopes refer to scopes that you can define for the application to request beyond the standard set of predefined scopes like "openid", "profile", "email", etc. Custom scopes allow for fine-grained control over the access permissions granted to the client application. Client applications must request these custom scopes when initiating the authentication process, and the authorization server will grant access only to the requested scopes if authorized by the user. Custom scopes enhance security by ensuring that client applications only receive access to the resources they explicitly request and are authorized to access.

IDSP supports the following Custom scopes:

- **Application can request secondary audiences in Access Token (urn:iam:at:appaud)** - With this custom scope, the client application can request secondary audiences to be included in the access token that it receives. By default, an access token is issued with a primary audience, representing the resource server directly associated with the requested scopes.
- **Application can request user's groups in Identity Token (urn:iam:it:groups)** - This custom scope enables a client application, during the authentication process, to request the inclusion of information about the user's groups in the identity token it receives. By including the user's groups in the identity token, the client application gains insight into the user's role or permissions within the system.
- **Application can request user's groups in Access Token (urn:iam:at:groups)** - This custom scope enables a client application, during the authentication process, to request the inclusion of information about the user's groups in the access token it receives. This information can be used by the application to make access control decisions, customize user experiences based on group memberships, or enforce authorization policies.
- **Application can use scope 'urn:iam:sso:token' to request a Session Token via JWT-BEARER grant**
- **Application can allow a user to initiate a password reset or replace a forgotten password (urn:iam:lcm:resetpassword)**
- **Application can request a short-lived Identity Token in the Refresh Token grant or JWT Assertion flows (urn:iam:sso:idprop)** - This custom scope enables a client application, during the authorization process, to request the issuance of a short-lived identity token alongside the refresh token in the Refresh Token grant or JWT Assertion flows. The ability to request a short-lived identity token enhances security by limiting the exposure of sensitive user information and ensures that tokens remain valid only or appropriate durations.
- **SCIM Core user profile attributes (urn:iam:scim:core)** - Allows the client application to request access to the core user profile attributes defined in the System for Cross-domain Identity Management (SCIM) protocol.
- **SCIM Enterprise user profile attributes (urn:iam:scim:enterprise)** - Allows the client application to request access to the enterprise user profile attributes defined in the System for Cross-domain Identity Management (SCIM) protocol.
- **Custom user profile attributes (urn:iam:scim:custom)** - Allows the client application to request access to the custom user profile attributes defined in the System for Cross-domain Identity Management (SCIM) protocol.
- **Operational user profile attributes (urn:iam:scim:ssp)** - Allows the client application to request access to the operational user profile attributes defined in the System for Cross-domain Identity Management (SCIM) protocol.
- **Credentials custom attribute (urn:iam:scim:credentials)** - Allows the client application to request access to the credential custom attributes defined in the System for Cross-domain Identity Management (SCIM) protocol.

Custom Claims

Custom claims in OIDC (OpenID Connect) define additional information about the authenticated user beyond the standard claims provided by the identity provider. These claims can be included in both the identity token and the access token during the OIDC flow.

The **Custom Claims** attribute allows to configure custom OIDC claims for this client application.

1. Click **+Add**.
   The **Name** and **Value** fields appear.
2. Enter custom claims in the Name and Value field.
3. Select whether to include these claims in the **Access Token** or **ID Token**, or both.
   - **Access Token** -
     - Custom claims in the access token provide additional context or permissions associated with the client application's access to protected resources.
     - These claims are typically used to convey authorization-related information, such as user roles, scopes, or other access permissions.
   - **ID Token** -
     - Custom claims in the identity token provide additional details about the authenticated user.
     - These claims are typically included in the identity token to convey specific user attributes or permissions to the client application.

See Custom Token Claims topic for examples.

ID Token Configuration

Configure this attribute in the client application to define how the client application handles the ID token. The ID Token is a JWT token which contains a JSON object with user information.

ID Token Encryption Target - Select one of the following options:

- **None** - Select this option if you do not want to encrypt the ID token.
- **Application** - Select this option if you intend to encrypt the ID token for this client application.
  - **ID Token Encryption Certificate** - If `Application` is selected, select the certificate used to encrypt the token. If the correct certificate is not listed in the drop-down, add it from the Certificates page. See Certificates.
- **Tenant** - Select this option if you intend to encrypt all the tokens for this specific tenant.
- **Skip Email For Identity Token** - Select this option if you do not intent to include the email attribute in the ID token.
- **Skip Issuer Audience For Identity Token** - Select this option if you do not intent to include the issuer audience attribute in the ID token.
- **User Token Subject Attribute Mapping Name** - This is the name of the logical id store attribute whose physical id store attribute's value is to be placed in the "sub" token claims for IT and AT issued to this application. This logical attribute has to appear as one of the mappings in the "attributeMapping" property of the id store config resource.

User Info Configuration

Configure this attribute in the client application to define how the user information (first name, family name and user name) should be retrieved and managed by IDSP.

- **User Info Endpoint Response Format** - Select the format in which the User Info object should be returned; a `JWT` token, a `Plain JSON` Object, or a `Base 64 JSON` object.
- **User Info Encryption Certificate** - (optional) If `JWT` was selected for the response format, select the certificate used to encrypt the JWT token.
  - Click **Select Certificate** to search for a certificate using an alias.
  - You can use the filter to search for a certificate using filters, such as Attributes, Operator, Value, Match Filters with **All** or **Any**.
  - If a certificate is not available, click on **Import** to upload a new certificate.

Secondary Audience

A secondary audience refers to an additional entity or application that is granted access to the user's identity information during the authentication process. Set another entity (also known as "aud" claims) to whom you intend to pass the identity token to receive and consume.

1. Click **Add** to enter a secondary audience value. This application property supports Dynamic Substitution. (This support is added from IDSP v3.4.4).

Supported JOSE Header Parameters

The following JOSE Header parameters are supported in IDSP.

Select the type of Jose Header for the client application. If this option is left empty, `x5t` and `x5t#S256` are added in the JOSE Header of JWS and JWE tokens by default.

- **X.509 Certificate SHA1 Thumbprint** - the X.509 Certificate SHA1 Thumbprint refers to the cryptographic hash value computed from the X.509 certificate used for securing communication between the OIDC provider and the application.

This thumbprint serves as a unique identifier for the certificate and is utilized for verifying the authenticity and integrity of the certificate during the OIDC authentication process.

- **X.509 Certificate SHA256 Thumbprint** - The X.509 Certificate SHA256 Thumbprint, also known as the SHA-256 fingerprint, is a cryptographic hash value computed from the X.509 certificate. The SHA256 Thumbprint provides a concise representation of the certificate, facilitating its identification and verification in secure communication protocols.
- **X.509 Certificate Chain** - The X.509 Certificate Chain refers to a sequence of X.509 digital certificates that are linked together in a hierarchical manner. Each certificate in the chain is issued by a Certificate Authority (CA) and is used to verify the authenticity and integrity of subsequent certificates in the chain.

1. Click **Save Application**.
   The application is saved to the Application dashboard.
2. To navigate to the application dashboard, click **<** Applications on the top left of the applications page.

Reference Links:

- For more information on Creating Application using the API flow, see here.

OAuth Resource Server

Learn about configuring a resource Server using the Admin Console UI.

A Resource Server is a specific server that allows an application to model and manage its own access control mechanism using the OAuth2 security model.

- **Definition**: The Resource Server (RS) is an app facet that models app's access control accessible and manageable through the OAuth2-based security model. It is a component of an application that specifically handles and protects its resources.
- **Role in OAuth2**: Its primary role is to accept and respond to protected resource requests made by client applications. These requests are typically accompanied by an access token, which the Resource Server validates to ensure the client (and often the user on whose behalf the client is acting) has the necessary authorization to access the requested resources.
- **Privileges and Scopes**: The Resource Server defines what privileges (also called scopes) are available for its resources. These scopes are aggregated into Authorization Policies, and clients request specific scopes in their OAuth2 authorization and token requests. The access token issued by the Authorization Server contains these granted "scope" claims, which the Resource Server then uses to enforce access control.
- **Audience**: A Resource Server has a Primary Audience which is a unique identifier. This audience is included in the access token, allowing the Resource Server to verify that the token is indeed intended for it.

Access the Resource Server App

To configure an application as an OAuth Resource Server, navigate to the Applications section in the Admin Console. Select the "**Service App as an OAuth Resource Server application**", and then click the **OAuth Resource Server** tab.

Key Configuration Elements

- **Enabled**: This checkbox activates or deactivates the Resource Server functionality for the application. When enabled, it transforms the application into a Resource Server, allowing it to define and protect its own resources using OAuth2.
- **Primary Audience**: This field defines a unique identifier for the Resource Server. It is a critical component as it establishes the intended recipient of access tokens issued for this Resource Server.
  Enter an absolute URI, that uniquely identifies this Resource Server. This value is paramount for token validation; the Resource Server verifies that any incoming access token has its Primary Audience listed in the token's audience (aud)

claim. This value must be unique across all applications within the tenant to ensure proper identification and prevent conflicts.

- **Privileges**: This section is where administrators define the specific permissions, also known as scopes, that client applications can request to access the resources protected by this Resource Server. The privileges exposed by the Resource Server get aggregated into Authorization Policies for assignment to users and application clients. Clients accessing the Resource Server request specific privileges using the 'scope' request parameter and get back access tokens containing 'scope' claim with granted privileges.
  - **Consent**: This toggle controls whether an end-user must explicitly grant consent for a specific privilege (scope) when a client application requests it. If set to ON, the user is prompted to approve this permission.
  - **Name**: This is the name for the privilege (scope). It is the identifier that client applications use in their OAuth2 requests.
  - **Description**: This provides a brief, internal description of the privilege, primarily for administrative understanding and management within the Admin Console.
  - **Consent Label**: This is the user-friendly text that is displayed to end-users on the consent page when a client application requests this privilege.
  - **Consent Description**: This optional field allows for a more detailed explanation of what the privilege entails. It provides additional context to the end-user during the consent process, helping them make an informed decision.
  - **+ Add** : Click this button to add a new privilege (scope) row to the Resource Server configuration, allowing you to define additional permissions.
  - **Consent URL**: This field allows you to configure a custom URL that links to your resource server.
    - **Usage Policy URL**: This field allows you to configure a custom URL that links to your resource server's usage policy. This URL typically points to legal documents such as a privacy policy, terms and conditions, or a data usage agreement that users should review before granting consent to an application.

Managing Application Roles for a Resource Server

Application Roles provide a structured way to manage access to an application's protected resources. These roles are created specifically to group various "Application Privileges" (the granular scopes defined by the Resource Server facet of an application) for easier assignment and management.

- **Aggregation of Privileges**: An Application Role is a collection of resource-specific privileges (scopes).
- **Backed by AZ Policies**: These Application Roles are modeled as app-specific Authorization (AuthZ) Policies. Once AuthZ Policy is mapped to a Resource Server app, such AZ Policy becomes a resource-specific App Role.
- **Granting to Principals**: Administrators can directly or indirectly grant these Application Roles to various principals, including users, user groups, application clients, or application client groups. This means that instead of granting individual privileges, you can grant a role that encompasses a set of privileges.
- **Management (LCM)**: The Resource Server is intended to support a full "App Roles LCM" (Lifecycle Management), including creating, deleting, and updating these roles.

Adding a new application role

NOTE

- The option to add an application role becomes available only after consent settings are configured and saved. If no consent is configured, this option remains unavailable.
- Only resource server applications can be assigned to application roles; normal applications are not supported.

1. Click the + Add Role button.
2. In the "Add New Application Role" dialog, define the following:

- **Name** - enter a unique name for the role (for example, HR Administrator).
- **Description** - Provide a brief description of what this role entails.
- **Only assign to clients** - Select this option if you want to assign the application role to only clients and not users.
- **Role can be delegated** - Select this option to delegate the role.

3. Click **Next**.
4. **Configure Role Conditions**. This is where you define criteria that must be met for this role to be applicable.
   - **Add Conditions**: Click the Add Conditions dropdown.
   - **Select Condition Types**: Choose one or more conditions from the list. The list includes IP Address, Country, Risk Score, Triggered Risk Rule, and Context.
     All conditions that you set must match before access is granted or restricted.
   - Configure Selected Conditions: For each selected condition, provide the necessary details (for example, specify an IP address range if "IP Address" is chosen)
   - Once all conditions are configured, click Next
5. **Define Role Results**: Select the result for this rule.
   - Grant Access Privilege: The role grants the specified privileges.
   - Revoke Access Privilege: The role revokes specified privileges.
   - Deny Access: The role denies access to the protected resource.
6. **Select Privileges**: This is where you associate specific privileges (scopes) with this Application Role.
   Perform either of the following:
   - Enter Privilege: Type the exact privilege name (scope) into the "Enter Privilege" field and click Add.
   - Select Privilege: Click the Select Privilege button to open a modal window displaying available privileges. Check the box next to each privilege you want to include in this role
   - Once all privileges are selected, click Next.
7. **Review Summary**: Review all the configurations for the new Application Role.
   - Conditions
   - Access
   - Privileges
   - After the final review, click Save.
8. **View the Created Application Role**: After saving, the new Application Role appears in the Application Roles table on the "OAuth Resource Server" tab. The table displays details such as NAME, DESCRIPTION, CONDITIONS, PRIVILEGES, MEMBERS, OPERATION, and STATUS.

Granting an application role

Once a application role is created, it can be assigned (granted) to users, groups or application clients.

Follow these steps to grant a role:

1. In the Application Roles table, either click on the MEMBERS count link or use the ... (ellipsis) button in the OPERATION column to open the "Grant Role" dialog.
   - If you choose the MEMBERS count link, then the Application Role Members screen is displayed where you can click the + Grant Role button to open the "Grant Role" dialog.
   - If you choose the ... (ellipsis) button, the "Grant " dialog is displayed.
2. **Configure Grant Role**: The 'Grant Role' dialog opens, showing the specific role name you chose to grant.
3. **Select Principal Type**: Choose the type of entity to whom you are granting the role from the following options:

   NOTE
   While creating a new application role, if you have selected the **"Only assign to clients"** option then only the application and application group is displayed. If you unselect this property, then all the options, user, application, user group and application groups are displayed.

  - User Group
  - User
  - Application Group
  - Application
4. **Specify Identity Source**: In the "Select or Enter Identity Source" field, use the dropdown or type to find and select the identity source where the selected principal type exists.
5. **Confirm Grant**: Click the Grant button to assign the role to the selected principal. After granting the role, automatically the role is saved.
6. **View the granted Application Role**: The granted users and clients appear in the Application Roles table on the "OAuth Resource Server" tab. The table displays details such as NAME, DESCRIPTION, CONDITIONS, PRIVILEGES, MEMBERS, OPERATION, and STATUS.
7. **(Optional) Deleting an Application role** : The ... (ellipsis) button next to the Application Role lets you delete the role.
8. **(Optional) Deleting Members from the Application role**: Click the **Clients** or **Users** link in the Application Role table, the Application Role Mombers dialog opens. You can select the source that you want to delete, and click the ... (ellipsis) button and click Revoke.

Reference Links

- Creating a Resource Server using REST API
- Resource Server Application Flow (Including OIDC Flow)
- Signin UI Consent for Resource Server Applications

SAML

Configure the SAML application attributes in the Admin Console.

In the SAML tab, select **Enabled**.
The SAML attributes are displayed.

Integration Setup: If you intend to integrate Symantec Identity Security Platform with your service provider, the information in the Integration Setup tab is useful.
Perform the following steps here:

- Click **View Setup**.
  The Integration Setup page is displayed. See details here.

  NOTE

  SAML applications support the ${saml_idp_init} placeholder for the launcherUrl field, which automatically populates the URL with the "IDP init URL (preferred)" value during launcherUrl invocation, eliminating the need for manual URL entry.

  The "IDP init URL (preferred)" attribute value can be found in **Admin Console > SAML App > View Integration Setup**.

  To enable launcher app, see Launcher App.

Configure the following SAML attributes:

- **Entity ID**: Enter the entity ID provided by the Service Provider.

  - In the Service Provider Initiated flow, the "EntityId" should match the issuer in the authentication request.
  - In the IDP initiated flow, if the entityId is provided in the request parameter, it should match the configured "EntityID" to point to the configured application.
- **ACS URL**: Enter the "AssertionConsumerService" URL obtained from the service provider, where the SAML response is sent to.
- **Additional ACS URLs**: These URLs are used to receive SAML assertion responses from identity providers (IDPs) during the authentication process. In scenarios where multiple ACS URLs are required, such as in federated identity setups or multi-tenant applications, administrators can configure these additional ACS URLs to accommodate the diverse authentication needs of their system.
  a. Enter the **Index** value. This is the identifier for each ACS URL entry. It helps in organizing and distinguishing multiple ACS URLs within the configuration.
  b. Enter the ACS **URL** corresponding the index value.
- **NameID Format** : Enter the "NameID" format that is supported for the Subject.
  Supported formats are: emailAddress, unspecified, X509SubjectName
- **Verify Request Signature** : Enable this attribute to verify the signature of the SAML Authentication request.
  - **Verification Certificate**: Enable this option to select the certificate to verify the SAML request signature.
    - Click **Select Certificate** to search for a certificate using alias.
    - You can search for a certificate using filters, such as Attributes, Operator, Value, Match Filters with **All** or **Any**.
    - If certificate is not available, click on **Import** to upload a new certificate.
- **Encrypt Assertion** : Enable this to encrypt the SAML assertion sent by IDSP.
  - **Encryption Certificate**: Use this option to select a certificate that should be used to encrypt the SAML assertion.
    - Click **Select Certificate** to search for a certificate by its alias.
    - Use **filters** (such as attributes, operators, values) to narrow your search. You can match filters using **"All"** or **"Any"**.
    - If no certificate is available, click **Import** to upload a new certificate. Once the certificate is selected, the certificate details (**Alias**, **Description**, **Expiration**, **Updated**) are shown in a table.
  - **Encryption Algorithm**: Select the algorithm that should be used to encrypt the SAML assertion from the following list:
    - http://www.w3org/2001/04/xmlenc#aes256-cbc
    - http://www.w3.org/2001/04/xmlenc#aes128-cbc
    - http://www.w3.org/2001/04/xmlenc#aes192-cbc
    - http://www.w3.org/2009/xmlenc11#aes128-gcm
    - http://www.w3.org/2009/xmlenc11#aes192-gcm
    - http://www.w3.org/2009/xmlenc11#aes256-gcm
  - **Key Transport Algorithm**: Select one of the following algorithm to choose how the encryption key must be securely delivered:
    - http://www.w3.org/2001/04/xmlenc#rsa-1_5
    - http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p
- **Sign Response** : Enable this attribute to sign the SAML response sent by IDSP. This verifies the integrity and authenticity of the response.
  - **Sign Response Key Alias** : Use this option to sign the key alias.
    - Click **Select Key** to search for a key using alias.
    - Use **filters** (such as attributes, operators, values) to narrow your search. You can match filters using **"All"** or **"Any"**.
    - If no key is available, click **Add Key** to upload a new key. Once the key is selected, the key details (**Alias**, **Description**, **Updated**) are shown in a table.
  - **Sign Response Algorithm**: Select the algorithm to generate the signature from the following list:

- http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
- http://www.w3.org/2000/09/xmldsig#rsa-sha1
- http://www.w3.org/2001/04/xmldsig-more#rsa-sha512
- http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256

  NOTE
  When using the Elliptic Curve Digital Signature Algorithm (ECDSA) algorithm, ensure to use the key type as "EC Key" and this key must be generated using one of the following IDSP supported curves:

  - prime256v1
  - P-256
  - P-384
  - P-521
  - secp256r1
  - secp384r1
  - secp521r1

– **Sign Response Canonicalization Algorithm**: Select the Sign Response Canonicalization Algorithm type from the following list:
  - http://www.w3.org/2001/10/xml-exc-c14n#
  - http://www.w3.org/TR/2001/REC-xml-c14n-20010315
  - http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments
  - http://www.w3.org/2001/10/xml-exc-c14n#WithComments
  - http://www.w3.org/2006/12/xml-c14n11
  - http://www.w3.org/2006/12/xml-c14n11#WithComments

- **Sign Assertion** : Enable this attribute to sign only the SAML assertion sent by IDSP as part of the SAML response.
  – **Sign Assertion Key Alias** : Use this option to sign the key alias.
    - Click **Select Key** to search for a key using alias.
    - Use **filters** (such as attributes, operators, values) to narrow your search. You can match filters using **"All"** or **"Any"**.
    - If no key is available, click **Add Key** to upload a new key. Once the key is selected, the key details (**Alias**, **Description**, **Updated**) are shown in a table.
  – **Sign Assertion Algorithm**: Select the algorithm type to sign the assertion:
    - http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
    - http://www.w3.org/2000/09/xmldsig#rsa-sha1
    - http://www.w3.org/2001/04/xmldsig-more#rsa-sha512
    - http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256
  – **Sign Assertion Canonicalization Algorithm**: Select the canonicalization algorithm to be used during the signing of the SAML assertion:
    - http://www.w3.org/2001/10/xml-exc-c14n#
    - http://www.w3.org/TR/2001/REC-xml-c14n-20010315
    - http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments
    - http://www.w3.org/2001/10/xml-exc-c14n#WithComments
    - http://www.w3.org/2006/12/xml-c14n11
    - http://www.w3.org/2006/12/xml-c14n11#WithComments
- **Sign with Key Information** : Enable this attribute to sign the SAML signature with key information.
- **Relay state mapping** : Define the mapping between the relay state that is provided during the IDP Initiated flow and the corresponding relative URL of the application, where the flow concludes.
  a. Click **+Add**.
     The **Key (regex)** and **Value** fields appear.

   b. Enter the key using regular expressions (regex), and associated values that denote the corresponding relay states or actions that should be triggered based on the matched state.
- **Custom Claims** : The **Custom Claims** attribute allows configure custom SAML claims for an application. IDSPincludes these custom SAML claims in the Identity and/or Access tokens on behalf of this application as a Relying Party.
   a. Click **+Add**.
      The **Name** and **Value** fields appear.
   b. Enter custom claims in the Name and Value field.
- **User Assertion Subject Attribute Mapping Name**: Enter the name of the logical id store attribute whose physical id store attribute's value is to be placed in the "sub" token claims for IT and AT issued to this application. This logical attribute has to appear as one of the mappings in the "attributeMapping" property of the id store config resource.

1. Click **Save Application**.
   The application is saved to the Application dashboard.
2. To navigate to the application dashboard, click **<** Applications on the top left of the applications page.

Reference Links:

- For more information on Creating Application using the API flow, see here.

Launcher App

The launcher app functions by directing users to the URL specified in its configuration, similar to a browser bookmark. The launcher app serves as a quick access shortcut for all applications, whether they use protocol flows like SAML/OIDC or whether they are simple web apps without any protocol integration.

To open your application using a specific URL or web address, select this option.

1. Select **Enable App Launcher.**
2. **Launch URL** - The URL to which users are redirected when they select this launcher application. This can point to a web application or a resource, regardless of whether it uses protocol flows (e.g., SAML/OIDC) or is a standalone web app without protocol integration.
3. **Effective LauncherApp Login URL** - This is a shortcut to the main application to which Launcher App is pointing.
4. **LauncherApp Logout URL** - This is the URL to logout the session from the application.
5. Click **Save Application**.
   The application is saved to the Application dashboard.
6. To navigate to the application dashboard, click **<** Applications on the top left of the applications page.

Reference Link: For details on creating a Launcher App using Symantec Identity Security Platform API, see here.

Application Groups

This section allows administrators to organize and manage applications into groups for streamlined security policy application and access control.

You can do the following in this page:

- **Search Bar:** Allows you to search for application groups by name or description. There is also a “Show Filters” option to refine your search (based on criteria like creation date, number of applications, etc.).
- **Table:** This is the main content area displaying a list of Application Groups:

  - **Name:** The name assigned to each application group (for example, "Apple Group", "elva").
  - **Description:** A brief explanation of what the application group is for.
  - **Created On:** The date and time when the group was created.
  - **Total Applications:** The number of applications associated with each group.
- **Pagination:** At the bottom, "Page x of x" indicates there is only x page of results currently displayed. You can adjust the “Items per page” dropdown to control how many groups are shown at a time.
- **Add Button**: The top right button allows you to add new application groups.

Create Application Group and Add Applications to the Group

This section describes the steps that are required to add an application group to the solution.

1. Log in to the Admin Console.
2. On the navigation panel, click **Application Groups**.
3. Define a new group by entering its **name**, providing a **description** and click **Add**.
   The newly created **application group** is displayed in the table in the **Application Groups** page.
4. To add applications into the group and assign policies to that group, click the newly created application group and click **Add Applications.**
5. Use the "search by app" option to search for applications that are already created and add them to this group by clicking **Submit**.
6. Once the applications are added to this group, now assign policies, and click **Done**.

Delete Application Groups

Once you have configured the application group, it appears on the **Applications** Group screen in a tabular format. The (**x**) at the end of each row lets you delete, the application group.

There are two ways of deleting an application group: a standard deletion that requires all references to be removed beforehand or forceful deletion. The administrator has the discretion to choose which option to use. In general, if a resource is not in-use for longer time, it gets cleaned up by the standard delete procedure.

Deleting an Application Group Referenced by Other IDSP Objects

You can delete an application group directly if it is not being used by other resources. A reconfirmation is requested before deleting an application group. Once confirmed, the application group is deleted. However, if one or more resource types have referenced this application group, a message is displayed stating that the application group cannot be deleted because it is still in use. The message looks like the following:

```
Application Group associated with one or more policies cannot be deleted.
[ txnId : ebe3ef4d-1c50-42ef-8989-f52bd1c10354 ]
```

In this case, you need to invoke the helper API, "**/ReferencingObjectsHelper**" which retrieves the list of dependent objects in case of a conflict. You need to review and manually remove the reference from each object before deleting the application group.

For more information, see ReferencingObjectsHelper API.

For example, the application group might be used in configurations such as Authentication (AuthN) policy, Authorization (AuthZ) Policy or Discovery Policy. **The delete is only allowed once all such references are removed.**

Forceful Deletion of Application Group

When attempting to delete an application group that is referenced by multiple policies, a warning dialog titled "Deleting this Application Group will remove all associations" appears. This dialog informs you about the potential impact of your action and displays a list of associated resources in a detailed table. You can then choose "Cancel" to abort the deletion or "OK"

to proceed. Clicking "OK" deletes the application group and automatically removes all its listed associations from policies. This process ensures full awareness of the consequences before finalizing the deletion. See Resource Deletion in IDSP using REST API.

Policies

Policies tab within the Symantec Identity Security Platform Admin Console encompasses Authentication (AuthN) Policies, and Identity Provider Discovery Policies.

> **NOTE**
> The admin console experience is tailored to match the current privileges of each administrator. Therefore, to access and modify information within the console, you must possess the necessary privileges. You are currently viewing this page because you hold the `urn:iam:t.authpolicies` privilege.

On the **POLICIES** page, you can select the type of policy you want to create or manage.

1. On the Admin Console navigation panel, click **POLICIES**.
   The POLICIES menu is displayed.
2. Select the Policy type you want to create, and click to navigate to the respective page for detailed information.
   - Authentication Policies
   - Identity Provider Discovery Policies

Authentication and Access Policies

Learn about creating an Authentication or Access Policy, including essential attributes like rules, and conditions using the Symantec Identity Security Platform Admin Console.

To know more about Authentication Policy APIs, its operations, and examples, see Authentication Policy APIs.

Authentication and access policies can be applied to the appropriate applications that are configured in the IDSP. A single policy can be applied to a single application or to a list of selected applications, or to all applications based on the requirement.

This topic includes the following sections:

- Steps to add or import an Authentication or Access Policy
- Adding an Authentication or Access Policy
- Importing an Authentication or Access Policy
- Testing an Authentication or Access Policy
- Using Search and Filter options

Follow these steps to create an Authentication or Access Policy:

1. Log in to the Admin Console.
2. On the navigation panel, click **Policies** > **Authentication and Access Policies**.
   The **Authentication and Access Policies** page is displayed.
3. On this page, you can perform the following tasks:
   - **Add** - Use this option to add a policy in the IDSP. Details of how to create an Authentication or Access policy is described in the Adding an Authentication Policy section.
   - **Import** - Use this option to import an existing policy into the IDSP. Details of how to import an Authentication or Access policy is described in the Importing an Authentication Policy section.
   - **Test Policy** - Use this policy to validate that the policy configurations are valid. For more information, see Testing an Authentication or Access Policy.

4. In addition to these options, you are also displayed with the following details regarding each policy that is created within the solution. This page includes information about previously created policies, and details of new policies upon saving. The following details are presented in tabular form on this page:
   - **Policy Name**- displays the title of the policy.
     The following five access policies are pre-configured in the IDSP Admin Console:
     - **AdminLoginPolicy** - To know more about this policy, see Admin Login Policy.
     - **SelfServiceLoginPolicy** - To know more about this policy, see Self-Service Login Policy.
     - **ActivateAccountPolicy** - To know more about this policy, see Activate Account Policy.
     - **UnlockAccountPolicy** - To know more about this policy, see Unlock Account Policy.
     - **ResetPasswordPolicy** - To know more about this policy, see Reset Password Policy.
   - **Priority**- displays the priority of the policy, if configured. To know more about policy priority, see Setting Priority in Authentication Policy and Rule.
   - **Description**- displays the description of the policy, provided you entered information into this field during its creation. If the field was left empty, no description is displayed.
   - **Modified**- displays the user name and date and time stamp for the most recent creation or modification of the policy.
   - **Policy Status**- displays whether a policy is **active** or **inactive**.
   - **Priority**- displays the priority set for this policy if configured while creating the policy. For more information on this , see Setting Priority in Authentication Policy.

   You can configure the number of items that you want to be displayed in the table by setting the number in the **Item per page** field. If multiple policies are created that run through multiple pages, you can type in the page number you want to navigate to in the **Page** field.
5. The ellipses (**...**) at the end of each policy in the table lets you create a duplicate of the selected policy, export the policy, and delete the policy.
   - **Duplicate**: Use this option to create a duplicate policy of the existing policy. A duplicate policy with a different suffix gets added just below the original policy.
   - **Export**: Use this option to export the existing policy content into a json file that is available to you in your downloads folder.
   - **Activate/Disable**: This button gets displayed only when a policy is in the inactive state. Use this option to activate the policy. If the policy is already active than the Disable button is displayed, which gives lets you deactivate a policy.
   - **Set Priority**: Lets you set the priority for the policy, in case you missed configuring it when creating the policy. To know more about policy priority, see Setting Priority in Authentication Policy and Rule.
   - **Delete**: Use this option to delete a policy. A reconfirmation is requested before deleting a policy. Once confirmed, the policy is deleted. This option is only available for policies that you have created and not for the out-of-the-box policies.
6. Click the policy name to view full details of, and manage, the policy. The **Policies** page displays the following tabs to only users that are stored in the Internal Identity Store:
   - The **Details** tab displays the full details of the policy, and lets you perform the following actions:
     - **Assign Manager**: Use this option to assign an entity to manage this policy.
     - **Duplicate**: Use this option to create a duplicate policy of the existing policy. A duplicate policy with a different suffix gets added just below the original policy.
     - **Export**: Use this option to export the existing policy content into a json file that is available to you in your downloads folder.
     - **Delete**: Use this option to delete a policy. A reconfirmation is requested before deleting a policy. Once confirmed, the policy is deleted.
   - The **Managers** tab lets you view and modify the entities that can currently manage this policy. From the vertical ellipses (**...**) next to each row, you can remove a user or a user group that has been assigned as a manager:

- Click **Remove and reconform your action** to remove the assigned manager.

– The **Authorization** tab lets you view and modify the resource authorizations that have been assigned to this policy. From this tab, you can add tags to this application.
  - **Tags**- The tag is a name-value pair that you use to assign an attribute to this resource. Once a resource is tagged with a criteria, you can include that criteria in a constraint statement for any policy that affects that resource.

    The format must be `name=value`. However, there are no restrictions on the name or value that you assign.

Adding a Policy

This section describes the steps that are required to add an Authentication Policy using the IDSP Admin Console application.

1. Click **+Add** on the **Authentication and Access Policies** page.
   The **Add Policy** pop-up is displayed.
2. Enter the **Name** and the **Description** of the policy and then click **Add**.
   You are displayed with the message - **Policy Created Successfully**. The newly created Authentication Policy page is displayed where you are required to add other details. The Policies page displays two tabs - Details, and Modified By.
3. The **Details** tab displays the following sections:
   – **General Info**:
     - **Policy ID**: displays the auto-generated IDSP policy ID.
     - **Priority**: (Optional) Set the priority of the policy ranging from 1 (highest) to 1,000,000 (lowest). Decimals can be used for finer control. For more information on how priority influences an Authentication Policy result, see Setting priority in Authentication Policy and Rule.

   – Applications or Application Groups: select either of the following options:

     NOTE
     These options are not available for the pre-configured Account Activation, Account Unlock, and Password Recovery access policies.

     - **All Applications**: Select this option if you want the policy to be associated with all the applications that have been added to IDSP.
     - **Selected Applications or Application Groups**: Select this option when you want to associate the policy with applications or application groups. A list of applications is displayed from where you can choose the applications with which this policy should be associated. If there are no applications added, then you are displayed with the out-of-the-box "AdminLoginPolicy."
     - **Add Admin Console or Self-service Console** : You can select the out-of-the-box Admin Console, or Self-Service Console option if you want to associate the authentication policy to any of these applications. Otherwise, you can click **Select Application** which displays the search option. You can search by app name or description, or use filters for finding an application.

   – **Rules**: In this section, you must incorporate rules that establish the conditions for an authentication policy. Rules consist of two primary attributes: **conditions** and **result**. Various conditions are available for different authentication purposes. You must select and configure conditions from the provided list. You can define a single condition or multiple conditions within a rule according to your needs. During the authentication process, when incoming user inputs align with the configured rules, the policy undergoes evaluation, and the obligations are presented in the result section of the policy.

     Perform the following steps to add rules to a policy:

     a. Click **+Add Rule**.
        The **Add Rule** page is displayed.
     b. Enter a **Description** of this rule.
     c. Each rule type fulfills a unique role within the authentication and access control framework, allowing administrators to create and implement diverse security policies to meet any organizational needs.

NOTE
You cannot modify the rules for the pre-configured Account Activation, Account Unlock, and Password Recovery access policies.

Select the type of rule that you want to create to identify its purpose from the following options:

- **Authenticate**: This type of rule is used to create conditions that are related to user authentication. The rule defines the criteria that users must meet to successfully authenticate before accessing a resource.
- **Post Authenticate Access**: After successful authentication, this type of rule governs access to specific resources. The rule is applied once the user identity has been authenticated, ensuring that access permissions are provided based on additional criteria such as user roles, group memberships, etc. For this rule type to apply, you need to configure the application setting, "**Require Post Authentication Access Control"**. For more information, see application setting.
- **Custom Action**: These rules provide flexibility in implementing custom authentication methods and access control process.
  For example, you could configure "Wealth Management" as a custom action. This action condition is added to test continuous authentication on a web channel, where "Nbruce" logs in and selects wealth management to invoke this condition.

d. **Priority**: (Optional) Set the priority of the rule ranging from 1 (highest) to 1,000,000 (lowest). Decimals can be used for finer control. For more information on how priority influences an Authentication Policy result, see Setting priority in Authentication Policy and Rule.

e. Click **Next**.
The **rule** page is displayed where you are required to add conditions.

f. Click **Add Conditions** drop-down.
The drop-down list displays the supported conditions. You can select single, multiple, or all the conditions to evaluate an authentication request. Conditions express the relationship of an attribute to a value. For example, if you specify "action" as one of the attributes, then the value could be "login".

g. Select **conditions** based on your business requirement, and click **Add**.
   - **IP Address**
     Define the IP addresses or a range of IP addresses from which the user must access the application.
     - **Select file to upload**: Use this option to upload a comma-separated values (CSV) file with IP Addresses or ranges.
       - **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from specified IP addresses or ranges to fulfill this condition.
         - **Value is within configured IP address or range** - Selecting this option implies that users coming from these IP addresses or ranges should be subjected to this condition to discover the Identity Provider.
         - **Value is outside configured IP address or range** - Selecting this option implies that users coming from external IP addresses or ranges outside the configured IP addresses are subjected to this condition to discover the Identity Provider.
     - **Enter manually**:
       a. **IP Address or Range**: Enter the IP Address or a range of IP addresses.
       Once you enter the details, click **Add**.
       The IP Addresses or the range of IP addresses that you entered gets displayed in a tabular form right after it. The table displays the **IP Address** and **Type**.
       To delete the IP Address or range of IP addresses that you entered, click the X button next to it.
       b. **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from specified IP addresses or ranges to fulfill this condition.

        - **Value is within configured IP address or range** - This option implies that users coming from these IP addresses or ranges should be subjected to this condition to discover the Identity Provider.
        - **Value is outside configured IP address or range** - This option implies that users coming from external IP addresses or ranges outside the configured IP addresses should be subjected to this condition to discover the Identity Provider.
- **Channel**
  This condition defines the various platforms through which users access the system.
  Select the channel that must be included in this condition from the following options:
  - **Web**: users accessing the system through a web browser
  - **Mobile**: users use a dedicated mobile application to access the system on their smartphones or tablets.
  - **Custom Channel**: This option allows you to define a specialized channel. For instance, it could be used for users accessing the system through a specific device or authentication method, such as a FIDO (Fast Identity Online) device.

  You can select one or more channels, based on your requirement.
  - **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from any of these specified channels to fulfill this condition:
    - **Value matches any selected channel** - This option implies that users coming from these channels should be subjected to this condition to discover the Identity Provider.
    - **Value does not match any selected channel** - This option implies that users coming from channels that are not configured should be subjected to this condition to discover the Identity Provider.
- **Country**
  Select the countries that the user or client configures to allow or restrict access to the resource. Choose countries from the **ALL COUNTRIES** list and click **>** to move them to the **SELECTED COUNTRIES** list.
  - **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from any of these specified countries to fulfill this condition:
    - **Value matches any selected country** - This option implies that users coming from these countries should be subjected to this condition to authenticate.
    - **Value does not match any selected country** - This option implies that users coming from countries that are not configured under the **SELECTED COUNTRIES** should be subjected to this condition to authenticate.
- **Groups**
  Set the user groups for this condition. From the identity source drop-down list, select the user group to allow or restrict access to a protected resource.
  - **Any Identity Source**: If you select this option, enter the names of the groups which may belong to any of the identity stores and to whom this condition can be applied. Click **Add**.
  - **internal (Internal Identity Store)**: The Internal Identity Store is the identity store that is provided out-of-the-box during the IDSP deployment. If you select this option, perform the following steps:
    - Select an option to filter groups in the internal identity store from the drop-down list. The drop-down includes the following filter options:
      - **contains**: This filter option returns groups that contain the specified value within their name.
      - **starts with**: This filter option returns groups whose names begin with the specified value.
      - **ends with**: This filter option returns groups whose name ends with the specified value.
      - **equals**: This filter option returns groups that exactly match the specified value. This filter option is a strict match where the name of the group must be identical to the search query.
    - Alternatively, you can enter a keyword to search for a specific group in the search field. The filter option helps to refine the search results.
    - If user groups are added to the internal identity store, the groups get populated in the drop-down. Select the group, and click **Add**.
      The added group is displayed in a tabular form with **Identity Source** and **Group** details.
  - **LDAP Config** : If you select this option, select the operator type and select a user group, and click **Add**.

- **Select Identity Provider**: If you select this option, you are displayed with the search option by name or description, or use filters for finding an Identity Provider. Once the Identity Provider is selected, click **Select**.
- **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from any of these specified groups to fulfill this condition:
  - **Value matches any selected group** - This option implies that users coming from these groups should be subjected to this condition to authenticate.
  - **Value does not match any selected group** - This option implies that users coming from groups that are not configured under the **SELECTED COUNTRIES** should be subjected to this condition to authenticate.

- **Users**
  From the identity source drop-down list, select the identity source from where the users can be selected to allow or restrict access to a protected resource. The configured identity stores are displayed.
  - **Any Identity Source**: If you select this option, enter the names of the users who may belong to any of the identity stores and to whom this condition can be applied. Click **Add**.
  - **internal (Internal Identity Store)**: The Internal Identity Store is the identity store that is provided out-of-the-box during the IDSP deployment. If you select this option, perform the following steps:
  - **LDAP Config** : If you select this option, select the operator type and select a user group, and click **Add**.
  - **Identity Providers**: If you select this option, you are displayed with the search option where you can search by name or description, or use filters for finding an Identity Provider. Once the Identity Provider is selected, click **Select**.

  Once you select the identity source, select one of the following options to add users:
  - **All Users**: Select all users available within the selected identity source who should be allowed or denied access.
  - **Selected Users**: Select specific users within the selected identity source who should be allowed or denied access.
  - **Expression**: IDSP supports Attribute-Based Access Control (ABAC) which is an authorization model that evaluates the user attributes in the authentication policy conditions to determine the level of user access. This condition allows the identification of users based on attributes that are found in their user record from external (LDAP/SCIM), Internal Identity Store or IDP, if applicable.
    For example, you can add an expression like this: `(email ne *@example.com)` where the User email should not be *@example.com, only then the rule that is defined in the policy is applied. To know more about supported expressions, see Supported expressions.
    a. Enter an expression.
    b. Click **Expression Builder**. The list of configured user attributes gets displayed
    c. Click **Add**.
  - **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from any of these specified identity stores to fulfill this condition:
    - **Value matches any selected username or expression** - This option implies that users or expressions coming from the selected identity stores should be subjected to this condition to authenticate.
    - **Value does not match any selected username or expression** - This option implies that users or expressions not coming from the selected identity stores should be subjected to this condition to authenticate.
- **Risk Score**:
  The Risk Score condition enables you to determine the likelihood of the authentication attempt being malicious or fraudulent. In this condition, define the risk score for this rule (enter values from 0 through 100). Higher scores indicate a higher level of risk.
  a. Select the risk score operator value from the **Risk threshold is** drop-down list:

- **=**: eq (equal to)
- **>**: eq (greater than)
- <: eq (Less than)
- **>=**: eq (greater than equal to)
- **<=**: eq (Less than equal to)

b. Enter the risk score.

- **Triggered Risk Rule**
  Select the risk rules that the user or client uses to allow or restrict access to the resource. The supported risk rules are:
  - **Behavior Anomaly**
  - **Device Recognition**
  - **Inactive user**
  - **Previous event is confirmed risk**
  - **User velocity**
  - **Work day time**
  - **Enhanced difficult travel**
  - **Risky country**
  - **Fraud IP Address**
  - **Risky IP Address**
  - **Allowed IP Address**
  - **Recognized computer**
  - **Country change**
  - **IP change**

    NOTE
    To know more about these risk rules, see the section on Understanding Risk Rules.
  - **Apply condition if**: Select one of the following conditions from the drop-down list to allow users using any of these specified risk rules to fulfill this condition:
    - **Value matches any selected risk rule** - This option implies that users using any of the selected risk rules should be subjected to this condition to authenticate.
    - **Value does not match any selected risk rule** - This option implies that users using any of the selected risk rules should be subjected to this condition to authenticate.
- **ACR**
  Set the authentication methods that should be included in this condition. Also, specify if the selected method should be used to allow or deny a user or client from accessing a resource. The ACR methods are processed based on the preferred order in which they are added to the policy. When a user provides input that contains the "acrValues" attribute, then this policy is triggered.
  - **Basic**: urn:iam:acr:aal:aal1
  - **Multi-Factor Authentication**: urn:iam:acr:aal:aal2
  - **Hardware Assist**: urn:iam:acr:aal:aal3
  - **Custom** : You can create a custom method of authentication by entering the custom ACR value. Click **Add** .
- **Context**
  Build an expression based on various context values to define the condition to trigger this rule.
  - Enter an expression.
    Example of an expression that can be used in the IDP context:
    `( ${idpmetadata.environment} eq ${clientcontext.env} )`
    This example denotes that the expression evaluates the environment of the client context and verifies whether the environment that is specified in the IDP metadata aligns with that of the client context to identify the Identity Provider.

h. Click **Next**.
The **Result** page is displayed.

i. On the **Result** page, specify the outcome of the policy. You must configure whether to allow or deny access, along with options for reauthentication, multi-factor authentication, and the authentication level to be populated in the ID token and Access Token. These settings determine what information should be provided to users once they meet the specified conditions.
   - Select the ***result*** for this rule:
     - **Allow Access**: Select this option if you want the user to access the protected resource.
     - **Deny Access**: Select this option if you do not want the user to access the protected resource.
   - Select whether the user must ***authenticate again*** if the app requires additional proof of access in the same session
     - **Yes**: Select this option if you want the user to reauthenticate
     - **No**: Select this option if you do not want the user to reauthenticate.
   - Prompt user for ***multi-factor authentication*** (MFA)
     - **Always**: select this option if you want the user to perform multi-factor authentication every time they log in to access the protected resource.
     - **First time only (if Trusted Device is enabled)**: Select this option if you only want the user to perform multi-factor authentication the first time they log in. If the "Trusted Device" feature is activated for the user, the user will no longer be required to perform multi-factor authentication.
   - **Authentication level** to be populated in the ID token and Access Token:
     - Enter the **Authentication Level**.

j. Click **Next**.
The **Authenticator Factors** page is displayed.

k. Select a single factor or multiple factors for the Primary authentication, and the additional factor levels as required.
You can apply up to a maximum of four-factor levels. The list of factors that the solution supports are displayed in the primary authentication box. Select the checkbox beside the factor/factors for the primary authentication and click the additional factors if you want to add an additional level of authentication factor. To undo any factors in any of the levels, unselect the checkbox beside the factor in the box. You can also delete any of the factor levels by clicking the cross available on the top-right corner of the factor box.

   NOTE
   The pre-configured account unlock, account activation, and password recovery actions do not allow inline credential registration. Your user must have already registered for credentials of the types that you set here.

l. Click **Next**.
The **Summary for Authentication Rule** page is displayed. In the Summary for Authentication Rule page, the details of the policy that you have created gets displayed. The page shows the conditions that are applied, the results selected for the policy, and the authentication factors applied to the policy.

m. To edit the **Conditions**, **Result**, or **Authentication Factors**, click **Edit** that is displayed next to these fields. This step takes you to the respective pages for modifications.

n. Click **Done**.
The Policy page is displayed.

o. In the policy page, you are displayed the rules that you have configured for this policy. The following details are presented in tabular form on this page:

- **Description**- displays the description of the rule, provided you entered information into this field during its creation. If the field was left empty, no description is displayed.
- **Priority**- displays the priority set for this rule if configured while creating the rule.
- **Type**- displays the type of the rule you created from the given choices.
- **Conditions**- displays the user name and date and time stamp for the most recent creation or modification of the policy.
- **Result**- displays whether a policy is **active** or **inactive**.
- **Prompt for MFA**- displays your selection of either "Always" or "First Time only" for MFA.
- **Factors**- displays the factors selected for this rule.

On the right of each rule, you can use the **Delete** option to delete a rule. A reconfirmation is requested before deleting a rule. Once confirmed, the rule is deleted. You can configure the number of items that you want to be displayed in the table by setting the number in the **Item per page** field. If multiple rules are created that run through multiple pages, you can type in the page number you want to navigate to in the **Page** field.

p. **Test Policy**: (Optional) Test your policy to validate that the policy configurations are valid. This has been explained here in details.
To perform this feature using the IDSP API flow, see Authentication Policy Evaluator API.

q. **Save & Activate**: You can perform the following actions:
- **Save Policy**: Selecting this option saves your policy. The policy is displayed in the Policy dashboard in the Inactive state.
- **Inactive Status**: Select this option to change the status of the policy from **Inactive** to **Active** state. Only active policies are evaluated during the authentication process.

Importing a Policy

You may choose to import a previously created policy. Only JSON-format files can be imported into the admin console.

To import a policy, perform these steps:

1. Click **+Import**.
   Select a JSON file from your system and click **Open**.
   NOTE
   If the policy is not saved in the supported format, the system displays an error message- "Unknown Property: info or Invalid JSON."
   After the policy is successfully imported, you are displayed with a confirmation message stating "The policy is imported successfully".
   You can access the imported policy on the Policy page.

Testing a Policy

(Optional) Test your policy to validate that the policy configurations are valid. You can do so by defining your test criteria and providing test data in the fields that are populated in the **Test Policies** page, and click **Submit**. IDSP identifies which policies and rules this data triggers and thereby displays the expected behavior of the policies or rules. You need to verify that the behavior matches your expectations before saving and activating the policy. If there are any conflicts between policies and the rules, correct them and retry this test.
To perform this feature using the IDSP API flow, see Authentication Policy Evaluator API.

Follow these steps to test the policy:

1. Enter the **application** for which this authentication policy is assigned.
2. Enter the **Identity Source** configured for this application.
3. Enter the **Username** defined in the policy condition.
4. (Optional) Enter the **IP Address** that is configured in the policy condition.
5. Click **Submit** to initiate the policy evaluation.

Advanced Testing Attributes

- Channel- Select the channel that is assigned for this policy
- Risk Score- Enter risk score, if it is defined in the policy condition.
- Client Context- Enter key-value pair if configured for the client context.
- Triggered Risk Rules- Select the risk rules that are triggered by the policy.

NOTE

- If you enabled a rule for access policy, it will be evaluated. If not enabled, it is ignored.
- Before submitting the policy for evaluation, ensure that the policy is activated.

Using Search and Filter

Leverage the search and filter options to find specific authentication or access policies by applying custom filters that refine your search.

Follow these steps:

1. Enter the name or description of the authentication policy that you want to find in the search box.
2. Enter the Identity Source configured for this policy.
3. Enter the Username that is available in the configured Identity Source.
4. (Optional) Enter the IP Address of the user
5. Click **Submit**.

Applying Filter

1. Click **Show Filters** to open the filter options if it is hidden.
2. Select the attribute from the dropdown with which you want to search the required authentication or access policies.
3. Set the operator.
4. Provide a specific value.

You can click (+) to add multiple filters if necessary or remove existing ones by clicking "X" next to each filter.

Matching Filters

To apply filters and get the search results for the configured authentication or access policies that match your criteria, perform the following steps:

1. On the top right of the page, the "**Match filters**" dropdown is available.
   Select whether you want to match either option:
   – "All" filters (return results that match all specified conditions)
   – "Any" filters (return results that match at least one condition)
2. Click **Submit**.

Clear Filters

Use this option if you want to reset your search and filters.

- Click **Clear Filters** to remove all applied conditions and start a new search.
  This filter allows you to narrow down the authentication policy search results using specific criteria and get the most relevant results efficiently.

Reference Links

- Creating Authentication Policy using REST API

Identity Provider Discovery Policies

Learn about creating an IDP Discovery Policy, including essential attributes like rules, and conditions using the Symantec Identity Security Platform Admin Console.

IDP Discovery Policy Overview

An IDP Discovery Policy is a set of rules and conditions that determine how Identity Providers (IDPs) are discovered and used during the authentication process within Symantec Identity Security Platform. When IDP authentication occurs through a front-channel browser in real time, the typical integration involves the use of industry-standard "Federated Identity Providers" like OpenID Connect or SAML. As part of this authentication, the knowledge about the principal's identity may be with, or may be delegated to, an external entity responsible for handling all or some of the assurance. Authentication involves validating the user's or client's identity at a certain level of assurance which is consistent with the requirement of the application that is being accessed by the user or client.

The IDP Discovery Policy typically includes:

- ***Rules***: These rules define conditions that must be met to determine which IDP should be used for authentication. IDP discovery uses a generic Discovery Policy that consists of multiple rules, mapping action-specific context conditions to the IDPs.
- ***Policy Evaluation***: Active policies are evaluated against given context resulting in a set of **unique** and **active** *applicable providers*.
- ***User Experience***: IDP Discovery Policies can also impact the user experience by determining whether users are automatically redirected to the appropriate IDP, or the user is required to manually select their identity provider.

Overall, the IDP Discovery Policy streamlines the authentication process by ensuring that users are directed to the correct IDP based on the defined rules and conditions, ultimately enhancing security and user convenience.

This page includes the following sections:

- Adding an IDP Discovery Policy
- "Before user enters the username" IDP flow
- "After user enters the username" IDP flow
- "When User is Found" IDP flow
- "When User is Not Found" IDP flow
- Importing a policy

Follow these steps:

1. Log in to the Admin Console.
2. On the navigation panel, click **Policies** > **Identity Provider Discovery Policies**.
   The **Identity Provider Discovery Policies** page is displayed.
3. On this page, you can perform the following tasks:
   - **Add** - Use this option to add a policy in the IDSP. Details of how to create an IDP discovery policy is described in the Adding an IDP Discovery Policy section.
   - **Import** - Use this option to import an existing policy into the IDSP. Details of how to import an IDP discovery policy is described in the Importing an IDP Discovery Policy section.
4. In addition to these options, you are also displayed with the following details regarding each policy that is created within the solution. This page includes information about previously created policies, and details for new policies upon saving. The following details are presented in tabular form on this page:

- **Policy Name**- displays the title of the policy.
- **Priority**- displays the priority of the policy, if configured. To know more about policy priority, see Setting Priority in Identity Policy and Rule.
- **Description**- displays the description of the policy, provided you entered information into this field during its creation. If the field was left empty, no description is displayed.
- **Modified**- displays the date and time of the most recent update to this policy.
- **Status**- displays whether a policy is **active** or **inactive**.

You can configure the number of items that you want to be displayed by setting the number in the **Item per page** field. If multiple policies are created that run through multiple pages, you can type in the page number you want to navigate to in the **Page** field.

5. The ellipses (**...**) at the end of each policy lets you create a duplicate of the selected policy, export the policy, and delete the policy.
   - **Duplicate**: Use this option to create a duplicate policy of the existing policy. A duplicate policy with a different suffix gets added just below the original policy.
   - **Export**: Use this option to export the existing policy content into a json file that is available to you in your downloads folder.
   - **Activate/Disable**: This button gets displayed only when a policy is in the inactive state. Use this option to activate the policy. If the policy is already active than the Disable button is displayed, which lets you deactivate a policy.
   - **Set Priority**: Lets you set the priority for the policy, in case you missed configuring it when creating the policy. To know more about policy priority, see Setting Priority in Authentication Policy and Rule.
   - **Delete**: Use this option to delete a policy. A reconfirmation is requested before deleting a policy. Once confirmed, the policy is deleted.
6. Click the policy name to view full details of, and manage, the policy. The **Policy** details page displays the following tabs to only users that are stored in the Internal Identity Store:
   - The **Details** tab displays the full details of the policy, and lets you perform the following actions:
     - **Assign Manager**: Use this option to assign an entity to manage this policy.
     - **Duplicate**: Use this option to create a duplicate policy of the existing policy. A duplicate policy with a different suffix gets added just below the original policy.
     - **Export**: Use this option to export the existing policy content into a json file that is available to you in your downloads folder.
     - **Delete**: Use this option to delete a policy. A reconfirmation is requested before deleting a policy. Once confirmed, the policy is deleted.
   - The **Managers** tab lets you view and modify the entities that can currently manage this policy. From the vertical ellipses (**...**) next to each row, you can remove a user or a user group that has been assigned as a manager:
     - Click **Remove and reconform your action** to remove the assigned manager.
   - The **Authorization** tab lets you view and modify the resource authorizations that have been assigned to this policy. From this tab, you can add tags to this application.
     - **Tags**- The tag is a name-value pair that you use to assign an attribute to this resource. Once a resource is tagged with a criteria, you can include that criteria in a constraint statement for any policy that affects that resource.

       The format must be `name=value`. However, there are no restrictions on the name or value that you assign.

Adding an Identity Provider Discovery Policy

This section describes the steps that are required to add an Identity Provider Discovery Policy using the IDSP Admin Console application.

1. Click **+Add** on the top-right corner of the **Identity Provider Discovery Policies** page.
   The **Add Policy** pop-up is displayed.
2. Enter the **Name** and the **Description** of the policy.

3. Set a **Priority** to the policy, if needed and then click **Add**. For more information on this, see Setting Priority in IDP Policy.
   You are displayed with the message - **Policy Created Successfully**. The newly created IDP Policy page is displayed where you are required to add other details. The Policies page displays two tabs - **Details**, and **Modified By**.
   – **Details** - This tab displays the **General Info** and the **Rules** sections.
   General Info: The general info section displays the following details:
      a. **Policy ID** - This field is generated automatically by IDSP.
      b. **Priority** - This field displays the priority level assigned to the policy, if set during creation. If no priority was initially configured, the field appears empty, but you will have the opportunity to set it later. For more information on this, Setting Priority in IDP Policy.
      a. Click +**Add Rule**.
      Adding Rules: These rules define the conditions for discovering the IDP for a user, client, or an application.
         a. Click +**Add Rule**.
         The **Add Rule** page is displayed.
         b. Enter a **Description** of this rule.
         c. Select one of the options from the **When does this rule apply** field.
            - **Before user enters username**
            - **After user enters username**
            - **When User is Found**
            - **When User is Not Found**
         d. Set a **Priority to the rule, if needed. For more information on this, see Setting Priority in IDP Policy.**
         e. Click **Next**
         The **Add Conditions** pop-up is displayed. Once you reach to this step, based on the IDP flow you intend to choose, appropriately select the section. To learn more about the IDP flows, see Before user enters username IDP flow or After user enters username IDP flow.
      b. **Modified by**: This tab displays the name of the administrator who initiated the policy creation, the date, and time of its creation.

"Before user enters username" IDP flow

To use the **Before user enters username** IDP flow, follow these steps:

1. On the **Conditions** page, select one of the options from the **Application** drop-down to configure the application to which this policy must be applied:
   – **All Applications**: applies the policy to all the applications that are created under the IDSP solution. If there are no other applications added, then only the out-of-the-box "AdminLoginPolicy" displays.
   – **Selected Applications**: enables you to filter the list of applications to which the policy should be applied. This list includes all existing applications that have been created in the IDSP solution.
2. (Optional step) Select an **Application Group** to which the IDP policy should be applied, and click **Submit**.
3. (Optional step) Add the applicable conditions to the policy. Supported conditions are:
   – **IP Address**
   Define the IP addresses or a range of IP addresses from which the user must access the application.
   - **Select file to upload**: Use this option to upload a comma-separated values (CSV) file with IP Addresses or ranges.
      - **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from specified IP addresses or ranges to fulfill this condition.

- **Value is within configured IP address or range** - Selecting this option implies that users coming from these IP addresses or ranges should be subjected to this condition to discover the Identity Provider.
- **Value is outside configured IP address or range** - Selecting this option implies that users coming from external IP addresses or ranges outside the configured IP addresses is subjected to this condition to discover the Identity Provider.

- **Enter manually**:
  a. **IP Address or Range**: Enter the IP Address or a range of IP addresses.
  Once you enter the details, click **Add**.
  The IP Addresses or the range of IP addresses that you entered gets displayed in a tabular form right after it.
  The table displays the **IP Address** and **Type**.
  To delete the IP Address or range of IP addresses that you entered, click the X button next to it.
  b. **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from specified IP addresses or ranges to fulfill this condition.
  - **Value is within configured IP address or range** - This option implies that users coming from these IP addresses or ranges should be subjected to this condition to discover the Identity Provider.
  - **Value is outside configured IP address or range** - This option implies that users coming from external IP addresses or ranges outside the configured IP addresses should be subjected to this condition to discover the Identity Provider.

– **Channel**
This condition defines the various platforms through which users access the system.
Select the channel that must be included in this condition from the following options:
- **Web**: users accessing the system through a web browser
- **Mobile**: users use a dedicated mobile application to access the system on their smartphones or tablets.
- **Custom Channel**: This option allows you to define a specialized channel. For instance, it could be used for users accessing the system through a specific device or authentication method, such as a FIDO (Fast Identity Online) device.

You can select one or more channels, based on your requirement.
- **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from any of these specified channels to fulfill this condition:
  - **Value matches any selected channel** - This option implies that users coming from these channels should be subjected to this condition to discover the Identity Provider.
  - **Value does not match any selected channel** - This option implies that users coming from channels that are not configured should be subjected to this condition to discover the Identity Provider.

– **Context**
Build an expression based on various context values to define the condition to trigger this rule.
- Enter an expression. Example of an expression that can be used in the IDP context:
  `( ${idpmetadata.environment} eq ${clientcontext.env} )`
  This example denotes that the expression evaluates the environment of the client context.

  - ${idpmetadata.environment} eq ${clientcontext.env} - This contextual expression verifies whether the environment that is specified in the IDP metadata aligns with that of the client context to identify the Identity Provider.
  - After completing adding the rules to the policy, click **Next**.
    The next step is to configure the allowed Identity Providers.

4. On the **Allowed Identity Providers** page, select the Identity Provider from the existing list of providers to which you want to assign the user that matches this policy.
   Use the Search box to filter the list of the available identity providers.
5. Click **Next**.

The **Summary** page is displayed which gives you a snapshot of the policy that you are going to save. At this point, you are given the option to edit the conditions and the Identity Providers.

a. Click **Edit** to take you back to the conditions or Identity Provider section where you can modify and click **Next** to come back to the Summary page.
b. Click **Done**.

6. Click **Save** the IDP policy.
   You are displayed with the message - **Policy Saved Successfully**.
7. Click **Active** to activate the policy.

"After the user enters the username" IDP flow

To use the **After user enters username** IDP flow, follow these steps:

1. On the **Conditions** page, select one of the options from the **Application** drop-down to configure the application to which this policy must be applied:
   – **All Applications**: applies the policy to all the applications that are created under the IDSP solution.
   – **Selected Applications**: enables you to filter the list of applications to which the policy should be applied. This list includes all existing applications that have been created in the IDSP solution.
2. (Optional step) Select an **Application Group** to which the IDP policy should be applied, and click **Submit**.
3. Select the **Login Name** to match the username of the user to an IDP. Following options provide flexibility in how user identities are mapped to Identity Providers, allowing administrators to define specific criteria based on their organizational requirements.
   Supported conditions are:
   – **Matches any domain**: This option indicates that the username of the user should match any domain that is associated with the specified IDP.
   Example: Any user with a valid email format will be matched to the IDP, regardless of their domain.
   – **Matches value**: With this option, you specify a specific value that the username must match exactly in order to be associated with the IDP.
   Example: If you set the value to *@example.*, users like test.user@example.com and test.user2@example.net will be matched with the IDP.
   – **Does not match value**: This option specifies a value that the username must not match in order to be associated with the IDP.
   Example: If you set the value to *@example.*, users like test.user@example.com and test.user2@example.net will NOT be matched with the IDP, but user@company.org would be eligible.
   – **Matches regex**: This option allows you to define a regular expression (regex) pattern that the username must match in order to be associated with the IDP. Regular expressions provide a flexible way to match patterns in strings.
   Example: Using the pattern "(?i)^.*@example\\\\.(com|net)$"*@", users like test.user@example.com and test.user2@example.net will be matched with the IDP.
   – **Does not match regex**: This option specifies a regex pattern that the username must not match to be associated with the IDP. The option excludes usernames that match the specified pattern from being associated with the IDP.
   Example: Using the pattern "(?i)^.*@example\\\\.(com|net)$"*@", users like test.user@example.com and test.user2@example.net will not be matched with the IDP but user@company.org would be eligible.
4. (Optional step) Add the applicable conditions to the policy. Supported conditions are:
   – **IP Address**
   Define the IP addresses or a range of IP addresses from which the user must access the application.
   • **Select file to upload**: Use this option to upload a comma-separated values (CSV) file with IP Addresses or ranges.
   • **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from specified IP addresses or ranges to fulfill this condition.

- **Value is within configured IP address or range** - Selecting this option implies that users coming from these IP addresses or ranges should be subjected to this condition to discover the Identity Provider.
- **Value is outside configured IP address or range** - Selecting this option implies that users coming from external IP addresses or ranges outside the configured IP addresses is subjected to this condition to discover the Identity Provider.

- **Enter manually**:
  a. **IP Address or Range**: Enter the IP Address or a range of IP addresses.
  Once you enter the details, click **Add**.
  The IP Addresses or the range of IP addresses that you entered gets displayed in a tabular form right after it.
  The table displays the **IP Address** and **Type**.
  To delete the IP Address or range of IP addresses that you entered, click the X button next to it.
  b. **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from specified IP addresses or ranges to fulfill this condition.
  - **Value is within configured IP address or range** - This option implies that users coming from these IP addresses or ranges should be subjected to this condition to discover the Identity Provider.
  - **Value is outside configured IP address or range** - This option implies that users coming from external IP addresses or ranges outside the configured IP addresses should be subjected to this condition to discover the Identity Provider.

– **Channel**
This condition defines the various platforms through which users access the system.
Select the channel that must be included in this condition from the following options:
- **Web**: users accessing the system through a web browser
- **Mobile**: users use a dedicated mobile application to access the system on their smartphones or tablets.
- **Custom Channel**: This option allows you to define a specialized channel. For instance, it could be used for users accessing the system through a specific device or authentication method, such as a FIDO (Fast Identity Online) device.

You can select one or more channels, based on your requirement.
- **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from any of these specified channels to fulfill this condition:
  - **Value matches any selected channel** - This option implies that users coming from these channels should be subjected to this condition to discover the Identity Provider.
  - **Value does not match any selected channel** - This option implies that users coming from channels that are not configured should be subjected to this condition to discover the Identity Provider.

– **Context**
Build an expression based on various context values to define the condition to trigger this rule.
- Enter an expression. Example of an expression that can be used in the IDP context:
  `( ${idpmetadata.environment} eq ${clientcontext.env} )`
  This example denotes that the expression evaluates the environment of the client context.

  - ${idpmetadata.environment} eq ${clientcontext.env} - This contextual expression verifies whether the environment that is specified in the IDP metadata aligns with that of the client context to identify the Identity Provider.
  - After completing adding the rules to the policy, click **Next**.
    The next step is to configure the allowed Identity Providers.

5. On the **Allowed Identity Providers** page, select the Identity Provider from the existing list of providers to which you want to assign the user that matches this policy.
   Use the Search box to filter the list of the available identity providers.
6. Click **Next**.

The **Summary** page is displayed which gives you a snapshot of the policy that you are going to save. At this point, you are given the option to edit the conditions and the Identity Providers.

a. Click **Edit** to take you back to the conditions or Identity Provider section where you can modify and click **Next** to come back to the Summary page.

b. Click **Done**.

7. Click **Save** the IDP policy.
   You are displayed with the message - **Policy Saved Successfully**.
8. Click **Active** to activate the policy.

"When user is Found" flow and "When user is Not Found" flow

Both these flows share common steps with the "beforeUserEntersUsername" flow. Therefore, refer to the steps in the "Before user enters the username" IDP flow section.

Importing a Policy

You can import a previously created policy. Only JSON-format files can be imported into the Admin Console.

To import a policy, perform these steps:

1. Click **+Import**.
   Select a JSON file from your system and click **Open**.

   NOTE
   If the policy is not saved in the supported format, the system displays an error message- "Unknown Property: info or Invalid JSON."

   After the policy is successfully imported, you are displayed with a confirmation message stating "The policy is imported successfully".
   You can access the imported policy on the Policy page.

Risk Policies

Learn about creating a Risk Policy, including essential attributes like rules, and conditions using the Symantec Identity Security Platform Admin Console.

A Risk Policy is a set of rules and conditions that determine how to evaluate a user's risk profile at an application-level.

This page includes the following sections:

- Steps to add or import a Risk Policy
- Adding a Risk Policy
- Importing a Policy

Steps to add or import a Risk Policy

Follow these steps:

1. Log in to the Admin Console.
2. On the navigation panel, click **Policies** > **Risk Policies**.
   The **Risk Policies** page is displayed.
3. On this page, you can perform the following tasks:
   – **Add** - Use this option to add a policy in the IDSP. Details of how to create an Risk policy is described in the Adding a Risk Policy section.
   – **Import** - Use this option to import an existing policy into the IDSP. Details of how to import a Risk policy is described in the Importing a Risk Policy section.

4. In addition to these options, you are also displayed with the following details regarding each policy that is created within the solution. This page includes information about previously created policies, and details for new policies upon saving. The following details are presented in tabular form on this page:
   – **Policy Name**- displays the title of the policy.
   – **Priority**- displays the priority of the policy, if configured. To know more about policy priority, see Setting Priority in Authentication Policy and Rule.
   – **Description**- displays the description of the policy, provided you entered information into this field during its creation. If the field was left empty, no description is displayed.
   – **Modified**- displays the date and time of the most recent update to this policy.
   – **Status**- displays whether a policy is **active** or **inactive**.

   You can configure the number of items that you want to be displayed by setting the number in the **Item per page** field. If multiple policies are created that run through multiple pages, you can type in the page number you want to navigate to in the **Page** field.
5. The ellipses (**...**) at the end of each policy lets you create a duplicate of the selected policy, export the policy, and delete the policy.
   – **Duplicate**: Use this option to create a duplicate policy of the existing policy. A duplicate policy with a different suffix gets added just below the original policy.
   – **Export**: Use this option to export the existing policy content into a json file that is available to you in your downloads folder.
   – **Activate/Disable**: This button gets displayed only when a policy is in the inactive state. Use this option to activate the policy. If the policy is already active than the Disable button is displayed, which gives lets you deactivate a policy.
   – **Set Priority**: Lets you set the priority for the policy, in case you missed configuring it when creating the policy. To know more about policy priority, see Setting Priority in Authentication Policy and Rule.
   – **Delete**: Use this option to delete a policy. A reconfirmation is requested before deleting a policy. Once confirmed, the policy is deleted.
6. Click the policy name to view full details of, and manage, the policy. The **Policy** details page displays the following tabs to only users that are stored in the Internal Identity Store:
   – The **Details** tab displays the full details of the policy, and lets you perform the following actions:
     • **Assign Manager**: Use this option to assign an entity to manage this policy.
     • **Duplicate**: Use this option to create a duplicate policy of the existing policy. A duplicate policy with a different suffix gets added just below the original policy.
     • **Export**: Use this option to export the existing policy content into a json file that is available to you in your downloads folder.
     • **Delete**: Use this option to delete a policy. A reconfirmation is requested before deleting a policy. Once confirmed, the policy is deleted.
   – The **Managers** tab lets you view and modify the entities that can currently manage this policy. From the vertical ellipses (**...**) next to each row, you can remove a user or a user group that has been assigned as a manager:
     • Click **Remove and reconform your action** to remove the assigned manager.
   – The **Authorization** tab lets you view and modify the resource authorizations that have been assigned to this policy. From this tab, you can add tags to this application.
     • **Tags**- The tag is a name-value pair that you use to assign an attribute to this resource. Once a resource is tagged with a criteria, you can include that criteria in a constraint statement for any policy that affects that resource.

       The format must be `name=value`. However, there are no restrictions on the name or value that you assign.

Adding a Policy

This section describes the steps that are required to add an Authentication Policy using the IDSP Admin Console application.

1. Click **+Add** on the **Authentication Policies** page.
   The **Add Policy** pop-up is displayed.
2. Enter the **Name** and the **Description** of the policy and then click **Add**.
   You are displayed with the message - **Policy Created Successfully**. The newly created Authentication Policy page is displayed where you are required to add other details. The Policies page displays two tabs - Details, and Modified By.
3. The **Details** tab displays the following sections:
   - **General Info**:
     - **Policy ID**: displays the auto-generated IDSP policy ID.
     - **Priority**: (Optional) Set the priority of the policy ranging from 1 (highest) to 1,000,000 (lowest). Decimals can be used for finer control. For more information on how priority influences an Authentication Policy result, see Setting priority in Authentication Policy and Rule.
   - Applications or Application Groups: select either of the following options:
     - **All Applications**: Select this option if you want the policy to be associated with all the applications that have been added to IDSP.
     - **Selected Applications or Application Groups**: Select this option when you want to associate the policy with applications or application groups. A list of applications is displayed from where you can choose the applications with which this policy should be associated. If there are no applications added, then you are displayed with the out-of-the-box "AdminLoginPolicy."
     - **Add Admin Console or Self-service Console** : You can select the out-of-the-box Admin Console, or Self-Service Console option if you want to associate the authentication policy to any of these applications. Otherwise, you can click **Select Application** which displays the search option. You can search by app name or description, or use filters for finding an application.
   - **Rules**: In this section, you must incorporate rules that establish the conditions for an authentication policy. Rules consist of two primary attributes: **conditions** and **result**. Various conditions are available for different authentication purposes. You must select and configure conditions from the provided list. You can define a single condition or multiple conditions within a rule according to your needs. During the authentication process, when incoming user inputs align with the configured rules, the policy undergoes evaluation, and the obligations are presented in the result section of the policy.

     Perform the following steps to add rules to a policy:
     a. Click **+Add Rule**.
        The **Add Rule** page is displayed.
     b. Enter a **Description** of this rule.
     c. Each rule type fulfills a unique role within the authentication and access control framework, allowing administrators to create and implement diverse security policies to meet any organizational needs.
        Select the type of rule that you want to create to identify its purpose from the following options:
        - **Authenticate**: This type of rule is used to create conditions that are related to user authentication. The rule defines the criteria that users must meet to successfully authenticate before accessing a resource.
        - **Custom Action**: These rules provide flexibility in implementing custom authentication methods and access control process.
          For example, you could configure "Wealth Management" as a custom action. This action condition is added to test continuous authentication on a web channel, where "Nbruce" logs in and selects wealth management to invoke this condition.
     d. **Priority**: (Optional) Set the priority of the rule ranging from 1 (highest) to 1,000,000 (lowest). Decimals can be used for finer control. For more information on how priority influences an Authentication Policy result, see Setting priority in Authentication Policy and Rule.
     e. Click **Next**.
        The **rule** page is displayed where you are required to add conditions.

f. Click **Add Conditions** drop-down.
The drop-down list displays the supported conditions. You can select single, multiple, or all the conditions to evaluate an authentication request. Conditions express the relationship of an attribute to a value. For example, if you specify "action" as one of the attributes, then the value could be "login".

g. Select **conditions** based on your business requirement, and click **Add**.

- **IP Address**
  Define the IP addresses or a range of IP addresses from which the user must access the application.
  - **Select file to upload**: Use this option to upload a comma-separated values (CSV) file with IP Addresses or ranges.
    - **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from specified IP addresses or ranges to fulfill this condition.
      - **Value is within configured IP address or range** - Selecting this option implies that users coming from these IP addresses or ranges should be subjected to this condition to discover the Identity Provider.
      - **Value is outside configured IP address or range** - Selecting this option implies that users coming from external IP addresses or ranges outside the configured IP addresses are subjected to this condition to discover the Identity Provider.
  - **Enter manually**:
    a. **IP Address or Range**: Enter the IP Address or a range of IP addresses.
    Once you enter the details, click **Add**.
    The IP Addresses or the range of IP addresses that you entered gets displayed in a tabular form right after it. The table displays the **IP Address** and **Type**.
    To delete the IP Address or range of IP addresses that you entered, click the X button next to it.
    b. **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from specified IP addresses or ranges to fulfill this condition.
      - **Value is within configured IP address or range** - This option implies that users coming from these IP addresses or ranges should be subjected to this condition to discover the Identity Provider.
      - **Value is outside configured IP address or range** - This option implies that users coming from external IP addresses or ranges outside the configured IP addresses should be subjected to this condition to discover the Identity Provider.
- **Channel**
  This condition defines the various platforms through which users access the system.
  Select the channel that must be included in this condition from the following options:
  - **Web**: users accessing the system through a web browser
  - **Mobile**: users use a dedicated mobile application to access the system on their smartphones or tablets.
  - **Custom Channel**: This option allows you to define a specialized channel. For instance, it could be used for users accessing the system through a specific device or authentication method, such as a FIDO (Fast Identity Online) device.

  You can select one or more channels, based on your requirement.
  - **Apply condition if**: Select one of the following conditions from the drop-down list to allow users coming from any of these specified channels to fulfill this condition:
    - **Value matches any selected channel** - This option implies that users coming from these channels should be subjected to this condition to discover the Identity Provider.
    - **Value does not match any selected channel** - This option implies that users coming from channels that are not configured should be subjected to this condition to discover the Identity Provider.
- **Context**
  Build an expression based on various context values to define the condition to trigger this rule.
  - Enter an expression.
    Example of an expression that can be used in the IDP context:
    ```
    ( ${idpmetadata.environment} eq ${clientcontext.env} )
    ```

This example denotes that the expression evaluates the environment of the client context and verifies whether the environment that is specified in the IDP metadata aligns with that of the client context to identify the Identity Provider.

h. Click **Next**.
The **Result** page is displayed.

i. On the **Result** page, specify the outcome of the policy. You must configure the risk rules applicable for this policy that should be evaluated when Policy conditions are met.
   - Select the ***result*** for this rule: You can either enable or disable the rules, and update the default values for the rules for the selected risk policy.

     NOTE
     To know more about these rules, see Understanding Risk Rules.
     - **Behavior Anomaly**
     - **Device Recognition**
     - **Inactive User**
     - **Previous Event is Confirmed Risk**
     - **User Velocity**
     - **Work Day Time**
     - **Enhanced Difficult Travel**
     - **Risky Country**
     - **Fraud IP Address**
     - **Risky IP Address**
     - **Allowed IP Address**
     - **Recognized Computer**
     - **Country Change**
     - **IP Change**

j. Click **Next**.
The **Summary for Authentication Rule** page is displayed. In the Summary for Authentication Rule page, the details of the policy that you have created gets displayed. The page shows the conditions that are applied, the results selected for the policy.

k. To edit the **Conditions**, **Result**, click **Edit** that is displayed next to these fields. This step takes you to the respective pages for modifications.

l. Click **Done**.

m. **Save & Activate**: You can perform the following actions:
   - **Save Policy**: Selecting this option saves your policy. The policy is displayed in the Policy dashboard in the Inactive state.
   - **Inactive Status**: Select this option to change the status of the policy from **Inactive** to **Active** state. Only active policies are evaluated during the authentication process.

Importing a Policy

You may choose to import a previously created policy. Only JSON-format files can be imported into the admin console.

To import a policy, perform these steps:

1. Click **+Import**.
   Select a JSON file from your system and click **Open**.

   NOTE
   If the policy is not saved in the supported format, the system displays an error message- "Unknown Property: info or Invalid JSON."

   After the policy is successfully imported, you are displayed with a confirmation message stating "The policy is imported successfully".

You can access the imported policy on the Policy page.

Reference Links

- Creating Risk Policy using REST API

Identity Providers

Learn how to configure Identity Providers using IDSP Admin Console.

> **NOTE**
> The admin console experience is tailored to match the current privileges of each administrator. Therefore, to access and modify information within the console, you must possess the necessary privileges. You are currently viewing this page because you hold the `urn:iam:t.idpconfigs` privilege.

Configuring an Identity Provider in IDSP using the Admin Console

Follow these steps:

1. Login to the Admin Console.
2. On the left panel of the Admin Console, hover to the icon, and click on the **Identity Providers** page.
   The Authentication Policies page is displayed.
   The Identity Providers page is displayed, where you may find identity providers that have previously been configured, if any, and view further information like the IDP Name, Description, Type, and Status of each identity provider in a tabular format. In case, there are no previously created identity providers, the table displays this message, "There are no identity providers to display".
3. To create a new identity provider, click **+ Add** icon on the top right of the page.
   The Add Identity Provider page is displayed.
4. Select an identity provider type from the options displayed:
   - ***SAML (Import Metadata)***: This option lets you use an XML metadata file to configure an Identity Provider that enables Single Sign On (SSO) using security Assertion Markup Language (SAML)
   - ***SAML (Manual Entry)***: This option lets you manualy configure an Identity Provider that enables Single Sign On (SSO) using security Assertion Markup Language (SAML)
   - ***OpenID Connect***: This option lets you manualy configure an Identity Provider that enables Single Sign On (SSO) using OpenID Connect (OIDC).

   Click **Next**.
   The newly created Id store configuration page is displayed with the ID Store name, description and type. Based on the type of identity provider selected, choose to perform the steps provided in the following sections.

Creating an Identity Provider by Uploading a SAML Metadata

Follow these steps:

1. Enter a unique name for the Identity Provider.
2. In the ***Details*** section, enter the the following information:
   a. **Domain**: Enter the domain name of the Identity Provider.
   Example: example.com
   b. Configure the following ***General*** settings:
      a. **Issuer**: Enter the issuer of the token in the response sent by the Identity Provider.
      b. **Identity Subject Claim**: Enter the claim or assertion consisting of the Authenticated Subject in the response of the Identity Provider.

- **Identity is in the NameIdentifier element of the subject statement**:
- **Identity is in an attribute element**:

c. **Verify Identity**: Set this option to "ON", if you want the user to identify against an attribute that maps to the User ID in IDSP.
d. **Account Mapping Attribute**: Enter the attribute that is present in the Authentication Hub (SSP) identity store, which will be mapped to the value specified by "idpTokenSubjectClaim" to disambiguate the user in the IDSP identity store.
e. **Verification Certificate Alias**: Enter the alias of the certificate that is to be used to verify assertion or token in response from the Identity Provider.
f. **Import IDP Metadata**: Upload the Base64 encoded Identity Provider's metadata that contains the Identity Provider's information in XML format.

3. Click **Save IDP**.

Creating an Identity Provider Using SAML Protocol

Follow these steps:

1. Enter a unique name for the Identity Provider.
   Adding a description to the Identity Store is optional. Click **Next**.
2. In the ***Details*** section, enter the the following information:
   a. **Domain**: Enter the domain name of the Identity Provider.
   Example: example.com
   b. Configure the following ***General*** settings:
      a. **Issuer**: Enter the issuer of the token in the response sent by the Identity Provider.
      b. **Service Provider Initiated Request Binding**:
         - **HTTP POST**: Enter the HTTP-POST Binding URL for the SAML Authentication request to be sent to the Identity Provider.
         - **HTTP Redirect**: Enter the the HTTP-REDIRECT Binding URL for the SAML Authentication request to be sent to the Identity Provider.
      c. **Identity Provider Login URL**: Enter the unique login URL representing the Identity Provider resource instance.
      d. **Identity Subject Claim**: Enter the claim or assertion consisting of the Authenticated Subject in the response of the Identity Provider.
      Select either of the following:
         - **Identity is in the NameIdentifier element of the subject statement**: Select this option if you want the Name ID to be the identity of the subject statement.
         - **Identity is in an attribute element**: Select this option if you want to specify an attribute that holds the identity of the subject statement.
      e. **Verify Identity**: Set this option to "ON", if you want the user to identify against an attribute that maps to the User ID in IDSP. If you set this option to "ON", you must set the attribute mapping in the "Account Mapping Attribute" field.
      f. **Account Mapping Attribute**: Enter the attribute that is present in the Authentication Hub (SSP) identity store, which will be used to map to the value specified in the "Identity Subject Claim" to disambiguate the user in the IDSP identity store.
      g. **Verify Signature**: Enter the alias of the certificate that is to be used to verify assertion or token in response from the Identity Provider.
         - In **Verification Certificate Alias** do the following:
            - Click **Select Certificate** to search for a certificate using alias.
            - You can use the filter to search for a certificate using filters such as, Attributes, Operator, Value, Match Filters with **All** or **Any**.
            - If certificate is not available, click on **Import** to upload a new certificate.

h. **Token Decryption Source**: Select the source from which the token should be decrypted.
Select one of the following:
- **None**: Select this option if you do not want to provide any encryption certificate for the tokens.
- **Tenant**: Select this option if the certificate corresponding to the tenant signing key is provided for encryption.
- **Identity Provider**: Select this option if you want to specify the use of the alias of the secret (of type rsakey), and provide the corresponding certificate for encryption.
You are required to select the decryption key from the drop-down list. If decryption key is unavailable in the drop-down, import one from the secrets store.

i. **Allow Subject in Request**: Set this option to "ON", if you want the identity provider to honor the "Subject" in the Authentication request. The identity provider will use the "Subject" as the user id when requesting authentication from IDSP. If the "Allowed Subject in Request" parameter is set to "ON," the value of "Account Mapping Attribute" must be defined. This is because the value of "Account Mapping Attribute" is searched up in the user profile and used as the Subject in the request sent to the identity provider. By default this option is set to "OFF".

NOTE

- If the "Allowed Subject in Request" parameter is set to "OFF, then defining the "Account Mapping Attribute" is optional as no subject will be sent in the request.

j. **Zero Footprint**: When enabled, this option simplifies the integration proccess by not requiring ID Store connection between VIP Authentication Hub and user directories. It also minimizes dependency on having multiple external identity stores
Set this option to one of the following:
- **None**: When set to "none," the ZFP behavior is determined by the "zeroFootprint" property configured on the application.
- **True**: If set to "true," it is assumed that there is no local footprint of the authenticated identity.
- **False**: If set to "false," it is assumed that there is a local footprint of the authenticated identity.

k. **JIT User Provisioning**: Configure whether to enable or disable Just-In-Time user provisioning.

c. Enter the ***Metadata*** name/value pairs for the Identity Provider in the **Name** and **Value** fields provided.
- To add multiple name/value pairs, click +**Add**.
An empty row is added to the table where you can add the name of the attribute and its value in the corresponding filed. The following OOTB metadata are provided in the console:
  - To add multiple name/value pairs, click +**Add**.
  An empty row is added to the table where you can add the name of the attribute and its value in the corresponding filed. The following OOTB metadata are provided in the console:
  - Select one of the following metadata names from the **Name** drop-down list:
    - **Ensure group membership in specific groups (group1,group2,...)** (API reference - urn:iam:idlcm:assignJitGroupMembership) - Grant group membership in given groups
    - **Allow local updates to synced user profile (true/false)** (API reference - urn:iam:idlcm:allowJitIdentityLocalUpdates) - Indicates whether to allow local updates to JIT created user profile.
    - **Assume primary credentials in identity assertion are verified (true/false)** (API reference - urn:iam:idlcm:saveJitPrimaryCredentialsAsVerified) - Indicates whether primary email/phone/address in identity assertion are assumed to be verified.
    - **Custom query string parameters in login request (a=b&c=d&...)** (API reference - urn:iam:idp:queryparams:login) - Allows to add additional query string params on the authentication request to this IDP.
    Example: customParam1=123&customParam2=abc

NOTE

This can be used with some IDPs to specify custom URL parameters understood by those IDPs. For example, loginHint=${idProfile.loginid) to communicate the loginHint to an Okta SAML IDP.

- **Do not show IDP on signin page (true/false)** (API reference - urn:iam:idp:hideOnDiscovery) - Specifies that the identified Identity Provider (IDP) should be excluded from appearing on the sign-in page. (default is false)
- **Ignore URL parameters on AssertionConsumerService URL check (true/false)** (API reference - urn:iam:idp:ignoreUrlParamsOnAcsCheck) - Specifies whether to disregard query parameters when comparing the Assertion Consumer Service (ACS) URL in the SAML authentication request with the ACS in the SAML response. If enabled, the IDP ignores any query parameters in the URL during this comparison process
- **IDP supports silent reauthentication (prompt=none for OpenID or IsPassive="true" for SAML) (true/false)** (API reference - urn:iam:idp:supportsPromptNone) - Indicates whether the IDP supports prompt=none.This determines if the IDP can perform silent re-authentication. (default is false)
- **IDP supports forced reauthentication (prompt=login for OpenID or ForceAuthn="true" for SAML) (true/false)** (API reference - urn:iam:idp:supportsPromptLogin) - Indicates whether the IDP supports prompt=login. This determines if the AMR claim in the IDP's user assertion can be trusted as evidence of re-authentication on the IDP side. (default is false)
- **Require login name resolved from IDP Account Mapping to match login name/hint (true/false)** (API reference - urn:iam:idp:requireLoginNameMatch) - Indicates whether the user subject in the successful authentication response from the IDP, after applying the account mapping rule, matches the identity requested via the request's subject hint. (default is false)
- **For BYOI and JWT-BEARER, construct group context from groups in identity store and identity assertion (true/false)** (API reference - urn:iam:idp:enableIdentityAssertionGroupPropagation) -
  If this metadata tag is set to "true", construct dynamic group context from groups in identity store and identity assertion. Otherwise group context is constructed from identity store groups only.
  This context is made available to policy evaluation through the context rule such as "${idProfile.groups} eq Group1" or user principal condition such as "(groups eq Group1)".
  This context is also available for dynamic substitution in custom claims:
  - To get all groups:
    - ${userProfile.groups} - returns CSV string
    - $[userProfile.groups] - returns MVA claim formatted according to the underlying protocol (SAML/ OpenID)
  - To discover whether user is a member of the given group: ${userProfile.<group name>}. This returns true or false.
- Enter the value of the metadata corresponding to the Name field , in the **Value** field.
- To delete any attribute, click the **X** displayed adjacent to each name/value pair.

d. Configure the following ***Authentication Method Reference Mapping (AMR)***:

- **Use Identity Provider AMR**: Enable this option to specify if a custom AMR claim can be introspected in the response of assertion or token. When enabled, you must enter a value for custom claim.
  - **Custom Claim**: Enter the name of the Custom AMR claim that needs to be looked at in the response of the assertion or token.
  - **Use Custom AMR Mapping**: **Values Mapping** :
    - **Identity Provider AMR Claim** : Enter the Identity Provider AMR claim value. If you select the option as "Any", then any value provided in this field is mapped to the **IDSP AMR.**
    - **IDSP AMR** : Enter the value of the IDSP AMR claim to which the Identity Provider AMR claim value should be mapped.

The following table contains the authentication factors and their mappings with the AMR values in Authentication Hub:

Authentication Factors	***AMR Values***
Password	PWD
SMSOTP, EmailOTP	OTP
Push	PUSH
MobileOTP	MOTP
Biometric	FIDO
Security Key	HWK

- To add multiple AMRs, click +**Add**.
  An empty row is added to the table where you can add the value for another Identity Provider AMR and corresponding IDSP AMR. Repeat this step untill you have completed entering all the AMRs.
- To delete any attribute, click the **X** displayed adjescent to each AMR value.

- **Default Identity Provider AMRs**: Select the default Identity Provider AMRs from the drop-down list. You can select multiple AMRs from the check boxes.
  - To add a new AMR and select that as a default, type the AMR value in the text box and click **Add**. The newly created AMR is displayed in the dropdown.

e. Configure the following fields for federated identities in the ***User Account Management*** section:

- **Claims Mapping**: If you want to specify the mapping between the attributes/claims returned in the response from the Identity Provider and the corresponding attributes in the user profile in Authentication Hub, then click **Add**.
  - **Identity Provider Claims**: Enter the name of the claim present in the token or userinfo obtained from the Identity Provider.
  - **User Profile Attribute**: Enter the corresponding IDSP attributes.
  - **Claim Source**: Select the source of the claim from the drop-down.
    Select one of the following options
    - **Token**: Select this option if the source of the claim is ID token.
    - **Userinfo**: Select this option to specify that the claim is present in the claims obtained from the "userinfo" endpoint.
    - **Static**: Select this option to specify the static value of the claim in the **Identity Provider Claim** field.
  - **User Data Claim**: Enable this option, if you want to add this attribute to the user profile claim in the ID token.
  - **Persist User Data**: This option is only available when the **User Data Claim** option is enabled. When enabled, this User Data claim needs to be persisted.
  - To add multiple name/value pairs, click +**Add**.
    An empty row is added to the table.
  - To delete any attribute, click the **X** displayed adjescent to each name/value pair.
- **Auto enroll Credential**: Enable this option to auto-enroll new user credentials during authentication. You must provide the attributes that you configured in Authentication Hub and the value that it maps to, in the IDP claim. Set this option to "ON", if you want to enable auto enrollment of mandatory factors. By default this parameter is set to "OFF".

f.

3. Click **Save IDP**.

Creating an Identity Provider Using OpenID Connect Protocol

Follow these steps:

1. Enter a unique name for the Identity Provider.
   Adding a description to the Identity Store is optional. Click **Next**.
2. In the ***Details*** section, enter the the following information:
   a. **Domain**: Enter the domain name of the Identity Provider.
   Example: example.com
   b. Configure the following ***General*** settings:
      a. **Discovery URL**: Enter the OpenID discovery URL where the OpenID server publishes its metadata at a well-known URL, typically. **https://server.com/.well-known/openid-configuration**.
      b. **Issuer**: Enter the unique identifier (URL) of the IDP responsible for issuing authentication tokens.
      c. **Identity Subject Claim**: Enter the claim or assertion consisting of the Authenticated Subject in the response of the Identity Provider.
      d. **Identity Subject Claim Source**: Enter the source of the claim, whether it is coming from the ID Token or Userinfo endpoint.
      e. **Authorization Endpoint**: Enter the OIDC endpoint which receives authentication request from Authentication Hub.
      f. **Token Endpoint**: Enter the OIDC endpoint that Authentication Hub uses to request for the Access Token and Identity Token.
      g. **UserInfo Endpoint**: Enter the OIDC endpoint that Authentication Hub uses to obtain the consented claims or assertions about the logged-in user.
      h. **JWKS Endpoint**: Enter the OIDC endpoint that Authentication Hub uses to obtain a JWKS within the public keys that enable clients to validate the JSON Web Token (JWT) issued by this OIDCprovider.
      i. **Requested Scopes**: Enter the scopes that Authentication Hub should use during authentication to authorize access to a user's details.
         - To add multiple name/value pairs, click +**Add**.
           An empty row is added to the table.
         - To delete any attribute, click the **X** displayed adjescent to each name/value pair.
      j. **Client ID**: Enter the client ID that is obtained from the OIDC provider when the Authentication Hub (SSP) is registered as an application.
      k. **Client Secret**: Enter the client secret that is obtained from the OIDC provider when the Authentication Hub (SSP) is registered as an application.
      l. **Verify Signature**: Enable this to allow Authentication Hub to verify the signature of the token that is sent by the IDP.
      m. **Token Decryption Source**: Select the source from which the token should be decrypted.
      Select one of the following:
         - **None**: Select this option if you do not want to provide any encryption certificate for the tokens.
         - **Tenant**: Select this option if you want to provide the certificate corresponding to the tenant signing key for encryption.
         - **Identity Provider**: Select this option if you want to specify the use of the alias of the secret (of type rsakey), and provide the corresponding certificate for encryption.
           You are required to select the decryption key from the drop-down list. If decryption key is unavailable in the drop-down, import one from the secrets store.
      n. **Verify Identity**: Enable this option to identify the user against an attribute that maps to the User ID in IDSP. If selected, you must set the attribute mapping.
      o. **Account Mapping Attribute**: Enter the attribute that is present in the Authentication Hub (SSP) identity store, to be mapped to the value specified by "idpTokenSubjectClaim" to disambiguate the user in the IDSP identity store.

p. **Allow Subject in Request**: Enable this option to allow the Subject value to be sent in the authentication request sent to the IDP. Default value- False.
   - Enable: If enabled, the IDP uses the Subject as the user ID when requesting authentication from Authentication Hub. You must provide the Identity Mapping attribute.
   - Disable: If disabled, defining the Identity Mapping attribute is optional as no subject will be sent in the request.

q. **Zero Footprint**: When enabled, this option simplifies the integration proccess by not requiring ID Store connection between VIP Authentication Hub and user directories. It also minimizes dependency on having multiple external identity stores
   Set this option to one of the following:
   - **None**: When set to "none," the ZFP behavior is determined by the "zeroFootprint" property configured on the application.
   - **True**: If set to "true," it is assumed that there is no local footprint of the authenticated identity.
   - **False**: If set to "false," it is assumed that there is a local footprint of the authenticated identity.

c. Configure how to secure the communications with the Identity Provider OpenID Endpoints in the ***Connection Management*** section:
   - **Allow SSL Validation**: Enable this option to use SSL to validate communication to the IDP OIDC endpoints. If enabled, you must provide a SSL certificate to use for validation.
     - **SSL Mode**:
       - **Verify Certificate Authority**: This parameter specifies whether the SSL certificate provided by the server should be verified against a trusted Certificate Authority (CA).
       - **Verify Identity**: This parameter specifies whether the identity of the server, as presented in the SSL certificate, should be verified.
     - **SSL Certificate**:
       - Click **Select Certificate** to search for a certificate using alias.
       - You can use the filter to search for a certificate using filters such as, Attributes, Operator, Value, Match Filters with **All** or **Any**.
       - If certificate is not available, click on **Import** to upload a new certificate.
   - **Connection Timeout**: Set the time in miliseconds that will be used when establishing the initial connection between the IDP and the Authentication Hub (ssp). Default Value = "10000".
   - **Read Timeout**: Set the time in miliseconds that will be used when waiting to read data from the IDP. Default Value = "10000".

d. Enter the ***Metadata*** name/value pairs for the Identity Provider in the **Name** and **Value** fields provided.
   - To add multiple name/value pairs, click +**Add**.
     An empty row is added to the table where you can add the name of the attribute and its value in the corresponding filed. The following OOTB metadata are provided in the console:
   - Select one of the following metadata names from the **Name** drop-down list:
     - **Ensure group membership in specific groups (group1,group2,...)** (API reference - urn:iam:idlcm:assignJitGroupMembership) - Grant group membership in given groups
     - **Allow local updates to synced user profile (true/false)** (API reference - urn:iam:idlcm:allowJitIdentityLocalUpdates) - Indicates whether to allow local updates to JIT created user profile.
     - **Assume primary credentials in identity assertion are verified (true/false)** (API reference - urn:iam:idlcm:saveJitPrimaryCredentialsAsVerified) - Indicates whether primary email/phone/address in identity assertion are assumed to be verified.
     - **Custom query string parameters in login request (a=b&c=d&...)** (API reference - urn:iam:idp:queryparams:login) - Allows to add additional query string params on the authentication request to this IDP.
       Example: customParam1=123&customParam2=abc

NOTE

This can be used with some IDPs to specify custom URL parameters understood by those IDPs. For example, loginHint=${idProfile.loginid) to communicate the loginHint to an Okta SAML IDP.

- **Do not show IDP on signin page (true/false)** (API reference - urn:iam:idp:hideOnDiscovery) - Specifies that the identified Identity Provider (IDP) should be excluded from appearing on the sign-in page. (default is false)
- **Ignore URL parameters on AssertionConsumerService URL check (true/false)** (API reference - urn:iam:idp:ignoreUrlParamsOnAcsCheck) - Specifies whether to disregard query parameters when comparing the Assertion Consumer Service (ACS) URL in the SAML authentication request with the ACS in the SAML response. If enabled, the IDP ignores any query parameters in the URL during this comparison process
- **IDP supports silent reauthentication (prompt=none for OpenID or IsPassive="true" for SAML) (true/false)** (API reference - urn:iam:idp:supportsPromptNone) - Indicates whether the IDP supports prompt=none.This determines if the IDP can perform silent re-authentication. (default is false)
- **IDP supports forced reauthentication (prompt=login for OpenID or ForceAuthn="true" for SAML) (true/false)** (API reference - urn:iam:idp:supportsPromptLogin) - Indicates whether the IDP supports prompt=login. This determines if the AMR claim in the IDP's user assertion can be trusted as evidence of re-authentication on the IDP side. (default is false)
- **Require login name resolved from IDP Account Mapping to match login name/hint (true/false)** (API reference - urn:iam:idp:requireLoginNameMatch) - Indicates whether the user subject in the successful authentication response from the IDP, after applying the account mapping rule, matches the identity requested via the request's subject hint. (default is false)
- **For BYOI and JWT-BEARER, construct group context from groups in identity store and identity assertion (true/false)** (API reference - urn:iam:idp:enableIdentityAssertionGroupPropagation) -
  If this metadata tag is set to "true", construct dynamic group context from groups in identity store and identity assertion. Otherwise group context is constructed from identity store groups only.
  This context is made available to policy evaluation through the context rule such as "${idProfile.groups} eq Group1" or user principal condition such as "(groups eq Group1)".
  This context is also available for dynamic substitution in custom claims:
  - To get all groups:
    - ${userProfile.groups} - returns CSV string
    - $[userProfile.groups] - returns MVA claim formatted according to the underlying protocol (SAML/ OpenID)
  - To discover whether user is a member of the given group: ${userProfile.<group name>}. This returns true or false.
- Enter the value of the metadata corresponding to the Name field , in the **Value** field.
- To delete any attribute, click the **X** displayed adjacent to each name/value pair.

e. Configure the following ***Authentication Method Reference Mapping (AMR)***:

- **Use Identity Provider AMR**: Enable this option to specify if a custom AMR claim can be introspected in the response of assertion or token. When enabled, you must enter a value for custom claim.
  - **Custom Claim**: Enter the name of the Custom AMR claim that needs to be looked at in the response of the assertion or token.
  - **Use Custom AMR Mapping**: Select this option to use a set of custom claims and mappings instead of the standard claims. If selected, you must provide the claim and attribute mappings.
    Values Mapping : Enter the IDP AMR attribute and set the AMR attribute that it maps to in Authentication Hub for each claim value that you use to authenticate a user.
    - **Identity Provider AMR Claim** : Enter the Identity Provider AMR claim value. If you select the option as **Any**, then any value provided in this field is mapped to the **IDSP AMR.**
    - **IDSP AMR** : Enter the value of the IDSP AMR claim to which the Identity Provider AMR claim value should be mapped.

The following table contains the authentication factors and their mappings with the AMR values in Authentication Hub:

Authentication Factors	***AMR Values***
Password	PWD
SMSOTP, EmailOTP	OTP
Push	PUSH
MobileOTP	MOTP
Biometric	FIDO
Security Key	HWK

- To add multiple AMRs, click +**Add**.
  An empty row is added to the table where you can add the value for another Identity Provider AMR and corresponding IDSP AMR. Repeat this step untill you have completed entering all the AMRs.
- To delete any attribute, click the **X** displayed adjescent to each AMR value.

- **Default Identity Provider AMRs**: Select the default Identity Provider AMRs from the drop-down list. You can select multiple AMRs.
  - To add a new AMR and select that as a default, type the AMR value from the drop-down list and click **Add**. The newly created AMR is displayed in the dropdown.

f. Configure the following fields for federated identities in the ***User Account Management*** section:

- **Claims Mapping**: If you want to specify the mapping between the attributes/claims returned in the response from the Identity Provider and the corresponding attributes in the user profile in Authentication Hub, then click **Add**.
  - **Identity Provider Claims**: Enter the name of the claim present in the token or userinfo obtained from the Identity Provider.
  - **User Profile Attribute**: Enter the corresponding IDSP attributes.
  - **Claim Source**: Select the source of the claim from the drop-down.
    Select one of the following options
    - **Token**: Select this option if the source of the claim is ID token.
    - **Userinfo**: Select this option to specify that the claim is present in the claims obtained from the "userinfo" endpoint.
    - **Static**: Select this option to specify the static value of the claim in the **Identity Provider Claim** field.
  - **User Data Claim**: Enable this option, if you want to add this attribute to the user profile claim in the ID token.
  - **Persist User Data**: This option is only available when the **User Data Claim** option is enabled. When enabled, this User Data claim needs to be persisted.
  - To add multiple name/value pairs, click +**Add**.
    An empty row is added to the table.
  - To delete any attribute, click the **X** displayed adjescent to each name/value pair.
- **Auto enroll Credential**: Enable this option to auto-enroll credentials. If enabled, new user credentials are also provisioned during authentication. You must provide the attributes that you configured in Authentication Hub and the value that it maps to, in the IDP claim

g. **Authentication Context Class Reference (ACR) Mapping**:

3. Click **Save IDP**.

Using Identity Federation to Access Admin Console

This topic explains how to use an identity federation to gain access to the Admin Console of Symantec Identity Security Platform through the OpenID and SAML protocols.

This topic includes the following sub-topics:

- Configuring IDSP as a Relying Party
- Creating Identity Provider Definition
- Updating the Authorization Policy
  - Performing POST operation on Authorization Policy
- Accessing the Admin Console and Initiating Federation

Step 1: Configuring IDSP as a Relying Party (RP) or a Service Provider (SP) for external IDP

Objective	Configure OpenID RP or SAML SP on the Identity Provider (Configure the external IDP with the following values:)	Configure Artifacts generated by IDP (Copy these values from IDP to use it when configuring the "/IdentityProvider" API in IDSP)
Set up Admin Console as an ***OpenID Relying Party*** App	**Redirect URL** • If you are using a vanity host:`https://<vanity_host>/common/oauth2/v1/rp/callback` • If you are not using a vanity host:`https://<AH_HOST>/<tenant_name>/oauth2/v1/rp/callback` **Allowed Grant Type** • Authorization Code **Groups** • Configure "groups" claim for IDP to send back a CSV value representing user groups through Identity Token. These groups are mapped to Authorization policies in the IDSP. When the OpenID relying Party app is configured, the OpenID provider generates the ***Client ID*** and ***Client Secret.***	**RP properties needed for integration** • Client ID and Secret **IDP properties needed for integration** • Issuer • Authorize Endpoint • Token Endpoint • Userinfo Endpoint (if required by IDP to send "groups") • JWKS Endpoint (if available, to be used for validating the IT signature)
Set up Admin Console as a ***SAML Service Provider*** App	**Assertion Consumer Service URL** • If using a vanity host: `https://<vanity_host>/common/saml/v1/sp/acs` • If not using a vanity host: `https://<AH_HOST>/<tenant_name>/saml/v1/sp/acs` **Entity Id (issuer)** • If using a vanity host: `https://<vanity_host>/common` • If not using a vanity host:`https://<AH_HOST>/<tenant_name>` **Groups** • Configure the "groups" attribute statement for IDP to send back a CSV value representing user groups through the SAML assertion. These groups are mapped to Authorization policies in IDSP.	**IDP SAML Metadata needed for integration** • Login URL • Issuer/Entity Id • Request Binding (possible values): – urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST – urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect • Signing certificate • Encryption certificate

Step 2: Creating Identity Provider definition in IDSP

Create a OpenID provider using the ***/IdentityProvider API*** in IDSP. Remember to configure the following parameters in the specified manner:

- ***IDP Domain Name*** - custom IDSP identifier naming the IDP. This identifier is used to request a federated sign-in from external IDP(<idp-domain-name>\[username-at-idp]).
- ***verifySignature*** - controls whether the IDSP validates the IDP-returned token signature. To get the end to end working, set this parameter to "false".
- ***ignoreSSLValidation*** - controls whether the SSL connection is validated by IDSP. To get the end to end working, set this parameter to "true".
- ***zeroFootPrint*** - set this parameter to "true" as the Admin Console can be virtualized without a local identity store. To authorize access, the claim "group" of IDP is used instead.
- ***validateIdpMappingRequired*** - ignored in this case as ZFP mode is used.
- ***customAmrMappings*** - configurable to map any AMR claimed by an IDP token into the"PWD" AMR of IDSP, hence satisfying the out-of-the-box Authentication Policy that protects the Admin Console. This parameter can be updated at any moment to meet the security requirements of the Admin Console.

NOTE
When you are using the APIs mentioned in the examples, remember to use either of the following base URL:

- If you are using a vanity host, the <base-url> =`https://<vanity_host>/common`
- If you are not using a vanity host, the <base-url> =`https://<AH_HOST>/<tenant_name>`

Example of ***OpenID*** Provider

NOTE
In the following example, the mention of "***step 1"*** is referred to the content in this section,- "Configuring IDSP as a Relying Party (RP) or a Service Provider (SP) for external IDP" (see, Set up Admin Console as an ***OpenID Relying Party*** App table column)

```
Request:

POST <base-url>/admin/v1/IdentityProviders
{
      "idpName": "FedIDP-OpenId",
      "idpType": "openid",
      "domainName": "<idp-domain-name>",
      "idpTokenSubjectClaim": "preferred_username",
      "idpSubjectClaimSource": "token",
      "spMappingAttribute": "email",
      "subjectAllowedInRequest": true,
      "oidAuthorizeEndpoint": "<Authorize Endpoint from Step 1>",
      "oidTokenEndpoint": "<Token Endpoint from Step 1>",
      "oidUserInfoEndpoint": "<UserInfo Endpoint from Step 1>",
      "oidJwksEndpoint": "<JWKS Endpoint from Step 1>",
      "oidRequestedScope": "openid profile phone",
      "oidClientId": "<Client ID from Step 1>",
      "oidClientSecret": "<Client Secret from Step 1>",
      "oidIgnoreSSL": true,
      "oidSignatureValidationStrategy": "jwks",
      "verifySignature": false,
      "validateIdpMappingRequired": true,
      "useCustomAmrClaim": false,
      "customAmrClaim": "amr",
```

```
        "customAmrMappings": {
            "any": "PWD"
        },
        "idpAmrAuthoritativeForPolicyObligations": true,
        "defaultAmr": null,
        "idpClaimsMapping": [
            {
                "idpClaimName": "<claim-name-containing-groups-csv>",
                "idpClaimSource": "token",
                "profileLogicalAttrName": "groups",
                "userData": false
            }
        ],
        "issuer": "<issuer>",
        "status": "active",
        "description": "<description>",
        "zeroFootPrint": "true",
        "ignoreSSLValidation": true,
        "readTimeoutMillis": "10000",
        "connectTimeoutMillis": "10000"
}
```

Example of ***SAML*** Provider

NOTE
In the following example, the mention of "***step 1"*** is referred to the content in this section,- "Configuring IDSP as a Relying Party (RP) or a Service Provider (SP) for external IDP" (see, Set up Admin Console as a ***SAML Service Provider*** App)

```
Request:

POST <base-url>/admin/v1/IdentityProviders
{
        "idpName": "FedIDP-SAML",
        "idpType": "saml",
        "domainName": "<idp-domain-name>",
        "samlMDidpPostBindingURL": "<Login URL from Step 1 , depending on the Request Binding>",
        "samlMDidpRedirectBindingURL": "<Login URL from Step 1 , depending on the Request Binding>",
        "samlAuthnRequestBinding": "<Request Binding from Step 1>",
        "idpTokenSubjectClaim": "NameID",
        "idpSubjectClaimSource": "token",
        "spMappingAttribute": "email",
        "subjectAllowedInRequest": false,
        "verifySignature": false,
        "validateIdpMappingRequired": false,
        "useCustomAmrClaim": false,
        "customAmrClaim": null,
        "customAmrMappings": {},
        "idpAmrAuthoritativeForPolicyObligations": false,
        "defaultAmr": null,
        "idpClaimsMapping": [
            {
                "idpClaimName": "<attribute-name-containing-groups-csv>",
                "idpClaimSource": "token",
```

```
            "profileLogicalAttrName": "groups",
            "userData": false
        }
    ],
    "autoInlineEnrollment": false,
    "issuer": "<Issuer/Entity Id from Step 1>",
    "status": "active",
    "description": <description>,
    "zeroFootPrint": "true",
    "ignoreSSLValidation": true
}
```

Step 3: Updating the authorization policy with the user group representing privilege to use the Admin Console

Perform a GET operation using the following API to get the Authorization Policy protecting the Admin Console:

```
GET <base-url>/admin/v1/AdminRolesDelegator?filter=policyName eq TenantAdminPolicy

Response:
[
  {
    "policyId": "1ccf9742-a066-4941-b719-2fca53802631",
    "policyName": "TenantAdminPolicy",
    "description": "Policy for Tenant Admin",
    "status": "active",
    "apps": [
        {
            "name": "SSP",
            "id": "$SSP_APP"
        }
    ],
    "rules": [
        {
            "id": "29ad3a18-02a5-47a5-acfa-bc76743bf73e",
            "conditions": {
                "principal": {
                    "group": {
                        "operator": "in",
                        "value": [
                            "Admin"
                        ]
                    },
                    "clientApp": {
                        "operator": "in",
                        "value": [
                            "$TENANT_ADMINCONSOLE",
                            "67734fbc-a359-46b7-9e49-19edcb44cca1"
                        ]
                    },
                    "user": {
                        "operator": "in",
                        "value": [
                            "malbail",
                            "kevinlucas",
```

```
                            "nbruce"
                        ]
                    }
                }
            },
            "result": {
                "msg": "This is Tenant Admin authorization policy",
                "privileges": [
                    "$TENANT_ADMIN_PRIVILEGES"
                ]
            }
        }
    ],
    "createdDate": "2022-09-23 09:36 PM UTC",
    "updatedBy": {
        "principalType": "CLIENT",
        "principalId": "6da2992f-0a6a-4571-83cc-1e7934d0dfc4",
        "principalName": "DefaultTenantClient"
    },
    "updatedDate": "2022-12-20 06:26 PM UTC",
    "preventedOperations": "Delete"
  }
]
```


Performing POST operation to grant access to the groups to use the Admin Console

Perform the patch operation by using the policyId and ruleId received from the get request in the previous step. This operation grants one or more IDP-specified groups access to the Admin Console.

Perform POST using the following API:

```
POST <base-url>/admin/v1/AdminRolesDelegator/1ccf9742-a066-4941-b719-2fca53802631
```

Request:


```
{
  "id": "1ccf9742-a066-4941-b719-2fca53802631",
  "principal": {
      "user": {
          "operator": "in",
          "value": [
              "droth"
          ]
      }
  }
}
```


Step 4: Accessing the Admin Console and Initiating Federation

Access the admin console using the following URL:

```
<base-url>/ui/v1/adminconsole
```

Sign-in using the <idp-domain-name> \[username-at-idp]

- Where,<idp-domain-name> refers to the "domainName" property of the ".../IdentityProviders" resource created here.

You are then redirected to the external Identity Provider to complete authentication, and subsequently back to the Admin Console. Be aware that depending on the IDP's configuration, you may be prompted to enter your username even if it was initially provided.

NOTE

- Authentication Policy governing signing to the Admin Console may need to be adjusted to align with Admin Console security requirements, where the AMR configuration of the IdentityProvider resource that was created here have to be adjusted.
- To achieve a seamless federated sign-in to the Admin Console after external IDP authentication (and if AMR configurations match), MITM protection must be enabled. Without it, the user will still be prompted for credentials.

Certificates

The **Certificates** dialog allows you to manage public keys or certificates (certificates) of other parties for token validation, token encryption, and SSL communication.

NOTE
The admin console experience is tailored to match the current privileges of each administrator. Therefore, to access and modify information within the console, you must possess the necessary privileges. You are currently viewing this page because you hold the `urn:iam:t.certs` , and `urn:iam:t.configs` privilege.

Certificates Dialog

Use the **Certificates** dialog to perform the following operations:

- View all certificates
- Search for a certificate
- Import a certificate
- Update a certificate
- Delete a certificate

The Certificates dialog contains the following sections and fields:

- **Search and Filter**
  Narrows down the certificates that are displayed in the Certificates dialog. You can filter your search using the alias or the description of a certificate. Type the complete or part of an alias or description, and click **Submit**.
  Applying Filter
  1. Click **Show Filters** to open the filter options if it is hidden.
  2. Select the attribute from the dropdown with which you want to search the required certificates.
  3. Set the operator.
  4. Provide a specific value.

  You can click (+) to add multiple filters if necessary or remove existing ones by clicking "X" next to each filter.
  Matching Filters
  To apply filters and get the search results for the certificates that match your criteria, perform the following steps:
  1. On the top right of the page, the "**Match filters**" dropdown is available.
     Select whether you want to match either option:
     – "All" filters (return results that match all specified conditions)
     – "Any" filters (return results that match at least one condition)
  2. Click **Submit**.

Clear Filters
Use this option if you want to reset your search and filters.
– Click **Clear Filters** to remove all applied conditions and start a new search.
This option allows you to narrow down the application search results using specific criteria and get the most relevant results efficiently.

- **A table that lists the available certificates**
Displays the list of certificates that are imported into Symantec Identity Security Platform using Admin Console in a table.
The table displays the following columns:
– **ALIAS**
Indicates the alias of a certificate.
– **DESCRIPTION**
Indicates the description about a certificate.
– **EXPIRATION**
Indicates the expiry date and time of a certificate.
– **STATE**
Indicates the validity of a certificate.
– **TOKEN ISSUER**
Indicates the iss claim value.
– **UPDATED**
Indicates when a certificate was imported or updated.
The table also allows you to perform the following operations on a certificate using the vertical ellipsis (⋮):
– **Delete**
Deletes the selected certificate. You can delete a certificate directly if it is not being used by other resources. A reconfirmation is requested before deleting a certificate. Once confirmed, the certificate is deleted. However, if one or more resource types have referenced this certificate, a message is displayed stating that the certificate cannot be deleted because it is still in use. The message also display a list of resource names and resource types that are using the certificate, allowing you to review its usage. The delete is only allowed once all such references are removed.
A helper API is available to retrieve the list of dependent objects in case of a conflict, see ReferencingObjectsHelper API.
– **Update**
Updates the already uploaded certificate with a new certificate. Only the certificate can be updated but not the already entered information such as ALIAS, DESCRIPTION, TOKEN ISSUER, and KEY ID. You can update a certificate directly if it is not being used by other resources. A reconfirmation is requested before updating the certificate. Once confirmed, the certificate is updated. However, if one or more resource types have referenced this certificate, a message is displayed stating that the certificate cannot be updated because it is still in use. The message also display a list of resource names and resource types that are using the certificate, allowing you to review its usage. The update is only allowed once all such references are removed.
A helper API is available to retrieve the list of dependent objects in case of a conflict, see ReferencingObjectsHelper API.
– **Export**
Exports the selected certificate.
- **Import Certificate**
Imports a certificate into the IDSP. This field opens the **Import Certificate** pop-up dialog that contains the following fields:
– **Upload**

Opens the file directory of a system to allow the selection of a certificate. Select only a DER-encoded certificate and click **Open**.

- **Create new alias**
  Imports a certificate using a new alias. Use this option to upload the first certificate into the IDSP or to upload a certificate using a new alias. If this option is selected, click **Next** and complete the following fields:
  - **ALIAS**
    Defines a name for the certificate.
  - **(Optional) DESCRIPTION**
    Specifies a brief description about the certificate.
  - **(Optional) KEY ID**
    When a user is redirected to IDSP for an additional authentication after the completion of the primary authentication, the received ID Token Hint contains the **kid** claim in the JOSE header. Enter the value that is received in the **kid** claim.
  - **(Optional) Token Issuer**
    When a user is redirected to IDSP for an additional authentication after the completion of the primary authentication, the received ID Token Hint contains the **iss** claim that indicates the principal who issued the JWT. Enter the value that is received in the **iss** claim.
  - **Import**
    Imports the selected certificate.
  - **Back**
    Navigates back to the previous screen of the pop-up dialog.
- **Associate with an existing alias**
  Imports a certificate using an existing alias. Use this option to add another certificate to an existing alias. If this option is selected, complete the following fields:
  - **A text field to search for a certificate by alias or description**
    Imports a certificate using an existing alias. Use this option to add another certificate to an existing alias. This is particularly useful for **certificate rotation**, where a new certificate is added alongside an existing one to ensure uninterrupted service during key updates or renewals. If this option is selected, complete the following fields:
    - **(Optional) DESCRIPTION**
      Specifies a brief description about the certificate.
    - **(Optional) KEY ID**
      Defines the key ID of the certificate.
    - **Import**
      Imports the selected certificate.
  - **Back**
    Navigates back to the previous screen of the pop-up dialog.

- **Next**
  Navigates to the next screen of the pop-up dialog to gather information about the selected certificate.
- **Cancel**
  Aborts the certificate import process.

Viewing and Managing Certificates

Once you have added a certificate, it appears on the **Certificates** dialog. The ellipses (**...**) at the end of each row lets you delete, update, or export the certificate. You can also import a new certificate from this page.

To view the full details of the certificate and to manage certificate options, click the name of the certificate. The **Certificate** details page displays the following tabs:

- The **Details** tab displays the full details of the certificate, and lets you perform the following actions:
  - **Delete**: Use this option to delete a certificate. You can delete a certificate directly if it is not being used by other resources. A reconfirmation is requested before deleting a certificate. Once confirmed, the certificate is deleted. However, if one or more resource types have referenced this certificate, a message is displayed stating that the certificate cannot be deleted because it is still in use. The message also display a list of resource names and resource types that are using the certificate, allowing you to review its usage before proceeding with an update or deletion. If the certificate is in use, you have the option to retain it.
  - **Update**: Use this option to update the previously uploaded certificate with a new certificate. Only the certificate can be updated; the existing information, such as ALIAS, DESCRIPTION, TOKEN ISSUER, and KEY ID cannot be updated.
  - **Assign Manager**: Use this option to assign an entity to manage this certificate.
  - **Export**: Use this option to export the selected certificate.
- The **Managers** tab lets you view and modify the entities that can currently manage this identity provider. From the vertical ellipses (**...**) next to each row, perform one of the following actions for that identity provider:
  - Select **Assign** to assign entities that can manage this identity provider.
  - Select **Revoke** to revoke the assignment of entities that can manage this identity provider.
- The **Authorization** tab lets you view and modify the resource authorizations that have been assigned to this identity provider. From this tab, you can add tags to this identity provider. The tag is a name-value pair that you use to assign an attribute to this resource. Once a resource is tagged with a criteria, you can include that criteria in a constraint statement for any policy that affects that resource.

  The format must be `name=value`. However, there are no restrictions on the name or value that you assign.

Secrets

The Secrets dialog allows you to add a separate secret LCM pattern for SCIMconfigs and LDAPconfigs.

> **NOTE**
> The admin console experience is tailored to match the current privileges of each administrator. Therefore, to access and modify information within the console, you must possess the necessary privileges. You are currently viewing this page because you hold either of the following privileges, or both of them:
>
> - `urn:iam:t.secrets`
> - `urn:iam:t.secretsgetter`

A secret is stored in an encrypted form and it is decrypted only when it is needed. Define the type of secret object that is stored as a secret. The secret type has no bearing on the runtime execution, it only helps you associate the secret with a well-known type.

The Secrets dialog contains the following fields:

- **A table that lists the stored secrets**

  Displays the list of secrets that are stored in Symantec Identity Security Platform for an LDAP or an SCIM configuration.

  The table displays the following columns:

  - **ALIAS**

    Indicates the name of a secret.
  - **DESCRIPTION**

    Indicates the description about a secret.
  - **TYPE**

    Indicates whether a secret is stored as a plain-text secret, as a private key that is generated by an RSA algorithm, or as a private key that is generated by an EC algorithm.
  - **JWKS**

For an RSA key or an EC key, indicates whether the public key of the secret is populated in the response of the JWKS endpoint.

The table also allows you to perform the following operations on a secret using the vertical ellipsis (⋮):

- **Delete**
  Deletes the selected secret. You can delete a secret directly if it is not being used by other resources. A reconfirmation is requested before deleting a secret. Once confirmed, the secret is deleted. However, if one or more resource types have referenced this secret, a message is displayed stating that the secret cannot be deleted because it is still in use. The message also display a list of resource names and resource types that are using the secret, allowing you to review its usage. The delete is only allowed once all such references are removed.
  A helper API is available to retrieve the list of dependent objects in case of a conflict, see ReferencingObjectsHelper API.
- **Update**
  Updates the information that is already entered for a secret. All the fields that are available for configuration in an add operation are available for modification. For more information about the fields, see Add New Secret. You can update a secret alias directly if it is not being used by other resources. A reconfirmation is requested before updating the secret. Once confirmed, the secret is updated. However, if one or more resource types have referenced this secret, a message is displayed stating that the secret alias cannot be updated because it is still in use. The message also display a list of resource names and resource types that are using the secret alias, allowing you to review its usage. The update is only allowed once all such references are removed.
  A helper API is available to retrieve the list of dependent objects in case of a conflict, see ReferencingObjectsHelper API.
- **(Only for an RSA key or an EC key) Add to JWKS**
  Adds the public key of the secret in the response of the JWKS endpoint. If this secret signs a token, the third party can authenticate the signature with the public key that is exposed in the JWKS response. Confirm the addition when prompted.

- **Add New Secret**
  Adds a new secret in IDSP. This field opens the **Add New Secret** pop-up dialog that contains the following fields:
  - **Alias**
    Defines the name of the secret.
  - **Description**
    Defines a brief description about the new secret.
  - **Type**
    Indicates whether a secret is stored as a plain-text secret, as a private key that is generated by an RSA algorithm, or as a private key that is generated by an EC algorithm. Select *one* of the following values and complete their relevant fields:
    - **Secret**
      Indicates that the secret must be stored as a plain-text secret. Enter the value of the secret in Secret Value.
    - **RSA Key** or **EC Key**
      Indicates that the secret must be stored as a private key that is generated by an RSA/EC algorithm accordingly. If you select this option, complete the following fields:
      – **Add to JWKS**
        Adds the public key of the secret in the response of the JWKS endpoint. If this secret signs a token, the third party can authenticate the signature with the public key that is exposed in the JWKS response. To enable the addition, move the slider to the right to view **ON**.
        Default Value: OFF
      – **Certificate**

Specifies an RSA/EC certificate that is a DER-encoded X.509 certificate in the .cer or .der format. Click **Upload** to view the file directory of a system that allows the selection of the required certificate. Select the certificate and click **Open**.

  - **Private Key**
    Specifies a private key that is generated by the RSA/EC algorithm. Click **Upload** to view the file directory of a system that allows the selection of the required private key. Select the required private key and click **Open**.
  - **Key Identifier**
    When a user is redirected to IDSP for an additional authentication after the completion of the primary authentication, the received ID Token Hint contains the **kid** claim in the JOSE header. Enter the value that is received in the **kid** claim. IDSP uses this value along with the value of the token issuer for certificate lookup during the verification of a token signature, if the certificate lookup using x5t#256 JOSE header and x5t JOSE header fails

- **Add**
  Adds a secret with the entered information.
- **Cancel**
  Aborts the secret addition process.

Accounts

The Accounts dialog displays the details of all the user accounts that are synchronized with Symantec Identity Security Platform.

NOTE
The admin console experience is tailored to match the current privileges of each administrator. Therefore, to access and modify information within the console, you must possess the necessary privileges. You are currently viewing this page because you hold any or all of the following privileges:

- `urn:iam:t.userdata`
- `urn:iam:t.users`
- `urn:iam:t.accountadmin`

This dialog displays the following fields:

- **A drop-down list to select an identity source**
  Displays the internal ID store and all the configured external ID stores. A user account can exist in the internal ID store that is provided by IDSP, or in an external ID store such as LDAP, SCIM, or any other user directory with users. The user accounts of external ID stores are not synchronized with the internal ID store.
  Each ID store displays the number of user accounts that are available within it.
- **A text field to enter a username**
  Defines the username of the account that you want to view or manage. Enter a complete, valid user ID.
- **Search**
  Provides advanced search capabilities with multiple filter combinations to efficiently locate user accounts. The enhanced system includes:
  - **Advanced Filters:** Apply multiple filter criteria using attributes including First Name, Last Name, Username, IDP User Guid, Service Account, and Universal Id
  - **Filter Operators:** Use flexible comparison operators such as "Starts with", "Contains", "Equals", and "Does not equal" for precise matching
  - **Logical Combinations:** Combine multiple filters using AND/OR logic with the "Match filters" option set to "All" or "Any"

Clicking the retrieved username displays the following tabs:

Status Tab

The status tab allows you to manage the status-related actions of a user account. This tab contains the following fields:

- **User Object ID**
  A unique identifier for the user account. Click the copy icon to copy the ID to your clipboard.
- **Locked**
  Specifies if the user account must be locked after three consecutive login attempts with an incorrect password. Move the slider to the right to view **YES**. To unlock a user account as an administrator, move the slider to the left to view **NO**.
- **Disabled**
  Indicates whether the user account is active or disabled. This flag is managed only using the API. The Admin Console only displays the status of the flag.
- **Service Account**
  Identifies whether this account is a service account. If Service Account is enabled, the Wallet and Service Account tabs are not visible for this user. Although an end user and a manager can enable or disable the service account for a user, only the assigned manager can lock or disable the service account.
- **(Only for the internal ID store) Password Must Change**
  Specifies whether the user must change the sign-in password after successfully logging into the Admin Console. If this flag is set, the user is challenged to change the password before viewing the requested resource. This operation is a one-time activity. The flag is reset to **NO** after the user changes the password when prompted. To enable the flag, move the slider to the right to view **YES**.
- **(Only for the internal ID store) Account Is Read-Only**
  Indicates whether the user account being imported can be modified by IDSP, or if it must be imported as is. This flag is applicable during a bulk import of user accounts into IDSP. To enable the flag, move the slider to the right to view **YES**.
- **Password Changed On**
  Identifies the last time and date that the password was modified.
- **Last Failed Login**
  Identifies the last time and date that authentication was attempted and failed.
- **Last Successful Login**
  Identifies the last time and date that authentication was attempted and succeeded
- **Last Locked Date**
  Identifies the last time and date that the user account was locked.

Credentials

This tab displays the credentials that are enrolled for the user. This tab displays the list of options that can be used during a multi-factor factor, as per the authentication policy configuration.

This tab displays the following categories for each multi-factor authentication factor, only if the user is enrolled to the corresponding authentication factor:

- For using a text: **Phone Numbers for SMS Messages**
- For IVR calls: **Phone Numbers for IVR Calls**
- For using an email: **Email Address for Security Codes**
- For using biometrics: **Biometrics**
- For using a security key: **Security Keys**
- For using a security code: **Mobile Apps for Security Codes**
- For using a push notification: **Mobile Apps for Push Requests**

For example, if the user is enrolled to use a text message for a multi-factor factor, only **Phone Numbers for SMS Messages** is displayed.

For each displayed authentication factor, the following options are displayed:

- **ENABLE**: Specifies whether the corresponding authentication factor must be enabled or disabled for the user. By default, this flag is set to **ON**. To restrict the user from using the authentication factor, move the slider to the left to view **OFF**.
- **TAG:** Allows for differentiation between various types of credentials that are associated with each of the user's phone numbers and email addresses, enabling policies to restrict their use based on these tags (for example, allowing only "business" emails for certain application flows). Just as the LDAP configuration (using mvAttributeMapping) defines the mapping of attributes like mail to a business tag and personalmail to a personal tag for the logical email attribute, the TAG column directly displays these assigned tags.
- **DELETE**: Deletes the authentication factor as a possible multi-factor authentication factor for the user. Click the **delete** icon and confirm the deletion when prompted.

Remembered Devices

This tab displays all the devices on which the user has selected the Remember this device option during a multi-factor authentication on the same device for the same user account. This tab displays the following information of a device:

- Device name in **DEVICE**
- Operating system in **OS**
- Web browser that is used for signing in **BROWSER**
- Last signed in timestamp in **LAST USED**

Known Devices

This tab allows the deletion of all the known devices that are used by a user for signing into the Admin Console. If this list is deleted, the user is marked as risky and a high-risk score is set for the user. To delete the list, click **Delete all known devices for the user** and confirm the deletion when prompted.

Service Accounts

Service accounts are specialized accounts for service level users such as trusted clients. Service accounts do not need access to the Symantec Identity Security Platform consoles and user-facing interfaces. As a result, service accounts are not subject to the same limits that an interactive user would be for authentication session lifetime and credential lifetime. To add or remove a service account for a user, access the user (from the **Accounts** > **User** page or from the **User** page) and enable or disable service accounts for that user.

You must be a manager of the user to otherwise manage the service account (lock/unlock, disable,enable). You can view and lock/disable a service account assigned to a user that you manage from the Service Accounts tab on the **Accounts** > **User** page.

To assign a service account to a manager, enable the service account setting for a user, and then assign the user to a specific manager.

The **Service Account** tab displays the users that are marked as service accounts and which are assigned to this manager. From this tab, you can manage the service account lock and disabled settings for the managed user. This tab can be viewed by the manager of the user for which the Service Account is enabled; this tab is not available to the users for which the service account is enabled.

Diagnostics

The Diagnostics dialog provides administrators with the ability to capture, view, and analyze detailed logs for troubleshooting purposes.

This feature enables real-time log collection tied to specific business processes identified by clientTransactionId correlation parameter, such as identifying issues during user's sign-in attempt.

Prerequisites

To access the Diagnostics feature, you must have one of the following permissions:

- Tenant Administrator role ("urn:iam:t.all" scope)
- Diagnostics Administrator** role ("urn:iam:t.diagnosticsadmin" scope)

Obtaining the Client Transaction Id (As End User)

The Client Transaction Id is a unique identifier that links all log entries for a specific user session, generated on the sign-in page and used to correlate logs across all backend services.

The Client Transaction ID can be obtained from the sign-in page before you attempt to sign in. This is crucial because the Client Transaction ID must be captured and configured in the Diagnostics console before the authentication attempt.

Steps to navigate to the Client Transaction Id in Signin UI

1. Navigate to the application's sign-in page.
2. On the sign-in page, locate the three-dot menu (⋮ or ⋯) found in the bottom right corner of the sign-in form.
3. Click the three-dot menu to reveal the options. Click "Copy Client Transaction ID". An example Id would look like the following:
   `a1b2c3d4-e5f6-7890-abcd-ef1234567890`
4. Save the Id.

   NOTE
   Do NOT proceed with sign-in yet!
   Paste the Client Transaction ID somewhere safe and communicate it to the administrator who is going to configure the diagnostics capture.

Accessing and Enabling the Diagnostic Console (By Tenant Admin)

1. On the Admin Console's left navigation panel, click **Diagnostics**.
2. Toggle to enable the Diagnostics. When enabled, toggle shows "Diagnostics Enabled".

Configuring the Capture

1. Paste the Client Transaction ID (obtained from the sign-in page) into the "Client Transaction ID" input field. Verify the ID is correctly pasted (no extra spaces or characters).
2. Click "Capture Logs".
   A success message will confirm: "Diagnostics capture started at DEBUG level for 15 minutes"
3. The system is now ready to capture logs for this Client Transaction ID. Now you can notify the user to proceed with their sign-in attempt.

Viewing the Logs

After the user completes their action, the logs will be ready for viewing and analysing.

1. Click "View Logs".
   The logs are retrieved and displayed.
2. Each log entry contains the following information:

- **Timestamp**: When the log entry was created (UTC)
- **API**: The API endpoint that was called.
- **Level**: Shows the log severity: ERROR, WARN, INFO, DEBUG.
- **Message**: The log message content
- **User Login ID**: The username attempting authentication
- **App Name**: The application being accessed
- **Transaction ID**: Internal transaction identifier

Auto Refresh

For continuous monitoring, enable the **Auto Refresh every 5s** checkbox to update the logs every 5 seconds. Simply uncheck the box when you are finished troubleshooting.

Profiles

The Profiles dialog displays information about your roles, the roles that you have delegated to others, and what resources have been delegated to you.

You can access the Profiles dialog by hovering over your name at the top right of the page and clicking **Profile**, or by clicking **Profile** in the left navigation bar. The Profiles dialog displays the following information:

- The **My Administrator Roles** tab lists the administrative roles that are assigned to you. From the vertical ellipses (**...**) next to each row, click Delegate to delegate that role to another administrator, group, or application.
- The **Delegations to Others** tab lists the roles that you have delegated to other administrators, groups, or applications. From the vertical ellipses (**...**) next to each row, click Revoke to remove that role from that administrator, group, or application.
- The **Resources Delegated to Me** lists the resources that have been delegated to you. From the vertical ellipses (**...**) next to each row, click Delegate to delegate that role to another administrator, group, or application.

Settings

Allows you to manage admin roles, customize the Signin UI's branding, configure tenant level settings, securely upload, and manage keys used during secured connections, configure Identity Stores where Symantec Identity Security Platform can search for users.

> **NOTE**
> The admin console experience is tailored to match the current privileges of each administrator. Therefore, to access and modify information within the console, you must possess the necessary privileges. You are currently viewing this page because you hold either one of the privileges from the following or of both of them:
>
> - `urn:iam:t.authenticate`
> - `urn:iam:t.configs`

1. On the Admin Console navigation panel, click **SETTINGS**.
   The SETTINGS page is displayed.
2. Select and click the tile you want to navigate to from the following options:
   - Administrators and Roles
   - Branding
   - Service Configuration
   - Identity Stores
   - Integration Setup

Clicking these links redirects you to the respective pages with detailed information.

Administrators and Roles

The Administrators and Roles dialog allows you to add an administrator with a selected pre-defined role/allows you to assign a specific pre-defined role to a new administrator.

> **NOTE**
> The admin console experience is tailored to match the current privileges of each administrator. Therefore, to access and modify information within the console, you must possess the necessary privileges. You are currently viewing this page because you hold the `urn:iam:t.adminroles` privilege.

This dialog displays a tab for Administrators and Roles.

Administrators

The Administrators tab contains the following fields:

- **A table that lists the available administrators**
  Displays the list of administrators that can manage Symantec Identity Security Platform in their assigned role.
  The table displays the following columns:
  - **IDENTITY SOURCE**
    Indicates whether an administrator belongs to a specific LDAP/SCIM configuration, an identity provider, an internal store, or to any of these identity sources.
  - **NAME**
    Indicates the name of an administrator as defined in the corresponding identity source.
  - **ROLE**
    Indicates the role that is assigned to an administrator.
  - **TYPE**
    Indicates whether an administrator belongs to a user (Person), a user group (Group), or an application (Application).

  The table also allows you to perform the following operation on an administrator using the vertical ellipsis (⋮):

  - **Revoke**
    Revokes the role that was assigned to the selected administrator and deletes the administrator from IDSP. Confirm the revocation when prompted.
- **Grant Role**
  Creates an administrator with an assigned role. This field opens the **Grant Role** pop-up dialog that contains the following fields:
  - **A drop-down with the following list of pre-defined roles**:

  - Account Administrator
  - Authentication Client
  - Configuration Administrator
  - Configuration Auditor
  - Credential Administrator
  - Diagnostics Administrator
  - Identity Administrator
  - Identity Client
  - Idstore Administrator
  - Integration Administrator
  - Integration Auditor
  - Risk Client
  - Tenant Administrator
  - Tenant Auditor
  - User Password Administrator

  For information about these roles, see Administrator Roles.
- An option to define whether the administrator belongs to one of the following options:
  - **User Group**
  - **User**
  - **Application Group**
  - **Application**

  Depending on the option selected, select an identity source or an application from the corresponding drop-down list, and enter the name of the user group, user, or application accordingly.

  For a user, you can select all users or a selected user depending on the selected identity source. For an internal identity source, use the search filter to retrieve the required user group or user.
- **Grant**
  Adds an administrator with the granted role.
- **Cancel**
  Aborts the administration creation process.

Roles

The Roles tab contains the following fields:

- **A table that displays the pre-defined set of roles with the following columns**:
  - **NAME**
    Indicates the name of a role.
  - **MEMBERS**
    Indicates the number of administrators that are assigned with this role.
  - **DESCRIPTION**
    Indicates the brief description about a role.
  - **CONSTRAINTS**
    Indicates how many constraints are applied to this role.

  The table also allows you to grant a role using the vertical ellipsis (⋮). For more information, see the description of the **Grant Role** field that follows.
- **Grant Role**
  Creates an administrator with an assigned role. This field opens the Grant Role pop-up dialog that contains the following fields:

- **A drop-down with the following list of pre-defined roles**:
  - Account Administrator
  - Authentication Client
  - Configuration Administrator
  - Configuration Auditor
  - Credential Administrator
  - Diagnostics Administrator
  - Identity Administrator
  - Identity Client
  - Idstore Administrator
  - Integration Administrator
  - Integration Auditor
  - Risk Client
  - Tenant Administrator
  - Tenant Auditor
  - User Password Administrator

  For information about these roles, see Administrator Roles.
- **An option to define where the administrator belongs to**:
  - **User Group**
  - **User**
  - **Application Group**
  - **Application**

  Depending on the option selected, select an identity source or an application, application group from the corresponding drop-down list, and enter the name of the user group, user, or application, application group accordingly.

  For a user, you can select all users or a selected user depending on the selected identity source. For an internal identity source, use the search filter to retrieve the required user group or user.
- **Grant**

  Adds an administrator with the granted role.
- **Cancel**

  Aborts the administration creation process.

- **View Roles**

  Click the role name to view the Role details page. The Role details page contains the following tabs:
  - ADMINISTRATORS

    Identifies the administrators that are assigned this role. You can revoke the role using the vertical ellipsis (⋮).
  - PRIVILEGES

    Lists the privileges that are assigned to this role.
  - CONSTRAINTS

    Identifies the constraints (Name, Description, and Resource) that are applied to this role. You can remove or export a constraint using the vertical ellipsis (⋮).

  This page also lets you grant a role or add a constraint to a role. You must be an administrator who is in the internal user store to view constraint information.

Delegations

The Delegations tab contains the following fields:

- **A table that displays the delegation operations that have been performed**:

- **DELEGATOR**
  Indicates the user, group, or application that made the delegation.
- **RESOURCE**
  Indicates the resource to which the delegation applies.
- **DELEGATED TYPE**
  Indicates what type of entity performed the delegation (user, group, or application).
- **DELEGATEE**
  Indicates the entity to which the resource was delegated (user, group, application).
- **TYPE**
  Indicates the type of entity to which the resource was delegated.

The table also allows you to revoke a delegation using the vertical ellipsis (⋮).

Administrator Roles

The following table lists the pre-defined administrator roles and their associated policies and privileges:

> **NOTE**
> For details about the privileges for each administrator role, see Scopes and APIs Table.

Administrator Role	Policy Name	Purpose	Privileges
Account Administrator	AccountAdminPolicy	The Account Administrator role is granted permission to manage user accounts in IDSP. For example, the Account Administrator can check the user's "Status", and also update the user account status as "Locked" or "Unlocked" in the ID Store.	"urn:iam:t.accountadmin" The privileges that are associated with the ***Account Administrator*** role extend to "***Client***", "***User***" and "***Group***" entities.

Authentication Client	AuthClientPolicy	The Authentication Client role is granted permission to perform operations that are related to authentication to implement a custom Sign-in app. The Authentication Client policy that represents this role is associated with the client called, **SigninClient** which performs authenticate operations for the Sign-in app. The Authentication Client role grants clients the capability to initiate and execute an authentication flow in IDSP. The role does not grant any additional privileges, and the desired purpose is to obtain an access token for the user with their designated scopes. The user may be a tenant administrator or a user with a self-service API. .	"urn:iam:t.authenticate" The privileges that are associated with the ***Authentication Client*** role are restricted to "***Client Only***," with no extension of these privileges to regular "User" and "Group" entities.
Configuration Administrator	ConfigurationAdminPolicy	The Configuration Administrator role lets users oversee tasks that are related to Security Settings, Authentication policies, ID Stores, Tenant Configurations, Certificates, Secrets, Signing Keys, and more.	• "urn:iam:t.configs" • "urn:iam:t.authpolicies" • "urn:iam:t.idstoreconfigs" • "urn:iam:t.certs" • "urn:iam:t.secrets" • "urn:iam:t.signingkey" • "urn:iam:t.idstorebrowser" • "urn:iam:t.adminroles" The privileges that are associated with the **Configuration Administrator** role extend to "***Client***", "***User***" and "***Group***" entities.
Configuration Auditor	ConfigurationAdminPolicy	The Configuration Administrator role lets users oversee various tasks that are related to Security Settings, Authentication policies, ID Stores, Tenant Configurations, Certificates, Secrets, Signing Keys, and more.	• "urn:iam:t.configs" • "urn:iam:t.authpolicies" • "urn:iam:t.idstoreconfigs" • "urn:iam:t.certs" • "urn:iam:t.secrets" • "urn:iam:t.signingkey" • "urn:iam:t.idstorebrowser" • "urn:iam:t.adminroles" The privileges that are associated with the **Configuration Administrator** role extend to "***Client***", "***User***" and "***Group***" entities.

Credential Administrator	CredentialAdminPolicy	The Credential Administrator role is granted permission to oversee various aspects in an organization, including managing user account lock status, credentials, trusted and recognized devices, and overseeing user sessions.	• "urn:iam:t.userdata" • "urn:iam:t.session" The privileges that are associated with the ***Credential Administrator*** role extend to "***Client***", "***User***" and "***Group***" entities.
Diagnostics Administrator	DiagnosticAdministratorPolicy	This role is for internal use only.	"urn:iam:t.diagnosticsadmin"
Identity Administrator	IdentityAdminPolicy	The Identity Administrator role is granted the authority to oversee the management of "users" and "groups" within the internal Identity Store. Also, users with this role have the permission to manage various other aspects such as user account lock status, credentials, trusted devices, known devices, and sessions.	• "urn:iam:t.users" • "urn:iam:t.groups" • "urn:iam:t.session" The privileges that are associated with the ***Identity Administrator*** role extend to "***Client***", "***User***" and "***Group***" entities.
Identity Client	IdentityClientPolicy	The Identity Client role is granted the permission to oversee the identity-related operations.	• "urn:iam:t.identityclient" The privileges that are associated with the ***Identity Client*** role are restricted to "***Client Only***," with no extension of these privileges to regular "User" and "Group."
Identity Credential Verifier	IdentityCredentialVerifier	The Identity Credential Verifier role is granted the permission to update the verified status of an email or phone number in the user profile.	"urn:iam:t.allowadmintoverifycredentials" The privileges that are associated with the ***Identity Credential Verifier*** role extend to "***Client***" and "***User***" entities.
Idstore Administrator	IdstoreAdminPolicy	The Idstore Administrator role is granted the permission to oversee the management of both "users" and "groups" within the configured identity stores.	• "urn:iam:t.idstoreadmin" The privileges that are associated with the ***Idstore Administrator*** role extend to "***Client***", "***User***" and "***Group***" entities.
Integration Administrator	IntegrationAdminPolicy	The Integration Administrator role is granted the authority to manage Applications, IDPs (Identity Providers), and IDP Discovery policies. Also, they manage Identity Stores, Certificates, and Secrets.	• "urn:iam:t.apps" • "urn:iam:t.idpconfigs" • "urn:iam:t.idstoreconfigs" • "urn:iam:t.certs" • "urn:iam:t.secrets" • "urn:iam:t.idstorebrowser" • "urn:iam:t.idpdiscoverypolicies" The privileges that are associated with the ***Integration Administrator*** role extend to "***Client***", "***User***" and "***Group***" entities.

Integration Auditor	IntegrationAuditorPolicy	The Integration Auditor role is granted the authority to audit Applications, IDPs, IDP Discovery policies. Also, users with this role can audit Identity Stores, Certificates, and Secrets. This role is equipped with the read-only access, and while cannot execute any operations, they are able to view all App, IDPs, and IDP Discovery policies.	• "urn:iam:t.apps" • "urn:iam:t.idpconfigs" • "urn:iam:t.idstoreconfigs" • "urn:iam:t.certs" • "urn:iam:t.secrets" • "urn:iam:t.auditonly" • "urn:iam:t.idpdiscoverypolicies" The privileges of ***Integration Auditor*** role extend to "***Client***", "***User***" and "***Group***" entities.
Risk Client	RiskClientPolicy	The Risk Client role is granted the authority to perform risk-related operations.	• "urn:iam:t.risk" The privileges of the ***Risk Client*** role are restricted to "***Client Only***," with no extension of these privileges to regular "User" and "Group."
ServiceAccounts Administrator	ServiceAccountsAdministrator	The ServiceAccounts Administrator is granted permissions to manage service accounts.	"urn:iam:t.serviceaccountsadmin"

Tenant Administrator	TenantAdminPolicy	The Tenant Administrator role is granted permission to perform all activities that are related to a tenant. The tenant policy that represents this role is associated with the client called, **DefaultTenantClient** and the scope notation, `urn:iam:t.*` represents tenant-level privileges. A user named, "***DefaultTenantAdminUser"*** with full administrative rights is also provisioned. The Tenant Administrator role can oversee various tasks that are related to Applications, ID Stores, Identity Providers, Authentication and Authorization Policies, Authentication Factors, Tenant Configurations, Global Settings, Admin Roles, Secrets, Signing Keys, and many more. Some of the tasks that a tenant admin performs are listed here: • Creating an "application" entity representing protected applications • Creating and managing Authentication Policies defining the level of authentication that is required by the application • Managing configuration settings such as certificates, message templates, available authentication factors, and so on. • Creating and managing a "LDAPConfig" entity representing the identity store with users to be authenticated by IDSP • Delegating Tenant Admin role to users and groups. Also to applications that wish to invoke IDSP APIs.	• "urn:iam:t.all" The privileges of ***Tenant Administrator*** role extend to "***Client***", "***User***" and "***Group***" entities
Tenant Auditor	TenantAuditorPolicy	The Tenant Auditor role is given the authorization to conduct audits on all operations within a tenant. They are equipped with the read-only access, and while they cannot execute any operations, they are able to view all tenant-related activities.	• "urn:iam:t.all" • "urn:iam:t.auditonly" • "urn:iam:t.adminroles" The privileges of ***Tenant Auditor*** role extend to "***Client***", "***User***" and "***Group***" entities.

User Password Administrator	UserPasswordPolicy	The User Password Administrator is granted permission to manage user passwords.	"urn:iam:t.userpasswordadmin" The privileges that are associated with the ***User Password Administrator*** role extend to "***Client***", "**User**" and "***Group***" entities.
Registration Client Administrator	RegistrationClientPolicy	The Registration Client Administrator is granted the authority to call the dynamic client registration API with the initial access token.	• "urn:iam:t.dcr" The privileges that are associated with the ***Registration Client*** role are restricted to "***Client Only***," with no extension of these privileges to regular "User" and "Group" entities.

Branding

The Branding dialog allows you to customize the various elements on the Sign In screen.

> **NOTE**
> The admin console experience is tailored to match the current privileges of each administrator. Therefore, to access and modify information within the console, you must possess the necessary privileges. You are currently viewing this page because you hold the `urn:iam:t.authenticate` privilege.

The dialog contains the following sections that can be customized:

- **Background**
  Specifies the image that is used as the background of the Sign In screen.
  - To use the default background, select **Use default background**.
  - To use a custom background, select **Use an image as a background** and click **Upload** to upload an image that meets the following file specifications:
    - **Format**: PNG, JPEG, or SVG
    - **Dimensions**: 1400px × 1400px, or less
    - **Size**: Less than 750 KB
- **Account Logo**
  Specifies the image that is displayed as the logo on the Sign In screen and in emails that are related to signing in.
  - To use the default logo, select **Use default logo**.
  - To use a custom logo, select **Use custom logo** and click **Upload** to upload an image that meets the following file specifications:
    - **Format**: PNG, JPEG, or SVG
    - **Dimensions**: 1400px × 1400px, or less
    - **Size**: Less than 750 KB

    After a custom file is uploaded, click **Preview** to view how the custom logo is displayed on the Sign In screen and in emails.
- **Custom Cascading Style Sheet (CSS)**
  Specifies the style sheet that is used for the Sign In screen.
  - To use the default style sheet, select **Use default styles**.
  - To use a custom CSS file, select **Use styles from an override CSS file** and click **Upload** to upload a custom CSS file that meets the following file specifications:

  - **Format**: CSS
  - **Size**: Less than 60 KB

  After a custom CSS file is uploaded, click **Preview** to view its contents.
- **Custom SignIn UI Content**
  Specifies the custom HTML markup to be inserted in the <head> and <body> sections for display of content or execution of scripts within the SignIn UI.
  - To customize the SignIn UI content, add the third-party code/script in the "Markup for <head>" or/and "Markup for <body>. For example, you may enter the code like the following:
    ```
    <div>Hello World!</div>
    ```
    You can provide any valid HTML string including JavaScript, HTML, or CSS.
    Default Behavior:
    These fields are empty by default.

  For more information on enabling Custom Embeds in SignIn UI, see here.
- **Favicon**
  Specifies the image that is displayed next to the page title in a web browser tab.
  - To use the default favicon, select **Use default favicon**.
  - To use a custom favicon, select **Use custom favicon** and click **Upload** to upload an image that meets the following file specifications:
    - **Format**: ICO, PNG, SVG, or GIF
    - **Dimensions**: 16px × 16px, 32px × 32px, 48px × 48px, 64px × 64px, or 128px × 128px
    - **Size**: Less than 750 KB

    After a custom favicon is uploaded, a preview of the favicon is displayed.
- **Save**
  Saves the selections that are made on the dialog.

Reference Links to:

- SignIn UI Branding using API flow

Service Configuration

All service configurations that can be set by a tenant administrator are defined in this page.

These service configurations enable granular control over the features of the application.

> **NOTE**
> The admin console experience is tailored to match the current privileges of each administrator. Therefore, to access and modify information within the console, you must possess the necessary privileges. You are currently viewing this page because you hold the `urn:iam:t.configs` privilege.

1. Log in to the Admin Console.
2. On the left panel of the Admin Console, click the icon and then click the icon, and then click **Service Configuration**.
   The **Service Configuration** page is displayed. Configure the settings that are required for your business enterprise.

Expiration Settings

Set expiration periods for the various codes and tokens that are required by Symantec Identity Security Platform.

- **Authorization Code**: An authorization code is a short-lived credential that is used to obtain an access token. The authorization code facilitates secure communication between the client application and the authorization server.

Use this setting to configure the duration, in seconds, for which an authorization code (azCode) remains valid. The default value is "120", the minimum value that is allowed is "60," while the maximum value is "300."

- **ID Token**: The ID token contains the authentication information for the user or service account in encoded format. Your application uses it to maintain a session for the user or servie account. Once it expires, the user or service account must log in again.

  Use this setting to configure the duration, in seconds, for which an ID token remains valid:
  - For users, the default value is "21600" and the minimum value that is allowed is "3600."
  - For service accounts, the default value is "3600" and the minimum value that is allowed is "3600."
- **Access Token**: Access Tokens are the confidential authorization for your application to access the information for the user or service account. Once it expires, the resource server no longer allows your application to access the user or service account data.

  Use this setting to configure the duration, in seconds, for which an Access token remains valid
  - For users, the default value is "3600", the minimum value that is allowed is "60", and the maximum value is ."28800"
  - For service accounts, the default value is "1200", the minimum value that is allowed is "60", and the maximum value is ."28800"
- **Refresh Token**: A refresh token is a credential that is used to obtain a new access token after the current access token has expired. It is also used to obtain additional Access Tokens with identical or narrower scope. The refresh token allows the client to maintain continuous access to resources without requiring the user to repeatedly log in.

  Use this setting to configure the duration in days for which a Refresh token remains valid. The default value is "7", the minimum value that is allowed is "1," while the maximum value is "180."
- **Signin Flow**: The signin flow represents the user authentication process. This value represents the maximum amount of time in which the authentication process must be completed.

Default Authentication Factor

When a user initiates the initial authentication process, or if no cookies or preferences are detected in the mobile app or web browser, the behavior is as follows:

- if the **Default Authentication Factor** property is ***not configured***, the SignIn UI displays the factors list as per the authentication policy evaluation.
- if the **Default Authentication Factor** property ***is configured***, the SignIn UI displays the configured default factors at each level of authentication.

Select the default credentials with IDSP for each authentication level from the following list:

- No Default
- Password from assigned Identity Store
- Password from Internal Identity Store
- Security code by SMS
- Security code by phone call
- Security code by email
- Biometrics
- Security Key
- Security code generated by authentication app
- Push to mobile app

User Credential Auto Enrollment

- **Auto enrollment allowed**: Toggle this option to automatically enroll or disable enrolling the user credentials from the ID Token Hint or identity store into IDSP, if they are not already enrolled.

User Credential Inline Enrollment

- **Inline enrollment allowed**: Toggle this option to enable or disable inline enrollment of user credentials during the Sign In flow for users whose credentials are not already registered in the user directory before authentication.
  If this option is selected, complete the following fields:
  - **Save credential information**: Select this option to store the user credential information provided during authentication in IDSP, eliminating the need for the user to re-enter it later.
  - **Required inline enrollment factors**: Select the mandatory factors from the following list that the user must authenticate with before proceeding with inline enrollment:
    - Password from assigned Identity Store
    - Password from Internal Identity Store
    - Security code by SMS
    - Security code by phone call
    - Security code by email
    - Biometrics
    - Security Key
    - Security code generated by authentication app
    - Push to mobile app

  You can select factors at multiple levels to be authenticated before proceeding with inline enrollment. Select the factors in the **Primary Authentication** list, and to add another level, click **Add** in the **Additional Authentication** list and select the factors.

  NOTE
  If the mandatory authentication factors are not met, the user authentication is denied (AUTH_DENIED).

Allowed Origin

Set the URLs from which IDSP accepts authentication request from your application. Requests from other URLs are rejected.

- **URL** : Define the URLs of the web applications that initiate authentication requests to IDSP. Authentication requests are only accepted from these specified URLs, and request from undefined URLs are rejected. This restriction is designed to mitigate the risks associated with cross-origin resource sharing (CORS).
  - Click **Add** to define multiple URLs.
  - To delete a URL, click the **X** button that appears next to the URL that you entered.

Remember Me

Set whether users can indicate that the username should be remembered.

- **Remember Me**: Toggle this option to enable or disable the "Remember Me" feature, which saves the username on a device after successful authentication. This option enables users to bypass the need to re-enter their username during subsequent login attempts, providing convenience in the login process.

Trusted Device

Set whether users can set a device as trusted.

- **Trusted Device**: Toggle this option to enable or disable the "Trusted Device" feature, which allows users to designate certain devices as trusted for authentication purposes. Once a device is marked as trusted, it is recognized as secure by IDSP, and authentication processes may be bypassed on subsequent login attempts from that device.
  If this option is selected, complete the following fields:

  - **Maximum allowed devices**: Select the number of devices that are allowed to be trusted. The maximum allowed limit is "10".
  - **Device Expiry**: Select the maximum life period to be maintained for a trusted device in the IDSP database. The device expiry can also be expressed as a fraction of a day for finer control, such as specifying eight hours by entering `0.33` (since 8/24 = 0.33). Valid values range from `0.0` upward with no upper bound. The maximum allowed limit is "365" days.

Device Code Settings

Set expiration and polling interval values that are required by IDSP for the Device Code flow.

- **Device Code Expiry**: Select the time period in seconds, after which the device code becomes invalid. The default value is "300", and the minimum value that is allowed is "60."
- **Polling Interval**: Select the time in seconds, the client application should check with the authorization server for updates on the status of the authentication process when using the device code flow. The default value is "5", the minimum value that is allowed is "5."

Biometric Settings

Set the maximum number of biometric devices and expiry period for the device.

- **Maximum Biometric Devices**: Select the maximum number of devices that are allowed per user for biometric authentication. The default value is "10" and the maximum value that is supported is "50".
- **FIDO Identifier Expiry Days**: Select the number of days a FIDO Identifier should be stored in the browser cache or mobile db. The default value and the maximum value is "365."

End-user Notifications

Set how end users receive notifications of operations that are performed for their accounts, such as account activations, account lock and unlock, and password reset.

- In the **Set user notification events** window, set for which of the following operations the notifications are sent. Move an operation into the right column to send notifications for that operation. Move the operation into the left column to stop sending notifications for that operation.
  - Account activation (sign in on new device)
  - Account Lock/Unlock
  - Reset Password
  - Credential Activity (credential added, updated, or removed)
  - End user email changed
  - End user account deleted
- Set which type of message or messages to send:
  - Verified Emails: send the notification to all verified emails registered for the end user
  - SMS to Verified Phones: send an SMS message to all verified telephone numbers registered for the end user.

Passkey Settings

These settings allow you to configure how passkeys are registered and authenticated within the system.

- **Registration Authenticator Attachment**: This setting determines where the passkey should be registered and stored during the registration process.

  - **Platform**: The passkey is registered and stored on the device platform (for example, iCloud Keychain or Google Password Manager) and may be synced across devices of that platform.
  - **Cross-platform**: The passkey is registered using a QR code or a hardware security key (roaming device) and does not get stored on the device itself.
  - "" (empty string) or null: The user may be given a choice on how the passkey should be registered.

  Default Value: platform. Possible values: platform, cross-platform, "" (empty string).

  NOTE
  The final user experience may depend on the platform and browser combination, and other configuration settings like "Passkey Registration Hints".
- **Registration Hints**: This setting provides hints during passkey registration to guide the user on the preferred method of registration.
  - **Client device**: The passkey should be registered and stored on the device platform (such as an iCloud Keychain or Google Password Manager). The passkey can be synced across the platform for authentication on other devices.
  - **Security key**: The passkey should be registered using a QR code or a hardware security key (roaming device) and should not be stored on the device.
  - **"" (empty string) or null**: The user may be given a choice on how the passkey should be registered.

  Default Value: "" (empty string), **Possible values**: security-key, client-device, security-key, client-device, hybrid, or a combination of these values.

  NOTE
  If there are a combination of values, the order of preference matters. The final user experience may depend on the platform and browser combination.
- **Authentication Transports**: This setting specifies the allowed transport methods for authenticating passkeys.
  - **Internal**: The passkey should be authenticated using a passkey that is stored on the device platform (such as an iCloud Keychain or Google Password Manager).
  - **USB, BLE, NFC**: The passkey should be authenticated using a QR code or a hardware security key that is connected to the authenticating device by USB, Bluetooth Low Energy (BLE), or Near Field Communication (NFC).
  - **Hybrid, null or "" (empty string)**: The user may be given a choice on how the passkey should be authenticated.

  Default Value: internal, **Possible values**: internal, usb, ble, nfc, hybrid or "" (empty string).

  NOTE
  If there are a combination of values, the order of preference matters. A blank value results in a hybrid. The final user experience may depend on the platform and browser combination.
- **Authentication Hints**: This setting provides hints during passkey authentication to guide the user on the preferred method of authentication.
  - **Client device**: The passkey should be authenticated using a passkey that is stored on the device platform (for example, iCloud Keychain or Google Password Manager).
  - **Security key**: The passkey should be authenticated using a hardware security key that is connected to the authenticating device.
  - **"" (empty string) or null**: The user may be given a choice on how the passkey should be authenticated.

  Default Value: "" (empty string), **Possible values**: security-key, client-device, security-key, client-device, hybrid, or a combination of these values.

  NOTE
  If there are a combination of values, the order of preference matters. The final user experience may depend on the platform and browser combination.
- **Enable FIDO Metadata Attestation**: This toggle enables or disables FIDO Metadata Attestation. When enabled, the system performs attestation checks against FIDO metadata for registered authenticators. This option enhances security by verifying the authenticity and characteristics of the FIDO authenticators used.

User Settings

Set commonly used user settings.

- **User Max Strike Count**: Select the maximum number of times the wrong user authorization is allowed. Once the maximum number of attempts are reached, the user is locked.
- **User Auto Unlock**: Select the time period after which the locked user gets automatically active.
- **Hide Authentication Failure Reason**: Toggle this option to disclose or hide the complete error message when an invalid user/locked user tries to start the authentication flow.
- **Skip Hide Authentication Failure Reason for Locked User**: This property determines whether to trigger PASSWORD obligation for a locked user when the tenant property “hideauthenticationFailurereason" is enabled. When "skipHideAuthenticationFailureReasonForLockedUser" is enabled, the actual error message is displayed for a locked user and the PASSWORD obligation is not activated even if “hideAuthenticationFailureReason” is enabled.
- **Track Last Login Details**: Toggle this option to enable or disable the tracking of last login success time or failure time for every user authentication attempt.
- **Days before password expiry to show a sign-in warning (0 disables warning)**: This setting defines the number of days before password expiration that a warning is displayed during sign-in, prompting users to change their password. No default values are configured for this setting.
  Possible values are 0-999.

Default Identity Store Assignment

Assign the default Identity Stores and define the search order.

Configure the Identity Stores that IDSP searches for user identities by default.

1. Select the Identity Stores from the **Available Identity Stores** list and move them to the **Assigned Identity Stores** list by using the right (→), and left (←) arrows.
2. Adjust the order of the assigned Identity Stores using the Up (↑), and Down (↓) arrows.
3. (Optional) Click **Lookup**.
   The **Lookup User** page is displayed.
   a. Enter the name of the user to search within the assigned identity store, and click **Lookup**.
      View the search results indicating whether the user was found or not.
   b. Click **Done**.
      Exits the **Lookup User** page and returns to the **Service Configuration** page.

Admin Console Identity Store Assignment

Assign the Identity Stores and define the search order when signing into the Admin Console. If no value is assigned, the default identity stores are used when signing in to the admin console.

Configure the Identity Stores that IDSP searches for user identities when signing in to the Admin Console.

1. Select the Identity Stores from the **Available Identity Stores** list and move them to the **Assigned Identity Stores** list by using the right (→), and left (←) arrows.
2. Adjust the order of the assigned Identity Stores using the Up (↑), and Down (↓) arrows.

Self Service Console Identity Store Assignment

This setting allows you to assign Identity Stores and define the search order when signing into the Self-Service Console. If this property is not configured, the system falls back to the assigned Default Identity Store.

Configure the Identity Stores that IDSP searches for user identities when signing in to the Self-Service Console.

1. Select the Identity Stores from the **Available Identity Stores** list and move them to the **Assigned Identity Stores** list by using the right (→), and left (←) arrows.
2. Adjust the order of the assigned Identity Stores using the Up (↑), and Down (↓) arrows.

Event Hook Configuration

Set the event hooks that apply to this account.

- **Event Hook Config**: This property defines audit event hooks to execute using a SCIM Config object.
  The supported hooks are:
  – factor.me.auth.lock.success
  – factor.me.register.complete.success
  – internal.user.sync.initiate
  – session.revoke.success
  The setting uses the following syntax:
  `<scim-config-id-or-name>, event1, event2,...`

  NOTE
  A SCIM config resource matching with the given id or name is validated.

Risk and Custom Service Provider Settings

Configure the Risk and Custom Service Provider Settings.

- **Primary Risk Provider**: Select the primary risk provider to be used during the authentication flow. The risk score that is returned by the risk provider in the authentication policy is used by the solution to determine the authentication obligations.
  Select a risk provider from the following drop-down list:
  – **None**: In this case, risk evaluation does not take place.
  – **IARISK**: Risk is evaluated using the VIP Risk Engine.
  – **AARISK**: Risk is evaluated using the Advanced Authentication Risk Engine.
- **Secondary Risk Provider**: Select the secondary risk engine to be used during step-up (re-authentication) flows.
- **SMS Provider**: Select the SMS provider from the drop-down list to deliver the security code through SMS. The default SMS providers that are supported are "Twilio."
- **Email Provider**: Select the Email provider from the drop-down list to deliver the security code through Email. The default Email providers that are supported in IDSP are "SMTP", "SMTP2", "Twilio SendGrid" and "Emailie." The default provider is "SMTP."
- **Push Provider**: Select the Push provider from the drop-down list to deliver a PUSH notification. The default Push providers that are supported is "VIP."
- **Claim Provider**: Select the provider from the drop-down list to be used for custom token claims.

Protection against Man-in-the-Middle (MITM) attack

Configure the protection level for MITM attack.

Select the required options from the following list:

- **Consider user's device during authentication flow**: This option considers the attributes of the user device, such as its type, model, or security status, during the authentication process to enhance security and verify the identity of the user.
- **Consider user's IP Address during authentication flow**: This option considers the user IP address during the authentication process, allowing for additional validation based on the geographical location or network origin of the device of the user.
- **Consider user's Session during API calls**: This option evaluates the session state of the user when processing API calls, ensuring that the user is authenticated and authorized to access the requested resources before executing the API operation.
- **Consider user's device during Admin Console operations**: This option considers attributes of the user device during administrative operations in the console, enhancing security and access control measures.
- **Consider user's IP Address during Admin Console operations**: This option takes into account the user's IP address during administrative operations in the console, enhancing security and access control measures.
- **Consider user's session during Admin Console operations (will cause immediate logout)**: This option requires active user sessions to be maintained throughout administrative operations in the console. If the session expires or is terminated during an operation, they are immediately logged out, enhancing security by preventing unauthorized access to administrative functions.

SSO Session Settings

Configure the settings that are related to the SSO session.

- **SSO Mode**: This property can be configured using the following values:
  - **None**: No Single sign-on session is created.
  - **Programmatic Only**: If this option is selected, implying no web cookie is being used, applications requesting sign-in are subjected to Single sign-on, leveraging the session token as follows:

    - passed in by a client as part of the OpenId "/authorize" call or programmatic "/authenticate" REST API.
    - returned to a client by OpenId token response along with Identity and/or Access token, or by "/factor" or "/authenticate" API at the end of successful completion of the authentication flow.
  - **Web SSO and Programmatic**: If this option is selected, applications requesting sign-in are subjected to Single sign-on that is tracked by the "sspsession" host-based web cookie and are detected by applications connecting through the OpenID protocol and HTTP URL.
  - **Cross Domain Web SSO and Programmatic**: If this option is selected, applications requesting sign-in are subjected to Single sign-on that is tracked by the "sspsession" host-based web cookie, and are redirected to the sign-in host to detect and use the cookie for the session identification
- **SSO Maximum Expiry**: When Single sign-on is enabled, this setting indicates the maximum duration of the SSO session, in seconds. Default is set to - 21600 seconds.
- **SSO Idle Expiry**: This value refers to the duration of inactivity a user can have within a Single Sign-On (SSO) session before being automatically logged out or prompted to reauthenticate.
- **Service Account session enabled**: When this option is enabled, service accounts are subject to SSO restrictions as defined in the following Service Account SSO settings.
- **SSO Maximum User Sessions for Service Accounts**: This value refers to the max number of Service Account sessions when accessing applications. Possible values are- 1-999. The default is set to 100.
- **SSO Maximum Expiry for Service Accounts**: Configure this setting to indicate the maximum duration of the Service Account SSO session, in seconds.
- **SSO Maximum User Sessions**: This value refers to the max number of end-user sessions when accessing applications. Possible values are- 1-999. The default is set to 10.
- **Admin Console SSO Maximum Expiry**: Configure this setting to indicate the maximum duration of the Admin Console SSO session, in seconds. Admin Console SSO gets enabled automatically when AdminConsoleUserSession MITM protection is enabled.
- **Admin Console SSO Idle Expiry**: This value refers to the duration of inactivity a user can have within a Single Sign-On (SSO) session in the Admin Console before being automatically logged out or prompted to reauthenticate.
- **Admin Console SSO Maximum User Sessions**: This value refers to the max number of administrative sessions when using the Admin Console. Possible values are- 1-99. The default is set to 3.

Identity Provider Discovery Settings

Configure the settings that are related to Identity Provider Discovery.

- **Discovery Enabled**: Select **Yes** or **No** from the drop-down list to determines whether Identity Provider (IdP) Discovery should be enabled or disabled.
  - When enabled, the system automatically determines the appropriate IDP based on user input or predefined rules.
  - When disabled, this option requires users to explicitly select their IDP during authentication.
- **Admin Console Discovery Enabled**: Select **Yes** or **No** from the drop-down list to determines whether Identity Provider (IDP) Discovery should be enabled or disabled in the Admin Console.
  - When enabled, the system automatically determines the appropriate IDP based on user input or predefined rules.
  - When disabled, this option requires users to explicitly select their IDP during authentication.

Post Authentication Access Control

Configure the settings that are related to Post Authentication Access Control.

- **Post Authentication Access Control**: Select **Yes** or **No** from the drop-down list to determine if the post-authentication access policy must be evaluated or not.
  The default value is "false".

NOTE

This value can be overridden by the setting-"Require Post Authentication Access Control" configuration that is made in the application configuration.

- **Post Authentication Access Control for Admin Console**: Select **Yes** or **No** from the drop-down list to determine if the post-authentication access policy must be evaluated for the AdminConsole Application. The default value is "false".

Custom URL Settings

Configure the custom URLs for IDSP.

- **Home URL**: Enter the Home URL which is used as the post-logout URL by the Identity Provider (IDP) to redirect users after executing a global logout process.
- **Login URL**: Enter the login URL where a user is directed to when authentication is required or when logging in to the application. This URL is where users enter their credentials to access protected resources.
- **Logout URL**: Enter the Logout URL which is maintained by the application within its configuration and used when discovering application-specific logout URL. If this URL is not configured in the application, this setting is used to discover the logout URL.
- **Post Logout URL**: Enter the Post Logout URL which an application requesting a logout configures in the app. If the application does not have a configured logout URL and does not indicate one in the logout request, this setting serves as the post-logout URL. If this setting is not provided, the homeUrl setting is used instead.
- **Error URL**: Enter the URL to redirect a user in the event of an authentication failure. If the application has its own error URL, redirection uses that URL. If the value is "protocol" (OIDC/SAML), control redirects back to the requesting application through a protocol-specific redirect.

  NOTE
  The following redirection behavior occurs for error handling within an application:

  1. If an app-level Error URL is not provided, or it is provided as "protocol," the error redirection occurs using the protocol.
  2. If the application intends to redirect to a specific Error URL, one of the following conditions must be met:
     a) The application explicitly specifies its own Error URL.
     b) The application includes the metadata urn:app:useGlobalErrorUrl=true.
     In this case:
     – If a tenant-level Error URL is provided, the redirection occurs to that URL.
     – If no tenant-level Error URL is provided, the redirection takes place to a built-in /signin/v1/error URL.

  – **Redirect to app**: If the application has its own error URL, redirection uses that specific URL to redirect the users.
  – **Redirect to custom URL**: If a custom URL has been specified in the application, redirection uses that specific URL to redirect the users.
- **Privacy URL**: Enter a Privacy URL to be displayed by the out-of-the-box Sign-in app. This URL redirects users to a webpage containing information about privacy practices, policies, and terms related to their application, service, or company.

Administrative Scope

This property governs the resource authorization enablement for constrained vs unconstrained access to resources and roles. By default, it is set to false.

- **Administrative scope enforced**: To enable resource authorization, set the value to true at Tenant level or at Global level to apply across all the tenants.

Identity Stores

Learn how to configure an ID store using the Symantec Identity Security Platform Admin Console.

NOTE
The admin console experience is tailored to match the current privileges of each administrator. Therefore, to access and modify information within the console, you must possess the necessary privileges. You are currently viewing this page because you hold the `urn:iam:t.idstoreconfigs` privilege.

The Identity Store represents an authoritative source of Identity for sourcing identity data that is used by the Symantec Identity Security Platform to manage authentication for non-Zero-Identity-Footprint applications. Access to such applications requires an identity to have an account in the Identity Store containing information that is used by IDSP to perform a user disambiguation, obtain, and update user's credentials such as email, phone number, validate user/group memberships, obtain/update disabled status.

Configuring an Identity Store using the Admin Console

Follow these steps:

1. Login to the Admin Console.
2. Click **Settings** > **Identity Stores**.
   The Identity Stores page is displayed, where you may find identity stores that have previously been configured, if any, and view further information like the Identity Store name, Description, Type, and Server of each store in a tabular format.
3. To create an identity store, click the **+ Add Identity Store** icon on the top right of the page.
   The Add Identity Stores page is displayed.
4. Enter a unique name for the Identity Store. Adding a description to the Identity Store is optional. Select the Type of the ID store you want to create, either **LDAP** or **SCIM**. Click **Next**.
   The newly created Id store configuration page is displayed with the ID Store name, description, and type.

To create either an LDAP configuration or SCIM configuration, see the following topics:

- LDAP Configuration
- SCIM Configuration

Identity Store - LDAP Configuration

This topic explains how to configure an LDAP Identity Store.

Server Configurations

Enter your LDAP server information here:

NOTE
The star (*) adjacent to the attribute name indicates that it is a mandatory filed.

- ***Server Name**: Enter the name of the host where the LDAP server is deployed. In case, both Symantec Identity Security Platform and the directory service is running on the same Kubernetes cluster, use the FQDN of the Kubernetes service, in this format: **dir-servicename.dir-namespace.svc**.
  Example: "sampleapp-ssp-symantec-dir.sampleapp.svc"
- ***Port**: Enter the port number of the LDAP server.
  Default: 389
  Example: "sampleapp-ssp-symantec-dir.sampleapp.svc"
- ***Enable SSL**: This is a boolean parameter that enables SSL communication between the LDAP server and the LDAP authenticator.
  Default: false
  If this parameter is set to true, then you must configure the following additional parameters:

  - **Ignore SSL Validation**: Set this parameter to "ON" to ignore SSL certificate validation during backchannel communication.
  - **Certificate**: Select the SSL certificate used to connect securely to the LDAP server.
    - Click **Select Certificate** to search for a certificate using an alias.
    - You can use the filter to search for a certificate using filters, such as Attributes, Operator, Value, Match Filters with **All** or **Any**.
    - If a certificate is not available, click on **Import** to upload a new certificate.
  - **CertificateAlias**: Enter the name of the entry in the certificate store containing the SSL cert.
- *__Enable Pool__: Set this parameter to "ON" to enable the LDAP directory server to maintain a pool of connections and assign them as needed. When a connection is closed, the directory server returns the connection to the pool for future use.
- *__Connection Timeout__: Enter the value of time in milliseconds that will be used in making the initial connection with the LDAP server.
  __Default__: 10000 milliseconds
- ***Read Timeout**: Enter the value of time in milliseconds that will be used on waiting to read data from the LDAP server.
  __Default__: 10000 milliseconds
- ***Bind DN**: Enter the distinguished name of the user that is used to connect to the LDAP directory server.
  __Example__: "cn=admin,o=democorp,c=US"
- ***Bind Password**: Enter the password of the user whose dn is passed in the "bindDN" attribute. The "bindDN" and "bindPassword" are used to connect to the LDAP directory server.
  __Example__: "password"
- ***Bind Password Alias**: If you create a secret with the "bindPassword" and want to use this secret instead of the password, then define an alias in the "bindPasswordSecretAlias" attribute.

  NOTE
  If both, "bindPasswordSecretAlias" and "bindPassword" values are provided, then the value of "bindPassword" takes precedence and this value is used. If "bindPassword" is not provided, "bindPasswordSecretAlias" is used.

  __Example__: "secretAlias"
- __Bind Controls__:
  Enter comma-separated list of LDAP connection controls that should be applied when establishing an LDAP connection.
  It serves two main purposes:
  1. __LDAP Connection Controls__: When connecting to an LDAP server, this field allows you to specify custom LDAP controls that should be sent during the bind operation. These controls are used to modify the behavior of the LDAP connection.
  2. __SiteMinder Integration__: This field also supports a special syntax for integrating with SiteMinder (CA Single Sign-On) for password validation. If a control starts with {sm} (the SiteMinder provider prefix), it is treated as a SiteMinder configuration ID.
  __Example: " \"bindControlsCsv\": \"{sm}" + customProviderId + "\",\r\n" +__
- Enter the following ***__User Configuration__*** details as follows:

  NOTE
  The star (*) adjacent to the attribute name indicates that it is a mandatory filed.
  - *__User Base DN__: Enter the distinguished name of the container where all the users are stored.
    __Example__: "o=democorp,c=us"
  - *__User Object Classes__: Enter the class name that holds attributes about users in the LDAP server.
    __Example__: "inetorgperson"
  - __User Attribute Mapping__: Enter the comma-separated string value listing all the mappings of logical attribute names to their physical LDAP attribute. See the User Attribute Mapping table for more information.

NOTE

- If you leave the attributeMapping value empty, the user's First Name and Last Name will not appear and the status will appear as 'UNKNOWN'.
- This applies to Active Directory only. The "status" attribute is used to determine whether the user is marked INACTIVE. The attribute follows the standard "UserAccountControl" flag values. If the value in the user status field is set to 514, the user is deemed as INACTIVE. This value is evaluated as a bitmask and if the bits representing 514 are not set then the status will be deemed as ACTIVE.

Example: "user_loginid=cn,name=cn,family_name=sn,given_name=givenName,middle_name=givenName,nickname=displa

– To add more user attributes, click **Add**.
  An empty row is added to the table.
– Select an IDSP attribute from the pre-populated list to map to the corresponding LDAP attribute.
– To delete any attribute, click the **X** displayed adjacent to each attribute.

- **Multi-value Attribute Mapping** -
  Enter a JSON formatted multi-value string that defines the mapping of logical attribute (like email and phone) to multiple single-valued LDAP attributes, each with an assigned tag (for example, 'business', 'personal'), to differentiate credentials.
  For example, if LDAP has two single-valued attributes, "mail" for business email and "personalmail" for personal email, they can be mapped to the logical "email" attribute, configured in the following manner:

```
"email":[
{"attribute": "mail", "tag":"business"},
{"attribute": "personalmail", "tag":"personal"}]
```

  To use tags in the authentication flow, configure these metadata in the application -

  - urn:iam:app:allowSmsCredsOfTag
  - urn:iam:app:allowEmailCredsOfTag
  - excludeCredsNotUsedForXDays

  See Application Metadata Properties.
- Enter the name of the logical mapping attribute representing the user's unique id in the **Login ID Attribute Mapping** field. This value should have a mapping from the logical to physical attribute.
- Enter the following ***Search Query*** details to perform an advanced search of user details:
  – **Use Regular Expression to adjust Lookup attribute**:
    - **REGEX TO BE APPLIED (FROM)**: This is a regular expression to be applied to the user login ID.
    - **REGEX TO BE USED (TO)**: This is a regular expression used to format the user login ID to match the corresponding attribute value in the identity store.

      NOTE
      Example:

      The "**loginIdRegexFrom**" and "**loginIdRegexTo**" parameter has been explained through the following example.

      In this example, when the following values are considered,

      - loginIdRegexFrom = ^(.*)\s(.*)$
      - loginIdRegexTo = 0{2}
      - Input value = +32 495551320

      The regular expression transforms the input value +32 495551320 into 0495551320 (result) by capturing the part after the space and adding 0 in front of it.
  – **Start**: Enter the beginning of a query to locate the user entry in LDAP.
    The end of the LDAP query will be appended to it. If the final query does not contain the substitution marker, $loginid then one is added between the beginning and the end part.

Example: "(cn=${loginid}"
See more examples here.

– **End**: Enter the end of a query.
Example: ")"
The values provided in the "Start" and "End" query can contain Dynamic Environment Variables. When used, these variables are substituted with their corresponding value in runtime. To learn more, see Dynamic Environment Variable Substitution.

- Enter the ***Group Configuration*** details as follows:
  – **Group Base DN**: Enter the base DN that is used for group search. If the groupBaseDN is not specified, the value of baseDN will be used instead.
  Example: "o=democorp,c=us"
  – **Object Classes**: Enter the class name that holds the list of comma-separated values that are used to search the group Names.
  Example: "groupOfNames"
  – **Group Attribute Mapping**: Select the LOGICAL ATTRIBUTE from the table dropdown and map it to the equivalent LDAP ATTRIBUTE. These are comma-separated list of key-value pairs representing the mapping between logical attributes and physical LDAP attributes for a group.
  Example: "name=cn", "members=member"
    a. To add more user attributes, click **Add**.
    An empty row is added to the table.
    b. Select an IDSP attribute from the pre-populated list to map to the corresponding LDAP attribute.
    c. To delete any attribute, click the **X** displayed adjacent to each attribute.
  – **Use memberOf Attribute**: Set this parameter to "ON" to enable 'memberOf' in the LDAP server.
    - If set to 'ON", the 'memberof' attribute of users who are defined in LDAP is used to get the group information. You can also use this behavior to obtain a list of values for custom attributes.
    - If set to 'OFF", the group membership is determined by querying for the groups where the username is listed in the "groupUserMappingAttribute" attribute.
  – **Group Name Attribute Mapping**: Select the value that represents the group name from the dropdown.
- Enter the ***General Configuration*** details as follows:
  – **Status Encoding**: Select the type of LDAP Directory you want to configure. This attribute is used by IDSP to perform native LDAP operations such as User Lockout. If the value is not specified, User Lockout operations fail unless the logical "status" attribute mapping is defined by LDAPconfig's "attributeMapping" property. By default, this attribute is set to "SITEMINDER".
  – Possible Values: AD, OUD, CADIR, DSEE, SITEMINDER, INTERNAL
    - For SiteMinder, the logical "status" attribute mapping must be defined by LDAPconfig's "attributeMapping" property, mapping to the LDAP user attribute representing "SiteMinder Disabled Attribute".
    - "INTERNAL" is meant to be used by certain LDAP implementations that have control over locked and/or disabled user states. For INTERNAL encoding, define two field mappings, "**is_locked**" and "**is_disabled**", within the "**attributeMapping**" field. These mappings must correspond to fields that hold the user lock and disable values in the LDAP. Upon user disambiguation, the user's status is determined based on the values within these mapped fields. Example mapping for INTERNAL: "is_locked=locked", "is_disabled=disabled".
  – **User locked status is synced to the identity store**: When this parameter is set to "ON", it specifies that the LDAP that is configured, is also the authoritative source for user locking. If this parameter is set to "OFF", IDSP becomes the authoritative locking source and the LDAP that has been configured will not be the locking source. The user locking is done when the user strike count reaches the maximum allowed value and the user is locked in the user store.
  – **Identity store is the source for detecting the locked state**: When this parameter is set to "ON", it specifies that the LDAP ID Store that is configured, is also the authoritative source for detecting the locked state of the user and

the user's locked state in the LDAP ID Store is used, but when this parameter is set to "OFF", the user's internal locked state is used.

- **Identity store is the source for user credential updates**: When this parameter is set to "ON", specifies if the LDAP ID Store that is configured, is also the authoritative source for user's credentials. In this case, the user's OTP creds that is established during inline registration will be synced back to the id store.
- **Identity store mapping strategy**: Enter the mapping strategy for mapping the disambiguation Id Store to the authentication Id Store.
  Example: `"attribute_mapping, user_loginid=cn, name=cn, family_name=sn, given_name=givenName, middle_name=givenName, nickname=displayName, preferred_username=displayName, profile=title,email=mail, phone_number=telephoneNumber, user_universalid=cn, status=employeeType, memberof=memberof"`
  The values provided in the "Identity store mapping strategy" can contain Dynamic Environment Variables. When used, these variables are substituted with their corresponding value in runtime. To learn more, see Dynamic Environment Variable Substitution.
  See ID Store Mapping in Password Authentication.
- **Password Reset URL**: Enter the URL that is presented to you during password authentication by the Signin UI. This URL will allow you to reset your password or will provide instructions on how to proceed.
  Example: "http://userPasswordResetURL"
- **Account Unlock URL**: Enter the URL that redirects users to a webpage to unlock or reactivate a user account that has been temporarily locked due to security reasons or user-initiated actions such as multiple failed login attempts.
- **Account Registration URL**: Enter the URL that redirects users to a webpage for registering new accounts.
- **Ignore connection error**: Select this parameter to save the Id store configuration even when the connection to the Id store server is not established.

- Click **Create Configuration**
  Once created, the LDAP Config ID is generated in the corresponding Id Store page. There is an option to copy the config id by clicking Copy.

User Attribute Mapping

This table shows how the IDSP attributes that can be mapped to the LDAP attributes.

Reserved Attributes (Logical Attribute name)	Description	LDAP Attribute-Mapping value Example (AD/CA Dir)	SCIM Attribute Mapping value Example
user_loginid* This logical attribute name is defined by the value of "userLoginIdAttributeMappingName"	This attribute represents the unique identifier of the user in the ID store. This unique identifier is used to associate the user with the IDSP account used to store additional information about the user, such as user credentials, device information.	AD: sAMAccountName/ userPrincipalName CA Dir: cn	userName
name	End-User's full name in displayable form including all name parts, possibly including titles and suffixes, ordered according to the End-User's locale and preferences.	displayName	displayName
family_name	Surnames or last names of the End-User.	sn	name.familyName
nickname			
preferred_username			
given_name	Given names or first names of the End-User.	givenName	name.givenName

Reserved Attributes (Logical Attribute name)	Description	LDAP Attribute-Mapping value Example (AD/CA Dir)	SCIM Attribute Mapping value Example
email	Preferred e-mail address for an OTP delivery. See the OTP section for more information.	mail	emails[primary]
phone_number	Preferred telephone number for an OTP delivery.	telephoneNumber	phoneNumbers.[mobile]
user_universalid	This attribute represents the Universal Id concept as used by SiteMinder.	<custom>	<custom>
status	Define the name of the attribute in the ID Store that contains the "Disabled Flag" value that is used by SiteMinder. See SiteMinder Documentation. This mapping is needed only when the statusEncoding is set to "siteminder" to indicate the same semantics for setting & discovering disabled status should be used.	<custom>	<custom>
memberof	This attribute is used to determine if 'memberOf' is enabled in the ID Store or not. • If the value is **True**, then the list of groups of the user is read from the user's attribute that is mapped to the "memberof" attribute. • If the value is **False**, then the group membership is determined by querying for the groups where the user's DN is listed in the group attribute that is defined by the **"groupUserMappingAttribute"** LDAPconfig/ SCIMconfig attribute.	memberOf	groups.display

See Configuring the ID Store for more information.

Identity Store Configuration - SCIM Protocol

This topic explains how to configure a SCIM Identity Store.

Server Configurations

Enter your SCIM server information here:

> **NOTE**
> The star (*) adjacent to the attribute name indicates that it is a mandatory filed.

- ***SCIM Server URL**: URL to connect to the SCIM server. Unless overwritten by "endpointSuffixOverride", it is expected that the "/Users" and "/Groups" will be appended to the SCIM service URL.

  > **NOTE**
  > It is mandatory to prefix the SCIM URL with the HTTP/HTTPS protocol.

  "https://scimservice.com"
- ***Security Method**: Type of security method to be used to connect to the SCIM server. Possible values are basic, oauth, oauthassertion.

For more information about the securityMethods and its associated mandatory properties that are required while creating the SCIM configuration, see Security Method Required Properties.

- ***Username**: Enter the name of the user who has access to this SCIM server.
- ***Secret Alias**: Enter an alias to a secret that can be used to access this SCIM server.
- **Test Connection URL**: Used to validate SCIM service connection during create/update of the configuration. If not specified, no validation is done when creating/saving configuration. For example, "https://scimservice.com/testconnection"
- **Status Encoding**:
  This attribute specifies the type of encoding used to represent the disabled status managed by the user lockout feature. The value can be "scim" to signal that user's disabled status should be determined using SCIM's standard user attribute "active", or it can be "siteminder" to manage the user's disabled state using a bit-mask notation where a custom ID store attribute represented by the 'status' logical attribute holds the physical bitmask representing various user states. For information about Sm_Api_Disabled_xxx values, see Exported Enumerations.
  Without the specific value, user lockout operations fail and the user's status will be assumed to be "unknown".
  This parameter is optional and when used should be set to the actual type of the Identity store.
  - See Supported User Status behavior table.
  - See User Locker

  For example, "scim" or "siteminder.
  The only value for SCIM config is "scim" (implying the use of SCIM's active attribute) or "siteminder" (implying the use of any xxx attribute indicated by status=xxx mapping)
- **Request Context**: Indicates hints to Symantec Identity Security Platform to execute when invoking SCIM endpoints.
  - **HTTP Header**: Specifies the HTTP Header to sent as part of the SCIM request.
  - **SCIM**: Only **scimIdAttributeMappingName** is supported. This setting specifies the attribute in the SCIM payload to be used for uniquely identifying the user resource.

Authoritative Source For Identity Lifecycle Management

- **User lock status is synced to the identity store**: This attribute specifies if the SCIM ID Store that is being configured, is also the authoritative source for user locking. If this value is set to "false", then IDSP becomes the authoritative locking source and the SCIM that has been configured is not authorized to be the locking source. The user locking is done when the user strike count reaches the maximum allowed value. If this value is set to "true", the user lock is done in the user store.
- **Identity store is the source for detecting the locked state**: This attribute specifies if the SCIM ID Store that is being configured, is also the authoritative source for detecting the locked state of the user or not. If this value is true, the user's locked state in the SCIM IDStore is used, otherwise the user's internal locked state is used.
- **Identity store is the source for user credential update**: This attribute specifies if the SCIM ID Store that is being configured, is also the authoritative source for the user's credentials. If this value is true, the user's OTP creds established during inline registration would be synced back to the id store.
- **Identity store mapping strategy**: The value of this property contains the mapping strategy for mapping the disambiguation Id Store to the authentication Id Store.
  See ID Store Mapping in Password Authentication.
  This attribute value can contain dynamic environment variables which will be substituted with their corresponding variable value, when this attribute is used.
  See Dynamic Environment Variable Substitution.
- ***Connection Timeout**: This is the value of time in milliseconds that will be used in making the initial connection.
- ***Read Timeout**: This is the value of time in milliseconds that will be used on waiting to read data.
- **Endpoint Suffix Override**: URL map in key-value pair representing custom SCIM endpoints. Only used in case a SCIM service does not support standard SCIM endpoints (/Users, /Groups) relative to the base URL identified by "scimURL"

```
{
```

```
    "GetUsers": "https://scimservice.com/getUsers",
    "PatchUsers":"https://scimservice.com/patchUsers"
}
```

- **Password Reset URL**: Enter the URL that is presented to you during password authentication by the Signin UI. This URL will allow you to reset your password or will provide instructions on how to proceed.
  Example: "http://userPasswordResetURL"
- **Account Unlock URL**: Enter the URL that redirects users to a webpage to unlock or reactivate a user account that has been temporarily locked due to security reasons or user-initiated actions such as multiple failed login attempts.
- **Account Registration URL**: Enter the URL that redirects users to a webpage for registering new accounts.
- **Privacy Policy URL**: Enter the URL where the user can go to view their privacy policy.

SSL Configurations

Enter your SSL information here:

- **Ignore SSL Validation**: This attribute indicates if the SSL validation should be ignored or not. The default is set to "true".
- **SSL Mode**: This indicates whether SSL validation should be performed or ignored. If the "sslMode" is enabled, it should validate the server's identity. Possible Values are - "VERIFY_CA", "VERIFY_IDENTITY". If the "sslMode" is not indicated, it defaults to "VERIFY_CA".
- ***Certificate**: Select the SSL certificate used to connect securely to the SCIM server.
  - Click **Select Certificate** to search for a certificate using an alias.
  - You can use the filter to search for a certificate using filters, such as Attributes, Operator, Value, Match Filters with **All** or **Any**.
  - If a certificate is not available, click on **Import** to upload a new certificate.

User Configurations

Describe how IDSP should work with users. Enter your user information here:

- ***Lookup Query**: This is the SCIM filter to locate the user entry in SCIM, the syntax for the SCIM filter can be found in RFC7644.
  The names of the attributes used in this query are physical attribute names.
  This value can contain dynamic environment variables which will be substituted with their corresponding variable value, when the query is used.
  See Dynamic Environment Variable Substitution.
  - **Use Regular Expression to adjust lookup attribute** : Enter the following ***Search Query*** details to perform an advanced search of user details:
    - **REGEX TO BE APPLIED (FROM)**: This is a regular expression to be applied to the user login ID.
    - **REGEX TO BE USED (TO)**: This is a regular expression used to format the user login ID to match the corresponding attribute value in the identity store.

      NOTE
      Example:

      The "**loginIdRegexFrom**" and "**loginIdRegexTo**" parameter has been explained through the following example.

      In this example, when the following values are considered,

      - loginIdRegexFrom = ^(.*)\s(.*)$
      - loginIdRegexTo = 0{2}
      - Input value = +32 495551320

The regular expression transforms the input value +32 495551320 into 0495551320 (result) by capturing the part after the space and adding 0 in front of it.

- **Start**: Enter the beginning of a query to locate the user entry in LDAP.
  The end of the SCIM query will be appended to it. If the final query does not contain the substitution marker, $loginid then one is added between the beginning and the end part.
  Example: "(cn=${loginid}"
  See more examples here.
- **End**: Enter the end of a query.
  Example: ")"
  The values provided in the "Start" and "End" query can contain Dynamic Environment Variables. When used, these variables are substituted with their corresponding value in runtime. To learn more, see Dynamic Environment Variable Substitution.

- **User Attribute Mapping**: This is a list of mappings that map logical attribute names to their physical SCIM attribute.
  Example

```
"user_loginid=userName, family_name=name.familyName, given_name=name.givenName, phone_number=phoneNumbers[primary],
```

  - To add more user attributes, click **Add**.
    An empty row is added to the table.
  - Select an IDSP attribute from the pre-populated list to map to the corresponding LDAP attribute.
  - To delete any attribute, click the **X** displayed adjacent to each attribute.
- **Login ID Attribute Mapping**: Enter the name of the logical mapping attribute representing the user's unique id in the **Login ID Attribute Mapping** field. This value should have a mapping from the logical to physical attribute.
- **User Login ID as SCIM ID**: Set this option to true, if you want the IDSP user Id to be used to login to the SCIM server.

Group Configurations

Define how your user groups are identified in IDSP if you are configuring group policies. Enter your group information here:

- **Group Attribute Mapping**: Select the LOGICAL ATTRIBUTE from the table dropdown and map it to the equivalent SCIM ATTRIBUTE. These are comma-separated list of key-value pairs representing the mapping between logical attributes and physical SCIM attributes for a group.
  Example: "name=displayName", "members=members.display"
  a. To add more user attributes, click **Add**.
     An empty row is added to the table.
  b. Select an IDSP attribute from the pre-populated list to map to the corresponding SCIM attribute.
  c. To delete any attribute, click the **X** displayed adjacent to each attribute.
- **Group Name Attribute Mapping**: Select the attribute to map to the group name in the SCIM server. If no value is set, this value defaults to "name".
- **Group Name as SCIM ID**: Set this option to true, if you want the IDSP group name to be used to login to the SCIM server.
- **Ignore Connection Error**: Select this parameter to save the Id store configuration even when the connection to the Id store server is not established.
- **Create Configuration**: Once created, the SCIM Config ID will be generated and displayed in the corresponding Id Store page. There is an option to copy the config id by clicking "Copy".

Internal Identity Store Configuration

Learn how to configure the Internal ID store using the Symantec Identity Security Platform Admin Console.

NOTE
The admin console experience is tailored to match the current privileges of each administrator. Therefore, to access and modify information within the console, you must possess the necessary privileges. You are currently viewing this page because you hold the `urn:iam:t.idstoreconfigs` privilege.

The Internal Identity Store is used to store local users and groups that are fully managed by the Identity Service. Similar to other Identity Stores, these users and groups are used by the IDSP to manage authentication/authorization for non-Zero-Identity-Footprint applications.

Access to such applications requires an identity that is used by IDSP to:

- Perform a user disambiguation
- Obtain and update user credentials (such as email or phone number)
- Validate user or group memberships
- Obtain/update disabled status.

Configuring the Internal Identity Store Using the Admin Console

Use the following steps to configure the internal Identity Store using the Admin Console. Optionally, you can configure the internal Identity Store through the Identity Store APIs (see Configuring the Internal ID Store).

Follow these steps:

1. Log in to the **Admin Console**.
2. Select **Settings** > **Identity Stores**.
   The **Identity Stores** page appears, displaying the Internal ID Store in the list of Identity Stores. Unlike other ID Stores, the internal ID Store is automatically deployed by IDSP; no lifecycle management operations are required.
   This page also allows you to view and update the default password policy. The default password policy is automatically deployed when the internal ID store is deployed. This policy cannot be disabled or deleted, it can only be updated. See (Setting a Password Policy).
3. Select the **Internal Identity Store** from the store list. The **Configuration** properties for the Internal Identity Store appear in the following table:

Property Name	Description	Property Value
Disambiguation Query	The query used to disambiguate the user. To format the user login ID to match the corresponding attribute value in the identity store, select the **Use Regular Expression to adjust lookup attribute** option. You must also enter the following values: • **Regex to be applied (from)**: A regular expression that is applied to the user login ID. • **Regex to be used (to)**: A regular expression that is used to format the user login ID to match the corresponding attribute value in the identity store.	`user_loginid eq ${loginid}` Example usage of the **Use Regular Expression to adjust lookup attribute** option: In this example, when the following values are considered, • **Regex to be applied (from)** = ^(.*)\s(.*)$ • **Regex to be used (to)** = 0{2} • Input value = +32 495551320 The regular expression transforms the input value +32 495551320 into 0495551320 (result) by capturing the part after the space and adding 0 in front of it.
Mapping Strategy	This property determines how to look up the user for Password Authentication.	`user_loginid= ${idprofile.user_loginid}`

Property Name	Description	Property Value
UserName Generation Rule	The rule that determines the structure of system-generated usernames. The username generation rule can be formed with the combination of one or more string-literals, place-holder-for-number-of-digits, and variables which can be separated by '+' character. See Username Generation Rules	`"${idprofile.given_name} + '.' + ${idprofile.family_name}"`
Display Name Template	Defines how the user display name is generated from the user attribute values. By default, the Display Name Template is set to `givenName` and `familyName` . If some or all of the values for the specified attributes are present, the display name is generated from the available values. If none of the values are present, the display name is generated from the user `Login` value.	`"${idprofile.given_name} ${idprofile.family_name}"`
Scim Search Limit	The search limit for SCIM User and Group search requests.	Default: 500
Search Timeout Seconds (max 1000):	The number of seconds after which the search fails if no values are returned.	• Default: 0 • Maximum: 1000
Self Service Unlock	Support for Self Service Account unlock. When enabled, initiating an Account unlock invokes a Password Reset flow which, upon completion, unlocks the user account.	**Off** – Self-service account unlock is disabled (default) **On** – Self-service unlock using Password Reset is enabled
Credential Display does not contain masked data	When a email or phone number from a user profile is used as a credential, a value of `false` indicates that the **displayName** value for the email or phone contains a masked value and that value should be used when displaying the credential.	**Off** – let the service provide the masked email and phone value to be displayed. **On** – assumes the persisted **email/phone.displayName** field on the users profile contains the masked value to be displayed
Allow StartIndex as a search parameter for SCIM requests	Start Index is used in the SCIM pagination model to indicate the next page set to retrieve. With large query results, specifying a high start Index value (for example, page number 10k out of a 20K user query) can have a significant performance effect.	**Off** – StartIndex is turned off (default). **On** – StartIndex is turned on.
Use Priority for 'primary' Designation	Some User multi-value user attributes support the designation of a primary value (for example, email.primary). In the absence of a primary designation, a priority value on the attribute can be used. The lowest priority value is deemed primary.	**Off** – do not use priority for designating a primary (default) **On** - use priority for designating a primary
Allow Admins to generate password	By default, an administrator cannot generate or reset a password for a user. If set to On, the scope is granted to generate or reset a password for a user. If this property is enabled, then you must logout and login back to the Admin Console UI for the changes to take effect.	**Off** – do not allow an administrator to generate or reset a password (default) **On** - grant the administrator the scope to generate or reset a password for a user.

Property Name	Description	Property Value
Allow Federated users to have local password	By default, federated users cannot have local passwords. Set this value to On to allow Federated users to have local passwords.	**Off** – do not allow local passwords (default) **On** - allow local passwords
Emails Must be unique	By default, email addresses for SCIM users must be unique. If set to Off, emails can be the same across different accounts.	**Off** – allow email addresses to be the same across user accounts **On** - require unique email addresses across accounts (default)
Skip Email Validation	When creating users, email addresses are validated for correctness. If set to Off email addresses are not validated.	**Off** – do not validate email addresses during user creation **On** - validate email addresses during user creation (default)
Skip Phone Validation	When creating users, phone numbers are validated for correctness. If set to Off phone numbers are not validated.	**Off** – do not validate phone numbers during user creation **On** - validate phone numbers during user creation (default)
Validate principals in policy definition	When saving an authorization or authentication policy, validate the existence of the principal of the policy.	**Off** – do not validate that the principal exists when saving an authentication or authorization policy (default) **On** - validate that the principal exists when saving an authentication or authorization policy
Allow Administrators Self Modify	By default, administrators cannot modify their own profile information. If set to On, administrators can modify their own profile information	**Off** – do not allow administrators to modify their own profile information (default) **On** - allow administrators to modify their own profile information
Require Account Activation	When creating SCIM users, the account must be activated. If set to Off, the account is automatically activated at creation.	**Off** – an account is automatically activated at creation **On** - the account must be activated (default)
Account Activation Notification Strategy (CSV)	Identifies which credentials (Email or phoneNumber) are returned by the Verified Credentials API for the user, which is then used to deliver the activation token (see Identity Self-Service Operations API.). **Writer:** Change xref to new self service account activation topic. This flag is used to determine where the activation token is delivered.	Comma-separated list of email addresses or SMS-enabled phone numbers. The value is "email, sms" by default.
Allow Self Service Account Activation	Allow a user to activate an account the first time that the user authenticates to the account.	**Off** – do not allow self-service account activation (default) **On** - allow self-service activation
Self Service Account Activation Notification Strategy (CSV)	Identifies which credentials (Email or phoneNumber) are returned by the Verified Credentials API for the user (see User Self Service API and Identity Self-Service Operations API.). These credentials can be used in subsequent API calls to deliver the token for Password Reset or Account Unlock operations.	Comma-separated list of email addresses or SMS-enabled phone numbers. The value is "email, sms" by default.

Property Name	Description	Property Value
Account Activation Token Expiry In Days	The number of days within which the account must be activated before the token that is created by the activation request expires.	
Password Hash Algorithm	Passwords that are stored in the identity store are encrypted using the SHA-256 encryption algorithm. Select how to apply the SHA-256 algorithm.	• Select **SHA256** to have the password encrypted using one iteration with the SHA256 algorithm. • Select **SHA256_600** to have the password encrypted using 600,000 iterations with the SHA256 algorithm. The SHA256-600 option provides an additional level of security. The significantly higher iteration count greatly increases the time that is required to compute the hash, making it more difficult for brute-force attacks to succeed.
Email Domain For Simulated Forgot Password Flow (CSV)	When hide authentication failure is true and a user requests to reset a password, the Forgot Password flow presents a randomly generated and obfuscated email address or phone number to the user. Use this property to define the email domains that appear on this page for the user. Set whether the **Forgot Password** page presents an email address or phone number in the **Self Service Account Activation Notification Strategy (CSV)** field.	Comma-separated list of email domains. The default value is **null**.
Password Recovery Flow Type	Define how the user is prompted for authentication when recovering a password.	**Verified Credential** – send an OOB message to the user that includes a link to reset the password (default). **Policy** – use the authentication method defined in the PasswordRecoveryPolicy (see Authentication and Access Policies)
Account Unlock Flow Type	Define how the user is prompted for authentication when unlocking an account	**Verified Credential** – send an OOB message to the user that includes a link to reset the password (default). **Policy** – use the authentication method defined in the AccountUnlockPolicy (see Authentication and Access Policies)
Account Activation Flow Type	Define how the user is prompted for authentication when activating an account.	**Verified Credential** – send an OOB message to the user that includes a link to reset the password (default). **Policy** – use the authentication method defined in the AccountActivationPolicy (see Authentication and Access Policies)
Register password credential with account activation	Require a user to create a password when activating an account.	**Off** – do not prompt the user to create a password when activating an account **On** – require the user to create a password when activating an account (default)

Property Name	Description	Property Value
Password Reset Initiator URL	An optional value that, when defined, provides a **Forgot Password** link on the **Sign-in App** login page. Optionally, you can set a custom URL value. If set, this URL is used in the Password Reset URL that is sent to the user. The custom URL supercedes any other URLs set for administrator-initiated password resets.	`"passwordResetInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiatePasswordReset",`
Password Reset Changer URL	An optional value that, when defined, is used in the Password Change URL that is sent to the user. Optionally, you can set a custom URL value. If set, this URL is used in the Password Reset URL that is sent to the user. The custom URL supercedes any other URLs set for administrator-initiated password resets.	`"passwordResetChangerUrl": "${system.issuerUrl}/ui/v1/signin/passwordReset",`
End User Password Reset Vanity Issuer URLs	Optional URLs to include in the Password Reset email sent to end users. The administrator initiating the password reset can select which URL to send. To use these vanity URLs, you must set **Password Reset Initiator URL** and **Password Changer URL** to **Use default URL**. Any URLs set in the **Use custom URL** setting supersedes the vanity URLs.	Vanity URL. Include "https".
Account Unlock Initiator URL	An optional value that, when defined, is used in the Account Unlock URL that is sent to the user.	`"accountUnlockInitiatorUrl": "${system.issuerUrl}/ui/v1/signin/initiateAccountUnlock"`
Account Registration URL	An optional value that, when defined, provides a **Register Account** link on the **Sign-in App** login page.	This value is empty by default.
Custom User Attributes	Defines the set of Custom User Attribute definitions.	One or more custom User attributes can be defined. See the following table for details.

Defining Custom User Attributes

The Internal ID Store User schema can be extended by deployments to define additional User attributes. New User attributes can be defined with the following properties:

Property	Description	Value
name	A unique name for the attribute. The name must be unique within the User schema. The name must not conflict with any of the existing SCIM attribute namespaces.	
type	The data type of the attribute: • STRING • INTEGER • BOOLEAN • DATE	One of the supported types

Property	Description	Value
searchable	An indicator if the attribute can be used in a search operation. This enforcement is only honored using the ID Store API, as the SCIM protocol does not define a searchable property.	**Off** – attribute cannot be used in a search expression **On** - attribute cannot be used in a search expression
enabled	Defines the enabled state. Attributes whose enabled state = false cannot be used in requests, searches, and are not returned in responses.	**Off** – the attribute is disabled and cannot be used in user operations **On** - the attribute is enabled and can be used in user operations
multivalued	Sets whether attributes can have more than one value. Applies only to attributes of type *string*, *integer*, or *date*. Once set to *multivalue*, an attribute cannot be changed to *single value*.	**Off** – the attribute can have only one value **On** - the attribute can have multiple values. If this property is not set, the attribute supports only a single value.

User Attributes

This section describes the full user schema, including any custom user attributes defined as described previously. The following table shows the schema properties:

Property	Description	Value
Name	A unique name for the attribute. The name must be unique within the User schema. The name may not conflict with any of the existing SCIM attribute namespaces.	
Type	The data type of the attribute: • STRING • INTEGER • BOOLEAN • DATE • CREDENTIAL	One of the supported types
User Disambiguation	Indicates if the attribute can be used in a disambiguation query.	**Off** – attribute cannot be used in disambiguation **On** - attribute can be used in disambiguation
Searchable	An indicator if the attribute can be used in a search operation. **Note** this enforcement is only honored using the ID Store API, the SCIM protocol does not define a searchable property	**Off** – attribute cannot be used in a search expression **On** - attribute cannot be used in a search expression

SCIM URI	The URI of the SCIM namespace.	One of the following values:
		`urn:ietf:params:scim:schemas:core:2.0:User`
		`urn:ietf:params:scim:schemas:extension:ente`
		`urn:ietf:params:scim:schemas:extension:iam:`
		`urn:ietf:params:scim:schemas:extension:iam:`

SCIM Writeable	A flag indicating of the attribute can be updated using the SCIM Interface	**Off** – attribute cannot be changed via SCIM **On** - attribute can be changed via SCIM

Setting a Password Policy

Create a password policy to set the password requirements and behavior for your end users.

Password policies set the password requirements and behavior for your end users. IDSP deploys a default password policy when it deploys the internal ID store. The default poassword policy cannot be disabled or deleted, it can only be updated.

If you have multiple policies, the policy to be applied for a given user is selected based on:

1. The directorySearchFilter match
2. The policy is in an Enabled state

If multiple policies match the criteria, then the priority of the policy in ascending order is applied. Only one policy is applied.

You can manage password policies from the Identity Stores page of the Admin Console:

1. Log in to the **Admin Console**.
2. Select **Settings** > **Identity Stores**, then select the **Password Policies** tab.
   The default password policy is listed, along with any custom password policies that you have created.
   – To create a new policy, click **Add**.
   – To update an existing policy, click the policy name.
3. You can set the following for a password policy. You can also export the policy as a JSON file and, except for the default policy, delete or disable the policy from this page.
   - Composition Rules
   - Password Age and History
   - Dictionary Rules

Composition Rules

Set the following rules to define how passwords are composed:

- **Minimum Password Length:** Specifies a minimum length for user passwords.
- **Maximum Password Length:** Specifies the maximum length for user passwords.
- **Maximum Repeating Characters:** Determines the maximum number of identical characters that can appear consecutively in a password.
  For example, if this value is set to 3, then “aaaa” cannot appear anywhere in the password. However, “aaa” is acceptable within a password.
- **Unique characters:** Determines the minimum number of unique characters that must appear in a password.
- **Upper Case Letters:** Specifies whether to allow upper case alphabetic characters and, if so, the minimum number a password must contain.
- **Lower Case Letters:** Specifies whether to allow lower case alphabetic characters and, if so, the minimum number a password must contain.
- **Letters:** Requires the user to use only letters in the password and defines the minumum number of letters that a password must contain.
- **Digits:** Requires the user to use only digits (numbers) in the password and defines the minumum number of digits that a password must contain.
- Letters & Digits: Allows the user to use letters and digits (numbers) in a password and defines the number of letters or digits the passowrd must contain.
- **Non-Alphanumeric (Special) Characters:** (/maxNonAlphaNumerals/minAlphaNumerals)
  Specifies whether to allow non-alphanumeric characters such as punctuation marks and other symbols (“@”, “$”, and “#”) and if so, the minimum/maximum number a password can contain.

  - **Special Characters:** Specifies the allowed symbols (such as “@”, “$”, and “#”).
  - **Excluded characters:** Specifies the disallowed symbols (such as “*”).
  - **Minimum Characters required:** Sets the minimum number of special characters that the password must contain.
  - **Maximum Characters allowed:** Sets the maximum number of special characters that the password can contain.
- **Excluded Words or Partial Words (Strings):** Specifies the words or parts of words that cannot be part of the password.
- **First Name Disallowed:** Specifies that the first name of the user cannot be part of the password.
- **Last Name Disallowed:** Specifies that the last name (surname or family name) of the user cannot be part of the password.
- **User Name Disallowed:** Specifies that the username of the user cannot be part of the password.
- **Regular Expressions:** Password regular expressions let you specify regular expression text patterns for matching or non-matching criteria, for example, when you want to require that the first character is a digit, and the last character is not. You configure multiple expressions for a single password policy. If you create multiple expressions, acceptable passwords must meet the criteria for all the expressions.
- **Evaluate Password Composition Rules on Authentication:** Enforces real-time validation by checking that the password meets the composition rules during every user authentication attempt.

When the Match Length field is set to four results in the following actions:"TeddyBear", rejected because Bear matches the bear entry in the dictionary file."dictionary1", rejected because "tige" matches the first four characters of the tiger entry in the dictionary file."dictonary2", accepted since "iger" does not include the first letter of the tiger entry in the dictionary file.

Password Age and History Rules

Set the following rules for how long passwords remain valid and when passwords can be reused:

- **Number of unique passwords before a password can be reused:** Sets the number of passwords that a user can create before using a previous password again. A value of 0 means that there is no password aging.
- **Password expires after:** Sets the number of days after which a password is no longer valid and must be changed.
- **Evaluate Password Expiration Rules on Authentication:** Enforces real-time validation by checking that the password meets the password age and history rules during every user authentication attempt.

Dictionary

Set the rules for whether dictionary words are excluded from passwords:

- **Dictionary Rules:** Specifies whether dictionary password verification is to be done as per the dictionary that is uploaded using the API /admin/v1/PasswordDictionary. See Password Management API
- **Match Length:** Controls the length of strings that are compared against values in the dictionary file. The comparison ignores the case of the strings. A value of 0 means that only passwords that match a string in the dictionary exactly are rejected. When the match length is greater than zero, passwords are rejected in the following conditions:
  - The password includes a substring which starts with the same series of characters as a dictionary entry.
  - The number of consecutive matching characters is greater than or equal to the number specified in the Match Length field.

  For example, consider a dictionary file that contains the following entries:
  - lion
  - tiger
  - bear

  When the Match Length field is set to 4:
  - *TeddyBear* is rejected because *Bear* matches the entry *bear* in the dictionary file.
  - *tige* is rejected because it matches the first four characters of the entry *tiger* in the dictionary file.
  - *iger* is accepted since it does not include the first letter of the entry *tiger* in the dictionary file.

Integration Setup

View the Integration Setup for Symantec Identity Security Platform.

1. Login to the Admin Console.
2. On the left panel of the Admin Console, click the icon and then click the Integration setup icon.
   The **Integration Setup** page is displayed.
   This is a read-only page that displays the necessary information required to integrate IDSP with external services as a OpenID/OAuth Provider, SAML Identity Provider, SAML Service Provider, or Launcher Apps.
3. This page provides the following information:
   - `Select the host` - Select the IDSP host based on which the integration setup information should be displayed. You could either select an **FQDN** host or a **Vanity** host. If you select a host that uses a vanity URL, then you are presented with another option, `Use tenant name in urls`.
     - **Use tenant name in urls**: This option is displayed if you select a vanity host. In general, when using a vanity host, the placeholder "**common"** is used as the tenant-name. However, selecting "Use tenant name in urls", replaces "common" with the current tenant name. To know more about vanity hosts, see Resolving Tenancy.
   - **OpenID - As an OpenID/OAuth Provider**
     - `Issuer` - The unique identifier for the OpenID Provider.
     - `Discovery URL` - The URL for discovering the OpenID Provider's configuration.
     - `JWKS URL` - The endpoint URL for retrieving the JSON Web Key Set (JWKS).
     - `Issuer Signing Certificate` - The certificate used to sign tokens issued by the provider.
   - **OpenID - As an Relying Party**
     - `Callback URL` - The URL where the OpenID Provider redirects after authentication.
     - `Relying Party Encryption Certificate` - The certificate used by the relying party to encrypt sensitive data.
   - **SAML - Metadata URL** - Provides the endpoint to access the IDP's metadata.
   - **SAML -As an Identity Provide (IDP)**
     - `Identity Provider Metadata File` - Download the Identity Provider metadata file and import it into your Service Provider. The metadata file includes the information needed to automatically integrate IDSP into your Service Provider.
     - Optionally, use the following Identity Provider information to manually integrate IDSP into your Service Provider
       - `Entity ID` - A unique identifier for the Identity Provider. This ID is shared with SPs to establish trust and validate authentication requests and responses.
       - `Login URL` - The URL endpoint for initiating SAML authentication. This is the endpoint of the SAML Identity Provider (IDP) where the Service Provider (SP) redirects users to authenticate themselves.
       - `Logot URL` - The URL endpoint for initiating SAML logout. This is the endpoint of the SAML Identity Provider (IDP) that facilitates user logout from the SSO session.
       - `Identity Provider Signing Cert` - The certificate used to sign SAML assertions and responses.
   - **SAML -As an Service Provider (SP)**
     - `Service Provider Metadata File` - Download the Service Provider metadata file and import it into your Identity Provider. The metadata file includes the information needed to automatically integrate IDSP into your Identity Provider.
     - Optionally, use the following Service Provider information to manually integrate IDSP into your Service Provider

- `Entity ID` - A unique identifier for the Service Provider. This ID is shared with IDPs to establish a trust relationship with the SP.
- `Assertion Consumer Service URL` - The URL where the SP receives authentication responses from the IDP.
- `Service Provider Signing Certificate` - The certificate used by the Service Provider (SP) to sign SAML assertions or requests sent to the IDP. It enables the Identity Provider (IDP) to verify that the requests or assertions originate from a trusted SP.
- `Service Provider Encryption Certificate` - The certificate used to encrypt SAML assertions or responses to ensure data confidentiality.

– **Launcher Apps**
- `Login URL` - The URL to initiate login for Launcher Apps.
- `Logout URL` - This is the URL to logout the session from the launcher application.

Risk Service

Learn about enabling and disabling different risk services.

Risk Services Overview

This topic provides an overview of the Risk Service and allows you to enable or disable the services.

Navigating to Risk Services

1. Log in to the Admin Console.
2. In the left navigation pane, click on Settings > Risk Service.

Risk Service Toggle: This toggle enables or disables the entire Risk Service functionality.

- To Enable: Click the toggle switch to the "ON" position
- To Disable: Click the toggle switch to the "OFF" position

> **NOTE**
> If the Risk Service is disabled, no risk rules will be evaluated, regardless of their individual settings.

Risk Threshold: This field defines the global risk threshold. If the calculated risk score for an event exceeds this threshold, the transaction is considered risky.

1. To configure, enter the desired numerical value for the risk threshold. The example shows a default value of 50.
2. After making any modifications to the Risk Service toggle or Risk Threshold, click Save.

Supported Risk Services

- Basic Rules
- IP Rues
- Country Rules

Basic Rules

The Basic Rules topic allows you to configure individual risk rules that contribute to the overall risk score.

Navigating to Basic Rules

1. Log in to the Admin Console.
2. In the left navigation pane, click `Settings > Risk Service > Basic Rules`.

This page is structured with expandable sections for each risk rule. Each section typically includes an "ON/OFF" toggle, a "Risk Score" input, and specific settings relevant to that rule.

Behavior Anomaly

The risk engine keeps track of all the locations from where a user performs various events and those locations are grouped as clusters. A cluster is identified by its current location, OS, and browser. Based on the event status, a cluster is marked as either an anomaly or non-anomaly. The rule checks if the current event is from a known cluster and determines the risk score.

If the current event does not match with any of the existing clusters, then the risk engine checks if the current event is closest to any non-anomaly clusters for the last 10 events. A comparison score is derived between the current event and clusters based on IP, OS, and browser. If the comparison score is greater than (>) the threshold, then the current event is marked as belonging to that cluster.

Toggle (ON/OFF): Click the toggle next to "Behavior Anomaly" to activate or deactivate this rule.

- **Risk Score**: Defines the risk score assigned when difficult travel is detected.
  - **To Configure**: Enter the desired risk score value. The default score is 50

The following table describes the "***Behavior Anomaly"*** settings that can be updated in the UI:

Settings Name	Description	Default Value
Enable Static Score	This toggle controls whether a static score is used for behavioral anomalies.	Disabled

Device Recognition

This rule assesses risk based on whether the device used for authentication is recognized.

When a user logs in using a device for the first time, the device recognition rule treats the user transaction as risky as the device is not registered, and creates a device tag for it. However, if the user logs in for a subsequent time using the same device, then this device signature has a device tag associated with this user from the user's initial transaction. This device is then treated as a registered device, and the risk score that is calculated is much lesser.

Toggle (ON/OFF): Click the toggle next to "Device Recognition" to activate or deactivate this rule.

- **Risk Score**: Defines the risk score assigned when difficult travel is detected.
  - **To Configure**: Enter the desired risk score value. The default score is 50.

The following table describes the "***Device Recognition"*** settings that can be updated in the UI:

Settings Name	Description
• unrecognizedPct (API terminology) • Unrecognized percentage (UI terminology)	Indicates the percentage of risk score that is assign rule if the device is not recognized. **For example**: If the risk score is 50 and "unrecognizedPct" is 50% risk score for device recognition would be 25, which total risk score.
• deviceIdPct (API terminology) • Device Id percentage (UI terminology)	Indicates the percentage of risk score that is assign rule if the deviceId is valid and active.

• deviceLimitAction • Device Limit Action (UI terminology)	This setting defines the action to be taken when a u to register a new device, but they have already reac Number Of Devices Per User" limit. The available o are: • rejectNewDeviceRegistration: If selected, any at to register a new device beyond the set limit are • deleteOldest: If selected, when a user tries to re device and has reached the limit, the system au removes the oldest registered device to allow th be registered.
• maxNumberOfDevicesPerUser (API terminology) • Max Number Of Devices Per User (UI terminology)	• This refers to the maximum number of devices t is allowed to have registered with their account. indicates that there is no explicit limit enforced, a to register an unlimited number of devices. If a p were set (for example, 5), a user would only be up to that many devices.
• deviceFPExpiryInactivityTime (API terminology) • Device fingerprint expiry inactivity time (UI terminology)	Indicates the max number of days after which an ina (FP) is deleted. This is tied to the last access time. Y risk reason as - "DEVICE_EXPIRED_NO_AUTH"
• registeredPct (API terminology) • Registered percentage (UI terminology)	Indicates the percentage of risk score that is assign rule if the browser is registered.
• simThreshold (API terminology) • Similar threshold (UI terminology)	Indicates the similarity threshold.
• registeredmobile (API terminology) • Registered mobile (UI terminology)	Indicates the risk score that is assigned to this risk r device is registered.
• tagExpireInSeconds (API terminology) • Tag expire in seconds (UI terminology)	Indicates the expiry of a new tag issued during the r
• deviceFPExpiryMaxBindingLife (API terminology) • Device fingerprint expiry max binding life (UI terminology)	Indicates the max allowed days for user to device Ta is tied to the last created time. You can get the risk r "DEVICE_EXPIRED_MAX_LIFE"
• lastNRiskEvaluation (API terminology) • Last N risk evaluations (UI terminology)	Specifies the number of past authentications used f the current device from which the authentication is a found in these past **N** records, the rule is triggered. To know when to enable "Last N risk evaluations" se ***Device Change rule descirption in the next secti***

Device Change

The **Device Change** rule enhances the Device Recognition rule by introducing additional risk evaluation criteria. To enable this rule alongside **Device Recognition**, set **lastNRiskEvaluation** to **1 or higher**.

The assigned risk score is determined based on the configured threshold. This rule always triggers when a device is unknown or new. If the device is known, the rule is triggered when the latest authentication attempts originates from a device not found in the last N authentications (explained in the previous table). The assigned risk score is based on the configured threshold.

Example

The following table illustrates how the **"Device Change"** rule behaves for a system with `lastNRiskEvaluations = 3` that is, it checks the last **3** risk evaluations for the authentication attempts):

Table 121: Device Change Risk Rule Example Scenarios

Trans action	Device	Rule Triggered?	Reason
1	Device1	No	Not enough previous transactions (rule requires 3 for evaluation).
2	Device2	No	Still not enough data to evaluate the rule.
3	Device3	No	Now we have 3 transactions, but still not enough data to evaluate the rule.
4	Device4	Yes	The device `(Device4)` is **not present** in the last 3 transactions (`Device1` , `Device2` , `Device3`), so the rule is triggered.
5	Device3	No	The device (`Device3`) **exists** in the last 3 transactions (`Device2` , `Device3` , `Device4`), so the rule is not triggered.
6	Device3	No	The device (`Device3`) is still in the last 3 transactions (`Device3` , `Device4` , `Device3`), so the rule is not triggered.
7	Device2	Yes	The device (`Device2`) is **not present** in the last 3 transactions (`Device4` , `Device3` , `Device3`), so the rule is triggered.

Enhanced Difficult Travel

The Difficult Travel rule detects when a user attempts to log in from geographically distant locations within an unreasonably short time period. This rule analyzes the IP address of the current authentication event and compares it with the IP address from the previous event for the same user entity. The rule calculates the geographical distance between these two IP locations and measures the time duration between the two events. It then determines the implied "travel speed" and compares this against a configurable threshold. If the calculated travel speed exceeds the difficult travel speed limit, the rule increments the risk score by the specified amount.

NOTE

The rule result is skipped if both the previous event and current event belong to non-anomaly clusters. This design reduces false positives in scenarios where an account is legitimately shared by multiple users from different geographical locations. This risk rule requires the Network Data chart to be deployed.

For more information about network data charts, see Uploading Network Data Into IDSP Risk Database.

Toggle (ON/OFF): Click the toggle switch next to "Enhanced Difficult Travel" to enable this rule.

- **Risk Score**: Defines the risk score assigned when difficult travel is detected.
  - **To Configure**: Enter the desired risk score value. The default score is 50.

The following table describes the "***Enhanced Difficult Travel***" settings that can be updated in the UI:

Settings Name	Description	Default Value
• maxSpeedUserCanTravel (API terminology) • Maximum speed user can travel (UI terminology)	Indicates the maximum reasonable travel speed tolerance allowed during distance calculation. Value is measured in miles per hour.	500
The following two settings are currently inactive and do not affect rule behavior. They are reserved for future functionality.		
• maxDistanceTolerance (API terminology) • Maximum Distance Tolerance (UI terminology)	Indicates the maximum threshold distance (in miles) a user can travel beyond the set "maxSpeedUserCanTravel" limit before triggering the risk rule.	50
• maxSharedSessionsPerUser (API terminology) • Maximum shared session per user (UI terminology)	Indicates the maximum number of concurrent sessions a user can maintain from different geographical locations.	1

Inactive User

Inactive User rule checks if the user is accessing the application after a long time.

Toggle (ON/OFF): Click the toggle switch next to "Inactive User" to enable this rule.

- **Risk Score**: Defines the risk score assigned when a user is detected as inactive.
  - **To Configure**: Enter the desired risk score value. The default score is 50.

The following table describes the "***Enhanced Difficult Travel***" settings that can be updated in the UI:

Settings Name	Description	Default Value
numberOfDaysUserInactive (API Terminology) Number of days user inactive (UI Terminology)	Indicates the maximum number of days a user is inactive after which this rule gets triggered.	90

Previous Event is Confirmed Risk

The rule checks if the previous event of this entity is marked as suspicious. If the previous event is marked suspicious, then the current event is marked risky and the risk score is set to maximum.

Toggle (ON/OFF): Click the toggle switch next to "Previous Event is Confirmed Risk" to enable this rule.

- **Risk Score**: Defines the risk score assigned when the previous event was marked suspicious.
  - **To Configure**: Enter the desired risk score value. The default score is 100.

Recognized Computer

This rule checks if the event is initiated from a recognized computer.

Toggle (ON/OFF): Click the toggle switch next to "Recognized Computer" to enable this rule.

- **Risk Score**: Defines the risk score assigned when the request is from the recognized computer.
  - **To Configure**: Enter the desired risk score value. The default score is -25.

The following table describes the "Recognized Computer" settings that can be updated in the UI:

Settings Name	Description	Default Value
recognizedScore (API Terminology) Recognized Score (UI Terminology)	the risk score that is assigned to this risk rule if the request is from a recognized computer.	-25

User Velocity

User Velocity rule gets triggered based on the number of times a user attempts to log into an application within a span of time. The average number of logins that triggers the rule is five times in a span of 10 minutes. This rule has been enabled in the system by default.

Toggle (ON/OFF): Click the toggle switch next to "User Velocity" to enable this rule.

- **Risk Score**: Defines the risk score assigned when the request is from the recognized computer.
  - **To Configure**: Enter the desired risk score value.

The following table describes the "User Velocity" settings that can be updated in the UI:

Settings Name	Description	Default V
timeIntervalInMinutes (API Terminology) Time interval in minutes (UI Terminology)	Indicates the minutes considered for user velocity calculation.	10

numberOfRiskEvaluationsPerUser (API Terminology) Number of risk evaluations per user (UI Terminology)	Indicates the max number of risk evaluations allowed per user within the defined timeInterval.	6

Work Day Time

Work Day rule checks if the user is accessing the application during work hours and working days.

Toggle (ON/OFF): Click the toggle switch next to "Work Day Time" to enable this rule.

- **Risk Score**: Defines the risk score assigned when user is accessing the application during work hours.
  - **To Configure**: Enter the desired risk score value.

The following table describes the "Work Day Time" settings that can be updated in the UI:

Settings Name	Description	Default Value
startHour	Indicates the start hour of a working day.	8
endHour	Indicates the end hour of a working day.	17
workDays	Indicates the days of the week considered as work days.	MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY

Risk Configurations usinf REST API is covered in Understanding Risk Rules.

IP Rules

This topic describes how to configure various IP-based rules within the risk assessment framework, including managing Allowed, Risky (Blocked), and Fraud IP addresses, and setting up rules to detect suspicious IP address changes.

Allowed IP Address

This section allows you to manage the IP addresses that are considered "allowed" within the risk assessment framework.

Allowed IP Address: Toggle this switch ON to activate the Allowed IP Address rule. When enabled, transactions originating from these IPs are factored into the risk assessment.

Risk Score: Enter a numerical value in this field. This score is applied to transactions that match the Allowed IP Address rule, influencing the overall risk assessment.

Upload Allowed IP List:

- Choose a file or drag and drop it here:
- Click the **Upload** button to import the IP list into the system.

Download: Click this button to download the currently configured list of Allowed IP Addresses file. Note that the UI has a limit of 500 displayed entries, but the downloaded file contains every entry. The downloaded file's name, such as **ALLOWED_1762508443630.txt**, incorporates the IP list type (in this example, 'allowed'), and a timestamp for easy identification.

Delete All: Click this button to remove all IP addresses from the Allowed IP Address list.

IP Address or Range Table: This table displays the IP addresses or ranges currently configured in the Allowed IP list, along with their respective TYPE (for example, IPV4).

Risky IP Address

This section allows you to manage the IP addresses that are considered "risky" (blocked) within the risk assessment framework.

Allowed IP Address: Toggle this switch ON to activate the Risky IP Address rule. When enabled, transactions originating from these IPs are factored into the risk assessment.

Risk Score: Enter a numerical value in this field. This score is applied to transactions that match the Risky IP Address rule, influencing the overall risk assessment. The default risk score for this risk rule is ***100***.

Upload Risky IP List:

- Choose a file or drag and drop it here:
- Click the **Upload** button to import the IP list into the system.

Download: Click this button to download the currently configured list of Risky IP Addresses file. Note that the UI has a limit of 500 displayed entries, but the downloaded file contains every entry. The downloaded file's name, such as **enterprise_ip_list_181b8f45_e1dc_45cf_8d5b_7811b6c4f19b_allowed_20251107_091332.txt**, incorporates the enterprise ID, the IP list type (in this example, 'allowed'), and a timestamp for easy identification.

Delete All: Click this button to remove all IP addresses from the Risky IP Address list.

IP Address or Range Table: This table displays the IP addresses or ranges currently configured in the Risky IP list, along with their respective TYPE (for example, IPV4).

Fraud IP Address

This section allows you to manage the IP addresses that are considered "fraud" within the risk assessment framework.

Fraud IP Address: Toggle this switch ON to activate the Fraud IP Address rule. When enabled, transactions originating from these IPs are factored into the risk assessment.

Risk Score: Enter a numerical value in this field. This score is applied to transactions that match the Fraud IP Address rule, influencing the overall risk assessment. The default risk score for this risk rule is ***100***.

The rule compares the IP of current event with the fraud IP static list loaded using a network data loader job. If the event IP matches the IP ranges in the list, and that fraud ip range has not expired, it increments the final risk score based on the risk level set by that fraud ip range. This risk rule works only when the Network Data chart is deployed. To know more about network data chart, see Uploading Network Data Into IDSP Risk Database.

IP Address Change

The 'IP Address Change' risk rule detects anomalous user behavior when the latest authentication originates from an IP address significantly different from the IP addresses of the last N authentication attempts within a short period. This triggers a risk score assignment based on a configured threshold, contributing to the overall risk assessment of user transactions.

IP Address Change: Toggle this switch ON to activate the IP Address Change rule. When enabled, transactions originating from these IPs are factored into the risk assessment.

Risk Score: Enter a numerical value in this field.

The default risk score for this risk rule is ***50***.

Configurable Setting for the "IP Address Change" Rule

Settings Name	Description	Default Value	Possible Valu
lastNRiskEvaluations	Specifies the number of past authentications used for evaluation. If the current authentication IP is not found in these past **N** records, the rule is triggered.	5	Any positive

Example

The following table illustrates how the **"IP Address Change"** rule behaves for a system with `lastNRiskEvaluations = 3` that is, it checks the last **3** risk evaluations for the authentication attempts):

Table 122: IP Address Change Risk Rule Example Scenarios

Trans action	IP Address	Rule Triggered?	Reason
1	1.0.0.1	No	Not enough previous transactions (rule requires 3 for evaluation).
2	1.0.0.2	No	Still not enough data to evaluate the rule.
3	1.0.0.3	No	Now we have 3 transactions, but still not enough data to evaluate the rule.
4	1.0.0.4	Yes	The IP (`1.0.0.4`) is **not present** in the last 3 transactions (`1.0.0.1`, `1.0.0.2`, `1.0.0.3`), so the rule is triggered.
5	1.0.0.3	No	The IP (`1.0.0.3`) **exists** in the last 3 transactions (`1.0.0.2`, `1.0.0.3`, `1.0.0.4`), so the rule is not triggered.
6	1.0.0.3	No	The IP (`1.0.0.3`) is still in the last 3 transactions (`1.0.0.3`, `1.0.0.4`, `1.0.0.3`), so the rule is not triggered.
7	1.0.0.2	Yes	The IP (`1.0.0.2`) is **not present** in the last 3 transactions (`1.0.0.3`, `1.0.0.4`, `1.0.0.3`), so the rule is triggered.

Reference Links

- IP List API examples using REST API
- Risk Configurations using REST API is covered in Understanding Risk Rules.

Country Rules

Learn about configuring risk country rules using Admin Console.

The Country Rules tab allows you to configure rules that assess the risk of transactions based on the country from which the user is authenticating. These rules can detect unusual country changes or flag transactions originating from high-risk countries.

Navigating to Country Rules Settings

1. Log in to the Admin Console.
2. In the left navigation pane, click on Settings.
3. Under Settings, select Risk Service.
4. Navigate to the Country Rules tab.

Country Rules - Country Change

The "Country Change" risk rule is triggered if the latest authentication attempt originates from a country that differs from the countries recorded in the last N authentication attempts. This rule helps detect suspicious changes in a user's geographical access patterns, which could indicate a compromised account or an unauthorized access attempt.

To configure the Risk Country rule:

1. Enable/Disable the rule:
   - To enable the rule, toggle the switch to ON.
   - To disable the rule, toggle the switch to OFF.
2. Set the Risk Score: Enter a numerical value (default: 50) to determine the base risk assigned when this rule is triggered.
3. Configure the following setting:
   - Last N risk evaluations: Specifies the number of past authentications used for evaluation. If the current country from which the authentication is attempted is not found in these past N records, the rule is triggered. (Default: 5, possible values: Any positive integer).

Country Rules - Risky Country

The "Risk Country" rule utilizes a list of countries identified as sources of fraudulent or malicious transactions. This list is maintained by enterprises in accordance with country-specific regulations. The system determines a user's country based on GPS coordinates (if available via SDK) or IP address. Transactions originating from countries marked as risky are assigned a higher risk score. This rule is active only when the Network Data chart is deployed.

To configure the Risk Country rule:

1. Enable/Disable the rule:
   - To enable the rule, toggle the switch to ON.
   - To disable the rule, toggle the switch to OFF.
2. Set the Risk Score: Enter a numerical value (default: 100) to determine the base risk assigned when this rule is triggered.
3. Select Countries:
   - From the ALL COUNTRIES list, select the countries you wish to designate as risky.
   - Click the right arrow button (>) to move selected countries to the SELECTED COUNTRIES list.
   - To remove a country from the SELECTED COUNTRIES list, select it and click the left arrow button (<).

     NOTE
     Country codes used in the API should be compliant with ISO2 two-letter country codes (e.g., "ie" for Ireland, "zw" for Zimbabwe).

Risk Configurations usinf REST API is covered in Understanding Risk Rules.

User Menu

Navigate to the "User Menu" dropdown in the Admin Console.

1. Locate User Menu: In the top-right corner of the Admin Console, click the user menu dropdown showing your username.
2. Access Dropdown Menu: The dropdown displays several options:
   - Profile: Access user profile settings and account information
   - Generate Token: Direct access to PAT generation
   - About: Platform version and information
   - Sign Out: Securely logout from the platform

Generation of Personal Access Token

Learn about generation of Personal access token in Admin Console.

Personal Access Tokens (PATs) provide a secure method to authenticate with Application APIs without using traditional username/password credentials. This topic describes the process of generating PATs in the Symantec Identity Security Platform.

Navigation

Navigate to the "User Menu" dropdown in the Admin Console.

1. Locate User Menu: In the top-right corner of the Identity Security Platform, click on the user menu dropdown showing your username.
2. Access Dropdown Menu: The dropdown displays several options:
   – Select Generate Token: Click "Generate Token" from the dropdown menu to directly access the PAT generation workflow.

Steps to generate PAT

When selecting the Generate Token option in the User Menu, the PAT generation dialog opens.

1. **Configure Token Settings**:
   – Token Expiration Settings -
     • Personal Access Token expiring in : Set duration in seconds (Max: 897 seconds)
     • Personal Refresh Token expiring in : Set duration in days (Max: 7 days)
     • Refresh Token Description: Enter a descriptive name for your refresh token in the text field. This helps identify the token's purpose later.
2. **Select Privileges**: Use the search bar to find specific privileges by name and description.
   – Privilege Selection:
     • Review available privileges in the table showing-
       • PRIVILEGE: Name of the privilege
       • DESCRIPTION: What the privilege allows
       • ROLE: Associated role
   – Check the boxes next to required privileges.
3. **Generate Tokens**:
   – Review all settings and selected privileges.
   – Click "Generate" to create the tokens.
   – The system processes your request and generates both:
     • Personal Access Token
     • Personal Refresh Token
4. **Download and Store Tokens**:
   – Token Download Dialog: A "Token Download" dialog appears with the message: "Copy your token(s) and store them securely. You won't be able to view them again after you click Close."
   – Available Actions for Each Token:
     • Personal Access Token: Click "Copy to Clipboard" to copy the token | Click "Download" to save as a file.
     • Personal Refresh Token: Click "Copy to Clipboard" to copy the token | Click "Download" to save as a file.
5. **Secure Storage**:
   – Copy both tokens immediately.
   – Store them in a secure location (password manager, encrypted file, etc.)

IMPORTANT

Tokens cannot be viewed again after closing this dialog.

6. Click "Close" when you have securely stored both tokens.

Self-Service Console

Learn how to log into the Self-Service Console, configure tenant settings, understand default SoftMFA behavior, and apply constraints effectively.

This section explains how to use the web-based user interface, the Symantec Identity Security Platform Self-Service Console to update a user's personal information, reset password, setting-up additional MFA and viewing their trusted devices that are registered in the IDSP and viewing their last account activity.

Logging Into the Self-Service Console

To access the Self-Service Console in your IDSP environment, use the following URL format and replace the {SSP_FQDN} with the appropriate value:

```
https://${SSP_FQDN}/default/ui/v1/selfserviceconsole
```

Where,

- ***${SSP_FQDN}*** is the fully-qualified-domain-name of the host where you have installed the IDSP. See Configuring the Environment Variables.
- ***default*** is the unique tenant name in IDSP

When you use the above URL to access the Self-Service Console, the "Sign In" page is displayed. On the "Sign In" page, enter your username followed by the password.

By default, the Self-Service Console is configured to use only "**Password**" as the authentication factor. However, since SoftMFA is enabled by default and inline enrollment of factors is not possible, we recommend configuring multiple authentication factors (MFA) to enhance security.

User Login Flows for the Self-Service Console - ZFP and Non-ZFP Users

There are two main categories of users who may require to log in to the Self-Service Console:

- **ZFP users** - To access the Self-Service Console, these users must first configure their Identity Provider (IDP) information within the SelfServiceLoginPolicy's **user** condition.
  When a Zero Footprint (ZFP) user accesses the Self-Service Console URL, they are redirected to the Sign-In page where they need to authenticate using the IDP configured in the `SelfServiceLoginPolicy`. Upon successful authentication, they gain access to the console.
- **Non-ZFP users** - Non-ZFP users are further divided into Internal and External users.
  - **Internal users** - These users can directly log in to the Self-Service Console using the console URL with their **username and password**.
  - **External users** -These users access the Self-Service Console by first navigating the console using the console URL. They are then required to complete **multi-factor authentication (MFA)**, as defined in the `SelfServiceLoginPolicy`, the default login policy for the Self-Service Console. Once authentication is successful, they gain access to the console.

Reference Links

- Self-Service Home

Home

The Home page provides quick access to modules like Personal Information, Security, Activity, and Password, allowing you to view and manage your account details with ease.

NOTE

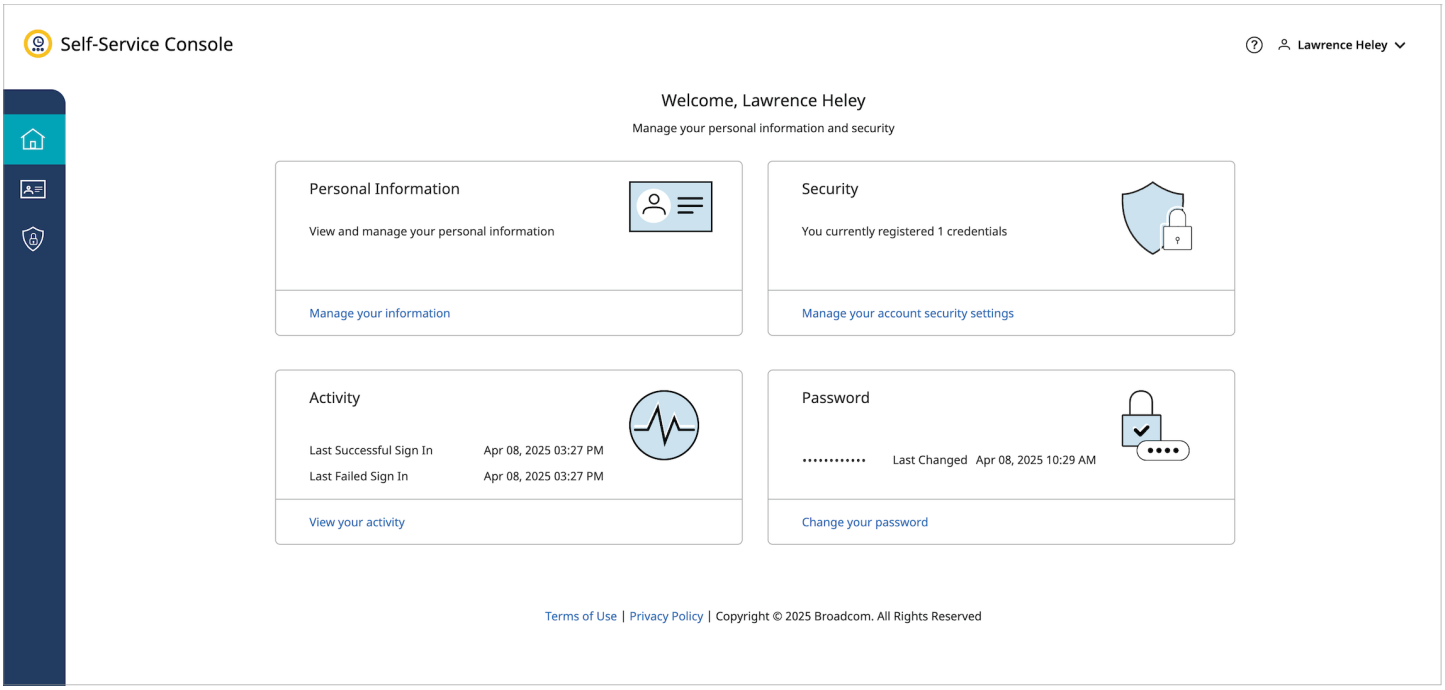
- The self-service console adapts to the scoping constraints set for each user. To access and modify information within the console, your actions must fall within your defined scoping constraints. You are currently viewing this page because it aligns with your configured constraint, "DefaultMeConstraints".
- This page is visible to you because you are identified as an internal Symantec Identity Security Platform user. Users configured outside the internal IDStore will not have access to this information.

On the Home page, you can view all sections and navigate to the relevant pages by selecting a module and clicking its link.

- Personal Information - View and manage your personal information
- Security - Manage your accounts security settings
- Activity - View your activity
- Password - Change your password

The following Self Service Console Home page shows all the sections that are displayed to the users.

Figure 35: Self Service Console - Homepage



My Applications

End Users can manage their applications and personalize the dashboard view.

The "My Applications" screen presents a collection of application tiles, organized into sections of your choice. It provides the users with a personalized dashboard of applications they can access. This help users quickly find, organize, and launch their various work-related applications.

Users can search for specific applications, sort the display, and add new sections to further customize their view.

To navigate to the "My Applications" screen, click the "Applications" screen on the left navigation panel of the Self-Service Console.

The sections included in this screen are as follows:

- **System Notification**: Located in the top navigation bar, the bell icon provides access to system-wide notifications that affect all users. These include maintenance schedules, service disruptions, security alerts, critical system announcements, etc.
- **User Notification**:
  Located next to "My Applications", the bell icon provides access to the user notifications that provides an essential communication channel between administrators and end users. The notifications system keeps users informed about important application assignments, updates, and administrative actions that affects their access. It provides users with a centralized location to view and manage all their notifications.
  Clear Notifications : Located at the bottom of the notification list, this option allows users to dismiss all notifications at once, helps maintain a clean notification history.
- **Application Search**: Allows users to search for an application by typing into the search bar, making it easier to find a specific app within a large collection of applications.
- **Application Sections**: Applications are grouped under collapsible sections, such as "Work". This helps categorize applications and improve navigability. The down arrow next to the section suggests the section can be expanded or collapsed.
- **Application Dashboard**: Each application is represented by a distinct tile, featuring the application's logo and name (for example, "Oracle E-Business Suite - ERP", "Google Keep"). Clicking on these tiles would launch the respective application.
- **Action Menu (Top Right)**: A dropdown menu providing options to control the display and sorting of applications:
  - **Show Sections**: Displays all the sections that are added in the dashboard.
  - **Sort Ascending**: Arranges applications in alphabetical order (A-Z).
  - **Sort Descending**: Arranges applications in reverse alphabetical order (Z-A).
- **Settings icon** next to "Show Sections" provides options to control the display and sorting of sections, such as resizing the tiles, reordering the tiles or adding new sections.
  - **Tile Size**: This option allows users to change the size of the tiles that are displayed in the dashboard. There are three options to choose from, Small, Medium and Large.
  - **Section Order**: This option allows users to change the order in which the applications are displayed in the dashboard. You can change the order of the applications and also edit the name of the sections.
  - **Add Section**: This option allows users to create new custom sections to personalize application organization. This can be done either here in the Settings panel or directly on the Applications screen by clicking the "**+ Add Section**" button.

Personal Information

Allows end users to review and manage their personal information through the Self-Service Console.

In this screen, you can update the personal information for your Symantec Identity Security Platform account without having to contact an administrator. This section provide you control over your personal data, allowing you to review and, in some cases, edit your information.

NOTE

- The self-service console adapts to the scoping constraints set for each user. To access and modify information within the console, your actions must fall within your defined scoping constraints. You are currently viewing this screen because it aligns with your configured constraint, "DefaultMeConstraints".
- This screen is visible to you because you are identified as an internal IDSPuser. Users configured outside the internal IDStore will not have access to this information.

The "Personal Information" screen displays user information under **Basic Information** section.

Basic Information

This section presents fields displaying your identity details. These fields are initially read-only, indicating that they are for viewing purposes. You are provided with an "Edit" button, suggesting that you can switch this section into an editable mode to update details.

The fields include:

- **Username**: The user's unique identifier within the system.
- **First Name**: The user's given name.
- **Middle Name**: The user's middle name (optional).
- **Last Name**: The user's family name.
- **Locale**: The user's preferred language and regional settings.
- **Time Zone**: The user's preferred time zone.

You can do the following tasks in this table:

1. Click **"Edit"** to update personal details. You can add emails and select the type of email from the dropdown, add the display name.
2. Add Email, select the type of email from the dropdown, and add the display name.
3. Click "**Save**" your changes. Upon saving, the "Verify Email" button gets enabled. Click the button to receive an email with information to verify your email.

Security

This section allows you to actively monitor recent activities, manage your registered authentication credentials, review and revoke access from trusted devices, and update your password, all within your defined access constraints.

The "Security" section of the Self-Service Console is your central hub for safeguarding your account. It provides information to ensure your digital identity remains protected. Here, you can:

This page displays the following user information:

- My Credentials: View and manage all your registered authentication methods, including their status and last usage. You can activate, deactivate, remove, or add new factors like SMS, Email, Biometric, Passkey, and Authenticator App.
- My Trusted Devices: Monitor and revoke access from devices you have previously marked as trusted, giving you direct control over where your account is active.
- Password: Review your last password change and update your password, adhering to the enforced password policy requirements for strong security.
- Activity: Access a detailed overview of your recent account-related events, including successful/failed sign-ins, password changes, and account updates. You can also manage your active sessions and refresh tokens.

NOTE

- The self-service console adapts to the scoping constraints set for each user. To access and modify information within the console, your actions must fall within your defined scoping constraints.
- You are currently viewing this page because it aligns with your configured constraint, "DefaultMeConstraints".

My Credentials

This section describes the "My Credentials" section within the Symantec Identity Security Platform Self-Service Console, detailing its purpose and functionalities.

This section displays the user credentials registered with Symantec Identity Security Platform.For each credential, it shows its current status (active or inactive) and the date of its last usage.

- **Toggle Status**: The "Status" button allows users to switch a credential between an active and inactive state. This is useful for temporarily disabling a credential without removing it entirely.
- **Remove Credentials**: The "X" icon next to each credential enables users to remove unused or unwanted credentials from their account.
- **Add New Authentication Method**: The "+ Add Method" button facilitates the registration of new authentication methods.
- **Available Authentication Methods**:
  - SMS Message
  - Email
  - Voice
  - Biometric
  - Passkey
  - Security Key
  - VIP Access App
  - Authenticator App

Authentication Methods

Learn about the various authentication methods supported in the Self-Service console.

Displays the authentication methods and allows the user to select a two-step verification method that best suits them.

Select an option to enhance the security of your enterprise applications. For any authentication method you select, you will first be redirected to re-authenticate before adding the new credential.

SMS Message

If you select this option, you are redirected to the "**Sign In**" page where you need to authenticate with your password. Once authentication is successful, you are redirected to the "Register your phone number" page.

Perform these steps:

1. Enter your phone number along with the country code.
2. Click **Continue**.
3. Enter the code sent to your phone number in the Security Code field.
4. Click **Continue**.
   The newly added SMS credential appears in the My Credential table.

For more information, see Server OTP.

Email

If you select this option, you are redirected to the "**Sign In**" page where you need to authenticate with your password. Once authentication is successful, you are redirected to the "Register your email" page.

Perform these steps:

1. Enter your email Id.
2. Click **Continue**.
3. Enter the code sent to your email Id.
4. Click **Continue**.
   The newly added email credential appears in the My Credential table.

For more information, see Server OTP.

Voice

If you select this option, you are redirected to the "**Sign In**" page where you need to authenticate with your password. Once authentication is successful, you are redirected to the "Register your phone number" page.

Perform these steps:

1. Enter your phone number along with the country code.
2. Click **Continue**.
3. Enter the code sent to your phone number through the voice message.
4. Click **Continue**.
   The newly added credential appears in the My Credential table.

For more information, see Custom IVR Support

Biometric

If you select this option, you are redirected to the "**Sign In**" page where you need to authenticate with your password. Once authentication is successful, you are redirected to the "Register your phone number" page.

Perform these steps:

1. Enter a friendly name to identify your biometrics.
2. Click **Continue**.
   You are asked to create a passkey that will be saved only in this device.
3. Tap your fingerprint.
   The newly added biometric credential appears in the My Credential table

For more information, see Biometric Authenticator.

Passkey

If you select this option, you are redirected to the "**Sign In**" page where you need to authenticate with your password. Once authentication is successful, you are redirected to the "Assign a name to identify your skey" page.

Perform these steps:

1. Enter a name to identify your passkey.
2. Click **Continue**.
   You are asked to create a passkey that will be saved only in this device.
3. Tap your fingerprint.
   The newly added passkey credential appears in the My Credential table

For more information, see Using Passkey.

Security Key

If you select this option, you are redirected to the "**Sign In**" page where you need to authenticate with your password. Once authentication is successful, you are redirected to the "Assign a name to identify your security key" page.

Perform these steps:

1. Enter a name to identify your security key in the **Device Name** field.
2. Click **Continue**.
   You are asked to use USB security key that will be saved only in this device.
3. Tap on the seurity key.
   The newly added Security Key credential appears in the My Credential table

For more information, see Security Key Authentication.

VIP Access App

The VIP Access app provides an extra layer of security by generating a credential ID and a one-time passcode (OTP) on your desktop computer or mobile which you need to enter in the Self-service console to register it as one of the authentication methods.

If you select this option, you are redirected to the "**Sign In**" page where you need to authenticate with your password. Once authentication is successful, you are redirected to the "VIP Access App" page where you need to register your device (Mobile or Desktop) using the Security Codes generated by the VIP Access App.

> **NOTE**
> When you install the VIP Access App in your mobile, the Mobile OTP and Push capabilities are activated for that installation by default. However, if you install the VIP Access desktop app, it will only allow you to execute Mobile OTP capabilities and not allow you to execute Push. To use Push flows, you must install and register with the VIP Access Mobile App only.

Prerequisite:

1. Download the **VIP Access App** on your desktop or mobile based on your device type and operating system. The app generates a **Credential ID** and **Security Code**, which are required for registration.
2. Perform the necessary VIP configuration to allow the tenant or system administrator to integrate Symantec VIP with Symantec Identity Security Platform. For more information on **VIP Configuration**, see here.

Steps to Register VIP Access App in Self-Service Console

After fulfilling the prerequisites, navigate back to the Self-Service console.

1. In the Self-Service Console, go to **Home > Security** or navigate to **Security** in the left panel.
2. Click **Add Method** and choose **VIP Access App**.
   The VIP Access App screen is displayed.
3. Enter Registration Details -
   a. Enter the **Credential ID** generated by the app. (this remains the same for future use)
   b. Enter the **security code** generated by the app. (refreshes every 30 seconds)
   c. Enter a **friendly name** to this authentication method.
   d. Click **Continue**.
      The newly registered VIP Access App credential appears in the **My Credentials** table.

For more information on VIP Push, see VIP Push Authenticator.

Authenticator App

If you select this option, you are redirected to the "**Sign In**" page where you need to authenticate with your password. Once authentication is successful, you are redirected to the "Authenticator App" page.

Perform these steps:

1. Install an authenticator app as your authenticator.
   > **NOTE**
   > You can download any desktop or mobile, non-VIP Access authenticator app. For example- Google Authenticator app.
2. Open the authenticator app and scan the QR code to begin registering your device.
   – If you are unable to scan the QR code, click "**Can't Scan a QR Code**" option. IDSP generates a secret. Copy this secret to use it in the authenticator app. In the authenticator app, you may be provided with multiple options like scan QR code, Manual Entry, Import QR images, or Import OTP URLs.
   • If you are not able to use the Scan QR code, choose other option, such as **Manual Entry**, you need to enter a name and the secret generated by the IDSP, and click **OK**.
   A new user is created in the authenticator app and a security code is displayed.

3. Enter the **security code** generated by the authenticator app in IDSP.
4. Enter a **friendly name** for this app.
5. Click **Continue**.
   Your authenticator app credentials are created in the IDSP.
   The newly added authenticator app credential appears in the My Credentials table

For more information on Authenticator Apps, see Mobile OTP Authenticator.

My Trusted Devices

This document describes the "My Trusted Devices" section of Symantec Identity Security Platform Self-Service Console.

"My Trusted Devices" provides a security feature allowing users to review and revoke access from devices they previously marked as trusted, enhancing control over their account security.

Displays the names of trusted devices along with the last time they were used by the user.

- To delete a trusted device, select the device name and click "**X**". Confirm the removal by clicking "**Remove**."

My Known Devices

This page describes the "My Known Devices" section of Symantec Identity Security Platform Self-Service Console.

"My Known Devices" provides a security feature allowing users to review and revoke access from devices they previously marked as known, enhancing control over their account security.

Displays the list of known devices that the user logs in with. It shows a table of known devices. For each device, the following details are provided:

- **DEVICE NAME**: The name of the known device.
- **LOCATION**: Indicates from where the user has logged in.
- **IP ADDRESS**: Indicates the IP Address from where the user has logged in.
- **LAST USED**: Provides the timestamp of the users last usage.

Deleting Known Devices

- To delete a trusted device, select the device name and click "**X**". Confirm the removal by clicking "**Remove**."

Change Password

Learn how to change your password as an internal user based on administrator configurations.

NOTE

- This page is visible to you because you are identified as an internal Symantec Identity Security Platform user. Users configured outside the internal IDStore does not have access to this information.
- Configure "**trackLastLogin**" tenant setting to "**true**" to ensure the "**Change your password**" section display the expected details in the Self-Service console. For more information on this setting, see Configure Tenant Setting for the Self-Service Console.

1. Navigate to the Self-Service Console.
2. In the top-right corner of the Self-Service Console, click the user menu dropdown showing your username.
3. Select "Change Password" from the dropdown menu.
4. Based on administrator configuration, you will see one of two options:
   – **Reset your Password** dialog - Perform the following steps here:

- Select the credential where you want to receive the password reset link
- A password reset link is sent to your verified email or phone number
- Click the "Change your password" link to open the password change page
- Update your password on the password change page

– **Change Password** pop-up - This pop-up displays a message, "You will be signed out of current session. Do you want to continue?".
  - Click "Continue" to proceed
  - You will be signed out and need to sign in again
  - Complete authentication challenges as configured by your administrator
  - After successful authentication, you will be redirected to the "Reset Password" screen

    NOTE

    Password Requirements: These password requirements are displayed based on the Password Policy configuration in IDSP internal db. See Password Management in Identity Service.

    The following password requirements must be kept in mind when changing your password.

    - Cannot have length less than 8 character(s)
    - Cannot exceed the length of 20 character(s)
    - Should contain at least 1 upper-case letter(s)
    - Should contain at least 1 upper-case letter(s)
    - Should contain at least 1 lower-case letter(s)
    - Should contain at least 1 numeral(s)
    - Cannot contain less than 1 non alpha-numeric character(s)
    - Cannot contain more than 3 non alpha-numeric character(s)
    - Should contain at least 2 unique character(s)
    - Can contain # % @ ! ^ : + . ~ | ? _ \ - $ /
    - Cannot be the same as last 3 password

Third-Party Applications

Keep track of your connections to third-party apps and services.

The Third-Party Applications allows users to manage and monitor their connections to third-party applications and services. It provides a centralized interface to view which external applications have been granted access to the user's account and enables control over these permissions through access removal capabilities.

Navigate to the "User Menu" dropdown in the Self-Service Console.

1. Locate User Menu: In the top-right corner of the Self-Service Console, click the user menu dropdown showing your username.
2. Access Dropdown Menu: The dropdown displays several options:
   – Third-party Apps: View the third-party apps that you have access to.
   – Generate Token: Direct access to PAT generation
   – About: Platform version and information
   – Sign Out: Securely logout from the platform
3. Click Third-party Apps. The Third-party Apps and Services screen is displayed.
4. In My Third-party Applications and services, you can search for the third-party apps and services that the user has access to.
5. Viewing application details: - Click the dropdown arrow next to any application name to expand details to view the following details:

  - Access given on: The date when you first granted access
  - Web address: The application's website URL
  - Permissions: What the application can access (if available)
6. Removing Access:
  - Click the blue "Remove Access" button next to any application
  - This step revokes the application's access to your account
  - The application is removed from your list

My Access

View the applications to which you have access.

The **Access** tab displays a list of applications that the user has access to. It shows a table of applications. For each application, the following details are provided:

- **APPLICATION**: The name of the application.
- **ROLE**: If any roles have been assigned to this user, the number of roles granted appears. You can click on the role value to open the edit roles dialog. You can revoke access to this role by clicking the ellipses button.
- **DESCRIPTION**: A brief description of the application.
- Generate Token: Direct access to Personal Access Token (PAT) generation

User Menu

Navigate to the "User Menu" dropdown in the Self-Service Console.

1. Locate User Menu: In the top-right corner of the Self-Service Console, click the user menu dropdown showing your username.
2. Access Dropdown Menu: The dropdown displays several options:
  - Generate Token: Direct access to PAT generation
  - About: Platform version and information
  - Sign Out: Securely logout from the platform

Generation of Personal Access Token (PAT)

Learn about generation of personal access token in Self-Service Console.

Personal Access Tokens (PATs) provide a secure method to authenticate with Application APIs without using traditional username/password credentials. This topic describes the process of generating PATs in the Symantec Identity Security Platform .

Navigation

Navigate to the "User Menu" dropdown in the Self-Service Console.

1. Locate User Menu: In the top-right corner of the Self-Service Console, click on the user menu dropdown showing your username.
2. Access Dropdown Menu: The dropdown displays several options:
  - Select Generate Token: Click "Generate Token" from the dropdown menu to directly access the PAT generation workflow.

Steps to generate PAT

When selecting the Generate Token option in the User Menu, the PAT generation dialog opens.

1. **Generate Personal Access Token**:

- Search or Select the application from the apps table for which you want to generate a PAT, and click Select.

2. **Configure Token Settings**: In the PAT generation dialog, configure the following:
   - Personal Access Token expiring in : Set duration in seconds (Max: 3600 seconds)
   - Personal Refresh Token expiring in : Set duration in days (Max: 7 days)
   - Refresh Token Description: Enter a descriptive name for your refresh token in the text field. This helps identify the token's purpose later.

   Token Expiration Settings -
3. **Select Privileges**: Use the search bar to find specific privileges by name and description.
   - Privilege Selection:
     - Review available privileges in the table showing-
       - PRIVILEGE: Name of the privilege
       - DESCRIPTION: What the privilege allows
       - ROLE: Associated role
   - Check the boxes next to required privileges.
4. **Generate Tokens**:
   - Review all settings and selected privileges.
   - Click "Generate" to create the tokens.
   - The system processes your request and generates both:
     - Personal Access Token
     - Personal Refresh Token
5. **Download and Store Tokens**:
   - Token Download Dialog: A "Token Download" dialog appears with the message: "Copy your token(s) and store them securely. You won't be able to view them again after you click Close."
   - Available Actions for Each Token:
     - Personal Access Token: Click "Copy to Clipboard" to copy the token | Click "Download" to save as a file.
     - Personal Refresh Token: Click "Copy to Clipboard" to copy the token | Click "Download" to save as a file.
6. **Secure Storage**:
   - Copy both tokens immediately.
   - Store them in a secure location (password manager, encrypted file, etc.)

     IMPORTANT
     Tokens cannot be viewed again after closing this dialog.
7. Click "Close" when you have securely stored both tokens.

Self Service Console for Admins

Learn how to set up and configure the Self-Service Console to enable users to manage their own accounts while maintaining proper administrative controls and security constraints.

This section pertains to configurations, deployments, and settings that are managed by an administrator.

Configure Tenant Settings for the Self-Service Console

Configure the following tenant settings using the `https://${SSP_FQDN}/{{tenantName}}/admin/v1/Configs` API to allow Self-Service Console to disambiguate a user based on the ID Store configuration:

Table 123: Tenant Settings

Tenant Setting	Description
selfServiceConsoleIdStoreToBeUsed	This tenant setting represents a list of ID Stores to be used (as a search order) for disambiguating Self-Service Console users going through the signin ceremony when signing into the Self-Service Console. If this property is not configured, the system fallsback to the "**idStoreToBeUsed**" setting in the tenant configuration. **Default Behavior**: Default Value: "" However, in case of fresh install, if this setting is empty, it automatically sets to the "internal" id store. **Possible values**: Comma-separated IDs of the identity stores.
trackLastLogin	This tenant setting is to track whether the last authentication was a success or a failure. **Default Behavior**: By default it is set to "false" which means the data related to last authentication success or failure does not get tracked. Configure this setting to "true" to ensure the "**View your activity**", and "**Change your password**" sections display the expected details in the Self-Service console.

These tenant settings are also covered in the Managing Settings page.

There are other tenant settings that an administrator can configure. See Self-Service Console Tenant and Global Settings.

Disabling the Self-Service Console Deployment

To disable the deployment of the Self-Service Console during the Symantec Identity Security Platform (SSP) helm chart deployment, include this additional parameter with the IDSP deployment command:

```
--set ssp.featureFlags.selfserviceconsole.enabled=false
```

By default, this feature flag is set to "**true**".

This parameter is also covered in the install section. See Disabling Self-Service Console Deployment.

Constraints used in the Self Service Console

The self-service console experience is tailored based on the scoping constraint, "DefaultMeConstraints". To access and modify information within the console, your actions must fall within your defined scoping constraints.

The user attributes displayed on each page of the Self Service Console are derived from the scoping constraints set by your administrator.

To know about Scoping Constraints, see the following topics:

- Overview of Scoping Constraints
- Understanding and Using the Default Constraint
- Localization of Self-Service Console

Cookies Used in Symantec Identity Security Platform

The Symantec Identity Security Platform Signin UI (Web) application stores the following client-specific information between different requests or sessions.

Table 124: The following table lists the data stored by the Signin UI App in the browser:

Name	Value	Type	Expiry	Purpose	Module
__Secure-ob-XXXX	Factor corresponding to a level	Cookie	365 days	This cookie stores information about a particular user's (XXXX) choice for the factor corresponding to a level. For example, if the user chooses FIDO as the second factor and PUSH as the third factor in an MFA scenario, the user will not be presented with the factor selection screen the next time but will be taken directly to the user's chosen factor screen. The username in the cookie is obfuscated.	Signin
__Secure-ssp-username	username	Cookie	365 days	This cookie is used for the "Remember Me" functionality. For example, if the user selects the "Remember Me" checkbox and completes authentication, then the user will not be presented with the username screen the next time but will be taken directly to the next step, which is, verify screen.	Signin
__Secure-fid-XXXX	FIDO Identifier	Cookie	Driven by the tenant setting: "fidoIdentifierExpiryDays" default: 365 days	This cookie stores a user's unique FIDO Identifier (XXXX) for associating a Biometric (Platform Authenticator) credential with a user on a given device (browser). The username in the cookie is obfuscated. For example, if the user is 'nbruce', the cookie name contains the obfuscated text for 'nbruce'.	Signin
__Secure-ssp-aa	AARISK DEVICEID	Cookie	365 days	This cookie contains the AARISK DEVICEID which is used during future authentications to establish the device and user association.	Signin
_iat1	Risk related information	Cookie	365 days	This encrypted device signature is generated and consumed by the IARisk JavaScript SDK.	Signin
_ia01	Risk related information	localStorage	N/A	This encrypted device signature is generated and consumed by the IARisk JavaScript SDK.	Signin

_ia01_timestamp_	Risk related information	localStorage	N/A	This encrypted device signature is generated and consumed by the IARisk JavaScript SDK.	Signin
<tenant>.ssp-signInAppPath	Base href path of Signin for loading static files	sessionStorage	session	This is a functional value needed for determining the correct path to load static files in the browser.	Signin
loggingEnabled	true/false	sessionStorage	session	This data indicates whether logging should be enabled in the Browsers console for debugging purposes.	Signin
<tenant>.ssp-appInitialUrl	Initial URL with which Signin was loaded	sessionStorage	This data is removed at the end of the authentication flow, that is, either after AUTH_ALLOWED or AUTH_DENIED	This data stores the initial URL used to load Signin so that it can be used at a later point of time during "Not User?" navigation.	Signin
<tenant>.ssp-x-service-baseurl	The IDSP Service Base URL	sessionStorage	N/A	This data stores the service baseURL with which Signin makes the IDSP API calls.	Signin
<tenant>.ssp-appRenderMode	Rendering Mode	sessionStorage	N/A	This indicates whether the app is rendered in SignIn mode or Devicecode Mode.	Signin

Reference Links to:

- Tenant Settings

API Error Codes

Displays the list of error codes and messages for the APIs.

You can use the error codes mentioned for each of the services to understand and resolve issues that you may encounter while running the APIs.

This topic includes the following links:

- Auth Manager Error Codes
- ID Store Error Codes
- Application Error Codes
- Factor Error Codes
- Provider Error Codes
- Policy Error Codes
- Credential Error Codes
- OAuth/OIDC Error Codes
- Risk Error Codes
- Signing Key and Certificate Error Codes
- API Security Error Codes
- Message Error Codes
- Settings Error Codes
- VIP Integration Error Codes
- assetlinks.json error codes
- Identity Service SCIM Error Codes
- Password Management API Error Codes

Auth Manager Error Codes

Auth Manager Error Codes

The following table provides information about the error codes that may be returned by the Symantec Identity Security Platform Auth Manager APIs.

Table 125: Error Codes for Auth Manager APIs

Error Codes	Error Messages
8800001	Authorization header missing with API Key
8800002	User name is required
8800003	Invalid API Key
8800004	Invalid user, LDAP lookup failed
8800005	Unknown error
88000010	Credential not found
88000011	Invalid Request
88000012	Invalid User
88000013	Missing required request header
88000014	JSON parse error

88000015	User is inactive
88000016	OPA Policy evaluate call failed
88000017	Invalid authorization header
88000018	Invalid IP address
88000019	Message not found
88000020	Provider not found
88000021	Message already present
88000022	Message not deleted as it is in English language

ID Store Error Codes

The following table provides information about the error codes that are returned by the Symantec Identity Security Platform ID Store Configurations APIs.

Table 126: Error Codes for ID Store Config APIs

Error Codes	Error Messages
1003000	Success
1003001	Invalid configuration ({0})
1003002	Unable to connect ({0})
1003004	Invalid connection credentials
1003005	Invalid search query ({0})
1003006	Invalid Base DN ({0})
1003007	Partial search results ({0})
1003008	Connection timed out
1003009	Invalid configuration - missing required configuration parameter(s) ({0})
1003010	Invalid SSL certificate
1003011	Multiple records found for user ({0})
1003013	Exception in creating a trust store ({0})
1003014	Failed to resolve id store for config type ({0})
1003015	Unable to retrieve connection details
1003016	User with ID ({0}) not found
1003017	Invalid request parameter
1003018	Provide a valid user name
1003019	Unable to complete requested id store operation
1003020	Length of parameter ({0}) more than expected
1003021	Unsupported encoding
1003022	Unable to update lockout state for user: Missing status encoding value in ID Store configuration
1003025	Not supported
1003026	Provided the secret alias is not valid
1003027	Provided certificate Alias is not valid
1003028	Provided secret Alias is not having the required type 'rsakey' in the secrets store
1004002	Password authentication failed: user ({0}) is locked

1004003	Password authentication failed: invalid credentials for user ({0})
1005001	Internal ID Store error
1005002	Configuration not found for Tenant ID
1005003	Configuration not found for Tenant ID and Config ID
1005004	Configuration exists
1005005	Tenant with ID ({0}) does not exist
1005006	Configuration not found for Tenant ID and Config Name

Error message displayed for Inactive User

In case of Authentication failure due to an INACTIVE user, the authentication response displays an error code with an error message, similar to this example, **`errorCode:`** `8800015 &` **`errorMessage:`** `"user is inactive..."`

However, in the case of LDAPconfig's statusEncoding = "SiteMinder," additional information is provided in the error message about the inability to authenticate, with "value" representing SMAuthReason as defined by LDAPconfig as per this Knowledge Article. The errorCode and ErrorMessage would look something like this:

```
[8800041] [value=<status-attribute-value>:<sm-auth-reason>,encoding=siteminder]
```

For example:

```
{
"errorCode": "8800015",
"errorMessage": "The user in the request is not active. [8800041] [value=2:7,encoding=siteminder]"
}
```

Reference Links to:

- ID Store Configuration

Application Error Codes

The following table provides information about the error codes that are returned by the Symantec Identity Security Platform Application APIs.

Table 127: Error Codes for Application APIs

Error Codes	Error Messages
8000003	Application Name is missing
8000005	At least one redirectURI is required
8000006	At least one scope is required
8000007	At least one grant type is required
8000008	One of the grant types is invalid
8000009	Invalid status
8000010	One of the value in 'allowedOperations' is blank
8000011	One of the value in 'allowedOperations' is invalid
8000012	Application with the name already exists
8000013	Application with the given id does not exist
8000014	Unknown exception
8000015	Invalid application Id

8000016	Application ID is missing
8000017	Invalid user Id
8000018	Invalid tenant Id
8000019	One of the scope value is empty
8000020	One of the redirect_uri value is empty
8000022	JSON parse error
8000024	Missing client type
8000025	Invalid client type
8000028	client_credentials or 'urn:ietf:params:oauth:grant-type:jwt-bearer' grant is not allowed for the public client
8000029	Invalid UserInfoEndpointResponseFormat. Supported values: PLAIN_JSON, B64_JSON, JWT
8000032	Invalid assertion verification cert key alias
8000033	'assertionVerificationCertAlias' is not allowed for public and confidential client types.
8000034	Invalid flow URL
8000035	Missing assertion verification cert key alias
8000036	Grant type 'refresh_token' is allowed only in combination of 'authorization_code' or 'urn:ietf:params:oauth:grant-type:jwt-bearer' grant type
8000037	Application associated with one or more policies cannot be deleted

Factor Error Codes

The following table provides information about the error codes that may be returned by the Symantec Identity Security Platform Factor APIs.

Table 128: Error Codes for Factor APIs

Error Codes	Error Messages
Password Authenticator Error Codes	
0000012	Internal server error
3001045	Password authentication failed: invalid credentials
3001151	Invalid ID Store handle
3001033	Incorrect credentials
1003033	Password must be changed
FIDO Error Codes	
2001001	Invalid username
2001002	Invalid/Missing device name
2001003	Missing type in client data JSON
2001004	Invalid attestations verify
2001005	Unable to parse client JSON message
2001006	Invalid client JSON parameters
2002007	Missing challenge in client data JSON
2001008	Invalid token binding
2001009	Invalid origin

2001011	Invalid status
2001012	User device combination exists
2001013	Invalid attestation Id
2001014	Invalid attestation type
2001015	Error while parsing date
2001016	Invalid/missing attestation raw id
2001018	Invalid attestation
2001019	Invalid tenant in config
2001020	Challenge record exists
2002006	Invalid Certificate
2003000	No user registration data found
2003002	Cannot find the associated key. Have you registered?
2003003	Invalid assertion type
2003004	Invalid assertion raw Id
2003005	Invalid authenticator data
2003006	Invalid signature
2003008	Invalid format
2003009	Cannot find a request with matching challenge and domain
2003010	Timeout
2003011	Request a timeout
2003012	Invalid domain
2003014	Invalid assertion Id
2003015	Device is locked or inactive
2003016	Challenge record exists
2003017	no user found in ldap
2003018	invalid domain
2003019	invalid fidoCredType
2003020	Count of registered devices reached the maximum allowed count
2003021	Missing/invalid Transaction Data
2003022	Transaction verification failed
2004000	Invalid data in the processor
2004001	Invalid aaguid
2004002	User not present
2004003	User not verified
2004004	Given encoding is not supported
2004005	Hashes do not match
2004006	Unable to verify the signature
2004007	Counter did not increase
2004008	General server error
2004009	Wrong Algorithm
2004010	Problem with processing authenticator data

2005000	No credential information found
2005002	Failed to save in database
2005003	Failed to delete from database
2010001	Invalid tenant
3001046	currentPassword and newPassword are the same
3001047	Invalid/Missing currentPassword
3001048	Invalid/Missing NewPassword
3001049	Current password and the existing password do not match
Mobile OTP Error Codes	
3001042	Credential value is missing
3001043	Invalid OTP value in input payload
Server OTP Error Codes	
3001027	Mobile number or email Id missing
3001028	Credential already present
3001029	Provide a valid mobile number
3001030	Provide a valid email address
3001031	Credential not found
3001032	No of authentication attempts exceeded
3001033	Incorrect authentication credentials
3001035	Credential not active
3001036	Invalid request
3001037	Credential not associated with the user
3001038	Mandatory header missing
3001039	Invalid type in input payload
3001051	OTP is expired
Push Error Codes	
3001093	Failed to verify as PUSH notification was not generated
3001098	Device name missing
3001099	Device token missing
3001101	Invalid Flow State Id
3001102	Push Registration Initiation Failed
3001103	Push Registration Timed Out
3001106	Push data missing
3001107	Credential type not PUSH
3001108	VIP SOAP API response empty
3001109	Failed to send Push Notification
3001110	Missing VIP Push TransctionId
3001111	VIP Pushes Authentication response empty
3001112	PushContext not found
VIP Error Codes returned by Mobile OTP and Push	
3001104	CredentialId missing

3001105	Credential data missing
3001120	Invalid VIP Credential Id
3001121	Invalid VIP Credential Type
3001122	Bad VIP Credential State
3001123	VIP Credential has Expired
3001124	Invalid VIP Credential
3001125	Invalid VIP Certificate
3001126	Entered Credential ID does not support PUSH. To enable PUSH, use the Credential ID generated from the VIP Access app.
3001127	Missing OTP in the request payload
3001128	Invalid VIP Credential Status
3001129	Missing VIP GetCredentialInfo API response
3001130	VIPConfiguration not configured for the Tenant
Identity Provider (IDP) Error Codes	
8000042	Invalid IDP metadata, could not be parsed
8000043	The Identity Provider corresponding to provided identityProviderId does not exist
8000044	dpName is missing in the request
8000045	domainName is missing/invalid in the request
8000046	Invalid identity provider configuration
8000048	Identity Provider with the provided idpName/domainName exists
8000049	The idpType can only be saml or openid
8000050	Invalid samlAuthnRequestBinding, it can only be one of the following: • "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" • "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect""
8000051	Custom AMR values are missing
8000052	issuerName is missing in the request
8000053	idpTokenSubjectClaim is missing in the request
8000054	spMappingAttribute is missing in the request or is null
8000055	verifyCertAlias cannot be blank
8000056	Base64 encoded IDP metadata is not provided
JWT Token Error Codes returned by Factors	
3001090	Incorrect device token in JWT
3001091	JWT token has expired
3001092	Failed to verify JWT token
Custom Service Provider Interface (SPI) Error Codes	
8800042	Message delivery failed for SPI
3001142	Custom SPI provider attributes missing
3001143	Failed to send Push Notification to custom SPI Push
3001144	Custom IVR SPI provider not configured

Provider Error Codes

Table 129: Error Codes for Provider Configuration APIs

Error Codes	Error Messages
8800020	Provider not found

Policy Error Codes

API Error Codes

The following table provides information about the error codes that are returned by the Symantec Identity Security Platform Policy APIs.

Table 130: Error Codes for Policy APIs

Error Codes	Error Messages
8800202	Policy not found
8800204	Invalid policy Id
8800206	Invalid policy schema
8800255	'operator' is required for the user principal
8800256	'operator' is required for the group principal
8800257	'operator' is required for the clientApp principal
8800258	'operator' is empty for the user principal
8800259	'operator' is empty for the group principal
8800260	'operator' is empty for the clientApp principal
8800261	'value' is required for the user principal
8800262	'value' is required for the group principal
8800263	'value' is required for the clientApp principal
8800264	'value' is empty for the user principal
8800265	'value' is empty for the group principal
8800266	'value' is empty for the clientApp principal
8800285	Invalid app Id in the path
8800286	Invalid app Id in the value
8800308	Missing principal condition
8800309	Invalid syntax for principal value
8800310	Unsupported characters in one or more of the properties
8800315	Remove operation cannot be performed as principalConditionRequiredInPolicy is enabled
8800324	Invalid constraint Id in the policy

Credential Error Codes

The following table provides information about the error codes that are returned by the Symantec Identity Security Platform Credential APIs.

Table 131: Error Codes for Credential APIs

Error Codes	Error Messages
8800010	Credential not found
8800011	Invalid request
8800012	Invalid user
8800244	Invalid tenant Id

OAuth/OIDC Error Codes

OIDC Error Codes

The following table provides information about the error codes that may be returned by the Symantec Identity Security Platform OIDC APIs.

Table 132: Error Codes for OIDC APIs

Error Codes	Error Messages
8100000	Server error. Failed to process request
8100001	"grant_type" is missing
8100002	"response_type" is missing
8100003	"client_id" is missing
8100004	"client_secret" is missing
8100005	"redirect_uri" is missing
8100006	authorization header is missing
8100007	"scope" is missing
8100008	"code" is missing
8100009	Invalid client_id
8100010	Invalid redirect_uri
8100011	Invalid grant_type
8100012	Invalid client credentials
8100013	Invalid code
8100014	"code" has expired
8100015	Both the authorization header and client_id parameter should not exist.
8100016	Invalid authorization header
8100017	Invalid id_token_hint
8100018	"access_token" is missing
8100019	Both bearer authorization header and access_token should not exist in the body
8100020	Authorization header must be of type, bearer
8100021	Invalid access_token
8100022	Invalid scope
8100023	Invalid response_type
8100024	GET method is not supported
8100025	Assertion is missing

8100026	Input grant_type allowed only for confidential client
8100027	Invalid assertion
8100028	"token" is missing
8100031	Invalid code_challenge_method
8100032	Invalid code_verifier
8100033	Missing code_verifier
8100034	Unauthorized request
8100035	Unsupported grant_type for the client

Risk Error Codes

The following table provides information about the error codes that may be returned by the Symantec Identity Security Platform Risk APIs.

Table 133: Error Codes for Risk APIs

Error Codes	Error Messages
0	Operation completed successfully
7700000	Unknown error
7700001	Tenant Id is missing
7700002	Failed to evaluate risk
7700003	JSON payload parsing failed
7700004	Invalid user Id
7700005	Invalid parameter-untrustedIpId
7700006	Failed to delete untrustedIpId
7700007	Missing parameter-untrustedIpId
7700008	Invalid parameter-negativeCountryId
7700009	Failed to delete negativeCountryId
7700010	Missing parameter-negativeCountryId
7700012	Invalid parameter-isoCode
7700013	'isoCode' already added as a Negative Country
7700014	Invalid parameter-riskFactorId
7700015	Invalid tenant Id
7700016	Invalid request
7700017	Missing required request header
7700018	Invalid IP Address
7700019	Invalid parameter-riskExceptionUserId
7700020	Failed to delete riskExceptionUserId
7700021	Start/End date is overlapping with an existing entry for the same User Id
7700022	Start date must be before the end date
7700023	Risk UserContext does not exist
7700024	Failed to delete UserContext
7700025	Failed to find UserContext

7700026	Risk UserContext exists
7700027	Invalid risk parameter
7700028	Risk parameters list empty
7700029	Overlapping untrustedIP list in database
7700030	'startIpAddress' cannot be larger than 'endIpAddress' in the untrustedIP list
7700031	User Id is missing
7700032	Missing parameter-reason
7700033	Missing parameter-startDate
7700034	Missing parameter-endDate
7700035	Start date cannot be before the current date
7700036	Missing parameter-isoCode
7700037	Missing parameter-startIpAddress
7700038	Missing parameter-endIpAddress
7700039	Missing parameter-riskScore
7700040	Missing parameter-priority
7700041	Missing parameter-enabled
7700043	'riskScore' value must be between 1-100
7700043	User Id is missing
7700044	IPAddress is missing
7700045	Settings property is not required for this factor
7700046	Invalid parameter-timeIntervalInMinutes
7700047	Missing parameter-timeIntervalInMinutes
7700048	Invalid parameter-numberOfRiskEvaluationsPerUser
7700049	Missing parameter-numberOfRiskEvaluationsPerUser
7700050	Invalid parameter-numberOfRiskEvaluationsPerDevice
7700051	Missing parameter-numberOfRiskEvaluationsPerDevice
7700052	Missing parameter-signatureMatchThreshold
7700053	Invalid parameter-signatureMatchThreshold
7700054	Invalid parameter-maxSpeedUserCanTravel
7700055	Invalid parameter-maxSharedSessionsPerUser
7700056	Invalid parameter-maxDistanceTolerance
7700057	Missing parameter-maxSpeedUserCanTravel
7700058	Missing parameter-maxSharedSessionsPerUser
7700059	Missing parameter-maxDistanceTolerance
7700060	Settings property is required for this factor
7700061	Unknown property under settings
7700062	"priority" cannot be negative
7700063	Risk factors not found in Database
7700064	"priority" cannot exceed the total risk factors count
7700065	DeviceId missing for RiskScoreUpdater request
7700066	Invalid deviceId

7700067	MFP not in proper JSON format
7700068	Reverse lookup failed for device
7700069	New device creation failed
7700070	Lookup for existing devices failed
7700071	Risk Provider configuration not found
7700072	Risk Provider configuration could not be updated
7700073	TransctionId missing during risk evaluation
7700074	Failed to insert UserRisk history record in database
7700075	Failed to update UserRisk history record in database
7700076	Invalid TransctionId passed during userRiskScoreUpdater call
7700077	Missing parameter-riskState
7700078	Missing parameter-transactionId
7700079	Missing parameter-deviceSignature
7700080	Parameter-timeIntervalInMinutes should be greater than 0
7700081	Parameter-numberOfRiskEvaluationsPerUser should be greater than 0
7700082	Parameter-numberOfRiskEvaluationsPerDevice should be greater than 0
7700083	Parameter-signatureMatchThreshold should be greater than 0
7700084	Parameter-maxSpeedUserCanTravel should be greater than 0
7700085	Parameter-maxSharedSessionsPerUser should be greater than 0
7700086	Parameter-maxDistanceTolerance should be greater than 0
7700087	Parameter-signatureMatchThreshold cannot be greater than 100

Signing Key and Certificate Error Codes

The following table provides information about the error codes that are returned by the Symantec Identity Security Platform Signing Key and Certificate APIs.

Table 134: Error Codes for Certificate APIs

Error Codes	Error Messages
9000001	Tenant Id is missing
9000002	Certificate Id is missing
9000003	Alias is missing
9000004	certificateType is missing
9000005	Invalid b64KeyData, Only DER format supported
9000006	Invalid b64CertData, Only DER format supported
9000007	Invalid certificateType, CertificateType must be either CERTIFICATE or KEYCERT
9000008	Certificate with the given alias exists
9000009	b64KeyData is missing
9000010	b64CertData is missing
9000011	Invalid alias
9000012	Invalid certificateId
9000013	Entry exists with the same issuer dn and serial number

9000014	Certificate is expired
9000015	Certificate is issued before the current date
9000016	Invalid tenant id
9000017	Invalid user id
9000018	Cannot delete as the Certificate is referenced in the Application
9000019	Certificate Type is CERTIFICATE but b64keydata provided
9000020	Alias cannot be updated as it is referenced in the Application
9000022	Certificate not provided
9000023	Invalid issuer
9000024	'tokenIssuer' is required
9000025	Entry exists with the same issuer
9000026	Failed to save private key for the tenant

API Security Error Codes

The following table provides information about the error codes that are returned by the Symantec Identity Security Platform Security APIs.

Table 135: Error Codes for Security APIs

Error Codes	Error Messages
0000003	Missing authorization header
0000004	Failed to decode access token from authorization header. Token must be of type, bearer or internal
0000005	Invalid access token
0000007	Unprivileged AccessToken. Cannot authorize the request with the given access token
0000008	Authorization failed

Message Error Codes

The following table provides information about the error codes that are returned by the Symantec Identity Security Platform Message Templates APIs.

Table 136: Error Codes for Message Templates APIs

Error Codes	Error Messages
8800021	Message already present
8800022	Message not deleted as it is in the English language
8800023	Invalid language
8800024	Invalid message type
8800025	Empty message provided
8800012	Invalid user
8800019	Message not found

Settings Error Codes

The following table provides information about the error codes that are returned by the Symantec Identity Security Platform Settings APIs.

Table 137: Error Codes for Settings APIs

Error Codes	Error Messages
91000001	Expected values for {1} are {2}
91000002	Property not found
91000003	Given Tenant is not found
91000004	Expected values for {1} are between {2}
91000005	Expected value for {1} is {2}

VIP Integration Error Codes

The following table provides information about the error codes that are returned by Symantec Identity Security Platform Symantec VIP Integration APIs.

Table 138: Error Codes for VIP Integration APIs

Error Codes	Error Messages
0000020	VIP Account Name missing
0000021	VIP Query Service URL missing
0000022	VIP Authentication Service URL missing
0000023	VIP Certificate data in encoded form is missing
0000024	VIP Certificate password missing
0000025	VIP Configuration DB update failed
0000026	VIP Configuration DB insert failed
0000027	VIP Configuration DB delete failed
0000028	VIP Configuration delete not allowed for System Tenant
0000029	VIP SOAP API response empty
0000030	Invalid VIP SOAP API Status
0000031	Unable to create Stub to VIP Services
0000032	VIP SOAP API GetServerTime failed
0000033	Unable to load VIP Certificate
0000034	Unable to connect with VIP Services. Check the VIP connection details and the Proxy settings
0000035	VIP Configuration request payload missing
0000036	VIP Certificate data is missing in the request payload
0000037	VIP Certificate password is missing in the request payload
0000038	Invalid tenant id
0000039	VIP Configuration not configured for the Tenant
0000040	Unable to delete as VIP Configuration not configured for the Tenant
0000042	Invalid VIP Credential registration response

assetlinks.json error codes

The following table provides information about the error codes that are returned by the Symantec Identity Security Platform assetlinks.json file.

Table 139: Error Codes for assetlinks.json file

Error Codes	Error Messages
91000007	Mandatory fields missing for assetlinks.json
91000008	Invalid type in relation
91000009	One or more values missing in android_app One or more values missing in the namespace
91000010	Invalid/missing value found in namespace Invalid/missing value found in site
91000011	Unknown parameters found in namespace Unknown parameters found in the namespace for android_app Unknown parameters found in assetlinksJson
91000012	Invalid input/value in assetlinksJson

Identity Service SCIM Error Codes

Error Code	Error Description
89400000	Error during SCIM operation. *Note: This default value is never explicitly used.*
89400001	Request is un-parsable, syntactically incorrect, or violates schema
89400002	The specified version number does not match the resource's latest version number, or a service provider refused to create a new, duplicate resource
89400003	An internal error.
89400004	Requested resource not found
89400005	Requested API not implemented
89400006	Authorization failure
89400000	Operation is not permitted based on the supplied authorization.
89410100	Post User error
89410101	Put User error
89410102	Patch User error
89410103	Get User error
89410104	Get List Users error
89410105	Post List Users error
89410106	Delete user error
89410110	User name must be unique.
89410111	User with this ID exists.
89410112	Deletion is already in progress.
89410113	Specified sort attribute is not valid.
89410114	Error in tokenization of the input filter

89410115	Specified filter on attribute is not supported
89410116	Unable to update user.
89410117	User name does not match an existing user name.
89410118	User id is not unique.
89410119	Unable to update user.
89410120	Password invalid.
89410121	Password does not meet password policy requirements.
89410122	Both passwordCannotBeChanged and passwordMustChange attributes cannot be 'true'
89410123	Manager not found.
89410124	Invalid Admin privileges for internal ID store user.
89410125	Email address(es) are are not unique.
89410126	Only one address can be primary
89410127	Missing activation credentials
89410130	SSP Extension attribute error
89410131	Custom attribute error
89410132	Credential attribute error
89410133	Enterprise Extension attribute error
89410150	The replacement manager specified is invalid
89410151	Users cannot reset their own manager
89410152	Users cannot be their own manager
89410200	Post Group error
89410201	Put Group error
89410202	Patch Group error
89410203	Get Group error
89410204	Get List Groups error
89410205	Post List Groups error
89410206	Delete Group error
89410210	Invalid sort attribute.
89410212	Invalid filter operator
89410214	Error updating Group
89410215	Group name not unique
89410217	Error updating Group members
89410218	Duplicate adminScope value.
89410219	The user cannot be added to a group because user's idpOriginId does not match token idpOriginIds
89410220	The user cannot be removed from a group because user's idpOriginId does not match token idpOriginIds
89410301	Updating Me resource error
89410302	Retrieving Me resource error
89410303	Updating Me resource error
89410304	Updating Me resource error
89410305	Retrieving Me resource error
89410306	Missing user login id.

89410307	Authorized user login id must match the user name in payload.
89410400	Bulk operation error
89410401	Bulk payload size exceeds maximum.
89410402	Bulk operations number size exceeds maximum.
89410403	CIM Bulk request body attribute 'data' is missing sub-attribute 'operations'
89410500	Server Provider Config error
89410600	Schema error
89410700	Resource Types error
89410800	Filter parameter is unparsable or syntactically incorrect
89410851	User idpOriginId does not match token idpOriginIds
89410852	Invalid idpOriginId
89410853	You are not authorized to get/modify/delete this user (user's idpOriginId does not match token idpOriginIds)
89410855	Invalid originId
89410857	You are not authorized to set this user as manager (user's idpOriginId does not match token idpOriginIds)
89410858	Unable to delete the group as it is read-only.
89410860	Query timeout Applies to user and group searches only.

Password Management API Error Codes

The Password Authenticator API (`PasswordAuthenticator`) returns specific response codes that are unique to that request:

Error Code		Error Description
200		Good credential, no response
4xx - Bad credential or other operational issue errors. The response JSON contains more error information		
	400	Bad credential, lockout counter will be incremented
	401	Unauthorized operation by client
	403	User disabled 404 - user not found
	412	Bad credential, user will be locked on next attempt
	423	User locked
	428	Good credential, user must change password
500		Internal server error

Deleting Symantec Identity Security Platform Objects Error Codes

Error Codes for deleting resources that are in use by other IDSP objects

The following table provides information about the error codes that may be returned by the Symantec Identity Security Platform object deletion that are in use by other resources.

Table 140: Error Codes for deleting resources that are in-use by other IDSP objects

Deleting Resource Objects	Error Code
Deleting certificate alias, key alias or secret alias when in use by other IDSP objects	80000134
Deleting application, when in use by other IDSP objects	8000037
Deleting identity provider, when in use by other IDSP objects	8000136
AppGroup associated with one or more policies cannot be deleted.	8000143
Custom Provider associated with one or more policies cannot be deleted.	8000146
Azpolicy associated with one or more AuthN policies cannot be deleted	8000151

Integrating

This section is written for administrators who are responsible for customizing and enabling integrations between Symantec Identity Security Platform and other products.

This section includes the following topics:

- Integration of SiteMinder with IDSP

Integration of SiteMinder with Symantec Identity Security Platform

Overview

Symantec Identity Security Platform provides enterprises with modern, secure user authentication capabilities for mobile client and web applications. It also serves as a common authentication service for other products in Broadcom, such as SiteMinder, Privilege Access Management, and APIM. IDSP helps enterprises to integrate authentication services into the development of modern applications. It is designed to seamlessly incorporate authentication policy flows into the user's daily workflow. For more information, see IDSP documentation.

Benefits of the Integration

With the integration of SiteMinder with IDSP, enterprises can roll-out true passwordless authentication using the FIDO2 standard to help protect the online identities of employees, partners, or consumers in both federation and non-federation transactions. For example, if an organization is using SiteMinder as SAML 2.0 IdP or OpenID Connect Provider, this integration can be used for delegating user authentication to IDSP. Also, advanced policies can be enforced based on login context like User IP address, risk scores, and so on, and provide more granular control with abilities to control authentication flows per user or per group.

The benefits include:

- Modern authentication schemes can be used and leveraged
- User authentication can be completely delegated to IDSP
- Use Risk service, a powerful risk-based, adaptive authentication solution that works in real time to evaluate context and calculate a risk score. The risk server calculates the risk score for any given access attempt based on a predefined set of rules. The calculated risk score is fed into the policies in IDSP to decide whether to continue to authenticate the current login request, request step-up authentication and/or send an alert or block the activity. Users are then presented with authentication options appropriate to that risk level. This approach provides the organization with a transparent layer of protection against identity theft, data breaches, and fraud

For information about policies in IDSP, see Managing Policies. For information about risk management, see Managing Risks.

Integration Overview

SiteMinder integration with IDSP uses the standard OpenID Connect protocol where IDSP acts as an OpenID Provider (OP) and SiteMinder acts as a Client/Relying Party (Client/RP). IDSP exposes REST APIs to manage its configuration.

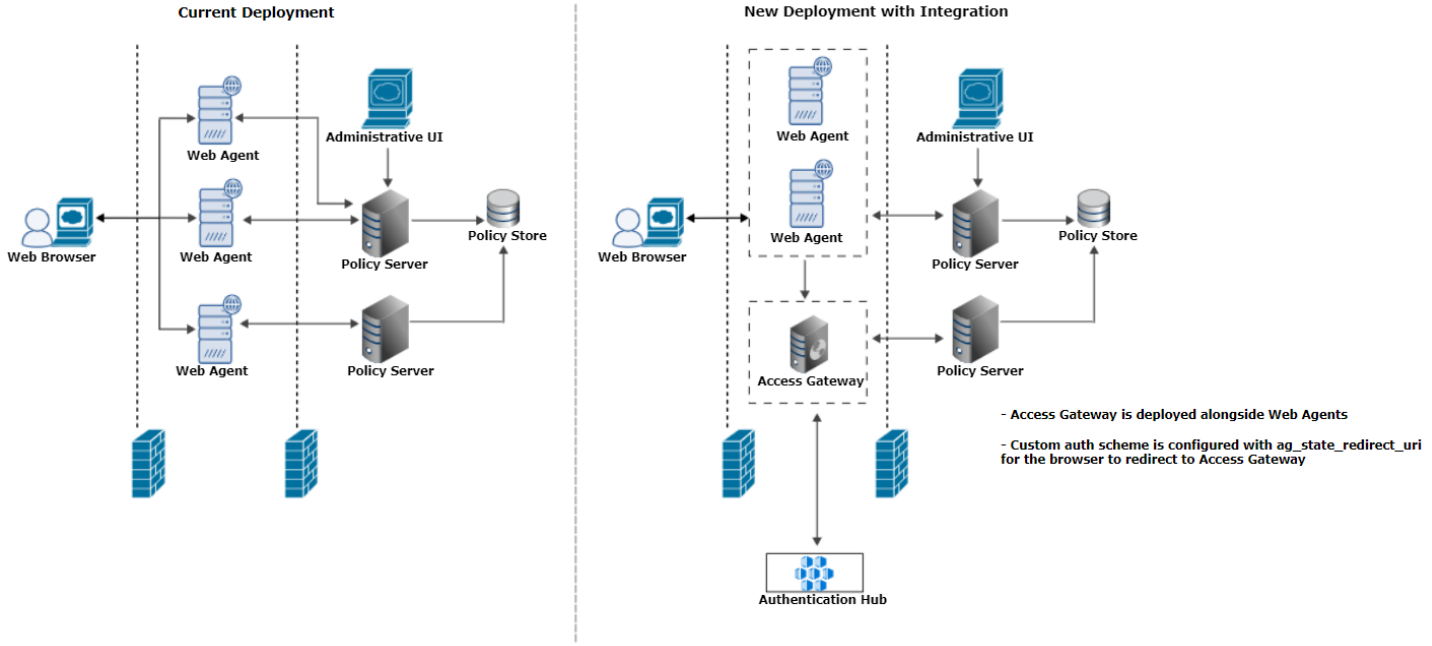
To support the integration, SiteMinder provides two new authentication scheme templates that can be deployed in SiteMinder Release 12.8.06a or in an in-progress SiteMinder release that is delivered along with the IDSP release. These authentication schemes can be easily implemented in existing SiteMinder environments and integrated with IDSP for multi-factor authentication. To test the new authentication scheme templates, perform a fresh installation of this release in your environment.

Integration Deployment

As an example of how the integration can be deployed in an environment, the following diagram details how the integration fits in an existing environment that contains Policy Servers and Web Agents:

Note:

- The integration requires Access Gateway.
- This is an example diagram to show how the integration seamlessly fits in an environment with Web Agents. Access Gateway does not replace Web Agents in this case. If your environment uses Access Gateway instead of Web Agents, you can relate to the simplicity even further.



Support for Authentication Context Class References for Stronger Authentication

AuthN policies of IDSP support multiple Authentication Context Class References (ACRs) to define different authentication flows. ACRs are mandatory and are defined as a part of authentication rules that a user has to satisfy to gain access to a protected resource. In SiteMinder, the ACRs to be used must be specified in the authentication scheme that is used by integration. ACRs define the authentication type that must be invoked for user authentication, thus ensuring stronger authentication security. For information on how ACRs are configured in AuthN policies, see ACRs. For information on specifying ACRs in authentication scheme, see the integration configuration sections that follow.

When SiteMinder requests IDSP to authenticate a user based on a specific ACR, the user is authenticated according to the rule that contains the ACR. Once IDSP authenticates a user, information about the ACR that is used for authentication is sent back to SiteMinder for validation. SiteMinder further validates that the ACR sent in its request matches the ACR that is used for user authentication, and only then proceeds to the next steps.

Let us take an example to understand how ACRs are used in this integration. Consider that IDSP rules in a policy are configured as follows:

Note: For simplicity, only ACR parameter is used in the rule. The other conditions and result of a rule are not considered.

Rule1: "acr": [authCredType1, authCredType3]

Rule2: "acr": [authCredType2]

Rule3: "acr": [authCredType3]

- If SiteMinder specifies authCredType2 in its authentication scheme, Rule2 is triggered if the other conditions of the rule are met.
- If SiteMinder specifies authCredType3 in its authentication scheme, Rule1 is triggered if the other conditions of the rule are met. User is challenged with the authentication methods that are defined in Rule1. Rule3 is not triggered as the condition is satisfied with Rule1 itself.
- If SiteMinder specifies authCredType1 and authCredType2 in its authentication scheme, Rule1 is processed first to check if other conditions in the rule are met. If the conditions are met, Rule1 is triggered. If the other conditions of Rule1 are not met, then Rule2 is processed.

Supported Authentication Scenarios

As part of the integration, SiteMinder supports the following two scenarios:

Scenario 1: SiteMinder performs the primary authentication and IDSP performs the secondary authentication

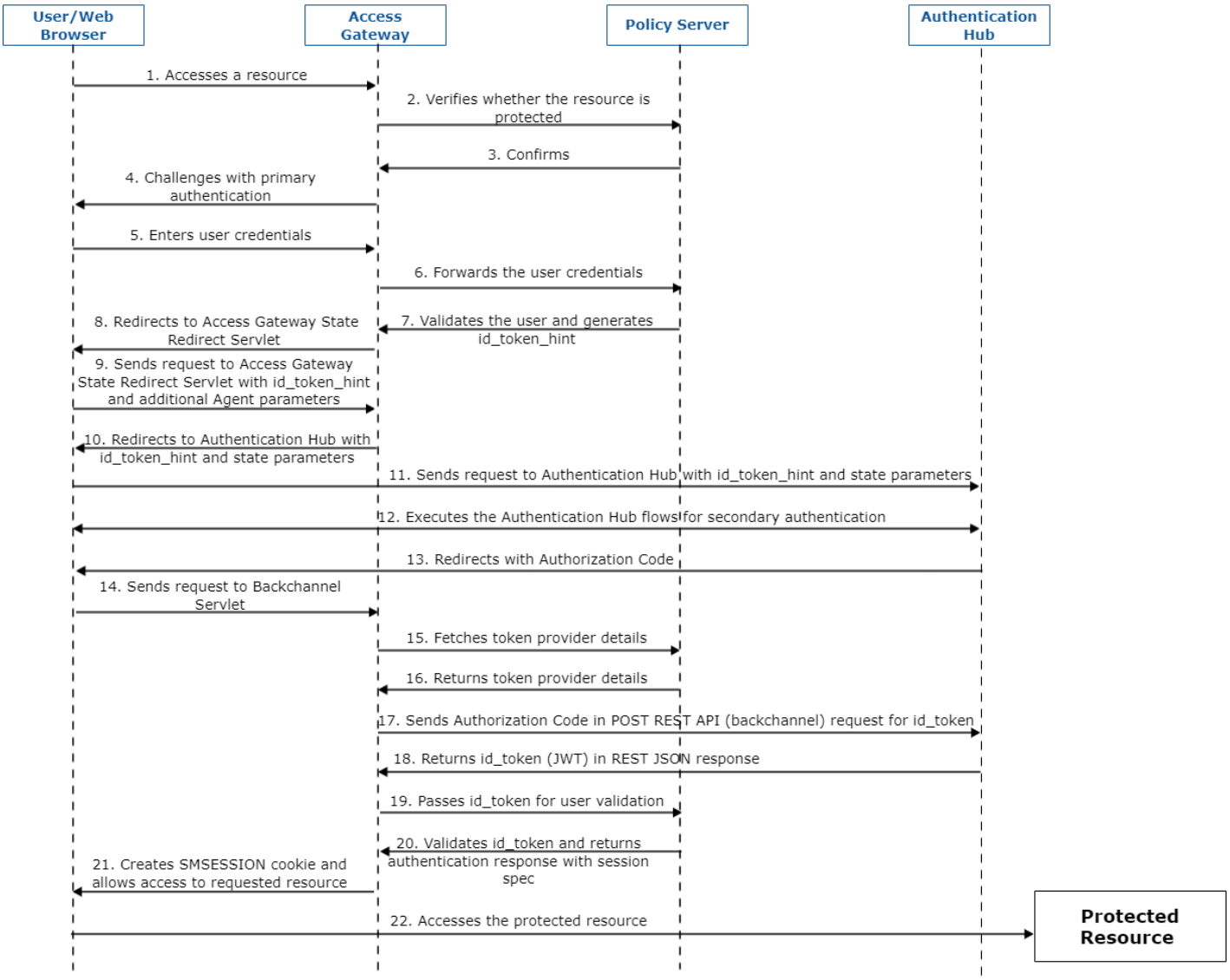
In this scenario:

1. SiteMinder performs the primary user authentication using the configured authentication scheme.
2. IDSP performs the secondary user authentication.

The following process describes how this scenario works at a *high-level*:

1. User accesses a resource that is protected with the authentication scheme that is designed for the integration.
2. SiteMinder challenges the user for credentials and user enters the required inputs.
3. SiteMinder authenticates the user, generates an id_token_hint upon successful user authentication, and redirects user to IDSP.
4. IDSP validates the id_token_hint and redirects the user to its authentication flow based on the configured AuthN policies.
5. IDSP challenges the user for authentication using one or multiple authentication factors as per the policy.
6. IDSP returns authorization code to SiteMinder upon successful user authentication.
7. SiteMinder uses the received authorization code to make a backchannel request to IDSP. If IDSP is unavailable for an immediate connection, Access Gateway retries five times to reconnect for the backchannel communication before the transaction fails.
8. IDSP generates an id_token and redirects the user back to SiteMinder.
9. SiteMinder validates the received id_token, disambiguates the user login from the user directory, generates SMSESSION upon successful authentication, and authorizes the user.
10. SiteMinder returns the requested resource to the user.

The following diagram describes the *detailed flow* between SiteMinder and IDSP:



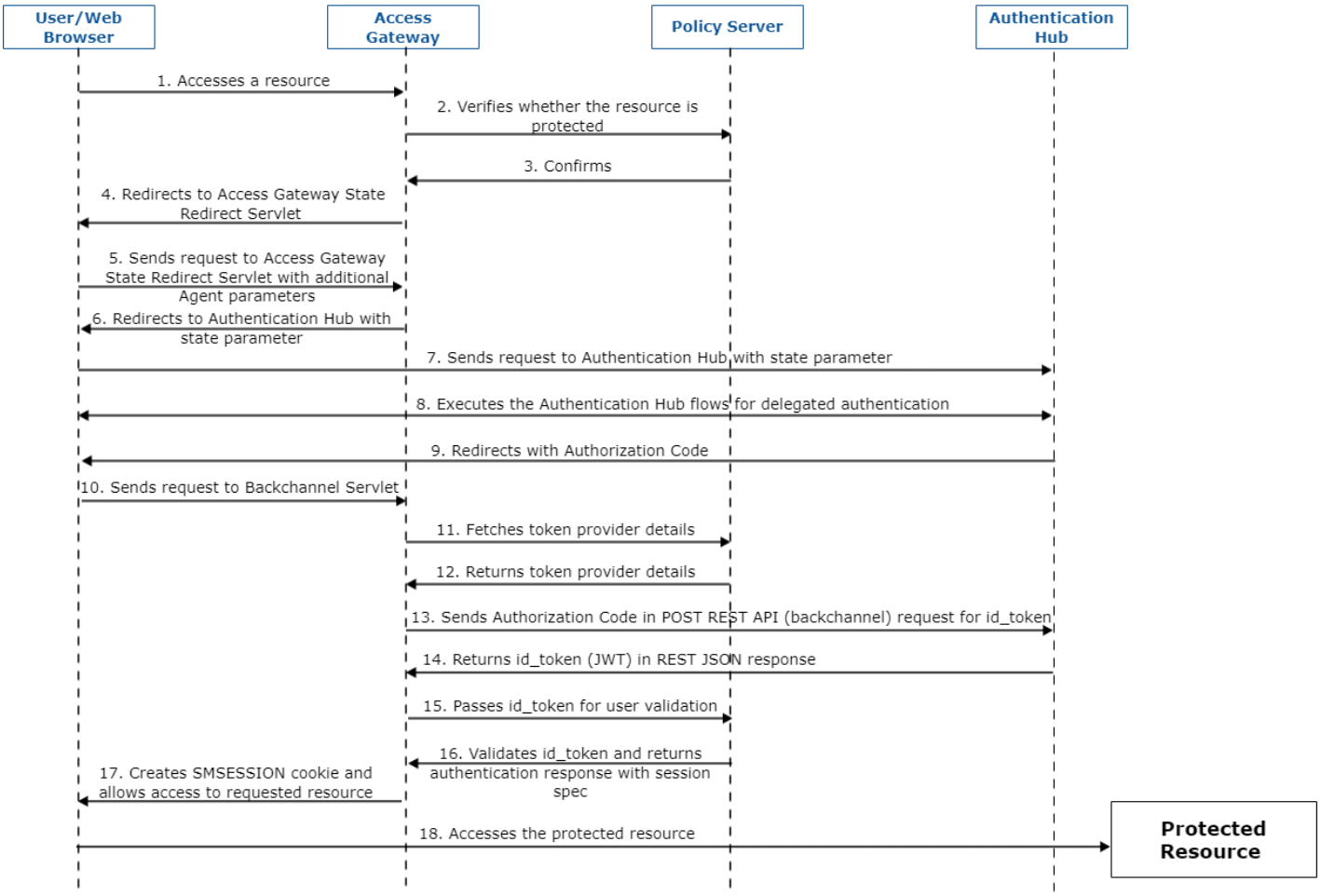
Scenario 2: SiteMinder delegates the user authentication to IDSP

In this scenario, IDSP performs complete user authentication based on AuthN policies. The following process describes how this scenario works at a *high-level*:

1. User accesses a resource that is protected with the authentication scheme that is designed for this integration.
2. SiteMinder redirects user to IDSP.
3. IDSP redirects the user to its authentication flow based on the configured AuthN policies.
4. IDSP challenges the user for authentication using one or multiple authentication factors as per the policy.
5. IDSP returns authorization code to SiteMinder upon successful user authentication.
6. SiteMinder uses the received authorization code to make a backchannel request to IDSP. If IDSP is unavailable for an immediate connection, Access Gateway retries five times to reconnect for the backchannel communication before the transaction fails.
7. IDSP generates an id_token and redirects the user back to SiteMinder.

8. SiteMinder validates the received id_token and retrieves the login claim of the user.
9. SiteMinder disambiguates the user login from the user directory, generates SMSESSION upon successful authentication, and authorizes the user.
10. SiteMinder returns the requested resource to the user.

The following diagram describes the *detailed flow* between SiteMinder and IDSP:



Now that you understand the overview of the integration, proceed to learn the limitations and prerequisites of this integration.

Release Information of SiteMinder Integration with Symantec Identity Security Platform

Describes the list of new features that SiteMinder supports as part of the integration with Symantec Identity Security Platform.

SiteMinder provides new features in the following builds of the IDSP release:

- SiteMinder Solution for Integration with IDSP 3.2
- SiteMinder 12.8.0700.2989 Build for Integration with IDSP
- SiteMinder 12.8.0700.2947 Build for Integration with IDSP

SiteMinder Solution for Integration with IDSP 3.2

- IDSP 3.2 mandates the presence of the **redirect_uri** query parameter in requests to its Token endpoint in an Authorization Code flow. You must configure SiteMinder in *one* of the following options to include the **redirect_uri** query parameter in its requests to the Token endpoint of IDSP:
  - **Option 1: Use the SM-AH_Native_Integration_Patch patch that is provided in IDSP -** Support Portal > My Downloads (Search for IDSP in the portal and select the appropriate version number)

    This patch provides the latest fedfws_obfs.jar file that allows SiteMinder to include the redirect_uri in its requests to the Token Endpoint. To use this method, perform the following steps:
    a. Download **SM-AH_Native_Integration_Patch.zip** from the - Support Portal > My Downloads (Search for IDSP in the portal and select the appropriate version number) onto the Access Gateway machine and unzip the file.
    b. Stop Access Gateway.
    c. Navigate to `<access-gateway-installation-path>\secure-proxy>\Tomcat\webapps\affwebservices\WEB-INF\lib` and take a backup of the existing **fedfws_obfs.jar** file into another folder.
    d. Open the **SM-AH_Native_Integration_Patch** folder, open the ***<version>*** folder that is relevant to you, copy the latest **fedfws_obfs.jar** file, and place it in `<access-gateway-installation-path>\secure-proxy\Tomcat\webapps\affwebservices\WEB-INF\lib` to replace the existing fedfws_obfs.jar file.
    e. Start Access Gateway.
  - **Option 2: Manually add metadata in new applications to be used with SiteMinder**

    In this method, for every new application that you create in IDSP 3.2 to be used for the integration with SiteMinder, add the following metadata in the application:

```
"metadata": [
{
"name": "urn:iam:app:allowAppNotToSendRedirectUri",
"value": "true"
},
]
```

- IDSP 3.2 mandates the use of HTTPS in the URLs of the new applications. If you want to use HTTP for the applications in your environment, add the following metadata in the application that is associated with SiteMinder:

```
"metadata": [
{
"name": "urn:iam:app:allowNonSslRedirectUri",
"value": "true"
},
]
```

SiteMinder 12.8.0700.2989 Build for Integration with IDSP

- **Support for IDSP as a fallback authentication scheme in Authentication Chain Template**: SiteMinder now allows IDSP as a fallback authentication scheme for Kerberos and Windows authentication schemes in Authentication Chain Template. For more information, see Support for IDSP as a Fallback Authentication Scheme in Authentication Chain Template.
- **Enhancement to the ID Token Hint flow to generate an SMSESSION**: In the previous releases, if the authentication level in an ID Token Hint is lower than the **Minimum Authentication Level** value that is defined in the associated authorization provider, a user authentication fails. From this release, a new field named **Support Step-up Authentication** is introduced to manage the behavior in this scenario. If this field is selected, a step-up authentication is performed. If this field is not selected, user authentication fails. For information about the field and the ID Token Hint flow, see Use ID Token Hint to Generate an SMSESSION.
- **Send agent name and domain name in ID Token Hint**: SiteMinder provides an option to include the agent name and domain name in an ID Token Hint that is sent to IDSP. To support this feature, the IDSP authentication scheme now includes the **ID Token Hint Generation Setup** section for the required configuration. If the **Send Additional Information in ID Token Hint** field in this section is selected, the **urn:iam:authn:clientContext** section of an

ID Token Hint includes the agent name in the **sm_agent** key and domain name in the **sm_domain** key. If the **urn:iam:app:clientContextClaimsToBeLogged** setting is configured in the application of IDSP, the agent name and domain name of SiteMinder are logged in the IDSP logs. For information about the new fields, see Configure the Integration in 12.8.0700.2989 Build. For information about ID Token Hint, see Identity Propagation in SiteMinder. For information about **urn:iam:app:clientContextClaimsToBeLogged**, see the IDSP documentation.

In the previous releases, SiteMinder can send an ID Token Hint to IDSP in the following scenarios:

- When SiteMinder performs the primary authentication
- When the authentication is delegated to IDSP and a prior SiteMinder user session exists

If this feature is enabled, SiteMinder can send an ID Token Hint to IDSP when the authentication is delegated to IDSP but a prior SiteMinder user session does not exist too.

- **Support for the principalConditionRequiredInPolicy setting of IDSP**: SiteMinder supports the principalConditionRequiredInPolicy tenant/global setting that controls whether an empty principal condition is allowed in an authentication policy in IDSP. By default, the setting is set to **true** in IDSP. For information about the setting, see Managing Settings.
- **New claim in ID Token Hint for indicating the expiry time of a signin flo**w: SiteMinder now sends the **urn:iam:login_flow:expires_in_seconds** claim in an ID Token Hint to IDSP. The value of **urn:iam:login_flow:expires_in_seconds** is set to the chain cookie expiry value. If the value of the chain cookie expiry is not modified, its default value of 600 seconds is sent in **urn:iam:login_flow:expires_in_seconds**. If the chain cookie expiry time is customized, the custom value is sent.
  IDSP uses the value of **urn:iam:login_flow:expires_in_seconds** as the expiry time of the signin flow. The value of **urn:iam:login_flow:expires_in_seconds** overrides the value of the **signinFlowExpiryMins** tenant setting.
- **Updated third-party libraries**: This build supports the following upgraded third-party library:
  - Apache Commons FileUpload 1.5
  - Apache HTTP Server 2.4.58
  - Apache Tomcat 9.0.82
  - Apache Xalan 2.7.3
  - CURL 8.4.0
  - MIT Kerberos 1.21.2
  - OpenSSL 2zi-fips

SiteMinder 12.8.0700.2947 Build for Integration with IDSP

- **Use a sitename to retrieve KDCs in Kerberos authentication scheme**: SiteMinder allows you to define a site using which MIT Kerberos libraries can retrieve all the KDCs that are configured in a site, instead of retrieving all the KDCs in a domain. For more information, see Define a Sitename to Retrieve KDCs in Kerberos Authentication Scheme.
- **Support for SiteMinder password policies in IDSP**: When SiteMinder delegates the authentication of a user to IDSP, IDSP can now leverage the password policies that are configured in SiteMinder to authenticate the user if PWD is one of the obligations. For more information, see Support for SiteMinder Password Policies in IDSP.
- **Support for the front-channel SLO**: SiteMinder OpenID Connect Provider now supports the front-channel SLO, a logout mechanism that uses front-channel communication between OP and RPs to be logged out, through a User Agent. If enabled, this feature instructs SiteMinder to revoke Authorization Code, Access Token, Refresh Token, and ID Token that are generated by OIDC Clients within a user session. Additionally, SiteMinder can be configured to log out an existing session too. For more information, see Support for Front-Channel SLO in SiteMinder OpenID Connect Provider.
- **Support for Refresh Token rotation**: SiteMinder as OpenID Connect Provider now supports the Refresh Token rotation feature to enhance the security of an OIDC implementation. When enabled, this feature instructs SiteMinder to invalidate the current Refresh Token and to generate a new Refresh Token with every Access Token refresh response. To support this feature, the following fields are introduced in the Clients Dialog when **Refresh Token** is selected as a **Grant Type**:
  - **Enable Refresh Token Rotation**

Specifies whether a Refresh Token must be rotated.
Default value: Enabled

- **Maximum Refresh Token Rotations**
  Defines the maximum number of times a Refresh Token can be rotated. This field is displayed only if **Enable Refresh Token Rotation** is selected.
  Default value: 600

For information about configuring an OIDC client, see Clients Dialog.

- **New Parameters in ID Token**: SiteMinder now includes the **sid** (session ID) and **jti** (token identifier in a session store) parameters in an ID Token.
- **Validate ID Token in the Introspection endpoint**: SiteMinder OpenID Connect Provider now allows the validation of an ID Token in its Introspection endpoint, along with an Access Token or Refresh Token.
- **Support for the change password functionality in authentication and authorization web services**: The web services now support the change password functionality. This feature requires a user to change an expired password upon a successful login. For more information, see Support for Change Password in Authentication and Authorization Web Services.
- **Use an existing virtual host for authentication and authorization web services**: In the previous releases, the authorization and authentication web services require only a dedicated virtual host to function along with an ACO for the web services. From this release, an existing virtual host, including the default virtual host, can be used to enable the web services instead of a dedicated virtual host. For more information, see Use an Existing Virtual Host for Authentication and Authorization Web Services.
- **Maintain a common session ID in a user session**: SiteMinder is enhanced to use a common session ID for a user in a user session, even after a step up authentication.
- **Support for AMR integrity and continuity in SAML 2.0 federation partnerships:** The AMR integrity and continuity feature is extended to IdP-initiated and SP-initiated SAML 2.0 transactions. For information about how the feature works, see AMR Integrity and Continuity.
- **Enhanced SecurID HTML Form Template and SecurID Template authentication schemes**: The SecurID HTML Form Template and SecurID Template authentication scheme dialogs now allow you to define an AMR type. To support this feature, the dialogs of these authentication schemes now contain the **Authentication Method Reference (AMR)** section where multiple AMR types can be assigned. Also, the Multi Factor Authentication authentication scheme now allows you to select SecurID HTML Form Template for primary authentication.
  - If an AMR type is defined in the SecurID HTML Form Template authentication scheme and this authentication scheme is configured for the primary authentication in a Multi Factor Authentication authentication scheme, the AMR integrity and continuity feature continues to work using the defined AMR types.
  - If an AMR type is defined in the SecurID HTML Form Template and/or SecurID Template authentication scheme that is used for authenticating a user, the AMR integrity and continuity feature continues to work in a step-up scenario.
  - If an AMR type is not defined in the SecurID HTML Form Template and/or SecurID Template authentication scheme, SWK is considered as the default AMR type of these authentication schemes and the same is used in the above scenarios.

  For information about how AMR integrity and continuity works, see AMR Integrity and Continuity.
- **Support for SecurID HTML Form authentication scheme in Multi Factor Authentication authentication scheme behavior**: The Multi Factor Authentication authentication scheme now supports SecurID HTML Form as the primary authentication scheme. For information about the authentication scheme, see Configure the Integration.
  If you want to use SecurID HTML Form as the primary authentication scheme to protect a sensitive resource, see Additional Configuration to Use SecurID HTML Form for Risk-based Authentication During Re-authentication of a Sensitive Resource.
- **Changes in the Multi Factor Authentication authentication scheme behavior**: The following changes have been made in how a Multi Factor Authentication authentication scheme works:

  - SiteMinder now uses a common session ID in both primary and secondary authentication schemes. This ensures that a common session ID is used in an end-to-end user flow. This session ID is logged in the **sm_sessionid** field in audit logs.
  - An AuthChallenge event in audit logs now logs the transaction ID that is generated by SiteMinder, during the primary authentication or request initiation, in the **sm_transactionid** field. The idletime, maxtime, and authlevel values are now logged in the **sm_status** field instead of the **sm_transactionid** field.
    Example:
    ```
    AuthChallenge pshostname [26/Dec/2022:14:43:31 +0530] "10.0.0.0 cn=user12,o=security.com" "spsagent
     GET /x509/x509.html" [121360a3-436fa55c-f6929510-5ad0a768-49f30c3f-cf9] [56]
    ```
  - An AuthAccept event in audit logs now logs the transaction ID that is generated by IDSP during the secondary authentication or the full delegation, along with the SiteMinder transaction ID that is generated by SiteMinder during the primary authentication or request initiation, in the **sm_transactionid** field. These two transaction IDs are separated by an underscore (**_**).
    Example:
    ```
    AuthAccept pshostname [26/Dec/2022:14:43:59 +0530] "10.0.0.0 cn=user12,o=security.com" "spsagent GET /
    x509/x509.html" [acd545ee-ddc6-40dd-9b4c-f3828277aba5_121360a3-436fa55c-f6929510-5ad0a768-49f30c3f-cf9]
     [0] idletime=3600;maxtime=7200;authlevel=5
    ```
  - SiteMinder calls to the Authorization and Token endpoints of IDSP now use a common SiteMinder transaction ID.
- **Support proxy for the backchannel communication between Access Gateway and IDSP**: In the previous releases, SiteMinder supports using a proxy for the communication between Administrative UI and IDSP. From this release, SiteMinder supports using a proxy for the communication between Access Gateway and IDSP too. To support this feature, the **Enable Proxy** field has been renamed to **Use Proxy for Administrative UI Communication** and the new field **Use Proxy for Access Gateway Communication** has been introduced. For information about these fields, see Configure the Integration in 12.8.0700.2947 Build.
- **Support for a different Issuer URL for tenant management operations**: If IDSP exposes tenant management and end user operations on different ingress controllers, SiteMinder now allows you to define the ingress controller that is used for tenant management operations in a new field named **Use a Different Ingress Controller for Tenant Management** in the Authentication Policy Management dialog. If enabled, another new field named **Tenant Management Issuer URL** is displayed to gather the URL that is used for tenant management operations. For more information, see Configure the Integration in 12.8.0700.2947 Build.
- **New rule conditions in IDSP authentication policies**: SiteMinder Administrative UI now supports the following new conditions that can be configured while creating a rule in a IDSP authentication policy:
  - IDP Names
  - IP Addresses or Range
  - IP Address Condition

  For more information, see Manage Authentication Policies in SiteMinder.
- **Persist SiteMinder information and data of a user (data) in the session storage of a web browser**: In the previous releases, during the implementation of the Multi Factor Authentication authentication scheme, the data from a user is included in all the requests and responses between SiteMinder and IDSP using the state field in IDSP. The relay of a lengthy data is not supported due to the restriction on the length of the state field. From this release, the data from a user is persisted in the session storage of the user's browser, before sending the user to IDSP for secondary authentication. Once IDSP successfully completes the secondary authentication, the data is sent back to the user from the session storage of the user's browser. This feature avoids the dependency on the length of the **state** field and depends only on the session storage limit of the user browser.
- **Updated third-party libraries**: This build supports the following upgraded third-party libraries:

- Apache Commons Text 1.10.0
- Apache HTTP Server 2.4.55
- Apache Tomcat 9.0.71
- Apache Xerces-C 3.2.4
- Google Protocol Buffers 3.21.12
- Jackson Databind 2.12.7.1
- jquery 3.6.3
- libcurl 7.87.0
- MIT Kerberos 1.20.1
- Network Security Services (NSS) 3.87
- OpenSSL 1.0.2zg

Integration Limitations and Known Issues

Limitations

Review the following limitations of the integration:

- Note that Symantec Identity Security Platform supports only LDAP user directory type. Symantec Directory 14.1 and Microsoft Active Directory 2016 are supported. For information about supported directories in IDSP, see Prerequisites.
- If **Persist Session Variables** option is selected in IDSP Template or any authentication scheme that is used in the Multi Factor Authentication Template, SiteMinder stores claims only from the secondary authentication scheme.
- A user name must not contain a single quote (') or hash (#).
- The SameSite option **Strict** is not supported.
  During the integration flow, the hostname of IDSP is auto-populated in the Referer field value as IDSP always initiates the first redirect to Access Gateway backchannel. The request is thus considered a cross-site request and web browser does not send the generated cookie in the response.
- Multi Factor Authentication Chain template does not support the **FCCCompatMode** ACO parameter if a resource is protected by Web Agent.
- While configuring the Multi Factor Authentication Chain and IDSP authentication schemes, ensure that the values of the following parameters do not exceed 1024 characters in length when combined:
  - Access Gateway State Redirect URL
  - ACRs
- (2022.Jun.01 Build) If any custom data is specified in an HTML Form or on any of its combination authentication scheme that acts as a primary authentication in the Multi Factor Authentication authentication scheme, during a step-up authentication, the custom data is not sent to IDSP as SiteMinder skips the primary authentication challenge with the introduction of the AMR integrity feature.

Known Issues

The following known issues exist in this integration:

- (2.1.0 1036 Build of IDSP) the IDSP Providers dialog in Administrative UI does not allow managing rules in an authentication policy. To allow managing rules through SiteMinder, set the **principalConditionRequiredInPolicy** tenant/global setting to **false** in IDSP. For information about the setting, see Managing Settings.
- (2022.Jun.01 Build) If an already authenticated user accesses a sensitive resource, configured with the same authentication level as the existing session but with a different authentication scheme than the existing session, the value of AMR in sessionspec contains only the AMR of the existing session.
- A scoped administrator with modify permission can edit the IDSP objects though the associated workspace is assigned only read-only permission to the IDSP objects.

As a workaround, do not associate the scoped administrator with a workspace.

- A scoped administrator with a view permission can see the options to manage rules of the IDSP policies. However, the scoped administrator cannot make any changes to the rules as the Save button is not enabled though the options to manage the rules are displayed.
- If the current IDSP service is down and the SiteMinder Administrator tries to modify the objects that use it to replace the instance with another instance, Administrative UI takes longer time than expected to display the object dialogs in the edit mode.

Proceed to complete the prerequisites that are required for implementing the integration.

Prerequisites to Implement the Integration

Complete the following steps before you proceed to configure and implement the integration:

1. Deploy and configure SiteMinder.
   - Deploy SiteMinder 12.8.06a, or the in progress version that is delivered along with the 2022.Jan.01 build or the later build of the Symantec Identity Security Platform release. For more information about the features that are supported in each build, see Summary of Features Supported in Each Build.
   - Configure a session store if you want to persist the user claims information that is received in id_token when X509 certificate or delegated authentication is performed. For information about persisting session variables, see Authentication Schemes.
   - Also, if the **Persist Session Variables** option is selected in the authentication scheme that is used by the integration, ensure that the option is selected in the realm with which the authentication scheme is mapped.
   - Ensure that Access Gateway and Administrative UI can communicate with IDSP over HTTPS.
2. Deploy and configure IDSP.
   - Note the host value in the **prefix.domain** format.
   - Understand how REST APIs work in IDSP. For information, see Using the REST APIs.
   - Create Application and Policy with ACRs for SiteMinder in IDSP.

     Create an application for SiteMinder using the certificate alias of the imported public certificate of SiteMinder. If CONFIDENTIAL client is configured, IDSP can perform only delegated authentication, where SiteMinder delegates complete user authentication to IDSP. If TRUSTED client is configured in IDSP, then IDSP can perform delegated or secondary authentication.

     Create a policy with required ACRs for SiteMinder. For information, see see Managing Policies.

     Example: Payload to create an application for SiteMinder when IDSP performs secondary user authentication

```
{
"name": "<appName>",
"status": "active",
"description": "Default Application for Security",
"clientType": "TRUSTED",
"assertionVerificationCertAlias": "<certAlias_of_imported_SM_certificate>",
"allowedOpenIDScopes":[
"openid",
"profile",
"email"
],
"redirectURIs": [
"<ssoGW>/affwebservices/public/bctokencontroller"
],
"allowedGrantTypes": [
"implicit",
"client_credentials",
"urn:ietf:params:oauth:grant-type:jwt-bearer",
```

```
"authorization_code"
],
"allowedOperations": [
"introspect"
],
"skipIssuerAudienceForIT": true,
"userInfoEndpointResponseFormat": "PLAIN_JSON"
}
```

- ACRs must be configured in AuthN policies of IDSP.
  AuthN policies of IDSP support multiple ACRs to define different authentication flows. ACRs are mandatory and are defined as a part of authentication rules that a user has to satisfy to gain access to a protected resource. When you configure ACRs in AuthN policies, note the ACR values for using them in the authentication scheme configuration.
  For basic information on ACRs, see Support for Authentication Context Class References for Stronger Authentication section in the integration overview.
  For information on configuring ACRs in IDSP, see ACRs.
  For information on using the ACRs in SiteMinder, see the integration configuration content that follows.
- Retrieve the Client ID and Client Secret of the default tenant from master node of IDSP and note the values. This information is required during the integration configuration.

```
kubectl get secret ${RELEASENAME}-ssp-secret-defaulttenantclient -n ${NAMESPACE} -o
 jsonpath="{.data.clientId}" | base64 --decode; echo
kubectl get secret ${RELEASENAME}-ssp-secret-defaulttenantclient -n ${NAMESPACE} -o
 jsonpath="{.data.clientSecret}" | base64 --decode; echo
```

3. Set up a trust relationship between SiteMinder and IDSP.
   – Configure SiteMinder as OIDC Relying Party in IDSP.
   – Determine how to manage the key rollover of certificates as per your organization's business security requirements.
4. Ensure that Access Gateway can communicate with the OIDC Endpoints of IDSP.
5. Ensure that the user schema of all the users is identical in the user stores of SiteMinder and IDSP.

For information about configuring, see IDSP documentation.

Complete these prerequisites and then proceed to learn how to configure and implement the integration.

Summary of Features Supported in Each Build

To support the integration, you can use SiteMinder Release 12.8.06a or the in progress version that is delivered along with the latest build of the Symantec Identity Security Platform release.

The following table helps you choose a build based on the features that you want to implement:

Feature	12.8.0700.2989 Build	12.8.0700.2947 Build	Release 12.8.06a
Use IDSP Template as a fallback authentication scheme in Authentication Chain Template	Yes	-	-

Perform a step-up authentication if the authentication level in an ID Token Hint is lower than the Minimum Authentication Level value that is defined in the associated authorization provider, in the ID Token Hint flow to generate an SMSESSION	Yes	-	-
Send agent name and domain name information in ID Token Hint	Yes	-	-
Support for the principalConditionRequiredInPolicy setting of IDSP	Yes	-	-
Define a sitename to retrieve KDCs in Kerberos authentication scheme	Yes	Yes	-
Support for front-channel SLO in SiteMinder OpenID Connect Provider	Yes	Yes	-
Support for change password in authentication and authorization web services	Yes	Yes	-
Use an existing virtual host for authentication and authorization web services	Yes	Yes	-
Support for SiteMinder password policies in IDSP	Yes	Yes	-
AMR integrity and continuity	Yes	Yes	-
Send custom claims in ID Token Hint	Yes	Yes	-
Send idp_type, idp_user_guid, and idp_id claims in ID Token Hint	Yes	Yes	-
Use a relative URL for Access Gateway Redirect URL and State Redirect URL	Yes	Yes	-
Support for security zones in Multi Factor Authentication authentication scheme	Yes	Yes	-
Add metadata to Trusted OIDC Provider and ID Token Policy objects	Yes	Yes	-
Use ID Token Hint to generate an SMSESSION	Yes	Yes	-
Configure AMR Mapping	Yes	Yes	-
Validate the revocation status of a JWT Bearer Token	Yes	Yes	-

Support for SiteMinder locale in IDSP	Yes	Yes	-
Configure re-authentication for sensitive resources that are protected by the Multi Factor Authentication or IDSP authentication schemes	Yes	Yes	-
Send user DN in an ID Token Hint	Yes	Yes	-
Support for OnAuthAcceptCredentials in JWT authentication scheme	Yes	Yes	-
Certificate linking to the Multi Factor Authentication or IDSP authentication schemes	Yes	Yes	-
Manage authentication policies of IDSP using SiteMinder Administrative UI	Yes	Yes	Yes
Configure multiple IDSP providers in SiteMinder UI	Yes	Yes	Yes
Support for password policies in Multi Factor Authentication Chain	Yes	Yes	Yes
Support for IDSP Risk Service	Yes	Yes	Yes
Support for Identity Propagation in a fresh/step-up authentication flow	Yes	Yes	Yes
Support for the jwt-bearer Token grant	Yes	Yes	Yes
Support for dynamically picking the tokenIssuer value of an application during the Multi Factor Authentication Chain authentication scheme configuration	Yes	Yes	Yes
Support for new attributes in ID Token Hint	Yes	Yes	Yes
Support for encrypted JWT (nested token)	Yes	Yes	Yes
Support for sending custom data from Client to IDSP	Yes	Yes	Yes
Support for claims validation in the JWT authentication scheme	Yes	Yes	Yes
Support for a secondary verification certification alias in the jwt-bearer Token grant flow	Yes	Yes	Yes
Support for the ServerErrorFile ACO parameter	Yes	Yes	Yes

Review the integration limitations and complete the prerequisites before you proceed to configure the integration.

Configure the Integration of SiteMinder with Symantec Identity Security Platform

For information about the configuration, use one of the following topics according to the SiteMinder version:

- Configure the Integration in 12.8.0700.2989 Build
- Configure the Integration in 12.8.0700.2947 Build

Configure the Integration in 12.8.0700.2989 Build

Perform the configuration only after you have understood the integration overview and completed all the steps mentioned in Prerequisites. The integration configuration process is as follows at a high level when this build is used:

1. Download the installers of the current release.
2. Configure Symantec Identity Security Platform provider.
3. Import certificates into SiteMinder.
4. Configure an authentication scheme in SiteMinder for the integration.
   - Use the **Authentication Hub Template** authentication scheme type for SiteMinder to delegate the user authentication to IDSP.
   - Use the **Multi Factor Authentication Chain Template** authentication scheme type for SiteMinder to perform the primary authentication and IDSP to perform the secondary authentication.
5. Map the authentication scheme to a realm.

Download Installers

You can use the latest installers to deploy a new environment or to upgrade from a previous release environment.

1. Download the integration zip file and unzip the file to navigate to the installers depending on the operating system.
2. Deploy the installers to set up the in-progress release environment. If you are upgrading to the latest release, follow the Upgrading instructions that are described in the SiteMinder 12.8 documentation as the upgrading process is identical in this integration environment too.

Configure IDSP Provider

To establish a communication between SiteMinder and IDSP, configure an IDSP provider in SiteMinder. This is a one-time setup.

Note: If you are a scoped administrator, ensure that you have access to the **IDSP** workspace content with permission to work on the **AuthHub Objects Administration** security category. For information about a scoped administrator, see Create a Scoped Administrator.

Follow these steps:

1. Navigate to **Infrastructure**, **Authentication**, **IDSP Providers** in Administrative UI.
2. Enter a name and description to identify this connection in **Provider Name** and **Description**.
3. Enter the Issuer URL of IDSP tenant. Enter the URL in the following format using the IDSP hostname that was noted as a part of prerequisites:

   ```
   https://ah_hostname/name_of_default_tenant
   ```

   This URL is used to gather the Authorization and Token endpoints of IDSP, and auto-populate them in the integration-specific authentication scheme template.
4. Enter the client ID and client secret of the application that is created in IDSP for the **default** tenant in **Administrative Client ID** and **Administrative Client Secret**. Use the tenant details that were noted as a part of prerequisites.
5. (Optional) To enable SiteMinder to communicate with IDSP using a proxy, select **Enable Proxy** option and perform the following additional steps:
   a. Specify the hostname and port number of the proxy server in **Proxy Host** and **Proxy Port**.
   b. Specify the user credentials to access the proxy server in **Proxy User** and **Proxy Password**.

Note: If you do not opt for proxy communication, ensure that SiteMinder can communicate with IDSP.

6. (Optional) Select **Disable SSL Certificate Validation** to disable the validation of SSL certificate while establishing communication between SiteMinder and IDSP.
7. Click **Save** to save the connection information.
8. Click **Test Connection** to test whether a connection has been established between SiteMinder and IDSP.
   If the **Disable SSL Certificate Validation** option is not selected, ensure that you import the certificate or the root CA of the certificate that is used for enabling SSL in Ingress of IDSP before you test the connection. For more information, see the Import Certificates section.
   If the connection is successful, the Authorization and Token Endpoints information of IDSP is displayed in the **Endpoints** section of the dialog.
9. (Optional) To modify the entered details, click **Modify** and make the necessary changes.
10. (Optional) To view the authentication policies that are created in IDSP for this provider and to create rules in a policy, see Manage Authentication Policies.

Import Certificates

The integration requires the following certificates:

- **The signing key-pair that is used for signing id_token_hint, which is sent to IDSP when SiteMinder performs the primary authentication**.
  This signing key-pair must be present in SiteMinder. You can use an existing signing key-pair. If you want to use a new signing key-pair, import it using the following steps:
  a. Navigate to **Infrastructure**, **X509 Certificate Management**, **Trusted Certificates and Private Keys** in SiteMinder Administrative UI.
  b. Import the new signing key-pair. For information about importing certificates, see Import Trusted Certificates and Key Certificate Pairs.

  Also, the public certificate within the signing key-pair must be present in IDSP. If the public certificate is readily available, proceed to **Step 2**. If it has to be retrieved from SiteMinder, perform **Step 1 and Step 2**.
  a. Navigate to **Infrastructure**, **X509 Certificate Management**, **Trusted Certificates and Private Keys** in SiteMinder Administrative UI, select the public certificate from the list of available certificates, and click **Export**.
  b. Import this certificate into IDSP using the `https://ah_hostname/tenant_name/admin/v1/Certs/` endpoint.
  For example, the following sample describes how to use **cURL** for importing the certificate into IDSP:
  Important! Ensure that **jq tool** is installed before you perform the following steps. For example, jq tool 1.6 was installed using the following commands in our test environment:

```
wget -O jq https://github.com/stedolan/jq/releases/download/jq-1.6/jq-linux64
chmod +x ./jq
cp jq /usr/bin
```

  a: Acquire Access Token from IDSP

```
export HOSTNAME=ah_hostname
export TENANT=ah_tenant
export CLIENTID=ah_tenant_clientid
export CLIENTSECRET=ah_tenant_clientsecret
export TOKENISSUER=sm_token_issuer

export TOKEN=$(curl -ks -X  POST https://${HOSTNAME}/${TENANT}/oauth2/v1/token -u "${CLIENTID}:
${CLIENTSECRET}" -d 'grant_type=client_credentials&scope=urn:iam:myscopes' | jq '.access_token' | sed
 's/"//g')
```

  b: Import the public certificate within the signing key-pair into IDSP

```
export CERTDATA=$(cat<absolutepath_to_publickeycert>| tr -d '\r' | sed '1d;$d' | tr '\n' ' ' | tr -d '
 ')
```

```
curl -k --location --request POST "https://${HOSTNAME}/${TENANT}/admin/v1/Certs/" \
--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
--header "Authorization: Bearer $TOKEN" \
--data "{
 \"tokenIssuer\": "\"$TOKENISSUER\"",
 \"alias\": \"SSOCert\",
 \"b64CertData\": "\"$CERTDATA\""
}"
```

Note: This certificate is required for regular secure operations in production after you have tested the user authentication flow between SiteMinder and IDSP with the **Disable Identity Token Signature Validation** option selected during testing.

- **The Ingress root CA/Ingress server certificate of IDSP**.

  This refers to the certificate or the root CA of the certificate that is used for enabling SSL in Ingress of IDSP. If the certificate or its root CA is readily available, proceed to **Step 2**. Else, perform **Step 1 and Step 2**.

  a. Run the following command on a machine where OpenSSL is installed:

  ```
  openssl s_client -showcerts -servername server -connect $HOSTNAME:443 > cacert.pem
  ```

  In the above command, **HOSTNAME** refers to the hostname of IDSP.

  b. Import the above mentioned certificate or its root CA into SiteMinder. Navigate to **Infrastructure**, **X509 Certificate Management**, **Certificate Authorities** in SiteMinder Administrative UI, and import the certificate or its root CA. For information about importing Certificate Authority, see Certificate Authorities.

  Note: This certificate is required for regular secure operations in production after you have tested connectivity between SiteMinder and IDSP with the **Disable SSL Certificate Validation** option selected during testing.

- **The public certificate of the signing key that is used by IDSP tenant to sign the ID Token**.

  a. Retrieve the certificate from IDSP using the `https://ah_hostname/tenant_name/admin/v1/SigningCert` API.

  For example, the following sample describes how to use **cURL** for retrieving the certificate from IDSP:

  Important! Ensure that **jq tool** is installed before you perform the following steps. For example, jq tool 1.6 was installed using the following commands in our test environment:

  ```
  wget -O jq https://github.com/stedolan/jq/releases/download/jq-1.6/jq-linux64
  chmod +x ./jq
  cp jq /usr/bin
  ```

  a: Acquire Access Token from IDSP

  ```
  export HOSTNAME=ah_hostname_value
  export TENANT=ah_tenant
  export CLIENTID=ah_tenant_clientid
  export CLIENTSECRET=ah_tenant_clientsecret
  export TOKENISSUER=sm_token_issuer
  export TOKEN=$(curl -ks -X  POST https://${HOSTNAME}/${TENANT}/oauth2/v1/token -u "${CLIENTID}:
  ${CLIENTSECRET}" -d 'grant_type=client_credentials&scope=urn:iam:myscopes' | jq '.access_token' | sed
   's/"//g')
  ```

  b: Retrieve the certificate from IDSP

  ```
  curl -ks --location --request GET "https://${HOSTNAME}/${TENANT}/admin/v1/SigningCert" --header
   "Authorization: Bearer ${TOKEN}" |  jq '.b64CertData' | sed 's/"//g' | tr -d '\n' > chk.crt
  ```

  b. Navigate to **Infrastructure**, **X509 Certificate Management**, **Trusted Certificates and Private Keys** in SiteMinder Administrative UI, and import the public certificate of IDSP that was retrieved in the above step. For information about importing certificates, see Import Trusted Certificates and Key Certificate Pairs.

Configure Authentication Scheme

The authentication scheme type depends on the scenario that you want to implement.

- Use the **Authentication Hub Template** authentication scheme type for SiteMinder to delegate the user authentication to IDSP.
- Use the **Multi Factor Authentication Chain Template** authentication scheme type for SiteMinder to perform the primary authentication and IDSP to perform the secondary authentication.
  SiteMinder can perform primary authentication using one of the following authentication schemes:
  - HTML Form
  - X509 Client Certificate
  - X509 Client Certificate or HTML Form
  - X509 Client Certificate and HTML Form
  - Windows
  - Windows Fallback to HTML Form
  - Kerberos
  - Kerberos Fallback to HTML Form
  - RADIUS

Both the templates support PKCE-enabled Authorization Code Flow.

Follow these steps:

1. Navigate to **Infrastructure**, **Authentication**, **Authentication Schemes** in Administrative UI.
2. Click **Create Authentication Scheme**, select **Create a new object of type Authentication Scheme**, and click **OK**.
3. Enter a **Name** and **Description** of the authentication scheme.
4. Select **Protection Level** to provide an additional measure of flexibility in access control. Define a level from 1 through 1000.
5. Perform *one* of the following steps depending on the integration scenario that you want to implement:
   SiteMinder to perform the primary authentication and IDSP to perform the secondary authentication
   a. Select **Multi Factor Authentication Chain Template** from the **Authentication Scheme Type** drop-down list.
   b. Specify the authentication scheme details with which SiteMinder must perform primary authentication in **Primary Authentication Scheme**. Select an authentication scheme type from the supported authentication scheme types that can be used for primary authentication, and then select an authentication scheme that is already created for the selected authentication scheme type.
   c. **(Only for Windows or Kerberos authentication scheme)** Select a fallback scheme that must be used if primary authentication fails from **Primary Authentication Fallback Scheme** drop-down list.
   d. Specify the authentication scheme details with which IDSP must perform secondary authentication in **Secondary Authentication Scheme**. Select **IDSP** as the authentication scheme type and then select an authentication scheme that is already created for this authentication scheme type.
      The authentication chain expression that is achieved using this authentication scheme type is auto-populated in **Expression**. **Token Issuer** displays the tokenIssuer value of the configured IDSP application, which is associated with the selected secondary authentication scheme.
   e. Specify the signing key alias of SiteMinder in **Signing Certificate Alias**. This value is used for signing id_token_hint that is sent to IDSP when SiteMinder performs the primary user authentication.
   f. Select the algorithm that SiteMinder must use for signing the ID Token Hint from the **Signing Algorithm** drop-down list.
   g. (Optional) Configure the following fields to encrypt ID Token Hint:
      a. Select the alias that was defined while importing the IDSP signing certificate into SiteMinder to encrypt ID Token Hint from the **Encryption Certificate Alias** drop-down list.
      b. Select the JWE algorithm that must be used for encrypting ID Token Hint from the **Encryption Algorithm** drop-down list.

c. Select the JWE method that must be used for encrypting ID Token Hint from the **Encryption Method** drop-down list.

h. Define the SiteMinder user store attribute whose value must be used to look up a user in IDSP in **Basic User Attribute Lookup**. This value is sent in the **sub** and **user_loginid** in id_token_hint.
The possible values are:
- SM_UNIVID, a SiteMinder reserved word that represents the UniversalID attribute of a user, as configured in the user directory
- The name of a virtual attribute, as defined in the **Attribute Mapping** section of the user directory
- The name of a physical user attribute that contains the required value

i. Select **Enable Propagation of Extended User Attributes in ID Token Hint** to enable SiteMinder to propagate an authentication context and a set of identity attributes to IDSP. For information about identity propagation, see Identity Propagation in SiteMinder.

j. (Optional) To send any custom claims in ID Token Hint, specify the names of the required custom claims in **Custom Claims in ID Token Hint**. Click **Add Claim** to add another custom claim. For each custom claim, including the **email** claim, the corresponding claim/lookup must be configured in the **Attribute Mapping List** section of the corresponding user directory to retrieve the claim values. All the defined custom claims are part of the **urn:iam:authn:userData** section in a key-multivalue attribute format in an ID Token Hint.

SiteMinder to delegate the user authentication to IDSP

a. Select **Authentication Hub Template** from the **Authentication Scheme Type** drop-down list.

b. Select an IDSP provider form the **Provider** drop-down list.
The Issuer URL value is auto-populated based on the value that was specified while configuring the selected IDSP provider.

c. Select the application that was created for SiteMinder in IDSP from the **Application** drop-down list.
The IDSP provider configuration retrieves all the applications that are configured for SiteMinder in IDSP and displays them in the Applications drop-down list.

d. Select **Use Relative Target** to specify whether a relative path name must be used for the Target or resource that the IDSP scheme protects. Select this option if multiple hosts with different domains are protecting the same resource. During runtime, Access Gateway determines the **Access Gateway Redirect URL** and **Access Gateway State Redirect URL** for each host using the relative hostname in an authentication request. If SSL offloading is used before Access Gateway, use the **HttpsPorts** ACO parameter to ensure that Access Gateway redirects using HTTPS URLs.
To define your HTTPS ports, set the value of the **HttpsPorts** ACO parameter to the port numbers that use SSL. Use commas to separate multiple port numbers.
Example: 80,7002
If a server is behind an HTTPS accelerator (that converts HTTPS to HTTP), the requests are treated as SSL connections by your browser but Access Gateway still serves them on HTTP and add appropriate port for the **HttpsPorts**ACO parameter.
Important! This feature works only for resources that reside on Access Gateway.

e. Select the redirect URL that IDSP uses to send authorization code to SiteMinder from the **Access Gateway Redirect URL** drop-down list.
This drop-down lists all the redirect URLs that are configured with the selected application.
The **Access Gateway State Redirect URL** value is displayed based on the selected redirect URL.

f. Select the alias of the public certificate of IDSP from the **Verification Certificate Alias** drop-down list. This list displays all the public certificates that are available in SiteMinder.

g. (If ID Token Hint is encrypted) Select the alias of the private key that must be used for decrypting ID Token Hint from the **Decryption Private Key Alias** drop-down list.

h. (Optional) Define a user attribute lookup in JWT claims that must be used for validating the received ID Token in **ID Token Claim Lookup**. You can specify either user_loginid or any other claim that contains information on the authenticated user. If this parameter is not specified, user_loginid claim that is present in the received ID Token is used for validation.

i. (Optional) Define the number of seconds subtracted from the current time to account for the difference in the system clocks of the machines where SiteMinder and IDSP are installed in **Skew Time**.

j. Select ACR values for different authentication flows in **ACR**. Click **Select** to view the ACR values and their corresponding obligations that are configured in the authentication policy rules of the selected application. Select the required values and click **OK**.

The **ACR** table displays the selected ACRs. If multiple ACRs exist with the same name, selecting one ACR automatically selects the others too. The ACR that is used at runtime is sent in the ID Token Hint. Use the arrow buttons in the table to sort the order of precedence in which the ACRs must be implemented.

Note: As password-based authentication schemes such as HTML Forms are already supported, SiteMinder does not support password-based authentication factors in IDSP.

For basic information on ACRs, see Support for Authentication Context Class References for Stronger Authentication section in the integration overview.

For information on configuring ACRs in IDSP, see ACRs.

k. (Optional) Configure the following fields in the ID Token Hint Generation Setup section:

   a. Select **Send Additional Information in ID Token Hint** field to include the agent name in the **sm_agent** key and domain name in the **sm_domain** key in the urn:iam:authn:clientContext section of an ID Token Hint. If the ID Token Hint is sent to IDSP as part of an MFA authentication chain or in a full delegation where a prior user session exists, these keys are added to the other keys in the urn:iam:authn:clientContext section in the ID Token Hint. If an ID Token Hint is sent to IDSP as part of a full delegation where a prior user session does not exist, the urn:iam:authn:clientContext section includes only these two keys in the ID Token Hint. For information about ID Token Hint, see Identity Propagation in SiteMinder.

   b. Select the signing key alias of SiteMinder in **Signing Certificate Alias**. This value is used for signing ID Token Hint that is sent to IDSP when SiteMinder performs the primary user authentication.

   c. Select the algorithm that SiteMinder must use for signing the ID Token Hint from the **Signing Algorithm** drop-down list.

   d. Specify an alias for the certificate that is used to encrypt ID Token Hint in the **Encryption Certificate Alias** drop-down list.

   e. Specify the JWE algorithm that must be used for encrypting ID Token Hint in the **Encryption Algorithm** drop-down list.

   f. Specify the JWE method that must be used for encrypting ID Token Hint in the **Encryption Method** drop-down list.

l. (Optional) To save the authentication context data that is retrieved in ID Token in the session store for later use in authentication decisions, select the **Persist Authentication Session Variables** option.

m. (Optional only in a test environment) To ignore SSL certificate validation during backchannel communication, select the **Disable SSL Certificate Validation in Authorization Code Flow** option. This option must be cleared in a production environment as the SSL certificate is required for regular secure operations.

n. (Optional only in a test environment) To ignore the signature validation of the ID Token that is generated by IDSP, select **Disable Identity Token Signature Validation**. This option must be cleared in a production environment.

6. Click **Save**.

Note: To configure Multi Factor Authentication Chain Template authentication scheme using XPSExplorer, use the **AuthChaining** option.

Customize Expiry Time of Chain Cookie

The Multi Factor Authentication authentication scheme uses a chain cookie to indicate the status of current authentication scheme in the flow. By default, the chain cookie expiry value is 600 seconds (10 minutes). You can customize the chain cookie expiry time using the **ChainCookieTimeInterval** registry key.

Follow these steps:

1. On Policy Server, add the **ChainCookieTimeInterval** registry key in the following location:

- **Windows**: HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\PolicyServer
- **UNIX**: HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\PolicyServer in the sm.registry file at <siteminder-home>/registry.

2. Set the value of the registry key to any hexadecimal value in seconds.
   - **Minimum value**: 1 second
   - **Maximum value**: Less than 3600 seconds (1 hour)

   Example: 0xA8C for 2700 seconds (45 minuntes)
3. Restart Policy Server.

Map the Authentication Scheme to a Realm
Follow these steps:

1. Navigate to **Policies**, **Domain**, **Realms**.
2. Create or modify a realm.
3. Select the new authentication scheme from the **Authentication Scheme** drop-down list.

For information about configuring a realm, see Realm Dialog Reference. SiteMinder is configured to integrate with IDSP. This concludes the configuration that is required for integrating SiteMinder with IDSP. You can proceed to perform transactions. When a runtime transaction is performed, an authentication scheme is invoked as per the configured authentication type. See Troubleshooting topic if you experience any issues.

Configure the Integration in 12.8.0700.2947 Build

Perform the configuration only after you have understood the integration overview and completed all the steps mentioned in Prerequisites.The integration configuration process is as follows at a high-level when this build is used:

1. Download the installers of the in-progress release.
2. Configure Symantec Identity Security Platform provider.
3. Import certificates into SiteMinder.
4. Configure an authentication scheme in SiteMinder for the integration.
   - Use the **Authentication Hub Template** authentication scheme type for SiteMinder to delegate the user authentication to IDSP.
   - Use the **Multi Factor Authentication Chain Template** authentication scheme type for SiteMinder to perform the primary authentication and IDSP to perform the secondary authentication.
5. Map the authentication scheme to a realm.

Download Installers
You can use the latest installers to deploy a new environment or to upgrade from a previous release environment.

1. Download the integration zip file and unzip the file to navigate to the installers depending on the operating system.
2. Deploy the installers to setup the in-progress release environment. If you are upgrading to the latest release, follow the Upgrading instructions that are described in the SiteMinder 12.8 documentation as the upgrading process is identical in this integration environment too.

Configure IDSP Provider
To establish a communication between SiteMinder and IDSP, configure an IDSP provider in SiteMinder. This is a one-time setup.

Note: If you are a scoped administrator, ensure that you have access to the **IDSP** workspace content with permission to work on the **AuthHub Objects Administration** security category. For information about a scoped administrator, see Create a Scoped Administrator.

Follow these steps:

1. Navigate to **Infrastructure**, **Authentication**, **IDSP Providers** in Administrative UI.
2. Enter a name and description to identify this connection in **Provider Name** and **Description**.
3. Enter the Issuer URL of IDSP tenant in **Issuer URL**. Enter the URL in the following format using the IDSP hostname that was noted as a part of prerequisites:
   `https://ah_hostname/name_of_default_tenant`
   This URL is used to gather the Authorization and Token endpoints of IDSP, and auto-populate them in the integration-specific authentication scheme template.
4. Enter the client ID and client secret of the application that is created in IDSP for the **default** tenant in **Administrative Client ID** and **Administrative Client Secret**. Use the tenant details that were noted as a part of prerequisites.
5. (Optional) Select **Disable SSL Certificate Validation** to disable the validation of SSL certificate while establishing communication between SiteMinder and IDSP.
6. (Optional) If IDSP exposes tenant management and end user operations on different ingress controllers, select the **Use a Different Ingress Controller for Tenant Management** option and define the URL of the ingress controller that is used for the tenant management operations in **Tenant Management Issuer URL**. For information about ingress in IDSP, see *Deploying Ingress* in the IDSP documentation.
   By default, SiteMinder uses the URL that is defined in **Issuer URL** for the tenant management and end user operations. If the **Use a Different Ingress Controller for Tenant Management** option is selected, SiteMinder performs the following steps:
   – Uses the **Issuer URL** for end user operations, and the URL that is defined in **Tenant Management Issuer URL** for tenant management operations.
   – Displays separate Authorization and Token endpoints for the end user and tenant management operations respectively in the **Endpoints** section.
7. (Optional) To allow communication between Administrative UI and IDSP using a proxy, select **Use Proxy for Administrative UI Communication** option and perform the following additional steps:
   a. Specify the hostname and port number of the proxy server in **Proxy Host** and **Proxy Port**.
   b. Specify the user credentials to access the proxy server in **Proxy User** and **Proxy Password**.
      Note: If you do not opt for proxy communication, ensure that Administrative UI can communicate with IDSP.
8. (Optional) To allow communication between Access Gateway and IDSP using a proxy, select **Use Proxy for Access Gateway Communication** option and perform the following additional steps:
   a. Specify the hostname and port number of the proxy server in **Proxy Host** and **Proxy Port**.
   b. Specify the user credentials to access the proxy server in **Proxy User** and **Proxy Password**.
      Note: If you do not opt for proxy communication, ensure that Access Gateway can communicate with IDSP.
9. Click **Save** to save the connection information.
10. Click **Test Connection** to test whether a connection has been established between SiteMinder and IDSP.
    If the **Disable SSL Certificate Validation** option is not selected, ensure that you import the certificate or the root CA of the certificate that is used for enabling SSL in Ingress of IDSP before you test the connection. For information, see the Import Certificates section.
    If the connection is successful, the Authorization and Token Endpoints information of IDSP is displayed in the **Endpoints** section of the dialog.
11. (Optional) To modify the entered details, click **Modify** and make the necessary changes.
12. (Optional) To view the authentication policies that are created in IDSP for this provider and to create rules in a policy, see Manage Authentication Policies.

Import Certificates

The integration requires the following certificates:

- **The signing key-pair that is used for signing id_token_hint, which is sent to IDSP when SiteMinder performs the primary authentication**.

This signing key-pair must be present in SiteMinder. You can use an existing signing key-pair. If you want to use a new signing key-pair, import it using the following steps:

a. Navigate to **Infrastructure**, **X509 Certificate Management**, **Trusted Certificates and Private Keys** in SiteMinder Administrative UI.
b. Import the new signing key-pair. For information about importing certificates, see Import Trusted Certificates and Key Certificate Pairs.

Also, the public certificate within the signing key-pair must be present in IDSP. If the public certificate is readily available, proceed to **Step 2**. If it has to be retrieved from SiteMinder, perform **Step 1 and Step 2**.

a. Navigate to **Infrastructure**, **X509 Certificate Management**, **Trusted Certificates and Private Keys** in SiteMinder Administrative UI, select the public certificate from the list of available certificates, and click **Export**.
b. Import this certificate into IDSP using the `https://ah_hostname/tenant_name/admin/v1/Certs/` endpoint. For example, the following sample decribes how to use **cURL** for importing the certificate into IDSP:

   Important! Ensure that **jq tool** is installed before you perform the following steps. For example, jq tool 1.6 was installed using the following commands in our test environment:

```
wget -O jq https://github.com/stedolan/jq/releases/download/jq-1.6/jq-linux64
chmod +x ./jq
cp jq /usr/bin
```

   a: Acquire Access Token from IDSP

```
export HOSTNAME=ah_hostname
export TENANT=ah_tenant
export CLIENTID=ah_tenant_clientid
export CLIENTSECRET=ah_tenant_clientsecret
export TOKENISSUER=sm_token_issuer

export TOKEN=$(curl -ks -X  POST https://${HOSTNAME}/${TENANT}/oauth2/v1/token -u "${CLIENTID}:
${CLIENTSECRET}" -d 'grant_type=client_credentials&scope=urn:iam:myscopes' | jq '.access_token' | sed
 's/"//g')
```

   b: Import the public certificate within the signing key-pair into IDSP

```
export CERTDATA=$(cat<absolutepath_to_publickeycert>| tr -d '\r' | sed '1d;$d' | tr '\n' ' ' | tr -d '
 ')

curl -k --location --request POST "https://${HOSTNAME}/${TENANT}/admin/v1/Certs/" \
--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
--header "Authorization: Bearer $TOKEN" \
--data "{
\"tokenIssuer\": "\"$TOKENISSUER\"",
\"alias\": \"SSOCert\",
\"b64CertData\": "\"$CERTDATA\""
}"
```

Note: This certificate is required for regular secure operations in production after you have tested the user authentication flow between SiteMinder and IDSP with the **Disable Identity Token Signature Validation** option selected during testing.

- **The Ingress root CA/Ingress server certificate of IDSP**.

  This refers to the certificate or the root CA of the certificate that is used for enabling SSL in Ingress of IDSP. If the certificate or its root CA is readily available, proceed to **Step 2**. Else, perform **Step 1 and Step 2**.

  a. Run the following command on a machine where OpenSSL is installed:

```
openssl s_client -showcerts -servername server -connect $HOSTNAME:443 > cacert.pem
```

  In the above command, **HOSTNAME** refers to the hostname of IDSP.

b. Import the above mentioned certificate or its root CA into SiteMinder. Navigate to **Infrastructure**, **X509 Certificate Management**, **Certificate Authorities** in SiteMinder Administrative UI, and import the certificate or its root CA. For information about importing Certificate Authority, see Certificate Authorities.

Note: This certificate is required for regular secure operations in production after you have tested connectivity between SiteMinder and IDSP with the **Disable SSL Certificate Validation** option selected during testing.

- **The public certificate of the signing key that is used by IDSP tenant to sign the ID Token**.
  a. Retrieve the certificate from IDSP using the `https://ah_hostname/tenant_name/admin/v1/SigningCert` API.

  For example, the following sample decribes how to use **cURL** for retrieving the certificate from IDSP:

  Important! Ensure that **jq tool** is installed before you perform the following steps. For example, jq tool 1.6 was installed using the following commands in our test environment:

```
wget -O jq https://github.com/stedolan/jq/releases/download/jq-1.6/jq-linux64
chmod +x ./jq
cp jq /usr/bin
```

  a: Acquire Access Token from IDSP

```
export HOSTNAME=ah_hostname_value
export TENANT=ah_tenant
export CLIENTID=ah_tenant_clientid
export CLIENTSECRET=ah_tenant_clientsecret
export TOKENISSUER=sm_token_issuer
export TOKEN=$(curl -ks -X  POST https://${HOSTNAME}/${TENANT}/oauth2/v1/token -u "${CLIENTID}:
${CLIENTSECRET}" -d 'grant_type=client_credentials&scope=urn:iam:myscopes' | jq '.access_token' | sed
 's/"//g')
```

  b: Retrieve the certificate from IDSP

```
curl -ks --location --request GET "https://${HOSTNAME}/${TENANT}/admin/v1/SigningCert" --header
 "Authorization: Bearer ${TOKEN}" |  jq '.b64CertData' | sed 's/"//g' | tr -d '\n' > chk.crt
```

  b. Navigate to **Infrastructure**, **X509 Certificate Management**, **Trusted Certificates and Private Keys** in SiteMinder Administrative UI, and import the public certificate of IDSP that was retrieved in the above step. For information about importing certificates, see Import Trusted Certificates and Key Certificate Pairs.

Configure Authentication Scheme

The authentication scheme type depends on the scenario that you want to implement.

- Use the **Authentication Hub Template** authentication scheme type for SiteMinder to delegate the user authentication to IDSP.
- Use the **Multi Factor Authentication Chain Template** authentication scheme type for SiteMinder to perform the primary authentication and IDSP to perform the secondary authentication.

  SiteMinder can perform primary authentication using one of the following authentication schemes:
  - HTML Form
  - X509 Client Certificate
  - X509 Client Certificate or HTML Form
  - X509 Client Certificate and HTML Form
  - Windows
  - Windows Fallback to HTML Form
  - Kerberos
  - Kerberos Fallback to HTML Form
  - RADIUS

Both the templates support PKCE-enabled Authorization Code Flow.

Follow these steps:

1. Navigate to **Infrastructure**, **Authentication**, **Authentication Schemes** in Administrative UI.
2. Click **Create Authentication Scheme**, select **Create a new object of type Authentication Scheme**, and click **OK**.
3. Enter a **Name** and **Description** of the authentication scheme.
4. Select **Protection Level** to provide an additional measure of flexibility in access control. Define a level from 1 through 1000.
5. Perform *one* of the following steps depending on the integration scenario that you want to implement:

   SiteMinder to perform the primary authentication and IDSP to perform the secondary authentication

   a. Select **Multi Factor Authentication Chain Template** from the **Authentication Scheme Type** drop-down list.
   b. Specify the authentication scheme details with which SiteMinder must perform primary authentication in **Primary Authentication Scheme**. Select an authentication scheme type from the supported authentication scheme types that can be used for primary authentication, and then select an authentication scheme that is already created for the selected authentication scheme type.
   c. **(Only for Windows or Kerberos authentication scheme)** Select a fallback scheme that must be used if primary authentication fails from **Primary Authentication Fallback Scheme** drop-down list.
   d. Specify the authentication scheme details with which IDSP must perform secondary authentication in **Secondary Authentication Scheme**. Select **IDSP** as the authentication scheme type and then select an authentication scheme that is already created for this authentication scheme type.

      The authentication chain expression that is achieved using this authentication scheme type is auto-populated in **Expression**. **Token Issuer** displays the tokenIssuer value of the configured IDSP application, which is associated with the selected secondary authentication scheme.
   e. Specify the signing key alias of SiteMinder in **Signing Certificate Alias**. This value is used for signing id_token_hint that is sent to IDSP when SiteMinder performs the primary user authentication.
   f. Select the algorithm that SiteMinder must use for signing the ID Token Hint from the **Signing Algorithm** drop-down list.
   g. (Optional) Configure the following fields to encrypt ID Token Hint:
      a. Select the alias that was defined while importing the IDSP signing certificate into SiteMinder to encrypt ID Token Hint from the **Encryption Certificate Alias** drop-down list.
      b. Select the JWE algorithm that must be used for encrypting ID Token Hint from the **Encryption Algorithm** drop-down list.
      c. Select the JWE method that must be used for encrypting ID Token Hint from the **Encryption Method** drop-down list.
   h. Define the SiteMinder user store attribute whose value must be used to look up a user in IDSP in **Basic User Attribute Lookup**. This value is sent in the **sub** and **user_loginid** in id_token_hint.
      The possible values are:
      - SM_UNIVID, a SiteMinder reserved word that represents the UniversalID attribute of a user, as configured in the user directory
      - The name of a virtual attribute, as defined in the **Attribute Mapping** section of the user directory
      - The name of a physical user attribute that contains the required value
   i. Select **Enable Propagation of Extended User Attributes in ID Token Hint** to enable SiteMinder to propagate an authentication context and a set of identity attributes to IDSP. For information about identity propagation, see Identity Propagation in SiteMinder.
   j. (Optional) To send any custom claims in ID Token Hint, specify the names of the required custom claims in **Custom Claims in ID Token Hint**. Click **Add Claim** to add another custom claim. For each custom claim, including the **email** claim, the corresponding claim/lookup must be configured in the **Attribute Mapping List** section of the corresponding user directory to retrieve the claim values. All the defined custom claims are part of the **urn:iam:authn:userData** section in a key-multivalue attribute format in an ID Token Hint.

   SiteMinder to delegate the user authentication to IDSP

   Select **Authentication Hub Template** from the **Authentication Scheme Type** drop-down list.

a. Select an IDSP provider form the **Provider** drop-down list.
   The Issuer URL value is auto-populated based on the value that was specified while configuring the selected IDSP provider.
b. Select the application that was created for SiteMinder in IDSP from the **Application** drop-down list.
   The IDSP provider configuration retrieves all the applications that are configured for SiteMinder in IDSP and displays them in the Applications drop-down list.
c. Select **Use Relative Target** to specify whether a relative path name must be used for the Target or resource that the IDSP scheme protects. Select this option if multiple hosts with different domains are protecting the same resource. During runtime, Access Gateway determines the **Access Gateway Redirect URL** and **Access Gateway State Redirect URL** for each host using the relative hostname in an authentication request. If SSL offloading is used before Access Gateway, use the **HttpsPorts** ACO parameter to ensure that Access Gateway redirects using HTTPS URLs.
   To define your HTTPS ports, set the value of the **HttpsPorts** ACO parameter to the port numbers that use SSL. Use commas to separate multiple port numbers.
   Example: 80,7002
   If a server is behind an HTTPS accelerator (that converts HTTPS to HTTP), the requests are treated as SSL connections by your browser but Access Gateway still serves them on HTTP and add appropriate port for the **HttpsPorts**ACO parameter.
   Important! This feature works only for resources that reside on Access Gateway.
d. Select the redirect URL that IDSP uses to send authorization code to SiteMinder from the **Access Gateway Redirect URL** drop-down list.
   This drop-down lists all the redirect URLs that are configured with the selected application.
   The **Access Gateway State Redirect URL** value is displayed based on the selected redirect URL.
e. Select the alias of the public certificate of IDSP from the **Verification Certificate Alias** drop-down list. This list displays all the public certificates that are available in SiteMinder.
f. (If ID Token Hint is encrypted) Select the alias of the private key that must be used for decrypting ID Token Hint from the **Decryption Private Key Alias** drop-down list.
g. (Optional) Define a user attribute lookup in JWT claims that must be used for validating the received ID Token in **ID Token Claim Lookup**. You can specify either user_loginid or any other claim that contains information on the authenticated user. If this parameter is not specified, user_loginid claim that is present in the received ID Token is used for validation.
h. (Optional) Define the number of seconds subtracted from the current time to account for the difference in the system clocks of the machines where SiteMinder and IDSP are installed in **Skew Time**.
i. Select ACR values for different authentication flows in **ACR**. Click **Select** to view the ACR values and their corresponding obligations that are configured in the authentication policy rules of the selected application. Select the requried values and click **OK**.
   The **ACR** table displays the selected ACRs. If multiple ACRs exist with the same name, selecting one ACR automatically selects the others too. The ACR that is used at runtime is sent in the ID Token Hint. Use the arrow buttons in the table to sort the order of precedence in which the ACRs must be implemented.
   Note: As password-based authentication schemes such as HTML Forms are already supported, SiteMinder does not support password-based authentication factors in IDSP.
   For basic information on ACRs, see Support for Authentication Context Class References for Stronger Authentication section in the integration overview.
   For information on configuring ACRs in IDSP, see ACRs.
j. (Optional) To save the authentication context data that is retrieved in ID Token in the session store for later use in authentication decisions, select the **Persist Authentication Session Variables** option.
k. (Optional only in a test environment) To ignore SSL certificate validation during backchannel communication, select the **Disable SSL Certificate Validation in Authorization Code Flow** option. This option must be cleared in a production environment as the SSL certificate is required for regular secure operations.

l. (Optional only in a test environment) To ignore the signature validation of the ID Token that is generated by IDSP, select **Disable Identity Token Signature Validation**. This option must be cleared in a production environment.

6. Click **Save**.

Note: To configure Multi Factor Authentication Chain Template authentication scheme using XPSExplorer, use the **AuthChaining** option.

Map the Authentication Scheme to a Realm

Follow these steps:

1. Navigate to **Policies**, **Domain**, **Realms**.
2. Create or modify a realm.
3. Select the new authentication scheme from the **Authentication Scheme** drop-down list.

For information about configuring a realm, see Realm Dialog Reference.SiteMinder is configured to integrate with IDSP. This concludes the configuration that is required for integrating SiteMinder with IDSP. You can proceed to perform transactions. When a runtime transaction is performed, an authentication scheme is invoked as per the configured authentication type.See Troubleshooting topic if you experience any issues.

Support for Symantec Identity Security Platform as a Fallback Authentication Scheme in Authentication Chain Template

From SiteMinder 12.8.0700.2989, you can now select Symantec Identity Security Platform as a fallback authentication scheme for Kerberos and Windows authentication schemes in an Authentication Chain Template.

Verify Prerequisites

Complete the following steps before you configure the feature:

1. Configure a Kerberos authentication scheme or Windows authentication scheme for using it as a default authentication scheme in an Authentication Chain Template.
2. Configure a IDSP authentication scheme for using it as a fallback authentication scheme in the Authentication Chain template.

Configure IDSP as a Fallback Authentication Scheme in Authentication Chain Template

Use the authentication schemes that are created in the prerequisites in the Authentication Chain template.

Follow these steps:

1. Log in to Administrative UI.
2. Navigate to **Infrastructure**, **Authentication**, **Authentication Schemes**.
3. Click **Create Authentication Scheme**, select **Create a new object of type Authentication Scheme**, and click **OK**.
4. Enter a **Name** and **Description** of the authentication scheme.
5. Select **Authentication Chain Template** from **Authentication Scheme Type**.
6. Enter **Protection Level** to provide an additional measure of flexibility in access control. Define a level from 1 through 1000.
7. Configure the following fields in the **Authentication Scheme Instance** section:
   a. For **Default Authentication Scheme**, select **Kerberos** or **Windows** authentication scheme templates from **Authentication Scheme Type** and then select the corresponding authentication scheme instance from **Authentication Instances**.
   b. For **Fallback Authentication Scheme**, select **IDSP** from **Authentication Scheme Type** and then select its authentication scheme instance from **Authentication Instances**.

8. Click **Submit**.

Manage Authentication Policies in SiteMinder

SiteMinder Administrative UI lets you view the authentication policies that are created in Symantec Identity Security Platform for a provider, and create rules in a policy. Any rules that are created in SiteMinder are automatically updated in IDSP.

View Authentication Policy and Create a Rule
Follow these steps:

1. Navigate to **Infrastructure**, **Authentication**, **IDSP Providers**.
2. Select an IDSP provider whose authentication policies you want to view. Click **Actions**, **View & Modify** option of the IDSP provider.
3. Click the **Authentication Policy Management** tab.
   A list of authentication policies that are created for the provider in IDSP is displayed.
4. Click **Actions**, **View & Modify** of the policy that you want to view or modify.
   The **Providers** dialog displays the information about the selected authentication policy, including its rules.
5. (Optional) Edit the policy title and description, if required, in **Name** and **Description** fields.
   Note: The **Policy Id** and **App Names** fields are read-only fields.
6. Complete the following fields to create a rule:
   Important! For detailed information about rules and each field, see Managing Policies.
   - **ACR**
     Defines the ACR values within the authentication request that the Authorization server is requested to use for the processing of requests from the Client. Enter a value without spaces. Once this rule is saved, you can click the **Dependencies** option in the table to view the authentication schemes and realms that are associated with this ACR.
   - **Description**
     Defines a brief description of the new rule.
   - **Channel**
     Defines the medium through which a user can connect to IDSP. The possible values are Web and/or Mobile. If the policy evaluation results in a match between the user input and the "channel" that has been configured in the condition, then the condition is met and a result is presented to the user. If the "effect" results in "allow", then the user is set forth with an obligation which defines the authentication flow. If the "effect" results in deny, the flow ends there.
   - **Risk Operator** and **Risk Score**
     Determines the likelihood of the authentication attempt being malicious or fraudulent. Enter an integer within the 0-100 range as a risk value and select the operator to determine the risk score.
   - **AuthN Factors**
     Defines the user workflow for authentication. Here, multiple obligations can be returned by the solution, based on the policy evaluation. The authentication factors supported by the solution are FIDO, SMS OTP/Email OTP, PUSH, and MOBILE OTP. As a user, you can start the authentication flow from here based on the obligation returned. The obligation can have primary authentication and secondary authentication.
     Enter a value in the *obligation_type*:*precedence* format. Separate multiple values with a space. For example, SMSOTP:1 EMAILOTP:2 PUSH:3
   - **IDP Names**
     Defines the logical name that represents the identity namespace of a user in a policy condition. It is used in scenarios where the client uses Zero Footprint Identity.
   - **IP Addresses or Range** and **IP Address Condition**

Specifies the connectivity to the VIDSP solution from a specific or a range of IP addresses. If the policy evaluation results in a match between the user input and the "ipAddress" range that has been configured in the condition, then the condition is met and a result is presented to the user.

7. Click **Add**.
8. Click **Save**.

Support for Password Policies in Multi Factor Authentication Chain

SiteMinder invokes a password policy after a user is authenticated. Password policies define rules and restrictions that govern the following:

- Password expiration
- Composition
- Usage

The following table describes the general behavior of password policies based on the user status:

Scenario	Behavior
Change Password, Expired Password	The Change Password dialog is displayed after only the completion of MFA chain.
Disabled User	A message indicating the disabled status of the user is displayed when SiteMinder completes its side of user authentication.

For more information about password policies and how to configure them, see Password Services and Policies.

When Multi Factor Authentication Chain Template is used for user authentication, SiteMinder invokes a password policy only when at least one password policies-enabled authentication scheme in the authentication chain has been run. If this condition is met, SiteMinder invokes the password policy after the MFA chain is complete.

For example, the following table describes scenarios when a password policy is invoked:

Primary Authentication Scheme	Password Policy Support Status	Primary Authentication Fallback Scheme	Password Policy Support in Primary Authentication Fallback Scheme	Secondary Authentication Scheme	Behavior
HTML Forms	Enabled	Not Applicable as fallback scheme is not supported in HTML Forms	Not Applicable	Symantec Identity Security Platform	If primary authentication is successful, password policy is invoked after MFA chain is complete.
HTML Forms	Disabled	Not Applicable as fallback scheme is not supported in HTML Forms	Not Applicable	IDSP	As password policy is not enabled in the primary authentication scheme, password policy is not invoked.
X509	Not Applicable as password policies is not supported in X509	Not Applicable	Not Applicable	IDSP	As X509 authentication scheme does not support password policy, a password policy is not invoked.

Kerberos	Enabled	HTML Forms	Enabled	IDSP	If primary authentication is successful, password policy is invoked after MFA chain is complete. If primary authentication fails and fallback authentication is successful, password policy is invoked after MFA chain is complete.
Kerberos	Enabled	HTML Forms	Disabled	IDSP	If primary authentication is successful, password policy is invoked after MFA chain is complete. If primary authentication fails and fallback authentication is successful, password policy is not invoked.
Kerberos	Disabled	HTML Forms	Enabled	IDSP	If primary authentication is successful, password policy is not invoked. If primary authentication fails and fallback authentication is successful, password policy is invoked after MFA chain is complete.
Kerberos	Disabled	HTML Forms	Disabled	IDSP	If primary authentication is successful, password policy is not invoked. If primary authentication fails and fallback authentication is successful, password policy is not invoked.

Support for Symantec Identity Security Platform Risk Service

Symantec Identity Security Platform provides Risk service that works in real time to evaluate context and calculate a risk score. The risk server calculates the risk score for any given access attempt based on a predefined set of rules. The calculated risk score is fed into the policies to decide whether to authorize the current activity, request step-up authentication, and/or send an alert or block the activity. Users are then presented with authentication options appropriate to that risk level. This approach provides the organization with a transparent layer of protection against identity theft, data breaches, and fraud. For information about the Risk service, see Managing Risks.

As a part of the integration, SiteMinder leverages the risk score that is provided by IDSP in ID Token when the IDSP template is used. The risk score can be used for the following purposes:

- SiteMinder can use the risk score in its authorization decisions. To enable this support, SiteMinder must be configured at either realm level or active policy expression level.
- Application can use the risk score for its own decision making purposes. To enable this support, SiteMinder must be configured at responses level to be able to pass the risk score to application through a cookie or response header.

The following process describes how SiteMinder uses the risk score for authorization decisions when Risk service is configured in IDSP and confidence level is configured in SiteMinder:

1. IDSP evaluates the risk score after user authentication.
2. IDSP includes risk score in the ID Token that is sent to SiteMinder. The risk score is populated in the user_riskscore parameter in ID Token.
3. SiteMinder converts the risk score into a confidence level using the following algebraic formula:
   (100-risk score) * 10 = confidence level
   A higher confidence level corresponds to a higher level of credential assurance.
4. SiteMinder includes the confidence level into the session cookie of the user in the SM_USER_CONFIDENCE_LEVEL parameter.
5. When the user requests access to protected resources, Policy Server compares the value of SM_USER_CONFIDENCE_LEVEL with the value either of the following values:
   - **Realm**: Minimum Confidence Level
   - **Active Policy Expression**: Function Parameters
6. If the SM_USER_CONFIDENCE_LEVEL value is equal to or greater than the confidence level value defined in realm or active policy expression, the user request is allowed. Else, the request is denied.

Configure Risk Service in IDSP

The Risk service must be configured in IDSP for the integration to support the feature. For information about the configuration, see Managing Risks.

Enable Confidence Level Support in SiteMinder

To support the Risk score in SiteMinder, enable the confidence level support in SiteMinder and then set a confidence level.

For enabling SiteMinder to use risk score for authorization decisions at realm level, follow these steps:

1. Log in to a Policy Server host system.
2. Start the XPSConfig utility.
   XPSConfig prompts for an option.
3. Enter **SM** and press **Enter**.
   The list of options displays and XPSConfig promtps for an option.
4. Filter the list by entering **F** and typing the string **Confidence**.
   The **ConfidenceLevelSupportEnabled** parameter is displayed.

5. Enter the number associated with the **ConfidenceLevelSupportEnabled** parameter to modify it.
6. Enter **C** to change the parameter value.
   Default Value: False
   The pending value of the parameter changes to **True**.
7. Quit the XPSConfig utility.
8. Restart Policy Server.
   The confidence level support is now enabled. The Realm dialog now displays the **Risk Factor Support** section with the **Minimum Confidence Level** field.
9. Open Administrative UI and navigate to **Policies**, **Domain**, **Realms**.
10. Create a realm or modify an existing realm.
11. Enter a minimum confidence level to apply to all resources in the realm in the **Minimum Confidence Level** field.
12. Complete the other necessary fields.
13. Save the changes.

For enabling SiteMinder to use risk score for authorization decisions at an active policy expression level, follow these steps:

1. From the Administrative UI, click **Policies**, **Domain**, **Domains**.
2. Modify the policy domain that is used for the integration.
3. Click the **Policies** tab.
4. Click **Create**.
5. Click the **Policies** tab.
6. Click the edit icon for the policy.
7. Complete the following steps in the Active Policy Expression area:
   a. Enter the library name as **smriskactiveexpr**.
   b. Enter the function name as **CheckConfidenceLevel**.
   c. Enter a confidence level within the 1-1000 range in the **Function Parameters** field.
8. Click **OK**.
9. Click **Submit**. The confidence level is applied to the resources (rules) bound to the policy.

Support for SiteMinder Locale in IDSP

Symantec Identity Security Platform displays its content in the same locale as that of SiteMinder in all the flows. By default, the following languages that are supported by SiteMinder are also supported by IDSP:

- en-US
- es-ES
- fr-FR
- ja-JP
- pt-BR

To indicate the locale used, SiteMinder includes a new parameter that is named **ui_locales** in the authentication request to IDSP. For example, if SiteMinder is configured to use French, the authentication request includes the **ui_locales=fr-FR** parameter. This parameter is not sent if the **Localization** ACO parameter is set to **no**.

To add a locale in SiteMinder, perform the following steps:

- **For Form as the primary authentication scheme in the Multi Factor Authentication authentication scheme**: Follow the steps in the Localization Name Value Pairs section.
- **For any other non-Form authentication scheme as the primary authentication scheme in the Multi Factor Authentication authentication scheme and the IDSP authentication scheme**: By default, the en-US, es-ES, fr-FR, ja-JP, and pt-BR locales are supported. To add a locale, perform the following steps:
  a. Open the Agent ACO that is used for the integration.
  b. Add the **SupportedLocales** ACO parameter.
  c. Add the new locale to be supported in the *language_code-COUNTRY_CODE* format. Use a comma to separate multiple values.
  Example: To add Swedish (Finland), add **sv-FI**.
  d. Save the change.

Send Custom Data from Application to IDSP

Applications can send custom data to Symantec Identity Security Platform through SiteMinder. This feature lets IDSP use any Client-specific information for user authentication. For example, consider that a SiteMinder-protected application uses an internal transaction ID and IDSP must use this transaction ID while evaluating the risk factor during user authentication. The application can send the transaction ID to IDSP using SiteMinder, and then IDSP uses the information for further processing. For information about how IDSP uses custom data to perform a risk evaluation, see Integrating with External Risk Provider.

To support this requirement, SiteMinder now provides a special @directive, **smcustomdata**, in its FCC file to let a Client application specify any custom data to be sent over to IDSP. This feature is *supported in only HTML Forms and its combination authentication schemes*.

The following process describes how custom data can be sent from application to IDSP:

1. Application customizes the forms authentication template and restarts Access Gateway. For information about customizing the forms authentication template, see Configure HTML Forms Authentication.
   Client application adds all the required directives in the FCC file along with the @smcustomdata directive in the following key-value pair format:
   `@smcustomdata=<keyname_1>=<keyvalue_1>,<keyname_2>=<keyvalue_2>,<keyname_n>=<keyvalue_n>`
   Examples:
   – To send a transaction ID along with its description, enter the following value:
   `@smcustomdata=TID=6319863,TID_DESC=ID_Post_AuthnAz`
   – To send a dynamic SiteMinder transaction ID, enter the following value:
   ```
   @username=%USER%
   @smheaders=SM_TRANSACTIONID
   @smcustomdata=TID=%SM_TRANSACTIONID%
   ```
   – To send the data that is entered in the HTML Input field (<input type="text" name="TRANSID">) in the FCC file, enter the following value:
   `@smcustomdata=TID=%TRANSID%`
   – To send the data that is in the HTML AutoPost field (<input type=hidden name="TRANSID">) in the FCC file, enter the following value:
   `@smcustomdata=TID=%TRANSID%`
2. Client application accesses a resource that is protected by SiteMinder and enters the authentication credentials.
3. SiteMinder processes the login information and generates ID Token Hint.
   The smcustomdata directive information is part of the urn:iam:authn:clientContext section in the generated ID Token Hint.
   Example:
   `"urn:iam:authn:clientContext": [`

```
{
"value": 6319863
"key": "TID"
},
{
"value": "ID_Post_AuthnAz"
"key": "TID_DESC"
}
]
```

In IDSP, the urn:iam:authn:clientContext section is sent to an SPI through the **clientContext** parameter in the /auth/v1/authenticate call. For information about using clientContext, see Integrating with External Risk Provider.

4. IDSP completes its authentication and generates ID Token.

Identity Propagation in SiteMinder

The identity propagation feature support in this integration enables SiteMinder to propagate an authentication context and a set of identity attributes to Symantec Identity Security Platform. SiteMinder and IDSP exchange information to perform the following actions:

- Perform a granular authentication
- Provide a frictionless authentication experience to a user, even during a step-up authentication

Benefits

This feature provides the following benefits:

- SiteMinder users can leverage the authentication capabilities of IDSP policies for a granular user authentication. For example, IDSP policies can be used to determine an authentication method at runtime.
- A user is challenged only for authentication methods that are not used in creating the existing user session.

Context Propagated in ID Token Hint

SiteMinder sends ID Token Hint to IDSP in the following scenarios:

- Scenario 1: When SiteMinder performs primary authentication using the Multi Factor Authentication Chain Template authentication scheme type.
- Scenario 2: During a step-up authentication, when IDSP performs delegated authentication using the IDSP Template authentication scheme type.
- (From 12.8.0700.2989 Build) Scenario 3: When IDSP performs delegated authentication using the IDSP Template authentication scheme type, without a prior user session, if **Send Additional Information in ID Token Hint** is selected in IDSP authentication scheme and if the **OnAccessInadequatePrivilege** event is added in the rules configuration for authorization events.

SiteMinder sends the following parameters in ID Token Hint to IDSP in Scenarios 1 and 2:

Note: These parameters contain default, non-configurable values.

Parameter	Description
sub	Identifies the user login that is used to assert the identity of the user.
aud	Identifies the Client ID of the application that is created for SiteMinder in IDSP

auth_time	Identifies the timestamp at which the user was authenticated by SiteMinder
iss	Identifies the issuer of ID Token Hint
exp	Identifies the expiry time of ID Token Hint
nonce	Identifies the unique identifier of ID Token Hint that prevents the misuse of it
iat	Identifies the timestamp at which ID Token Hint was issued
jti	Identifies the unique identifier of ID Token Hint
nbf	Identifies the timestamp before which ID Token Hint must not be used
user_universalid	Identifies the Universal ID attribute of the user that is configured in the user directory in SiteMinder
user_dn	Identifies user DN of the authenticated user
urn:iam:login_flow:expires_in_seconds	Defines the time in which the entire authentication flow must complete at IDSP. The default value is 600 seconds. This value can be modified using **ChainCookieTimeInterval** registry key. For information, see Configure the Integration in 12.8.0700.2989 Build.

To propagate identity, the following additional parameters are sent in the ID Token Hint in the Multi Factor Authentication Chain and IDSP authentication schemes:

Parameter	**Description**
email (if configured in user directory)	Identifies the email address of the user, if this value is configured in the user directory in SiteMinder. **(From 2022.Jun.01 build)** If **zeroFootPrint** is enabled in IDSP and multiple values are defined in the email claim in the user directory, ID Token Hint sends only the first value that is defined in the claim. To send the rest of the values in this claim, you must define them as custom claims using the Multi Factor Authentication authentication scheme. For information about configuring custom claims, see Configure the Integration of SiteMinder with IDSP.
phone_number (if configured in user directory)	Identifies the phone number of the user, if this value is configured in the user directory in SiteMinder. **(From 2022.Jun.01 build)** If **zeroFootPrint** is enabled in IDSP and multiple values are defined in the phone_number claim in the user directory, ID Token Hint sends only the first value that is defined in the claim. To send the rest of the values in this claim, you must define them as custom claims using the Multi Factor Authentication authentication scheme. For information about configuring custom claims, see Configure the Integration of SiteMinder with IDSP.
amr	Lists the authentication methods that were used by SiteMinder for authenticating the user
idp_name	Identifies the name of the user directory where the user was disambiguated in Siteminder

auto_inline_enrollment	Specifies whether the user must be auto-enrolled in IDSP using the email and phone number that are provided in ID Token Hint. If this value is set to true, IDSP enrolls the user in its user repository, based on the ID Token Hint. Also, setting this value to true overrides the auto-enroll setting that is configured in IDSP. If this value is set to false, the user is not enrolled in IDSP.
given_name (if configured in user directory)	Identifies the given name of the user, if this value is configured in the user directory in SiteMinder
family_name (if configured in user directory)	Identifies the family name of the user, if this value is configured in the user directory in SiteMinder
middle_name (if configured in user directory)	Identifies the middle name of the user, if this value is configured in the user directory in SiteMinder
idp_type	Identifies the identity provider type information as siteminder. This information is required by IDSP to distinguish the identity provider type.
idp_user_guid	Identifies the GUID of a user that is authenticated by SiteMinder. The claim value is retrieved from the **Definition** field of the Attribute Mappings List section of the corresponding user directory. If the claim value is stored in a binary format in the user directory, suffix the value in the **Definition** field with **;binary**. **Example**: objectGUID;binary If the claim value is not defined in the user directory, its value is empty in ID Token Hint.
idp_id	Identifies the unique, immutable identifier of a user within a user directory. The claim value is retrieved from the Definition field of the Attribute Mappings List section of the corresponding user directory. If the claim value is not defined in the user directory, its value is empty in ID Token Hint.
urn:iam:authn:userData	Lists any custom claims that are defined in the Multi Factor Authentication Chain Template authentication scheme
urn:iam:authn:clientContext	Lists the custom data that is propogated to IDSP through SiteMinder. For information, see Send Custom Data from Application to IDSP. From 12.8.0700.2989 Build, if **Send Additional Information in ID Token Hint** is selected in IDSP authentication scheme, this includes the agent name in the **sm_agent** key and domain name in the **sm_domain** key. If a user session exists, these keys are sent along with the other keys in the clientContext section. If a user session does not exist, only these keys are sent in the section.

Example: ID Token Hint when a user session exists

```
{
"sub": "NadiaKITE",
"idp_name": "us",
"user_universalid": "NadiaKITE",
"user_dn": "uid=NadiaKITE,ou=People,o=security.com",
"amr": [
 "PWD"
```

```
],
"user_loginid": "NadiaKITE",
"idp_id": "",
"iss": "https://example.com/iam/siteminder",
"urn:iam:authn:userData": [
 {
   "key": "userid",
   "value": [
     "NadiaKITE"
   ]
 }
],
"urn:iam:login_flow:expires_in_seconds": 600,
"nonce": "27d4a44-d87f85a4-3611fad-6a73ae94-63d27852-5",
"aud": "8aa4651-fb66-415e-9408-801e5ca8c1",
"nbf": 16341440,
"auth_time": 169341440,
"idp_user_guid": "",
"exp": 1696341740,
"iat": 1696341440,
"urn:iam:authn:clientContext": [
 {
   "key": "sm_key1",
   "value": "value1"
 },
 {
   "key": "sm_key2",
   "value": "value2"
 },
 {
   "key": "sm_agent",
   "value": "agentNA"
 },
 {
   "key": "sm_domain",
   "value": "domainNA"
 }
],
"idp_type": "siteminder",
"jti": "27d4a414-d87f85a4-3611cfad-6a73ae94-63d27852-5"
}
```


Example: ID Token Hint when a user session does not exist


```
{
"aud": "8aa46561-fb66-415e-9408-801e5cdba8c1",
"nbf": 1696342137,
"auth_time": 1696342137,
"iss": "https://example.com/iam/siteminder",
"exp": 1696342437,
"nonce": "20fab8a8-4fb84d95-038bab62-5974a1ca-e1cf5236-ab4",
"iat": 1696342137,
"urn:iam:authn:clientContext": [
```

```
  {
    "key": "sm_agent",
    "value": "agentNA"
  },
  {
    "key": "sm_domain",
    "value": "domainNA"
  }
 ],
"jti": "20fab8a8-4fb84d95-038bab62-5974a1ca-e1cf5236-ab4"
}
```

Supported Use Cases

Use Case	Behavior
A user with an existing session accesses a protected resource that requires step-up authentication. The resource authentication is delegated to IDSP.	• SiteMinder sends **user ID** in ID Token Hint. • IDSP challenges the user only for OTP/password/etc but not for a username.
A user without an existing session (new user) accesses an MFA-protected resource. IDSP is required to perform granular authentication depending on the UD type.	• SiteMinder sends **user ID** and user directory name in ID Token Hint. • IDSP uses the information to decide how the user must be further challenged. For example, consider UD1 and UD2 as the user directories in the environment. Users of UD1 must be authenticated using FIDO and users of UD2 must be authenticated using OTP. Once a user is authenticated by SiteMinder, the name of the user directory to which the user belongs is sent in ID Token Hint to IDSP. If the user belongs to UD1, user is challenged using FIDO. If the user belongs to UD2, user is challenged with OTP to complete authentication by IDSP. The user is challenged only for OTP/password/etc but not for a username.
A user exists in the user repository of SiteMinder but not in the user repository of IDSP. User must be auto-registered in the IDSP user repository based on the information from the user directory that is configured in SiteMinder.	• SiteMinder sends **universal ID**, **email**, **phone_number** in ID Token Hint. • IDSP uses this information to auto-register the user.
A user with an existing session accesses a protected resource that requires step-up authentication and IDSP is required to perform granular authentication depending on the UD type. The resource authentication is delegated to IDSP.	• SiteMinder sends **user ID** and **user directory name** in ID Token Hint. • IDSP uses the information to decide how the user must be further authenticated. The user is challenged only for OTP/password/etc but not for a username. For example, consider UD1 and UD2 as the user directories in the environment. Users of UD1 must be authenticated using FIDO and users of UD2 must be authenticated using OTP. Once a user is authenticated by SiteMinder, the name of the user directory to which the user belongs is sent in ID Token Hint to IDSP. If the user belongs to UD1, user is challenged using FIDO. If the user belongs to UD2, user is challenged with OTP to complete authentication by IDSP. The user is challenged only for OTP/password/etc but not for a username.

A user with an existing session accesses a protected resource that requires step-up authentication and the user is not present in IDSP user repository. User must be auto-registered in IDSP user repository based on the information from the user directory that is configured in SiteMinder. The resource authentication is delegated to IDSP.	• SiteMinder sends **Universal ID**, **email**, **phone_number** in ID Token Hint. • IDSP uses this information to auto-register the user and then continues with the authentication challenge. The user is challenged only for OTP/password/etc but not for a username.
Identity Propagation Using the amr value in ID Token Hint	
A new user accesses an MFA-protected resource. The resource authentication is delegated to IDSP after SiteMinder performs the primary authentication.	• SiteMinder performs the primary authentication and sends ID Token Hint to IDSP. The value of **amr** in ID Token Hint contains the primary authentication method information. • IDSP uses the information in ID Token Hint IDSP uses the amr value to avoid performing the same obligation and challenges the user only for the other configured obligations. IDSP sends ID Token to SiteMinder with information about primary and secondary authentications in the amr value. The Session Spec attribute of the SMSESSION cookie that is newly created contains information about the authentication methods that are executed from SiteMinder and IDSP.
A user with an existing session in SiteMinder accesses a protected resource that requires step-up authentication. The resource authentication is delegated to IDSP.	• SiteMinder sends ID Token Hint with **amr** value indicating the previously executed authentication method that was used to create the Session Spec attribute of the SMSESSION cookie. • IDSP uses the information to avoid challenging the user for the same authentication method that was used to create the SMSESSION cookie and to challenge the user only for other configured authentication methods. For example, consider that a user requests access to a resource that is protected using HTML Forms authentication scheme. SiteMinder challenges the user for username and password to authenticate the user and uses this information to create SMSESSION cookie. Now, consider that the user requests access to another resource that is protected by IDSP Scheme where an IDSP policy is configured with the PWD and OTP obligations. Once the step-up authentication flow is initiated, SiteMinder sends ID Token Hint with amr value indicating that the user was already authenticated using PWD. IDSP uses this information to avoid challenging the user for PWD as per its configured obligations, and to challenge the user only for OTP. The Session Spec attribute of the newly created SMSESSION cookie contains information about the authentication methods that are executed from SiteMinder and IDSP.

A user with an existing session that is created using Multi Factor Authentication Scheme, accesses a protected resource that requires step-up authentication. The resource authentication is delegated to IDSP.	• SiteMinder sends ID Token Hint with **amr** value indicating the previously executed authentication methods that were used by SiteMinder and IDSP to create the Session Spec attribute of the SMSESSION cookie. • IDSP uses the information to avoid challenging the user for the same authentication method that was used to create the SMSESSION cookie and to challenge the user only for other configured authentication methods. The Session Spec attribute of the SMSESSION cookie that is newly created contains information about all the authentication methods that are executed by SiteMinder and IDSP. For example, consider that a user requests access to a resource that is protected using HTML Forms authentication scheme and OTP in a Multi Factor Authentication Chain. SiteMinder challenges the user for username and password to authenticate the user and IDSP uses OTP to authenticate the user. Now, consider that the user requests access to another resource that is protected by IDSP Scheme where an IDSP policy is configured with the PWD and FIDO obligations. Once the step-up authentication flow is initiated, SiteMinder sends ID Token Hint with amr value indicating that the user was already authenticated using PWD and OTP. IDSP uses this information to avoid challenging the user for PWD as per its configured obligations, and to challenge the user only for FIDO. The Session Spec attribute of the newly created SMSESSION cookie contains PWD, OTP, and FIDO as the authentication methods.
A user with an existing session in SiteMinder accesses an MFA-protected resource that requires step-up authentication.	SiteMinder performs the primary authentication and sends ID Token Hint with amr value set to the authentication method that is used during step-up authentication. The authentication method that was used to create the previous user session is not sent in the amr value. Due to this behavior, amr propagation is not supported in this use case. For example, consider that a user requests access to a resource that is protected using HTML Forms authentication scheme. SiteMinder challenges the user for username and password to authenticate the user and uses this information to create SMSESSION cookie. Now, consider that the user requests access to another resource that is protected by Multi Factor Authentication Chain where the primary authentication is performed by X509 and secondary authentication by PWD and OTP obligations. Once the step-up authentication flow is initiated, SiteMinder sends ID Token Hint with amr value indicating that the user is authenticated by X509 and does not include information about the previous PWD authentication.

amr Values Propagated in ID Token Hint

The following table describes the value of amr depending on the authentication method:

Authentication Method	amr Value
FORM	PWD
X509	POP
IDSP	List of values passed from IDSP

MFA	Primary authentication scheme + List of values passed from IDSP
IWA	WIA
Kerberos	WIA
X509 or FORM	If X509 is successful - POP If FORM is successful - PWD
IWA fallback to FORM	If IWA is successful - WIA If FORM is successful - PWD
Kerberos fallback to FORM	If Kerberos is successful - WIA If FORM is successful - PWD
RADIUS	PWD

Configure Identity Propagation to Support User Enrollment

To auto-enroll a user in IDSP, perform the following configurations in IDSP and SiteMinder:

Configuration in IDSP

To support the feature, the configuration in IDSP involves configuring the authentication policy and the application entity.

Configuring Authentication Policy

The authentication policy includes the **idpName** condition for zeroFootPrint. This condition indicates the name of the logical identity namespace in a policy condition. This condition is optional, and can be added to the policy by using a POST/PATCH operation.

Important! The integration does not support the identity propagation feature for authentication policies that use the group condition.

Configuring Application Entity

A new property is added as a boolean value to the application configuration. To make use of this property, set **zeroFootPrint** to true.

For detailed information about the feature support and configuration in IDSP, see Configuring the Zero Footprint Identity.

Configuration in SiteMinder

1. **For Multi Factor Authentication Chain Template:**
   Select the **Enable Propagation of Extended User Attributes in ID Token Hint** option while configuring the integration authentication scheme using the Multi Factor Authentication Chain Template authentication scheme type. For information about configuring the authentication scheme, see Configure the Integration of SiteMinder with IDSP.
2. Configure phone number, email address, universal ID, and other required attributes in the associated user directory as follows:

User Claim	Attribute Name in User Directory	Attribute Type
user_universalid	Universal ID	User attribute
email	Email	User attribute
phone_number	phone_number	Attribute mapping parameter
given_name	given_name	Attribute mapping parameter
middle_name	middle_name	Attribute mapping parameter
family_name	family_name	Attribute mapping parameter

For information about configuring a user directory, see User Attribute Mapping. If any attributes such as phone number or email are not configured, these values are not sent in the ID Token Hint.

Step-up Authentication Configuration

To support identity propagation in full delegation where IDSP performs delegated authentication, perform the following steps:

1. Configure the **OnAccessInadequatePrivilege** event that is added in the rules configuration for authorization events. Add this along with a response attribute of the WebAgent-OnReject-Redirect type as follows:
   - **Library Name**: 'smauthextattrresponse'
   - **Function Name**: "prepareStateRedirectUrl"
   - **Param**:
     (**Signed ID Token Hint**): "**user_authlookup**=*user_attribute*|*virtual_attribute*|SM_UNIVID;**signing_key_alias**=*value*;**signingAlgorithm**=RS256|RS384|RS512;**zfp_enabled**=true|false;"
     (**Encrypted and signed ID Token Hint**): "**user_authlookup**=*user_attribute*|*virtual_attribute*|SM_UNIVID;**signing_key_alias**=*value*;**signingAlgorithm**=RS256|RS384|RS512;**zfp_enabled**=true|false;**encryption_key_alias**=*value*;**encryptionAlgorithm**=RSA-OAEP|RSA-OAEP-256;**encryptionMethod**=A128CBC-HS256|A192CBC-HS384|A128GCM|A192GCM|A256GCM;"
     Example:
     "user_authlookup=cn;signing_key_alias=smahcert;signingAlgorithm=RS256;zfp_enabled=false;encryption_key_alias=smah
     OAEP;encryptionMethod=A128GCM;"
     (From 2022.Jun.01 build) To send a custom claim in ID Token Hint, specify the claim name in the claim_names parameter. You can specify multiple custom claims that are separated by a space. The claims lookup must be configured in the Attribute Mapping List section of the corresponding user directory to retrieve the claim values.
     Example: claim_names=address mobile_number "user_authlookup=user_attribute|virtual_attribute|SM_UNIVID;signing_key_alias=value;signingAlgorithm=RS256|RS384|RS512;zfp_enabled=true|false;claim_names=address mobile_number;"

   Note: All these values are case-sensitive.

   Ensure that this rule is attached to the policy that is configured with the realm, which is used for this integration. For information about configuring rules, see Rule Dialog.
2. Ensure that the **Process Authorization Events** option is selected while configuring the realm or application. For information about the option, see Realm Dialog Reference or Modify Component Dialog.

AMR Integrity and Continuity

AMR Integrity

From 2022.Jun.01 build, a step-up authentication using the Multi Factor Authentication authentication scheme has the following behavior:

- If the primary authentication scheme is same as the authentication scheme with which a user is already authenticated, the user is not challenged with the primary authentication irrespective of the difference in the authentication level. The user is directly challenged with the secondary authentication.

  Example: Assume that a user is authenticated using the HTML Form authentication scheme when a protected resource (with authentication level 5) is accessed for the first time. Now, the user wants to access another protected resource (with authentication level 10) that is configured with the Multi Factor Authentication authentication scheme, where HTLM Form is the primary authentication and OTP is the secondary authentication. During the step-up authentication, the user is not challenged with HTML Form but is directly challenged for OTP.
- If the step-up authentication uses the same set of primary and secondary authentications with which a user is already authenticated, the user is given direct access to the required resource without another challenge during the step-up authentication.

Example: Assume that a user is authenticated using the Multi Factor Authentication authentication scheme, where Kerberos is the primary authentication and SMS is the secondary authentication, when a protected resource (with authentication level 5) is accessed for the first time. If the user accesses another protected resource (with authentication level 10) that is configured with another Multi Factor Authentication authentication scheme, where Kerberos is the primary authentication and SMS is the secondary authentication, the user is given access to the protected resource without any further challenges during the step-up authentication.

Important!

- The **FCCCompatMode** ACO parameter must be disabled to use this feature.
- This feature is not supported by RADIUS authentication scheme. When RADIUS is configured as the primary authentication in the Multi Factor Authentication authentication scheme, a user is always challenged with the primary authentication during a step-up authentication.

AMR Continuity

In the previous build, the value of the AMR claim in an ID Token contains only the AMR that is a result of a step-up authentication. From this build, the value of the AMR claim contains the AMR of the previous SMSESSION too along with the AMR that is a result of the step-up authentication.

Example: Assume that the AMR of an existing SMSESSION is PWD and the new values after a step-up authentication are POP and OTP. The final AMR value in the ID Token is PWD POP OTP.

Additional Examples for AMR Integrity and Continuity

Assume that a user accesses a resource as a new user and then wants to access two other resources. The following table lists how the step-up authentication works when the Multi Factor Authentication authentication scheme is used and how the AMR value is populated in the AMR claim in an ID Token:

Resource 1 with Authentication Level = 5	Resource 2 with Authentication Level = 10	Resource 3 with Authentication Level = 15	Flow	AMR Value in ID Token
X509 Client Certificate	HTML Form	X509 Client Certificate and HTML Form + OTP	At Resource 3: • Skips the challenge for X509 Client Certificate and HTML Form. • Challenges user for OTP.	PWD POP OTP
HTML Form	X509 Client Certificate	HTML Form + PUSH	At Resource 3: • Skips the challenge for X509 Client Certificate and HTML Form. • Challenges user for PUSH.	PWD POP PUSH
Symantec Identity Security Platform using OTP	X509 Client Certificate	HTML Form + OTP	At Resource 3: • Challenges for HTML Form. • Skips the challenge for OTP.	PWD POP OTP

IDSP using OTP	X509 Client Certificate and HTML Form	X509 Client Certificate and HTML Form + OTP	At Resource 3: • Skips the challenge for X509 Client Certificate and HTML Form. • Skips the challenge for OTP.	PWD POP OTP

Track the Session Activity of a User in the Integration of SiteMinder with IDSP

In the integration, the activity of a user can be tracked using transaction IDs that are generated by SiteMinder and Symantec Identity Security Platform.

A transaction ID of SiteMinder can be used to track the activity of a user in IDSP. Similarly, a transaction ID of IDSP can be used to track the activity of a user in IDSP and in SiteMinder too. Additionally, SiteMinder uses a common session ID in both primary and secondary authentication schemes to ensure that a common session ID is used for all requests until a session expires.

The following use cases describe how a user activity can be tracked in the integration when a Multi Factor Authentication Chain Template authentication scheme is used, where SiteMinder performs the primary authentication and IDSP performs the secondary authentication:

- Scenario 1: How SiteMinder and IDSP Generate and Use Transaction IDs in the End to End Integration Flow
- Scenario 2: How to Use SiteMinder Session ID to Track User Activity in the SiteMinder and IDSP Logs
- Scenario 3: How to Use Session ID of SiteMinder to Track User Activity in the IDSP Logs
- Scenario 4: How to Use Transaction ID of IDSP to Track User Activity in the SiteMinder Logs

Scenario 1: How SiteMinder and IDSP Generate and Use Transaction IDs in the End to End Integration Flow

The following process flow describes how SiteMinder and IDSP generate transaction IDs in the end to end integration flow:

1. Agent generates a transaction ID when a user logs into SiteMinder to perform the primary authentication and stores it in the **TransactionID** field in the agent trace log file.
2. Policy Server completes the primary authentication, generates an ID Token Hint, and logs the transaction ID for these actions in the **TransactionId** field in the Policy Server trace log file, smtracedefault. The corresponding AuthChallenge event in the audit log now logs this transaction ID in the **TransactionId** field in a text log or in the **sm_transactionid** column in an ODBC log.

   Example: SiteMinder transaction ID that is logged in the **sm_transactionid** column in an ODBC audit log

	sm_categoryid	sm_eventid	sm_sessionid	sm_username	sm_res...	sm_reason	sm_transactionid
1	1	4	L06t8YF86e6HKHnc7dMMLbGD1yw=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	56	17d7a71a-d60075ee-eaa6accc-fde3faca-00cf7376-5b
2	1	2	L06t8YF86e6HKHnc7dMMLbGD1yw=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	20	0834267e-7338-4e2d-ab4a-2a46e71b8afe_17d7a71a-d60075ee-eaa6accc-fde3faca-00cf7376-5b
3	1	4	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	56	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8
4	1	1	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	5cfa0b7e-606f-498a-9339-1fd672c9d353_3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8
5	1	11	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	11868949-6cee44ee-3f1cfaf6-bbd3d592-2231ecf0-2f
6	2	5	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	11868949-6cee44ee-3f1cfaf6-bbd3d592-2231ecf0-2f

3. SiteMinder sends its transaction ID to IDSP in the **X-CLIENT-TRANSACTION-ID** query parameter within the request to the Authorization Endpoint of IDSP, to initiate the secondary authentication flow.
4. IDSP logs the received transaction ID of SiteMinder in **clientTxnId** field, completes the obligation and generates a transaction ID, sends an authorization code along with its transaction ID in the **X-TRANSACTION-ID** query parameter in the response to SiteMinder.

   Example: SiteMinder transaction ID that is logged in the clientTxnId field in IDSP

@timestamp	clientTxnId	msg	txnId
Jul 23, 2024 @ 11:17:47.523	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8	Enter in processTokenRequestWithCode	28dfc638-106a-4200-8615-9dee1c575a5f
Jul 23, 2024 @ 11:17:47.523	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8	Generate AT for az code	28dfc638-106a-4200-8615-9dee1c575a5f
Jul 23, 2024 @ 11:17:47.523	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8	grant type: authorization_code, client type: TRUSTED	28dfc638-106a-4200-8615-9dee1c575a5f
Jul 23, 2024 @ 11:17:47.522	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8	Basic validation of a token request is successful	28dfc638-106a-4200-8615-9dee1c575a5f

5. SiteMinder includes its transaction ID in the **X-CLIENT-TRANSACTION-ID** header within the request to the Token Endpoint of IDSP.
6. IDSP logs the received transaction ID of SiteMinder again in **clientTxnId** and generates an ID Token.
7. Access Gateway combines the transaction IDs of IDSP and SiteMinder in the ***<authentication-hub-transaction-id>_<siteminder-transaction-id>*** format and sends the combined transaction ID to Policy Server as part of the secondary authentication login call.
8. Policy Server and Agent store the combined transaction ID in the smtracedefault and FWSTrace log files. Also, the corresponding AuthAccept or AuthReject event in the audit log now logs the combined transaction ID in the audit log file in **TransactionId/sm_transactionID**.

   Example: Combined transaction IDs of IDSP and SiteMinder

	sm_categoryid	sm_eventid	sm_sessionid	sm_username	sm_res...	sm_reason	sm_transactionid
1	1	4	L0t8YF86e6HKHnc7dMMLbGD1yw=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	56	17d7a71a-d60075ee-eaa6accc-fde3faca-00cf7376-5b
2	1	2	L0t8YF86e6HKHnc7dMMLbGD1yw=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	20	0834267e-7338-4e2d-ab4a-2a46e71b8afe_17d7a71a-d60075ee-eaa6accc-fde3faca-00cf7376-5b
3	1	4	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	56	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8
4	1	1	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	5cfa0b7e-606f-498a-9339-1fd672c9d353_3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8
5	1	11	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	11868949-6cee44ee-3f1cfaf6-bbd3d592-2231ecf0-2f
6	2	5	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	11868949-6cee44ee-3f1cfaf6-bbd3d592-2231ecf0-2f

Scenario 2: How to Use SiteMinder Session ID to Track User Activity in the SiteMinder and IDSP Logs

This scenario describes how a session ID of SiteMinder can be used to track a user in the logs of SiteMinder and IDSP after the primary authentication:

1. Understand how SiteMinder and IDSP generate transaction IDs in the end to end integration flow.
2. Determine the session ID that is related to the required user using the audit log of SiteMinder.
3. Use the determined session ID to retrieve the corresponding transaction ID of SiteMinder from the audit log. Use this transaction ID to track the user activity in SiteMinder.

   Example: SiteMinder transaction ID that is logged in the **sm_transactionid** column in an ODBC audit log

	sm_categoryid	sm_eventid	sm_sessionid	sm_username	sm_res...	sm_reason	sm_transactionid
1	1	4	L0t8YF86e6HKHnc7dMMLbGD1yw=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	56	17d7a71a-d60075ee-eaa6accc-fde3faca-00cf7376-5b
2	1	2	L0t8YF86e6HKHnc7dMMLbGD1yw=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	20	0834267e-7338-4e2d-ab4a-2a46e71b8afe_17d7a71a-d60075ee-eaa6accc-fde3faca-00cf7376-5b
3	1	4	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	56	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8
4	1	1	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	5cfa0b7e-606f-498a-9339-1fd672c9d353_3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8
5	1	11	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	11868949-6cee44ee-3f1cfaf6-bbd3d592-2231ecf0-2f
6	2	5	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	11868949-6cee44ee-3f1cfaf6-bbd3d592-2231ecf0-2f

4. To track the user activity in the IDSP logs, use its **clientTxnId** field in which the transaction ID of SiteMinder is logged when SiteMinder sends this value in the **X-CLIENT-TRANSACTION-ID** query parameter in the Authorization Endpoint request to IDSP during the primary authentication.

   Example: SiteMinder transaction ID that is logged in the clientTxnId field in IDSP

@timestamp	clientTxnId	msg	txnId
Jul 23, 2024 @ 11:17:47.523	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8	Enter in processTokenRequestWithCode	28dfc638-106a-4200-8615-9dee1c575a5f
Jul 23, 2024 @ 11:17:47.523	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8	Generate AT for az code	28dfc638-106a-4200-8615-9dee1c575a5f
Jul 23, 2024 @ 11:17:47.523	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8	grant type: authorization_code, client type: TRUSTED	28dfc638-106a-4200-8615-9dee1c575a5f
Jul 23, 2024 @ 11:17:47.522	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8	Basic validation of a token request is successful	28dfc638-106a-4200-8615-9dee1c575a5f

Scenario 3: How to Use Session ID of SiteMinder to Track User Activity in the IDSP Logs

This scenario describes how a session ID of SiteMinder can be used to retrieve the transaction ID of IDSP from the SiteMinder audit log after a complete authentication flow and how to use this transaction ID to track a user in the IDSP logs:

1. Understand how SiteMinder and IDSP generate transaction IDs in the end to end integration flow.
2. Determine the session ID that is related to the required user using the audit log of SiteMinder.

   Example: SiteMinder transaction ID that is logged in the **sm_transactionid** column in an ODBC audit log

	sm_categoryid	sm_eventid	sm_sessionid	sm_username	sm_res...	sm_reason	sm_transactionid
1	1	4	L0t8YF86e6HKHnc7dMMLbGD1yw=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	56	17d7a71a-d60075ee-eaa6accc-fde3faca-00cf7376-5b
2	1	2	L0t8YF86e6HKHnc7dMMLbGD1yw=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	20	0834267e-7338-4e2d-ab4a-2a46e71b8afe_17d7a71a-d60075ee-eaa6accc-fde3faca-00cf7376-5b
3	1	4	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	56	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8
4	1	1	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	5cfa0b7e-606f-498a-9339-1fd672c9d353_3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8
5	1	11	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	11868949-6cee44ee-3f1cfaf6-bbd3d592-2231ecf0-2f
6	2	5	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	11868949-6cee44ee-3f1cfaf6-bbd3d592-2231ecf0-2f

3. Use the determined session ID to retrieve the corresponding transaction ID of IDSP from the audit log. In the combined transaction ID value that is logged in **TransactionId/sm_transactionID**, the value before the underscore (_) corresponds to the transaction ID of IDSP and it matches the value that is logged in **txnId** in IDSP.

   Example: Combined transaction IDs of IDSP and SiteMinder

	sm_categoryid	sm_eventid	sm_sessionid	sm_username	sm_res...	sm_reason	sm_transactionid
1	1	4	L0t8YF86e6HKHnc7dMMLbGD1yw=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	56	17d7a71a-d60075ee-eaa6accc-fde3faca-00cf7376-5b
2	1	2	L0t8YF86e6HKHnc7dMMLbGD1yw=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	20	0834267e-7338-4e2d-ab4a-2a46e71b8afe_17d7a71a-d60075ee-eaa6accc-fde3faca-00cf7376-5b
3	1	4	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	56	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8
4	1	1	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	5cfa0b7e-606f-498a-9339-1fd672c9d353_3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8
5	1	11	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	11868949-6cee44ee-3f1cfaf6-bbd3d592-2231ecf0-2f
6	2	5	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	11868949-6cee44ee-3f1cfaf6-bbd3d592-2231ecf0-2f

4. Use the retrieved transaction ID of IDSP to track the user activity in the IDSP logs.

   Example: IDSP transaction ID is logged in its **txnId** field

@timestamp	clientTxnId	msg	txnId
Jul 23, 2024 @ 11:17:47.297	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8	response_type is code	5cfa0b7e-606f-498a-9339-1fd672c9d353
Jul 23, 2024 @ 11:17:47.297	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8	authComplete: device signature timestamp from signin app: 1721713667 secs, offset: 0	5cfa0b7e-606f-498a-9339-1fd672c9d353
Jul 23, 2024 @ 11:17:47.277	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8	Auth complete request for UserId: e6e6f6d8-6f12-49df-926a-cccd0bfc9889, UserLoginId: user1, identitySourceId: UD, identitySourceName: UD, …	5cfa0b7e-606f-498a-9339-1fd672c9d353

Scenario 4: How to Use Transaction ID of IDSP to Track User Activity in the SiteMinder Logs

This scenario describes how a transaction ID of IDSP can be retrieved from IDSP after a complete authentication flow and how to use this transaction ID to track a user in the SiteMinder logs:

1. Understand how SiteMinder and IDSP generate transaction IDs in the end to end integration flow.
2. Determine the transaction ID of the user from the **txnID** field in IDSP.

   Example: IDSP transaction ID is logged in its **txnId** field

@timestamp	clientTxnId	msg	txnId
Jul 23, 2024 @ 11:17:47.297	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8	response_type is code	5cfa0b7e-606f-498a-9339-1fd672c9d353
Jul 23, 2024 @ 11:17:47.297	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8	authComplete: device signature timestamp from signin app: 1721713667 secs, offset: 0	5cfa0b7e-606f-498a-9339-1fd672c9d353
Jul 23, 2024 @ 11:17:47.277	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8	Auth complete request for UserId: e6e6f6d8-6f12-49df-926a-cccd0bfc9889, UserLoginId: user1, identitySourceId: UD, identitySourceName: UD, …	5cfa0b7e-606f-498a-9339-1fd672c9d353

3. Use the determined transaction ID of IDSP to retrieve the corresponding transaction ID of SiteMinder from the audit log of SiteMinder. In the combined transaction ID value that is logged in **TransactionId/sm_transactionID**, the value after the underscore (_) corresponds to the transaction ID of SiteMinder.

   Example: Combined transaction IDs of IDSP and SiteMinder

	sm_categoryid	sm_eventid	sm_sessionid	sm_username	sm_res...	sm_reason	sm_transactionid
1	1	4	L06t8YF86e6HKHnc7dMMLbGD1yw=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	56	17d7a71a-d60075ee-eaa6accc-fde3faca-00cf7376-5b
2	1	2	L06t8YF86e6HKHnc7dMMLbGD1yw=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	20	0834267e-7338-4e2d-ab4a-2a46e71b8afe_17d7a71a-d60075ee-eaa6accc-fde3faca-00cf7376-5b
3	1	4	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	56	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8
4	1	1	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	5cfa0b7e-606f-498a-9339-1fd672c9d353_3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8
5	1	11	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	11868949-6cee44ee-3f1cfaf6-bbd3d592-2231ecf0-2f
6	2	5	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	11868949-6cee44ee-3f1cfaf6-bbd3d592-2231ecf0-2f

4. Use the retrieved transaction ID of SiteMinder to track the user activity in the audit, smtracedefault, and FWStrace logs of SiteMinder.

   Example: SiteMinder transaction ID that is logged in the **sm_transactionid** column in an ODBC audit log

	sm_categoryid	sm_eventid	sm_sessionid	sm_username	sm_res...	sm_reason	sm_transactionid
1	1	4	L06t8YF86e6HKHnc7dMMLbGD1yw=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	56	17d7a71a-d60075ee-eaa6accc-fde3faca-00cf7376-5b
2	1	2	L06t8YF86e6HKHnc7dMMLbGD1yw=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	20	0834267e-7338-4e2d-ab4a-2a46e71b8afe_17d7a71a-d60075ee-eaa6accc-fde3faca-00cf7376-5b
3	1	4	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	56	3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8
4	1	1	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	5cfa0b7e-606f-498a-9339-1fd672c9d353_3bfd1dc9-9f385241-11f404f9-c5d9a727-f60d6bad-a8
5	1	11	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	11868949-6cee44ee-3f1cfaf6-bbd3d592-2231ecf0-2f
6	2	5	AFI3T/EP/H9dofYtGI+3wojwXkg=	cn=user1,ou=OrgUnit0,o=security.com	/sa/	0	11868949-6cee44ee-3f1cfaf6-bbd3d592-2231ecf0-2f

Configure Re-authentication for Sensitive Resources Protected by the Multi Factor Authentication or IDSP Authentication Schemes

SiteMinder defines sensitive resources as those resources that always require users to re-authenticate before they access. The transfer money section of an online banking website is one possible example of a sensitive resource. If the transfer-money section is defined as a sensitive resource, users must provide their credentials before accessing that section though they were authenticated during login. This sensitive resource protection prevents an unauthorized user from transferring money in case a user stepped away from the system without logging out first. It also prevents unauthorized users from accessing the transfer-money section without being challenged to confirm their identity.

You can define the resources that are protected by the Multi Factor Authentication or Symantec Identity Security Platform authentication schemes as sensitive resources.

Perform the following steps to configure re-authentication for these sensitive resources:

Review the Prerequisites

Complete the following steps before you proceed to the configuration of re-authentication for sensitive resources:

1. Install and configure Policy Server. Define a policy domain with a user directory connection, realm, rule, and policy.
2. Configure an agent with an authentication scheme.
3. Configure a fully operating session store.
4. Authenticate a user before sending a request to access the sensitive resource.

Disable the FCCCompatMode ACO Parameter

Disable the **FCCCompatMode** ACO parameter for a better security when using certain features.

Note: This procedure assumes you configure your agents centrally, using an agent configuration object on a Policy Server. If you use the local agent configuration method instead, disable the **FCCCompatMode** parameter in your LocalConfig.conf file.

Follow these steps:

1. From the Administrative UI, click Infrastructure, Agent Configuration Objects.
2. Click the edit icon next to the agent configuration object that you want.
3. Locate the FCCCompatMode parameter.
4. Verify that the value is set to no. If the value is yes, continue with the following steps:
   a. Click the edit icon to the left of the parameter.
   b. Highlight the Value field, and then type no.
   c. Click OK.
5. Click Submit.
   The FCCCompatMode parameter is disabled and a confirmation message appears.

Modify Policy to Protect the Sensitive Resource

Modify a policy to protect the sensitive resources. This modification designates some of the resources that the policy protects are considered sensitive.

Follow these steps:

1. From the Administrative UI, click Policies, Domain, Domains.
2. Click the edit icon that corresponds to the domain containing the sensitive resources.
3. Click the Policy tab.
   The Modify Policy: screen is displayed with the General tab selected.
4. Select the Validate Identity check box.
5. Click OK.
6. Click Submit.
7. Continue with the next stop of creating a GET/POST rule in the policy.

Protect the URL of Sensitive Resource

Protect the URL of the sensitive resource with a GET/POST rule to indicate that the URL protected by the policy is considered sensitive.

Follow these steps:

1. From the Administrative UI, click Policies, Domain, Domain Policies.
2. Click the edit icon that corresponds to the policy containing the URL of sensitive resource.
3. Click the Rules tab.

4. Click Add Rule.
5. Click Create in the Available Rules dialog.
6. Select the option button for the realm under which you want to create the rule, and then click Next.
7. Create the GET/POST rule with the following steps:
   a. Enter a name and an optional description.
   b. Click the Resource field and enter the URL of the sensitive resource.
      The following example defines an HTML page named transfer_funds as the sensitive resource:
      transfer_funds.html
   c. Verify that the Web Agent actions option button is selected.
   d. Click the following items under the Action list:
      - Get
      - Post
   e. (Optional) Define any time restrictions that you want.
   f. Click Finish.

   The Available rules dialog is displayed. The new GET/POST rule appears in the list.
8. Click OK.
9. Click Submit.
   The rule is added to the policy, and a confirmation message appears.

Create an OnAccessValidateIdentity Rule

Create an OnAccessValidateIdentity rule in the policy. This rule rejects the current credentials of the user which started the session, and forces the user to re-authenticate before accessing the sensitive resource.

Follow these steps:

1. From the Administrative UI, click Policies, Domain, Domain Policies.
2. Click the edit icon that corresponds to the policy containing the URL of sensitive resource.
3. Click the Rules tab.
4. Click Add Rule.
5. Click Create in the Available Rules dialog.
6. Click the option button for the realm under which you want to create the rule, and then click Next.
7. Create the OnAccessValidateIdentity rule with the following steps:
   a. Enter a name and an optional description.
   b. Click the Resource field and enter the URL of the sensitive resource.
   c. The following example defines an HTML page named transfer_funds as the sensitive resource:
      transfer_funds.html
   d. Select the Authorization events option and then select OnAccessValidateIdentity from the Actions drop-down list.
   e. (Optional) Define any time restrictions that you want.
8. Click Finish.
   The Available rules dialog appears. The new OnAccessValidateIdentity rule appears in the list.
9. Click OK.
10. Click Submit.
    The rule is added to the policy, and a confirmation message appears.

(For Multi Factor Authentication-protected resources) Create an Active Response that Redirects to the URL of the Primary Authentication Scheme Credentials Collector

Create a response that redirects users to the URL of the primary authentication scheme credentials collector as defined in the authentication scheme. The OnAccessValidateIdentity rule rejects the credentials that the user provided originally. The response redirects the user to the defined primary authentication scheme.

Follow these steps:

1. From the Administrative UI, click Policies, Domain, Domain Policies.
2. Click the edit icon that corresponds to the policy containing the URL of sensitive resource.
3. Click the Rules tab.
4. Locate the OnAccessValidateIdentity rule, and click the Add Response button next to it.
5. Click Create.
6. Create a response with the following information:
   a. Type a name and an optional description.
   b. Verify that the Agent Type is CA Single Sign-On and Web Agent.
   c. Click Create Response Attribute in the Attribute List section.
   d. Select WebAgent-OnValidate-Redirect from the Attribute drop-down list.
   e. Select the Attribute Kind as Active Respons
   f. In the Attribute Fields section, enter the following information:
      - Library Name: smauthextattrresponse
      - Function Name: propagateAuthURL
      - Parameters: Empty
   g. Click OK.

   The Create Response screen appears with the response attribute from Step 6c.
7. Click OK.
8. Click Submit.

   The response and response attribute are added to the rule, and a confirmation message appears.
9. Restart Policy Server.

(For IDSP-protected resource) Create an Active Response that Redirects to IDSP

Create an active response that redirects to IDSP using the ID Token Hint that is sent by SiteMinder.

Important!

- The AMR value is not included in ID Token Hint.
- ZFP is not supported.

Follow these steps:

1. From the Administrative UI, click Policies, Domain, Domain Policies.
2. Click the edit icon that corresponds to the policy containing the URL of sensitive resource.
3. Click the Rules tab.
4. Locate the OnAccessValidateIdentity rule, and click the Add Response button next to it.
5. Click Create.
6. Create a response with the following information:
   a. Type a name and an optional description.
   b. Verify that the Agent Type is CA Single Sign-On and Web Agent.
   c. Click Create Response Attribute in the Attribute List section.
   d. Select WebAgent-OnValidate-Redirect from the Attribute drop-down list.
   e. Select the Attribute Kind as Active Respons

f. In the Attribute Fields section, enter the following information:
   a. **Library Name**: smauthextattrresponse
   b. **Function Name**: propagateIDOnValidate
   c. **Parameters**: user_authlookup=cn;signing_key_alias=signingcert;signingAlgorithm=RS256
g. Click OK.
   The Create Response screen appears with the response attribute from Step 6c.

7. Click OK.
8. Click Submit.
   The response and response attribute are added to the rule, and a confirmation message appears.Restart the Policy Server.

Additional Configuration to Use SecurID HTML Form for Risk-based Authentication During Re-authentication of a Sensitive Resource

From this release, SecurID HTML Form is configured as the primary authentication scheme in the Multi Factor Authentication authentication scheme.

When SecurID HTML Form is configured as the primary authentication scheme in the Multi Factor Authentication authentication scheme to protect a sensitive resource, perform the following steps to ensure that the authentication scheme works during a re-authentication of the sensitive resource:

1. Open the **am_integration.js** file that is downloaded from RSA server for use in the RBA configuration.
2. Add the following code below the **logonForm.action** line of code:

```
 // add new input containing the siteminder sensitive reauth flag
let url = document.location.href;
if (url.includes("smreauth")) {
var smReauth = document.createElement('input');
smReauth.setAttribute('type', 'hidden');
smReauth.setAttribute('name', 'SMREAUTH');
smReauth.setAttribute('value', '1');
logonForm.appendChild(smReauth);
}
```

3. Save the changes.
4. Restart Access Gateway.

Support for jwt-bearer Token Grant

When SiteMinder acts as an OpenID Connect Provider, only browser-based request flows are supported in which an SMSESSION is mandatory to return Authorization Code. The typical OIDC flow is followed where the Authorization and Token endpoints are involved in generating the required tokens.

SiteMinder OpenID Connect Provider is enhanced to support the jwt-bearer Token grant. This feature enables Token endpoint to exchange an ID Token for an Access Token, and optionally, a Refresh Token. The support for jwt-bearer Token grant has the following benefits:

- Any Client (such as a mobile app) can authenticate itself and can share an ID Token with SiteMinder.
- The number of communication calls between Client and Authorization Server reduces as Authorization endpoint is no more involved in the generation of Access Token.

This grant type allows a client to use an existing trust relationship that is defined through a JWT, without a direct user approval step at SiteMinder.

The following process describes the overall process:

1. Client (such as a mobile app) authenticates using any application, for example, Symantec Identity Security Platform.
2. IDSP authenticates the user, generates ID Token, and sends it to Client.
3. Client invokes the JWT-bearer Token grant to exchange ID Token for an Access Token.
4. SiteMinder uses the ID Token to authenticate the user and generates Access Token and Refresh Token, if the Refresh Token grant type is selected in Client configuration.

Configure SiteMinder to Consume ID Token in jwt-bearer Token Grant Format

Follow these steps:

1. Navigate to **Federation**, **OpenID Connect**, **Client** in SiteMinder Administrative UI. For information about Client configuration, see Clients Dialog.
2. Create or edit a client.
3. Select **JWT Bearer** as the **Grant Type**.
4. (Optional) To generate a Refresh Token, select the **Refresh Token** grant type too.
5. Complete the following fields:
   - **User Lookup**
     Defines the claim that must be used in the received ID Token for the user lookup.
   - **Verification Certificate Alias**
     Specifies the primary public key alias to validate the signature of the received ID Token. This is the signing key certificate of IDSP tenant.
   - **Secondary Verification Certificate Alias**
     Specifies the secondary public key alias to validate the signature of the received ID Token. This value is used if the validation fails using the primary public key alias.
   - **Enable Audience Validation**
     Specifies whether the audience claim in the received ID Token must be validated. Enable this option in a production environment. If this option is enabled, the **secondary audience** attribute must be configured in the application of the Client that invokes the jwt-bearer Token grant in SiteMinder.
   - **Show Audiences**
     Lists the allowed values of audiences. This option is displayed only if **Enable Audience Validation** is selected. If **Show Audiences** is selected, the following list of supported formats of the audience value is displayed:
     - *<token_endpoint_url>*
     - *<token_endpoint_url_without_client_name>*
     - urn:iam:aud:hosturl=*<token_endpoint_url>*
     - urn:iam:aud:hosturl=*<token_endpoint_url_without_client_name>*

     Example: If Token endpoint is `https://example.net/affwebservices/SM/oidc/Voonair/token` , the following audience values are supported:
     - https://example.net/affwebservices/SM/oidc/Voonair/token
     - https://example.net/affwebservices/SM/oidc/token
     - urn:iam:aud:hosturl=https://example.net/affwebservices/SM/oidc/Voonair/token
     - urn:iam:aud:hosturl=https://example.net/affwebservices/SM/oidc/token
   - **Issuer**
     Defines the list of allowed issuers of the IDSP tenant to be used for validating the received ID Token.
     Example:
     ```
     https://myauthhub.example.com/default/
     ```
     where **default** is the name of the tenant of the IDSP instance that is hosted on the **myauthhub.example.com** domain.
   - **Skew Time**

Defines the number of seconds subtracted from the current time to account for the difference in the system clocks of the machines where SiteMinder and Client (such as IDSP) are installed.

6. Complete the required client configuration.
7. Save the changes.

Example: The following is the request payload format of ID Token:

```
POST /affwebservices/SM/oidc/sample_client/token HTTP/1.1
Host: wa.example.com
Content-Type: application/x-www-form-urlencoded


scope=sub&client_id=47e5c4c1-4ff8-4a04-adff-59abc4433725&client_secret=DYYk4wDolt7eh37CEcS%2Bk8eOW
%2FlSkY2oJn3a%2BTFOrRI%3D&grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&assertion=<jwt>
```

Note that **scope** is a mandatory parameter in this request.

Validate Revocation Status of ID Token

SiteMinder can validate the revocation status of a received ID Token. This feature adds an extra layer of security on the OIDC transactions.

To use the feature, ensure that you import the certificate or the root CA of the certificate that is used for enabling SSL in the identity provider that generated the ID Token, into SiteMinder.

The Clients dialog now displays the following new parameters:

Validate Revocation Status of ID Token

Specifies whether the revocation status of a received ID Token must be validated. If this option is selected, the **Use Revocation Endpoint from Selected Trusted OIDC Providers** field is displayed along with the **Available Trusted OIDC Providers** and **Selected Trusted OIDC Providers** fields.

The **Available Trusted OIDC Providers** field lists all the trusted OIDC providers whose Issuer URL matches the list of Issuer URLs that are configured in the **Issuer** field of the **JWT Validation** section in the Client configuration dialog. Move the required providers into the **Selected Trusted OIDC Providers** field. If multiple trusted OIDC providers with the same Issuer URL are listed as available, move *only* one of them into the selected list. If multiple such providers are moved into the selected list, *only* the first one in the sequential order is saved when the Client configuration is saved.

Ensure that the selected trusted OIDC providers are configured with the revocation status check information such as introspection endpoint and client credentials. When an ID Token is received, during runtime, if the issuer field value in the incoming token matches the issuer URL value of a selected trusted OIDC provider, the revocation status check is run on the ID Token using the revocation status check information of the corresponding trusted OIDC provider. If the revocation status check information is not configured in the selected trusted OIDC provider, SiteMinder alerts about the missing configuration while saving the Client configuration.

If a trusted OIDC provider is used by multiple clients, the **aud** or **azp** claim in an ID Token is used to identify the Client ID, which in turn is used to retrieve the client credentials to access the Introspection endpoint of the trusted OIDC provider.

Note: This feature uses only the revocation information that is configured in a trusted OIDC provider. The other configurations within the trusted OIDC provider such as user lookup or signature verification are not used by this feature.

For information about configuring the rest of the fields in a Client, see Clients Dialog.

Validate Claims Using JWT Authentication Scheme

The JWT authentication scheme is enhanced to validate claims. The authentication scheme now contains the **JWT Claim Fields** section with the following fields:

- **User Lookup**
  Defines the claim that must be used in the received token for the user lookup.
  Note: The **User Lookup** field has been moved from the **JWT Token Authentication Attribute for Signature Validation** section to the **JWT Claim Fields** section. The rest of the fields are newly added.
- **Issuers List**
  Specifies the list of allowed issuers of the Symantec Identity Security Platform tenant to be used for validating the received token. SiteMinder validates the allowed issuers list in the received token only if this field is configured. You can enter multiple values that are separated by a comma.
  Example:
  `https://myauthhub.example.com/default/`
  where **default** is the name of the tenant of the IDSP instance that is hosted on the **myauthhub.example.com** domain.
- **Claims Required for Validation**
  Specifies whether Policy Server must validate the selected claims and their values in the received token. If a claim is selected in the **Claims Required for Validation** field, Policy Server validates that the received token contains the selected claim and also validates the value of the claim. If any of the claims are not selected but the received token contains them, Policy Server does not validate their presence in the received token, however, it validates the values of the claims.
- **Skew Time (Seconds)**
  Define the number of seconds subtracted from the current time to account for the difference in the system clocks of the machines where SiteMinder and IDSP are installed. This value is used to validate time-based fields such as iat, exp, and nbf.
- **Audiences List**
  Specifies the allowed values of audiences. SiteMinder validates the allowed audiences list in the received token only if this field is configured. You can enter multiple values in any format, and separate them by a comma.
  Example: If Token Endpoint is `https://example.net/affwebservices/SM/oidc/Voonair/token`, the following audience values can be specified:
  - https://example.net/affwebservices/SM/oidc/Voonair/token
  - https://example.net/affwebservices/SM/oidc/token
  - urn:iam:aud:hosturl=https://example.net/affwebservices/SM/oidc/Voonair/token
  - urn:iam:aud:hosturl=https://example.net/affwebservices/SM/oidc/token

For information about the configuration of the rest of the fields, see JWT Authentication Scheme.

Use ID Token Hint to Generate an SMSESSION

The Authorization Endpoint of SiteMinder OpenID Connect Provider can generate an SMSESSION for an unauthenticated mobile app user of SiteMinder, who requests an access to a SiteMinder-protected resource using a web browser. Instead of redirecting the mobile user to a protected authentication URL for an SMSESSION, the Authorization Endpoint allows the mobile app user to exchange an ID Token that is generated by any trusted OIDC provider for an SMSESSION and authorization code. This feature enables single sign-on for a user to seamlessly navigate between mobile and web apps to access SiteMinder-protected resources.

SiteMinder can now consume an ID Token as an ID Token Hint. In this feature context, a generated ID Token is referred to as a received ID Token Hint or ID Token Hint. The ID Token can be generated by SiteMinder itself or any trusted OIDC provider. SiteMinder uses the configuration of a trusted OIDC provider and a received ID Token Hint to generate an SMSESSION.

Configure SiteMinder to Use an ID Token Hint to Generate an SMSESSION

To implement this feature, perform the following steps in SiteMinder:

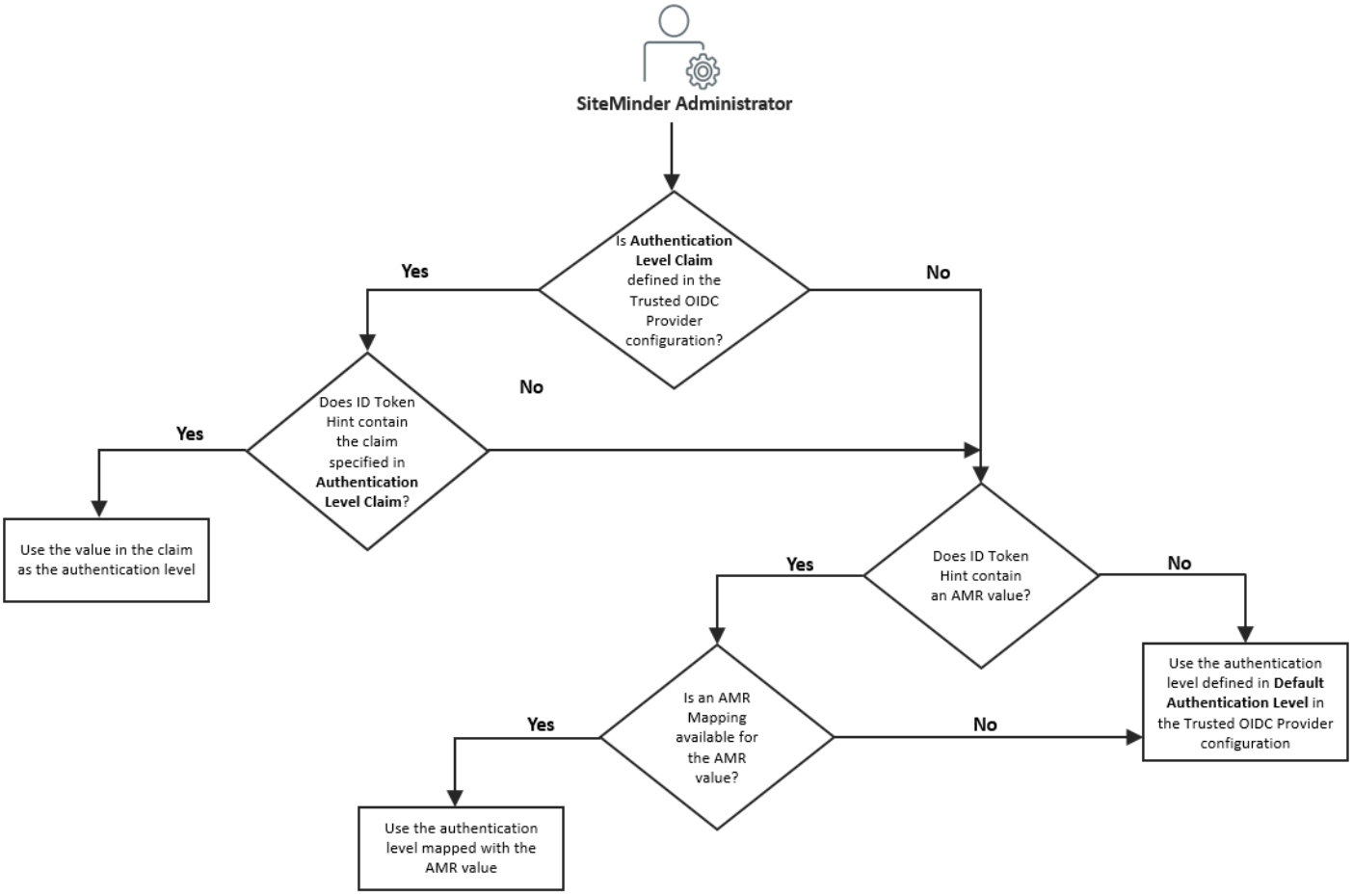
1. Create a trusted OIDC Provider.
2. Create an ID Token policy.
3. Configure the Authorization Provider that is associated with a Client, to define the user directory that must be used for authenticating the user in an ID Token Hint and creating an SMSESSION.
4. Configure an OIDC Client to use the created ID Token policy.
5. Send an authentication request to the Authorization Endpoint.

Create a Trusted OIDC Provider

Provide the configuration of the OIDC provider that generates an ID Token, be it from SiteMinder too, as a trusted OIDC provider in SiteMinder. SiteMinder considers the ID Token as an ID Token Hint in this feature. This configuration is used for validating the received ID Token Hint, and for exchanging an SMSESSION and authorization code.

The following diagram describes how SiteMinder determines the authentication level that must be used in the new SMSESSION when it receives an ID Token Hint:

Figure 36: Determine the Authentication Level for the New Session



The following steps describe how SiteMinder determines the authentication level of a new session when it receives an ID Token Hint:

1. SiteMinder verifies whether the **Authentication Level Claim** field in the trusted OIDC provider configuration specifies a claim name that contains an authentication level.

- If the field is defined, verifies whether the received ID Token Hint contains the claim name that is specified in the field. If the claim name is present in the received ID Token Hint, the authentication level value that is specified in the claim name is used for the new session.
- If the field is defined but the specified claim name is not present in the received ID Token Hint, performs the next check as described in Step 2.

2. SiteMinder verifies whether the received ID Token Hint contains an AMR value.

- If an AMR value is present, verifies whether an AMR Mapping exists for the AMR value. If the AMR Mapping for the AMR value is available, the authentication level that is defined for the AMR value is used for the new session.
- If an AMR value is not present, the authentication level that is defined in the **Default Authentication Level** field in the trusted OIDC provider configuration is used for the new session.

Follow these steps:

1. Open Administrative UI.
2. Navigate to **Federation**, **OpenID Connect**, **Trusted OIDC Provider**.
3. Click **Create Trusted OIDC Provider**.
4. Complete the following fields:
   - **Name**
     Defines a unique name to identify the OIDC provider.
   - **Description**
     Defines a brief description of the OIDC provider.
   - **Issuer**
     Defines the issuer URL of the OIDC provider that must be used for validating the received ID Token Hint. This value must match the **issuer** value provided by the Well-Known Endpoint of the OIDC provider.
   - **User Lookup**
     Defines the claim to be used in the received ID Token Hint for the user authentication.
     Default Value: sub
   - **Authentication Level Claim**
     Defines the claim in the received ID Token Hint that specifies the dynamic authentication level to be used in the SiteMinder session.
     Default Value: user_authlevel
   - **Default Authentication Level**
     Defines the authentication level to be used in the SiteMinder session. This value is used in the following scenarios, in a sequential check, as described in the diagram in this section:
     a. If the claim is not defined or if the received ID Token Hint does not contain a claim with the authentication level, though the claim name is specified in **Authentication Level Claim**.
     b. If the received ID Token Hint does not contain AMR information or if an AMR mapping does not exist for the AMR value.

     If multiple AMR values are present, the AMR value with the highest authentication level is considered.
     Default Value: 5
   - **Enable Audience Validation**
     Specifies to validate the audience claim in the received ID Token Hint. The audience value must be in one of the following formats:
     - <*authorization-server-base-url*>/affwebservices/CASSO/oidc/<*client-name*>/authorize
     - <*authorization-server-base-url*>/affwebservices/CASSO/oidc/authorize
     - urn:iam:aud:hosturl=<*authorization-server-base-url*>/affwebservices/CASSO/oidc/<*client-name*>/authorize
     - urn:iam:aud:hosturl=<*authorization-server-base-url*>/affwebservices/CASSO/oidc/authorize
   - **Skew Time**

Defines the time period to be subtracted from the current time to account for the difference in the system clocks of the SiteMinder and OIDC provider machines.

- **Verification Certificate Alias**
  Specifies the public certificate alias to validate the signature of a received ID Token Hint.
- **Secondary Verification Certificate Alias**
  Specifies the secondary public certificate alias to validate the signature of a received ID Token Hint.
- **Revocation Status Check**
  Specifies to validate the revocation status of the received ID Token Hint. If this option is selected, the following additional fields are displayed for configuration:
  - **Introspection Endpoint**
    Defines the URL of the Introspection Endpoint to be used for validating the revocation status of the received ID Token Hint.
    For example, if the ID Token Hint is from Symantec Identity Security Platform, enter the endpoint in the `<issue-url>/oauth2/v1/introspect` format.
  - **Client ID and Client Secret**
    Defines the client credentials to be used to validate the defined Introspection Endpoint. Multiple client credentials can be specified as the OIDC provider can be used by multiple clients. If an OIDC provider is used by multiple clients, the **aud** or **azp** claim in an ID Token is used to identify the Client ID, which in turn is used to retrieve the client credentials to access the Introspection Endpoint of the OIDC provider.
  - **Show Client Secret**
    Specifies whether the entered client secret must be displayed in plain text.

5. Click **Create** to save the changes.

Create an ID Token Policy

Create an ID Token policy to establish trust between the defined trusted OIDC provider and an OIDC Client. The policy can be associated with multiple OIDC Clients. An OIDC Client dynamically invokes the policy during the runtime, based on the **issuer** claim in an ID Token Hint, to find the OIDC provider that can validate an ID Token.

Follow these steps:

1. Open Administrative UI
2. Navigate to **Federation**, **OpenID Connect**, **ID Token Policy**.
3. Click **Create ID Token Policy**.
4. Complete the following fields:
   - **Name**
     Defines a unique name to identify the policy.
   - **Description**
     Defines a brief description of the policy.
   - **Maximum ID Token Life**
     Defines the maximum time period between the issuance and consumption of the ID Token Hint. If a value is not entered, the expiry time of the ID Token Hint is considered.
   - **JWE Required**
     Specifies to mandate that the received ID Token Hint must be signed and encrypted. Ensure that the private key of SiteMinder is selected in the corresponding authorization provider configuration.
   - **Ignore Existing Session**
     Specifies whether an existing SMSESSION, if sent along with an ID Token Hint, must be ignored or allowed. If this option is selected, the existing SMSESSION is ignored and a new SMSESSION is generated for the user. If this option is not selected, the SMSESSION is validated along with the ID Token to ensure that they both belong to the same user.
   - **(From 12.8.0700.2989 Build) Support Step-up Authentication**

Specifies how SiteMinder behaves when the authentication level in an ID Token Hint is lower than the **Minimum Authentication Level** value that is defined in the associated authorization provider. If this field is selected, a step-up authentication is performed. If this field is not selected, user authentication fails.

- **Trusted OIDC Providers Filter**
  Specifies the list of OIDC providers to be filtered and used by OIDC Clients at runtime. For example, if five OIDC providers are filtered at runtime, then an OIDC Client uses the **issuer** claim in the ID Token Hint to verify which OIDC provider in the filtered list had generated the provided ID Token Hint. You can create a search filter expression using rules and rule groups that include the AND/OR logical operators. Nested rules and rule groups are also supported.
- **Maximum Timeout**
  Defines the maximum time period that a user session can be active before the user is re-authenticated.
- **Idle Timeout**
  Defines the time period that an authorized user session can remain inactive before the current session is terminated.
- **Persistent Session**
  Specifies that the session is persistent. User sessions are tracked in the session store and cookies. If this option is selected, Idle Timeout must be selected. Optionally, specify a validation period in the Validation Period field that is displayed only if this option is selected.
- **Validation Period**
  Defines the time period that the result of a session validation call to Policy Server is cached.
- **Synchronous Auditing**
  Specifies that a user is given access to a resource only if the auditing of all the transactions is complete.
  Note: Session assurance is not supported.

5. Click **Create** to save the changes.

Configure the Authorization Provider to Define the User Directory for Creating SMSESSION

In the **ID Token Authentication** section of the **Authorization Provider** dialog, move the user directory that must be used for creating an SMSESSION from the **Available Directories** list to the **Selected Directories** list. The selected user directories are only used for creating an SMSESSION when this feature is implemented.

Configure a Client to Use the ID Token Policy

In the **ID Token Hint Configuration** section of the Clients dialog, select the ID token policy that was created in the previous step from the **ID Token Policy** drop-down list. Ensure that **Authorization Code** is selected in **Grant Types**.

Send Authentication Request to the Authorization Endpoint

Send an authentication request to the Authorization Endpoint in the Authorization Code Flow in the format as shown in the following example request:

Note: This feature is not supported in the Implicit Flow.

Example request:

```
https://wa.example.com/affwebservices/CASSO/oidc/sample_client/authorize?
client_id=C1aaaa3cd799&response_type=code&scope=openid%20address&state=UoS53-
qFk0oI43icRjURWpk0OiM&redirect_uri=https://
wa.example.org&Fredirect_uri&state=123&id_token_hint=eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJ0Z
```

Example response:

```
HTTP/1.1 302 302
Date: Wed, 19 Jan 2022 12:22:40 GMT
Server: Apache
```

```
Set-Cookie:
 SMSESSION=S2Xz0MZnrOtWSfzsqmBT7AfjfcL3BJqg6vN2sv9IT56FO3iBqkWcYmwwrRViLPhvF91987gehTr9jcoDagMWhnUa90irCqkcWWbxzFsoBDNE
+D7yCjIM7Q1YuioMvx0i834AIr/+CPhFEfLxucVia8qXhkKYKk+OIox6n5PawWJRg+22aRCn2+5q5CLVzO; Domain=eaxmple.com; Path=/
Location: https://wa.cons.com?
code=N2JjODc5ZmYtMDI2My00ZjY5LTk3OTAtNjE3YmZmNDM5YjY0LUtzUTRNQzFabVdnejBSUG5mclQ2OWNlbUxnVT0%3D&state=123
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
```

Manage Trusted OIDC Provider and ID Token Policy Using REST APIs

You can manage a trusted OIDC provider and an ID token policy using REST APIs.

Trusted OIDC Provider

Endpoints:

- **GET**: /ca/api/sso/services/policy/v1/FedTrustedOIDCProviders
- **POST**: /ca/api/sso/services/policy/v1/FedTrustedOIDCProviders
- **PUT**: /ca/api/sso/services/policy/v1/FedTrustedOIDCProviders/{FedTrustedOIDCProviderName}
- **DELETE**: /ca/api/sso/services/policy/v1/FedTrustedOIDCProviders/{FedTrustedOIDCProviderName}

Example request with minimum parameters:

```
{
  "Name": "TrustedOIDCProvider_MinPayload",
  "Issuer": "https://example.com/default/",
  "UserLookup": "user_loginid",
  "DefaultAuthnLevel": 5,
  "ValidateAudience": true,
  "VerificationCertLink": {
    "path": "/FedCertificates/ah"
  },
  "RevocationStatusCheck": false
}
```

Example response for a response with minimum parameters:

```
{
  "responseType": "object",
  "data": {
    "id": "CA.FED::TrustedOIDCProvider@5316face-8d51-4c02-a964-3d0a00277714",
    "type": "FedTrustedOIDCProvider",
    "Name": "TrustedOIDCProvider_MinPayload",
    "Issuer": "https://example.com/default/",
    "DefaultAuthnLevel": 5,
    "ValidateAudience": true,
    "UserLookup": "user_loginid",
    "RevocationStatusCheck": false,
    "VerificationCertLink": {
      "id": "CA.FED::Certificate@fa9e4a11-b7c7-4d8d-bac8-bcc546ead344",
      "path": "/FedCertificates/ah",
    }
  },
```

```
}
```

Example request with maximum parameters:

```
{
  "Name": "TrustedOIDCProvider_MaxPayload",
  "Issuer": "https://example.com/default/",
  "UserLookup": "user_loginid",
  "DefaultAuthnLevel": 5,
  "AuthnLevelClaim": "user_authlevel",
  "ValidateAudience": true,
  "SkewTime": 300,
  "VerificationCertLink": {
    "path": "/FedCertificates/ah"

  },
  "RevocationStatusCheck": true,
  "IntrospectionEndPoint":
  "https://example.com/default/oauth2/v1/introspect",
  "IntrospectionClientCred": [
    {
      "ClientID": "71fec478-9b64-4b97-a277-0a8cdde7c5",
      "ClientSecret": "2c6d0948-ad5f-4370-94f8-569b77727d63"
    }
  ]
}
```

Example response for a response with maximum parameters:

```
{
  "responseType": "object",
  "data": {
    "id": "CA.FED::TrustedOIDCProvider@534bd4ba-1f5e-4ce5-b1e1-53d01f543ef6",
    "type": "FedTrustedOIDCProvider",
    "Name": "TrustedOIDCProvider_MaxPayload",
    "Issuer": "https://example.com/default/",
    "DefaultAuthnLevel": 5,
    "AuthnLevelClaim": "user_authlevel",
    "ValidateAudience": true,
    "IntrospectionEndPoint": "https://example.com/default/oauth2/v1/introspect",
    "UserLookup": "user_loginid",
    "SkewTime": 300,
    "RevocationStatusCheck": true,
    "IntrospectionClientCred": [
      {
        "id": "CA.FED::IntrospectionClientCredentials@c50c5fdd-52a8-4b20-abb0-75042a323dc7",
        "type": "FedIntrospectionClientCredentials",
        "ClientID": "71fec478-9b64-4b97-a277-0a8cdde7c5",
        "ClientSecret": "2c6d0948-ad5f-4370-94f8-569b77727d63"
      }
    ],
    "VerificationCertLink": {
      "id": "CA.FED::Certificate@fa9e4a11-b7c7-4d8d-bac8-bcc546ead344",
      "path": "/FedCertificates/ah",
```

```
    }
  },
}
```

ID Token Policy

Endpoints:

- GET: /ca/api/sso/services/policy/v1/FedIDTokenPolicies
- POST: /ca/api/sso/services/policy/v1/FedIDTokenPolicies
- PUT: /ca/api/sso/services/policy/v1/FedIDTokenPolicies/{FedIDTokenPolicyName}
- DELETE: /ca/api/sso/services/policy/v1/FedIDTokenPolicies/{FedIDTokenPolicyName}

Example request with minimum parameters:

```
{
  "Name": "IDTokenPolicy_MinPayload",
  "TokenProviderFilter": "{\"condition\":\"AND\",\"rules\":
     [{\"id\":\"name\",\"operator\":\"equal\",\"value\":\"
     TrustedOIDCProvider_MinPayload\"}],\"valid\":true}",
  "SessionMaxTimeout": 3,
  "SessionIdleTimeout": 1,
  "SyncAudit": true,
  "JWERequired": true,
  "IgnoreExistingSMSession": false,
  "PersistentSession": false
}
```

Example response for a response with minimum parameters:

```
{
  "responseType": "object",
  "data": {
    "id": "CA.FED::IDTokenPolicy@463979cb-7f2a-418e-8ebe-5698b1e289b5",
    "type": "FedIDTokenPolicy",
    "Name": "IDTokenPolicy_MinPayload",
    "TokenProviderFilter": "{\"condition\":\"AND\",\"rules\": \n     [{\"id\":\"name\",\"operator\":\"equal\",
\"value\":\"\n     TrustedOIDCProvider_MinPayload\"}],\"valid\":true}",
    "SessionDrift": 0,
    "IgnoreExistingSMSession": false,
    "SessionIdleTimeout": 1,
    "SessionMaxTimeout": 3,
    "JWERequired": true,
    "SyncAudit": true,
    "PersistentSession": false
  },
}
```

Example request with maximum parameters:

```
{
  "Name": "IDTokenPolicy_MaxPayload",
  "TokenProviderFilter": "{\"condition\":\"AND\",\"rules\":
     [{\"id\":\"name\",\"operator\":\"equal\",\"value\":\"
     TrustedOIDCProvider_MaxPayload\"}],\"valid\":true}",
  "SessionMaxTimeout": 3,
```

```
  "SessionIdleTimeout": 1,
  "MaxTokenLife": 300,
  "SyncAudit": true,
  "JWERequired": true,
  "IgnoreExistingSMSession": false,
  "PersistentSession": true,
  "SessionDrift": 0
}
```


Example response for a response with maximum parameters:


```
{
  "responseType": "object",
  "data": {
    "id": "CA.FED::IDTokenPolicy@9d79d0aa-08b9-4c48-87b4-c9314ccb3de2",
    "type": "FedIDTokenPolicy",
    "Name": "IDTokenPolicy_MaxPayload",
    "TokenProviderFilter": "{\"condition\":\"AND\",\"rules\": \n     [{\"id\":\"name\",\"operator\":\"equal\",
\"value\":\"\n     TrustedOIDCProvider_MaxPayload\"}],\"valid\":true}",
    "MaxTokenLife": 300,
    "SessionDrift": 0,
    "IgnoreExistingSMSession": false,
    "SessionIdleTimeout": 1,
    "SessionMaxTimeout": 3,
    "JWERequired": true,
    "SyncAudit": true,
    "PersistentSession": true
  },
}
```


AMR Mapping

SiteMinder can consume a custom AMR value in a customer environment, and can normalize it for using it in the SiteMinder and Symantec Identity Security Platform environments. SiteMinder is enhanced to normalize an authentication method to an AMR value, and then propagate the AMR value in an ID Token Hint to IDSP. This feature ensures a seamless login experience to a customer across multiple applications, by propagating the authentication context.

Also, this feature is used in the implementation of allowing an unauthenticated mobile app user of SiteMinder to exchange an ID Token that is generated by any trusted OIDC provider for an SMSESSION at the Authorization endpoint. For information about how AMR mapping can be used to determine an authentication level in this feature, see Use ID Token Hint to Generate an SMSESSION.

Overview

To implement the feature, SiteMinder uses the default AMR value in certain authentication schemes, or the authentication method information that is present in an assertion or a token. The following scenarios describe how an AMR value is determined:

- The AMR value is automatically determined in the following authentication schemes using the default AMR value that is sent in an ID Token Hint to IDSP:

  - Basic
  - Basic over SSL
  - HTML Forms
  - IWA
  - Kerberos
  - Authentication Chain
    - Multi Factor Authentication Chain
    - NTLM fallback to form
    - Kerberos Fallback to FORM
    - IWA Fallback to FORM
  - RADIUS Server
  - RADIUS CHAP/PAP
  - IDSP Template
  - X.509 Client Certificate
  - X.509 Client Certificate and Basic
  - X.509 Client Certificate or Basic
  - X.509 Client Certificate and HTML Forms
  - X.509 Client Certificate or HTML Forms

  As the AMR value is automatically determined, an AMR mapping is not required for these authentication schemes.
- For a JWT authentication scheme or an SAML 2.0 SP-> IdP federation partnership or while using an ID Token Hint to generate an SMSESSION in OIDC authorization code flow, the authentication method information that is present in a token or an assertion is used to determine the AMR type. The AMR Mapping feature enables you to map an authentication method to an AMR type. SiteMinder uses this mapping to determine the AMR type.

  For example, consider that a mapping exists between the **urn:oasis:names:tc:SAML:2.0:ac:classes:X509** authentication method and the **POP** AMR type. When SiteMinder receives an assertion with this authentication method, the AMR value is set to **POP** in the SiteMinder session and the same value is sent in an ID Token Hint.

The AMR value of a new session is stored in the SM_USER_AMR_KEY cookie in a sessionspec and it can be used in policies or responses. In step-up authentication using the IDSP authentication scheme, the value of SM_USER_AMR_KEY can be used in an ID Token Hint. SiteMinder cannot determine the AMR type if the authentication method is not present in an assertion or a token, or a mapping between an authentication method and an AMR type does not exist.

Configure AMR Mapping in SAML 2.0 SP-> IdP Federation Partnerships

In SAML 2.0 SP-> IdP federation partnerships, an ACR is considered as the AMR. To support the feature in SAML 2.0 SP-> IdP federation partnerships, the **Configure AuthnContext** dialog of federation partnership now includes the **AMR Configuration** section with the following fields:

- **Read AMR from Authentication Context Class**

  Instructs SiteMinder to determine AMR from the AuthnContextClassRef attribute in an assertion and store the AMR in the SM_USER_AMR_KEY cookie.

  Example:

  ```
  <ns2:AuthnContext> <ns2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:Password</
  ns2:AuthnContextClassRef> </ns2:AuthnContext>
  ```
- **Read AMR from Assertion Attribute**

  Instructs SiteMinder to determine AMR from an assertion attribute and store the AMR in the SM_USER_AMR_KEY cookie.
  - **AMR Assertion Attribute**

    Specifies the attribute from which AMR must be determined if **Read AMR from Assertion Attribute** is selected.

Example: If **amr** is defined as the AMR Assertion Attribute

```
<saml:Attribute Name="amr" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
  <saml:AttributeValue xsi:type="xs:string">pwd</saml:AttributeValue>
</saml:Attribute>
```

For information about how authentication context works in SP->IdP partnerships and how to configure the rest of the fields that are related to authentication context, see Enable Authentication Context Requests at the SP-to-IdP Partnership.

List of Authentication Methods Mapped to AMR Types

By default, the AMR Mapping feature provides a list of possible authentication method attributes that are mapped to their respective AMR types for the supported objects. The table in the AMR Mapping dialog lists these default mappings. To view the list, navigate to **Infrastructure**, **Authentication**, **AMR Mapping**.

Note: The authentication schemes with a default AMR type are not listed in the default mappings list as the AMR type information is available already.

Assign AMR Types to a Pre-existing Authentication Method

By default, each provided authentication method is mapped to the possible AMR types. You can add other AMR types to an authentication method.

Follow these steps:

1. Open Administrative UI.
2. Navigate to **Infrastructure**, **Authentication**, **AMR Mapping**.
3. Locate the required authentication method in the table, click **Actions** corresponding to it, and then click **Modify**.
4. Select an AMR type that must be associated with the authentication method from the **AMR Type** drop-down list. Multiple AMR types can be assigned to an authentication method.
5. Save the change.

Create an Authentication Method and Assign AMR Types

If the provided list of authentication methods does not contain a required authentication method, you can use the AMR Mapping feature to create the required authentication method. Before you create an authentication method, verify if the required AMR type is provided by default. To view the list of available AMR types, navigate to **Infrastructure**, **Authentication**, **AMR Type**. If the required AMR type is not displayed in the list, create the required AMR type and then proceed to create an authentication method.

Create an AMR Type

You can add an AMR type to be associated with an authentication method.

Follow these steps:

1. Open Administrative UI.
2. Navigate to **Infrastructure**, **Authentication**, **AMR Type**.
3. Click **Create AMR Type**.
4. Enter the name of the new AMR type in **AMR Type**.
5. Enter a description of the new AMR type in **Description**.
6. (Optional) Enter an authentication level to the selected AMR type in **Authentication Level**.
   Range: 1-1000
   Note: In the payload of AMR Type in REST APIs, the default value of the authentication level is set to 0. Ensure that you modify it to a valid value in the range of 1 - 1000 before you send a request.
7. Save the change.

Create an Authentication Method

Ensure that the required AMR type exists before you proceed.

Follow these steps:

1. Open Administrative UI.
2. Navigate to **Infrastructure**, **Authentication**, **AMR Mapping**.
3. Click **Create AMR Mapping**.
4. Enter the name of the new attribute that indicates the authentication method in **Authentication Method**.
5. Enter a description of the new attribute in **Description**.
6. Select an AMR type that must be associated with the authentication method from the **AMR Type** drop-down list. Multiple AMR types can be assigned to an authentication method.
7. Save the change.

Mitigate the Binary Attribute Issue in an Encrypted Password Blob in Oracle Unified Directory 12c

When an Oracle Unified Directory (OUD) 11g user directory that is configured with basic password policies is upgraded to OUD 12c, SiteMinder fails to read the encrypted password blob that was created in OUD 11g. This results in the loss of password history in OUD 12c.

To mitigate the loss of password history, perform the following steps on Policy Server:

1. Upgrade the Policy Server to 2022.Jun.01 build. For information about upgrading Policy Server in an in-place upgrade, see Upgrade Policy Server.
2. Upgrade OUD 11g to OUD 12c. Ensure that the updated password blob information is reflected in the **metsiteminderpassworddatatext;binary** user attribute.
3. Log in to Administrative UI.
4. Navigate to **Infrastructure**, **Directory**, **User Directories**.
5. Open the record of the user directory that has been migrated from OUD 11g to OUD 12c, and click Modify.
6. Append the existing value of the **Password Data (RW)** field in the **User Attributes** section with **;binary**. After the change, the value in the **Password Data (RW)** field must be **metsiteminderpassworddatatext;binary**.
7. Click Submit.
8. Restart all Policy Servers that are connected to the policy store, or wait for the cache update of CA.XPS:: $CacheCheckDelay (default refresh interval is 5 minutes).

Define a Sitename to Retrieve KDCs in Kerberos Authentication Scheme

The Kerberos authentication scheme has been enhanced in the 12.8.0700.2947 build.

To configure a Kerberos authentication scheme, MIT Kerberos libraries are configured to use the krb5.conf for information on how to communicate with a KDC. MIT Kerberos libraries read the krb5.conf file to fetch the KDCs to be used in the authentication service. If a KDC is not defined, MIT Kerberos libraries retrieve all the KDCs in a domain when dns_look_up is set to true in the krb5.conf file.

From 12.8.0700.2947 build, SiteMinder provides the sitesPlugin plugin that allows you to define a site, which allows MIT Kerberos libraries to retrieve all the KDCs that are configured in a site instead of retrieving all the KDCs in a domain. Typically, a site defines location and a domain may span across multiple sites.

How a KDC is Retrieved

The following steps describe how the Kerberos authentication works in Policy Server and Access Gateway:

Existing flow without sitesPlugin:

1. A client accesses a resource that is protected with the Kerberos authentication scheme.
2. MIT Kerberos libraries read krb5.conf to retrieve KDCs. The DNS query that is used is in the following format:
   ```
   _kerberos._UDP.REALM.COM
   _kerberos._tcp.REALM.COM
   ```
   Example:
   ```
   _kerberos._UDP.IWAAD.COM
   _kerberos._tcp.IWAAD.COM
   ```
3. MIT Kerberos libraries use the retrieved KDCs for authentication services.

With sitesPlugin:

1. A client accesses a resource that is protected with the Kerberos authentication scheme.
2. MIT Kerberos libraries call sitesPlugin.so to retrieve KDCs.
3. sitesPlugin reads krb5.conf to retrieve the configured sitename, and then sends a site-specific DNS query in the following format:
   ```
   _kerberos._udp.sitename._sites.REALM.COM
   _kerberos._tcp.sitename._sites.REALM.COM
   ```
   Example:
   ```
   _kerberos._udp.Default-First-Site-Name._sites.IWAAD.COM
   _kerberos._tcp.Default-First-Site-Name._sites.IWAAD.COM
   ```
   The query response contains the list of servers (KDCs) that are configured for the specified sitename.
   Example: example.net
4. MIT Kerberos libraries use the retrieved KDCs for authentication services.

Configure the Feature

The feature configuration includes the configuration of Policy Server and Access Gateway.

Follow these steps:

1. Retrieve the sitename from your server.
   Example: In Active Directory, click Active Directory Sites and Services in Server Manager. The Active Directory Sites and Services dialog displays all the available sites (for example, Default-First-Site-Name) and the KDCs configured within them as Servers.
   On AD, to find a sitename that corresponds to a server, run the following command:
   ```
   nltest /DSADDRESSTOSITE:<server-name>
   ```
2. Create the **libkrb5** folder in the following location on Policy Server and Access Gateway:
   - **Windows**: C:\krb5\plugins
   - **Linux**: /opt/siteminder/lib/krb5/plugins/
3. On Policy Server, navigate to <*siteminder-installation-path*>/bin and copy sitesPlugin.dll (Windows) or sitesPlugin.so (Linux), and place the plugin in the corresponding libkrb5 folder that was created in Step 2.
   - **Windows**: C:\krb5\plugins\libkrb5\sitesPlugin.dll
   - **Linux**: /opt/siteminder/lib/krb5/plugins/libkrb5/sitesPlugin.so
4. On Access Gateway, navigate to <access-gateway-installation-path>\agentframework\bin and copy sitesPlugin.dll (Windows) or sitesPlugin.so (Linux), and place it in the libkrb5 folder
   - **Windows**: C:\krb5\plugins\libkrb5\sitesPlugin.dll
   - **Linux**: /opt/siteminder/lib/krb5/plugins/libkrb5/sitesPlugin.so

5. Take a backup of the krb5 file on Policy Server and Access Gateway, and then perform the following steps on Policy Server and Access Gateway:
   a. Open the krb5 file.
      - **Windows:** C:\windows\krb5.ini
      - **Linux:** /etc/krb5/krb5.conf
   b. Comment or delete the KDCs that are defined in **[realms]**.
      Example:
      ```
      [realms]
       IWAAD.COM = {
       # kdc = hostname.iwaad.com
         default_domain = iwaad.com
      }
      ```
   c. Add the following section after **[domain_realm]**:
      ```
      [locate]
          service_locator = {
               site = <sitename-as-retrieved-in-step1>
          }
      ```
      Example:
      ```
      [locate]
          service_locator = {
               site = Default-First-Site-Name
          }
      ```
      To define multiple sitenames, enter multiple values that are separated by a space.
      ```
      site = <sitename1>
      <sitename2>
      <sitenameN>
      ```

The feature is configured.

Verify the sitesPlugin

To verify whether sitesPlugin is able to retrieve KDCs, enable KRB5_TRACE to view the sitesPlugin log messages, which are denoted by **SITE**.

```
Export KRB5_TRACE=/tmp/Kerberos_trace.log
```

Example:

```
[4161126] 1677221454.604137: SITE: Initializing DNS site-specific service locator plugin
[4161126] 1677221454.604138: SITE: Got site from profile path locate/service_locator/site: Default-First-Site-Name
realm=IWAAD.COM
```

Support for Front-Channel SLO in SiteMinder OpenID Connect Provider

From 12.8.0700.2947 build, SiteMinder as an OpenID Connect Provider (OP) supports front-channel SLO, a logout mechanism that uses a front-channel communication between OP and RPs to be logged out, through a User Agent.

You can configure the front-channel SLO feature in SiteMinder to enable the following features in RP-initiated and OP-initiated logout requests:

- Revoke Authorization Code, Access Token, Refresh Token, and ID Token (OIDC tokens) that have been generated for OIDC Clients within a user session.

This feature is enabled at an OIDC Client-level using the **Enable Front-Channel SLO** field. You must enable and configure this feature in each OIDC Client whose OIDC tokens must be revoked when an SLO request is initiated with a user session.

When an SLO request is received, SiteMinder revokes the OIDC tokens that are generated for *all* the OIDC Clients in which the feature is enabled, within a user session, even though the SLO request contains only one OIDC Client ID.
Example: Consider that an Authorization Provider API is mapped with the following Clients:

- C1, in which the front-channel SLO feature is enabled and OIDC tokens are generated
- C2, in which the front-channel SLO feature is enabled and OIDC tokens are generated
- C3, in which the front-channel SLO feature is enabled but OIDC tokens are not generated
- C4, in which the front-channel SLO feature is disabled and OIDC tokens are generated

If an SLO request with C1 is initiated, SiteMinder revokes the OIDC tokens that are generated for C1 and C2.

You can also instruct SiteMinder to revoke the OIDC tokens, when a session is logged out using the URI of the **LogOffURI** ACO parameter.

- Log out an existing session. This feature is enabled at an OIDC Authorization-provider level using the **Invalidate SMSESSION on Logout** field.

How the Front-Channel SLO Feature Works

When the front-channel SLO feature is enabled, SiteMinder maintains a list of all the OIDC Clients that were issued OIDC tokens within a user session. An SLO request can be initiated with an ID Token (that is sent in the ID Token Hint parameter) or a user session. If an SMSESSION and ID Token Hint are sent in an SLO request, the session in ID Token Hint takes precedence over the SMSESSION for further processing. If both are not sent in an SLO request, the request is considered invalid.

If the feature is enabled, SiteMinder performs the following steps when an SLO request is received:

1. Revokes all OIDC tokens that were issued to all the OIDC Clients in which the feature has been enabled, within the user session.
2. Logs out the existing SMSESSION in all/any browsers, if the **Invalidate SMSESSION on Logout** field is selected in the OIDC Authorization Provider that is associated with the OIDC Clients.
3. Displays the default front-channel logout page of OP, **oidcsignoutconfirmurl.jsp**, in the https://*<hostname>*/affwebservices/public/oidcsignoutconfirmurl.jsp format. The Revoke OIDC Tokens section in this topic contains more information about oidcsignoutconfirmurl.jsp.
4. The front-channel logout page of OP sends logout calls to all the registered RPs that are mapped with OIDC Clients, which were issued OIDC tokens within the session when the feature is enabled. The RPs act on the logout call request according to their custom business logic.

   If the feature is enabled in an OIDC Client but the OIDC Client is not issued any OIDC tokens within the user session that is received in the SLO request, a logout call is not sent to its RP as the OIDC Client is not associated with the specific user session that is received in the SLO request.
5. Redirects to the post logout URI that is received in the request, if a valid post logout redirect URI is sent in the SLO request.

Configure SiteMinder to Support Front-channel SLO in OIDC

The front-channel SLO feature in SiteMinder allows you to revoke OIDC tokens and also log out an existing SMSESSION, with an additional configuration.

Prerequisites

- Ensure that a session store is configured.
- To revoke or introspect an ID Token, ensure that **Persist ID Token** is selected in the Client configuration.

Revoke OIDC Tokens

Configure an OIDC Client to allow revocation of OIDC tokens that are issued to an OIDC Client when an SLO logout request is initiated.
Follow these steps:

1. Navigate to **Federation**, **OpenID Connect**, **Clients** in Administrative UI and create/modify an OIDC Client.
2. In the **Front-Channel SLO** section, select **Enable Front-Channel SLO**.
3. Define the front-channel logout URI of RP to which SiteMinder sends a logout call request in **RP Logout URL**.
4. (Optional) To add the issuer and session ID values as query parameters in the logout call request to RP, select **Add iss and sid in Logout Call/Request to RP**.
5. (Optional) To revoke OIDC tokens using the URI that is configured in the **LogOffURI** ACO parameter, select **Revoke Tokens through the URI of LogOffURI ACO Parameter**.
6. Define the additional time period that an expired ID Token must be considered as a valid token in an SLO logout request in **ID Token Hint Grace Period**. In this feature, the ID Token Hint is used to validate that the request is from a valid RP.
7. Define the redirect URIs that must be used after the logout call request is sent to all the RP logout URLs, in the case of RP-initiated logout requests. One of the configured URIs must match the URI that is received in a logout request. The front-channel logout page of OP, oidcsignoutconfirmurl.jsp, is provided by default. This page provides a template with a sample script on how to send RP logout URLs in iFrames and to redirect to a post logout redirect URL. The sample script is at <access-gateway-installation-path>\Tomcat\webapps\affwebservices\public. After rendering this page using iFrames, the script waits for a response from RPs for the time period as defined in the initialwaittime, waitTime, and MAXWAIT variables in the script. If there is no response from RPs within the time period that is defined in MAXWAIT, the script redirects the user to post logout redirect URI. Customize this script per your business needs.
8. Select **Persist ID Token**.
9. Complete the other required configurations.
10. Save the changes.

The feature is enabled. SiteMinder displays a Logout URL in the following format, along with the endpoints information:

```
Logout URL: https://<access-gateway_server>:<port>/affwebservices/CASSO/oidc/<oidc-client-name>/logout
```

Any OIDC tokens that are generated after enabling the feature are revoked when SiteMinder receives an SLO request. Any OIDC tokens that were previously generated before the enabling the feature are ignored.

Log Out an Existing SMSESSION

Configure an Authorization Provider to allow logging out an existing SMSESSION from a session store and a web browser when an SLO logout request is initiated.

Follow these steps:

1. Navigate to **Federation**, **OpenID Connect**, **Authorization Providers** in Administrative UI and create/modify an OIDC Authorization Providers.
2. In the **Front-Channel SLO** section, select **Invalidate SMSESSION on Logout**.
3. Complete the other required configurations.
4. Save the changes.

SLO Request Format

Use the Logout URL that is displayed once the OIDC Client is configured to send a front-channel SLO request.
For an RP-initiated request, an SLO request is in the following format:

```
https://<hostname>/affwebservices/CASSO/oidc/<oidc-client-name>/logout?id_token_hint=<id-token-hint>&post_logout_redirect_uri=<post-logout-uri>&state=123456
```

Example:

```
https://example.net/affwebservices/CASSO/oidc/Client1/logout?
id_token_hint=eyJraWQiOiIwMDA1MGZjZC00NTI1LTEzZTctODdiNi0yYTQ1MGE4MzAwMDAiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJ1c2V&post_log
example.com/test&state=123456
```

For an OP-initiated request, an SMSESSION cookie is mandatory and its SLO request must be in the following format:

```
https://<hostname>/affwebservices/CASSO/oidc/<oidc-client-name>/logout
```

Example:

```
https://example.net/affwebservices/CASSO/oidc/Client1/logout
```

Changes in SiteMinder to Support Front-Channel SLO in OIDC

- **Session ID**: A session ID remains constant in a user session, even after a step up authentication.
- **Well-Known Configuration**: If the Well-Known configuration is enabled, the Provider Metadata endpoint now includes the following additional information:

```
"end_session_endpoint":
"https://Access_Gateway_server:port/affwebservices/CASSO/oidc/validationClient/logout",
   "frontchannel_logout_supported": true|false,
   "frontchannel_logout_session_supported": true|false,
```

Support for Change Password in Authentication and Authorization Web Services

From 12.8.0700.2947 build, the authentication and authorization web services now support the change password functionality, in which a user must change an expired password upon a successful login.

The change password flow is invoked when the **Force Password Change** option is enabled for a user. The flow consists of the following steps:

1. Client invokes the Login API.
2. Login API fails and its response includes SM_AUTHREASON as 20 and also provides an SMTOKEN that must be used in the ChangePWD API.
3. Client invokes the ChangePWD API with a new password and the SMTOKEN provided by the Login API in its response.
4. SiteMinder changes the user password if the new password meets the SiteMinder password criteria and provides an SMSESSION. If the change password action fails due to a mismatch with the password criteria, the response of the ChangePWD API contains the exact reason of mismatch in the **userPasswordMessage** parameter.

The following examples describe the request and response of change password in the supported formats.

- Example: Change Password – SOAP
- Example: Change Password – REST XML
- Example: Change Password – REST JSON

Example: Change Password – SOAP

1. Login Flow
Request:

URI of Login Request: http://authaz.ca.local:88/authazws/auth

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:aut="http://www.ca.com/siteminder/
authaz/2010/04/15/authaz.xsd">
   <soap:Header/>
   <soap:Body>
      <aut:login>
         <identityContext>
            <userName>user1</userName>
            <password>user1pwd</password>
         </identityContext>
         <appId>app1</appId>
```

```
         <resource>/sspapi/</resource>
         <action>GET</action>
      </aut:login>
   </soap:Body>
</soap:Envelope>
```

Response

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns2:loginResponse xmlns:ns2="http://www.ca.com/siteminder/authaz/2010/04/15/authaz.xsd">
         <return>
            <message>Authentication Failed</message>
            <resultCode>LOGIN_FAILED</resultCode>
            <authenticationResponses>
               <response>
                  <name>SM_AUTHREASON</name>
                  <value>20</value>
               </response>
               <response>
                  <name>SM_REALMOID</name>
                  <value>06-3b2fb84b-45d8-47ed-82ff-394402e0ad9b</value>
               </response>
               <response>
                  <name>user_mail</name>
                  <value>user1@broadcom.com</value>
               </response>
               <response>
                  <name>SM_REDIRECTURL</name>
                  <value>/siteminderagent/forms/smpwservices.fcc?SMENC=UTF-8&amp;SMTOKEN=-
SM-%7bRC2%7dK5TtpqoctAgSloUBVeZazOvMKJKLELTYT%2f5nhNxgdMVUhXF%2fVDcRDDJF1ZN97Eo3ZXCDNvlIQGI
%3d&amp;USERNAME=user1&amp;SMAUTHREASON=20&amp;SMAGENTNAME=-SM-kpF5Hvv2iTLWFGxu5BMM8zs
%2fHmT0mHY2uY4v6KbcZGGY5rFa80GpRki9HirHb9Aq&amp;TARGET=</value>
               </response>
               <response>
                  <name>SMTOKEN</name>   <value>nVbN8bvIY9u9yGzycs8b71mNMVD8ZtupUQrkR55cbfun
+cGyBcgDNsZJ25oHpCKlQBbbckLXQxIg3ulsQOJW9w+BOJfPWTE/6Op8iYDFLVQEN3L7WY5PyeRbP0/
h4QxSRb99A7o25eT4HLUJhKN6UYD8rliWr9sdGavuv3Y48SuFKeFUUg/HQHpRZKupC09I</value>
               </response>
               <response>
                  <name>SM_REALM</name>
                  <value>authaz_rlm</value>
               </response>
               <response>
                  <name>SMLOCALE</name>
                  <value>en-US</value>
               </response>
               <response>
                  <name>SM_TRANSACTIONID</name>
                  <value>000080fe000000008ab05b79f5bef264-1434-63f35b23-0768-01436784</value>
               </response>
               <response>
                  <name>SM_AUTHTYPE</name>
                  <value>Basic</value>
```

```
               </response>
               <response>
                  <name>SM_USERDISABLEDSTATE</name>
                  <value>16777216</value>
               </response>
            </authenticationResponses>
         </return>
      </ns2:loginResponse>
   </S:Body>
</S:Envelope>
```

2. Change password flow

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:aut="http://www.ca.com/siteminder/
authaz/2010/04/15/authaz.xsd">
   <soap:Header/>
   <soap:Body>
      <aut:changepwd>
         <identityContext>
            <userName>user1</userName>
            <password>user1</password>
            <newpassword>firewall</newpassword>
 <smtoken>KOvfXkSJ5gLMckCbWO8MrZknYBjkYp5/3LH77IJh2DOzecF3CAabZ78e1uo40kxCOp17WWZyE8O8/91pj8QMvm8Tr8aIzXyYeni
+zg9SATeTTxsVpPzeMOze3oq07wFraaQ5zhy0nPYGwrRQnIzNgepTz39mvPe/dmj8auQgPTpQf9MsNluvCn43hmVv2Iyx</smtoken>
         </identityContext>
         <appId>app1</appId>
         <resource>/sspapi/</resource>
         <action>GET</action>
      </aut:changepwd>
   </soap:Body>
</soap:Envelope>
```

Response:

```
S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns2:changepwdResponse xmlns:ns2="http://www.ca.com/siteminder/authaz/2010/04/15/authaz.xsd">
         <return>
            <message>Authentication Successful</message>
            <resultCode>LOGIN_SUCCESS</resultCode>
 <sessionToken>k06aCFKxvIujX0GG7sKDMwb8LRt7IbWMEelsMYTFCM8glHGrNpJGtYlsETwDllmyjmEcL0ySWFTqasOEjtrb7o52aXQtPHVzs04T92hE
+gFHzbK1fNpalCpj9pI109WmCd59AAm17B7SL6SEd/ep1KP9DJDKmwY7973mNY0++kPmiIqxBjFERoNCL5mSdvZw/
HQ7e0TZENbxslNQEo2jtzWuX1lZYbcq2F7YHOiiu9zX4FAQ6FSysoqF0Ijf
+UGV6vp8hkh1lo1Tj8ckyLXsvsqZ52Il7XYzMwAUiVIUEyUGbv7WebmggOcsM9+XfuXTWZQ0mZ7aNCRqV4/5zym0i8vUUi4oXO4aQZnwFskKzIk0DxWHlU4
+ZtjTc8zmeynuYjMzEllWT2tnekfRZBp3BmE1Bc3srfy+cXlk/1tVPQxjFaA9FwEJGyILoYYPOycttgRR+G+XyPyhvQH5i5vAEdJ5zBAT/
Nez2VnJ/l5TkC9FRf6Pezc++q
+1+DqSq98WryTh4ZoQFB4rq8KQtFibMaYTUBWqTpXJmRN1ug9a9Yen1QVU7r1ZV4RK5dpHM7dUkwX5IcivmcXyTfeE3y90B8ZjCMlj8rxYJ99dHLOMJQs9V
+roNv5mCwXaRSB/p5XWDBqC6Gj
+YohvNHCoiJ09prMcTXMtVeVUREt64IG6nOeiarMXbrI1ZovoJmxQORJ0iZKN1HHMlbT6DqMSY1gfrjMvYxmo1VXlP8QOt5sDZ
+ugiHRRIpPEtAD80cFCp5Y0CU/USixxZKMi0akOI7BRwBMuQzrk7WFI96I/jN/
OVdexiP8eslAr4nUfVYF269Ys7+mXjUyJ7rYplICz1yLpooFWbD+r+14QfAz1x/2c6bSyp+kI65Bv5Rip2TcWIz0rH/
P8z6RYOz7kYSurrCA6Oj9O9T2RNsnQdO96Y/gWrhHEysH50o0DrSmYMnoXvUGNzanmYLRwZ6QSOhYbNeSO6+zLUVW9yGpVu</sessionToken>
            <authenticationResponses>
               <response>
```

```
                  <name>SM_AUTHREASON</name>
                  <value>23</value>
               </response>
               <response>
                  <name>SM_SERVERIDENTITYSPEC</name>
                  <value/>
               </response>
               <response>
                  <name>SM_USERDN</name>
                  <value>uid=user1,ou=People,o=security.com</value>
               </response>
               <response>
                  <name>SM_AUTHDIROID</name>
                  <value>0e-3ced2af8-51e6-4a28-9c15-81bc89663123</value>
               </response>
               <response>
                  <name>SM_REALMOID</name>
                  <value>06-3b2fb84b-45d8-47ed-82ff-394402e0ad9b</value>
               </response>
               <response>
                  <name>SM_SERVERSESSIONID</name>
                  <value>/RSyQXFKZGe9IqMxNGrT1Ou13Sw=</value>
               </response>
               <response>
                  <name>user_mail</name>
                  <value>user1@broadcom.com</value>
               </response>
               <response>
                  <name>SM_SESSIONDRIFT</name>
                  <value>0</value>
               </response>
               <response>
                  <name>SM_REDIRECTURL</name>
                  <value>/siteminderagent/forms/smpwservices.fcc?SMENC=UTF-8&amp;SMTOKEN=-
SM-%7bRC2%7d0m4dcccfUuNovTd165k%2fZkDUL63gFLyM8DyfiB7Wd9wq5MDm3iouIBUBNNkxdenZUOGVUxjuF
%2f0%3d&amp;USERNAME=user1&amp;SMAUTHREASON=23&amp;SMAGENTNAME=-SM-kpF5Hvv2iTLWFGxu5BMM8zs
%2fHmT0mHY2uY4v6KbcZGGY5rFa80GpRki9HirHb9Aq&amp;TARGET=</value>
               </response>
               <response>
                  <name>SM_UNIVERSALID</name>
                  <value>user1</value>
               </response>
               <response>
                  <name>SM_AUTHDIRNAME</name>
                  <value>ud</value>
               </response>
               <response>
                  <name>SM_REALM</name>
                  <value>authaz_rlm</value>
               </response>
               <response>
                  <name>SM_USER</name>
                  <value>user1</value>
```

```
               </response>
               <response>
                  <name>SM_TRANSACTIONID</name>
                  <value>000080fe000000008ab05b79f5bef264-1434-63f35fdd-0768-037772ae</value>
               </response>
               <response>
                  <name>SM_AUTHDIRSERVER</name>
                  <value>example.net:20001</value>
               </response>
               <response>
                  <name>SM_TIMETOEXPIRE</name>
                  <value>7200</value>
               </response>
               <response>
                  <name>SM_SERVERSESSIONSPEC</name>
                  <value>lctbWch7pnLhtSnGnZQHLaedjkNk/
qjEAxjqlB0A0ha0C4uz5aSPPMjeWZhuQ1Z3odf6Oaovi46+5hWpBDvAYlRci+WBInaGmSY1KprbE6P3hI0JrQvzzHK/9/
etflI9MV2xTZnjuvDCqERAFKNAIwtJ+qmZh2Xogh3ns7n/3pPWWEQKmeR1OfG9naLTdM2PCt6XS
+I0YWGSjHlWx0CV1nbkIt4xd9ZKspTQKfzoXHguaPIkDsqHblr8p4+NFq8VTA8HedDisLEiRzWKnemO7SVJbPskNNyClnZx9kjDlC8BPwEw2WhNxux2Wvr4
+Y3+B4Zjs0nL7HdRer3kZtB5jH5hpUug8rw9SkMBJcZ05yqqdiP0T+U3NHRpw4V7IFikJ2/h4ff+uIarHBkDnNGFx4zTDAehnlkviUB
+HYp2Q==</value>
               </response>
               <response>
                  <name>SM_AUTHDIRNAMESPACE</name>
                  <value>LDAP:</value>
               </response>
               <response>
                  <name>SM_AUTHTYPE</name>
                  <value>Basic</value>
               </response>
               <response>
                  <name>SMSSOZONE</name>
                  <value>SM</value>
               </response>
               <response>
                  <name>SM_USERDISABLEDSTATE</name>
                  <value>0</value>
               </response>
            </authenticationResponses>
         </return>
      </ns2:changepwdResponse>
   </S:Body>
</S:Envelope>
```

Example: Change Password – REST XML

1. Login Flow

Request:

URI of Login Request: http://authaz233.ca.com:88/authazws/AuthRestService/login/app1/sspapi/

```
<loginRequest>
       <userName>user2</userName>
       <password>user12</password>
       <action>GET</action>
```

```
</loginRequest>
```

Response:

```
<loginResponse>
   <message>Authentication Failed</message>
   <resultCode>LOGIN_FAILED</resultCode>
   <authenticationResponses>
      <response>
         <name>SM_AUTHREASON</name>
         <value>20</value>
      </response>
      <response>
         <name>SM_REALMOID</name>
         <value>06-3b2fb84b-45d8-47ed-82ff-394402e0ad9b</value>
      </response>
      <response>
         <name>user_mail</name>
         <value>user2@ca.com</value>
      </response>
      <response>
         <name>SM_REDIRECTURL</name>
         <value>/siteminderagent/forms/smpwservices.fcc?SMENC=UTF-8&amp;SMTOKEN=-SM-
%7bRC2%7diCprKEHtCfJLjPtw2ijtJkhOH2T73H0lemelemOFsw0cWh5CpqHKxOiHXsGLltImAx6sLwVPYFs
%3d&amp;USERNAME=user2&amp;SMAUTHREASON=20&amp;SMAGENTNAME=-SM-kpF5Hvv2iTLWFGxu5BMM8zs
%2fHmT0mHY2uY4v6KbcZGGY5rFa80GpRki9HirHb9Aq&amp;TARGET=</value>
      </response>
      <response>
         <name>SMTOKEN</name>
<value>oTykutygy0ZT10t5CGXDVEyuzlAitFguWTJ2eWEqeNi30MYwc14+bd2yoouxQeGkw0714Ps9BZg7dEGmM0GQgsWgV25IEpQaA7K7LNymSNH5fJ7w
+fnULk7CdrHmYgNeFazPCHGp2c5EFcvEIITJCNISlgK3WzLAHBuUsdPtewBrynFgIJ8u6e2ga</value>
      </response>
      <response>
         <name>SM_REALM</name>
         <value>authaz_rlm</value>
      </response>
      <response>
         <name>SMLOCALE</name>
         <value>en-US</value>
      </response>
      <response>
         <name>SM_TRANSACTIONID</name>
         <value>000080fe000000008ab05b79f5bef264-1434-63ec77fe-1804-02e44823</value>
      </response>
      <response>
         <name>SM_AUTHTYPE</name>
         <value>Basic</value>
      </response>
      <response>
         <name>SM_USERDISABLEDSTATE</name>
         <value>16777216</value>
      </response>
   </authenticationResponses>
</loginResponse>
```

2. Change password flow

Request:

URI of Change Password Request: https://hostname:port/authazws/AuthRestService/changepwd/appID/Resource

Example: http://authaz233.ca.com:88/authazws/AuthRestService/changepwd/app1/sspapi/

```
<changePasswordRequest>
       <userName>user2</userName>
       <password>user2</password>
       <action>POST</action>
<smtoken>lXERVnFxXFNdJavDiHtkQuwGMAp1ljkl1WxQN4fmnefTbBriUF3/rcUSPzertp6YomFL2pWD2XuWI3Ev4Y4ELJrMOw6aOKYwci
+j5lCfFVfWPdKIbHMPBvBWEdq9TQPlBsJ0bP7jr2spYN+uSBEWX3Pzw5l89doSXYY+iPfVQpVfRIx/W3kb2wxoZo9NH1wS</smtoken>
       <newpassword>user12</newpassword>
</changePasswordRequest>
```

Response:

```
<changePasswordResponse>
   <message>Authentication Successful</message>
   <resultCode>LOGIN_SUCCESS</resultCode>
<sessionToken>3HfzMoKfYSrmVGUGrycO+WMy2L0hCSRw0Pf2Cq/JKSwtvsb/kcN3xa0hYbpbjh4Dl8/g
+aJazrk97mGQt5iDIRiHdCuV/g585vaeIwrrVPIvNBFTHH6akAQKOIaiE1pznEJgpa2xEhjv04UEjFwcsJGXc/
A1AfYj3s5JHGvT2bQlDK8aSqvjerFLtgGZfFJu3+SJy23XwtEmsfUqyJnx3UVXel6sOr7Y98WtqztRHD/
WSqROyoBUQBh6FnLBmTce8dBopk9Qwq+F84h1IoYXHyiKCPOB
+2CjllaYcqIugDKBtmepIkUZvWcLfzSrOEEQjSUQ4E31Ed93NjSykXkTIgEYlA3SgVEvKE0oqwnQhlFWdVspO0wMpXIR1+y7OCppmtNpaIXGW2M2fKPwkkA
+nwTSEAAPEirD97CYlyexKtw7a82VepIzTgcC012mKMUKwMgF3viYXgqlT9XFhDX/mat37ElfsTnf9ge/
CLFz2P6UEn8bJA0BGQMFFHy2K2bOxaBqZsSzIKeyOwh5rISPyV9YPjL2CVPBUccS0ZFipVnPOfMmrjhJoTjnfxMfshw1ooGGj1qTtfaMhKRqtxDA3i2lUk7
vPmZrUixDmqn3hs1qxfh6L+kCeae6dVRx2v/6sNgiS/eP6r1eZ6QW093vLzpTISrC0KQXIEA8hWFcV20pnRJJgXsnxV8jlBgJM/
hWqi8ZIh0ZPrZo2d/MMaKgtp9YW6aO7+xUNhOsMQ0TArU+y1Lr4QiESOqwxfZjGMxr6X5npAY3ReEpfm4/
XpcSXIvGIcYECuohA8/oLzqc+Z+lFVRQ8wud5nF4cAenkpHGybLJ5AOJn/3kgsjwQKXN/
MRre9Fogi0bFYU8I4Wc9ypydDaK7sBTpmrHyEgNkl5N6AJkfWLp9cFjja7cfGqIZHXYY6c3NDqltMGd3J1GSJPED4kbiMHvx0SrerXCI2sVkBHBpOZR9B/8
+V4Hc9yyr6Dbt4RMhrdIB951CX4cnfvQO7j3EKe</sessionToken>
   <authenticationResponses>
      <response>
         <name>SM_AUTHREASON</name>
         <value>23</value>
      </response>
      <response>
         <name>SM_SERVERIDENTITYSPEC</name>
         <value/>
      </response>
      <response>
         <name>SM_USERDN</name>
         <value>uid=user2,ou=People,o=security.com</value>
      </response>
      <response>
         <name>SM_AUTHDIROID</name>
         <value>0e-3ced2af8-51e6-4a28-9c15-81bc89663123</value>
      </response>
      <response>
         <name>SM_REALMOID</name>
         <value>06-3b2fb84b-45d8-47ed-82ff-394402e0ad9b</value>
      </response>
```

```
      <response>
         <name>SM_SERVERSESSIONID</name>
         <value>uNJNn1AB7Y6MgamQeneYgLISVmc=</value>
      </response>
      <response>
         <name>user_mail</name>
         <value>user2@broadcom.com</value>
      </response>
      <response>
         <name>SM_SESSIONDRIFT</name>
         <value>0</value>
      </response>
      <response>
         <name>SM_REDIRECTURL</name>
         <value>/siteminderagent/forms/smpwservices.fcc?SMENC=UTF-8&amp;SMTOKEN=-SM-
%7bRC2%7dVCQX12Gzi2s8oPcgL2hH0kBRftl%2fGC4hd2cOui3PfWKssfthjY8dnvDApLNr%2fB7iX1gAkXXrP6o
%3d&amp;USERNAME=user2&amp;SMAUTHREASON=23&amp;SMAGENTNAME=-SM-kpF5Hvv2iTLWFGxu5BMM8zs
%2fHmT0mHY2uY4v6KbcZGGY5rFa80GpRki9HirHb9Aq&amp;TARGET=</value>
      </response>
      <response>
         <name>SM_UNIVERSALID</name>
         <value>user2</value>
      </response>
      <response>
         <name>SM_AUTHDIRNAME</name>
         <value>ud</value>
      </response>
      <response>
         <name>SM_REALM</name>
         <value>authaz_rlm</value>
      </response>
      <response>
         <name>SM_USER</name>
         <value>user2</value>
      </response>
      <response>
         <name>SM_TRANSACTIONID</name>
         <value>000080fe000000008ab05b79f5bef264-1434-63ec74d1-0b0c-00254823</value>
      </response>
      <response>
         <name>SM_AUTHDIRSERVER</name>
         <value>example.net:20001</value>
      </response>
      <response>
         <name>SM_TIMETOEXPIRE</name>
         <value>7200</value>
      </response>
      <response>
         <name>SM_SERVERSESSIONSPEC</name>
```

```
<value>NKzU8qGfHsUm+qBn860/8zzMV1lUEHZnQjPJ8PdoqdiYj0FZg49hvZ8cGnIUBd/
L1m2EUGfMfwmhajhX5a5B2UuU4DOri424rmLKZfU4OZDO6q+jUET/Jav/RPlSTYMFiCXVYUAymmoQlRLbDbNR74w693B2/
zHqrr+7fqm3BLPUbCUOKku4/OAV7fteW7EHLZK0fZ6FYlxxyg+PbPSL4V7RpZqiVeQ8YFdzKf7gNr7GFYTo4X
+GvAs8mTO9b2VfPUVRowC7svj0Go78HwTKd3zEtB8ioZtjxsR1uYddpzZaEAs11Qajc4FkboMyrgY9kFpF2pdGFNG3y4x4s2t9K7FDGKkx2Db9CLq4rj/
WoEhF9PbhUyNW3EU0AsLxCRBwFow25SjZZ5Wa8WqgEK9AefGNodGTcCTsIi+7htTlwLggCkl1GdtBaQ==</value>
      </response>
      <response>
         <name>SM_AUTHDIRNAMESPACE</name>
         <value>LDAP:</value>
      </response>
      <response>
         <name>SM_AUTHTYPE</name>
         <value>Basic</value>
      </response>
      <response>
         <name>SMSSOZONE</name>
         <value>SM</value>
      </response>
      <response>
         <name>SM_USERDISABLEDSTATE</name>
         <value>0</value>
      </response>
   </authenticationResponses>
</changePasswordResponse>
```

Example: Change Password – REST JSON

1. Login Flow

Request:

URI of Login Request: http://authaz233.ca.com:88/authazws/AuthRestService/login/app1/sspapi/

```
{
    "userName": "user2",
    "password": "user2",
    "action": "GET"
}
```

Response:

```
{
   "message": "Authentication Failed",
   "resultCode": "LOGIN_FAILED",
   "authenticationResponses": {"response":    [
            {
         "name": "SM_AUTHREASON",
         "value": "20"
      },
            {
         "name": "SM_REALMOID",
         "value": "06-3b2fb84b-45d8-47ed-82ff-394402e0ad9b"
      },
            {
         "name": "user_mail",
```

```
         "value": "user2@ca.com"
      },
            {
         "name": "SM_REDIRECTURL",
         "value": "/siteminderagent/forms/smpwservices.fcc?SMENC=UTF-8&SMTOKEN=-SM-
%7bRC2%7dYsKz0bWGfSfsqlE9%2f78UkfkurQt73N0mUxigoa6iVQ4uXSyzRXtT5lSqTboB%2fvMoZyn1IRe1rSg
%3d&USERNAME=user2&SMAUTHREASON=20&SMAGENTNAME=-SM-kpF5Hvv2iTLWFGxu5BMM8zs
%2fHmT0mHY2uY4v6KbcZGGY5rFa80GpRki9HirHb9Aq&TARGET="
      },
            {

          "name": "SMTOKEN",
          "value": "lXERVnFxXFNdJavDiHtkQuwGMAp1ljkl1WxQN4fmnefTbBriUF3/
rcUSPzertp6YomFL2pWD2XuWI3Ev4Y4ELJrMOw6aOKYwci+j5lCfFVfWPdKIbHMPBvBWEdq9TQPlBsJ0bP7jr2spYN
+uSBEWX3Pzw5l89doSXYY+iPfVQpVfRIx/W3kb2wxoZo9NH1wS"

      },
            {
         "name": "SM_REALM",
         "value": "authaz_rlm"
      },
            {
         "name": "SMLOCALE",
         "value": "en-US"
      },
            {
         "name": "SM_TRANSACTIONID",
         "value": "000080fe000000008ab05b79f5bef264-1434-63ec7418-0768-02854823"
      },
            {
         "name": "SM_AUTHTYPE",
         "value": "Basic"
      },
            {
         "name": "SM_USERDISABLEDSTATE",
         "value": "16777216"
      }
   ]}
}
```


2. Change password flow

Request

URI of Change Password Request: http://hostname:port/authazws/AuthRestService/changepwd/appID/Resource

Example: http://authaz233.ca.com:88/authazws/AuthRestService/changepwd/app1/sspapi/


```
  {
    "newpassword":  "firewall@1",
    "password": "firewall",
    "userName": "user2",
    "action": "POST",
```

```
    "smtoken":"lXERVnFxXFNdJavDiHtkQuwGMAp1ljkl1WxQN4fmnefTbBriUF3/
rcUSPzertp6YomFL2pWD2XuWI3Ev4Y4ELJrMOw6aOKYwci+j5lCfFVfWPdKIbHMPBvBWEdq9TQPlBsJ0bP7jr2spYN
+uSBEWX3Pzw5l89doSXYY+iPfVQpVfRIx/W3kb2wxoZo9NH1wS"
}
```


Response


```
{
   "message": "Authentication Successful",
   "resultCode": "LOGIN_SUCCESS",
   "sessionToken": "E34LbZ1z1ICxG4DoC+PvgH4wLPCVahBh9CGVdOjPw+SW24/PFspksorD9dtbaSU
+oKoGP0ChG5STOZcvxN3BsV2lvnQcVhm5JoTfaJRloMOZZFvUJ2dRfa2ayuMEcY989ksKyAszNBfGrFx/lASkKBwmSJ0prvAI/
ewNIP+3a3NnMr6FDlqZ3Kfds6DXnUCVo9OXNmU1gtniou1ZPUOTAIQaySXfXg5tZ/QOUst8NfXnzvaEtb7WoSR/4KBWqs8W/S//
ERnbU2H86Gwk3Y1IRAG0eNOQCsl1oT65nK9AEWSdBnkfnvPgB3b4RONAUsIBpJeCUonW/BDsuz6XzoZ961w846jecV13e6Pt79kTwbx8nQS
+vnYK/Rety7IG+DTiQIVcJF2BFvUCK13XyyZAQh1KZMw46vDKP9lT62+yiWDbEb7+bpVlIwfp//
EpkgDtOms1P/uvAs0P8crrlvmoCQ1B1MuEXcEDmLqxf4JosB3JTGKPL/uUB1xj7ofTmvnMPlXIJJhWvz
+H/0HSA9bL83Ri6NuGouchDhs2lS9Uiu2I2YVbTHQACa1K7erewUWDmRIIpwDTM4P+wkA
+JneW17Dg6xtqLaklAA0ziNbpfX1aePplVr9nh3mSvwryHNKtA/MhWxXgy8B
+CuiiD2tSaEZp91vSzGpq34jQlp1ldtOK9Lu4NykRSO1wt5WSMcq6YjJhZR/g2DtjhRryiaoLQ6B1sTCLykl
+DGiKQAc9YaOurhabrblfYZ4AxHzsTP3rZhQQTZZZNqNaZXvFJEENxUdx8k3L9iApwhPVyBDzWm9s+tR/2uQ9dX5fsjPYq/
cLvBTYqKaE5ucqVNwG7OZxpCVnR1MaKXsiHIocfHOLulEY4BuPnD/LYfv13i2Rv63zYY1De0lGQ/xx7JfbTpZAi4Xk+S4tlI/
Kf8R4WbxXnTHcGmOt6gWrS/7ChLyX2BN8/rVRc4ify/s6WoteZvNH70NTmr8jesCFTj+Zs1VOtaIiFWi+PNotHrB02Cn+06r/
P02eWuOjMsk2iFQo9Tioa/m/iK8p5H5VEW6ThNVW/raAw50X3D5ozxwLsTTkTVUy",
   "authenticationResponses": {"response":    [
            {
         "name": "SM_AUTHREASON",
         "value": "23"
      },
            {
         "name": "SM_SERVERIDENTITYSPEC",
         "value": ""
      },
            {
         "name": "SM_USERDN",
         "value": "uid=user1,ou=People,o=security.com"
      },
            {
         "name": "SM_AUTHDIROID",
         "value": "0e-3ced2af8-51e6-4a28-9c15-81bc89663123"
      },
            {
         "name": "SM_REALMOID",
         "value": "06-3b2fb84b-45d8-47ed-82ff-394402e0ad9b"
      },
            {
         "name": "SM_SERVERSESSIONID",
         "value": "7Rd87LkhKAhXQ54tACPRRejevC4="
      },
            {
         "name": "user_mail",
         "value": "user1@broadcom.com"
      },
            {
         "name": "SM_SESSIONDRIFT",
```

```
         "value": "0"
      },
            {
         "name": "SM_REDIRECTURL",
         "value": "/siteminderagent/forms/smpwservices.fcc?SMENC=UTF-8&SMTOKEN=-SM-
%7bRC2%7d9vEN1h27iYNra%2bOLpYLPN20ZovTrwHdHdUTz6b52xXr7wu6p1UXvg%2fSyTmA6l5gY8KNZiV5fsTw
%3d&USERNAME=user1&SMAUTHREASON=23&SMAGENTNAME=-SM-kpF5Hvv2iTLWFGxu5BMM8zs
%2fHmT0mHY2uY4v6KbcZGGY5rFa80GpRki9HirHb9Aq&TARGET="
      },
            {
         "name": "SM_UNIVERSALID",
         "value": "user1"
      },
            {
         "name": "SM_AUTHDIRNAME",
         "value": "ud"
      },
            {
         "name": "SM_REALM",
         "value": "authaz_rlm"
      },
            {
         "name": "SM_USER",
         "value": "user1"
      },
            {
         "name": "SMLOCALE",
         "value": "en-US"
      },
            {
         "name": "SM_TRANSACTIONID",
         "value": "000080fe000000008ab05b79f5bef264-1434-63f85618-1804-01ad1649"
      },
            {
         "name": "SM_AUTHDIRSERVER",
         "value": "example.net:20001"
      },
            {
         "name": "SM_TIMETOEXPIRE",
         "value": "7200"
      },
            {
         "name": "SM_SERVERSESSIONSPEC",
         "value": "He4sVHTD/pu7O3JrvcN1/OuaBqHH0KAvagjGUq7KdoEXG4rR+bazFPt/
OLA5uHQDgL2Qrwi4mRxS1RIODo8Fd7IrY4WVDzUQyGVaqGB6y3agd+2vpRcJxShLMoSfG5gXkCLywAW0e0IHsuk3TD/
N8uc0NVDZlFYmS4f60sQD6qW70iUaJOEQITOmyQGhfNH9pnPwtf7FTf8d7WyvnJEwAZALjgd8x8zz0nNeavtKEeigurydD410DhbRucbe5TFL0FLJjw67/
KNIzur/1FWAgYH0odC8JQI8tKFtW+86yHNllfMhS/MAQERTvwSQPqkV94W3X32iuJ8fAYqHHNLjNANJmTugKAoNplMizUSoBHkETD
+pBqIJEA3GBFZgg7R4tbgAwiDT0ZNRW0VD3k3eVyfaTQ8/dy4vR5MR+UltglCR5+v2i9+6DA=="
      },
            {
         "name": "SM_AUTHDIRNAMESPACE",
         "value": "LDAP:"
      },
```

```
            {
          "name": "SM_AUTHTYPE",
          "value": "Basic"
       },
            {
          "name": "SMSSOZONE",
          "value": "SM"
       },
            {
          "name": "SM_USERDISABLEDSTATE",
          "value": "0"
       }
    ]}
 }
```

Use an Existing Virtual Host for Authentication and Authorization Web Services

The SiteMinder authorization and authentication web services require a dedicated virtual host to function along with an ACO for the web services. From 12.8.0700.2947 build, an existing virtual host, including the default virtual host, can be used to enable the web services instead of a dedicated virtual host.

The configuration requires changes in Policy Server and Access Gateway.

Configuration of Policy Server

Configure the ACO of the existing virtual host that you want to use for the web services, to support the ACO parameters that are required for the web services.

Follow these steps:

1. Log in to Administrative UI.
2. Navigate to **Infrastructure**, **Agent**, and **Agent Configuration Objects**.
3. Modify the ACO of the existing virtual host that you want to use for the web services.
4. Add the following ACO parameters in the ACO of the existing virtual host:
   - **AgentName**
     Defines the names of the web agent that protects a resource. You can define one or more web agents where each web agent protects an application. Enter a multi-value pair in the following format:
     agent_name,*appID*
     where,
     - ***agent_name*** defines the name of the web agent that protects a resource.
     - ***appID*** defines the reference name of the web agent that was specified in ***agent_name*** or of the application that is protected by the web agent. SiteMinder uses this value in the web services requests, thus protecting the agent name from the users.
   - **enableauth**
     Specifies the status of the authentication web service. Set the value to yes to use the authentication web service, set the value to yes.
   - **enableaz**
     Specifies the status of the authorization web service. Set the value to yes to use the authorization web service.
   - **RequireAgentEnforcement**
     Specifies whether the web services must be protected by a SiteMinder agent. Set this value to **yes** and protect the web services by a SiteMinder Agent. If you set the value to **yes** and the web services are not protected, the requests to web services fail.

5. Click **Save**.

Configuration of Access Gateway

Once the ACO of the existing virtual host that you want to use for the web services is configured, select the virtual host to instruct Access Gateway to use it.

Configure using Access Gateway Administrative UI

Follow these steps:

1. Log in to Access Gateway Administrative UI.
2. Navigate to **Web Services**, **Auth and Az Web Services**.
3. Select **Enable Authentication and Authorization Web Services**.
4. Select the existing virtual hostname from **Virtual Hostname**.
   Important! If a dedicated virtual host is already configured in your environment, the selection of an existing virtual host replaces the configured dedicated virtual host when the change is saved.
5. Click **Save**.

Configure using in server.conf

Follow these steps:

1. Add the **virtualhostname="<*virtual-hostname*>"** line in the **<Context name="Authentication/Authorization web services">** section.
   Example:
   ```
   <Context name="Authentication/Authorization web services">
       docBase="CA_AuthAZ"
       path="authazws"
       enable="yes"
       virtualhostname="default"
   </Context>
   ```
2. If a dedicated virtual host is already configured in your environment, delete the related information from the **<VirtualHost name="WebServicesAgentVirtualHost">** section.
3. Save the changes.

Support for SiteMinder Password Policies in IDSP

When SiteMinder delegates the authentication of a user to Symantec Identity Security Platform, IDSP can leverage the password policies that are configured in SiteMinder to authenticate the user if PWD is one of the obligations.

SiteMinder supports the use of password policies to define rules and restrictions that govern the expiration, composition, and usage of a password. SiteMinder invokes a configured password policy when a user attempts to access a protected resource. The policy configuration is used to evaluate the credentials of the user. For more information, see Password Services and Policies.

To support this feature in IDSP from 12.8.0700.2947, SiteMinder must be configured as a custom provider in IDSP with a specific set of configurations.

Assume that the feature is configured in IDSP. When IDSP has to authenticate a user where PWD is one of the obligations, IDSP performs the following tasks:

1. Invokes the Login API of the authentication and authorization web services of SiteMinder for the user disambiguation.
2. Invokes the Change Password API of the authentication and authorization web services of SiteMinder, if the password of the user has to be changed per the password policy.
3. Synchronizes changes such as incorrect or maximum password attempts in SiteMinder.

Configuration in SiteMinder to Synchronize Password Attempt Changes

Configure the following new fields in SiteMinder:

- **Skip Password History Update on Successful Authentications** in the Basic authentication scheme template **(Valid for PWD obligation)**
  When PWD is one of the obligations, you must ensure that a password policy is updated in SiteMinder *only* after fulfilling all the obligations but not just after fulfilling the PWD obligation. Select this field to skip updating the password history of a user when the PWD obligation is successfully fulfilled. This field must be selected if IDSP has to leverage the password policies of SiteMinder.
  – If this option is selected, the password blob is not updated though the PWD obligation is fulfilled. To update the password blob when all the obligations are complete, select **Update Password History Using a Custom Claim**.
  – If this option is not selected, the password blob is updated in SiteMinder once the PWD obligation is fulfilled, which results in an incorrect flow in the integration.
- **Update Password History Using a Custom Claim** in the JWT authentication scheme template **(Valid for all obligations)**
  Select to update the password history of a user using the custom claim, **SM_RELUSERLOGINFAILURESCOUNT**.
  – If all the obligations of IDSP are successfully fulfilled, select this field and send **0** as the value in **SM_RELUSERLOGINFAILURESCOUNT**.
  – If any obligation fails, select this field and send the failure count in increments in **SM_RELUSERLOGINFAILURESCOUNT**. For example, send 1 when the first obligation fails. Then send 2 when the authentication fails, and so on.
  – If this field is not selected, any updates to the password policy such as failed password attempts are not updated in SiteMinder.

Troubleshooting

Logs Locations

- Policy Server logs are logged in the **smtracedefault.log** file. You can use **SmAuthServiceJWT** as a keyword for searching logs in the file.
- Access Gateway logs are logged in **FWSTrace.log** file, if enabled. Enable logging using the LoggerConfig.properties file located at *accessgateway_installation_home*`/secure-proxy/Tomcat/webapps/affwebservices/WEB-INF/classes` . Set **LoggingOn=Y** in the file.

For transaction logs, Access Gateway generates a transaction ID and that ID is used for logging in FWSTrace.log and smtracedefault.log files accordingly. Also, the same ID is used as a suffix for Symantec Identity Security Platform related flow transaction ID. You can use the transaction ID to find all the logs related to a specific transaction for troubleshooting.

For example: If Policy Server generates transaction ID **vbbiujoi1u38**, this value is used in FWSTrace.log and smtracedefault.log as follows:

- **SiteMinder flow logs**: vbbiujoi1u38
- **IDSP flow logs for SiteMinder**: vbbiujoi1u38_bfenfieuw78812u3912 where *bfenfieuw78812u3912* is the transaction ID generated by IDSP for user authentication

General Issues and Solutions

Issue: HTTP 500 error is displayed with transaction ID

Workaround: Search for the **[Login failure [CHECKPOINT = SSO_LOGINFAILURE_RSP]]** string in the FWSTrace.log file. If this string is found, search for the the following strings in the smdefaultrace.log file:

- **Certificate with alias <alias name> not found,cannot continue**. If this string is present, look for alias name mentioned in tool config.properties or authentication scheme to understand which certificate is missing. If the alias name mentioned is part of Verification Cert Alias, then public certificate of IDSP is not imported.
- **Certificate with alias <alias name> is expired**. If this string is present, update the certificate that has the alias name mentioned in error.

Issue: IDSP prompts for user name after SiteMinder completes primary authentication

Solution: User is not present in the IDSP user store.

Issue: Access Gateway returns HTTP 503 error in IPv6 environment

Solution: Set the worker.ajp13.host parameter to ::1 in the server.conf file.

```
worker.ajp13.host=::1
```

Issue: Authenticatin Hub returned the AUTH_DENIED error message

IDSP log displays the following error message:

```
@timestamp:Jul 1, 2020 @ 10:10:51.240 stream:stdout level:info type:log timestamp:Jul 1, 2020 @ 10:10:51.000
 thread:qtp1688526221-22 msg:doPolicyEvaluate() returned null, returning AUTH_DENIED api:/auth/v1/authenticate
```

Solution: Verify the Authn policy configuration and then verify the authentication scheme configuration. Ensure that the ACR values configured in the authentication scheme match the values in Authn policy.

General Issues During Transactions

- Invalid client_id
  Cause: Client ID mentioned in authentication scheme is incorrect
- Auth Denied. Please try again
  Causes: ACR did not match or Authorization endpoint is invalid
- Server Error. The server was unable to process your request.
  Cause: Unsupported authentication type is defined
- redirect_uri is missing or empty
  Cause: Value for redirect_uri in authentication scheme is empty
- redirect_uri didn't match with the configured redirect_uri's
  Cause: The redirect_uri value mentioned in authentication scheme does not match the value in IDSP application

Symantec Identity Security Platform Training Videos on Support Site

The Identity Management Security (IMS) division of Broadcom offers free training videos to help you get the most out of our IMS products.

To access the training resources, follow these steps:

1. Go to support.broadcom.com and log in (or register for a free account) using your existing Broadcom credentials or corporate email address.
2. On the dashboard, scroll to the bottom right to find the **Education** section.
3. Click **learning@broadcom** to access the training resources.

List of Available Training Videos

The following list of IDSP training videos are available on IMS Software Academy:

- **Symantec IDSP Basics**: The modules in this course describe how to integrate SiteMinder with IDSP for the following use cases:
  a. Demonstrating theIDSP's ability to extend SiteMinder Protected Applications with Modern Authentication Mechanisms.
  b. Demonstrating the Configuration and Setup of an External Risk Provider utilizing the IDSP's Service Provider Interface (SPI).
  – Auth Hub - SiteMinder Extension Overview
  – Auth Hub - Service Provider Interface
- **Leveraging IDSP as OIDC Provider**: This module will help understand the process of integrating IDSP as an OIDC Provider to Spring Boot Web Application using Spring Security libraries for enabling multi-factor user authentication.
  – Leveraging IDSP as OIDC Provider

Let us know what you would like to learn more about in IDSP.

End User License Agreement (EULA)

Displays the End User License Agreement (EULA)

CA End User License Agreement (the "Agreement") for the CA software product that is being installed as well as the associated documentation and any SDK, as defined below, included within the product ("the Product").

Carefully read the following terms and conditions regarding your use of the Product before installing and using the Product. Throughout this Agreement, you will be referred to as "You" or "Licensee."

By installing or using the Product, or by selecting the "I accept the terms of the License Agreement" option button, and then clicking on the "Next" button, you are

(I) Representing that you are not a minor, and have full legal capacity and have the authority to bind yourself and your employer, as applicable, to the terms of this Agreement;
(II) Consenting on behalf of yourself and/or as an authorized representative of your employer, as applicable, to be bound by this Agreement.

By selecting the "I do NOT accept the terms of the License Agreement" option button, and then clicking on the "Cancel" button, the installation process will cease.

1. CA, Inc. (or the CA entity respectively identified after Section 15 below for the country / countries in which the Product is being supplied), (“CA”) provides Licensee with one (1) copy of the Product, for use in accordance with such (a) quantity and (b) CA published criteria for measuring the usage of the Product (such as, but not limited to, MIPS, CPUs, tiers, servers, or users), designated as the authorized use limitation ("Authorized Use Limitation") on any Order Form (defined below) or CD sleeve included within the Product box. CA licenses the Product to Licensee on a limited, non-exclusive, non-transferable basis only for internal business use during the Term and other terms and conditions of (a) any CA Order Form or Registration Form which has been signed or otherwise contracted between Licensee and a CA affiliate; or (b) a License Program Certificate which is provided by CA to Licensee, as applicable, referencing and incorporating the terms of this Agreement (each hereafter referred to as the "Order Form").

2. If the Product is an alpha or beta version of the program, hereinafter referred to as the "beta program" or "beta version" and not generally available to date, CA does not guarantee that the generally available release will be identical to the beta program or that the generally available release will not require reinstallation. Licensee agrees that if it registers for support or if otherwise required by CA, Licensee shall provide CA with specific information concerning Licensee’s experiences with the operation of the Product. Licensee agrees and acknowledges that the beta version of the Product (a) is to be used only for testing purposes and not to perform any production activities unless CA shall have otherwise approved in writing and (b) has not been tested or debugged and is experimental and that the documentation may be in draft form and will, in many cases, be incomplete. Licensee agrees that CA makes no representations regarding the completeness, accuracy or Licensee’s use or operation of the beta version of the Product. BETA PRODUCTS ARE PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR REPRESENTATIONS OF TITLE OR NON-INFRINGEMENT. If Licensee is also a tester of the beta version of the Product (as defined by the beta testing or pre-release testing agreement (“Beta Testing Agreement”) that was agreed to by Licensee during the registration process before obtaining the beta version of the Product), Licensee agrees that the terms of this Agreement are in addition to, and do not supersede, the terms of the Beta Testing Agreement.

3. If the Product is being licensed on a trial or evaluation basis, Licensee agrees to use the Product solely for evaluation purposes, in accordance with the usage restrictions set forth in Section 1, for a thirty-day evaluation period unless a different period is otherwise noted (the "Trial Period"). At the end of the Trial Period, Licensee’s right to use the Product automatically expires and Licensee agrees to de-install the Product and return to CA all copies or partial copies of the Product or certify to CA in writing that all copies or partial copies of the Product have been deleted from Licensee’s computer libraries and/or storage devices and destroyed. If Licensee desires to continue its use of the Product beyond the Trial Period, Licensee may contact CA or a CA affiliate to acquire a license to the Product for

the applicable fee. LICENSEE'S USE OF THE PRODUCT DURING THE TRIAL PERIOD IS ON AN "AS IS" BASIS WITHOUT ANY WARRANTY, AND CA DISCLAIMS ALL WARRANTIES INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AS WELL AS ANY EXPRESS WARRANTIES PROVIDED ELSEWHERE IN THIS AGREEMENT.

4. If the Product includes a Software Development Kit ("SDK"), the terms and conditions of this paragraph apply solely for the use of the SDK. The SDK may include software, APIs and associated documentation. The SDK is provided solely for Licensee's internal use to develop software that enables the integration of third party software or hardware with the Product, or to develop software that functions with the Product, such as an agent. Licensee's use of the SDK is restricted solely to enhance Licensee's internal use of the Product. No distribution rights of any kind are granted to Licensee regarding the Product. In addition to the limitations on use set forth in Section 8, below, Licensee may not reproduce, disclose, market, or distribute the SDK or the documentation or any applications containing any executable versions of the SDK to third parties, on the internet, or use such executables in excess of the applicable Authorized Use Limitation. If there is a conflict between the terms of this section and the terms of any other section in this Agreement, the terms of this section will prevail solely with respect to the use of the SDK.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, AND NOTWITHSTANDING ANYTHING CONTAINED HEREIN TO THE CONTRARY, THE SDK IS PROVIDED AND LICENSED "AS IS" WITHOUT WARRANTY OF ANY KIND.

5. Payment of the fees specified on the Order Form or as agreed between Licensee and an authorized reseller of CA or any of the CA affiliates, shall entitle Licensee to use the Product for the term specified on the Order Form (the "Term"), which use may include the right to receive maintenance services therefore for the period set forth on the Order Form. All fees payable hereunder shall be payable as stated in the Order Form, or if not stated, shall be payable net thirty (30) days from the CA invoice date. Licensee will install each new release of the Product delivered to Licensee. After the initial Term, continued usage and/or maintenance of the Product as provided herein shall be subject to the payment by Licensee of the fees described on the Order Form. If You are acquiring licenses of the Product from an authorized CA reseller, the terms of this Agreement governing payments, pricing and discounts shall not apply as such terms are between You and your chosen CA reseller. Any terms that may appear on a Licensee purchase order (including without limitation pre-printed terms), or as part of Licensee's order with an authorized CA reseller, that conflict or vary from the terms and conditions of this Agreement shall not apply to the Product and shall be deemed null and void. Notwithstanding the foregoing, with respect to any Product that relies on continuous content updates, such as signature files and security updates, Licensee shall be entitled to such content updates for a period of one (1) year from the effective date of the license.

6. If maintenance is provided by CA or a CA affiliate, it shall be renewed annually as specified in the Order form. All fees are net of applicable taxes. Licensee agrees to pay any tariffs, duties or taxes imposed or levied by any government or governmental agency including, without limitation, federal, state and local, sales, use, value added and personal property taxes, (other than franchise and income taxes for which CA is responsible) upon a presentation of invoices by CA or a CA affiliate, as applicable. Any claimed exemption from such tariffs, duties or taxes must be supported by proper documentary evidence delivered to CA. Any invoice which is unpaid by Licensee when due shall be subject to an interest charge equal to the lower of 1.5% per month or the highest applicable legal rate.

7. Licensee may install and deploy the Product in the territory specified in the Order Form up to the Authorized Use Limitation. Licensee may permit its Authorized End Users access to the Product for Licensee's and its Affiliates' internal business wherever located, provided that Licensee hereby expressly agrees that a breach by an Authorized End User shall be considered to be a breach by and the responsibility of Licensee. Licensee may relocate the Product to a new Licensee location within the territory specified in the Order Form upon prior written notice. For purposes hereof, "Authorized End Users" means Licensee, its Affiliates and their employees and independent contractors (but excluding any outsourcer, facilities management providers, managed service provider, or application service provider) that are bound by terms and conditions no less restrictive than those contained herein and are acting on behalf of Licensee and not a third party; "Affiliate" with respect to Licensee means any legal entity in which the Licensee directly or indirectly Controls; and "Control" means ownership or control of greater than 50% of an entity's shares or control of the board of such entity by force of law or contract, or the equivalent. If Licensee desires to use the Product beyond such restrictions, it

shall notify CA or the CA affiliate the Licensee has procured the Product from, and Licensee will be invoiced for and shall pay the applicable fees for such expanded use.

8. The Product, including any source or object code that may be provided to Licensee hereunder, as well as documentation, appearance, structure and organization, is the proprietary property of CA and/or its licensors, if any, and may be protected by copyright, patent, trademark, trade secret and/or other laws. Title to the Product, or any copy, modification, translation, partial copy, compilation, derivative work or merged portion of any applicable SDK, shall at all times remain with CA and/or its licensors. Licensee agrees that CA may use any feedback provided by Licensee related to the Product for any CA business purpose, without requiring consent including reproduction and preparation of derivative works based upon such feedback, as well as distribution of such derivative works. Usage rights respecting the Product may not be exchanged for any other CA product. The Product is licensed as a single product. Its component parts may not be separated for use. Licensee and its Authorized End Users will keep the Product and the terms of this license strictly confidential and use its best efforts to prevent and protect the Product from unauthorized disclosure or use. Licensee may not (i) disclose, de-compile, disassemble nor otherwise reverse engineer the Product except to the extent the foregoing restriction is expressly prohibited under applicable law; (ii) create any derivative works based on the Product; (iii) use the Product to provide facilities management, outsourcing, service bureau, hosted services, cloud services, on demand services or like activity whereby Licensee, without a CA license authorizing such purpose, operates or uses the Product for the benefit of a third party; or (iv) permit the use of the Product by any third party, except as authorized by CA in writing. Licensee shall not release the results of any benchmark testing of the Product to any third party without the prior written consent of CA. Licensee will not transfer, assign, rent, lease, use, copy or modify the Product, in whole or in part, or permit others to do any of the foregoing with regard to the Product without CA's prior written consent, except to the extent the foregoing restriction is expressly prohibited under applicable law. Licensee will not remove any proprietary markings of CA or its licensors. Licensee may make a reasonable number of copies of the Product for disaster recovery "cold standby", backup and archival purposes; provided that use of such copies is limited to testing Licensee's disaster recovery procedures and effectiveness and as is necessary during any reasonable period subsequent to the occurrence of an actual disaster during which Licensee cannot operate the Product. If this license terminates for any reason, Licensee shall certify to CA in writing that all copies and partial copies of the Product have been deleted from all computers and storage devices and are returned to CA or destroyed and are no longer in use. Licensee acknowledges that the Product is subject to control under European and U.S. law, including the Export Administration Regulations (15 CFR 730-774) and agrees to comply with all applicable import and export laws and regulations. Licensee agrees that the Product will not be exported, reexported or transferred in violation of U.S. law or used for any purpose connected with chemical, biological or nuclear weapons or missile applications, nor be transferred or resold, if Licensee has knowledge or reason to know that the Product is intended or likely to be used for such purpose. The Product and any accompanying documentation have been developed entirely at private expense and are "commercial item(s)" and "commercial computer software" as those terms are defined in Federal Acquisition Regulation Subpart 2.101 "Definitions." The Product is exempt from disclosure under the Freedom of Information Act (FOIA), 5 U.S.C. §552(b) under one or more exemptions to that Act (or a similar U.S. state statute, as applicable). Any Product previously delivered to You may not be delivered again. Any Product not previously delivered to Licensee will be delivered to Licensee or its chosen reseller following receipt by CA of an acceptable order. The Product shall be delivered either by electronic delivery ("ESD") or, if CA or a CA affiliate respectively requires delivery in tangible media, CPT, as defined in INCOTERMS 2010, from CA's or such CA affiliate's shipping point. CA or the CA affiliate from which Licensee is procuring the Product agrees to be responsible for all customs duties and clearances and title to any CA hardware if included will pass upon point of delivery to carrier at CA's or such CA affiliate's shipping location. In the event of electronic delivery, no tangible personal property will be delivered. Such electronic delivery may not automatically provide for an exemption from applicable sales or use tax. Any operating system identified as "Generic" or "GA" denotes such operating systems for which the Product is made generally available by CA in accordance with CA current published specifications.. Acceptance is waived and deemed to have occurred at the earliest of point of physical shipment or delivery of keys/access codes for electronic delivery. CA Inc. is the manufacturer of the Product.CA reserves the right, on notice to You, to conduct an audit remotely or onsite of Licensee and/or Your Affiliates facilities to verify compliance by Licensee and its Authorized End Users with the terms of this Agreement. CA agrees that such audit shall be conducted during regular business hours at Your offices and CA shall endeavor to conduct such audit so as not to interfere unreasonably with Your activities and/or use an independent third party to conduct the audit subject to terms of non-disclosure if required. This Agreement shall be governed by and interpreted in accordance with the laws of the State

of New York, without regard to its choice of law provisions, and any action arising under or relating to the Agreement shall lie within the exclusive jurisdiction of the State and Federal Courts located in Suffolk County, New York.

9. CA warrants that it can enter into this Agreement and that it will indemnify Licensee, or, at its option, settle any third party claim that CA is not so authorized or that Licensee's use of the Product as authorized hereby infringes any United States patent or copyright within the jurisdictions where Licensee is authorized to use the Product at the time of delivery. CA also warrants that its distributed Product will operate materially in accordance with its published specifications set forth within the documentation for a period of ninety (90) days after delivery of the Product to Licensee, provided that CA's only responsibility will be to use reasonable efforts, consistent with industry standards, to cure any defect. If, within a reasonable time after receiving Licensee's written notice of breach of either of the above warranties, CA is unable to cause the Product to operate (a) without infringing a third party's intellectual property rights, or (b) materially in accordance with CA's written specifications, then CA may terminate the license and provide or arrange for a pro-rata refund to Licensee or its authorized CA reseller of the license fees and or the support and maintenance fees paid. In the event of such termination, the pro-rata refund shall be calculated on (i) the number of months left remaining on the Term of the applicable Order Form or (ii) if the Product is licensed under a perpetual license, using (only for purposes of a refund calculation) an amortization schedule of three (3) years. The warranties set forth in this Section do not apply to beta, trial, evaluation or demonstration versions of the Product, or to Software Development Kits. CA shall have no liability to indemnify or to remedy a warranty claim: (i) in the event the allegation of infringement or warranty claim is a result of a modification of the Product except a modification by CA, (ii) if the Product is not being used in accordance with CA's specifications, related documentation and guidelines, (iii) if the alleged infringement or warranty claim would be avoided or otherwise eliminated by the use of a CA published update or patch, (iv) if the alleged infringement or warranty claim is a result of use of the Product in combination with any third party product, or (v) if the applicable fees due for the Product have not been paid or Licensee is otherwise in breach of this Agreement. The indemnifications contained herein shall not apply and CA shall have no liability in relation to any Product produced by CA at the specific direction of Licensee. THE FOREGOING PROVISIONS STATE THE ENTIRE LIABILITY AND OBLIGATIONS OF CA REGARDING CLAIMS OF INFRINGEMENT, AND THE EXCLUSIVE REMEDY AVAILABLE TO LICENSEE WITH RESPECT TO ANY ACTUAL OR ALLEGED INFRINGEMENT OR MISAPPROPRIATION OF ANY INTELLECTUAL PROPERTY OR OTHER PROPRIETARY RIGHTS.

10. EXCEPT AS SET FORTH ABOVE, TO THE FULL EXTENT PERMITTED BY APPLICABLE LAW:

(I) NO OTHER WARRANTIES, WHETHER EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE MADE BY CA;
(II) IN NO EVENT WILL CA BE LIABLE TO LICENSEE OR ANY OTHER PARTY FOR ANY CLAIM FOR LOSS, INCLUDING TIME, MONEY, GOODWILL, AND CONSEQUENTIAL OR INDIRECT DAMAGES, WHICH MAY ARISE FROM THE USE, OPERATION OR MODIFICATION OF THE PRODUCT, EVEN IF CA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN THE EVENT THAT THE ABOVE LIABILITY LIMITATION IS FOUND TO BE INVALID UNDER APPLICABLE LAW, THEN CA'S LIABILITY FOR SUCH CLAIM SHALL BE LIMITED TO THE AMOUNT OF THE LICENSE FEE ACTUALLY PAID FOR THE PRODUCT BY LICENSEE. NO THIRD PARTY, INCLUDING AGENTS, DISTRIBUTORS, OR AUTHORIZED CA RESELLERS IS AUTHORIZED TO MODIFY ANY OF THE ABOVE WARRANTIES OR MAKE ANY ADDITIONAL WARRANTIES ON BEHALF OF CA. CA DOES NOT WARRANT THAT THE PRODUCT WILL MEET LICENSEE'S REQUIREMENTS OR THAT USE OF THE PRODUCT WILL BE UNINTERRUPTED OR ERROR FREE.

11. Licensee may assign this Agreement only if Licensee complies with CA's then prevailing policies respecting assignment of licenses, which includes a requirement that the scope of use of the Product not be expanded beyond the business of Licensee and the business of Licensee's majority-owned subsidiaries.

12. If Licensee breaches any term of this Agreement or if Licensee becomes insolvent or if bankruptcy or receivership proceedings are initiated by or against Licensee, CA shall have the right to withhold its own performance hereunder and/or to terminate this Agreement immediately and, in addition to all other rights of CA, all amounts due or to become due hereunder, if any, will immediately be due and payable to CA or the CA affiliate the Licensee was procuring the Product from.

13. If Licensee fails to pay the applicable maintenance fee, then Licensee may reinstate maintenance thereafter by paying to CA or the respective CA affiliate a fee equal to 150% of CA's then prevailing maintenance fee for each year for which the maintenance fee has not been paid.

14. If a court holds that any provision of this Agreement to be illegal, invalid or unenforceable, the remaining provisions shall remain in full force and effect. No waiver of any breach of this Agreement shall be a waiver of any other breach, and no waiver shall be effective unless made in writing and signed by an authorized representative of the waiving party. Any questions concerning this Agreement should be referred to CA, Inc., One CA Plaza, Islandia, NY 11749, Attention: Worldwide Law Department.

15. In the event Licensee acquires a license for the Product outside of the United States, the following Sections will apply to the use of the Product:

Notwithstanding the terms of the last sentence of Section 8, the laws of the country in which Licensee acquires a license for the Product shall govern this Agreement, except as otherwise provided below:

Europe, Middle East and Africa

1. In EMEA the CA entity that is the licensor is CA Europe Sàrl ("CA Europe"). CA Europe Sàrl is the licensor for Products which have been made available to Licensee by way of license from CA Europe Sàrl through a local CA subsidiary in EMEA or through an authorized CA reseller. The CA support and maintenance, if any, is being provided by a local CA subsidiary or by an authorized CA reseller.

2. EMEA means Europe, Middle East and Africa.

3. Section 9, third sentence and fifth sentence shall only apply to third party rights infringement. Section 9, second sentence is deleted and replaced with:
"CA Europe also warrants that the Product will operate materially in accordance with the applicable specifications set forth within the documentation of the Product subject always to Licensee's compliance with the terms of this Agreement. If CA Europe has breached this warranty Licensee's remedy is for CA Europe, in consultation with Licensee, to either (i) use reasonable efforts consistent with industry standards to cure the defect, or (ii) replace the Product with one that materially complies with the documentation. If the defect cannot be cured within a reasonable period of time or if the rectification of the defect or replacement has finally failed, Licensee shall have (i) in case of a subscription license the right to reasonably reduce the fees agreed and/or terminate immediately for cause, if the legal or statutory requirements are met; (ii) in case of a perpetual license, at its option, the right (1) to rescind or reduce the fees agreed in the applicable transaction document and (2) claim damages or to claim reimbursement of futile expenditures. The right to claim damages or futile expenditures shall be subject to the limitations of liability set forth below in section 10. In case of a perpetual license the warranty claims stated herein shall become time-barred within ninety (90) days after delivery of the Product. Warranty remedies are conditioned upon (i) any error or defect complained of is reasonably reproducible by CA Europe, (ii) the Product is not modified and is being used in accordance with the documentation, and (iii) the breach is not attributable in whole or in part to any non-CA product(s) or service(s). The above warranties are the sole warranties provided by CA Europe. No other warranties, including that the Product is error free, whether express or implied, including, without limitation, the implied warranties of satisfactory quality, non-infringement, or suitability and/or the warranty of fitness for a particular purpose are made by CA Europe or its suppliers. If Licensee claims under this warranty section, Licensee is not entitled or eligible to seek the same warranty remedies from any other CA affiliate."

4. Section 10 is deleted and replaced with:

"10.1 CA Europe's liability shall, regardless of the reason for the liability, be unlimited in cases of death or bodily injury or injury of health and damages caused by gross negligence or willful default of CA Europe or the grossly negligent or willful default of CA Europe's legal representatives or persons whom CA Europe occupies with the performance of its contractual obligations and in cases of liability under the Product Liability Act ("Produktehaftpflichtgesetz").
10.2 In case of slight negligence CA Europe shall, regardless of the reason for the liability, only be liable, if CA Europe violates an obligation, which is essential for the execution of the Agreement and in the fulfillment of which the other party regularly trusts. In this case, CA Europe's liability to Licensee will be limited to damages which have been foreseeable and which can typically arise in connection with this Agreement.

10.3 Further to the above CA Europe's liability to the Licensee for indirect, special and consequential damages (including, without limitation, loss of profits, loss of business, loss of opportunity or loss of goodwill) shall be limited to damages which have been foreseeable and which can typically arise in connection with this Agreement.
10.4 It is the parties' understanding that the foreseeable damages that can typically arise in connection with the licenses granted in this Agreement in the meaning of sections 10.2 and 10.3 above shall be limited to a maximum of the fees paid or owed for the then current initial or renewal Term for which the Licensee has procured the Product.
10.5 The liability for loss of data shall be limited to the typical recovery efforts in the case of regular and adequate data back-up.
10.6 The remedies provided in this Agreement are the exclusive remedies of the parties."

5. The following sections are added to this Agreement:

"Force Majeure. Except for payment obligations and obligations pertaining to non-disclosure, notwithstanding any contrary provision in this Agreement, neither Party will be liable for any action taken, or any failure to take any action required to be taken, in the event and to the extent that the taking of such action or such failure arises out of causes beyond a party's control, including, without limitation, war, civil commotion, act of God, strike or other stoppage (whether partial or total) of labor, any law, decree, regulation or order of any government or governmental body (including any court or tribunal)."

"Licensee Data. If Licensee transfers any personal data to CA Europe as a requirement pursuant to any Product, then Licensee represents that (i) it is duly authorized to provide personal data to CA Europe and it does so lawfully in compliance with relevant legislation, (ii) CA Europe and any entity within the CA group of companies (each a "CA entity") or its subcontractors can process such data for the purposes of performing its obligations and (iii) CA Europe may disclose such data to any CA entity and its subcontractors for this purpose and may transfer such data to countries outside of the country of origin. CA, Inc. is Safe Harbour certified and the CA entities have committed to comply with relevant data protection/privacy legislation."

6. The laws of Switzerland (excluding its conflict of laws provisions) shall govern the construction and enforceability of this Agreement. The parties agree that any action arising under or relating to this Agreement shall lie within the exclusive jurisdiction of the Swiss courts located in Zürich. The United Nations Convention on Contracts for the International Sale of Goods will not apply to this Agreement.

7. Any questions concerning this Agreement for EMEA should be referred to CA Europe Sàrl located at Building A, Lake Geneva Centre, Route de la Longeraie 9, 1110 Morges, Switzerland, Attention: Worldwide Law Department.

Argentina
The CA subsidiary that is the licensor is CA Argentina S.A.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of Argentina. Any dispute hereunder shall be determined by the Tribunales de la Cuidad de Buenos Aires.

The last sentence of Section 14 is deleted and replaced with:
Any questions concerning this Agreement should be referred to CA de Argentina S.A, Avenida Alicia Moreau de Justo, 400, 2 piso, 1107, Buenos Aires – At.: Finance Department.

Australia
The CA subsidiary that is the licensor is CA (Pacific) Pty. Ltd (ABN 20 001 146 345).

The following is added to each of the end of Sections 2, 3 and 10:
Although CA specifies that there are no warranties, Licensee may have certain rights under the Competition and Consumer Act 2010 and other state and territory legislation which may not be excluded but may be limited. To the full extent permitted by law CA excludes all terms not expressly set out in the express terms of this Agreement, and limits any terms imposed by the Competition and Consumer Act 2010 and other state and territory legislation to the full extent permitted by the applicable legislation.

The last sentence of Section 8 is deleted and replaced with:
The laws of the State or Territory in which the transaction is performed govern this Agreement.

The following is added to Section 10:

Where CA is in breach of a condition or warranty implied by the Competition and Consumer Act 2010 or other state and territory legislation, CA's liability is limited, in the case of goods, to the repair or replacement of the goods, or payment for the repair or replacement of the goods, and in the case of services, the supplying of the services again or payment for the re-supply of the services, as CA may elect. Where that condition or warranty relates to a right to sell, quiet possession or clear title, in respect of goods or if the goods supplied by CA are of a kind ordinarily acquired for personal, domestic or household use or consumption, then none of the limitations in this Section apply.

Brazil
The CA subsidiary that is the licensor is CA Programas de Computador, Participaçðes e Serviços Ltda.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of Brazil. Any dispute hereunder shall be determined by a court of the São Paulo City Hall.

The last sentence of Section 14 is deleted and replaced with:
Any questions concerning this Agreement should be referred to CA Programas de Computador, Participaçðes e Serviços Ltda., Avenida das Nações Unidas, 12901 – 6 andar – Torre Norte – São Paulo – SP, 04578-000, At.: Worldwide Law Department.

Canada
The CA subsidiary that is the licensor is CA Canada Company.

The last sentence of Section 8 is deleted and replaced with:
The laws in the Province of Ontario shall govern this Agreement.

Chile
The CA subsidiary that is the licensor is CA de Chile, S.A.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of Chile. Any dispute hereunder shall be determined by the Tribunales Ordinarios de la Cuidad de Santiago.

The last sentence of Section 14 is deleted and replaced with:
Any questions concerning this Agreement should be referred to CA de Chile, S.A, Avenida Providencia 1760, piso 15 – Edificio Palladio, oficina 1501 - 6640709 Providencia - Santiago – At.: Finance Department.

China
The CA subsidiary that is the licensor is CA (China) Co., Ltd.

The second sentence of Section 6 is deleted and replaced with:
All fees are inclusive of VAT.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of the People’s Republic of China, without regard to its choice of law provisions. Any dispute hereunder shall be determined by a competent court located in Beijing.

Colombia
The CA subsidiary that is the licensor is CA Software de Colombia S.A.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of Colombia. Any dispute hereunder shall be determined by the Tribunales Ordinarios de la Cuidad de Bogotá.

The last sentence of Section 14 is deleted and replaced with:
Any questions concerning this Agreement should be referred to CA Software de Colombia S.A, Edificio Grupo Santander Central Hispano Torre 2 - Oficina 401 Carrera 7 - Nº 99-53 - Bogotá D.C. - Colombia – At.: Finance Department.

Hong Kong

The CA subsidiary that is the licensor is CA (Hong Kong) Limited which is also its principal place of business at Suites 2301 2306, 23rd Floor, Dah Sing Financial Centre, 108 Gloucester Road, Wanchai, Hong Kong.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of Hong Kong. The courts of Hong Kong will have sole and exclusive jurisdiction with respect to any disputes arising out of this Agreement.

The following is added at the end of Section 10: The aforementioned liability limitation and the aforementioned maximum liability amount will not affect or prejudice the statutory rights of the licensee under the sale of goods ordinance, the supply of services (implied terms) ordinance or the control of exemption sections ordinance, nor will they limit or exclude any liability for death or personal injury solely caused by CA's negligence.

India
The CA subsidiary that is the licensor is CA (India) Technologies Private Limited.

The last sentence of Section 8 is deleted and replaced with:
This Agreement and the terms hereof shall be governed and construed in accordance with the laws of India and the courts of Mumbai shall have sole and exclusive jurisdiction with respect to any disputes arising out of this Agreement. In the event the Product is delivered electronically, the said Product shall be made available by CA for downloading from a server situated in a country other than India.

Indonesia (refer to Singapore)

Japan
The CA subsidiary that is the licensor is CA Japan, Ltd.

The third sentence of Section 6 is deleted and replaced with:
Licensee agrees to pay any tariffs, duties or taxes imposed or levied by any government or governmental agency other than the taxes for which CA is responsible upon a presentation of invoices by CA.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of the country of Japan, without regard to its choice of law provisions. Any dispute hereunder shall finally be determined by Tokyo District Court located in Tokyo Japan.

The last sentence of Section 14 is deleted and replaced with:
Any questions concerning this Agreement should be referred to CA Japan, Ltd., 2-7-9, Hirakawa-cho, Chiyoda-ku, Tokyo, 102-0093, Japan, Attention: Worldwide Law Department.

Korea
The CA subsidiary that is the licensor is CA Korea Inc., Ltd.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of Republic of Korea, without regard to its choice of law provisions.

The last sentence of Section 14 is deleted and replaced with:
Any questions concerning this Agreement should be referred to CA Korea Inc., Ltd, City Air Tower (18th Fl.), 159-9, Samsung-Dong, Kangnam-Ku, Seoul 135-973 Korea, Attention: Worldwide Law Department.

Malaysia
The CA subsidiary that is the licensor is CA (Malaysia) Sdn. Bhd.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of Malaysia. The courts of Malaysia will have sole and exclusive jurisdiction with respect to any disputes arising out of this Agreement.

The following is added to Section 10:
Although CA specifies that there are no other warranties, Licensee may have certain rights under the Consumer Protection Act 1999 and the warranties are only limited to the extent permitted by the applicable legislation.

Mexico
The CA subsidiary that is the licensor is CA Software de MÉxico S.A. de C.V.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of MÉxico. Any dispute hereunder shall be determined by the Tribunales de la Cuidad de MÉxico.

The last sentence of Section 14 is deleted and replaced with:
Any questions concerning this Agreement should be referred to CA Software de MÉxico S.A. de C.V, Av. Miguel de Cervantes Saavedra 193, Suite 502, Col. Granada, Mexico City, MX 11500 – At.: Finance Department.

New Zealand
The CA subsidiary that is the licensor is CA Pacific (NZ) Ltd.

Notwithstanding the final sentence of Section 6, the applicable interest charge on invoices unpaid by Licensee is 1.5% per month.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of New Zealand. The courts of New Zealand will have sole and exclusive jurisdiction with respect to any disputes arising out of this Agreement.

The following is added to Section 10:
Although CA specifies that there are no warranties, Licensee may have certain rights under the Consumer Guarantees Act 1993 or other legislation which cannot be excluded or limited. The Consumer Guarantees Act 1993 will not apply in respect of any goods or services which CA supplies, if Licensee acquires the goods and services for the purposes of a business as defined in that Act. Where the Product is not acquired for the purposes of a business as defined in the Consumer Guarantees Act 1993, the limitations in this Section are subject to the limitations in that Act.

The following is added to Section 12:
CA’s rights under this Section shall also apply if any resolution is passed or proceedings are commenced for the liquidation or winding up of Licensee.

Peru
The CA subsidiary that is the licensor is CA de Peru S.A.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of Peru. Any dispute hereunder shall be determined by the Tribunales Ordinarios de La Cuidad de Lima.

The last sentence of Section 14 is deleted and replaced with:
Any questions concerning this Agreement should be referred to CA de Peru S.A, Avenida Paseo de La Republica, 3211, Piso 11, San Isidro, Lima 27, Peru – At.: Finance Department.

Philippines
The CA subsidiary that is the licensor is Philippine Computer Associates International, Inc.

The first eight sentences of Section 8 are deleted and replaced with:
Title to the Product and all modifications thereto shall remain with CA. The Product is a trade secret and the proprietary property of CA or its licensors. Licensee agrees that CA may use any feedback provided by Licensee related to the Product for any CA business purpose, without requiring consent including reproduction and preparation of derivative works based upon such feedback, as well as distribution of such derivative works. Usage rights respecting the Product may not be exchanged for any other CA product. Licensee and its employees will keep the Product and the terms of this Agreement strictly confidential. To the maximum extent permitted by applicable law, Licensee will not disclose, de-compile, disassemble nor otherwise reverse engineer the Product.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of the Philippines. The courts of Makati City will have sole and exclusive jurisdiction with respect to any disputes arising out of this Agreement.

Section 12 is deleted and replaced with:

If Licensee breaches any term of this Agreement or if Licensee becomes insolvent or if bankruptcy or receivership proceedings are initiated by or against Licensee, CA shall have the right to withhold its own performance hereunder and/or to terminate this Agreement immediately upon notice and, in addition to all other rights of CA, all amounts due or to become due hereunder will immediately be due and payable to CA.

Singapore
The CA subsidiary that is the licensor is CA (Singapore) Pte. Ltd.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of Singapore. The courts of Singapore will have sole and exclusive jurisdiction with respect to any disputes arising out of this Agreement.

The following is added to the end if Section 9:
To the full extent permitted by applicable law, CA disclaims all implied conditions or warranties of satisfactory quality or fitness for purpose.

Taiwan
The CA subsidiary that is the licensor is CA (Taiwan) Limited whose registered office is situated at 17F/B, No. 167, Tun Hwa North Road, Taipei City 105, Taiwan.

The second sentence of Section 6 is deleted and replaced with:
All fees are exclusive of VAT.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of Taiwan, without regard to its choice of law provisions. Any dispute hereunder shall be determined by Taipei District Court.

Thailand
The CA subsidiary that is the licensor is CA Sales (Thailand) Co., Ltd.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of Thailand. The courts of Thailand will have sole and exclusive jurisdiction with respect to any disputes arising out of this Agreement.

Venezuela
The CA subsidiary that is the licensor is Computer Associates (CAI) de Venezuela, CA.

The last sentence of Section 8 is deleted and replaced with:
This Agreement shall be governed by and interpreted in accordance with the laws of Venezuela. Any dispute hereunder shall be determined by the Tribunales Ordinarios de la Cuidad de Caracas.

The last sentence of Section 14 is deleted and replaced with:
Any questions concerning this Agreement should be referred to Computer Associates (CAI) de Venezuela, CA, Avenue Francisco de Miranda, Centro Lido, Torre B, pisso 5, officina B-51, El Rosal, Caracas 1060, Venezuela – At.: Finance Department.

16. If the Product contains third party software, and the licensor requires the incorporation of specific license terms and conditions for such software into this Agreement, those specific terms and conditions, which are hereby incorporated by this reference, are located below this Agreement.

Licensee acknowledges that this license has been read and understood and by selecting the "I accept the terms of the License Agreement " option button, licensee accepts its terms and conditions. Licensee also agrees that this license (including any order form referencing this Agreement and any terms relating to third party software which are set forth below this Agreement) constitutes the complete Agreement between the parties regarding this subject matter and that it supersedes any information licensee has received relating to the subject matter of this Agreement, except that this Agreement (excluding the third party terms below) will be superseded by any written Agreement, executed by both licensee and CA, granting licensee a license to use the product. This Agreement may only be amended by a written Agreement signed by authorized representatives of both parties.

Select the "I accept the terms of the License Agreement" option button, and then click on the "Next" button to accept the terms and conditions of this Agreement as set forth above and proceed with the installation process.

Select the "I do NOT accept the terms of the License Agreement" option button and then click on the "Cancel" button to halt the installation process.

Documentation Legal Notice

This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by Broadcom at any time. This Documentation is proprietary information of Broadcom and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of Broadcom.

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all Broadcom copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to Broadcom that all copies and partial copies of the Documentation have been returned to Broadcom or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, BROADCOM PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL BROADCOM BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF BROADCOM IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice

The manufacturer of this Documentation is Broadcom Inc.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2005–2026 Broadcom. All Rights Reserved. The term “Broadcom” refers to Broadcom Inc. and/or its subsidiaries. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Symantec
by Broadcom Software