SSL/TLS User Guide

1vv0300989 Rev.6 – 2013-09-13

Making machines talk.

APPLICABILITY TABLE

	SW Versions
GC Family (Compact)	
GC864-QUAD	10.00.xx7
GC864-QUAD V2	10.00.xx7
GC864-DUAL V2	10.00.xx7
GE/GL Family (Embedded)	
GE864-QUAD	10.00.xx7
GE864-QUAD V2	10.00.xx7
GE864-QUAD Automotive V2	10.00.xx7
GE864-QUAD ATEX	10.00.xx7
GE864-DUAL V2	10.00.xx7
GE864-GPS	10.00.xx7
GE865-QUAD	10.00.xx7
GL865-DUAL	10.00.xx7
GL865-QUAD	10.00.xx7
GL868-DUAL	10.00.xx7
GE910-QUAD	13.00.xx3
GE910-GNSS	13.00.xx4
GL865-DUAL V3	16.00.xx2
GL865-QUAD V3	16.00.xx3
GL868-DUAL V3	16.00.xx2
GE910-QUAD V3	16.00.xx3
GT Family (Terminal)	
GT863-PY	10.00.xx7
GT864-QUAD	10.00.xx7
GT864-PY	10.00.xx7
HE910 Family	
HE910 [1]	12.00.xx4
HE910-GA	12.00.xx4
HE910-D	12.00.xx4
HE910-EUR / HE910-EUD	12.00.xx4
HE910-EUG / HE910-NAG	12.00.xx4
HE910-NAR / HE910-NAD	12.00.xx4
UE910 Family	
UE910-EUR / UE910-EUD	12.00.xx4
UE910-NAR / UE910-NAD	12.00.xx4

Note: the features described in the present document are provided by the products equipped with the software versions equal or higher than the versions shown in the table.

[1] HE910 is the "type name" of the products marketed as HE910-G & HE910-DG.

SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE

Notice

While reasonable efforts have been made to assure the accuracy of this document, Telit assumes no liability resulting from any inaccuracies or omissions in this document, or from use of the information obtained herein. The information in this document has been carefully checked and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies or omissions. Telit reserves the right to make changes to any products described herein and reserves the right to revise this document and to make changes from time to time in content hereof with no obligation to notify any person of revisions or changes. Telit does not assume any liability arising out of the application or use of any product, software, or circuit described herein; neither does it convey license under its patent rights or the rights of others.

It is possible that this publication may contain references to, or information about Telit products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that Telit intends to announce such Telit products, programming, or services in your country.

Copyrights

This instruction manual and the Telit products described in this instruction manual may be, include or describe copyrighted Telit material, such as computer programs stored in semiconductor memories or other media. Laws in the Italy and other countries preserve for Telit and its licensors certain exclusive rights for copyrighted material, including the exclusive right to copy, reproduce in any form, distribute and make derivative works of the copyrighted material. Accordingly, any copyrighted material of Telit and its licensors contained herein or in the Telit products described in this instruction manual may not be copied, reproduced, distributed, merged or modified in any manner without the express written permission of Telit. Furthermore, the purchase of Telit products shall not be deemed to grant either directly or by implication, estoppel, or otherwise, any license under the copyrights, patents or patent applications of Telit, as arises by operation of law in the sale of a product.

Computer Software Copyrights

The Telit and 3rd Party supplied Software (SW) products described in this instruction manual may include copyrighted Telit and other 3rd Party supplied computer programs stored in semiconductor memories or other media. Laws in the Italy and other countries preserve for Telit and other 3rd Party supplied SW certain exclusive rights for copyrighted computer programs, including the exclusive right to copy or reproduce in any form the copyrighted computer program. Accordingly, any copyrighted Telit or other 3rd Party supplied SW computer programs contained in the Telit products described in this instruction manual may not be copied (reverse engineered) or reproduced in any manner without the express written permission of Telit or the 3rd Party SW supplier. Furthermore, the purchase of Telit products shall not be deemed to grant either directly or by implication, estoppel, or otherwise, any license under the copyrights, patents or patent applications of Telit or other 3rd Party supplied SW, except for the normal non-exclusive, royalty free license to use that arises by operation of law in the sale of a product.

Usage and Disclosure Restrictions

License Agreements

The software described in this document is the property of Telit and its licensors. It is furnished by express license agreement only and may be used only in accordance with the terms of such an agreement.

Copyrighted Materials

Software and documentation are copyrighted materials. Making unauthorized copies is prohibited by law. No part of the software or documentation may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, without prior written permission of Telit

High Risk Materials

Components, units, or third-party products used in the product described herein are NOT fault-tolerant and are NOT designed, manufactured, or intended for use as on-line control equipment in the following hazardous environments requiring fail-safe controls: the operation of Nuclear Facilities, Aircraft Navigation or Aircraft Communication Systems, Air Traffic Control, Life Support, or Weapons Systems (High Risk Activities"). Telit and its supplier(s) specifically disclaim any expressed or implied warranty of fitness for such High Risk Activities.

Trademarks

TELIT and the Stylized T Logo are registered in Trademark Office. All other product or service names are the property of their respective owners.

Contents

Tables

1. Introduction

1.1. Scope

Scope of this document is to provide the description of the set of the Telit AT commands relating to the SSL/TLS protocol use.

1.2. Audience

This document is intended for people that need to develop applications using secure sockets.

The reader is expected to have knowledge in wireless technology as well as in Telit's AT Commands interface. A basic knowledge of SSL/TLS security protocol is also needed.

1.3. Contact Information, Support

For general contact, technical support, to report documentation errors and to order manuals, contact Telit Technical Support Center (TTSC) at:

TS-EMEA@telit.com
TS-NORTHAMERICA@telit.com
TS-LATINAMERICA@telit.com
TS-APAC@telit.com

Alternatively, use:

http://www.telit.com/en/products/technical-support-center/contact.php

For detailed information about where you can buy the Telit modules or for recommendations on accessories and components visit:

http://www.telit.com

To register for product news and announcements or for product questions contact Telit Technical Support Center (TTSC).

Our aim is to make this guide as helpful as possible. Keep us informed of your comments and suggestions for improvements.

Telit appreciates feedback from the users of our information.

1.4. Document Organization

This document contains the following chapters:

Chapter 1: provides a scope for this document, target audience, contact and support information, and text conventions.

Chapter 2: is about context setting, activation, data states. Furthermore it gives some information about SSL server requirements.

Chapter 3: describes the steps to be followed in order to configure security settings and data as well as general parameters.

Chapter 4: describes how to connect the module to an SSL server and how to perform data exchange.

Chapter 5: describes how to use an FTPS connection.

Chapter 6: SSL Error Codes.

Chapter 7: Document History.

Chapter 8: Abbreviation and acronyms.

1.5. Text Conventions

STOP ***Danger – This information MUST be followed or catastrophic equipment failure or bodily injury may occur.***

⚠ ***Caution or Warning – Alerts the user to important points about integrating the module, if these points are not followed, the module and end user equipment may fail or malfunction.***

ℹ **Tip or Information – Provides advice and suggestions that may be useful when integrating the module.**

All dates are in ISO 8601 format, i.e. YYYY-MM-DD.

1.6. Related Documents

[1] AT Command Reference Guide, 80000ST10025a

[2] Telit Modules Software User Guide, 1vv0300784

[3] IP Easy User Guide, 80000ST10028A

[4] Virtual Serial Device, Application Note, 80000NT10045A

[5] HE910/UE910 Families Ports Arrangements, User Guide, 1vv0300971

[6] HE910/UE910 AT Commands Reference Guide, 80378ST10091A

2. Preliminary operations

Warning: the present document deals with several AT commands and Telit’s modules families using different technologies (GSM/GPRS, HSPA). To have detailed syntax information about the AT commands please, refer to the AT Commands Reference Guide [1] and [6] in accordance with the used Telit module.

2.1. IP Preliminary Operations

2.1.1. Configuring the PDP Context

Before working with AT commands concerning secure sockets, the activation of a PDP context is needed.

The PDP context parameters consist in a set of information identifying the Internet entry point interface provided by the ISP. This can be done using the following command:

```
AT+CGDCONT=<cid>,IP,<APN>,…
```

<cid> is the PDP Context Identifier, a numeric parameter which specifies a particular PDP context definition.

<APN> is the Access Point Name, a string that represents logical name used to select GGSN or external packet data network.

More information about this command, such as optional parameters omitted in this description, can be found in document [1] or [6] as stated in chapter 2.

Warning: recalling that the SSL socket can be opened only with <cid> = 1, see chapter 3.4, therefore in AT+CGDCONT command you must set <cid> = 1, see the following example:

<APN> = “ibox.tim.it” (Italian operator “TIM”)

```
AT+CGDCONT=1,IP,”ibox.tim.it”, …
OK
```

2.1.2. Activating the PDP Context

The activation of the PDP context previously defined is performed via the next command. More information about the context activation command can be found in the document [3].

AT#SGACT= <Cntx Id>,<Status>, [<Username>],[<Password>]

Where:

<Cntx Id> is the context that you want to activate/deactivate.

<Status> is the desired context status (0 means deactivation, 1 activation).

Warning: recalling that the SSL socket can be opened only with <cid> = 1, see chapter 3.4, therefore in AT#SGACT command you must set <Cntx Id> = 1, see the following example:

```
AT#SGACT=1,1                ← Activate the PDP context
#SGACT:212.195.45.65
OK
```

In the successful case, the response message of the AT#SGACT command reports an IP address provided by the network.

2.2. SSL/TLS

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.

For protocol details refer to [RFC 2246; The TLS Protocol Version 1.0] .

For details about certificates refer to [RFC 2459; X509v3].

2.2.1. Cipher Suites

The cipher suite represents the set of algorithms which are used to negotiate the security settings for a network connection using the SSL/TLS network protocol. The cipher suite includes a key exchange algorithm (used for the authentication during the handshake), an encryption algorithm (used to encrypt the message) and the hash function for data integrity. The Tab. 1 shows the cipher suites provided by the Telit modules families. Hereafter are specified the algorithms that are used by the cipher suites listed in the table:

TLS_RSA_WITH_RC4_128_MD5 uses:

- RSA for authentication
- RC4_128 as encryption algorithm
- MD5 as hash function for data integrity

TLS_RSA_WITH_RC4_128_SHA uses:

- RSA for authentication
- RC4_128 as encryption algorithm
- SHA as hash function for data integrity

TLS_RSA_WITH_AES_256_CBC_SHA uses:

- RSA for authentication
- AES_256 as encryption algorithm
- SHA as hash function for data integrity

TLS _RSA_WITH_AES_128_CBC_SHA uses:

- RSA for authentication
- AES_128 as encryption algorithm
- SHA as hash function for data integrity

TLS_RSA_WITH_NULL_SHA uses:

- RSA for authentication
- NULL as encryption algorithm
- SHA as hash function for data integrity

TLS v 1 Cipher Suites			
GC, GE/GL, and GT families		HE910/UE910 families	
<Cipher Suite>[2]	Cipher Suite Name	<Cipher Suite>	Cipher Suite Name
0	Refer to chapter 3.2	0	Refer to chapter 3.2
1	TLS_RSA_WITH_RC4_128_MD5	1	TLS_RSA_WITH_RC4_128_MD5
2	TLS_RSA_WITH_RC4_128_SHA	2	TLS_RSA_WITH_RC4_128_SHA
3	TLS_RSA_WITH_AES_256_CBC_SHA	3	TLS_RSA_WITH_AES_128_CBC_SHA
N/A	/	4	TLS_RSA_WITH_NULL_SHA

Tab. 1: Cipher Suites

If the server involved in the connection doesn't support one of these cipher suites, the protocol handshake fails.

2.2.2. Security Data Size

The Telit modules allow the storage of different type of security data (Certificates, CA Certificates, and Private Key). The maximum size of security data is 2047 bytes; this means that a server with a bigger certificate cannot be authenticated, refer to chapter 3.3.

A procedure to get the CA certificate to be used in case of a connection to an HTTPS server is explained in chapter 3.3.1.

[2] Refer to the AT#SSLSECDATA command, chapter 3.2

3. Configuring SSL

Before opening an SSL socket and exchanging data through it, you need to perform the following steps:

- SSL channel must be enabled
- Authentication settings and timeouts can be configured
- Security data can be stored within the module if the authentication is needed

Basically, these are the main steps that you have to follow, but there are slight differences among the Telit modules families. The differences are pointed out, where it is needed, on the next pages.

3.1. Enabling an SSL Channel

The first step to be done to provide communication security over a channel is to enable an SSL socket. This can be performed using the following command:

AT#SSLEN= <SSId>,<Enable>

Where:

<SSId>: must be set to 1[3]. It is the only Secure Socket ID available

<Enable>: is the desired status: 0 = disable, 1 = enable.

Warning –without entering this command, any attempt to set SSL parameters by means of an SSL AT command fails.

```
AT#SSLEN=1,1          ← Enable the SSL socket
OK
```

Note: After enabling SSL on an AT instance, SSL can not be used by other AT instances, it is needed to disable it (#SSLEN=1,0) and activate it in the other instance. To have a detailed description about instance refer to [4] and [5] in accordance with the used module.

[3] At the moment, Telit modules provide a single Secure Socket.

3.2. Configuring SSL Security

The AT#SSLSECCFG command provides two parameters to configure the communication channel according to the user's security architecture:

- **cipher suite**
- **authentication mode**

Modules belonging to the GC, GE/GL, and GT families:

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

Where:

<SSId>: must be set to 1. It is the only Secure Socket ID available

<Cipher Suite>: setting the value 0 all the available cipher suites are proposed to the server. It is responsibility of the remote server to select one of them.

0 = TLS_RSA_WITH_RC4_128_MD5 + TLS_RSA_WITH_RC4_128_SHA + TLS_RSA_WITH_AES_256_CBC_SHA

Setting values different from zero, only one cipher suite is proposed as follows:

1 = TLS_RSA_WITH_RC4_128_MD5

2 = TLS_RSA_WITH_RC4_128_SHA

3 = TLS_RSA_WITH_AES_256_CBC_SHA

<auth_mode>: is the authentication mode

0 **= SSL verify none:** no authentication, no security data is needed at all

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 authentication mode is set to 1 or 2 the storing of the security data is needed. Only PEM format is available, refer to the ***Warning*** in chapter 3.3.

Modules belonging to the HE910/UE910 families:

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

Where:

<SSId>: must be set to 1. It is the only Secure Socket ID available

<Cipher Suite>: setting the value 0 all the available cipher suites are proposed to the server. It is responsibility of the remote server to select one of them.

0 =TLS_RSA_WITH_AES_128_CBC_SHA + TLS_RSA_WITH_RC4_128_SHA + TLS_RSA_WITH_RC4_128_MD5

Setting values different from zero, only one cipher suite is proposed as follows:

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

TLS_RSA_WITH_NULL_SHA cipher suite is not included when the <Cipher Suite> parameter is set to 0. To select this cipher suite it is required to set <Cipher Suite> equal to 4.

<auth_mode>: is the authentication mode

0 **= SSL verify none:** no authentication, no security data is needed at all

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 authentication mode is set to 1 or 2 the storing of the security data is needed.

<cert_format>: is an optional parameter. It selects the format of the certificate to be stored via #SSLSECDATA command, refer to chapter 3.3.

0 = DER format

1 = PEM format

Note: it is assumed that the module is powered on and the AT#SSLSECCFG command is entered without <cert_format> parameter, the default format is PEM. In this case the AT#SSLSECCFG? read command doesn't return the setting of the format in order to meet retro compatibility with other families. Now, let's assume that AT#SSLSECCFG command

is entered again, but using 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 AT#SSLSECCFG? read command reports the parameter value entered the last time.

3.3. Storing Security Data

Server or Server/Client authentication can be accomplished only if you store the proper security data (certificate(s) and/or private key) into the module’s NVM. This operation is performed via the AT#SSLSECDATA command.

Modules belonging to the GC, GE/GL, and GT families:

The command supports the operations of: writing, reading, and deleting. The command syntax is:

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

Where:

<SSId>: must be set to 1. It is the only Secure Socket ID available

<Action>: is the action to be performed

0 = deleting

1 = writing

2 = reading

<DataType>: identifies the certificate/key to be stored or read

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>: is the size of the stored security data.

Immediately after entering the command, the ‘>’ prompt is displayed: you can enter the security data to be stored and confirm the editing end with the 0x1A character (<ctrl>Z).

Warning: Certificates MUST be in PEM format. When you store them within the module, remember that at the end of each command-line only the <LF> character is needed (without <CR>). Furthermore, be aware that some serial terminals add an undesired EOF at the end of the certificate; in this case EOF must be removed before entering <ctrl>Z.
It is advisable to note that when using PEM format, the reserved chars "backspace" and "escape" are interpreted as control characters and the corresponding action is immediately executed.

Modules belonging to the HE910/UE910 families:

Note: in case of CA Certificate is already stored (for instance: SUPL), it could be possible to avoid the using of the following command.

The command supports the operations of: writing, reading, and deleting. The command syntax is:

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

Where:

<SSId>: must be set to 1. It is the only Secure Socket ID available

<Action>: is the action to be performed

0 = deleting

1 = writing

2 = reading

<DataType>: is the certificate/key to be stored or read.

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.

2 RSA private key of the client (module). It is needed if the Server/Client authentication mode has been configured.

Note: only PKCS#1 format is supported for HE910/UE910 families

<size> is the size of the stored security data.

Immediately after entering the command, the '>' prompt is displayed. There are two entering mode depending on the used certificate format. The format is configured via the <**cert_format**> parameter of the #SSLSECCFG command:

PEM format: you enter the certificate to be stored and confirm its end with the 0x1A character (<ctrl>Z).

Note: when PEM format is selected, the reserved chars "backspace" and "escape" are interpreted as control characters and the corresponding action is immediately executed.

DER format: you enter the certificate to be stored. When <size> bytes are entered, the security data is stored and an OK message is displayed.

Note: when DER format (file in binary format) is selected, the reserved chars "backspace" and "escape" are not interpreted as control characters because the binary file could include them inside it.

3.3.1. How to Get the CA Certificate

This chapter describes an example of procedure to get the CA certificate to be used in case of a connection to an HTTPS server. During the handshake, the server sends a certificate chain, in other words it sends 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 chain terminates with a root CA certificate. 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. Follow and example of Certificate Chain:

ServerCert -> AuthorityCert1 -> AuthorityCert2 … -> AuthorityCertN -> RootCACert

Where:

ServerCert: is the certificate of the server at which the user wants to connect to;

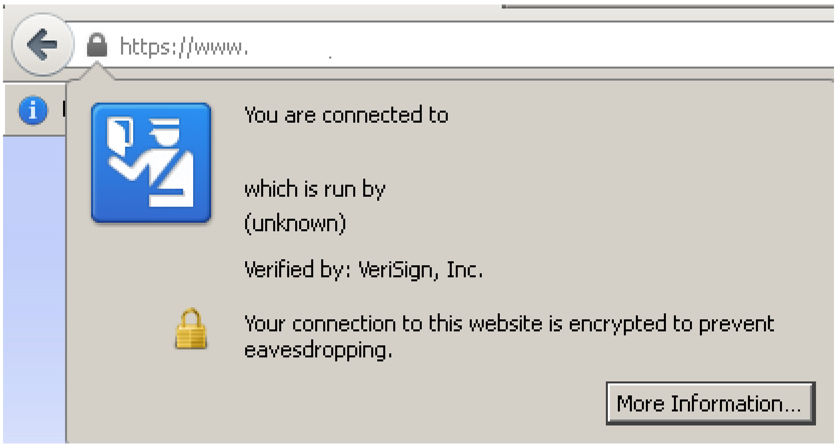
AuhorityCert1…N: are CA certificates of intermediate authorities;

RootCACert: is the certificate of a global recognized Certificate Authority.

Referring to the Certificate Chain example: each certificate of the chain (e.g. Cert1) is signed by the authorities of the next certificate (on its right in the chain, e.g. Cert2), and so on up to CertN. RootCACert is self-signed. The CA certificate to be stored in the Telit module in order to perform the server authentication is the RootCACert.

If you need to be connected to an HTTPS server via a Telit module, first of all you must store in the NVM of the module the relating Root CA Certificate to allow with successful result the server authentication activity. To get the needed Root CA Certificate you can use, for example, a browser connected to the desired HTTPS server as shown in the next pages. After getting the Root CA Certificate, it can be stored in the module. In the following 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 screenshot is displayed.



https://www.

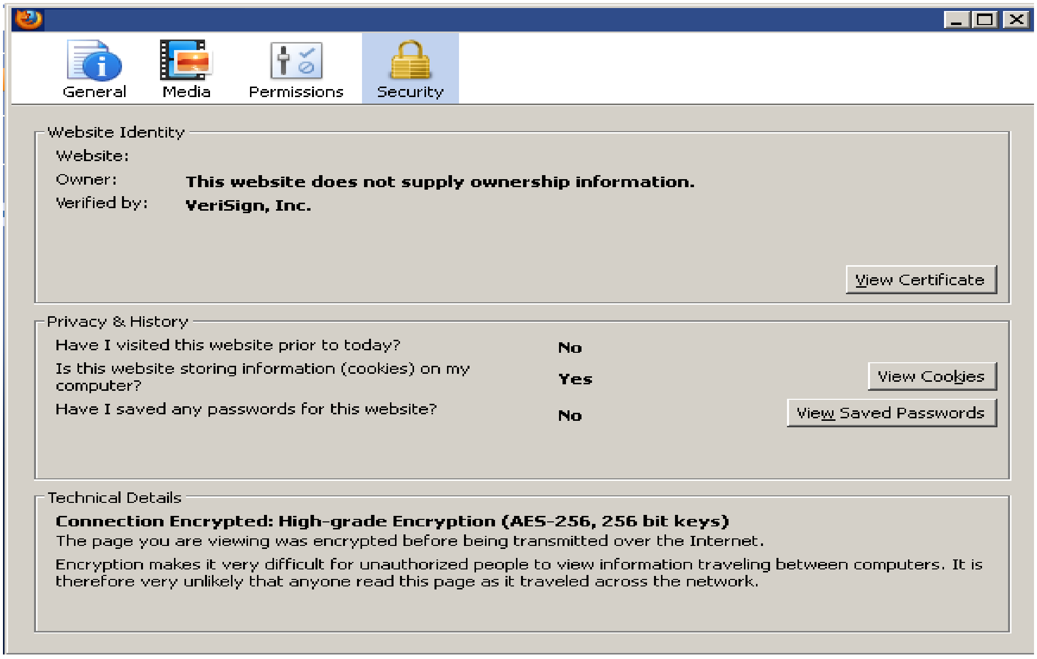
You are connected to

which is run by
(unknown)

Verified by: VeriSign, Inc.

Your connection to this website is encrypted to prevent eavesdropping.

Then click on "More Information" button. In the new window, see below, select the "Security" tab, and click on "View Certificate" button.



Website Identity

Website:

Owner: **This website does not supply ownership information.**

Verified by: **VeriSign, Inc.**

Privacy & History

Have I visited this website prior to today?	**No**	
Is this website storing information (cookies) on my computer?	**Yes**	View Cookies
Have I saved any passwords for this website?	**No**	View Saved Passwords

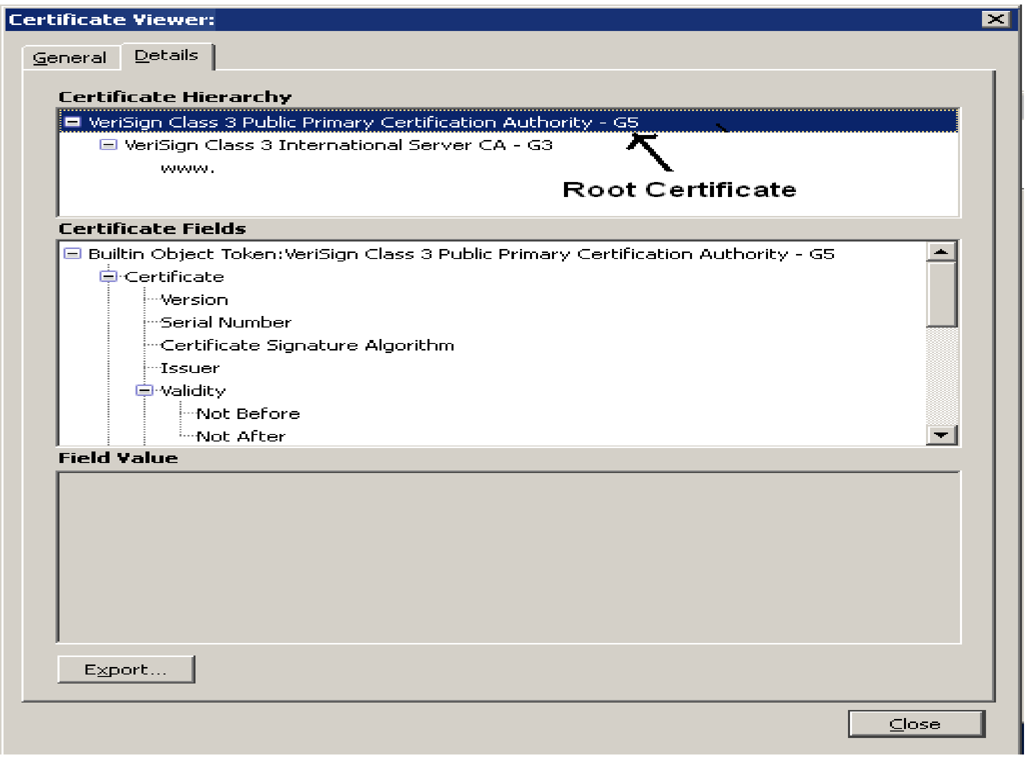
Technical Details

Connection Encrypted: High-grade Encryption (AES-256, 256 bit keys)

The page you are viewing was encrypted before being transmitted over the Internet.

Encryption makes it very difficult for unauthorized people to view information traveling between computers. It is therefore very unlikely that anyone read this page as it traveled across the network.

Now, select “Detail” Tab, the following screenshot is displayed.



Certificate Hierarchy

- VeriSign Class 3 Public Primary Certification Authority - G5
  - VeriSign Class 3 International Server CA - G3
    - www.

Root Certificate

Certificate Fields

- Builtin Object Token:VeriSign Class 3 Public Primary Certification Authority - G5
  - Certificate
    - Version
    - Serial Number
    - Certificate Signature Algorithm
    - Issuer
    - Validity
      - Not Before
      - Not After

Field Value

The screenshot above shows a “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 certificate is saved into a file in PEM format, now open the file via a text editor, the following structure is displayed; see the example described in chapter 4.5.5:

```
-----BEGIN CERTIFICATE-----
………..
………..
………..
-----END CERTIFICATE-----
```

Note: the CA certificate obtained via the procedure described above may be different from the one actually sent by the server during the handshake. In these cases, contact the server administrator in order to obtain the CA certificate to be used.

Note: if you use a CA certificate that is expired, the Telit's module (client) detects the certificate expiration when it tries to perform the connection. An error message is displayed. To emulate the behaviour of Telit's modules that ignore this check, it is necessary to disable automatic date/time updating using the AT#NITZ AT command and set current date before expiry data using AT#CCLK AT command.

3.4. SSL Socket Setting

Before opening the SSL socket, several parameters can be configured via the following command:

AT#SSLCFG=<SSId>,<cid>,<pktSize>,<maxTo>,<defTo>,<txTo>

Where:

<SSId>: must be set to 1. It is the only Secure Socket ID available

<cid>: is the PDP Context Identifier, its value must be set to 1, see chapter 2.1.1.

<pktSize>: is the 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 <pktSize> values introduce a higher communication overhead.

<maxTo>: is the socket inactivity timeout. In online mode: if there's no data exchange within this timeout period the connection is closed**.** Increment it if it is needed a longer idle time period.

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

<txTo>: is the time period after which data is sent even if **<pktSize>** 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.

3.5. Examples

The next three chapters show examples concerning the configuration of the different authentication modes and the relating certificates storage. Each Telit's modules families have dedicated examples.

3.5.1. SSL Verify None Mode

This example is valid for all module families. The next command shows the SSL Verify None mode setting. In this case no security data have to be stored in NVM, the module is ready for SSL socket dial.

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

3.5.2. Server Authentication Mode

In this example it is assumed to use modules belonging to the GC, GE/GL, or GT families, and set up the TLS_RSA_WITH_RC4_128_MD5 cipher suite.

```
AT#SSLSECCFG=1,1,1
OK
```

```
AT#SSLSECDATA=1,1,1,<size>                ← store CA Certificate in PEM format
> -----BEGIN CERTIFICATE-----<LF>
[…]
-----END CERTIFICATE-----<LF>
<ctrl>Z
OK
```

Now the module is ready for SSL socket dial.

In the next examples it is assumed to use modules belonging to the HE910/UE910 families:

-- Example using DER format --

```
AT#SSLSECCFG=1,0,1,0                ← <cert_format> = 0, DER format is selected
OK
```

Note: if you don’t specify any <cert_format> with #SSLSECCFG, PEM is assumed as default. But read command doesn’t show any value due to retro compatibility with other platforms”

```
AT#SSLSECDATA=1,1,1,<size>          ← store CA Certificate in DER format
> …………………….                 ← when <size> bytes are entered, the Certificate is stored and OK
                                      message is displayed
OK
```

Now the module is ready for SSL socket dial.

-- Example using PEM format --

```
AT#SSLSECCFG=1,0,1,1                ← <cert_format> = 1, PEM format is selected
OK
```

```
AT#SSLSECDATA=1,1,1,<size>          ← store CA Certificate in PEM format
> -----BEGIN CERTIFICATE-----<LF>
[…]
-----END CERTIFICATE-----<LF>
<ctrl>Z
OK
```

Now the module is ready for SSL socket dial.

3.5.3. Server/Client Authentication Mode

This example is valid for all module families. Using the server/client authentication mode you need to store all the security data: Certificate, CA Certificate and private key.

For modules belonging to the GC, GE/GL, and GT families certificates have to be in PEM format, for HE910/UE910 families both PEM and DER format are supported.

```
AT#SSLSECCFG=1,0,2                 ← set Server/Client authentication mode
OK
```

Store all needed security data as follows:

```
AT#SSLSECDATA=1,1,0,<size>                 ← store Certificate
> -----BEGIN CERTIFICATE-----<LF>
[…]
-----END CERTIFICATE-----<LF>
<ctrl>Z
OK

AT#SSLSECDATA=1,1,1,<size>                 ← store CA Certificate
> -----BEGIN CERTIFICATE-----<LF>
[…]
-----END CERTIFICATE-----<LF>
<ctrl>Z
OK

AT#SSLSECDATA=1,1,2,<size>                 ← store RSA private key
[… private key …]
<ctrl>Z
OK
```

Now the module is ready for SSL socket dial.

4. Working with SSL Socket

This section describes the command that allow the user to open an SSL socket and exchange data using it. There are two data exchange modes:

- **online mode**
- **command mode**

After the AT commands introduction, some examples are illustrated.

Warning: the Telit's GC, GE/GL, and GT modules families have a maximum SSL record size = 15800 bytes. This limits to 15800[4] ***bytes the maximum size of a web page downloadable from a HTTPS server (except when the server itself divides the page in smaller chunks).***

4.1. Open a Secure Socket

An SSL socket can be opened using the following command:

AT#SSLD=<SSId>,<remotePort>,<remoteHost>,<closureType>,<mode>[,<timeout>]

Where:

<SSId>: must be set to 1. It is the only Secure Socket ID available

<remotePort>: is the remote port of the SSL server (usually 443)

<remoteHost>: is the IP/hostname of the SSL server.

<closureType>: enable/disable the capability to restore later the session without repeating the handshake phase using the #SSLFASTD command**.** This parameter can be set only to 0 for the modules belonging to the HE910/UE910 families.

<mode>: is the data exchange mode:

0 = **online mode**: the response message **CONNECT** is shown on success and from that moment all bytes sent to the serial port are treated as data.

1 = **command mode:** the response message **OK** is shown on success. After that, AT interpreter is still alive, and data can be exchanged by means of AT#SSLSEND and AT#SSLRECV commands.

[4] This restriction is not valid for GE910 family

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

<timeout>: is the timeout of the TCP socket on which the handshake message will be sent.

Refer to the following table to see how to manage the timeout expiration:

Software Versions[5]	
10.00.xxx, and 16.00.xxx	12.00.xxx, and 13.00.xxx
Modules equipped with this software version provide the capability to improve the CPU clock - in order to manage large certificates and avoid timeout expiration - by means of the #CPUMODE=2 or 4 command.	Modules equipped with these SW versions provide the capability to manage large certificates without the use of #CPUMODE. Therefore the command is not supported.

4.2. Exchange Data through a Secure Socket

4.2.1. Online Mode

After the **CONNECT** message, the user can send data to the module. Data is encrypted and sent to the server through the secure socket as soon as the packet size has been reached or the txTo timeout expires, see chapter 3.4 to configure these parameters.

In **online mode** the user cannot run AT commands on the used serial port or virtual port, refer to [4] or [5] to have detailed information about the serial/virtual ports. Anyway, it is possible suspend the connection (without closing it) by sending the escape sequence (+++). After that, the module returns the **OK** response and is able to parse AT commands again.

Data mode can be restored at any time by sending the following command:

AT#SSLO=<SSId>

Where:

<SSId>: must be set to 1. It is the only Secure Socket ID available.

After the #SSLO restore command, the **CONNECT** message is displayed, and SSL communication can continue.

If the idle inactivity timeout expires (<maxTo>, see paragraph chapter 3.4) or the remote server forces the connection closure, the **NO CARRIER** message is displayed.

[5] See the Applicability Table

4.2.2. Command Mode

In **command mode** data can be exchanged through an SSL socket by means of the AT#SSLSEND and AT#SSLRECV commands. The data exchange is performed in blocking mode.

Note: at any moment the user can switch to online mode by sending the AT#SSLO command (described in the previous paragraph).

4.2.2.1. Send Data

Use the following command to send data:

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

Where:

<SSId>: must be set to 1. It is the only Secure Socket ID available.

<Timeout>: is the maximum blocking timeout. It can be omitted, and in this case the default timeout set via AT#SSLCFG will be used (<defTo>, refer to chapter 3.4).

After the '>' prompt the data to be sent can be entered. To complete the operation send <ctrl>Z, then the data is forwarded through the secure socket.

Response:

OK on success

ERROR on failure

4.2.2.2. Receive Data

Use the following command to receive data:

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

Where:

<SSId>: must be set to 1. It is the only Secure Socket ID available.

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

<Timeout>: is the maximum blocking timeout. It can be omitted, and in this case the default timeout set via AT#SSLCFG will be used (<defTo>, refer to chapter 3.4).

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

#SSLRECV: <numBytesRead>
… received data ….
OK

Where:

<numBytesRead> is the number of bytes actually read (equal or less than **<MaxNumBytes>**.

If the timeout expires, the module displays the following response

#SSLRECV: 0
TIMEOUT
OK

The **ERROR** message is displayed on failure.

4.3. Close a Secure Socket

The SSL socket can be closed by means of the following command:

AT#SSLH=<SSId>,<closureType>

Where:

<SSId>: must be set to 1. It is the only Secure Socket ID available

<closureType>: enable/disable the capability to restore later the session without repeating the handshake phase using the #SSLFASTD command**.** This parameter can be set only to 0 for the modules belonging to the HE910/UE910 families.

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

If the socket has been opened in **command mode**, when communication is remotely closed and all data have been retrieved (#SSLRECV), you can close on client side also and NO CARRIER message is displayed.

At any moment, it is also possible to close the socket on client side by means of AT#SSLH.

4.4. Perform a Fast Dial

The restore of a previous session avoids full handshake and performs a fast dial, which saves time and reduces the TCP payload.

The restoring is accomplished by means of the AT#SSLFASTD command, which can be used if #SSLD or #SSLH command have been entered with <closureType> parameter set to =1, refer to chapters 4.1 and 4.3 respectively. In this case, the previous data security is not deleted on socket closure.

⚠ ***Warning: this command is not provided by the HE910/UE910 families***

The syntax of the command is:

AT#SSLFASTD=<SSId>,<connMode>,<Timeout>

Where:

<SSId>: must be set to 1. It is the only Secure Socket ID available

<connMode>: is the data exchange mode (0 = **online mode**, 1 = **command mode**).

<Timeout>: is the maximum blocking timeout. It can be omitted, and in this case the default timeout set via AT#SSLCFG will be used (<defTo>, refer to chapter 3.4).

4.5. Examples

In the next sub-sections are described examples concerning the AT commands introduced in the previous chapters.

4.5.1. Online Mode Communication

In the following example it is assumed that the socket is opened and connected to the IP 123.124.125.126, where there’s an SSL server listening at port 443. After data exchange the connection is suspended, the AT#SSLS command is entered to check the SSL status, and then the data mode is restored using AT#SSLO command. At the end the SSL socket is closed. Moreover, suppose that the PDP context activation, SSL socket enabling, and SSL socket security configuration are already performed.

Modules belonging to the GC, GE/GL, GT, and HE910/UE910 families:

```
AT#SSLD=1,443,”123.124.125.126”,0,0           ← Open the SSL socket in on line mode
CONNECT
..
[bidirectional data exchange]
..
[send +++]                 ← 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]
..
[send +++]             ← 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
```

4.5.2. Command Mode Communication

In the following example it is assumed that the socket is opened and connected to the IP 123.124.125.126, where there's an SSL server listening at port 443. The data exchange is performed via #SSLSEND and #SSLRECV commands. At the end the SSL socket is closed. Moreover, suppose that the PDP context activation, SSL socket enabling, and SSL socket security configuration are already performed.

Modules belonging to the GC, GE/GL, GT, and HE910/UE910 families:

```
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 at the end of its data sending and no more data are available to be retrieved, communication is closed on client side also. NO CARRIER message is displayed, and then no #SSLH is needed.

4.5.3. Sending/Receiving Data in Command Mode

In the following example it is assumed that the socket is opened and connected to the IP 123.124.125.126, where there’s an SSL server listening at port 443. After data exchange in online mode the connection is suspended and is entered the command mode. In the command mode the AT interface is active and by means of the #SSLSEND and #SSLRECV commands it is possible to continue to receive and send data using the SSL socket that is still connected. At the end the SSL socket is closed. Moreover, suppose that the PDP context activation, SSL socket enabling, and SSL socket security configuration are already performed.

Modules belonging to the GC, GE/GL, GT, and HE910/UE910 families:

```
AT#SSLD=1,443,"123.124.125.126",0,0        ← Open the SSL socket in Online Mode
CONNECT
..
[bidirectional data exchange]
..
[send +++]                  ← Suspend the connection and enter into 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 remote server closes data communication at the end of its data sending and no more data are available to be retrieved, communication is closed on client side also. NO CARRIER message is displayed, and then no #SSLH is needed.

4.5.4. Open an SSL Socket and Restore it later

In the following example it is assumed that the socket is opened and connected to the IP 123.124.125.126, where there's an SSL server listening at port 443; in addition suppose that the <closureType> parameter is set to 1. Data exchange is performed in **online mode**, and then the socket is closed and reopened via #SSLFASTD command. After a new data exchange the socket is closed definitively. Moreover, suppose that the PDP context activation, SSL socket enabling, and SSL socket security configuration are already performed.

Modules belonging to the GC, GE/GL, and GT families:

```
AT#SSLD=1,443,"123.124.125.126",1,0         ← Open the SSL socket in Online Mode
CONNECT
..
[bidirectional data exchange]
..
[send +++]                   ← Suspend the connection and enter into Command Mode
OK
```

```
AT#SSLH=1                    ← Close SSL socket
OK
```

Warning: HE910/UE910 families do not support #SSLFASTD command.

```
AT#SSLFASTD=1,0      ← Restore the session in Online Mode
..
[bidirectional data exchange]
..
[send +++]                   ← Suspend the connection
OK
```

```
AT#SSLH=1,0                  ← Force definitive closure
OK
```

4.5.5. Connect to an HTTPS Server

Referring to the example illustrated in chapter 3.3.1: it is assumed that you have got the CA Certificate. Moreover, suppose that the PDP context activation and SSL socket enabling are already performed.

The following SSL commands perform: the configuration of the SSL socket in server authentication mode, the storing of the CA certificate, the opening of the socket and starting of the data exchange. After that, the HTTPS server responds to the module and closes the socket.

```
AT#SSLSECCFG=1,0,1           ← Set Server Authentication Mode
OK
```

```
AT#SSLSECDATA=1,1,1,1760          ← Store the CA Certificate
> -----BEGIN CERTIFICATE-----
………..
Write the certificate got by using the procedure describe in chapter 3.3.1
……….
-----END CERTIFICATE-----
<ctrl>Z
OK


AT#SSLD=1,443,”www.---“,0,0        ← Open the SSL socket in Online Mode
CONNECT
……
The module receives from the HTTPS server a response
……
NO CARRIER                       ← Server remote closure: some servers are configured in order to close
                                 the socket after a single request.
```

5. FTP with TLS

FTPS is useful when an application needs to connect securely using FTP. As described in RFC 4217, FTPS permits authentication, integrity and confidentiality during an FTP connection over a SSL/TLS secure socket. The Telit’s modules support the explicit mode, as described in RFC 4217, in this mode the FTPS client must explicitly request security from an FTPS server (implicit mode is a deprecated). When FTPS connection is opened towards an FTPS server, FTP command AUTH (refer to RFC2228, and RFC4217) is sent to the server to explicitly request a secure FTP connection.

To enable an FTPS connection it is necessary to follow the steps below:

- Use the AT#FTPCFG command to enable FTPS security.
- Use AT#SSLSECCFG and AT #SSLSECDATA commands to configure the SSL, see chapters 3.2 and 3.3.

As usual, use the FTP commands to open control connection and data connection. When #FTPOPEN is used, 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. Server shall be configured so that TLS reuse is not required.

The same certificates saved via #SSLSECDATA command are used for both TLS sessions, as strongly recommended by RFC 4217.

Example for modules belonging to the GC, GE/GL, GT, and HE910/UE910 families

```
AT#FTPCFG=<tout>,<IPPignoring>,1        ← Enable the FTPS security
OK


AT#SSLSECCFG=1,0,1                      ← Set Server Authentication Mode
OK


AT#SSLSECDATA=1,1,1,1159                ← store CA Certificate[6]
> -----BEGIN CERTIFICATE-----
[…]
   -----END CERTIFICATE-----
<ctrl>Z
OK
```

[6] GC, GE/GL, and GT families support PEM format;
HE910/UE910 families support PEM and DER formats.

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

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

Now, FTP control connection is secured via TLS protocol

```
AT#FTPGET=”file.txt”          ← Get the file from the FTP server
CONNECT
```

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

```
…….
…….
…….
NO CARRIER
```

```
AT#FTPCLOSE                   ← Close the FTPS connection
OK
```

In this example server authentication and cipher suite 0 have been considered. See chapters 3.2, and 3.3 for other available configurations.

Note: GC, GE/GL, and GT families support PEM format. HE910/UE910 families support PEM and DER formats.

6. SSL Error Codes

Telit’s modules provide the AT+CMEE command to enable/disable the error report. The error report can assume 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

7. Document History

Revision	Date	Product/ SW Version	Changes
0	2011-10-11	/	First issue
1	/		/
2	2012-11-07		Added GE910 module and HE910 family modules. The document has been updated in accordance with the added modules.
3	2012-12-14		Added notes in chapters 3.2, and 3.3.1
4	2013-03-15		Modified figures in chapter 3.3.1. Added note in chapter 3.3. Added explanation for HE910 of: 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.
5	2013-05-02		Update for HE910: client authentication support, FTP over TLS support. Enhancements regarding FTP over TLS for all families.
6	2013-09-13	GE910-GNSS/13.00.xx4 GL865-QUAD V3/16.00xx3 GE910-QUAD V3/16.00.xx3 UE910/12.00.004	In the Applicability Table have been added the products shown on the left side: first issue.

8. Abbreviation and acronyms

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
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