Telit Cinterion

SSL/TLS

User Guide

1vv0300989 Rev.23
2024-09-19
Released
Confidential

Technical Documentation

Contents

1 Applicability Table

Table 1: Applicability Table

Products	Platform Version ID	Technology
GL865 SERIES	10	2G
GE865-QUAD		
GE864 V2 SERIES		
GL868-DUAL		
GE910 SERIES	13	
GE910-QUAD-V3	16	
GE866-QUAD		
GL865 V3 SERIES		
GL868-DUAL V3		
HE910 SERIES	12	3G
UE910 SERIES		
UL865 SERIES		
UE866 SERIES		
LE910 Cat1 SERIES	20	4G
LE910 V2 SERIES		
LE866 SERIES	23	
ME866A1 SERIES		
LE910Cx SERIES	25	
ME910C1 SERIES	30	
ML865C1 SERIES		
ME310G1 SERIES	37	
ME910G1 SERIES		
ML865G1 SERIES		
ELS63 SERIES	52	
LE910Q1 SERIES		

Note: Platform Version ID is a reference used in the document. It identifies the different SW versions, for example 10 for SW version 10.xx.xxx, 13 for SW version 13.xx.xxx, and so on.

2 Introduction

2.1 Scope

This document describes the set of the Telit AT commands regarding the SSL/TLS protocols use.

2.2 Audience

The guide is intended for users that need to develop applications based on secure connection channels. The reader is expected to have knowledge in wireless technology as well as in SSL/TLS security protocols.

2.3 Contact Information, Support

For technical support and general questions, e-mail:

- TS-EMEA@telit.com
- TS-AMERICAS@telit.com
- TS-APAC@telit.com
- TS-SRD@telit.com
- TS-ONEEDGE@telit.com

Alternatively, use: https://www.telit.com/contact-us/

For Product information and technical documents, visit: https://www.telit.com

2.4 Conventions

Note:	Provide advice and suggestions that may be useful when integrating the module.
Danger:	This information MUST be followed, or catastrophic equipment failure or personal injury may occur.
Warning:	Alerts the user on important steps about the module integration.

All dates are in ISO 8601 format, that is YYYY-MM-DD.

2.5 Terms and Conditions

Refer to https://www.telit.com/hardware-terms-conditions/.

2.6 Disclaimer

3 AT Commands List

The following list, organized in alphabetical order, shows the AT commands covered in this User Guide. The number next to each command indicates the page of the first AT command occurrence.

4 Preliminary Information

Warning: This guide introduces the AT commands that handle SSL sockets and provides examples that describe their use. The guide does not contain examples for all the modules listed in the Applicability Table, it contains examples relating to some modules to give a trace to the reader about the use of the SSL commands.

For detailed information on command syntax, refer to the AT Commands Reference Guide [1], [6], [7], [11], [14], [15], [16], [17] or [18] depending on the module you are using.

5 PDP Context Configuration and Activation

To start working with sockets, you need to configure a PDP context using the +CGDCONT command and activate it as shown below. For more information refer to document [3].

AT+CGDCONT=<cid>,<PDP_type>,<APN>,···

Where:

<cid> PDP Context Identifier. Use the test command to know the <cid> range of the used module.

<PDP_type> a string which specifies the type of Packet Data Protocol.

<APN> Access Point Name, a string containing the logical name used to select GGSN or external packet data network. The ISP provides this parameter.

... other parameters.

Use the #SGACT command to activate the PDP.

AT#SGACT= <cid>,<stat>[,<userId>,<pwd>]

Where:

<cid> PDP Context Identifier. Use the test command to know the <cid> range of the used module.

<stat> context status: 0 = deactivate the context, 1 = activate the context.

... optional parameters.

Example

Define PDP context.

AT+CGDCONT=1,"IP","Access_Point_Name",···
OK

Before activating a PDP context, it must be bound to a socket through the #SCFG command.

AT#SGACT=1,1 ← activate the PDP context

#SGACT:212.195.45.65 ← returns the IP address provided by the network

OK

6 SSL Configuration

Before opening an SSL socket and exchange data with it, you must perform the following steps.

- Enable SSL channel
- Set authentication mode and timeouts
- Store Security data in the module if the authentication is required

6.1 Enabling an SSL Channel, #SSLEN Command

To provide communication security over a channel, enable an SSL socket using the #SSLEN command. If <Enable> parameter is not set to 1, any attempt to set SSL parameters fails.

AT#SSLEN= <SSId>,<Enable>

Where:

<SSId> Secure Socket ID. Use the AT#SSLEN=? test command to know the <SSId> range of the used module.

<Enable> status: 0 = deactivate secure socket (default), 1 = activate secure socket.

Example

AT#SSLEN=1,1 ← enable the SSL socket identified by <SSId>=1
OK

The #SSLEN command behavior depends on the number of the SSL sockets that the module supports, and on the AT instance you are using to enter the command. See chapter Examples.

Warning: SSL is a shared resource so it can be enabled only on one protocol (SSLD, HTTP,FTP, MQTT, SMTP) so, if it is already enable on one of those protocols, SSLEN=1,1 returns ERROR (SSL already activated).

6.2 SSL Security Configuration, #SSLSECCFG Command

The cipher suite is the set of algorithms used to negotiate the security settings for a network connection using the SSL/TLS network protocol. The cipher suite includes:

- Key exchange algorithm used for the authentication during handshake
- Encryption algorithm used to encrypt the message
- Hash function for data integrity

If the remote server does not support one of the cipher suites provided by the module the handshake fails.

The #SSLSECCFG command manages the cipher suites and the authentication modes as shown in the following chapters.

Note: The module uses its internal time and date to validate the certificate validity period.
If time and date are incorrectly set, the certificate validation may fail.
For additional information refer to AT+CCLK, AT#NITZ or AT#NTP.

6.2.1 2G Modules (ID 10, 13, 16)

Here is the #SSLSECCFG command syntax.

AT#SSLSECCFG= <SSId>,<CipherSuite>,<auth_mode>

Where:

<SSId> must be set to 1. Only one secure socket is available.

<CipherSuite> setting the value to 0 (default), all the available cipher suites are proposed to the remote server, see chapter Preinstalled Cipher Suites. It is the responsibility of the remote server to select one of them.
Setting a value other than zero (1÷6), the module proposes to the remote server one of the following cipher suite:

1 = TLS_RSA_WITH_RC4_128_MD5
2 = TLS_RSA_WITH_RC4_128_SHA

3 = TLS_RSA_WITH_AES_256_CBC_SHA
4 = TLS_RSA_WITH_AES_128_CBC_SHA256
5 = TLS_RSA_WITH_AES_256_CBC_SHA256
6 = TLS_RSA_WITH_AES_128_GCM_SHA256

<auth_mode>: authentication mode:
0 = SSL verify none: no authentication, no security data is needed.
1 = Server authentication mode: CA Certificate storage is needed, the most common case.
2 = Server/Client authentication mode: CA Certificate (server), Certificate (client) and Private Key (client) are needed.

The authentication mode depends on the user's application and the desired protection against intruders. If security data is required, they must be stored in PEM format via #SSLSECDATA command, refer to chapter Storing Security Data.

Warning: The usage of "*SSL verify none*" shall not be used in production environments. This option is supported only for testing/debugging purposes.

Please refer to the chapter Security Considerations.

Refer to:

- document [1] for command syntax and parameters values
- chapter Preinstalled Cipher Suites for supported protocols and preinstalled cipher suites

If you enable the unique SSL socket, identified by <SSId>=1, on an AT instance through the #SSLEN command, other AT instances cannot use the <SSId>=1 socket. To use the <SSId>=1 socket on another AT instance, you must disable the <SSId>=1 socket (enter #SSLEN=1,0 on the AT instance used to enable <SSId>=1) and activate it on the new AT instance. See chapter Examples. To have information on AT instances refer to documents [4].

6.2.2 3G Modules (ID 12), 4G Modules (ID 20, 23)

Here is the #SSLSECCFG command syntax.

AT#SSLSECCFG= <SSId>,<CipherSuite>,<auth_mode>[,<cert_format>]

Where:

<SSId> Secure Socket ID. Use the AT#SSLSECCFG=? test command to know the <SSId> range of the used module.

<CipherSuite> when 0 value is set, all the available cipher suites are proposed to the remote server within TLS handshake (i.e.: client hello), see chapter Preinstalled Cipher Suites according to the module used. It is responsibility of the remote server to select one of them.

TLS_RSA_WITH_NULL_SHA cipher suite is not included when the <CipherSuite> parameter is set to 0. Set <CipherSuite> = 4 to select this cipher suite.

Setting <CipherSuite> value different from zero (1÷5), only one cipher suite is proposed:

1 = TLS_RSA_WITH_RC4_128_MD5
2 = TLS_RSA_WITH_RC4_128_SHA
3 = TLS_RSA_WITH_AES_128_CBC_SHA
4 = TLS_RSA_WITH_NULL_SHA
5 = TLS_RSA_WITH_AES_256_CBC_SHA

<auth_mode> authentication mode:

0 = SSL verify none: no authentication, no security data is needed.
1 = Server authentication mode: CA Certificate storage is needed, the most common case.
2 = Server/Client authentication mode: CA Certificate (server), Certificate (client) and Private Key (client) are needed.

The authentication mode depends on the user's application and the desired protection against intruders. If the security data is required, it can be stored in one of the two formats: DER or PEM.

<cert_format> optional parameter. It selects the format of the certificate to be stored via #SSLSECDATA command, refer to chapter Examples.

0 = DER format
1 = PEM format, default

Assume that the module is powered on at this time, and the #SSLSECCFG command is entered without <cert_format> parameter. In this case, the default format is PEM.
If you enter the #SSLSECCFG? read command, it does not return the setting of the format to meet backward compatibility with other series. Now, enter the #SSLSECCFG command again with the <cert_format> parameter for the first time. If the read command is entered, it reports the parameter value just used. If subsequently the <cert_format> is omitted, the #SSLSECCFG? read command reports the parameter value entered the last time.

Warning: The usage of "*SSL verify none*" shall not be used in production environments. This option is supported only for testing/debugging purposes.

Please refer to the chapter Security Considerations.

Assume to use a module providing a set of SSL sockets. If you enable a SSL socket, identified by <SSId>=x, on an AT instance through the #SSLEN command, other AT instances cannot use the same <SSId>=x socket. To use the <SSId>=x socket on another AT instance, you must disable the <SSId>=x socket (enter #SSLEN=x,0 on the AT instance used to enable <SSId>=x) and activate it on the new AT instance. Different SSL sockets can be enabled on different AT instances. See chapter Examples. To have information on AT instances refer to documents [5], [10], and [12] according to the module used.

6.2.3 4G Modules (ID 25 Linux)

Here is the #SSLSECCFG command syntax.

AT#SSLSECCFG= <SSId>,<CipherSuite>,<auth_mode>[,<cert_format>]

Where:

<SSId> Secure Socket ID. Use the AT#SSLSECCFG=? test command to know the <SSId> range of the used module.

<CipherSuite> when 0 value is set, all the available cipher suites are proposed to the remote server within TLS handshake (i.e.: client hello), see chapter Preinstalled Cipher Suites. It is responsibility of the remote server to select one of them.
Setting <CipherSuite> value different from zero, only one cipher suite is proposed:

1 = TLS_RSA_WITH_3DES_EDE_CBC_SHA
2 = TLS_RSA_WITH_AES_128_CBC_SHA
3 = TLS_RSA_WITH_AES_128_CBC_SHA256
4 = TLS_RSA_WITH_AES_256_CBC_SHA
5 = TLS_RSA_WITH_AES_256_CBC_SHA256
6 = TLS_DHE_RSA_WITH_AES_128_CBC_SHA
7 = TLS_DHE_RSA_WITH_AES_256_CBC_SHA
8 = TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
9 = TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
10 = TLS_DHE_RSA_WITH_AES_256_CBC_SHA256

<auth_mode> authentication mode:
0 = SSL verify none: no authentication, no security data is needed.
1 = Server authentication mode: CA Certificate storage is needed, the most common case.
2 = Server/Client authentication mode: CA Certificate (server), Certificate (client) and Private Key (client) are needed.
The authentication mode depends on the user's application and the desired protection against intruders. If the security data is required, they can be stored in one of the two formats: DER or PEM.

<cert_format> optional parameter. It selects the format of the certificate to be stored via #SSLSECDATA command, refer to chapter Storing Security Data.
0 = DER format
1 = PEM format, default

Assume that the module is powered on at this time, and the #SSLSECCFG command is entered without <cert_format> parameter. In this case, the default format is PEM.
If you enter the #SSLSECCFG? read command, it does not return the setting of the format to meet retro compatibility with other series.
Now, enter again #SSLSECCFG command with the <cert_format> parameter for the first time. If the read command is entered, it reports the parameter value just used. If subsequently the <cert_format> is omitted, the #SSLSECCFG? read command reports the parameter value entered the last time.

Warning: The usage of "*SSL verify none*" shall not be used in production environments. This option is supported only for testing/debugging purposes.

Please refer to the chapter Security Considerations.

Assume to use a module providing a set of SSL sockets. If you enable a SSL socket, identified by <SSId>=x, on an AT instance through the #SSLEN command, other AT instances cannot use the same <SSId>=x socket. To use the <SSId>=x socket on another AT instance, you must disable the <SSId>=x socket (enter #SSLEN=x,0 on the AT instance used to enable <SSId>=x) and activate it on the new AT instance. Different SSL sockets can be enabled on different AT instances, see chapter Examples. To have information on AT instances refer to documents [5], [10], and [17] according to the used module.

6.2.4 4G Modules (ID 25 ThreadX)

Here is the #SSLSECCFG command syntax.

AT#SSLSECCFG= <SSId>,<CipherSuite>,<auth_mode>[,<cert_format>]

Where:

<SSId> Secure Socket ID. Use the AT#SSLSECCFG=? test command to know the <SSId> range of the used module.

<CipherSuite> when 0 value is set, all the available cipher suites are proposed to the remote server within TLS handshake (i.e.: client hello), see chapter

Preinstalled Cipher Suites. It is responsibility of the remote server to select one of them.
Setting <CipherSuite> value different from zero, only one cipher suite is proposed:

1 = TLS_RSA_WITH_AES_128_CBC_SHA
2 = TLS_RSA_WITH_AES_128_CBC_SHA256
3 = TLS_RSA_WITH_AES_256_CBC_SHA
4 = TLS_RSA_WITH_AES_256_CBC_SHA256
5 = TLS_DHE_RSA_WITH_AES_128_CBC_SHA
6 = TLS_DHE_RSA_WITH_AES_256_CBC_SHA
7 = TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
8 = TLS_DHE_RSA_WITH_AES_256_CBC_SHA256

<auth_mode> authentication mode:
0 = SSL verify none: no authentication, no security data is needed.
1 = Server authentication mode: CA Certificate storage is needed, the most common case.
2 = Server/Client authentication mode: CA Certificate (server), Certificate (client) and Private Key (client) are needed.
The authentication mode depends on the user's application and the desired protection against intruders. If the security data is required, they can be stored in one of the two formats: DER or PEM.

<cert_format> optional parameter. It selects the format of the certificate to be stored via #SSLSECDATA command, refer to chapter Storing Security Data.
0 = DER format
1 = PEM format, default

Assume that the module is powered on at this time, and the #SSLSECCFG command is entered without <cert_format> parameter. In this case, the default format is PEM.
If you enter the #SSLSECCFG? read command, it does not return the setting of the format to meet retro compatibility with other series.
Now, enter again #SSLSECCFG command with the <cert_format> parameter for the first time. If the read command is entered, it reports the parameter value just used. If subsequently the <cert_format> is

omitted, the #SSLSECCFG? read command reports the parameter value entered the last time.

Warning: The usage of "*SSL verify none*" shall not be used in production environments. This option is supported only for testing/debugging purposes.

Please refer to the chapter Security Considerations.

Assume to use a module providing a set of SSL sockets. If you enable a SSL socket, identified by <SSId>=x, on an AT instance through the #SSLEN command, other AT instances cannot use the same <SSId>=x socket. To use the <SSId>=x socket on another AT instance, you must disable the <SSId>=x socket (enter #SSLEN=x,0 on the AT instance used to enable <SSId>=x) and activate it on the new AT instance. Different SSL sockets can be enabled on different AT instances, see chapter Examples. To have information on AT instances refer to documents [5], [10], and [18] according to the used module.

6.2.5 4G Modules (ID 30, 37, 52)

Here is the #SSLSECCFG command syntax.

AT#SSLSECCFG=<SSId>,<CipherSuite>,<auth_mode>

For platform ID 30 refer to:

- document [14] for command syntax and parameters values
- chapter Preinstalled Cipher Suites for supported protocols and preinstalled cipher suites

For platform ID 37 refer to:

- document [15] for command syntax and parameters values
- chapter Preinstalled Cipher Suites for supported protocols and preinstalled cipher suites

For platform ID 52 refer to:

- document [16] for command syntax and parameters values
- chapter Preinstalled Cipher Suites for supported protocols and preinstalled cipher suites

6.3 Protocol Selection, #SSLSECCFG2 Command

TLS and its predecessor SSL are cryptographic protocols used over the Internet to provide secure data communication in several applications. A classic example is the HTTPS connection between Web browsers and Web servers, see chapter HTTPS Connection.

For TLS protocol, see standards:

- RFC 2246 - TLS Protocol Version 1.0
- RFC 4346 - TLS Protocol Version 1.1
- RFC 5246 - TLS Protocol Version 1.2
- RFC 8446 – TLS Protocol Version 1.3

6.3.1 2G Modules (ID 10, 13, 16)

These modules do not support #SSLSECCFG2 command, see chapter SSL Security Configuration, #SSLSECCFG Command.

6.3.2 3G Modules (ID 12), 4G modules (ID 20, 25 Linux)

Here is the #SSLSECCFG2 command syntax.

AT#SSLSECCFG2=<SSId>,<version>

[,<unused_A>[,<unused_B>[,<unused_C>[,<unused_D>]]]]

Where:

<SSId> Secure Socket ID. Use the AT#SSLSECCFG2=? test command to know the <SSId> range of the module used.

<version> It selects the SSL/TLS protocol version.

Refer to:

- document [6], [7], [17] for command syntax and parameters values

- chapter Preinstalled Cipher Suites for supported protocols and preinstalled cipher suites

6.3.3 4G Modules (ID 23, 30)

Here is the #SSLSECCFG2 command syntax.

AT#SSLSECCFG2=<SSId>,<version>[,<SNI>[,<unused_A>[,<unused_B> [,<unused_C>]]]]

Where:

<SSId> Secure Socket ID. Use the AT#SSLSECCFG2=? test command to know the <SSId> range of the module used.

<version> It selects the SSL/TLS protocol version.

<SNI> enable/disable Server Name Indication

Refer to:

- document [11], [14] for command syntax and parameters values
- chapter Preinstalled Cipher Suites for supported protocols and preinstalled cipher suites

6.3.4 4G Modules (ID 37, 25 ThreadX, 52)

Here is the #SSLSECCFG2 command syntax.

AT#SSLSECCFG2=<SSId>,<version>,<SNI>
[,customCA>[,PreloadedCA>[,MinVersion]]]

Where:

<SSId> Secure Socket ID. Use the AT#SSLSECCFG2=? test command to know the <SSId> range of the module used.

<version> select SSL/TLS protocol version.

<SNI> enable/disable Service Name Indication.

<CustomCA> mask indicating which CA certificate is used from AT#SSLSECDATA

<PreloadedCA> mask indicating which CA certificate is used from AT#SSLSECCA

<MinVersion> Set the Minimum acceptable TLS version

For platform ID 37 refer to:

- document [15] for command syntax and parameters values
- clause 12.1.7 for supported protocols and preinstalled cipher suites

For platform ID 25 ThreadX refer to:

- document [18] for command syntax and parameters values
- clause 12.1.7 for supported protocols and preinstalled cipher suites

For platform ID 52 refer to:

- document [16] for command syntax and parameters values
- clause 12.1.7 for supported protocols and preinstalled cipher suites

6.4 SNI (Server Name Indication)

SNI is an extension to the Transport Layer Security (TLS) header which allow a client to specify which hostname it is attempting to connect to at the start of the handshaking process.
The requested hostname is not encrypted in the original SNI extension, so an eavesdropper can see which site is being requested.

This allows a server to present one of multiple possible certificates on the same IP address and TCP port number and hence allows multiple secure (HTTPS) websites (or any other service over TLS) to be served by the same IP address without requiring all those sites to use the same certificate.
SNI has to be enabled when the server module is connecting to is hosted in a cloud or VPN; if SNI is not set module could receive a non-host specific SSL/TLS configuration (version/cipher suites/certificate).
This also allows a proxy to forward client traffic to the right server during TLS/SSL handshake.

6.5 Storing Security Data

The following types of security data can be stored in the modules:

- Certificates
- CA Certificates
- Private Key

The maximum size of security data depends on the used module. If a remote server has a certificate larger than the maximum size supported by the module, the authentication fails.

Chapter Get the Root CA Certificate describes a procedure to get the root CA certificate to use in a connection to an HTTPS server. See standards RFC 2459, and X509v3.

Server or Server/Client authentication is fulfilled only if you store the proper security data (certificate(s) and/or private key) in the module's NVM.

Use the following command to store, read, and delete security data.

AT#SSLSECDATA=< SSId >,<Action>,<DataType>[,<Size>]

Where:

<SSId> store identifier. Use the AT#SSLSECDATA=? test command to know the <SSId> range provided by the used module.

<Action> action identifier. Use the AT#SSLSECDATA=? test command to know the <Actions> range supported by the used module.
0 = delete security data from NVM
1 = store security data in NVM
2 = read security data from NVM
3 = store security data in RAM (supported by Platform ID 23)

<DataType> identifies the certificate/key to be stored, read or delete.

0 = Certificate of the client (module). It is needed when the Server/Client authentication mode has been configured.
1 = CA Certificate of the remote server, it is used to authenticate the remote server. It is needed when <auth_mode> parameter of the #SSLSECCFG command is set to 1 or 2.
2 = RSA private key of the client (module). It is needed if the Server/Client authentication mode has been configured.

<Size> size of the stored security data. Use the AT#SSLSECDATA=? test command to know the <Size> range provided by the used module.

Warning: To get information on AT commands syntax and related parameters, refer to the AT Command Reference Guide according to the module you are using, see chapter Related documents.

Example: storing security data

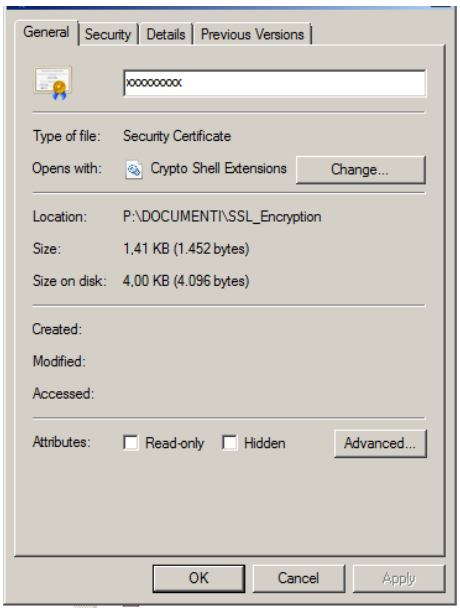
After entering the #SSLSECDATA command, the '>' prompt appears. There are two security data downloading modes according to the used certificate format set through the AT#SSLSECCFG command, see table below.

2G MODULES SERIES	3G AND 4G MODULES SERIES
Certificates can be set only in PEM format	Certificates can be in PEM or DER format

Table 2: PEM and DER Formats

Here are the downloading modes.

PEM format is supported by 2G/3G/4G.



Before downloading the certificate, you need to know the size of the certificate expressed in bytes. Use the Property dialog box, shown on the left side, to get this information.

After entering the #SSLSECDATA command, the ">" prompt is displayed. Now, you can enter the security data to be stored in NVM or RAM. Each certificate line must be terminated only with <LF> character (no <CR>), and no EOF character must be added at the end of the certificate file. Enter <ctrl>Z to close the certificate downloading.

Remember that the reserved chars "backspace" and "escape" are interpreted as control characters and the corresponding action is immediately executed.

DER format is supported by 3G/4G (not supported by ID 30 platform).

Before downloading the certificate, you need to know the size of the certificate expressed in bytes. Use the Property dialog box, shown above. When <size> bytes are downloaded, the security data is stored, and an OK message is displayed. The DER format uses the binary format; therefore, the reserved chars "backspace" and "escape" are not interpreted as control characters, and the binary file includes them inside it. The data security downloading can be done with the Telit AT Controller tool.

6.6 Get the Root CA Certificate

Assume that it is required a connection to an HTTPS server via a module, and the authentication of the remote server is needed. First, you need to know the root CA Certificate of the server, and then store it in the NVM of the module. Here is an example to get the root CA certificate.

To obtain the root CA certificate you can use a browser, running on a PC, connected to the desired HTTPS server.

During the handshake, the server sends a certificate chain, which is a list of certificates. The chain begins with the certificate of the server, and each certificate in the chain is signed by the entity identified by the next certificate in the chain. The server chain could terminate with a root CA certificate, if root CA is not sent by server it must be present locally on the client to solve the chain. The root CA certificate is always signed by the CA itself. The signatures of all certificates in the chain must be verified until the root CA certificate is reached.

Here is an example of solved certificate chain.

ServerCert → AuthorityCert1 → AuthorityCert2 ... → AuthorityCertN → RootCACert

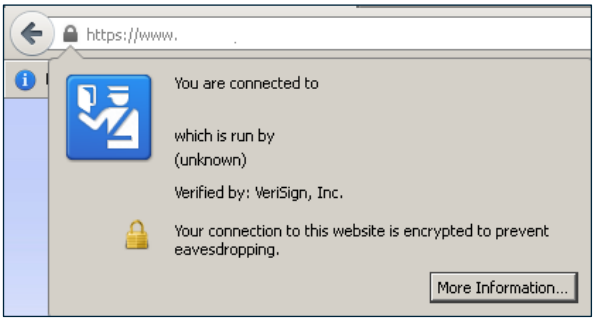
Where:

ServerCert	is the server certificate at which the client wants to be connected
AuhorityCert1...N	are certificates of intermediate authorities
RootCACert	is the certificate of a global recognized Certificate Authority

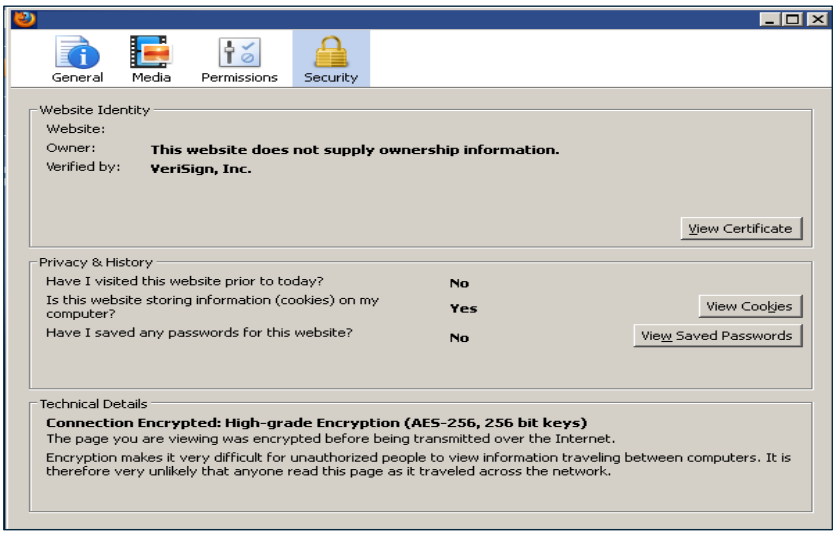
In this example is used the browser Mozilla Firefox.

After being connected to the HTTPS server, click on the lock icon on the left side of the page browser and the following dialog box appears.

Then click on "More Information" button, the next dialog box appears.

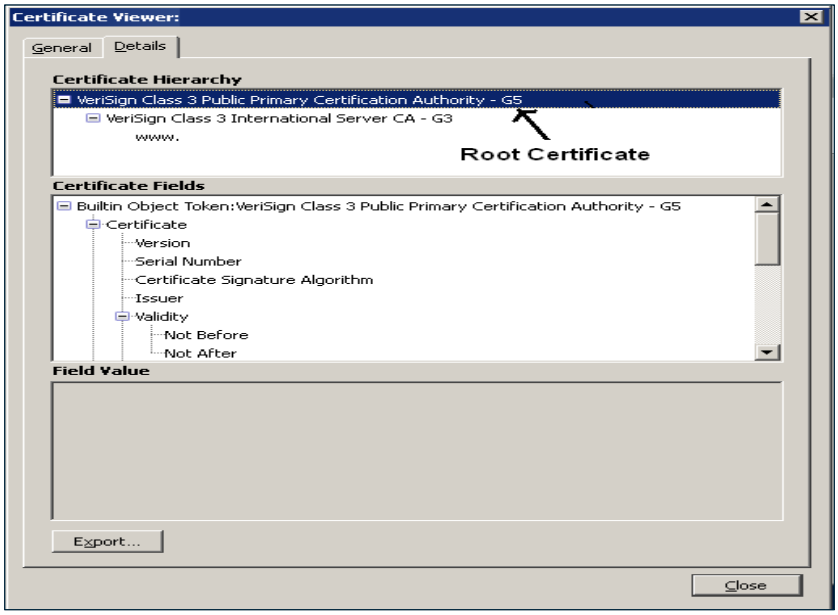


Select the "Security" tab and click on "View Certificate" button. The following dialog box appears.



Now, select "Details" Tab. The dialog box shows the "Certificate Hierarchy" section that contains the certificate chain for the selected website. The root CA certificate is the first one, select it and click on the "Export" button.

The root CA certificate is saved in a file in PEM format, now open the file via a text editor, the following structure is displayed:



-----BEGIN CERTIFICATE-----
.....
.....
.....
-----END CERTIFICATE-----

If the server is not reachable from a browser then openSSL can be used to retrieve the CA certificate.

openssl s_client -showcerts -connect www.example.com:443

openSSL will return the certificate chain and last one received is the rootCA

CONNECTED(00000004)

Certificate chain
 0 s:C = US, ST = California, L = Los Angeles, O = Internet\C2\A0Corporation\C2\A0for\C2\A0Assigned\C2\A0Names\C2\A0and\C2\A0Numbers, CN = www.example.org
   i:DC = com, DC = telital, DC = tmt, CN = Telit-Corporate-Enrollment-Service-ILT
-----BEGIN CERTIFICATE-----
...
...
-----END CERTIFICATE-----
 1 s:DC = com, DC = telital, DC = tmt, CN = Telit-Corporate-Enrollment-Service-ILT
   i:DC = com, DC = telital, DC = tmt, CN = Telit-Corporate-Root-Authority
-----BEGIN CERTIFICATE-----
...
...
-----END CERTIFICATE-----

Server certificate

The root CA certificate obtained from the procedure may be different from the one sent by the server during the handshake. In this case, contact the server administrator to obtain the root CA certificate to use.

If the root CA certificate has expired, the module (client) detects the certificate expiration when it tries to perform the connection, and an error message is returned.

See example in chapter #SSLD Command Example.

6.7 SSL Communication Configuration, #SSLCFG Command

Use the following command to configure the SSL socket, before opening it.

6.7.1 2G, 3G, 4G (ID 20, 23) Modules

Here is the #SSLCFG command syntax.

AT#SSLCFG=<SSId>,<cid>,<pktSz>,<maxTo>,<defTo>,<txTo>[,<sslSRingMode>[,<noCarrierMode>]]

Where:

<SSId> Secure Socket ID. Use the AT#SSLCFG=? test command to know the <SSId> range of the used module.

<cid> PDP Context Identifier. Use the AT#SSLCFG=? test command to know the <cid> range of the used module.

<pktSz> size of the packet used by the SSL/TCP/IP stack for data sending in ONLINE mode. The packet size can be changed according to the user's application standard message size. Small <pktSz> values introduce a higher communication overhead.

<maxTo> socket inactivity timeout. In ONLINE mode, if there is no data exchange within this timeout period the connection is closed. Increment it if a longer idle time interval is required.

<defTo> timeout value used as default value by other SSL commands whenever their timeout parameters are not set.

<txTo> time interval after which data is sent even if <pktSz> is not reached (only in ONLINE mode). The parameter value must be tuned with user's application requirements. Small <txTo> values introduce a higher communication overhead.

<sslSRingMode> presentation mode of the SSLSRING: unsolicited indication, which informs the user about new incoming data that can be read in COMMAND mode. It can be disabled using value 0.

<noCarrierMode> permits to choose between the standard NO CARRIER indication (when the socket is closed) and two verbose modes in which additional information is added to the NO CARRIER indication.

6.7.2 4G Modules (ID 25 Linux)

Here is the #SSLCFG command syntax.

AT#SSLCFG=<SSId>,<cid>,<pktSz>,<maxTo>,<defTo>,<txTo>[,<sslSRingMode>
[,<noCarrierMode>[<skipHostMismatch]]]

Where:

<SSId> Secure Socket ID. Use the AT#SSLCFG=? test command to know the <SSId> range of the used module.

<cid> PDP Context Identifier. Use the AT#SSLCFG=? test command to know the <cid> range of the used module.

<pktSz> size of the packet used by the SSL/TCP/IP stack for data sending in ONLINE mode. The packet size can be changed according to the user's application standard message size. Small <pktSz> values introduce a higher communication overhead.

<maxTo> socket inactivity timeout. In ONLINE mode, if there is no data exchange within this timeout period the connection is closed. Increment it if a longer idle time interval is required.

<defTo> timeout value used as default value by other SSL commands whenever their timeout parameters are not set.

<txTo> time interval after which data is sent even if <pktSz> is not reached (only in ONLINE mode). The parameter value must be tuned with user's application requirements. Small <txTo> values introduce a higher communication overhead.

<sslSRingMode> presentation mode of the SSLSRING: unsolicited indication, which informs the user about new incoming data that can be read in COMMAND mode. It can be disabled using value 0.

<noCarrierMode> permits to choose between the standard NO CARRIER indication (when the socket is closed) and two verbose modes in which additional information is added to the NO CARRIER indication.

<skipHostMismatch> permits to ignore Host Mismatch Alert

6.7.3 4G Modules (ID 30, 37, 25 ThreadX, 52)

Here is the #SSLCFG command syntax.

For platform ID 30:

AT#SSLCFG=<SSId>,<cid>,<pktSz>,<maxTo>,<defTo>,<txTo>[,<SSLSRingMode>
[,<noCarrierMode>[,<skipHostMismatch>[,<equalizeTx>]]]]

See document [14] for command syntax and parameters values

For platform ID 37:

AT#SSLCFG=<SSId>,<cid>,<pktSz>,<maxTo>,<defTo>,<txTo>[,<SSLSRingMode>
[,<noCarrierMode>[,<skipHostMismatch>[,<equalizeTX> [,<connTo >[,<Unused1>]]]]]]

See document [15] for command syntax and parameters values

For platform ID 25 ThreadX:

AT#SSLCFG=<SSId>,<cid>,<pktSz>,<maxTo>,<defTo>,<txTo>[,<SSLSRingMode>
[,<noCarrierMode>[,<skipHostMismatch>[,<equalizeTX> [,<connTo >[,<Unused1>]]]]]]

See document [18] for command syntax and parameters values

For platform ID 52:

AT#SSLCFG=<SSId>,<cid>,<pktSz>,<maxTo>,<defTo>,<txTo>[,<SSLSRingMode>
[,<noCarrierMode>[,<skipHostMismatch>[,<equalizeTX> [,<connTo >[,<Unused1>]]]]]]

See document [16] for command syntax and parameters values

6.8 Examples

The next section describes examples concerning the AT commands introduced in the previous chapters.

6.8.1 #SSLEN in Modules Providing one SSL Socket

This example shows the behavior of the #SSLEN command in a module providing only one SSL socket (identified by <SSId>=1). Two terminal emulators are connected to the module. In this example, the first one is connected to USIF0/COM1, the second one is connected to USB0/COM4.

Use COM1, instance 1 (parser AT0)

AT+CGMM
HE910
OK

AT#PORTCFG?
#PORTCFG: 0,0
OK

The module provides only one SSL socket.

AT#SSLCFG=?
#SSLCFG: (1),(1),(0-1500),(0-65535),(10-5000),(0-255),(0-2),(0-2),(0),(0)
OK

AT+CMEE=2
OK

AT#SSLEN?
#SSLEN: 1,0
OK

AT#SSLEN=1,1
OK

AT#SSLEN?
#SSLEN: 1,1
OK

Connect USB cable, and use COM4, instance 2 (Parser AT1)

AT+CMEE?
+CMEE: 0
OK

AT+CMEE=2
OK

AT#SSLEN?
#SSLEN: 1,1
OK

AT#SSLEN=1,0
+CME ERROR: Resource used by another instance

Use COM1
AT#SSLEN=1,0
OK

AT#SSLEN?
#SSLEN: 1,0
OK

Use COM4
AT#SSLEN?
#SSLEN: 1,0
OK

AT#SSLEN=1,1

OK

Use COM1
AT#SSLEN?
#SSLEN: 1,1
OK

AT#SSLEN=1,0
+CME ERROR: Resource used by another instance.

6.8.2 #SSLEN in Modules Providing Several SSL Sockets

This example shows the behavior of the #SSLEN command in a module providing a set of SSL sockets (example: <SSId>=1-6). Two terminal emulators are connected to the module. In this example, the first one is connected to USIF0/COM1 port, the second one is connected to USB0/COM25 port.

Use COM1, instance 1 (parser AT0)

AT+CGMM
LE866-SV1
OK

Check the current #PORTCFG configuration.

AT#PORTCFG?
#PORTCFG: 1,1
OK

The module provides a set of SSL sockets.

AT#SSLCFG=?
#SSLCFG: (1-6),(1-5),(0-1500),(0-65535),(10-5000),(0-255),(0-2),(0-2),(0),(0)
OK

AT+CMEE=2
OK

Assume to start from this SSL sockets configuration.

AT#SSLEN?
#SSLEN: 1,0
#SSLEN: 2,0
#SSLEN: 3,0
#SSLEN: 4,0
#SSLEN: 5,0
#SSLEN: 6,0
OK

AT#SSLEN=1,1
OK

AT#SSLEN?
#SSLEN: 1,1
#SSLEN: 2,0

#SSLEN: 3,0
#SSLEN: 4,0
#SSLEN: 5,0
#SSLEN: 6,0
OK

Connect USB cable, and use COM25, instance 2 (Parser AT1)

AT+CMEE?
+CMEE: 0
OK

AT+CMEE=2
OK

AT#SSLEN?
#SSLEN: 1,1
#SSLEN: 2,0
#SSLEN: 3,0
#SSLEN: 4,0
#SSLEN: 5,0
#SSLEN: 6,0
OK

AT#SSLEN=1,0
+CME ERROR: Resource used by another instance

AT#SSLEN=2,1
OK

Use COM1

AT#SSLEN=1,0
OK

AT#SSLEN?
#SSLEN: 1,0
#SSLEN: 2,1
#SSLEN: 3,0
#SSLEN: 4,0
#SSLEN: 5,0
#SSLEN: 6,0
OK

AT#SSLEN=2,0
+CME ERROR: Resource used by another instance

AT#SSLEN=3,1
OK

AT#SSLEN=4,1
OK

AT#SSLEN?
#SSLEN: 1,0
#SSLEN: 2,1
#SSLEN: 3,1
#SSLEN: 4,1
#SSLEN: 5,0
#SSLEN: 6,0
OK

Use COM25
AT#SSLEN?
#SSLEN: 1,0
#SSLEN: 2,1
#SSLEN: 3,1
#SSLEN: 4,1
#SSLEN: 5,0
#SSLEN: 6,0
OK

AT#SSLEN=1,1
OK

AT#SSLEN?
#SSLEN: 1,1
#SSLEN: 2,1
#SSLEN: 3,1
#SSLEN: 4,1
#SSLEN: 5,0
#SSLEN: 6,0

OK

AT#SSLEN=4,0
+CME ERROR: Resource used by another instance

6.8.3 The #SSLEN Command and the other SSL Commands

6.8.3.1 3G Modules (ID 12)

Before working with SSL parameters, the Secure Socket must be activated through #SSLEN command.

AT+CGMM
HE910
OK

AT+CMEE=2
OK

Use the AT#SSLEN=? test command to know the <SSId> range of the used HE910 module. It provides only one Secure Socket.

AT#SSLEN=?
#SSLEN: (1),(0,1)
OK

Check the status of Secure Socket. It is not activated.
AT#SSLEN?
#SSLEN: 1,0
OK

If Secure Socket is not activated, any attempt to work with SSL commands fails.
AT#SSLSECCFG2?
+CME ERROR: SSL not activated

Check the current SSL Security Configuration.
If Secure Socket is not activated, any attempt to work with SSL commands fails.
AT#SSLSECCFG?
+CME ERROR: SSL not activated

Check the current SSL Communication Configuration.
If Secure Socket is not activated, any attempt to work with SSL commands fails.

AT#SSLCFG?
+CME ERROR: SSL not activated

Activate the Secure Socket <SSId>=1.
AT#SSLEN=1,1
OK

Check the current SSL/TLS Protocol.
AT#SSLSECCFG2?
#SSLSECCFG2: 1,1,0,0,0,0
OK

Check the current SSL Security Configuration
AT#SSLSECCFG?
#SSLSECCFG: 1,0,0
OK

Check the current SSL Communication Configuration
AT#SSLCFG?
#SSLCFG: 1,1,300,90,100,50,0,0,0,0
OK

6.8.4 Verify None Mode

6.8.4.1 All Modules

Using the following #SSLSECCFG command configuration, the remote server chooses the cipher suite, and the authentication mode is SSL Verify None.

AT#SSLSECCFG=1,0,0
OK

In this case, no security data is required to be stored in NVM, the module is ready for SSL socket dial.

Warning: The usage of "*SSL verify none*" shall not be used in production environments. This option is supported only for testing/debugging purposes.

Please refer to the chapter Security Considerations.

6.8.5 Server Authentication Mode

6.8.5.1 2G Modules (ID 10, 13, 16)

The following #SSLSECCFG command configuration uses the TLS_RSA_WITH_RC4_128_MD5 cipher suite, and the server authentication mode.

AT#SSLSECCFG=1,1,1
OK

Store the CA certificate of the remote server in PEM format.

AT#SSLSECDATA=1,1,1,<size>
> -----BEGIN CERTIFICATE-----<LF>
[...]
-----END CERTIFICATE-----<LF>
<ctrl>Z
OK

Now, the module is ready for SSL socket dial.

6.8.5.2 3G/4G Modules

DER Format

Using the following #SSLSECCFG command configuration, the remote server chooses the cipher suite, and the Server authentication mode is set. DER format is selected, <cert_format> = 0.

AT#SSLSECCFG=1,0,1,0
OK

To store the CA certificate of the remote server in DER format use AT#SSLSECDATA with parameter <dataType>=1. If configured in DER format, when <size> bytes are entered the CA certificate will be stored. If the CA Certificate is stored successfully, the OK message will be displayed.

AT#SSLSECDATA=1,1,1,<size>
> .....
OK

Now, the module is ready for SSL socket dial.

PEM Format

Using the following #SSLSECCFG command configuration, the remote server chooses the cipher suite, and the Server authentication mode is set. PEM format is selected, <cert_format> = 1.

AT#SSLSECCFG=1,0,1,1
OK

To store the CA certificate of the remote server in PEM format use AT#SSLSECDATA with parameter <dataType>=1. If configured in PEM format, after certificates bytes are entered and Ctrl-Z char (0x1A hex) is entered the CA certificate will be stored. If the CA Certificate is stored successfully, the OK message will be displayed.

AT#SSLSECDATA=1,1,1,<size>
> -----BEGIN CERTIFICATE-----<LF>
[...]
-----END CERTIFICATE-----<LF>
<ctrl>Z
OK

Now, the module is ready for SSL socket dial.

6.8.6 Server/Client Authentication Mode

6.8.6.1 2G Modules (ID 10, 13,16)

Using the following #SSLSECCFG command configuration, the remote server chooses the cipher suite, and the Server/Client authentication mode is set.

AT#SSLSECCFG=1,0,2
OK

Store the certificate of the client (module) in PEM format.

AT#SSLSECDATA=1,1,0,<size>
> -----BEGIN CERTIFICATE-----<LF>
[...]
-----END CERTIFICATE-----<LF>

<ctrl>Z
OK

Store the CA certificate of the remote server in PEM format.

AT#SSLSECDATA=1,1,1,<size>
> -----BEGIN CERTIFICATE-----<LF>
[...]
-----END CERTIFICATE-----<LF>
<ctrl>Z
OK

Store the RSA private key of the client (module).

AT#SSLSECDATA=1,1,2,<size>
[... private key ...]
<ctrl>Z
OK

6.8.6.2 3G/4G Modules

DER Format

Using the following #SSLSECCFG command configuration, the remote server chooses the cipher suite, and the Server authentication mode is set. DER format is selected, <cert_format> = 0.

AT#SSLSECCFG=1,0,2,0
OK

To store the certificate of the client (module) in DER format use AT#SSLSECDATA with parameter <dataType>=0. If configured in DER format, when <size> bytes are entered the certificate of the client (module) will be stored. If the certificate of the client (module) is stored successfully, the OK message will be displayed.

AT#SSLSECDATA=1,1,0,<size>
> .....
OK

To store the CA certificate of the remote server in DER format use AT#SSLSECDATA with parameter <dataType>=1.

AT#SSLSECDATA=1,1,1,<size>
> .....
OK

To store the RSA private key of the client (module) in DER format use AT#SSLSECDATA with parameter <dataType>=2.

AT#SSLSECDATA=1,1,2,<size>
> .....
OK

Now, the module is ready for SSL socket dial.

PEM Format

Using the following #SSLSECCFG command configuration, the remote server chooses the cipher suite, and the Server authentication mode is set. PEM format is selected, <cert_format> = 1.

AT#SSLSECCFG=1,0,2,1
OK

To store the certificate of the client (module) in PEM format use AT#SSLSECDATA with parameter <dataType>=0. If configured in PEM format, after certificates bytes are entered and Ctrl-Z char (0x1A hex) is entered the certificate of the client (module) will be stored. If the certificate of the client (module) is stored successfully, the OK message will be displayed.

AT#SSLSECDATA=1,1,0,<size>
> -----BEGIN CERTIFICATE-----<LF>
[...]
-----END CERTIFICATE-----<LF>
<ctrl>Z
OK

To store the CA certificate of the remote server in PEM format use AT#SSLSECDATA with parameter <dataType>=1.

AT#SSLSECDATA=1,1,1,<size>
> -----BEGIN CERTIFICATE-----<LF>
[...]
-----END CERTIFICATE-----<LF>
<ctrl>Z
OK

To store the RSA private key of the client (module) in PEM format use AT#SSLSECDATA with parameter <dataType>=2.

AT#SSLSECDATA=1,1,2,<size>
[... private key ...]

<ctrl>Z
OK

Now, the module is ready for SSL socket dial.

7 Working with SSL Socket

This section describes how to open SSL socket, and exchange data using one of the following modes.

- ONLINE mode
- COMMAND mode

Use the following command to open an SSL socket.

AT#SSLD=<SSId>,<rPort>,<IPAddress>,<ClosureType>[,<connMode>[,<Timeout>]]

Where:

<SSId> Secure Socket ID. Use the test command to know the <SSId> range of the used module.

<rPort> remote port of the SSL server (usually 443).

<IPAddress> string containing an IP or hostname of the SSL server.

<ClosureType> enable/disable the capability to restore later the session, using the #SSLFASTD command, without repeating the handshake phase. See table below, and chapter Fast Dial, #SSLFASTD Command.

	<CLOSURETYPE> PARAMETER	
Platform Version ID	0	1
10, 13, 16 (2G)	SSL session id and keys are released, therefore #SSLFASTD command cannot be used to recover the last SSL session (default).	SSL session id and keys are saved, and a new connection can be established without a complete handshake using #SSLFASTD command.
12 (3G)	Zero is the only allowed value, #SSLFASTD command is not supported.	N/A
20 (4G)	Zero is the only allowed value, #SSLFASTD command is not supported.	N/A
23 (4G)	Zero is the only allowed value, #SSLFASTD command is not supported.	N/A
25 (4G)	Zero is the only allowed value, #SSLFASTD command is not supported.	N/A
30, 37, 52 (4G)	Zero is the only allowed value, #SSLFASTD command is not supported.	N/A

Table 3: #SSLFASTD Command Availability

<connMode> data exchange mode:

0 = ONLINE mode. On success, the CONNECT message is returned, and from now all bytes sent to the serial port are forwarded to the remote server.

1 = COMMAND mode. On success, the OK message is returned. After that, AT parser is still alive, and data can be exchanged by means of #SSLSEND and #SSLRECV commands.

If for any reason the handshake fails (network or remote server overload, wrong certificate, timeout expiration, etc.) an ERROR response message appears.

<Timeout> maximum allowed TCP inter-packet delay. Modules belonging to the Platform Version ID 10 and 16 (2G technology) to manage large certificates and avoid timeout expiration, must improve the CPU clock by means of the #CPUMODE=2 or 4 command.

7.1 Exchange Data with Secure Socket

7.1.1 ONLINE Mode

Open the SSL socket and wait for the CONNECT message. After receiving the CONNECT message, you can send data to the module. The Data are encrypted and sent to the server through the secure socket as soon as the packet size is reached or the txTo timeout expires; see chapter SSL Communication Configuration, #SSLCFG Command to configure these parameters.

In ONLINE mode, it is not possible to enter AT commands on the used serial port or virtual port, refer to documents [4] or [5] to have information about the serial/virtual ports. However, it is possible to suspend the connection, without closing it, by sending the escape sequence (+++). After that, the module returns the OK response and can parse the AT commands again.

ONLINE mode can be restored at any time by sending the following command.

AT#SSLO=<SSId>

Where:

<SSId> Secure Socket ID. Use the test command to know the <SSId> range of the used module.

After entering the #SSLO restore command, the CONNECT message appears, and SSL communication can continue.

If the idle inactivity timeout expires (<maxTo>, see chapter SSL Communication Configuration, #SSLCFG Command or the remote server closes the connection, the NO CARRIER message is displayed.

7.1.2 COMMAND Mode

In COMMAND mode, data can be exchanged through a SSL socket by means of the #SSLSEND, #SSLSENDEXT and #SSLRECV commands. The data exchange is performed in blocking mode.

If SSLSRING unsolicited message has been enabled by means of the #SSLCFG command (<sslSRingMode> set to 1 or 2), any new incoming data will be notified.

At any moment, the user can switch to ONLINE mode by entering the #SSLO command described in the previous chapter.

7.1.2.1 Send Data, #SSLSEND, #SSLSENDEXT Commands

Use one of the following commands to send data:

AT#SSLSEND=<SSId>[,<Timeout>]

Where:

<SSId> Secure Socket ID. Use the test command to know the <SSId> range of the used module.

<Timeout> Timeout expressed in 100 msec unit. If it is omitted, the default timeout set via AT#SSLCFG will be used (<defTo>, refer to chapter SSL Communication Configuration, #SSLCFG Command).

When the command is closed with a <CR>, the '>' prompt appears. Now, you can enter the data to be sent. To close the data block, enter <ctrl>Z, then the data are forwarded to the remote server through the secure socket. Response: OK on success, ERROR on failure.

AT#SSLSENDEXT=<SSId>, <bytestosend>[,<Timeout>]

Where:

<SSId> Secure Socket ID. Use the test command to know the <SSId> range of the used module.

<bytestosend> Number of bytes to be sent. Use the test command to know the <SSId> range of the used module.

<Timeout> Timeout expressed in 100 msec unit. If it is omitted, the default timeout set via AT#SSLCFG will be used (<defTo>, refer to chapter SSL Communication Configuration, #SSLCFG Command).

When the command is closed with <CR>, the '>' prompt appears. Now, you can enter the data to be sent. When <bytestosend> bytes have been sent, operation is automatically completed. Response: OK on success, ERROR on failure.

7.1.2.2 Receive Data

Data can be received in two different ways:

- using the #SSLRECV command (the "standard" way),
- reading data from the SSLSRING: unsolicited message.

#SSLRECV Command

Use the following command to receive data.

AT#SSLRECV=<SSId>,<MaxNumByte>[,<Timeout>]

Where:

<SSId> Secure Socket ID. Use the test command to know the <SSId> range of the used module.

<MaxNumByte> Maximum number of bytes that will be read from socket. The user can set it according to the expected amount of data.

<Timeout> Timeout expressed in 100 msec unit. If it is omitted, the default timeout set via #SSLCFG will be used (<defTo>, refer to chapter SSL Communication Configuration, #SSLCFG Command).

On success, the data are displayed in the following format:

#SSLRECV: <numBytesRead>
... received data ....
OK
Where:
<numBytesRead> number of bytes read (equal or less than <MaxNumBytes>).

If the timeout expires, the module displays the following response

#SSLRECV: 0
TIMEOUT
OK

The ERROR message appears on failure.

SSLSRING: Unsolicited Message

The SSLSRING: unsolicited message, if enabled, notifies the user about any new incoming data. Configuring <sslSRingMode>=2 by means of the #SSLCFG command (see chapter SSL Communication Configuration, #SSLCFG Command) data is displayed in the URC in this format:

SSLSRING:<SSId>,<dataLen>,<data>

Where:

<SSId> Secure Socket ID. Use the test command to know the <SSId> range of the used module.

<dataLen> Number of bytes presented in the current URC. Its maximum value within a single unsolicited message is:
256 for 2G modules
1300 for 3G/4G modules

<data> bytes of data in ASCII format. The number of bytes is <dataLen>.

7.2 Close a Secure Socket, #SSLH Command

The following command closes the SSL socket.

AT#SSLH=<SSId>,<ClosureType>

Where:

<SSId> Secure Socket ID. Use the test command to know the <SSId> range of the used module.

<ClosureType> enable/disable the capability to restore the session later, using the #SSLFASTD command, without repeating the handshake phase. See chapter Exchange Data with Secure Socket.

If the secure socket was opened in ONLINE mode, the user needs to send the escape sequence (+++) before closing it with #SSLH command, unless the communication is remotely closed, or the idle inactivity timeout expires (NO CARRIER message).

If the secure socket was opened in COMMAND mode, when the communication is remotely closed, and all data has been retrieved (#SSLRECV), you can also close on the client side and NO CARRIER message is displayed. At any moment, it is also possible to close the secure socket on client side by means of #SSLH.

7.3 Fast Dial, #SSLFASTD Command

#SSLFASTD command restores a previous suspended session avoiding full handshake and performs a speed dial, which saves time and reduces the TCP payload. It can be used if #SSLD or #SSLH command was entered with <ClosureType> parameter set to 1, in this case the previous data security is not deleted on socket closure. Refer to chapter Exchange Data with Secure Socket

Warning: #SSLFAST command is supported only by 2G Modules (Platform ID 10, 13, 16).

7.4 Examples

The next section describes examples concerning the AT commands introduced in the previous chapters.

7.4.1 ONLINE Mode

Suppose that the PDP context definition/activation, SSL socket enabling, and SSL socket security configuration are performed.

In this example, the secure socket is opened, connected to an SSL server having IP 123.124.125.126, and listening on port 443. After data exchange, the connection is suspended (+++). The #SSLS command is entered to check the SSL status, and then the

ONLINE mode is restored using #SSLO command, and so on. At the end, the SSL socket is closed.

AT#SSLD=1,443,"123.124.125.126",0,0 ← open the SSL socket in ONLINE mode
CONNECT
...
[Bidirectional data exchange]
...
+++ ← suspend the connection
OK

AT#SSLS=1 ← query the status of the Secure Socket Id = 1
#SSLS: 1,2,<cipher_suite> ← the connection is open
OK

AT#SSLO=1 ← restore the connection
CONNECT
...
[Bidirectional data exchange]
...
+++ ← suspend again the connection
OK

AT#SSLH=1 ← close SSL socket
OK

AT#SSLS=1 ← query the status of the Secure Socket Id = 1
#SSLS: 1,1 ← the connection is closed
OK

7.4.2 COMMAND Mode

Suppose that the PDP context definition/activation, SSL socket enabling, and SSL socket security configuration are performed.

In this example, the socket is opened, connected to an SSL server having IP 123.124.125.126, and listening on port 443. The data exchange is performed using #SSLSEND, #SSLSENDEXT, and #SSLRECV commands. At the end, the SSL socket is closed.

AT#SSLD=1,443,"123.124.125.126",0,1 ← open the SSL socket in COMMAND mode
OK

AT#SSLS=1 ← query the status of the Secure Socket Id = 1
#SSLS: 1,2,<cipher_suite> ← the connection is open
OK

AT#SSLSEND=1 ← sending data
> Send this string to the SSL server!<ctrl>Z
OK

AT#SSLRECV=1,15 ← receiving data
#SSLRECV: 0
TIMEOUT ← the server has not sent a response within the timeout.
OK

AT#SSLRECV=1,15
#SSLRECV: 15
Response of the ← received data
OK

AT#SSLRECV=1,15
#SSLRECV: 6
Server ← received data
OK

"Response of the Server" is the string sent by the server

AT#SSLH=1 ← close SSL socket
OK

Note: If remote server closes data communication after the data is sent and there is no more data are available to retrieve, the communication is also closed on the client side. NO CARRIER message is displayed, no #SSLH is needed.

7.4.3 Sending/Receiving Data in COMMAND Mode

Suppose that the PDP context definition/activation, SSL socket enabling, and SSL socket security configuration are performed.

In this example, the socket is opened, connected to an SSL server with IP 123.124.125.126, and listening on port 443. After data exchange in ONLINE mode, the connection is suspended and is entered the COMMAND mode. In this mode, the AT interface is active and by means of the #SSLSEND, #SSLSENDEXT and #SSLRECV commands, it is possible to continue receiving and sending data using the SSL socket still connected. At the end, the SSL socket is closed.

AT#SSLD=1,443,"123.124.125.126",0,0 ← open the SSL socket in ONLINE mode
CONNECT
...
[Bidirectional data exchange]
...
+++ ← suspend the connection and enter COMMAND mode
OK

AT#SSLS=1 ← query the status of the Secure Socket Id = 1
#SSLS: 1,2,<cipher_suite> ← the connection is open
OK

AT#SSLSEND=1 ← AT interface is still active. Send data in COMMAND mode
> Send data in command mode<ctrl>Z
OK

AT#SSLRECV=1,100 ← AT interface is still active. Receive data in COMMAND mode
#SSLRECV: 24
Response in command mode
OK

AT#SSLH=1 ← close SSL socket
OK

Note: If the remote server closes the data communication after the data is sent and there is no more data to retrieve, the communication is also closed on the client side. NO CARRIER message is displayed, and then no #SSLH is needed.

7.4.4 COMMAND Mode and SSLSRING: Unsolicited Message

These examples show how to take advantage of the unsolicited SSLSRING: feature. Mode 1 and 2 notify any incoming new records. Mode 2 also shows data, therefore #SSLRECV command is not needed.

7.4.4.1 SSLSRING: Mode = 1

Configure SSLSRING mode 1

AT#SSLCFG=1,1,300,90,100,50,1
OK

AT#SSLD=1,443,"123.124.125.126",0,1 ← open the SSL socket in COMMAND mode
OK

AT#SSLSEND=1 ← send data in COMMAND mode
> Make a request to the server<ctrl>Z
OK

SSLSRING: 1,400 ← 400 bytes are ready to be read

AT#SSLRECV=1,300← read only a part of received data
#SSLRECV: 300
<300 bytes>
OK

SSLSRING: 1,100 ← new SSLSRING with remaining data

AT#SSLRECV=1,100← read remaining data
#SSLRECV: 100
<100 bytes>
OK

NO CARRIER ← in this example the server closes the connection

7.4.4.2 SSLSRING: Mode = 2

Configure SSLSRING mode 2 plus data

AT#SSLCFG=1,1,300,90,100,50,2
OK

AT#SSLD=1,443,"123.124.125.126",0,1 ← open the SSL socket in COMMAND mode
OK

AT#SSLSEND=1 ← send data in COMMAND mode
> Make the same request of the example 1<ctrl>Z
OK

SSLSRING: 1,256,<256 bytes> ← first chunk of bytes.

SSLSRING: 1,144,<144 bytes> ← second chunk of bytes.
The module has received 400 bytes (256+144)
NO CARRIER ← in this example the server closes the connection

7.4.5 Open/Restore a SSL Socket

Suppose that the PDP context definition/activation, SSL socket enabling, and SSL socket security configuration are performed.

In this example, the socket is open, connected to an SSL server with IP 123.124.125.126, and listening on port 443; in addition, suppose that the <ClosureType> parameter is set to 1, see chapter Exchange Data with Secure Socket. Data exchange is performed in ONLINE mode, and then the connection is suspended and restored using the #SSLFASTD command. After a new data exchange, the socket is permanently closed.

AT#SSLD=1,443,"123.124.125.126",1,0 ← open the SSL socket in ONLINE mode
CONNECT
...
[Bidirectional data exchange]
...
+++ ← suspend the connection and enter COMMAND mode

OK

AT#SSLH=1 ← close SSL socket
OK

AT#SSLFASTD=1,0 ← restore the session in ONLINE mode
...
[Bidirectional data exchange]
...
+++ ← suspend the connection
OK

AT#SSLH=1,0 ← force definitive closure
OK

8 HTTPS Connection

8.1 #SSLD Command Example

Assume you have the root CA Certificate, refer to chapter Get the Root CA Certificate, and the PDP context definition/activation are performed.

This example shows the configuration of the SSL socket in server authentication mode, storing the root CA certificate, opening the socket, and starting data exchange. Thereafter, the HTTPS server responds to the module and closes the socket.

If the <Enable> parameter is not set to 1, any attempt to set SSL security configuration fails.

Warning: Security configuration is valid for SSL, HTTP, FTP services.

Enable the SSL socket <SSId>=1.

AT#SSLEN=1,1
OK

Set SSL Security Configuration: Secure Socket, CipherSuite, Authentication Mode, Certificate Format.
AT#SSLSECCFG=1,0,1
OK

Store the CA Certificate
AT#SSLSECDATA=1,1,1,<size>
> -----BEGIN CERTIFICATE-----
.....
Write the certificate got by using the procedure described in chapter Get the Root CA Certificate
.....
-----END CERTIFICATE-----
<ctrl>Z
OK

Open the SSL socket identified by <SSId>=1. The connection is open in ONLINE mode, <connMode>=0. In this example, HTTPS use the <rPort>=443.

AT#SSLD=1,443,"www.---",0,0

CONNECT

.....

The module receives a response from the HTTPS server

.....

NO CARRIER ← Server remote closure: some servers are configured to close the socket after a single request.

8.2 HTTP Get Command Example

This example uses a 3G module.

To have information on HTTP GET command request refer to RFC 2616 standard.

Define PDP context.

AT+CGDCONT=1,"IP", "Access_Point_Name"

OK

Check the current Multi-sockets/PDP contexts configuration (default).

AT#SCFG?

#SCFG: 1,1,300,90,600,50

#SCFG: 2,1,300,90,600,50

#SCFG: 3,1,300,90,600,50

#SCFG: 4,2,300,90,600,50

#SCFG: 5,2,300,90,600,50

#SCFG: 6,2,300,90,600,50

OK

Before activating a PDP context, it must be bound to a socket. Activate PDP Context <cid>=1. The command returns the IP address assigned by the network.

AT#SGACT=1,1

#SGACT: 10.7.125.7

OK

If <Enable> parameter is not set to 1, any attempt to set SSL security configuration fails.

Warning: Security configuration is valid for SSL, HTTP, FTP services.

Enable the SSL socket <SSId>=1. The SSL

AT#SSLEN=1,1

OK

Set SSL security configuration: Secure Socket, CipherSuite, Authentication Mode, Certificate Format.

AT#SSLSECCFG=1,0,1,1

OK

Store the CA certificate of the remote server in PEM format.

AT#SSLSECDATA=1,1,1,<size>

> -----BEGIN CERTIFICATE-----<LF>

[...]

-----END CERTIFICATE-----<LF>

<ctrl>Z

OK

SSL encryption can be used only by one service at a time. Therefore, to use the SSL encryption with HTTP protocol, it must be disabled for SSL and FTP services. To do this, set to 0 the following parameters: <Enable> of the #SSLEN command, and <FTPSEn> of the #FTPCFG command.

Disable the SSL encryption for SSL service: <Enable>=0

AT#SSLEN=1,0

OK

Check the current value of the <FTPSEn> parameter.

AT#FTPCFG?

#FTPCFG: 100,0,0

OK

Enable the SSL encryption, <ssl_enabled>=1, for the HTTP service, and configure the parameters of the HTTPS server.

AT#HTTPCFG=0,"server_address",443,0,,,1,120,1

OK

Send GET command to the HTTP server.

AT#HTTPQRY=0,0,"/"

OK ← GET command succeeds.

When the HTTP server answer is received, an URC is displayed on the terminal emulator.

#HTTPRING: 0,200,"text/html", ···

Type in the #HTTPRCV command to read data from HTTP server.

AT#HTTPRCV=0

<!doctype html>

<html>

.....

</html>

OK

9 FTP with TLS

FTPS is used when an application needs to connect securely using FTP. FTPS supports:

- authentication
- message integrity
- confidentiality

during a connection over an SSL/TLS secure socket, see standard [8].

The default is that the modules support the explicit mode described in the standard [8] also known as FTPES. In this mode, the FTPS client must explicitly request security from an FTPS server (implicit mode is a deprecated). When the FTPS connection is opened towards an FTPS server, the FTP command AUTH (refer to standards [8], [9]) is sent to the server to explicitly request a secure FTP connection.

To enable an FTPS connection, use:

- #FTPCFG command to enable FTPS security.
- #SSLSECCFG and #SSLSECDATA commands to configure the SSL socket, see chapter SSL Configuration .

Use the FTP commands to open control connection and data connection, see document [3]. When #FTPOPEN is used, the FTPS connection is opened toward the FTPS server. Any subsequent data port opening (#FTPLIST, #FTPGET, #FTPPUT ...) will be in protected mode.

No TLS session reuse is performed when data connection is opened: two TLS sessions are performed within an FTP session, one for control and one for data port. The Server shall be configured so that TLS reuse is not required.

The same certificates saved through #SSLSECDATA command are used for both TLS sessions, as strongly recommended by the standard [8].

For platform ID 52 the modules also support the implicit mode but need a specific configuration (not default).

9.1 #FTPOPEN, #FTPGET Commands Example

This example uses a 3G module.

Define PDP context.

```
AT+CGDCONT=1,"IP", "Access_Point_Name"
OK
```

Check the current Multi-sockets/PDP contexts configuration (default).

AT#SCFG?

#SCFG: 1,1,300,90,600,50

#SCFG: 2,1,300,90,600,50

#SCFG: 3,1,300,90,600,50

#SCFG: 4,2,300,90,600,50

#SCFG: 5,2,300,90,600,50

#SCFG: 6,2,300,90,600,50

OK

Before activating a PDP context, it must be bound to a socket. Activate PDP Context <cid>=1. The command returns the IP address assigned by the network.

AT#SGACT=1,1

#SGACT: 10.7.125.7

OK

If <Enable> parameter is not set to 1, any attempt to set SSL security configuration fails.

Warning: Security configuration is valid for SSL, HTTP, FTP services.

Enable the SSL socket <SSId>=1.

AT#SSLEN=1,1

OK

Set SSL security configuration: Secure Socket, CipherSuite, Authentication Mode, Certificate Format.

AT#SSLSECCFG=1,0,1,1

OK

Store the CA certificate of the remote server in PEM format.

AT#SSLSECDATA=1,1,1,<size>

> -----BEGIN CERTIFICATE-----<LF>

[...]

-----END CERTIFICATE-----<LF>

<ctrl>Z

OK

SSL encryption can be used only by one service at a time. Therefore, to use the SSL encryption with FTP protocol, it must be disabled for SSL and HTTP services. To do this, set to 0 the following parameters: <Enable> of the #SSLEN command, and <ssl_enabled> of the #HTTPCFG command.

Disable the SSL encryption for SSL service: <Enable>=0

AT#SSLEN=1,0
OK

Disable the SSL encryption for HTTP service: <ssl_enabled>=0
AT#HTTPCFG=0,"server_address",443,0,,,0,120,1
OK

Enable the SSL encryption for FTP service: <FTPSEn>=1.
AT#FTPCFG=<tout>,<IPPignoring>,1
OK

Enter #FTPOPEN command to send toward the FTPS server the AUTH TLS command to use the explicit TLS mode. When the TLS handshake is performed, and a secure connection is established, the <username> and <password> are sent.

AT#FTPOPEN=<server:port>,<username>,<password>[,<mode>]
OK

Now, FTP control connection is secured through TLS protocol.

Use the #FTPGET command to open a data connection and get the "file.txt" from the FTPS server.
AT#FTPGET="file.txt"
CONNECT

Now, the data port is connected, and the TLS handshake is performed, FTP data connection is secured through TLS protocol and the "file.txt" downloading is started.

.....
.....
.....
NO CARRIER

AT#FTPCLOSE ← close the FTPS connection
OK

10 MQTT

MQTT is an OASIS standard lightweight, publish-subscribe network protocol for the Internet of Things (IoT).

You can use #MQEN command that initializes MQTT client, #MQCFG command that configures Broker URL and port and #MQCONN command that establishes the socket and connects to the broker to open a connection.

You can initialize and connect up to two MQTT client instances simultaneously.

To get commands and parameters descriptions see documents [3] and [14], [15] or [16] according to the module used.

10.1 Examples

10.1.1 MQTT client connection secured (ID 30, 37, 52)

This example shows the establishing a secured MQTT connection (MQTT with SSL).

Enable reports in verbose format.

AT+CMEE=2

OK

Check PDP contexts.

AT+CGDCONT?

OK ← no PDP context are defined.

Define PDP context <cid>=1.

AT+CGDCONT=1,"IP", "Access_Point_Name"

OK

Check PDP contexts.

AT+CGDCONT?

+CGDCONT: 1,"IP","Access_Point_Name","0.0.0.0",0,0

OK

Activate PDP context <cid>=1. The command returns the IP address assigned by the network to the module.

AT#SGACT=1,1

#SGACT: 37.176.124.199

OK

Enable MQTT client.

AT#MQEN=1,1

OK

Check MQTT client state

AT#MQEN?

#MQEN: 1,1 ← Instance 1 is enabled.

#MQEN: 2,0

OK

Configure server URL, Port number and PDP cid initialized before. Also, enable SSL if required. If SSL is to be enabled but ine instance of it already enable, this command will give error.

AT#MQCFG=1,"mqtt_broker_address",mqtt_broker_port,1,1 ← the last field set equal to 1 is to enable TLS over MQTT

OK

Starting from this point, all #SSL commands can be used to provide TLS related configurations. Also do note that SSL instance in the AT#SSL commands should be same as the MQTT instance of the client.

Select the TLS version, authentication that you need. Client-server authentication is enabled in this example.

AT#SSLSECCFG=1,0,2

OK

Charge certificate or certificates + private key as per your authentication method.

AT#SSLSECDATA=1,1,1,<size>

← Charge CA certificate here

OK

AT#SSLSECDATA=1,1,0,<size>

← Charge client certificate here

OK

AT#SSLSECDATA=1,1,2,<size>

← Charge RSA Private key here

OK

Provide other MQTT configurations if required.

AT#MQCFG2=1,60,1

OK

AT#MQTCFG=1,30

OK

Open the MQTT connection. Provide client id, username and password. If username and password are not required enter empty strings

AT#MQCONN=1,"client_id","",""

OK

Check the MQTT client status.

AT#MQCONN?

#MQCONN: 1,1 ← Instance 1 is in connected state.

OK

.. Perform MQTT operations ..

Disconnect the MQTT client

AT#MQDISC=1

OK

Check the MQTT client status

AT#MQCONN?

#MQCONN: 1,0 ← Instance 1 is in disconnected state.

OK

Disable MQTT client

AT#MQEN =1,0

OK

Check MQTT client state

AT#MQEN?

#MQEN: 1,0 ← Instance 1 is disabled.

#MQEN: 2,0

OK.

10.1.2 Connection with AWS server (ID 30)

This example shows the establishing secured MQTT connection with AWS server on 4G modules with Platform ID 30.

Enable reports in verbose format.

AT+CMEE=2

OK

Check PDP contexts.

AT+CGDCONT?

OK ← no PDP context are defined.

Define PDP context <cid>=1.

AT+CGDCONT=1,"IP", "Access_Point_Name"

OK

Check PDP contexts.

AT+CGDCONT?

+CGDCONT: 1,"IP","Access_Point_Name","0.0.0.0",0,0

OK

Activate PDP context <cid>=1. The command returns the IP address assigned by the network to the module.

AT#SGACT=1,1

#SGACT: 37.176.124.199

OK

Enable MQTT client.

AT#MQEN=1,1

OK

Configure server URL, Port number and PDP cid initialized before. Also, enable SSL.

AT#MQCFG=1,"mqtt_broker_address",8883,1,1 ← the last field set equal to 1 is to enable TLS over MQTT

OK

Starting from this point, all #SSL commands can be used to provide TLS related configurations. Also do note that SSL instance in the AT#SSL commands should be same as the MQTT instance of the client.

Select the TLS version, authentication that you need. Client-server authentication is enabled in this example.

AT#SSLSECCFG=1,0,2

OK

Charge certificate or certificates + private key as per your authentication method.

AT#SSLSECDATA=1,1,1,<size>

← Charge CA certificate here

OK

AT#SSLSECDATA=1,1,0,<size>

← Charge client certificate here

OK

AT#SSLSECDATA=1,1,2,<size>

← Charge RSA Private key here

OK

Set Other general SSL parameters

AT#SSLCFG=1,1,300,90,100,50,0,0,1,0

OK

Enable SNI and select TLS version.

AT#SSLSECCFG2=1,3,1

OK

Provide other MQTT configurations if required.

AT#MQCFG2=1,60,1

OK

AT#MQTCFG=1,30

OK

Open the MQTT connection. Provide client id, username and password. If username and password are not required enter empty strings

AT#MQCONN=1,"client_id","",""

OK

Check the MQTT client status.

AT#MQCONN?

#MQCONN: 1,1 ← Instance 1 is in connected state.

OK.

10.1.3 Connection with AWS server (ID 37)

This example shows the establishing secured MQTT connection with AWS server on 4G modules with Platform ID 37 and 52.

Enable reports in verbose format.

AT+CMEE=2

OK

Check PDP contexts.

AT+CGDCONT?

OK ← no PDP context are defined.

Define PDP context <cid>=1.

AT+CGDCONT=1,"IP", "Access_Point_Name"

OK

Check PDP contexts.

AT+CGDCONT?

+CGDCONT: 1,"IP","Access_Point_Name","0.0.0.0",0,0

OK

Activate PDP context <cid>=1. The command returns the IP address assigned by the network to the module.

AT#SGACT=1,1

#SGACT: 37.176.124.199

OK

Enable MQTT client.

AT#MQEN=1,1

OK

Configure server URL, Port number and PDP cid initialized before. Also, enable SSL.

AT#MQCFG=1,"mqtt_broker_address",8883,1,1 ← the last field set equal to 1 is to enable TLS over MQTT

OK

Starting from this point, all #SSL commands can be used to provide TLS related configurations. Also do note that SSL instance in the AT#SSL commands should be same as the MQTT instance of the client.

Select the TLS version, authentication that you need. Client-server authentication is enabled in this example.

AT#SSLSECCFG=1,0,2

OK

Charge certificate or certificates + private key as per your authentication method.

AT#SSLSECDATA=1,1,1,<size>

← Charge CA certificate here

OK

AT#SSLSECDATA=1,1,0,<size>

← Charge client certificate here

OK

AT#SSLSECDATA=1,1,2,<size>

← Charge RSA Private key here

OK

Set Other general SSL parameters

AT#SSLCFG=1,1,300,90,100,50,0,0,1,0,0,0

OK

Enable SNI and select custom/preloaded certificate to be used. Here we are selecting customer certificate 1 and preloaded certificate 1. If you want to use custom starfield certificate, manage it using command AT#SSLSECCA and configure the same.

AT#SSLSECCFG2=1,3,1,1,1

OK

Provide other MQTT configurations if required.

AT#MQCFG2=1,60,1

OK

AT#MQTCFG=1,30

OK

Open the MQTT connection. Provide client id, username and password. If username and password are not required enter empty strings

AT#MQCONN=1,"client_id","",""

OK

Check the MQTT client status.

AT#MQCONN?

#MQCONN: 1,1 ← Instance 1 is in connected state.

OK

11 Security

11.1 Security Considerations

The primary usage of TLS (Transport Layer Security) protocol is to provide encryption, authentication, and data integrity for the data in transit communications to avoid possible eavesdropping and/or alteration of the transmitted data.

A TLS connection is initiated by using the TLS handshake process where the client and the server:

- Define the TLS version (e.g., TLS1.2, TLS1.3) they will use
- Definition on the cipher suites they will use
- The client authenticates the identity of the remote server using the server public x.509 certificate
- The client and the server share securely a temporary session key for encrypting the communication after the TLS handshake phase is complete.

In this scenario the client and the server need to define different configurations to setup correctly the secure TLS channel. Telit provides a large set of options inside the AT SSL/TLS command interface, to securely setup a communication channel and support a broad range of customer scenarios. SSL/TLS was established in the mid-1990s, and in these years has undergone many changes due to vulnerabilities exploited throughout the years. In detail multiple security vulnerabilities have been discovered on different SSL/TLS protocol implementations. Moreover other design weaknesses turned into real vulnerabilities such as *WeakDH*, *LogJam*, *FREAK*, *SWEET32*, which could be exploited by highly resource attackers, or more practical vulnerabilities such as *POODLE*, *ROBOT*.

In this scenario Telit recommends the following cybersecurity best practices:

- NIST SP 800-52 rev2 – "Guidelines for TLS Implementations"

  https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r2.pdf

- SSLLabs – "SSL and TLS Deployment Best Practices"

  https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices

- NSA – "Eliminating Obsolete Transport Layer Security (TLS) Protocol Configurations

  https://media.defense.gov/2021/Jan/05/2002560140/-1/-1/0/ELIMINATING_OBSOLETE_TLS_UOO197443-20.PDF

In detail, at the time of writing, the following TLS configuration is the recommended cybersecurity standard to secure data in transit:

- Use TLS1.3 or TLS1.2 protocols which offers modern authenticated encryption schemes and improved security
- Prefer, when available, AEAD (Authenticated Encryption with Associated Data) *cipher suites* (e.g., CHACHA20_POLY1305, GCM) which guarantee performance improvements compared to other cipher suites and prevents some attack scenarios
- Key Exchange algorithm with Perfect Forward Secrecy (PFS), (e.g., ECDHE) to enforce the protection against the use of compromised old keys. PFS assures that a compromised private key will not also compromise past session keys
- Always enable server authentication based on the x.509 certificate validation chain to avoid potential Man-in-the-Middle attack scenarios.

Warning: SSL and earlier TLS versions are considered a deprecated solution and should not be used in production environments.

These protocols could be affected by various security design flaws vulnerabilities as discussed in the previous paragraphs. SSL/TLS AT command interface implements a set of broad cryptographic capabilities, to better fit each customer needs and environment scenario (including testing/debugging environments). Telit supports some earlier SSL/TLS versions and different type of cipher suites, only for backward compatibility with legacy infrastructures and for testing/debugging purposes.

It's a customer responsibility to choose the TLS configuration suitable for their specific cyber risk scenario.

12 Appendix

12.1 Preinstalled Cipher Suites

Here are the cipher suites supported by the modules (clients).

Warning: SSL V3.0, TLS V1.0 and TLS V1.1 are obsolete or not recommended security protocol or cipher suite.

Please refer to the chapter Security Considerations.

12.1.1 2G Modules (ID 10, 13, 16)

The table shows the cipher suites supported by the protocols provided by the 2G modules (Platform ID 10, 13, 16). The cipher suites can be set individually through <CipherSuites> parameter (#SSLSECCFG command) using one of the values, different from 0, indicated in the <CipherSuites> column. <CipherSuites>=0 proposes to the server all the cipher suites in the table.

		PROTOCOLS			
CIPHER SUITES	<CIPHERSUITES>	SSL V3.0	TLS V1.0	TLS V1.1	TLS V1.2
TLS_RSA_WITH_RC4_128_MD5	0, 1		•		
TLS_RSA_WITH_RC4_128_SHA	0, 2		•	•	•
TLS_RSA_WITH_AES_256_CBC_SHA	0, 3		•	•	•
TLS_RSA_WITH_AES_128_CBC_SHA256	0, 4				•
TLS_RSA_WITH_AES_256_CBC_SHA256	0, 5				•
TLS_RSA_WITH_AES_128_GCM_SHA256	0, 6				•

Table 4: Cipher Suites 2G Modules (ID 10, 13, 16)

12.1.2 3G Modules (ID 12)

The table shows the cipher suites supported by the protocols provided by the 3G modules (Platform ID 12). The cipher suites in the gray area can be set individually through <CipherSuites> parameter (#SSLSECCFG command) using one of the values, other than 0, indicated in the <CipherSuites> column. <CipherSuites>=0 proposes to the server all the cipher suites in the table, except TLS_RSA_WITH_NULL_SHA that must be selected using <CipherSuites>=4.

		PROTOCOLS			
CIPHER SUITES	<CIPHERSUITES>	SSL v3.0	TLS v1.0	TLS v1.1	TLS v1.2
TLS_RSA_WITH_RC4_128_MD5	0, 1		•	•	•
TLS_RSA_WITH_RC4_128_SHA	0, 2		•	•	•
TLS_RSA_WITH_AES_128_CBC_SHA	0, 3		•	•	•
TLS_RSA_WITH_NULL_SHA	4	+	+	+	+
TLS_RSA_WITH_AES_256_CBC_SHA	0, 5		•	•	•
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	0				•
TLS_RSA_WITH_AES_256_CBC_SHA256	0				•
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	0				•
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	0				•
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	0				•
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	0				•
TLS_RSA_WITH_AES_128_GCM_SHA256	0				•
TLS_RSA_WITH_AES_128_CBC_SHA256	0				•

Table 5: Cipher Suites 3G Modules (ID 12)

(+): Generally, the cipher suite is supported by all protocols, but this may not be valid for some SSL stack versions.

12.1.3 4G Modules (ID 20)

The table shows the cipher suites supported by the protocols provided by the 4G modules (Platform ID 20). The cipher suites in the gray area can be set individually through <CipherSuites> parameter (#SSLSECCFG command) using one of the values, different from 0, indicated in the <CipherSuites> column. <CipherSuites>=0 proposes to the server all the cipher suites in the table, except TLS_RSA_WITH_NULL_SHA that must be selected using <CipherSuites>=4.

		PROTOCOLS			
CIPHER SUITES	<CipherSuites>	SSL v3.0	TLS v1.0	TLS v1.1	TLS v1.2
TLS_RSA_WITH_RC4_128_MD5	0, 1	•	•	•	•
TLS_RSA_WITH_RC4_128_SHA	0, 2	•	•	•	•
TLS_RSA_WITH_AES_128_CBC_SHA	0, 3	•	•	•	•
TLS_RSA_WITH_NULL_SHA	4	+	+	+	+
TLS_RSA_WITH_AES_256_CBC_SHA1	0, 5	•	•	•	•
TLS_RSA_WITH_AES_128_CBC_SHA256	0				•
TLS_RSA_WITH_AES_256_CBC_SHA256	0				•
TLS_RSA_WITH_AES_128_GCM_SHA256	0				•
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	0				•
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	0				•
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	0				•
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	0				•
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	0				•

Table 6: Cipher Suites 4G Modules (ID 20)

(+): Generally, the cipher suite is supported by all protocols, but this may not be valid for some SSL stack versions.

12.1.4 4G Modules (ID 23)

The table shows the cipher suites supported by the protocols provided by the 4G modules (Platform ID 23). The cipher suites in the gray area can be set individually through the <CipherSuites> parameter (#SSLSECCFG command) using one of the values, different from 0, indicated in the <CipherSuites> column. <CipherSuites>=0 proposes to the server all the cipher suites in the table, except TLS_RSA_WITH_NULL_SHA that must be selected using <CipherSuites>=4.

		PROTOCOLS			
CIPHER SUITES	<CipherSuites>	SSL v3.0	TLS v1.0	TLS v1.1	TLS v1.2
TLS_RSA_WITH_RC4_128_MD5	0, 1	•	•	•	•
TLS_RSA_WITH_RC4_128_SHA	0, 2	•	•	•	•
TLS_RSA_WITH_AES_128_CBC_SHA	0, 3	•	•	•	•
TLS_RSA_WITH_NULL_SHA	4	+	+	+	+
TLS_RSA_WITH_AES_256_CBC_SHA	0, 5	•	•	•	•
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	0				•
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	0				•
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	0				•
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	0				•
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	0	•	•	•	•
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	0	•	•	•	•
TLS_SRP_SHA_DSS_WITH_AES_256_CBC_SHA	0	•	•	•	•
TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA	0	•	•	•	•
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	0				•
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	0				•
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	0				•
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	0				•
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	0	•	•	•	•
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	0	•	•	•	•
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA	0	•	•	•	•

		PROTOCOLS			
CIPHER SUITES	<CipherSuites>	SSL v3.0	TLS v1.0	TLS v1.1	TLS v1.2
TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA	0		•	•	•
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	0	•			•
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	0				•
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	0				•
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	0				•
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	0	•	•	•	•
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	0	•	•	•	•
TLS_RSA_WITH_AES_256_GCM_SHA384	0				•
TLS_RSA_WITH_AES_256_CBC_SHA256	0				•
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA	0	•	•	•	•
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	0	•	•	•	•
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	0	•	•	•	•
TLS_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA	0	•	•	•	•
TLS_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA	0	•	•	•	•
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	0	•	•	•	•
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	0	•	•	•	•
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA	0	•	•	•	•
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA	0	•	•	•	•
TLS_RSA_WITH_3DES_EDE_CBC_SHA	0	•	•	•	•
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	0				•
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	0				•
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	0				•
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	0				•
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	0	•	•	•	•
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	0	•	•	•	•
TLS_SRP_SHA_DSS_WITH_AES_128_CBC_SHA	0	•	•	•	•
TLS_SRP_SHA_RSA_WITH_AES_128_CBC_SHA	0	•	•	•	•
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256	0				•
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	0				•
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	0				•

CIPHER SUITES	<CipherSuites>	PROTOCOLS			
		SSL v3.0	TLS v1.0	TLS v1.1	TLS v1.2
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	0				•
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	0	•	•	•	•
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	0	•	•	•	•
TLS_DHE_RSA_WITH_SEED_CBC_SHA	0	•	•	•	•
TLS_DHE_DSS_WITH_SEED_CBC_SHA	0	•	•	•	•
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA	0	•	•	•	•
TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA	0	•	•	•	•
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256	0				•
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256	0				•
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	0				•
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	0				•
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	0	•	•	•	•
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	0	•	•	•	•
TLS_RSA_WITH_AES_128_GCM_SHA256	0				•
TLS_RSA_WITH_AES_128_CBC_SHA256	0				•
TLS_RSA_WITH_SEED_CBC_SHA	0	•	•	•	•
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA	0	•	•	•	•
TLS_ECDHE_RSA_WITH_RC4_128_SHA	0	•	•	•	•
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	0	•	•	•	•
TLS_ECDH_RSA_WITH_RC4_128_SHA	0	•	•	•	•
TLS_ECDH_ECDSA_WITH_RC4_128_SHA	0	•	•	•	•
TLS_DHE_RSA_WITH_DES_CBC_SHA	0	•	•	•	•
TLS_DHE_DSS_WITH_DES_CBC_SHA	0	•	•	•	•
TLS_RSA_WITH_DES_CBC_SHA	0	•	•	•	•
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	0	•	•	•	•
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	0	•	•	•	•
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA	0	•	•	•	•
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5	0	•	•	•	•
TLS_RSA_EXPORT_WITH_RC4_40_MD5	0	•	•	•	•
TLS_EMPTY_RENEGOTIATION_INFO_SCSV	0	•	•	•	•

Table 7: Cipher Suites 4G Modules (ID 23)

(+): Generally, the cipher suite is supported by all protocols, but this may not be valid for some SSL stack versions.

12.1.5 4G Modules (ID 25 Linux)

The table shows the cipher suites supported by the protocols provided by the 4G modules (Platform ID 25 Linux). The cipher suites can be set individually through <CipherSuites> parameter (#SSLSECCFG command) using one of the values, different from 0, indicated in the <CipherSuites> column. <CipherSuites>=0 proposes to the server all the cipher suites in the table.

		PROTOCOLS			
CIPHER SUITES	<CipherSuites>	SSL v3.0	TLS v1.0	TLS v1.1	TLS v1.2
TLS_RSA_WITH_3DES_EDE_CBC_SHA	0, 1		•	•	•
TLS_RSA_WITH_AES_128_CBC_SHA	0, 2		•	•	•
TLS_RSA_WITH_AES_128_CBC_SHA256	0, 3				•
TLS_RSA_WITH_AES_256_CBC_SHA	0, 4		•	•	•
TLS_RSA_WITH_AES_256_CBC_SHA256	0, 5				•
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	0,6		•	•	•
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	0,7		•	•	•
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	0,8		•	•	
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	0,9				•
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	0,10				•

Table 8: Cipher Suites 4G Modules (ID 25 Linux)

12.1.6 4G Modules (ID 30)

The table shows the cipher suites supported by the protocols provided by the 4G modules (Platform ID 30). The cipher suites can be set individually through <CipherSuites> parameter (#SSLSECCFG command) using one of the values, different from 0, indicated in the <CipherSuites> column. <CipherSuites>=0 proposes to the server all the cipher suites in the table.

		PROTOCOLS			
CIPHER SUITES	<CipherSuites>	TLS v1.0	TLS v1.1	TLS v1.2	TLS v1.3
TLS_RSA_WITH_AES_128_CBC_SHA	0, 3	•	•	•	
TLS_RSA_WITH_AES_256_CBC_SHA	0, 5	•	•	•	
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	0, 7	•	•	•	
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	0, 9	•	•	•	
TLS_RSA_WITH_AES_128_CBC_SHA256	0, 10			•	
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	0, 11			•	
TLS_RSA_WITH_AES_256_CBC_SHA256	0, 12			•	
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	0, 13			•	
TLS_AES_128_GCM_SHA256 (0x1301)					+
TLS_AES_256_GCM_SHA384 (0x1302)					+
TLS_CHACHA20_POLY1305_SHA256 (0x1303)					+

Table 9: Cipher Suites 4G Modules (ID 30)

(+) TLS v 1.3 protocol will be supported by software version xx9 only for FTP service.

12.1.7 4G Modules (ID 37, ID 25 ThreadX)

The table shows the cipher suites supported by the protocols provided by the 4G modules (Platform ID 37, ID 25 ThreadX). The cipher suites inside gray area can be set individually through the <CipherSuites> parameter (#SSLSECCFG command) using one of the decimal values, different from 0, indicated in the <CipherSuites> column, or one hex value shown in the Cipher Suite column between round brackets. <CipherSuites>=0 proposes to the server all the cipher suites in the gray area.

All other cipher suites, not inside gray area, can be set individually using only the hex value shown in the Cipher Suite column between round brackets.

	Protocols				
Cipher Suite	<CipherSuites>	TLS v1.0	TLS v1.1	TLS v1.2	TLS v1.3
TLS_RSA_WITH_AES_128_CBC_SHA (0x002F)	0, 3	•	•	•	
TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)	0, 5	•	•	•	
TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)	0, 7	•	•	•	
TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)	0, 9	•	•	•	
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003C)	0, 10			•	
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x0067)	0, 11			•	
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003D)	0, 12			•	
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x006B)	0, 13			•	
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009C)				•	
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009D)				•	
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x009E)				•	
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x009F)				•	
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xC009)		•	•	•	
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xC00A)		•	•	•	
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xC013)		•	•	•	
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xC014)		•	•	•	
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xC023)				•	

	Protocols				
Cipher Suite	<CipherSuites>	TLS v1.0	TLS v1.1	TLS v1.2	TLS v1.3
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xC024)				•	
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xC027)				•	
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xC028)				•	
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xC02B)				•	
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xC02C)				•	
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xC02F)				•	
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xC030)				•	
TLS_RSA_WITH_AES_128_CCM_8 (0xC0A0)				•	
TLS_RSA_WITH_AES_256_CCM_8 (0xC0A1)				•	
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xCCA8)				•	
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256(0xCCA9)				•	
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xCCAA)				•	
TLS_AES_128_GCM_SHA256 (0x1301)					+
TLS_AES_256_GCM_SHA384 (0x1302)					+
TLS_CHACHA20_POLY1305_SHA256 (0x1303)					+

Table 10: Cipher Suites 4G Modules (ID 37, ID 25 ThreadX)

(+) TLS v 1.3 protocol will be supported from software version xx2.

12.1.8 4G Modules (ID 52)

The table shows the cipher suites supported by the protocols provided by the 4G modules (Platform ID 52). The cipher suites can be set individually through <CipherSuites> parameter (#SSLSECCFG command) using one of the values, different from 0, indicated in the <CipherSuites> column. <CipherSuites>=0 proposes to the server all the cipher suites in the table.

		PROTOCOLS			
CIPHER SUITES	<CipherSuites>	TLS v1.0	TLS v1.1	TLS v1.2	TLS v1.3
TLS_RSA_WITH_AES_128_CBC_SHA	0, 0x002f	•	•	•	
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	0, 0x0033	•	•	•	
TLS_RSA_WITH_AES_256_CBC_SHA	0, 0x0035	•	•	•	
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	0, 0x0039	•	•	•	
TLS_RSA_WITH_AES_128_CBC_SHA256	0, 0x003c			•	
TLS_RSA_WITH_AES_256_CBC_SHA256	0, 0x003d			•	
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	0, 0x0067			•	
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	0, 0x006b			•	
TLS_RSA_WITH_AES_128_GCM_SHA256	0, 0x009c			•	+
TLS_RSA_WITH_AES_256_GCM_SHA384	0, 0x009d			•	+
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	0, 0x009e			•	+
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	0, 0x009f			•	+
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	0, 0xc004	•	•	•	+
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	0, 0xc005	•	•	•	+
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	0, 0xc009	•	•	•	+
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	0, 0xc00a	•	•	•	+
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	0, 0xc00e	•	•	•	
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	0, 0xc00f	•	•	•	
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	0, 0xc013	•	•	•	
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	0, 0xc014	•	•	•	
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	0, 0xc023			•	
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	0, 0xc024			•	
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	0, 0xc025			•	

		PROTOCOLS			
CIPHER SUITES	<CipherSuites>	TLS v1.0	TLS v1.1	TLS v1.2	TLS v1.3
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	0, 0xc026			•	
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	0, 0xc027			•	
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	0, 0xc028			•	
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	0, 0xc029			•	
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	0, 0xc02a			•	
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	0, 0xc02b			•	
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	0, 0xc02c			•	
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256	0, 0xc02d			•	
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	0, 0xc02e			•	
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	0, 0xc02f			•	
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	0, 0xc030			•	
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256	0, 0xc031			•	
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	0, 0xc032			•	
TLS_RSA_WITH_AES_128_CCM_8	0, 0xc0a0			•	
TLS_RSA_WITH_AES_256_CCM_8	0, 0xc0a1			•	

Table 11: Cipher Suites 4G Modules (ID 52)

(+) TLS v 1.3 protocol is not supported by software.

12.2 SSL Error Codes

Telit's modules provide the AT+CMEE command to enable/disable and select the error report format. The error report can take two formats: numerical and verbose. The table below summarizes the error reports generated by the SSL AT commands in accordance with the selected format.

Numerical format: AT+CMEE=1	Verbose format: AT+CMEE=2
830	SSL generic error
831	SSL cannot activate
832	SSL socket error
833	SSL not connected
834	SSL already connected
835	SSL already activated
836	SSL not activated
837	SSL certs and keys wrong or not stored
838	SSL error enc/dec data
839	SSL error during handshake
840	SSL disconnected
841	SSL invalid state

Table 11: SSL Error Codes

13 Acronyms and Abbreviations

Table 12: Acronyms and Abbreviations

Acronym	Definition
AEAD	Authenticated Encryption with Associated Data
CA	Certification Authority
DER	Distinguished Encoding Rules
FTPS	File Transfer Protocol Secure
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
HTTPS	Hyper Text Transfer Protocol over Secure Socket Layer
ISP	Internet Service Provider
NVM	Non-Volatile Memory
PDP	Packet Data Protocol
PEM	Privacy Enhanced Mail
PFS	Perfect Forward Secrecy
PKCS	Public-Key Cryptography Standards
RSA	Stands for the first letter of the names of the algorithm designers
SSL	Secure Socket Layer
SUPL	Secure User Plane Location
TLS	Transport Layer Security
URC	Unsolicited Result Code

14 Related Documents

For current documentation and downloads, refer to https://dz.telit.com/.

Table 13: Related Documents

S.no	Doc Code	Document Title
1	80000ST10025A	AT Command Reference Guide
2	1VV0300784	Telit Modules Software User Guide
3	80000ST10028A	IP Easy User Guide
4	80000NT10045A	Virtual Serial Device, Application Note
5	1VV0300971	Telit 3G Modules Ports Arrangements, User Guide
6	80378ST10091A	Telit 3G Modules AT Commands Reference Guide
7	80446ST10707A	LE910 V2 Series AT Commands Reference Guide
8	-	RFC 4217 Standard
9	-	RFC 2228 Standard
10	1VV0301252	LE910 V2, LE910 Cat1 Ports Arrangements User Guide
11	80471ST10691A	LE/ME 866 Series AT Commands Reference Guide
12	1VV0301469	LE866, ME866A1 Ports Arrangements User Guide
13	80502ST10950A	LE910Cx AT Command Reference Guide
14	80529ST10815A	ME910C1/ML865C1 AT Commands Reference Guide
15	80617ST10991A	ME310G1/ME910G1/ML865G1 AT Commands Reference Guide
16	80798ST11194A	ELS63 and LE910Q1 AT Commands Reference Guide
17	80586ST11192A	LE910Cx Linux AT Commands Reference Guide
18	80586ST11193A	LE910Cx ThreadX AT Commands Reference Guide

15 Document History

Table 14: Document History

Revision	Date	Changes
23	2024-09-19	New Document Template Added chapters: 6.4. SNI (Server Name Indication) 11. Security Considerations Added products ELS63 and LE910Q1 Added several "Warning note" where unsecure or default unsecure functionalities are defined Modified chapters: Former chapter 4. Protocol Selection, #SSLSECCFG2 Command moved and modified as subchapter 6.3 of chapter 6. SSL configuration 6.2. SSL Security Configuration, #SSLSECCFG Command 6.6. Get the Root CA Certificate Former chapter 5.4 Examples moved and modified into subchapter 6.8 Examples 13. Acronyms and Abbreviations
22	2022-05-27	Updated clause 5.6 SSL Communication Configuration, #SSLCFG Command for Platform ID 30, 37 Modules (sub-clause 5.6.3)
21	2022-05-05	Added Note in clause 5.2. Minor changes on the layout. Legal Notices updated
20	2021-07-06	Added chapter MQTT Protocol Minor changes on the language and on the layout Legal Notices updated
19	2020-07-03	The table in chapter 9.1.7 has been updated, the following cipher suites has been dropped out: TLS_PSK_WITH_AES_128_GCM_SHA256, TLS_PSK_WITH_AES_256_GCM_SHA384, TLS_PSK_WITH_AES_128_CBC_SHA256, TLS_PSK_WITH_AES_256_CBC_SHA384, TLS_PSK_WITH_AES_128_CBC_SHA, TLS_PSK_WITH_AES_256_CBC_SHA. The #SSLSECCFG2 syntax in chapter 4.5 has been updated.
18	2020-05-29	Product series added: ME910C1 SERIES, NE910C1 SERIES, ML865C1 SERIES; ME310G1 SERIES, ME910G1 SERIES, ML865G1 SERIES. The format of the document has been reviewed, new chapters have been added, other changed. The cipher suites tables of modules belonging to different platforms versions has been updated. The SSL Error Codes table in chapter 9.2 has been updated The document [13] in chapter 1.5 has been updated.
17	2019-10-03	Added note in chapter 5.
16	2019-02-07	Added chapter: 3G (Platform ID 12) Modules
15	2019-01-29	Added product series: LE910Cx
14	2017-11-28	Updated chapter 4.2.1, and the table in chapter 8.1.1.

13	2017-10-24	The chapters structure has been reorganized. Added: Product series: LE910 Cat1, and ME866A1; AT Command List; Chapters 6.2 HTTP Get Command Example, and 8.1 Preinstalled Cipher Suite. Updated: Chapter 1.5 Related Documents.
12	2017-06-15	The document is fully revised, and chapters are reorganized. A new template is used. Product series removed: GC864, GE864, GT86x, HE920, UE910 V2, DE910, CE910, CL865 Product series added: UL865, LE910 V2, and LE866. In the Applicability Table, has been added the Platform Version Identifier (ID). It is used as reference in the document.
11	2015-10-28	Updated Applicability Table: CE910-DUAL 18.22.003, CL865-DUAL 18.42.013
10	2015-04-07	Update for TLS_RSA_WITH_AES_256_CBC_SHA supported by HE910 Update for #SSLSECCFG2 to set TLS version Updated Applicability Table
9	2014-07-11	Update for HE910 regarding additional cmd mode features introduced with CR700: SSLSRING mode 2, noCarrierMode and extended range for minimum timeout of #SSLSEND/RECV
8	2014-05-30	Added reference to SSLSRING feature. New sslSRingMode and noCarrierMode config parameters. Removed chunk size limitation.
7	2013-10-10	Added products in the applicability table: UE910 V2 19.10.x21, HE910 V2 14.20.xx1, HE910 V2 14.10.xx1
6	2013-09-13	In the Applicability Table have been added the following products: GE910-GNSS/13.00.xx4, GL865-QUAD V3/16.00xx3, GE910-QUAD V3/16.00.xx3, UE910/12.00.004
5	2013-05-02	Update for HE910: client authentication support, FTP over TLS support. Enhancements regarding FTP over TLS for all families.
4	2013-03-15	Modified figures in chapter 3.3.1. Added note in chapter 3.3. Added explanation for HE910: new values 1 to 4 available of #SSLSECCFG param <cipher_suite>, new value 0 available of #SSLSECCFG param <auth_mode>, Updated Applicability Table: added GL865-DUAL V3, GL868-DUAL V3 and updated software versions.
3	2012-12-14	Added notes in chapters 3.2, and 3.3.1
2	2012-11-07	Added GE910 module and HE910 family modules. The document has been updated according to the added modules.
1	2012-02-03	Minor changes
0	2011-10-11	First issue

From Mod.0818 Rev.13