

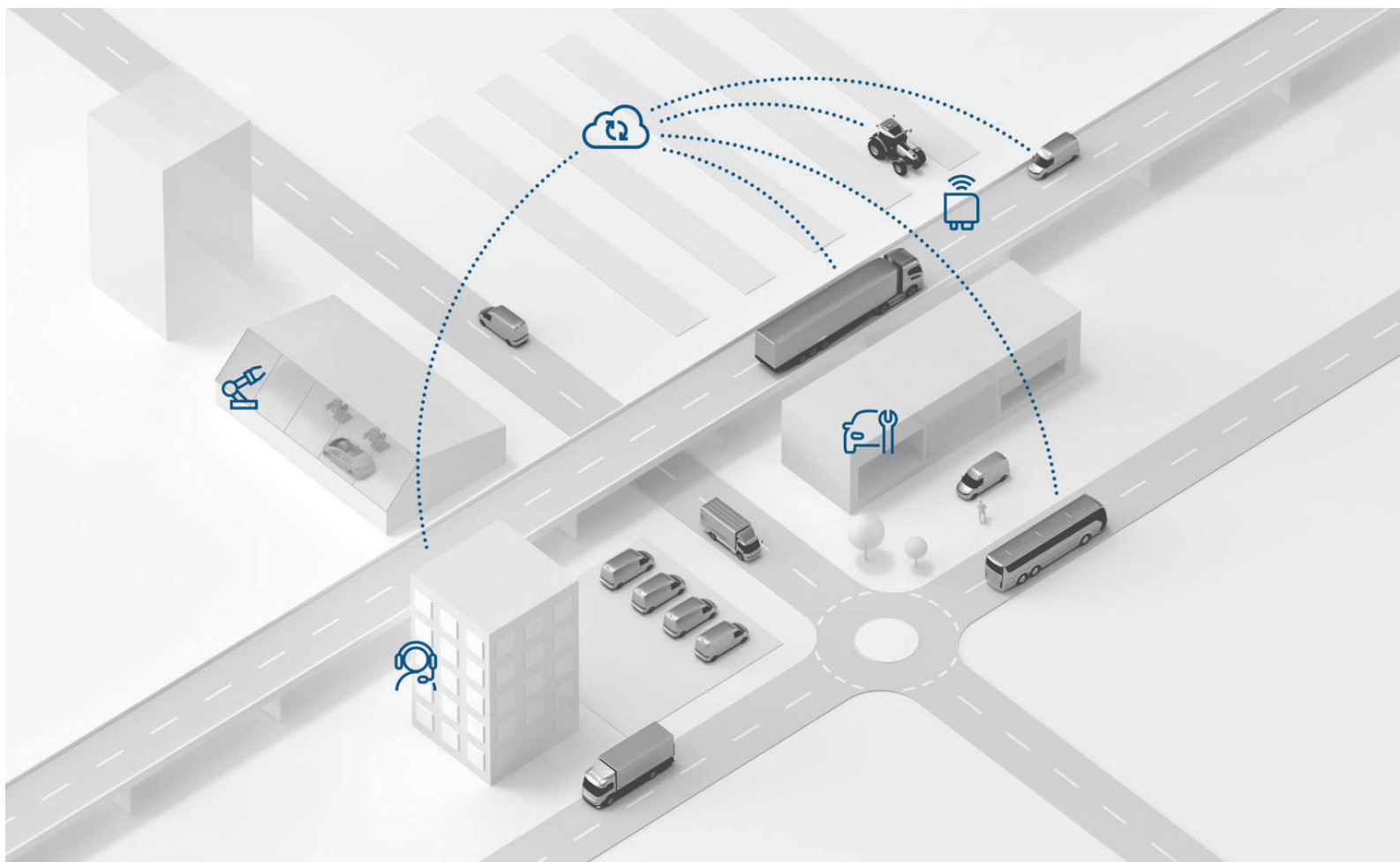
Invented for life



BOSCH

Automotive Connectivity Hub

User guide



© 2023 - 2026 Robert Bosch GmbH

All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

Table of contents

1	Introduction	8
1.1	Key benefits	9
1.1.1	Benefits for telematics service providers	9
1.1.2	Benefits for vehicle manufacturers	9
1.2	Features	10
1.2.1	C-Hub TCU2 device and accessories	10
1.2.2	C-Hub backend and cloud infrastructure	11
1.2.3	Operations and support	11
1.3	Data as a Service - Base and Premium Service Plans	12
1.3.1	Core service components	12
1.3.2	Operations and configuration	13
	Remote measurement configuration	13
	Cloud data transfer options	13
1.3.3	Support	14
1.4	Connectivity as a Service	14
1.4.1	Hardware and accessories	14
	Telematic control unit - C-Hub TCU2 device	14
	Main connector	14
	Antenna	15
1.4.2	Cloud infrastructure - C-Hub backend	15
	Updates over the air - optional for other OEM ECUs	15
1.4.3	Support	15
1.5	Intended use	15
2	Basic concepts	16
2.1	C-Hub TCU2 device and on-board components	18
2.1.1	C-Hub TCU2 device	18
2.1.2	Accessories applicable to vehicles in the field - Data as a Service customers	19
	DIN Adapter EU	19
	Wiring harness EU	20
	Antenna	20
2.1.3	Accessories applicable to vehicles in production - Connectivity as a Service customers	20
	Main connector	20
	Antenna	21

2.1.4	Software installed on the C-Hub TCU2 device	21
2.1.5	Custom code on the C-Hub TCU2 device	21
2.1.6	Antennas for EU and NA	22
	Installation notes	22
	Approved antennas and cables	22
	For usage in Europe	23
	For usage in North America	23
2.2	C-Hub backend overview for vehicle manufacturers	24
2.2.1	Vehicle Provisioning - Device Commissioning	24
	How it works	24
	Steps applicable at C-Hub	25
	Connectivity as a Service customers	25
2.2.2	Device Connectivity and Device Management	26
	How it works	26
	Steps applicable at C-Hub	27
	Connectivity as a Service customers	27
	Device Connectivity and Device Management details	27
	How it works on the backend	27
	How it works on the device	29
2.2.3	Device Maintenance	30
	How it works	30
	Steps applicable at C-Hub	31
	C-Hub Connectivity as a Service customers	31
2.2.4	Remote Measurement	32
	How it works	32
	Steps applicable at C-Hub	33
	C-Hub Connectivity as a Service customers	33
	Remote measurement configuration	33
2.2.5	Remote Diagnostics	34
	How it works	34
	Steps applicable at C-Hub	35
	Connectivity as a Service customers	35
2.2.6	Data Integration	36
	How it works	36
	Steps applicable at C-Hub	36
	C-Hub Connectivity as a Service customers	37
2.3	Customer applications communicating to the C-Hub backend	38
2.4	Authentication and authorization	39
2.4.1	C-Hub TCU2 device	39

2.4.2	C-Hub backend	39
2.4.3	User	40
2.4.4	Customer solution or another third-party application	40
3	C-Hub TCU2 device installation	41
3.1	Installation concept	41
3.2	Installation concept for telematics service providers (TSP)	41
3.3	Installation concept for vehicle manufacturers (OEM)	42
3.4	Further hints on the installation concept for vehicle manufacturers (OEM)	42
3.4.1	C-Hub TCU2 device - all connectors	42
3.4.2	C-Hub TCU2 device - main connector	43
3.4.3	Antennas	43
4	Data configuration concepts	44
4.1	FMS standard parameters	45
4.2	OBD-II diagnostic data	45
5	Data consumption options	46
5.1	Data consumption for fleet operators and mobility providers	46
5.1.1	AMQP	47
5.1.2	AWS S3	48
5.1.3	Azure Blob Store	49
5.1.4	HTTP based API or backend	50
5.1.5	Bosch IoT Insights data recorder API	50
5.1.6	Apache Kafka	51
5.2	Data consumption for OEMs	51
5.2.1	Data consumption by customer application via Apache Kafka PubSub	52
	Environments	52
	Provisioning of PubSub	53
	Onboarding	53
	Access to topic	53
	Topics for async API	54
	Example - getting started with Kafka	55
	Prerequisites	55
	Connecting a Kafka client to PubSub	55
5.2.2	Data consumption by customer application via REST API	56
	Authentication	56
	Scopes	56
	C-Hub defined scopes	57
	Service endpoints	57
	Examples	57

OAuth2.0 Based Authorization	58
Authorize service account via confidential client	58
Use Access Token	58
RESTful API docs	59
Device Management API (V2)	59
VMS Inventory Service (1.0.0)	59
FOTA Distribution (1.0.0)	59
VMS Campaign Service (1.0.0)	59
Context Pipeline (1.0.0)	59
Example - data consumption by customer application via REST API	61
Request a Bearer Token	61
GET a list of devices	61
5.2.3 Data consumption by customer application via Data Integration service	63
AMQP	63
AWS S3	64
Azure Blob Store	65
HTTP based API or backend	66
Bosch IoT Insights data recorder API	66
Apache Kafka	67
6 Data protection and cybersecurity aspects	67
6.1 General information about cybersecurity	67
6.2 Cybersecurity claims	68
7 Safety and security notes	68
7.1 Safety and security notes - drivers	70
7.1.1 Notice	70
7.2 Safety and security notes - transport and assembly partners	71
7.2.1 Warning	71
7.2.2 Caution	71
7.2.3 Notice	72
Device cleaning related	73
Antenna-related	73
Cable-related	74
DIN-Adapter related	75
7.3 Safety and security notes - developers and operators of the cloud services	75
7.3.1 Danger	75
7.3.2 Warning	77
7.3.3 Notice	77
7.4 Additional notes for developers	77

7.4.1	Responsibilities and obligations	77
	Responsibilities and obligations of the customer	78
	Joint responsibilities and obligations between Bosch and the customer	78
	Responsibilities and obligations of Bosch	78
7.4.2	Recommendations for developers	79
7.4.3	Notes about containerized vehicle apps	80
8	Lifetime of the hardware	81
9	Recycling, repair, and disposal of the hardware	81
9.1	Recycling and reusing the device	81
9.2	Repair	81
9.3	Disposal	82
10	Support - Helpdesk	82
10.1	Ask a question	82
10.2	Report an issue	82
10.3	Manage fleets	83
10.3.1	User Management	83
10.3.2	Fleets & Vehicles	83
10.3.3	Use the API	84
11	Annex	84
11.1	FMS standard excerpt	84
11.1.1	Parameters for FMS Gateway - according to SAE J1939	84
	Parameters for Bus and Truck FMS-Standard	85
	FMS Tell Tale Status details	86
	Parameters for Truck FMS-Standard	87
	Parameters for Bus FMS-Standard	88
11.2	Kafka references	90
11.2.1	Customer load test	90
	Operations	90
	SUB customer-loadtest	90
11.2.2	Device-Vehicle mapping	90
	Operations	90
	SUB ps.\$tenantId.device-vehicle-mapping.events.vin-mismatch	90
11.2.3	OTA Data Exporter	91
	Operations	92
	SUB ps.\$tenantId.measurements.results	92
	SUB ps.\$tenantId.diagnostics.results	102
	SUB ps.\$tenantId.vehicles.management.events	102
11.2.4	OTA Function Calls	106

Operations	106
SUB ps.\$tenantId.ota-function-calls.events.function-call-status	106
11.2.5 OTA Updates Campaign Exporter	106
Operations	107
SUB ps.\$tenantId.campaigns.events	107
11.2.6 OTA Updates Distribution Exporter	125
Operations	125
SUB ps.\$tenantId.ota-updates.events	125
SUB ps.\$tenantId.desired-states.events	136
11.2.7 OTA Updates Inventory Exporter	137
Operations	138
SUB ps.\$tenantId.inventories.events	138
11.3 C-Hub glossary	140
11.4 Declaration of conformity - C-Hub TCU2 device	141
11.4.1 Regions subject to CE marking	142
Countries in the EU	142
Other CE Countries	144
11.4.2 Other region-specific statements	145
European Economic Area (EEA) countries	145
Single markets and others	145
11.4.3 Regions subject to UKCA marking	145
UNITED KINGDOM (GB)	145
British Overseas Territories (BOTs)	145
11.4.4 USA	146
11.4.5 Canada	146
11.5 Additional accessories for TSP customers - not part of the standard offering for OEM	147
11.5.1 DIN Adapter - for TSP	147
Description	147
Technical data C-Hub DIN adapter	148
Device status indicated by 2 LEDs	148
Installation concept for telematics service providers (TSP)	149
11.5.2 Declaration of conformity C-Hub DIN adapter	149
11.6 Additional information for OEM customers	150
11.6.1 Main connector - pinning info	151
Main connector pinning	151
11.6.2 Diagnostic Trouble Codes - C-Hub TCU2 device	154

1 Introduction

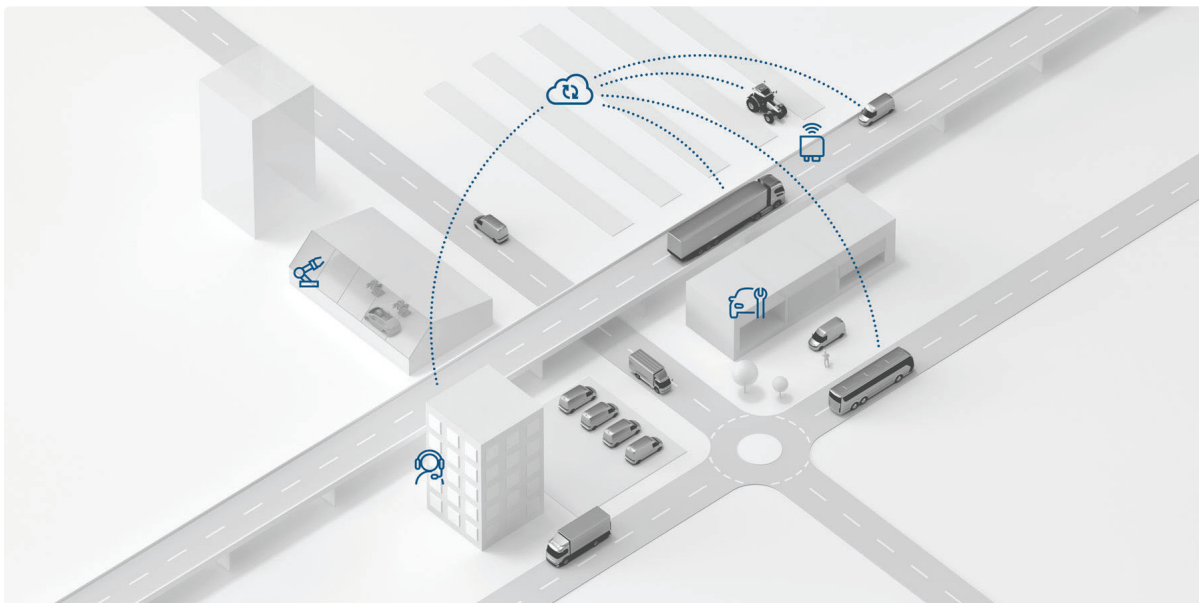
Welcome to C-Hub: Unlock your fleet's full potential

Managing a modern fleet is complex, especially with vehicles from different manufacturers. To truly optimize operations and make informed decisions, you need a reliable source of high-quality vehicle data.

The Automotive Connectivity Hub (C-Hub) is a product designed for B2B customers—including commercial fleets, telematic service providers (TSPs), and vehicle manufacturers (OEMs). It provides the foundational data needed to power advanced, data-driven applications.

C-Hub serves as a holistic enabler, combining a high-performance in-vehicle device (the C-Hub TCU2) with a powerful cloud backend to ensure the smooth, continuous collection of live vehicle data across your entire fleet.

Quick overview



- **Simple installation:**
A certified professional mounts the C-Hub TCU2 device in the vehicle (e.g., in an empty DIN slot).
- **Automatic operation:**
The device activates when the ignition is turned on. It runs silently in the background, without affecting vehicle performance or requiring any driver interaction.
- **Seamless data transfer:**
The TCU2 continuously records vehicle data and transmits it securely to the C-Hub cloud backend using a mobile network.
- **Easy data access:**
Data-consuming applications can access the collected vehicle information via a simple API, enabling a new level of efficiency and digital service integration.

Preliminary

1.1 Key benefits

With C-Hub, Bosch offers a holistic product which allows high-quality vehicle data to be collected during the vehicle operation.

The C-Hub TCU2 device is a future-proof connectivity unit fulfilling automotive standards. The end-to-end communication from the device to the backend and back to the device also fulfills the automotive standards regarding security.

- C-Hub acts as an enabler for further vehicle applications from Bosch and third-party providers
- A selection of standardized, high-quality vehicle data is configurable via the cloud
- The managed C-Hub TCU2 devices in the vehicles receive regular feature and security updates over the air
- The C-Hub backend offers comprehensive device management



Additionally, the C-Hub TCU2 devices have a vehicle app capability, which allows customers to develop and deploy their own vehicle apps based on container technology and independently from Bosch.

A detailed guideline for developers can be provided to customers upon request.

1.1.1 Benefits for telematics service providers



- Easy to use, standardized connectivity solution for vehicles of different manufactures
- Easy to install into vehicles in the field, without the need for vehicle modifications
- Cooperation with installation partners for in-vehicle device installation & activation
- Bosch service for device configuration and Bosch-operated cloud backend services

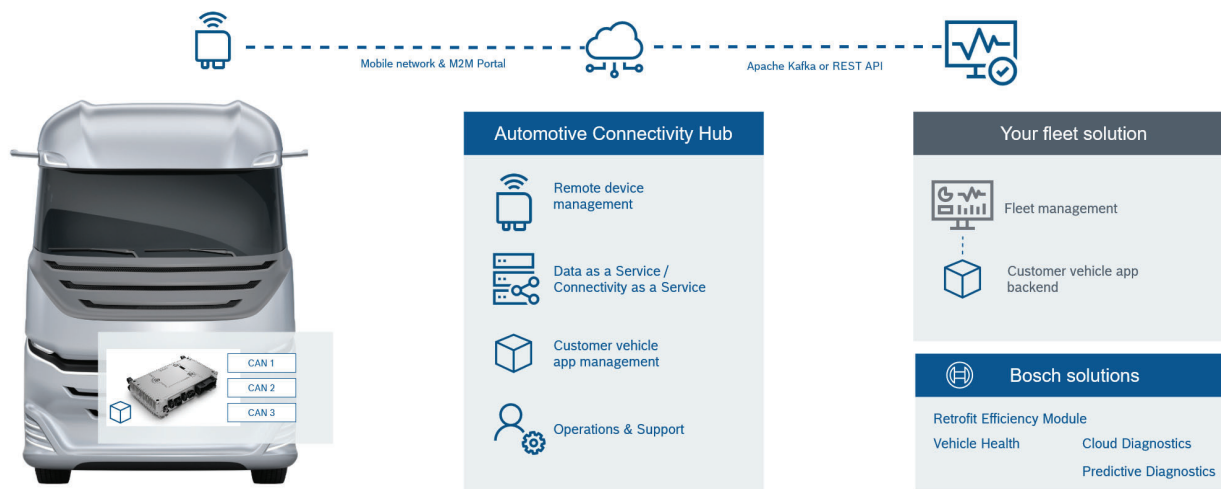
1.1.2 Benefits for vehicle manufacturers



- Independent plug-and-play connectivity solution
- Straightforward integration into development and production
- Full access to the cloud backend service

1.2 Features

The following picture shows the main building blocks of the C-Hub offering.



- The devices are configured in a way that they can automatically set up a connection to the C-Hub backend and cloud infrastructure.
- The C-Hub backend provides services for remotely configuring the devices and sending software updates over the air
- The solution which benefits from the collection of vehicle data can be either a third-party solution of yours or another vendor, or one of the Bosch services which can additionally be booked as an add-on.

Here is a zoom-in on the features of the various aspects which make C-Hub an end-to-end vehicle connectivity product: hardware, software services, operations, and support.

1.2.1 C-Hub TCU2 device and accessories



Hardware for customers in Europe

- TCU2 EU IP30 C-Hub - for short **C-Hub TCU2**
- DIN Adapter
- Main wiring harness
- OBD-II cable (upon request)
- Accessories for OEM customers: Main connector
- Antenna is required but not part of the C-Hub offering

Hardware for customers in North America

- TCU2 NA IP30 C-Hub for short **C-Hub TCU2**
- Main wiring harness
- OBD-II Vehicle Harness and T harness
- J1939 Vehicle Harness and T harness
- Accessories for OEM customers: Main connector
- Antenna is required but not part of the C-Hub offering

Software on the **C-Hub TCU2**

- OS - Yocto-based Linux
- C-Hub functionality

Preliminary

- **Runtime for Containerized Vehicle Apps**
which are the customer's own applications

1.2.2 C-Hub backend and cloud infrastructure



C-Hub comprises the following online services:

- Vehicle Provisioning
- Device Connectivity & Management
- Device Maintenance
- Remote Measurement
- Remote Diagnostics
- Data Integration service for applications consuming data from the backend
- Containerized Vehicle Apps service for deployment of customer's own applications

Authentication and authorization

- Data separation by tenant
- Auth mechanisms for devices, users, and product internal cloud services
- Auth mechanisms for technical users of customer applications

1.2.3 Operations and support



- Fully managed services
- Cloud infrastructure operated online by Bosch
- Over-the-air updates (OTA)
 - Security updates of the C-Hub TCU2 devices
 - Functional updates
- Change management in case of changes in the requirements towards the data to be collected
- API credentials and their renewal upon request
- Support
 - [Support service desk](#)
 - Support availability:
General office hours on working days in Germany: 9 am to 5 pm



Containerized Vehicle Apps

For customers interested in building their own **Containerized Vehicle Apps**: ask at an early stage for the dedicated onboarding of your developers, and for the *C-Hub developer guide*.

1.3 Data as a Service - Base and Premium Service Plans

This section provides an overview of the C-Hub service offering, its core components, data package details, and available add-on services.

Overview

Data as a Service is a foundational offering that provides essential hardware management, application hosting, and data services.

It is designed as an enabling service that can be extended with optional add-on services e.g. other custom or Bosch solutions to meet specific customer needs.

Service plans:

- **C-Hub Base Service**
Base Service includes standardized vehicle data and standardized diagnostics data.
- **C-Hub Premium Service**
Premium Service includes all Base capabilities and additionally provides access to non-standardized, OEM-specific data.

1.3.1 Core service components

The **Automotive Connectivity Hub** includes the following main components:

	Base	Premium
Managed connected hardware		
Ensures the smooth connectivity, operation, and security of your connected hardware. ▪ Bosch delivers hardware and accessories for your own installation in commercial vehicles ▪ Device connectivity and device management ▪ Maintenance and security updates over the air ▪ Data packages - Mobile data required for the functionality of the C-Hub in general are included.		
Data as a Service		
Data as a Service Base Enables the standardized delivery and processing of data. ▪ Remote Measurement and Remote Diagnostics Custom configuration to read standardized vehicle data (e.g., J1939, FMS) and standardized diagnostics data (e.g. any OBD-II standard). ▪ Delivery of standardized data to the customer's backend application or system		
Data as a Service Premium Enables the delivery and processing of OEM-specific premium data. Custom configuration to read OEM specific vehicle data, and OEM specific diagnostics data	not included	

Preliminary

Enables the remote execution of function calls on the vehicles in the field	not included	
Active function capability, e.g. clear all DTCs on the vehicle		
Optional additional custom containerized apps		
Secure and reliable environment for running customer-specific applications on the hardware installed in the commercial vehicles.		
You can create containerized applications to run on the Bosch hardware		
Get access to the tool-chain to securely deploy your app and update the C-Hub TCU2 devices in the field over the air		
Data package (mobile data)		
- Data-as-a-Service: Includes max. 100 MB/month shared among diverse services, e.g.: - Ingestion of standardized data based on customer request (e.g. data transfer from the C-Hub TCU2 to the C-Hub Backend) - Custom containerized custom apps - Rollout of software updates of containerized custom apps (e.g. over the air campaigns to the C-Hub TCU2) - Data transfer from and towards containerized custom apps (e.g. data transfer from and to the C-Hub TCU2)		
	Included: 100 MB/month	Included: 100 MB/month
Optional data packages: Additional data packages can be purchased to supplement included data or to support add-on services.	available	available

Preliminary

1.3.2 Operations and configuration

All services are managed by Bosch and provided in a cloud infrastructure.

Remote measurement configuration

The set of required vehicle signals, such as J1939, CAN, GNSS and available device signals can be easily configured. Configuration updates over the air to the vehicles in the field to change the desired signal measurement can be performed separately from the software updates, including campaign management.

We expect that you as a customer will send Bosch the requirements, e.g. which data set to collect and how often to retrieve each data point.

Bosch will execute these jobs and make the results accessible for authorized customer applications via API.

Cloud data transfer options

- API access of custom application to Bosch's Apache Kafka infrastructure
See user guide: [Data consumption by customer application via Apache Kafka PubSub](#)

- Routing the data to your application (e.g. to Bosch solutions listed as optional add-on services)
See user guide: [Data consumption by customer application via Data Integration service](#)
- Future: Vertical Integration

1.3.3 Support

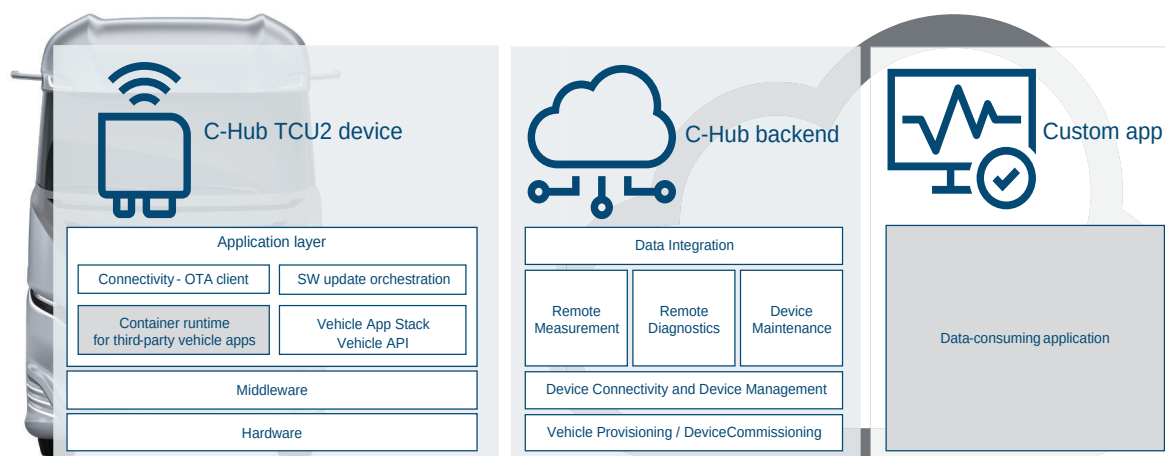
In the onboarding phase a Bosch customer success manager will be your guide.

The C-Hub support team can be contacted via the helpdesk: <https://bos.ch/helpdesk>.

1.4 Connectivity as a Service

The Connectivity as a Service offering is designed for manufacturers of commercial vehicles, aka OEM.

It includes the necessary hardware, software, as well as operations and support.



Preliminary

1.4.1 Hardware and accessories

Telematic control unit - C-Hub TCU2 device

- C-Hub uses the latest generation of the Telematic Control Unit manufactured by Bosch. Technical details are described in a data sheet which accompanies the hardware.
- Using over-the-air updates for the device, Bosch continuously maintains the product with the newest software to secure the operation for OEMs and their customers.
- A C-Hub TCU2 device needs to be installed in each commercial vehicle from which data is going to be collected and sent to the C-Hub backend.
The collected vehicle data will be processed on-board by the C-Hub TCU2 device and then sent to the cloud infrastructure via a mobile data network.
- The OEM have the option to configure the data in which they are interested in self-service mode.
- Additionally, the C-Hub TCU2 device provides the possibility to host customer's code as containerized Vehicle Apps.

Note: Licenses for the use of LTE in vehicles are not part of this product.

Main connector

C-Hub provides a main connector for connecting the C-Hub TCU2 device to the OEM's wire harness.

The main connector is part of the delivery.

Antenna

Bosch is **not** the device manufacturer but will describe the specification and can (upon customer request) conduct compatibility testing.

However, an antenna needs to be **provided by the customer** when the C-Hub TCU2 device is mounted into the truck as part of the manufacturing process. See Auxiliary equipment - antenna and cabling EU and NA.

1.4.2 Cloud infrastructure - C-Hub backend

C-Hub uses a cloud infrastructure with a fully managed, secure, cloud-based integration platform for over-the-air services.

The C-Hub backend is provided and operated by Bosch.

However, the customer will have full ownership on all data and configuration on his "tenant".

Connectivity as a Service model

- The customer can use the C-Hub backend in self-service mode (cloud-based UIs and user guides are accessible for OEM)
- The customer's applications or third-party services can access the data via API
- The customer has full ownership of any additional applications which he chooses to run on the C-Hub TCU2 hardware
- Bosch manages only the cloud services and device software

By agreeing with this model, we expect that you as a customer prefer to use the cloud services without Bosch as an intermediary.

For any help needed, you can approach additional professional services.

Updates over the air - optional for other OEM ECUs

- This optional service can be booked additionally for OEM customers of C-Hub.
- The service allows OEM customers to use the same approach as used for the C-Hub TCU2 to distribute updates over the air (OTA) to other custom specific devices in the vehicle.

1.4.3 Support

Whether you are exploring our products or running a commercial application which processes vehicle data based on the C-Hub – we are there for you whenever you need support!

The C-Hub support team can be contacted via the helpdesk: <https://bos.ch/helpdesk>.

1.5 Intended use

The product **Automotive Connectivity Hub**, C-Hub for short, is released on the basis of the regulatory requirements directly applicable to the product at the time of writing this document in the target markets listed below.

The product is designed and released for civil applications only.

Deliveries and services (fulfillment of contract) shall be subject to the provision that there are no obstacles to performance due to national or international (re-)export control regulation, in particular embargoes or other sanctions.

Target market for the C-Hub device and services are most European countries. Additionally, the product can be offered in North America.

List of countries in alphabetical order: Andorra (AD), Austria (AT), Belgium (BE), Bosnia and Herzegovina (BA), Bulgaria (BG), Canada (CA), Croatia (HR), Cyprus (CY), Czechia (CZ), Denmark (DK), Estonia (EE), Finland (FI), France (FR), Germany (DE), Gibraltar (GI), Greece (GR), Hungary (HU), Iceland (IS), Ireland (IE), Isle of Man (IM), Italy (IT), Jersey (JE), Latvia (LV), Liechtenstein (LI), Lithuania (LT), Luxembourg (LU), Malta (MT), Monaco (MC), Netherlands (NL), North

Macedonia (MK), Norway (NO), Poland (PL), Portugal (PT), Romania (RO), San Marino (SM), Slovakia (SK), Slovenia (SI), Spain (ES), Sweden (SE), Switzerland (CH), United Kingdom of Great Britain and Northern Ireland (GB), United States of America (US).

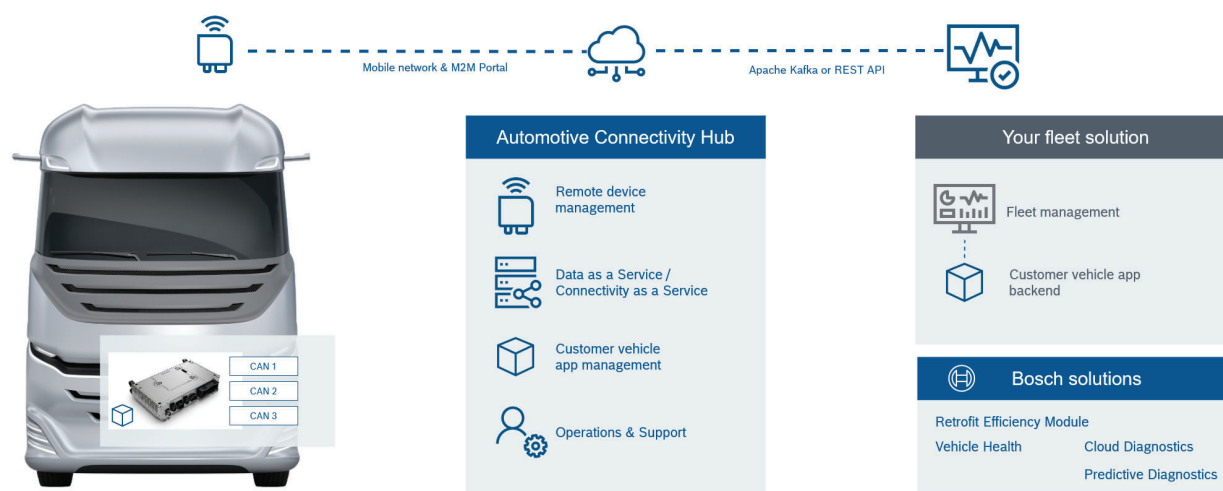
In the case that the customer wants to use the product outside the *Intended Use*:

- the customer shall, on its own responsibility, evaluate and comply with any regulatory requirements resulting therefrom for the product, evaluate and ensure the usability of the product for the area of application intended, or
- the customer shall obtain a new, extending Bosch release, which shall be ordered separately from Bosch (e.g., by means of a change request).

2 Basic concepts

C-Hub is an end-to-end product which can be integrated into a vehicle in order to connect it to the cloud and collect business-relevant or technically relevant data. The product can be applied both to vehicles in the field, and to vehicles in the factory.

The C-Hub product encompasses the hardware, software, operations, and support which are necessary for a connected vehicle.



Preliminary

C-Hub TCU2 device and on-board components

The cloud connectivity of a vehicle is enabled by the C-Hub telematic control unit, aka C-Hub TCU2 device, which is mounted in the participating vehicles.

Bidirectional communication between the C-Hub TCU2 device and the C-Hub backend is performed via a mobile data network.

Find more details in the [C-Hub TCU2 device and on-board components](#) section.

C-Hub backend

The cloud services part of the C-Hub backend are described in dedicated sections.

See [C-Hub backend overview for vehicle manufacturers](#).

Customer applications communicating to the C-Hub backend

The data collected from the connected vehicles can be used as a basis for data-driven decisions in various applications.

To make the vehicle data accessible to your applications, C-Hub offers API access to the data.

See [Customer applications communicating to the C-Hub backend](#).

Authentication and authorization

The devices, as well as every human or technical user of the cloud services have to authenticate and to be authorized for the relevant scope of interaction.

Find details on the authentication and authorization concept in section [Authentication and authorization](#).

2.1 C-Hub TCU2 device and on-board components

The hardware of the C-Hub product includes:

- a second generation high-performance telematic control unit, in particular the TCU2 EU IP30 C-Hub, or C-Hub TCU2 device for short, manufactured by Robert Bosch GmbH, as well as
- a set of accessories, specific to the customer type.

This hardware needs to be installed in each vehicle from which the data is going to be collected and sent to the cloud.

Proper installation of the C-Hub TCU2 into vehicles is performed either by a trained installation partner of Robert Bosch GmbH (for Data as a Service customers) or by the vehicle manufacturer (for Connectivity as a Service customers).

The secure operation of the C-Hub TCU2 device is assured by Robert Bosch GmbH. System security is maintained through ongoing over-the-air (OTA) software updates (rolled out by the C-Hub team), which ensures that the C-Hub TCU2 device Operating System and Application software is kept up to date.

Find details in following sections:

- [2.1.1 C-Hub TCU2 device](#)
- [2.1.2 Accessories applicable to vehicles in the field - Data as a Service customers](#)
 - [DIN Adapter EU](#)
 - [Wiring harness EU](#)
 - [Antenna](#)
- [2.1.3 Accessories applicable to vehicles in production - Connectivity as a Service customers](#)
 - [Main connector](#)
 - [Antenna](#)
- [2.1.4 Software installed on the C-Hub TCU2 device](#)
- [2.1.5 Custom code on the C-Hub TCU2 device](#)

2.1.1 C-Hub TCU2 device

The C-Hub TCU2 device is a telematic control unit with the following features:



- Power and Processing
 - Nominal supply voltages are 12 V and 24 V (the full operational range is 9 V to 32 V)
 - Co-processor for vehicle communication
 - LTE cellular module with integrated application processor and memory
- Wireless Interfaces:
 - 4G LTE Cellular Interface
 - SIM Card support: Internal embedded SIM
 - Variant EU: GNSS: GPS, GLONASS, BeiDou, Galileo, QZSS
 - Variant NA: GNSS: GPS, Galileo
 - WLAN b/g/n/ac - HW ready, but deactivated
 - Bluetooth 5.0
 - Bluetooth BLE - HW ready, but not usable once the C-Hub TCU2 device is installed into the vehicle
- Wired interfaces:
 - 3x CAN-FD Bus (compatible to CAN 2.0)
 - 1x Automotive Ethernet 100Base-T1
 - 5x Digital Inputs
 - 2x Low-side switch Outputs

- 1x USB 2.0 - not usable once the C-Hub TCU2 device is installed into the vehicle

2.1.2 Accessories applicable to vehicles in the field - Data as a Service customers

DIN Adapter EU



For a commercial vehicle with DIN slots, the C-Hub TCU2 device will first be installed into the DIN Adapter. Afterwards, the combined parts (DIN Adapter, C-Hub TCU2, antenna) will be mounted into the vehicle's DIN slot.

The DIN adapter itself will use the same power source as the C-Hub TCU2 device, which is the vehicle battery.

Its main function is to allow the mounting of the C-Hub TCU2 device into a DIN slot and to display the functional status of the C-Hub TCU2 device.

The device status is indicated by 2 LEDs:

Power LED	Connection LED	Meaning
Off	Off	No power supply to the device Troubleshooting: Check the cable connections and whether there is power supply reaching the C-Hub TCU2 device
On	Off	C-Hub TCU2 starting/booting LTE not connected (<i>connecting</i>) GNSS not connected (<i>connecting</i>)
On	Slow blinking: once, every 3 seconds 1. On for 1 second, off for 2 seconds 2. Repeat	✓ C-Hub TCU2 is powered up LTE connecting GNSS connecting
On	Fast blinking: twice, every 3 seconds 1. On for 0.5 seconds, off for 1 second 2. Repeat	✓ C-Hub TCU2 is powered up ✓ LTE connected GNSS connecting Troubleshooting: Check the network coverage for example on your mobile phone
On	Faster blinking: 3 times every 3 seconds 1. On for 0.5 seconds, off for 0.5 seconds 2. Repeat	✓ C-Hub TCU2 is powered up LTE connecting ✓ GNSS connected Troubleshooting: Check the network coverage for example on your mobile phone

Preliminary

Power LED	Connection LED	Meaning
On	On	✓ C-Hub TCU2 is powered up ✓ LTE connected ✓ GNSS connected

Wiring harness EU

C-Hub uses a main **wiring harness** which connects the C-Hub TCU2 device, the DIN Adapter, and the different interfaces of the vehicle.



One side of the wiring harness ends in a **main connector** - the white plug on the example picture - which is to be plugged into the main port of the C-Hub TCU2 device.

The other side of the wiring harness has the three ends, as follows:

- **LED** - the small black plug on the example picture - used for connection of the C-Hub TCU2 device to the LED cable of the DIN adapter
- **FMS** - the green plug on the example picture - Fleet Management interface (from German: Flotten Management Schnittstelle), used for collecting data from a CAN bus of the vehicle.
Such data is then transferred to the [Remote Measurement](#) service on the C-Hub back-end.
- **System Connector** - the gray plug on the example picture - used for connection to the OBD II port in the vehicle (by means of an additional OBD II cable), as well as for access to an additional CAN interface and/or the I/O ports.
The collected data from the OBD II port is in the form of Diagnostic Trouble Codes (DTC) which vehicle technicians use to identify problems with the vehicle.
Such data is then transferred to the [Remote Diagnostics](#) service on the C-Hub back-end.

Antenna

The antenna needs to be **provided by the customer** at the time when the installation partner starts preparing for the installation activities.

See [Antennas for EU and NA](#).

2.1.3 Accessories applicable to vehicles in production - Connectivity as a Service customers

Main connector

For connecting the C-Hub TCU2 device to the vehicle, the OEM needs to create a wiring harness, to be plugged into the main connector of the C-Hub TCU2 device.



Antenna

The antenna needs to be **provided by the customer** at the time when the installation partner starts preparing for the installation activities.

See [Antennas for EU and NA](#)

2.1.4 Software installed on the C-Hub TCU2 device

At the time of delivery, all necessary software is pre-installed on the C-Hub TCU2 device so that it can connect to the C-Hub back-end.



To keep the device up to date, the device software is updated over-the-air (OTA). Such updates are triggered by Robert Bosch GmbH.

A list of the installed open-source software can be requested via our support channel.

The vehicle data of interest to the customer is first processed on board the vehicle by the C-Hub TCU2 device and is then sent to the C-Hub back-end through the mobile data network.

2.1.5 Custom code on the C-Hub TCU2 device

The C-Hub TCU2 device provides the possibility to host customer's code as containerized Vehicle Apps.

Developers who need to implement and test such Vehicle Apps will be provided with a developer guide upon request.

2.1.6 Antennas for EU and NA

The antenna system can be selected from the ones listed below.

The customer is responsible for only using cables and/or antennas that ensure that limits with regards to minimum health and safety requirements regarding the exposure of workers to the risks from electromagnetic fields are fulfilled. See directives 2004/40/EU and 2013/35/EU.

The requirements in this chapter are deemed to be fulfilled when the product successfully has passed the release tests.



Data as a Service - For retrofit scenarios

For retrofit scenarios, the currently recommended antenna is Taoglas MA240 (Genesis). This antenna complies with the C-Hub TCU2 device certification parameters, supporting both GNSS and LTE functionalities.

You can order it at Taoglas by requesting a quote: <https://www.taoglas.com/contact/request-a-quote/>. In the form, include the Bosch-specific part number

MA240R.LBI.056.rb and the required quantity.

If you need further support with ordering or have questions, contact sales@taoglas.com via email, mentioning the Bosch C-Hub project and antenna part number.

Table of contents:

- [Installation notes](#)
- [Approved antennas and cables](#)
 - [For usage in Europe](#)
 - [Main antenna \(GNSS + Cellular\)](#)
 - [BT/WLAN antenna \(optional\)](#)
 - [For usage in North America](#)
 - [Main antenna \(GNSS + Cellular\)](#)
 - [BT/WLAN antenna \(optional\)](#)

Installation notes

The equipment shall be installed and operated according to the defined installation requirements including the minimum distance between the C-Hub TCU2 device's internal / external antenna and the nearest interior Class-A surface of 200 mm.

- Min distance for external antenna: 200 mm (20 cm)
- Min distance for internal antenna: 200 mm (20 cm)

The distance between any human body and the antennas must be minimum 20 cm.

Approved antennas and cables

The C-Hub TCU2 device underwent testing and homologation with the antennas and cables specified in this section.

It is only allowed to use the device in combination with one of the approved antenna/cable combinations. Cable length must be in the approved range.

In case usage of a different antenna/cable combination is required, additional re-homologation might be required – please contact Bosch for further clarifications.



Note on selection of new antenna system

The isolation between each frequency band (cellular, BT/WLAN and GNSS) of the antenna system needs to be as high as possible to avoid interference and intermodulation during co-existence testing. This is important to pass legal EMC requirements. This isolation also ensures proper operating performance and co-existent operation of the different systems cellular, BT/WLAN and GNSS.

For usage in Europe

Main antenna (GNSS + Cellular)

Taoglas Genesis

- Type: 3-in-1 patch antenna (2*Cellular, 1*GNSS), adhesive mount
- Manufacturer: Taoglas
- Part number: **MA240R.LBI.056.rb**
- Cable: integrated (cable length 2m)

BT/WLAN antenna (optional)

In case you additionally need a BT/WLAN antenna in combination with the Main antenna **Taoglas Genesis**, use:

Taoglas Dual WiFi antenna

- Type: Dual WiFi antenna, adhesive mount (2*BT/WLAN)
- Manufacturer: Taoglas
- Part number: **MA214.A.CG.001.sf**
- Cable: integrated (cable length: 400mm)

For usage in North America

Main antenna (GNSS + Cellular)

Taoglas Genesis

- Type: 3-in-1 patch antenna (2*Cellular, 1*GNSS), adhesive mount
- Manufacturer: Taoglas
- Part number: **MA240R.LBI.056.rb**
- Cable: integrated (cable length 2m)

Taoglas Genesis II

- Type: 5-in-1 patch antenna (2*Cellular, 1*GNSS, 2*BT/WLAN), adhesive mount
- Manufacturer: Taoglas
- Part number: **MA285.LBICG.026**
- Cable: integrated (cable length 2m)

BT/WLAN antenna (optional)

In case you additionally need a BT/WLAN antenna in combination with the Main antenna **Taoglas Genesis**, use:

Taoglas Dual WiFi antenna

- Type: Dual WiFi antenna, adhesive mount (2*BT/WLAN)
- Manufacturer: Taoglas
- Part Number: **MA214.A.CG.001.sf**
- Cable: integrated (cable length: 400mm)

No additional BT/WLAN antenna is necessary for Main antenna **Taoglas Genesis II**, as this is already included.



In case you need to use another antenna, please align with your Bosch customer success manager.

2.2 C-Hub backend overview for vehicle manufacturers

The C-Hub backend uses a cloud infrastructure, which offers a fully managed, secure, cloud-based integration platform for over-the-air services.

The already integrated cloud services are listed below.

- [Vehicle Provisioning - Device Commissioning](#)
- [Device Connectivity and Device Management](#)
- [Device Maintenance](#)
- [Remote Measurement](#)
- [Remote Diagnostics](#)
- [Data Integration](#)

2.2.1 Vehicle Provisioning - Device Commissioning



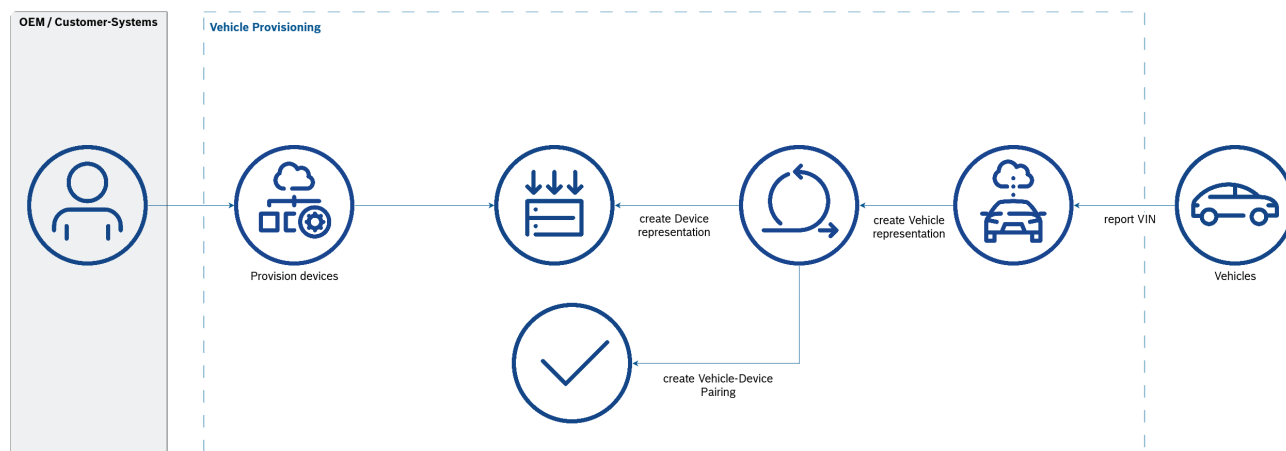
The **Vehicle Provisioning / Device Commissioning** service ensures **automatic creation** of digital representations of the connected vehicles.

This process has three main aspects:

- **Device Provisioning**
The devices are automatically registered to your *tenant* after your order and before being shipped to you. This registration ensures that only the certified devices of a customer can connect to a specific tenant.
- **Vehicle Creation**
The Vehicle Provisioning service creates automatically the digital representations of the vehicles based on the VIN received as part of the process.
- **Device-Vehicle Pairing**
When a device reports the detected VIN to the backend, the device is paired automatically with the digital representation of the vehicle.

How it works

- In order to connect to the C-Hub backend, a device must be registered to a specific tenant (representing the customer).
- This tenant has a configured [Certificate Authority \(CA\)](#), which ensures, together with the registered unique ID of the device, that only known and approved devices can connect.
- The registered device sends a compliant pairing message to the backend, which creates a digital representation of the vehicle in the system.



Steps applicable at C-Hub

1. A user or a system provisions the C-Hub TCU2 device on the C-Hub backend.
The provisioning process requires the *Integrated Circuit Card Identifier* (ICC-ID) and the *Serial Number* of the respective device.
2. As a result of the provisioning, a digital representation of the device is created and stored at the backend.
3. When mounted in the vehicle, the C-Hub TCU2 device sends a message to the backend to report the detected VIN.
4. When a C-Hub TCU2 device reports the detected VIN to the backend, the device is paired automatically with the digital representation of the vehicle.

Connectivity as a Service customers

Customers of the Connectivity as a Service offering - such as vehicle manufacturers - will have full access to a comprehensive user guide in order to fulfill the required steps.

Preliminary

2.2.2 Device Connectivity and Device Management



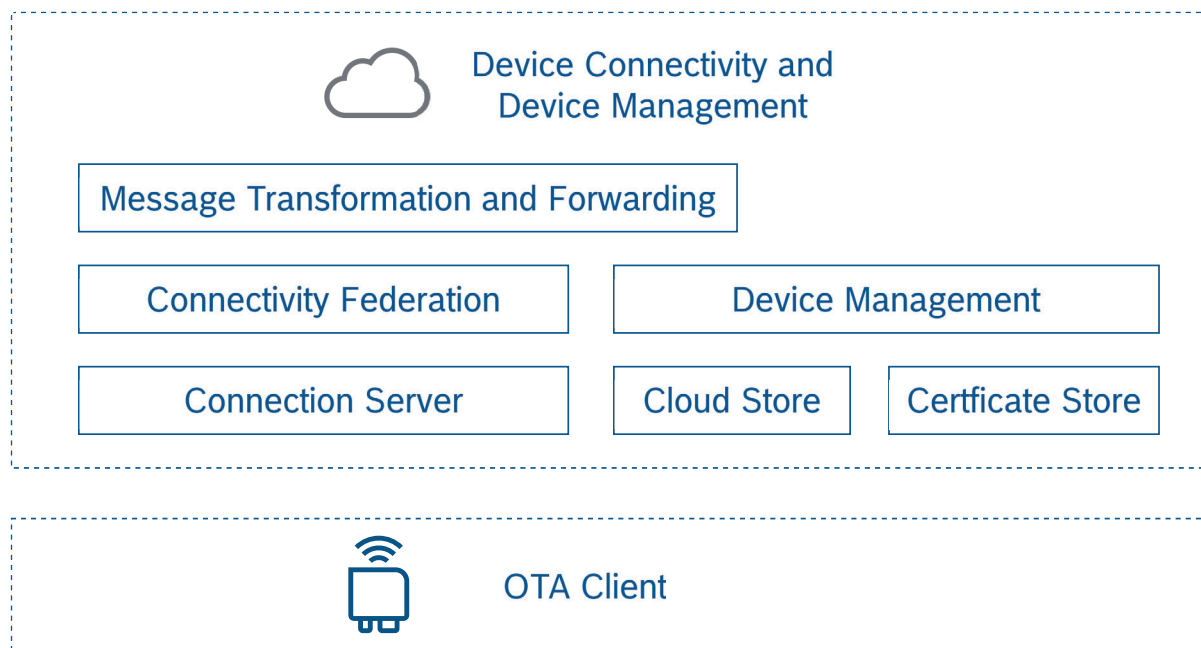
The **Device Connectivity and Management** service is a prerequisite for the C-Hub use cases and the proper functioning of the other C-Hub cloud services.

It arranges for the various aspects which enable the communication between the C-Hub TCU2 devices and the C-Hub backend, as well as for the creation, grouping and enrichment of the digital representations of the devices and their associated vehicles.

The main aspects of this service are presented below and are described in more detail in dedicated sections.

How it works

The following diagram illustrates the main building blocks of the Device Connectivity and Management service.



Preliminary

Referring to the elements on the diagram, bottom-up:

- The C-Hub TCU2 device, installed in the vehicle, communicates with the C-Hub cloud backend through an **OTA client** installed on the device.
- A **Connection Server** on the backend checks the identity and permissions of the device which requests to connect and either grants or rejects the connection.
- In particular, the server checks whether the certificate presented by the device matches the data configured in the **Certification Authority** of the respective tenant on the backend. The information about the authorized device certificates is stored in the **Certificate Store**.
- The **Connectivity Federation** allows devices with different protocols to connect to the backend and serves as an abstraction layer. The C-Hub TCU2 devices in particular connect via the MQTT protocol.
- The messages exchanged between the devices and the backend, as well as between the different services on the backend, and between the backend and external services (e.g., customer applications), are transformed in the **Message Transformation and Forwarding Service** to ensure a uniform and understandable format for all participants.
- The **Device Management** is the central registry of the connected devices. This service allows to create, update and delete digital representations of the physical C-Hub TCU2 devices and

to enrich them with attributes. In addition, it allows accessing device alert and device log files.

- The **Cloud Store** allows for exchange of large files. The devices and the backend can exchange only references to the artifacts (blobs) which can be uploaded to and downloaded from the cloud store respectively.

For further details on the building blocks and interfaces, please check the dedicated page which zooms in on the following aspects:

- [How it works on the backend](#)
 - [Connectivity Abstraction and Federation](#)
 - [Certificate Management](#)
 - [Device Management](#)
 - [Message Transformation & Forwarding](#)
 - [Apache Kafka Interface](#)
 - [Cloud Store](#)
- [How it works on the device](#)
 - [OTA Client](#)

Steps applicable at C-Hub

1. The device is installed in the vehicle and after successful authentication connects to the backend.
2. The device and vehicle digital representations are stored at the backend.
3. The vehicles can be organized in groups or be enriched with metadata.
4. All data received from the devices is stored in the cloud.
5. Customer applications get permission to access this data via API.

Connectivity as a Service customers

Customers of the Connectivity as a Service offering - such as vehicle manufacturers - will have full access to a comprehensive user guide in order to fulfill the required steps.

Device Connectivity and Device Management details

Table of contents

- [How it works on the backend](#)
 - [Connectivity Abstraction and Federation](#)
 - [Certificate Management](#)
 - [Device Management](#)
 - [Message Transformation & Forwarding](#)
 - [Apache Kafka Interface](#)
 - [Cloud Store](#)
- [How it works on the device](#)
 - [OTA Client](#)

How it works on the backend

Connectivity Abstraction and Federation

For C-Hub TCU2 devices, the protocol used for connection to and communication with the C-Hub cloud backend is MQTT.

This is a lightweight message protocol designed for efficient communication between physical devices in the field and applications.

This protocol is suitable for resource-constrained devices as it provides a simple publish-subscribe model, while minimizing network bandwidth.

Compared to other solutions, MQTT has a low overhead, support for unreliable networks and the ability to handle large numbers of messages and connected devices.

Certificate Management

A Certificate Authority (CA) is a trusted entity that verifies the identity of a device. It can be used to issue a unique certificate for each device.

Devices that want to connect to a particular tenant on the cloud backend have to present such a certificate.

On the backend side, the corresponding CA has to be configured for the tenant, so that the cloud backend can verify whether the device that wants to connect is authorized to do so.

Device Management

The digital representation of a connected device (and its associated vehicle) is managed on the cloud backend through the Device Management service.

The Device Management is the central registry for the devices that can connect to the system.

It allows to create, update, and delete devices and notifies neighboring systems via lifecycle events about changes of a device.

Further, it allows to manage custom attributes about devices, find the capabilities of a device, and filter for alarm messages that were sent by a device.

Message Transformation & Forwarding

The standardized and compatible communication between C-Hub TCU2 devices and the cloud backend is ensured by means of a behavioral contract, expressed in the form of Backend Functions Binding (BFB).

Its essential components are messages and sections. Each BFB section describes messages related to a specific service.

The messages exchanged between the cloud backend and the device are serialized as Protocol Buffers in a BFB compliant message.

Protobuf is a platform-neutral data serialization format, which allows to define the structure of data using a simple language.

It allows for efficient serialization and de-serialization of data for network communication, by providing a compact binary format that is smaller in size compared to plain JSON or XML.

On the cloud backend, the Device Connectivity takes care of translating the messages sent to or received from a device and makes them available as JSON for other cloud backend services.

Apache Kafka Interface

The data produced and sent by the connected C-Hub TCU2 devices is available at the provided Apache Kafka interface.

Apache Kafka is a distributed streaming platform designed to handle high-volume streams efficiently.

It follows the publish-subscribe pattern and provides a scalable system for publishing, subscribing and processing streams of data.

Producers of data write to topics and consumers read data from those topics. This allows for an event-driven architecture, where producers and consumers are decoupled and flexible in the way they process the data.

The event-based approach is the right solution for systems where it is uncertain when data will arrive, for example due to an offline device or unstable internet connection. This approach allows for a subscribed software system to get notified and to react to data received from the device at the moment it arrives.

Cloud Store

The Cloud Store is a secure and scalable storage service which allows to store and retrieve large arbitrary data (blobs).

Following the [claim-check pattern](#), it allows a C-Hub TCU2 device to upload large amounts of measured data without over-straining the Device Connectivity.

Thus the device can send only the reference to the uploaded data in the message which will be received by the cloud backend.

In a similar way, a device can receive a reference, for example to an available software update, and download it from the Cloud Store at its own pace, supported by resumable uploads and downloads.

How it works on the device

OTA Client

C-Hub TCU2 devices have an installed OTA client which handles the communication with the C-Hub cloud backend.

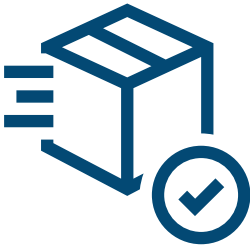
The OTA Client is the reference implementation for the Backend Functions Bindings (BFB) and is the counterpart of the cloud backend on the device.

Written with a flexible plugin architecture in mind, this client serves as a minimally-functional solution that only requires a small set of runtime components.

The OTA Client has the following features:

- Connects to any supported backend endpoint (e.g. using MQTT in the C-Hub case)
- Implements supported serialization protocols like [Protocol Buffers](#)
- Automatically de-/serializes messages exchanged with the backend
- Provides a flexible plugin-architecture on section level for connecting BFB semantics with specific in-vehicle stacks
- Can be extended to fit new requirements

2.2.3 Device Maintenance



The **Device Maintenance** service enables over-the-air (OTA) updates of C-Hub software on the C-Hub TCU2 device.

Regular self-updates of the device software will be rolled out automatically over the air to ensure the security and performance of the C-Hub TCU2 device.

Bosch reserves the right to update the C-Hub TCU2 devices in the field for various purposes, including but not limited to:

- Security updates
- Functional updates
- Bug fixing

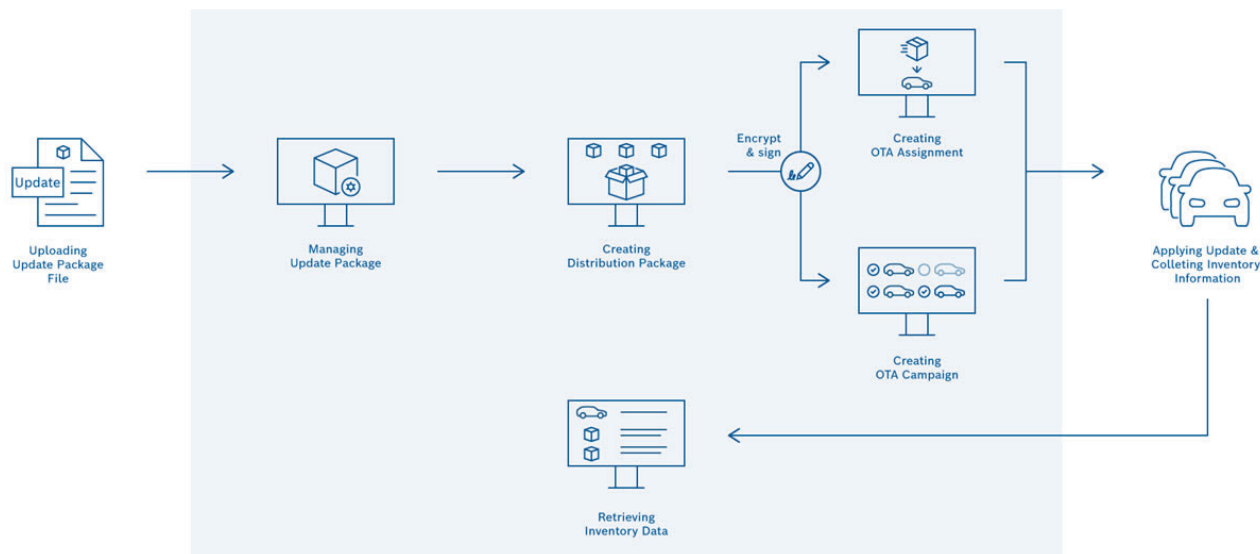
How it works

The *OTA Updates* user interface provides an overview of the OTA updates of your vehicles and devices.

Update packages can be uploaded, distribution packages can be prepared, OTA assignments for individual vehicles or devices can be created and OTA update campaigns for vehicle fleets can be managed.

An inventory provides an overview of the software versions of the entire fleet.

- **Uploading Update Package File**
An update package is created by uploading an update file to the backend.
- **Managing Update Package**
Manage the metadata of the update package in order to prepare it for the update.
- **Creating Distribution Package**
A distribution package can be created to bundle several update packages. An appropriate format must be specified for the distribution package so that the devices to be updated can recognize the distribution package. The distribution package must then be converted into a container that can be downloaded from the corresponding devices. The container is encrypted and signed by the backend.
- **Creating OTA Assignment or OTA Campaign**
To update a single vehicle or device, an OTA assignment needs to be created using the respective distribution package. After creation, the update process can be observed via the created OTA assignment.
If the entire fleet or only certain vehicles are to be updated using the same distribution package, a campaign can be created. With the filter function of the campaign, you can use the vehicle properties or the vehicle inventory to define which vehicles should be updated.
- **Uploading Update and Displaying Inventory**
The progress of an update on campaign level or on vehicle level can be monitored through the OTA Campaign view or the OTA assignment view, respectively.
The Vehicles view, especially the Vehicle Inventory sub-view shows which SW version is currently running on which vehicle.



Updates are distributed multiple times a year. Depending on the introduced changes, either a device software is updated completely (full update) or incrementally. Incremental update means that only the relevant parts are updated (delta update).

Compared to a full update, a delta update is more time and data efficient. In the rare case that an update could not be applied, the device is automatically rolled back to the most recent working version. Depending on the error, additional follow-up steps might be necessary.

In real world scenarios, the software to be distributed is not assigned manually to each and every device. Instead, it is assigned to multiple vehicles through a campaign.

With campaigns, it is possible to dynamically add and remove vehicles if they match defined criteria. There is the possibility to add complex vehicle filtering logic during assignment to a distribution and can assign start and stop dates for the rollout, check software compatibility versions, etc.

Steps applicable at C-Hub

1. Bosch will prepare and test the updates separately with no impact on the vehicles in the field.
2. The planned update will be communicated to the customers.
3. When the rollout is planned in campaigns, it might happen that not all of the devices of a customer are updated on the same day, because the vehicle would need to be powered so that the C-Hub TCU2 device itself is also on.
4. The updated software versions will be applied to each C-Hub TCU2 device, which is included in the campaign, after reboot, i.e. upon the next "ignition switch on" (KL15) of the respective vehicle.

C-Hub Connectivity as a Service customers

Customers of the Connectivity as a Service offering - such as vehicle manufacturers - who might need to interact with this service will have full access to a comprehensive user guide in order to fulfill the required steps. Thus they will be able to configure own updates in self-service mode.

However, the updates planned for device maintenance will be rolled out by Bosch.

2.2.4 Remote Measurement



The **Remote Measurement** service enables you to remotely retrieve signals from the vehicle's CAN bus, during the vehicle's operation.

- Simple configuration of required vehicle signals (J1939, CAN)
- Application of remote measurement configurations to individual vehicles in the field or complete vehicle fleets

How it works

The Remote Measurement service provides recorded data from signals and messages from a vehicle to the backend.

Remote Measurement measures data from CAN buses or TCU2 device internal signals, like e.g. GPS, while a vehicle is in operation. The recorded data is buffered on the device and transferred to the backend. The backend makes the data available for download or forwards it to third-party IT systems.

- **Uploading Description File and Creating Signal Collection**

A signal collection is created by uploading a bus description file (DBC) to the backend. A signal collection thus contains all the information required to measure signals and messages. It also enables the selection of the individual signals in the measurement configuration.

- **Creating Measurement Configuration**

In a measurement configuration, you can select and configure which signals and messages, originating from which buses, you want to be recorded.

The buses are represented as **sources** and are configured through the uploaded signal collection (DBC file).

Out of these sources, you can select the **signals and messages** which have to be measured. Then you can set individual sampling rates for each signal, in order to define the frequency of the data sampling.

In addition, you should configure a **trigger** in order to control the start and end of the recording and to ensure that only the relevant data is recorded.

- **Activating Measurement Configuration**

After assigning the measurement configuration to the vehicle representation, it is transferred remotely to the vehicle.

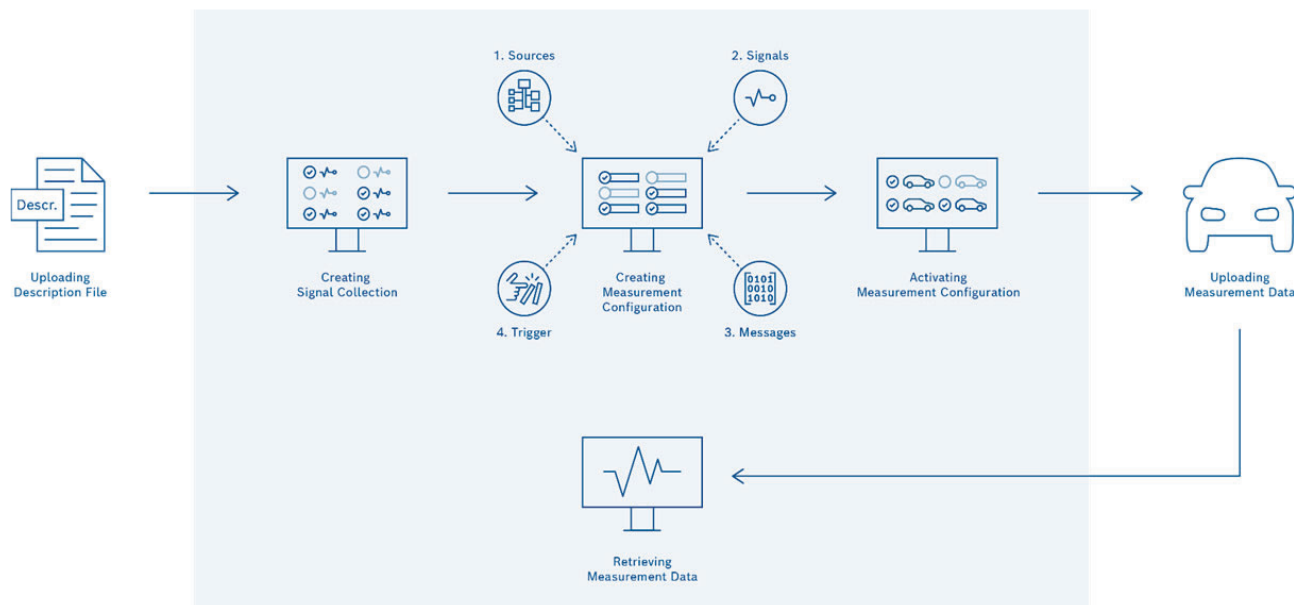
The resulting measurement job is always performed when the vehicle is in operation and continuously collects measurement data that is temporarily buffered in the vehicle.

- **Retrieving Measurement Data**

The recorded data is always transferred to the backend when a remote connection is established.

The data is saved in the backend and made available for preview and download.

Received data is automatically forwarded to connected third-party IT systems.



Steps applicable at C-Hub

- Before a measurement configuration can be created in the backend, there needs to be an agreement between:
 - the specialist responsible for the customer application which will consume the measurement data, and
 - the C-Hub support team (for Data as a Service customers) or the customer's IT professionals (for Connectivity as a Service customers) who will create the configuration specifying the required **data set** and the **frequency** of measurement.
- The configuration and its activation require *read* and *write* access on the respective C-Hub tenant. Changes should only be done by trained IT professionals who are aware of the consequences of potentially wrongly formatted configurations.
- The "consuming" customer application can retrieve data sent by the vehicle by subscribing to the respective Apache Kafka topics, or by requesting the data via the REST API.

C-Hub Connectivity as a Service customers

Customers of the Connectivity as a Service offering - such as vehicle manufacturers - will have full access to a comprehensive user guide in order to fulfill the required steps. Thus they will be able to configure the data set in self-service.

Remote measurement configuration

Vehicle manufacturers can employ their own IT experts to configure or change configurations of the data set to be collected.

Additionally to the user guides in the portal, we have prepared a set of videos to help you get familiar with the service.

Playlist:

- Introduction to measurement configurations
- Use case 1: Route recording using device signals
- Use case 2: Route recording including vehicle signals
- Use case 3: Route recording including a trigger
- Use case 4: Trip end message
- Use case 5: Trip end message including statistics | Preparation and overview
- Use case 5.1: Trip end message including statistics | Speed
- Use case 5.2: Trip end message including statistics | Time in idle
- Use case 5.3: Trip end message including statistics | EngineSpeed histogram
- Use case 5.4: Trip end message including statistics | EngineSpeed EngineTorque heatmap

Preliminary

2.2.5 Remote Diagnostics



The **Remote Diagnostics** service enables the remote retrieval of preconfigured signals from the vehicle's OBD-II interface, during its operation.

- Access to and configuration of diagnostic data (OBD-II, UDS, J1939)
- Application of remote diagnostic scripts to individual vehicles in the field or complete vehicle fleets

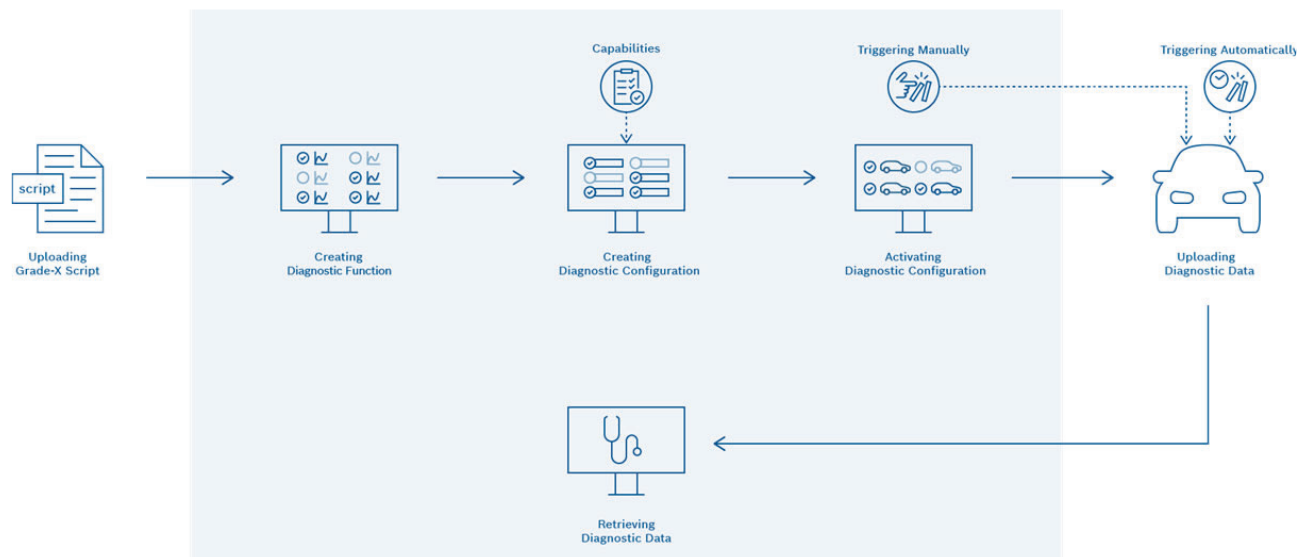
How it works

Remote Diagnostics allows you to perform basic diagnostic functionality on a vehicle during its operation. The functionality is controlled from the backend and the results are provided in the backend.

The diagnostic data, such as OBD II, UDS, and J1939 is collected on-board of the vehicle, and is buffered on the C-Hub TCU2 device, which is installed in the vehicle.

The collected diagnostic data is then transferred to the backend and can be accessed by authorized third-party IT systems via API.

- **Uploading Diagnostic Script and Creating Diagnostic Function**
 - A diagnostic function is created by uploading a container file which contains the diagnostic script.
A diagnostic function thus contains all the information that is necessary to execute a specific diagnostic task.
 - The diagnostic function always provides a basic diagnostic operation, e.g. read a Diagnostic Trouble Code (DTC).
- **Creating Diagnostic Configuration**
 - Each diagnostic configuration provides a predefined set of diagnostic capabilities.
 - For each capability a diagnostic function can be assigned.
 - The assigned function defines the activity which will take place when the diagnostic capability is triggered.
- **Assigning Diagnostic Configuration**
 - When the diagnostic configuration is assigned to the digital vehicle representation, it is transferred remotely to the physical vehicle.
 - The corresponding diagnostic capabilities are installed on the C-Hub TCU2 device.
 - The installed diagnostic capabilities are triggered cyclically in the vehicle or by the backend as required.
 - The collected result data, e.g. DTC and freeze frames, is temporarily buffered in the vehicle.
- **Retrieving Diagnostic Data**
 - The recorded diagnostic data is always transferred to the backend when a remote connection is established.
 - The data is saved in the backend and made available in the UI for preview.
 - The data can be accessed by authorized third-party IT systems via API.



Steps applicable at C-Hub

- Before a diagnostic configuration can be created in the backend, there needs to be an agreement between:
 - the specialist responsible for the customer application which will consume the diagnostic data, and
 - the C-Hub support team (for Data as a Service customers) or the customer's IT professionals (for Connectivity as a Service customers) preparing the configuration for this service need to agree on the data set which is required and its cadence.
- The configuration and its activation require *read* and *write* access on the tenant, and should only be done by IT professionals who are aware of the consequences of potentially wrongly formatted configurations.
- The authorized "consuming" customer application can access the data via API.

Connectivity as a Service customers

Customers of the Connectivity as a Service offering - such as vehicle manufacturers - will have full access to a comprehensive user guide in order to fulfill the required steps.

Thus they will be able to configure the data set in self-service.

Details on how the data consuming application can access the tenant data are described in following sections:

- [Data consumption by customer application via REST API](#)
- [Data consumption by customer application via Apache Kafka PubSub](#)

Preliminary

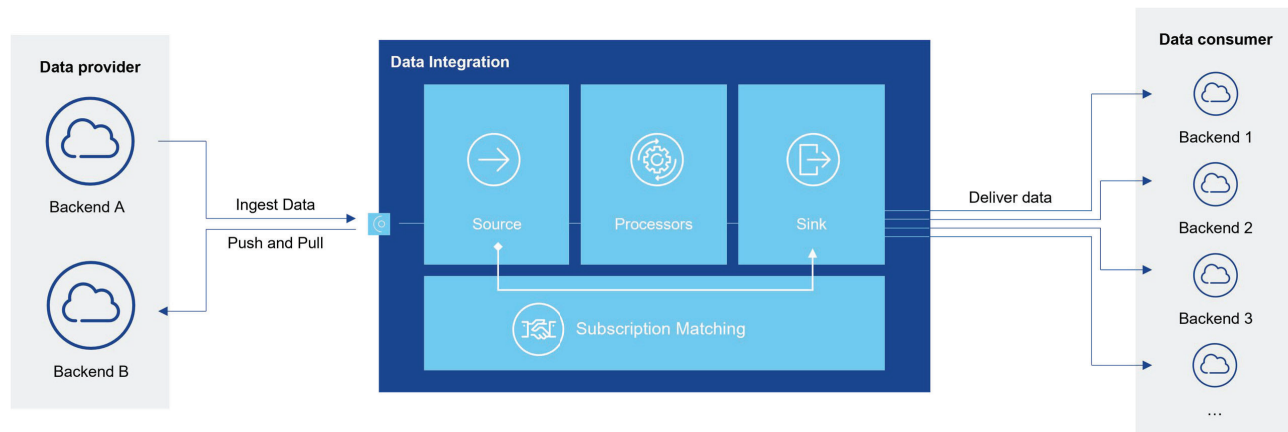
2.2.6 Data Integration



The **Data Integration** service provides easy access to your vehicle data via fully managed data pipelines. It allows producers as well as consumers of data to use different channels for data distribution.

How it works

- **Data Ingest** - There are multiple possibilities to ingest data into the system. The C-Hub TCU2 device is already configured respectively.
- **Metadata Extraction** - Based on the customer need, fields of the payload are extracted and added to the message as metadata.
- **Subscription Matching** - Every consumer needs a subscription, which defines the ingested data that should be forwarded to them.
- **Consumer Notification** - With a subscription it is also defined where messages should be sent to. This includes a wide variety of different storage systems, from blob storages to data analytics platforms.



Steps applicable at C-Hub

For C-Hub customers who want to use the service, the vehicle data is ingested into the Data Integration service.

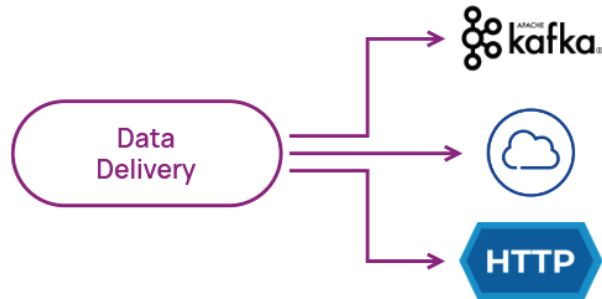
While C-Hub backend is in the role of the data provider, additionally there needs to be an agreement regarding which data should be forwarded to your data consuming application.

1. Data providers and data consumers are both registered as tenants (in the Bosch identity and authorization service IAM) in order to establish data exchange relationships.
2. The subscription matching is an informal agreement with the C-Hub support team regarding the preferred data sink and the handover of all required configuration parameters which are needed in order to establish a connection to your system.
The type of credentials for authorization / authentication differs depending on the preferred data sink. Find details at [Data consumption for fleet operators and mobility providers](#).

Preliminary

Flexible data delivery via rich notification channels

Data is automatically routed to consumers and sent to the system of their choice. Various target systems are available, for example Azure Blob Storage, AWS S3, AMQP 1.0, a generic HTTP POST, Bosch IoT Insights.



Find details at [Data consumption for fleet operators and mobility providers](#).

Data in transit

When interfacing with Data Integration, all traffic is encrypted using TLS 1.2 with only strong cipher suites enabled.

We follow the recommendations of the Open Web Application Security Project (OWASP).

Prerequisites and dependencies

To be able to use Data Integration, the following prerequisites have to be fulfilled:

- Information on processed data is required to define a data security and privacy classification prior to usage.
- Creation of tenant and technical users is conducted upon project request.

Known limitations

The following items are known limitations at the moment:

- The overall request size of a message is currently limited to 10 MB.
- It is not permitted to conduct any penetration testing or similar activities.
- Load or performance tests that exceed the regular use of the systems significantly have to be discussed upfront and planned accordingly.

C-Hub Connectivity as a Service customers

Customers of the Connectivity as a Service offering - such as vehicle manufacturers - can be served by this service and have additional options.

See [Data consumption options](#).

2.3 Customer applications communicating to the C-Hub backend

Your application is the actual consumer of the data which is collected from the connected vehicles.

In such an application, the vehicle data can be stored, processed by custom algorithms, as well as visualized in a custom dashboard, depending on the type of solution you already have or want to create.

Robert Bosch GmbH, for example, offers such applications in the areas of Vehicle Health, Safety and Security, as well as Efficiency.

The vehicle data can be directly consumed by the respective application(s) in a backend-to-backend communication mode through the APIs of the C-Hub cloud services.

After initial onboarding by the C-Hub support team, the customer's IT professionals can use the data consumption APIs described below.

Support may be provided additionally upon request.

Learn more in the dedicated sections:

- [Data consumption options](#)
- [Data consumption for fleet operators and mobility providers](#)
- [Data consumption for OEMs](#)
 - [Data consumption by customer application via Apache Kafka PubSub](#)
 - [Data consumption by customer application via REST API](#)
 - [Data consumption by customer application via Data Integration service](#)

2.4 Authentication and authorization

To enable secure and seamless data flows from the connected vehicles to the C-Hub backend and from this backend to your applications, several actors need to identify themselves respectively.



C-Hub TCU2 device



C-Hub backend



User



Customer solution or another third-party application

2.4.1 C-Hub TCU2 device



The C-Hub TCU2 device installed in the vehicle communicates with and authenticates itself to the C-Hub backend.

Type of authentication

The C-Hub TCU2 device has a TLS certificate installed.

The Certificate Authority (CA), which verifies the device identity based on this certificate, is configured by the C-Hub support team.

Permission examples

- Get software and firmware updates OTA
- Install OTA updates received from the cloud backend
- Get new configurations for the data set to be collected from the vehicle's CAN bus or OBD II interface
- Send data to the cloud in the format and cadence scripted in the remote measurement configuration or in the remote diagnostics script

2.4.2 C-Hub backend



The C-Hub TCU2 device communicates with various cloud services on the C-Hub backend

C-Hub cloud services:

- Vehicle Provisioning
- Device Connectivity and Device Management
- Device Maintenance
- Remote Measurement
- Remote Diagnostics

These services communicate with each other as described below.

Type of authentication

Service instances booked for the same tenant are allowed to inter-communicate. They authenticate by using the same mechanism which is used for authentication by external services. For example, the OAuth2 client credential flow is used for REST-based communication.

Permission examples

- Ingest data from authorized devices
- Store data sent by the device
- Send notifications to other cloud service instances which belong to the same tenant
- Send configuration updates to the devices in the field
- Send software and firmware updates to the devices in the field
- Ingest data from other cloud service instances which belong to the same tenant
- Expose APIs to authorized technical clients

2.4.3 User



In certain cases, human users may be provided with access to a Web UI through which they can interact with the C-Hub backend and the C-Hub TCU2 devices.

In certain cases, human users may be provided with access to a Web UI through which they can interact with the C-Hub backend and the C-Hub TCU2 devices.

Such users could be, for example, members of the C-Hub Operations and Support team (for the Data as a Service scenario), or members of the vehicle manufacturer team (for the Connectivity as a Service scenario).

They must be registered users of the [Bosch SingleKey ID](#) service, which are additionally assigned to a specific tenant of the C-Hub backend.

Type of authentication

The users perform basic authentication on the UI with username and password.

To access any resources there, these users need to be granted the relevant roles by the tenant owner, e.g.:

- read and write access
- read-only access

Permission examples

- Read all data displayed at the UI, e.g. current list of registered devices and vehicles.
- Edit some data displayed at the UI, e.g. create groups of vehicles.

2.4.4 Customer solution or another third-party application



A customer solution or another third-party application which wants to communicate with the C-Hub backend is treated as a technical client. All technical clients need to be registered with the respective C-Hub tenant.

Type of authentication

For using the **REST API**

- An OAuth 2.0 token - client credentials flow - see <https://oauth.net/2/>.
- Scopes specific to the use case will be granted so as to enable access to the relevant APIs.

- The client credentials are provided by the C-Hub support team.

For using the **Apache Kafka messaging**

- A PubSub client with Client ID, API Key, and API Secret
- The client credentials are provided by the C-Hub support team.

Permission examples

- Get data retrieved from a vehicle.
- For using the REST API
 - Interact with the exposed cloud services APIs respecting the scope
- For using the Apache Kafka messaging
 - Subscribe for specific topics

3 C-Hub TCU2 device installation

For collecting and transferring vehicle data to the cloud, the Automotive Connectivity Hub provides the C-Hub TCU2 device and various accessories.

3.1 Installation concept

Step	What
#1	Bosch registers the devices of the respective customer with the C-Hub backend.
#2	The C-Hub TCU2 devices are delivered together with accessories, which vary depending on the customer group.
#3	The customer's installation team mounts each C-Hub TCU2 device into a DIN Adapter, connects it with a compliant antenna, and provides uninterrupted power supply for at least 45 minutes. Once the C-Hub TCU2 device is powered up, it automatically retrieves the latest software update from the C-Hub backend. This first self-update may take about 45 min, therefore uninterrupted power supply to the device for this period is required.
#4	Once the C-Hub TCU2 device is installed into the vehicle, the data collected from the vehicle can be transmitted to the C-Hub backend. This happens automatically and only when the vehicle is powered up. From the C-Hub backend, the customer's authorized applications can retrieve the collected data via API.

Preliminary

3.2 Installation concept for telematics service providers (TSP)



- The initial installation into the vehicle is performed by a trained installation partner.
- The data which needs to be collected can be described by the customer in a human-readable form. The requirements will be then scripted, and the configuration will be deployed on the customer's C-Hub TCU2 devices by Bosch.



A detailed **installation guide** will be provided as a dedicated document upon request.

3.3 Installation concept for vehicle manufacturers (OEM)



- Installation into the vehicle is to be done by the vehicle manufacturer in self-service mode.
- The data which needs to be collected from the vehicles can be scripted by IT professionals from the vehicle manufacturer's team and the configuration will be installed on the customer's C-Hub TCU2 devices in self-service mode.

3.4 Further hints on the installation concept for vehicle manufacturers (OEM)

Table of contents

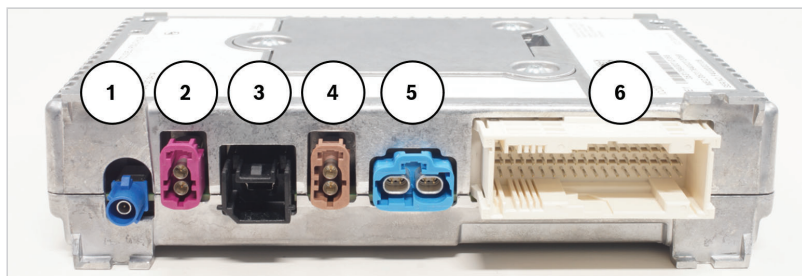
- [3.4.1 C-Hub TCU2 device - all connectors](#)
- [3.4.2 C-Hub TCU2 device - main connector](#)
- [3.4.3 Antennas](#)

3.4.1 C-Hub TCU2 device - all connectors

The following connectors are available on the back of a C-Hub TCU2 device:

1. GNSS antenna
for GPS, Galileo, Beidou
Rosenberger FAKRA HF Type C
2. Mobile network antenna
for LTE
Rosenberger FAKRA-HFM Type D, vertical dual connector
3. USB mini
Molex USCAR-Header Type Mini-B
4. Mobile network antenna
for WLAN and/or Bluetooth antenna
Rosenberger FAKRA-HFM Type F, vertical dual connector
5. Ethernet port
for development and testing purposes only
double-plug, 100BASE-T1 (automotive / 2-wire)
Rosenberger H-MTD Type C
6. **Main connector** for the main wiring harness
Tyco MQS Header, 54 poles Code B

Preliminary



Note: Bosch is only responsible for the compliance of the product side plug (interface) with the agreed upon customer specification.

Since the plug system is used per customer request, Bosch is not responsible and does not warrant for the connection assembly, especially not for its electrical function, durability, and sealing.

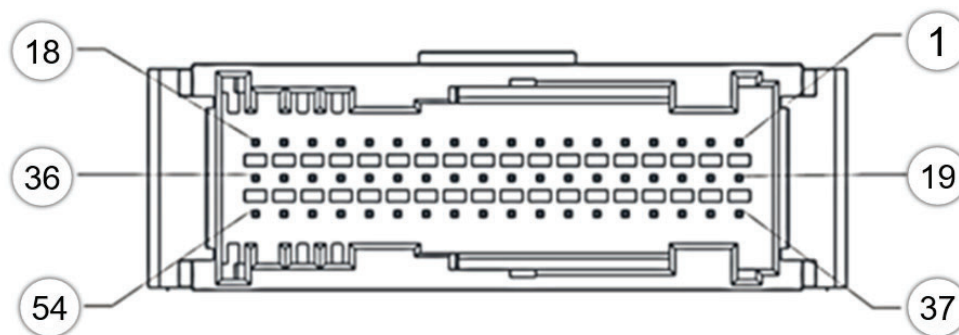
3.4.2 C-Hub TCU2 device - main connector

Customers who need to create a wiring harness, which deviates from the standard main wiring harness designed for retrofit use cases, can order this part from Bosch.

Please reach out via your Bosch sales partner or via the C-Hub support email channel.

Type designation	TCU2-IP30-MAIN-CONNECTOR-PARTS-SET-C-HUB
Main connector	Product part number 0261V1004C

The **main connector on the C-Hub TCU2 device** has the following design and pin designation.



See Annex > [Main connector - pinning info](#) for a table of all pins.

3.4.3 Antennas

The C-Hub TCU2 device underwent homologation and testing with the cables and antennas specified in the Annex.

See Annex > Auxiliary equipment - antenna and cabling EU and NA

i In case you prefer or need to use another antenna, please contact us via the C-Hub support email and Bosch will re-homologate the C-Hub TCU2 device with the antenna which best fits your vehicle.
Keep in mind however, that homologation will require time and will lead to extra costs.

Preliminary

4 Data configuration concepts

Thanks to the online collection of data, any authorized business services can obtain the required vehicle data for their solutions.

The discussion and decisions about which vehicle data is needed for the consumer application can be started at a very early stage.

For customers of a telematics service provider, we recommend that the telematics service provider **specifies the data** to be obtained from the **FMS** (CAN bus) or **OBD-II** interface and asks the C-Hub team for assistance.

Example:

- required data set
- sampling frequency
- upload frequency
- the specific event, in case the upload should be event-based

Based on the requirements, the Bosch engineers will create the configuration and deploy it to the C-Hub TCU2 devices as defined in the agreed deployment strategy.

To change the desired measurement configuration, Bosch can perform configuration updates over the air even separately from the software updates, or update them together with the next software update using the campaign management functionality.

Details and further hints

During the vehicle startup, as part of the boot process, it normally takes a few minutes until the measurement configurations are loaded and the measurement agent, responsible for data acquisition, is up and running.

During this boot time (to avoid losing any data), the CAN data is buffered and is processed once the agent is up and running.

While the device is running, the measurement agent will acquire the configured data and will prepare it for sending to the cloud backend.

Depending on the availability of the mobile network, it might not be always possible to send out the data.

- In such a case, when the device is not online (i.e. when network connectivity is down), the data is stored on the device until a mobile network connection can be established again.
- Similar to above situation, if the cloud backend becomes unavailable, the data will be stored on the device and will be sent once the cloud backend is available.

The update of measurement configurations involves deactivation of the old version and activation of the new version.

This means that the data collection (for the old version) is stopped, and data collection starts again after the new version is activated.

The deactivation and activation of the new version is done as one step on the device to reduce data loss.

The option to update the version only after a reboot exists. However, this is not configurable per measurement configuration, but only on tenant level.

During the vehicle/device shutdown, the measurement agent will be stopped, so that no further data can be acquired.

However, any data that has already been acquired and processed will be uploaded eventually.

As we cannot guarantee that the upload happens before the device shutdown, it might happen that the upload proceeds in the next boot cycle.

- If the network connectivity becomes unavailable meanwhile, this upload package will reside on the device.
- The upload will take place during the next drive cycle.



In fact, the configuration can be adjusted anytime, however, changing the configuration in self-service mode is offered only for customers who have subscribed the Connectivity as a Service contract, i.e. OEMs.

4.1 FMS standard parameters

The FMS interface is an **optional** interface in trucks and buses of different manufacturers, like DAF, Daimler, IVECO, MAN, Renault, Scania.

This interface enables C-Hub TCU2 devices to **securely** collect vehicle data, whereas the C-Hub backend enables the sharing of such data with third parties via cloud services.



The complete FMS standard is available online for free
https://fms-standard.com/Truck/download/fms%20document_v_05_vers.11.11.2025.pdf

Find an excerpt in the **Annex** > [FMS standard excerpt](#).

4.2 OBD-II diagnostic data

OBD, or on-board diagnostics, is a term used to describe a vehicle's ability to self-diagnose and report on its various systems.

OBD-II is the current standard for on-board diagnostics, specifying the type of diagnostic connector, electrical signaling protocols, and messaging format.

This system allows to get data from all control units that can be accessed via the OBD-II port of the vehicle, providing valuable information for troubleshooting of vehicle issues.

The SAE J1979 standard defines a method for requesting diagnostic data and a list of standard parameters available from the vehicle. Manufacturers are not required to implement all of these parameters and can include proprietary ones.

The OBD-II system provides access to real-time performance data and flagged Diagnostic Trouble Codes (DTCs), as well as various vehicle information parameters such as VIN, emissions control system counters etc.

In general, the following vehicle information should be available via the OBD-II:

- Vehicle Identification Number (VIN)
- Calibration Identification Number
- Ignition counter
- Emissions Control System counters
- Diagnostic Trouble Codes (DTCs) for
 - Powertrain (Engine and transmission)
 - Emission Control Systems

You can find a list of OBD-II data, including parameters and their definitions, in the SAE J1979 standard (see https://www.sae.org/standards/content/j1979_201702/).

Additionally, you can refer to resources provided by vehicle manufacturers, automotive diagnostic tool manufacturers, or automotive repair manuals for specific lists of OBD-II data and parameters. There are also online databases and resources that provide comprehensive lists of OBD-II data and parameters for various vehicle makes and models.

We can run any diagnostic services based on DoCAN (ISO 15765).

In particular, the following overlying protocols are supported:

- ISO 14229 (UDS on CAN),
- ISO 15031 (OBD on CAN) series,
- SAE J1979 (OBD on CAN),
- SAE J1979-2 (OBD on UDS), and

- ISO 27145 (WWH OBD on CAN).

Other physical layers, such as DoIP, are currently not supported.

Data transfer of diagnostic results to the cloud is limited by size and pre-determined data structures in remote diagnostics.

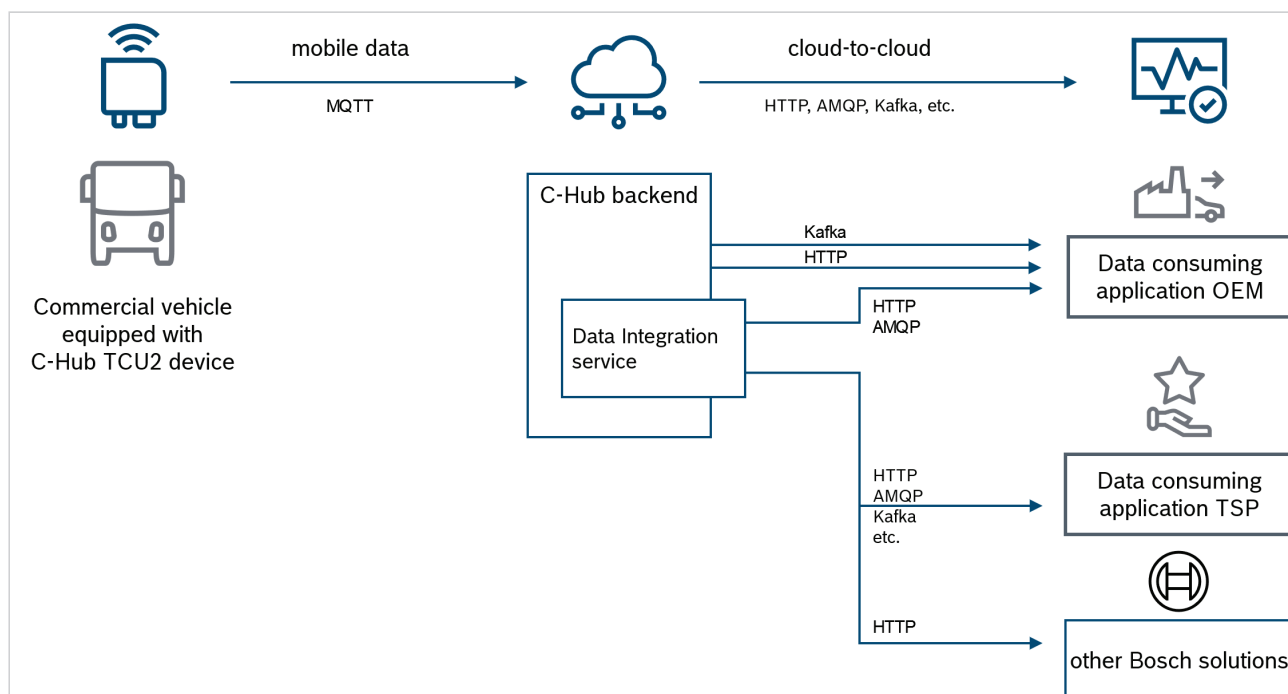


Diagnostic data is typically consumed by vehicle health services. The Cloud Diagnostics vehicle health service provided by Bosch is already integrated with C-Hub. See <https://www.bosch-mobility.com/en/solutions/software-and-services/vehicle-health-services/>

5 Data consumption options

The C-Hub backend provides various options to forward the vehicle data to your data-consuming application.

The following diagram depicts the main data flow from device to a solution.



Preliminary

5.1 Data consumption for fleet operators and mobility providers

The Data Integration service supports the following interfaces as a data sink:

- 5.1.1 AMQP
- 5.1.2 AWS S3
- 5.1.3 Azure Blob Store
- 5.1.4 HTTP based API or backend
- 5.1.5 Bosch IoT Insights data recorder API
- 5.1.6 Apache Kafka



In order to establish the connection, the C-Hub support team will require the details outlined in the "Required configuration" section.

5.1.1 AMQP

Supported infrastructure	Active MQ IBM MQ Rabbit MQ Azure Event Hub Solace
Required configuration	▪ Broker URL ▪ e.g. <i>amqp://my-amqp-broker.amqp:5672</i> ▪ Topic to send data to ▪ Authentication mechanism ▪ Password ▪ Certificate ▪ Username ▪ Password ▪ Certificate (in case of certificate-based authentication) ▪ Compression on / off ▪ Can compress data before sending it
Capabilities	▪ Supports protocol AMQP 1.0
Default meta data	Meta data will be attached as AMQP header. ▪ uid - random ID attached to each outgoing message ▪ messageId - the internal unique ingest ID identifying data throughout the Data Integration system

Preliminary

5.1.2 AWS S3

Supported infrastructure	Amazon S3 Minio
Required configuration	▪ Endpoint ▪ e.g. <code>s3.eu-central-1.amazonaws.com</code> ▪ Region ▪ e.g. <code>eu-central-1</code> ▪ URL path style ▪ defines if path style for the bucket URL should be used ▪ true: <code>aws.s3.com/bucket</code> ▪ false: <code>bucket.aws.s3.com</code> ▪ Authentication ▪ AWS-IAM ▪ Access Key ▪ Access Secret ▪ Processor ▪ Compression ▪ <code>gzip/zip/none</code> ▪ Forwarder configuration ▪ Match configuration ▪ which cloud event attribute should be matched ▪ Forwarding configuration ▪ bucket ▪ endpoint ▪ in case a different endpoint should be used than the root one ▪ filename pattern ▪ e.g. <code>getExtension('ingestuid') + '.bin'</code> resulting in <code>9823934-3424-2342.bin</code> ▪ duplicate prevention
Capabilities	▪ Can upload data to one or different S3 bucket combinations ▪ Data blob name can be dynamically built from header fields ▪ Data can be compressed before uploading it to the S3 bucket
Default meta data	▪ <code>dianadatastreamid</code> (indicates how the data was entering the system e.g. http-ingest) ▪ <code>dianadatasize</code> (data size in bytes) ▪ <code>dianaproducertenant</code> (tenant that owns/produced the data) ▪ <code>dianaconsumertenant</code> (tenant that consumes the data) ▪ <code>dianaingesttimestamp</code> (timestamp of the data ingestion into the Data Integration system) ▪ <code>dianaingestuid</code> (unique ID representing the data itself throughout the system) ▪ <code>producerclient</code> (in case the data was ingested via HTTP, the OAuth client name) ▪ Content-Type (type of the content derived from the ingested data) ▪ Content-MD5, dianamd5hash (MD5 hash of the data)

5.1.3 Azure Blob Store

Supported infrastructure	Azure Blob Store Azurite Dev Container
Required configuration	- Container name - e.g. data - Connection URL - e.g. <i>storageaccountfoo.microsoft.com</i> - Blob naming pattern <code>getExtension('ingestuuid')</code> + '.bin' resulting in e.g. 9823934-3424-2342.bin - Duplicate prevention on/off - Detects duplicated blob with the help of an Azure feature - Authentication type - Azure AD / Service Principal - Shared Key - In case of Shared Key: - Account name - Account key - In case of Azure AD / Service Principal - Client ID - Client Secret - Tenant ID - Compression on/off - Can compress data before uploading it to the blob store
Capabilities	- Can upload data to one or different storage account and container combinations - Data blob name can be dynamically built from header fields - Data can be compressed before uploading it to the blob store
Default meta data	dianadatastreamid (indicates how the data was entering the system e.g. http-ingest) dianadatasize (data size in bytes) dianaproducertenant (tenant that owns/produced the data) dianaconsumertenant (tenant that consumes the data) dianaingesttimestamp (timestamp of the data ingestion into the Data Integration system) dianaingestuuid (unique ID representing the data itself throughout the system) producerclient (in case the data was ingested via HTTP, the OAuth client name) - Content-Type (type of the content derived from the ingested data) - Content-MD5, dianamd5hash (MD5 hash of the data)

5.1.4 HTTP based API or backend

Supported infrastructure	Any HTTP based API or backend
Required configuration	- Target backend URL - e.g. <code>http://foo.bar/data</code> - Authentication type - Basic Authentication/OAuth - Username - Password - In case of OAuth - IDP (identity provider) token URL - Scope - Grant-Type: Only <code>client_credentials</code> supported
Capabilities	- Can upload data to exactly one endpoint - Performs POST requests - The content type is taken over from the originally ingested data - The response is only evaluated to detect errors (4XX, 5XX status codes), response itself is discarded
Default meta data	- No meta data is forwarded, because it is a generic component and not every API would accept additional headers - Content-Type (type of the content derived from the ingested data)

5.1.5 Bosch IoT Insights data recorder API

Supported infrastructure	Bosch IoT Insights instance
Required configuration	- IoT Insights data recorder project URL - e.g. <code>https://bosch-iot-insights.com/data-recorder-service/v2/abc1234</code> - Authentication - IoT Insights username - IoT Insights password
Capabilities	- Can upload data to exactly one IoT Insights project - Supports basic authentication as provided by IoT Insights
Default meta data	<code>dianadatastreamid</code> (indicates how the data was entering the system e.g. http-ingest) <code>dianadatasize</code> (data size in bytes) <code>dianamd5hash</code> (MD5 hash of the data) <code>dianaproducertenant</code> (tenant that owns/produced the data) <code>dianaconsumertenant</code> (tenant that consumes the data) <code>dianaingesttimestamp</code> (timestamp of the data ingestion into the Data Integration system) <code>dianaingestuuid</code> (unique ID representing the data itself throughout the system) <code>producerclient</code> (in case the data was ingested via HTTP, the OAuth client name) - Content-Type (type of the content derived from the ingested data) E.g. <code>application/json</code>, or in case a compression is configured, it will be <code>application/gzip</code> or <code>application/zip</code> - Content-MD5 (MD5 hash of the data)

5.1.6 Apache Kafka

Supported infrastructure	Azure Event Hub Kafka Mode AWS MKS Kafka Mode Native Kafka Broker (e.g. Confluent)
Required configuration	▪ Bootstrap Server URL ▪ e.g. <i>my-kafka-cluster.kafka:9092</i> ▪ Topic to retrieve data from ▪ Authentication mechanism ▪ SASL ▪ PLAINTEXT ▪ Profiles: ▪ sasl ▪ aws ▪ mtls ▪ eventhub ▪ Username ▪ Password ▪ Certificate (in case of certificate based authentication) ▪ Event Hub Mode Authentication: ▪ Azure Tenant ID ▪ Azure Client ID ▪ Azure Client Secret
Capabilities	▪ Supports Azure Event Hub Kafka Mode ▪ Supports AWS MKS Kafka Mode ▪ Supports usage of mTLS certificates ▪ Supports creation of Kafka headers from event meta data
Default meta data	Meta data will be attached as Kafka header: ▪ dianadatastreamid (indicates how the data was entering the system e.g. http-ingest) ▪ dianadatasize (data size in bytes) ▪ dianamd5hash (MD5 hash of the data) ▪ dianaproducertenant (tenant that owns/produced the data) ▪ dianaconsumertenant (tenant that consumes the data) ▪ dianaingesttimestamp (timestamp of the data ingestion into the Data Integration system) ▪ dianaingestuuid (unique ID representing the data itself throughout the system) ▪ producerclient (in case the data was ingested via HTTP the OAuth client name) ▪ Content-Type (type of the content derived from the ingested data)

Preliminary

5.2 Data consumption for OEMs

Customer applications which need to consume data from the C-Hub backend can use the C-Hub backend APIs.

- [Data consumption by customer application via Apache Kafka PubSub](#)
 - [Example - getting started with Kafka](#)
- [Data consumption by customer application via REST API](#)
 - [OAuth2.0 Based Authorization](#)

- [RESTful API docs](#)
- [Example - data consumption by customer application via REST API](#)
- [Data consumption by customer application via Data Integration service](#)

5.2.1 Data consumption by customer application via Apache Kafka PubSub

The information generated and transmitted by the connected C-Hub TCU2 devices can be accessed through the provided Apache Kafka interface.

Apache Kafka is a distributed streaming platform that is specifically designed to efficiently handle high-volume streams of data. It operates on a publish-subscribe pattern and offers a scalable system for publishing, subscribing, and processing data streams. Data producers write to topics, and data consumers read from those topics. This enables an event-driven architecture, where producers and consumers are independent and adaptable in their data processing methods.

This event-based approach is ideal for systems where the timing of data arrival is uncertain, such as when dealing with offline devices or unstable internet connections. It allows a subscribed software system to receive notifications and respond to the data from the device as soon as it is received.

For Apache Kafka hosting, the C-Hub backend relies on *Confluent Cloud* as the SaaS provider. It enables real-time data processing, event-driven architectures, and seamless integration between various components in a distributed system.

- [Environments](#)
- [Provisioning of PubSub](#)
 - [Onboarding](#)
 - [Access to topic](#)
 - [Client resource](#)
 - [Consumer group for your consuming client](#)
 - [Topics for async API](#)

Environments

The following PubSub environments are available:

PubSub Instance	Purpose
prod-eu	Production cluster for region: West Europe
prod-us	Production cluster for region: North America

Provisioning of PubSub

This section describes the PubSub onboarding process for API providers and consumers.

PubSub implements the publish-subscribe pattern, where publishers send messages to topics without knowledge of any subscribers. Subscribers express interest in messages by subscribing to topics.

- PubSub implements a topic-based system where messages are published to topics.
- A Publisher can publish a message to a single topic only
- Subscribers in a topic-based system will receive all messages which are being published to the topics to which they are subscribed.

The PubSub provisioning is based on so called **manifests** which specify the resources to be provisioned within the PubSub infrastructure.

By resources in the context of PubSub we mean: **Topics**, **Clients** (incl. credentials) and **Access policies**.

The schemas for those manifests are YAML based.

Onboarding

To onboard your product to PubSub, you have to be familiar with the provisioning concept.

- As a provider, normally you offer a Kafka topic to other services so that they can consume your data.
- As a consumer, you rely on data from other services to do your business.

To onboard your product, you have to specify your topics resource. Your client will be able to access your topic, and under specific conditions also topics from other products.



Upon request through the email channel, the C-Hub support team will assist you during your on-boarding phase and provide you with:

- a Client ID, API-Key & API-Secret for your PubSub client
- the bootstrap-URL to the Kafka cluster for your region and
- the topic template for connection

Access to topic

To get access to topics as a PubSub consumer, you need a dedicated PubSub client to connect to.

Client resource

The client resource creates Kafka clients for access of PubSub.

Template of a client resource:

```
targets:
  pubsub:
    resources:
      - name: xxx-ota-client
        type: client
        owner: ota-qa
        acls:
          - topic: ota.exporter-topic
            operation: READ
```



The **clientId** of the client will always be created by the PubSub provisioning. The pattern is **ps-{region}-{clientName}**. Please, check the environments for the region name.

Consumer group for your consuming client

The `consumer.group` of your consumer will always be the generated using `clientId`.

The `clientId` is constructed in the provisioning step and the pattern used is: `ps-{region}-{clientIdName}`.

For the environment `EU` and the client `ota-pubsub-client`, the name would be: `ps-prod-eu-ota-pubsub-client`.

For the environment `NA` and the client `ota-pubsub-client`, the name would be `ps-prod-us-ota-pubsub-client`.

Topics for async API

- Customer load test
- Device-Vehicle mapping
- OTA Data Exporter
- OTA Function Calls
- OTA Updates Campaign Exporter
- OTA Updates Distribution Exporter
- OTA Updates Inventory Exporter

Example - getting started with Kafka

The following examples will use Kafka consumer and producer command line tools/binaries.

Prerequisites

Before you can connect a Kafka client to PubSub, make sure you have the following prerequisites in place:

- Onboarding to PubSub guided by the C-Hub support team.
You will get:
 - A PubSub client with Client ID, API-Key & API-Secret
 - A bootstrap-URL to the Kafka cluster
 - A topic to connect to
- The Kafka binaries installed on your local machine.
You can download them from the Apache Kafka website: <https://kafka.apache.org/downloads>.

Connecting a Kafka client to PubSub

To connect a Kafka client to PubSub, you need to perform the following steps:

Step 1: Configure authentication

To authenticate with PubSub, you need to configure your Kafka client with the necessary security credentials.

1. Open a terminal and navigate to the directory where you have installed the Kafka binaries.
2. Create a new file named `client.properties` and add the following configuration settings:

```
bootstrap.servers=<your-bootstrap-servers>
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="<your-api-key>" password="<your-api-secret>";
consumer.group-id: <your-clientId>
```

3. Replace `<your-bootstrap-servers>` with the URL for the EU or NA environment.
Also, replace `<your-clientId>`, `<your-api-key>` and `<your-api-secret>` with your provided information.

Step 2: Create and consume from a test topic

Now that your Kafka client is configured, let's create and consume messages from a test topic.

Use your local terminal:

1. Navigate to the Kafka binaries directory.
2. Start a consumer that listens to the `<your-topic>` by running the following command:

```
./kafka-console-consumer.sh --topic <your-topic> --bootstrap-server <your-bootstrap-servers> --
consumer.config client.properties --from-beginning
```

You should now see the consumer waiting for messages from `<your-topic>`.

3. In a separate terminal, produce a test message to the `<your-topic>` by running the following command:

```
./kafka-console-producer.sh --topic <your-topic> --bootstrap-server <your-bootstrap-servers> --
producer.config client.properties
```

Enter a test message and press the enter key to send it to `<your-topic>`.

4. Switch back to the consumer terminal, and you should see the test message being consumed and displayed.

Conclusion

Congratulations! You have successfully connected a Kafka client to PubSub and performed basic operations like producing or consuming messages from a test topic.

Further information and tips

- <https://kafka.apache.org/documentation/>
- <https://www.confluent.de/blog/5-things-every-kafka-developer-should-know/#tip-5-power-of-record-headers>
- <https://dzone.com/articles/20-best-practices-for-working-with-apache-kafka-at>
- <https://blog.softwaremill.com/7-mistakes-when-using-apache-kafka-44358cd9cd6>
- <https://technicallyeasy.net/top-6-mistakes-to-avoid-while-building-apache-kafka-driven-solutions/>

5.2.2 Data consumption by customer application via REST API

Authentication

To use the REST API of the C-Hub services, you need to specify a valid [bearer token](#). C-Hub accepts tokens from the following OAuth2 identity services:

Region	OAuth2 Token Endpoint
EU	https://p2.authz.bosch.com/auth/realms/ {yourTenantId} / protocol/openid-connect/token
NA	https://na2.authz.bosch.com/auth/realms/ {yourTenantId} / protocol/openid-connect/token



Your data consuming application can implement for example the client credentials flow. See <https://auth0.com/docs/get-started/authentication-and-authorization-flow/client-credentials-flow>

Scopes

Application scope

By requesting the token for a project-specific activation, the [least-privilege principle](#) is applied. This prevents tokens from growing too big (token bloating).

To request the token for the correct project context, add the combination of **project_id** and **application_id** in the form of **project_id:application_id** to the list of **scopes**.

Example:

Given that your current **project_id** is **abcdef12-xxx** and the **application_id** is **12345678-xxx**, the correct scope value would be:

abcdef12-xxx:12345678-xxx

- To determine the current project context and its **project_id**, you can leverage the placeholder feature as part of your **External URL**.
- Your specific **application_id** can be derived from your PKCE Client ID because they are both the same.

Standard scopes

OpenID Connect defines a couple of standard scopes out of which the following ones are supported:

- **openid** - required, indicates usage of OpenID Connect

C-Hub defined scopes

In the token request you will need to specify the **scope**. The scope will grant the permissions necessary for a specific service.

The following scopes are supported for applications which consume from the C-Hub backend:

Scope	C-Hub service
device-management	Device Connectivity and Device Management
fleet-management	Device Connectivity and Device Management
vms-campaign-service	Device Maintenance
fota-distribution	Device Maintenance
vms-inventory-service	Device Maintenance
remote-measurement	Remote Measurement
remote-diagnostics	Remote Diagnostics
ota-function-calls	Remote Diagnostics
ctxpl-manager	Context Pipeline API

Service endpoints

See [RESTful API docs](#)

Examples

See [Example - data consumption by customer application via REST API](#)

OAuth2.0 Based Authorization

Context

The current implementation supports [Keycloak](#) as the identity provider. Keycloak is based on the open authentication and authorization standards OAuth2.0 and OpenID Connect (OIDC).

The suitable authorization flow depends on the type of client you are using to authorize your user. A user could be either technical or personalized.

- Technical users are also called Service Accounts and usually are used to authorize a request from a backend service to another backend service.
- Personalized users usually interact with a backend via web applications.

For further information please consult the official OAuth flow documentation (see <https://datatracker.ietf.org/doc/html/rfc6749>).

Authorize service account via confidential client

To authorize a technical user via a **backend service** you can use the **client credentials grant**.

Example request:

```
curl -X POST https://p2.authz.bosch.com/auth/realms/<realm_name>/protocol/openid-connect/token
-H 'Content-Type: application/x-www-form-urlencoded'
--data-urlencode "grant_type=client_credentials"
--data-urlencode "client_id=<client_id>"
--data-urlencode "client_secret=<client_secret>"
--data-urlencode "scope=<scope>"
```

Placeholder description:

- The **realm_name** is your tenant ID.
- The **grant_type** defines which type of authorization flow is used. For the client credentials grant the correct value is **client_credentials**.
- The **client_id** and **client_secret** are required to authorize a service account.
- Scopes can be used to limit the actual roles in the access token for a specific use case. Leave this parameter blank (space character) to receive the default client scopes which are assigned to your service account.



If you do not have any credentials yet, please send a request to the C-Hub support team. A service account with proper roles will be created and the credentials will be forwarded by our support team.

Example response:

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "access_token": "eyJhb...yLWxA",
  "expires_in": 600,
  "refresh_expires_in": 36000,
  "refresh_token": "eyJhb...04f8k",
  "token_type": "bearer",
  "not-before-policy": 0,
  "session_state": "8876d872-xxx-358e0b5e0dd1",
  "scope": "email profile"
}
```

Use Access Token

To access any API endpoint provided by C-Hub backend it is required to add the Access Token as **Authorization Header "Bearer <Base64Encoded-JWT>"** to the request.

RESTful API docs



Vehicle Connectivity Services | API

- EU: <https://ota.eu.bosch-mobility-cloud.com/api-hub/>
- NA: <https://ota.na.bosch-mobility-cloud.com/api-hub/>

This is an overview of APIs offered, to simplify your daily work in automotive development.

To use the listed APIs, a dedicated onboarding is required.

- [Device Management API \(V2\)](#)
- [VMS Inventory Service \(1.0.0\)](#)
- [FOTA Distribution \(1.0.0\)](#)
- [VMS Campaign Service \(1.0.0\)](#)
- [Context Pipeline \(1.0.0\)](#)

Device Management API (V2)

Description: Device Management is responsible for maintaining device records in C-Hub backend.

API docs EU: <https://ota.eu.bosch-mobility-cloud.com/api-hub/render/openapi/device-management>

API docs NA: <https://ota.na.bosch-mobility-cloud.com/api-hub/render/openapi/device-management>

VMS Inventory Service (1.0.0)

Description: Inventory Service API description

API docs EU: <https://ota.eu.bosch-mobility-cloud.com/api-hub/render/openapi/inventory>

API docs NA: <https://ota.na.bosch-mobility-cloud.com/api-hub/render/openapi/inventory>

FOTA Distribution (1.0.0)

Description: FOTA (firmware updates over the air) Distribution API description

API docs EU: <https://ota.eu.bosch-mobility-cloud.com/api-hub/render/openapi/distribution>

API docs NA: <https://ota.na.bosch-mobility-cloud.com/api-hub/render/openapi/distribution>

Vehicle manufacturers (OEM) can optionally book firmware updates over the air to be performed by Bosch.

VMS Campaign Service (1.0.0)

Description: Management for campaigns to update C-Hub TCU2 devices over the air.

API docs EU: <https://ota.eu.bosch-mobility-cloud.com/api-hub/render/openapi/campaign>

API docs NA: <https://ota.na.bosch-mobility-cloud.com/api-hub/render/openapi/campaign>

Context Pipeline (1.0.0)

Description: Context Pipeline API description

API docs EU: <https://ota.eu.bosch-mobility-cloud.com/api-hub/render/openapi/context-pipeline>

API docs NA: <https://ota.na.bosch-mobility-cloud.com/api-hub/render/openapi/context-pipeline>

Preliminary

Example - data consumption by customer application via REST API

This page shows an example on how to use the REST API.

At this point we assume that your company is allowed to manage the tenant and may access the Platform user interface, or the C-Hub support team has informed you about your client ID and secret.

Request a Bearer Token

The **curl** command for getting a token based would follow this pattern.

```
curl --request POST \
  --url 'https://p2.authz.bosch.com/auth/realms/{yourTenantId}/openid-connect/token' \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data grant_type=client_credentials \
  --data 'client_id={yourClientId}' \
  --data 'client_secret={yourClientSecret}' \
  --data 'scope={yourScope}'
```

The resulting token can be used as an authorization header for the next request.

GET a list of devices

The **curl** command for requesting an endpoint of device management would follow this pattern.

```
curl --request GET \
  --url 'https://api.devices.eu.bosch-mobility-cloud.com/v2/devices?page=1&size=20' \
  --header 'accept: application/json' \
  --header 'Authorization: Bearer {yourBearerTokenWhichWasTheResponseAbove}'
```

Response example

With the assumption that you have 100 devices but requested only for page 1 with a size of 20, the result will hold the first 20 devices which are "visible" for the authorized subject.

In our example, we list only one, to show how the JSON object for a device will look like.

```
{
  "_embedded": {
    "devices": [
      {
        "deviceId": "CCU_1234567890",
        "model": "NATIVE",
        "serialNumber": "123abc",
        "iccId": "8991101200003204000",
        "customAttributes": {
          "additionalProp1": "string",
          "additionalProp2": "string",
          "additionalProp3": "string"
        },
        "capabilities": {
          "additionalProp1": 0,
          "additionalProp2": 0,
          "additionalProp3": 0
        },
        "connectedTo": "mc-mqtt",
        "onlineStatus": {
          "state": "ONLINE",
          "since": "2024-08-26T14:54:47.315Z"
        },
        "createdAt": "2024-08-25T14:54:47.315Z",
        "updatedAt": "2024-08-25T14:54:47.315Z"
      },
      {"more": "the other 19 devices in the same pattern"}
    ]
  },
  "_links": {
    "first": {
      "href": "https://api.devices.eu.bosch-mobility-cloud.com/v2/devices?page=1&size=20"
    },
    "prev": {
      "href": "the link to the previous page is only listed in case you would have requested page 2"
    },
    "self": {

```

```
    "href" : "https://api.devices.eu.bosch-mobility-cloud.com/v2/devices?page=1&size=20"
  },
  "next" : {
    "href" : "https://api.devices.eu.bosch-mobility-cloud.com/v2/devices?page=2&size=20"
  },
  "last" : {
    "href" : "https://api.devices.eu.bosch-mobility-cloud.com/v2/devices?page=5&size=20"
  }
},
"page" : {
  "size" : 20,
  "totalElements" : 100,
  "totalPages" : 5,
  "number" : 1
}
}
```

5.2.3 Data consumption by customer application via Data Integration service

The Data Integration service supports the following interfaces as a data sink:

- [AMQP](#)
- [AWS S3](#)
- [Azure Blob Store](#)
- [HTTP based API or backend](#)
- [Bosch IoT Insights data recorder API](#)
- [Apache Kafka](#)



In order to establish the connection, the C-Hub support team will require the details outlined in the "Required configuration" section.

AMQP

Supported infrastructure	Active MQ IBM MQ Rabbit MQ Azure Event Hub Solace
Required configuration	▪ Broker URL ▪ e.g. <i>amqp://my-amqp-broker.amqp:5672</i> ▪ Topic to send data to ▪ Authentication mechanism ▪ Password ▪ Certificate ▪ Username ▪ Password ▪ Certificate (in case of certificate-based authentication) ▪ Compression on / off ▪ Can compress data before sending it
Capabilities	▪ Supports protocol AMQP 1.0
Default meta data	Meta data will be attached as AMQP header. ▪ uid - random ID attached to each outgoing message ▪ messageId - the internal unique ingest ID identifying data throughout the Data Integration system

Preliminary

AWS S3

Supported infrastructure	Amazon S3 Minio
Required configuration	▪ Endpoint ▪ e.g. <code>s3.eu-central-1.amazonaws.com</code> ▪ Region ▪ e.g. <code>eu-central-1</code> ▪ URL path style ▪ defines if path style for the bucket URL should be used ▪ true: <code>aws.s3.com/bucket</code> ▪ false: <code>bucket.aws.s3.com</code> ▪ Authentication ▪ AWS-IAM ▪ Access Key ▪ Access Secret ▪ Processor ▪ Compression ▪ <code>gzip/zip/none</code> ▪ Forwarder configuration ▪ Match configuration ▪ which cloud event attribute should be matched ▪ Forwarding configuration ▪ bucket ▪ endpoint ▪ in case a different endpoint should be used than the root one ▪ filename pattern ▪ e.g. <code>getExtension('ingestuid') + '.bin'</code> resulting in <code>9823934-3424-2342.bin</code> ▪ duplicate prevention
Capabilities	▪ Can upload data to one or different S3 bucket combinations ▪ Data blob name can be dynamically built from header fields ▪ Data can be compressed before uploading it to the S3 bucket
Default meta data	▪ <code>dianadatastreamid</code> (indicates how the data was entering the system e.g. http-ingest) ▪ <code>dianadatasize</code> (data size in bytes) ▪ <code>dianaproducertenant</code> (tenant that owns/produced the data) ▪ <code>dianaconsumertenant</code> (tenant that consumes the data) ▪ <code>dianaingesttimestamp</code> (timestamp of the data ingestion into the Data Integration system) ▪ <code>dianaingestuid</code> (unique ID representing the data itself throughout the system) ▪ <code>producerclient</code> (in case the data was ingested via HTTP, the OAuth client name) ▪ Content-Type (type of the content derived from the ingested data) ▪ Content-MD5, <code>dianamd5hash</code> (MD5 hash of the data)

Azure Blob Store

Supported infrastructure	Azure Blob Store Azurite Dev Container
Required configuration	- Container name - e.g. data - Connection URL - e.g. <i>storageaccountfoo.microsoft.com</i> - Blob naming pattern <code>getExtension('ingestuuid') + '.bin'</code> resulting in e.g. 9823934-3424-2342.bin - Duplicate prevention on/off - Detects duplicated blob with the help of an Azure feature - Authentication type - Azure AD / Service Principal - Shared Key - In case of Shared Key: - Account name - Account key - In case of Azure AD / Service Principal - Client ID - Client Secret - Tenant ID - Compression on/off - Can compress data before uploading it to the blob store
Capabilities	- Can upload data to one or different storage account and container combinations - Data blob name can be dynamically built from header fields - Data can be compressed before uploading it to the blob store
Default meta data	dianadatastreamid (indicates how the data was entering the system e.g. http-ingest) dianadatasize (data size in bytes) dianaproducertenant (tenant that owns/produced the data) dianaconsumertenant (tenant that consumes the data) dianaingesttimestamp (timestamp of the data ingestion into the Data Integration system) dianaingestuuid (unique ID representing the data itself throughout the system) producerclient (in case the data was ingested via HTTP, the OAuth client name) - Content-Type (type of the content derived from the ingested data) - Content-MD5, dianamd5hash (MD5 hash of the data)

Preliminary

HTTP based API or backend

Supported infrastructure	Any HTTP based API or backend
Required configuration	- Target backend URL - e.g. <i>http://foo.bar/data</i> - Authentication type - Basic Authentication/OAuth - Username - Password - In case of OAuth - IDP (identity provider) token URL - Scope - Grant-Type: Only client_credentials supported
Capabilities	- Can upload data to exactly one endpoint - Performs POST requests - The content type is taken over from the originally ingested data - The response is only evaluated to detect errors (4XX, 5XX status codes), response itself is discarded
Default meta data	- No meta data is forwarded, because it is a generic component and not every API would accept additional headers - Content-Type (type of the content derived from the ingested data)

Bosch IoT Insights data recorder API

Supported infrastructure	Bosch IoT Insights instance
Required configuration	- IoT Insights data recorder project URL - e.g. <i>https://bosch-iot-insights.com/data-recorder-service/v2/abc1234</i> - Authentication - IoT Insights username - IoT Insights password
Capabilities	- Can upload data to exactly one IoT Insights project - Supports basic authentication as provided by IoT Insights
Default meta data	dianadatastreamid (indicates how the data was entering the system e.g. http-ingest) dianadatasize (data size in bytes) dianamd5hash (MD5 hash of the data) dianaproducertenant (tenant that owns/produced the data) dianaconsumertenant (tenant that consumes the data) dianaingesttimestamp (timestamp of the data ingestion into the Data Integration system) dianaingestuuid (unique ID representing the data itself throughout the system) producerclient (in case the data was ingested via HTTP, the OAuth client name) - Content-Type (type of the content derived from the ingested data) E.g. <i>application/json</i>, or in case a compression is configured, it will be <i>application/gzip</i> or <i>application/zip</i> - Content-MD5 (MD5 hash of the data)

Apache Kafka

Supported infrastructure	Azure Event Hub Kafka Mode AWS MKS Kafka Mode Native Kafka Broker (e.g. Confluent)
Required configuration	▪ Bootstrap Server URL ▪ e.g. <i>my-kafka-cluster.kafka:9092</i> ▪ Topic to retrieve data from ▪ Authentication mechanism ▪ SASL ▪ PLAINTEXT ▪ Profiles: ▪ sasl ▪ aws ▪ mtls ▪ eventhub ▪ Username ▪ Password ▪ Certificate (in case of certificate based authentication) ▪ Event Hub Mode Authentication: ▪ Azure Tenant ID ▪ Azure Client ID ▪ Azure Client Secret
Capabilities	▪ Supports Azure Event Hub Kafka Mode ▪ Supports AWS MKS Kafka Mode ▪ Supports usage of mTLS certificates ▪ Supports creation of Kafka headers from event meta data
Default meta data	Meta data will be attached as Kafka header: ▪ dianadatastreamid (indicates how the data was entering the system e.g. http-ingest) ▪ dianadatasize (data size in bytes) ▪ dianamd5hash (MD5 hash of the data) ▪ dianaproducertenant (tenant that owns/produced the data) ▪ dianaconsumertenant (tenant that consumes the data) ▪ dianaingesttimestamp (timestamp of the data ingestion into the Data Integration system) ▪ dianaingestuuid (unique ID representing the data itself throughout the system) ▪ producerclient (in case the data was ingested via HTTP the OAuth client name) ▪ Content-Type (type of the content derived from the ingested data)

Preliminary

6 Data protection and cybersecurity aspects

6.1 General information about cybersecurity

The product was developed within an ISO/SAE 21434 certified Cyber Security Management System and corresponding cybersecurity activities were done.

6.2 Cybersecurity claims

The product contains functions which could process personal data. If so, the product retains the risk of disclosure of person-identifiable information. As the data controller, the customer designs the processing in detail. The customer is expected to evaluate if the protective measures of the product are sufficient for the dedicated use case.

Irrespective of the high quality of the delivered component, the product retains the risk that currently unknown cybersecurity vulnerabilities may get discovered in the future. As risk mitigation, Bosch has installed a 'cybersecurity monitoring' and further continual cybersecurity activities according to ISO/SAE 21434. In case a cybersecurity vulnerability in a Bosch product is discovered or disclosed by or to the customer, it is highly recommended to inform Bosch upfront, before its publication, via PSIRT@bosch.com.

The product retains the risk that intellectual property (IP) from Bosch, the customer, and in some cases third-parties can be disclosed during the software distribution. For the purpose of IP protection and avoidance of re-engineering, it is expected that delivered software is handled as confidential during the distribution process to the customer, into plants, workshops and Firmware Over-the-Air (FOTA) updates. It is expected that the customer has implemented appropriate measures within his Customer's Software Logistic Infrastructure (IT network) and associated processes to secure and restrict the access to delivered software.

The product retains the risk of disclosure of confidential data (including but not limited to personal data) during the decommissioning process of the customer or third-parties. It is expected that the customer evaluates and adequately mitigates the risk on vehicle level or within the logistic infrastructure.

The product retains the risk of data manipulation in the memory of the software executing control unit. It is expected that the interfaces, which are designed or implemented by the customer or third-parties, do not offer any mechanism to alter the memory content, besides the regular software update (reflash) and intended changes by documented diagnostic services (e.g., UDS).

It is further expected that interfaces, which are designed or implemented by the customer or third-parties, and which are capable to change the memory content, are well secured and limited to engineering purposes (e.g., XCP).

The product retains the risk of disclosure of confidential data by third-party software, if implemented. It is expected that the interfaces designed or implemented by the customer or third-parties do not read confidential data, cryptographic material or software with intellectual property, unless Bosch expressly permits this.

The product retains the risk that software, which is designed or implemented by the customer or third-parties, could compromise the intended function of the product. It is expected that the customer validates the operational behavior.

7 Safety and security notes

Definitions:

-  **Danger**
indicates a hazardous situation which - if not avoided - **will** result in **death or serious injury**.
-  **Warning**
indicates a hazardous situation which - if not avoided - **could** result in **death or serious injury**.
-  **Caution**
indicates a hazardous situation which - if not avoided - could result in **minor or moderate injury**.
- **Notice** is used to address practices not related to physical injury.

Additionally, we use blue boxes to highlight on info level, and green boxes for tips.

- Example info

 A blue box without a title is used to highlight the hint for the reader.

- Example tip

 A green box without a title is used to highlight a tip for the reader.

The actual safety and security notes are detailed in the following sections:

- [Safety and security notes - drivers](#)
- [Safety and security notes - transport and assembly partners](#)
- [Safety and security notes - developers and operators of the cloud services](#)
- [Additional notes for developers](#)

7.1 Safety and security notes - drivers

7.1.1 Notice

The driver of the connected vehicle must be informed about the usage of the Automotive Connectivity Hub, and explicitly about the C-Hub TCU2 device.
This can be achieved by delivering the C-Hub data sheet into each vehicle equipped with a C-Hub TCU2 device.
As the C-Hub TCU2 device does not have its own power supply, the driver should be aware that the C-Hub TCU2 device is connected to the vehicle's battery.

If a customer or driver stops/interrupts the power supply to the C-Hub TCU2 device during its self-update, the self-update will fail.
Failures due to such stopping/interruption of power supply are not considered in any service level or report calculation.

Preliminary

7.2 Safety and security notes - transport and assembly partners

Before you start with the installation, please consider the following main points related to personal and vehicle safety.

Transport and storage	The device must be stored according to the specified storage conditions. Do not throw or drop the unit. Dropping the unit may result in a malfunction that may only become evident at a later point in time. Such units will have to be scrapped or sent to Bosch for inspection. Storing temperature range -40° C to 85° C.
Assembly	The product's operating safety is only ensured if the permissible conditions are maintained. Voltage supply and grounding: Observe the correct sequence when connecting to /disconnecting from the vehicle's electrical system.
Start and end of operation	If the device is connected incorrectly, it may draw the full operating current even when switched off. In this case, without any further deactivation on the vehicle side, there is a risk of battery depletion. No liability is assumed for any damage occurring because of this. Make sure to provide the required information to any service partner who needs to receive specific information for start and end of operation of the device.

Preliminary

7.2.1 Warning

Warning

The product will be provided and approved for a rated voltage of 12 or 24 Volts. Short circuits must be securely prevented, even in the event of a fault. Creeping short-circuits lead to high arc energies that cannot be switched off by conventional melting fuses. If other rated voltages are present in the vehicle (e.g. 24/48 V for example in hybrid vehicles, or 400/800 V in electric vehicles) we recommend that safety precautions are taken in the vehicle itself (e.g. additional insulation of high voltage lines and the use of edge protectors when passing through sharp-edged areas, in order to prevent short circuits to inputs/outputs of our device, e. g. to loudspeaker lines).

7.2.2 Caution

Caution

The product is intended, and has been approved, for installation and operation in vehicles with a rated voltage of 12 or 24 Volts. It may be necessary to adapt the product's factory-set state to suit the specific country.

 Caution

Only connecting cables and external devices that are appropriate for the device in question may be used (e.g. proper current carrying capability, proper EMC shielding, flammability verified with appropriate certificates). Compliance with the applicable standards can no longer be guaranteed if the device – including the software – is modified without the agreement of Bosch.

 Caution

Bosch assumes no responsibility for damage as a result of incorrect indicators/displays. These may arise if the device has not been connected or has been incorrectly connected, or if the device receives false or erroneous signals from the system.

7.2.3 Notice

Formation of condensation on the device must be avoided. In case of condensation formation, you must allow an acclimatization time of up to two hours to pass when mounting the device into the vehicle.

Do not cover ventilation openings and heat sinks. Otherwise, a build-up of heat that could lead to malfunction may occur in the device.

The housing/surface of the C-Hub TCU2 device may be very hot when the device is operational. Please exercise caution and use appropriate protective equipment (PPE).

Do not insert foreign bodies into the insertion slots or openings of the device. Injury, or damage to the device, may occur otherwise.

The device must not come into contact with hot or burning objects (e.g. cigarettes).

Emissions coming from portable transceivers (e.g. mobile phones, walkie-talkie) in the immediate vicinity can exceed the limits given in the customer specs with respect to resistance to interference. At distances of less than 200 mm between the vehicle component and all the possible positions of portable transceivers in the passenger compartment or the vehicle trunk, it is not possible to completely rule out any interference that may limit the correct functioning of the vehicle component. If the customer intends to fall below the minimum safety distance of 200 mm, it is the customer's responsibility to check whether additional measures are necessary for this application and implement these measures.

Bosch is only responsible for the compliance of the product side plug (interface) with the agreed upon customer specification. Since the plug system is used per customer request, Bosch is not responsible and does not warrant for the connection assembly, especially not for its electrical function, durability, and sealing.

Preliminary

Device cleaning related

To clean the device, never use hard or sharp objects that could damage the protective pane or housing. Do not use aggressive cleaning agents such as thinners, gasoline, abrasive cleaners, spray cleaners, acidic or alkaline solutions, or wax. Do not spray any liquids onto the device. To clean the housing and protective pane / display, moisten a soft cloth with tepid water and wipe off the dirt. Make sure that no liquid enters the inside of the display. Afterwards, wipe the cleaned surface with a clean, dry cloth.

If the device is to be cleaned before installation in the vehicle, you must make sure the openings (in particular the connector contact points) are kept sealed so that no liquid can enter the device.

Any media compatibility test does not represent an extension to the permitted cleaning agents. Passing a test according to ISO 16750-5 does not imply that the chemicals and the performed application method (e.g., spray acetone, pour denatured alcohol) are permitted as a cleaning approach.

Antenna-related

The WLAN antennas must be installed inside the vehicle cabin, because the WLAN 5.1 GHz band (channels 36-48) are not allowed outside the cabin in some countries. In case the WLAN antennas shall be placed outside the vehicle, a change in the software is needed to disable the WLAN 5.1 GHz band (channels 36-48). A recertification will be necessary.

The customer must ensure that the Bluetooth/WLAN functionality is always configured according to the target market's specific regulations, e.g. with regards to receiving frequency bands. The customer must ensure correct configuration of frequency bands in case of software updates.

The customer must ensure that GNSS functionalities are configured according to the regulations of the different target markets.
Bosch sets the devices end-of-line to "All enabled" for the EU variant.
E.g. USA: GPS and Galileo (E1, E5) enabled, others e.g. GLONASS disabled.

The antenna must be installed at least 20 cm away from the nearest *interior Class A surface* in the vehicle. In particular, these are surfaces which may affect the antenna signal quality due to curvature and reflection.

The antenna must be installed at least 20 cm away from the driver or passengers.

If you have chosen to install the antenna into a DIN slot, make sure that there is a layer of physical separation between the antenna and the driver and/or passengers. Be aware that this installation location is feasible when the cabin roof is made of plastic and not metal. In any case, the line of sight (LOS) to satellites will be reduced because of the cabin roof.

For the cases when the antenna is installed directly on the interior glass surface of the windshield, the antenna must not block the vision of the driver. If the windshield window is tinted or has a thick coating, it can interfere with or in extreme cases even block GNSS.

Avoid routing the antenna and adapter cables through the areas with highest radiated emissions to reduce the possibility of any signal interference. For example, do not wrap these around another ECU.

Do not group antennas closely together, as they will interfere with each other. For optimal performance and to avoid interference, install the antenna as far away as possible from any existing antennas. Please, refer to the technical specifications of each mounted antenna for detailed installation guidelines and separation requirements.

The isolation between each frequency band (cellular, BT/WLAN and GNSS) of the antenna system needs to be as high as possible to avoid interference and intermodulation during co-existence testing. This is important to pass legal EMC requirements. This isolation also ensures proper operating performance and co-existent operation of the different systems cellular, BT/WLAN and GNSS.

To ensure optimal performance, an isolation of at least 60dB is recommended. Isolation values below 60dB can result in performance degradation of the affected radio service. The isolation can be achieved by different measures, for example proper distance between the antennas for different services, implementation of suitable filters or combinations of any suitable measures.

The customer is responsible to use only such cables and/or antennas that ensure that limits with regard to minimum health and safety requirements concerning the exposure of workers to the risks from electromagnetic fields are fulfilled.

The antenna must not be installed on glass surfaces.

Cable-related

Insufficient cable harness fixation can cause premature failures.

The main wiring harness delivered as an accessory of the C-Hub TCU2 device is designed with the assumption that the CAN topology in the vehicle is not terminating. In case the CAN topology in the vehicle already provides a CAN termination, you **must** cut and insulate both sides of the additional loop (the longer orange wire) on the main wiring harness, near the FMS connector. This is only the case if you **measure a resistance of 60 Ohm at the vehicle-side FMS connector**. This results in a CAN resistance which acts as a CAN termination. For details, see the pinning table at FMS connector.

The C-Hub main wiring harness is compliant with the IEC 62368-1 PS2 circuitry requirements, according to which the power available in normal and failure modes must be limited to < 100 W. The customer is responsible to ensure that an appropriate centralized fuse, depending on actual setup, is installed in the vehicle common positive power line to the C-Hub main wiring harness, and subsequently to the C-Hub TCU2 device. Therefore, an automotive grade fuse with an operating range of up to 125 °C is required as follows:

- a 4 A fuse for vehicles with 12 V supply
- a 2 A fuse for vehicles with 24 V supply

If the FMS connector is used, KL 30 from the System connector **must not be used**.

Preliminary

DIN-Adapter related

The DIN Adapter has a LED cable, on which there are two pieces of adhesive tape.

1. Preserve the piece of adhesive tape which is closer to the front panel of the DIN Adapter.
2. Remove the piece of adhesive tape which is closer to the edge of the DIN Adapter.

As the edges of the DIN adapter are rather sharp, they could damage the LED cable at the position where this cable goes out of the DIN Adapter. The safest path for the LED cable out of the DIN Adapter is under the main connector (white). It is also recommended that the LED cable should be securely fixed to the other wires of the main wiring harness, for example by using a cable tie.

7.3 Safety and security notes - developers and operators of the cloud services

It is the customer's responsibility to ensure safe usage of the system, especially in the following aspects:

7.3.1 Danger

Danger | Unexpected Vehicle Behavior

OTA Update

The assignment of improper, incomplete, or incompatible software packages can lead to safety-critical situations.

Each change to the software setup can influence the behavior of the ECUs and the systems they control. This can lead to unexpected behavior of the vehicle, e.g. braking, acceleration or swerving.

- Only make changes to the software setup if you are sufficiently qualified to do so.
- Before the OTA update process, the software update package must have been tested, verified and released for the corresponding application case.
- Ensure that the update package is assigned to the correct vehicle or fleet.

⚠ Danger | Unexpected Vehicle Behavior**Electric-Electronic Architecture Safety Features**

The effect of the vehicle electronics via the software can lead to safety-critical situations. To avoid unpredictable vehicle behavior, the electric-electronic architecture used in the vehicle must include the following safety features:

- The connected ECUs must be robust against malfunctions.
- Safety-critical malfunctions due to incorrect bus communication must be prevented on the vehicle level.
- The installation of incorrect or incompatible updates must be prevented.
- Diagnostic Trouble Codes (DTC) in the fault memory are not deleted until after correction of the fault.

Until the fault is corrected, continuous deletion of safety-critical entries in the fault memory must be prevented.

Communication with the fault memory (DTC) via the On-Board Diagnostics (OBD) interface must not cause any safety-critical malfunctions.

⚠ Danger | Unexpected Vehicle Behavior

Any change to the hardware setup, software settings, including flash procedures, can influence the behavior of the ECU and the systems controlled by the ECU. This can lead to unexpected behavior of the vehicle, e.g. braking, acceleration, or swerving. Therefore:

- Only use the product if you are sufficiently qualified to do so.
- Ensure that the parameters of the bus communication (e.g. baud rate) are correctly set.
- Instruct the driver prior to starting the run about the following points vehicle malfunctions can occur at any time.

⚠ Danger | Unexpected Vehicle Behavior**Electric-Electronic Architecture Changes**

Each change to the electric-electronic architecture can lead to safety-critical situations.

- The electric-electronic architecture of the vehicle must match the data of the C-Hub cloud backend vehicle database.
- If C-Hub TCU2 devices are replaced in the vehicle, the C-Hub cloud backend vehicle database must be updated prior to further use of the software.
- The vehicle-specific settings of the C-Hub TCU2 device (e.g. CAN communication) must match the existing electric-electronic architecture of the vehicle.

⚠ Danger | Unexpected Vehicle Behavior**Remote Diagnostics**

The assignment of Remote Diagnostics jobs can lead to safety-critical situations.

Each change to the software setup can influence the behavior of the ECUs and the systems they control. This can lead to unexpected behavior of the vehicle, e.g. braking, acceleration or swerving.

- Only make changes to the software setup if you are sufficiently qualified to do so.
- Ensure that the Remote Diagnostics job is correctly programmed: The Remote Diagnostics job must only be executable if the vehicle is in a safe state.

⚠ Danger | Unexpected Vehicle Behavior**OTA, Measurement, or Remote Diagnostics - Job Assignment**

Incorrectly assigned jobs can lead to safety-critical situations.

Ensure that each job (OTA, Measurement, or Remote Diagnostics) is assigned to the correct vehicle or vehicle pool.

7.3.2 Warning

Warning

This product allows you to influence safety-relevant systems or data (e.g. in motor vehicles, vehicle components and test benches). In the event of a malfunction or a hazardous situation, it must be possible to put the system into a safe state (e.g., emergency stop or emergency operation).

Warning

The product must only be operated if you have the necessary qualifications and experience for this product.
Incorrect operation or operation by users without sufficient qualifications may lead to injuries, death, or property damage.

7.3.3 Notice

The driver of the connected vehicle must be informed about the usage of the product.

This product was developed and released for applications in the automotive area or the development and application in development patterns.
It makes networking of vehicles and cloud-based services possible. The product does not comply with ISO 26262 and must therefore **not be used for safety-related applications** in series vehicles.

This product is not developed according to ISO 26262 and has no safety integrity. It is classified as Quality Management (QM), and does not meet ASIL safety requirements in accordance with ISO 26262.
The target vehicles and electronic control unit (ECUs) are considered as inherently safe. Validate this assumption before the use of this product.
The vehicle manufacturer is responsible for the overall safety analysis and safety concept and must consider the legal safety requirements.

Preliminary

7.4 Additional notes for developers

- [Responsibilities and obligations](#)
- [Recommendations for developers](#)
- [Notes about containerized vehicle apps](#)

7.4.1 Responsibilities and obligations

 This page provides a non-exhaustive overview of the responsibilities and obligations related to the C-Hub TCU2 devices and their software. There may be some differences related to the different customer types, i.e. TSP and OEM.

Responsibilities and obligations of the customer

The customer needs to:

- Ensure that the C-Hub TCU2 devices receive uninterrupted power supply during the pairing process
- Ensure that the C-Hub TCU2 devices receive uninterrupted power supply during self-updates
- Ensure that availability of the mobile network in the vehicles is not disturbed, as this is required for communication with the C-Hub backend
- Store and process the data in its system, before we delete it due to expiry of the retention period
- For custom vehicle apps:
 - Ensure that the customer's vehicle apps, which will be deployed to the C-Hub TCU2 devices, are checked and tested in the customer's vehicle setup by the supplier/ developer of the respective vehicle app and are delivered as an OCI container image via an OCI container registry
 - Ensure that these container images are built in a dedicated development environment, as this is neither intended nor supported to be performed on the C-Hub TCU2 device itself

Joint responsibilities and obligations between Bosch and the customer

The C-Hub team and the customer will cooperate in order to:

- Ensure that the C-Hub TCU2 devices can connect to the MNO provider
- Ensure that the C-Hub TCU2 devices are powered on and have a stable internet connection during the pairing process
- Ensure the correct functional behavior of any third-party vehicle apps deployed or to be deployed on the C-Hub TCU2 devices
- Maintain signal collections and messages, upload DBC files
- *With regard to native apps of OEM customers:* The customer needs to provide third-party software for updates of possible native apps, so they can be integrated into the new software version by Bosch

Responsibilities and obligations of Bosch

In general, Bosch, as provider of the C-Hub service, will take care of the following aspects. However, there may be differences for the different customer types, i.e. TSP and OEM.

- Ensure a valid contract for mobile data for the connected C-Hub TCU2 devices
- Rollout new device software, monitor and report progress
- Test and validate new software to approve the rollout of new device software
- Pair the vehicle with the C-Hub backend
- Basic Fleet management: Create and manage individual vehicles, vehicle setup groups, and campaigns
- Create and manage updates and version control, status monitoring, activation/deactivation and C-Hub TCU2 software version compatibility
- Assign measurement jobs to selected vehicles or vehicle groups
- Perform validations of measurement results acquired from measurement configurations
- Store data up to the agreed number of retention days; make the data available within the agreed retention window; delete the processed data after the retention time. The customer is responsible to store and process the data in its system, before we delete it.
- Data retrieval and C-Hub cloud backend interaction with REST APIs
- Offer customers the possibility to manage their vehicle measurements configuration, secure data transfer from vehicles to the C-Hub cloud backend, and job updates
- For custom vehicle apps:
 - Handle container image as "black box" and test the boundaries and interfaces provided to the vehicle app containers (image checksum, successful start of container, internet connectivity, access to the Vehicle API, ...)

7.4.2 Recommendations for developers

The following should be avoided with regard to the C-Hub TCU2 devices and the C-Hub cloud services:

What to avoid	Why
Do not interrupt the power supply of the C-Hub TCU2 devices during an auto-pairing process.	To successfully finish a pairing process, the device should only be powered off when the response from the cloud backend was received. Interruption in the power supply during the process might lead to an inconsistent state between the device and a cloud backend service.
Do not start multiple pairing attempts at once.	Before starting a new pairing attempt wait for the cloud backend response.
Do not interrupt the power supply of the C-Hub TCU2 devices during a self-update.	If customers (or drivers) interrupt the power supply during a self-update, the self-update will fail. Failures due to such interruptions of power supply are not considered in any service level or report calculation.
Do not exceed the message size limitation of the Vehicle API for communication of the containerized vehicle apps with the C-Hub cloud backend, in particular 100 KB.	Messages that exceed the defined maximum message size will be dropped by the backend and impact your service in a negative way.
Do not consider the online state of C-Hub TCU2 device to be the real-time state.	Due to the nature of unstable mobile internet connections, the online state of a device cannot be considered real time. It might take a few minutes for a device to be shown as offline in the backend.
Do not open a very high number of connections to the Kafka broker.	The cloud backend might close connections to the broker unexpectedly if a client initiates too many connections.
Do not poll the REST API for real-time updates.	A client exceeding a certain threshold for polling the REST API might get blocked for some time. Use the Kafka API for all things that are considered event-based. Use the Kafka API for low-latency, programmatic function call status updates.
Do not use remote diagnostics for use cases which exceed the risk evaluation of "Quality Management" level in the sense of ISO 26262.	
Do not use remote diagnostics to trigger functions with safety requirements, unless suitable safety measures are implemented to guarantee the safe operation of the vehicle.	The definition and implementation of safety measures is a responsibility of the customer.

What to avoid	Why
Do not use remote diagnostics for use cases which require real-time execution of in-vehicle functions (with low latency requirements).	Due to over-the-air communication, both commands and status updates might be delayed.
Do not use remote diagnostics to transfer binary large objects (blobs) from backend to vehicle or vice versa.	Request and response validation will not be applicable to data of this kind and performance may decrease. Also, payload size for function call bodies and response bodies is limited.
Do not trigger the execution of any untested and unverified scripts/functions in vehicles.	
Do not report any inventory which contains personal data without the knowledge and/or consent of the driver or other people involved.	
Do not report data via an inventory which is not intended for the purpose of reporting installed hardware and software.	The collection of data needs to be evaluated per use case.
Do not decrease the device performance.	Adding multiple measurement configurations and capturing redundant vehicle data will increase the device load and thus decrease the performance of the device (in case of device overload). The device performance may not be up to the expected level.
No misconfiguration of the signal sampling / up-sampling.	Technically, a signal can only be down-sampled, i.e. with the time frequency equal or less to that of the frequency of the signal in CAN. For example, if Signal1 arrives in CAN every 5 seconds, the signal (down-)sampling rate can be greater than or equal to 5 seconds, but not less than 5 seconds. Even if specified as less than 5 seconds, the signal will only be acquired every 5 seconds.
Do not exceed the data buffer limits.	The designing of pre-trigger and post-trigger time intervals needs to be consciously defined, and in case of very high time intervals, this might exceed the buffer limits.
Do not change any pre-processing algorithms on the device.	Adding new preprocessing algorithms or changing existing preprocessing algorithms can only be done through an update of the device software and is therefore also limited to the release timeline of the device software.

7.4.3 Notes about containerized vehicle apps

Please, consider the following information regarding the additional containerized vehicle apps which can be deployed on your C-Hub TCU2 devices:

- To securely run third-party *containerized vehicle apps* on in-vehicle embedded devices (beside Bosch software components), such third-party software components must be packaged and executed as OCI containers (industry standard).

This provides basic isolation mechanisms (resource consumption, system access, etc.) for executing third-party vehicle apps as "black boxes" on these devices.

- Third-party software running inside a OCI container can access a programming language-agnostic Vehicle API, which is available via gRPC (industry standard)
- We utilize the desired state feature of OTA Updates to manage the lifecycle of such *containerized vehicle apps*.
- These *containerized vehicle apps* will have their own update lifecycle, which is independent from the self-updates of the C-Hub TCU2 device software image.
No third-party *containerized vehicle app* is "preinstalled" as part of a Bosch device software image (via initial plant installation or self-update)
- All *containerized vehicle apps* in the fleet can be updated individually via the C-Hub backend *tenant* of the OEM customer (e.g. by an OEM customer's fleet manager).
Therefore, we assume that the OEM customer's suppliers will not have direct access to the productive C-Hub backend tenant.
- The *containerized vehicle apps* can be uninstalled, in contrast to OTA Updates.
- It is possible to remotely activate/deactivate the *containerized vehicle apps* (i.e. to start, stop, and restart containers based on the desired state description via OTA Updates.
- When configuring resource limits for containers, the sum of all granted resources should not exceed the actual available amount of resources - meaning the resources shall not be over-committed.

8 Lifetime of the hardware

The life of the C-Hub TCU2 device and its accessories is designed for the parameters stipulated in the list below.

As soon as one of the stipulated parameters is fulfilled, the lifetime of the product is fulfilled, even if other parameters have not yet been fulfilled.

This lifetime statement must be differentiated from the applicable warranty period which expires independently from the duration of this lifetime statement.

- Lifetime: 10 years
- Milage: 1.200.000 km
- ECU operating time: 20.000 h

Bosch trusts that the CUSTOMER will assess if the Bosch-product's lifetime specified here is in line with the relevant legal lifetime requirements applicable to the CUSTOMER's system in the specific target country and whether any particular measures need to be taken in order to comply with relevant legal lifetime requirements (e.g. if the Bosch-product needs to be replaced after the Bosch-product's lifetime period).

9 Recycling, repair, and disposal of the hardware

Bosch marks all mechanical parts according to VDA260.

9.1 Recycling and reusing the device

In case the C-Hub TCU2 device is unmounted of a vehicle and you want to re-use the device in a different vehicle, please reach out to the C-Hub support for further instructions.

9.2 Repair

Repairing the C-Hub TCU2 device is not possible.

9.3 Disposal

According to the [EU directive for end-of-life vehicles \(ELVs\)](#) Bosch declares that the C-Hub TCU2 device cannot be repaired.

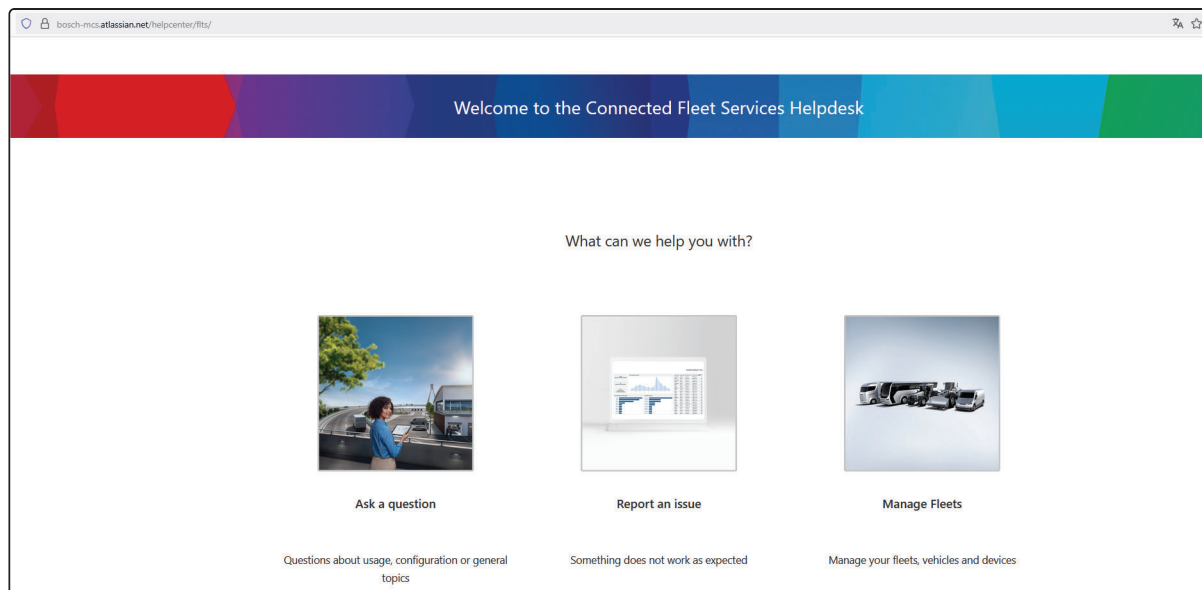
In case the hardware is unmounted of the vehicle because of a defect it needs to be disposed of in accordance with the country-specific regulation.

Disposing of this product correctly will help save valuable resources and prevent any potential negative effects on human health and the environment, which could otherwise arise from inappropriate waste handling.

Please contact your local authority for further details of your nearest designated collection point. Penalties may be applicable for incorrect disposal of this waste, in accordance with your national legislation.

10 Support - Helpdesk

Use the helpdesk to raise various types of service requests: <https://bos.ch/helpdesk>.



In case you cannot access the board yet, please get in touch with your Bosch customer success manager, who will assist with your onboarding.

10.1 Ask a question

This section can be used for any type of questions about existing or upcoming features of any Bosch service for fleets.

This category covers everything related to C-Hub which is not an incident or a service request, for example this could be a request for information.

10.2 Report an issue

This section can be used if the system does not work as described in the documentation.

This category is suitable for any unexpected, unusual, or wrong behavior of a service part. By default, the priority is set to low.

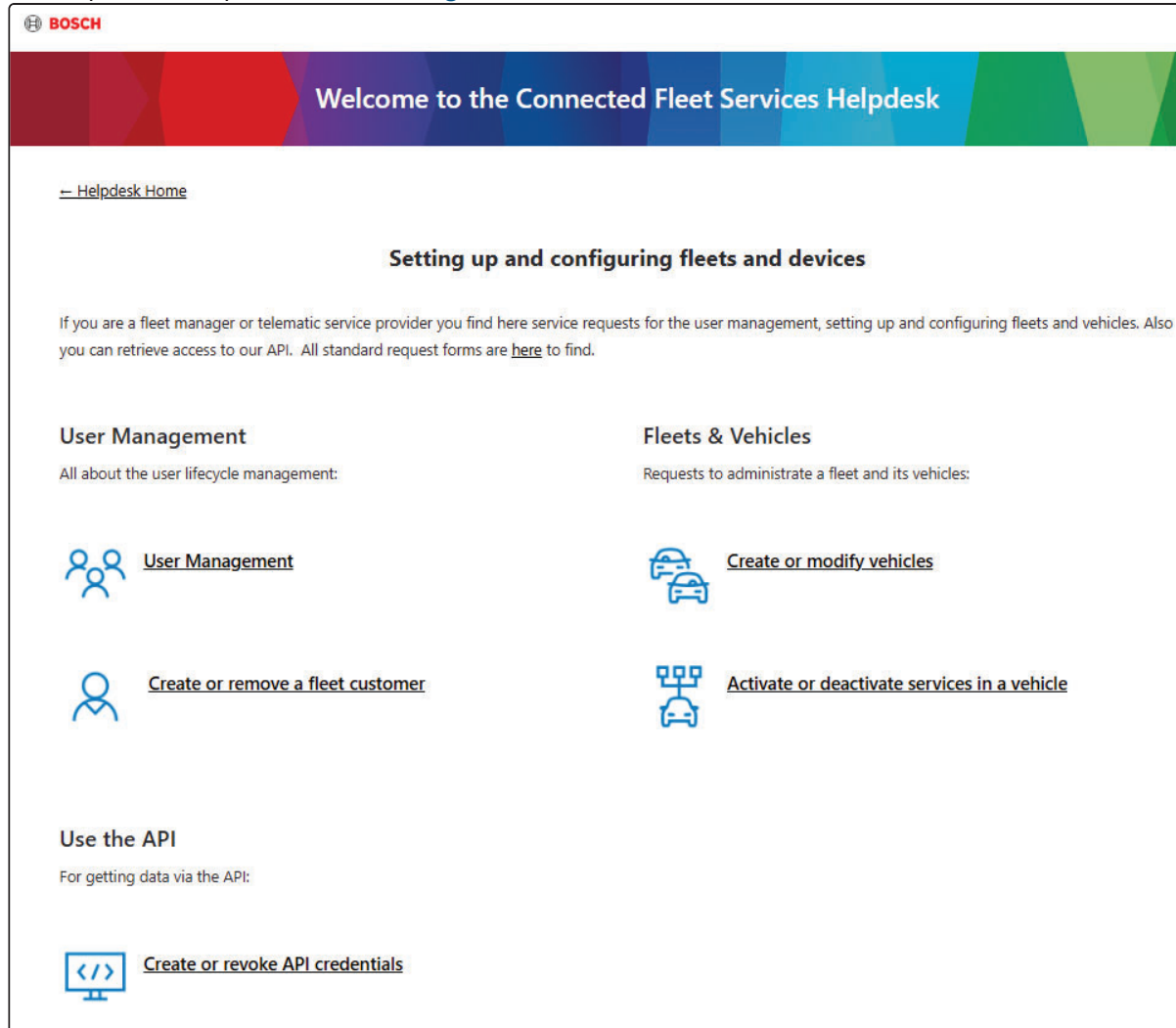
The priority options are **low**, **medium**, and **high**, depending on the impact. Please, only use **high** priority for major incidents, as fixing the problem would involve high costs due to the much higher priority. Such could be for example a problem affecting an entire business service or a major part

of it. An incident is considered major if it makes a customer's service unavailable, impacts the customer's business, and/or has a potential financial impact.

10.3 Manage fleets

This section can be used for different types of tickets.

More options are provided at [Manage Fleets](#).



Preliminary

10.3.1 User Management

- Technical onboarding of a new customer
- User Management
- Create or remove a fleet customer

10.3.2 Fleets & Vehicles

Requests to administrate a fleet and its vehicles:

- Create, modify or delete a C-Hub tenant
- Create, modify or delete a client (for a C-Hub tenant)
- Create or modify vehicles
- Activate or deactivate services in a vehicle

10.3.3 Use the API

- Create or revoke API credentials



For any type of ticket raised at <https://bos.ch/helpdesk>, the system will automatically inform you by email regarding its progress.

11 Annex

- FMS standard excerpt
- Kafka references
 - Customer load test
 - Device-Vehicle mapping
 - OTA Data Exporter
 - OTA Function Calls
 - OTA Updates Campaign Exporter
 - OTA Updates Distribution Exporter
 - OTA Updates Inventory Exporter
- C-Hub glossary
- Declaration of conformity - C-Hub TCU2 device
- Additional accessories for TSP customers - not part of the standard offering for OEM
 - DIN Adapter - for TSP
 - Declaration of conformity C-Hub DIN adapter
- Additional information for OEM customers
 - Main connector - pinning info
 - Diagnostic Trouble Codes - C-Hub TCU2 device
 - Auxiliary equipment - antenna and cabling EU and NA
 - V2 - Auxiliary equipment - antenna and cabling EU and NA

11.1 FMS standard excerpt

The FMS interface is an **optional** interface in trucks and busses of different manufacturers, like DAF, Daimler, IVECO, MAN, Renault, Scania.

This interface enables C-Hub TCU2 devices to **securely** collect vehicle data, whereas the C-Hub backend enables the sharing of such data with third-parties via cloud services.



The complete standard is available online for free https://fms-standard.com/Truck/download/fms%20document_v_05_vers.11.11.2025.pdf

11.1.1 Parameters for FMS Gateway - according to SAE J1939

- Parameters for Bus and Truck FMS-Standard
 - FMS Tell Tale Status details
- Parameters for Truck FMS-Standard
- Parameters for Bus FMS-Standard

Parameters for Bus and Truck FMS-Standard

Parameter	ID	PGN	Rep. Rate
Fuel Consumption	LFC	65,257	1 000 ms
Dash Display 1	DD1	65,276	1 000 ms
Electronic Engine Controller #1	EEC1	61,444	20 ms
Engine Hours, Revolutions	HOURS	65,253	1 000 ms
Vehicle Identification	VI	65,260	10 000 ms
FMS-standard Interface Identity / Capabilities	FMS	64,977	10 000 ms
High Resolution Vehicle Distance	VDHR	65,217	1 000 ms
Tachograph	TCO1	65,132	20 ms/ 50 ms
Engine Temperature 1	ET1	65,262	1 000 ms
Ambient Conditions	AMB	65,269	1 000 ms
Driver's Identification	DI	65,131	10 000 ms
Fuel Economy	LFE	65,266	100 ms
Air Supply Pressure	AIR1	65,198	1 000 ms
High Resolution Fuel Consumption (Liquid)	HRLFC	64,777	1 000 ms
Aftertreatment 1 Diesel Exhaust Fluid Tank 1 Information	AT1T1I	65,110	1 000 ms
FMS Tell Tale Status see details in table below	FMS1	64,893	1 000 ms
Electronic Brake Controller 1	EBC1	61,441	100 ms
Electronic Engine Controller 14	EEC14	64,962	10 000 ms
Fuel Consumption (Gaseous)	GFC	65,199	1 000 ms
Electronic Retarder Controller 1	ERC1	61,440	100 ms
Vehicle Weight	VW	65,258	1 000 ms

Preliminary

Parameter	ID	PGN	Rep. Rate
EVSE 1 DC Status 1	EVSE1 DCS1	64,184	1 000 ms
EV Operator Indicators 1	EVOI1	64,617	every 10 s and on change but no faster than 1 s
Tire Condition Message 1	TIRE1	65,268	10 000 ms
Operator Indicators	OI	65,279	10 s or on change but no more often than 1 s

FMS Tell Tale Status details

Block ID	Tell Tale Status	Name	Tell Tale ID
0	▪ 1 ▪ 2 ▪ 3 ▪ 4 ▪ 5 ▪ 6 ▪ 7 ▪ 8 ▪ 9 ▪ 10 ▪ 11 ▪ 12 ▪ 13 ▪ 14 ▪ 15	▪ Cooling air conditioning ▪ High beam, main beam ▪ Low beam, dipped beam ▪ Turn signals ▪ Hazard warning ▪ Provision for the disabled or handicapped persons ▪ Parking Brake ▪ Brake failure/brake system malfunction ▪ Hatch open ▪ Fuel level ▪ Engine coolant temperature ▪ Battery charging condition ▪ Engine oil ▪ Position lights,side lights ▪ Front fog light	1 to 15
1	▪ 1 ▪ 2 ▪ 3 ▪ 4 ▪ 5 ▪ 6 ▪ 7 ▪ 8 ▪ 9 ▪ 10 ▪ 11 ▪ 12 ▪ 13 ▪ 14 ▪ 15	▪ Rear fog light ▪ Park Heating ▪ Engine / Mil indicator ▪ Service, call for maintenance ▪ Transmission fluid temperature ▪ Transmission failure/ malfunction ▪ Anti-lock brake system failure ▪ Worn brake linings ▪ Windscreen washer fluid/ windshield washer fluid ▪ Tire failure/malfunction ▪ Malfunction/general failure ▪ Engine oil temperature ▪ Engine oil level ▪ Engine coolant level ▪ Steering fluid level	16 to 30

Preliminary

Block ID	Tell Tale Status	Name	Tell Tale ID
2	▪ 1 ▪ 2 ▪ 3 ▪ 4 ▪ 5 ▪ 6 ▪ 7 ▪ 8 ▪ 9 ▪ 10 ▪ 11 ▪ 12 ▪ 13 ▪ 14 ▪ 15	▪ Steering failure ▪ Height Control (Levelling) ▪ Retarder ▪ Engine Emission system failure (Mil indicator) ▪ ESC indication ▪ Brake lights ▪ Articulation ▪ Stop Request ▪ Pram request ▪ Bus stop brake ▪ AdBlue level ▪ Raising ▪ Lowering ▪ Kneeling ▪ Engine compartment temperature	31 to 45
3	▪ 1 ▪ 2 ▪ 3 ▪ 4 ▪ 5 ▪ 6 ▪ 7 ▪ 8 ▪ 9 ▪ 10 ▪ 11 ▪ 12 ▪ 13 ▪ 14 ▪ 15	▪ Auxiliary air pressure ▪ Air filter clogged ▪ Fuel filter differential pressure ▪ Seat belt ▪ EBS ▪ Lane departure indication ▪ Advanced emergency braking system ▪ ACC ▪ Trailer connected ▪ ABS Trailer 1,2 ▪ Airbag ▪ EBS Trailer 1,2 ▪ Tachograph indication ▪ ESC switched off ▪ Lane departure warning switched off	46 to 60
4	▪ 1 ▪ 2 ▪ 3 ▪ 4 ▪ 5 ▪ 6 ▪ 7 ▪ 8 ▪ 9 ▪ 10 ▪ 11 ▪ 12 ▪ 13 ▪ 14 ▪ 15	▪ Engine emission filter (Soot Filter) ▪ Electric motor failures ▪ AdBlue tampering ▪ Multiplex System ▪ Battery pack ▪ High voltage system caution ▪ Battery pack temperature ▪ Limited performance electric motor ▪ Battery pack cooling ▪ Reserved for FMS-Standard ▪ Reserved for FMS-Standard ▪ Reserved for FMS-Standard ▪ Reserved for FMS-Standard ▪ Reserved for FMS-Standard ▪ Reserved for FMS-Standard	61 to 75

Preliminary

Parameters for Truck FMS-Standard

Parameter	ID	PGN	Rep. Rate
Cruise Control/Vehicle Speed 1	CCVS1	65,265	100 ms

Parameter	ID	PGN	Rep. Rate
Electronic Engine Controller #2	EEC2	61,443	50 ms
Service Information	SERV	65,216	1 000 ms
PTO Drive Engagement	PTODE	64,932	100 ms
Combination Vehicle Weight	CVW	65,136	10 000 ms

Parameters for Bus FMS-Standard

Parameter	ID	PGN	Rep. Rate
Cruise Control/Vehicle Speed	CCVS	65,265	100 ms
Electronic Engine Controller #2	EEC2	61,443	50 ms
Door Control 1	DC1	65,102	100 ms
Door Control 2	DC2	64,933	100 ms
Time / Date	TD	65,254	1 000 ms
Alternator Speed	AS	65,237	1 000 ms
Electronic Transmission Controller 2	ETC2	61,445	100 ms
Air Suspension Control 4	ASC4	65,112	100 ms
Vehicle Electrical Power #4	VEP4	64,695	10 000 ms
Vehicle Dynamic Stability Control 2	VDC2	61,449	100 ms
Engine Temperature 1	ET1	65,262	1 000 ms
Ambient Conditions	AMB	65,269	1 000 ms
High Voltage Energy Storage System Thermal Management System Status 1	HVESST S1	61,705	100 ms
High Voltage Energy Storage System History	HVESSTH IST	64,606	10 000 ms
Hybrid System Status 1	HSS1	64,706	every 1 s and on change but no faster than 100 ms

Preliminary

Parameter	ID	PGN	Rep. Rate
Engine Fluid Level/Pressure 1	EFL/P1	65,263	500 ms
Transmission Fluids 1	TRF1	65,272	1 000 ms

Preliminary

11.2 Kafka references

- [Customer load test](#)
- [Device-Vehicle mapping](#)
- [OTA Data Exporter](#)
- [OTA Function Calls](#)
- [OTA Updates Campaign Exporter](#)
- [OTA Updates Distribution Exporter](#)
- [OTA Updates Inventory Exporter](#)

11.2.1 Customer load test

Customer load test 1.0.0

Kafka topics for customer to test against the C-Hub confluent cluster/PubSub

- [Operations](#)
 - [SUB customer-loadtest](#)

Operations

SUB customer-loadtest

Topic to test load

Accepts the following message: **<anonymous-message-1>**

11.2.2 Device-Vehicle mapping

Self Mapping Service 1.0.0

Kafka topics provided by the Device-Vehicle mapping (Self-Mapping) service.

This API allows you to receive events that occur in case a new VIN is mapped, which is not matching to the originally mapped VIN. In this case, a so-called VIN-mismatch is detected, and an event is triggered.

At some point in the future, these topics may be migrated into a topic belonging to Device Management or Vehicle Management.

- [Operations](#)
 - [SUB ps.\\$tenantId.device-vehicle-mapping.events.vin-mismatch](#)
 - [VIN mismatch event](#)
 - [Payload example - VIN mismatch event](#)

Operations

SUB ps.\$tenantId.device-vehicle-mapping.events.vin-mismatch

Receive VIN mismatch events that occur in the context of a device-initiated vehicle mapping (also known as self-mapping).

Accepts the following message: VIN mismatch event

VIN mismatch event

This event occurs if the device reports a VIN that is not marked as trusted (and the service is configured not to accept untrusted VINs).

Payload example - VIN mismatch event

```
{
  "deviceId": "string - The device ID of the device.",
  "deviceSerialNumber": "string - The serial number of the device.",
  "detectedVin": "string - The VIN reported by the device.",
  "pairedVin": "string - The VIN of the vehicle that the device representation is paired to in the backend.
  Undefined if the device is not paired to a vehicle."
}
```

All properties are required, except **pairedVin** . Additional properties are allowed.

11.2.3 OTA Data Exporter

OTA Data Exporter API 1.0.0

The OTA Data Exporter API allows you to consume events that happen on behalf of the OTA Data.

This subscription is contrasting to the REST API, because there the consumers of the API have to actively poll to notice any changes.

- [Operations](#)
 - [SUB ps.\\$tenantId.measurements.results](#)
 - [measurementDataExtendedExportMessage](#)
 - [Header example - measurementDataExtendedExportMessage](#)
 - [Payload example 1 - measurementDataExtendedExportMessage](#)
 - [Payload example 2 - measurementDataExtendedExportMessage](#)
 - [Payload example 3 - measurementDataExtendedExportMessage](#)
 - [measurementDataShortExportMessage](#)
 - [Header example - measurementDataShortExportMessage](#)
 - [Payload example - measurementDataShortExportMessage](#)
 - [measurementDataReducedExportMessage](#)
 - [Header example - measurementDataReducedExportMessage](#)
 - [Payload example - measurementDataReducedExportMessage](#)
 - [SUB ps.\\$tenantId.diagnostics.results](#)
 - [dtcsSnapshotMessage](#)
 - [Header example - dtcsSnapshotMessage](#)
 - [Payload example - dtcsSnapshotMessage](#)
 - [SUB ps.\\$tenantId.vehicles.management.events](#)
 - [deviceMappingExportMessage](#)
 - [Header example - deviceMappingExportMessage](#)
 - [Payload example - deviceMappingExportMessage](#)
 - [vehicleCreationExportMessage](#)
 - [Header example - vehicleCreationExportMessage](#)
 - [Payload example - vehicleCreationExportMessage](#)
 - [vehicleUpdatedExportMessage](#)
 - [Header example - vehicleUpdatedExportMessage](#)
 - [Payload example - vehicleUpdatedExportMessage](#)
 - [deviceCreationExportMessage](#)
 - [Header example - deviceCreationExportMessage](#)
 - [Payload example - deviceCreationExportMessage](#)
 - [deviceDeletionExportMessage](#)
 - [Header example - deviceDeletionExportMessage](#)
 - [Payload example - deviceDeletionExportMessage](#)
 - [gpsStateChangedExportMessage](#)
 - [Header example - gpsStateChangedExportMessage](#)
 - [Payload example - gpsStateChangedExportMessage](#)
 - [deviceUnmappingExportMessage](#)
 - [Header example - deviceUnmappingExportMessage](#)
 - [Payload example - deviceUnmappingExportMessage](#)

Operations

SUB ps.\$tenantId.measurements.results

Message that is sent for every Measurement Job Result.

A measurement result that is exported, mainly contains the following sections:

- metaData,
 - resultData and
 - reasonData (optional).
- reasonData** is always present for single shot measurement, as these measurements are mostly defined with start triggers.
- reasonData** is optional for continuous measurement results, depending on the trigger definitions.

The level of details in the **metaData** and **resultData** exported is determined by the selected export format.

Operation ID: **subscribeMeasurementsResults**

Accepts **one of** the following messages:

- **measurementDataExtendedExportMessage**
- **measurementDataShortExportMessage**
- **measurementDataReducedExportMessage**

measurementDataExtendedExportMessage

Message that is sent for every Measurement Job Result in EXTENDED format.

- The **metaData** section at the beginning of the payload contains vehicle and device information.
- The **resultData** holds **seriesParts** which includes **series** and **dataPoints** with timestamps.
- The **reasonData** contains the complete true tree with the fired trigger elements.
- Additionally, each signal/message will contain name, actual values along with the timestamp i.e. recorded on the device, unique identifier, source and signal information.

Header example - measurementDataExtendedExportMessage

```
{
  "mjrId": "mjrUUID",
  "measurementConfigurationId": "measurementConfigurationUUID",
  "measurementConfigurationName": "measurementConfigurationName",
  "releasedMeasurementConfigurationVersion": 899,
  "testStepId": "testStepUUID",
  "tenant-id": "tenantId",
  "bootCycleId": "bootCycleUUID",
  "mjrCounter": 0,
  "message-type": "measurementDataExportMessage",
  "vin": "VIN123",
  "vehicleId": "vehicleUUID"
}
```

Payload example 1 - measurementDataExtendedExportMessage

```
{
  "metaData": {
    "vehicle": {
      "vehicleId": "vehicleUUID",
      "vehicleName": "vehicleName123",

```

```

    "vin": "VIN123",
    "manufacturerName": "manufacturer1",
    "modelName": "AUTO_PAired",
    "devices": [
      {
        "deviceId": "deviceId123",
        "deviceType": "GENERIC_CCU",
        "deviceSerialNumber": "deviceSerialNumber123",
        "deviceSwVersion": "deviceSoftwareVersion"
      }
    ]
  },
  "resultData": {
    "testStepId": "testStepUUID",
    "measurementJobResultId": "measurementJobResultUUID",
    "packageIndex": 0,
    "lastPackage": true,
    "seriesParts": [
      {
        "series": {
          "uid": "seriesUUID",
          "seriesName": "seriesName",
          "valueType": "DOUBLE",
          "signalCollectionId": "signalCollectionUUID",
          "sourceId": "sourceUUID",
          "sourceName": "sourceName",
          "unit": "%",
          "protocol": "CAN_RAW",
          "type": "SIGNAL",
          "samplingRate": 0
        },
        "dataPoints": [
          {
            "uniqueId": "UUID",
            "timeStamp": "1999-07-25T16:30:32.743Z",
            "value": 20
          },
          {
            "uniqueId": "UUID",
            "timeStamp": "1999-07-25T16:30:53.623Z",
            "value": 40
          }
        ]
      }
    ]
  },
  "start": "1999-07-25T16:30:32.743Z",
  "end": "1999-07-25T16:30:53.623Z"
},
"reasonData": {
  "start": {
    "reason": "START_TRIGGER",
    "timeStamp": "1999-07-25T16:30:32.743Z",
    "name": "triggerStartGroup",
    "triggerElements": [
      {
        "name": "triggerCondition",
        "triggerFired": true
      }
    ]
  },
  "stop": {
    "reason": "STOP_TRIGGER",
    "timeStamp": "1999-07-25T16:30:53.623Z",
    "name": "triggerStopGroup",
    "triggerElements": [
      {
        "name": "triggerCondition",
        "triggerFired": true
      }
    ]
  }
},
"correlations": {
  "vehicleId": "vehicleUUID",
  "vin": "VIN123",
  "measurementConfigurationId": "measurementConfigurationUUID",
  "measurementConfigurationName": "measurementConfigurationName",
  "releasedMeasurementConfigurationVersion": 899,
  "bootCycleId": "bootCycleUUID"
}
}

```

Preliminary

Payload example 2 - measurementDataExtendedExportMessage

EXTENDED export format with matrix of **ndArray** in the payload.

```
{
  "metaData": {
    "vehicle": {
      "vehicleId": "vehicleUUID",
      "vehicleName": "Doc Vehicle",
      "vin": "VIN123",
      "manufacturerName": "Doc manufacturer",
      "modelName": "modelName"
    },
    "customMetadata": {
      "key": "value"
    }
  },
  "resultData": {
    "testStepId": "testStepUUID",
    "measurementJobResultId": "measurementJobResultUUID",
    "packageIndex": 2,
    "lastPackage": true,
    "seriesParts": [
      {
        "series": {
          "uid": "seriesUUID",
          "seriesName": "Series_A",
          "valueType": "FLOAT",
          "signalCollectionId": "signalCollectionUUID",
          "sourceId": "sourceUUID",
          "sourceName": "sourceName",
          "unit": "bar",
          "protocol": "CAN_RAW",
          "type": "SIGNAL",
          "samplingRate": 100
        },
        "dataPoints": [
          {
            "uniqueId": "UUID",
            "timeStamp": "2017-08-01T08:00:00Z",
            "value": 0
          },
          {
            "uniqueId": "UUID",
            "timeStamp": "2017-08-01T08:00:01Z",
            "value": 1
          },
          {
            "uniqueId": "UUID",
            "timeStamp": "2017-08-01T08:00:02Z",
            "value": 2
          },
          {
            "uniqueId": "UUID",
            "timeStamp": "2017-08-01T08:00:03Z",
            "value": 3
          },
          {
            "uniqueId": "UUID",
            "timeStamp": "2017-08-01T08:00:04Z",
            "value": 4
          },
          {
            "uniqueId": "UUID",
            "timeStamp": "2017-08-01T08:00:05Z",
            "value": 5
          },
          {
            "uniqueId": "UUID",
            "timeStamp": "2017-08-01T08:00:06Z",
            "value": 6
          },
          {
            "uniqueId": "UUID",
            "timeStamp": "2017-08-01T08:00:07Z",
            "value": 7
          },
          {
            "uniqueId": "UUID",
            "timeStamp": "2017-08-01T08:00:08Z",
            "value": 8
          },
          {
            "uniqueId": "UUID",
            "timeStamp": "2017-08-01T08:00:09Z",
            "value": 9
          }
        ]
      }
    ]
  }
}
```

Preliminary

```

        "uniqueId": "UUID",
        "timeStamp": "2017-08-01T08:00:10Z",
        "value": 10
    }
  ],
},
{
  "series": {
    "uid": "seriesUUID",
    "seriesName": "Series_B",
    "valueType": "SCALAR_LIST",
    "signalCollectionId": "signalCollectionUUID",
    "sourceId": "sourceUUID",
    "sourceName": "sourceName",
    "unit": "l",
    "protocol": "CAN_RAW",
    "type": "SIGNAL",
    "samplingRate": 100
  },
  "dataPoints": [
    {
      "uniqueId": "UUID",
      "timeStamp": "2017-08-01T08:00:12Z",
      "value": [
        6,
        12,
        24
      ]
    },
    {
      "uniqueId": "UUID",
      "timeStamp": "2017-08-01T08:00:08Z",
      "value": [
        4,
        8,
        16
      ]
    },
    {
      "uniqueId": "UUID",
      "timeStamp": "2017-08-01T08:00:04Z",
      "value": [
        2,
        4,
        8
      ]
    },
    {
      "uniqueId": "UUID",
      "timeStamp": "2017-08-01T08:00:00Z",
      "value": [
        0,
        0,
        0
      ]
    }
  ]
},
{
  "series": {
    "uid": "seriesUUID",
    "seriesName": "Message_A",
    "valueType": "FLOAT",
    "signalCollectionId": "signalCollectionUUID",
    "sourceId": "sourceUUID",
    "sourceName": "sourceName",
    "unit": "grad",
    "protocol": "CAN_RAW",
    "type": "MESSAGE",
    "samplingRate": 100
  },
  "dataPoints": [
    {
      "uniqueId": "UUID",
      "timeStamp": "2017-08-01T08:00:00Z",
      "value": 0
    },
    {
      "uniqueId": "UUID",
      "timeStamp": "2017-08-01T08:00:03Z",
      "value": 5
    },
    {
      "uniqueId": "UUID",
      "timeStamp": "2017-08-01T08:00:06Z",
      "value": 10
    },
    {
      "uniqueId": "UUID",

```

```

        "timestamp": "2017-08-01T08:00:09Z",
        "value": 15
    },
    {
        "uniqueId": "UUID",
        "timestamp": "2017-08-01T08:00:12Z",
        "value": 20
    }
]
},
{
    "series": {
        "uid": "seriesUUID",
        "seriesName": "TwoDStatistik",
        "valueType": "CLASSIFICATION_ND",
        "unit": "",
        "samplingRate": 0
    },
    "dataPoints": [
        {
            "uniqueId": "UUID",
            "timestamp": "2017-08-01T08:00:00Z",
            "value": {
                "dimension": 2,
                "shape": [
                    10,
                    5
                ],
                "startTime": "2017-08-01T08:00:00Z",
                "endTime": "2017-08-01T08:00:12Z",
                "ndArray": [
                    [
                        0,
                        120.394,
                        0,
                        0,
                        0
                    ],
                    [
                        0,
                        0,
                        138.407,
                        0,
                        0
                    ],
                    [
                        0,
                        0,
                        2.011,
                        0,
                        0
                    ],
                    [
                        0,
                        0,
                        0,
                        4.012,
                        0
                    ],
                    [
                        2.001,
                        0,
                        0,
                        0,
                        0
                    ],
                    [
                        0,
                        0,
                        4.01,
                        0,
                        0
                    ],
                    [
                        0,
                        0,
                        0,
                        0,
                        26.064
                    ],
                    [
                        0,
                        0,
                        0,
                        0,
                        0
                    ]
                ]
            }
        ]
    ]
}

```

Preliminary

```

        0,
        0,
        0,
        0
      ],
      [
        0,
        0,
        0,
        0,
        0
      ]
    ]
  }
}
],
"start": "2017-08-01T08:00:00Z",
"end": "2017-08-01T08:00:12Z"
},
"reasonData": {
  "start": {
    "reason": "START_TRIGGER",
    "timeStamp": "2017-08-01T08:00:00.181Z",
    "name": "startTriggerName",
    "triggerElements": [
      {
        "name": "triggerElementConditionSignalName1",
        "triggerFired": true
      },
      {
        "name": "triggerElementConditionSignalName2",
        "triggerFired": true
      }
    ]
  },
  "stop": {
    "reason": "JOB_UNINSTALLATION",
    "timeStamp": "2017-08-01T08:00:12.123Z"
  }
},
"correlations": {
  "vehicleId": "vehicleUUID",
  "vin": "VIN123",
  "measurementConfigurationId": "measurementConfigurationUUID",
  "measurementConfigurationName": "measurementConfigurationName",
  "releasedMeasurementConfigurationVersion": 0,
  "bootCycleId": "bootCycleUUID"
}
}
}

```

Preliminary

Payload example 3 - measurementDataExtendedExportMessage

EXTENDED export format with sparse matrix in Yale format in the payload

```

{
  "metaData": {
    "vehicle": {
      "vehicleId": "vehicleUUID",
      "vehicleName": "Doc Vehicle",
      "vin": "VIN123",
      "manufacturerName": "Doc manufacturer",
      "modelName": "modelName"
    },
    "customMetadata": {
      "key": "value"
    }
  },
  "resultData": {
    "testStepId": "testStepUUID",
    "measurementJobResultId": "measurementJobResultUUID",
    "packageIndex": 2,
    "lastPackage": true,
    "seriesParts": [
      {
        "series": {
          "uid": "seriesUUID",
          "seriesName": "seriesName",
          "valueType": "CLASSIFICATION_ND",
          "unit": "",
          "samplingRate": 0
        }
      }
    ]
  }
}

```

```

    "dataPoints": [
      {
        "uniqueId": "UUID",
        "timeStamp": "2017-08-01T08:00:00Z",
        "value": {
          "dimension": 2,
          "shape": [
            10,
            5
          ],
          "startTime": "2017-08-01T08:00:00Z",
          "endTime": "2017-08-01T08:00:12Z",
          "sparseMatrix": {
            "A": [
              120.394,
              138.407,
              2.011,
              4.012,
              2.001,
              4.01,
              26.064
            ],
            "IA": [
              0,
              1,
              2,
              3,
              4,
              5,
              6,
              7,
              7,
              7,
              7
            ],
            "JA": [
              1,
              2,
              2,
              3,
              0,
              2,
              4
            ]
          }
        }
      ]
    },
    "start": "2017-08-01T08:00:00Z",
    "end": "2017-08-01T08:00:12Z"
  },
  "correlations": {
    "vehicleId": "vehicleUUID",
    "vin": "VIN123",
    "measurementConfigurationId": "measurementConfigurationUUID",
    "measurementConfigurationName": "measurementConfigurationName",
    "releasedMeasurementConfigurationVersion": 0,
    "bootCycleId": "bootCycleUUID"
  }
}

```

measurementDataShortExportMessage

Message that is sent for every Measurement Job Result in SHORT format.

For Short export format, only the series name and value are provided (without individual timestamps) in the export. Instead only measurement start and end, timestamps are provided in the metaData section.

MetaData section at the beginning containing vehicle and device information, containing start and end timestamp of the overall measured data.

Each signal/message will contain only name and value and does not contain signal/message level timestamps.

Header example - measurementDataShortExportMessage

```
{
```

```

    "mjrId": "mjrUUID",
    "measurementConfigurationId": "measurementConfigurationUUID",
    "measurementConfigurationName": "measurementConfigurationName",
    "releasedMeasurementConfigurationVersion": 3,
    "testStepId": "testStepUUID",
    "tenant-id": "tenantId",
    "bootCycleId": "bootCycleUUID",
    "mjrCounter": 0,
    "message-type": "measurementDataExportMessage",
    "vin": "VIN123",
    "vehicleId": "vehicleUUID"
  }

```

Payload example - measurementDataShortExportMessage

SHORT export format

```

{
  "metaData": {
    "vehicleId": "vehicleUUID",
    "deviceId": "deviceId123",
    "deviceSwVersion": "4.4.4",
    "vin": "VIN123",
    "measurementConfigurationId": "measurementConfigurationUUID",
    "measurementConfigurationName": "measurementConfigurationName",
    "releasedMeasurementConfigurationVersion": 3,
    "bootCycleId": "bootCycleUUID",
    "testStepId": "testStepUUID",
    "measurementJobResultId": "measurementJobResultUUID",
    "start": 172154000000,
    "end": 172154770000,
    "packageIndex": 0,
    "lastPackage": false,
    "uploaded": 172154000000,
    "exported": 172154770000,
    "customMetadata": {
      "key1": "value1",
      "key2": "value2"
    }
  },
  "resultData": {
    "FirstWarningsList": [
      155,
      156
    ],
    "BrakePedalPosition": 0,
    "EngineCoolantTemperature": 99,
    "EngineSpeed": 555.5,
    "TotalVehicleDistanceHighResolution": 20000000,
    "WarningId_List": [],
    "FuelLevel1": 76.4,
    "AmbientAirTemperature": 25,
    "CruiseControlActive": 0,
    "GPS": {
      "longitude": 47.421066,
      "latitude": 10.985365,
      "elevation": 1,
      "gpsTime": 172154700000,
      "direction": 111,
      "gpsSpeed": 0,
      "lastReadingTime": 0
    },
    "RetarderTorqueMode": 0,
    "AtLeastOnePtoEngaged": 0,
    "TachographVehicleSpeed": 0,
    "TransmissionCurrentGear": 0,
    "BOI03": "0x0F00000000000000",
    "ActualRetarderPercentTorque": 0,
    "ActualEnginePercentTorque": 9,
    "WheelBasedVehicleSpeed": 0,
    "TransID": 1,
    "OperatorKeySwitchIgnitionPower": 1,
    "TransmissionSelectedGear": 0,
    "MessageType": 10
  },
  "reasonData": {
    "start": {
      "reason": "START_TRIGGER",
      "timeStamp": 172204089999,
      "name": "RootStartConditionGroup",
      "triggerElements": [
        "triggerElement"
      ]
    }
  }
}

```

Preliminary

```

    }
  }
}

```

measurementDataReducedExportMessage

Message that is sent for every Measurement Job Result in REDUCED format.

ResultData contains individual signal/message level timestamps for Reduced export format.

MetaData section at the beginning containing vehicle and device information.

Each signal/message will contain name, value and timestamp while recorded on the device.

Header example - measurementDataReducedExportMessage

```

{
  "mjrId": "mjrUUID",
  "measurementConfigurationId": "measurementConfigurationUUID",
  "measurementConfigurationName": "measurementConfigurationName",
  "releasedMeasurementConfigurationVersion": 3,
  "testStepId": "testStepUUID",
  "tenant-id": "tenantId",
  "bootCycleId": "bootCycleUUID",
  "mjrCounter": 0,
  "message-type": "measurementDataExportMessage",
  "vin": "VIN123",
  "vehicleId": "vehicleUUID"
}

```

Payload example - measurementDataReducedExportMessage

REDUCED export format

```

{
  "metaData": {
    "vehicleId": "vehicleUUID",
    "deviceId": "deviceId123",
    "vin": "VIN123",
    "measurementConfigurationId": "measurementConfigurationUUID",
    "measurementConfigurationName": "measurementConfigurationName",
    "releasedMeasurementConfigurationVersion": 9,
    "bootCycleId": "bootCycleUUID",
    "testStepId": "testStepUUID",
    "measurementJobResultId": "measurementJobResultUUID",
    "start": 1722203900000,
    "end": 1722204104000,
    "packageIndex": 0,
    "lastPackage": false,
    "uploaded": 1722204110000,
    "exported": 1722204110000,
    "customMetadata": {
      "key": "value"
    }
  },
  "resultData": {
    "resultData1": [
      {
        "timeStamp": 1722204089999,
        "value": 30.000000000000004
      },
      {
        "timeStamp": 1722204099999,
        "value": 31.000000000000004
      }
    ],
    "resultData2": [
      {
        "timeStamp": 1722204089999,
        "value": [
          31
        ]
      }
    ]
  },
}

```

```

    {
      "timeStamp": 1722204089999,
      "value": [
        31
      ]
    }
  ],
  "resultData3": [
    {
      "timeStamp": 1722203990994,
      "value": [
        0,
        0,
        0,
        7,
        7,
        7
      ]
    },
    {
      "timeStamp": 1722204089999,
      "value": [
        0,
        0,
        3,
        7,
        7,
        7
      ]
    }
  ],
  "resultData4": {
    "timeStamp": 1722204089999,
    "value": [
      127
    ]
  },
  "resultData5": [
    {
      "timeStamp": 1722204089999,
      "value": [
        31
      ]
    },
    {
      "timeStamp": 1722204089999,
      "value": [
        31
      ]
    }
  ],
  "resultData6": [
    {
      "timeStamp": 1722204089999,
      "value": [
        46,
        47,
        48,
        49,
        50
      ]
    },
    {
      "timeStamp": 1722204089999,
      "value": [
        31,
        32,
        33,
        34,
        35,
        36
      ]
    }
  ],
  "reasonData": {
    "start": {
      "reason": "DRIVE_CYCLE_START",
      "timeStamp": 1722204089999
    }
  }
}

```

Preliminary

SUB ps.\$tenantId.diagnostics.results

Event that is sent for a Diagnostic Result (only DTC's at the moment).

Operation ID **subscribeDiagnosticsResults**

Accepts **dtcsSnapshotMessage** Event that is sent when a DTC occurs.

dtcsSnapshotMessage

Event that is sent when a DTC occurs.

Header example - dtcsSnapshotMessage

```
{
  "message-type": "dtcsSnapshotMessage"
}
```

Payload example - dtcsSnapshotMessage

```
{
  "dtcsSnapshotId": "string",
  "timestamp": "2019-08-24T14:15:22Z",
  "vehicleDiagnosticConfigurationId": "string",
  "vehicle": {
    "vehicleId": "string",
    "vehicleName": "string",
    "vin": "string",
    "manufacturerName": "string",
    "modelName": "string"
  },
  "dtcs": [
    {
      "dtcId": "string",
      "dtcIdDecoded": "string",
      "dtcStatus": "string",
      "milStatus": true,
      "ecu": {
        "ecuId": "string"
      },
      "dtcIdDescription": "string",
      "dtcStatusText": "string",
      "snapshots": [
        {
          "snapshotNumber": 0,
          "trace": {
            "traceId": "string",
            "timestamp": "2019-08-24T14:15:22Z",
            "request": "string",
            "response": "string",
            "serviceId": "string",
            "responseId": "string"
          },
          "parameters": [
            {
              "name": "string",
              "value": "string",
              "unit": "string"
            }
          ]
        }
      ]
    }
  ],
  "rawData": [
    {
      "request": "string",
      "response": "string"
    }
  ]
}
```

Preliminary

SUB ps.\$tenantId.vehicles.management.events

Event that is sent for updates related to the management of vehicles.

Operation ID **subscribeVehiclesManagementEvents**

Accepts **one** of the following messages:

- **deviceMappingExportMessage**
Event that is sent when a device is paired with a vehicle
- **vehicleCreationExportMessage**
Event that is sent when a new vehicle is created
- **vehicleUpdatedExportMessage**
Event that is sent when a vehicle is updated
- **deviceCreationExportMessage**
Event that is sent when a new device is created
- **deviceDeletionExportMessage**
Event that is sent when a device is removed
- **gpsStateChangedExportMessage**
Event that is sent when the GPS tracking is changed for a vehicle
- **deviceUnmappingExportMessage**
Event that is sent when a device is unmapped from a vehicle

deviceMappingExportMessage

Event that is sent when a device is paired with a vehicle.

Header example - deviceMappingExportMessage

```
{
  "message-type": "deviceMappingExportMessage",
  "deviceId": "string",
  "vehicleId": "string"
}
```

Payload example - deviceMappingExportMessage

```
{
  "vin": "string",
  "deviceIdentifier": "string",
  "deviceSerialNumber": "string",
  "correlations": {
    "deviceId": "string",
    "vehicleId": "string"
  }
}
```

vehicleCreationExportMessage

Event that is sent when a new vehicle is created.

Header example - vehicleCreationExportMessage

```
{
  "message-type": "vehicleCreationExportMessage",
  "vehicleId": "string"
}
```

Payload example - vehicleCreationExportMessage

```
{
  "vehicleName": "string",
  "vin": "string",
  "manufacturerName": "string",
  "modelName": "string",
  "description": "string",
  "geoPositioning": "ENABLED",
  "vehicleCustomAttributeList": [
    {
      "vehicleCustomAttributeId": "string",
      "name": "string",
      "internalName": "string",

```

```
    "assignedValue": "string"
  },
  "correlations": {
    "vehicleId": "string"
  }
}
```

vehicleUpdatedExportMessage

Event that is sent when a vehicle is updated.

Header example - vehicleUpdatedExportMessage

```
{
  "message-type": "vehicleUpdatedExportMessage",
  "vehicleId": "string"
}
```

Payload example - vehicleUpdatedExportMessage

```
{
  "vehicleName": "string",
  "vin": "string",
  "manufacturerName": "string",
  "modelName": "string",
  "description": "string",
  "geoPositioning": "ENABLED",
  "vehicleCustomAttributeList": [
    {
      "vehicleCustomAttributeId": "string",
      "name": "string",
      "internalName": "string",
      "assignedValue": "string"
    }
  ],
  "correlations": {
    "vehicleId": "string"
  }
}
```

deviceCreationExportMessage

Event that is sent when a new device is created.

Header example - deviceCreationExportMessage

```
{
  "message-type": "deviceCreationExportMessage",
  "deviceId": "string"
}
```

Payload example - deviceCreationExportMessage

```
{
  "deviceIdentifier": "string",
  "deviceSerialNumber": "string",
  "iccId": "string",
  "correlations": {
    "deviceId": "string"
  }
}
```

deviceDeletionExportMessage

Event that is sent when a device is removed.

Header example - deviceDeletionExportMessage

```
{
  "message-type": "deviceDeletionExportMessage",
  "deviceId": "string"
}
```

Payload example - deviceDeletionExportMessage

```
{
  "deviceIdentifier": "string",
  "deviceSerialNumber": "string",
  "correlations": {
    "deviceId": "string"
  }
}
```

gpsStateChangedExportMessage

Event that is sent when the GPS tracking is changed for a vehicle.

Header example - gpsStateChangedExportMessage

```
{
  "message-type": "gpsStateChangedExportMessage",
  "deviceId": "string",
  "vehicleId": "string"
}
```

Payload example - gpsStateChangedExportMessage

```
{
  "vin": "string",
  "deviceIdentifier": "string",
  "deviceSerialNumber": "string",
  "geoPositioningState": "ENABLED,",
  "correlations": {
    "deviceId": "string",
    "vehicleId": "string"
  }
}
```

deviceUnmappingExportMessage

Event that is sent when a device is unmapped from a vehicle.

Header example - deviceUnmappingExportMessage

```
{
  "message-type": "deviceUnmappingExportMessage",
  "deviceId": "string",
  "vehicleId": "string"
}
```

Payload example - deviceUnmappingExportMessage

```
{
  "vin": "string",
  "deviceIdentifier": "string",
  "deviceSerialNumber": "string",
  "correlations": {
    "deviceId": "string",
    "vehicleId": "string"
  }
}
```

11.2.4 OTA Function Calls

OTA Function Calls Service 1.0.0

Kafka topics provided by the OTA Function Calls (OFC) service.

This API allows you to receive events that occur in the OTA Function Calls service.

- [Operations](#)
 - [SUB ps.\\$tenantId.ota-function-calls.events.function-call-status](#)
 - [Function call status update](#)
 - [Payload example - Function call status update](#)

Operations

SUB ps.\$tenantId.ota-function-calls.events.function-call-status

Receive function call status update.

Accepts the following message: Function call status update

Function call status update

This event occurs when the service receives status update for previously successfully created function call.

Payload example - Function call status update

```
{
  "deviceId": "Required - string
    The deviceId of the device.",
  "vehicleId": "string
    The vehicleId of the vehicle.",
  "vin": "string
    The VIN of the vehicle.",
  "callId": "Required - string
    callId is the value returned upon successfully created function call that can be used to match status
    update event to a specific function call.",
  "name": "Required - string
    comfort.seats.current_position",
  "status": "Required - string
    Allowed values: "PENDING""DELIVERED""UNKNOWN""SUCCESS""FAILURE".",
  "inArguments": "string
    Input argument, defined as JSON string.
    \\\\\"{\\\\"row\\\\\\":0,\\\\"index\\\\\\":0}\\\\"\\",
  "outArguments": "string
    Output argument, defined as JSON string.
    \\\\\"{\\\\"seat\\\\\\":{\\\\"location\\\\\\":{\\\\"row\\\\\\":0,\\\\"index\\\\\\":0},\\\\"position\\\\\\":{\\\\"base\\\\\\":500,\\\\"cushion\\\\\\":500,\\\\"lumbar\\\\\\":0,\\\\"side_booster\\\\\\":0,\\\\"head_restraint\\\\\\":0}}\\\\"\\",
}
```

Preliminary

11.2.5 OTA Updates Campaign Exporter

Campaign Kafka Exporter API 1.0.0

The Campaign Exporter API allows you to consume updates from Campaigns in an event-driven way. The Campaign Service actively pushes notifications out, if they are available, allowing consumers to handle events in a push-based way.

This is in contrast to the REST API where the consumers of the API have to actively poll to notice any changes.

- [Operations](#)
 - [SUB ps.\\$tenantId.campaigns.events](#)
 - [campaignCreatedEvent](#)
 - [Header example](#)
 - [Payload example](#)
 - [campaignStateUpdatedEvent](#)
 - [Header example](#)

- Payload example
- campaignPropertyUpdatedEvent
 - Header example
 - Payload example
- campaignFilterEvaluationStateChangedEvent
 - Header example
 - Payload example
- campaignDeletedEvent
 - Header example
 - Payload example
- campaignGroupCreatedEvent
 - Header example
 - Payload example
- campaignGroupStateChangedEvent
 - Header example
 - Payload example
- campaignGroupDeletedEvent
 - Header example
 - Payload example
- campaignTargetAddedEvent
 - Header example
 - Payload example
- campaignTargetDeletedEvent
 - Header example
 - Payload example
- campaignTargetAssignmentCreatedEvent
 - Header example
 - Payload example
- campaignTargetAssignmentFinishedEvent
 - Header example
 - Payload example
- campaignTargetAssignmentStateUpdatedEvent
 - Header example
 - Payload example
- campaignTargetsAddedEvent
 - Header example
 - Payload example
- campaignTargetsDeletedEvent
 - Header example
 - Payload example
- campaignTargetGroupAssignmentChangedEvent
 - Header example
 - Payload example
- campaignTargetAssignmentRetriedEvent
 - Header example
 - Payload example
- campaignTargetAssignmentReusedEvent
 - Header example
 - Payload example

Operations

SUB ps.\$tenantId.campaigns.events

Event that is sent when a Campaign has been updated

Operation ID: **subscribeCampaignUpdates**

Accepts **one of** the following messages:

- `campaignCreatedEvent`
- `campaignStateUpdatedEvent`
- `campaignPropertyUpdatedEvent`
- `campaignFilterEvaluationStateChangedEvent`
- `campaignDeletedEvent`
- `campaignGroupCreatedEvent`
- `campaignGroupStateChangedEvent`
- `campaignGroupDeletedEvent`
- `campaignTargetAddedEvent`
- `campaignTargetDeletedEvent`
- `campaignTargetAssignmentCreatedEvent`
- `campaignTargetAssignmentFinishedEvent`
- `campaignTargetAssignmentStateUpdatedEvent`
- `campaignTargetsAddedEvent`
- `campaignTargetsDeletedEvent`
- `campaignTargetGroupAssignmentChangedEvent`
- `campaignTargetAssignmentRetriedEvent`
- `campaignTargetAssignmentReusedEvent`

campaignCreatedEvent

Event that is sent when a campaign has been created.

Header example

```
{
  "message-type": "campaignCreatedEvent",
  "distributionPackage": "string",
  "baseline": "string",
  "remoteDiagnosticConfiguration": "string",
  "remoteMeasurementConfiguration": "string"
}
```

Payload example

```
{
  "campaign": {
    "actualEnd": "2019-08-24T14:15:22Z",
    "actualStart": "2019-08-24T14:15:22Z",
    "assignableItem": {
      "id": "string",
      "version": 0,
      "usageContext": "FILTER"
    },
    "assignmentsFailureThreshold": 100,
    "autoSchedule": false,
    "campaignId": "string",
    "createdOn": "2019-08-24T14:15:22Z",
    "deleted": true,
    "description": "string",
    "devicePairing": {
      "deviceSlot": "string"
    },
    "dynamic": true,
    "eventCount": 0,
    "filter": {
      "vehicle": "string",
      "device": "string",
      "inventory": "string",
      "inventoryCompatibility": [
        "string"
      ],
      "capabilities": [
        "string"
      ]
    }
  }
}
```

```

    },
    "filterEvaluationState": "EVALUATING",
    "groupCount": 0,
    "hasGroups": true,
    "lastUpdatedOn": "2019-08-24T14:15:22Z",
    "maxRunningAssignments": 100,
    "name": "string",
    "plannedEnd": "2019-08-24T14:15:22Z",
    "plannedStart": "2019-08-24T14:15:22Z",
    "privileged": true,
    "state": "CREATED",
    "stateChangedOn": "2019-08-24T14:15:22Z",
    "stateUpdateReason": "string",
    "targetOptions": {
      "matchingBehaviour": {
        "revokeOnUnmatch": false,
        "restartOnRematch": false
      },
      "assignmentOptions": {
        "timeouts": {
          "totalExpiry": "P30D",
          "signoffExpiry": "P2D",
          "managerApprovalExpiry": "P1D"
        },
        "behaviour": {
          "requireSignoff": false,
          "requireManagerApproval": false
        }
      }
    },
    "targetType": "VEHICLE",
    "type": "OTA_ASSIGNMENT",
    "targetCount": 0,
    "vehicleCount": 0,
    "targetSource": "FILTER",
    "assignmentCreation": {
      "limits": {
        "amount": 1,
        "slidingWindowDuration": "string",
        "amountCreated": 0,
        "rateLimitedUntil": "2019-08-24T14:15:22Z"
      },
      "triggerConditions": {
        "vehicle": "string",
        "inventory": "string",
        "inventoryCompatibility": [
          "string"
        ],
        "capabilities": [
          "string"
        ]
      }
    },
    "correlations": [
      {
        "correlationId": "string",
        "type": "string"
      }
    ],
    "priority": 100
  },
  "correlations": {
    "distributionPackage": "string",
    "baseline": "string",
    "remoteDiagnosticConfiguration": "string",
    "remoteMeasurementConfiguration": "string"
  }
}

```

This example has been generated automatically.

```

{
  "message-type": "campaignCreatedEvent",
  "distributionPackage": "string",
  "baseline": "string",
  "remoteDiagnosticConfiguration": "string",
  "remoteMeasurementConfiguration": "string"
}

```

campaignStateUpdatedEvent

Event that is sent when the state of a campaign has been changed.

Header example

Preliminary

```
{
  "message-type": "campaignStateUpdatedEvent",
  "distributionPackage": "string",
  "baseline": "string",
  "remoteDiagnosticConfiguration": "string",
  "remoteMeasurementConfiguration": "string"
}
```

Payload example

```
{
  "campaign": {
    "actualEnd": "2019-08-24T14:15:22Z",
    "actualStart": "2019-08-24T14:15:22Z",
    "assignableItem": {
      "id": "string",
      "version": 0,
      "usageContext": "FILTER"
    },
    "assignmentsFailureThreshold": 100,
    "autoSchedule": false,
    "campaignId": "string",
    "createdOn": "2019-08-24T14:15:22Z",
    "deleted": true,
    "description": "string",
    "devicePairing": {
      "deviceSlot": "string"
    },
    "dynamic": true,
    "eventCount": 0,
    "filter": {
      "vehicle": "string",
      "device": "string",
      "inventory": "string",
      "inventoryCompatibility": [
        "string"
      ],
      "capabilities": [
        "string"
      ]
    },
    "filterEvaluationState": "EVALUATING",
    "groupCount": 0,
    "hasGroups": true,
    "lastUpdatedOn": "2019-08-24T14:15:22Z",
    "maxRunningAssignments": 100,
    "name": "string",
    "plannedEnd": "2019-08-24T14:15:22Z",
    "plannedStart": "2019-08-24T14:15:22Z",
    "privileged": true,
    "state": "CREATED",
    "stateChangedOn": "2019-08-24T14:15:22Z",
    "stateUpdateReason": "string",
    "targetOptions": {
      "matchingBehaviour": {
        "revokeOnUnmatch": false,
        "restartOnRematch": false
      },
      "assignmentOptions": {
        "timeouts": {
          "totalExpiry": "P30D",
          "signoffExpiry": "P2D",
          "managerApprovalExpiry": "P1D"
        },
        "behaviour": {
          "requiresSignoff": false,
          "requireManagerApproval": false
        }
      }
    },
    "targetType": "VEHICLE",
    "type": "OTA_ASSIGNMENT",
    "targetCount": 0,
    "vehicleCount": 0,
    "targetSource": "FILTER",
    "assignmentCreation": {
      "limits": {
        "amount": 1,
        "slidingWindowDuration": "string",
        "amountCreated": 0,
        "rateLimitedUntil": "2019-08-24T14:15:22Z"
      },
      "triggerConditions": {
        "vehicle": "string",
        "inventory": "string",

```

Preliminary

```

        "inventoryCompatibility": [
            "string"
        ],
        "capabilities": [
            "string"
        ]
    },
    "correlations": [
        {
            "correlationId": "string",
            "type": "string"
        }
    ],
    "priority": 100
},
"correlations": {
    "distributionPackage": "string",
    "baseline": "string",
    "remoteDiagnosticConfiguration": "string",
    "remoteMeasurementConfiguration": "string"
}
}

```

campaignPropertyUpdatedEvent

Event that is sent when the property of a campaign has been changed.

Header example

```

{
  "message-type": "campaignPropertyUpdatedEvent",
  "distributionPackage": "string",
  "baseline": "string",
  "remoteDiagnosticConfiguration": "string",
  "remoteMeasurementConfiguration": "string"
}

```

Payload example

```

{
  "campaign": {
    "actualEnd": "2019-08-24T14:15:22Z",
    "actualStart": "2019-08-24T14:15:22Z",
    "assignableItem": {
      "id": "string",
      "version": 0,
      "usageContext": "FILTER"
    },
    "assignmentsFailureThreshold": 100,
    "autoSchedule": false,
    "campaignId": "string",
    "createdOn": "2019-08-24T14:15:22Z",
    "deleted": true,
    "description": "string",
    "devicePairing": {
      "deviceSlot": "string"
    },
    "dynamic": true,
    "eventCount": 0,
    "filter": {
      "vehicle": "string",
      "device": "string",
      "inventory": "string",
      "inventoryCompatibility": [
        "string"
      ],
      "capabilities": [
        "string"
      ]
    },
    "filterEvaluationState": "EVALUATING",
    "groupCount": 0,
    "hasGroups": true,
    "lastUpdatedOn": "2019-08-24T14:15:22Z",
    "maxRunningAssignments": 100,
    "name": "string",
    "plannedEnd": "2019-08-24T14:15:22Z",
    "plannedStart": "2019-08-24T14:15:22Z",

```

```

    "privileged": true,
    "state": "CREATED",
    "stateChangedOn": "2019-08-24T14:15:22Z",
    "stateUpdateReason": "string",
    "targetOptions": {
      "matchingBehaviour": {
        "revokeOnUnmatch": false,
        "restartOnRematch": false
      },
      "assignmentOptions": {
        "timeouts": {
          "totalExpiry": "P30D",
          "signoffExpiry": "P2D",
          "managerApprovalExpiry": "P1D"
        },
        "behaviour": {
          "requireSignoff": false,
          "requireManagerApproval": false
        }
      }
    },
    "targetType": "VEHICLE",
    "type": "OTA_ASSIGNMENT",
    "targetCount": 0,
    "vehicleCount": 0,
    "targetSource": "FILTER",
    "assignmentCreation": {
      "limits": {
        "amount": 1,
        "slidingWindowDuration": "string",
        "amountCreated": 0,
        "rateLimitedUntil": "2019-08-24T14:15:22Z"
      },
      "triggerConditions": {
        "vehicle": "string",
        "inventory": "string",
        "inventoryCompatibility": [
          "string"
        ],
        "capabilities": [
          "string"
        ]
      }
    },
    "correlations": [
      {
        "correlationId": "string",
        "type": "string"
      }
    ],
    "priority": 100
  },
  "correlations": {
    "distributionPackage": "string",
    "baseline": "string",
    "remoteDiagnosticConfiguration": "string",
    "remoteMeasurementConfiguration": "string"
  }
}

```

campaignFilterEvaluationStateChangedEvent

Event that is sent when the filter evaluation state of a campaign has been changed.

Header example

```

{
  "message-type": "campaignFilterEvaluationStateChangedEvent",
  "distributionPackage": "string",
  "baseline": "string",
  "remoteDiagnosticConfiguration": "string",
  "remoteMeasurementConfiguration": "string"
}

```

Payload example

```

{
  "campaign": {
    "actualEnd": "2019-08-24T14:15:22Z",

```

```

"actualStart": "2019-08-24T14:15:22Z",
"assignableItem": {
  "id": "string",
  "version": 0,
  "usageContext": "FILTER"
},
"assignmentsFailureThreshold": 100,
"autoSchedule": false,
"campaignId": "string",
"createdOn": "2019-08-24T14:15:22Z",
"deleted": true,
"description": "string",
"devicePairing": {
  "deviceSlot": "string"
},
"dynamic": true,
"eventCount": 0,
"filter": {
  "vehicle": "string",
  "device": "string",
  "inventory": "string",
  "inventoryCompatibility": [
    "string"
  ],
  "capabilities": [
    "string"
  ]
},
"filterEvaluationState": "EVALUATING",
"groupCount": 0,
"hasGroups": true,
"lastUpdatedOn": "2019-08-24T14:15:22Z",
"maxRunningAssignments": 100,
"name": "string",
"plannedEnd": "2019-08-24T14:15:22Z",
"plannedStart": "2019-08-24T14:15:22Z",
"privileged": true,
"state": "CREATED",
"stateChangedOn": "2019-08-24T14:15:22Z",
"stateUpdateReason": "string",
"targetOptions": {
  "matchingBehaviour": {
    "revokeOnUnmatch": false,
    "restartOnRematch": false
  },
  "assignmentOptions": {
    "timeouts": {
      "totalExpiry": "P30D",
      "signoffExpiry": "P2D",
      "managerApprovalExpiry": "P1D"
    },
    "behaviour": {
      "requireSignoff": false,
      "requireManagerApproval": false
    }
  }
},
"targetType": "VEHICLE",
"type": "OTA_ASSIGNMENT",
"targetCount": 0,
"vehicleCount": 0,
"targetSource": "FILTER",
"assignmentCreation": {
  "limits": {
    "amount": 1,
    "slidingWindowDuration": "string",
    "amountCreated": 0,
    "rateLimitedUntil": "2019-08-24T14:15:22Z"
  },
  "triggerConditions": {
    "vehicle": "string",
    "inventory": "string",
    "inventoryCompatibility": [
      "string"
    ],
    "capabilities": [
      "string"
    ]
  }
},
"correlations": [
  {
    "correlationId": "string",
    "type": "string"
  }
],
"priority": 100
},
"correlations": {

```

```
"distributionPackage": "string",
"baseline": "string",
"remoteDiagnosticConfiguration": "string",
"remoteMeasurementConfiguration": "string"
}
```

campaignDeletedEvent

Event that is sent when a campaign has been deleted.

Header example

```
{
  "message-type": "campaignDeletedEvent",
  "distributionPackage": "string",
  "baseline": "string",
  "remoteDiagnosticConfiguration": "string",
  "remoteMeasurementConfiguration": "string"
}
```

Payload example

```
{
  "campaign": {
    "actualEnd": "2019-08-24T14:15:22Z",
    "actualStart": "2019-08-24T14:15:22Z",
    "assignableItem": {
      "id": "string",
      "version": 0,
      "usageContext": "FILTER"
    },
    "assignmentsFailureThreshold": 100,
    "autoSchedule": false,
    "campaignId": "string",
    "createdOn": "2019-08-24T14:15:22Z",
    "deleted": true,
    "description": "string",
    "devicePairing": {
      "deviceSlot": "string"
    },
    "dynamic": true,
    "eventCount": 0,
    "filter": {
      "vehicle": "string",
      "device": "string",
      "inventory": "string",
      "inventoryCompatibility": [
        "string"
      ],
      "capabilities": [
        "string"
      ]
    },
    "filterEvaluationState": "EVALUATING",
    "groupCount": 0,
    "hasGroups": true,
    "lastUpdatedOn": "2019-08-24T14:15:22Z",
    "maxRunningAssignments": 100,
    "name": "string",
    "plannedEnd": "2019-08-24T14:15:22Z",
    "plannedStart": "2019-08-24T14:15:22Z",
    "privileged": true,
    "state": "CREATED",
    "stateChangedOn": "2019-08-24T14:15:22Z",
    "stateUpdateReason": "string",
    "targetOptions": {
      "matchingBehaviour": {
        "revokeOnUnmatch": false,
        "restartOnRematch": false
      },
      "assignmentOptions": {
        "timeouts": {
          "totalExpiry": "P30D",
          "signoffExpiry": "P2D",
          "managerApprovalExpiry": "P1D"
        },
        "behaviour": {
          "requireSignoff": false,
          "requireManagerApproval": false
        }
      }
    }
  }
}
```

Preliminary

```

    },
    "targetType": "VEHICLE",
    "type": "OTA_ASSIGNMENT",
    "targetCount": 0,
    "vehicleCount": 0,
    "targetSource": "FILTER",
    "assignmentCreation": {
      "limits": {
        "amount": 1,
        "slidingWindowDuration": "string",
        "amountCreated": 0,
        "rateLimitedUntil": "2019-08-24T14:15:22Z"
      },
      "triggerConditions": {
        "vehicle": "string",
        "inventory": "string",
        "inventoryCompatibility": [
          "string"
        ],
        "capabilities": [
          "string"
        ]
      }
    },
    "correlations": [
      {
        "correlationId": "string",
        "type": "string"
      }
    ],
    "priority": 100
  },
  "correlations": {
    "distributionPackage": "string",
    "baseline": "string",
    "remoteDiagnosticConfiguration": "string",
    "remoteMeasurementConfiguration": "string"
  }
}

```

campaignGroupCreatedEvent

Event that is sent when a campaign group has been created.

Header example

```

{
  "message-type": "campaignGroupCreatedEvent"
}

```

Payload example

```

{
  "campaignId": "string",
  "campaignGroup": {
    "name": "string",
    "successThreshold": 0,
    "currentSuccessRatio": 0,
    "filter": {
      "vehicle": "string",
      "inventory": "string"
    },
    "vehicleIds": [
      "string"
    ],
    "state": "NOT_STARTED",
    "thresholdState": "SUCCESS",
    "vehicleCount": 0,
    "targetCount": 0
  },
  "correlations": {
    "distributionPackage": "string",
    "baseline": "string",
    "remoteDiagnosticConfiguration": "string",
    "remoteMeasurementConfiguration": "string"
  }
}

```

campaignGroupStateChangedEvent

Event that is sent when a campaign group's state has been changed.

Header example

```
{
  "message-type": "campaignGroupStateChangedEvent"
}
```

Payload example

```
{
  "campaignId": "string",
  "campaignGroup": {
    "name": "string",
    "successThreshold": 0,
    "currentSuccessRatio": 0,
    "filter": {
      "vehicle": "string",
      "inventory": "string"
    },
    "vehicleIds": [
      "string"
    ],
    "state": "NOT_STARTED",
    "thresholdState": "SUCCESS",
    "vehicleCount": 0,
    "targetCount": 0
  },
  "correlations": {
    "distributionPackage": "string",
    "baseline": "string",
    "remoteDiagnosticConfiguration": "string",
    "remoteMeasurementConfiguration": "string"
  }
}
```

campaignGroupDeletedEvent

Event that is sent when a campaign group has been deleted.

Header example

```
{
  "message-type": "campaignGroupDeletedEvent"
}
```

Payload example

```
{
  "campaignId": "string",
  "campaignGroup": {
    "name": "string",
    "successThreshold": 0,
    "currentSuccessRatio": 0,
    "filter": {
      "vehicle": "string",
      "inventory": "string"
    },
    "vehicleIds": [
      "string"
    ],
    "state": "NOT_STARTED",
    "thresholdState": "SUCCESS",
    "vehicleCount": 0,
    "targetCount": 0
  },
  "correlations": {
    "distributionPackage": "string",
```

```
"baseline": "string",
"remoteDiagnosticConfiguration": "string",
"remoteMeasurementConfiguration": "string"
}
```

campaignTargetAddedEvent

Event that is sent when a campaign target has been added.

Header example

```
{
  "message-type": "campaignTargetAddedEvent",
  "vehicle": "string",
  "device": "string",
  "assignment": "string"
}
```

Payload example

```
{
  "campaignTarget": {
    "campaignId": "string",
    "target": {
      "id": "string"
    },
    "assignedGroup": {
      "index": 0
    },
    "assignment": {
      "id": "string",
      "state": "string",
      "retries": {
        "totalCount": 0,
        "toBeRetried": false,
        "lastAttempted": "2019-08-24T14:15:22Z"
      },
      "userActionRequired": true,
      "failureReason": "string"
    },
    "assignmentSummary": "SUCCESSFUL",
    "matchState": "MATCHING",
    "waitingForAssignmentCreation": true,
    "assignmentHistory": [
      {
        "id": "string",
        "state": "string",
        "retries": {
          "totalCount": 0,
          "toBeRetried": false,
          "lastAttempted": "2019-08-24T14:15:22Z"
        },
        "userActionRequired": true,
        "failureReason": "string"
      }
    ]
  },
  "correlations": {
    "vehicle": "string",
    "device": "string",
    "assignment": "string"
  }
}
```

Preliminary

campaignTargetDeletedEvent

Event that is sent when a campaign target has been deleted.

Header example

```
{
  "message-type": "campaignTargetDeletedEvent",
  "vehicle": "string",
  "device": "string",
  "assignment": "string"
}
```

Payload example

```
{
  "campaignTarget": {
    "campaignId": "string",
    "target": {
      "id": "string"
    },
    "assignedGroup": {
      "index": 0
    },
    "assignment": {
      "id": "string",
      "state": "string",
      "retries": {
        "totalCount": 0,
        "toBeRetried": false,
        "lastAttempted": "2019-08-24T14:15:22Z"
      },
      "userActionRequired": true,
      "failureReason": "string"
    },
    "assignmentSummary": "SUCCESSFUL",
    "matchState": "MATCHING",
    "waitingForAssignmentCreation": true,
    "assignmentHistory": [
      {
        "id": "string",
        "state": "string",
        "retries": {
          "totalCount": 0,
          "toBeRetried": false,
          "lastAttempted": "2019-08-24T14:15:22Z"
        },
        "userActionRequired": true,
        "failureReason": "string"
      }
    ]
  },
  "correlations": {
    "vehicle": "string",
    "device": "string",
    "assignment": "string"
  }
}
```

This example has been generated automatically.

```
{
  "message-type": "campaignTargetDeletedEvent",
  "vehicle": "string",
  "device": "string",
  "assignment": "string"
}
```

campaignTargetAssignmentCreatedEvent

Event that is sent when an assignment for the campaign target has been created.

Header example

```
{
  "message-type": "campaignTargetAssignmentCreatedEvent",
  "vehicle": "string",
  "device": "string",
  "assignment": "string"
}
```

Payload example

```
{
  "campaignTarget": {
    "campaignId": "string",
    "target": {
      "id": "string"
    },
    "assignedGroup": {
      "index": 0
    },
    "assignment": {
      "id": "string",
      "state": "string",
      "retries": {
        "totalCount": 0,
        "toBeRetried": false,
        "lastAttempted": "2019-08-24T14:15:22Z"
      },
      "userActionRequired": true,
      "failureReason": "string"
    },
    "assignmentSummary": "SUCCESSFUL",
    "matchState": "MATCHING",
    "waitingForAssignmentCreation": true,
    "assignmentHistory": [
      {
        "id": "string",
        "state": "string",
        "retries": {
          "totalCount": 0,
          "toBeRetried": false,
          "lastAttempted": "2019-08-24T14:15:22Z"
        },
        "userActionRequired": true,
        "failureReason": "string"
      }
    ]
  },
  "correlations": {
    "vehicle": "string",
    "device": "string",
    "assignment": "string"
  }
}
```

Preliminary

campaignTargetAssignmentFinishedEvent

Event that is sent when an assignment for the campaign target has been finished.

Header example

```
{
  "message-type": "campaignTargetAssignmentFinishedEvent",
  "vehicle": "string",
  "device": "string",
  "assignment": "string"
}
```

Payload example

```
{
  "campaignTarget": {
    "campaignId": "string",
    "target": {
      "id": "string"
    },
    "assignedGroup": {
      "index": 0
    },
    "assignment": {
      "id": "string",
      "state": "string",
      "retries": {
        "totalCount": 0,

```

```

        "toBeRetried": false,
        "lastAttempted": "2019-08-24T14:15:22Z"
      },
      "userActionRequired": true,
      "failureReason": "string"
    },
    "assignmentSummary": "SUCCESSFUL",
    "matchState": "MATCHING",
    "waitingForAssignmentCreation": true,
    "assignmentHistory": [
      {
        "id": "string",
        "state": "string",
        "retries": {
          "totalCount": 0,
          "toBeRetried": false,
          "lastAttempted": "2019-08-24T14:15:22Z"
        },
        "userActionRequired": true,
        "failureReason": "string"
      }
    ]
  },
  "correlations": {
    "vehicle": "string",
    "device": "string",
    "assignment": "string"
  }
}

```

campaignTargetAssignmentStateUpdatedEvent

Event that is sent when the assignment for the campaign target has been updated.

Header example

```

{
  "message-type": "campaignTargetAssignmentStateUpdatedEvent",
  "vehicle": "string",
  "device": "string",
  "assignment": "string"
}

```

Payload example

```

{
  "campaignTarget": {
    "campaignId": "string",
    "target": {
      "id": "string"
    },
    "assignedGroup": {
      "index": 0
    },
    "assignment": {
      "id": "string",
      "state": "string",
      "retries": {
        "totalCount": 0,
        "toBeRetried": false,
        "lastAttempted": "2019-08-24T14:15:22Z"
      },
      "userActionRequired": true,
      "failureReason": "string"
    },
    "assignmentSummary": "SUCCESSFUL",
    "matchState": "MATCHING",
    "waitingForAssignmentCreation": true,
    "assignmentHistory": [
      {
        "id": "string",
        "state": "string",
        "retries": {
          "totalCount": 0,
          "toBeRetried": false,
          "lastAttempted": "2019-08-24T14:15:22Z"
        },
      }
    ]
  },
}

```

```

        "userActionRequired": true,
        "failureReason": "string"
    }
  ],
  "correlations": {
    "vehicle": "string",
    "device": "string",
    "assignment": "string"
  }
}

```

campaignTargetsAddedEvent

Event that is sent when a list of campaign targets has been added.

Header example

```

{
  "message-type": "campaignTargetsAddedEvent",
  "vehicle": "string",
  "device": "string",
  "assignment": "string"
}

```

Payload example

```

{
  "campaign": {
    "campaignId": "string"
  },
  "campaignTargets": [
    {
      "campaignId": "string",
      "target": {
        "id": "string"
      },
      "assignedGroup": {
        "index": 0
      },
      "assignment": {
        "id": "string",
        "state": "string",
        "retries": {
          "totalCount": 0,
          "toBeRetried": false,
          "lastAttempted": "2019-08-24T14:15:22Z"
        }
      },
      "userActionRequired": true,
      "failureReason": "string"
    },
    "assignmentSummary": "SUCCESSFUL",
    "matchState": "MATCHING",
    "waitingForAssignmentCreation": true,
    "assignmentHistory": [
      {
        "id": "string",
        "state": "string",
        "retries": {
          "totalCount": 0,
          "toBeRetried": false,
          "lastAttempted": "2019-08-24T14:15:22Z"
        }
      },
      "userActionRequired": true,
      "failureReason": "string"
    }
  ]
}

```

campaignTargetsDeletedEvent

Event that is sent when a list of campaign targets has been deleted.

Header example

```
{
  "message-type": "campaignTargetsAddedEvent",
  "vehicle": "string",
  "device": "string",
  "assignment": "string"
}
```

Payload example

```
{
  "campaign": {
    "campaignId": "string"
  },
  "campaignTargets": [
    {
      "campaignId": "string",
      "target": {
        "id": "string"
      },
      "assignedGroup": {
        "index": 0
      },
      "assignment": {
        "id": "string",
        "state": "string",
        "retries": {
          "totalCount": 0,
          "toBeRetried": false,
          "lastAttempted": "2019-08-24T14:15:22Z"
        },
        "userActionRequired": true,
        "failureReason": "string"
      },
      "assignmentSummary": "SUCCESSFUL",
      "matchState": "MATCHING",
      "waitingForAssignmentCreation": true,
      "assignmentHistory": [
        {
          "id": "string",
          "state": "string",
          "retries": {
            "totalCount": 0,
            "toBeRetried": false,
            "lastAttempted": "2019-08-24T14:15:22Z"
          },
          "userActionRequired": true,
          "failureReason": "string"
        }
      ]
    }
  ]
}
```

Preliminary

campaignTargetGroupAssignmentChangedEvent

Event that is sent when the group assignment of a campaign target has been changed.

Header example

```
{
  "message-type": "campaignTargetGroupAssignmentChangedEvent",
  "vehicle": "string",
  "device": "string",
  "assignment": "string"
}
```

Payload example

```
{
  "campaignTarget": {
    "campaignId": "string",
    "target": {
      "id": "string"
    },
    "assignedGroup": {
      "index": 0
    },
    "assignment": {
      "id": "string",
      "state": "string",
      "retries": {
        "totalCount": 0,
        "toBeRetried": false,
        "lastAttempted": "2019-08-24T14:15:22Z"
      },
      "userActionRequired": true,
      "failureReason": "string"
    },
    "assignmentSummary": "SUCCESSFUL",
    "matchState": "MATCHING",
    "waitingForAssignmentCreation": true,
    "assignmentHistory": [
      {
        "id": "string",
        "state": "string",
        "retries": {
          "totalCount": 0,
          "toBeRetried": false,
          "lastAttempted": "2019-08-24T14:15:22Z"
        },
        "userActionRequired": true,
        "failureReason": "string"
      }
    ],
    "correlations": {
      "vehicle": "string",
      "device": "string",
      "assignment": "string"
    }
  }
}
```

Preliminary

campaignTargetAssignmentRetriedEvent

Event that is sent when the assignment of a Campaign Target has been retried.

Header example

```
{
  "message-type": "campaignTargetAssignmentRetriedEvent",
  "vehicle": "string",
  "device": "string",
  "assignment": "string"
}
```

Payload example

```
{
  "campaignTarget": {
    "campaignId": "string",
    "target": {
      "id": "string"
    },
    "assignedGroup": {
      "index": 0
    },
    "assignment": {
      "id": "string",
      "state": "string",
      "retries": {
        "totalCount": 0,
```

```

        "toBeRetried": false,
        "lastAttempted": "2019-08-24T14:15:22Z"
      },
      "userActionRequired": true,
      "failureReason": "string"
    },
    "assignmentSummary": "SUCCESSFUL",
    "matchState": "MATCHING",
    "waitingForAssignmentCreation": true,
    "assignmentHistory": [
      {
        "id": "string",
        "state": "string",
        "retries": {
          "totalCount": 0,
          "toBeRetried": false,
          "lastAttempted": "2019-08-24T14:15:22Z"
        },
        "userActionRequired": true,
        "failureReason": "string"
      }
    ]
  },
  "correlations": {
    "vehicle": "string",
    "device": "string",
    "assignment": "string"
  }
}

```

campaignTargetAssignmentReusedEvent

Event that is sent when the assignment of a Campaign Target has been reused.

Header example

```

{
  "message-type": "campaignTargetAssignmentReusedEvent",
  "vehicle": "string",
  "device": "string",
  "assignment": "string"
}

```

Payload example

```

{
  "campaignTarget": {
    "campaignId": "string",
    "target": {
      "id": "string"
    },
    "assignedGroup": {
      "index": 0
    },
    "assignment": {
      "id": "string",
      "state": "string",
      "retries": {
        "totalCount": 0,
        "toBeRetried": false,
        "lastAttempted": "2019-08-24T14:15:22Z"
      },
      "userActionRequired": true,
      "failureReason": "string"
    },
    "assignmentSummary": "SUCCESSFUL",
    "matchState": "MATCHING",
    "waitingForAssignmentCreation": true,
    "assignmentHistory": [
      {
        "id": "string",
        "state": "string",
        "retries": {
          "totalCount": 0,
          "toBeRetried": false,
          "lastAttempted": "2019-08-24T14:15:22Z"
        },

```

```

        "userActionRequired": true,
        "failureReason": "string"
    }
  ],
  "correlations": {
    "vehicle": "string",
    "device": "string",
    "assignment": "string"
  }
}

```

11.2.6 OTA Updates Distribution Exporter

OTA Updates Kafka Exporter API 1.0.0

The OTA Assignments Exporter API allows you to consume events that happen on behalf of the OTA Updates.

This is in contrast to the REST API where the consumers of the API have to actively poll to notice any changes.

- [Operations](#)
 - [SUB ps.\\$tenantId.ota-updates.events](#)
 - [assignmentCreatedEvent](#)
 - [Header example](#)
 - [Payload example](#)
 - [assignmentStatusUpdatedEvent](#)
 - [Header example](#)
 - [Payload example](#)
 - [assignmentCorrelatedWithLogEvent](#)
 - [Header example](#)
 - [Payload example](#)
 - [assignmentSoftwareNodeVerifiedEvent](#)
 - [Header example](#)
 - [Payload example](#)
 - [assignmentUpdateConditionFulfilledEvent](#)
 - [Header example](#)
 - [Payload example](#)
 - [assignmentDeletedEvent](#)
 - [Header example](#)
 - [Payload example](#)
 - [SUB ps.\\$tenantId.desired-states.events](#)
 - [desiredStateAssignmentCreatedEvent](#)
 - [Header example](#)
 - [Payload example](#)
 - [desiredStateAssignmentStatusUpdatedEvent](#)
 - [Header example](#)
 - [Payload example](#)

Operations

SUB ps.\$tenantId.ota-updates.events

Event that is sent when an OTA Update assignment has been created or updated.

Operation ID: `subscribeFotaDistributionUpdateMessage`

Accepts **one** of the following messages:

- `assignmentCreatedEvent`
- `assignmentStatusUpdatedEvent`
- `assignmentCorrelatedWithLogEvent`
- `assignmentSoftwareNodeVerifiedEvent`
- `assignmentUpdateConditionFulfilledEvent`

- **assignmentDeletedEvent**

assignmentCreatedEvent

Event that is sent when an OTA Update assignment has been created.

Header example

```
{
  "message-type": "assignmentCreatedEvent",
  "campaign": "string",
  "logFile": "string",
  "distributionPackage": "string",
  "device": "string",
  "downloadableContainer": "string",
  "vehicle": "string",
  "assignment": "string",
  "baseline": "string",
  "VIN": "string"
}
```

Payload example

```
{
  "assignment": {
    "assignmentId": "string",
    "status": "WAITING_FOR_BACKEND_SIGNOFF",
    "vehicle": {
      "vehicleId": "string",
      "VIN": "string",
      "modelName": "string"
    },
    "type": "CORE_ASSIGNMENT",
    "createdOn": "2019-08-24T14:15:22Z",
    "lastUpdateReported": "2019-08-24T14:15:22Z",
    "statusCode": 0,
    "statusMessage": "string",
    "privileged": true,
    "target": {
      "cuId": "string"
    },
    "signoffRequired": true,
    "managerApprovalRequired": true,
    "revocationRequested": true,
    "managerApproval": {
      "approvalDecision": "APPROVAL_DECISION_APPROVED",
      "schedulingTime": "2019-08-24T14:15:22Z",
      "approvalMessage": "string"
    },
    "revocationRequest": {
      "reason": "string",
      "requestedOn": "2019-08-24T14:15:22Z"
    },
    "retries": {
      "totalCount": 0,
      "lastAttempted": "2019-08-24T14:15:22Z"
    },
    "timeouts": {
      "expiresOn": "2019-08-24T14:15:22Z",
      "signoffExpiry": "string",
      "managerApprovalExpiry": "string",
      "signoffExpiresOn": "2019-08-24T14:15:22Z",
      "managerApprovalExpiresOn": "2019-08-24T14:15:22Z"
    },
    "softwareVersionVerifications": {
      "summary": "VERIFIED",
      "lastEvaluatedOn": "2019-08-24T14:15:22Z",
      "versions": [
        {
          "nodeId": "string",
          "rawVersion": "string",
          "verified": true,
          "verifiedOn": "2019-08-24T14:15:22Z"
        }
      ]
    },
    "updateConditions": {
      "summary": "FULFILLED",
      "lastEvaluatedOn": "2019-08-24T14:15:22Z",

```

Preliminary

```

    "conditions": {
      "estimatedDuration": {
        "conditionValue": "string",
        "fulfilledOn": "2019-08-24T14:15:22Z",
        "fulfilled": true
      }
    },
    "externalReference": {
      "reference": "string"
    },
    "assignable": {
      "packageId": "string",
      "descriptorType": "BASIC",
      "packageName": "string",
      "containerId": "string"
    },
    "lastUpdate": "2019-08-24T14:15:22Z",
    "lastUpdatedOn": "2019-08-24T14:15:22Z",
    "id": "string",
    "baseline": {
      "name": "string"
    },
    "actionStatusOverview": {
      "identified": 0,
      "running": 0,
      "success": 0,
      "failed": 0
    },
    "actions": [
      {
        "component": {
          "id": "string",
          "domain": "string"
        },
        "lastUpdateReported": "2019-08-24T14:15:22Z",
        "status": "string",
        "statusSummary": "IDENTIFIED",
        "message": "string",
        "progress": 100
      }
    ],
    "priority": 100,
    "deleted": true
  },
  "correlations": {
    "campaign": "string",
    "logFile": "string",
    "distributionPackage": "string",
    "device": "string",
    "downloadableContainer": "string",
    "vehicle": "string",
    "assignment": "string",
    "baseline": "string",
    "VIN": "string"
  }
}

```

Preliminary

assignmentStatusUpdatedEvent

Event that is sent when the status of an OTA Update assignment has been updated.

Header example

```

{
  "message-type": "assignmentStatusUpdatedEvent",
  "campaign": "string",
  "logFile": "string",
  "distributionPackage": "string",
  "device": "string",
  "downloadableContainer": "string",
  "vehicle": "string",
  "assignment": "string",
  "baseline": "string",
  "VIN": "string"
}

```

Payload example

```
{
  "assignment": {
    "assignmentId": "string",
    "status": "WAITING_FOR_BACKEND_SIGNOFF",
    "vehicle": {
      "vehicleId": "string",
      "VIN": "string",
      "modelName": "string"
    },
    "type": "CORE_ASSIGNMENT",
    "createdOn": "2019-08-24T14:15:22Z",
    "lastUpdateReported": "2019-08-24T14:15:22Z",
    "statusCode": 0,
    "statusMessage": "string",
    "privileged": true,
    "target": {
      "cuId": "string"
    },
    "signoffRequired": true,
    "managerApprovalRequired": true,
    "revocationRequested": true,
    "managerApproval": {
      "approvalDecision": "APPROVAL_DECISION_APPROVED",
      "schedulingTime": "2019-08-24T14:15:22Z",
      "approvalMessage": "string"
    },
    "revocationRequest": {
      "reason": "string",
      "requestedOn": "2019-08-24T14:15:22Z"
    },
    "retries": {
      "totalCount": 0,
      "lastAttempted": "2019-08-24T14:15:22Z"
    },
    "timeouts": {
      "expiresOn": "2019-08-24T14:15:22Z",
      "signoffExpiry": "string",
      "managerApprovalExpiry": "string",
      "signoffExpiresOn": "2019-08-24T14:15:22Z",
      "managerApprovalExpiresOn": "2019-08-24T14:15:22Z"
    },
    "softwareVersionVerifications": {
      "summary": "VERIFIED",
      "lastEvaluatedOn": "2019-08-24T14:15:22Z",
      "versions": [
        {
          "nodeId": "string",
          "rawVersion": "string",
          "verified": true,
          "verifiedOn": "2019-08-24T14:15:22Z"
        }
      ]
    },
    "updateConditions": {
      "summary": "FULFILLED",
      "lastEvaluatedOn": "2019-08-24T14:15:22Z",
      "conditions": {
        "estimatedDuration": {
          "conditionValue": "string",
          "fulfilledOn": "2019-08-24T14:15:22Z",
          "fulfilled": true
        }
      }
    },
    "externalReference": {
      "reference": "string"
    },
    "assignable": {
      "packageId": "string",
      "descriptorType": "BASIC",
      "packageName": "string",
      "containerId": "string"
    },
    "lastUpdate": "2019-08-24T14:15:22Z",
    "lastUpdatedOn": "2019-08-24T14:15:22Z",
    "id": "string",
    "baseline": {
      "name": "string"
    },
    "actionStatusOverview": {
      "identified": 0,
      "running": 0,
      "success": 0,
      "failed": 0
    }
  },
}
```

Preliminary

```

    "actions": [
      {
        "component": {
          "id": "string",
          "domain": "string"
        },
        "lastUpdateReported": "2019-08-24T14:15:22Z",
        "status": "string",
        "statusSummary": "IDENTIFIED",
        "message": "string",
        "progress": 100
      }
    ],
    "priority": 100,
    "deleted": true
  },
  "correlations": {
    "campaign": "string",
    "logFile": "string",
    "distributionPackage": "string",
    "device": "string",
    "downloadableContainer": "string",
    "vehicle": "string",
    "assignment": "string",
    "baseline": "string",
    "VIN": "string"
  }
}

```

assignmentCorrelatedWithLogEvent

Event that is sent when an OTA Update assignment has been correlated with a log.

Header example

```

{
  "message-type": "assignmentCorrelatedWithLogEvent",
  "campaign": "string",
  "logFile": "string",
  "distributionPackage": "string",
  "device": "string",
  "downloadableContainer": "string",
  "vehicle": "string",
  "assignment": "string",
  "baseline": "string",
  "VIN": "string"
}

```

Payload example

```

{
  "assignment": {
    "assignmentId": "string",
    "status": "WAITING_FOR_BACKEND_SIGNOFF",
    "vehicle": {
      "vehicleId": "string",
      "VIN": "string",
      "modelName": "string"
    },
    "type": "CORE_ASSIGNMENT",
    "createdOn": "2019-08-24T14:15:22Z",
    "lastUpdateReported": "2019-08-24T14:15:22Z",
    "statusCode": 0,
    "statusMessage": "string",
    "privileged": true,
    "target": {
      "cuId": "string"
    },
    "signoffRequired": true,
    "managerApprovalRequired": true,
    "revocationRequested": true,
    "managerApproval": {
      "approvalDecision": "APPROVAL_DECISION_APPROVED",
      "schedulingTime": "2019-08-24T14:15:22Z",
      "approvalMessage": "string"
    }
  }
}

```

```

    },
    "revocationRequest": {
      "reason": "string",
      "requestedOn": "2019-08-24T14:15:22Z"
    },
    "retries": {
      "totalCount": 0,
      "lastAttempted": "2019-08-24T14:15:22Z"
    },
    "timeouts": {
      "expiresOn": "2019-08-24T14:15:22Z",
      "signoffExpiry": "string",
      "managerApprovalExpiry": "string",
      "signoffExpiresOn": "2019-08-24T14:15:22Z",
      "managerApprovalExpiresOn": "2019-08-24T14:15:22Z"
    },
    "softwareVersionVerifications": {
      "summary": "VERIFIED",
      "lastEvaluatedOn": "2019-08-24T14:15:22Z",
      "versions": [
        {
          "nodeId": "string",
          "rawVersion": "string",
          "verified": true,
          "verifiedOn": "2019-08-24T14:15:22Z"
        }
      ]
    },
    "updateConditions": {
      "summary": "FULFILLED",
      "lastEvaluatedOn": "2019-08-24T14:15:22Z",
      "conditions": {
        "estimatedDuration": {
          "conditionValue": "string",
          "fulfilledOn": "2019-08-24T14:15:22Z",
          "fulfilled": true
        }
      }
    },
    "externalReference": {
      "reference": "string"
    },
    "assignable": {
      "packageId": "string",
      "descriptorType": "BASIC",
      "packageName": "string",
      "containerId": "string"
    },
    "lastUpdate": "2019-08-24T14:15:22Z",
    "lastUpdatedOn": "2019-08-24T14:15:22Z",
    "id": "string",
    "baseline": {
      "name": "string"
    },
    "actionStatusOverview": {
      "identified": 0,
      "running": 0,
      "success": 0,
      "failed": 0
    },
    "actions": [
      {
        "component": {
          "id": "string",
          "domain": "string"
        },
        "lastUpdateReported": "2019-08-24T14:15:22Z",
        "status": "string",
        "statusSummary": "IDENTIFIED",
        "message": "string",
        "progress": 100
      }
    ],
    "priority": 100,
    "deleted": true
  },
  "correlations": {
    "campaign": "string",
    "logFile": "string",
    "distributionPackage": "string",
    "device": "string",
    "downloadableContainer": "string",
    "vehicle": "string",
    "assignment": "string",
    "baseline": "string",
    "VIN": "string"
  }
}

```

assignmentSoftwareNodeVerifiedEvent

Event that is sent when an OTA Update assignment software node has been verified.

Header example

```
{
  "message-type": "assignmentSoftwareNodeVerifiedEvent",
  "campaign": "string",
  "logFile": "string",
  "distributionPackage": "string",
  "device": "string",
  "downloadableContainer": "string",
  "vehicle": "string",
  "assignment": "string",
  "baseline": "string",
  "VIN": "string"
}
```

Payload example

```
{
  "assignment": {
    "assignmentId": "string",
    "status": "WAITING_FOR_BACKEND_SIGNOFF",
    "vehicle": {
      "vehicleId": "string",
      "VIN": "string",
      "modelName": "string"
    },
    "type": "CORE_ASSIGNMENT",
    "createdOn": "2019-08-24T14:15:22Z",
    "lastUpdateReported": "2019-08-24T14:15:22Z",
    "statusCode": 0,
    "statusMessage": "string",
    "privileged": true,
    "target": {
      "cuId": "string"
    },
    "signoffRequired": true,
    "managerApprovalRequired": true,
    "revocationRequested": true,
    "managerApproval": {
      "approvalDecision": "APPROVAL_DECISION_APPROVED",
      "schedulingTime": "2019-08-24T14:15:22Z",
      "approvalMessage": "string"
    },
    "revocationRequest": {
      "reason": "string",
      "requestedOn": "2019-08-24T14:15:22Z"
    },
    "retries": {
      "totalCount": 0,
      "lastAttempted": "2019-08-24T14:15:22Z"
    },
    "timeouts": {
      "expiresOn": "2019-08-24T14:15:22Z",
      "signoffExpiry": "string",
      "managerApprovalExpiry": "string",
      "signoffExpiresOn": "2019-08-24T14:15:22Z",
      "managerApprovalExpiresOn": "2019-08-24T14:15:22Z"
    },
    "softwareVersionVerifications": {
      "summary": "VERIFIED",
      "lastEvaluatedOn": "2019-08-24T14:15:22Z",
      "versions": [
        {
          "nodeId": "string",
          "rawVersion": "string",
          "verified": true,
          "verifiedOn": "2019-08-24T14:15:22Z"
        }
      ]
    },
    "updateConditions": {
```

```

    "summary": "FULFILLED",
    "lastEvaluatedOn": "2019-08-24T14:15:22Z",
    "conditions": {
      "estimatedDuration": {
        "conditionValue": "string",
        "fulfilledOn": "2019-08-24T14:15:22Z",
        "fulfilled": true
      }
    },
    "externalReference": {
      "reference": "string"
    },
    "assignable": {
      "packageId": "string",
      "descriptorType": "BASIC",
      "packageName": "string",
      "containerId": "string"
    },
    "lastUpdate": "2019-08-24T14:15:22Z",
    "lastUpdatedOn": "2019-08-24T14:15:22Z",
    "id": "string",
    "baseline": {
      "name": "string"
    },
    "actionStatusOverview": {
      "identified": 0,
      "running": 0,
      "success": 0,
      "failed": 0
    },
    "actions": [
      {
        "component": {
          "id": "string",
          "domain": "string"
        },
        "lastUpdateReported": "2019-08-24T14:15:22Z",
        "status": "string",
        "statusSummary": "IDENTIFIED",
        "message": "string",
        "progress": 100
      }
    ],
    "priority": 100,
    "deleted": true
  },
  "correlations": {
    "campaign": "string",
    "logFile": "string",
    "distributionPackage": "string",
    "device": "string",
    "downloadableContainer": "string",
    "vehicle": "string",
    "assignment": "string",
    "baseline": "string",
    "VIN": "string"
  }
}

```

Preliminary

assignmentUpdateConditionFulfilledEvent

Event that is sent when an OTA Update assignment condition has been fulfilled.

Header example

```

{
  "message-type": "assignmentUpdateConditionFulfilledEvent",
  "campaign": "string",
  "logFile": "string",
  "distributionPackage": "string",
  "device": "string",
  "downloadableContainer": "string",
  "vehicle": "string",
  "assignment": "string",
  "baseline": "string",
  "VIN": "string"
}

```

Payload example

```
{
  "assignment": {
    "assignmentId": "string",
    "status": "WAITING_FOR_BACKEND_SIGNOFF",
    "vehicle": {
      "vehicleId": "string",
      "VIN": "string",
      "modelName": "string"
    },
    "type": "CORE_ASSIGNMENT",
    "createdOn": "2019-08-24T14:15:22Z",
    "lastUpdateReported": "2019-08-24T14:15:22Z",
    "statusCode": 0,
    "statusMessage": "string",
    "privileged": true,
    "target": {
      "cuId": "string"
    },
    "signoffRequired": true,
    "managerApprovalRequired": true,
    "revocationRequested": true,
    "managerApproval": {
      "approvalDecision": "APPROVAL_DECISION_APPROVED",
      "schedulingTime": "2019-08-24T14:15:22Z",
      "approvalMessage": "string"
    },
    "revocationRequest": {
      "reason": "string",
      "requestedOn": "2019-08-24T14:15:22Z"
    },
    "retries": {
      "totalCount": 0,
      "lastAttempted": "2019-08-24T14:15:22Z"
    },
    "timeouts": {
      "expiresOn": "2019-08-24T14:15:22Z",
      "signoffExpiry": "string",
      "managerApprovalExpiry": "string",
      "signoffExpiresOn": "2019-08-24T14:15:22Z",
      "managerApprovalExpiresOn": "2019-08-24T14:15:22Z"
    },
    "softwareVersionVerifications": {
      "summary": "VERIFIED",
      "lastEvaluatedOn": "2019-08-24T14:15:22Z",
      "versions": [
        {
          "nodeId": "string",
          "rawVersion": "string",
          "verified": true,
          "verifiedOn": "2019-08-24T14:15:22Z"
        }
      ]
    },
    "updateConditions": {
      "summary": "FULFILLED",
      "lastEvaluatedOn": "2019-08-24T14:15:22Z",
      "conditions": {
        "estimatedDuration": {
          "conditionValue": "string",
          "fulfilledOn": "2019-08-24T14:15:22Z",
          "fulfilled": true
        }
      }
    },
    "externalReference": {
      "reference": "string"
    },
    "assignable": {
      "packageId": "string",
      "descriptorType": "BASIC",
      "packageName": "string",
      "containerId": "string"
    },
    "lastUpdate": "2019-08-24T14:15:22Z",
    "lastUpdatedOn": "2019-08-24T14:15:22Z",
    "id": "string",
    "baseline": {
      "name": "string"
    },
    "actionStatusOverview": {
      "identified": 0,
      "running": 0,
      "success": 0,
      "failed": 0
    }
  },
}
```

Preliminary

```

    "actions": [
      {
        "component": {
          "id": "string",
          "domain": "string"
        },
        "lastUpdateReported": "2019-08-24T14:15:22Z",
        "status": "string",
        "statusSummary": "IDENTIFIED",
        "message": "string",
        "progress": 100
      }
    ],
    "priority": 100,
    "deleted": true
  },
  "correlations": {
    "campaign": "string",
    "logFile": "string",
    "distributionPackage": "string",
    "device": "string",
    "downloadableContainer": "string",
    "vehicle": "string",
    "assignment": "string",
    "baseline": "string",
    "VIN": "string"
  }
}

```

assignmentDeletedEvent

Event that is sent when an OTA Update assignment has been deleted.

Header example

```

{
  "message-type": "assignmentDeletedEvent",
  "campaign": "string",
  "logFile": "string",
  "distributionPackage": "string",
  "device": "string",
  "downloadableContainer": "string",
  "vehicle": "string",
  "assignment": "string",
  "baseline": "string",
  "VIN": "string"
}

```

Payload example

```

{
  "assignment": {
    "assignmentId": "string",
    "status": "WAITING_FOR_BACKEND_SIGNOFF",
    "vehicle": {
      "vehicleId": "string",
      "VIN": "string",
      "modelName": "string"
    },
    "type": "CORE_ASSIGNMENT",
    "createdOn": "2019-08-24T14:15:22Z",
    "lastUpdateReported": "2019-08-24T14:15:22Z",
    "statusCode": 0,
    "statusMessage": "string",
    "privileged": true,
    "target": {
      "cuId": "string"
    },
    "signoffRequired": true,
    "managerApprovalRequired": true,
    "revocationRequested": true,
    "managerApproval": {
      "approvalDecision": "APPROVAL_DECISION_APPROVED",
      "schedulingTime": "2019-08-24T14:15:22Z",
      "approvalMessage": "string"
    }
  }
}

```

```

    },
    "revocationRequest": {
      "reason": "string",
      "requestedOn": "2019-08-24T14:15:22Z"
    },
    "retries": {
      "totalCount": 0,
      "lastAttempted": "2019-08-24T14:15:22Z"
    },
    "timeouts": {
      "expiresOn": "2019-08-24T14:15:22Z",
      "signoffExpiry": "string",
      "managerApprovalExpiry": "string",
      "signoffExpiresOn": "2019-08-24T14:15:22Z",
      "managerApprovalExpiresOn": "2019-08-24T14:15:22Z"
    },
    "softwareVersionVerifications": {
      "summary": "VERIFIED",
      "lastEvaluatedOn": "2019-08-24T14:15:22Z",
      "versions": [
        {
          "nodeId": "string",
          "rawVersion": "string",
          "verified": true,
          "verifiedOn": "2019-08-24T14:15:22Z"
        }
      ]
    },
    "updateConditions": {
      "summary": "FULFILLED",
      "lastEvaluatedOn": "2019-08-24T14:15:22Z",
      "conditions": {
        "estimatedDuration": {
          "conditionValue": "string",
          "fulfilledOn": "2019-08-24T14:15:22Z",
          "fulfilled": true
        }
      }
    },
    "externalReference": {
      "reference": "string"
    },
    "assignable": {
      "packageId": "string",
      "descriptorType": "BASIC",
      "packageName": "string",
      "containerId": "string"
    },
    "lastUpdate": "2019-08-24T14:15:22Z",
    "lastUpdatedOn": "2019-08-24T14:15:22Z",
    "id": "string",
    "baseline": {
      "name": "string"
    },
    "actionStatusOverview": {
      "identified": 0,
      "running": 0,
      "success": 0,
      "failed": 0
    },
    "actions": [
      {
        "component": {
          "id": "string",
          "domain": "string"
        },
        "lastUpdateReported": "2019-08-24T14:15:22Z",
        "status": "string",
        "statusSummary": "IDENTIFIED",
        "message": "string",
        "progress": 100
      }
    ],
    "priority": 100,
    "deleted": true
  },
  "correlations": {
    "campaign": "string",
    "logFile": "string",
    "distributionPackage": "string",
    "device": "string",
    "downloadableContainer": "string",
    "vehicle": "string",
    "assignment": "string",
    "baseline": "string",
    "VIN": "string"
  }
}

```

SUB ps.\$tenantId.desired-states.events

Event that is sent when a desired state assignment has been created or updated.

Operation ID: `subscribeDesiredStateAssignmentMessage`

Accepts **one** of the following messages:

- `desiredStateAssignmentCreatedEvent`
- `desiredStateAssignmentStatusUpdatedEvent`

desiredStateAssignmentCreatedEvent

Event that is sent when a desired state assignment has been created.

Header example

```
{
  "message-type": "desiredStateAssignmentCreatedEvent"
}
```

Payload example

```
{
  "desiredStateAssignment": {
    "desiredStateAssignmentId": "string",
    "target": {
      "id": "string",
      "type": "VEHICLE"
    },
    "desiredState": {
      "name": "string"
    },
    "status": "ASSIGNED",
    "overallStatusSummary": {
      "total": 0,
      "pending": 0,
      "running": 0,
      "successful": 0,
      "failed": 0,
      "finished": 0
    },
    "actions": [
      {
        "status": "IDENTIFIED",
        "message": "string",
        "component": {
          "id": "string",
          "version": "string"
        }
      }
    ],
    "statusCode": 0,
    "statusMessage": "string",
    "createdOn": "2019-08-24T14:15:22Z",
    "lastUpdatedOn": "2019-08-24T14:15:22Z",
    "lastUpdateReported": "2019-08-24T14:15:22Z",
    "criteria": [
      "string"
    ]
  },
  "correlations": {}
}
```

desiredStateAssignmentStatusUpdatedEvent

Event that is sent when a desired state assignment has been updated.

Header example

```
{
  "message-type": "desiredStateAssignmentStatusUpdatedEvent"
}
```

Payload example

```
{
  "desiredStateAssignment": {
    "desiredStateAssignmentId": "string",
    "target": {
      "id": "string",
      "type": "VEHICLE"
    },
    "desiredState": {
      "name": "string"
    },
    "status": "ASSIGNED",
    "overallStatusSummary": {
      "total": 0,
      "pending": 0,
      "running": 0,
      "successful": 0,
      "failed": 0,
      "finished": 0
    },
    "actions": [
      {
        "status": "IDENTIFIED",
        "message": "string",
        "component": {
          "id": "string",
          "version": "string"
        }
      }
    ],
    "statusCode": 0,
    "statusMessage": "string",
    "createdOn": "2019-08-24T14:15:22Z",
    "lastUpdatedOn": "2019-08-24T14:15:22Z",
    "lastUpdateReported": "2019-08-24T14:15:22Z",
    "criteria": [
      "string"
    ]
  },
  "correlations": {}
}
```

Preliminary

11.2.7 OTA Updates Inventory Exporter

Inventory Kafka Exporter API 1.0.0

The Inventory Exporter API allows you to consume Inventory updates of devices in an event-driven way.

This is in contrast to the REST API where the consumers of the API have to actively poll to notice any changes.

- **Operations**
 - `SUB ps.$tenantId.inventories.events`
 - `inventoryUpdatedExportEvent`
 - [Header example](#)
 - [Payload example](#)
 - `vehicleInventoryUpdatedExportEvent`
 - [Header example](#)
 - [Payload example](#)

Operations

SUB ps.\$tenantId.inventories.events

Event that is sent when an Inventory item has been created or updated.

Operation ID: **subscribeInventoryUpdates**

Accepts **one** of the following messages:

- **inventoryUpdatedExportEvent**
- **vehicleInventoryUpdatedExportEvent**

inventoryUpdatedExportEvent

Event that is sent when an Inventory has been updated.

Header example

```
{
  "message-type": "inventoryUpdatedEvent",
  "device": "string",
  "assignment": "string",
  "baselineAssignment": "string",
  "campaign": "string"
}
```

Payload example

```
{
  "inventory": {
    "device": {
      "id": "string",
      "registered": true,
      "capabilities": {
        "inventory": 3
      }
    },
    "outdated": true,
    "lastUpdated": "2019-08-24T14:15:22Z",
    "lastUpdateReported": "2019-08-24T14:15:22Z",
    "requestedOn": "2019-08-24T14:15:22Z",
    "notifiedOn": "2019-08-24T14:15:22Z",
    "inventoryItems": [
      {
        "type": "SOFTWARE_HARDWARE",
        "swUid": "string",
        "swVersion": "string",
        "hwUid": "string",
        "hwVersion": "string"
      }
    ],
    "securityData": [
      {
        "softwareNodeIds": [
          "string"
        ],
        "security": "string"
      }
    ]
  },
  "inventoryItemList": [
    {
      "type": "SOFTWARE_HARDWARE",
      "swUid": "string",
      "swVersion": "string",
      "hwUid": "string",
      "hwVersion": "string"
    }
  ],
  "correlations": {
    "device": "string",
    "assignment": "string",
    "baselineAssignment": "string",
    "campaign": "string"
  }
}
```

```
}  
}
```

vehicleInventoryUpdatedExportEvent

Event that is sent when a Vehicle Inventory has been updated.

Header example

```
{  
  "message-type": "vehicleInventoryUpdatedEvent",  
  "device": "string",  
  "assignment": "string",  
  "baselineAssignment": "string",  
  "campaign": "string"  
}
```

Payload example

```
{  
  "vehicleInventory": {  
    "vehicleId": "string",  
    "inventoryItems": [  
      {  
        "type": "SOFTWARE_HARDWARE",  
        "swUid": "string",  
        "swVersion": "string",  
        "hwUid": "string",  
        "hwVersion": "string"  
      }  
    ],  
    "devices": [  
      {  
        "deviceId": "string",  
        "outdated": true,  
        "lastUpdateReported": "2019-08-24T14:15:22Z",  
        "requestedOn": "2019-08-24T14:15:22Z",  
        "notifiedOn": "2019-08-24T14:15:22Z",  
        "registered": true,  
        "capabilities": {  
          "inventory": 3  
        },  
        "securityData": [  
          {  
            "softwareNodeIds": [  
              "string"  
            ],  
            "security": "string"  
          }  
        ]  
      }  
    ],  
    "correlations": {  
      "device": "string",  
      "assignment": "string",  
      "baselineAssignment": "string",  
      "campaign": "string"  
    }  
  }  
}
```

Preliminary

11.3 C-Hub glossary

Term	Description
Automotive Connectivity Hub	The Automotive Connectivity Hub addresses commercial vehicles, and especially B2B customers which are interested in collecting live data of their vehicles. Once stored in a cloud infrastructure, such data can be consumed by other digital services and solutions. See https://www.bosch-mobility.com/en/solutions/connectivity/automotive-connectivity-hub/
C-Hub	C-Hub is the short name used for the <i>Automotive Connectivity Hub</i> .
Connectivity as a Service Caas	Connectivity as a Service - short form is CaaS - is a business model where the customer uses the cloud services in self-service mode without Bosch as an intermediary. ▪ The customer can use the C-Hub backend in self-service mode (cloud-based UIs and user guides are accessible) ▪ The customer's applications or third-party services can access the data via API ▪ The customer has full ownership of any additional applications which he chooses to run on the C-Hub TCU2 hardware ▪ Robert Bosch GmbH manages only the cloud services and device software
Data as a Service DaaS	Data as a Service - short form is DaaS - is a business model where data is made available online on demand and forwarded to the customer's infrastructure or data consumer applications. The customer's user have no direct interaction with the C-Hub backend but all configuration is done by requesting it via the C-Hub support. C-Hub provides 2 service plans ▪ Data as a Service Base DaaS Base is designed to read standardized vehicle data (e.g., J1939, FMS) and standardized diagnostics data (e.g. any OBD-II standard). ▪ Data as a Service Premium DaaS Premium includes all Base capabilities and additionally provides access to non-standardized, OEM-specific data.
Diagnostic Trouble Code DTC	OBD-II defines five-character-long Diagnostic Trouble Codes (DTC) to indicate to vehicle technicians the existing on-board problems. See https://en.wikipedia.org/wiki/On-board_diagnostics#OBD-II_diagnostic_trouble_codes
On-board diagnostics OBD-II	On-board diagnostics - short form is OBD - is a term referring to a vehicle's self-diagnostic and reporting capability. OBD systems give the vehicle owner or repair technician access to the status of the various vehicle sub-systems. OBD-II is one of the current standards for vehicle on-board diagnostics. It specifies the type of diagnostic connector and its pinout, the electrical signaling protocols available, and the messaging format. See https://en.wikipedia.org/wiki/On-board_diagnostics

Term	Description
Telematic Control Unit TCU TCU2 CCU	The Automotive Connectivity Hub uses a Bosch-manufactured telematic control unit - hereinafter called C-Hub TCU2 - as well as some required accessories for its installation into the vehicle. This is the main physical device necessary for the data collection of a vehicle. The technical designation of the device for the European market is TCU2 EU IP30 C-Hub. The technical designation of the device for the North America market is TCU2 NA IP30 C-Hub. The short name used in our guides is "C-Hub TCU2". Some sections might mention CCU = Connectivity Control Unit, however in the C-Hub context we always have the same device in mind, namely the C-Hub TCU2.
DIN Adapter	Hardware device used to be merged together with a C-Hub TCU2 device. The adapter facilitates the installation in a commercial vehicle which provides a DIN-slot.
Vehicle Management Solution VMS	The Vehicle Management Solution is an enabler for cloud-based services such as FOTA, Predictive Diagnostics, Fleet Management, and Remote Measurement. In the context of C-Hub most of the services on the cloud infrastructure are described in their User Guide under this label. https://ota.eu.bosch-mobility-cloud.com/INSTANCE/#/vms/user-guide
Telematics Service Provider TSP	Telematics Service Providers (TSPs) are companies which offer services to fleet operators, enabling them to monitor their vehicles and drivers for a subscription fee. Some of the services provided by TSPs include vehicle tracking, driver behavior monitoring, remote diagnostics, and predictive maintenance. These services allow fleet managers to track the location and movements of their vehicles in real-time, monitor the performance and behavior of their drivers, and identify potential maintenance issues before they become major problems. See https://www.ptolemus.com/what-are-telematics-service-providers/
Tenant	In the C-Hub backend context, a tenant represents a customer. - Cloud services which support multi-tenancy can assure the clear data separation between different customers although they share the same instances. - Multi-tenancy contrasts with multi-instance architectures, where separate software instances operate on behalf of different tenants.

11.4 Declaration of conformity - C-Hub TCU2 device

Type approval validity and re-certification

Target Market Type approvals are subject to expiration, particularly when regulatory frameworks evolve. If this occurs, the CUSTOMER can request re-certification from Bosch. This process typically involves additional costs as Bosch must reevaluate the product against the updated regulations to renew its approval status.

11.4.1 Regions subject to CE marking



Countries in the EU

Country	Simplified declaration of conformity
Austria (AT)	Hiermit erklärt Robert Bosch GmbH, dass der Funkanlagentyp TCU2 EU IP30 C-Hub der Richtlinie 2014/53/EU entspricht. Der vollständige Text der EU-Konformitätserklärung ist unter der folgenden Internetadresse verfügbar: https://doc-ita.bosch.com/?r=EU
Belgium (BE)	Le soussigné, Robert Bosch GmbH, déclare que l'équipement radioélectrique du type TCU2 EU IP30 C-Hub est conforme à la directive 2014/53/UE. Le texte complet de la déclaration UE de conformité est disponible à l'adresse internet suivante: https://doc-ita.bosch.com/?r=EU
Bulgaria (BG)	С настоящото Robert Bosch GmbH декларира, че този тип радиосъоръжение TCU2 EU IP30 C-Hub е в съответствие с Директива 2014/53/ЕС. Цялостният текст на ЕС декларацията за съответствие може да се намери на следния интернет адрес: https://doc-ita.bosch.com/?r=EU
Croatia (HR)	Robert Bosch GmbH ovime izjavljuje da je radijska oprema tipa TCU2 EU IP30 C-Hub u skladu s Direktivom 2014/53/EU. Cjeloviti tekst EU izjave o sukladnosti dostupan je na sljedećoj internetskoj adresi: https://doc-ita.bosch.com/?r=EU
Cyprus (CY)	Με την παρούσα ο/η Robert Bosch GmbH, δηλώνει ότι ο ραδιοεξοπλισμός TCU2-EUIP30 C-HUB πληροί την οδηγία 2014/53/ΕΕ. TCU2 EU IP30 C-Hub Το πλήρες κείμενο της δήλωσης συμμόρφωσης ΕΕ διατίθεται στην ακόλουθη ιστοσελίδα στο διαδίκτυο: https://doc-ita.bosch.com/?r=EU
Czechia (CZ)	Tímto Robert Bosch GmbH prohlašuje, že typ rádiového zařízení TCU2 EU IP30 C-Hub je v souladu se směrnicí 2014/53/EU. Úplné znění EU prohlášení o shodě je k dispozici na této internetové adrese: https://doc-ita.bosch.com/?r=EU
Denmark (DK)	Hermed erklærer Robert Bosch GmbH, at radioudstyrstypen TCU2 EU IP30 C-Hub er i overensstemmelse med direktiv 2014/53/EU. EU-overensstemmelseserklæringens fulde tekst kan findes på følgende internetadresse: https://doc-ita.bosch.com/?r=EU
Estonia (EE)	Käesolevaga deklareerib Robert Bosch GmbH, et käesolev raadioseadme tüüp TCU2 EU IP30 C-Hub vastab direktiivi 2014/53/EL nõuetele. ELi vastavusdeklaratsiooni täielik tekst on kättesaadav järgmisel internetiaadressil: https://doc-ita.bosch.com/?r=EU
Finland (FI)	Robert Bosch GmbH vakuuttaa, että radiolaitetyyppi TCU2 EU IP30 C-Hub on direktiivin 2014/53/EU mukainen. EU-vaatimustenmukaisuusvakuutuksen täysimittainen teksti on saatavilla seuraavassa internetosoitteessa: https://doc-ita.bosch.com/?r=EU

Preliminary

France (FR)	Le soussigné, Robert Bosch GmbH, déclare que l'équipement radioélectrique du type TCU2 EU IP30 C-Hub est conforme à la directive 2014/53/UE. Le texte complet de la déclaration UE de conformité est disponible à l'adresse internet suivante: https://eu-doc.bosch.com/
Germany (DE)	Hiermit erklärt Robert Bosch GmbH, dass der Funkanlagentyp TCU2-EU-IP30-C-HUB der Richtlinie 2014/53/EU entspricht. Der vollständige Text der EU-Konformitätserklärung ist unter der folgenden Internetadresse verfügbar: https://eu-doc.bosch.com/
Greece (GR)	Με την παρούσα ο/η Robert Bosch GmbH, δηλώνει ότι ο ραδιοεξοπλισμός TCU2 EU IP30 C-Hub πληροί την οδηγία 2014/53/ΕΕ. Το πλήρες κείμενο της δήλωσης συμμόρφωσης ΕΕ διατίθεται στην ακόλουθη ιστοσελίδα στο διαδίκτυο: https://eu-doc.bosch.com/
Hungary (HU)	Robert Bosch GmbH igazolja, hogy a TCU2 EU IP30 C-Hub típusú rádióberendezés megfelel a 2014/53/EU irányelvnek. Az EU-megfelelőségi nyilatkozat teljes szövege elérhető a következő internetes címen: https://eu-doc.bosch.com/
Ireland (IE)	Hereby, Robert Bosch GmbH declares that the radio equipment type TCU2 EU IP30 C-Hub is in compliance with Directive 2014/53/EU. The full text of the EU declaration of conformity is available at the following internet address: https://eu-doc.bosch.com/
Italy (IT)	Il fabbricante, Robert Bosch GmbH, dichiara che il tipo di apparecchiatura radio TCU2 EU IP30 C-Hub è conforme alla direttiva 2014/53/UE. Il testo completo della dichiarazione di conformità UE è disponibile al seguente indirizzo Internet: https://doc-ita.bosch.com/?r=EU
Latvia (LV)	Ar šo Robert Bosch GmbH deklarē, ka radioiekārta TCU2 EU IP30 C-Hub atbilst Direktīvai 2014/53/ES. Pilns ES atbilstības deklarācijas teksts ir pieejams šādā interneta vietnē: https://doc-ita.bosch.com/?r=EU
Lithuania (LT)	Aš, Robert Bosch GmbH, patvirtinu, kad radijo įrenginių tipas TCU2 EU IP30 C-Hub atitinka Direktyvą 2014/53/ES. Visas ES atitikties deklaracijos tekstas prieinamas šiuo interneto adresu: https://doc-ita.bosch.com/?r=EU
Luxembourg (LU)	Le soussigné, Robert Bosch GmbH, déclare que l'équipement radioélectrique du type TCU2 EU IP30 C-Hub est conforme à la directive 2014/53/UE. Le texte complet de la déclaration UE de conformité est disponible à l'adresse internet suivante: https://doc-ita.bosch.com/?r=EU
Malta (MT)	B'dan, Robert Bosch GmbH, niddikjara li dan it-tip ta' tagħmir tar-radju TCU2 EU IP30 C-Hub huwa konformi mad-Direttiva 2014/53/UE. It-test kollu tad-dikjarazzjoni ta' konformità tal-UE huwa disponibbli f'dan l-indirizz tal-Internet li ġej: https://doc-ita.bosch.com/?r=EU
Netherlands (NL)	Hierbij verklaar ik, Robert Bosch GmbH, dat het type radioapparatuur TCU2 EU IP30 C-Hub conform is met Richtlijn 2014/53/EU. De volledige tekst van de EU-conformiteitsverklaring kan worden geraadpleegd op het volgende internetadres: https://doc-ita.bosch.com/?r=EU

Poland (PL)	Robert Bosch GmbH niniejszym oświadcza, że typ urządzenia radiowego TCU2 EU IP30 C-Hub jest zgodny z dyrektywą 2014/53/UE. Pełny tekst deklaracji zgodności UE jest dostępny pod następującym adresem internetowym: https://doc-ita.bosch.com/?r=EU
Portugal (PT)	O(a) abaixo assinado(a) Robert Bosch GmbH declara que o presente tipo de equipamento de rádio TCU2 EU IP30 C-Hub está em conformidade com a Diretiva 2014/53/UE. O texto integral da declaração de conformidade está disponível no seguinte endereço de Internet: https://doc-ita.bosch.com/?r=EU
Romania (RO)	Prin prezenta, Robert Bosch GmbH declară că tipul de echipamente radio TCU2 EU IP30 C-Hub este în conformitate cu Directiva 2014/53/UE. Textul integral al declarației UE de conformitate este disponibil la următoarea adresă internet: https://doc-ita.bosch.com/?r=EU
Slovakia (SK)	Robert Bosch GmbH týmto vyhlasuje, že rádiové zariadenie typu TCU2 EU IP30 C-Hub je v súlade so smernicou 2014/53/EU. Uplné EU vyhlásenie o zhode je k dispozícii na tejto internetovej adrese: https://doc-ita.bosch.com/?r=EU
Slovenia (SI)	Robert Bosch GmbH potrjuje, da je tip radijske opreme TCU2 EU IP30 C-Hub skladen z Direktivo 2014/53/EU. Celotno besedilo izjave EU o skladnosti je na voljo na naslednjem spletnem naslovu: https://doc-ita.bosch.com/?r=EU
Spain (ES)	Por la presente, Robert Bosch GmbH declara que el tipo de equipo radioeléctrico TCU2 EU IP30 C-Hub es conforme con la Directiva 2014/53/UE. El texto completo de la declaración UE de conformidad está disponible en la dirección Internet siguiente: https://doc-ita.bosch.com/?r=EU
Sweden (SE)	Härmed försäkrar Robert Bosch GmbH att denna typ av radioutrustning TCU2 EU IP30 C-Hub överensstämmer med direktiv 2014/53/EU. Den fullständiga texten till EU-försäkran om överensstämmelse finns på följande webbadress: https://doc-ita.bosch.com/?r=EU

Other CE Countries

- Certification via France (FRA)
 - Andorra (AD)
 - Guadalupe (GP)
 - Martinique (MQ)
 - Monaco (MC)
 - Reunion (RE)
- Certification via Italy (ITA)
 - San Marino (SM)
- Other
 - Bosnia Herzegovina (BA)
 - Montenegro (ME)
 - North Macedonia (MK)

For this list of countries, the below reduced Declaration of Conformity must be translated before consideration:

Hereby, Robert Bosch GmbH declares that the radio equipment type TCU2 EU IP30 C-Hub is in compliance with the relevant statutory requirements. The full text of the Declaration of Conformity is available at the following internet address: <https://doc-ita.bosch.com/?r=EU>

11.4.2 Other region-specific statements

European Economic Area (EEA) countries

Lichtenstein (LI)	This device restricts the transmission of WLAN when the target is in Lichtenstein. Declaration of WLAN Transmission Power 2014/53/EU for Lichtenstein: WLAN 2412 – 2462 MHz <20 dBm WLAN 5180 – 5240 MHz <14 dBm WLAN 5745 – 5825 MHz <14 dBm
Iceland (IS)	No special restrictions.
Norway (NO)	No special restrictions.

Single markets and others

Switzerland (CH)	This device restricts the transmission of WLAN when the target is in Switzerland. Declaration of WLAN Transmission Power 2014/53/EU for Switzerland: WLAN 2412 – 2462 MHz <20 dBm WLAN 5180 – 5240 MHz <14 dBm WLAN 5745 – 5825 MHz <14 dBm
------------------	---

11.4.3 Regions subject to UKCA marking



UNITED KINGDOM (GB)

Hereby, Robert Bosch GmbH declares that the radio equipment type TCU2 EU IP30 C-Hub is in compliance with the relevant statutory requirements.

The full text of the Declaration of Conformity is available at the following internet address:

<https://doc-ita.bosch.com/search.html?r=GB&p=0&q=TCU2%2520>.

British Overseas Territories (BOTs)

The following countries are part of the British Overseas Territories and therefore follow the same regulatory requirements as the United Kingdom.

- Gibraltar (GI)
- Isle of Man (IM)
- Jersey (JE)

11.4.4 USA



FCC ID: 2AUXS-TCU2NAIP30A

Federal Communication Commission (FCC) regulatory information

Hereby, Robert Bosch GmbH declares that the equipment type **TCU2 NA IP30 C-Hub** is in compliance with Part 15 of the FCC Rules.

Operation is subject to the following two conditions:

1. This device may not cause harmful interference, and
2. This device must accept any interference received, including interference that may cause undesired operation of the device.

To comply with FCC Exposure requirements the CUSTOMER is instructed by the Grantee to assure a minimum separation distance of 20 cm between the antennas and any human body as documented in the filing.

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

11.4.5 Canada



IC ID: 25847-TCU2NAIP30

Contains IC ID: 4441A-UMCSTD31BPN

Innovation, Science and Economic Development Canada (ISED) regulatory information

Hereby, Robert Bosch GmbH declares that the radio equipment type **TCU2 NA IP30 C-Hub** is in compliance with the ISED rules.

This device contains licence-exempt transmitter(s)/receiver(s) that comply with Innovation, Science and Economic Development Canada's licence-exempt RSS(s).

Operation is subject to the following two conditions: (1) This device may not cause interference, and (2) This device must accept any interference, including interference that may cause undesired operation of the device.

Radiofrequency exposure information:

This equipment complies with Canada radiation exposure limits set forth for an uncontrolled environment.

Preliminary

This equipment should be installed and operated with minimum distance of 20 cm between the radiator and your body.

Avis de conformité à la réglementation d'Innovation, Sciences et Développement économique Canada (ISDE)

L'émetteur/récepteur exempt de licence contenu dans le présent appareil est conforme aux CNR d'Innovation, Sciences et Développement économique Canada applicables aux appareils radio exempts de licence.

L'exploitation est autorisée aux deux conditions suivantes: (1) L'appareil ne doit pas produire de brouillage, et (2) L'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Déclaration d'exposition aux radiations:

Cet équipement est conforme aux limites d'exposition aux rayonnements IC établies pour un environnement non contrôlé.

Cet équipement doit être installé et utilisé avec un minimum de 20 cm de distance entre la source de rayonnement et votre corps.

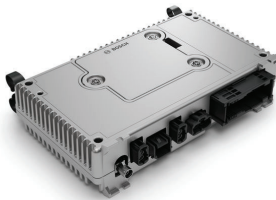
11.5 Additional accessories for TSP customers - not part of the standard offering for OEM

- [DIN Adapter - for TSP](#)
- [Declaration of conformity C-Hub DIN adapter](#)

11.5.1 DIN Adapter - for TSP

Description

The Automotive Connectivity Hub telematic control unit – for short C-Hub TCU2 device – is the second generation of the high-performance telematic control unit manufactured by Bosch.



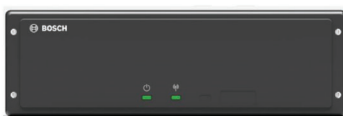
With the C-Hub TCU2 device, you can collect vehicle data and send this data to a cloud infrastructure.

The collected vehicle data will be processed on-board by the device and then sent to the cloud infrastructure via a mobile data network.

The accurate installation in the vehicles needs to be performed by a trained installation partner.

The device has been designed for operation in a commercial vehicle with a 12 V or 24 V electrical system.

Bosch assures the secure operation of the C-Hub TCU2 with over-the-air (OTA) updates of the device operating system and of application software.

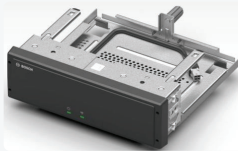
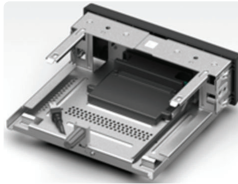


Direct access to important or commonly used functions of the device is not provided.

However, in case the device is mounted into a DIN adapter, two LEDs will show if the device is on or off.

A steady green light of both LEDs signals that the device is up and running and has successfully connected.

Technical data C-Hub DIN adapter

Power	Powered by the vehicle via the C-Hub main wiring harness.
Ambient temperature range	-40°C... +60°C full operational
Storage temperature	-40°C to +85°C
Dimensions	▪ Height 58.5 mm ▪ Width 187.5 mm ▪ Length 176.9 mm (200.6 mm with Guide Pin)
Weight	600 g +/-10%
Front view	
Back view	
Connectors	2 LEDs and ground

Device status indicated by 2 LEDs

Power LED	Connection LED	Meaning
Off	Off	No power supply to the device Troubleshooting: Check the cable connections and whether there is power supply reaching the C-Hub TCU2 device
On	Off	C-Hub TCU2 starting/booting LTE not connected (<i>connecting</i>) GNSS not connected (<i>connecting</i>)
On	Slow blinking: once, every 3 seconds 1. On for 1 second, off for 2 seconds 2. Repeat	✓ C-Hub TCU2 is powered up LTE connecting GNSS connecting

Preliminary

Power LED	Connection LED	Meaning
On	Fast blinking: twice, every 3 seconds 1. On for 0.5 seconds, off for 1 second 2. Repeat	✓ C-Hub TCU2 is powered up ✓ LTE connected GNSS connecting Troubleshooting: Check the network coverage for example on your mobile phone
On	Faster blinking: 3 times every 3 seconds 1. On for 0.5 seconds, off for 0.5 seconds 2. Repeat	✓ C-Hub TCU2 is powered up - LTE connecting ✓ GNSS connected Troubleshooting: Check the network coverage for example on your mobile phone
On	On	✓ C-Hub TCU2 is powered up ✓ LTE connected ✓ GNSS connected

Installation concept for telematics service providers (TSP)



- The initial installation into the vehicle is performed by a trained installation partner.
- The data which needs to be collected can be described by the customer in a human-readable form.
The requirements will be then scripted, and the configuration will be deployed on the customer's C-Hub TCU2 devices by Bosch.



A detailed **installation guide** will be provided as a dedicated document upon request.

11.5.2 Declaration of conformity C-Hub DIN adapter



Hereby, Robert Bosch GmbH declares that the equipment type **TCU2-IP30-DIN-ADAPTER-C-HUB** is in compliance with Directive 2014/30/EU.
The full text of the EU declaration of conformity is available at the following internet address: <https://doc-ita.bosch.com/index.html>



Hereby, Robert Bosch GmbH declares that the equipment type **TCU2-IP30-DIN-ADAPTER-C-HUB** is in compliance with the relevant statutory requirements.
The full text of the declaration of conformity is available at the following internet address: <https://doc-ita.bosch.com/index.html>

11.6 Additional information for OEM customers

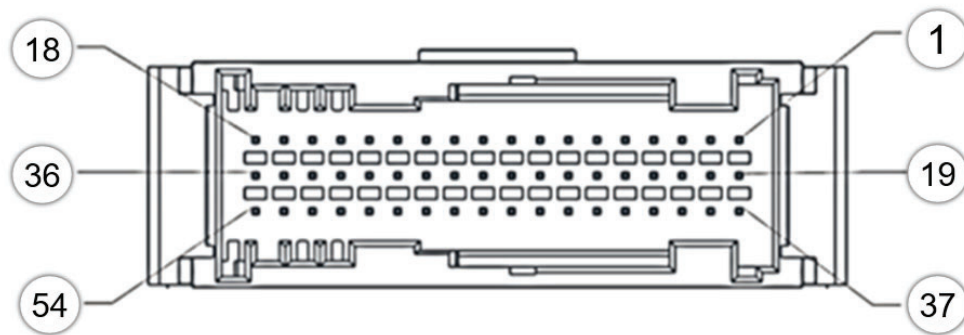
For OEM customers Bosch will provide upon request an additional document which covers following topics

- Product identification and part numbers
- Intended use and target markets
 - WiFi / WLAN frequency regulations
 - Field data acquisition and active field observation
- Technical data C-Hub TCU2 device
 - Details on power supply, consumption and power output
 - Electrical characteristics of the battery inside the TCU2 device
 - Technical data with measured variables and measuring conditions
 - Diagnostic Trouble Codes - C-Hub TCU2 device
 - Material and ingress protection
 - Cellular network
- Product safety
 - Restrictions in use
 - Declaration of conformity - long - not printed
- Connectors - C-Hub TCU2 device
 - GNSS Connector
 - Cellular radio connectors
 - USB Connector
 - BT/WLAN Connector
 - Main connector with pinning info
- System description
 - Software on the device
 - Time management
 - Bluetooth service
 - Cellular network
 - Diagnostics
 - Digital I/O
 - System resources
- Lifetime
- Recycling, repair, and disposal
- Auxiliary equipment - antenna and cabling
 - Auxiliary equipment - antenna and cabling for TCU2 EU IP30 C-Hub Equipment
 - Auxiliary equipment - antenna and cabling for TCU2 NA IP30 C-Hub Equipment
- Declaration of conformity C-Hub TCU2 device
 - Regions subject to CE marking
 - Other region-specific statements
 - Regions subject to UKCA marking
 - USA
 - Canada
- Further contact
- Appendices and referneces
 - UNECE Vehicle Categories
 - C-Hub glossary
 - Supported connections to the C-Hub TCU2 device

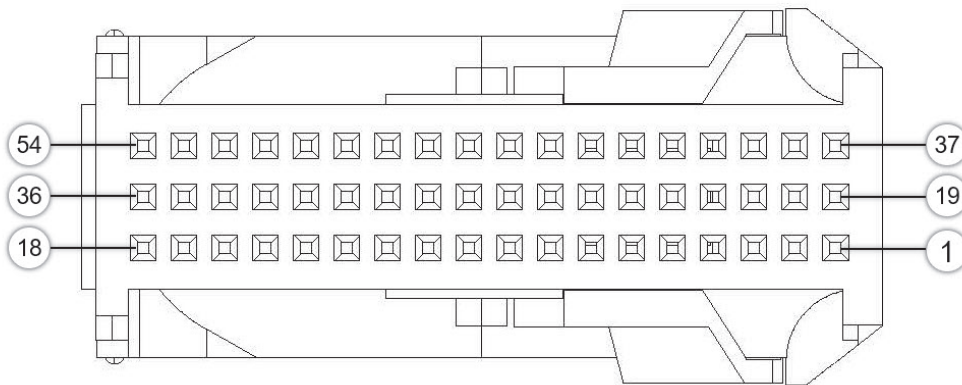
Preliminary

11.6.1 Main connector - pinning info

The **main connector on the C-Hub TCU2 device** has the following design and pin designation.



Its counterpart, the main connector **on the main wiring harness**, has the following design and pin designation.



Main connector pinning

1	KL.30	Supply Input
2	KL.30	Supply Input
3	CAN3_GND	Ground
4	CAN3_L	Input / Output
5	KL.15 (Ignition)	Input
6	CAN1_GND	CAN1 Ground
7	CAN1_GND	CAN1 Ground
8	DIGITAL_IN4_GND	Ground

Preliminary

9	DIGITAL_IN3	Input
10	CAN1_TR (External activation of Termination resistor)	Input / Output
11	Not Connected	--
12	CAN2_L DaisyChain	Input / Output
13	CAN2_H DaisyChain	Input / Output
14	LED2	Output
15	Not Connected	--
16	Not Connected	--
17	LS_SWITCH_1_GND	Ground
18	LS_SWITCH_1	Output
19	KL.31 (GND_Z)	Main Ground
20	KL.31 (GND_Z)	Main Ground
21	CAN3_H	Input / Output
22	CAN2_GND	Ground
23	CAN2_GND	Ground
24	CAN1_H	Input / Output
25	DIGITAL_IN5_GND	Ground
26	DIGITAL_IN4	Input
27	DIGITAL_IN2_GND	Ground
28	DIGITAL_IN1	Input
29	Not Connected	--
30	Not Connected	--
31	LED1	Output
32	Not Connected	--

Preliminary

33	Not Connected	--
34	Not Connected	--
35	LS_SWITCH_2_GND	Ground
36	LS_SWITCH_2	Output
37	KL.31 (GND_Z)	Main Ground
38	KL.31 (GND_Z)	Main Ground
39	CAN3_GND	Ground
40	CAN2_H Main	Input / Output
41	CAN2_L Main	Input / Output
42	CAN1_L	Input / Output
43	DIGITAL_IN5	Input
44	DIGITAL_IN3_GND	Ground
45	DIGITAL_IN2	Input
46	DIGITAL_IN1_GND	Ground
47	Not Connected	--
48	Not Connected	--
49	Not Connected	--
50	Not Connected	--
51	Not Connected	--
52	Not Connected	--
53	Not Connected	--
54	Not Connected	--

Preliminary

**CAN1 Termination**

If a termination resistor is required for CAN1, please connect/bridge PIN 10 (CAN1_TR) to PIN 24 (CAN1_H).

11.6.2 Diagnostic Trouble Codes - C-Hub TCU2 device

DTC (Hex)	Error text
0x09F1EC	GNSS Module error
0x0CF1EC	GNSS Jamming detected
0x63F2EC	IMU Error
0x64F2E0	System temperature - Critical High
0x64F2E1	System temperature - Critical Low
0x6DF1E5	Cellular Network Antenna 2 - Open Circuit
0x6DF1E6	Cellular Network Antenna 2 - Short to GND
0x6DF2EC	Modem Module error
0x6DF3EC	SIM Card cannot be used
0x7F0209	CAN1 - Bus off
0x8C1E05	GNSS Antenna - Open circuit
0x8C1E06	GNSS Antenna - Short to GND
0x8E1E05	Cellular Network Antenna 1 - Open Circuit
0x8E1E06	Cellular Network Antenna 1 - Short to GND

DTC (Hex)	Error text
0xA80003	Supply voltage - Critical high
0xA80004	Supply voltage - Critical low
0xAD210C	NAD Secure Boot failure
0xCF0409	CAN2 - Bus off
0xD1F0E5	BT/WLAN Antenna 1 - Open circuit
0xD1F0E6	BT/WLAN Antenna 1 - Short to GND
0xD1F0EC	USB Error
0xD1F1E5	BT/WLAN Antenna 2 - Open circuit
0xD1F1E6	BT/WLAN Antenna 2 - Short to GND
0xD1F2EC	BT/WLAN Module error
0xD30409	CAN3 - Bus off
0xFDF1E2	SW Update - General failure
0xFDF2E2	SW Update - Signature verification failure

Preliminary

Imprint

Name and address

Robert Bosch GmbH
Robert-Bosch-Platz 1
70839 Gerlingen-Schillerhöhe
GERMANY

Board of management

Dr. Christian Fischer, Dr. Markus Forschner, Stefan Grosch,
Dr. Markus Heyn, Dr. Frank Meyer, Katja von Raven, Dr. Tanja Rückert

Your contact at Bosch

kontakt@bosch.de
+49 711 400 40990
Corporate headquarters: +49 711 811-0

Registration court

District court Stuttgart HRB 14000

VAT identification number

DE811128135

Automotive Connectivity Hub

<https://www.bosch-mobility.com/en/solutions/connectivity/automotive-connectivity-hub/>

