Digi XBee Mobile

SDK

User Guide

Revision history—90002361

Revision	Date	Description
A	August 2019	Initial release.
B	January 2020	Clarified BLE communication.

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 Mobile SDK User Guide, 90002361 B) in the subject line of your email.

Contents

Digi XBee Mobile SDK User Guide

The Digi XBee 3 family brings a new Bluetooth Low Energy (BLE) communication interface that extends the functionality offered by these devices. It allows you to connect an XBee device with a mobile phone application through a Bluetooth interface and perform different tasks wirelessly:

- Configure the different firmware settings during provisioning operations using a mobile application.
- Exchange data between a mobile application and the MicroPython application running inside the XBee to get and send data to peripherals or perform advanced configuration tasks.
- Exchange data between a mobile application and the micro-controller connected to the serial interface of the XBee device to get and send data to peripherals or even update the firmware of the micro-controller.

Digi has developed the Digi XBee Mobile app that uses the Bluetooth interface to configure different firmware settings of an XBee device and communicate with other interfaces wirelessly. The application is useful during the development process of an XBee solution or for testing purposes, but you may want to develop your own mobile applications to perform more specific tasks or configurations. Developing these mobile applications may become difficult because communicating with the Bluetooth interface of the XBee device involves some authentication and encryption steps. For that reason Digi has created the XBee Mobile SDK.

The XBee Mobile SDK is a set of libraries, examples and documentation that help you develop mobile applications to interact with XBee devices through their BLE interface. For this purpose, Digi provides two easy-to-use libraries that allow you to create XBee mobile native apps:

- XBee Library for Xamarin, to develop cross-platform mobile applications using C# language (iOS and Android).
- XBee Library for Android, to develop Android applications using Java.

These libraries provide a set of APIs that handle all the authentication, encryption and communication processes with the Bluetooth interface of the XBee device. They speed up the development of custom XBee BLE apps with simple examples for different communication models.

- Bluetooth Low Energy in the XBee devices
- Use cases
- Create an XBee Mobile application

Bluetooth Low Energy in the XBee devices

Bluetooth® Low Energy (BLE) is an RF protocol that enables you to connect an XBee device to another device. The latest Digi XBee products include a dual-mode radio that allows the device to communicate through the BLE interface and the RF/Cellular network at the same time.

The libraries included in the XBee Mobile SDK provide all the abstractions and methods required to create mobile applications that communicate with XBee devices over BLE in an easy way.

The XBee is the server and allows client devices, such as a cellphone, to configure the XBee or data transfer with the User Data Relay frame. The XBee cannot communicate with another XBee over BLE, as the XBee is strictly a BLE server. The possibilities are:

- XBee 3: can communicate with mobile devices over BLE
- XBee 3: can communicate with third party devices such as the Nordic nRF and SiLabs BGM over BLE
- XBee 3: cannot communicate with another XBee 3 over BLE

Enable and configure the BLE interface

On the XBee device, the BLE protocol is disabled by default, so the first thing to do if you want to work with the BLE interface is to enable it and configure the authentication password. Follow these steps to do so:

1. Launch XCTU.
2. **Add** your XBee device to the list of radio modules.
3. Switch to **Configuration** working mode.
4. Locate the **Bluetooth Options** settings group and configure the **BT Bluetooth Enable** option to **Enabled [1]**.
5. Click the **Write settings** button from the toolbar.
   a. If the **Bluetooth authentication is not set** dialog appears, click the **Configure** button on that dialog. The **Configure Bluetooth Authentication** dialog appears.
   b. If the **Bluetooth authentication is not set** dialog does not appear, click the **Configure** button of the **Bluetooth Authentication** setting within the **Bluetooth Options** settings group. The **Configure Bluetooth Authentication** dialog appears.
6. Enter the authentication password you want to use to authenticate with the XBee module in the **Password** field.
7. Click **OK** to save the authentication configuration of the XBee module.

Once the BLE interface is enabled, the XBee device starts a GATT service with two characteristics to communicate with it via BLE. The libraries included with the XBee Mobile SDK contain a set of methods that abstract the internals of the BLE communication, so you do not need to worry about writing or reading from characteristics.

Note For more information about the services and characteristics exposed by the GATT server, see the XBee API BLE Service page of your XBee device's documentation.

Now you are ready to start communicating securely with the XBee device.

Open a secure connection with the XBee device

Aside from enabling the BLE interface, it is mandatory to authenticate and unlock communication with the XBee device. This authentication process between a mobile application and the XBee device over BLE is an implementation of the Secure Remote Password (SRP) algorithm. Once the authentication is completed, all communication between the mobile device and XBee device is encrypted following the AES-256-CTR specification.

Note For more information about the SRP authentication process, see the BLE Unlock API page of your XBee device's documentation.

BLE authentication using the XBee Mobile SDK

The libraries included with the XBee Mobile SDK make the authentication and encryption processes transparent. You only need to provide the BLE authentication password configured in the XBee device when instantiating an XBee device class. After opening the connection with the XBee device, the libraries execute the entire authentication process in the background and encrypt/decrypt the data when communicating with the device.

Communicate with the XBee device

The libraries provided by the XBee Mobile SDK include methods to configure and communicate directly with the XBee device via Bluetooth. You do not need to write or read from the characteristics exposed by the XBee API GATT service, the methods provided by the libraries do these tasks underneath.

Once you open the connection and have authenticated with the XBee device you can:

- Configure the XBee device
- Exchange data with other XBee interfaces

Configure the XBee device

One of the features available through the BLE interface is configuring the XBee device. The following methods provided by the XBee Mobile SDK libraries set and get any firmware parameter:

XBee Library for Xamarin

Set a parameter:

```
XBeeBLEDevice.SetParameter(string, byte[])
```

Read a parameter:

```
XBeeBLEDevice.GetParameter(string)
```

XBee Library for Android

Set a parameter:

```
XBeeBLEDevice.setParameter(String, byte[])
```

Read a parameter:

```
XBeeBLEDevice.getParameter(String)
```

Exchange data with other XBee interfaces

The Bluetooth communication interface allows you to receive and send data to other interfaces of the XBee device such as the MicroPython and serial interfaces. It is not possible to communicate with other XBee devices of the network via Bluetooth.

The methods provided by the XBee Mobile SDK libraries to send and receive data from an XBee device via BLE follow:

XBee library for Xamarin

Communicate with the MicroPython interface

Send data:

```
XBeeBLEDevice.SendMicroPythonData(byte[])
```

Receive data:

```
XBeeBLEDevice.MicroPythonDataReceived +=
EventHandler<MicroPythonDataReceivedEventArgs>
```

Communicate with the serial interface

Send data:

```
XBeeBLEDevice.SendSerialData(byte[])
```

Receive data:

```
XBeeBLEDevice.SerialDataReceived += EventHandler<SerialDataReceivedEventArgs>
```

XBee library for Android

Communicate with the MicroPython interface

Send data:

```
XBeeBLEDevice.sendMicroPythonData(byte[])
```

Receive data:

```
XBeeBLEDevice.addMicroPythonDataListener(IMicroPythonDataReceiveListener)
```

Communicate with the serial interface

Send data:

```
XBeeBLEDevice.sendSerialData(byte[])
```

Receive data:

```
XBeeBLEDevice.addSerialDataListener(ISerialDataReceiveListener)
```

For more information about reading and sending data to the XBee device via BLE, refer to the corresponding library's user guide:

- XBee Library for Android user guide
- XBee library for Xamarin user guide

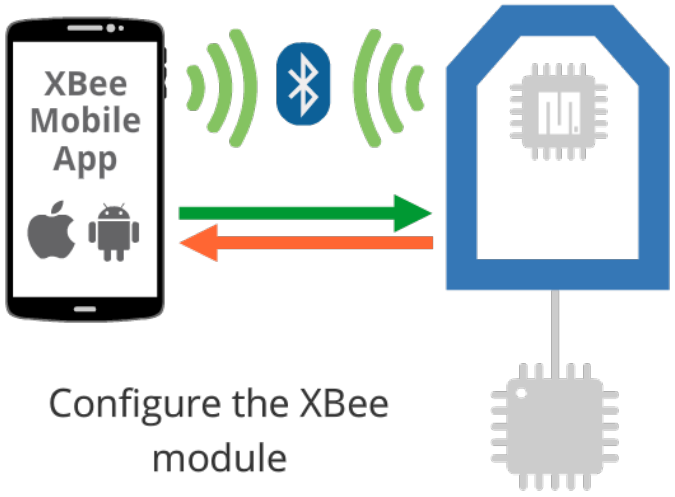
Use cases

The BLE interface of XBee devices is useful to configure the device, send data to the integrated MicroPython interpreter or even forward data to the serial interface. In all cases, a mobile application is needed to communicate with the XBee device over BLE.

Depending on the requirements of your implementation, you will use one of the following cases when communicating with an XBee device through the BLE interface:

Configure the XBee device

One of the purposes of the XBee's BLE interface is to configure the firmware settings of the XBee device from a mobile application.



In this case, the data you send and read from the XBee device is a set of AT configuration parameters. It is useful if you want to check and update the configuration of an XBee device that is already deployed where it is not feasible to establish a connection with it through the serial interface.

Some of the scenarios you may want to configure the XBee device via BLE are:

- Perform a device provisioning operation (initial configuration) of XBee devices during the deployment process of a network. This includes XBee firmware parameters such as the Node Identifier (**NI**), Network ID (**ID**) or Bluetooth password.
- Read diagnostic parameters from XBee devices already deployed in a network.
- Re-configure or update the value of specific parameters in XBee devices already deployed in a network.

Example

The XBee Mobile SDK provides an example that demonstrates this use case. The example shows you how to configure some basic parameters of the XBee firmware from the mobile application. Because there is not an application running inside the XBee device (MicroPython app) or in an external micro-controller, the example only covers the mobile phone side:

Mobile phone side
XBee BLE Configuration Sample - Xamarin
XBee BLE Configuration Sample - Android

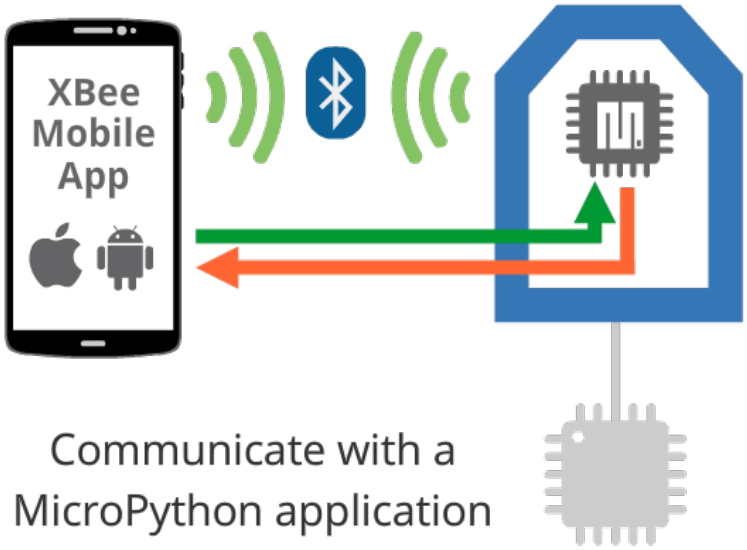
Other resources

In addition, for this use case Digi provides a mobile application called Digi XBee App that allows you to configure XBee devices over BLE. You can find it in both iOS and Android markets:

- Digi XBee Mobile - iOS
- Digi XBee Mobile - Android

Communicate with a MicroPython application

Current XBee modules (XBee 3) integrate MicroPython programmability for edge computing. This allows you to provide some intelligence to the XBee device and create smart end nodes, eliminating the need for an external micro-controller.



In this case, a mobile application communicates with the MicroPython application running inside the XBee device either to act over the XBee device or to collect any kind of data from it.

This use case requires programmability on both the mobile phone and the XBee device side. The MicroPython application that runs inside the XBee device should be developed and transferred using the XBee MicroPython Pycharm Plugin.

Note For more information about the XBee MicroPython PyCharm Plugin, see the XBee MicroPython Programming Guide.

Some of the scenarios you may want to communicate with a MicroPython application via BLE are:

- Perform a device provisioning operation (initial configuration) of XBee devices during the network deployment process. This includes XBee firmware parameters and other settings stored in the internal memory and accessible by the MicroPython application.
- Read diagnostic information from XBee devices already deployed in a network.
- Read transformed values from peripherals connected to the XBee device or command actions to the MicroPython application.
- Provide a graphic UI for the product or system using the mobile phone screen as the interface.

Example

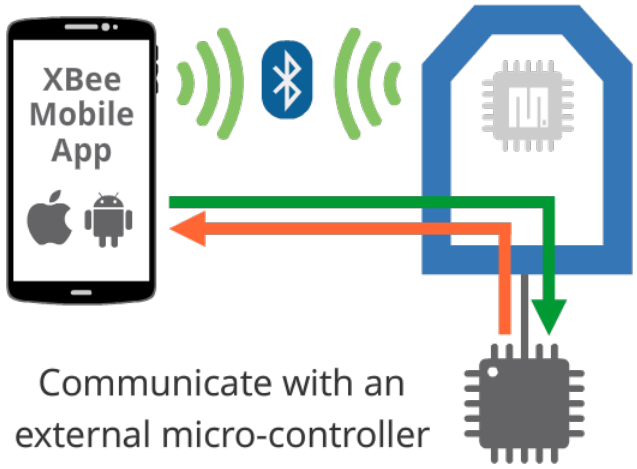
The XBee Mobile SDK provides an example that demonstrates this kind of communication between the mobile phone and the MicroPython application of the XBee device. The example shows you how a

MicroPython application gets the temperature and humidity values from an I2C sensor connected to the XBee device and sends them to a mobile application to be displayed in the screen.

Mobile phone side	XBee module side (MicroPyton)
XBee BLE MicroPython Sample - Xamarin	Relay Frames Temperature Sample - MicroPython
XBee BLE MicroPython Sample - Android	

Communicate with an external micro-controller

In some cases, the XBee device is used as a wireless (RF or Cellular) interface enabler and the intelligence of the device resides in an external micro-controller.



In this case, the purpose of the BLE interface is to communicate with the external micro-controller, forwarding the data sent from the mobile device to the serial interface of the XBee device and vice-versa. This use case is similar to the MicroPython one.

This use case requires a mobile application running on the mobile phone and another application running in the external micro-controller. This last application can be developed using any of the XBee libraries that Digi provides.

Note For more information about the XBee libraries, see XBee Java Library and XBee Python Library.

Some of the scenarios you may want to communicate with an external micro-controller via BLE are:

- Perform a device provisioning operation (initial configuration) of XBee devices during the network deployment process. This includes XBee firmware parameters and other settings stored in the external micro-controller.
- Read diagnostic information from XBee devices already deployed in a network.
- Read transformed values from peripherals connected to the XBee device or to the external micro-controller or command actions to it.
- Provide a graphic UI for the product or system using the mobile phone screen as the interface.
- Transfer files to the external micro-controller or update its firmware over-the-air.

Example

The XBee Mobile SDK includes an example that demonstrates how a mobile application can communicate via BLE with an external micro-controller connected to the XBee device. The example shows you how to send a file from the mobile application and receive it in the micro-controller connected to the XBee device.

Mobile phone side	External micro-controller side
XBee BLE Microcontroller sample - Xamarin	Receive Bluetooth file sample - Java
XBee BLE Microcontroller sample - Android	Receive Bluetooth file sample - Python

Create an XBee Mobile application

The XBee Mobile SDK offers two libraries to develop XBee mobile applications, either cross-platform or Android native applications:

Cross-platform applications

Cross-platform application development has become a cost-effective solution to develop mobile applications that behave almost like native applications for multiple platforms. Cross-platform application development frameworks allow you to create mobile applications that are compatible with different mobile operating systems—commonly Android and iOS. Instead of developing two separate apps you only develop one, then share that code among operating systems, making platform-specific adjustments as necessary. The pros and cons of cross-platform application development depend on the framework used to develop them.

Digi has chosen Xamarin as the cross-platform application development framework to create an XBee mobile library: the XBee Library for Xamarin. Some of the reasons are:

- Xamarin uses C# as single language to create apps for all mobile platforms. In C# you can do anything that can be achieved with Objective-C, Swift, or Java.
- Very high percentage of code share and reuse between platforms (up to 96%).
- Xamarin allows you to create platform-specific UI code layer when the cross-platform solution does not work.
- Native UI controllers allow Xamarin to render native user interfaces for each operating system.
- Provides access to native APIs and libraries

The XBee Library for Xamarin includes an specific API to handle the connection and communication with XBee devices through BLE in mobile applications developed with Xamarin.

Note See the Software requirements of the XBee Library for Xamarin before creating or importing an application.

Create an application from scratch

Follow these steps to create an XBee mobile cross-platform application with Xamarin from scratch:

- Create an XBee Xamarin application from scratch

Import a sample application

Follow these steps to import an XBee mobile cross-platform sample application with Xamarin:

- Import an XBee Xamarin sample application

Android native applications

Since Digi already has a Java library to manage and communicate with XBee devices, we crated an extension of that library that would help create XBee mobile applications for Android: the XBee library for Android.

The XBee library for Android handles the connection and communication processes with XBee devices in Android applications developed with Android Studio.

Create an application from scratch

Follow these steps to create an XBee mobile Android application from scratch:

- Create an XBee Android application from scratch

Import a sample application

Follow these steps to import an XBee mobile Android sample application:

- Import an XBee Android sample application