



LEXI-R10 series

Internet applications development guide

Application note



Abstract

This document provides detailed examples of how to use AT commands to develop IP applications in LEXI-R10 series modules.

Document information

Title	LEXI-R10 series	
Subtitle	Internet applications development guide	
Document type	Application note	
Document number	UBXDOC-686885345-2004	
Revision and date	R01	14-Jun-2024
Disclosure sensitivity	C1-Public	

This document applies to the following products:

Product name
LEXI-R10 series

u-blox or third parties may hold intellectual property rights in the products, names, logos, and designs included in this document. Copying, reproduction, or modification of this document or any part thereof is only permitted with the express written permission of u-blox. Disclosure to third parties is permitted for clearly public documents only.

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	5
2 PS data connection	6
2.1 PDP contexts.....	6
2.2 MTU configuration.....	7
2.2.1 MTU in IPv4	7
2.2.2 MTU in IPv6	7
2.3 Socket and PDP context activation.....	7
2.3.1 Default PDP and preferred protocol type configuration	7
2.4 Other relevant AT commands	8
2.4.1 UPING	8
2.4.2 DNS resolution	9
3 Data security	10
3.1 Certificates manager +USECMNG.....	10
3.2 Profile configuration +USECPRF	10
3.2.1 Cipher suites	11
3.3 Complete example	12
3.4 (D)TLS session resumption	12
3.4.1 Session resumption with session ticket.....	13
3.4.2 (D)TLS session resumption examples	13
3.5 Troubleshooting secure connection.....	16
4 Dial-up network (PPP)	17
4.1 Basic setup	17
4.1.1 Dial-up configuration	18
4.1.2 PPP and IPv6	18
4.1.3 PPP over multiple PDP contexts	18
4.2 Terminate cellular packet data connection	19
5 TCP/UDP internal stack.....	20
5.1 Socket connect.....	20
5.2 Socket listening.....	20
5.3 Socket write (+USOWR)	21
5.3.1 Binary mode	21
5.3.2 Base syntax.....	21
5.3.3 Queue FULL	22
5.4 Socket read (+USORD).....	22
5.5 Socket operations with "Keep Alive" option.....	24
5.6 Socket write (+USOST).....	24
5.7 Socket read (+USORF)	25
5.8 Socket state.....	25

5.9	Socket close	26
5.10	Socket always-on	26
5.10.1	Sockets creation	27
5.10.2	Deep-sleep use case	27
5.11	Secure socket	28
5.12	Testing sockets	28
6	MQTT	30
6.1	Basic setup	30
6.1.1	Default and minimal configuration	30
6.1.2	Last will configuration	30
6.1.3	Profile management	30
6.1.4	Deep-sleep handling	31
6.1.5	Internal PDP context mapping	31
6.2	Start and end a MQTT session	32
6.3	Subscribe to a topic and publish a message to the same topic	32
6.4	Publish a message with hexadecimal mode set	32
6.5	Publish a binary message to a topic	34
6.6	Ping the MQTT broker	34
6.7	Last will packet	34
6.8	Debug	35
6.9	Secure MQTT	35
7	HTTP	36
7.1	Basic setup	36
7.1.1	Profile management	37
7.1.2	Deep-sleep handling	38
7.2	HTTP POST using file system	38
7.3	HTTP POST using Direct link	39
7.4	Error handling	41
7.5	Secure HTTP	41
	Appendix	42
A	Glossary	42
	Related documentation	44
	Revision history	44
	Contact	44

1 Introduction

This document provides guidance for developing applications based on the internet protocol (IP) that use LEXI-R10 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, HTTP, and 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 for managing and configuring 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 [6], 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 scope	Document name	Notes
Product evaluation	EVK-R10 user guide [4]	Starting guide for the LEXI-R10 evaluation kit.
Product essentials	Data sheet [2]	Performance and characteristics of a product (family)
	System integration manual [3]	Describes how to design a product (family) into a customer application
	AT commands manual [1]	Reference guide for protocols, detailed AT command descriptions. Refer to this manual for details of any AT command listed in this app note.
Notifications	Sample Delivery Note / Information Note / Product Change Note / End of Life	Notifications of SW / HW / Certification changes.
Application design	Application development guide [6]	The first document to be used to develop on LEXI-R10.
	FW update app note [5]	FW update procedures (FOAT, FOTA, and EasyFlash).
	Internet application app note	This document.
	Production and validation test app note [11]	Guidelines of OEM production test and validation test. Contact tech support for this document.
	Mux implementation [9]	Use of multiplexer with cellular modules.
Tools	m-center AT scripts collection	https://github.com/u-blox/m-center

Table 1: LEXI-R10 product documentation overview

From section 3.5 on, the document provides examples of internet-related applications built with the LEXI-R10 series modules.

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 [6].

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.

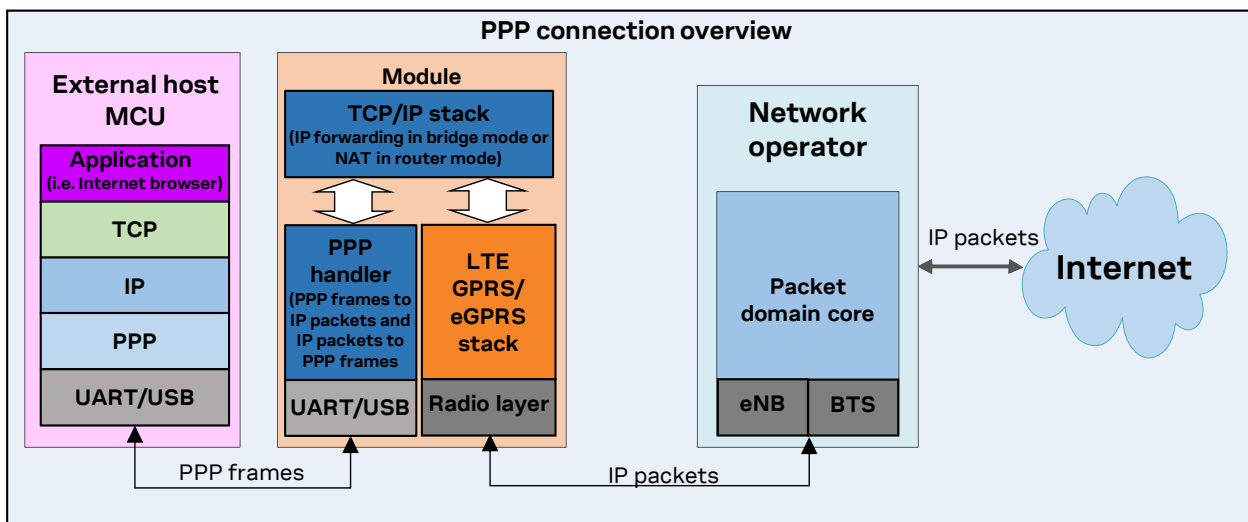


Figure 1: Example of external context structure

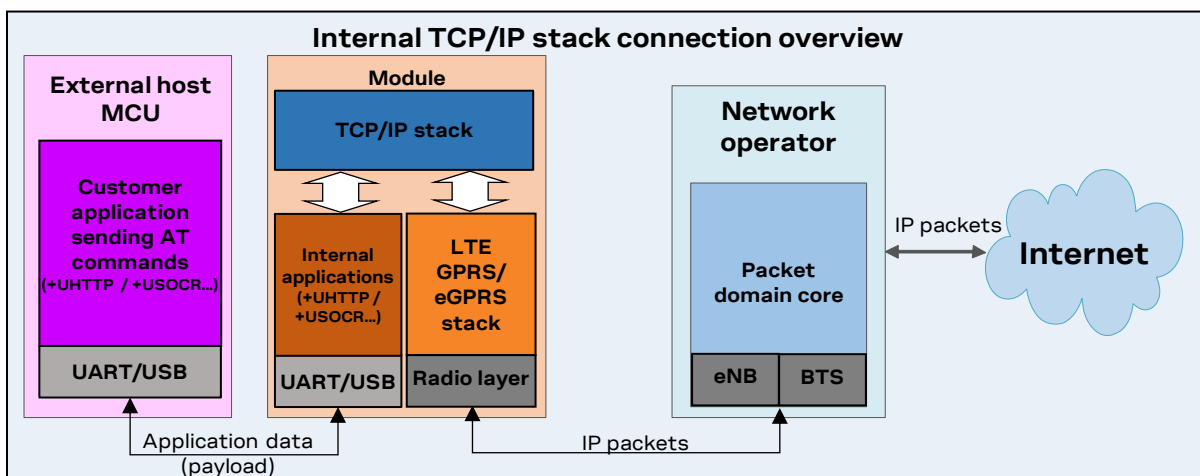


Figure 2: Example of internal context structure

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.
-  It is strongly recommended to use a proper APN for the initial default bearer. The APN value shall be provided by the SIM card provider.

For further details on the APN configuration, see the “Network registration” section of the application development guide [\[6\]](#).

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 PDP contexts that can be activated at the same time is 3 while the product allow to configure up to 15 different context.

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), the MTU size is assigned by the LTE network during the PDN connection establishment. 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.

Network settings verification

Command	Response	Description
AT+CFUN=0	OK	Turn off radio functionality. ➔ 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 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

Command	Response	Description
AT+CFUN=1	OK	Turn on radio functionality.
AT+COPS?	+COPS: 0,0,"I TIM",7 OK	Verify if module is currently registered to the network. ➔ 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 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.

➔ After PDN activation the AT+CGDCONT and AT+CGDCONTRDP read commands return 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.

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

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" OK	DNS resolution request.
		 Only the asynchronous mode is supported, thus a final result code (OK or an error result code) is returned immediately unlocking the AT interface and making it available for the execution of other AT commands. Once the result of DNS resolution becomes available, it is notified to the AT interface through the +UUDNSRN URC.

-  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.

3 Data security

Every internet client or socket can be configured to use a secure profile to perform a (D)TLS secure connection.

Use the +USECPRF AT command to configure the secure profiles. Use the +USECMNG AT command to store, delete, and handle the TLS certificates and keys that would be used by the secure profiles.

3.1 Certificates manager +USECMNG

The +USECMNG AT command enables managing 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 [\[1\]](#).

 The (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 1024 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_certificate.crt"	+USECMNG: 1,0,"ca_cert","d10137cee624fcee624418db5eaa" 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_certificate","b137ce137ce5edd6723d8b13" 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" 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" CC,"client_cert","A client certificate","2032/06/22 12:34:48" PK,"client_key" 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 TLS/DTLS connection.

The command manages security profiles for the configuration of the following TLS/DTLS connections properties:

- Certificate validation level
- Minimum (D)TLS version
- Cipher suites to be proposed: legacy, IANA nomenclature
- 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
- Application-Layer Protocol Negotiation (ALPN)
- (D)TLS session resumption.

Command	Response	Description
AT+USECPRF=0	OK	Reset (set to factory-programmed value) all the parameters of security profile #0.  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.  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 (D)TLS handshake to negotiate the security setting for the secure connection. The cipher suite for the (D)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.

3.3 Complete example

Command	Response	Description
Step 1: 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.  Unlike the example in section 3.1, here the certificate is transferred as a byte stream and is not stored in the LEXI-R10 file system.
-----BEGIN CERTIFICATE----- MIIEDCCAwigAwIBAgIQNE7VvyDV7ex J9C/OjVaMaA== -----END CERTIFICATE-----	+USECMNG: 1,0,"ThawteCA", "8ccadc0b22cef5be72ac411a 11a8d812" OK	Input PEM formatted trusted root certificate data bytes. Output MD5hash string of the stored trusted root certificate DER.
Step 2: List all available certificates and private key		
AT+USECMNG=3	CA, "ThawteCA", "thawte Primary Root CA", "2036/07/17" 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 TLS for the UHTTP profile #0 and specify the 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 TLS connection, the number of concurrent TLS connections is limited. The +USECMNG AT command and the underlying TLS infrastructure allows two overall concurrent TLS connections (i.e., 2 HTTPS requests or 1 HTTPS and 1 MQTTS 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 [12][13] of the session resumption provides the following concept:

- **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.

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 stored in the corresponding security profile, which is volatile. 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 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).

3.4.2 (D)TLS session resumption examples

The following session will present two examples of the use of the session resumption. In the first example the session data is not encrypted, while in the second example the session resumption data is encrypted with a local encryption feature provided by the RoT.

3.4.2.1 Session resumption with session ticket not encrypted

Step 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>,13,1,1	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> OK	Get the session data configured during the handshake.  The session resumption data should be stored to be reused for the resumption on the next session

Step 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		

Command	Response	Description
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.
	+USECPRF: <profile_id>,13,0,2	URC with session resumption status.
	+UUHTTPCR: <app_profile>,1,1	HTTP GET URC response.

3.4.2.2 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.
	+USECPRF: <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> OK	Get the session data configured during the handshake.  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		
		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		
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"		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

This section reports a list of recommendations to correctly configure the secure TLS connection between cellular modules and server. We recommend application designer to review this list, if the application cannot complete a secure connection.

- 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 TLS CA certificate based on server TLS certificate chain by using the +USECMNG and +USECPRF AT commands.
- Check the 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 (also in this case by using 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 using the +UNETCFG AT command. 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 the 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.
-  When the deep-sleep functionality is enabled (with the +USPV AT command), the module can enter only in the deep-sleep Sleep-1 if PPP is active.

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

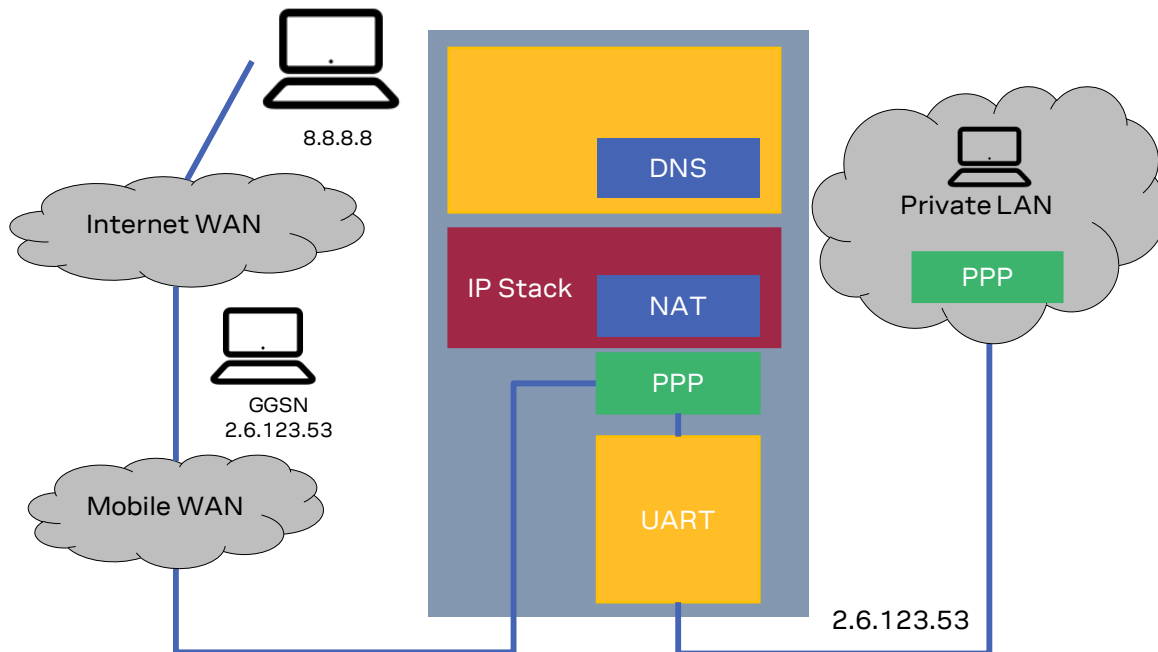


Figure 3: Example of a dial-up configuration

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. Only one PPP instances can be simultaneously active in the LEXI-R10 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 [\[1\]](#).

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 [\[1\]](#) 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.

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.

5.1 Socket connect

Command	Response	Description
AT+USOCR=6	+USOCR: 0 OK	TCP socket creation. In this example socket #0 is created. 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.  Created socket is by default mapped to the context ID 1.  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 OK	Create another socket (in this case the socket is UDP, and its identifier is 1).  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" 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. 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 +UUSOCL: 0	If the connection is not successfully performed, an error result code is returned and the socket used for the connection attempt <u>is closed</u>. The notification is provided by +UUSOCL URC.

5.2 Socket listening

Command	Response	Description
AT+USOCR=6	+USOCR: 0 OK	TCP socket creation with ID #0.
AT+USOLI=0,1099	OK	Set the socket in listening mode on port 1099.  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.

Command	Response	Description
	+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: • 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 • 151.9.34.66: IP of the remote server • 39912: service port • 0: listening socket. It is the socket identifier specified with the +USOLI AT command • 151.9.34.74: module IP address • 1099: listening port assigned to the connection. Configured with the +USOLI AT command Socket #1 is now ready for reading/writing data.
	+UUSORD: 1, 18	18 bytes of incoming data over the previously established connection.  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). After the @ prompt reception, wait for a minimum of 50ms before sending data.
12	+USOWR: 0, 2 OK	Write data bytes. It is not allowed to write fewer bytes than previously specified with +USOWR AT command. If more bytes are written with respect to the threshold, the remaining bytes will be truncated. The interface is blocked until all bytes are written. 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 OK	Write 2 data bytes data on socket #0. 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.  Some characters are not allowed in base syntax mode. For the allowed characters, see the AT commands manual [1].

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. In this case an error result code is returned.  The socket queue size is set to 7 kB in the writing upload and 6 kB in reading download.  The secure socket queue size is limited to 1kB in the writing upload.
AT+USOCTL=0,10	+USOCTL: 0,10,4 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 OK	It is also possible to query for TCP outgoing unacknowledged data of the socket (this command is valid only for TCP socket). In this case, 0 bytes of data is unacknowledged.

5.4 Socket read (+USORD)

Scenario 1

Command	Response	Description
	+UUSORD: 0,2	The remote server sends 2 data bytes on socket #0. 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" OK	Read data. The data is returned between quotation marks.

Scenario 2

Command	Response	Description
	+UUSORD: 0,30	The remote server sends 30 data bytes on socket #0. 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" OK	Read only part of data (in this example 10 bytes of data are read). 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.

Scenario 3

Command	Response	Description
	+UUSORD: 0,30	The remote server sends 30 data bytes on socket #0. If a socket buffer is empty +UUSORD URC indicates a TCP packet has been received from the

Command	Response	Description
		remote host the socket is connected to and the amount of data bytes of the packet.
AT+USORD=0,10	+USORD: 0,10,"hfgyrhgfty" OK	Only part of the data bytes (10 bytes in this example) is read. 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" OK	Only part of the data bytes (10 bytes in this example) is read. 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" OK	All the bytes are read.
AT+USORD=0,0	+USORD: 0,0 OK	Verifies how much unread data is in the buffer. In this example 0 bytes are in socket #0.

Scenario 4

Command	Response	Description
	+UUSORD: 0,30	The remote host sends 30 bytes of data on the socket #0. 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" OK	Only part of the data bytes (10 bytes in this example) is read. 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. After the @ prompt reception, wait for a minimum of 50 ms before sending data.
123	+USOWR: 0,0 OK	Write data. After the last byte the data prompt is closed.  It is not allowed to write fewer bytes than previously specified with +USOWR AT command.  If more bytes are written with respect to the threshold, the remaining bytes will be truncated.

Command	Response	Description
		The interface is blocked until all bytes are written. 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" 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.

5.5 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 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. • 0: socket number to be set to enable keep alive option • 65535: specifies socket level option • 8: specify the "Keep Alive" option • 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. • 0: socket number to be set to enable keep alive option • 6: specifies TCP level option • 2: specifies option TCP "keep idle" timer option • 30000: the module will send dummy TCP packets every 30000 ms

5.6 Socket write (+USOST)

Command	Response	Description
AT+USOCR=17	+USOCR: 0 OK	UDP socket creation. In this example the socket #0 is created. 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 OK	The local port to be used for data sending can be configured during the UDP socket creation. 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" OK	DNS resolution of the URL "ftp.u-blox.com".
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).

Command	Response	Description
12	+USOST: 0,2 OK	Write data. After the last data byte is written, the prompt is closed.  It is not allowed to write fewer bytes than previously specified with +USOST AT command.  If more bytes are written with respect to the threshold, the remaining bytes will be truncated. The interface is blocked until all bytes are written. 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" OK	Read data. The information text response indicates: • Read socket identifier • Remote IP address • Remote UDP port • Number of read data bytes • 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" 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" OK	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.
	+UUSORD: 0,20	 After the first URC has been returned, a second URC is returned (only after a reading operation) indicating: • If a reading operation of a packet is not finished it will provide the remaining data of the specific packet • 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 [\[1\]](#).

Command	Response	Description
AT+USOCTL=0,0	+USOCTL: 0,0,6 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 OK	It is possible to query the state of TCP connection associated with the socket; in this example the

Command	Response	Description
		socket #0 (this command is valid only for TCP socket). 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 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 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 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 the +USOER AT command to retrieves the last error occurred in the last socket operation.

Command	Response	Description
AT+USOER	+USOER: 104 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.  The URC is received only when the socket is closed and the buffer is empty.

By the module

Command	Response	Description
AT+USOCL=0	OK	The socket is closed by the module (socket #0).  No +UUSOCL URC is returned.  After sending this command, the socket buffer is cleared.

5.10 Socket always-on

Always-on (AoN) sockets retain their state when the module enters Sleep-2 and Hibernate deep-sleep states. For more details on the module's low power features, see the AT commands manual [1] and the application development app note [6].

Only one TCP and one UDP socket can be configured as AoN. The first socket of the specific transport protocol will be automatically created as AoN, additional sockets will not be AoN, but if a AoN socket is closed, the next socket of the same transport protocol will be created as AoN.

-  An AoN socket set as secure loses its AoN functionality.
-  A TCP socket set in listening mode loses its AoN functionality.
-  The use of direct link mode (e.g., with the AT+USODL command) prevents entering Sleep-2 and Hibernate deep-sleep states even if the socket is configured as AoN.

-  If an AoN active socket has data waiting to be written or read (the TX and RX buffer are not empty), the module can enter only Sleep-1 state.
-  Any created socket not configured as AoN prevents the module from entering Sleep-2 and Hibernate deep-sleep states.
-  Data sent by a remote host triggers the establishment of a radio connection and the module's exit from the deep-sleep states. The module's wake up delay is determined by the network and depends on the setting of DRX or eDRX (if used). If this delay is very long (i.e. in the eDRX case), the remote host may have already closed the connection. For more details, refer to the AT command manual [1].

Refer to the AT+USOCR? read command to list the open sockets and their AoN state.

5.10.1 Sockets creation

Creation of 4 sockets, 2 TCP and 2 UDP: only the first opened per type is always-on.

Command	Response	Description
AT+USOCR=6,,,1	+USOCR: 0,6,1 OK	TCP socket creation. In this example socket #0 is created.  Using the option <report_AoN> set to 1, the command response reports the type and the AoN state of the socket just created.
AT+USOCR=6,,,1	+USOCR: 1,6,0 OK	Create another TCP socket.  The socket is not configured as AoN because another socket (with identifier #0) of the same type (TCP) has already been set as AoN.
AT+USOCR=17	+USOCR: 2 OK	Create another socket (in this case the socket is UDP, and its identifier is #2).  The socket is set as AoN but the response does not contains all the details.
AT+USOCR=17,,,1	+USOCR: 3,17,0 OK	Create a second UDP socket (in this case the socket is UDP, and its identifier is #3).  The socket is not configured as AoN because another socket of the same type (UDP) has already been set as AoN.
AT+USOCR?	+USOCR: 0,6,1 +USOCR: 1,6,0 +USOCR: 2,17,1 +USOCR: 3,17,0 OK	The AT+USOCR? read command reports the type and the AoN state of all open sockets.

5.10.2 Deep-sleep use case

Write scenario

Command	Response	Description
AT+USOCR=6,,,1	+USOCR: 0,6,1 OK	TCP socket creation. In this example socket #0 is created and configured as AoN.
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. The connection is now uniquely associated to the socket. The socket is now ready for read/write operations.

	+UUSLPURC: SLP2 1	If there is no pending RX or TX data and the modem is in idle mode with large DRX periods, the module can enter the deep-sleep state Sleep-2.  This URC is received only if it has been enabled by AT+USLPURC="SLEEP2",1 AT command.
	+UUSLPURC: SLP2 0	The module exits the deep-sleep state Sleep-2.
AT+USOWR=0,2,"12"	+USOWR: 0,2 OK	Write two bytes on socket #0 in binary syntax.

Read scenario

Command	Response	Description
AT+USOCR=6,,,1	+USOCR: 0,6,1 OK	TCP socket creation. In this example socket #0 is created and configured as AoN.
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.
	+UUSLPURC: HIB 1	If there is no pending RX or TX data and the modem is in idle mode with large eDRX periods, the module can enter the deep-sleep state Hibernate.  This URC is received only if it has been enabled by AT+USLPURC="HIBNATE",1 AT command.
		The remote server sends 30 bytes on socket #0.
	+UUSLPURC: HIB 0	The module exits the deep-sleep state Sleep-2.
	+UUSORD: 0,30	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.

5.11 Secure socket

Use the +USOSEC AT command to enable or disable the use of 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 SSL socket configuration with the +USOSEC AT command can be performed only after the socket has been created (+USOCR AT command).
-  Even if the maximum number of sockets that can be opened simultaneously is 6 (maximum 2 secure sockets), 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.
-  Secure sockets cannot be configured as AoN sockets, therefore their usage prevents the module from entering Sleep-2 and Hibernate deep-sleep states.

5.12 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.

Command	Response	Description
AT+USOCR=17	+USOCR: 0 OK	Create a UDP socket.
AT+UDNSRN=0,"echo.u-blox.com"	+UDNSRN: "195.34.89.241" OK	DNS resolution of the URL.
AT+USOST=0,"195.34.89.241",7,5, "Hello"	+USOST: 0,5 OK +UUSORD: 0,5	Write 5 characters to server.
AT+USORF=0,5	+USORF: 0,"195.34.89.241" ,7,5,"Hello" OK	Read 5 echoed characters.

For additional details and examples on the use of the u-blox echo server, see the dedicated application note [\[14\]](#).

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.
-  If the power saving is enabled (see +UPSV command), it is important to prevent entering into deep-sleep while operating with the MQTT application. See an example in section 6.1.4.

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" +UMQTT: 2, "", 1883 +UMQTT: 3, "", 1883 +UMQTT: 4, "" +UMQTT: 6, 0 +UMQTT: 7, 0 +UMQTT: 8, "" +UMQTT: 9, 0, "" +UMQTT: 10, 0 +UMQTT: 11, 0 OK	Read the current profile configuration. All the reported values can be modified; see the AT commands manual [1] for a detailed description.  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" +UMQTT: 2, "", 1883 +UMQTT: 3, "", 1883 +UMQTT: 4, "" +UMQTT: 6, 0	Read the current profile configuration.

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

6.1.4 Deep-sleep handling

Use the +USLPVOTE AT command to prevent entering into any level of deep-sleep while configuring the MQTT profile or while an MQTT session is active. Store the profile (see the section 6.1.3) to avoid losing the MQTT configuration between deep-sleep cycles. Reload the profile configuration from NVM before starting an MQTT session.

Command	Response	Description
AT+USLPVOTE=1,2	OK	Configure the +USLPVOTE to block the deep-sleep Sleep-1 state (and also lower deep-sleep level). Command needed only for the first configuration.
AT+USLPVOTE=0,0	OK	Configure the vote down option so to start the MQTT configuration.
MQTT configuration: Configure the MQTT profile as described in section 6.1.1		
AT+UMQTTNV=2	OK	Store the MQTT profile into NVM. See session 6.1.3.
AT+USLPVOTE=0,1		Vote-up, the module is allowed to enter in the deep-sleep state.
Deep-sleep cycle		
AT+USLPVOTE=0,0	OK	Configure the vote down option so to start the MQTT client use.
AT+UMQTTNV=1	OK	Reload the MQTT profile from the NVM.
Start a session with a MQTT server, see 6.2.		
AT+USLPVOTE=0,1	OK	Vote-up, the module is allowed to enter in the deep-sleep state after the MQTT session ends.

6.1.5 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).

 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+UMQTT=1	OK	Connect to the broker.
	+UUMQTT: 1,1	The MQTT session request is successfully performed. The MQTT session can start. The +UUMQTT URC provides the result of the requested action from the MQTT broker
AT+UMQTT=0	OK	Disconnect from the broker, end of the MQTT session.
	+UUMQTT: 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+UMQTT=4,0,"module/lights"	OK	Subscribe to a topic.
	+UUMQTT: 4,1,0,"module/light"	The broker granted QoS level is 0.
AT+UMQTT=2,0,0,0,"module/light s","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.
	+UUMQTT: 6,1	Notification of the received publish message.
AT+UMQTT=6,1	+UMQTT: 6,0,27,13,"module/lights",14,"light_1 is red" OK	Read the received publish message.
AT+UMQTT=5,"module/lights"	OK	Unsubscribe from the previously subscribed topic.
	+UUMQTT: 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+UMQTT=4,0,"module/ascii"	OK	Subscribe to the "module/ascii" topic.
	+UUMQTT: 4,1,0,"module/ascii"	

Command	Response	Description
AT+UMQTTTC=2,0,0,0,"module/ascii", "ABCD3031"	OK +UUMQTTTC: 2,1 +UUMQTTTC: 6,1	Send a Publish message, the "ABCD3031" payload is encoded with ASCII characters (the 4 th parameter value (<hex_mode>) is 0). Notification of the received publish message.
AT+UMQTTTC=6,1 or AT+UMQTTTC=6,1,0	+UMQTTTC: 6,0,20,12,"module/ascii", 8,"ABCD3031" OK	Read the received publish message in ASCII mode, so the received string is same as the one sent: 8 characters. The payload bytes in the MQTT packet are: 41 42 43 44 33 30 33 31
AT+UMQTTTC=2,0,0,0,"module/ascii", "ABCD3031"	OK +UUMQTTTC: 2,1 +UUMQTTTC: 6,1	Send again the previous Publish message Notification of the received publish message.
AT+UMQTTTC=6,1,1	+UMQTTTC: 6,0,20,12,"module/ascii", 8," 4142434433303331" OK	Read the received publish message in hex mode, the payload is displayed as a string of hexadecimal, the received string is same as the one sent: 8 bytes. The payload bytes in the MQTT packet are: 41 42 43 44 33 30 33 31
HEX mode		
AT+UMQTTTC=4,0,"module/hex"	OK +UUMQTTTC: 4,1,0,"module/hex"	Subscribe to the "module/hex" topic.
AT+UMQTTTC=2,0,0,1,"module/hex", "ABCD3031"	OK +UUMQTTTC: 2,1 +UUMQTTTC: 6,1	Send a Publish message with the same payload encoded as hexadecimal (the 4 th parameter value (<hex_mode>) is 1). Notification of the received publish message.
AT+UMQTTTC=6,1 or AT+UMQTTTC=6,1,0	+UMQTTTC: 6,0,14,10,"module/hex", 4,"«í01" OK	Read the received publish message, the payload length is 4 because each pair of characters is considered as one byte. The payload bytes in the MQTT packet are: AB CD 30 31 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. The other 2 bytes "30" and "31" are respectively the standard ASCII characters "0" and "1".
AT+UMQTTTC=2,0,0,1,"module/hex", "ABCD3031"	OK +UUMQTTTC: 2,1 +UUMQTTTC: 6,1	Send again the previous Publish message Notification of the received publish message.
AT+UMQTTTC=6,1,1	+UMQTTTC: 6,0,14,10,"module/hex", 4," ABCD3031" OK	Read the received publish message in hex mode, the payload is same as the one sent: 4 bytes.

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+UMQTTTC=9 command should be used.

Command	Response	Description
AT+UMQTTTC=4,0,"module/special"	OK +UUMQTTTC: 4,1,0,"u-blox/special"	Subscribe to the "module/special" topic.
AT+UMQTTTC=9,1,0,"module/special",21	>	Send a Publish message with special characters in the payload.
"this is an example"<CR>	OK +UUMQTTTC: 2,1 +UUMQTTTC: 6,1	Notification of the received publish message.
AT+UMQTTTC=6,1	+UMQTTTC: 6,0,35, 14,"module/special",21,""this is an example" " 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+UMQTTTC=1	OK +UUMQTTTC: 1,1	Connect to the broker and start a MQTT session.
AT+UMQTTTC=8,1	OK +UUMQTTTC: 8,0	Start a "ping loop". 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. The ping request is approximately triggered after 80% of the keep alive time. 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+UMQTTTC=1	OK +UUMQTTTC: 1,1	Connect to the broker and start a MQTT session.
Module #2		
AT+UMQTTTC=1	OK +UUMQTTTC: 1,1	Connect to the same broker and start a MQTT session.

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

6.8 Debug

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

Command	Response	Description
AT+UMQTTT=1	OK +UUMQTTT: 1,0	Unsuccessful session start.
AT+UMQTTT	+UMQTTT: 13,50 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+UMQTTT=1	OK +UUMQTTT: 1,1	Connect to the broker and start a secure MQTT session.

7 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.

7.1 Basic setup

This section shows an example about usage of the u-blox proprietary +UHTTP and +UHTTTPC 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 [1].

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"	OK	Set the server domain name and port.
AT+UHTTP=0,5,80	OK	 HTTP server name (e.g., "httpbin.org"). The factory-programmed value is an empty text string.
AT+UHTTP=0,10,0	OK	Set HTTP input/output mode option to use FS.
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).  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" OK	DNS resolution of httpbin.org.
AT+UHTTTPC=0,0,"/", "head.ffs"	OK +UUHTTTPCR: 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 +UUHTTTPCR URC notifies the success/failure of the operation (in this example: success).
AT+UHTTTPC=0,1,"/", "get.ffs"	OK +UUHTTTPCR: 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 +UUHTTTPCR URC notifies the success/failure of the operation (in this example: success).
AT+UHTTTPC=0,5,"/post", "post.ffs", "name_post=MyName&age_post=30", 0	OK +UUHTTTPCR: 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 +UUHTTTPCR notifies the success/failure of the operation (in this example: success).
AT+UHTTP=0,3,"P455w0rd"	OK	HTTP server password.
AT+UHTTP=0,4,1	OK	HTTP server authentication method (basic authentication).
AT+UHTTTPC=0,1,"/basic-auth/test_user/P455w0rd", "get_auth.ffs"	OK	GET request returning information on authenticated user. The page requires basic authentication. The result is saved in "get_auth.ffs" file on the local file system of the module. The +UUHTTTPCR URC

Command	Response	Description
		notifies the success/failure of the operation (in this example: success).
	+UUHTTPCR: 0,1,1	
AT+UHTTP=0,10,1	OK	Set HTTP output mode option to use Direct link mode.
AT+UHTTPC=0,0,"/","head.ffs"	CONNECT HTTP/1.1 200 OK Date: Tue, 07 May 2024 08:42:40 GMT Content-Type: text/html; charset=utf-8 Content-Length: 9593 Connection: close Server: unicorn/19.9.0 Access-Control-Allow-Origin: * Access-Control-Allow-Credentials: true DISCONNECT OK +UUHTTPCR: 0,0,1	HEAD request of the default page and print out the result into the terminal. File "head.ffs" parameter is ignored. The +UUHTTPCR URC notifies the success/failure of the operation (in this example: success).
AT+UHTTPC=0,1,"/","get.ffs"	CONNECT HTTP/1.1 200 OK Date: Tue, 07 May 2024 08:47:45 GMT Content-Type: text/html; charset=utf-8 Content-Length: 9593 Connection: close Server: unicorn/19.9.0 Access-Control-Allow-Origin: * Access-Control-Allow-Credentials: true <html_data_9593> DISCONNECT OK +UUHTTPCR: 0,1,1	GET request of the default page and print out the result into terminal. The file "get.ffs" parameter is ignored in direct link mode. "html_data_9593" is used to summarize file received from server that contains html data. The +UUHTTPCR URC notifies the success/failure of the operation (in this example: success).

7.1.1 Profile management

Command	Response	Description
AT+UHTTPNV=0,0	OK	Restore HTTP client profile parameters to the factory-programmed setting. This operation reset usage of NVM parameter after deep-sleep wake up. This means that configuration will be set to default after sleep.
AT+UHTTPNV=0,2	OK	Store the current HTTP client profile parameters to the NVM. This configuration will be used after sleep wake up when HTTP commands are issued.
AT+UHTTPNV=0,1	OK	Set HTTP client profile parameters to values previously stored in the NVM. Enable usage of NVM configuration after sleep wake-up.
AT+UHTTPNV?	+UHTTPNV: 0,1 OK	Return 1 if profile status is loaded/stored into NVM. This means that NVM configuration will be used

Command	Response	Description
		after deep-sleep wake up to perform +UHTTPC operation. Setting is not permanent, and should be activated using +UHTTPNV=0,1 or +UHTTPNV=0,2 command after boot.

7.1.2 Deep-sleep handling

Use the +USLPVOTE AT command to prevent entering into deep-sleep (any level) while configuring the HTTP profile. Store the profile (see section 7.1.1) to avoid losing the HTTP configuration between deep-sleep cycles. Changes in the HTTP configuration that are not stored into NVM will be lost during a deep-sleep cycle.

7.1.2.1 First boot procedure

Command	Response	Description
AT+USLPVOTE=1,2	OK	Configure the +USLPVOTE to block the deep-sleep Sleep-1 state (and also lower deep-sleep level). Command needed only for the first configuration.
AT+USLPVOTE=0,0	OK	Configure the vote down option so to start the HTTP configuration.
HTTP configuration: Configure parameter as in section 7.1.		
AT+UHTTPNV=0,2	OK	Store the current HTTP client profile parameters into the NVM. This configuration will be used after the deep-sleep wake up when HTTP client is required.
AT+USLPVOTE=0,1		Vote-up, the module is allowed to enter in the deep-sleep state.
Deep-sleep cycle		
AT+UHTTPC=0,1,"/path/file.html" , "responseFilename"	OK	Perform HTTP operation after sleep wake-up using stored NVM configuration.  Configuration changes not stored in NVM using command +UHTTPNV=0,2 are not available after sleep wake up.

7.1.2.2 Procedure after boot

Command	Response	Description
AT+UHTTPNV=0,1	OK	After boot is required to load the HTTP client profile parameters from the NVM. This configuration will be used after deep-sleep wake up when +UHTTPC command are issued. Changes in HTTP configuration can be done as explained into 7.1.2.1 preventing the sleep and store parameters into NVM.
Deep-sleep cycle		
AT+UHTTPC=0,1,"/path/file.html" , "responseFilename"		Perform HTTP operation after sleep wake-up using stored NVM configuration.

7.2 HTTP POST using file system

Command	Response	Description
AT+CMEE=2	OK	Set the verbose error result codes.

Command	Response	Description
AT+FOPEN="postdata.txt"	+FOPEN:1 OK	Open file and get file handler number
AT+FWRITE=1,11	CONNECT hello world +FWRITE: 11, 11 OK	Write some data in the file to send.
AT+FSEEK=1,0	OK	Set file to initial position
AT+FREAD=1	CONNECT 11 hello world OK	Optionally check whether the data is present.
AT+FCLOSE=1	OK	Close file handler
AT+UHTTP=0	OK	Reset the HTTP profile #0.
AT+UHTTP=0,1,"httpbin.org"	OK	Set up a connection to an echo server (httpbin.org) that checks and echoes post commands.
AT+UHTTP=0,5,80	OK	Set the port of the HTTP request to 80
AT+UHTTP=0,10,0	OK	Set to use filesystem as input/output mode
AT+UHTTPC=0,4,"/post","result.txt","postdata.txt",1	OK +UUHTTPCR: 0,4,1	Submit a post command in text format and store the answer in result.txt.
AT+FOPEN="result.txt"	+FOPEN:1 OK	Open file and get file handler number
AT+FREAD=1	CONNECT 498 HTTP/1.1 200 OK Content-Type: application/json Date: Tue, 15 Jan 2013 16:06:11 GMT Server: gunicorn/0.16.1 Content-Length: 345 Connection: Close { "headers": { "Content-Length": "11", "Host": "httpbin.org", "Content-Type": "text/plain", "User-Agent": "UBlox Leon G200/1.0 (N7/HTTP 1.0)", "Connection": "keep- alive" }, "args": {}, "data": "hello world", "url": "http://httpbin.org/post" , "files": {}, "json": null, "form": {}, "origin": "10.82.21.198" }	Check the server's reply.
AT+FCLOSE=1	OK	Close file handler

7.3 HTTP POST using Direct link

Command	Response	Description
AT+CMEE=2	OK	Set the verbose error result codes.
AT+UHTTP=0	OK	Reset the HTTP profile #0.

Command	Response	Description
AT+UHTTP=0,1,"httpbin.org"	OK	Set up a connection to an echo server (httpbin.org) that checks and echoes post commands.
AT+UHTTP=0,5,80	OK	Set the port of the HTTP request to 80
AT+UHTTP=0,10,1	OK	Set direct link as input/output mode
AT+UHTTPCR=0,7,"/post",1,,11	CONNECT hello world HTTP/1.1 200 OK Date: Tue, 07 May 2024 09:26:36 GMT Content-Type: application/json Content-Length: 383 Connection: close Server: gunicorn/19.9.0 Access-Control-Allow-Origin: * Access-Control-Allow-Credentials: true { "args": {}, "data": "hello world", "files": {}, "form": {}, "headers": { "Content-Length": "11", "Content-Type": "text/plain", "Host": "httpbin.org", "User-Agent": "UBLOX-HttpClient V3.0", "X-Amzn-Trace-Id": "Root=1-6639f3ca-3e882df46ce4a7ea53906312" }, "json": null, "origin": "185.215.193.160", "url": "http://httpbin.org/post" } DISCONNECT OK OK +UUHTTPCR: 0,7,1	Submit a post command in via direct link mode, insert an eleven length string data into terminal and get result from server. The +UUHTTPCR URC notifies the success/failure of the operation (in this example: success).

7.4 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 +UUHTTPCR: 0,4,1	Successfully submit a post command in text format and store the answer in result.txt.
AT+UHTTPER=1	+UHTTPER: 1,0,0 OK	In HTTP profile 1 the error code 0 is "No error".

7.5 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 enabled and the HTTP server port set to 443.  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.

Appendix

A Glossary

Abbreviation	Definition
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
APN	Access Point Name
ARIA	a block cipher technique
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
CPU	Central Processing Unit
CSD	Circuit-Switched Data
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
EPS	Evolved Packet System
FOTA	Firmware updates Over-The-Air
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
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

Abbreviation	Definition
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
TLS	Transport Layer Security
TTL	Time To Live
URC	Unsolicited Result Code
WAN	Wide Area Network

Related documentation

- [1] u-blox LEXI-R10 series AT commands manual, [UBXDOC-686885345-1786](#)
- [2] u-blox LEXI-R10 series data sheet, [UBX-UBX-23007594](#)
- [3] u-blox LEXI-R10 series system integration manual, [UBX-23008149](#)
- [4] u-blox EVK-R10 user guide, [UBXDOC-686885345-1985](#)
- [5] u-blox LEXI-R10 FW update application note, [UBXDOC-686885345-2005](#)
- [6] u-blox LEXI-R10 Application Development AppNote, [UBXDOC-686885345-1983](#)
- [7] 3GPP TS 27.010 V3.4.0 - Terminal Equipment to User Equipment (TE-UE) multiplexer protocol (Release 1999)
- [8] GSMA TS.34 - IoT Device Connection Efficiency Guidelines (Version 4)
- [9] u-blox Mux implementation in cellular modules application note, [UBX-13001887](#)
- [10] u-blox SARA-R422 / LEXI R422 application development guide application note, [UBX-20050829](#)
- [11] u-blox LEXI-L10 production and prototype validation guidelines application note. UBXDOC-686885345-2043. Contact tech support for this document.
- [12] RFC 5077 - Transport Layer Security (TLS) Session Resumption without Server-Side State
- [13] RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2
- [14] u-blox test server configuration, [UBX-14005690](#)

 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	14-Jun-2024	mreb	Initial release

Contact

u-blox AG

Address: Zürcherstrasse 68
8800 Thalwil
Switzerland

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