AppZone m2mb Sample Apps

80000NT11840A Rev. 1 - 2021-01-29

TELIT
TECHNICAL
DOCUMENTATION

1 AppZone m2mb Sample Apps

Package Version: **1.1.19-G1**

Minimum Firmware Version: **37.00.004.1**

1.1 Features

This package goal is to provide sample source code for common activities kick-start.

2 Quick start

2.1 Deployment Instructions

To manually deploy the Sample application on the devices perform the following steps:

1. Have **37.00.004.1** FW version flashed (`AT#SWPKGV` will give you the FW version)
2. Copy m2mapz.bin to /mod/

   ```
   AT#M2MWRITE="/mod/m2mapz.bin",<size>,1
   ```

   where <size> is in bytes
3. Configure the module to run the downloaded binary as default app: `AT#M2MRUN=2,m2mapz.bin`
4. Restart the module and if no AT commands are sent within **10** seconds, start the app: `AT+M2M=4,10`

2.2 References

More info on

- Getting started with ME910C1 (doc ID 80529NT11661A)
- How to run applications with AppZone

2.3 Known Issues

None

2.4 Contact Information, Support

For general contact, technical support services, technical questions and report documentation errors contact Telit Technical Support at: TS-EMEA@telit.com.

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

http://www.telit.com

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

Telit appreciates feedback from the users of our information.

2.5 Troubleshooting

- Application does not work/start:
  - Delete application binary and retry

    ```
    AT#M2MDEL="/mod/m2mapz.bin"
    ```

  - Delete everything, reflash and retry

    ```
    AT#M2MDEL="/mod/m2mapz.bin"
    AT#M2MDEL="/mod/appcfg.ini"
    ```

- Application project does not compile
  - Right click on project name
  - Select Properties
  - Select AppZone tab
  - Select the right plugin (firmware) version
  - Press “Restore Defaults”, then “Apply”, then “OK”
  - Build project again
- Application project shows missing symbols on IDE
  - Right click on project name
  - Select Index
  - Select Rebuild. This will regenerate the symbols index.

2.6 Making source code changes

2.6.1 Folder structure

The applications code follow the structure below:

- `hdr`: header files used by the application
  - `app_cfg.h`: the main configuration file for the application
- `src`: source code specific to the application
- `azx`: helpful utilities used by the application (for GPIOs, LOGGING etc)
  - `hdr`: generic utilities' header files
  - `src`: generic utilities' source files
- `Makefile.in`: customization of the Make process

2.7 Import a Sample App into an IDE project

Consider that the app HelloWorld that prints on Main UART is a good starting point. To import it in a project, please follow the steps below:

On IDE, create a new project: "File"-> "New" -> "Telit Project"

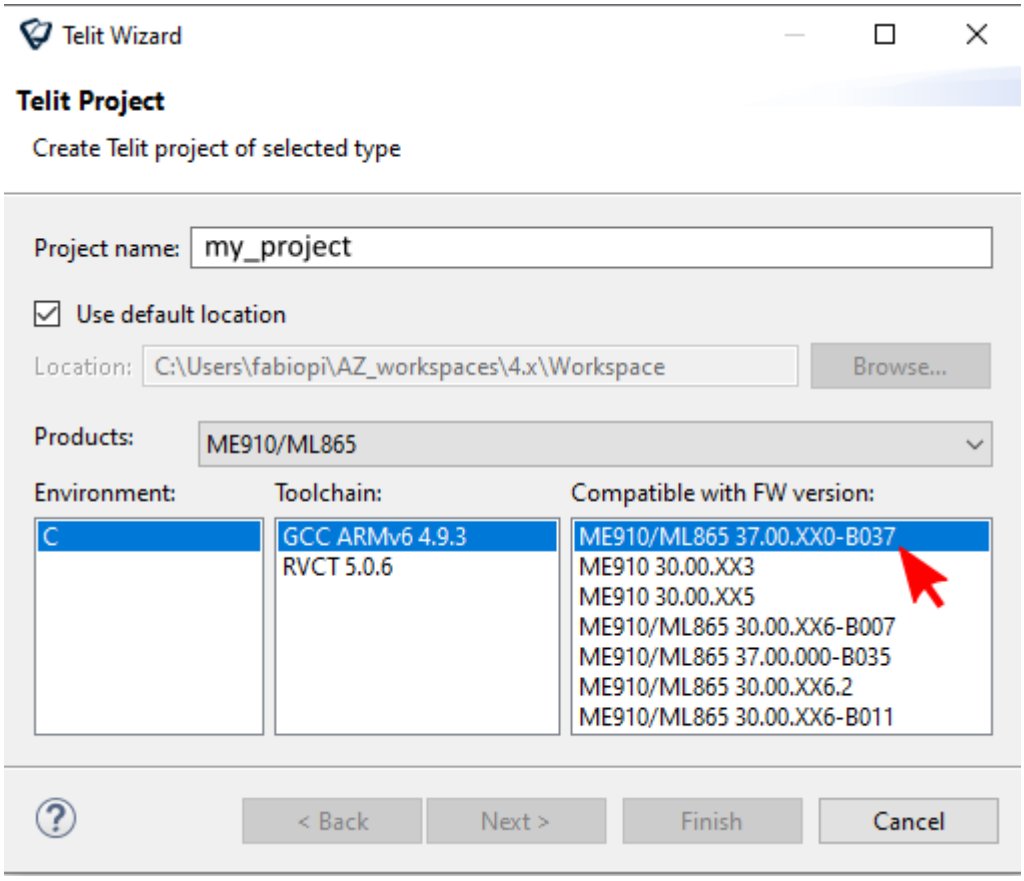

Figure 1

Select the preferred firmware version (e.g. 30.00.xx7) and create an empty project.

in the samples package, go in the HelloWorld folder (e.g. `AppZoneSampleApps-MAIN_UART\HelloWorld`), copy all the files and folders in it (as `src`, `hdr`, `azx`) and paste them in the root of the newly created IDE project. You are now ready tyo build and try the sample app on your device.

2.8 Heap and starting address

By default, every application defines a memory HEAP size and its start address in memory.

They are usually provided by the linking phase of the build process:

```
"[...]arm-none-eabi-ld" --defsym __ROM=0x40000000 --defsym __HEAP_PUB_SIZE=0x40000 -
-defsym
```

`__ROM` is the default starting address, `__HEAP_PUB_SIZE` is the default HEAP size in bytes. Both are expressed in hexadecimal format.

These values can be customized through makefile variables:

```
HEAP=<new size in bytes>

ROM_START=<new address>
```

IMPORTANT allowed address range is 0x40000000 - 0x4FFFF000.

Contents

1 AppZone m2mb Sample Apps **2**
1.1 Features 2

2 Quick start **2**
2.1 Deployment Instructions 2
2.2 References 2
2.3 Known Issues 2
2.4 Contact Information, Support 3
2.5 Troubleshooting 3
2.6 Making source code changes 4
2.6.1 Folder structure 4
2.7 Import a Sample App into an IDE project 4
2.8 Heap and starting address 5

3 Applications **17**
3.1 MISC 17
3.1.1 GPIO toggle example 17
3.2 BASIC 18
3.2.1 Basic Hello World (Aux UART) 18
3.2.1.1 Application workflow 18
3.2.2 Basic Hello World (Main UART) 19
3.2.2.1 Application workflow 19
3.2.3 Basic Hello World (USB0) 20
3.2.3.1 Application workflow 20
3.2.4 Basic Task 21
3.2.4.1 Application workflow 21
3.2.5 UART USB tunnel example 22
3.2.5.1 Application workflow 22
3.3 C++ 23
3.3.1 Logging C++ 23
3.3.1.1 Application workflow 23
3.3.2 C++ method to function pointer 24
3.3.2.1 Application workflow 24
3.4 AUX UART 25
3.4.1 ATI (AT Instance) 25
3.4.1.1 Application workflow, sync mode 25
3.4.1.2 Application workflow, async mode 26
3.4.2 AWS demo 27
3.4.2.1 Application workflow 27
3.4.2.2 How to get started with AWS IoT 28

3.4.2.3 Application setup 29
3.4.2.4 Device setup 29
3.4.3 App Manager 31
3.4.3.1 Prerequisites 31
3.4.3.2 Application workflow 31
3.4.4 App update OTA via FTP 33
3.4.4.1 Application workflow 33
3.4.5 CJSON example: 35
3.4.5.1 Application workflow 35
3.4.6 Crypto Elliptic Curve Cryptography (ECC) example 37
3.4.6.1 Application workflow 37
3.4.7 EEPROM 24AA256 40
3.4.7.1 Application workflow 40
3.4.8 Easy AT example 42
3.4.8.1 AT#MYCMD 42
3.4.8.2 AT#MYINPUT 42
3.4.9 Events 43
3.4.9.1 Application workflow 43
3.4.10 Events - Barrier (multi events) 44
3.4.10.1 Application workflow 44
3.4.11 FOTA example 45
3.4.11.1 Application workflow 45
3.4.12 FOTA from Local File example 47
3.4.13 FTP 49
3.4.13.1 Application workflow 49
3.4.14 File System example 51
3.4.14.1 Application workflow 51
3.4.15 GNSS example 52
3.4.15.1 Application workflow 52
3.4.16 GPIO interrupt example 53
3.4.16.1 Application workflow 53
3.4.17 GTP example 54
3.4.18 General_INFO example 55
3.4.18.1 Application workflow 55
3.4.19 HTTP Client 57
3.4.19.1 Application workflow 57
3.4.20 HW Timer (Hardware Timer) 59
3.4.20.1 Application workflow 59
3.4.21 Hello World 60
3.4.21.1 Application workflow 60
3.4.22 I2C example 61

3.4.22.1 Application workflow 61
3.4.23I2C Combined 63
3.4.23.1 Application workflow 63
3.4.24LWM2M 65
3.4.24.1 Application workflow 65
3.4.24.2 Device Profile upload 66
3.4.24.3 Custom Object configuration 66
3.4.24.4 Onboard the device 70
3.4.24.5 Application execution example 72
3.4.25LWM2M FOTA ACK management (AT URCs) 82
3.4.25.1 Device Profile upload 82
3.4.25.2 Onboard the device 83
3.4.25.3 Application workflow 85
3.4.25.4 Application execution example 85
3.4.26LWM2M OBJ_GET AND OBJ_SET 87
3.4.26.1 Application workflow 87
3.4.26.2 Device Profile upload 88
3.4.26.3 Custom Object configuration 88
3.4.26.4 Onboard the device 92
3.4.26.5 Application execution example 94
3.4.27Logging Demo 98
3.4.27.1 Application workflow 98
3.4.28Low power mode 99
3.4.29MD5 example 100
3.4.29.1 Application workflow 100
3.4.30MQTT Client 101
3.4.30.1 Application workflow 101
3.4.31MultiTask 103
3.4.31.1 Application workflow 103
3.4.32MutEx 105
3.4.32.1 Application workflow 105
3.4.33NTP example 108
3.4.33.1 Application workflow 108
3.4.34RTC example 109
3.4.34.1 Application workflow 109
3.4.35SIM event handler example 110
3.4.36SMS PDU 111
3.4.36.1 Application workflow 111
3.4.37SMS_atCmd example 112
3.4.37.1 Application workflow 112
3.4.38SMTP Client 114

3.4.39 SW Timer (Software Timer) 116
3.4.39.1 Application workflow 116
3.4.40 Secure MicroService 117
3.4.40.1 Application workflow 117
3.4.41 TCP IP 119
3.4.41.1 Application workflow 119
3.4.42 TCP Socket status 121
3.4.42.1 Application workflow 121
3.4.43 TCP Server 123
3.4.43.1 Application workflow 123
3.4.44 TLS SSL Client 126
3.4.44.1 Application workflow 126
3.4.45 UDP client 129
3.4.45.1 Application workflow 129
3.4.46 USB Cable Check 130
3.4.46.1 Application workflow 130
3.4.47 Watchdog example 131
3.4.47.1 Application workflow 131
3.4.48 ZLIB example 132
3.4.48.1 Application workflow 132
3.5 USB0 133
3.5.1 ATI (AT Instance) 133
3.5.1.1 Application workflow, sync mode 133
3.5.1.2 Application workflow, async mode 134
3.5.2 AWS demo 135
3.5.2.1 Application workflow 135
3.5.2.2 How to get started with AWS IoT 136
3.5.2.3 Application setup 137
3.5.2.4 Device setup 137
3.5.3 App Manager 139
3.5.3.1 Prerequisites 139
3.5.3.2 Application workflow 139
3.5.4 App update OTA via FTP 141
3.5.4.1 Application workflow 141
3.5.5 CJSON example: 143
3.5.5.1 Application workflow 143
3.5.6 Crypto Elliptic Curve Cryptography (ECC) example 145
3.5.6.1 Application workflow 145
3.5.7 EEPROM 24AA256 148
3.5.7.1 Application workflow 148
3.5.8 Easy AT example 150

3.5.8.1 AT#MYCMD 150
3.5.8.2 AT#MYINPUT 150
3.5.9 Events 151
3.5.9.1 Application workflow 151
3.5.10 Events - Barrier (multi events) 152
3.5.10.1 Application workflow 152
3.5.11 FOTA example 153
3.5.11.1 Application workflow 153
3.5.12 FOTA from Local File example 155
3.5.13 FTP 157
3.5.13.1 Application workflow 157
3.5.14 File System example 159
3.5.14.1 Application workflow 159
3.5.15 GNSS example 160
3.5.15.1 Application workflow 160
3.5.16 GPIO interrupt example 161
3.5.16.1 Application workflow 161
3.5.17 GTP example 162
3.5.18 General_INFO example 163
3.5.18.1 Application workflow 163
3.5.19 HTTP Client 165
3.5.19.1 Application workflow 165
3.5.20 HW Timer (Hardware Timer) 167
3.5.20.1 Application workflow 167
3.5.21 Hello World 168
3.5.21.1 Application workflow 168
3.5.22 I2C example 169
3.5.22.1 Application workflow 169
3.5.23 I2C Combined 171
3.5.23.1 Application workflow 171
3.5.24 Little FileSystem 2 173
3.5.24.1 Application workflow 173
3.5.25 LWM2M 176
3.5.25.1 Application workflow 176
3.5.25.2 Device Profile upload 177
3.5.25.3 Custom Object configuration 177
3.5.25.4 Onboard the device 181
3.5.25.5 Application execution example 183
3.5.26 LWM2M FOTA ACK management (AT URCs) 193
3.5.26.1 Device Profile upload 193
3.5.26.2 Onboard the device 194

3.5.26.3 Application workflow 196
3.5.26.4 Application execution example 196
3.5.27 LWM2M OBJ_GET AND OBJ_SET 198
3.5.27.1 Application workflow 198
3.5.27.2 Device Profile upload 199
3.5.27.3 Custom Object configuration 199
3.5.27.4 Onboard the device 203
3.5.27.5 Application execution example 205
3.5.28 Logging Demo 209
3.5.28.1 Application workflow 209
3.5.29 MD5 example 210
3.5.29.1 Application workflow 210
3.5.30 MQTT Client 211
3.5.30.1 Application workflow 211
3.5.31 MultiTask 213
3.5.31.1 Application workflow 213
3.5.32 MutEx 215
3.5.32.1 Application workflow 215
3.5.33 NTP example 218
3.5.33.1 Application workflow 218
3.5.34 RTC example 219
3.5.34.1 Application workflow 219
3.5.35 SIM event handler example 220
3.5.36 SMS PDU 221
3.5.36.1 Application workflow 221
3.5.37 SMS_atCmd example 222
3.5.37.1 Application workflow 222
3.5.38 SMTP Client 224
3.5.39 SPI Echo 226
3.5.39.1 Application workflow 226
3.5.40 SPI sensors 227
3.5.40.1 Application workflow 227
3.5.41 SW Timer (Software Timer) 229
3.5.41.1 Application workflow 229
3.5.42 Secure MicroService 230
3.5.42.1 Application workflow 230
3.5.43 TCP IP 232
3.5.43.1 Application workflow 232
3.5.44 TCP non blocking example 234
3.5.45 TCP Socket status 236
3.5.45.1 Application workflow 236

3.5.46 TCP Server . . . 238
3.5.46.1 Application workflow . . . 238
3.5.47 TLS SSL Client . . . 241
3.5.47.1 Application workflow . . . 241
3.5.48 UDP client . . . 244
3.5.48.1 Application workflow . . . 244
3.5.49 UDP_Server example . . . 245
3.5.50 Watchdog example . . . 247
3.5.50.1 Application workflow . . . 247
3.5.51 ZLIB example . . . 248
3.5.51.1 Application workflow . . . 248
3.6 MAIN UART . . . 249
3.6.1 ATI (AT Instance) . . . 249
3.6.1.1 Application workflow, sync mode . . . 249
3.6.1.2 Application workflow, async mode . . . 250
3.6.2 AT Tunnel . . . 251
3.6.2.1 Application workflow . . . 251
3.6.3 AWS demo . . . 253
3.6.3.1 Application workflow . . . 253
3.6.3.2 How to get started with AWS IoT . . . 254
3.6.3.3 Application setup . . . 255
3.6.3.4 Device setup . . . 255
3.6.4 App Manager . . . 257
3.6.4.1 Prerequisites . . . 257
3.6.4.2 Application workflow . . . 257
3.6.5 App update OTA via FTP . . . 259
3.6.5.1 Application workflow . . . 259
3.6.6 CJSON example: . . . 261
3.6.6.1 Application workflow . . . 261
3.6.7 Crypto Elliptic Curve Cryptography (ECC) example . . . 263
3.6.7.1 Application workflow . . . 263
3.6.8 EEPROM 24AA256 . . . 266
3.6.8.1 Application workflow . . . 266
3.6.9 Easy AT example . . . 268
3.6.9.1 AT#MYCMD . . . 268
3.6.9.2 AT#MYINPUT . . . 268
3.6.10 Events . . . 269
3.6.10.1 Application workflow . . . 269
3.6.11 Events - Barrier (multi events) . . . 270
3.6.11.1 Application workflow . . . 270
3.6.12 FOTA example . . . 271

3.6.12.1 Application workflow 271
3.6.13FOTA from Local File example 273
3.6.14FTP 275
3.6.14.1 Application workflow 275
3.6.15File System example 277
3.6.15.1 Application workflow 277
3.6.16GNSS example 278
3.6.16.1 Application workflow 278
3.6.17GPIO interrupt example 279
3.6.17.1 Application workflow 279
3.6.18GTP example 280
3.6.19General_INFO example 281
3.6.19.1 Application workflow 281
3.6.20HTTP Client 283
3.6.20.1 Application workflow 283
3.6.21HW Timer (Hardware Timer) 285
3.6.21.1 Application workflow 285
3.6.22Hello World 286
3.6.22.1 Application workflow 286
3.6.23I2C example 287
3.6.23.1 Application workflow 287
3.6.24I2C Combined 289
3.6.24.1 Application workflow 289
3.6.25Little FileSystem 2 291
3.6.25.1 Application workflow 291
3.6.26LWM2M 294
3.6.26.1 Application workflow 294
3.6.26.2 Device Profile upload 295
3.6.26.3 Custom Object configuration 295
3.6.26.4 Onboard the device 299
3.6.26.5 Application execution example 301
3.6.27LWM2M FOTA ACK management (AT URCs) 311
3.6.27.1 Device Profile upload 311
3.6.27.2 Onboard the device 312
3.6.27.3 Application workflow 314
3.6.27.4 Application execution example 314
3.6.28LWM2M OBJ_GET AND OBJ_SET 316
3.6.28.1 Application workflow 316
3.6.28.2 Device Profile upload 317
3.6.28.3 Custom Object configuration 317
3.6.28.4 Onboard the device 321

3.6.28.5 Application execution example 323
3.6.29Logging Demo 327
3.6.29.1 Application workflow 327
3.6.30Low power mode 328
3.6.31MD5 example 329
3.6.31.1 Application workflow 329
3.6.32MQTT Client 330
3.6.32.1 Application workflow 330
3.6.33MultiTask 332
3.6.33.1 Application workflow 332
3.6.34MutEx 334
3.6.34.1 Application workflow 334
3.6.35NTP example 337
3.6.35.1 Application workflow 337
3.6.36RTC example 338
3.6.36.1 Application workflow 338
3.6.37SIM event handler example 339
3.6.38SMS PDU 340
3.6.38.1 Application workflow 340
3.6.39SMS_atCmd example 341
3.6.39.1 Application workflow 341
3.6.40SMTP Client 343
3.6.41SPI Echo 345
3.6.41.1 Application workflow 345
3.6.42SPI sensors 346
3.6.42.1 Application workflow 346
3.6.43SW Timer (Software Timer) 348
3.6.43.1 Application workflow 348
3.6.44Secure MicroService 349
3.6.44.1 Application workflow 349
3.6.45TCP IP 351
3.6.45.1 Application workflow 351
3.6.46TCP non blocking example 353
3.6.47TCP Socket status 355
3.6.47.1 Application workflow 355
3.6.48TCP Server 357
3.6.48.1 Application workflow 357
3.6.49TLS SSL Client 360
3.6.49.1 Application workflow 360
3.6.50Uart To Server 363
3.6.50.1 Application workflow 363

3.6.51 UDP client 364
3.6.51.1 Application workflow 364
3.6.52 UDP_Server example 365
3.6.53 USB Cable Check 367
3.6.53.1 Application workflow 367
3.6.54 Basic USB read/write example 368
3.6.54.1 Application workflow 368
3.6.55 Watchdog example 369
3.6.55.1 Application workflow 369
3.6.56 ZLIB example 370
3.6.56.1 Application workflow 370

4 Installing beta version libraries Plug-in 371
4.1 New beta plug-in installation 371
4.2 Change existing project libraries 374
4.3 Create a project with the new plug-in 375

3 Applications

3.1 MISC

Applications that provide usage examples for various functionalities, without prints

3.1.1 GPIO toggle example

Sample application showcasing GPIO usage with M2MB API

Features

- How to open a gpio in output mode and change its status

3.2 BASIC

Basic applications showing simple operations with minimum code overhead

3.2.1 Basic Hello World (Aux UART)

The application prints “Hello World!” on Auxiliary UART every 2 seconds using

Features

- How to open Auxiliary UART as an output channel
- How to print messages out of the channel

3.2.1.1 Application workflow

`M2MB_main.c`

- Open Auxiliary UART with **`m2mb_uart_open`** function
- write a welcome message using **`m2mb_uart_write`**
- write “Hello World!” every 2 seconds in a while loop, using **`m2mb_uart_write`**

```
Start Hello world Application  [ version: 2.000000 ]

Hello world 2.0 [ 000001 ]
Hello world 2.0 [ 000002 ]
Hello world 2.0 [ 000003 ]
Hello world 2.0 [ 000004 ]
Hello world 2.0 [ 000005 ]
Hello world 2.0 [ 000006 ]
Hello world 2.0 [ 000007 ]
Hello world 2.0 [ 000008 ]
Hello world 2.0 [ 000009 ]
```

Figure 2

3.2.2 Basic Hello World (Main UART)

The application prints “Hello World!” on Main UART every 2 seconds using

Features

- How to open Main UART as an output channel
- How to print messages out of the channel

3.2.2.1 Application workflow

M2MB_main.c

- Open Main UART with **m2mb_uart_open** function
- write a welcome message using **m2mb_uart_write**
- write “Hello World!” every 2 seconds in a while loop, using **m2mb_uart_write**

```
Start Hello world Application  [ version: 2.000000 ]

Hello world 2.0 [ 000001 ]
Hello world 2.0 [ 000002 ]
Hello world 2.0 [ 000003 ]
Hello world 2.0 [ 000004 ]
Hello world 2.0 [ 000005 ]
Hello world 2.0 [ 000006 ]
Hello world 2.0 [ 000007 ]
Hello world 2.0 [ 000008 ]
Hello world 2.0 [ 000009 ]
```

Figure 3

3.2.3 Basic Hello World (USB0)

The application prints “Hello World!” on USB 0 every 2 seconds using

Features

- How to open USB 0 as an output channel
- How to print messages out of the channel

3.2.3.1 Application workflow

M2MB_main.c

- Open USB 0 with **m2mb_usb_open** function
- write a welcome message using **m2mb_usb_write**
- write “Hello World!” every 2 seconds in a while loop, using **m2mb_usb_write**

```
Start Hello world Application  [ version: 2.000000 ]

Hello world 2.0 [ 000001 ]
Hello world 2.0 [ 000002 ]
Hello world 2.0 [ 000003 ]
Hello world 2.0 [ 000004 ]
Hello world 2.0 [ 000005 ]
Hello world 2.0 [ 000006 ]
Hello world 2.0 [ 000007 ]
Hello world 2.0 [ 000008 ]
Hello world 2.0 [ 000009 ]
```

Figure 4

3.2.4 Basic Task

The application shows how to create and manage tasks with m2mb APIs. Debug prints on MAIN UART (can be changed in M2MB_Main function)

Features

- How to create a new task using m2mb APIs
- How to start the task and send messages to it
- how to destroy the task

3.2.4.1 Application workflow

`M2MB_main.c`

- Open UART
- Print welcome message
- Configure and create message queue for task
- Configure and create task
- Send 2 messages to the task queue

`task_entry_function`

- Receive messages from the task queue in a loop
- Print the message data when one arrives

```
Starting Basic Task demo app. This is v1.0.8 built on Apr 16 2020 06:40:40.
Successfully created a queue area buffer of 720 bytes.
Queue successfully created.
Creating the task...
Task created and ready to receive messages!
[DEBUG] 16.88  M2MB_main:411 - M2MB_main{M2M_DamsStart}$ Sending a message to the task...
[DEBUG] 16.88  M2MB_main:125 - task_entry_function{mytask}$ Received a message with a 5 bytes payload: <hello>
[DEBUG] 18.90  M2MB_main:420 - M2MB_main{M2M_DamsStart}$ Sending a second message to the task...
[DEBUG] 18.90  M2MB_main:430 - M2MB_main{M2M_DamsStart}$ Result code at the end: 0
[DEBUG] 18.91  M2MB_main:125 - task_entry_function{mytask}$ Received a message with a 5 bytes payload: <world>
Clearing resources...
Done. App complete
```

Figure 5

3.2.5 UART USB tunnel example

Sample application that opens a tunnel between main UART and USB0 port.

Features

- Opens `Main UART` port with a callback function
- Opens `USB0` port with a callback function
- Creates a simple task to manage data exchange between ports

3.2.5.1 Application workflow

`M2MB_main function`

- Create `Main UART` handle and configure its parameters
- Create `USB0` handle and configure its parameters
- Create the data management task
- Write **`READY`** on both ports when the tunneling is ready

`USB_Cb`

- When data are received on the `USB0` port, retrieve the available amount and send the value to the data management task with the proper command

`UART_Cb`

- When data are received on the `Main UART` port, retrieve the available amount and send the value to the data management task with the proper command

`dataTask_Cb`

- if command is `TASK_UART_READ_AND_USB_WRITE`, read the requested amount from the `Main UART` port and write it on `USB0`
- if command is `TASK_USB_READ_AND_UART_WRITE`, read the requested amount from the `USB0` port and write it on `Main UART`

UART output received from USB0 (in RED, the user input data from UART)

```
READYHello from UART
Hello from USB0
```

USB0 output received from UART (in RED, the user input data from USB0)

```
READYHello from UART
Hello from USB0
```

3.3 C++

Applications that provide usage examples with C++

3.3.1 Logging C++

Sample application showcasing how to create a C++ OO code, providing a logging class (equivalent to the one in Logging demo)

Features

- how to define a class object
- how to instantiate and call the class from a C++ main
- how to configure makefile flags to build the application

3.3.1.1 Application workflow

`M2MB_main.c`

- Call C++ main function

`main.cpp`

- Create a Logger class instance and set it to USB/UART/UART_AUX
- Print one message for every log level

```
[TRACE] 15.20  main.cpp:68 - cpp_main{M2M_DamsStart}$ C++ Trace print example
[DEBUG] 15.22  main.cpp:69 - cpp_main{M2M_DamsStart}$ C++ Debug print example
C++ Info print example
[WARN ] 15.22  main.cpp:71 - cpp_main{M2M_DamsStart}$ C++ Warning print example
[ERROR] 15.23  main.cpp:72 - cpp_main{M2M_DamsStart}$ C++ Error print example
[CRITICAL] 15.24  main.cpp:73 - cpp_main{M2M_DamsStart}$ C++ Critical print example
```

Figure 6

3.3.2 C++ method to function pointer

Sample application showing how to manage class methods as function pointers. Debug prints on MAIN_UART

Features

- how to define a class object with a generic method with the same prototype as a m2mb callback function (in this case, a hw timer callback)
- how to use a single static function in the class workspace to call multiple class instances method by using “this” as argument in the timer creation
- how to configure the static function to convert the input parameter with a static cast and call the input class instance method

3.3.2.1 Application workflow

M2MB_main.c

- Call C++ main function

main.cpp

- Create two HwTimer class instance with different timeouts
- Start both timers.
- Each will expire at a different time, and both m2mb timers will call the static function, which will run the appropriate class instance method as callback.

```
Starting C++ method as function pointer example. This is v1.0.11-C1 built on Jul  9 2020 14:58:25.
Timer "first" created with 1000 ms timeout
Timer "second" created with 1500 ms timeout
[DEBUG] 18.70  hwtimer:167 - start{M2M_DamsStart}$ Starting "first" timer
[DEBUG] 18.71  hwtimer:167 - start{M2M_DamsStart}$ Starting "second" timer
In the static timer callback. Calling class method...
[DEBUG] 19.73  hwtimer:179 - timer_cb{pubTspt_0}$
Timer "first" class callback called. Class instance: 0x400212e0; handle: 0x4002b288
In the static timer callback. Calling class method...
[DEBUG] 20.25  hwtimer:179 - timer_cb{pubTspt_0}$
Timer "second" class callback called. Class instance: 0x400212c8; handle: 0x4002b30c
```

Figure 7

3.4 AUX UART

Applications that provide usage examples for various functionalities, log output on Auxiliary UART

3.4.1 ATI (AT Instance)

Sample application showing how to use AT Instance functionality (sending AT commands from code). The example supports both sync and async (using a callback) modes. Debug prints on **AUX UART**

Features

- How to open an AT interface from the application
- How to send AT commands and receive responses on the AT interface

3.4.1.1 Application workflow, sync mode

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init AT0 (first AT instance)
- Send AT+CGMR command
- Print response.
- Release AT0

`at_sync.c`

- Init ati functionality and take AT0
- Send AT+CGMR command, then read response after 2 seconds, then return it
- Deinit ati, releasing AT0

```
Starting AT demo app. This is v1.0.7 built on Apr  1 2020 15:12:58.
[DEBUG] 17.15  at_sync.c:53 - at_cmd_sync_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Sending command AT+CGMR in sync mode
[DEBUG] 17.16  at_sync.c:79 - send_sync_at_command{M2M_DamsStart}$ Sending AT Command: AT+CGMR
Command response: <AT+CGMR
M0B.950004-B008

OK
>

[DEBUG] 19.21  at_sync.c:61 - at_cmd_sync_deinit{M2M_DamsStart}$ m2mb_ati_deinit() on instance 0
Application end
```

Figure 8

3.4.1.2 Application workflow, async mode

M2MB_main.c

- Open USB/UART/UART_AUX
- Init AT0 (first AT instance)
- Send AT+CGMR command
- Print response.
- Release AT0

at_async.c

- Init ati functionality and take AT0, register AT events callback
- Send AT+CGMR command, wait for response semaphore (released in callback), then read it and return it
- Deinit ati, releasing AT0

```
Starting AT demo app. This is v1.0.7 built on Apr  1 2020 15:07:45.
[DEBUG] 17.13  at_async.c:116 - at_cmd_async_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Sending command AT+CGMR in async mode
[DEBUG] 17.15  at_async.c:153 - send_async_at_command{M2M_DamsStart}$ Sending AT Command: AT+CGMR
[DEBUG] 17.15  at_async.c:169 - send_async_at_command{M2M_DamsStart}$ waiting command response...
[DEBUG] 17.17  at_async.c:88 - at_cmd_async_callback{pubTspt_0}$ Callback - available bytes: 25
[DEBUG] 17.18  at_async.c:181 - send_async_at_command{M2M_DamsStart}$ Receive response...
Command response: <AT+CGMR
M0B.950004-B008

OK
>

[DEBUG] 17.19  at_async.c:136 - at_cmd_async_deinit{M2M_DamsStart}$ m2mb_ati_deinit() on instance 0
Application end
```

Figure 9

3.4.2 AWS demo

Sample application showcasing AWS Iot Core MQTT communication. Debug prints on **AUX UART**

Features

- How to check module registration and enable PDP context
- How to load certificates into device SSL session storage
- How to configure MQTT client parameters
- How to connect to AWS server with SSL and exchange data over a topic

3.4.2.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage MQTT client and start it

`aws_demo.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init MQTT client
- Configure it with all parameters (Client ID, PDP context ID, keepalive timeout…)
- Initialize the TLS parameters (TLS1.2) andh auth mode (server+client auth in the example)
- Create SSL context
- Read certificates files and store them
- Connect MQTT client to broker
- Subscribe to topic
- Publish 10 messages with increasing counter
- Print received message in mqtt_topc_cb function
- Disconnect MQTT client and deinit it
- Disable PDP context

3.4.2.2 How to get started with AWS IoT

- Go to AWS console and create an account if one is not available yet.
- Go to **IoT Core** section
- Go to **Secure** > **Policies** section
- Create a new policy, which describes what the device will be allowed to do (e.g. subscribe, publish)
- Give it a name, then configure it using the configuration below (it is possible to copy/paste by clicking on **Add statements** section, then **Advanced mode**) :

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
"Action": [
  "iot:Publish",
  "iot:Subscribe",
  "iot:Connect",
  "iot:Receive"
],
"Effect": "Allow",
"Resource": [
  "*"
]
  }
  ]
}
```

- Click on create to complete the policy creation.
- Go to **Manage** section
- Press **Create**, then **Create a single thing**
- Give the new thing a name, then click on Next
- Select **One-click certificate creation (recommended)** by clicking on **Create certificate**
- Once presented with the **Certificate created** page, download all certificates and keys
- Click on the **Activate** button to enable the certificate authentication of the newly created device
- Click on **Attach a policy** and select the policy created in a previous step

For further information, please refer to the full AWS IoT documentation

3.4.2.3 Application setup

- Set **CLIENTCERTFILE** and **CLIENTKEYFILE** defines in **aws_demo.c file** in order to match the certificate and key created in the previous section.
- Set **AWS_BROKER_ADDRESS** to the correct AWS URL. It can be retrieved from AWS IoT **Manage** > **Things** > **Interact** in the HTTPS **Rest API Endpoint** URL.
- Set **CLIENT_ID** to the desired Client ID for your AWS device
- (Optional) if required, change **CACERTFILE** to match the one to be used.

3.4.2.4 Device setup

The application requires the certificates (provided in sample app **certs** subfolder) to be stored in **/mod/ssl_certs/** folder. It can be created with

```
AT#M2MMKDIR=/mod/ssl_certs
```

Certificates can then be loaded with

```
AT#M2MWRITE="/mod/ssl_certs/preload_CACert_01.crt",1468 AT#M2MWRITE="/mod/ssl_certs
IoT.crt",1646
```

providing the file content in RAW mode (for example using the “Transfer Data” button in Telit AT Controller)

For client certificates, the commands will be

```
AT#M2MWRITE="/mod/ssl_certs/xxxxx.crt",yyyy
AT#M2MWRITE="/mod/ssl_certs/xxxxx.key",zzzz
```

PLEASE NOTE: always verify the file sizes to be used in the commands above as they might change

```
Starting AWS IoT Core MQTT demo app. This is v1.1.5 built on Apr 30 2021 09:05:17.
[DEBUG] 15.51  aws_demo:607 - AWS_Task{MQTT_TASK}$ Init MQTT client for AWS
[DEBUG] 15.52  aws_demo:265 - PrepareSSLEnvironment{MQTT_TASK}$ m2mb_ssl_config SNI succeeded
[DEBUG] 15.52  aws_demo:271 - PrepareSSLEnvironment{MQTT_TASK}$ Root CA cert file /mod/ssl_certs/preload_CACert_01.crt
[DEBUG] 15.52  aws_demo:297 - PrepareSSLEnvironment{MQTT_TASK}$ Buffer successfully received from file. 1468 bytes were loaded.
[DEBUG] 15.52  aws_demo:308 - PrepareSSLEnvironment{MQTT_TASK}$ Cross Signed CA cert file /mod/ssl_certs/Amazon-IoT.crt
[DEBUG] 15.52  aws_demo:334 - PrepareSSLEnvironment{MQTT_TASK}$ Buffer successfully received from file. 1646 bytes were loaded.
[DEBUG] 15.52  aws_demo:360 - PrepareSSLEnvironment{MQTT_TASK}$ Client certificate file /mod/ssl_certs/ab71          -certificate.pem.crt
[DEBUG] 15.52  aws_demo:384 - PrepareSSLEnvironment{MQTT_TASK}$ Buffer successfully received from file. 1224 bytes were loaded.
[DEBUG] 15.52  aws_demo:396 - PrepareSSLEnvironment{MQTT_TASK}$ Client Key file /mod/ssl_certs/ab71          -private.pem.key
[DEBUG] 15.52  aws_demo:422 - PrepareSSLEnvironment{MQTT_TASK}$ Buffer successfully received from file. 1679 bytes were loaded.

SSL environment preparation completed
[DEBUG] 15.52  aws_demo:726 - AWS_Task{MQTT_TASK}$ Waiting for registration...
[DEBUG] 15.52  aws_demo:514 - NetCallback{pubTspt_0}$ Module is registered
[DEBUG] 15.52  aws_demo:738 - AWS_Task{MQTT_TASK}$ PDP context initialization
[DEBUG] 17.55  aws_demo:753 - AWS_Task{MQTT_TASK}$ Activate PDP with APN web.omnitel.it on CID 1....
[DEBUG] 18.37  aws_demo:557 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 18.37  aws_demo:561 - PdpCallback{pubTspt_0}$ IP address: 109.114.102.21

Connecting to Server <angy83rl5oizs-ats.iot.eu-west-2.amazonaws.com>:8883...
Done.
[DEBUG] 27.87  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 2, "data": {"RSSI": -72, "tm":"2021-04-30,09:05:50"} }> to topic
device/35308109C          /updates
[DEBUG] 27.87  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 30.94  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 3, "data": {"RSSI": -72, "tm":"2021-04-30,09:05:53"} }> to topic
device/353081090          /updates
[DEBUG] 30.94  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 33.99  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 4, "data": {"RSSI": -72, "tm":"2021-04-30,09:05:56"} }> to topic
device/35308109C          /updates
[DEBUG] 34.00  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 37.00  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 5, "data": {"RSSI": -72, "tm":"2021-04-30,09:05:59"} }> to topic
device/35308109           /updates
[DEBUG] 37.00  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 40.03  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 6, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:02"} }> to topic
device/35308109           /updates
[DEBUG] 40.03  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 43.05  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 7, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:05"} }> to topic
device/353081             /updates
[DEBUG] 43.05  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 46.13  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 8, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:08"} }> to topic
device/3530810            /updates
[DEBUG] 46.13  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 49.15  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 9, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:11"} }> to topic
device/3530810            /updates
[DEBUG] 49.15  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 52.19  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 10, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:14"} }> to topic
device/3530810            /updates
[DEBUG] 52.19  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 55.22  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 11, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:17"} }> to topic
device/3530810            /updates
[DEBUG] 55.22  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.

Disconnecting from MQTT broker..
[DEBUG] 58.27  aws_demo:878 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 58.27  aws_demo:908 - AWS_Task{MQTT_TASK}$ application exit
[DEBUG] 58.27  aws_demo:918 - AWS_Task{MQTT_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 58.27  aws_demo:924 - AWS_Task{MQTT_TASK}$ Application complete.
```

Figure 10

Data received from a subscriber:

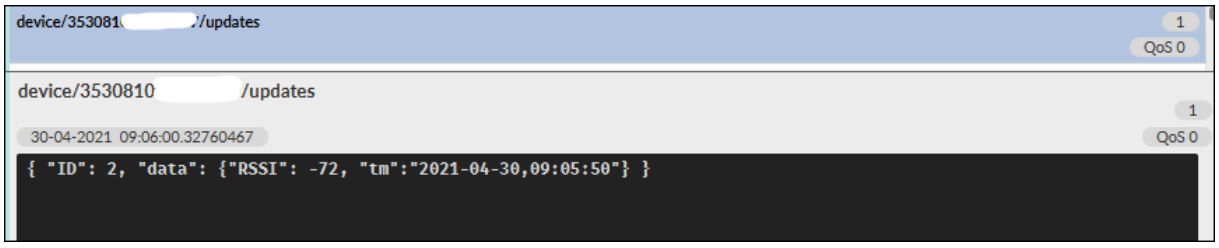

Figure 11

3.4.3 App Manager

Sample application showing how to manage AppZone apps from m2mb code. Debug prints on **AUX UART**

Features

- How to get how many configured apps are available
- How to get the handle to manage the running app (change start delay, enable/disable)
- How to create the handle for a new binary app, enable it and set its parameters
- How to start the new app without rebooting the device, then stop it after a while.

3.4.3.1 Prerequisites

This app will try to manage another app called “second.bin”, which already exists in the module filesystem and can be anything (e.g. another sample app as GPIO toggle). the app must be built using the flag ROM_START=

in the Makefile to set a different starting address than the main app (by default, 0x40000000). For example, 0x41000000.

3.4.3.2 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- get a non existing app handle and verify it is NULL
- get the current app handle, then get the start delay **set in the INI file (so persistent)**
- change the current app delay value **in the INI file**
- verify that the change has been stored
- get current app state
- create an handle for a second application binary.
- add it to the INI file
- set its execution flag to 0
- get the delay time and the state from INI file for the new app
- get the current set address for the new app
- set the app delay **in RAM, INI will not be affected**.
- start the new app without reboot, using the right set delay
- wait some time, then get the app state and the used RAM amount
- wait 10 seconds, then stop the second app.
- set its execution flag to 1 so it will run at next boot.

```
Starting App Manager demo app. This is v1.0.14-C1 built on Sep 24 2020 12:33:25.
There are 2 configured apps.
Not existing app handle test (should be 0): 0x0
Manager app handle: 0x809e20e0
Manager app delay from nv memory: 5 seconds

Changing Manager app delay time (on non volatile configuration) to 5 seconds..
Manager app delay from nv memory is now 5 seconds
Manager app state is M2MB_APPMNG_STATE_RUN

Trying to get Second app handle...
Second app handle is valid
2nd app delay from nv memory is 1
2nd app current state is M2MB_APPMNG_STATE_READY
Second app current address is 0x41000000
Setting volatile Second app delay (not stored in nvm) to 0 seconds...
Starting Second app on the fly (without reboot)...
Waiting 2 seconds...
2nd app current state is M2MB_APPMNG_STATE_RUN
Second app is running!
Second App is using 475136 bytes of RAM
Stopping Second app now...
wait 10 seconds...
2nd app current state is M2MB_APPMNG_STATE_STOP
Set permanent run permission for Second app.
Done. Second App will also run from next boot-up
```

Figure 12

3.4.4 App update OTA via FTP

Sample application showcasing Application OTA over FTP with AZX FTP. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to connect to a FTP server
- How to download an application binary and update the local version

The app uses a predefined set of parameters. To load custom parameters, upload the `ota_config.txt` file (provided in project's `/src` folder) in module's `/mod` folder, for example with

```
AT#M2MWRITE="/mod/ota_config.txt",<filesize>
```

3.4.4.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage app OTA and start it

`ftp_utils.c`

- Set parameters to default
- Try to load parameters from `ota_config.txt` file
- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Initialize FTP client
- Connect to FTP server and log in
- Get new App binary file size on remote server
- Download the file in `/mod` folder, with the provided name
- Close FTP connection
- Disable PDP context
- Update applications configuration in **app_utils.c**

`app_utils.c`

- Set new application as default
- Delete old app binary
- Restart module

```
Starting FTP APP OTA demo app. This is v1.0.7 built on Apr  7 2020 17:04:05.
[DEBUG] 21.23  ftp_utils.c:447 - msgFTPTask{FTPOTA_TASK}$ INIT
[DEBUG] 21.25  ftp_utils.c:152 - readConfigFromFile{FTPOTA_TASK}$ Reading parameters from file
[DEBUG] 21.26  ftp_utils.c:154 - readConfigFromFile{FTPOTA_TASK}$ Opening /mod/ota_config.txt in read mode..
Set APN to: <<web.omnitel.it>>
Set FTP URL to: <<ftp.telit.com>>
Set FTP PORT to: 21
Set FTP USER to: <<[illegible]>>
Set FTP PASS to: <<[illegible]>>
Set FTP FILE URI to: <</samples/APP_OTA/helloworld.bin>>
Set LOCAL FINAL APP NAME to: <<helloworld.bin>>
Set LOCAL ORIGINAL APP NAME to: <<m2mapz.bin>>
[DEBUG] 23.53  ftp_utils.c:464 - msgFTPTask{FTPOTA_TASK}$ m2mb_os_ev_init success
[DEBUG] 23.54  ftp_utils.c:470 - msgFTPTask{FTPOTA_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.55  ftp_utils.c:478 - msgFTPTask{FTPOTA_TASK}$ Waiting for registration...
[DEBUG] 23.56  ftp_utils.c:371 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 23.56  ftp_utils.c:491 - msgFTPTask{FTPOTA_TASK}$ Pdp context activation
[DEBUG] 23.57  ftp_utils.c:495 - msgFTPTask{FTPOTA_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 25.61  ftp_utils.c:504 - msgFTPTask{FTPOTA_TASK}$ Activate PDP with APN web.omnitel.it on cid 3....
[DEBUG] 26.30  ftp_utils.c:398 - PdpCallback{pubTspt_0}$ Context active
[DEBUG] 26.30  ftp_utils.c:401 - PdpCallback{pubTspt_0}$ IP address: 176.246.110.148
Start ftp client...
[DEBUG] 27.36  ftp_utils.c:533 - msgFTPTask{FTPOTA_TASK}$ Connected.
[DEBUG] 28.87  ftp_utils.c:546 - msgFTPTask{FTPOTA_TASK}$ FTP login successful.
Get remote file /samples/APP_OTA/helloworld.bin size
[DEBUG] 29.31  ftp_utils.c:568 - msgFTPTask{FTPOTA_TASK}$ Done. File size: 116224.
Starting download of remote file /samples/APP_OTA/helloworld.bin into local /mod/helloworld.bin
/samples/APP_OTA/helloworld.bin  4.68% 5440
/samples/APP_OTA/helloworld.bin  9.36% 10880
/samples/APP_OTA/helloworld.bin 14.04% 16320
/samples/APP_OTA/helloworld.bin 18.72% 21760
/samples/APP_OTA/helloworld.bin 23.40% 27200
/samples/APP_OTA/helloworld.bin 28.08% 32640
/samples/APP_OTA/helloworld.bin 32.76% 38080
/samples/APP_OTA/helloworld.bin 37.44% 43520
/samples/APP_OTA/helloworld.bin 42.13% 48960
/samples/APP_OTA/helloworld.bin 46.81% 54400
/samples/APP_OTA/helloworld.bin 51.49% 59840
/samples/APP_OTA/helloworld.bin 56.17% 65280
/samples/APP_OTA/helloworld.bin 60.85% 70720
/samples/APP_OTA/helloworld.bin 65.53% 76160
/samples/APP_OTA/helloworld.bin 70.21% 81600
/samples/APP_OTA/helloworld.bin 74.89% 87040
/samples/APP_OTA/helloworld.bin 79.57% 92480
/samples/APP_OTA/helloworld.bin 84.25% 97920
/samples/APP_OTA/helloworld.bin 88.93% 103360
/samples/APP_OTA/helloworld.bin 93.61% 108800
/samples/APP_OTA/helloworld.bin 97.42% 113220
[DEBUG] 43.54  ftp_utils.c:608 - msgFTPTask{FTPOTA_TASK}$ download successful.
FTP quit...
[DEBUG] 43.77  ftp_utils.c:632 - msgFTPTask{FTPOTA_TASK}$ Deactivating PDP
[DEBUG] 43.77  ftp_utils.c:642 - msgFTPTask{FTPOTA_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 44.20  ftp_utils.c:407 - PdpCallback{pubTspt_0}$ Context deactive
[DEBUG] 45.44  app_utils.c:76 - update_app{FTPOTA_TASK}$ Application successfully configured.
[DEBUG] 45.45  app_utils.c:82 - update_app{FTPOTA_TASK}$ Deleting old application /mod/m2mapz.bin
€ÿStarting. This is v1.0.7 built on Apr  7 2020 17:02:52. LEVEL: 2

 Start Hello world Application  [ version: 2.000000 ]

 Hello world 2.0 [ 000001 ]
 Hello world 2.0 [ 000002 ]
 Hello world 2.0 [ 000003 ]
```

Figure 13

3.4.5 CJSON example:

Sample application showcasing how to manage JSON objects. Debug prints on **AUX UART**

Features

- How to read a JSON using cJSON library
- How to write a JSON
- How to manipulate JSON objects

3.4.5.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Parse an example string into a JSON object and print the result in a formatted string
- Print some test outcomes (e.g. non existing item correctly not found)
- Retrieve single elements from the parsed JSON object and use them to format a descriptive string
- Delete the JSON object
- Create a new JSON object appending elements to it
- Print the result JSON string from the object

```
Starting Logging demo app. This is v1.0.7 built on Apr  7 2020 08:33:03.
And here is what we got:
{
	"name":	"Atlantic Ocean",
	"format":	{
		"type":	"salt",
		"volume":	310410900,
		"depth":	-8486,
		"volume_percent":	23.300000,
		"tide":	-3.500000,
		"calm":	false,
		"life":	["plankton", "corals", "fish", "mammals"]
	}
}
inexistent key not found
name found: Atlantic Ocean
format found (null)
Our JSON string contains info about an ocean named Atlantic Ocean,
has a volume of 310410900 km^3 of salt water with -8486 meters max depth,
represents 23.3% of total oceans volume,
has an average low tide of -3.5 meters,
hosts a huge number of living creatures such as plankton, corals, fish, mammals,
and is not always calm.

Let's build a TR50 command with a proprety.publish and an alarm.publish for MQTT (no auth).
And here is what we got:
{
	"1":	{
		"command":	"property.publish",
		"params":	{
			"thingKey":	"mything",
			"key": "mykey",
			"value":	123.144000
		}
	},
	"2":	{
		"command":	"alarm.publish",
		"params":	{
			"thingKey":	"mything",
			"key": "mykey",
			"state":	3,
			"msg": "Message."
		}
	}
}
END.
```

Figure 14

3.4.6 Crypto Elliptic Curve Cryptography (ECC) example

Sample application showcasing how to manage Elliptic Curve Cryptography functionalities. Debug prints on **AUX UART**

Features

- How to initialize ECC contexts A (Alice) and B (Bob). Alice is emulating a remote host, from which a public key is known.
- How to generate keypairs for contexts and export public keys
- how to export keyblobs from a context (a keyblob is encrypted with hw specific keys, and can only be used on the module where it was created)
- How to save a keyblob in secured TrustZone.
- How to reload a keyblob from the TrustZone into an initialized context
- How to sign a message with ECDSA from context B (Bob) and verify it from another context A (Alice) with the signature and public key of Bob.
- How to make Bob and Alice derive a shared session keys using each other's public key.
- How to make Bob and Alice create an AES context with the newly created shared keys, encode data and decode it on the other side

3.4.6.1 Application workflow

`M2MB_main.c`

- Create Bob ECC context, create a keypair and export it in a keyblob
- Open a file in secured Trust Zone, then store the keyblob in it.
- Destroy Bob ECC context
- Recreate Bob ECC context, open the file from Trust Zone and read the keyblob.
- Import the keyblob in Bob context.
- Export Bob public key
- Create Alice ECC context, to simulate an external host. Generate a keypair and export the public key.
- Sign a message with Bob context, generating a signature.
- Use Alice to verify the signed message using Bob's signature and public key
- Derive a shared key for Bob, using Alice's public key
- Create an AES context for Bob
- Import the shared key into the AES context
- Encrypt a message using Bob's AES context.

- Derive a shared key for Alice, using Bob's public key
- Create an AES context for Alice
- Import the shared key into the AES context
- Decrypt the message using Alice's AES context.
- Check the decrypted message and the original one match
- Clear all resources

```
Starting Crypto ECC demo app. This is v1.0.9-C1 built on May 11 2020 16:30:23.

Bob (local) and Alice (remote) scenario
Bob's keypair generated
Bob's keyblob length is 224
Bob exported the keyblob to be securely stored.

Bob already had an item in Secure Data Area, it was removed to create a new one
Bob securely saved the keyblob in Secure Data Area
Releasing resources

Close Bob's context...
Done. Now Bob context does not exist anymore.

Re-initialize Bob Context and load the keyblob from the secure zone
Bob securely loaded the keyblob from the SDA
Import keyblob in Bob's context..
Done. Now export Bob's public key...
Bob's public key successfully exported

Alice's keypair generated
Alice's public key successfully exported

Bob's message signed with ECDSA!
Alice verified bob's message with his pubkey and signature!

------------------------------------------------------------
Bob and Alice will now exchange a message with AES encrypt
------------------------------------------------------------

Bob retrieved the generated shared key size
Bob's shared keyblob length is: 32. Allocate the required memory to store it.
Bob created a shared key using Alice's public key!

Bob created an AEX context to exchange encrypted data with Alice
Bob's AES context imported the shared keyblob
Bob Encrypted the message using AES and the shared key!
Encrypted data:
 94EE531E3B84B2A4EF05502186BFF5DA

Alice retrieved the generated shared key size
Alice's shared keyblob length is: 32. Allocate the required memory to store it.
Alice created a shared key using Bob's public key!

Alice created an AEX context to exchange encrypted data with Bob
Alice's AES context imported the shared keyblob
Alice decrypted the message using AES and the shared key!
Decrypted:
 414094941E8942A4445548035BFAE943

Original, plain message:
 414094941E8942A4445548035BFAE943

Plain and decrypted messages match!
```

Figure 15

3.4.7 EEPROM 24AA256

Sample application showing how to communicate with a MicroChip 24AA256T I2C EEPROM chip using azx eeprom utility APIs. Debug prints on **AUX UART**

Setup

This demo application requires that: - A0, A1, and A2 pins (1,2,3 chip pins) are connected to ground (pin 4) for device address 0xA0 - Pin 7 (WP) is connected to ground - Pin 6 (SCL) is connected to module GPIO 3 - Pin 5 (SDA) is connected to module GPIO 2 - Pin 4 is connected to one of the ground pins of the module - Pin 8 is connected to 1v8 supply (e.g. VPWRMON pin on the module)

Features

- Initialize the logs on the output channel
- configure the EEPROM utility, setting the slave address and the memory parameters (page size, memory size)
- Write single bytes on a random address
- Read written bytes as a page
- Write data using pages
- Read the new data using pages
- Read again using sequential reading
- Read a single byte from a specific address
- Read next byte using read from current address
- Erase the EEPROM
- Deinit EEPROM utility

3.4.7.1 Application workflow

`M2MB_main.c`

- call azx_eeprom_init() to set the utility parameters (SDA and SCL pins, page and memory sizes)
- call azx_eeprom_writeByte() to store a single byte with value “5” at the address 0x0213
- call azx_eeprom_writeByte() to store a single byte with value “6” at the address 0x0214
- call azx_eeprom_readPages() from address 0x0213 to retrieve the 2 bytes from the EEPROM
- call azx_eeprom_writePages to write 1024 bytes from a buffer, starting from address 0x00
- call azx_eeprom_readPages() again, to read 256 bytes from address 0x00
- call azx_eeprom_readSequentially() to read 256 bytes from 0x00 by without pages (less overhead on I2C protocol)

- call azx_eeprom_readByte() to get a single byte from address 0x00
- call azx_eeprom_readByteFromCurrentAddress() to get a byte from next address (0x01)
- call azx_eeprom_eraseAll() to completely erase the EEPROM memory (this writes 0xFF in each byte)
- call azx_eeprom_readPages from address 0x0213 to get 2 bytes and verify the values have been written to 0xFF
- call azx_eeprom_deinit to close the eeprom handler and the I2C channel

```
Starting I2C EEPROM 24AA256T demo app. This is v1.0.13-C1 built on Nov  3 2020 16:28:23.

Configuring the I2C device...
Opening I2C channel /dev/I2C-160 ( device address is 0xA0 )
Writing 1 byte at address 0x0213...
Done.
Writing 1 byte at address 0x0214...
Done.
Reading the 2 bytes from address 0x0213...
Done. Data: [0xFF 0xFF]

Writing 1024 bytes at address 0x0000..
Done.

Reading 256 bytes from address 0x0000...
Done. Data:
<<ABCDEFGHIJKLMNOPQRSTUVWXYZ.............................................abcdefghijklmnopqrstuvwxyz................

Reading 256 bytes sequentially from address 0x0000...
Done. Data:
<<ABCDEFGHIJKLMNOPQRSTUVWXYZ.............................................abcdefghijklmnopqrstuvwxyz................


Reading 1 byte from address 0x0000...
Done. Data: 'A'

Reading 1 byte from current address (should be 0x0001)...
Done. Data: 'B'


[DEBUG] 17.47  M2MB_main:177 - run_I2C_EEPROM_Demo{M2M_DamsStart}$ Erasing all the eeprom...
[DEBUG] 28.05  M2MB_main:185 - run_I2C_EEPROM_Demo{M2M_DamsStart}$ Done

Reading the 2 bytes from address 0x0213...
Done. Data: [0xFF 0xFF]

Deinit EEPROM...
Done
```

Figure 16

3.4.8 Easy AT example

Sample application showcasing Easy AT functionalities. Debug prints on **AUX UART**

Features

- Shows how to register custom commands

The application adds two custom commands to the list of available ones:

- AT#MYCMD
- AT#MYINPUT

3.4.8.1 AT#MYCMD

This is a simple parameter-waiting command. It expects one string parameter, and will print it on the logging interface once received. The command simply returns OK

3.4.8.2 AT#MYINPUT

This command expects a numeric parameter, which indicates how many bytes will be received over the interface at most (the command will provide a prompt indicating it is waiting data). Then the data management callback will print when data is received, and if CTRL+Z (0x1A in hex) is received, it will complete the process, printing in the log interface what was received. sending ESC will terminate the process discarding any pending data.

3.4.9 Events

Sample application showcasing events setup and usage. Debug prints on **AUX UART**

Features

- How to setup OS events with a custom bitmask
- How to wait for events and generate them in callback functions to synchronize blocks of code

3.4.9.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create an event handler
- Create a timer to generate an event, with a 2 seconds expiration time
- Wait for a specific event bit on the event handler
- At timer expiration, set the same event bit and verify that the code flow went through after the event.

```
Starting Events demo app. This is v1.0.7 built on Apr  7 2020 08:44:29.
[DEBUG] 20.55  M2MB_main.c:171 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
Set the timer attributes structure success.
Timer successfully created
[DEBUG] 20.57  M2MB_main.c:125 - setup_timer{M2M_DamsStart}$ Start the timer, success.
[DEBUG] 22.60  M2MB_main.c:60 - hwTimerCb{pubTspt_0}$ Timer Callback, generate event!
[DEBUG] 22.61  M2MB_main.c:183 - M2MB_main{M2M_DamsStart}$ event occurred!
```

Figure 17

3.4.10 Events - Barrier (multi events)

Sample application showcasing how to setup and use multiple events to create a barrier. Debug prints on **AUX UART**

Features

- How to setup OS events to be used as a barrier
- How to wait for multiple events in the same point, and generate them in call-back functions to synchronize blocks of code

3.4.10.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create an event handler
- Create a timer to generate an event, with a 3 seconds expiration time
- Create another timer to generate an event, with a 6 seconds expiration time
- Start both timers
- Wait for both event bits on the event handler (each one will be set by one of the timers)
- At first timer expiration, set the first event bit and verify that the code flow does not procede.
- At second timer expiration, set the second event bit and verify that the code flow went through after the event (implementing a barrier).

```
Starting Barrier demo app. This is v1.0.7 built on Apr  7 2020 08:48:30.
[DEBUG] 20.01  M2MB_main.c:179 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
Set the timer attributes structure success.
Timer successfully created with 3000 timeout (ms)
Set the timer attributes structure success.
Timer successfully created with 6000 timeout (ms)
[DEBUG] 23.08  M2MB_main.c:66 - hwTimerCb1{pubTspt_0}$ Timer Callback, generate event 1!
[DEBUG] 26.12  M2MB_main.c:75 - hwTimerCb2{pubTspt_0}$ Timer Callback, generate event 2!
[DEBUG] 26.13  M2MB_main.c:214 - M2MB_main{M2M_DamsStart}$ BOTH events occurred!
```

Figure 18

3.4.11 FOTA example

Sample application showcasing FOTA usage with M2MB API. Debug prints on **AUX UART**

Features

- How download a delta file from a remote server
- How to apply the delta and update the module firmware

3.4.11.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a main task to manage connectivity.
- create a fota task to manage FOTA and start it with INIT option

`fota.c`

fotaTask()

- Initialize FOTA system then reset parameters.
- Check current FOTA state, if not in IDLE, return error.
- Send a message to mainTask so networking is initialized.
- after PdPCallback() notifies the correct context activation, configure the fota client parameters such as FTP server URL, username and password
- get delta file from server. when it is completed, FOTADownloadCallback is called.
- If delta download went fine, check it.
- If delta file is correct, apply it. Once complete, restart the module.

mainTask()

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context. Event will be received on **PdP-Callback** function
- Disable PDP context when required to stop the app

PdpCallback()

- When PDP context is enabled, send a message to fotaTask to start the download

```
Starting FOTA demo app. This is v1.1.7 built on Jun 11 2021 12:20:43.
[DEBUG] 23.60  fota:187 - fotaTask{FOTA_TASK}$ Init FOTA...

Session file not present, procede with FOTA...
[DEBUG] 23.61  fota:236 - fotaTask{FOTA_TASK}$ m2mb_fota_reset PASS
[DEBUG] 23.61  fota:260 - fotaTask{FOTA_TASK}$ m2mb_fota_state_get  M2MB_FOTA_STATE_IDLE
[DEBUG] 23.62  fota:379 - mainTask{MAIN_TASK}$ INIT
[DEBUG] 23.62  fota:392 - mainTask{MAIN_TASK}$ m2mb_os_ev_init success
[DEBUG] 23.63  fota:398 - mainTask{MAIN_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.63  fota:405 - mainTask{MAIN_TASK}$ Waiting for registration...
[DEBUG] 23.64  fota:131 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 23.65  fota:418 - mainTask{MAIN_TASK}$ Pdp context initialization
[DEBUG] 25.70  fota:431 - mainTask{MAIN_TASK}$ Activate PDP with APN web.omnitel.it on cid 1....
[DEBUG] 35.42  fota:152 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 35.43  fota:155 - PdpCallback{pubTspt_0}$ IP address: 2.41.116.139

[DEBUG] 35.43  fota:285 - fotaTask{FOTA_TASK}$
Trying to download "samples/FOTA/37.00.003.3_to_37.00.003.1_ME310G1_NANVWWAU.bin" delta file...
[DEBUG] 35.45  fota:295 - fotaTask{FOTA_TASK}$ m2mb_fota_get_delta OK - Waiting for the completion callback
[DEBUG] 119.43  fota:96 - FOTADownloadCallBack{pubTspt_0}$ FOTA download Success - performing packet validation...
[DEBUG] 119.44  fota:301 - fotaTask{FOTA_TASK}$ Validating delta file...
[DEBUG] 156.36  fota:317 - fotaTask{FOTA_TASK}$ Packet is valid, start update...
[DEBUG] 156.40  fota:329 - fotaTask{FOTA_TASK}$ m2mb_fota_start PASS
[DEBUG] 158.36  fota:342 - fotaTask{FOTA_TASK}$
Rebooting...After reboot there will be the new FW running on module!

#OTAEV: Module Upgraded To New Fw
Starting FOTA demo app. This is v1.1.7 built on Jun 11 2021 12:20:43.
[DEBUG] 29.24  fota:187 - fotaTask{FOTA_TASK}$ Init FOTA...

Session file is already present, stop.
```

Figure 19

3.4.12 FOTA from Local File example

Sample application that shows how perform FOTA upgrade using a delta file stored into file system. Debug prints on **AUX UART**

Features

- How to store and get FOTA upgrade information to/from a file
- How to get delta file from module file system
- How to apply the delta and update module firmware

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Check if module has been already upgraded or needs to be upgraded reading FOTA upgrade status from a file
- Create a fota task to manage FOTA and start it with INIT option

smartFotaTask() - Initialize FOTA system then reset parameters. - Get FOTA partiton size and block size - Copy delta file from file system to FOTA paartition. when it is completed, FOTADownloadCallback is called. - If delta file is correct, apply it. Once complete, write FOTA status flag and current fw version to a file, restart the module.

```
Starting FOTA application from local delta file demo app. This is v1.0.14-C1 built on May 16 2022 15:50:38.


Check FOTA upgrade status in file system

Module current fw version is:
37.00.214-P0C.210001
M0C.200003
P0C.210001
A0C.210000
[WARN ] 13.83  M2MB_main:190 - FOTAUpgradeStatusInit{M2M_DamsStart}$ File doesn't exist create it, first app execution
File created, store current fw version and fota upgrade flag=0

Read stored data

FOTA Flag: 0 Module previous fw version:
37.00.214-P0C.210001
M0C.200003
P0C.210001
A0C.210000

Start FOTA process
[DEBUG] 15.42  M2MB_main:450 - smartFotaTask{FOTA_TASK}$ INIT
[DEBUG] 15.43  M2MB_main:461 - smartFotaTask{FOTA_TASK}$ m2mb_fota_init success
[DEBUG] 15.44  M2MB_main:464 - smartFotaTask{FOTA_TASK}$
Get block and FOTA partition size
OTA blockSize: 131072
OTA partitionSize: 11272192
[DEBUG] 15.47  M2MB_main:472 - smartFotaTask{FOTA_TASK}$
 Copy delta file from File system to FOTA partition
[DEBUG] 15.47  M2MB_main:277 - copyFromFileSystemToDelta{FOTA_TASK}$ File size: 746552
[DEBUG] 15.87  M2MB_main:480 - smartFotaTask{FOTA_TASK}$ Delta file writing completed
[DEBUG] 15.88  M2MB_main:484 - smartFotaTask{FOTA_TASK}$
Delta file check...
[DEBUG] 15.89  M2MB_main:347 - check_fota_delta{FOTA_TASK}$ -- check_fota_delta...
[DEBUG] 54.73  M2MB_main:366 - check_fota_delta{FOTA_TASK}$ Fota check integrity PASS
[DEBUG] 54.74  M2MB_main:492 - smartFotaTask{FOTA_TASK}$ ...delta file OK
[DEBUG] 54.75  M2MB_main:496 - smartFotaTask{FOTA_TASK}$
--> Start update...
[DEBUG] 56.83  M2MB_main:403 - startUpdate{FOTA_TASK}$
Reboot module to start delta deployment
```

Figure 20

3.4.13 FTP

Sample application showcasing FTP client demo with AZX FTP. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to connect to a FTP server
- How to exchange data with the server

3.4.13.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage FTP client and start it

`ftp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init FTP client and set the debug function for it
- Connect to the server
- Perform log in
- Check remote file size and last modification time
- Download file from server to local filesystem. A data callback is set to report periodic info about the download status
- Upload the same file to the server with a different name. A data callback is set to report periodic info about the upload status
- Download another file content in a buffer instead of a file. A data callback is set to report periodic info about the download status
- Close the connection with FTP server
- Disable PDP context

```
Starting FTP demo app. This is v1.0.7 built on Apr  7 2020 11:17:36.
[DEBUG] 21.23  ftp_test.c:290 - msgFTPTask{FTP_TASK}$ INIT
[DEBUG] 21.23  ftp_test.c:304 - msgFTPTask{FTP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.23  ftp_test.c:310 - msgFTPTask{FTP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.23  ftp_test.c:318 - msgFTPTask{FTP_TASK}$ Waiting for registration...
[DEBUG] 21.25  ftp_test.c:214 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 21.26  ftp_test.c:331 - msgFTPTask{FTP_TASK}$ Pdp context activation
[DEBUG] 21.27  ftp_test.c:335 - msgFTPTask{FTP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.31  ftp_test.c:344 - msgFTPTask{FTP_TASK}$ Activate PDP with APN web.omnitel.it on cid 3....
[DEBUG] 24.09  ftp_test.c:241 - PdpCallback{pubTspt_0}$ Context active
[DEBUG] 24.10  ftp_test.c:244 - PdpCallback{pubTspt_0}$ IP address: 176.244.166.181
Start ftp client...
[DEBUG] 24.82  ftp_test.c:373 - msgFTPTask{FTP_TASK}$ Connected.
[DEBUG] 26.32  ftp_test.c:386 - msgFTPTask{FTP_TASK}$ FTP login successful.
Get remote file /samples/pattern_big.txt size
[DEBUG] 26.69  ftp_test.c:428 - msgFTPTask{FTP_TASK}$ Done. File size: 20026.
Get remote file /samples/pattern_big.txt last modification date
[DEBUG] 26.89  ftp_test.c:450 - msgFTPTask{FTP_TASK}$ Done. File last mod date: 20200407090654
.

Starting download of remote file /samples/pattern_big.txt into local /mod/_pattern_big.txt
/samples/pattern_big.txt 47.54% 9520
/samples/pattern_big.txt 100.00% 20026
[DEBUG] 29.75  ftp_test.c:488 - msgFTPTask{FTP_TASK}$ download successful.
[DEBUG] 29.76  ftp_test.c:522 - msgFTPTask{FTP_TASK}$
Local file /mod/_pattern_big.txt size: 20026

Starting upload of local file /mod/_pattern_big.txt
/mod/_pattern_big.txt 81.81% 16384
Upload successful.

Starting download of remote file /samples/pattern.txt into local buffer
Getting remote file /samples/pattern.txt size..
[DEBUG] 32.97  ftp_test.c:583 - msgFTPTask{FTP_TASK}$ Done. File size: 988.
Starting download of remote file /samples/pattern.txt to buffer
[DEBUG] 34.08  ftp_test.c:145 - buf_data_cb{FTP_TASK}$ Received START event
[DEBUG] 34.09  ftp_test.c:149 - buf_data_cb{FTP_TASK}$ Received DATA: 988 bytes on buffer 0x400399e0
[DEBUG] 34.26  ftp_test.c:153 - buf_data_cb{FTP_TASK}$ Received END event
[DEBUG] 34.26  ftp_test.c:623 - msgFTPTask{FTP_TASK}$ Download successful. Received 988 bytes<<<
0   |---------|   |---------|   |---------|   |---------|   |---------|  *
1   |    A    |   |    A    |   |    A    |   |    A    |   |    A    |   *
2   |   AAA   |   |   AAA   |   |   AAA   |   |   AAA   |   |   AAA   |    *
3   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |     *
4   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |      *
5   |AAAAAAAAA|   |AAAAAAAAA|   |AAAAAAAAA|   |AAAAAAAAA|   |AAAAAAAAA|       *
6   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |      *
7   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |     *
8   |   AAA   |   |   AAA   |   |   AAA   |   |   AAA   |   |   AAA   |    *
9   |    A    |   |    A    |   |    A    |   |    A    |   |    A    |   *
10  |---------|   |---------|   |---------|   |---------|   |---------|  *
11                                                                      *
12  |---------|   |---------|   |---------|   |---------|   |--->>>
```

Figure 21

3.4.14 File System example

Sample application showcasing M2MB File system API usage. Debug prints on **AUX UART**

Features

- How to open a file in write mode and write data in it
- How to reopen the file in read mode and read data from it

3.4.14.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Print welcome message
- Open file in write mode
- Write data in file
- Close file
- Reopen file in read mode
- Read data from file and print it
- Close file and delete it

```
Starting FileSystem demo app. This is v1.0.7 build on Mar 26 2020 09:50:19. LEVEL: 2
Opening /my_text_file.txt in write mode..
Buffer written successfully into file. 15 bytes were written.
Closing file.
Opening /my_text_file.txt in read only mode..
Received 15 bytes from file:
<Hello from file>
Closing file.
Deleting File
File deleted
App Completed
```

Figure 22

3.4.15 GNSS example

Sample application showing how to use GNSS functionality. Debug prints on **AUX UART**

Features

- How to enable GNSS receiver on module
- How to collect location information from receiver

Note: on MEx10G1 product family both M2MB_GNSS_SERVICE_NMEA_REPORT and M2MB_GNSS_SERVICE_POSITION_REPORT services are available, while on ME910C1 product family only M2MB_GNSS_SERVICE_POSITION_REPORT is available

3.4.15.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print a welcome message
- Create GNSS task and send a message to it

`gps_task.c` - Init Info feature and get module type - Init gnss, enable position/NMEA report and start it. - When a fix or a NMEA sentence is available, a message will be printed by the GNSS callback function

```
Starting GNSS demo app. This is v1.1.4 built on Oct  1 2021 15:27:44.

Model: ME910C1-E2

m2mb_gnss_enable, POSITION OK

m2mb_gnss_start OK, waiting for position/nmea sentences...
latitude_valid: 1 - latitude: 45.713643
longitude_valid: 1 - longitude: 13.738041
altitude_valid: 1 - altitude: 195.000000
uncertainty_valid: 1 - uncertainty: 95.000000
velocity_valid: 1 - codingType: 0
speed_horizontal: 0.650000
bearing: 0.000000
timestamp_valid: 1 -timestamp: 1633095357439
speed_valid: 1 - speed: 1.471360


***** Wait 120 seconds and then stop GPS *****
```

Figure 23

3.4.16 GPIO interrupt example

Sample application showing how to use GPIOs and interrupts. Debug prints on **AUX UART**

Features

- How to open a GPIO in input mode with interrupt
- How to open a second GPIO in output mode to trigger the first one

3.4.16.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Open GPIO 4 as output
- Open GPIO 3 as input and set interrupt for any edge (rising and falling). **A jumper must be used to short GPIO 3 and 4 pins.**
- Toggle GPIO 4 status high and low every second
- An interrupt is generated on GPIO 3

```
Starting GPIO interrupt demo app. This is v1.0.7 built on Mar 26 2020 16:33:01.
Setting gpio 3 interrupt...
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
```

Figure 24

3.4.17 GTP example

Sample application that shows hot to get the position using GTP feature. Debug prints on **AUX UART**

Features

- How to init and enable GTP feature
- How to get the position using GTP

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Init NET functionality and wait for module to be registered
- Init PDP functionality and set APN on CID1
- Init GTP functionality
- Check if GTP is already enabled. If not enable it and reboot module
- If GTP is enabled, get position

```
Starting GTP demo app. This is v1.1.8 built on Dec 20 2022 15:29:19.

[DEBUG] 15.08  M2MB_main:237 - M2MB_main{M2M_DamsStart}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 15.09  M2MB_main:245 - M2MB_main{M2M_DamsStart}$ Waiting for registration...
[DEBUG] 15.10  M2MB_main:253 - M2MB_main{M2M_DamsStart}$ Pdp context setting
[DEBUG] 15.11  M2MB_main:82 - NetCallback{pubTspt_0}$ Module is registered to cell 0x20!
[DEBUG] 15.11  M2MB_main:257 - M2MB_main{M2M_DamsStart}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 15.13  M2MB_main:268 - M2MB_main{M2M_DamsStart}$ m2mb_pdp_deinit returned M2MB_RESULT_SUCCESS

Check if GTP has been already enabled
[DEBUG] 16.19  M2MB_main:312 - M2MB_main{M2M_DamsStart}$ GTP status: 1 => Enabled
[DEBUG] 16.20  M2MB_main:317 - M2MB_main{M2M_DamsStart}$ Get the position...

m2mb_gnss_GTP OK, waiting for position...
[DEBUG] 107.96  M2MB_main:160 - gnssCallbackFN{pubTspt_0}$ gnssCallback event: 1
GTP position got
latitude: 41.900002
longitude: 12.500000
altitude: 0.000000
altitudeMeanSeaLevel: -0.000000
accuracy: 0.000000
speed: -0.000000
bearing: -0.000000
timestamp: 0
verticalAccuracy 0.000000
speedAccuracy -0.000000
bearingAccuracy: -0.000000
```

Figure 25

3.4.18 General_INFO example

Sample application prints some Module/SIM information as IMEI, fw version, IMSI and so on; it prints also some information about registration. Debug prints on **AUX UART**

Features

- How to print some Module information as IMEI, FW version etc
- How to print some SIM information as IMSI, ICCID
- How to get and print some informatio about Module registration as Netowrk Operator, AcT, RSSI, etc

3.4.18.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Init NET functionality
- Init INFO functionality
- Get and print Module and SIM info
- Wait form module to register to network
- Get and print registration INFO

```
Starting. This is v1.1.4 built on Mar 31 2021 09:56:03. LEVEL: 2


 Start General INFO application [ version: 1.000000 ]


=======================

MODULE ME910C1-E2 INFO

=======================

MANUFACTURER: Telit

IMEI: 353080091125422

MODEM FIRMWARE VERSION: M0B.700005

PACKAGE VERSION:
30.00.709-B005-P0B.700100
M0B.700005
P0B.700100
A0B.700000


========

SIM INFO

========

IMSI: 222015602268648

ICCID: 89390100001138084906


============================

Waiting for registration...

============================

Module is registered to HOME network cellID 0x5221
NETWORK OPERATOR (mcc mnc): 222 01
Network Technology 2G (AcT: 0) RSSI: -81
```

Figure 26

3.4.19 HTTP Client

Sample application showing how to use HTTPs client functionalities. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to initialize the http client, set the debug hook function and the data callback to manage incoming data
- How to perform GET, HEAD or POST operations (GET also with single range support)

NOTE: the sample app has an optional dependency on azx_base64.h if basic authentication is required (refer to `HTTP_BASIC_AUTH_GET` define in `M2MB_main.c` for further details)

3.4.19.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage HTTP client and start it

`httpTaskCB`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create HTTP client options and initialize its functionality
- Create HTTP SSL config and initialize the SSL options
- Configure data management options for HTTP client
- Appy all configurations to HTTP client
- Perform a GET request to a server
- Disable PDP context

`DATA_CB`

- Print incoming data
- Set the abort flag to 0 to keep going.

```
Starting HTTP(s) client demo app. This is v1.1.15 built on Nov 28 2022 12:12:27.
[DEBUG] 17.12  read_parame:101 - readConfigFromFile{HttpClient}$ Reading parameters from file
[DEBUG] 17.13  read_parame:103 - readConfigFromFile{HttpClient}$ Opening /mod/HTTP_Client_config.txt in read mode..
Set APN to: <<web.omnitel.it>>
Set APN USER to: <<>>
Set APN PASS to: <<>>
Set CID to: 3
Set CACERTFILE to: <<>>
Set CLIENTCERTFILE to: <<>>
Set CLIENTKEYFILE to: <<>>
Set REQUEST_TYPE to: 2
Set SERVER to: <<http://linux-ip.net>>
[DEBUG] 17.16  m2mb_HTTP_t:267 - activatePdP{HttpClient}$ m2mb_os_ev_init success
[DEBUG] 17.17  m2mb_HTTP_t:273 - activatePdP{HttpClient}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 17.18  m2mb_HTTP_t:281 - activatePdP{HttpClient}$ Waiting for registration...
[DEBUG] 17.19  m2mb_HTTP_t:101 - NetCallback{pubTspt_0}$ Module is registered to cell 0x468E!
[DEBUG] 17.20  m2mb_HTTP_t:295 - activatePdP{HttpClient}$ Pdp context initialization
[DEBUG] 19.26  m2mb_HTTP_t:305 - activatePdP{HttpClient}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 20.16  m2mb_HTTP_t:128 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 20.18  m2mb_HTTP_t:131 - PdpCallback{pubTspt_0}$ IP address: 176.243.212.114
Performing a GET request...
Connecting to linux-ip.net:80//

Socket connected!

<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="author" content="Martin A. Brown" />
    <meta name="robots" content="index, follow"/>

    <meta property="og:title" content="http://linux-ip.net/"/>
    <meta property="og:url" content="http://linux-ip.net"/>
    <meta property="og:site_name" content="http://linux-ip.net/"/>
    <meta property="og:type" content="website"/>

    <link rel="canonical" href="http://linux-ip.net" />

    <title>http://linux-ip.net/</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" type="text/css" href="//netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.css" />
    <link rel="stylesheet" type="text/css" href="//netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css" />
    <link rel="stylesheet" type="text/css" href="http://linux-ip.net/theme/css/main.css" />

</head>
   ▪▪▪
    <footer id="site-footer">
        <div class="row-fluid">
            <div class="span10 offset1">
                <address>
                    <p>
                        Powered by <a href="http://getpelican.com/">Pelican</a>
                        and <a href="http://python.org">Python</a>.
                        Theme based on <a href="http://github.com/jsliang/pelican-fresh/">Fresh</a>
                        by <a href="http://jsliang.com/">jsliang</a>
                    </p>
                </address>
            </div>
        </div>
    </footer>
</body>
</html>
Result: 200
[DEBUG] 26.80  m2mb_HTTP_t:137 - PdpCallback{pubTspt_0}$ Context deactivated!
```

Figure 27

3.4.20 HW Timer (Hardware Timer)

The sample application shows how to use HW Timers M2MB API. Debug prints on **AUX UART**

Features

- How to open configure a HW timer
- How to use the timer to manage recurring events

3.4.20.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create hw timer structure
- Configure it with 100 ms timeout, periodic timer (auto fires when expires) and autostart
- Init the timer with the parameters
- Wait 10 seconds
- Stop the timer

`TimerCb`

- Print a message with an increasing counter

```
Starting HW Timers demo app. This is v1.0.7 built on Mar 26 2020 13:04:14.
[DEBUG] 14.06  M2MB_main.c:114 - M2MB_main{M2M_DamsStart}$ Set the timer attributes structure: success.
Timer successfully created
Start the timer, success.
[DEBUG] 14.18  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [0]
[DEBUG] 14.28  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [1]
[DEBUG] 14.38  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [2]
[DEBUG] 14.48  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [3]
[DEBUG] 14.58  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [4]
[DEBUG] 14.69  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [5]
[DEBUG] 14.79  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [6]
[DEBUG] 14.88  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [7]
[DEBUG] 14.98  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [8]
[DEBUG] 15.08  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [9]


[DEBUG] 23.90  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [96]
[DEBUG] 24.01  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [97]
[DEBUG] 24.11  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [98]
Stop a running timer: success
Application end
```

Figure 28

3.4.21 Hello World

The application prints “Hello World!” over selected output every two seconds. Debug prints on **AUX UART**, using AZX log example functions

Features

- How to open an output channel using AZX LOG sample functions
- How to print logging information on the channel using AZX LOG sample functions

3.4.21.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Print “Hello World!” every 2 seconds in a while loop

```
Starting. This is v1.0.7 built on Mar 26 2020 09:34:16. LEVEL: 2

 Start Hello world Application  [ version: 2.000000 ]

 Hello world 2.0 [ 000001 ]
 Hello world 2.0 [ 000002 ]
 Hello world 2.0 [ 000003 ]
 Hello world 2.0 [ 000004 ]
 Hello world 2.0 [ 000005 ]
 Hello world 2.0 [ 000006 ]
 Hello world 2.0 [ 000007 ]
 Hello world 2.0 [ 000008 ]
 Hello world 2.0 [ 000009 ]
```

Figure 29

3.4.22 I2C example

Sample application showing how to communicate with an I2C slave device. Debug prints on **AUX UART**

Features

- How to open a communication channel with an I2C slave device
- How to send and receive data to/from the slave device

Setup

- Connect sensor VDD to 1v8 supply (e.g. Vaux/PwrMon pin of the module)
- Connect sensor GND to a GND pin of the module
- Connect sensor SDA to module GPIO2
- Connect sensor SCL to module GPIO3

3.4.22.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open I2C bus, setting SDA an SCL pins as 2 and 3 respectively
- Set registers to configure accelerometer -Read in a loop the 6 registers carrying the 3 axes values and show the g value for each of them

```
Starting I2C demo app. This is v1.0.7 built on Mar 26 2020 16:50:40.

Configuring the Kionix device...
opening channel /dev/I2C-30
[DEBUG] 20.18  M2MB_main.c:218 - test_I2C{M2M_DamsStart}$ -
WHOAMI content: 0x01
Configuring I2C Registers - Writing 0x4D into 0x1D register (CTRL_REG3)...
Write: success

I2C reading data from 0x1D register (CTRL_REG3)...
Read: success.
Accelerometer Enabled. ODR tilt: 12.5Hz, ODR directional tap: 400Hz, ORD Motion Wakeup: 50Hz
Configuring I2C Registers - Writing 0xC0 into 0x1B register (CTRL_REG1)...
Write: success

I2C reading data from 0x1B register (CTRL_REG1)...
Read: success.
Accelerometer Enabled. Operative mode, 12bit resolution
I2C read axes registers
------------
Reading Success.

X: -0.050 g
Y: -0.046 g
Z: 1.006 g
Reading Success.

X: -0.049 g
Y: -0.044 g
Z: 1.004 g
Reading Success.

X: -0.052 g
Y: -0.044 g
Z: 1.007 g
Reading Success.

X: -0.048 g
Y: -0.045 g
Z: 1.005 g
```

Figure 30

3.4.23 I2C Combined

Sample application showing how to communicate with an I2C slave device with I2C raw mode. Debug prints on MAIN UART

Features

- How to open a communication channel with an I2C slave device
- How to send and receive data to/from the slave device using raw mode API

Setup

- Connect sensor VDD to 1v8 supply (e.g. Vaux/PwrMon pin of the module)
- Connect sensor GND to a GND pin of the module
- Connect sensor SDA to module GPIO2
- Connect sensor SCL to module GPIO3

3.4.23.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Open I2C bus, setting SDA an SCL pins as 2 and 3 respectively
- Set registers to configure accelerometer -Read in a loop the 6 registers carrying the 3 axes values and show the g value for each of them

```
Starting I2C raw demo app. This is v1.0.13-C1 built on Jul 30 2020 11:28:18.

Configuring the I2C device...
Opening I2C channel /dev/I2C-30 ( device address is 0x0F << 1 )
Accelerometer Enabled. ODR tilt: 12.5Hz, ODR directional tap: 400Hz, ORD Motion Wakeup: 50Hz
Accelerometer Enabled. Operative mode, 12bit resolution
I2C read axes registers
------------

X: 0.000 g
Y: 0.000 g
Z: 0.000 g

X: -0.270 g
Y: 0.016 g
Z: 0.917 g

X: -0.268 g
Y: 0.013 g
Z: 0.925 g

X: -0.271 g
Y: 0.015 g
Z: 0.922 g

X: -0.267 g
Y: 0.016 g
Z: 0.918 g

X: -0.274 g
Y: 0.019 g
Z: 0.915 g
```

Figure 31

3.4.24 LWM2M

Sample application showcasing LWM2M client usage with M2MB API. Debug prints on **AUX UART**

Features

- Configure LWM2M client and enable it
- Create an instance of a custom object
- Set an integer value on a read only resource
- Set two integer values on a multi-instance read only resource
- write a string on a read/write resource
- Manage exec requests from the portal
- Manage write, read and monitoring resources changed from the portal

Requirements

This application expects the user to configure the PDP context ID 1 with the proper APN. it can be done with the following AT command:

```
AT+CGDCONT=1,"IPV4V6","<user apn>"
```

Depending on the Mobiler Network Operator and Access Technology, the APN might be automatically set by the network itself. In this case, nothing must be done by the user.

3.4.24.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the LWM2M client and start it

`lwm2m_demo.c`

`msgLWM2MTask` - Check registration status

- Initialize LWM2M client,
- Check for XML file fo custom object
- Enable unsolicited messages from client
- Create a task (lwm2m_taskCB is its callback function)to manage events from Portal
- Enable LwM2M client

- Create a new instance for the custom object
- Wait for client to register to Portal
- Performs all operations (set, read, get, write) on the related resources
- Performs a set with notify ack enabled
- Wait for events from server

`lwm2mIndicationCB`

- Manage events arriving from client (operations completion status and unsolicited events)
- Run lwm2m_taskCB when a monitored resource changes, to manage the action to be done

3.4.24.2 Device Profile upload

LWM2M resources demo device profile must be imported to have a real-time update of resources values on the LWM2M browser.

To do so, import the file `json/lwm2m_resources_demo.json` (provided with the sample files) on section `Developer` > `Device profiles` of OneEdge IoT portal:

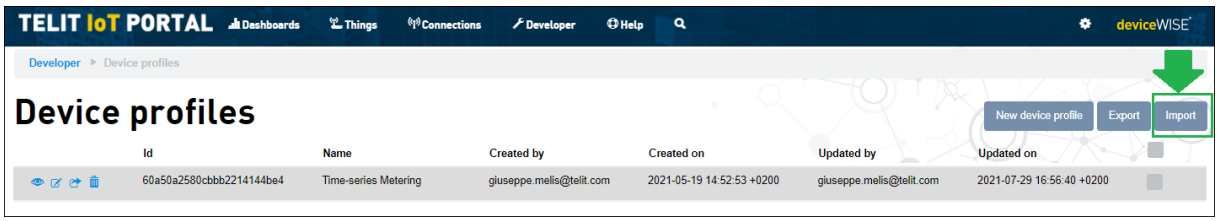

Figure 32

3.4.24.3 Custom Object configuration

The XML file content must be loaded on the Telit IoT Portal for the demo application to be fully executed.

First, enter Developer section from the top menu

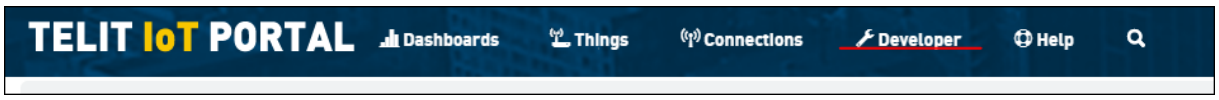

Figure 33

Choose Object Registry

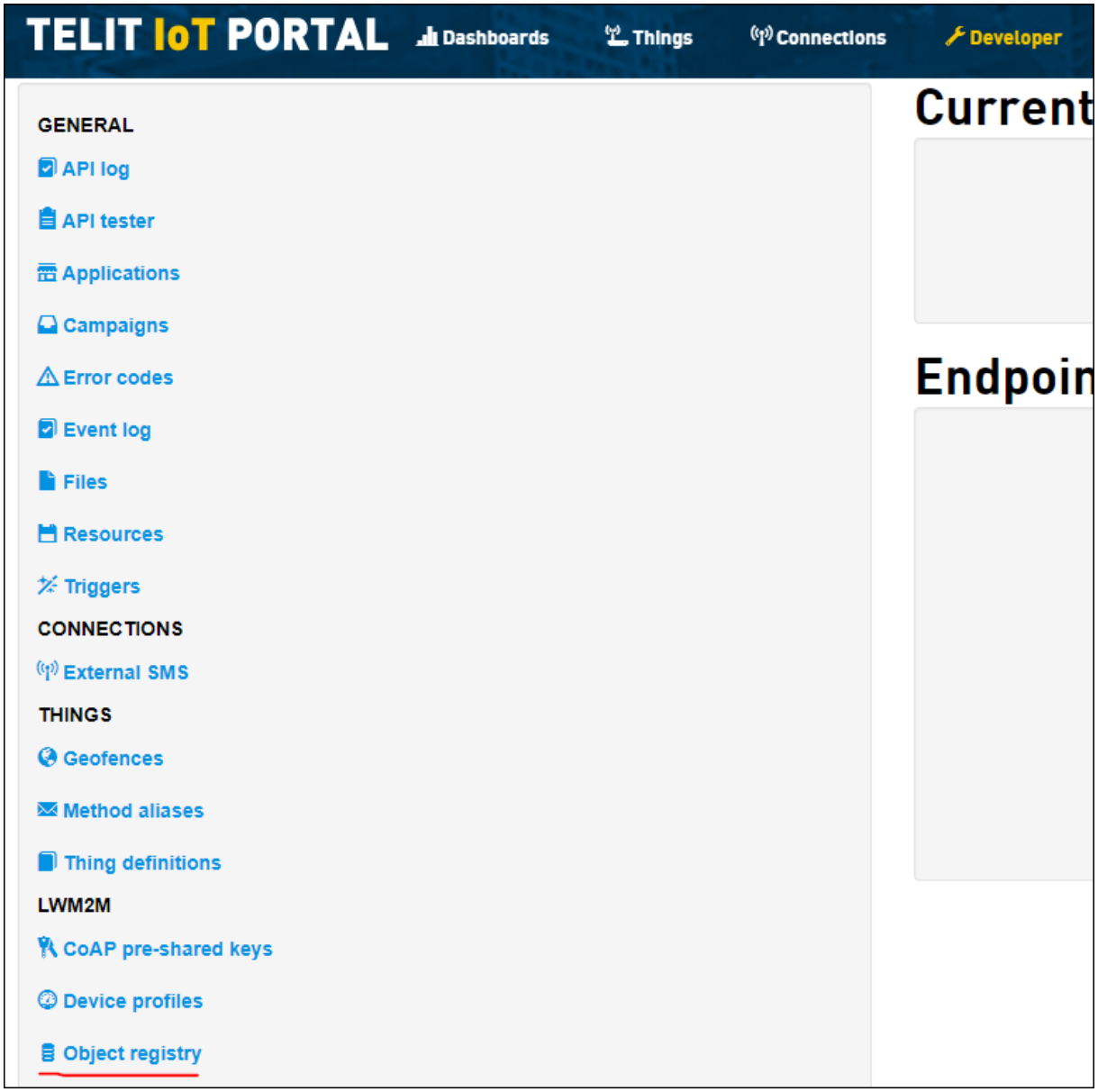

Figure 34

Create a New Object

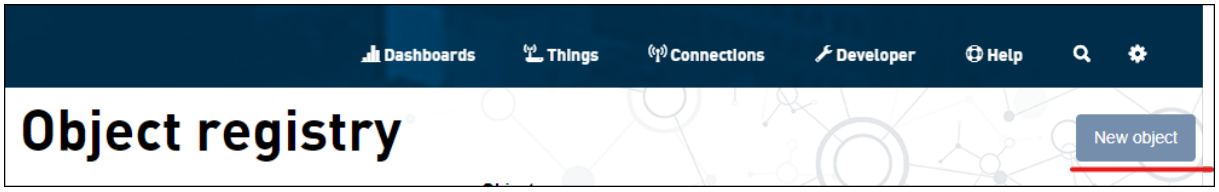

Figure 35

Copy the xml file content and paste it in the new Object form

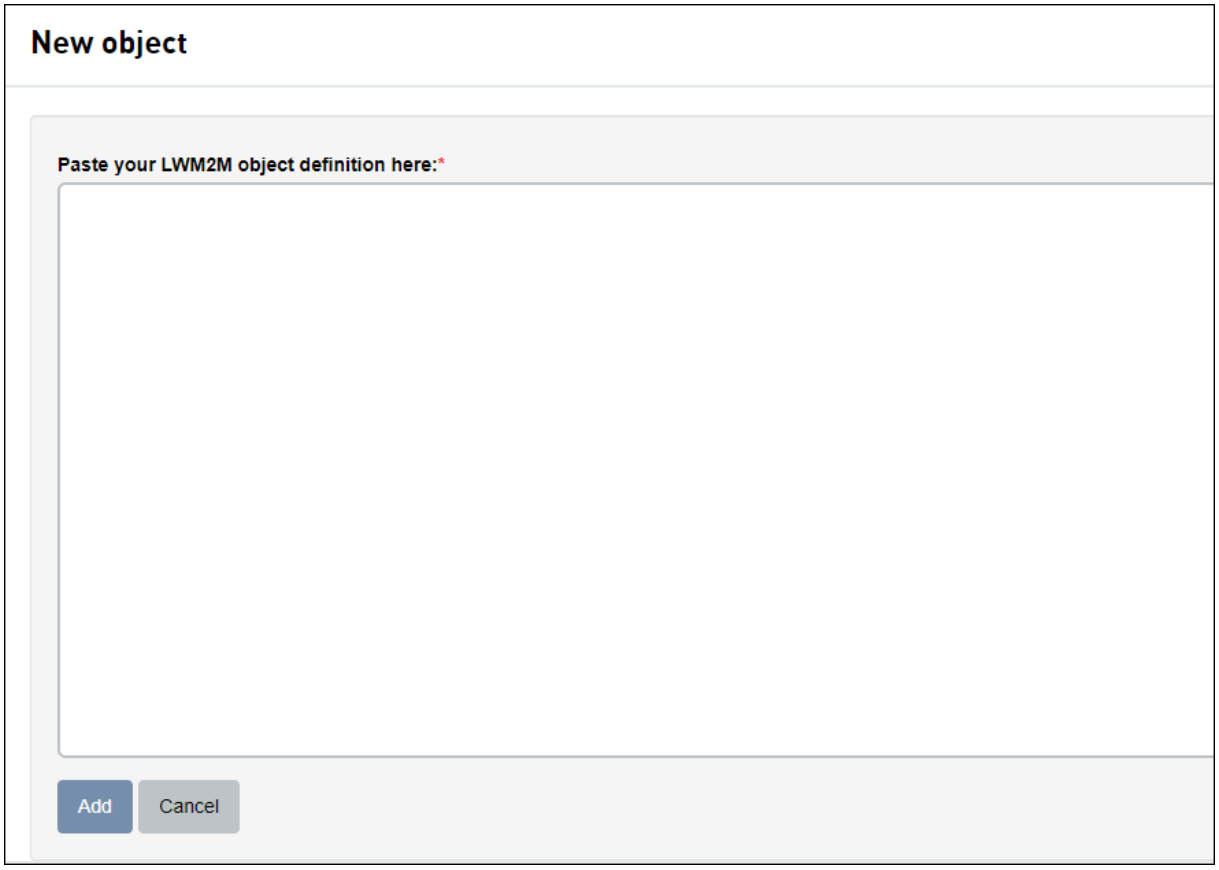

Figure 36

Also, the application requires the XML file `xml/object_32010.xml` (provided with the sample files) to be stored in module's `/XML/` folder. It can be done with

`AT#M2MWRITE=/XML/object_32010.xml,<size_in_bytes>`

To load the XML file in the module, Telit AT Controller (TATC) can be used. Once the command above is issued, press the load content button:

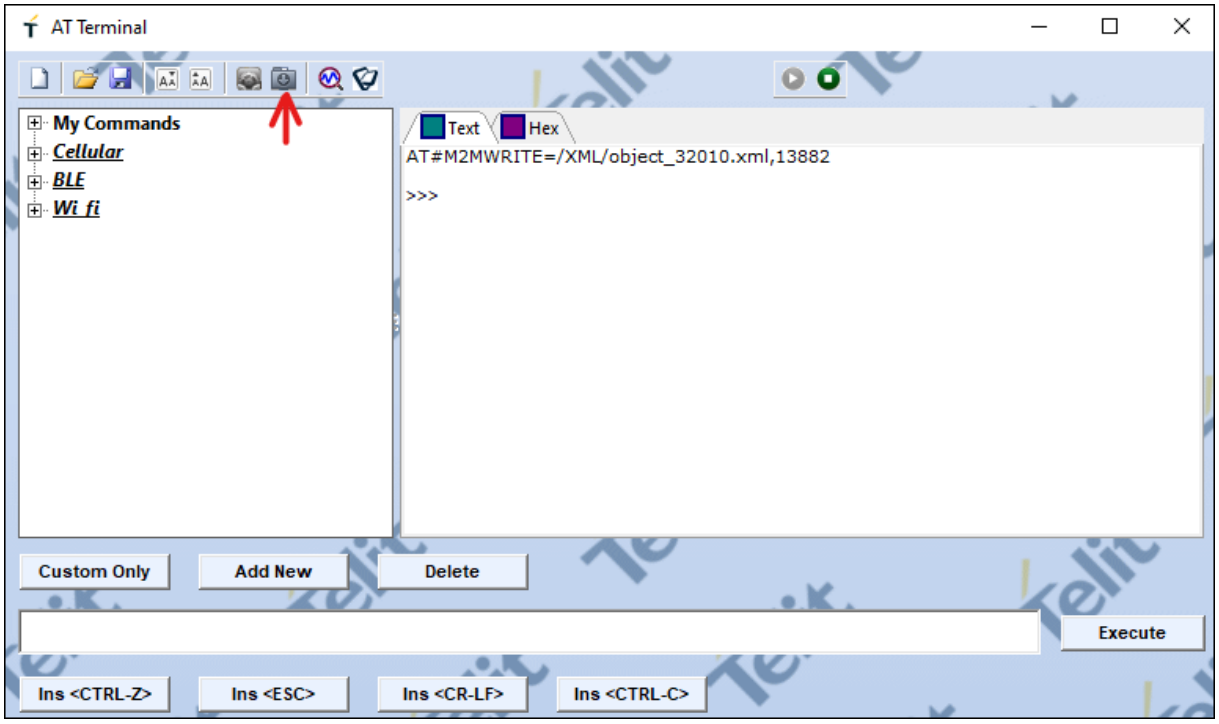

Figure 37

Select the file from your computer

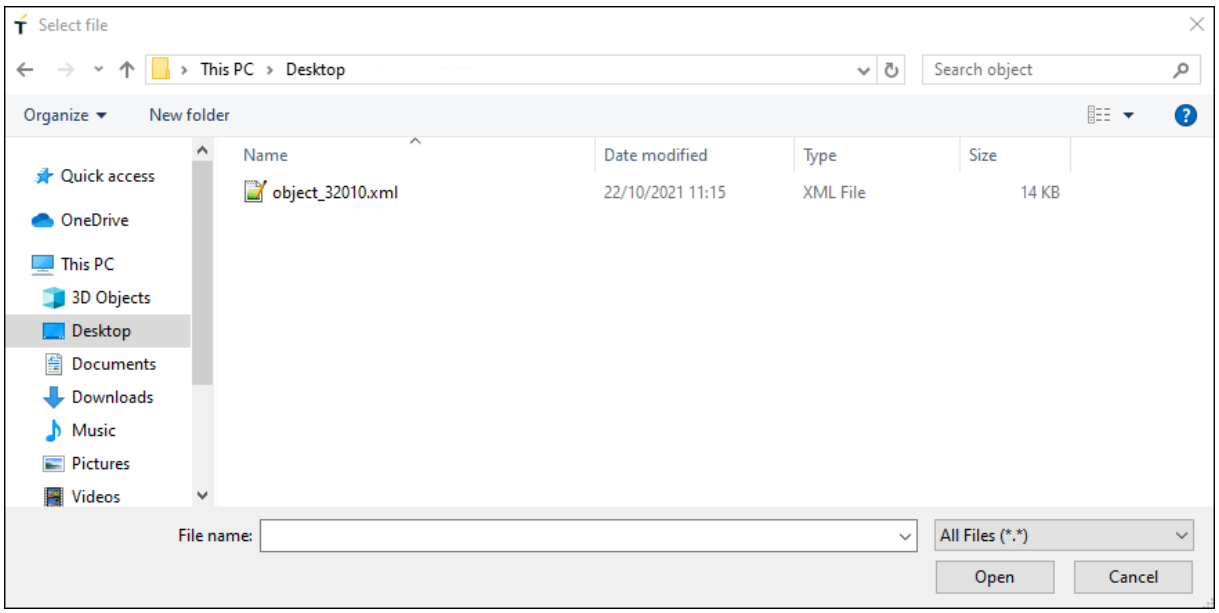

Figure 38

The file is successfully loaded on the module

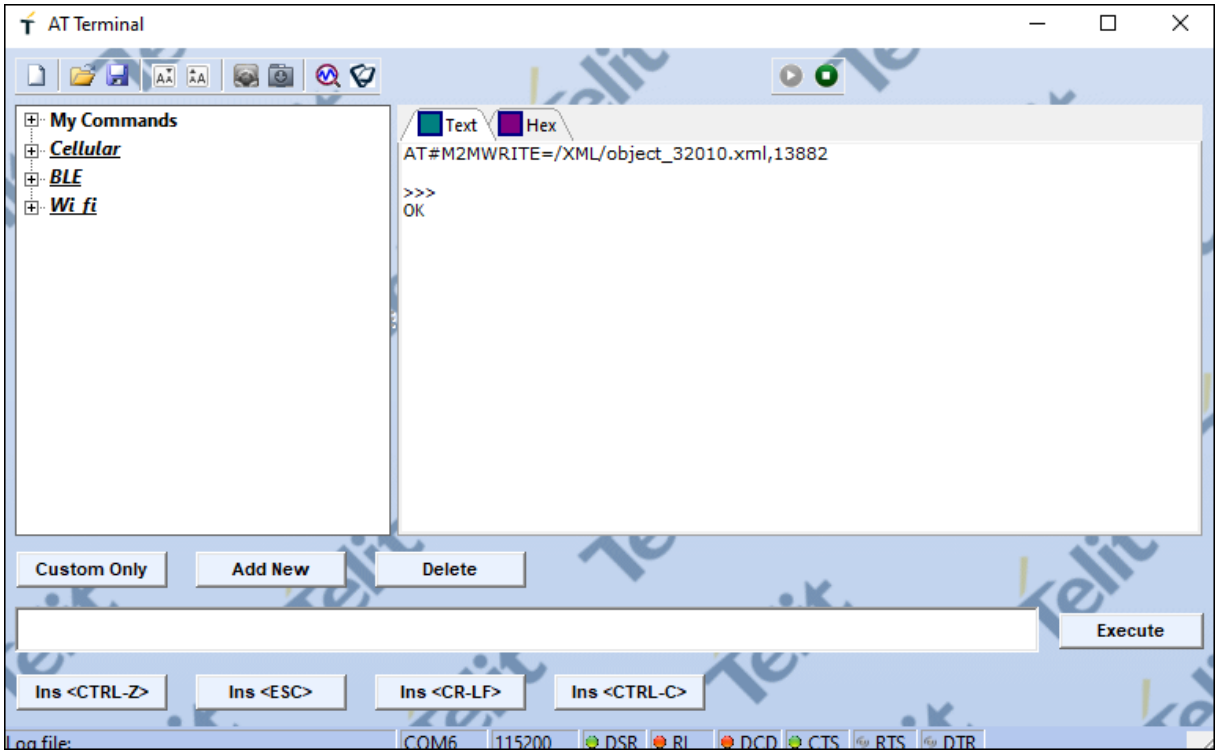

Figure 39

3.4.24.4 Onboard the device

Get the Telit ID

To retrieve the Telit ID data, issue AT#TID to get the Telit ID. The command response will be similar to

#TID: **xxxxxxxxxxxxxxxxxxxxxxxxxxxx**,1 OK

Take note of the Telit ID highlighted in **bold** above (or copy it on a text editor): this ID it will be needed for the onboarding process.

Create a new Thing

From the OneEdge portal, on **"Things"** section, click **"New Thing"** button in the top right corner.

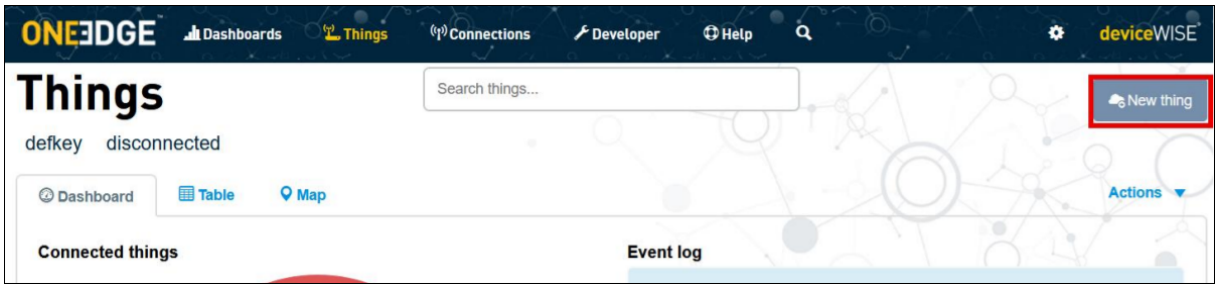

Figure 40

In the Create a new thing dialog, select “Telit Module”

Figure 41

A dialog appears: select “Default” thing definition

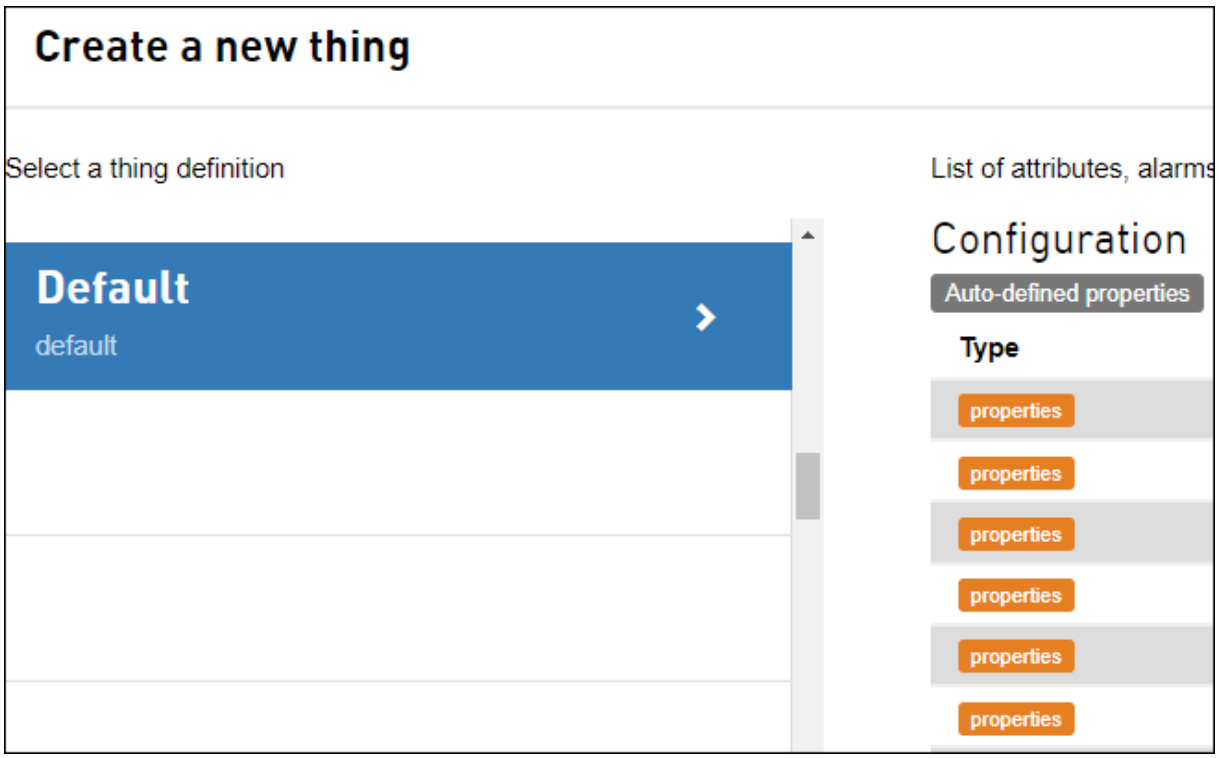

Figure 42

In the following screen, provide the Telit ID as “Identifier” Click on “Find” and make sure that model, firmware and the other details are properly populated.

Click on lwm2m tab and set the device profile previously imported as shown in the

screenshot below

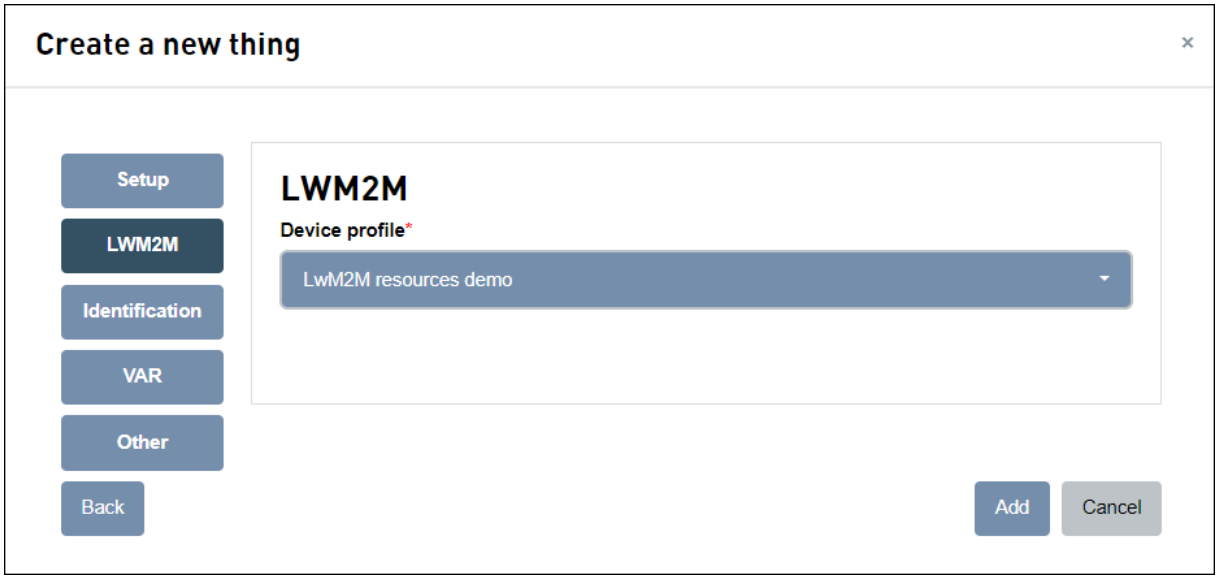

Figure 43

Click **"Add"** to complete the new thing creation procedure.

If the Thing already exists, its device profile can be changed by following the steps shown in the picture below

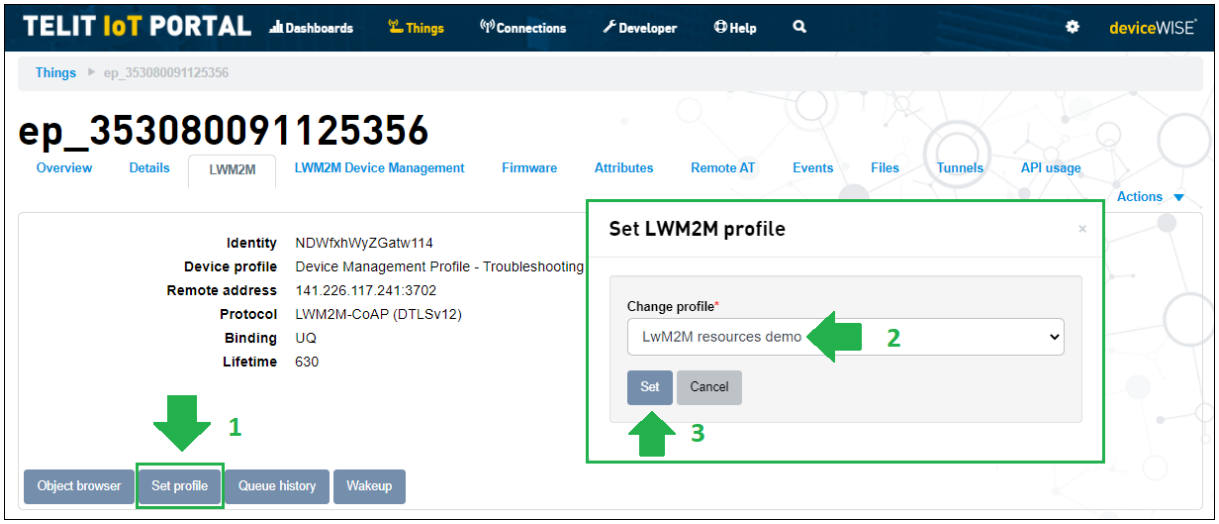

Figure 44

3.4.24.5 Application execution example

```
Starting lwm2m demo. This is v1.1.10 built on Oct 22 2021 10:49:52.
On OneEdge portal, be sure that the demo Device Profile has been uploaded on the IoT Portal
Initializing resources...

LWM2M enable result OK
[DEBUG] 38.20  lwm2m_demo:998 - lwm2mIndicationCB{pubTspt_0}$ Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED
GET STATUS.
IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED
```

Figure 45

```
=================================
READ-ONLY RESOURCES

=================================

Setting integer resource {32010/0/2} value to 50 on LWM2M client.

Reading integer resource {32010/0/2} value on LWM2M client.

Resource /32010/0/2/0 changed!

Reading integer resource {32010/0/2/0} value on LWM2M client.

---Integer value is now 50

Integer data in {32010/0/2/0} resource was updated to new value: 50

-------------------------------------------------

Setting integer resource {32010/0/22/0} value to 10 on LWM2M client.

Resource /32010/0/22/0 changed!

Reading integer resource {32010/0/22/0} value on LWM2M client.

Integer data in {32010/0/22/0} resource was updated to new value: 10

Setting integer resource {32010/0/22/1} value to 11 on LWM2M client.

Resource /32010/0/22/1 changed!

Reading integer resource {32010/0/22/1} value on LWM2M client.

Integer data in {32010/0/22/1} resource was updated to new value: 11

-------------------------------------------------

Setting double resource {32010/0/3} value to 20.500000 on LWM2M client.

Reading double resource {32010/0/3} value on LWM2M client.

Resource /32010/0/3/0 changed!

Reading double resource {32010/0/3/0} value  on LWM2M client.

---Double value is now 20.500000

Float data in {32010/0/3/0} resource was updated to new value: 20.500000

-------------------------------------------------

Setting boolean resource {32010/0/4} value to 1 on LWM2M client.

Reading boolean resource {32010/0/4} value on LWM2M client.

Resource /32010/0/4/0 changed!

Reading boolean resource {32010/0/4/0} value on LWM2M client.

---Boolean value is now true

Boolean data in {32010/0/4/0} resource was updated to new value: true

-------------------------------------------------
```

```
Current time in seconds from the epoch: 1634900084

Setting timestamp resource {32010/0/6} value to 1634900084 on LWM2M client.

Reading timestamp resource {32010/0/6} value on LWM2M client.

Resource /32010/0/6/0 changed!

Reading timestamp resource {32010/0/6/0} value on LWM2M client.

---Timestamp value is now 1634900084

Time data in {32010/0/6/0} resource was updated to new value: 1634900084 (2021/10/22T10:54:44+00:00)

--------------------------------------------------

Setting opaque resource {32010/0/5} on LWM2M client.

Resource /32010/0/5/0 changed!

Reading opaque resource {32010/0/5/0} on LWM2M client.

Opaque data in {32010/0/5/0} resource was updated to new content (10 bytes):
00 01 02 03 04 05 06 07 08 09

Reading opaque resource {32010/0/5} on LWM2M client.

---Opaque content is now (10 bytes): 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09

--------------------------------------------------

Setting string resource {32010/0/1} value to Hello World! on LWM2M client.

Reading string resource {32010/0/1} value on LWM2M client.

Resource /32010/0/1/0 changed!

Reading integer resource {32010/0/1/0} value on LWM2M client.

---String content is now: <Hello World!>

String data in {32010/0/1/0} resource was updated to new content: <Hello World!>

--------------------------------------------------
```

```
=================================
READ-WRITE RESOURCES
=================================

Writing integer resource {32010/0/12} value to 50 on LWM2M client.

Reading integer resource {32010/0/12} value on LWM2M client.

Resource /32010/0/12/0 changed!

Reading integer resource {32010/0/12/0} value on LWM2M client.

---Integer value is now 50

Integer data in {32010/0/12/0} resource was updated to new value: 50

--------------------------------------------------

Writing double resource {32010/0/13} value to 20.500000 on LWM2M client.

Reading double resource {32010/0/13} value on LWM2M client.

Resource /32010/0/13/0 changed!

Reading double resource {32010/0/13/0} value  on LWM2M client.

---Double value is now 20.500000

Float data in {32010/0/13/0} resource was updated to new value: 20.500000

--------------------------------------------------

Writing boolean resource {32010/0/14} value to 1 on LWM2M client.

Reading boolean resource {32010/0/14} value on LWM2M client.

Resource /32010/0/14/0 changed!

Reading boolean resource {32010/0/14/0} value on LWM2M client.

---Boolean value is now true

Boolean data in {32010/0/14/0} resource was updated to new value: true

--------------------------------------------------
```

```
Current time in seconds from the epoch: 1634900125

Writing timestamp resource {32010/0/16} value to 1634900125 on LWM2M client.

Reading timestamp resource {32010/0/16} value on LWM2M client.

Resource /32010/0/16/0 changed!

Reading timestamp resource {32010/0/16/0} value on LWM2M client.

---Timestamp value is now 1634900125

Time data in {32010/0/16/0} resource was updated to new value: 1634900125 (2021/10/22T10:55:25+00:00)

--------------------------------------------------

Writing opaque resource {32010/0/15} on LWM2M client.

Resource /32010/0/15/0 changed!

Reading opaque resource {32010/0/15/0} on LWM2M client.

Opaque data in {32010/0/15/0} resource was updated to new content (10 bytes):
00 01 02 03 04 05 06 07 08 09

Reading opaque resource {32010/0/15} on LWM2M client.

---Opaque content is now (10 bytes): 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09

--------------------------------------------------

Writing string resource {32010/0/11} value to <Hello World!> on LWM2M client.

Reading string resource {32010/0/11} value on LWM2M client.

Resource /32010/0/11/0 changed!

Reading integer resource {32010/0/11/0} value on LWM2M client.

---String content is now: <Hello World!>
```

```
==================================
WRITE-ONLY RESOURCES
==================================

Writing integer resource {32010/0/42} value to 50 on LWM2M client.

String data in {32010/0/11/0} resource was updated to new content: <Hello World!>

Getting integer resource {32010/0/42} valueon LWM2M client.

Resource /32010/0/42/0 changed!

Getting integer resource {32010/0/42/0} value on LWM2M client.

---Integer value is now 50

Integer data in {32010/0/42/0} resource was updated to new value: 50

--------------------------------------------------

Writing double resource {32010/0/43} value to 20.500000 on LWM2M client.

Getting double resource {32010/0/43} value on LWM2M client.

Resource /32010/0/43/0 changed!

Getting double resource {32010/0/43/0} value  on LWM2M client.

---Double value is now 20.500000

Float data in {32010/0/43/0} resource was updated to new value: 20.500000

--------------------------------------------------

Writing boolean resource {32010/0/44} value to 1 on LWM2M client.

Getting boolean resource {32010/0/44} value on LWM2M client.

Resource /32010/0/44/0 changed!

Getting boolean resource {32010/0/44/0} value on LWM2M client.

---Boolean value is now true

Boolean data in {32010/0/44/0} resource was updated to new value: true

--------------------------------------------------
```

```
Current time in seconds from the epoch: 1634900163

Writing timestamp resource {32010/0/46} value to 1634900163 on LWM2M client.

Getting timestamp resource {32010/0/46} value on LWM2M client.

Resource /32010/0/46/0 changed!

Getting timestamp resource {32010/0/46/0} value on LWM2M client.

---Timestamp value is now 1634900163

Time data in {32010/0/46/0} resource was updated to new value: 1634900163 (2021/10/22T10:56:03+00:00)

--------------------------------------------------

Writing opaque resource {32010/0/45} on LWM2M client.

Resource /32010/0/45/0 changed!

Getting opaque resource {32010/0/45/0} on LWM2M client.

Opaque data in {32010/0/45/0} resource was updated to new content (10 bytes):
00 01 02 03 04 05 06 07 08 09

Getting opaque resource {32010/0/45} on LWM2M client.

---Opaque content is now (10 bytes): 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09

--------------------------------------------------

Writing string resource {32010/0/41} value to <Hello World!> on LWM2M client.

Getting string resource {32010/0/41} value on LWM2M client.

Resource /32010/0/41/0 changed!

Getting integer resource {32010/0/41/0} value on LWM2M client.

---String content is now: <Hello World!>

String data in {32010/0/41/0} resource was updated to new content: <Hello World!>

-----------------------------------
Resources operations examples done.
-----------------------------------
```

```
-----------------------------------
Will perform a SET with notify ACK enabled.
-----------------------------------
Enable notify ack

Setting integer resource {32010/0/2} value to 60 on LWM2M client.

Resource /32010/0/2/0 changed!

Reading integer resource {32010/0/2/0} value on LWM2M client.

Integer data in {32010/0/2/0} resource was updated to new value: 60

ACK received from server!

Reading integer resource {32010/0/2} value on LWM2M client.

---Integer value is now 60

Disable notify ack
Done.

Waiting for events from the OneEdge portal. Please write on monitored resources or call an 'exec' one.
```

After the Demo completes the initialization, it is possible to access the object resources from the section Things: select your device, then “LwM2M” tab of OneEdge IoT portal:

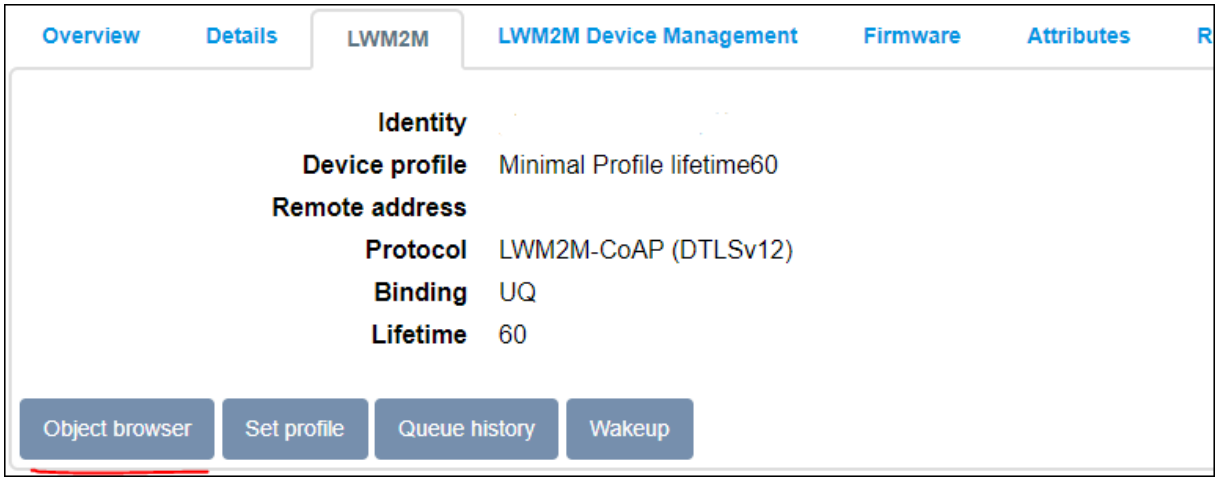

Figure 46

An instance of the object will be present and the resources can be modified.

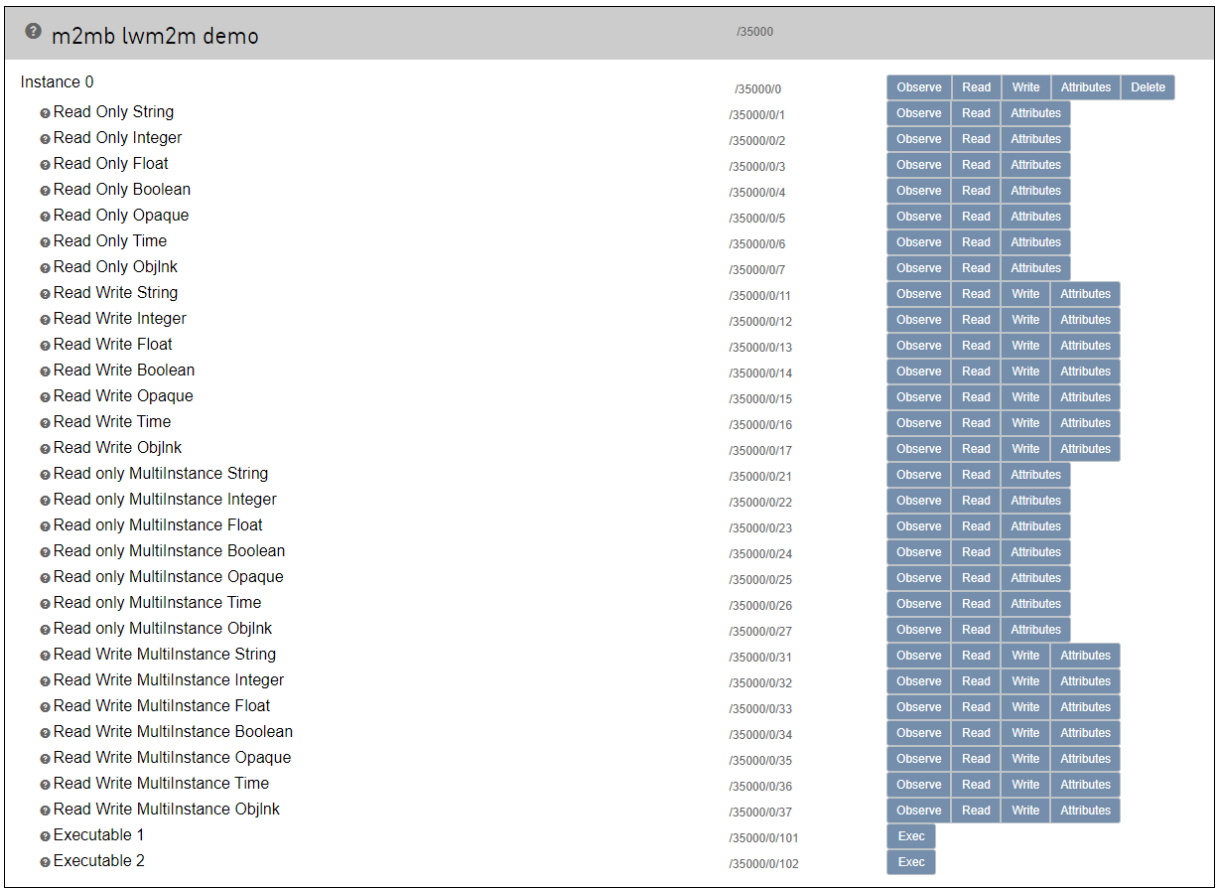

Figure 47

For example, executing the two Exec Resources at the bottom of the list, the application will react accordingly:

```
Info Exec Ind: 32010/0/101/0
Asked to execute resource 101

Resource /32010/0/11/0 changed!
```

Figure 48

Writing a string resource (id /32010/0/11), the application will notify the change

```
Reading integer resource {32010/0/11/0} value on LWM2M client.

String data in {32010/0/11/0} resource was updated to new content: <Hello from the IoT Portal!>
```

Figure 49

3.4.25 LWM2M FOTA ACK management (AT URCs)

Sample application showcasing LWM2M client FOTA events and ACKs management via AT URCs. Debug prints on **AUX UART**

Features

- Configure LWM2M client and enable it (if not already running)
- Enable monitoring on object 5 (Firmware Update)
- Configure lwm2m FOTA parameters to enable ACK requirement for each operation (Download, Update)
- Wait for an externally generated FOTA request to handle the steps by sending the acknoledgements when needed

Please refer to “80654NT11889A OneEdge Firmware Management Application Note” for further details about FOTA workflow. Get in touch with TS-OneEdge support to receive the document.

Requirements

This application expects the user to configure the PDP context ID 1 with the proper APN. it can be done with the following AT command:

```
AT+CGDCONT=1,"IPV4V6","<user apn>"
```

Depending on the Mobile Network Operator and Access Technology, the APN might be automatically set by the network itself. In this case, nothing must be done by the user.

3.4.25.1 Device Profile upload

Minimal FOTA profile (short lifetime) device profile must be imported and selected to improve the responsiveness of the FOTA operations

To do so, import the file `json/lwm2m_fota_profile_short.json` (provided with the sample files) on section `Developer` > `Device profiles` of OneEdge IoT portal:

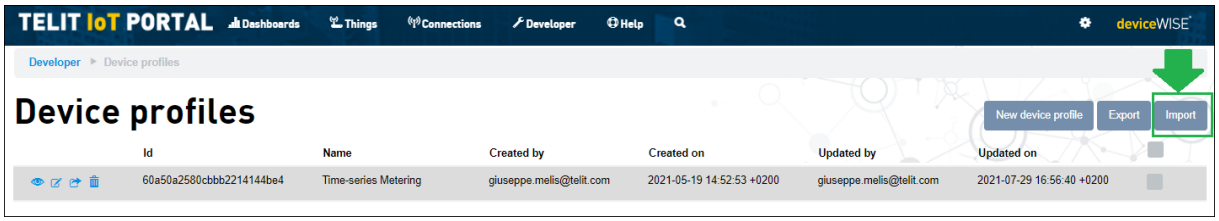

Figure 50

3.4.25.2 Onboard the device

Get the Telit ID

To retrieve the Telit ID data, issue AT#TID to get the Telit ID. The command response will be similar to

#TID: **xxxxxxxxxxxxxxxxxxxxxxxxxxxx**,1 OK

Take note of the Telit ID highlighted in **bold** above (or copy it on a text editor): this ID it will be needed for the onboarding process.

Create a new Thing

From the OneEdge portal, on **"Things"** section, click **"New Thing"** button in the top right corner.

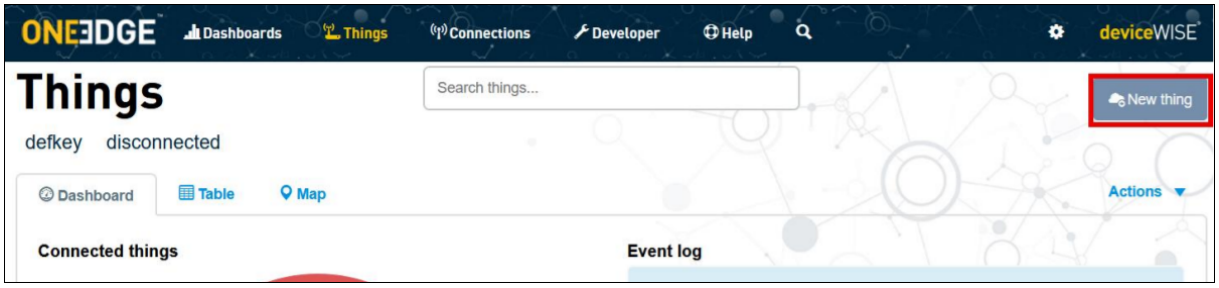

Figure 51

In the Create a new thing dialog, select "Telit Module"

Figure 52

A dialog appears: select "Default" thing definition

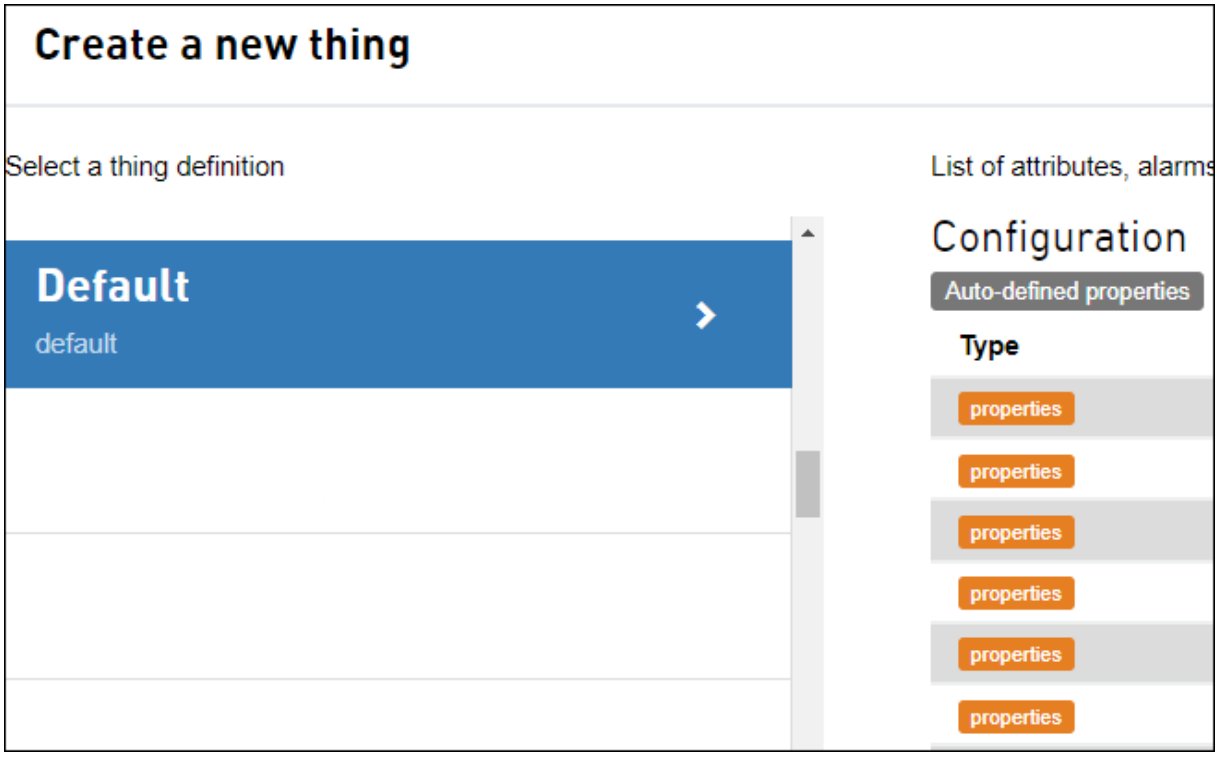

Figure 53

In the following screen, provide the Telit ID as “Identifier” Click on “Find” and make sure that model, firmware and the other details are properly populated.

Click on lwm2m tab and set the device profile previously imported as shown in the screenshot below

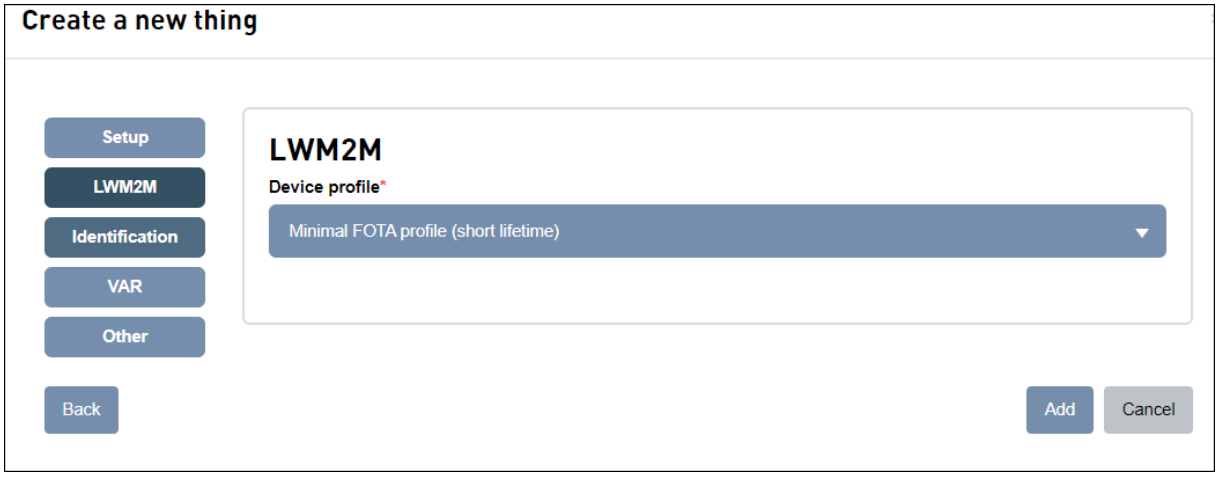

Figure 54

Click **“Add”** to complete the new thing creation procedure.

3.4.25.3 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Create a task to manage the LWM2M client and start it

lwm2m_demo.c

msgLWM2MTask

- Initialize LWM2M client,
- Enable unsolicited messages from client
- Enable FOTA ACK configuration
- Create a task (lwm2m_taskCB is its callback function)to manage events from Portal
- Check if LwM2M client is running, if not, enable it
- Wait for client to register to Portal
- Wait for FOTA events to arrive, and when the Update request arrives, notify the user that everything must stop before sending the last ACK (as the module will start updating and then reboot)

lwm2mIndicationCB

- Manage events arriving from client (operations completion status and unsolicited events)
- Run lwm2m_taskCB when a monitored resource changes, to manage the action to be done

3.4.25.4 Application execution example

```
Starting lwm2m FOTA management demo. This is v1.1.14 built on Jan 21 2022 14:39:47.
Initializing resources...
[DEBUG] 22.14  lwm2m_demo:283 - at_cmd_async_init{LWM2M_TASK}$ m2mb_ati_init() on instance 1

IF Status: M2MB_LWM2M_IF_STATE_DISABLED
Client Status: M2MB_LWM2M_CL_STATE_DISABLED
Enable lwm2m client
LWM2M enable result OK
Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED

IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED

Waiting for events from the OneEdge portal. Please start a FOTA process.
```

Figure 55

```
Resource /5/0/1/0 changed!
[DEBUG] 108.42  lwm2m_demo:544 - lwm2m_taskCB{LWM2M_TASK}$ Firmware update event!
FOTA process is waiting for Download ACK, sending it!
Waiting for Update ACK!
Before sending the ACK to apply FOTA, stop any ongoing operation (filesystem, sockets)

Asked to perform UPDATE step, stopping all operations now.
Application complete.

------- Info, DEVICE REBOOT FOR FOTA ------
????????
#OTAEV: Module Upgraded To New Fw
```

FOTA request arrives from server, ACK is sent for Download and Update. Device restarts with the new firmware version

```
Starting lwm2m FOTA management demo. This is v1.1.14 built on Jan 21 2022 14:39:47.
Initializing resources...
[DEBUG] 26.61  lwm2m_demo:283 - at_cmd_async_init{LWM2M_TASK}$ m2mb_ati_init() on instance 1
[WARN ] 34.98  lwm2m_demo:738 - lwm2mIndicationCB{pubTspt_0}$ write res 1

IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
Enable already done, skip. State: M2MB_LWM2M_IF_STATE_ACTIVE
Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED

IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED

Waiting for events from the OneEdge portal. Please start a FOTA process.
```

Application restarts and is ready for a new FOTA operation

3.4.26 LWM2M OBJ_GET AND OBJ_SET

Sample application showcasing LWM2M client m2mb_lwm2m_objget and m2mb_lwm2m_objset M2MB APIs usage. Debug prints on **AUX UART**

Features

- Configure LWM2M client and enable it
- Create an instance of a custom object
- Create a Json string
- Set string, integer, float, boolean, timestamp and opaque values with `m2mb_lwm2m_objset`
- Get all resources values with `m2mb_lwm2m_objget`
- Manage write, read and monitoring resources changed from the portal

Requirements

This application expects the user to configure the PDP context ID 1 with the proper APN. it can be done with the following AT command:

```
AT+CGDCONT=1,"IPV4V6","<user apn>"
```

Depending on the Mobiler Network Operator and Access Technology, the APN might be automatically set by the network itself. In this case, nothing must be done by the user.

3.4.26.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the LWM2M client and start it

`lwm2m_demo.c`

`msgLWM2MTask` - Check registration status

- Initialize LWM2M client,
- Check for XML file fo custom object
- Enable unsolicited messages from client
- Create a task (lwm2m_taskCB is its callback function)to manage events from Portal
- Enable LwM2M client

- Create a new instance for the custom object
- Wait for client to register to Portal
- Performs obj_set and obj_get operations on the related resources
- Wait for events from server

`lwm2mIndicationCB`

- Manage events arriving from client (operations completion status and unsolicited events)
- Run lwm2m_taskCB when a monitored resource changes, to manage the action to be done

3.4.26.2 Device Profile upload

LWM2M resources demo device profile must be imported to have a real-time update of resources values on the LWM2M browser.

To do so, import the file `json/lwm2m_resources_demo.json` (provided with the sample files) on section `Developer` > `Device profiles` of OneEdge IoT portal:

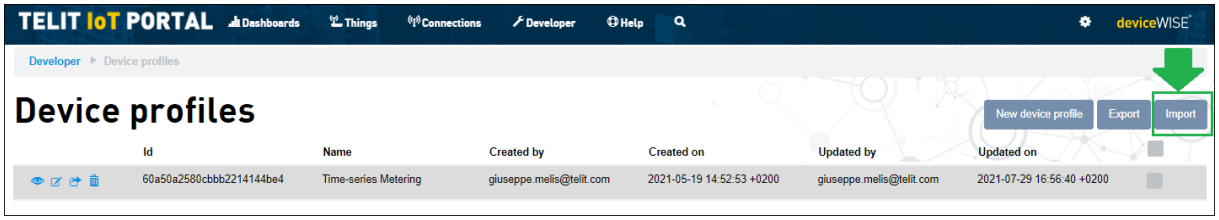

Figure 56

3.4.26.3 Custom Object configuration

The XML file content must be loaded on the Telit IoT Portal for the demo application to be fully executed.

First, enter Developer section from the top menu

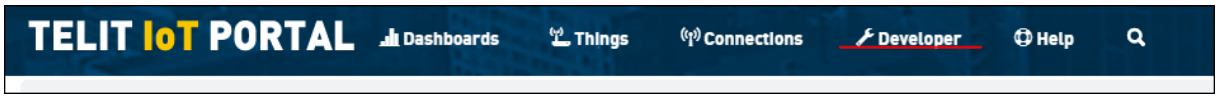

Figure 57

Choose Object Registry

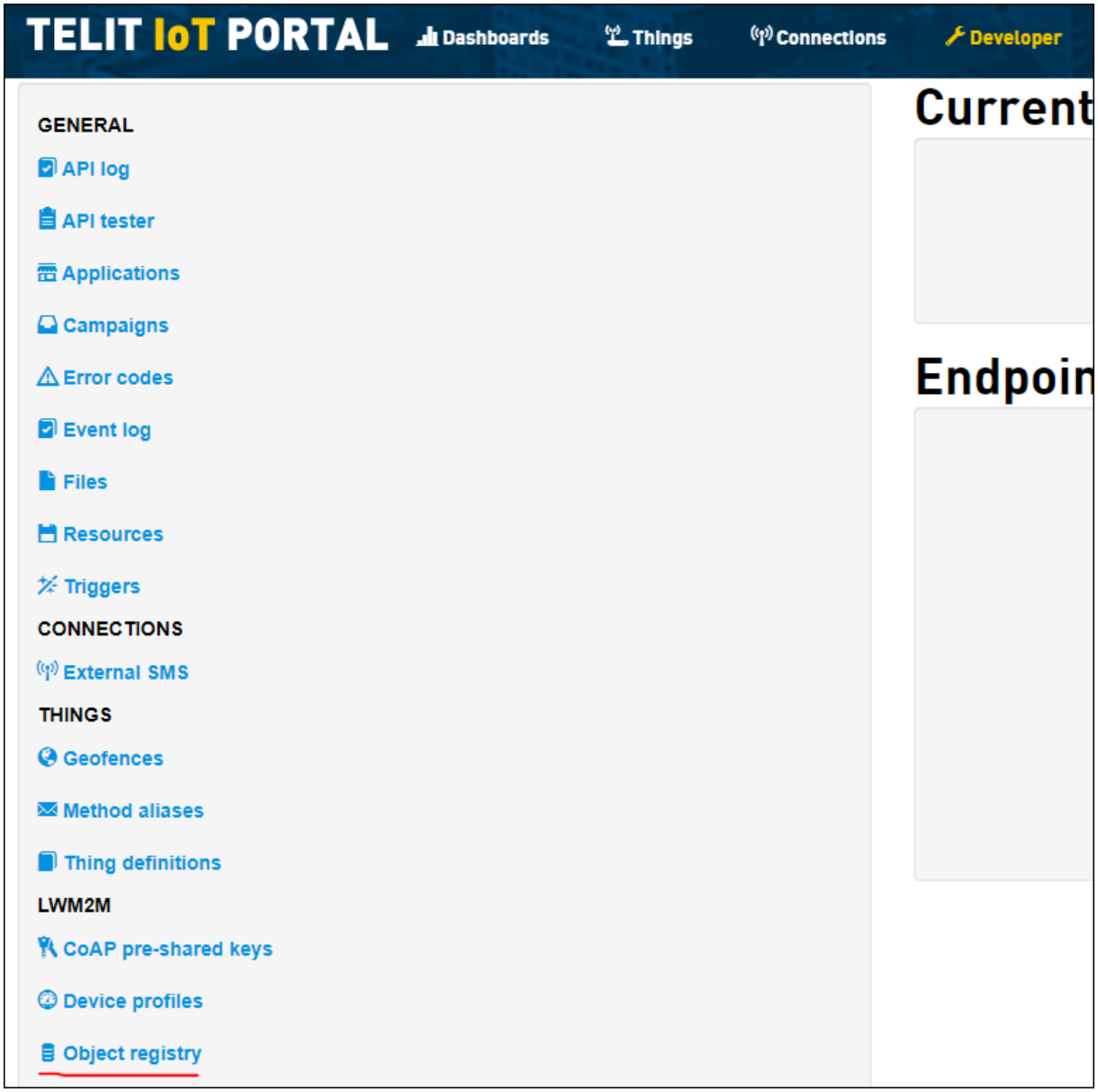

Figure 58

Create a New Object

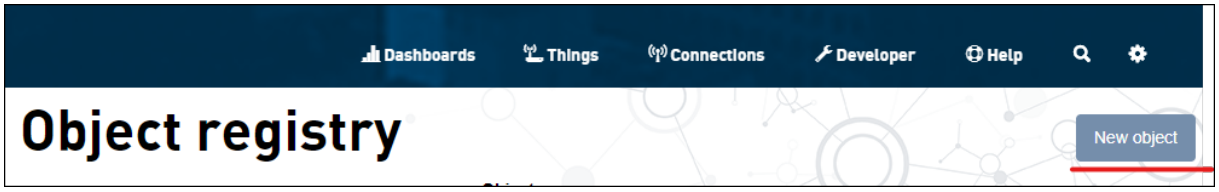

Figure 59

Copy the xml file content and paste it in the new Object form

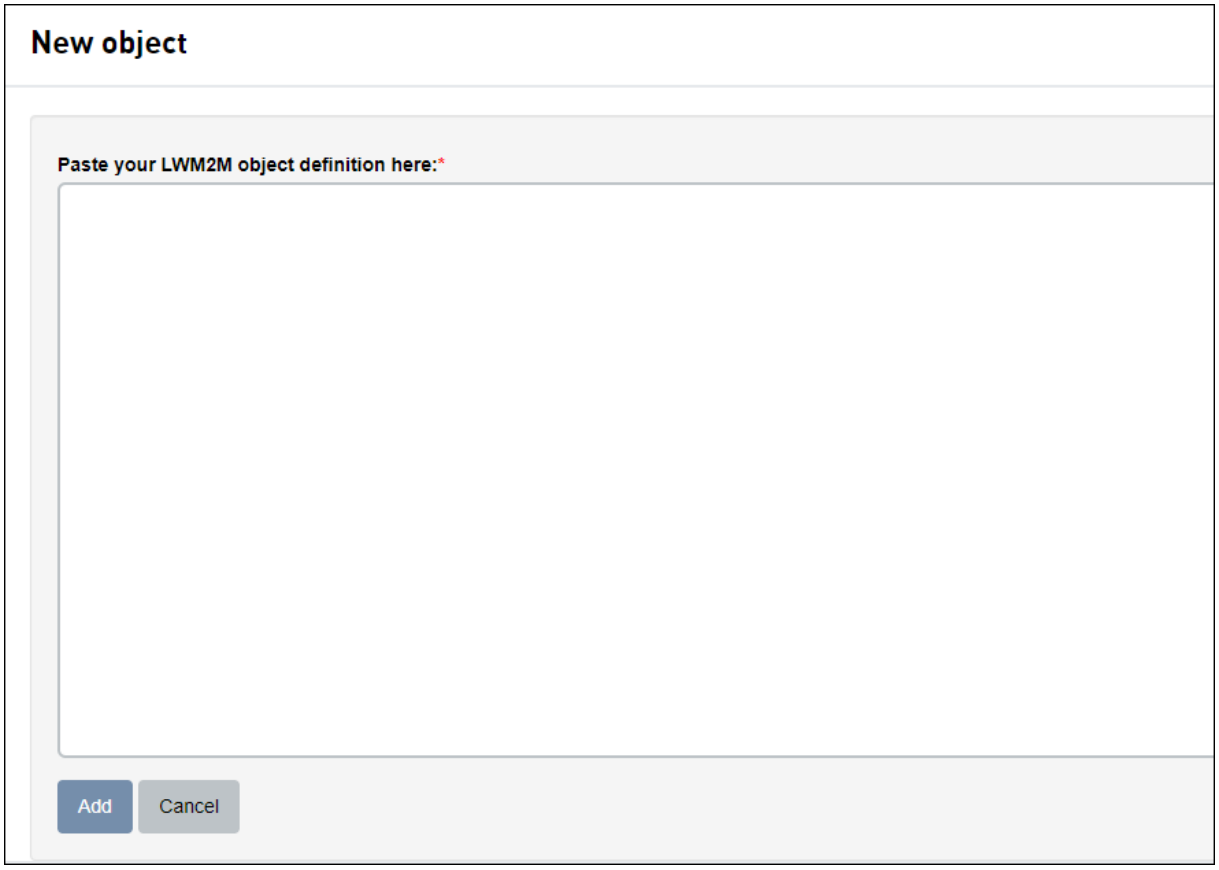

Figure 60

Also, the application requires the XML file `xml/object_32011.xml` (provided with the sample files) to be stored in module's `/XML/` folder. It can be done with

```
AT#M2MWRITE=/XML/object_32011.xml,<size_in_bytes>
```

To load the XML file in the module, Telit AT Controller (TATC) can be used. Once the command above is issued, press the load content button:

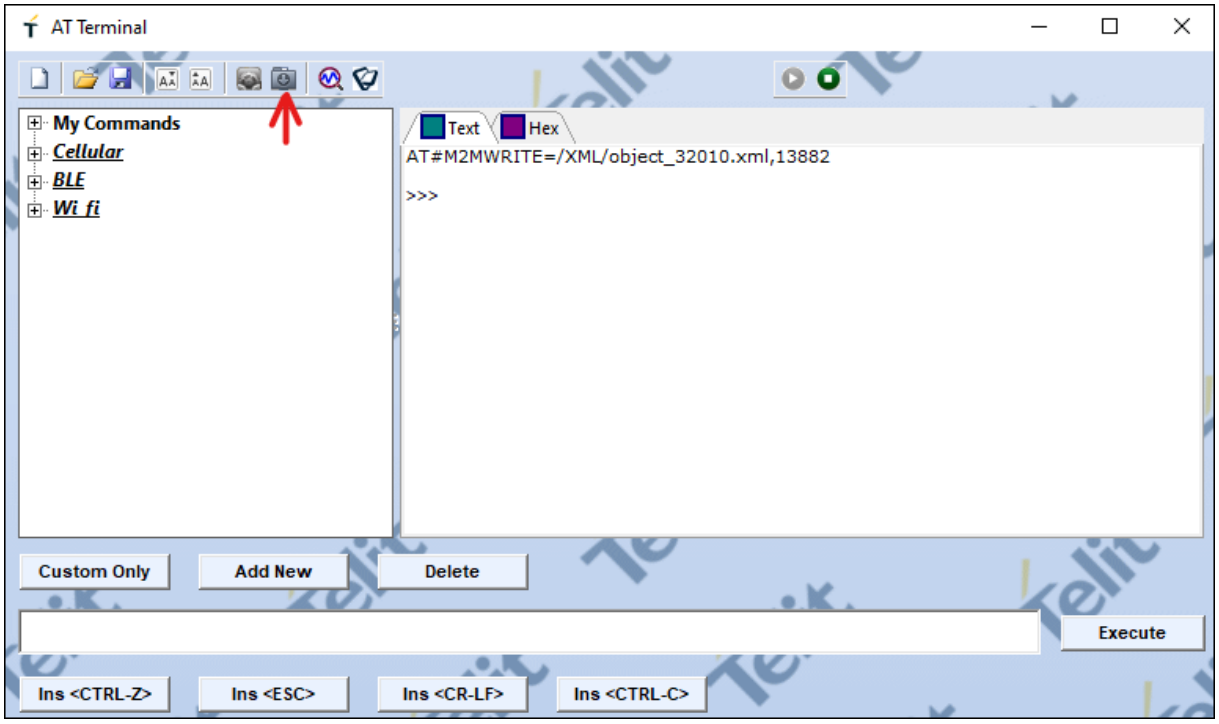

Figure 61

Select the file from your computer

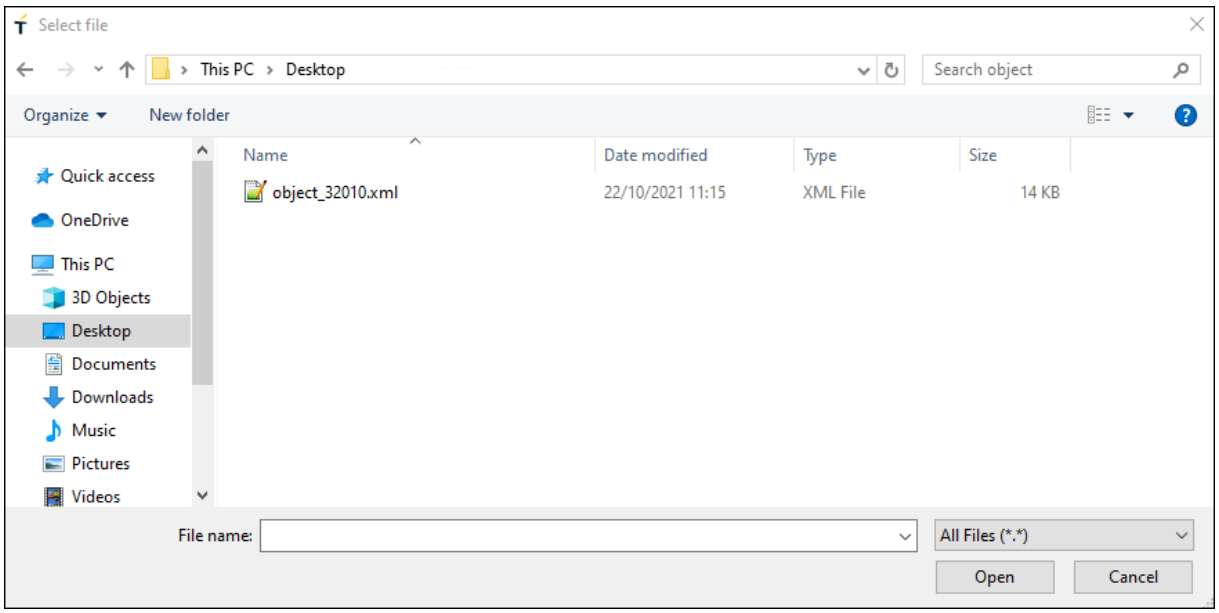

Figure 62

The file is successfully loaded on the module

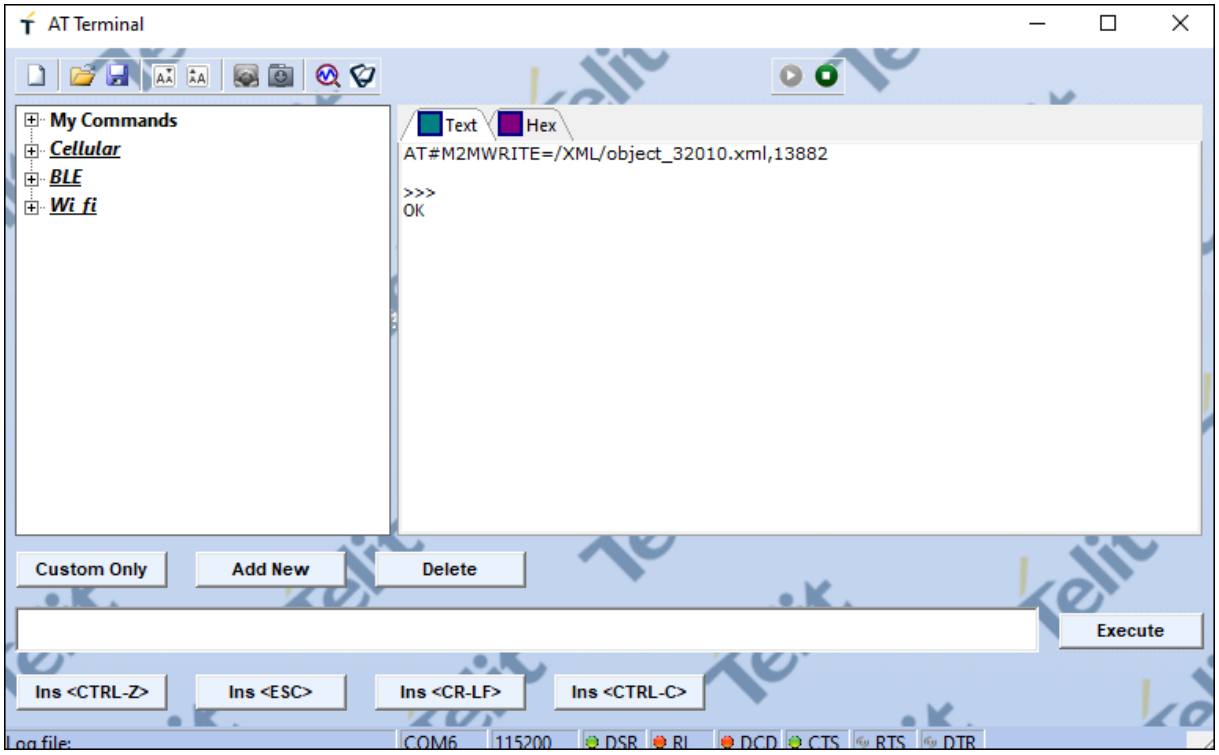

Figure 63

3.4.26.4 Onboard the device

Get the Telit ID

To retrieve the Telit ID data, issue AT#TID to get the Telit ID. The command response will be similar to

#TID: **xxxxxxxxxxxxxxxxxxxxxxxxxxxx**,1 OK

Take note of the Telit ID highlighted in **bold** above (or copy it on a text editor): this ID it will be needed for the onboarding process.

Create a new Thing

From the OneEdge portal, on **"Things"** section, click **"New Thing"** button in the top right corner.

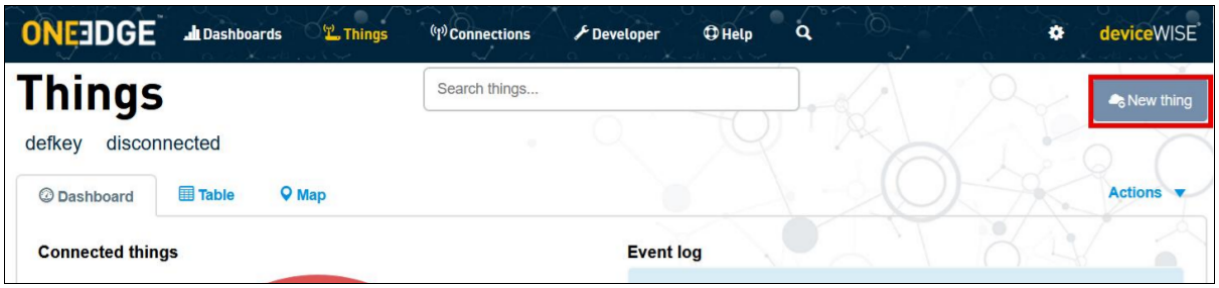

Figure 64

In the Create a new thing dialog, select “Telit Module”

Figure 65

A dialog appears: select “Default” thing definition

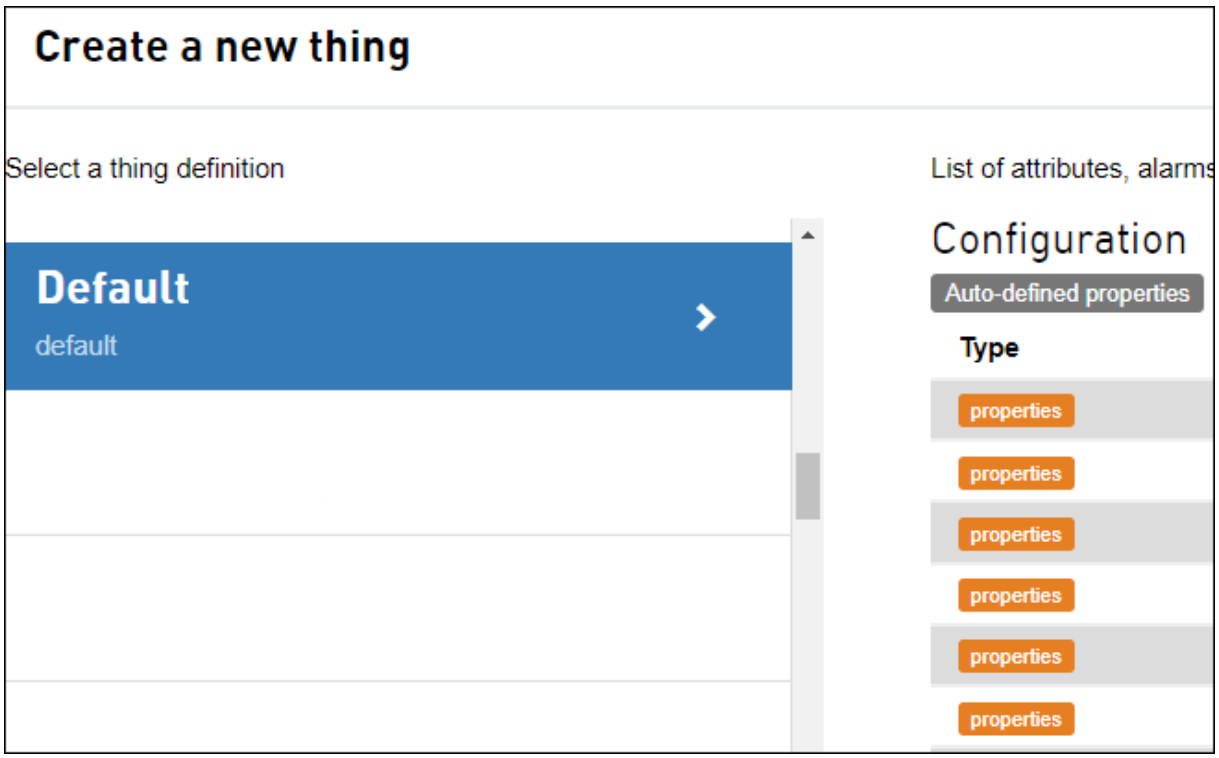

Figure 66

In the following screen, provide the Telit ID as “Identifier” Click on “Find” and make sure that model, firmware and the other details are properly populated.

Click on lwm2m tab and set the device profile previously imported as shown in the

screenshot below

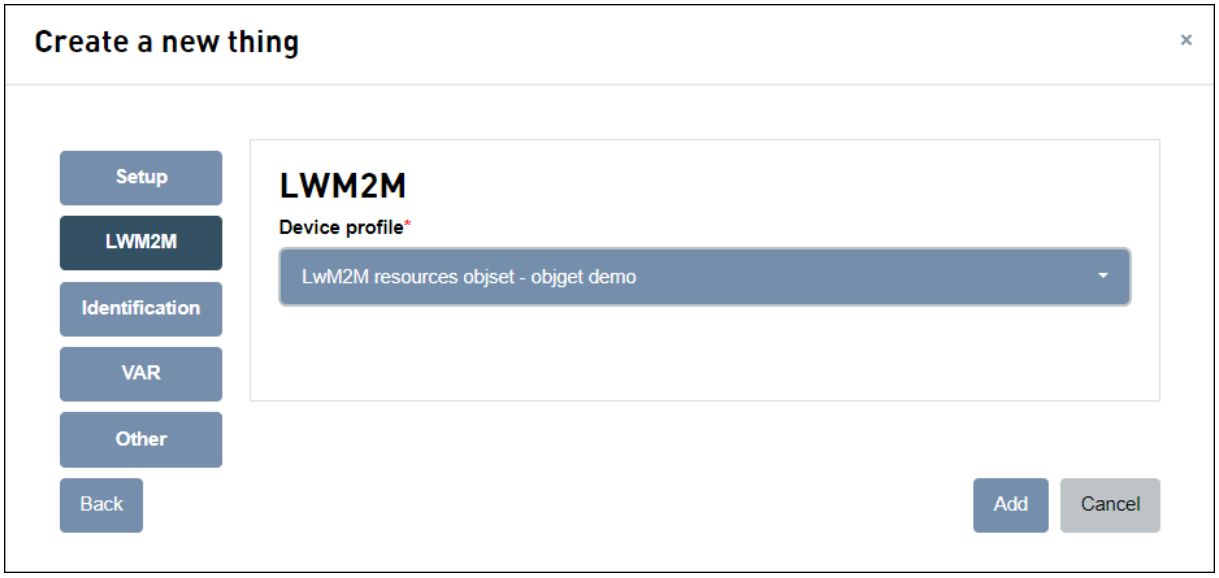

Figure 67

Click **"Add"** to complete the new thing creation procedure.

If the Thing already exists, its device profile can be changed by following the steps shown in the picture below

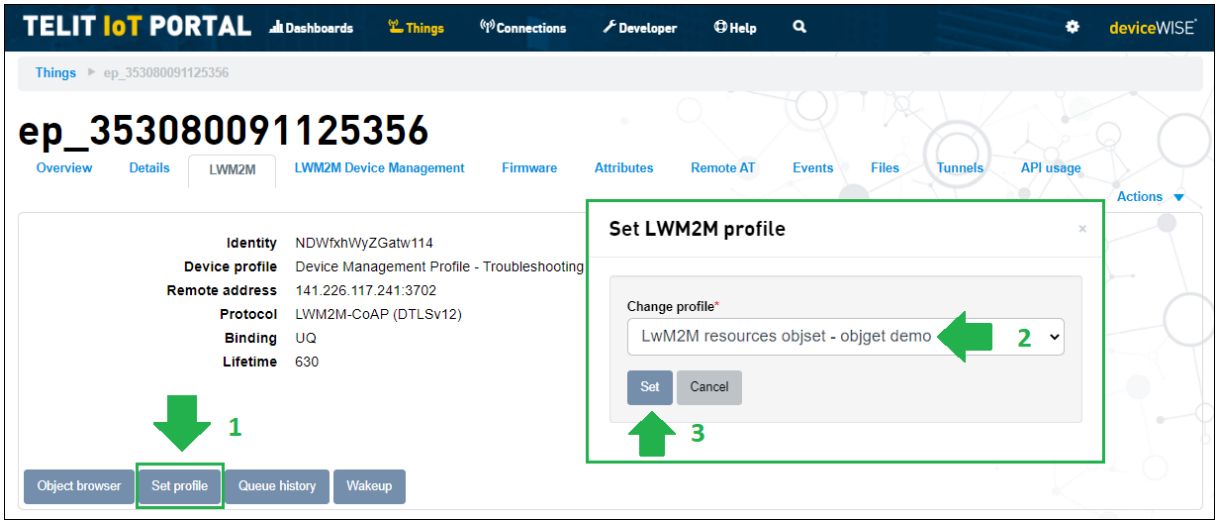

Figure 68

3.4.26.5 Application execution example

```
Starting lwm2m demo. This is v1.1.10 built on Oct 22 2021 10:49:52.
On OneEdge portal, be sure that the demo Device Profile has been uploaded on the IoT Portal
Initializing resources...

LWM2M enable result OK
[DEBUG] 38.20  lwm2m_demo:998 - lwm2mIndicationCB{pubTspt_0}$ Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED
GET STATUS.
IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED
```

Figure 69

```
Starting lwm2m demo. This is v1.1.11 built on Nov 24 2021 13:32:52.
On OneEdge portal, be sure that the 'lwm2m resources objset - objget demo' Device Profile has been uploaded on the IoT Portal

Initializing resources...
LWM2M enable result OK
[DEBUG] 29.42  lwm2m_demo:593 - lwm2mIndicationCB{pubTspt_0}$ Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED
GET STATUS.
IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED

m2mb_lwm2m_objset SUCCESS

Resource /32011/0/11/0 changed!

Reading string resource {32011/0/11/0} value on LWM2M client.

Resource /32011/0/12/0 changed!

Resource /32011/0/13/0 changed!

Resource /32011/0/15/0 changed!

Resource /32011/0/16/0 changed!

Resource /32011/0/31/0 changed!

Resource /32011/0/31/1 changed!

Resource /32011/0/31/2 changed!

Resource /32011/0/34/0 changed!

Resource /32011/0/34/1 changed!

Resource /32011/0/34/2 changed!

Resource /32011/0/34/3 changed!
```

```
String data in {32011/0/11/0} resource was updated to new content: <hello world>

Reading integer resource {32011/0/12/0} value on LWM2M client.

Integer data in {32011/0/12/0} resource was updated to new value: 70

Reading double resource {32011/0/13/0} value  on LWM2M client.

Float data in {32011/0/13/0} resource was updated to new value: -12.500000

Reading opaque resource {32011/0/15/0} on LWM2M client.

Opaque data in {32011/0/15/0} resource was updated to new content (11 bytes):
68 65 6C 6C 6F 20 77 6F 72 6C 64

Reading timestamp resource {32011/0/16/0} value on LWM2M client.

Time data in {32011/0/16/0} resource was updated to new value: 1620828702 (2021/05/12T14:11:42+00:00)

Reading string resource {32011/0/31/0} value on LWM2M client.

String data in {32011/0/31/0} resource was updated to new content: <Greetings>

Reading string resource {32011/0/31/1} value on LWM2M client.

String data in {32011/0/31/1} resource was updated to new content: <from>

Reading string resource {32011/0/31/2} value on LWM2M client.

String data in {32011/0/31/2} resource was updated to new content: <AppZone!>

Reading boolean resource {32011/0/34/0} value on LWM2M client.

Boolean data in {32011/0/34/0} resource was updated to new value: true

Reading boolean resource {32011/0/34/1} value on LWM2M client.

Boolean data in {32011/0/34/1} resource was updated to new value: false

Reading boolean resource {32011/0/34/2} value on LWM2M client.

Boolean data in {32011/0/34/2} resource was updated to new value: true

Reading boolean resource {32011/0/34/3} value on LWM2M client.

Boolean data in {32011/0/34/3} resource was updated to new value: false
```

```
Waiting 5 seconds before the m2mb_lwm2m_objget

Waiting: 5

Waiting: 4

Waiting: 3

Waiting: 2

Waiting: 1

m2mb_lwm2m_objget SUCCESS

Content:
<{"bn":"/32011/0/",
"e":[
{"n":"11","sv":"hello world"},
{"n":"12","v":70},
{"n":"13","v":-12.500000},
{"n":"14","bv":false},
{"n":"15","sv":"aGVsbG8gd29ybGQ="},
{"n":"16","v":1620828702},
{"n":"17","ov":"0:0"},
{"n":"31/0","sv":"Greetings"},
{"n":"31/1","sv":"from"},
{"n":"31/2","sv":"AppZone!"},
{"n":"31/3","sv":""},
{"n":"31/4","sv":""},
{"n":"31/5","sv":""},
{"n":"31/6","sv":""},
{"n":"31/7","sv":""},
{"n":"31/8","sv":""},
{"n":"31/9","sv":""},
{"n":"34/0","bv":true},
{"n":"34/1","bv":false},
{"n":"34/2","bv":true},
{"n":"34/3","bv":false},
{"n":"34/4","bv":false},
{"n":"34/5","bv":false},
{"n":"34/6","bv":false},
{"n":"34/7","bv":false},
{"n":"34/8","bv":false},
{"n":"34/9","bv":false}]
}
>
```

After the Demo completes the initialization, it is possible to access the object re-

sources from the section Things: select your device, then “LwM2M” tab of OneEdge IoT portal:

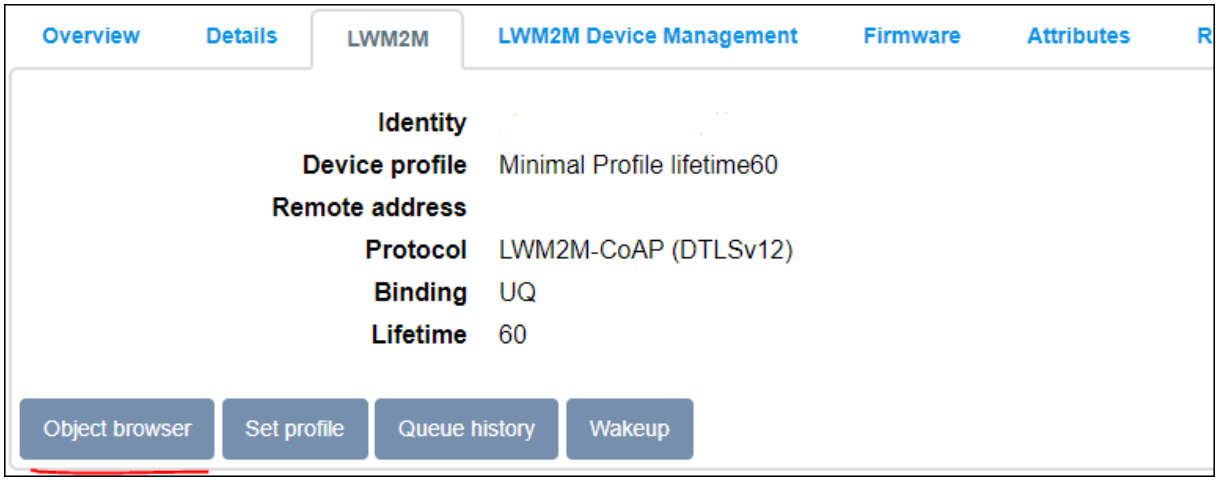

Figure 70

An instance of the object will be present and the resources can be modified.

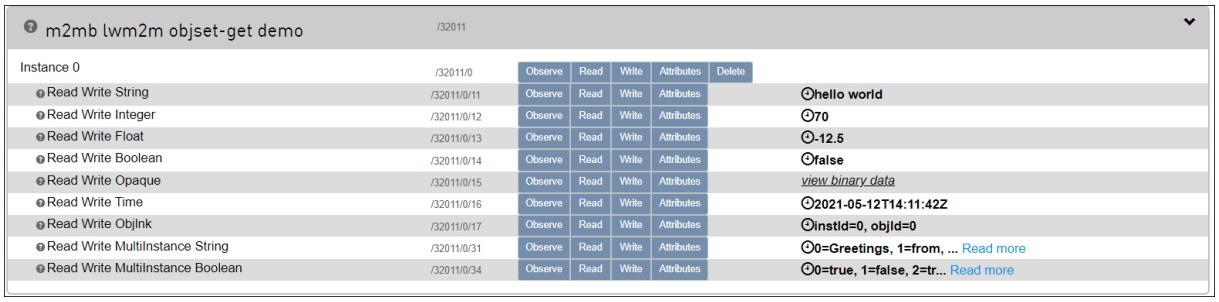

Figure 71

3.4.27 Logging Demo

Sample application showing how to print on one of the available output interfaces. Debug prints on **AUX UART**

Features

- How to open a logging channel
- How to set a logging level
- How to use different logging macros

3.4.27.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Print welcome message
- Print a message with every log level

```
Starting Logging demo app. This is v1.0.7 built on Mar 26 2020 13:57:06.
[WARN ] 20.17  M2MB_main.c:74 - M2MB_main{M2M_DamsStart}$ This is a WARNING MESSAGE
[ERROR] 20.18  M2MB_main.c:76 - M2MB_main{M2M_DamsStart}$ THIS IS AN ERROR MESSAGE
[CRITICAL] 20.19  M2MB_main.c:78 - M2MB_main{M2M_DamsStart}$ THIS IS AN CRITICAL MESSAGE
[DEBUG] 20.19  M2MB_main.c:80 - M2MB_main{M2M_DamsStart}$ This is a DEBUG message
[TRACE] 20.20  M2MB_main.c:82 - M2MB_main{M2M_DamsStart}$ This is a TRACE message
END.
```

Figure 72

3.4.28 Low power mode

The application shows how to set the module in low power modes (by disabling UART and RF). Debug prints on **AUX UART** which it is enabled/disabled to reach low power mode, using AZX log example functions

Features

- How to enable/disable LOG UART interfaces by azx apis
- How to enable/disable UART interfaces by m2mb apis
- How to set radio operating mode
- How to put the modem in low power mode

Application workflow

`M2MB_main.c`

- Print welcome message
- Print warning message about unplugging USB native port
- Init system events handler
- Sleep 20 seconds
- Disable RF
- Disable LOG UART
- Sleep 20 seconds
- Enable LOG UART
- Enable RF
- Sleep 60 seconds
- Deinit system events handler

```
Starting low power demo. This is v1.1.17 built on Jan 10 2023 12:27:24. LEVEL: 2
Please ensure USB native port is unplugged during low power state
Sleeping 20 seconds...
Disable LOG UART to reach low power mode and sleep for 20 seconds!
LOG UART is enabled again!
Sleeping 60 seconds...
Application end
```

Figure 73

3.4.29 MD5 example

Sample application showing how to compute MD5 hashes using m2mb crypto. Debug prints on **AUX UART**

Features

- Compute MD5 hash of a file
- Compute MD5 hash of a string

3.4.29.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Create a temporary file with the expected content
- Compute MD5 hash of the provided text file
- Compare the hash with the expected one
- Compute MD5 hash of a string
- Compare the hash with the expected one
- Delete test file

```
Starting MD5 demo app. This is v1.0.7 built on Apr  7 2020 10:19:54.
Buffer written successfully into file. 45 bytes were written.

Computing hash from file...
Computed hash: bb0fa6eff92c305f166803b6938dd33a
Expected hash: bb0fa6eff92c305f166803b6938dd33a
Hashes are the same!

Computing hash from string...
Computed hash: bb0fa6eff92c305f166803b6938dd33a
Expected hash: bb0fa6eff92c305f166803b6938dd33a
Hashes are the same!
```

Figure 74

3.4.30 MQTT Client

Sample application showcasing MQTT client functionalities (with SSL). Debug prints on **AUX UART**

Features

- How to check module registration and enable PDP context
- How to configure MQTT client parameters
- How to connect to a broker with SSL and exchange data over a subscribed topic

3.4.30.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage MQTT client and start it

`mqtt_demo.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init MQTT client
- Configure it with all parameters (Client ID, username, password, PDP context ID, keepalive timeout...)
- Connect MQTT client to broker
- Subscribe to two topics
- Publish 10 messages with increasing counter. Even messages are sent to topic 1, odd messages on topic 2.
- Print received message in mqtt_topc_cb function
- Disconnect MQTT client and deinit it
- Disable PDP context

```
Starting MQTT demo app. This is v1.0.7 built on Apr  7 2020 10:34:08.
[DEBUG] 16.18  mqtt_demo.c:192 - MQTT_Task{MQTT_TASK}$ INIT
[DEBUG] 16.18  mqtt_demo.c:206 - MQTT_Task{MQTT_TASK}$ m2mb_os_ev_init success
[DEBUG] 16.19  mqtt_demo.c:214 - MQTT_Task{MQTT_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 16.19  mqtt_demo.c:221 - MQTT_Task{MQTT_TASK}$ Waiting for registration...
[DEBUG] 16.20  mqtt_demo.c:131 - NetCallback{pubTspt_0}$ Module is registered
[DEBUG] 16.21  mqtt_demo.c:232 - MQTT_Task{MQTT_TASK}$ Pdp context activation
[DEBUG] 18.26  mqtt_demo.c:246 - MQTT_Task{MQTT_TASK}$ Activate PDP with APN web.omnitel.it on CID 3....
[DEBUG] 18.95  mqtt_demo.c:155 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 18.96  mqtt_demo.c:159 - PdpCallback{pubTspt_0}$ IP address: 37.118.201.56
[DEBUG] 18.96  mqtt_demo.c:268 - MQTT_Task{MQTT_TASK}$ Init MQTT
[DEBUG] 18.97  mqtt_demo.c:278 - MQTT_Task{MQTT_TASK}$ m2mb_mqtt_init succeeded

Connecting to broker <api-dev.devicewise.com>:1883...
Done.
Subscribing to test_topic and test_topic2..
[DEBUG] 20.35  mqtt_demo.c:367 - MQTT_Task{MQTT_TASK}$ Done.

[DEBUG] 20.36  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 2> to topic test_topic
[DEBUG] 20.37  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 20.71  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 20.72  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 2>
[DEBUG] 23.37  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 3> to topic test_topic2
[DEBUG] 23.38  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 23.92  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 23.93  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 3>
[DEBUG] 26.40  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 4> to topic test_topic
[DEBUG] 26.41  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 26.93  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 26.93  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 4>
[DEBUG] 29.42  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 5> to topic test_topic2
[DEBUG] 29.43  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 29.99  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 30.00  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 5>
[DEBUG] 32.46  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 6> to topic test_topic
[DEBUG] 32.48  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 33.00  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 33.01  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 6>
[DEBUG] 35.47  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 7> to topic test_topic2
[DEBUG] 35.48  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 36.01  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 36.02  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 7>
[DEBUG] 38.50  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 8> to topic test_topic
[DEBUG] 38.51  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 39.15  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 39.16  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 8>
[DEBUG] 41.52  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 9> to topic test_topic2
[DEBUG] 41.53  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 42.10  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 42.12  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 9>
[DEBUG] 44.56  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 10> to topic test_topic
[DEBUG] 44.57  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 45.09  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 28
[DEBUG] 45.11  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 10>
[DEBUG] 47.58  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 11> to topic test_topic2
[DEBUG] 47.59  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 48.12  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 28
[DEBUG] 48.13  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 11>

Disconnecting from MQTT broker..
[DEBUG] 50.60  mqtt_demo.c:414 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 50.61  mqtt_demo.c:443 - MQTT_Task{MQTT_TASK}$ application exit
[DEBUG] 50.62  mqtt_demo.c:453 - MQTT_Task{MQTT_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 50.63  mqtt_demo.c:457 - MQTT_Task{MQTT_TASK}$ Application complete.
[DEBUG] 51.23  mqtt_demo.c:164 - PdpCallback{pubTspt_0}$ Context deactivated!
```

Figure 75

3.4.31 MultiTask

Sample application showcasing multi tasking functionalities with M2MB API. Debug prints on **AUX UART**

Features

- How to create tasks using azx utilities
- How to use send messages to tasks
- How to use a semaphore to synchronize two tasks

3.4.31.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create three tasks with the provided utility (this calls public m2mb APIs)
- Send a message to the task1, its callback function azx_msgTask1 will be called

`azx_msgTask1`

- Print received parameters from main
- Send modified parameters to task2 (its callback function azx_msgTask2 will be called)
- wait for an InterProcess Communication semaphore to be available (released by task3)
- Once the semaphore is available, print a message and return

`azx_msgTask2`

- Print received parameters from caller
- If first parameter is bigger than a certain value, Send modified parameters to task3
- Else, use the second parameter as a task handle and print the corresponding name plus the value of the first parameter

`azx_msgTask3`

- Print received parameters from task 2
- release IPC semaphore
- send message to task 2 with first parameter below the threshold and second parameter with task3 handle

```
Starting MultiTask demo app. This is v1.0.12-C1 built on Jun 23 2020 15:36:31.

Inside "myTask1" user callback function. Received parameters from MAIN: 3 4 5
Task1 - Sending a message to task 2 with modified parameters...
Task1 - Waiting for semaphore to be released by task 3 now...

Inside "myTask2" user callback function. Received parameters: 5 7 10
Task2 - Sending a message to task 3 with modified parameters...
Task2 - Done.

Inside "myTask3" user callback function. Received parameters from Task 2: 15 14 9
Task3 - Releasing IPC semaphore...

Task1 - After semaphore! return...

Task3 - IPC semaphore released.
Task3 - Sending a message to task 2 with specific 'type' parameter value of 0 and task 3 handle as param1...

Inside "myTask2" user callback function. Received parameters: 0 1073951320 9
Task3 - Done.
Task2 - Received type 0 from task "myTask3"
Task2 - Done.
```

Figure 76

3.4.32 MutEx

Sample application showing mutex usage, with ownership and prioritization usage. Debug prints on **AUX UART**

Features

- How to create a mutex
- How to use the mutex with tasks having different priorities
- how to reorder the pending tasks queue for the mutex

3.4.32.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create four tasks with the provided utility (this calls public m2mb APIs). The first task is a “producer”, putting data on a shared buffer. The second is a “consumer” of said data, the other two are used for prioritization demo
- run producer and consumer tasks at the same pace. the shared buffer will stay empty, because the resource is consumed right after creation
- run producer twice as fast as consumer. The buffer is slowly filled
- run consumer twice as fast as publisher. The buffer is always empty.
- reserve the mutex in the main task and run producer, support and support2 tasks (in this order). Then release the mutex and check the execution order. It should be by arrival.
- reserve the mutex in the main task and run the same three task, but before releasing the mutex, call the prioritization API. the task with highest priority (producer) is put as first in the queue.

```
Starting MutEx app. This is v1.0.12-C1 built on Jul  1 2020 08:37:15.
[DEBUG] 14.50  M2MB_main:90 - mutex_init{M2M_DamsStart}$ [MUTEX] Mutex initialized

[CASE 1 ] Producer and consumer have same idle time

[DEBUG] 14.51  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 14.52  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 14.53  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 14.53  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 14.54  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 14.54  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 14.55  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 14.56  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 15.56  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 15.56  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 15.57  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 15.58  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 15.58  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 15.59  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 15.60  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 15.60  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 16.61  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 16.61  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 16.62  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 16.63  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 16.63  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 16.64  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 16.64  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 16.65  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
```

Figure 77

```
[CASE 2 ] Producer has double idle time

[DEBUG] 17.56  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 17.56  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 17.57  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 17.58  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 17.58  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 17.59  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 17.59  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 17.60  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 18.63  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 18.64  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 0 items
[DEBUG] 18.64  M2MB_main:268 - msgConsumer{CONSUMER}$ Can't consume anything, buffer size is 0
[DEBUG] 18.65  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 19.62  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 19.62  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 19.63  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 19.64  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 19.68  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 19.69  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 19.69  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 19.70  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 20.73  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 20.74  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 0 items
[DEBUG] 20.75  M2MB_main:268 - msgConsumer{CONSUMER}$ Can't consume anything, buffer size is 0
[DEBUG] 20.75  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 21.67  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 21.68  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 21.68  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 21.69  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 21.77  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 21.79  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 21.80  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 21.80  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
```

Figure 78

```
[CASE 3 ] Producer has half idle time

[DEBUG] 22.62  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 22.63  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 22.64  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 22.64  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 22.65  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 22.65  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 22.66  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 22.67  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 23.67  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 23.68  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 23.68  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 23.69  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 24.71  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 24.72  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 24.72  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 24.73  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 24.74  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 24.74  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 24.75  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 24.76  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 25.79  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 25.79  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 1 items
[DEBUG] 25.80  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 1
[DEBUG] 25.81  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 26.78  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 26.78  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 2 items
[DEBUG] 26.79  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 1
[DEBUG] 26.79  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 26.84  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 26.84  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 1 items
[DEBUG] 26.85  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 1
[DEBUG] 26.86  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
```

Figure 79

```
[CASE 4 ] NO HTPF

Reserve MUTEX so all tasks are enqueued
[DEBUG] 30.77  M2MB_main:387 - msgSupport{HPTF_SUPPORT}$ freepos = 0 | evaluate[freepos]= 3
[DEBUG] 30.78  M2MB_main:416 - msgSupport2{HPTF_SUPPORT2}$ freepos = 1 | evaluate[freepos]= 4
[DEBUG] 30.79  M2MB_main:223 - msgProducer{PRODUCER}$ producer: freepos = 2 | evaluate[freepos]= 1
[DEBUG] 35.85  M2MB_main:586 - M2MB_main{M2M_DamsStart}$ EVALUATE SEQUENCE IS 3 4 1. Expected: 3 4 1
NO HTPF OK

[CASE 4.1 ] HTPF USED

Reserve MUTEX so all tasks are enqueued
m2mb_os_mtx_hptf OK
[DEBUG] 41.98  M2MB_main:223 - msgProducer{PRODUCER}$ producer: freepos = 0 | evaluate[freepos]= 1
[DEBUG] 41.99  M2MB_main:387 - msgSupport{HPTF_SUPPORT}$ freepos = 1 | evaluate[freepos]= 3
[DEBUG] 42.00  M2MB_main:416 - msgSupport2{HPTF_SUPPORT2}$ freepos = 2 | evaluate[freepos]= 4
[DEBUG] 44.03  M2MB_main:650 - M2MB_main{M2M_DamsStart}$ EVALUATE SEQUENCE IS 1 3 4, expected 1 3 4
HTPF DEMO OK
The application has ended...
```

Figure 80

3.4.33 NTP example

The application connects to an NTP server, gets current date and time and updates module's internal clock. Debug prints on **AUX UART**

Features

- How to get current date and time from an NTP server
- How to set current date and time on module

3.4.33.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Send message to ntpTask

`ntp_task.c`

NTP_task() - Waits module registration - When module is registered, initializes ntp setting CID, server url and timeout - When PDP context is correctly opened, a query to NTP server is done to get current date and time - On SET_MODULE_RTC message type reception, module RTC is set with date time value got from NTP server.

m2mb_ntp_ind_callback() - As soon as M2MB_NTP_VALID_TIME event is received, current date and time is printend and a message (with SET_MODULE_RTC type) is sent to NTP_task

```
Start NTP demo application. This is v1.0 built on Apr 16 2021 09:36:12.

Waiting for registration...
Module is registered!

Activate PDP context with APN ibox.tim.it on CID 3
Context activated, IP address: 2.195.170.123
Get current time from server 0.pool.ntp.org, PORT: 123

Current time is: Friday 2021-04-16, 07:37:33

Current time correctly set on module
Module system time is: 2021-04-16, 07:37:33
```

Figure 81

3.4.34 RTC example

Sample application that shows RTC apis functionalities: how to get/set moudle system time and timestamp. Debug prints on **AUX UART**

Features

- How to read module timestamp
- How to read module system time
- How to set new system time

3.4.34.1 Application workflow

M2MB_main.c

- Init log azx and print a welcome message
- Init net functionality and wait for module registration
- Init RTC functionality and get module time in timestamp format (seconds from the epoch)
- Get moudle system time in date/time format
- Add 1 hour to timestamp, convert it to system time and set it to module

```
 Start RTC demo application. This is v1.0 built on Oct  1 2021 15:01:40.

Waiting for registration...
Module is registered!

Current time in seconds from the epoch: 1633101266
Module system time is: 2021-10-01, 15:14:26

Get current time and add an hour

Current time in seconds from the epoch: 1633101266
New time to be set : 2021-10-01, 16:14:26, tz:4, dlst:0

Set new time and check the setting
NEW module system time is: 2021-10-01, 16:14:26
```

Figure 82

3.4.35 SIM event handler example

Sim Event Demo application. Debug prints on **AUX UART**, using AZX log example functions

Features

- How to use ATI function for asynchronous management
- How to cath URC from an AppZone application
- How to catch SIM related events and handle them

Application workflow

`M2MB_main.c`

- Print welcome message
- Initialize AT interface
- Initialize AT URC manager task
- Initialize SIM event manager task
- Send “AT#SIMPR=1” to activate SIM URCs
- Insert SIM in SIM slot 1 and receive SIM inserted message
- Remove SIM from SIM slot 1 and receive SIM removed message

```
Starting SIM Presence Demo app. This is v1.1.17 built on Dec 16 2022 15:14:28.
Please ensure SIM is not inserted before starting this sample application
Sending command AT#SIMPR=1 to enable SIM presence URC messages...
Command set.
SIM 0 state changed to 1!

SIM CARD HAS BEEN INSERTED!

SIM 0 state changed to 0!

SIM CARD HAS BEEN REMOVED!
```

Figure 83

3.4.36 SMS PDU

Sample application showcasing how to create and decode PDUs to be used with m2mb_sms_* API set. A SIM card and antenna must be present. Debug prints on **AUX UART**

Features

- How to enable SMS functionality
- How to use encode an SMS PDU to be sent with m2mb_api
- How to decode a received SMS response from PDU to ASCII mode.

3.4.36.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init sms functionality
- Create PDU from text message
- Send message to destination number
- Wait for response
- When SMS PDU response is received, decode it and print information about it, plus the message content

```
m2mb_sms_init() succeeded

Sending message <How are you?>...
 m2mb_sms_send() - succeeded
M2MB_SMS_SEND_RESP Callback
Send resp msg ID 10
SMS received!
SMS correctly received!

Reading SMS from memory...
 m2mb_sms_read() request succeeded

--- SMS read ---
SMS tag M2MB_SMS_TAG_MT_NOT_READ
SMS format M2MB_SMS_FORMAT_3GPP
Code type: 0
Sender type: 145
Msg len: 12
Msg bytes: 11
Msg date 19/7/17 16:7:58 (timezone: 2)
Received SMS, content: <<Fine thanks >>
Sender: +[redacted]
```

Figure 84

3.4.37 SMS_atCmd example

Sample application showcasing how to receive an SMS containing an AT command, process the AT command and send its answer to sender (configurable in sms_config.txt). A SIM card and antenna must be present. Debug prints on **AUX UART**

Features

- How to receive an SMS with an AT command as text inside
- How to send AT command to parser and read the answer
- How to send the AT command answer back to sender via SMS

Optional configuration file to be put in `/mod` folder, copy `sms_config.txt` file into your module running the following AT command:

```
AT#M2MWRITE="/mod/sms_config.txt",138
>>> here receive the prompt; then type or send the file, sized 138 bytes
```

3.4.37.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Init SMS functionality
- Read configuration file sms_config.txt (send SMS with AT command answer back, delte SMS received)
- Init AT command parser
- Create a task to handle SMS parsing and AT command sending
- Wait for an incoming SMS

`callbacks.c`

`msgSMSparse()`

- When SMS has been received, content is decoded and printed. If there is an AT command inside, command is executed and answer printed and sent back to sender as an SMS (depending on sms_config.txt setting)

```
ÿStarting SMS with AT command demo app. This is v1.0.13-C1 built on Mar 18 2021 12:42:22.
[DEBUG] 16.61  M2MB_main:135 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
m2mb_sms_init() succeeded
[DEBUG] 16.62  M2MB_main:168 - M2MB_main{M2M_DamsStart}$ M2MB_SMS_INCOMING_IND indication enabled
[DEBUG] 16.63  M2MB_main:179 - M2MB_main{M2M_DamsStart}$ M2MB_SMS_INCOMING_IND MEMORY FULL indication enabled
[DEBUG] 16.64  M2MB_main:196 - M2MB_main{M2M_DamsStart}$ Storage set to M2MB_SMS_STORAGE_SM
[DEBUG] 16.65  callbacks:114 - readConfigFromFile{M2M_DamsStart}$ Reading parameters from file
[DEBUG] 16.66  callbacks:116 - readConfigFromFile{M2M_DamsStart}$ Opening /mod/sms_config.txt in read mode..
Default: SMS with answer sending DISABLED, delete sms DISABLED
[DEBUG] 16.67  at_async:115 - at_cmd_async_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Please send an SMS with a configuration as ("ATCMD: <atcmd>")...
```

Figure 85

3.4.38 SMTP Client

Sample application showing SMTP echo demo with M2MB API. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to open a SMTP client
- How to send a mail

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage SMTP client and start it

`M2MB_main.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Initialize SMTP client and connect to SMTP server
- Prepare email and send it
- Close SMTP client
- Disable PDP context

```
Starting SMTP demo app. This is v1.1.17 built on Jan  9 2023 09:57:51.
Reading parameters from file...
Set APN to: <<nxt17.net>>
Set APN USER to: <<>>
Set APN PASS to: <<>>
Set PDP_CTX to: 1
Set MAIL SERVER to: <<           >>
Set MAIL PORT to: <<     >>
Set MAIL USER to: <<                            >>
Set MAIL PASS to: <<                    >>
Set MAIL FROM to: <<                            >>
Set MAIL FROM NAME to: <<   >>
Set MAIL SUBJECT to: <<Hello from AppZone>>
Set MAIL BODY to: <<AppZone Test>>
Set MAIL TO to: <<                           >>
Set MAIL TO NAME to: <<  >>
Waiting for registration...
Pdp context activation
Connecting to SMTP Server...
SMTP Client connected!
Preparing example's mail...
Sending example's mail...
Example's mail sent!
SMTP client closed
Application complete.
```

Figure 86

3.4.39 SW Timer (Software Timer)

The sample application shows how to use SW Timers M2MB API. Debug prints on **AUX UART**

Features

- How to open configure a SW timer
- How to use the timer to manage recurring events

3.4.39.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create sw timer structure
- Configure it with 4 seconds timeout, periodic timer (auto fires when expires)
- Init the timer with the parameters
- Start the timer
- Wait 10 seconds
- Stop the timer

`timerCb`

- Print a message with inside the callback

```
Starting SW Timers demo app. This is v1.0.7 built on Apr  7 2020 09:51:25.
timer expired!
[DEBUG] 21.41  M2MB_main.c:59 - timerCb{pubTspt_0}$ timer handle: 0x4002b004
timer expired!
[DEBUG] 25.47  M2MB_main.c:59 - timerCb{pubTspt_0}$ timer handle: 0x4002b004
stopping the timer
Stop a running timer: success
Application end
```

Figure 87

3.4.40 Secure MicroService

Sample application showcasing how to manage secure microservice functionalities. Debug prints on **AUX UART**

Features

- Write data in Secure Data Area (SDA), non protected
- Read the written data and compare with the original buffer
- Write a cripty key in Secure Data Area (SDA), non protected
- Perform a rotate of the written key data
- Perform MD5 sum of written data from TZ file
- Compare computed digest with expected one
- Write data in trust zone as a trusted object (it will not be possible to read it again but only use its content for crypto operations)
- Try to read the trusted object and verify it fails
- Rotate trusted item and verify retrieving the content fails
- compute MD5 sum of trusted item and compare with the expected one
- Try to pass data from a trusted item to a non trusted item using untrusted TZ buffers, and verify it fails

3.4.40.1 Application workflow

`M2MB_main.c`

- Write a buffer in a SDA item using `m2mb_secure_ms_write`
- Read the same item using `m2mb_secure_ms_read`
- Write a buffer containing some cripty key in a SDA item using `m2mb_secure_ms_write`
- Rotate the content of the key item
- Read it with `m2mb_secure_ms_read`
- Load the key content using `m2mb_secure_ms_crypto_alloc` and `m2mb_secure_crypto_add_it` in a SECURE_MS buffer
- Compute MD digest with `m2mb_secure_ms_crypto_md`
- Write a buffer containing some cripty key in a SDA item using `m2mb_secure_ms_write` but with **TRUSTED** option in `m2mb_secure_ms_open`
- Verify that `m2mb_secure_ms_read` on the trusted item fails
- Verify that `m2mb_secure_ms_crypto_rotate` fails for the trusted item
- Verify the MD5 digest
- Try to copy the trusted item data in a SECURE_MS buffer with `m2mb_secure_ms_crypto_alloc` and `m2mb_secure_crypto_add_item`, then load it in an untrusted object with `m2mb_secure_ms_crypto_write`, and verify it fails.

```
Starting secure ms demo app. This is v1.0.13-C1 built on Jul 30 2020 12:19:02.
Writing data in normal item
Stored input data in Secure Data Area
Reading data from normal item
Data length in SDA: 11 bytes
Securely loaded the data from the SDA
Read 11 bytes: <hello world>
original and retrieved strings are the same

Writing key in normal item
Stored input data in Secure Data Area

Rotate data in normal item
Original key: AA_THIS_IS_MY_SECRET_KEY_BB
Rotated key:

Compute MD5 of data in normal item
Data length in SDA: 27 bytes
MD5: 8EDAD26E26E1C74C7C02386C1C7F541D
hash is the expected one!

Writing data in trusted item
Stored input data in Secure Data Area
Reading data from trusted item (should fail!)
Data length in SDA: 27 bytes
m2mb_secure_ms_read() failed for trusted item, as expected!

Rotate data in trusted item
[ERROR] 17.01  M2MB_main:329 - read_rotate{M2M_DamsStart}$ Cannot read data from SECURE_MS_BUFFER to user buffer
Original key: AA_THIS_IS_MY_SECRET_KEY_BB
Rotated key:

Compute MD5 of data in trusted item
Data length in SDA: 27 bytes

MD5: 8EDAD26E26E1C74C7C02386C1C7F541D
Hash is the expected one!

Try to pass data from trusted to untrusted through TZ buffers
Cannot store data from SECURE_MS_BUFFER to SDA 'non-trusted', as expected
```

Figure 88

3.4.41 TCP IP

Sample application showcasing TCP echo demo with M2MB API. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to open a TCP client socket
- How to communicate over the socket

3.4.41.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server
- Send data and receive response
- Close socket
- Disable PDP context

```
Starting TCP-IP demo app. This is v1.0.7 built on Mar 26 2020 16:20:30.
[DEBUG] 21.23  m2m_tcp_test.c:201 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 21.25  m2m_tcp_test.c:217 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.26  m2m_tcp_test.c:223 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.26  m2m_tcp_test.c:231 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 21.28  m2m_tcp_test.c:128 - NetCallback{pubTspt_0}$ Module is registered to cell 0x816B!
[DEBUG] 21.29  m2m_tcp_test.c:244 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 21.30  m2m_tcp_test.c:248 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.34  m2m_tcp_test.c:263 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 24.52  m2m_tcp_test.c:155 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.52  m2m_tcp_test.c:158 - PdpCallback{pubTspt_0}$ IP address: 83.225.44.56
[DEBUG] 24.54  m2m_tcp_test.c:273 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 24.54  m2m_tcp_test.c:284 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 24.55  m2m_tcp_test.c:294 - M2M_msgTCPTask{TCP_TASK}$ Socket ctx set to 3
[DEBUG] 24.95  m2m_tcp_test.c:307 - M2M_msgTCPTask{TCP_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 25.17  m2m_tcp_test.c:322 - M2M_msgTCPTask{TCP_TASK}$ Socket Connected!
[DEBUG] 25.18  m2m_tcp_test.c:329 - M2M_msgTCPTask{TCP_TASK}$ Sending data over socket..
[DEBUG] 25.19  m2m_tcp_test.c:342 - M2M_msgTCPTask{TCP_TASK}$ Data send successfully (16 bytes)
[DEBUG] 27.20  m2m_tcp_test.c:356 - M2M_msgTCPTask{TCP_TASK}$ trying to receive 16 bytes..
[DEBUG] 27.21  m2m_tcp_test.c:364 - M2M_msgTCPTask{TCP_TASK}$ Data received (16): <hello from m2mb!>
[DEBUG] 27.21  m2m_tcp_test.c:373 - M2M_msgTCPTask{TCP_TASK}$ application exit
[DEBUG] 27.22  m2m_tcp_test.c:385 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 27.24  m2m_tcp_test.c:388 - M2M_msgTCPTask{TCP_TASK}$ Application complete.
[DEBUG] 29.43  m2m_tcp_test.c:164 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 89

3.4.42 TCP Socket status

Sample application showcasing how to check a TPC connected socket current status. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to open a TCP client socket
- How to check if the TCP socket is still valid

3.4.42.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server
- Check in a loop the current socket status using the adv_select function with a 2 seconds timeout
- Close socket when the remote host closes it
- Disable PDP context

```
Starting TCP socket status check demo app. This is v1.0.14-C1 built on Sep  8 2020 14:59:25.
[DEBUG] 21.33  m2m_tcp_tes:324 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 21.34  m2m_tcp_tes:338 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.34  m2m_tcp_tes:344 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.35  m2m_tcp_tes:352 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 21.36  m2m_tcp_tes:365 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 21.37  m2m_tcp_tes:369 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.41  m2m_tcp_tes:384 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN NXT17.NET....
[DEBUG] 24.09  m2m_tcp_tes:281 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.10  m2m_tcp_tes:284 - PdpCallback{pubTspt_0}$ IP address: 100.77.5.223
[DEBUG] 24.10  m2m_tcp_tes:394 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 24.11  m2m_tcp_tes:405 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 24.11  m2m_tcp_tes:415 - M2M_msgTCPTask{TCP_TASK}$ Socket ctx set to 3
[DEBUG] 24.60  m2m_tcp_tes:428 - M2M_msgTCPTask{TCP_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 24.93  m2m_tcp_tes:443 - M2M_msgTCPTask{TCP_TASK}$ Socket Connected!
[DEBUG] 26.98  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 29.03  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...

...

[DEBUG] 82.18  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 84.23  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 86.28  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 88.31  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 88.90  m2m_tcp_tes:154 - adv_select{TCP_TASK}$ Data is available on socket <0x40032b3c>
[DEBUG] 88.92  m2m_tcp_tes:160 - adv_select{TCP_TASK}$ There are <0> pending bytes on socket
Socket was closed by remote!
[DEBUG] 88.92  m2m_tcp_tes:494 - M2M_msgTCPTask{TCP_TASK}$ application exit
[DEBUG] 88.94  m2m_tcp_tes:506 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 88.94  m2m_tcp_tes:509 - M2M_msgTCPTask{TCP_TASK}$ Application complete.
[DEBUG] 89.31  m2m_tcp_tes:290 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 90

3.4.43 TCP Server

Sample application showcasing TCP listening socket demo with M2MB API. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to open a TCP listening socket
- How to manage external hosts connection and exchange data

3.4.43.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and set it in non-blocking mode
- Bind the socket to the listening port
- Start listening for incoming connection
- Check if a connection is incoming using m2mb_socket_bsd_select function
- If a client connects, perform accept on the child socket
- Send a “START” message to the client
- Send some data
- Wait for data from client and print it
- Close the child socket
- Start listening again, up to 3 times
- Close listening socket
- Disable PDP context

Debug Log

```
Starting TCP Server demo app. This is v1.0.7 built on Apr  7 2020 13:28:24.
[DEBUG] 14.55  m2m_tcp_test.c:220 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 14.55  m2m_tcp_test.c:236 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 14.57  m2m_tcp_test.c:242 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 14.57  m2m_tcp_test.c:250 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 14.58  m2m_tcp_test.c:138 - NetCallback{pubTspt_0}$ Module is registered to cell 0x5222!
[DEBUG] 14.59  m2m_tcp_test.c:263 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 14.60  m2m_tcp_test.c:267 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 16.57  m2m_tcp_test.c:282 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN ibox.tim.it....
[DEBUG] 17.16  m2m_tcp_test.c:165 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 17.17  m2m_tcp_test.c:168 - PdpCallback{pubTspt_0}$ IP address: 2.195.165.137


--------------------------------------------
|                                          |
|  Start TCP server                        |
|                                          |
--------------------------------------------


[DEBUG] 19.15  m2m_tcp_test.c:301 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 19.15  m2m_tcp_test.c:312 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 19.16  m2m_tcp_test.c:313 - M2M_msgTCPTask{TCP_TASK}$ m2mb_socket_bsd_socket(): valid socket ID [0x4002E79C] - PASS
[DEBUG] 20.16  m2m_tcp_test.c:319 - M2M_msgTCPTask{TCP_TASK}$ issuing m2m_socket_bsd_ioctl() to set non-blocking mode ...
[DEBUG] 20.17  m2m_tcp_test.c:331 - M2M_msgTCPTask{TCP_TASK}$ Binding Socket...
[DEBUG] 22.12  m2m_tcp_test.c:343 - M2M_msgTCPTask{TCP_TASK}$ Socket Bind Pass


Start TCP listening on port 6500...

[DEBUG] 24.13  m2m_tcp_test.c:368 - M2M_msgTCPTask{TCP_TASK}$ select...
Select result: 0
[DEBUG] 28.13  m2m_tcp_test.c:368 - M2M_msgTCPTask{TCP_TASK}$ select...
Select result: 1

TCP Server Coming Connection

--> Accept
[DEBUG] 30.52  m2m_tcp_test.c:397 - M2M_msgTCPTask{TCP_TASK}$ Socket Accept Pass


Connected! (socket dial n.1)

[DEBUG] 30.53  m2m_tcp_test.c:403 - M2M_msgTCPTask{TCP_TASK}$ Client Source Address: 185.86.42.254
[DEBUG] 30.54  m2m_tcp_test.c:404 - M2M_msgTCPTask{TCP_TASK}$ Client Port: 58658
[DEBUG] 30.54  m2m_tcp_test.c:405 - M2M_msgTCPTask{TCP_TASK}$ Client Family: 2
[DEBUG] 31.56  m2m_tcp_test.c:410 - M2M_msgTCPTask{TCP_TASK}$
-------------------------------
[DEBUG] 31.57  m2m_tcp_test.c:411 - M2M_msgTCPTask{TCP_TASK}$ |    Send/receive data test    |
[DEBUG] 31.57  m2m_tcp_test.c:412 - M2M_msgTCPTask{TCP_TASK}$ -------------------------------

[DEBUG] 32.58  m2m_tcp_test.c:416 - M2M_msgTCPTask{TCP_TASK}$
--> issuing m2mb_socket_bsd_send(): transmit "__START____" packet...
[DEBUG] 32.59  m2m_tcp_test.c:423 - M2M_msgTCPTask{TCP_TASK}$  --> done (11 have been transmitted)
[DEBUG] 32.60  m2m_tcp_test.c:425 - M2M_msgTCPTask{TCP_TASK}$      ALL data transmitted - PASS
[DEBUG] 32.61  m2m_tcp_test.c:430 - M2M_msgTCPTask{TCP_TASK}$
--> issuing m2mb_socket_bsd_send(): transmit 58 bytes...
[DEBUG] 32.62  m2m_tcp_test.c:437 - M2M_msgTCPTask{TCP_TASK}$  --> done (58 have been transmitted)
[DEBUG] 32.63  m2m_tcp_test.c:440 - M2M_msgTCPTask{TCP_TASK}$      ALL data transmitted - PASS
[DEBUG] 32.64  m2m_tcp_test.c:448 - M2M_msgTCPTask{TCP_TASK}$
Waiting for data...

[DEBUG] 39.64  m2m_tcp_test.c:457 - M2M_msgTCPTask{TCP_TASK}$ test
[DEBUG] 99.61  m2m_tcp_test.c:465 - M2M_msgTCPTask{TCP_TASK}$
m2mb_socket_bsd_recv() has received 6 bytes

[DEBUG] 102.60  m2m_tcp_test.c:469 - M2M_msgTCPTask{TCP_TASK}$
Server TCP is closing the current connection ...
```

Figure 91

Data on a PuTTY terminal

```
START

aaaaaaaaaa-bbbbbbbbbb-cccccccccc-dddddddddd-eeeeeeeeee

test
```

Figure 92

3.4.44 TLS SSL Client

Sample application showcasing TLS/SSL with client certificates usage with M2MB API. Debug prints on **AUX UART**

Features

- How to check module registration and enable PDP context
- How to open a SSL client socket
- How to communicate over SSL socket

3.4.44.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the connection and start it

`ssl_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server over TCP socket
- Initialize the TLS parameters (TLS1.2) andh auth mode (server+client auth in the example)
- Create SSL context
- Read certificates files and store them
- Create secure socket and connect to the server using SSL
- Send data and receive response
- Close secure socket
- Close socket
- Delete SSL context
- Disable PDP context

The application requires the certificates to be stored in `/mod/ssl_certs/` folder. It can be created with

`AT#M2MMKDIR=/mod/ssl_certs`

Certificates can then be loaded with

AT#M2MWRITE="/mod/ssl_certs/modulesCA.crt",1740

and providing the file content in RAW mode (for example using the “Transfer Data” button in Telit AT Controller)

For client certificates (if required), the commands will be

AT#M2MWRITE="/mod/ssl_certs/modulesClient.crt",1651
AT#M2MWRITE="/mod/ssl_certs/modulesClient_pkcs1.key",1679

PLEASE NOTE: always verify the file sizes to be used in the commands above as they might change

```
Starting TLS-SSL demo app. This is v1.1.2 built on Mar  3 2021 10:15:00.
[DEBUG] 10.85  ssl_test:252 - msgHTTPSTask{TLS_TASK}$ INIT
[DEBUG] 10.85  ssl_test:266 - msgHTTPSTask{TLS_TASK}$ m2mb_os_ev_init success
[DEBUG] 10.85  ssl_test:270 - msgHTTPSTask{TLS_TASK}$ Init SSL session test app
[DEBUG] 10.85  ssl_test:285 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config sslConfigHndl = 0x40037958, sslRes= 0
[DEBUG] 10.85  ssl_test:294 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config PASSED
[DEBUG] 10.85  ssl_test:306 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_ctxt PASSED
[DEBUG] 10.85  ssl_test:311 - msgHTTPSTask{TLS_TASK}$ loading CA CERT from file /mod/ssl_certs/modulesCA.crt
[DEBUG] 10.85  ssl_test:315 - msgHTTPSTask{TLS_TASK}$ file size: 1740
[DEBUG] 10.85  ssl_test:328 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1740
Buffer successfully received from file. 1740 bytes were loaded.
Closing file.
[DEBUG] 10.85  ssl_test:361 - msgHTTPSTask{TLS_TASK}$ loading client CERT from file /mod/ssl_certs/modulesClient.crt
[DEBUG] 10.85  ssl_test:365 - msgHTTPSTask{TLS_TASK}$ file size: 1651
[DEBUG] 10.85  ssl_test:378 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1651
Buffer successfully received from file. 1651 bytes were loaded.
Closing file.
[DEBUG] 10.85  ssl_test:401 - msgHTTPSTask{TLS_TASK}$ loading client KEY from file /mod/ssl_certs/modulesClient_pkcs1.key
[DEBUG] 10.85  ssl_test:405 - msgHTTPSTask{TLS_TASK}$ file size: 1679
[DEBUG] 10.85  ssl_test:418 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1679
Buffer successfully received from file. 1679 bytes were loaded.
Closing file.
[DEBUG] 10.85  ssl_test:448 - msgHTTPSTask{TLS_TASK}$ certificates successfully stored!
[DEBUG] 10.85  ssl_test:457 - msgHTTPSTask{TLS_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 10.85  ssl_test:465 - msgHTTPSTask{TLS_TASK}$ Waiting for registration...
[DEBUG] 10.86  ssl_test:171 - NetCallback{pubTspt_0}$ Module is registered to cell 0x468E!
[DEBUG] 10.86  ssl_test:477 - msgHTTPSTask{TLS_TASK}$ Pdp context activation
[DEBUG] 10.86  ssl_test:481 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 12.87  ssl_test:496 - msgHTTPSTask{TLS_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 13.71  ssl_test:197 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 13.71  ssl_test:200 - PdpCallback{pubTspt_0}$ IP address: 2.41.76.63
[DEBUG] 13.71  ssl_test:514 - msgHTTPSTask{TLS_TASK}$ Creating Socket...
[DEBUG] 13.71  ssl_test:525 - msgHTTPSTask{TLS_TASK}$ Socket created
[DEBUG] 13.71  ssl_test:535 - msgHTTPSTask{TLS_TASK}$ Socket ctx set to 3
[DEBUG] 13.92  ssl_test:548 - msgHTTPSTask{TLS_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 14.05  ssl_test:562 - msgHTTPSTask{TLS_TASK}$ Socket Connected!
[DEBUG] 15.97  ssl_test:587 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_connect ret 0
[DEBUG] 17.99  ssl_test:593 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
```

```
[DEBUG] 17.99  ssl_test:593 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
[DEBUG] 17.99  ssl_test:596 - msgHTTPSTask{TLS_TASK}$ SSL write result = 44
[DEBUG] 22.03  ssl_test:608 - msgHTTPSTask{TLS_TASK}$ pending bytes: 1087
[DEBUG] 22.03  ssl_test:612 - msgHTTPSTask{TLS_TASK}$ trying to receive 1087 bytes..
[DEBUG] 22.03  ssl_test:618 - msgHTTPSTask{TLS_TASK}$ Server response: (269)<HTTP/1.1 200 OK
Date: Wed, 03 Mar 2021 09:18:22 GMT
Server: Apache/2.2.15 (CentOS)
Last-Modified: Mon, 22 Jan 2018 10:57:39 GMT
ETag: "1fffc-27f-5635b4c6f12b3"
Accept-Ranges: bytes
Content-Length: 639
Connection: close
Content-Type: text/html; charset=UTF-8

>
[DEBUG] 22.03  ssl_test:634 - msgHTTPSTask{TLS_TASK}$ pending bytes: 762
[DEBUG] 22.03  ssl_test:638 - msgHTTPSTask{TLS_TASK}$ trying to receive remaining 762 bytes..
[DEBUG] 22.03  ssl_test:644 - msgHTTPSTask{TLS_TASK}$ Server response: (639)<<html>
<head>
 <title>module.telit.com</title>
 <meta content="text/html; charset=utf-8" />
</head>
<body>
 <table border=0 align=center>
 <tr>
 <td height="100" align=center><h2>modules.telit.com - Test HTML page</h2></td>
 </tr>
 <tr>
 <td align=center><img src=Telit.jpg alt="Telit logo" height="126" width="410"></img></td>
 </tr>
 <tr>
 <td height="200" align=center> This is a simple HTML page, </br>
 made with simple HTML code,</br>
 just for test!
 </td>
 </tr>
 <tr>
 <td height="100" align=center><font size="3">Telit &copy; 2015 - 2017 All rights reserved</font></td>
 </tr>
 </table>
</body>
</html>
>
[DEBUG] 22.03  ssl_test:662 - msgHTTPSTask{TLS_TASK}$ application exit
[DEBUG] 22.03  ssl_test:680 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 22.03  ssl_test:683 - msgHTTPSTask{TLS_TASK}$ Application complete.
[DEBUG] 22.77  ssl_test:206 - PdpCallback{pubTspt_0}$ Context deactivated!
```

3.4.45 UDP client

Sample application showcasing UDP echo demo with M2MB API. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to open a UDP client socket
- How to communicate over the socket

3.4.45.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task and start it

`m2m_udp_test.c` - Initialize Network structure and check registration - Initialize PDP structure and start PDP context - Create socket and link it to the PDP context id - Send data and receive response - Close socket - Disable PDP context

```
Starting UDP client demo app. This is v1.0.7 built on Apr  1 2020 14:57:13.
INIT
[DEBUG] 21.23  m2m_udp_test.c:223 - M2M_msgUDPTask{UDP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
Waiting for registration...
[DEBUG] 21.25  m2m_udp_test.c:131 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 21.26  m2m_udp_test.c:241 - M2M_msgUDPTask{UDP_TASK}$ Pdp context initialization
[DEBUG] 21.26  m2m_udp_test.c:245 - M2M_msgUDPTask{UDP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
Activate PDP with APN web.omnitel.it....
[DEBUG] 24.11  m2m_udp_test.c:157 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.11  m2m_udp_test.c:160 - PdpCallback{pubTspt_0}$ IP address: 109.113.222.12
[DEBUG] 24.12  m2m_udp_test.c:268 - M2M_msgUDPTask{UDP_TASK}$ Creating Socket...
[DEBUG] 24.13  m2m_udp_test.c:280 - M2M_msgUDPTask{UDP_TASK}$ Socket created
Socket ctx set to 3
[DEBUG] 24.41  m2m_udp_test.c:306 - M2M_msgUDPTask{UDP_TASK}$ Retrieved IP: 185.86.42.218
Socket ready.
Data successfully sent (16 bytes)
Socket recv...
[DEBUG] 26.47  m2m_udp_test.c:352 - M2M_msgUDPTask{UDP_TASK}$ m2mb_socket_bsd_set_sock_opt() M2MB_SOCKET_BSD_SO_RCVTIMEO - success
trying to receive 16 bytes..
Data received (16): <hello from m2mb!>
[DEBUG] 26.48  m2m_udp_test.c:377 - M2M_msgUDPTask{UDP_TASK}$ application exit
Socket Closed
[DEBUG] 26.49  m2m_udp_test.c:399 - M2M_msgUDPTask{UDP_TASK}$ m2mb_pdp_deactivate returned success
Application complete.
[DEBUG] 27.04  m2m_udp_test.c:166 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 93

3.4.46 USB Cable Check

Sample application showing how to check if USB cable is plugged in or not. Debug prints on **AUX UART**

Features

- How to open an USB channel and configure it with a callback function
- How to manage USB cable events in the callback function

3.4.46.1 Application workflow

M2MB_main.c

- Open UART/UART_AUX for debug
- open usb channel and set the callback
- Print greeting message
- Print current usb status

USB_Cb

- if the event is a connection/disconnection, show the current status

```
Starting USB cable check demo app. This is v1.0.0 built on Aug 19 2020 10:27:40.

m2mb_usb_open succeeded
m2mb_usb_ioctl: set usb callback
m2mb_usb_ioctl: got cable status
USB cable CONNECTED, status: 1

 Waiting for USB cable to be plugged/unplugged...
Usb cable check event, USB status: 0
Usb cable check event, USB status: 1
Usb cable check event, USB status: 0
Usb cable check event, USB status: 1
```

Figure 94

3.4.47 Watchdog example

Sample application that shows how to set and implement a watchdog. Task has been locked waitng for an event with a timeout longer than wd inactivity timeout. If no wd kick or no actions

Features

- Enable watchdog on Task_1
- Handle watchdog kicks with a timer

3.4.47.1 Application workflow

M2MB_main.c

- Create Task_1 that will be put under watchdog control
- In Task_1 watchdog is ebnabled on it and a timer created to handle watchdog kicks. Then task enters in a loop where kicks are sent. After 5 kicks a task lock is simulated waiting for an event release (with a timeout longer that task watchdog inactivity timeout)
- As soon as watchdog inactivity timeout expired M2MB_WDOG_TIMEOUT_IND is generated and handled in WDcallback (in this case the event is released, no further actions are done)
- No further actions are performed by Task_1 (no kicks are sent) so after watchdog system timeout expiration app is rebooted.

```
Starting WATCHDOG demo app. This is v1.0.0 built on Nov  4 2021 14:34:06.

Event correctly initialized

Task started

Init WatchDog
m2mb_wDog_init OK
Tick duration: 1s
Adding Task under WD control with inactivity timeout of 60s
m2mb_wDog_addTask OK
Enabling the WatchDog
m2mb_wDog_enable OK
Timer creation OK

Start WD kick every 5s...
WD kick
WD kick done
```

Figure 95

3.4.48 ZLIB example

Sample application showing how to compress/uncompress with ZLIB. Debug prints on **AUX UART**

Features

- How to compress a file
- How to uncompress a file

In order to execute the entire test, copy `test.gz` file into your module running the following AT command:

```
AT#M2MWRITE="/mod/test.gz",138
>>> here receive the prompt; then type or send the file, sized 138 bytes
```

3.4.48.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Test the compression and decompression of a data string
- Test the decompression of a .gz file (test.gz), expected to be in `/mod` folder, into its content `test.txt`. The file must be uploaded by the user (see steps above).

```
Starting Logging demo app. This is v1.0.7 built on Apr  7 2020 09:02:35.

Starting TEST_COMPR_UNCOMPR.
len: 138; comprLen: 57
Compressed message:
 W+ÉHU(,ÌLÏVH*É/ÌSHÉ¨ PÉ*Ì-HMQÉ/K-R(  Êç$VU*ä§ë y4RÌ«¥1、
comprLen: 57; uncomprLen: 138
uncompress():
the quick brown fox jumped over the lazy dog. the quick brown fox jumped over the lazy dog. the quick brown fox jumped over the lazy dog.
Ending TEST_COMPR_UNCOMPR with SUCCESS.

Starting test_uncompress.

Data extracted correctly into the file ./mod/test.txt
test_uncompress finished correctly!
```

Figure 96

3.5 USB0

Applications that provide usage examples for various functionalities, log output on USB0

3.5.1 ATI (AT Instance)

Sample application showing how to use AT Instance functionality (sending AT commands from code). The example supports both sync and async (using a callback) modes. Debug prints on **USB0**

Features

- How to open an AT interface from the application
- How to send AT commands and receive responses on the AT interface

3.5.1.1 Application workflow, sync mode

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init AT0 (first AT instance)
- Send AT+CGMR command
- Print response.
- Release AT0

`at_sync.c`

- Init ati functionality and take AT0
- Send AT+CGMR command, then read response after 2 seconds, then return it
- Deinit ati, releasing AT0

```
Starting AT demo app. This is v1.0.7 built on Apr  1 2020 15:12:58.
[DEBUG] 17.15  at_sync.c:53 - at_cmd_sync_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Sending command AT+CGMR in sync mode
[DEBUG] 17.16  at_sync.c:79 - send_sync_at_command{M2M_DamsStart}$ Sending AT Command: AT+CGMR
Command response: <AT+CGMR
M0B.950004-B008

OK
>

[DEBUG] 19.21  at_sync.c:61 - at_cmd_sync_deinit{M2M_DamsStart}$ m2mb_ati_deinit() on instance 0
Application end
```

Figure 97

3.5.1.2 Application workflow, async mode

M2MB_main.c

- Open USB/UART/UART_AUX
- Init AT0 (first AT instance)
- Send AT+CGMR command
- Print response.
- Release AT0

at_async.c

- Init ati functionality and take AT0, register AT events callback
- Send AT+CGMR command, wait for response semaphore (released in callback), then read it and return it
- Deinit ati, releasing AT0

```
Starting AT demo app. This is v1.0.7 built on Apr  1 2020 15:07:45.
[DEBUG] 17.13  at_async.c:116 - at_cmd_async_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Sending command AT+CGMR in async mode
[DEBUG] 17.15  at_async.c:153 - send_async_at_command{M2M_DamsStart}$ Sending AT Command: AT+CGMR
[DEBUG] 17.15  at_async.c:169 - send_async_at_command{M2M_DamsStart}$ waiting command response...
[DEBUG] 17.17  at_async.c:88 - at_cmd_async_callback{pubTspt_0}$ Callback - available bytes: 25
[DEBUG] 17.18  at_async.c:181 - send_async_at_command{M2M_DamsStart}$ Receive response...
Command response: <AT+CGMR
M0B.950004-B008

OK
>

[DEBUG] 17.19  at_async.c:136 - at_cmd_async_deinit{M2M_DamsStart}$ m2mb_ati_deinit() on instance 0
Application end
```

Figure 98

3.5.2 AWS demo

Sample application showcasing AWS Iot Core MQTT communication. Debug prints on **USB0**

Features

- How to check module registration and enable PDP context
- How to load certificates into device SSL session storage
- How to configure MQTT client parameters
- How to connect to AWS server with SSL and exchange data over a topic

3.5.2.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage MQTT client and start it

`aws_demo.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init MQTT client
- Configure it with all parameters (Client ID, PDP context ID, keepalive timeout…)
- Initialize the TLS parameters (TLS1.2) andh auth mode (server+client auth in the example)
- Create SSL context
- Read certificates files and store them
- Connect MQTT client to broker
- Subscribe to topic
- Publish 10 messages with increasing counter
- Print received message in mqtt_topc_cb function
- Disconnect MQTT client and deinit it
- Disable PDP context

3.5.2.2 How to get started with AWS IoT

- Go to AWS console and create an account if one is not available yet.
- Go to **IoT Core** section
- Go to **Secure** > **Policies** section
- Create a new policy, which describes what the device will be allowed to do (e.g. subscribe, publish)
- Give it a name, then configure it using the configuration below (it is possible to copy/paste by clicking on **Add statements** section, then **Advanced mode**) :

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
"Action": [
  "iot:Publish",
  "iot:Subscribe",
  "iot:Connect",
  "iot:Receive"
],
"Effect": "Allow",
"Resource": [
  "*"
]
  }
  ]
}
```

- Click on create to complete the policy creation.
- Go to **Manage** section
- Press **Create**, then **Create a single thing**
- Give the new thing a name, then click on Next
- Select **One-click certificate creation (recommended)** by clicking on **Create certificate**
- Once presented with the **Certificate created** page, download all certificates and keys
- Click on the **Activate** button to enable the certificate authentication of the newly created device
- Click on **Attach a policy** and select the policy created in a previous step

For further information, please refer to the full AWS IoT documentation

3.5.2.3 Application setup

- Set **CLIENTCERTFILE** and **CLIENTKEYFILE** defines in **aws_demo.c file** in order to match the certificate and key created in the previous section.
- Set **AWS_BROKER_ADDRESS** to the correct AWS URL. It can be retrieved from AWS IoT **Manage** > **Things** > **Interact** in the HTTPS **Rest API Endpoint** URL.
- Set **CLIENT_ID** to the desired Client ID for your AWS device
- (Optional) if required, change **CACERTFILE** to match the one to be used.

3.5.2.4 Device setup

The application requires the certificates (provided in sample app **certs** subfolder) to be stored in **/mod/ssl_certs/** folder. It can be created with

```
AT#M2MMKDIR=/mod/ssl_certs
```

Certificates can then be loaded with

```
AT#M2MWRITE="/mod/ssl_certs/preload_CACert_01.crt",1468 AT#M2MWRITE="/mod/ssl_certs
IoT.crt",1646
```

providing the file content in RAW mode (for example using the “Transfer Data” button in Telit AT Controller)

For client certificates, the commands will be

```
AT#M2MWRITE="/mod/ssl_certs/xxxxx.crt",yyyy
AT#M2MWRITE="/mod/ssl_certs/xxxxx.key",zzzz
```

PLEASE NOTE: always verify the file sizes to be used in the commands above as they might change

```
Starting AWS IoT Core MQTT demo app. This is v1.1.5 built on Apr 30 2021 09:05:17.
[DEBUG] 15.51  aws_demo:607 - AWS_Task{MQTT_TASK}$ Init MQTT client for AWS
[DEBUG] 15.52  aws_demo:265 - PrepareSSLEnvironment{MQTT_TASK}$ m2mb_ssl_config SNI succeeded
[DEBUG] 15.52  aws_demo:271 - PrepareSSLEnvironment{MQTT_TASK}$ Root CA cert file /mod/ssl_certs/preload_CACert_01.crt
[DEBUG] 15.52  aws_demo:297 - PrepareSSLEnvironment{MQTT_TASK}$ Buffer successfully received from file. 1468 bytes were loaded.
[DEBUG] 15.52  aws_demo:308 - PrepareSSLEnvironment{MQTT_TASK}$ Cross Signed CA cert file /mod/ssl_certs/Amazon-IoT.crt
[DEBUG] 15.52  aws_demo:334 - PrepareSSLEnvironment{MQTT_TASK}$ Buffer successfully received from file. 1646 bytes were loaded.
[DEBUG] 15.52  aws_demo:360 - PrepareSSLEnvironment{MQTT_TASK}$ Client certificate file /mod/ssl_certs/ab71          -certificate.pem.crt
[DEBUG] 15.52  aws_demo:384 - PrepareSSLEnvironment{MQTT_TASK}$ Buffer successfully received from file. 1224 bytes were loaded.
[DEBUG] 15.52  aws_demo:396 - PrepareSSLEnvironment{MQTT_TASK}$ Client Key file /mod/ssl_certs/ab71          -private.pem.key
[DEBUG] 15.52  aws_demo:422 - PrepareSSLEnvironment{MQTT_TASK}$ Buffer successfully received from file. 1679 bytes were loaded.

SSL environment preparation completed
[DEBUG] 15.52  aws_demo:726 - AWS_Task{MQTT_TASK}$ Waiting for registration...
[DEBUG] 15.52  aws_demo:514 - NetCallback{pubTspt_0}$ Module is registered
[DEBUG] 15.52  aws_demo:738 - AWS_Task{MQTT_TASK}$ PDP context initialization
[DEBUG] 17.55  aws_demo:753 - AWS_Task{MQTT_TASK}$ Activate PDP with APN web.omnitel.it on CID 1....
[DEBUG] 18.37  aws_demo:557 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 18.37  aws_demo:561 - PdpCallback{pubTspt_0}$ IP address: 109.114.102.21

Connecting to Server <angy83rl5oizs-ats.iot.eu-west-2.amazonaws.com>:8883...
Done.
[DEBUG] 27.87  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 2, "data": {"RSSI": -72, "tm":"2021-04-30,09:05:50"} }> to topic
device/35308109          /updates
[DEBUG] 27.87  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 30.94  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 3, "data": {"RSSI": -72, "tm":"2021-04-30,09:05:53"} }> to topic
device/353081090         /updates
[DEBUG] 30.94  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 33.99  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 4, "data": {"RSSI": -72, "tm":"2021-04-30,09:05:56"} }> to topic
device/35308109          /updates
[DEBUG] 34.00  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 37.00  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 5, "data": {"RSSI": -72, "tm":"2021-04-30,09:05:59"} }> to topic
device/35308109          /updates
[DEBUG] 37.00  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 40.03  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 6, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:02"} }> to topic
device/35308109          /updates
[DEBUG] 40.03  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 43.05  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 7, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:05"} }> to topic
device/353081            /updates
[DEBUG] 43.05  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 46.13  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 8, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:08"} }> to topic
device/3530810           /updates
[DEBUG] 46.13  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 49.15  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 9, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:11"} }> to topic
device/353081            /updates
[DEBUG] 49.15  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 52.19  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 10, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:14"} }> to topic
device/353081            /updates
[DEBUG] 52.19  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 55.22  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 11, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:17"} }> to topic
device/353081            /updates
[DEBUG] 55.22  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.

Disconnecting from MQTT broker..
[DEBUG] 58.27  aws_demo:878 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 58.27  aws_demo:908 - AWS_Task{MQTT_TASK}$ application exit
[DEBUG] 58.27  aws_demo:918 - AWS_Task{MQTT_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 58.27  aws_demo:924 - AWS_Task{MQTT_TASK}$ Application complete.
```

Figure 99

Data received from a subscriber:

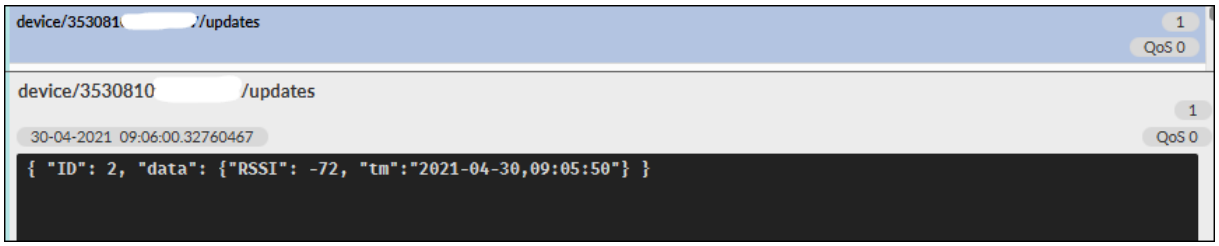

Figure 100

3.5.3 App Manager

Sample application showing how to manage AppZone apps from m2mb code. Debug prints on **USB0**

Features

- How to get how many configured apps are available
- How to get the handle to manage the running app (change start delay, enable/disable)
- How to create the handle for a new binary app, enable it and set its parameters
- How to start the new app without rebooting the device, then stop it after a while.

3.5.3.1 Prerequisites

This app will try to manage another app called “second.bin”, which already exists in the module filesystem and can be anything (e.g. another sample app as GPIO toggle). the app must be built using the flag ROM_START=

in the Makefile to set a different starting address than the main app (by default, 0x40000000). For example, 0x41000000.

3.5.3.2 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- get a non existing app handle and verify it is NULL
- get the current app handle, then get the start delay **set in the INI file (so persistent)**
- change the current app delay value **in the INI file**
- verify that the change has been stored
- get current app state
- create an handle for a second application binary.
- add it to the INI file
- set its execution flag to 0
- get the delay time and the state from INI file for the new app
- get the current set address for the new app
- set the app delay **in RAM, INI will not be affected**.
- start the new app without reboot, using the right set delay
- wait some time, then get the app state and the used RAM amount
- wait 10 seconds, then stop the second app.
- set its execution flag to 1 so it will run at next boot.

```
Starting App Manager demo app. This is v1.0.14-C1 built on Sep 24 2020 12:33:25.
There are 2 configured apps.
Not existing app handle test (should be 0): 0x0
Manager app handle: 0x809e20e0
Manager app delay from nv memory: 5 seconds

Changing Manager app delay time (on non volatile configuration) to 5 seconds..
Manager app delay from nv memory is now 5 seconds
Manager app state is M2MB_APPMNG_STATE_RUN

Trying to get Second app handle...
Second app handle is valid
2nd app delay from nv memory is 1
2nd app current state is M2MB_APPMNG_STATE_READY
Second app current address is 0x41000000
Setting volatile Second app delay (not stored in nvm) to 0 seconds...
Starting Second app on the fly (without reboot)...
Waiting 2 seconds...
2nd app current state is M2MB_APPMNG_STATE_RUN
Second app is running!
Second App is using 475136 bytes of RAM
Stopping Second app now...
wait 10 seconds...
2nd app current state is M2MB_APPMNG_STATE_STOP
Set permanent run permission for Second app.
Done. Second App will also run from next boot-up
```

Figure 101

3.5.4 App update OTA via FTP

Sample application showcasing Application OTA over FTP with AZX FTP. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to connect to a FTP server
- How to download an application binary and update the local version

The app uses a predefined set of parameters. To load custom parameters, upload the `ota_config.txt` file (provided in project's `/src` folder) in module's `/mod` folder, for example with

```
AT#M2MWRITE="/mod/ota_config.txt",<filesize>
```

3.5.4.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage app OTA and start it

`ftp_utils.c`

- Set parameters to default
- Try to load parameters from `ota_config.txt` file
- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Initialize FTP client
- Connect to FTP server and log in
- Get new App binary file size on remote server
- Download the file in `/mod` folder, with the provided name
- Close FTP connection
- Disable PDP context
- Update applications configuration in **app_utils.c**

`app_utils.c`

- Set new application as default
- Delete old app binary
- Restart module

```
Starting FTP APP OTA demo app. This is v1.0.7 built on Apr  7 2020 17:04:05.
[DEBUG] 21.23  ftp_utils.c:447 - msgFTPTask{FTPOTA_TASK}$ INIT
[DEBUG] 21.25  ftp_utils.c:152 - readConfigFromFile{FTPOTA_TASK}$ Reading parameters from file
[DEBUG] 21.26  ftp_utils.c:154 - readConfigFromFile{FTPOTA_TASK}$ Opening /mod/ota_config.txt in read mode..
Set APN to: <<web.omnitel.it>>
Set FTP URL to: <<ftp.telit.com>>
Set FTP PORT to: 21
Set FTP USER to: <<[illegible]>>
Set FTP PASS to: <<[illegible]>>
Set FTP FILE URI to: <</samples/APP_OTA/helloworld.bin>>
Set LOCAL FINAL APP NAME to: <<helloworld.bin>>
Set LOCAL ORIGINAL APP NAME to: <<m2mapz.bin>>
[DEBUG] 23.53  ftp_utils.c:464 - msgFTPTask{FTPOTA_TASK}$ m2mb_os_ev_init success
[DEBUG] 23.54  ftp_utils.c:470 - msgFTPTask{FTPOTA_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.55  ftp_utils.c:478 - msgFTPTask{FTPOTA_TASK}$ Waiting for registration...
[DEBUG] 23.56  ftp_utils.c:371 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 23.56  ftp_utils.c:491 - msgFTPTask{FTPOTA_TASK}$ Pdp context activation
[DEBUG] 23.57  ftp_utils.c:495 - msgFTPTask{FTPOTA_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 25.61  ftp_utils.c:504 - msgFTPTask{FTPOTA_TASK}$ Activate PDP with APN web.omnitel.it on cid 3....
[DEBUG] 26.30  ftp_utils.c:398 - PdpCallback{pubTspt_0}$ Context active
[DEBUG] 26.30  ftp_utils.c:401 - PdpCallback{pubTspt_0}$ IP address: 176.246.110.148
Start ftp client...
[DEBUG] 27.36  ftp_utils.c:533 - msgFTPTask{FTPOTA_TASK}$ Connected.
[DEBUG] 28.87  ftp_utils.c:546 - msgFTPTask{FTPOTA_TASK}$ FTP login successful.
Get remote file /samples/APP_OTA/helloworld.bin size
[DEBUG] 29.31  ftp_utils.c:568 - msgFTPTask{FTPOTA_TASK}$ Done. File size: 116224.
Starting download of remote file /samples/APP_OTA/helloworld.bin into local /mod/helloworld.bin
/samples/APP_OTA/helloworld.bin  4.68% 5440
/samples/APP_OTA/helloworld.bin  9.36% 10880
/samples/APP_OTA/helloworld.bin 14.04% 16320
/samples/APP_OTA/helloworld.bin 18.72% 21760
/samples/APP_OTA/helloworld.bin 23.40% 27200
/samples/APP_OTA/helloworld.bin 28.08% 32640
/samples/APP_OTA/helloworld.bin 32.76% 38080
/samples/APP_OTA/helloworld.bin 37.44% 43520
/samples/APP_OTA/helloworld.bin 42.13% 48960
/samples/APP_OTA/helloworld.bin 46.81% 54400
/samples/APP_OTA/helloworld.bin 51.49% 59840
/samples/APP_OTA/helloworld.bin 56.17% 65280
/samples/APP_OTA/helloworld.bin 60.85% 70720
/samples/APP_OTA/helloworld.bin 65.53% 76160
/samples/APP_OTA/helloworld.bin 70.21% 81600
/samples/APP_OTA/helloworld.bin 74.89% 87040
/samples/APP_OTA/helloworld.bin 79.57% 92480
/samples/APP_OTA/helloworld.bin 84.25% 97920
/samples/APP_OTA/helloworld.bin 88.93% 103360
/samples/APP_OTA/helloworld.bin 93.61% 108800
/samples/APP_OTA/helloworld.bin 97.42% 113220
[DEBUG] 43.54  ftp_utils.c:608 - msgFTPTask{FTPOTA_TASK}$ download successful.
FTP quit...
[DEBUG] 43.77  ftp_utils.c:632 - msgFTPTask{FTPOTA_TASK}$ Deactivating PDP
[DEBUG] 43.77  ftp_utils.c:642 - msgFTPTask{FTPOTA_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 44.20  ftp_utils.c:407 - PdpCallback{pubTspt_0}$ Context deactive
[DEBUG] 45.44  app_utils.c:76 - update_app{FTPOTA_TASK}$ Application successfully configured.
[DEBUG] 45.45  app_utils.c:82 - update_app{FTPOTA_TASK}$ Deleting old application /mod/m2mapz.bin
€ÿStarting. This is v1.0.7 built on Apr  7 2020 17:02:52. LEVEL: 2

 Start Hello world Application  [ version: 2.000000 ]

 Hello world 2.0 [ 000001 ]
 Hello world 2.0 [ 000002 ]
 Hello world 2.0 [ 000003 ]
```

Figure 102

3.5.5 CJSON example:

Sample application showcasing how to manage JSON objects. Debug prints on **USB0**

Features

- How to read a JSON using cJSON library
- How to write a JSON
- How to manipulate JSON objects

3.5.5.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Parse an example string into a JSON object and print the result in a formatted string
- Print some test outcomes (e.g. non existing item correctly not found)
- Retrieve single elements from the parsed JSON object and use them to format a descriptive string
- Delete the JSON object
- Create a new JSON object appending elements to it
- Print the result JSON string from the object

```
Starting Logging demo app. This is v1.0.7 built on Apr  7 2020 08:33:03.
And here is what we got:
{
	"name":	"Atlantic Ocean",
	"format":	{
		"type":	"salt",
		"volume":	310410900,
		"depth":	-8486,
		"volume_percent":	23.300000,
		"tide":	-3.500000,
		"calm":	false,
		"life":	["plankton", "corals", "fish", "mammals"]
	}
}
inexistent key not found
name found: Atlantic Ocean
format found (null)
Our JSON string contains info about an ocean named Atlantic Ocean,
has a volume of 310410900 km^3 of salt water with -8486 meters max depth,
represents 23.3% of total oceans volume,
has an average low tide of -3.5 meters,
hosts a huge number of living creatures such as plankton, corals, fish, mammals,
and is not always calm.

Let's build a TR50 command with a proprety.publish and an alarm.publish for MQTT (no auth).
And here is what we got:
{
	"1":	{
		"command":	"property.publish",
		"params":	{
			"thingKey":	"mything",
			"key": "mykey",
			"value":	123.144000
		}
	},
	"2":	{
		"command":	"alarm.publish",
		"params":	{
			"thingKey":	"mything",
			"key": "mykey",
			"state":	3,
			"msg": "Message."
		}
	}
}
END.
```

Figure 103

3.5.6 Crypto Elliptic Curve Cryptography (ECC) example

Sample application showcasing how to manage Elliptic Curve Cryptography functionalities. Debug prints on **USB0**

Features

- How to initialize ECC contexts A (Alice) and B (Bob). Alice is emulating a remote host, from which a public key is known.
- How to generate keypairs for contexts and export public keys
- how to export keyblobs from a context (a keyblob is encrypted with hw specific keys, and can only be used on the module where it was created)
- How to save a keyblob in secured TrustZone.
- How to reload a keyblob from the TrustZone into an initialized context
- How to sign a message with ECDSA from context B (Bob) and verify it from another context A (Alice) with the signature and public key of Bob.
- How to make Bob and Alice derive a shared session keys using each other's public key.
- How to make Bob and Alice create an AES context with the newly created shared keys, encode data and decode it on the other side

3.5.6.1 Application workflow

`M2MB_main.c`

- Create Bob ECC context, create a keypair and export it in a keyblob
- Open a file in secured Trust Zone, then store the keyblob in it.
- Destroy Bob ECC context
- Recreate Bob ECC context, open the file from Trust Zone and read the keyblob.
- Import the keyblob in Bob context.
- Export Bob public key
- Create Alice ECC context, to simulate an external host. Generate a keypair and export the public key.
- Sign a message with Bob context, generating a signature.
- Use Alice to verify the signed message using Bob's signature and public key
- Derive a shared key for Bob, using Alice's public key
- Create an AES context for Bob
- Import the shared key into the AES context
- Encrypt a message using Bob's AES context.

- Derive a shared key for Alice, using Bob's public key
- Create an AES context for Alice
- Import the shared key into the AES context
- Decrypt the message using Alice's AES context.
- Check the decrypted message and the original one match
- Clear all resources

```
Starting Crypto ECC demo app. This is v1.0.9-C1 built on May 11 2020 16:30:23.

Bob (local) and Alice (remote) scenario
Bob's keypair generated
Bob's keyblob length is 224
Bob exported the keyblob to be securely stored.

Bob already had an item in Secure Data Area, it was removed to create a new one
Bob securely saved the keyblob in Secure Data Area
Releasing resources

Close Bob's context...
Done. Now Bob context does not exist anymore.

Re-initialize Bob Context and load the keyblob from the secure zone
Bob securely loaded the keyblob from the SDA
Import keyblob in Bob's context..
Done. Now export Bob's public key...
Bob's public key successfully exported

Alice's keypair generated
Alice's public key successfully exported

Bob's message signed with ECDSA!
Alice verified bob's message with his pubkey and signature!

------------------------------------------------------------
Bob and Alice will now exchange a message with AES encrypt
------------------------------------------------------------

Bob retrieved the generated shared key size
Bob's shared keyblob length is: 32. Allocate the required memory to store it.
Bob created a shared key using Alice's public key!

Bob created an AEX context to exchange encrypted data with Alice
Bob's AES context imported the shared keyblob
Bob Encrypted the message using AES and the shared key!
Encrypted data:
 94EE531E3B84B2A4EF05502186BFF5DA

Alice retrieved the generated shared key size
Alice's shared keyblob length is: 32. Allocate the required memory to store it.
Alice created a shared key using Bob's public key!

Alice created an AEX context to exchange encrypted data with Bob
Alice's AES context imported the shared keyblob
Alice decrypted the message using AES and the shared key!
Decrypted:
 414094941E8942A4445548035BFAE943

Original, plain message:
 414094941E8942A4445548035BFAE943

Plain and decrypted messages match!
```

Figure 104

3.5.7 EEPROM 24AA256

Sample application showing how to communicate with a MicroChip 24AA256T I2C EEPROM chip using azx eeprom utility APIs. Debug prints on **USB0**

Setup

This demo application requires that: - A0, A1, and A2 pins (1,2,3 chip pins) are connected to ground (pin 4) for device address 0xA0 - Pin 7 (WP) is connected to ground - Pin 6 (SCL) is connected to module GPIO 3 - Pin 5 (SDA) is connected to module GPIO 2 - Pin 4 is connected to one of the ground pins of the module - Pin 8 is connected to 1v8 supply (e.g. VPWRMON pin on the module)

Features

- Initialize the logs on the output channel
- configure the EEPROM utility, setting the slave address and the memory parameters (page size, memory size)
- Write single bytes on a random address
- Read written bytes as a page
- Write data using pages
- Read the new data using pages
- Read again using sequential reading
- Read a single byte from a specific address
- Read next byte using read from current address
- Erase the EEPROM
- Deinit EEPROM utility

3.5.7.1 Application workflow

`M2MB_main.c`

- call azx_eeprom_init() to set the utility parameters (SDA and SCL pins, page and memory sizes)
- call azx_eeprom_writeByte() to store a single byte with value “5” at the address 0x0213
- call azx_eeprom_writeByte() to store a single byte with value “6” at the address 0x0214
- call azx_eeprom_readPages() from address 0x0213 to retrieve the 2 bytes from the EEPROM
- call azx_eeprom_writePages to write 1024 bytes from a buffer, starting from address 0x00
- call azx_eeprom_readPages() again, to read 256 bytes from address 0x00
- call azx_eeprom_readSequentially() to read 256 bytes from 0x00 by without pages (less overhead on I2C protocol)

- call azx_eeprom_readByte() to get a single byte from address 0x00
- call azx_eeprom_readByteFromCurrentAddress() to get a byte from next address (0x01)
- call azx_eeprom_eraseAll() to completely erase the EEPROM memory (this writes 0xFF in each byte)
- call azx_eeprom_readPages from address 0x0213 to get 2 bytes and verify the values have been written to 0xFF
- call azx_eeprom_deinit to close the eeprom handler and the I2C channel

```
Starting I2C EEPROM 24AA256T demo app. This is v1.0.13-C1 built on Nov  3 2020 16:28:23.

Configuring the I2C device...
Opening I2C channel /dev/I2C-160 ( device address is 0xA0 )
Writing 1 byte at address 0x0213...
Done.
Writing 1 byte at address 0x0214...
Done.
Reading the 2 bytes from address 0x0213...
Done. Data: [0xFF 0xFF]

Writing 1024 bytes at address 0x0000..
Done.

Reading 256 bytes from address 0x0000...
Done. Data:
<<ABCDEFGHIJKLMNOPQRSTUVWXYZ.............................................abcdefghijklmnopqrstuvwxyz................

Reading 256 bytes sequentially from address 0x0000...
Done. Data:
<<ABCDEFGHIJKLMNOPQRSTUVWXYZ.............................................abcdefghijklmnopqrstuvwxyz................


Reading 1 byte from address 0x0000...
Done. Data: 'A'

Reading 1 byte from current address (should be 0x0001)...
Done. Data: 'B'


[DEBUG] 17.47  M2MB_main:177 - run_I2C_EEPROM_Demo{M2M_DamsStart}$ Erasing all the eeprom...
[DEBUG] 28.05  M2MB_main:185 - run_I2C_EEPROM_Demo{M2M_DamsStart}$ Done

Reading the 2 bytes from address 0x0213...
Done. Data: [0xFF 0xFF]

Deinit EEPROM...
Done
```

Figure 105

3.5.8 Easy AT example

Sample application showcasing Easy AT functionalities. Debug prints on **USB0**

Features

- Shows how to register custom commands

The application adds two custom commands to the list of available ones:

- AT#MYCMD
- AT#MYINPUT

3.5.8.1 AT#MYCMD

This is a simple parameter-waiting command. It expects one string parameter, and will print it on the logging interface once received. The command simply returns OK

3.5.8.2 AT#MYINPUT

This command expects a numeric parameter, which indicates how many bytes will be received over the interface at most (the command will provide a prompt indicating it is waiting data). Then the data management callback will print when data is received, and if CTRL+Z (0x1A in hex) is received, it will complete the process, printing in the log interface what was received. sending ESC will terminate the process discarding any pending data.

3.5.9 Events

Sample application showcasing events setup and usage. Debug prints on **USB0**

Features

- How to setup OS events with a custom bitmask
- How to wait for events and generate them in callback functions to synchronize blocks of code

3.5.9.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Create an event handler
- Create a timer to generate an event, with a 2 seconds expiration time
- Wait for a specific event bit on the event handler
- At timer expiration, set the same event bit and verify that the code flow went through after the event.

```
Starting Events demo app. This is v1.0.7 built on Apr  7 2020 08:44:29.
[DEBUG] 20.55  M2MB_main.c:171 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
Set the timer attributes structure success.
Timer successfully created
[DEBUG] 20.57  M2MB_main.c:125 - setup_timer{M2M_DamsStart}$ Start the timer, success.
[DEBUG] 22.60  M2MB_main.c:60 - hwTimerCb{pubTspt_0}$ Timer Callback, generate event!
[DEBUG] 22.61  M2MB_main.c:183 - M2MB_main{M2M_DamsStart}$ event occurred!
```

Figure 106

3.5.10 Events - Barrier (multi events)

Sample application showcasing how to setup and use multiple events to create a barrier. Debug prints on **USB0**

Features

- How to setup OS events to be used as a barrier
- How to wait for multiple events in the same point, and generate them in call-back functions to synchronize blocks of code

3.5.10.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create an event handler
- Create a timer to generate an event, with a 3 seconds expiration time
- Create another timer to generate an event, with a 6 seconds expiration time
- Start both timers
- Wait for both event bits on the event handler (each one will be set by one of the timers)
- At first timer expiration, set the first event bit and verify that the code flow does not procede.
- At second timer expiration, set the second event bit and verify that the code flow went through after the event (implementing a barrier).

```
Starting Barrier demo app. This is v1.0.7 built on Apr  7 2020 08:48:30.
[DEBUG] 20.01  M2MB_main.c:179 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
Set the timer attributes structure success.
Timer successfully created with 3000 timeout (ms)
Set the timer attributes structure success.
Timer successfully created with 6000 timeout (ms)
[DEBUG] 23.08  M2MB_main.c:66 - hwTimerCb1{pubTspt_0}$ Timer Callback, generate event 1!
[DEBUG] 26.12  M2MB_main.c:75 - hwTimerCb2{pubTspt_0}$ Timer Callback, generate event 2!
[DEBUG] 26.13  M2MB_main.c:214 - M2MB_main{M2M_DamsStart}$ BOTH events occurred!
```

Figure 107

3.5.11 FOTA example

Sample application showcasing FOTA usage with M2MB API. Debug prints on **USB0**

Features

- How download a delta file from a remote server
- How to apply the delta and update the module firmware

3.5.11.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a main task to manage connectivity.
- create a fota task to manage FOTA and start it with INIT option

`fota.c`

fotaTask()

- Initialize FOTA system then reset parameters.
- Check current FOTA state, if not in IDLE, return error.
- Send a message to mainTask so networking is initialized.
- after PdPCallback() notifies the correct context activation, configure the fota client parameters such as FTP server URL, username and password
- get delta file from server. when it is completed, FOTADownloadCallback is called.
- If delta download went fine, check it.
- If delta file is correct, apply it. Once complete, restart the module.

mainTask()

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context. Event will be received on **PdP-Callback** function
- Disable PDP context when required to stop the app

PdpCallback()

- When PDP context is enabled, send a message to fotaTask to start the download

```
Starting FOTA demo app. This is v1.1.7 built on Jun 11 2021 12:20:43.
[DEBUG] 23.60  fota:187 - fotaTask{FOTA_TASK}$ Init FOTA...

Session file not present, procede with FOTA...
[DEBUG] 23.61  fota:236 - fotaTask{FOTA_TASK}$ m2mb_fota_reset PASS
[DEBUG] 23.61  fota:260 - fotaTask{FOTA_TASK}$ m2mb_fota_state_get  M2MB_FOTA_STATE_IDLE
[DEBUG] 23.62  fota:379 - mainTask{MAIN_TASK}$ INIT
[DEBUG] 23.62  fota:392 - mainTask{MAIN_TASK}$ m2mb_os_ev_init success
[DEBUG] 23.63  fota:398 - mainTask{MAIN_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.63  fota:405 - mainTask{MAIN_TASK}$ Waiting for registration...
[DEBUG] 23.64  fota:131 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 23.65  fota:418 - mainTask{MAIN_TASK}$ Pdp context initialization
[DEBUG] 25.70  fota:431 - mainTask{MAIN_TASK}$ Activate PDP with APN web.omnitel.it on cid 1....
[DEBUG] 35.42  fota:152 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 35.43  fota:155 - PdpCallback{pubTspt_0}$ IP address: 2.41.116.139

[DEBUG] 35.43  fota:285 - fotaTask{FOTA_TASK}$
Trying to download "samples/FOTA/37.00.003.3_to_37.00.003.1_ME310G1_NANVWWAU.bin" delta file...
[DEBUG] 35.45  fota:295 - fotaTask{FOTA_TASK}$ m2mb_fota_get_delta OK - Waiting for the completion callback
[DEBUG] 119.43  fota:96 - FOTADownloadCallBack{pubTspt_0}$ FOTA download Success - performing packet validation...
[DEBUG] 119.44  fota:301 - fotaTask{FOTA_TASK}$ Validating delta file...
[DEBUG] 156.36  fota:317 - fotaTask{FOTA_TASK}$ Packet is valid, start update...
[DEBUG] 156.40  fota:329 - fotaTask{FOTA_TASK}$ m2mb_fota_start PASS
[DEBUG] 158.36  fota:342 - fotaTask{FOTA_TASK}$
Rebooting...After reboot there will be the new FW running on module!

#OTAEV: Module Upgraded To New Fw
Starting FOTA demo app. This is v1.1.7 built on Jun 11 2021 12:20:43.
[DEBUG] 29.24  fota:187 - fotaTask{FOTA_TASK}$ Init FOTA...

Session file is already present, stop.
```

Figure 108

3.5.12 FOTA from Local File example

Sample application that shows how perform FOTA upgrade using a delta file stored into file system. Debug prints on **USB0**

Features

- How to store and get FOTA upgrade information to/from a file
- How to get delta file from module file system
- How to apply the delta and update module firmware

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Check if module has been already upgraded or needs to be upgraded reading FOTA upgrade status from a file
- Create a fota task to manage FOTA and start it with INIT option

smartFotaTask() - Initialize FOTA system then reset parameters. - Get FOTA partiton size and block size - Copy delta file from file system to FOTA paartition. when it is completed, FOTADownloadCallback is called. - If delta file is correct, apply it. Once complete, write FOTA status flag and current fw version to a file, restart the module.

```
Starting FOTA application from local delta file demo app. This is v1.0.14-C1 built on May 16 2022 15:50:38.


Check FOTA upgrade status in file system

Module current fw version is:
37.00.214-P0C.210001
M0C.200003
P0C.210001
A0C.210000
[WARN ] 13.83  M2MB_main:190 - FOTAUpgradeStatusInit{M2M_DamsStart}$ File doesn't exist create it, first app execution
File created, store current fw version and fota upgrade flag=0

Read stored data

FOTA Flag: 0 Module previous fw version:
37.00.214-P0C.210001
M0C.200003
P0C.210001
A0C.210000

Start FOTA process
[DEBUG] 15.42  M2MB_main:450 - smartFotaTask{FOTA_TASK}$ INIT
[DEBUG] 15.43  M2MB_main:461 - smartFotaTask{FOTA_TASK}$ m2mb_fota_init success
[DEBUG] 15.44  M2MB_main:464 - smartFotaTask{FOTA_TASK}$
Get block and FOTA partition size
OTA blockSize: 131072
OTA partitionSize: 11272192
[DEBUG] 15.47  M2MB_main:472 - smartFotaTask{FOTA_TASK}$
 Copy delta file from File system to FOTA partition
[DEBUG] 15.47  M2MB_main:277 - copyFromFileSystemToDelta{FOTA_TASK}$ File size: 746552
[DEBUG] 15.87  M2MB_main:480 - smartFotaTask{FOTA_TASK}$ Delta file writing completed
[DEBUG] 15.88  M2MB_main:484 - smartFotaTask{FOTA_TASK}$
Delta file check...
[DEBUG] 15.89  M2MB_main:347 - check_fota_delta{FOTA_TASK}$ -- check_fota_delta...
[DEBUG] 54.73  M2MB_main:366 - check_fota_delta{FOTA_TASK}$ Fota check integrity PASS
[DEBUG] 54.74  M2MB_main:492 - smartFotaTask{FOTA_TASK}$ ...delta file OK
[DEBUG] 54.75  M2MB_main:496 - smartFotaTask{FOTA_TASK}$
--> Start update...
[DEBUG] 56.83  M2MB_main:403 - startUpdate{FOTA_TASK}$
Reboot module to start delta deployment
```

Figure 109

3.5.13 FTP

Sample application showcasing FTP client demo with AZX FTP. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to connect to a FTP server
- How to exchange data with the server

3.5.13.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage FTP client and start it

`ftp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init FTP client and set the debug function for it
- Connect to the server
- Perform log in
- Check remote file size and last modification time
- Download file from server to local filesystem. A data callback is set to report periodic info about the download status
- Upload the same file to the server with a different name. A data callback is set to report periodic info about the upload status
- Download another file content in a buffer instead of a file. A data callback is set to report periodic info about the download status
- Close the connection with FTP server
- Disable PDP context

```
Starting FTP demo app. This is v1.0.7 built on Apr  7 2020 11:17:36.
[DEBUG] 21.23  ftp_test.c:290 - msgFTPTask{FTP_TASK}$ INIT
[DEBUG] 21.23  ftp_test.c:304 - msgFTPTask{FTP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.23  ftp_test.c:310 - msgFTPTask{FTP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.23  ftp_test.c:318 - msgFTPTask{FTP_TASK}$ Waiting for registration...
[DEBUG] 21.25  ftp_test.c:214 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 21.26  ftp_test.c:331 - msgFTPTask{FTP_TASK}$ Pdp context activation
[DEBUG] 21.27  ftp_test.c:335 - msgFTPTask{FTP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.31  ftp_test.c:344 - msgFTPTask{FTP_TASK}$ Activate PDP with APN web.omnitel.it on cid 3....
[DEBUG] 24.09  ftp_test.c:241 - PdpCallback{pubTspt_0}$ Context active
[DEBUG] 24.10  ftp_test.c:244 - PdpCallback{pubTspt_0}$ IP address: 176.244.166.181
Start ftp client...
[DEBUG] 24.82  ftp_test.c:373 - msgFTPTask{FTP_TASK}$ Connected.
[DEBUG] 26.32  ftp_test.c:386 - msgFTPTask{FTP_TASK}$ FTP login successful.
Get remote file /samples/pattern_big.txt size
[DEBUG] 26.69  ftp_test.c:428 - msgFTPTask{FTP_TASK}$ Done. File size: 20026.
Get remote file /samples/pattern_big.txt last modification date
[DEBUG] 26.89  ftp_test.c:450 - msgFTPTask{FTP_TASK}$ Done. File last mod date: 20200407090654
.

Starting download of remote file /samples/pattern_big.txt into local /mod/_pattern_big.txt
/samples/pattern_big.txt 47.54% 9520
/samples/pattern_big.txt 100.00% 20026
[DEBUG] 29.75  ftp_test.c:488 - msgFTPTask{FTP_TASK}$ download successful.
[DEBUG] 29.76  ftp_test.c:522 - msgFTPTask{FTP_TASK}$
Local file /mod/_pattern_big.txt size: 20026

Starting upload of local file /mod/_pattern_big.txt
/mod/_pattern_big.txt 81.81% 16384
Upload successful.

Starting download of remote file /samples/pattern.txt into local buffer
Getting remote file /samples/pattern.txt size..
[DEBUG] 32.97  ftp_test.c:583 - msgFTPTask{FTP_TASK}$ Done. File size: 988.
Starting download of remote file /samples/pattern.txt to buffer
[DEBUG] 34.08  ftp_test.c:145 - buf_data_cb{FTP_TASK}$ Received START event
[DEBUG] 34.09  ftp_test.c:149 - buf_data_cb{FTP_TASK}$ Received DATA: 988 bytes on buffer 0x400399e0
[DEBUG] 34.26  ftp_test.c:153 - buf_data_cb{FTP_TASK}$ Received END event
[DEBUG] 34.26  ftp_test.c:623 - msgFTPTask{FTP_TASK}$ Download successful. Received 988 bytes<<<
0   |---------|   |---------|   |---------|   |---------|   |---------|  *
1   |    A    |   |    A    |   |    A    |   |    A    |   |    A    |   *
2   |   AAA   |   |   AAA   |   |   AAA   |   |   AAA   |   |   AAA   |    *
3   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |     *
4   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |      *
5   |AAAAAAAAA|   |AAAAAAAAA|   |AAAAAAAAA|   |AAAAAAAAA|   |AAAAAAAAA|       *
6   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |      *
7   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |     *
8   |   AAA   |   |   AAA   |   |   AAA   |   |   AAA   |   |   AAA   |    *
9   |    A    |   |    A    |   |    A    |   |    A    |   |    A    |   *
10  |---------|   |---------|   |---------|   |---------|   |---------|  *
11                                                                      *
12  |---------|   |---------|   |---------|   |---------|   |--->>>
```

Figure 110

3.5.14 File System example

Sample application showcasing M2MB File system API usage. Debug prints on **USB0**

Features

- How to open a file in write mode and write data in it
- How to reopen the file in read mode and read data from it

3.5.14.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Print welcome message
- Open file in write mode
- Write data in file
- Close file
- Reopen file in read mode
- Read data from file and print it
- Close file and delete it

```
Starting FileSystem demo app. This is v1.0.7 build on Mar 26 2020 09:50:19. LEVEL: 2
Opening /my_text_file.txt in write mode..
Buffer written successfully into file. 15 bytes were written.
Closing file.
Opening /my_text_file.txt in read only mode..
Received 15 bytes from file:
<Hello from file>
Closing file.
Deleting File
File deleted
App Completed
```

Figure 111

3.5.15 GNSS example

Sample application showing how to use GNSS functionality. Debug prints on **USB0**

Features

- How to enable GNSS receiver on module
- How to collect location information from receiver

Note: on MEx10G1 product family both M2MB_GNSS_SERVICE_NMEA_REPORT and M2MB_GNSS_SERVICE_POSITION_REPORT services are available, while on ME910C1 product family only M2MB_GNSS_SERVICE_POSITION_REPORT is available

3.5.15.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print a welcome message
- Create GNSS task and send a message to it

`gps_task.c` - Init Info feature and get module type - Init gnss, enable position/NMEA report and start it. - When a fix or a NMEA sentence is available, a message will be printed by the GNSS callback function

```
Starting GNSS demo app. This is v1.1.4 built on Oct  1 2021 15:27:44.

Model: ME910C1-E2

m2mb_gnss_enable, POSITION OK

m2mb_gnss_start OK, waiting for position/nmea sentences...
latitude_valid: 1 - latitude: 45.713643
longitude_valid: 1 - longitude: 13.738041
altitude_valid: 1 - altitude: 195.000000
uncertainty_valid: 1 - uncertainty: 95.000000
velocity_valid: 1 - codingType: 0
speed_horizontal: 0.650000
bearing: 0.000000
timestamp_valid: 1 -timestamp: 1633095357439
speed_valid: 1 - speed: 1.471360


***** Wait 120 seconds and then stop GPS *****
```

Figure 112

3.5.16 GPIO interrupt example

Sample application showing how to use GPIOs and interrupts. Debug prints on **USB0**

Features

- How to open a GPIO in input mode with interrupt
- How to open a second GPIO in output mode to trigger the first one

3.5.16.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Open GPIO 4 as output
- Open GPIO 3 as input and set interrupt for any edge (rising and falling). **A jumper must be used to short GPIO 3 and 4 pins.**
- Toggle GPIO 4 status high and low every second
- An interrupt is generated on GPIO 3

```
Starting GPIO interrupt demo app. This is v1.0.7 built on Mar 26 2020 16:33:01.
Setting gpio 3 interrupt...
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
```

Figure 113

3.5.17 GTP example

Sample application that shows hot to get the position using GTP feature. Debug prints on **USB0**

Features

- How to init and enable GTP feature
- How to get the position using GTP

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Init NET functionality and wait for module to be registered
- Init PDP functionality and set APN on CID1
- Init GTP functionality
- Check if GTP is already enabled. If not enable it and reboot module
- If GTP is enabled, get position

```
Starting GTP demo app. This is v1.1.8 built on Dec 20 2022 15:29:19.

[DEBUG] 15.08  M2MB_main:237 - M2MB_main{M2M_DamsStart}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 15.09  M2MB_main:245 - M2MB_main{M2M_DamsStart}$ Waiting for registration...
[DEBUG] 15.10  M2MB_main:253 - M2MB_main{M2M_DamsStart}$ Pdp context setting
[DEBUG] 15.11  M2MB_main:82 - NetCallback{pubTspt_0}$ Module is registered to cell 0x20!
[DEBUG] 15.11  M2MB_main:257 - M2MB_main{M2M_DamsStart}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 15.13  M2MB_main:268 - M2MB_main{M2M_DamsStart}$ m2mb_pdp_deinit returned M2MB_RESULT_SUCCESS

Check if GTP has been already enabled
[DEBUG] 16.19  M2MB_main:312 - M2MB_main{M2M_DamsStart}$ GTP status: 1 => Enabled
[DEBUG] 16.20  M2MB_main:317 - M2MB_main{M2M_DamsStart}$ Get the position...

m2mb_gnss_GTP OK, waiting for position...
[DEBUG] 107.96  M2MB_main:160 - gnssCallbackFN{pubTspt_0}$ gnssCallback event: 1
GTP position got
latitude: 41.900002
longitude: 12.500000
altitude: 0.000000
altitudeMeanSeaLevel: -0.000000
accuracy: 0.000000
speed: -0.000000
bearing: -0.000000
timestamp: 0
verticalAccuracy 0.000000
speedAccuracy -0.000000
bearingAccuracy: -0.000000
```

Figure 114

3.5.18 General_INFO example

Sample application prints some Module/SIM information as IMEI, fw version, IMSI and so on; it prints also some information about registration. Debug prints on **USB0**

Features

- How to print some Module information as IMEI, FW version etc
- How to print some SIM information as IMSI, ICCID
- How to get and print some informatio about Module registration as Netowrk Operator, AcT, RSSI, etc

3.5.18.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Init NET functionality
- Init INFO functionality
- Get and print Module and SIM info
- Wait form module to register to network
- Get and print registration INFO

```
Starting. This is v1.1.4 built on Mar 31 2021 09:56:03. LEVEL: 2


 Start General INFO application [ version: 1.000000 ]


=======================

MODULE ME910C1-E2 INFO

=======================

MANUFACTURER: Telit

IMEI: 353080091125422

MODEM FIRMWARE VERSION: M0B.700005

PACKAGE VERSION:
30.00.709-B005-P0B.700100
M0B.700005
P0B.700100
A0B.700000


========

SIM INFO

========

IMSI: 222015602268648

ICCID: 89390100001138084906


============================

Waiting for registration...

============================

Module is registered to HOME network cellID 0x5221
NETWORK OPERATOR (mcc mnc): 222 01
Network Technology 2G (AcT: 0) RSSI: -81
```

Figure 115

3.5.19 HTTP Client

Sample application showing how to use HTTPs client functionalities. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to initialize the http client, set the debug hook function and the data callback to manage incoming data
- How to perform GET, HEAD or POST operations (GET also with single range support)

NOTE: the sample app has an optional dependency on azx_base64.h if basic authentication is required (refer to `HTTP_BASIC_AUTH_GET` define in `M2MB_main.c` for further details)

3.5.19.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage HTTP client and start it

`httpTaskCB`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create HTTP client options and initialize its functionality
- Create HTTP SSL config and initialize the SSL options
- Configure data management options for HTTP client
- Appy all configurations to HTTP client
- Perform a GET request to a server
- Disable PDP context

`DATA_CB`

- Print incoming data
- Set the abort flag to 0 to keep going.

```
Starting HTTP(s) client demo app. This is v1.1.15 built on Nov 28 2022 12:12:27.
[DEBUG] 17.12  read_parame:101 - readConfigFromFile{HttpClient}$ Reading parameters from file
[DEBUG] 17.13  read_parame:103 - readConfigFromFile{HttpClient}$ Opening /mod/HTTP_Client_config.txt in read mode..
Set APN to: <<web.omnitel.it>>
Set APN USER to: <<>>
Set APN PASS to: <<>>
Set CID to: 3
Set CACERTFILE to: <<>>
Set CLIENTCERTFILE to: <<>>
Set CLIENTKEYFILE to: <<>>
Set REQUEST_TYPE to: 2
Set SERVER to: <<http://linux-ip.net>>
[DEBUG] 17.16  m2mb_HTTP_t:267 - activatePdP{HttpClient}$ m2mb_os_ev_init success
[DEBUG] 17.17  m2mb_HTTP_t:273 - activatePdP{HttpClient}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 17.18  m2mb_HTTP_t:281 - activatePdP{HttpClient}$ Waiting for registration...
[DEBUG] 17.19  m2mb_HTTP_t:101 - NetCallback{pubTspt_0}$ Module is registered to cell 0x468E!
[DEBUG] 17.20  m2mb_HTTP_t:295 - activatePdP{HttpClient}$ Pdp context initialization
[DEBUG] 19.26  m2mb_HTTP_t:305 - activatePdP{HttpClient}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 20.16  m2mb_HTTP_t:128 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 20.18  m2mb_HTTP_t:131 - PdpCallback{pubTspt_0}$ IP address: 176.243.212.114
Performing a GET request...
Connecting to linux-ip.net:80//

Socket connected!

<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="author" content="Martin A. Brown" />
    <meta name="robots" content="index, follow"/>

    <meta property="og:title" content="http://linux-ip.net/"/>
    <meta property="og:url" content="http://linux-ip.net"/>
    <meta property="og:site_name" content="http://linux-ip.net/"/>
    <meta property="og:type" content="website"/>

    <link rel="canonical" href="http://linux-ip.net" />

    <title>http://linux-ip.net/</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" type="text/css" href="//netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.css" />
    <link rel="stylesheet" type="text/css" href="//netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css" />
    <link rel="stylesheet" type="text/css" href="http://linux-ip.net/theme/css/main.css" />

</head>
  ▪▪▪
    <footer id="site-footer">
        <div class="row-fluid">
            <div class="span10 offset1">
                <address>
                    <p>
                        Powered by <a href="http://getpelican.com/">Pelican</a>
                        and <a href="http://python.org">Python</a>.
                        Theme based on <a href="http://github.com/jsliang/pelican-fresh/">Fresh</a>
                        by <a href="http://jsliang.com/">jsliang</a>
                    </p>
                </address>
            </div>
        </div>
    </footer>
</body>
</html>
Result: 200
[DEBUG] 26.80  m2mb_HTTP_t:137 - PdpCallback{pubTspt_0}$ Context deactivated!
```

Figure 116

3.5.20 HW Timer (Hardware Timer)

The sample application shows how to use HW Timers M2MB API. Debug prints on **USB0**

Features

- How to open configure a HW timer
- How to use the timer to manage recurring events

3.5.20.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create hw timer structure
- Configure it with 100 ms timeout, periodic timer (auto fires when expires) and autostart
- Init the timer with the parameters
- Wait 10 seconds
- Stop the timer

`TimerCb`

- Print a message with an increasing counter

```
Starting HW Timers demo app. This is v1.0.7 built on Mar 26 2020 13:04:14.
[DEBUG] 14.06  M2MB_main.c:114 - M2MB_main{M2M_DamsStart}$ Set the timer attributes structure: success.
Timer successfully created
Start the timer, success.
[DEBUG] 14.18  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [0]
[DEBUG] 14.28  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [1]
[DEBUG] 14.38  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [2]
[DEBUG] 14.48  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [3]
[DEBUG] 14.58  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [4]
[DEBUG] 14.69  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [5]
[DEBUG] 14.79  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [6]
[DEBUG] 14.88  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [7]
[DEBUG] 14.98  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [8]
[DEBUG] 15.08  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [9]


[DEBUG] 23.90  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [96]
[DEBUG] 24.01  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [97]
[DEBUG] 24.11  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [98]
Stop a running timer: success
Application end
```

Figure 117

3.5.21 Hello World

The application prints “Hello World!” over selected output every two seconds. Debug prints on **USB0**, using AZX log example functions

Features

- How to open an output channel using AZX LOG sample functions
- How to print logging information on the channel using AZX LOG sample functions

3.5.21.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Print “Hello World!” every 2 seconds in a while loop

```
Starting. This is v1.0.7 built on Mar 26 2020 09:34:16. LEVEL: 2

 Start Hello world Application  [ version: 2.000000 ]

 Hello world 2.0 [ 000001 ]
 Hello world 2.0 [ 000002 ]
 Hello world 2.0 [ 000003 ]
 Hello world 2.0 [ 000004 ]
 Hello world 2.0 [ 000005 ]
 Hello world 2.0 [ 000006 ]
 Hello world 2.0 [ 000007 ]
 Hello world 2.0 [ 000008 ]
 Hello world 2.0 [ 000009 ]
```

Figure 118

3.5.22 I2C example

Sample application showing how to communicate with an I2C slave device. Debug prints on **USB0**

Features

- How to open a communication channel with an I2C slave device
- How to send and receive data to/from the slave device

Setup

- Connect sensor VDD to 1v8 supply (e.g. Vaux/PwrMon pin of the module)
- Connect sensor GND to a GND pin of the module
- Connect sensor SDA to module GPIO2
- Connect sensor SCL to module GPIO3

3.5.22.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open I2C bus, setting SDA an SCL pins as 2 and 3 respectively
- Set registers to configure accelerometer -Read in a loop the 6 registers carrying the 3 axes values and show the g value for each of them

```
Starting I2C demo app. This is v1.0.7 built on Mar 26 2020 16:50:40.

Configuring the Kionix device...
opening channel /dev/I2C-30
[DEBUG] 20.18  M2MB_main.c:218 - test_I2C{M2M_DamsStart}$ -
WHOAMI content: 0x01
Configuring I2C Registers - Writing 0x4D into 0x1D register (CTRL_REG3)...
Write: success

I2C reading data from 0x1D register (CTRL_REG3)...
Read: success.
Accelerometer Enabled. ODR tilt: 12.5Hz, ODR directional tap: 400Hz, ORD Motion Wakeup: 50Hz
Configuring I2C Registers - Writing 0xC0 into 0x1B register (CTRL_REG1)...
Write: success

I2C reading data from 0x1B register (CTRL_REG1)...
Read: success.
Accelerometer Enabled. Operative mode, 12bit resolution
I2C read axes registers
------------
Reading Success.

X: -0.050 g
Y: -0.046 g
Z: 1.006 g
Reading Success.

X: -0.049 g
Y: -0.044 g
Z: 1.004 g
Reading Success.

X: -0.052 g
Y: -0.044 g
Z: 1.007 g
Reading Success.

X: -0.048 g
Y: -0.045 g
Z: 1.005 g
```

Figure 119

3.5.23 I2C Combined

Sample application showing how to communicate with an I2C slave device with I2C raw mode. Debug prints on MAIN UART

Features

- How to open a communication channel with an I2C slave device
- How to send and receive data to/from the slave device using raw mode API

Setup

- Connect sensor VDD to 1v8 supply (e.g. Vaux/PwrMon pin of the module)
- Connect sensor GND to a GND pin of the module
- Connect sensor SDA to module GPIO2
- Connect sensor SCL to module GPIO3

3.5.23.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Open I2C bus, setting SDA an SCL pins as 2 and 3 respectively
- Set registers to configure accelerometer -Read in a loop the 6 registers carrying the 3 axes values and show the g value for each of them

```
Starting I2C raw demo app. This is v1.0.13-C1 built on Jul 30 2020 11:28:18.

Configuring the I2C device...
Opening I2C channel /dev/I2C-30 ( device address is 0x0F << 1 )
Accelerometer Enabled. ODR tilt: 12.5Hz, ODR directional tap: 400Hz, ORD Motion Wakeup: 50Hz
Accelerometer Enabled. Operative mode, 12bit resolution
I2C read axes registers
------------

X: 0.000 g
Y: 0.000 g
Z: 0.000 g

X: -0.270 g
Y: 0.016 g
Z: 0.917 g

X: -0.268 g
Y: 0.013 g
Z: 0.925 g

X: -0.271 g
Y: 0.015 g
Z: 0.922 g

X: -0.267 g
Y: 0.016 g
Z: 0.918 g

X: -0.274 g
Y: 0.019 g
Z: 0.915 g
```

Figure 120

3.5.24 Little FileSystem 2

Sample application showing how use lfs2 porting with RAM disk and SPI data flash. Debug prints on **USB0**

Features

- How to create and manage Ram Disk
- How to manage file-system in Ram disk partition
- How to create and manage SPI Flash memory partition
- How to manage file-system in SPI Flash memory partition

3.5.24.1 Application workflow

`M2MB_main.c`

- Init logging system
- Call Ram Disk tests
- Call Flash memory tests

`ram_utils_usage.c`

- Initialize Ram Disk
- Format and Mount partition
- List files
- Files creation and write content
- List files
- Read files
- Unmount and Release resources

`spi_utils_usage.c` - Initialize SPI Flash chip - Initialize SPI Flash Disk - Format and Mount partition - List files - Files creation and write content - List files - Read files - Delete files - Directories creation and deletion - Unmount and Release resources

Notes:

For SPI Flash a JSC memory is used with chip select pin connected to module GPIO2 pin. For better performances, a 33kOhm pull-down resistor on SPI clock is suggested. Please refer to SPI_echo sample app for SPI connection details.

For LE910Cx (both Linux and ThreadX based devices), `AT#SPIEN=1` command must be sent once before running the app

```
Starting lfs2 demo app. This is v1.0.14-C1 built on Oct 22 2020 09:43:08.

>>>>>>> Starting RAMDiskDemo ...
[DEBUG] 18.28  azx_lfs_uti:125 - azx_ram_initialize{M2M_DamsStart}$ Ram Memory allocated correctly from 0x40042228 to 0x40046228!!

Mounting partition...

Formatting...

Mounting...


Mounted partition...

<><><><>fileListUtils

List:
., 0, 2
.., 0, 2

file_name: file000.txt
size: 10
buffer: content000
mode: 0
RAM TYPE size: 10000


File created and closed: file000.txt


<><><><>fileListUtils
____INSIDE --->file000.txt, 10, 1

List:
., 0, 2
.., 0, 2
file000.txt, 10, 1

------>File reading

File: file000.txt, Size: 10, Buffer: content000

Nand released
Partition unmounted
[DEBUG] 20.31  azx_lfs_uti:165 - azx_ram_releaseResources{M2M_DamsStart}$ Ram Memory released correctly!!
```

```
>>>>>>> Starting FlashDiskDemo ...
Starting initialization...

table id[0] = 191
table id[1] = 1
table id[2] = 0

nandLFSCallback Callback event <1>
NAND Callback event: NAND_JSC_INITIALIZED <1>
nandLFSCallback Callback event <1>
NAND Callback event: NAND_JSC_INITIALIZED <1>

Mounting partition...

Formatting...
spiErase: address = 0, len = 131072

spiErase: address = 131072, len = 131072


Mounting...


Mounted partition...


<><><><>fileListUtils

List:
., 0, 2
.., 0, 2


Formatting...
spiErase: address = 0, len = 131072

spiErase: address = 131072, len = 131072


Mounting...


Mounted partition...


<><><><>fileListUtils

List:
., 0, 2
.., 0, 2

file_name: file000.txt
size: 10
buffer: content000
mode: 0


File created and closed: file000.txt
```

```
<><><><>fileListUtils

List:
., 0, 2
.., 0, 2
file000.txt, 10, 1
file001.txt, 10, 1
file002.txt, 10, 1
file003.txt, 10, 1
file004.txt, 10, 1

------>File reading

File: file000.txt, Size: 10, Buffer: content000


File: file004.txt, Size: 10, Buffer: content004


File: file002.txt, Size: 10, Buffer: content002
------>File removing
file001.txt<<<<<<<


File removed: file001.txt
file000.txt<<<<<<<


File removed: file000.txt
file004.txt<<<<<<<


File removed: file004.txt


<><><><>fileListUtils

List:
., 0, 2
.., 0, 2
file002.txt, 10, 1
file003.txt, 10, 1

spiErase: address = 59637760, len = 131072

[DEBUG] 58.61  azx_lfs_uti:648 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory created: dir000!!
[DEBUG] 59.78  azx_lfs_uti:631 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory already exists: dir000!!
spiErase: address = 59899904, len = 131072

[DEBUG] 61.70  azx_lfs_uti:648 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory created: dir001!!
spiErase: address = 60162048, len = 131072

[DEBUG] 63.67  azx_lfs_uti:648 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory created: dir002!!
```

```
<><><><>fileListUtils

List:
., 0, 2
.., 0, 2
dir000, 0, 2
dir001, 0, 2
dir002, 0, 2
file002.txt, 10, 1
file003.txt, 10, 1



<><><><>fileListUtils

List:
., 0, 2
.., 0, 2
dir001, 0, 2
dir002, 0, 2
file002.txt, 10, 1
file003.txt, 10, 1

Nand released
Partition unmounted

Unmounted process ended...

testAllInOneFunction ended...
```

3.5.25 LWM2M

Sample application showcasing LWM2M client usage with M2MB API. Debug prints on **USB0**

Features

- Configure LWM2M client and enable it
- Create an instance of a custom object
- Set an integer value on a read only resource
- Set two integer values on a multi-instance read only resource
- write a string on a read/write resource
- Manage exec requests from the portal
- Manage write, read and monitoring resources changed from the portal

Requirements

This application expects the user to configure the PDP context ID 1 with the proper APN. it can be done with the following AT command:

```
AT+CGDCONT=1,"IPV4V6","<user apn>"
```

Depending on the Mobiler Network Operator and Access Technology, the APN might be automatically set by the network itself. In this case, nothing must be done by the user.

3.5.25.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the LWM2M client and start it

`lwm2m_demo.c`

`msgLWM2MTask` - Check registration status

- Initialize LWM2M client,
- Check for XML file fo custom object
- Enable unsolicited messages from client
- Create a task (lwm2m_taskCB is its callback function)to manage events from Portal
- Enable LwM2M client

- Create a new instance for the custom object
- Wait for client to register to Portal
- Performs all operations (set, read, get, write) on the related resources
- Performs a set with notify ack enabled
- Wait for events from server

`lwm2mIndicationCB`

- Manage events arriving from client (operations completion status and unsolicited events)
- Run lwm2m_taskCB when a monitored resource changes, to manage the action to be done

3.5.25.2 Device Profile upload

LWM2M resources demo device profile must be imported to have a real-time update of resources values on the LWM2M browser.

To do so, import the file `json/lwm2m_resources_demo.json` (provided with the sample files) on section `Developer` > `Device profiles` of OneEdge IoT portal:

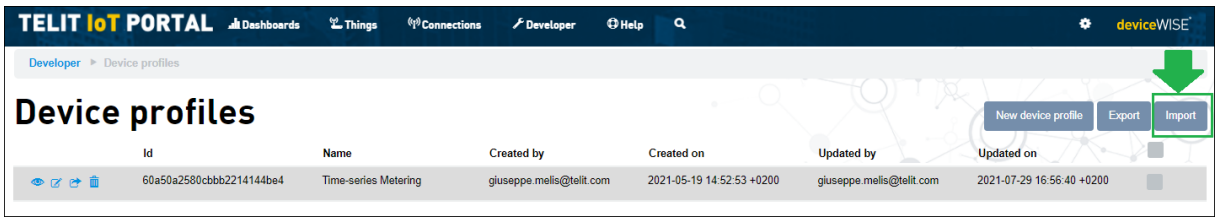

Figure 121

3.5.25.3 Custom Object configuration

The XML file content must be loaded on the Telit IoT Portal for the demo application to be fully executed.

First, enter Developer section from the top menu

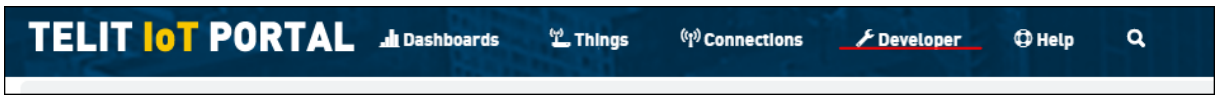

Figure 122

Choose Object Registry

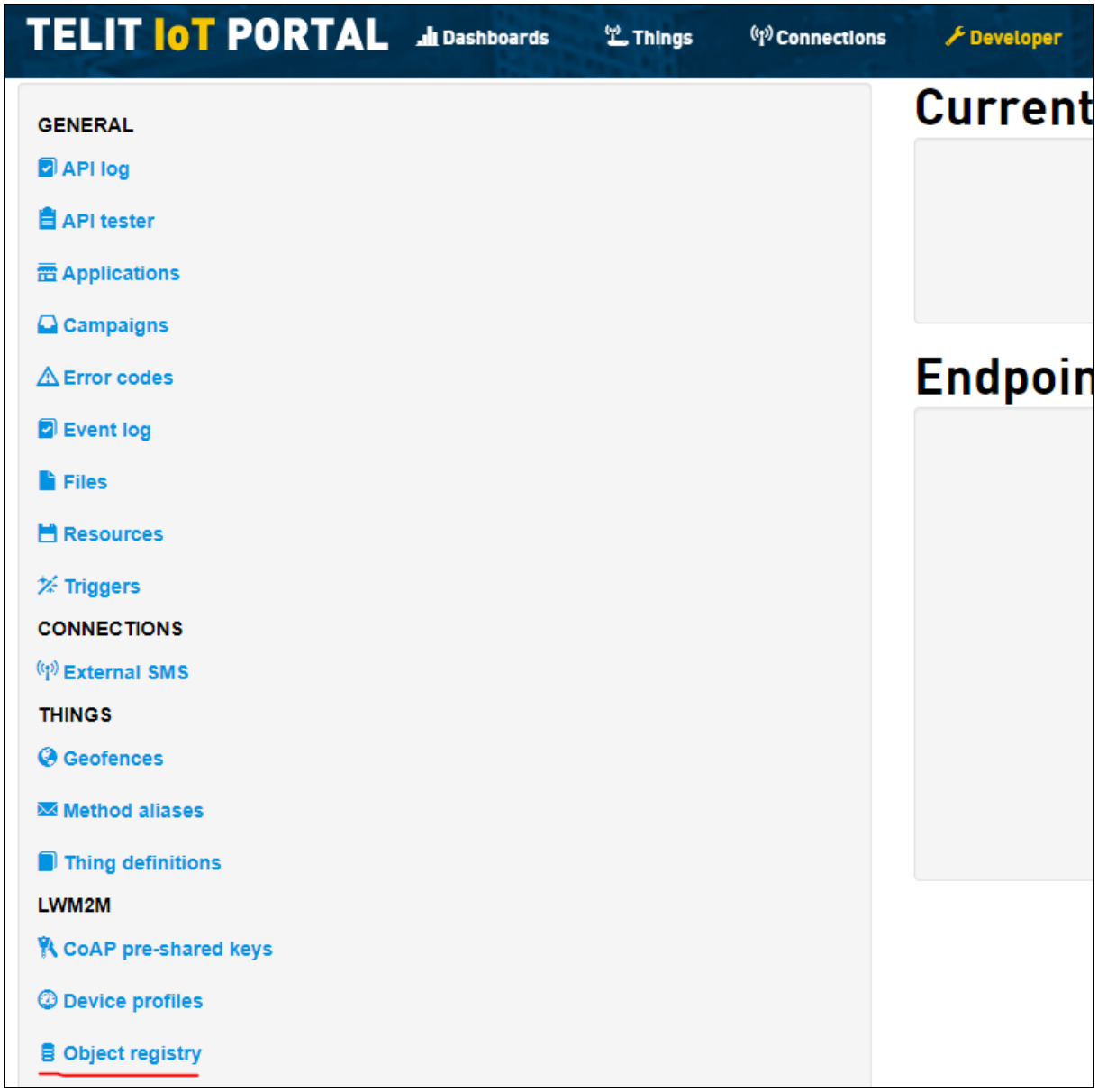

Figure 123

Create a New Object

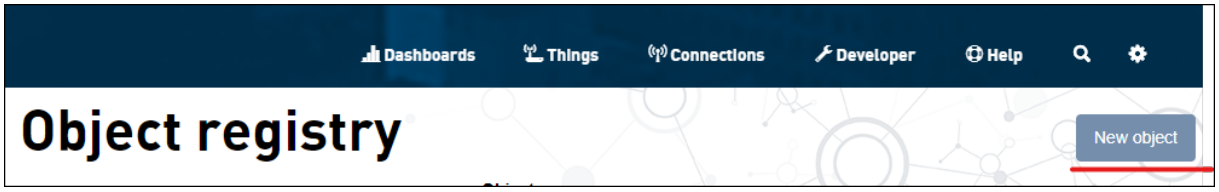

Figure 124

Copy the xml file content and paste it in the new Object form

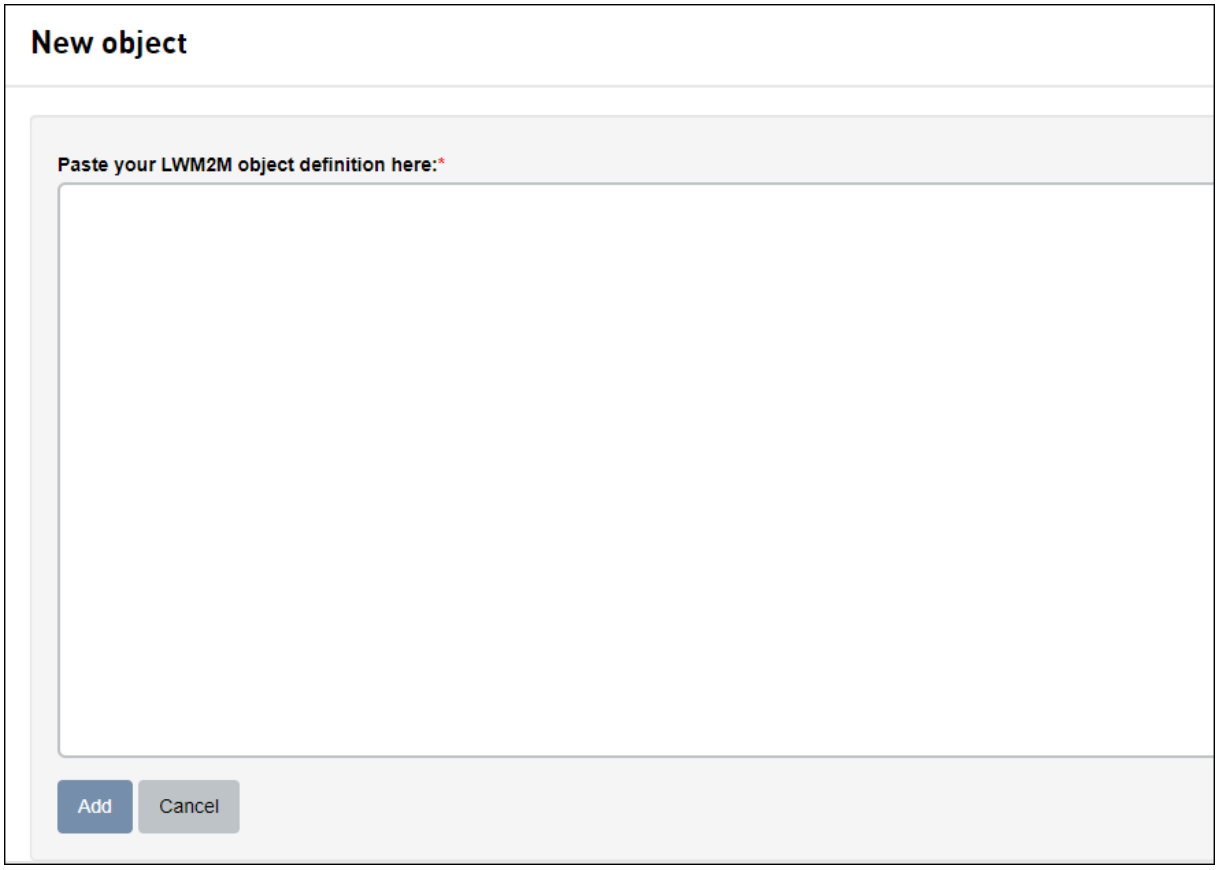

Figure 125

Also, the application requires the XML file `xml/object_32010.xml` (provided with the sample files) to be stored in module's `/XML/` folder. It can be done with

```
AT#M2MWRITE=/XML/object_32010.xml,<size_in_bytes>
```

To load the XML file in the module, Telit AT Controller (TATC) can be used. Once the command above is issued, press the load content button:

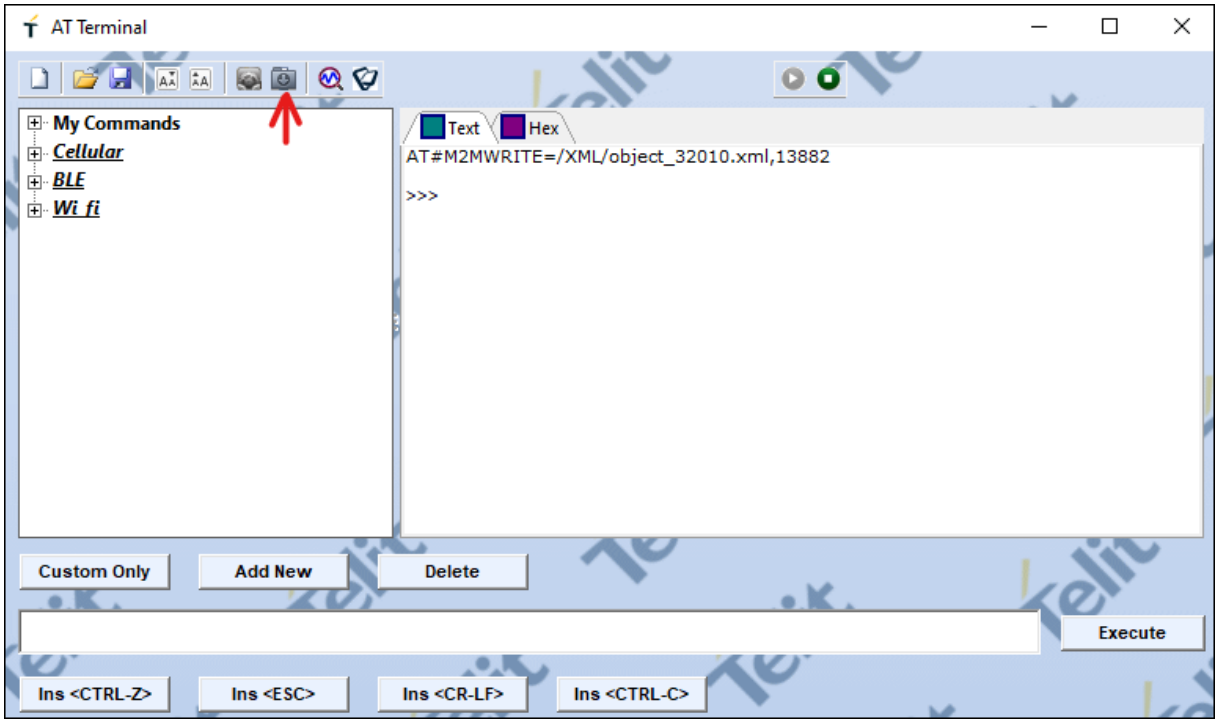

Figure 126

Select the file from your computer

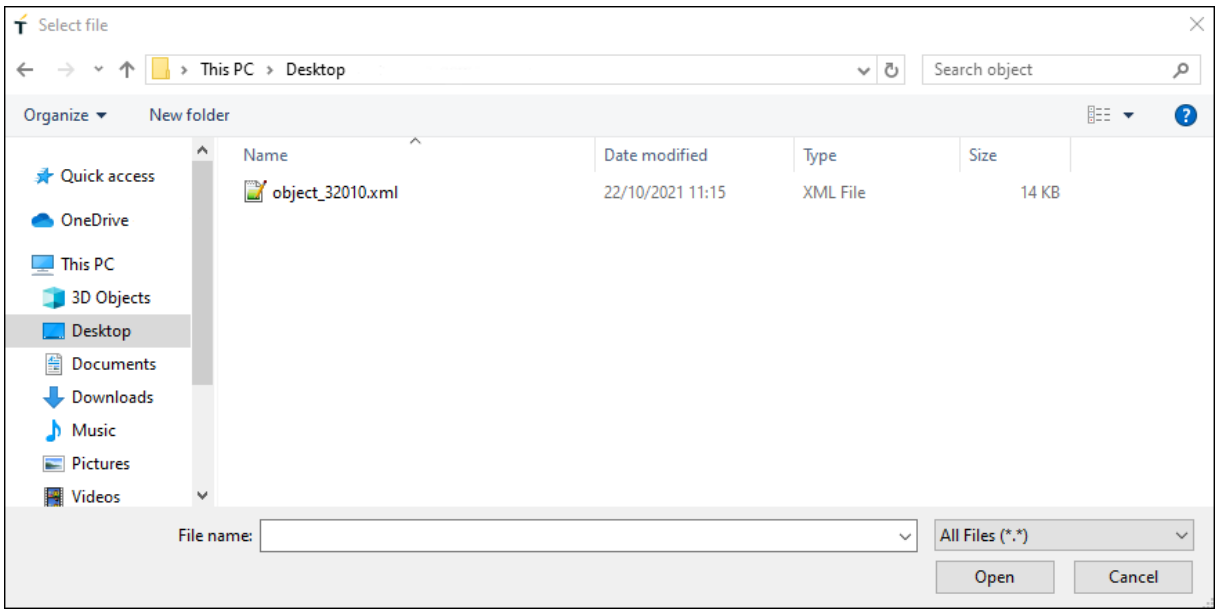

Figure 127

The file is successfully loaded on the module

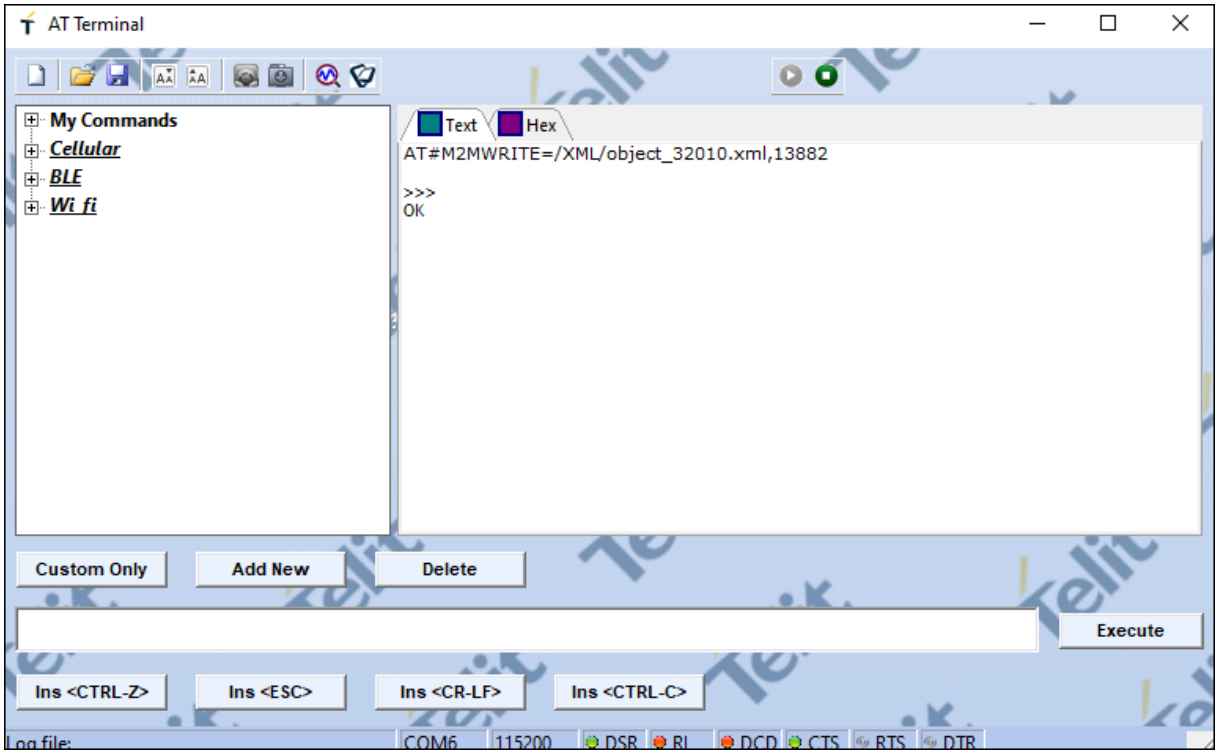

Figure 128

3.5.25.4 Onboard the device

Get the Telit ID

To retrieve the Telit ID data, issue AT#TID to get the Telit ID. The command response will be similar to

#TID: **xxxxxxxxxxxxxxxxxxxxxxxxxxxx**,1 OK

Take note of the Telit ID highlighted in **bold** above (or copy it on a text editor): this ID it will be needed for the onboarding process.

Create a new Thing

From the OneEdge portal, on **"Things"** section, click **"New Thing"** button in the top right corner.

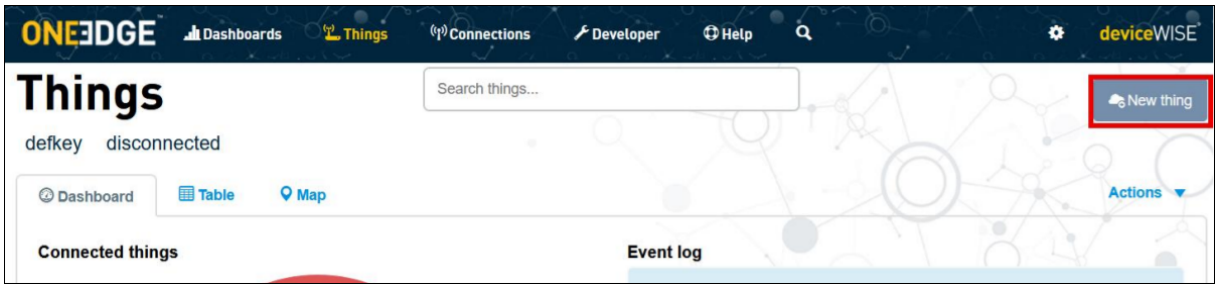

Figure 129

In the Create a new thing dialog, select “Telit Module”

Figure 130

A dialog appears: select “Default” thing definition

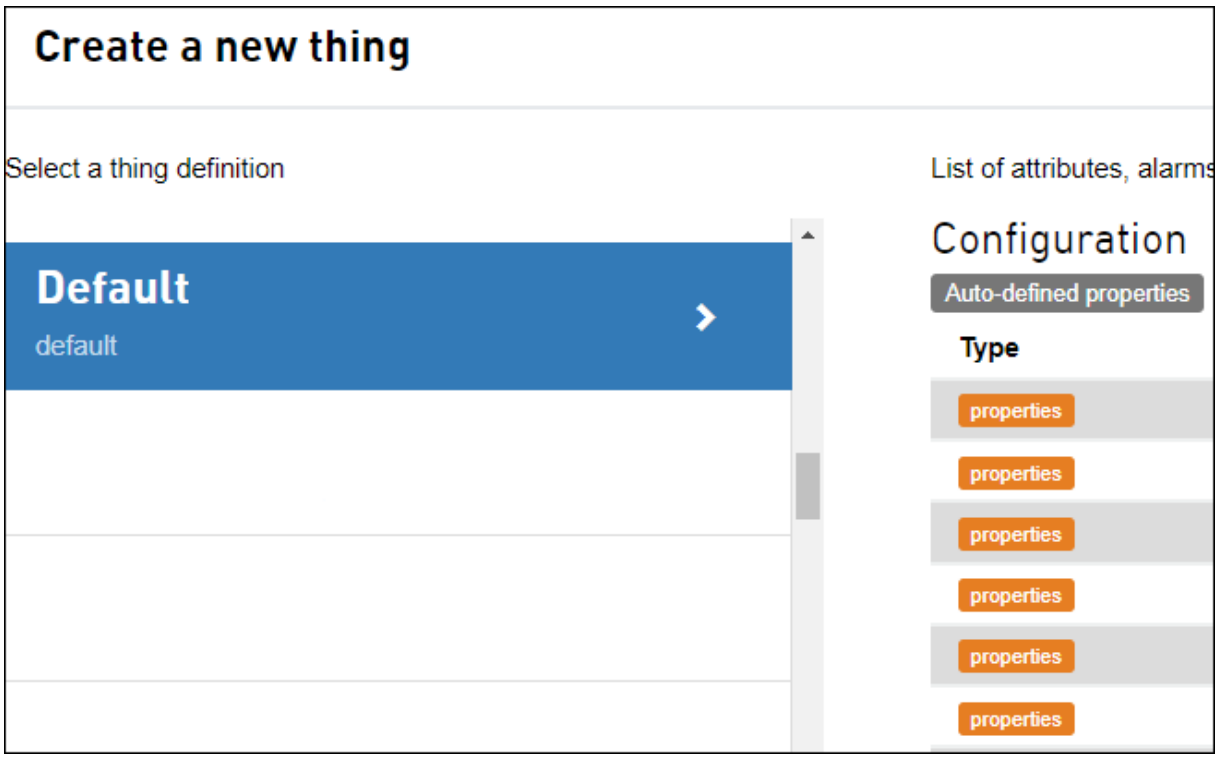

Figure 131

In the following screen, provide the Telit ID as “Identifier” Click on “Find” and make sure that model, firmware and the other details are properly populated.

Click on lwm2m tab and set the device profile previously imported as shown in the

screenshot below

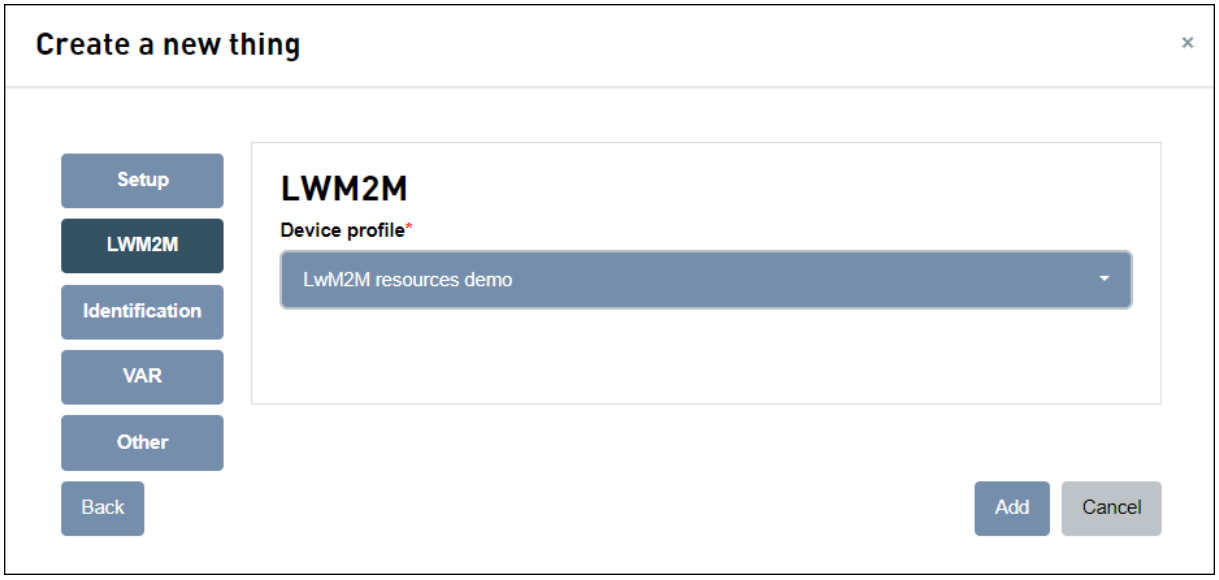

Figure 132

Click **"Add"** to complete the new thing creation procedure.

If the Thing already exists, its device profile can be changed by following the steps shown in the picture below

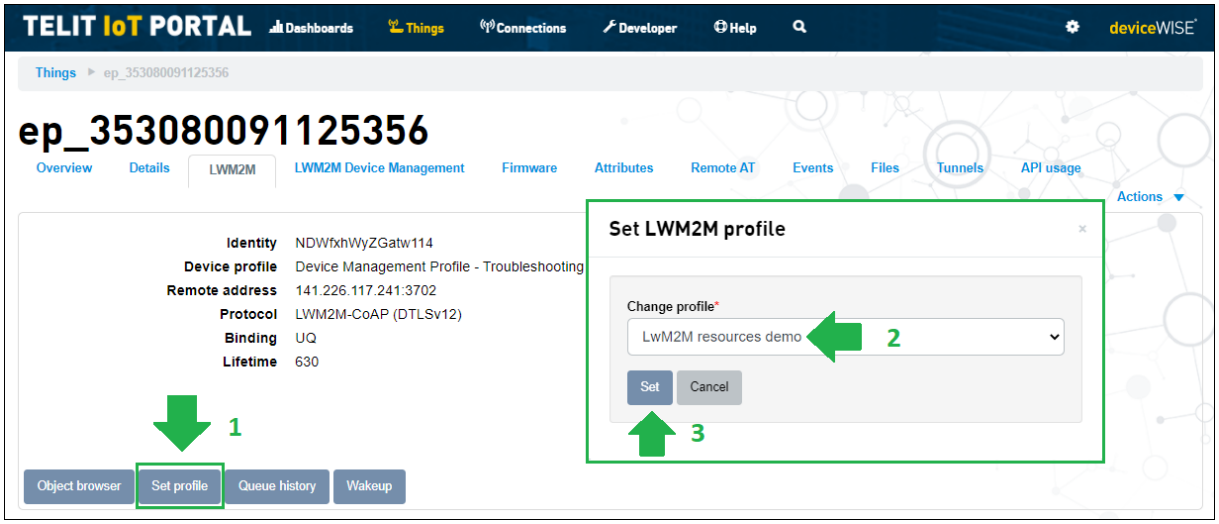

Figure 133

3.5.25.5 Application execution example

```
Starting lwm2m demo. This is v1.1.10 built on Oct 22 2021 10:49:52.
On OneEdge portal, be sure that the demo Device Profile has been uploaded on the IoT Portal
Initializing resources...

LWM2M enable result OK
[DEBUG] 38.20  lwm2m_demo:998 - lwm2mIndicationCB{pubTspt_0}$ Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED
GET STATUS.
IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED
```

Figure 134

```
=================================
READ-ONLY RESOURCES

=================================

Setting integer resource {32010/0/2} value to 50 on LWM2M client.

Reading integer resource {32010/0/2} value on LWM2M client.

Resource /32010/0/2/0 changed!

Reading integer resource {32010/0/2/0} value on LWM2M client.

---Integer value is now 50

Integer data in {32010/0/2/0} resource was updated to new value: 50

-------------------------------------------------

Setting integer resource {32010/0/22/0} value to 10 on LWM2M client.

Resource /32010/0/22/0 changed!

Reading integer resource {32010/0/22/0} value on LWM2M client.

Integer data in {32010/0/22/0} resource was updated to new value: 10

Setting integer resource {32010/0/22/1} value to 11 on LWM2M client.

Resource /32010/0/22/1 changed!

Reading integer resource {32010/0/22/1} value on LWM2M client.

Integer data in {32010/0/22/1} resource was updated to new value: 11

-------------------------------------------------

Setting double resource {32010/0/3} value to 20.500000 on LWM2M client.

Reading double resource {32010/0/3} value on LWM2M client.

Resource /32010/0/3/0 changed!

Reading double resource {32010/0/3/0} value  on LWM2M client.

---Double value is now 20.500000

Float data in {32010/0/3/0} resource was updated to new value: 20.500000

-------------------------------------------------

Setting boolean resource {32010/0/4} value to 1 on LWM2M client.

Reading boolean resource {32010/0/4} value on LWM2M client.

Resource /32010/0/4/0 changed!

Reading boolean resource {32010/0/4/0} value on LWM2M client.

---Boolean value is now true

Boolean data in {32010/0/4/0} resource was updated to new value: true

-------------------------------------------------
```

```
Current time in seconds from the epoch: 1634900084

Setting timestamp resource {32010/0/6} value to 1634900084 on LWM2M client.

Reading timestamp resource {32010/0/6} value on LWM2M client.

Resource /32010/0/6/0 changed!

Reading timestamp resource {32010/0/6/0} value on LWM2M client.

---Timestamp value is now 1634900084

Time data in {32010/0/6/0} resource was updated to new value: 1634900084 (2021/10/22T10:54:44+00:00)

--------------------------------------------------

Setting opaque resource {32010/0/5} on LWM2M client.

Resource /32010/0/5/0 changed!

Reading opaque resource {32010/0/5/0} on LWM2M client.

Opaque data in {32010/0/5/0} resource was updated to new content (10 bytes):
00 01 02 03 04 05 06 07 08 09

Reading opaque resource {32010/0/5} on LWM2M client.

---Opaque content is now (10 bytes): 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09

--------------------------------------------------

Setting string resource {32010/0/1} value to Hello World! on LWM2M client.

Reading string resource {32010/0/1} value on LWM2M client.

Resource /32010/0/1/0 changed!

Reading integer resource {32010/0/1/0} value on LWM2M client.

---String content is now: <Hello World!>

String data in {32010/0/1/0} resource was updated to new content: <Hello World!>

--------------------------------------------------
```

```
=================================
READ-WRITE RESOURCES
=================================

Writing integer resource {32010/0/12} value to 50 on LWM2M client.

Reading integer resource {32010/0/12} value on LWM2M client.

Resource /32010/0/12/0 changed!

Reading integer resource {32010/0/12/0} value on LWM2M client.

---Integer value is now 50

Integer data in {32010/0/12/0} resource was updated to new value: 50

--------------------------------------------------

Writing double resource {32010/0/13} value to 20.500000 on LWM2M client.

Reading double resource {32010/0/13} value on LWM2M client.

Resource /32010/0/13/0 changed!

Reading double resource {32010/0/13/0} value  on LWM2M client.

---Double value is now 20.500000

Float data in {32010/0/13/0} resource was updated to new value: 20.500000

--------------------------------------------------

Writing boolean resource {32010/0/14} value to 1 on LWM2M client.

Reading boolean resource {32010/0/14} value on LWM2M client.

Resource /32010/0/14/0 changed!

Reading boolean resource {32010/0/14/0} value on LWM2M client.

---Boolean value is now true

Boolean data in {32010/0/14/0} resource was updated to new value: true

--------------------------------------------------
```

```
Current time in seconds from the epoch: 1634900125

Writing timestamp resource {32010/0/16} value to 1634900125 on LWM2M client.

Reading timestamp resource {32010/0/16} value on LWM2M client.

Resource /32010/0/16/0 changed!

Reading timestamp resource {32010/0/16/0} value on LWM2M client.

---Timestamp value is now 1634900125

Time data in {32010/0/16/0} resource was updated to new value: 1634900125 (2021/10/22T10:55:25+00:00)

--------------------------------------------------

Writing opaque resource {32010/0/15} on LWM2M client.

Resource /32010/0/15/0 changed!

Reading opaque resource {32010/0/15/0} on LWM2M client.

Opaque data in {32010/0/15/0} resource was updated to new content (10 bytes):
00 01 02 03 04 05 06 07 08 09

Reading opaque resource {32010/0/15} on LWM2M client.

---Opaque content is now (10 bytes): 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09

--------------------------------------------------

Writing string resource {32010/0/11} value to <Hello World!> on LWM2M client.

Reading string resource {32010/0/11} value on LWM2M client.

Resource /32010/0/11/0 changed!

Reading integer resource {32010/0/11/0} value on LWM2M client.

---String content is now: <Hello World!>
```

```
==================================
WRITE-ONLY RESOURCES
==================================

Writing integer resource {32010/0/42} value to 50 on LWM2M client.

String data in {32010/0/11/0} resource was updated to new content: <Hello World!>

Getting integer resource {32010/0/42} valueon LWM2M client.

Resource /32010/0/42/0 changed!

Getting integer resource {32010/0/42/0} value on LWM2M client.

---Integer value is now 50

Integer data in {32010/0/42/0} resource was updated to new value: 50

-------------------------------------------------

Writing double resource {32010/0/43} value to 20.500000 on LWM2M client.

Getting double resource {32010/0/43} value on LWM2M client.

Resource /32010/0/43/0 changed!

Getting double resource {32010/0/43/0} value  on LWM2M client.

---Double value is now 20.500000

Float data in {32010/0/43/0} resource was updated to new value: 20.500000

-------------------------------------------------

Writing boolean resource {32010/0/44} value to 1 on LWM2M client.

Getting boolean resource {32010/0/44} value on LWM2M client.

Resource /32010/0/44/0 changed!

Getting boolean resource {32010/0/44/0} value on LWM2M client.

---Boolean value is now true

Boolean data in {32010/0/44/0} resource was updated to new value: true

-------------------------------------------------
```

```
Current time in seconds from the epoch: 1634900163

Writing timestamp resource {32010/0/46} value to 1634900163 on LWM2M client.

Getting timestamp resource {32010/0/46} value on LWM2M client.

Resource /32010/0/46/0 changed!

Getting timestamp resource {32010/0/46/0} value on LWM2M client.

---Timestamp value is now 1634900163

Time data in {32010/0/46/0} resource was updated to new value: 1634900163 (2021/10/22T10:56:03+00:00)

--------------------------------------------------

Writing opaque resource {32010/0/45} on LWM2M client.

Resource /32010/0/45/0 changed!

Getting opaque resource {32010/0/45/0} on LWM2M client.

Opaque data in {32010/0/45/0} resource was updated to new content (10 bytes):
00 01 02 03 04 05 06 07 08 09

Getting opaque resource {32010/0/45} on LWM2M client.

---Opaque content is now (10 bytes): 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09

--------------------------------------------------

Writing string resource {32010/0/41} value to <Hello World!> on LWM2M client.

Getting string resource {32010/0/41} value on LWM2M client.

Resource /32010/0/41/0 changed!

Getting integer resource {32010/0/41/0} value on LWM2M client.

---String content is now: <Hello World!>

String data in {32010/0/41/0} resource was updated to new content: <Hello World!>

-----------------------------------
Resources operations examples done.
-----------------------------------
```

```
-----------------------------------
Will perform a SET with notify ACK enabled.
-----------------------------------
Enable notify ack

Setting integer resource {32010/0/2} value to 60 on LWM2M client.

Resource /32010/0/2/0 changed!

Reading integer resource {32010/0/2/0} value on LWM2M client.

Integer data in {32010/0/2/0} resource was updated to new value: 60

ACK received from server!

Reading integer resource {32010/0/2} value on LWM2M client.

---Integer value is now 60

Disable notify ack
Done.

Waiting for events from the OneEdge portal. Please write on monitored resources or call an 'exec' one.
```

After the Demo completes the initialization, it is possible to access the object resources from the section Things: select your device, then “LwM2M” tab of OneEdge IoT portal:

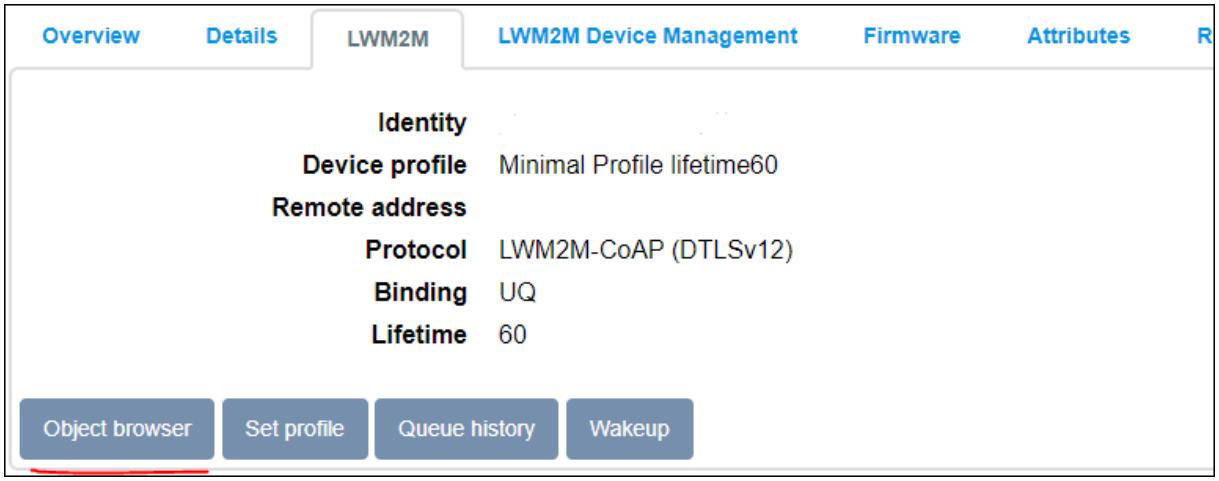

Figure 135

An instance of the object will be present and the resources can be modified.

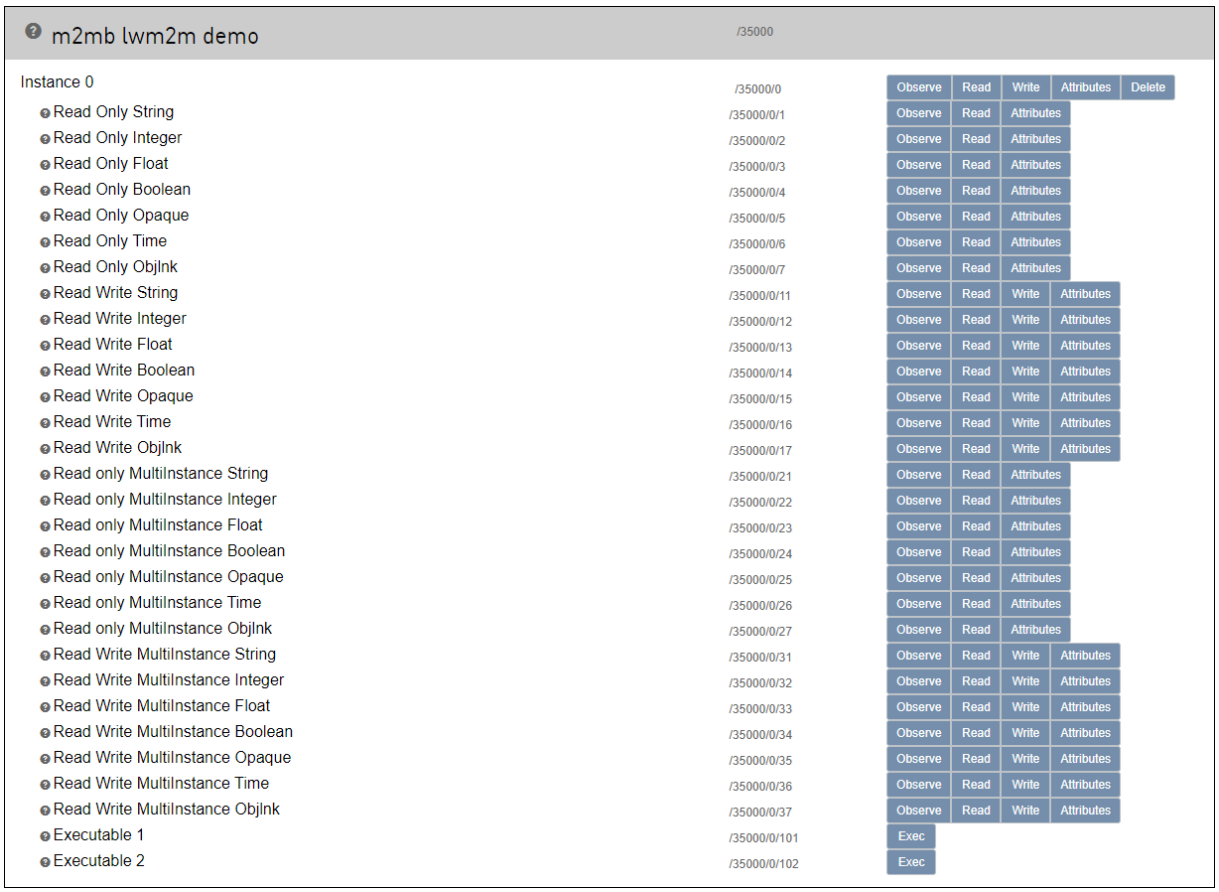

Figure 136

For example, executing the two Exec Resources at the bottom of the list, the application will react accordingly:

```
Info Exec Ind: 32010/0/101/0
Asked to execute resource 101

Resource /32010/0/11/0 changed!
```

Figure 137

Writing a string resource (id /32010/0/11), the application will notify the change

```
Reading integer resource {32010/0/11/0} value on LWM2M client.

String data in {32010/0/11/0} resource was updated to new content: <Hello from the IoT Portal!>
```

Figure 138

3.5.26 LWM2M FOTA ACK management (AT URCs)

Sample application showcasing LWM2M client FOTA events and ACKs management via AT URCs. Debug prints on **USB0**

Features

- Configure LWM2M client and enable it (if not already running)
- Enable monitoring on object 5 (Firmware Update)
- Configure lwm2m FOTA parameters to enable ACK requirement for each operation (Download, Update)
- Wait for an externally generated FOTA request to handle the steps by sending the acknoledgements when needed

Please refer to "80654NT11889A OneEdge Firmware Management Application Note" for further details about FOTA workflow. Get in touch with TS-OneEdge support to receive the document.

Requirements

This application expects the user to configure the PDP context ID 1 with the proper APN. it can be done with the following AT command:

```
AT+CGDCONT=1,"IPV4V6","<user apn>"
```

Depending on the Mobile Network Operator and Access Technology, the APN might be automatically set by the network itself. In this case, nothing must be done by the user.

3.5.26.1 Device Profile upload

Minimal FOTA profile (short lifetime) device profile must be imported and selected to improve the responsiveness of the FOTA operations

To do so, import the file `json/lwm2m_fota_profile_short.json` (provided with the sample files) on section `Developer` > `Device profiles` of OneEdge IoT portal:

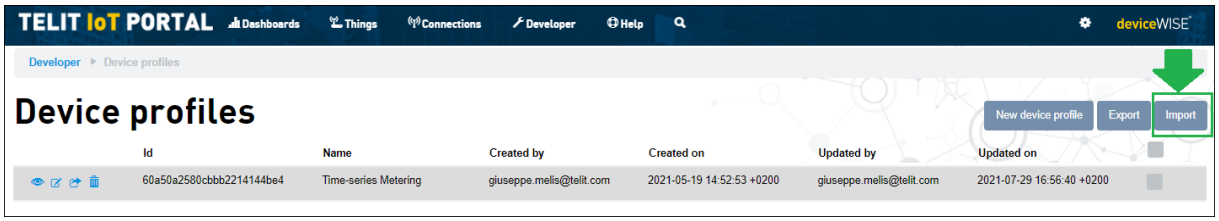

Figure 139

3.5.26.2 Onboard the device

Get the Telit ID

To retrieve the Telit ID data, issue AT#TID to get the Telit ID. The command response will be similar to

#TID: **xxxxxxxxxxxxxxxxxxxxxxxxxxxx**,1 OK

Take note of the Telit ID highlighted in **bold** above (or copy it on a text editor): this ID it will be needed for the onboarding process.

Create a new Thing

From the OneEdge portal, on **"Things"** section, click **"New Thing"** button in the top right corner.

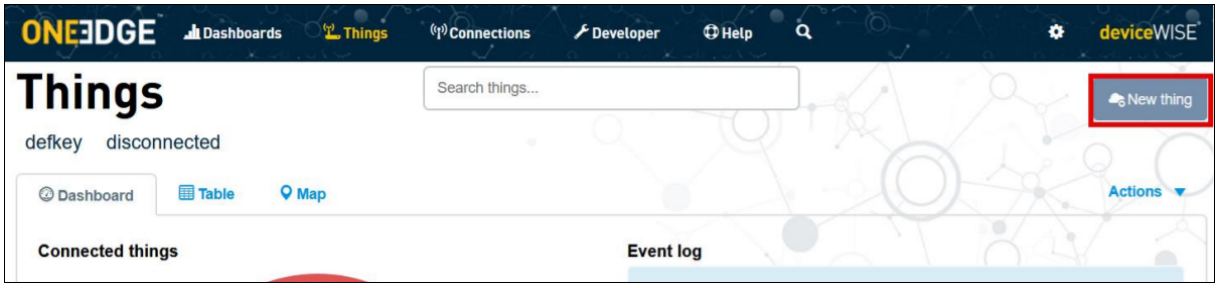

Figure 140

In the Create a new thing dialog, select "Telit Module"

Figure 141

A dialog appears: select "Default" thing definition

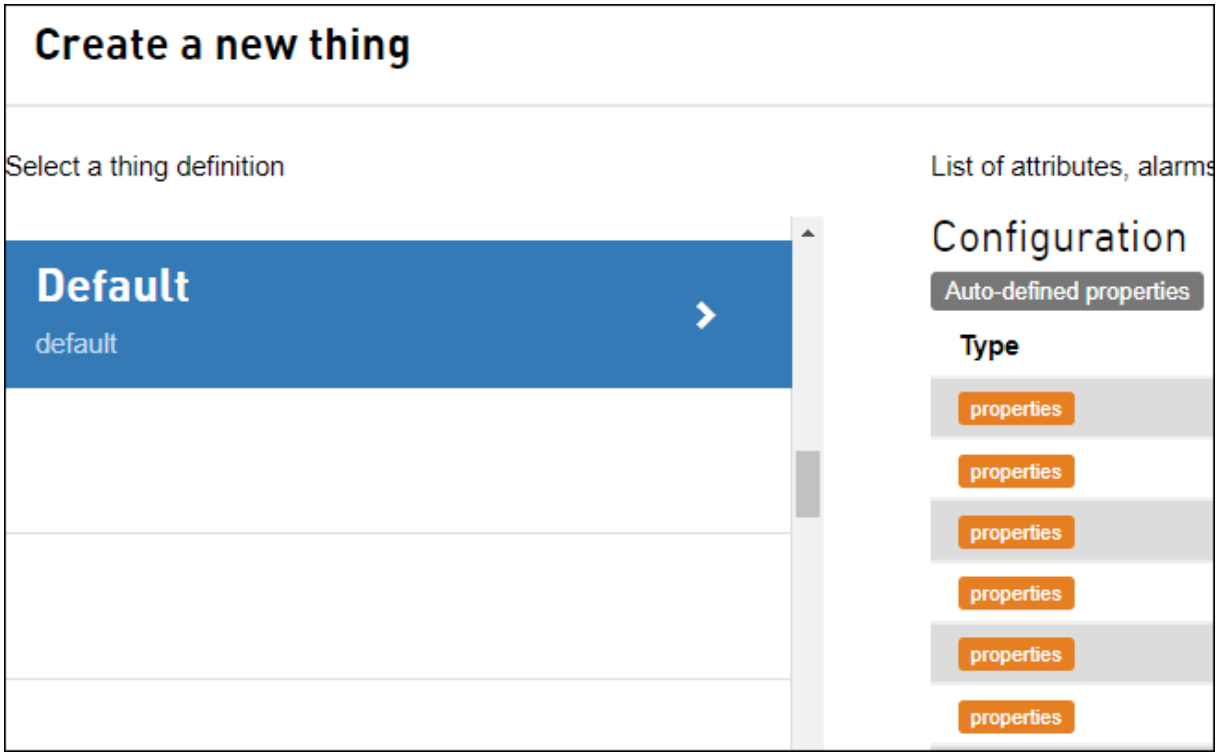

Figure 142

In the following screen, provide the Telit ID as “Identifier” Click on “Find” and make sure that model, firmware and the other details are properly populated.

Click on lwm2m tab and set the device profile previously imported as shown in the screenshot below

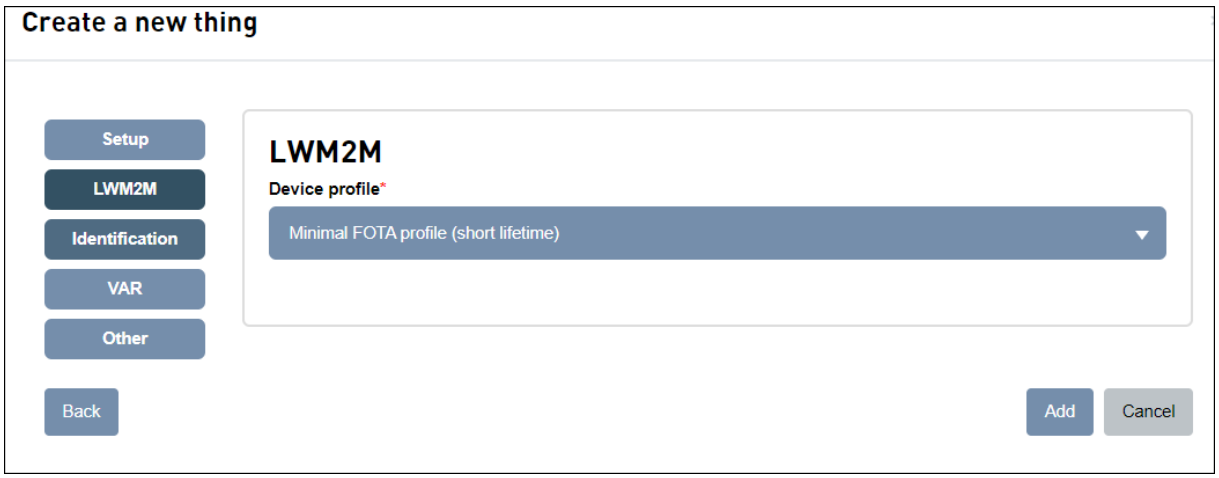

Figure 143

Click **“Add”** to complete the new thing creation procedure.

3.5.26.3 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Create a task to manage the LWM2M client and start it

lwm2m_demo.c

msgLWM2MTask

- Initialize LWM2M client,
- Enable unsolicited messages from client
- Enable FOTA ACK configuration
- Create a task (lwm2m_taskCB is its callback function)to manage events from Portal
- Check if LwM2M client is running, if not, enable it
- Wait for client to register to Portal
- Wait for FOTA events to arrive, and when the Update request arrives, notify the user that everything must stop before sending the last ACK (as the module will start updating and then reboot)

lwm2mIndicationCB

- Manage events arriving from client (operations completion status and unsolicited events)
- Run lwm2m_taskCB when a monitored resource changes, to manage the action to be done

3.5.26.4 Application execution example

```
Starting lwm2m FOTA management demo. This is v1.1.14 built on Jan 21 2022 14:39:47.
Initializing resources...
[DEBUG] 22.14  lwm2m_demo:283 - at_cmd_async_init{LWM2M_TASK}$ m2mb_ati_init() on instance 1

IF Status: M2MB_LWM2M_IF_STATE_DISABLED
Client Status: M2MB_LWM2M_CL_STATE_DISABLED
Enable lwm2m client
LWM2M enable result OK
Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED

IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED

Waiting for events from the OneEdge portal. Please start a FOTA process.
```

Figure 144

```
Resource /5/0/1/0 changed!
[DEBUG] 108.42  lwm2m_demo:544 - lwm2m_taskCB{LWM2M_TASK}$ Firmware update event!
FOTA process is waiting for Download ACK, sending it!
Waiting for Update ACK!
Before sending the ACK to apply FOTA, stop any ongoing operation (filesystem, sockets)

Asked to perform UPDATE step, stopping all operations now.
Application complete.

------- Info, DEVICE REBOOT FOR FOTA ------
????????
#OTAEV: Module Upgraded To New Fw
```

FOTA request arrives from server, ACK is sent for Download and Update. Device restarts with the new firmware version

```
Starting lwm2m FOTA management demo. This is v1.1.14 built on Jan 21 2022 14:39:47.
Initializing resources...
[DEBUG] 26.61  lwm2m_demo:283 - at_cmd_async_init{LWM2M_TASK}$ m2mb_ati_init() on instance 1
[WARN ] 34.98  lwm2m_demo:738 - lwm2mIndicationCB{pubTspt_0}$ write res 1

IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
Enable already done, skip. State: M2MB_LWM2M_IF_STATE_ACTIVE
Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED

IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED

Waiting for events from the OneEdge portal. Please start a FOTA process.
```

Application restarts and is ready for a new FOTA operation

3.5.27 LWM2M OBJ_GET AND OBJ_SET

Sample application showcasing LWM2M client m2mb_lwm2m_objget and m2mb_lwm2m_objset M2MB APIs usage. Debug prints on **USB0**

Features

- Configure LWM2M client and enable it
- Create an instance of a custom object
- Create a Json string
- Set string, integer, float, boolean, timestamp and opaque values with `m2mb_lwm2m_objset`
- Get all resources values with `m2mb_lwm2m_objget`
- Manage write, read and monitoring resources changed from the portal

Requirements

This application expects the user to configure the PDP context ID 1 with the proper APN. it can be done with the following AT command:

```
AT+CGDCONT=1,"IPV4V6","<user apn>"
```

Depending on the Mobiler Network Operator and Access Technology, the APN might be automatically set by the network itself. In this case, nothing must be done by the user.

3.5.27.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the LWM2M client and start it

`lwm2m_demo.c`

`msgLWM2MTask` - Check registration status

- Initialize LWM2M client,
- Check for XML file fo custom object
- Enable unsolicited messages from client
- Create a task (lwm2m_taskCB is its callback function)to manage events from Portal
- Enable LwM2M client

- Create a new instance for the custom object
- Wait for client to register to Portal
- Performs obj_set and obj_get operations on the related resources
- Wait for events from server

`lwm2mIndicationCB`

- Manage events arriving from client (operations completion status and unsolicited events)
- Run lwm2m_taskCB when a monitored resource changes, to manage the action to be done

3.5.27.2 Device Profile upload

LWM2M resources demo device profile must be imported to have a real-time update of resources values on the LWM2M browser.

To do so, import the file `json/lwm2m_resources_demo.json` (provided with the sample files) on section `Developer` > `Device profiles` of OneEdge IoT portal:

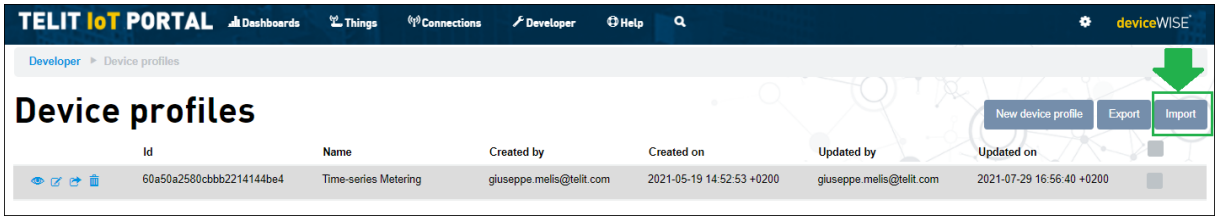

Figure 145

3.5.27.3 Custom Object configuration

The XML file content must be loaded on the Telit IoT Portal for the demo application to be fully executed.

First, enter Developer section from the top menu

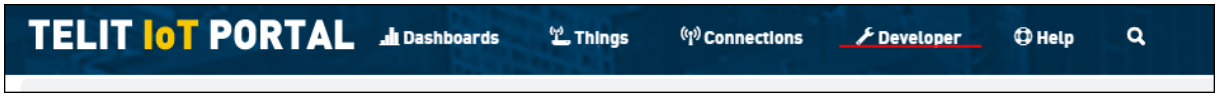

Figure 146

Choose Object Registry

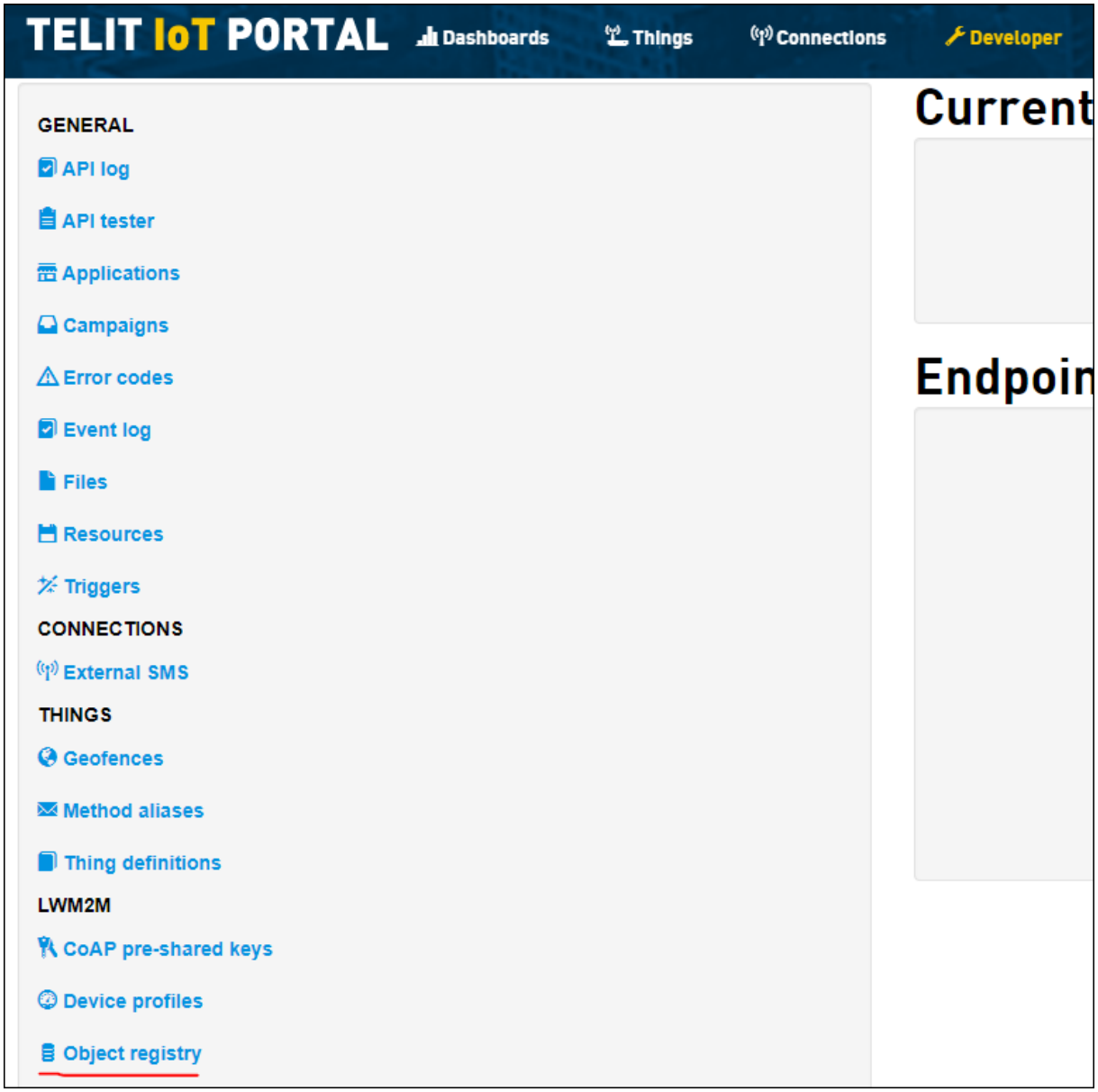

Figure 147

Create a New Object

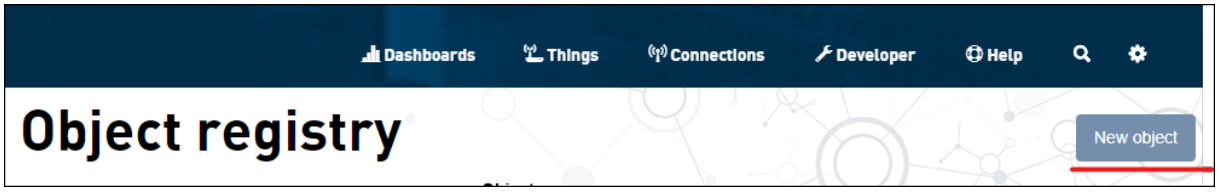

Figure 148

Copy the xml file content and paste it in the new Object form

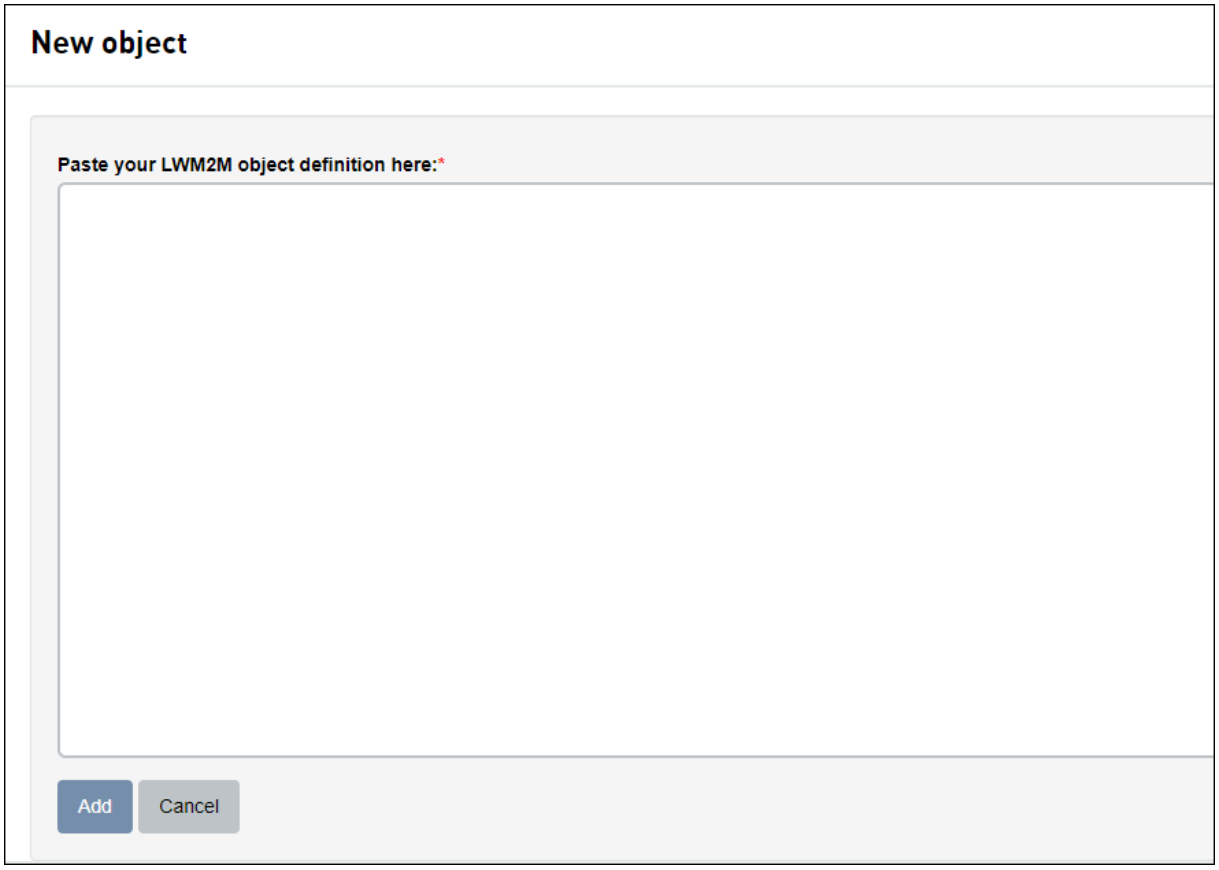

Figure 149

Also, the application requires the XML file `xml/object_32011.xml` (provided with the sample files) to be stored in module's `/XML/` folder. It can be done with

```
AT#M2MWRITE=/XML/object_32011.xml,<size_in_bytes>
```

To load the XML file in the module, Telit AT Controller (TATC) can be used. Once the command above is issued, press the load content button:

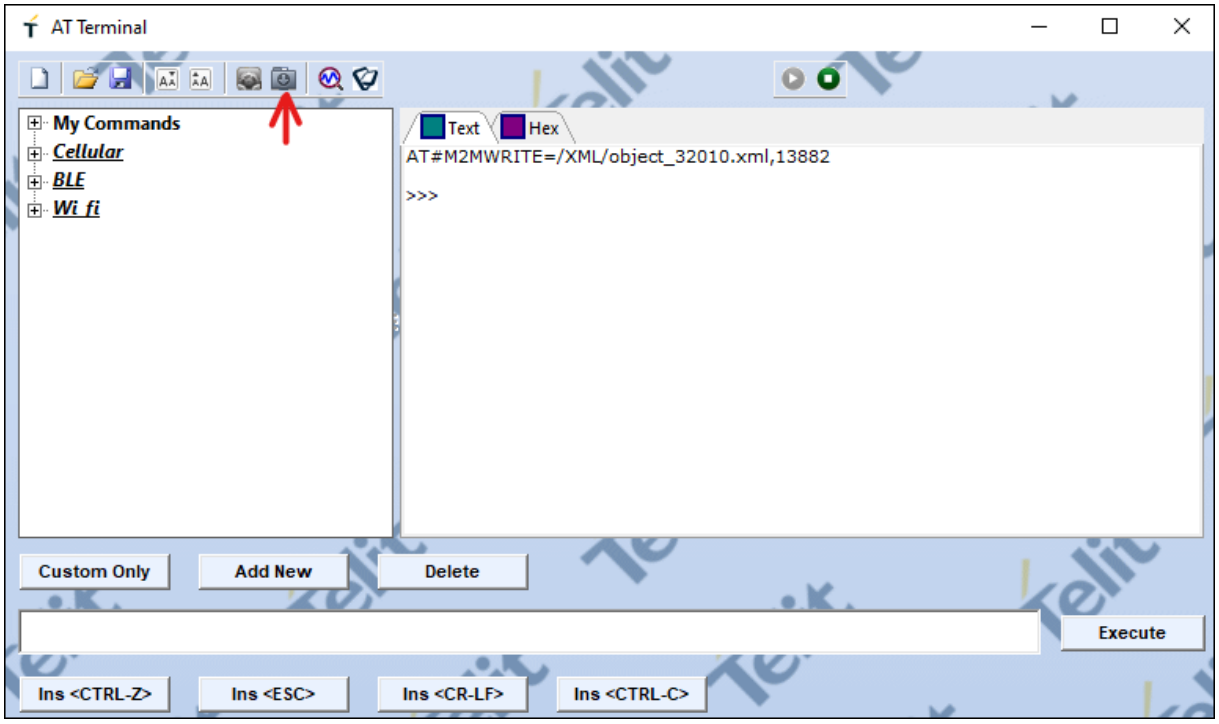

Figure 150

Select the file from your computer

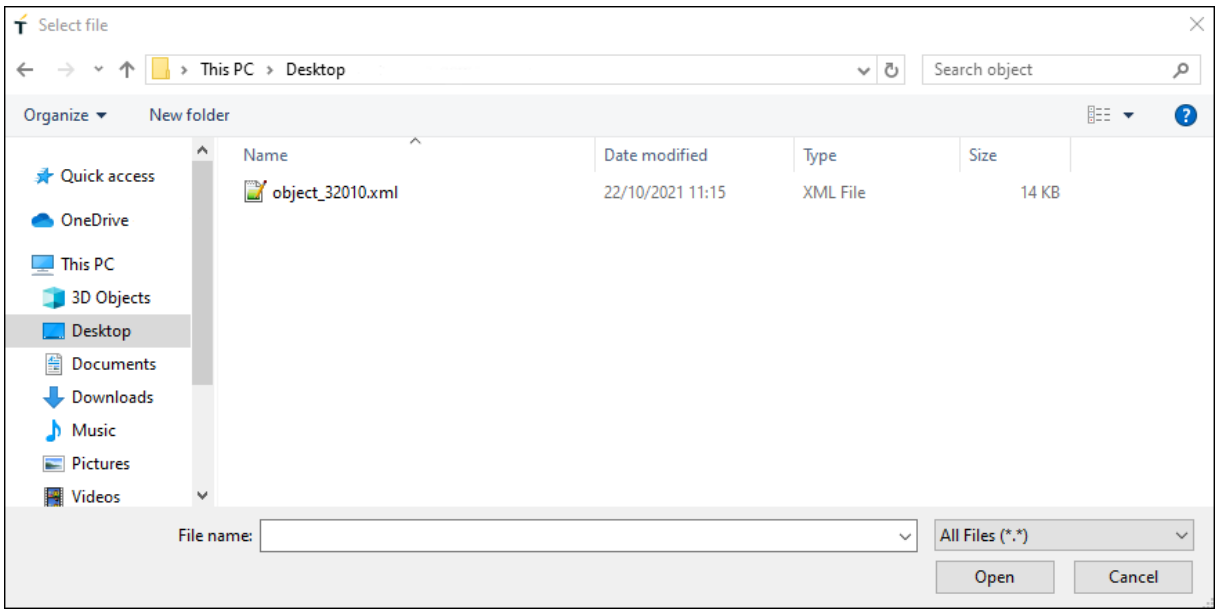

Figure 151

The file is successfully loaded on the module

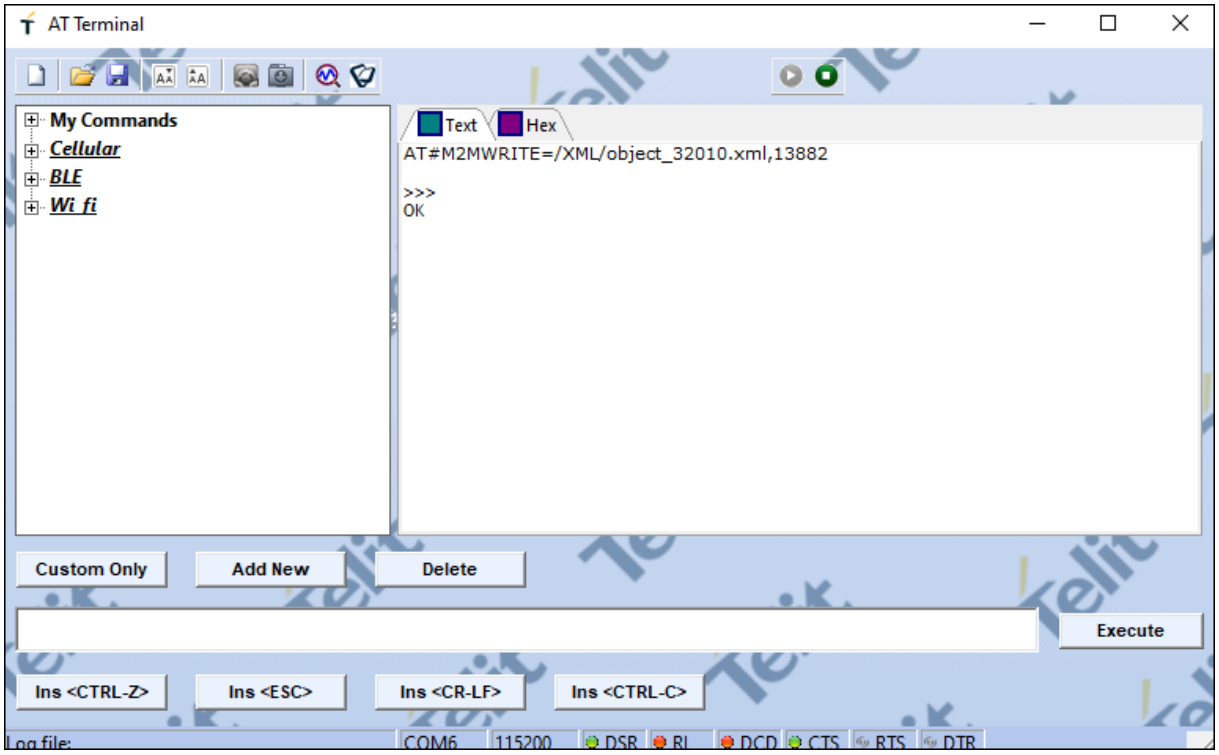

Figure 152

3.5.27.4 Onboard the device

Get the Telit ID

To retrieve the Telit ID data, issue AT#TID to get the Telit ID. The command response will be similar to

#TID: **xxxxxxxxxxxxxxxxxxxxxxxxxxxx**,1 OK

Take note of the Telit ID highlighted in **bold** above (or copy it on a text editor): this ID it will be needed for the onboarding process.

Create a new Thing

From the OneEdge portal, on **"Things"** section, click **"New Thing"** button in the top right corner.

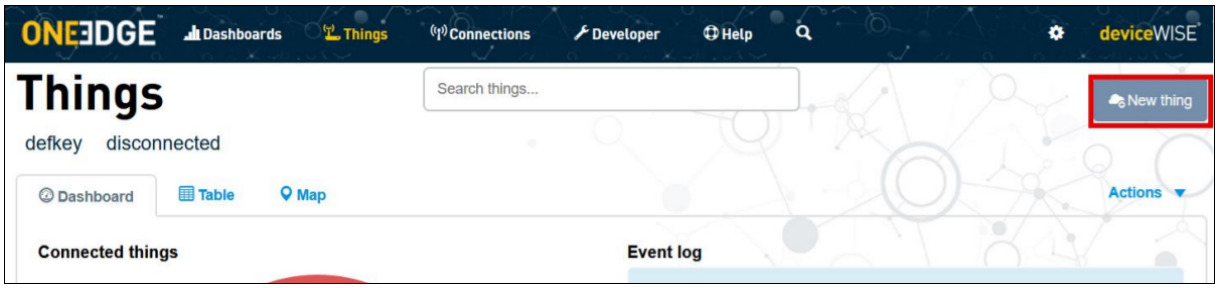

Figure 153

In the Create a new thing dialog, select “Telit Module”

Figure 154

A dialog appears: select “Default” thing definition

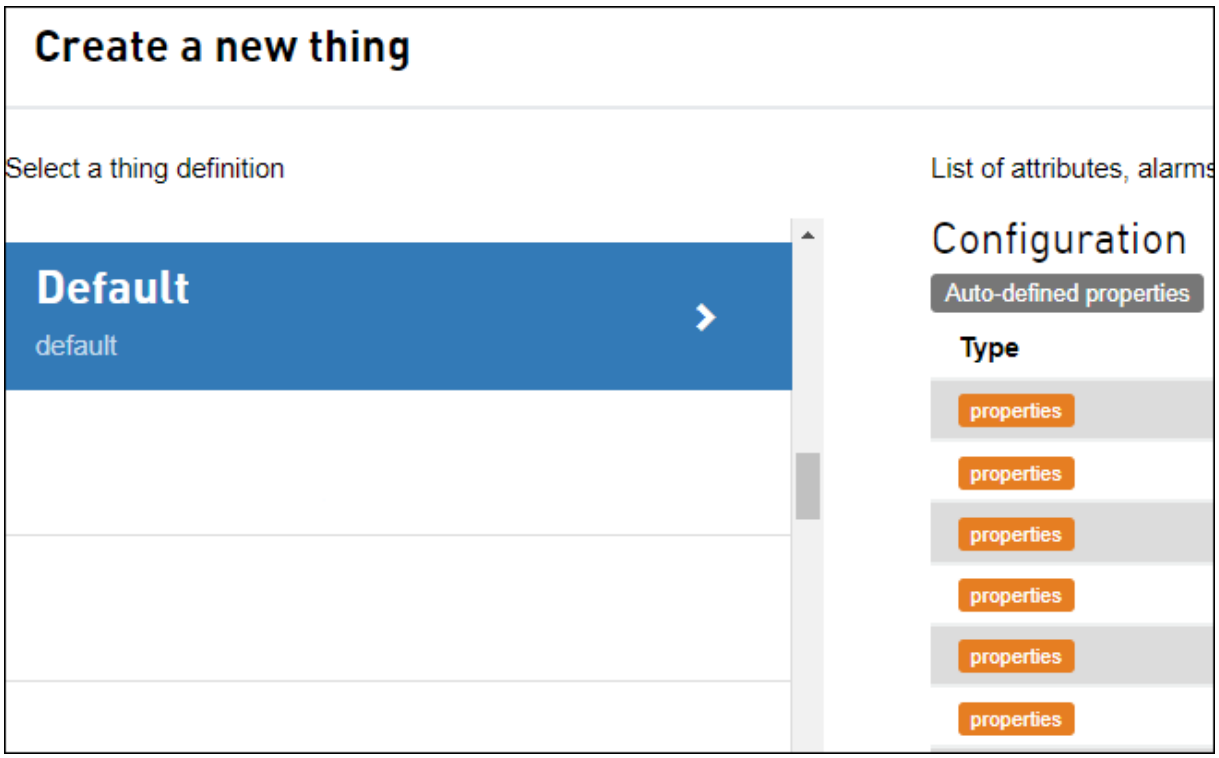

Figure 155

In the following screen, provide the Telit ID as “Identifier” Click on “Find” and make sure that model, firmware and the other details are properly populated.

Click on lwm2m tab and set the device profile previously imported as shown in the

screenshot below

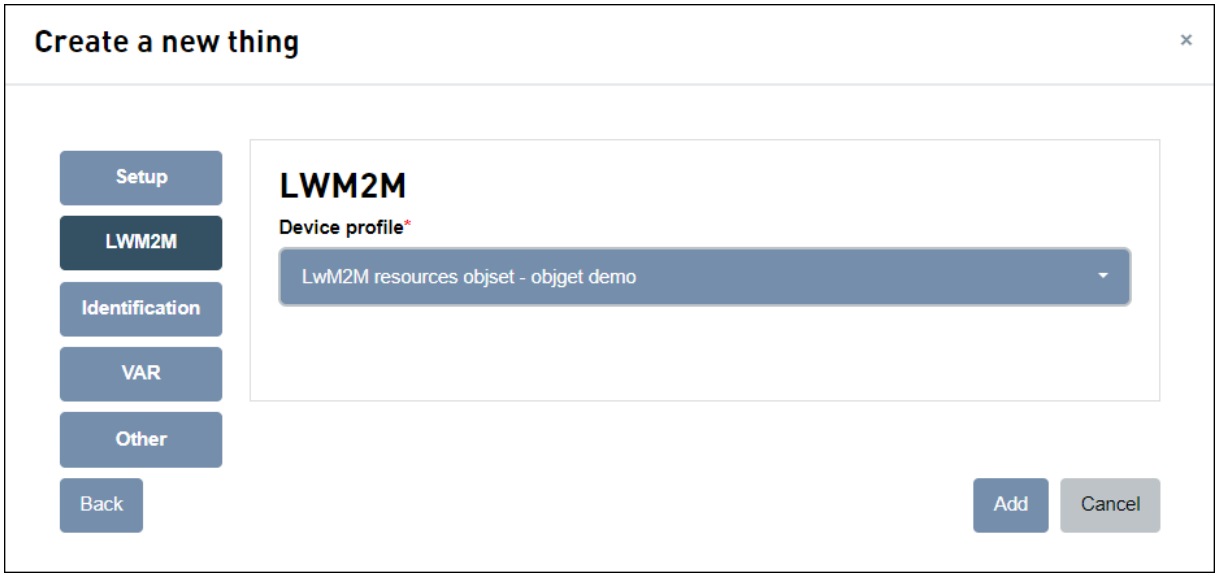

Figure 156

Click **"Add"** to complete the new thing creation procedure.

If the Thing already exists, its device profile can be changed by following the steps shown in the picture below

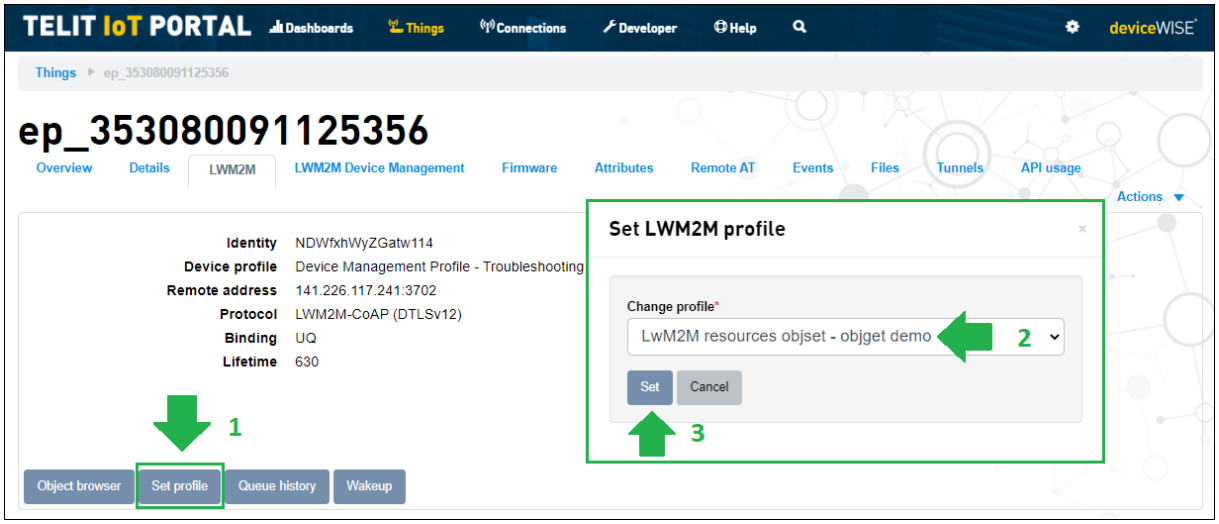

Figure 157

3.5.27.5 Application execution example

```
Starting lwm2m demo. This is v1.1.10 built on Oct 22 2021 10:49:52.
On OneEdge portal, be sure that the demo Device Profile has been uploaded on the IoT Portal
Initializing resources...

LWM2M enable result OK
[DEBUG] 38.20  lwm2m_demo:998 - lwm2mIndicationCB{pubTspt_0}$ Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED
GET STATUS.
IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED
```

Figure 158

```
Starting lwm2m demo. This is v1.1.11 built on Nov 24 2021 13:32:52.
On OneEdge portal, be sure that the 'lwm2m resources objset - objget demo' Device Profile has been uploaded on the IoT Portal

Initializing resources...
LWM2M enable result OK
[DEBUG] 29.42  lwm2m_demo:593 - lwm2mIndicationCB{pubTspt_0}$ Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED
GET STATUS.
IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED

m2mb_lwm2m_objset SUCCESS

Resource /32011/0/11/0 changed!

Reading string resource {32011/0/11/0} value on LWM2M client.

Resource /32011/0/12/0 changed!

Resource /32011/0/13/0 changed!

Resource /32011/0/15/0 changed!

Resource /32011/0/16/0 changed!

Resource /32011/0/31/0 changed!

Resource /32011/0/31/1 changed!

Resource /32011/0/31/2 changed!

Resource /32011/0/34/0 changed!

Resource /32011/0/34/1 changed!

Resource /32011/0/34/2 changed!

Resource /32011/0/34/3 changed!
```

```
String data in {32011/0/11/0} resource was updated to new content: <hello world>

Reading integer resource {32011/0/12/0} value on LWM2M client.

Integer data in {32011/0/12/0} resource was updated to new value: 70

Reading double resource {32011/0/13/0} value  on LWM2M client.

Float data in {32011/0/13/0} resource was updated to new value: -12.500000

Reading opaque resource {32011/0/15/0} on LWM2M client.

Opaque data in {32011/0/15/0} resource was updated to new content (11 bytes):
68 65 6C 6C 6F 20 77 6F 72 6C 64

Reading timestamp resource {32011/0/16/0} value on LWM2M client.

Time data in {32011/0/16/0} resource was updated to new value: 1620828702 (2021/05/12T14:11:42+00:00)

Reading string resource {32011/0/31/0} value on LWM2M client.

String data in {32011/0/31/0} resource was updated to new content: <Greetings>

Reading string resource {32011/0/31/1} value on LWM2M client.

String data in {32011/0/31/1} resource was updated to new content: <from>

Reading string resource {32011/0/31/2} value on LWM2M client.

String data in {32011/0/31/2} resource was updated to new content: <AppZone!>

Reading boolean resource {32011/0/34/0} value on LWM2M client.

Boolean data in {32011/0/34/0} resource was updated to new value: true

Reading boolean resource {32011/0/34/1} value on LWM2M client.

Boolean data in {32011/0/34/1} resource was updated to new value: false

Reading boolean resource {32011/0/34/2} value on LWM2M client.

Boolean data in {32011/0/34/2} resource was updated to new value: true

Reading boolean resource {32011/0/34/3} value on LWM2M client.

Boolean data in {32011/0/34/3} resource was updated to new value: false
```

```
Waiting 5 seconds before the m2mb_lwm2m_objget

Waiting: 5

Waiting: 4

Waiting: 3

Waiting: 2

Waiting: 1

m2mb_lwm2m_objget SUCCESS

Content:
<{"bn":"/32011/0/",
"e":[
{"n":"11","sv":"hello world"},
{"n":"12","v":70},
{"n":"13","v":-12.500000},
{"n":"14","bv":false},
{"n":"15","sv":"aGVsbG8gd29ybGQ="},
{"n":"16","v":1620828702},
{"n":"17","ov":"0:0"},
{"n":"31/0","sv":"Greetings"},
{"n":"31/1","sv":"from"},
{"n":"31/2","sv":"AppZone!"},
{"n":"31/3","sv":""},
{"n":"31/4","sv":""},
{"n":"31/5","sv":""},
{"n":"31/6","sv":""},
{"n":"31/7","sv":""},
{"n":"31/8","sv":""},
{"n":"31/9","sv":""},
{"n":"34/0","bv":true},
{"n":"34/1","bv":false},
{"n":"34/2","bv":true},
{"n":"34/3","bv":false},
{"n":"34/4","bv":false},
{"n":"34/5","bv":false},
{"n":"34/6","bv":false},
{"n":"34/7","bv":false},
{"n":"34/8","bv":false},
{"n":"34/9","bv":false}]
}
>
```

After the Demo completes the initialization, it is possible to access the object re-

sources from the section Things: select your device, then “LwM2M” tab of OneEdge IoT portal:

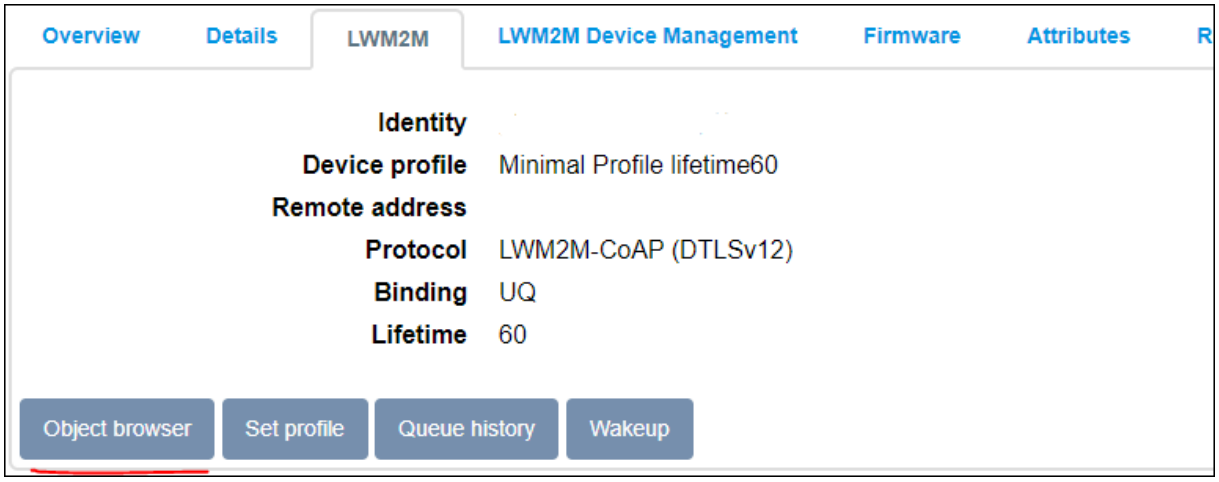

Figure 159

An instance of the object will be present and the resources can be modified.

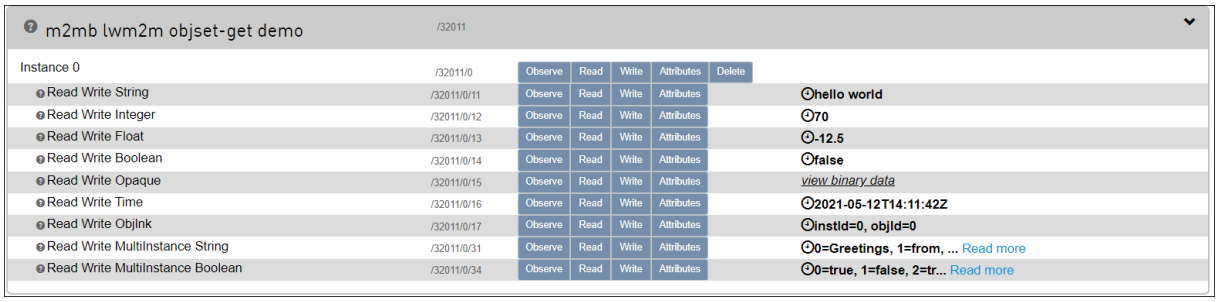

Figure 160

3.5.28 Logging Demo

Sample application showing how to print on one of the available output interfaces. Debug prints on **USB0**

Features

- How to open a logging channel
- How to set a logging level
- How to use different logging macros

3.5.28.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Print welcome message
- Print a message with every log level

```
Starting Logging demo app. This is v1.0.7 built on Mar 26 2020 13:57:06.
[WARN ] 20.17  M2MB_main.c:74 - M2MB_main{M2M_DamsStart}$ This is a WARNING MESSAGE
[ERROR] 20.18  M2MB_main.c:76 - M2MB_main{M2M_DamsStart}$ THIS IS AN ERROR MESSAGE
[CRITICAL] 20.19  M2MB_main.c:78 - M2MB_main{M2M_DamsStart}$ THIS IS AN CRITICAL MESSAGE
[DEBUG] 20.19  M2MB_main.c:80 - M2MB_main{M2M_DamsStart}$ This is a DEBUG message
[TRACE] 20.20  M2MB_main.c:82 - M2MB_main{M2M_DamsStart}$ This is a TRACE message
END.
```

Figure 161

3.5.29 MD5 example

Sample application showing how to compute MD5 hashes using m2mb crypto. Debug prints on **USB0**

Features

- Compute MD5 hash of a file
- Compute MD5 hash of a string

3.5.29.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Create a temporary file with the expected content
- Compute MD5 hash of the provided text file
- Compare the hash with the expected one
- Compute MD5 hash of a string
- Compare the hash with the expected one
- Delete test file

```
Starting MD5 demo app. This is v1.0.7 built on Apr  7 2020 10:19:54.
Buffer written successfully into file. 45 bytes were written.

Computing hash from file...
Computed hash: bb0fa6eff92c305f166803b6938dd33a
Expected hash: bb0fa6eff92c305f166803b6938dd33a
Hashes are the same!

Computing hash from string...
Computed hash: bb0fa6eff92c305f166803b6938dd33a
Expected hash: bb0fa6eff92c305f166803b6938dd33a
Hashes are the same!
```

Figure 162

3.5.30 MQTT Client

Sample application showcasing MQTT client functionalities (with SSL). Debug prints on **USB0**

Features

- How to check module registration and enable PDP context
- How to configure MQTT client parameters
- How to connect to a broker with SSL and exchange data over a subscribed topic

3.5.30.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage MQTT client and start it

`mqtt_demo.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init MQTT client
- Configure it with all parameters (Client ID, username, password, PDP context ID, keepalive timeout...)
- Connect MQTT client to broker
- Subscribe to two topics
- Publish 10 messages with increasing counter. Even messages are sent to topic 1, odd messages on topic 2.
- Print received message in mqtt_topc_cb function
- Disconnect MQTT client and deinit it
- Disable PDP context

```
Starting MQTT demo app. This is v1.0.7 built on Apr  7 2020 10:34:08.
[DEBUG] 16.18  mqtt_demo.c:192 - MQTT_Task{MQTT_TASK}$ INIT
[DEBUG] 16.18  mqtt_demo.c:206 - MQTT_Task{MQTT_TASK}$ m2mb_os_ev_init success
[DEBUG] 16.19  mqtt_demo.c:214 - MQTT_Task{MQTT_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 16.19  mqtt_demo.c:221 - MQTT_Task{MQTT_TASK}$ Waiting for registration...
[DEBUG] 16.20  mqtt_demo.c:131 - NetCallback{pubTspt_0}$ Module is registered
[DEBUG] 16.21  mqtt_demo.c:232 - MQTT_Task{MQTT_TASK}$ Pdp context activation
[DEBUG] 18.26  mqtt_demo.c:246 - MQTT_Task{MQTT_TASK}$ Activate PDP with APN web.omnitel.it on CID 3....
[DEBUG] 18.95  mqtt_demo.c:155 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 18.96  mqtt_demo.c:159 - PdpCallback{pubTspt_0}$ IP address: 37.118.201.56
[DEBUG] 18.96  mqtt_demo.c:268 - MQTT_Task{MQTT_TASK}$ Init MQTT
[DEBUG] 18.97  mqtt_demo.c:278 - MQTT_Task{MQTT_TASK}$ m2mb_mqtt_init succeeded

Connecting to broker <api-dev.devicewise.com>:1883...
Done.
Subscribing to test_topic and test_topic2..
[DEBUG] 20.35  mqtt_demo.c:367 - MQTT_Task{MQTT_TASK}$ Done.

[DEBUG] 20.36  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 2> to topic test_topic
[DEBUG] 20.37  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 20.71  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 20.72  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 2>
[DEBUG] 23.37  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 3> to topic test_topic2
[DEBUG] 23.38  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 23.92  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 23.93  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 3>
[DEBUG] 26.40  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 4> to topic test_topic
[DEBUG] 26.41  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 26.93  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 26.93  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 4>
[DEBUG] 29.42  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 5> to topic test_topic2
[DEBUG] 29.43  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 29.99  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 30.00  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 5>
[DEBUG] 32.46  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 6> to topic test_topic
[DEBUG] 32.48  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 33.00  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 33.01  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 6>
[DEBUG] 35.47  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 7> to topic test_topic2
[DEBUG] 35.48  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 36.01  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 36.02  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 7>
[DEBUG] 38.50  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 8> to topic test_topic
[DEBUG] 38.51  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 39.15  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 39.16  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 8>
[DEBUG] 41.52  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 9> to topic test_topic2
[DEBUG] 41.53  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 42.10  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 42.12  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 9>
[DEBUG] 44.56  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 10> to topic test_topic
[DEBUG] 44.57  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 45.09  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 28
[DEBUG] 45.11  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 10>
[DEBUG] 47.58  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 11> to topic test_topic2
[DEBUG] 47.59  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 48.12  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 28
[DEBUG] 48.13  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 11>

Disconnecting from MQTT broker..
[DEBUG] 50.60  mqtt_demo.c:414 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 50.61  mqtt_demo.c:443 - MQTT_Task{MQTT_TASK}$ application exit
[DEBUG] 50.62  mqtt_demo.c:453 - MQTT_Task{MQTT_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 50.63  mqtt_demo.c:457 - MQTT_Task{MQTT_TASK}$ Application complete.
[DEBUG] 51.23  mqtt_demo.c:164 - PdpCallback{pubTspt_0}$ Context deactivated!
```

Figure 163

3.5.31 MultiTask

Sample application showcasing multi tasking functionalities with M2MB API. Debug prints on **USB0**

Features

- How to create tasks using azx utilities
- How to use send messages to tasks
- How to use a semaphore to synchronize two tasks

3.5.31.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create three tasks with the provided utility (this calls public m2mb APIs)
- Send a message to the task1, its callback function azx_msgTask1 will be called

`azx_msgTask1`

- Print received parameters from main
- Send modified parameters to task2 (its callback function azx_msgTask2 will be called)
- wait for an InterProcess Communication semaphore to be available (released by task3)
- Once the semaphore is available, print a message and return

`azx_msgTask2`

- Print received parameters from caller
- If first parameter is bigger than a certain value, Send modified parameters to task3
- Else, use the second parameter as a task handle and print the corresponding name plus the value of the first parameter

`azx_msgTask3`

- Print received parameters from task 2
- release IPC semaphore
- send message to task 2 with first parameter below the threshold and second parameter with task3 handle

```
Starting MultiTask demo app. This is v1.0.12-C1 built on Jun 23 2020 15:36:31.

Inside "myTask1" user callback function. Received parameters from MAIN: 3 4 5
Task1 - Sending a message to task 2 with modified parameters...
Task1 - Waiting for semaphore to be released by task 3 now...

Inside "myTask2" user callback function. Received parameters: 5 7 10
Task2 - Sending a message to task 3 with modified parameters...
Task2 - Done.

Inside "myTask3" user callback function. Received parameters from Task 2: 15 14 9
Task3 - Releasing IPC semaphore...

Task1 - After semaphore! return...

Task3 - IPC semaphore released.
Task3 - Sending a message to task 2 with specific 'type' parameter value of 0 and task 3 handle as param1...

Inside "myTask2" user callback function. Received parameters: 0 1073951320 9
Task3 - Done.
Task2 - Received type 0 from task "myTask3"
Task2 - Done.
```

Figure 164

3.5.32 MutEx

Sample application showing mutex usage, with ownership and prioritization usage. Debug prints on **USB0**

Features

- How to create a mutex
- How to use the mutex with tasks having different priorities
- how to reorder the pending tasks queue for the mutex

3.5.32.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create four tasks with the provided utility (this calls public m2mb APIs). The first task is a "producer", putting data on a shared buffer. The second is a "consumer" of said data, the other two are used for prioritization demo
- run producer and consumer tasks at the same pace. the shared buffer will stay empty, because the resource is consumed right after creation
- run producer twice as fast as consumer. The buffer is slowly filled
- run consumer twice as fast as publisher. The buffer is always empty.
- reserve the mutex in the main task and run producer, support and support2 tasks (in this order). Then release the mutex and check the execution order. It should be by arrival.
- reserve the mutex in the main task and run the same three task, but before releasing the mutex, call the prioritization API. the task with highest priority (producer) is put as first in the queue.

```
Starting MutEx app. This is v1.0.12-C1 built on Jul  1 2020 08:37:15.
[DEBUG] 14.50  M2MB_main:90 - mutex_init{M2M_DamsStart}$ [MUTEX] Mutex initialized

[CASE 1 ] Producer and consumer have same idle time

[DEBUG] 14.51  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 14.52  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 14.53  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 14.53  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 14.54  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 14.54  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 14.55  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 14.56  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 15.56  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 15.56  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 15.57  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 15.58  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 15.58  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 15.59  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 15.60  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 15.60  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 16.61  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 16.61  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 16.62  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 16.63  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 16.63  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 16.64  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 16.64  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 16.65  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
```

Figure 165

```
[CASE 2 ] Producer has double idle time

[DEBUG] 17.56  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 17.56  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 17.57  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 17.58  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 17.58  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 17.59  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 17.59  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 17.60  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 18.63  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 18.64  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 0 items
[DEBUG] 18.64  M2MB_main:268 - msgConsumer{CONSUMER}$ Can't consume anything, buffer size is 0
[DEBUG] 18.65  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 19.62  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 19.62  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 19.63  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 19.64  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 19.68  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 19.69  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 19.69  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 19.70  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 20.73  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 20.74  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 0 items
[DEBUG] 20.75  M2MB_main:268 - msgConsumer{CONSUMER}$ Can't consume anything, buffer size is 0
[DEBUG] 20.75  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 21.67  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 21.68  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 21.68  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 21.69  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 21.77  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 21.79  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 21.80  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 21.80  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
```

Figure 166

```
[CASE 3 ] Producer has half idle time

[DEBUG] 22.62  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 22.63  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 22.64  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 22.64  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 22.65  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 22.65  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 22.66  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 22.67  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 23.67  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 23.68  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 23.68  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 23.69  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 24.71  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 24.72  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 24.72  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 24.73  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 24.74  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 24.74  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 24.75  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 24.76  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 25.79  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 25.79  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 1 items
[DEBUG] 25.80  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 1
[DEBUG] 25.81  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 26.78  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 26.78  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 2 items
[DEBUG] 26.79  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 1
[DEBUG] 26.79  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 26.84  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 26.84  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 1 items
[DEBUG] 26.85  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 1
[DEBUG] 26.86  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
```

Figure 167

```
[CASE 4 ] NO HTPF

Reserve MUTEX so all tasks are enqueued
[DEBUG] 30.77  M2MB_main:387 - msgSupport{HPTF_SUPPORT}$ freepos = 0 | evaluate[freepos]= 3
[DEBUG] 30.78  M2MB_main:416 - msgSupport2{HPTF_SUPPORT2}$ freepos = 1 | evaluate[freepos]= 4
[DEBUG] 30.79  M2MB_main:223 - msgProducer{PRODUCER}$ producer: freepos = 2 | evaluate[freepos]= 1
[DEBUG] 35.85  M2MB_main:586 - M2MB_main{M2M_DamsStart}$ EVALUATE SEQUENCE IS 3 4 1. Expected: 3 4 1
NO HTPF OK

[CASE 4.1 ] HTPF USED

Reserve MUTEX so all tasks are enqueued
m2mb_os_mtx_hptf OK
[DEBUG] 41.98  M2MB_main:223 - msgProducer{PRODUCER}$ producer: freepos = 0 | evaluate[freepos]= 1
[DEBUG] 41.99  M2MB_main:387 - msgSupport{HPTF_SUPPORT}$ freepos = 1 | evaluate[freepos]= 3
[DEBUG] 42.00  M2MB_main:416 - msgSupport2{HPTF_SUPPORT2}$ freepos = 2 | evaluate[freepos]= 4
[DEBUG] 44.03  M2MB_main:650 - M2MB_main{M2M_DamsStart}$ EVALUATE SEQUENCE IS 1 3 4, expected 1 3 4
HTPF DEMO OK
The application has ended...
```

Figure 168

3.5.33 NTP example

The application connects to an NTP server, gets current date and time and updates module's internal clock. Debug prints on **USB0**

Features

- How to get current date and time from an NTP server
- How to set current date and time on module

3.5.33.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Print welcome message
- Send message to ntpTask

ntp_task.c

NTP_task() - Waits module registration - When module is registered, initializes ntp setting CID, server url and timeout - When PDP context is correctly opened, a query to NTP server is done to get current date and time - On SET_MODULE_RTC message type reception, module RTC is set with date time value got from NTP server.

m2mb_ntp_ind_callback() - As soon as M2MB_NTP_VALID_TIME event is received, current date and time is printend and a message (with SET_MODULE_RTC type) is sent to NTP_task

```
Start NTP demo application. This is v1.0 built on Apr 16 2021 09:36:12.

Waiting for registration...
Module is registered!

Activate PDP context with APN ibox.tim.it on CID 3
Context activated, IP address: 2.195.170.123
Get current time from server 0.pool.ntp.org, PORT: 123

Current time is: Friday 2021-04-16, 07:37:33

Current time correctly set on module
Module system time is: 2021-04-16, 07:37:33
```

Figure 169

3.5.34 RTC example

Sample application that shows RTC apis functionalities: how to get/set moudle system time and timestamp. Debug prints on **USB0**

Features

- How to read module timestamp
- How to read module system time
- How to set new system time

3.5.34.1 Application workflow

M2MB_main.c

- Init log azx and print a welcome message
- Init net functionality and wait for module registration
- Init RTC functionality and get module time in timestamp format (seconds from the epoch)
- Get moudle system time in date/time format
- Add 1 hour to timestamp, convert it to system time and set it to module

```
 Start RTC demo application. This is v1.0 built on Oct  1 2021 15:01:40.

Waiting for registration...
Module is registered!

Current time in seconds from the epoch: 1633101266
Module system time is: 2021-10-01, 15:14:26

Get current time and add an hour

Current time in seconds from the epoch: 1633101266
New time to be set : 2021-10-01, 16:14:26, tz:4, dlst:0

Set new time and check the setting
NEW module system time is: 2021-10-01, 16:14:26
```

Figure 170

3.5.35 SIM event handler example

Sim Event Demo application. Debug prints on **USB0**, using AZX log example functions

Features

- How to use ATI function for asynchronous management
- How to cath URC from an AppZone application
- How to catch SIM related events and handle them

Application workflow

`M2MB_main.c`

- Print welcome message
- Initialize AT interface
- Initialize AT URC manager task
- Initialize SIM event manager task
- Send “AT#SIMPR=1” to activate SIM URCs
- Insert SIM in SIM slot 1 and receive SIM inserted message
- Remove SIM from SIM slot 1 and receive SIM removed message

```
Starting SIM Presence Demo app. This is v1.1.17 built on Dec 16 2022 15:14:28.
Please ensure SIM is not inserted before starting this sample application
Sending command AT#SIMPR=1 to enable SIM presence URC messages...
Command set.
SIM 0 state changed to 1!

SIM CARD HAS BEEN INSERTED!

SIM 0 state changed to 0!

SIM CARD HAS BEEN REMOVED!
```

Figure 171

3.5.36 SMS PDU

Sample application showcasing how to create and decode PDUs to be used with m2mb_sms_* API set. A SIM card and antenna must be present. Debug prints on **USB0**

Features

- How to enable SMS functionality
- How to use encode an SMS PDU to be sent with m2mb_api
- How to decode a received SMS response from PDU to ASCII mode.

3.5.36.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init sms functionality
- Create PDU from text message
- Send message to destination number
- Wait for response
- When SMS PDU response is received, decode it and print information about it, plus the message content

```
m2mb_sms_init() succeeded

Sending message <How are you?>...
 m2mb_sms_send() - succeeded
M2MB_SMS_SEND_RESP Callback
Send resp msg ID 10
SMS received!
SMS correctly received!

Reading SMS from memory...
 m2mb_sms_read() request succeeded

--- SMS read ---
SMS tag M2MB_SMS_TAG_MT_NOT_READ
SMS format M2MB_SMS_FORMAT_3GPP
Code type: 0
Sender type: 145
Msg len: 12
Msg bytes: 11
Msg date 19/7/17 16:7:58 (timezone: 2)
Received SMS, content: <<Fine thanks >>
Sender: +[redacted]
```

Figure 172

3.5.37 SMS_atCmd example

Sample application showcasing how to receive an SMS containing an AT command, process the AT command and send its answer to sender (configurable in sms_config.txt). A SIM card and antenna must be present. Debug prints on **USB0**

Features

- How to receive an SMS with an AT command as text inside
- How to send AT command to parser and read the answer
- How to send the AT command answer back to sender via SMS

Optional configuration file to be put in `/mod` folder, copy `sms_config.txt` file into your module running the following AT command:

```
AT#M2MWRITE="/mod/sms_config.txt",138
>>> here receive the prompt; then type or send the file, sized 138 bytes
```

3.5.37.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Init SMS functionality
- Read configuration file sms_config.txt (send SMS with AT command answer back, delte SMS received)
- Init AT command parser
- Create a task to handle SMS parsing and AT command sending
- Wait for an incoming SMS

`callbacks.c`

`msgSMSparse()`

- When SMS has been received, content is decoded and printed. If there is an AT command inside, command is executed and answer printed and sent back to sender as an SMS (depending on sms_config.txt setting)

```
ÿStarting SMS with AT command demo app. This is v1.0.13-C1 built on Mar 18 2021 12:42:22.
[DEBUG] 16.61  M2MB_main:135 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
m2mb_sms_init() succeeded
[DEBUG] 16.62  M2MB_main:168 - M2MB_main{M2M_DamsStart}$ M2MB_SMS_INCOMING_IND indication enabled
[DEBUG] 16.63  M2MB_main:179 - M2MB_main{M2M_DamsStart}$ M2MB_SMS_INCOMING_IND MEMORY FULL indication enabled
[DEBUG] 16.64  M2MB_main:196 - M2MB_main{M2M_DamsStart}$ Storage set to M2MB_SMS_STORAGE_SM
[DEBUG] 16.65  callbacks:114 - readConfigFromFile{M2M_DamsStart}$ Reading parameters from file
[DEBUG] 16.66  callbacks:116 - readConfigFromFile{M2M_DamsStart}$ Opening /mod/sms_config.txt in read mode..
Default: SMS with answer sending DISABLED, delete sms DISABLED
[DEBUG] 16.67  at_async:115 - at_cmd_async_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Please send an SMS with a configuration as ("ATCMD: <atcmd>")...
```

Figure 173

3.5.38 SMTP Client

Sample application showing SMTP echo demo with M2MB API. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to open a SMTP client
- How to send a mail

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage SMTP client and start it

`M2MB_main.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Initialize SMTP client and connect to SMTP server
- Prepare email and send it
- Close SMTP client
- Disable PDP context

```
Starting SMTP demo app. This is v1.1.17 built on Jan  9 2023 09:57:51.
Reading parameters from file...
Set APN to: <<nxt17.net>>
Set APN USER to: <<>>
Set APN PASS to: <<>>
Set PDP_CTX to: 1
Set MAIL SERVER to: <<           >>
Set MAIL PORT to: <<     >>
Set MAIL USER to: <<                            >>
Set MAIL PASS to: <<                    >>
Set MAIL FROM to: <<                            >>
Set MAIL FROM NAME to: <<   >>
Set MAIL SUBJECT to: <<Hello from AppZone>>
Set MAIL BODY to: <<AppZone Test>>
Set MAIL TO to: <<                           >>
Set MAIL TO NAME to: <<  >>
Waiting for registration...
Pdp context activation
Connecting to SMTP Server...
SMTP Client connected!
Preparing example's mail...
Sending example's mail...
Example's mail sent!
SMTP client closed
Application complete.
```

Figure 174

3.5.39 SPI Echo

Sample application showing how to communicate over SPI with m2mb API. Debug prints on **USB0**

Features

- How to open an SPI bus. MOSI and MISO will be shorted, to have an echo.
- How to communicate over SPI bus

3.5.39.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Open SPI bus, set parameters
- Send data on MOSI and read the same in MISO

Notes:

For LE910Cx (both Linux and ThreadX based devices), AT#SPIEN=1 command must be sent once before running the app

```
Starting SPI demo app. This is v1.0.7 built on Apr  1 2020 13:48:05.
Transfer successful. Received: hello from spi echo
```

Figure 175

3.5.40 SPI sensors

Sample application showing SPI usage, configuring two ST devices: a magnetometer (ST LIS3MDL) and a gyroscope (ST L3G4200D). The application will read values from both devices using GPIO4 and 3 (respectively) as magnetometer CS and gyro CS. Debug prints on **USB0**

Features

- How to open an SPI bus with a slave device
- How to communicate with the device over the SPI bus

Setup

- Connect sensor VDD to 3v8 supply (e.g. Vbatt on the module)
- Connect sensor GND to a GND pin of the module
- Connect sensors MOSI to module SPI_MOSI
- Connect sensors MISO to module SPI_MISO
- Connect sensors CLK to module SPI_CLK
- Connect magnetometer CS to module GPIO 2
- Connect gyroscope CS to module GPIO 3

3.5.40.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open SPI bus, set parameters
- Configure `GPIO 2` and `GPIO 3` as output, set them high (idle)
- Set registers to configure magnetometer
- Read in a loop (10 iterations) the registers carrying the 3 axes values and show the gauss value for each of them. A metal object is put close to the sensor to change the read values.
- Set registers to configure gyroscope
- Read in a loop (10 iterations) the registers carrying the 3 axes values and show the degrees per second value for each of them. The board is rotated to change the read values.

Notes:

For LE910Cx (both Linux and ThreadX based devices), `AT#SPIEN=1` command must be sent once before running the app

```
Starting SPI demo app. This is v1.0.7 built on Apr  1 2020 13:58:25.
SPI start

Magnetometer SPI Demo start
Reading Magnetometer WHOAMI. Expected: 0x3D
Expected response received!
Setting continuous conversion mode...
Continuous conversion mode successfully set.
Setting 10 Hz Output Data Rate, Medium performance mode X Y axis...
Magnetometer Enabled. 10Hz ODR, Medium Perf. Mode (X,Y).
Setting Medium performance for Z axis, little endian...
Medium Perf. Mode (Z), little endian.
Setting complete, starting reading loop...

X: 0.204 gauss
Y: -0.321 gauss
Z: 0.305 gauss

X: 0.290 gauss
Y: -0.103 gauss
Z: 0.043 gauss

X: -2.513 gauss
Y: -0.353 gauss
Z: -4.000 gauss

X: 1.980 gauss
Y: 0.174 gauss
Z: -1.945 gauss

X: 4.000 gauss
Y: -0.090 gauss
Z: -4.000 gauss

X: -0.605 gauss
Y: -0.154 gauss
Z: 0.210 gauss

X: -0.580 gauss
Y: 2.004 gauss
Z: -0.047 gauss

X: 0.177 gauss
Y: -0.359 gauss
Z: 0.295 gauss

X: 0.173 gauss
Y: -0.356 gauss
Z: 0.301 gauss

X: 0.174 gauss
Y: -0.356 gauss
Z: 0.298 gauss
Reading complete.
```

Figure 176

3.5.41 SW Timer (Software Timer)

The sample application shows how to use SW Timers M2MB API. Debug prints on **USB0**

Features

- How to open configure a SW timer
- How to use the timer to manage recurring events

3.5.41.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create sw timer structure
- Configure it with 4 seconds timeout, periodic timer (auto fires when expires)
- Init the timer with the parameters
- Start the timer
- Wait 10 seconds
- Stop the timer

`timerCb`

- Print a message with inside the callback

```
Starting SW Timers demo app. This is v1.0.7 built on Apr  7 2020 09:51:25.
timer expired!
[DEBUG] 21.41  M2MB_main.c:59 - timerCb{pubTspt_0}$ timer handle: 0x4002b004
timer expired!
[DEBUG] 25.47  M2MB_main.c:59 - timerCb{pubTspt_0}$ timer handle: 0x4002b004
stopping the timer
Stop a running timer: success
Application end
```

Figure 177

3.5.42 Secure MicroService

Sample application showcasing how to manage secure microservice functionalities. Debug prints on **USB0**

Features

- Write data in Secure Data Area (SDA), non protected
- Read the written data and compare with the original buffer
- Write a cripty key in Secure Data Area (SDA), non protected
- Perform a rotate of the written key data
- Perform MD5 sum of written data from TZ file
- Compare computed digest with expected one
- Write data in trust zone as a trusted object (it will not be possible to read it again but only use its content for crypto operations)
- Try to read the trusted object and verify it fails
- Rotate trusted item and verify retrieving the content fails
- compute MD5 sum of trusted item and compare with the expected one
- Try to pass data from a trusted item to a non trusted item using untrusted TZ buffers, and verify it fails

3.5.42.1 Application workflow

`M2MB_main.c`

- Write a buffer in a SDA item using `m2mb_secure_ms_write`
- Read the same item using `m2mb_secure_ms_read`
- Write a buffer containing some cripty key in a SDA item using `m2mb_secure_ms_write`
- Rotate the content of the key item
- Read it with `m2mb_secure_ms_read`
- Load the key content using `m2mb_secure_ms_crypto_alloc` and `m2mb_secure_crypto_add_it` in a SECURE_MS buffer
- Compute MD digest with `m2mb_secure_ms_crypto_md`
- Write a buffer containing some cripty key in a SDA item using `m2mb_secure_ms_write` but with **TRUSTED** option in `m2mb_secure_ms_open`
- Verify that `m2mb_secure_ms_read` on the trusted item fails
- Verify that `m2mb_secure_ms_crypto_rotate` fails for the trusted item
- Verify the MD5 digest
- Try to copy the trusted item data in a SECURE_MS buffer with `m2mb_secure_ms_crypto_alloc` and `m2mb_secure_crypto_add_item`, then load it in an untrusted object with `m2mb_secure_ms_crypto_write`, and verify it fails.

```
Starting secure ms demo app. This is v1.0.13-C1 built on Jul 30 2020 12:19:02.
Writing data in normal item
Stored input data in Secure Data Area
Reading data from normal item
Data length in SDA: 11 bytes
Securely loaded the data from the SDA
Read 11 bytes: <hello world>
original and retrieved strings are the same

Writing key in normal item
Stored input data in Secure Data Area

Rotate data in normal item
Original key: AA_THIS_IS_MY_SECRET_KEY_BB
Rotated key:

Compute MD5 of data in normal item
Data length in SDA: 27 bytes
MD5: 8EDAD26E26E1C74C7C02386C1C7F541D
hash is the expected one!

Writing data in trusted item
Stored input data in Secure Data Area
Reading data from trusted item (should fail!)
Data length in SDA: 27 bytes
m2mb_secure_ms_read() failed for trusted item, as expected!

Rotate data in trusted item
[ERROR] 17.01  M2MB_main:329 - read_rotate{M2M_DamsStart}$ Cannot read data from SECURE_MS_BUFFER to user buffer
Original key: AA_THIS_IS_MY_SECRET_KEY_BB
Rotated key:

Compute MD5 of data in trusted item
Data length in SDA: 27 bytes

MD5: 8EDAD26E26E1C74C7C02386C1C7F541D
Hash is the expected one!

Try to pass data from trusted to untrusted through TZ buffers
Cannot store data from SECURE_MS_BUFFER to SDA 'non-trusted', as expected
```

Figure 178

3.5.43 TCP IP

Sample application showcasing TCP echo demo with M2MB API. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to open a TCP client socket
- How to communicate over the socket

3.5.43.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server
- Send data and receive response
- Close socket
- Disable PDP context

```
Starting TCP-IP demo app. This is v1.0.7 built on Mar 26 2020 16:20:30.
[DEBUG] 21.23  m2m_tcp_test.c:201 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 21.25  m2m_tcp_test.c:217 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.26  m2m_tcp_test.c:223 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.26  m2m_tcp_test.c:231 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 21.28  m2m_tcp_test.c:128 - NetCallback{pubTspt_0}$ Module is registered to cell 0x816B!
[DEBUG] 21.29  m2m_tcp_test.c:244 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 21.30  m2m_tcp_test.c:248 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.34  m2m_tcp_test.c:263 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 24.52  m2m_tcp_test.c:155 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.52  m2m_tcp_test.c:158 - PdpCallback{pubTspt_0}$ IP address: 83.225.44.56
[DEBUG] 24.54  m2m_tcp_test.c:273 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 24.54  m2m_tcp_test.c:284 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 24.55  m2m_tcp_test.c:294 - M2M_msgTCPTask{TCP_TASK}$ Socket ctx set to 3
[DEBUG] 24.95  m2m_tcp_test.c:307 - M2M_msgTCPTask{TCP_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 25.17  m2m_tcp_test.c:322 - M2M_msgTCPTask{TCP_TASK}$ Socket Connected!
[DEBUG] 25.18  m2m_tcp_test.c:329 - M2M_msgTCPTask{TCP_TASK}$ Sending data over socket..
[DEBUG] 25.19  m2m_tcp_test.c:342 - M2M_msgTCPTask{TCP_TASK}$ Data send successfully (16 bytes)
[DEBUG] 27.20  m2m_tcp_test.c:356 - M2M_msgTCPTask{TCP_TASK}$ trying to receive 16 bytes..
[DEBUG] 27.21  m2m_tcp_test.c:364 - M2M_msgTCPTask{TCP_TASK}$ Data received (16): <hello from m2mb!>
[DEBUG] 27.21  m2m_tcp_test.c:373 - M2M_msgTCPTask{TCP_TASK}$ application exit
[DEBUG] 27.22  m2m_tcp_test.c:385 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 27.24  m2m_tcp_test.c:388 - M2M_msgTCPTask{TCP_TASK}$ Application complete.
[DEBUG] 29.43  m2m_tcp_test.c:164 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 179

3.5.44 TCP non blocking example

Sample application that shows how to configure and connect a TCP-IP non blocking socket. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to open a TCP client non Blocking socket
- How to communicate over the socket

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Set the socket as non Blocking and connect to server. Uses m2mb_socket_bsd_select, m2mb_socket_bsd_fd_isset_func to check when socket is connected.
- Send data and receive response
- Close socket
- Disable PDP context

```
Starting TCP-IP non blocking demo app. This is v1.1.9 built on Mar 22 2022 15:35:56.
[DEBUG] 19.24  m2mb_tcp_te:215 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 19.25  m2mb_tcp_te:231 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 19.25  m2mb_tcp_te:237 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 19.26  m2mb_tcp_te:245 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 19.27  m2mb_tcp_te:142 - NetCallback{pubTspt_0}$ Module is registered to cell 0x20!
[DEBUG] 19.28  m2mb_tcp_te:258 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 19.29  m2mb_tcp_te:262 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.34  m2mb_tcp_te:277 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN ibox.tim.it....
[DEBUG] 21.89  m2mb_tcp_te:169 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 21.90  m2mb_tcp_te:172 - PdpCallback{pubTspt_0}$ IP address: 2.195.160.174
[DEBUG] 21.90  m2mb_tcp_te:287 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 21.91  m2mb_tcp_te:299 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 21.92  m2mb_tcp_te:309 - M2M_msgTCPTask{TCP_TASK}$ Socket ctx set to 1
[DEBUG] 22.10  m2mb_tcp_te:322 - M2M_msgTCPTask{TCP_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 22.11  m2mb_tcp_te:332 - M2M_msgTCPTask{TCP_TASK}$ Socket set to nonBlocking
[DEBUG] 22.11  m2mb_tcp_te:351 - M2M_msgTCPTask{TCP_TASK}$ Connection in progress...
[DEBUG] 22.29  m2mb_tcp_te:377 - M2M_msgTCPTask{TCP_TASK}$ Socket <0x400343a0> is connected!
[DEBUG] 22.30  m2mb_tcp_te:398 - M2M_msgTCPTask{TCP_TASK}$ Socket Connected!
[DEBUG] 22.32  m2mb_tcp_te:411 - M2M_msgTCPTask{TCP_TASK}$ Sending data over socket..
[DEBUG] 22.32  m2mb_tcp_te:424 - M2M_msgTCPTask{TCP_TASK}$ Data send successfully (16 bytes)
[DEBUG] 24.37  m2mb_tcp_te:438 - M2M_msgTCPTask{TCP_TASK}$ trying to receive 16 bytes..
[DEBUG] 24.38  m2mb_tcp_te:446 - M2M_msgTCPTask{TCP_TASK}$ Data received (16): <hello from m2mb!>
[DEBUG] 24.39  m2mb_tcp_te:456 - M2M_msgTCPTask{TCP_TASK}$ application exit
[DEBUG] 24.43  m2mb_tcp_te:468 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 24.45  m2mb_tcp_te:471 - M2M_msgTCPTask{TCP_TASK}$ Application complete.
[DEBUG] 24.79  m2mb_tcp_te:178 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 180

3.5.45 TCP Socket status

Sample application showcasing how to check a TPC connected socket current status. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to open a TCP client socket
- How to check if the TCP socket is still valid

3.5.45.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server
- Check in a loop the current socket status using the adv_select function with a 2 seconds timeout
- Close socket when the remote host closes it
- Disable PDP context

```
Starting TCP socket status check demo app. This is v1.0.14-C1 built on Sep  8 2020 14:59:25.
[DEBUG] 21.33  m2m_tcp_tes:324 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 21.34  m2m_tcp_tes:338 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.34  m2m_tcp_tes:344 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.35  m2m_tcp_tes:352 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 21.36  m2m_tcp_tes:365 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 21.37  m2m_tcp_tes:369 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.41  m2m_tcp_tes:384 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN NXT17.NET....
[DEBUG] 24.09  m2m_tcp_tes:281 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.10  m2m_tcp_tes:284 - PdpCallback{pubTspt_0}$ IP address: 100.77.5.223
[DEBUG] 24.10  m2m_tcp_tes:394 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 24.11  m2m_tcp_tes:405 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 24.11  m2m_tcp_tes:415 - M2M_msgTCPTask{TCP_TASK}$ Socket ctx set to 3
[DEBUG] 24.60  m2m_tcp_tes:428 - M2M_msgTCPTask{TCP_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 24.93  m2m_tcp_tes:443 - M2M_msgTCPTask{TCP_TASK}$ Socket Connected!
[DEBUG] 26.98  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 29.03  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...

...

[DEBUG] 82.18  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 84.23  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 86.28  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 88.31  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 88.90  m2m_tcp_tes:154 - adv_select{TCP_TASK}$ Data is available on socket <0x40032b3c>
[DEBUG] 88.92  m2m_tcp_tes:160 - adv_select{TCP_TASK}$ There are <0> pending bytes on socket
Socket was closed by remote!
[DEBUG] 88.92  m2m_tcp_tes:494 - M2M_msgTCPTask{TCP_TASK}$ application exit
[DEBUG] 88.94  m2m_tcp_tes:506 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 88.94  m2m_tcp_tes:509 - M2M_msgTCPTask{TCP_TASK}$ Application complete.
[DEBUG] 89.31  m2m_tcp_tes:290 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 181

3.5.46 TCP Server

Sample application showcasing TCP listening socket demo with M2MB API. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to open a TCP listening socket
- How to manage external hosts connection and exchange data

3.5.46.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and set it in non-blocking mode
- Bind the socket to the listening port
- Start listening for incoming connection
- Check if a connection is incoming using m2mb_socket_bsd_select function
- If a client connects, perform accept on the child socket
- Send a “START” message to the client
- Send some data
- Wait for data from client and print it
- Close the child socket
- Start listening again, up to 3 times
- Close listening socket
- Disable PDP context

Debug Log

```
Starting TCP Server demo app. This is v1.0.7 built on Apr  7 2020 13:28:24.
[DEBUG] 14.55  m2m_tcp_test.c:220 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 14.55  m2m_tcp_test.c:236 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 14.57  m2m_tcp_test.c:242 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 14.57  m2m_tcp_test.c:250 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 14.58  m2m_tcp_test.c:138 - NetCallback{pubTspt_0}$ Module is registered to cell 0x5222!
[DEBUG] 14.59  m2m_tcp_test.c:263 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 14.60  m2m_tcp_test.c:267 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 16.57  m2m_tcp_test.c:282 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN ibox.tim.it....
[DEBUG] 17.16  m2m_tcp_test.c:165 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 17.17  m2m_tcp_test.c:168 - PdpCallback{pubTspt_0}$ IP address: 2.195.165.137


--------------------------------------------
|                                          |
|  Start TCP server                        |
|                                          |
--------------------------------------------


[DEBUG] 19.15  m2m_tcp_test.c:301 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 19.15  m2m_tcp_test.c:312 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 19.16  m2m_tcp_test.c:313 - M2M_msgTCPTask{TCP_TASK}$ m2mb_socket_bsd_socket(): valid socket ID [0x4002E79C] - PASS
[DEBUG] 20.16  m2m_tcp_test.c:319 - M2M_msgTCPTask{TCP_TASK}$ issuing m2m_socket_bsd_ioctl() to set non-blocking mode ...
[DEBUG] 20.17  m2m_tcp_test.c:331 - M2M_msgTCPTask{TCP_TASK}$ Binding Socket...
[DEBUG] 22.12  m2m_tcp_test.c:343 - M2M_msgTCPTask{TCP_TASK}$ Socket Bind Pass


Start TCP listening on port 6500...

[DEBUG] 24.13  m2m_tcp_test.c:368 - M2M_msgTCPTask{TCP_TASK}$ select...
Select result: 0
[DEBUG] 28.13  m2m_tcp_test.c:368 - M2M_msgTCPTask{TCP_TASK}$ select...
Select result: 1

TCP Server Coming Connection

--> Accept
[DEBUG] 30.52  m2m_tcp_test.c:397 - M2M_msgTCPTask{TCP_TASK}$ Socket Accept Pass


Connected! (socket dial n.1)

[DEBUG] 30.53  m2m_tcp_test.c:403 - M2M_msgTCPTask{TCP_TASK}$ Client Source Address: 185.86.42.254
[DEBUG] 30.54  m2m_tcp_test.c:404 - M2M_msgTCPTask{TCP_TASK}$ Client Port: 58658
[DEBUG] 30.54  m2m_tcp_test.c:405 - M2M_msgTCPTask{TCP_TASK}$ Client Family: 2
[DEBUG] 31.56  m2m_tcp_test.c:410 - M2M_msgTCPTask{TCP_TASK}$
-------------------------------
[DEBUG] 31.57  m2m_tcp_test.c:411 - M2M_msgTCPTask{TCP_TASK}$ |    Send/receive data test    |
[DEBUG] 31.57  m2m_tcp_test.c:412 - M2M_msgTCPTask{TCP_TASK}$ -------------------------------

[DEBUG] 32.58  m2m_tcp_test.c:416 - M2M_msgTCPTask{TCP_TASK}$
--> issuing m2mb_socket_bsd_send(): transmit "__START____" packet...
[DEBUG] 32.59  m2m_tcp_test.c:423 - M2M_msgTCPTask{TCP_TASK}$  --> done (11 have been transmitted)
[DEBUG] 32.60  m2m_tcp_test.c:425 - M2M_msgTCPTask{TCP_TASK}$      ALL data transmitted - PASS
[DEBUG] 32.61  m2m_tcp_test.c:430 - M2M_msgTCPTask{TCP_TASK}$
--> issuing m2mb_socket_bsd_send(): transmit 58 bytes...
[DEBUG] 32.62  m2m_tcp_test.c:437 - M2M_msgTCPTask{TCP_TASK}$  --> done (58 have been transmitted)
[DEBUG] 32.63  m2m_tcp_test.c:440 - M2M_msgTCPTask{TCP_TASK}$      ALL data transmitted - PASS
[DEBUG] 32.64  m2m_tcp_test.c:448 - M2M_msgTCPTask{TCP_TASK}$
Waiting for data...

[DEBUG] 39.64  m2m_tcp_test.c:457 - M2M_msgTCPTask{TCP_TASK}$ test
[DEBUG] 99.61  m2m_tcp_test.c:465 - M2M_msgTCPTask{TCP_TASK}$
m2mb_socket_bsd_recv() has received 6 bytes

[DEBUG] 102.60  m2m_tcp_test.c:469 - M2M_msgTCPTask{TCP_TASK}$
Server TCP is closing the current connection ...
```

Figure 182

Data on a PuTTY terminal

```
START

aaaaaaaaaa-bbbbbbbbbb-cccccccccc-dddddddddd-eeeeeeeeee

test
```

Figure 183

3.5.47 TLS SSL Client

Sample application showcasing TLS/SSL with client certificates usage with M2MB API. Debug prints on **USB0**

Features

- How to check module registration and enable PDP context
- How to open a SSL client socket
- How to communicate over SSL socket

3.5.47.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the connection and start it

`ssl_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server over TCP socket
- Initialize the TLS parameters (TLS1.2) andh auth mode (server+client auth in the example)
- Create SSL context
- Read certificates files and store them
- Create secure socket and connect to the server using SSL
- Send data and receive response
- Close secure socket
- Close socket
- Delete SSL context
- Disable PDP context

The application requires the certificates to be stored in `/mod/ssl_certs/` folder. It can be created with

`AT#M2MMKDIR=/mod/ssl_certs`

Certificates can then be loaded with

AT#M2MWRITE="/mod/ssl_certs/modulesCA.crt",1740

and providing the file content in RAW mode (for example using the “Transfer Data” button in Telit AT Controller)

For client certificates (if required), the commands will be

AT#M2MWRITE="/mod/ssl_certs/modulesClient.crt",1651
AT#M2MWRITE="/mod/ssl_certs/modulesClient_pkcs1.key",1679

PLEASE NOTE: always verify the file sizes to be used in the commands above as they might change

```
Starting TLS-SSL demo app. This is v1.1.2 built on Mar  3 2021 10:15:00.
[DEBUG] 10.85  ssl_test:252 - msgHTTPSTask{TLS_TASK}$ INIT
[DEBUG] 10.85  ssl_test:266 - msgHTTPSTask{TLS_TASK}$ m2mb_os_ev_init success
[DEBUG] 10.85  ssl_test:270 - msgHTTPSTask{TLS_TASK}$ Init SSL session test app
[DEBUG] 10.85  ssl_test:285 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config sslConfigHndl = 0x40037958, sslRes= 0
[DEBUG] 10.85  ssl_test:294 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config PASSED
[DEBUG] 10.85  ssl_test:306 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_ctxt PASSED
[DEBUG] 10.85  ssl_test:311 - msgHTTPSTask{TLS_TASK}$ loading CA CERT from file /mod/ssl_certs/modulesCA.crt
[DEBUG] 10.85  ssl_test:315 - msgHTTPSTask{TLS_TASK}$ file size: 1740
[DEBUG] 10.85  ssl_test:328 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1740
Buffer successfully received from file. 1740 bytes were loaded.
Closing file.
[DEBUG] 10.85  ssl_test:361 - msgHTTPSTask{TLS_TASK}$ loading client CERT from file /mod/ssl_certs/modulesClient.crt
[DEBUG] 10.85  ssl_test:365 - msgHTTPSTask{TLS_TASK}$ file size: 1651
[DEBUG] 10.85  ssl_test:378 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1651
Buffer successfully received from file. 1651 bytes were loaded.
Closing file.
[DEBUG] 10.85  ssl_test:401 - msgHTTPSTask{TLS_TASK}$ loading client KEY from file /mod/ssl_certs/modulesClient_pkcs1.key
[DEBUG] 10.85  ssl_test:405 - msgHTTPSTask{TLS_TASK}$ file size: 1679
[DEBUG] 10.85  ssl_test:418 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1679
Buffer successfully received from file. 1679 bytes were loaded.
Closing file.
[DEBUG] 10.85  ssl_test:448 - msgHTTPSTask{TLS_TASK}$ certificates successfully stored!
[DEBUG] 10.85  ssl_test:457 - msgHTTPSTask{TLS_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 10.85  ssl_test:465 - msgHTTPSTask{TLS_TASK}$ Waiting for registration...
[DEBUG] 10.86  ssl_test:171 - NetCallback{pubTspt_0}$ Module is registered to cell 0x468E!
[DEBUG] 10.86  ssl_test:477 - msgHTTPSTask{TLS_TASK}$ Pdp context activation
[DEBUG] 10.86  ssl_test:481 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 12.87  ssl_test:496 - msgHTTPSTask{TLS_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 13.71  ssl_test:197 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 13.71  ssl_test:200 - PdpCallback{pubTspt_0}$ IP address: 2.41.76.63
[DEBUG] 13.71  ssl_test:514 - msgHTTPSTask{TLS_TASK}$ Creating Socket...
[DEBUG] 13.71  ssl_test:525 - msgHTTPSTask{TLS_TASK}$ Socket created
[DEBUG] 13.71  ssl_test:535 - msgHTTPSTask{TLS_TASK}$ Socket ctx set to 3
[DEBUG] 13.92  ssl_test:548 - msgHTTPSTask{TLS_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 14.05  ssl_test:562 - msgHTTPSTask{TLS_TASK}$ Socket Connected!
[DEBUG] 15.97  ssl_test:587 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_connect ret 0
[DEBUG] 17.99  ssl_test:593 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
```

```
[DEBUG] 17.99  ssl_test:593 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
[DEBUG] 17.99  ssl_test:596 - msgHTTPSTask{TLS_TASK}$ SSL write result = 44
[DEBUG] 22.03  ssl_test:608 - msgHTTPSTask{TLS_TASK}$ pending bytes: 1087
[DEBUG] 22.03  ssl_test:612 - msgHTTPSTask{TLS_TASK}$ trying to receive 1087 bytes..
[DEBUG] 22.03  ssl_test:618 - msgHTTPSTask{TLS_TASK}$ Server response: (269)<HTTP/1.1 200 OK
Date: Wed, 03 Mar 2021 09:18:22 GMT
Server: Apache/2.2.15 (CentOS)
Last-Modified: Mon, 22 Jan 2018 10:57:39 GMT
ETag: "1fffc-27f-5635b4c6f12b3"
Accept-Ranges: bytes
Content-Length: 639
Connection: close
Content-Type: text/html; charset=UTF-8

>
[DEBUG] 22.03  ssl_test:634 - msgHTTPSTask{TLS_TASK}$ pending bytes: 762
[DEBUG] 22.03  ssl_test:638 - msgHTTPSTask{TLS_TASK}$ trying to receive remaining 762 bytes..
[DEBUG] 22.03  ssl_test:644 - msgHTTPSTask{TLS_TASK}$ Server response: (639)<<html>
<head>
 <title>module.telit.com</title>
 <meta content="text/html; charset=utf-8" />
</head>
<body>
 <table border=0 align=center>
 <tr>
 <td height="100" align=center><h2>modules.telit.com - Test HTML page</h2></td>
 </tr>
 <tr>
 <td align=center><img src=Telit.jpg alt="Telit logo" height="126" width="410"></img></td>
 </tr>
 <tr>
 <td height="200" align=center> This is a simple HTML page, </br>
 made with simple HTML code,</br>
 just for test!
 </td>
 </tr>
 <tr>
 <td height="100" align=center><font size="3">Telit &copy; 2015 - 2017 All rights reserved</font></td>
 </tr>
 </table>
</body>
</html>
>
[DEBUG] 22.03  ssl_test:662 - msgHTTPSTask{TLS_TASK}$ application exit
[DEBUG] 22.03  ssl_test:680 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 22.03  ssl_test:683 - msgHTTPSTask{TLS_TASK}$ Application complete.
[DEBUG] 22.77  ssl_test:206 - PdpCallback{pubTspt_0}$ Context deactivated!
```

3.5.48 UDP client

Sample application showcasing UDP echo demo with M2MB API. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to open a UDP client socket
- How to communicate over the socket

3.5.48.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task and start it

`m2m_udp_test.c` - Initialize Network structure and check registration - Initialize PDP structure and start PDP context - Create socket and link it to the PDP context id - Send data and receive response - Close socket - Disable PDP context

```
Starting UDP client demo app. This is v1.0.7 built on Apr  1 2020 14:57:13.
INIT
[DEBUG] 21.23  m2m_udp_test.c:223 - M2M_msgUDPTask{UDP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
Waiting for registration...
[DEBUG] 21.25  m2m_udp_test.c:131 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 21.26  m2m_udp_test.c:241 - M2M_msgUDPTask{UDP_TASK}$ Pdp context initialization
[DEBUG] 21.26  m2m_udp_test.c:245 - M2M_msgUDPTask{UDP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
Activate PDP with APN web.omnitel.it....
[DEBUG] 24.11  m2m_udp_test.c:157 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.11  m2m_udp_test.c:160 - PdpCallback{pubTspt_0}$ IP address: 109.113.222.12
[DEBUG] 24.12  m2m_udp_test.c:268 - M2M_msgUDPTask{UDP_TASK}$ Creating Socket...
[DEBUG] 24.13  m2m_udp_test.c:280 - M2M_msgUDPTask{UDP_TASK}$ Socket created
Socket ctx set to 3
[DEBUG] 24.41  m2m_udp_test.c:306 - M2M_msgUDPTask{UDP_TASK}$ Retrieved IP: 185.86.42.218
Socket ready.
Data successfully sent (16 bytes)
Socket recv...
[DEBUG] 26.47  m2m_udp_test.c:352 - M2M_msgUDPTask{UDP_TASK}$ m2mb_socket_bsd_set_sock_opt() M2MB_SOCKET_BSD_SO_RCVTIMEO - success
trying to receive 16 bytes..
Data received (16): <hello from m2mb!>
[DEBUG] 26.48  m2m_udp_test.c:377 - M2M_msgUDPTask{UDP_TASK}$ application exit
Socket Closed
[DEBUG] 26.49  m2m_udp_test.c:399 - M2M_msgUDPTask{UDP_TASK}$ m2mb_pdp_deactivate returned success
Application complete.
[DEBUG] 27.04  m2m_udp_test.c:166 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 184

3.5.49 UDP_Server example

Sample application that shows UDP listening socket demo with m2mb apis. Debug prints on **USB0**

Features

- How to configure an UDP socket into listen mode
- How to receive data using m2mb_socket_bsd_select
- How to read data received and send data to client

Application workflow

`M2MB_main.c`

- Print welcome message
- Init task apis and create M2M_msgUDPTask to handle UDP socket

`m2mb_udp_test.c`

`M2M_msgUDPTask`

- Wait for module registration
- Activate PDP context
- Create UDP listen socket
- Wait for incoming data from client using m2mb_socket_bsd_select
- When there are data on socket, read them and send some data back to client

```
Starting UDP Server demo app. This is v1.1.10 built on Mar  3 2022 15:42:39.

[DEBUG] 19.28  m2mb_udp_te:221 - M2M_msgUDPTask{UDP_TASK}$ INIT
[DEBUG] 19.29  m2mb_udp_te:237 - M2M_msgUDPTask{UDP_TASK}$ m2mb_os_ev_init success
[DEBUG] 19.30  m2mb_udp_te:243 - M2M_msgUDPTask{UDP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 19.31  m2mb_udp_te:251 - M2M_msgUDPTask{UDP_TASK}$ Waiting for registration...
[DEBUG] 19.32  m2mb_udp_te:140 - NetCallback{pubTspt_0}$ Module is registered to cell 0x5222!
[DEBUG] 19.33  m2mb_udp_te:264 - M2M_msgUDPTask{UDP_TASK}$ Pdp context activation
[DEBUG] 19.33  m2mb_udp_te:268 - M2M_msgUDPTask{UDP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.39  m2mb_udp_te:283 - M2M_msgUDPTask{UDP_TASK}$ Activate PDP with APN ibox.tim.it....
[DEBUG] 22.00  m2mb_udp_te:167 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 22.01  m2mb_udp_te:170 - PdpCallback{pubTspt_0}$ IP address: 2.195.163.102

----    Start UDP server  Listen IP: 2.195.163.102   PORT: 6500    ----

[DEBUG] 24.07  m2mb_udp_te:299 - M2M_msgUDPTask{UDP_TASK}$ Creating Socket...
[DEBUG] 24.08  m2mb_udp_te:310 - M2M_msgUDPTask{UDP_TASK}$ Socket created
[DEBUG] 24.08  m2mb_udp_te:311 - M2M_msgUDPTask{UDP_TASK}$ m2mb_socket_bsd_socket(): valid socket ID [0x40035490] - PASS
[DEBUG] 25.12  m2mb_udp_te:317 - M2M_msgUDPTask{UDP_TASK}$ issuing m2m_socket_bsd_ioctl() to set non-blocking mode ...
[DEBUG] 25.13  m2mb_udp_te:329 - M2M_msgUDPTask{UDP_TASK}$ Binding Socket...
[DEBUG] 27.18  m2mb_udp_te:341 - M2M_msgUDPTask{UDP_TASK}$ Socket Bind Pass

Waiting for Incoming UDP data, Listen IP: 2.195.163.102 PORT: 6500

[DEBUG] 29.25  m2mb_udp_te:359 - M2M_msgUDPTask{UDP_TASK}$ select...
[DEBUG] 32.32  m2mb_udp_te:362 - M2M_msgUDPTask{UDP_TASK}$ Select result: 0
[DEBUG] 33.34  m2mb_udp_te:359 - M2M_msgUDPTask{UDP_TASK}$ select...
[DEBUG] 36.41  m2mb_udp_te:362 - M2M_msgUDPTask{UDP_TASK}$ Select result: 0
[DEBUG] 37.45  m2mb_udp_te:359 - M2M_msgUDPTask{UDP_TASK}$ select...
[DEBUG] 40.52  m2mb_udp_te:362 - M2M_msgUDPTask{UDP_TASK}$ Select result: 0
[DEBUG] 41.54  m2mb_udp_te:359 - M2M_msgUDPTask{UDP_TASK}$ select...
[DEBUG] 44.61  m2mb_udp_te:362 - M2M_msgUDPTask{UDP_TASK}$ Select result: 0
[DEBUG] 45.64  m2mb_udp_te:359 - M2M_msgUDPTask{UDP_TASK}$ select...
[DEBUG] 45.94  m2mb_udp_te:362 - M2M_msgUDPTask{UDP_TASK}$ Select result: 1
[DEBUG] 45.95  m2mb_udp_te:366 - M2M_msgUDPTask{UDP_TASK}$
Incoming UDP data available
----    Receive/send data test    ----

trying to receive bytes..
[DEBUG] 46.99  m2mb_udp_te:399 - M2M_msgUDPTask{UDP_TASK}$ Client Source Address: 151.99.1.17
[DEBUG] 47.00  m2mb_udp_te:400 - M2M_msgUDPTask{UDP_TASK}$ Client Port: 36368
[DEBUG] 47.01  m2mb_udp_te:401 - M2M_msgUDPTask{UDP_TASK}$ Client Family: 2
Data received (4): <test>
Data successfully sent (16 bytes)
```

Figure 185

3.5.50 Watchdog example

Sample application that shows how to set and implement a watchdog. Task has been locked waitng for an event with a timeout longer than wd inactivity timeout. If no wd kick or no actions

Features

- Enable watchdog on Task_1
- Handle watchdog kicks with a timer

3.5.50.1 Application workflow

M2MB_main.c

- Create Task_1 that will be put under watchdog control
- In Task_1 watchdog is ebnabled on it and a timer created to handle watchdog kicks. Then task enters in a loop where kicks are sent. After 5 kicks a task lock is simulated waiting for an event release (with a timeout longer that task watchdog inactivity timeout)
- As soon as watchdog inactivity timeout expired M2MB_WDOG_TIMEOUT_IND is generated and handled in WDcallback (in this case the event is released, no further actions are done)
- No further actions are performed by Task_1 (no kicks are sent) so after watchdog system timeout expiration app is rebooted.

```
Starting WATCHDOG demo app. This is v1.0.0 built on Nov  4 2021 14:34:06.

Event correctly initialized

Task started

Init WatchDog
m2mb_wDog_init OK
Tick duration: 1s
Adding Task under WD control with inactivity timeout of 60s
m2mb_wDog_addTask OK
Enabling the WatchDog
m2mb_wDog_enable OK
Timer creation OK

Start WD kick every 5s...
WD kick
WD kick done
```

Figure 186

3.5.51 ZLIB example

Sample application showing how to compress/uncompress with ZLIB. Debug prints on **USB0**

Features

- How to compress a file
- How to uncompress a file

In order to execute the entire test, copy `test.gz` file into your module running the following AT command:

```
AT#M2MWRITE="/mod/test.gz",138
>>> here receive the prompt; then type or send the file, sized 138 bytes
```

3.5.51.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Test the compression and decompression of a data string
- Test the decompression of a .gz file (test.gz), expected to be in `/mod` folder, into its content `test.txt`. The file must be uploaded by the user (see steps above).

```
Starting Logging demo app. This is v1.0.7 built on Apr  7 2020 09:02:35.

Starting TEST_COMPR_UNCOMPR.
len: 138; comprLen: 57
Compressed message:
 W+ÉHU(,ÍLÏVH*É/ÍSHÉ¨ PÉ*Í-HMQÉ/K-R(  Ëç$VU*ëä§ë y4RÍ«¥1、
comprLen: 57; uncomprLen: 138
uncompress():
the quick brown fox jumped over the lazy dog. the quick brown fox jumped over the lazy dog. the quick brown fox jumped over the lazy dog.
Ending TEST_COMPR_UNCOMPR with SUCCESS.

Starting test_uncompress.

Data extracted correctly into the file ./mod/test.txt
test_uncompress finished correctly!
```

Figure 187

3.6 MAIN UART

Applications that provide usage examples for various functionalities, log output on MAIN UART

3.6.1 ATI (AT Instance)

Sample application showing how to use AT Instance functionality (sending AT commands from code). The example supports both sync and async (using a callback) modes. Debug prints on **MAIN UART**

Features

- How to open an AT interface from the application
- How to send AT commands and receive responses on the AT interface

3.6.1.1 Application workflow, sync mode

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init AT0 (first AT instance)
- Send AT+CGMR command
- Print response.
- Release AT0

`at_sync.c`

- Init ati functionality and take AT0
- Send AT+CGMR command, then read response after 2 seconds, then return it
- Deinit ati, releasing AT0

```
Starting AT demo app. This is v1.0.7 built on Apr  1 2020 15:12:58.
[DEBUG] 17.15  at_sync.c:53 - at_cmd_sync_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Sending command AT+CGMR in sync mode
[DEBUG] 17.16  at_sync.c:79 - send_sync_at_command{M2M_DamsStart}$ Sending AT Command: AT+CGMR
Command response: <AT+CGMR
M0B.950004-B008

OK
>

[DEBUG] 19.21  at_sync.c:61 - at_cmd_sync_deinit{M2M_DamsStart}$ m2mb_ati_deinit() on instance 0
Application end
```

Figure 188

3.6.1.2 Application workflow, async mode

M2MB_main.c

- Open USB/UART/UART_AUX
- Init AT0 (first AT instance)
- Send AT+CGMR command
- Print response.
- Release AT0

at_async.c

- Init ati functionality and take AT0, register AT events callback
- Send AT+CGMR command, wait for response semaphore (released in callback), then read it and return it
- Deinit ati, releasing AT0

```
Starting AT demo app. This is v1.0.7 built on Apr  1 2020 15:07:45.
[DEBUG] 17.13  at_async.c:116 - at_cmd_async_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Sending command AT+CGMR in async mode
[DEBUG] 17.15  at_async.c:153 - send_async_at_command{M2M_DamsStart}$ Sending AT Command: AT+CGMR
[DEBUG] 17.15  at_async.c:169 - send_async_at_command{M2M_DamsStart}$ waiting command response...
[DEBUG] 17.17  at_async.c:88 - at_cmd_async_callback{pubTspt_0}$ Callback - available bytes: 25
[DEBUG] 17.18  at_async.c:181 - send_async_at_command{M2M_DamsStart}$ Receive response...
Command response: <AT+CGMR
M0B.950004-B008

OK
>

[DEBUG] 17.19  at_async.c:136 - at_cmd_async_deinit{M2M_DamsStart}$ m2mb_ati_deinit() on instance 0
Application end
```

Figure 189

3.6.2 AT Tunnel

Sample application showcasing how to perform an AT tunnel from Main UART to an AT instance. Debug prints on **USB1**.

Features

- How to open an AT interface from the application
- How to receive data from main UART and tunnel it to the AT interface, then report back to UART the AT response

3.6.2.1 Application workflow

`M2MB_main.c`

- Open USB1 for debug
- Initialize UART with callback function to manage input data
- Initialize AT system to manage AT commands from UART
- wait 5 minutes then deinit AT system

Main UART:

```
Starting AT tunnel demo app. Waiting for AT commands...
AT+CGMM
ME910C1-P2

OK
AT+CGREG?
+CGREG: 0,1

OK
```

Figure 190

USB1 debug log:

```
Starting AT tunnel demo app. This is v1.0.7 built on Apr  7 2020 08:21:41.
Uart opened, setting callback for data..
[DEBUG] 17.21  M2MB_main.c:183 - at_cmd_async_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
[DEBUG] 20.43  M2MB_main.c:144 - UART_Cb{pubTspt_0}$ Received 8 bytes
[DEBUG] 20.43  M2MB_main.c:84 - msgUARTTask{uart_task}$ Received data on uart, read it and send on ATI
UART IN: <AT+CGMM
>. Sending to ATI...
[DEBUG] 20.43  M2MB_main.c:171 - at_cmd_async_callback{pubTspt_0}$ Callback - available bytes: 8
[DEBUG] 20.43  M2MB_main.c:107 - msgUARTTask{uart_task}$ Received data on ATI, read it and send on UART
[DEBUG] 20.43  M2MB_main.c:116 - msgUARTTask{uart_task}$ Received: <AT+CGMM
>
[DEBUG] 20.43  M2MB_main.c:171 - at_cmd_async_callback{pubTspt_0}$ Callback - available bytes: 20
[DEBUG] 20.43  M2MB_main.c:107 - msgUARTTask{uart_task}$ Received data on ATI, read it and send on UART
[DEBUG] 20.43  M2MB_main.c:116 - msgUARTTask{uart_task}$ Received: <
ME910C1-P2

OK
>
[DEBUG] 32.82  M2MB_main.c:144 - UART_Cb{pubTspt_0}$ Received 10 bytes
[DEBUG] 32.82  M2MB_main.c:84 - msgUARTTask{uart_task}$ Received data on uart, read it and send on ATI
UART IN: <AT+CGREG?
>. Sending to ATI...
[DEBUG] 32.82  M2MB_main.c:171 - at_cmd_async_callback{pubTspt_0}$ Callback - available bytes: 10
[DEBUG] 32.82  M2MB_main.c:107 - msgUARTTask{uart_task}$ Received data on ATI, read it and send on UART
[DEBUG] 32.82  M2MB_main.c:116 - msgUARTTask{uart_task}$ Received: <AT+CGREG?
>
[DEBUG] 32.83  M2MB_main.c:171 - at_cmd_async_callback{pubTspt_0}$ Callback - available bytes: 21
[DEBUG] 32.83  M2MB_main.c:107 - msgUARTTask{uart_task}$ Received data on ATI, read it and send on UART
[DEBUG] 32.83  M2MB_main.c:116 - msgUARTTask{uart_task}$ Received: <
+CGREG: 0,1

OK
>
```

Figure 191

3.6.3 AWS demo

Sample application showcasing AWS Iot Core MQTT communication. Debug prints on **MAIN UART**

Features

- How to check module registration and enable PDP context
- How to load certificates into device SSL session storage
- How to configure MQTT client parameters
- How to connect to AWS server with SSL and exchange data over a topic

3.6.3.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage MQTT client and start it

`aws_demo.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init MQTT client
- Configure it with all parameters (Client ID, PDP context ID, keepalive timeout…)
- Initialize the TLS parameters (TLS1.2) andh auth mode (server+client auth in the example)
- Create SSL context
- Read certificates files and store them
- Connect MQTT client to broker
- Subscribe to topic
- Publish 10 messages with increasing counter
- Print received message in mqtt_topc_cb function
- Disconnect MQTT client and deinit it
- Disable PDP context

3.6.3.2 How to get started with AWS IoT

- Go to AWS console and create an account if one is not available yet.
- Go to **IoT Core** section
- Go to **Secure** > **Policies** section
- Create a new policy, which describes what the device will be allowed to do (e.g. subscribe, publish)
- Give it a name, then configure it using the configuration below (it is possible to copy/paste by clicking on **Add statements** section, then **Advanced mode**) :

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
"Action": [
  "iot:Publish",
  "iot:Subscribe",
  "iot:Connect",
  "iot:Receive"
],
"Effect": "Allow",
"Resource": [
  "*"
]
  }
  ]
}
```

- Click on create to complete the policy creation.
- Go to **Manage** section
- Press **Create**, then **Create a single thing**
- Give the new thing a name, then click on Next
- Select **One-click certificate creation (recommended)** by clicking on **Create certificate**
- Once presented with the **Certificate created** page, download all certificates and keys
- Click on the **Activate** button to enable the certificate authentication of the newly created device
- Click on **Attach a policy** and select the policy created in a previous step

For further information, please refer to the full AWS IoT documentation

3.6.3.3 Application setup

- Set **CLIENTCERTFILE** and **CLIENTKEYFILE** defines in **aws_demo.c file** in order to match the certificate and key created in the previous section.
- Set **AWS_BROKER_ADDRESS** to the correct AWS URL. It can be retrieved from AWS IoT **Manage** > **Things** > **Interact** in the HTTPS **Rest API Endpoint** URL.
- Set **CLIENT_ID** to the desired Client ID for your AWS device
- (Optional) if required, change **CACERTFILE** to match the one to be used.

3.6.3.4 Device setup

The application requires the certificates (provided in sample app **certs** subfolder) to be stored in **/mod/ssl_certs/** folder. It can be created with

```
AT#M2MMKDIR=/mod/ssl_certs
```

Certificates can then be loaded with

```
AT#M2MWRITE="/mod/ssl_certs/preload_CACert_01.crt",1468 AT#M2MWRITE="/mod/ssl_certs
IoT.crt",1646
```

providing the file content in RAW mode (for example using the “Transfer Data” button in Telit AT Controller)

For client certificates, the commands will be

```
AT#M2MWRITE="/mod/ssl_certs/xxxxx.crt",yyyy
AT#M2MWRITE="/mod/ssl_certs/xxxxx.key",zzzz
```

PLEASE NOTE: always verify the file sizes to be used in the commands above as they might change

```
Starting AWS IoT Core MQTT demo app. This is v1.1.5 built on Apr 30 2021 09:05:17.
[DEBUG] 15.51  aws_demo:607 - AWS_Task{MQTT_TASK}$ Init MQTT client for AWS
[DEBUG] 15.52  aws_demo:265 - PrepareSSLEnvironment{MQTT_TASK}$ m2mb_ssl_config SNI succeeded
[DEBUG] 15.52  aws_demo:271 - PrepareSSLEnvironment{MQTT_TASK}$ Root CA cert file /mod/ssl_certs/preload_CACert_01.crt
[DEBUG] 15.52  aws_demo:297 - PrepareSSLEnvironment{MQTT_TASK}$ Buffer successfully received from file. 1468 bytes were loaded.
[DEBUG] 15.52  aws_demo:308 - PrepareSSLEnvironment{MQTT_TASK}$ Cross Signed CA cert file /mod/ssl_certs/Amazon-IoT.crt
[DEBUG] 15.52  aws_demo:334 - PrepareSSLEnvironment{MQTT_TASK}$ Buffer successfully received from file. 1646 bytes were loaded.
[DEBUG] 15.52  aws_demo:360 - PrepareSSLEnvironment{MQTT_TASK}$ Client certificate file /mod/ssl_certs/ab71          -certificate.pem.crt
[DEBUG] 15.52  aws_demo:384 - PrepareSSLEnvironment{MQTT_TASK}$ Buffer successfully received from file. 1224 bytes were loaded.
[DEBUG] 15.52  aws_demo:396 - PrepareSSLEnvironment{MQTT_TASK}$ Client Key file /mod/ssl_certs/ab71          -private.pem.key
[DEBUG] 15.52  aws_demo:422 - PrepareSSLEnvironment{MQTT_TASK}$ Buffer successfully received from file. 1679 bytes were loaded.

SSL environment preparation completed
[DEBUG] 15.52  aws_demo:726 - AWS_Task{MQTT_TASK}$ Waiting for registration...
[DEBUG] 15.52  aws_demo:514 - NetCallback{pubTspt_0}$ Module is registered
[DEBUG] 15.52  aws_demo:738 - AWS_Task{MQTT_TASK}$ PDP context initialization
[DEBUG] 17.55  aws_demo:753 - AWS_Task{MQTT_TASK}$ Activate PDP with APN web.omnitel.it on CID 1....
[DEBUG] 18.37  aws_demo:557 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 18.37  aws_demo:561 - PdpCallback{pubTspt_0}$ IP address: 109.114.102.21

Connecting to Server <angy83rl5oizs-ats.iot.eu-west-2.amazonaws.com>:8883...
Done.
[DEBUG] 27.87  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 2, "data": {"RSSI": -72, "tm":"2021-04-30,09:05:50"} }> to topic
device/35308109C          /updates
[DEBUG] 27.87  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 30.94  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 3, "data": {"RSSI": -72, "tm":"2021-04-30,09:05:53"} }> to topic
device/353081090          /updates
[DEBUG] 30.94  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 33.99  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 4, "data": {"RSSI": -72, "tm":"2021-04-30,09:05:56"} }> to topic
device/35308109C          /updates
[DEBUG] 34.00  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 37.00  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 5, "data": {"RSSI": -72, "tm":"2021-04-30,09:05:59"} }> to topic
device/35308109           /updates
[DEBUG] 37.00  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 40.03  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 6, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:02"} }> to topic
device/35308109           /updates
[DEBUG] 40.03  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 43.05  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 7, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:05"} }> to topic
device/353081             /updates
[DEBUG] 43.05  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 46.13  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 8, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:08"} }> to topic
device/3530810            /updates
[DEBUG] 46.13  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 49.15  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 9, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:11"} }> to topic
device/3530810            /updates
[DEBUG] 49.15  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 52.19  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 10, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:14"} }> to topic
device/3530810            /updates
[DEBUG] 52.19  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 55.22  aws_demo:852 - AWS_Task{MQTT_TASK}$ PUBLISHING <{ "ID": 11, "data": {"RSSI": -72, "tm":"2021-04-30,09:06:17"} }> to topic
device/3530810            /updates
[DEBUG] 55.22  aws_demo:857 - AWS_Task{MQTT_TASK}$ Done.

Disconnecting from MQTT broker..
[DEBUG] 58.27  aws_demo:878 - AWS_Task{MQTT_TASK}$ Done.
[DEBUG] 58.27  aws_demo:908 - AWS_Task{MQTT_TASK}$ application exit
[DEBUG] 58.27  aws_demo:918 - AWS_Task{MQTT_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 58.27  aws_demo:924 - AWS_Task{MQTT_TASK}$ Application complete.
```

Figure 192

Data received from a subscriber:

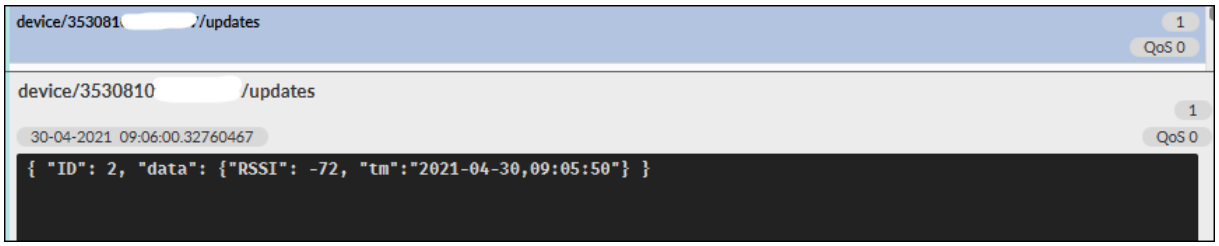

Figure 193

3.6.4 App Manager

Sample application showing how to manage AppZone apps from m2mb code. Debug prints on **MAIN UART**

Features

- How to get how many configured apps are available
- How to get the handle to manage the running app (change start delay, enable/disable)
- How to create the handle for a new binary app, enable it and set its parameters
- How to start the new app without rebooting the device, then stop it after a while.

3.6.4.1 Prerequisites

This app will try to manage another app called “second.bin”, which already exists in the module filesystem and can be anything (e.g. another sample app as GPIO toggle). the app must be built using the flag ROM_START=

in the Makefile to set a different starting address than the main app (by default, 0x40000000). For example, 0x41000000.

3.6.4.2 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- get a non existing app handle and verify it is NULL
- get the current app handle, then get the start delay **set in the INI file (so persistent)**
- change the current app delay value **in the INI file**
- verify that the change has been stored
- get current app state
- create an handle for a second application binary.
- add it to the INI file
- set its execution flag to 0
- get the delay time and the state from INI file for the new app
- get the current set address for the new app
- set the app delay **in RAM, INI will not be affected**.
- start the new app without reboot, using the right set delay
- wait some time, then get the app state and the used RAM amount
- wait 10 seconds, then stop the second app.
- set its execution flag to 1 so it will run at next boot.

```
Starting App Manager demo app. This is v1.0.14-C1 built on Sep 24 2020 12:33:25.
There are 2 configured apps.
Not existing app handle test (should be 0): 0x0
Manager app handle: 0x809e20e0
Manager app delay from nv memory: 5 seconds

Changing Manager app delay time (on non volatile configuration) to 5 seconds..
Manager app delay from nv memory is now 5 seconds
Manager app state is M2MB_APPMNG_STATE_RUN

Trying to get Second app handle...
Second app handle is valid
2nd app delay from nv memory is 1
2nd app current state is M2MB_APPMNG_STATE_READY
Second app current address is 0x41000000
Setting volatile Second app delay (not stored in nvm) to 0 seconds...
Starting Second app on the fly (without reboot)...
Waiting 2 seconds...
2nd app current state is M2MB_APPMNG_STATE_RUN
Second app is running!
Second App is using 475136 bytes of RAM
Stopping Second app now...
wait 10 seconds...
2nd app current state is M2MB_APPMNG_STATE_STOP
Set permanent run permission for Second app.
Done. Second App will also run from next boot-up
```

Figure 194

3.6.5 App update OTA via FTP

Sample application showcasing Application OTA over FTP with AZX FTP. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to connect to a FTP server
- How to download an application binary and update the local version

The app uses a predefined set of parameters. To load custom parameters, upload the `ota_config.txt` file (provided in project's `/src` folder) in module's `/mod` folder, for example with

```
AT#M2MWRITE="/mod/ota_config.txt",<filesize>
```

3.6.5.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage app OTA and start it

`ftp_utils.c`

- Set parameters to default
- Try to load parameters from `ota_config.txt` file
- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Initialize FTP client
- Connect to FTP server and log in
- Get new App binary file size on remote server
- Download the file in `/mod` folder, with the provided name
- Close FTP connection
- Disable PDP context
- Update applications configuration in **app_utils.c**

`app_utils.c`

- Set new application as default
- Delete old app binary
- Restart module

```
Starting FTP APP OTA demo app. This is v1.0.7 built on Apr  7 2020 17:04:05.
[DEBUG] 21.23  ftp_utils.c:447 - msgFTPTask{FTPOTA_TASK}$ INIT
[DEBUG] 21.25  ftp_utils.c:152 - readConfigFromFile{FTPOTA_TASK}$ Reading parameters from file
[DEBUG] 21.26  ftp_utils.c:154 - readConfigFromFile{FTPOTA_TASK}$ Opening /mod/ota_config.txt in read mode..
Set APN to: <<web.omnitel.it>>
Set FTP URL to: <<ftp.telit.com>>
Set FTP PORT to: 21
Set FTP USER to: <<[illegible]>>
Set FTP PASS to: <<[illegible]>>
Set FTP FILE URI to: <</samples/APP_OTA/helloworld.bin>>
Set LOCAL FINAL APP NAME to: <<helloworld.bin>>
Set LOCAL ORIGINAL APP NAME to: <<m2mapz.bin>>
[DEBUG] 23.53  ftp_utils.c:464 - msgFTPTask{FTPOTA_TASK}$ m2mb_os_ev_init success
[DEBUG] 23.54  ftp_utils.c:470 - msgFTPTask{FTPOTA_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.55  ftp_utils.c:478 - msgFTPTask{FTPOTA_TASK}$ Waiting for registration...
[DEBUG] 23.56  ftp_utils.c:371 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 23.56  ftp_utils.c:491 - msgFTPTask{FTPOTA_TASK}$ Pdp context activation
[DEBUG] 23.57  ftp_utils.c:495 - msgFTPTask{FTPOTA_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 25.61  ftp_utils.c:504 - msgFTPTask{FTPOTA_TASK}$ Activate PDP with APN web.omnitel.it on cid 3....
[DEBUG] 26.30  ftp_utils.c:398 - PdpCallback{pubTspt_0}$ Context active
[DEBUG] 26.30  ftp_utils.c:401 - PdpCallback{pubTspt_0}$ IP address: 176.246.110.148
Start ftp client...
[DEBUG] 27.36  ftp_utils.c:533 - msgFTPTask{FTPOTA_TASK}$ Connected.
[DEBUG] 28.87  ftp_utils.c:546 - msgFTPTask{FTPOTA_TASK}$ FTP login successful.
Get remote file /samples/APP_OTA/helloworld.bin size
[DEBUG] 29.31  ftp_utils.c:568 - msgFTPTask{FTPOTA_TASK}$ Done. File size: 116224.
Starting download of remote file /samples/APP_OTA/helloworld.bin into local /mod/helloworld.bin
/samples/APP_OTA/helloworld.bin  4.68% 5440
/samples/APP_OTA/helloworld.bin  9.36% 10880
/samples/APP_OTA/helloworld.bin 14.04% 16320
/samples/APP_OTA/helloworld.bin 18.72% 21760
/samples/APP_OTA/helloworld.bin 23.40% 27200
/samples/APP_OTA/helloworld.bin 28.08% 32640
/samples/APP_OTA/helloworld.bin 32.76% 38080
/samples/APP_OTA/helloworld.bin 37.44% 43520
/samples/APP_OTA/helloworld.bin 42.13% 48960
/samples/APP_OTA/helloworld.bin 46.81% 54400
/samples/APP_OTA/helloworld.bin 51.49% 59840
/samples/APP_OTA/helloworld.bin 56.17% 65280
/samples/APP_OTA/helloworld.bin 60.85% 70720
/samples/APP_OTA/helloworld.bin 65.53% 76160
/samples/APP_OTA/helloworld.bin 70.21% 81600
/samples/APP_OTA/helloworld.bin 74.89% 87040
/samples/APP_OTA/helloworld.bin 79.57% 92480
/samples/APP_OTA/helloworld.bin 84.25% 97920
/samples/APP_OTA/helloworld.bin 88.93% 103360
/samples/APP_OTA/helloworld.bin 93.61% 108800
/samples/APP_OTA/helloworld.bin 97.42% 113220
[DEBUG] 43.54  ftp_utils.c:608 - msgFTPTask{FTPOTA_TASK}$ download successful.
FTP quit...
[DEBUG] 43.77  ftp_utils.c:632 - msgFTPTask{FTPOTA_TASK}$ Deactivating PDP
[DEBUG] 43.77  ftp_utils.c:642 - msgFTPTask{FTPOTA_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 44.20  ftp_utils.c:407 - PdpCallback{pubTspt_0}$ Context deactive
[DEBUG] 45.44  app_utils.c:76 - update_app{FTPOTA_TASK}$ Application successfully configured.
[DEBUG] 45.45  app_utils.c:82 - update_app{FTPOTA_TASK}$ Deleting old application /mod/m2mapz.bin
€ÿStarting. This is v1.0.7 built on Apr  7 2020 17:02:52. LEVEL: 2

 Start Hello world Application  [ version: 2.000000 ]

 Hello world 2.0 [ 000001 ]
 Hello world 2.0 [ 000002 ]
 Hello world 2.0 [ 000003 ]
```

Figure 195

3.6.6 CJSON example:

Sample application showcasing how to manage JSON objects. Debug prints on **MAIN UART**

Features

- How to read a JSON using cJSON library
- How to write a JSON
- How to manipulate JSON objects

3.6.6.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Parse an example string into a JSON object and print the result in a formatted string
- Print some test outcomes (e.g. non existing item correctly not found)
- Retrieve single elements from the parsed JSON object and use them to format a descriptive string
- Delete the JSON object
- Create a new JSON object appending elements to it
- Print the result JSON string from the object

```
Starting Logging demo app. This is v1.0.7 built on Apr  7 2020 08:33:03.
And here is what we got:
{
	"name":	"Atlantic Ocean",
	"format":	{
		"type":	"salt",
		"volume":	310410900,
		"depth":	-8486,
		"volume_percent":	23.300000,
		"tide":	-3.500000,
		"calm":	false,
		"life":	["plankton", "corals", "fish", "mammals"]
	}
}
inexistent key not found
name found: Atlantic Ocean
format found (null)
Our JSON string contains info about an ocean named Atlantic Ocean,
has a volume of 310410900 km^3 of salt water with -8486 meters max depth,
represents 23.3% of total oceans volume,
has an average low tide of -3.5 meters,
hosts a huge number of living creatures such as plankton, corals, fish, mammals,
and is not always calm.

Let's build a TR50 command with a proprety.publish and an alarm.publish for MQTT (no auth).
And here is what we got:
{
	"1":	{
		"command":	"property.publish",
		"params":	{
			"thingKey":	"mything",
			"key": "mykey",
			"value":	123.144000
		}
	},
	"2":	{
		"command":	"alarm.publish",
		"params":	{
			"thingKey":	"mything",
			"key": "mykey",
			"state":	3,
			"msg": "Message."
		}
	}
}
END.
```

Figure 196

3.6.7 Crypto Elliptic Curve Cryptography (ECC) example

Sample application showcasing how to manage Elliptic Curve Cryptography functionalities. Debug prints on **MAIN UART**

Features

- How to initialize ECC contexts A (Alice) and B (Bob). Alice is emulating a remote host, from which a public key is known.
- How to generate keypairs for contexts and export public keys
- how to export keyblobs from a context (a keyblob is encrypted with hw specific keys, and can only be used on the module where it was created)
- How to save a keyblob in secured TrustZone.
- How to reload a keyblob from the TrustZone into an initialized context
- How to sign a message with ECDSA from context B (Bob) and verify it from another context A (Alice) with the signature and public key of Bob.
- How to make Bob and Alice derive a shared session keys using each other's public key.
- How to make Bob and Alice create an AES context with the newly created shared keys, encode data and decode it on the other side

3.6.7.1 Application workflow

`M2MB_main.c`

- Create Bob ECC context, create a keypair and export it in a keyblob
- Open a file in secured Trust Zone, then store the keyblob in it.
- Destroy Bob ECC context
- Recreate Bob ECC context, open the file from Trust Zone and read the keyblob.
- Import the keyblob in Bob context.
- Export Bob public key
- Create Alice ECC context, to simulate an external host. Generate a keypair and export the public key.
- Sign a message with Bob context, generating a signature.
- Use Alice to verify the signed message using Bob's signature and public key
- Derive a shared key for Bob, using Alice's public key
- Create an AES context for Bob
- Import the shared key into the AES context
- Encrypt a message using Bob's AES context.

- Derive a shared key for Alice, using Bob's public key
- Create an AES context for Alice
- Import the shared key into the AES context
- Decrypt the message using Alice's AES context.
- Check the decrypted message and the original one match
- Clear all resources

```
Starting Crypto ECC demo app. This is v1.0.9-C1 built on May 11 2020 16:30:23.

Bob (local) and Alice (remote) scenario
Bob's keypair generated
Bob's keyblob length is 224
Bob exported the keyblob to be securely stored.

Bob already had an item in Secure Data Area, it was removed to create a new one
Bob securely saved the keyblob in Secure Data Area
Releasing resources

Close Bob's context...
Done. Now Bob context does not exist anymore.

Re-initialize Bob Context and load the keyblob from the secure zone
Bob securely loaded the keyblob from the SDA
Import keyblob in Bob's context..
Done. Now export Bob's public key...
Bob's public key successfully exported

Alice's keypair generated
Alice's public key successfully exported

Bob's message signed with ECDSA!
Alice verified bob's message with his pubkey and signature!

------------------------------------------------------------
Bob and Alice will now exchange a message with AES encrypt
------------------------------------------------------------

Bob retrieved the generated shared key size
Bob's shared keyblob length is: 32. Allocate the required memory to store it.
Bob created a shared key using Alice's public key!

Bob created an AEX context to exchange encrypted data with Alice
Bob's AES context imported the shared keyblob
Bob Encrypted the message using AES and the shared key!
Encrypted data:
 94EE531E3B84B2A4EF05502186BFF5DA

Alice retrieved the generated shared key size
Alice's shared keyblob length is: 32. Allocate the required memory to store it.
Alice created a shared key using Bob's public key!

Alice created an AEX context to exchange encrypted data with Bob
Alice's AES context imported the shared keyblob
Alice decrypted the message using AES and the shared key!
Decrypted:
 414094941E8942A4445548035BFAE943

Original, plain message:
 414094941E8942A4445548035BFAE943

Plain and decrypted messages match!
```

Figure 197

3.6.8 EEPROM 24AA256

Sample application showing how to communicate with a MicroChip 24AA256T I2C EEPROM chip using azx eeprom utility APIs. Debug prints on **MAIN UART**

Setup

This demo application requires that: - A0, A1, and A2 pins (1,2,3 chip pins) are connected to ground (pin 4) for device address 0xA0 - Pin 7 (WP) is connected to ground - Pin 6 (SCL) is connected to module GPIO 3 - Pin 5 (SDA) is connected to module GPIO 2 - Pin 4 is connected to one of the ground pins of the module - Pin 8 is connected to 1v8 supply (e.g. VPWRMON pin on the module)

Features

- Initialize the logs on the output channel
- configure the EEPROM utility, setting the slave address and the memory parameters (page size, memory size)
- Write single bytes on a random address
- Read written bytes as a page
- Write data using pages
- Read the new data using pages
- Read again using sequential reading
- Read a single byte from a specific address
- Read next byte using read from current address
- Erase the EEPROM
- Deinit EEPROM utility

3.6.8.1 Application workflow

`M2MB_main.c`

- call azx_eeprom_init() to set the utility parameters (SDA and SCL pins, page and memory sizes)
- call azx_eeprom_writeByte() to store a single byte with value “5” at the address 0x0213
- call azx_eeprom_writeByte() to store a single byte with value “6” at the address 0x0214
- call azx_eeprom_readPages() from address 0x0213 to retrieve the 2 bytes from the EEPROM
- call azx_eeprom_writePages to write 1024 bytes from a buffer, starting from address 0x00
- call azx_eeprom_readPages() again, to read 256 bytes from address 0x00
- call azx_eeprom_readSequentially() to read 256 bytes from 0x00 by without pages (less overhead on I2C protocol)

- call azx_eeprom_readByte() to get a single byte from address 0x00
- call azx_eeprom_readByteFromCurrentAddress() to get a byte from next address (0x01)
- call azx_eeprom_eraseAll() to completely erase the EEPROM memory (this writes 0xFF in each byte)
- call azx_eeprom_readPages from address 0x0213 to get 2 bytes and verify the values have been written to 0xFF
- call azx_eeprom_deinit to close the eeprom handler and the I2C channel

```
Starting I2C EEPROM 24AA256T demo app. This is v1.0.13-C1 built on Nov  3 2020 16:28:23.

Configuring the I2C device...
Opening I2C channel /dev/I2C-160 ( device address is 0xA0 )
Writing 1 byte at address 0x0213...
Done.
Writing 1 byte at address 0x0214...
Done.
Reading the 2 bytes from address 0x0213...
Done. Data: [0xFF 0xFF]

Writing 1024 bytes at address 0x0000..
Done.

Reading 256 bytes from address 0x0000...
Done. Data:
<<ABCDEFGHIJKLMNOPQRSTUVWXYZ.............................................abcdefghijklmnopqrstuvwxyz................

Reading 256 bytes sequentially from address 0x0000...
Done. Data:
<<ABCDEFGHIJKLMNOPQRSTUVWXYZ.............................................abcdefghijklmnopqrstuvwxyz................


Reading 1 byte from address 0x0000...
Done. Data: 'A'

Reading 1 byte from current address (should be 0x0001)...
Done. Data: 'B'


[DEBUG] 17.47  M2MB_main:177 - run_I2C_EEPROM_Demo{M2M_DamsStart}$ Erasing all the eeprom...
[DEBUG] 28.05  M2MB_main:185 - run_I2C_EEPROM_Demo{M2M_DamsStart}$ Done

Reading the 2 bytes from address 0x0213...
Done. Data: [0xFF 0xFF]

Deinit EEPROM...
Done
```

Figure 198

3.6.9 Easy AT example

Sample application showcasing Easy AT functionalities. Debug prints on **MAIN UART**

Features

- Shows how to register custom commands

The application adds two custom commands to the list of available ones:

- AT#MYCMD
- AT#MYINPUT

3.6.9.1 AT#MYCMD

This is a simple parameter-waiting command. It expects one string parameter, and will print it on the logging interface once received. The command simply returns OK

3.6.9.2 AT#MYINPUT

This command expects a numeric parameter, which indicates how many bytes will be received over the interface at most (the command will provide a prompt indicating it is waiting data). Then the data management callback will print when data is received, and if CTRL+Z (0x1A in hex) is received, it will complete the process, printing in the log interface what was received. sending ESC will terminate the process discarding any pending data.

3.6.10 Events

Sample application showcasing events setup and usage. Debug prints on **MAIN UART**

Features

- How to setup OS events with a custom bitmask
- How to wait for events and generate them in callback functions to synchronize blocks of code

3.6.10.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Create an event handler
- Create a timer to generate an event, with a 2 seconds expiration time
- Wait for a specific event bit on the event handler
- At timer expiration, set the same event bit and verify that the code flow went through after the event.

```
Starting Events demo app. This is v1.0.7 built on Apr  7 2020 08:44:29.
[DEBUG] 20.55  M2MB_main.c:171 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
Set the timer attributes structure success.
Timer successfully created
[DEBUG] 20.57  M2MB_main.c:125 - setup_timer{M2M_DamsStart}$ Start the timer, success.
[DEBUG] 22.60  M2MB_main.c:60 - hwTimerCb{pubTspt_0}$ Timer Callback, generate event!
[DEBUG] 22.61  M2MB_main.c:183 - M2MB_main{M2M_DamsStart}$ event occurred!
```

Figure 199

3.6.11 Events - Barrier (multi events)

Sample application showcasing how to setup and use multiple events to create a barrier. Debug prints on **MAIN UART**

Features

- How to setup OS events to be used as a barrier
- How to wait for multiple events in the same point, and generate them in call-back functions to synchronize blocks of code

3.6.11.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create an event handler
- Create a timer to generate an event, with a 3 seconds expiration time
- Create another timer to generate an event, with a 6 seconds expiration time
- Start both timers
- Wait for both event bits on the event handler (each one will be set by one of the timers)
- At first timer expiration, set the first event bit and verify that the code flow does not procede.
- At second timer expiration, set the second event bit and verify that the code flow went through after the event (implementing a barrier).

```
Starting Barrier demo app. This is v1.0.7 built on Apr  7 2020 08:48:30.
[DEBUG] 20.01  M2MB_main.c:179 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
Set the timer attributes structure success.
Timer successfully created with 3000 timeout (ms)
Set the timer attributes structure success.
Timer successfully created with 6000 timeout (ms)
[DEBUG] 23.08  M2MB_main.c:66 - hwTimerCb1{pubTspt_0}$ Timer Callback, generate event 1!
[DEBUG] 26.12  M2MB_main.c:75 - hwTimerCb2{pubTspt_0}$ Timer Callback, generate event 2!
[DEBUG] 26.13  M2MB_main.c:214 - M2MB_main{M2M_DamsStart}$ BOTH events occurred!
```

Figure 200

3.6.12 FOTA example

Sample application showcasing FOTA usage with M2MB API. Debug prints on **MAIN UART**

Features

- How download a delta file from a remote server
- How to apply the delta and update the module firmware

3.6.12.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a main task to manage connectivity.
- create a fota task to manage FOTA and start it with INIT option

`fota.c`

fotaTask()

- Initialize FOTA system then reset parameters.
- Check current FOTA state, if not in IDLE, return error.
- Send a message to mainTask so networking is initialized.
- after PdPCallback() notifies the correct context activation, configure the fota client parameters such as FTP server URL, username and password
- get delta file from server. when it is completed, FOTADownloadCallback is called.
- If delta download went fine, check it.
- If delta file is correct, apply it. Once complete, restart the module.

mainTask()

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context. Event will be received on **PdP-Callback** function
- Disable PDP context when required to stop the app

PdpCallback()

- When PDP context is enabled, send a message to fotaTask to start the download

```
Starting FOTA demo app. This is v1.1.7 built on Jun 11 2021 12:20:43.
[DEBUG] 23.60  fota:187 - fotaTask{FOTA_TASK}$ Init FOTA...

Session file not present, procede with FOTA...
[DEBUG] 23.61  fota:236 - fotaTask{FOTA_TASK}$ m2mb_fota_reset PASS
[DEBUG] 23.61  fota:260 - fotaTask{FOTA_TASK}$ m2mb_fota_state_get  M2MB_FOTA_STATE_IDLE
[DEBUG] 23.62  fota:379 - mainTask{MAIN_TASK}$ INIT
[DEBUG] 23.62  fota:392 - mainTask{MAIN_TASK}$ m2mb_os_ev_init success
[DEBUG] 23.63  fota:398 - mainTask{MAIN_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.63  fota:405 - mainTask{MAIN_TASK}$ Waiting for registration...
[DEBUG] 23.64  fota:131 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 23.65  fota:418 - mainTask{MAIN_TASK}$ Pdp context initialization
[DEBUG] 25.70  fota:431 - mainTask{MAIN_TASK}$ Activate PDP with APN web.omnitel.it on cid 1....
[DEBUG] 35.42  fota:152 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 35.43  fota:155 - PdpCallback{pubTspt_0}$ IP address: 2.41.116.139

[DEBUG] 35.43  fota:285 - fotaTask{FOTA_TASK}$
Trying to download "samples/FOTA/37.00.003.3_to_37.00.003.1_ME310G1_NANVWWAU.bin" delta file...
[DEBUG] 35.45  fota:295 - fotaTask{FOTA_TASK}$ m2mb_fota_get_delta OK - Waiting for the completion callback
[DEBUG] 119.43  fota:96 - FOTADownloadCallBack{pubTspt_0}$ FOTA download Success - performing packet validation...
[DEBUG] 119.44  fota:301 - fotaTask{FOTA_TASK}$ Validating delta file...
[DEBUG] 156.36  fota:317 - fotaTask{FOTA_TASK}$ Packet is valid, start update...
[DEBUG] 156.40  fota:329 - fotaTask{FOTA_TASK}$ m2mb_fota_start PASS
[DEBUG] 158.36  fota:342 - fotaTask{FOTA_TASK}$
Rebooting...After reboot there will be the new FW running on module!

#OTAEV: Module Upgraded To New Fw
Starting FOTA demo app. This is v1.1.7 built on Jun 11 2021 12:20:43.
[DEBUG] 29.24  fota:187 - fotaTask{FOTA_TASK}$ Init FOTA...

Session file is already present, stop.
```

Figure 201

3.6.13 FOTA from Local File example

Sample application that shows how perform FOTA upgrade using a delta file stored into file system. Debug prints on **MAIN UART**

Features

- How to store and get FOTA upgrade information to/from a file
- How to get delta file from module file system
- How to apply the delta and update module firmware

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Check if module has been already upgraded or needs to be upgraded reading FOTA upgrade status from a file
- Create a fota task to manage FOTA and start it with INIT option

smartFotaTask() - Initialize FOTA system then reset parameters. - Get FOTA partiton size and block size - Copy delta file from file system to FOTA paartition. when it is completed, FOTADownloadCallback is called. - If delta file is correct, apply it. Once complete, write FOTA status flag and current fw version to a file, restart the module.

```
Starting FOTA application from local delta file demo app. This is v1.0.14-C1 built on May 16 2022 15:50:38.


Check FOTA upgrade status in file system

Module current fw version is:
37.00.214-P0C.210001
M0C.200003
P0C.210001
A0C.210000
[WARN ] 13.83  M2MB_main:190 - FOTAUpgradeStatusInit{M2M_DamsStart}$ File doesn't exist create it, first app execution
File created, store current fw version and fota upgrade flag=0

Read stored data

FOTA Flag: 0 Module previous fw version:
37.00.214-P0C.210001
M0C.200003
P0C.210001
A0C.210000

Start FOTA process
[DEBUG] 15.42  M2MB_main:450 - smartFotaTask{FOTA_TASK}$ INIT
[DEBUG] 15.43  M2MB_main:461 - smartFotaTask{FOTA_TASK}$ m2mb_fota_init success
[DEBUG] 15.44  M2MB_main:464 - smartFotaTask{FOTA_TASK}$
Get block and FOTA partition size
OTA blockSize: 131072
OTA partitionSize: 11272192
[DEBUG] 15.47  M2MB_main:472 - smartFotaTask{FOTA_TASK}$
 Copy delta file from File system to FOTA partition
[DEBUG] 15.47  M2MB_main:277 - copyFromFileSystemToDelta{FOTA_TASK}$ File size: 746552
[DEBUG] 15.87  M2MB_main:480 - smartFotaTask{FOTA_TASK}$ Delta file writing completed
[DEBUG] 15.88  M2MB_main:484 - smartFotaTask{FOTA_TASK}$
Delta file check...
[DEBUG] 15.89  M2MB_main:347 - check_fota_delta{FOTA_TASK}$ -- check_fota_delta...
[DEBUG] 54.73  M2MB_main:366 - check_fota_delta{FOTA_TASK}$ Fota check integrity PASS
[DEBUG] 54.74  M2MB_main:492 - smartFotaTask{FOTA_TASK}$ ...delta file OK
[DEBUG] 54.75  M2MB_main:496 - smartFotaTask{FOTA_TASK}$
--> Start update...
[DEBUG] 56.83  M2MB_main:403 - startUpdate{FOTA_TASK}$
Reboot module to start delta deployment
```

Figure 202

3.6.14 FTP

Sample application showcasing FTP client demo with AZX FTP. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to connect to a FTP server
- How to exchange data with the server

3.6.14.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage FTP client and start it

`ftp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init FTP client and set the debug function for it
- Connect to the server
- Perform log in
- Check remote file size and last modification time
- Download file from server to local filesystem. A data callback is set to report periodic info about the download status
- Upload the same file to the server with a different name. A data callback is set to report periodic info about the upload status
- Download another file content in a buffer instead of a file. A data callback is set to report periodic info about the download status
- Close the connection with FTP server
- Disable PDP context

```
Starting FTP demo app. This is v1.0.7 built on Apr  7 2020 11:17:36.
[DEBUG] 21.23  ftp_test.c:290 - msgFTPTask{FTP_TASK}$ INIT
[DEBUG] 21.23  ftp_test.c:304 - msgFTPTask{FTP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.23  ftp_test.c:310 - msgFTPTask{FTP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.23  ftp_test.c:318 - msgFTPTask{FTP_TASK}$ Waiting for registration...
[DEBUG] 21.25  ftp_test.c:214 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 21.26  ftp_test.c:331 - msgFTPTask{FTP_TASK}$ Pdp context activation
[DEBUG] 21.27  ftp_test.c:335 - msgFTPTask{FTP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.31  ftp_test.c:344 - msgFTPTask{FTP_TASK}$ Activate PDP with APN web.omnitel.it on cid 3....
[DEBUG] 24.09  ftp_test.c:241 - PdpCallback{pubTspt_0}$ Context active
[DEBUG] 24.10  ftp_test.c:244 - PdpCallback{pubTspt_0}$ IP address: 176.244.166.181
Start ftp client...
[DEBUG] 24.82  ftp_test.c:373 - msgFTPTask{FTP_TASK}$ Connected.
[DEBUG] 26.32  ftp_test.c:386 - msgFTPTask{FTP_TASK}$ FTP login successful.
Get remote file /samples/pattern_big.txt size
[DEBUG] 26.69  ftp_test.c:428 - msgFTPTask{FTP_TASK}$ Done. File size: 20026.
Get remote file /samples/pattern_big.txt last modification date
[DEBUG] 26.89  ftp_test.c:450 - msgFTPTask{FTP_TASK}$ Done. File last mod date: 20200407090654
.

Starting download of remote file /samples/pattern_big.txt into local /mod/_pattern_big.txt
/samples/pattern_big.txt 47.54% 9520
/samples/pattern_big.txt 100.00% 20026
[DEBUG] 29.75  ftp_test.c:488 - msgFTPTask{FTP_TASK}$ download successful.
[DEBUG] 29.76  ftp_test.c:522 - msgFTPTask{FTP_TASK}$
Local file /mod/_pattern_big.txt size: 20026

Starting upload of local file /mod/_pattern_big.txt
/mod/_pattern_big.txt 81.81% 16384
Upload successful.

Starting download of remote file /samples/pattern.txt into local buffer
Getting remote file /samples/pattern.txt size..
[DEBUG] 32.97  ftp_test.c:583 - msgFTPTask{FTP_TASK}$ Done. File size: 988.
Starting download of remote file /samples/pattern.txt to buffer
[DEBUG] 34.08  ftp_test.c:145 - buf_data_cb{FTP_TASK}$ Received START event
[DEBUG] 34.09  ftp_test.c:149 - buf_data_cb{FTP_TASK}$ Received DATA: 988 bytes on buffer 0x400399e0
[DEBUG] 34.26  ftp_test.c:153 - buf_data_cb{FTP_TASK}$ Received END event
[DEBUG] 34.26  ftp_test.c:623 - msgFTPTask{FTP_TASK}$ Download successful. Received 988 bytes<<<
0   |---------|   |---------|   |---------|   |---------|   |---------|  *
1   |    A    |   |    A    |   |    A    |   |    A    |   |    A    |   *
2   |   AAA   |   |   AAA   |   |   AAA   |   |   AAA   |   |   AAA   |    *
3   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |     *
4   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |      *
5   |AAAAAAAAA|   |AAAAAAAAA|   |AAAAAAAAA|   |AAAAAAAAA|   |AAAAAAAAA|       *
6   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |   | AAAAAAA |      *
7   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |   |  AAAAA  |     *
8   |   AAA   |   |   AAA   |   |   AAA   |   |   AAA   |   |   AAA   |    *
9   |    A    |   |    A    |   |    A    |   |    A    |   |    A    |   *
10  |---------|   |---------|   |---------|   |---------|   |---------|  *
11                                                                      *
12  |---------|   |---------|   |---------|   |---------|   |--->>>
```

Figure 203

3.6.15 File System example

Sample application showcasing M2MB File system API usage. Debug prints on **MAIN UART**

Features

- How to open a file in write mode and write data in it
- How to reopen the file in read mode and read data from it

3.6.15.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Print welcome message
- Open file in write mode
- Write data in file
- Close file
- Reopen file in read mode
- Read data from file and print it
- Close file and delete it

```
Starting FileSystem demo app. This is v1.0.7 build on Mar 26 2020 09:50:19. LEVEL: 2
Opening /my_text_file.txt in write mode..
Buffer written successfully into file. 15 bytes were written.
Closing file.
Opening /my_text_file.txt in read only mode..
Received 15 bytes from file:
<Hello from file>
Closing file.
Deleting File
File deleted
App Completed
```

Figure 204

3.6.16 GNSS example

Sample application showing how to use GNSS functionality. Debug prints on **MAIN UART**

Features

- How to enable GNSS receiver on module
- How to collect location information from receiver

Note: on MEx10G1 product family both M2MB_GNSS_SERVICE_NMEA_REPORT and M2MB_GNSS_SERVICE_POSITION_REPORT services are available, while on ME910C1 product family only M2MB_GNSS_SERVICE_POSITION_REPORT is available

3.6.16.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print a welcome message
- Create GNSS task and send a message to it

`gps_task.c` - Init Info feature and get module type - Init gnss, enable position/NMEA report and start it. - When a fix or a NMEA sentence is available, a message will be printed by the GNSS callback function

```
Starting GNSS demo app. This is v1.1.4 built on Oct  1 2021 15:27:44.

Model: ME910C1-E2

m2mb_gnss_enable, POSITION OK

m2mb_gnss_start OK, waiting for position/nmea sentences...
latitude_valid: 1 - latitude: 45.713643
longitude_valid: 1 - longitude: 13.738041
altitude_valid: 1 - altitude: 195.000000
uncertainty_valid: 1 - uncertainty: 95.000000
velocity_valid: 1 - codingType: 0
speed_horizontal: 0.650000
bearing: 0.000000
timestamp_valid: 1 -timestamp: 1633095357439
speed_valid: 1 - speed: 1.471360


***** Wait 120 seconds and then stop GPS *****
```

Figure 205

3.6.17 GPIO interrupt example

Sample application showing how to use GPIOs and interrupts. Debug prints on **MAIN UART**

Features

- How to open a GPIO in input mode with interrupt
- How to open a second GPIO in output mode to trigger the first one

3.6.17.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Open GPIO 4 as output
- Open GPIO 3 as input and set interrupt for any edge (rising and falling). **A jumper must be used to short GPIO 3 and 4 pins.**
- Toggle GPIO 4 status high and low every second
- An interrupt is generated on GPIO 3

```
Starting GPIO interrupt demo app. This is v1.0.7 built on Mar 26 2020 16:33:01.
Setting gpio 3 interrupt...
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
```

Figure 206

3.6.18 GTP example

Sample application that shows hot to get the position using GTP feature. Debug prints on **MAIN UART**

Features

- How to init and enable GTP feature
- How to get the position using GTP

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Init NET functionality and wait for module to be registered
- Init PDP functionality and set APN on CID1
- Init GTP functionality
- Check if GTP is already enabled. If not enable it and reboot module
- If GTP is enabled, get position

```
Starting GTP demo app. This is v1.1.8 built on Dec 20 2022 15:29:19.

[DEBUG] 15.08  M2MB_main:237 - M2MB_main{M2M_DamsStart}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 15.09  M2MB_main:245 - M2MB_main{M2M_DamsStart}$ Waiting for registration...
[DEBUG] 15.10  M2MB_main:253 - M2MB_main{M2M_DamsStart}$ Pdp context setting
[DEBUG] 15.11  M2MB_main:82 - NetCallback{pubTspt_0}$ Module is registered to cell 0x20!
[DEBUG] 15.11  M2MB_main:257 - M2MB_main{M2M_DamsStart}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 15.13  M2MB_main:268 - M2MB_main{M2M_DamsStart}$ m2mb_pdp_deinit returned M2MB_RESULT_SUCCESS

Check if GTP has been already enabled
[DEBUG] 16.19  M2MB_main:312 - M2MB_main{M2M_DamsStart}$ GTP status: 1 => Enabled
[DEBUG] 16.20  M2MB_main:317 - M2MB_main{M2M_DamsStart}$ Get the position...

m2mb_gnss_GTP OK, waiting for position...
[DEBUG] 107.96  M2MB_main:160 - gnssCallbackFN{pubTspt_0}$ gnssCallback event: 1
GTP position got
latitude: 41.900002
longitude: 12.500000
altitude: 0.000000
altitudeMeanSeaLevel: -0.000000
accuracy: 0.000000
speed: -0.000000
bearing: -0.000000
timestamp: 0
verticalAccuracy 0.000000
speedAccuracy -0.000000
bearingAccuracy: -0.000000
```

Figure 207

3.6.19 General_INFO example

Sample application prints some Module/SIM information as IMEI, fw version, IMSI and so on; it prints also some information about registration. Debug prints on **MAIN UART**

Features

- How to print some Module information as IMEI, FW version etc
- How to print some SIM information as IMSI, ICCID
- How to get and print some informatio about Module registration as Netowrk Operator, AcT, RSSI, etc

3.6.19.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Init NET functionality
- Init INFO functionality
- Get and print Module and SIM info
- Wait form module to register to network
- Get and print registration INFO

```
Starting. This is v1.1.4 built on Mar 31 2021 09:56:03. LEVEL: 2


 Start General INFO application [ version: 1.000000 ]


=======================

MODULE ME910C1-E2 INFO

=======================

MANUFACTURER: Telit

IMEI: 353080091125422

MODEM FIRMWARE VERSION: M0B.700005

PACKAGE VERSION:
30.00.709-B005-P0B.700100
M0B.700005
P0B.700100
A0B.700000


========

SIM INFO

========

IMSI: 222015602268648

ICCID: 89390100001138084906


============================

Waiting for registration...

============================

Module is registered to HOME network cellID 0x5221
NETWORK OPERATOR (mcc mnc): 222 01
Network Technology 2G (AcT: 0) RSSI: -81
```

Figure 208

3.6.20 HTTP Client

Sample application showing how to use HTTPs client functionalities. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to initialize the http client, set the debug hook function and the data callback to manage incoming data
- How to perform GET, HEAD or POST operations (GET also with single range support)

NOTE: the sample app has an optional dependency on azx_base64.h if basic authentication is required (refer to `HTTP_BASIC_AUTH_GET` define in `M2MB_main.c` for further details)

3.6.20.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage HTTP client and start it

`httpTaskCB`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create HTTP client options and initialize its functionality
- Create HTTP SSL config and initialize the SSL options
- Configure data management options for HTTP client
- Appy all configurations to HTTP client
- Perform a GET request to a server
- Disable PDP context

`DATA_CB`

- Print incoming data
- Set the abort flag to 0 to keep going.

```
Starting HTTP(s) client demo app. This is v1.1.15 built on Nov 28 2022 12:12:27.
[DEBUG] 17.12  read_parame:101 - readConfigFromFile{HttpClient}$ Reading parameters from file
[DEBUG] 17.13  read_parame:103 - readConfigFromFile{HttpClient}$ Opening /mod/HTTP_Client_config.txt in read mode..
Set APN to: <<web.omnitel.it>>
Set APN USER to: <<>>
Set APN PASS to: <<>>
Set CID to: 3
Set CACERTFILE to: <<>>
Set CLIENTCERTFILE to: <<>>
Set CLIENTKEYFILE to: <<>>
Set REQUEST_TYPE to: 2
Set SERVER to: <<http://linux-ip.net>>
[DEBUG] 17.16  m2mb_HTTP_t:267 - activatePdP{HttpClient}$ m2mb_os_ev_init success
[DEBUG] 17.17  m2mb_HTTP_t:273 - activatePdP{HttpClient}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 17.18  m2mb_HTTP_t:281 - activatePdP{HttpClient}$ Waiting for registration...
[DEBUG] 17.19  m2mb_HTTP_t:101 - NetCallback{pubTspt_0}$ Module is registered to cell 0x468E!
[DEBUG] 17.20  m2mb_HTTP_t:295 - activatePdP{HttpClient}$ Pdp context initialization
[DEBUG] 19.26  m2mb_HTTP_t:305 - activatePdP{HttpClient}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 20.16  m2mb_HTTP_t:128 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 20.18  m2mb_HTTP_t:131 - PdpCallback{pubTspt_0}$ IP address: 176.243.212.114
Performing a GET request...
Connecting to linux-ip.net:80//

Socket connected!

<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="author" content="Martin A. Brown" />
    <meta name="robots" content="index, follow"/>

    <meta property="og:title" content="http://linux-ip.net/"/>
    <meta property="og:url" content="http://linux-ip.net"/>
    <meta property="og:site_name" content="http://linux-ip.net/"/>
    <meta property="og:type" content="website"/>

    <link rel="canonical" href="http://linux-ip.net" />

    <title>http://linux-ip.net/</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" type="text/css" href="//netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.css" />
    <link rel="stylesheet" type="text/css" href="//netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css" />
    <link rel="stylesheet" type="text/css" href="http://linux-ip.net/theme/css/main.css" />

</head>
   ▪▪▪
    <footer id="site-footer">
        <div class="row-fluid">
            <div class="span10 offset1">
                <address>
                    <p>
                        Powered by <a href="http://getpelican.com/">Pelican</a>
                        and <a href="http://python.org">Python</a>.
                        Theme based on <a href="http://github.com/jsliang/pelican-fresh/">Fresh</a>
                        by <a href="http://jsliang.com/">jsliang</a>
                    </p>
                </address>
            </div>
        </div>
    </footer>
</body>
</html>
Result: 200
[DEBUG] 26.80  m2mb_HTTP_t:137 - PdpCallback{pubTspt_0}$ Context deactivated!
```

Figure 209

3.6.21 HW Timer (Hardware Timer)

The sample application shows how to use HW Timers M2MB API. Debug prints on **MAIN UART**

Features

- How to open configure a HW timer
- How to use the timer to manage recurring events

3.6.21.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create hw timer structure
- Configure it with 100 ms timeout, periodic timer (auto fires when expires) and autostart
- Init the timer with the parameters
- Wait 10 seconds
- Stop the timer

`TimerCb`

- Print a message with an increasing counter

```
Starting HW Timers demo app. This is v1.0.7 built on Mar 26 2020 13:04:14.
[DEBUG] 14.06  M2MB_main.c:114 - M2MB_main{M2M_DamsStart}$ Set the timer attributes structure: success.
Timer successfully created
Start the timer, success.
[DEBUG] 14.18  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [0]
[DEBUG] 14.28  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [1]
[DEBUG] 14.38  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [2]
[DEBUG] 14.48  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [3]
[DEBUG] 14.58  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [4]
[DEBUG] 14.69  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [5]
[DEBUG] 14.79  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [6]
[DEBUG] 14.88  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [7]
[DEBUG] 14.98  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [8]
[DEBUG] 15.08  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [9]


[DEBUG] 23.90  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [96]
[DEBUG] 24.01  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [97]
[DEBUG] 24.11  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [98]
Stop a running timer: success
Application end
```

Figure 210

3.6.22 Hello World

The application prints “Hello World!” over selected output every two seconds. Debug prints on **MAIN UART**, using AZX log example functions

Features

- How to open an output channel using AZX LOG sample functions
- How to print logging information on the channel using AZX LOG sample functions

3.6.22.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Print “Hello World!” every 2 seconds in a while loop

```
Starting. This is v1.0.7 built on Mar 26 2020 09:34:16. LEVEL: 2

 Start Hello world Application  [ version: 2.000000 ]

 Hello world 2.0 [ 000001 ]
 Hello world 2.0 [ 000002 ]
 Hello world 2.0 [ 000003 ]
 Hello world 2.0 [ 000004 ]
 Hello world 2.0 [ 000005 ]
 Hello world 2.0 [ 000006 ]
 Hello world 2.0 [ 000007 ]
 Hello world 2.0 [ 000008 ]
 Hello world 2.0 [ 000009 ]
```

Figure 211

3.6.23 I2C example

Sample application showing how to communicate with an I2C slave device. Debug prints on **MAIN UART**

Features

- How to open a communication channel with an I2C slave device
- How to send and receive data to/from the slave device

Setup

- Connect sensor VDD to 1v8 supply (e.g. Vaux/PwrMon pin of the module)
- Connect sensor GND to a GND pin of the module
- Connect sensor SDA to module GPIO2
- Connect sensor SCL to module GPIO3

3.6.23.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open I2C bus, setting SDA an SCL pins as 2 and 3 respectively
- Set registers to configure accelerometer -Read in a loop the 6 registers carrying the 3 axes values and show the g value for each of them

```
Starting I2C demo app. This is v1.0.7 built on Mar 26 2020 16:50:40.

Configuring the Kionix device...
opening channel /dev/I2C-30
[DEBUG] 20.18  M2MB_main.c:218 - test_I2C{M2M_DamsStart}$ -
WHOAMI content: 0x01
Configuring I2C Registers - Writing 0x4D into 0x1D register (CTRL_REG3)...
Write: success

I2C reading data from 0x1D register (CTRL_REG3)...
Read: success.
Accelerometer Enabled. ODR tilt: 12.5Hz, ODR directional tap: 400Hz, ORD Motion Wakeup: 50Hz
Configuring I2C Registers - Writing 0xC0 into 0x1B register (CTRL_REG1)...
Write: success

I2C reading data from 0x1B register (CTRL_REG1)...
Read: success.
Accelerometer Enabled. Operative mode, 12bit resolution
I2C read axes registers
------------
Reading Success.

X: -0.050 g
Y: -0.046 g
Z: 1.006 g
Reading Success.

X: -0.049 g
Y: -0.044 g
Z: 1.004 g
Reading Success.

X: -0.052 g
Y: -0.044 g
Z: 1.007 g
Reading Success.

X: -0.048 g
Y: -0.045 g
Z: 1.005 g
```

Figure 212

3.6.24 I2C Combined

Sample application showing how to communicate with an I2C slave device with I2C raw mode. Debug prints on MAIN UART

Features

- How to open a communication channel with an I2C slave device
- How to send and receive data to/from the slave device using raw mode API

Setup

- Connect sensor VDD to 1v8 supply (e.g. Vaux/PwrMon pin of the module)
- Connect sensor GND to a GND pin of the module
- Connect sensor SDA to module GPIO2
- Connect sensor SCL to module GPIO3

3.6.24.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Open I2C bus, setting SDA an SCL pins as 2 and 3 respectively
- Set registers to configure accelerometer -Read in a loop the 6 registers carrying the 3 axes values and show the g value for each of them

```
Starting I2C raw demo app. This is v1.0.13-C1 built on Jul 30 2020 11:28:18.

Configuring the I2C device...
Opening I2C channel /dev/I2C-30 ( device address is 0x0F << 1 )
Accelerometer Enabled. ODR tilt: 12.5Hz, ODR directional tap: 400Hz, ORD Motion Wakeup: 50Hz
Accelerometer Enabled. Operative mode, 12bit resolution
I2C read axes registers
------------

X: 0.000 g
Y: 0.000 g
Z: 0.000 g

X: -0.270 g
Y: 0.016 g
Z: 0.917 g

X: -0.268 g
Y: 0.013 g
Z: 0.925 g

X: -0.271 g
Y: 0.015 g
Z: 0.922 g

X: -0.267 g
Y: 0.016 g
Z: 0.918 g

X: -0.274 g
Y: 0.019 g
Z: 0.915 g
```

Figure 213

3.6.25 Little FileSystem 2

Sample application showing how use lfs2 porting with RAM disk and SPI data flash. Debug prints on **MAIN UART**

Features

- How to create and manage Ram Disk
- How to manage file-system in Ram disk partition
- How to create and manage SPI Flash memory partition
- How to manage file-system in SPI Flash memory partition

3.6.25.1 Application workflow

`M2MB_main.c`

- Init logging system
- Call Ram Disk tests
- Call Flash memory tests

`ram_utils_usage.c`

- Initialize Ram Disk
- Format and Mount partition
- List files
- Files creation and write content
- List files
- Read files
- Unmount and Release resources

`spi_utils_usage.c` - Initialize SPI Flash chip - Initialize SPI Flash Disk - Format and Mount partition - List files - Files creation and write content - List files - Read files - Delete files - Directories creation and deletion - Unmount and Release resources

Notes:

For SPI Flash a JSC memory is used with chip select pin connected to module GPIO2 pin. For better performances, a 33kOhm pull-down resistor on SPI clock is suggested. Please refer to SPI_echo sample app for SPI connection details.

For LE910Cx (both Linux and ThreadX based devices), `AT#SPIEN=1` command must be sent once before running the app

```
Starting lfs2 demo app. This is v1.0.14-C1 built on Oct 22 2020 09:43:08.

>>>>>>> Starting RAMDiskDemo ...
[DEBUG] 18.28  azx_lfs_uti:125 - azx_ram_initialize{M2M_DamsStart}$ Ram Memory allocated correctly from 0x40042228 to 0x40046228!!

Mounting partition...

Formatting...

Mounting...


Mounted partition...

<><><><>fileListUtils

List:
., 0, 2
.., 0, 2

file_name: file000.txt
size: 10
buffer: content000
mode: 0
RAM TYPE size: 10000


File created and closed: file000.txt


<><><><>fileListUtils
____INSIDE --->file000.txt, 10, 1

List:
., 0, 2
.., 0, 2
file000.txt, 10, 1

------>File reading

File: file000.txt, Size: 10, Buffer: content000

Nand released
Partition unmounted
[DEBUG] 20.31  azx_lfs_uti:165 - azx_ram_releaseResources{M2M_DamsStart}$ Ram Memory released correctly!!
```

```
>>>>>>> Starting FlashDiskDemo ...
Starting initialization...

table id[0] = 191
table id[1] = 1
table id[2] = 0

nandLFSCallback Callback event <1>
NAND Callback event: NAND_JSC_INITIALIZED <1>
nandLFSCallback Callback event <1>
NAND Callback event: NAND_JSC_INITIALIZED <1>

Mounting partition...

Formatting...
spiErase: address = 0, len = 131072

spiErase: address = 131072, len = 131072


Mounting...


Mounted partition...


<><><><>fileListUtils

List:
., 0, 2
.., 0, 2


Formatting...
spiErase: address = 0, len = 131072

spiErase: address = 131072, len = 131072


Mounting...


Mounted partition...


<><><><>fileListUtils

List:
., 0, 2
.., 0, 2

file_name: file000.txt
size: 10
buffer: content000
mode: 0


File created and closed: file000.txt
```

```
<><><><>fileListUtils

List:
., 0, 2
.., 0, 2
file000.txt, 10, 1
file001.txt, 10, 1
file002.txt, 10, 1
file003.txt, 10, 1
file004.txt, 10, 1

------>File reading

File: file000.txt, Size: 10, Buffer: content000


File: file004.txt, Size: 10, Buffer: content004


File: file002.txt, Size: 10, Buffer: content002
------>File removing
file001.txt<<<<<<<


File removed: file001.txt
file000.txt<<<<<<<


File removed: file000.txt
file004.txt<<<<<<<


File removed: file004.txt


<><><><>fileListUtils

List:
., 0, 2
.., 0, 2
file002.txt, 10, 1
file003.txt, 10, 1

spiErase: address = 59637760, len = 131072

[DEBUG] 58.61  azx_lfs_uti:648 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory created: dir000!!
[DEBUG] 59.78  azx_lfs_uti:631 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory already exists: dir000!!
spiErase: address = 59899904, len = 131072

[DEBUG] 61.70  azx_lfs_uti:648 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory created: dir001!!
spiErase: address = 60162048, len = 131072

[DEBUG] 63.67  azx_lfs_uti:648 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory created: dir002!!
```

```
<><><><>fileListUtils

List:
., 0, 2
.., 0, 2
dir000, 0, 2
dir001, 0, 2
dir002, 0, 2
file002.txt, 10, 1
file003.txt, 10, 1



<><><><>fileListUtils

List:
., 0, 2
.., 0, 2
dir001, 0, 2
dir002, 0, 2
file002.txt, 10, 1
file003.txt, 10, 1

Nand released
Partition unmounted

Unmounted process ended...

testAllInOneFunction ended...
```

3.6.26 LWM2M

Sample application showcasing LWM2M client usage with M2MB API. Debug prints on **MAIN UART**

Features

- Configure LWM2M client and enable it
- Create an instance of a custom object
- Set an integer value on a read only resource
- Set two integer values on a multi-instance read only resource
- write a string on a read/write resource
- Manage exec requests from the portal
- Manage write, read and monitoring resources changed from the portal

Requirements

This application expects the user to configure the PDP context ID 1 with the proper APN. it can be done with the following AT command:

```
AT+CGDCONT=1,"IPV4V6","<user apn>"
```

Depending on the Mobiler Network Operator and Access Technology, the APN might be automatically set by the network itself. In this case, nothing must be done by the user.

3.6.26.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the LWM2M client and start it

`lwm2m_demo.c`

`msgLWM2MTask` - Check registration status

- Initialize LWM2M client,
- Check for XML file fo custom object
- Enable unsolicited messages from client
- Create a task (lwm2m_taskCB is its callback function)to manage events from Portal
- Enable LwM2M client

- Create a new instance for the custom object
- Wait for client to register to Portal
- Performs all operations (set, read, get, write) on the related resources
- Performs a set with notify ack enabled
- Wait for events from server

`lwm2mIndicationCB`

- Manage events arriving from client (operations completion status and unsolicited events)
- Run lwm2m_taskCB when a monitored resource changes, to manage the action to be done

3.6.26.2 Device Profile upload

LWM2M resources demo device profile must be imported to have a real-time update of resources values on the LWM2M browser.

To do so, import the file `json/lwm2m_resources_demo.json` (provided with the sample files) on section `Developer` > `Device profiles` of OneEdge IoT portal:

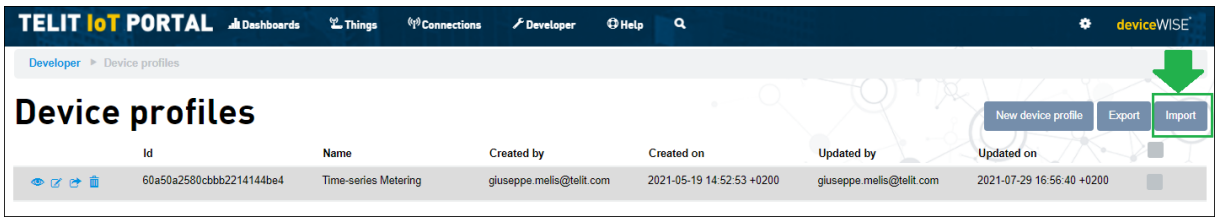

Figure 214

3.6.26.3 Custom Object configuration

The XML file content must be loaded on the Telit IoT Portal for the demo application to be fully executed.

First, enter Developer section from the top menu

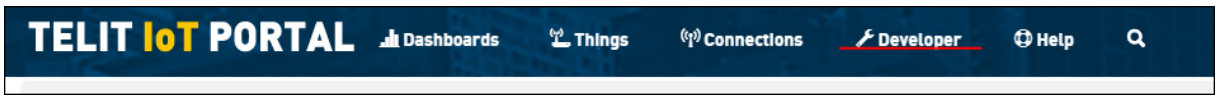

Figure 215

Choose Object Registry

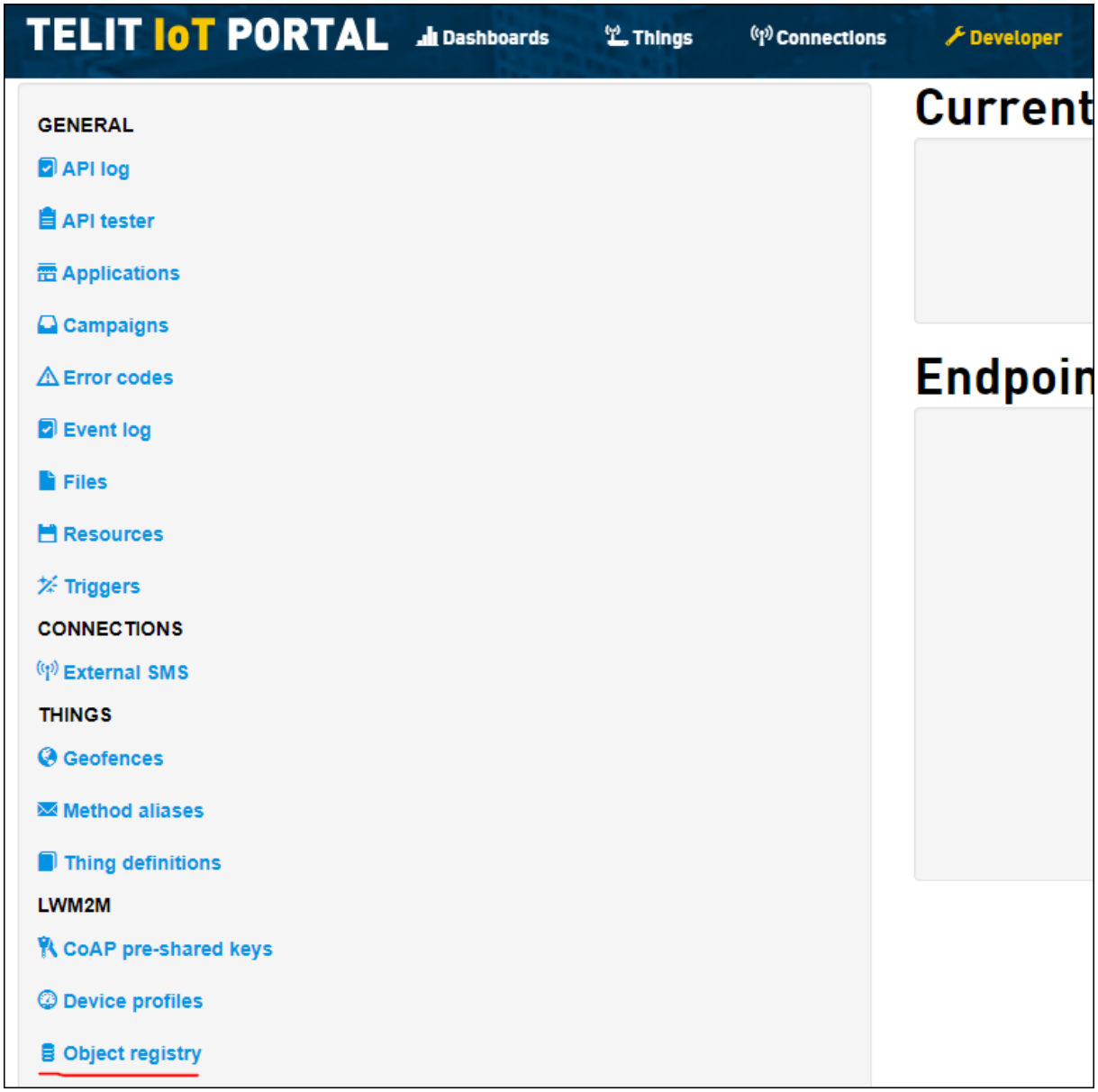

Figure 216

Create a New Object

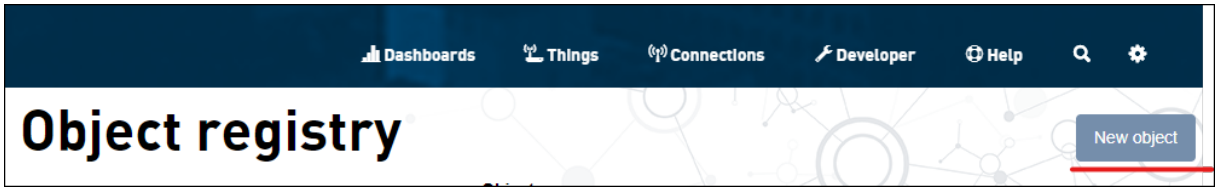

Figure 217

Copy the xml file content and paste it in the new Object form

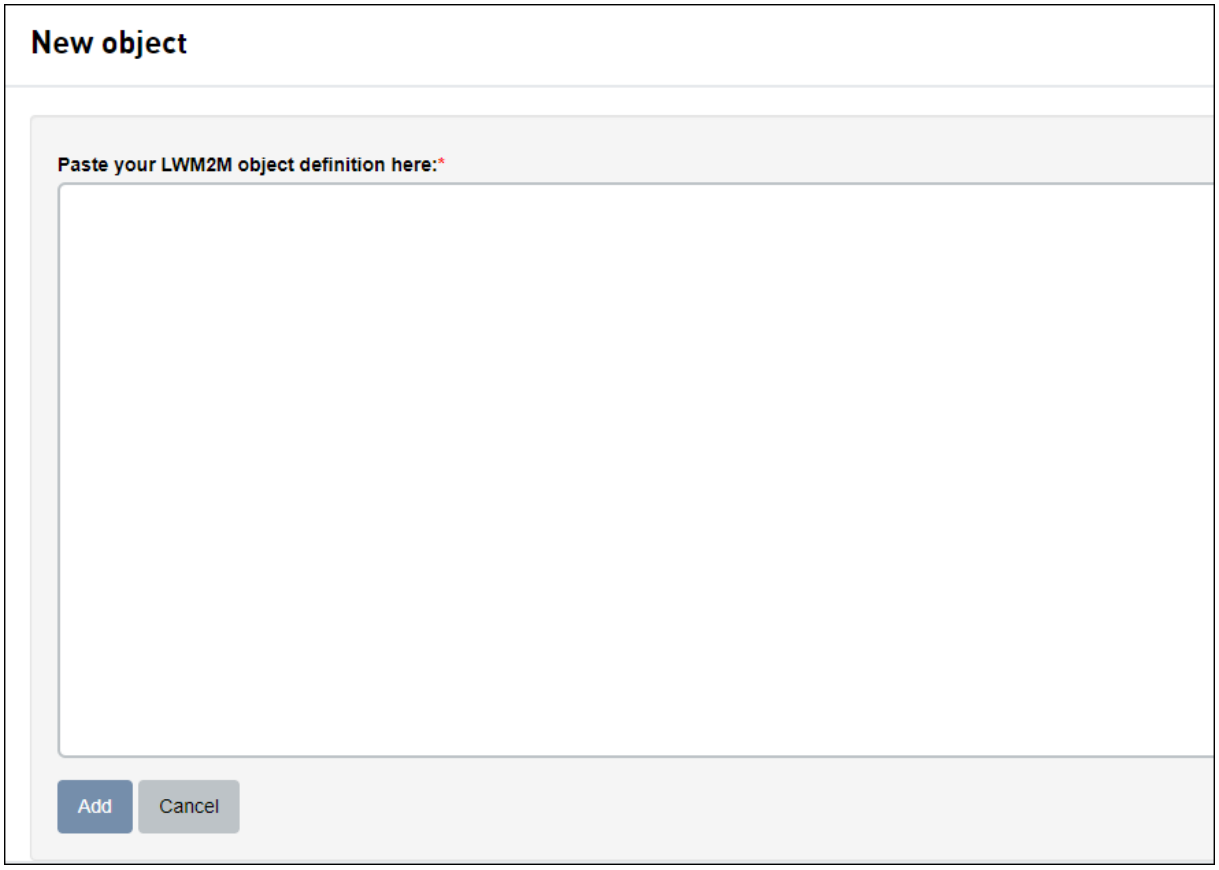

Figure 218

Also, the application requires the XML file `xml/object_32010.xml` (provided with the sample files) to be stored in module's `/XML/` folder. It can be done with

```
AT#M2MWRITE=/XML/object_32010.xml,<size_in_bytes>
```

To load the XML file in the module, Telit AT Controller (TATC) can be used. Once the command above is issued, press the load content button:

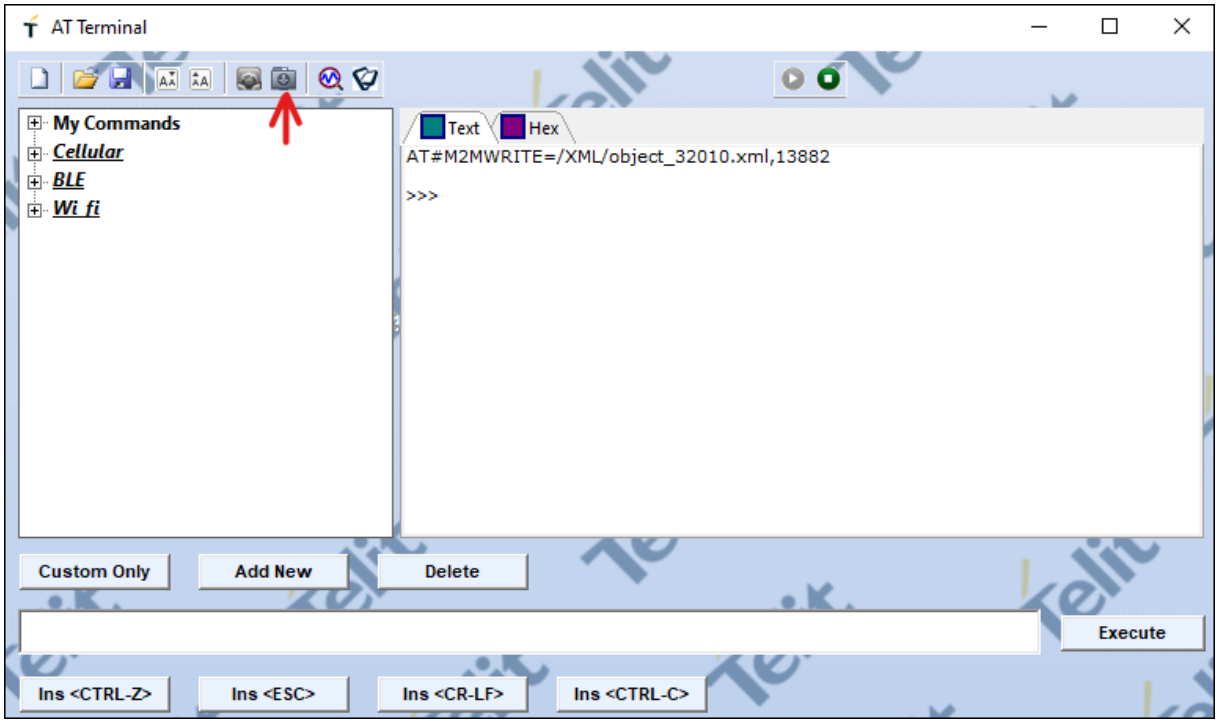

Figure 219

Select the file from your computer

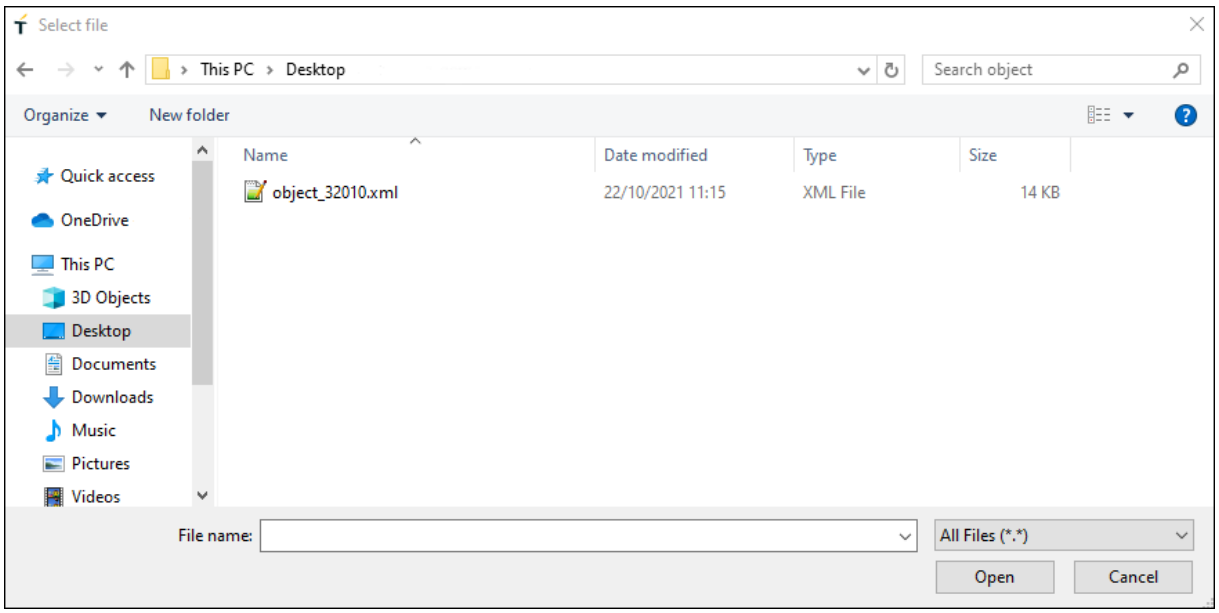

Figure 220

The file is successfully loaded on the module

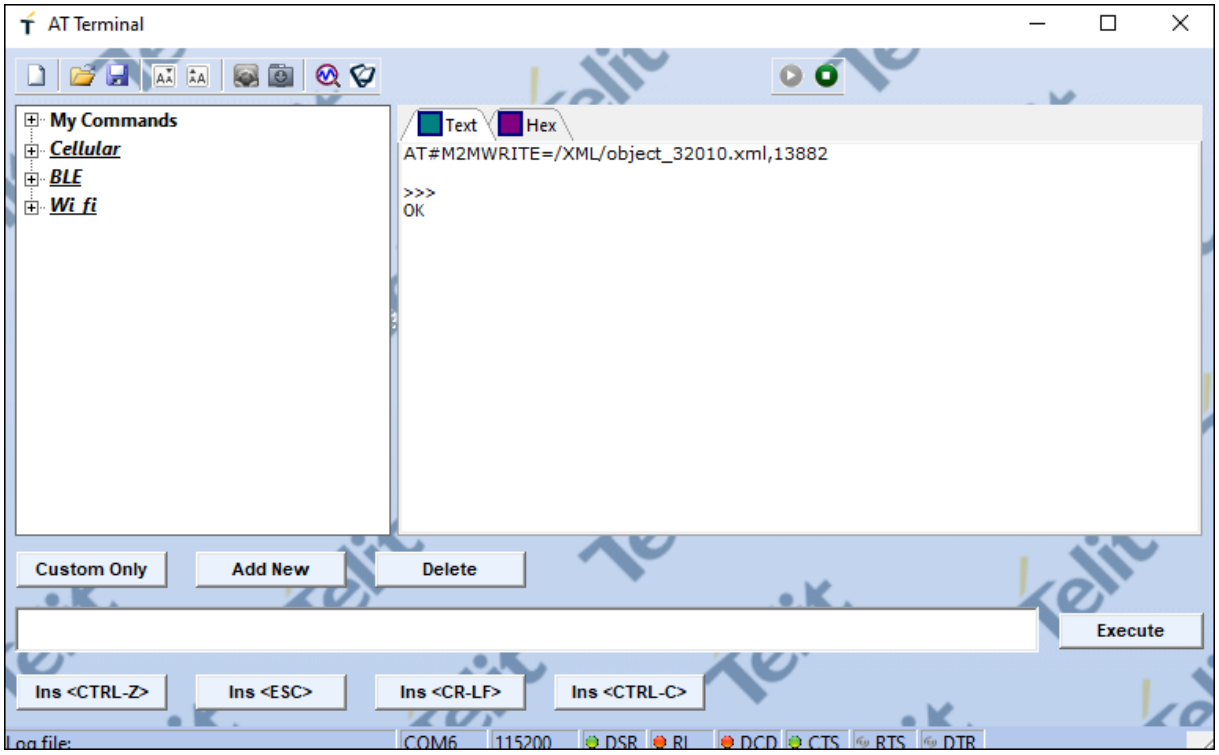

Figure 221

3.6.26.4 Onboard the device

Get the Telit ID

To retrieve the Telit ID data, issue AT#TID to get the Telit ID. The command response will be similar to

#TID: **xxxxxxxxxxxxxxxxxxxxxxxxxxxx**,1 OK

Take note of the Telit ID highlighted in **bold** above (or copy it on a text editor): this ID it will be needed for the onboarding process.

Create a new Thing

From the OneEdge portal, on **"Things"** section, click **"New Thing"** button in the top right corner.

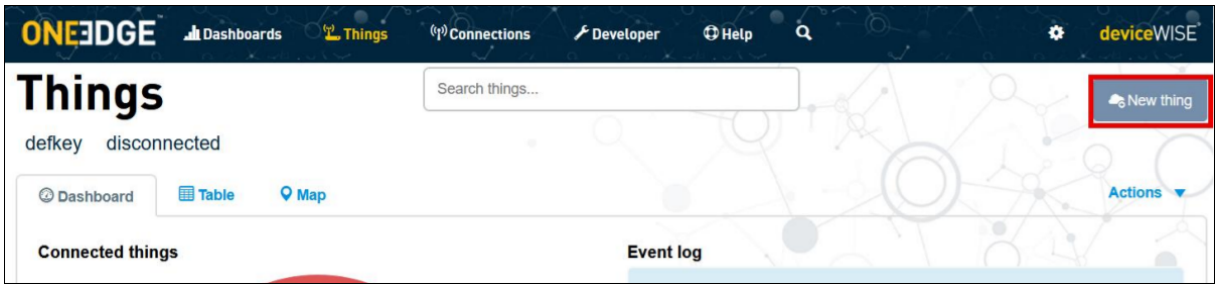

Figure 222

In the Create a new thing dialog, select “Telit Module”

Figure 223

A dialog appears: select “Default” thing definition

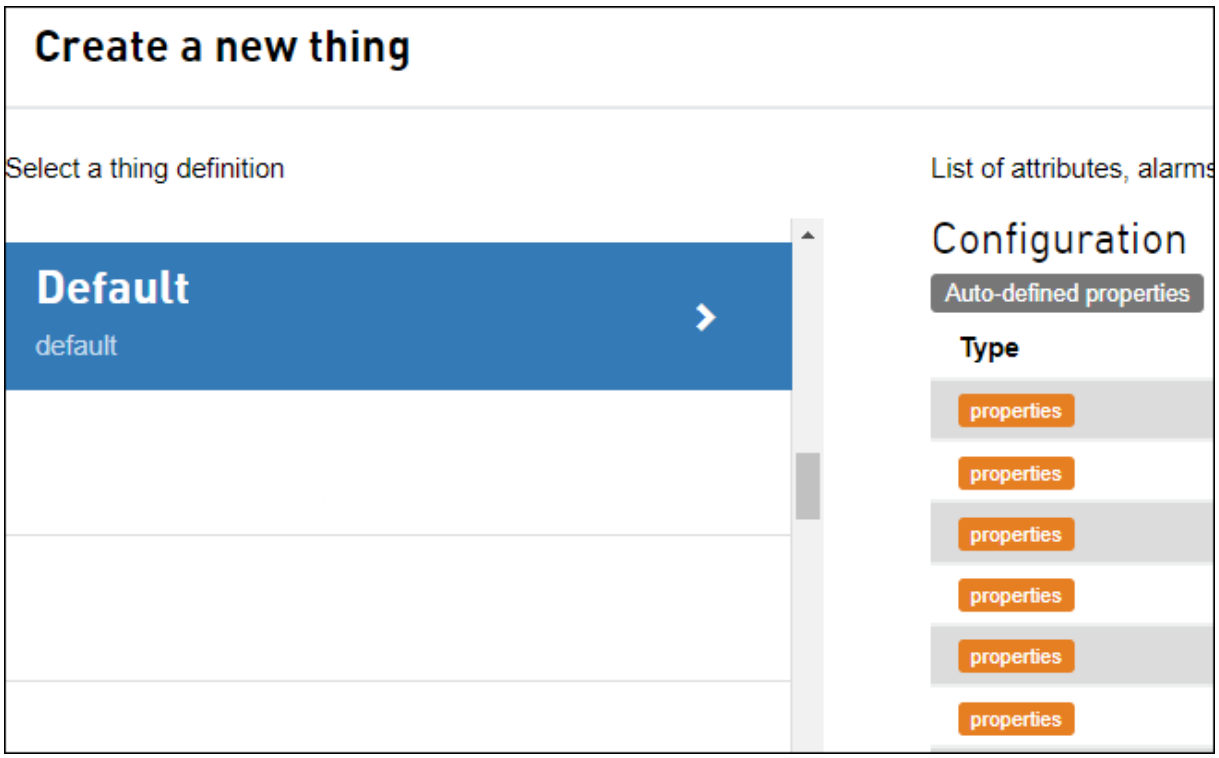

Figure 224

In the following screen, provide the Telit ID as “Identifier” Click on “Find” and make sure that model, firmware and the other details are properly populated.

Click on lwm2m tab and set the device profile previously imported as shown in the

screenshot below

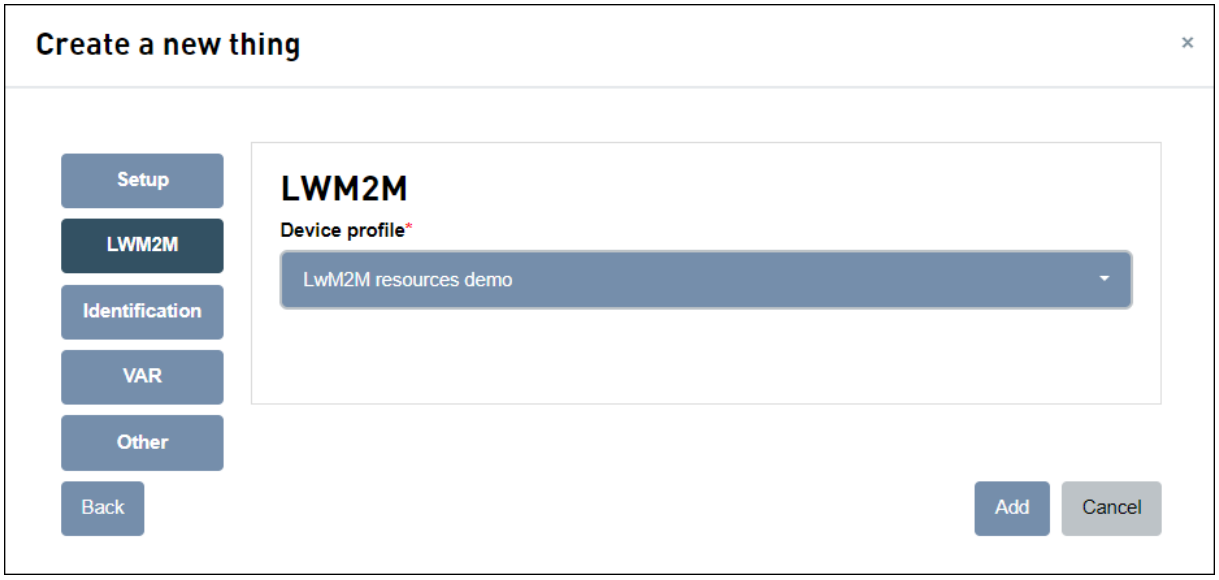

Figure 225

Click **"Add"** to complete the new thing creation procedure.

If the Thing already exists, its device profile can be changed by following the steps shown in the picture below

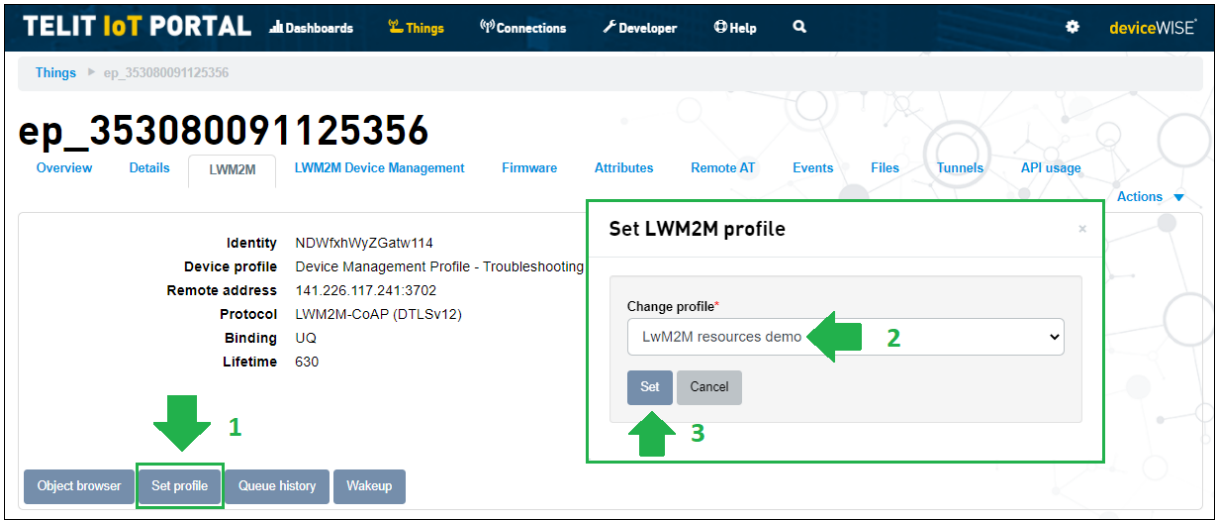

Figure 226

3.6.26.5 Application execution example

```
Starting lwm2m demo. This is v1.1.10 built on Oct 22 2021 10:49:52.
On OneEdge portal, be sure that the demo Device Profile has been uploaded on the IoT Portal
Initializing resources...

LWM2M enable result OK
[DEBUG] 38.20  lwm2m_demo:998 - lwm2mIndicationCB{pubTspt_0}$ Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED
GET STATUS.
IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED
```

Figure 227

```
=================================
READ-ONLY RESOURCES

=================================

Setting integer resource {32010/0/2} value to 50 on LWM2M client.

Reading integer resource {32010/0/2} value on LWM2M client.

Resource /32010/0/2/0 changed!

Reading integer resource {32010/0/2/0} value on LWM2M client.

---Integer value is now 50

Integer data in {32010/0/2/0} resource was updated to new value: 50

-------------------------------------------------

Setting integer resource {32010/0/22/0} value to 10 on LWM2M client.

Resource /32010/0/22/0 changed!

Reading integer resource {32010/0/22/0} value on LWM2M client.

Integer data in {32010/0/22/0} resource was updated to new value: 10

Setting integer resource {32010/0/22/1} value to 11 on LWM2M client.

Resource /32010/0/22/1 changed!

Reading integer resource {32010/0/22/1} value on LWM2M client.

Integer data in {32010/0/22/1} resource was updated to new value: 11

-------------------------------------------------

Setting double resource {32010/0/3} value to 20.500000 on LWM2M client.

Reading double resource {32010/0/3} value on LWM2M client.

Resource /32010/0/3/0 changed!

Reading double resource {32010/0/3/0} value  on LWM2M client.

---Double value is now 20.500000

Float data in {32010/0/3/0} resource was updated to new value: 20.500000

-------------------------------------------------

Setting boolean resource {32010/0/4} value to 1 on LWM2M client.

Reading boolean resource {32010/0/4} value on LWM2M client.

Resource /32010/0/4/0 changed!

Reading boolean resource {32010/0/4/0} value on LWM2M client.

---Boolean value is now true

Boolean data in {32010/0/4/0} resource was updated to new value: true

-------------------------------------------------
```

```
Current time in seconds from the epoch: 1634900084

Setting timestamp resource {32010/0/6} value to 1634900084 on LWM2M client.

Reading timestamp resource {32010/0/6} value on LWM2M client.

Resource /32010/0/6/0 changed!

Reading timestamp resource {32010/0/6/0} value on LWM2M client.

---Timestamp value is now 1634900084

Time data in {32010/0/6/0} resource was updated to new value: 1634900084 (2021/10/22T10:54:44+00:00)

--------------------------------------------------

Setting opaque resource {32010/0/5} on LWM2M client.

Resource /32010/0/5/0 changed!

Reading opaque resource {32010/0/5/0} on LWM2M client.

Opaque data in {32010/0/5/0} resource was updated to new content (10 bytes):
00 01 02 03 04 05 06 07 08 09

Reading opaque resource {32010/0/5} on LWM2M client.

---Opaque content is now (10 bytes): 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09

--------------------------------------------------

Setting string resource {32010/0/1} value to Hello World! on LWM2M client.

Reading string resource {32010/0/1} value on LWM2M client.

Resource /32010/0/1/0 changed!

Reading integer resource {32010/0/1/0} value on LWM2M client.

---String content is now: <Hello World!>

String data in {32010/0/1/0} resource was updated to new content: <Hello World!>

--------------------------------------------------
```

```
=================================
READ-WRITE RESOURCES
=================================

Writing integer resource {32010/0/12} value to 50 on LWM2M client.

Reading integer resource {32010/0/12} value on LWM2M client.

Resource /32010/0/12/0 changed!

Reading integer resource {32010/0/12/0} value on LWM2M client.

---Integer value is now 50

Integer data in {32010/0/12/0} resource was updated to new value: 50

--------------------------------------------------

Writing double resource {32010/0/13} value to 20.500000 on LWM2M client.

Reading double resource {32010/0/13} value on LWM2M client.

Resource /32010/0/13/0 changed!

Reading double resource {32010/0/13/0} value  on LWM2M client.

---Double value is now 20.500000

Float data in {32010/0/13/0} resource was updated to new value: 20.500000

--------------------------------------------------

Writing boolean resource {32010/0/14} value to 1 on LWM2M client.

Reading boolean resource {32010/0/14} value on LWM2M client.

Resource /32010/0/14/0 changed!

Reading boolean resource {32010/0/14/0} value on LWM2M client.

---Boolean value is now true

Boolean data in {32010/0/14/0} resource was updated to new value: true

--------------------------------------------------
```

```
Current time in seconds from the epoch: 1634900125

Writing timestamp resource {32010/0/16} value to 1634900125 on LWM2M client.

Reading timestamp resource {32010/0/16} value on LWM2M client.

Resource /32010/0/16/0 changed!

Reading timestamp resource {32010/0/16/0} value on LWM2M client.

---Timestamp value is now 1634900125

Time data in {32010/0/16/0} resource was updated to new value: 1634900125 (2021/10/22T10:55:25+00:00)

--------------------------------------------------

Writing opaque resource {32010/0/15} on LWM2M client.

Resource /32010/0/15/0 changed!

Reading opaque resource {32010/0/15/0} on LWM2M client.

Opaque data in {32010/0/15/0} resource was updated to new content (10 bytes):
00 01 02 03 04 05 06 07 08 09

Reading opaque resource {32010/0/15} on LWM2M client.

---Opaque content is now (10 bytes): 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09

--------------------------------------------------

Writing string resource {32010/0/11} value to <Hello World!> on LWM2M client.

Reading string resource {32010/0/11} value on LWM2M client.

Resource /32010/0/11/0 changed!

Reading integer resource {32010/0/11/0} value on LWM2M client.

---String content is now: <Hello World!>
```

```
==================================
WRITE-ONLY RESOURCES
==================================

Writing integer resource {32010/0/42} value to 50 on LWM2M client.

String data in {32010/0/11/0} resource was updated to new content: <Hello World!>

Getting integer resource {32010/0/42} valueon LWM2M client.

Resource /32010/0/42/0 changed!

Getting integer resource {32010/0/42/0} value on LWM2M client.

---Integer value is now 50

Integer data in {32010/0/42/0} resource was updated to new value: 50

--------------------------------------------------

Writing double resource {32010/0/43} value to 20.500000 on LWM2M client.

Getting double resource {32010/0/43} value on LWM2M client.

Resource /32010/0/43/0 changed!

Getting double resource {32010/0/43/0} value  on LWM2M client.

---Double value is now 20.500000

Float data in {32010/0/43/0} resource was updated to new value: 20.500000

--------------------------------------------------

Writing boolean resource {32010/0/44} value to 1 on LWM2M client.

Getting boolean resource {32010/0/44} value on LWM2M client.

Resource /32010/0/44/0 changed!

Getting boolean resource {32010/0/44/0} value on LWM2M client.

---Boolean value is now true

Boolean data in {32010/0/44/0} resource was updated to new value: true

--------------------------------------------------
```

```
Current time in seconds from the epoch: 1634900163

Writing timestamp resource {32010/0/46} value to 1634900163 on LWM2M client.

Getting timestamp resource {32010/0/46} value on LWM2M client.

Resource /32010/0/46/0 changed!

Getting timestamp resource {32010/0/46/0} value on LWM2M client.

---Timestamp value is now 1634900163

Time data in {32010/0/46/0} resource was updated to new value: 1634900163 (2021/10/22T10:56:03+00:00)

--------------------------------------------------

Writing opaque resource {32010/0/45} on LWM2M client.

Resource /32010/0/45/0 changed!

Getting opaque resource {32010/0/45/0} on LWM2M client.

Opaque data in {32010/0/45/0} resource was updated to new content (10 bytes):
00 01 02 03 04 05 06 07 08 09

Getting opaque resource {32010/0/45} on LWM2M client.

---Opaque content is now (10 bytes): 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09

--------------------------------------------------

Writing string resource {32010/0/41} value to <Hello World!> on LWM2M client.

Getting string resource {32010/0/41} value on LWM2M client.

Resource /32010/0/41/0 changed!

Getting integer resource {32010/0/41/0} value on LWM2M client.

---String content is now: <Hello World!>

String data in {32010/0/41/0} resource was updated to new content: <Hello World!>

-----------------------------------
Resources operations examples done.
-----------------------------------
```

```
-----------------------------------
Will perform a SET with notify ACK enabled.
-----------------------------------
Enable notify ack

Setting integer resource {32010/0/2} value to 60 on LWM2M client.

Resource /32010/0/2/0 changed!

Reading integer resource {32010/0/2/0} value on LWM2M client.

Integer data in {32010/0/2/0} resource was updated to new value: 60

ACK received from server!

Reading integer resource {32010/0/2} value on LWM2M client.

---Integer value is now 60

Disable notify ack
Done.

Waiting for events from the OneEdge portal. Please write on monitored resources or call an 'exec' one.
```

After the Demo completes the initialization, it is possible to access the object resources from the section Things: select your device, then “LwM2M” tab of OneEdge IoT portal:

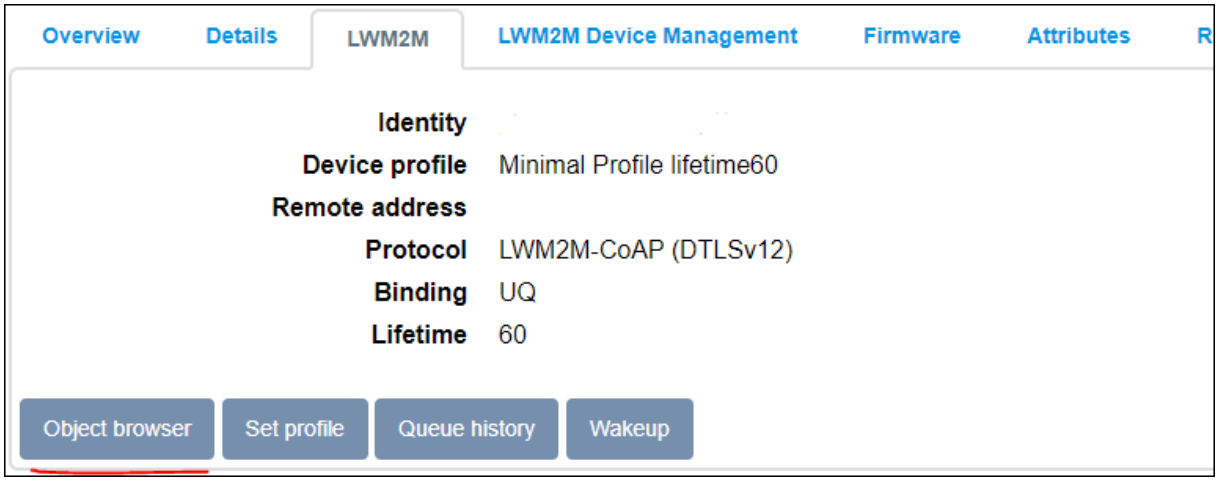

Figure 228

An instance of the object will be present and the resources can be modified.

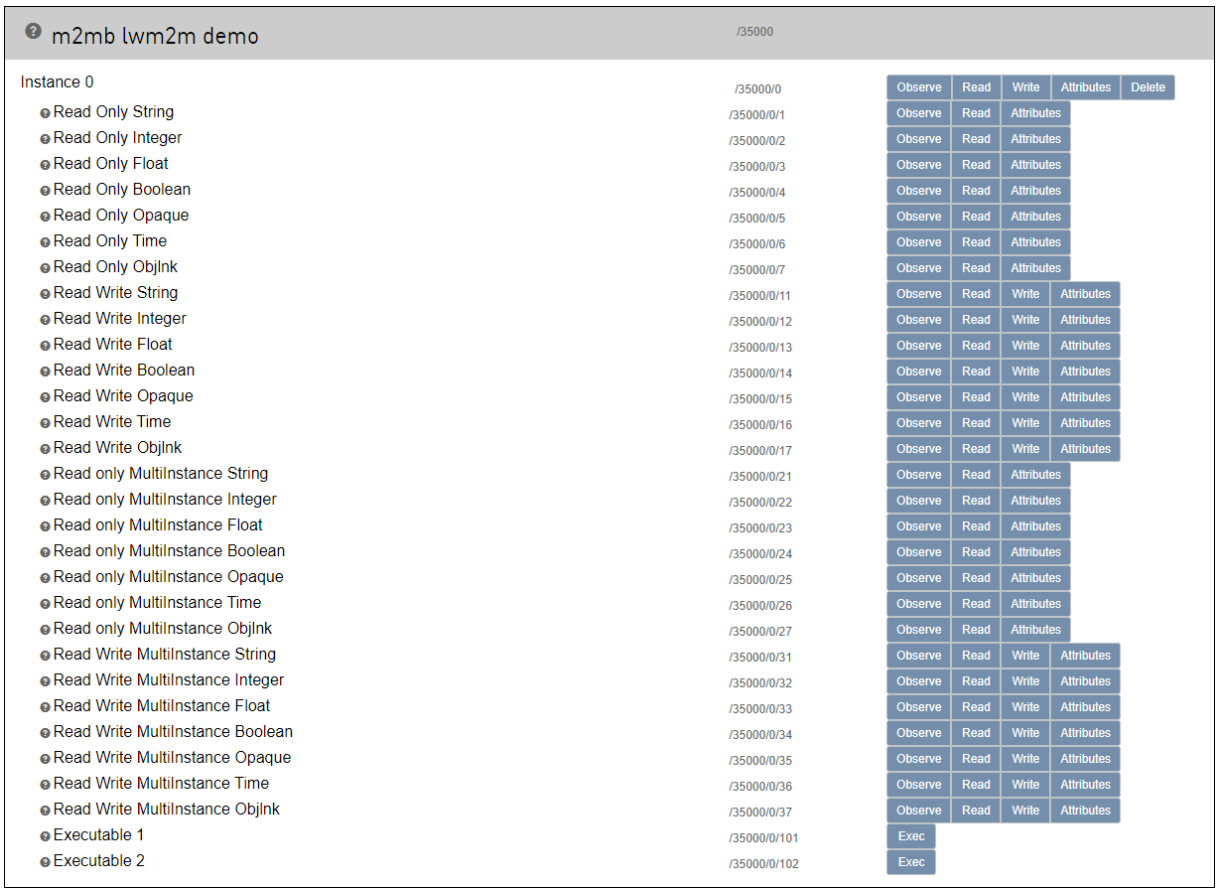

Figure 229

For example, executing the two Exec Resources at the bottom of the list, the application will react accordingly:

```
Info Exec Ind: 32010/0/101/0
Asked to execute resource 101

Resource /32010/0/11/0 changed!
```

Figure 230

Writing a string resource (id /32010/0/11), the application will notify the change

```
Reading integer resource {32010/0/11/0} value on LWM2M client.

String data in {32010/0/11/0} resource was updated to new content: <Hello from the IoT Portal!>
```

Figure 231

3.6.27 LWM2M FOTA ACK management (AT URCs)

Sample application showcasing LWM2M client FOTA events and ACKs management via AT URCs. Debug prints on **MAIN UART**

Features

- Configure LWM2M client and enable it (if not already running)
- Enable monitoring on object 5 (Firmware Update)
- Configure lwm2m FOTA parameters to enable ACK requirement for each operation (Download, Update)
- Wait for an externally generated FOTA request to handle the steps by sending the acknoledgements when needed

Please refer to "80654NT11889A OneEdge Firmware Management Application Note" for further details about FOTA workflow. Get in touch with TS-OneEdge support to receive the document.

Requirements

This application expects the user to configure the PDP context ID 1 with the proper APN. it can be done with the following AT command:

```
AT+CGDCONT=1,"IPV4V6","<user apn>"
```

Depending on the Mobile Network Operator and Access Technology, the APN might be automatically set by the network itself. In this case, nothing must be done by the user.

3.6.27.1 Device Profile upload

Minimal FOTA profile (short lifetime) device profile must be imported and selected to improve the responsiveness of the FOTA operations

To do so, import the file `json/lwm2m_fota_profile_short.json` (provided with the sample files) on section `Developer` > `Device profiles` of OneEdge IoT portal:

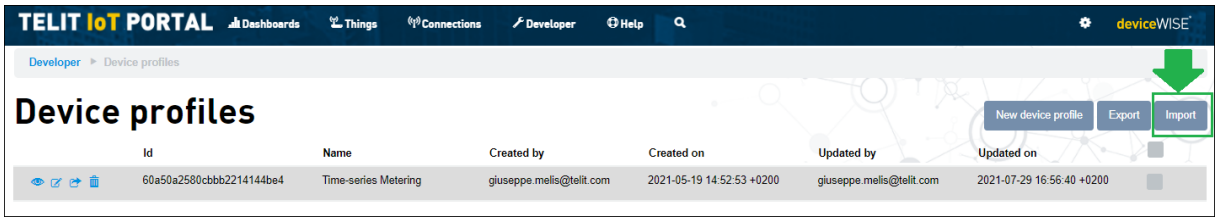

Figure 232

3.6.27.2 Onboard the device

Get the Telit ID

To retrieve the Telit ID data, issue AT#TID to get the Telit ID. The command response will be similar to

#TID: **xxxxxxxxxxxxxxxxxxxxxxxxxxxx**,1 OK

Take note of the Telit ID highlighted in **bold** above (or copy it on a text editor): this ID it will be needed for the onboarding process.

Create a new Thing

From the OneEdge portal, on **"Things"** section, click **"New Thing"** button in the top right corner.

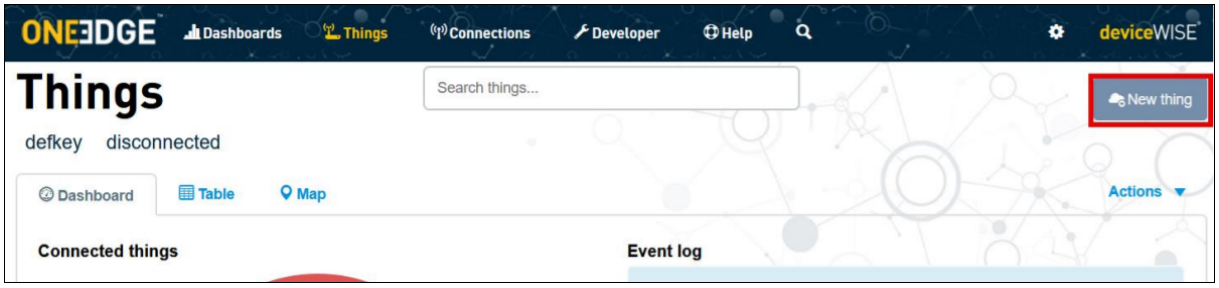

Figure 233

In the Create a new thing dialog, select "Telit Module"

Figure 234

A dialog appears: select "Default" thing definition

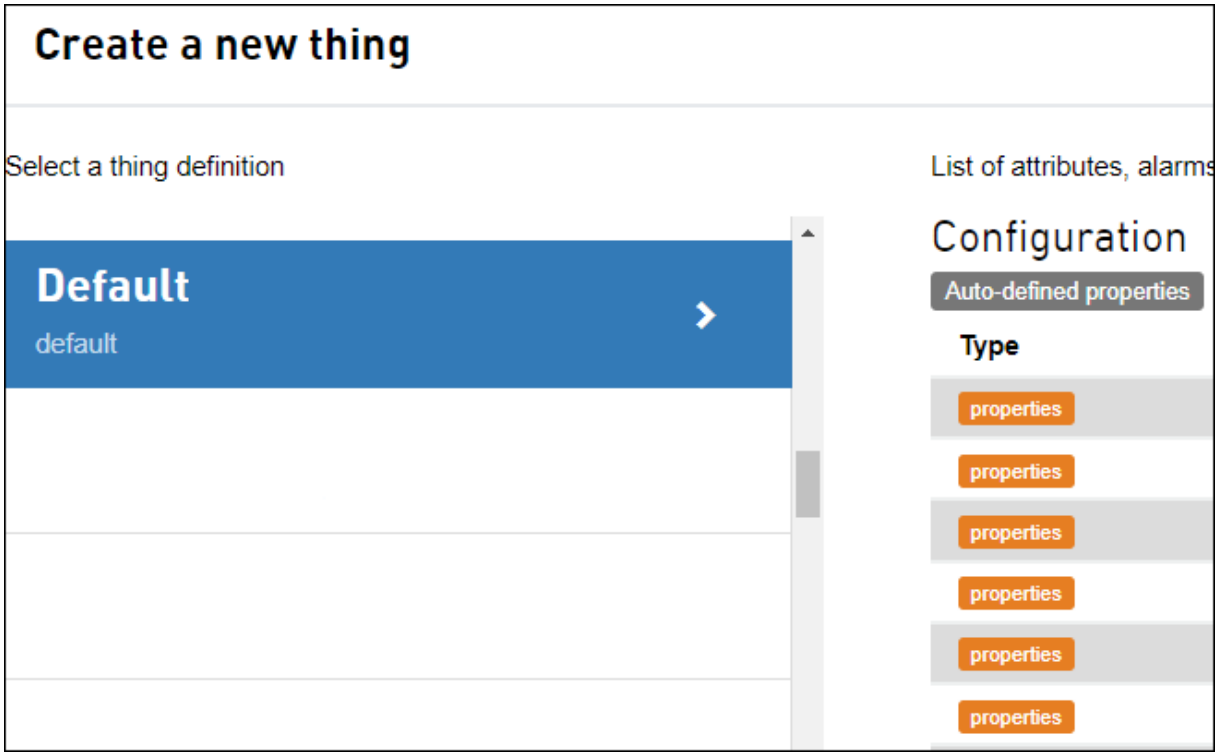

Figure 235

In the following screen, provide the Telit ID as “Identifier” Click on “Find” and make sure that model, firmware and the other details are properly populated.

Click on lwm2m tab and set the device profile previously imported as shown in the screenshot below

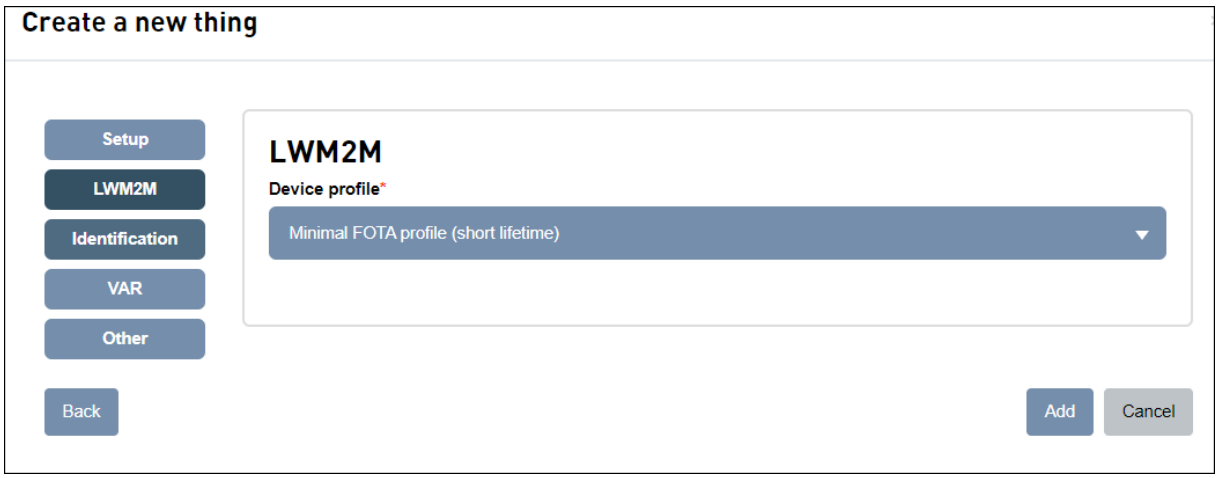

Figure 236

Click **“Add”** to complete the new thing creation procedure.

3.6.27.3 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Create a task to manage the LWM2M client and start it

lwm2m_demo.c

msgLWM2MTask

- Initialize LWM2M client,
- Enable unsolicited messages from client
- Enable FOTA ACK configuration
- Create a task (lwm2m_taskCB is its callback function)to manage events from Portal
- Check if LwM2M client is running, if not, enable it
- Wait for client to register to Portal
- Wait for FOTA events to arrive, and when the Update request arrives, notify the user that everything must stop before sending the last ACK (as the module will start updating and then reboot)

lwm2mIndicationCB

- Manage events arriving from client (operations completion status and unsolicited events)
- Run lwm2m_taskCB when a monitored resource changes, to manage the action to be done

3.6.27.4 Application execution example

```
Starting lwm2m FOTA management demo. This is v1.1.14 built on Jan 21 2022 14:39:47.
Initializing resources...
[DEBUG] 22.14  lwm2m_demo:283 - at_cmd_async_init{LWM2M_TASK}$ m2mb_ati_init() on instance 1

IF Status: M2MB_LWM2M_IF_STATE_DISABLED
Client Status: M2MB_LWM2M_CL_STATE_DISABLED
Enable lwm2m client
LWM2M enable result OK
Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED

IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED

Waiting for events from the OneEdge portal. Please start a FOTA process.
```

Figure 237

```
Resource /5/0/1/0 changed!
[DEBUG] 108.42  lwm2m_demo:544 - lwm2m_taskCB{LWM2M_TASK}$ Firmware update event!
FOTA process is waiting for Download ACK, sending it!
Waiting for Update ACK!
Before sending the ACK to apply FOTA, stop any ongoing operation (filesystem, sockets)

Asked to perform UPDATE step, stopping all operations now.
Application complete.

------- Info, DEVICE REBOOT FOR FOTA ------
????????
#OTAEV: Module Upgraded To New Fw
```

FOTA request arrives from server, ACK is sent for Download and Update. Device restarts with the new firmware version

```
Starting lwm2m FOTA management demo. This is v1.1.14 built on Jan 21 2022 14:39:47.
Initializing resources...
[DEBUG] 26.61  lwm2m_demo:283 - at_cmd_async_init{LWM2M_TASK}$ m2mb_ati_init() on instance 1
[WARN ] 34.98  lwm2m_demo:738 - lwm2mIndicationCB{pubTspt_0}$ write res 1

IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
Enable already done, skip. State: M2MB_LWM2M_IF_STATE_ACTIVE
Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED

IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED

Waiting for events from the OneEdge portal. Please start a FOTA process.
```

Application restarts and is ready for a new FOTA operation

3.6.28 LWM2M OBJ_GET AND OBJ_SET

Sample application showcasing LWM2M client m2mb_lwm2m_objget and m2mb_lwm2m_objset M2MB APIs usage. Debug prints on **MAIN UART**

Features

- Configure LWM2M client and enable it
- Create an instance of a custom object
- Create a Json string
- Set string, integer, float, boolean, timestamp and opaque values with `m2mb_lwm2m_objset`
- Get all resources values with `m2mb_lwm2m_objget`
- Manage write, read and monitoring resources changed from the portal

Requirements

This application expects the user to configure the PDP context ID 1 with the proper APN. it can be done with the following AT command:

```
AT+CGDCONT=1,"IPV4V6","<user apn>"
```

Depending on the Mobiler Network Operator and Access Technology, the APN might be automatically set by the network itself. In this case, nothing must be done by the user.

3.6.28.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the LWM2M client and start it

`lwm2m_demo.c`

`msgLWM2MTask` - Check registration status

- Initialize LWM2M client,
- Check for XML file fo custom object
- Enable unsolicited messages from client
- Create a task (lwm2m_taskCB is its callback function)to manage events from Portal
- Enable LwM2M client

- Create a new instance for the custom object
- Wait for client to register to Portal
- Performs obj_set and obj_get operations on the related resources
- Wait for events from server

`lwm2mIndicationCB`

- Manage events arriving from client (operations completion status and unsolicited events)
- Run lwm2m_taskCB when a monitored resource changes, to manage the action to be done

3.6.28.2 Device Profile upload

LWM2M resources demo device profile must be imported to have a real-time update of resources values on the LWM2M browser.

To do so, import the file `json/lwm2m_resources_demo.json` (provided with the sample files) on section `Developer` > `Device profiles` of OneEdge IoT portal:

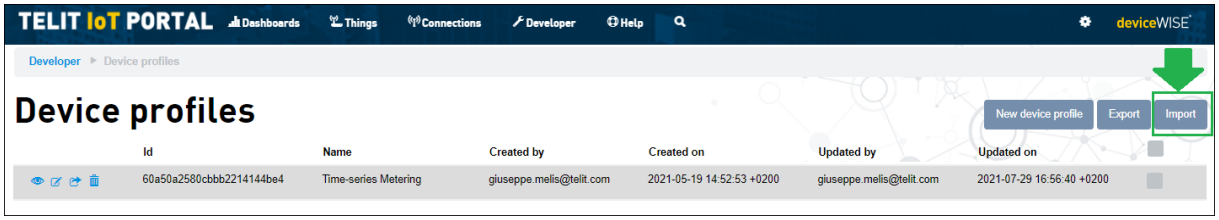

Figure 238

3.6.28.3 Custom Object configuration

The XML file content must be loaded on the Telit IoT Portal for the demo application to be fully executed.

First, enter Developer section from the top menu

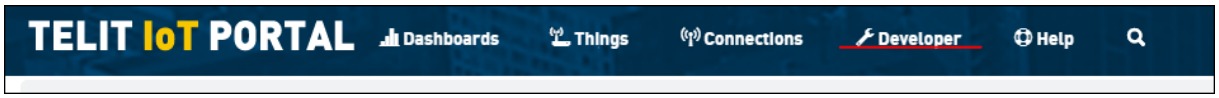

Figure 239

Choose Object Registry

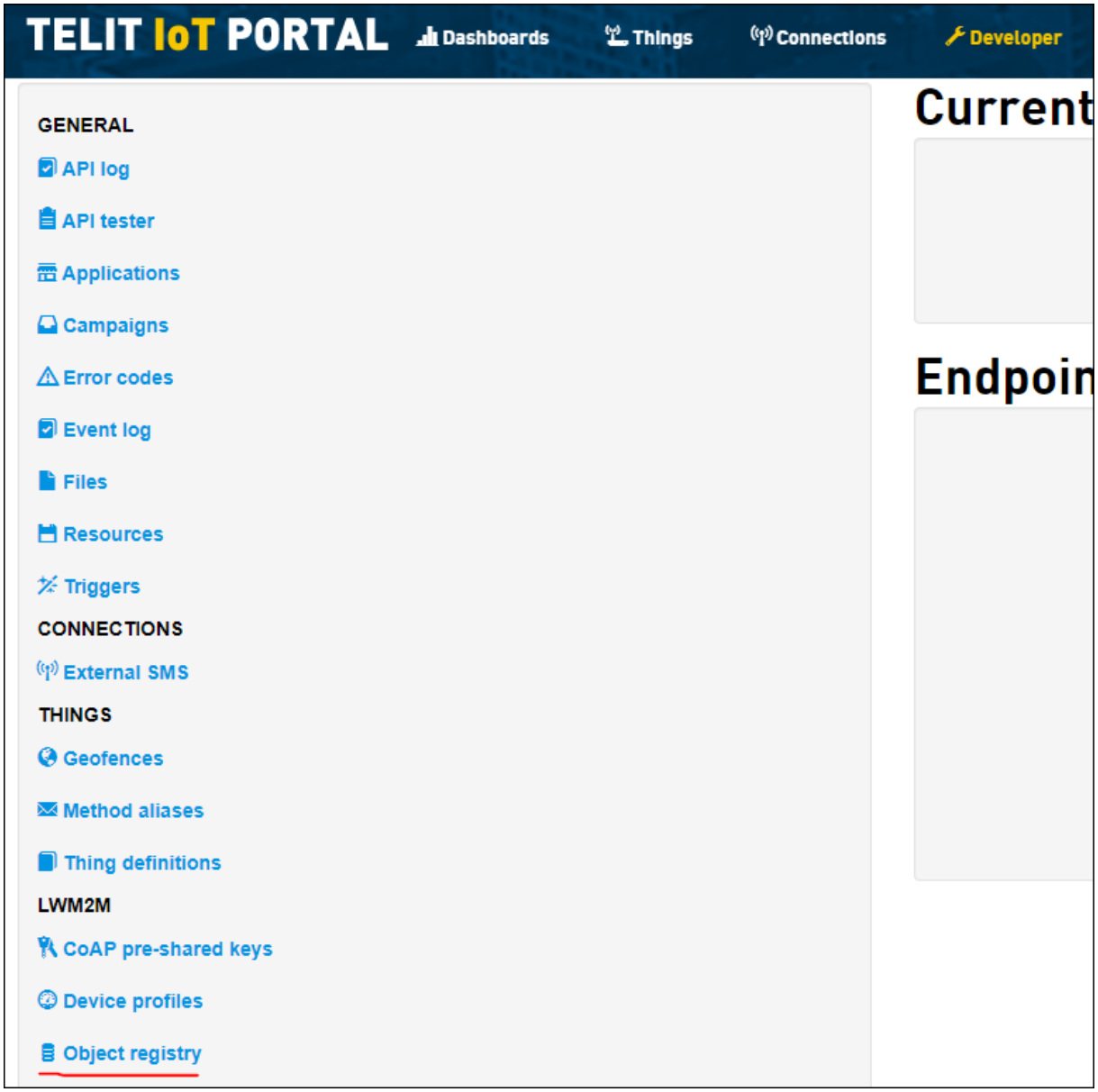

Figure 240

Create a New Object

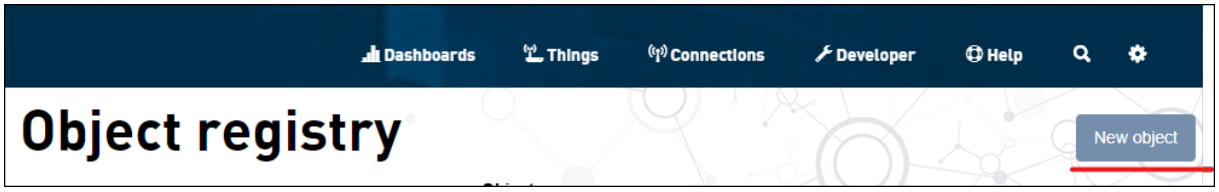

Figure 241

Copy the xml file content and paste it in the new Object form

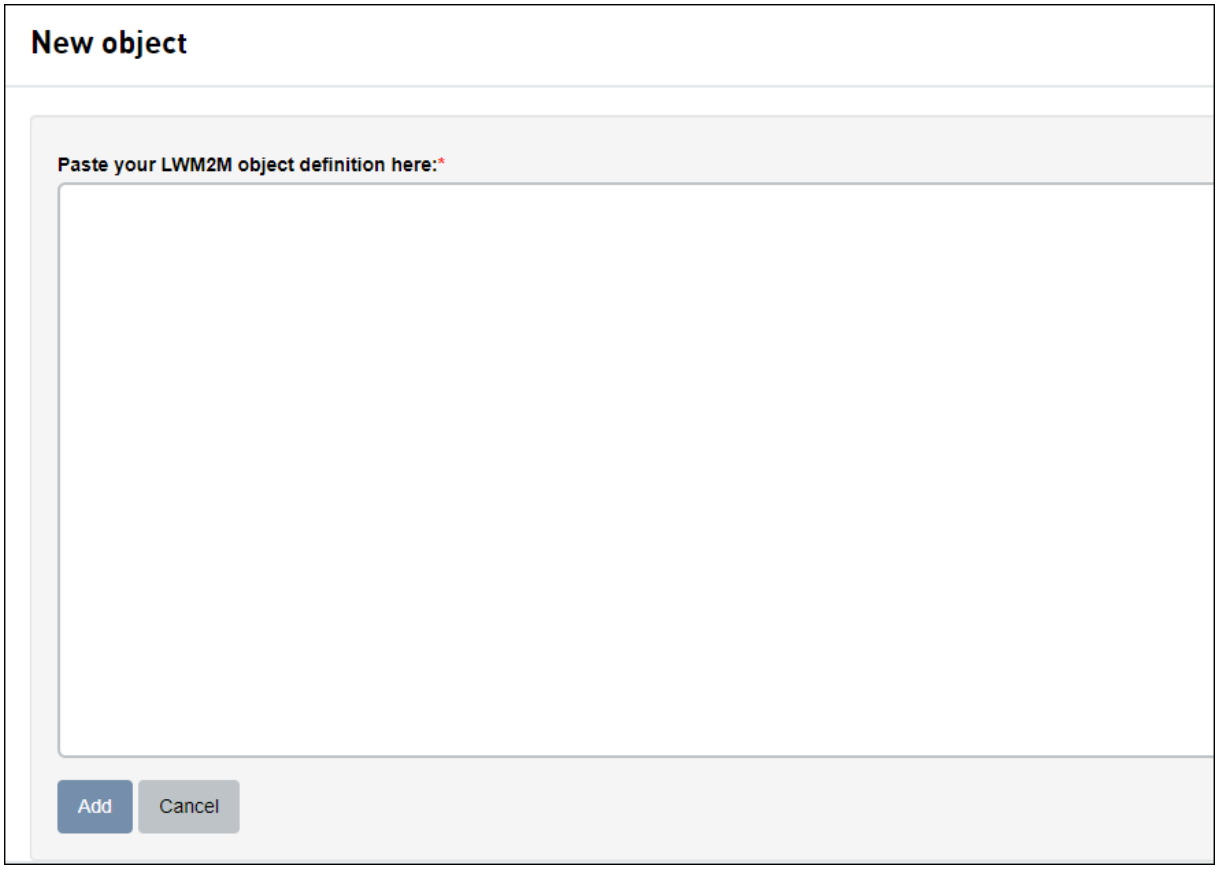

Figure 242

Also, the application requires the XML file `xml/object_32011.xml` (provided with the sample files) to be stored in module's `/XML/` folder. It can be done with

```
AT#M2MWRITE=/XML/object_32011.xml,<size_in_bytes>
```

To load the XML file in the module, Telit AT Controller (TATC) can be used. Once the command above is issued, press the load content button:

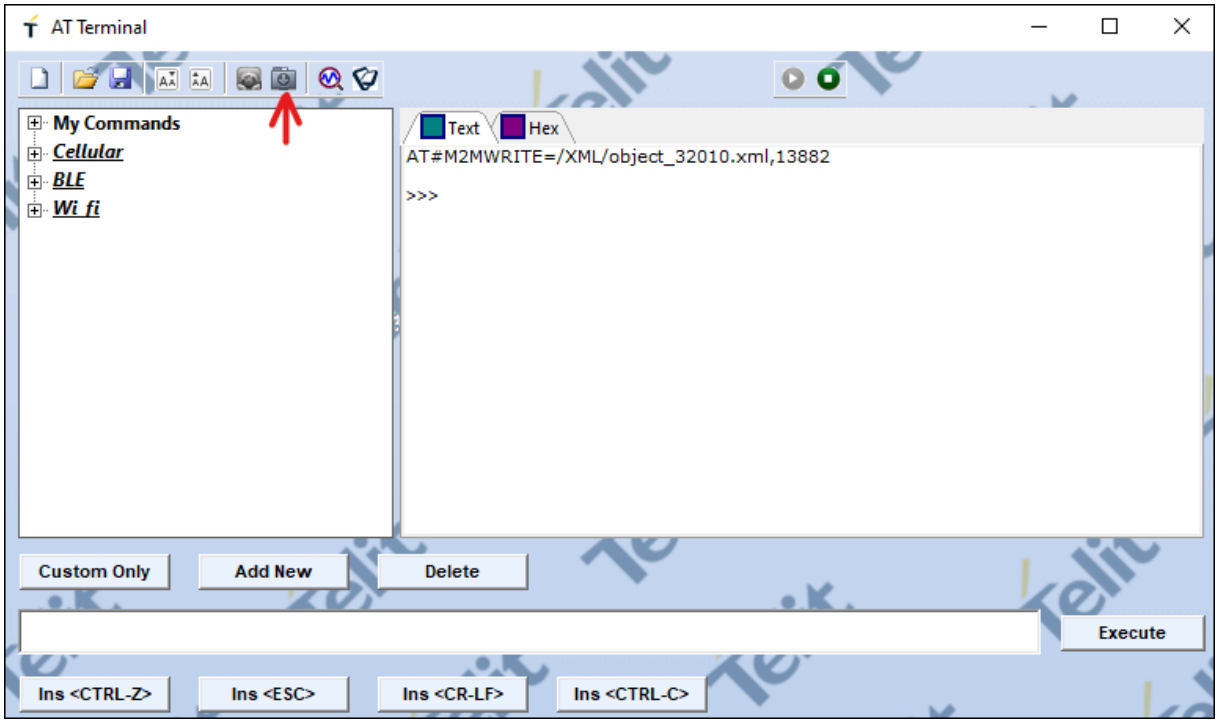

Figure 243

Select the file from your computer

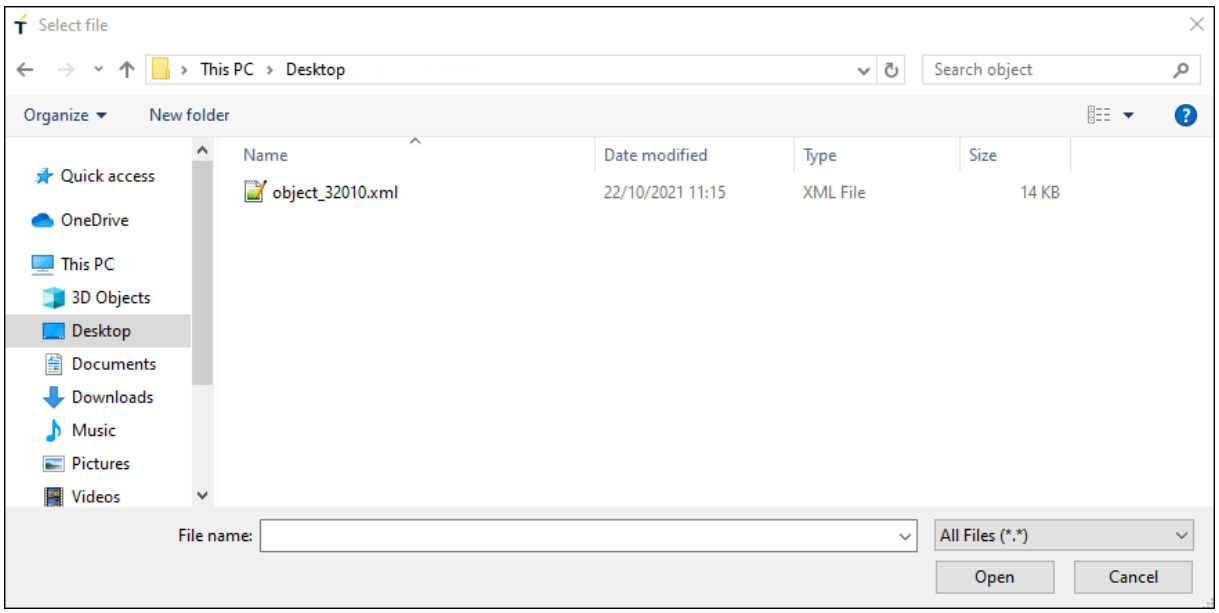

Figure 244

The file is successfully loaded on the module

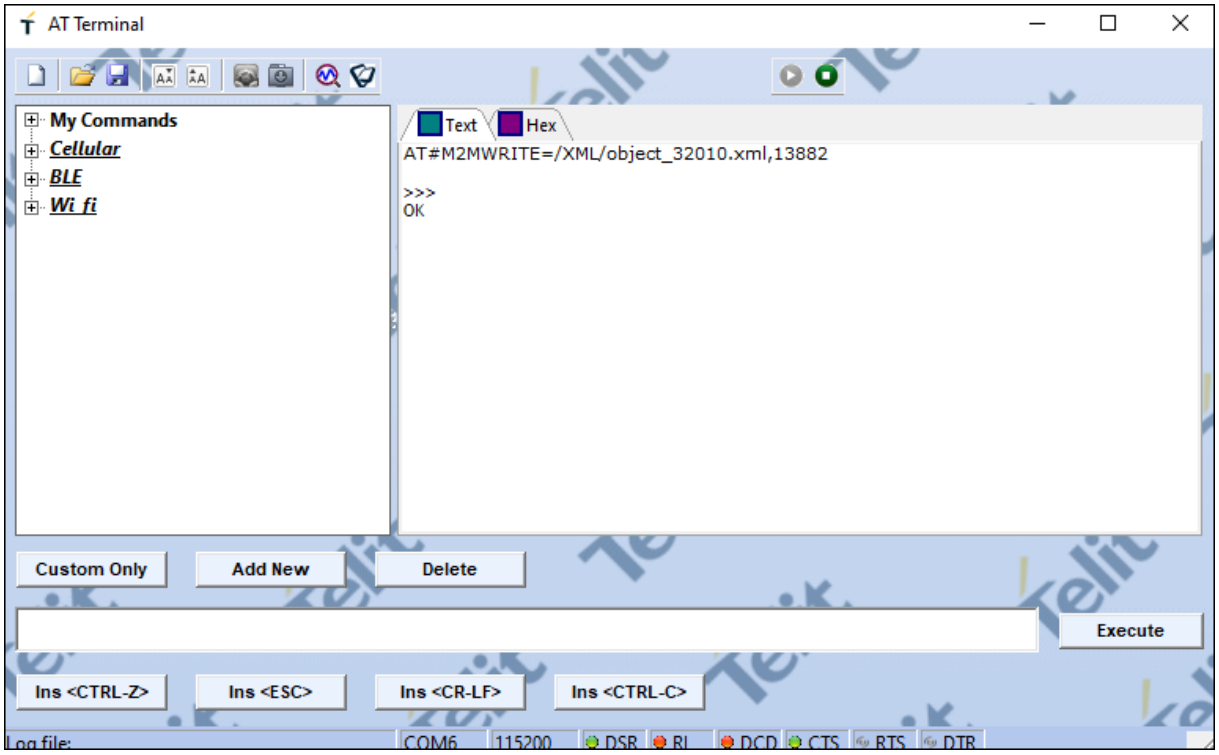

Figure 245

3.6.28.4 Onboard the device

Get the Telit ID

To retrieve the Telit ID data, issue AT#TID to get the Telit ID. The command response will be similar to

#TID: **xxxxxxxxxxxxxxxxxxxxxxxxxxxx**,1 OK

Take note of the Telit ID highlighted in **bold** above (or copy it on a text editor): this ID it will be needed for the onboarding process.

Create a new Thing

From the OneEdge portal, on **"Things"** section, click **"New Thing"** button in the top right corner.

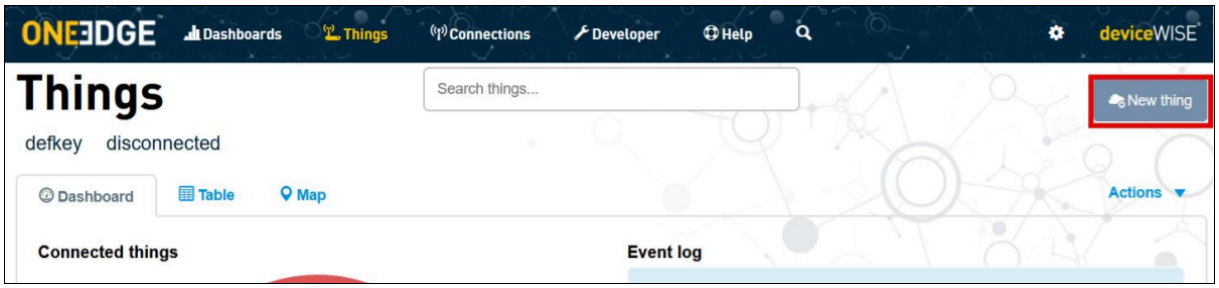

Figure 246

In the Create a new thing dialog, select “Telit Module”

Figure 247

A dialog appears: select “Default” thing definition

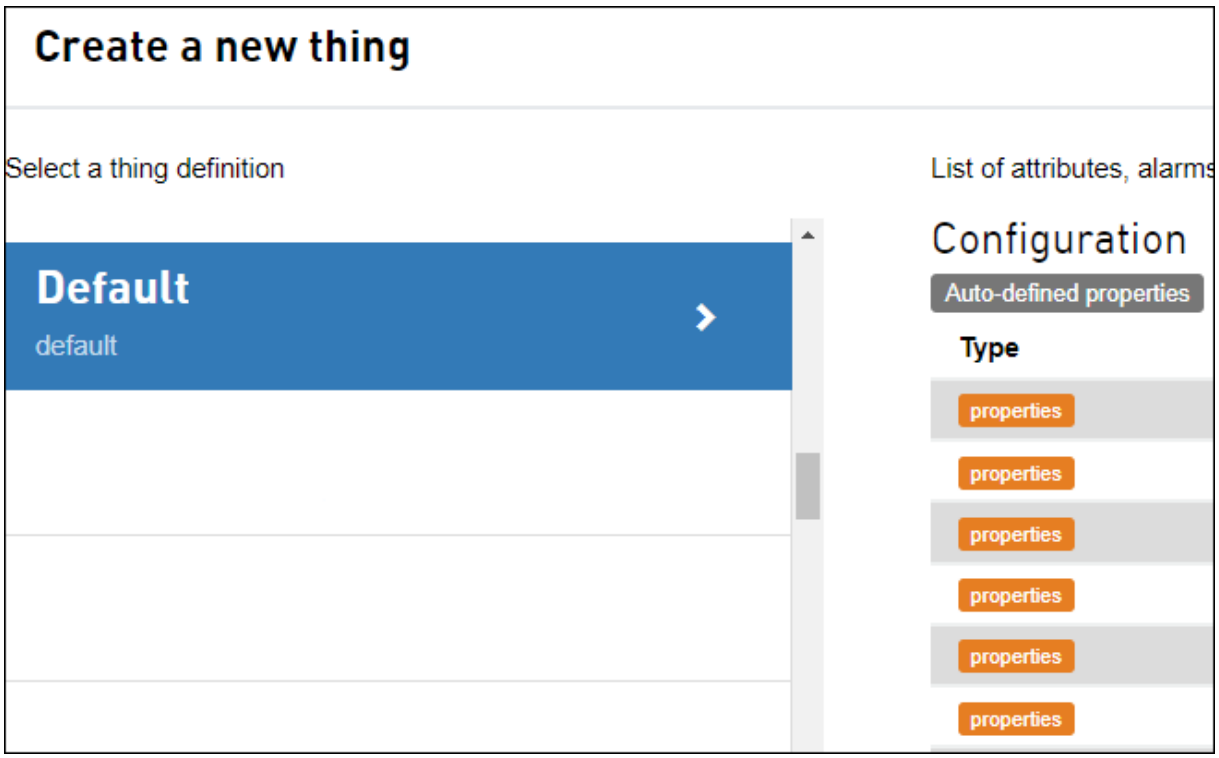

Figure 248

In the following screen, provide the Telit ID as “Identifier” Click on “Find” and make sure that model, firmware and the other details are properly populated.

Click on lwm2m tab and set the device profile previously imported as shown in the

screenshot below

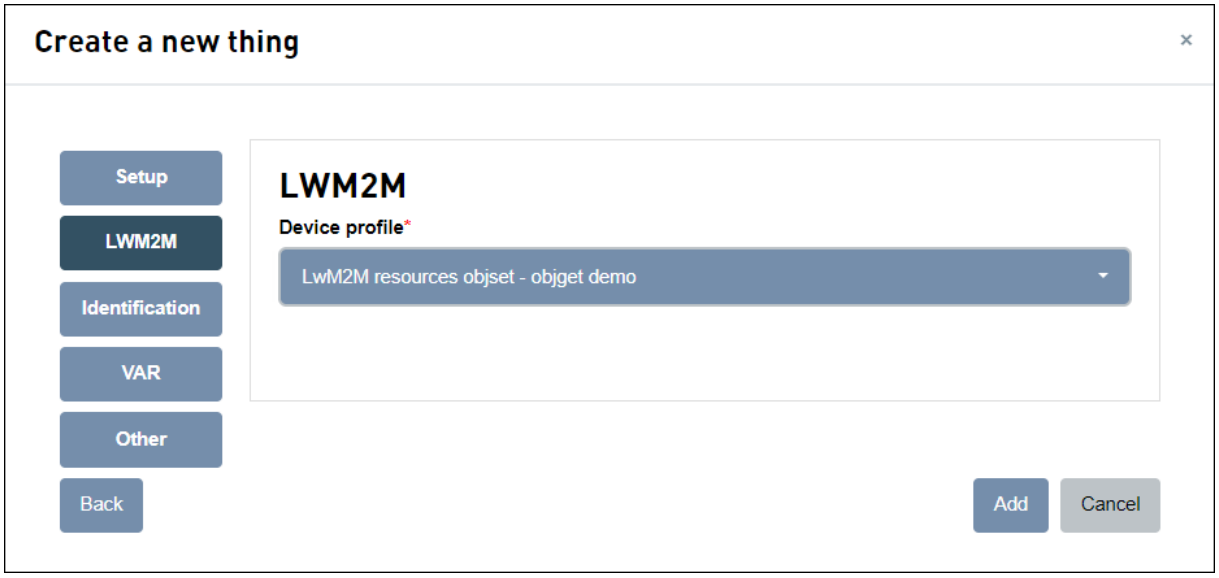

Figure 249

Click **"Add"** to complete the new thing creation procedure.

If the Thing already exists, its device profile can be changed by following the steps shown in the picture below

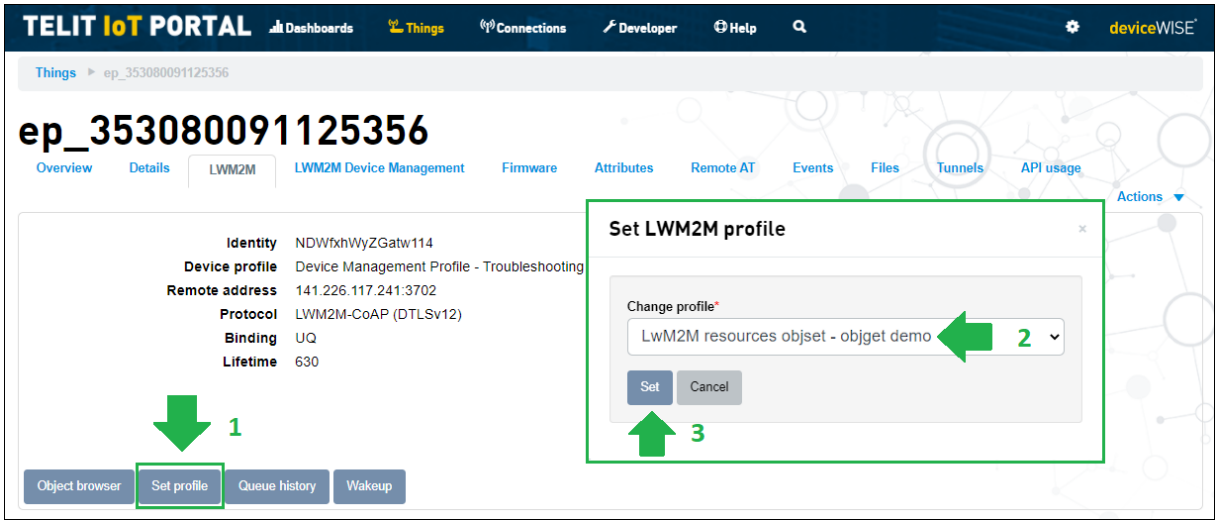

Figure 250

3.6.28.5 Application execution example

```
Starting lwm2m demo. This is v1.1.10 built on Oct 22 2021 10:49:52.
On OneEdge portal, be sure that the demo Device Profile has been uploaded on the IoT Portal
Initializing resources...

LWM2M enable result OK
[DEBUG] 38.20  lwm2m_demo:998 - lwm2mIndicationCB{pubTspt_0}$ Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED
GET STATUS.
IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED
```

Figure 251

```
Starting lwm2m demo. This is v1.1.11 built on Nov 24 2021 13:32:52.
On OneEdge portal, be sure that the 'lwm2m resources objset - objget demo' Device Profile has been uploaded on the IoT Portal

Initializing resources...
LWM2M enable result OK
[DEBUG] 29.42  lwm2m_demo:593 - lwm2mIndicationCB{pubTspt_0}$ Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED
GET STATUS.
IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERED

m2mb_lwm2m_objset SUCCESS

Resource /32011/0/11/0 changed!

Reading string resource {32011/0/11/0} value on LWM2M client.

Resource /32011/0/12/0 changed!

Resource /32011/0/13/0 changed!

Resource /32011/0/15/0 changed!

Resource /32011/0/16/0 changed!

Resource /32011/0/31/0 changed!

Resource /32011/0/31/1 changed!

Resource /32011/0/31/2 changed!

Resource /32011/0/34/0 changed!

Resource /32011/0/34/1 changed!

Resource /32011/0/34/2 changed!

Resource /32011/0/34/3 changed!
```

```
String data in {32011/0/11/0} resource was updated to new content: <hello world>

Reading integer resource {32011/0/12/0} value on LWM2M client.

Integer data in {32011/0/12/0} resource was updated to new value: 70

Reading double resource {32011/0/13/0} value  on LWM2M client.

Float data in {32011/0/13/0} resource was updated to new value: -12.500000

Reading opaque resource {32011/0/15/0} on LWM2M client.

Opaque data in {32011/0/15/0} resource was updated to new content (11 bytes):
68 65 6C 6C 6F 20 77 6F 72 6C 64

Reading timestamp resource {32011/0/16/0} value on LWM2M client.

Time data in {32011/0/16/0} resource was updated to new value: 1620828702 (2021/05/12T14:11:42+00:00)

Reading string resource {32011/0/31/0} value on LWM2M client.

String data in {32011/0/31/0} resource was updated to new content: <Greetings>

Reading string resource {32011/0/31/1} value on LWM2M client.

String data in {32011/0/31/1} resource was updated to new content: <from>

Reading string resource {32011/0/31/2} value on LWM2M client.

String data in {32011/0/31/2} resource was updated to new content: <AppZone!>

Reading boolean resource {32011/0/34/0} value on LWM2M client.

Boolean data in {32011/0/34/0} resource was updated to new value: true

Reading boolean resource {32011/0/34/1} value on LWM2M client.

Boolean data in {32011/0/34/1} resource was updated to new value: false

Reading boolean resource {32011/0/34/2} value on LWM2M client.

Boolean data in {32011/0/34/2} resource was updated to new value: true

Reading boolean resource {32011/0/34/3} value on LWM2M client.

Boolean data in {32011/0/34/3} resource was updated to new value: false
```

```
Waiting 5 seconds before the m2mb_lwm2m_objget

Waiting: 5

Waiting: 4

Waiting: 3

Waiting: 2

Waiting: 1

m2mb_lwm2m_objget SUCCESS

Content:
<{"bn":"/32011/0/",
"e":[
{"n":"11","sv":"hello world"},
{"n":"12","v":70},
{"n":"13","v":-12.500000},
{"n":"14","bv":false},
{"n":"15","sv":"aGVsbG8gd29ybGQ="},
{"n":"16","v":1620828702},
{"n":"17","ov":"0:0"},
{"n":"31/0","sv":"Greetings"},
{"n":"31/1","sv":"from"},
{"n":"31/2","sv":"AppZone!"},
{"n":"31/3","sv":""},
{"n":"31/4","sv":""},
{"n":"31/5","sv":""},
{"n":"31/6","sv":""},
{"n":"31/7","sv":""},
{"n":"31/8","sv":""},
{"n":"31/9","sv":""},
{"n":"34/0","bv":true},
{"n":"34/1","bv":false},
{"n":"34/2","bv":true},
{"n":"34/3","bv":false},
{"n":"34/4","bv":false},
{"n":"34/5","bv":false},
{"n":"34/6","bv":false},
{"n":"34/7","bv":false},
{"n":"34/8","bv":false},
{"n":"34/9","bv":false}]
}
>
```

After the Demo completes the initialization, it is possible to access the object re-

sources from the section Things: select your device, then “LwM2M” tab of OneEdge IoT portal:

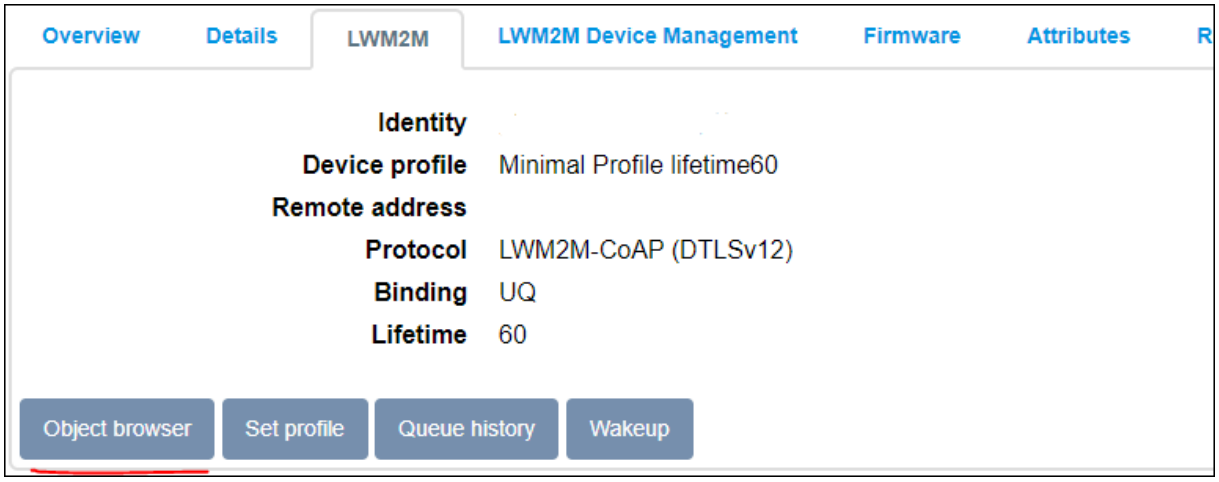

Figure 252

An instance of the object will be present and the resources can be modified.

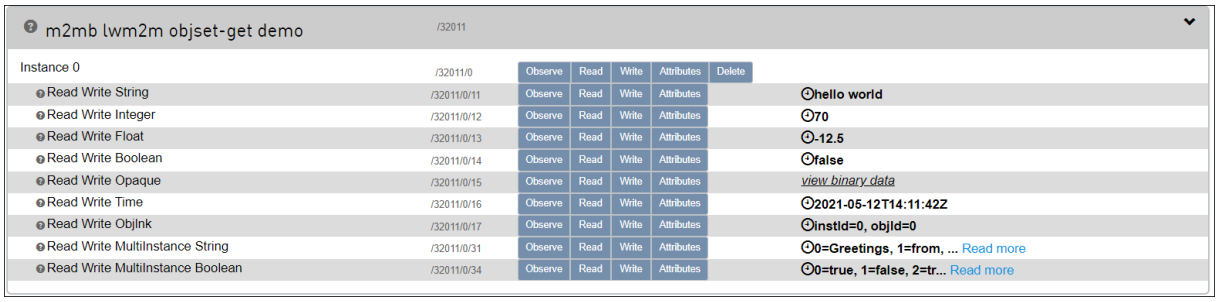

Figure 253

3.6.29 Logging Demo

Sample application showing how to print on one of the available output interfaces. Debug prints on **MAIN UART**

Features

- How to open a logging channel
- How to set a logging level
- How to use different logging macros

3.6.29.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Print welcome message
- Print a message with every log level

```
Starting Logging demo app. This is v1.0.7 built on Mar 26 2020 13:57:06.
[WARN ] 20.17  M2MB_main.c:74 - M2MB_main{M2M_DamsStart}$ This is a WARNING MESSAGE
[ERROR] 20.18  M2MB_main.c:76 - M2MB_main{M2M_DamsStart}$ THIS IS AN ERROR MESSAGE
[CRITICAL] 20.19  M2MB_main.c:78 - M2MB_main{M2M_DamsStart}$ THIS IS AN CRITICAL MESSAGE
[DEBUG] 20.19  M2MB_main.c:80 - M2MB_main{M2M_DamsStart}$ This is a DEBUG message
[TRACE] 20.20  M2MB_main.c:82 - M2MB_main{M2M_DamsStart}$ This is a TRACE message
END.
```

Figure 254

3.6.30 Low power mode

The application shows how to set the module in low power modes (by disabling UART and RF). Debug prints on **MAIN UART** which it is enabled/disabled to reach low power mode, using AZX log example functions

Features

- How to enable/disable LOG UART interfaces by azx apis
- How to enable/disable UART interfaces by m2mb apis
- How to set radio operating mode
- How to put the modem in low power mode

Application workflow

`M2MB_main.c`

- Print welcome message
- Print warning message about unplugging USB native port
- Init system events handler
- Sleep 20 seconds
- Disable RF
- Disable LOG UART
- Sleep 20 seconds
- Enable LOG UART
- Enable RF
- Sleep 60 seconds
- Deinit system events handler

```
Starting low power demo. This is v1.1.17 built on Jan 10 2023 12:27:24. LEVEL: 2
Please ensure USB native port is unplugged during low power state
Sleeping 20 seconds...
Disable LOG UART to reach low power mode and sleep for 20 seconds!
LOG UART is enabled again!
Sleeping 60 seconds...
Application end
```

Figure 255

3.6.31 MD5 example

Sample application showing how to compute MD5 hashes using m2mb crypto. Debug prints on **MAIN UART**

Features

- Compute MD5 hash of a file
- Compute MD5 hash of a string

3.6.31.1 Application workflow

M2MB_main.c

- Open USB/UART/UART_AUX
- Create a temporary file with the expected content
- Compute MD5 hash of the provided text file
- Compare the hash with the expected one
- Compute MD5 hash of a string
- Compare the hash with the expected one
- Delete test file

```
Starting MD5 demo app. This is v1.0.7 built on Apr  7 2020 10:19:54.
Buffer written successfully into file. 45 bytes were written.

Computing hash from file...
Computed hash: bb0fa6eff92c305f166803b6938dd33a
Expected hash: bb0fa6eff92c305f166803b6938dd33a
Hashes are the same!

Computing hash from string...
Computed hash: bb0fa6eff92c305f166803b6938dd33a
Expected hash: bb0fa6eff92c305f166803b6938dd33a
Hashes are the same!
```

Figure 256

3.6.32 MQTT Client

Sample application showcasing MQTT client functionalities (with SSL). Debug prints on **MAIN UART**

Features

- How to check module registration and enable PDP context
- How to configure MQTT client parameters
- How to connect to a broker with SSL and exchange data over a subscribed topic

3.6.32.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage MQTT client and start it

`mqtt_demo.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init MQTT client
- Configure it with all parameters (Client ID, username, password, PDP context ID, keepalive timeout...)
- Connect MQTT client to broker
- Subscribe to two topics
- Publish 10 messages with increasing counter. Even messages are sent to topic 1, odd messages on topic 2.
- Print received message in mqtt_topc_cb function
- Disconnect MQTT client and deinit it
- Disable PDP context

```
Starting MQTT demo app. This is v1.0.7 built on Apr  7 2020 10:34:08.
[DEBUG] 16.18  mqtt_demo.c:192 - MQTT_Task{MQTT_TASK}$ INIT
[DEBUG] 16.18  mqtt_demo.c:206 - MQTT_Task{MQTT_TASK}$ m2mb_os_ev_init success
[DEBUG] 16.19  mqtt_demo.c:214 - MQTT_Task{MQTT_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 16.19  mqtt_demo.c:221 - MQTT_Task{MQTT_TASK}$ Waiting for registration...
[DEBUG] 16.20  mqtt_demo.c:131 - NetCallback{pubTspt_0}$ Module is registered
[DEBUG] 16.21  mqtt_demo.c:232 - MQTT_Task{MQTT_TASK}$ Pdp context activation
[DEBUG] 18.26  mqtt_demo.c:246 - MQTT_Task{MQTT_TASK}$ Activate PDP with APN web.omnitel.it on CID 3....
[DEBUG] 18.95  mqtt_demo.c:155 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 18.96  mqtt_demo.c:159 - PdpCallback{pubTspt_0}$ IP address: 37.118.201.56
[DEBUG] 18.96  mqtt_demo.c:268 - MQTT_Task{MQTT_TASK}$ Init MQTT
[DEBUG] 18.97  mqtt_demo.c:278 - MQTT_Task{MQTT_TASK}$ m2mb_mqtt_init succeeded

Connecting to broker <api-dev.devicewise.com>:1883...
Done.
Subscribing to test_topic and test_topic2..
[DEBUG] 20.35  mqtt_demo.c:367 - MQTT_Task{MQTT_TASK}$ Done.

[DEBUG] 20.36  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 2> to topic test_topic
[DEBUG] 20.37  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 20.71  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 20.72  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 2>
[DEBUG] 23.37  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 3> to topic test_topic2
[DEBUG] 23.38  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 23.92  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 23.93  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 3>
[DEBUG] 26.40  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 4> to topic test_topic
[DEBUG] 26.41  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 26.93  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 26.93  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 4>
[DEBUG] 29.42  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 5> to topic test_topic2
[DEBUG] 29.43  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 29.99  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 30.00  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 5>
[DEBUG] 32.46  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 6> to topic test_topic
[DEBUG] 32.48  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 33.00  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 33.01  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 6>
[DEBUG] 35.47  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 7> to topic test_topic2
[DEBUG] 35.48  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 36.01  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 36.02  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 7>
[DEBUG] 38.50  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 8> to topic test_topic
[DEBUG] 38.51  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 39.15  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 39.16  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 8>
[DEBUG] 41.52  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 9> to topic test_topic2
[DEBUG] 41.53  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 42.10  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 42.12  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 9>
[DEBUG] 44.56  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 10> to topic test_topic
[DEBUG] 44.57  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 45.09  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 28
[DEBUG] 45.11  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 10>
[DEBUG] 47.58  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 11> to topic test_topic2
[DEBUG] 47.59  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 48.12  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 28
[DEBUG] 48.13  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 11>

Disconnecting from MQTT broker..
[DEBUG] 50.60  mqtt_demo.c:414 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 50.61  mqtt_demo.c:443 - MQTT_Task{MQTT_TASK}$ application exit
[DEBUG] 50.62  mqtt_demo.c:453 - MQTT_Task{MQTT_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 50.63  mqtt_demo.c:457 - MQTT_Task{MQTT_TASK}$ Application complete.
[DEBUG] 51.23  mqtt_demo.c:164 - PdpCallback{pubTspt_0}$ Context deactivated!
```

Figure 257

3.6.33 MultiTask

Sample application showcasing multi tasking functionalities with M2MB API. Debug prints on **MAIN UART**

Features

- How to create tasks using azx utilities
- How to use send messages to tasks
- How to use a semaphore to synchronize two tasks

3.6.33.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create three tasks with the provided utility (this calls public m2mb APIs)
- Send a message to the task1, its callback function azx_msgTask1 will be called

`azx_msgTask1`

- Print received parameters from main
- Send modified parameters to task2 (its callback function azx_msgTask2 will be called)
- wait for an InterProcess Communication semaphore to be available (released by task3)
- Once the semaphore is available, print a message and return

`azx_msgTask2`

- Print received parameters from caller
- If first parameter is bigger than a certain value, Send modified parameters to task3
- Else, use the second parameter as a task handle and print the corresponding name plus the value of the first parameter

`azx_msgTask3`

- Print received parameters from task 2
- release IPC semaphore
- send message to task 2 with first parameter below the threshold and second parameter with task3 handle

```
Starting MultiTask demo app. This is v1.0.12-C1 built on Jun 23 2020 15:36:31.

Inside "myTask1" user callback function. Received parameters from MAIN: 3 4 5
Task1 - Sending a message to task 2 with modified parameters...
Task1 - Waiting for semaphore to be released by task 3 now...

Inside "myTask2" user callback function. Received parameters: 5 7 10
Task2 - Sending a message to task 3 with modified parameters...
Task2 - Done.

Inside "myTask3" user callback function. Received parameters from Task 2: 15 14 9
Task3 - Releasing IPC semaphore...

Task1 - After semaphore! return...

Task3 - IPC semaphore released.
Task3 - Sending a message to task 2 with specific 'type' parameter value of 0 and task 3 handle as param1...

Inside "myTask2" user callback function. Received parameters: 0 1073951320 9
Task3 - Done.
Task2 - Received type 0 from task "myTask3"
Task2 - Done.
```

Figure 258

3.6.34 MutEx

Sample application showing mutex usage, with ownership and prioritization usage. Debug prints on **MAIN UART**

Features

- How to create a mutex
- How to use the mutex with tasks having different priorities
- how to reorder the pending tasks queue for the mutex

3.6.34.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create four tasks with the provided utility (this calls public m2mb APIs). The first task is a “producer”, putting data on a shared buffer. The second is a “consumer” of said data, the other two are used for prioritization demo
- run producer and consumer tasks at the same pace. the shared buffer will stay empty, because the resource is consumed right after creation
- run producer twice as fast as consumer. The buffer is slowly filled
- run consumer twice as fast as publisher. The buffer is always empty.
- reserve the mutex in the main task and run producer, support and support2 tasks (in this order). Then release the mutex and check the execution order. It should be by arrival.
- reserve the mutex in the main task and run the same three task, but before releasing the mutex, call the prioritization API. the task with highest priority (producer) is put as first in the queue.

```
Starting MutEx app. This is v1.0.12-C1 built on Jul  1 2020 08:37:15.
[DEBUG] 14.50  M2MB_main:90 - mutex_init{M2M_DamsStart}$ [MUTEX] Mutex initialized

[CASE 1 ] Producer and consumer have same idle time

[DEBUG] 14.51  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 14.52  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 14.53  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 14.53  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 14.54  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 14.54  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 14.55  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 14.56  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 15.56  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 15.56  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 15.57  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 15.58  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 15.58  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 15.59  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 15.60  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 15.60  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 16.61  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 16.61  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 16.62  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 16.63  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 16.63  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 16.64  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 16.64  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 16.65  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
```

Figure 259

```
[CASE 2 ] Producer has double idle time

[DEBUG] 17.56  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 17.56  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 17.57  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 17.58  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 17.58  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 17.59  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 17.59  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 17.60  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 18.63  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 18.64  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 0 items
[DEBUG] 18.64  M2MB_main:268 - msgConsumer{CONSUMER}$ Can't consume anything, buffer size is 0
[DEBUG] 18.65  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 19.62  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 19.62  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 19.63  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 19.64  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 19.68  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 19.69  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 19.69  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 19.70  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 20.73  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 20.74  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 0 items
[DEBUG] 20.75  M2MB_main:268 - msgConsumer{CONSUMER}$ Can't consume anything, buffer size is 0
[DEBUG] 20.75  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 21.67  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 21.68  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 21.68  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 21.69  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 21.77  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 21.79  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 21.80  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 21.80  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
```

Figure 260

```
[CASE 3 ] Producer has half idle time

[DEBUG] 22.62  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 22.63  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 22.64  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 22.64  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 22.65  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 22.65  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 22.66  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 22.67  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 23.67  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 23.68  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 23.68  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 23.69  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 24.71  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 24.72  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 24.72  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 24.73  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 24.74  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 24.74  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 24.75  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 24.76  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 25.79  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 25.79  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 1 items
[DEBUG] 25.80  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 1
[DEBUG] 25.81  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 26.78  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 26.78  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 2 items
[DEBUG] 26.79  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 1
[DEBUG] 26.79  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 26.84  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 26.84  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 1 items
[DEBUG] 26.85  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 1
[DEBUG] 26.86  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
```

Figure 261

```
[CASE 4 ] NO HTPF

Reserve MUTEX so all tasks are enqueued
[DEBUG] 30.77  M2MB_main:387 - msgSupport{HPTF_SUPPORT}$ freepos = 0 | evaluate[freepos]= 3
[DEBUG] 30.78  M2MB_main:416 - msgSupport2{HPTF_SUPPORT2}$ freepos = 1 | evaluate[freepos]= 4
[DEBUG] 30.79  M2MB_main:223 - msgProducer{PRODUCER}$ producer: freepos = 2 | evaluate[freepos]= 1
[DEBUG] 35.85  M2MB_main:586 - M2MB_main{M2M_DamsStart}$ EVALUATE SEQUENCE IS 3 4 1. Expected: 3 4 1
NO HTPF OK

[CASE 4.1 ] HTPF USED

Reserve MUTEX so all tasks are enqueued
m2mb_os_mtx_hptf OK
[DEBUG] 41.98  M2MB_main:223 - msgProducer{PRODUCER}$ producer: freepos = 0 | evaluate[freepos]= 1
[DEBUG] 41.99  M2MB_main:387 - msgSupport{HPTF_SUPPORT}$ freepos = 1 | evaluate[freepos]= 3
[DEBUG] 42.00  M2MB_main:416 - msgSupport2{HPTF_SUPPORT2}$ freepos = 2 | evaluate[freepos]= 4
[DEBUG] 44.03  M2MB_main:650 - M2MB_main{M2M_DamsStart}$ EVALUATE SEQUENCE IS 1 3 4, expected 1 3 4
HTPF DEMO OK
The application has ended...
```

Figure 262

3.6.35 NTP example

The application connects to an NTP server, gets current date and time and updates module's internal clock. Debug prints on **MAIN UART**

Features

- How to get current date and time from an NTP server
- How to set current date and time on module

3.6.35.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Send message to ntpTask

`ntp_task.c`

NTP_task() - Waits module registration - When module is registered, initializes ntp setting CID, server url and timeout - When PDP context is correctly opened, a query to NTP server is done to get current date and time - On SET_MODULE_RTC message type reception, module RTC is set with date time value got from NTP server.

m2mb_ntp_ind_callback() - As soon as M2MB_NTP_VALID_TIME event is received, current date and time is printend and a message (with SET_MODULE_RTC type) is sent to NTP_task

```
Start NTP demo application. This is v1.0 built on Apr 16 2021 09:36:12.

Waiting for registration...
Module is registered!

Activate PDP context with APN ibox.tim.it on CID 3
Context activated, IP address: 2.195.170.123
Get current time from server 0.pool.ntp.org, PORT: 123

Current time is: Friday 2021-04-16, 07:37:33

Current time correctly set on module
Module system time is: 2021-04-16, 07:37:33
```

Figure 263

3.6.36 RTC example

Sample application that shows RTC apis functionalities: how to get/set moudle system time and timestamp. Debug prints on **MAIN UART**

Features

- How to read module timestamp
- How to read module system time
- How to set new system time

3.6.36.1 Application workflow

M2MB_main.c

- Init log azx and print a welcome message
- Init net functionality and wait for module registration
- Init RTC functionality and get module time in timestamp format (seconds from the epoch)
- Get moudle system time in date/time format
- Add 1 hour to timestamp, convert it to system time and set it to module

```
 Start RTC demo application. This is v1.0 built on Oct  1 2021 15:01:40.

Waiting for registration...
Module is registered!

Current time in seconds from the epoch: 1633101266
Module system time is: 2021-10-01, 15:14:26

Get current time and add an hour

Current time in seconds from the epoch: 1633101266
New time to be set : 2021-10-01, 16:14:26, tz:4, dlst:0

Set new time and check the setting
NEW module system time is: 2021-10-01, 16:14:26
```

Figure 264

3.6.37 SIM event handler example

Sim Event Demo application. Debug prints on **MAIN UART**, using AZX log example functions

Features

- How to use ATI function for asynchronous management
- How to cath URC from an AppZone application
- How to catch SIM related events and handle them

Application workflow

`M2MB_main.c`

- Print welcome message
- Initialize AT interface
- Initialize AT URC manager task
- Initialize SIM event manager task
- Send “AT#SIMPR=1” to activate SIM URCs
- Insert SIM in SIM slot 1 and receive SIM inserted message
- Remove SIM from SIM slot 1 and receive SIM removed message

```
Starting SIM Presence Demo app. This is v1.1.17 built on Dec 16 2022 15:14:28.
Please ensure SIM is not inserted before starting this sample application
Sending command AT#SIMPR=1 to enable SIM presence URC messages...
Command set.
SIM 0 state changed to 1!

SIM CARD HAS BEEN INSERTED!

SIM 0 state changed to 0!

SIM CARD HAS BEEN REMOVED!
```

Figure 265

3.6.38 SMS PDU

Sample application showcasing how to create and decode PDUs to be used with m2mb_sms_* API set. A SIM card and antenna must be present. Debug prints on **MAIN UART**

Features

- How to enable SMS functionality
- How to use encode an SMS PDU to be sent with m2mb_api
- How to decode a received SMS response from PDU to ASCII mode.

3.6.38.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init sms functionality
- Create PDU from text message
- Send message to destination number
- Wait for response
- When SMS PDU response is received, decode it and print information about it, plus the message content

```
m2mb_sms_init() succeeded

Sending message <How are you?>...
 m2mb_sms_send() - succeeded
M2MB_SMS_SEND_RESP Callback
Send resp msg ID 10
SMS received!
SMS correctly received!

Reading SMS from memory...
 m2mb_sms_read() request succeeded

--- SMS read ---
SMS tag M2MB_SMS_TAG_MT_NOT_READ
SMS format M2MB_SMS_FORMAT_3GPP
Code type: 0
Sender type: 145
Msg len: 12
Msg bytes: 11
Msg date 19/7/17 16:7:58 (timezone: 2)
Received SMS, content: <<Fine thanks >>
Sender: +[redacted]
```

Figure 266

3.6.39 SMS_atCmd example

Sample application showcasing how to receive an SMS containing an AT command, process the AT command and send its answer to sender (configurable in sms_config.txt). A SIM card and antenna must be present. Debug prints on **MAIN UART**

Features

- How to receive an SMS with an AT command as text inside
- How to send AT command to parser and read the answer
- How to send the AT command answer back to sender via SMS

Optional configuration file to be put in `/mod` folder, copy `sms_config.txt` file into your module running the following AT command:

```
AT#M2MWRITE="/mod/sms_config.txt",138
>>> here receive the prompt; then type or send the file, sized 138 bytes
```

3.6.39.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Init SMS functionality
- Read configuration file sms_config.txt (send SMS with AT command answer back, delte SMS received)
- Init AT command parser
- Create a task to handle SMS parsing and AT command sending
- Wait for an incoming SMS

`callbacks.c`

`msgSMSparse()`

- When SMS has been received, content is decoded and printed. If there is an AT command inside, command is executed and answer printed and sent back to sender as an SMS (depending on sms_config.txt setting)

```
ÿStarting SMS with AT command demo app. This is v1.0.13-C1 built on Mar 18 2021 12:42:22.
[DEBUG] 16.61  M2MB_main:135 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
m2mb_sms_init() succeeded
[DEBUG] 16.62  M2MB_main:168 - M2MB_main{M2M_DamsStart}$ M2MB_SMS_INCOMING_IND indication enabled
[DEBUG] 16.63  M2MB_main:179 - M2MB_main{M2M_DamsStart}$ M2MB_SMS_INCOMING_IND MEMORY FULL indication enabled
[DEBUG] 16.64  M2MB_main:196 - M2MB_main{M2M_DamsStart}$ Storage set to M2MB_SMS_STORAGE_SM
[DEBUG] 16.65  callbacks:114 - readConfigFromFile{M2M_DamsStart}$ Reading parameters from file
[DEBUG] 16.66  callbacks:116 - readConfigFromFile{M2M_DamsStart}$ Opening /mod/sms_config.txt in read mode..
Default: SMS with answer sending DISABLED, delete sms DISABLED
[DEBUG] 16.67  at_async:115 - at_cmd_async_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Please send an SMS with a configuration as ("ATCMD: <atcmd>")...
```

Figure 267

3.6.40 SMTP Client

Sample application showing SMTP echo demo with M2MB API. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to open a SMTP client
- How to send a mail

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage SMTP client and start it

`M2MB_main.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Initialize SMTP client and connect to SMTP server
- Prepare email and send it
- Close SMTP client
- Disable PDP context

```
Starting SMTP demo app. This is v1.1.17 built on Jan  9 2023 09:57:51.
Reading parameters from file...
Set APN to: <<nxt17.net>>
Set APN USER to: <<>>
Set APN PASS to: <<>>
Set PDP_CTX to: 1
Set MAIL SERVER to: <<           >>
Set MAIL PORT to: <<     >>
Set MAIL USER to: <<                            >>
Set MAIL PASS to: <<                    >>
Set MAIL FROM to: <<                            >>
Set MAIL FROM NAME to: <<   >>
Set MAIL SUBJECT to: <<Hello from AppZone>>
Set MAIL BODY to: <<AppZone Test>>
Set MAIL TO to: <<                           >>
Set MAIL TO NAME to: <<  >>
Waiting for registration...
Pdp context activation
Connecting to SMTP Server...
SMTP Client connected!
Preparing example's mail...
Sending example's mail...
Example's mail sent!
SMTP client closed
Application complete.
```

Figure 268

3.6.41 SPI Echo

Sample application showing how to communicate over SPI with m2mb API. Debug prints on **MAIN UART**

Features

- How to open an SPI bus. MOSI and MISO will be shorted, to have an echo.
- How to communicate over SPI bus

3.6.41.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open SPI bus, set parameters
- Send data on MOSI and read the same in MISO

Notes:

For LE910Cx (both Linux and ThreadX based devices), `AT#SPIEN=1` command must be sent once before running the app

```
Starting SPI demo app. This is v1.0.7 built on Apr  1 2020 13:48:05.
Transfer successful. Received: hello from spi echo
```

Figure 269

3.6.42 SPI sensors

Sample application showing SPI usage, configuring two ST devices: a magnetometer (ST LIS3MDL) and a gyroscope (ST L3G4200D). The application will read values from both devices using GPIO4 and 3 (respectively) as magnetometer CS and gyro CS. Debug prints on **MAIN UART**

Features

- How to open an SPI bus with a slave device
- How to communicate with the device over the SPI bus

Setup

- Connect sensor VDD to 3v8 supply (e.g. Vbatt on the module)
- Connect sensor GND to a GND pin of the module
- Connect sensors MOSI to module SPI_MOSI
- Connect sensors MISO to module SPI_MISO
- Connect sensors CLK to module SPI_CLK
- Connect magnetometer CS to module GPIO 2
- Connect gyroscope CS to module GPIO 3

3.6.42.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open SPI bus, set parameters
- Configure `GPIO 2` and `GPIO 3` as output, set them high (idle)
- Set registers to configure magnetometer
- Read in a loop (10 iterations) the registers carrying the 3 axes values and show the gauss value for each of them. A metal object is put close to the sensor to change the read values.
- Set registers to configure gyroscope
- Read in a loop (10 iterations) the registers carrying the 3 axes values and show the degrees per second value for each of them. The board is rotated to change the read values.

Notes:

For LE910Cx (both Linux and ThreadX based devices), `AT#SPIEN=1` command must be sent once before running the app

```
Starting SPI demo app. This is v1.0.7 built on Apr  1 2020 13:58:25.
SPI start

Magnetometer SPI Demo start
Reading Magnetometer WHOAMI. Expected: 0x3D
Expected response received!
Setting continuous conversion mode...
Continuous conversion mode successfully set.
Setting 10 Hz Output Data Rate, Medium performance mode X Y axis...
Magnetometer Enabled. 10Hz ODR, Medium Perf. Mode (X,Y).
Setting Medium performance for Z axis, little endian...
Medium Perf. Mode (Z), little endian.
Setting complete, starting reading loop...

X: 0.204 gauss
Y: -0.321 gauss
Z: 0.305 gauss

X: 0.290 gauss
Y: -0.103 gauss
Z: 0.043 gauss

X: -2.513 gauss
Y: -0.353 gauss
Z: -4.000 gauss

X: 1.980 gauss
Y: 0.174 gauss
Z: -1.945 gauss

X: 4.000 gauss
Y: -0.090 gauss
Z: -4.000 gauss

X: -0.605 gauss
Y: -0.154 gauss
Z: 0.210 gauss

X: -0.580 gauss
Y: 2.004 gauss
Z: -0.047 gauss

X: 0.177 gauss
Y: -0.359 gauss
Z: 0.295 gauss

X: 0.173 gauss
Y: -0.356 gauss
Z: 0.301 gauss

X: 0.174 gauss
Y: -0.356 gauss
Z: 0.298 gauss
Reading complete.
```

Figure 270

3.6.43 SW Timer (Software Timer)

The sample application shows how to use SW Timers M2MB API. Debug prints on **MAIN UART**

Features

- How to open configure a SW timer
- How to use the timer to manage recurring events

3.6.43.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create sw timer structure
- Configure it with 4 seconds timeout, periodic timer (auto fires when expires)
- Init the timer with the parameters
- Start the timer
- Wait 10 seconds
- Stop the timer

`timerCb`

- Print a message with inside the callback

```
Starting SW Timers demo app. This is v1.0.7 built on Apr  7 2020 09:51:25.
timer expired!
[DEBUG] 21.41  M2MB_main.c:59 - timerCb{pubTspt_0}$ timer handle: 0x4002b004
timer expired!
[DEBUG] 25.47  M2MB_main.c:59 - timerCb{pubTspt_0}$ timer handle: 0x4002b004
stopping the timer
Stop a running timer: success
Application end
```

Figure 271

3.6.44 Secure MicroService

Sample application showcasing how to manage secure microservice functionalities. Debug prints on **MAIN UART**

Features

- Write data in Secure Data Area (SDA), non protected
- Read the written data and compare with the original buffer
- Write a cripty key in Secure Data Area (SDA), non protected
- Perform a rotate of the written key data
- Perform MD5 sum of written data from TZ file
- Compare computed digest with expected one
- Write data in trust zone as a trusted object (it will not be possible to read it again but only use its content for crypto operations)
- Try to read the trusted object and verify it fails
- Rotate trusted item and verify retrieving the content fails
- compute MD5 sum of trusted item and compare with the expected one
- Try to pass data from a trusted item to a non trusted item using untrusted TZ buffers, and verify it fails

3.6.44.1 Application workflow

`M2MB_main.c`

- Write a buffer in a SDA item using `m2mb_secure_ms_write`
- Read the same item using `m2mb_secure_ms_read`
- Write a buffer containing some cripty key in a SDA item using `m2mb_secure_ms_write`
- Rotate the content of the key item
- Read it with `m2mb_secure_ms_read`
- Load the key content using `m2mb_secure_ms_crypto_alloc` and `m2mb_secure_crypto_add_it` in a SECURE_MS buffer
- Compute MD digest with `m2mb_secure_ms_crypto_md`
- Write a buffer containing some cripty key in a SDA item using `m2mb_secure_ms_write` but with **TRUSTED** option in `m2mb_secure_ms_open`
- Verify that `m2mb_secure_ms_read` on the trusted item fails
- Verify that `m2mb_secure_ms_crypto_rotate` fails for the trusted item
- Verify the MD5 digest
- Try to copy the trusted item data in a SECURE_MS buffer with `m2mb_secure_ms_crypto_alloc` and `m2mb_secure_crypto_add_item`, then load it in an untrusted object with `m2mb_secure_ms_crypto_write`, and verify it fails.

```
Starting secure ms demo app. This is v1.0.13-C1 built on Jul 30 2020 12:19:02.
Writing data in normal item
Stored input data in Secure Data Area
Reading data from normal item
Data length in SDA: 11 bytes
Securely loaded the data from the SDA
Read 11 bytes: <hello world>
original and retrieved strings are the same

Writing key in normal item
Stored input data in Secure Data Area

Rotate data in normal item
Original key: AA_THIS_IS_MY_SECRET_KEY_BB
Rotated key:

Compute MD5 of data in normal item
Data length in SDA: 27 bytes
MD5: 8EDAD26E26E1C74C7C02386C1C7F541D
hash is the expected one!

Writing data in trusted item
Stored input data in Secure Data Area
Reading data from trusted item (should fail!)
Data length in SDA: 27 bytes
m2mb_secure_ms_read() failed for trusted item, as expected!

Rotate data in trusted item
[ERROR] 17.01  M2MB_main:329 - read_rotate{M2M_DamsStart}$ Cannot read data from SECURE_MS_BUFFER to user buffer
Original key: AA_THIS_IS_MY_SECRET_KEY_BB
Rotated key:

Compute MD5 of data in trusted item
Data length in SDA: 27 bytes

MD5: 8EDAD26E26E1C74C7C02386C1C7F541D
Hash is the expected one!

Try to pass data from trusted to untrusted through TZ buffers
Cannot store data from SECURE_MS_BUFFER to SDA 'non-trusted', as expected
```

Figure 272

3.6.45 TCP IP

Sample application showcasing TCP echo demo with M2MB API. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to open a TCP client socket
- How to communicate over the socket

3.6.45.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server
- Send data and receive response
- Close socket
- Disable PDP context

```
Starting TCP-IP demo app. This is v1.0.7 built on Mar 26 2020 16:20:30.
[DEBUG] 21.23  m2m_tcp_test.c:201 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 21.25  m2m_tcp_test.c:217 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.26  m2m_tcp_test.c:223 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.26  m2m_tcp_test.c:231 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 21.28  m2m_tcp_test.c:128 - NetCallback{pubTspt_0}$ Module is registered to cell 0x816B!
[DEBUG] 21.29  m2m_tcp_test.c:244 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 21.30  m2m_tcp_test.c:248 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.34  m2m_tcp_test.c:263 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 24.52  m2m_tcp_test.c:155 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.52  m2m_tcp_test.c:158 - PdpCallback{pubTspt_0}$ IP address: 83.225.44.56
[DEBUG] 24.54  m2m_tcp_test.c:273 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 24.54  m2m_tcp_test.c:284 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 24.55  m2m_tcp_test.c:294 - M2M_msgTCPTask{TCP_TASK}$ Socket ctx set to 3
[DEBUG] 24.95  m2m_tcp_test.c:307 - M2M_msgTCPTask{TCP_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 25.17  m2m_tcp_test.c:322 - M2M_msgTCPTask{TCP_TASK}$ Socket Connected!
[DEBUG] 25.18  m2m_tcp_test.c:329 - M2M_msgTCPTask{TCP_TASK}$ Sending data over socket..
[DEBUG] 25.19  m2m_tcp_test.c:342 - M2M_msgTCPTask{TCP_TASK}$ Data send successfully (16 bytes)
[DEBUG] 27.20  m2m_tcp_test.c:356 - M2M_msgTCPTask{TCP_TASK}$ trying to receive 16 bytes..
[DEBUG] 27.21  m2m_tcp_test.c:364 - M2M_msgTCPTask{TCP_TASK}$ Data received (16): <hello from m2mb!>
[DEBUG] 27.21  m2m_tcp_test.c:373 - M2M_msgTCPTask{TCP_TASK}$ application exit
[DEBUG] 27.22  m2m_tcp_test.c:385 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 27.24  m2m_tcp_test.c:388 - M2M_msgTCPTask{TCP_TASK}$ Application complete.
[DEBUG] 29.43  m2m_tcp_test.c:164 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 273

3.6.46 TCP non blocking example

Sample application that shows how to configure and connect a TCP-IP non blocking socket. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to open a TCP client non Blocking socket
- How to communicate over the socket

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Set the socket as non Blocking and connect to server. Uses m2mb_socket_bsd_select, m2mb_socket_bsd_fd_isset_func to check when socket is connected.
- Send data and receive response
- Close socket
- Disable PDP context

```
Starting TCP-IP non blocking demo app. This is v1.1.9 built on Mar 22 2022 15:35:56.
[DEBUG] 19.24  m2mb_tcp_te:215 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 19.25  m2mb_tcp_te:231 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 19.25  m2mb_tcp_te:237 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 19.26  m2mb_tcp_te:245 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 19.27  m2mb_tcp_te:142 - NetCallback{pubTspt_0}$ Module is registered to cell 0x20!
[DEBUG] 19.28  m2mb_tcp_te:258 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 19.29  m2mb_tcp_te:262 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.34  m2mb_tcp_te:277 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN ibox.tim.it....
[DEBUG] 21.89  m2mb_tcp_te:169 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 21.90  m2mb_tcp_te:172 - PdpCallback{pubTspt_0}$ IP address: 2.195.160.174
[DEBUG] 21.90  m2mb_tcp_te:287 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 21.91  m2mb_tcp_te:299 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 21.92  m2mb_tcp_te:309 - M2M_msgTCPTask{TCP_TASK}$ Socket ctx set to 1
[DEBUG] 22.10  m2mb_tcp_te:322 - M2M_msgTCPTask{TCP_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 22.11  m2mb_tcp_te:332 - M2M_msgTCPTask{TCP_TASK}$ Socket set to nonBlocking
[DEBUG] 22.11  m2mb_tcp_te:351 - M2M_msgTCPTask{TCP_TASK}$ Connection in progress...
[DEBUG] 22.29  m2mb_tcp_te:377 - M2M_msgTCPTask{TCP_TASK}$ Socket <0x400343a0> is connected!
[DEBUG] 22.30  m2mb_tcp_te:398 - M2M_msgTCPTask{TCP_TASK}$ Socket Connected!
[DEBUG] 22.32  m2mb_tcp_te:411 - M2M_msgTCPTask{TCP_TASK}$ Sending data over socket..
[DEBUG] 22.32  m2mb_tcp_te:424 - M2M_msgTCPTask{TCP_TASK}$ Data send successfully (16 bytes)
[DEBUG] 24.37  m2mb_tcp_te:438 - M2M_msgTCPTask{TCP_TASK}$ trying to receive 16 bytes..
[DEBUG] 24.38  m2mb_tcp_te:446 - M2M_msgTCPTask{TCP_TASK}$ Data received (16): <hello from m2mb!>
[DEBUG] 24.39  m2mb_tcp_te:456 - M2M_msgTCPTask{TCP_TASK}$ application exit
[DEBUG] 24.43  m2mb_tcp_te:468 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 24.45  m2mb_tcp_te:471 - M2M_msgTCPTask{TCP_TASK}$ Application complete.
[DEBUG] 24.79  m2mb_tcp_te:178 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 274

3.6.47 TCP Socket status

Sample application showcasing how to check a TPC connected socket current status. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to open a TCP client socket
- How to check if the TCP socket is still valid

3.6.47.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server
- Check in a loop the current socket status using the adv_select function with a 2 seconds timeout
- Close socket when the remote host closes it
- Disable PDP context

```
Starting TCP socket status check demo app. This is v1.0.14-C1 built on Sep  8 2020 14:59:25.
[DEBUG] 21.33  m2m_tcp_tes:324 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 21.34  m2m_tcp_tes:338 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.34  m2m_tcp_tes:344 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.35  m2m_tcp_tes:352 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 21.36  m2m_tcp_tes:365 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 21.37  m2m_tcp_tes:369 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.41  m2m_tcp_tes:384 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN NXT17.NET....
[DEBUG] 24.09  m2m_tcp_tes:281 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.10  m2m_tcp_tes:284 - PdpCallback{pubTspt_0}$ IP address: 100.77.5.223
[DEBUG] 24.10  m2m_tcp_tes:394 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 24.11  m2m_tcp_tes:405 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 24.11  m2m_tcp_tes:415 - M2M_msgTCPTask{TCP_TASK}$ Socket ctx set to 3
[DEBUG] 24.60  m2m_tcp_tes:428 - M2M_msgTCPTask{TCP_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 24.93  m2m_tcp_tes:443 - M2M_msgTCPTask{TCP_TASK}$ Socket Connected!
[DEBUG] 26.98  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 29.03  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...

...

[DEBUG] 82.18  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 84.23  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 86.28  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 88.31  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 88.90  m2m_tcp_tes:154 - adv_select{TCP_TASK}$ Data is available on socket <0x40032b3c>
[DEBUG] 88.92  m2m_tcp_tes:160 - adv_select{TCP_TASK}$ There are <0> pending bytes on socket
Socket was closed by remote!
[DEBUG] 88.92  m2m_tcp_tes:494 - M2M_msgTCPTask{TCP_TASK}$ application exit
[DEBUG] 88.94  m2m_tcp_tes:506 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 88.94  m2m_tcp_tes:509 - M2M_msgTCPTask{TCP_TASK}$ Application complete.
[DEBUG] 89.31  m2m_tcp_tes:290 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 275

3.6.48 TCP Server

Sample application showcasing TCP listening socket demo with M2MB API. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to open a TCP listening socket
- How to manage external hosts connection and exchange data

3.6.48.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and set it in non-blocking mode
- Bind the socket to the listening port
- Start listening for incoming connection
- Check if a connection is incoming using m2mb_socket_bsd_select function
- If a client connects, perform accept on the child socket
- Send a “START” message to the client
- Send some data
- Wait for data from client and print it
- Close the child socket
- Start listening again, up to 3 times
- Close listening socket
- Disable PDP context

Debug Log

```
Starting TCP Server demo app. This is v1.0.7 built on Apr  7 2020 13:28:24.
[DEBUG] 14.55  m2m_tcp_test.c:220 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 14.55  m2m_tcp_test.c:236 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 14.57  m2m_tcp_test.c:242 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 14.57  m2m_tcp_test.c:250 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 14.58  m2m_tcp_test.c:138 - NetCallback{pubTspt_0}$ Module is registered to cell 0x5222!
[DEBUG] 14.59  m2m_tcp_test.c:263 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 14.60  m2m_tcp_test.c:267 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 16.57  m2m_tcp_test.c:282 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN ibox.tim.it....
[DEBUG] 17.16  m2m_tcp_test.c:165 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 17.17  m2m_tcp_test.c:168 - PdpCallback{pubTspt_0}$ IP address: 2.195.165.137


--------------------------------------------
|                                          |
|  Start TCP server                        |
|                                          |
--------------------------------------------


[DEBUG] 19.15  m2m_tcp_test.c:301 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 19.15  m2m_tcp_test.c:312 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 19.16  m2m_tcp_test.c:313 - M2M_msgTCPTask{TCP_TASK}$ m2mb_socket_bsd_socket(): valid socket ID [0x4002E79C] - PASS
[DEBUG] 20.16  m2m_tcp_test.c:319 - M2M_msgTCPTask{TCP_TASK}$ issuing m2m_socket_bsd_ioctl() to set non-blocking mode ...
[DEBUG] 20.17  m2m_tcp_test.c:331 - M2M_msgTCPTask{TCP_TASK}$ Binding Socket...
[DEBUG] 22.12  m2m_tcp_test.c:343 - M2M_msgTCPTask{TCP_TASK}$ Socket Bind Pass


Start TCP listening on port 6500...

[DEBUG] 24.13  m2m_tcp_test.c:368 - M2M_msgTCPTask{TCP_TASK}$ select...
Select result: 0
[DEBUG] 28.13  m2m_tcp_test.c:368 - M2M_msgTCPTask{TCP_TASK}$ select...
Select result: 1

TCP Server Coming Connection

--> Accept
[DEBUG] 30.52  m2m_tcp_test.c:397 - M2M_msgTCPTask{TCP_TASK}$ Socket Accept Pass


Connected! (socket dial n.1)

[DEBUG] 30.53  m2m_tcp_test.c:403 - M2M_msgTCPTask{TCP_TASK}$ Client Source Address: 185.86.42.254
[DEBUG] 30.54  m2m_tcp_test.c:404 - M2M_msgTCPTask{TCP_TASK}$ Client Port: 58658
[DEBUG] 30.54  m2m_tcp_test.c:405 - M2M_msgTCPTask{TCP_TASK}$ Client Family: 2
[DEBUG] 31.56  m2m_tcp_test.c:410 - M2M_msgTCPTask{TCP_TASK}$
-------------------------------
[DEBUG] 31.57  m2m_tcp_test.c:411 - M2M_msgTCPTask{TCP_TASK}$ |    Send/receive data test    |
[DEBUG] 31.57  m2m_tcp_test.c:412 - M2M_msgTCPTask{TCP_TASK}$ -------------------------------

[DEBUG] 32.58  m2m_tcp_test.c:416 - M2M_msgTCPTask{TCP_TASK}$
--> issuing m2mb_socket_bsd_send(): transmit "__START____" packet...
[DEBUG] 32.59  m2m_tcp_test.c:423 - M2M_msgTCPTask{TCP_TASK}$  --> done (11 have been transmitted)
[DEBUG] 32.60  m2m_tcp_test.c:425 - M2M_msgTCPTask{TCP_TASK}$      ALL data transmitted - PASS
[DEBUG] 32.61  m2m_tcp_test.c:430 - M2M_msgTCPTask{TCP_TASK}$
--> issuing m2mb_socket_bsd_send(): transmit 58 bytes...
[DEBUG] 32.62  m2m_tcp_test.c:437 - M2M_msgTCPTask{TCP_TASK}$  --> done (58 have been transmitted)
[DEBUG] 32.63  m2m_tcp_test.c:440 - M2M_msgTCPTask{TCP_TASK}$      ALL data transmitted - PASS
[DEBUG] 32.64  m2m_tcp_test.c:448 - M2M_msgTCPTask{TCP_TASK}$
Waiting for data...

[DEBUG] 39.64  m2m_tcp_test.c:457 - M2M_msgTCPTask{TCP_TASK}$ test
[DEBUG] 99.61  m2m_tcp_test.c:465 - M2M_msgTCPTask{TCP_TASK}$
m2mb_socket_bsd_recv() has received 6 bytes

[DEBUG] 102.60  m2m_tcp_test.c:469 - M2M_msgTCPTask{TCP_TASK}$
Server TCP is closing the current connection ...
```

Figure 276

Data on a PuTTY terminal

```
START

aaaaaaaaaa-bbbbbbbbbb-cccccccccc-dddddddddd-eeeeeeeeee

test
```

Figure 277

3.6.49 TLS SSL Client

Sample application showcasing TLS/SSL with client certificates usage with M2MB API. Debug prints on **MAIN UART**

Features

- How to check module registration and enable PDP context
- How to open a SSL client socket
- How to communicate over SSL socket

3.6.49.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the connection and start it

`ssl_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server over TCP socket
- Initialize the TLS parameters (TLS1.2) andh auth mode (server+client auth in the example)
- Create SSL context
- Read certificates files and store them
- Create secure socket and connect to the server using SSL
- Send data and receive response
- Close secure socket
- Close socket
- Delete SSL context
- Disable PDP context

The application requires the certificates to be stored in `/mod/ssl_certs/` folder. It can be created with

```
AT#M2MMKDIR=/mod/ssl_certs
```

Certificates can then be loaded with

AT#M2MWRITE="/mod/ssl_certs/modulesCA.crt",1740

and providing the file content in RAW mode (for example using the “Transfer Data” button in Telit AT Controller)

For client certificates (if required), the commands will be

AT#M2MWRITE="/mod/ssl_certs/modulesClient.crt",1651
AT#M2MWRITE="/mod/ssl_certs/modulesClient_pkcs1.key",1679

PLEASE NOTE: always verify the file sizes to be used in the commands above as they might change

```
Starting TLS-SSL demo app. This is v1.1.2 built on Mar  3 2021 10:15:00.
[DEBUG] 10.85  ssl_test:252 - msgHTTPSTask{TLS_TASK}$ INIT
[DEBUG] 10.85  ssl_test:266 - msgHTTPSTask{TLS_TASK}$ m2mb_os_ev_init success
[DEBUG] 10.85  ssl_test:270 - msgHTTPSTask{TLS_TASK}$ Init SSL session test app
[DEBUG] 10.85  ssl_test:285 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config sslConfigHndl = 0x40037958, sslRes= 0
[DEBUG] 10.85  ssl_test:294 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config PASSED
[DEBUG] 10.85  ssl_test:306 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_ctxt PASSED
[DEBUG] 10.85  ssl_test:311 - msgHTTPSTask{TLS_TASK}$ loading CA CERT from file /mod/ssl_certs/modulesCA.crt
[DEBUG] 10.85  ssl_test:315 - msgHTTPSTask{TLS_TASK}$ file size: 1740
[DEBUG] 10.85  ssl_test:328 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1740
Buffer successfully received from file. 1740 bytes were loaded.
Closing file.
[DEBUG] 10.85  ssl_test:361 - msgHTTPSTask{TLS_TASK}$ loading client CERT from file /mod/ssl_certs/modulesClient.crt
[DEBUG] 10.85  ssl_test:365 - msgHTTPSTask{TLS_TASK}$ file size: 1651
[DEBUG] 10.85  ssl_test:378 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1651
Buffer successfully received from file. 1651 bytes were loaded.
Closing file.
[DEBUG] 10.85  ssl_test:401 - msgHTTPSTask{TLS_TASK}$ loading client KEY from file /mod/ssl_certs/modulesClient_pkcs1.key
[DEBUG] 10.85  ssl_test:405 - msgHTTPSTask{TLS_TASK}$ file size: 1679
[DEBUG] 10.85  ssl_test:418 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1679
Buffer successfully received from file. 1679 bytes were loaded.
Closing file.
[DEBUG] 10.85  ssl_test:448 - msgHTTPSTask{TLS_TASK}$ certificates successfully stored!
[DEBUG] 10.85  ssl_test:457 - msgHTTPSTask{TLS_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 10.85  ssl_test:465 - msgHTTPSTask{TLS_TASK}$ Waiting for registration...
[DEBUG] 10.86  ssl_test:171 - NetCallback{pubTspt_0}$ Module is registered to cell 0x468E!
[DEBUG] 10.86  ssl_test:477 - msgHTTPSTask{TLS_TASK}$ Pdp context activation
[DEBUG] 10.86  ssl_test:481 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 12.87  ssl_test:496 - msgHTTPSTask{TLS_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 13.71  ssl_test:197 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 13.71  ssl_test:200 - PdpCallback{pubTspt_0}$ IP address: 2.41.76.63
[DEBUG] 13.71  ssl_test:514 - msgHTTPSTask{TLS_TASK}$ Creating Socket...
[DEBUG] 13.71  ssl_test:525 - msgHTTPSTask{TLS_TASK}$ Socket created
[DEBUG] 13.71  ssl_test:535 - msgHTTPSTask{TLS_TASK}$ Socket ctx set to 3
[DEBUG] 13.92  ssl_test:548 - msgHTTPSTask{TLS_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 14.05  ssl_test:562 - msgHTTPSTask{TLS_TASK}$ Socket Connected!
[DEBUG] 15.97  ssl_test:587 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_connect ret 0
[DEBUG] 17.99  ssl_test:593 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
```

```
[DEBUG] 17.99  ssl_test:593 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
[DEBUG] 17.99  ssl_test:596 - msgHTTPSTask{TLS_TASK}$ SSL write result = 44
[DEBUG] 22.03  ssl_test:608 - msgHTTPSTask{TLS_TASK}$ pending bytes: 1087
[DEBUG] 22.03  ssl_test:612 - msgHTTPSTask{TLS_TASK}$ trying to receive 1087 bytes..
[DEBUG] 22.03  ssl_test:618 - msgHTTPSTask{TLS_TASK}$ Server response: (269)<HTTP/1.1 200 OK
Date: Wed, 03 Mar 2021 09:18:22 GMT
Server: Apache/2.2.15 (CentOS)
Last-Modified: Mon, 22 Jan 2018 10:57:39 GMT
ETag: "1fffc-27f-5635b4c6f12b3"
Accept-Ranges: bytes
Content-Length: 639
Connection: close
Content-Type: text/html; charset=UTF-8

>
[DEBUG] 22.03  ssl_test:634 - msgHTTPSTask{TLS_TASK}$ pending bytes: 762
[DEBUG] 22.03  ssl_test:638 - msgHTTPSTask{TLS_TASK}$ trying to receive remaining 762 bytes..
[DEBUG] 22.03  ssl_test:644 - msgHTTPSTask{TLS_TASK}$ Server response: (639)<<html>
<head>
 <title>module.telit.com</title>
 <meta content="text/html; charset=utf-8" />
</head>
<body>
 <table border=0 align=center>
 <tr>
 <td height="100" align=center><h2>modules.telit.com - Test HTML page</h2></td>
 </tr>
 <tr>
 <td align=center><img src=Telit.jpg alt="Telit logo" height="126" width="410"></img></td>
 </tr>
 <tr>
 <td height="200" align=center> This is a simple HTML page, </br>
 made with simple HTML code,</br>
 just for test!
 </td>
 </tr>
 <tr>
 <td height="100" align=center><font size="3">Telit &copy; 2015 - 2017 All rights reserved</font></td>
 </tr>
 </table>
</body>
</html>
>
[DEBUG] 22.03  ssl_test:662 - msgHTTPSTask{TLS_TASK}$ application exit
[DEBUG] 22.03  ssl_test:680 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 22.03  ssl_test:683 - msgHTTPSTask{TLS_TASK}$ Application complete.
[DEBUG] 22.77  ssl_test:206 - PdpCallback{pubTspt_0}$ Context deactivated!
```

3.6.50 Uart To Server

Sample application showcasing how to send data from main UART to a connected TCP server. Debug messages are printed on AUX UART port.

Features

- How to open main UART to receive data
- How to connect to a server
- How to transmit received data from the UART to the server and viceversa

3.6.50.1 Application workflow

M2MB_main.c

- Open UART for data and USB1 for debug
- Init socket, activate PDP context and connect to server
- Init UART, set its callback function, create tasks to handle input from UART and response from server (optional)
- Send a confirmation on UART
- Wait for data, when it is received, send it to the server
- When a response is received, print it on UART.

Main UART:

```
Ready to receive data and send to socket.
<<<test message
<<<test 2
```

Figure 278

Debug log on USB1:

```
Starting. This is build: Jul 17 2019 16:39:24. MASK: 000F
Waiting for registration...
Activate PDP with APN internet.wind.biz....
Context activated!
Socket created
Server IP address: 185.86.42.218
Socket Connected and ready to receive data!
Uart opened, setting callback for data..
Waiting for data from uart.
UART IN: <test message>. Sending to socket...
Data sent to socket!
Response from server (12 bytes): <test message>
UART IN: <test 2>. Sending to socket...
Data sent to socket!
Response from server (6 bytes): <test 2>
```

Figure 279

3.6.51 UDP client

Sample application showcasing UDP echo demo with M2MB API. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to open a UDP client socket
- How to communicate over the socket

3.6.51.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task and start it

`m2m_udp_test.c` - Initialize Network structure and check registration - Initialize PDP structure and start PDP context - Create socket and link it to the PDP context id - Send data and receive response - Close socket - Disable PDP context

```
Starting UDP client demo app. This is v1.0.7 built on Apr  1 2020 14:57:13.
INIT
[DEBUG] 21.23  m2m_udp_test.c:223 - M2M_msgUDPTask{UDP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
Waiting for registration...
[DEBUG] 21.25  m2m_udp_test.c:131 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 21.26  m2m_udp_test.c:241 - M2M_msgUDPTask{UDP_TASK}$ Pdp context initialization
[DEBUG] 21.26  m2m_udp_test.c:245 - M2M_msgUDPTask{UDP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
Activate PDP with APN web.omnitel.it....
[DEBUG] 24.11  m2m_udp_test.c:157 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.11  m2m_udp_test.c:160 - PdpCallback{pubTspt_0}$ IP address: 109.113.222.12
[DEBUG] 24.12  m2m_udp_test.c:268 - M2M_msgUDPTask{UDP_TASK}$ Creating Socket...
[DEBUG] 24.13  m2m_udp_test.c:280 - M2M_msgUDPTask{UDP_TASK}$ Socket created
Socket ctx set to 3
[DEBUG] 24.41  m2m_udp_test.c:306 - M2M_msgUDPTask{UDP_TASK}$ Retrieved IP: 185.86.42.218
Socket ready.
Data successfully sent (16 bytes)
Socket recv...
[DEBUG] 26.47  m2m_udp_test.c:352 - M2M_msgUDPTask{UDP_TASK}$ m2mb_socket_bsd_set_sock_opt() M2MB_SOCKET_BSD_SO_RCVTIMEO - success
trying to receive 16 bytes..
Data received (16): <hello from m2mb!>
[DEBUG] 26.48  m2m_udp_test.c:377 - M2M_msgUDPTask{UDP_TASK}$ application exit
Socket Closed
[DEBUG] 26.49  m2m_udp_test.c:399 - M2M_msgUDPTask{UDP_TASK}$ m2mb_pdp_deactivate returned success
Application complete.
[DEBUG] 27.04  m2m_udp_test.c:166 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 280

3.6.52 UDP_Server example

Sample application that shows UDP listening socket demo with m2mb apis. Debug prints on **MAIN UART**

Features

- How to configure an UDP socket into listen mode
- How to receive data using m2mb_socket_bsd_select
- How to read data received and send data to client

Application workflow

`M2MB_main.c`

- Print welcome message
- Init task apis and create M2M_msgUDPTask to handle UDP socket

`m2mb_udp_test.c`

`M2M_msgUDPTask`

- Wait for module registration
- Activate PDP context
- Create UDP listen socket
- Wait for incoming data from client using m2mb_socket_bsd_select
- When there are data on socket, read them and send some data back to client

```
Starting UDP Server demo app. This is v1.1.10 built on Mar  3 2022 15:42:39.

[DEBUG] 19.28  m2mb_udp_te:221 - M2M_msgUDPTask{UDP_TASK}$ INIT
[DEBUG] 19.29  m2mb_udp_te:237 - M2M_msgUDPTask{UDP_TASK}$ m2mb_os_ev_init success
[DEBUG] 19.30  m2mb_udp_te:243 - M2M_msgUDPTask{UDP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 19.31  m2mb_udp_te:251 - M2M_msgUDPTask{UDP_TASK}$ Waiting for registration...
[DEBUG] 19.32  m2mb_udp_te:140 - NetCallback{pubTspt_0}$ Module is registered to cell 0x5222!
[DEBUG] 19.33  m2mb_udp_te:264 - M2M_msgUDPTask{UDP_TASK}$ Pdp context activation
[DEBUG] 19.33  m2mb_udp_te:268 - M2M_msgUDPTask{UDP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.39  m2mb_udp_te:283 - M2M_msgUDPTask{UDP_TASK}$ Activate PDP with APN ibox.tim.it....
[DEBUG] 22.00  m2mb_udp_te:167 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 22.01  m2mb_udp_te:170 - PdpCallback{pubTspt_0}$ IP address: 2.195.163.102

----    Start UDP server  Listen IP: 2.195.163.102   PORT: 6500    ----

[DEBUG] 24.07  m2mb_udp_te:299 - M2M_msgUDPTask{UDP_TASK}$ Creating Socket...
[DEBUG] 24.08  m2mb_udp_te:310 - M2M_msgUDPTask{UDP_TASK}$ Socket created
[DEBUG] 24.08  m2mb_udp_te:311 - M2M_msgUDPTask{UDP_TASK}$ m2mb_socket_bsd_socket(): valid socket ID [0x40035490] - PASS
[DEBUG] 25.12  m2mb_udp_te:317 - M2M_msgUDPTask{UDP_TASK}$ issuing m2m_socket_bsd_ioctl() to set non-blocking mode ...
[DEBUG] 25.13  m2mb_udp_te:329 - M2M_msgUDPTask{UDP_TASK}$ Binding Socket...
[DEBUG] 27.18  m2mb_udp_te:341 - M2M_msgUDPTask{UDP_TASK}$ Socket Bind Pass

Waiting for Incoming UDP data, Listen IP: 2.195.163.102 PORT: 6500

[DEBUG] 29.25  m2mb_udp_te:359 - M2M_msgUDPTask{UDP_TASK}$ select...
[DEBUG] 32.32  m2mb_udp_te:362 - M2M_msgUDPTask{UDP_TASK}$ Select result: 0
[DEBUG] 33.34  m2mb_udp_te:359 - M2M_msgUDPTask{UDP_TASK}$ select...
[DEBUG] 36.41  m2mb_udp_te:362 - M2M_msgUDPTask{UDP_TASK}$ Select result: 0
[DEBUG] 37.45  m2mb_udp_te:359 - M2M_msgUDPTask{UDP_TASK}$ select...
[DEBUG] 40.52  m2mb_udp_te:362 - M2M_msgUDPTask{UDP_TASK}$ Select result: 0
[DEBUG] 41.54  m2mb_udp_te:359 - M2M_msgUDPTask{UDP_TASK}$ select...
[DEBUG] 44.61  m2mb_udp_te:362 - M2M_msgUDPTask{UDP_TASK}$ Select result: 0
[DEBUG] 45.64  m2mb_udp_te:359 - M2M_msgUDPTask{UDP_TASK}$ select...
[DEBUG] 45.94  m2mb_udp_te:362 - M2M_msgUDPTask{UDP_TASK}$ Select result: 1
[DEBUG] 45.95  m2mb_udp_te:366 - M2M_msgUDPTask{UDP_TASK}$
Incoming UDP data available
----    Receive/send data test    ----

trying to receive bytes..
[DEBUG] 46.99  m2mb_udp_te:399 - M2M_msgUDPTask{UDP_TASK}$ Client Source Address: 151.99.1.17
[DEBUG] 47.00  m2mb_udp_te:400 - M2M_msgUDPTask{UDP_TASK}$ Client Port: 36368
[DEBUG] 47.01  m2mb_udp_te:401 - M2M_msgUDPTask{UDP_TASK}$ Client Family: 2
Data received (4): <test>
Data successfully sent (16 bytes)
```

Figure 281

3.6.53 USB Cable Check

Sample application showing how to check if USB cable is plugged in or not. Debug prints on **MAIN UART**

Features

- How to open an USB channel and configure it with a callback function
- How to manage USB cable events in the callback function

3.6.53.1 Application workflow

`M2MB_main.c`

- Open UART/UART_AUX for debug
- open usb channel and set the callback
- Print greeting message
- Print current usb status

`USB_Cb`

- if the event is a connection/disconnection, show the current status

```
Starting USB cable check demo app. This is v1.0.0 built on Aug 19 2020 10:27:40.

m2mb_usb_open succeeded
m2mb_usb_ioctl: set usb callback
m2mb_usb_ioctl: got cable status
USB cable CONNECTED, status: 1

 Waiting for USB cable to be plugged/unplugged...
Usb cable check event, USB status: 0
Usb cable check event, USB status: 1
Usb cable check event, USB status: 0
Usb cable check event, USB status: 1
```

Figure 282

3.6.54 Basic USB read/write example

Sample application that shows how to use the basic read/write USB apis. Synchronous or asynchronous mode is available setting SYNC to 1 or 0. Debug prints on **MAIN UART**

Features

- Read and write on USB (synchoronous mode)
- Read and write on USB (asynchronous mode)

3.6.54.1 Application workflow

M2MB_main.c

- Open USB port (USB0)
- Set rx and tx timeouts
- **SYNC**
- read until some data are availableon USB
- as soon as some data are available on USB read them and write on USB data received
- **ASYNC**
- set the USB callback
- write some data on USB and wait for data to be read
- as soon as some data are available on USB M2MB_USB_RX_EVENT is generated and handled by callback. Data are read and printed on serial com port.

```
Starting USB read write demo app. This is v1.0.0 built on Nov  4 2021 15:38:20.


Open USB port
m2mb_usb_open succeeded
m2mb_usb_ioctl: got cable status
USB cable CONNECTED, status: 1
Synchronous read and write
Read until some bytes are received...
rx timeout expired
```

Figure 283

3.6.55 Watchdog example

Sample application that shows how to set and implement a watchdog. Task has been locked waitng for an event with a timeout longer than wd inactivity timeout. If no wd kick or no actions

Features

- Enable watchdog on Task_1
- Handle watchdog kicks with a timer

3.6.55.1 Application workflow

M2MB_main.c

- Create Task_1 that will be put under watchdog control
- In Task_1 watchdog is ebnabled on it and a timer created to handle watchdog kicks. Then task enters in a loop where kicks are sent. After 5 kicks a task lock is simulated waiting for an event release (with a timeout longer that task watchdog inactivity timeout)
- As soon as watchdog inactivity timeout expired M2MB_WDOG_TIMEOUT_IND is generated and handled in WDcallback (in this case the event is released, no further actions are done)
- No further actions are performed by Task_1 (no kicks are sent) so after watchdog system timeout expiration app is rebooted.

```
Starting WATCHDOG demo app. This is v1.0.0 built on Nov  4 2021 14:34:06.

Event correctly initialized

Task started

Init WatchDog
m2mb_wDog_init OK
Tick duration: 1s
Adding Task under WD control with inactivity timeout of 60s
m2mb_wDog_addTask OK
Enabling the WatchDog
m2mb_wDog_enable OK
Timer creation OK

Start WD kick every 5s...
WD kick
WD kick done
```

Figure 284

3.6.56 ZLIB example

Sample application showing how to compress/uncompress with ZLIB. Debug prints on **MAIN UART**

Features

- How to compress a file
- How to uncompress a file

In order to execute the entire test, copy `test.gz` file into your module running the following AT command:

```
AT#M2MWRITE="/mod/test.gz",138
>>> here receive the prompt; then type or send the file, sized 138 bytes
```

3.6.56.1 Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Test the compression and decompression of a data string
- Test the decompression of a .gz file (test.gz), expected to be in `/mod` folder, into its content `test.txt`. The file must be uploaded by the user (see steps above).

```
Starting Logging demo app. This is v1.0.7 built on Apr  7 2020 09:02:35.

Starting TEST_COMPR_UNCOMPR.
len: 138; comprLen: 57
Compressed message:
 W+ÉHU(,ÌLÏVH*É/ÌSHÉ¨ PÉ*Ì-HMQÉ/K-R(  Ëç$VU*ëä§ë y4RÌ«¥1、
comprLen: 57; uncomprLen: 138
uncompress():
the quick brown fox jumped over the lazy dog. the quick brown fox jumped over the lazy dog. the quick brown fox jumped over the lazy dog.
Ending TEST_COMPR_UNCOMPR with SUCCESS.

Starting test_uncompress.

Data extracted correctly into the file ./mod/test.txt
test_uncompress finished correctly!
```

Figure 285

4 Installing beta version libraries Plug-in

4.1 New beta plug-in installation

To install a new plug-in for a beta firmware into the IDE, first receive plug-in “.zip” packet, then unzip the file in a local folder and open the SDK IDE.

PLEASE DO NOT USE BETA PLUGINS FOR PRODUCTION DEPLOYMENTS, SOFTWARE IS PROVIDED AS IS AND CUSTOMER ACKNOWLEDGES THAT IT IS POSSIBLE THE DEVICE MAY MISFUNCTION. PLEASE REFER TO Contact Information, Support section

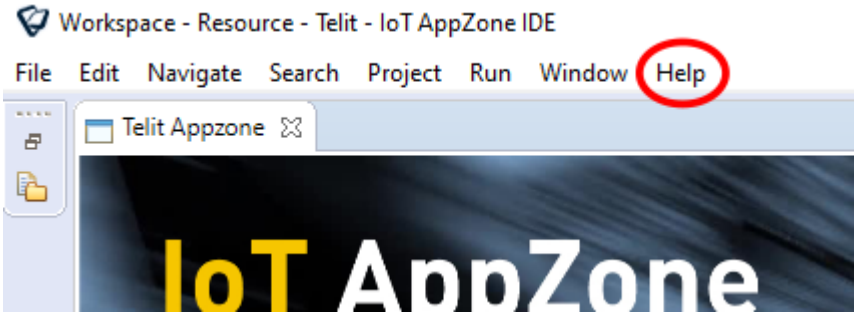

Figure 286

Click on “Help” tag and choose “Install New Software...”. This window will appear:

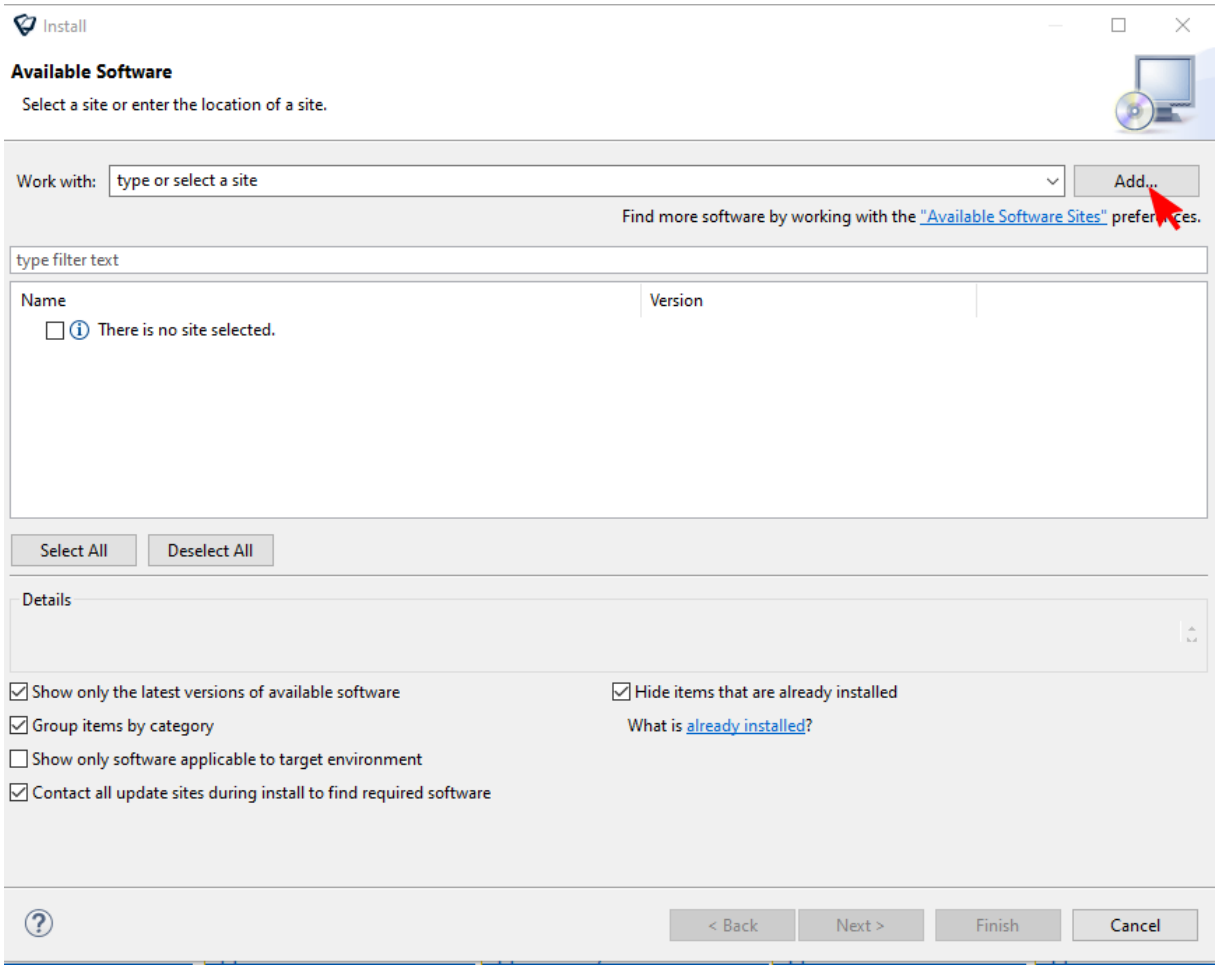

Figure 287

Click on “Add…” button and then in the following window click on “Local…” to select the unzipped folder with the plug-in content.

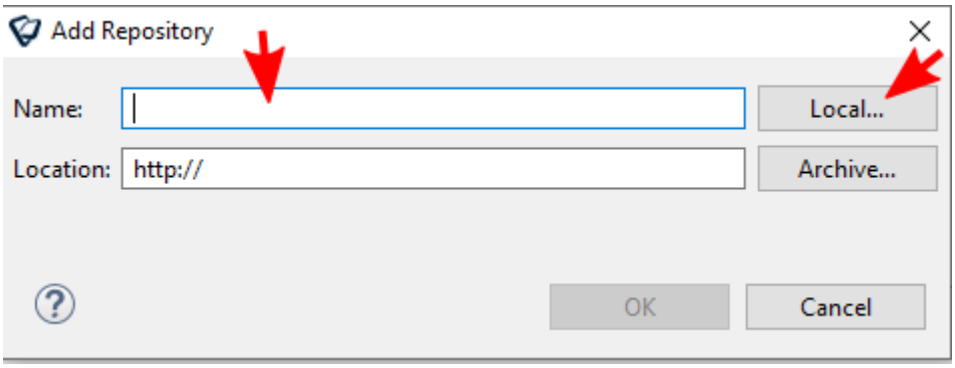

Figure 288

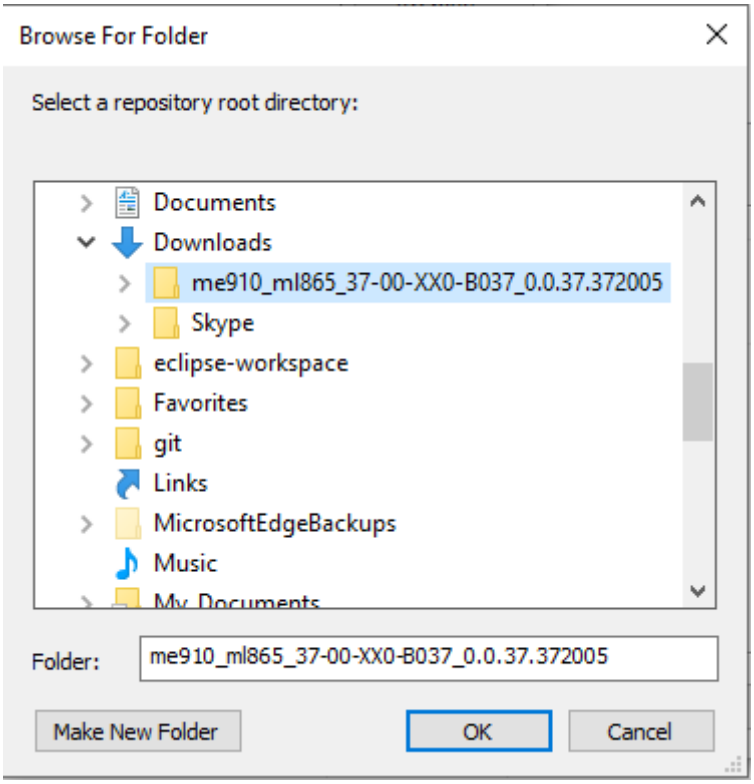

Figure 289

Once selected the plug-in folder, the “Location:” form will present the selected path. Now in “Name:” write a name for the new libraries (for example 37.00.xx0_B037) and click on “OK” button.

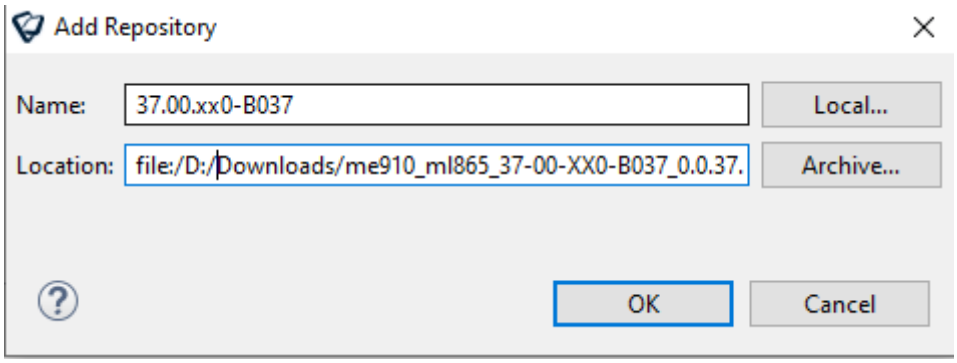

Figure 290

The new packet is now ready to be installed: select it and click on “Next >” button until “Review Licenses” window will appear.

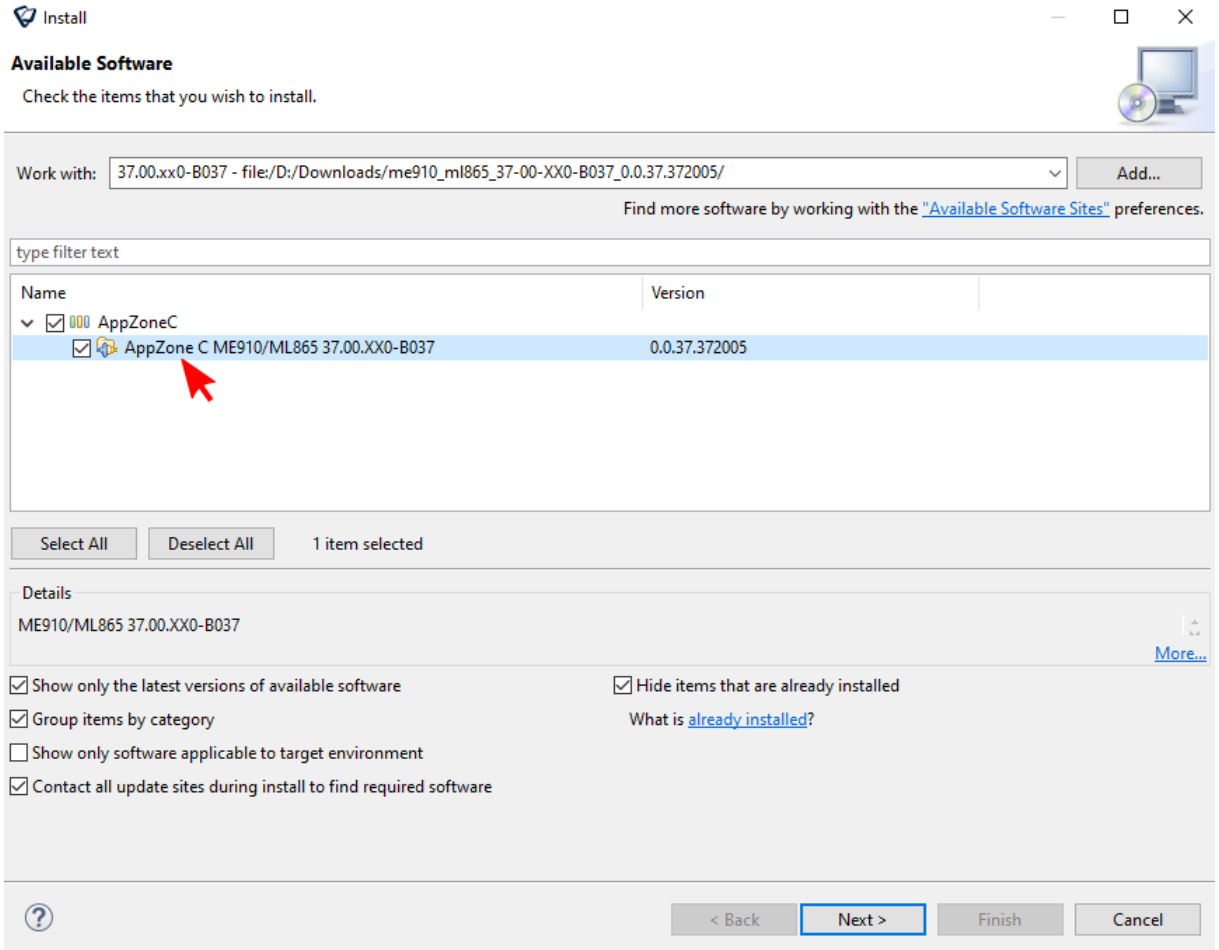

Figure 291

Accept the licenses when required and click on “Finish” button to complete the installation.

4.2 Change existing project libraries

To align an old project to the new libraries, right click on the project and choose “Properties”.

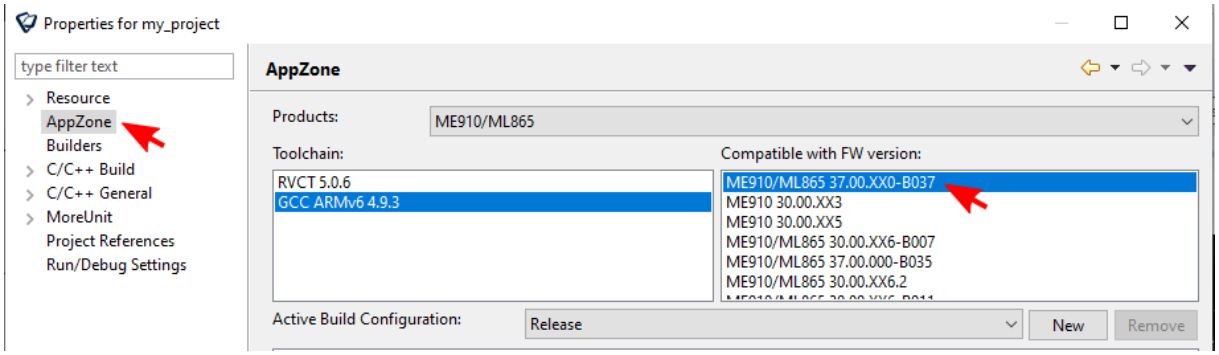

Figure 292

Now select “AppZone” on the left side of the window, and on the right choose the packet with the same name as the firmware version to be used. Then click on “OK” (or “Apply”) button.

4.3 Create a project with the new plug-in

To use the new libraries, create a new project: “File”-> “New” -> “Telit Project”

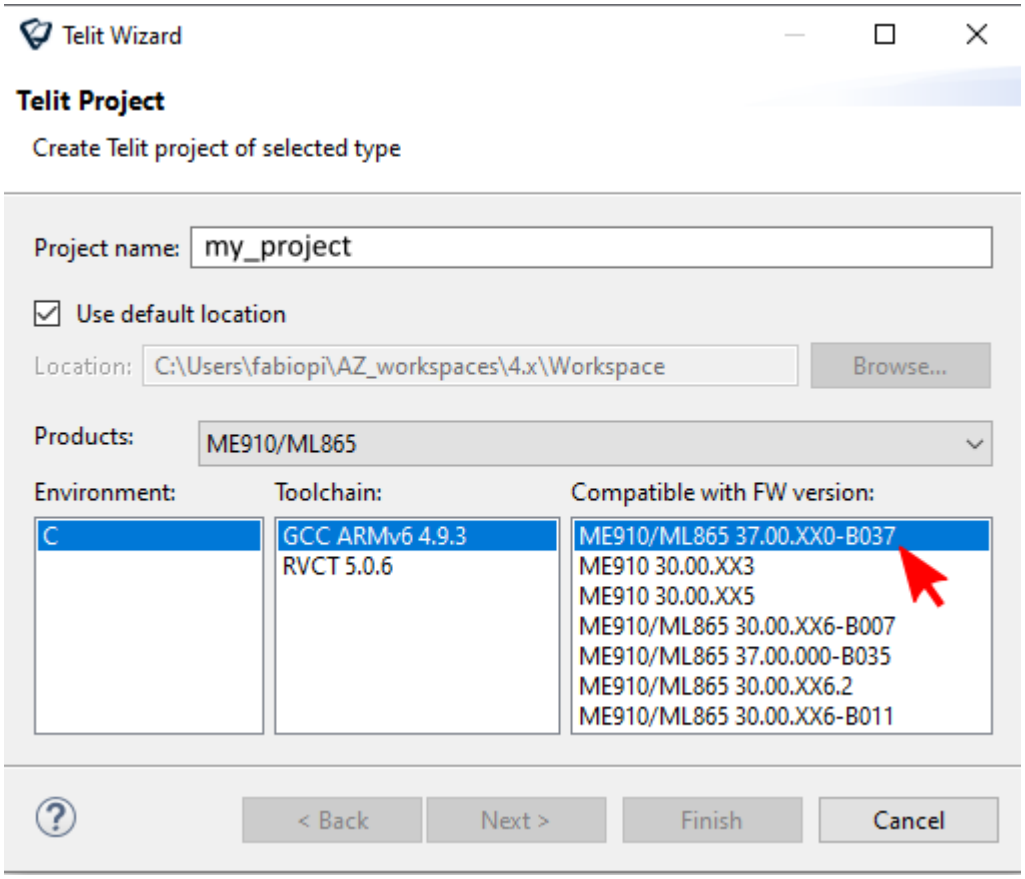

Figure 293

Select the new firmware version (37.00.xx0-B037) and create an empty project.

SUPPORT INQUIRIES

Link to **www.telit.com** and contact our technical support team for any questions related to technical issues.

www.telit.com

Telit Communications S.p.A.
Via Stazione di Prosecco, 5/B
I-34010 Sgonico (Trieste), Italy

Telit IoT Platforms LLC
5300 Broken Sound Blvd, Suite 150
Boca Raton, FL 33487, USA

Telit Wireless Solutions Inc.
3131 RDU Center Drive, Suite 135
Morrisville, NC 27560, USA

Telit Wireless Solutions Co., Ltd.
8th Fl., Shinyoung Securities Bld.
6, Gukjegeumyung-ro8-gil, Yeongdeungpo-gu
Seoul, 150-884, Korea

Telit Wireless Solutions Ltd.
10 Habarzel St.
Tel Aviv 69710, Israel

Telit Wireless Solutions
Technologia e Servicos Ltda
Avenida Paulista, 1776, Room 10.C
01310-921 São Paulo, Brazil

Telit reserves all rights to this document and the information contained herein. Products, names, logos and designs described herein may in whole or in part be subject to intellectual property rights. The information contained herein is provided "as is". No warranty of any kind, either express or implied, is made in relation to the accuracy, reliability, fitness for a particular purpose or content of this document. This document may be revised by Telit at any time. For most recent documents, please visit www.telit.com

Copyright © 2016, Telit