gemalto
security to be free

Transport Layer Security for Client TCP/IP Services

Application Note 62

Version:	04
DocId:	wm01_wm02_an62_tls_v04

Application Note 62:	**Transport Layer Security for Client TCP/IP Services**
Version:	**04**
Date:	**2013-10-15**
DocId:	**wm01_wm02_an62_tls_v04**
Status	**Confidential / Released**

GENERAL NOTE
THE USE OF THE PRODUCT INCLUDING THE SOFTWARE AND DOCUMENTATION (THE "PRODUCT") IS SUBJECT TO THE RELEASE NOTE PROVIDED TOGETHER WITH PRODUCT. IN ANY EVENT THE PROVISIONS OF THE RELEASE NOTE SHALL PREVAIL. THIS DOCUMENT CONTAINS INFORMATION ON GEMALTO M2M PRODUCTS. THE SPECIFICATIONS IN THIS DOCUMENT ARE SUBJECT TO CHANGE AT GEMALTO M2M'S DISCRETION. GEMALTO M2M GMBH GRANTS A NON-EXCLUSIVE RIGHT TO USE THE PRODUCT. THE RECIPIENT SHALL NOT TRANSFER, COPY, MODIFY, TRANSLATE, REVERSE ENGINEER, CREATE DERIVATIVE WORKS; DISASSEMBLE OR DECOMPILE THE PRODUCT OR OTHERWISE USE THE PRODUCT EXCEPT AS SPECIFICALLY AUTHORIZED. THE PRODUCT AND THIS DOCUMENT ARE PROVIDED ON AN "AS IS" BASIS ONLY AND MAY CONTAIN DEFICIENCIES OR INADEQUACIES. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, GEMALTO M2M GMBH DISCLAIMS ALL WARRANTIES AND LIABILITIES. THE RECIPIENT UNDERTAKES FOR AN UNLIMITED PERIOD OF TIME TO OBSERVE SECRECY REGARDING ANY INFORMATION AND DATA PROVIDED TO HIM IN THE CONTEXT OF THE DELIVERY OF THE PRODUCT. THIS GENERAL NOTE SHALL BE GOVERNED AND CONSTRUED ACCORDING TO GERMAN LAW.

Copyright

Trademark Notice

Contents

0 Document History

Preceding document: "AN62: Transport Layer Security for Client TCP/IP Services", v03
New document: "AN62: Transport Layer Security for Client TCP/IP Services", v**04**

Chapter	What is new
6	Secure commands data structures added

Preceding document: "AN62: Transport Layer Security for Client TCP/IP Services", v02
New document: "AN62: Transport Layer Security for Client TCP/IP Services", v03

Chapter	What is new
3.2.1	Attached files containing tools to manage certificates and keys.

Preceding document: "AN62: Transport Layer Security for Client TCP/IP Services", v01
New document: "AN62: Transport Layer Security for Client TCP/IP Services", v02

Chapter	What is new
1.1	Updated list of supported modules.

New document: "AN62: Transport Layer Security for Client TCP/IP Services", v01

Chapter	What is new
--	Initial document setup.

1 Introduction

The embedded TCP/IP stack of Cinterion® wireless modules supports Transport Layer Security (TLS) for the services

- "TCP Socket Client"
- "Transparent TCP Client"
- "HTTP Client".

The concept of TLS for Cinterion® wireless modules is targeted on industrial applications capable of secure communication. The Cinterion® wireless module acts as TLS/SSL client and supports server authentication. Server authentication can be done in two scenarios:

- Server and module have identical root certificate.
- Server certificate forms a chain with the certificate stored in the module.

All required certificates and keys can be stored in the module's NVRAM. The major benefit is that managing certificates and keys can be part of the process of manufacturing mobile applications.

Cinterion® wireless modules can be configured to establish secure TCP/IP connections at different security levels:

- TLS with server authentication requires a certificate infrastructure comprising up to 10 server certificates and one dedicated public client certificate that protects access to the (server) certificate store in the module's NVRAM. All certificates shall be coded in DER format. The public client certificate is also referred to as *local client certificate*.
  Guidelines for loading certificates into the module's NVRAM and setting up secure TCP/IP connections based on a combination of symmetric and asymmetric encryption can be found in Chapter 3.
- TLS with deactivated server authentication requires no certificates to be stored, but implies only a low level of security. Guidelines can be found in Chapter 4.

Basic information about generating certificates and key stores can be found in Chapter 5.

To save time and effort on generating and handling certificates you can take advantage of ready-to-use Java tools tested by Gemalto M2M and supplied along with the firmware packages. For details see particularly Sections 2.1, 3.2, 5.1.

Assuming that you are familiar with all AT commands and procedures controlling the embedded TCP/IP stack, the focus of this document is only on TLS specific aspects. An understanding of the basic concepts of TLS and certificate management is assumed as well.

1.1 Supported Products

The following Cinterion® wireless modules are supported:

WM01 modules: MC75i, EES3, TC63i, TC65i, TC65i-X, BGS3, EGS3, EGS5, EGS5-X
WM02 modules: BGS2-E/BGS2-W (as of Release 2), AGS2-W

1.2 Related Documents

[1] AT Command Set for your Cinterion® wireless module

1.3 Abbreviations

Table 1: Abbreviations

Abbreviation	Meaning
AES	Advanced Encryption Standard
CA	Certificate Authority
CBC	Cipher Block Chaining
DER	Distinguished Encoding Rules
DES	Data Encryption Standard
EDE	Encrypt-Decrypt-Encrypt
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
MD5	Message Digest algorithm 5
ME	Mobile Equipment
NVRAM	Non-Volatile Random Access Memory
OS	Operating System
PKCS	Public Key Cryptography Standard
RSA	Rivest, Shamir, Adleman algorithm for public-key cryptography
SHA	Secure Hash Algorithm
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TE	Terminal Equipment
TLS	Transport Layer Security
URC	Unsolicited Result Code

2 Overview of Procedures

This chapter summarizes the procedures required to prepare Cinterion® wireless modules for TLS and set up secure TCP/IP connections.

2.1 Procedures to Manage Certificates in Module's NVRAM

AT commands and tools for controlling the module's certificate storage:

- **AT^SBNW** command or **"cmd_ipCertMgr.jar"** tool (see Section 3.2): Load (overwrite, remove) existing certificates coded in DER format into the module's NVRAM.
  The tool handles all necessary AT^SBNW procedures eliminating the need for application manufacturers to use AT^SBNW. Please contact Gemalto M2M if you require more information on how to handle certificates using the AT command AT^SBNW (Data structures are listed in Chapter 6).
- **AT^SBNR**: Read the certificates stored in module's NVRAM.
  AT^SBNR procedure can also be done by using "cmd_ipCertMgr.jar" (see Section 3.2).
- **AT+GSN**: Delivers the module's IMEI. The IMEI is required to access the module's NVRAM.

Note: If you wish to create your own public/private key pairs and associated certificates, e.g. for self-authentication, refer to Chapter 5.

2.2 Procedures to Set up Secure TCP/IP Connections

AT commands required to set up secure TCP/IP connections:

- **AT^SICS**: Create Internet Connection Profile(s) as usual.
- **AT^SISS**: Create Internet Service Profile(s) with AT^SISS and specify TLS specific settings:
  - Add optional parameter **"secOpt"** to enable secure connections. "secOpt" values:
    "": TLS is deactivated (default)
    "-1": TLS without server authentication
    "1...10": TLS with server authentication, where numbers 1...10 = server certificate stored at index 1...10. Certificate numbers can be separated by comma, e.g. 1,5,9.
  - For HTTP client set **HTTPS** within address.
- **AT^SIND**: Enable certificate indicator "+CIEV: is_cert" which reports server certificate details after opening a secure Internet connection with AT^SISO.
  AT+CMER: Necessary to switch on event reporting for indicator "+CIEV: is_cert"
- **AT^SISO / AT^SISC**: Open and close TCP/IP connections as usual.

2.3 List of Related AT Command Chapters in [1]

- "Internet Service Commands"
- "AT^SBNW Binary Write", AT^SBNR Binary Read"
- "AT^SIND Extended Indicator Control", "AT+CMER Common Event Reporting"
- "AT+GSN Request International Mobile Equipment Identity (IMEI)"

3 Secure TCP/IP Connections with Server Authentication

All certificates stored in the module’s NVRAM must be coded in DER format. Maximum size of one certificate is 2kB. Any certificates coded in different formats (e.g. PEM) must be converted to DER format before loading them into the module’s NVRAM. Conversion from PEM to DER is described in Chapter 5.

Server certificates shall be stored to the module’s NVRAM at indexes from 1 to 10. The local client certificate required to protect the certificate store shall be located at index 0.

Preconditions

- The local client certificate at index 0 must be stored before any server certificates can be loaded at indexes 1 - 10.
- Loading, reading or deleting a certificate requires the module’s IMEI be given. To get the IMEI you can use the command AT+GSN.
- When using one of the Java tools take care that the interface is free (not used by application or Terminal program).

3.1 Supported SSL/TLS Cipher Suites

The embedded TCP/IP stack supports the cipher suites listed in Table 2.

Table 2: Supported cipher suites

Cipher Suite	RFC	SSL/TLS version Comments
TLS_RSA_WITH_NULL_MD5	RFC 2246	TLS 1.0 (and SSL 3.0, RFC 6101)
TLS_RSA_WITH_NULL_SHA	RFC 2246	TLS 1.0 (and SSL 3.0, RFC 6101)
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RFC 4346	mandatory in TLS v1.1
TLS_RSA_WITH_AES_128_CBC_SHA	RFC 3268	mandatory in TLS v1.2 (RFC 5246)

3.2 "cmd_ipCertMgr.jar" Java Tool

"cmd_ipCertMgr.jar" is a Java command line tool designed to load certificates into the module and to read or delete the loaded certificates. For these procedures the tool executes the commands AT^SBNW and AT^SBNR using the arguments shown in Table 3.

3.2.1 System Requirements for "cmd_ipCertMgr.jar"

The "cmd_ipCertMgr.jar" tool is written in Java in order to be OS independent. To run the application at least 5MB disc space is required and Java Runtime Environment JRE6 is recommended. The Java RXTX library delivered along with the "cmd_ipCertMgr.jar" tool needs to be copied to the same directory as the "cmd_ipCertMgr.jar" tool.

The "cmd_ipCertMgr.jar" tool together with a Windows and Linux OS version of the required Java RXTX library is attached as a zipped file to this PDF, depending on your product. Please open the Attachments navigation panel to save and then unpack the appropriate zipped file:

For **WM01 modules**, i.e., for MC75i, EES3, TC63i, TC65i, TC65i-X, BGS3, EGS3, EGS5 and EGS5-X, save and unpack the "wm01_tls_tools.7z" file. For **WM02 modules**, i.e., for BGS2-E/ BGS2-W (as of Release 2) and AGS2-W, save and unpack the "wm02_tls_tools.7z" file.

3.2.2 Input Arguments for "cmd_ipCertMgr.jar" Java Tool

Table 3: Input arguments for "cmd_ipCertMgr.jar"

Argument	Accepted values	Description	Occurrence	Examples used below
`-h or -help`		Display help message with all input arguments	Optional	
`-serialPort`	String	ID of the serial port. If not specified default values are assumed: **COM1** for Windows, **/dev/ttyS0** for Linux	Optional	COM3
`-serialSpd`	Integer value	Baud rate of serial interface set with AT+IPR. Default: 115200.	Optional	115200
`-cmd`	`WriteCert`	Write local client/server certificate on module	Mandatory	
	`ReadCert`	Display local client/ server certificate data	Mandatory	
	`DelCert`	Delete local client/server certificate from module	Mandatory	
`-certfile`	String	Specifies filename of certificate. Required for set commands.	Mandatory for **WriteCert** command	client.der server1.der server2.der ...
`-certindex`	Integer value within range 0-10	Index for certificate 0: local client certificate 1-10: server certificates	Mandatory	
`-imei`	15 decimal digit	IMEI number of the module	Mandatory	004401080551530
`-alias`	String	Name of entry in keystore containing local client private key.	Mandatory	client01
`-keypass`	String	Password for local client private key	Mandatory	pwdclient
`-keystore`	String	Name of keystore file	Mandatory	client.ks
`-storepass`	String	Password for keystore	Mandatory	pwdclient

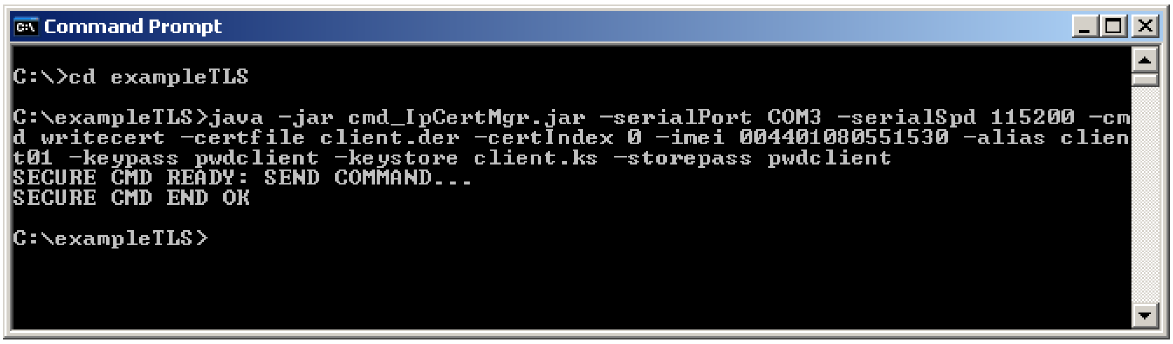
See Chapter 5 for an example of how to create certificate and keystore files required for the Java tool.

3.3 Loading Certificates to NVRAM with "cmd_IpCertMgr.jar"

The "cmd_ipCertMgr.jar" tool can be used to load local client and server certificates into the module’s NVRAM.

Loading local client certificate at index 0:

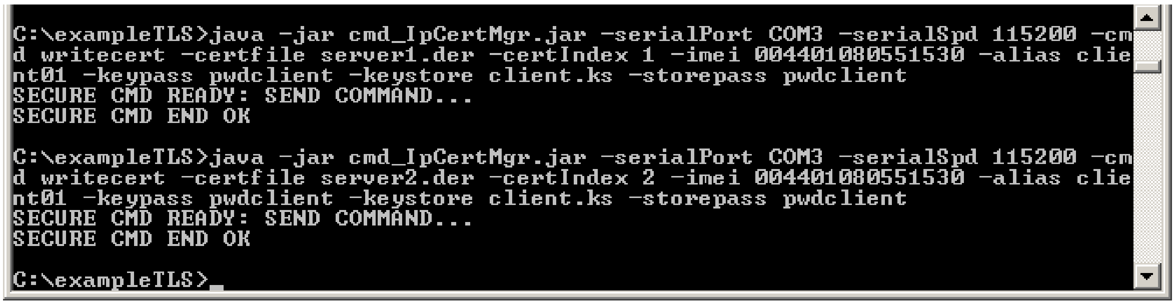
```
java -jar cmd_IpCertMgr.jar -serialPort COM3 -serialSpd 115200\
-cmd writecert -certfile client.der -certIndex 0 -imei 004401080551530\
-alias client01 -keypass pwdclient -keystore client.ks\
-storepass pwdclient
```



```
C:\>cd exampleTLS

C:\exampleTLS>java -jar cmd_IpCertMgr.jar -serialPort COM3 -serialSpd 115200 -cm
d writecert -certfile client.der -certIndex 0 -imei 004401080551530 -alias clien
t01 -keypass pwdclient -keystore client.ks -storepass pwdclient
SECURE CMD READY: SEND COMMAND...
SECURE CMD END OK

C:\exampleTLS>
```

Loading server certificate at index 1 and 2:

```
java -jar cmd_IpCertMgr.jar -serialPort COM3 -serialSpd 115200\
-cmd writecert -certfile server1.der -certIndex 1 -imei 004401080551530\
-alias client01 -keypass pwdclient -keystore client.ks\
-storepass pwdclient
java -jar cmd_IpCertMgr.jar -serialPort COM3 -serialSpd 115200\
-cmd writecert -certfile server2.der -certIndex 2 -imei 004401080551530\
-alias client01 -keypass pwdclient -keystore client.ks\
-storepass pwdclient
```



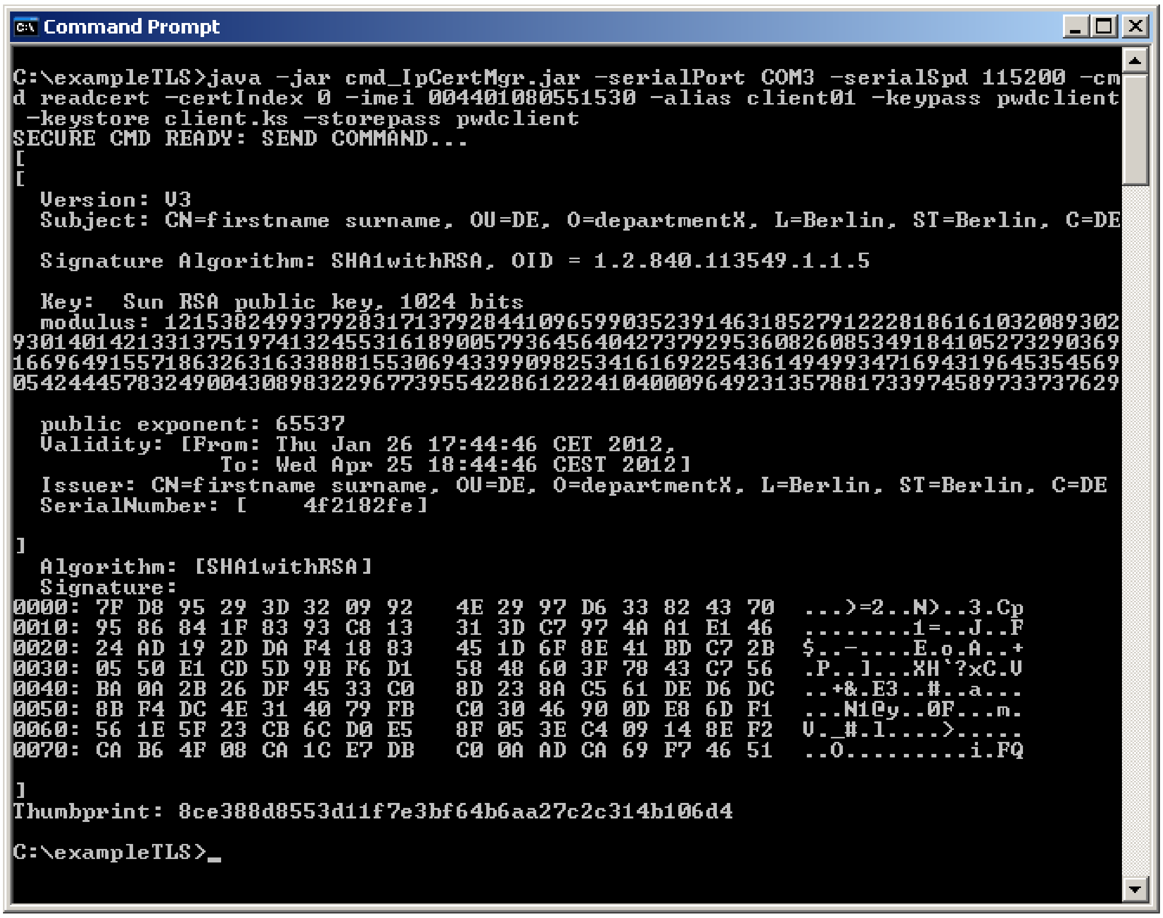
```
C:\exampleTLS>java -jar cmd_IpCertMgr.jar -serialPort COM3 -serialSpd 115200 -cm
d writecert -certfile server1.der -certIndex 1 -imei 004401080551530 -alias clie
nt01 -keypass pwdclient -keystore client.ks -storepass pwdclient
SECURE CMD READY: SEND COMMAND...
SECURE CMD END OK

C:\exampleTLS>java -jar cmd_IpCertMgr.jar -serialPort COM3 -serialSpd 115200 -cm
d writecert -certfile server2.der -certIndex 2 -imei 004401080551530 -alias clie
nt01 -keypass pwdclient -keystore client.ks -storepass pwdclient
SECURE CMD READY: SEND COMMAND...
SECURE CMD END OK

C:\exampleTLS>_
```

3.4 Reading Stored Certificates with "cmd_IpCertMgr.jar"

The "cmd_ipCertMgr.jar" tool can be used to read certificates stored in the NVRAM.

Reading certificate stored at index 0:

```
java -jar cmd_IpCertMgr.jar -serialPort COM3 -serialSpd 115200\
-cmd readcert -certIndex 0 -imei 004401080551530 -alias client01\
-keypass pwdclient -keystore client.ks -storepass pwdclient
```



```
C:\exampleTLS>java -jar cmd_IpCertMgr.jar -serialPort COM3 -serialSpd 115200 -cm
d readcert -certIndex 0 -imei 004401080551530 -alias client01 -keypass pwdclient
 -keystore client.ks -storepass pwdclient
SECURE CMD READY: SEND COMMAND...
[
[
  Version: V3
  Subject: CN=firstname surname, OU=DE, O=departmentX, L=Berlin, ST=Berlin, C=DE

  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
  modulus: 121538249937928317137928441096599035239146318527912228186161032089302
93014014213313751974132455316189005793645640427379295360826085349184105273290369
16696491557186326316338881553069433990982534161692254361494993471694319645354569
05424445783249004308983229677395542286122241040009649231357881733974589733737629

  public exponent: 65537
  Validity: [From: Thu Jan 26 17:44:46 CET 2012,
               To: Wed Apr 25 18:44:46 CEST 2012]
  Issuer: CN=firstname surname, OU=DE, O=departmentX, L=Berlin, ST=Berlin, C=DE
  SerialNumber: [    4f2182fe]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 7F D8 95 29 3D 32 09 92   4E 29 97 D6 33 82 43 70  ...)=2..N)..3.Cp
0010: 95 86 84 1F 83 93 C8 13   31 3D C7 97 4A A1 E1 46  ........1=..J..F
0020: 24 AD 19 2D DA F4 18 83   45 1D 6F 8E 41 BD C7 2B  $..-....E.o.A..+
0030: 05 50 E1 CD 5D 9B F6 D1   58 48 60 3F 78 43 C7 56  .P..]...XH`?xC.V
0040: BA 0A 2B 26 DF 45 33 C0   8D 23 8A C5 61 DE D6 DC  ..+&.E3..#..a...
0050: 8B F4 DC 4E 31 40 79 FB   C0 30 46 90 0D E8 6D F1  ...N1@y..0F...m.
0060: 56 1E 5F 23 CB 6C D0 E5   8F 05 3E C4 09 14 8E F2  V._#.l....>.....
0070: CA B6 4F 08 CA 1C E7 DB   C0 0A AD CA 69 F7 46 51  ..O.........i.FQ

]
Thumbprint: 8ce388d8553d11f7e3bf64b6aa27c2c314b106d4

C:\exampleTLS>_
```

3.5 Reading Stored Certificates Stored with AT^SBNR

The AT^SBNR=is_cert command prints a list of all certificates located at indexes 0 - 10 inside the module’s NVRAM and displays the content of stored certificates.

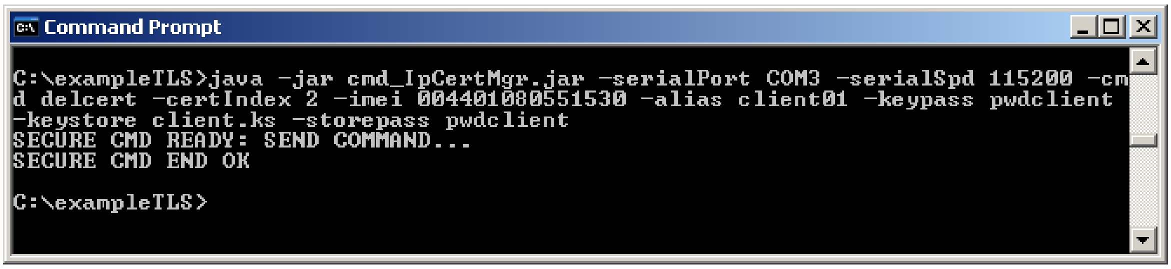
```
AT^SBNR=is_cert
^SBNR: 0, size: "599", issuer: "/C=DE/ST=Berlin/L=Berlin/O=departmentX/
OU=DE/CN=firstname surname", serial number: "4F2182FE", subject: "/C=DE/
ST=Berlin/L=Berlin/O=departmentX/OU=DE/CN=firstname surname", signa-
ture: "sha1RSA", thumbprint algorithm: "sha1", thumbprint:
"8CE388D8553D11F7E3BF64B6AA27C2C314B106D4"
^SBNR: 1, size: "583", issuer: "/C=DE/ST=Berlin/L=Berlin/O=orgOne/
OU=unitOne/CN=myServer1", serial number: "4F229DB4", subject: "/C=DE/
ST=Berlin/L=Berlin/O=orgOne/OU=unitOne/CN=myServer1", signature:
"sha1RSA", thumbprint algorithm: "sha1",
 thumbprint: "E748E89F094D21323C250A3DF2B455AADA58A337"
^SBNR: 2, size: "583", issuer: "/C=DE/ST=Berlin/L=Berlin/O=orgTwo/
OU=unitTwo/CN=myServer2", serial number: "4F229ED8", subject: "/C=DE/
ST=Berlin/L=Berlin/O=orgTwo/OU=unitTwo/CN=myServer2", signature:
"sha1RSA", thumbprint algorithm: "sha1",
 thumbprint: "D8EF19F5124B78CB2C07DC63BAB4B8E321998098"
^SBNR: 3, size: "0", issuer: "", serial number: "", subject: "", signa-
ture: "",
thumbprint algorithm: "", thumbprint: ""
^SBNR: 4, size: "0", issuer: "", serial number: "", subject: "", signa-
ture: "",
thumbprint algorithm: "", thumbprint: ""
^SBNR: 5, size: "0", issuer: "", serial number: "", subject: "", signa-
ture: "",
thumbprint algorithm: "", thumbprint: ""
^SBNR: 6, size: "0", issuer: "", serial number: "", subject: "", signa-
ture: "",
thumbprint algorithm: "", thumbprint: ""
^SBNR: 7, size: "0", issuer: "", serial number: "", subject: "", signa-
ture: "",
thumbprint algorithm: "", thumbprint: ""
^SBNR: 8, size: "0", issuer: "", serial number: "", subject: "", signa-
ture: "",
thumbprint algorithm: "", thumbprint: ""
^SBNR: 9, size: "0", issuer: "", serial number: "", subject: "", signa-
ture: "",
thumbprint algorithm: "", thumbprint: ""
^SBNR: 10, size: "0", issuer: "", serial number: "", subject: "", signa-
ture: "",
 thumbprint algorithm: "", thumbprint: ""

OK
```

3.6 Deleting Certificates with "cmd_IpCertMgr.jar"

The "cmd_ipCertMgr.jar" tool can be used to delete certificates stored in the NVRAM.

Deleting server certificate stored at index 2:

```
java -jar cmd_IpCertMgr.jar -serialPort COM3 -serialSpd 115200\
-cmd delcert -certIndex 2 -imei 004401080551530 -alias client01\
-keypass pwdclient -keystore client.ks -storepass pwdclient
```



```
C:\exampleTLS>java -jar cmd_IpCertMgr.jar -serialPort COM3 -serialSpd 115200 -cm
d delcert -certIndex 2 -imei 004401080551530 -alias client01 -keypass pwdclient
-keystore client.ks -storepass pwdclient
SECURE CMD READY: SEND COMMAND...
SECURE CMD END OK

C:\exampleTLS>
```

3.7 Secure TCP/IP Connections with Server Authentication

Create Internet Connection Profiles with AT^SICS and Internet Service profiles with AT^SISS as usual and set the additional "secOpt" parameter. "secOpt" determines the server certificate(s) to be used for the connection. Certificate numbers can be separated by comma, e.g. 1,5,9. For "HTTP" client profiles, replace HTTP with HTTPS within the address field.

To verify whether or not the server sends a certificate activate the certificate indicator "+CIEV: is_cert" with AT^SIND.

When opening the connection with AT^SISO, a secure connection will be set up, i.e., all transferred data is encrypted, and the certificate received from the server will be checked.

Example: Secure connection set up by Transparent TCP Client

```
AT^SIND=is_cert,1                               Activate  certificate "+CIEV:"
^SIND: is_cert,1                                indicator.
OK
AT+CMER=3,0,0,2                                 Activate event reporting (neces-
OK                                              sary for "+CIEV:" indicators).

AT^SICS=0,conType,"GPRS0"                       Create Internet Connection Pro-
OK                                              file
at^sics=0,apn,internet.t-d1.de
OK

AT^SISS=2,srvType"Transparent"                  Create secure Internet Service
OK                                              Profile for Transparent TCP Cli-
AT^SISS=2,conId,0                               ent.
OK
AT^SISS=2,address,"192.168.1.60:443:timer=200"

OK
AT^SISS=2,secOpt,1                              Enable secure Internet connec-
OK                                              tion based on certifcate stored at
                                                index 1.

AT^SISO=2                                       Open Internet connection.
OK                                              Server certificate is shown.

+CIEV: is_cert,2,/C=DE/ST=Berlin/O=demo/OU=demo/CN=level2CA,00F1F456533
0802E0D,\ /C=DE/ST=Berlin/O=demo/OU=demo/CN=level3.exam-
ple.de,sha1RSA,sha1,6360CE34138D1F80818F25E58B55BF2FA486E7B9

^SISW: 2, 1                                     Service ready to read or write
^SISR: 2, 1                                     data.

AT^SISC=2                                       Close connection.
OK
```

Example: Secure connection with certificate chain set up by HTTP client:
The server’s certificate chain has been loaded into the module’s NVRAM.

Step 1: The example starts off reading the stored certificates:

```
AT^SBNR=is_cert
^SBNR: 0, size: "587", issuer: "/C=De/ST=De/L=Berlin/O=Cinterion/OU=Cin-
terion/CN=JohnPublic", serial number: "4EC116A4", subject: "/C=De/ST=De/
L=Berlin/O=Cinterion/OU=Cinterion/CN=JohnPublic", signature: "sha1RSA",
thumbprint algorithm: "sha1", thumbprint:
"4DF0D1F4046DCC742030173AEE2C79F710335E51"
^SBNR: 1, size: "1239", issuer: "/C=US/O=VeriSign, Inc./OU=VeriSign Trust
Network/OU=(c) 2006 VeriSign, Inc. - For authorized use only/CN=VeriSign
Class 3 Public Primary Certification Authority - G5", serial number:
"18DAD19E267DE8BB4A2158CDCC6B3B4A", subject: "/C=US/O=VeriSign, Inc./
OU=VeriSign Trust Network/OU=(c) 2006 VeriSign, Inc. - For authorized use
only/CN=VeriSign Class 3 Public Primary Certification Authority - G5",
signature: "sha1RSA"
, thumbprint algorithm: "sha1", thumbprint:
"4EB6D578499B1CCF5F581EAD56BE3D9B6744A5E5"
^SBNR: 2, size: "1570", issuer: "/C=US/O=VeriSign, Inc./OU=VeriSign Trust
Network/OU=(c) 2
006 VeriSign, Inc. - For authorized use only/CN=VeriSign Class 3 Public
Primary Certificat
ion Authority - G5", serial number: "2C48DD930DF5598EF93C99547A60ED43",
subject: "/C=US/O=
VeriSign, Inc./OU=VeriSign Trust Network/OU=Terms of use at https://
www.verisign.com/rpa (
c)06/CN=VeriSign Class 3 Extended Validation SSL SGC CA", signature:
"sha1RSA", thumbprint
 algorithm: "sha1", thumbprint:
"B18039899831F152614667CF23FFCEA2B0E73DAB"
^SBNR: 3, size: "1638", issuer: "/C=US/O=VeriSign, Inc./OU=VeriSign Trust
Network/OU=Terms
 of use at https://www.verisign.com/rpa (c)06/CN=VeriSign Class 3
Extended Validation SSL
SGC CA", serial number: "266C5BEC2C489C013B6FD32C17187215", subject: "/
C=DE/ST=Nordrhein-W
estfalen/L=Bonn/O=Servicegesellschaft der PSD Banken mbH/OU=Vertrieb/
OU=Terms of use at ww
w.verisign.com/rpa (c)05/CN=onlinebanking.psd-bank.de", signature:
"sha1RSA", thumbprint a
lgorithm: "sha1", thumbprint: "4CFF234A6DB89B3F72DB57643D14A6CB8FACD999"
^SBNR: 4, size: "1144", issuer: "/C=US/ST=UT/L=Salt Lake City/O=The USER-
TRUST Network/OU=h
ttp://www.usertrust.com/CN=UTN-USERFirst-Hardware", serial number:
"44BE0C8B500024B411D336
2AFE650AFD", subject: "/C=US/ST=UT/L=Salt Lake City/O=The USERTRUST Net-
work/OU=http://www.
usertrust.com/CN=UTN-USERFirst-Hardware", signature: "sha1RSA", thumb-
print algorithm: "sha
1", thumbprint: "0483ED3399AC3608058722EDBC5E4600E3BEF9D7"
t algorithm: "", thumbprint: ""
```

```
^SBNR: 5, size: "1082", issuer: "/C=SE/O=AddTrust AB/OU=AddTrust External
TTP Network/CN=A
ddTrust External CA Root", serial number: "01", subject: "/C=SE/
O=AddTrust AB/OU=AddTrust
External TTP Network/CN=AddTrust External CA Root", signature: "sha1RSA",
thumbprint algor
ithm: "sha1", thumbprint: "02FAF3E291435468607857694DF5E45B68851868"
^SBNR: 6, size: "0", issuer: "", serial number: "", subject: "", signa-
ture: "", thumbprint
 algorithm: "", thumbprint: ""
^SBNR: 7, size: "0", issuer: "", serial number: "", subject: "", signa-
ture: "", thumbprint
 algorithm: "", thumbprint: ""
^SBNR: 8, size: "0", issuer: "", serial number: "", subject: "", signa-
ture: "", thumbprint
 algorithm: "", thumbprint: ""
^SBNR: 9, size: "0", issuer: "", serial number: "", subject: "", signa-
ture: "", thumbprint
 algorithm: "", thumbprint: ""
^SBNR: 10, size: "0", issuer: "", serial number: "", subject: "", signa-
ture: "", thumbprint: ""

OK
```

Step 2: Configuring Internet Connection and Service profiles:

```
AT^SIND=is_cert,1                                Activate  certificate "+CIEV:"
^SIND: is_cert,1                                 indicator.
OK
AT+CMER=3,0,0,2                                  Activate event reporting (neces-
OK                                               sary for "+CIEV:" indicators).

AT^SICS=0,conType,"GPRS0"                        Create Internet Connection Pro-
OK                                               file
at^sics=0,apn,internet.t-d1.de
OK

at^siss=3,srvType,http                           Create Internet Service Profile
OK
at^siss=3,conId,0
OK
at^siss=3,address,https://www.psd-berlin-bran-
denburg.de/
OK
at^siss=3,hcMethod,0
OK
at^siss=3,secOpt,1,2,3,4,5
OK
```

```
AT^SISS?
^SISS: 0,"srvType",""
^SISS: 1,"srvType",""
^SISS: 2,"srvType",""
^SISS: 3,"srvType","Http"
^SISS: 3,"conId","0"
^SISS: 3,"alphabet","0"
^SISS: 3,"hcMethod","0"
^SISS: 3,"hcContLen","0"
^SISS: 3,"hcAuth","0"
^SISS: 3,"hcRedir","1"
^SISS: 3,"address","https://www.psd-berlin-
brandenburg.de/"
^SISS: 3,"hcContent",""
^SISS: 3,"hcProp","Accept-Encoding: identity"
^SISS: 3,"user",""
^SISS: 3,"passwd","*****"
^SISS: 3,"hcUsrAgent","MC75/4.1"
^SISS: 3,"tcpMR","30"
.....
^SISS: 9,"srvType",""
OK
```

Check Internet service profile.

Step 3: Open Internet connection:

```
AT^SISO=3
OK
```

Open Internet connection.

```
+CIEV: is_cert,3,/C=US/ST=UT/L=Salt Lake City/
O=The USERTRUST Network/OU=http://www.user-
trust.com/CN=UTN-USERFirst-Hard-
ware,168EB6F9F707599EA8460C62515F6D76,/C=DE/
ST=Nordrhein-Westfalen/L=Bonn/O=Servicegesell-
schaft der PSD Banken mbH/OU=Internet/OU=Comodo
InstantSSL/CN=www.psd-berlin-branden-
burg.de,sha1RSA,sha1,E01B6B7D565CAF2276F475A9B
1072634F0C08438
```

"+CIEV:" indicator displays server certificate.

```
^SIS: 3, 0, 2201, "HTTP/1.1 302 Found"

^SIS: 3, 0, 2200, "HTTP Redirect to:http://
www.psd-berlin-brandenburg.de:80/c51/
Default.html"

^SIS: 3, 0, 2201, "HTTP/1.1 302 Moved Temporar-
ily"

^SIS: 3, 0, 2200, "HTTP Redirect to:http://
www.psd-berlin-brandenburg.de:80/c51.html"

^SIS: 3, 0, 2201, "HTTP/1.1 200 OK"
```

"^SIS:" URCs show the connection status.

```
^SISR: 3, 1
```

"^SISR:" URC confirms that data is available for reading.

```
AT^SISR=3,1500                                  Start reading data.
^SISR: 3, 1209
...data...                                      Data transfer not shown in example.
...data...
OK
AT^SISR=3,1500
^SISR: 3, 1500
...data...
...data...
OK
AT^SISC=3                                       Close Internet connection.
OK
```

3.8 Trying to Make Secure TCP/IP Connection without Valid Certificate Stored

This chapter shows what happens if you try to set up a secure Internet connection, although no valid certificates are stored yet.

Example: Secure connection without valid server certificate

```
AT^SIND=is_cert,1                                  Activate  certificate "+CIEV:" indi-
^SIND: is_cert,1                                   cator.
OK
AT+CMER=3,0,0,2                                    Activate event reporting (neces-
OK                                                 sary for "+CIEV:" indicators).

AT^SICS=0,conType,"GPRS0"                          Create Internet Connection Pro-
OK                                                 file
at^sics=0,apn,internet.t-d1.de
OK

AT^SISS=3,srvType"HTTP"                            Create secure Internet service
OK                                                 profile for HTTPS download.
AT^SISS=3,conId,2                                  Set ConID created before with
OK                                                 AT^SICS.
AT^SISS=3,hcMethod,0
OK
AT^SISS=3,address,https://www.google.de            Set address type HTTPS.
OK
AT^SISS=3,secOpt,1                                 Select secure Internet connec-
OK                                                 tion inlcuding check for certificate
                                                   stored at index 1.

AT^SISO=3                                          Open Internet connection.
OK                                                 Server certificate is shown.

+CIEV: is_cert,3,/C=ZA/O=Thawte Consulting (Pty) Ltd./CN=Thawte SGC
CA,4F9D96D966B0992B54C2957CB4157D4D,/C=US/ST=California/L=Mountain
View/O=Google Inc/CN=www.google.com,sha1RSA,sha1,C1956DC8A7DFB2A5A56
934DA09778E3A11023358

^SIS: 3, 0, 66, "Peer certificate is not con-     "^SIS" URC reports connection
firmed"                                            error.

^SIS: 3, 0, 14, "An established connection was
aborted (transmission time-out or protocol
error)"


AT^SISC=3                                          Close connection.
OK
```

4 Secure TCP/IP Connections without Server Authentication

Create Internet Connection Profiles with AT^SICS and Service profiles with AT^SISS as usual, and simply add the "secOpt" parameter "-1" to the Service Profile. For the HTTP client set HTTPS within the address. To verify whether or not the server sends a certificate activate the certificate indicator "+CIEV: is_cert" with AT^SIND.

When opening the connection with AT^SISO, a secure connection will be set up, i.e., all transferred data is encrypted, but the certificate received from the server will not be checked.

Example: Secure TCP/IP connection without verifying server certificate

```
AT^SIND=is_cert,1                              Activate  certificate "+CIEV:" indi-
^SIND: is_cert,1                               cator.
OK
AT+CMER=3,0,0,2                                Activate event reporting (neces-
OK                                             sary for "+CIEV:" indicators).

AT^SICS=0,conType,"GPRS0"                      Create Internet Connection Pro-
OK                                             file
at^sics=0,apn,internet.t-d1.de
OK

AT^SISS=3,srvType"HTTP"                        Create secure Internet service
OK                                             profile for HTTPS download.
AT^SISS=3,conId,2                              Set ConID created before with
OK                                             AT^SICS.
AT^SISS=3,hcMethod,0
OK
AT^SISS=3,address,https://www.google.de        Set address type HTTPS.
OK
AT^SISS=3,secOpt,-1                            Select secure Internet connec-
OK                                             tion without certificate check.

AT^SISO=3                                      Open Internet connection.
OK                                             Server certificate is shown.

+CIEV: is_cert,3,/C=ZA/O=Thawte Consulting (Pty) Ltd./CN=Thawte SGC
CA,4F9D96D966B0992B54C2957CB4157D4D,/C=US/ST=California/L=Mountain
View/O=Google Inc/CN=www.google.com,sha1RSA,sha1,C1956DC8A7DFB2A5A56
934DA09778E3A11023358
```

^SIS: 3, 0, 2201, "HTTP/1.1 302 Found" ^SIS: 3, 0, 2200, "HTTP Redirect to:http://www.google.com:80/" ^SIS: 3, 0, 2201, "HTTP/1.1 302 Found" ^SIS: 3, 0, 2200, "HTTP Redirect to:http://www.google.de:80/"	"^SIS" URC reports connection status. Example shows that the connection is redirected to HTTP (non-TLS communication).
^SIS: 3, 0, 2201, "HTTP/1.1 200 OK" ^SISR: 3, 1	"SISR" URC indicates that service is ready to read.
AT^SISC=3 OK	Close connection.

5 Appendix: Creating a Secure Certificate Infrastructure

The Appendix provides basic step-by-step instructions based on the tools listed in Section 2.1. The example shows how to create a certificate chain on a Linux OS. The example uses the passwords and names listed in Section 5.3 and Section 5.4.

5.1 Recommended Environment and Tools

Environment and tools recommended for generating and managing certificates and keys:

- **For Microsoft Windows: Cygwin**, a Unix-like command line tool for Microsoft Windows. See http://www.cygwin.com/
- **For Linux: Linux Terminal**
- **Openssl**: Cryptography toolkit implementing SSL/TLS network protocols and standards. See http://www.openssl.org/docs/apps/openssl.html. Needed to generate CA root certificate and server certificates.
- **Keytool.exe:** Application used to generate pairs of keys for data signing and create keystore files. Keytool.exe is part of the Java SDK and can be obtained from http://www.oracle.com/technetwork/java/javase/downloads/index.html.
- Tools supplied by Gemalto M2M and attached to this PDF (see Section 3.2.1):
  - **cmd_ipCertMgr.jar:** Tool for loading certificates into the module's NVRAM, reading and deleting stored certificates. For explanations see Section 3 and especially Section 3.2.
  - **RXTX serial library:** Required for and supplied with cmd_ipCertMgr.jar tool. See also http://users.frii/jarvi/rxtx/.
  - **getPrivateKey.jar:** Keystore tool for extracting private and public keys from keystore
  - **setPrivateKey.jar:** Keystore tool for importing stored client private keys and client certificate from NVRAM into keystore

5.2 Reference System for Linux

Server:

- Ubuntu 11.04 Desktop Edition
- Apache 2 standard installation, SSL activated

Client:

- Cinterion® wireless module supporting TLS
- No certificates stored yet (including client certificate), certificate store empty

System used to create the certificates:

- Ubuntu 11.04 Desktop Edition
- Java Runtime Environment installed
- Openssl installed
- Module connected to /dev/ttyS0, baudrate is 115200 baud
- Directory for certificate management, for example "demoCA"
- Java tools cmd_IpCertMgr.jar, getprivatekey.jar, setprivatekey.jar, RXTX serial library, all of them located in the same directory
- Required only once to allow to sign CA certificates: add two lines to ssl configuration:

```
echo "[ my_v3_ext ]" >> /etc/ssl/openssl.cnf
echo "basicConstraints = CA:true" >> /etc/ssl/openssl.cnf
```

5.3 Certificate Content

Table 4 lists the X.500 distinguished name strings needed for the fields **issuer** and **subject** in X.509 certificates. The table shows the names used for the following examples.

Table 4: Samples names used for certificate content

Field item	Abbreviation	Examples used below
countryName	C	DE
stateOrProvinceName	ST	Berlin
LocalityNam	L	Berlin
organizationName	O	demo
organizationalUnitName	OU	demo
commonName	CN	rootCA.example.de
emailAddress	emailAddress	rootCA@example.de

5.4 Sample Names Used for Key Stores, Certificates, Passwords

Below you can find a list of names used for keystores, certificates, passwords in the following examples.

Table 5: Samples for Local Client Certificate

Item	Used examples
local client keystore file: keystore password:	client.ks pwdclient
local client certificate name: client certificate password:	client01 pwdclient01

Table 6: Samples for server certificates

Item	Used examples
server keystore file: server keystore password:	server.ks pwdserver
root CA certificate: root CA password:	rootCA pwdrootCA
level2 CA certificate: level2 CA password:	level2CA pwdlevel2CA
level3 certificate: level3 password:	level3 pwdlevel3

Table 7: Module

Item	Used example
IMEI	123456789123456

5.5 Generating Local Client Certificate and Key Store

Create local client keystore and local client certificate:

```
keytool -genkeypair -alias client01 -keypass pwdclient01\
-keystore client.ks -storepass pwdclient -sigalg SHA1withRSA -keyalg RSA
```

Export the created certificate to a dedicated file in DER format:

```
keytool -exportcert -v -keystore client.ks -storepass pwdclient\
-alias client01 -file client01_pub.der
```

5.6 Generating Server CA, Certficates and Keystore

Create a new self-signed CA.

```
perl /usr/lib/ssl/misc/CA.pl -newca
```

Export private and public key to dedicated files:

```
openssl x509 -in ./demoCA/cacert.pem -inform PEM -out rootCA_pub.der\
-outform DER
openssl pkcs8 -in ./demoCA/private/cakey.pem -inform PEM\
-out rootCA_priv.der -outform DER -nocrypt -topk8
openssl x509 -in rootCA_pub.der -inform DER -out rootCA_pub.pem\
-outform PEM
```

Create server keystore and level2 CA keys:

```
keytool -genkeypair -alias level2CA -keypass pwdlevel2CA\
-keystore server.ks -storepass pwdserver -sigalg SHA1withRSA -keyalg RSA
```

Generate certificate signing request and store this request to a ".csr" file:

```
keytool -certreq -alias level2CA -keystore server.ks\
-storepass pwdserver -file level2CA_req.csr
```

Certify the level2 key, create the certificate, and add CA flag. This flag is necessary to certify further key, e.g. the level3 key. Also, convert the certificate to DER format:

```
openssl ca -in level2CA_req.csr -out level2CA_cnf.pem\
-extensions my_v3_ext
openssl x509 -in level2CA_cnf.pem -inform PEM -out level2CA_pub.der\
-outform DER
```

Create the level3 CA:

```
keytool -genkeypair -alias level3 -keypass pwdlevel3\
-keystore server.ks -storepass pwdserver -sigalg SHA1withRSA -keyalg RSA
```

Generate certificate signing request and store this request to a ".csr" file:

```
keytool -certreq -alias level3 -file level3_req.csr -keypass pwdlevel3\
-keystore server.ks -storepass pwdserver
```

Extract the private and public key from level2 and convert them to PEM format

```
java -jar getPrivateKey.jar -alias level2CA -keypass pwdlevel2CA\
-keystore server.ks -storepass pwdserver -keyfile level2CA_priv.der
openssl pkcs8 -in level2CA_priv.der -inform DER -out level2CA_priv.pem\
-outform PEM -nocrypt
openssl pkcs8 -in level2CA_pub.der -inform DER -out level2CA_pub.pem\
-outform PEM -nocrypt
```

Certify the level3 key via level2 certificate and create level3 certificate. The certificate is converted to the required formats

```
openssl ca -keyfile level2CA_priv.pem -cert level2CA_cnf.pem\
-out level3_cnf.pem -infiles level3_req.csr
openssl x509 -in level3_cnf.pem -inform PEM -out level3_pub.der\
-outform DER
```

For the Apache server, private and public keys have to be created in PEM format:

```
openssl pkcs8 -in level3_priv.der -inform DER -out level3_priv.pem\
-outform PEM -nocrypt
openssl x509 -in level3_pub.der -inform DER -out level3_pub.pem\
-outform PEM
```

5.7 Loading Certificates into Module’s NVRAM

Use "cmd_ipCertMgr.jar" tool to load the generated certificates into the module’s NVRAM:

```
java -jar cmd_IpCertMgr.jar -serialPort /dev/ttyS0 -serialSpd 115200\
-cmd writecert -certfile client01_pub.der -certIndex 0\
-imei 099999001234561 -alias client01 -keypass pwdclient01\
-keystore client.ks -storepass pwdclient
java -jar cmd_IpCertMgr.jar -serialPort /dev/ttyS0 -serialSpd 115200\
-cmd writecert -certfile rootCA_pub.der -certIndex 1\
-imei 099999001234561 -alias client01 -keypass pwdclient01\
-keystore client.ks -storepass pwdclient
```

5.8 Reading Certificates Stored on Module

Enter command AT^SBNR=is_cert to read the stored certificates:

```
AT^SBNR=is_cert
^SBNR: 0, size: "571", issuer: "/C=DE/ST=Berlin/L=Berlin/O=demo/OU=demo/
CN=client01", serial number: "4E79E634", subject:\ "/C=DE/ST=Berlin/L=Ber-
lin/O=demo/OU=demo/CN=client01", signature: "SHAwRSA", thumbprint algo-
rithm: "SHA-0",\ thumbprint: "85C96375EB579C9BBA0122924D4DB91061D2C837"
^SBNR: 1, size: "854", issuer: "/C=DE/ST=Berlin/O=demo/OU=demo/
CN=rootCA.example.de/emailAddress=rootCA@example.de",\ serial number:
"00F1F4565330802E0B", subject:\ "/C=DE/ST=Berlin/O=demo/OU=demo/
CN=rootCA.example.de/emailAddress=rootCA@example.de", signature:\
"SHAwRSA", thumbprint algorithm: "SHA-0", thumbprint:
"E0A60E9EFB09099D2017B2B9140A3BBAC155FD6F"
^SBNR: 2, size: "0", issuer: "", serial number: "", subject: "", signature:
"", thumbprint algorithm: "", thumbprint: ""
^SBNR: 3, size: "0", issuer: "", serial number: "", subject: "", signature:
"", thumbprint algorithm: "", thumbprint: ""
^SBNR: 4, size: "0", issuer: "", serial number: "", subject: "", signature:
"", thumbprint algorithm: "", thumbprint: ""
^SBNR: 5, size: "0", issuer: "", serial number: "", subject: "", signature:
"", thumbprint algorithm: "", thumbprint: ""
^SBNR: 6, size: "0", issuer: "", serial number: "", subject: "", signature:
"", thumbprint algorithm: "", thumbprint: ""
^SBNR: 7, size: "0", issuer: "", serial number: "", subject: "", signature:
"", thumbprint algorithm: "", thumbprint: ""
^SBNR: 8, size: "0", issuer: "", serial number: "", subject: "", signature:
"", thumbprint algorithm: "", thumbprint: ""
^SBNR: 9, size: "0", issuer: "", serial number: "", subject: "", signature:
"", thumbprint algorithm: "", thumbprint: ""
^SBNR: 10, size: "0", issuer: "", serial number: "", subject: "", signature:
"", thumbprint algorithm: "", thumbprint: ""
OK
```

5.9 Creating Apache Certificates

Certificate files on the Apache server are required in PEM format. The following commands can be used:

```
cat level3_priv.pem level3_cnf.pem > apache.pem
cat level2CA_cnf.pem rootCA_pub.pem > chain.pem
```

A little manual effort is needed: Manually edit these files; remove the certificate description (all which is tagged with "Certificate:"), only the parts framed with

```
-----BEGIN/END RSA PRIVATE KEY-----
```

and

```
-----BEGIN/END CERTIFICATE-----
```

(including these headers) are allowed.

5.10 Copying Server Certificate (Including Private Key) to Server

Apache has to be restarted with the new certificates. Also, Apache needs a link with a hash.

1. Stop Apache server:

   ```
   /etc/init.d/apache2 stop
   ```

2. Copy edited apache.pem file to /etc/apache2/ssl/apache.pem.

3. Create a symbolik link to the .pem file so that Apache will find it by its hash:

   ```
   ln -sf /etc/apache2/ssl/apache.pem /etc/apache2/ssl/`/usr/bin/openssl\
   x509 -noout -hash < /etc/apache2/ssl/apache.pem`.0
   ```

4. Copy edited chain.pem file to /etc/apache2/ssl/chain.pem.

5. Modify /etc/apache2/sites-enabled/ssl as shown below. Especially add SSLCertificateChainFile directive.
   Example configuration file for Apache2 site (e.g. /etc/apache2/sites-enabled/ssl):

   ```
   NameVirtualHost *:443
   <virtualhost *:443>
      ServerName level3.example.de
      SSLEngine On
      SSLCertificateFile /etc/apache2/ssl/apache.pem
      SSLCertificateChainFile /etc/apache2/ssl/chain.pem
      DocumentRoot /var/www
   </virtualhost>
   ```

6. Restart Apache server:

   ```
   /etc/init.d/apache2 start
   ```

5.11 Secure Connection to Server

Please refer to Section 3.7 for an AT command example of a secure connection set up to the Apache server by a transparent TCP client on the module.

6 Secure Commands

Secure commands are a communication protocol between PC Application and the module (AT^SBNW=is_cert,1).

6.1 General secure command structure

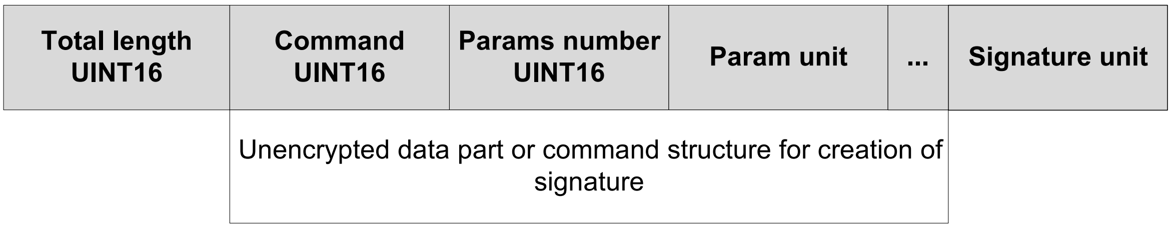


Figure 1: General secure command structure

6.2 Command type

Table 8: Command type

Command ID	Description
0x0001	WRITE Certificate
0x0002	READ Certificate
0x0003	DELETE Certificate

6.3 Param unit structure

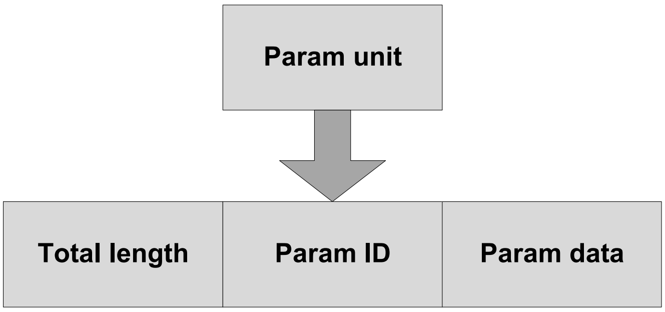


Figure 2: Param unit structure

6.4 Param unit type

Table 9: Param unit type

Param ID	Param Name	Param values	Description
0x0001	cert index	integer from 0 to 10	Certificate index. Certificate with index=0 is handled as client certificate (only 1 allowed).
0x0002	cert data	binary content of *.der file	DER encoded certificate file. Assumptions: maximum 2kB per certificate
0x0003	signature	zero terminated string	SHA-1 signature of the command, base 64 coded.
0x0004	IMEI	zero terminated string	numeric numbers in ASCII format
0x0005	private key	binary content of *.der file	DER encoded client private key. Assumptions: maximum 4kB per private key only allowed to be written together with client certificate (index=0)

6.5 Command structure

6.5.1 WRITE command structure

Table 10: WRITE command structure

Length	Command	Param num	Param unit	Param unit	Param unit	Param unit
	0x0001	4	cert index	cert data	IMEI	signature

6.5.2 WRITE command structure for client certificate

Table 11: Structure of WRITE command for client certificate

Length	Command	Param num	Param unit	Param unit	Param unit	Param unit	Param unit
	0x0001	5	cert index	cert data	private key	IMEI	signature

6.5.3 READ command structure

Table 12: READ command structure

Length	Command	Param num	Param unit	Param unit	Param unit
	0x0002	3	cert index	IMEI	signature

6.5.4 READ command response structure

Table 13: READ command response structure

Length	Command	Param num	Param unit	Param unit	Param unit
	0x0002	3	cert index	cert data	signature

6.5.5 DELETE command structure

Table 14: DELETE command structure

Length	Command	Param num	Param unit	Param unit	Param unit
	0x0003	3	cert index	IMEI	signature

6.6 Command security

The module allows only secure commands with valid IMEI and signature. For using secure commands the client certificate has to be written first and is always stored at index 0.

About Gemalto

Gemalto (Euronext NL0000400653 GTO) is the world leader in digital security with 2011 annual revenues of €2 billion and more than 10,000 employees operating out of 74 offices and 14 Research & Development centers, located in 43 countries.

We are at the heart of the rapidly evolving digital society. Billions of people worldwide increasingly want the freedom to communicate, travel, shop, bank, entertain and work - anytime, everywhere - in ways that are enjoyable and safe. Gemalto delivers on their expanding needs for personal mobile services, payment security, authenticated cloud access, identity and privacy protection, eHealthcare and eGovernment efficiency, convenient ticketing and dependable machine-to-machine (M2M) applications.

Gemalto develops secure embedded software and secure products which we design and personalize. Our platforms and services manage these secure products, the confidential data they contain and the trusted end-user services they enable. Our inovations enable our clients to offer trusted and convenient digital services to billions of individuals.

Gemalto thrives with the growing number of people using its solutions to interact with the digital and wireless world.

For more information please visit
m2m.gemalto.com, www.facebook.com/gemalto, or Follow@gemaltom2m on twitter.

Gemalto M2M GmbH
St.-Martin-Str. 60
81541 Munich
Germany

gemalto
security to be free