# OneEdge Advanced Guide

## Application Note

80710NT12045A Rev 0
2023-10-10
Released
Confidential

Technical Documentation

# Contents


**1 Applicability Table ..... 6**

**2 Introduction ..... 7**

2.1 Scope 7

2.2 Audience 8

2.3 Contact Information, Support 8

2.4 Conventions 8

2.5 Terms and Conditions 8

2.6 Disclaimer 8

**3 Application Architecture ..... 9**

3.1 Customer MCU (Hosted Architecture) 9

3.2 AppZone (Hostless Architecture) 10

3.3 Telit Module 10

Connection Agent 10

3.4 LwM2M Client 11

3.5 LwM2M Server 12

3.6 OneEdge Service 12

3.7 OneEdge IoT Portal 13

**4 LwM2M Overview ..... 14**

4.1 LwM2M Protocol Description 14

4.2 Definitions 15

4.3 Logical Interfaces 15

Bootstrap 15

Device Discovery and Registration 16

Device Management and Service Enablement 16

Information Reporting 17

4.4 CoAP Protocol 17

4.5 DTLS Protocol 18

4.6 LwM2M Data Model 18

Object 18

Instance 20

Resource 20

**5 LwM2M Registration Interface ..... 22**

5.1 Lifetime Parameter 22

5.2 NAT Environment and UDP Expire Time 23

Effect of NAT on LwM2M User Experience 24

5.3 Binding Parameter and Queue Mode 26

5.4 Wakeup SMS 28

5.5 How to Choose Lifetime According to Application Scenarios 29

5.6 OneEdge Direct 33

**6 LwM2M Registration Features in OneEdge ..... 36**

6.1 Bootstrap and Registration 36

AT#LWM2MENA 38

AT#LWM2MSTS 40

AT#LWM2MLIST 40

AT#LWM2MREG 41

AppZone APIs 43

Useful AppZone Links 43
6.2 Bootstrap and Registration retries 43
6.3 Registration Update 44
Sad Path 44
6.4 Best Practices for Host Applications 45
6.5 Additional LwM2M Configuration 47
AT#LWM2MINJKEYS 47
AT#LWM2MCFG 47
AppZone APIs 48
**7 Operations to LwM2M Resources in OneEdge ..... 49**
7.1 Operations on Client Side 49
AT#LWM2MW 50
AT#LWM2MSET 50
AT#LWM2MR 52
Floating Point Constrain for Float Type 52
AT#LWM2MGET 52
AT#LWM2ME 53
AT#LWM2MOBJSET 53
AT#LWM2MOBJGET 55
Single or Multiple Resource 56
CoAP BlockWise 56
AppZone APIs 58
7.2 Operations on Server Side 59
Writing 59
AT#LWM2MMON 60
Reading 61
Executing 62
**8 LwM2M Observations in OneEdge ..... 64**
8.1 Operations on Server Side 64
Observe Request 64
Observe Attributes 65
Cancel Observe 67
Strategies Related to Observations 67
Observation at Resource Vs Instance level 67
Observation Attributes Use Cases 70
8.2 Operations on Client Side 71
AT#LWM2MNFYLIST 71
AT#LWM2MNFYACKURI and AT#LWM2MNFYACKENA 72
AppZone APIs 74
**9 LwM2M Objects in OneEdge ..... 75**
9.1 Default LwM2M OMA Objects 75
Security (Object ID 0) 76
Server (Object ID 1) 76
Access Control List (Object ID 2) 77
Device (Object ID 3) 77
Reboot Behavior 78
Factory Reset Behavior 79
Battery Level and Battery Status Behavior 79
Current Time Behavior 80

NITZ 80
NTP 81
LwM2M Remote Setting 81
Connectivity Monitoring (Object ID 4) 82
Firmware Update (Object ID 5) 83
Location (Object ID 6) 85
Connectivity Statistics (Object ID 7) 86
Cellular Connectivity (Object ID 10) 86
APN Connection Profile (Object ID 11) 86
CoAP Configuration (Object ID 500) 86
9.2 OneEdge Telit Proprietary Objects 88
Telit Application Data (Object ID 33205) 88
Telit AT Command (object ID 33206) 89
Telit IDs (object ID 33207) 90
Connection Agent (Object ID 33210) 90
Telit Services Management (Object ID 33211) 91
9.3 Resources Managed Internally Vs Host Application 93
**10 LwM2M CUSTOM OBJECTS ..... 96**
10.1 OMA Objects and Custom Objects 96
10.2 How to Create a LwM2M Custom Object 96
OMA LwM2M Object Editor 96
How to Select the Operations of the Resources 99
How to Select the Resource Type 99
Default Value for the Custom Object Resources 100
Best Practice to Design LwM2M Data Models 100
10.3 Manage LwM2M Custom Objects 102
Operations on Client Side 103
AT#M2MWRITE 103
AT#M2MLIST 104
AT#M2MDEL 105
AT#LWM2MNEWINST 105
AT#LWM2MDELINST 106
AppZone APIs 106
Operations on OneEdge IoT Portal Side 107
Load New Object on OneEdge IoT Portal 107
Delete an Object from OneEdge IoT Portal 107
Create a New Instance from OneEdge IoT Portal Side 108
Delete an Instance from OneEdge IoT Portal Side 108
TR50 APIs 108
**11 OneEdge IOT Portal and Appcenter ..... 109**
11.1 Management Portal Application 110
Dashboards 111
Things 112
Table Tab 112
Creating a Things – Onboarding 113
Viewing a Thing 114
Map Tab 117
Connections 119
Developer 120

Thing Definition 120
Device Profile 122
Reads 123
Bindings 124
Application Servers 126
Triggers 127
Campaigns 130
API log and API Tester 130
Administration 131
11.2 Device Management Application 131
11.3 Firmware Management Application 133
11.4 Users 133
11.5 TR50 and REST API interfaces 134
**12 Typical OneEdge Applications ..... 135**
12.1 Battery-Powered Applications 135
Data Sending Through Observation 135
Data Polling Through Trigger 136
Remote Settings from the OneEdge IoT Portal 136
12.2 Always on Applications 137
Data Sending through Observation 137
Data Polling Through Trigger 138
Remote Settings from the OneEdge IoT Portal 139
**13 OneEdge Counters ..... 140**
13.1 API Usage, LwM2M Messages and Billing 140
**14 OneEdge Application Guidelines ..... 146**
14.1 Check-list for Designing the OneEdge Application 146
14.2 Use Case Examples 147
14.3 Good Practices 148
**15 Acronyms and Abbreviations ..... 149**
**16 Related Documents ..... 150**
**17 Document History ..... 151**

# 1 Applicability Table

Table 1: Applicability Table

| Products | Platform Version ID |
|---|---|
| ME910C1-AU<br>ME910C1-E1<br>ME910C1-E2<br>ME910C1-NA<br>ME910C1-NV<br>ME910C1-WW<br>ML865C1-EA<br>ML865C1-NA | 30 |
| ME910G1-W1<br>ME910G1-WW<br>ME310G1-W1<br>ME310G1-WW<br>ME310G1-W2<br>ML865G1-WW | 37 |
| LE910C1-EUX<br>LE910C1-SAX\SAXD<br>LE910C1-SVX<br>LE910C1-WWX\WWXD<br>LE910C4-WWX\WWXD | 25 |
| FN980m | 38 |
| FN990A28<br>FN990A40 | 45 |

# 2 Introduction

## 2.1 Scope

This document provides an overview of the advanced functionalities of LwM2M protocol and the main features of the OneEdge service. A guideline on how to get started using the OneEdge functionalities and resources describing the methods and available functions.

The table below summarizes all the chapters of this document and a brief description of them.

**Table 2: Summary of the chapters in this document**

| Chapter | Title | Description |
|---|---|---|
| General information about this document | | |
| 2 | Introduction | Basic information about this document and symbol convention |
| 3 | Application Architecture | Generic overview of the typical architecture of the customer applications and actors involved |
| LwM2M key concepts from OMA specs | | |
| 4 | LwM2M Overview | Basic LwM2M protocol concepts according to OMA specs: logical interfaces and data model |
| 5 | LwM2M Registration Interface | Detailed explanation of the “Registration” logical interface: registration and registration update, lifetime concept, NAT effect and constrains of the applications |
| LwM2M key concepts in OneEdge environment | | |
| 6 | LwM2M Registration features in OneEdge | Detailed AT commands to be used to manage the features of the “Registration” interface on the Telit LwM2M client and OneEdge |
| 7 | Operations to LwM2M resources in OneEdge | Detailed AT commands to be used to manage the data model on the Telit LwM2M client and OneEdge |
| 8 | LwM2M Observations in OneEdge | Details about “Information reporting” interface on the Telit LwM2M client and OneEdge |
| 9 | LwM2M objects in OneEdge | List and description of all the objects supported as default on the Telit LwM2M client and OneEdge |
| 10 | LwM2M Custom Objects | Management of LwM2M custom objects |
| OneEdge specific features | | |
| 11 | OneEdge IoT portal and AppCenter | Description of the features of the OneEdge IoT portal and web applications available |
| 12 | Typical OneEdge Applications | Description of typical strategies to send and receive data in OneEdge environment and supply modes |
| 13 | OneEdge Counters | Description of the counters related to the billing service |
| 14 | OneEdge Applications Guidelines | Guidelines and best practices to develop applications using Telit LwM2M client and OneEdge environment |

## 2.2 Audience

This guide is addressed to users that need to learn and start using the OneEdge functionality of Telit Module.

## 2.3 Contact Information, Support

For technical support and general questions, e-mail: TS-ONEEDGE@telit.com

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

Product information and technical documents are accessible 24/7 on our website: https://www.telit.com

## 2.4 Conventions

**Note:** Provide advice and suggestions that may be useful when integrating the module.

**Note:OneEdge:** Provide specific advice and suggestions related to OneEdge and valueble features of OneEdge compared to basic LwM2M protocol.

**Danger:** This information MUST be followed, or catastrophic equipment failure or personal injury may occur.

**ESD Risk:** Notifies the user to take proper grounding precautions before handling the product.

**Warning:** Alerts the user on important steps about the module integration.

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

## 2.5 Terms and Conditions

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

## 2.6 Disclaimer

THE MATERIAL IN THIS DOCUMENT IS FOR INFORMATIONAL PURPOSES ONLY. TELIT CINTERION RESERVES THE RIGHT TO MAKE CHANGES TO THE PRODUCTS DESCRIBED HEREIN. THE SPECIFICATIONS IN THIS DOCUMENT ARE SUBJECT TO CHANGE AT THE DISCRETION OF TELIT CINTERION WITHOUT PRIOR NOTICE. THIS DOCUMENT IS PROVIDED ON “AS IS” BASIS ONLY AND MAY CONTAIN DEFICIENCIES OR INADEQUACIES. TELIT CINTERION DOES NOT ASSUME ANY LIABILITY FOR INFORMATION PROVIDED IN THE DOCUMENT OR ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT DESCRIBED HEREIN.

TELIT CINTERION GRANTS A NON-EXCLUSIVE RIGHT TO USE THE DOCUMENT. THE RECIPIENT SHALL NOT COPY, MODIFY, DISCLOSE, OR REPRODUCE THE DOCUMENT EXCEPT AS SPECIFICALLY AUTHORIZED BY TELIT CINTERION.

TELIT CINTERION AND THE TELIT CINTERION LOGO, ARE TRADEMARKS OF TELIT CINTERION AND ARE REGISTERED IN CERTAIN COUNTRIES. ALL OTHER REGISTERED TRADEMARKS OR TRADEMARKS MENTIONED IN THIS DOCUMENT ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS AND ARE EXPRESSLY RESERVED BY TELIT (AND ITS LICENSORS).

# 3 Application Architecture

The main actors considered in this document are represented in the following image and described in next paragraphs.

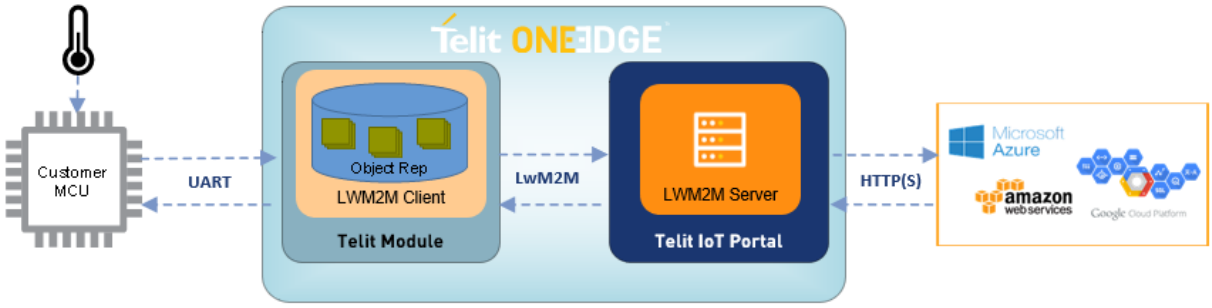



Figure 1: "hosted" high-level architecture.

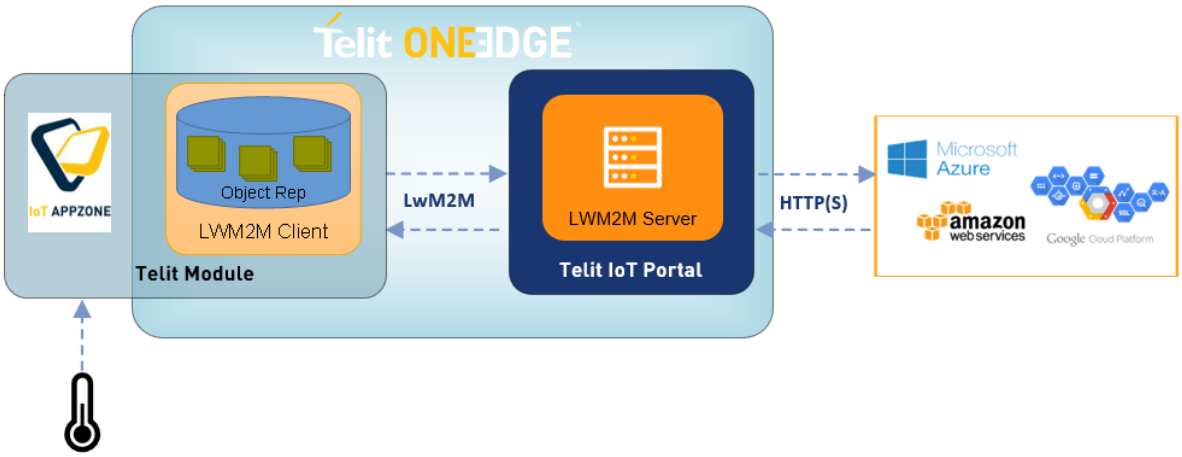



Figure 2: "hostless" high-level architecture

## 3.1 Customer MCU (Hosted Architecture)

In Hosted architecture the Customer's microcontroller interfaces with the Telit module via **UART, USB** or **SPI** and controls the module and interactions with the cellular network via AT commands.

All the customer application logic is located in the MCU, the Sensors are connected directly to the MCU. It is customer's responsibility handling the communication with the Sensors and to keep the synchronization with the representation contained in **LwM2M** objects. For this purpose, specific AT commands and URCs are provided (i.e. **AT#LWM2MR, AT#LWM2MW**).

It is also possible to connect a Sensor to the hardware interfaces within the Telit Module. Also in this case it is customer's responsibility to handle the communication with the external sensor using software interfaces available in the module (**read/write SPI, I2C** etc).

## 3.2 AppZone (Hostless Architecture)

In case of Hostless Architecture the Telit module provides both the connectivity and the operating environment for the on-module embedded application (i.e. using AppZone). The customer develops an application in C and controls the modem and interactions with the cellular network via AppZone APIs.

The Sensors are directly connected to the Telit modules through the integrated hardware interfaces (SPI, I2C etc.).

The AppZone environment provides the APIs to handle the LwM2M objects and to read/write from/to physical interfaces. The customer develops in AppZone both the application logic and the code to handle the Sensor communication.

Refer to https://www.telit.com/support-tools/telit-iot-app-zone/

And ts-emea@telit.com for more details about AppZone.

## 3.3 Telit Module

The Telit modules listed in applicability table are cellular modems with OneEdge software suite and security, which includes the security and personalization layer, the **LwM2M** client and data model, APIs and other OneEdge value features.

For more details about the features available on the Telit modules, refer to documents listed on chapter 16.

### Connection Agent

An interesting feature supported by default on Telit modules and integrated in OneEdge is the Connection agent.

The Telit Connection Agent provides a high level and easy to use interface to handle the connection to cellular network and automate most common operations involved in this process.

CA is integrated as a background application on the Telit module. The module exposes two interfaces for the user applications according to the system architecture: AT commands and AppZone APIs, see Figure 3.

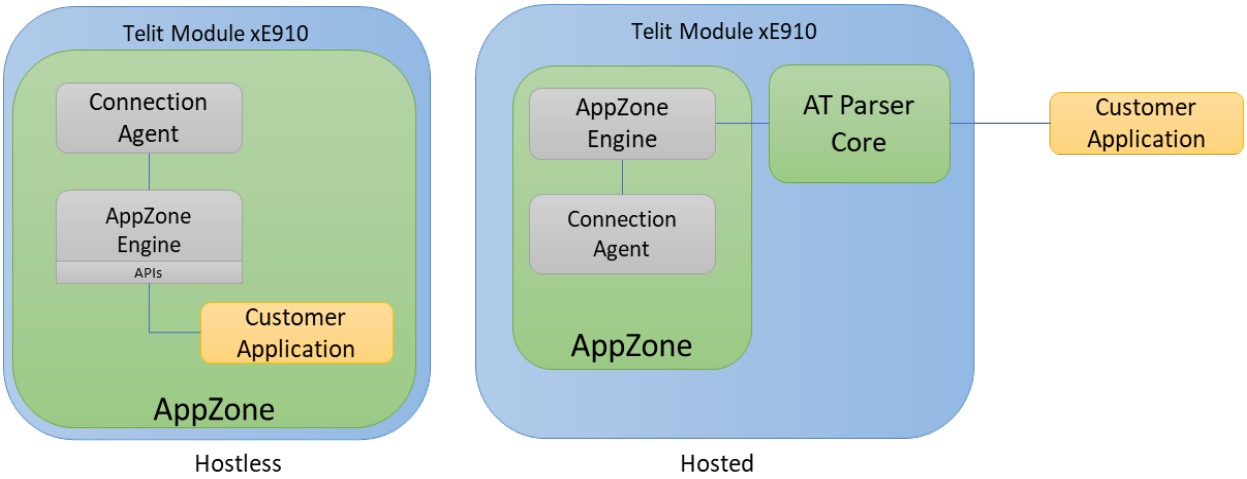



Figure 3: Possible Scenarios

Main goals of CA are to:

- Create an abstraction of network connection and provide **simple interfaces** and **configuration parameters** hiding the underlying complexity.
- Solve typical pains and issues related to **network connectivity**.
- Optimize the behavior of the module for a more efficient usage of local resources and network (e.g., reconnection timers and logic to prevent **aggressive** behavior).

The connection agent is based on Telit vast cellular experience and inspired by industry best practices and common guidelines.

The Connection Agent implements the following configurable policies:

- Always attached: PDP context always on.
- Always registered to a network
- On demand: open the PDP context (if needed agent must register before) only when the application needs to send/receive data.

Typical use cases are:

- Always attach for example, Vending machine with credit card payment. Keeps PDP context always open and ready for immediate credit card clearing.
- Always registered to a network: e.g., alarm panel which reports once a day. Device is registered to the network and opens a PDP when needs to report.
- On demand: Water meter (battery sensitive). Device powers on once a day, registers, opens a PDP context, transmits data, and powers off.

## 3.4 LwM2M Client

The LwM2M client is a process running in background responsible to handle all the functionalities related to LwM2M tasks.

Figure 4 shows a high-level architecture of the LwM2M client. The schema reports the main modules, each one is responsible of a specific functionality.

- Object Repository contains the LwM2M Object instances.
- LwM2M State Machine is the core component of the Client. This component is linked to all the others and manage the full path of the information.
- APIs in the module handle the communications with the user applications (via AppZone or AT commands).
- LwM2M Data Exchange is responsible for encoding/decoding the LwM2M messages coming from the Server and to be sent to the LwM2M server.

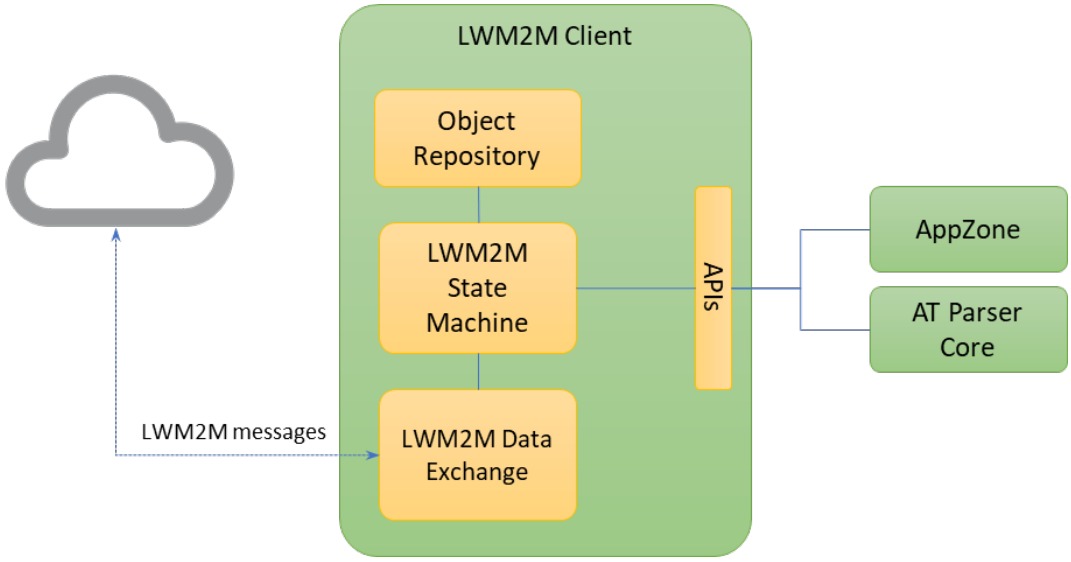



Figure 4: LwM2M Client Structure

**Note:** LwM2M version supported by Telit client is 1.0

## 3.5 LwM2M Server

The LwM2M server is a software service responsible of the communication with multiple LwM2M clients.

More details about LwM2M can be found in next paragraphs.

## 3.6 OneEdge Service

OneEdge is a Telit set of services and functionalities to simplify management of the IoT deployments through the LwM2M protocol.

The image below shows the most important components and services of OneEdge, that will be described across this document.

| Centralized Device Management | Edge Logic | Embedded Security | Connectivity Service (SIM) Management | Enterprise Integration |
|---|---|---|---|---|
| Leverage **zero-touch management** for your devices from one place and simplify life cycle management:<br>• Onboarding<br>• Provisioning<br>• Configuration management<br>• Monitoring<br>• Remote troubleshooting<br>• Software updates | Create edge device logic to **speed up deployments** and optimize behavior and resource use. OneEdge enables you to set control logic remotely and simplify edge life cycle management. | Ensure **end-to-end protection** through:<br>• Embedded, personalized security at the point of manufacture<br>• Encrypted device-to-cloud communications<br>• Server threat protection | Easily design and manage your service profiles and self-care subscription. Monitor and maintain your SIMs and data plans using **real-time diagnostics and controls**. | **Seamlessly integrate data** from each device across your existing enterprise applications and third-party business systems. |

Figure 5: OneEdge components

**Note:** Every time a specific **OneEdge feature** is involved in the document, added as **Note:OneEdge**

## 3.7 OneEdge IoT Portal

The Telit OneEdge is a SaaS portal with **Device Management** capabilities and with an integrated LwM2M server.

Across the OneEdge service, the Telit IoT Portal is responsible for providing a high-level representation of the devices and to show their data in different formats (charts and texts).

The management of the data and the events is made easy by triggers and campaigns that are special workflows on the portal, easy and fast to be developed.

Furthermore, the Telit IoT portal can interact with external Third-party services like Microsoft Azure, Amazon AWS, Google Cloud Platform and generic HTTP/HTTPS services through connectors and API interfaces.

# 4 LwM2M Overview

## 4.1 LwM2M Protocol Description

Lightweight M2M (LwM2M) is a standard protocol defined by the Open Mobile Alliance (https://www.omaspecworks.org/) for machine-to-machine (M2M) and Internet of Things (IoT) device management and service enablement.

For more details about LwM2M the OMA web site https://omaspecworks.org/what-is-oma-specworks/iot/lightweight-m2m-lwm2m/ is available.

LwM2M provides a **light** and **compact secure communication** interface along with an **efficient data model**, which together enables device management and service enablement for **M2M** devices.

Lightweight M2M enabler defines the application layer communication protocol between an LwM2M Server and an LwM2M Client, which is in an LwM2M Device. The diagram below shows the **LwM2M** architecture.

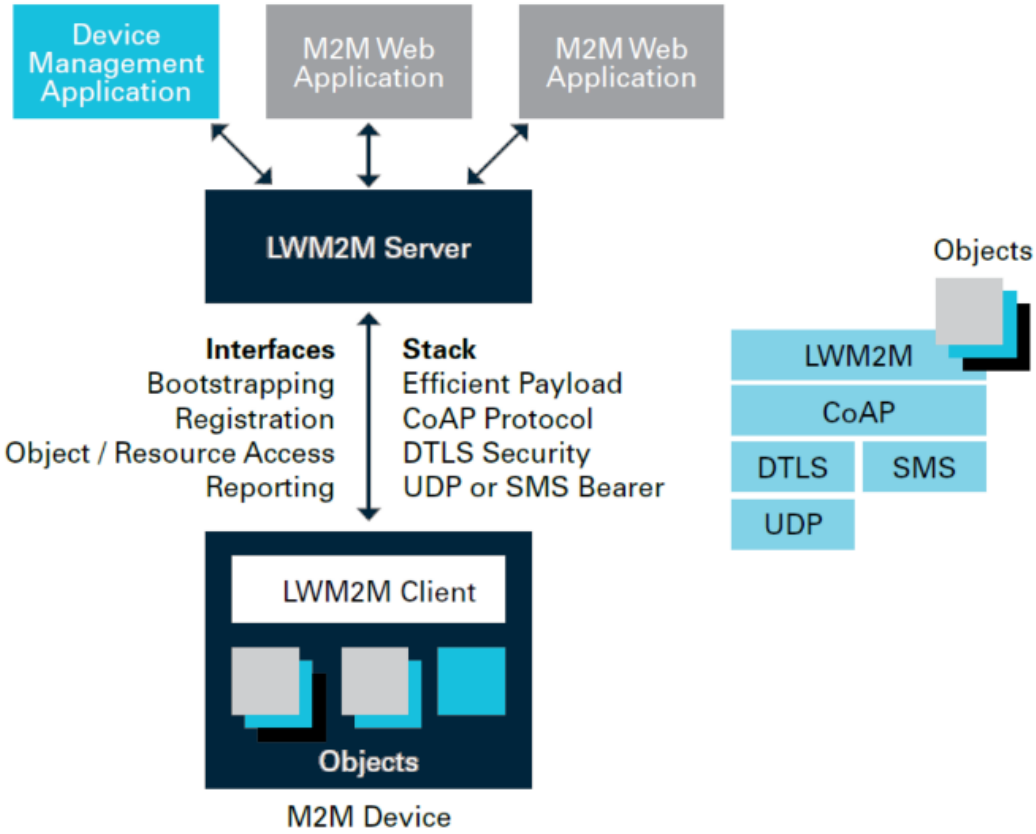



Figure 6: LwM2M Architecture Diagram

This architectural diagram describes four logical interfaces as the method of communication between the **LwM2M** Client and the LwM2M Server.

It also shows the overall communication stack being used by the LwM2M Client and Server.

In addition, it identifies the concept of a bootstrap and application servers used to define a system topology. Finally, the device information is contained in the LwM2M Client Objects and accessed by the LwM2M Server by invoking the appropriate URI mapping to the device's **Object/Instance/Resource (OIR)**.

## 4.2 Definitions

Below a brief description of the main items of this architecture:

- **Bootstrap Server**: It is a logical component that is contacted by the client during its first boot-up. Its only purpose is to initialize the data model, including parameters of LwM2M servers.
- **LwM2M Server**: A logical component residing within the OneEdge IoT Portal which serves as an endpoint of the LwM2M protocols, and it is used to remotely manage the device hosting the LwM2M client.
- **LwM2M Client**: A logical component residing in the Telit Module that serves as an endpoint of the LwM2M protocol and communicates with the LwM2M Server to execute the operations from the LwM2M Server for the device and the service management. The Client can also be controlled by the module SW via AT commands and AppZone APIs. LwM2M Client is uniquely identified independently of its network address by an Endpoint Client Name **- a URN** uniquely assigned to a device by its manufacturer.
- **LwM2M Device**: A LwM2M device is a device that runs LwM2M Client and communicates through the Network Service Provider's network. In our case it is the Telit modem.

## 4.3 Logical Interfaces

Here is a definition of the four logical interfaces defined between the server and client:

### Bootstrap

It provides the LwM2M bootstrap server the ability to provision the LwM2M client with keying, access control, and configuration of a device, which allows the LwM2M client to register with an LwM2M server.

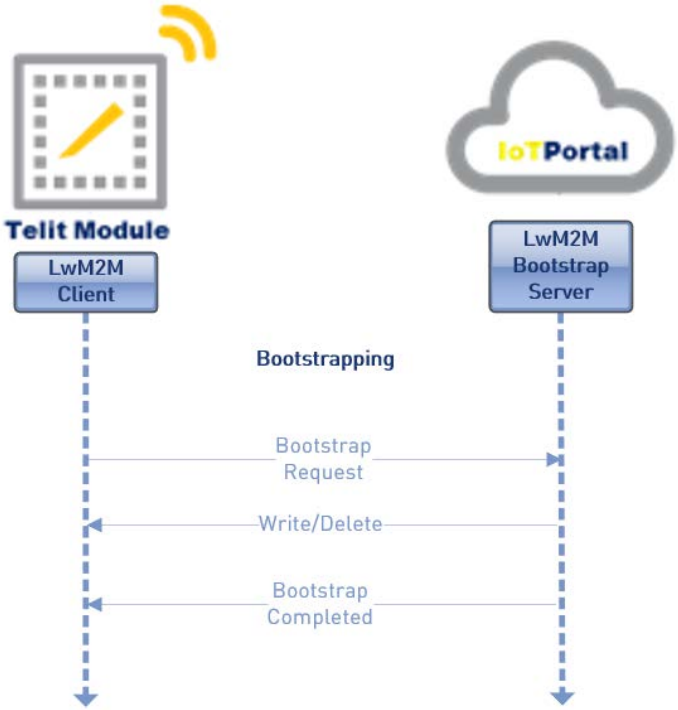



Figure 7: Bootstrap

## Device Discovery and Registration

It allows a LwM2M client to register with the LwM2M server and to report on the capabilities of the device to the LwM2M server.

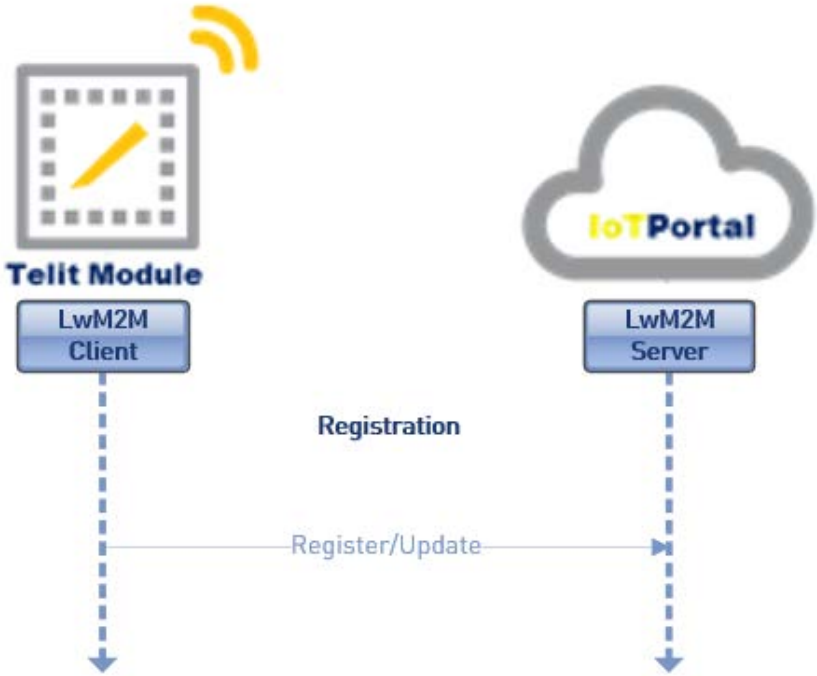



Figure 8: Discovery and registration

## Device Management and Service Enablement

It allows the LwM2M server to access the LwM2M client's OIR and perform the following actions on the device's **OIR: Read, Write, Execute, Create, Delete, Write Attribute, Discover**.

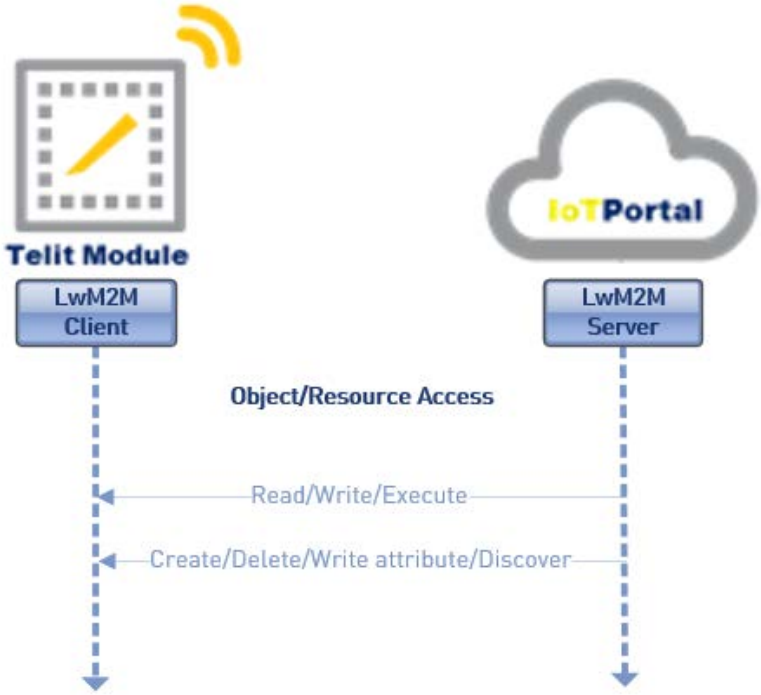



Figure 9: Object/Resource Access

## Information Reporting

It allows the LwM2M Client to report on periodical or event-based changes of the LwM2M values to the LwM2M server.

The server sends an **observation** request for a specific **URI** (resource, instance or object level) to the client, which must send a **notify** message every time the value of the observed URI changes.

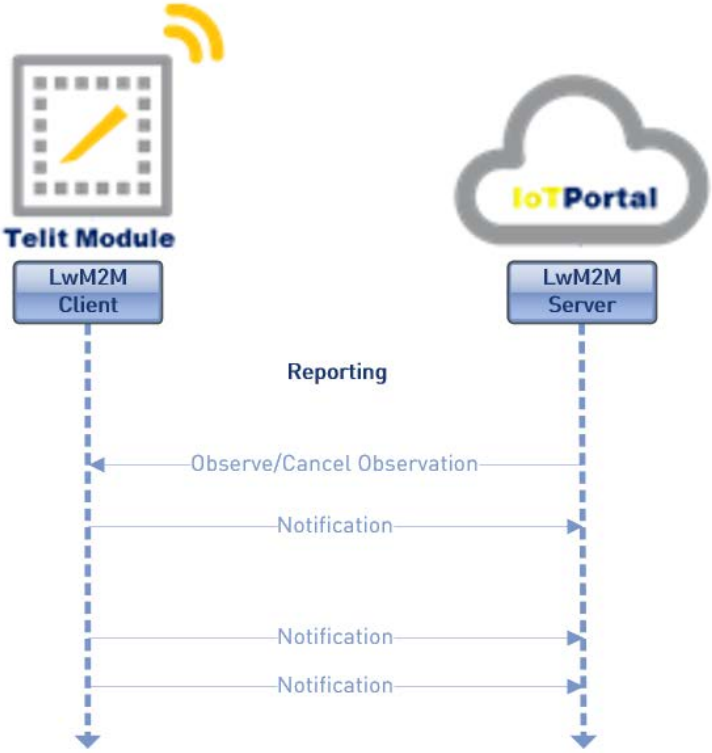



Figure 10: Information reporting

## 4.4 CoAP Protocol

The **Constrained Application Protocol** (CoAP) is a specialized Internet application protocol extremely important for Internet of Things (IoT) and **Machine-to-Machine** (M2M) devices.

The LwM2M communication model is based on CoAP methods such as **GET, PUT, POST,** and **DELETE** with bindings over UDP or SMS as transport layer. The binary encoded message overheads will only be a few bytes and the flat, simple objects with uniform URI across devices makes the protocol best suited for constrained device connectivity and easy management.

Even if **UDP** protocol is a connection-less protocol and doesn't guarantee any feedback or the reception of the messages, the **CoAP** protocol supports the ACK feature such that both **LwM2M** client and server are aware if the message is received at destination and, if not, manage retransmission and other strategies at LwM2M level.

## 4.5 DTLS Protocol

One important aspect is the protection of the exchanged data.

The **Datagram Transport Layer Security** (DTLS) is a communications protocol used to add the security protection to the communication between the Client and the Server to prevent eavesdropping, tampering, or message forgery.

The DTLS protocol is based on the stream-oriented **Transport Layer Security** (TLS) protocol applied to User Datagram Protocol (UDP) instead of TCP.

For more details about DTLS, the reference document is RFC 9147 (https://datatracker.ietf.org/doc/html/rfc9147).

> **Note:** DTLS version in Telit LwM2M client is **1.2**

DTLS on Telit LwM2M client supports only PreSharedKeys (PSK) and the key is unique for each device.

> **Note: OneEdge feature:** Every Telit LwM2M device is configured on production site with its own PSK key and OneEdge IoT portal retrieves the right key from a trusted procedure.
>
> In this way the right security parameters are provisioned to both LwM2M client and server without any further configuration: this means that, after the device is produced, it is ready to be switched on and easily managed by the portal.
>
> This specific OneEdge feature is called **OneEdge Zero-touch onboarding**.

The DTLS cipher-suite used during communications is based on the **TLS_PSK_WITH_AES_128_CCM_8**, as defined in the RFC 6655 (https://datatracker.ietf.org/doc/html/rfc6655).

## 4.6 LwM2M Data Model

The LwM2M data model is a well-organized data structure defined by three levels: Object, Instance and Resource. Each level is identified by a numerical identifier.

The object is a container of generic data elements which are the resources.

Inside an object, the same set of resources can be multiplied through the instances.

Resources within Objects are accessed through an URI

/{Object ID}/{Object Instance}/{Resource ID}.

For example, to access the Battery Level resource in the Device object, the URI is /3/0/9.

### Object

Each object is identified by a unique number between 0 and 65535 and defines a set of resources.

For example, the OMA LwM2M location object contains the following 6 resources:

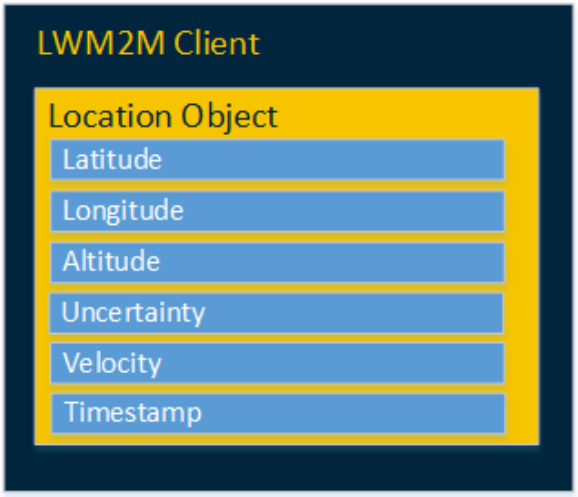



**Figure 11: Object example**

The object identifiers are assigned by the Open Mobile Alliance (OMA) that lists the standard IDs and manages the requests of vendors for specific ranges.

More details are available on OMA web site:
http://www.openmobilealliance.org/wp/omna/lwm2m/lwm2mregistry.html

The table 3 lists the items to define an object.

**Table 3: Object Definition**

| Item | Description |
|---|---|
| Name | Name assigned to the object |
| Object ID | Numerical identifier of the Object ( 0 to 65535) |
| Instances | Single or Multiple |
| Mandatory | Mandatory (must be supported by the lwm2m client)<br>Optional (could not be supported) |
| Object URN | The URN to be used to access the specific Object ( i.e. urn:oma:lwm2m:oma:3) |

The table 4 shows the object identifiers ranges allocated by OMA.

**Table 4: Object ID classes**

| Object ID range | Description |
|---|---|
| 0 - 1023 | Object produced by OMA. Only OMA can use this range |
| 1024 - 2047 | Reserved for future use. |
| 2048 - 10240 | Object registered by 3rd party standards organizations or alliances |
| 10241 – 26240<br>26241 – 32768<br>32769 - 42768 | Objects registered by companies or individuals.<br>Objects Produced by Vendors to re-use.<br>Block of objects reserved by vendors (Max 50). Private range, Objects will not be published. |

The LwM2M objects supported by the Telit LwM2M client are listed and described in chapter LwM2M Objects in OneEdge

## Instance

As per the structure explained before, each Object Instance of a given Object, supports the same set of Resources.

An object can be defined as **Single** or **Multiple** instance:

- **Single:** the object can have only one instance of the resources; it's the case of **Device** object because a LwM2M client must be referred to one device only
- **Multiple:** the object can have multiple sets of the same resources; for example, a LwM2M client could manage multiple instances of Temperature objects, to address multiple sensors connected to the same device

The Figure 12 below shows the comparison between a single-instance object (X) and a multiple-instance one (Y).

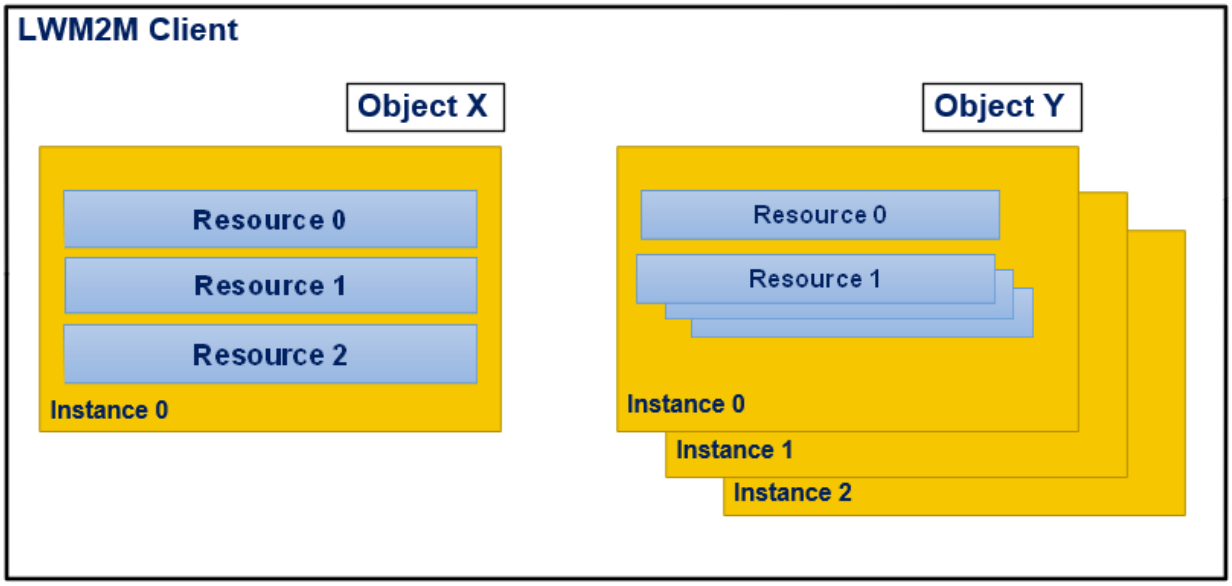



Figure 12: Object Model and comparison between single and multiple instance objects

In the Figure 12, object X is a single-instance object, so only one instance is present (Instance 0).

On the other hand, object Y is a multiple-instance object: the sensor value is contained into Resource 0 and multiple sensors can be addressed by multiple instances.

A resource can be single, like Resource 0, or multiple, as Resource 1 of the object Y.

## Resource

A resource into an object is defined by the following fields:

Table 5: Resources Definition

| Item | Description |
|---|---|
| ID | Numerical identifier of the Resource (0 to 65535) |
| Name | Name assigned to the Resource |
| Operations | Allowed activities on the specific resource on server side:<br>R – Read Only<br>W – Write Only<br>RW – Read/Write<br>E – Executable |

| Item | Description |
|---|---|
| Instances | Single or Multiple |
| Mandatory | Mandatory (must be supported by the lwm2m client)<br>Optional (could not be supported) |
| Type | Data type of the resource value (i.e. String , Integer etc.): refer to next Table 6 |
| Range or Enumeration | Range of accepted values for this resource |
| Units | Unit for the numerical value |
| Description | Detailed description of the Resource |

The following table describes the available data types (for more details refer to Appendix C of OMA specification document).

**Table 6: Types of Resources**

| Data Types | Description |
|---|---|
| String | A UTF-8 string. (i.e. "Hello world!"). |
| Integer | A 64-bit signed integer (i.e. -20). |
| Float | A 64-bit floating point value (i.e. 6.667). |
| Boolean | An 8 bit unsigned integer with the value 0 for false and the value 1 for true. |
| Opaque | A sequence of binary octets (i.e. "09AF") |
| Time | Unix Time. A signed integer representing the number of seconds since Jan 1st, 1970 in the UTC time zone (i.e. 1615911084 seconds since Jan 01 1970, represents: Tue, 16 Mar 2021 16:11:24 +0000) |
| ObjInk | The object link is used to refer an Instance of a given Object. The format is <objId><instId> where both are 2 bytes long |

The Figure 13 below shows a device that supports Temperature IPSO object with 3 instances.

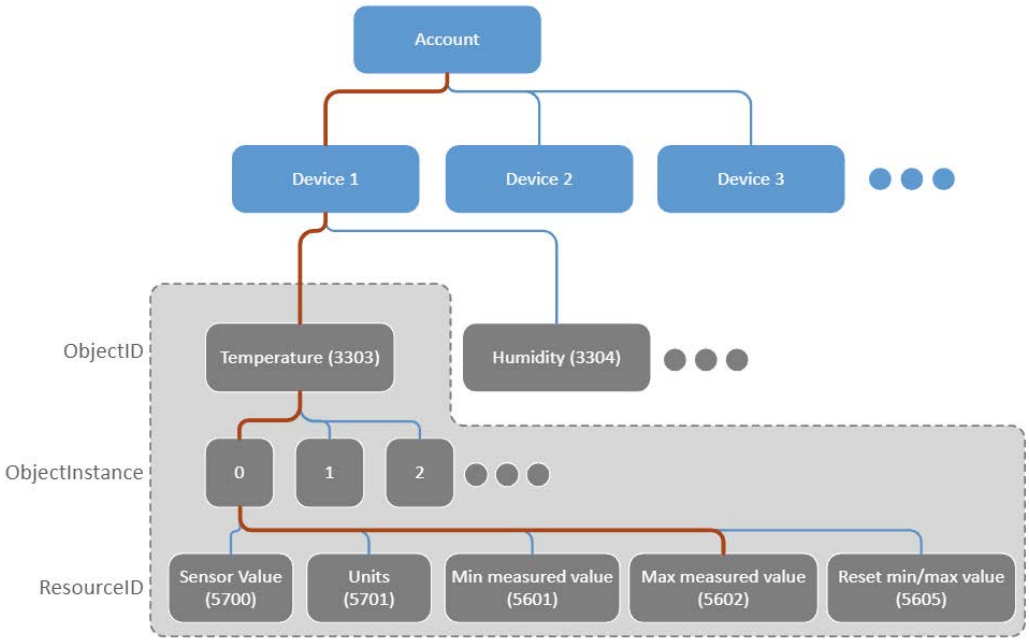



**Figure 13: Example of Temperature object with 3 instances**

# 5 LwM2M Registration Interface

In this chapter, the features related to LwM2M registration interface are described.

## 5.1 Lifetime Parameter

The lifetime is one of the most important parameters of the LwM2M protocol and it's addressed on resource 1 of **Server object (URI /1/x/1)**.

It's the number of seconds of validity of the current registration state of the **LwM2M** client into the server space.

The lifetime period starts when the **LwM2M** client performs the registration on the server.

The client must send a registration update before the lifetime expiration to keep the LwM2M registration alive.

When the server receives a registration update, the lifetime period restarts.

The registration update sending can be considered as a **keepalive** for the server and the User must configure the lifetime according to the application requirements.

If the server doesn't receive it within the agreed lifetime, the server considers the client as **disconnected** and the client must send a full registration again to re-establish the connection with the LwM2M server.

> **Note: OneEdge:** In OneEdge environment, the registered status is indicated through a green clip that becomes red when the device is disconnected.

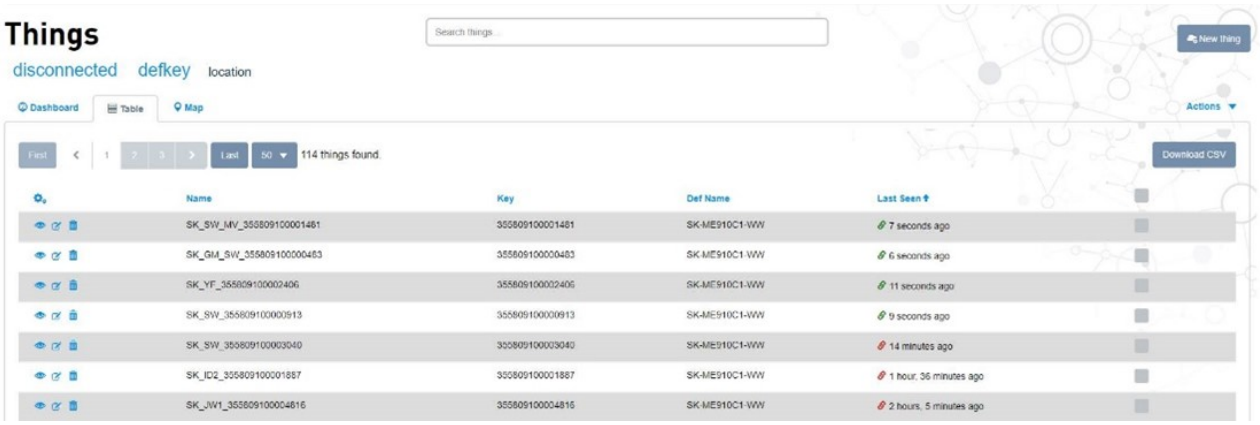



**Figure 14: list of things on OneEdge IoT portal: first 4 are connected, the last 3 are disconnected.**

The lifetime must be configured on LwM2M Bootstrap server side and can be modified writing a new value to Lifetime resource **(/1/x/1 URI)** from LwM2M DM server.

> **Note: OneEdge:** In OneEdge environment, the lifetime must be configured on OneEdge IoT portal into Application servers tab of the Device profiles section.

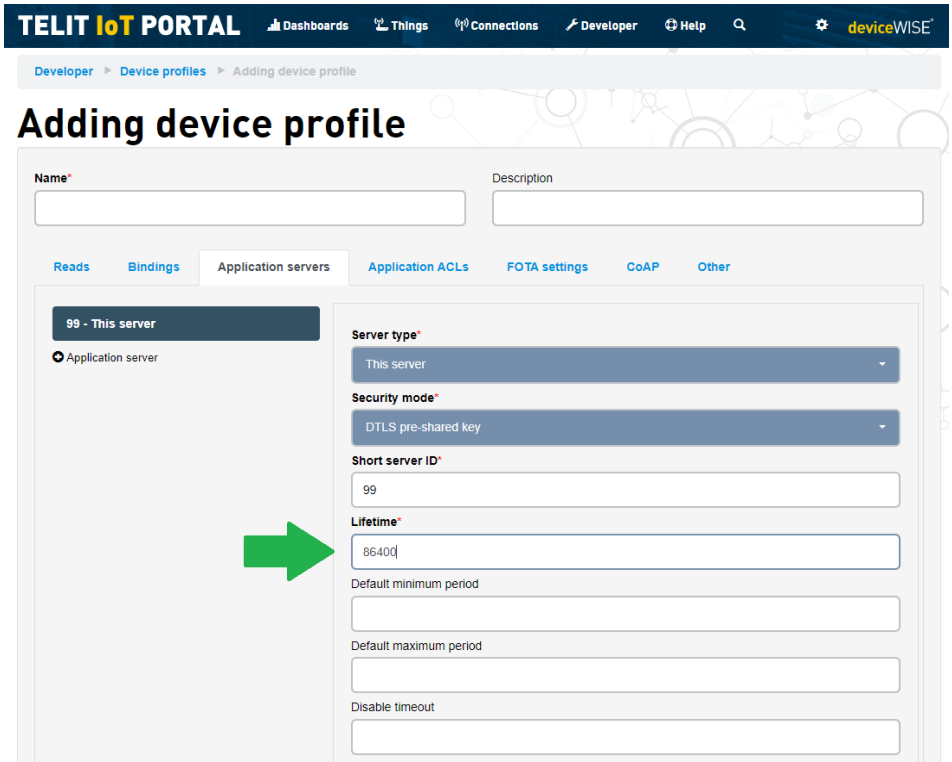



Figure 15: Application servers" tab of device profile configuration on OneEdge IoT portal

In this way, the LwM2M bootstrap server provisions lifetime and other parameters to the client during the bootstrap phase (refer to section Bootstrap and Registration).

The lifetime can be modified also at runtime writing the resource 1 of Server object (1/x/1).

**References on OMA Specifications:**

- OMA-TS-LightweightM2M-V1_0_2-20180209-A

## 5.2 NAT Environment and UDP Expire Time

The LwM2M user experience is negatively conditioned by the Network Address Translation (as called as NAT) behavior and configuration of the cellular network providers.

The NAT is a mapping that network providers usually apply to translate an IP address (and port) of a device into another IP address (and/or port), in order to masquerade the local address behind a public one.

The mapping between the addresses and ports is kept in a sort of database (NAT table) on a network provider entity where the entries usually have a **limited time validity**.

The generic representation of a NAT environment is explained in the Figure 16 below:

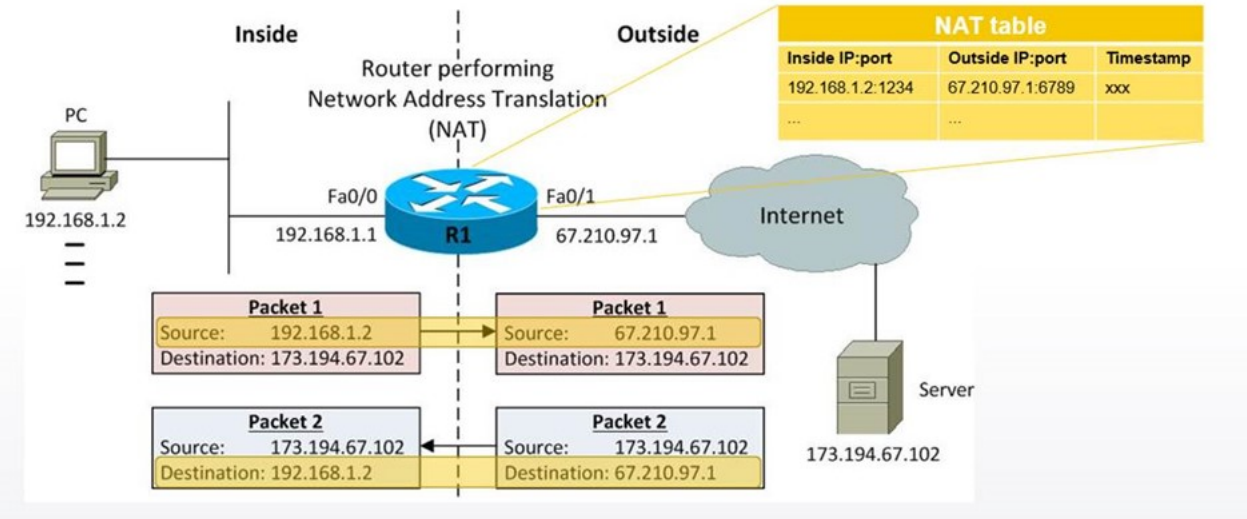



Figure 16: NAT Environment

**Reference:**

For more details about the NAT behavior, the reference RFC is 4787 (https://datatracker.ietf.org/doc/html/rfc4787).

# Effect of NAT on LwM2M User Experience

The validity of the UDP entries (which is the case of LwM2M protocol) is usually few minutes such that an entry is disabled or removed in case the addresses are not used within the validity timeout.

Referring to Figure 17 below, there are 3 cases:

**Outgoing messages:**

When the LwM2M client sends a LwM2M message to the server (events on t=0 on the flow chart), a new entry is created on the NAT table to translate the real address of the device into a masked one, so the **server receives** the LwM2M message **from a new IP address**.

If an entry is already present and it's valid, the entry is just renewed and the validity timeout is restored (event on t=T3 on the flow chart).

**Incoming messages:**

When the LwM2M server sends a LwM2M message to the client, the destination address is the masked one because it's the last known address of the device on server side:

a) If the entry is valid, the NAT translates the masked address into the real address of the device and the **communication is successfully** (event on t=T2 on the flow chart).

b) On the other hand, if the entry is not more valid or present because the timeout is expired, the NAT cannot find the real address of the device, so the **LwM2M message** from the server **cannot be delivered** and the client doesn't receive it (event on t=T4 on the flow chart).

**Note:** In case of LwM2M protocol, using "queue" binding mode, the undelivered messages are queued on the LwM2M server and delivered when the LwM2M client sends the next registration update (event on t=T5 on the flow chart).

For more details about the **Queue mode**, refer to Binding Parameter and Queue Mode.

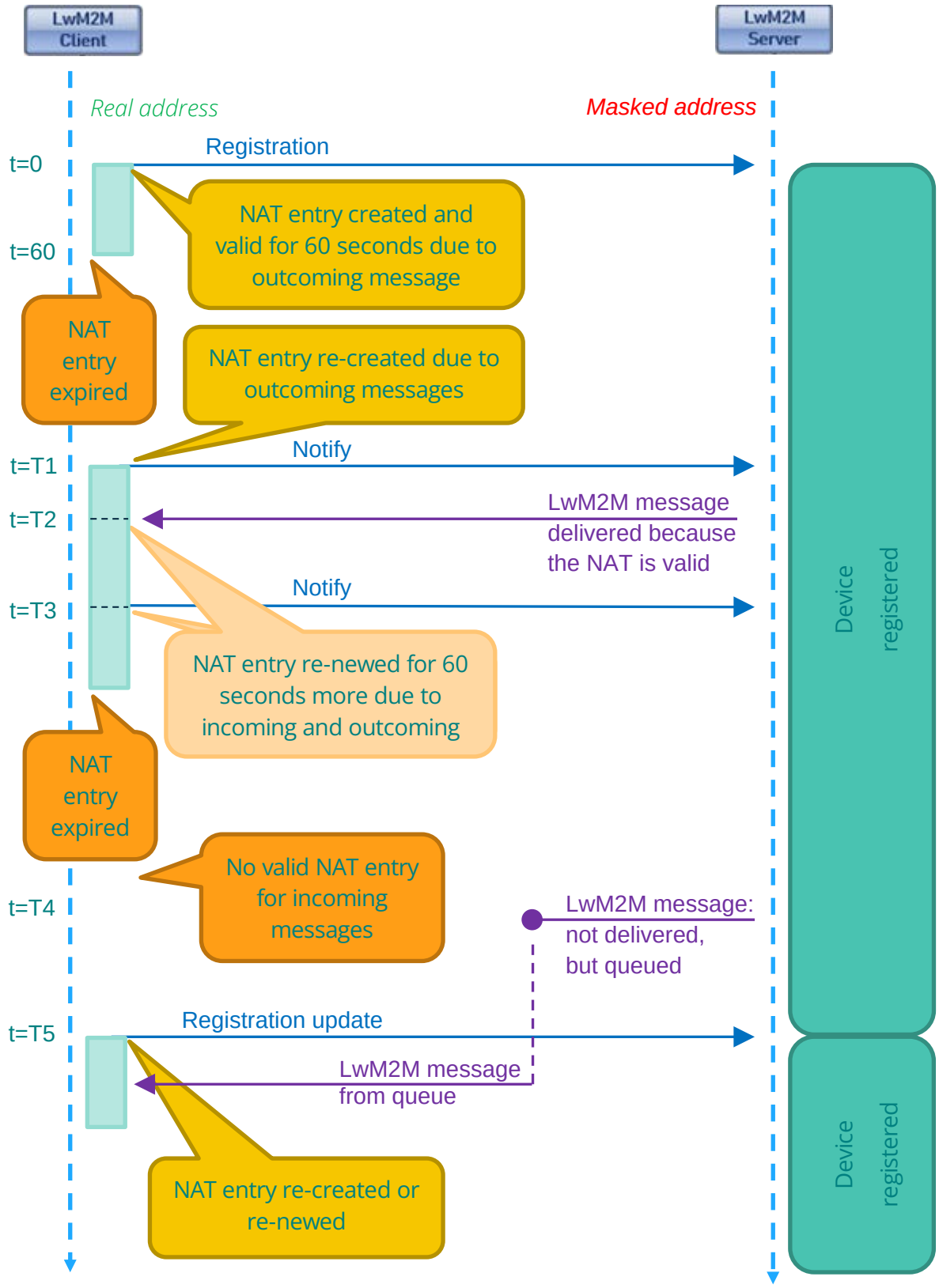



Figure 17: Three different scenarios in case of NAT of 60 seconds

Hence, according to the cases above, the NAT behavior can be considered **asymmetric** because the outgoing messages from the device are **always guaranteed** (case a), but the incoming messages **could be not delivered** immediately, but delayed instead (case bb).

For this reason, the NAT constrain doesn't affect the metering scenarios (i.e. sensor data from device to the LwM2M server) but only the "**remote settings**" scenarios.

**Note:** Some network providers could have different settings and different management related to NAT.Furthermore, some network providers could offer alternative paid options to avoid the NAT, for example fixed public IP addresses and ports, VPN, etc: refer to your network operator.

**Note: OneEdge:** Since the NAT constrains could limit the customer scenarios, Telit offers a no NAT service on Telit Next connectivity called OneEdge Direct functionality. Refer to OneEdge Direct for more details.

## 5.3 Binding Parameter and Queue Mode

The Binding resource of the LwM2M Server Object (URI 1/x/7) is related to the transport preferences for the communication session between the LwM2M Client and LwM2M Server.

According to the LwM2M OMA specification, the possible binding options are listed below:

**Table 7: list of supported bindings on Telit LwM2M client**

| Binding | Behavior |
|---|---|
| U | The LwM2M client and server communicates through UDP binding at any time. |
| UQ | The LwM2M client and server communicates through UDP binding at any time.<br>Anyway, if the device is not reachable, the LwM2M server **queues** all the requests and delivers them when the device comes back on-line (registration update) |

Refer to Figure 17 in the previous paragraph for an example of queued message (events on t=T4 and t=T5 on the flow chart).

**Note:** In the worst case, a LwM2M message from the server to the device (i.e. a remote setting) could be effectively applied on device side by a maximum delay equal to the lifetime: this could be a constrain to be considered in the application logic.

**Warning:** If the lifetime is expired, the server does not try to send any LwM2M message to the client, so it is lost and not stored in the server queue.

The Binding parameter must be configured on LwM2M Bootstrap server side and can be modified at run-time writing a new value to Binding resource (/1/x/7 URI) from LwM2M DM server.

**Note: OneEdge:** In OneEdge environment, the Binding must be configured on OneEdge IoT portal into Application servers tab of the Device profiles section.

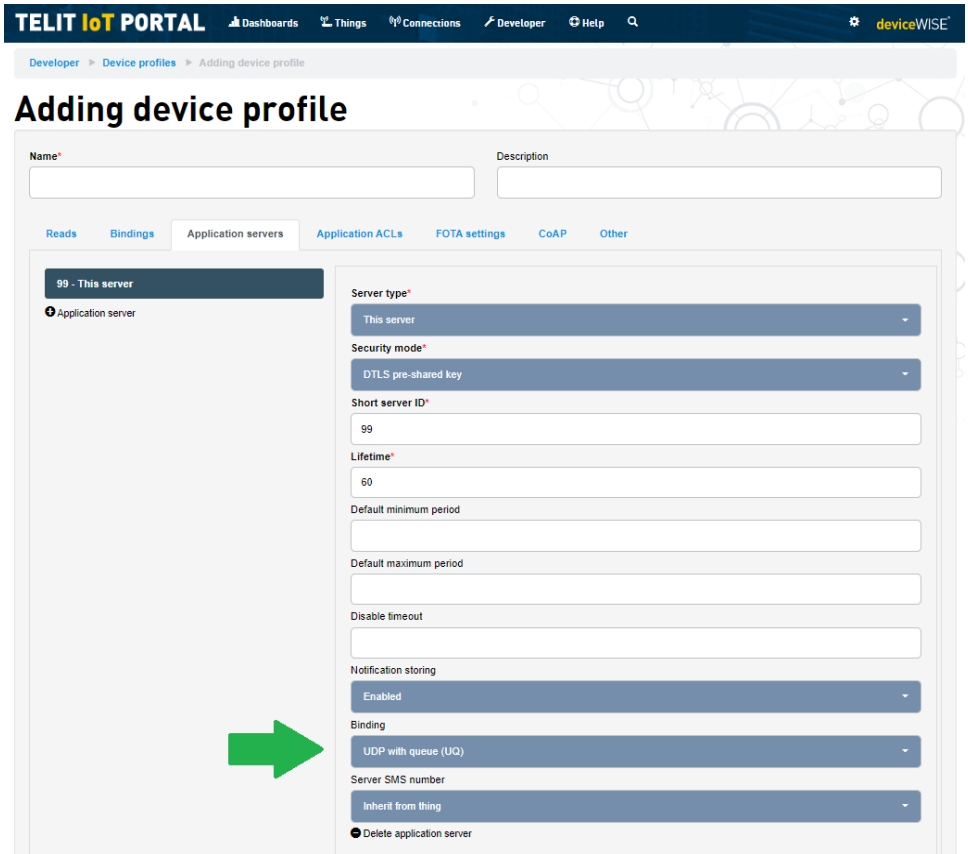



Figure 18: "Application servers" tab of device profile configuration on OneEdge IoT portal

In this way, the LwM2M bootstrap server provisions Binding and other parameters to the client during the bootstrap phase (refer to Bootstarp and Registration).

The Binding can be modified also at runtime writing the resource 7 of Server object (1/x/7).

**Note: OneEdge:** OneEdge IoT portal provides a special functionality to allow the portal User to force the LwM2M client to send a registration update **on demand** such that the queued messages are delivered as soon as possible: it is the **Wakeup SMS** described in next paragraph.

Furthermore, also the host application could force the **LwM2M** client to send a registration update before the scheduled time through the **AT#LWM2MREG** command. Refer to AT#LWM2MREG

**References on OMA specifications related to Binding and Queue mode:**

- OMA-TS-LightweightM2M-V1_0_2-20180209-A
- Section 5.3.1.1 Behavior with Current Transport Binding and Mode

- Section 8.3 Queue Mode Operation
- Section 8.6 Transport Bindings
- Appendix E.2 LwM2M Object: LwM2M Server

## 5.4 Wakeup SMS

According to LwM2M OMA specification and **SMS** binding option, LwM2M client must send a registration update on-demand when a special SMS is received.

The SMS must follow a specific format and is called **Wakeup SMS**.

This feature is very useful in **remote settings** scenarios, when a device, even if it's seen as **registered** on LwM2M server side, is not able to receive incoming LwM2M messages immediately, for example due to NAT entry expired.

In this case the User can asks for the client to send a registration update immediately without waiting for next scheduled one: the device becomes **online** and receive incoming LwM2M messages at runtime or receive the queued messages.

> **Warning:** The SMS could have an additional cost.

Furthermore, according to **LwM2M OMA** specification, the LwM2M client must accept SMS only from trusted SMS sender.

The number of the trusted SMS sender must be configured on LwM2M Bootstrap server side on **LwM2M Server SMS Number** resource of Security object (/0/x/9 URI) and cannot be modified from **LwM2M DM server**.

> **Note: OneEdge:** In OneEdge environment, the SMS senser number must be configured on OneEdge IoT portal into Application servers tab of the Device profiles section

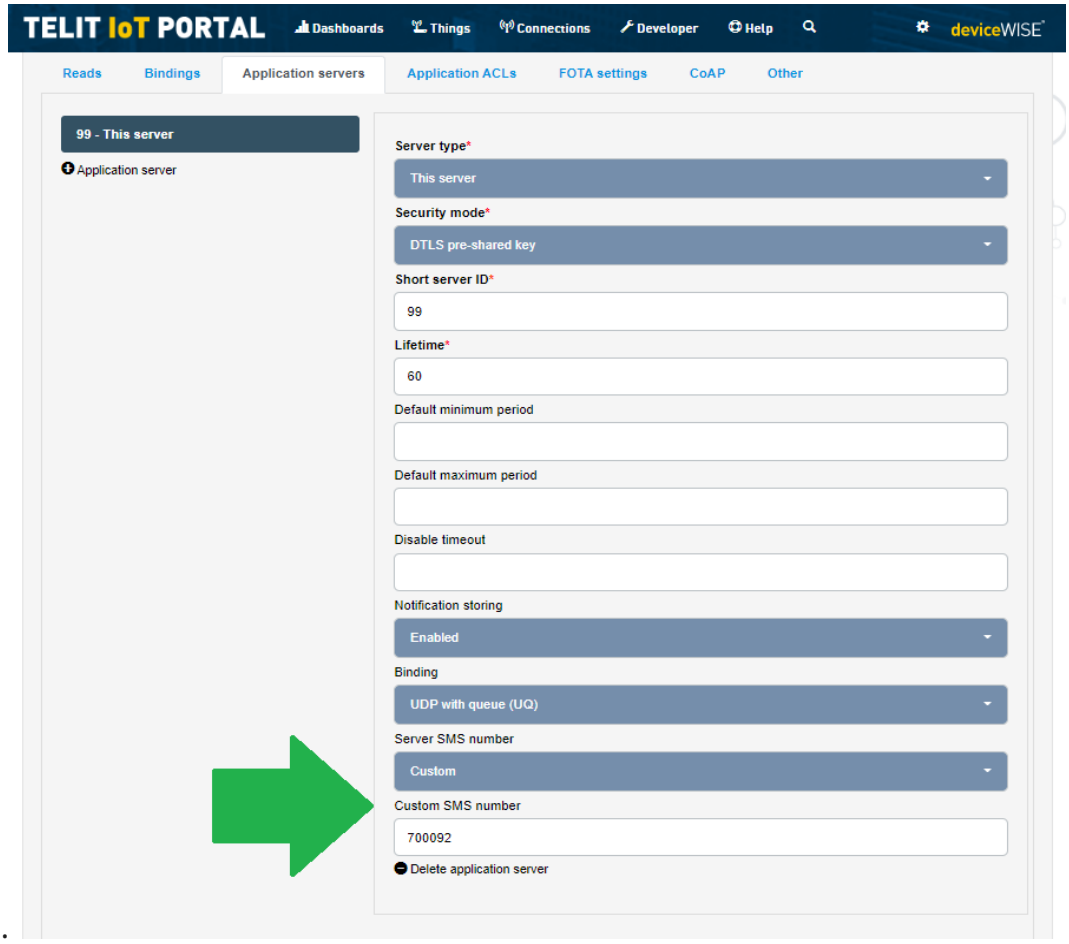



Figure 19:Application servers tab of device profile configuration on OneEdge IoT portal

In this way, the LwM2M bootstrap server provisions this parameter to the client during the bootstrap phase, refer to section Bootstarp and Registration.

**Warning:** Since the SMS server number resource is on Security object and so it cannot be modified at runtime from DM server, a new SMS sender can be applied only modifying the device profile and then performing a new Bootstrap procedure.

**References on OMA specification related to Wakeup SMS:**

- OMA-TS-LightweightM2M-V1_0_2-20180209-A
- Paragraph 8.4
- Paragraph 8.6.2
- Appendix L

## 5.5 How to Choose Lifetime According to Application Scenarios

Some typical application scenarios are listed below, considering the relationship between the value of the lifetime and some typical User constraints:

a If the device is kept always powered on and the LwM2M client is active all the time:

The lifetime value defines **how often** the LwM2M client will send the **registration update** to the server:

lifetime = keepalive_frequency

Refer to events on t=lifetime x1 and x2 on the Figure 20 below.

**Note:** To reduce the amount of data transmitted, the typical lifetime can be pretty high, for example, a few hours, 1 day or 1 week.

If a device is unable to connect, it will be considered as “disconnected” only after the lifetime expiration.

So, the lifetime defines the **maximum timeout** that the User can accept to detect a device that is **not communicating as expected**:

Disconnect_timeout <= lifetime

Refer to **disconnection delay** on the Figure 20 below.

If the User sends a LwM2M command to the device through the server, but the device is not able to receive it, the undelivered message is stored on the queue of the LwM2M server and delivered at next registration update. In this context, a device with high lifetime will receive the LwM2M command with a delay that, in the worst case, is equal to the lifetime.

So, the lifetime can be chosen as the maximum delay accepted by the User to apply LwM2M commands to the device (Refer to Binding Parameter and Queue Mode) delivery_delay <= lifetime

Refer to **delivery delay** on the Figure 20 below.

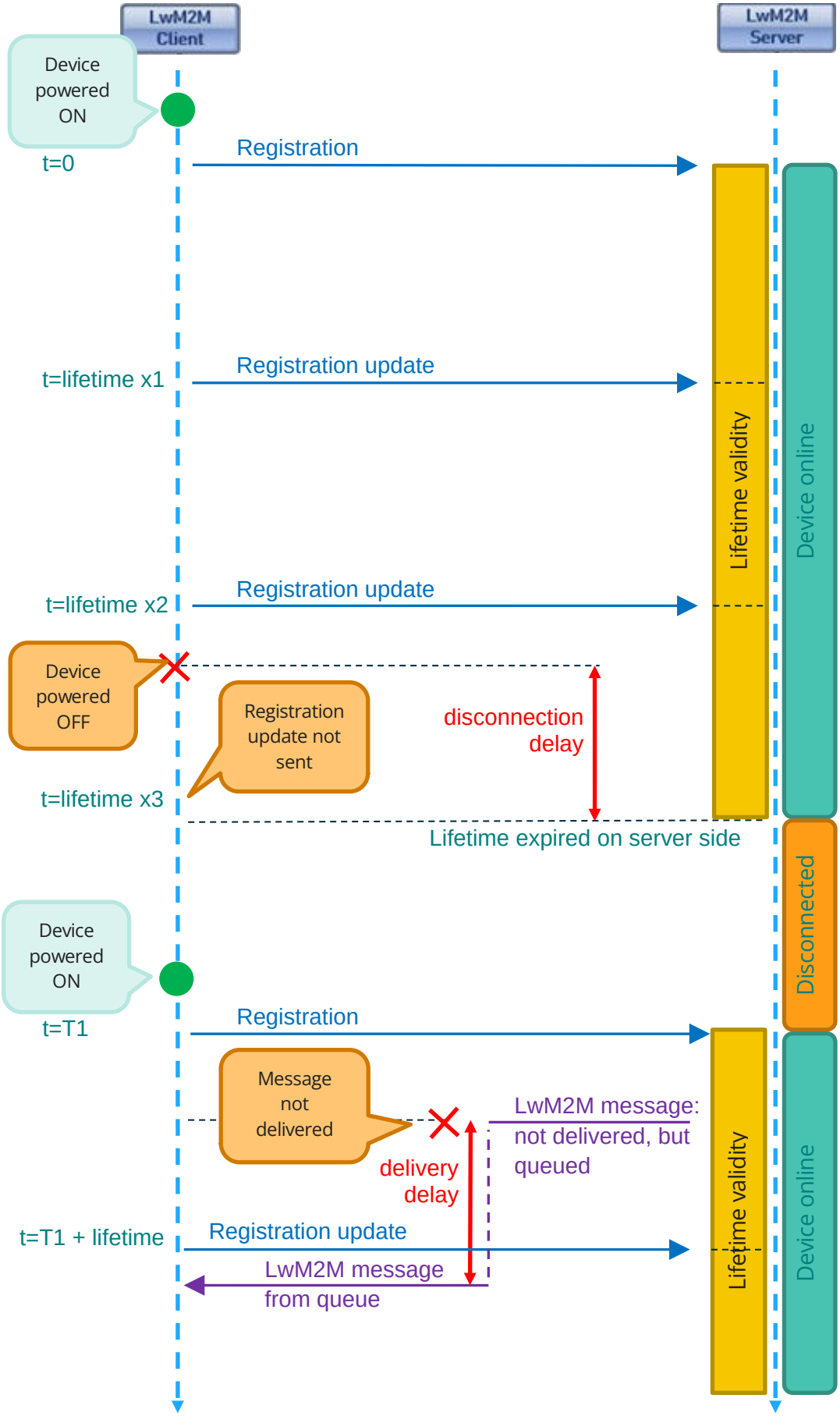



**Figure 20: LwM2M registration update and queue example**

b If the device is **not power supplied all the time** (i.e., battery powered) and it's switched on periodically to send data; the device performs a LwM2M registration at least every power up.

So, to be sure that the device is seen as **Online** from the server for all the time and the undelivered messages are stored in the queue, **lifetime should be greater than the gap between two consecutive LwM2M registrations/power up events.**

For example, if the device powers on and registers every hour, lifetime should be configured greater than 1 hour. lifetime >= powerUp_frequency.

**Warning:** In this scenario it's necessary that the LwM2M client must not send the de-registration request to the server.On the other hand, if the LwM2M client sends the de-registration, the server will consider the device as "disconnected", so the messages will not be stored in the queue. Refer to event t=T4 on the Figure 21 below.

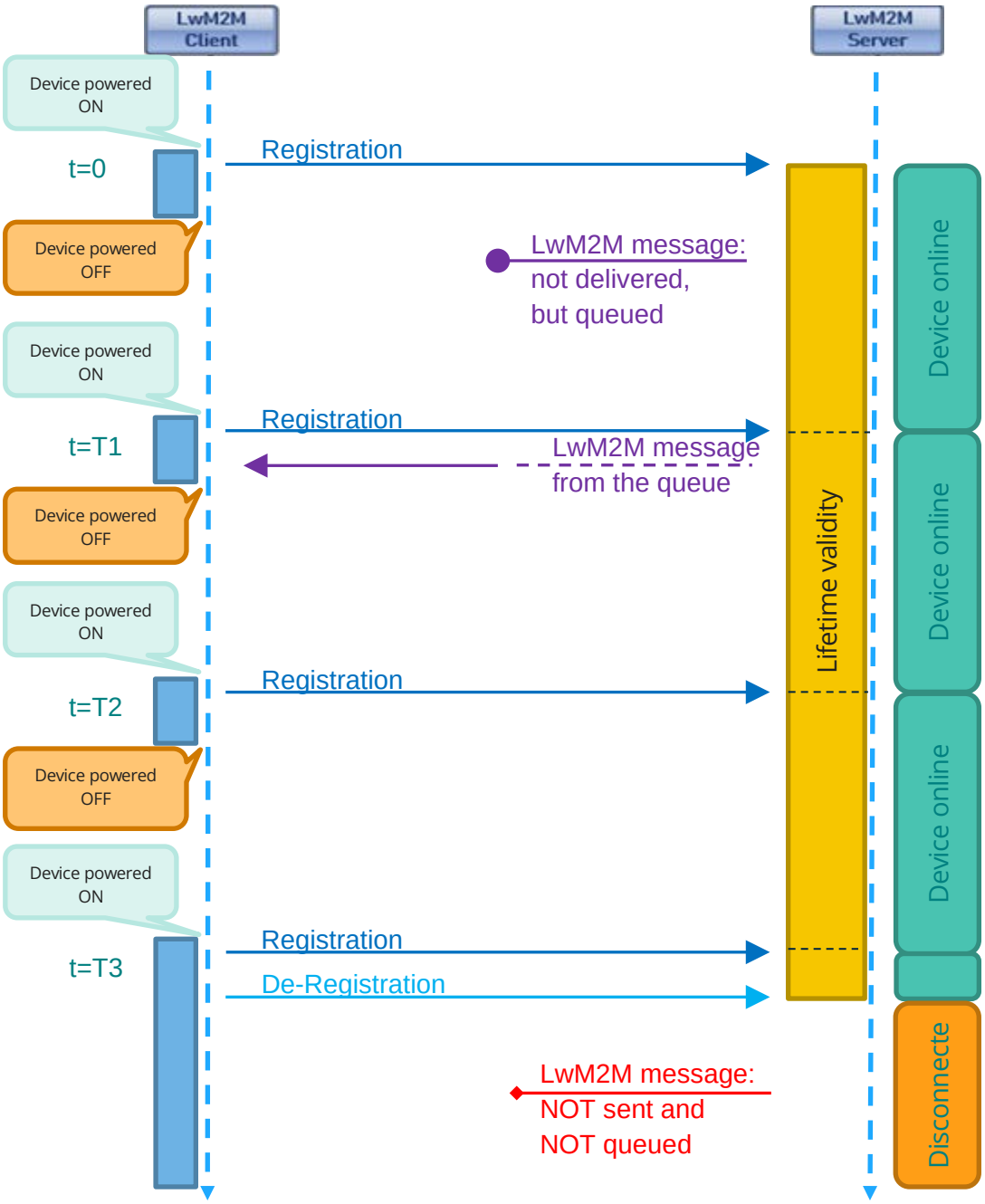



Figure 21: Example of device powered on periodically without and with LwM2M de-registration.

**Warning:** Be aware that all LwM2M messages consume data on traffic channel and LwM2M messages on OneEdge billing, so the User must choose carefully the registration update frequency (so the lifetime) according to customer requirements and data consumption estimation.

The table below summarizes the User scenarios involving lifetime, NAT, wakeup SMS and some typical application constraints:

**Table 8: Suggested lifetime setting according to User scenario and requirements.**

| Scenario | User Requirements | Suggested settings | Notes/Pros/Cons |
|---|---|---|---|
| Metering | Data from the device to the portal to be delivered **as soon as possible** | Lifetime can be small or high because doesn't affect metering data delivery | Note: lifetime should be high enough (hours or days) to save data and LwM2M messages |
| Remote settings | Data from the portal to the device to be delivered **as soon as possible** | A small lifetime (i.e. 60 seconds) guarantees fast delivery because the NAT entry doesn't expire | Cons: Small lifetime consumes big amount of data and LwM2M messages |
| | | high lifetime with WakeUp SMS on demand | Pro: save data and API<br>Cons: SMS is usually an additional cost<br>Note: convenient if used very rarely |
| | | high lifetime with No-NAT service | Pro: save data and API<br>Cons: it is usually an additional cost<br>Note1: convenient if the events are many and important<br>Note2: OneEdge Direct service provided by Telit |
| | Data from the portal to the device to be delivered **within accepted delay** | lifetime equal to maximum accepted delay | |

## 5.6 OneEdge Direct

In order to overcome the NAT known limitations described in paragraphs, NAT Environment and UDP Expire Time, Binding Parameter and Queue Mode

Telit provides a service on Next connectivity called OneEdge Direct.

Using this specific service, the typical NAT expiration rules will not be applied to LwM2M data between LwM2M client and server on OneEdge IoT portal.

This means that the LwM2M client will always receive incoming data from the server without the need of the queue mechanism.

This service is mainly valuable for applications where the data must be delivered as soon as possible to the remote device, for example opening a gate, raising a fire alarm bell, and so on.

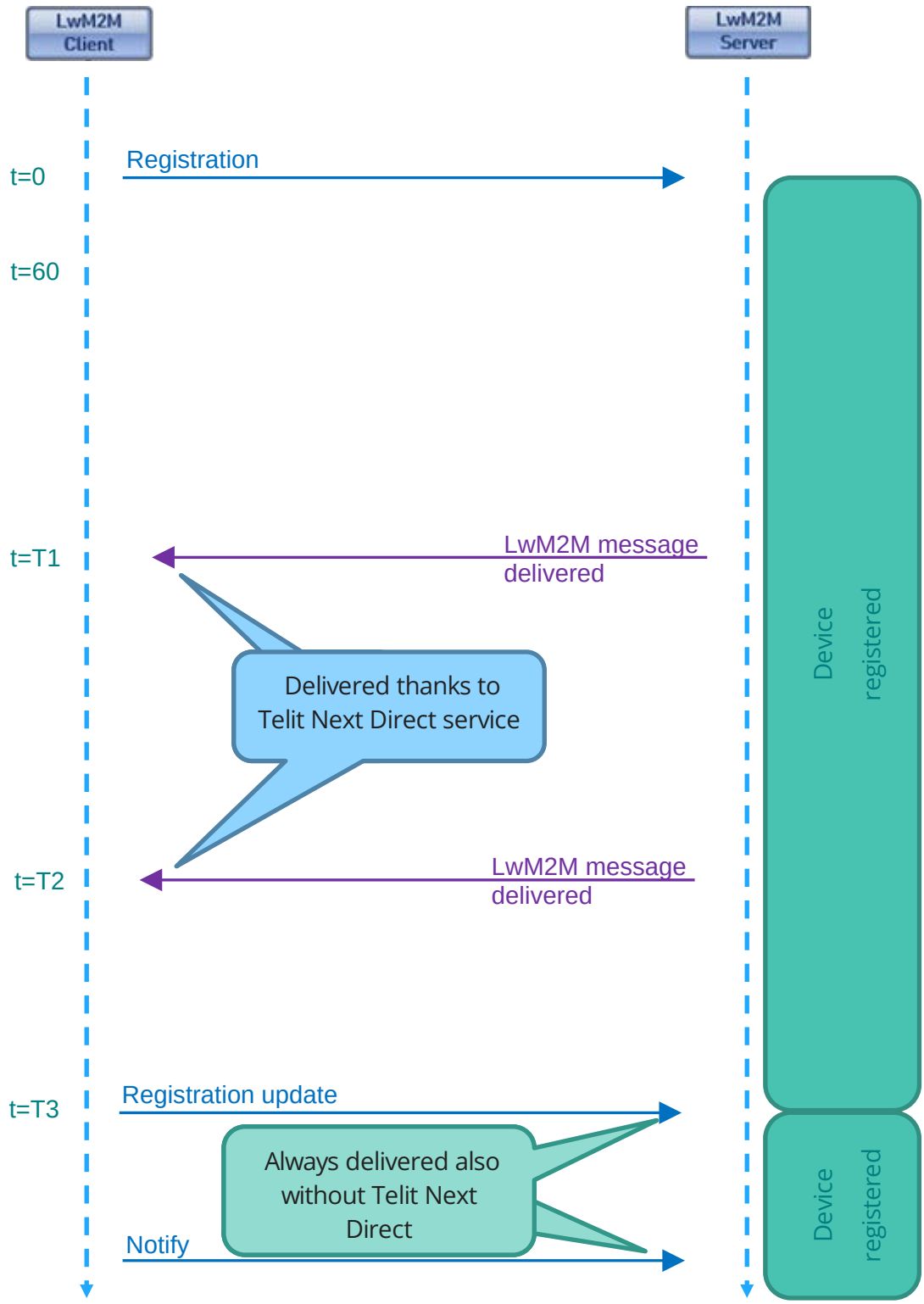


Figure 22:Example of behavior in case of OneEdge Direct to overcome NAT known limitations.

# 6 LwM2M Registration Features in OneEdge

In this chapter, an overview about the AT commands to manage the basic LwM2M registration functionalities is provided.

More details are available on **LwM2M AT reference guide** documents referenced in chapter 16.

## 6.1 Bootstrap and Registration

The main rules and constraints related to bootstrap and DM registration procedures are reported in the following list and referred to the Figure 23 below.

These rules are compliant to the LwM2M OMA specifications and the additional

OneEdge valuable features are highlighted.

a The default LwM2M server configured on the Telit device is the **OneEdge Bootstrap server**.

b Since in OneEdge environment the LwM2M communications must the encrypted through **DTLS protocol and Pre-Shared keys** (as known as PSK), the DTLS identifier and the Pre-Shared key are configured in every device during the production in the factory and saved in a **secure area** of the modem (TrustZone), that cannot be accessible by the host application.

c Every device has a unique **EndPoint**, a unique **DTLS Id** and a unique **PSK key**.

d The Bootstrap server knows all the DTLS Identifiers and the associated PSK keys.

e Before enabling the LwM2M client on the device for the first time, it must be **onboarded on the OneEdge IoT portal**, such that the specific LwM2M and DTLS parameters are loaded on the specific target organization.

> **Note: OneEdge:** This feature is called OneEdge Zero-touch onboarding: application developers don't need to configure any LwM2M and security parameters because it's already done by Telit.

f Once the **LwM2M** client is enabled for the first time, it performs the bootstrap procedure using the default LwM2M and DTLS parameters.

g During the bootstrap procedure, the bootstrap server sends the address of the target DM server with the new LwM2M and DTLS parameters to the client, to be used in the next session with the DM server.

h In OneEdge environment, the DM address, the LwM2M and DTLS parameters are configured in the **Device profile** section of OneEdge IoT portal.

After the bootstrap procedure, the LwM2M client automatically registers to the DM server using the new LwM2M and DTLS parameters.

Some parameters are coming from the Bootstrap provisioning (Device Profile configuration in OneEdge) like lifetime and binding, and others are hard coded like the supported LwM2M version (1.0) and the list of available LwM2M objects.

Once the bootstrap provisioning is completed, every time the LwM2M client is again enabled, it will directly register to the DM server.

The LwM2M client can be restored to factory settings through the **AT#LWM2MSTS** command: in this case the LwM2M client restarts from the Bootstrap procedure.

According to OMA specifications, the **registration update** must be sent **before the lifetime expiration**, so it could be sent in any moment within the lifetime.

The LwM2M client in Telit modules is able to register to any LwM2M server: the custom server address must be configured through **AT#LWM2MSTS** command and custom security parameters through **AT#LWM2MINJKEYS** one.

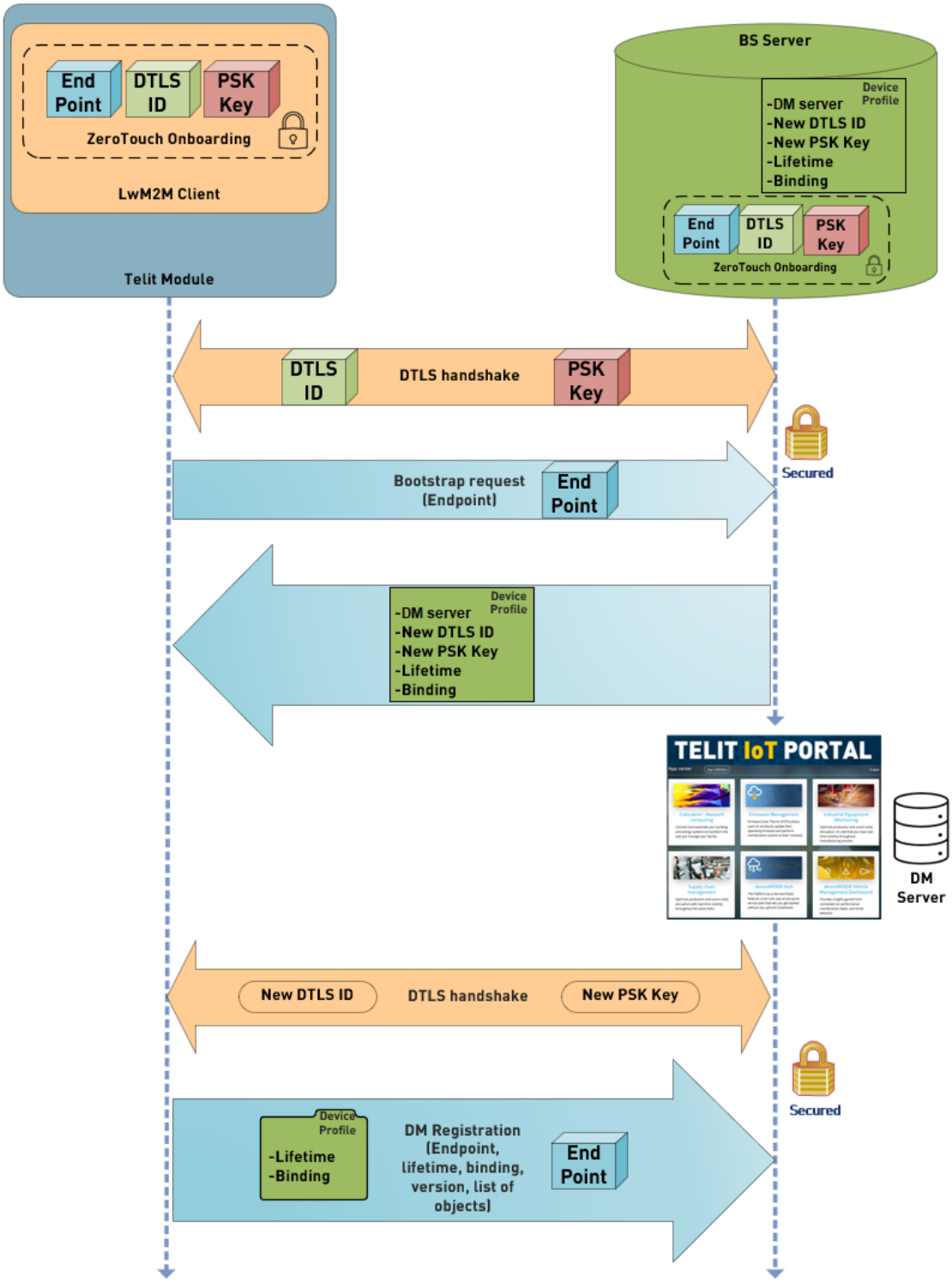



Figure 23: LwM2M parameters management in OneEdge BS and DM servers

# AT#LWM2MENA

When the LwM2M client is enabled for the first time, it performs the registration to the BS server and then to the DM one, according to the provisioned parameters during the Bootstrap procedure.

In OneEdge IoT portal, the DM parameters are configured in the Device profile and the default DM server is the OneEdge server itself.

Once the LwM2M client is successfully registered to a DM server, it will never perform the Bootstrap procedure again, unless it's forced manually (refer to j point in the next sequence).

Since the LwM2M client is disabled as default, the AT command to activate and deactivate is AT#LWM2MENA=n, where n must be 0 to deactivate and 1 to activate.

> **Warning:** If the LwM2M client was active before to enter to PSM state:
> The client will start automatically in case of periodic wakeup due to T3412 timer expiration.
> The client will **not** start in case of **manual wakeup due to ON-OFF button**, so the host application must enable manually the LwM2M client through **AT#LWM2MENA=1** command.

The LwM2M client, as default, provides only the bootstrap URC messages to the host application; by the way, additional URC related to DM registration can be activated through the AT command **AT#LWM2MCFG=0,7,0,1**. more details on paragraphs Additional LwM2M Configuration and OneEdge Telit Proprietary Objects

The example below shows some cases across the bootstrap and registration:

1. Power on the modem.
2. Enable the registration URC messages: **`AT#LWM2MCFG=0,7,0,1`** (or **`AT#LWM2MW=0,33211,0,0,1,1`** if not available)
3. Enable the LwM2M client: **`AT#LWM2MENA=1`**
4. Wait for bootstrap and registration URC:
```
LWM2M-TLT:"BOOTSTRAPPING",SSID=0,"coaps://bs.telit.io"
LWM2M-TLT:"BOOTSTRAPPED",SSID=0,"coaps://bs.telit.io"
LWM2M-TLT:"REGISTERING",SSID=99,"coaps://api.devicewise.com"
LWM2M-TLT:"REGISTERED",SSID=99,"coaps://api.devicewise.com"
```
5. Disable the LwM2M client: **`AT#LWM2MENA=0`**
6. Wait for deregistration URCs:
```
LWM2M-TLT:"DEREGISTERING",SSID=99,"coaps://api.devicewise.com"
LWM2M-TLT:"DEREGISTERED",SSID=99,"coaps://api.devicewise.com"
LWM2M-TLT:"CLIENT_DISABLED”
```
7. Power cycle the modem: **`AT#REBOOT`**
8. Enable the LwM2M client: **`AT#LWM2MENA=1`**
9. Wait for **registration URC** (no bootstrap in this case because the client is already bootstrapped):
```
LWM2M-TLT:"REGISTERING",SSID=99,"coaps://api.devicewise.com"
```

```
LWM2M-TLT:"REGISTERED",SSID=99,"coaps://api.devicewise.com"
```

**Note:** A host application can verify if the LwM2M client is active through the **AT#LWM2MENA?** Command.

```
AT#LWM2MENA?
#LWM2MENA: 1,1,0,5,5,"ACTIVE"
OK
```

## AT#LWM2MSTS

The AT command to force the client to restart from the Bootstrap procedure is **AT#LWM2MSTS=0,999,"coaps://bs.telit.io",1** and the setting is applied only at next power cycle.

**Restore the LwM2M client to default configuration** and restart from Bootstrap:
`AT#LWM2MSTS=0,999,"coaps://bs.telit.io",1`

Power cycle: **AT#REBOOT**

Enable the registration URCs: **AT#LWM2MCFG=0,7,0,1** (or **AT#LWM2MW=0,33211,0,0,1,1** if not available)

Enable the LwM2M client: **AT#LWM2MENA=1**

Wait for bootstrap and registration URCs:

```
LWM2M-TLT:"BOOTSTRAPPING",SSID=0,"coaps://bs.telit.io"
LWM2M-TLT:"BOOTSTRAPPED",SSID=0,"coaps://bs.telit.io"
LWM2M-TLT:"REGISTERING",SSID=99,"coaps://api.devicewise.com"
LWM2M-TLT:"REGISTERED",SSID=99,"coaps://api.devicewise.com"
```

## AT#LWM2MLIST

The command **AT#LWM2MLIST** provides the list of available objects and instances on the LwM2M client repository.

It can be used, for example, to verify if the LwM2M client is already bootstrapped:

AT#LWM2MLIST=0:

1. If the device is not yet bootstrapped, the Server object has no instances:

   ```
   #LWM2MLIST: </1>
   ```

2. On the other hand, if the device is already bootstrapped, at least an instance, typically only the number 0, is present:

   ```
   #LWM2MLIST: </1/0>
   ```

The host application could use this command to understand if a custom object must be loaded or object instances must be created.

# AT#LWM2MREG

The current status and the timestamp information related to the registration of the LwM2M client on the server are provided by the **AT#LWM2MREG=0,3** command.

The response is in the following format:

```
#LWM2MREG:
<ShortServerID>,<regState>,<lastReg>,<nextUpdate>,<lastActivit
y>
```

The examples below are related to the main status of the LwM2M client state machine:

- Device not bootstrapped → ERROR

```
AT#LWM2MREG=0,3
ERROR
```

- Registration in progress (not registered)→ <regState> = 0:

```
AT#LWM2MENA?
#LWM2MENA: 1
OK
LWM2M-TLT:"REGISTERING",SSID=99,"coaps://api.devicewise.com"
AT#LWM2MREG=0,3
#LWM2MREG: 99,0,0,0,0
OK
```

- Device successfully registered → <regState> = 2:

```
AT#LWM2MENA?
#LWM2MENA: 1
OK
LWM2M-TLT:"REGISTERED",SSID=99,"coaps://api.devicewise.com"
AT#LWM2MREG=0,3
#LWM2MREG: 99,2,1635267678,1635267708,1635267678
OK
```

- Client disabled (not registered to any DM server) → <regState> = 0:

```
AT#LWM2MENA?
#LWM2MENA: 0
OK
AT#LWM2MREG=0,3
#LWM2MREG: 99,0,0,0,0
OK
```

- Client suspended through execution of 1/0/4 resource → <regState> = 1:

```
LWM2M-TLT:"SUSPENDED",SSID=99,"coaps://api.devicewise.com"
AT#LWM2MREG=0,3
#LWM2MREG: 99,1,1639139574,1639139604,1639139582
OK
```

**Note:** In any LwM2M application, it's strongly recommended to check periodically the status of the LwM2M registration to guarantee that the LwM2M client is always connected to the LwM2M server. Refer to paragraph Best Practices for Host Applications.

The Figure 24 shows different registration status values according to LwM2M events.

The messages in the yellow tips are the URC as described in AT#LWM2MENA

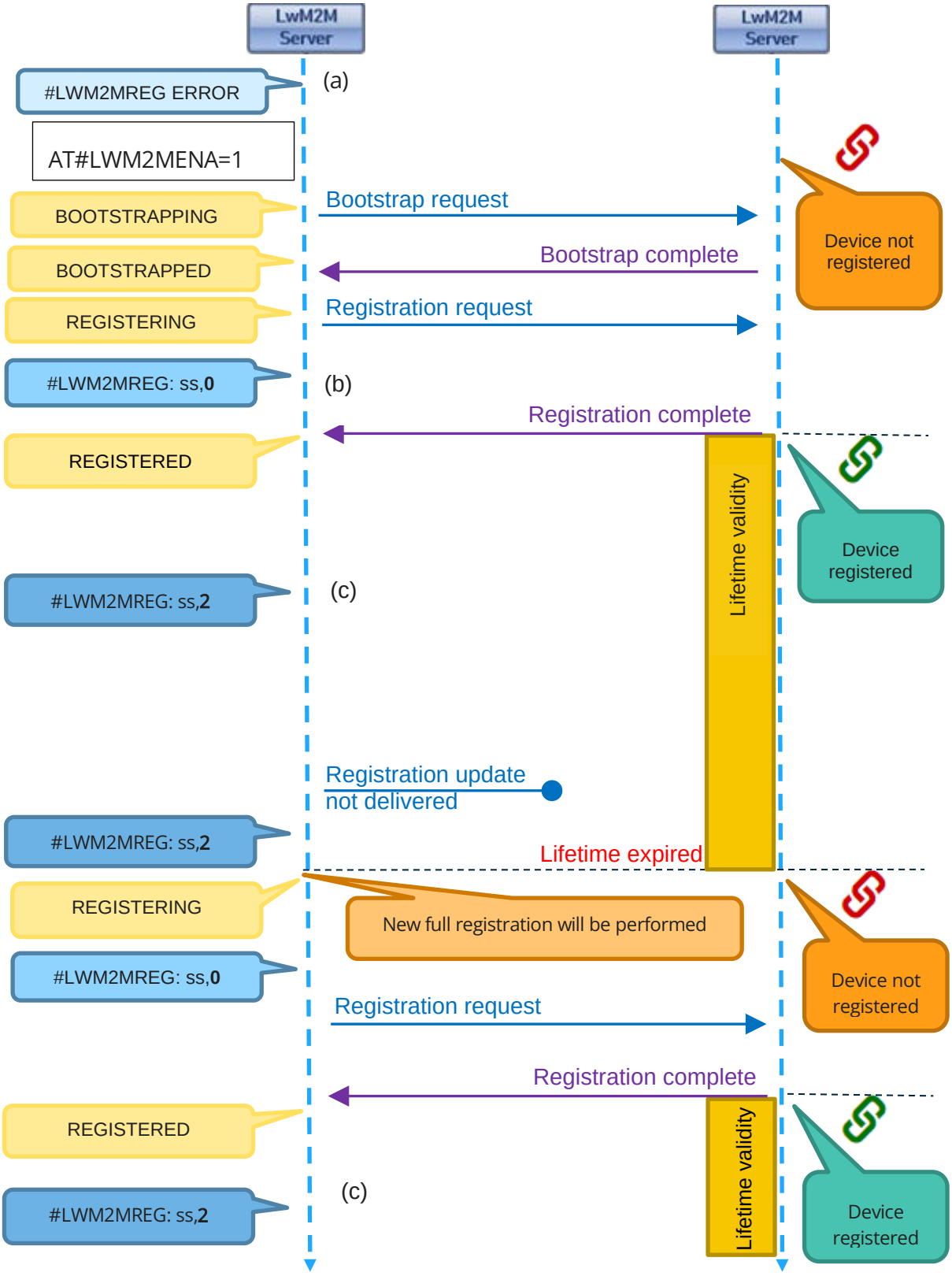


Figure 24: AT#LWM2MREG=0,3 responses according to different LwM2M events; "ss" is the Short Server ID

Furthermore, in Telit LwM2M client, it's possible to force the sending of a registration update asynchronously through the following AT command from the host application:

```
AT#LWM2MREG=0,2,<ShortServerID>
```

Where short_server_id is configured on bootstrap server and listed in the registration URCs (default value for OneEdge DM server is 99).

## AppZone APIs

The table below provides the matching between the AT commands of this chapter and the AppZone APIs for the same actions.

Table 9: List of AppZone APIs

| AT command | AppZone API |
|---|---|
| AT#LWM2MENA | m2mb_lwm2m_enable, m2mb_lwm2m_disable, m2mb_lwm2m_get_stat |
| LWM2M-TLT:"BOOTSTRAPPING"<br>LWM2M-TLT:"BOOTSTRAPPED"<br>LWM2M-TLT:"REGISTERING"<br>LWM2M-TLT:"REGISTERED" | M2MB_LWM2M_SRV_INFO_IND event on the LwM2M callback |
| AT#LWM2MSTS | m2mb_lwm2m_sts |
| AT#LWM2MLIST | m2mb_lwm2m_list |
| AT#LWM2MREG | m2mb_lwm2m_reg |

### Useful AppZone Links

Refer to the online documentation at following link

https://s3.amazonaws.com/site_support/Telit/m2mb_docs/ME910G1-ME310G1-ML865G1/api-reference/37.00.xx6/index.html

AppZone sample applications available at following repository

https://github.com/telit/IoT-AppZone-SampleApps/tree/master/ME310G1-ME910G1-ML865G1/AppZoneSampleApps-MAIN_UART/LWM2M

## 6.2 Bootstrap and Registration retries

Since LwM2M is based on UDP transport protocol, some messages between client and server could be lost or delayed due to network conditions.

So, in case the bootstrap and registration procedures fail, the LwM2M client applies backoff procedures to automatically retry to connect to LwM2M server.

The typical rules of the backoff procedures are:

- If the bootstrap fails, the LwM2M client **retries for a limited number of times** (i.e., 3 retries) and after few minutes (i.e., 3 minutes), then gives up.
- If the DM registration fails after AT#LWM2MENA=1, the LwM2M client **retries for some times** (i.e., 6 times), but with **incremental time gaps** (i.e., 1, 2, 3, 4 minutes and then 12 hours for the sixth retry);
- If the LwM2M registration fails again after all the retries, the client comes back to the bootstrap (case c)

## 6.3 Registration Update

When the LwM2M client is successfully registered to a LwM2M server, it must refresh periodically the registration status through the **registration update** message.

This message is like a keepalive and must be sent by the LwM2M client before the lifetime expires.

For more details about lifetime parameter, refer to paragraph Lifetime Parameter

Since it is an automatic operation of the LwM2M client performed in background, no indication is provided to the serial interface so the host application is not aware of it.

By the way, the host application can retrieve the timestamps related to the last and next activities through the command **AT#LWM2MREG=0,3**.

The information are in the format.

```
#LWM2MREG:
<shortServerID>,<regState>,<lastReg>,<nextUpd>,<lastActivity>
```

Example:

```
AT#LWM2MREG=0,3
#LWM2MREG: 99,2,1635267678,1635267708,1635267678
OK
```

Furthermore, the host application can also force the LwM2M client to send a registration upgrade to the server through the command

```
AT#LWM2MREG=0,2,<shortServerID>
OK
```

### Sad Path

If the registration update process fails for any reason (bad network coverage, lost UDP messages, outage of the server, etc), the LwM2M client performs some retries of the registration update process.

If the lifetime expires when the registration update has not been successful yet, then the LwM2M client gives it up and performs a full registration, which is already described in the previous paragraph Bootstrap and Registration Retries

Below is an example of possible scenarios:

```
AT#LWM2MCFG=0,7,0,1 (or AT#LWM2MW=0,33211,0,0,1,1 if not
available)
OK
AT#LWM2MENA=1
OK
LWM2M-TLT:"BOOTSTRAPPING",SSID=0,"coaps://bs.telit.io:5684"
LWM2M-TLT:"BOOTSTRAPPED",SSID=0,"coaps://bs.telit.io:5684"
LWM2M-TLT:"REGISTERING",SSID=99,"coaps://api.devicewise.com"
LWM2M-TLT:"REGISTERED",SSID=99,"coaps://api.devicewise.com"
…
```

Here the host application works as expected, sending and receiving data.

Normal case:

After some time (i.e. 300 seconds earlier than the lifetime expiration), the LwM2M client successfully sends the registration update to the server: registration refreshed and no URC is received on serial port.

Failure case:

After some time (again, 300 seconds earlier than lifetime expiration), the LwM2M client sends the registration update to the server, but the procedure fails: no URC is received on the serial port.

After few seconds, the LwM2M client sends a new registration update retry and it fails again: no URC is received on the serial port.

In the meantime the lifetime is expired: according to LwM2M OMA specification, the registration update is not more sufficient so the LwM2M client must send a full registration to the server: only in this case the REGISTERING URC is sent to the serial port

```
LWM2M-TLT:"REGISTERING",SSID=99,"coaps://api.devicewise.com"
```

When the registration procedure is successfully completed, the REGISTERED URC is received:

```
LWM2M-TLT:"REGISTERED",SSID=99,"coaps://api.devicewise.com"
```

Otherwise, if the first registration attempt fails, other **REGISTERING URC** messages could be received waiting for the **REGISTERED** one.

> **Warning:** **REGISTERING** and **REGISTERED URC** are disabled as default and can be enabled through the command **AT#LWM2MCFG=0,7,0,1**.

> **Note:** The number of retries and the timeouts between **REGISTERING** and **REGISTERED** are described in the previous paragraph Bootstrap and Registration Retries.

## 6.4 Best Practices for Host Applications

One of the key points of the LwM2M client in Telit environment is that the client itself is able to monitor and recover some lost messages and context activation.

Also the automatic retries are very useful according to previous paragraphs Bootstrap and Registration Retries Registration Update

By the way, if the host application would take the control of the logic, the following actions can be applied, periodically or when needed, for example before to send data.

> **Note:** The order of the actions follow a step-by-step flow according to the priority at every startup:

Check **SIM presence** (AT+CPIN?)

- Wait for the "+CPIN: READY" response before to continue.

Check cellular connectivity (AT+CREG?, AT+CEREG?, AT+CGREG?, AT+COPS?)

- Wait for GSM cellular registration ("+CREG: x,1" or "+CREG: x,5") or CAT-M/NBIoT cellular registration ("+CEREG: x,1" or "+CEREG: x,5")
- If the modem is not registered to any cellular network, the host application could force a new cellular scan (AT+COPS=2, AT+COPS=0) or reboot the modem as last option.

Check APN settings (AT+CGDCONT?)

- If the APN is not configured as expected, it must be set again (AT+CGDCONT=1, IP,"myapn")

Verify if the LwM2M client has a valid bootstrap (AT#LWM2MLIST=0)

- #LWM2MLIST: </1> in case of no bootstrap
- #LWM2MLIST: </1/0> in case of bootstrap already performed

Enable additional DM URC (AT#LWM2MCFG=0,7,0,1)

Enable the LwM2M client (AT#LWM2MENA=1)

- Wait for BOOTSTRAPPING, BOOTSTRAPPED if the client has no bootstrap.
- Wait for REGISTERING and REGISTERED URC if the client has been already bootstrapped.

If the **REGISTERED URC** is not received within a timeout (i.e. 20-30 seconds, the host application could perform some "recovery" actions.

Below are some strategies that a host application could perform:

- Wait for automatic retries according to paragraph Bootstrap and Registration Retries
- Disable the LwM2M client before the automatic retries and enable again.
- The host application could have a logic such that, after some registration failures, the host application could reset the LwM2M client through the command AT#LWM2MSTS. Refer to AT#LWM2MSTS
- to perform the bootstrap again.

Periodically or before to send data:

Check cellular connectivity (AT+CREG?, AT+CEREG?, AT+CGREG?, AT+COPS?)

- Wait for GSM cellular registration ("+CREG: x,1" or "+CREG: x,5") or CAT-M/NBIoT cellular registration ("+CEREG: x,1" or "+CEREG: x,5").
- If the modem is not registered to any cellular network, the host application could force a new cellular scan (AT+COPS=2, AT+COPS=0) or deactivate the SIM card (AT+CFUN=4, AT+CFUN=1) or reboot the modem as last option.

Check **LwM2M client enable** status (AT#LWM2MENA?)

- If the LwM2M client is disabled unexpectedly, it must be enabled again (AT#LWM2MENA=1)

Check **LwM2M client registration** status (AT#LWM2MREG=0,3)

- If the status is "**not registered**" (and the client is enabled from previous step), the registration is in progress (REGISTERING), so the host application could wait for automatic retry logic or disable the client and enable it again (refer to steps above).

## 6.5 Additional LwM2M Configuration

### AT#LWM2MINJKEYS

This command is used to configure custom LwM2M and DTLS parameters (EndPoint, PSK Identity and PSK Key) to LwM2M client different than default factory configuration.

The same parameters must be configured for every device also on server side.

> **Note:** Since every Telit device is provisioned with a specific set of parameters on factory and on OneEdge server side, this command is not needed in OneEdge environment.

### AT#LWM2MCFG

This command is used to customize some parameters of the LwM2M client.

The syntax is:

```
AT#LWM2MCFG=<agentId>,<paramId>,<actionId>[,<value>]
```

The **agentId** parameter must be set to 0.

The **paramId** parameter is used to select the setting to modify or read.

Possible values are:

**Table 10: possible values for "paramId" parameter**

| paramId | description | value |
|---|---|---|
| 0 | reserved for internal use | |
| 1 | PDN type connection | 0: IPv4 + IPv6 (default in MEx10G1 family)<br>1: IPv4 (default in MEx10C1 family) |
| 2 | COAP acknowledge timeout (covered by object 500, refer to tip below) | 1-30 (default: 2) |
| 3 | COAP maximum retransmission count (covered by object 500, refer to tip below) | 1-5 (default: 4) |
| 4 | External observations polling time | 2-2592000 |
| 5 | reserved for internal use | |
| 6 | GNSS service (only for Telit agent) | 0: disable (**default**)<br>1: enable |
| 7 | registration URCs (only for Telit agent) | 0: disable (**default**)<br>1: enable |
| 8 | user activated PDP (only for Telit agent) | 0: disable (**default**)<br>1: enable |
| 9 | DTLS retransmission (only for Telit agent) | 0: disable (**default**)<br>1: enable |

**Warning:** Some param Ids could be not available in some SW versions, especially old ones, ensure to check LwM2M AT commands reference guide document for more details.

The **actionId** parameter can be 0 or 1:

- 0 to set a value into a parameter.
- 1 to get current value.

The last field **value** is the value to be set in case of **actionId** equal to 0.

**Note: OneEdge:** Suggested settings at startup:

AT#LWM2MCFG=0,7,0,1 to enable registration URCs.
This setting is available also on object 33211, but the User should prefer this AT command.
AT#LWM2MCFG=0,9,0,1 to enable DTLS retries.

**Note: OneEdge:** The setting of CoAP timeout and retries (parameters 2 and 3) are applied only to LwM2M client, but not at server side.

By the way, these parameters are covered also by object 500 that applies the values on **both client and server side**.

The User should **prefer Object 500** when configuring these settings.

## AppZone APIs

The table below provides the matching between the AT commands of this chapter and the AppZone APIs for the same actions.

**Table 11: AppZone APIs**

| AT command | AppZone API |
|---|---|
| AT#LWM2MINJKEYS | m2mb_lwm2m_injkeys |
| AT#LWM2MCFG | m2mb_lwm2m_cfg |

- More AppZone documentation resources are available on the AppZone section of paragraph Bootstrap and Registration.

# 7 Operations to LwM2M Resources in OneEdge

According to LwM2M Device Management interface and Data Model described in Logical Interfaces LwM2M Data Model , the LwM2M Resource can be accessed from client and server side.

In this chapter, the AT commands on host application side and the specific section on OneEdge IoT portal side will be described.

## 7.1 Operations on Client Side

Following AT commands are available on Telit modem to interact with LwM2M resources on client side:

- AT#LWM2MW allows to modify the value of a writable resource (RW).
- AT#LWM2MR allows to read the value of a readable resource (RW).
- AT#LWM2MSET allows to force the modification of the value of a read-only resource (R).
- AT#LWM2MGET allows to force the reading of the value of a write-only resource (W).
- AT#LWM2ME allows to execute an executable resource (E).
- AT#LWM2MOBJSET allows to force the modification of multiple resources of the same object instance for both RW and R resources.
- AT#LWM2MOBJGET allows to force the reading of multiple resources of the same object instance for both RW and W resources.

**Warning:** Writing a value to a resource doesn't mean that the value will be sent to the server: refer to observations on chapter LwM2M Observations in OneEdge.

The table below is a summary of main details about operations of the resources:

Table 12: Summary of actions related to resource operations.

| Resource Access type | Example of application | Client side | | Server side | |
|---|---|---|---|---|---|
| | | READ | WRITE | READ | WRITE |
| R | Temperature sensor from device to portal | AT#LWM2M**R** | AT#LWM2M**SET** | ✓ | ⊘ |
| W | Secret remote setting from portal to device (i.e. a password) | AT#LWM2M**GET** | AT#LWM2M**W** | ⊘ | ✓ |
| RW | Remote setting from portal to device | AT#LWM2M**R** | AT#LWM2M**W** | ✓ | ✓ |
| E | Remote action from portal to device | AT#LWM2M**E** | | ✓ | |

The syntax of these AT commands has some common parameters:

```
AT#LWM2M<W|R|E>=agentInst,objId,objInstId,resId,resInstId...
AT#LWM2M<SET|GET>=type,objId,objInstId,resId,resInstId...
AT#LWM2M<OBJSET|OBJGET>=agentInst,objId,objInstId
```

The **agentInst** parameter must be set to 0.

The **objId, objInstId** and **resId** parameters are related to the resource URI according to the format **/objId/objInstId/resId**.

The "**ResInstId**" parameter is used to address multiple instance resources: if the resource is single, the parameter must be 0 otherwise it's the identifier of the instance resource.

The **type** is used to declare the type of the resource only in AT#LWM2MSET and **AT#LWM2MGET** commands, according to Table 3 in Object.

More specific details related to the AT commands are provided in the next paragraphs.

# AT#LWM2MW

The **AT#LWM2MW** command is used to set the value of writable resources.

The syntax is:

```
AT#LWM2MW=agentInst,objId,objInstId,resId,resInstId,value
```

The encoding of the **value** parameter is explained in the Table 14 below.

Example to write the value 25.1 to the Temperature resource:

```
AT#LWM2MW=1,3303,0,5700,0,25.1
OK
```

# AT#LWM2MSET

The AT#LWM2MSET command is used to set the value of ReadOnly resources.

The syntax is:

```
AT#LWM2MSET=type,objId,objInstId,resId,resInstId,value
```

The "type" parameter must be set according to the Table 14 below.

Example to set the value 99 to the Battery level resource (3/0/9):

```
AT#LWM2MSET=0,3,0,9,0,99
OK
```

The value parameter must be encoded in the following format:

**Table 13:LwM2M resource types in OneEdge environment**

| Resource type | "type" field | "value" parameter format | Examples |
|---|---|---|---|
| Integer | 0 | Signed number without the decimal part | AT#LWM2MW=0,obj,inst,res,resInst,**123**<br>AT#LWM2MSET=**0**,obj,inst,res,resInst,-**100** |
| Boolean | 0 | Unsigned number. Only 0 for False meaning and 1 for True are allowed | AT#LWM2MW=0,obj,inst,res,resInst, **0**<br>AT#LWM2MSET=**0**,obj,inst,res,resInst,**1** |
| Float | 1 | Signed number x with the decimal part y mandatory in the format x.y (use x.0 if the | AT#LWM2MW=0,obj,inst,res,resInst,**123.45**<br>AT#LWM2MSET=**1**,obj,inst,res,resInst,-**100.0** |

| Resource type | "type" field | "value" parameter format | Examples |
|---|---|---|---|
| | | number has no decimal part) | |
| String | 2 | Text string between double quotes. Max length is 255 bytes so 255 characters on serial ports | AT#LWM2MW=0,obj,inst,res,resInst,**"Hello"**<br>AT#LWM2MSET=**2**,obj,inst,res,resInst,**"World"** |
| Opaque | 3 | Text in hexadecimal format between double quotes.<br>The length of an atomic data is 2 characters.<br>Allowed characters are 0-9,A-F.<br>Max length is 1024 atomic info (bytes), so 2048 characters on serial port | AT#LWM2MW=0,obj,inst,res,resInst,**"01234"**<br>AT#LWM2MSET=**3**,obj,inst,res,resInst,**"ABCD"** |
| Object Link | 4 | The object link is used to refer an Instance of a given Object. The format is a text string in the format <objId>:<instId>. Both objId and instId are 2 bytes long | AT#LWM2MW=0,obj,inst,res,resInst,**"4:0"**<br>AT#LWM2MSET=**4**,obj,inst,res,resInst,**"4:0"** |
| Time | 5 | Unsigned number in Epoch Unix format (number of seconds since Jan 1st, 1970 in the UTC time zone) | AT#LWM2MW=0,obj,inst,res,resInst,**1640390400**<br>AT#LWM2MSET=**5**,obj,inst,res,resInst,**1640390400** |

**Warning:** The float value in the AT#LWM2MSET and AT#LWM2MW commands must always have the decimal point. For example, the value 25 must be written as 25.0.

**Note:** If the User needs to write ASCII characters into a LwM2M string resource, the AT#SETHEXSTR command can be used to include the hexadecimal value of the ASCII character inside the double quotes of the AT command parser.

Below is the example to write a JSON content inside a string where the double quotes character " is encoded according to its hex value 0x22 in the format \22:

```
AT#SETHEXSTR=2
```

```
OK
AT#LWM2MW=0,26253,0,21,0,"{\22test1\22:2000}"
OK
AT#LwM2MR=0,26253,0,21,0
#LWM2MR: "{"test1":2000}"
OK
```

The AT#SETHEXSTR supports only the characters in the range0x20-0x7F

# AT#LWM2MR

The AT#LWM2MR command is used to read the value of readable resources.

The syntax is:

```
AT#LWM2MR=agentInst, objId, objInstId, resId, resInstId
```

Example to read the value of the Battery level resource:

```
AT#LWM2MR=0,3,0,9,0
#LWM2MR: 99
OK
```

## Floating Point Constrain for Float Type

The following additional warning must be considered for the management of the float resources:

> **Warning:** In case of float, there might be a possible mismatch between the set and the read values due to a technical limitation of the floating point representation of the float number in the system. The host application can bypass this behaviour using a proper approximation.

The example below is to write the value 75.11 as temperature value of **Temperature** IPSO object (object ID 3303, resource ID 5700, float type, R) but the value really stored in the system through the floating point representation is 75.109999999999999.

```
AT#LWM2MSET=1,3303,0,5700,0,75.11
OK
AT#LWM2MR=0,3303,0,5700,0
#LWM2MR: 75.109999999999999
OK
```

# AT#LWM2MGET

The AT#LWM2MGET command is used to get the value of WriteOnly resources.

The syntax is:

```
AT#LWM2MGET=type, objId, objInstId, resId, resInstId
```

Example to read the value of the Battery level resource:

```
AT#LWM2MGET=2,9,0,3,0
```

```
#LWM2MR: "http://hosting.com"
OK
```

## AT#LWM2ME

The AT#LWM2ME command is used to execute a resource.

The syntax is:

```
AT#LWM2ME=agentInst, objId, objInstId, resId, resInstId
```

**Note:** Even if an executable resource can be executed from client side, usually this action is done from server side.

## AT#LWM2MOBJSET

The AT#LWM2MOBJSET command is used to set the value of more resources of the same object instance in one-shot command.

The syntax is:

```
AT#LWM2MOBJSET=agentInst, objId, objInstId
```

After the command is issued, a prompt ">" is returned to receive the resource values in JSON format according to OMA specification.

The JSON string is finally sent to the LwM2M client through CTRL+z (0x1A) character.

The example below shows a generic JSON string that covers all the LwM2M resource types: integer, float, timestamp, boolean, string, opaque in Base64 format, objectLink, multiple integer.

The bold parameters must be replaced with real URI values.

```
AT#LWM2MOBJSET=0,objId,objInstId
>
{
  "bn":"/objId/objInstId/",
  "e":
     [
      {"n":"<Integer_resId>","v":123},
      {"n":"<Float_resId>","v":99.9},
      {"n":"<Time_resId>","v":1620828702},
      {"n":"<Boolean_resId>","bv":true},
      {"n":"<String_resId>","sv":"hello world"},
      {"n":"<Opaque_resId>","sv":"aGVsbG8gd29ybGQ="},
      {"n":"<ObjLink_resId>","ov":"6:0"},
      {"n":"<Integer_multiple_resId/resInstId>","v":456}
     ]
}
```

```
CTRL+z
OK
```

**Warning:** Some resources are managed in different ways comparing AT#LWM2MSET and AT#LWM2MOBJSET.

Boolean is 0/1 in AT#LWM2MSET rather than false/true in AT#LWM2MOBJSET.

Opaque is in hexadecimal format in AT#LWM2MSET rather than Base64 in AT#LWM2MOBJSET.

**Warning:** Maximum length of JSON string is **6144 characters** and maximum 512 tokens, which means **101 resources**.

The AT#LWM2MOBJSET command is very useful to set multiple resource of the same object in one-only command instead of performing multiple AT#LWM2MSET commands.

This command covers the typical use case of a set of sensors, i.e. temperature, humidity, and pressure, connected to the same device and to be written at the same time on the object.

In this way, the number of actions to be performed by the host application is reduced from n to 1 and also the execution time is reduced, which is a typical requirement for power-saved devices.

Furthermore, if an observation is in place at the object instance level, in case of AT#LWM2MOBJSET, the LwM2M client sends a unique notify with all the values to the LwM2M server instead of a notify for each resource sent through multiple AT#LWM2MSET commands (refer to paragraph 6.1.2).

This allows to reduce the data sent over the air too.

**Note:** Summary of key points of AT#LWM2MOBJSET:

a. Reduces number of actions of host application rather than AT#LWM2MSET command.
b. Reduces the time to set more resources of the same object instance.
c. If an observation at object instance level is in place, all the resource values are sent from the client to the server in one unique notify message.

**Warning:** The operation is a Partial Update, meaning that resources already existing in the target object instance and not present in the command will be neither changed nor removed.

# AT#LWM2MOBJGET

The AT#LWM2MOBJGET command is used to get the values of all the resources of an object instance in one-shot command.

The syntax is:

```
AT#LWM2MOBJGET=agentInst, objId, objInstId
```

Example to get all the values of Device object (3/0):

```
AT#LWM2MOBJGET=0,3,0
{"bn":"/3/0/",
"e":[
{"n":"0","sv":"Telit"},
{"n":"1","sv":"ME910G1-WW"},
{"n":"2","sv":"359206109992594"},
{"n":"3","sv":"M0C.400003"},
{"n":"6/0","v":0},
{"n":"6/1","v":0},
{"n":"6/2","v":0},
{"n":"7/0","v":0},
{"n":"7/1","v":0},
{"n":"7/2","v":0},
{"n":"8/0","v":0},
{"n":"8/1","v":0},
{"n":"8/2","v":0},
{"n":"9","v":100},
{"n":"10","v":3156},
{"n":"11/0","v":0},
{"n":"13","v":1639584771},
{"n":"14","sv":"+01:00"},
{"n":"16","sv":"UQ"},
{"n":"17","sv":"CellularModule"},
{"n":"18","sv":"Rev 0"},
{"n":"19","sv":"37.00.414-
P0C.410001\r\nM0C.400003\r\nP0C.410001\r\nA0C.410000"},
{"n":"20","v":0},
{"n":"21","v":4078}]
}
OK
```

Also this command is very useful to reduce the number of actions of a host application: 1 unique **AT#LWM2MOBJGET** command can replace many **AT#LWM2MGET** and **AT#LWM2MR** commands.

**Note:** Summary of key points of AT#LWM2MOBJGET:

a) Reduces number of actions of host application rather than AT#LWM2MGET and AT#LWM2MR command.

b) Reduces the time to set get resources of the same object instance.

## Single or Multiple Resource

All the previous AT commands are valid for both single and multiple resources.

In case of single resources, the parameter "resInstId" is not considered, so it must be equal to 0.

On the other hand, in case of a multiple resource, "resInstId" must be used to address the specific instance of the resource.

The Telit LwM2M client maps the resource instance identifiers starting from 0.

For example, according to the OMA specifications, the "IP address" resource of "Connectivity Monitoring" object is defined as multiple resource, so the Telit LwM2M client allocates the IPv6 address value on the first resource instance id (resInstId=0) and the IPv4 address value on the second one (resInstId=1):

```
AT#LWM2MR=0,4,0,4,0
#LWM2MR: "::"
OK
AT#LWM2MR=0,4,0,4,1
#LWM2MR: "10.10.95.190"
OK
```

**Note: OneEdge:** If a resource is defined as multiple on a custom object, the Telit LwM2M client allocates 10 multiple instances, with identifiers from 0 to 9.

**Warning:** In Telit LwM2M client the number of resource instances of custom objects is always 10 (resInstId from 0 to 9) and cannot be increased or decreased.

If an application requires more instances to represent a LwM2M resource, they can be split into more multiple resources:

Resource1: first 10 resource instances

Resource2: second set of 10 resource instances

Resource3: third set of 10 resource instances

## CoAP BlockWise

Both the Telit LwM2M client and server support CoAP BlockWise feature to transport up to 6KB of data in one LwM2M transaction.

Briefly, at high level, the BlockWise feature allows the host application on device side and the user application on server side to manage a data content bigger than the single CoAP packet maximum size (1KB).

At low level, the LwM2M client and server segment the data content into multiple CoAP packets that are then re-assembled at destination side, such that this procedure is transparent for the host/user experience.

**Warning:** Maximum CoAP payload that can be transferred through CoAP BlockWise feature in Telit LwM2M client and OneEdge server is **6KB**. Since the CoAP payload includes both LwM2M headers and payload the real maximum capacity for LwM2M data is **less than 6KB** and depends on the LwM2M headers specific for every data model and object. OneEdge technical support team is available to provide estimation and simulation of the allowed size case by case.

**Note:** Below two examples that involve the BlockWise feature:

The host application writes a long binary content, for example a data blob, of size 5KB bytes split into 5 opaque resources of an object instance; when the server reads the entire instance, the BlockWise feature is used between client and server to manage the big content in **one unique logical operation instead of reading the resources one by one.**

The user at server side wants to send a long string content, for example an SSL certificate, of 2000 characters; the user splits the string content into pieces of 255 characters and writes every piece in a string resource, performing a **unique write operation for all the resources** into the instance; the host application receives the string content distributed into the strings resources when the client re-assembles all the BlockWise blocks.

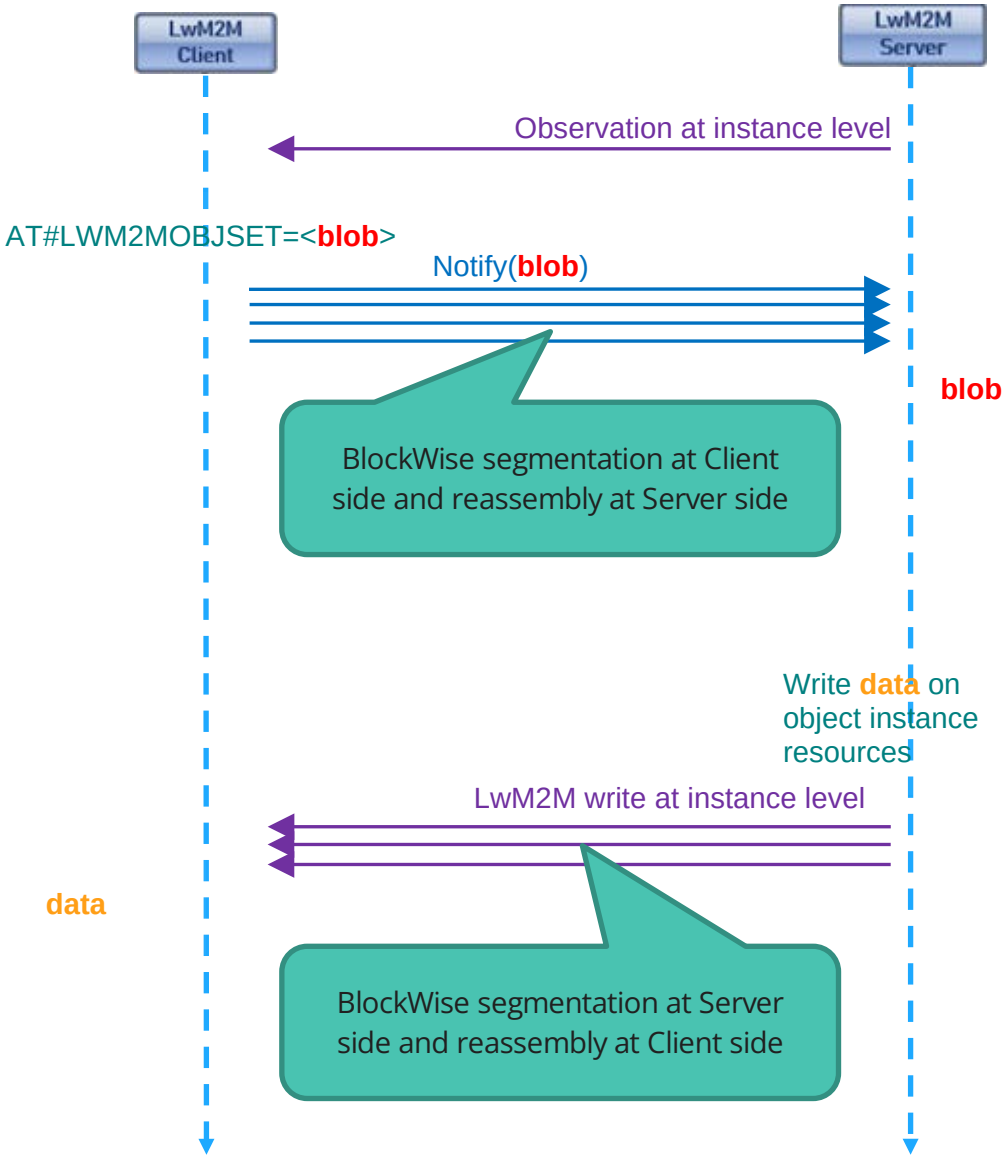



Figure 25: Example of BlockWise Flow

**Reference:**

- RFC 7959 https://www.rfc-editor.org/rfc/rfc7959

# AppZone APIs

The table below provides the matching between the AT commands of this chapter and the AppZone APIs for the same actions.

Table 14: list of AppZone APIs

| AT command | AppZone API |
|---|---|
| AT#LWM2MW | m2mb_lwm2m_write |
| AT#LWM2MSET | m2mb_lwm2m_set |
| AT#LWM2MR | m2mb_lwm2m_read |
| AT#LWM2MGET | m2mb_lwm2m_get |
| AT#LWM2ME | m2mb_lwm2m_exec |
| AT#LWM2MOBJSET | m2mb_lwm2m_objset |
| AT#LWM2MOBJGET | m2mb_lwm2m_objget |

More AppZone documentation resources are available on the AppZone section of paragraph Bootstrap and Registration.

Furthermore, a specific sample code for writing and reading all the types of resources is available on the following link

https://github.com/telit/IoT-AppZone-SampleApps/tree/master/ME310G1-ME910G1-ML865G1/AppZoneSampleApps-MAIN_UART/LWM2M

and another sample code to show how m2mb_lwm2m_objset and m2mb_lwm2m_objget work:

https://github.com/telit/IoT-AppZone-SampleApps/tree/master/ME310G1-ME910G1-ML865G1/AppZoneSampleApps-MAIN_UART/LWM2M_OBJ_SET_GET

## 7.2 Operations on Server Side

According to **Device Management and Service Enablement** interface, the LwM2M resources can be accessed from server side.

The next paragraphs describe the suitable actions.

### Writing

The LwM2M resources configured as **writable** (RW or W only) can be written from LwM2M server side.

The Figure 26 below shows the **Write** button on OneEdge IoT portal web interface.

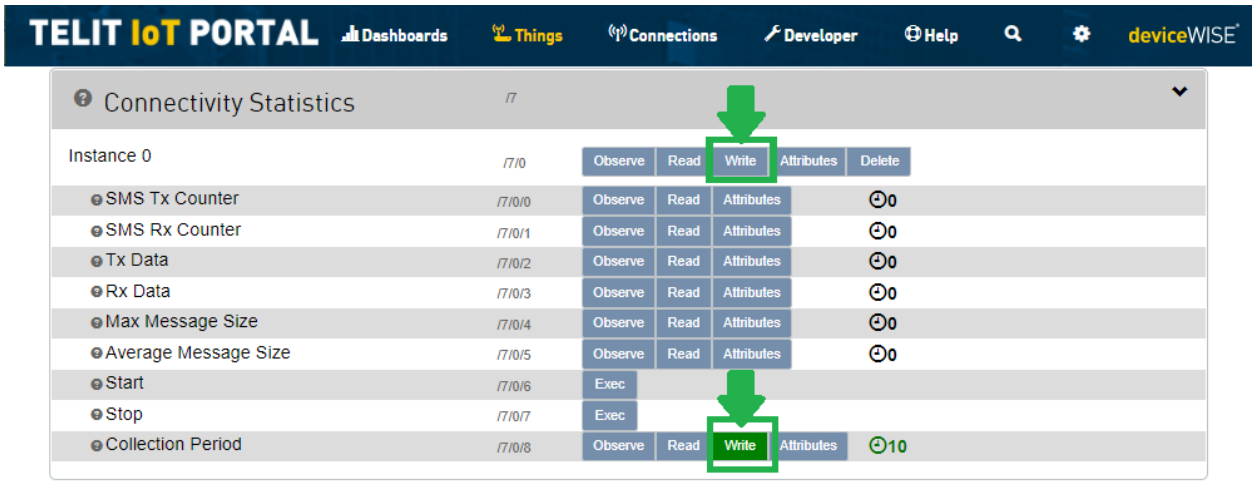



Figure 26: "Write" button on OneEdge IoT portal web interface.

The **write** operation is also applicable to instance and object level if at least a writable resource is present and allow to write multiple resources in one unique LwM2M command.

> **Note: OneEdge:** The **write** operation is available on OneEdge IoT portal also through the **lwm2m.device.oir.write** TR50 API.
>
> The example below is related to the write of value "10" to resource 7/0/8.

```
Request
{
    "data": {
```

```
        "command": "lwm2m.device.oir.write",
        "params": {
            "thingId": "<thingId>"
            "objId": 7,
            "instId": 0,
            "resId": "8",
            "encoding": 11542,
            "data": "10"
        }
    }
}
Response
{
    "data": {"success": true}
}
```

**Note:** The host application on client side can be configured to receive URCs on serial port when a resource is modified; refer to LWM2MMON AT command.

## AT#LWM2MMON

When the value of a resource is modified, as default the host application is not informed about the event.

By the way, the host application can configure the LwM2M client to receive a URC message on the serial port every time a resource is modified.

The **AT#LWM2MMON** command is used to enable or disable the **UPD** URC messages.

The syntax is:

```
AT#LWM2MMON=action,objId
```

This is very useful feature to manage the receiving of remote settings on host application side.

Example, already reported on **OneEdge Getting started** guide.

```
AT#LWM2MMON=1,7
OK
```

Here the portal User writes the value “10” to the resource /7/0/8 on the OneEdge IoT portal User interface.

The host application receives the following message:

```
#LWM2MMON: UPD,"/7/0/8/0"
```

Since the URC does not contain the value of the resource, the host application can achieve it through a “Read” operation.

```
AT#LWM2MR=0,7,0,8,0
```

```
#LWM2MR: 10
OK
```

## Reading

The LwM2M resources configured as **readable** (RW or R only) can be read from LwM2M server side.

The Figure 27 below shows the **Read** button on OneEdge IoT portal web interface.

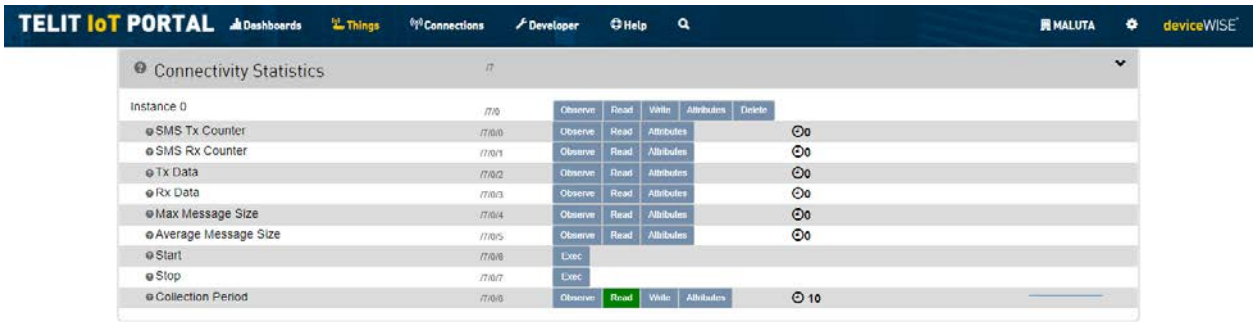



Figure 27: "Read" button on OneEdge IoT portal web interface.

The "read" operation is also applicable to instance and object level if at least a readable resource is present and allow to read multiple resources in one unique LwM2M command.

> **Note: OneEdge:** The **read** operation is available on OneEdge IoT portal also through the "lwm2m.device.oir.read" TR50 API.
>
> The example below is related to the read of value all the resources of instance 0 of object 7.

```
Request
{
    "data": {
        "command": "lwm2m.device.oir.read",
        "params": {
            "thingId": "621e4f01447cfb1b46227a14"
            "objId": 7,
            "instId": 0,
            "encoding": 11542,
        }
    }
}
Response
{
"data": {
  "success": true,
  "params": {
    "result": {
```

```
      "7":      {
        "0":      {
          "0":{"ts":"2022-03-03T17:32:45Z","value":0},
          "1":{"ts":"2022-03-03T17:32:45Z","value":0},
          "2":{"ts":"2022-03-03T17:32:45Z","value":0},
          "3":{"ts":"2022-03-03T17:32:45Z","value":0},
          "4":{"ts":"2022-03-03T17:32:45Z","value":0},
          "5":{"ts":"2022-03-03T17:32:45Z","value":0},
          "8":{"ts":"2022-03-03T17:32:45Z","value":10}
                }
              }
            }
           }
       }
}
```


## Executing

The LwM2M resources configured as "**executable**" (E) can be executed from LwM2M server side.

The Figure 28 below shows the "**Exec**" button on OneEdge IoT portal web interface.

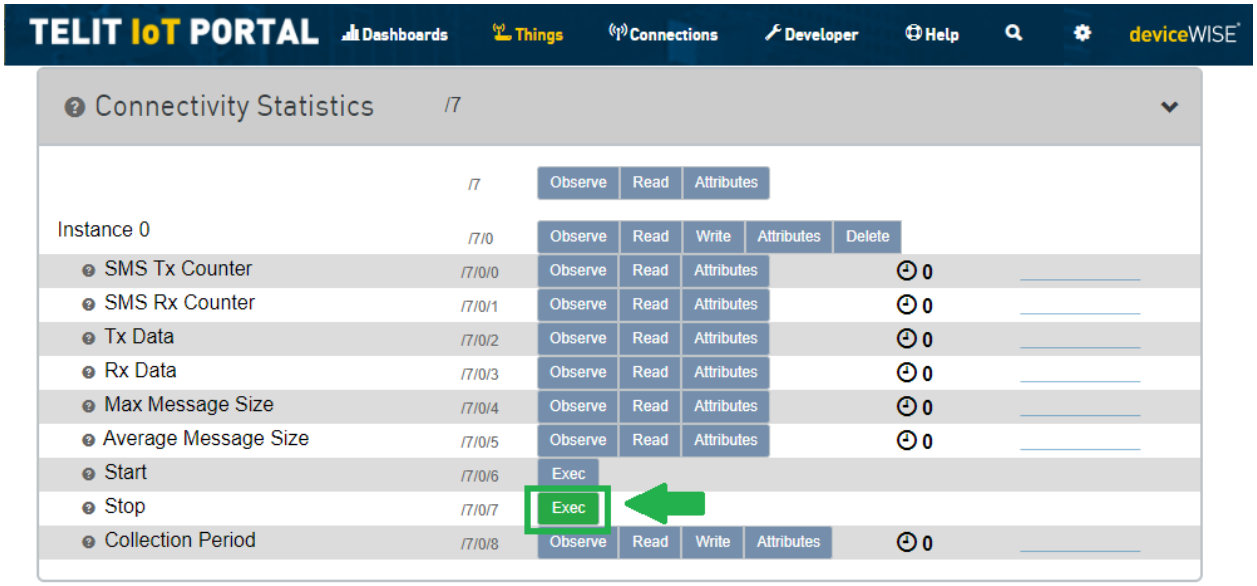



Figure 28: "Exec" button on OneEdge IoT portal web interface

**Notes: OneEdge:** The **exec** operation is available on OneEdge IoT portal also through the "lwm2m.device.oir.exec" TR50 API.

The example below is related to the execution of the resource 7/0/7.


```
Request
{
    "data": {
```

```
        "command": "lwm2m.device.oir.exec",
        "params": {
            "thingId": "<thingId>"
         "objId": 7,
            "instId": 0,
            "resId": "7"
        }
    }
}
Response
{
    "data": {"success": true}
}
```

**Note:** The host application on client receive the following URC on serial port when a resource is executed

```
#LWM2MINFO:"GEN","EXEC: /objId/instId/resId/resInstId"
```

Refer to *OneEdge Getting started* guide. Paragraph 4.3.1.3

# 8 LwM2M Observations in OneEdge

## 8.1 Operations on Server Side

### Observe Request

According to Information Reporting interface, a LwM2M resource (or an instance or an object) can be monitored from server side through the Observation functionality, activated by the **observe** command.

According to the OMA specification, when the server sets up an observation on a resource (or an instance, or an object) without any attributes as default, the client sends a notify message to the server every time the observed URI changes.

The notify message contains itself the new value.

The Figure 28 below shows the **Observe** buttons on OneEdge IoT portal interface.

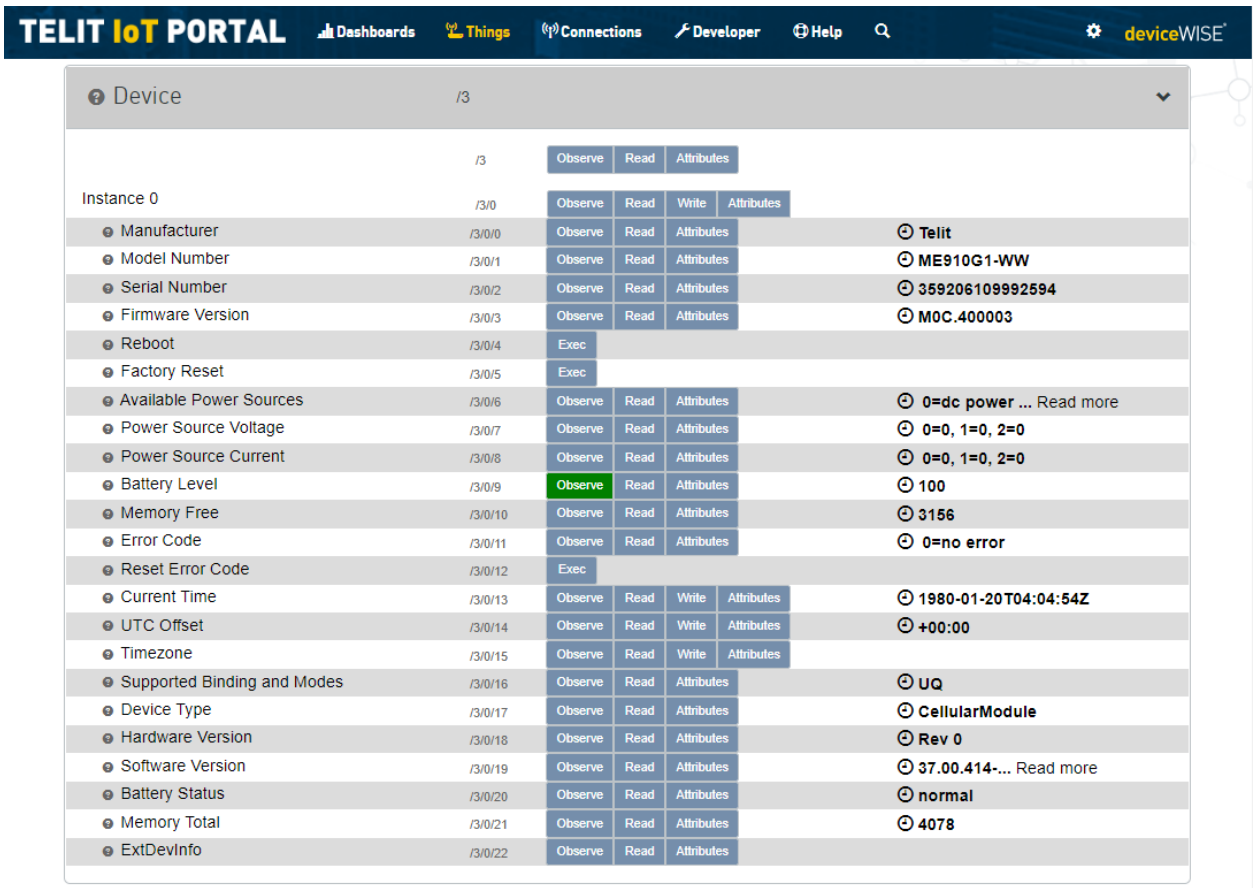



Figure 29: "Observe" button for resource 3/0/9 on OneEdge IoT portal interface.

**Note: OneEdge:** The **observe** operation is available on OneEdge IoT portal also through the **lwm2m.device.oir.observe TR50** API.

The example below is related to the observation request for the resource 7/0/4.

```
Request
{
    "data": {
    "command": "lwm2m.device.oir.observe",
```

```
        "params": {
            "thingId": "<thingId>"
            "objId": 3,
            "instId": 0,
            "resId": "9"
        }
    }
}
Response
{
    "data": {
        "success": true,
        "params": {"token": "Jzt0CO0K"}
        }
}
```

Observations are one of the most features in IoT applications and can be tuned to optimize the consumption of the data exchanged.

## Observe Attributes

The default rule for the observation functionality is that the LwM2M client must send a notify message to the server every time the observed URI value changes.

Five parameters, called in LwM2M specification as “attributes”, are present to customize special behaviors on the client rules.

The attributes are listed below:

- **Minimum period (pmin)** and **Maximum period (pmax)**: values expressed in seconds; they define a time range within which the client sends a notification in case the resource value changes. The time count starts from the last change of the value of the resource. In case the resource value changes before the minimum period, the notification will be sent only after the Minimum period. When the elapsed time reaches the Maximum period, a notification will be sent even if the value is not changed.
- **Step (st)**: the client sends a notification to the server if the difference between the old and the new value is equal to or greater than the step value.
- **Greater than (gt)** and **Less than (lt)**: a notification is sent only if the new value crosses one of these thresholds.

> **Warning:** Step, Greater than or Less than are considered only when they fall within the time range defined by “Minimum Period” and “Maximum Period”.

> **Note:** The five attributes can be set separately or in different combinations.

Refer to the below that shows the **Write-attributes** button and the popup with the edit-box fields on OneEdge IoT portal web interface.

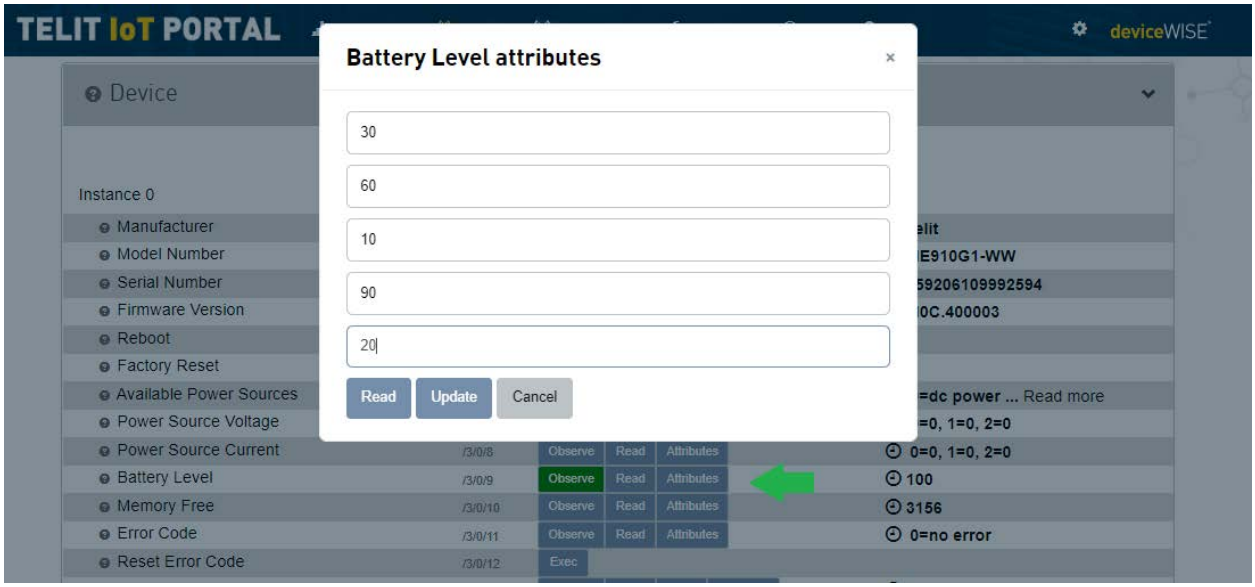



Figure 30: "write attributes" button and popup on OneEdge IoT portal.

**Note: OneEdge:** The **write-attribute** operation is available on OneEdge IoT portal also through the **lwm2m.device.oir.write.attr** TR50 API.

The example below is related to the observation request for the resource 7/0/4.

```
Request
{
    "data": {
        "command": "lwm2m.device.oir.write.attr",
        "params": {
            "thingId": "<thingId>"
            "objId": 3,
            "instId": 0,
            "resId": "9",
            "pmin": "30",
            "pmax": "60",
            "st": "10",
            "gt": "90",
            "lt": "20"
        }
    }
}
Response
{
    "data": {"success": true}
}
```

# Cancel Observe

An observation in progress can be stopped through the “cancel observe” LwM2M command.

On OneEdge IoT portal web interface the “observe” and “cancel observe” commands share the same **“Observe” button** in Figure 26 in a **toggle logic**: the LwM2M server sends an observation request if no active observation is present and sends a “cancel observe” if the observation is already active.

> **Note: OneEdge:** The **cancel observe** operation is available on OneEdge IoT portal also through the **lwm2m.device.oir.observe_cancel** TR50 API.
>
> The example below is to cancel the observation in place with token **Jzt0CO0K** on the resource 7/0/4

```
Request
{
    "data": {
        "command": "lwm2m.device.oir.observe",
        "params": {
            "thingId": "<thingId>"
            "objId": 7,
            "instId": 0,
            "resId": "4",
            "token": "Jzt0CO0K"
        }
    }
}
Response
{
    "data": {"success": true}
}
```

# Strategies Related to Observations

## Observation at Resource Vs Instance level

Since the observations can be set on resource and instance level, the User should develop its application according to advantages and disadvantages of them.

Using an observation at **resource level**, when the observed resource changes, the LwM2M client sends a notify to the server with **only the value of the observed resource** in the payload (refer to the first part of the Figure 30 below).

On the other hand, using an observation at **instance level**, when any of the resources in the instance change, the LwM2M client sends a notify to the server with **all the resource values** in the payload (refer to the second part of the Figure 31 below).

So, in case of observation at instance level, the payload of the notify is typically bigger than one at resource level.

By the way, the best **advantage** in case of observation on instance level is that multiple resources can be set all together through **AT#LWM2MOBJSET** command and this causes the sending of one unique notify with all the resource values in the payload instead of sending multiple messages (refer to the third part of the Figure 30 below): this is the most optimized way to save data and power consumption in sending data.

As showed in the second part of the Figure 30 below, the setting of a single resource is still possible in case of observation at instance level, but not convenient if the single resources are set in short time.

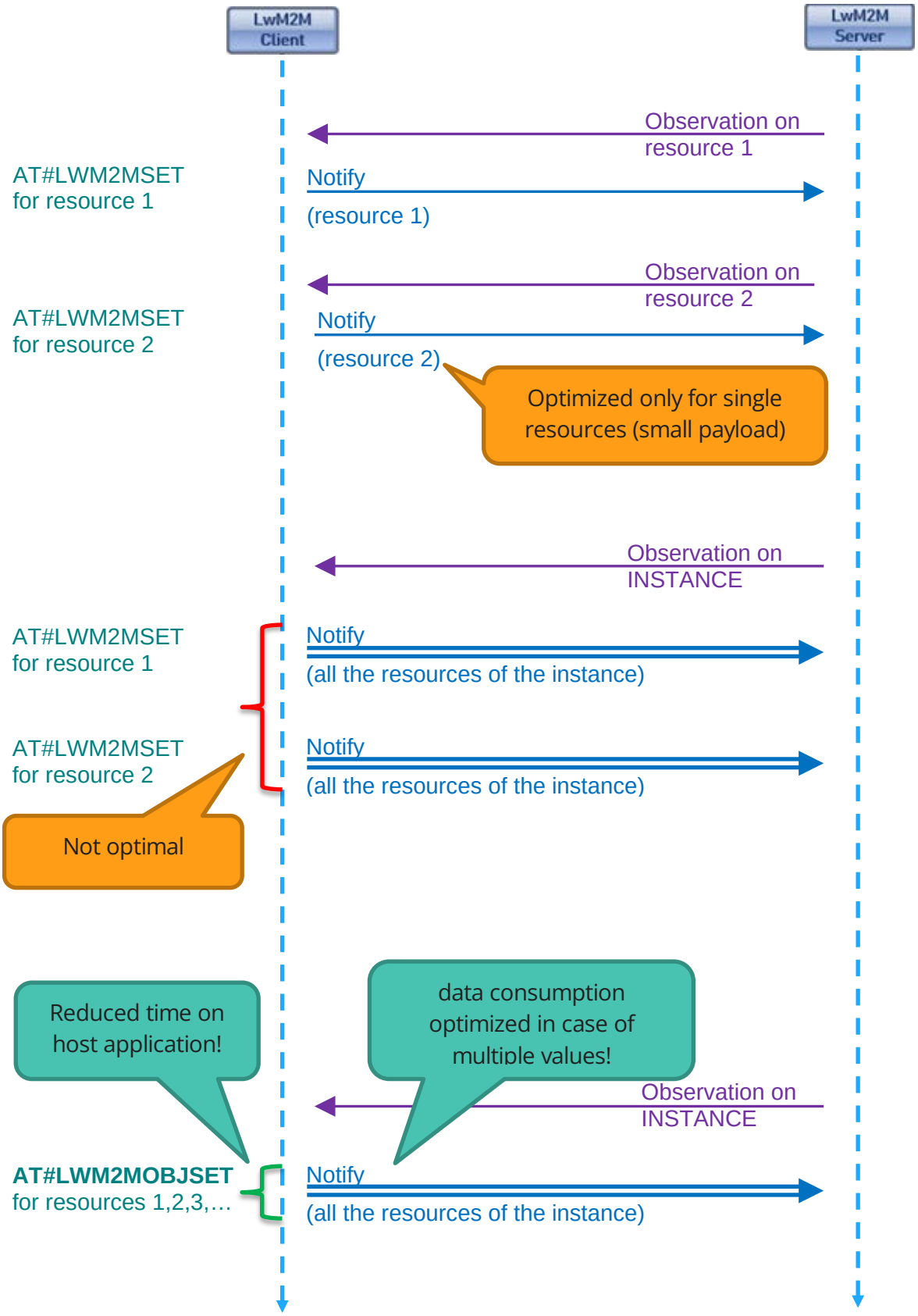



Figure 31: Observation on resource Vs instance

Related to the discussion and the Figure 30 above, the following general guidelines are suggested:

**Metering use case**

- A metering device usually sends a set of sensors values.
- The sampling timestamp of the sensors is usually the same.

- The suggested strategy is the **observation at instance level** and set the sensor data at instance level through **AT#LWM2MOBJSET** (third part of the Figure 30)

**Alarm use case (single alarm)**

- An alarm device usually sends **one** info mapped on a resource when an event occurs.
- The suggested strategy is the **observation at resource level** and set the alarm info at resource level through **AT#LWM2MSET** (first part of the Figure 30)

**Alarm use case (multiple alarms)**

- If the device must send the status of some sensors or other data related to the alarm
- The suggested strategy is putting the alarm info and all the sensors in the same object instance, set the **observation at instance level** and then send all of them through **AT#LWM2MOBJSET** command that will send one unique notify (third part of the Figure 30)

In case of doubts, the OneEdge technical support team (ts-oneedge@telit.com) is available to provide application feedbacks and data model review.

## Observation Attributes Use Cases

The attributes of the observations are a very powerful way to customize remotely the logics of the applications.

The example below is a use case where the observation attributes are used to optimize the data consumption.

Let's consider a LwM2M application developed to monitor the battery level of a device.

The host application is designed to **measure periodically the voltage of the battery** and **write** the corresponding percentage value on **the battery level resource** (/3/0/9).

On server side, an **observation on the same "battery level" resource** is present without any attribute, such that every time the host application sets a new value of battery level percentage, the LwM2M client sends a notify.

After some time, the User wants to reduce the data consumption of the application, due to the notify messages.

Hence, some corrective actions can be applied, for example:

Without LwM2M, the User should modify the host application logic to write the "battery level" value less often, for example every 5 units, to have less notify messages:
**Cons**: every time the host application changes, it must be upgraded on the devices in the field.
With LwM2M, the User should configure the observation attributes from server side, for example:
Step=5, to configure the LwM2M client to send the notify messages only if the "battery level" value gap is at least every 5 units rather the previous one.
Lt=20, to configure the LwM2M client to send the notify messages only if the "battery level" value crosses the 20 threshold.

Pmin=600, to configure the LwM2M client to send the notify messages not before 600 seconds (10 minutes) from the previous one.
Any combination of the settings above.
Pro: the **new attribute configuration** is applied **remotely** from server side, so no need to change the host application code.
Pro: **the logic is centralized on the server**, so the change is applied only in one place.

## 8.2 Operations on Client Side

### AT#LWM2MNFYLIST

If an application is designed to send data through observations, the host application must be sure that the **observation** is established **before to write data** on the resources, otherwise the LwM2M notify message will not be sent by the LwM2M client to the server.

For this reason, the command AT#LWM2MNFYLIST provides the list of all the observations set on the LwM2M client: if the host application finds the specific URI, it can write the value to the URI to cause the LwM2M client sending the LwM2M notify message.

The syntax is according to parameters below:

```
AT#LWM2MNFYLIST=agentInst[,shortServerId]
```

The “agentInst” parameter must be set to 0 for Telit client

The **shortServerId** optional parameter can be retrieved from registration URC messages or AT#LWM2MREG=0,3 response (ref paragraph 5.1.4): for **OneEdge it’s 99**.

Example:

If no observation is present:

```
AT#LWM2MNFYLIST=0
#LWM2MNFYLIST:
99,"EMPTYLIST"
OK
```

After that an observation is configured from server side on resource 3/0/9, the URI is listed in the command response:

```
AT#LWM2MNFYLIST=0
#LWM2MNFYLIST:
99,3,0,9
OK
```

**Note:** This command is very useful in battery-powered devices that must be optimize all the actions in the shortest time as possible (refer to example in Figure 32).

# AT#LWM2MNFYACKURI and AT#LWM2MNFYACKENA

According to the paragraphObservation Attributes Use Cases when an observation is set for a specific resource without attributes, the LwM2M client sends a notify message to the server every time the resource value changes.

This is the main functionality to send data from the remote device to the LwM2M server, like in a metering use case.

Hence, every time the host application writes the “metering” data to the observed resource, the LwM2M client detects the variation of the value and sends the notify message to the LwM2M server.

As default, the host application has no feedback regarding the notify sent, so it cannot evaluate if the notify message was successfully delivered or not.

By the way, the host application can enable the “notify ACK” functionality to **receive feedback on the serial port about the delivery of the notify**.

The commands to be used are the two below.

The command AT#LWM2MNFYACKURI is used to add, remove or list the URIs to be monitored on host application side:

```
AT#LWM2MNFYACKURI=agentInst,operation,objId,objInstId[,resId]
```

The operation parameter must be set according to the following values:

- 0: remove URI
- 1: add URI
- 2: list URIs

The command AT#LWM2MNFYACKENA is used to enable or disable the functionality:

```
AT#LWM2MNFYACKENA=agentInst,action
```

The action parameter must be set according to the following values:

- 0: disable URC reporting
- 1: enable URC reporting
- 2: read URC reporting status

The URC message is in the format:

```
#LWM2MNFYACK:agentInst,SSID,"/objId/objInstId/resId","ACK"
```

The last parameter is "NACK" in case the LwM2M client doesn’t receive any CoAP ACK message from the server related to the notify.

The example below is related to the monitoring of the battery level (resource /3/0/9 of the Device object):

The host application enables the “notify ACK” functionality on the specific resource:

```
AT#LWM2MNFYACKURI=0,1,3,0,9
AT#LWM2MNFYACKENA=0,1
```

As precondition, the host application must be sure that the observation on 3/0/9 resource has been set from server side so the AT#LWM2MNFYLIST command can be used:

```
AT#LWM2MNFYLIST=0
#LWM2MNFYLIST:
99,3,0,9
OK
```

Considering 100 as current value of the resource, the host application writes the new value 80:

```
AT#LWM2MSET=0,3,0,9,0,80
OK
```

The LwM2M client sends the LwM2M notify message related to the value 80.

If the LwM2M client receives the expected ACK at CoAP level from the server, it means that the server successfully received the notify and the host application receives the ACK URC:

```
#LWM2MNFYACK: 0,99,"/3/0/9","ACK"
```

Otherwise, in case of lossy data channel, the notify message and/or the ACK can be lost so the host application receives the NACK URC:

```
#LWM2MNFYACK: 0,99,"/3/0/9","NACK"
```

**Warning:** Another precondition to receive the "notify ACK" on serial port is that the URI on the observation and the URI on the AT#LWM2MNFYACKURI must match.

**For example:**

observation on 3/0/9 and "AT#LWM2MNFYACK" on 3/0/9:

#LWM2MNFYACK: 0,99,"/3/0/9"

observation on 3/0 and "AT#LWM2MNFYACK" on 3/0:

#LWM2MNFYACK: 0,99,"/3/0"

observation on 3/0/9 and "AT#LWM2MNFYACK" on 3/0:

#LWM2MNFYACK not sent on serial port

observation on 3/0 and "AT#LWM2MNFYACK" on 3/0/9:

#LWM2MNFYACK not sent on serial port

Table 15: LWM2MNFYCK Observation

| #LWM2MNFYACK URC message on the serial port | | Observation | |
|---|---|---|---|
| | | /3/0 | /3/0/9 |
| AT#LWM2MNFYACK | /3/0 | #LWM2MNFYACK: 0,99,"/3/0" | #LWM2MNFYACK not sent |
| | /3/0/9 | #LWM2MNFYACK not sent | #LWM2MNFYACK: 0,99,"/3/0/9 |

An example of flow diagram related to AT#LWM2MNFYLIST and ACK management to optimize the time in a battery powered device is showed in Figure 32 below:

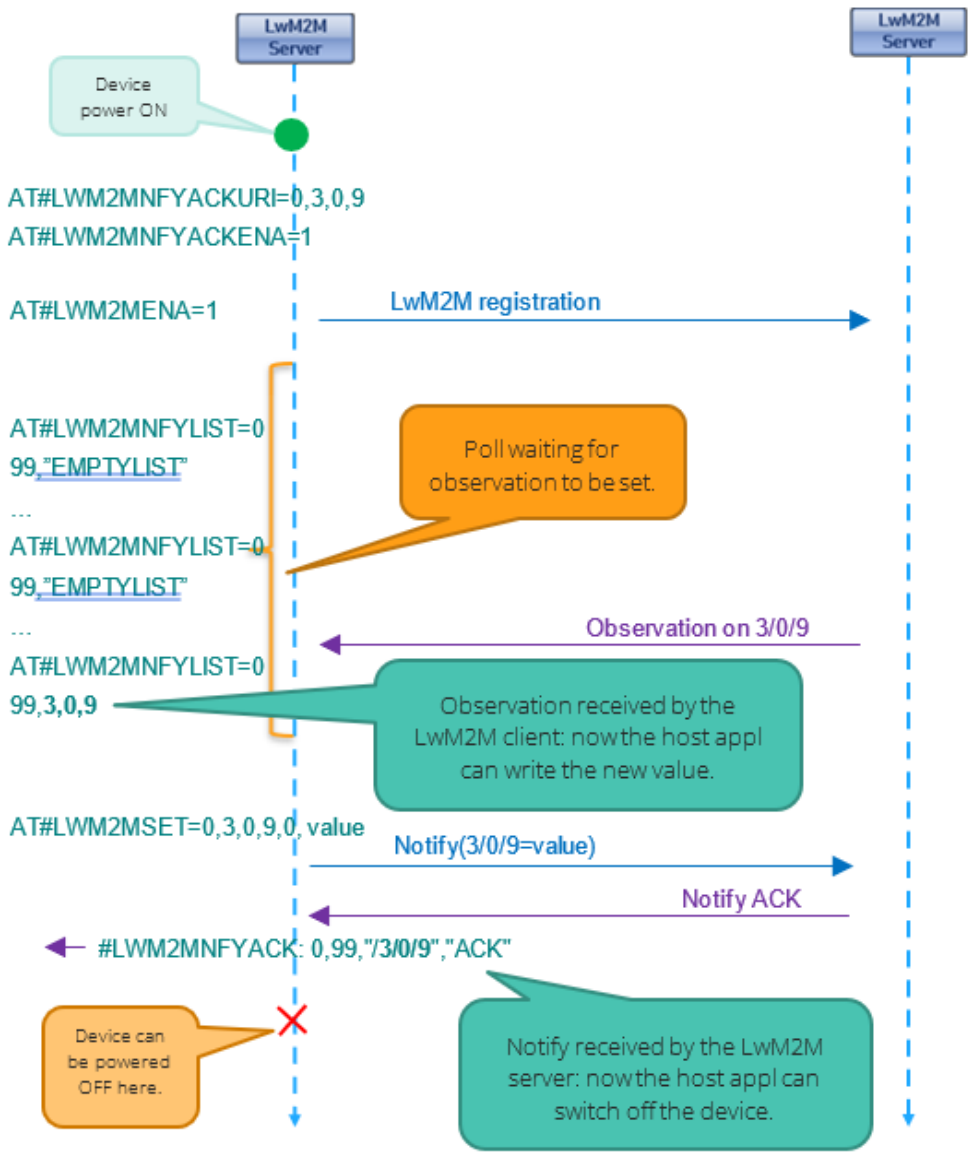



Figure 32: Example of optimized timing for battery powered device.

# AppZone APIs

The table below provides the matching between the AT commands of this chapter and the AppZone APIs for the same actions.

Table 16: list of AppZone APIs

| AT command | AppZone API |
|---|---|
| AT#LWM2MNFYLIST | m2mb_lwm2m_nfylist |
| AT#LWM2MNFYACKURI | m2mb_lwm2m_nfy_ack_uri,<br>m2mb_lwm2m_nfy_ack_list, |
| AT#LWM2MNFYACKENA | m2mb_lwm2m_nfy_ack_status |
| #LWM2MNFYACK | M2MB_LWM2M_NFYACK_INFO_IND event on the LwM2M callback |

More AppZone documentation resources are available on the paragraph Bootstrap and Registration.

# 9 LwM2M Objects in OneEdge

The Telit LwM2M client supports 3 categories of LwM2M objects:

- **Basic OMA objects**: range 0-1023, always **available as default** and built in in the LwM2M client code (no need to inject them as XML files); refer to paragraph 6.1.
- **OneEdge Telit proprietary objects**: private range 33205-33254, provide specific and valuable features to OneEdge environment, always **available as default** and built in in the LwM2M client code (no need to inject them as XML files); refer to paragraph 6.2.
- **Custom objects**: range 2048-10240 for IPSO objects and range 26241-32768 for free custom objects; these objects **must be injected as XML files into the modem files system** through the procedure.

A more detailed description related to custom objects is available at chapter 10.

**Note:** The objects supported as built in as default (first and second categories) are managed internally by the LwM2M client, so the resources are written and consumed from the LwM2M client itself: for example, the resource RSSI (/4/0/2) is written by the LwM2M client according to the values coming from the RF chip on the modem. On the other hand, the custom objects (third category) must be managed by the host application: for example, managing the Temperature IPSO object (3303), the host application must retrieve the value of the temperature according to its physical transducer/sensor and then it must write the value into the Sensor value resource (/3303/x/5700) through AT#LWM2MSET command.

## 9.1 Default LwM2M OMA Objects

The list of OMA basic objects supported as default by the Telit LwM2M client is shown in the Table below and described one by one in next paragraphs:

**Table 17: list of OMA standard objects supported as default by Telit LwM2M client.**

| Name | Object ID | Description |
|---|---|---|
| LWM2M Security | 0 | Provides the keying material of a LwM2M Client appropriate to access a specified LwM2M Server. |
| LwM2M Server | 1 | Provides the data related to a LwM2M Server. A Bootstrap-Server has no such an Object Instance associated to it. |
| LwM2M Access Control | 2 | Access Control Object is used to check whether the LwM2M Server has access right for performing an operation. |
| Device | 3 | Provides a range of device related information which can be queried by the LwM2M Server, and a device reboot and factory reset function. |
| Connectivity Monitoring | 4 | Enables monitoring of parameters related to network connectivity. RSSI, RSSQ, etc. |
| Firmware Update | 5 | Enables management of firmware which is to be updated. |

| Name | Object ID | Description |
|---|---|---|
| Location | 6 | Provides location information based on GNSS data. |
| Connectivity Statistics | 7 | Enables client to collect statistical information and enables the LwM2M Server to retrieve this information, set the collection duration and reset the statistical parameters. |
| Cellular connectivity | 10 | This object specifies resources to enable a device to connect to a 3GPP bearer. |
| APN connection profile | 11 | Specifies resources to enable a device to connect to an APN. |
| CoAP Config | 500 | This object is used to configure the CoAP transmission parameters (RFC7252) and other parameters that affect CoAP transmission. |

OMA specification can be found at following link:
https://technical.openmobilealliance.org/index.html.

All the XML files of the OMA objects can be found at the following link:
https://technical.openmobilealliance.org/OMNA/LwM2M/LwM2MRegistry.html.

**Note:** These default object definitions are hard coded into the LwM2M client so the XML files for them mustn't be injected as custom objects.

## Security (Object ID 0)

This object provides the keying material of a LwM2M Client to access a specified LwM2M Server.

Most important resources are: LwM2M Server URI, Security Mode, PSK Identity and Key.

This object must only be changed by a LwM2M Bootstrap Server during the Bootstrap procedure and must not be accessible by any other LwM2M Server.

Also the host application cannot access this object.

**Reference on OMA specification:**

- OMA-TS-LightweightM2M-V1_0_2-20180209-A
- Appendix E.1

## Server (Object ID 1)

This object provides the data related to a LwM2M Server.

Most important resources are Lifetime, Binding, Notification Storing.

Lifetime is one of the most important parameters of the LwM2M protocol and it's explained in paragraph 5.1.1.

Since these resources are managed internally by the LwM2M client, the host application cannot write them.

**Reference on OMA spec:**

- OMA-TS-LightweightM2M-V1_0_2-20180209-A

- Appendix E.2

## Access Control List (Object ID 2)

Access Control Object is used to check whether the LwM2M Server has access right for performing an operation.

It can be used to define different access rights for different objects across different LwM2M servers: for example, Server A can read object 123, Server B can read and write the same object 123.

Since OneEdge environment has only one LwM2M DM server, this object is not used.

**Reference on OMA specification:**

- OMA-TS-LightweightM2M-V1_0_2-20180209-A
- Appendix E.3 LwM2M Object: Access Control

## Device (Object ID 3)

This object provides a range of device related information which can be queried by the LwM2M Server, and a device reboot and factory reset function.

Most of these resources are **managed internally** by the LwM2M client, so the host application cannot write them, for example the manufacturer, model and serial number, FW and SW version.

By the way, other resources must be **managed by the host application** through AT commands; the list is below:

- Resource 6: Available Power Sources
- Resource 7: Power Source Voltage
- Resource 8: Power Source Current
- Resource 9: Battery level as percentage
- Resource 13: Current Time in Unix Epoch format
- Resource 14: UTC offset
- Resource 20: Battery Status

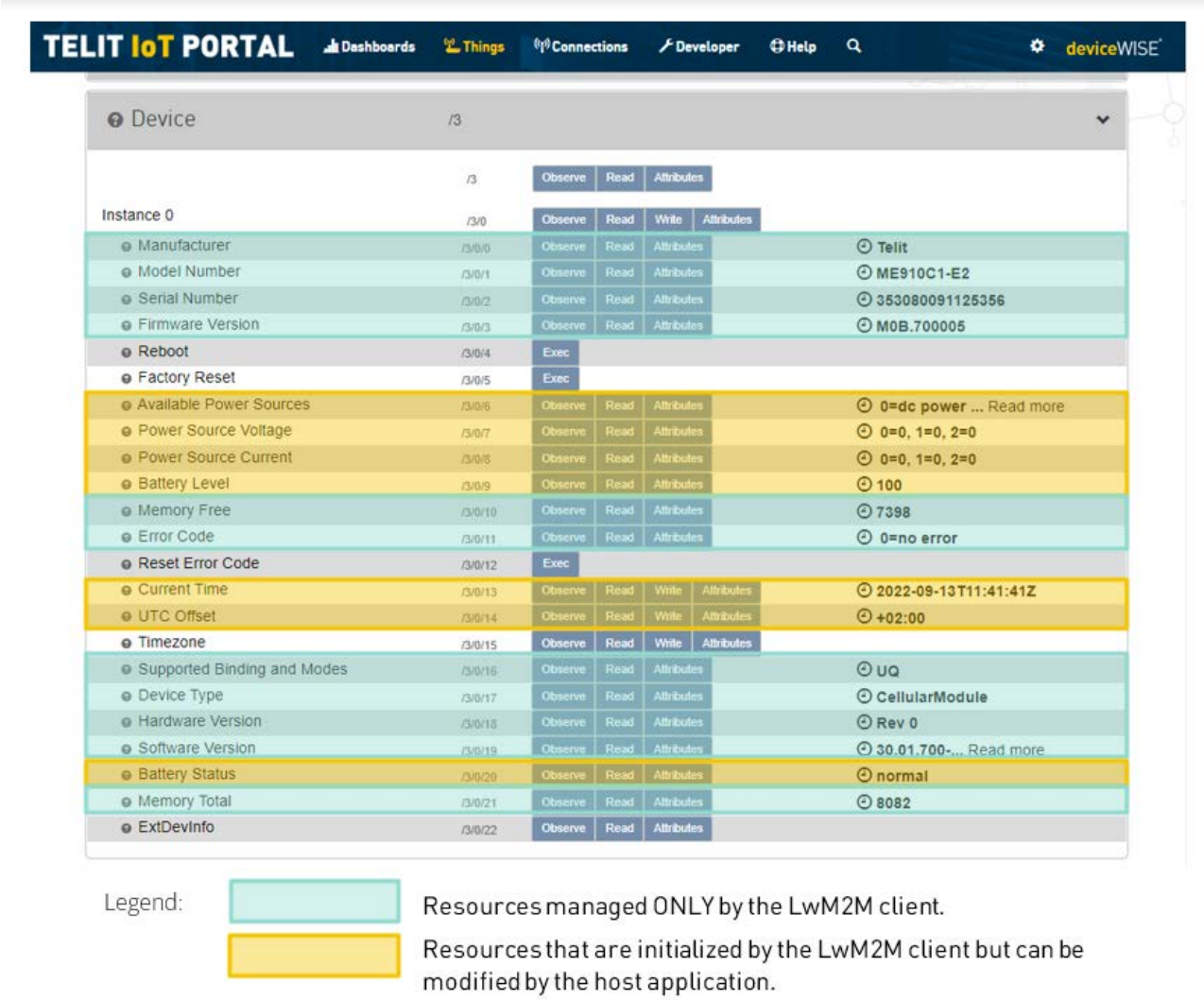



Figure 33: Different management for the resources of Device object

**Reference on OMA specification:**

- OMA-TS-LightweightM2M-V1_0_2-20180209-A
- Paragraph: Appendix E.4 LwM2M Object: Device

In the following paragraphs some specific resources will be explained.

## Reboot Behavior

The **Reboot** resource, executed from server side, causes the LwM2M client to perform a **power cycle** of the Telit modem.

The following URC is printed out on the serial port:

```
#LWM2MINFO: "GEN","DEVICE REBOOT"
```

At the next power up, the LwM2M client will perform a new **Registration** procedure.

> **Danger:** Be aware that the LwM2M client doesn't start automatically after the power up, so the LwM2M communication between client and server will be broken until the LwM2M client is enabled again through the AT#LWM2MENA=1 command on device side.

## Factory Reset Behavior

The **Factory Reset** resource, executed from server side, causes the LwM2M client to perform a **factory reset** of the Telit modem: all the modem and LwM2M settings will be restored to default values and the device will perform a power cycle.

The same URC as "**reboot**" is reported on serial port:

#LWM2MINFO: "GEN","DEVICE REBOOT"

Furthermore, all the values in the LwM2M object repository will be reset to default values.

At the next power up, the LwM2M client will perform a new **Bootstrap** procedure.

> **Danger:** Be aware that the LwM2M client doesn't start automatically after the power up, so the LwM2M communication between client and server will be broken until the LwM2M client is enabled again through the AT#LWM2MENA=1 command.

## Battery Level and Battery Status Behavior

These resources 3/0/9 and 3/0/20 don't contain the real data from the battery supply, but they are just containers to be filled by the host application according to its battery level management.

In particular, the "Battery Level" resource is an integer number as the percentage of the battery capacity and the default value in Telit LwM2M client is 100.

The host application developer should convert the voltage signal of its device circuit to the percentage.

Example to set the battery level to 90%:

AT#LWM2MSET=0,3,0,9,0,90

The "Battery Status" resource must contain an integer number to represent a high level status information about a battery.

According to the LwM2M OMA specifications, numbers from 0 to 6 are mapped to the following meanings:

0 = Normal (default value)

1 = Charging

2 = Charge Complete

3 = Damaged

4 = Low Battery

5 = Not Installed

6 = Unknown

Example to set the battery status to "Damaged":

AT#LWM2MSET=0,3,0,20,0,3

## Current Time Behavior

This resource 3/0/13 is managed internally by the LwM2M client and its value is a mirror of modem RTC value in Unix Epoch format (number of seconds from 1970).

The RTC time is aligned to the AT+CCLK and AT#CCLK time setting.

Since the RTC counts the seconds starting from its current power session, the modem time could be not aligned to the real current time.

The first date on RTC is 06/01/1980.

> **Note:** The good synchronization of the time on the device is very important in IoT devices that performs data sampling, in order to assign the right time to the data samples, for example Time-series use case.

Below are listed some different procedures to synchronize the time on the modem RTC.

## NITZ

Enabling the **NITZ** service through the **AT#NITZ=7** command, the modem can get the right time from the cellular network signaling and the RTC and so also the Current Time resource is updated.

The **#NITZ** service is enabled as default at start up in Telit devices.

Example:

If the NITZ service is not enabled or it's enabled but the network doesn't provide the service, the RTC starts from year 1980:

```
AT+CCLK?
+CCLK: "80/01/06,00:00:30+00"
OK
```

The Current time resource reports the number of seconds for the current date:

```
AT#LWM2MR=0,3,0,13,0
#LWM2MR: 315964831
OK
```

When the **#NITZ** service is enabled, the network provides the new date to the modem at next cellular registration:

```
AT#NITZ=7,1
OK
AT+CFUN=4
OK
AT+CFUN=1
OK
#NITZ: 22/03/31,17:31:16+08
```

The RTC is updated with the new date from #NITZ service, and the **Current Time** resource is updated:

```
AT+CCLK?
+CCLK: "22/03/31,17:31:22+08"
OK
AT#LWM2MR=0,3,0,13,0
#LWM2MR: 1648740678
OK
```

**Warning:** Not all the networks support the #NITZ service, so the host application should consider another way or an alternative strategy.

## NTP

Enabling the **NTP** service through the **AT#NTP** command, the modem requests the current time to a NTP server. The modem must activate the PDP context to access the NTP server.

**Example:**

```
AT+CCLK?
+CCLK: "80/01/06,00:00:30+08"
OK
AT#LWM2MR=0,3,0,13,0
#LWM2MR: 315964830
OK
AT#SGACT=1,1
#SGACT: 100.81.18.187
OK
AT#NTP="0.it.pool.ntp.org",123,1,2,4
#NTP: 22/03/31, 16:45:37+04
OK
AT+CCLK?
+CCLK: "22/03/31,16:45:52+04"
OK
AT#LWM2MR=0,3,0,13,0
#LWM2MR: 1648741544
OK
```

## LwM2M Remote Setting

Since the **Current Time** resource is writable, the value can be overwritten on LwM2M Server side.

**Warning:** Since the new value is into the payload of a LwM2M message over UDP, the message delivery could be affected by an offset due to the transmission

and retransmission time. This offset is usually few seconds according to the latency of the cellular network. Host developers must consider this error in the Current Time accuracy when this clock management is used.

**Warning:** Not all the networks support the #NITZ service, so the host application should consider another way or an alternative strategy.

**Note: OneEdge:** A trigger on OneEdge IoT portal side can be used to read the Current Time resource value and write back to the device the new value in case the time has to be adjusted.

## Connectivity Monitoring (Object ID 4)

This object provides main information related to network connectivity.

Most important resources are: Network Bearer, RSSI, Link Quality, IP address, CellId, etc.

A Telit proprietary resource (Id 28000) is present to provided LAC/TAC information.

Since these resources are **managed internally** by the LwM2M client, the host application cannot write them.

The example below shows the matching between the values of Connectivity object and the same values on AT command responses in case of GSM.

```
AT#LWM2MOBJGET=0,4,0
{"bn":"/4/0/",
"e":[
{"n":"0","v":0},
{"n":"1/0","v":0},
{"n":"2","v":-77},                    (RSSI)
{"n":"3","v":0},                      (LinkQuality)
{"n":"4/0","sv":"::"},
{"n":"4/1","sv":"100.109.44.249"},    (IP address)
{"n":"7/0","sv":"nxt20.net"},
{"n":"7/1","sv":""},
{"n":"7/2","sv":""},
{"n":"7/3","sv":""},
{"n":"7/4","sv":""},
{"n":"7/5","sv":""},
{"n":"8","v":18062},              (CellId)
{"n":"9","v":10},                     (MNC)
{"n":"10","v":222},                   (MCC)
{"n":"28000","v":31066}               (LAC)
]
```

```
}
AT#SERVINFO
#SERVINFO: 41,-77,"vodafone
IT","22210",66,795A,05,1,,"II",46,6
OK
AT#RFSTS
#RFSTS: "222 10",41,
-77,795A,46,29,19,18,2,468E,"208090117108131","vodafone
IT",3,2
OK
AT#MONI
#MONI: vodafone IT BSIC:66 RxQual:0 LAC:795A Id:468E ARFCN:41
PWR:-77dbm TA:5
OK
AT+CGCONTRDP
+CGCONTRDP: 1,5,"","100.109.44.249",,"8.8.8.8","8.8.4.4"
OK
```

**Note: OneEdge:** The OneEdge IoT portal is able to detect new values of the resources of object 4 and provide a position on the world map on the OneEdge IoT portal according to a third-party location service. Refer to paragraph Map Tab in section Management Portal Application.

The accuracy of the position coming from the cellular serving cell is usually low (few kilometers).

According to application requirements, this accuracy could be good enough, otherwise a more precise position can be achieved from object 6.

**Reference on OMA specifications:**

- OMA-TS-LightweightM2M-V1_0_2-20180209-A
- Appendix E.5

## Firmware Update (Object ID 5)

This object provides settings and information related to management of firmware which is to be updated.

According to OMA specification, this object can manage only the upgrade of firmware of the modem, so not the image of an external processor or AppZone code.

In Telit modems, the FOTA upgrade is performed downloading a “special” file on the modem.

This file is called **delta** file.

Even if the size of the entire image of the firmware of the modem is about 25 MB, the maximum dimension of the delta file is about 3 MB.

The **Firmware Update** object is used to synchronize the FOTA state machine (Figure 33 below) between LwM2M client and server according to the OMA specification and it's **managed internally** by the LwM2M client.

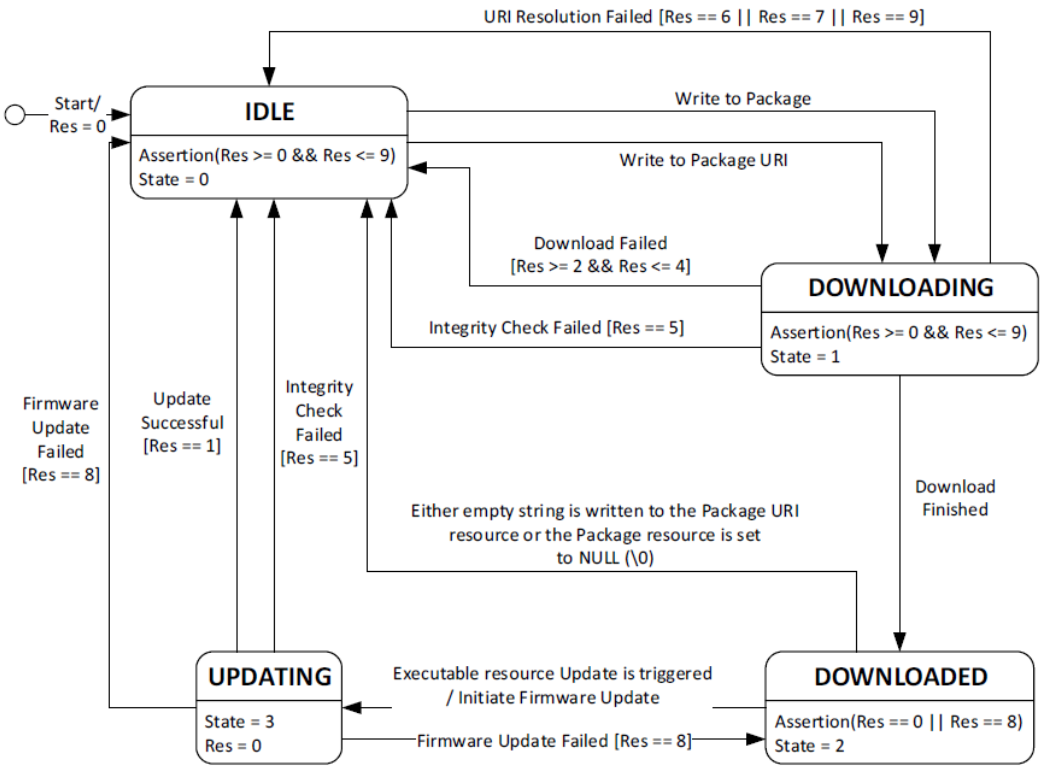



Figure 34: FOTA state machine

The basic state machine behavior is briefly explained below:

The **State** resource keeps the current state of the state machine.

When the server writes a URL on the **Package URI** resource, the state machine moves from **idle** to **downloading** state and the LwM2M client starts to download the delta file stored into the link in **Package URI** resource.

> **Warning:** Telit LwM2M client doesn't support the Package resource (Id 0), but only Package URI (Id 1), so it's not possible to send the target firmware content as opaque/binary flow.

The download is performed according to the available protocols in **Firmware Update Protocol Support** resource (Id 8): Telit LwM2M client supports only HTTP and HTTPS.

> **Warning:** Telit LwM2M client doesn't support the CoAP and CoAPS protocols, but only HTTP and HTTPS.

When the download of the delta is finished, the state machine moves from **downloading** to **downloaded** state and this value can be detected by the server side on the **State** resource. An observation here is useful to notify the value to the server.

When the state is **downloaded**, the server executes the **Update** resource and the client applies all the actions to perform the firmware upgrade.

The Telit modem will reboot few times and finally it will power up with the target firmware version.

Only in case of FOTA upgrade via LwM2M:

the **LwM2M client** will be **activated automatically** by the modem (AT#LWM2MENA command not needed by the host application) to restore the same working state like before the upgrade;

The LwM2M client will start **from Bootstrap**;

All **files**, also XML files, **are kept** by the FOTA process, so no need to inject them again;

Refer to OMA specification for more details and the complete list of available resources.

**Reference on OMA specification:**

- OMA-TS-LightweightM2M-V1_0_2-20180209-A
- Paragraph: Appendix E.6

> **Note: OneEdge:** A dedicated application is available on OneEdge IoT portal to manage the FOTA state machine on server side and it allows to run the same upgrade procedure to multiple devices at the same time. Refer to paragraph Manage LwM2M Customer Objects and OneEdge Firmware Management App application note.

## Location (Object ID 6)

The **Location** object provides location information based on GNSS data provided by the internal GNSS receiver on the Telit devices.

Most important resources are: Latitude, Longitude, Altitude, Speed, Timestamp and Radius.

The "Radius" resource is the accuracy of the GNSS data according to the size in meters of a circular area around the point.

Since these resources are **managed internally** by the LwM2M client, the host application cannot write them.

**Reference on OMA specifications:**

- OMA-TS-LightweightM2M-V1_0_2-20180209-A
- Paragraph: Appendix E.7

The GNSS receiver in Telit devices is disabled as default so a specific command must be used to switch on and off it.

Refer to **AT#LWM2MCFG** (paramId=6) in paragraph AT#LWM2MCFG or **Telit Services Management** object in paragraph Telit Services Management (object ID 33211). if paramId 6 is not supported yet in the current SW version.

## Connectivity Statistics (Object ID 7)

This object provides statistical information and enables the LwM2M Server to retrieve this information, set the collection duration and reset the statistical parameters.

Most important resources are Tx Data and Rx Data.

Since these resources are **managed internally** by the LwM2M client, the host application cannot write them.

The server can write the **Collection Period** resource and execute the **Start** and **Stop** resources.

**Reference on OMA specifications:**

- OMA-TS-LightweightM2M-V1_0_2-20180209-A
- Paragraph: Appendix E.8

## Cellular Connectivity (Object ID 10)

This object provides information and settings related 3GPP cellular connectivity.

Most important resources are: Power Saving modes, PSM and e-DRX timers.

These resources are managed internally by the LwM2M client but can be written also by the host application.

If the host application enables the PSM through AT+CPSMS command, the timers are reported on the specific resources.

The resources can be read and written from both host application and server side.

**Reference on OMA specifications:**

- OMA-TS-LWM2M_ConnMgmt-V1_0-20170314-A
- Chapter 5

## APN Connection Profile (Object ID 11)

This object provides information and settings to manage the APN on the modem.

The supported resources are: Profile name, APN, Enable status, Authentication Type and PDN type.

These resources are managed internally by the LwM2M client but can be written also by the host application.

The LwM2M client dynamically creates an instance for every APN configured on the modem: according to AT+CGDCONT command, the number of the APN is 6.

**Reference on OMA specifications:**

- OMA-TS-LWM2M_ConnMgmt-V1_0-20170314-A
- Chapter 6

## CoAP Configuration (Object ID 500)

This object is used to configure the CoAP transmission parameters (RFC7252) and other parameters that affect CoAP transmission.

The supported resources are: Network Bearer, Ack timeout and max retransmit.

Since this object allows multiple instances, the LwM2M client automatically creates one instance for each cellular RAT supported by the modem.

Furthermore, the instance IDs are linked to the RAT IDs of “Network bearer” resource of object 4 (4/0/0).

For example, in case of ME910C1-WW that supports GSM, CAT-M and NBIoT, the LwM2M client creates 3 instances with the IDs 0, 6 and 7.

The Figure 35 below shows the object default values related to the example above.

**Figure 35: Object 500 in case of ME910G1-WW**

Different settings of ACK timeout and maximum retransmissions are available for every instance/Network bearer.

This object is managed internally by the LwM2M client, but the values of the resources ACK timeout (500/x/1) and maximum retransmissions (500/x/3) can be modified by both the host application and server side.

The LwM2M client applies the new values to the CoAP management of the internal protocol stack at run-time.

The resource **Network bearer** cannot be modified because it's fixed internally by the LwM2M client.

The values of **ACK timeout** (500/x/1) and **maximum retransmissions** (500/x/3) can be customized by the User or host application according to current network performances.

**Reference:**

- RFC7242 Paragraph 4.2 and 4.8

## 9.2 OneEdge Telit Proprietary Objects

Telit developed some LwM2M custom objects to provide additional and valuable features to OneEdge applications.

They are Telit proprietary objects and registered in the private Telit OMA space in the range 33205-33254 of objects IDs.

The list of **Telit objects,** supported as default by the Telit LwM2M client, is shown in the Table below and described one by one in next paragraphs:

**Table 18: list of Telit proprietary objects supported as default by Telit LwM2M client.**

| Name | Object ID | Description |
|---|---|---|
| Telit Application Data | 33205 | Provides the ability to send application data from the device to the server. |
| Telit AT Command | 33206 | Provides the ability to execute AT commands on the module |
| Telit IDs | 33207 | Provides alternative unique IDs for the Telit module. |
| Connection Agent | 33210 | Provides the ability to manage the settings for the Connection Agent functionalities |
| Telit Services Management | 33211 | Provides the ability to enable or disable some services and check their status. |

**Note: OneEdge:** Telit objects are available only if the module is used in conjunction with Telit OneEdge service.

### Telit Application Data (Object ID 33205)

This object allows a host application to exchange **generic application data** with the portal through the **opaque** resources **Uplink data** and **Downlink data**.

The "**Uplink data**" resource is read only and must be used to send data from the client to the server.

Related to the opposite direction, the server must write the application data for the client on the **Downlink data** resource.

This object is managed by the host application.

Figure 36: Telit Application data object

# Telit AT Command (object ID 33206)

This object allows the LwM2M server to manage AT commands remotely.

Since this object is **managed internally** by the LwM2M client, the host application cannot modify its resources.

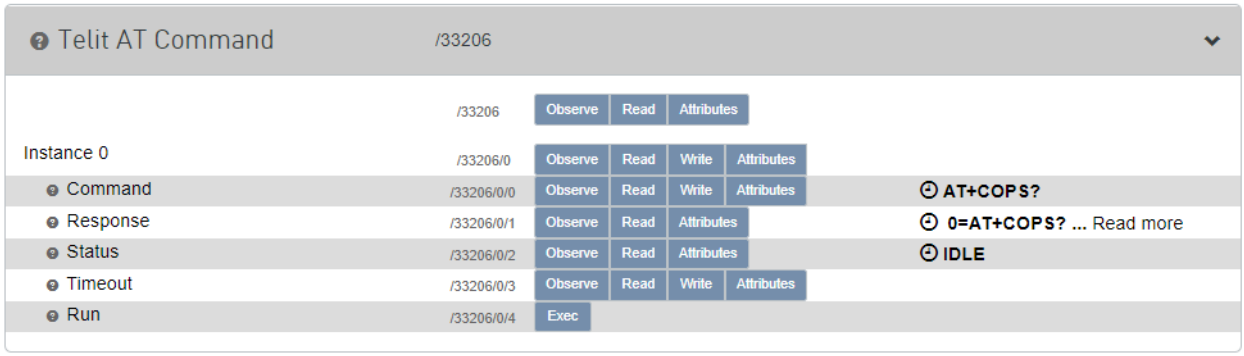



Figure 37: Telit AT command object

**Warning:** The ATcommands with a prompt are not supported.

**Note: OneEdge:** In OneEdge environment, the state machine on server side is already available as built-in on OneEdge IoT portal on Remote AT tab or in the DM Application.

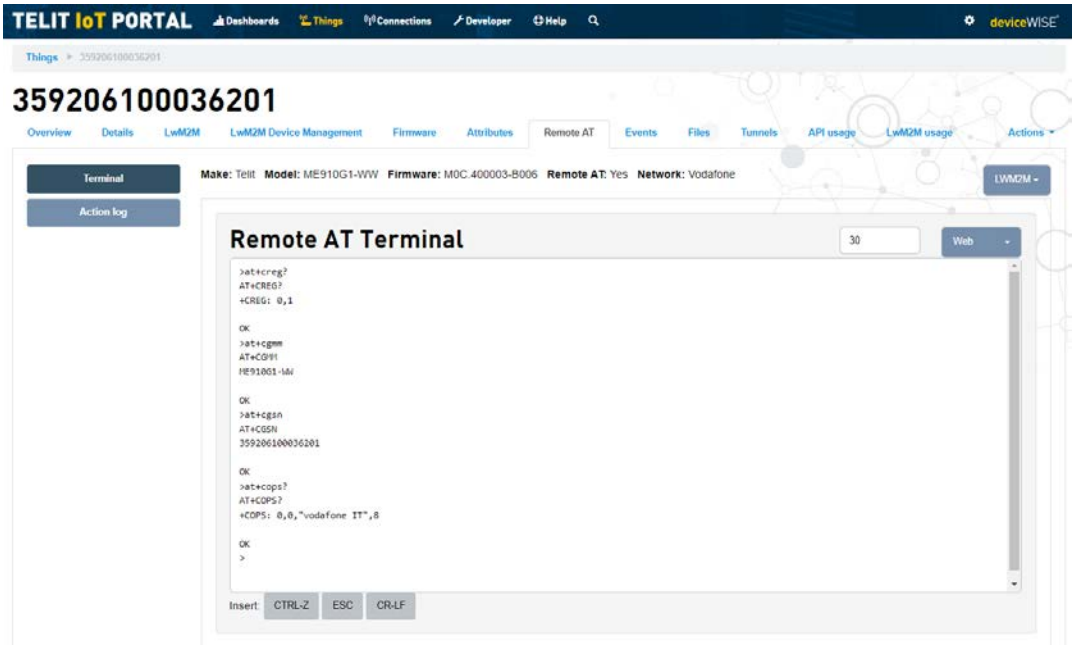



Figure 38:"Remote AT" feature on OneEdge IoT portal

# Telit IDs (object ID 33207)

This object provides additional parameters specific of the device.

Since this object is **managed internally** by the LwM2M client, the host application cannot write its resources.

Figure 39: Telit IDs object

# Connection Agent (Object ID 33210)

This object allows the User and the host application to manage and configure the parameters of the Connection Agent tool.

For more info related to the connection Agent, refer to paragraph *Connection agentand Connection Agent User Guide [11].*

| Connection Agent | /33210 | | | | | | |
|---|---|---|---|---|---|---|---|
| | /33210 | Observe | Read | Attributes | | | |
| Instance 0 | /33210/0 | Observe | Read | Write | Attributes | Delete | |
| CID | /33210/0/0 | Observe | Read | Write | Attributes | | 1 |
| APN Name | /33210/0/1 | Observe | Read | Write | Attributes | | |
| PDP type | /33210/0/2 | Observe | Read | Write | Attributes | | 2 |
| Log level | /33210/0/3 | Observe | Read | Write | Attributes | | 0 |
| URC level | /33210/0/4 | Observe | Read | Write | Attributes | | 2 |
| Main timer | /33210/0/5 | Observe | Read | Write | Attributes | | 10 |
| Signal Level Timer | /33210/0/6 | Observe | Read | Write | Attributes | | 20 |
| Net Registration Timer | /33210/0/7 | Observe | Read | Write | Attributes | | 50 |
| Data Service Timer | /33210/0/8 | Observe | Read | Write | Attributes | | 60 |
| Interval Timer Shutdown | /33210/0/9 | Observe | Read | Write | Attributes | | 100 |
| RSSI Threshold | /33210/0/10 | Observe | Read | Write | Attributes | | 13 |
| RSRP Threshold | /33210/0/11 | Observe | Read | Write | Attributes | | 30 |
| BER Threshold | /33210/0/12 | Observe | Read | Write | Attributes | | 3 |
| RSRQ Threshold | /33210/0/13 | Observe | Read | Write | Attributes | | 19 |
| Policy | /33210/0/14 | Observe | Read | Write | Attributes | | 0 |
| PDP Retry Mechanism | /33210/0/15 | Observe | Read | Write | Attributes | | 0 |
| Roaming Allowed | /33210/0/16 | Observe | Read | Write | Attributes | | 1 |
| Net Statistics | /33210/0/17 | Observe | Read | Write | Attributes | | 1 |
| Autostart | /33210/0/18 | Observe | Read | Write | Attributes | | 0 |
| Network IoT_technology | /33210/0/19 | Observe | Read | Write | Attributes | | 0 |
| Agent Release | /33210/0/20 | Observe | Read | Attributes | | | CA.10.0-09 |
| Username (APN) | /33210/0/21 | Observe | Read | Write | Attributes | | |
| Password (APN) | /33210/0/22 | Observe | Read | Write | Attributes | | |
| LWM2M Enable | /33210/0/23 | Observe | Read | Write | Attributes | | 0 |
| Agent Statistics Total Attempts | /33210/0/100 | Observe | Read | Attributes | | | 2 |
| Agent Statistics RSSI Fail count | /33210/0/101 | Observe | Read | Attributes | | | 0 |
| Agent Statistics Net Registration fail count | /33210/0/102 | Observe | Read | Attributes | | | 0 |
| Agent Statistics PDP Success count | /33210/0/103 | Observe | Read | Attributes | | | 0 |
| Agent Statistics PDP Fail count | /33210/0/104 | Observe | Read | Attributes | | | 0 |
| Agent Statistics Generic error count | /33210/0/105 | Observe | Read | Attributes | | | 0 |
| Agent restart | /33210/0/200 | Exec | | | | | |

Figure 40: Example of Connection Agent object

# Telit Services Management (Object ID 33211)

This object allows the User and the host application to configure some additional services and parameters.

Service Enable resource

The available services are mapped into the multiple instance resource **Service Enable** (33211/0/0):

Multiple instance resource 0: **GNSS** service (value 0 to disable and 1 to enable);

Multiple instance resource 1: **URC** service (value 0 to disable REGISTERING/REGISTERED URC messages and 1 to enable);

Multiple instance resource 2: **PDP service management in charge of Connection Agent** service (value 0 to disable and 1 to enable);

**Service Status Resource**

The meaning is the same of **Service Enable** resource, but the server can only read the current values of the flags.

### External Observations Default Period Resource

Related to the observations and according to OMA objects supported as default, there are different behaviors across two different managements of the resources on the LwM2M client.

The 2 resource categories are listed below and described deeply in paragraph 8.3.

Resources that are managed internally by the LwM2M client, for example the **Radio Signal Strength** resource in Connectivity management object (4/0/2) is filled by the LwM2M client polling the RSSI level from the RF chip of the modem periodically;

Resources that can be written by the host application,

The **External Observations Default Period** resource is applicable only to the first resource category (managed internally by the client) and defines a timer (in seconds) for which, when expired, the client is forced to send a notify to the server for the resources under observation.

When the timer expires, in case the resource value has changed (since the last notify) then the client sends a notify to the server, if the value is the same nothing changes.

The default value is **1800 seconds**.

The typical example is an application that is monitoring the RSSI level of a device in the field: the LwM2M client gets RSSI data from the RF chip every 1-2 seconds and writes the values on the 4/0/2 resource.

In case of observation on /4/0/2 resource without attributes, according to LwM2M specification, the LwM2M client would send a notify every 1-2 seconds for every minimal variation.

In order to avoid this unexpected annoying sending behavior, Telit provides the **External Observations Default Period** parameter.

Considering the default value equal to 1800 seconds, the LwM2M client still continues to get the values from the RF chip and write them on the resource every 1-2 seconds, but it evaluates the resource value changes every 1800 seconds.

In this case the LwM2M client sends the notifies every 1800 seconds.

The **External Observations Default Period** value is valid for all the resources that are managed internally by the LwM2M client: refer to paragraph 8.3 for the complete list.

### Critical Observations Resource.

This resource is multiple instances and can contain up to 10 resource URIs.

The behavior is very similar to the “External Observations Default Period” resource but in this case the User can select up to 10 resources that must be evaluated as soon as possible for notifies.

Figure 41: Telit Services Management object with the resources explained above

## 9.3 Resources Managed Internally Vs Host Application

Related to the 2 object types in the previous paragraphs 8.1 and 8.2, default **LwM2M OMA** and **Telit proprietary** objects, they are always available as default on Telit modules and internally managed by the LwM2M client.

This means that the LwM2M client itself creates the needed instances and writes the values on the resources according to events on the LwM2M client and on the modem, so there is no need to write custom values to these objects from host application or server side.

Anyway, some exceptions are present, so the host application can modify the resource values in some cases: the exceptions are highlighted in the tables below.

For example, on Device object (ID 3), the LwM2M client writes some info like manufacturer name in the resource 0 at the startup and cannot be modified by the host application.

On the other hand, the value of the **Battery level** resource is just initialized to value **100** by the LwM2M client, but the host application can modify its value.

The tables below report a summary of the available actions for the resources of the object loaded as default by the Telit modem.

Table 19: Summary of the available actions for the resources of the OMA objects loaded as default by the Telit modem.

| Default LwM2M OMA objects | | | | |
|---|---|---|---|---|
| Object | Resources | Managed by LwM2M client or host application | Host application side | Server side |
| 1 | all | Managed by the LwM2M client | read/write NOT allowed | read/write allowed |
| 2 | all | Managed by the LwM2M client | read/write NOT allowed | read/write allowed |
| 3 | 6: Available Power Sources<br>7: Power Source Voltage<br>8: Power Source Current<br>9: Battery level<br>13: Current Time<br>14: UTC offset | Managed by the LwM2M client but filled by host application | Read/write allowed through AT#LWM2MSET and AT#LWM2MR commands | read/write allowed |

| Default LwM2M OMA objects<br>Object | Resources | Managed by LwM2M client or host application | Host application side | Server side |
|---|---|---|---|---|
| | 20: Battery Status | | | |
| | The remaining resources | Managed by the LwM2M client | write NOT allowed | read/write allowed |
| 4 | all | Managed by the LwM2M client | write NOT allowed | read/write allowed |
| 5 | all | Managed by the LwM2M client | read/write NOT allowed | read/write allowed |
| 6 | all | Managed by the LwM2M client | write NOT allowed | read allowed |
| 7 | 8: Collection Period | Managed by the LwM2M client but the host application can modify the value | Read/write allowed through AT#LWM2MW and AT#LWM2MR commands | read/write allowed |
| | the remaining resources | Managed by the LwM2M client | write NOT allowed | read allowed |
| 10 | All | Managed by the LwM2M client but the host application can modify the values | Read/write allowed through AT#LWM2MW and AT#LWM2MR commands | read/write allowed |
| 11 | All | Managed by the LwM2M client but the host application can modify the values | Read/write allowed through AT#LWM2MW and AT#LWM2MR commands | read/write allowed |
| 500 | All | Managed by the LwM2M client but the host application can modify the values | Read/write allowed through AT#LWM2MW and AT#LWM2MR commands | read/write allowed |

**Table 20: Summary of the available actions for the resources of the Telit proprietary objects loaded as default by the Telit modem.**

| Telit proprietary objects | | | | |
|---|---|---|---|---|
| Object | Resources | Managed by LwM2M client or host application | Host application side | Server side |
| 33205 | All | Managed by host application | Read/write allowed through AT#LWM2MW and AT#LWM2MR commands | read/write allowed |
| 33206 | All | Managed by the LwM2M client | write NOT allowed | read allowed |
| 33207 | All | Managed by the LwM2M client | Read allowed, through. AT#LWM2MR command | Read allowed |
| 33210 | All | Managed by the LwM2M client but the host application can modify the values | Read/write allowed through AT#LWM2MW and AT#LWM2MR commands | read/write allowed |
| 33211 | All | Managed by the LwM2M client but the host application can modify the values | Read/write allowed through AT#LWM2MW and AT#LWM2MR commands | read/write allowed |

# 10 LwM2M CUSTOM OBJECTS

## 10.1 OMA Objects and Custom Objects

During a LwM2M data model design, one of the first steps is to map the high-level application requirements related to sensor values and application parameters into LwM2M objects.

The first question to be answered is if there is a standard or **IPSO OMA object** that already fits the requirements: in this case, according to OMA best practice, existing OMA objects could be re-used or extended.

**Reference link on wiki OMA web site:**
https://wiki.openmobilealliance.org/display/TOOL/LwM2M+Best+Practice

For example, a metering application that manages temperature and humidity sensors could use existing IPSO 3303 and 3304 objects.

The updated list of OMA objects can be found at OMA web site

http://www.openmobilealliance.org/wp/omna/lwm2m/lwm2mregistry.html

Anyway, if none of the IPSO objects is suitable for the application requirements, a custom object can be developed.

> **Note:** Designing a data model over a custom object is usually preferred rather than using IPSO objects in order to optimize the data consumption. Refer to next paragraph How to Create a LwM2M Custom Object.

## 10.2 How to Create a LwM2M Custom Object

## OMA LwM2M Object Editor

To create a custom object, the online OMA editor available at the following link can be used:

https://devtoolkit.openmobilealliance.org/OEditor/

The User has to open the **New 1.0** link to create an object of LwM2M version 1.0.

**Figure 42: online OMA editor with "New 1.0" link in detail**

The following web form box shows 5 editable sections, with the details about the most important fields:

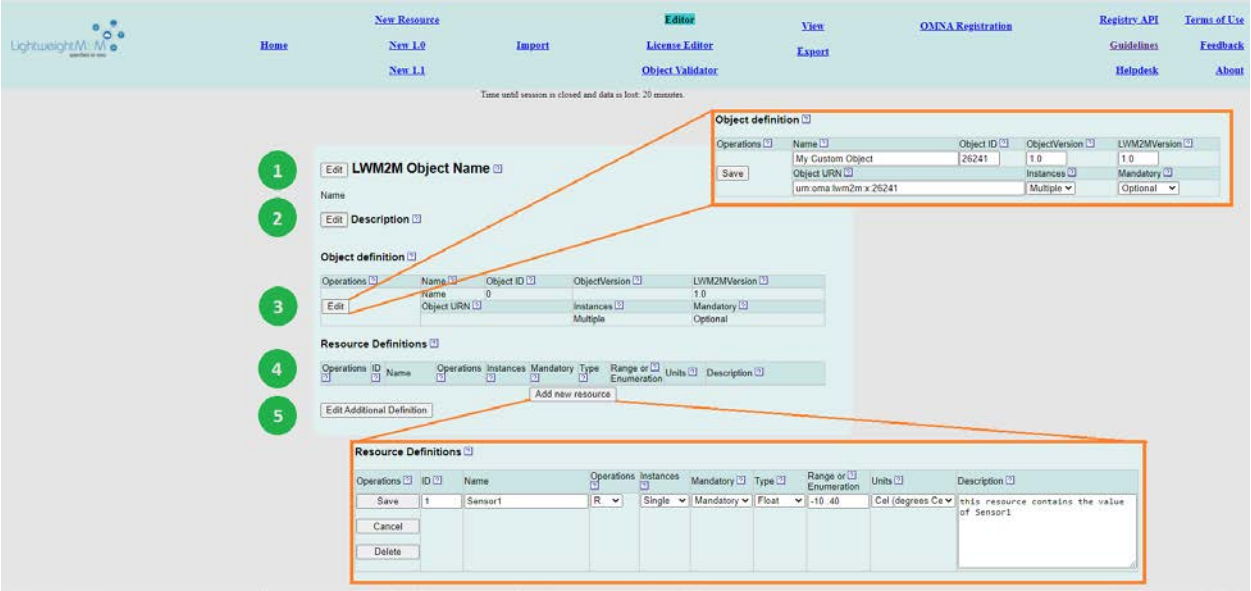



Figure 43: LwM2M OMA editor first view with most interesting sections in detail

All the fields in the 5 sections must be filled according to the editor guide available at the following links:

- https://wiki.openmobilealliance.org/display/TOOL/eLwM2M+Editor
- https://github.com/OpenMobileAlliance/lwm2m-registry/wiki/Guidelines.

**Note:** The question mark icon available near every editable field gives guidelines and filling rules.

Section LwM2M Object Name (#1):

It's the name of the XML file that the online OMA editor will generate.

Section **Description** (#2):

It's an optional description of the object.

Section Object Definition (#3):

List of the fields, which must be filled after pressing the "Edit" button:

Table 21: list of available fields

| Field | Description |
|---|---|
| Name | the name of the object (limited to 49 characters in OneEdge) |
| Object ID | a specific number in the **range 26241 – 32768** for each object |
| ObjectVersion | this optional field is useful to create more versions of the same object |
| LWM2MVersion | it must be **1.0** according to LwM2M 1.0 |
| Object URN | a string in the format "urn:oma:lwm2m:x:<ObjID>" |
| Instances | Single or Multiple |
| Mandatory | Mandatory or Optional (typical value for custom objects) |

**Warning:** The maximum length of the name of the object accepted on OneEdge LwM2M client is 49.

**Note:** The orange box in Figure 42 above is the secondary web form after pressing the Edit button: the fields are filled with some example values for a custom object.

Section Resource Definitions (#4):

The Figure 44 below shows the editable field for the resources, after pressing **Add new resource** button:

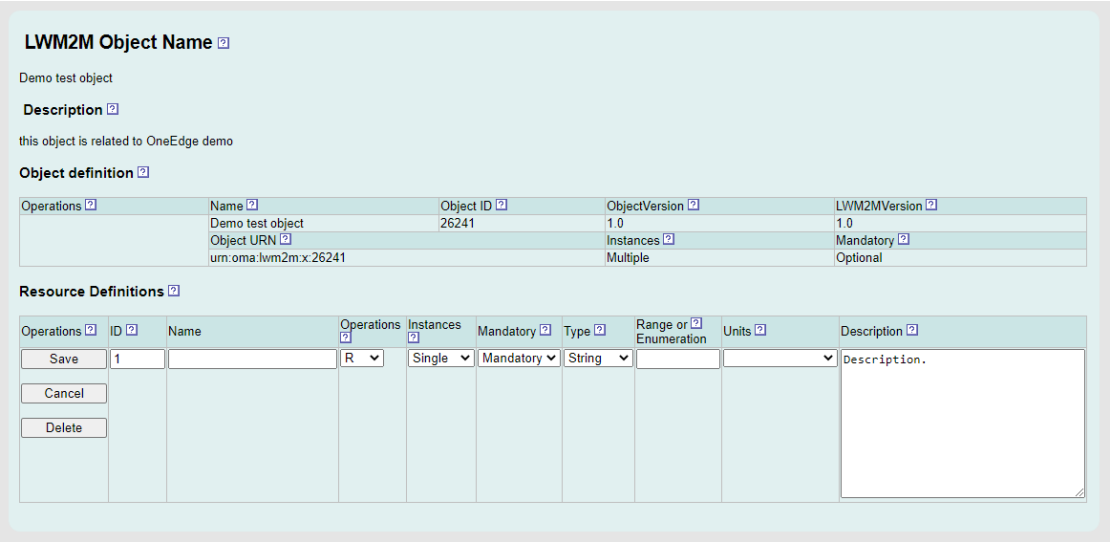



Figure 44: "Resource definition" fields of online OMA editor

List of the fields, which must be filled:

Table 22: List of fields to be filled

| Field | Description |
|---|---|
| ID | A unique id for each resource in the object |
| Name | The name of the object |
| Operations | The kind of operation that can be applied to the resource (refer to Table 12: Summary of actions related to resource operations. for details):<br>"R" for read only<br>"W" for write only<br>"RW" for both read and write<br>"E" for execute |
| Instance | Single or Multiple |
| Mandatory | Optional or Mandatory |
| Type | the type of the value to be managed: string, integer, float, Boolean, time, opaque, ObjLnk. Refer to Table 13:LwM2M resource types in OneEdge environment for details. |
| Range or Enumeration | Definition of valid values for the resource; anyway, Telit LwM2M client doesn't perform any check on the value of the resources |
| Units | unit of the value, i.e. "m/s" for speed |
| Description | A description of the meaning of the resource |

Section Additional Definition (#5):

Optional additional info to describe the object

When the User completes the custom object creation, he must export the last defined object structure as an XML file pressing the "**Export**" button on the menu on the top of the page.

The XML file must be loaded on both device file system and OneEdge IoT portal side according to section Manage LwM2M Custom Objects.

If the User has to modify the object definition, for example adding a new resource or changing operation flags, he must import the XML file through **Import** button, apply the changes and then save the new object definition. In this case a best practice is to increment the ObjectVersion number in the section #3.

Figure 45: "Import" and "Export" buttons in the top menu

# How to Select the Operations of the Resources

The operations allowed on the resources are always related to LwM2M server side actions.

A **ReadOnly** ("R") resource can only be read by the LwM2M server, which cannot modify its value remotely.

A typical example is a metering device application that reports a temperature value to the LwM2M server through a **Read Only** resource (refer to **IPSO object 3303**).

On the other hand, a resource related to a setting received by the **LwM2M** client is **Read** and **Write** (**RW**) because the LwM2M server can both modify and read the value of the resource remotely.

A typical example is a light control device application that receives the remote command from the server (refer to **IPSO object 3311**).

The **Write Only** operation is also suitable for remote setting, but the portal User cannot read the value of the resource.

An example is a password information that is sent from the server to the device: the host application is able to read it through the **AT#LWM2MGET** command, but the server cannot read the value (refer to standard object 9).

The "**Exec**" operation is suitable for actions to be forced on a device from the server.

For example, the **Reboot** resource of **Device** object (3/0/4) is used by the server to force a power cycle of the remote device.

# How to Select the Resource Type

According to table 6 in paragraph Resource, a resource can be defined by different types according to the meaning of the represented value:

The **integer** type represents a signed number without fractional part, for example counters.

The **float** type represents a signed number with fractional part, managed internally as floating point, for example a precise temperature, a voltage, etc.

The **boolean** type is suitable for **true-false** conditions.

The **time** type represents the number of seconds from 1970 according to Epoch Unix format, so it's managed as an unsigned integer.

The **string** type is suitable for resources that represent a text in **ASCII format**, for example manufacture info, application name, address, etc.

The **opaque** type is suitable for resources that represents a binary content in **hexadecimal format**, which is suitable for example for long binary flow, encrypted data, etc.

## Default Value for the Custom Object Resources

The table below reports the default values of the custom resources.

**Table 23: Default values of the resources in Telit LwM2M client**

| Resource type | Default value |
|---|---|
| Integer | 0 |
| Float | 0.0 |
| Boolean | 0 as False |
| Time | 0 seconds |
| String | Empty string |
| Opaque | 0x00 |
| ObjLink | 0:0 |

# Best Practice to Design LwM2M Data Models

The first basic rule to design a LwM2M data model is to map all the application data into one or more LwM2M custom objects.

For example, let's consider an environmental meter application to monitor temperature and humidity values and manage some parameters for the host application like frequency of the sensor sampling.

The first **"best-practice"** is to **split the application data** involved in the data model into **multiple custom objects** according to the direction of the data (from client to server or vice versa) and the usage of the data on host application and portal side:

**Sensor data**

- For example, in a metering device, the temperature value periodically reported.
- Data that change often and are sent often from the device to the portal.

**Tips:**

Mapped into a specific custom object.

The name of the object could be something like **Metering data**.

**Static info related to the host application.**

- Info that rarely change and usually there is no strong need to send them from the device to the server.
- For example, the name of the serial number of the metering device and the SW version of the host application.

Tips:

Mapped into another custom object.

The name of the object could be something like "**Metering info**"

**Note:** This information is important and must be available on LwM2M server side, but usually it's not needed that are sent to the server with the same frequency as the sensor metering data, so, to save data, this custom object could be sent from the device to the server only at specific events (i.e., LwM2M registration).

### Host application parameters

- For example, the sampling rate of the sensor values and thresholds
- These data must be resident on the LwM2M data model on the client and consumed by the host application.
- These data can be modified by the portal.

Tips:

Mapped into another custom object.

The name of the object could be something like "**Metering. settings/parameters**".

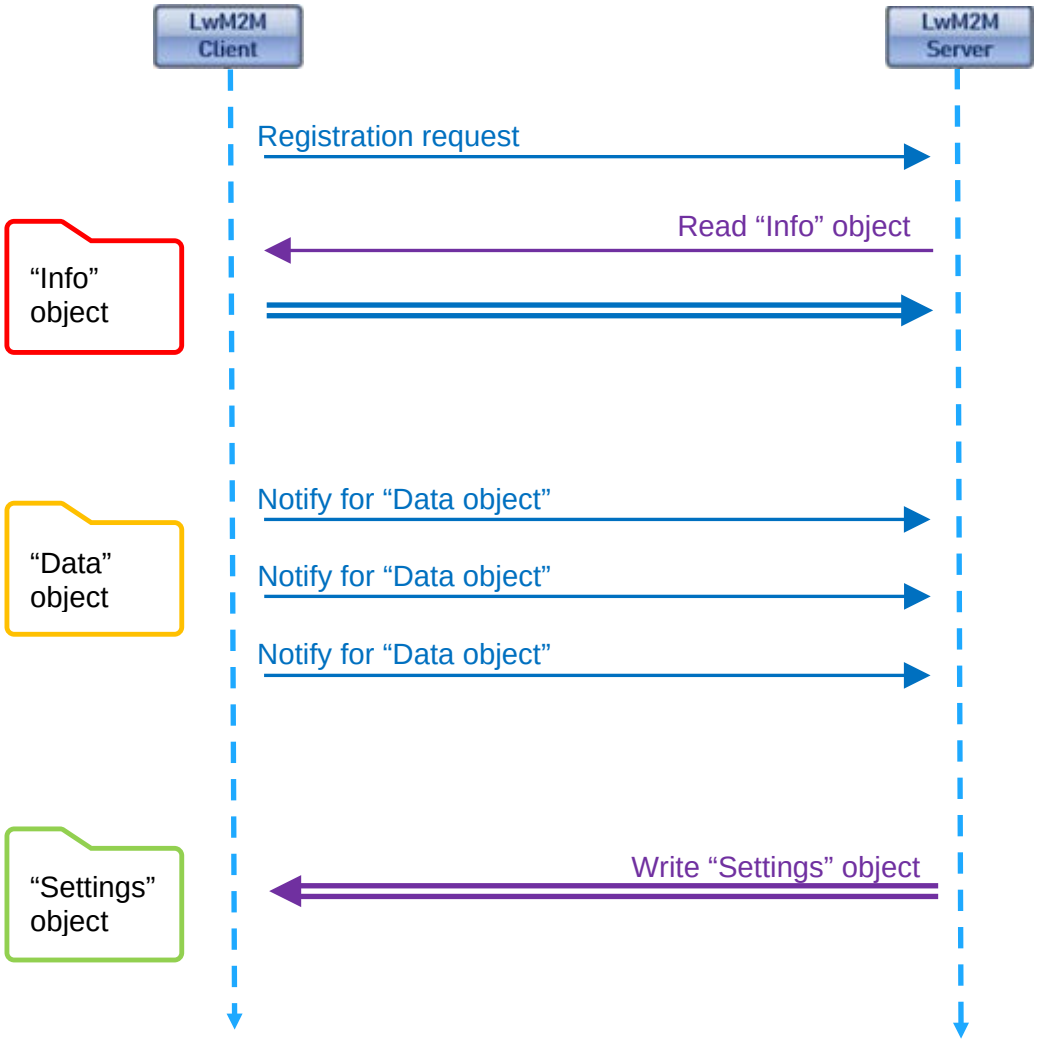



Figure 46: Typical actions related to different objects.

The second **best-practice** is to assign the Id of the resources leaving some free spare Ids for future developments.

For example, if a metering device manages now the 3 sensors temperature, humidity and pressure, a good assignment could be:

- resource Id #1 for temperature
- resource Id #11 for humidity
- resource Id #21 for pressure

In this way, if the device will be extended in the future with additional/optional sensors, the new temperature sensors will be addressed to Id numbers 2, 3, 4, etc., the new humidity sensors will be addressed to Id numbers 12, 13, 14, and so on.

In this way all the temperature values will be grouped between 1 and 10, the humidity ones between 11 and 20 and so on.

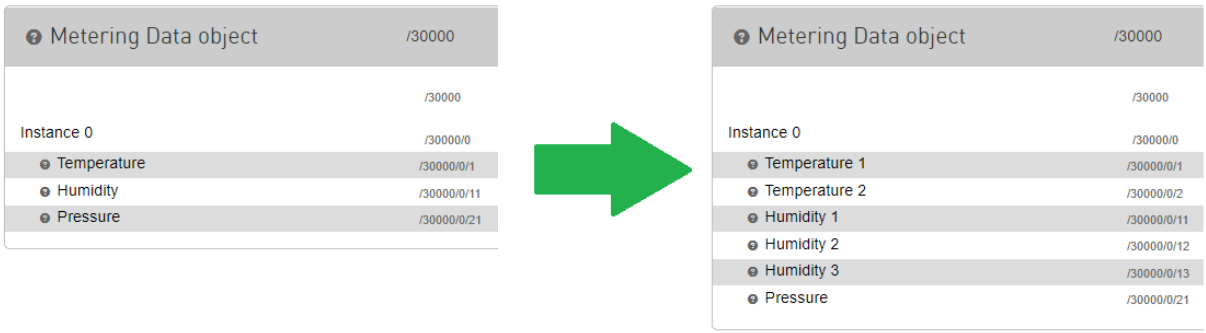



**Figure 47: Example of original Metering object and extended one.**

This second best practice allows the host applications to be extended in a in a modular and scalable way.

In case of doubts, the OneEdge technical support team (ts-oneedge@telit.com) is available to provide application feedbacks and data model review.

## 10.3 Manage LwM2M Custom Objects

If an application needs to manage an additional object, the User should be sure that the object definition (XML file) is present on both modem file system and OneEdge IoT portal.

XML files related to IPSO objects can be downloaded from OMA web site https://technical.openmobilealliance.org/OMNA/LwM2M/LwM2MRegistry.html.

> **Warning:** In OneEdge environment, some IPSO objects are already present on the Object registry section of the OneEdge IoT portal, but XML objects on OMA repository could be newer, so the User should always refer to the latest object definition from OMA repository.

Custom objects can be created through online editor available at link http://devtoolkit.openmobilealliance.org/OEditor/; Refer to chapter 7 for details.

**Warning:** The XML file is supposed to be well-formed when loaded in the module. The validation of the XML definition is up to the user. In case the XML format is not valid, the LwM2M client will not be able to correctly manage it. We strongly recommend to use the official OMA LwM2M object editor.

# Operations on Client Side

## AT#M2MWRITE

To load a LwM2M object XML file on modem file system the following steps must be performed:

Ensure that Hardware Flow control is enabled on both the Telit AT controller tool according to the following image and on modem side through AT&K3 command.

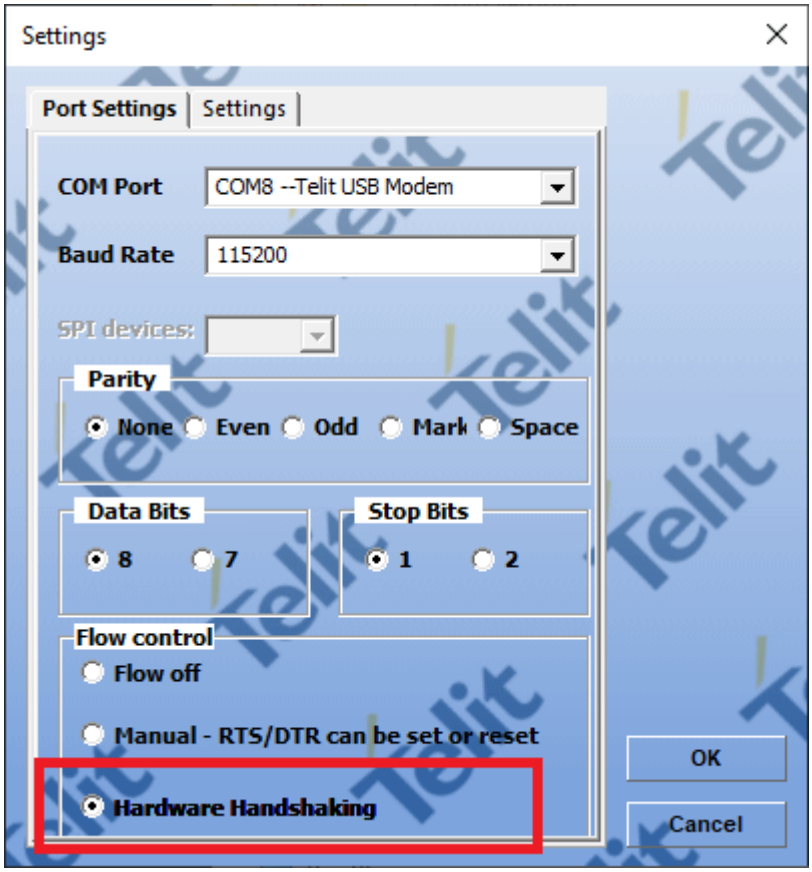



Figure 48: Flow control set as "Hardware Handshaking."

AT#M2MWRITE=/XML/object_xxxxx.xml,<file_size>, where xxxxx is the object ID (i.e. 3303 for Temperature IPSO object in Figure 48 below).

After the prompt >>>, the User or host application has to send the content of the XML file to the serial port.

On TATC, this can be done through the **Transfer Data** icon selecting the <xml_file_name>.xml from the PC file system.

The tool sends the file content to the serial port and **OK** is returned.

The LwM2M client consider the new object only at next power up of the modem, so AT#REBOOT command can be issues.

the User or host application can now start the LwM2M client through AT#LWM2MENA=1 command.

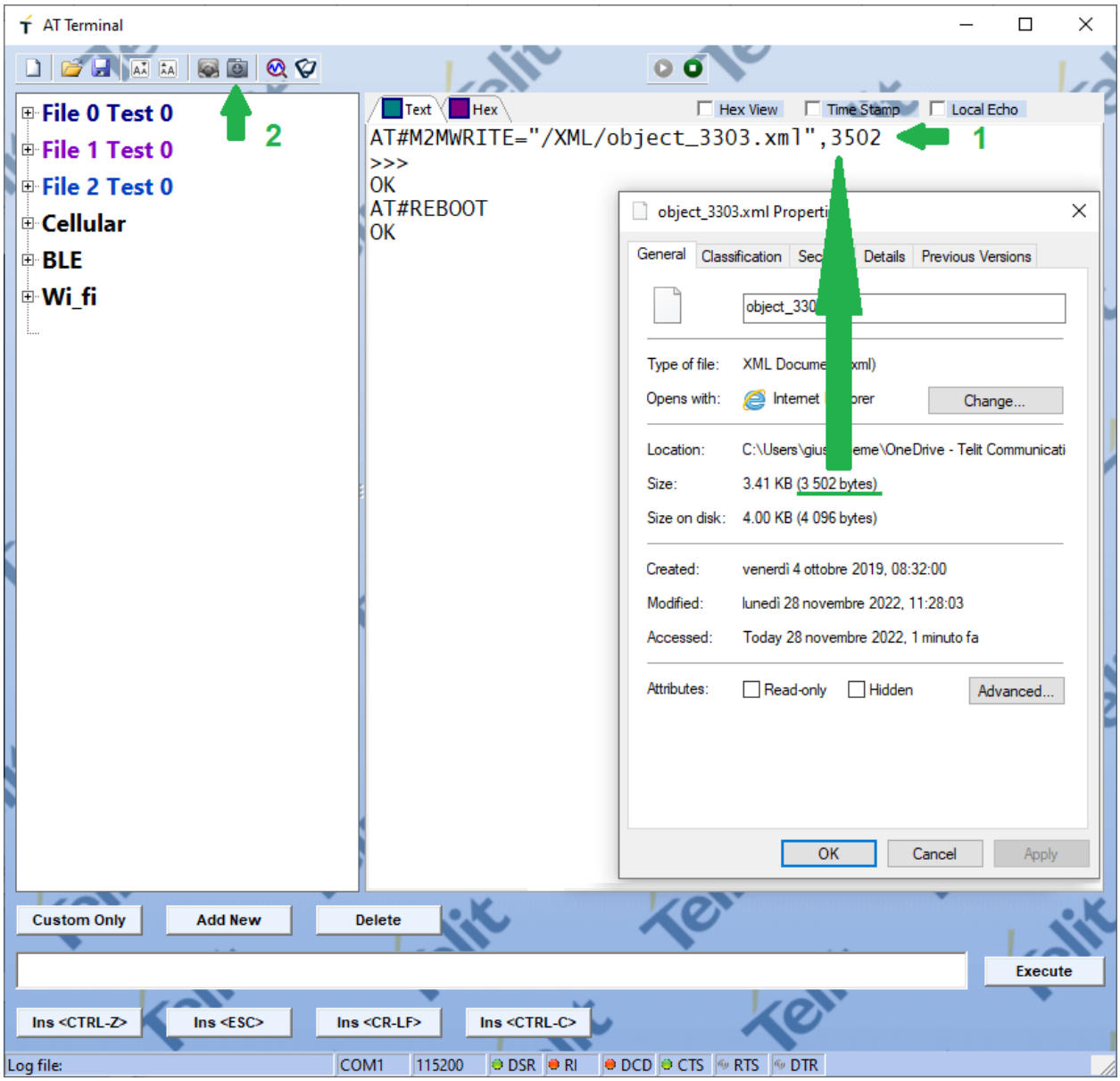



Figure 49: Steps to load an XML file on TATC

The new object is now installed in the module and the new instances can be created.

## AT#M2MLIST

The AT#M2MLIST can be used to list the content of a specific path of the File System.

In particular, the path **/XML** contains the XML files related to LwM2M custom objects.

Example related to the 3303 object loaded on the previous paragraph:

```
AT#M2MLIST="/XML"
#M2MLIST: <.>
#M2MLIST: <..>
#M2MLIST: "connagent.xml",15664
#M2MLIST: "empty",1
#M2MLIST: "object_3303.xml",3502
```

```
#M2MLIST: free bytes: 2713600
OK
```

## AT#M2MDEL

To delete a custom object, the XML object file has to be removed from the modem file system.

Below is the example to remove the Temperature object:

```
AT#M2MDEL=/XML/object_3303.xml
```

At next power up, the LwM2M client will not report the Temperature object to the server.

## AT#LWM2MNEWINST

To create an instance on the LwM2M client side, the `AT#LWM2MNEWINST` command has to be issued; the syntax is the following:

```
AT#LWM2MNEWINST=agentInst,objId,objInstId
```

Below is an example to create instances 0 and 1 of Temperature object:

```
AT#LWM2MNEWINST=0,3303,0
OK
AT#LWM2MNEWINST=0,3303,1
OK
```

The OneEdge IoT portal shows the new instances on object browser section (refresh of the web page could be needed).

If an instance is already present, an error is returned.

> **Note: OneEdge:** The maximum number of instances that can be created on a custom object is 100.

A host application could check if an instance is present or not through the **AT#LWM2MLIST** and **AT#LWM2MEXIST** commands:

Example related to the two instances of Temperature object created in the previous example:

```
AT#LWM2MLIST=0
#LWM2MLIST:
</1/0>,</2/1>,</2/2>,</2/3>,</3/0>,</4/0>,</5/0>,</6/0>,</10/0
>,</7/0>,</11/1>,</11/2>,</11/3>,</11/4>,</11/5>,</11/6>,</500
/0>,</500/6>,</500/7>,</33205/0>,</33206/0>,</33207/0>,</33211
/0>,</33210>,</3303/0>,</3303/1>
OK
AT#LWM2MEXIST=0,3303
OK
AT#LWM2MEXIST=0,3303,0
OK
AT#LWM2MEXIST=0,3303,1
```

```
OK
AT#LWM2MEXIST=0,3303,0,5700
OK
AT#LWM2MEXIST=0,3303,2
ERROR (+CME ERROR: Invalid URI)
```

According to this example, the first AT command provides the entire list of objects and instances on the LwM2M client; the second one returns OK if the URI (object, instance and resource) is available or ERROR otherwise.

## AT#LWM2MDELINST

The object instances can be deleted through the AT#LWM2MDELINST command.

The syntax is the following:

```
AT#LWM2MDELINST=agentInst,objId[,objInstId]
```

If objInstId is not provided, the command deletes all the instances of the given object.

Note that the instance delection functionality is used rarely because an host application usually creates the instances at startup once at the beginning of the device life through AT#LWM2MNEWINST command and then uses them at runtime for the rest of the life.

Anyway, there are 2 valid use cases that involves the instance deletion:

- An host application that creates and deletes instances according to the needs, but it's very rare use case
- An host application developer deletes the instances to restore the device to factory settings during developments and testing

> **Note:** it is possible to delete all the created instances restoring the LwM2M client to LwM2M default settings and restarting the client from Bootstrap procedure through the command:
> AT#LWM2MSTS=0,999,"coaps://bs.telit.io",1.

## AppZone APIs

The table below provides the matching between the AT commands of this chapter and the AppZone APIs for the same actions.

**Table 24: AppZone APIs**

| AT command | AppZone API |
|---|---|
| AT#LWM2MNEWINST | m2mb_lwm2m_newinst |

Refer to the online documentation at following link

https://s3.amazonaws.com/site_support/Telit/m2mb_docs/ME910C1-ML865C1/api-reference/30.01.xx0/m2mb__lwm2m_8h.html

https://s3.amazonaws.com/site_support/Telit/m2mb_docs/ME910C1-ML865C1/api-reference/30.01.xx0/m2mb__fs__posix_8h.html

AppZone sample applications available at following repository:
https://github.com/telit/IoT-AppZone-SampleApps/tree/master/LE910Cx_X/AppZoneSampleApps-AUX_UART/File_System

# Operations on OneEdge IoT Portal Side

## Load New Object on OneEdge IoT Portal

The procedure to load an XML file on OneEdge IoT portal is:

Click Developer tab > Object Registry.

Click New Object.

Paste the XML definition.

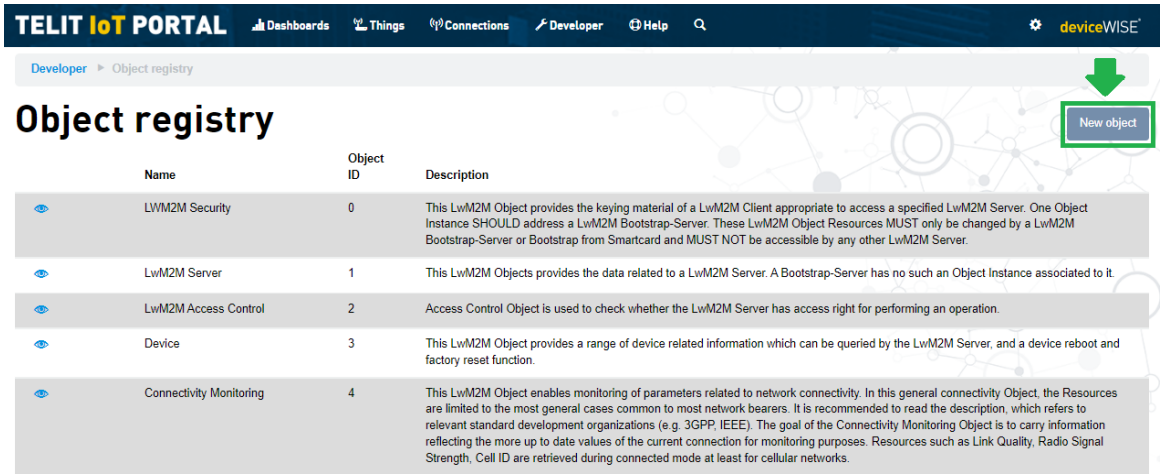



Figure 50: Object registry

**Warning:** Be aware that the Object ID 0 “LWM2M Security” cannot be modified.

## Delete an Object from OneEdge IoT Portal

On **Object Registry** section of **Developer** menu, the **Garbage bin** button has to be used to delete an object definition from the OneEdge IoT portal.

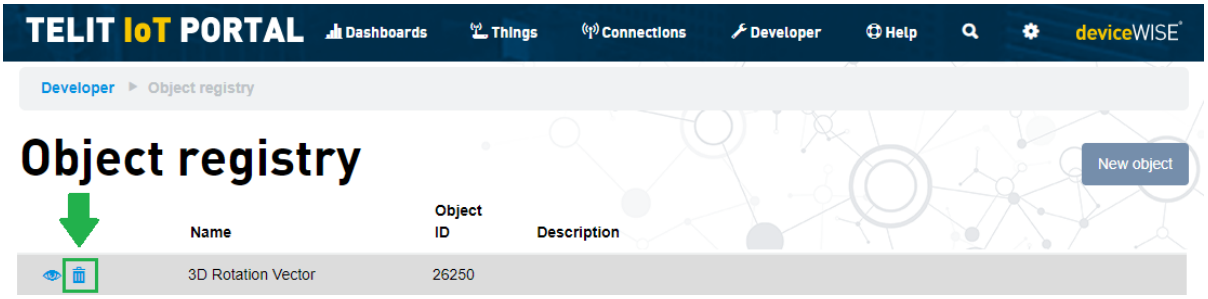



Figure 51: “Object Delete” button on object registry

**Warning:** Be aware that the the Bootstrap procedure restore the device at default so all the instances are deleted, but the XML files are kept.

## Create a New Instance from OneEdge IoT Portal Side

Even if instances are usually created on module side because the host application is responsible to map sensors into instances, according to LwM2M specification, an instance can be created also from server side through the "**New Instance**" button on object browser section of OneEdge IoT portal.

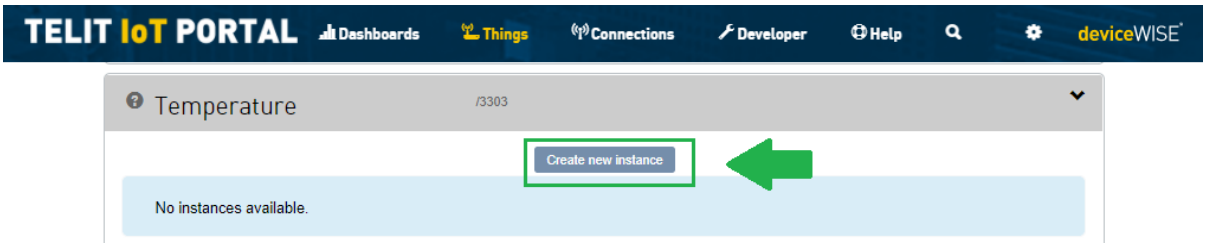



Figure 52: "Create new instance" button on object browser

## Delete an Instance from OneEdge IoT Portal Side

An instance can be deleted also from server side through the "**Delete**" button on object browser section of OneEdge IoT portal.

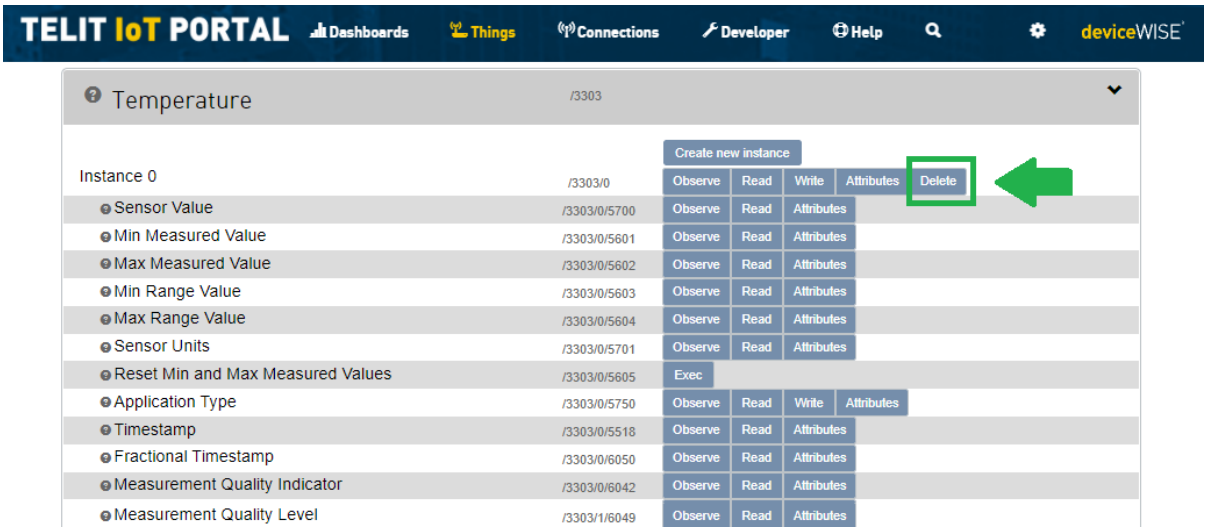



Figure 53: "Delete" button on object browser

## TR50 APIs

The table below provides the matching between the operations on OneEdge IoT portal side and corresponding TR50 commands.

Table 25: AppZone APIs

| AT command | AppZone API |
|---|---|
| Create an instance | lwm2m.device.oir.create |
| Delete an instance | lwm2m.device.oir.delete |

# 11 OneEdge IOT Portal and Appcenter

The OneEdge IoT portal, an IoT space composed by a LwM2M server integrated into a web server, hosts 3 independent instances, listed below:

**Development** portal (portal-dev.telit.com), where customers usually evaluate OneEdge functionalities, free of charge during the evaluation period;

**US Production** portal (portal-us.telit.com, portal.telit.com), located in United States, where customers manage OneEdge devices deployed in the field;

**DE Production** portal (portal-de.telit.com): similar to US production portal but located in Germany, for European customers;

Customer can manage its own devices on production portals only after a signed contract related to OneEdge service.

An organization is a private space on a portal where a customer manages its own devices, called as "things" and SIM cards, known as "**Connections**".

A portal can host several organizations related to several customers (usually 1:1) and a customer cannot see the other organizations.

The customer must indicate at least a user to be **administrator** of the organization when the org creation is requested.

First users are invited when Telit creates the organization and then any administrator can invite other people.

Users are assigned a **role** in the organization: roles are used to define access rights (view, modify, execute, and other actions.) to items.

The information and data of the things are organized and displays in **different web applications** on the OneEdge IoT portal and below are listed the main ones:

"**Device Management**" app, to show predefined info related to basic LwM2M objects, explained in paragraph Device Management Application

"**Firmware Management**" app, to manage the FOTA state machine in order to upgrade devices in the field easily, explained in paragraph Firmware Management Application.

"**Management portal**" app, to have full control of the LwM2M objects, explained in detail in paragraph Management Portal Application.

The main web link of the OneEdge IoT portal that provides access to all the web applications is http://appcenter.telit.com

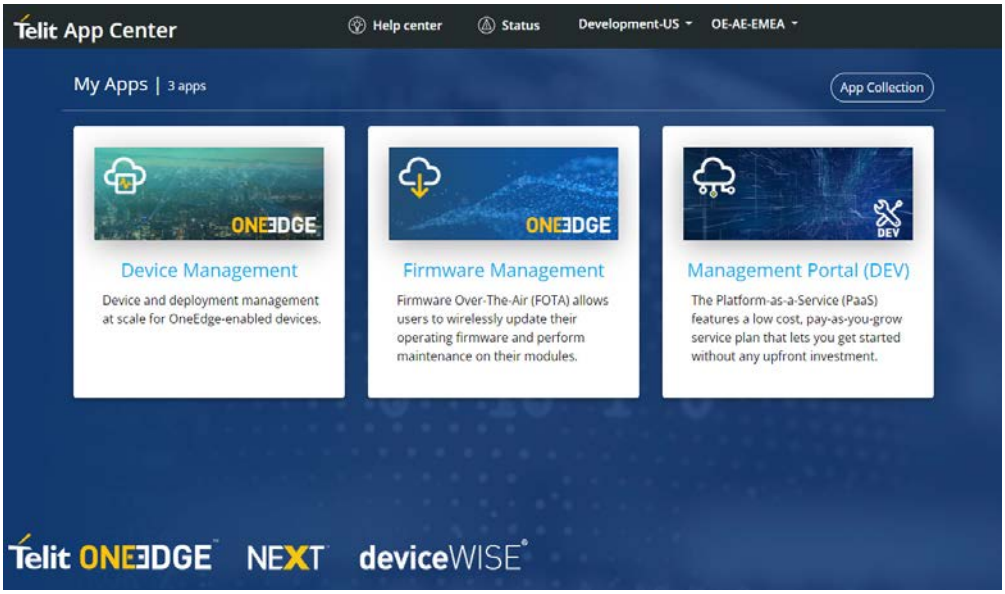



Figure 54: AppCenter home page with the 3 applications listed above.

**Reference:**

https://docs.devicewise.com/Content/Products/AppCenter/Tour-of-AppCenter-UI.htm

## 11.1 Management Portal Application

The Figure 54 below shows a logical representation of the elements involved in the **Management portal** app: both things as LwM2M devices and Connections can be managed and linked together into the same organization.

The things must be assigned to a specific **Thing definition** to share the same web **aspect** such that all the things have the same attributes and properties: more details are available at paragraph *Thing Definition*.

The same concept can be used for the configurations of the connections.

Related to the LwM2M configuration, the things must belong to a specific **Device profile**, which will be described in detail in paragraph *Device Profile*.

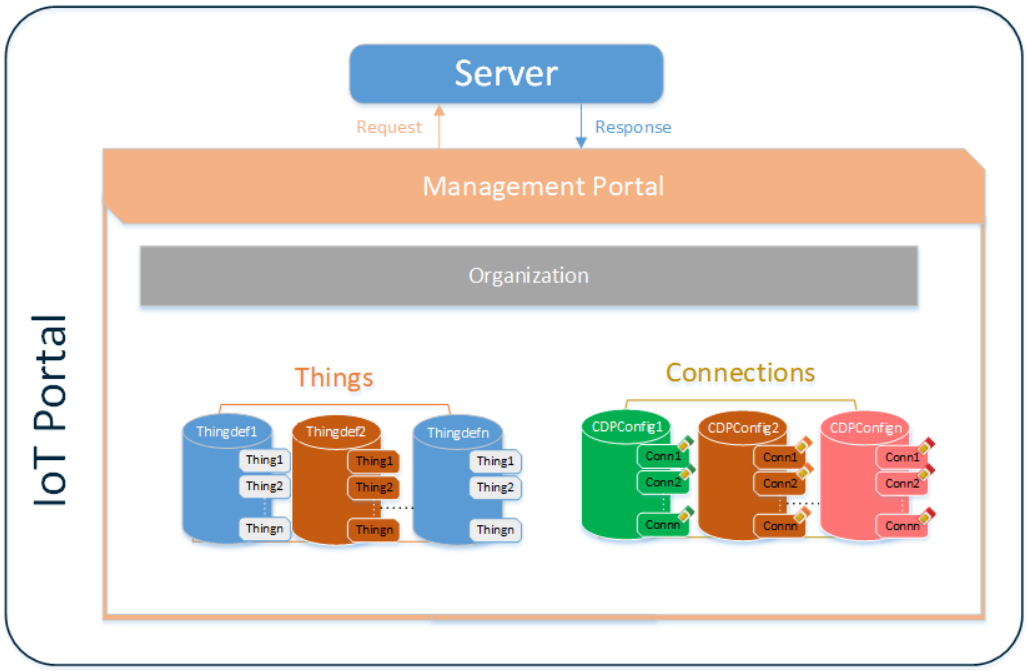



Figure 55: IoT portal architecture

More information can be found at link

https://docs.devicewise.com/Content/GettingStarted/Architecture-IoT-Portal.htm

The home page of the **Management Portal** app shows a summary related to the activities on the organization.

An example is in the figure 55 below.

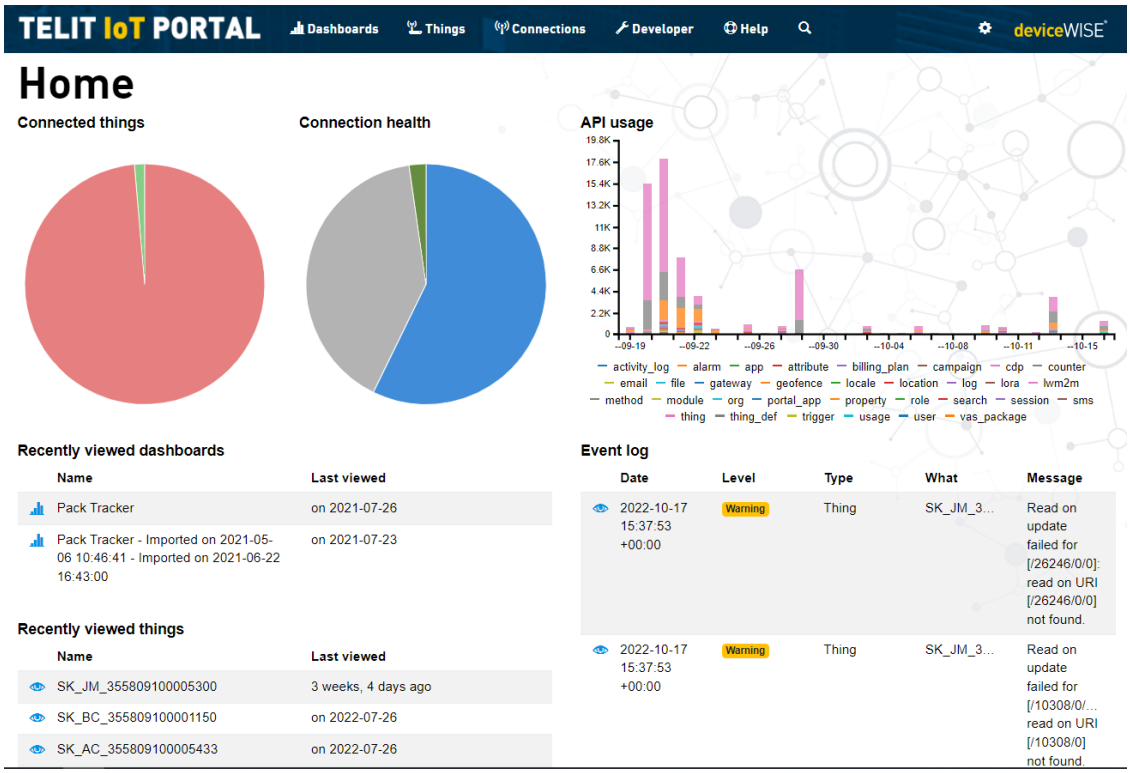



Figure 56: home page of a test organization

The **top** menu of the web page shows 5 main sections of the **Management Portal** application:

- Dashboards
- Things, the most important for OneEdge devices
- Connections
- Developer
- Help

**Reference:**

https://docs.devicewise.com/Content/Products/Management_Portal/Management-Portal-Users-Guide.htm

# Dashboards

The User can create its own web pages to show the things and connections information according to specific graphical structure.

The web elements (as calls as “dashboard widgets”) that can be used are pre-defined and listed in a sort of library.

**Reference:**
https://docs.devicewise.com/Content/Products/Management_Portal/Dashboards/Dashboards.htm

# Things

In the **Management Portal** there is a specific **Things** section dedicated to the things management.

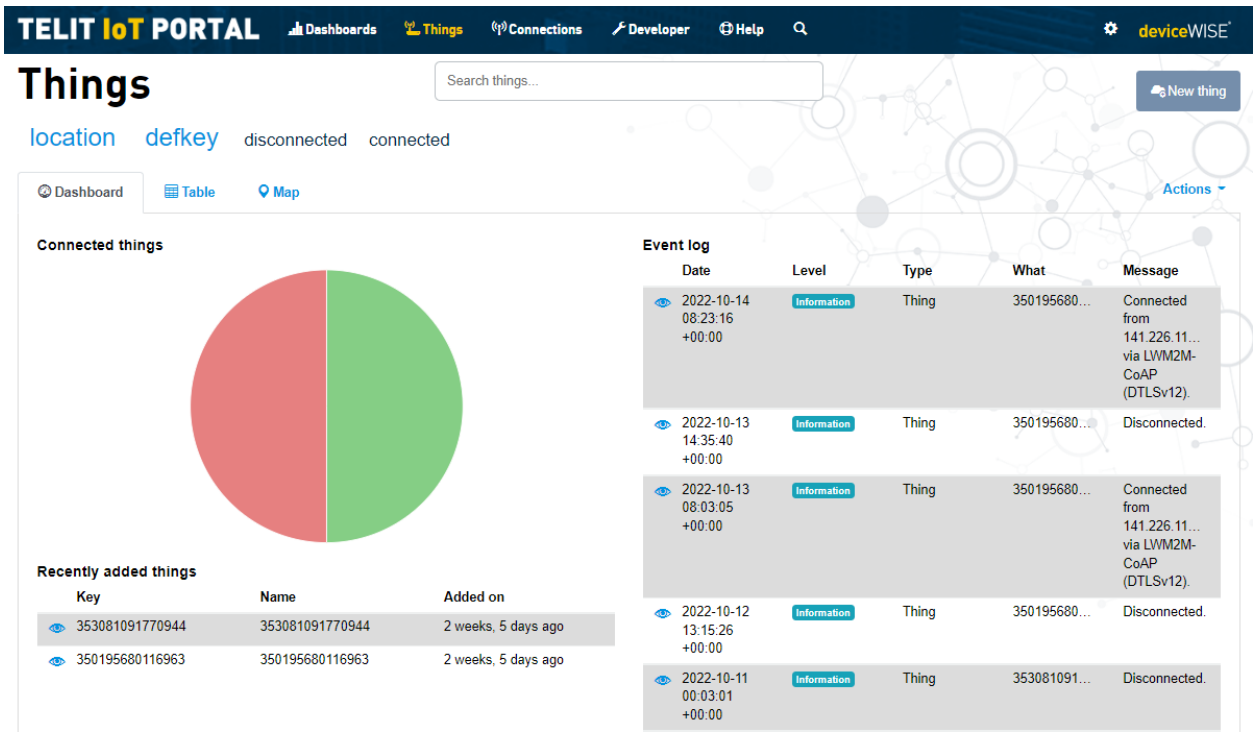



Figure 57: home page of the Thing section

This section contains three main tabs:

**Dashboard**: the Thing Dashboard shows a pie chart with a quick overview of connected things, a table of recently added things and event logs. It also displays the tags, if available, for quick filtering;

**Table**: provides a table with all the things of the customer organization. In this section also the details of each thing like properties, attributes, alarms, events, files and so on, can be included;

**Map**: it displays a map with the geographical deployment of the things population.

The main scope of the following sub-paragraphs is to guide the user through the **Table** and **Map** sections.

**Reference:**

https://docs.devicewise.com/Content/Products/Management_Portal/Things/Things.htm

## Table Tab

The default configuration of the table contains 4 columns, the things are sorted by name but the user can choose another order, for example by **Last Seen**, click the column name.

> **Warning:** Last Seen: the information reported in this field represents the time when a device publishes any data to the OneEdge IoT Portal, but not the last communication between LwM2M client and server.

Below some examples are reported:

if the device is attached to the cellular network using a SIM card managed in the Connections section of the same org, the “Last Seen” field reports the information related to the last connectivity activity, for example cellular signaling, also if there was no LwM2M data exchange.

If any attribute, property, or alarm is modified, the Last Seen field is updated, even if there is no LwM2M communication.

Pressing the multi-tool icon as title of the first column, the user can customize the table with the preferred columns.

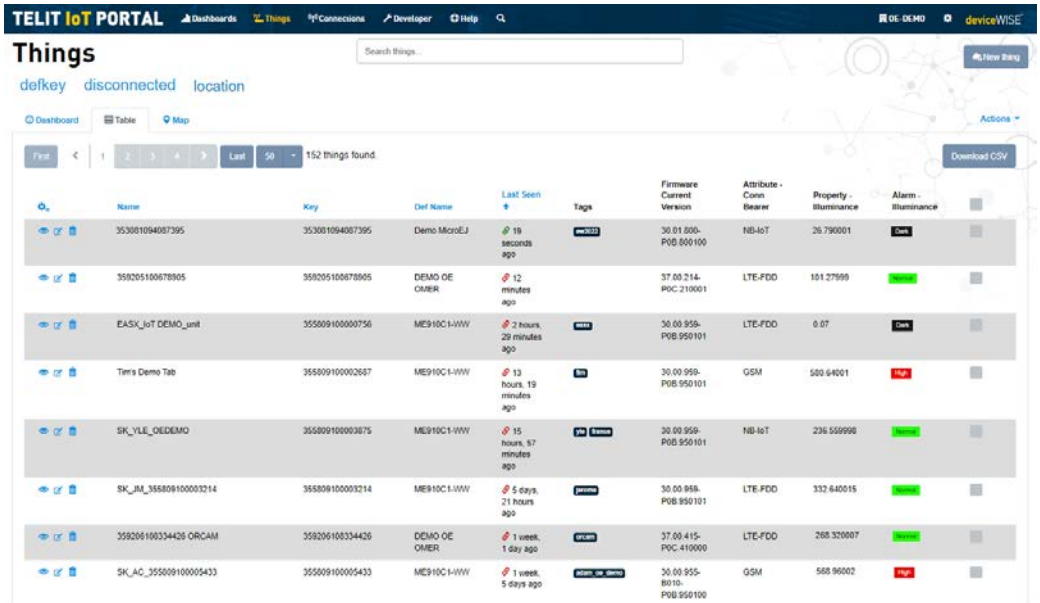



Figure 58:Example of Thing Table with some custom columns

**Reference:**

https://docs.devicewise.com/Content/Products/Management_Portal/Things/Customizing-the-Things-view.htm

## Creating a Things – Onboarding

A new thing can be created through the New Thing button on the right-top side of the Table page.

Anyway, this operation, as known as onboarding, is in charge to Telit Cinterion teams: since this process involves the security credentials as described in Bootstrap and Registration section, the portal Users shouldn’t create new things and modify the already onboarded ones.

**Note: OneEdge:** The thing onboarding done by Telit cinterion in OneEdge environment is know as “OneEdge zero-touch onboarding”.

The added value of this feature is that a OneEdge User that orders a number of OneEdge devices, will find the devices already loaded on the OneEdge portal as things and doesn’t need to manage the security credentials: the devices are

already configured and ready to communicate through the LwM2M protocol on OneEdge portal.

**Warning:** The "OneEdge zero-touch onboarding" feature is not available if the LwM2M client on Telit Cinterion modules is configured to register to an external LwM2M server (different that OneEdge server which is the default one).

One of the major downsides of this setup rather than OneEdge is that the developer should create a "security credentials set" for each device, store and manage the database with these security info and configure them on the external LwM2M server side.

**Reference commands:**

- AT#LWM2MSTS and AT#LWM2MINJKEYS.

## Viewing a Thing

The user can have a complete view of a specific thing information clicking on the **Eye** icon

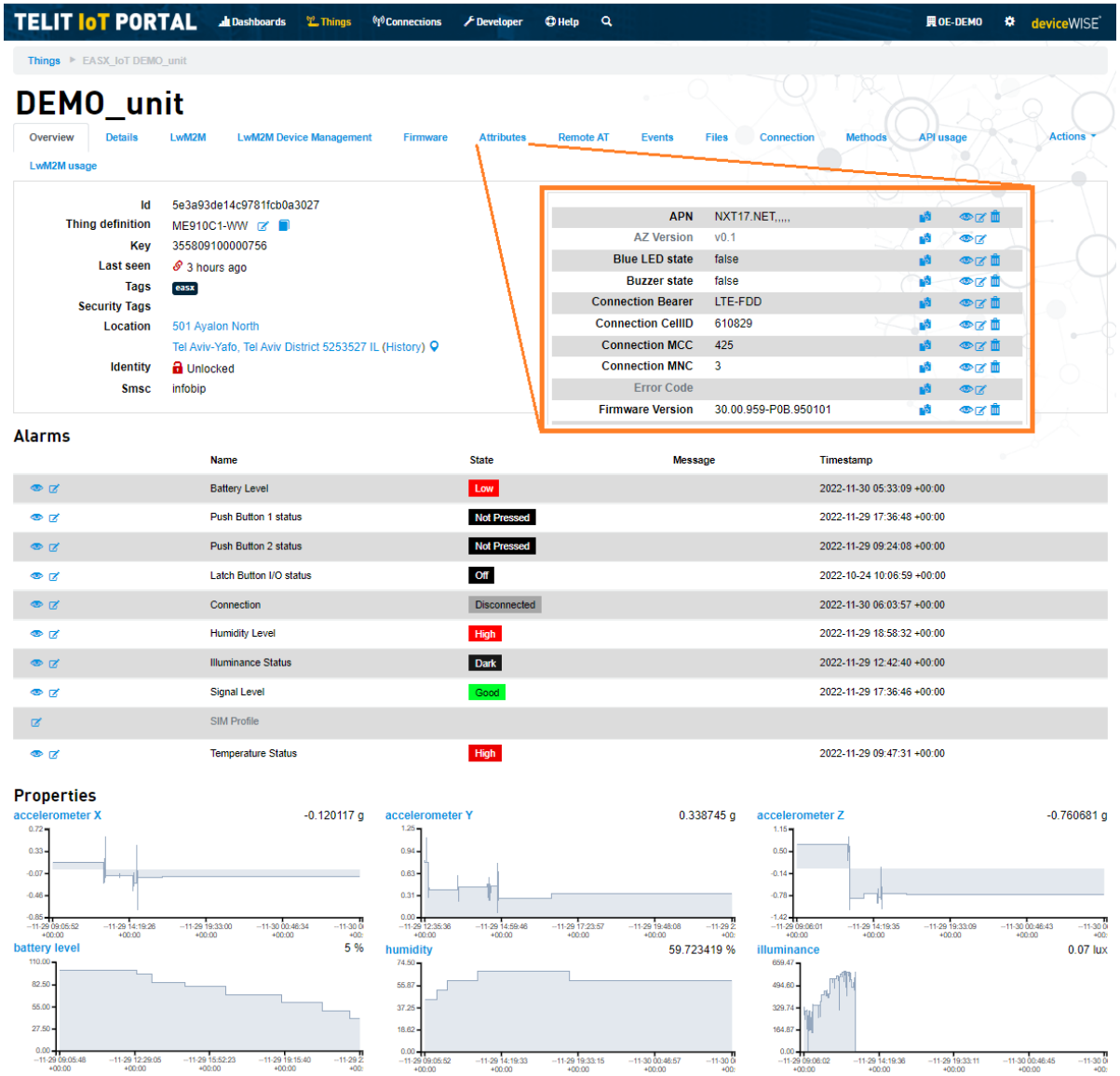



Figure 59:Details related to a specific thing.

The **aspect** of this page depends to the **Thing definition** assigned during the onboarding phase: refer to paragraph *Thing definition* for configuration.

The user can change it and create its preferred configuration showing the data of the thing in 3 different ways:

- **Attribute**: information container suitable for texts and showing the last value; the history is available inside each attribute
- **Alarm**: a graphical representation of information based on multiple states, suitable for levels, for example **On/Off, Open/Closed, Low/Medium/High**
- **Property**: a graphical representation of the information through several chart types, suitable for numerical data.

> **Note: OneEdge:** Attributes and properties can be linked to a LwM2M resource and automatically published through the OneEdge Binding feature on the Device profile and the OneEdge triggers.

On the other hand, the **alarms** must be published **only** through the **triggers**.

> **Note:** The alarms can be linked to the properties to show in the property chart when the alarm event happened.

Several tabs are present in the view of each thing and the most important ones for OneEdge devices are listed below:

- **LwM2M**: Information related to LwM2M parameters and events.
- **LwM2M Device Management**: Collection of most relevant LwM2M objects.
- **Remote AT**: A specific web interface to send AT commands to the device and get back the response through the LwM2M protocol.

This is a very appreciated feature for remote device management and troubleshooting.

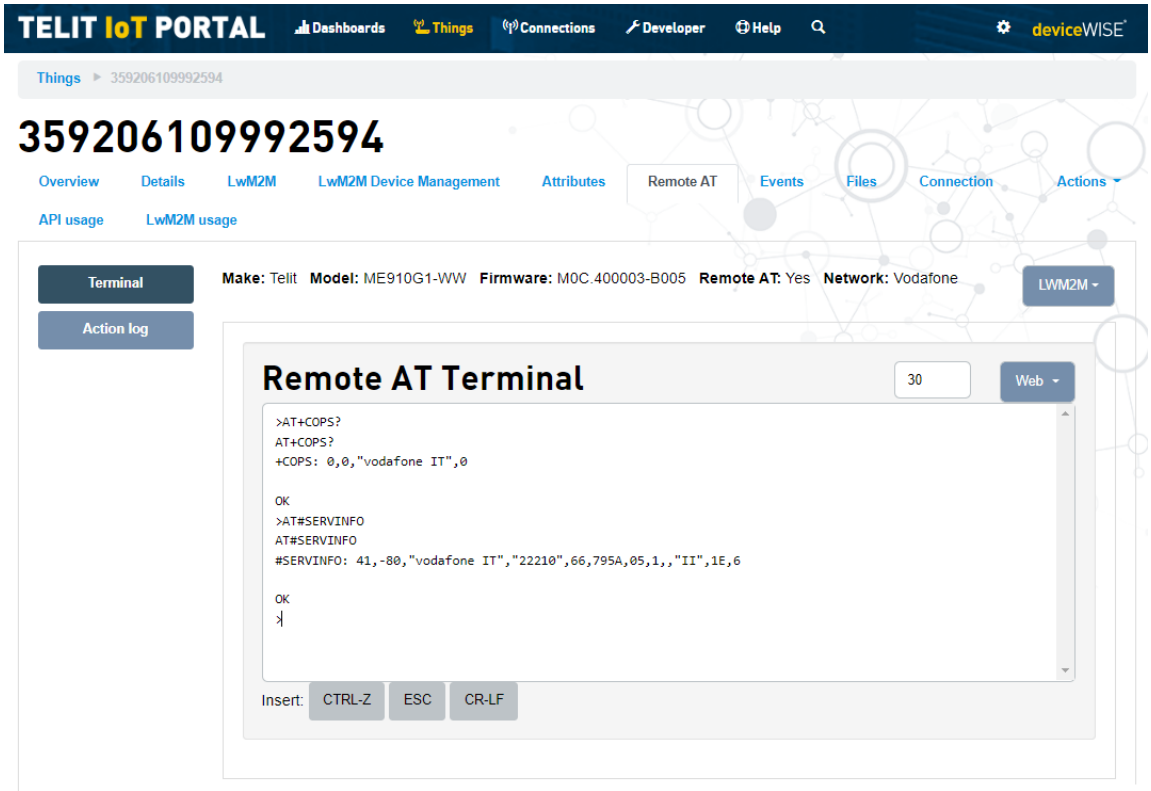



Figure 60: Remote AT feature

- **Events**: Relevant events, useful for debug and logging
- **Connection**: If the SIM connection is managed in the same org, this is the link to access to the connection of the specific thing.
- **LwM2M usage**: A chart to report the LwM2M messages send and received by the specific thing.

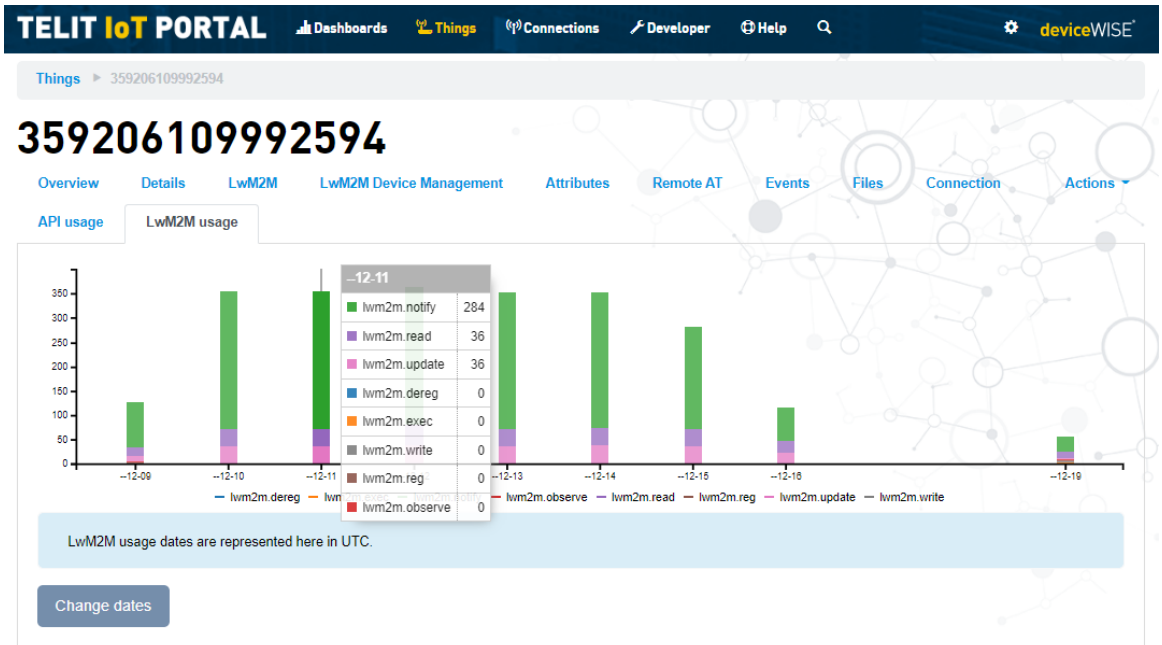



Figure 61: LwM2M usage chart, to monitor the LwM2M messages consumption

**Reference:**

https://docs.devicewise.com/Content/Products/Management_Portal/Things/Viewing-a-Thing.htm

## Map Tab

The Map tab shows the location of the things on the specific org.

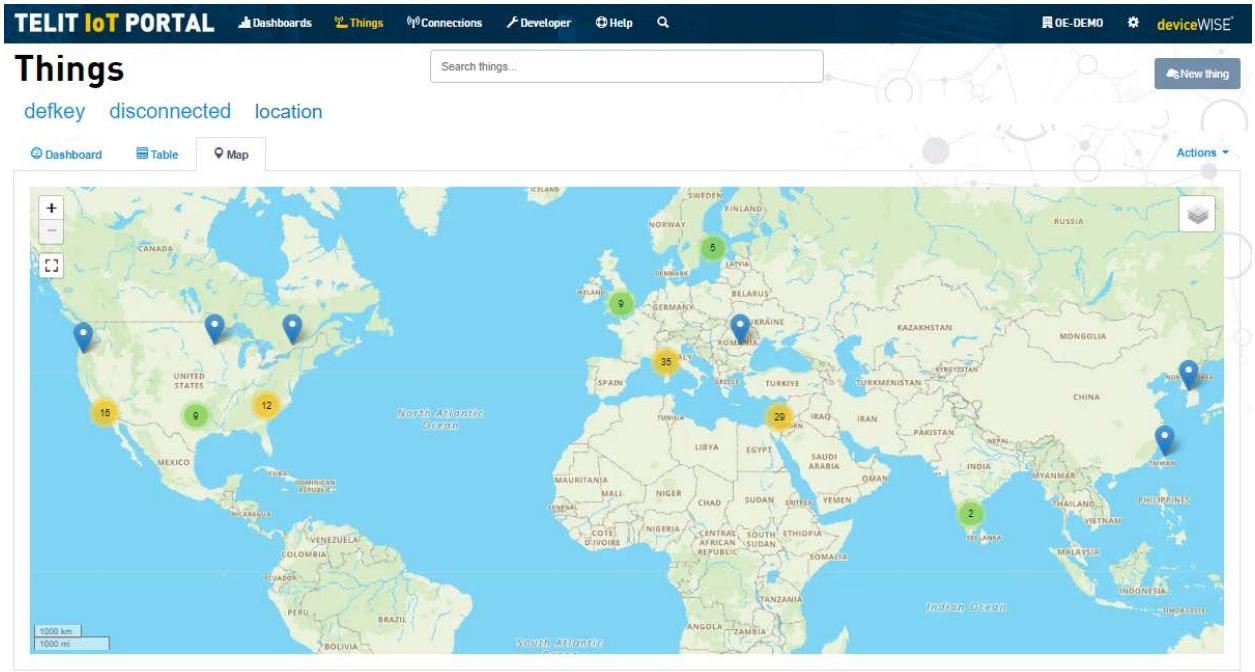



Figure 62: Example of map

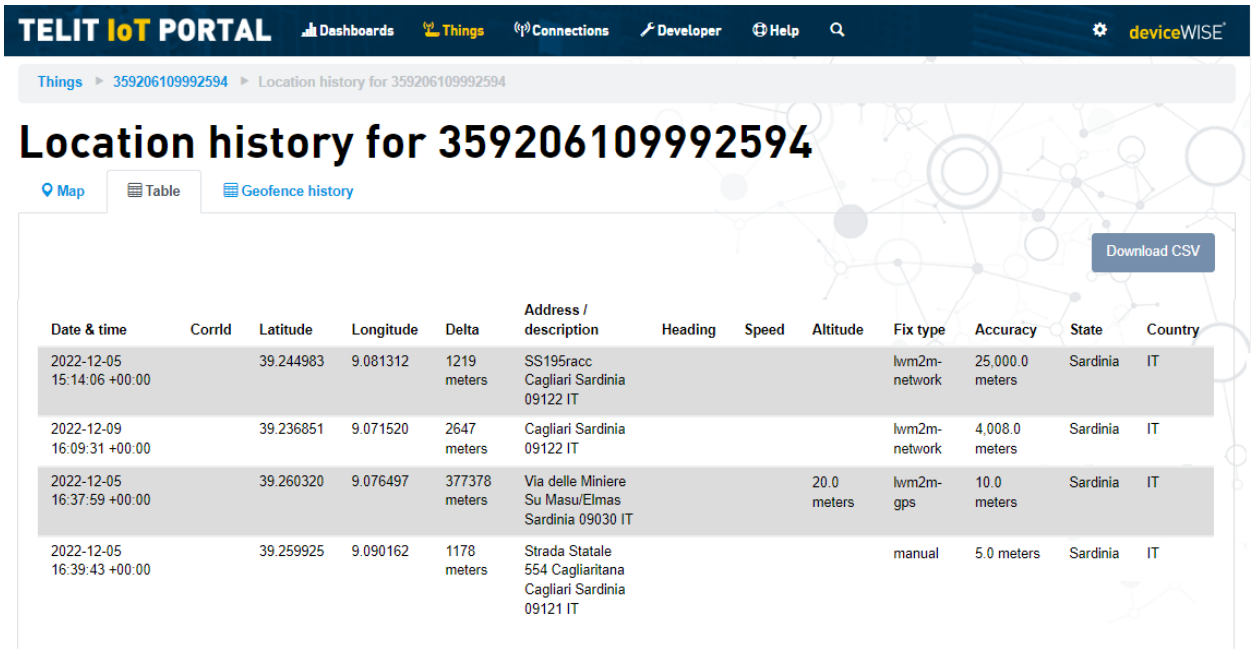
The location can be published on the OneEdge IoT portal in different ways, listed below:

Table 26: Different kinds of locations on a thing

| Location source | Description | Accuracy | Publish type | Fix type in location history | Constrain |
|---|---|---|---|---|---|
| Connection | the cellular network provides a rough position according to the serving cell | Rough (1-5 km) | Automatic | network | Connection on the same org of the thing |
| LwM2M object 4 | the cellular network provides a rough position according to the serving cell as reported on LwM2M object 4 to the OneEdge IoT portal | Rough (1-5 km) | Automatic | lwm2m-network | Read LwM2M object 4 on server side at least once |
| LwM2M object 6 | the latitude and longitude coming from the internal GNSS receiver of the device are reported on the object 6; usually this position is more | High (5-50 m) | Automatic | lwm2m-gps | Read LwM2M object 6 on server side at least once |

| Location source | Description | Accuracy | Publish type | Fix type in location history | Constrain |
|---|---|---|---|---|---|
| | accurate than the "serving cell" based | | | | |
| Trigger | the location can be published upon a specific event, for example coming from different formats (string, LwM2M object 3336, etc).<br>Refer to paragraph 10.1.4.3 | custom | Event driven on a trigger | manual | |
| User | the location can be published manually by the User on the OneEdge IoT portal.<br>Refer to location.publish API | custom | manual | manual | |

The Figure 63 below shows an example of location history with different kinds of locations.




| Date & time | CorrId | Latitude | Longitude | Delta | Address / description | Heading | Speed | Altitude | Fix type | Accuracy | State | Country |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2022-12-05 15:14:06 +00:00 | | 39.244983 | 9.081312 | 1219 meters | SS195racc Cagliari Sardinia 09122 IT | | | | lwm2m-network | 25,000.0 meters | Sardinia | IT |
| 2022-12-09 16:09:31 +00:00 | | 39.236851 | 9.071520 | 2647 meters | Cagliari Sardinia 09122 IT | | | | lwm2m-network | 4,008.0 meters | Sardinia | IT |
| 2022-12-05 16:37:59 +00:00 | | 39.260320 | 9.076497 | 377378 meters | Via delle Miniere Su Masu/Elmas Sardinia 09030 IT | | | 20.0 meters | lwm2m-gps | 10.0 meters | Sardinia | IT |
| 2022-12-05 16:39:43 +00:00 | | 39.259925 | 9.090162 | 1178 meters | Strada Statale 554 Cagliaritana Cagliari Sardinia 09121 IT | | | | manual | 5.0 meters | Sardinia | IT |

**Figure 63:Example of different fix types on a specific thing; to be noted the good accuracy of the location coming from GNSS receiver.**

The User can disable the location fix types that are not needed, in order to avoid multiple location sources on the map and the history: it can be configured on Thing definition as showed on Figure 63.

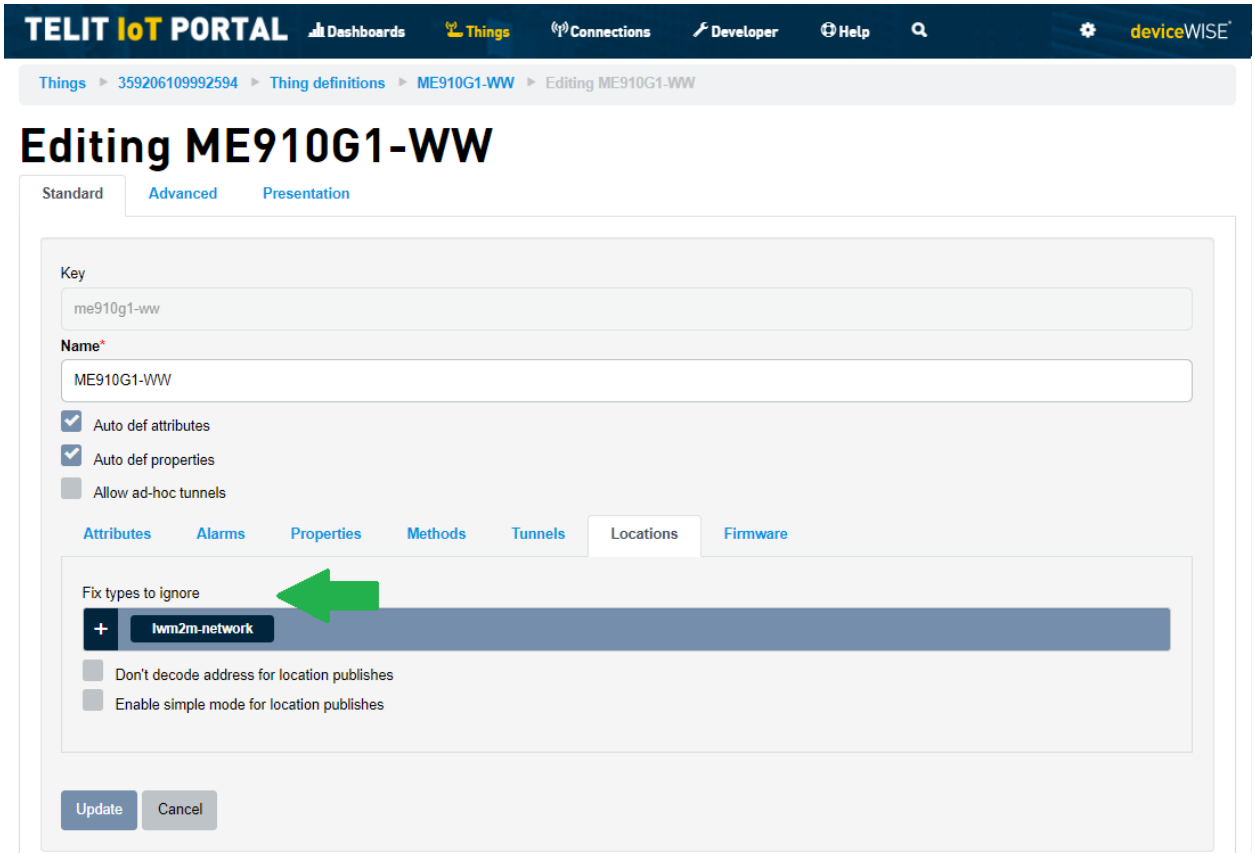



**Figure 64: Removing "lwm2m-network" fix type from Thing definition configuration, Location tab**

**Reference:**
https://docs.devicewise.com/Content/Products/Management_Portal/Things/Things-Dashboard.htm
https://docs.devicewise.com/Content/Products/Management_Portal/Things/Thing-Location-History.htm

https://docs.devicewise.com/Content/Products/Management_Portal/Developer/ThingDef/Defining-Locations.htm

# Connections

This section of the **Management portal** manages SIM cards (as called as connections) belonging to the organization.

The connections are listed in the same way as the things and useful feature and information related to the connectivity are present for each connection.

The most important are listed below:

- Details: configuration information
- Usage: details related to data consumption
- SMS: the possibility to send SMS messages

> **Warning:** The "Remote AT" through Connection is based on SMSATRun feature which is NOT supported by OneEdge devices.

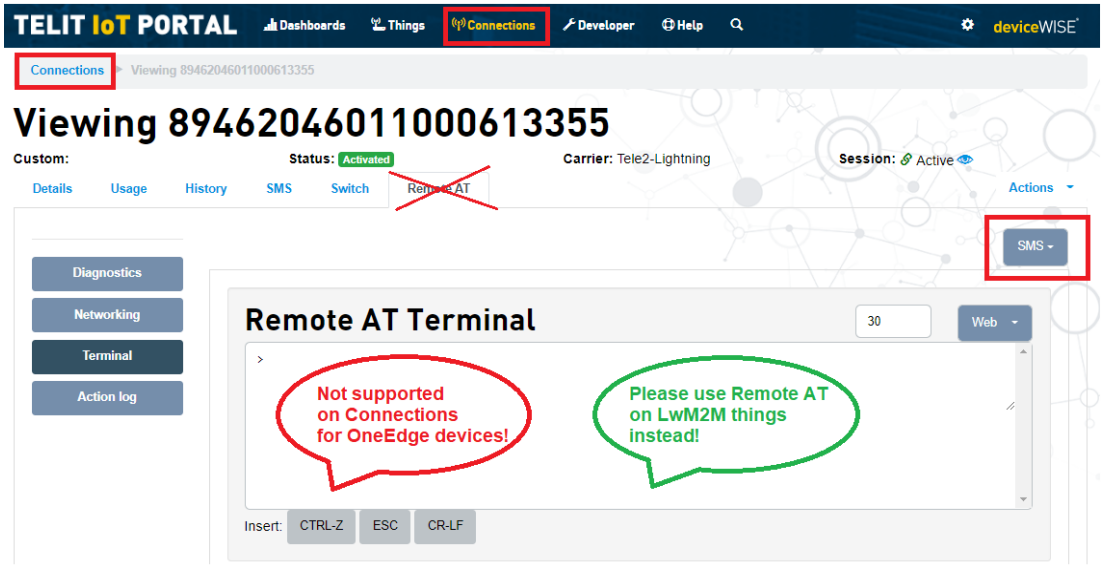



Figure 65: Comparison between "Remote AT" feature on Thing and on Connection

The **Remote AT** through LwM2M must be used instead.

If the SIM card of a LwM2M thing is managed on the same organization, then the connection and the thing are linked together such that the User can navigate the connection details from the LwM2M thing.

**Reference:**
https://docs.devicewise.com/Content/Products/Management_Portal/Connections/Connections.htm

# Developer

This section provides the tools and configurations needed to build applications using the Management Portal.

The most important and **mandatory configuration elements** for OneEdge applications are "Thing definitions" and "Device profiles": the first one is related to the graphical aspect of the thing and the latter is the LwM2M configuration for the thing.

"Triggers" and "Campaigns" are **powerful tools** to provide event-driven actions and data post-elaboration.

Other tools are present on the OneEdge IoT portal for **debug, logging and troubleshooting**, for example "Event log", "API log" and "API tester".

The tools named above will be briefly explained in the next paragraphs.

**Reference:**
https://docs.devicewise.com/Content/Products/Management_Portal/Developer/Developer.htm

## Thing Definition

A Thing definition is used to define all the characteristics related to the **aspect** of a thing.

**Multiple things** can belong to the same thing definition such that the thing definition can be modified in one place only to apply the new aspect to all the things associated to it.

The most used elements in OneEdge applications are the attributes, properties and alarms and are used to report the information of the things in **different graphical ways** on the thing web page:

**Attribute**: information container suitable for texts and showing the last value; the history is available inside each attribute

**Alarm**: a graphical representation of information based on multiple states, suitable for levels, for example On/Off, Open/Closed, Low/Medium/High

**Property**: a graphical representation of the information through several chart types, suitable for numerical data

Refer to paragraph *Viewing a Thhing* for an example about attributes, alarms and properties aspect.

The Figure 66 below shows an example of an existing Thing definition with 4 properties and specific settings for "**Temp**" property chart.

The type of the chart is under the "**Presentation**" tab.

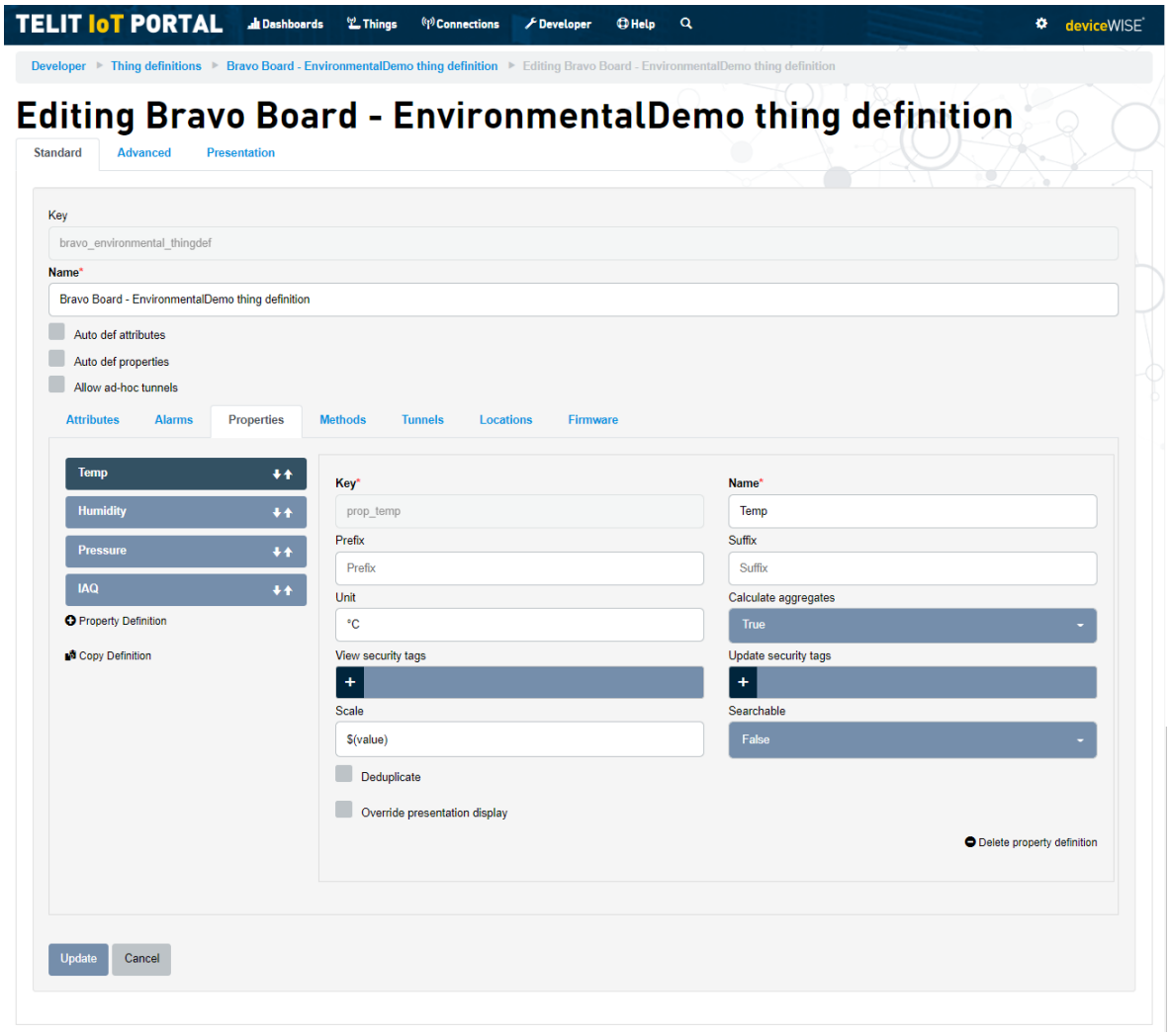



**Figure 66: "Thing definition" configuration and "Temp" property details**

Furthermore, the last "**Firmware**" tab contains the details for delta files for FOTA upgrade.

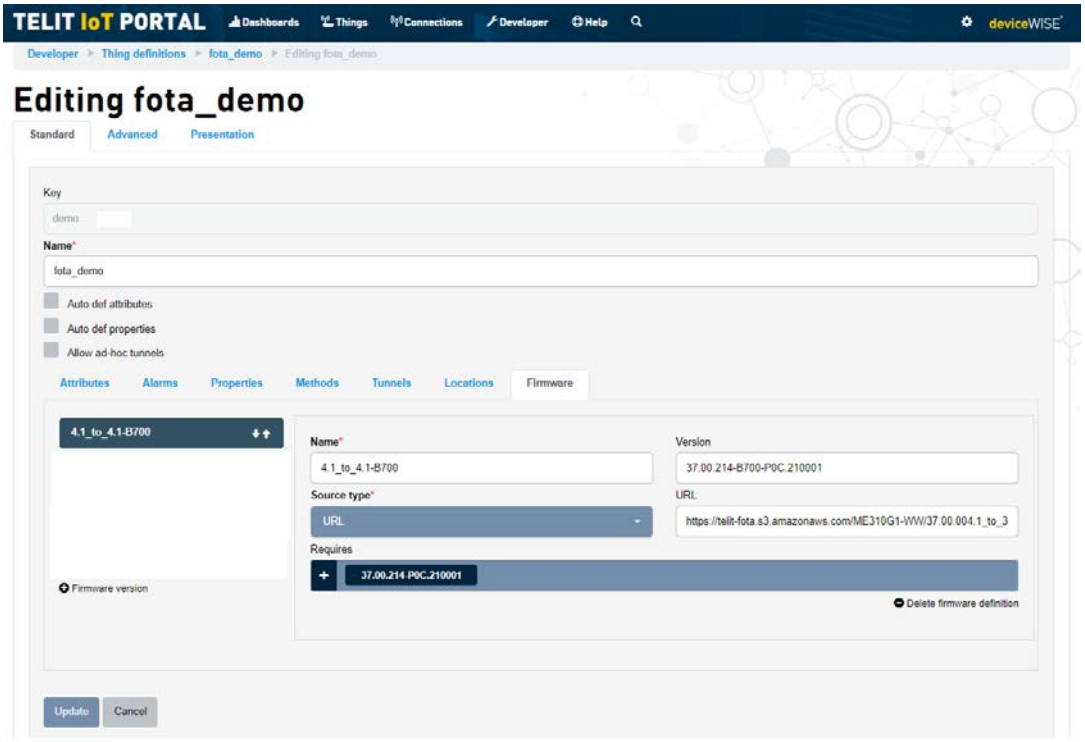



Figure 67: "Firmware" details configuration on Thing definition

**Reference:**
https://docs.devicewise.com/Content/Products/Management_Portal/Developer/ThingDef/Thing-definitions.htm

https://docs.devicewise.com/Content/Products/Management_Portal/Developer/ThingDef/Defining-Attributes.htm

https://docs.devicewise.com/Content/Products/Management_Portal/Developer/ThingDef/Defining-Properties.htm

https://docs.devicewise.com/Content/Products/Management_Portal/Developer/ThingDef/Defining-Alarms.htm

## Device Profile

All the details related to the LwM2M configuration of the thing are defined in the Device profile associated to the things.

For this reason the Device profile is a fundamental element for OneEdge applications.

Furthermore, multiple things can be associated to the same Device profile and this improve and speed up the development and maintenance.

The main configuration sections of the Device profile will be explained in next paragraphs.

**Reference:**
https://docs.devicewise.com/Content/Products/Management_Portal/Developer/DeviceProfiles/Device-Profiles.htm

## Reads

The "**Reads**" section contains the automatic LwM2M operations (read and observation) that the server can perform in case of the main LwM2M events: full registration and registration update.

The Figure 68 below shows an example of 2 automatic actions applied to URI path /26251/0:

No read in case of full registration ("Read on register" selector)

Read /26251/0 for every registration update ("Read on update" selector)

Observation on /26251/0, set up during every full registration ("Observe" selector)

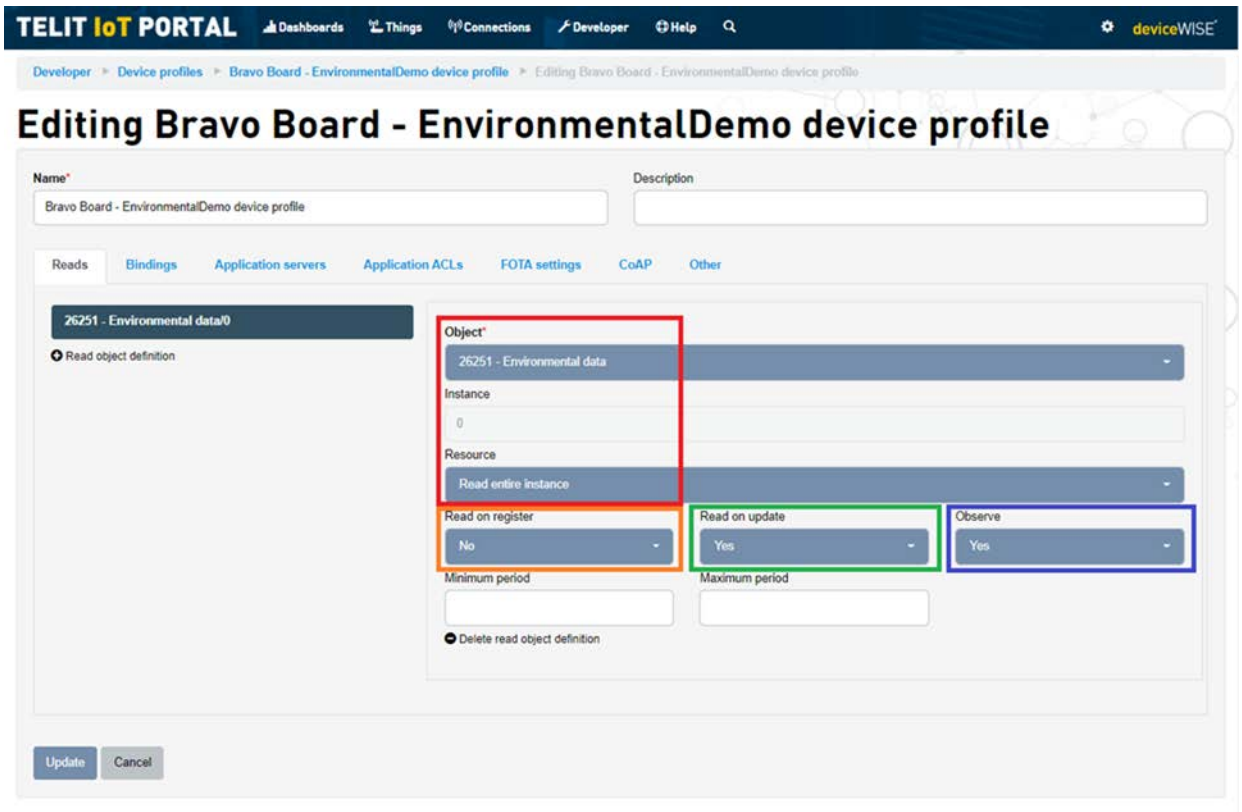



Figure 68: Example of "Reads" section in a Device profile.

**Note:OneEdge:** The Reads feature on the Device profile is a very appreciated functionality. A typical example is the server that performs the following actions:

Read the "**Device**" object (ID 3) every time the LwM2M client performs the full registration.
Read the "**Connectivity Management**" object (ID 4) at every registration update.
This feature is called as "**Device Management**" in OneEdge environment.
lifetime = DeviceManagement_frequency

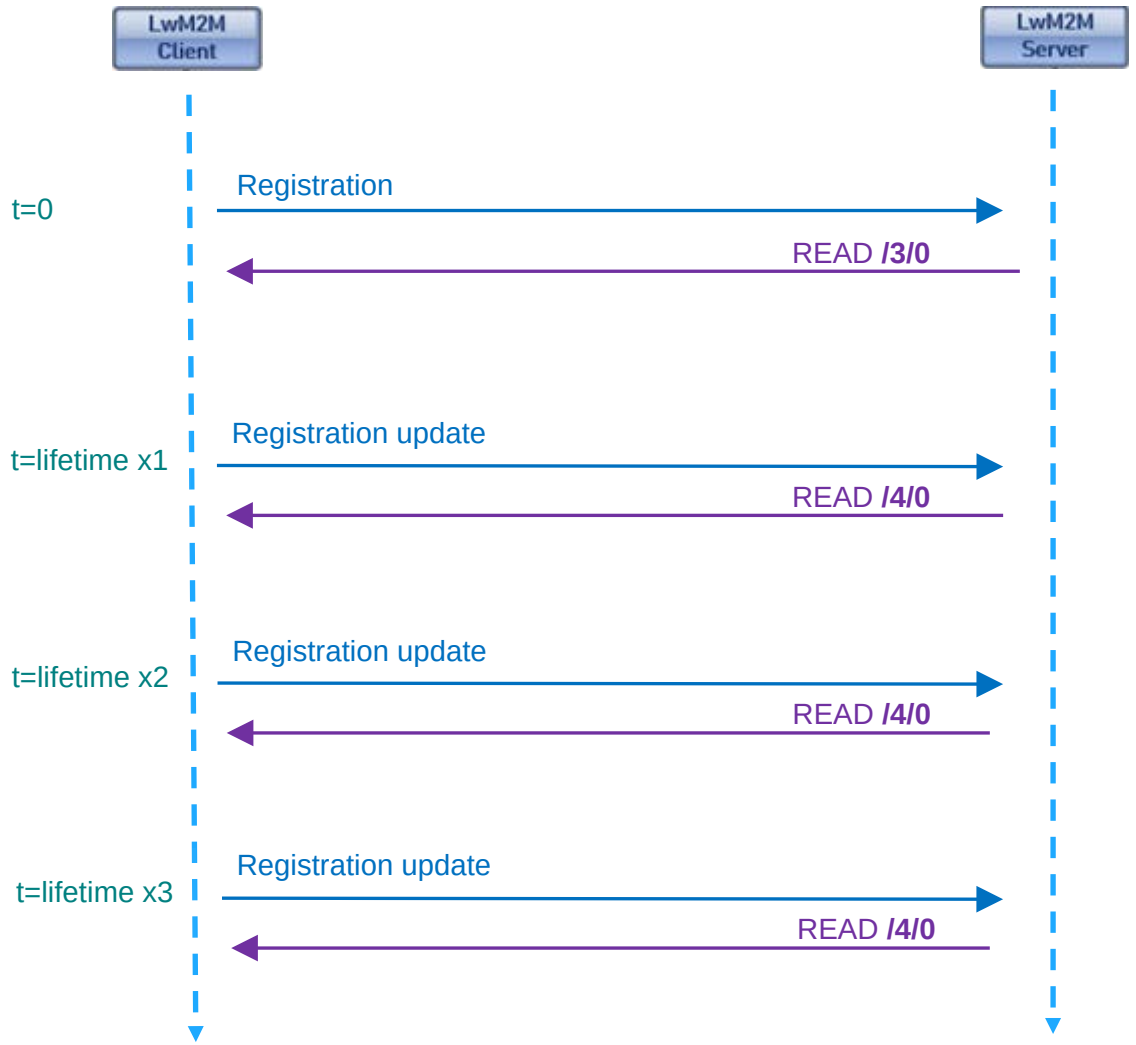



**Figure 69: Example of OneEdge "Device Management" feature and relationship with lifetime**

## Bindings

Another important section of the Device profile is the binding.

It is used to link a LwM2M resource with an attribute or a property on the thing definition.

In this way, every time the LwM2M server receives a value of a LwM2M resource, the value is **automatically published** on the attribute or the property configured on the binding.

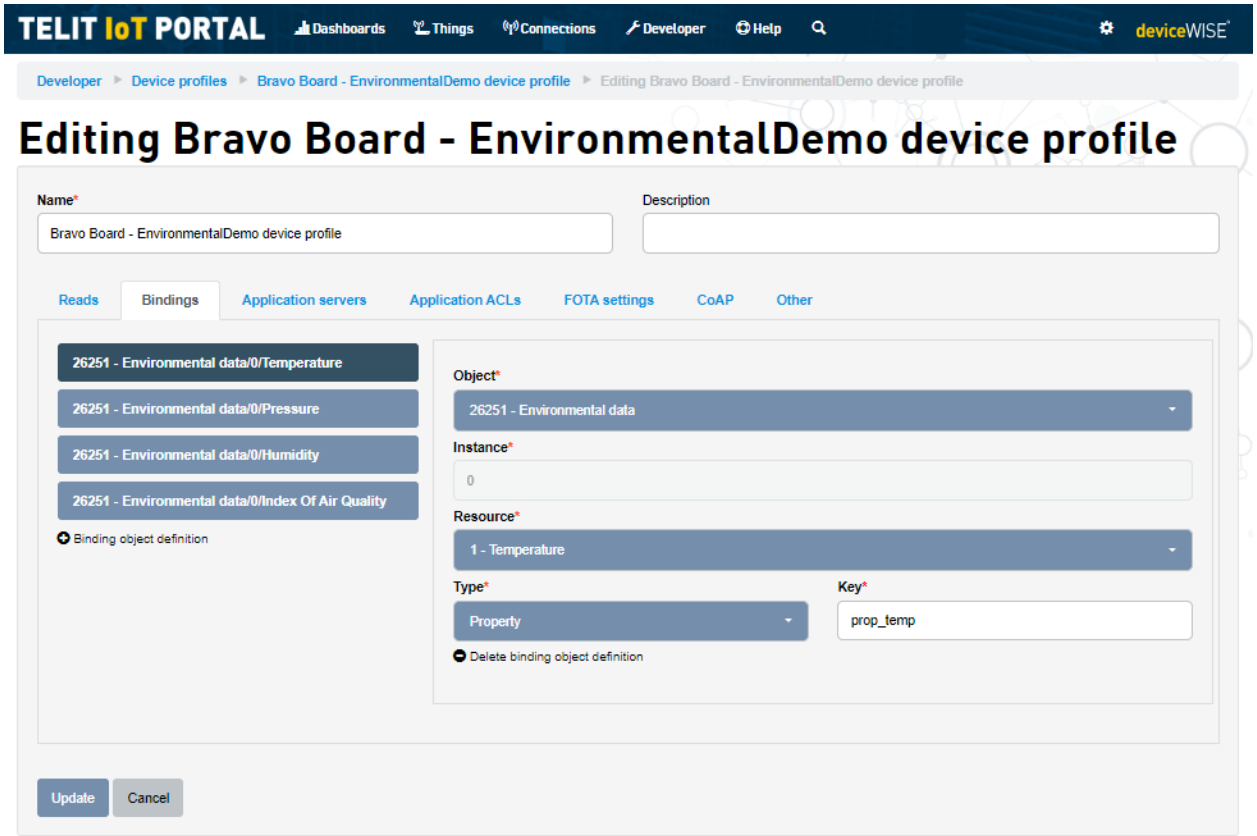



**Figure 70: Example of 4 bindings between 4 resources of "Environmental data" object and 4 properties**

The Figure 71 below shows the logical data flow of 2 bindings configured in the "Binding demo" device profile:

- Resource 3/0/0 **Manufacturer** to the attribute **attr_manuf**
- Resource 4/0/2 **RSSI** to the property **prop_rssi**

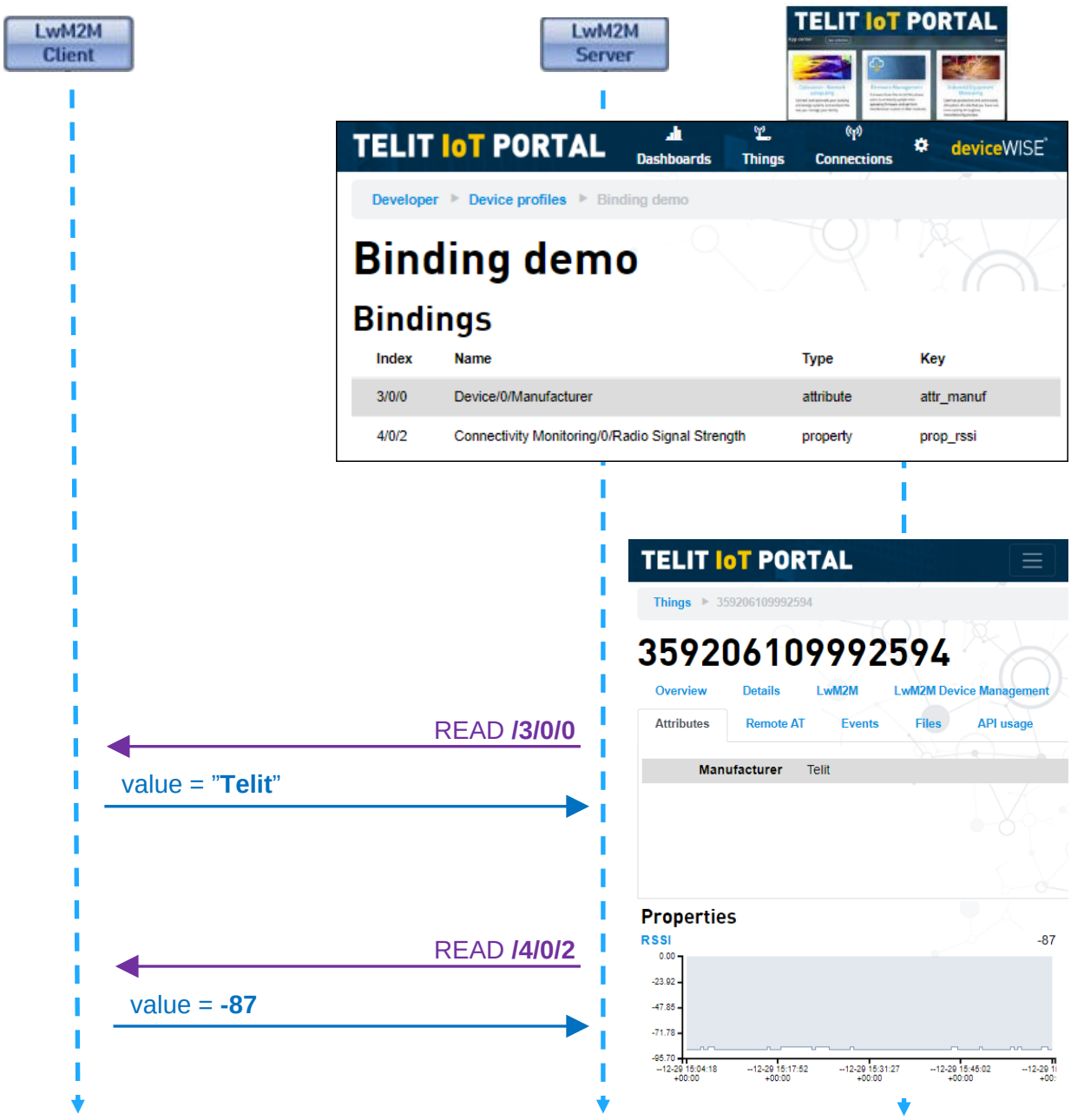



Figure 71: logical flow of binding behavior

**Reference:**
https://docs.devicewise.com/Content/Products/Management_Portal/Developer/DeviceProfiles/Defining-Bindings.htm

## Application Servers

This section of the device profile is used to configure the LwM2M parameters that the Bootstrap server will provision to the LwM2M client at the Bootstrap phase.

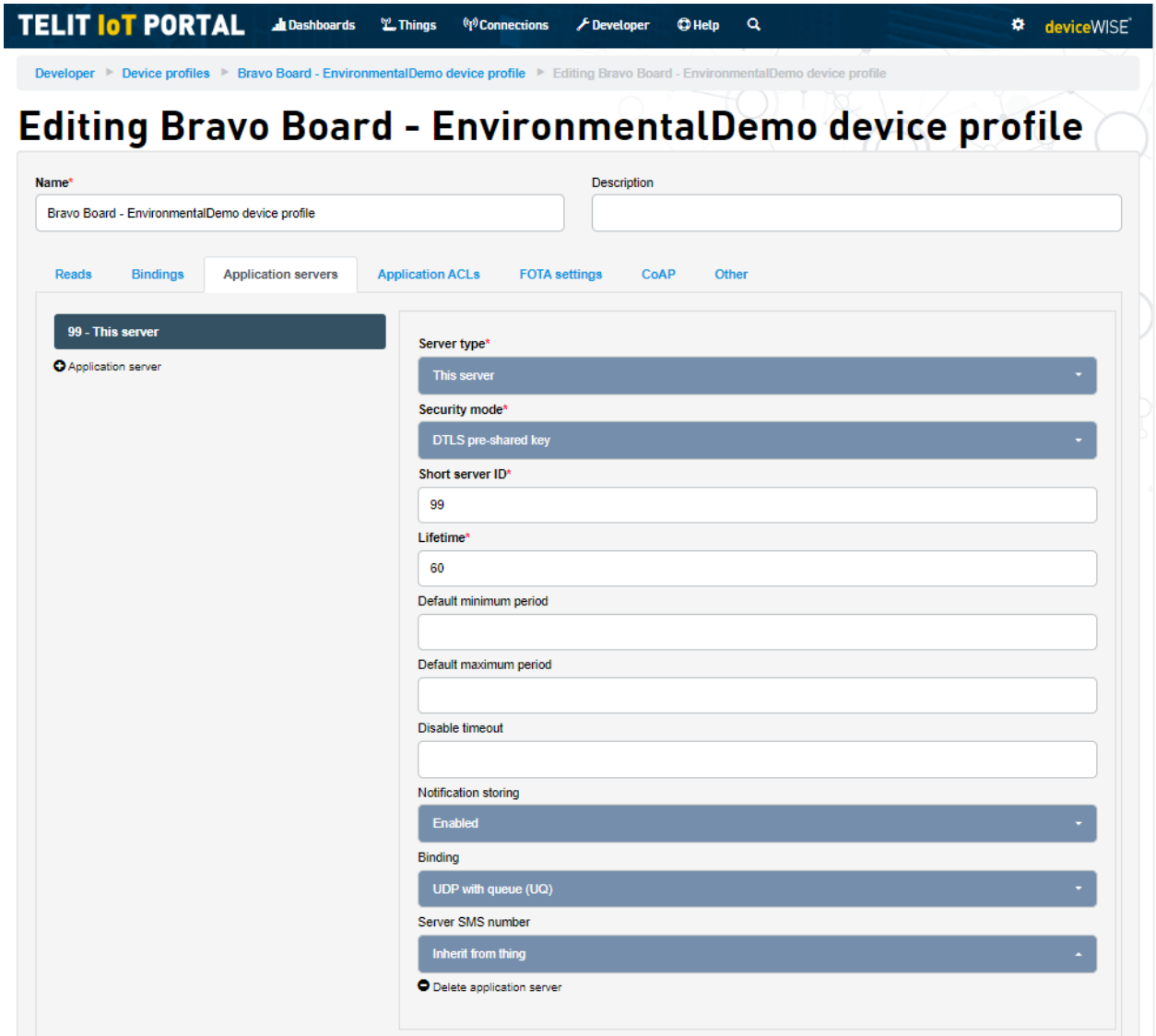



Figure 72:Example of parameters to be configured on BS server.

## Triggers

The Cloud Triggers provide event-driven (Cloud trigger events) functionalities to instantiate the appropriate actions (Cloud trigger actions).

Basically, a trigger is a sequence of actions performed by the trigger engine on the OneEdge IoT portal.

The trigger is activated by an **event** occurred on the OneEdge IoT portal; the available events are several and various, for example, the LwM2M registration or the registration update of a device, a LwM2M resource change, or the connection/disconnection event on a SIM connectivity.

The available **actions** are grouped on **categories** (on the left side of the trigger draw panel) and below is a list of some typical actions.

The trigger can:

- Execute operations (i.e. lwm2m.oir.read action)
- **Acquire data** from attributes and properties (i.e. property.current action)
- Perform **mathematical and string elaborations** (i.e. math.if and string.split)
- Set a value to a **trigger variable** (i.e. trigger.var.set action)

- Send data to **third party clouds** (i.e. aws.lambda.invoke and azure.event_hub.send actions)
- Send data to a generic HTTP/HTTPS server and an email (i.e. http.client and email.send actions)

The way to develop a trigger is very easy and fast and doesn't require any knowledge of programming languages.

Below the basic steps are listed:

The first step is to identify the event that activates the trigger.

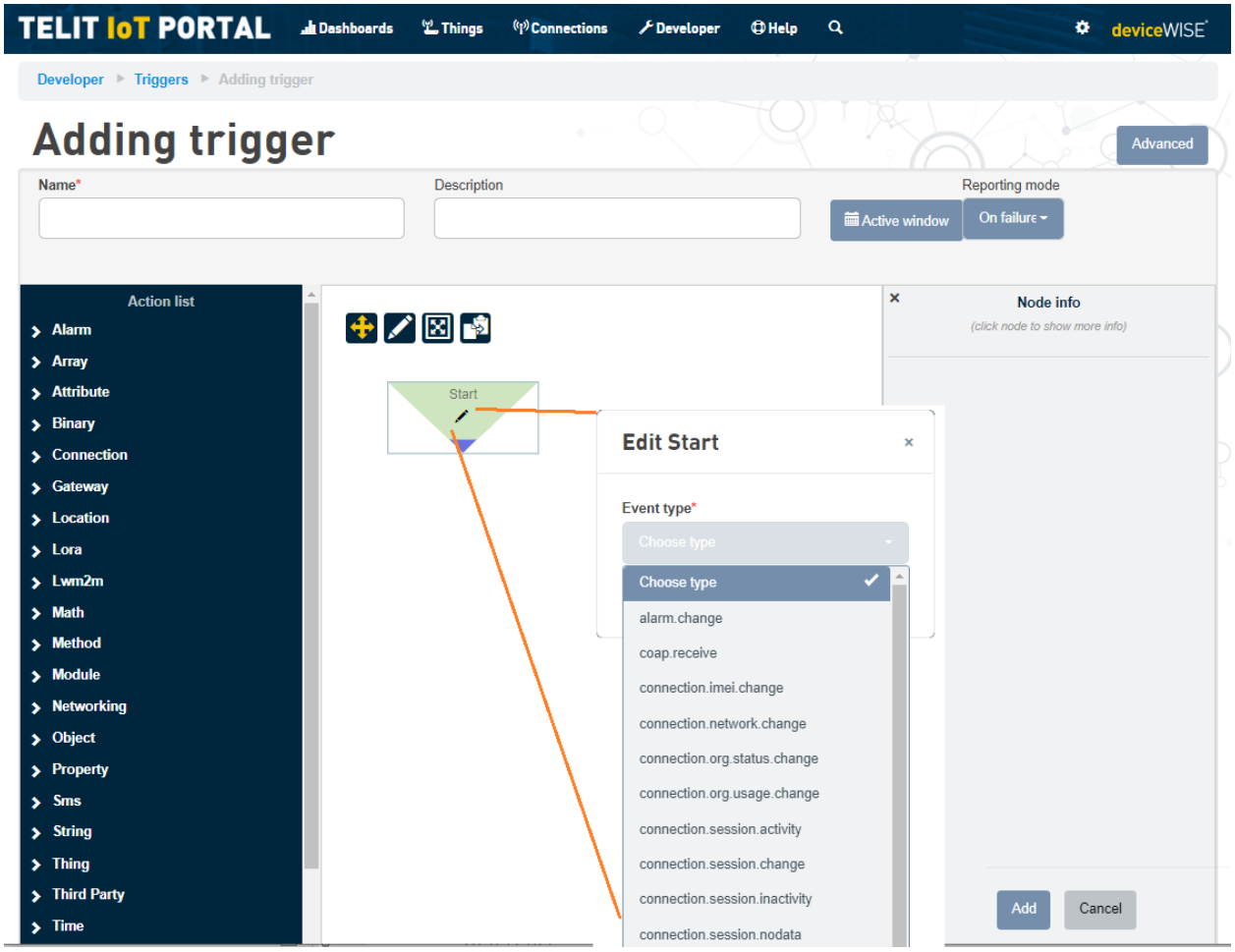



Figure 73: Trigger event selection

Then the User search for an action from the categories on the left side of the trigger draw panel and configure the fields of the elements.

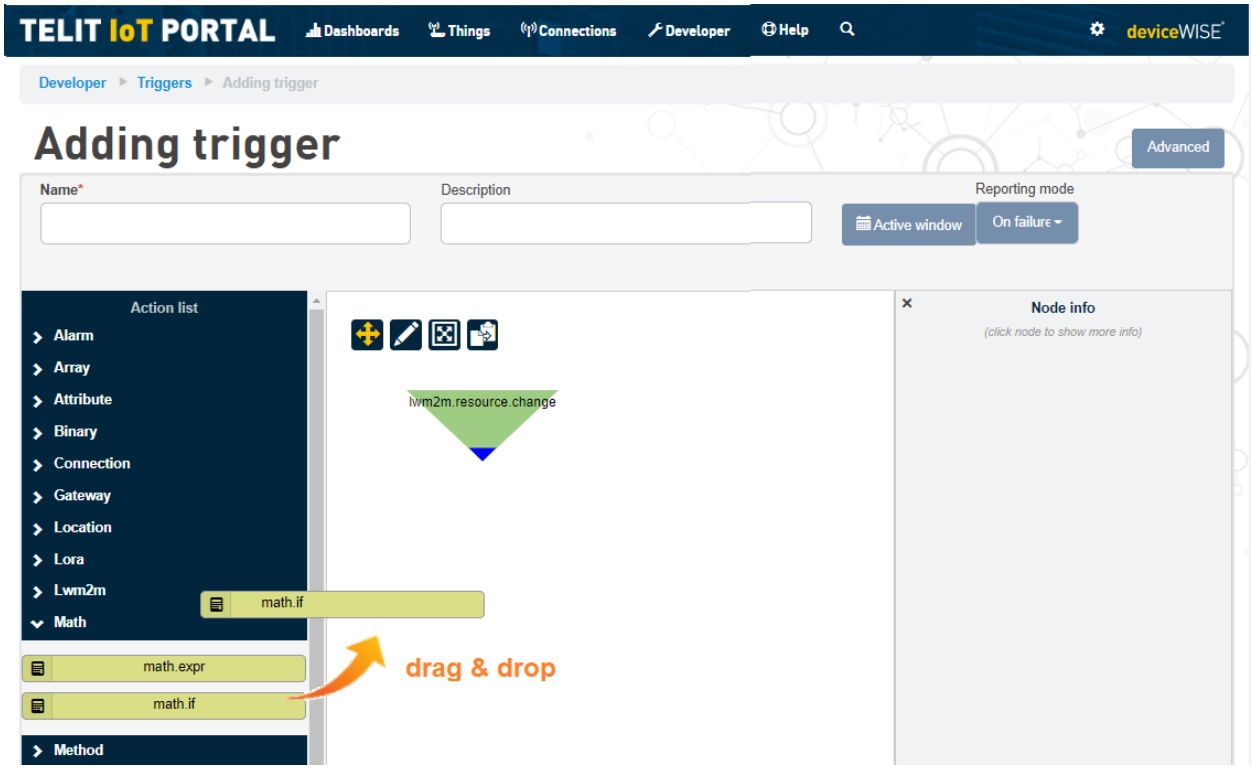



Figure 74: adding an action through drag&drop

Connect the elements through arrows

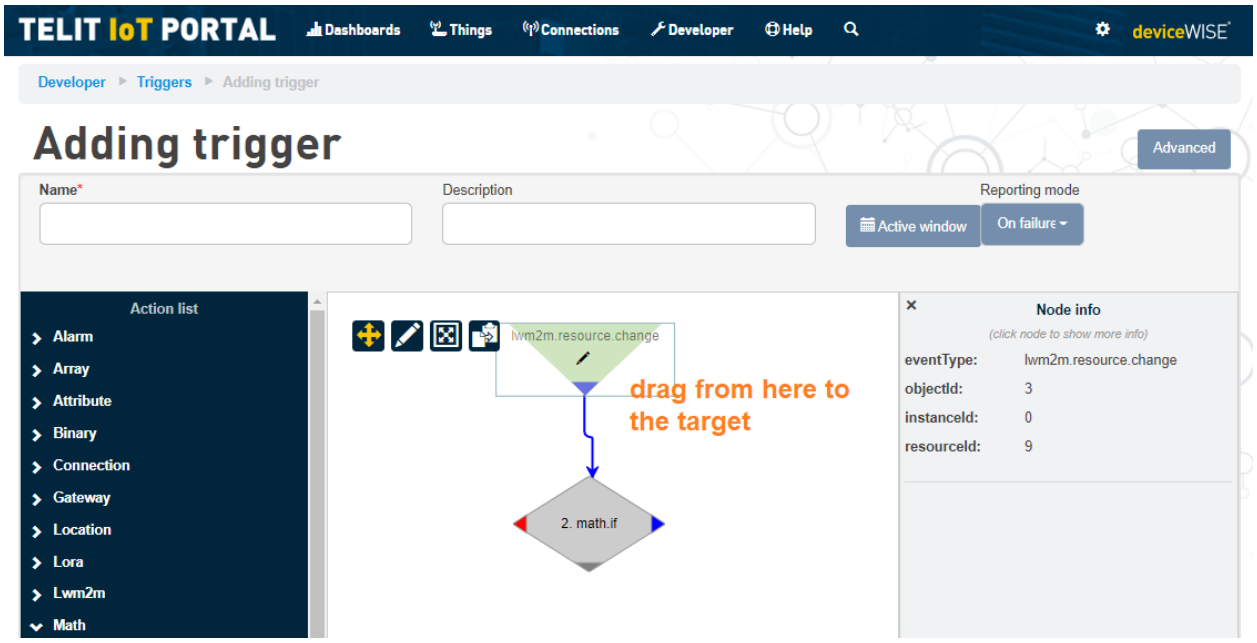



Figure 75: Connect the trigger elements.

Connect the "**Success**" and "**Failure**" special actions.

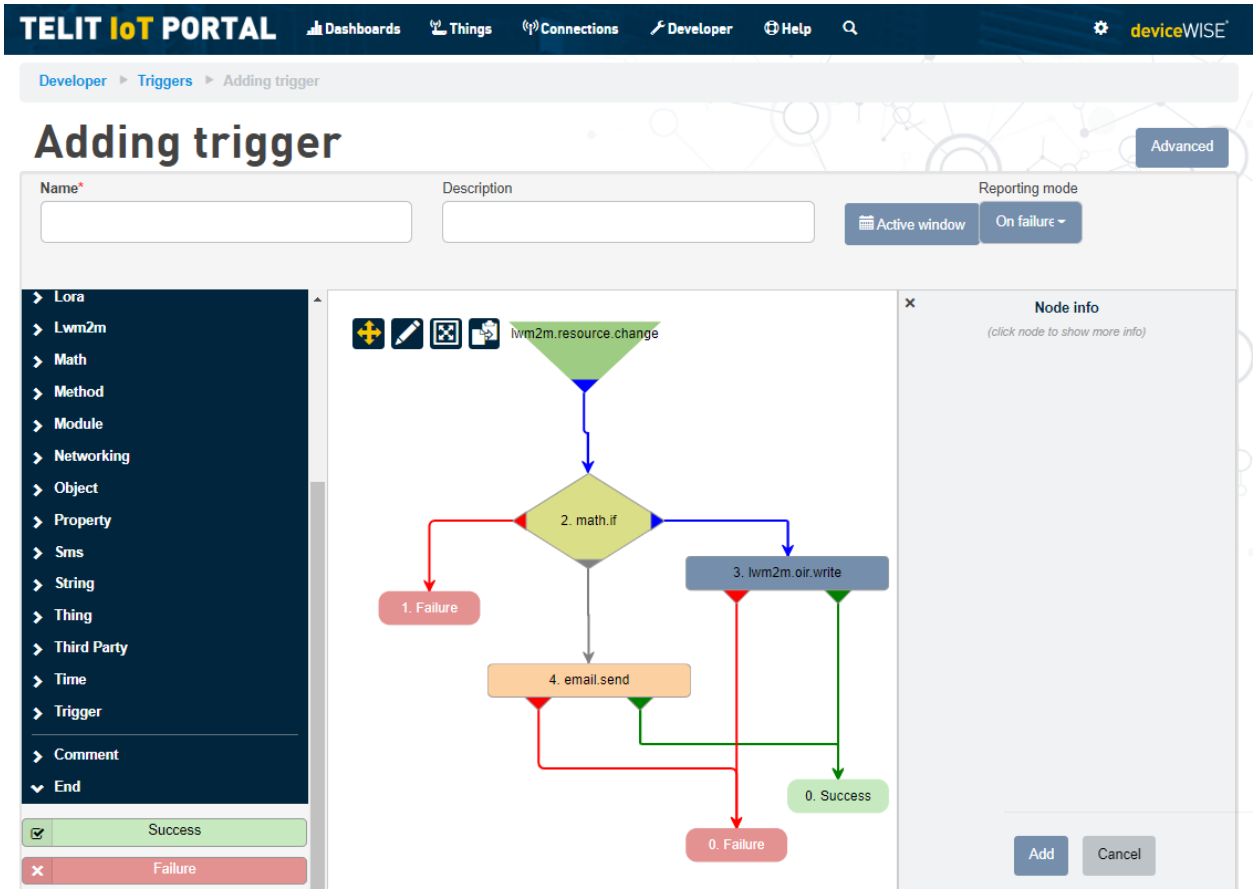



Figure 76: Success and Failure special elements

More details are available on online documentation on OneEdge IoT portal.

**Reference:**
https://docs.devicewise.com/Content/Products/Management_Portal/Developer/Triggers/Triggers.htm

## Campaigns

The campaigns are a tool to perform a set of operations on multiple things or connections of the same organization.

For example, a campaign can be used to add or delete a tag to a group of things or to write the same value to a LwM2M resource of many devices.

The things and connections to be involved in the campaign execution are called as "**members**" and can be selected by different ways like thing/connection key, tag and lists.

The campaign is developed in the same graphical way as the triggers.

**Reference:**
https://docs.devicewise.com/Content/Products/Management_Portal/Developer/Campaigns/Campaigns.htm

## API log and API Tester

The API log reports all the API executed to the server by the User or a thing or a connection, so it's a valid troubleshooting and logging tool.

The API tester is a tool to allow the OneEdge developer to test the API calls before implementing them in an application.

**Reference:**
https://docs.devicewise.com/Content/Products/Management_Portal/Developer/APILog/API-log.htm

https://docs.devicewise.com/Content/Products/Management_Portal/Developer/APITester/API-tester.htm

### Administration

The administration section provides advanced features for organization admin users.

As default view, some usage data graphs and tables are showed.

Other interesting features are roles and users, where different access rights can be defined to different kinds of users, for example "view only" and where new users can be invited.

> **Warning:** Administration settings should be modified carefully because can cause strong effects. Refer to OneEdge technical support in case of doubts.

**Reference:**
https://docs.devicewise.com/Content/Products/Management_Portal/Administration/Administration.htm

## 11.2 Device Management Application

The "**Device Management**" web application provides a view of the main diagnostic parameters of the devices according to the info on LwM2M objects Device, Connectivity Monitoring, Location and Connectivity Statistics (object identifiers 3, 4, 6, 7).

The Figure 77 below shows the home page related to the OE-AE-EMEA org on Development portal as example.

The device parameters are represented according to the most suitable chart on the boxes on the webpage structure.

The collection of the device parameters to be showed on the boxes are fixed and cannot be customized by the User.

> **Warning:** The collection of the device parameters to be showed on the boxes is fixed and cannot be customized by the User.

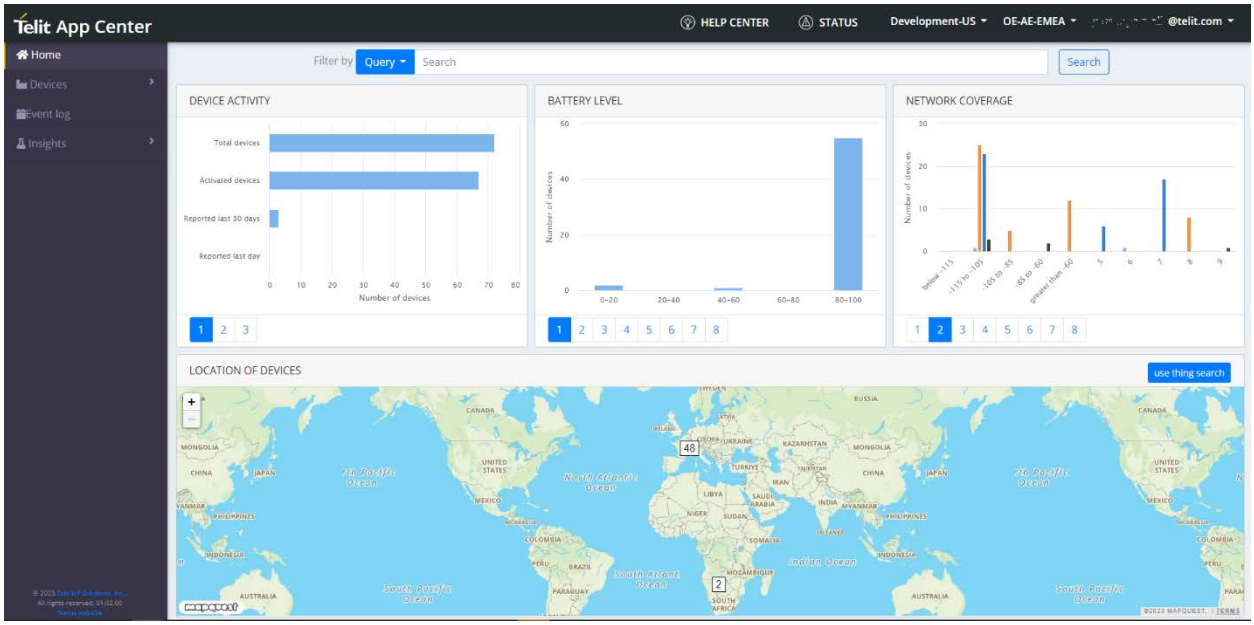



Figure 77: "Device Management" app home page

The devices and most representative parameters (last seen, RSSI, battery level) are represented also by a list and a map as showed on Figure 78 below.

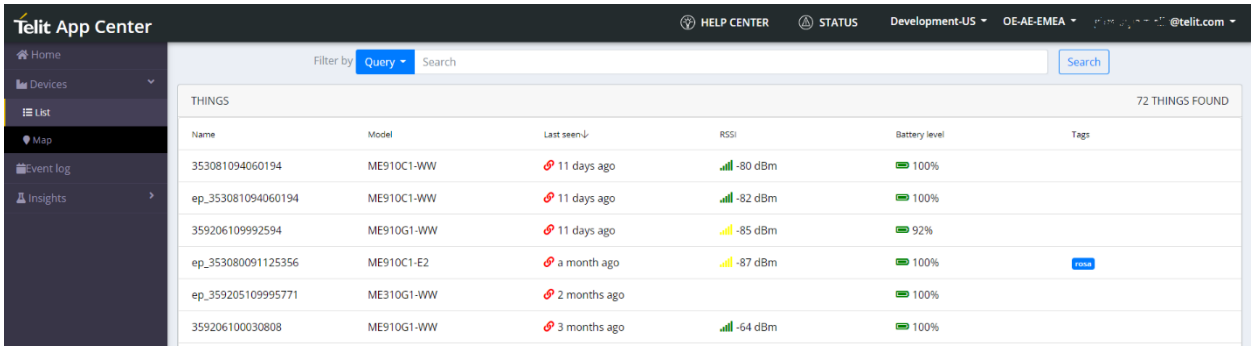



Figure 78: Devices represented as list of things.

Furthermore, also some insight charts are available.

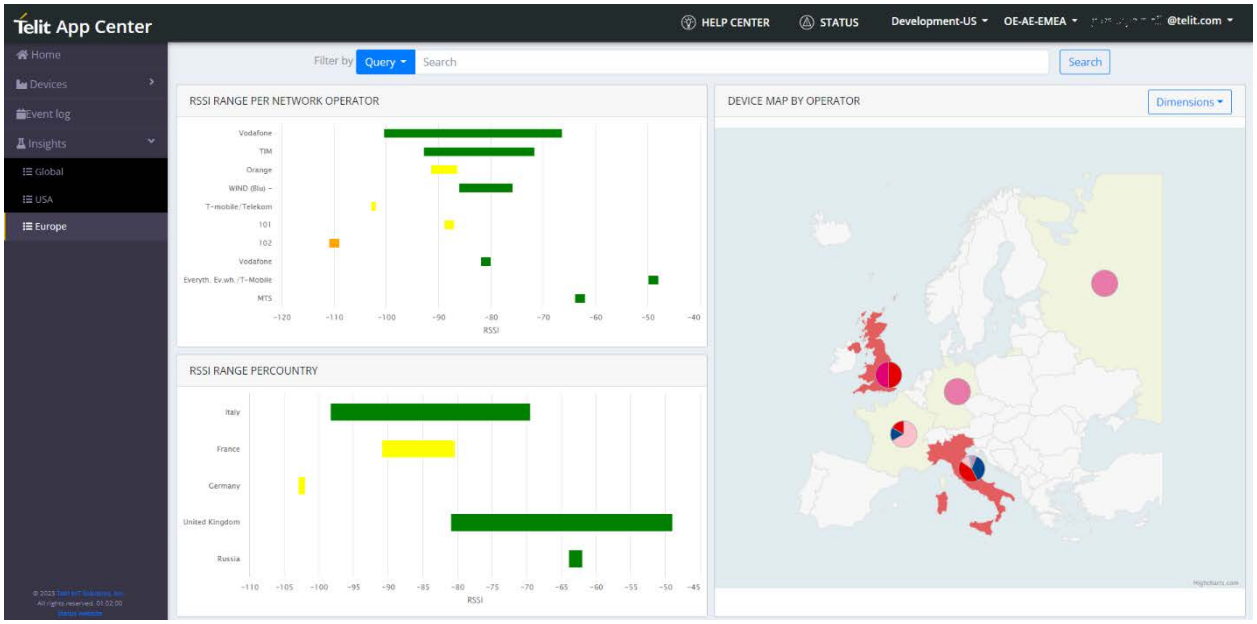



Figure 79: "Insights" section

## 11.3 Firmware Management Application

The **Firmware Management** web application provides a graphical interface to configure and run Firmware upgrade campaigns.

The main functionality of this application is to allow the User to create a campaign through a step-by-step guided procedure to upgrade the firmware of multiple devices on the field all together.

The figure 80 below shows the home page related to a demo org on Development portal as example.

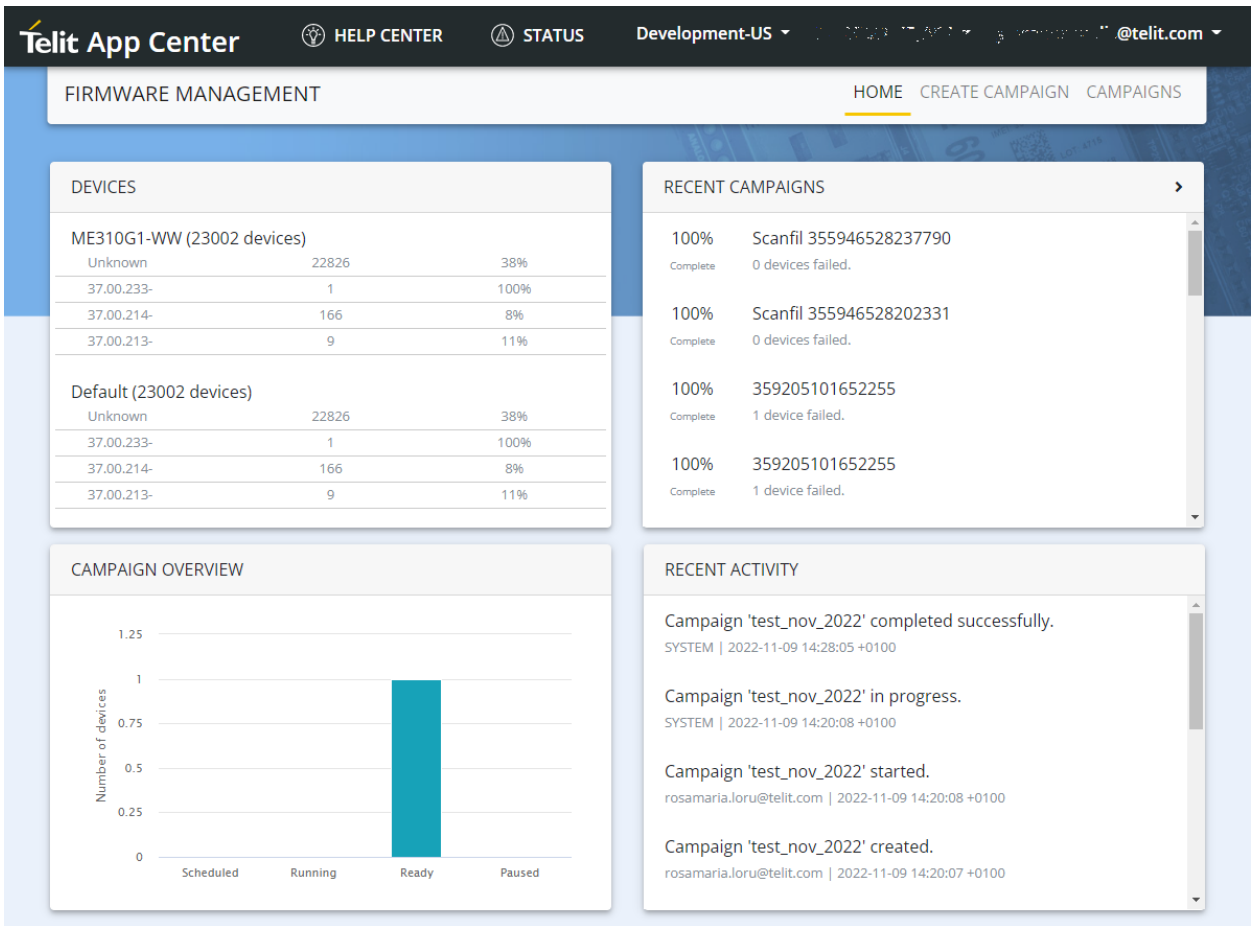



**Figure 80: "Firmware management" home page**

For more information about the **Firmware Management** application and the procedure to upgrade the devices through this app, refer to **OneEdge Firmware Management app** application note [14].

## 11.4 Users

Every organization on the OneEdge IoT portal must have at least a user, with username and password to access to the web applications of the portal.

Every user can have administrator rights or be assigned to a role with specific sets of features allowed and forbidden.

For example, a user assigned to “view only” role can view all the parts of the portal but cannot edit and modify them.

**Reference:**
https://docs.devicewise.com/Content/Products/Management_Portal/Administration/Users/Users.htm

## 11.5 TR50 and REST API interfaces

The OneEdge IoT portal provides a HTTP/HTTPS interface to allow external applications to interact with the functionalities on the portal.

Typical external applications can be, for example:

a script on a machine managed by the customer, that access to the portal every day to read some values, for example retrieve statistical data

a routine on the customer cloud interface that perform some actions on the OneEdge IoT portal according to some events

a smartphone application to allow the end users to manage their devices easily from the smartphone

These scripts and routines must be developed to connect to a specific IP address and port and send commands according to the TR50 format and manage the responses from the portal.

**Reference:**
https://docs.devicewise.com/Content/Products/IoT_Portal_API_Reference_Guide/IoT-Portal-API-Reference-Guide.htm

# 12 Typical OneEdge Applications

Since the LwM2M protocol provides different functionalities, also the applications can be designed in different ways.

In this chapter some examples will be provided with advantages and drawbacks.

## 12.1 Battery-Powered Applications

In case of battery powered devices, the host application usually keeps the modem powered off for most of the time and switch on it periodically for a short time just to send the data.

The following paragraphs describe two strategies to deliver data to the OneEdge IoT portal.

### Data Sending Through Observation

The Device profile on OneEdge IoT portal can be configured to automatically set an observation when the LwM2M client registers to the server (Refer to paragraph Reads.)

The Figure 81 shows an example of observation usage and the highlighted info that the host application should use as “data delivery” feedback.

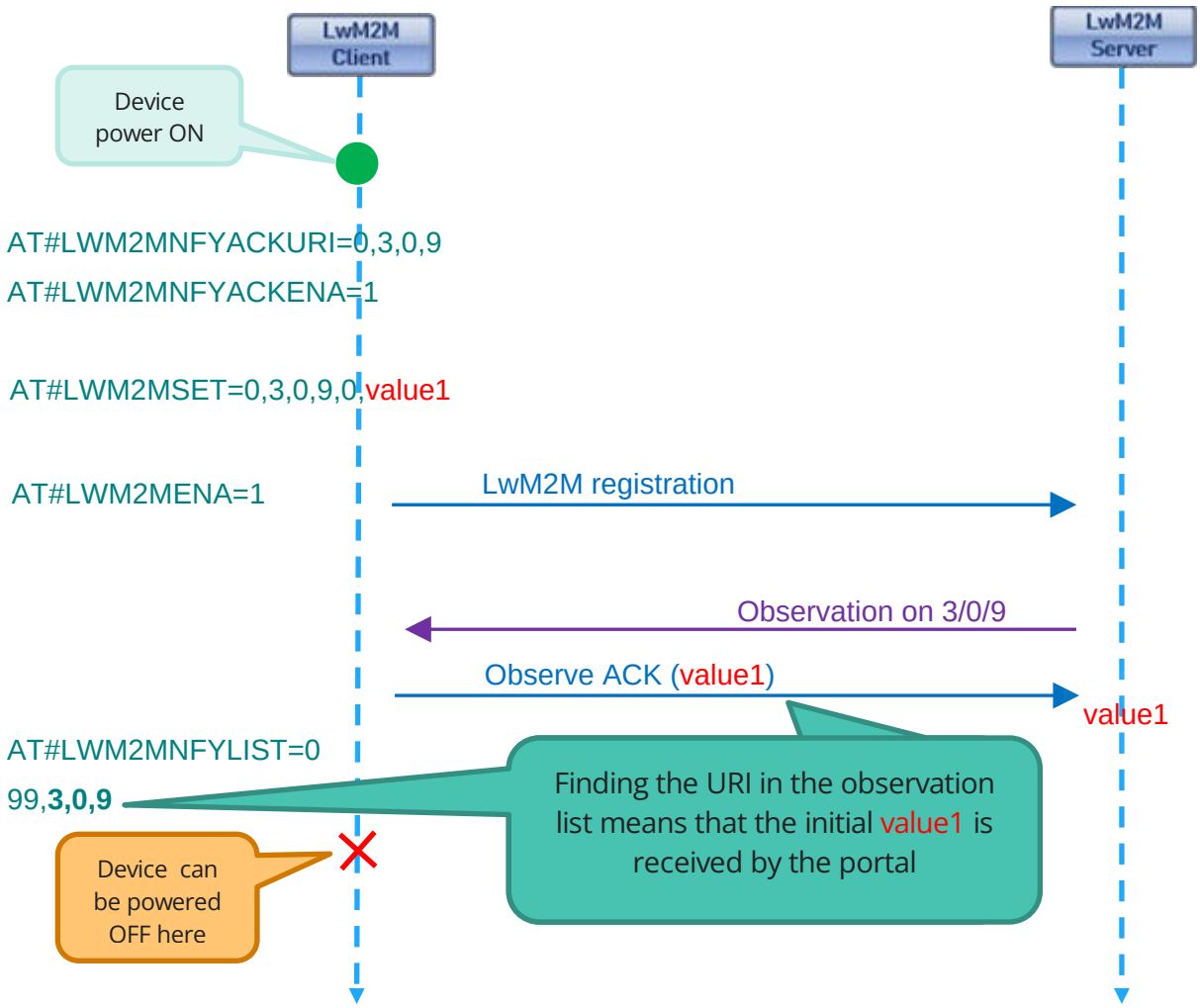



**Figure 81: Example of observation usage where the “data delivery” feedback is highlighted.**

# Data Polling Through Trigger

A different/additional strategy to deliver data from the device to the OneEdge IoT portal is to manage the "polling" data from the portal through a **trigger**.

For example, when the LwM2M client registers to the server, this event can be used to raise a trigger (refer to trigger section on Management Portal Application) that can read objects and resources and apply custom logics according to the values.

An example of customer use case could be that the OneEdge IoT portal sends a parameter if the battery level (resource 3/0/9) is below a specified threshould.

The Figure 82 shows the flow diagram with the relevant events for the use case above.

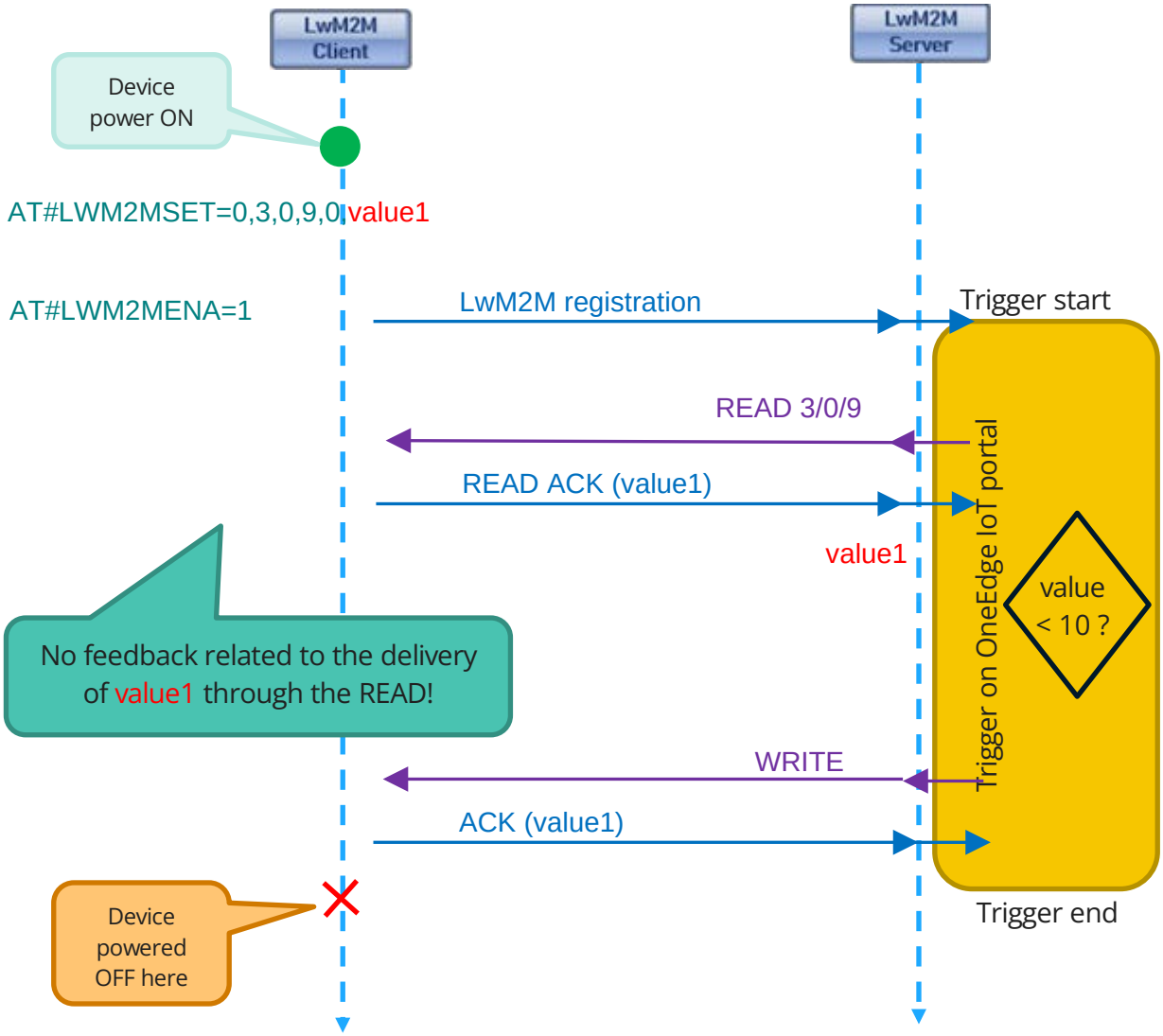



Figure 82:Example of polling data through a trigger

# Remote Settings from the OneEdge IoT Portal

The User can send a setting to battery-powered devices on the field through campaigns and triggers.

There are different behaviors depending on the lifetime value:

a If the campaign sends the LwM2M message to the device when the device is switched on and the LwM2M client registered, the LwM2M message is received by the device.

Anyway, this scenario is very unlikely because battery-powered devices stay switched on for very short period, so hitting the power-on gap for every device on the field is not the right strategy

b If the device is powered off but the **lifetime** has **not expired** yet, the **Campaign** sends the LwM2M message to the device, which cannot receive the message so it will be stored on the server queue and delivered at the next registration event.

c If the device is powered off but the **lifetime** is **already expired**, the OneEdge IoT portal considers the device as "Disconnected" (not registered at LwM2M level), so the Campaign fails and cannot deliver the LwM2M message in this case: a trigger fired by LwM2M registration can be used instead, as explained in paragraph Data Polling Trough Trigger

In all the cases, refer to chapter 4 for more details.

## 12.2 Always on Applications

In case of supply-powered devices, the host application usually keeps the modem powered on for all the time and the LwM2M client is always registered.

Also in this case both observation and read through trigger can be used: these scenarios are explained in the following paragraphs.

### Data Sending through Observation

When the LwM2M client is active for long time or all the time, the host application usually uses the observation to send the data periodically or alarms when specific events happen.

Furthermore, the LwM2M client can also receive incoming data at any time, with some constrains and rules (refer to chapter 4 for more details).

The Figure 83 shows an example of observation usage in always-on device.

**Figure 83: Example of observation usage in always-on device: the "data delivery" feedbacks are highlighted.**

## Data Polling Through Trigger

The concepts described on paragraph Polling Trough Trigger are still valid also in case of always-on devices: in this case the event to raise the trigger could be the data application receiving or the registration update.

# Remote Settings from the OneEdge IoT Portal

Also in case of always-on devices, the User can send a setting to devices on the field through campaigns and triggers.

Anyway, this kind of devices are always **Connected** at LwM2M level because the lifetime is kept refreshed by the LwM2M registration update mechanism.

The User can send settings to the devices in the field through a **campaign**:

If the device can receive the incoming message and the IP path through NAT is open, the campaign delivers the message

If the device is not able to receive the incoming message, the LwM2M message will be stored on the server queue and delivered at the next registration (either full and registration update) event.

Anyway, in some cases, the User is interested in more complex logics, for example reading a LwM2M resource and then sending different settings according to the read value: in this case the flow cannot be managed through queued messages, so a trigger should be used instead.

For these reasons, the **campaign** is the preferred way for **simple settings** and the **trigger** is more suitable for **complex scenarios**.

In all the cases, refer to chapter "LwM2M Registration Interface" for more details about NAT and OneEdge Direct concepts.

# 13 OneEdge Counters

## 13.1 API Usage, LwM2M Messages and Billing

The OneEdge service is usually based on monthly fee that includes a certain amount of transactions between the LwM2M client and server and on the OneEdge IoT portal.

According to Figure 84 below, the “transactions” are divided into 2 categories:

- LwM2M messages
- REST APIs.

Every LwM2M command between a client and the server is counted as a LwM2M message, for example the registration, a read or a notify.

> **Warning:** Every LwM2M command includes a response, but the responses and acknowledges are considered as part of the same transaction and not counted as a LwM2M message.
>
> For Example:
>
> The client sends the registration message, and the server replies with “2.01 Created”: this is counted as 1 LwM2M message.
>
> The server sends a read command, and the client replies with “2.05 Content”: only 1 LwM2M message.
>
> The client sends a notify **2.05 Content** and the server replies with the ACK: again 1 LwM2M message.
>
> Furthermore, if a command involves the BlockWise CoAP feature, every block is counted as 1 LwM2M message.
>
> For Example:
>
> The server, reading an object of 4KB of payload, sends 4 read commands and the client replies with **2.05 Content** for 4 times: 4 LwM2M messages.

On the other hand, all the actions done on the OneEdge IoT portal through the HTTP/TR50 interface are counted as **APIs**, for example an application running on the customer external cloud calling the **thing.attribute.history** command to collect the samples of the day.

Another example is a smartphone application that allow the end user to change a parameter on the remote device.

Most of the TR50 commands are counted as 1 API, but some **batch** commands consumes more: Refer to the specific section **Understanding API Weights** on online documentation.

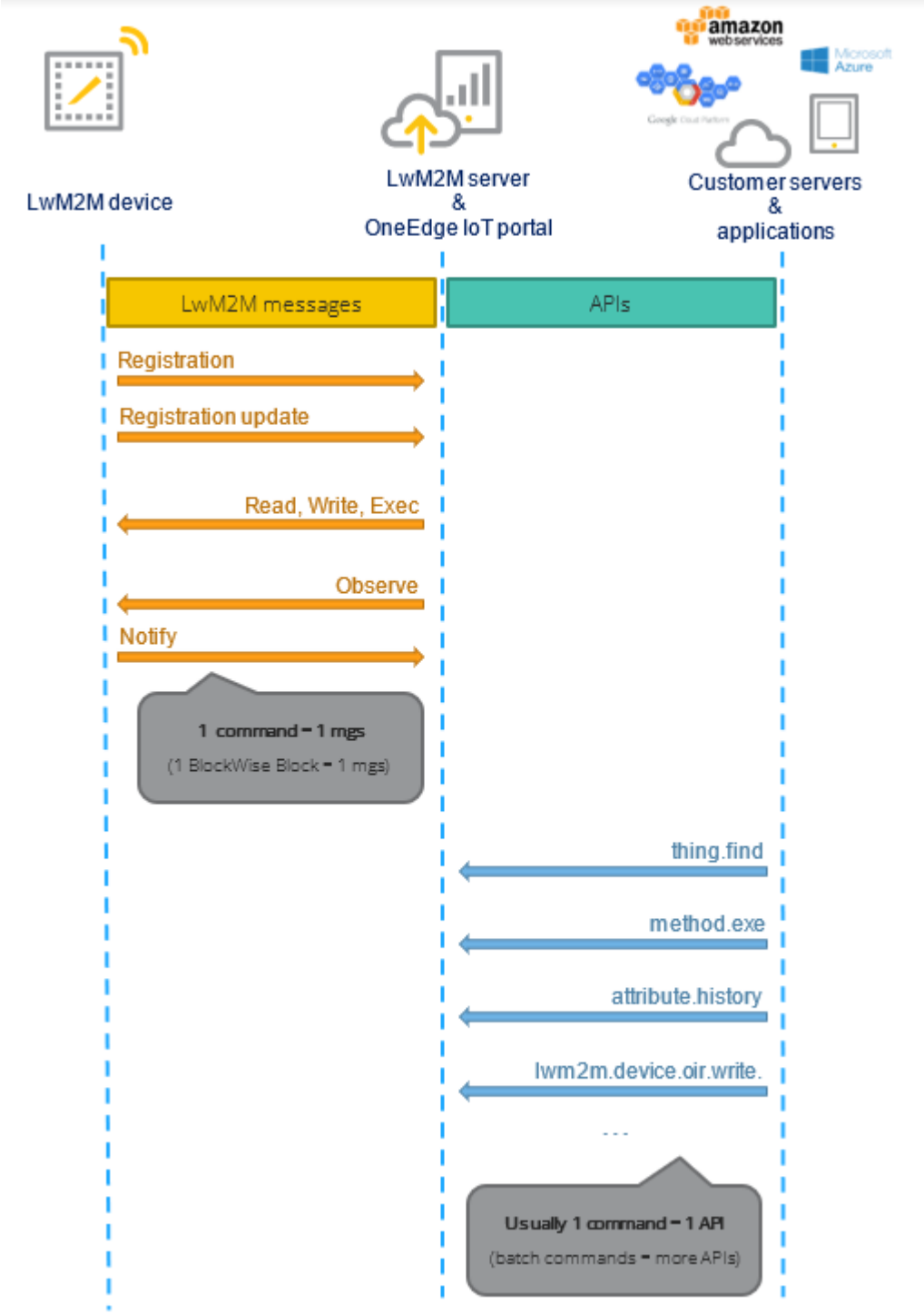



**Figure 84: Comparison between LwM2M messages and APIs on OneEdge IoT portal billing**

**Reference:**
https://docs.devicewise.com/Content/Products/IoT_Portal_API_Reference_Guide/IoT-Portal-API-Reference-Guide.htm

https://docs.devicewise.com/Content/Products/IoT_Portal_API_Reference_Guide/Understanding-API-Weights.htm

The number of **LwM2M messages** and **APIs** are pooled between the devices in the organization.

This means that if a device consumes 1500 LwM2M messages in a specific month and another device consumes 2300 in the same month, the average is 1900 LwM2M

messages, so, if the message allowance is for example 2000 messages per device, no additional cost is charged.

> **Note:** OneEdge users must always verify and discuss the terms of service, specifically for LwM2M messages and API thresholds.

A OneEdge user/developer should estimate the consumption of **LwM2M messages** and **APIs** according to its application constrains and requirements.

For example, below is a fast and rough estimation for a typical IoT application:

Modem always powered on and LwM2M client active all the time.

1 registration in a day: 30 LwM2M messages in a month

Device management, reading Connectivity object once a day.

Lifetime set to 1 day.
1 registration update in a day: 30 LwM2M messages in a month.
1 read object 4 in a day: 30 LwM2M messages in a month.

1 object data sent from the device to the portal every hour through observation/notify.

Observation linked to registration: 60 LwM2M messages in a month.
24 notifies in a day: 720 LwM2M messages in a month.

A script running on an external cloud that retrieves the data from an attribute of the portal every 4 hours.

6 TR50 REST commands in a day: 180 APIs in a month.

A user changes an application parameter (for example sampling frequency) from the smartphone application once a week

1 TR50 REST commands in a week: 4 APIs in a month.
1 LwM2M write command in a week: 4 LwM2M messages in a month.

At the end of the month, the device consumes 874 LwM2M messages and 184 APIs in a month.

> **Note: OneEdge:** OneEdge technical support team is available to provide application overview and the best application strategies to optimize LwM2M messages and API consumption.

The OneEdge user can monitor the LwM2M messages and APIs consumption on each thing on the **LwM2M usage** and **API usage** tabs on OneEdge IoT portal, as shown on the Figure 85 and 86 below.

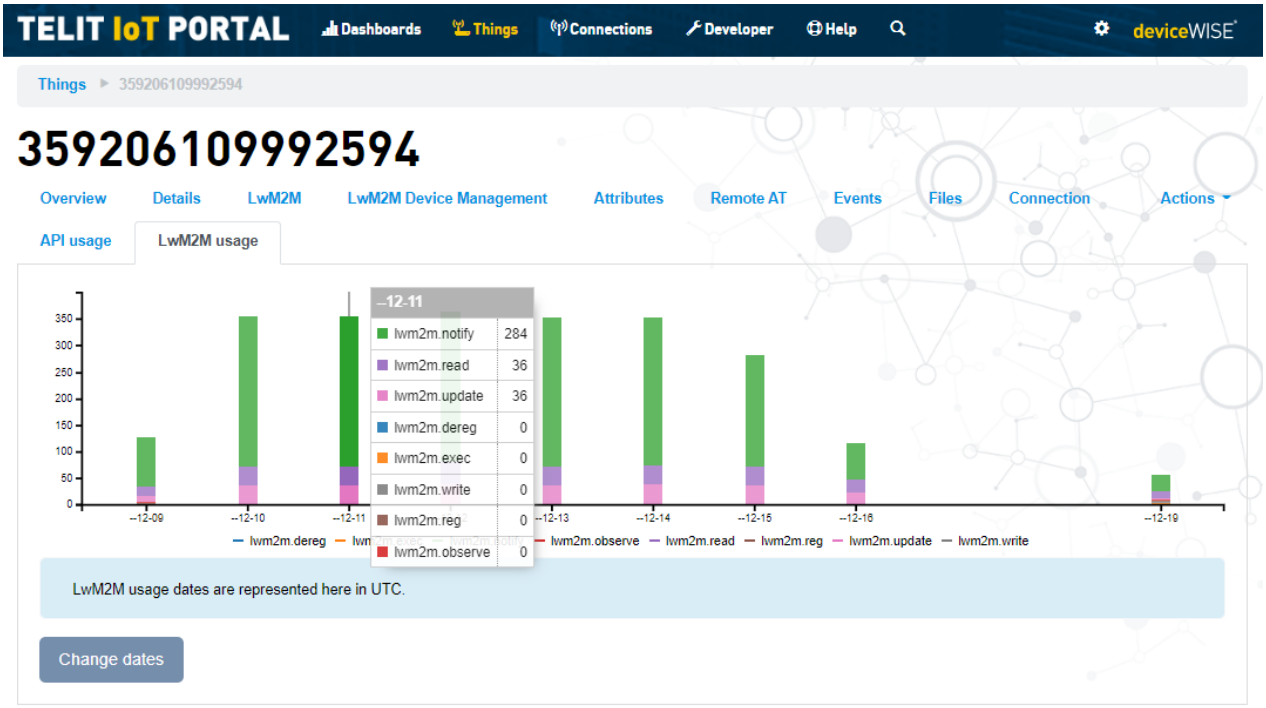



Figure 85: "LwM2M usage" tab on a specific thing

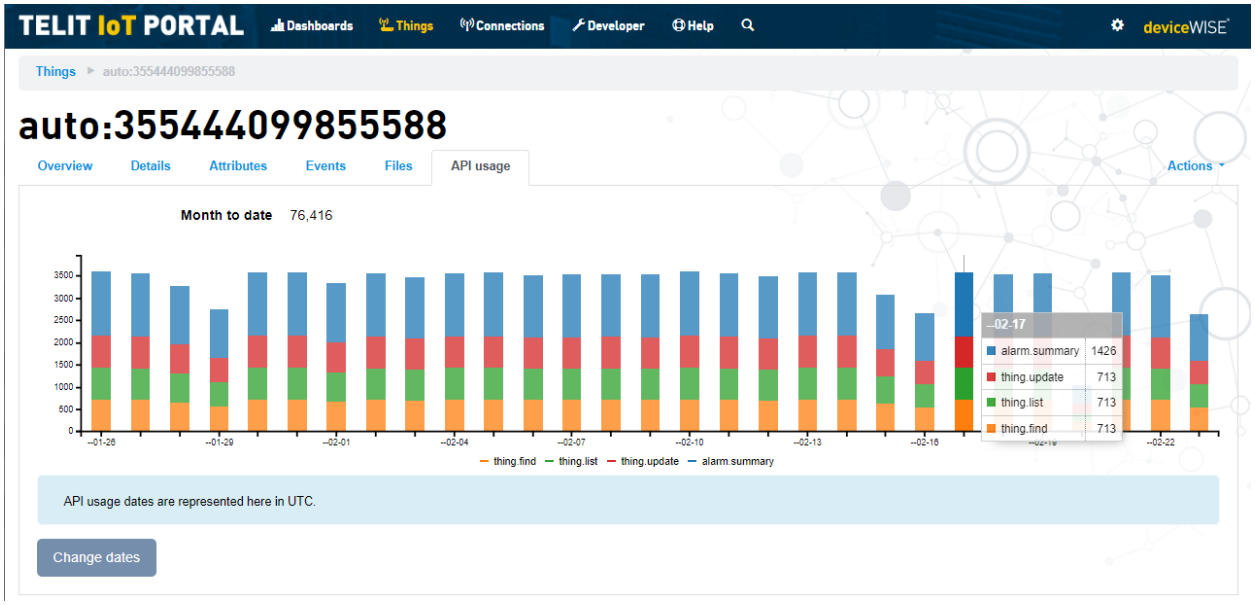



Figure 86: "API usage" tab for a specific external application

The OneEdge user can monitor also the overall consumption of all the things on the organization on the **Administration** section of the OneEdge IoT portal, as shown in the Figure 87 below.

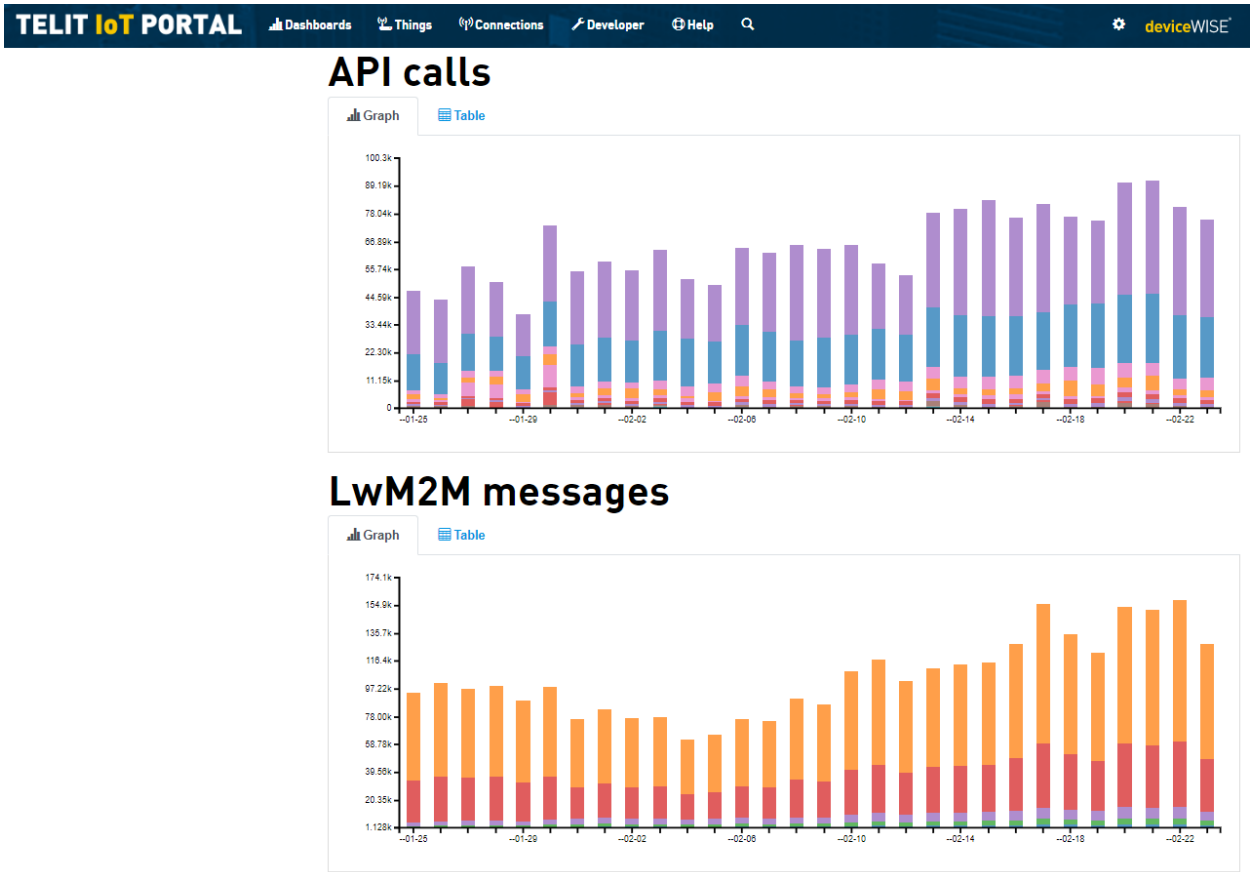
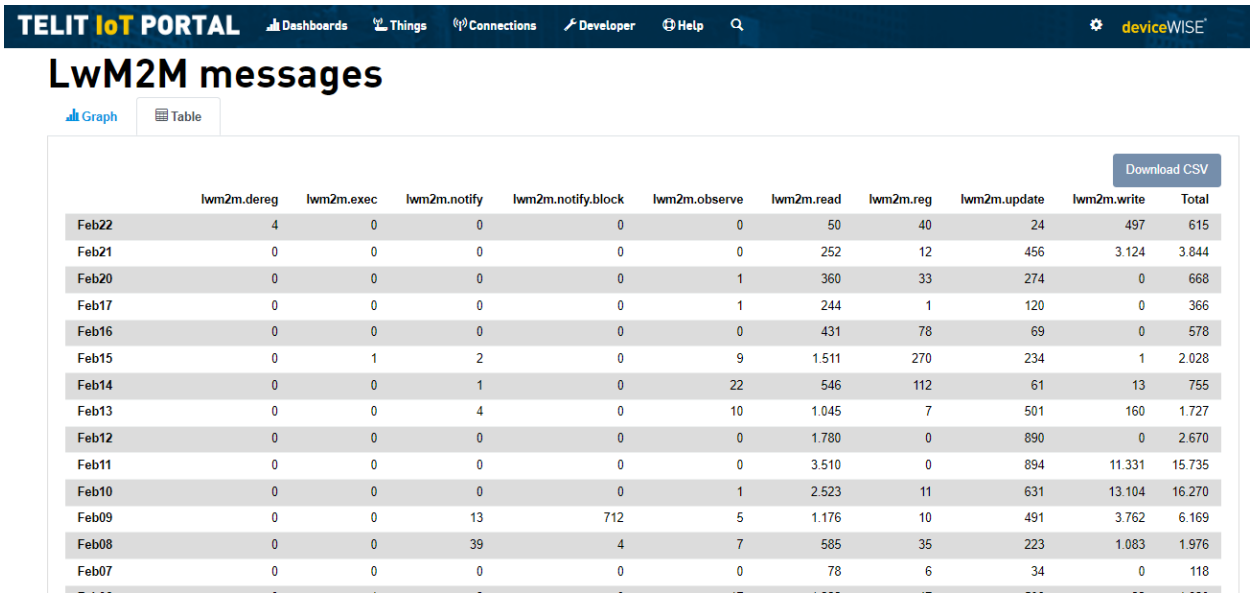



Figure 87: consumption charts on "Administration" section

**Note:** The Table tab provides the single counters of all LwM2M messages and API involved by all the things on the organization listed day by day, as showed in Figure 88.




| | lwm2m.dereg | lwm2m.exec | lwm2m.notify | lwm2m.notify.block | lwm2m.observe | lwm2m.read | lwm2m.reg | lwm2m.update | lwm2m.write | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| Feb22 | 4 | 0 | 0 | 0 | 0 | 50 | 40 | 24 | 497 | 615 |
| Feb21 | 0 | 0 | 0 | 0 | 0 | 252 | 12 | 456 | 3.124 | 3.844 |
| Feb20 | 0 | 0 | 0 | 0 | 1 | 360 | 33 | 274 | 0 | 668 |
| Feb17 | 0 | 0 | 0 | 0 | 1 | 244 | 1 | 120 | 0 | 366 |
| Feb16 | 0 | 0 | 0 | 0 | 0 | 431 | 78 | 69 | 0 | 578 |
| Feb15 | 0 | 1 | 2 | 0 | 9 | 1.511 | 270 | 234 | 1 | 2.028 |
| Feb14 | 0 | 0 | 1 | 0 | 22 | 546 | 112 | 61 | 13 | 755 |
| Feb13 | 0 | 0 | 4 | 0 | 10 | 1.045 | 7 | 501 | 160 | 1.727 |
| Feb12 | 0 | 0 | 0 | 0 | 0 | 1.780 | 0 | 890 | 0 | 2.670 |
| Feb11 | 0 | 0 | 0 | 0 | 0 | 3.510 | 0 | 894 | 11.331 | 15.735 |
| Feb10 | 0 | 0 | 0 | 0 | 1 | 2.523 | 11 | 631 | 13.104 | 16.270 |
| Feb09 | 0 | 0 | 13 | 712 | 5 | 1.176 | 10 | 491 | 3.762 | 6.169 |
| Feb08 | 0 | 0 | 39 | 4 | 7 | 585 | 35 | 223 | 1.083 | 1.976 |
| Feb07 | 0 | 0 | 0 | 0 | 0 | 78 | 6 | 34 | 0 | 118 |

Figure 88: LwM2M messages counters showed as table view.

Reference:
https://docs.devicewise.com/Content/Products/Management_Portal/Administration/Administration.htm

**Danger:** Enabling the "**live updates**" feature on "**My Profile**" section, the OneEdge portal performs an automatic refresh of the web interface, so some APIs are recalled periodically.

This feature is useful during OneEdge learning because provides a "**real-time**" effect to the OneEdge portal users.

Anyway, it's very dangerous on production environment because it can consume an unexpected amount of APIs.

In the Figure 89 below, the check boxes to enable and disable this feature are showed.

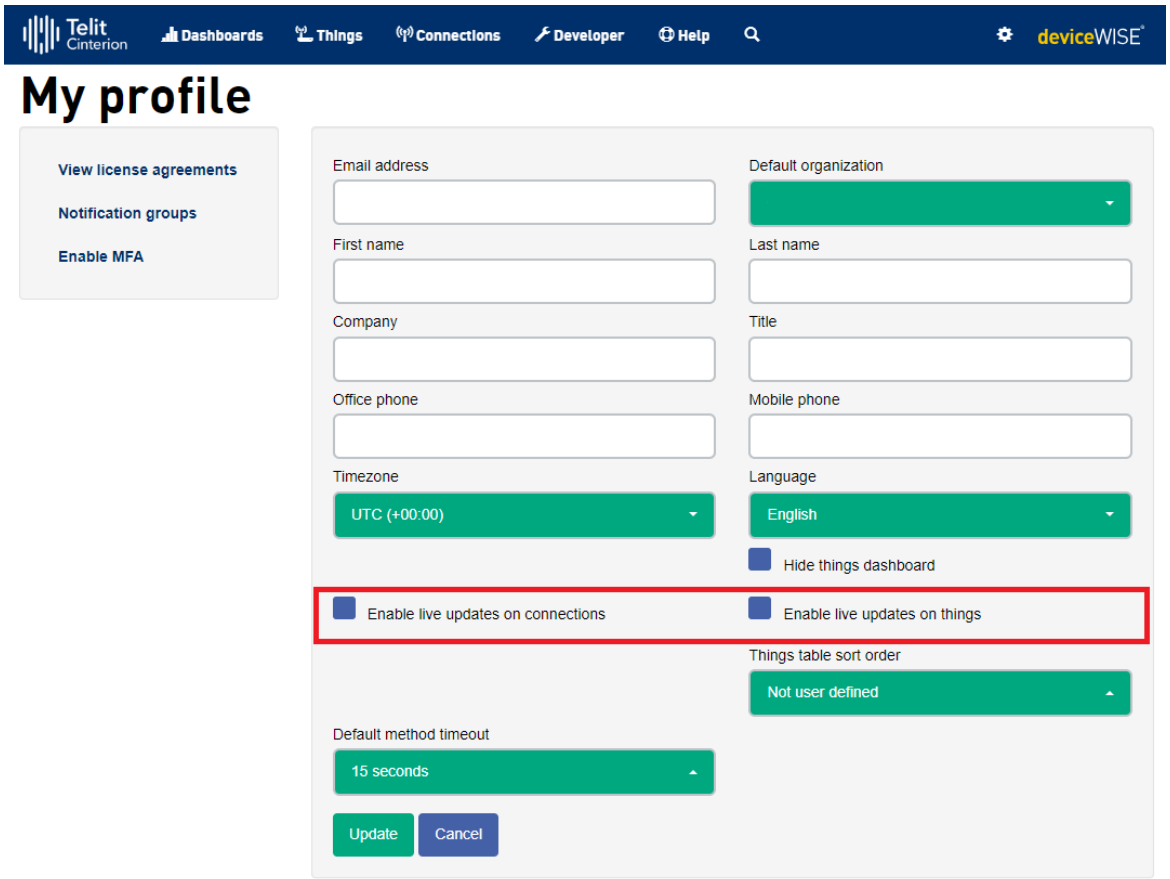



Figure 89: check boxes to enable and disable the "live updates" feature on Connections and Things

**Note:** The "**live updates**" check boxes are unchecked as default.

Reference:
https://docs.devicewise.com/Content/Products/Management_Portal/Administration/Users/Updating-user-profile.htm

# 14 OneEdge Application Guidelines

This section provides some guidelines and tips to develop a OneEdge application.

A good starting point for a OneEdge application is to define the data model to be mapped into the LwM2M objects and the following guidelines can be useful.

## 14.1 Check-list for Designing the OneEdge Application

Refer to the following steps:

1. What **data** has to be sent **from the device to the portal**? How often?

   An IPSO or custom object is needed: refer to LwM2M Objects in OneEdge

   If the host application stores data to be sent all together at periodic and scheduled time, refer to "**OneEdge Time-series**" application note.

2. What **data** has to be sent **from the portal to the device**? How often?

   An IPSO or custom object is needed: refer to chapter 9.

   New parameters for the host application can be sent through **trigger or campaign** (refer to paragraphs *Triggers* and *Campaigns*) or through **TR50 REST API** interface (refer to paragraph TR50 and REST API interfaces).

3. Which **attributes, alarms and properties** have to be published on the portal?

   Design the thing definition and configure the bindings: refer to paragraph Thing Definition and Bindings

4. Is the device battery or main powered?

   If battery-powered, the device is usually switched off most of the time and powered on just to send the data periodically of when an event occurs; also, other strategies can be used to save power like PSM, eDRX and CFUN5. Request for review and battery life estimation to OneEdge Technical Support team.

5. Which value for **lifetime**?

   If battery-powered, a good practice is a value of lifetime greater than the time between two consecutive wake ups.

   If main powered, a typical value can be one day.

   In both the cases, refer to considerations in chapter LWM2M Registration Interface.

   Lifetime must be configured on Device profile: refer to paragraph *Device Profile*

6. How is the data transmitted from the device to the portal?

   The **observation-notify mechanism** is usually used; alternatively, the server can read a resource/object from Device Profile or triggers. Refer to chapter section Management Portal Application and chapter Typical OneEdge Applications.

7 When is the data transmitted from the portal to the device?

Write and exec LwM2M commands can be used.

If the device switched on and the LwM2M client is registered to the server with a lifetime not expired, the LwM2M client could receive or not the incoming data.

If the client can receive the incoming LwM2M messages, it will receive them immediately.

If the client cannot receive the incoming LwM2M messages (for example due to the NAT), the messages are stored on the server queue and delivered at the next registration event, with a maximum delay equal to the lifetime.

If the application requirements do not accept a delay, so immediate delivery is requested, OneEdge Direct should be used. For more details about these concepts, refer to section OneEdge Direct

8 Has the **data to be processed** on portal side or **forwarded** to external cloud?

A **trigger** should be developed, and generic HTTP and specific AWS/Azure trigger connectors are available. Refer to Section Management Portal Application

9 Is the feature "**Device Management**" used?

Configure the relevant objects, typically Connectivity Monitoring and Statistics objects, as "**Read on update**" in the Device profile: refer to paragraph *Reads*.

Lifetime should be configured according to the frequency of the readings: refer to paragraph DTLS Protocol

**Note:** In any case, OneEdge technical support is available to analyse the customer requirements and developer feedbacks to provide the best strategy and most optimized data model on OneEdge environment.

## 14.2 Use Case Examples

1 Does the host application need to manage the upgrade of the modem?

Refer to "OneEdge **Firmware Management** app" application note.

2 Does the user need to manage the **upgrade** of the **host application** code or the **AppZone** binary?

Request for OneEdge Technical Support for the best solution.

3 Does the host application need to save the data and send the historical samples at periodic time to the portal?

Refer to "**OneEdge Time-Series**" application note.

4 Does the host application need to upload or download a generic file from an FTP/HTTP storage?

Refer to "**OneEdge File Transfer**" application note.

5 Does the host application need to manage the device location to perform GNSS tracking or positioning?

Refer to "**OneEdge GNSS management**" application note.

## 14.3 Good Practices

1 The host application of the devices in the field should be developed to manage **FOTA** processes, both modem FW and host application/AppZone code.

2 The host application of the devices in the field should be developed to **periodically check** both the **cellular and the LwM2M registration status** in order to guarantee that the device is able to **receive incoming data**.

3 The host application of the devices in the field should be developed to **check** both the **cellular and the LwM2M registration status** before to send outcoming data in order to guarantee that the device is able to send data.

4 In order to improve the robustness of the communications, it is strongly suggested to enable the "**DTLS retransmission**" feature on device side through the command AT#LWM2MCFG=0,9,0,1. Refer to paragraph Additional LwM2M configuration.

5 The host application of the devices in the field could be developed to store **debug messages** about the activities of the application on some files on the FileSystem or some LwM2M resources: these info could be retrieved to analyse remotely unexpected behaviours of the devices in the field. Furthermore, a log file on the modem File System could be uploaded through the FileTransfer concept.

6 When the design is completed, the application developer should **estimate** the API and LwM2M messages needed for the application: refer to chapter 12. OneEdge Technical Support team is available to review the OneEdge design looking for any optimization.

7 The application developers should disable "**live updates**" feature on production environments: refer to chapter OneEdge Counters

8 The application developers should **avoid synchronizing the actions to specific times**, for example device waking up and connecting to the server at the same time.

9 The OneEdge portal Users must not onboard the things by theirself but should find the things already onboarded according to the ordered devices: refer to "OneEdge zero-touch onboarding" feature on Management Portal Application section. If some things are missing on the Thing list, Request for OneEdge Technical support team.

# 15 Acronyms and Abbreviations

Table 27: Acronyms and Abbreviations

| Acronym | Definition |
|---|---|
| ADC | Analog – Digital Converter |
| CLK | Clock |
| CMOS | Complementary Metal – Oxide Semiconductor |
| CS | Chip Select |
| DAC | Digital – Analog Converter |
| DTE | Data Terminal Equipment |
| ESR | Equivalent Series Resistance |
| GPIO | General Purpose Input Output |
| HS | High Speed |
| HSDPA | High-Speed Downlink Packet Access |
| HSIC | High-Speed Inter Chip |
| HSUPA | High-Speed Uplink Packet Access |
| I/O | Input Output |
| MISO | Master Input – Slave Output |
| MOSI | Master Output – Slave Input |
| MRDY | Master Ready |
| PCB | Printed Circuit Board |
| RTC | Real-Time Clock |
| SIM | Subscriber Identification Module |
| SPI | Serial Peripheral Interface |
| SRDY | Slave Ready |
| TTSC | Telit Technical Support Centre |
| UART | Universal Asynchronous Receiver Transmitter |
| UMTS | Universal Mobile Telecommunication System |
| USB | Universal Serial Bus |
| VNA | Vector Network Analyzer |
| VSWR | Voltage Standing Wave Radio |
| WCDMA | Wideband Code Division Multiple Access |

# 16 Related Documents

Refer to https://dz.telit.com/ for current documentation and downloads.

**Table 28: Related Documents**

| S.no | Book Code | Document Title |
|---|---|---|
| 1 | 80529ST10974A | ME910C1 LwM2M AT commands Reference Guide |
| 2 | 80617ST11022A | MEx10G1 LwM2M AT commands Reference Guide |
| 3 | 80646ST11059A | LE910Cx LwM2M AT commands Reference Guide |
| 4 | 80000NT11696A | 2G 3G 4G Registration Process |
| 5 | 1VV0301586 | Connection Agent User Guide |
| 6 | V1_0_2-20180209-A | OMA Lightweight M2M Specification |
| 7 | 1vv0301585 | OneEdge Getting Started Guide |
| 8 | 80654NT11889A | OneEdge Firmware Management Application Note |
| 9 | 80654NT11973A | OneEdge FileTransfer Application Note |
| 10 | 80654NT11932A | OneEdge Time-series Application Note |
| 11 | 80654NT11966A | OneEdge GNSS Tracking through OneEdge |

# 17 Document History

Table 29: Document History

| Revision | Date | Changes |
|---|---|---|
| 0 | 2023-10-10 | First Issue |

From Mod.0818 Rev.0

© Telit 2023 All rights reserved.

Technical Documentation