Getting Started Guide – XBee Cellular

Overview

Digi XBee Global LTE modules offers the easiest way to integrate cellular connectivity into an OEM device. With the introduction of modern standards like LTE Cat 1, Cat 4, LTE-M and NB-IoT, Digi has the cellular modem for your design. Digi XBee Cellular modems provide easy cellular connectivity without having to go through a costly FCC or carrier end-device certification process.

Digi XBee is more than a module. It's a complete ecosystem of radio frequency modems, gateways, adapters, and software, all engineered to accelerate wireless development for global deployments. One socket allows you to connect to IoT networks around the globe. With the authentic Digi XBee footprint, you can future-proof your design and know that Digi has you covered for new technologies as they emerge.

Why You'll Love It

- FCC certified and carrier end-device certified
- Excellent coverage and building penetration
- LTE-M or NB-IoT protocol
- MicroPython: on-module programmability to add local intelligence
- Available with Digi-provided SIM cards and data plans
- Bluetooth Low Energy
- GNSS/GPS
- Digital and analog I/O, i2c, SPI, UART
- Extended battery life with PSM, deep sleep, pin sleep and cyclic sleep modes
- Digi TrustFence security
- Remotely manage and configure with XBee Studio or Digi Remote Manager

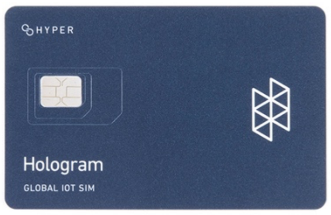
Getting Started

The fastest learning happens hands-on. This guide will help you set up your hardware, connect to the mobile data LTE network, create and run XBee MicroPython applications, use Bluetooth services, and leverage remote operations with Digi Remote Manager. Let's get started!

Identify the kit contents

This evaluation and development kit includes the following:

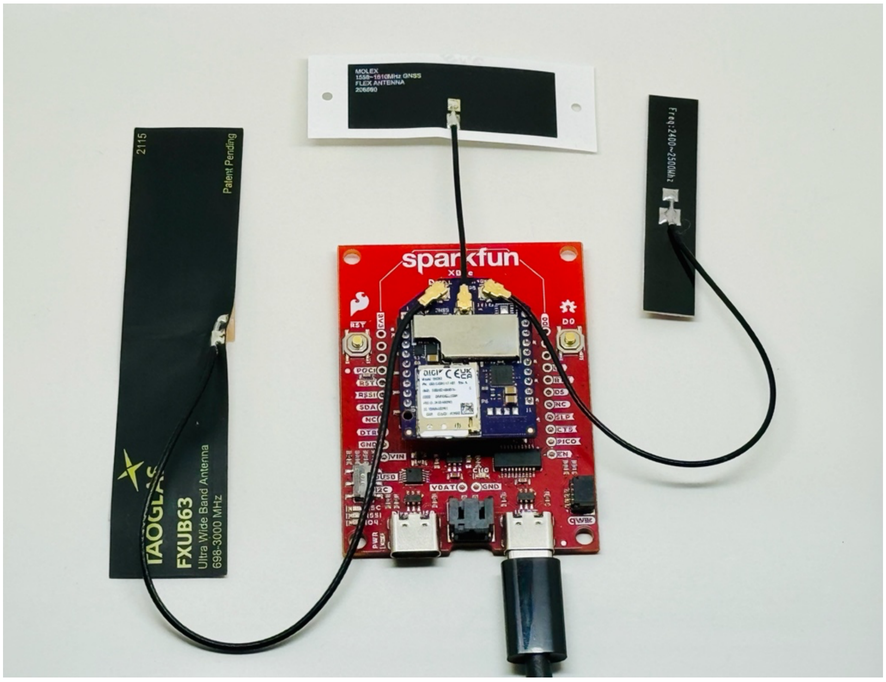
Item	Description
One XBee 3 Cellular LTE-M/NB-IoT module Includes LTE, BLE, MicroPython, GNSS, security/encryption chip, SPI, UART, GPIOs, ADCs, secure remote access to all features.	
One SparkFun XBee Development Board Provides USB-C connectivity, battery power, battery charge controller, user button switch, user LED, firmware programming interface, access to all pinouts.	
One cellular antenna with a U.FL connector Receives LTE signals worldwide	
One Bluetooth Low Energy (BLE) antenna Receives BLE signals from smartphones, beacons or sensors	

Item	Description
One GNSS / GPS antenna Receives GNSS location satellite data outdoors.	
One SparkFun Temperature/Humidity Sensor Breakout - SHTC3 (Qwiic) Delivers temperature and humidity information via a standardized I2C interface	
One SparkFun Qwiic Cable - 50mm Connects Qwiic sensors to the development board	
One USB-C cable Powers the development board and provides serial UART access	
One SIM card Supports LTE-M connectivity, a low-bandwidth standard for cellular data communications. (Module also supports NB-IoT standard when used with third-party SIM cards.)	

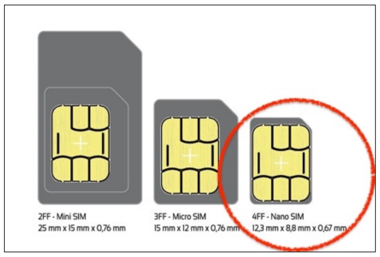
SIM card

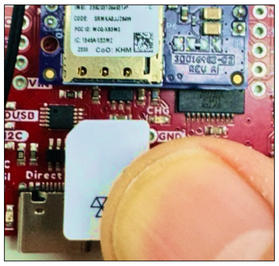
A SIM card is included in your kit that supports the LTE-M standard. If you would like to acquire an additional or different type SIM, contact Digi Sales at www.digi.com/contactus and ask about Cellular Bundled Service plans. You can also purchase a subscription for cell service or a Digi Remote Manager package at shop.digi.com, or when purchasing new XBee Cellular modules, choosing the option to have a SIM included (SKUs ending -101 or -102).

Connect the hardware



1. Insert the XBee 3 Global Cellular LTE-M/NB-IoT into the sockets provided, with the antenna connectors at the top as shown. Be sure that all 20 pins are properly inserted. It's easy to be one pin off, so check carefully for alignment. For more information about this development board, see SparkFun's product page.
2. A SIM card is included with your kit. If your SIM card is not inserted yet, install the SIM card into its slot on the lower left of the XBee, with the metal part facing down and the cut corner toward the upper left. XBees use the smallest "Nano" or 4FF size of SIM card.





3. Connect the antennas.
   a. Connect the cellular antenna (FXUB63) to the upper left **CELL** socket. Align the U.FL connector carefully, then firmly press straight down to seat the connector.

   You should hear a click when the antenna attaches correctly. This may take a few tries, especially if it's your first time using this type of connector—be patient, you'll get there!

   b. Connect the BLE antenna to the upper right **2.4 GHz** U.FL socket. You must do this any time you are using BLE functionality, which you will in the following exercises. Otherwise the BLE antenna is optional.
   c. Connect the GNSS antenna to its **GNSS** U.FL socket to use location services. The GNSS antenna will not be used in these initial exercises and is optional.

4. Connect the USB-C cable from a computer to the right-hand USB port on the development board. Windows OS computers may search for a driver, which can take a few minutes to install.

Activate SIM Card

If your kit came with a Hologram SIM card, you can activate it yourself using the steps below. If you have a different SIM, use the activation instructions that came with it, being sure to note the APN used by that carrier. For Hologram SIMs you can use their official instructions to activate the Hologram SIM. These steps are summarized below:

1. Set up an account and then sign in to the Hologram Dashboard.
2. Click the button to **Activate a Hologram SIM**
3. Locate the Hologram SIM's **ICCID number**. It is printed on the plastic card that your SIM came on, and also on that SIM card itself. ICCIDs are 19 or 20 digits.
4. Enter the **ICCID** for your SIM when prompted.
5. Select your **coverage area** from the list provided
6. Select a **data plan** from the list provided. (Pay as you go is a good default choice.)
7. Set a **data usage** limit. This limit is optional. (If you do want a limit we suggest at least 200 MB.)
8. Set up your **auto-refill** preferences. Auto-refill is also optional.
9. Review your selections and click **Pay and Activate** to complete the setup
10. Note the APN, which is usually: `hologram`

Install and update XBee Studio

XBee Studio is a desktop application for configuring Digi XBee Cellular radio modules. It's available for Windows, MacOS and Linux. The program includes built-in tools to make it easy to set up, configure, program and test cellular XBees. For full information on using XBee Studio, see the XBee Studio user guide.

You will use XBee Studio to complete the getting started process.

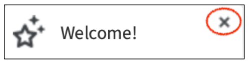
1. Download the version for your operating system and run the installer.
2. You can skip the USB drivers for now to save time and, if desired, update them later using these instructions.

3. The software may ask to update itself to the current version. If so, allow that to happen before continuing.

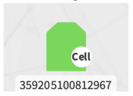
Add device to XBee Studio

These instructions show you how to add an XBee 3 Cellular to XBee Studio.

1. Launch XBee Studio.
2. A welcome screen will be displayed. You can uncheck the box for "Always show Welcome screen at startup" and then close that tab using the small **x** displayed.



3. Make sure that your XBee on its development board is plugged into one of the USB ports on your computer.
4. Click the **Home** icon at the top of the left-side navigation menu.
5. In the middle of the screen you may see a message "Looking for devices…"
   Once your module is successfully discovered, it will be displayed with its **IMEI** number (*The IMEI identifies the XBee hardware itself, and is different from the ICCID that identifies the SIM.*)

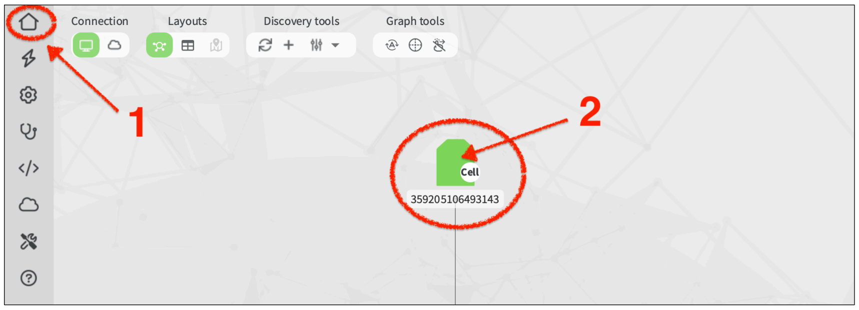


If there is a problem with discovery, you'll see a message "No devices found." In this case, check that you have the proper drivers, that your XBee development board is connected to a USB port on your computer, and that the power light on the development board is lit, with the association indicator light marked **ASC** either steady or blinking, indicating that the XBee is properly seated and receiving power. Additional information on discovery can be found in the user guide.

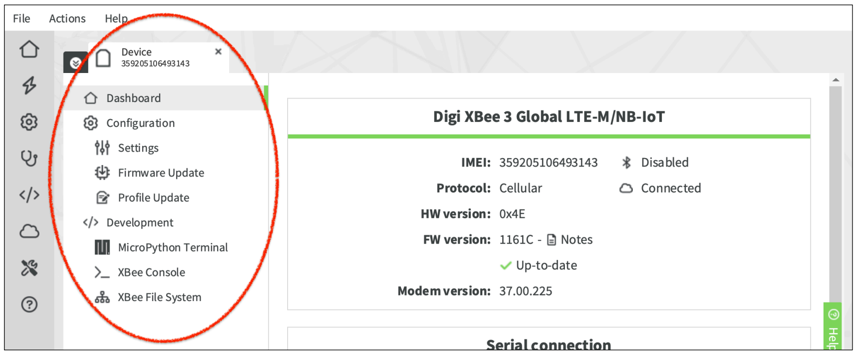
Accessing the device menu

Each device added to XBee Studio has its own menu. Throughout this guide you will need to access your device menu. Here's how to do that.

1. Click the **Home** icon at the top of the left-side navigation menu.
2. Next, click on your device icon to select it.



3. The **Device** menu will be displayed.



Digi XBee 3 Global LTE-M/NB-IoT

IMEI: 359205106493143 Disabled
Protocol: Cellular Connected
HW version: 0x4E
FW version: 1161C - Notes
Up-to-date
Modem version: 37.00.225

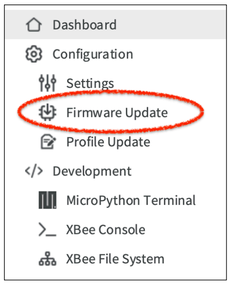
Serial connection

The Device menu items operate only on the selected device, as identified by the IMEA number in its top tab. Note that the navigation menu to the very left has several similar icons but our exercises will focus on the Device menu items indicated above.

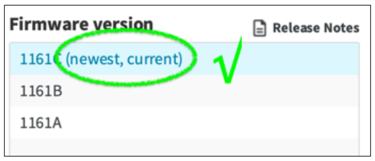
Update device and cellular firmware

You should use XBee Studio to update the device firmware on your XBee 3 to the most recent version. This ensures that you can take advantage of all the latest fixes and features. XBee Studio can update the device firmware, and if needed, also will attempt to update your cellular modem firmware.

1. From the Device menu select **Firmware Update**.

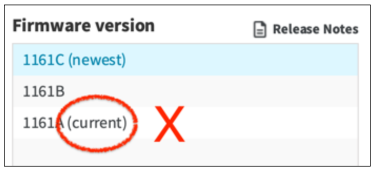


2. If the version marked **newest** is *also* marked **current** then you are done!



1161C (newest, current)
1161B
1161A

Otherwise, if the **current** firmware is NOT the **newest**…



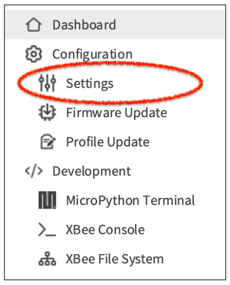
1161C (newest)
1161B
1161A (current)

…click the **Update device** button at the bottom to begin the update process. There are two sets of firmware, one for the overall XBee and another for the underlying cellular component. The instructions will guide you through updating both. If an update for the cellular component is needed you will be guided through downloading additional drivers and connecting to the USB direct port on your development board for that portion of the update.

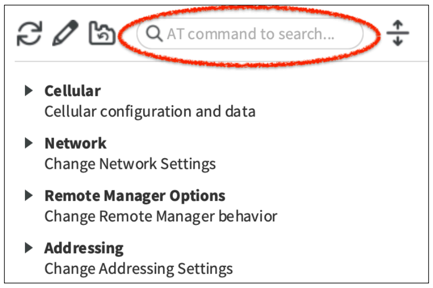
Set APN, protocol, carrier profile (AN, N#, CP)

To connect to the LTE-M network, you will need to configure a few settings on your XBee Cellular. Note that while NB-IoT is fully supported by the module, the steps outlined in this Getting Started Guide are specifically designed and optimized for LTE-M.

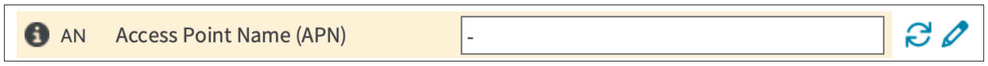
1. From the Device menu select **Settings**. The Device Settings interface will be shown.



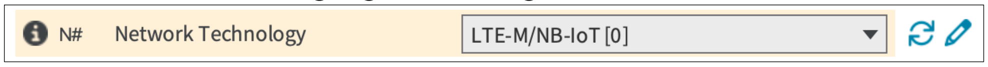
2. Each setting is identified with a two-letter code, also known as an *AT command*. You can search for particular settings using these codes, it's the easiest way to find them in the list.



3. Click in the **AT command to search…** field at the top and type **AN** to highlight the Access Point Name (AN) setting. By default there is just a hyphen shown here.


AN Access Point Name (APN) -

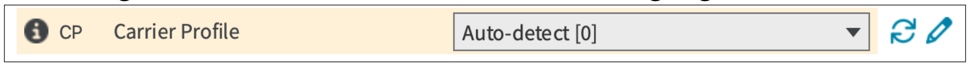
4. Enter the APN value that your SIM's carrier requires. For example if your carrier is Hologram, you can use the APN found on their web site: "**hologram**". Carriers often have more than one APN, and many offer private ones so be sure to get the correct one for your SIM. Finally, click the pencil icon to write that value to the XBee.
5. Next, search for **N#** to display the Network Technology setting. (You may need to scroll down a bit to see the highlighted setting.)


N# Network Technology LTE-M/NB-IoT [0]

6. In the **Network Technology** (N#) note that the **LTE-M/NB-IoT [0]** item is selected. This means that the network protocol will be automatically selected. This is fine in most situations. However if you have difficulty connecting you can later try changing this to

LTE-M Only [2] which will prevent the device from searching for other protocols. If you do this, remember to click the pencil icon to save that value to the XBee Cellular.

7. Search again, this time for CP. Scroll to see the highlighted command if needed.



CP Carrier Profile Auto-detect [0]

8. In the **Carrier Profile** (CP) field, note that the **Auto-detect [0]** item is selected. This means that the module will try to pick the best settings for your carrier. While this also generally works well, in some situations it may be easier to select either **No Profile [1]** or the profile for your specific carrier or country if listed. If you do this, remember to click the pencil icon to save that value to the XBee Cellular.

XBee Cellulars attempt to connect to the network automatically any time they have a SIM and valid settings. You will monitor the connection status in the next step.

---

*Note: There are lots of settings available. Luckily XBee Studio can create stored device profiles that include all settings, to make restoring a particular configuration or sharing it with a large number of devices easy. See the **Profile Editor** under the **</> Development** menu in the leftmost menu for that toolset.*

Check for cellular registration and connection (LED and AI command)

Initial registration can take 5 - 6 minutes the first time a device is connected to the network. Make sure you have a working SIM card, the correct APN, good cellular coverage and that the cellular antenna is connected to the correct (top left) U.FL connector on the device.

Development Board LED

A blinking **ASC** light on the development board (*marked Conn/Status on XBIB boards*) indicates the device is connected. If this light is on but *not* blinking (steady), the device is not yet connected to the LTE network.

XBee Studio

In the device's Settings, search for AI for the **Association Indication** value. Refresh it with the button to see the current value. It will be set to **0** to indicate a valid connection. Any other number shown here indicates that the device is not yet fully connected.

Troubleshooting

Still having problems getting connected? The Digi Support Forum is a great place to look for help. You can also contact Digi Support directly: tech.support@digi.com

XBee connection examples

These examples create some basic remote interactions with the XBee in "Transparent Mode" via its serial port, with a human sending the data. If you are interested in using a host device like an external microcontroller or computer to send information, see API Mode. Direct data acquisition using the intelligence built into the XBee will be covered in the Get Started with MicroPython section below.

Echo server

This basic example shows how to use XBee Cellular's Transparent Mode to confirm round-trip communications with a remote server that simply echoes back whatever text you type.

This exercise will require configuring a few more settings. It's good practice to do that manually, and full instructions are below. However, there's a second option: loading in a stored Profile that includes all the settings. Both options are described below, you can use either method.

Option 1: Manual Configuration

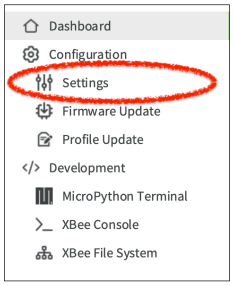
The following table explains the commands that you will use in this example.

Command code	Value	Description
IP (IP Protocol)	`1`	TCP: Set the expected transmission mode to TCP communications.
TD (Text Delimiter)	`D (0x0D)`	The text delimiter to be used for Transparent mode, as an ASCII hex code. No information is sent until this character is entered, unless the maximum number of characters has been reached. Set to **0** to disable text delimiter checking. Set to **D** for a carriage return.
DL (Destination Address)	`52.43.121.77`	The target IP address of the echo server. **Note** Some carriers may require whitelisted IP addresses. If this IP is not whitelisted by your carrier you will not be able to run this example.
DE (Destination Port)	`2329 (0x2329)`	TCP: The target port number of the TCP echo server. This port in decimal is 9001.

To communicate with the Echo server:

1. Ensure that the device is set up correctly with the SIM card installed and the antennas connected as described in Connect the Hardware (above).

2. Ensure XBee Studio is open and that your XBee Cellular has been discovered.
3. From the Device menu select **Settings**.



4. Click in the **AT command to search…** field at the top and type **IP** to highlight the **IP Protocol** setting. (*Remember that you may need to scroll to see it.*) Confirm that **TCP [1]** is selected.
6. To enable the XBee to recognize carriage return as a message delimiter, search for **TD** to highlight the **Text Delimiter** field, type **D** into it and click the **Write** button .
7. Search for AP to highlight **API Enable** and confirm it is set to **Transparent Mode [0]**.
8. To enter the destination address of the echo server, search for **DL** and in the **Destination Address** field, type **52.43.121.77** here and click the **Write** button .
9. To set the correct TCP/IP port number, search for DE to see the **Destination Port** field, type **2329** here and click the **Write** button .
10. With the settings entered, it's a good idea to click the **Write** button at the top of the Settings screen one last time to ensure everything got saved. If there are no more changes to be written it will let you know.

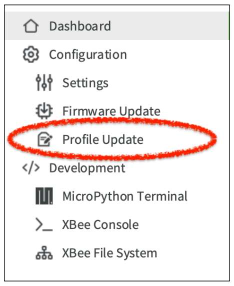
NOTE: XBee Studio uses only raw hexadecimal numbers. The leading 0x is not used in XBee Studio.

Option 2: Configure with Profile

Using stored profiles is a great way to completely configure an XBee with a single step. If you are configuring 100 or more of them, profiles will be absolutely indispensable. An XBee profile stores everything needed to configure a device including settings, files, and firmware requirements. They are fast to make and even faster to use.

1. Download the Echo Server profile to your computer.

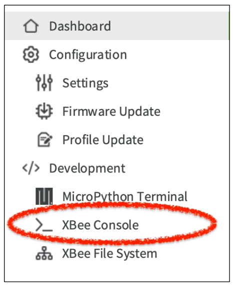
2. From the Device menu select **Profile Update**.



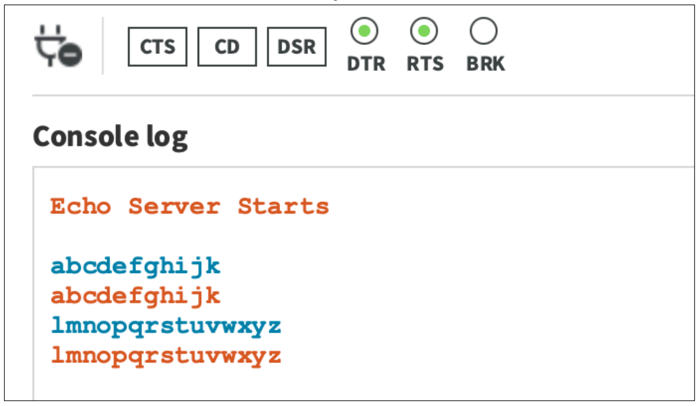
3. Click on **Load a Profile**.
4. Navigate to where you downloaded the **echo_server_profile.xpro** file onto your computer and open it.
5. The profile's details will be shown. Click on **Apply Profile** at the bottom to load it onto your XBee.

Echo Server Interaction

1. From the Device menu select **XBee Console**.



2. Click the **Open** button to open a serial connection to the device.
3. Click in the main console window to select it, then type in the Console to talk to the echo server, pressing Return or Enter after each message. The system typically responds first with "Echo Server Starts" and then sends back whatever you typed. The following screenshot provides an example of this chat, with the human's text in blue and the echo server's responses in red.



Console log

```
Echo Server Starts

abcdefghijk
abcdefghijk
lmnopqrstuvwxyz
lmnopqrstuvwxyz
```

Note: For full instructions on using the Console, see the XBee Console topic in the XBee Studio User Guide.

TCP/IP transparent mode

Now that you have a simple round-trip transaction working, let's extend this concept to any TCP/IP interaction. Here's how you can set up a connection to any IP address using TCP protocol. This exercise is optional. It is very similar to the echo server one above.

Option 1: Configure with Profile

The best learning comes from hands-on configuration. However if you're in a hurry or simply want to skip some typing, this profile can be loaded to communicate with Digi's example daytime server.

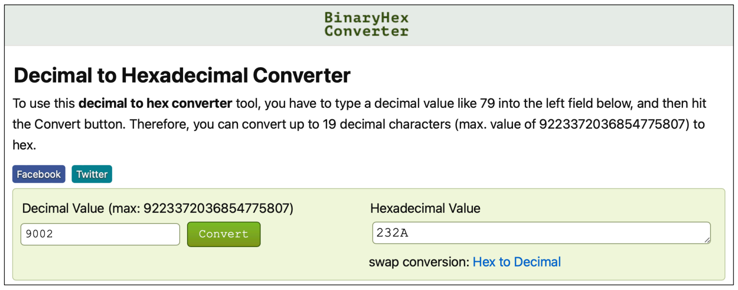
Option 2: Configure Manually

The following table explains the AT commands that you use in this example.

Command code	Value	Description
IP (IP Protocol)	`1`	TCP: Set the expected transmission mode to TCP communications.
TD (Text Delimiter)	`0`	The text delimiter to be used for Transparent mode, as an ASCII hex code. No information is sent until this character is entered, unless the maximum number of characters has been reached. Set to **0** to disable text delimiter checking. Set to **D** for a carriage return.
DL (Destination Address)	`<Target IP address>`	The target IP address of the server.
DE (Destination Port)	`<Target port number>`	The target port number of the TCP echo server. Remember to convert decimal numbers to hexadecimal (for example port 80 would be 0x50 in hex, and is entered as simply 50 in XBee Studio).

You can you any destination address and port you like, but something needs to accept the connection. Perhaps you've opened port **1234** on a server that has the address of **207.159.88.121**. Your DL would be **207.159.88.121** and your DE would be **4D2** (1234 in decimal is 4D2 in hex.)

If you don't have access to a server where you can open ports, Digi has provided one that you can connect to for testing. The server is at **52.43.121.77** with an open port **9002**. Remember to translate the port to hexadecimal before entering it. If you use the decimal 9002 it won't work! There's a converter at https://www.binaryhexconverter.com/decimal-to-hex-converter that can change the number to hex (**232A**) for you.


BinaryHex Converter

Decimal to Hexadecimal Converter

To use this **decimal to hex converter** tool, you have to type a decimal value like 79 into the left field below, and then hit the Convert button. Therefore, you can convert up to 19 decimal characters (max. value of 9223372036854775807) to hex.

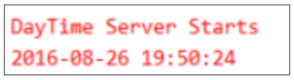
Decimal Value (max: 9223372036854775807): 9002

Hexadecimal Value: 232A

swap conversion: Hex to Decimal

DayTime Server

If you use the XBee Console to connect to the server above and send any character, you will get a response with the current UTC date and time.


```
DayTime Server Starts
2016-08-26 19:50:24
```

Software libraries (Java, Python, C) and API Mode

In the above examples we used "Transparent Mode" on the XBee's serial port, meaning that whatever is received on that port (your typed text) is passed directly to the destination IP address and port. This is a very simple interface for a human to use in getting to know the XBee, however most applications will benefit from "API Mode" which allows structured communications to and from the XBee in well-defined data frames. Complete information on API mode can be found in the User Guide's API section. Code libraries for sending and receiving information in API mode are available for Java, Python, and C. See Software Libraries for more information.

Get started with MicroPython

The Digi XBee 3 Cellular line features edge intelligence to help execute local tasks and a whole lot more. By running simple Python-based scripts on Digi XBee 3, you can actuate equipment, process sensor data, save money, extend battery life, improve responsiveness and enhance system reliability.

Why You'll Love MicroPython

- rapidly prototype intelligent behaviors at the edges of your IoT network
- open-source programming language based on Python 3
- wide variety of sensor and actuator libraries
- optimized to run on microcontrollers
- interactive prompt for easy learning
- no additional microcontroller required
- Cryptic sensor readings can be transformed into useful data
- excess transmissions can be intelligently filtered out
- modern sensors and actuators can be employed directly
- a developer with basic skills can create a useful application in 50 lines of code or less.

MicroPython enables customers to add intelligence to end products without the added time, cost, and complexity of designing in a separate host microcontroller. And even in cases where that host microcontroller is already present, the XBee's onboard intelligence can supplement its existing capabilities with logic that can be upgraded over the air.

Use cases

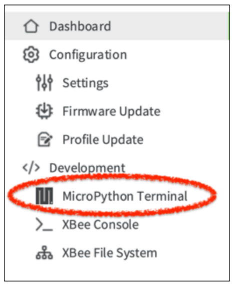
Even with simple environmental monitoring, the addition of edge computing logic can open all kinds of useful capabilities. We can read sensors and send their values. But we can also transform those readings to useful units (voltage levels to humidity values), select only the highest and lowest values to send, create "heartbeats" to ensure updates even when data is stable, aggregate readings and batch them, aggregate readings and average them, suppress duplicate data or exert local control of the sensor to retune it based on the environment. Serial communications from the host machine can be interpreted and locally adjusted, data can be requested in real time, GPIO pins controlled and sophisticated I2C sensors can be managed directly by the XBee. The only real limit is your imagination.

MicroPython on the XBee 3 Cellular LTE-M/NB-IoT

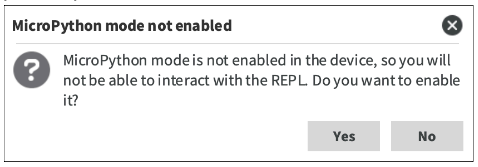
The XBee has MicroPython running on the device itself. You can access a MicroPython prompt from the XBee via XBee Studio, Digi Remote Manager, or any serial terminal program. In addition, the file system can hold MicroPython programs and modules that are deployed at runtime.

Use XBee Studio to enter the MicroPython environment

1. From the Device menu select **MicroPython Terminal**.



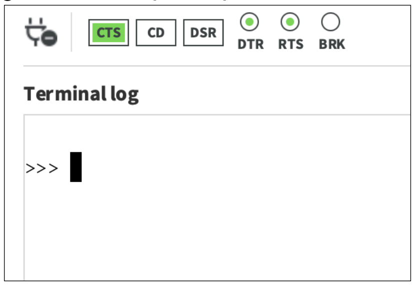
2. At the top of the Terminal screen, click the **Open connection** icon to connect. If MicroPython mode was not previously enabled on the serial interface you will be prompted to do that. Click **Yes** to enable it.



MicroPython mode not enabled

MicroPython mode is not enabled in the device, so you will not be able to interact with the REPL. Do you want to enable it?

3. Press return on the keyboard to see a "REPL" prompt **>>>** as shown below. (*In cases where a program was previously running, try pressing* ***Ctrl+C*** *to stop that program and get the* ***>>>*** *prompt.)*

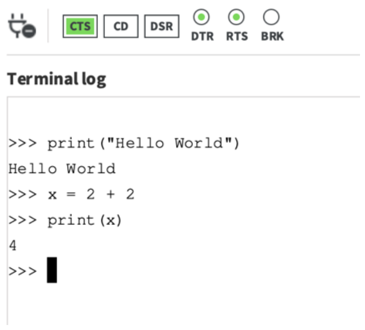


```
>>> 
```

"REPL" stands for read-evaluate-print-loop, a simple interactive prompt that takes single user inputs, executes them, and returns the result to the user. This command-line interface makes it very easy for beginners to experiment with a new language they are learning, and for experienced users to try out new ideas interactively.

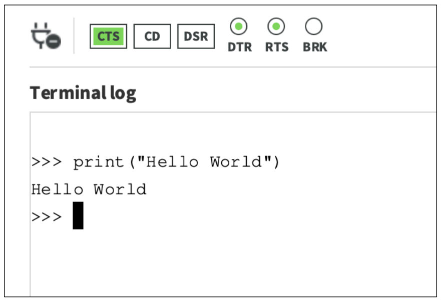
Example: Hello World

1. At the MicroPython **>>>** prompt, type the MicroPython command: **`print("Hello, World!")`**



```
>>> print("Hello World")
Hello World
>>> x = 2 + 2
>>> print(x)
4
>>> 
```

2. Press **Enter** to execute the command. The terminal echoes back: **`Hello, World!`**



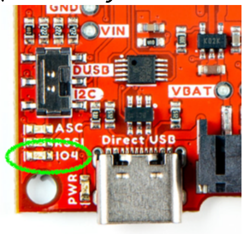
```
>>> print("Hello World")
Hello World
>>> 
```

3. Any line of MicroPython code will work, and variables will be stored between commands. Try typing **x = 2 + 2** then pressing Enter. Finally, type **print(x)** to see the answer.

Example: Turn on an LED (digital pin)

Let's try something with physical output. It's very easy to light up one of the LEDs on the development board. Eventually you will want to learn the full range of input/output controls, to help you create any interaction. For now, we'll start out with the simplest version of the commands and keep things easy.

1. Note the **IO4** LED on the SparkFun XBee Development board. This LED is normally off. (Note: *If you have Digi's XBIB evaluation board use its **DIO10** LED instead.*)



2. At the MicroPython **>>>** prompt, either copy and paste or type in the commands below, pressing **Enter** after each one. After entering the last line of code, the LED illuminates. Anything after a **#** symbol is a comment, and you do not need to type it.

```
from machine import Pin   # Loads the Pin library
led = Pin("D4", Pin.OUT)  # Defines a pin as output
led.on()                  # Turns on that LED
```

3. To turn it off, type the following and press **Enter**:

```
led.off()                # Turns off that LED
```

We will return to this example in the next section, where you will create a program to blink the LED and then load it onto the XBee's file system to run automatically.

Exit MicroPython mode

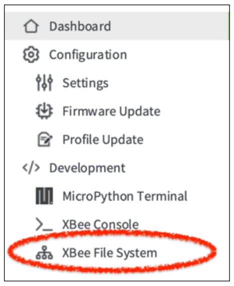
When you are done with the MicroPython terminal, click on the icon to disconnect. This is optional. If you move to another mode in XBee Studio, you will be prompted to disconnect if needed.

File System mode

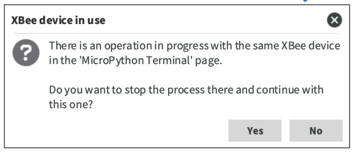
Typing commands into the REPL is a great way to experiment with and learn MicroPython. However, to run programs independently, we will want to store them on the XBee and ensure they are started automatically. The Digi XBee 3 line includes a file system for storing code and other assets, so that applications can be running on the XBee when it starts up.

To access the file system:

1. From the Device menu select **XBee File System**.



NOTE: If you just came from using the MicroPython REPL, you may see a warning that the XBee is already in use. Click Yes to stop that process and continue with the File System.


XBee device in use

There is an operation in progress with the same XBee device in the 'MicroPython Terminal' page.

Do you want to stop the process there and continue with this one?

2. When the XBee File System is displayed, you will see your computer's files listed on the left. On the right you may see a loading indicator Listing directory… ,

followed by a list of files on your XBee. The XBee file root is always `/flash`. Inside it you will find a **lib** and **cert** folder. There is also a folder with two dots (**..**) instead of a name. Use this to go back up a level in the hierarchy. The **lib** folder can hold libraries, also known as modules in MicroPython. The **cert** folder is for securely storing certificates. For the moment it's fine to ignore these directories.

To move a file from your computer to the XBee, you can simply drag and drop from the left to the right window. But first you'll need a file! Let's create a simple MicroPython application that blinks one of the LEDs on the Development board.

Download File

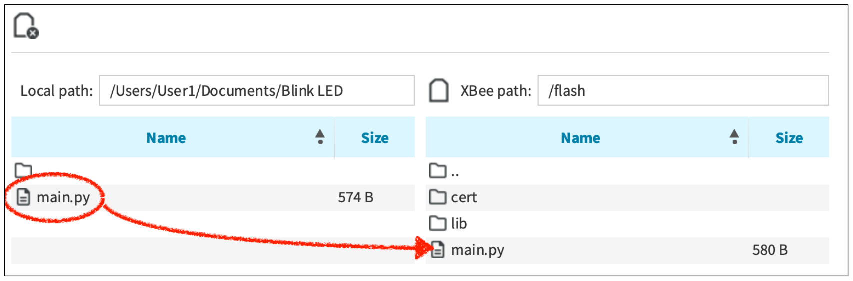
You can download the main.py file for "Blink LED" from here.

… or Create File

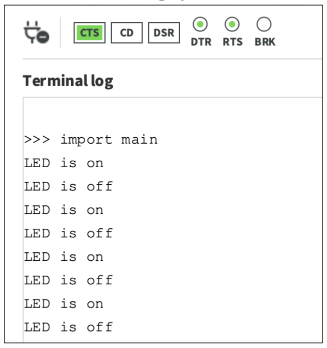
1. Open a text editor such as Notepad on Windows or TextEdit on MacOS.
2. Copy in the following code. Each line ends with a comment that explains what that line does.
   NOTE: if you have a Digi XBIB evaluation board, use D10 instead of D4 as the LED pin.

```
from machine import Pin   # loads the GPIO Pin library from the machine module
import time               # loads the time module
led = Pin('D4', Pin.OUT)  # defines "led" as an output on pin D4 (or D10)
while True:               # run this section of code forever
    led.on()              # turn on the LED
    print('LED on')       # show LED state in the terminal
    time.sleep(1)         # wait for one second
    led.off()             # turn off the LED
    print('LED off')      # show LED state in the terminal
    time.sleep(1)         # wait for one second
```

3. Save the file as "**main.py**" on your local hard drive.
4. Next, in the left side of the XBee Studio's File System interface, navigate to the "**main.py**" file that you created or downloaded. Remember that folders named with two dots will take you up one level in the hierarchy.
5. Once you've located the main.py file, drag it to the XBee's **/flash** directory on the left side of the File System window, anywhere below the "lib" folder.



6. Next, navigate back to the MicroPython Terminal. Notice that you can do this using the submenu for the device: MicroPython Terminal
7. At the MicroPython Terminal click the icon to connect
8. We'll load and run from the REPL prompt first. At the **>>>** prompt, type **import main** to start running your code.



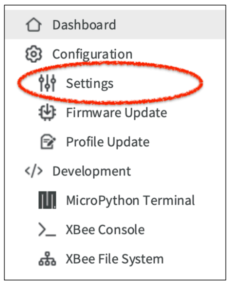
```
>>> import main
LED is on
LED is off
LED is on
LED is off
LED is on
LED is off
LED is on
LED is off
```

9. The LED on the Development board should turn on and off every second.

10. Press **Ctrl+C** in the MicroPython Terminal window to stop the program from running. Note that if you want to run it *again* in the terminal you will need to reset MicroPython using **Ctrl+D**, so that the **import main** command can be loaded again. Otherwise the command will not generate any response.

Running a program from the file system is easier than typing all the code line by line into the REPL, but we want to run this program *every* time the XBee starts up, just the way we'd have to with a device in the field. The **Python Auto Start** (PS) setting on the XBee controls whether MicroPython code is launched at startup. To enable automatic code launching:

1. From the Device menu select **Settings**.

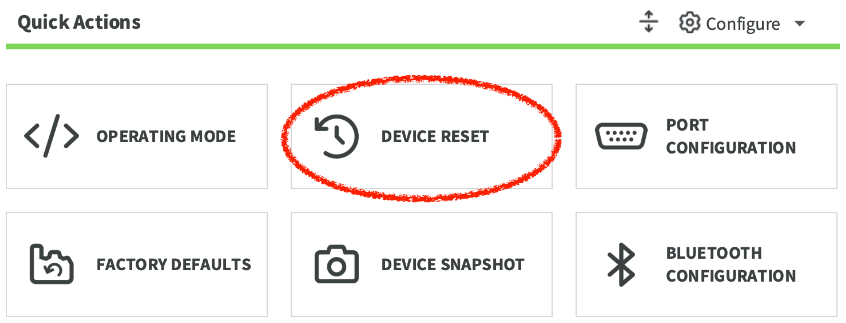


2. Click in the **AT command to search…** field at the top and type **PS** to highlight the **Python Auto Start** setting.
3. Switch **Python Auto Start** to **Enabled [1]** and then click on the pencil icon to save that change to the XBee. Click **Yes** to continue if you get a warning message about the MicroPython Terminal.

With **PS** enabled, the **main.py** program you loaded into the XBee's /**flash** directory will be run automatically on startup. So now all we need to do is reset the XBee:

Reset the XBee

1. In XBee Studio, with your device selected, click on its Dashboard icon in the sidebar
2. Scroll down to the **Quick Actions** section and click on **DEVICE RESET**. This will request a simple software reset.



3. When you are asked to confirm, click **Reset** to restart the firmware on the module and run your MicroPython code.

Once the module is restarted, you should see the LED blinking just as you did when you were running that code from the REPL.

Clean Up

To get ready for the next step, try setting **Python Auto Start** back to **Disabled [0]** and doing another device reset. Your MicroPython code will remain in the file system, but it will not be run on startup.

Get started with Bluetooth Low Energy

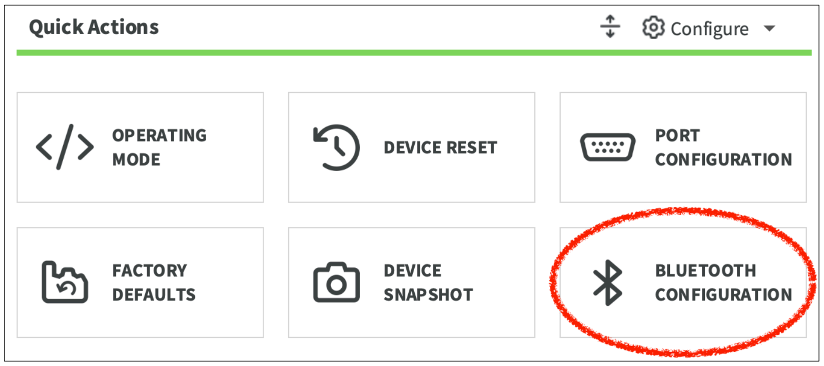
Cellular LTE is only one way to communicate wirelessly with Digi XBee 3 Cellular. If you'd like to configure your XBee from a phone or tablet, or set up an indoor location system using proximity beaconing, or read information from local low-power sensors, Bluetooth Low Energy (BLE) can be enabled on the XBee. BLE is a radio frequency protocol that enables connections from smartphones to your XBee device, or to your XBee from to another device, like local environmental sensors. All current Digi XBee 3 products include a dual-mode radio that allows the device to communicate via the BLE interface and the RF/Cellular network at the same time.

Enable BLE and configure the BLE password using XBee Studio

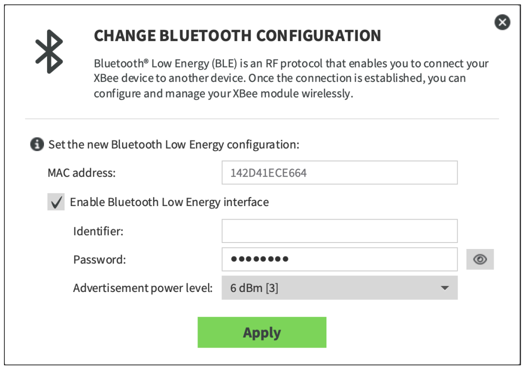
To start using BLE, we must first enable it and configure its security. The BLE password is configured using XBee Studio.

Before you begin, you should determine the password you want to use for BLE on the XBee device and store it in a secure place. Digi recommends a password of *at least* 8 characters with a random combination of letters, numbers, and special characters. Here's how to set it up on an XBee 3 Cellular:

1. In XBee Studio, with your device selected, click on its Dashboard icon in the sidebar
2. Scroll down to the **Quick Actions** section and click on **BLUETOOTH CONFIGURATION**.



3. In the Change Bluetooth Configuration interface, check the **Enable Bluetooth Low Energy Interface** box. Next, **enter a password** to secure the BLE connection.



CHANGE BLUETOOTH CONFIGURATION

Bluetooth® Low Energy (BLE) is an RF protocol that enables you to connect your XBee device to another device. Once the connection is established, you can configure and manage your XBee module wirelessly.

Set the new Bluetooth Low Energy configuration:

MAC address: 142D41ECE664

[✓] Enable Bluetooth Low Energy interface

Identifier:

Password: ••••••••

Advertisement power level: 6 dBm [3]

Apply

(*The identifier field can be left blank. It's also okay to enter a short name here.)*

4. Click **Apply** to save the configuration. The authentication values will be written to your XBee automatically.

Get the Digi XBee Mobile phone application

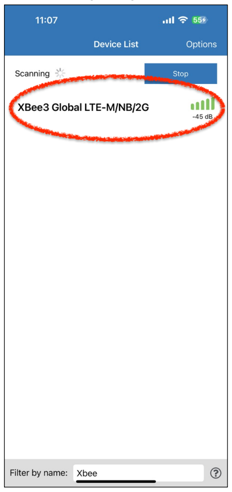
To see the nearby XBee 3 devices that have BLE enabled, you must get the free Digi XBee Mobile application from the iOS App Store or Google Play and install it on your phone. The app is compatible with Android 5.0 or higher or iOS 11 or higher

1. On your smartphone, go into the App Store or Google Play.
2. Search for "**Digi XBee Mobile**".
3. Download and install the application.

Connect with BLE and configure your XBee device (configuration app)

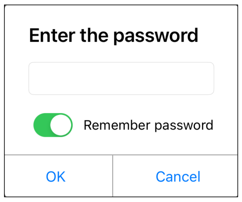
You can use the Digi XBee Mobile application to verify that BLE is enabled on your XBee device.

1. Make sure the Bluetooth antenna is installed to the U.FL connector on the top right of the XBee, as shown in the Connect the Hardware section above.
2. Open the Digi XBee Mobile application on your smartphone, allowing it access to Bluetooth and your location if asked. The Device List screen appears and the app automatically begins scanning for devices. All nearby XBee 3 devices with BLE enabled are displayed in the list.



3. You should see your device listed. If the signal is very weak, confirm that the BLE antenna is properly connected. (*The first time you open the app on a phone and scan for devices, the device list contains only the name of the device and the BLE signal strength. After you have authenticated, the IMEI for the device will also appear in the device list.*)

4. Tap the XBee device name in the list. A password dialog appears.

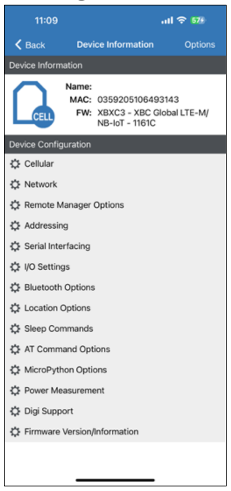


Enter the password

Remember password

OK Cancel

5. Enter the password you previously configured for the device in XBee Studio.
6. Tap OK. The Device Information screen displays. You can now scroll through the settings for the XBee device and change the device's configuration as needed. *Always tap the item you want to change first to expand it!* The color of the settings code shows its state: gray is set to default, blue is set to custom, green is set but not yet written—tap the setting and then the **Write** button to save it on the XBee.



11:09

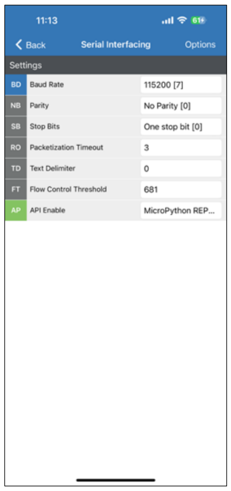
Back Device Information Options

Device Information

Name:
MAC: 0359205106493143
FW: XBXC3 - XBC Global LTE-M/ NB-IoT - 1161C

Device Configuration

- Cellular
- Network
- Remote Manager Options
- Addressing
- Serial Interfacing
- I/O Settings
- Bluetooth Options
- Location Options
- Sleep Commands
- AT Command Options
- MicroPython Options
- Power Measurement
- Digi Support
- Firmware Version/Information

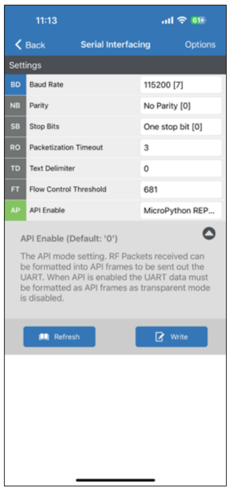


11:13

Back Serial Interfacing Options

Settings

BD	Baud Rate	115200 [7]
NB	Parity	No Parity [0]
SB	Stop Bits	One stop bit [0]
RO	Packetization Timeout	3
TD	Text Delimiter	0
FT	Flow Control Threshold	681
AP	API Enable	MicroPython REP...



11:13

Back Serial Interfacing Options

Settings

BD	Baud Rate	115200 [7]
NB	Parity	No Parity [0]
SB	Stop Bits	One stop bit [0]
RO	Packetization Timeout	3
TD	Text Delimiter	0
FT	Flow Control Threshold	681
AP	API Enable	MicroPython REP...

API Enable (Default: '0')

The API mode setting. RF Packets received can be formatted into API frames to be sent out the UART. When API is enabled the UART data must be formatted as API frames as transparent mode is disabled.

Refresh Write

7. Try opening **I/O Settings**, tap the **D4 DIO4/SPI_MOSI** item to expand it, then change it to **Digital Out, High [5]** to turn on the IO4 LED on the Development Board. Be sure to click **Write** to complete the change, lighting the LED.
8. To turn the LED back off, simply switch the **D4 DIO4/SPI_MOSI** setting to **Disabled [0]** and click **Write** again.

Beaconing

Securely configuring an XBee is a good use of BLE, but Digi XBee 3 modules are capable of much more. For indoor location and positioning, try out the iBeacon or EddyStone Bluetooth beaconing examples for XBee MicroPython. These can be used for wayfinding applications, for indoor asset tracking, such as locating medical equipment within a hospital, or for context-aware systems like customized unattended retail interactions.

BLE Sensors

Another common use of Bluetooth Low Energy is wireless sensors, such as very low power temperature/humidity monitors for restaurant food storage locations. The XBee 3 can read environmental measurements broadcast from these sensors, aggregate them, and pass them along over LTE to online systems. Try out the XBee MicroPython Read and Write Sample Application, with a Silicon Labs Thunderboard BG22 as a remote sensor for this example.

Get started with Digi Remote Manager

Digi Remote Manager (DRM) is a secure platform for monitoring and controlling distributed IoT devices. DRM can be used to configure and update any Digi XBee Cellular, modify its file system, monitor for outages and many other useful remote interactions. Users can create or manage data streams coming from the XBee device, and send command messages to control it asynchronously.

We will show how to create a DRM account, add devices, configure them, monitor them and update their firmware. Next, we will demonstrate how to load MicroPython applications and run them remotely. Finally, we'll look at data streams, to get you ready for remote development and maintenance on your full deployment of Digi XBee Cellular units.

Note: *Digi Remote Manager requires TCP and will not work with NB-IoT, unless the carrier supports TCP.*

Why You'll Love DRM

- Remote device configuration, either manual or automated
- Real-time device health monitoring with alerts
- Full API with push monitors allowing integration with any application
- Remote firmware updates for security and enhancements
- Data Streams for fast, secure access to sensor data
- Scripted automations to handle complex deployments
- Supports sleeping and offline devices with sophisticated asynchronous capabilities

Create a Remote Manager account

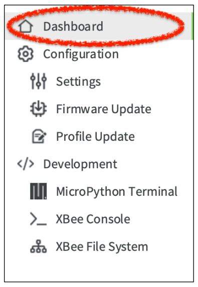
Let's begin by setting up a new Remote Manager account:

1. Go to https://www.digi.com/products/iot-software-services/digi-remote-manager.
2. Click the **90 DAY FREE TRIAL** button: 90 Day Free Trial
3. Follow the online instructions to complete account registration. You can upgrade your Developer account to a paid account at any time to access the full suite of Remote Manager features.
4. An account manager will approve your request. Wait for an email with setup instructions.

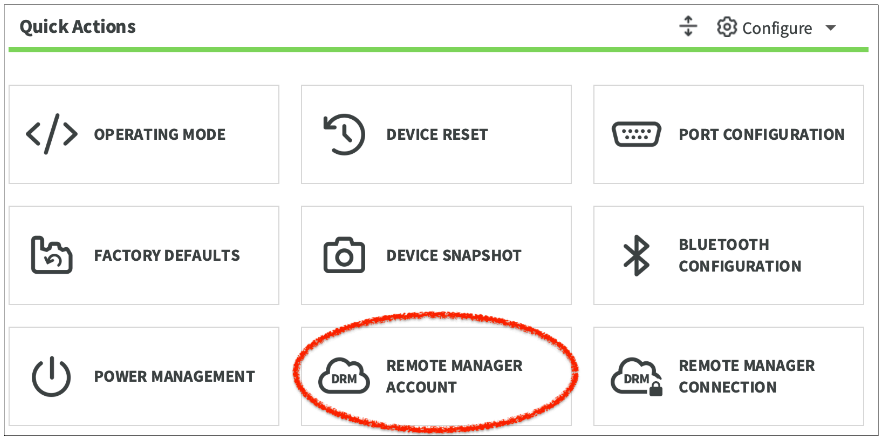
Once your account is approved you will receive a username. Log in to Digi Remote Manager and click **Forgot Password?** to set up a new password. With this username and password, you will be able to attach XBee Studio to your Remote Manager cloud account, so be sure to make a note of them. The next steps will be in XBee Studio, but you will return to Remote Manager's web interface later in this session.

Add Remote Manager account to XBee Studio

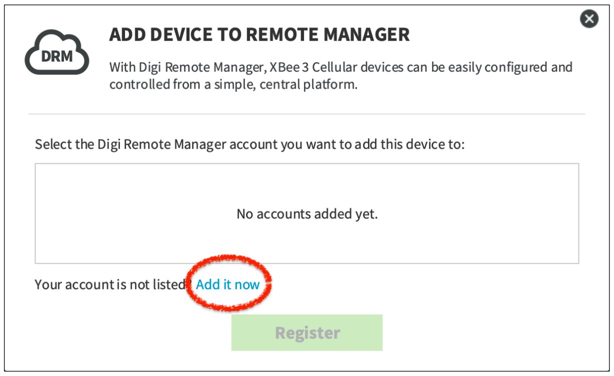
1. In XBee Studio, with your device connected and selected, click on its **Dashboard** icon in device menu.



2. Scroll down to the **Quick Actions** section and click on **REMOTE MANAGER ACCOUNT**



3. An account list will be displayed, which is probably blank. Use the **Add it now** link at the bottom of this list to access your new Remote Manager account.



ADD DEVICE TO REMOTE MANAGER

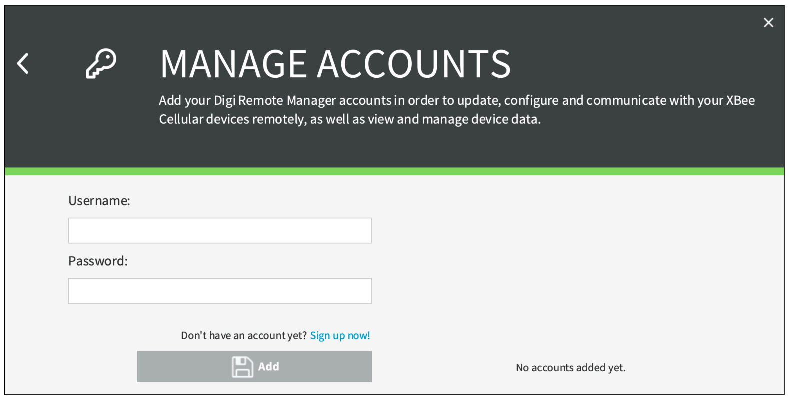
With Digi Remote Manager, XBee 3 Cellular devices can be easily configured and controlled from a simple, central platform.

Select the Digi Remote Manager account you want to add this device to:

No accounts added yet.

Your account is not listed Add it now

4. On the **Manage Accounts** screen, enter your new Remote Manager **username** and **password**, then click the **Add** button.



MANAGE ACCOUNTS

Add your Digi Remote Manager accounts in order to update, configure and communicate with your XBee Cellular devices remotely, as well as view and manage device data.

Username:

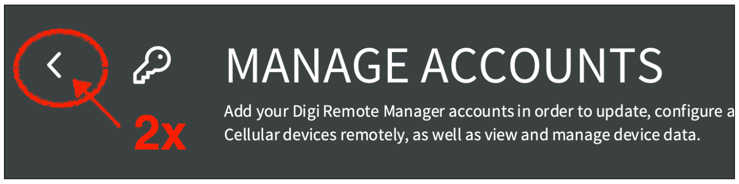
Password:

Don't have an account yet? Sign up now!

Add device to Remote Manager

Now that your Remote Manager account is attached to XBee Studio, you can use XBee Studio to add XBee devices to your Remote Manager account inventory.

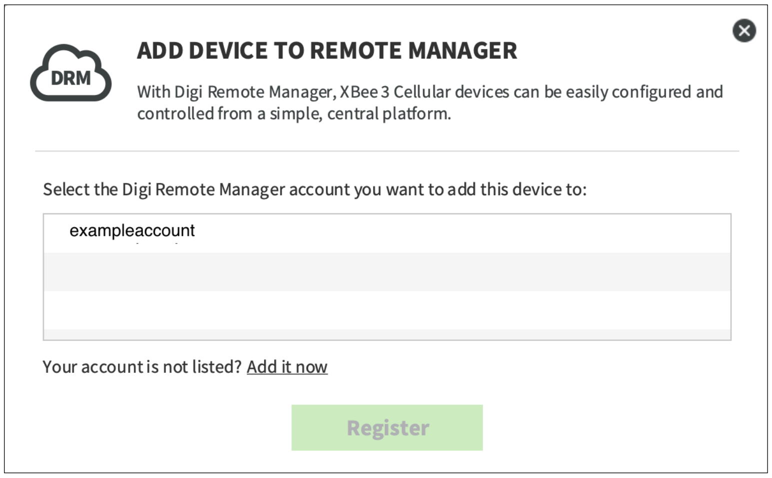
5. From the **Manage Accounts** screen, click the back arrow **<** at the top left ***twice*** to return to the device interface.



MANAGE ACCOUNTS

Add your Digi Remote Manager accounts in order to update, configure a
Cellular devices remotely, as well as view and manage device data.

6. The **ADD DEVICE TO REMOTE MANAGER** should still be displayed, now showing your account. Select this account name and click **Register**. The addition will be confirmed.



ADD DEVICE TO REMOTE MANAGER

With Digi Remote Manager, XBee 3 Cellular devices can be easily configured and controlled from a simple, central platform.

Select the Digi Remote Manager account you want to add this device to:

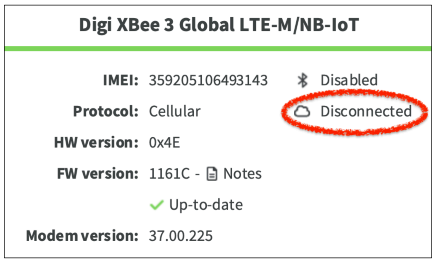
exampleaccount

Your account is not listed? Add it now

Stay Connected to Remote Manager

Digi XBee Cellulars connect to Remote Manager once a day by default. However, for this exercise we'd like it to remain connected all the time.

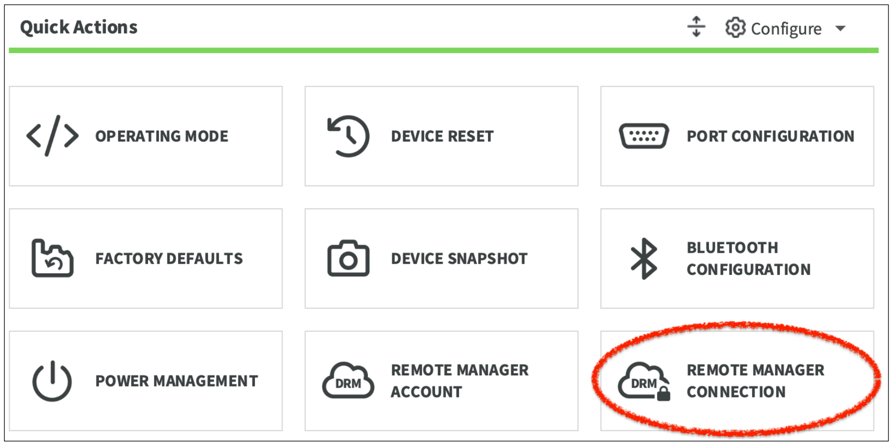
1. Staying at the Dashboard for your device, note that the cloud icon at the top of the Dashboard shows **Disconnected**.



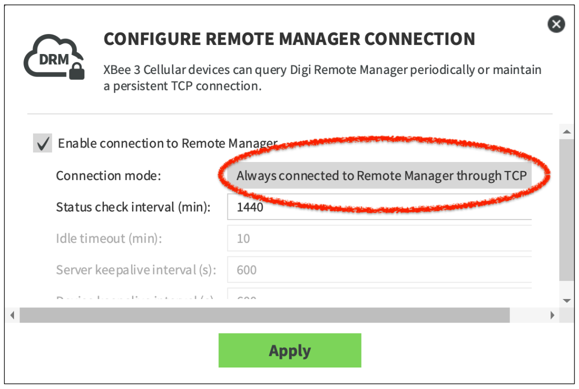
Digi XBee 3 Global LTE-M/NB-IoT

IMEI: 359205106493143 Disabled
Protocol: Cellular Disconnected
HW version: 0x4E
FW version: 1161C - Notes
Up-to-date
Modem version: 37.00.225

2. Next, scroll down to the **Quick Actions** section again, but this time click on **REMOTE MANAGER CONNECTION**.



3. In the **CONFIGURE REMOTE MANAGER CONNECTION** interface, for **Connection Mode** select **Always connected to Remote Manager through TCP** and click **Apply**.



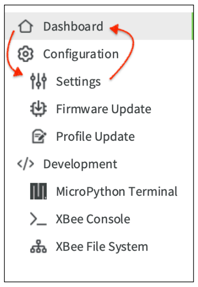
CONFIGURE REMOTE MANAGER CONNECTION

XBee 3 Cellular devices can query Digi Remote Manager periodically or maintain a persistent TCP connection.

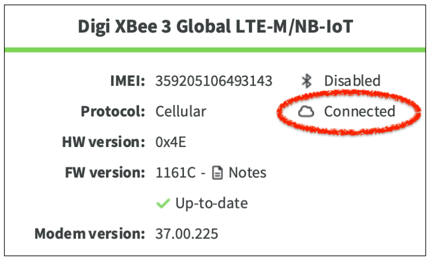
✓ Enable connection to Remote Manager
Connection mode: Always connected to Remote Manager through TCP
Status check interval (min): 1440
Idle timeout (min): 10
Server keepalive interval (s): 600

Apply

4. **Now take a look back at the cloud icon near the top of the Dashboard. If it still says Disconnected** switch to Settings in the device menu briefly, then back to the Dashboard.



5. After about 10 seconds, **Connected** should be displayed.



Digi XBee 3 Global LTE-M/NB-IoT

IMEI:	359205106493143	Disabled
Protocol:	Cellular	Connected
HW version:	0x4E	
FW version:	1161C - Notes	
	✓ Up-to-date	
Modem version:	37.00.225	

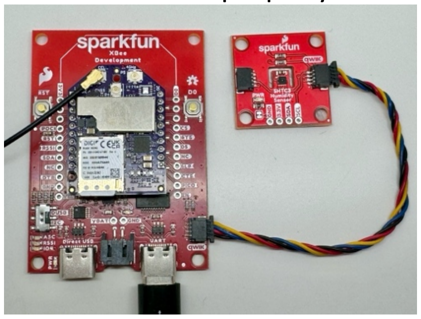
Sensor Data Stream Example

For this next exercise we will create a basic IoT application. You will load a MicroPython program into the file system that reads a local temperature/humidity sensor and send the values to Remote Manager as a Data Stream.

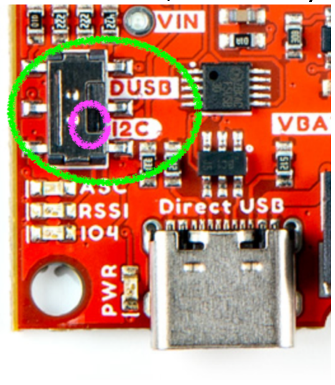
Attach Temperature/Humidity Sensor

At this point your XBee should already be mounted in the Development board with a SIM inserted and the cellular antenna attached.

1. Plug the Qwiic cable into one of the sockets on the SparkFun SHT3 Humidity sensor. Plug the other end of the cable into the Qwiic connector on the Development Board. Be sure the connector is properly seated.

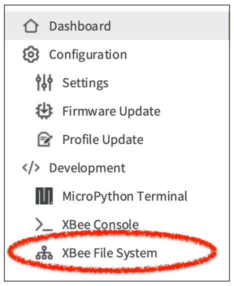


2. Confirm that the switch on the lower left side of the Development Board is in the down position for I2C. If this switch is set incorrectly the sensor will not be able to communicate, so it’s very important!



Load MicroPython Code

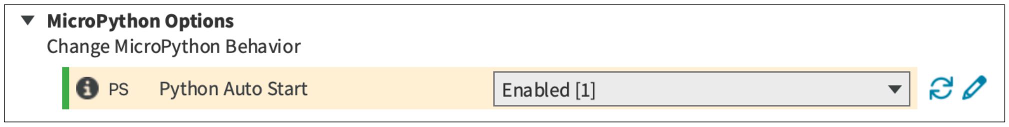
1. Download this code repository and unzip it
2. In the device menu, click on **XBee File System**.



3. Use the left panel of the File System to navigate to the xbee-temp-humidity-demo directory you just unzipped on your computer.
4. On the right panel you should see the contents of the XBee's **/flash** directory, with folders for **cert** and **lib**.
5. Copy **main.py** into the root /flash directory by dragging and dropping. If you get a warning about overwriting the previous file, click **Yes** to overwrite it.
6. Open the **lib** folder and copy the **shtc3.py** file into it.

Auto-start with main.py

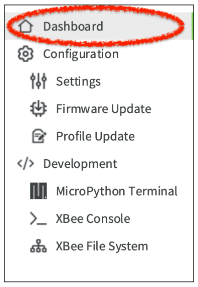
1. In the device menu, click on **Settings** and in the top search box type **PS** to locate **Python AutoStart** in the MicroPython Options section.
2. Change this to **Enabled [1]** and use the pencil icon to save the change.



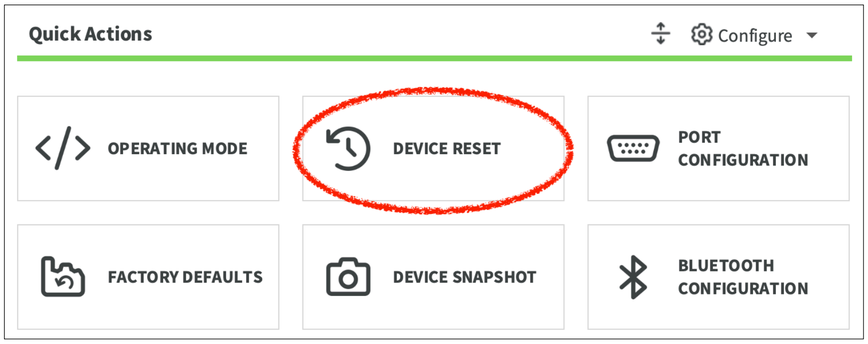
MicroPython Options
Change MicroPython Behavior
PS Python Auto Start Enabled [1]

Reset the XBee

1. Finally, click on its **Dashboard** icon in the device menu.



2. Scroll down to the **Quick Actions** section and click on **DEVICE RESET**.



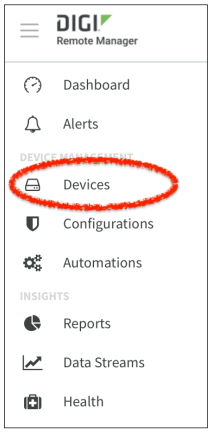
3. When you are asked to confirm, click **Reset** to restart the firmware on the module.
4. After about one minute the device should be fully reconnected and the MicroPython "main.py" program will be running automatically.

NOTE: If you want to stop this in the future, simply change Python Auto Start to Disabled [0]

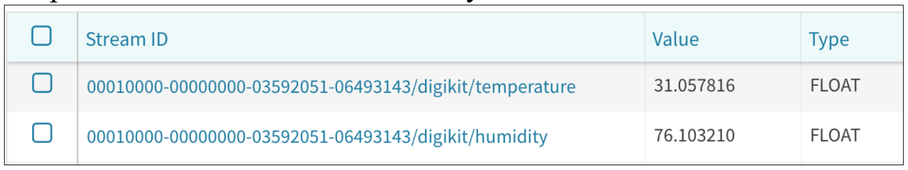
View data streams

To see all the Digi Remote Manager data streams for your devices, follow these steps:

1. Log into Remote Manager.
2. Click **Data Streams** in the left navigation menu.



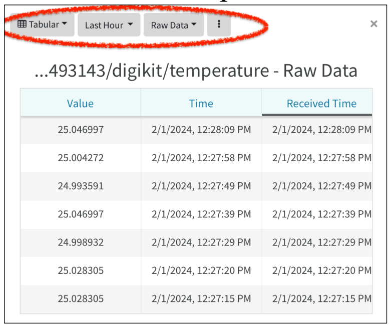
3. Look for two new data streams with **digikit** in their Stream ID. There should be one for temperature and another for humidity:



	Stream ID	Value	Type
☐	00010000-00000000-03592051-06493143/digikit/temperature	31.057816	FLOAT
☐	00010000-00000000-03592051-06493143/digikit/humidity	76.103210	FLOAT

4. Click on the stream ID that ends with **/temperature** to see the data points in the right panel.

5. In the right panel above the graph, select **Tabular**, **Last Hour**, and **Raw Data** to see the most recent data points.



...493143/digikit/temperature - Raw Data

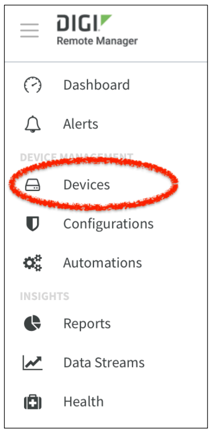
Value	Time	Received Time
25.046997	2/1/2024, 12:28:09 PM	2/1/2024, 12:28:09 PM
25.004272	2/1/2024, 12:27:58 PM	2/1/2024, 12:27:58 PM
24.993591	2/1/2024, 12:27:49 PM	2/1/2024, 12:27:49 PM
25.046997	2/1/2024, 12:27:39 PM	2/1/2024, 12:27:39 PM
24.998932	2/1/2024, 12:27:29 PM	2/1/2024, 12:27:29 PM
25.028305	2/1/2024, 12:27:20 PM	2/1/2024, 12:27:20 PM
25.028305	2/1/2024, 12:27:15 PM	2/1/2024, 12:27:15 PM

6. The data you see here can be exported locally. Use the three-dot menu to select **Export as CSV** to download the selected data as a comma-separated values file that can be loaded into Excel or any other data management program.

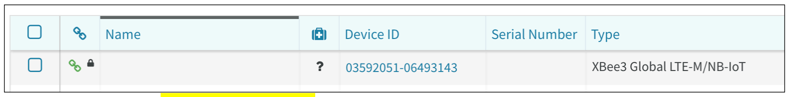
Digi Remote Manager offers a full API for other online applications to manage data streams and access their data. It also offers Push Monitors that send information to other online applications in real time.

Explore Settings

1. In Remote Manager, click on **Devices** in the left navigation menu.

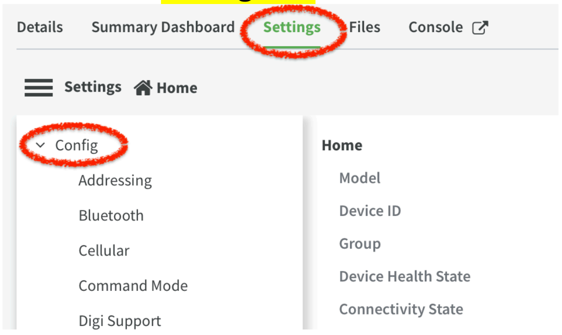


2. Click on your XBee's Device ID in the list to view device **Details**.



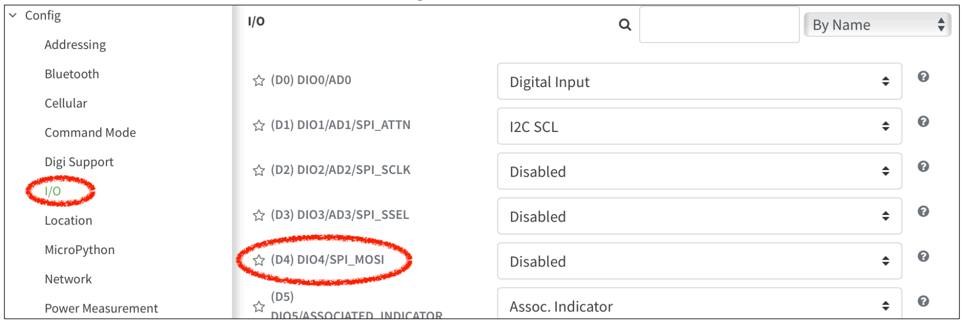
		Name		Device ID	Serial Number	Type
			?	03592051-06493143		XBee3 Global LTE-M/NB-IoT

3. Click on the Settings tab and then click **Config** to expand the device settings.



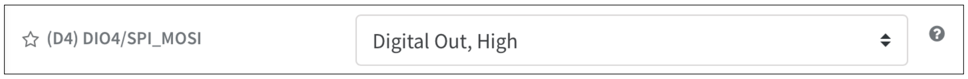
Settings are grouped into categories for convenience. As an example we will use one of the I/O settings to send power to a general-purpose input/output (GPIO) pin connected to the LED on your Development Board.

1. Click on **I/O** in the list of Config settings. Here we can control the state of each GPIO pin on the XBee.
2. Locate the setting labeled **(D4) DIO4/SPI_MOSI**. (*Note: if you are using the Digi XBIB board, use the (P0) DIO10 setting instead.*)



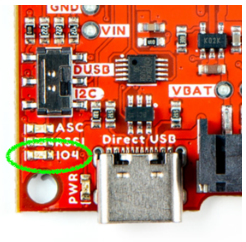
(D0) DIO0/AD0 — Digital Input
(D1) DIO1/AD1/SPI_ATTN — I2C SCL
(D2) DIO2/AD2/SPI_SCLK — Disabled
(D3) DIO3/AD3/SPI_SSEL — Disabled
(D4) DIO4/SPI_MOSI — Disabled
(D5) DIO5/ASSOCIATED_INDICATOR — Assoc. Indicator

3. Adjust the setting to be **Digital Out, High** and then click the **Apply** button at the upper right.

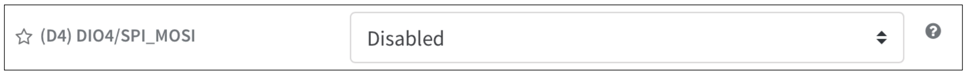


(D4) DIO4/SPI_MOSI — Digital Out, High

4. The user LED on your Development Board will illuminate, demonstrating remote actuation success.



5. Change the setting back to **Disabled** to turn the LED off. Don't forget to hit **Apply** to save your changes!



(D4) DIO4/SPI_MOSI — Disabled

More information about these settings can be found in the User Guide's AT commands section.

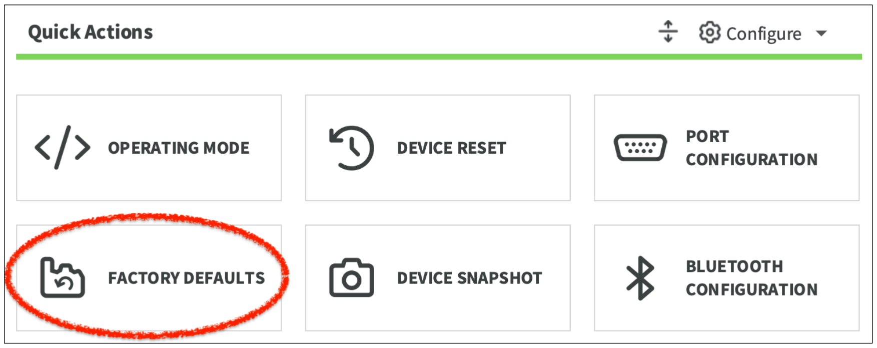
Wrapping Up

Reset XBee to Factory Defaults

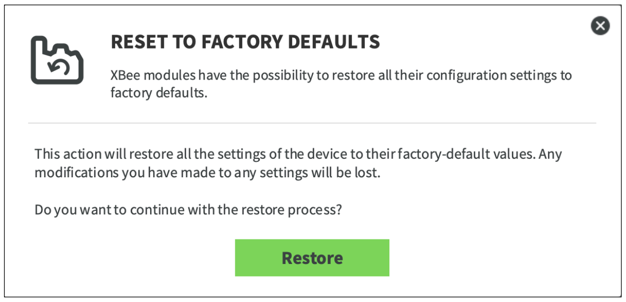
When you are done with these exercises, you may want to return your XBee to its original state. This can be accomplished in two steps:

1. To factory reset the XBee itself, in XBee Studio select the device and then click on Dashboard to view basic info about it.

2. Scroll down to the Quick Actions section and click on **Factory Defaults**



3. You will be asked to confirm. Press the **Restore** button. *This will restore all settings on the device to the factory-default values. Any modifications you have made to any settings will be lost*.



RESET TO FACTORY DEFAULTS

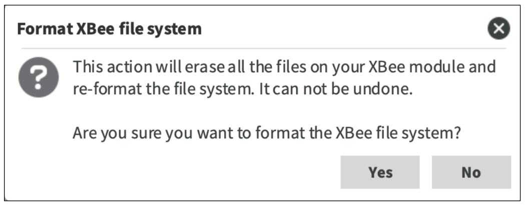
XBee modules have the possibility to restore all their configuration settings to factory defaults.

This action will restore all the settings of the device to their factory-default values. Any modifications you have made to any settings will be lost.

Do you want to continue with the restore process?

If desired, you can also re-format the file system to erase any MicroPython or other files you've uploaded. This is optional.

1. In XBee Studio, select the device and then click on XBee File System.
2. At the top left of the file system interface, click on the icon to format the XBee file system.
3. You will be asked for confirmation. Click Yes to completely reformat and erase the files.



Format XBee file system

This action will erase all the files on your XBee module and re-format the file system. It can not be undone.

Are you sure you want to format the XBee file system?

NOTE: Before completely powering off your XBee, it is recommended to shut down the cellular component first, using the ***Clean Shutdown*** *button in XBee Studio at the top of the Settings List.*

Next Steps

This getting started guide is only the tip of the documentation iceberg. There's plenty of detailed technical information available online. Use these links to learn more.

Digi XBee Cellular Module
Digi XBee 3 Global LTE-M/NB-IoT User Guide
Digi XBee 3 Global LTE-M/NB-IoT Datasheet
Digi XBee 3 Global LTE-M/NB-IoT Product Page

MicroPython
Digi MicroPython Programming Guide
MicroPython Resources for XBee Devices

Remote Manager
Digi Remote Manager User Guide
Digi Remote Manager API Guide
Digi Support Services

Bluetooth
Practical Guide to Using BLE on Digi XBee 3: Application Note

XBee Studio
XBee Studio User Guide
Getting Started with Digi XBee Studio

Hardware
SparkFun Digi XBee Kit
SparkFun Digi XBee Development Board
SparkFun Humidity Sensor Breakout - SHTC3 (Qwiic)

* * *