**Bluetooth in Android**

The Android platform includes support for the Bluetooth network stack, which allows a device to wirelessly exchange data with other Bluetooth devices. The application framework provides access to the Bluetooth functionality through the Android Bluetooth APIs. These APIs let applications wirelessly connect to other Bluetooth devices, enabling point-to-point and multipoint wireless features.

Using the Bluetooth APIs, an Android application can perform the following:

* Scan for other Bluetooth devices
* Query the local Bluetooth adapter for paired Bluetooth devices
* Establish RFCOMM channels
* Connect to other devices through service discovery
* Transfer data to and from other devices
* Manage multiple connections

**Packages and class required for Bluetooth connection**

All of the Bluetooth APIs are available in the [android. Bluetooth](http://developer.android.com/reference/android/bluetooth/package-summary.html) package. Here's a summary of the classes and interfaces you will need to create Bluetooth connections:

**BluetoothAdapter**

Represents the local Bluetooth adapter (Bluetooth radio). The **BluetoothAdapter** is the entry-point for all Bluetooth interaction. Using this, you can discover other Bluetooth devices, query a list of bonded (paired) devices, instantiate **a BluetoothDevice** using a known MAC address, and create **a BluetoothServerSocket** to listen for communications from other devices.

[**BluetoothDevice**](http://developer.android.com/reference/android/bluetooth/BluetoothDevice.html)

Represents a remote Bluetooth device. Use this to request a connection with a remote device through a **BluetoothSocket** or query information about the device such as its name, address, class, and bonding state.

[**BluetoothSocket**](http://developer.android.com/reference/android/bluetooth/BluetoothSocket.html)

Represents the interface for a Bluetooth socket (similar to a TCP [Socket](http://developer.android.com/reference/java/net/Socket.html)). This is the connection point that allows an application to exchange data with another Bluetooth device via **InputStream** and **OutputStream**.

[**BluetoothServerSocket**](http://developer.android.com/reference/android/bluetooth/BluetoothServerSocket.html)

Represents an open server socket that listens for incoming requests (similar to a TCP **ServerSocket**). In order to connect two Android devices, one device must open a server socket with this class. When a remote Bluetooth device makes a connection request to the device, the [BluetoothServerSocket](http://developer.android.com/reference/android/bluetooth/BluetoothServerSocket.html) will return a connected [BluetoothSocket](http://developer.android.com/reference/android/bluetooth/BluetoothSocket.html) when the connection is accepted.

[**BluetoothClass**](http://developer.android.com/reference/android/bluetooth/BluetoothClass.html)

Describes the general characteristics and capabilities of a Bluetooth device. This is a read-only set of properties that define the device's major and minor device classes and its services. However, this does not reliably describe all Bluetooth profiles and services supported by the device, but is useful as a hint to the device type.

**Bluetooth Permission**

* **Android.permission.BLUETOOTH:** In order to use Bluetooth features in your application, you must declare the Bluetooth permission BLUETOOTH. You need this permission to perform any Bluetooth communication, such as requesting a connection, accepting a connection, and transferring data.
* **Android.permission.BLUETOOTH\_ADMIN:**  If you want your app to initiate device discovery or manipulate Bluetooth settings, you must also declare the [BLUETOOTH\_ADMIN](http://developer.android.com/reference/android/Manifest.permission.html#BLUETOOTH_ADMIN) permission. Most applications need this permission solely for the ability to discover local Bluetooth devices. The other abilities granted by this permission should not be used, unless the application is a "power manager" that will modify Bluetooth settings upon user request.

**Note:** If you use [BLUETOOTH\_ADMIN](http://developer.android.com/reference/android/Manifest.permission.html#BLUETOOTH_ADMIN) permission, then you must also have the [BLUETOOTH](http://developer.android.com/reference/android/Manifest.permission.html#BLUETOOTH) permission.

<manifest ... >  
  <uses-permission android:name="android.permission.BLUETOOTH" />  
  ...  
</manifest>

**SETTING UP A BLUTOOTH**

Before your application can communicate over Bluetooth, you need to verify that Bluetooth is supported on the device, and if so, ensure that it is enabled.

If Bluetooth is not supported, then you should gracefully disable any Bluetooth features. If Bluetooth is supported, but disabled, then you can request that the user enable Bluetooth without leaving your application. This setup is accomplished in two steps, using the [BluetoothAdapter](http://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html).

1. Get the [BluetoothAdapter](http://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html)

The [BluetoothAdapter](http://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html) is required for any and all Bluetooth activity. To get the [BluetoothAdapter](http://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html), call the static [getDefaultAdapter()](http://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html#getDefaultAdapter%28%29) method. This returns a [BluetoothAdapter](http://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html) that represents the device's own Bluetooth adapter (the Bluetooth radio). There's one Bluetooth adapter for the entire system, and your application can interact with it using this object. If [getDefaultAdapter()](http://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html#getDefaultAdapter%28%29) returns null, then the device does not support Bluetooth and your story ends here.

BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();  
if (mBluetoothAdapter == null) {  
    // Device does not support Bluetooth  
}

1. Enable Bluetooth

Next, you need to ensure that Bluetooth is enabled. Call [isEnabled()](http://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html#isEnabled%28%29) to check whether Bluetooth is currently enable. If this method returns false, then Bluetooth is disabled. To request that Bluetooth be enabled, call [startActivityForResult()](http://developer.android.com/reference/android/app/Activity.html#startActivityForResult%28android.content.Intent,%20int%29) with the ACTION\_REQUEST\_ENABLE action Intent. This will issue a request to enable Bluetooth through the system settings (without stopping your application). For example:

If (!mBluetoothAdapter.isEnabled()) {  
    Intent enableBtIntent = new Intent (BluetoothAdapter.ACTION\_REQUEST\_ENABLE);  
    startActivityForResult(enableBtIntent, REQUEST\_ENABLE\_BT);  
}

A dialog will appear requesting user permission to enable Bluetooth, as shown in Figure 1. If the user responds "Yes," the system will begin to enable Bluetooth and focus will return to your application once the process completes (or fails).

The REQUEST\_ENABLE\_BT constant passed to [startActivityForResult ()](http://developer.android.com/reference/android/app/Activity.html#startActivityForResult%28android.content.Intent,%20int%29) is a locally defined integer (which must be greater than 0), that the system passes back to you in your onActivityResult() implementation as the requestCode parameter.

If enabling Bluetooth succeeds, your activity receives the [RESULT\_OK](http://developer.android.com/reference/android/app/Activity.html#RESULT_OK) result code in the onActivityResult() callback. If Bluetooth was not enabled due to an error (or the user responded "No") then the result code is RESULT\_CANCELED.

Optionally, your application can also listen for the ACTION\_STATE\_CHANGED broadcast Intent, which the system will broadcast whenever the Bluetooth state has changed. This broadcast contains the extra fields EXTRA\_STATE and EXTRA\_PREVIOUS\_STATE, containing the new and old Bluetooth states, respectively. Possible values for these extra fields are STATE\_TURNING\_ON, [STATE\_ON](http://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html#STATE_ON), STATE\_TURNING\_OFF, and STATE\_OFF. Listening for this broadcast can be useful to detect changes made to the Bluetooth state while your app is running.

Reference:

1)Android.developer.com