Elvaco NB-IoT MCM
Integrator’s manual
English

V1.2

Contents

1 Document notes

All information in this manual, including product data, diagrams, charts, etc. represents information on products at the time of publication, and is subject to change without prior notice due to product improvements or other reasons. It is recommended that customers contact Elvaco AB for the latest product information.

The documentation and product are provided on an “as is” basis only and may contain deficiencies or inadequacies. Elvaco AB takes no responsibility for damages, liabilities or other losses by using this product.

1.1 Copyright and trademark

1.2 Contacts

Elvaco AB
Kabelgatan 2T
434 37 Kungsbacka
SWEDEN
Phone: +46 300 30250
E-Mail: info@elvaco.com

Elvaco AB Technical Support
Phone: +46 300 434300
E-Mail: support@elvaco.se

Online: http://www.elvaco.com

2 Using this manual

2.1 Purpose and audience

This manual provides information needed to integrate an Elvaco NB-IoT module into a system in order to receive meter data and be able to configure the device remotely. It mainly targets system integrators.

All Elvaco NB-IoT MCMs uses the same structure for both Device Management and Meter Data transport described in this general document.

In addition to the general specification, each NB-IoT module has a User's Manual, describing all the unique settings for that specific product.

2.2 Online resources

To download the latest version of this document, or to find information in other languages, please visit https://www.elvaco.com/.

2.3 Symbols

The following symbols are used throughout the manual to emphasize important information and useful tips:

The Note symbol is used to mark information that is important to take into consideration for safety reasons or to assure correct operation of the meter connectivity module.

The Tip symbol is used to mark information intended to help you get the most out of your product. It can for example be used to highlight a possible customization option related to the current section.

3 Introduction

3.1 Purpose

This chapter provides an initial understanding of the typical architecture setup for an Elvaco NB-IoT module. The following chapters provides all necessary details required in order to integrate the device with a device management system and a receiving meter data server.

3.2 Architecture

All Elvaco NB-IoT modules can be integrated with a meter data server and a device management server. The typical setup is illustrated in Figure 1.

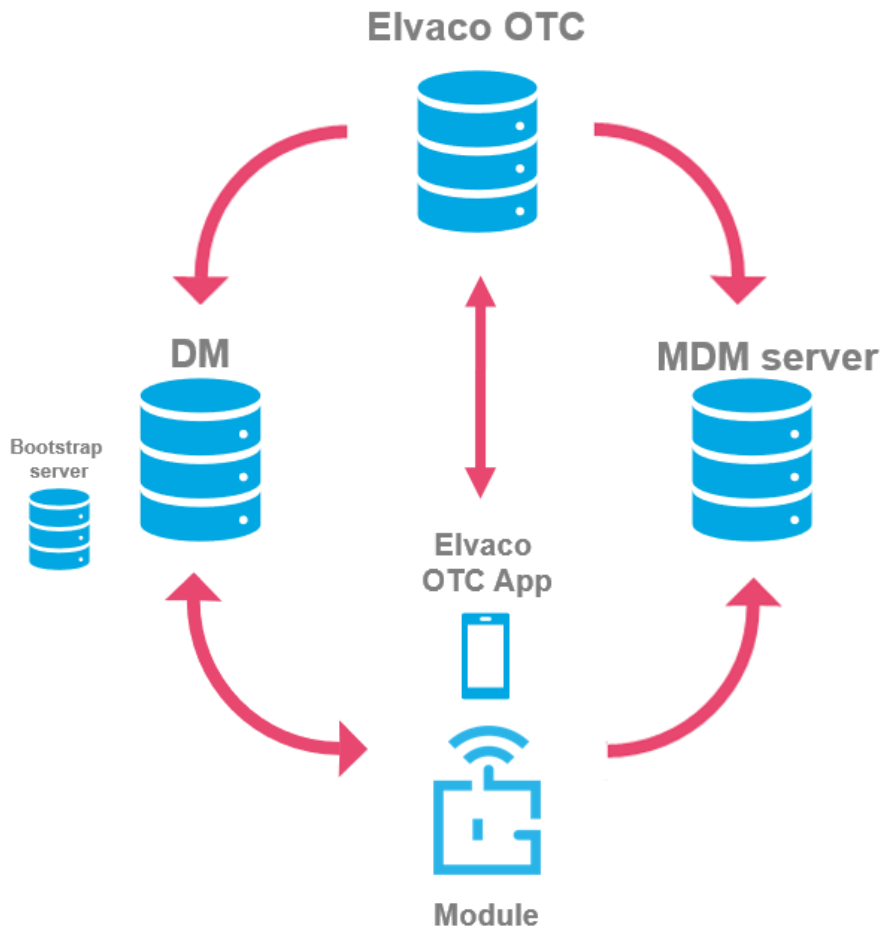


Figure 1: NB-IoT module integration setup

Below, each ingoing component is described.

- **Device Management (DM) system.** The Device Management system enables configuration and monitoring of an Elvaco NB-IoT MCM device remotely. This includes setting configuration parameters, update the firmware and trigger momentaneous/historical readouts of the module.
    - **Bootstrap server.** The Bootstrap server is the part of the Device Management system that is responsible for assigning keys and IP addresses to the module. When activated, the module will connect to the bootstrap server to receive security context

and IP addresses to the Device Management server (for configuration) and the Meter Data server (for meter data transport.)

- **Meter Data Management (MDM) Server.** The meter data server will receive all meter data sent from the module.
- **Elvaco OTC App.** The Elvaco OTC App is an Android mobile application (downloadable in Google Play), used to configure and activate the module at time of installation. Settings will be written to the device via the phone's NFC. Via the Elvaco OTC App, the user is able to set what bootstrap server to use, thus where the module will connect upon activation to receive keys and IP addresses.
- **Elvaco OTC web interface.** The Elvaco OTC web interface is used to assign ownership of devices and claim device keys. By logging in on the OTC account, the user is able to download Pre-Shared keys used for encryption between module and servers.

4 Integration guide

4.1 Purpose

This chapter provides the technical details needed to integrate an Elvaco NB-IoT module with a MDM and DM server. Note that this document is meant be used along with the Integrator's manual of the product you're using, where all device-specific settings are described.

4.2 Start-up

Elvaco NB-IoT modules uses UDP/IP for all communication on the NB-IoT network. For device management, the module will act as a LWM2M device connecting to a LWM2M 1.1 server [1]. For meter data transport, the module will use the MQTT-SN protocol.

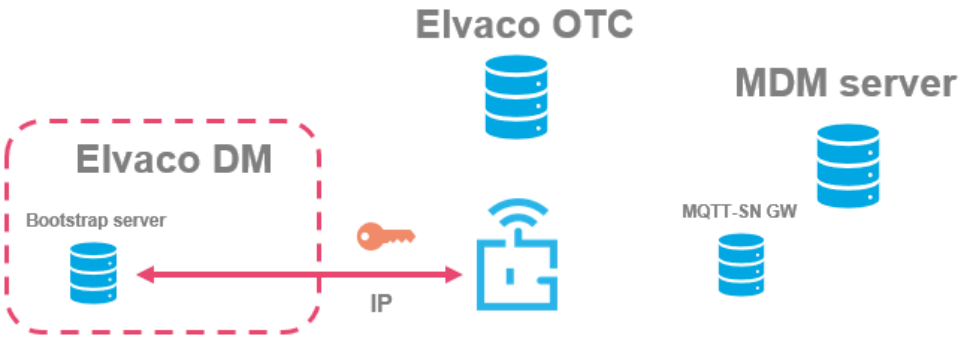
Upon activation, the device will attempt to connect to its configured LWM2M Bootstrap server [1, section 6.1.1] via the mobile (NB-IoT) network. When successful, the module will receive connection credentials, i.e. IP addresses to the DM and the meter data server and pre-shared keys used to encrypt the data between module and DM server/MQTT-SN Gateway.

The module will thereafter connect to the IP address of the MQTT-SN Gateway received from the Bootstrap server. When successful, the module and the MQTT-SN Gateway will use the DTLS pre-shared key to generate the session key used to encrypt the meter data transport.

The figures below illustrate the start-up procedure for Elvaco NB-IoT modules step-by-step.

Step 1: Pre-shared key of module is claimed via the Elvaco OTC web interface or APIs and added to the MQTT-SN Gateway.

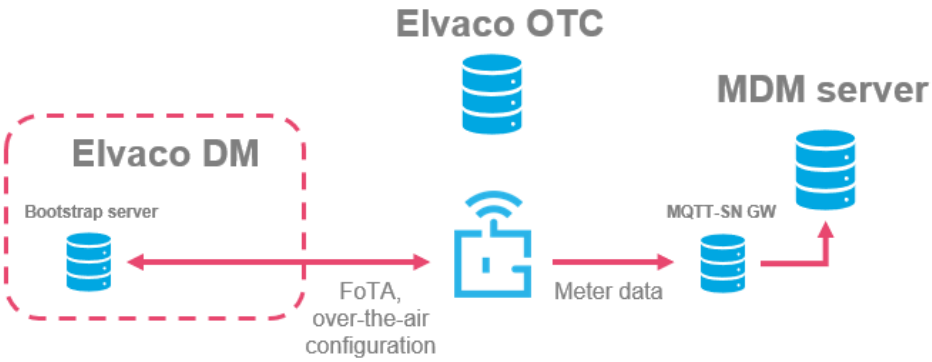
Step 2: The module is activated. It connects to the bootstrap server to receive the DTLS pre-shared keys and to receive IP addresses to the DM and MDM server.



Step 3: The module connects to the DM system to handshake a session key for module-DM encryption.

Step 4: The module connects to the MQTT-SN Gateway to handshake a session key used for encryption of the meter data transport.

Step 5: The end-device transmits meter data to the MDM server and transmits / receives status and configurations from the DM server.



4.3 Start-up using third party DM server

The start-up procedure when a non-Elvaco DM system will work in the same way described in section 3.1. Note that a third party DM server needs to be provisioned with the devices pre-shared keys in order for the Bootstrap server to be able to handshake a session with the device. These pre-shared keys can be accessed in Elvaco OTC after the devices are claimed.

4.4 Battery vs Power Supply

If the device is battery powered, the NB-IoT modem will enter PSM when not used. This means that the device will sleep and not be able to receive downlink commands, except during a short time window directly after an uplink transmission. In this case, the device will use LWM2M Queue mode.

If the device utilizes a PSU, the NB-IoT modem will stay in Idle mode when not used. This means that the device will be able to receive downlink commands at any time.

4.5 Bootstrap server

When the device has established a connection to the mobile NB-IoT network, it will connect to its configured LwM2M Bootstrap server to connection credentials. All modules will by default connect to Elvaco's Bootstrap server upon activation. This can be changed using the Elvaco OTC App [13]. Customers can also order pre-configuration of other Bootstrap servers.

The bootstrap server shall provide the device with one server instances for the DM server. The server object needs to be associated with a security object. By setting the Security mode to "0", the transport will be encrypted using DTLS. By setting the Security mode to "3", no encryption will be used.

The bootstrap server shall also provide the device with one instance of the Elvaco object "MDM Server provisioning the MDM server.

The security object and the MDM Server object contains a PSK for transport encryption that the bootstrap server must set if transport encryption is used. All devices contain a set of PSK that is pre-provisioned during the device production (see 5.6). The bootstrap server can optionally write the "magic" four byte key { 0xDE, 0xAD, 0xC0, 0xDE } to use the pre-provisioned PSK. The Elvaco bootstrap server will always write the "magic" PSK for the MDM server.

4.6 Security

Each module has a security chip where a device-unique set of keys are stored. These are provisioned to the module during production. The UDP transport can be secured using DTLS 1.2. There are three DTLS pre-shared keys in the module that are used for this purpose: one for the Bootstrap server, one for the DM and one for the MDM.

The DTLS pre-shared keys (PSK) are used to generate the session keys that encrypt the data sent between module and server. If a user wants to host his own DM system, the pre-shared keys of his devices will have to be downloaded from the Elvaco OTC server and added to the user's Bootstrap server.

The DTLS pre-shared keys are used to encrypt the meter data sent from the module to the MQTT-SN Gateway. If the user wants to host his own MQTT-SN Gateway, the pre-shared keys of his devices need to be downloaded from the Elvaco OTC server and added to the MQTT-SN Gateway.

The TLS 1.2 specification [8] recommends 24 hours maximum time [8, section F.1.4]. This is not optimal for a battery powered device that might only send uplink data once every 24 hours. In order maintain battery life on a battery powered device the server should have a session time of at least 20 days

4.7 Device Management over LWM2M

All Elvaco NB-IoT modules have support for remote configuration using LWM2M 1.1. The module will send a status report to the DM server once a day using the LWM2M 1.1 Send operation. Which resources that are sent are specified in the device-specific integrators manual.

For information about the configuration and status parameters, objects and the daily status report of a module, please refer to the device-specific documentation.

Firmware update

A firmware update of a module is performed according to the LWM2M 1.1 standard. The firmware is fetched using CoAP [5], where a standard CoAP file server supporting block-wise transfer is required. (The Californium Simple File Server demo application [10] can be used as a reference.)

The device supports PULL strategy only.

The firmware file is delivered encrypted and signed by Elvaco.

The device does not support DNS, hence the IP-number must be used in the Package URI.

Once the encrypted firmware file has been downloaded, it is verified and validated. If the verification and validation is successful, the module's firmware update state is set to "Downloaded". The server will then need to trigger the firmware update by executing the "Update" command.

When the firmware of the module has been updated, the device will perform a reboot and its embedded bootloader will decrypt the firmware file and reprogram the flash memory.

4.8 Meter data transport over MQTT-SN

MQTT-SN [7] stands for "MQTT for Sensor Networks". It is designed to be very similar to MQTT, but is better adapted to low-bandwidth, battery-operated application. The biggest difference is that MQTT-SN utilizes UDP for data transport instead of TCP.

Sequence

The first time the module connects to the MQTT-SN Gateway, it will send a CONNECT message with the Clean session flag set. A CONNACK is expected as a reply from the MQTT-SN Gateway.

If the device is configured with a custom topic, the CONNECT message will be followed by a REGISTER message which serves to register the topic. A REGACK is expected as a reply from the MQTT-SN Gateway.

When the device sends CONNECT it will use the DevEUI (as a 16-character capital hex string) as ClientID.

After the CONNECT message (and possibly a REGISTER message), a PUBLISH message will be used to send the meter data. The QoS flags will be to ”01” and the retain flag will be cleared. A PUBACK is expected as a reply from the MQTT-SN Gateway.

The Duration field in the CONNECT message will be set to the maximum value which is approx. 18 hours. When the configuration option "MDM Server Connection Config" is set to "Compliant" the device will make sure that the connection is not lost by sending extra CONNECT messages with the Clean session flag cleared before the duration times out.

The following times the device sends meter data the PUBLISH message is used. If the configuration option "MDM Server Connection Config" is set to "Compliant" the PUBLISH message will be preceded by an CONNECT message with the Clean session flag cleared, to make sure the gateway can match the Client ID with the correct IP-number and port (which might have changed if NAT is used).

If the module does no receive an acknowledge from the gateway it will try to resend the command with the DUP flag set.

The module will only send a new REGISTER message in two cases: 1) after a CONNECT message with Clean Session flag set, or 2) if the topic has changed.

Firmware versions earlier than 2.1.0 of the CMi6110 has a slightly different sequence where the CONNECT message with Clean session flag clear that precedes the PUBLISH message is omitted.

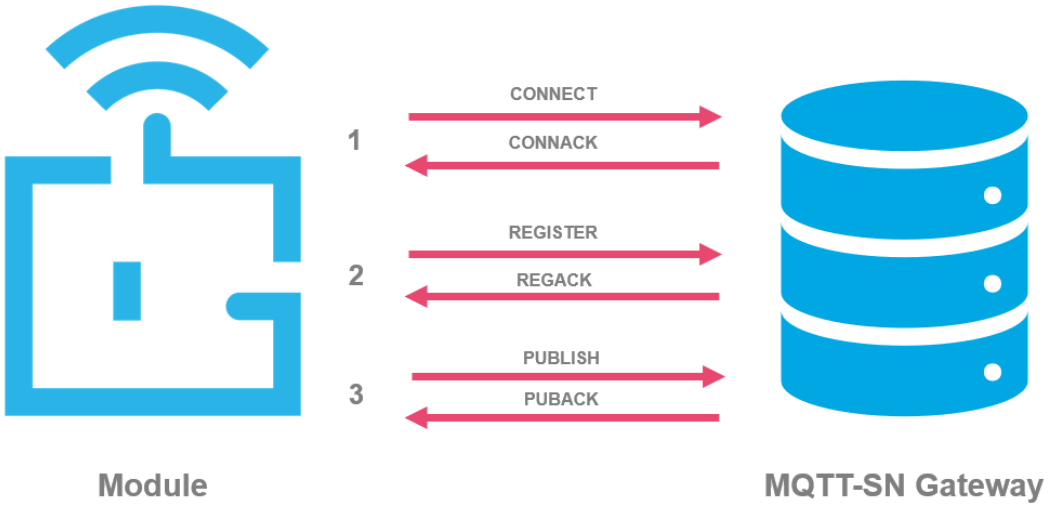


Figure 2: Connection procedure, Module - MQTT-SN Gateway

The module will only publish messages and not subscribe. Hence, it is not possible to send any downlink messages to the module over MQTT-SN.

The module will use QoS 1 (”At least once”).

Topic

A module will by default send data using the Short Topic “MD”. Optionally, a custom topic can be set by using the Elvaco OTC App, during bootstrap or using LwM2M. The custom topic will be then be shared with the MQTT-SN Gateway during the REGISTRATION.

By using the codes of Table 1 in the custom topic of the module, device-specific information can be added. For example, by setting the custom topic elvaco/nbiot/#D, the devEUI of each specific device will be added in the topic that is published to the MQTT broker.

Code	Topic information	Data type
#D	Device EUI	16-character Hex string
#M	Meter ID	8-character ASCII string
#E	Message encoding	ASCII string describing the message format. Can currently be "m-bus", "json" or "mlist".
#T	Message type	ASCII string describing the message type used, see product integration manual.
#P	Product name	Product name as ASCII string, e.g. "cmi6110"

Figure 3: Codes for custom topics

Payload

There are different encodings of the payload. The following encodings are currently available:

- JSON encoded values and units.
- M-bus encoded values and units
- SenML/CBOR with M-Bus encoded values and units.

The User's Manual specifies the encodings available for each specific product.

JSON payload

When the Payload is JSON encoded the payload consists of a single JSON object. The first character of the payload is '{' and the last is '}'. The object contains a value list of strings and numbers. The keys used are described in the Users's Manual but the timestamp field (key "TS") is always available. Example:

```
{
  "ID":87654321,
  "TS":"2019-11-28T20:00Z",
  "E":12345.678,
  "U":"MWh",
  "V":3456.7,
  "VU":"m3",
  "P":5012,
  "PU":"W",
  "F":212,
  "FU":"l/h",
  "FT":80.3,
  "TU":"C",
  "RT":53.8,
  "RU":"C",
```

```
  "EF":"0x4012"
}
```

M-Bus payload

When the Payload is M-Bus encoded the payload consists of a M-Bus record which is a list of DIBs.

Example (with the same content as the JSON example)

```
046D00147C2B 0C7821436587 04064E61BC00 041507870000 022B9413 023BD400 025A2303
025E1A02 02FD171240   (spaces added for clarity)
```

SenML + CBOR with M-Bus

For battery-powered devices it might be necessary to send several measurements in the same UDP frame to save energy. In order to achieve this SenML [13] + CBOR [6] is used to define a measurement list.

Each measurement consists of an M-bus record which may contain one or several DIBs (values). The M-bus record does not contain the timestamp DIB, nor the meter identification, but these are included in the SenML "Base Name" and "Base Time" / "Time" instead.

The most recent measurement is the first entry in the SenML list and the "Base Time" is used for this measurement. The following measurement uses a "Time" that is relative to the "Base Time".

The M-bus record might contain a different set of data for the different measurements. Typically the most recent measurement contains all data, but the other measurement contains only a subset of data to save space.

Example:

```
98 02                       # Array with length 2
   A3                       # Map with length 3
     21                     # Key 1 = -2 = Base name
     68                     # Value 1 = Text with length 8
        3132333435363738    # "12345678"
     22                     # Key 2 = -3 = Base time
     1A 5DE02740            # Value 2 = 1574971200 =
                            #   Time "2019-11-28T20:00Z"
     08                     # Key 3 = 8 = Data value
     58 21                  # Value 3 = Byte array, length 33
        04064E61BC0004150787000022B9413023BD400025A2303025E1A0202FD171240
                            #   Same record as M-bus example
   A2                       # Map with length 2
     06                     # Key 1 = 6 = Time
     39 0E0F                # Value 1 = -3600 =
                            #   Time "2019-11-28T19:00Z"
     08                     # Key 2 = 8 = Data value
     46                     # Value 2 = Byte array, length 6
        0406F24FBC00        #   M-bus record with one DIB:
                            #     Energy = 12341,234 MWh
```

5 References

5.1 Terms and abbreviations

Abbreviation	Description
CBOR	Concise Binary Object Representation
COSE	CBOR Object Signing and Encryption
DevEUI	Device Extended Unique Identifier
DM	Device Management
DNS	Domain Name Server
DTLS	Datagram Transport Layer Security
IP	Internet Protocol
LPWAN	Low Power Wide Area Network
LWM2M	Lightweight Machine to Machine
MDM	Meter Data Managment
MQTT	MQ Telemetry Transport
MQTT-SN	MQTT for Sensor Netoworks
NB-IoT	Narrowband Internet of Things
OSCORE	Object Security Constrained RESTful Environments
OTC	One-Touch Commissioning
PSK	Pre-Shared Key
PSM	Power Save Mode
PSU	Power Supply Unit
SenML	Sensor Measurement List
TLS	Transport Layer Security
UDP	User Datagram Protocol
URI	Universal Resource Identifier

5.2 Reference guide

1. LwM2M Core specification, v1.1.1 – 2019-06-17
   http://openmobilealliance.org/release/LightweightM2M/V1_1_1-20190617-A/OMA-TS-LightweightM2M_Core-V1_1_1-20190617-A.pdf
2. LwM2M Transport specification v1.1.1 – 2019-06-17
   http://openmobilealliance.org/release/LightweightM2M/V1_1_1-20190617-A/OMA-TS-LightweightM2M_Transport-V1_1_1-20190617-A.pdf
3. RFC6347 Datagram Transport Layer Security Version 1.2 – 2012-01
   https://tools.ietf.org/html/rfc6347
4. Connection Identifiers for DTLS 1.2 – 2019-10-21
   draft-ietf-tls-dtls-connection-id-07
   https://tools.ietf.org/html/draft-ietf-tls-dtls-connection-id-07
5. RFC7252 The Constrained Application Protocol (CoAP) – 2014-06
   https://tools.ietf.org/html/rfc7252
6. RFC7049 Concise Binary Object Representation (CBOR) – 2012-10
   https://tools.ietf.org/html/rfc7049
7. MQTT For Sensor Networks (MQTT-SN) Protocol Specification v1.2 – 2013-11-14
   http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf
8. RFC5246 The Transport Layer Security (TLS) Protocol Version 1.2 – 2008-08
   https://tools.ietf.org/html/rfc5246
9. Elvaco OTC App
   https://play.google.com/store/apps/details?id=com.elvaco.otc.app
10. Californium Simple File server demo application
    https://github.com/eclipse/californium/tree/master/demo-apps/cf-simplefile-server
11. mbedTLS Changelog, release 2.18.1, Features – 2019-06-11
    https://github.com/ARMmbed/mbedtls/blob/development/ChangeLog#L168
12. Californium DTLS 1.2 connection ID bypassing NATs – 2019-10-26
    https://github.com/eclipse/californium/wiki/DTLS-1.2-connection-ID-bypassing-NATs
13. Sensor Measurement Lists (SenML)
    https://tools.ietf.org/html/rfc8428

5.3 Revision history

Version	Date	Description	Author
V1.0	2019-11	First release	Niklas Granhage
V1.1	2020-02	- Updated MDM server provisioning. - Updated trigger to use pre-provisioned keys	Niklas Granhage
V1.2	2020-05	- Add SenML + CBOR with M-bus as MQTT-SN payload	Niklas Granhage