

Bluetooth: Managing a Bluetooth Device and Supporting RFCOMM and HDP

This tutorial demonstrates how you can manage Bluetooth and exchange data with a peer device.

The Bluetooth API is optional for both Tizen mobile and wearable profiles, which means that it may not be supported in all mobile and wearable devices. The Bluetooth API is not supported on any Tizen Emulators.



Warm-up

Become familiar with the Bluetooth API basics by learning about:

- [Managing the Local Bluetooth Adapter](#)
Enable and disable the local Bluetooth adapter, and change the device name for it.
- [Discovering Bluetooth Devices](#)
Search remote devices and get a list of the known devices.
- [Creating a Bonding with a Bluetooth Device](#)
Create and end a bonding with a Bluetooth device.
- [Connecting to and Exchanging Data with a Bluetooth Device](#)
Register a service as a server, connect as a client to the service provided by the server device, and exchange data with the device.
- [Communicating with a Health Source Device](#)
Act as a sink and communicate with a health source device.
- **Bluetooth Low Energy**
 - [Discovering Bluetooth Low Energy Devices](#)
Search for remote devices.
 - [Managing the Advertising Options](#)
Manage what information is advertised to any Bluetooth Low Energy device in the proximity.
 - [Connecting to a Bluetooth Low Energy Device](#)
Connect to a detected Bluetooth Low Energy device.
 - [Receiving Notifications on Connection State Changes](#)
Monitor the connection state to a remote Bluetooth Low Energy device.
 - [Retrieving Bluetooth GATT Services](#)
Check information about the Bluetooth GATT services provided by a remote device.
 - [Accessing the Bluetooth GATT Characteristic Value](#)
Read and write a value of the Bluetooth GATT characteristic of a remote device.
 - [Receiving Notifications on Characteristic Value Changes](#)
Monitor changes in a Bluetooth GATT characteristic value.
 - [Accessing the Bluetooth GATT Descriptor Value](#)
Read and write a value of the Bluetooth GATT characteristic descriptor of a remote device.

Task

In the [Bluetooth Chat \(/development/tutorials/web-application/tizen-features/network/bluetooth/task-bluetooth-chat\)](#) task, we will walk through how to use the device as a Bluetooth server or client in a chat application.

Prerequisites

To use the Application and Bluetooth APIs, the application has to request permission by adding the following privileges to the config.xml file:

Content

- [Prerequisites](#)
- [Managing the Local Bluetooth Adapter](#)
- [Discovering Bluetooth Devices](#)
- [Creating a Bonding with a Bluetooth Device](#)
- [Connecting to and Exchanging Data with a Bluetooth Device](#)
- [Communicating with a Health Source Device](#)
- **Bluetooth Low Energy**
 - [Discovering Bluetooth Low Energy Devices](#)
 - [Managing the Advertising Options](#)
 - [Connecting to a Bluetooth Low Energy Device](#)
 - [Receiving Notifications on Connection State Changes](#)
 - [Retrieving Bluetooth GATT Services](#)
 - [Accessing the Bluetooth GATT Characteristic Value](#)
 - [Receiving Notifications on Characteristic Value Changes](#)
 - [Accessing the Bluetooth GATT Descriptor Value](#)

Related Info

- [Bluetooth Guide \(/development/guides/web-application/tizen-features/network/bluetooth\)](#)
- [Bluetooth API for Mobile Web \(https://developer.tizen.org/development/guide/2.4/org.tizen.web.api.reference/html/device_api/mobile/tizen/bluetooth.html\)](#)
- [Bluetooth API for Wearable Web \(https://developer.tizen.org/development/guide/2.4/org.tizen.web.api.reference/html/device_api/wearable/tizen/bluetooth.html\)](#)

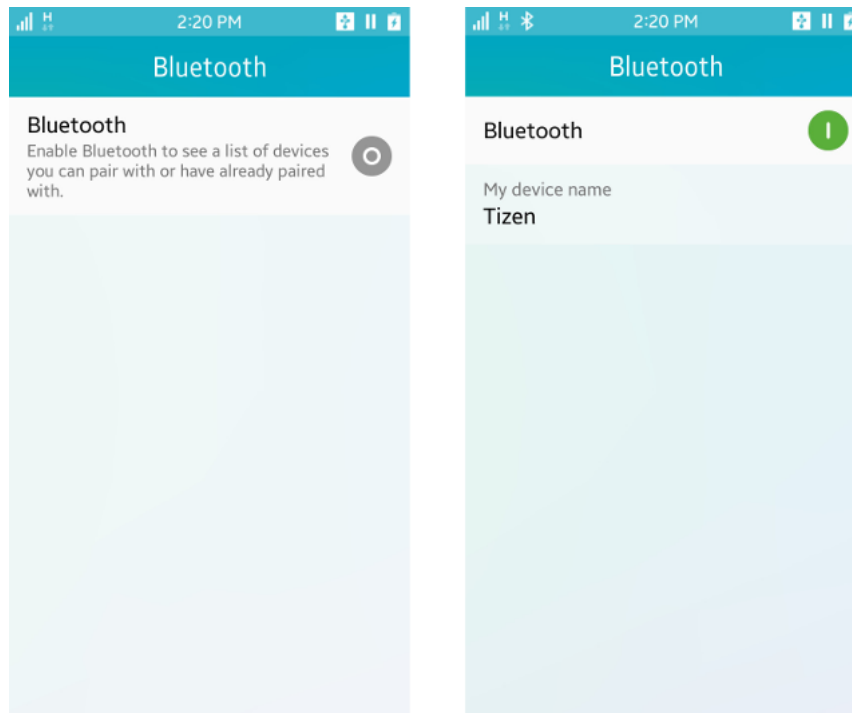
```
<tizen:privilege name="http://tizen.org/privilege/application.launch"/>
<tizen:privilege name="http://tizen.org/privilege/bluetooth"/>
```

Managing the Local Bluetooth Adapter

Learning how to enable or disable the local Bluetooth adapter, and set the device name is a basic Bluetooth management skill:

The Bluetooth API does not provide a method to enable or disable the Bluetooth adapter of the device directly. Whenever Bluetooth is required, request a built-in Settings application to present the relevant switch to the user so that they can enable or disable the Bluetooth.

Figure: Bluetooth setting screen



1. Retrieve a `BluetoothAdapter` object (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothAdapter) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothAdapter) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothAdapter) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothAdapter) applications) with the `getDefaultAdapter()` method and prepare the `ApplicationControl` object (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/application.html#ApplicationControl) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/application.html#ApplicationControl) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/application.html#ApplicationControl) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/application.html#ApplicationControl) applications) to request the Bluetooth switching operation:

```
var bluetoothSwitchAppControl = new tizen.ApplicationControl("http://tizen.org/appcontrol/operation/edit", null, "application/x-bluetooth-on-off");
var adapter = tizen.bluetooth.getDefaultAdapter();
```

2. Define a callback for the `launchAppControl()` method:

```
function launchSuccess()
{
    console.log("Bluetooth Settings application is successfully launched.");
}
function launchError(error)
{
    alert("An error occurred: " + error.name + ". Please enable Bluetooth through the Settings application.");
}
```

3. Define the reply callback of the application control which implements the `ApplicationControlDataArrayReplyCallback` (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/application.html#ApplicationControlDataArrayReplyCallback) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/application.html#ApplicationControlDataArrayReplyCallback) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/application.html#ApplicationControlDataArrayReplyCallback) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/application.html#ApplicationControlDataArrayReplyCallback) applications):

```

var serviceReply =
{
    /* Called when the launched application reports success */
    onsuccess: function(data)
    {
        if (adapter.powered)
        {
            console.log("Bluetooth is successfully turned on.");
        }
        else
        {
            console.log("Bluetooth is still switched off.");
        }
    },
    /* Called when launched application reports failure */
    onfailure: function()
    {
        alert("Bluetooth Settings application reported failure.");
    }
};

```

4. If necessary, request launching the Bluetooth Settings with the prepared `bluetoothSwitchAppControl`:

```

if (adapter.powered)
{
    console.log("Bluetooth is already enabled");
}
else
{
    console.log("Try to launch the Bluetooth Settings application.");
    tizen.application.launchAppControl(bluetoothSwitchAppControl, null, launchSuccess, launchError, serviceReply);
}

```

5. To display the Bluetooth visibility switch, use the `application/x-bluetooth-visibility` mime option. Bluetooth visibility means that the device is discoverable by other Bluetooth devices.

```

var bluetoothVisibilityAppControl = new tizen.ApplicationControl("http://tizen.org/appcontrol/operation/edit", null, "application/x-bluetooth-visibility");
function launchVisibilityError(error)
{
    alert("An error occurred: " + error.name + ". Please enable Bluetooth visibility through the Settings application.");
}
var serviceVisibilityReply =
{
    /* Called when the launched application reports success */
    onsuccess: function(data)
    {
        console.log("Bluetooth is " + (adapter.visible ? "now discoverable." : "still not visible."));
    },
    /* Called when launched application reports failure */
    onfailure: function()
    {
        alert("Bluetooth Settings application reported failure.");
    }
};
tizen.application.launchAppControl(bluetoothVisibilityAppControl, null, null, launchVisibilityError, serviceVisibilityReply);

```

6. Set a friendly name for the device using the `setName()` method.

The name helps to recognize the device in a list of [retrieved devices](#).

```

adapter.setName(chatServerName);

```

Discovering Bluetooth Devices

Learning how to search for remote devices and get the known devices is a basic Bluetooth management skill:

1. Retrieve a `BluetoothAdapter` object (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothAdapter\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothAdapter) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothAdapter\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothAdapter) applications) with the `getDefaultAdapter()` method:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
```

2. To search for remote devices, use the `discoverDevices()` method.

The results of the search are returned in the `BluetoothDiscoverDevicesSuccessCallback` (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothDiscoverDevicesSuccessCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothDiscoverDevicesSuccessCallback) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothDiscoverDevicesSuccessCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothDiscoverDevicesSuccessCallback) applications).

```
var discoverDevicesSuccessCallback =
{
  /* When a device is found */
  ondevicefound: function(device)
  {
    console.log("Found device - name: " + device.name);
  }
}

/* Discover devices */
adapter.discoverDevices(discoverDevicesSuccessCallback, null);
```

Note:

To allow other Bluetooth devices to find your device, you must set the device to be visible through the system settings.

3. To retrieve known devices (which have been previously paired or searched for), use the `getKnownDevices()` method.

The results of the search are returned in the `BluetoothDeviceArraySuccessCallback` (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothDeviceArraySuccessCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothDeviceArraySuccessCallback) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothDeviceArraySuccessCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothDeviceArraySuccessCallback) applications).

```
/* When a known device is found */
function onGotDevices(devices)
{
  console.log("The number of known devices: " + devices.length);
}

/* Retrieve known devices */
adapter.getKnownDevices(onGotDevices);
```

Creating a Bonding with a Bluetooth Device

Learning how to create a bonding with other devices is a basic Bluetooth management skill:

1. Retrieve a `BluetoothAdapter` object (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothAdapter\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothAdapter) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothAdapter\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothAdapter) applications) with the `getDefaultAdapter()` method:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
```

2. To create a bonding with another device, use the `createBonding()` method:

```
function onBondingSuccessCallback(device)
{
    console.log("A bonding is created - name: " + device.name);
}

function onErrorCallback(e)
{
    console.log("Cannot create a bonding, reason: " + e.message);
}

adapter.createBonding("35:F4:59:D1:7A:03", onBondingSuccessCallback, onErrorCallback);
```

Note:

The MAC address of the Bluetooth device is a `BluetoothAddress` object (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothAddress) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothAddress) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothAddress) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothAddress) applications). You can get the MAC address of the peer device from the `BluetoothDevice` object (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothDevice) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothDevice) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothDevice) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothDevice) applications), which is returned in the success callback of the `BluetoothAdapter`'s `getKnownDevices()` and `discoverDevices()` methods.

3. To end the bonding with a remote device, use the `destroyBonding()` method:

```
adapter.destroyBonding("35:F4:59:D1:7A:03");
```

Connecting to and Exchanging Data with a Bluetooth Device

The Radio Frequency Communication (RFCOMM) is a set of transport protocols which allows multiple simultaneous connections to a device. The device that provides a service is called a server device, and devices that request the service are called client devices.

Learning how to connect to services provided by a server device to the client devices is a basic Bluetooth management skill:

1. Retrieve a `BluetoothAdapter` object (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothAdapter) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothAdapter) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothAdapter) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothAdapter) applications) with the `getDefaultAdapter()` method:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
```

2. To register a service and allow client devices to connect to it, use the `registerRFCOMMServiceByUUID()` method on the server device:

```
adapter.registerRFCOMMServiceByUUID(serviceUUID, "My service");
```

Note:

For P2P communication between 2 instances of the same application, the UUID can be hard-coded in your application. To retrieve the UUID of a Bluetooth device, use the `BluetoothDevice` object (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothDevice) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothDevice) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothDevice) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothDevice) applications). The object has an array of UUIDs available for the device.

When the service has been successfully registered, the `BluetoothServiceSuccessCallback` interface (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothServiceSuccessCallback) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothServiceSuccessCallback) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothServiceSuccessCallback) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothServiceSuccessCallback) applications) is triggered.

3. Before establishing a connection, your device must be bonded with a peer device. For more information, see [Creating a Bonding with a Bluetooth Device](#).
4. To connect to the server device, use the `connectToServiceByUUID()` method on the client device:

```

device.connectToServiceByUUID(serviceUUID, function(sock)
{
    console.log("socket connected");
    socket = sock;
},
function(error)
{
    console.log("Error while connecting: " + error.message);
}
);

```

When a connection between 2 devices is established, the `BluetoothSocketSuccessCallback` interface (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothSocketSuccessCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothSocketSuccessCallback) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothSocketSuccessCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothSocketSuccessCallback) applications) on the client device and the `onconnect` event handler in the `BluetoothServiceHandler` interface (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothServiceHandler\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothServiceHandler) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothServiceHandler\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothServiceHandler) applications) on the server device are triggered.

5. To send data to the peer device, use the `writeData()` method:

```

var somemsg = [3, 2, 1];
var length = socket.writeData(somemsg);

```

To send data between the devices, use a socket mechanism with the `BluetoothSocket` interface (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothSocket\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothSocket) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothSocket\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothSocket) applications). The proper socket is received when the devices are connected.

6. To read the data in the server device, use the `readData()` method:

```

var data = socket.readData();

```

When an incoming message is received from the peer device, the `onmessage` event handler in the `BluetoothSocket` interface is triggered.

Communicating with a Health Source Device

To increase the communication capabilities of your application, you must learn to communicate with a health source device:

1. Retrieve a `BluetoothHealthProfileHandler` object (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothHealthProfileHandler\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothHealthProfileHandler) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothHealthProfileHandler\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothHealthProfileHandler) applications):

```

var adapter = tizen.bluetooth.getDefaultAdapter();
var healthProfileHandler = adapter.getBluetoothProfileHandler("HEALTH");
var healthApplication = null, healthChannel = null;

```

2. Register an application as a sink to wait for connection requests from health source devices (4100 means oximeter):

```

function onSinkApp(app)
{
    console.log("Success");
    healthApplication = app;
}

healthProfileHandler.registerSinkApplication(4100, "testSinkApp", onSinkApp);

```

When the sink application is registered successfully, the `BluetoothHealthApplicationSuccessCallback` interface (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothHealthApplicationSuccessCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothHealthApplicationSuccessCallback) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothHealthApplicationSuccessCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothHealthApplicationSuccessCallback) applications) is invoked and you can get the registered sink application object.

3. Before establishing a connection, your device must be bonded with a health source device. For more information, see [Creating a Bonding with a Bluetooth Device](#).

4. To connect to the health source device, use the `connectToSource()` method of the `BluetoothHealthProfileHandler` interface (in [mobile](#) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothHealthProfileHandler) and [wearable](#) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothHealthProfileHandler) applications):

```
function onConnect(channel)
{
    console.log("Success");
    healthChannel = channel;
}

adapter.getDevice("35:F4:59:D1:7A:03", function(device)
{
    healthProfileHandler.connectToSource(device, healthApplication, onConnect);
});
```

When a connection between 2 devices is established, the success callback of the `connectToSource()` method is called. In addition, the `onconnect` event handler of the `BluetoothHealthApplication` instance (in [mobile](#) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothHealthApplication) and [wearable](#) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothHealthApplication) applications) is called, if the success callback attribute is set. You can get the connected `BluetoothHealthChannel` object (in [mobile](#) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothHealthChannel) and [wearable](#) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothHealthChannel) applications) from the callbacks.

5. To send data to the source device, use the `sendData()` method:

```
var dataToSend = [0, 0, 0];
var length = healthChannel.sendData(dataToSend);
```

The `onmessage` event handler in the `BluetoothHealthChannelChangeCallback` interface (in [mobile](#) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothHealthChannelChangeCallback) and [wearable](#) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothHealthChannelChangeCallback) applications) is called when the data is received, if you set a listener on the connected channel by using the `setListener()` method.

6. Disconnect from the health source device:

```
healthChannel.close();
```

When the channel is disconnected, the `onclose` event handler in the `BluetoothHealthChannelChangeCallback` interface is called.

Discovering Bluetooth Low Energy Devices

Learning how to search for remote devices is a basic Bluetooth management skill:

1. Define a scan event handler by implementing the `BluetoothLEScanCallback` callback (in [mobile](#) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEScanCallback) and [wearable](#) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEScanCallback) applications).

The callback is invoked when a remote device has been detected.

```
function successcallback(device)
{
    console.log("Found device: " + device.name + " [" + device.address + "]");
};
```

Note:

To allow other Bluetooth devices to find your device, you must set the device to be visible through the system settings.

2. Retrieve a `BluetoothLEAdapter` object (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEAdapter\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEAdapter) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEAdapter\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEAdapter) applications) with the `getLEAdapter()` method of the `BluetoothManager` interface (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothManager\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothManager) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothManager\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothManager) applications):

```
var adapter = tizen.bluetooth.getLEAdapter();
```

3. To search for remote devices, use the `startScan()` method of the `BluetoothLEAdapter` interface:

```
adapter.startScan(successcallback);
```

4. When you find the right remote device or the user cancels the scanning, disable the scan using the `stopScan()` method of the `BluetoothLEAdapter` interface:

```
adapter.stopScan();
```

Managing the Advertising Options

The Bluetooth Low Energy technology allows a device to broadcast some information without a connection between devices. The Bluetooth Low Energy API provides methods to control this advertising (broadcasting).

Learning how to control what information is advertised by the device is a useful Bluetooth Low Energy skill:

1. Retrieve a `BluetoothLEAdapter` object (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEAdapter\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEAdapter) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEAdapter\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEAdapter) applications) with the `getLEAdapter()` method of the `BluetoothManager` interface (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothManager\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothManager) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothManager\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothManager) applications):

```
var adapter = tizen.bluetooth.getLEAdapter();
```

2. Set up options and start advertising with the `startAdvertise()` method of the `BluetoothLEAdapter` interface:

```
var advertiseData = new tizen.BluetoothLEAdvertiseData(
{
  includeName: true,
  serviceuuids: ["180f"] /* 180f is 16bit Battery Service UUID */
});
var connectable = true;

adapter.startAdvertise(advertiseData, "ADVERTISE",
  function onstate(state)
  {
    console.log("Advertising configured: " + state);
  },
  function(error)
  {
    console.log("startAdvertise() failed: " + error.message);
  },
  "LOW_LATENCY", connectable);
```

Note:

To learn how to make your mobile device visible to other Bluetooth devices, see [Managing the Local Bluetooth Adapter](#).

3. To disable the advertising, use the `stopAdvertise()` method of the `BluetoothLEAdapter` interface:

```
adapter.stopAdvertise();
```


Connecting to a Bluetooth Low Energy Device

Learning how to connect to other devices is a basic Bluetooth Low Energy management skill:

1. Retrieve a **BluetoothLEAdapter** object (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEAdapter\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEAdapter) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEAdapter\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEAdapter) applications) with the **getLEAdapter()** method of the **BluetoothManager** interface (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothManager\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothManager) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothManager\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothManager) applications):

```
var adapter = tizen.bluetooth.getLEAdapter();
```

2. Define success and error callbacks for the connect operation:

```
function connectFail(error)
{
    console.log("Failed to connect to device: " + e.message);
}

function connectSuccess()
{
    console.log("Connected to device");
}
```

3. Define a callback for the scan operation that connects to a found device and stops the scan.

Within the callback request, establish a connection with the found device using the **connect()** method of the **BluetoothLEDevice** interface (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEDevice\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEDevice) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEDevice\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEDevice) applications):

```
var remoteDevice = null;

function onDeviceFound(device)
{
    if (remoteDevice === null)
    {
        remoteDevice = device;
        console.log("Found device " + device.name + ". Connecting...");

        device.connect(connectSuccess, connectFail);
    }

    adapter.stopScan();
}
```

4. When the callbacks are completed, initiate the Bluetooth Low Energy scan using the **startScan()** method of the **BluetoothLEAdapter** adapter:

```
adapter.startScan(onDeviceFound);
```

5. When the connection to the remote device is no longer required, disconnect from the device by calling the **disconnect()** method of the **BluetoothLEDevice** interface:

```
remoteDevice.disconnect();
```

Receiving Notifications on Connection State Changes

Learning how to receive notifications whenever the device connection is established or lost is a useful Bluetooth management skill:

1. Retrieve a **BluetoothLEAdapter** object (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEAdapter\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEAdapter) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEAdapter\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEAdapter) applications) with the **getLEAdapter()** method of the **BluetoothManager** interface (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothManager\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothManager) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothManager\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothManager) applications):

[guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothManager](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothManager)) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothManager\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothManager) applications):

```
var adapter = tizen.bluetooth.getLEAdapter();
```

2. Define a connection state change listener by implementing the `BluetoothLEConnectChangeCallback` callback (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEConnectChangeCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEConnectChangeCallback) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEConnectChangeCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEConnectChangeCallback) applications):

```
var connectionListener =
{
  onconnected: function(device)
  {
    console.log("Connected to the device: " + device.name + " [" + device.address + "]");
  },
  ondisconnected: function(device)
  {
    console.log("Disconnected from the device " + device.name + " [" + device.address + "]");
  }
};
```

3. Define a callback for the scan operation that connects to a found device and stops the scan.

Within the callback, register a connection state change listener using the `addConnectStateChangeListener()` method of the `BluetoothLEDevice` interface (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEDevice\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEDevice) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEDevice\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEDevice) applications):

```
var remoteDevice = null;
var watchId;

function onDeviceFound(device)
{
  if (remoteDevice === null)
  {
    remoteDevice = device;
    console.log("Found device " + device.name + ". Connecting...");

    watchId = remoteDevice.addConnectStateChangeListener(connectionListener);

    remoteDevice.connect();
  }

  adapter.stopScan();
}
```

4. When the callbacks are completed, initiate the Bluetooth Low Energy scan:

```
adapter.startScan(onDeviceFound);
```

5. When the notifications about the connection are no longer required, unregister the listener from the device by calling the `removeConnectStateChangeListener()` method of the `BluetoothLEDevice` interface:

```
remoteDevice.removeConnectStateChangeListener(watchId);
```

Retrieving Bluetooth GATT Services

Learning how to retrieve a list of GATT services (Generic Attribute) provided by a remote device is basic Bluetooth Low Energy management skill:

1. [Connect to a Bluetooth Low Energy device.](#)

2. Define a connection state change listener by implementing the `BluetoothLEConnectChangeCallback` (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEConnectChangeCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEConnectChangeCallback) and [wearable](#)

(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEConnectChangeCallback) applications):

```
function showGATTService(service, indent)
{
    if (indent === undefined)
    {
        indent = "";
    }

    console.log(indent + "Service " + service.uuid + ". Has " + service.characteristics.length
        + " characteristics and " + service.services.length + " sub-services.");

    for (var i = 0; i < service.services.length; i++)
    {
        showGATTService(service.services[i], indent + "  ");
    }
}
```

3. Retrieve a list of GATT service UUIDs from the `uuids` attribute of the `BluetoothLEDevice` interface (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEDevice) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEDevice) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEDevice) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEDevice) applications):

```
var serviceUUIDs = remoteDevice.uuids;
```

4. Retrieve GATT service information using the `getService()` method of the `BluetoothLEDevice` interface for every service UUID:

```
var i = 0, service = null;

for (i; i < serviceUUIDs.length; i++)
{
    service = remoteDevice.getService(serviceUUIDs[i]);

    showGATTService(service);
}
```

Accessing the Bluetooth GATT Characteristic Value

Learning how to read and write a value of the Bluetooth characteristic is a useful Bluetooth Low Energy management skill:

1. [Connect to a Bluetooth Low Energy device.](#)
2. Retrieve a list of GATT service UUIDs from the `uuids` attribute of the `BluetoothLEDevice` interface (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEDevice) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEDevice) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEDevice) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEDevice) applications):

```
var serviceUUIDs = remoteDevice.uuids;
```

3. Select a GATT service and use the `getService()` method of the `BluetoothLEDevice` interface to retrieve an object representing the service. In this example, the first service is used:

```
var gattService = remoteDevice.getService(serviceUUIDs[0]);
```

4. Select an interesting characteristic from the `characteristics` attribute of the `BluetoothGATTService` interface (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothGATTService) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothGATTService) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothGATTService) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothGATTService) applications). In this example, the first characteristic is used:

```
var property = gattService.characteristics[0];
```

5. Define a callback implementing the `ReadValueSuccessCallback` callback (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#ReadValueSuccessCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#ReadValueSuccessCallback) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#ReadValueSuccessCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#ReadValueSuccessCallback) applications), which receives the value of the characteristic:

```
function readSuccess(value)
{
    console.log("Characteristic value: " + value);
}

function readFail(error)
{
    console.log("readValue() failed: " + error);
}
```

6. To retrieve the GATT characteristic value, use the `readValue()` method of the `BluetoothGATTCharacteristic` interface (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothGATTCharacteristic\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothGATTCharacteristic) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothGATTCharacteristic\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothGATTCharacteristic) applications):

```
if (!property.isReadable)
{
    console.log("Property seems not to be readable. Attempting to read...");
}
property.readValue(readSuccess, readFail);
```

7. To change the characteristic value, define callbacks and use the `writeValue()` method of the `BluetoothGATTCharacteristic` interface:

```
function writeSuccess(value)
{
    console.log("Written");
}

function writeFail(error)
{
    console.log("writeValue() failed: " + error);
}

if (!property.isWritable)
{
    console.log("Property seems not to be writable. Attempting to write...");
}
var newValue = [82];

property.writeValue(newValue, writeSuccess, writeFail);
```

Receiving Notifications on Characteristic Value Changes

Learning how to monitor a changes in a Bluetooth characteristic is a useful Bluetooth Low Energy management skill:

1. [Connect to a Bluetooth Low Energy device.](#)
2. Retrieve a list of GATT service UUIDs from the `uuids` attribute of the `BluetoothLEDevice` interface (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEDevice\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEDevice) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEDevice\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEDevice) applications):

```
var serviceUUIDs = remoteDevice.uuids;
```

3. Select a GATT service and use the `getService()` method of the `BluetoothLEDevice` interface to retrieve an object representing the service. In this example, the first service is used:

```
var gattService = remoteDevice.getService(serviceUUIDs[0]);
```

4. Select an interesting characteristic from the `characteristics` attribute of the `BluetoothGATTService` interface (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothGATTService) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothGATTService) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothGATTService) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothGATTService) applications). In this example, the first characteristic is used:

```
var property = gattService.characteristics[0];
```

5. Define a callback implementing the `ReadValueSuccessCallback` callback (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#ReadValueSuccessCallback) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#ReadValueSuccessCallback) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#ReadValueSuccessCallback) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#ReadValueSuccessCallback) applications), which receives the value of the characteristic every time the value changes:

```
function onValueChange(value)
{
    console.log("Characteristic value is now: " + value);
}
```

6. Register a value change listener using the `addValueChangeListener()` method of the `BluetoothGATTCharacteristic` interface (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothGATTCharacteristic) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothGATTCharacteristic) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothGATTCharacteristic) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothGATTCharacteristic) applications):

```
var watchId = property.addValueChangeListener(onValueChange);
```

7. When the notifications about the connection are no longer required, unregister the listener from the device by calling the `removeValueChangeListener()` method of the `BluetoothGATTCharacteristic` interface:

```
property.removeValueChangeListener(watchId);
```

Accessing the Bluetooth GATT Descriptor Value

Learning how to read and write a value of the Bluetooth descriptor is a useful Bluetooth Low Energy management skill:

1. [Connect to a Bluetooth Low Energy device.](#)
2. Retrieve a list of GATT service UUIDs from the `uuids` attribute of the `BluetoothLEDevice` interface (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEDevice) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothLEDevice) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEDevice) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothLEDevice) applications):

```
var serviceUUIDs = remoteDevice.uuids;
```

3. Select a GATT service and use the `getService()` method of the `BluetoothLEDevice` interface to retrieve an object representing the service. In this example, the first service is used:

```
var gattService = remoteDevice.getService(serviceUUIDs[0]);
```

4. Select an interesting characteristic from the `characteristics` attribute of the `BluetoothGATTService` interface (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothGATTService) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothGATTService) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothGATTService) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothGATTService) applications). In this example, the first characteristic is used:

```
var characteristic = gattService.characteristics[0];
```

5. Select an interesting descriptor from the `descriptors` attribute of the `BluetoothGATTCharacteristic` interface (in [mobile](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothGATTCharacteristic) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothGATTCharacteristic) and [wearable](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothGATTCharacteristic) (https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothGATTCharacteristic) applications). In this example, the first descriptor is used:

```
var descriptor = characteristic.descriptors[0];
```

6. Define a callback implementing the `ReadValueSuccessCallback` callback (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#ReadValueSuccessCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#ReadValueSuccessCallback) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#ReadValueSuccessCallback\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#ReadValueSuccessCallback) applications), which receives the value of the descriptor:

```
function readSuccess(value)
{
    console.log("Descriptor value: " + value);
}

function readFail(error)
{
    console.log("readValue() failed: " + error);
}
```

7. To retrieve the GATT descriptor value, use the `readValue()` method of the `BluetoothGATTDescriptor` interface (in [mobile \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothGATTDescriptor\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/mobile/tizen/bluetooth.html#BluetoothGATTDescriptor) and [wearable \(https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothGATTDescriptor\)](https://developer.tizen.org/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/bluetooth.html#BluetoothGATTDescriptor) applications):

```
descriptor.readValue(readSuccess, readFail);
```

8. To change the descriptor value, define callbacks and use the `writeValue()` method of the `BluetoothGATTDescriptor` interface:

```
function writeSuccess(value)
{
    console.log("Written");
}

function writeFail(error)
{
    console.log("writeValue() failed: " + error);
}

var newValue = [3];

descriptor.writeValue(newValue, writeSuccess, writeFail);
```

SDK Version Since: 2.4.0

[^ PREV](#) [Network \(/development/tutorials/web-application/tizen-features/network\)](#)

Tags: [tizen \(/category/tags/tizen\)](#) [bluetooth \(/category/tags/bluetooth\)](#) [web api \(/category/tags/web-api\)](#)

Was this document helpful?

We value your feedback. Please let us know what you think.

Yes

No