life.augmented

ST87MXX TLS Application Note

Purpose and scope

This document gives details on the AT Commands and general usage for provisioning security credentials and establishing a secure connection based on embedded TLS stack supported by the ST87MXX NB-IoT module.

Document status
Official

1 General information

1.1 Acronyms and terms

Table 1. Definitions of terms

Term	Definition
DNS	Domain Name System
IP	Internet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
TLS	Transport Layer Security
DTLS	Datagram Transport Layer Security
CA	Certificate Authority
IANA	Internet Assigned Numbers Authority
CI	Certificate Issuer
CSR	Certificate Signing Request
DER	Distinguished Encoding Rule (binary encoding of ASN.1 notation, used to encapsulate certificates and keys)
PEM	Privacy Enhanced Mail Certificate (Base64 encoding of DER certificate)
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
PKI	Public Key Infrastructure
PSK	Pre-Shared Key
RoT	Root of Trust
TRNG	True Random Number Generator
ITS	Internal Trusted Storage
UART	Universal Asynchronous Receiver Transmitter
URC	Unsolicited Result Code
ME	Mobile Equipment
TA	Terminal Adaptor (e.g., GSM data card; equal to DCE - Data Circuit terminating Equipment)
TE	Terminal Equipment (e.g., computer; equal to DTE - Data Terminal Equipment)
IoT	Internet of Things

1.2 Reference documents

The documents listed in Table 2 provide further information.

Table 2. Document references

Reference	Document
[1]	ST87MXX UM AT commands description
[2]	IANA TLS Parameters (https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml)

1.3 Revision history

Table 3. Document revision history

Date	Version	Changes
2024-10-01	V1.0	- Official version
2025-01-09	V1.1	- Remove Error and URC lists
2025-06-26	V1.2	- Update the provisioning commands syntax
2025-06-30	V1.3	- Remove AT command tables and add ST87MXX UM AT commands description reference
2025-07-30	V2.0	- Update for release

1.4 Table of contents

1.5 List of tables

1.6 List of Figures

2 Introduction

This document describes the use of the embedded TLS stack in the ST87MXX, which provides essential security functions to ensure communication confidentiality and protect data exchanged between the server and client from interception, tampering or falsification through encryption. It offers a selection of AT commands to facilitate TLS communication.
This document assumes that the reader is familiar with the fundamental concepts of TLS protocol, including handshake and session mechanisms, as well as cryptographic security principles. **For a complete description of the AT commands and parameters, please refer to the AT Command User Manual [1].**

2.1 TLS version

The module supports the following versions of the TLS protocol.

Table 4. Supported TLS Versions

TLS version
TLS 1.2
TLS 1.3

2.2 DTLS version

The module supports the following versions of the DTLS protocol.

Table 5. Supported DTLS Versions

DTLS version
DTLS 1.2

2.3 Cipher Suite version

The table below lists the cipher suites supported by the modules to ensure secure communication. Please refer to https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml [2] for details of the cipher suites.

Table 6. Supported Cipher Suites (Official IANA Names)

Category	Cipher Suite Name	Cipher Suite Code
TLS 1.3	TLS_AES_128_GCM_SHA256	0x1301
TLS 1.3	TLS_AES_128_CCM_SHA256	0x1304
TLS 1.3	TLS_AES_128_CCM_8_SHA256	0x1305
TLS/DTLS 1.2	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	0xc02b

TLS/DTLS 1.2	TLS_ECDHE_ECDSA_WITH_AES_128_CCM	0xc0ac
TLS/DTLS 1.2	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8	0xc0ae
TLS/DTLS 1.2	TLS_ECDHE_ECDSA_WITH_AES_256_CCM	0xc0ad
TLS/DTLS 1.2	TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8	0xc0af
TLS/DTLS 1.2	TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256	0xc037
TLS/DTLS 1.2	TLS_PSK_WITH_AES_128_GCM_SHA256	0x00a8
TLS/DTLS 1.2	TLS_PSK_WITH_AES_128_CBC_SHA256	0x00ae
TLS/DTLS 1.2	TLS_PSK_WITH_AES_128_CCM_8	0xc0a8
TLS/DTLS 1.2	TLS_PSK_WITH_AES_128_CCM	0xc0a4
TLS/DTLS 1.2	TLS_PSK_WITH_AES_256_CCM	0xc0a5
TLS/DTLS 1.2	TLS_PSK_WITH_AES_256_CCM_8	0xc0a9

3 Provisioning

3.1 Preamble

To enable secure IP data exchange over DTLS/TLS protocols, the user should perform provisioning to store cryptographic materials, such as keys and certificates, within the ME. This process enables mutual authentication between the client and server and establishes a secure communication channel.

The ME has a built-in Secure Element that safely keeps keys and certificates in Internal Trusted Storage (ITS). It includes a True Random Number Generator (TRNG) and can create key pairs with a private key stored securely within the ITS and a public key for external use. It can also sign Certificate Signing Requests (CSRs) using the stored private key. These signed CSRs can be then sent to a Certificate Issuer (CI) or Certificate Authority (CA) to obtain a valid certificate.

Cryptographic materials are organized into security profiles and securely stored within the ITS, allowing flexible and secure credential management. Note that access to ITS storage is available only through the ME's provisioning mechanism via AT commands.

3.2 Provisioning Workflow

This diagram below illustrates the provisioning workflow for configuring cryptographic materials, covering both symmetric key (PSK) and asymmetric key/certificate scenarios, including server and client authentication processes.

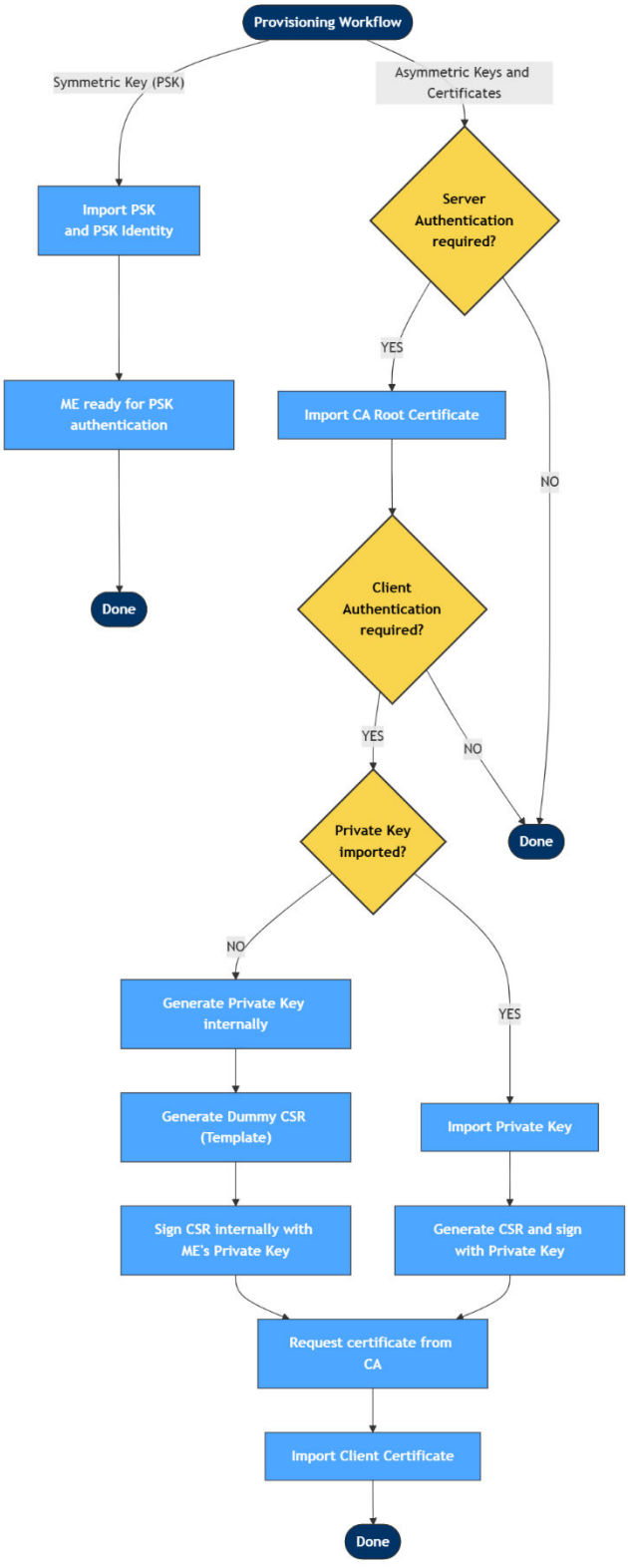


Figure 1. Provisioning Workflow Diagram

3.3 Provisioning Details

The ME supports flexible provisioning methods to configure the cryptographic credentials required to establish a secure communication channel.
These methods include:

- **Symmetric Keys (PSK):** The user generates and imports the PSK into the ME via UART using AT commands, as the ME cannot generate PSKs internally.
- **Asymmetric Keys and Certificates:**
  - **Key Pairs (private and public keys):** Private keys can be imported by the user via UART using AT commands or generated internally within the ME's Secure Element. In both cases, the ME internally derives the corresponding public key from the stored private key.
  - **Certificates**: The user imports the certificates into the ME via UART.

The cryptographic materials are stored as individual files in the ITS, which can hold up to $\boldsymbol{N = 10}$ files. Each security profile includes one header file containing the profile information ($\boldsymbol{F_{SecurProfile}}$) and a variable number of credential files ($\boldsymbol{F_{Credentials,i}}$).
Given $\boldsymbol{P}$ security profiles stored in ITS, where the $\boldsymbol{i}$-th profile contains $\boldsymbol{F_{Credentials,i}}$ credentials (such as key and certificates), the total number of files can be expressed as:

- Number of Secure Profile ID files:

$$\boldsymbol{F_{SecurProfile\,Total} = P}$$

- Total number of credential files:

$$\boldsymbol{F_{Credentials\,Total} = \sum_{i=1}^{P} F_{Credentials,i}}$$

Therefore, the total files stored in ITS should satisfy:

$$\boldsymbol{F_{Total} = F_{SecurProfile\,Total} + F_{Credentials\,Total} \leq N}$$

Example:
Consider two profiles stored in ITS with the following credential configuration:

- **Secure Profile 1**: a single credential file.
- **Secure Profile 2**: three credential files.

In this case:

- Number of Secure Profile ID files:

$$\boldsymbol{F_{SecurProfile\,Total} = 2}$$

- Total number of credential files:

$$\boldsymbol{F_{Credentials\,Total} = 1 + 3 = 4}$$

- Total files stored:

$$\boldsymbol{F_{Total} = 2 + 4 = 6 \leq 10}$$

This leaves 4 free slots in ITS for additional files, allowing for certificate updates while preserving the original credentials necessary for establishing the initial secure connection.

3.3.1 Symmetric Key Scenario

For PSK-based authentication, both the PSK and the corresponding PSK identity should be imported using AT commands. These credentials can be provided in binary or hexadecimal string format. After importing both, the module is ready to initiate DTLS/TLS sessions using PSK authentication.

3.3.2 Asymmetric Keys and Certificates Scenario

For certificate-based authentication, the provisioning process involves several steps:

- **Importing the CA Root Certificate:** if server authentication is required, the user should import into the ME the trusted CA Root Certificate. This certificate enables the module to authenticate the server's identity and establish a secure communication channel during the handshake process. Please note that certificate chain validation is not supported.
- **Client Authentication:** if client authentication is required, the user should import into the ME a private key. This key can be imported via UART using AT commands or generated internally within the module's Secure Element.
  - **Using an External Private Key:** the user imports a private key generated externally and securely stores it in ITS.
  - **Generating a Private Key Internally:** the ME generates the private key internally and securely stores it in ITS.
- **Generating and signing the CSR:** Once the private key is stored within the ME, the module can internally sign a Certificate Signing Request (CSR) imported via UART using AT commands. The CSR signing process will use the stored private key.
- **Importing the Client Certificate:** After the client certificate is issued, it should be imported into the ME to complete the provisioning process. This certificate enables authentication of the client's identity and the establishment of a secure communication channel during the handshake process.

3.3.3 Authentication Policy

The following table summarizes the authentication requirements for certificates and keys depending on the TLS/DTLS version used. It clarifies when the CA certificate and client credentials are mandatory or optional, and the behaviour of the ME in case of missing cryptographic materials during the authentication process.

Table 7. TLS/DTLS Authentication Policy

Credential Type	TLS 1.3	TLS/DTLS 1.2	Remarks
CA Certificate	Mandatory	Optional	Imported into the ME. Mandatory for server authentication in TLS 1.3; optional for TLS 1.2 and DTLS 1.2 (used if imported).

Client Certificate and Private Key	Mandatory if server requests client authentication	Mandatory if server requests client authentication	The ME returns an error during the handshake if client credentials are missing when requested by the server.

3.4 AT commands

The provisioning of the Pre-Shared Key (PSK) and its associated identity can be performed by executing the following AT commands:

AT#TLSKEYADD	Import the PSK into the ME
AT#TLSCERTADD	Import the PSK identity string into the ME
AT#RESET=1	Save the parameters to the ITS and restart the system

Note that once the PSK is imported via UART using AT commands, it cannot be read back, ensuring the confidentiality of the cryptographic material. However, the PSK identity can still be accessed via AT commands if needed.

For asymmetric keys and certificates scenario, the following commands are used to import the CA Root Certificate, private keys, and client certificates:

AT#TLSCERTADD	Import the CA Root and Client Certificates
AT#TLSKEYADD	Import a private key or generate it internally within the ME
AT#RESET=1	Save the parameters to the ITS and restart the system

Only certificates in DER format are supported and can be provided as binary data or hexadecimal strings. The private key can be imported in DER or RAW format; however, once stored—whether imported or generated—it cannot be read back, ensuring the confidentiality of the cryptographic material. Providing the public key is not necessary, as it is generated internally by the module.

The signing of a Certificate Signing Request (CSR) is performed using the following command:

AT#TLSCERTSIGN	Sign a CSR using the private key stored within the ME.

This command requires the import of a dummy CSR in DER format. The dummy CSR, created externally using a temporary private key, contains placeholder fields for the subject, public key, and signature. Upon import, the ME replaces the public key and signature with those generated internally using the stored private key, while the subject remains unchanged. Signing the CSR with this command is mandatory when the private key is generated within the ME; if the private key is imported via UART using AT commands, signing the CSR with the ME is optional.

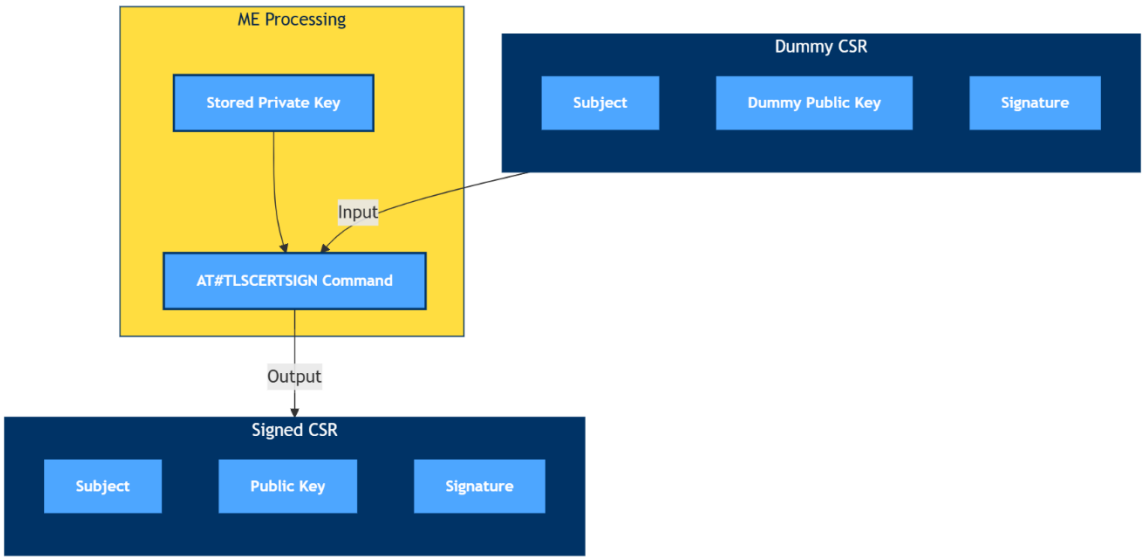


Figure 2. CSR Signing Workflow in ME

4 Provisioning scenario examples

4.1 Command descriptions

The provisioning of security credentials on the ST87MXX module is performed using the following AT commands:

- **AT#TLSCERTADD**=<sec_id>,<type>[,<data_length>[,<data>]]
- **AT#TLSKEYADD**=<sec_id>,<type>,<storage>,<data_flag><data_length>[,<data>]
- **AT#TLSCERTSIGN**=<sec_id>,<type>,<data_length>[,<data>]

See the AT command user manual [1] for more details on the parameters.

4.2 Symmetric Key Scenario

4.2.1 Parameters

Here is an example of AT commands sequence to perform provisioning with a 32-byte PSK.

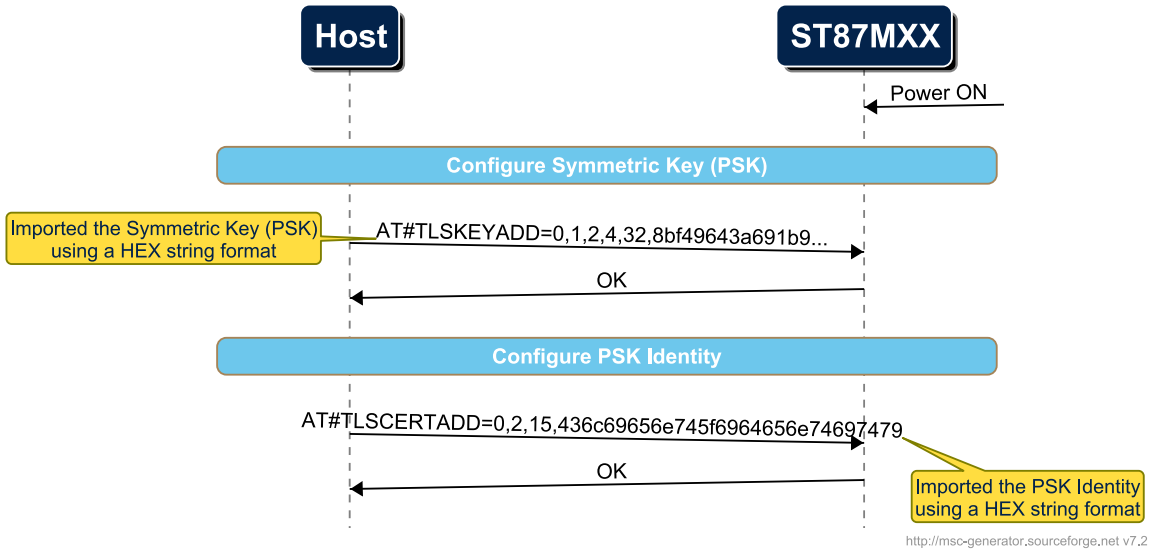
The parameters for importing the PSK are set as follows:

- **<sec_id> = 0** to link the key to Security Profile ID 0.
- **<type> = 1** to specify the symmetric key type.
- **<storage> = 2** to store the key in Flash.
- **<data_flag> = 4** to indicate that the imported PSK is in raw format.
- **<data_ length>** to specify the length in bytes of the data.
- **<data>** to represent the PSK raw data as a hexadecimal string.
-

The parameters for importing the PSK identity are set as follows:

- **<sec_id> = 0** to link the key to Security Profile ID 0.
- **<type> = 2** to specify the PSK identity type.
- **<data_ length>** to specify the length in bytes of the data.
- **<data>** to represent the PSK identity as a hexadecimal string.

4.2.2 MSC



4.2.3 Terminal

Below is the sequence obtained in the UART terminal when executing the scenario (in **Bold** the commands sent to the module):

```
AT#TLSKEYADD=0,1,2,4,32,8bf49643a691b9835a79c7c2a2de44ac0a4e2f1ec816c3ad1cea92a6
4d3127aa
OK

AT#TLSCERTADD=0,2,15,436c69656e745f6964656e74697479
OK
```

4.3 Asymmetric Keys and Certificates Scenario: Private Key imported

4.3.1 Parameters

Here is an example of the AT command sequence to perform provisioning using an imported 32-byte asymmetric private key.

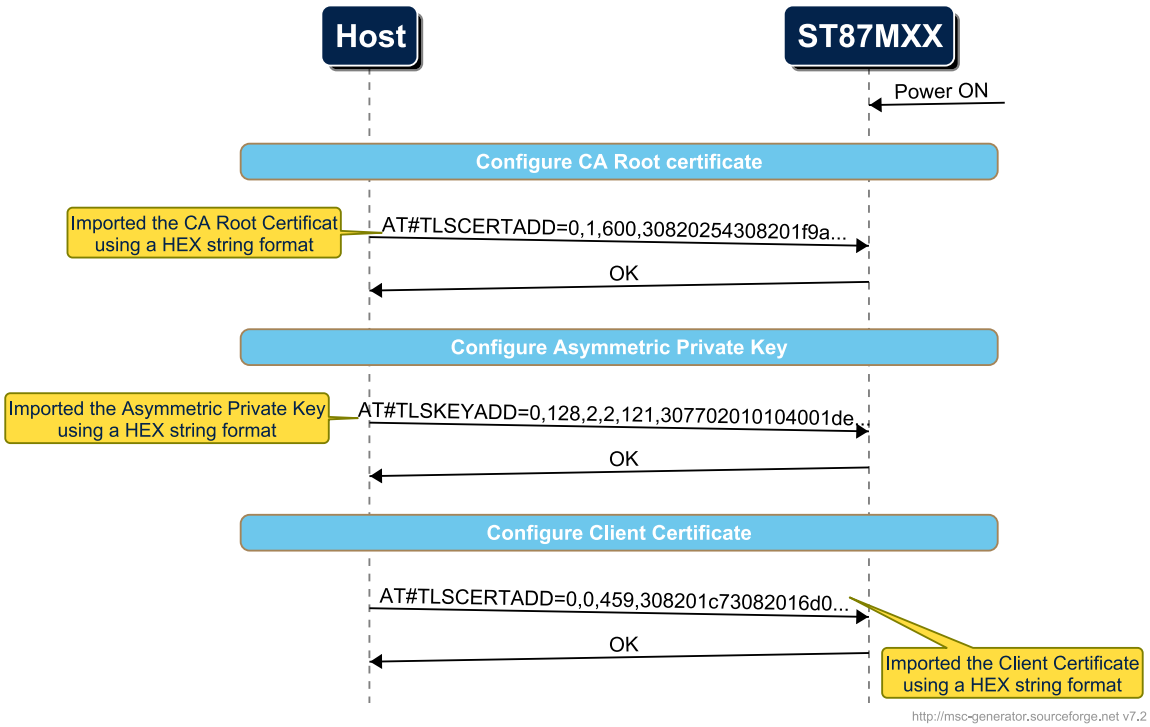
The parameters for importing the CA root certificate are set as follows:
- **<sec_id> = 0** to link the certificate to Security Profile ID 0.
- **<type> = 1** to specify the CA root certificate type.
- **<data_length>** to specify the length in bytes of data.
- **<data>** to represent the CA root certificate in DER format as a hexadecimal string.

The parameters for importing the private key are set as follows:
- **<sec_id> = 0** to link the key to Security Profile ID 0.
- **<type> = 128** to specify that the imported key is of type private key and is encrypted using the ECC SECP R1 algorithm, and that if the corresponding public key is requested, it will be returned as a hexadecimal string.
- **<data_flag> = 2** to enable internal generation of the public key paired with the imported private key.
- **<data_length>** to specify the length in bytes of the data.
- **<data>** to represent the private key in DER format as a hexadecimal string.

The parameters for importing the Client certificate are set as follows:
- **<sec_id> = 0** to link the certificate to Security Profile ID 0.
- **<type> = 0** to specify the Client certificate type.
- **<data_length>** to specify to specify the length in bytes of data.
- **<data>** to represent the Client certificate in DER format as a hexadecimal string.

Important Note:
The commands to import the private key and the client certificate will return an error message if there is a key-certificate mismatch between the imported certificate and private key. The ME verifies this correspondence during both processes and returns an error if such inconsistency is detected.

4.3.2 MSC



4.3.3 Terminal

Below is the sequence obtained in the UART terminal when executing the scenario (in **Bold** the commands sent to the module):

```
AT#TLSCERTADD=0,1,600,30820254308201f9a00302010202146df0f0a64be186d9b4577a9067db
a8dd1550de4c300a06082a8648ce3d040302307f310b300906035504061302495431 0e300c060355
04080c054372656d61310e300c06035504070c054372656d613111300f060355040a0c0853544d20
496e632e310c300a060355040b0c034950433113301106035504030c0a7777772e73742e636f6d31
1a301806092a864886f70d010901160b696e666f4073742e636f6d301e170d323131323036313530
3135355a170d343930343233313530313535a307f310b300906035504061302495431 0e300c0603
5504080c054372656d61310e300c06035504070c054372656d613111300f060355040a0c0853544d
20496e632e310c300a060355040b0c034950433113301106035504030c0a7777772e73742e636f6d
311a301806092a864886f70d010901160b696e666f4073742e636f6d3059301306072a8648ce3d02
0106082a8648ce3d03010703420004ce15303af666c92e16f48cf88b65573834abe2d60409abc408
d9d9acdd1c71b360e6af0c45b55a445ca6b14332b207d3505bd0bfe9c6cf4190ed23774a08cb4da3
533051301d0603551d0e04160414158a41e653b14807612a3fe7e60d2694ab38efdb301f0603551d
23041830168014158a41e653b14807612a3fe7e60d2694ab38efdb300f0603551d130101ff040530
030101ff300a06082a8648ce3d040302034900304602210 0f3320b24f1859770b766e55030d21c61
a20cdad92481e5df86c75cacbf2b3a8e022100968c4c419b2601b4561e57ef38dcbebc7468056efb
588f863f206286372d69da
OK


AT#TLSKEYADD=0,128,2,2,121,3077020101042001dea9c8a68f8b2983d1bde16e71ec040bac4ee
fea007a88d73d4fc699a38758a00a06082a8648ce3d030107a14403420004c2a8159107bc66e14fc
bacc1ad2659cbab7d0ea7400444ba4bb277d5d3cad6e3df2984cbb19c2e0470744f0a7ed49a7c2fd
e852cca3d24ad192b8025cf6387d3
#TLSKEYADD: 0,130,2,2,91
3059301306072A8648CE3D020106082A8648CE3D03010703420004C2A8159107BC66E14FCBACC1AD
2659CBAB7D0EA7400444BA4BB277D5D3CAD6E3DF2984CBB19C2E0470744F0A7ED49A7C2FDE852CCA
```

```
3D24AD192B8025CF6387D3
OK

AT#TLSCERTADD=0,0,459,308201c73082016d02145cf0ada5ec4a129bc5dde507539baec8f475e4
f9300a06082a8648ce3d040302307f310b30090603550406130249543l0e300c06035504080c0543
72656d61310e300c06035504070c054372656d613111300f060355040a0c0853544d20496e632e31
0c300a060355040b0c0349504331133011060355040304030c0a7777772e73742e636f6d311a30180609
2a864886f70d010901160b696e666f4073742e636f6d301e170d323530383237313831343535a17
0d323630383238313831343535a304d310b30090603550406130249543l0d300b06035504070c04
4c6568693111300f060355040a0c0853544d20496e632e310b3009060355040b0c024954310f300d
06035504030c0673742e636f6d3059301306072a8648ce3d020106082a8648ce3d03010703420004
c2a8159107bc66e14fcbacc1ad2659cbab7d0ea7400444ba4bb277d5d3cad6e3df2984cbb19c2e04
70744f0a7ed49a7c2fde852cca3d24ad192b8025cf6387d3300a06082a8648ce3d040302034800030
45022100d8f7eeb3ce70743aceb7c8a615a3f768b5fe053b28664d6e658bab1d67836041022070c0
0d900f75ab92c39791fc287767aab724f8307121d40167eb6ea8817568d0
OK
```

4.4 Asymmetric Keys and Certificates Scenario: Private Key generated in ME

4.4.1 Parameters

Here is an example of the AT command sequence to perform provisioning using a 32-byte asymmetric private key generated in the ME.

The parameters for importing the CA root certificate are set as follows:
- **<sec_id> = 0** to link the certificate to Security Profile ID 0.
- **<type> = 1** to specify the CA root certificate type.
- **<data_length>** to specify the length in bytes of data.
- **<data>** to represent the CA root certificate in DER format as a hexadecimal string.

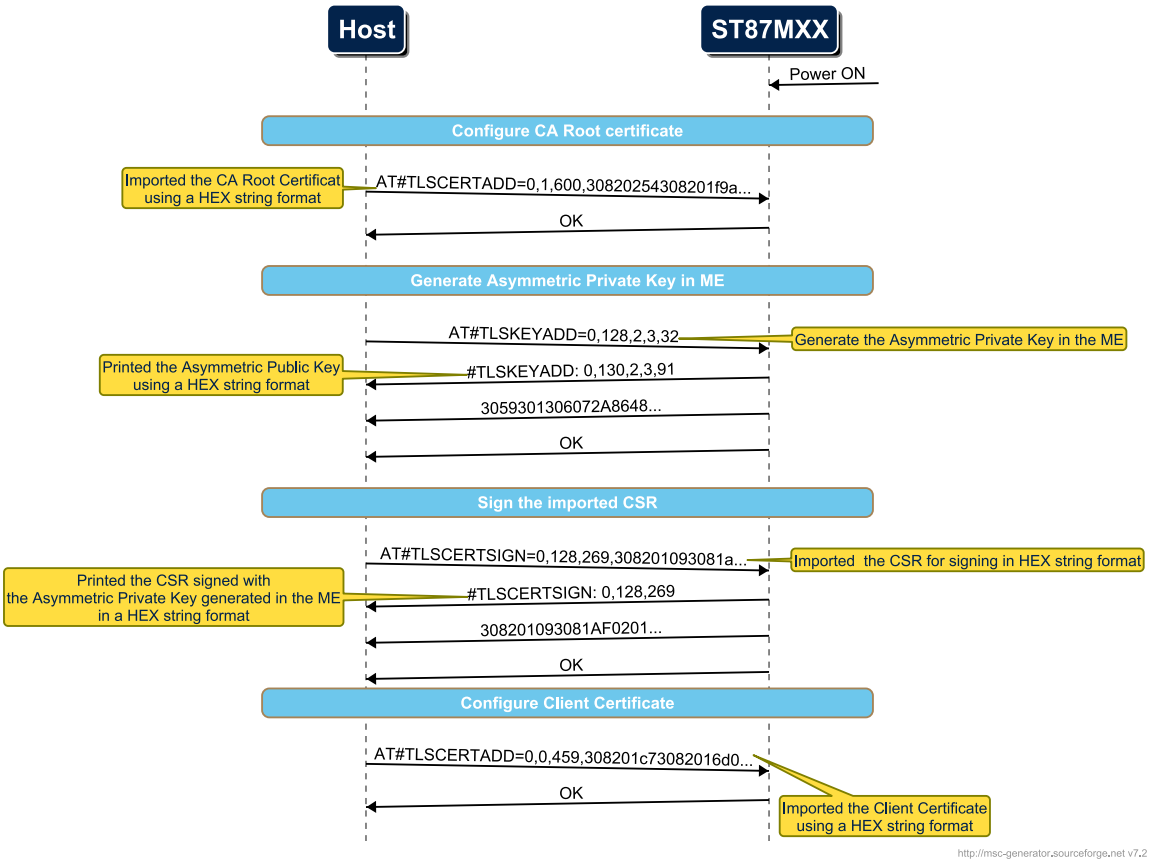
The parameters for generating the private key in the ME are set as follows:
- **<sec_id> = 0** to link the key to Security Profile ID 0.
- **<type> = 128** To specify that the key generation request is for a private key that should be encrypted using the ECC SECP R1 algorithm, and that if the corresponding public key is requested, it will be returned as a hexadecimal string.
- **<storage> = 2** to store the key in Flash.
- **<data_flag> = 3** to enable internal generation of the private key and to return the paired public key.
- **<data_length>** to specify the length in bytes of the private key.

The parameters for signing the CSR with the internally generated private key are set as follows:
- **<sec_id> = 0** to link the certificate to Security Profile ID 0.
- **<type> = 1** to specify the format for the signed CSR as a hexadecimal string.
- **<data_length>** to specify the length in bytes of data.
- **<data>** to represent the dummy CSR in DER format as a hexadecimal string.

The parameters for importing the Client certificate are set as follows:
- **<sec_id> = 0** to link the certificate to Security Profile ID 0.
- **<type> = 1** to specify the Client certificate type.
- **<data_length>** to specify the length in bytes of data.
- **<data>** to represent the Client certificate in DER format as a hexadecimal string.

Important Note:
The commands to generate the private key and to import the client certificate will return an error message if there is a key-certificate mismatch between the imported certificate and the internally generated private key. The ME verifies this correspondence during both processes and returns an error if such inconsistency is detected.

4.4.2 MSC



4.4.3 Terminal

Below is the sequence obtained in the UART terminal when executing the scenario (in **Bold** the commands sent to the module):

```
AT#TLSCERTADD=0,1,600,30820254308201f9a003020102021 46df0f0a64be186d9b4577a9067db
a8dd1550de4c300a06082a8648ce3d040302307f310b3009060355040613024954310e300c060355
04080c054372656d61310e300c06035504070c054372656d613111300f060355040a0c0853544d20
496e632e310c300a060355040b0c034950433113301106035504030c0a7777772e73742e636f6d31
1a301806092a864886f70d010901160b696e666f4073742e636f6d301e170d323131323036313530
3135355a170d343930343233313535303135355a307f310b3009060355040613024954310e300c0603
5504080c054372656d61310e300c06035504070c054372656d613111300f060355040a0c0853544d
20496e632e310c300a060355040b0c034950433113301106035504030c0a7777772e73742e636f6d
311a301806092a864886f70d010901160b696e666f4073742e636f6d3059301306072a8648ce3d02
0106082a8648ce3d03010703420004ce15303af666c92e16f48cf88b65573834abe2d60409abc408
d9d9acdd1c71b360e6af0c45b55a445ca6b14332b207d3505bd0bfe9c6cf4190ed23774a08cb4da3
533051301d0603551d0e04160414158a41e653b14807612a3fe7e60d2694ab38efdb301f0603551d
23041830168014158a41e653b14807612a3fe7e60d2694ab38efdb300f0603551d130101ff040530
030101ff300a06082a8648ce3d0403020349003046022100f3320b24f1859770b766e55030d21c61
```

```
a20cdad92481e5df86c75cacbf2b3a8e022100968c4c419b2601b4561e57ef38dcbebc7468056efb
588f863f206286372d69da
OK

AT#TLSKEYADD=0,128,2,3,32
#TLSKEYADD: 0,130,2,3,91
3059301306072A8648CE3D020106082A8648CE3D03010703420004C2A8159107BC66E14FCBACC1AD
2659CBAB7D0EA7400444BA4BB277D5D3CAD6E3DF2984CBB19C2E0470744F0A7ED49A7C2FDE852CCA
3D24AD192B8025CF6387D3
OK

AT#TLSCERTSIGN=0,128,269,308201093081af020100304d310b3009060355040613024954310d3
00b06035504070c044c6568693111300f060355040a0c0853544d20496e632e310b3009060355040
b0c024954310f300d06035504030c0673742e636f6d3059301306072a8648ce3d020106082a8648c
e3d030107034200042ff1339ab1c81010dbdff24efaf92a27880c223e1b71407ea14bd4b1f4e577b
32a4f8651364ffdaa507db5b12fc2a6f1ca50ae8cd40c011d7a2b50a66fd647bea000300a06082a8
648ce3d040302034900304602210 0d8827cf40a71e2115b0c709def5c79bf3bf692b41d8e093d900
154d1fddeab3c02210080f8121f57353df9c7680286f9c83b1f53fe6b549daccdaa8b9ae62880727
f0c

#TLSCERTSIGN: 0,128,269
308201093081AF020100304D310B3009060355040613024954310D300B06035504070C044C656869
3111300F060355040A0C0853544D20496E632E310B3009060355040B0C024954310F300D06035504
030C0673742E636F6D3059301306072A8648CE3D020106082A8648CE3D030107034200042FF1339A
B1C81010DBDFF24EFAF92A27880C223E1B71407EA14BD4B1F4E577B32A4F8651364FFDAA507DB5B1
2FC2A6F1CA50AE8CD40C011D7A2B50A66FD647BEA000300A06082A8648CE3D040302034900304602
2100C8AE5D8C3A403D80AE90391E865664EE2BCB5B1A34C9949DE2D915FB27F93D7C022100ABA6E5
8C076A1F33F80625F51F910771EF87172474A393BB7037363FED8E1290
OK

AT#TLSCERTADD=0,0,459,308201c73082016d02145cf0ada5ec4a129bc5dde507539baec8f475e4
f9300a06082a8648ce3d040302307f310b3009060355040613024954310e300c06035504080c0543
72656d61310e300c06035504070c054372656d613111300f060355040a0c0853544d20496e632e31
0c300a060355040b0c0349504331133011060355040 30c0a7777772e73742e636f6d311a30180609
2a864886f70d010901160b696e666f4073742e636f6d301e170d3235303832373138313435325a17
0d3236303832383138313435325a304d310b3009060355040613024954310d300b06035504070c04
4c6568693111300f060355040a0c0853544d20496e632e310b3009060355040b0c024954310f300d
06035504030c0673742e636f6d3059301306072a8648ce3d020106082a8648ce3d03010703420004
c2a8159107bc66e14fcbacc1ad2659cbab7d0ea7400444ba4bb277d5d3cad6e3df2984cbb19c2e04
70744f0a7ed49a7c2fde852cca3d24ad192b8025cf6387d3300a06082a8648ce3d04030203480030
45022100d8f7eeb3ce70743aceb7c8a615a3f768b5fe053b28664d6e658bab1d67836041022070c0
0d900f75ab92c39791fc287767aab724f8307121d40167eb6ea8817568d0
OK
```

5 TLS/DTLS Handshake

5.1 Preamble

After providing cryptographic credentials, the ME can initiate the DTLS or TLS handshake, during which the client and server negotiate security parameters, authenticate each other using keys and certificates stored in the Secure Element, and generate session keys to establish a secure channel that ensures the integrity and confidentiality of IP communications. The ME also supports session resumption by storing session data in the ITS, allowing secure connections to be quickly re-established by reusing previously negotiated security parameters. This significantly reduces handshake time and lowers processing overhead.
Once the handshake is successfully completed, the ME can securely transmit application data through the established session. The session remains active until explicitly closed or timed out, at which point a new handshake is required to re-establish security parameters.

5.2 TLS Handshake and Session

5.2.1 Parameters

Here is an example of the AT command sequence to start the TLS handshake. The ME should have the cryptographic credentials stored in a secure storage ID, wait to connect to the network, and finally configure the IP layer. For detailed information on IP configuration and data transmission and reception commands, please refer to the document **ST87MXX_TCP_UDP_IP_Application_Note**.

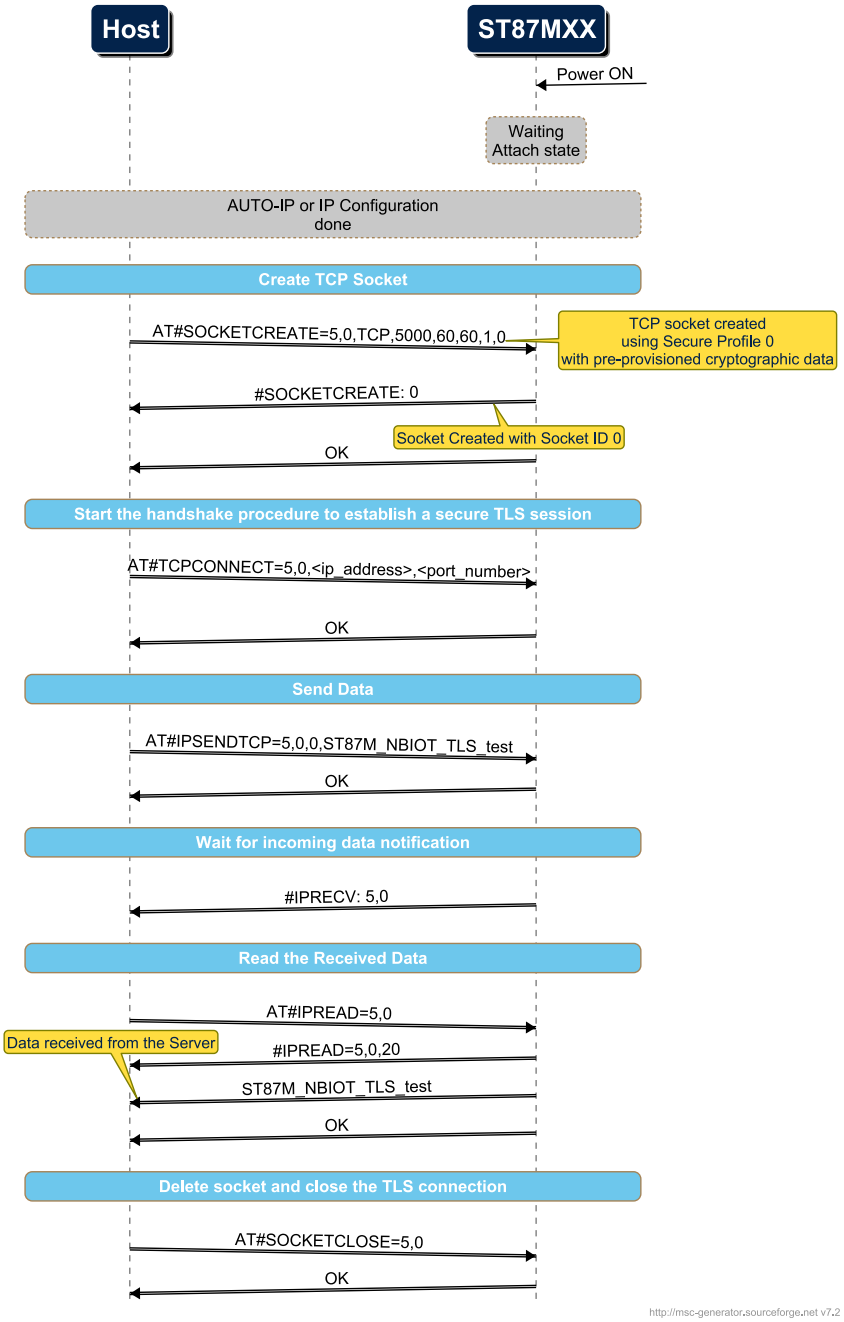
The parameters for creating the TCP socket linked to the configured security profile are as follows:

- **<context_id> = 5** to specify the context ID used for the IP connection.
- **<ip_version> = 0** for IPv4, 1 for IPv6.
- **<socket_type> = TCP** to specify the type of socket to create.
- **<local_port> = 5000** as the local port number used by the server to send back data.
- **<send_timeout> = 60** to set the timeout in seconds for sending the request to the server.
- **<receive_timeout> = 60** to set the timeout in seconds for receiving the response from the server.
- **<frame_received_urc> = 1** to enable URC notifications when TCP or UDP data is received from the network.
- **<security_profile_id> = 0** to specify the security profile ID to use.

The parameters for initiating the handshake and establishing a secure session are as follows:

- **<context_id> = 5** to specify the context ID used for the IP connection.
- **<socket_id> = 0** to specify the Socket ID received during socket creation.
- **<ip_address>** to specify the IP address of the remote server.
- **<port_number>** to specify the TCP port number of the remote server.

5.2.2 MSC



5.2.3 Terminal

Below is the sequence obtained in the UART terminal when executing the scenario (in **Bold** the commands sent to the module):

```
AT#SOCKETCREATE=5,0,TCP,5000,60,60,1,0
#SOCKETCREATE: 0
OK

AT#TCPCONNECT=5,0,192.168.21.4,4433
OK
```

```
AT#IPSENDTCP=5,0,0,ST87M_NBIOT_TLS_test
OK

#IPRECV: 5,0

AT#IPREAD=5,0
#IPREAD: 5,0,20
ST87M_NBIOT_TLS_test
OK

AT#SOCKETCLOSE=5,0
OK
```

5.3 DTLS Handshake and Session

5.3.1 Parameters

Here is an example of the AT command sequence to start the DTLS handshake. The ME should have the cryptographic credentials stored in a secure storage ID, wait to connect to the network, and finally configure the IP layer. For detailed information on IP configuration and data transmission and reception commands, please refer to the document **ST87MXX_TCP_UDP_IP_Application_Note**.

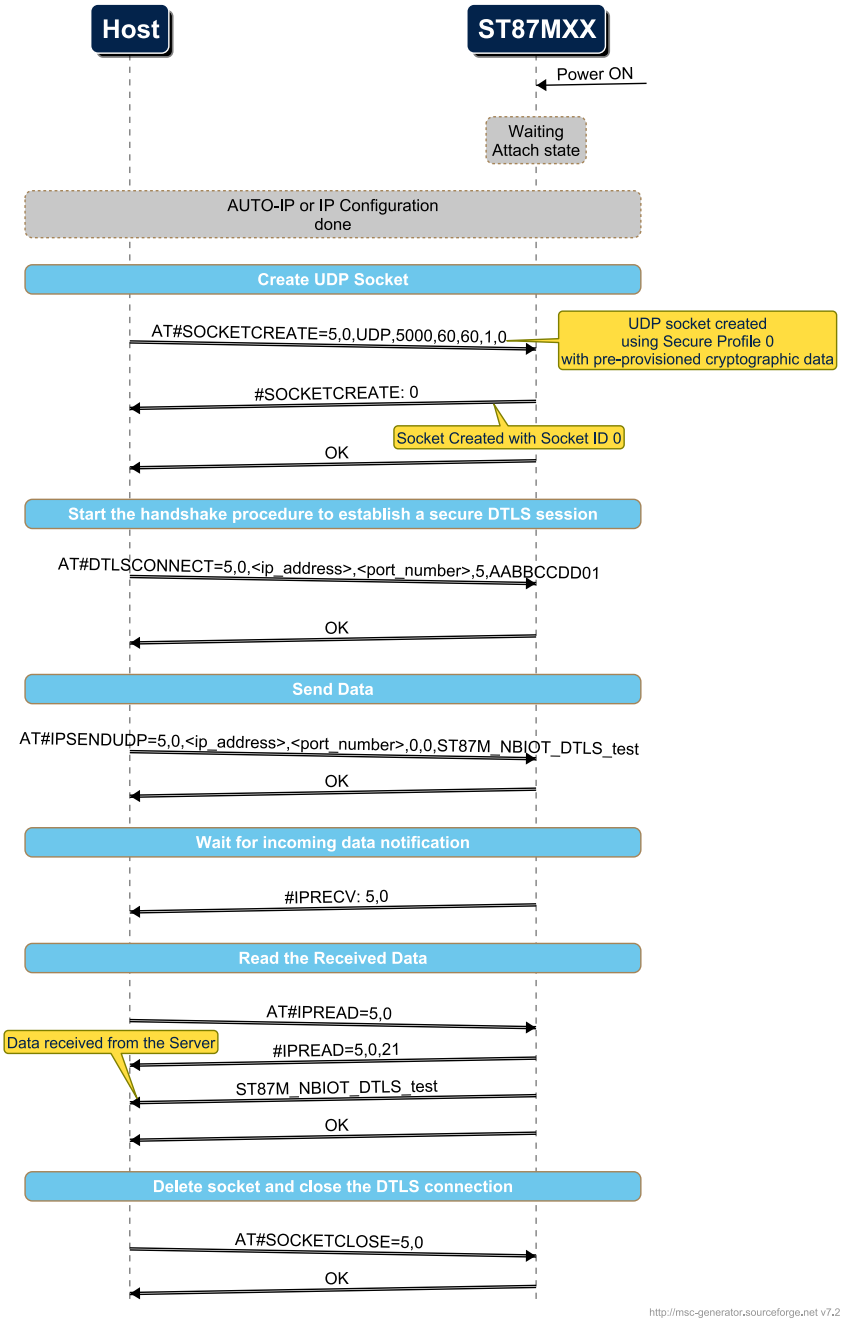
The parameters for creating the TCP socket linked to the configured security profile are as follows:

- **<context_id> = 5** to specify the context ID used for the IP connection.
- **<ip_version> = 0** for IPv4, 1 for IPv6.
- **<socket_type> = UDP** to specify the type of socket to create.
- **<local_port> = 5000** as the local port number used by the server to send back data.
- **<send_timeout> = 60** to set the timeout in seconds for sending the request to the server.
- **<receive_timeout> = 60** to set the timeout in seconds for receiving the response from the server.
- **<frame_received_urc> = 1** to enable URC notifications when TCP or UDP data is received from the network.
- **<security_profile_id> = 0** to specify the security profile ID to use.

The parameters for initiating the handshake and establishing a secure session are as follows:

- **<context_id> = 5** to specify the context ID used for the IP connection.
- **<socket_id> = 0** to specify the Socket ID received during socket creation.
- **<ip_address>** to specify the IP address of the remote server.
- **<port_number>** to specify the TCP port number of the remote server.
- **<cid_len> = 5** to specify the length in bytes of Connection ID data.
- **<cid> = AABBCCDD01** to specify the Connection ID as a hexadecimal string.

5.3.2 MSC



5.3.3 Terminal

Below is the sequence obtained in the UART terminal when executing the scenario (in **Bold** the commands sent to the module):

```
AT#SOCKETCREATE=5,0,TCP,5000,60,60,1,0
#SOCKETCREATE: 0
OK

AT#DTLSCONNECT=5,0,192.168.21.4,4433,5,AABBCCDD01
OK
```

```
AT#IPSENDUDP= 5,0,192.168.21.4,4433,0,0,ST87M_NBIOT_DTLS_test
OK

#IPRECV: 5,0

AT#IPREAD=5,0
#IPREAD: 5,0,21
ST87M_NBIOT_DTLS_test
OK

AT#SOCKETCLOSE=5,0
OK
```