# LARA-R6 series

## Internet applications development guide

Application note

### Abstract

This document provides detailed examples of how to use AT commands to develop IP applications in LARA-R6 series modules.



UBX-22001854 - R02
C1-Public

# Document information

| | | |
|---|---|---|
| **Title** | **LARA-R6 series** | |
| **Subtitle** | Internet applications development guide | |
| **Document type** | Application note | |
| **Document number** | UBX-22001854 | |
| **Revision and date** | R02 | 23-Feb-2024 |
| **Disclosure restriction** | C1-Public | |

This document applies to the following products:

| Product name | Notes |
|---|---|
| LARA-R6 series | |

u-blox or third parties may hold intellectual property rights in the products, names, logos and designs included in this document. Copying, reproduction, modification or disclosure to third parties of this document or any part thereof is only permitted with the express written permission of u-blox.
The information contained herein is provided “as is” and u-blox assumes no liability for its use. No warranty, either express or implied, is given, including but not limited to, with respect to the accuracy, correctness, reliability and fitness for a particular purpose of the information. This document may be revised by u-blox at any time without notice. For the most recent documents, visit www.u-blox.com.
Copyright © u-blox AG.

# Contents

**Document information** ..... **2**
**Contents** ..... **3**
**1 Introduction** ..... **6**
**2 PS data connection** ..... **8**
2.1 PDP contexts ..... 8
2.2 MTU configuration ..... 9
2.2.1 MTU in IPv4 ..... 9
2.2.2 MTU in IPv6 ..... 9
2.3 Socket and PDP context activation ..... 9
2.3.1 Default PDP and preferred protocol type configuration ..... 9
2.4 Other relevant AT commands ..... 11
2.4.1 UPING ..... 11
2.4.2 DNS resolution ..... 12
2.5 IPv4 Translation for T-Mobile US IPv6 data connection ..... 13
**3 Data security** ..... **14**
3.1 Certificates manager +USECMNG ..... 14
3.2 Profile configuration +USECPRF ..... 14
3.2.1 Cipher suites ..... 15
3.3 Complete example ..... 16
3.4 (D)TLS session resumption ..... 17
3.4.1 Session resumption with session ID ..... 18
3.4.2 Session resumption with session ticket ..... 18
3.4.3 Session resumption with PSK-based session ticket ..... 19
3.4.4 (D)TLS session resumption examples ..... 19
3.5 Troubleshooting secure connection ..... 28
**4 Dial-up network (PPP)** ..... **29**
4.1 Basic setup ..... 29
4.1.1 Dial-up configuration ..... 30
4.1.2 PPP and IPv6 ..... 30
4.1.3 PPP over multiple PDP contexts ..... 30
4.2 Terminate cellular packet data connection ..... 31
4.3 Port filtering feature for embedded IP applications ..... 31
4.3.1 Example: +UEMBPF with PPP & LwM2M client handling ..... 32
4.3.2 +UEMBPF not configured ..... 33
**5 TCP/UDP internal stack** ..... **34**
5.1 Socket connect ..... 34
5.2 Socket listening ..... 34
5.3 Socket write (+USOWR) ..... 35
5.3.1 Binary mode ..... 35
5.3.2 Base syntax ..... 35

5.3.3 Queue FULL ....36
5.4 Socket operations with "Keep Alive" option ....36
5.5 Socket read (+USORD) ....37
5.6 Socket write (+USOST) ....38
5.7 Socket read (+USORF) ....39
5.8 Socket state ....39
5.9 Socket close ....40
5.10 Testing sockets ....40
5.11 Secure socket ....41
**6 MQTT .... 42**
6.1 Basic setup ....42
6.1.1 Default and minimal configuration ....42
6.1.2 Last will configuration ....42
6.1.3 Profile management ....42
6.1.4 Internal PDP context mapping ....43
6.2 Start and end a MQTT session ....43
6.3 Subscribe to a topic and publish a message to the same topic ....43
6.4 Publish a message with hexadecimal mode set ....44
6.5 Publish a binary message to a topic ....45
6.6 Ping the MQTT broker ....45
6.7 Last will packet ....45
6.8 Debug ....46
6.9 Secure MQTT ....46
**7 MQTT-SN .... 47**
7.1 Basic setup ....47
7.1.1 Default and minimal configuration ....47
7.1.2 Last will configuration ....47
7.1.3 Profile management ....47
7.1.4 Internal PDP context mapping ....48
7.2 Start and end a MQTT-SN session ....48
7.3 Subscribe to a normal topic ....48
7.4 Publish and read a message to a topic ....49
7.5 Unsubscribe from a normal topic ....49
7.6 Register to a topic and publish a message to the same topic ....49
7.7 Subscribe to a short topic name and publish a message to the same topic ....49
7.8 Last will ....50
7.9 Error handling ....50
7.10 Secure MQTT-SN ....50
7.11 MQTT Anywhere ....51
7.12 MQTT Flex ....51
7.12.1 Specify the client key and certificate to be used ....52
7.12.2 Configure the DTLS security profile ....52

7.12.3 Configure the MQTT-SN client ....52
**8 CoAP .... 53**
8.1 Good practices on writing CoAP application ....53
8.2 Basic setup ....54
8.2.1 Current configuration ....54
8.2.2 Profile management ....54
8.2.3 Internal PDP context mapping ....55
8.3 Basic operation ....55
8.3.1 GET ....55
8.3.2 PUT/POST ....55
8.3.3 Block transfer ....56
8.3.4 TCP connection ....56
8.4 Error handling ....56
8.4.1 Configuration error ....56
8.4.2 Error on server reply ....56
8.5 Secure connection ....57
8.5.1 Basic configuration ....57
8.5.2 DTLS connection ....57
8.5.3 TLS connection ....57
**9 FTP .... 58**
9.1 Direct link ....59
9.1.1 Retrieve a file from FTP server ....59
9.1.2 Aborting retrieve file request ....60
9.1.3 Store a file on FTP server ....60
9.1.4 About "+++" escape sequence use ....61
9.2 Using secure option ....61
9.3 Error handling ....62
**10 HTTP .... 63**
10.1 Basic setup ....63
10.2 HTTP POST ....64
10.3 Error handling ....65
10.4 Secure HTTP ....65
**11 LwM2M .... 65**
**Appendix .... 66**
**A Glossary .... 66**
**Related documentation .... 68**
**Revision history .... 68**
**Contact .... 68**

# 1 Introduction

This document provides guidance for developing applications based on the internet protocol (IP) that use LARA-R6 series modules. It includes examples of AT commands to interface with the u-blox cellular modules for network connectivity and IP protocols use. It gives examples of applications relying on the IP stack (sockets, MQTT, MQTT-SN, HTTP, FTP, CoAP, SSL/TLS).

Sections 2 and 3 describe the packet switched (PS) data connection with the context definition and procedure to obtain a valid IP address from the network. Then, it provides information on security aspects useful to manage and configure a secure data connection.

Table 1 shows a summary of the documentation available for u-blox cellular modules. We recommend, as a starting point, to read the application development guide app note [5], which has highly relevant guidelines for developing applications that interface with u-blox cellular modules. Moreover, it contains details to complete the network registration process, which is a mandatory precondition to activate a PS data connection and use any internet application.

| | Document name | Notes |
|---|---|---|
| **Application integration** | Application development guide app note [5] | This should be the first document to read when working on an application for cellular modules. |
| | FW update app note [8] | FW update procedures (uFOTA, FOAT, FOTA, +UFWINSTALL, EasyFlash). |
| | Internet applications development guide app note | **This document.** |
| | Positioning implementation app note [9] | Implementation of the GNSS interface, hybrid positioning and timing information in u-blox cellular modules. |
| | LwM2M objects and commands app note [6] | How to use LwM2M protocol stack to interact with the LwM2M server. |
| | Mux implementation app note [7] | Usage of multiplexer with cellular modules. |
| | EVK-R6 user guide [4] | Starting guide for the evaluation kit. |
| **Reference documentation** | Data sheet [1] | Use these documents as hardware integration and AT commands API reference. |
| | System integration manual [2] | |
| | AT commands manual [3] | |
| **Product release documents** | Sample delivery note / Information note | Delivered with every FW release. |

**Table 1: LARA-R6 product documentation overview**

From section 3.5 on, the document provides examples of internet-related applications built with the LARA-R6 series modules. The modular structure of these applications is shown in Figure 1.

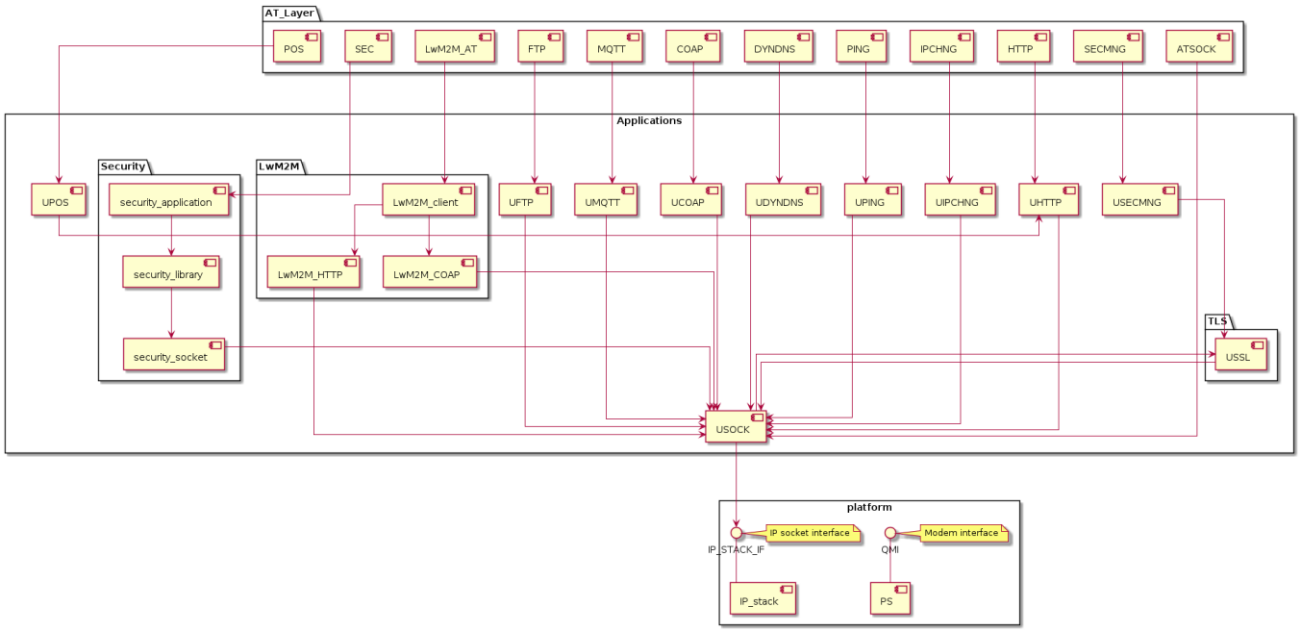



**Figure 1: LARA-R6 "IP" applications**

As additional reference documentation, see the data sheet [1], the system integration manual [2], and the AT commands manual [3] for a detailed AT command descriptions.

The following symbols are used to highlight important information within this document:

☞ An index finger points out key information pertaining to module integration and performance.

⚠ A warning symbol indicates actions that could negatively impact or damage the module.

# 2 PS data connection

Ensure the module is correctly registered to the network before executing any procedure or example shown in this document. Steps to complete the network registration operation can be found in the "Network registration" section of the application development guide [5].

## 2.1 PDP contexts

Packet-switched services rely on the packet data protocol (PDP). The PDP context is a data structure that contains the subscriber's session information. Two types of PDP context are defined:

- "External" PDP context: IP packets are built by the Data Terminal Equipment (DTE), the module's IP instance runs the IP relay function only.
- "Internal" PDP context, or PSD profile: the PDP context (relying on the module's embedded TCP/IP stack) is configured, established, and handled via the data connection management AT commands.

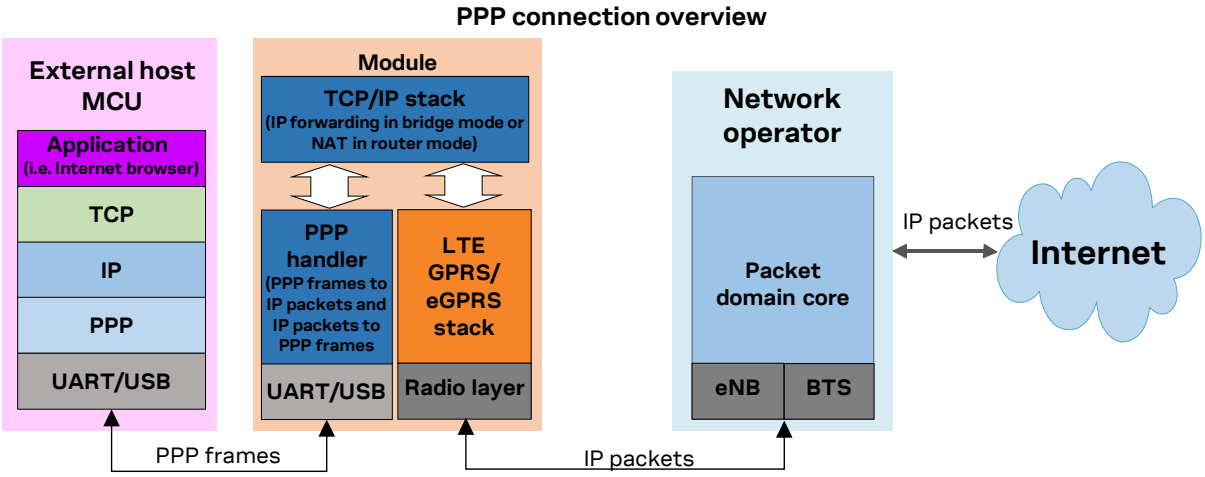



**Figure 2: Example of external context structure**

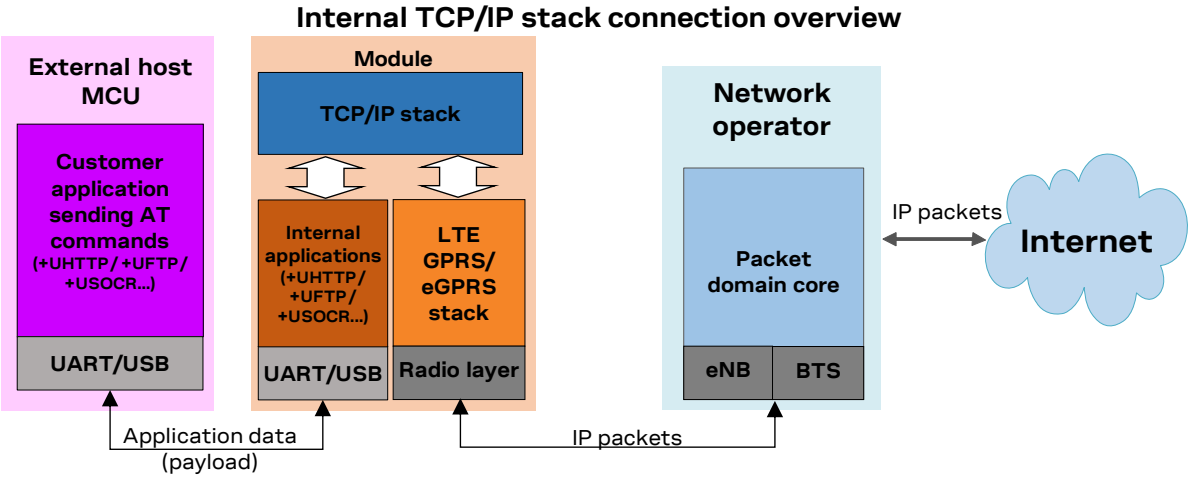



**Figure 3: Example of internal context structure**

Networks offer connectivity to different IP domains (internet or private intranet) selected by specifying the Access Point Name (APN) at PDP context activation.

☞ In LTE RAT, if the access point name (APN) is not specified, an anchor APN (e.g., “admin”) can be assigned by the network to the module, with an IP address, which may give no actual connectivity. Check the APN to use with your mobile network operator.

For further details on the APN configuration, see the “Network registration” section of the application development guide [5].

Each PDP context defined has a related identifier called “context ID” (CID). The <cid>=1 is mapped to the initial default EPS bearer (primary PDP context, established during the LTE attach procedure).

The maximum number of primary PDP contexts that can be activated is 5. At most 7 secondary PDP contexts may be associated to a primary PDP context. The total number of active PDP contexts, both primary and secondary, is at most 8.

## 2.2 MTU configuration

### 2.2.1 MTU in IPv4

The maximum transmission unit (MTU) configuration is stored in the MNO profiles and its configuration depends on specific network operator requirements. The configuration modes can be:

- Fixed value.
- In the protocol configuration options (PCO), set the MTU size during PDP activation and use the network assigned value. If the network does not assign any value, the fixed value is used as a fallback.

☞ The default values vary depending on the current MNO profile loaded via the +UMNOPROF AT command.

### 2.2.2 MTU in IPv6

The MTU configuration is stored in the MNO profiles and its configuration depends on specific network operator requirements. The configuration modes can be:

- Fixed value.
- From the Router Advertisement during SLAAC. If the MTU is not present in the RA, the fixed value is used as a fallback.

☞ The default values vary depending on the current MNO profile loaded via +UMNOPROF AT command.

## 2.3 Socket and PDP context activation

Starting from the power-up of the modules and the control of the pre-existing settings, the following example shows all the AT commands necessary to reach the activation of a PDP context at first and then a data socket.

### 2.3.1 Default PDP and preferred protocol type configuration

It’s possible to configure a different default PDP context ID and the preferred IP type with the +UDCONF=19 AT command. A reboot of the module is necessary to make the change effective and new configuration is stored on NVM. If not specified otherwise, these parameters are used by internal applications that require IP connectivity, e.g., MQTT and HTTP protocols.

Below is an example of the +UDCONF=19 AT command.

| Command | Response | Description |
|---|---|---|
| AT+UDCONF=19,2,0 | OK | Configures the default PDP context <cid>=2 and the preferred IP type (0: IPv4) to be used by AT commands that require connectivity. |
| AT+CFUN=16 | OK | Reboot of the module to make the new setting effective. |

☞ Embedded TCP/UDP IP clients and internet suite clients are automatically mapped to the CID 1 (initial default EPS bearer).

☞ When using the Verizon profile (+UMNOPROF=3) in VZW HPLMN, the <cid>=1 is reserved for IMS (APN class 1) while the PDN connection shall be mapped to the <cid>=3 (APN class 3). If the Verizon profile is set, the preferred PDP default context is automatically configured to <cid>=3.

☞ For LARA-R6401D modules, in roaming condition, the PDN connection is maintained on <cid>=3 (APN class 3) with IPv4-only type. While the <cid>=1 usually used for IMS (APN class 1) is not activated.

☞ For LARA-R6401 and LARA-R6401D modules, any active MNO profile change by +UMNOPROF AT command will restore to the factory-programmed configuration for the specified profile.

☞ For LARA-R6001D "00B" product version, the +UDCONF=19 AT command is not supported.

### Network settings verification

| Command | Response | Description |
|---|---|---|
| AT+CFUN=0 | OK | Turn off radio functionality.<br>☞ While setting the network profile and parameters, the radio functionality must be turned off. |
| AT+UMNOPROF=100 | OK | Set the MNO profile for usage in Europe (LTE bands 3, 8, 20). The MNO profiles are sets of modem configurations specific for MNO/regions. This setting is stored in NVM and its configuration shall be performed only at the first boot or after the module is flashed. |
| AT+CFUN=16 | OK | Reboot the module to make the new setting effective. |
| AT+UMNOPROF? | +UMNOPROF: 100<br>OK | Verify that the new profile has been set. |
| AT+CFUN=0 | OK | Turn off radio functionality. |
| AT+CGDCONT=1,"IPV4V6","apn_name" | OK | Define the PDP context 1 with PDP type "IPV4V6" and APN "apn_name" of the MNO. |

### Check network registration: LTE radio access technology

| Command | Response | Description |
|---|---|---|
| AT+CFUN=1 | OK | Turn on radio functionality. |
| AT+COPS? | +COPS: 0,0,"I TIM",7<br>OK | Verify if module is currently registered to the network.<br>☞ Issue AT+COPS=0 if the +COPS read command returns +COPS: 2. |
| AT+CGDCONT? | +CGDCONT: 1,"IP","apn_name","10.40.50.500",0,0,0,0,0<br>OK | Return IPv4 address (in this case, only IPv4 address is assigned by the network). |
| AT+CSCON=1 | OK | Enabled URC that returns details of the current terminal's perceived radio connection status. |

| Command | Response | Description |
|---|---|---|
| | | ☞ CSCON functionalities are available only on LTE radio access technology. |

**Check network registration: GSM/GPRS and UMTS/HSPA radio access technologies**

| Command | Response | Description |
|---|---|---|
| AT+CFUN=1 | OK | Turn on radio functionality. |
| AT+COPS? | +COPS: 0,0,"I TIM",2<br>OK | Verify if module is currently registered to the network.<br>☞ Issue AT+COPS=0 if the +COPS read command returns +COPS: 2.<br>In this case <act>=2 means the module is registered on UTRAN. |
| AT+CGACT=1,1 | OK | Activate PDP context 1. |
| AT+CGDCONT? | +CGDCONT: 1,"IP","apn_name","100.108.232.233",0,0,0,0,0,0<br>OK | Return IPv4 address (in this case, only IPv4 address is assigned by the network). |

☞ In GSM/GPRS and UMTS/HSPA radio access technologies, it is recommended to always issue the AT+CGACT=1,<cid> command when the application processor requires PDP at <cid>, even if the PDP context is already active. IP embedded applications (such as BIP, USEC or LwM2M) may activate it and if the PDP is used by the external processor without AT+CGACT=1,<cid> the PDP may be de-activated without any notification and the external application will lose connectivity.

☞ In 2G and 3G (legacy RATs), the IP address is maintained if 2G/3G is reselected from LTE.

☞ After PDN activation the AT+CGDCONT and AT+CGDCONTRDP read commands return the user-configured APN, not the one assigned by the network.

## 2.4 Other relevant AT commands

### 2.4.1 UPING

The ping command finds out if a remote host is reachable on the internet, and checks if the module connectivity is still available.

The ping functionality is based on the ICMP protocol. The ping command sends an ICMP echo request to the remote host and waits for its ICMP echo reply. If the echo reply packet is not received, the remote host might be unreachable. The ping command could be used to measure the round-trip time (RTT, the time needed by a packet to go to the remote host and come back) and the time to live (TTL, which is a value to understand how many gateways a packet has gone through).

The +UPING AT command allows the user to execute a ping command from the module to a remote peer. The results of the ping command execution are notified by these URCs:

- +UUPING: returns the +UPING AT command result when no error occurred.
- +UUPINGER: raised if an error occurs while processing the +UPING AT command.

For further details on the +UPING AT command and its possible errors, see the AT commands manual [3].

⚠ Some network operators may disallow ICMP packets traffic on their network, this means that the +UPING AT command may not work.

⚠ Some remote hosts might not reply to ICMP echo requests for security reasons (e.g., firewall settings).

⚠ Some remote hosts might not reply to ICMP echo requests if the data size of the echo request is too big.

⚠ If a remote peer does not reply to an ICMP echo request, it does not mean that for sure the peer cannot be reached in another way.

☞ The command is not supported in context using the IPv6 protocol type.

| Command | Response | Description |
|---|---|---|
| AT+UPING="www.google.com" | OK | Ping request. |
| | +UUPING: 1,32,"www.google.com","216.58.206.68",115,62<br>+UUPING: 2,32,"www.google.com","216.58.206.68",115,53<br>+UUPING: 3,32,"www.google.com","216.58.206.68",115,53<br>+UUPING: 4,32,"www.google.com","216.58.206.68",115,53 | URC ping responses. |

☞ If the +UUPING URC reports <rtt>=-1, then the timeout is elapsed (no response received).

☞ If the first +UUPING URC reports <rtt>=-2, then the TTL used in the ping request is too low.

☞ Some network operators may return an ICMP time exceeded message when the remote host is not reachable. In those cases, the first +UUPING URC reports <rtt>=-1 and the subsequent +UUPING URC reports <rtt>=-2.

### 2.4.2 DNS resolution

The +UDNSRN AT command translates a domain name to an IP address, or an IP address to a domain name by using an available DNS. There are two available DNSs, primary and secondary. The network usually provides them after a PS data activation. They are automatically used in the resolution process if available. The resolver first uses the primary DNS, if no answer, it uses the second DNS.

| Command | Response | Description |
|---|---|---|
| AT+UDNSRN=0,"www.google.com" | +UDNSRN: "216.239.59.147"<br>OK | DNS resolution request. |

☞ If the application is not subjected to low power consumption constraints, it is suggested to use either the +UPING or the +UDNSRN AT command to verify that the module is registered with the network, and a PS data connection is activated before start using any IP application.

#### 2.4.2.1 Override DNS configuration

The +UDNSCFG AT command overrides the primary and/or the secondary DNS defined for a selected context CID.

| Command | Response | Description |
|---|---|---|
| AT+UDNSCFG=1,1,0,"8.8.8.8" | OK | Override with IP “8.8.8.8” (thus IPv4) the primary DNS of context ID 1. |

## 2.5 IPv4 Translation for T-Mobile US IPv6 data connection

LARA-R6 series modules provide functionality to translate IPv4 type IP traffic for IPv6-only network (464XLAT). This functionality is disabled by default. This is relevant for UEs willing to perform IPv4 traffic from application and operating in T-Mobile US network which may provide IPv6-only data traffic, depending on the APN and SIM subscription.

By default, UE is configured with IPV4V6 IP type request when performing a data connection in T-Mobile US networks. However, once module is attached and registered to the LTE network, the application must check the output of the AT+CGDCONT read command.

- If the dual stack IPV4V6 is confirmed by the network, no further action is required.
- Otherwise, if the IPV6 protocol type is selected by the network, it is recommended to enable the XLAT functionality to operate on T-Mobile with the following sequence:
  - AT+CFUN=0
  - AT+UDCONF=66,4,1
  - AT+CFUN=16
  - Module initialization commands.

☞ Every time the module switches to a roaming network and an IPV4 is obtained, XLAT functionality must be disabled by reverting the above configuration. The XLAT functionality is supported only when in T-Mobile network.

# 3 Data security

## 3.1 Certificates manager +USECMNG

The +USECMNG AT command enables managing SSL/TLS certificates and private keys. Particularly, the command is used to:

- Import certificates and private keys
- List and retrieve information of imported certificates and private keys
- Remove certificates and private keys
- Calculate MD5 hash for imported certificate or private key

For additional details on this AT command, the number and the format of the certificates, and the private keys accepted, see the AT commands manual [3].

☞ The SSL/(D)TLS connection with server and/or mutual authentication can be successfully performed using the following key size:

- for Rivest-Shamir-Adleman (RSA) keys at least 2048 bits
- for Elliptic Curve Digital Signature Algorithm (ECDSA) keys at least 192 bits

The same limitation is also applied to the keys used in the generation of certificates.

The following example shows the use of the +USECMNG AT command to perform a mutual authentication using certification authority (CA) certificate, client certificate, and client private key.

| Command | Response | Description |
|---|---|---|
| `AT+USECMNG=1,0,"ca_cert","ca_cert.crt"` | `+USECMNG: 1,0,"ca_cert","d10137cee624fcee624418db5eaa"`<br>`OK` | Import the CA certificate from the "ca_cert.crt" file stored on the file system. |
| `AT+USECMNG=1,1,"client_cert","client_cert.crt"` | `+USECMNG: 1,1,"client_cert","b137ce137ce5edd6723d8b13"`<br>`OK` | Import the client certificate from the "client_cert.crt" file stored on the file system. |
| `AT+USECMNG=1,2,"client_key","client_key.key"` | `+USECMNG: 1,2,"client_key","087ab34c9aa03fbce5edd6723d8b8e05"`<br>`OK` | Import the client private key from the "client_key.key" file stored on the file system. |
| `AT+USECMNG=3` | `CA,"ca_cert","An MQTT broker","2032/10/18 08:23:32"`<br>`CC,"client_cert","A client certificate","2032/06/22 12:34:48"`<br>`PK,"client_key"`<br>`OK` | List all imported certificates or private keys. |

## 3.2 Profile configuration +USECPRF

The +USECPRF AT command allows the configuration of USECMNG (u-blox SECurity MaNaGement) profiles used for an SSL/TLS/DTLS connection.

In particular, the command manages security profiles for the configuration of the following SSL/TLS/DTLS connections properties:

- Certificate validation level
- Minimum SSL/(D)TLS version

- Cipher suites to be proposed: legacy, IANA nomenclature, list of cipher suites
- Certificate to be used for server and mutual authentication
- Expected server hostname, when using certificate validation level 1, 2 or 3
- Password for the client private key if it is password protected
- Pre-shared key used for connection
- Server name indication (SNI)
- Server certificate pinning
- Pre-shared key generated in production
- (D)TLS session resumption.

For additional details on this AT command and all the related configurations, see the AT commands manual [3].

| Command | Response | Description |
|---|---|---|
| `AT+USECPRF=0` | `OK` | Reset (set to factory-programmed value) all the parameters of security profile #0.<br>☞ We recommend issuing the reset as the first command to erase all previously stored values. |
| `AT+USECPRF=0,0,1` | `OK` | Enable certificate validation without URL integrity check for profile #0. The server certificate will be verified with a specific trusted certificate or with each of the imported trusted root certificates. |
| `AT+USECPRF=0,2,3` | `OK` | Select legacy cipher suite for profile #0. |
| `AT+USECPRF=0,3,"ca_cert"` | `OK` | Select trusted root certificate internal name for profile #0. |
| `AT+USECPRF=0,5,"client_cert"` | `OK` | Select trusted client certificate internal name for profile #0. |
| `AT+USECPRF=0,6,"client_key"` | `OK` | Select trusted client key internal name for profile #0. |
| `AT+USECPRF=0,10,"<SNI_address>"` | `OK` | Configure the server name indication.<br>☞ Some servers require this configuration to correctly perform the secure connection. |

### 3.2.1 Cipher suites

A cipher suite is a set of algorithms and protocols used in the SSL/(D)TLS handshake to negotiate the security setting for the secure connection. The cipher suite for the TLS protocol mainly consists of:

- Key Exchange Algorithm: determines the way symmetric keys are exchanged (RSA, DH, ECDH, DHE, ECDHE, PSK).
- Authentication/ Digital Signature Algorithm: determines how server authentication and client authentication (if required) are performed (RSA, ECDSA, DSA, etc.).
- Bulk Data Encryption: determines which symmetric key algorithm is used to encrypt the actual data (AES, CHACHA20, Camellia, ARIA, etc.). The Bulk Data Encryption is defined by an algorithm, his strength, and operating mode (block cipher mode or stream cipher mode).
- Message Authentication Code (MAC) algorithm: Determines the method that the connection should use to perform data integrity checks (SHA, SHA-256, SHA-384, POLY1305, etc.). Hash-Based Message Authentication Code (HMAC) is used.

A cipher suite is defined by a string representing a named combination of the algorithms and protocol:

TLS_{ Key Exchange }_{ Authentication/ Digital Signature }_WITH_{ Bulk Data Encryption }_{ Message Authentication Code }

As an example, for the TLS 1.0, TLS 1.1, and TLS 1.2 protocols, the following paragraph shows each part of the cipher suite string **TLS_RSA_WITH_AES_256_CBC_SHA:**

- Key Exchange Algorithm: **RSA**.
- Bulk Data Encryption: **AES_256_CBC**.
- Message Authentication Code (MAC) Algorithm: **SHA**.

The Authenticated Encryption with Associated Data (AEAD) bulk ciphers can perform authentication and encryption of the message. For the AEAD bulk ciphers in the string representation the Bulk Data Encryption part and Message Authentication Code part are merged.

☞ If the remote server does not support one of these cipher suites selected in the security profile settings, the handshake fails, and module will be unable to connect to the server.

#### 3.2.1.1 TLS 1.3 cipher suites

The TLS 1.3 protocol introduces several changes to increase the security level. In comparison to the TLS 1.2, it has mainly:

- eliminated Block ciphers (CBC) and non-AEAD ciphers
- dropped the support for older SSL ciphers (DSA, SHA, Weak Elliptic Curves, RSA Key Exchange, Static Diffie-Hellman (DH, ECDH))

In the TLS 1.3:

- RSA or ECDSA are used as authentication/ digital signature algorithms
- DHE or ECDHE are used as a key exchange mechanism.

The string representing the cipher suite for the TLS 1.3 protocols has been reduced as follows:

TLS_{ Bulk Data Encryption }_{ Message Authentication Code }

The Bulk Data Encryption uses an AEAD cipher while the Message Authentication Code, in particular the Hash-based Message Authentication Code used in the previous TLS versions, has been substituted by the HMAC-based Key Derivation Functions (HKDF).

As an example, for the TLS 1.3 protocol, the following paragraph shows each part of the cipher suite string **TLS_AES_256_GCM_SHA384**:

- Bulk Data Encryption: **AES_256_GCM** - AEAD bulk cipher
- Message Authentication Code: **SHA384** – HKDF

A list of available cipher suite is maintained in the “Cipher suite applicability” section of the AT commands manual [3].

## 3.3 Complete example

| Command | Response | Description |
|---|---|---|
| **Step1: Import a trusted root certificate using the byte stream similar to the +UDWNFILE AT command** | | |
| `AT+USECMNG=0,0,"ThawteCA",1516` | `>` | Start the data transfer using the stream of byte.<br>☞ Unlike the example in section 3.1, here the certificate is transferred as a byte stream and is not stored in the LARA-R6 file system. |
| `-----BEGIN CERTIFICATE-----`<br>`MIIEIDCCAwigAwIBAgIQNE7VVyDV7ex`<br>`J9C/OjVaMaA==`<br>`-----END CERTIFICATE-----` | `+USECMNG: 1,0,"ThawteCA",`<br>`"8ccadc0b22cef5be72ac411a`<br>`11a8d812"`<br>`OK` | Input PEM formatted trusted root certificate data bytes. Output MD5hash string of the stored trusted root certificate DER. |

| Command | Response | Description |
|---|---|---|
| **Step 2: List all available certificates and private key** | | |
| AT+USECMNG=3 | CA, "ThawteCA","thawte Primary Root CA","2036/07/17"<br>OK | List all available certificates and private keys. |
| **Step 3: Set the security profile 2 validation level to a trusted root** | | |
| AT+USECPRF=2,0,1 | OK | Security profile 2 has the validation level set to a trusted root. |
| **Step 4: Set the security profile 2 trusted root certificate to the CA certificate imported as "ThawteCA"** | | |
| AT+USECPRF=2,3,"ThawteCA" | OK | Security profile 2 will use the CA certificate imported as "ThawteCA" for server certificate validation. |
| **Step 5: Use the configured USECMNG profile 2 with the UHTTP application** | | |
| AT+UHTTP=0,1,"www.ssl_tls_test_server.com" | OK | Configure the UHTTP server name. |
| AT+UHTTP=0,6,1,2 | OK | Enable the SSL/TLS for the UHTTP profile #0 and specify the SSL/TLS security profile #2. |
| AT+UHTTPC=0,1,"/","https.resp" | OK | Execute the HTTP GET command. |
| | +UUHTTPCR: 0,1,1 | HTTP GET URC response. |

☞ Due to the significant memory fingerprint of an SSL/TLS connection, the number of concurrent SSL/TLS connections is limited. The +USECMNG AT command and the underlying SSL/TLS infrastructure allows 4 concurrent SSL/TLS connections (i.e., 4 HTTPS requests or 2 HTTPS and 2 FTPS requests).

## 3.4 (D)TLS session resumption

This section gives details and examples on the use of the (D)TLS session resumption feature, a useful approach that speeds up the handshake negotiation process.

The session resumption allows the caching of TLS/DTLS session information and hence can be used to shorten the handshake procedure when consequential sessions must be established with the same server. The RFC 5246 and RFC 5077 specification [17][15] of the session resumption provides 2 concepts:

- **Session ID**: the connection properties (IP address /port) in the resumed session need to be the same as in the suspended session. The server in this case has a cache where tracks the IP address /port /session ID. This concept is also called server-side stateful session resumption in the sense that the server should keep a per-client session state.
- **Session Ticket**: the connection properties (IP address /port) do not need to be the same. In this case, the module needs to keep the *session ticket* so for the server there is less work. This concept is also called server-side stateless session resumption and does not require the server to keep the per-client session state. This allows servers to handle many transactions from different users, the sessions can be cached for a long time, load balancing of the requests can be performed across different servers, and the possibility to run server instances on an embedded platform with little memory.

The TLS 1.3 protocol [18] offers an alternative of the session resumption with session ID and session ticket present in TLS 1.2 called:

- **PSK-based session ticket**: this session resumption mechanism is similar to the TLS 1.2 session ticket. The client is requesting the server a "new ticket" and the server replies with a session ticket that includes the server session state and resumption PSK.

In general terms, the session resumption is performed within the following steps:

1. Acquiring the session data
2. Reusing the session data

☞ The session resumption feature configuration and secure session data are not stored in the NVM, and the session data is displayed via URC. Therefore, the session can be restored after waking up from PSM mode or a module reboot only if the session data is stored by the user application.

### 3.4.1 Session resumption with session ID

For the session resumption with session ID these two steps can be summarized as following:

1. Acquiring the session ID:
   - The client sends in the Client Hello message the session ID length equal to 0.
   - The server responds in the Server Hello message with the session ID and the session ID length (not equal to 0).
   - The client should store the session ID and the session ID length to reuse them.
   - The server should store the session information (session ID, session ID length, connection properties) to accept session resumption from the client.
2. reusing previous session:
   - The client sends in the Client Hello message the session ID and session ID length (not equal to 0).
   - The server responds in the Server Hello message with the same session ID and session ID length if it accepts the session resumption. In this case, no certificate or key information is exchanged during abbreviated TLS handshake and previously negotiated keys are re-used. If the previous session is not accepted the server replies in the Server Hello with a new session ID and session ID length, and then certificates and new session keys must be exchanged.

Since the session resumption is enabled, the URC will be displayed every time a secure connection is performed. Once the session data of the URC are not equal to the session data already set, it means that the server does not accept the session resumption. In this case, the complete handshake has been performed and the new session data has been displayed. The new session data needs to be set again.

### 3.4.2 Session resumption with session ticket

For the session resumption with session ticket these two steps can be summarized as following:

1. Acquiring the session ID:
   - The client sends an empty Session Ticket TLS extension in the Client Hello message.
   - The server responds with an empty Session Ticket TLS extension in the Server Hello message to indicate to the client that it will send a new session ticket within the NewSessionTicket handshake message.
   - Further handshake messages between the client and the server are exchanged.
   - After the client's Finished message and before the ChangeCipherSpec message, the server stores its session state (ciphersuite and master secret) to a ticket that is sent to the client using the NewSessionTicket TLS handshake message.
   - The client should store the session ticket to reuse it.

2. Reusing previous session:
   - The client sends in the Client Hello message including the session ticket in the SessionTicket extension.
   - The server retrieves the session state from the content of the received ticket to resume the session.
   - Further messages are exchanged to complete the reduced handshake procedure

Since the session resumption is enabled, the session resumption status URC will be displayed every time a secure connection is performed. The session resumption status will be set to “configured” (2) once the session data have been obtained, otherwise the status will remain to “enabled” (1).

### 3.4.3 Session resumption with PSK-based session ticket

For the session resumption with PSK-based session ticket these two steps can be summarized as following:

1. Acquiring the session ID:
   - The client sends the Client Hello message.
   - The server responds with "pre_shared_key" extension the Server Hello message to indicate to the client that it will send a new session ticket using the NewSessionTicket handshake message.
   - Further handshake messages between the client and the server are exchanged.
   - After the client's Finished message and before the ChangeCipherSpec message, the server stores its session state and resumption PSK to a ticket that is sent to the client within the NewSessionTicket TLS handshake message.
   - The client should store the session ticket to reuse it.
2. Reusing previous session:
   - The client sends in the Client Hello message including the PSK-based session ticket in the "pre_shared_key" extension.
   - The server retrieves the session state from the content of the received ticket to resume the session.
   - Further messages are exchanged to complete the reduced handshake procedure.

Since the session resumption is enabled, the session resumption status URC will be displayed every time a secure connection is performed. The session resumption status will be set to “configured” (2) once the session data have been obtained, otherwise the status will remain to “enabled” (1).

### 3.4.4 (D)TLS session resumption examples

#### 3.4.4.1 Session resumption with session ID not encrypted

**Phase 1: Acquiring the session data**

| Command | Response | Description |
|---|---|---|
| **Step 1: Preparation steps** | | |
| | | AT command sequence to ensure Internet connectivity. |
| | | Required +USECMGN AT commands to handle the certificates/keys. |
| | | Required +USECPRF AT commands to configure the security profile <profile_id>. |
| | | Required AT commands to configure application profile <app_profile> (in the example the application will be +UHTTP – HTTP client). |

| Command | Response | Description |
|---|---|---|
| **Step 2: Enable the session resumption** | | |
| AT+USECPRF=<profile_id>,13,0,1 | OK | Enable the session resumption for the security profile <profile_id>. |
| **Step 3: Set the session resumption type** | | |
| AT+USECPRF=<profile_id>,13,1,0 | OK | Set the session resumption type for the security profile <profile_id>. |
| **Step 4: Associate the application profile to the security profile** | | |
| AT+UHTTP=<app_profile>,6,1,<profile_id> | OK | The application profile <app_profile> is associated to the security profile <profile_id>. |
| **Step 5: Execute HTTP GET request** | | |
| AT+UHTTPC=<app_profile>,1,"/index.html","response_file" | OK | Perform HTTP GET request. |
| | +UUSECPRF: <profile_id>,13,1,0 | URC with session resumption type.<br>☞ The session resumption type should be stored to be reused for the resumption on the next session. |
| | +UUSECPRF: <profile_id>,13,2,<session_id_b64>,<master_secret_b64> | URC with session resumption data.<br>☞ The session resumption data should be stored to be reused for the resumption on the next session. |
| | +UUHTTPCR: <app_profile>,1,1 | HTTP GET URC response. |

## Phase 2: Reusing previous session

| Command | Response | Description |
|---|---|---|
| **Step 1: Preparation steps** | | |
| | | AT command sequence to ensure Internet connectivity. |
| | | Required AT commands to configure application profile <app_profile> (in the example the application will be +UHTTP). |
| **Step 2: Enable the session resumption** | | |
| AT+USECPRF=<profile_id>,13,0,1 | OK | Enable the session resumption for the security profile <profile_id>. |
| **Step 3: Set the session resumption type** | | |
| AT+USECPRF=<profile_id>,13,1,0 | OK | Set the session resumption type for the security profile <profile_id>. |
| **Step 4: Set the session resumption data** | | |
| AT+USECPRF=<profile_id>,13,2,<session_id_b64>,<master_secret_b64> | OK | Set the session resumption data for the security profile <profile_id>. |
| **Step 5: Associate the application profile to the security profile** | | |
| AT+UHTTP=<app_profile>,6,1,<profile_id> | OK | The application profile <app_profile> is associated to the security profile <profile_id>. |
| **Step 6: Execute HTTP GET request** | | |
| | OK | Perform HTTP GET request. |

| Command | Response | Description |
|---|---|---|
| AT+UHTTPC=<app_profile>,1,"/index.html","response_file" | +UUSECPRF: <profile_id>,13,1,0 | URC with session resumption type.<br>☞ If the displayed session type is not equal to the session type already set, the session has not been resumed. The new session type should be set. |
| | +UUSECPRF: <profile_id>,13,2,<session_id_b64>,<master_secret_b64> | URC with session resumption data.<br>☞ If the displayed session data is not equal to the session data already set, the session has not been resumed. The new session type should be set. |
| | +UUHTTPCR: <app_profile>,1,1 | HTTP GET URC response. |

### 3.4.4.2 Session resumption with session ID encrypted with local encryption

#### Phase 1: Acquiring the session data

| Command | Response | Description |
|---|---|---|
| **Step 1: Preparation steps** | | |
| | | AT command sequence to ensure Internet connectivity. |
| | | Required +USECMGN AT commands to handle the certificates/keys. |
| | | Required +USECPRF AT commands to configure the security profile <profile_id>. |
| | | Required AT commands to configure application profile <app_profile> (in the example the application will be +UHTTP – HTTP client). |
| **Step 2: Enable the session resumption** | | |
| AT+USECPRF=<profile_id>,13,0,1 | OK | Enable the session resumption for the security profile <profile_id>. |
| **Step 3: Set the session resumption type** | | |
| AT+USECPRF=<profile_id>,13,1,10 | OK | Set the session resumption type for the security profile <profile_id>. |
| **Step 4: Associate the application profile to the security profile** | | |
| AT+UHTTP=<app_profile>,6,1,<profile_id> | OK | The application profile <app_profile> is associated to the security profile <profile_id>. |
| **Step 5: Execute HTTP GET request** | | |
| AT+UHTTPC=<app_profile>,1,"/index.html","response_file" | OK | Perform HTTP GET request. |
| | +UUSECPRF: <profile_id>,13,1,10 | URC with session resumption type.<br>☞ The session resumption type should be stored to be reused for the resumption on the next session. |
| | +UUSECPRF: <profile_id>,13,12,<enc_session_data_b64>,<enc_session_data_b64_size> | URC with encrypted session resumption data.<br>☞ The session resumption data should be stored to be reused for the resumption on the next session. |
| | +UUHTTPCR: <app_profile>,1,1 | HTTP GET URC response. |

#### Phase 2: Reusing previous session

| Command | Response | Description |
|---|---|---|
| **Step 1: Preparation steps** | | |

| Command | Response | Description |
|---|---|---|
| | | AT command sequence to ensure Internet connectivity. |
| | | Required AT commands to configure application profile <app_profile> (in the example the application will be +UHTTP). |
| **Step 2: Enable the session resumption** | | |
| AT+USECPRF=<profile_id>,13,0,1 | OK | Enable the session resumption for the security profile <profile_id>. |
| **Step 3: Set the session resumption type** | | |
| AT+USECPRF=<profile_id>,13,1,10 | OK | Set the session resumption type for the security profile <profile_id>. |
| **Step 4: Set the encrypted session resumption data** | | |
| AT+USECPRF=<profile_id>,13,12,<enc_session_data_b64>,<enc_session_data_b64_size> | OK | Set the encrypted session resumption data for the security profile <profile_id>. |
| **Step 5: Associate the application profile to the security profile** | | |
| AT+UHTTP=<app_profile>,6,1,<profile_id> | OK | The application profile <app_profile> is associated to the security profile <profile_id>. |
| **Step 6: Execute HTTP GET request** | | |
| AT+UHTTPC=<app_profile>,1,"/index.html","response_file" | OK | Perform HTTP GET request. |
| | +UUSECPRF: <profile_id>,13,1,10 | URC with session resumption type.<br>☞ If the displayed session type is not equal to the session type already set, the session has not been resumed. The session type should be updated. |
| | +UUSECPRF: <profile_id>,13,12,<enc_session_data_b64>,<enc_session_data_b64_size> | URC with encrypted session resumption data.<br>☞ If the displayed session data is not equal to the session data already set, the session has not been resumed. The session data should be updated. |
| | +UUHTTPCR: <app_profile>,1,1 | HTTP GET URC response. |

### 3.4.4.3 Session resumption with session ticket not encrypted

#### Phase 1: Acquiring the session data

| Command | Response | Description |
|---|---|---|
| **Step 1: Preparation steps** | | |
| | | AT command sequence to ensure Internet connectivity. |
| | | Required +USECMGN AT commands to handle the certificates/keys. |
| | | Required +USECPRF AT commands to configure the security profile <profile_id>. |
| | | Required AT commands to configure application profile <app_profile> (in the example the application will be +UHTTP – HTTP client). |
| **Step 2: Enable the session resumption** | | |
| AT+USECPRF=<profile_id>,13,0,1 | OK | Enable the session resumption for the security profile <profile_id>. |

| Command | Response | Description |
|---|---|---|
| **Step 3: Set the session resumption type** | | |
| AT+USECPRF=<profile_id>,13,1,1 | OK | Set the session resumption type for the security profile <profile_id>. |
| **Step 4: Associate the application profile to the security profile** | | |
| AT+UHTTP=<app_profile>,6,1,<profile_id> | OK | The application profile <app_profile> is associated to the security profile <profile_id>. |
| **Step 5: Execute HTTP GET request** | | |
| AT+UHTTPC=<app_profile>,1,"/index.html","response_file" | OK | Perform HTTP GET request. |
| | +UUSECPRF: <profile_id>,13,0,2 | URC with session resumption status. |
| | +UUHTTPCR: <app_profile>,1,1 | HTTP GET URC response. |
| **Step 6: Get the session resumption data** | | |
| AT+USECPRF=<profile_id>,13,3 | +USECPRF: <profile_id>,13,3,<session_data_b64>,<session_data_b64_size><br>OK | Get the session data configured during the handshake.<br>☞ The session resumption data should be stored to be reused for the resumption on the next session |

## Phase 2: Reusing previous session

| Command | Response | Description |
|---|---|---|
| **Step 1: Preparation steps** | | |
| | | AT command sequence to ensure Internet connectivity. |
| | | Required AT commands to configure application profile <app_profile> (in the example the application will be +UHTTP). |
| **Step 2: Enable the session resumption** | | |
| AT+USECPRF=<profile_id>,13,0,1 | OK | Enable the session resumption for the security profile <profile_id>. |
| **Step 3: Set the session resumption type** | | |
| AT+USECPRF=<profile_id>,13,1,1 | OK | Set the session resumption type for the security profile <profile_id>. |
| **Step 4: Set the session resumption data** | | |
| AT+USECPRF=<profile_id>,13,3,<session_data_b64>,<session_data_b64_size> | OK | Set the session resumption data for the security profile <profile_id>. |
| **Step 5: Associate the application profile to the security profile** | | |
| AT+UHTTP=<app_profile>,6,1,<profile_id> | OK | The application profile <app_profile> is associated to the security profile <profile_id>. |
| **Step 6: Execute HTTP GET request** | | |
| AT+UHTTPC=<app_profile>,1,"/index.html","response_file" | OK | Perform HTTP GET request. |
| | +UUSECPRF: <profile_id>,13,0,2 | URC with session resumption status. |
| | +UUHTTPCR: <app_profile>,1,1 | HTTP GET URC response. |

#### 3.4.4.4 Session resumption with session ticket encrypted with local encryption

**Phase 1: Acquiring the session data**

| Command | Response | Description |
|---|---|---|
| **Step 1: Preparation steps** | | |
| | | AT command sequence to ensure Internet connectivity. |
| | | Required +USECMGN AT commands to handle the certificates/keys. |
| | | Required +USECPRF AT commands to configure the security profile <profile_id>. |
| | | Required AT commands to configure application profile <app_profile> (in the example the application will be +UHTTP – HTTP client). |
| **Step 2: Enable the session resumption** | | |
| AT+USECPRF=<profile_id>,13,0,1 | OK | Enable the session resumption for the security profile <profile_id>. |
| **Step 3: Set the session resumption type** | | |
| AT+USECPRF=<profile_id>,13,1,11 | OK | Set the session resumption type for the security profile <profile_id>. |
| **Step 4: Associate the application profile to the security profile** | | |
| AT+UHTTP=<app_profile>,6,1,<profile_id> | OK | The application profile <app_profile> is associated to the security profile <profile_id>. |
| **Step 5: Execute HTTP GET request** | | |
| AT+UHTTPC=<app_profile>,1,"/index.html","response_file" | OK | Perform HTTP GET request. |
| | +UUSECPRF: <profile_id>,13,0,2 | URC with session resumption status. |
| | +UUHTTPCR: <app_profile>,1,1 | HTTP GET URC response. |
| **Step 6: Get the session resumption data** | | |
| AT+USECPRF=<profile_id>,13,13 | +USECPRF: <profile_id>,13,13,<enc_session_data_b64>,<enc_session_data_b64_size><br>OK | Get the session data configured during the handshake.<br>☞ The session resumption data should be stored to be reused for the resumption on the next session |

**Phase 2: Reusing previous session**

| Command | Response | Description |
|---|---|---|
| **Step 1: Preparation steps** | | |
| | | AT command sequence to ensure Internet connectivity. |
| | | Required AT commands to configure application profile <app_profile> (in the example the application will be +UHTTP). |
| **Step 2: Enable the session resumption** | | |
| AT+USECPRF=<profile_id>,13,0,1 | OK | Enable the session resumption for the security profile <profile_id>. |
| **Step 3: Set the session resumption type** | | |
| AT+USECPRF=<profile_id>,13,1,11 | OK | Set the session resumption type for the security profile <profile_id>. |
| **Step 4: Set the encrypted session resumption data** | | |

| Command | Response | Description |
|---|---|---|
| AT+USECPRF=<profile_id>,13,13,<enc_session_data_b64>,<enc_session_data_b64_size> | OK | Set the encrypted session resumption data for the security profile <profile_id>. |
| **Step 5: Associate the application profile to the security profile** | | |
| AT+UHTTP=<app_profile>,6,1,<profile_id> | OK | The application profile <app_profile> is associated to the security profile <profile_id>. |
| **Step 6: Execute HTTP GET request** | | |
| AT+UHTTPC=<app_profile>,1,"/index.html","response_file" | OK | Perform HTTP GET request. |
| | +UUSECPRF: <profile_id>,13,0,2 | URC with session resumption status. |
| | +UUHTTPCR: <app_profile>,1,1 | HTTP GET URC response. |

### 3.4.4.5 Session resumption with PSK-based session ticket not encrypted

#### Phase 1: Acquiring the session data

| Command | Response | Description |
|---|---|---|
| **Step 1: Preparation steps** | | |
| | | AT command sequence to ensure Internet connectivity. |
| | | Required +USECMGN AT commands to handle the certificates/keys. |
| | | Required +USECPRF AT commands to configure the security profile <profile_id>.<br>☞ The TLS 1.3 version should be set in order to enable this type of session resumption |
| | | Required AT commands to configure application profile <app_profile> (in the example the application will be +UHTTP – HTTP client). |
| **Step 2: Enable the session resumption** | | |
| AT+USECPRF=<profile_id>,13,0,1 | OK | Enable the session resumption for the security profile <profile_id>. |
| **Step 3: Set the session resumption type** | | |
| AT+USECPRF=<profile_id>,13,1,3 | OK | Set the session resumption type for the security profile <profile_id>. |
| **Step 4: Associate the application profile to the security profile** | | |
| AT+UHTTP=<app_profile>,6,1,<profile_id> | OK | The application profile <app_profile> is associated to the security profile <profile_id>. |
| **Step 5: Execute HTTP GET request** | | |
| AT+UHTTPC=<app_profile>,1,"/index.html","response_file" | OK | Perform HTTP GET request. |
| | +UUSECPRF: <profile_id>,13,0,2 | URC with session resumption status. |
| | +UUHTTPCR: <app_profile>,1,1 | HTTP GET URC response. |
| **Step 6: Get the session resumption data** | | |
| AT+USECPRF=<profile_id>,13,5 | +USECPRF: <profile_id>,13,5,<session_data_b64>,<session_data_b64_size><br>OK | Get the session data configured during the handshake. |

| Command | Response | Description |
|---|---|---|
| | | ☞ The session resumption data should be stored to be reused for the resumption on the next session |

### Phase 2: Reusing previous session

| Command | Response | Description |
|---|---|---|
| **Step 1: Preparation steps** | | |
| | | AT command sequence to ensure Internet connectivity. |
| | | Required +USECPRF AT commands to configure the security profile <profile_id>.<br>☞ The TLS 1.3 version should be set in order to enable this type of session resumption |
| | | Required AT commands to configure application profile <app_profile> (in the example the application will be +UHTTP). |
| **Step 2: Enable the session resumption** | | |
| `AT+USECPRF=<profile_id>,13,0,1` | `OK` | Enable the session resumption for the security profile <profile_id>. |
| **Step 3: Set the session resumption type** | | |
| `AT+USECPRF=<profile_id>,13,1,3` | `OK` | Set the session resumption type for the security profile <profile_id>. |
| **Step 4: Set the session resumption data** | | |
| `AT+USECPRF=<profile_id>,13,5,<session_data_b64_size>`<br>`>`<br>`<session_data_b64>` | `OK` | Set the session resumption data for the security profile <profile_id>. |
| **Step 5: Associate the application profile to the security profile** | | |
| `AT+UHTTP=<app_profile>,6,1,<profile_id>` | `OK` | The application profile <app_profile> is associated to the security profile <profile_id>. |
| **Step 6: Execute HTTP GET request** | | |
| `AT+UHTTPC=<app_profile>,1,"/index.html","response_file"` | `OK` | Perform HTTP GET request. |
| | `+UUSECPRF: <profile_id>,13,0,2` | URC with session resumption type. |
| | `+UUHTTPCR: <app_profile>,1,1` | HTTP GET URC response. |

## 3.4.4.6 Session resumption with PSK-based session ticket encrypted with local encryption

### Phase 1: Acquiring the session data

| Command | Response | Description |
|---|---|---|
| **Step 1: Preparation steps** | | |
| | | AT command sequence to ensure Internet connectivity. |
| | | Required +USECMNG AT commands to handle the certificates/keys. |
| | | Required +USECPRF AT commands to configure the security profile <profile_id>.<br>☞ The TLS 1.3 version should be set to enable this type of session resumption |

| Command | Response | Description |
|---|---|---|
| | | Required AT commands to configure application profile <app_profile> (in the example the application will be +UHTTP – HTTP client). |
| **Step 2: Enable the session resumption** | | |
| AT+USECPRF=<profile_id>,13,0,1 | OK | Enable the session resumption for the security profile <profile_id>. |
| **Step 3: Set the session resumption type** | | |
| AT+USECPRF=<profile_id>,13,1,13 | OK | Set the session resumption type for the security profile <profile_id>. |
| **Step 4: Associate the application profile to the security profile** | | |
| AT+UHTTP=<app_profile>,6,1,<profile_id> | OK | The application profile <app_profile> is associated to the security profile <profile_id>. |
| **Step 5: Execute HTTP GET request** | | |
| AT+UHTTPC=<app_profile>,1,"/index.html","response_file" | OK | Perform HTTP GET request. |
| | +UUSECPRF: <profile_id>,13,0,2 | URC with session resumption status.<br>☞ The session resumption type should be stored to be reused for the resumption on the next session. |
| | +UUHTTPCR: <app_profile>,1,1 | HTTP GET URC response. |
| **Step 6: Get the session resumption data** | | |
| AT+USECPRF=<profile_id>,13,15 | +USECPRF: <profile_id>,13,15,<enc_session_data_b64>,<enc_session_data_b64_size><br>OK | Get the session data configured during the handshake.<br>☞ The session resumption data should be stored to be reused for the resumption on the next session |

## Phase 2: Reusing previous session

| Command | Response | Description |
|---|---|---|
| **Step 1: Preparation steps** | | |
| | | AT command sequence to ensure Internet connectivity. |
| | | Required +USECPRF AT commands to configure the security profile <profile_id>.<br>☞ The TLS 1.3 version should be set to enable this type of session resumption |
| | | Required AT commands to configure application profile <app_profile> (in the example the application will be +UHTTP). |
| **Step 2: Enable the session resumption** | | |
| AT+USECPRF=<profile_id>,13,0,1 | OK | Enable the session resumption for the security profile <profile_id>. |
| **Step 3: Set the session resumption type** | | |
| AT+USECPRF=<profile_id>,13,1,13 | OK | Set the session resumption type for the security profile <profile_id>. |
| **Step 4: Set the encrypted session resumption data** | | |
| AT+USECPRF=<profile_id>,13,15,<enc_session_data_b64_size><br>><br><enc_session_data_b64> | OK | Set the encrypted session resumption data for the security profile <profile_id>. |

| Command | Response | Description |
|---|---|---|
| **Step 5: Associate the application profile to the security profile** | | |
| AT+UHTTP=<app_profile>,6,1,<profile_id> | OK | The application profile <app_profile> is associated to the security profile <profile_id>. |
| **Step 6: Execute HTTP GET request** | | |
| AT+UHTTPC=<app_profile>,1,"/index.html","response_file" | OK | Perform HTTP GET request. |
| | +UUSECPRF: <profile_id>,13,0,2 | URC with session resumption status. |
| | +UUHTTPCR: <app_profile>,1,1 | HTTP GET URC response. |

## 3.5 Troubleshooting secure connection

If your application cannot complete a secure connection, check the list below to correctly configure the secure SSL/TLS connection between cellular modules and server.

- Decide the certification validation level that is required for your system and configure the module accordingly with the <op_code>=0 of the +USECPRF AT command.
- Ensure the server certificate used for the TLS handshake is flagged as CA certificate.
- Install the SSL/TLS CA certificate based on server TLS certificate chain by using the +USECMNG and +USECPRF AT commands.
- Check the SSL/TLS protocol version required at the server and configure the module accordingly with the <op_code>=1 of the +USECPRF AT command.
- Be sure that cipher suite required by the destination server is present in the list of cipher suites available by default in the u-blox module. Alternatively, configure it with the <op_code>=2 of the +USECPRF AT command.
- If mutual authentication is adopted, properly configure the module with the specific device certificates and keys, via the +USECMNG and +USECPRF AT commands.
- Finally, ensure the SNI and the expected server “host name” are properly configured and aligned with the destination server. This can be achieved with the <op_code>=10 and <op_code>=4, respectively, of the +USECPRF AT command.

See the example of this configuration in Section 3.2.

# 4 Dial-up network (PPP)

The module can perform dial-up network (DUN) connections supporting the Point-to-Point Protocol (PPP). The PPP connection is established between the host (e.g., Windows device) and the DCE.

When a data call is initiated by means of the D* AT command, the module switches to the PPP mode just after the CONNECT intermediate result code. If a PDN connection is not active on the specific CID, it will be activated.

After the CONNECT message has been sent from DCE to the DTE, the DTE can start the PPP negotiation sending the configuration request. The following PPP negation steps must be followed as described in RFC-1662.

For all CIDs except the CID=1 in LTE (the initial default EPS bearer, which is configurable via AT commands), the host can control the authentication parameters and the MTU (maximum transmission unit) size directly through PPP. The MTU IPv4 size assigned by the network can be read with the AT+ CGCONTRDP=<cid> command.

☞ If the network throughput is less than the data sent from the host to the module (which is limited by the radio resources assigned by the network to the transmission in the uplink), then packet data loss may occur, even with hardware flow control enabled. To avoid this issue, do either or both:

- reduce the baud rate used on the serial COM port.
- slow down data transfer load by adding pauses between data payloads or breaking up their payload and adding delay.

☞ The dial-up is independent of the USB suspension. In the case of USB suspension, the PPP functionalities will remain in an idle state; while if data activity is performed the USB port will be re-established.

## 4.1 Basic setup

☞ The module must be attached to the network and the APN must be properly configured into the PDP context before starting the dial-up.

| Command | Response | Description |
|---|---|---|
| `ATD*99***1#` | | Perform the dial-up on the PDP context on CID=1. |

Using the dial command for establishing PPP connection, `ATD*99***1#`, the “1” in this example refers to the first active PDP context returned by the +CGDCONT read command.

### 4.1.1 Dial-up configuration

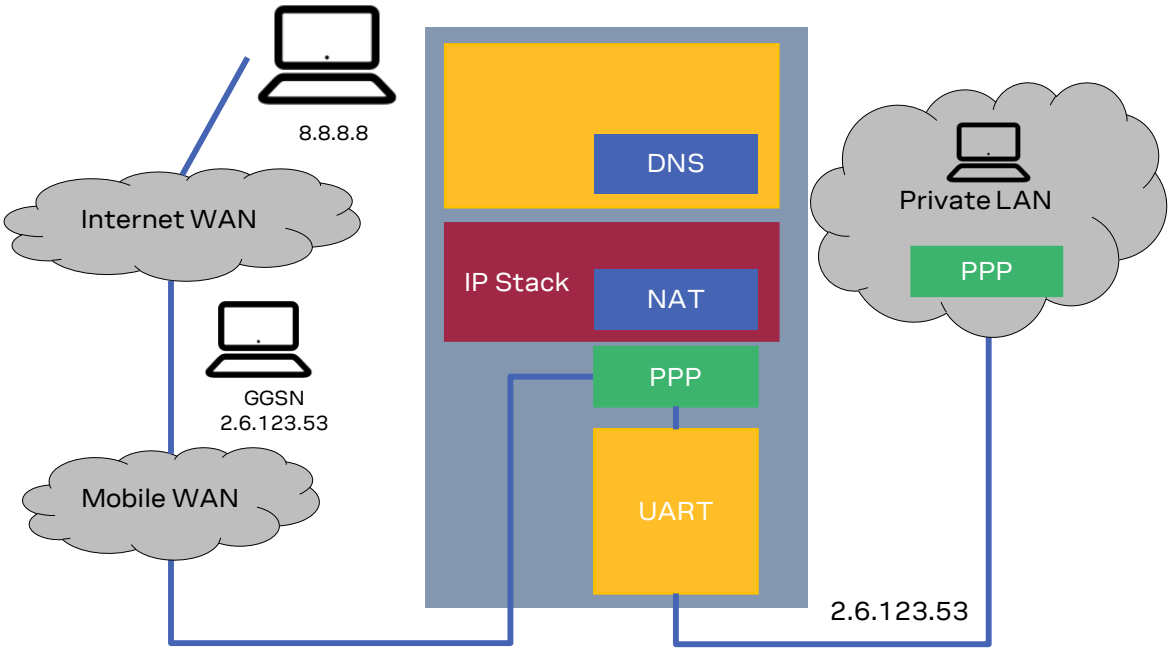



**Figure 4: Example of a dial-up configuration**

The application processor gets the module IP address that was previously assigned by the network during the PDN activation.

The module acts as a transparent data pipe and the internal TCP/IP stack is not involved at all.

### 4.1.2 PPP and IPv6

Unlike IPv4, IPv6 does not offer private addressing or NAT features. By design, IPv6 allows each node in the network to obtain its own IPv6 global address (i.e., an address reachable from any other host on the internet) via the StateLess Address AutoConfiguration (SLAAC) procedure.

With PPPv6 the PPP framework in the module does not directly provide any global IPv6 address to the PPP client (DTE), and it is up to the PPP client to start the SLAAC procedure with the network sending a Router Solicitation (RS) message. The network then replies with a Router Advertisement (RA) packet containing the IPv6 address prefix to be used by the DTE to generate its own IPv6 global address. At the end of the procedure, the DTE and the module will own two different IPv6 addresses sharing the same IPv6 address prefix, and both the peers will be reachable from the internet. In other words, the application processor's IPv6 address is in the same network (i.e., same global prefix) of the module's one, but the IIDs differ (i.e., two distinct IPv6 addresses are assigned); for example "`2A0B:AD40:1:102A:2A0B:AD40:1:102A`" and "`2A0B:AD40:1:102A:90A1:5CFC:3CC9:7301`".

Once the IPv6 address is generated, the DTE will be able to perform data traffic and the module will act as a transparent data pipe (it will just forward IPv6 packet to/from the network). This behavior is very similar to the PPP for IPv4, the only difference is that DTE and the module will own two different IPv6 addresses. The DTE will be exposed to any incoming connection from the internet, there will not be any filtering because of incoming data.

### 4.1.3 PPP over multiple PDP contexts

Optionally, a second PDP context can be set up for PPP. Up to 3 PPP instances can be simultaneously active in the LARA-R6 series modules.

☞ A different and unique APN is required per PDP context.

In the below example there are two PDP contexts defined and activated. The second context on CID=2 may be utilized by PPP. Do not activate the second context manually, instead establishing and terminating the PPP session on CID=2 will automatically activate and deactivate it. In the example the second PDP context has been activated by the PPP session.

Example of a second PDP context for PPP dial-up connection:

```
+CGDCONT: 1,"IP","APN1","166.130.71.189",0,0,0,0
+CGDCONT: 2,"IPV4V6","APN2","10.117.32.103 38.0.3.128.178.65.129.209.0.0.0.74.87.68.176
.1",0,0,0,0
```

For more details on multiple PDP contexts, see the AT commands manual [3].

## 4.2 Terminate cellular packet data connection

The PPP data session can be terminated by one of the following events:

- via a DTR hardware transition of the pin from ON to OFF
- sending an LCP "Terminated request"
- sending the string "+++" in the AT interface (see the "Circuit 108/2 behavior &D" section of the AT commands manual [3] for further details on "+++" different behaviors)

When using MUX and PPP combined, toggling the DTR line does not terminate the PPP session and return the device to the command mode. In this configuration, it is recommended that the host terminates the PPP session, which can be done by sending LCP_TERM REQ. Another method to terminate the PPP session is to send a MSC MUX frame for logical DTR de-assert.

☞ Issue the AT+CVHU=0 command to make ATH over Online Command Mode (OLCM) work, according to 3GPP requirements. If the module has a DUN/PPP activated and is in OLCM, the command deactivates the PPP and the associated PDP context (if possible).

## 4.3 Port filtering feature for embedded IP applications

The PPP link between the network and the DTE is exclusive for other traffic sources inside the module, meaning that any incoming IP packet will be directed to the DTE by default. This will prevent correct functionalities of embedded IP applications such as:

- Bearer Independent Protocol (BIP), used for UICC provisioning and SIM OTA sessions.
- Lightweight Machine-to-Machine client (LwM2M), used for uFOTA and MNO-based device management.
- u-blox Security Client (USEC), used for internal module security provisioning.
- Any other internal application as UFTP, MQTT, and CoAP.

☞ It is recommended to disable the embedded applications that host application does not use.

To overcome this, the +UEMBPF (Embedded port filtering) AT command shall be configured in the module.

The command requires a port range which will be used as source port for any TCP/UDP traffic sourced from embedded applications. As any DL external traffic targeting these ports will not be routed to DTE, the input port range must be outside of the range commonly used by DTE via the PPP session.

☞ It is recommended to use a port range above port 1024 considering that each port below 1024 is associated by default with a specific protocol.

See Figure 5 for a scheme of the functionality. The command is required only if a dial-up connection is used. For a complete syntax description, see the LARA-R6 series AT commands manual [3].

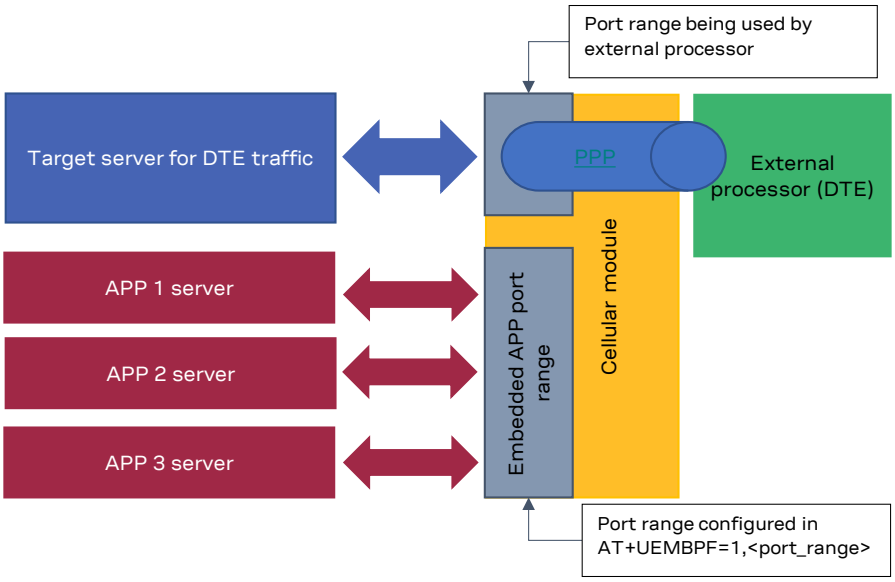
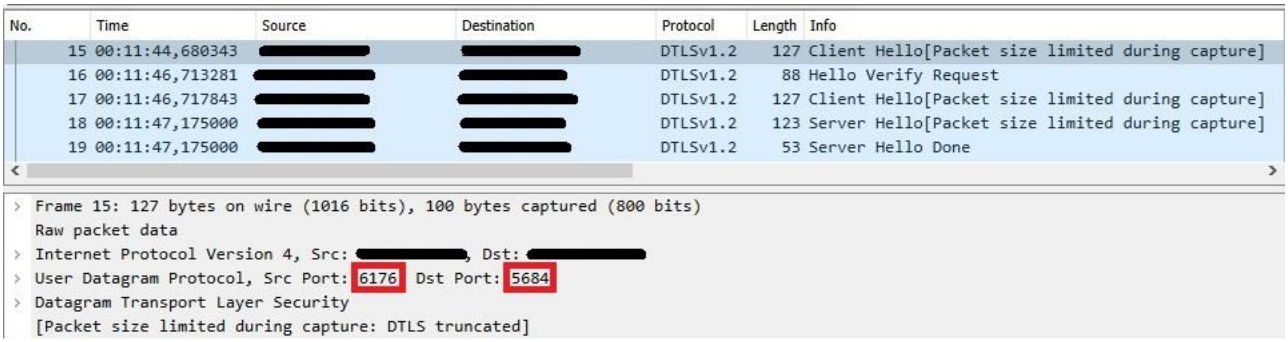



**Figure 5: Scheme of a possible router mode configuration**

### 4.3.1 Example: +UEMBPF with PPP & LwM2M client handling

The port filtering feature configures the port range from which the embedded applications will pick the source port when opening the socket. Normally the source port is picked randomly by the modem IP stack, while the destination port is protocol-dependent and server-dependent. See the example below, where the AT+UEMBPF=1,"6000-6200" command is set and LwM2M performs a connection to server "lwm2m-fota.services.u-blox.com:5684".

LwM2M, which server expects traffic to UDP port 5684, picks source port UDP 6167 which is indeed inside the configured range.



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 15 | 00:11:44,680343 | [redacted] | [redacted] | DTLSv1.2 | 127 | Client Hello[Packet size limited during capture] |
| 16 | 00:11:46,713281 | [redacted] | [redacted] | DTLSv1.2 | 88 | Hello Verify Request |
| 17 | 00:11:46,717843 | [redacted] | [redacted] | DTLSv1.2 | 127 | Client Hello[Packet size limited during capture] |
| 18 | 00:11:47,175000 | [redacted] | [redacted] | DTLSv1.2 | 123 | Server Hello[Packet size limited during capture] |
| 19 | 00:11:47,175000 | [redacted] | [redacted] | DTLSv1.2 | 53 | Server Hello Done |

```
> Frame 15: 127 bytes on wire (1016 bits), 100 bytes captured (800 bits)
  Raw packet data
> Internet Protocol Version 4, Src: [redacted], Dst: [redacted]
> User Datagram Protocol, Src Port: 6176  Dst Port: 5684
> Datagram Transport Layer Security
  [Packet size limited during capture: DTLS truncated]
```

**Figure 6: Outgoing traffic from module to LwM2M server**

Accordingly, the range configured via +UEMBPF AT command is not linked to a specific protocol/server/client.

In general, the command configures filtering for incoming packets. This means that any incoming packet which has destination port inside the configured range, will be directed to the embedded applications instead of the PPP DTE.

Continuing to look at the example, outgoing packet has src=6176, dst=5684. The incoming then has src=5684, dst=6176.

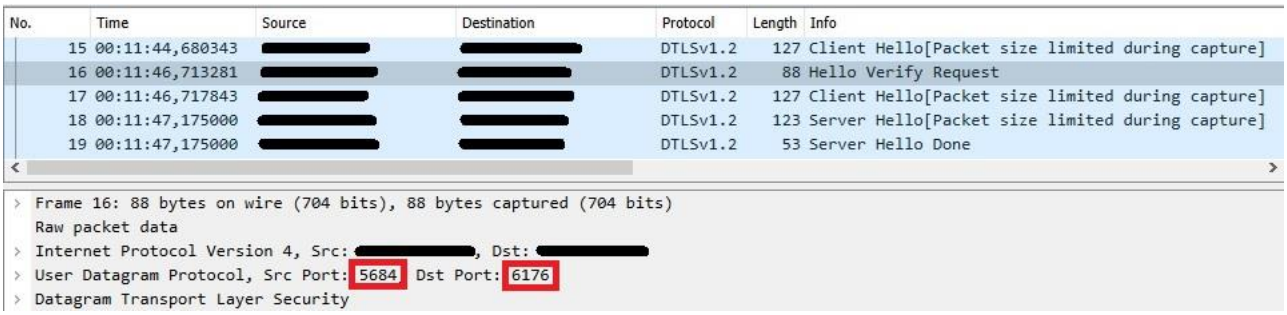



**Figure 7: Incoming traffic from LwM2M server to module**

This packet's destination port is independent of the configured filtering range, so it will be directed to the LwM2M client.

This means that the DTE will not receive any packet with destination port inside that range: referring the provided example, when using a source port inside the selected interval (i.e., "6000-6200") to send packets, their response will never be routed to DTE, but to embedded applications.

Thus, the range to be configured in +UEMBPF AT command must be outside the source port range being used by DTE via PPP.

| Command | Response | Description |
|---|---|---|
| AT+UEMBPF=1,"6000-6200" | OK | Embedded port filtering configuration. |
| AT+CFUN=16 | OK | Module reboot to apply from the following power up the desired settings. |
| AT+UEMBPF? | +UEMBPF: 1,"6000-6200"<br>OK | Embedded port filtering configuration check after the applied reboot. |

☞ In general, it is advisable to always apply the +UEMBPF AT command in scenarios in which a dial-up connection is referred. Basically, it could be sufficient to configure in this command a limited set of ports (at least 50 ports, e.g., "6000-6050") to be reserved for the embedded applications. These ports will be used by the internal IP applications (e.g., LwM2M client, security services, etc.) and they will not be functional for data traffic related to the PPP DTE, so some proper tuning could be needed at the user side.

### 4.3.2 +UEMBPF not configured

PPP is a mutually exclusive protocol. This means that once PPP DTE has established the connection, all the incoming traffic from network interface will be directed to DTE. If the port filtering is disabled with AT+UEMBPF=0 command, the embedded applications can send outgoing traffic, but will not receive the server response, which are routed to the DTE.

Therefore, it is important to properly configure the +UEMBPF AT command to avoid unexpected behaviors.

# 5 TCP/UDP internal stack

☞ Verify that the module is registered with the network and a PS data connection is activated. Make sure to follow the steps in section 2 before using the AT commands in this section.

☞ For UDP it is highly recommended to use +USOST and +USORF AT commands instead of +USOCO, +USOWR and +USORD AT commands.

☞ The use of +USOST and +USORF AT commands is recommended without the use of the +USOCO AT command. Precisely, the +USOCO AT command is compatible only with +USORD and +USOWR AT commands.

☞ In LARA-R6001, LARA-R6401, and LARA-R6801 it is possible to configure the system socket feature such as TCP selective acknowledge (TCP S-ACK) with the AT+USOCFG command. The system configuration requires a module reboot to make the change effective.

## 5.1 Socket connect

| Command | Response | Description |
|---|---|---|
| AT+USOCR=6 | +USOCR: 0<br>OK | TCP socket creation. In this example socket #0 is created.<br>The information text response returns the created socket identifier (in this case #0). If a new socket is created (without closing the already existent), a new socket identifier will be returned.<br>☞ Created socket is by default mapped to the context ID 1.<br>☞ It is possible to manually configure a mapping between the embedded socket and another PDP context by specifying the IP type and context ID. E.g., with the following AT command: AT+USOCR=<protocol>[,<local_port>[,<IP_type>[,<cid>]]]. |
| AT+USOCR=17 | +USOCR: 1<br>OK | Create another socket (in this case the socket is UDP, and its identifier is 1).<br>☞ Created socket is by default mapped to the context ID 1. |
| AT+USOCL=1 | OK | Close socket #1. The socket #1 is free. |
| AT+UDNSRN=0,"ftp.u-blox.com" | +UDNSRN: "195.34.89.241"<br>OK | DNS resolution of the URL "ftp.u-blox.com". |
| AT+USOCO=0,"195.34.89.241",7 | OK | Connect socket #0 to port 7 of a remote host with IP address 195.34.89.241.<br>**The connection is now uniquely associated to the socket. The socket is now ready for read/write operations.** |
| AT+USOCO=0,"195.34.89.241",7 | ERROR<br><br>+UUSOCL: 0 | If the connection is not successfully performed, an error result code is returned and the socket used for the connection attempt is closed. The notification is provided by +UUSOCL URC. |

## 5.2 Socket listening

| Command | Response | Description |
|---|---|---|
| AT+USOCR=6 | +USOCR: 0<br>OK | TCP socket creation with ID #0. |

| Command | Response | Description |
|---|---|---|
| AT+USOLI=0,1099 | OK | Set the socket in listening mode on port 1099.<br>⚠ The ability to reach the opened port on the server depends also on the network operator. Some network operators do not allow incoming connection on opened TCP/UDP port. |
| | +UUSOLI: 1,"151.9.34.66",39912,0,"151.9.34.74",1099 | When a connection request arrives from a remote host, a new socket is created with the first integer identifier available. In this example the socket ID is #1. The +UUSOLI URC indicates:<br>• **1**: the new socket created. Incoming data from the established connection will be received on this socket. Data to be sent must be written into this socket<br>• **151.9.34.66**: IP of the remote server<br>• **39912**: service port<br>• **0**: listening socket. It is the socket identifier specified with the +USOLI AT command<br>• **151.9.34.74**: module IP address<br>• **1099**: listening port assigned to the connection. Configured with the +USOLI AT command<br>Socket #1 is now ready for reading/writing data. |
| | +UUSORD: 1,18 | 18 bytes of incoming data over the previously established connection.<br>☞ The incoming data will always be sent on the related socket. |

## 5.3 Socket write (+USOWR)

### 5.3.1 Binary mode

| Command | Response | Description |
|---|---|---|
| AT+USOWR=0,2 | @ | Request to write 2 data bytes into socket #0. Wait "@" symbol indicating the data prompt is now open (AT commands are not allowed in data prompt).<br>After the @ prompt reception, wait for a minimum of 50ms before sending data. |
| 12 | +USOWR: 0,2<br>OK | Write data bytes.<br>It is not allowed to write fewer bytes than previously specified with +USOWR AT command.<br>If more bytes are written with respect to the threshold, the remaining bytes will be truncated.<br>The interface is blocked until all bytes are written.<br>If the final result code is returned then the data is sent to a lower level of the protocol stack. **This is not a notification of an acknowledgment received from the remote host data bytes that have been sent to.** |

### 5.3.2 Base syntax

| Command | Response | Description |
|---|---|---|
| AT+USOWR=0,2,"12" | +USOWR: 0,2<br>OK | Write 2 data bytes data on socket #0.<br>If the final result code is returned then the data is sent to a lower level of the protocol stack. **This is not an acknowledgment from the remote host where the data bytes were sent.** |

| Command | Response | Description |
|---|---|---|
| | | ☞ Some characters are not allowed in base syntax mode. For the allowed characters, see the AT commands manual [3]. |

### 5.3.3 Queue FULL

| Command | Response | Description |
|---|---|---|
| `AT+USOWR=0,2,"12"` | `ERROR` | If the socket buffer is full, then the data bytes inserted in data prompt will be discarded: this may happen if the network is congested or if network coverage is lost.<br>**In this case an error result code is returned.**<br>☞ The socket queue size is set to 8 kB in the writing upload and 12 kB in reading download.<br>☞ The secure socket queue size is limited to 1kB in the writing upload. |
| `AT+USOCTL=0,10` | `+USOCTL: 0,10,4`<br>`OK` | In case of an error result code, it is recommended to query the state of TCP connection associated to the socket to verify the socket is still connected. The third parameter of the information text response is the state; if its value is 4, it means the connection is established. |
| `AT+USOCTL=0,11` | `+USOCTL: 0,11,0`<br>`OK` | It is also possible to query for TCP outgoing unacknowledged data of the socket (this command is valid only for TCP socket).<br>In this case, 0 bytes of data is unacknowledged. |

## 5.4 Socket operations with "Keep Alive" option

☞ In "Keep Alive" mode, the module periodically sends dummy TCP packets to prevent the network from closing the inactive context. The network operator may close inactive TCP connections without notification to the module.

| Command | Response | Description |
|---|---|---|
| `AT+USOCR=6` | `+USOCR: 0`<br>`OK` | Create a TCP socket #0. |
| `AT+USOSO=0,65535,8,1` | `OK` | Enable the "keep alive" option. This socket option enables the module to send dummy IP packets to keep the connection alive.<br>• **0**: socket number to be set to enable keep alive option<br>• **65535**: specifies socket level option<br>• **8**: specify the "Keep Alive" option<br>• **1**: enable the keep alive (set to 0 to disable it) |
| `AT+USOSO=0,6,2,30000` | `OK` | Set the inactivity timeout after which the module will start to send "keep alive" packets.<br>• **0**: socket number to be set to enable keep alive option<br>• **6**: specifies TCP level option<br>• **2**: specifies option TCP "keep idle" timer option<br>• **30000**: the module will send dummy TCP packets every 30000 ms |

## 5.5 Socket read (+USORD)

### First scenario

| Command | Response | Description |
|---|---|---|
| | `+UUSORD: 0,2` | The remote server sends 2 data bytes on socket #0.<br>A URC is returned indicating the socket on which the data is received and the total amount of data received. |
| `AT+USORD=0,2` | `+USORD: 0,2,"ar"`<br>`OK` | Read data. The data is returned between quotation marks. |

### Second scenario

| Command | Response | Description |
|---|---|---|
| | `+UUSORD: 0,30` | The remote server sends 30 data bytes on socket #0.<br>If a socket buffer is empty, the +UUSORD URC indicates a TCP packet has been received from the remote host the socket is connected to and the amount of data bytes of the packet. |
| `AT+USORD=0,10` | `+USORD: 0,10,"hfgyrhgfty"`<br>`OK` | Read only part of data (in this example 10 bytes of data are read).<br>Data is returned between quotation marks. |
| | `+UUSORD: 0,20` | The +UUSORD URC indicates the total amount of data bytes stored in the buffer after the last +USORD AT command execution. In this example 20 bytes are stored in the buffer. |

### Third scenario

| Command | Response | Description |
|---|---|---|
| | `+UUSORD: 0,30` | The remote server sends 30 data bytes on socket #0.<br>If a socket buffer is empty +UUSORD URC indicates a TCP packet has been received from the remote host the socket is connected to and the amount of data bytes of the packet. |
| `AT+USORD=0,10` | `+USORD: 0,10,"hfgyrhgfty"`<br>`OK` | Only part of the data bytes (10 bytes in this example) is read.<br>The data is returned between quotation marks. |
| | `+UUSORD: 0,25` | The remote server sent more data after the first part was received. The +UUSORD URC indicates the total amount of data bytes stored the buffer after the last +USORD AT command execution. In this example 25 bytes are stored in the buffer. |
| `AT+USORD=0,10` | `+USORD: 0,10,"hfgbchs7[o"`<br>`OK` | Only part of the data bytes (10 bytes in this example) is read.<br>Data is returned between quotation marks. |
| | `+UUSORD: 0,34` | The remote server sent more data. The +UUSORD URC indicates the total amount of data bytes stored the buffer after the last +USORD AT command execution. In this example 34 bytes are stored in the buffer. |
| `AT+USORD=0,34` | `+USORD: 0,34,"jghfbv74ksHDFUEçpjè0'@èpyujfnvhfyù"`<br>`OK` | All the bytes are read. |
| `AT+USORD=0,0` | `+USORD: 0,0`<br>`OK` | Verifies how much unread data is in the buffer. In this example 0 bytes are in socket #0. |

### Fourth scenario

| Command | Response | Description |
|---|---|---|
| | `+UUSORD: 0,30` | The remote host sends 30 bytes of data on the socket #0.<br>If a socket buffer is empty the +UUSORD URC indicates a TCP packet has been received from the remote host the socket is connected to and the amount of data bytes of the packet. |
| `AT+USORD=0,10` | `+USORD: 0,10,"hfgyrhgfty"`<br>`OK` | Only part of the data bytes (10 bytes in this example) is read.<br>Data is returned between quotation marks. |
| | `+UUSORD: 0,25` | The remote server sent other data after the first data bytes had been received. The +UUSORD URC indicates the total amount of data bytes stored the buffer after the last +USORD AT command execution. In this example 25 bytes are in the buffer. |
| | | The remote host closes the TCP connection associated to socket #0. |
| `AT+USOWR=0,3` | `@` | Request to write 3 data bytes into the socket #0. Wait for "@" symbol indicating the data prompt is now open.<br>After the @ prompt reception, wait for a minimum of 50 ms before sending data. |
| `123` | `+USOWR: 0,0`<br>`OK` | Write data. After the last byte the data prompt is closed.<br>☞ It is not allowed to write fewer bytes than previously specified with +USOWR AT command.<br>☞ If more bytes are written with respect to the threshold, the remaining bytes will be truncated.<br>The interface is blocked until all bytes are written.<br>The +USOWR: 0,0 URC indicates 0 bytes have been sent to the remote host. **This means the TCP connection is now closed.** |
| `AT+USORD=0,25` | `+USORD: 0,25,"23dfgt5uhj89ikdftevlpazwe"`<br>`OK` | Read the remaining data bytes still stored in the buffer of socket #0. |
| | `+UUSOCL: 0` | The URC indicates the TCP connection associated to socket #0 is now closed and socket #0 is cleared. |

## 5.6 Socket write (+USOST)

| Command | Response | Description |
|---|---|---|
| `AT+USOCR=17` | `+USOCR: 0`<br>`OK` | UDP socket creation. In this example the socket #0 is created.<br>The information text response returns the new socket identifier (in this example #0). If a new socket is created, a new socket identifier will be returned. |
| `AT+USOCR=17,12000` | `+USOCR: 0,12000`<br>`OK` | The local port to be used for data sending can be configured during the UDP socket creation.<br>In this example the socket #0 is created and bound with port 12000. Data written on socket #0 will be sent from this specific port. |
| `AT+UDNSRN=0,"ftp.u-blox.com"` | `+UDNSRN: "195.34.89.241"`<br>`OK` | DNS resolution of the URL "ftp.u-blox.com". |

| Command | Response | Description |
|---|---|---|
| AT+USOST=0,"195.34.89.241",7,2 | @ | Request to write 2 bytes of data into socket #0 specifying IP address and UDP port of the remote host UDP packet has to be sent to. Wait for "@" symbol indicating the data prompt is now open (AT commands are not allowed in data prompt). |
| 12 | +USOST: 0,2<br>OK | Write data. After the last data byte is written, the prompt is closed.<br>☞ It is not allowed to write fewer bytes than previously specified with +USOST AT command.<br>☞ If more bytes are written with respect to the threshold, the remaining bytes will be truncated.<br>The interface is blocked until all bytes are written.<br>The final result code is returned. This means the data is sent to a lower level of the protocol stack. This is not an acknowledgment, UDP is a connectionless protocol. |

## 5.7 Socket read (+USORF)

| Command | Response | Description |
|---|---|---|
| | +UUSORD: 0,2 | A UDP packet with 2 data bytes has been received. |
| AT+USORF=0,2 | +USORF: 0,"195.34.89.241",7,2,"12"<br>OK | Read data.<br>The information text response indicates:<br>• Read socket identifier<br>• Remote IP address<br>• Remote UDP port<br>• Number of read data bytes<br>• Read data bytes (between quotation marks) |
| | +UUSORD: 0,20 | UDP packet with 20 data bytes has been received from the remote server. |
| AT+USORF=0,10 | +USORF: 0,"195.34.89.241",7,2,"1234567890"<br>OK | Read 10 data bytes. |
| | +UUSORD: 0,10 | The +UUSORD URC indicates that 10 bytes are still unread. |
| | | The remote host sends a UDP packet with 20 data bytes. |
| AT+USORF=0,10 | +USORF: 0,"195.34.89.241",7,2,"1234567890"<br>OK<br><br>+UUSORD: 0,20 | Read the remaining 10 data bytes of the previous packet. The URC indicates 20 data bytes have been received and are still stored in the socket buffer.<br>☞ After the first URC has been returned, a second URC is returned (only after a reading operation) indicating:<br>• If a reading operation of a packet is not finished it will provide the remaining data of the specific packet<br>• Otherwise it will provide the number of data bytes of packets stored in the socket buffer |

## 5.8 Socket state

For a detailed description of TCP socket states, see the +USOCTL AT command description in AT commands manual [3].

| Command | Response | Description |
|---|---|---|
| AT+USOCTL=0,0 | +USOCTL: 0,0,6<br>OK | Query the socket type of the socket #0. The socket type information is provided by the third parameter (in this case 6 – TCP). |
| AT+USOCTL=0,10 | +USOCTL: 0,10,4<br>OK | It is possible to query the state of TCP connection associated with the socket; in this example the socket #0 (this command is valid only for TCP socket).<br>The third parameter of information text response provides the socket status (in this case 4 - the socket is in ESTABLISHED status). |
| AT+USOCTL=0,10 | +USOCTL: 0,10,7<br>OK | The third parameter of the information text response provides the socket status (in this case 7 - a TCP connection termination procedure is being performed). |
| AT+USOCTL=0,11 | +USOCTL: 0,11,0<br>OK | Query for TCP outgoing unacknowledged data of the socket #0 (this command is valid only for TCP socket). In this case 0 bytes of data are unacknowledged. |
| AT+USOCTL=0,1 | +USOCTL: 0,1,0<br>OK | Query for the last socket error for socket #0. If there are no errors the value is 0. |

In case of unexpected socket condition, use +USOER AT command to retrieve the last error occurred in the last socket operation.

| Command | Response | Description |
|---|---|---|
| AT+USOER | +USOER: 104<br>OK | Retrieve the last error occurred in a socket operation. |

## 5.9 Socket close

### By remote server

| Command | Response | Description |
|---|---|---|
| | +UUSOCL: 1 | The URC indicates the connection associated to socket 1 is closed. The socket #1 is cleared.<br>⚠ After this indication has been received the socket buffer is cleared. |

### By the module

| Command | Response | Description |
|---|---|---|
| AT+USOCL=0 | OK | The socket is closed by the module (socket #0).<br>⚠ No +UUSOCL URC returned. |

## 5.10 Testing sockets

A simple way to test TCP/UDP sockets over the network is to send data to an echo server.

☞ u-blox provides an echo server for testing purposes: echo.u-blox.com.

Here below an example using IPv4 UDP socket:

| Command | Response | Description |
|---|---|---|
| | | The module is already registered on the network, and a data connection is active. |
| AT+USOCR=17 | +USOCR: 0<br>OK | Create a UDP socket. |
| AT+UDNSRN=0,"echo.u-blox.com" | +UDNSRN: "195.34.89.241" | DNS resolution of the URL. |

| Command | Response | Description |
|---|---|---|
| | OK | |
| AT+USOST=0,"195.34.89.241",7,5,"Hello" | +USOST: 0,5<br>OK | Write 5 characters to server. |
| | +UUSORD: 0,5 | |
| AT+USORF=0,5 | +USORF: 0,"195.34.89.241",7,5,"Hello"<br>OK | Read 5 echoed characters. |

For additional details and examples on the use of the u-blox echo server, see the dedicated application note [16] .

## 5.11 Secure socket

Use the +USOSEC AT command to enable or disable the use of SSL/TLS/DTLS connection on a TCP or UDP socket.

A secure manager profile must be configured before starting a secure socket session. See section 3 for more details on this aspect.

☞ The enable or disable operation can be performed only after the socket has been created with +USOCR AT command.

☞ Even if the maximum number of sockets that can be opened simultaneously is 7, during any TLS procedure (e.g., during handshake) the socket client is blocked, and any other socket commands, either secure or not, cannot be issued until the first procedure is completed.

# 6 MQTT

☞ Make sure to follow the steps in section 2 before using the AT commands in this section. This is necessary because a PS data connection must be activated before using MQTT AT commands.

## 6.1 Basic setup

### 6.1.1 Default and minimal configuration

The configuration required to start a MQTT session depends on the broker (server) configuration, the most important of which is the MQTT remote server information. Use the broker configuration to correctly set up the module before starting a session.

| Command | Response | Description |
|---|---|---|
| AT+CMEE=2 | OK | Set verbose error result codes. |
| AT+UMQTT? | +UMQTT: 0,"357862090033897"<br>+UMQTT: 2,"",1883<br>+UMQTT: 3,"",1883<br>+UMQTT: 4,""<br>+UMQTT: 6,0<br>+UMQTT: 7,0<br>+UMQTT: 8,""<br>+UMQTT: 9,0,""<br>+UMQTT: 10,0<br>+UMQTT: 11,0<br>OK | Read the current profile configuration.<br>All the reported values can be modified; see the AT commands manual [3] for a detailed description.<br>☞ The default client id value is the IMEI of the module because it guarantees the uniqueness of the client to the server. |
| AT+UMQTT=2,"192.168.105.30",1883 | OK | Set the remote MQTT server's IP address and port. Alternatively, the server name can be set with the AT+UMQTT=3 command. |

### 6.1.2 Last will configuration

The "last will" parameters configure the message that the MQTT clients connected to the broker will receive in case of a module disconnection due to an error. Following is an example of setup.

| Command | Response | Description |
|---|---|---|
| AT+UMQTT=6,1 | OK | Set the last will quality of service (QoS) level to 1. |
| AT+UMQTT=8,"u-blox/publish" | OK | Set the last will topic. |
| AT+UMQTT=9,"Unrequested disconnect." | OK | Set the last will message. |

### 6.1.3 Profile management

| Command | Response | Description |
|---|---|---|
| AT+UMQTTNV=2 | OK | Store the current MQTT client profile parameters to the NVM. |
| AT+UMQTTNV=0 | OK | Restore MQTT client profile parameters to the factory-programmed setting. |
| AT+UMQTT? | +UMQTT: 0,"357862090033897"<br>+UMQTT: 2,"",1883<br>+UMQTT: 3,"",1883<br>+UMQTT: 4,""<br>+UMQTT: 6,0<br>+UMQTT: 7,0<br>+UMQTT: 8,"" | Read the current profile configuration. |

| Command | Response | Description |
|---|---|---|
| | +UMQTT: 9,0,""<br>+UMQTT: 10,0<br>+UMQTT: 11,0<br>OK | |
| AT+UMQTTNV=1 | OK | Set MQTT client profile parameters to values previously stored in the NVM. |
| AT+UMQTT? | +UMQTT: 0,"357862090033897"<br>+UMQTT: 2,"185.215.193.15",1883<br>+UMQTT: 3,"",1883<br>+UMQTT: 4,""<br>+UMQTT: 6,0<br>+UMQTT: 7,0<br>+UMQTT: 8,""<br>+UMQTT: 9,0,""<br>+UMQTT: 10,0<br>+UMQTT: 11,0<br>OK | Read the current profile configuration. |

### 6.1.4 Internal PDP context mapping

As an alternative to the default behavior, it is also possible to manually configure a mapping between the embedded MQTT client and another PDP context (different than default CID 1).

| Command | Response | Description |
|---|---|---|
| AT+UMQTT=20,2,1 | OK | Mapping the embedded MQTT client to use the context ID 2. With preferred protocol type 1 (thus, IPv6).<br>☞ This configuration is optional. Necessary only if the embedded MQTT client needs to use a PDP context different than CID 1. |

## 6.2 Start and end a MQTT session

See the section 6.1 to configure the MQTT profile before starting a connection.

| Command | Response | Description |
|---|---|---|
| AT+UMQTTC=1 | OK | Connect to the broker. |
| | +UUMQTTC: 1,1 | The MQTT session request is successfully performed. The MQTT session can start.<br>The +UUMQTTC URC provides the result of the requested action from the MQTT broker |
| AT+UMQTTC=0 | OK | Disconnect from the broker, end of the MQTT session. |
| | +UUMQTTC: 0,1 | The disconnection is successfully performed. |

## 6.3 Subscribe to a topic and publish a message to the same topic

The following example is a demonstration of the main functionalities that can be performed with the AT commands. In this MQTT session the module subscribes to a topic, publishes a message to the topic and receives the published message (since it is subscribed to topic of the published message).

| Command | Response | Description |
|---|---|---|
| AT+UMQTTC=4,0,"module/lights" | OK | Subscribe to a topic. |

| Command | Response | Description |
|---|---|---|
| | `+UUMQTTC: 4,1,0,"module/light"` | The broker granted QoS level is 0. |
| `AT+UMQTTC=2,0,0,0,"module/lights","light_1 is red"` | `OK` | Publish "light_1 is red " message to the "module/lights" topic with requested QoS level and retain value set to 0. |
| | `+UUMQTTC: 2,1` | |
| | `+UUMQTTC: 6,1` | Notification of the received publish message. |
| `AT+UMQTTC=6,1` | `+UMQTTC: 6,0,27,13,"module/lights",14,"light_1 is red"`<br>`OK` | Read the received publish message. |
| `AT+UMQTTC=5,"module/lights"` | `OK` | Unsubscribe from the previously subscribed topic. |
| | `+UUMQTTC: 5,1` | |

## 6.4 Publish a message with hexadecimal mode set

The following example shows how to publish a message whose payload is composed of hexadecimal bytes instead of ASCII characters. There are two possibilities to publish the sample "ABCD3031" string: the first is to publish it in "ASCII mode" and the second is to publish it in "HEX mode".

| Command | Response | Description |
|---|---|---|
| **ASCII mode** | | |
| `AT+UMQTTC=4,0,"module/ascii"` | `OK` | Subscribe to the "module/ascii" topic. |
| | `+UUMQTTC: 4,1,0,"module/ascii"` | |
| `AT+UMQTTC=2,0,0,0,"module/ascii","ABCD3031"` | `OK` | Send a Publish message, the "`ABCD3031`" payload is encoded with ASCII characters (the 4th parameter value (<hex_mode>) is 0). |
| | `+UUMQTTC: 2,1` | |
| | `+UUMQTTC: 6,1` | Notification of the received publish message. |
| `AT+UMQTTC=6,1` | `+UMQTTC: 6,0,20,12,"module/ascii",`<br>`8,"ABCD3031"`<br>`OK` | Read the received publish message, the payload is displayed in ASCII, so the received string is same as the one sent: 8 characters.<br>The payload bytes in the MQTT packet are:<br>`41 42 43 44 33 30 33 31` |
| **HEX mode** | | |
| `AT+UMQTTC=4,0,"module/hex"` | `OK` | Subscribe to the "module/hex" topic. |
| | `+UUMQTTC: 4,1,0,"module/hex"` | |
| `AT+UMQTTC=2,0,0,1,"module/hex","ABCD3031"` | `OK` | Send a Publish message with the same payload encoded as hexadecimal (the 4th parameter value (<hex_mode>) is 1). |
| | `+UUMQTTC: 2,1` | |
| | `+UUMQTTC: 6,1` | Notification of the received publish message. |
| `AT+UMQTTC=6,1` | `+UMQTTC: 6,0,14,10,"module/hex",`<br>`4,"«Í01"`<br>`OK` | Read the received publish message, the payload length is 4 because each pair of characters is considered as one byte.<br>The payload bytes in the MQTT packet are:<br>`AB CD 30 31`<br><br>Since "AB" and "CD" are not strict ASCII characters their output depends on the interface of the terminal application used to communicate with the module. In this example, the m-center is used: the "AB" and "CD" bytes are respectively displayed as "«" and "Í" characters.<br>The other 2 bytes "30" and "31" are respectively the standard ASCII characters "0" and "1". |

## 6.5 Publish a binary message to a topic

If the message payload contains special characters like quotation marks (“”), carriage return (<CR>), etc., the AT+UMQTTC=9 command should be used.

| Command | Response | Description |
|---|---|---|
| AT+UMQTTC=4,0,"module/special" | OK | Subscribe to the "module/special" topic. |
| | +UUMQTTC: 4,1,0,"u-blox/special" | |
| AT+UMQTTC=9,1,0,"module/special",21 | > | Send a Publish message with special characters in the payload. |
| "this is an example"<CR> | OK | |
| | +UUMQTTC: 2,1 | |
| | +UUMQTTC: 6,1 | Notification of the received publish message. |
| AT+UMQTTC=6,1 | +UMQTTC: 6,0,35, 14,"module/special",21,""this is an example"<br>"<br>OK | Read the received publish message, the quotation marks and the carriage return are displayed. |

## 6.6 Ping the MQTT broker

The ping command starts a session of ping requests to the broker server. The ping requests are sent at intervals, the length of the interval depends on the inactivity timeout (keep-alive time) set when configuring the MQTT profile.

| Command | Response | Description |
|---|---|---|
| AT+UMQTT=10,30 | OK | Configure the inactivity timeout as 30 s. |
| AT+UMQTTC=1 | OK | Connect to the broker and start a MQTT session. |
| | +UUMQTTC: 1,1 | |
| AT+UMQTTC=8,1 | OK | Start a “ping loop”.<br>A PINGREQ packet is sent to the broker when there is no activity with the broker, in this example after 24 s of inactivity a PINGREQ packet is sent and PINGRESP is received.<br>The ping request is approximately triggered after 80% of the keep alive time. |
| | +UUMQTTC: 8,0 | Notification of a ping failure, the broker is not responding. |

## 6.7 Last will packet

To see the last will publish message, two modules shall start a MQTT session with the same gateway. For the first module, before starting a MQTT session, the last will parameter shall be configured; see section 6.1.2. The second module shall subscribe to the last will topic of the first module.

| Command | Response | Description |
|---|---|---|
| **Module #1** | | |
| AT+UMQTTC=1 | OK | Connect to the broker and start a MQTT session. |
| | +UUMQTTC: 1,1 | |
| **Module #2** | | |
| AT+UMQTTC=1 | OK | Connect to the same broker and start a MQTT session. |
| | +UUMQTTC: 1,1 | |

| Command | Response | Description |
|---|---|---|
| AT+UMQTTC=4,0,"u-blox/publish" | OK<br>+UUMQTTC: 4,1,0,"u-blox/publish" | Subscribe to the last will topic "u-blox/publish". |
| **Module #1** | | |
| AT+CFUN=4 | OK<br>+UUMQTTC: 0,101 | Simulate a network error.<br>The URC notifies that the network connection is lost. |
| **Module #2** | | |
| | +UUMQTTC: 6,1 | Notification of the received publish message. |
| AT+UMQTTC=6,1 | +UMQTTC: 6,0,37,14,"u-blox/publish",23,"Unrequested disconnect."<br>OK | Read the received last will publish message. |

## 6.8 Debug

If the broker returns errors with the +UUMQTTC: x,0 URC, it is possible to investigate the type of error using the +UMQTTER AT command.

| Command | Response | Description |
|---|---|---|
| AT+UMQTTC=1 | OK<br>+UUMQTTC: 1,0 | Unsuccessful session start. |
| AT+UMQTTER | +UMQTTER: 13,50<br>OK | Error code 50 is “PSD or CSD connection not established”, that means the context is not active. |

## 6.9 Secure MQTT

Configure a secure manager profile before starting a secure MQTT session (using the TLS encryption protocol). For more details, see section 3.

The following example shows how to configure the MQTT profile before starting a secure session with the broker. Only the secure manager profile and the remote port must be configured; the other MQTT commands will behave as in the case of unencrypted session.

| Command | Response | Description |
|---|---|---|
| AT+UMQTT=11,1,2 | OK | Enable the secure MQTT option using the USECMNG profile 2. |
| AT+UMQTT=2,"192.168.105.30",8883 | OK | Set the remote MQTT broker IP address and port. The default port for secure MQTT is 8883. |
| AT+UMQTTC=1 | OK<br>+UUMQTTC: 1,1 | Connect to the broker and start a secure MQTT session. |

# 7 MQTT-SN

☞ Make sure to follow the steps in section 2 before using the AT commands in this section. This is necessary because a PS data connection must be activated before using MQTT-SN AT commands.

## 7.1 Basic setup

### 7.1.1 Default and minimal configuration

The configuration required to start a MQTT-SN session depends on the gateway configuration, most importantly, the MQTT-SN remote server information. Before starting a session, be sure to correctly set up the module with the gateway configuration.

| Command | Response | Description |
|---|---|---|
| AT+CMEE=2 | OK | Set verbose error result codes. |
| AT+UMQTTSN? | +UMQTTSN: 0,"357862090033897"<br>+UMQTTSN: 1,"",1883<br>+UMQTTSN: 2,"",1883<br>+UMQTTSN: 4,0<br>+UMQTTSN: 5,0<br>+UMQTTSN: 6,""<br>+UMQTTSN: 7,0,""<br>+UMQTTSN: 8,0<br>+UMQTTSN: 9,0<br>OK | Read the current profile configuration.<br>All the reported values can be modified; see the AT commands manual [3] for a detailed description.<br>☞ The default client id value is the IMEI of the module because it guarantees the uniqueness of the client to the server. |
| AT+UMQTTSN=2,"192.168.105.30",10000 | OK | Set the IP address and port of the remote MQTTSN gateway.<br>Alternatively, the gateway's server name can be set with the AT+UMQTTSN=1 command. |

### 7.1.2 Last will configuration

The "last will" parameters configure the message that the MQTT-SN clients connected to the gateway will receive in case of a module disconnection due to an error. Following is a set up example.

| Command | Response | Description |
|---|---|---|
| AT+UMQTTSN=4,1 | OK | Set the last will QoS level to 1. |
| AT+UMQTTSN=6,"u-blox/publish" | OK | Set the last will topic. |
| AT+UMQTTSN=7,"Unrequested disconnect." | OK | Set the last will message. |

### 7.1.3 Profile management

| Command | Response | Description |
|---|---|---|
| AT+UMQTTSNNV=2 | OK | Store the current MQTT-SN client profile parameters to the NVM. |
| AT+UMQTTSNNV=0 | OK | Restore MQTT-SN client profile parameters to the factory-programmed setting. |
| AT+UMQTTSN? | +UMQTTSN: 0,"357862090033897"<br>+UMQTTSN: 1,"",1883<br>+UMQTTSN: 2,"",1883<br>+UMQTTSN: 4,0<br>+UMQTTSN: 5,0<br>+UMQTTSN: 6,"" | |

| Command | Response | Description |
|---|---|---|
| | +UMQTTSN: 7,0,""<br>+UMQTTSN: 8,0<br>+UMQTTSN: 9,0<br>OK | |
| AT+UMQTTSNNV=1 | OK | Set MQTT-SN client profile parameters to values previously stored in the NVM. |
| AT+UMQTTSN? | +UMQTTSN: 0,"357862090033897"<br>+UMQTTSN: 1,"",1883<br>+UMQTTSN: 2,"192.168.105.30",1883<br>+UMQTTSN: 4,0<br>+UMQTTSN: 5,0<br>+UMQTTSN: 6,""<br>+UMQTTSN: 7,0,""<br>+UMQTTSN: 8,0<br>+UMQTTSN: 9,0<br>OK | |

### 7.1.4 Internal PDP context mapping

As an alternative to the default behavior, it is also possible to manually configure a mapping between the embedded MQTT-SN client and another PDP context (different than default CID 1).

| Command | Response | Description |
|---|---|---|
| AT+UMQTTSN=20,2,1 | OK | Mapping the embedded MQTT-SN client to use the context ID 2. With preferred protocol type 1 (thus, IPv6).<br>☞ This configuration is optional. Necessary only if the embedded MQTT-SN client needs to use a PDP context different than CID 1. |

## 7.2 Start and end a MQTT-SN session

See the default and minimal configuration described in section 7.1.1, to configure the MQTT-SN profile before starting a connection.

| Command | Response | Description |
|---|---|---|
| AT+UMQTTSNC=1 | OK<br>+UUMQTTSNC: 1,1 | Connect to the gateway and start a MQTT-SN session. |
| AT+UMQTTSNC=0 | OK<br>+UUMQTTSNC: 0,1 | Disconnect from the gateway, end of the MQTT-SN session. |

## 7.3 Subscribe to a normal topic

Example of MQTT-SN session subscription to a topic.

| Command | Response | Description |
|---|---|---|
| AT+UMQTTSNC=5,0,0,"room/temperature" | OK<br>+UUMQTTSNC: 5,1,0,1 | Subscribe to a normal topic (0) with requested QoS level set to 0.<br>The gateway granted QoS level is 0 and the topic ID for "room/temperature" is 1. |

## 7.4 Publish and read a message to a topic

In this MQTT-SN session the module publishes a message to the topic and receives the published message (assuming it is subscribed to topic of the published message).

| Command | Response | Description |
|---|---|---|
| AT+UMQTTSNC=4,0,0,0,0,"1","20 degrees Celsius" | OK<br>+UUMQTTSNC: 4,1 | Publish the "20 degrees Celsius" message to the topic ID 1 with requested QoS level and retain value set to 0. |
| | +UUMQTTSNC: 9,1 | Notification of the received publish message. |
| AT+UMQTTSNC=9,1 | +UMQTTSNC: 9,1,0,19,1,"1",18,"20 degrees Celsius"<br>OK | Read the received publish message. |

## 7.5 Unsubscribe from a normal topic

| Command | Response | Description |
|---|---|---|
| AT+UMQTTSNC=6,0,"1" | OK<br>+UUMQTTSNC: 6,1 | Unsubscribe from the subscribed topic ID '1'. |

## 7.6 Register to a topic and publish a message to the same topic

The following example differs from the previous one only for the non-receipt of publish message since the module is not subscribed to the topic.

| Command | Response | Description |
|---|---|---|
| AT+UMQTTSNC=2,"kitchen/temperature" | OK<br>+UUMQTTSNC: 2,1,2 | Register to a normal topic.<br>The returned topic ID for "room/temperature" is 2. |
| AT+UMQTTSNC=4,1,0,0,0,"2","25 degrees Celsius" | OK<br>+UUMQTTSNC: 4,1 | Publish the "25 degrees Celsius" message to the "kitchen/temperature" topic using the above topic ID. |

## 7.7 Subscribe to a short topic name and publish a message to the same topic

The short topic is composed of only 2 characters.

| Command | Response | Description |
|---|---|---|
| AT+UMQTTSNC=5,0,2,"aa" | OK<br>+UUMQTTSNC: 5,1,2,0 | Subscribe to a short topic (2) with requested QoS level set to 0.<br>The gateway granted QoS level is 0 |
| AT+UMQTTSNC=4,1,0,0,2,"aa","test" | OK<br>+UUMQTTSNC: 4,1 | Publish the "test" message to the "aa" topic with requested QoS level and retain value set to 0. |
| | +UUMQTTSNC: 9,1 | Notification of the received publish message. |
| AT+UMQTTSNC=9,1 | +UMQTTSNC: 9,0,2,6,2,"aa",4,"test"<br>OK | Read the received publish message. |
| AT+UMQTTSNC=6,2,"aa" | OK<br>+UUMQTTSNC: 6,1 | Unsubscribe from the previously subscribed topic. |

## 7.8 Last will

To see the last will publish message, two modules shall start a MQTT-SN session with the same gateway. For the first module, before starting a MQTT-SN session, the last will parameter shall be configured; see last will configuration in section 7.1.2. The second module shall subscribe to the last will topic of the first module.

| Command | Response | Description |
|---|---|---|
| **Module #1** | | |
| AT+UMQTTSNC=1 | OK<br>+UUMQTTSNC: 1,1 | Connect to the gateway and start a MQTT-SN session. |
| **Module #2** | | |
| AT+UMQTTSNC=1 | OK<br>+UUMQTTSNC: 1,1 | Connect to the same gateway and start a MQTT-SN session. |
| AT+UMQTTSNC=5,0,0,"u-blox/publish" | OK<br>+UUMQTTSNC: 5,1,0,1 | Subscribe to the last will topic "u-blox/publish". |
| **Module #1** | | |
| AT+CFUN=4 | OK<br>+UUMQTTSNC: 0,101 | Simulate a network error.<br>The URC notifies that the network connection is lost. |
| **Module #2** | | |
| | +UUMQTTSNC: 9,1 | Notification of the received publish message. |
| AT+UMQTTSNC=9,1 | UMQTTSNC: 9,0,0,24,1,"1",23,"Unrequested disconnect."<br>OK | Read the received last will publish message. |

## 7.9 Error handling

If the gateway returns errors with the +UUMQTTSNC: x,0 URC, it is possible to investigate the type of error using the +UMQTTSNER AT command.

| Command | Response | Description |
|---|---|---|
| AT+UMQTTSNC=5,1,0,"kitchen/temperature" | OK<br>+UUMQTTSNC: 5,0 | Unsuccessful subscribe. |
| AT+UMQTTSNER | +UMQTTSNER: 14,21<br>OK | Error code 21 is "Timeout error" that means the gateway did not replay to the subscribe request. |

## 7.10 Secure MQTT-SN

Configure a secure manager profile before starting a secure MQTT-SN session (using the DTLS encryption protocol). For further details, see the section 3.

The following example shows how to configure the MQTT-SN profile before starting a secure session with the gateway. Only the secure manager profile and the remote port must be configured; the other MQTT-SN commands will behave as in the case of an unencrypted session.

| Command | Response | Description |
|---|---|---|
| AT+UMQTTSN=9,1,2 | OK | Enable the secure MQTT-SN option using the USECMNG profile 2. |
| AT+UMQTTSN=2,"192.168.105.30",10001 | OK | Set the remote MQTT-SN gateway IP address and port. |
| AT+UMQTTSNC=1 | OK<br>+UMQTTSNC: 1,1 | Connect to the gateway and start a secure MQTT-SN session. |

## 7.11 MQTT Anywhere

MQTT Anywhere is a u-blox IoT communication SIM-based LPWA service that can operate around the world without the need for specific cellular agreements with multiple MNOs.

This service uses the MQTT-SN protocol, and it is directly integrated into u-blox products. Additionally, devices are authenticated via the hardware IoT SIM card, ensuring that the user traffic is never exposed to the public internet. Device payloads can be enriched and transformed using the Data Flow Manager within u-blox Thingstream, which also provides integration with virtually any 3rd party enterprise system or IoT platform.

☞ When using a Thingstream SIM card (IoT SIM card), be aware that there are two different APNs available:

- APN 'tsudp' allows only connectivity to Thingstream MQTT Anywhere server and it is mandatory to access this service;
- APN 'tsiot' allows generic data traffic:
- For more information on Thingstream and u-blox services offering please visit this webpage.

☞ Please be aware that security services can be used with the 'tsudp' APN and the Thingstream SIM card only if the AT+USECOPCMD="cfgipv4",4 command is issued. With this additional configuration, the security functionalities are enabled via a u-blox proxy enclose in the Thingstream portal.

Additional details on this topic are available on the product webpage.

In addition to the MQTT-SN basic settings, the MQTT Anywhere service required the configuration of a unique client ID and the clean session. See an example of these configurations in the table below.

| Command | Response | Description |
|---|---|---|
| `AT+UMQTTSN=0,"identity:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"` | `OK` | To ensure a unique value, the identity value of the SN thing needs to be used as the client ID.<br>☞ The identity value can be found on the management console page of u-blox Thingstream platform. |
| `AT+UMQTTSN=8,600` | `OK` | Set the connection duration (in seconds). |
| `AT+UMQTTSN=10,1` | `OK` | Set the MQTT-SN clean session. |

See a complete example of the 'MQTT Anywhere' configuration on the IoT Communication-as-a-Service guide webpage.

## 7.12 MQTT Flex

MQTT Flex is a u-blox IoT communication BYO-SIM LPWA service that provides the flexibility to choose your own cellular connectivity, combined with the advantages of IoT Communication-as-a-Service. This service uses the MQTT-SN protocol, and it is directly integrated into u-blox products. Additionally, devices are securely authenticated via the client key and certificate generated by the Thingstream platform, ensuring that the user traffic is never exposed to the public internet. Device payloads can be enriched and transformed using the Data Flow Manager within u-blox Thingstream, which also provides integration with virtually any 3rd party enterprise system or IoT platform.

Additional details on this topic are available on the product webpage.

Differently from the MQTT Anywhere configuration, the MQTT Flex service required the configuration of the security profile together with the configuration of the MQTT-SN internal application. All the relevant steps are described in the following sections.

### 7.12.1 Specify the client key and certificate to be used

This example assumes the client key and certificate generated by the Thingstream platform associated with the Flex Thing are correctly stored in the file system of the module using the +UDWNFILE AT command and using the file names device.pem for the certificate and device.key for the key.

| Command | Response | Description |
|---|---|---|
| AT+USECMNG=1,1,"MQTTFlex_cert", "device.pem" | +USECMNG: 1,1,"MQTTFlex_cert ","73A0...8D"<br>OK | Import the client certificate from the file "device.pem" stored on the file system. |
| AT+USECMNG=1,2,"MQTTFlex_key", "device.key" | +USECMNG: 1,2,"MQTTFlex_key ","73A0...8D"<br>OK | Import the client key from the file "device.key" stored on the file system. |

### 7.12.2 Configure the DTLS security profile

The following commands set up the DTLS security profile which the MQTT-SN client will make use of.

| Command | Response | Description |
|---|---|---|
| AT+USECPRF=0 | OK | Reset (set to factory-programmed value) all the parameters of security profile #0. |
| AT+USECPRF=0,0,0 | OK | Disable the certificate validation; the server certificate will not be checked or verified. |
| AT+USECPRF=0,1,3 | OK | Select TLSv1.2 as SSL/TLS to use. |
| AT+USECPRF=0,2,99,"C0","30" | OK | Select a specific cipher suite for profile #0. |
| AT+USECPRF=0,5,"MQTTFlex_cert" | OK | Select trusted client certificate internal name for profile #0. |
| AT+USECPRF=0,6,"MQTTFlex_key" | OK | Select trusted client key internal name for profile #0. |

### 7.12.3 Configure the MQTT-SN client

| Command | Response | Description |
|---|---|---|
| AT+UMQTTSN=0,"device:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" | OK | To ensure a unique value, the device value of the SN thing needs to be used as the client ID.<br>☞ The device value can be found on the management console page of u-blox Thingstream platform. |
| AT+UMQTTSN=1,"mqtt-flex.thingstream.io",2443 | OK | Configure the MQTT-SN gateway endpoint. |
| AT+UMQTTSN=9,1,0 | OK | Set the MQTT-SN internal client to use the required DTLS profile. |

Once the MQTT-SN client configuration is completed, it is possible to use the feature and perform the connection like a generic MQTT-SN application (e.g., with the AT+UMQTTSNC=1 command).

Please note that a complete example of the 'MQTT Flex' configuration can be seen on the IoT Communication-as-a-Service guide webpage.

# 8 CoAP

CoAP is an application layer protocol based on UDP for resource-constrained internet devices as described in RFC 7252 [11].

Main CoAP features are:

- Web protocol fulfilling M2M requirements in constrained environments.
- UDP (RFC 768 [10]) binding with optional reliability supporting unicast and multicast requests.
- Asynchronous message exchanges.
- Low header overhead and parsing complexity.
- URI and Content-type support.
- Simple proxy and caching capabilities.
- A stateless HTTP mapping, allowing proxies to be built providing access to CoAP resources via HTTP in a uniform way or for HTTP simple interfaces to be realized alternatively over CoAP.
- Security binding to Datagram Transport Layer Security (DTLS) [12].
- Transfer block-wise as described into RFC 7959 [13].
- Transmission over TCP and TLS: described into RFC 8323 [14].

Implementation details:

- Maximum URI length is 785 characters. URI_HOST, URI_PATH and URI_QUERY are limited to 255 characters as per RFC 7252 [11].
- Maximum supported payload in uplink is 512 bytes. Use block transfer for data block greater than 512 bytes.
- In DTLS and TLS connection, the OK final result code will be returned only after the security connection handshake is completed successfully.
- It is allowed only one CoAP transmission until it is completed.
- Only 4 profiles can be stored.

## 8.1 Good practices on writing CoAP application

To create a good CoAP application keep the following in mind:

- **Register the module to the network and establish a data connection**: The module should be connected to the network and have a data connection to send CoAP commands. This is a mandatory step.
- **Configure a profile and store it**: Create a basic configuration that can be stored for further restart.
- **Restore the profile**: This can be used to avoid to trigger always the same commands at the module boot and it can be used to verify if the current configuration is aligned with one expected.
- **Configure and send CoAP commands**: Change the configuration parameter in order to send CoAP commands.
- **Wait and parse URCs**: Wait until the final +UCOAPCR URC is received and stored, then parse the +UCOAPCD URC to have a complete reply from the server.
- **Error handling**: Procedure that can be used to debug errors from the server or from configuration.

This flow is described in the Figure 8.

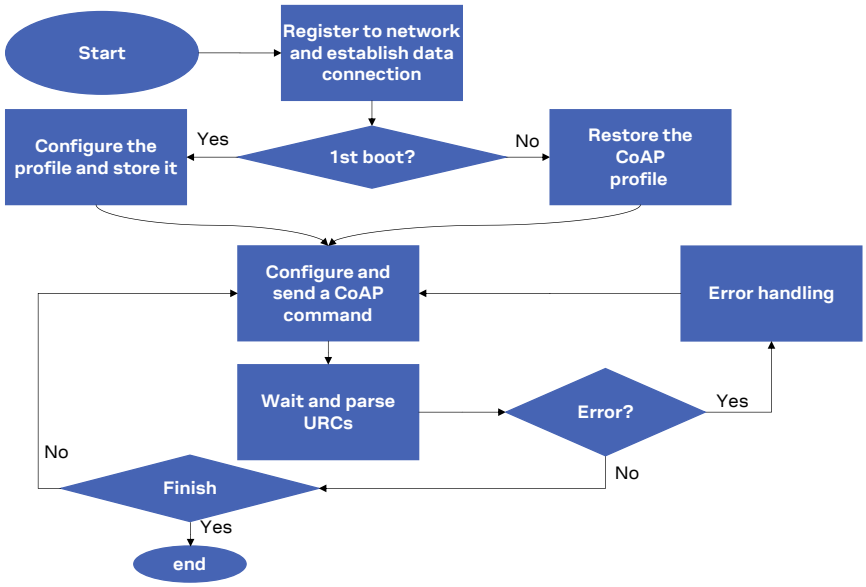



**Figure 8: CoAP workflow**

## 8.2 Basic setup

☞ Make sure to follow the steps in section 2 before using the AT commands in this section. This is necessary because a PS data connection must be activated before using CoAP AT commands.

### 8.2.1 Current configuration

| Command | Response | Description |
|---|---|---|
| AT+CMEE=2 | OK | Set verbose error result codes. |
| AT+UCOAP=4,1 | OK | Set current profile as valid. |
| AT+UCOAP=2,0,1<br>AT+UCOAP=2,1,1<br>AT+UCOAP=2,2,1<br>AT+UCOAP=2,3,1 | OK<br>OK<br>OK<br>OK | Enable automatic recognition of URI_HOST, URI_PORT, URI_PATH and URI_QUERY directly from URI. |
| AT+UCOAP=2,4,1 | OK | Set content format for PUT/POST as "Plain/Text". |

### 8.2.2 Profile management

| Command | Response | Description |
|---|---|---|
| AT+UCOAP=6,0 | OK | Store current profile to be stored as profile number 0. |
| AT+UCOAP=7 | +UCOAP: "URI_STR","<server_uri>"<br>+UCOAP: "OPT_MASK",7<br>+UCOAP: "PROFILE_NUM",0<br>+UCOAP: "STATUS_FLAG",1<br>+UCOAP: "USECMNG PROFILE",0<br>+UCOAP: "RAI FLAG",0<br>+UCOAP: INVALID PROFILE NUMBER 1<br>+UCOAP: INVALID PROFILE NUMBER 2<br>+UCOAP: INVALID PROFILE NUMBER 3<br>OK | Check all stored profile configuration. In this configuration only profile 0 is a valid profile that can be used. |

| Command | Response | Description |
|---|---|---|
| AT+UCOAP=5,0 | OK | Restore profile number 0 as current profile. |
| AT+UCOAP? | +UCOAP: "URI_STR","<server_uri>"<br>+UCOAP: "OPT_MASK",7<br>+UCOAP: "PROFILE_NUM",0<br>+UCOAP: "STATUS_FLAG",1<br>+UCOAP: "USECMNG PROFILE",0<br>+UCOAP: "RAI FLAG",0<br>OK | Verify that current profile is configured as expected. |

### 8.2.3 Internal PDP context mapping

As an alternative to the default behavior, it is also possible to manually configure a mapping between the embedded CoAP client and another PDP context (different than default CID 1).

| Command | Response | Description |
|---|---|---|
| AT+UCOAP=20,2,1 | OK | Mapping the embedded CoAP client to use the context ID 2. With preferred protocol type 1 (thus, IPv6).<br>☞ This configuration is optional. Necessary only if the embedded CoAP client needs to use a PDP context different than CID 1. |

## 8.3 Basic operation

### 8.3.1 GET

| Command | Response | Description |
|---|---|---|
| AT+UCOAP=1,"<server_uri>" | OK | Set URI to be used in GET request. In this example we use coap.me server.<br>Reply can be different in the future. |
| AT+UCOAP=2,5,1 | OK | Set message to be NON confirmable |
| AT+UCOAPC=1 | OK<br>+UCOAPCD: 2,0,"776F726C64",0<br>+UCOAPCR: 1,1 | Command trigger reply from the server in hexadecimal format that corresponds to “world”. |

### 8.3.2 PUT/POST

| Command | Response | Description |
|---|---|---|
| AT+UCOAP=1,"<server_uri>" | OK | Set URI to be used in PUT/POST request. In this example we use coap.me server.<br>Reply can be different in the future. |
| AT+UCOAP=2,5,0 | OK | Set message to be CONfirmable, so server should reply with an ACK |
| AT+UCOAPC=3,"736f6d655f74657874",0 | OK<br>+UCOAPCD: 2,0,"505554204F4B",0<br>+UCOAPCR: 3,1 | Server replied in hex that correspond to “PUT OK” when PUT command is triggered. |
| AT+UCOAPC=4,"736f6d655f74657874",0 | OK<br>+UCOAPCD: 2,0,"504F5354204F4B",0<br>+UCOAPCR: 4,1 | Server replied in hex that correspond to “POST OK” when POST command is triggered. |

### 8.3.3 Block transfer

| Command | Response | Description |
|---|---|---|
| AT+UCOAP=1,"<server_uri>" | OK | Set URI to be used in PUT request. In this example we use coap.me server.<br>Reply can be different in the future. |
| AT+UCOAPC=3,"4d657373616765206e756d626572203020666972737420 6d65737361676520 6f6620736572696520737562646976696465642069 6e746f2036342062797465 73",0,0,1 | OK<br>+UCOAPCD: 2,0,"",0<br>+UCOAPCR: 3,1 | Send data separated into two messages.<br>First message is 64 bytes and is set that more blocks will be transmitted, so the server can be prepared for adding other data. |
| AT+UCOAPC=3,"4d657373616765206e756d626572203120666972737420 6d65737361676520 6f6620736572696520737562646976696465642069 6e746f2036342062797465 73",0,1,0 | OK<br>+UCOAPCD: 2,0,"",0,1,64<br>+UCOAPCR: 3,1 | Second message (about 64 bytes) and the message number is different. No more blocks will be transmitted after. |

### 8.3.4 TCP connection

| Command | Response | Description |
|---|---|---|
| AT+UCOAP=1,"<server_uri>" | OK | Set URI to be used in GET request. In this example we use a local server that supports CoAP over TCP. |
| AT+UCOAPC=1 | OK<br>+UCOAPCD: 2,0,"5468697320 697320612074657374207365 7266572206D61646520776974 68206C6962636F61702028736 5652068747470733A2F2F6C69 62636F61702E6E6574290A436 F707972696768742028432920 323031302D2D32303230204F6 C616620426572676D616E6E20 3C626572676D616E6E40747A6 92E6F72673E20616E64206F74 686572730A0A",0<br>+UCOAPCR: 1,1 | Command trigger reply from server in hex that corresponds to "This is a test server made with libcoap (see https://libcoap.net)<br>Copyright (C) 2010--2020 Olaf Bergmann <bergmann@tzi.org> and others". |

## 8.4 Error handling

### 8.4.1 Configuration error

| Command | Response | Description |
|---|---|---|
| AT+UCOAPC=1 | +CME ERROR: operation not allowed | The connection is started but some configuration parameters are not configured properly. |
| AT+UCOAPER | +UCOAPER: 15,17<br>OK | Returns the error code of the latest CoAP operation. Check the error result codes in the appendix of the AT commands manual [3]. In this case, “Current profile invalid”, meaning the profile is not configured properly. |

### 8.4.2 Error on server reply

| Command | Response | Description |
|---|---|---|
| AT+UCOAPC=3,"736f6d655f74657874",0 | OK<br>+UCOAPCD: 4,0,"4572726F72 20342E30353A204D6574686F6 4206E6F7420737570706F7274 65642068657265",0<br>+UCOAPCR: 3,1 | Command is executed successfully.<br>The server replies with <error_code> and <error_description>, which is shown as “Error 4.05: Method not supported here”. |

## 8.5 Secure connection

### 8.5.1 Basic configuration

Configure a secure manager profile before trigger a CoAP command with secure connection. For further details, see section 3.

| Command | Response | Description |
|---|---|---|
| AT+UCOAP=8,0 | OK | Use profile 0 in a secure connection. |

### 8.5.2 DTLS connection

| Command | Response | Description |
|---|---|---|
| AT+UCOAP=1,"<server_uri>" | OK | Set the URI to be used in GET request.<br>In this example it is used a server configured with proper security setup. |
| AT+UCOAPC=1 | OK<br>+UCOAPCD: 2,0,"5468697320697320612074657373742073657276657220 6D61646520776974 68206C6962636F61702028736565206874747073 3A2F2F6C6962636F61702E6E6574290A436F707972696768742028432920323031302D2D32303230204F6C616620426572676D616E6E203C626572676D616E6E40747A692E6F72673E20616E64206F74686572730A0A",0<br>+UCOAPCR: 1,1 | Trigger reply from server in hex that corresponds to "This is a test server made with libcoap (see https://libcoap.net)<br>Copyright (C) 2010--2020 Olaf Bergmann <bergmann@tzi.org> and others". |

### 8.5.3 TLS connection

Even if AT commands are the same, unlike the DTLS connection shown in the previous example, the TLS connection uses TCP protocol and therefore, a dedicated URI must be used for security using TCP (TLS).

☞ Ensure the CoAP server is supporting the TCP extension. This is not guaranteed because the native protocol is UDP. Fort this reason, a dedicated URI should be used, and the user must verify also that the server supports TLS connections.

| Command | Response | Description |
|---|---|---|
| AT+UCOAP=1,"<server_uri>" | OK | Set the URI to be used in GET request.<br>In this example we use a server that is configured with proper security setup. |
| AT+UCOAPC=1 | OK<br>+UCOAPCD: 2,0,"5468697320697320612074657373742073657276657220 6D61646520776974 68206C6962636F61702028736565206874747073 3A2F2F6C6962636F61702E6E6574290A436F707972696768742028432920323031302D2D32303230204F6C616620426572676D616E6E203C626572676D616E6E40747A692E6F72673E20616E64206F74686572730A0A",0<br>+UCOAPCR: 1,1 | Trigger reply from server in hexadecimal format that corresponds to "This is a test server made with libcoap (see https://libcoap.net)<br>Copyright (C) 2010--2020 Olaf Bergmann <bergmann@tzi.org> and others". |

# 9 FTP

☞ Make sure to follow the steps in section 2 before using the AT commands in this section. This is necessary because a PS data connection must be activated before using FTP AT commands.

| Command | Response | Description |
|---|---|---|
| AT+UFTP=1,"ftp.u-blox.com"<br>AT+UFTP=2,"anonymous"<br>AT+UFTP=3,"password"<br>AT+UFTP=6,1 | OK<br>OK<br>OK<br>OK | Parameter configuration for FTP server connection. These parameters will be set:<br>• FTP server hostname<br>• FTP username<br>• FTP password<br>• FTP connection mode (PASSIVE connection). Most FTP servers prefer the PASSIVE mode due to security issues. |
| AT+UFTP=20,2,1 | OK | Map the embedded FTP client to use the context ID 2. With preferred protocol type 1 (thus, IPv6).<br>☞ This configuration is optional. Necessary only if the embedded FTP client needs to use a PDP context different than default CID 1. |
| AT+UDNSRN=0,"ftp.u-blox.com" | +UDNSRN: "195.34.89.241"<br>OK | Hostname resolution. |
| | | Connect to the server and manage the FTP connection using the +UFTPC AT command. Let's start connecting to the server. |
| AT+UFTPC=1 | OK<br>+UUFTPCR: 1,1 | The +UUFTPCR URC provides the FTP command result (the last parameter provides the result, 1 if is successfully performed). |
| AT+UFTPC=13 | OK | Request the file list on the server. |
| | +UUFTPCD: 13,194,"-rw-r--r-- 1 ftp ftp 1037 Aug 5 09:45 dat_000<br>-rw-r--r-- 1 ftp ftp 21041 Aug 5 09:12 data.zip<br>-rw-r--r-- 1 ftp ftp 12 Aug 5 09:42 xlog.zip<br>" | The +UUFTPCD URC provides the FTP data. |
| | +UUFTPCR: 13,1 | |
| AT+UFTPC=10,"uploads" | OK | Directory creation on the FTP server. |
| | +UUFTPCR: 10,1 | |
| AT+UFTPC=13 | OK | Request again the file list. |
| | +UUFTPCD: 13,258,"-rw-r--r-- 1 ftp ftp 1037 Aug 5 09:45 dat_000<br>-rw-r--r-- 1 ftp ftp 21041 Aug 5 09:12 data.zip<br>drwxr-xr-x 2 ftp ftp 4096 Aug 5 09:48 uploads<br>-rw-r--r-- 1 ftp ftp 12 Aug 5 09:42 xlog.zip<br>" | |
| | +UUFTPCR: 13,1 | |
| | | Change directory to directory name "uploads". |
| AT+UFTPC=8,"uploads" | OK | |

| Command | Response | Description |
|---|---|---|
| | +UUFTPCR: 8,1 | ☞ Use AT+UFTPC=8,".." to return back in the parent directory. |
| AT+UFTPC=5,"gps_positions","gps_positions" | OK<br>+UUFTPCR: 5,1 | Upload a file from the module to the FTP server from the local file system of the module (in this example filename "gps_positions"). |
| AT+UFTPC=5,"gps_positions","gps_positions",250 | OK<br>+UUFTPCR: 5,1 | Restart the upload file from the module to FTP server from the local module file system (in this example filename "gps_positions"), starting from byte 250.<br>☞ The FTP server should support the REST command to support these functionalities. The server should write the file starting from byte indicated. |
| AT+UFTPC=13 | OK<br>+UUFTPCD: 13,70,"-rw-r--r-- 1 ftp ftp 176673 Aug 5 10:03 gps_positions"<br>+UUFTPCR: 13,1 | File list and information request. |
| AT+UFTPC=14 | OK<br>+UUFTPCD: 14,15,"gps_positions"<br>+UUFTPCR: 14,1 | File list request. |
| AT+UFTPC=8,".." | OK<br>+UUFTPCR: 8,1 | Return to the parent directory. |
| AT+UFTPC=4,"data.zip","data.zip" | OK<br>+UUFTPCR: 4,1 | Download a file from the FTP server to the local file system of the module. |
| AT+UFTPC=4,"data.zip","data.zip",1 | OK<br>+UUFTPCR: 4,1 | Restart the file download from the FTP server to the local module file system from the latest byte saved on the file system (this is automatically calculated). The data received is written after the latest byte available on the file system. |
| AT+UFTPC=0 | OK<br>+UUFTPCR: 0,1 | FTP server disconnection. |

# 9.1 Direct link

## 9.1.1 Retrieve a file from FTP server

| Command | Response | Description |
|---|---|---|
| AT+UFTP=1,"ftp.u-blox.com"<br>AT+UFTP=2,"anonymous"<br>AT+UFTP=3,"password"<br>AT+UFTP=6,1 | OK<br>OK<br>OK<br>OK | Parameter configuration for FTP server connection. These parameters will be set:<br>• FTP server hostname<br>• FTP username<br>• FTP password<br>• FTP connection mode (PASSIVE connection) |
| AT+UDNSRN=0,"ftp.u-blox.com" | +UDNSRN: "195.34.89.241"<br>OK | Hostname resolution. |
| | | Connect to the server and manage the FTP connection using the +UFTPC AT command. |
| AT+UFTPC=1 | OK<br>+UUFTPCR: 1,1 | The +UUFTPCR URC is returned when the connection is established. |
| AT+UFTPC=6,"file_to_retrieve" | CONNECT | Send to the FTP server a RETRIEVE file request for file_to_retrieve. |

| Command | Response | Description |
|---|---|---|
| | | The CONNECT intermediate result code means the direct link mode is activated: the data received from FTP connection will be redirected to the serial port. |
| AT+UFTPC=6,"file_to_retrieve",250 | CONNECT | Restart a RETRIEVE file request for file_to_retrieve file from byte 250.<br>The CONNECT intermediate result code means the direct link mode activation: the data received from FTP connection is redirected to the serial port.<br>The data reception begins with the byte indicated. |
| +++ | DISCONNECT<br>OK | ⚠ When the file has entirely been retrieved the module does not exit from the direct link mode. It is necessary to manually exit using the "+++" escape sequence. |
| | +UUFTPCR: 6,1 | The +UUFTPCR URC notifies how the retrieve operation has been concluded (1 means success). |

### 9.1.2 Aborting retrieve file request

| Command | Response | Description |
|---|---|---|
| +++ | DISCONNECT<br>OK | If entering "+++" escape sequence before the requested file has been entirely retrieved from FTP server, the module exits from the direct link. |
| | +UUFTPCR: 6,0 | The +UUFTPCR URC notifies that the retrieve operation has not been concluded successfully (0 means fail). |

### 9.1.3 Store a file on FTP server

| Command | Response | Description |
|---|---|---|
| AT+UFTP=1,"ftp.u-blox.com"<br>AT+UFTP=2,"anonymous"<br>AT+UFTP=3,"password"<br>AT+UFTP=6,1 | OK<br>OK<br>OK<br>OK | Parameter configuration for FTP server connection. These parameters will be set:<br>• FTP server hostname;<br>• FTP username;<br>• FTP password;<br>• FTP connection mode (PASSIVE connection). |
| AT+UDNSRN=0,"ftp.u-blox.com" | +UDNSRN: "195.34.89.241"<br>OK | Hostname resolution. |
| | | Connect to the server and manage the FTP. Connection using the +UFTPC command. Let's start connecting to the server. |
| AT+UFTPC=1 | OK<br>+UUFTPCR: 1,1 | The +UUFTPCR URC is returned when the connection is established. |
| AT+UFTPC=7,"file_to_store" | CONNECT | Send to FTP server a STORE file request for file_to_store.<br>The CONNECT intermediate result code means the direct link mode is activated: the data sent through the serial port will be redirected to the FTP server through the FTP connection. |
| AT+UFTPC=7,"file_to_store",250 | CONNECT | Restart the STORE file request for file_to_store from byte 250.<br>The CONNECT intermediate result code means the direct link mode activation: the data sent through the serial port is redirected to the FTP server through the FTP connection. |

| Command | Response | Description |
|---|---|---|
| | | The data is written on the FTP server starting from byte indicated.<br>☞ The FTP server should support REST command to support this functionality. |
| +++ | DISCONNECT<br>OK | When the data upload is completed use the "+++" escape sequence for exiting from the direct link mode. |
| | +UUFTPCR: 7,1 | The +UUFTPCR URC notifies if the STORE operation has been concluded successfully. |

### 9.1.4 About "+++" escape sequence use

To switch from the data mode to the command mode, the application shall send a proper escape sequence to the module.

The escape sequence “+++” is detected when it is received by the module in a single separate frame of 3 bytes in length. This will happen if the host waits 2 seconds after all data has been transmitted before issuing the “+++” string.

The application can configure the escape sequence using the following command:

- ATS2: this command sets the character used as the escape character (by default it is "+").

For more details on the AT commands, see the AT commands manual [3].

☞ If the host application shall send “+++” as the final part of the payload, an additional byte must be added to avoid false detection.

☞ If flow control is activated by the module (e.g., when data is transmitted over a congested or low throughput network), there is the risk that the escape sequence is queued in the host connectivity buffers and delivered to the module in frames also containing data payload. To avoid missed detection of the escape sequence, it is suggested to send the “+++” string when the COM port has CTS asserted/flow control disabled.

☞ The module does not recognize the escape sequence “+++” if a delay bigger than 500 ms is placed between the three “+” characters.

## 9.2 Using secure option

| Command | Response | Description |
|---|---|---|
| | | Parameters configuration for the FTP server connection in secure mode.<br>These parameters are set: |
| AT+UFTP=0,"123.213.132.231" | OK | • FTP server address |
| AT+UFTP=2,"myname" | OK | • FTP username |
| AT+UFTP=3,"mypwd" | OK | • FTP password |
| AT+UFTP=8,1 | OK | • FTP SSL encryption control channel enabled |
| AT+UFTP=12,1 | OK | • FTP SSL encryption data channel enabled |
| AT+UFTPC=1 | | FTP login. |
| | | Connect to the server and manage the FTP connection using the +UFTPC AT command. Let's start connecting to the server. |
| | OK | |
| | +UUFTPCR: 1,1 | The +UUFTPCR URC provides the FTP command result (the second parameter provides the result, 1 if is successfully performed). |

| Command | Response | Description |
|---|---|---|
| | | Some operators may not accept a secure FTP connection: |
| AT+UFTPC=1 | OK<br>+UUFTPCR: 1,0 | The URC provides the FTP command result: the second parameter is 0, an error has occurred. |
| AT+UFTPER | +UFTPER: 8,63<br>OK | Retrieving of error class and code:<br>• Error class 8: "Wrong FTP API usage"<br>• Error code 63: "Cannot set secure socket" |

☞ When the FTP client is using a secure connection, only the explicit mode is supported (ftpes://). Moreover, in the explicit mode, the secure connection will be established after the FTP connection (before login) on the same port of the control channel.

☞ When the FTP client is using a secure connection, the FTPS server may request that the session data of the control channel connection should be reused to establish secure connection on the data channel. In this case, the session resumption feature for the FTPS client shall be configured via <op_code>: 13 of the +USECPRF AT command.

## 9.3 Error handling

In case of errors returned in the last FTP operation, it is possible to investigate the type of error using the +UFTPER AT command.

| Command | Response | Description |
|---|---|---|
| AT+UFTPC=1 | OK<br>+UUFTPCR: 1,0 | Unsuccessful login. |
| AT+UFTPER | +UFTPER: 1,1<br>OK | Error code 1 is “User missing”. |

# 10 HTTP

☞ Make sure to follow the steps in section 2 before using the AT commands in this section. This is necessary because a PS data connection must be activated before using HTTP AT commands.

## 10.1 Basic setup

This section shows an example use of the u-blox proprietary +UHTTP and +UHTTPC AT commands. These commands are used for sending requests to a remote HTTP server, receiving the server responses, and transparently storing them in the file system. The supported methods are: HEAD, GET, DELETE, PUT, POST file, and POST data. For detailed AT command descriptions, see the AT commands manual [3].

| Command | Response | Description |
|---|---|---|
| AT+CMEE=2 | OK | Set verbose error result codes. |
| AT+UHTTP=0 | OK | Reset the HTTP profile #0. |
| AT+UHTTP=0,1,"httpbin.org"<br>AT+UHTTP=0,5,80 | OK<br>OK | Set the server domain name and port.<br>☞ HTTP server name (e.g., " httpbin.org "). The factory-programmed value is an empty text string. |
| AT+UHTTP=0,20,2,1 | OK | Mapping the embedded HTTP client profile 0 to use the context ID 2. With preferred protocol type 1 (thus, IPv6).<br>☞ This configuration is optional. Necessary only if the embedded HTTP client needs to use a PDP context different than default CID 1. |
| AT+UDNSRN=0,"httpbin.org" | +UDNSRN: "54.72.52.58"<br>OK | DNS resolution of httpbin.org. |
| AT+UHTTPC=0,0,"/","head.ffs" | OK<br>+UUHTTPCR: 0,0,1 | HEAD request of the default page and store the result into the "head.ffs" file on the local file system of the module. The +UUHTTPCR URC notifies the success/failure of the operation (in this example: success). |
| AT+UHTTPC=0,1,"/","get.ffs" | OK<br>+UUHTTPCR: 0,1,1 | GET request of the default page and store the result into the "get.ffs" file on the local file system of the module. The +UUHTTPCR URC notifies the success/failure of the operation (in this example: success). |
| AT+UHTTPC=0,5,"/post","post.ffs","name_post=MyName&age_post=30",0 | OK<br>+UUHTTPCR: 0,5,1 | POST request sending data using content-type application/x-www-form-urlencoded. The result is saved in the "post.ffs" file on the local file system of the module. The +UUHTTPCR notifies the success/failure of the operation (in this example: success). |
| AT+UHTTP=0,2,"test_user"<br>AT+UHTTP=0,3,"P455w0rd"<br>AT+UHTTP=0,4,1 | OK<br>OK<br>OK | Set the authentication for the HTTP server:<br>HTTP server username.<br>HTTP server password.<br>HTTP server authentication method (basic authentication). |
| AT+UHTTPC=0,1," /basic-auth/test_user/P455w0rd","get_auth.ffs" | OK<br>+UUHTTPCR: 0,1,1 | GET request returning information on authenticated user.<br>The page requires basic authentication.<br>The result is saved in "get_auth.ffs" file on the local file system of the module. The +UUHTTPCR URC |

| Command | Response | Description |
|---|---|---|
| | | notifies the success/failure of the operation (in this example: success). |

## 10.2 HTTP POST

| Command | Response | Description |
|---|---|---|
| AT+CMEE=2 | OK | Set the verbose error result codes. |
| AT+UDWNFILE="postdata.txt",11 | >hello world<br>OK | Write some data in the file to send. |
| AT+URDFILE="postdata.txt" | +URDFILE: postdata.txt,11<br>,"hello world"<br>OK | Optionally check whether the data is present. |
| AT+UHTTP=0 | OK | Reset the HTTP profile #0. |
| AT+UHTTP=0,1,"httpbin.org" | OK | Set up a connection to an echo server (httbin.org) that checks and echoes post commands. |
| AT+UHTTP=0,5,80 | OK | Set the port of the HTTP request to 80 |
| AT+UHTTPC=0,4,"/post","result.txt","postdata.txt",1 | OK<br>+UUHTTPCR: 0,4,1 | Submit a post command in text format and store the answer in result.txt. |
| AT+URDFILE="result.txt" | +URDFILE: result.txt,498,<br>"HTTP/1.1 200 OK<br>Content-Type:<br>application/json<br>Date: Tue, 15 Jan 2013<br>16:06:11 GMT<br>Server: gunicorn/0.16.1<br>Content-Length: 345<br>Connection: Close<br><br>{<br>"headers": {<br>"Content-Length": "11",<br>"Host": "httpbin.org",<br>"Content-Type":<br>"text/plain",<br>"User-Agent": "UBlox Leon<br>G200/1.0 (N7/HTTP 1.0)",<br>"Connection": "keep-<br>alive"<br>},<br>"args": {},<br>"data": "hello world",<br>"url":<br>"http://httpbin.org/post"<br>,<br>"files": {},<br>"json": null,<br>"form": {},<br>"origin": "10.82.21.198"<br>}"<br>OK" | Check the server's reply. |

## 10.3 Error handling

In case of errors returned in the last HTTP operation of a specified HTTP profile, it is possible to investigate the type of error using the +UHTTPER AT command.

| Command | Response | Description |
|---|---|---|
| AT+UHTTPC=0,4,"/post","result.txt","postdata.txt",1 | OK<br>+UUHTTPCR: 0,4,1 | Successfully submit a post command in text format and store the answer in result.txt. |
| AT+UHTTPER=1 | +UFTPER: 1,0,0<br>OK | In HTTP profile 1 the error code 0 is “No error”. |

## 10.4 Secure HTTP

Configure a secure manager profile before starting a secure HTTP. See section 3 for further details on this.

The following example describes how to configure the secure HTTP. Only the secure manager profile must be configured, the other HTTP commands will behave as in the case of unencrypted session.

| Command | Response | Description |
|---|---|---|
| AT+UHTTP=0,6,1 | OK | Enable secure HTTP. HTTPS (SSL encryption) enabled and the HTTP server port set to 443.<br>☞ The port number is set automatically to 443 (standard value for HTTPS). If the application is turning back to HTTP (using AT+UHTTP=0,6,0 command), the port number is changed automatically to 80. Differently, if the port number is changed manually (e.g., using AT+UHTTP=0,5,9000 command), a change in the security option (e.g., with +UHTTP=0,6,1) will not modify the port manually selected. |

# 11 LwM2M

See LwM2M application examples in the LwM2M objects and commands application note [6].

# Appendix

## A Glossary

| Abbreviation | Definition |
|---|---|
| AEAD | Authenticated Encryption with Associated Data |
| AEC | Automotive Electronics Council |
| AES | Advanced Encryption Standard |
| APN | Access Point Name |
| ARIA | a block cipher technique |
| ARM | Arm (Advanced RISC Machines) Holdings |
| ASCII | American Standard Code for Information Interchange |
| BBR | Battery Backed RAM |
| BER | Bit Error Rate |
| CA | certification authority |
| CBC | Block ciphers |
| CHACHA20 | A high-speed stream cipher |
| CID | Context identifier |
| CoAP | Constrained Application Protocol |
| CPU | Central Processing Unit |
| CSD | Circuit-Switched Data |
| CUT | Coordinated Universal Time |
| DC | Direct Current |
| DCE | Data Circuit-terminating Equipment* / Data Communication Equipment* |
| DDC | Display Data Channel |
| DER | Distinguished Encoding Rules |
| DH or DHE | Diffie–Hellman |
| DL | Down Link (Reception) |
| DNS | Domain Name System |
| DRX | Discontinuous Reception |
| DSA | Digital Signature Algorithm |
| DTE | Data Terminal Equipment |
| DTLS | Datagram Transport Layer Security |
| ECDH | Elliptic-Curve Diffie–Hellman |
| ECDHE | Elliptic-Curve Diffie–Hellman |
| ECDSA | Elliptic-Curve Digital Signature Algorithm |
| ePCO | Extended Protocol Configuration Options |
| EPS | Evolved Packet System |
| FOTA | Firmware updates Over-The-Air |
| HKDF | HMAC-based Key Derivation Functions |
| HMAC | Hash-Based Message Authentication |
| ICMP | Internet Control Message Protocol |
| IoT | Internet of Things |
| LPWA | Low-Power Wide-Area |
| LPWAN | Low-Power Wide-Area Network |
| M2M | Machine to Machine |

| Abbreviation | Definition |
|---|---|
| MAC | Message Authentication Code |
| MCU | MicroController Unit |
| MNO | Mobile Network Operator |
| MTU | Maximum transmission unit |
| NAT | Network Address Translation |
| NVM | Non-Volatile Memory |
| PEM | Privacy-Enhanced Mail |
| PS | Packet switched |
| PSD | Packet-Switched Data |
| PSK | Pre-Shared Key |
| PSM | Power Saving Mode |
| RA | Router Advertisement |
| RAI | Release Assistance Indication |
| RAT | Radio Access Technologies |
| RFC | Request for Comments |
| RoT | Root of Trust |
| RS | Router Solicitation |
| RSA | Rivest-Shamir-Adleman |
| RST | Reset (referred to a TCP reset packet) |
| SAO | Socket Always On |
| SHA | Secure Hash Algorithm |
| SLAAC | StateLess Address AutoConfiguration |
| SNI | Server name indication |
| SSL | Secure Sockets Layer |
| TLS | Transport Layer Security |
| TTL | Time To Live |
| URC | Unsolicited Result Code |
| WAN | Wide Area Network |

# Related documentation

[1] u-blox LARA-R6 series data sheet, UBX-21004391
[2] u-blox LARA-R6 series system integration manual, UBX-21010011
[3] u-blox LARA-R6 series AT commands manual, UBX-21046719
[4] u-blox EVK-R6 user guide, UBX-21035387
[5] u-blox LARA-R6 series application development guide, UBX-22001850
[6] u-blox LwM2M objects and commands application note, UBX-18068860
[7] u-blox Multiplexer implementation application note, UBX-13001887
[8] LARA-R6 FW update application note, UBX-22008011
[9] u-blox Positioning implementation application note, UBXDOC-686885345-1826
[10] RFC 768 - User Datagram Protocol (UDP)
[11] RFC 7252 - Constrained Application Protocol (CoAP)
[12] RFC 6347 - Datagram Transport Layer Security Version 1.2
[13] RFC 7959 - Block-Wise Transfers in the Constrained Application Protocol (CoAP)
[14] RFC 8323 - Constrained Application Protocol (CoAP) over TCP, TLS, and WebSockets
[15] RFC 5077 - Transport Layer Security (TLS) Session Resumption without Server-Side State
[16] u-blox test server configuration, UBX-14005690
[17] RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2
[18] RFC 8446 - The Transport Layer Security (TLS) Protocol Version 1.3

☞ For product change notifications and regular updates of u-blox documentation, register on our website, www.u-blox.com.

# Revision history

| Revision | Date | Name | Comments |
|---|---|---|---|
| R01 | 05-May-2022 | mreb | Initial release |
| R02 | 23-Feb-2024 | mreb | Extended the document for LARA-R6 " 01B" product version.<br>Added the troubleshooting secure connection section and other minor changes. |

# Contact

**u-blox AG**

Address: Zürcherstrasse 68
8800 Thalwil
Switzerland

For further support and contact information, visit us at www.u-blox.com/support.