XBee® Multi Programmer User Guide

User Guide

Revision history—90002263

Revision	Date	Description
A	March 2018	Initial Release
B	March 2019	Updated supported devices. Added **Install USB drivers for cellular modems**. Added through-hole break-in procedure. Made changes for version 1.1.0 of the application software.
C	June 2019	Added items included in the package. Clarified terminology.
D	October 2019	1.2.0 release.

Trademarks and copyright

Disclaimers

Information in this document is subject to change without notice and does not represent a commitment on the part of Digi International. Digi provides this document “as is,” without warranty of any kind, expressed or implied, including, but not limited to, the implied warranties of fitness or merchantability for a particular purpose. Digi may make improvements and/or changes in this manual or in the product(s) and/or the program(s) described in this manual at any time.

Warranty

To view product warranty information, go to the following website:

www.digi.com/howtobuy/terms

Customer support

Gather support information: Before contacting Digi technical support for help, gather the following information:

- Product name and model
- Product serial number (s)
- Firmware version
- Operating system/browser (if applicable)
- Logs (from time of reported issue)
- Trace (if possible)
- Description of issue
- Steps to reproduce

Contact Digi technical support: Digi offers multiple technical support plans and service packages. Contact us at +1 952.912.3444 or visit us at www.digi.com/support.

Feedback

To provide feedback on this document, email your comments to

techcomm@digi.com

Include the document title and part number (Digi XBee® Multi Programmer, 90002263 C) in the subject line of your email.

Contents

Connect an XBee Multi Programmer board

Load a profile

Program the XBee devices

Settings

Update software

How-to articles

Known issues

Digi XBee® Multi Programmer

The XBee Multi Programmer is a combination of hardware and software that enables users to program multiple Digi Radio frequency (RF) devices simultaneously. It provides a fast and easy way to prepare devices for distribution or large network deployment. Some of the features include:

- The XBee Multi Programmer allows you to program up to six devices simultaneously.
- Connect more XBee Multi Programmers to increase the number of devices you can program simultaneously.
- Multiple Multi programmers may be attached to the same computer.
- Three interchangeable header board variants support all the XBee form factors to program surface-mount (SMT), through-hole (TH) and micro-mount (MMT) Digi RF devices.
- Intuitive application interface makes it easy to start programming devices in just a few minutes.
- Unattended programming process allows you to focus on substituting devices in the boards without wasting any time.
- Export your programming session report or save it in a database to track your progress.
- Automatic application update keeps you up to date with the latest software version.
- Online documentation can be accessed directly from the application.

Hardware

The XBee Multi Programmer tool is an enclosed hardware component that allows you to program up to six RF devices at a time thanks to its six external XBee sockets. For more information about this tool, see Hardware overview.

There are three variants of the XBee Multi Programmer tool, one for each footprint of the XBee product line; see Replace XBee socket boards.

Software

The XBee Multi Programmer application communicates with the boards and allows you to easily set up and execute programming sessions. For more information about the application, see Application software overview.

Package contents

The XBee Multi Programmer package contains the following components:

- One XBee Multi Programmer
- One USB-C cable
- One power supply

Download and install the XBee Multi Programmer application software

This section provides instructions for downloading and installing the XBee Multi Programmer. If the XBee Multi Programmer is not automatically detected when it is attached to your computer, you may also need to install the USB drivers.

Requirements

To program Digi RF devices with the Multi Programmer application software, you must connect the tool to your computer. Programming requires two pieces of external hardware:

- USB-C cable
- Power supply of 9 VDC or 12 VDC with a current rating of at least 1.5 A

Operating systems

XBee Multi Programmer is compatible with the Windows Vista/7/8/10 (32-bit or 64-bit versions) operating systems.

System requirements

Property	Minimum	Recommended
HDD space	300 MB	500 MB
RAM memory	2 GB	4 GB
CPU	Dual-core processor	Quad-core processor

Supported RF devices

- XBee3 Zigbee
- XBee3 DigiMesh
- XBee3 802.15.4
- XBee3 Cellular LTE CAT 1
- XBee3 Cellular LTE-M/NB-IoT
- XBee Cellular LTE Cat 1
- XBee Cellular 3G
- XBee S2C
- XBee SX
- XBee SX 868
- XBee 900HP
- XBee XSC

Note XBee Multi Programmer supports all of the devices listed above in all hardware variants, including surface-mount (SMT), through-hole (TH) and XBee3 micro-mount technology (MMT).

Install XBee Multi Programmer application software

To download and install the XBee Multi Programmer application software:

1. Navigate to digi.com/xbeemultiprogrammer.
2. Click **Diagnostics, Utilities & MIBs**.
3. Click **Digi XBee Multi Programmer - Windows x86**.
4. When the file finishes downloading, run the executable file and follow the steps in the XBee Multi Programmer Setup Wizard.

Install USB drivers

The required USB drivers are automatically installed the first time you connect an XBee Multi Programmer tool to your computer. If the board drivers do not automatically install, use the following instructions to install the board drivers manually:

1. Go to the FTDI drivers page.
2. Locate the correct driver for your operating system.
3. For the Windows operating system, click the **setup executable** link. A zip file downloads.
4. Right-click the zip file and select **Extract All**. A folder displays with the setup file.
5. Double-click the setup file to run it.
6. Follow the steps in the installation wizard.

Install USB drivers for cellular modems

The XBee Multi Programmer application requires additional drivers to update the modems of newer XBee3 Cellular devices. If you have not installed them, use the following instructions to do so depending on the devices you want to program.

Note This step is only required if you are going to program the modem of the XBee3 Cellular LTE CAT 1 or XBee3 Cellular LTE-M/NB-IoT devices.

Drivers for XBee3 Cellular LTE CAT 1

1. Go to the Telit drivers page.
2. Select the Telit Windows Desktop Drivers Installer.
3. Run the executable file.
4. Follow the steps in the installation wizard.

Drivers for XBee3 Cellular LTE-M/NB-IoT

1. Download the u-blox drivers.
2. Uncompress the file and run the executable.
3. Follow the steps in the installation wizard.
4. When prompted select the **ETHERNET-DHCP** option.
5. After driver installation completes, reboot your computer.

CAUTION! If you are using Windows 7 or Vista, we highly recommend that you disable the drivers installation from Windows Update in order to speed up the modem update process. For more information on how to do this, see https://support.microsoft.com/en-us/help/2500967/how-to-stop-windows-7-automatically-installing-drivers.

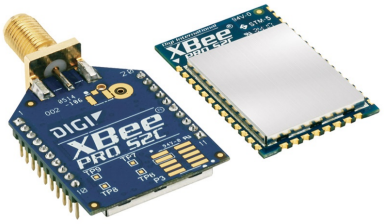
RF concepts and terminology

This section contains concepts related to RF devices and the XBee Multi Programmer application. Understanding these concepts will help you work with the XBee Multi Programmer.

RF modules

A radio frequency (RF) module is a small electronic circuit used to transmit and receive radio signals on different frequencies. Digi produces a wide variety of RF modules to meet the requirements of almost any wireless solution, such as long-range, low-cost, and low-power modules. The most popular wireless products are the XBee RF modules.

XBee RF modules



XBee is the brand name of a family of RF modules produced by Digi. They are modular products that make deploying wireless technology easy and cost-effective. Digi has made multiple protocols and RF features available in the popular XBee footprint, giving you flexibility to choose the best technology for your needs.

XBee RF modules are available in three form-factors, through-hole, surface-mount, and micro-mount, each with various antenna options. Most modules are available in the through-hole form factor and each share the same footprint.

Radio firmware

Radio firmware is program code stored in a radio module's persistent memory that provides the control program for the device. The main goal of the XBee Multi Programmer application is to program the same radio firmware in multiple devices simultaneously.

The XBee Multi Programmer gets the radio firmware that you program from the configuration profile loaded in the application. For more information about configuration profiles, see Configuration profile.

Configuration profile

A configuration profile is a snapshot of a specific radio firmware configuration. The profile is useful in a production environment when you need to set the same radio firmware and parameters on multiple radios. A configuration profile is an XPRO file containing the following elements:

- Radio firmware to be programmed in the device.
- Firmware settings to configure with their respective values.
- File system to be flashed in the XBee device.
- Pre and post-scripts to be executed during the programming process.
  - Pre-script is executed just before starting the programming process in the XBee device.
  - Post-script is executed when the entire programming process—firmware, settings and file-system—is finished.
- Other configurations and metadata to identify the profile, such as the flash firmware policy, profile description, and so on.

XBee Multi Programmer requires a configuration profile to be loaded before starting the programming process for the necessary information to be available.

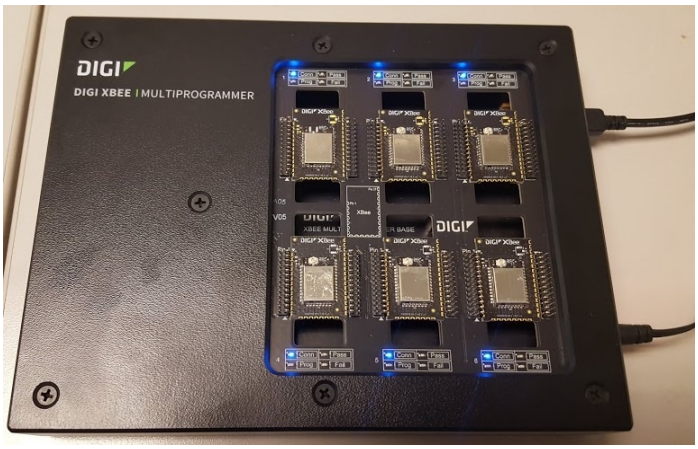
Note XCTU is required to generate and save configuration profiles. XCTU is a free multi-platform application designed to enable developers to interact with Digi RF modules through a graphical interface. See How to create a profile using XCTU for more information about generating profiles.

Hardware overview

This section provides information about the steps required to work with the XBee Multi Programmer tool.

General features

The XBee Multi Programmer tool is a hardware device designed to allow for concurrent XBee programming using the XBee Multi Programmer application software.



There are three types of interchangeable header boards, one per XBee form factor (each sold separately):

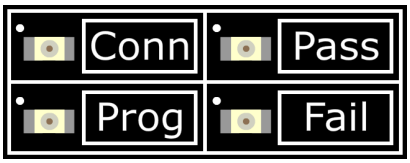
- Through-hole sockets (TH)
- Surface-mount sockets (SMT)
- Micro-mount sockets (MMT)

These boards are interchangeable depending on the needs of the user.

Note To change the current header board, you must temporarily remove the plastic enclosure of the hardware device.

Status LEDs

Each XBee socket of the XBee Multi Programmer header board has four LEDs that indicate the programming status of the XBee device attached to that socket.



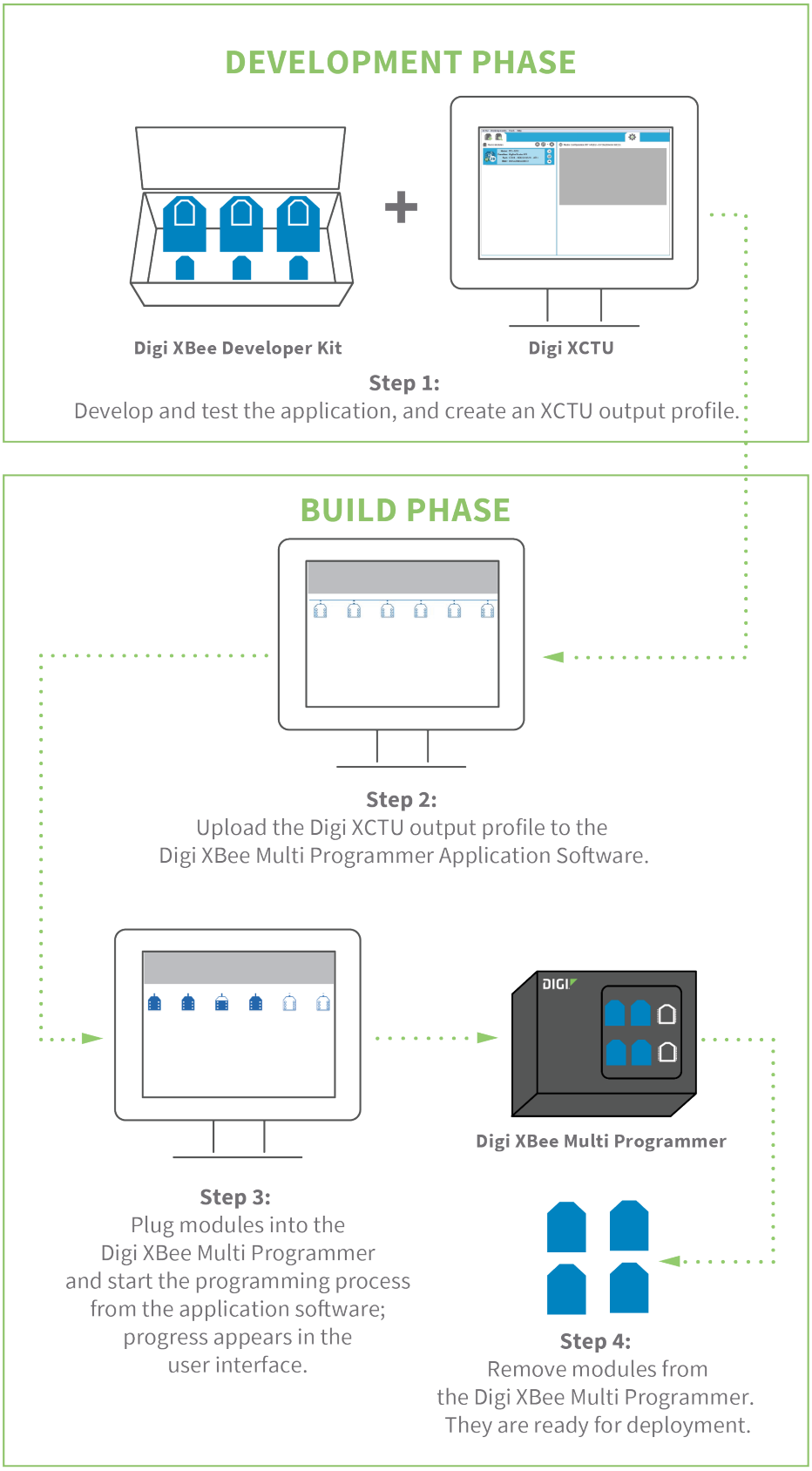
LED	Color	Description
Conn	Blue	Indicates whether an XBee device is attached to the socket (ON) or not (OFF).
Prog	Yellow/Orange	This LED blinks when the programming process is taking place. When finished, the LED is turned off and any of the **Pass** or **Fail** LEDs are turned on to indicate the final state of the programming operation.

LED	Color	Description
Pass	Green	This LED illuminates when the programming process of the XBee device attached to the socket finishes successfully.
Fail	Red	If the programming process finishes with any error, this LED illuminates.

Workflow

The normal workflow for this tool is:

1. Open the **XBee Multi Programmer** application software.
2. Connect the power supply and USB-C connector.
3. Select a valid profile.
4. Start the session.
5. Plug one or more XBee devices into the board and verify they are detected by the tool.
6. Whenever a device is successfully programmed, you can take it out of the socket and replace it with a new one without disconnecting the board.


DEVELOPMENT PHASE
Digi XBee Developer Kit
Digi XCTU
Step 1:
Develop and test the application, and create an XCTU output profile.
BUILD PHASE
Step 2:
Upload the Digi XCTU output profile to the
Digi XBee Multi Programmer Application Software.
DIGI
Digi XBee Multi Programmer
Step 3:
Plug modules into the
Digi XBee Multi Programmer
and start the programming process
from the application software;
progress appears in the
user interface.
Step 4:
Remove modules from
the Digi XBee Multi Programmer.
They are ready for deployment.

Connect the XBee Multi Programmer tool

The XBee Multi Programmer tool has a power source socket (left) and a USB-C socket (right).

To power on the XBee Multi Programmer, connect a 9 VDC or 12 VDC power supply to the power source socket. The current rating on the supply should be at least 1.5 A. The red LED next to the power socket should light up to indicate the programmer is properly powered. After powering the tool, connect a USB-C cable to the USB socket to allow for proper programming. The USB-C on this tool uses USB 2.0 technology.

Plug in the XBee devices

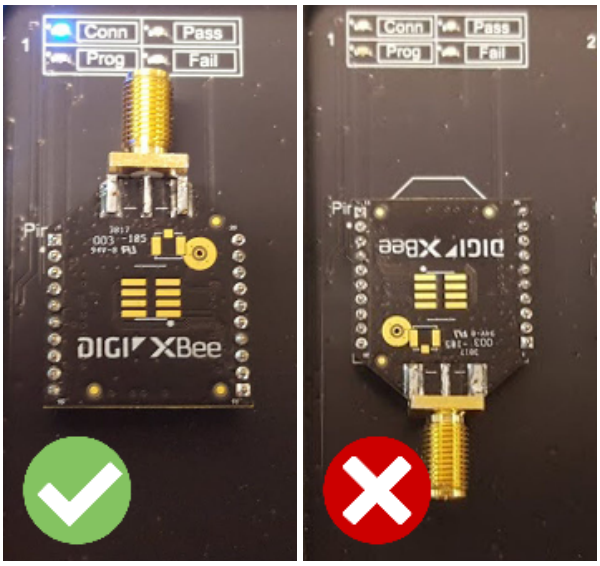
XBee Multi Programmer allows you to connect up to six of the same type of XBee devices. The application detects the connection event of any device.

To connect one XBee device to one of the sockets, complete the following steps for the appropriate XBee device.

Through-hole devices

Note See Through-hole break-in procedure.

XBee through-hole devices have a flat edge and a more angular, diagonal edge. Match that footprint with the white lines on your board and carefully insert it, taking care not to bend any of the pins.

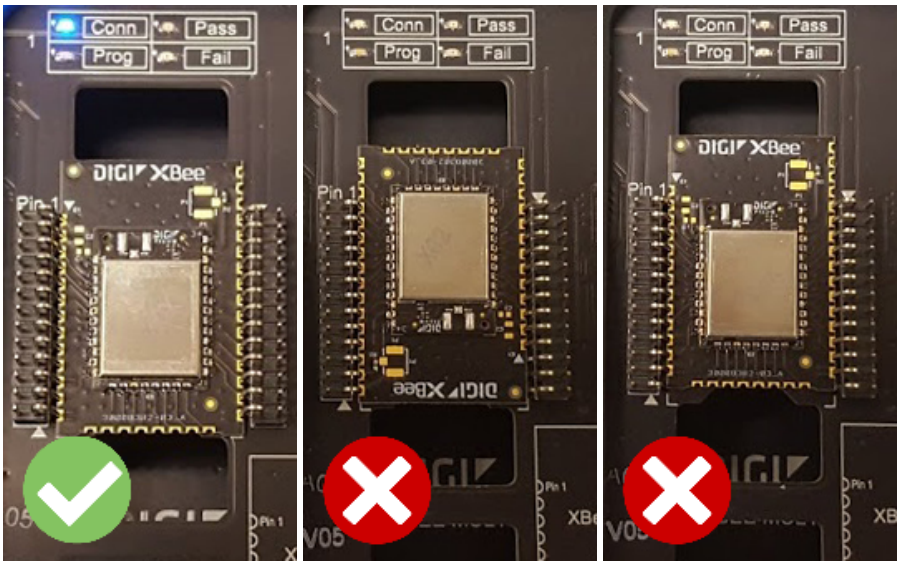


XBee through-hole alignment tip

Touch pins 10 and 11 first; this allows time to align the pins without false detection of the XBee prior to complete contact of the pins to the socket.

Surface-mount devices

For XBee surface-mount devices, align all XBee pins with the spring header and carefully push the device until it is hooked to the board. Ensure that pin 1 of the XBee device matches pin 1 of the socket.



XBee3 micro-mount devices

For XBee3 micro-mount devices, align all XBee pins with the spring header and carefully push the device until it is hooked to the board. Ensure that pin 1 of the XBee device matches pin 1 of the socket.

You can repeat this process on the remainder of XBee sockets. No waiting time is required between connecting or disconnecting devices.

Note Make sure the modules are correctly plugged into the board and their footprints match the white lines on the board.

Unplug the XBee devices

This board has been designed to allow for XBee devices to be unplugged whether the board is powered or not. The application will detect the disconnection event of any device.

Note The devices should not be disconnected while they are being programmed. Wait until the process is completed before disconnecting the devices.

To unplug an XBee from the board:

- For XBee through-hole devices, take the device from its top and bottom edges and pull up carefully, taking care not to bend any of the pins.
- For XBee surface-mount and micro-mount devices, holes have been cut in the programmer above and below each unit to facilitate gripping the device. Take care not to bend the socket pins when lifting the device out of the socket.

Through-hole break-in procedure

The XBee Multi Programmer through-hole has sockets that can stand up to many insertions before needing replacement. These sockets tend to be very tight when new and loosen over time. The first full insertions should be done without the programming software engaged and trying to program the units.

1. In each socket, plug in an XBee fully.
2. Remove the XBee by grabbing the top and bottom and wiggling it back and forth as it is being removed. Be careful not to bend the pins of the XBee during this process.
3. Repeat this five times before clicking **Play** and programming the devices.

The insertion and extraction will continue to be very stiff for the first 10 to 20 insertions. After 20 full insertions and extractions, the socket will still function well with a half insertion. Insert with enough force to make good contact with the pins (about half way into the sockets). When the programming completes the extraction will require less force.

Replace XBee socket boards

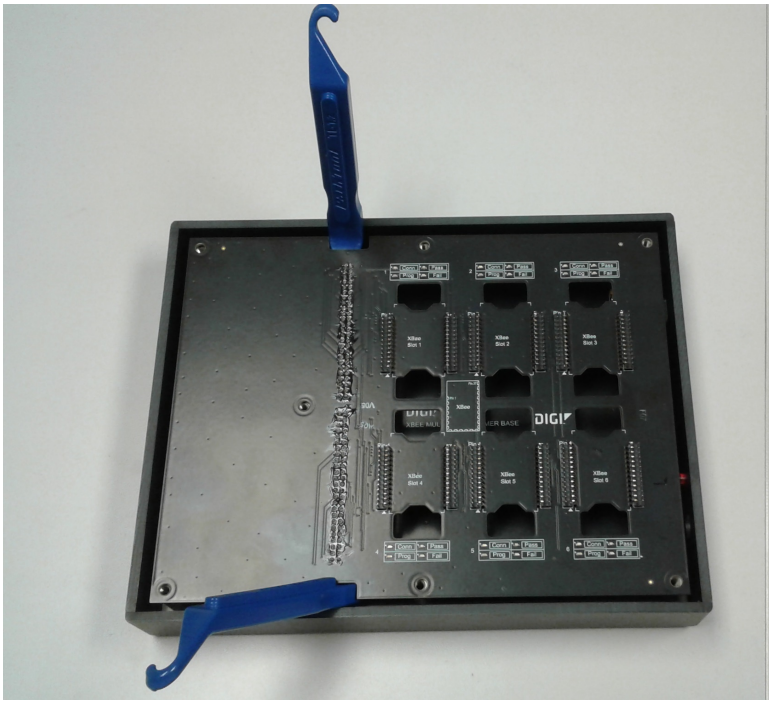
To replace XBee socket boards, order interchangeable board(s) by part number:

XBee device form-factor	Part number
Micro-mount (MMT)	XBEE-MP-MCRO-PCB
Surface-mount (SMT)	XBEE-MP-SMT-PCB
Through-hole (TH)	XBEE-MP-TH-PCB

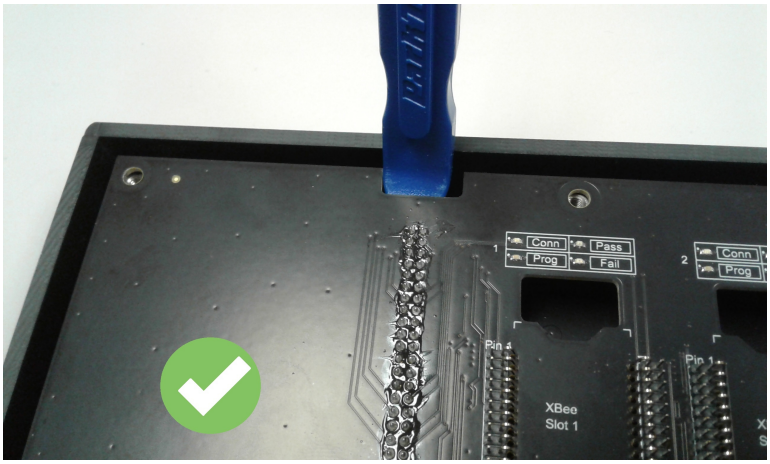
When you have the board(s):

1. Remove the USB connector and power cable.
2. Remove seven screws.
3. Remove the plastic cover. You may need a thin knife blade or a flat-head screw driver to help pry off the plastic cover.
4. Use the extraction levers shown in the following picture. They are XBee Multi Programmer PCB Extraction Levers.

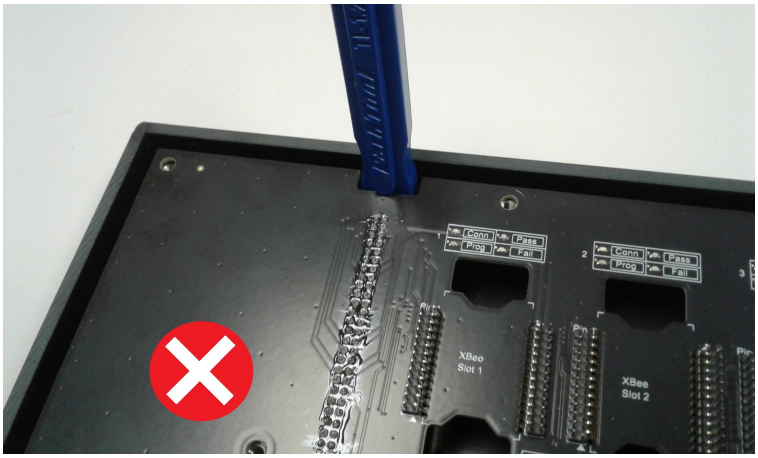
This picture shows the location to insert the levers:



This picture shows a lever inserted correctly:



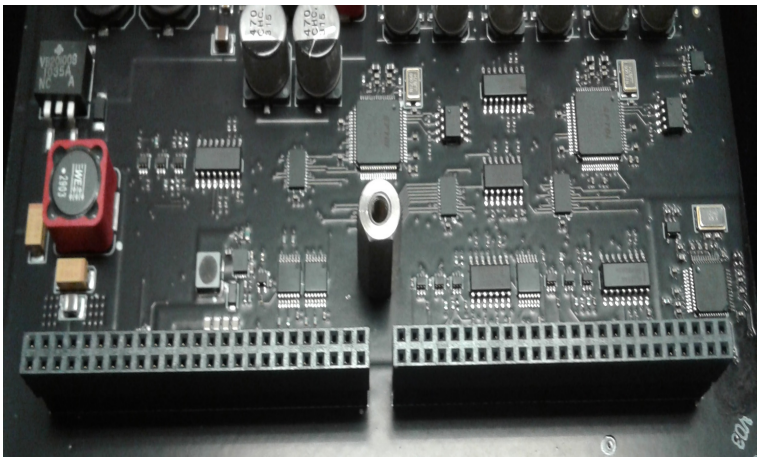
This picture shows a lever inserted incorrectly:

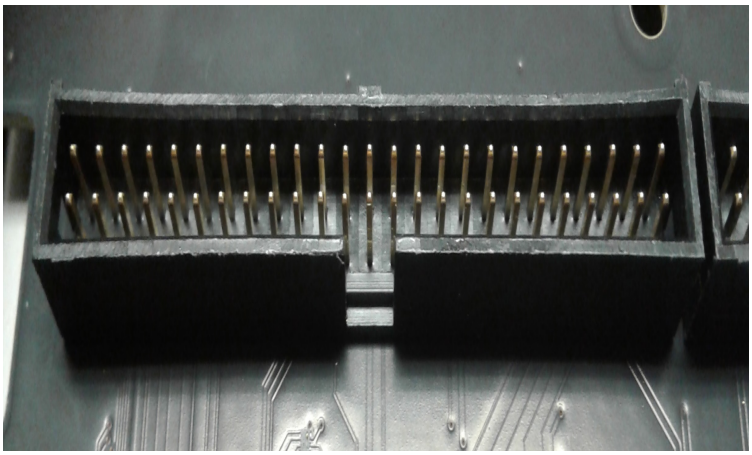


5. Apply pressure such that the top board rises evenly, so that pins are not bent or damaged during the removal process. Ensure that the lever is not flexing the plastic housing more than a few millimeters; if so, the extraction lever is inserted too far.

WARNING! Lifting only one side at a time will bend the pins!

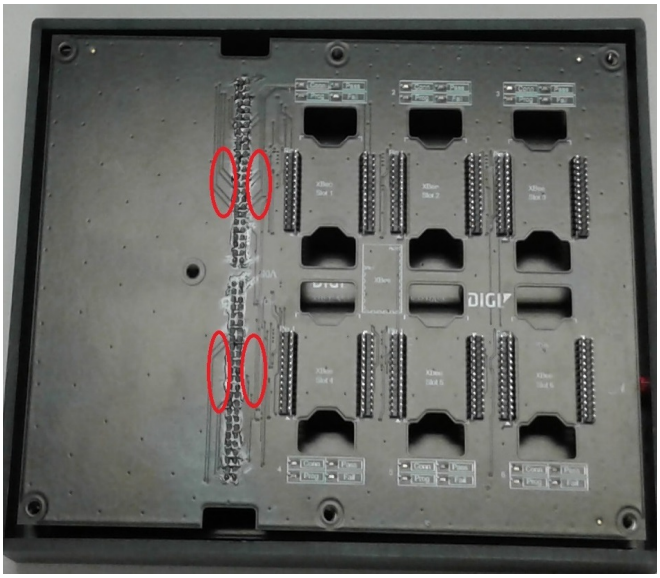
6. Inspect the connectors for any damaged or bent pins.





7. Insert the new board.

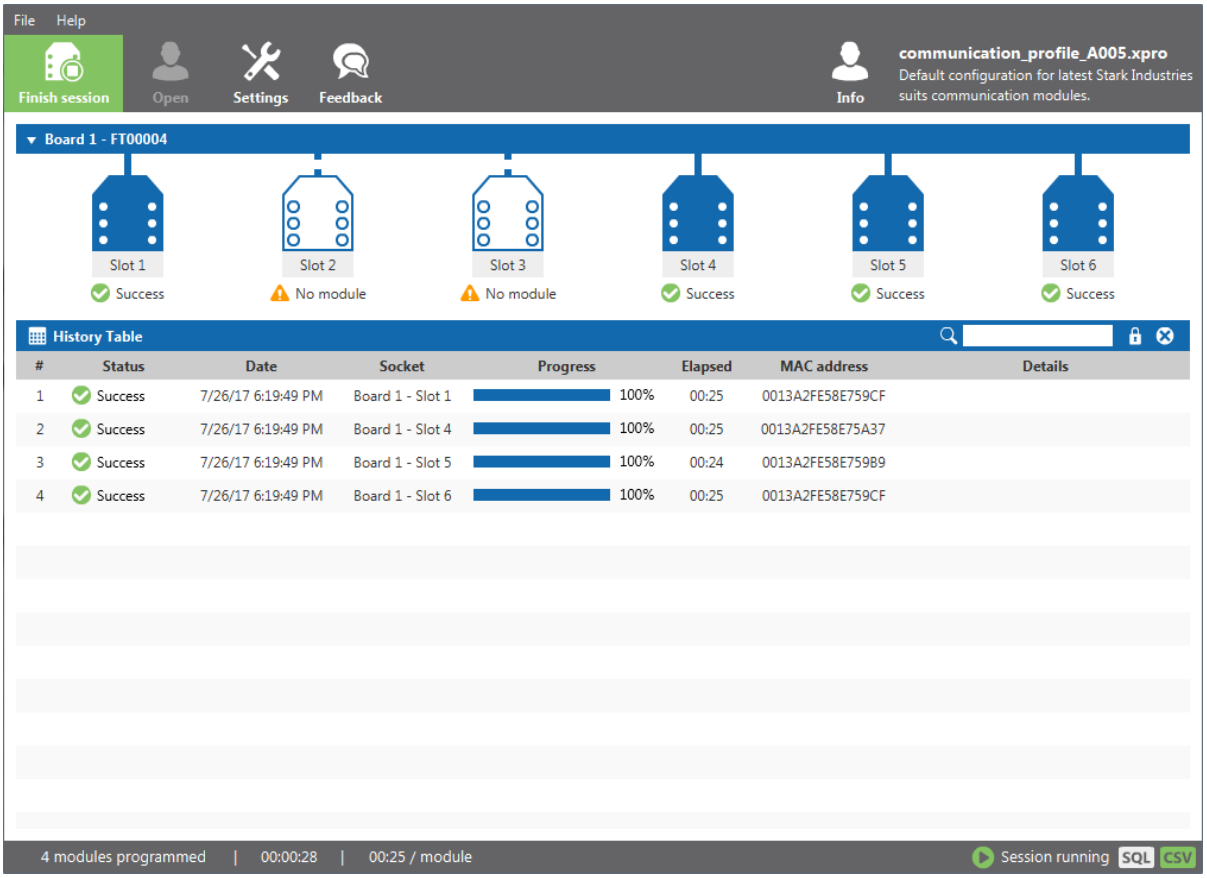
8. Press firmly on the board next to the connectors such that the board seats evenly until touching the standoffs.



9. Place the plastic cover on the XBee Multi Programmer again.
10. Place and tighten the screws.

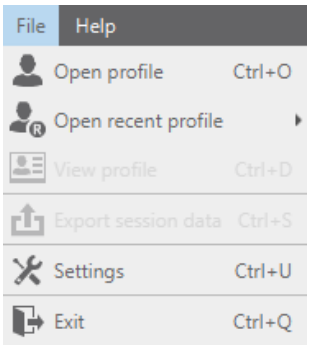
Application software overview

The XBee Multi Programmer application software is divided into five main sections:



Menu bar

The menu bar is located at the top of the user interface. Use the menu bar to access all XBee Multi Programmer features.

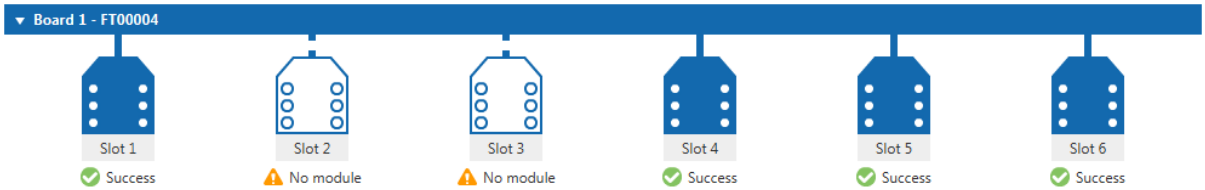


Toolbar

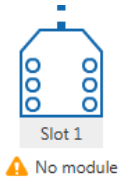
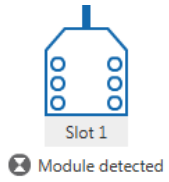
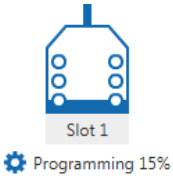
The toolbar is located below the menu bar near the top of the page.

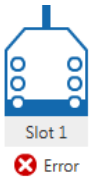
Board panels

Board panels are displayed below the toolbar of the application and represent physical XBee Multi Programmer devices. There is one board panel per multi programmer connected to your computer. When the application software detects a new XBee Multi Programmer, it is assigned an index starting at 1 and appears in the board panel.



Each board panel contains six slot elements representing the XBee slots of the actual XBee Multi Programmer board. When there is no module, the slot representation shows that information. The slot elements can display the following statuses:

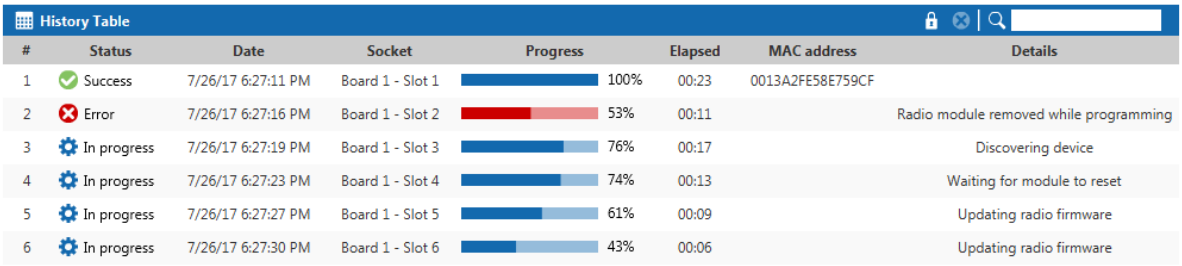
Status	Image
XBee radio module not detected	
XBee radio module detected	
Programming task in progress	
Programming task succeeded	

Status	Image
Programming task failed	

Note You can collapse board panels using their corresponding collapse button, providing more space for the history table. For more information, see History table.

History table

The history table is the main control of the application and is located in the center of the user interface. The history table displays all the programming tasks that have been finished and those that are taking place in the XBee slots of the XBee Multi Programmer tools connected to your computer.



#	Status	Date	Socket	Progress	Elapsed	MAC address	Details
1	Success	7/26/17 6:27:11 PM	Board 1 - Slot 1	100%	00:23	0013A2FE58E759CF	
2	Error	7/26/17 6:27:16 PM	Board 1 - Slot 2	53%	00:11		Radio module removed while programming
3	In progress	7/26/17 6:27:19 PM	Board 1 - Slot 3	76%	00:17		Discovering device
4	In progress	7/26/17 6:27:23 PM	Board 1 - Slot 4	74%	00:13		Waiting for module to reset
5	In progress	7/26/17 6:27:27 PM	Board 1 - Slot 5	61%	00:09		Updating radio firmware
6	In progress	7/26/17 6:27:30 PM	Board 1 - Slot 6	43%	00:06		Updating radio firmware

Each programming task displays the following information in the table:

- **#**. Index identifier of the programming task. This index starts at 1 and increments by 1 for each new programming task that is executed.
- **Status**. Status of the programming task. The available statuses are listed in the following table:

Status		Description
	Waiting	XBee device is connected in the socket, but the programming process has not started yet.
	In progress	Programming process is in progress.
	Success	Programming process finished successfully.
	Error	Programming process failed.

- **Date**. Complete date when the programming task started.
- **Socket.** XBee Multi Programmer board index and XBee socket ID of the device being programmed.
- **Progress**. Total percentage of the task's programming process.
- **Elapsed**. Total elapsed time in minutes and seconds for the programming task from beginning to end.
- **MAC address**. MAC address of the XBee device corresponding to the task.

  Note The MAC address does not display until the programming task writes the firmware settings in the radio module.

- **Details.** Information about the actions taking place in the programming process. If the programming task failed, this field displays the reason.

Note History table columns allow you to sort programming tasks in ascending or descending order based on the column criteria.

The header pane of the history table contains a small toolbar located at the right side that allows you to perform some tasks in the table. For more information, see Table toolbar.

Table toolbar

The table toolbar is located at the right side of the history table header and contains the following elements:

Name	Description	Control image
Lock scroll button	Toggles to lock scroll and unlock.	
Clear completed tasks button	Allows you to clear completed programming tasks.	
Search box	Allows you to search for programming tasks.	

Search for programming tasks

You can use the search box of the history table toolbar to find programming tasks by Status, Board, Slot and unique address. Type your search expression in the search box.

You can type the following search prefixes:

Search prefix	Search by
STATUS:	Status of the programming task.
BOARD:	Board number of the device associated with a programming task.
SLOT:	Slot number of the device associated with a programming task.
ADDR:	Unique address of the device associated with a programming task.

Note By default, the search box filters by address (if no prefix is added), and all the filters contain a colon. It is necessary to specify the column you want to filter.

For example, ***4F*** shows only the devices with a unique address that contains **4F**. For more information, see the Search examples.

You can also use a wildcard if you do not want to specify the entire parameter or if you want to find more than one programming task.

Wildcard	Equals
*	Any string
?	Any character
\	Escape for literals (i.e. *, ?, or \)

Search examples

The following table lists some examples of searches using prefixes and wildcards:

Description	Example Search box text
Get all the tasks done in board 2.	BOARD:2 |BOARD:*2 | BOARD:Board 2
Get all the tasks for modules whose unique address is of this range: 0013A20040F2XXXX.	ADDR:0013A20040F*
Get all the tasks for modules whose unique address is of this range: 0013A20040F213XE.	ADDR:0013A20040F213?E
Get all the tasks that failed.	STATUS:Err | STATUS:Err* | STATUS:Error

It is always necessary to include the complete unique address in the search box or a partial unique address with wildcards.

As shown in the previous table, you can search **BOARD** and **SLOT** columns using only the number of the target board or slot. For example, type **SLOT:4** to show only the devices programmed with the Slot 4.

For the **STATUS** column, you can also start typing a valid status value in the search box as follows (without using wildcards): **STATUS:Su**

In this case, only successfully programmed devices will be displayed. To see the valid status values, see History table.

Note Blank spaces are included. Be careful not to include spaces after the colon that separates the prefix and the search expression.

STATUS:Error (correct).

STATUS: Error (incorrect) - There is a space between the colon and the expression.

Lock scroll

Whenever a new programming task is generated and added to the table, it scrolls automatically to that task. Click the **lock scroll** button on the toolbar to disable this feature and maintain the scroll at its current position. Click the **lock scroll** button a second time to enable the feature.

Clear completed programming tasks

After you have finished programming, the programming tasks display a status of **Success** or **Error**. Click the **Clear completed tasks** button to clear all the completed tasks from the table and view only those in progress.

Status bar

The status bar is located at the bottom of the screen and displays the programming session statistics, such as the number of radio modules programmed, running time, average programming time per module, and the session status (for example, running or stopped). When the application is looking for new updates or installing them, the status bar also displays the status of the process.

The SQL and CSV status icons located at the right side of the status bar indicate whether the session is being recorded in CSV and stored in a database.

Note By default, all the sessions are always recorded in CSV format, which allows you to export it later to a CSV or PDF file, but you can also store the session in a database. For more information, see Store sessions in a database.

Status icon	Description
	Session is not being saved in CSV format. Either session is not started yet or there was a problem generating the CSV file.
	Session has started and is being saved in CSV format. You will be able to export the session later to a CSV or PDF file.

Status icon	Description
	Session is not being recorded in a database. Either the session is not started yet, the database recording is not enabled, or there was a problem writing to the database.
	Session has started and is being recorded in a database.

Connect an XBee Multi Programmer board

The XBee Multi Programmer tool can only detect a specific type of board called the XBee Multi Programmer board. For more information about this board, see Hardware overview.

Note The application automatically detects when an XBee Multi Programmer board is plugged into the PC. You can attach the board before or after the application has started.

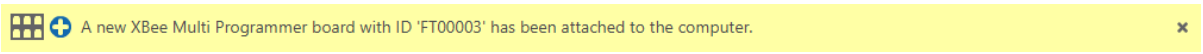
Steps to properly attach the board to your PC

Before plugging the board into your PC, make sure the board is powered externally. To properly attach the board to your PC:

1. Connect a 9 VDC or 12 VDC power supply to the power source socket of the board (the current rating on the supply should be at least 1.5 A).
2. Attach the board to the PC using a USB host cable.

If the application is not running when the board is connected to the PC, the application detects the board at startup. If you attach the board once the tool is running, the tool may take a few seconds before it detects the board.

Once a board is detected, a notification appears indicating the board ID of the newly detected board:

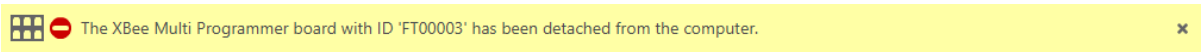


When you add a new board to the tool, a graphic representation of it is displayed as a board panel.

Note Once the application is running, detecting a new board can take several seconds.

Detach an XBee Multi Programmer board

The process of detaching or connecting a board is basically the same. The application detects if the board's USB host cable is disconnected from the PC. After a few seconds, the tool displays a notification indicating the board is disconnected.



At this point, if any module was being programmed, the update process fails and the appropriate message is displayed.

To safely remove the board:

1. Make sure there are no programming tasks in progress.
2. Remove the board from the PC by disconnecting the USB host cable.
3. Disconnect the board from the external power supply.

Load a profile

Before programming the radio modules, you must load the configuration profile containing the information being programmed. For more information about configuration profiles, see Configuration profile.

Load a new profile

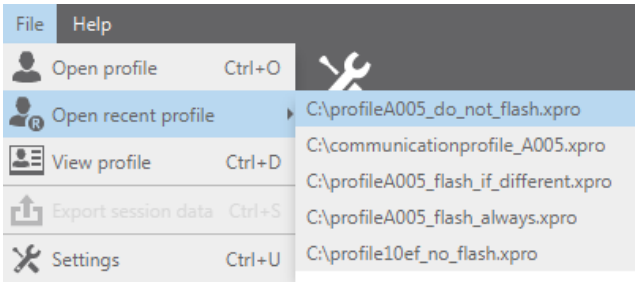
To load a new profile with the XBee Multi Programmer tool:

1. On the main window, click the **Open** button  . An **Open file** dialog appears prompting you for the configuration file you want to load.
2. Locate the configuration profile (XPRO file), and click **Open**.

Load a recent profile

To load a previously loaded profile with the XBee Multi Programmer tool:

1. On the **File** menu, hover over the **Open recent profile** option to display a list of up to five recently opened profiles.



2. Select the profile you want to load.

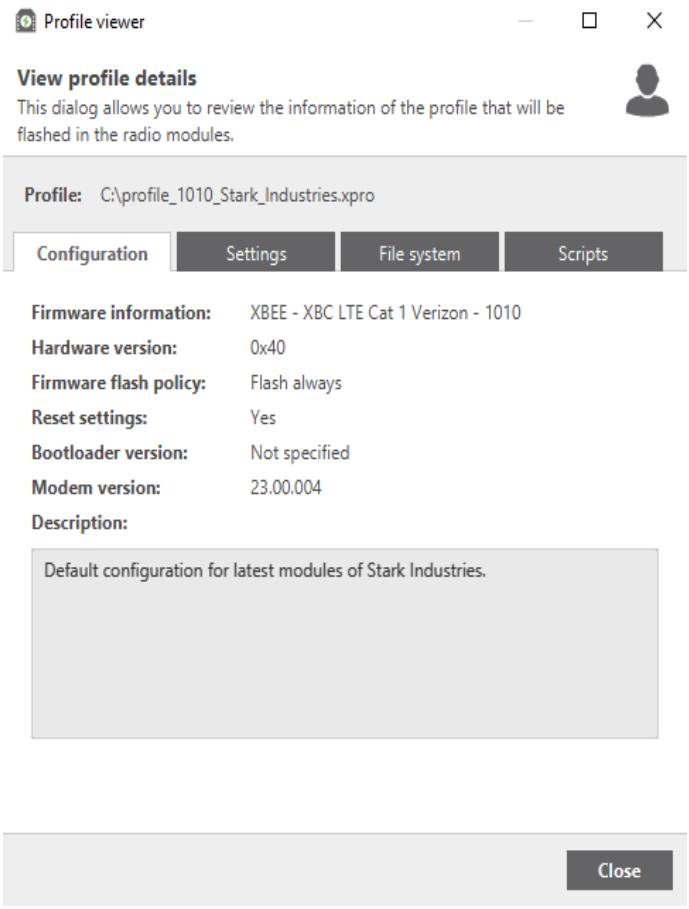
After a profile loads, the profile panel on the toolbar displays the configuration profile path and description and continues to display until a new profile is loaded. For more information about the toolbar, see Toolbar.

Note For more information about viewing the details of the loaded profile, see View the profile details.

View the profile details

The XBee Multi Programmer tool allows you to view the details of the currently loaded profile. Access the Profile viewer after loading a configuration profile to view the details.

Click the **Info** button in the top right corner of the toolbar of the main window . The **View profile details** window appears.



This profile viewer window is divided in four sections: Configuration, Settings, File system and Scripts.

Configuration

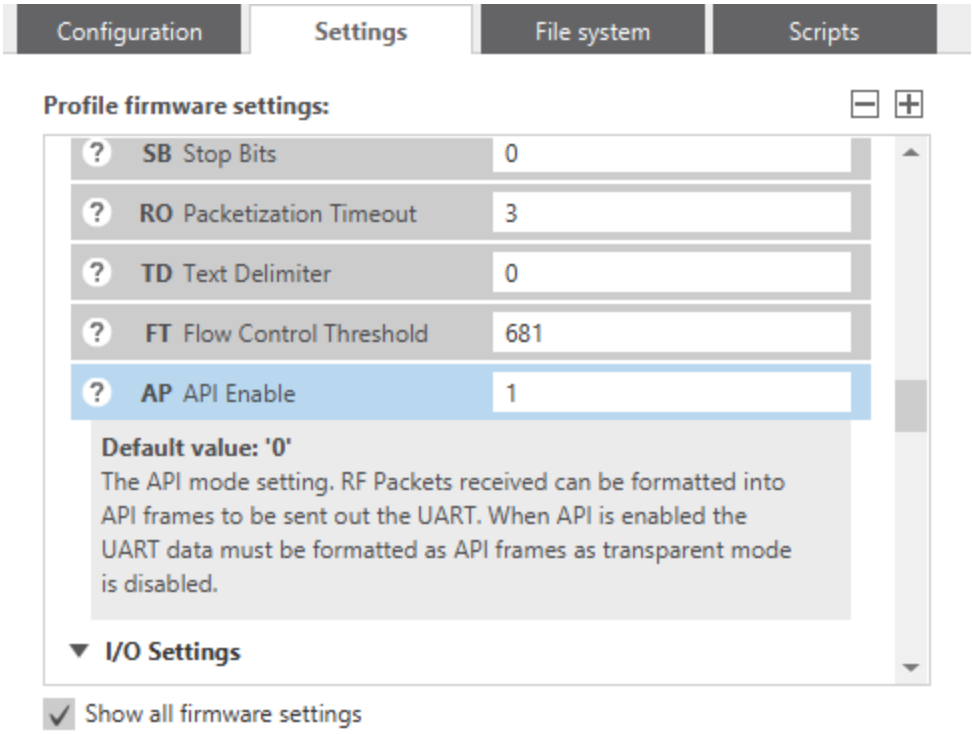
This tab shows general information of the profile:

- Firmware and hardware information this profile is suitable for.
- Firmware flash policy: Flash always, Flash if firmware is different, Do not flash firmware.
- Whether the device's settings will be reset prior to loading the profile settings.
- Bootloader version, in case the firmware requires a specific one.
- Modem version, in case the firmware requires a specific one (only for cellular devices).
- Description of the profile (optional).

Settings

By default, this tab shows the firmware settings that have been modified for the current profile.

To display all the configurable settings for the selected profile, check the **Show all firmware settings** checkbox:



You can hide the non-modified settings again by clearing the **Show all firmware settings** checkbox.

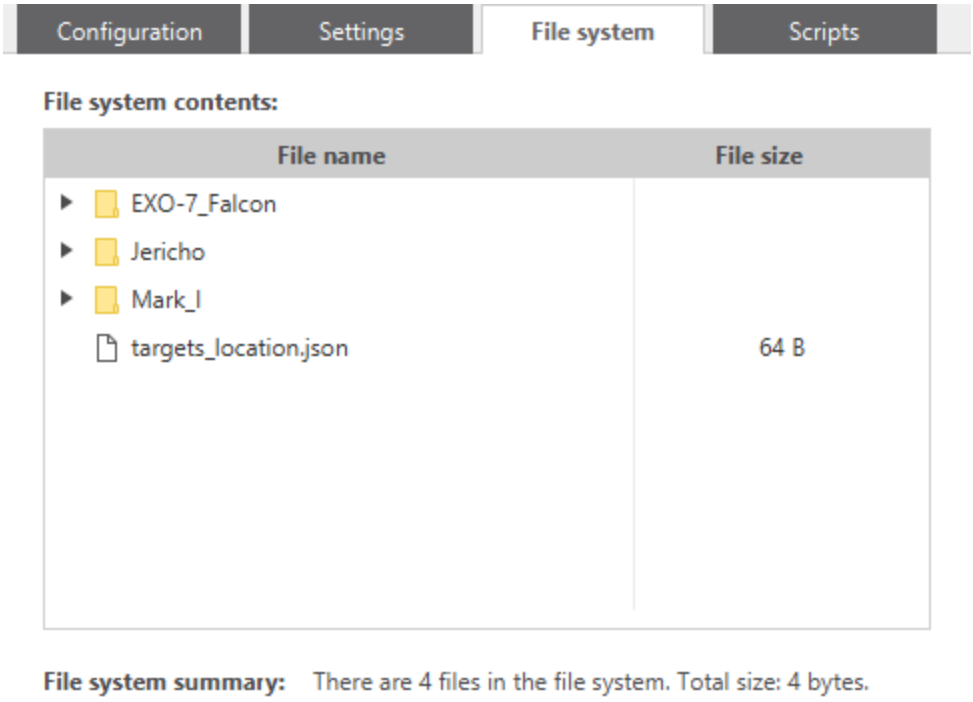
The values configured in the profile are represented with a blue background whereas the rest remain gray. To view the specific setting description (see the ID in the previous graphic), click the help button (?). Each firmware setting has a help button.

All the settings sections, such as **Network**, are collapsible to help you display only parts of the profile you want to see. You can expand or collapse all sections independently or as a group using the plus (**+**) and minus (**-**) buttons from the top right corner of the window.

Note All of the settings in the XBee Multi Programmer application are view-only. To modify settings, see the *XCTU User Guide*.

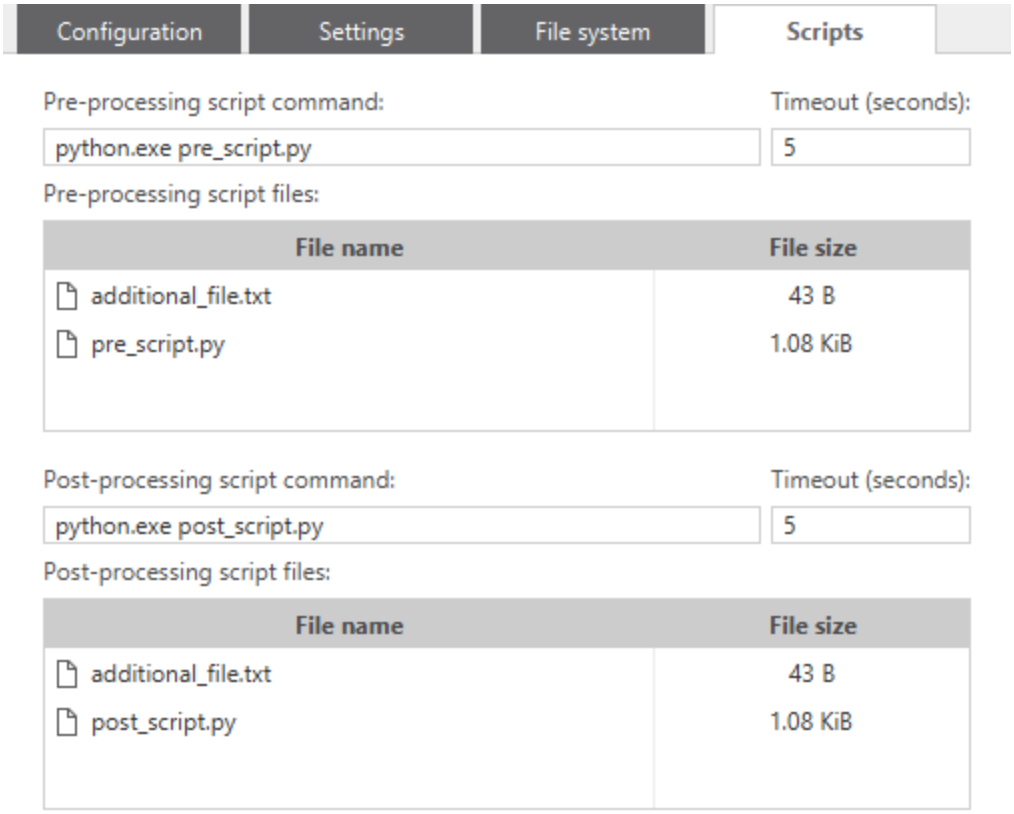
File system

If the profile contains a file system to be flashed on the XBee device, this tab is enabled and shows its contents and a summary with the number of files and total size. You can navigate through the file system by expanding or collapsing the folders.



Scripts

Profiles can be also configured to run a pre-script and/or a post-script during the programming process of an XBee device. If any script is configured in the profile, this tab is enabled and shows the scripts commands to run as well as the list of files associated to each one.


Configuration
Settings
File system
Scripts
Pre-processing script command:
Timeout (seconds):
python.exe pre_script.py
5
Pre-processing script files:
File name
File size
additional_file.txt
43 B
pre_script.py
1.08 KiB
Post-processing script command:
Timeout (seconds):
python.exe post_script.py
5
Post-processing script files:
File name
File size
additional_file.txt
43 B
post_script.py
1.08 KiB

Program the XBee devices

Once the profile is loaded, you can start programming the XBee devices. This process uses the loaded profile to update the firmware of the device (depending on the specified program policy) and load the setting values.

When you start the process, the tool automatically creates a programming session. This session contains the following information:

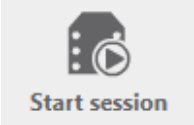
- Devices that have been programmed.
- Start and end time, total time, and average time per device.
- Profile information.

This information appears in the main window and can also be saved when you finish the session.

This section explains how to:

Start the programming session

To start the programming session, click the **Start session** button located on the toolbar:



If there are any XBee devices connected to the XBee Multi Programmer board, the application starts programming those devices. If none are connected, the application waits until you connect any device and then starts the programming process automatically.

The XBee Multi Programmer board has four LEDs in each slot to indicate the status of the device:

- Connected (blue)
- Being programmed (yellow/orange)
- Finished successfully (green)
- Finished with error (red)

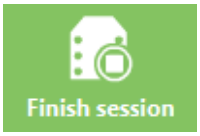
Note For more information about the XBee Multi Programmer board, see Hardware overview.

The programming process of each device is independent of the others and runs until you stop the session. When a device has been programmed, either successfully (green LED) or with errors (red LED), you can detach it from the board and connect another device to that slot. The application software detects the connection and automatically starts the programming process.

You can view the current progress of the programming process and some additional information both in the board panels and in the history table.

Finish the programming session

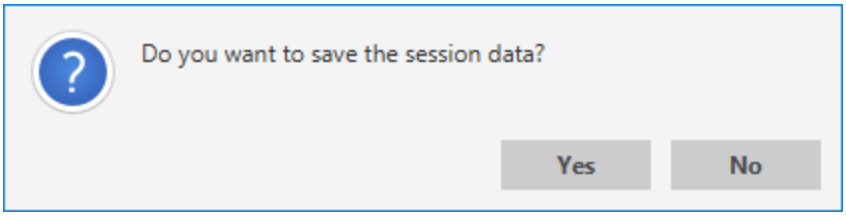
When you have programmed all your devices, click the **Finish session** button to end the programming session.



This action is required if you want to change the profile that is being used to program the devices or if you want to export the session report.

WARNING! If you click **Finish session** when there are still modules being programmed, you are prompted to confirm that you want to stop the process. Devices could become unresponsive if you finish a session while they are being programmed.

Once you press the **Finish session** button, the application prompts you to save the session report.



Click **Yes** to save the session data in that moment or **No** to save the session data later. See Export the session report for more information about exporting the session data.

Store sessions in a database

The XBee Multi Programmer application software allows you to save all the information related to the sessions in a database, such as the profile used and its settings, user, session information and devices programmed. This is helpful if the application is used in multiple computers, and all the information is stored in the same place.

To store sessions in a database, you must first enable the option in the settings.

1. Click the **Settings** button of the toolbar or select the **File > Settings** option on the menu. The Settings dialog appears.
2. On the left side of the Settings dialog, select **Storage**.
3. Check the **Save sessions in a database** checkbox.
4. Enter the database server address, port, user and password.
5. Click **Apply** and then **Close**.

Note Only MySQL is supported for database recording. The database user must have the following privileges: CREATE, EXECUTE, INSERT, REFERENCES, SELECT and UPDATE.

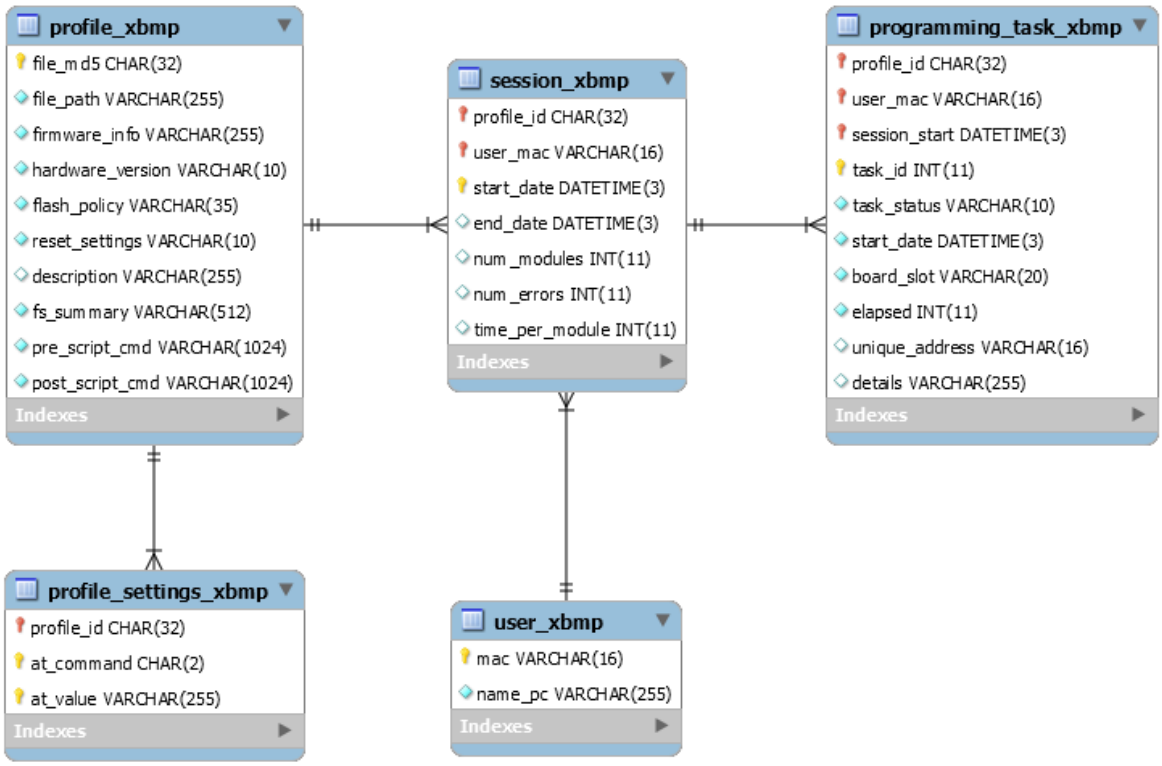
When you start the programming session, the application software creates a new database called **xbee_Multi_programmer** and the required tables to store the information (see Database structure for more information). As the devices are programmed, the application automatically stores their information in the database.

The **SQL** indicator of the status bar toggles green when the session is being stored in a database.

For more information about the contents of the database, see Database structure.

Database structure

The **xbee_multi_programmer** database has five tables:



The following tables provide information about the contents of the database tables.

profile_xbmp

The **profile_xbmp** table contains information about the profiles used in the tool.

Column	Type	Description
file_md5	**char(32)**	Primary key. MD5 hash of the profile zip package.
file_path	**varchar(255)**	Profile path.
firmware_info	**varchar(255)**	Firmware information.
hardware_version	**varchar(10)**	Hardware information.
flash_policy	**varchar(35)**	Flash policy.
reset_settings	**varchar(10)**	Reset settings.
description	**varchar(255)**	Profile description.
fs_summary	**varchar(512)**	File system summary.
pre_script_cmd	**varchar(1024)**	Pre-script command.
post_script_cmd	**varchar(1024)**	Post-script command.

profile_settings_xbmp

The **profile_settings_xbmp** table contains the different settings associated to the profiles.

Column	Type	Description
profile_id	**char(32)**	Primary key. References to the **file_md5 column of profile_xbmp**.
at_command	**char(2)**	Primary key. AT command.
at_value	**varchar(40)**	Primary key. AT command value.

user_xbmp

The **user_xbmp** table contains information about the users of the tool.

Column	Type	Description
mac	**varchar(16)**	Primary key. MAC address of the user's computer.
name_pc	**varchar(255)**	Computer name.

session_xbmp

The **session_xbmp** table contains information about the sessions.

Column	Type	Description
profile_id	**char(32)**	Primary key. References to the **file_md5 column of profile_ xbmp**.
user_mac	**varchar(16)**	Primary key. References to the mac column of **user_xbmp**.
start_date	**datetime**	Primary key. Date and time when the session started.
end_date	**datetime**	Date and time when the session finished.
num_modules	**int(11)**	Number of devices programmed successfully.
num_errors	**int(11)**	Number of devices not programmed due to an error.
time_per_ module	**int(11)**	Average programming time per device.

programming_task_xbmp

The **programming_task_xbmp** table contains information about the programming tasks.

Column	Type	Description
profile_id	**char(32)**	Primary key. References to the file_md5 column of **profile_xbmp**.
user_mac	**varchar(16)**	Primary key. References to the mac column of **user_xbmp**.
session_ date	**datetime**	Primary key. References to the **start_date** column of **session_xbmp**.
task_id	**int(11)**	Primary key. Index identifier of the programming task.
task_status	**varchar(10)**	Status of the programming task.

Column	Type	Description
start_date	**datetime**	Complete date at which the programming task started.
board_slot	**varchar(20)**	XBee Multi Programmer board index and socket ID of the device.
elapsed	**int(11)**	Total time (in seconds) elapsed since the programming task started until finished.
mac_ address	**varchar(16)**	MAC address of the XBee device corresponding to the task.
details	**varchar (255)**	Information about the error in case the programming task failed.

Export the session report

Once you have finished the programming process, you can save a report and export it to a PDF or CSV file.

Complete the following steps to export the session report:

1. Select the **File > Export** session data option from the menu.
2. Select the destination folder, the name of the file and the file type of the report:
   - PDF file (*.pdf)
   - CSV file (*.csv)
3. Click **Save** to generate the session report.

PDF report

The PDF report provides the following information:

1. **Session statistics**. Start and end time, total session time, number of devices programmed successfully, and number of errors and average time per module.
2. **Profile information**. Profile path, firmware information, hardware version, flash policy, reset settings, description, bootloader version, modem version, file system summary, pre-script, post-script and list of settings.
3. **History table**. List with all the devices that have been programmed (both successfully or with errors) and details.

CSV report

The CSV report has four sections:

1. **Session statistics**. Start and end time, total session time, number of devices programmed successfully, number of errors and average time per module.
2. **Profile information**. Profile path, firmware information, hardware version, flash policy, reset settings, description, bootloader version, modem version, file system summary, pre-script and post-script.
3. **Firmware settings**. List of firmware settings and their values.

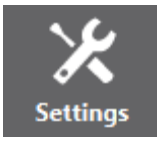
4. **History table**. List with all the devices that have been programmed (both successfully or with error) and details.

The following is an example of the CSV file:

```
Start,End,Total time,Successes,Errors,Time per module
6/28/17 4:22:42 PM,6/28/17 4:43:50 PM,00:21:07,6 (100 %),0 (0 %),01:05

Profile,Information,HW version,Flash policy,Reset
settings,Description,Bootloader version,Modem version,File System summary,Pre-
script,Post-script
C:\profile_31010.xpro,"XBEE - XBC LTE Cat 1 AT&T - 31010",0x49,Flash
always,Yes,"",1.6.7,23.00.303,"","pre:python.exe pre_
script.py","post:python.exe

Setting,Value
NI (Node Identifier),DIGI
AP (API Enable),1

Status,Date,Socket,Elapsed,MAC address,Details
SUCCESS,6/28/17 4:22:42 PM,Board 1 - Slot 6,00:56,0013A200XXXXXXXX,""
SUCCESS,6/28/17 4:22:42 PM,Board 1 - Slot 1,00:58,0013A200XXXXXXXX,""
SUCCESS,6/28/17 4:22:42 PM,Board 1 - Slot 3,01:02,0013A200XXXXXXXX,""
SUCCESS,6/28/17 4:22:42 PM,Board 1 - Slot 5,01:03,0013A200XXXXXXXX,""
SUCCESS,6/28/17 4:22:42 PM,Board 1 - Slot 2,01:05,0013A200XXXXXXXX,""
SUCCESS,6/28/17 4:22:42 PM,Board 1 - Slot 4,01:12,0013A200XXXXXXXX,""
```

Settings

This section describes how to configure several XBee Multi Programmer settings. To open the Settings dialog, click the **Settings** button of the toolbar or select the **File > Settings** option of the menu.

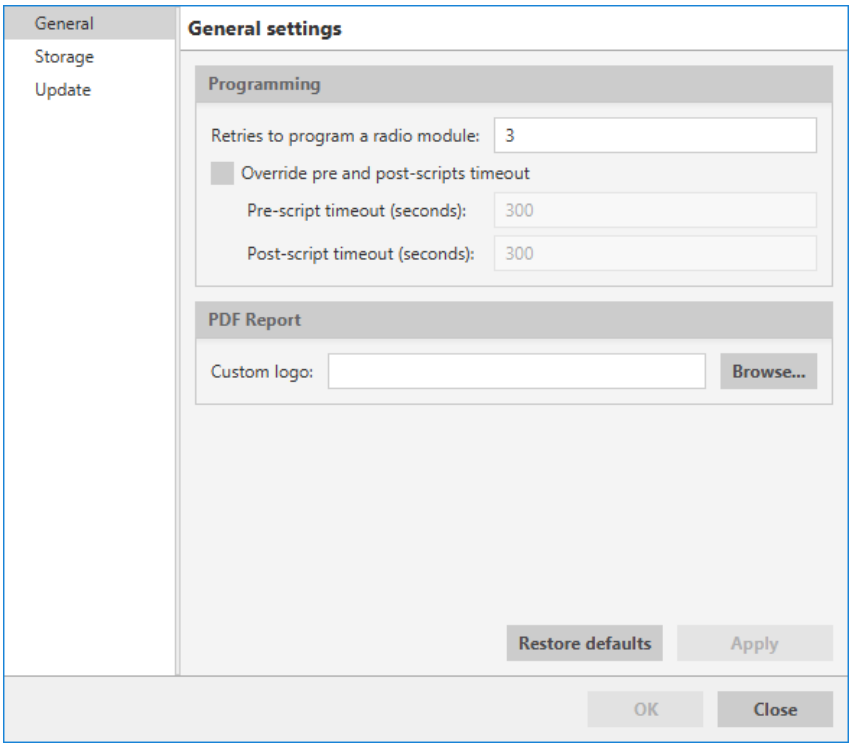


The setting categories are listed on the left side of the Settings dialog. You can configure settings for the following categories:

General settings

To configure some general settings of the application, complete the following steps:

1. Click the **Settings** button on the toolbar or select the **File > Settings** option of the menu. The Settings dialog appears.
2. On the left side of the **Settings** dialog, select **General**.



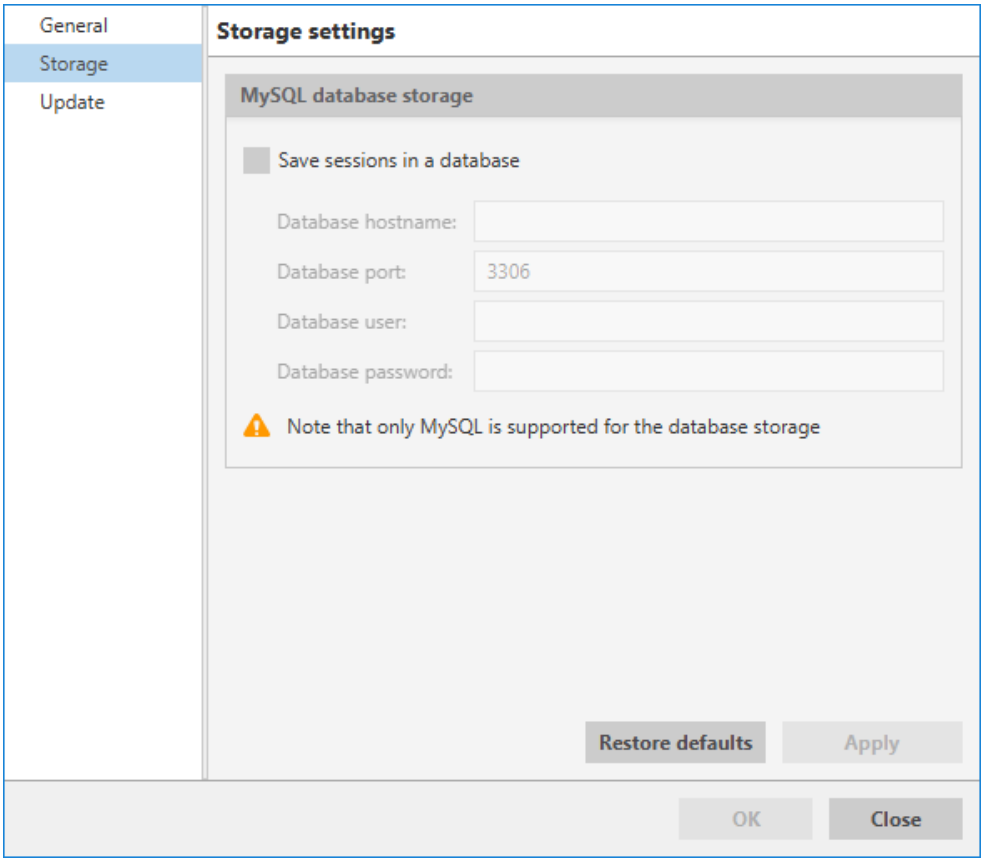
3. Configure the following settings and click **Apply**.

Setting	Description
Retries to program a radio module	If the programming process of a device fails, it retries the process the number of times you specify in this setting.
Override pre and postscripts timeout	Check this setting to use a different pre and post-scripts timeout than the one established in the configuration profile. Once the setting is checked you can enter the new timeout values for the pre and post-scripts (in seconds).
Custom logo	Configures the logo to be drawn in the PDF report document. If the custom logo value is empty, the application draws the Digi logo by default. See Export the session report for more information.

Storage settings

Complete the following steps to enable and configure storage mechanisms for the programming sessions.

1. Click the **Settings** button of the toolbar or select the **File > Settings** option of the menu. The Settings dialog appears.
2. On the left side of the **Settings** dialog, select **Storage**.



3. Configure the following settings and click **Apply**.

Setting	Description
Save sessions in a database	Enables or disables the database storage mechanism for the programming sessions. When this setting is selected, all of the MySQL database storage settings are enabled.
Database hostname	Address of the database server.
Database port	Port to access the database.
Database user	User value of the database account.

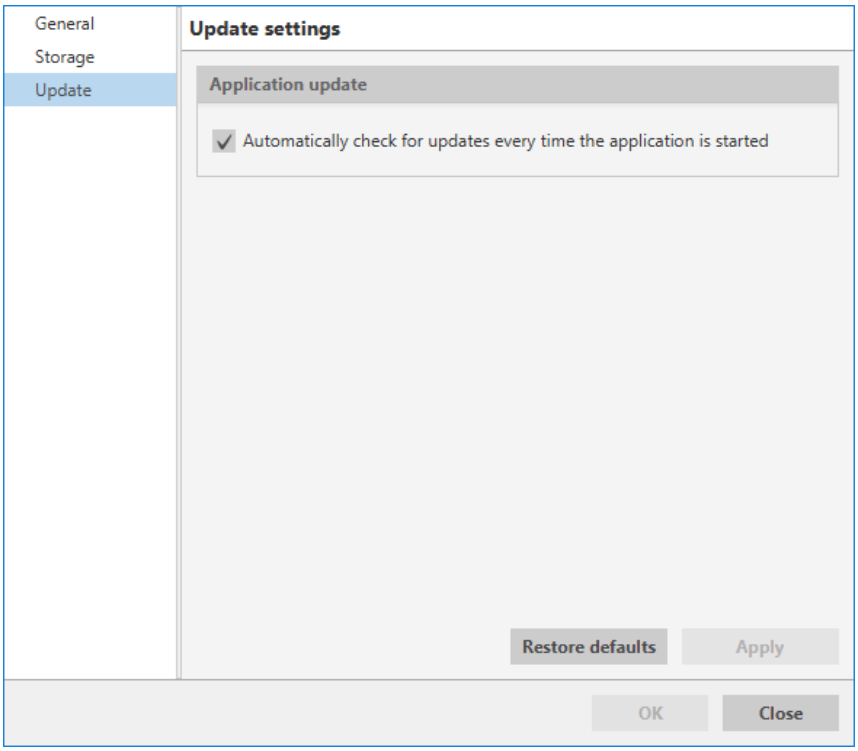
Setting	Description
Database password	Password value of the database account.

Note Only MySQL is supported for the database storage.

Update settings

Complete the following steps to enable or disable the option to automatically check for new application updates when the application is started.

1. Click the **Settings** button of the toolbar or select **File > Settings** option of the menu. The Settings dialog appears.
2. On the left side of the **Settings** dialog, select **Update**.



3. Select or clear the **Automatically check for updates every time the application is started** checkbox to enable or disable the automatic updates feature.
4. Click **Apply**.

Update software

XBee Multi Programmer allows you to automatically update the application without downloading any extra files. This process can be configured to execute automatically, but you can also execute it manually at any time. For more information about configuring automatic updates, see Update settings.

If you have enabled the automatic updates, you may be notified about software updates when you open XBee Multi Programmer. You should always run the latest version of the tool.

1. When a new version is available, a notification window appears asking you if you want to update the application.

2. Click **Yes** to start the update process.
3. When the installation process is finished, you must restart XBee Multi Programmer so new changes can be applied. When prompted, click **Yes** to restart the tool.

You can also check for updates and manually update the tool by clicking **Help > Check** for updates.

Note Click the **Run in background** button of the progress dialog to execute this process in the background. The status bar displays the update process.

How-to articles

The following pages contain how-to articles describing some of the most common procedures to work with the XBee Multi Programmer tool.

How to create a profile using XCTU

XCTU is required to generate and save configuration profiles. This section provides the steps to create a profile using XCTU.

For more information about configuration profiles, see Configuration profile.

Step 1: Create the profile

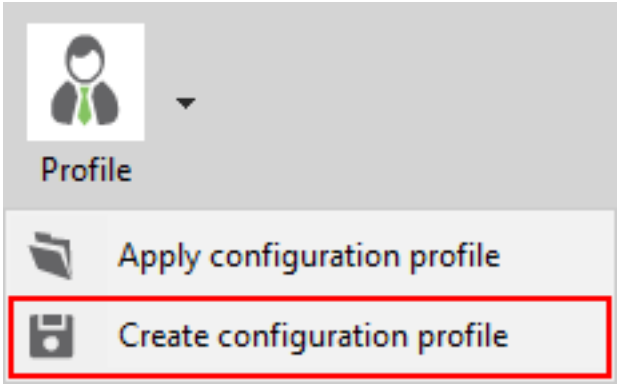
XCTU offers two methods for configuring a profile:

- Configuration working mode
- Profile Editor

Configuration working mode

Configuration working mode allows you to quickly generate profiles for devices attached to your computer.

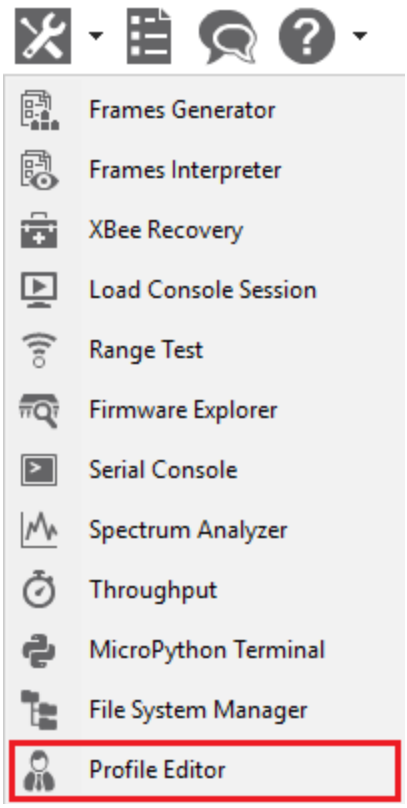
1. In the main window of XCTU, switch to **Configuration working mode** .
2. Select a device from the device list.
3. Configure the radio module with the appropriate values. You only need to change the values; they do not need to be written to the specific device.
4. Click the **Configuration profiles** drop-down menu on the configuration toolbar and select **Create configuration profile**.



Profile Editor

The Profile Editor is the main tool to create, visualize and edit profiles (regardless of if you have a physical device connected to your computer or not).

1. In the main window of XCTU, select the **Profile Editor** tool in the tools menu.

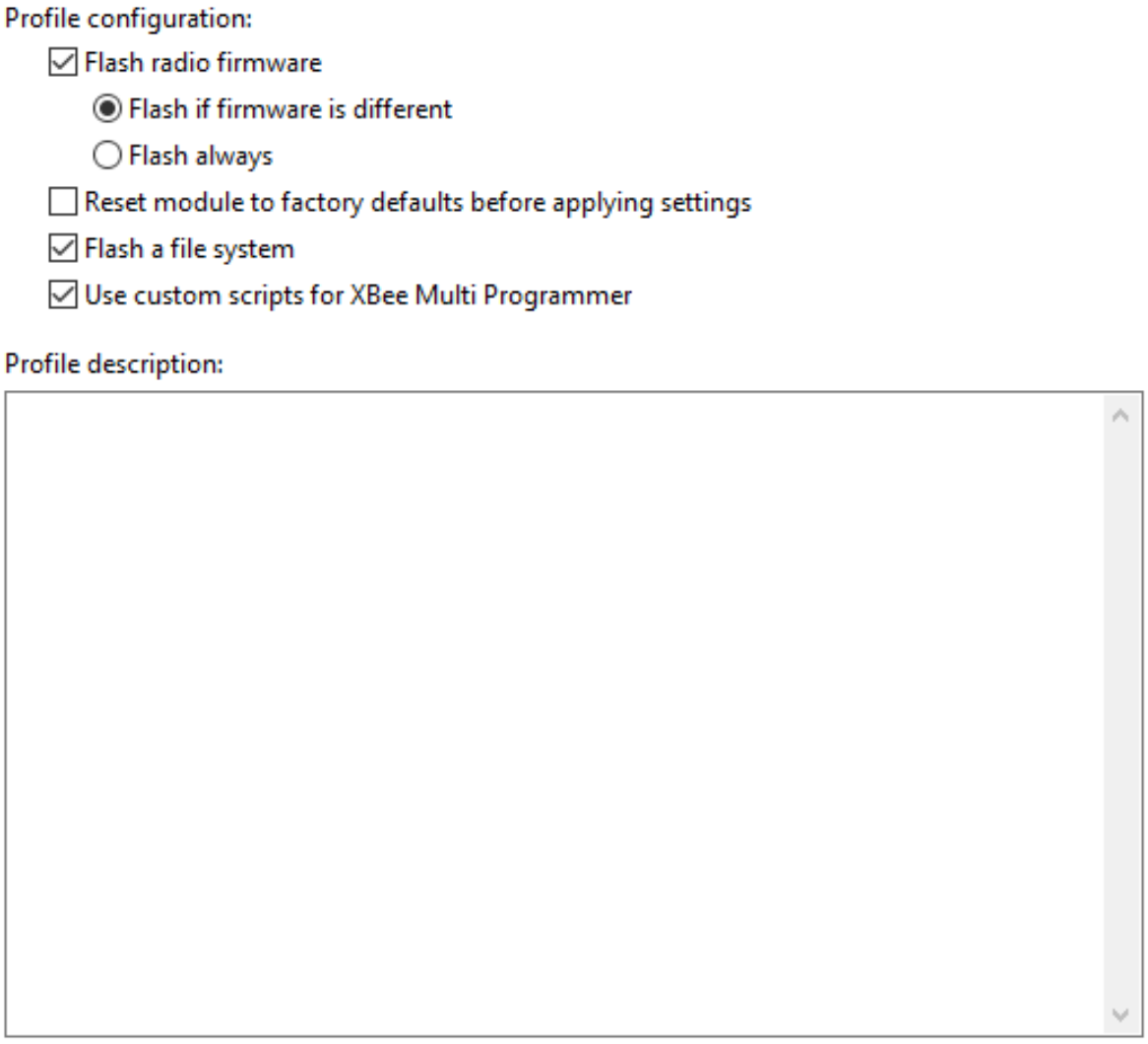


2. Click the **Create** button.

Step 2: Configure the profile

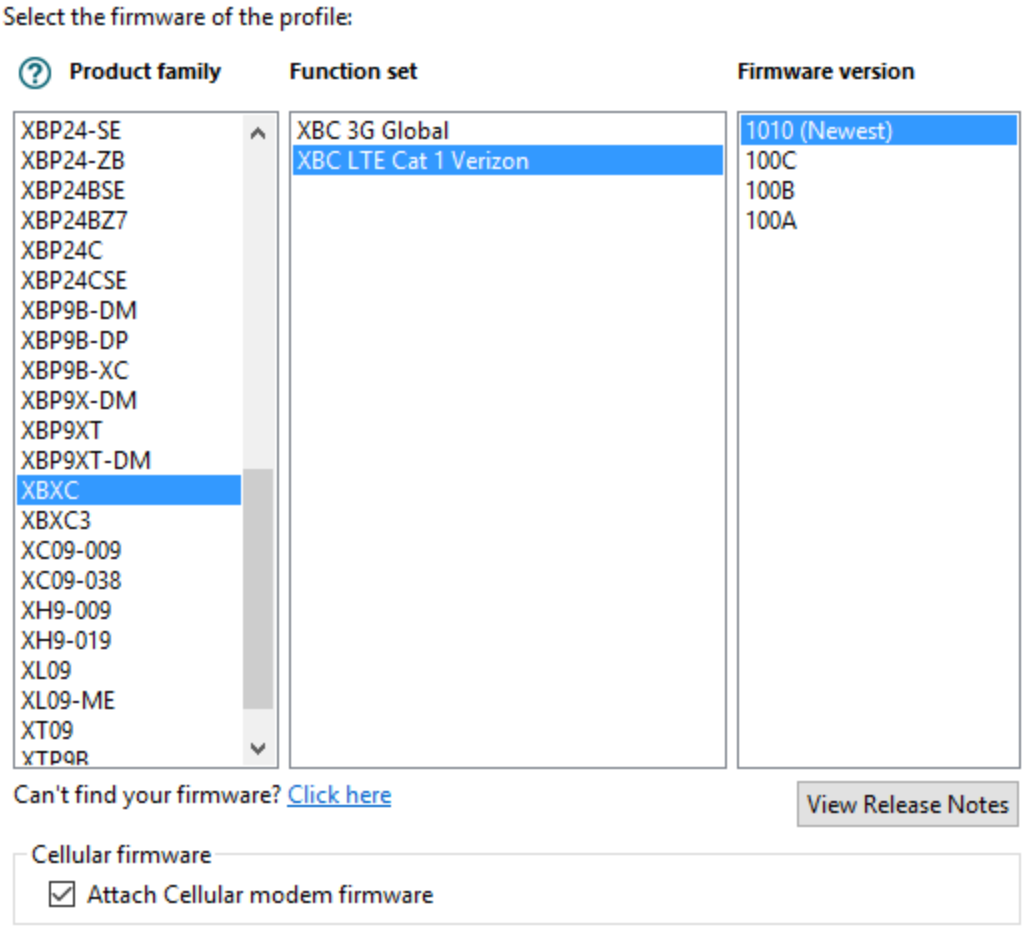
In this step you have to create a representative model upon which to base the configuration profile by specifying the general configuration, firmware version and settings. A wizard guides you through the different steps in order to create the profile.

1. Specify the general profile configuration:

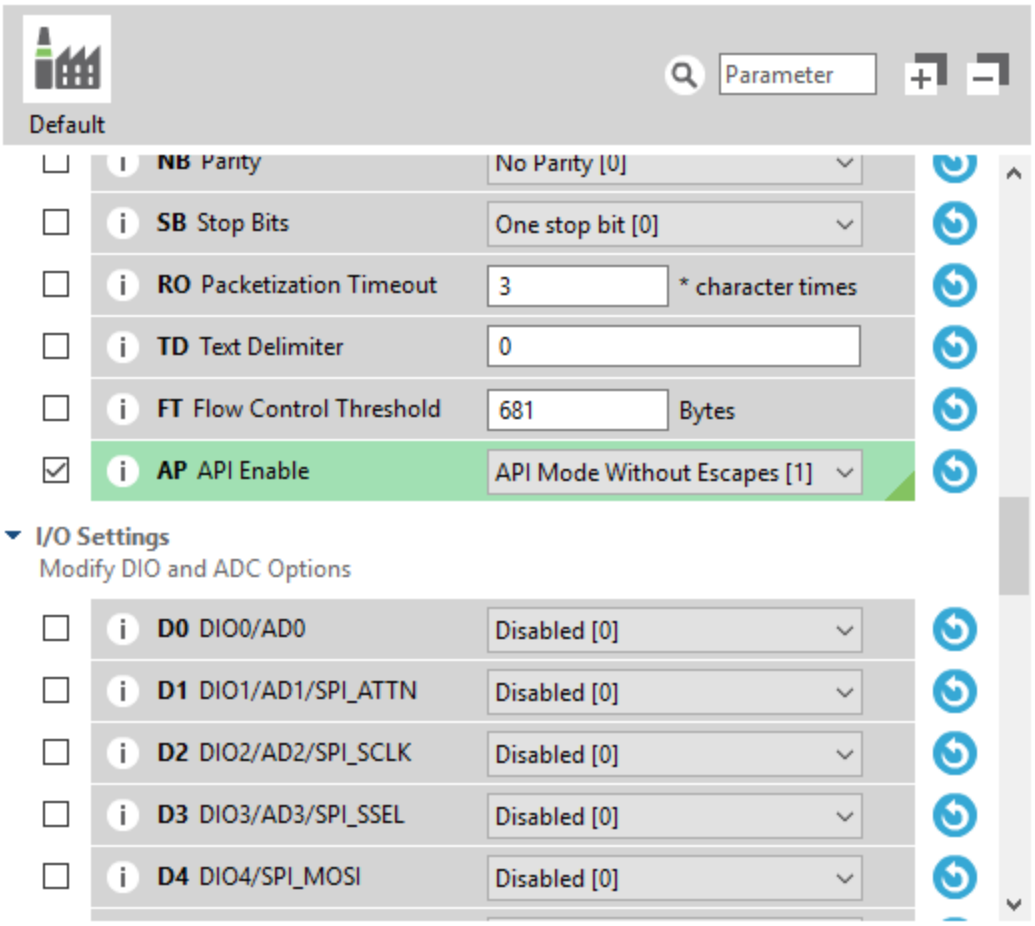


- **Flash radio firmware**. Check this option if you want the profile to flash the radio firmware. If you do so, you have also to specify the flash policy:
  - **Flash if firmware is different**. If the target device has the same firmware version as the one selected in the profile, firmware is not flashed. If the target device does not have the same firmware version as the one selected in the profile, the firmware is flashed into the device.
  - **Flash always**. The firmware image is always programmed into the device.
- **Reset module to factory defaults before applying settings**. Check this option if you want to reset the XBee device settings to their default values prior to applying the profile settings.
- **Flash a file system**. Check this option if you want to include a file system in the profile.
- **Use custom scripts for** XBee Multi Programmer. Check this option to include pre and post scripts to be run by the XBee Multi Programmer application when flashing an XBee device.
- **Profile description** (optional). Include a short description of the profile to be generated.

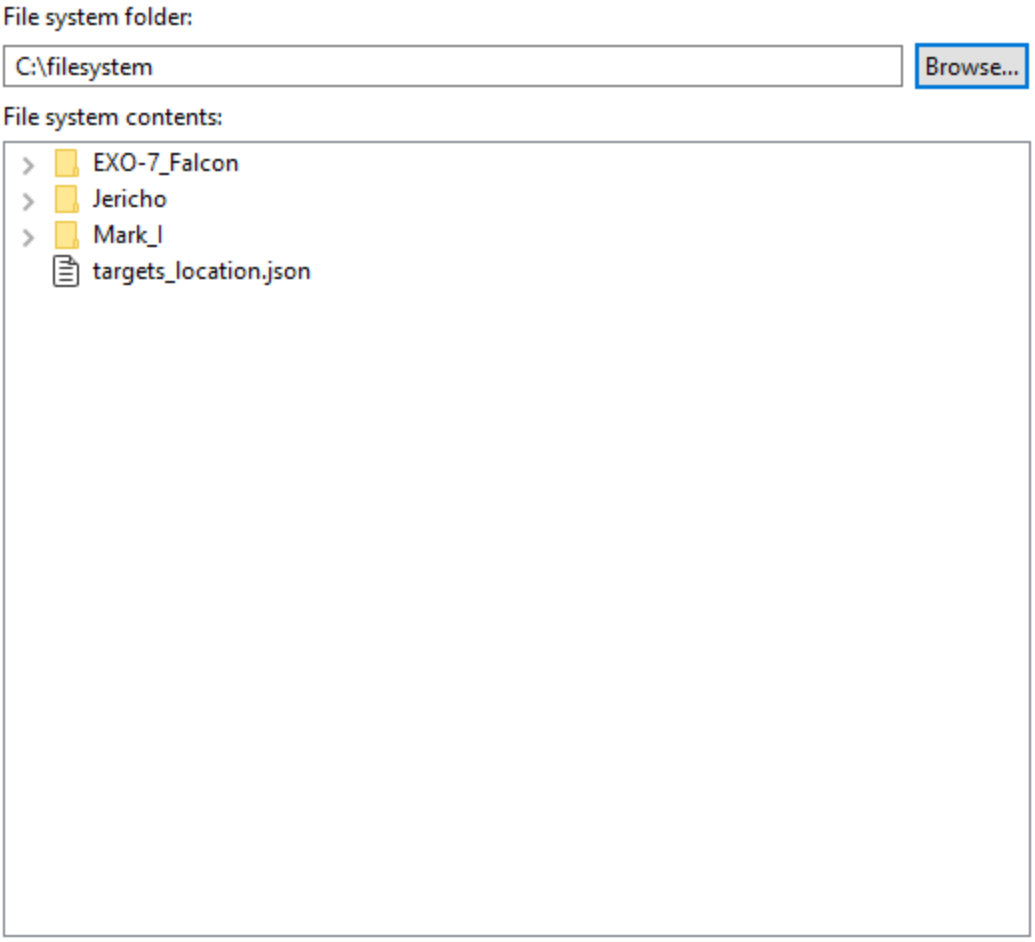
2. Select the firmware of the profile: **Product family**, **Function set** and **Firmware version**. If the selected firmware version is for an XBee Cellular device, you have the option to attach the cellular modem firmware too.



3. Choose and configure the firmware settings (optional). If you want to add any specific setting to the profile, select it and specify its value.



4. Attach a file system (optional). If you checked the **Flash a file system** option in the first page of the wizard, you have to select the folder containing the file system to be flashed with the profile. Once you have done that, you can visualize its structure in the following panel.



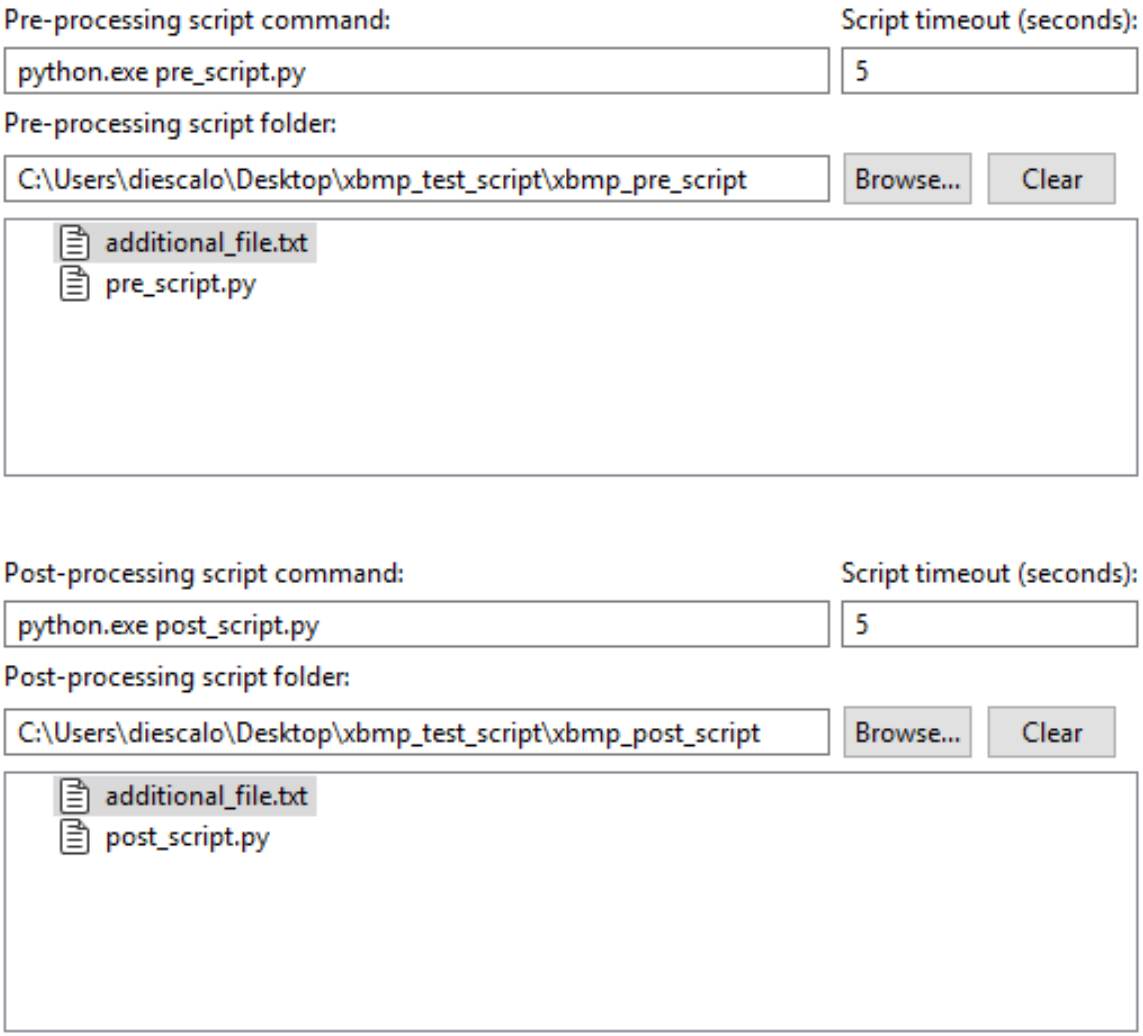
5. Specify Multi Programmer scripts (optional). If you checked the **Use custom scripts for XBee Multi Programmer** option in the first page of the wizard, you can specify the pre and post scripts to be executed by the XBee Multi Programmer application when programming an XBee device. Each script is defined by the following elements:
   - **Script command**. Process executed by the XBee Multi Programmer application.
   - **Script timeout**. Maximum time the XBee Multi Programmer application waits for the script to finish before considering a timeout error.
   - **Script folder (optional)**. Folder containing the script file or files. This is the execution path of the script.

   The XBee Multi Programmer application provides the following parameters to the scripts when they are run, so you need to consider them when writing the script:
   - **--portID**. The FTD2XX identifier of the serial port that the XBee that has been programmed is attached to.
   - **--portName**. The name of the serial port (COMX) that the XBee that has been programmed is attached to.
   - **--portBaudrate**. The baudrate of the serial port that the XBee that has been programmed is attached to.

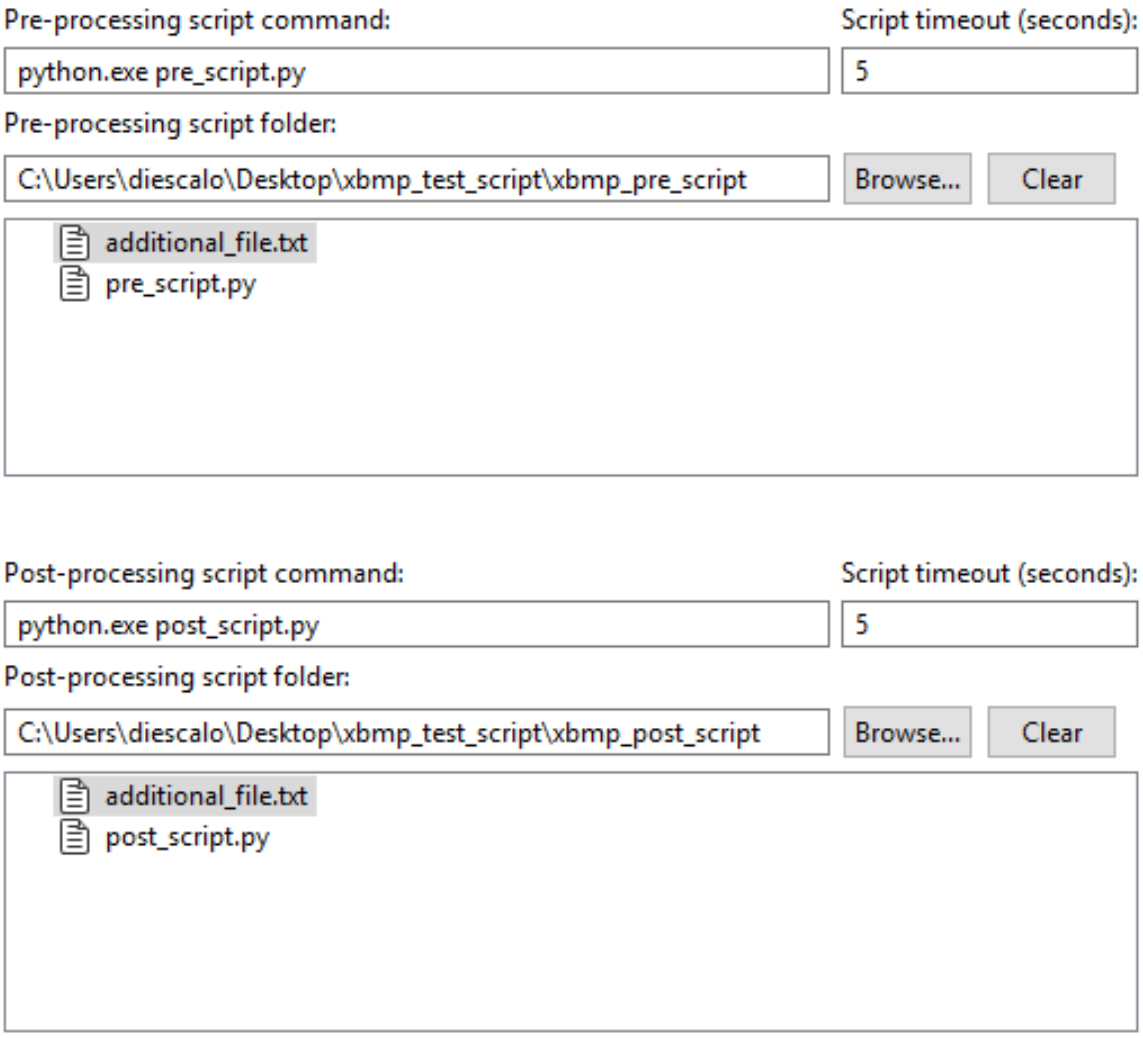
- **--boardIndex**. The index of the board within the XBee Multi Programmer application containing the slot that the XBee that has been programmed is attached to.
- **--slotIndex**. The XBee slot index within the XBee Multi Programmer board that the XBee that has been programmed is attached to.
- **--xbeeAddress**. The XBee 64-bit or IMEI address of the XBee that has been programmed.

Note If the script is a pre-script instead of a post-script, XBee Multi Programmer will not provide the **--portBaudrate** or **--xbeeAddress** parameters when running it because they are still unknown for the application.

CAUTION! If your script communicates with an XBee device over the serial port, you must implement a retry system to ensure a more robust connection with the target device. The FTDI D2XX library used by XBee Multi Programmer locks the access to all ports when opening any of them. This could cause the port you are trying to access to be in use when your script tries to open the connection.



6. Specify Multi Programmer scripts (optional). If you checked the **Use custom scripts for XBee Multi Programmer** option in the first page of the wizard, you can specify the pre and post scripts to be run by the XBee Multi Programmer application when programming an XBee device. Each script is defined by the following elements:
    - **Script command**. Process run by the XBee Multi Programmer application.
    - **Script timeout**. Maximum time the XBee Multi Programmer application waits for the script to finish before considering a timeout error.
    - **Script folder (optional)**. Folder containing the script file or files. This is the execution path of the script.



7. Once you are finished, click **Create profile**. A **Save file** dialog box appears.
8. Choose a name and path and click **Save**.

How to use a custom script to update the name of XBee devices individually

One of the latest features added to XBee Multi Programmer application is the ability to use custom scripts before and/or after programming XBee devices to configure custom settings or perform any operation with them individually.

In this how-to article you will learn how to use a custom script to change the name of an XBee device after it has been programmed by the application. This is very useful because the XBee Multi Programmer applies the same profile to all the XBee devices being programmed in a session, meaning that all of the programmed devices will have the same parameters configured, like the device name.

The post-script changes the name of the XBee devices as they are programmed with a fixed text (XBEE_DEVICE_) plus an index obtained from a counter file. The value of the index is increased and saved in the counter file every time an XBee device is programmed.

Step 1: Create the post-script

First you need to develop the post-script that will be automatically run just after the XBee Multi Programmer application programs an XBee device. You can write the script in any programming language, but you need to make sure the computer running it has the necessary resources installed and available—Python interpreter, Java machine, libraries, and so forth.

The script explained in the section uses Python 3 as programming language plus the Digi XBee Python library to cover communicating with the XBee device.

Follow these steps to create the post-script file:

1. Create a folder in your computer named **xbmp_custom_script**.
2. Create a file named **counter.txt** inside the **xbmp_custom_script** folder.
3. Edit the **counter.txt** file and write the following text:

```
0
```

4. Create a file named **change_name.py** inside the **xbmp_custom_script** folder.
5. Edit the content of the **change_name.py** file and paste the following code:

change_name.py

```
# Copyright 2019, Digi International Inc.
#
# Permission to use, copy, modify, and/or distribute this software for any
# purpose with or without fee is hereby granted, provided that the above
# copyright notice and this permission notice appear in all copies.
#
# THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
# WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
# MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
# ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
# WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
# ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
# OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

import argparse
import os
import sys
import time

from digi.xbee.devices import XBeeDevice
from serial import SerialException

FILE_COUNTER = "counter.txt"
FILE_LOCK = ".lock"

MODULE_NAME = "XBEE_DEVICE_%s"

LOCK_TIMEOUT_MS = 3000
```

```
def get_name_index():
    """
    Returns the name index from the counter file and updates it with the new
    one. Use a lock file to avoid concurrent accesses when modifying the
    counter file.

    Returns:
        Integer: The name index to be used for the Node Identifier.
    """
    # Wait until the lock file is released.
    deadline = time.time() + LOCK_TIMEOUT_MS
    while os.path.isfile(FILE_LOCK) and time.time() < deadline:
        time.sleep(0.1)

    # Lock file was not released, exit with error.
    if os.path.isfile(FILE_LOCK):
        sys.exit("Lock file was not released.")

    # Acquire the lock file.
    try:
        with open(FILE_LOCK, "w+"):
            with open(FILE_COUNTER, "r+") as f:
                # Read the index from the file.
                read_data = f.read()
                index = int(read_data)

                # Update the file with the next index.
                f.seek(0)
                f.write(str(index + 1))
                f.truncate()
    finally:
        # Release (remove) the lock file.
        if os.path.isfile(FILE_LOCK):
            os.remove(FILE_LOCK)

    return index


def main():
    """
    Main execution of the script. Updates the name of the XBee device located
    in the port specified in the arguments with a constant name + an index read
    from an external file. The index is incremented by one when the name
    """
    # Get all the arguments from XBee Multi Programmer.
    parser = argparse.ArgumentParser()
    parser.add_argument("--portID", required=False)
    parser.add_argument("--portName", required=False)
    parser.add_argument("--portBaudrate", required=False, type=int)
    parser.add_argument("--boardIndex", required=False, type=int)
    parser.add_argument("--slotIndex", required=False, type=int)
    parser.add_argument("--xbeeAddress", required=False)
    args = parser.parse_args()

    # Instantiate the XBee device.
    device = XBeeDevice(args.portName, args.portBaudrate)

    try:
        # Open the device connection.
```

```
        # Implement a retry system to ensure a more robust connection with the
target
        # module. The FTDI D2XX library, used by XBee Multi Programmer, locks
the access
        # to all ports when opening any of them. This could cause the port you
are trying
        # to access to be in use when your script tries to open the connection.
        retries = 10
        while not device.is_open() and retries > 0:
            try:
                device.open()
            except SerialException:
                time.sleep(0.1)
                retries -= 1

        # Ensure the device is open.
        if not device.is_open():
            sys.exit("Could not open the device.")

        # Update the XBee device name.
        new_name = MODULE_NAME % get_name_index()
        device.set_node_id(new_name)

        # Save the XBee device name.
        device.write_changes()
    finally:
        # Close the connection with the device.
        if device is not None and device.is_open():
            device.close()


if __name__ == '__main__':
    main()
```

Notice that the script is expecting the following arguments which are provided by the XBee Multi Programmer application when running it:

- **--portID**. The FTD2XX identifier of the serial port that the XBee that has been programmed is attached to.
- **--portName**. The name of the serial port (COMX) that the XBee that has been programmed is attached to.
- **--portBaudrate**. The baud rate of the serial port that the XBee that has been programmed is attached to.
- **--boardIndex**. The index of the board within the XBee Multi Programmer application containing the slot that the XBee that has been programmed is attached to.
- **--slotIndex**. The XBee slot index within the XBee Multi Programmer board that the XBee that has been programmed is attached to.
- **--xbeeAddress**. The XBee 64-bit or IMEI address of the XBee that has been programmed.

Note If the script is a pre-script instead of a post-script, XBee Multi Programmer will not provide the **--portBaudrate** or **--xbeeAddress** parameters when running it because they are still unknown for the application.

CAUTION! If your script communicates with an XBee device over the serial port, you must implement a retry system to ensure a more robust connection with the target device. The FTDI D2XX library used by XBee Multi Programmer locks the access to all ports when opening any of them. This could cause the port you are trying to access to be in use when your script tries to open the connection.

With the parameters given by the XBee Multi Programmer application, the script can open communication with the XBee device and change its name based on the index number contained in the **counter.txt** file.

6. Save the **change_name.py** file.

Step 2: Create the configuration profile

The next step is to create the configuration profile with the post-script attached so that XBee Multi Programmer can execute it. There is another tutorial that explains How to create a profile using XCTU, but you can follow these simplified steps if you prefer:

1. Open **XCTU**.
2. In the main window, select the **Profile Editor** tool from the tools menu. The **Profile Editor** tool displays.
3. Click the **Create** button from the **Profile Editor** toolbar. The Create a profile wizard displays.
4. On the first page of the wizard you need to specify the general profile configuration. Select your desired settings but make sure you check the **Use custom scripts for XBee Multi Programmer** option. This enables a final page in the wizard to configure the pre and post-scripts. Click **Next** when you are done.
5. On the second page of the wizard you must select the firmware version of the profile. You can do so specifying the **Product family**, **Function** set and **Firmware version** options. Once you have selected your desired firmware, click **Next** to continue.
6. On the next page you can configure the value of the firmware settings that you want to apply to your XBee devices. Click **Next** when you are ready.
7. If you did not check the **Flash a file system** option on the first page of the wizard, skip to step 8. Otherwise, the **Attach a file system** page displays and you are forced to configure a file system for the profile. When you are done, click **Next**.
8. The **Attach your custom pre/post scripts** page and this is where you must to specify the post-script that XBee Multi Programmer will run. Leave all the pre-script options empty and configure the post-script ones:

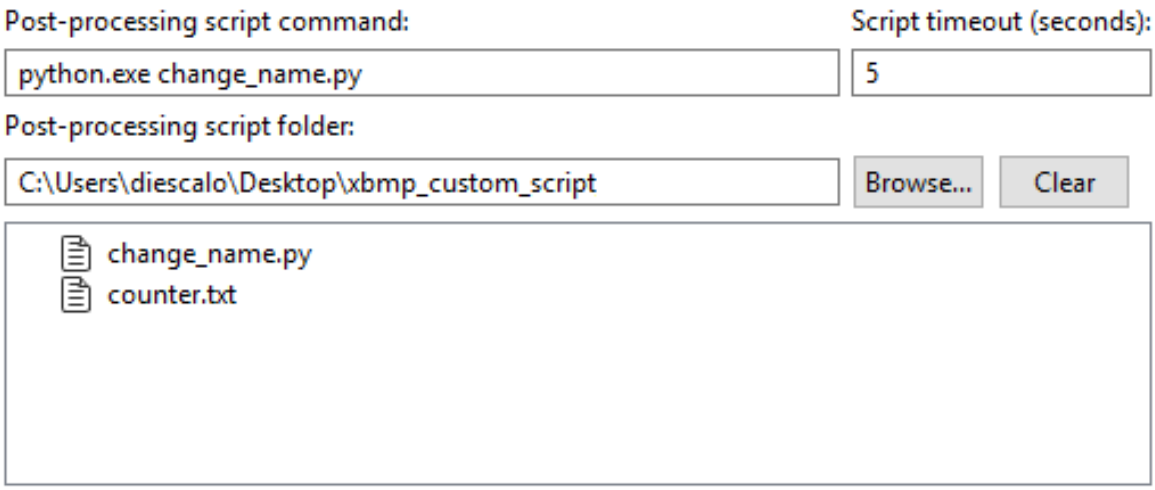
a. **Post-processing script command**. Fill it with the command to be run by the XBee Multi Programmer application:

```
python.exe change_name.py
```

Note Notice that you must add **Python 3** to the Path environment variable of the computer where you run the XBee Multi Programmer application, otherwise **python.exe** will not be found and the script execution will fail.

b. **Script timeout (seconds)**. This is the maximum time XBee Multi Programmer waits for the script to finalize running before considering it as failed. It is required, so fill it with **5 seconds** for example—the communication with the device should not take longer than 1 second.

c. **Post-processing script folder**. Here you need to specify the folder within your computer where the post-script is located at. All the files contained in the folder are attached to the configuration profile and XBee Multi Programmer is able to access them as that folder will be the execution path of the script. Configure this option with the **xbmp_custom_script** folder you created previously. When you have configured it, you can see its structure in the panel below.

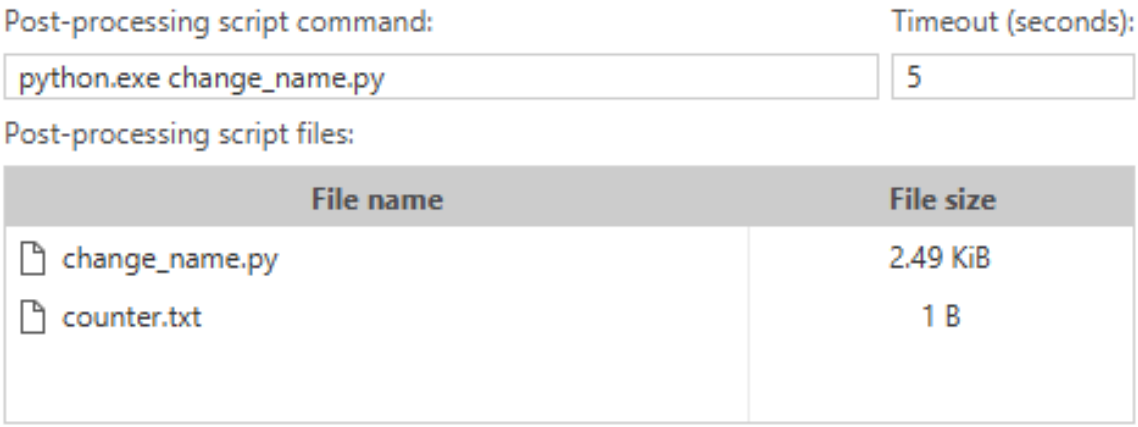


9. Click **Create profile** to create your configuration profile specifying the destination file when prompted. Name it **custom_script_profile.xpro** for example.

Step 3: Test the post-script

The final step is to test that the post-script attached to the configuration profile works properly. To do so follow these steps:

1. Attach the XBee Multi Programmer board to your computer.
2. Open the XBee Multi Programmer application.
3. Select the **File > Open profile** option from the main menu and browse the **custom_script_profile.xpro** you created previously to load it in the application.
4. Select the **File > View profile Information** option from the main menu to open the **Profile viewer dialog** that displays the configuration of the profile. Select the **Scripts** tab and verify that the post-script configuration is correct.



5. Close the **Profile viewer** dialog.
6. Attach six XBee devices to the XBee Multi Programmer board.
7. Click the **Start Session** button from the toolbar to start the programming session.
8. A programming task is run for each device attached to the board. After flashing the firmware and configuring the settings, the **Details** column of the history table should display the **Executing post-script...** message, indicating that the post-script is being executed for that device. After that, the process should complete successfully.
9. Once all the devices have been programmed, click the **Finish session** button from the toolbar to end the programming session.
10. Open **XCTU**.
11. Attach each device to your computer through a development board and add them to XCTU.
12. Verify that the names of the devices go from **XBEE_DEVICE_0** to **XBEE_DEVICE_5**.

Known issues

The XBee Multi Programmer application currently has the following known issues and limitations:

- It is not possible to update **XBee Cellular LTE Cat 1** modules to firmware versions lower than **100A**.
- If you use pre/post scripts to communicate with an XBee device over the serial port, you must implement a retry system to ensure a more robust connection with the target device. The FTDI D2XX library used by XBee Multi Programmer locks the access to all ports when opening any of them. This could cause the port you are trying to access to be in use when your script tries to open the connection.

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

DIGI:

XBEE-MP-MCRO XBEE-MP-SMT-PCB XBEE-MP-TH