

The **INTERNET** of **THINGS**  
made **Plug&Play**

**Telit**<sup>®</sup> wireless  
solutions



---

# APPZONE C USER GUIDE

# APPLICABILITY TABLE

## PRODUCTS

■	■	GE910 SERIES
■	■	UE910 SERIES
■	■	HE910 SERIES
■	■	UL865 SERIES
■	■	UE866 SERIES
■	■	LE910V2 SERIES
■	■	LE866 SERIES

# SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE

## LEGAL NOTICE

These Guidelines are general guidelines pertaining to the installation and use of Telit's Integrated Application Development Environment ("IDE"). Telit and its agents, licensors and affiliated companies make no representation as to the suitability of these Guidelines for your particular needs and Telit disclaims any and all warranties, expressed or implied, relating to the contents of this document. Furthermore, this document shall not be construed as providing any representation or warranty with respect to any Telit Product whether referenced herein or not.

It is possible that this document may contain references to, or information about Telit products, services and programs, that are not available in your region. Such references or information shall not be construed to mean that Telit intends to make available such products, services and programs in your area.

## HIGH RISK USES

The IDE is not intended for the design of software for use in hazardous environments or environments requiring fail-safe controls, including the operation of nuclear facilities, aircraft navigation or aircraft communication systems, air traffic control, life support, or weapons systems ("High Risk Activities"). Without derogation, Telit, its licensors and its supplier(s) specifically disclaim any expressed or implied warranties with respect to the use of the IDE for development of software for such High Risk Activities and specifically disclaim all liability with respect to such use.

## TRADEMARKS

The names Telit, deviceWISE, deviceWISE by Telit, secureWISE, secureWISE by Telit, Telit IoT MODULES, Telit IoT PORTAL, Telit IoT PLATFORM, Telit IoT PLATFORMS, Telit IoT CONNECTIVITY, Telit IoT SERVICES and their associated logos are trademarks of Telit Communications PLC, its subsidiaries or affiliates in the United States and/or other countries. Other company or product names are the trademarks of their respective owners. All trademark rights are reserved.

## THIRD PARTY RIGHTS

The IDE may contain or may be provided in conjunction with third party software (the "third party software"), including some or all of those detailed in the NOTICES file provided to you during installation of the IDE. You acknowledge that not all third party software detailed in the NOTICES file are necessarily used or provided in the particular version of the IDE you install and use. Refer to the IDE's EULA and the NOTICES file for licensing information pertaining to the third party software.

# CONTENTS

1.1.	<b>Scope</b>	7
1.2.	<b>Audience</b>	7
1.3.	<b>Contact Information, Support</b>	7
1.4.	<b>Text Conventions</b>	7
1.5.	<b>Acronyms and Abbreviations</b>	8
1.6.	<b>Related Documents</b>	8
2.1.	<b>AppZone C overview</b>	9
2.2.	<b>AppZone vs. Application Processor</b>	10
3.1.	<b>Hardware and Software Requirements</b>	11
3.2.	<b>Install the IoT AppZone C IDE</b>	11
3.3.	<b>Connect the Module</b>	13
3.4.	<b>Update Versions Automatically</b>	13
3.4.1.	Update the Software Versions	13
3.4.2.	Update the Firmware	14
3.4.3.	Update Manually	15
3.4.4.	Change Automatic Upgrade Preferences	15
3.4.5.	Migrate project after upgrading the IDE	15
3.4.6.	Uninstall Updates	15
3.4.7.	Install an Older Version	16
4.1.	<b>Main Window</b>	17
4.2.	<b>AZ C Console</b>	18
4.2.1.	Filesystem Password	19
5.1.	<b>Create an Application</b>	20
5.2.	<b>Sample Applications</b>	25
5.2.1.	Hello World	25
5.2.2.	AT Tunnel	25
5.2.3.	Sample Information	26
5.2.4.	Default Application	26
5.2.5.	SMS-AT	27
5.2.6.	UART/USB to Server	28
5.2.7.	Control Panel	30
6.1.	<b>Set Project Settings</b>	32
6.2.	<b>Compile the Project</b>	32
8.1.	<b>Run Applications on the Module – Automated process</b>	36
8.2.	<b>Run Applications on the Module – Manual process</b>	36
8.2.1.	Connect the COM Port	36
8.2.2.	Upload and Run the Application	37

8.2.3.	Managing Files on the Module	38
<b>8.3.</b>	<b>Run and Debug Applications on Emulator</b>	<b>39</b>
<b>9.1.</b>	<b>Creating an AppZoneC static library</b>	<b>41</b>
<b>10.1.</b>	<b>M2M_initialize.c File</b>	<b>44</b>
<b>10.2.</b>	<b>M2M_proc1.c (default) M2M_procX.c Files</b>	<b>44</b>
<b>10.3.</b>	<b>M2M_main.c File</b>	<b>45</b>
<b>10.4.</b>	<b>M2M_arRsp.c File</b>	<b>45</b>
<b>10.5.</b>	<b>M2M_hwEvents.c File</b>	<b>45</b>
<b>10.6.</b>	<b>M2M_net.c File</b>	<b>46</b>
<b>10.7.</b>	<b>M2M_shell.c File</b>	<b>47</b>
<b>12.1.</b>	<b>Hardware and software requirements</b>	<b>51</b>
<b>12.2.</b>	<b>Initial configuration</b>	<b>51</b>
<b>12.3.</b>	<b>Debug setup</b>	<b>52</b>
<b>12.4.</b>	<b>Debug an application</b>	<b>54</b>
<b>14.1.</b>	<b>AT+M2M</b>	<b>59</b>
<b>14.2.</b>	<b>AT#M2MWRITE</b>	<b>60</b>
<b>14.3.</b>	<b>AT#M2MLIST</b>	<b>61</b>
<b>14.4.</b>	<b>AT#M2MDEL</b>	<b>61</b>
<b>14.5.</b>	<b>AT#M2MDELALL</b>	<b>61</b>
<b>14.6.</b>	<b>AT#M2MREAD</b>	<b>62</b>
<b>14.7.</b>	<b>AT#M2MRUN</b>	<b>62</b>
<b>14.8.</b>	<b>AT#M2MCHDRIVE</b>	<b>63</b>
<b>14.9.</b>	<b>AT#M2MCHDIR</b>	<b>63</b>
<b>14.10.</b>	<b>AT#M2MMKDIR</b>	<b>64</b>
<b>14.11.</b>	<b>AT#M2MRMDIR</b>	<b>64</b>
<b>14.12.</b>	<b>AT#M2MBA</b>	<b>65</b>
<b>14.13.</b>	<b>M2M AT Command Examples</b>	<b>65</b>
14.13.1.	AT+M2M=0	65
14.13.2.	AT+M2M=1	66
14.13.3.	AT+M2M=2	69
14.13.4.	AT+M2M=3	70
14.13.5.	AT+M2M=4,30	72

# FIGURES

Fig. 1: Module with AppZone Software Layer .....	9
Fig. 2: Module without AppZone C Layer .....	10
Fig. 3: Module with AppZone C Layer .....	10
Fig. 4: Skeleton with the Header Files (Includes folders).....	44
Fig. 5: DTE Connected to the Module .....	65

# 1. INTRODUCTION

This document describes how to use AppZone C to develop applications for Telit modules.

## 1.1. Scope

This document describes the AppZone C development environment and how to use the development environment to develop applications for Telit modules. It also provides examples using the +M2M AT commands to configure the module and manage the M2M applications uploaded into the module.

The development environment is based on Eclipse IDE. For complete documentation of the Eclipse IDE, refer to Eclipse documentation at <http://www.eclipse.org/>.

## 1.2. Audience

This guide is intended for users who need to develop a custom application (M2M application) and run it on a Telit's module as an embedded application that uses the services provided by the module itself.

## 1.3. Contact Information, Support

For general contact, technical support services, technical questions and report documentation errors contact Telit Technical Support at:

[TS-EMEA@telit.com](mailto:TS-EMEA@telit.com)

[TS-AMERICAS@telit.com](mailto:TS-AMERICAS@telit.com)

[TS-APAC@telit.com](mailto:TS-APAC@telit.com)

Alternatively, use:

<http://www.telit.com/support>

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.

## 1.4. Text Conventions



Caution or Warning – Alerts the user to important points about integrating the module, if these points are not followed, the module and end user equipment may fail or malfunction.



Tip or Information – Provides advice and suggestions that may be useful when integrating the module.

---

All dates are in ISO 8601 format, i.e. YYYY-MM-DD.

## 1.5. Acronyms and Abbreviations

IDE	Integrated Development Environment
API	Application Programming Interface
APN	Access Point Name
DTE	Data Terminal Equipment
GCC	GNU Compiler Collection
GPIO	General Purpose Input/Output
GUI	Graphic User Interface
I2C	Inter-Integrated Circuit
NVM	Non-Volatile Memory
PDP	Packet Data Protocol
SDK	Software Development Kit
SMS	Short Message Service
SPI	Serial Peripheral Interface
SSL	Secure Socket Layer
UART	Universal Asynchronous Receiver Transmitter

## 1.6. Related Documents

- [1] UE910 Hardware User Guide, 1v0301012
- [2] AppZone C APIs Reference Guide, 80496ST10723A
- [3] HE910 Hardware User Guide, 1v03700925
- [4] GE910 Hardware User Guide, 1v0300962
- [5] File System Tool User Guide, 1v0301192
- [6] Telit 3G Modules AT Commands Reference Guide, 80378ST10091A
- [7] AT Commands Reference Guide, 80000ST10025A
- [8] Running AT Commands Remotely, 80000NT10029A
- [11] Telit 3G Modules Ports Arrangements, 1v0300971
- [12] GE910 Series Ports Arrangements, 1v0301049
- [13] UL865 Hardware User Guide, 1v0301050
- [14] UE866 Hardware User Guide, 1v0301157
- [15] AppZone Sample Applications Guide, 1v0301193



In the next pages is used the notation [x]/[y] to refer to documents of different modules. You have to see the document in accordance with the module you are using.

---



## 2. ABOUT APPZONE C

AppZone C is an integrated development environment based on Eclipse IDE and running on Windows and Linux. This environment provides a set of template files that form the skeleton of a future user M2M application, along with tools for interacting with Telit modules. You can write the code within the callback functions contained in the .c files of the skeleton.

### 2.1. AppZone C overview

AppZone C is a software layer that provides a set of APIs to access the modem features of major operational interest. By means of AppZone C, the user M2M application runs on the module CPU; this solution does not require an external application processor.

You can develop M2M applications that can address a wide range of different applications such as remote monitoring and control, security and surveillance, telemetry, location services, billing, and fleet management. The generic user application can access the following modem resources:

- Operating System: Signals, Semaphores, Timers, Dynamic Memory Management, etc.
- HW/SW: GPIO, I2C, UART, SPI, Keypad, File-System, RTC, etc.
- GSM/GPRS: Communication services.
- Networking: BSD socket support, SSL capabilities.

Refer to documents [1]/[3]/[4] for information on module hardware resources.

Fig. 1 shows the high-level architecture of a module provided with the AppZone C software layer. Use AppZone C to develop custom M2M applications, in accordance with the AppZone layer, which provides a set of files called skeleton.

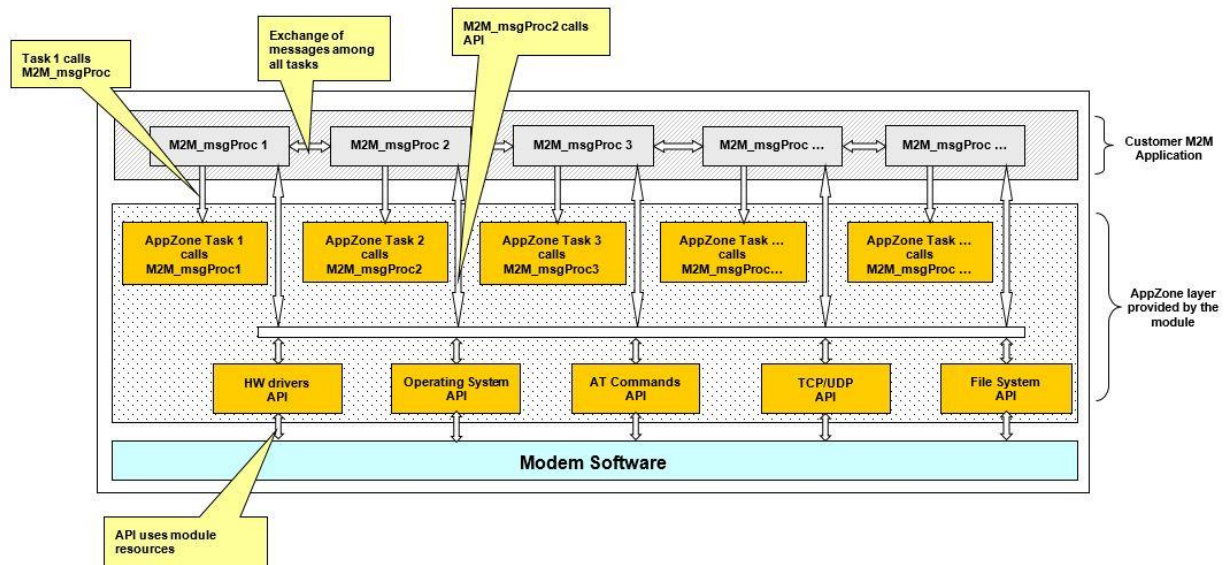


Fig. 1: Module with AppZone Software Layer

The skeleton factory default configuration includes a single M2M\_procX.c template file, named M2M\_proc1.c that contains the M2M\_msgProc1() callback function. You can write new M2M\_procX.c files, new M2M\_msgProcX() callbacks, and create new AppZone Tasks. AppZone layer can support up to 32 tasks in total. In general, each AppZone Task calls its callback function, for more information refer to *AppZone C APIs Reference Guide*:

- AppZone Task\_1 calls M2M\_msgProc1 (default)
- AppZone Task\_2 calls M2M\_msgProc2 (created by the user)
- AppZone Task\_3 calls M2M\_msgProc3 (created by the user)
- AppZone Task\_... calls M2M\_msgProc... (created by the user)

Use the M2M\_msgProcX() callback functions to develop the M2M applications. Tasks communicate with each other exchanging messages using the suitable API.



Use the C programming language to develop the M2M applications.

## 2.2. AppZone vs. Application Processor

The figures below show two generic examples of hardware/software configurations that point out the differences between the use of a module providing the AppZone C Software Layer and another not providing that feature.

Fig. 2 shows a configuration where the user application is running on an external application processor. The user application uses the GPRS/HSPA services and User Device #2 by means of the “A” interface, e.g. a serial port.

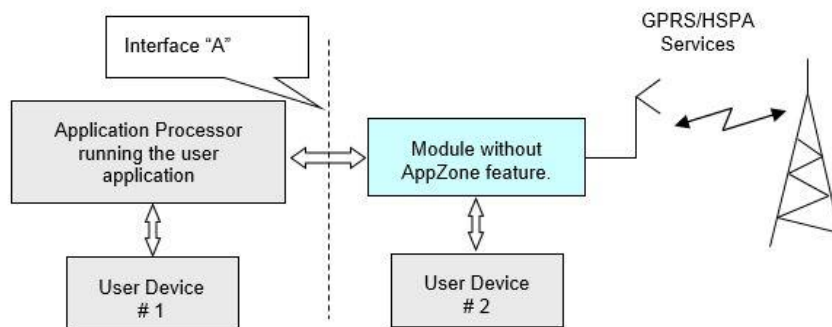


Fig. 2: Module without AppZone C Layer

Fig. 3 shows a configuration where the module provides the AppZone layer. In this scenario, the user application is running on the module CPU. The user application uses the GPRS/HSPA services, User Device #1, and #2 by means of the set of AppZone layer interfaces (APIs). The external user application processor is not required.

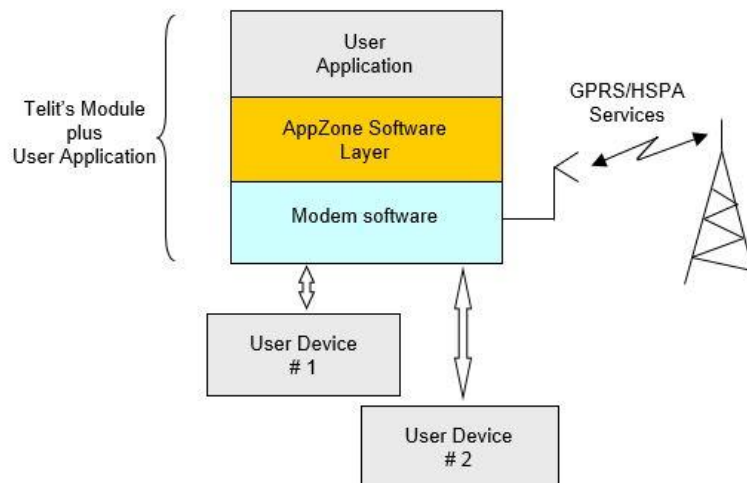


Fig. 3: Module with AppZone C Layer

## 3. INSTALL APPZONE C

AppZone C is part of AppZone, the integrated development environment based on Eclipse IDE and running on Windows and Linux. The AppZone C plugin is an environment provides a set of template files that form the skeleton of a future user M2M application. The user writes the code within the callback functions contained in the .c files of the skeleton.

During installation, AppZone C announces itself as *IoT AppZone IDE*.



We strongly recommended that you use an AppZone C version that is aligned with the software version of the module that you are using.

AppZone C checks for software and firmware updates. If a new version is found, AppZone C displays a notification. You can update the software versions automatically. AppZone C also downloads new firmware versions of the module that is connected to your computer. If you do not enable automatic updates, you can update the software and the firmware manually at any time.

AppZone C uses the Eclipse platform to check for Software updates during startup. You can change the automatic update configuration at any time. See [Change Automatic Upgrade Preferences](#) for more information.

### 3.1. Hardware and Software Requirements

To install AppZone C you must:

- Have administrator permissions
- Know which module family you will be using

The following table lists the requirements for installing AppZone C:

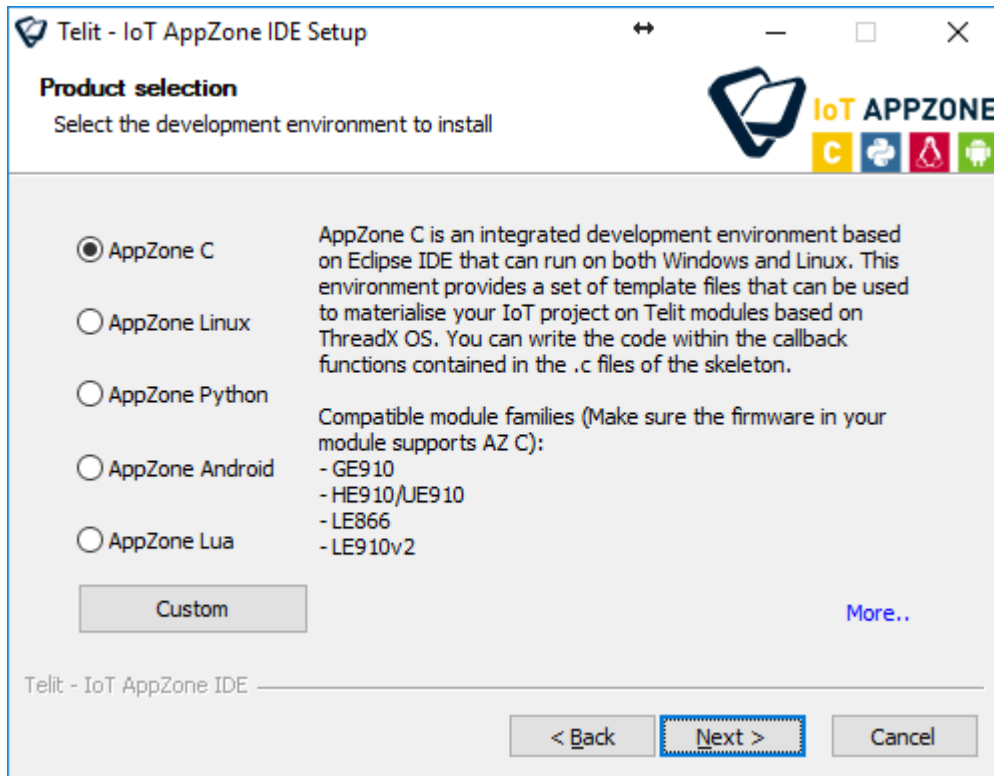
<b>OS</b>	Windows 7 (32bit or 64bit), Windows 8, Windows 10
<b>RAM</b>	1 Gb
<b>Free disk space</b>	1250 MB free disk space
<b>Java SE Runtime</b>	Java™ SE Runtime Environment 32-bit <b>Note:</b> 64-bit is not compatible with AppZone IDE.

### 3.2. Install the IoT AppZone C IDE

**To install AppZone C on Windows:**

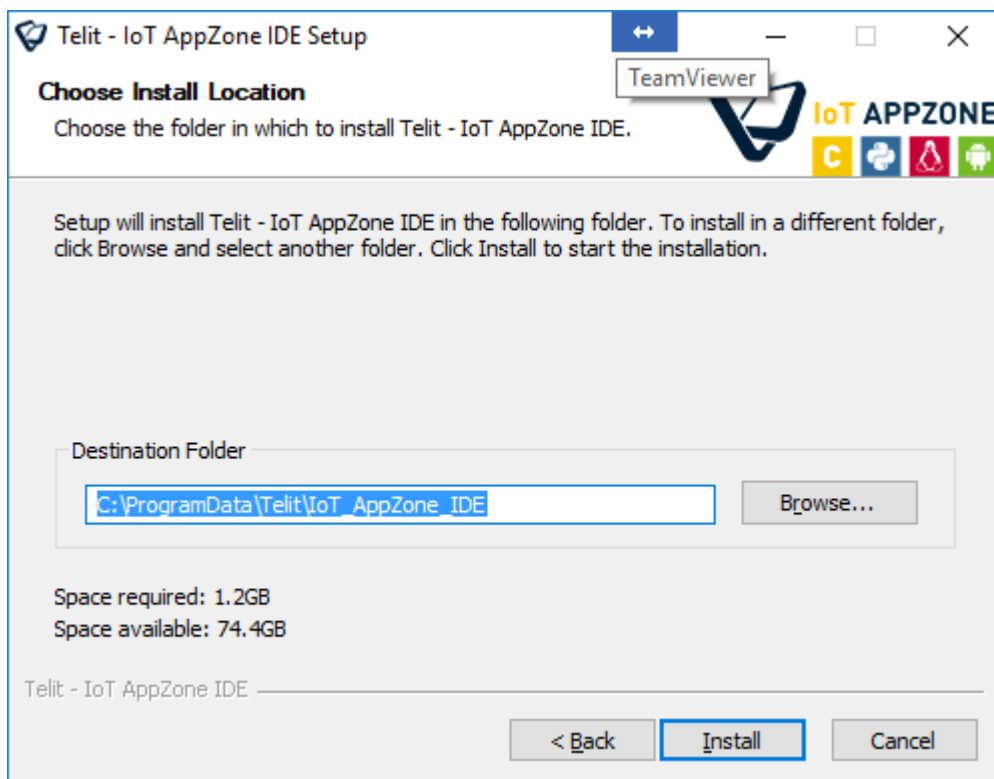
1. Access the Telit portal: <https://www.telit.com/developer-zone/iot-app-zone/iot-app-zone-developer-resources/>
2. Click the AppZone installation file.  
The installation wizard starts.
3. If the correct Java version is not installed, the wizard downloads the appropriate package and prompts the user to install it. Once it is installed, restart the installer.
4. Click **Next** to continue.
5. Read the license agreement. If you agree, click **I Agree** to continue.

6. Select AppZone C, and then click **Next**.



To install more than one development environment, click **Custom** and select the development environments that you want to install.

7. Choose the destination folder in which to install AppZone C. You can leave the default folder or click **Browse** to select a new destination folder.



8. Click **Install** to begin the installation.

The installation wizard downloads the files that are required for the development environment that you selected.

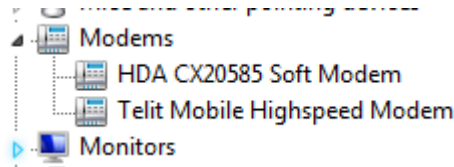
9. Click **Finish** to close the AppZone C installation wizard.

### 3.3. Connect the Module

To confirm that your module is recognized by your computer:

1. Click **Start**, and then click **Control Panel**.
2. Click **Hardware and Sound**.
3. Click **Device Manager**.
4. Expand **Modems** and verify that a Telit module is listed.

For example:



### 3.4. Update Versions Automatically

AppZone C checks for newer versions and enables installing them.

You can update the SDK and the firmware versions:

- [Update the Software Versions](#) – update the IDE and the libraries that are used to communicate with the firmware
- [Update the Firmware](#)

If you dismissed an update notification, you can check for new versions at any time:

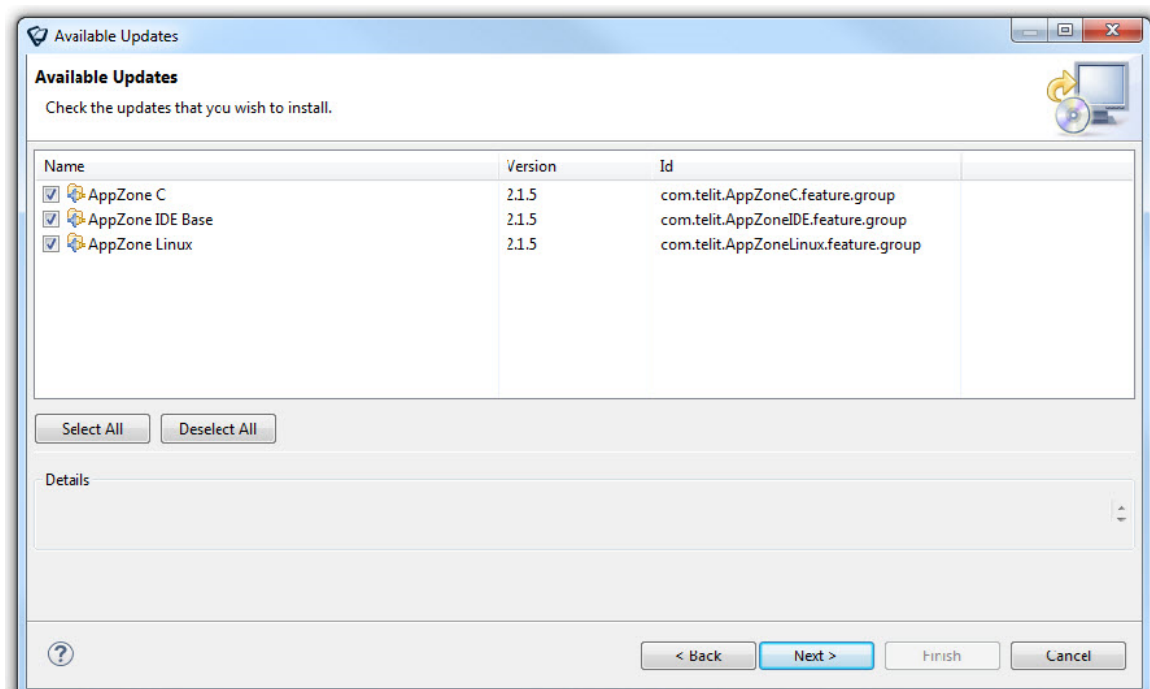
- From the menu, select **Help > Check for updates**

#### 3.4.1. Update the Software Versions

If a new software version exists, AppZone C displays a notification.

**To update the version:**

1. Click the Updates Available notification. The Available Updates window opens.



2. Select the versions to install, and then click **Next**:



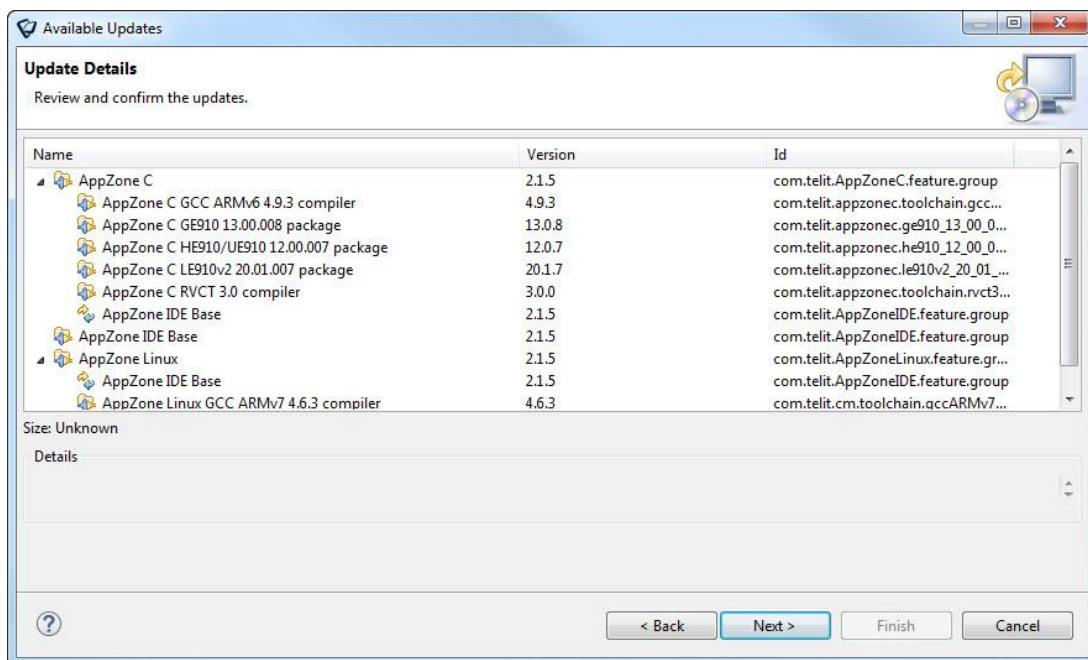
We recommend that you update all versions and firmware to avoid inconsistencies between versions.

- AppZone IDE Base– The AppZone SDK
- AppZone C –The libraries that are used to develop apps for the specified firmware



If you update the library (AppZone C) you must also update the firmware or you will not be able to work with the module.

The Available Updates window opens, listing the packages that will be installed.



3. Review the installed packages, and then click **Next**.

The License Agreement window opens.

4. Read the license agreement. If you agree, select **I Agree** and then click **Finish**.

The packages that you selected are downloaded.

5. After the packages are downloaded, a message opens requesting your permission to install the new packages. Click **OK** to install the new packages.

6. A message indicates that the new version has been installed. For the changes to take effect, click **OK** to restart Eclipse.

### 3.4.2. Update the Firmware

When you connect to the board using AutoConnect, AppZone C compares the library version to the firmware installed on the module. If the libraries have a higher version than the firmware, AppZone C displays a message.

#### To update the firmware version:

1. In the Target Firmware Check window, click **Download latest version**. AppZone C downloads the latest firmware.

If you want to update the firmware version at a later time, view the location of the firmware. You can use XFP at any time to flash the new firmware.

2. To flash the new firmware, click **Flash Firmware**. The Xfp window opens.
3. Follow the instructions in the window, based on your type of module, and then click **Program**.



### 3.4.3. Update Manually

If a software or firmware versions was detected but you did not upgrade the version, you can upgrade each of the versions at any time.



Make sure that the software version that you install is aligned with the firmware version of the module.

To update the software versions:

1. From the menu, select **Help > Install new Software**. The Available Software window opens.
2. From the **Work With** list, select the site from which to download software versions. The list of software versions is updated.
3. Select the checkboxes next to the versions that you want to install, and then click **Next**.
4. Verify the version to install, and then click **Next**.
5. If you accept the license agreement, select **I accept the terms of the license agreements** and then click **Finish**.

### 3.4.4. Change Automatic Upgrade Preferences

**To change the default time for updates:**

1. From the menu, select **Window > Preferences > Install/Update > Automatic Updates**. The Preferences window opens.
2. Change your preferences, and then click **OK**.  
You can disable automatic updates, change the schedule, specify download options, and how to handle new updates.

In some cases, Telit personnel may provide an alternative site for specified packages.

**To change the site from which update packages are downloaded:**

1. From the menu, select **Window > Preferences > Install/Update > Available Software Sites > Add**. The Add Site window opens.
2. In the Name field, enter a name for the site so that you can identify the site later easily.
3. In the Location field, type the URL to the site that contain the new version.
4. Click **OK**.

### 3.4.5. Migrate project after upgrading the IDE

From version 4.0.0 of the IDE projects from older versions of the IDE will automatically be upgraded to the new version when you continue working on them.

### 3.4.6. Uninstall Updates

To uninstall updates, you must uninstall AppZone C packages and then reinstall packages of a specified version.

**To uninstall updates:**

1. From the menu, select **Help > Installation Details**. The Telit IoT AppZone IDE Installation Details window opens
2. Select AppZone C, and then click **Uninstall**.
3. Restart AppZone C.
4. Install a new version.

For information on how to change the URL from which to install a new version, see [Change Automatic Upgrade Preferences](#).

For information on how to install the version manually, see [Update Manually](#).

### 3.4.7. Install an Older Version

If you had a beta version with unique features and now you need to install the last official version, you might need to go back to the last official version, which will have a lower version number.

**To install an older version:**

1. Uninstall the AppZone C version. For details, see [3.4.5 Migrate project](#) after upgrading the IDE

From version 4.0.0 of the IDE projects from older versions of the IDE will automatically be upgraded to the new version when you continue working on them.

2. Uninstall Updates.
3. If required, change the URL from which to install a new version, see [Change Automatic Upgrade Preferences](#).
4. Install the version manually. For details, see [Update Manually](#).

In the **Help > Install new Software** window you must clear the **Contact all update sites during install to find required software** option, which is located at the bottom of the window.



If you do not clear this checkbox, the Telit IoT IDE dependencies will not be installed correctly.

---



# 4. GET STARTED

To open the development environment on Windows:

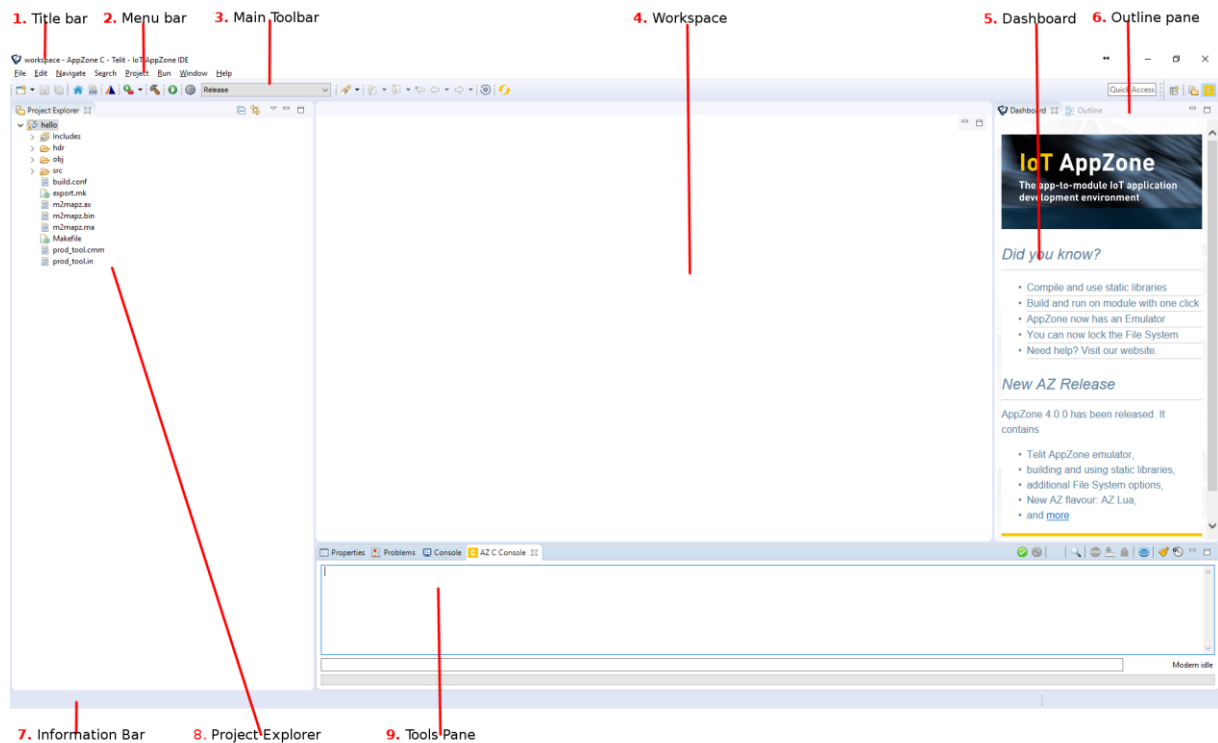


**Note:** You must have administrator privileges or the development environment will not load.

1. Click **Start > Telit > AppZoneIDE > AppZoneIDE**. The Workspace Launcher window opens.
2. Select the workspace in which you want to work, and then click **OK**.

## 4.1. Main Window

The following figure describes the main sections of the AppZone C window:



You can optimize the location of the panes and the views to suit your workflow. You can rearrange the layout of the panes and the views.

The main window contains the following sections:

1. Titlebar – Displays information on the environment, the current project, and the file that is currently displayed in the workspace in the following format: `AppZone C <Project name>/<File name>`
2. Menu Bar – Provides options that enable you to perform the most common operations on items. Most of these options also appear in the pop-up menus when you right-click items. The menu bar also contains options related to the entire application.
3. Main Toolbar – Provides buttons for navigation and most commonly used operations.
4. Workspace – Displays the details of the file that you select in the Project Explorer pane. In the workspace you define write and edit the code.
5. Dashboard – Keep up to date with new tools and features of the AppZone IDE
6. Outline Pane – Provides an outline of your current project.

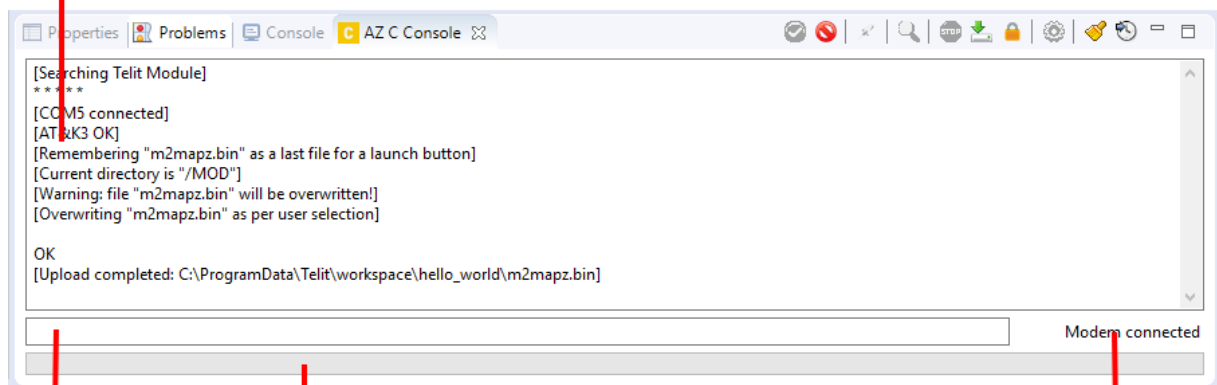
7. Information Bar – Displays information on the file that is currently displayed in the workspace in the following format: /<Project name>/<File name>
8. Project Explorer – Displays all files in the current project.
9. Tools Pane – Displays the following views:
  - Properties – Displays the properties of the file that is currently selected in the Project Explorer.
  - Problems – Displays system-generated errors, warnings, or information associated with a resource.
  - Console – Displays a build console that enables viewing the status of the current build.
  - AZ C Console – Enables exchanging AT Commands and files with the module and displaying the results. For more information, see AZ C Console.

## 4.2. AZ C Console

The AZ C Console enables exchanging AT Commands and files with the module and displaying the results. The AZ C Console enables to:

- transfer a .bin file to the module file system (uploading)
- read a file stored on the module file system and store it on PC (downloading)
- list files, check current directory, change directories on the module file system
- set COM ports, Baud Rate, etc.
- set the selected file as AppZone Application (refer to AT+M2MRUN command)
- enter AT commands

### 1. Display area



### 2. Command area




### 3. Progress bar








### 4. Connectivity status

The AZ C Console contains the following areas:

1. Display area – Displays the messages from the module, such as the AT Commands results and the connection COM result.
2. Command area – Enables entering AT commands.
3. Progress bar – Displays the progress of download and upload operations.
4. Connectivity status – Shows if the connection to the modem is active and if modem, or port is reachable

The following table describes the icons in the AZ C Console toolbar:

Icon	Name	Description
	Connect COM port	Connects to the COM port that is selected in the Settings.
	Disconnect COM port	Disconnects from the COM port.
	Launch application on target	Starts the application on the module that is connected.

Icon	Name	Description
	AutoConnect COM Port	Automatically connects to the COM port that is defined in the Settings.
	Abort transfer	Stops transferring the file to the module.
	File Manager	Opens the File Manager, which enables viewing files in the module.
	Settings	Configures the settings that are used to connect to the module.
	Clear Log	Clears the display area.
	Time Logging	Enables time logging.
	Filesystem Password	Set or remove a password which protects the filesystem from reads and writes over the Serial port

#### 4.2.1. Filesystem Password

If you would like to protect the data on the module's filesystem, then you can do so by using the Filesystem Password feature from the AZ Console.

To enable just click the icon and enter a new password. From that point on files from the modem cannot be retrieved through the serial connection anymore without the password.



Once a modem is unlocked by a correct password, it will stay unlocked until it has been reset.

To disable the feature, click on the icon and provide an empty string for the password.

If a modem is protected by a password the IDE will ask the user to provide the password before performing Filesystem operations. Once a correct password is entered, the IDE remembers the password in memory allowing the user to work without a need to reenter it. Once the IDE is closed down the password is forgotten.

# 5. CREATE APPLICATIONS

This chapter describes how to create a new application. You can create an empty application or base the new application on existing examples.



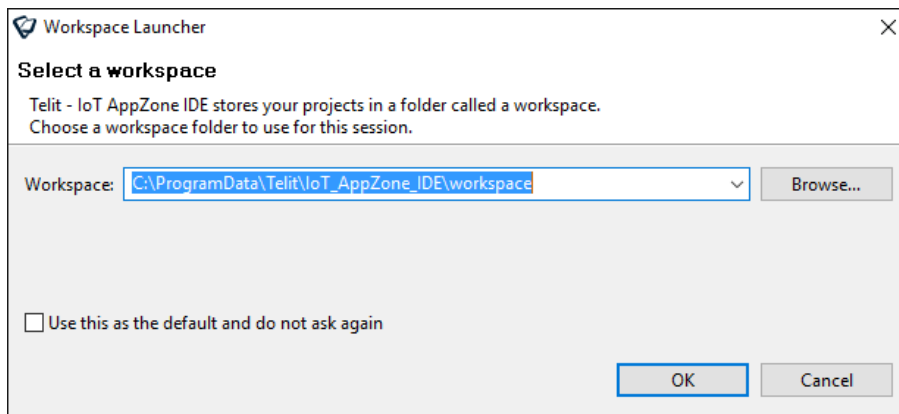
On the module, AppZone C and AppZone Python services are mutually exclusive. Choose the module type in accordance with your target.

## 5.1. Create an Application

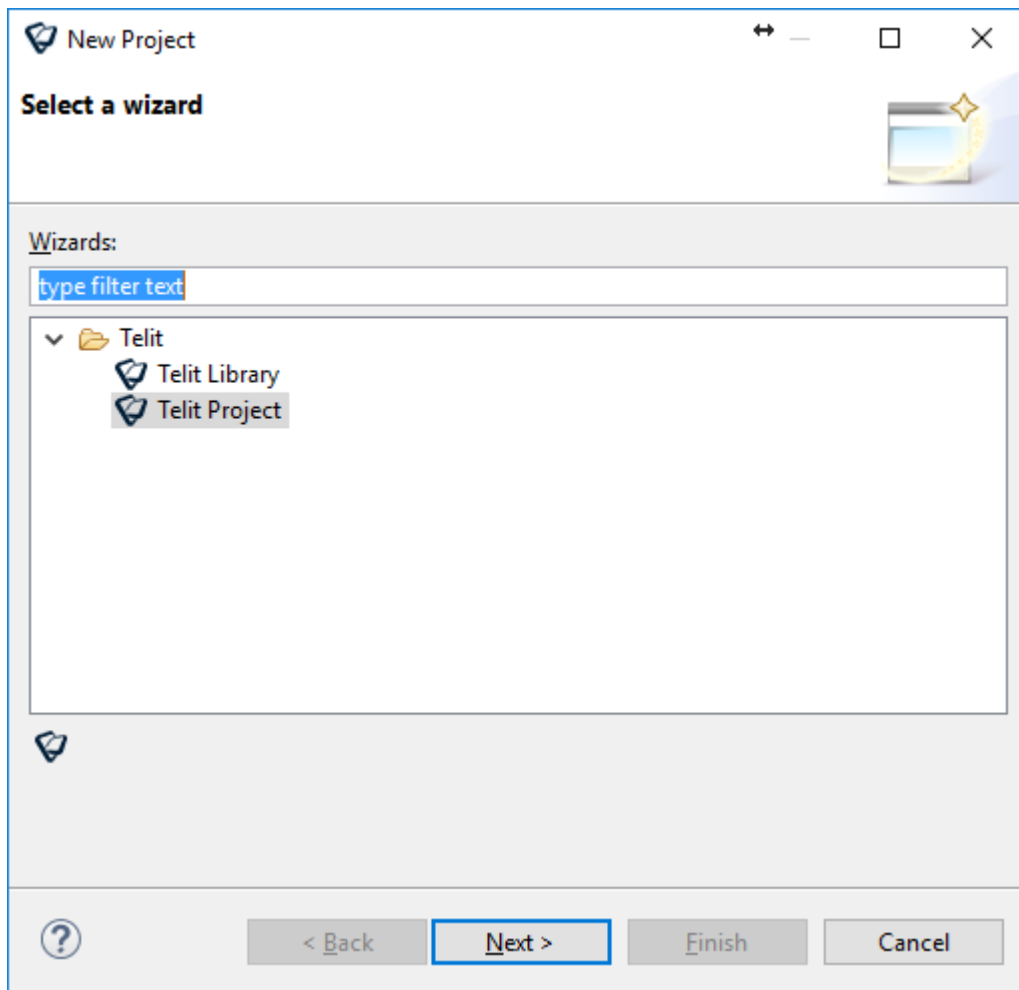
You can create new applications from scratch or use an example application.

To create a new application:

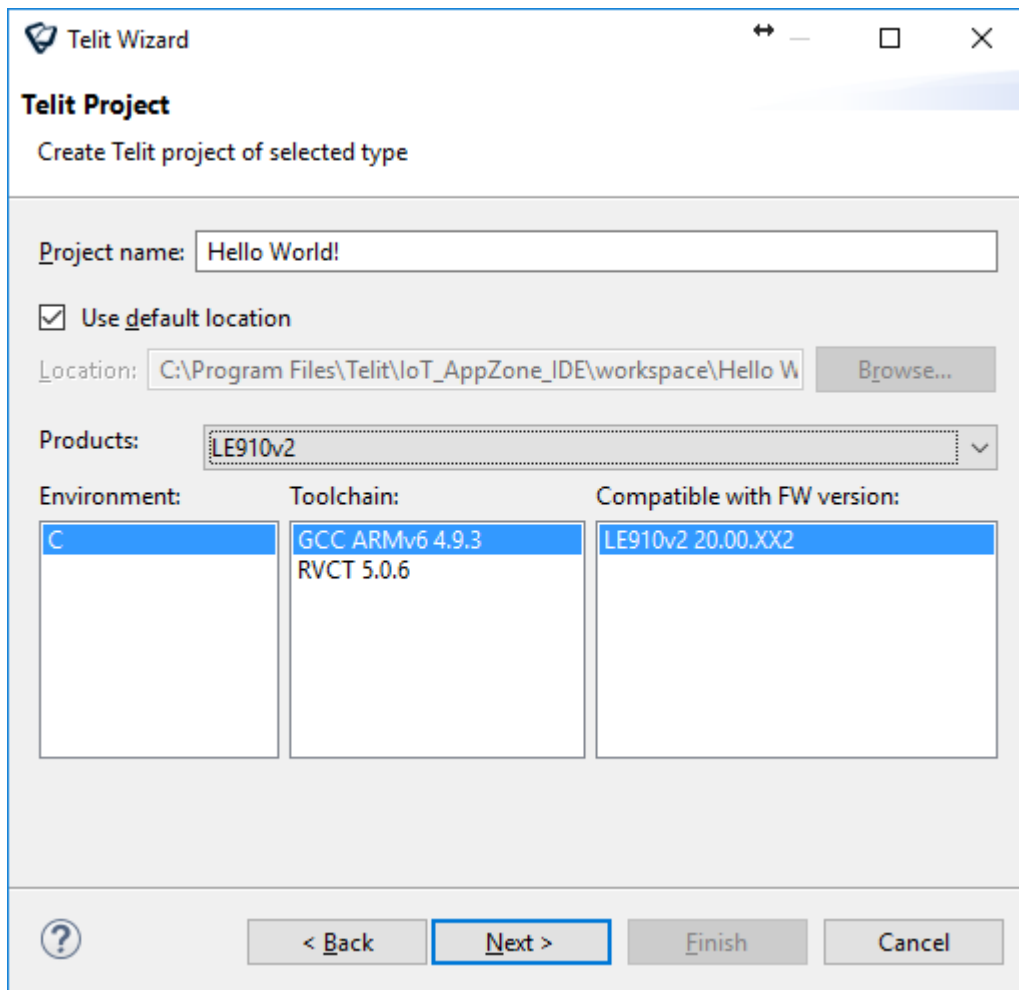
1. Click Start > All applications > Telit > AppZoneIDE > AppZoneIDE. The Workspace Launcher window opens.
2. In the **Workspace** field, type the location of the workspace or click **Browse**.



3. Click **OK**. The AppZone C IDE opens.
4. From the menu select **File > New > Project**. The New Project window opens.



5. To create a new project, expand **Telit** and then select **Telit Project**.
6. Click **Next**. The Telit Project window opens.



- a. In the **Project name** field, type a name for the project.
- b. To change the default location in which the project is saved:

Clear the **Use default location** checkbox.

In the **Location** field type the new location or click **Browse**.

- c. In the **Products** field, select the module for which you want to develop the application.
- d. In the **Environment** field, select C.
- e. In the **Toolchain** field, select a toolchain:

GCC – Open source compiler

RVCT – ARM RVCT 3.0 compiler that is included in the AppZone C SDK package. To buy the license, contact Telit. After you have a license, see [Using the RVCT Compiler](#) for information on how to set the license.

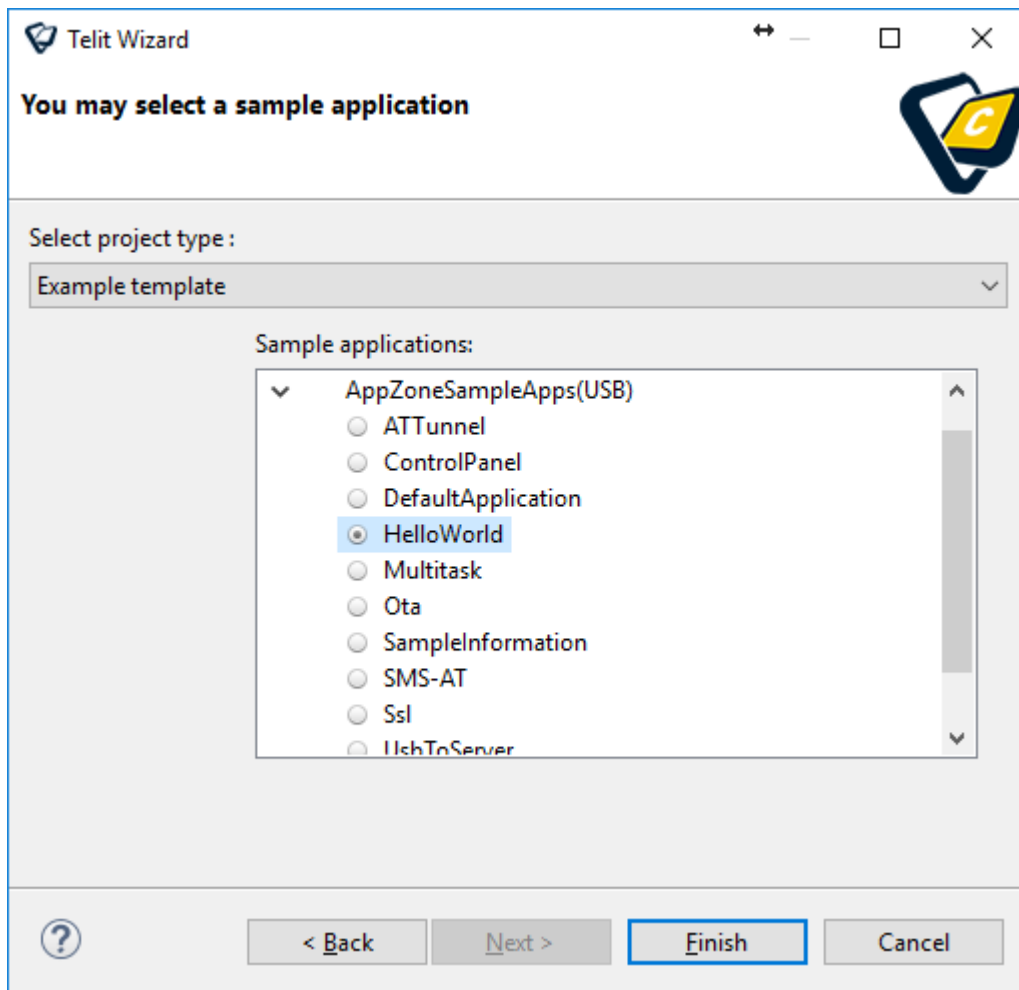
- f. In the **Firmware version** field, select the firmware version of your module.  
If the module is connected, the SDK identifies the firmware version of the connected module.

7. Click **Next**. A window enabling to select the project type opens.

8. Select the type of project that you want to create:

Empty project – Create an empty project. From the Select project type list, select **Empty Project**.

Example template – Create a project based on an example application. From the Select project type list, select **Example Template**. For a description of the sample applications click the question mark help button, or see [Sample Applications](#).



9. Select the checkbox next to the example application that you want to use as a basis for your project. For information on the example applications provided with AppZone C, see [Sample Applications](#).





## 5.2. Sample Applications

This section describes some of the sample applications provided with the AppZone C IDE. For a complete description on how to create a project containing the code of these examples, see [Create an Application](#).

For additional examples and information, visit: <https://www.telit.com/developer-zone/iot-app-zone/iot-app-zone-developer-resources/>.

For the full list of Sample Applications and their description, please see [15].

### 5.2.1. Hello World

The application prints "Hello World!" to the terminal every two seconds.

#### 5.2.1.1. *Application workflow*

##### **M2M\_main.c**

- Send to Process 1

##### **M2M\_proc1.c**

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

```

Welcome to Hello World Application
1. Hello World!
2. Hello World!
3. Hello World!
4. Hello World!
5. Hello World!
6. Hello World!
7. Hello World!
8. Hello World!

```

### 5.2.2. AT Tunnel

This application reads the AT Commands that were entered by the user and displays their results in the terminal.

Instead of using the AT-Commands parser, this application bypasses the parser and uses call back functions to present the commands and the results in the terminal.

#### 5.2.2.1. *Application workflow*

##### **M2M\_main.c**

- Set the AT commands parser
- Send ATE1 to enable commands echo
- Open USB/UART
- Configure AT mode
- Print welcome message and wait for incoming AT commands

##### **M2M\_atRsp.c**

- Print to USB/UART the AT command response

```

Welcome To AT-Commands Tunnel Application
Please enter AT-Commands:

at
OK

at+m2m?
+M2M: 1,10

OK

```

### 5.2.3. Sample Information

The application displays information about the module.

The application displays the module's Manufacturer, Model, IMEI, Factory S/N, SW version, FW version, IMSI, and MSISDN.

#### 5.2.3.1. Application workflow

##### M2M\_main.c

- Send to Process 1

##### M2M\_proc1.c

- Open USB/UART
- Call the function 'getInfo()'
- The function gets the module's information using the dedicated API and then
- Prints it.

```

Welcome To Sample Information Application

*****
* 11-1-15 12:17:3 - [INFO] - Manufacturer: Telit
* 11-1-15 12:17:3 - [INFO] - Model.....: HE910
* 11-1-15 12:17:4 - [INFO] - IMEI.....: 357164040636253
* 11-1-15 12:17:4 - [INFO] - Factory S/N.: 0000000599
* 11-1-15 12:17:5 - [INFO] - Version SW..: 0.9.5
* 11-1-15 12:17:5 - [INFO] - Version FW..: 12.00.005-8045
* 11-1-15 12:17:6 - [INFO] - IMSI.....: 425020358639273
* 11-1-15 12:17:6 - [INFO] - MSISDN.....: "+9725-----"
*****

```

### 5.2.4. Default Application

The application puts the module in its default status, displaying the module's information.

Afterwards, the user can write commands to the terminal via the AppZone layer.

#### 5.2.4.1. Application workflow

##### M2M\_main.c

- Send to Process 1

##### M2M\_proc1.c

- Set the AT commands parser
- Send ATE1 to enable commands echo
- Open USB/UART
- Call the function 'getInfo()'

The function gets the module's information using the dedicated API and then Prints it.

- Configure AT mode
- Wait for incoming AT commands

### M2M\_atRsp.c

- Print to USB/UART the AT command response

```

AppZone Default Application
General Information:
* 11-1-15 12:17:3 - [INFO] - Manufacturer: Telit
* 11-1-15 12:17:3 - [INFO] - Model.....: HE910
* 11-1-15 12:17:4 - [INFO] - IMEI.....: 357164040636253
* 11-1-15 12:17:4 - [INFO] - Factory S/N.: 000000599
* 11-1-15 12:17:5 - [INFO] - Version SW..: 0.9.5
* 11-1-15 12:17:5 - [INFO] - Version FW..: 12.00.005-8045
* 11-1-15 12:17:6 - [INFO] - IMSI.....: 425020358639273
* 11-1-15 12:17:6 - [INFO] - MSISDN.....: "+9725-----"

AT-Commands Mode, Please enter commands:

at+m2m?
+M2M: 1,10

OK

```

## 5.2.5. SMS-AT

The application allows the user to send AT Commands to the module as SMS messages and receive an SMS message with their response.

\* Before every command sent, the string "TelitAT:" must be entered.

Example: TelitAT:at+m2m?

### 5.2.5.1. Application workflow

#### M2M\_main.c

- Open USB/UART
- Enable receiving SMS indications using the function 'm2m\_sms\_enable\_new\_message\_indication()'
- Wait for incoming SMS messages

#### M2M\_net.c

- The function 'M2M\_onMsgIndEvent' is called when a new SMS message arrives
- Send to Process 1 with the message index

#### M2M\_proc1.c

- Set the variable 'key\_word' to the key word that should be sent as part of the SMS. In this example the key word is 'TelitAT:'
- Set the AT commands parser
- Set the SMS mode to text
- Read the SMS message
- Search for the key word ' TelitAT:' in the SMS message
- If the key word exists, execute the AT command that follows it.

## M2M\_atRsp.c

- Send SMS with the AT command response to the sending number

```

Welcome to SMS-AT Application!
Please send AT command through SMS ("TelitAT:AT-COMMAND").

SMS received: "TelitAT:at+m2m?"

Sending SMS to +972 ----- with answer:
+M2M: 1,10

OK

```

### 5.2.6. UART/USB to Server

The application sends text to a website via UART or USB.

You can see the text that was entered by going to the address:

<http://appzone.telit.com/Demo/GETserver/>

#### 5.2.6.1. Application workflow

##### M2M\_main.c

- Send to Process 1.

##### M2M\_proc1.c

- Set the variable 'APN' according to the network provider (for example: INTERNETG).



The APN must be set according to the SIM card network provider in order to connect to the internet.

- Set the variable 'SERVER' to the AppZone server address (APPZONE.TELIT.COM).
- Open USB or UART.
- Call the function 'get\_IP'.  
The function activates PDP context and gets the IP from the network.
- Set USB or UART callback.  
After setting the callback, the function 'hw\_usb\_read\_cb' (or 'hw\_uart\_read\_cb') will be called each time data is transferred over USB or UART.  
Wait for data to be sent over USB or UART.
- Gather the data that is received over USB or UART and wait until the 'Enter' key is entered.  
When the 'Enter' key is entered, call the function 'send\_data'.
- Call 'create\_socket' function  
The function creates a TCP socket that will be used to send data to the servers.
- Call 'create\_socket' function.  
The function sets the socket port (80 for HTTP) and the server Address (APPZONE.TELIT.COM).  
Then, it connects the socket.
- Send HTTP GET request to the server with the string sent over USB or UART.  
The HTTP GET request format is as follows:  
GET /Demo/GETserver/report.php?text=%s HTTP/1.1\r\n  
Host: %s\r\n  
\r\n\r\n\r\n\r\n
- Read and print the results from the server.

The following figures provide an example for sending information using UART to the server:

```

Welcome to UART-to-Server Application!
Getting IP...
IP Received: 46.210.113.117
Please enter your data (To send row, please press ENTER key):
Hello From UART-to-Server Application!
Sending Row...

Data Sent To Server: Hello%20From%20UART-to-Server%20Application!
Please go to 'http://appzone.telit.com/Demo/GETserver' to see the row.
Please enter data to send..
    
```

**Echo Text Demo**

**Recieved Text:**

[2015/01/11 15:17:50] Hello From UART-to-Server Application!

The following figures provide an example for sending information using USB to the server:

```

Welcome to USB-to-Server Application!
Getting IP...
IP Received: 46.210.140.18
Please enter your data (To send row, please press ENTER key):
Hello From USB-to-Server Application!
Sending Row...

Data Sent To Server: Hello%20From%20USB-to-Server%20Application!
Please go to 'http://appzone.telit.com/Demo/GETserver' to see the row.
Please enter data to send..
    
```

**Echo Text Demo**

**Recieved Text:**

[2015/01/11 15:12:20] Hello From USB-to-Server Application!

## 5.2.7. Control Panel

The application displays the module's information in the terminal and sends it to a website, where the information is presented.

You can see the module's information by going to this address:

<http://appzone.telit.com/Demo/GETserver//Demo/GETserver/Demo/ControlPanel/>

### 5.2.7.1. Application workflow

#### M2M\_main.c

- Send to Process 1

#### M2M\_proc1.c

- Set the variable 'TCP\_SERVER\_PORT' to 80 (HTTP port).
- Set the variable 'GPRS\_APN' according to the network provider (for example: INTERNETG).



The APN must be set according to the SIM card network provider in order to connect to the internet.

---

- Set the variable 'SERVER' to the AppZone server address (APPZONE.TELIT.COM).

- Open USB/UART

- Call the function 'get\_IP'

The function activates PDP context and gets IP from the network

- Call the function 'socketProcess'

- Create TCP socket

- Set the socket port (80 for HTTP) and server address (APPZONE.TELIT.COM)

- Connect the socket.

- Call the function 'UpdateSendString'

The function gets the module's information (IMEI, FW version, signal level, GPIO, etc.) and prepare the string to be sent.

- Send HTTP GET request to the server with the prepared string

The HTTP GET request format is as follows:

```
GET /Demo/ControlPanel/update.php?update=%s HTTP/1.1\r\n
```

```
Host: %s\r\n
```

```
\r\n\r\n\r\n\r\n
```

- Close the socket

```

Welcome to Control Panel Application!

Getting IP...

IP Received: 109.253.59.139

Sending Data To Socket:
IMEI: 357164040636253
IMSI: 425020358639273
ISSI: 27
Operator: Cellcom
SW version: 0.9.5
MSISDN: +972524605668
GPIO1: 0
GPIO2: 1

Sending buffer: GET /Demo/ControlPanel/update.php?update=357164040636253%7c4250203586
9273%7c27%7cCellcom%7c0.9.5%7c%2b9725 - - - - - %7c0%7c1 HTTP/1.1



Sending result = 134
Host result = 25
Receive result = 293

Receive buffer:


HTTP/1.1 200 OK
Date: Tue, 13 Jan 2015 15:49:01 GMT
Server: Apache
X-Powered-By: PHP/5.2.17
Cache-Control: must-revalidate, post-check=0, pre-check=0
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Pragma: no-cache
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html

1
1

Finish, Please check your web server at 'http://appzone.telit.com/Demo/ControlPanel'.
    
```

### Control Panel



**Unit #359785020176106**

Status: Updating  
Last update: October 8<sup>th</sup>, 10:57:16

**SIM Details:**  
IMSI: 425010801123438  
Phone#:

**Network Status:**  
Operator: 012 Mobile  
RSSI: 23

GPIO 1  
 GPIO 2

## 6. BUILD APPLICATIONS

Before compiling the application, in the M2M\_main.c file write some code to avoid warning messages.

You can change the project toolchain (GCC/RVCT) and the product (2G/3G/4G) at any time in the project properties.

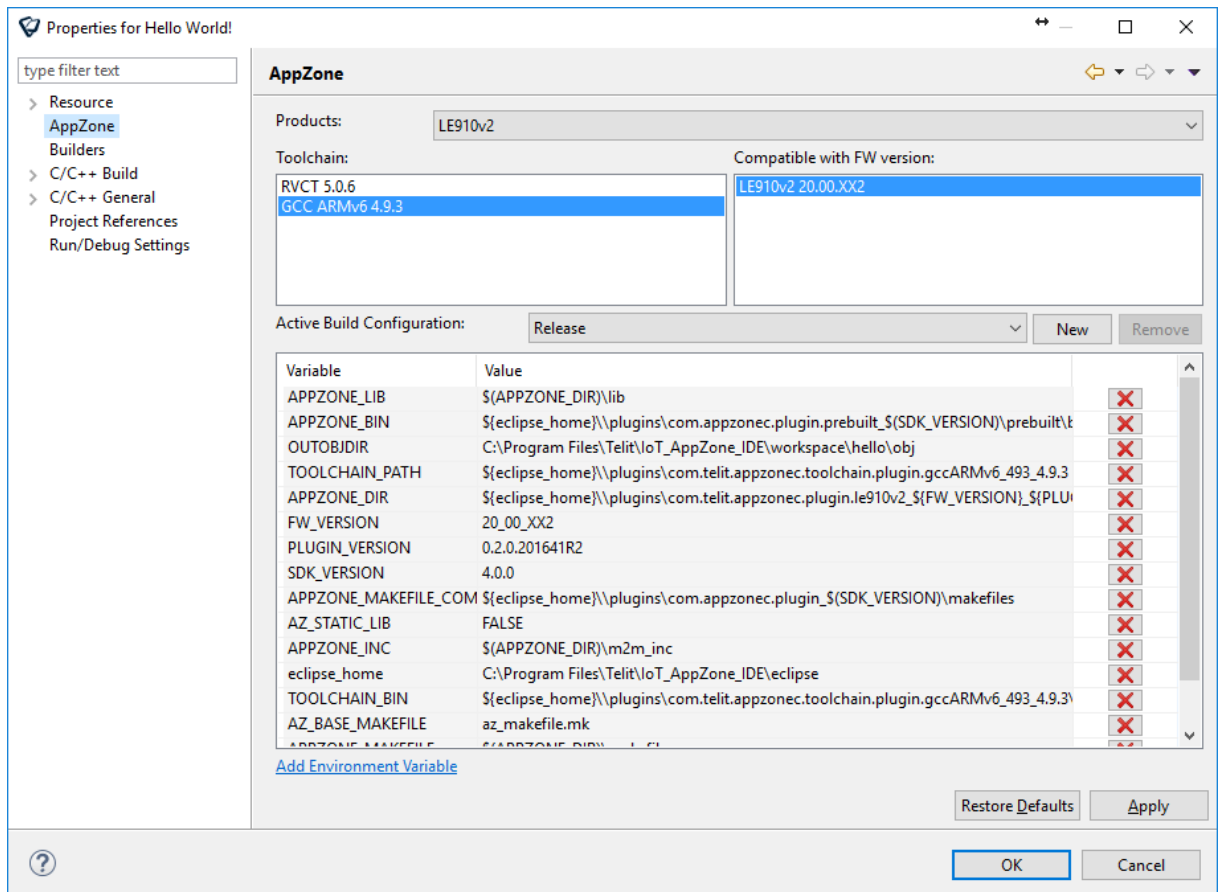
For information on the APIs, see *AppZone C API Reference Guide*.

### 6.1. Set Project Settings

You can change the project toolchain (GCC/RVCT) and the product (2G/3G/4G) at any time in the project properties.

To change the project settings:

1. In the Project pane, right-click the project and then select **Properties**. The Properties for project window opens.
2. Select the Product and Toolchain that you want to use for the app, and then click **OK**.



### 6.2. Compile the Project

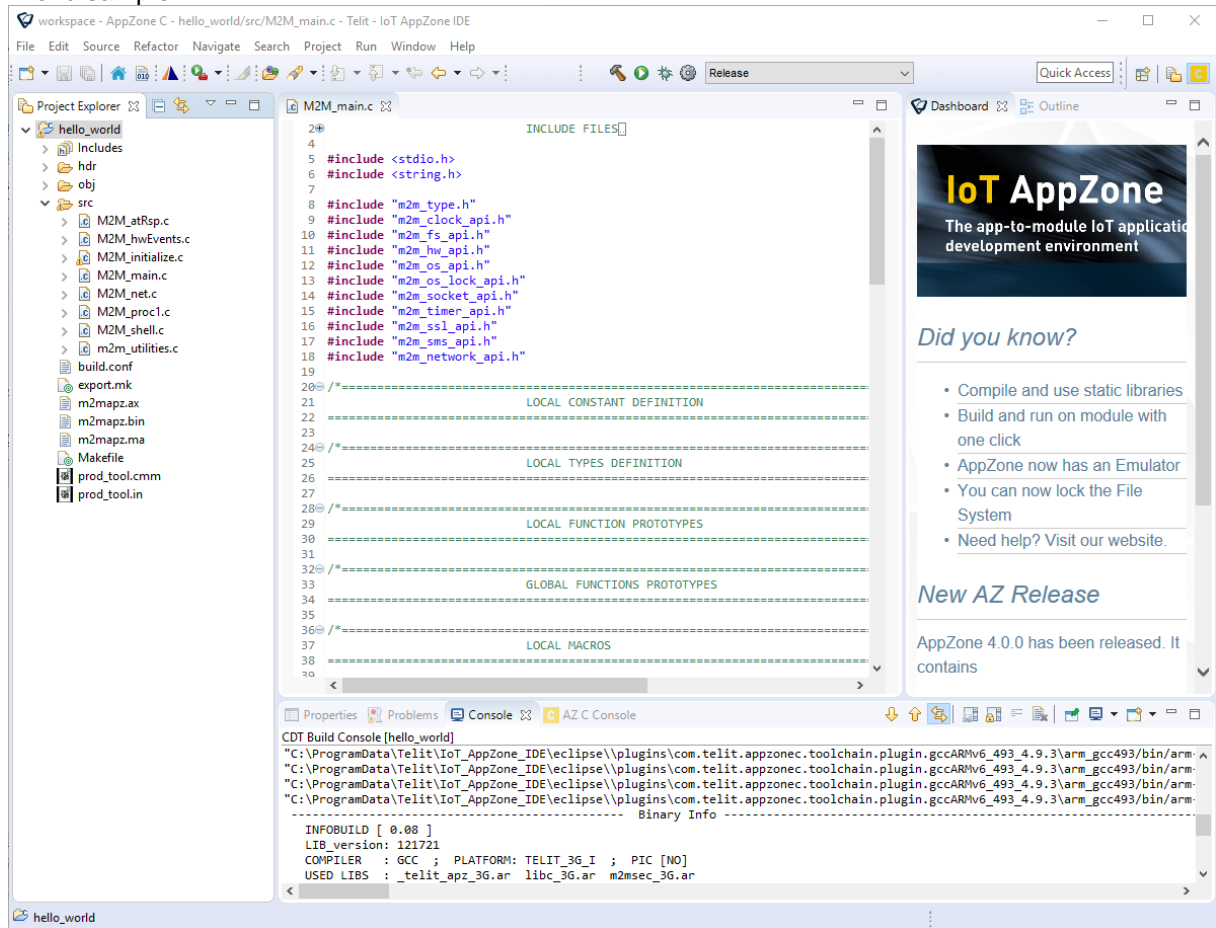
**To compile the project:**

1. Select the project and choose **Release** as the configuration (this is the default one)
2. Click the build button (🔨)
3. See the results of the build process in the **Console** view. Click on its tab in the Tools pane to bring the view forward if it is not currently visible.

If it is not visible, you can open it by selecting **Window > Show View > Console**.



The following figure shows an example of the compilation results of the Hello World sample:



The application that you built is located in the Project main folder. By default, the name of the example project application is m2mapz.bin.

# 7. CUSTOMIZE APPLICATION PROJECTS

You can customize the project using the Makefile file, export.mk file or add environment variables using the project properties.

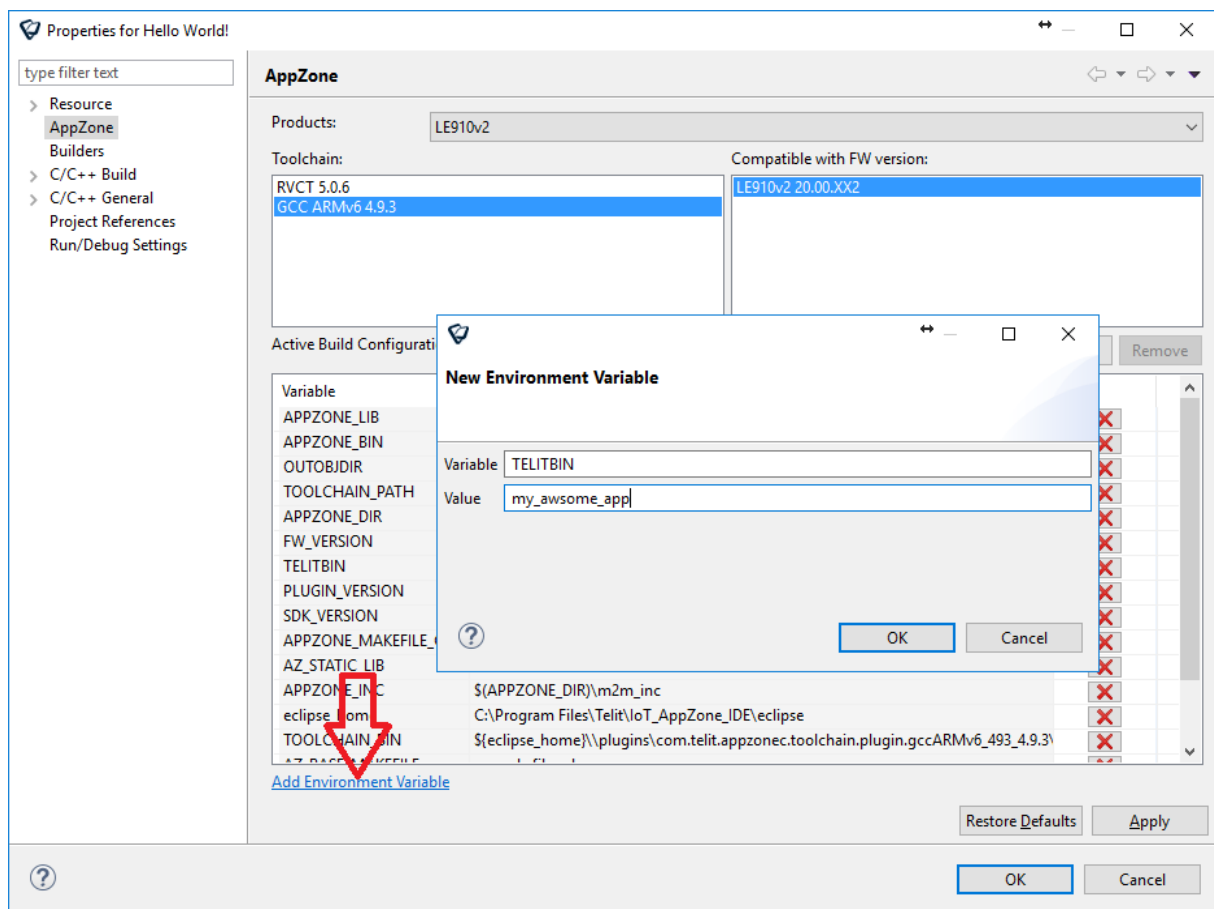
For example, to change the default name of the compiled application from m2mapz.bin to a custom name, add the TELITBIN environment variable:



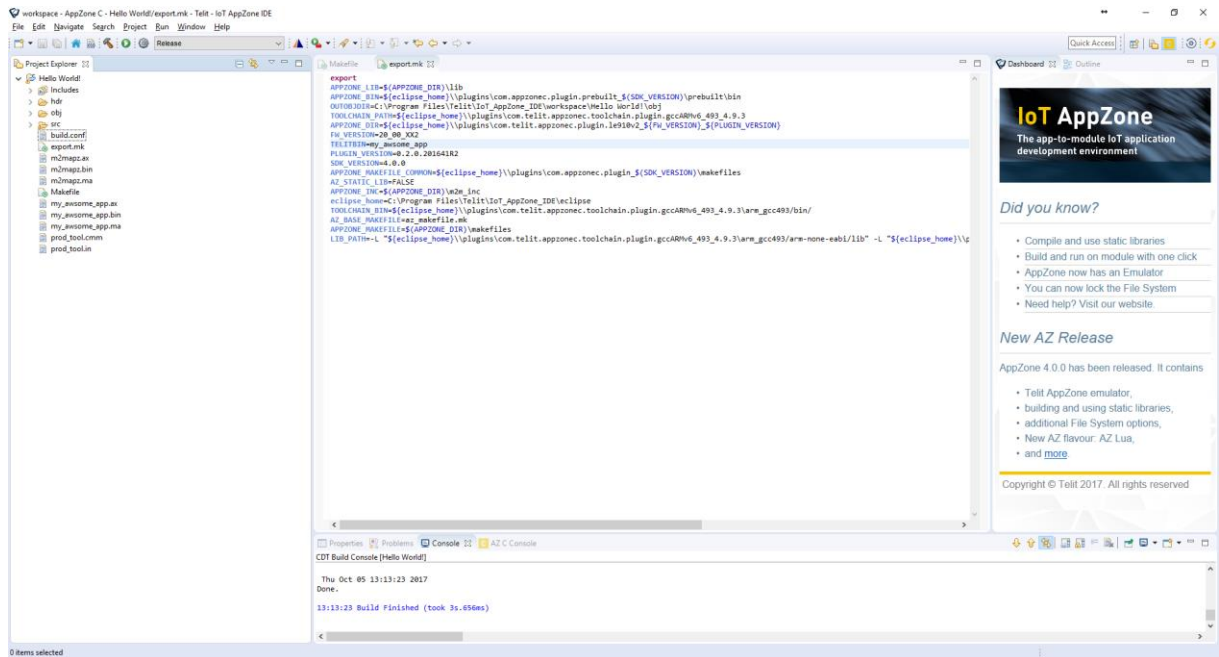
Changing the name of the compiled application would affect some features, like automatic deployment. If a new name is used the manual steps need to be followed first to ensure correct configuration.

Open the Project properties.

- Add the variable TELITBIN and assign it the value COMPANY\_NAME



Rebuilding the project will generate a binary with the custom name



## 8. RUN YOUR APPLICATION

You can build and run your application either on a connected Telit module or on Telit's AppZone Emulator. The emulator is installed by default during the AppZone C installation.

### 8.1. Run Applications on the Module – Automated process

With the **Release** configuration selected, you can click the **Run** button. This will automatically do the following:

1. Compile the project for the module
2. Search for and connect to an available port of the module
3. Copy the application to the module
4. Instruct the module to start the application



If you have more than one module connected, there is no way to specify which module to auto-connect to. Either leave only one module connected, or use the manual procedure.

The build bar looks like this:





The Debug option will not work for the module.

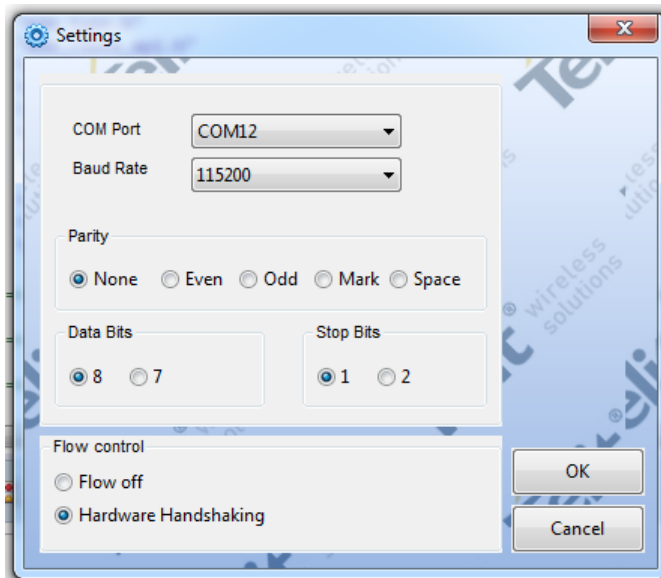
### 8.2. Run Applications on the Module – Manual process

After you have built your application, connect to the module and then upload and run your application. If you have uploaded more than one application, use the File Manager to control which application is currently active. See [Managing Files on the Module](#) for more information.

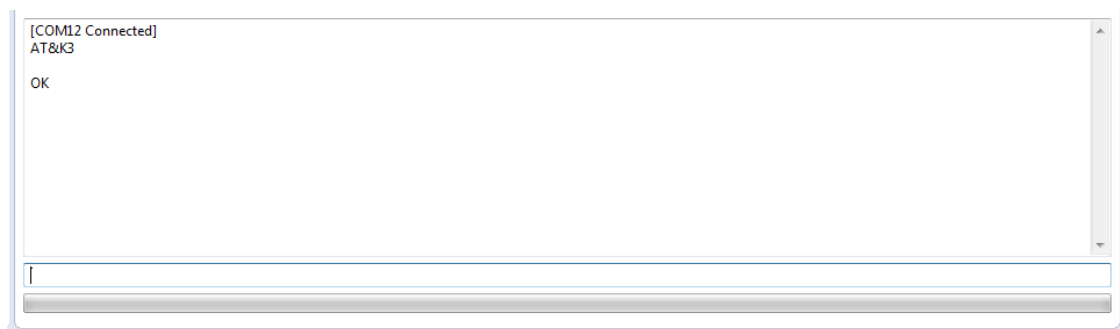
#### 8.2.1. Connect the COM Port

To run the application on the module:

1. In the Tool pane, open the AZ C Console.
2. Click **AutoConnect COM Port** (). The AZ C Console starts searching for available COM Ports.
3. If the module is not connected, you can:
  - a. Click **Settings** (). The Settings window opens.



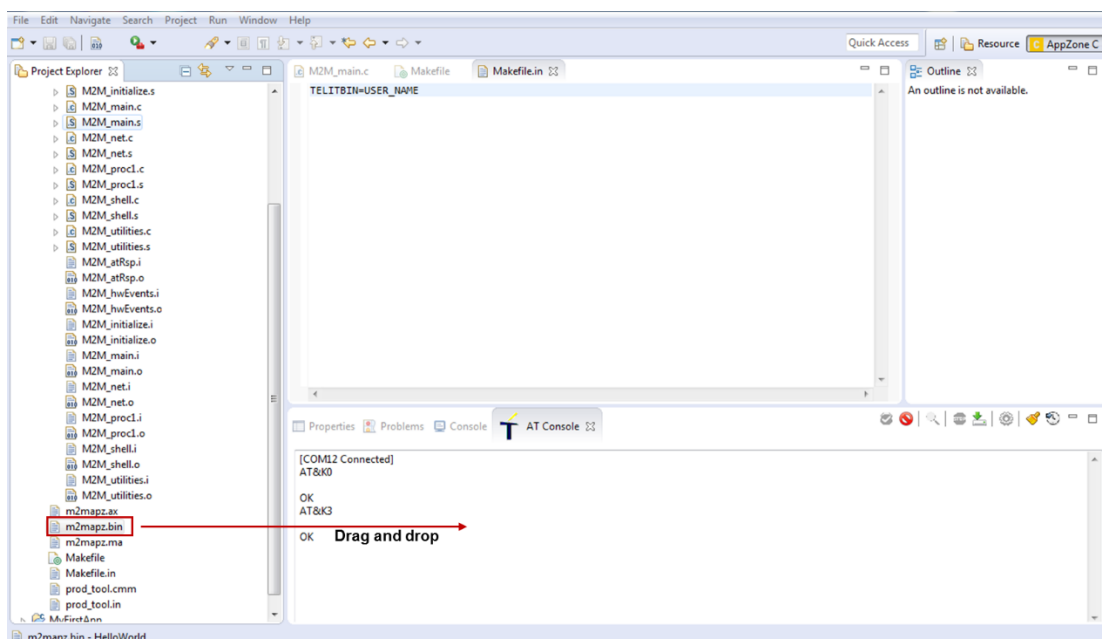
- b. In the COM Port field, select the COM port to which the module is connected. Usually COM12.
- c. Click **OK** to close the Settings window.
- d. Click **Connect COM Port** (with a green checkmark icon) to connect to the module. The AZ C Console displays a message if the connection succeeds.



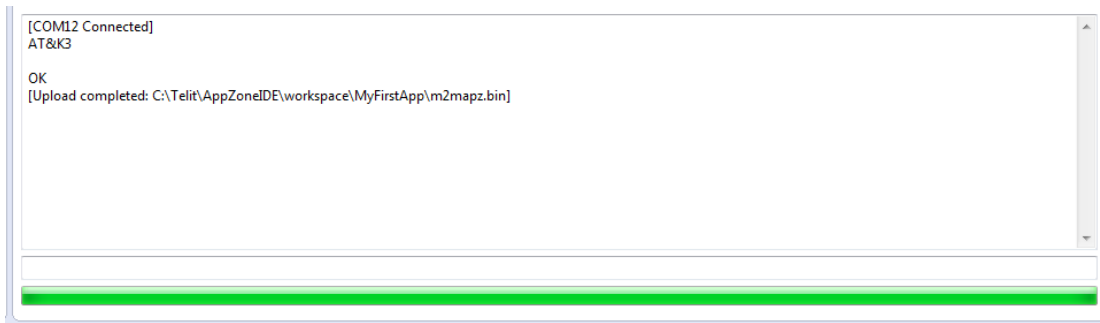
### 8.2.2. Upload and Run the Application

To upload the application to the module and then run the application:

1. Select the bin file of the application that you created (the default name is m2mapz.bin) and drag it to the AZ C Console.



A message notifies you that the upload was successful.

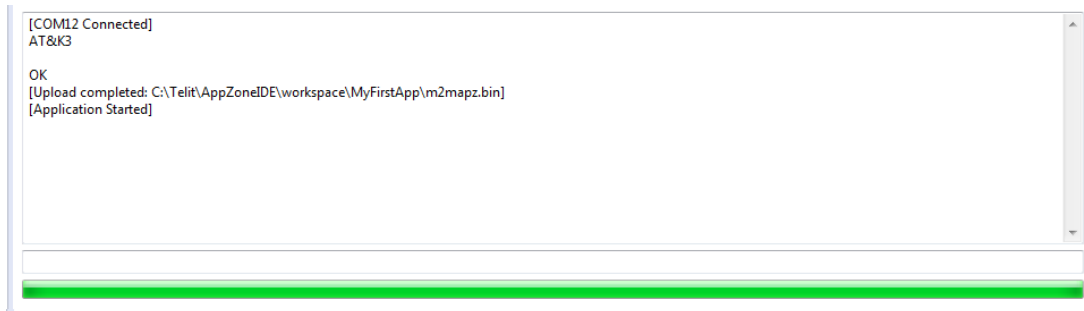


2. Set the application as the default application:
  - a. Right-click inside the AZ C Console and then select **File System > Change Dir -> \MOD**.
  - b. In the AZ C Console click **File Manager** (📁) to open the File Manager.
  - c. Right click the file and then select Set as **AppZone Application**.

For more information, see [Managing Files on the Module](#).

3. To run the application, right-click in the AZ C Console and then select **AppZone > Start Application > AT+M2M=4,10**.

A message notifies that the application has started.

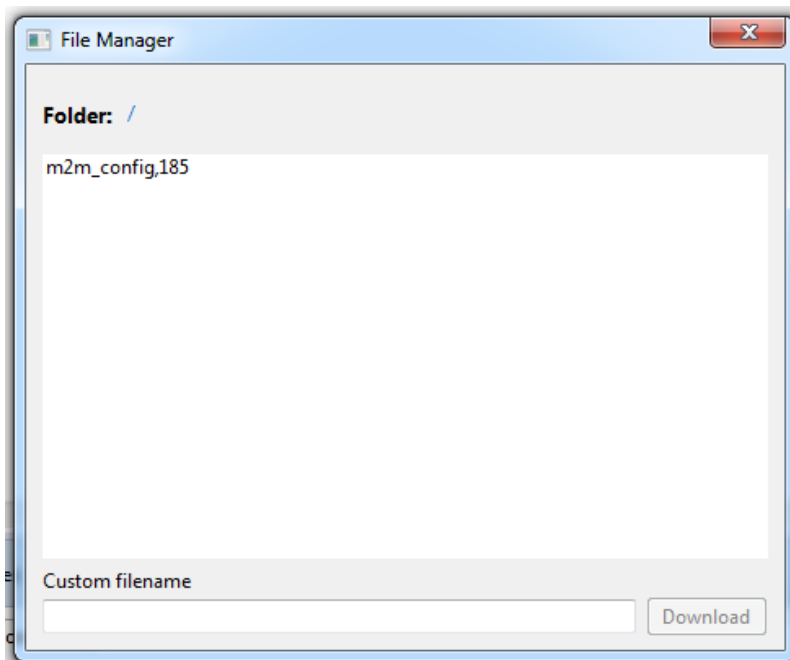


### 8.2.3. Managing Files on the Module

Using the File Manager you can access to the file system of the module and perform basic operations.

After uploading the application on the module:

1. Right-click inside the AZ C Console and then select **File System > Change Dir -> \MOD**.
2. In the AZ C Console click **File Manager** (📁) to open the File Manager.



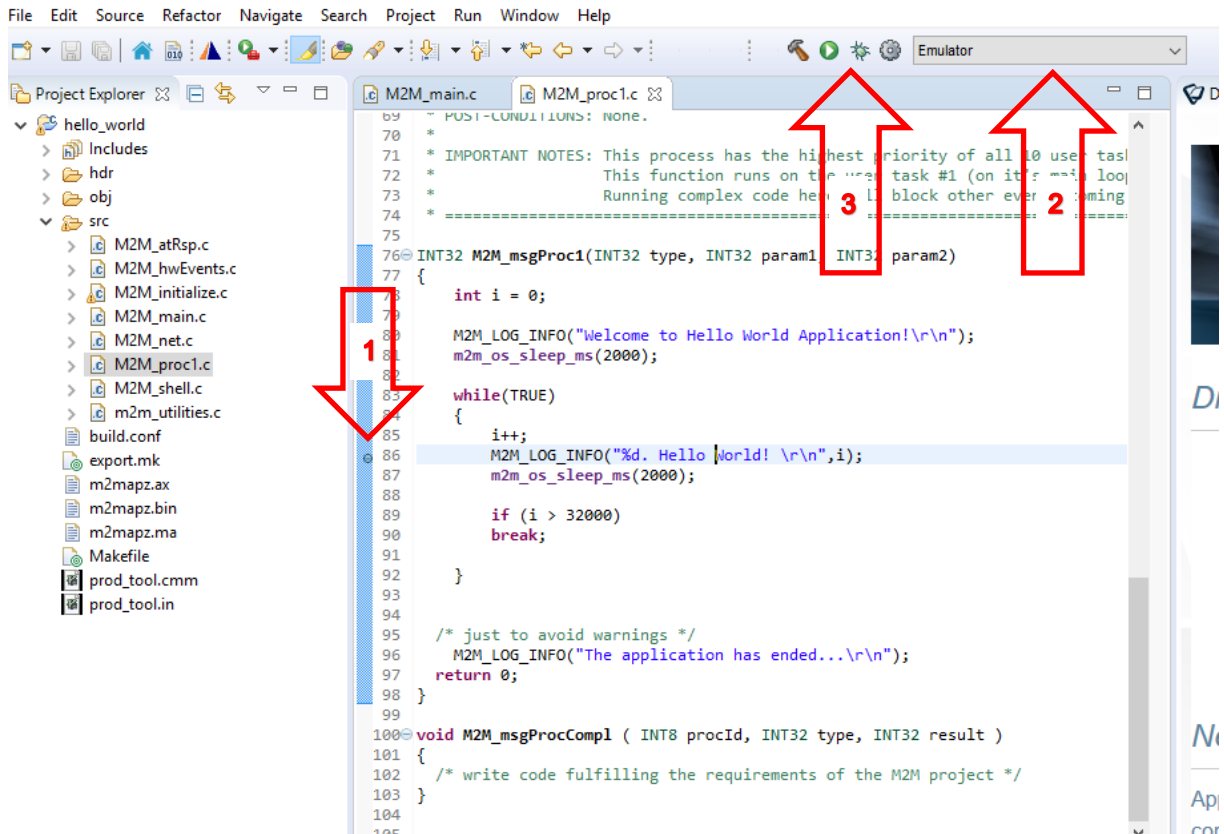
To remove a file, right click the file and then select **Remove**.

To set the file as the active application, right click the file and then select Set as **AppZone Application**.

### 8.3. Run and Debug Applications on Emulator

To launch the emulator with your custom binary,

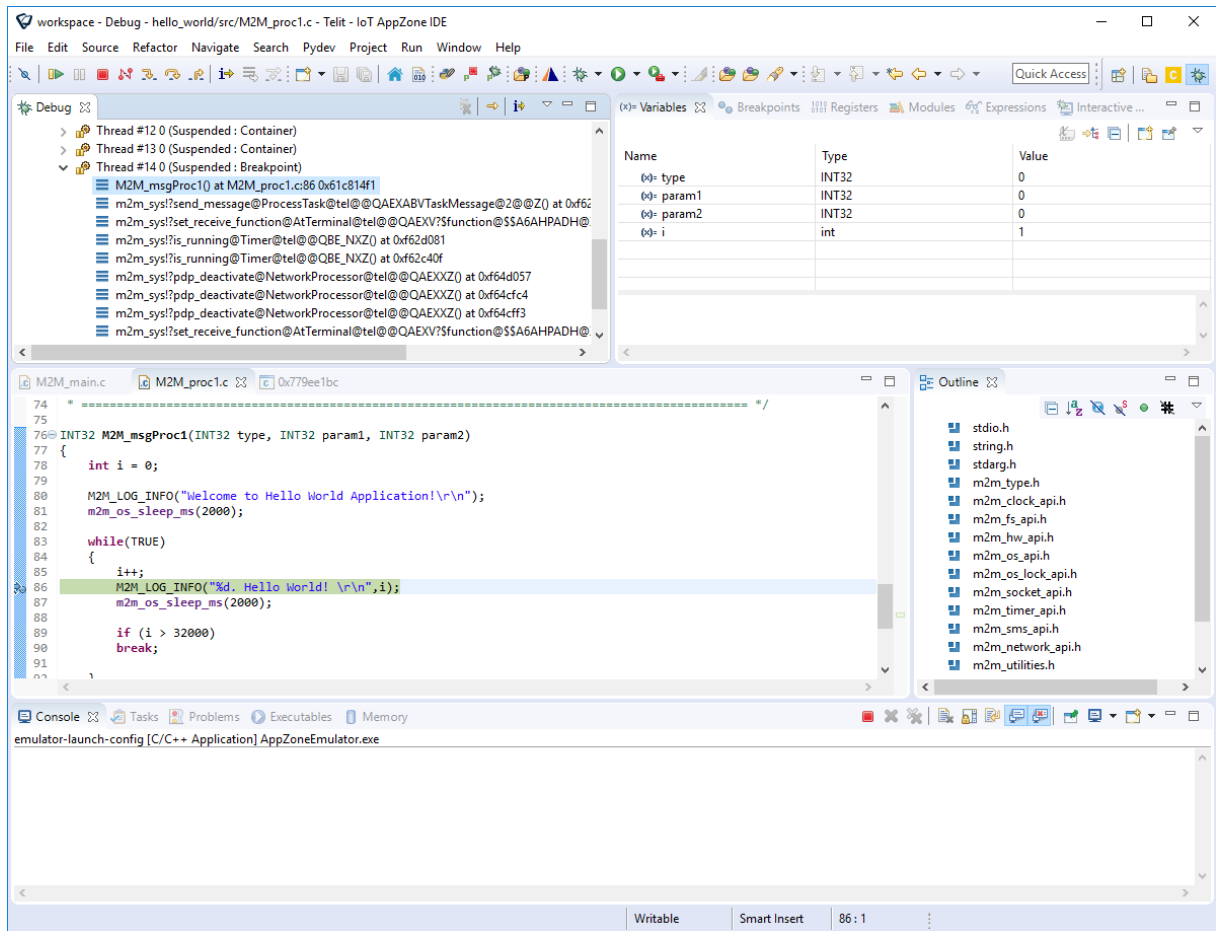
1. [Optional] Add a breakpoint on the source code.
2. From the toolbar, select “Emulator”.
3. Then press either the “Build and Run”, or the “Build and Debug” button on the toolbar.



4. The binary is then automatically started on the emulator



If you have placed a breakpoint and pressed the Debug button, the IDE will offer to switch to debug mode and show the current state of your app. You can check the value of variables and step through the program.



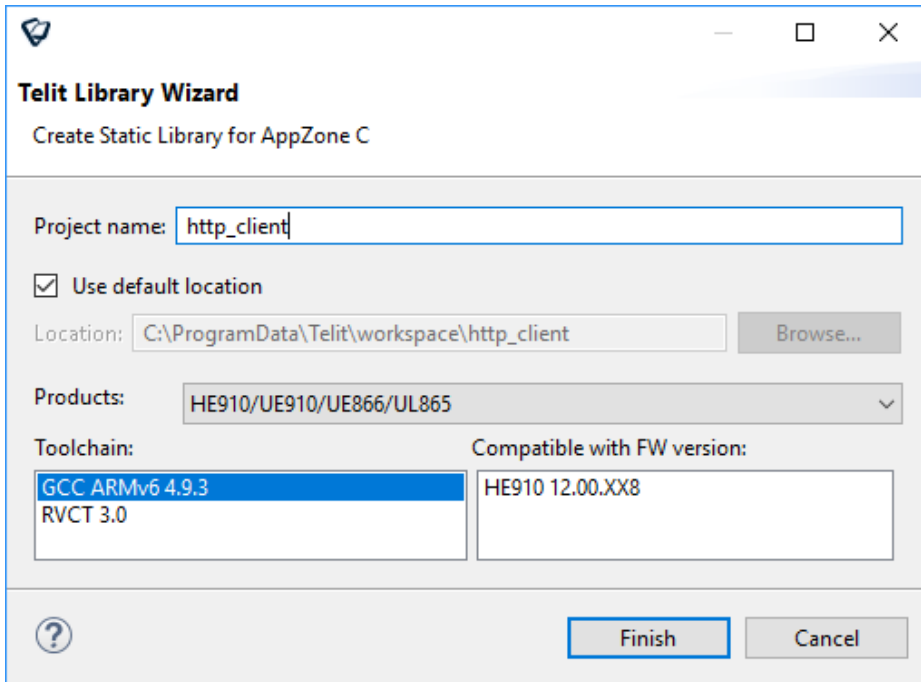


# 9. STATIC LIBRARIES

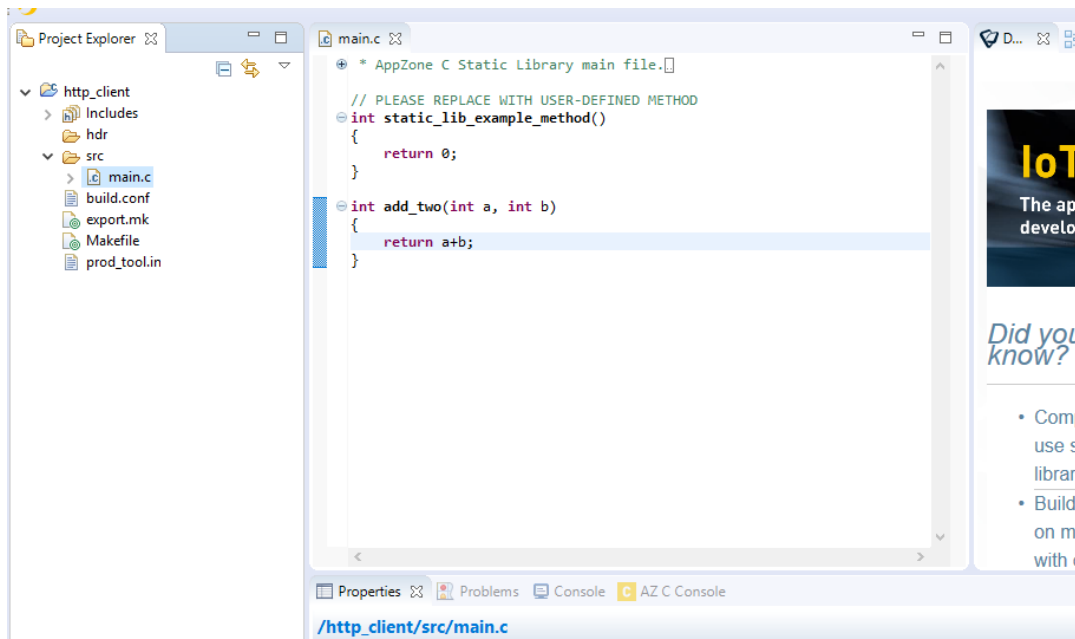
The AppZone IDE allows you to create static libraries which you can distribute as binaries and import into other projects.

## 9.1. Creating an AppZoneC static library

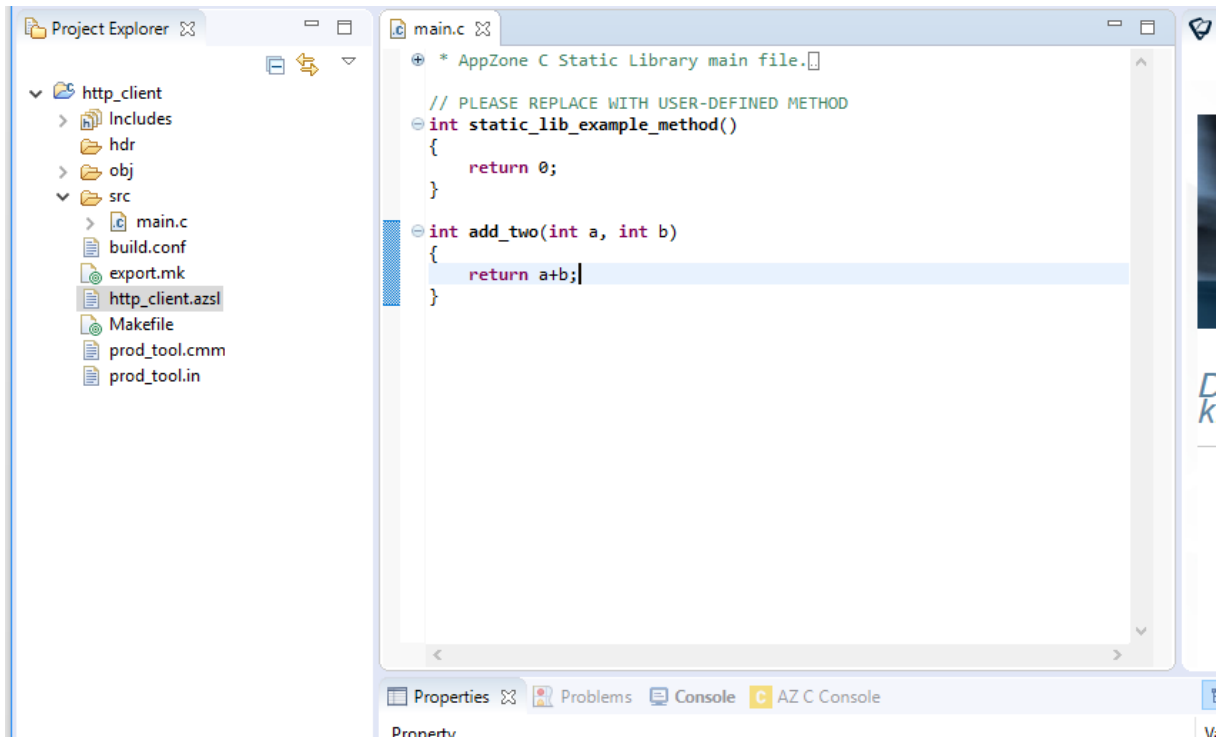
Users can create static libraries using the Telit Library Wizard. A product, toolchain and FW version has to be chosen. If the library is to be distributed for several configuration, it needs to be rebuilt for each configuration.



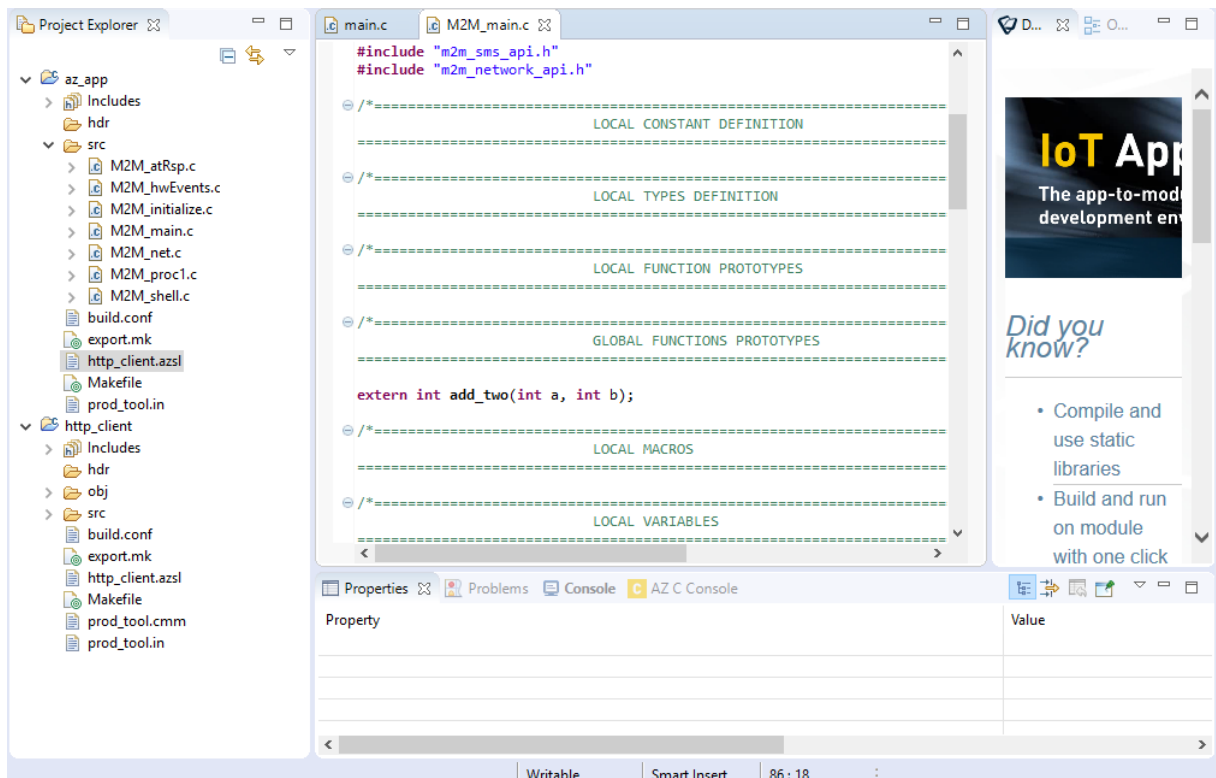
The wizard creates an AZC static lib project with a main.c in which the user can implement the static library functions. The user can rename the main source file or add/remove as many source files as he wants as long as they are in src folder.



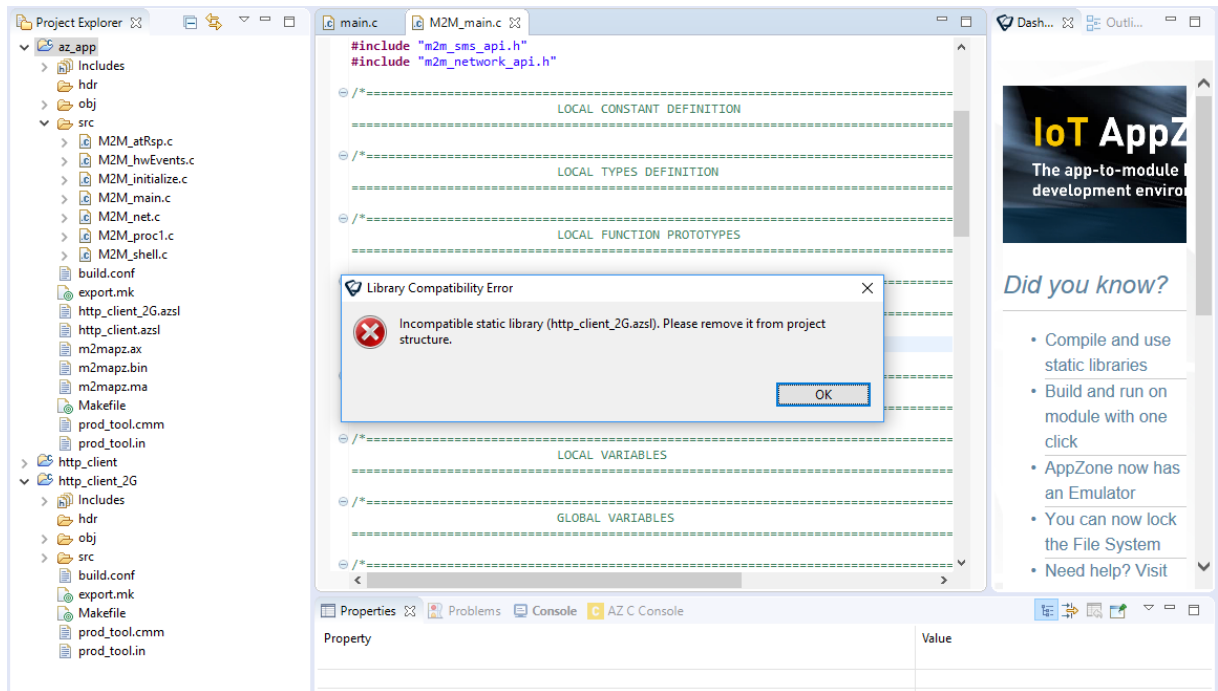
Static library compilation produces an archive with .azsl (AppZone Static Library) extension which basically contains the static library file (.a) and a metadata text file in order to check the library's integrity as well as compatibility when linked against projects targeted for incompatible firmware.



The produced azsl can be simply dragged and dropped into the structure of a compatible (3G in our example) AppZoneC project and project developer can use the needed library functionality. When distributing the azsl file it is recommended to provide a header file with the API implemented in the library, otherwise the project code can use the extern keyword to access the functions.



In case a user imports an incompatible library to a project the SDK expects an error will be produced to inform the user that the incompatibility was spotted..



# 10. UNDERSTAND THE SKELETON FILES

The following sub-chapters describe how to use the functions included in the .c files of the skeleton. Refer to document *AppZone C API Reference Guide* to have more information on the number of AppZone tasks and M2M\_msgProcX callback functions.

The following figure shows the files of the skeleton and the "Includes" folder; the developer fills the files with code, in accordance with the requirements, to develop the M2M application. The prod\_tool.ini file contains information on products and toolchain.

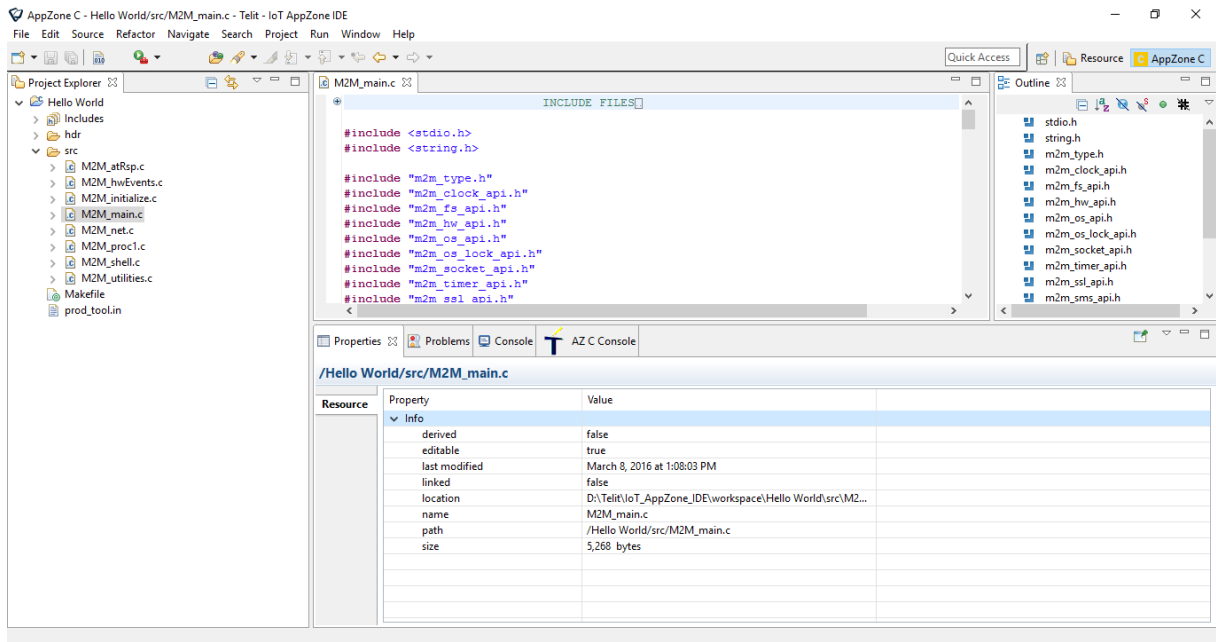


Fig. 4: Skeleton with the Header Files (Includes folders)

## 10.1. M2M\_initialize.c File

The M2M\_initialize.c file contains the following callback functions:

Callback	Event starting the callback:
InitUserInterface(...)	is executed on the startup of the user M2M application. The user may configure the application modifying this function.
M2M_onAppUpgradeAvailable(...)	for future use.
M2M_initGPIO(...)	for future use.

## 10.2. M2M\_proc1.c (default) M2M\_procX.c Files

The M2M\_proc1.c file (default) contains the following callback functions:

Callback	Event starting the callback:
M2M_msgProc1(...)	is the default callback. It is called by Task_1, see chapter <b>Error! Reference source not found.</b>
M2M_msgProcComp1(...)	is called by the control when the generic (default or user) M2M_msgProcX(...) callback has finished its execution.

You can write custom M2M\_procX.c files that may contain one or more M2M\_msgProX(...) callback functions that are called by the connected AppZone Tasks.

The AppZone tasks, calling the M2M\_msgProcX(...) callback, execute the code written by the user. The user may dedicate each task to a well-defined group of activities in accordance with the M2M application requirements and the needed execution priority.

Task\_1 has the higher priority and Task\_32 has the lower priority of the 32 tasks that can be provided by the AppZone Software Layer. The AppZone tasks have lower priorities than the modem tasks.

Example:

- Use the m2m\_timer\_create(...) API and its callback function to manage the timer expiration. Start the timer with the m2m\_timer\_start(...) API;
- When the timer is expired, the control executes the callback function to manage the timer expiration. Its code, written by the developer, in accordance with the timer identifier sends a message, for example, to Task\_1 using the m2m\_os\_send\_message\_to\_task(...) API;
- Task\_1 detects the received message in its queue and calls the M2M\_msgProc1(...) callback function that executes the code written by the developer;
- M2M\_msgProc1(...) callback function, for example, sends a message to Task\_5. In turn, it detects the message and calls the M2M\_msgProc5(...) callback function.

### 10.3. M2M\_main.c File

The M2M\_main.c file contains the following callback functions:

Callback	Event starting the callback:
M2M_main (...)	is the first function that the control executes when the M2M application is started. Refer to document [2] to have information on “argc” and “argv” parameters.
M2M_suspend (void)	will be used for future internal debugging activity.
M2M_resume (void)	will be used for future internal debugging activity.

### 10.4. M2M\_arRsp.c File

The M2M\_arRsp.c file contains the following callback function.

Callback	Event starting the callback:
M2M_onReceiveResultCmd(...)	When an AT command is entered into the module, it is executed by the AT parser. The callback function captures the AT command response generated by the AT parser. The code, written in the callback by the user, in accordance with the AT response performs the programmed action. The callback works either in Command and Data mode.

M2M\_onReceiveResultCmd(...) example:

- Send an AT command to AT parser by means of the m2m\_os\_iat\_at\_command(...) API ;
- AT parser executes the received AT command and the M2M\_on\_ReceiveResultCmd(...) callback is called;
- M2M\_onReceiveResultCmd(...) callback executes the code written by the developer in accordance with the AT command result.

### 10.5. M2M\_hwEvents.c File

The M2M\_hwEvents.c file contains the following callback functions:

Callback	Event starting the callback:
M2M_onWakeup(...)	expiration of the alarm time
M2M_onInterrupt(...)	interrupt created by a GPIO
M2M_onHWTimer(...)	expiration of the hardware timer

M2M_onUSbCableEvent(...)	plugging in/unplugging USB cable
M2M_onI2CEvent(...)	for future use
M2M_onKeyEvent(...)	actions on the "ON" key

M2M\_onWakeup(...) example:

- Use the `m2m_rtc_set_date(...)` and `m2m_rtc_set_time(...)` APIs to set the date and time of the module. Use the `m2m_rtc_set_alarm(...)` API to set the date and time of the alarm;
- When the date/time alarm is reached the `M2M_onWakeup(...)` callback is executed.

M2M\_onInterrupt(...) example:

- Connect GPIO X with GPIO Y using a wire;
- Use the `m2m_hw_gpio_write(...)` API to force to 0 the GPIO X. Use the `m2m_hw_gpio_int_enable(...)` to configure the GPIO Y as a source of interrupt. Force to 1 the GPIO X via the `m2m_hw_gpio_write(...)` API;
- When the transition on the GPIO Y is detected, the `M2M_onInterrupt(...)` callback is executed.

M2M\_onHWTimer(...) example:

When one or more hardware timers are expired, the control calls the `M2M_onHWTimer(...)` callback. The developer writes in the callback the code that will be executed in accordance with the identifier of the expired timer.

- Use the `m2m_hw_timer_start(...)` API to start one or more hardware timers;
- When an hardware timer expires the `M2M_onHWTimer(...)` callback is activated;
- `M2M_onHWTimer(...)` callback executes the code written by the developer in accordance with the identifier of the expired timer.

M2M\_onUSbCableEvent(UINT32 usb\_cable\_event) example:

The function checks the plugging in/unplugging of the USB cable.

- When the USB cable is plugged in, the control calls the callback with the "usb\_cable\_event" parameter equal to 1.
- When the USB cable is unplugged, the control calls the callback with the "usb\_cable\_event" parameter equal to 0.

M2M\_onKeyEvent(INT32 val1, INT32 val2) example:

The function checks the "ON" key that turns on/off the module.

- When the "ON" key is pushed, the control calls the callback with the "val1" parameter equal to 1 (key is pressed) and the "val2" parameter equal to 12 (key code of the "ON" key).
- When the "ON" key is released, the control calls the callback with the "val1" parameter equal to 0 (key is released) and the "val2" parameter equal to 12 (key code of the "ON" key).
- If the "ON" key is pushed down for a long time the control calls the callback with the "val1" parameter equal to 2 (key is held down) and the "val2" parameter equal to 12 (key code of the "ON" key). The callback is continuously called. Use the UART API to print out key status and code.

## 10.6. M2M\_net.c File

The `M2M_net.c` file contains the following callback functions:

Callback	Event starting the callback:
<code>M2M_onNetEvent(...)</code>	PDP context activation/deactivation, SOCKET closed /error, etc.
<code>M2M_onRegStatusEvent(...)</code>	Cell change
<code>M2M_onGprsRegStatusEvent(...)</code>	GPRS registration update
<code>M2M_onMsgIndEvent(...)</code>	reception of a SMS
<code>M2M_onIP6RawEvent(...)</code>	reception of an ip6 raw packet

`M2M_onNetEvent(...)` example:

- Use the `m2m_timer_create(...)` API and write its callback function to manage the timer expiration. Start the timer through the `m2m_timer_start(...)` API;
- When the timer is expired, the control calls the function to manage the timer expiration. Its code, written by the developer, in accordance with the timer identifier sends a message to the `Task_1` using the `m2m_os_send_message_to_task(...)` API;
- `Task_1` detects the received message in its queue and calls the `M2M_msgProc1(...)` callback function that executes the code written by the developer;
- `M2M_msgProc1(...)` callback, using the `m2m_pdp_activate(...)` API, create a PDP context. To wait for the module registration and the PDP context activation the `M2M_msgProc1(...)` starts a timer. When the timer expires, the control sends again a message to `Task_1` to check the PDP status.
- When the PDP is active, the `M2M_onNetEvent` callback is executed. In general, this callback is activated by several network events. It is responsibility of the developer to wait for the desired event.

`M2M_onRegStatusEvent(...)` example:

- Enable the notification of the location registration by means of the `m2m_network_enable_registration_location_unsolicited()` API;
- When `cell_id` or LAC changes, the `M2M_onRegStatusEvent(...)` callback is executed.

`M2M_onGprsRegStatusEvent(...)` example:

- Enable the notification of the GPRS registration by means of the `m2m_network_enable_gprs_registration_location_unsolicited()` API;
- When GPRS registration change, the `M2M_onGprsRegStatusEvent(...)` callback is executed.

`M2M_onMsgIndEvent(...)` example:

- Enable the messages indication by means of the `m2m_sms_enable_new_message_indication(...)` API;
- Send a SMS to the module;
- When the module receives the SMS message, the `M2M_onMsgIndEvent(...)` callback is executed.

`M2M_onIP6RawEvent(...)` example:

- Enable the ip6 raw mode by means of the `m2m_ip6_raw(...)` API;
- When the module receives an ip6 packet in raw mode, the `M2M_onIP6RawEvent(...)` callback is executed.

## 10.7. M2M\_shell.c File

The `M2M_shell.c` file contains the following callback function:

Callback	Event starting the callback:
<code>M2M_cmdShell(...)</code>	for future use

# 11. OVER THE AIR UPGRADE (OTA)

Telit modules feature the TCPATRUN service allowing the user to run on the module the AT Commands sent from a PC through TCP/IP protocol. Examples: the PC sends the AT Command to the module to collect information concerning the current Network Operator on which the module is camped, or the AT Command to set/read the selected GPIO. To have more information on the TCPATRUN service refer to document [8].

To perform remotely the M2M user application uploading, the module must be configured in TCPATRUN server mode, and the PC must run a terminal emulator. The PC works as an IP client and Telnet is used as a connection method.

Here is an example to configure the TCPATRUN service in server mode. To have detailed information on the AT commands syntax refer to documents [6]/[7].

```

AT+CGATT?                ← returns the GPRS service state
+CGATT:1                 ← attached
OK

AT+CGDCONT=1,"IP","myAPN" ← sets up PDP context
OK

AT#SGACT=1,1             ← activates the context
#SGACT: 2.192.101.140    ← IP address assigned by the Network.
OK

AT#TCPATRUNL=0           ← disables TCPATRUN service
OK

AT#SCFG=1,1,1500,0,600,50 ← sets socket configuration. Inactivity timeout is set to 0
OK

AT#TCPATRUNCFG=1,2,1024,1024,"",1,5,1,5,2 ←sets the TCPATRUN service, listening on
OK                                     port 1024. Instance number #21 is assigned to
the service.

AT#TCPATRUNFRWL=2        ← drops the old firewall configuration
OK

AT#TCPATRUNFRWL=1,"000.000.000.000","000.000.000.000" ← sets up new firewall

```

<sup>1</sup> The modules provide three AT Commands Parser Instances that interpret and execute the AT commands received on the assigned channel communication, and return the relative results; the three instances are independent and named #1, #2, and #3. The TCPATRUN services (channel communication) can be connected to the #2 or #3; the default instance is the #2.



OK  
configuration

AT#TCPATRUNAUTH=2 ← drops old authentication parameters  
OK

AT#TCPATRUNAUTH=1,"USERTEST","TEST" ← sets up new authentication parameters  
OK

AT#TCPATRUNL=1 ← starts TCPATRUN service in server mode  
OK

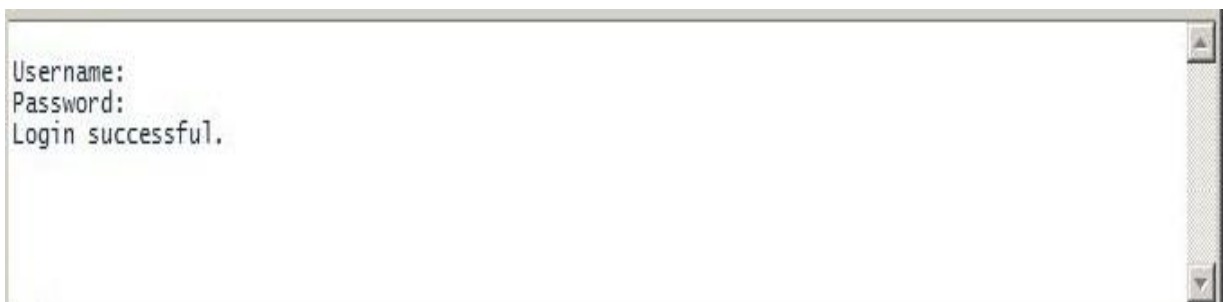
AT#SS ← check the Socket Status  
#SS: 1,4, 2.192.101.140,1024  
#SS: 2,0  
#SS: 3,0  
#SS: 4,0  
#SS: 5,0  
#SS: 6,0  
OK

You have to install on the PC a Terminal Emulator to perform the following activities:

- Create a Telnet connection (supporting TCP option) to send AT Commands and receive the related results.
- Transfer the M2M user AppZone application.

Following the GUI used by your selected Terminal Emulator, configure the Telnet session using the IP address returned by the AT#SGACT command, the port 1024, and start the connection.

After connection, enter the Username and Password used to configure the module in TCPATRUN server mode.



Now, enter next command to begin the uploading:

AT#M2MWRITE="m2mapztest.bin",81524,1

After the module in server mode has responded with the ">>>" prompt, you can send the AppZone application using the Transfer menu provided by your Terminal Emulator, select the Send Binary File option.

```

Username:
Password:
Login successful.

>>>
Progress
  Filename:  m2mapztest.bin
  Filesize:  79.6 KB
  Bytes sent: 40.4 KB (50%)

```

On success, the module returns the OK message.

Referring to the following screenshot.

The first OK is the result of the uploading, described above.

Use AT#M2MCHDIR="\MOD" to enter the \MOD directory, the second OK is its result.

Now, use AT#M2MLIST to list the just transferred file.

```

Username:
Password:
Login successful.

>>>
OK

OK

#M2MLIST: <.>
#M2MLIST: <.>
#M2MLIST: "m2mapz.bin", 81524
#M2MLIST: "m2mapztest.bin", 81524
#M2MLIST: free bytes: 5988352

OK

```

# 12. DEBUG APPLICATIONS USING LAUTERBACH

You can debug applications from within Eclipse using Lauterbach development tools.

## 12.1. Hardware and software requirements

The following table lists the requirements for Lauterbach:

Software requirements	
TRACE32® PowerView for ARM	Software_Version:_P.2015.02.000062632 Build:_60219—62632 or later
Hardware requirements	
Lauterbach debugger Emulator	LA-3505
Header plus cable	LA-7765 (JTAG ARM11)

For additional Lauterbach products and support see <http://www.lauterbach.com/frames.html?home.html>.

We suggest that you install Trace 32 software using the default path (in most cases C:\T32) because it prevents additional configuration and reduces configuration errors.

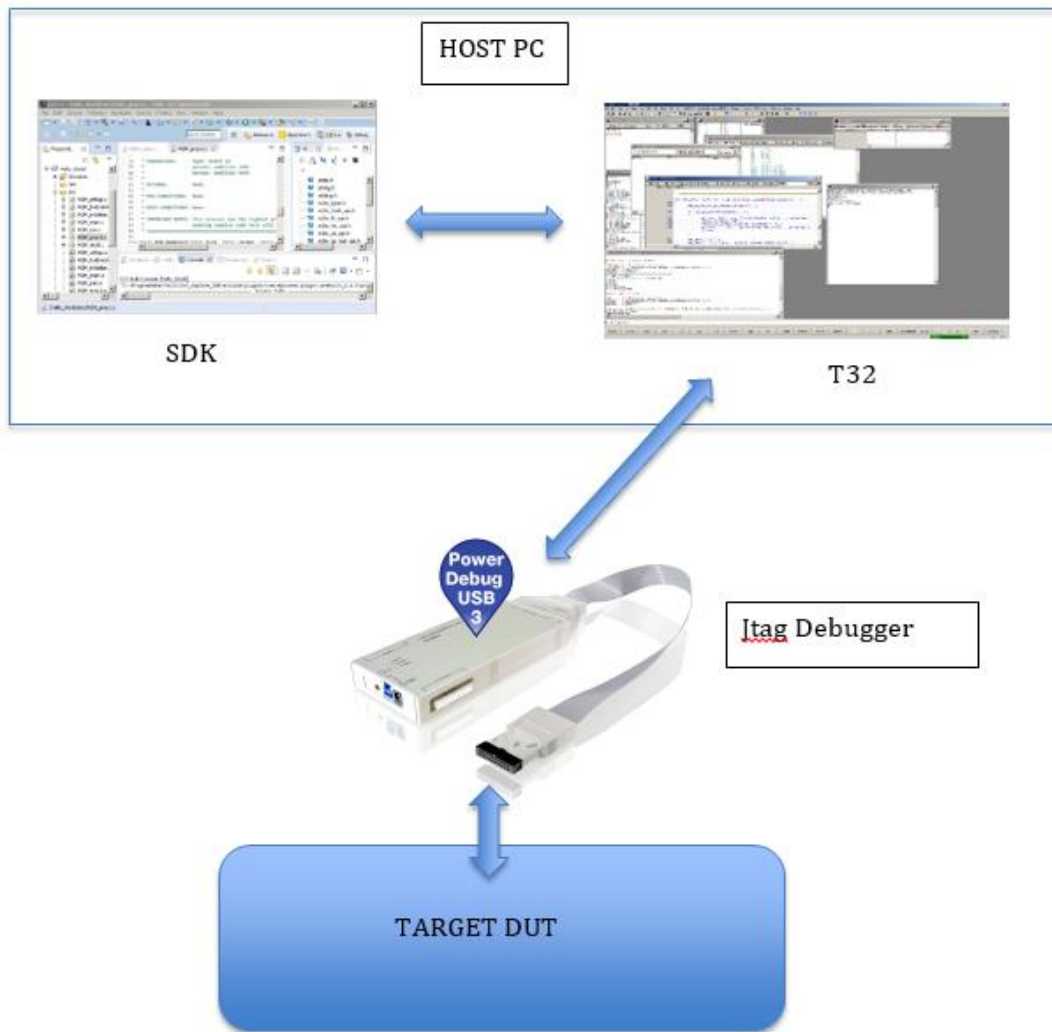
## 12.2. Initial configuration

To configure AppZone Linux to work with Trace32:

1. Navigate to the following folder:  
`..\IoT_AppZone_IDE\eclipse\plugins\com.appzonec.plugin.prebuilt_<version>\lauterbach`  
 where <version> is the current version AppZone Linux version, such as 2.1.4:
2. Copy the t32.cmm file.
3. Navigate to the folder in which you installed Trace32 (C:\T32).
4. Rename the original installed file to t32\_orig.cmm.
5. Paste the file you just copied.

The t32.cmm file allows communication between the SDK and the T32 environment. Eclipse calls the t32.cmm through an internal socket at debug session startup, and redirects the commands to the SDK scripts that manage the debugging processes.

The following figure displays an example of the data flow during debug:

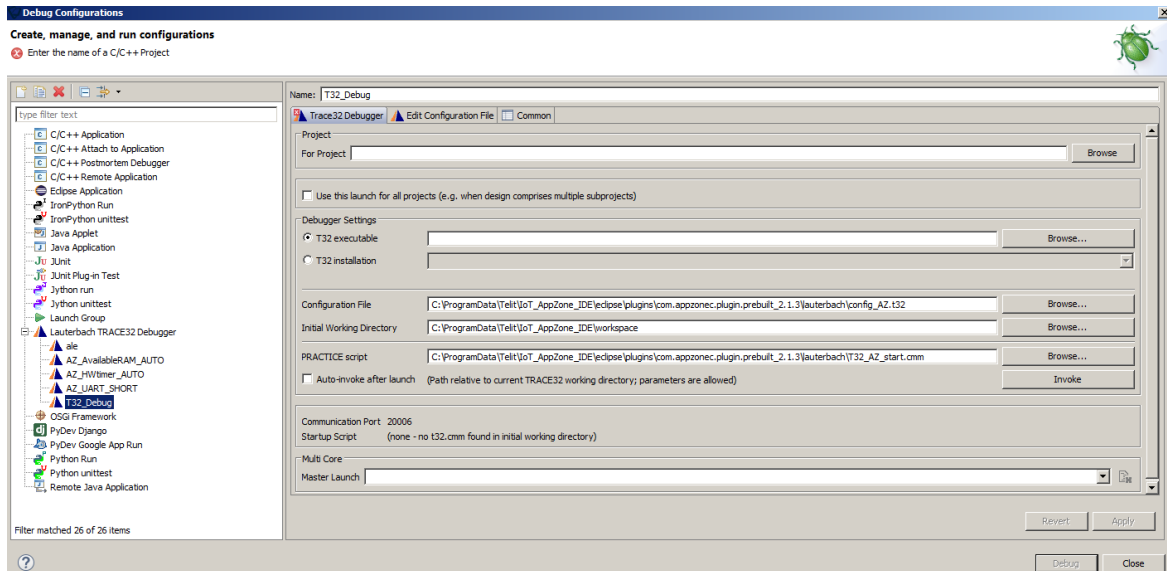


## 12.3. Debug setup

**To set up debug in Eclipse:**

1. From the menu, select **Window>Perspective>Open Perspective>Other**.
2. Select **C/C++** perspective and then click **OK**.
3. From the menu, select **Window>Perspective>Open Perspective>Other**.
4. Select the **Debug** perspective, and then click **OK**.
5. Move to the Debug perspective (you can select it on the upper right corner of the window)
6. From the menu, select **Run > Debug Configuration (🔧) > T32\_Debug**.

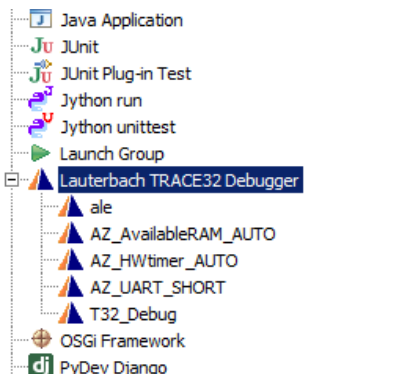
7. A template to configure the session opens (T32\_Debug).



8. Copy the following fields to a text editor:

- **Configuration File**
- **Initial Working Directory** – Directory used as the default in SDK IOT Installation. You can change this directory based on workspace that you intend to use.
- **PRACTICE script**

9. Right-click **Lauterbach Trace32 Debugger** and then select **New**.



10. In the Project pane, select your current project.

11. Verify that the **Initial Working Directory** field has been automatically filled.

12. In the **Name** field, copy and paste the value of For Project field.

13. In **Debugger Settings** pane, select the T32 executable and set the path to the t32marm.exe file.

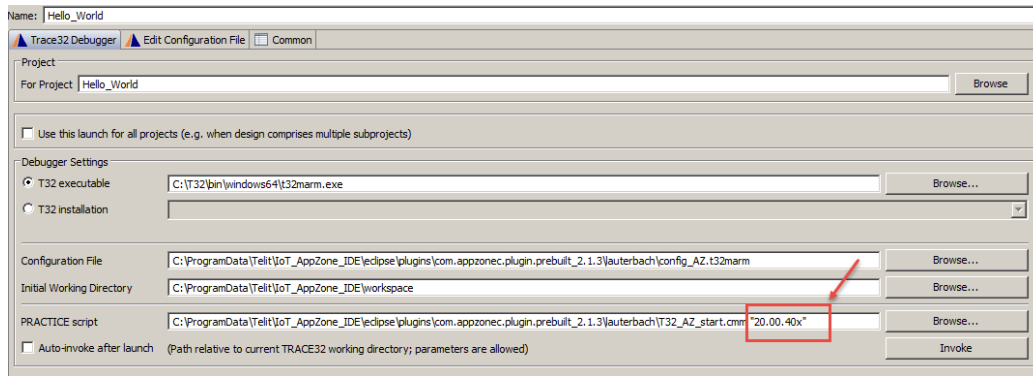
The default path is usually C:\T32\bin\windows64\t32marm.exe for typical T32 installation, or C:\T32\bin\t32marm.exe.

14. Copy the **Configuration File** and **PRACTICE script** fields from the text file that you saved to the corresponding fields.

15. In the **PRACTICE script** field, add the DUT (device under test) product number at the end enclosed in double quotes. The DUT number is the product number of the Telit module.

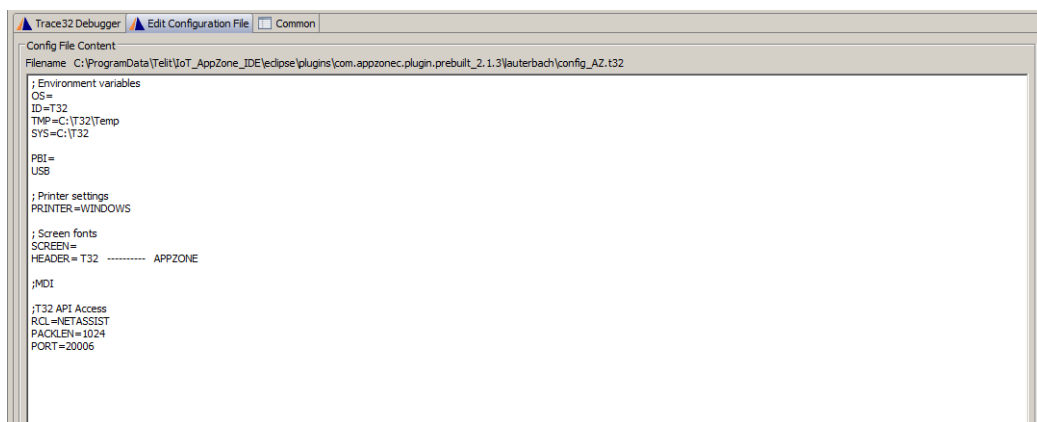
If you do not know the DUT product number, type AT+CGMR on your Telit module to display the number.

For Example for LE910-EU V2, type “20.00.40x”. Last number is not considered (you can set the exact number that the at+cgmr command returns or “x”).



16. In the **Edit Configuration File** tab, verify the T32 installation path.

We recommend that you do not change the default T32 installation path. If your T32 installation path is different from C:\T32, set SYS as your T32 installation path.



If you encounter communication problems between T32 and Eclipse, you can change the Port value.



**NOTE:**

Do not delete empty lines or add white spaces around the equal operators (=) in the configuration debug file or the T32 might not work properly.

17. (Optional) Add an icon to easily lunch your debug configuration:

- a. From the debug menu, select **Organize Favorites** and then click **Add**.
- b. Select the configuration that you want to add to the menu, and then click **OK**.

## 12.4. Debug an application

**To debug an application:**

1. Compile and build your new project.  
Make sure that the binary file generated has the name m2mapz.bin set by default because Lauterbach only loads m2mapz.bin files.
2. Connect the Debugger with a JTAG header attached to your computer using USB.
3. Power on the debugger.
4. Connect JTAG debugger cable to the Telit module that you want to debug.
5. Turn on your Telit module.
6. Launch your setup debug on Eclipse and select Debug Perspective.

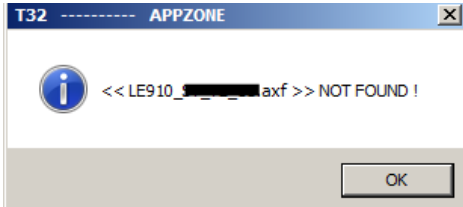
When Eclipse communicates with the Telit module, the T32 software is automatically called and executed. The communication is performed through debugger and USB cable via Semi-hosting.

You can send an AT command (at+m2m=4,10) to the device when you turned it on to inform it to execute the AppZone application if the application is not configured to start automatically.

If the Telit module is running and already configured to start the AppZone user application, the SDK automatically resets the module and starts the application.

If the Telit module is not configured to auto-start the AppZone user application, the user application does not start until an AppZone activation command is sent to the module (such as: at+m2m=4, 10. ...)

If a T32 error message such as the following appears:



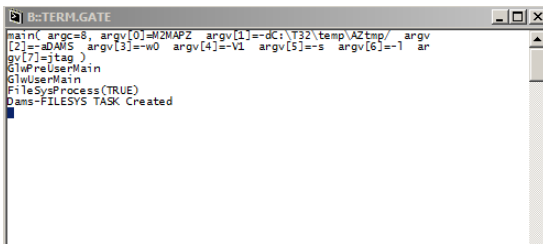
Contact Telit support for an updated symbol file to debug the module. The symbol file is located in the following folder:

..\\IoT\_AppZone\_IDE\\eclipse\\plugins\\com.appzonec.plugin.prebuilt\_<version>\\lauterbach\\APPZONE\\axf )

Where <version> is the AppZone version.

To set breakpoints when T32 is running, wait at least until you see some messages in the T32 TERM:GATE window to ensure that communication with Eclipse is not lost.

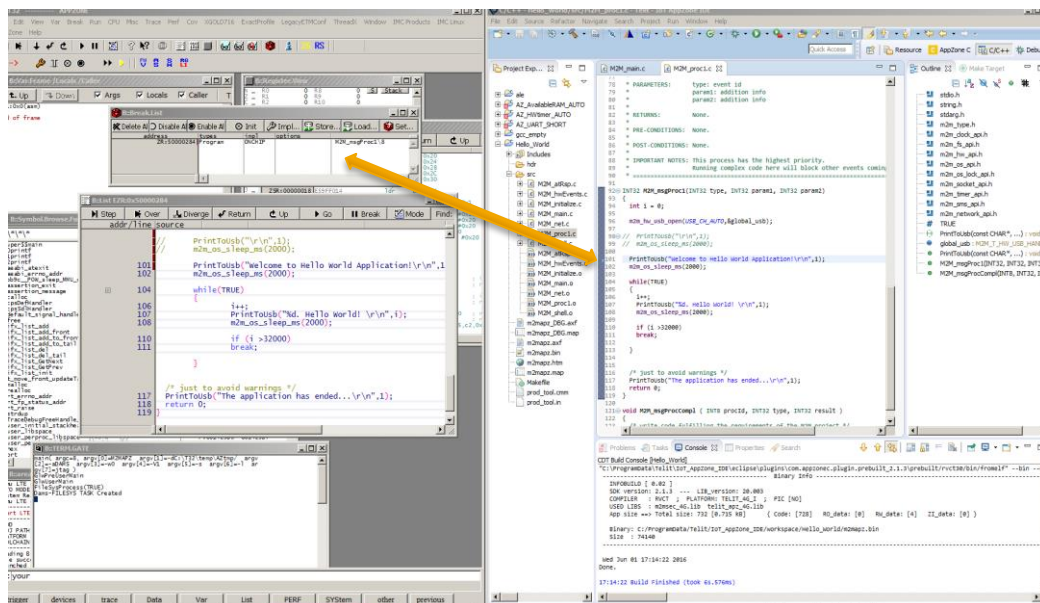
The following figure shows an example of a TERM:GATE window:



When you launch additional debug sessions, the breakpoints are not set automatically. You must set the breakpoints again for each session.

You can set breakpoints in Eclipse or in T32 because they are automatically synchronized.

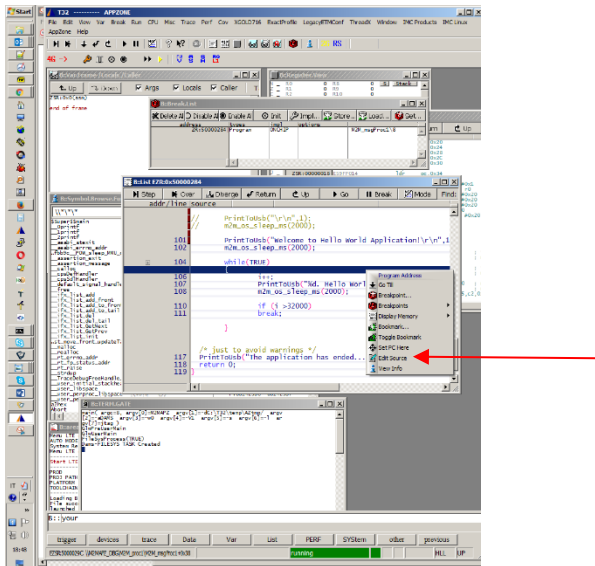
The following figure displays synchronized breakpoints:



You can display files in both Eclipse and T32.

- In Eclipse, right-click the source file and then select **Open In Trace32**.

- In Trace32, right-click in the List window and then select **Edit Source**.

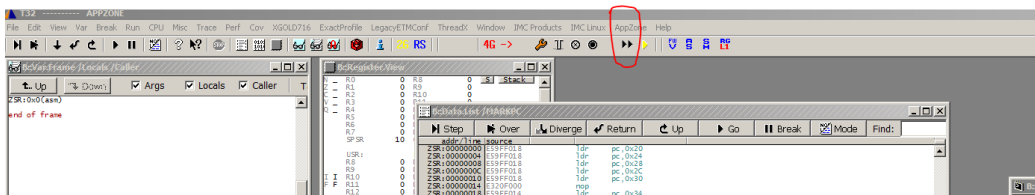


To terminate a debug session, In Eclipse Debug Perspective click the stop button (  ).

We recommend that you do not close Trace32 to refrain from resetting it, which is time consuming.

If a message appear to reset the device and connect, click **Yes**.

To relaunch the current section using Trace32 only, click the following icon:





# 13. MODULES & SW VER. TABLES

## SOFTWARE VER. TABLE

The table below summarizes the Services provided by the modules, and shows their coexistence. The available Service depends on the software version installed on the modules. Embedded/External GPS are beyond the scope of this guide.

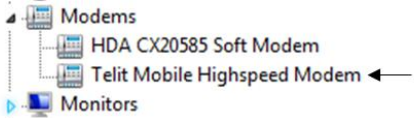
### SERVICES COEXISTENCE TABLE

	Services			
	Embedded GPS	External GPS	AppZone Python	AppZone C
	AZ Python and AZ C are mutually exclusive			
<b>HE910 Series</b>				
HE910	✓		✓	✓ *
<b>UE910 Series</b>				
UE910-EUR / UE910-EUD		✓	✓	✓ *
UE910-NAR / UE910-NAD		✓	✓	✓ *
<b>UL865 Series</b>				
UL865-EUR / UL865-EUD		✓	✓	✓ *
UL865-NAR / UL865-NAD		✓	✓	✓ *
<b>UE866 Series</b>				
UE866-N3G		✓	✓	✓ *
<b>GE910 Series</b>				
GE910-QUAD		✓	✓	✓ **
GE910-GNSS	✓		✓	✓ **
<b>LE910 Series</b>				
LE910v2		✓		✓

(\*): AppZone C available on specific part numbers. Available by default starting from release 12.00.xx8.

# 14. TROUBLESHOOTING

The following table lists known issues and how to solve them.

Issue	Cause	Solution
<p>When you access the environment, the following error message appears:</p> <p>Cannot open the application. See the log file null.</p>	<p>User does not have administrator permissions.</p> <p>!MESSAGE Error reading configuration: C:\ProgramData\Telit\AppZoneIDE\eclipse\configuration\org.eclipse.osgi\manager\.fileTableLock (Access is denied)</p>	<p>Change the permissions of the following file: .fileTableLock</p>
<p>When you try to load the application to the module, the AZ C Console displays the following error:</p> <p>Error in AT&amp;K3 command</p>	<p>The module is not connected.</p>	<p>Check the module is connected and is on. To verify that the module is connected in a Windows environment:</p> <ol style="list-style-type: none"> <li>1. From the Control Panel, open Device Manager.</li> <li>2. Expand Modems.</li> <li>3. Verify that a Telit modem is displayed.</li> </ol> 

# APPENDIX A. AT SYNTAX

This chapter describes the syntax of all AT Commands dedicated to manage the M2M applications.

You can operate the module using the AT commands. For example, enable and disable, and check size. All the commands that are used by the AZ C Console are implemented using AT commands.

## 14.1. AT+M2M

<b>+M2M – Enable/disable M2M Application execution</b>		<b>SELINT 2</b>
<b>AT+M2M=&lt;start_mode&gt;[,&lt;start_to&gt;,&lt;start_shell&gt;]</b>	<p>Set command sets the M2M Application execution start mode</p> <p>Parameters:</p> <p><b>&lt;start_mode&gt;</b></p> <ul style="list-style-type: none"> <li>0 – disable the M2M Application execution at the next startup (default).</li> <li>1 – enable the M2M Application execution at the next startup.</li> <li>2 – enable the default M2M Application execution at the next startup.</li> <li>3 – enable the M2M Application execution only if, at the next startup, the DTR line is found Low (that is: COM is not open on a PC).</li> <li>4 – enable the M2M Application execution only if, at the next startup, the user does not send any AT command on the serial port for time interval specified in. <b>&lt;start_to&gt;</b> parameter. The DTR line is not tested.</li> </ul> <p><b>&lt;start_to&gt;</b> – M2M Application execution start time-out; 10..60 - time interval in seconds; this parameter is used only if parameter <b>&lt;start_mode&gt;</b> is set to 4; it is the waiting time for an AT command on the serial port to disable active script execution start. If the user does not send any AT command on the serial port for the time specified in this parameter the M2M Application will not be executed (default is 10).</p> <p><b>&lt;start_shell&gt;</b> – Reserved for future use.</p> <p>Note: after issuing the AT command, the module will automatically restart. Note: an optional way to disable the M2M application execution (when <b>&lt;start_mode&gt;</b>=1 or 2) consists to flash an appropriate the stream.</p>	
<b>AT+M2M?</b>	<p>Read command reports the M2M Application execution start mode, start time-out, and start shell in the format:</p> <p><b>+M2M: &lt;start_mode&gt;,&lt; start_to&gt;,&lt;start_shell&gt;</b></p>	
<b>AT+M2M=?</b>	<p>Test command returns the range of available values for parameters <b>&lt; start_mode&gt;</b>, <b>&lt; start_to&gt;</b>, and <b>&lt;start_shell&gt;</b>.</p>	

## 14.2. AT#M2MWRITE

#M2MWRITE – M2M File System File Write	SELINT 2
<b>AT#M2MWRITE=&lt;file_name&gt;,&lt;size&gt;[,&lt;permission&gt;]</b>	<p>Execution command causes the module to store a generic file, for example M2M Application binary file, in the M2M file system in the current working directory, naming it <b>&lt;file_name&gt;</b></p> <p>The file should be sent using RAW ASCII file transfer. It is important to set properly the port settings. In particular: Flow control: hardware.</p> <p>Parameters:</p> <p><b>&lt;file_name&gt;</b> – file name in M2M file system, quoted string type (up to max 16 chars depending on current working directory, case sensitive).</p> <p><b>&lt;size&gt;</b> – file size in bytes</p> <p><b>&lt;permission&gt;</b> – file permission (optional); sum of integers each representing a file permission; default value if not present is 0;</p> <ul style="list-style-type: none"> <li>0 - nothing</li> <li>1 - executable binary (.bin)</li> <li>16 - compressed (.gz)</li> </ul> <p>If the parameter <b>&lt;permission&gt;</b> is present, it is not 0, its value matches file name extension (e.g.: 1 - .bin; 16 - .gz, 17 - .bin.gz) the file will be stored in the M2M file system in drive 0 in the directory \MOD setting the requested file permission. If the parameter <b>&lt;permission&gt;</b> is not present the file will be stored in the M2M file system in the current working directory without setting any file permission.</p> <p>The device shall prompt a five character sequence <b>&lt;CR&gt;&lt;LF&gt;&lt;greater_than&gt;&lt;greater_than&gt;&lt;greater_than&gt;</b> (IRA 13, 10, 62, 62, 62) after command line is terminated with <b>&lt;CR&gt;</b>; after that a file can be entered from TE, sized <b>&lt;size&gt;</b> bytes.</p> <p>The operations complete when all bytes are received.</p> <p>If writing ends successfully, the response is <b>OK</b>; otherwise, an error code is reported.</p> <p>Note: the file name should be passed between quotes; file names are case sensitive.</p>
<b>AT#M2MWRITE=?</b>	Test command returns <b>OK</b> result code.
Example	<pre>AT#M2MWRITE="M2MAPZ.bin",58044 &gt;&gt;&gt; here receive the prompt; then type or send the file, sized 58044 bytes OK File has been stored</pre>

## 14.3. AT#M2MLIST

#M2MLIST – M2M File System List		SELINT 2
<b>AT#M2MLIST</b>	<p>Execution command reports the list of directories names and file names of directories and files currently stored in the M2M file system in the current working directory. In the end reports the available free memory in the current drive. The report is in the format:</p> <pre>[&lt;CR&gt;&lt;LF&gt;#M2MLIST: &lt;.&gt; &lt;CR&gt;&lt;LF&gt;#M2MLIST: &lt;..&gt;] [&lt;CR&gt;&lt;LF&gt;#M2MLIST: &lt;dir_name1&gt;... [&lt;CR&gt;&lt;LF&gt;#M2MLIST: &lt;dir_namen&gt;]] [&lt;CR&gt;&lt;LF&gt;#M2MLIST: &lt;file_name1&gt;,&lt;size1&gt;... [&lt;CR&gt;&lt;LF&gt;#M2MLIST: &lt;file_namen&gt;,&lt;size1&gt;]] &lt;CR&gt;&lt;LF&gt;#M2MLIST: free bytes: &lt;free_mem&gt;</pre> <p>where:</p> <ul style="list-style-type: none"> <li>&lt;.&gt; current directory</li> <li>&lt;..&gt; upper directory</li> <li>&lt;dir_namen&gt; – directory name, string type delimited by &lt; and &gt; (max 16 chars, case sensitive)</li> <li>&lt;file_namen&gt; – file name, quoted string type (max 16 chars, case sensitive)</li> <li>&lt;size1&gt; – size of file in bytes</li> <li>&lt;free_mem&gt; – size of available free memory in the current drive in bytes</li> </ul>	
<b>AT#M2MLIST=?</b>	Test command returns <b>OK</b> result code.	
Example	<pre>AT#M2MLIST #M2MLIST: &lt;.&gt; #M2MLIST: &lt;..&gt; #M2MLIST: &lt;dir1&gt; #M2MLIST: "M2MAPZ.bin",58044 #M2MLIST: free bytes: 458752</pre> <p>OK</p>	

## 14.4. AT#M2MDEL

#M2MDEL – M2M File System File Delete		SELINT 2
<b>AT#M2MDEL=[&lt;file_name&gt;]</b>	<p>Execution command deletes the file from the M2M file system in the current working directory.</p> <p>Parameter:</p> <ul style="list-style-type: none"> <li>&lt;file_name&gt; – file name to delete, quoted string type (max 16 chars, case sensitive)</li> </ul> <p>Note: the file name should be passed between quotes; file names are case sensitive. Note: If the file &lt;file_name&gt; is not present in the current working directory, an error code is reported.</p>	
<b>AT#M2MDEL=?</b>	Test command returns <b>OK</b> result code.	
Example	<pre>AT#M2MDEL="M2MAPZ.bin"</pre> <p>OK</p>	

## 14.5. AT#M2MDELALL

#M2MDELALL – M2M File System Delete All Files		SELINT 2
<b>AT#M2MDELALL</b>	<p>Execution command deletes all files from the current working directory in the M2M file system.</p> <p>Note: execution command does not delete directories.</p>	
<b>AT#M2MDELALL=?</b>	Test command returns <b>OK</b> result code.	
Example	<pre>AT#M2MDELALL</pre> <p>OK</p>	

## 14.6. AT#M2MREAD

#M2MREAD – M2M File System File Read		SELINT 2
AT#M2MREAD=[<file_name>]	<p>Execution command reports the content of the file &lt;file_name&gt; stored in the M2M file system in the current working directory.</p> <p>Parameter: &lt;file_name&gt; – file name, quoted string type (max 16 chars, case sensitive)</p> <p>The device shall prompt a five character sequence &lt;CR&gt;&lt;LF&gt;&lt;less_than&gt;&lt;less_than&gt;&lt;less_than&gt; (IRA 13, 10, 60, 60, 60) followed by the file content.</p> <p>Note: the file name should be passed between quotes; file names are case sensitive. Note: If the file &lt;file_name&gt; is not present in the current working directory, an error code is reported.</p>	
AT#M2MREAD=?	Test command returns <b>OK</b> result code.	
Example	<p>AT#M2MREAD="config.txt "</p> <p>&lt;&lt;&lt; here receive the prompt; then the file is displayed, immediately after the prompt</p> <p>OK</p>	

## 14.7. AT#M2MRUN

#M2MRUN – M2M Set Run File Permission		SELINT 2
AT#M2MRUN=<mode>[,<file_name>]	<p>Set command sets and resets the RUN file permission for the executable binary files and compressed files stored in the drive 0 in the directory \MOD in the M2M file system.</p> <p>Parameters:</p> <p>&lt;mode&gt; - integer type, set/reset mode value:</p> <ul style="list-style-type: none"> <li>0 – resets RUN file permission for all the files stored in the drive 0, directory \MOD.</li> <li>1 – sets RUN file permission for the file &lt;file_name&gt; stored in the drive 0, directory \MOD. Reserved for future use.</li> <li>2 – sets RUN file permission for the file &lt;file_name&gt; stored in the drive 0, directory \MOD, and resets RUN file permission for all the other files stored in the drive 0, directory \MOD.</li> </ul> <p>&lt;file_name&gt; - file name to set RUN file permission, quoted string type (max 16 chars, case sensitive). File name extension must be either .bin or .gz or .bin.gz. &lt;file_name&gt; parameter must not be present if &lt;mode&gt; is 0.</p> <p>Note: the file name should be passed between quotes; file names are case sensitive.</p>	
AT#M2MRUN?	<p>Read command reports the files with the RUN file permission between those in the drive 1 in the directory \MOD in the M2M file system in the format:</p> <p>[&lt;CR&gt;&lt;LF&gt;#M2MRUN: &lt;file_name1&gt;... [&lt;CR&gt;&lt;LF&gt;#M2MRUN: &lt;file_namen&gt;]]</p> <p>where: &lt;file_namen&gt; – file name, quoted string type (max 16 chars, case sensitive)</p>	
AT#M2MRUN=?	Test command returns the allowed values for parameter <mode>.	
Example	<p>AT#M2MRUN =2,"M2MAPZ.bin"</p> <p>OK</p> <p>AT#M2MRUN?</p> <p>#M2MRUN: "M2MAPZ.bin"</p> <p>OK</p>	

## 14.8. AT#M2MCHDRIVE

#M2MCHDRIVE – M2M File System Change Current Drive		SELINT 2
<b>AT#M2MCHDRIVE=&lt;drive&gt;</b>	<p>Set command sets the current drive in the M2M file system.</p> <p>Parameter: <b>&lt;drive&gt;</b> – integer type, current drive integer value.</p> <p>Note: the only available drive value in the M2M file system is 0.</p> <p><i>If standard file system is the same as M2M file system, the following applies.</i> Note: if the current drive value in the M2M file system is not 0 then AT commands related to SCRIPT family and MMS family that make use of the M2M file system will have ERROR response.</p>	
<b>AT#M2MCHDRIVE?</b>	<p>Read command reports the current drive in the M2M file system in the format:</p> <p><b>#M2MCHDRIVE: &lt;drive&gt;</b></p>	
<b>AT#M2MCHDRIVE=?</b>	Test command returns the allowed values for parameter <b>&lt;drive&gt;</b> .	
Example	<p>AT#M2MCHDRIVE? #M2MCHDRIVE: 0 OK</p>	

## 14.9. AT#M2MCHDIR

#M2MCHDIR – M2M File System Change Current Directory		SELINT 2
<b>AT#M2MCHDIR=&lt;path_name&gt;</b>	<p>Set command sets the current working directory in the current drive in the M2M file system.</p> <p>Parameter: <b>&lt;path_name&gt;</b> – directory name, quoted string type (up to max 16 chars depending on current working directory, case sensitive) or relative path name, quoted string type (up to max 124 chars depending on current working directory, case sensitive) or absolute path name, quoted string type (max 124 chars, case sensitive)</p> <p>Note: the directory name, relative path name or absolute path name should be passed between quotes; directory and path names are case sensitive.</p> <p>Note: path separator can be either \ or /. Directory name begins with a character different from path separator and is relative to the current working directory. Relative path name begins with a character different from path separator and is relative to the current working directory. Absolute path name begins with path separator. System max path name length (current directory name length + file name length) is 128. System reserves 2 characters for internal use.</p> <p>Note: if the directory name, relative path name or absolute path name <b>&lt;path_name&gt;</b> is not present an error code is reported.</p> <p>Note: the current directory in the drive 0 in the M2M file system at every power on is \.</p>	
<b>AT#M2MCHDIR?</b>	<p>Read command reports the current working directory in the current drive in the M2M file system in the format:</p> <p><b>#M2MCHDIR: &lt;path_name&gt;</b></p> <p>Where: <b>&lt;path_name&gt;</b> – absolute path name, quoted string type (max 124 chars, case sensitive)</p> <p>Note: path separator used in this report is \.</p>	

#M2MCHDIR – M2M File System Change Current Directory		SELINT 2
<b>AT#M2MCHDIR=?</b>	Test command returns <b>OK</b> result code.	
Example	AT#M2MCHDIR? #M2MCHDIR: "\MOD" OK  AT#M2MCHDIR="dir1" OK  AT#M2MCHDIR? #M2MCHDIR: "\MOD\dir1" OK	

## 14.10. AT#M2MMKDIR

#M2MMKDIR – M2M File System Make Directory		SELINT 2
<b>AT#M2MMKDIR=&lt;dir_name&gt;</b>	Set command makes a new directory in the current working directory in the M2M file system.	
	Parameter: <b>&lt;dir_name&gt;</b> – directory name, quoted string type (up to max 16 chars depending on current working directory, case sensitive)  Note: the directory name should be passed between quotes; directory names are case sensitive.	
<b>AT#M2MMKDIR=?</b>	Test command returns <b>OK</b> result code.	
Example	AT#M2MMKDIR="dir1" OK	

## 14.11. AT#M2MRMDIR

#M2MRMDIR – M2M File System Remove Directory		SELINT 2
<b>AT#M2MRMDIR=&lt;dir_name&gt;</b>	Set command removes the directory from the current working directory in the M2M file system.	
	Parameter: <b>&lt;dir_name&gt;</b> – directory name, quoted string type (max 16 chars, case sensitive)  Note: the directory name should be passed between quotes; directory names are case sensitive.  Note: if the directory <b>&lt;dir_name&gt;</b> is not present in the current working directory, an error code is reported.  Note: if the directory <b>&lt;dir_name&gt;</b> is not empty, it is not possible to remove it and an error code is reported.	
<b>AT#M2MRMDIR=?</b>	Test command returns <b>OK</b> result code.	
Example	AT#M2MRMDIR="dir1" OK	



## 14.12. AT#M2MBA

#M2MBA – Get M2M Application base addresses		SELINT 2
<b>AT#M2MBA</b>	<p>Execution command gets the M2M Application base addresses. Base addresses are useful to debug M2M Application.</p> <p>If no M2M Application have been already loaded the returned string is: 0x00000000 0x00000000</p> <p>Otherwise, depending on the toolchain used to compile the M2M Application: RVCT: --ro_base 0XXXXXXXXX --rw_base 0Yyyyyyyyy GCC: __ROM__ = 0XXXXXXXXX; __RAM__ = 0xYYYYYYYY;</p> <p>Generally base addresses are stored on RAM, but the storage can be forced on NVM using +M2M command.</p>	
<b>AT#M2MBA=?</b>	Test command returns the <b>OK</b> result code	

## 14.13. M2M AT Command Examples

Assume that the developer has created a user M2M application contained in the m2mapz.bin file (default name). In addition, suppose that the developer has renamed the m2mapz.bin file into user\_m2mapz.bin and uploaded it into the file system of the module.

The modules with AppZone layer provide several M2M AT commands to manage the user M2M applications and to configure the start mode of the selected application on next module reboot.

The AT commands examples described in the next chapters refer to Fig. 5. The DTE is based on a Windows-PC running a Terminal tool that sends AT commands and files to the module, and receives AT commands results from it. To have more information on AT commands syntax refer to chapter **Error! Reference source not found..**

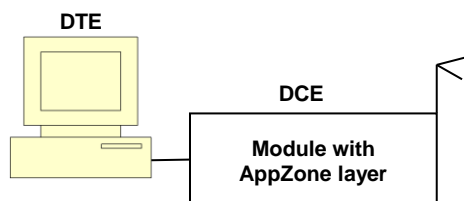


Fig. 5: DTE Connected to the Module



The [AZC](#) Console is integrated with AppZone C and enables entering AT Commands and upload/download files into/from module. It simplifies and speeds up the activities performed with the DTE..

### 14.13.1. AT+M2M=0

AT+M2M=0 disables the execution of both user and default M2M applications.

This example introduces the default M2M application provided by the AppZone layer.

Assume that the module has the factory-setting configuration +M2M=0,10,0, and no user M2M application installed on it. Power on the module and enter the following command to check the current +M2M value defining the application start mode on next reboot:

AT+M2M?

+M2M: 0,10,0 → the execution of any application is disabled on next reboot (default)

OK

Enter the command below to reboot the module.

AT+M2M=1

OK

After rebooting, the module starts its default M2M application because no user M2M application is available. The default application displays on DTE the following message:

Telit Communications S.p.A - AppZone M2M Default Application...

OK

The AT parser can still receive AT commands from the operator on the serial line. Enter the following command to reboot again the module.

AT+M2M=0

OK

After rebooting, the module does not execute the default M2M application; the operator may continue to enter AT commands into the module:

AT+M2M?

+M2M: 0,10,0 → the execution of any application is disabled on next reboot (default)

OK

### 14.13.2. AT+M2M=1

AT+M2M=1 starts the user M2M application having the RUN permission set through AT#M2MRUN command. If no user M2M application has set the RUN permission, AT+M2M=1 command starts the default M2M application.

For example, assume that the module has the factory-setting configuration +M2M=0,10,0, and no user M2M application installed on it. Power on the module and check the current drive:

AT#M2MCHDRIVE?

#M2MCHDRIVE: 0 → the module provides only drive 0

OK

Check the current directory:

AT#M2MCHDIR?

#M2MCHDIR: "/" → factory setting

OK

Enter the following command to verify the free space of file systems:

AT#M2MLIST

#M2MLIST: <MOD>

#M2MLIST: "m2m\_config",134

#M2MLIST: free bytes: 6160384 → free bytes

OK



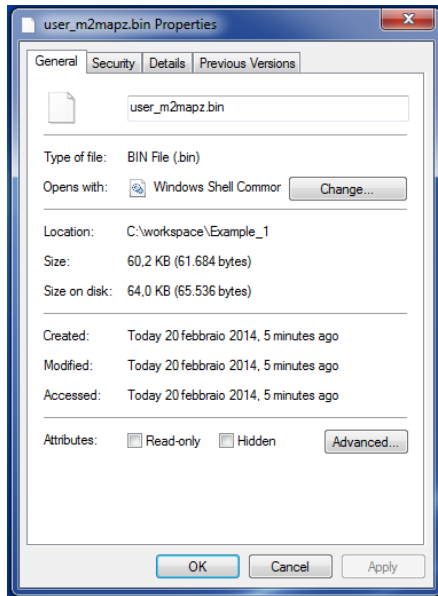
The size of the file system depends on the module type under test.

Now, assume that the user has created a M2M application as show in the previous chapters, and renamed the m2mapz.bin file in user\_m2mapz.bin.

Here are the steps to upload the user M2M application into the module.

Before uploading the file into the file system of the module, you need to know the size of the file expressed in bytes. Use the Properties dialog box to get this information. The figure on the left side shows 61684 bytes, use this value with the AT#M2MWRITE command. Telit provides a tool to upload

the and run user M2M application. See **Error! Reference source not found.**[Error! Reference source not found.](#) for more information.



Enter the command below. After the command has responded with the “>>>” prompt, you are allowed to send the M2M application using the Send File option provided by the Terminal tool (use RAW ASCII protocol). On success, the module returns the OK message.

```
AT#M2MWRITE="user_m2mapz.bin",61684,1
```

```
>>>
```

```
OK
```



If a file is already present in the module with the same name of the file just uploaded, the new one without notification overwrites the old one. The new uploaded file assumes the run permission of the old one.

If the third parameter is equal to 1, the AT#M2MWRITE command uploads automatically the .bin file into drive 0 and directory /MOD. To verify it enter the following commands.

Change the directory:

```
AT#M2MCHDIR="MOD"
```

```
OK
```

Check the current directory:

```
AT#M2MCHDIR?
```

```
#M2MCHDIR: "/MOD"
```

```
OK
```

Check if the uploaded .bin file has been stored in the /MOD directory.

```
AT#M2MLIST
```

```
#M2MLIST: <.>
```

```
#M2MLIST: <..>
```

```
#M2MLIST: "user_m2mapz.bin",61684
```

#M2MLIST: free bytes: 6097920

OK

Enter the next command to check which M2M application has RUN permission. The command works on executable binary files and compressed files stored in the /MOD directory (drive 0).

AT#M2MRUN?

OK → no user applications have RUN permission

Check the current +M2M value defining the application start mode on next reboot.

AT+M2M?

+M2M: 0,10,0 → M2M application execution on next reboot is disabled (default)

OK

Enter the next command to reboot the module.

AT+M2M=1

OK

After rebooting, the module starts the default M2M application because the just uploaded user application has not RUN permission. The default M2M application displays on DTE the following message:

Telit Communications S.p.A - AppZone M2M Default Application...

OK

The AT parser can still receive AT commands from the operator on the serial line. Check the current +M2M value stored in NVM after the previous rebooting.

AT+M2M?

+M2M: 1,10,0 → M2M application execution on next reboot is disabled (default)

OK

Use the following command to set the RUN permission for the user\_m2mapz.bin application, and reset RUN permission for all other applications included the default M2M application. The command below works on the executable binary files and compressed files stored in the /MOD directory of drive 0.

AT#M2MRUN=2,"user\_m2mapz.bin"

OK

Enter again the AT#M2MRUN? command to check which M2M application has RUN permission.

AT#M2MRUN?

#M2MRUN: "user\_m2mapz.bin" →user application has RUN permission

OK

Enter the next command to reboot the module.

AT+M2M=1

OK

After rebooting, the module starts the user M2M application enabled via the previous AT#M2MRUN command. For example, the user M2M application displays on DTE the following message:

Hello World

The AppZone layer has executed the user application. The AT parser cannot receive AT commands from the operator because the user application has permanently captured the serial line. If the module is turned off/on, the user application runs again; to exit from this scenario the module must be re-flashed.

### 14.13.3. AT+M2M=2

AT+M2M=2 starts only the default M2M application, regardless if a user M2M application has the RUN permission set through AT#M2MRUN command.

For example, assume that the module has factory-setting configuration +M2M=0,10,0, and no user M2M application installed on it. Enter the following command to upload the user\_m2mapz.bin application. Use the procedure shown in previous examples.

```
AT#M2MWRITE="user_m2mapz.bin",61684,1
```

```
>>>
```

```
OK
```

Enter /MOD directory, and verify if the just uploaded .bin file has been stored in it.

```
AT#M2MLIST
```

```
#M2MLIST: <.>
```

```
#M2MLIST: <..>
```

```
#M2MLIST: "user_m2mapz.bin",61684
```

```
#M2MLIST: free bytes: 6097920
```

```
OK
```

Enter the next command to check which M2M application has RUN permission. The command works on executable binary files and compressed files stored in the /MOD directory (drive 0).

```
AT#M2MRUN?
```

```
OK
```

→ no user applications have RUN permission

Use the following command to set the RUN permission for user\_m2mapz.bin application and reset RUN permission for all other applications included the default M2M application.

```
AT#M2MRUN=2,"user_m2mapz.bin"
```

```
OK
```

Enter again the AT#M2MRUN? command to check which M2M application has RUN permission.

```
AT#M2MRUN?
```

```
#M2MRUN: "user_m2mapz.bin" → user application has RUN permission
```

```
OK
```

Check the current +M2M value defining the application start mode on next reboot.

```
AT+M2M?
```

```
+M2M: 0,10,0  
(default)
```

→the execution of any application is disabled on next reboot

```
OK
```

Enter the next command to reboot the module.

AT+M2M=2

OK

After rebooting, the default M2M application is executed even if the user M2M application has the RUN permission. The default M2M application displays on DTE the following message:

Telit Communications S.p.A - AppZone M2M Default Application...

OK

The AT parser can still receive AT commands from the operator on the serial line. Check the current +M2M value stored in NVM after the previous rebooting.

AT+M2M?

+M2M: 2,10,0

OK

Enter again the AT#M2MRUN? command to check which M2M application has RUN permission.

AT#M2MRUN?

#M2MRUN: "user\_m2mapz.bin" → user application has RUN permission

OK

Now, turn off/on the module. The module reboots and starts the default application even if the user M2M application has the RUN permission. The default M2M application displays on DTE the following message:

Telit Communications S.p.A - AppZone M2M Default Application...

OK

The AT parser can still receive AT commands from the operator on the serial line.

#### 14.13.4. AT+M2M=3

AT+M2M=3 starts the user M2M application execution in accordance with the level of DTR control line.

To perform the example described in the next chapter on DTE - refer to Fig. 5 - is installed the Telit AT Controller application because it provides the feature to change manually the DTR line. DTR is an output control line of the DTE, when it is low (active) informs the module that the DTE is ready to establish a communication.

For example, assume that the module has the factory-setting configuration +M2M=0,10,0, and no user M2M application installed on it. Power on the module and upload the user M2M application into /MOD directory (driver 0).

AT#M2MWRITE="user\_m2mapz.bin",61684,1

>>>

OK

Check the current +M2M value defining the application start mode on next reboot. AT+M2M?

+M2M: 0,10,0 → the execution of any application is disabled on next reboot (default)

OK

Force the DTR control line to low using the proper button provided by the Telit AT Controller application (click twice on the green DTR check marker, it becomes red). Enter the command below to

reboot the module. DTR is low, the module starts the default M2M application because the just uploaded user application does not have the RUN permission, refer to **Error! Reference source not found..**

AT+M2M=3

OK

After rebooting, the default M2M application is executed and displays on DTE the following message:  
Telit Communications S.p.A - AppZone M2M Default Application...

OK

The AT parser can still receive AT commands from the operator on the serial line. Check the current +M2M value stored in NVM after the previous rebooting.

AT+M2M?

+M2M: 3,10,0

OK

Force the DTR control line to high (click twice on the read DTR check marker, it becomes green), and power off/on the module. After rebooting, no M2M application is started (DTR is high), the operator may continue to enter AT commands into the module, refer to **Error! Reference source not found..**

Check the current +M2M value stored in NVM after the previous rebooting.

AT+M2M?

+M2M: 3,10,0

OK

Enter the next command to check which user M2M application has RUN permission.

AT#M2MRUN?

OK

→ no user M2M application has RUN permission

Use the following command to set the RUN permission for user\_m2mapz.bin application and reset RUN permission for all other applications included the default M2M application.

AT#M2MRUN=2,"user\_m2mapz.bin"

OK

Check which M2M application has RUN permission.

AT#M2MRUN?

#M2MRUN: "user\_m2mapz.bin"

OK

Force the DTR control line to low (click twice on the green DTR check marker, it becomes red), and power off/on the module, refer to **Error! Reference source not found..** After rebooting, the user M2M application is started and displays on DTE, for example, the following message:

Hello World

The operator may not enter more AT commands into the module. The user M2M application has permanently captured the serial line. Follow the steps below to exit this scenario:

1. Force the DTR control line to high (click twice on the read DTR check marker, it becomes green);
2. Turn off/on the module.

After rebooting, no M2M applications have been started, the AT parser accepts again AT commands, refer to **Error! Reference source not found..** Check the current +M2M value stored in NVM after the previous rebooting.

AT+M2M?

+M2M: 3,10,0

OK

The following table summarizes the four cases described.

DTR	Run Permission	Enter	Start
Low	No	AT+M2M=3	Default M2M Appl.
High	No	OFF/ON	No M2M Appl.
Low	Yes	OFF/ON	User M2M Appl.
High	Yes	OFF/ON	No M2M Appl.

#### 14.13.5. AT+M2M=4,30

AT+M2M=4,xx starts the user M2M application having the RUN permission set, if xx seconds are expired. If the user types an AT command before the expiration of the xx seconds, the control does not execute the M2M application. The RUN permission is set by means of the AT#M2MRUN command.

For example, assume that the module has the factory-setting configuration +M2M=0,10,0, and no user M2M application installed on it. Power on the module and upload the user M2M application into /MOD directory (drive 0). For details, refer to chapter **Error! Reference source not found..**

```
AT#M2MWRITE="user_m2mapz.bin",61684,1
```

```
>>>
```

OK

Change the directory:

```
AT#M2MCHDIR="/MOD"
```

OK

Check the current directory:

```
AT#M2MCHDIR?
```

```
#M2MCHDIR: "/MOD"
```

OK

Verify, after the uploading, the list of files stored in the current directory (/MOD).

```
AT#M2MLIST
```

```
#M2MLIST: <.>
```

```
#M2MLIST: <..>
```

```
#M2MLIST: "user_m2mapz.bin",61684 → the user M2M application
```

```
#M2MLIST: free bytes: 6097920
```

OK

Use the following command to set the RUN permission for user\_m2mapz.bin application and reset RUN permission for all other application included the default M2M application. The next command works on the executable binary files and compressed files stored in the /MOD directory.



AT#M2MRUN=2,"user\_m2mapz.bin"

OK

Enter the next command to check which M2M application has RUN permission.

AT#M2MRUN?

#M2MRUN: "user\_m2mapz.bin" → user M2M application has RUN permission

OK

Check the current +M2M value defining the application start mode on next reboot.

AT+M2M?

+M2M: 0,10,0 → the execution of any application is disabled on next reboot (default)

OK

Use the AT+M2M command with two parameters: the first one (4) sets the start mode, the second one (30, as example) defines a time interval expressed in sec.

AT+M2M=4,30

OK

After entering this command, the module reboots, and the following two choices are available:

1. If the operator enters an AT command (different from AT<CR>), before the expiration of the time interval, the control does not start the user M2M application, the user may enter AT commands. Enter the next command to verify the +M2M values:

AT+M2M?

+M2M: 4,30 → the +M2M values are memorized in NVM

OK

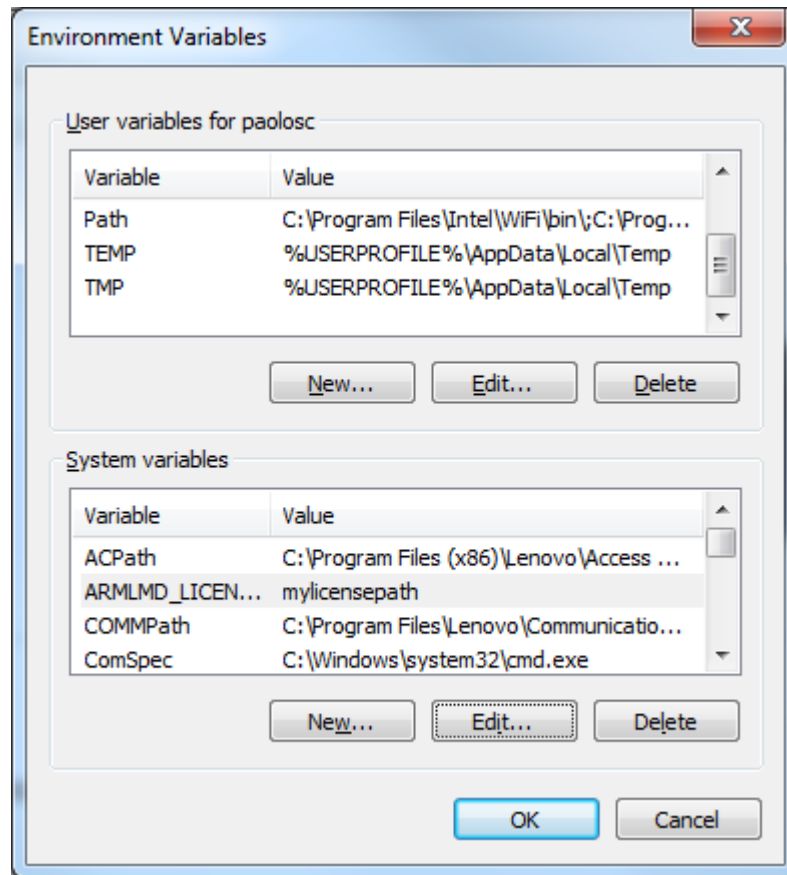
After entering again the AT+M2M=4,30 command the module reboots, and again the two choices are available.

2. If the time interval expires and the operator has not entered AT commands, the control starts the user M2M application. If no user application has RUN permission, the default application is executed.

If the module is powered OFF/ON, it uses the AT+M2M=4,30 configuration memorized in NVM and one of the scenarios above described may be used again.

# APPENDIX B. USING THE RVCT COMPILER

RVCT compiler is included in the AppZone C SDK package. To use your RVCT license set the System Variable named ARMLMD\_LICENSE\_FILE in Windows settings, as reported in the example below:



# DOCUMENT HISTORY

Revision	Date	Changes
0	2016-09-22	Updated entire document to support integrated IDE and renamed to 1VV0301335_AppZoneC-User Guide (previous document code was 80496ST10722A)
1	2017-10-10	Added Emulator

