Bluetooth API

The Bluetooth API defines interfaces and methods to manage Bluetooth.

The following Bluetooth functionalities are provided:

- Controls local Bluetooth device, that is, turn Bluetooth on/off, etc.
- Sets visibility
- Discovers nearby Bluetooth devices (Device discovery, including Bluetooth LE devices)
- Gets bonded devices information
- Controls bonding
- Connects to a service on a remote device and exchanges data
- Registers a service (RFCOMM) on a local device, which can be consumed by remote devices to exchange data
- Advertise for remote devices (including Bluetooth LE devices)
- Act as a GATT client (Generic Attribute Profile client)

For more information on the Bluetooth features, see Bluetooth Guide (https://developer.tizen.org/development/guides/web-application/tizen-features/network/bluetooth).

Since: 2.3.1

Table of Contents

Summary of Interfaces and Methods

Interface	Method
BluetoothManagerObject	
BluetoothLEServiceData	
BluetoothLEManufacturerData	
BluetoothLEAdvertiseDataInit	
BluetoothLEAdvertiseData	
BluetoothManager	BluetoothAdapter getDefaultAdapter () BluetoothLEAdapter getLEAdapter ()

Interface	Method
BluetoothAdapter	void setName (DOMString name, optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) void setPowered (boolean state, optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) void setChangeListener (BluetoothAdapterChangeCallback listener) void unsetChangeListener () void discoverDevices (BluetoothDiscoverDevicesSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) void stopDiscovery (optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) void getKnownDevices (BluetoothDeviceArraySuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) void getDevice (BluetoothAddress address, BluetoothDeviceSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) void createBonding (BluetoothAddress address, BluetoothDeviceSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) void destroyBonding (BluetoothAddress address, optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) void registerRFCOMMServiceByUUID (BluetoothUUID uuid, DOMString name, BluetoothServiceSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) BluetoothProfileHandler getBluetoothProfileHandler (BluetoothProfileType profileType)
BluetoothLEAdapter	void startScan (BluetoothLEScanCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) void stopScan () void startAdvertise (BluetoothLEAdvertiseData advertiseData, BluetoothAdvertisePacketType packetType, BluetoothLEAdvertiseCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback, optional BluetoothAdvertisingMode? mode, optional boolean? connectable) void stopAdvertise ()
BluetoothGATTService	

Interface	Method
BluetoothGATTCharacteristic	void readValue (ReadValueSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) void writeValue (byte[] value, optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) long addValueChangeListener (ReadValueSuccessCallback callback) void removeValueChangeListener (long watchID)
BluetoothGATTDescriptor	void readValue (ReadValueSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) void writeValue (byte[] value, optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback)
BluetoothLEScanCallback	void onsuccess (BluetoothLEDevice device)
BluetoothLEAdvertiseCallback	void onstate (BluetoothAdvertisingState state)
BluetoothLEConnectChangeCallback	void onconnected (BluetoothLEDevice device) void ondisconnected (BluetoothLEDevice device)
ReadValueSuccessCallback	void onread (byte[] value)
BluetoothDevice	void connectToServiceByUUID (BluetoothUUID uuid, BluetoothSocketSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback)
BluetoothLEDevice	void connect (optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) void disconnect (optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) BluetoothGATTService getService (BluetoothUUID uuid) long addConnectStateChangeListener (BluetoothLEConnectChangeCallback listener) void removeConnectStateChangeListener (long watchID)
BluetoothSocket	unsigned long writeData (byte[] data) byte[] readData () void close ()
BluetoothClass	boolean hasService (unsigned short service)
BluetoothClassDeviceMajor	
BluetoothClassDeviceMinor	
BluetoothClassDeviceService	

Interface	Method
BluetoothServiceHandler	void unregister (optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback)
BluetoothProfileHandler	
BluetoothHealthProfileHandler	void registerSinkApplication (unsigned short dataType, DOMString name, BluetoothHealthApplicationSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) void connectToSource (BluetoothDevice peer, BluetoothHealthApplication application, BluetoothHealthChannelSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback)
BluetoothHealthApplication	void unregister (optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback)
BluetoothHealthChannel	void close () unsigned long sendData (byte[] data) void setListener (BluetoothHealthChannelChangeCallback listener) void unsetListener ()
BluetoothAdapterChangeCallback	void onstatechanged (boolean powered) void onnamechanged (DOMString name) void onvisibilitychanged (boolean visible)
BluetoothDeviceSuccessCallback	void onsuccess (BluetoothDevice device)
BluetoothDeviceArraySuccessCallback	void onsuccess (BluetoothDevice[] devices)
BluetoothDiscoverDevicesSuccessCallback	void onstarted () void ondevicefound (BluetoothDevice device) void ondevicedisappeared (BluetoothAddress address) void onfinished (BluetoothDevice[] foundDevices)
BluetoothSocketSuccessCallback	void onsuccess (BluetoothSocket socket)
BluetoothServiceSuccessCallback	void onsuccess (BluetoothServiceHandler handler)
BluetoothHealthApplicationSuccessCallback	void onsuccess (BluetoothHealthApplication application)
BluetoothHealthChannelSuccessCallback	void onsuccess (BluetoothHealthChannel channel)
BluetoothHealthChannelChangeCallback	void onmessage (byte[] data) void onclose ()

1. Type Definitions

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

1.1. BluetoothAddress

The address of a Bluetooth device.

```
typedef DOMString BluetoothAddress;
```

Since: 2.3.1

[(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)](/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

1.2. BluetoothUUID

The UUID of a Bluetooth service.

```
typedef DOMString BluetoothUUID;
```

Since: 2.3.1

[(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)](/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

1.3. BluetoothSocketState

The Bluetooth socket state.

```
enum BluetoothSocketState { "CLOSED", "OPEN" };
```

Since: 2.3.1

[(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)](/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

1.4. BluetoothProfileType

The Bluetooth profile.

```
enum BluetoothProfileType { "HEALTH" };
```

Since: 2.3.1

[(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)](/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

1.5. BluetoothHealthChannelType

The channel type of health device profile.

```
enum BluetoothHealthChannelType { "RELIABLE", "STREAMING" };
```

Since: 2.3.1

[(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)](/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

1.6. BluetoothLESolicitationUUID

The service solicitation UUID of the Bluetooth LE device.

```
typedef DOMString BluetoothLESolicitationUUID;
```

Since: 2.3.1

[(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)](/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

1.7. BluetoothAdvertisePacketType

The Bluetooth LE packet type.

```
enum BluetoothAdvertisePacketType { "ADVERTISE", "SCAN_RESPONSE" };
```

Since: 2.3.1

- ADVERTISE - The advertising packet

- SCAN_RESPONSE- The scan response packet

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

1.8. BluetoothAdvertisingState

The Bluetooth advertising state.

```
enum BluetoothAdvertisingState { "STARTED", "STOPPED" };
```

Since: 2.3.1

- STARTED - Advertising has started
- STOPPED - Advertising has stopped

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

1.9. BluetoothAdvertisingMode

The Bluetooth advertising mode.

```
enum BluetoothAdvertisingMode { "BALANCED", "LOW_LATENCY", "LOW_ENERGY" };
```

Since: 2.3.1

- BALANCED- Balanced advertising mode
- LOW_LATENCY- Low latency advertising mode
- LOW_ENERGY - Low energy advertising mode

2. Interfaces

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.1. BluetoothManagerObject

The BluetoothManagerObject interface defines what is instantiated by the Tizen object from the Tizen platform.

```
[NoInterfaceObject] interface BluetoothManagerObject {
  readonly attribute BluetoothManager bluetooth;
};
```

```
Tizen (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#Tizen) implements
BluetoothManagerObject;
```

Since: 2.3.1

The tizen.bluetooth object allows access to the Bluetooth API.

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.2. BluetoothLEServiceData

The BluetoothLEServiceData interface is a service specific data container of Bluetooth LE device.

```
[Constructor(DOMString uuid, DOMString data)]
 interface BluetoothLEServiceData {
   attribute BluetoothUUID uuid;
   attribute DOMString data;
 };
```

Since: 2.3.1

Constructors

```
BluetoothLEServiceData(DOMString uuid, DOMString data);
```

Attributes

- BluetoothUUID **uuid**
  The 16 bit UUID of service data
  Since: 2.3.1
- DOMString **data**
  The service data of the Bluetooth LE device
  Since: 2.3.1

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.3. BluetoothLEManufacturerData

The BluetoothLEManufacturerData interface is a manufacturer specific data container for an advertise or scan response data.

```
[Constructor(DOMString id, DOMString data)]
interface BluetoothLEManufacturerData {
  attribute DOMString id;
  attribute DOMString data;
};
```

Since: 2.3.1

Constructors

```
BluetoothLEManufacturerData(DOMString id, DOMString data);
```

Code example:

```
// Creates a manufacturerData.
var manufacture = new tizen.BluetoothLEManufacturerData("127", "0x0057");
```

Attributes

- DOMString **id**
  The manufacturer assigned ID
  Since: 2.3.1
- DOMString **data**
  The manufacturer data content
  Since: 2.3.1

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.4. BluetoothLEAdvertiseDataInit

Provides a dictionary for specifying advertise or scan response data.

```
dictionary BluetoothLEAdvertiseDataInit {
  boolean? includeName;
  BluetoothUUID[]? uuids;
  BluetoothLESolicitationUUID[]? solicitationuuids;
  unsigned long? appearance;
  boolean? includeTxPowerLevel;
  BluetoothLEServiceData? serviceData;
  BluetoothLEManufacturerData? manufacturerData;
};
```

Since: 2.3.1

This dictionary is used as an input parameter of the BluetoothLEAdvertiseData constructor.

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.5. BluetoothLEAdvertiseData

The BluetoothLEAdvertiseData interface is an advertise or scan response data container for Bluetooth advertising.

```
[Constructor(optional BluetoothLEAdvertiseDataInit? init)]
interface BluetoothLEAdvertiseData {
  attribute boolean? includeName;
  attribute BluetoothUUID[]? uuids;
  attribute BluetoothLESolicitationUUID[]? solicitationuuids;
  attribute unsigned long? appearance;
  attribute boolean? includeTxPowerLevel;
  attribute BluetoothLEServiceData? serviceData;
  attribute BluetoothLEManufacturerData? manufacturerData;
};
```

Since: 2.3.1

The BluetoothLEAdvertiseData container for Bluetooth LE advertising. This represents the data to be advertised as well as the scan response data for active scans.

Constructors

```
BluetoothLEAdvertiseData(optional BluetoothLEAdvertiseDataInit? init);
```

Attributes

- boolean **includeName** *[nullable]*
  The flag indicating whether the device name should be included in advertise or scan response data. If this attribute is set to null, the default value is set to false.
  Since: 2.3.1
- BluetoothUUID[] **uuids** *[nullable]*
  The service UUID for advertise or scan response data.
  Since: 2.3.1
- BluetoothLESolicitationUUID[] **solicitationuuids** *[nullable]*
  The service solicitation UUID for advertise or scan response data.
  Since: 2.3.1
- unsigned long **appearance** *[nullable]*
  The external appearance of this device for advertise or scan response data.
  See the list of appearance codes (https://developer.bluetooth.org/gatt/characteristics/Pages/CharacteristicViewer.aspx?u=org.bluetooth.characteristic.gap.appearance.xml) for sample values.
  Since: 2.3.1
- boolean **includeTxPowerLevel** *[nullable]*
  The flag indicating whether transmission power level should be included in advertise or scan response data. If attribute is set to null, the default value is set to false.
  Since: 2.3.1
- BluetoothLEServiceData[] **serviceData** *[nullable]*
  The service data for advertise or scan response data.
  Since: 2.3.1
- BluetoothLEManufacturerData **manufacturerData** *[nullable]*
  The manufacturer specific data for advertise or scan response data.
  Since: 2.3.1

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.6. BluetoothManager

The BluetoothManager interface provides access to the BluetoothAdapter object.

```
[NoInterfaceObject] interface BluetoothManager {
  readonly attribute BluetoothClassDeviceMajor deviceMajor;
  readonly attribute BluetoothClassDeviceMinor deviceMinor;
  readonly attribute BluetoothClassDeviceService deviceService;
  BluetoothAdapter getDefaultAdapter() raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

  BluetoothLEAdapter getLEAdapter() raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
};
```

Since: 2.3.1

Attributes

- **readonly** BluetoothClassDeviceMajor **deviceMajor**
  The major device class identifier of Bluetooth class of device (CoD).
  Since: 2.3.1
- **readonly** BluetoothClassDeviceMinor **deviceMinor**
  The minor device class identifier of Bluetooth class of device (CoD).
  Since: 2.3.1
- **readonly** BluetoothClassDeviceService **deviceService**
  The major service class identifier of Bluetooth class of device (CoD).
  Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *getDefaultAdapter*

Gets the default local Bluetooth adapter.

```
BluetoothAdapter getDefaultAdapter();
```

Since: 2.3.1

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.gap (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Return value:
BluetoothAdapter The local BluetoothAdapter object.

Exceptions:

- WebAPIException
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type UnknownError, if any other error occurs.

Code example:

```
try {
   var adapter = tizen.bluetooth.getDefaultAdapter() ;
} catch (err) {
   console.log (err.name +": " + err.message);
}
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *getLEAdapter*

Gets the default Low Energy Bluetooth adapter.

```
BluetoothLEAdapter getLEAdapter();
```

Since: 2.3.1

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.admin (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Remark : To check if this method is supported or not, use tizen.systeminfo.getCapability (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/systeminfo.html#SystemInfo::getCapability)

("http://tizen.org/feature/network.bluetooth.le").

Return value:
BluetoothLEAdapter The local BluetoothLEAdapter object.

Exceptions:

- WebAPIException
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type NotSupportedError, if the feature is not supported.
  - with error type UnknownError, if any other error occurs.

Code example:

```
try {
   var adapter = tizen.bluetooth.getLEAdapter();
} catch (err) {
   console.log(err.name +": " + err.message);
}
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.7. BluetoothAdapter

The BluetoothAdapter interface provides access to control the device's Bluetooth adapter.

```
[NoInterfaceObject] interface BluetoothAdapter {
   readonly attribute DOMString name;
   readonly attribute BluetoothAddress address;
   readonly attribute boolean powered;
   readonly attribute boolean visible;

   void setName(DOMString name,
            optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback,
            optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

   void setPowered(boolean state,
             optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback,
             optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

   void setChangeListener(BluetoothAdapterChangeCallback listener) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

   void unsetChangeListener() raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

   void discoverDevices(BluetoothDiscoverDevicesSuccessCallback successCallback,
              optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) raises(WebAPIException (?
```

Since: 2.3.1

This interface offers methods to control local Bluetooth behavior, such as:

- Turning on/off Bluetooth radio
- Changing device visibility
- Scanning for remote devices
- Accessing known devices
- Registering a service in the device service database

Attributes

- **readonly** DOMString **name**
  The readable name of the Bluetooth adapter.
  Since: 2.3.1

  Code example:

  ```
  // Access adapter name
  var adapter = tizen.bluetooth.getDefaultAdapter();
  console.log ("Bluetooth adapter name: " + adapter.name);
  ```

- **readonly** BluetoothAddress **address**

The unique hardware address of the Bluetooth adapter, also known as the MAC address.
Since: 2.3.1

Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
console.log("Bluetooth device address: " + adapter.address);
```

- **readonly** boolean **powered**
  The current state of the Bluetooth adapter.
  This attribute holds one of the following 2 values:
  - *true* - If Bluetooth adapter is currently on
  - *false* - If Bluetooth adapter is currently off
  Since: 2.3.1

  Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
console.log("Bluetooth state: " + (adapter.powered ? "On" : "Off"));
```

- **readonly** boolean **visible**
  The current visibility state of the Bluetooth adapter, that is, whether the local device is discoverable by remote devices.
  Since: 2.3.1

  Code example:

```
// Queries the current visible state
var adapter = tizen.bluetooth.getDefaultAdapter();
console.log ("Bluetooth Visibility: " + (adapter.visible ? "On" : "Off"));
```

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*setName*

Sets the local Bluetooth adapter name.

```
void setName(DOMString name, optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

Sends a request to Bluetooth hardware to change the name of the local Bluetooth adapter to name.

The ErrorCallback is launched with these error types:

- InvalidValuesError: If any of the input parameters contain an invalid value.
- ServiceNotAvailableError: If a Bluetooth device is turned off.
- UnknownError: In any other error case.
- NotSupportedError: If a device doesn't allow a Tizen Web application to change the name of the local Bluetooth adapter.

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.admin (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Remark : To check if this method is supported or not, use *tizen.systeminfo.getCapability("http://tizen.org/capability/network.bluetooth.always_on")*

Parameters:

- name: Name to set
- successCallback *[optional] [nullable]*: Callback function that is called when the asynchronous call completes successfully
- errorCallback *[optional] [nullable]*: Callback function that is called when an error occurs

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if any of the input parameter is not compatible with the expected type for that parameter.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

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

var bt_always_on = tizen.systeminfo.getCapability("http://tizen.org/capability/network.bluetooth.always_on");

function changeName(newName) {
  if (adapter.name != newName) {
    // initiate change name
    adapter.setName(newName, function() {
      console.log("Adapter name changed to " + adapter.name);
    },
    function(e) {
      console.log("Failed to change name: " + e.message);
    });
  }
}
if (bt_always_on === false) {
  changeName("myDevice");
}
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*setPowered*

Sets the state of a Bluetooth adapter to on or off by sending a request to Bluetooth hardware to change the power state. For most Bluetooth actions, the Bluetooth adapter must be powered on.

Deprecated. *Deprecated since 2.3. Instead, let the user turn on/off Bluetooth through the Settings application. see the Bluetooth Tutorial (https://developer.tizen.org/development/tutorials/web-application/tizen-features/network/bluetooth).*

void setPowered(boolean state, optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)**? successCallback, optional** ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)**? errorCallback);**

Since: 2.3.1

The ErrorCallback is launched with these error names:

- ServiceNotAvailableError - If a Bluetooth device is busy.
- NotSupportedError - If a device does not allow a Tizen Web application to turn on or off the Bluetooth service on a device. Bluetooth must be enabled on the device which the remote Bluetooth devices always work with.
- UnknownError - If any other error occurs.

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.admin (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Remark : To check if this method is supported or not, use this.systeminfo.getCapability (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/systeminfo.html#SystemInfo::getCapability)("http://tizen.org/capability/network.bluetooth.always_on" (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/systeminfo_capability_keys.html#network)).

Parameters:

- state: State to set: *true* to power on Bluetooth, *false* to power it off
- successCallback *[optional] [nullable]*: Callback function that is called on successful Bluetooth activation/deactivation
- errorCallback *[optional] [nullable]*: Callback function that is called on failure of a Bluetooth activation/deactivation

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if any of the input parameter is not compatible with the expected type for that parameter.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

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

function powerOn() {
  // If adapter is not powered on
  if (!adapter.powered) {
```

```
        // Initiates power on
        adapter.setPowered(true, function() {
          console.log("Bluetooth powered on success.");
        },
        function(e) {
          console.log("Failed to power on Bluetooth: " + e.message);
        });
    }
}

function powerOff() {
    // If powered on
    if (adapter.powered) {
        // Initiates power off
        adapter.setPowered(false, function() {
          console.log("Bluetooth powered off successfully.");
        },
        function(e) {
          console.log("Failed to power off Bluetooth: " + e.message);
        });
    }
}

var bt_always_on = tizen.systeminfo.getCapability("http://tizen.org/capability/network.bluetooth.always_on");
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*setChangeListener*

Sets the listener to receive notifications about changes of Bluetooth adapter.

```
void setChangeListener(BluetoothAdapterChangeCallback listener);
```

Since: 2.3.1

Parameters:

- listener: The Bluetooth Adapter event listener to set

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type.
  - with error type SecurityError, if this functionality is not allowed.
  - with error type UnknownError, if any other error occurs.

Code example:

```
var changeListener = {
    onstatechanged: function(powered) {
        console.log ("Power state is changed into: " + powered);
    },
    onnamechanged: function(name) {
        console.log("Name is changed to: " + name);
    },
    onvisibilitychanged: function(visible) {
        console.log("Visibility is changed into: " + visible);
    }
};

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

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*unsetChangeListener*

Unsets the listener to stop receiving notifications about changes of Bluetooth adapter.

```
void unsetChangeListener();
```

Since: 2.3.1

Exceptions:

- WebAPIException
  - with error type SecurityError, if this functionality is not allowed.
  - with error type UnknownError, if any other error occurs.

Code example:

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

var changeListener = {
    onstatechanged: function(powered) {
        console.log ("Power state is changed into: " + powered);
        if (!powered)
            adapter.unsetChangeListener();
    },
    onnamechanged: function(name) {
        console.log("Name is changed to: " + name);
    },
    onvisibilitychanged: function(visible) {
        console.log("Visibility is changed into: " + visible);
    }
};

adapter.setChangeListener(changeListener);
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *discoverDevices*

Discovers nearby Bluetooth devices if any, that is, devices within proximity to the local device.

```
void discoverDevices(BluetoothDiscoverDevicesSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

This method initiates the device discovery process. Depending on the progress of this process the following methods are invoked:

- BluetoothDiscoverDevicesSuccessCallback.onstarted() - when a discovery process starts successfully.
- BluetoothDiscoverDevicesSuccessCallback.ondevicefound() - when any device is found in the process and this method is invoked with the device information. If no device is found, this method will never be invoked.
- BluetoothDiscoverDevicesSuccessCallback.ondevicedisappeared() - when a device goes out of proximity and this method is invoked with the address of the device.
- BluetoothDiscoverDevicesSuccessCallback.onfinished() - when a discovery process is completed.

A discovery process can be canceled anytime, by calling stopDiscovery() on the BluetoothAdapter.

The ErrorCallback is launched with these error types:

- ServiceNotAvailableError - If a Bluetooth device is turned off
- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.gap (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- successCallback: Callback function that is called when an asynchronous call completes successfully
- errorCallback *[optional] [nullable]*: Callback function that is called when an error occurs

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if any of the input parameter is not compatible with the expected type for that parameter.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

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

function startDiscovery() {

 var discoverDevicesSuccessCallback = {
```

```
    onstarted: function() {
      console.log ("Device discovery started...");
    },
    ondevicefound: function(device) {
      console.log("Found device - name: " + device.name + ", Address: "+ device.address);
    },
    ondevicedisappeared: function(address) {
      console.log("Device disappeared: " + address);
    },
    onfinished: function(devices) {
      console.log("Found Devices");
      for (var i = 0; i < devices.length; i++) {
        console.log("Name: " + devices[i].name + ", Address: " + devices[i].address);
      }
      console.log("Total: " + devices.length);
    }
  };

  // Starts searching for nearby devices, for about 12 sec.
  adapter.discoverDevices(discoverDevicesSuccessCallback, function(e) {
    console.log ("Failed to search devices: " + e.message + "(" + e.name + ")");
  });
}
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*stopDiscovery*

Stops an active device discovery session.

```
void stopDiscovery(optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

Device discovery is a heavyweight procedure, hence we recommend stopping discovery as soon as the required device is found. This method cancels an active discovery session.

The ErrorCallback is launched with these error types:

- ServiceNotAvailableError - If a Bluetooth device is turned off
- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.gap (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- successCallback *[optional] [nullable]*: Callback function to invoke when an asynchronous call completes successfully
- errorCallback *[optional] [nullable]*: Callback function to invoke when an error occurs

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if any of the input parameter is not compatible with the expected type for that parameter.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

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

// Calls this method whenever user finds one of the device.
function cancelDiscovery() {
  adapter.stopDiscovery(function() {
    console.log("Stop discovery success.");
  },
  function (e) {
    console.log("Error while stopDiscovery:" + e.message);
  });
}

function startDiscovery() {
```

```
var discoverDevicesSuccessCallback = {
    onstarted: function() {
        console.log ("Device discovery started...") ;
    },
    ondevicefound: function(device) {
        console.log("Found device - name: " + device.name + ", Address: "+ device.address);
        // Shows the device to user to check if this is the device user is looking for.
        // For example, add this to list view.

        cancelDiscovery();
    },
    ondevicedisappeared: function(address) {
        console.log("Device disappeared: " + address);
        // Removes from list, as it is no longer valid.
    },
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *getKnownDevices*

Gets all the known devices that have information stored in the local Bluetooth adapter.

```
void getKnownDevices(BluetoothDeviceArraySuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

A known device is one of the following:

- a bonded device
- a device found in last inquiry process

On success, it returns the list of currently known devices through BluetoothDeviceArraySuccessCallback.

The ErrorCallback is launched with these error types:

- ServiceNotAvailableError - If a Bluetooth device is turned off
- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.gap (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- successCallback: Callback function to invoke at retrieval of a list of Bluetooth devices that were bonded (paired) to the local Bluetooth adapter
- errorCallback *[optional] [nullable]*: Callback function to invoke in case of failure in finding bonded devices

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type for that parameter.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

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

function onGotDevices(devices) {
  console.log("Devices");
  for (var i = 0; i < devices.length; i++) {
    console.log(" Name: " + devices[i].name + ", Address: " + devices[i].address);
  }
  console.log("Total: " + devices.length);
}

function onError(e) {
 console.log ("Error: " + e.message);
}

function onBluetoothsetPowered() {
  adapter.getKnownDevices(onGotDevices, onError);
}

// Turns on Bluetooth
adapter.setPowered(true, onBluetoothsetPowered);
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *getDevice*

Gets the BluetoothDevice object for a given device hardware address.

```
void getDevice(BluetoothAddress address, BluetoothDeviceSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

This method returns device information stored in the local Bluetooth adapter for the specified device address through BluetoothDeviceSuccessCallback. A valid hardware address must be passed, such as "35:F4:59:D1:7A:03".

The ErrorCallback is launched with these error types:

- NotFoundError - If there is no device with the given address
- ServiceNotAvailableError - If a Bluetooth device is turned off
- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.gap (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- address: Address of a remote Bluetooth device to get
- successCallback: Callback function that is called when an asynchronous call completes successfully
- errorCallback *[optional] [nullable]*: Callback function that is called when an error occurs

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type for that parameter.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

```
function gotDeviceInfo(device) {
  console.log("Device Name: " + device.name);
  console.log("Device Address: " + device.address);
  console.log("Device Class: " + device.deviceClass.major);
  console.log("Is Bonded: " + (device.isBonded ? "Yes" : "No"));
}

function onError(e) {
  console.log ("Could not get device info:" + e.message);
}

var adapter = tizen.bluetooth.getDefaultAdapter();
adapter.getDevice("35:F4:59:D1:7A:03", gotDeviceInfo, onError);
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *createBonding*

Creates a bond with a remote device by initiating the bonding process with peer device, using the given MAC address. The remote device must be bonded with the local device in order to connect to services of the remote device and then exchange data with each other.

```
void createBonding(BluetoothAddress address, BluetoothDeviceSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

If the bonding process is successful, the device information is sent in successCallback.

The ErrorCallback is launched with these error types:

- NotFoundError - If there is no device with the given address
- ServiceNotAvailableError - If a Bluetooth device is turned off
- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.gap (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- address: MAC address of remote Bluetooth address to bond with
- successCallback: Callback function that is called when an asynchronous call completes successfully
- errorCallback *[optional] [nullable]*: Callback function that is called when an error occurs

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type for that parameter.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

```
function onBondingSuccess(device) {
  console.log("Device Name:" + device.name);
  console.log("Device Address:" + device.address);
  console.log("Device Service UUIDs:" + device.uuids.join("
"));
}

function onError(e) {
  console.log ("Could not create bonding, reason:" + e.message);
}

var adapter = tizen.bluetooth.getDefaultAdapter();
adapter.createBonding("35:F4:59:D1:7A:03", onBondingSuccess, onError);
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *destroyBonding*

Destroys the bond with a remote device.

```
void destroyBonding(BluetoothAddress address, optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

This method initiates the process of removing the specified address from the list of bonded devices.

The ErrorCallback is launched with these error types:

- NotFoundError - If there is no device with the given address
- ServiceNotAvailableError - If a Bluetooth device is turned off
- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.gap (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- address: Address of a bonded device
- successCallback *[optional] [nullable]*: Callback function that is called when an asynchronous call completes successfully
- errorCallback *[optional] [nullable]*: Callback function that is called when an error occurs

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type for that parameter.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

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

function gotDevice(device) {
  if (device.isBonded) {
    // Initiates destroying bonding
    adapter.destroyBonding(device.address, function() {
       console.log("Succeeded to destroy the bond success with:" + device.address);
     },
     function(e) {
       console.log("Failed to destroy the bond with " + device.address + ", reason: " + e.message);
```

```
    });
  }
}

var deviceAddress = "35:F4:59:D1:7A:03";
adapter.getDevice(deviceAddress, gotDevice, function(e) {
  console.log("Failed to get device info for " + deviceAddress + ", reason: " + e.message);
});
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*registerRFCOMMServiceByUUID*

Registers a service record in the device service record database with the specified uuid, name.

```
void registerRFCOMMServiceByUUID(BluetoothUUID uuid, DOMString name, BluetoothServiceSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

On success of the service registration, it returns a BluetoothServiceHandler object as the first parameter of successCallback, and listens for client connections. The service handler can be used to be notified on client connections or to unregister the service. User interaction is mandatory to connect to a registered service.

If any client(remote device) connects to this service, then BluetoothServiceHandler.onconnect() is invoked with BluetoothSocket object.

BluetoothServiceHandler.unregister() can be used to unregister the service record from the device service database and stop listening for client connections.

The ErrorCallback is launched with these error types:

- InvalidValuesError - If any of the input parameters contain an invalid value
- ServiceNotAvailableError - If a Bluetooth device is turned off
- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.spp (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- uuid: UUID of the new service, to which clients connect
- name: Name of the service
- successCallback: Callback function that is called on successful service registration
- errorCallback *[optional] [nullable]*: Callback function that is called when an error occurs

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type for that parameter.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
// Holds currently registered service record
var chatServiceHandler = null;
// Holds currently open socket
var serviceSocket = null;

function chatServiceSuccessCb(recordHandler) {
  console.log("Chat service registration succeeds!");
  chatServiceHandler = recordHandler;
  recordHandler.onconnect = function(socket) {
    console.log("Client connected: " + socket.peer.name + "," + socket.peer.address);
    serviceSocket = socket;
    // Messages received from remote device
    socket.onmessage = function() {
      var data = socket.readData();
      // Handles message code goes here

    };

    socket.onclose = function() {
      console.log('The socket is closed.');
      serviceSocket = null;
    };
```

```
  };
};

function publishChatService() {
  var CHAT_SERVICE_UUID = "5BCE9431-6C75-32AB-AFE0-2EC108A30860";
  adapter.registerRFCOMMServiceByUUID(CHAT_SERVICE_UUID, "Chat service", chatServiceSuccessCb,
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *getBluetoothProfileHandler*

Gets the profile handler for the given type.

```
BluetoothProfileHandler getBluetoothProfileHandler(BluetoothProfileType profileType);
```

Since: 2.3.1

Remark : To check if *HEALTH* type is supported or not, use *tizen.systeminfo.getCapability("http://tizen.org/feature/network.bluetooth.health")*

Parameters:

- profileType: Bluetooth Profile type

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type for that parameter.
  - with error type NotSupportedError, if the given profileType is not supported on a device.
  - with error type UnknownError, if any other error occurs.

Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
var healthProfileHandler;
var healthCapability = tizen.systeminfo.getCapability("http://tizen.org/feature/network.bluetooth.health");

if (healthCapability) {
  healthProfileHandler = adapter.getBluetoothProfileHandler("HEALTH");
} else {
  console.log("Bluetooth Health Profile is not supported on this device.");
}
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.8. BluetoothLEAdapter

The BluetoothLEAdapter interface provides access to control the device's Bluetooth Low Energy adapter.

```
[NoInterfaceObject] interface BluetoothLEAdapter {
  void startScan(BluetoothLEScanCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
  void stopScan() raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
  void startAdvertise(BluetoothLEAdvertiseData advertiseData, BluetoothAdvertisePacketType packetType, BluetoothLEAdvertiseCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback, optional BluetoothAdvertisingMode? mode, optional boolean? connectable) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
  void stopAdvertise() raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
};
```

Since: 2.3.1

This interface offers methods to control local Bluetooth Low Energy behavior, such as:

- Scanning and Advertising for remote devices

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *startScan*

Starts scanning for Low Energy advertisement.

```
void startScan(BluetoothLEScanCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

The scanning process can be canceled anytime, by calling the stopScan() method in the BluetoothLEAdapter interface.

The ErrorCallback will be launched in the following situations:

- ServiceNotAvailableError - If a Bluetooth device is turned off
- UnknownError - If any other error occurs.

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.admin (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- successCallback: Callback method that is called when advertisement is found.
- errorCallback *[optional] [nullable]*: Callback method that is called when an error occurs.

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if any input attribute is not compatible with the expected type for this attribute.
  - with error type InvalidStateError, if device is currently in progress of scanning, if the local Bluetooth le adapter is currently not enabled.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

```
var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
  onsuccess: function(device) {
    console.log("[Found device] address: " + device.address);
  }
});
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*stopScan*

Stops scanning for Low Energy advertisement.

```
void stopScan();
```

Since: 2.3.1

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.admin (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Exceptions:

- WebAPIException
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type UnknownError, if any other error occurs

Code example:

```
var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
  onsuccess: function(device) {
    if (device.address == "11:22:33:44:55:66") {
      console.log("Found device: " + device.name);
      adapter.stopScan();
    }
  }
});
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *startAdvertise*

Starts advertising for Low Energy Devices.

void startAdvertise(BluetoothLEAdvertiseData **advertiseData,** BluetoothAdvertisePacketType **packetType,** BluetoothLEAdvertiseCallback **successCallback, optional** ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)**? errorCallback, optional** BluetoothAdvertisingMode**? mode, optional boolean? connectable);**

Since: 2.3.1

An advertising process can be canceled anytime, by calling stopAdvertise() on the BluetoothLEAdapter.

The ErrorCallback will be launched in the following situations:

- ServiceNotAvailableError - If a Bluetooth device is turned off
- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.admin (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- advertiseData: BluetoothLEAdvertiseData object to be added
- packetType: The bluetooth LE packet type
- successCallback: Callback function that is called when advertise is started successfully.
- errorCallback *[optional] [nullable]*: Callback function that is called when an error occurs.
- mode *[optional] [nullable]*: The power and latency mode of advertising. The default mode is "BALANCED".
- connectable *[optional] [nullable]*: The connectable status. It's *true* if the advertisement is connectable. The default value of the parameter is *true*.

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if any input attribute is not compatible with the expected type for this attribute.
  - with error type QuotaExceededError, if any input attribute is not compatible with the maximum data size for this attribute.
  - with error type InvalidStateError, if device is currently in progress of advertising, if the local Bluetooth le adapter is currently not enabled.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

```
var adapter = tizen.bluetooth.getLEAdapter();
var battery_svc_uuid_16 = "180f"; // the service UUID, 16-bit UUID or 128-bit UUID is supported. (e.g. 180F, 0000180F-0000-1000-8000-00805F9B34FB)
var heart_rate_svc_uuid_16 = "180d"; // the service solicitation UUID, 16-bit UUID or 128-bit UUID is supported. (e.g. 180F, 0000180F-0000-1000-8000-
00805F9B34FB)
var advertiseOptions = {
  includeName: true,        // Whether the device name should be included
  includeTxPowerLevel: true,   // Whether the transmission power level should be included
  appearance: 192,    // The external appearance of device, 192 - Generic Watch
  uuids: [battery_svc_uuid_16],
  solicitationuuids: [heart_rate_svc_uuid_16]
};

var advertiseData = new tizen.BluetoothLEAdvertiseData(advertiseOptions);
var connectable = true;

adapter.startAdvertise(
  advertiseData,
  "ADVERTISE",
  function onstate(state) {
    console.log("Advertiser state: " + state);
  },
  function(e) {
    console.log("Failed to startAdvertise : " + e.message);
  },
  "LOW_LATENCY",
  connectable);
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *stopAdvertise*

Stops advertising for Low Energy Devices.

```
void stopAdvertise();
```

Since: 2.3.1

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.admin (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Exceptions:

- WebAPIException
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type UnknownError, if any other error occurs

Code example:

```
var adapter = tizen.bluetooth.getLEAdapter();
var battery_svc_uuid_16 = "180f"; // the service UUID, 16-bit UUID or 128-bit UUID is supported. (e.g. 180F, 0000180F-0000-1000-8000-00805F9B34FB)
var heart_rate_svc_uuid_16 = "180d"; // the service solicitation UUID, 16-bit UUID or 128-bit UUID is supported. (e.g. 180F, 0000180F-0000-1000-8000-00805F9B34FB)
var advertiseOptions = {
  includeName: true,        // Whether the device name should be included
  includeTxPowerLevel: true,   // Whether the transmission power level should be included
  appearance: 192,    // The external appearance of device, 192 - Generic Watch
  uuids: [battery_svc_uuid_16],
  solicitationuuids: [heart_rate_svc_uuid_16]
};

var advertiseData = new tizen.BluetoothLEAdvertiseData(advertiseOptions);

adapter.startAdvertise(advertiseData, "ADVERTISE",
  function onstate(state) {
    console.log("Advertiser state: " + state);
  });
adapter.stopAdvertise();
```

[(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)](/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.9. BluetoothGATTService

Bluetooth Low Energy Service. The service can be retrieved with BluetoothLEDevice.getService().

```
[NoInterfaceObject] interface BluetoothGATTService {
  readonly attribute BluetoothUUID uuid;
  readonly attribute BluetoothGATTService[] services;
  readonly attribute BluetoothGATTCharacteristic[] characteristics;
};
```

Since: 2.3.1

Attributes

- **readonly** BluetoothUUID **uuid**
  UUID of the service.
  Since: 2.3.1
- **readonly** BluetoothGATTService[] **services**
  A list of services included in this service.
  Since: 2.3.1
- **readonly** BluetoothGATTCharacteristic[] **characteristics**
  A list of characteristics in this service.
  Since: 2.3.1

[(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)](/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.10. BluetoothGATTCharacteristic

A characteristic provided by Bluetooth Low Energy service.

```
[NoInterfaceObject] interface BluetoothGATTCharacteristic {
  readonly attribute BluetoothGATTDescriptor[] descriptors;
  readonly attribute boolean isBroadcast;
  readonly attribute boolean hasExtendedProperties;
  readonly attribute boolean isNotify;
  readonly attribute boolean isIndication;
  readonly attribute boolean isReadable;
  readonly attribute boolean isSignedWrite;
  readonly attribute boolean isWritable;
  readonly attribute boolean isWriteNoResponse;
  void readValue(ReadValueSuccessCallback successCallback,
        optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
  void writeValue(byte[] value, optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback,
        optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

  long addValueChangeListener(ReadValueSuccessCallback callback) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

  void removeValueChangeListener(long watchID);
};
```

Since: 2.3.1

Attributes

- **readonly** BluetoothGATTDescriptor[] **descriptors**
  A list of descriptors in this characteristic.
  Since: 2.3.1
- **readonly** boolean **isBroadcast**
  Indicates if the characteristic is broadcastable.
  Since: 2.3.1
- **readonly** boolean **hasExtendedProperties**
  Indicates if the characteristic has extended properties.
  Since: 2.3.1
- **readonly** boolean **isNotify**
  Indicates if the characteristic supports notification.
  Since: 2.3.1
- **readonly** boolean **isIndication**
  Indicates if the characteristic supports indication.
  Since: 2.3.1
- **readonly** boolean **isReadable**
  Indicates if the characteristic is readable.
  Since: 2.3.1
- **readonly** boolean **isSignedWrite**
  Indicates if the characteristic supports write with the signature.
  Since: 2.3.1
- **readonly** boolean **isWritable**
  Indicates if the characteristic is writable.
  Since: 2.3.1
- **readonly** boolean **isWriteNoResponse**
  Indicates if the characteristic supports writing without response.
  Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*readValue*

Reads the characteristic value from the remote device. Updates characteristic value attribute.

```
void readValue(ReadValueSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

The ErrorCallback is launched with these error types:

- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.admin (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- successCallback: Callback function that is called when the characteristic value is read successfully
- errorCallback *[optional] [nullable]*: Callback function that is called in case of failure

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type for that parameter.
  - with error type InvalidStateError, if the local Bluetooth le adapter is currently not enabled.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

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

function proccessDevice(device) {
   device.connect(onConnected, onerror);
   function onConnected() {
      var service = device.getService(device.uuids[0]);
      if (service.characteristics.length > 0) {
         var characteristic = service.characteristics[0];
         characteristic.readValue(function(val) {
            console.log("Value read: " + val);
            device.disconnect();
         });
      }
   }
}

var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
   onsuccess: function(device) {
      if (device.address == "11:22:33:44:55:66") {
         console.log("Found device: " + device.name);
         proccessDevice(device);
      }
   }
});
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *writeValue*

Writes the characteristic value to the remote device.

```
void writeValue(byte[] value, optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

The ErrorCallback is launched with these error types:

- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.admin (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- value: The characteristic value to write
- successCallback *[optional] [nullable]*: Callback function that is called when the characteristic value is written successfully
- errorCallback *[optional] [nullable]*: Callback function that is called in case of failure

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type for that parameter.
  - with error type InvalidStateError, if the local Bluetooth le adapter is currently not enabled.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

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

function proccessDevice(device) {
  device.connect(onConnected, onerror);
  function onConnected() {
    var service = device.getService(device.uuids[0]);
    if (service.characteristics.length > 0) {
      var characteristic = service.characteristics[0];
      var data = new Array(1, 2, 3, 4, 5, 6);
      characteristic.writeValue(data, function() {
        console.log("Value written");
        device.disconnect();
      }, function(e) {
        console.log("Failed to write: " + e.message);
      });
    }
  }
}

var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
  onsuccess: function(device) {
    if (device.address == "11:22:33:44:55:66") {
      console.log("Found device: " + device.name);
      proccessDevice(device);
    }
  }
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*addValueChangeListener*

Registers a callback to be called when characteristic value of the characteristic changes.

```
long addValueChangeListener(ReadValueSuccessCallback callback);
```

Since: 2.3.1

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.admin (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- callback: Listener function that is called when the connection state changes

Return value:

long The watchID to be used to unregister the listener

Exceptions:

- WebAPIException
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

```
function proccessDevice(device) {

  function onConnected() {
    var service = device.getService("5BCE9431-6C75-32AB-AFE0-2EC108A30860");
    if (service.characteristics.length > 0) {
      var characteristic = service.characteristics[0];
      var watchID;

      watchID = characteristic.addValueChangeListener(function(value) {
        console.log("Characteristic value changed: " + value);
        characteristic.removeValueChangeListener(watchID);
        device.disconnect();
      });
    }
  }

  device.connect(onConnected);
}

var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
  onsuccess: function(device) {
    if (device.address == "11:22:33:44:55:66") {
      console.log("Found device: " + device.name);
      adapter.stopScan();
      proccessDevice(device);
    }
  }
});
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *removeValueChangeListener*

Unregisters a characteristic value change listener.

```
void removeValueChangeListener(long watchID);
```

Since: 2.3.1

Parameters:

- watchID: The watchID identifier returned by the addValueChangeListener() method.

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.11. BluetoothGATTDescriptor

Bluetooth Low Energy Descriptor.

```
[NoInterfaceObject] interface BluetoothGATTDescriptor {
  void readValue(ReadValueSuccessCallback successCallback,
       optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
  void writeValue(byte[] value, optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback,
       optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
};
```

Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *readValue*

Reads descriptor value from remote device. Updates descriptor value attribute.

void readValue(ReadValueSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);

Since: 2.3.1

The ErrorCallback is launched with these error types:

- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.admin (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- successCallback: Callback function that is called when the descriptor value is read successfully
- errorCallback *[optional] [nullable]*: Callback function that is called in case of failure

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type for that parameter.
  - with error type InvalidStateError, if the local Bluetooth le adapter is currently not enabled.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

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

function proccessDevice(device) {
  device.connect(onConnected, onerror);
  function onConnected() {
    var service = device.getService(device.uuids[0]);
    if (service.characteristics.length > 0) {
      var characteristic = service.characteristics[0];
      var descriptor = characteristic.descriptors[0];
      descriptor.readValue(function(val) {
        console.log("Value read: " + val);
        device.disconnect();
      });
    }
  }
}

var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
  onsuccess: function(device) {
    if (device.address == "11:22:33:44:55:66") {
      console.log("Found device: " + device.name);
      proccessDevice(device);
    }
  }
});
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*writeValue*

Writes the descriptor value to the remote device.

void writeValue(byte[] value, optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);

Since: 2.3.1

The ErrorCallback is launched with these error types:

- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.admin (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- value: the descriptor value to write
- successCallback *[optional] [nullable]*: Callback function that is called when the descriptor value is written successfully.
- errorCallback *[optional] [nullable]*: Callback function that is called in case of failure.

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type for that parameter.
  - with error type InvalidStateError, if the local Bluetooth le adapter is currently not enabled.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

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

function proccessDevice(device) {
  device.connect(onConnected, onerror);
  function onConnected() {
    var service = device.getService(device.uuids[0]);
    if (service.characteristics.length > 0) {
      var characteristic = service.characteristics[0];
      var data = new Array(1, 2, 3, 4, 5, 6);
      var descriptor = characteristic.descriptors[0];
      descriptor.writeValue(data, function() {
        console.log("Value written");
        device.disconnect();
      }, function(e) {
        console.log("Failed to write: " + e.message);
      });
    }
  }
}

var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
  onsuccess: function(device) {
    if (device.address == "11:22:33:44:55:66") {
      console.log("Found device: " + device.name);
      proccessDevice(device);
    }
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.12. BluetoothLEScanCallback

Bluetooth scanning process interface that defines the success callback for BluetoothLEAdapter.startScan().

```
[Callback, NoInterfaceObject] interface BluetoothLEScanCallback {
  void onsuccess(BluetoothLEDevice device);
};
```

Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)***onsuccess***

Called when a new device is successfully discovered in the process of scanning.

```
void onsuccess(BluetoothLEDevice device);
```

Since: 2.3.1

Parameters:

- device: Device that is found

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.13. BluetoothLEAdvertiseCallback

Bluetooth advertising process interface that defines the success callback for BluetoothLEAdapter.startAdvertise().

```
[Callback=FunctionOnly, NoInterfaceObject] interface BluetoothLEAdvertiseCallback {
  void onstate(BluetoothAdvertisingState state);
};
```

Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)***onstate***

Called when the advertising state is changed.

```
void onstate(BluetoothAdvertisingState state);
```

Since: 2.3.1

Parameters:

- state: State that is Advertising process

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.14. BluetoothLEConnectChangeCallback

Bluetooth LE connecting process interface that defines callbacks for getting notified about changes of connection to a specific LE based service on a remote Bluetooth LE device.

```
[Callback, NoInterfaceObject] interface BluetoothLEConnectChangeCallback {
  void onconnected(BluetoothLEDevice device);
  void ondisconnected(BluetoothLEDevice device);
};
```

Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)***onconnected***

Called at the beginning of connection to a specific LE based service on a remote Bluetooth LE device.

```
void onconnected(BluetoothLEDevice device);
```

Since: 2.3.1

Parameters:

- device

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)***ondisconnected***

Called at the beginning of disconnection from a specific LE based service on a remote Bluetooth LE device.

```
void ondisconnected(BluetoothLEDevice device);
```

Since: 2.3.1

Parameters:

- device

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.15. ReadValueSuccessCallback

The ReadValueSuccessCallback interface implements the callback for BluetoothGATTCharacteristic.readValue() and BluetoothGATTDescriptor.readValue() methods.

```
[Callback=FunctionOnly, NoInterfaceObject] interface ReadValueSuccessCallback {
  void onread(byte[] value);
};
```

Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)***onread***

Called when a characteristic value has been read.

```
void onread(byte[] value);
```

Since: 2.3.1

Parameters:

- value: Read characteristic value

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.16. BluetoothDevice

The BluetoothDevice interface represents a remote Bluetooth device.

```
[NoInterfaceObject] interface BluetoothDevice {
  readonly attribute DOMString name;
  readonly attribute BluetoothAddress address;
  readonly attribute BluetoothClass deviceClass;
  readonly attribute boolean isBonded;
  readonly attribute boolean isTrusted;
  readonly attribute boolean isConnected;
  readonly attribute BluetoothUUID[] uuids;

  void connectToServiceByUUID(BluetoothUUID uuid,
                              BluetoothSocketSuccessCallback successCallback,
                              optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
};
```

Since: 2.3.1

A BluetoothDevice object can be retrieved using one of the following APIs:

- BluetoothAdapter.getDevice()
- BluetoothAdapter.getKnownDevices()
- BluetoothAdapter.discoverDevices()
- BluetoothAdapter.createBonding()

Attributes

- **readonly DOMString name**
  The readable name of this remote device.
  Since: 2.3.1

  Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
adapter.getDevice("11:22:33:44:55:66", function(device) {
  console.log("Device Name: " + device.name);
});
```

- **readonly** BluetoothAddress **address**
  The hardware address of this remote device.
  Since: 2.3.1

  Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
adapter.getDevice("11:22:33:44:55:66", function(device) {
  console.log("Device Address: " + device.address);
});
```

- **readonly** BluetoothClass **deviceClass**
  The device class, which represents the type of the device and the services it provides.
  Since: 2.3.1

  Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
adapter.getDevice("11:22:33:44:55:66", function(device) {
  console.log("Device Major Class: " + device.deviceClass.major);
});
```

- **readonly** boolean **isBonded**
  The bond state of this remote device with the local device.
  Since: 2.3.1

  Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
adapter.getDevice("11:22:33:44:55:66", function(device) {
  console.log("Is bonded: " + (device.isBonded ? "Yes" : "No"));
});
```

- **readonly** boolean **isTrusted**
  The flag indicating whether the local device recognizes this remote device as a trusted device or not.
  Since: 2.3.1

  Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
adapter.getDevice("11:22:33:44:55:66", function(device) {
  console.log("Is trusted: " + (device.isTrusted ? "Yes" : "No"));
});
```

- **readonly** boolean **isConnected**
  The flag indicating the connection state of this remote device with the local device.
  Since: 2.3.1

  Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
adapter.getDevice("11:22:33:44:55:66", function(device) {
  console.log("Is connected: " + (device.isConnected ? "Yes" : "No"));
});
```

- **readonly** BluetoothUUID[] **uuids**

The list of 128 bit service UUIDs available on this remote device.
Since: 2.3.1

Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
var CHAT_SERVICE_UUID = "5BCE9431-6C75-32AB-AFE0-2EC108A30860";
adapter.getDevice("11:22:33:44:55:66", function(device) {
  var uuids = device.uuids;
  var services = "";
  for (var i = 0; i < uuids.length; i++) {
    services += uuids[i] + "
";
  }
  console.log ("Services found: " + services);
  if (uuids.indexOf(CHAT_SERVICE_UUID) != -1) {
    // Connects to service
    device.connectToServiceByUUID(CHAT_SERVICE_UUID, function(socket) {
      //
      // Connected to service, handle socket
      //
    }, function (e) {
      console.log("Could not connect to chat service !!!. Error: " + e.message);
    });
  }
});
```

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*connectToServiceByUUID*

Connects to a specified service identified by uuid on this remote device.

```
void connectToServiceByUUID(BluetoothUUID uuid, BluetoothSocketSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

If opening a connection is successful, then a BluetoothSocket object with open state is sent using successCallback, through which data can be exchanged by both devices.

The ErrorCallback is launched with these error types:

- NotFoundError - If there is no service with the specified uuid
- InvalidValuesError - If any of the input parameters contain an invalid value
- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.spp (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- uuid: 128 bit unique identifier, which specifies the service on the remote device
- successCallback: Callback function that is called when an asynchronous call completes successfully
- errorCallback *[optional] [nullable]*: Callback function that is called when opening of a socket fails

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type for that parameter.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

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

// Calls a method that is invoked when user wants to send a message to a remote device.
function sendMessage(msg) {
```

```
    // Validates socket state, if everything is ok.
    if (clientSocket != null && clientSocket.state == "OPEN") {
      // Sends the message.
      clientSocket.writeData(msg);
    }
  }

  // Calls a method that is invoked when a socket is open.
  function onSocketConnected(socket) {
    clientSocket = socket;
    console.log("Opening a socket successfully!!!");
    socket.onmessage = function () {
      var data = socket.readData();
      var recvmsg = "";
      for (var i = 0; i < data.length; i++)
      {
        recvmsg += String.fromCharCode(data[i]);
      }
      console.log("server msg >> " + recvmsg);
    };

    socket.onclose = function() {
      console.log("socket disconnected.");
    };
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.17. BluetoothLEDevice

The BluetoothLEDevice interface represents a remote Bluetooth LE device.

```
  [NoInterfaceObject] interface BluetoothLEDevice {
    readonly attribute BluetoothAddress address;
    readonly attribute DOMString? name;
    readonly attribute long? txpowerlevel;
    readonly attribute unsigned long? appearance;
    readonly attribute BluetoothUUID[]? uuids;
    readonly attribute BluetoothLESolicitationUUID[]? solicitationuuids;
    readonly attribute BluetoothLEServiceData[]? serviceData;
    readonly attribute BluetoothLEManufacturerData? manufacturerData;

    void connect(optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback,
                 optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
    void disconnect(optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback,
                 optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

    BluetoothGATTService getService(BluetoothUUID uuid) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

    long addConnectStateChangeListener(BluetoothLEConnectChangeCallback listener) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

    void removeConnectStateChangeListener(long watchID);
  };
```

Since: 2.3.1

A BluetoothLEDevice object can be retrieved by using the following API:

- BluetoothLEAdapter.startScan()

Attributes

- **readonly** BluetoothAddress **address**
  The address of the Bluetooth LE device from the scan result information.

Since: 2.3.1

Code example:

```
var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
  onsuccess: function(device) {
    console.log("Found device: " + device.address);
  }
});
```

- **readonly** DOMString **name** *[nullable]*

  The name of the Bluetooth LE device from the scan result information.

  Since: 2.3.1

  Code example:

  ```
  var adapter = tizen.bluetooth.getLEAdapter();
  adapter.startScan({
    onsuccess: function(device) {
      console.log("Found device: " + device.name);
    }
  });
  ```

- **readonly** unsigned long **txpowerlevel** *[nullable]*

  The transmission power level of the Bluetooth LE device from the scan result information.

  Since: 2.3.1

  Code example:

  ```
  var adapter = tizen.bluetooth.getLEAdapter();
  adapter.startScan({
    onsuccess: function(device) {
      console.log("Found device: " + device.txpowerlevel);
    }
  });
  ```

- **readonly** unsigned long **appearance** *[nullable]*

  The appearance of the Bluetooth LE device from the scan result information.

  Since: 2.3.1

  Code example:

  ```
  var adapter = tizen.bluetooth.getLEAdapter();
  adapter.startScan({
    onsuccess: function(device) {
      console.log("Found device: " + device.appearance);
    }
  });
  ```

- **readonly** BluetoothUUID[] **uuids** *[nullable]*

  The list of 128 bit service UUIDs available on this remote device.

  Since: 2.3.1

  Code example:

  ```
  var adapter = tizen.bluetooth.getLEAdapter();
  adapter.startScan({
    onsuccess: function(device) {
      var uuids = device.uuids;
      var services = "";
      for (var i = 0; i < uuids.length; i++) {
        services += uuids[i] + "\n";
      }
  ```

```
        console.log ("Service found: " + services);
    }
});
```

- **readonly** BluetoothLESolicitationUUID[] **solicitationuuids** *[nullable]*
  The list of service solicitation UUIDs available on Bluetooth LE device from the scan result information.
  Since: 2.3.1

  Code example:

```
var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
    onsuccess: function(device) {
        var uuids = device.solicitationuuids;
        var services = "";
        for (var i = 0; i < uuids.length; i++) {
            services += uuids[i] + "\n";
        }
        console.log ("Service solicitations found: " + services);
    }
});
```

- **readonly** BluetoothLEServiceData[] **serviceData** *[nullable]*
  The list of service data available on Bluetooth LE device from the scan result information.
  Since: 2.3.1

  Code example:

```
var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
    onsuccess: function(device) {
        var serviceData = device.serviceData;
        var data = "";
        for (var i = 0; i < serviceData.length; i++) {
            data += serviceData[i].id + serviceData[i].data + "\n";
        }
        console.log ("Service data found: " + data);
    }
});
```

- **readonly** BluetoothLEManufacturerData **manufacturerData** *[nullable]*
  The manufacturer data from the scan result information.
  Since: 2.3.1

  Code example:

```
var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
    onsuccess: function(device) {
        var manufacturerData = device.manufacturerData;
        console.log ("Manufacture id: " + manufacturerData.id);
        console.log ("Manufacture data: " + manufacturerData.data);
    }
});
```

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*connect*

Establishes Low Energy connection to the device.

void connect(optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)**? successCallback, optional** ErrorCallback (?

redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);

Since: 2.3.1

Connection is required to readValue() and writeValue() from the remote device.

The ErrorCallback is launched with these error types:

- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.spp (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- successCallback *[optional] [nullable]*: Callback function that is called when the connection is established successfully
- errorCallback *[optional] [nullable]*: Callback function that is called in case of failure

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type for that parameter.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

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

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

var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
  onsuccess: function(device) {
    if (device.address == "11:22:33:44:55:66") {
      console.log("Found device: " + device.name);
      device.connect(onconnected, onerror);
    }
  }
});
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*disconnect*

Disconnects from the device.

void disconnect(optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);

Since: 2.3.1

The ErrorCallback is launched with these error types:

- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.admin (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- successCallback *[optional] [nullable]*: Callback function that is called when the connection is finished successfully
- errorCallback *[optional] [nullable]*: Callback function that is called in case of failure

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type for that parameter.
  - with error type InvalidStateError, if device is currently not connected.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

```
function onerror(e) {
    console.log("Error occured: " + e.message);
}

function onDeviceFound(device) {
  function ondisconnect() {
     console.log("Disconnected");
  }

  function onconnected() {
     console.log("Connected to device");
     device.disconnect(ondisconnect, onerror);
  }

  if (device.address == "11:22:33:44:55:66") {
     console.log("Found device: " + device.name);
     device.connect(onconnected, onerror);
  }
}

var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
    onsuccess: onDeviceFound
}, onerror);
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*getService*

Retrieves a service from the device for the given UUID.

```
BluetoothGATTService getService(BluetoothUUID uuid);
```

Since: 2.3.1

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.admin (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- uuid: UUID of the service

Exceptions:

- WebAPIException
  - with error type NotFoundError, if there is no service with the given UUID.
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type InvalidStateError, if the GATT service is not available.
  - with error type UnknownError, if any other error occurs.

Code example:

```
function onconnected(device) {
     console.log("Connected to device");
     var service = device.getService(device.uuids[0]);
     console.log("Service got");
}

function onerror(e) {
    console.log("Error occured: " + e.message);
}

function onDeviceFound(device) {
  if (device.address == "11:22:33:44:55:66") {
     console.log("Found device: " + device.name);
     device.connect(onconnected.bind(null, device), onerror);
  }
```

```
}

var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
  onsuccess: onDeviceFound
}, onerror);
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*addConnectStateChangeListener*

Registers a listener to be called when the device connects or disconnects.

```
long addConnectStateChangeListener(BluetoothLEConnectChangeCallback listener);
```

Since: 2.3.1

Parameters:

- listener: Listener functions that are called when the connection state changes

Return value:

long The watchID to be used to unregister the listener

Exceptions:

- WebAPIException
  - with error type UnknownError, if any other error occurs.

Code example:

```
function onerror(e) {
  console.log("Error occured: " + e.message);
}

function onDeviceFound(device) {
  var onConnectionStateChange = {
    onconnected: function(device) {
      console.log("Device " + device.name + " connected");
    },
    ondisconnected: function(device) {
      console.log("Device " + device.name + " disconnected");
    }
  }

  if (device.address === "11:22:33:44:55:66") {
    console.log("Found device: " + device.name);
    device.addConnectStateChangeListener(onConnectionStateChange);
    device.connect();
  }
}

var adapter = tizen.bluetooth.getLEAdapter();
adapter.startScan({
  onsuccess: onDeviceFound
}, onerror);
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*removeConnectStateChangeListener*

Unregisters a Bluetooth device connection listener

```
void removeConnectStateChangeListener(long watchID);
```

Since: 2.3.1

Parameters:

- watchID: The watchID identifier returned by the addConnectStateChangeListener() method.

Code example:

```
function onerror(e) {
  console.log("Error occured: " + e.message);
}

function onDeviceFound(device) {
  var onConnectionStateChange = {
    onconnected: function(device) {
      console.log("Device " + device.name + " connected");
    },
    ondisconnected: function(device) {
      console.log("Device " + device.name + " disconnected");
      device.removeConnectStateChangeListener(listenerID);
    }
  }

  if (device.address === "11:22:33:44:55:66") {
    console.log("Found device: " + device.name);
    listenerID = device.addConnectStateChangeListener(onConnectionStateChange);
    device.connect();
  }
}

var adapter = tizen.bluetooth.getLEAdapter();
var listenerID = null;
adapter.startScan({
  onsuccess: onDeviceFound
}, onerror);
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.18. BluetoothSocket

The BluetoothSocket interface represents the Bluetooth socket.

```
[NoInterfaceObject] interface BluetoothSocket {
  readonly attribute BluetoothUUID uuid;
  readonly attribute BluetoothSocketState state;
  readonly attribute BluetoothDevice peer;
  [TreatNonCallableAsNull] attribute SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? onmessage raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
  [TreatNonCallableAsNull] attribute SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? onclose raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
  unsigned long writeData(byte[] data) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));


  byte[] readData() raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));


  void close() raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
};
```

Since: 2.3.1

The socket object is created by BluetoothDevice.connectToServiceByUUID() or BluetoothAdapter.registerRFCOMMServiceByUUID().

Attributes

- **readonly** BluetoothUUID **uuid**
  The service UUID to which this socket is connected.
  Since: 2.3.1
- **readonly** BluetoothSocketState **state**
  The socket state.

Since: 2.3.1

- **readonly** BluetoothDevice **peer**
  The peer device to which this socket is connected.
  Since: 2.3.1

- SuccessCallback **onmessage** ***[nullable]***
  Called when an incoming message is received successfully from the peer. By default, this attribute is set to null.
  Since: 2.3.1

  Exceptions:

  WebAPIException

  with error type TypeMismatchError, if any input attribute is not compatible with the expected type for this attribute.

- SuccessCallback **onclose** ***[nullable]***
  Called when the socket is closed successfully. By default, this attribute is set to null.
  Since: 2.3.1

  Exceptions:

  WebAPIException

  with error type TypeMismatchError, if any input attribute is not compatible with the expected type for this attribute.

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *writeData*

Writes data as a sequence of bytes onto the socket and returns the number of bytes actually written.

```
unsigned long writeData(byte[] data);
```

Since: 2.3.1

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.spp (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- data: The data to send

Return value:

unsigned long The number of bytes actually sent

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if any input parameters in not compatible with the expected type for that parameter.
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type UnknownError, if any other error occurs.

Code example:

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

function onSocketConnected(socket) {
  console.log ("Opened connection to remote device");
  socket.onmessage = function () {
   console.log ("Message received: " + socket.readData());
  };

  socket.onclose = function() {
    console.log("Socket closed with " + socket.peer.name);
  };

  // Sends data to peer.
  var textmsg = "Test";
  var sendtextmsg = new Array();
  for (var i = 0; i < textmsg.length; i++)
  {
    sendtextmsg[i] = textmsg.charCodeAt(i);
```

```
  }
  socket.writeData (sendtextmsg);
}

function onSocketError(e) {
  console.log ("Error connecting to service. Reason: " + e.message);
}

function onDeviceReady(device) {
  // Validates device and service uuid.
  if (device.uuids.indexOf("5BCE9431-6C75-32AB-AFE0-2EC108A30860") != -1) {
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *readData*

Reads data from the socket.

```
byte[] readData();
```

Since: 2.3.1

This method should be called only in the BluetoothSocket.onmessage handler, that is, when data is ready on the socket.

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.spp (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Return value:

byte[] The sequence of bytes successfully read

Exceptions:

- WebAPIException
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type UnknownError, if any other error occurs.

Code example:

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

function onError(e) {
  console.log ("Error connecting to service. Reason: " + e.message);
}

function onSocketConnected(socket) {
  console.log ("Opening socket success!!!");

  socket.onmessage = function() {
    //  Gets a message from peer, reads it
    var data = socket.readData();

    //
    // Code to evaluate message goes here
    //
  };

  socket.onclose = function() {
    console.log("Socket closed with " + socket.peer.name);
  };
}

function onDeviceReady(device) {
  // Validates device and service uuid
  if (device.uuids.indexOf("5BCE9431-6C75-32AB-AFE0-2EC108A30860") != -1) {
    // Opens socket
    device.connectToServiceByUUID("5BCE9431-6C75-32AB-AFE0-2EC108A30860", onSocketConnected, onError);
  }
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *close*

Closes the socket.

```
void close();
```

Since: 2.3.1

BluetoothSocket.state changes to *CLOSED*, and BluetoothSocket.onclose() is invoked on success.

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.spp (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Exceptions:

- WebAPIException
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type UnknownError, if any other error occurs.

[(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)](/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.19. BluetoothClass

The BluetoothClass interface represents Bluetooth Class of Device/Service (CoD).

```
[NoInterfaceObject] interface BluetoothClass {
  readonly attribute octet major;

  readonly attribute octet minor;

  readonly attribute unsigned short [] services ;

  boolean hasService(unsigned short service) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
};
```

Since: 2.3.1

Bluetooth device class describes the characteristics and capabilities of a device.

Bluetooth CoD is a 24 bit integer created by the union of three components:

- Exactly one **Major Device Class** (bits 8-12 of CoD) - This is the highest level of granularity for defining a Bluetooth Device.
- Exactly one **Minor Device Class** (bits 2-7 of CoD) - This is to be interpreted only in the context of the Major Device Class. Thus, the meaning of these bits may change, depending on the value of 'Major Device Class'.
- Zero or more **Major Service Classes** (bits 13-23) - Represents the services supported by the device.

The Major and Minor classes are intended to define a general family of devices with which any particular implementation wishes to be associated. No assumptions should be made about specific functionality or characteristics of any application, based solely on the assignment of a Major or minor device class.

Attributes

- **readonly** octet **major**
  The major device class.
  The BluetoothClassDeviceMajor interface contains the list of known values.
  Since: 2.3.1

  Code example:

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

function evaluateDevice(address) {
  adapter.getDevice(address, function(device) {
    if (device.deviceClass.major == tizen.bluetooth.deviceMajor.COMPUTER) {
      // Shows computer icon for this device
      console.log("Device is computer");
    } else if (device.deviceClass.major == tizen.bluetooth.deviceMajor.PHONE) {
      // Shows phone icon
      console.log("Device is a Phone");
    }
  }, function(e) {
    console.log("Couldn't get any device with the given address: " + e.message);
  });
```

```
}

evaluateDevice("11:22:33:44:55:66");
```

- **readonly** octet **minor**
  The minor device class.
  The BluetoothClassDeviceMinor interface contains the list of known values.
  Since: 2.3.1
- **readonly** unsigned short[] **services**
  The services provided by this device and it refers to the BluetoothClassDeviceService interface for the list of possible values.
  Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*hasService*

Checks whether the given service exists in the services.

```
boolean hasService(unsigned short service);
```

Since: 2.3.1

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.gap (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- service: Service to check

Exceptions:

- WebAPIException
  - with error type InvalidValuesError, if any of the input parameters contain an invalid value.
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type UnknownError, if any other error occurs.

Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
adapter.getDevice("12:34:56:78:9A:BC", function(device) {
  if (device.deviceClass.hasService(tizen.bluetooth.deviceService.POSITIONING)) {
     console.log("Device supports Positioning service");
  }
}, function(e) {
  console.log("Couldn't get device for given address: " + e.message);
});
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.20. BluetoothClassDeviceMajor

The BluetoothClassDeviceMajor interface holds the identifiers for major device classes of Bluetooth CoD.

```
[NoInterfaceObject] interface BluetoothClassDeviceMajor {

  const octet MISC = 0x00;
  const octet COMPUTER = 0x01;
  const octet PHONE = 0x02;
  const octet NETWORK = 0x03;
  const octet AUDIO_VIDEO = 0x04;
  const octet PERIPHERAL = 0x05;
  const octet IMAGING = 0x06;
  const octet WEARABLE = 0x07;
  const octet TOY = 0x08;
```

```
    const octet HEALTH = 0x09;
    const octet UNCATEGORIZED = 0x1F;
};
```

Since: 2.3.1

[(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)](/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.21. BluetoothClassDeviceMinor

The BluetoothClassDeviceMinor interface holds the identifiers for minor device classes of Bluetooth CoD.

```
[NoInterfaceObject] interface BluetoothClassDeviceMinor {
    const octet COMPUTER_UNCATEGORIZED = 0x00;
    const octet COMPUTER_DESKTOP = 0x01;
    const octet COMPUTER_SERVER = 0x02;
    const octet COMPUTER_LAPTOP = 0x03;
    const octet COMPUTER_HANDHELD_PC_OR_PDA = 0x04;
    const octet COMPUTER_PALM_PC_OR_PDA = 0x05;
    const octet COMPUTER_WEARABLE = 0x06;

    const octet PHONE_UNCATEGORIZED = 0x00;
    const octet PHONE_CELLULAR = 0x01;
    const octet PHONE_CORDLESS = 0x02;
    const octet PHONE_SMARTPHONE = 0x03;
    const octet PHONE_MODEM_OR_GATEWAY = 0x04;
    const octet PHONE_ISDN = 0x05;

    const octet AV_UNRECOGNIZED = 0x00;
    const octet AV_WEARABLE_HEADSET = 0x01;
    const octet AV_HANDSFREE = 0x02;
    const octet AV_MICROPHONE = 0x04;
    const octet AV_LOUDSPEAKER = 0x05;
    const octet AV_HEADPHONES = 0x06;
    const octet AV_PORTABLE_AUDIO = 0x07;
    const octet AV_CAR_AUDIO = 0x08;
    const octet AV_SETTOP_BOX = 0x09;
    const octet AV_HIFI = 0x0a;
    const octet AV_VCR = 0x0b;
    const octet AV_VIDEO_CAMERA = 0x0c;
    const octet AV_CAMCORDER = 0x0d;
```

Since: 2.3.1

[(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)](/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.22. BluetoothClassDeviceService

The BluetoothClassDeviceService interface holds identifiers for the major service classes of Bluetooth CoD.

```
[NoInterfaceObject] interface BluetoothClassDeviceService {
    const unsigned short LIMITED_DISCOVERABILITY = 0x0001;
    const unsigned short POSITIONING = 0x0008;
    const unsigned short NETWORKING = 0x0010;
    const unsigned short RENDERING = 0x0020;
    const unsigned short CAPTURING = 0x0040;
    const unsigned short OBJECT_TRANSFER = 0x0080;
    const unsigned short AUDIO = 0x0100;
    const unsigned short TELEPHONY = 0x0200;
    const unsigned short INFORMATION = 0x0400;
};
```

Since: 2.3.1

[(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)](/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.23. BluetoothServiceHandler

The BluetoothServiceHandler interface provides methods to handle Bluetooth service.

```
[NoInterfaceObject] interface BluetoothServiceHandler {
  readonly attribute BluetoothUUID uuid;
  readonly attribute DOMString name;
  readonly attribute boolean isConnected;
  [TreatNonCallableAsNull] attribute BluetoothSocketSuccessCallback? onconnect raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

  void unregister(optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback)
raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));
};
```

Since: 2.3.1

Attributes

- **readonly** BluetoothUUID **uuid**
  The UUID of the service.
  Since: 2.3.1
- **readonly** DOMString **name**
  The name of the service.
  Since: 2.3.1
- **readonly** boolean **isConnected**
  The flag indicating whether any remote devices is using this service.
  Since: 2.3.1
- BluetoothSocketSuccessCallback **onconnect** ***[nullable]***
  Called when a remote device is connected successfully to this service. By default, this attribute is set to null.
  Since: 2.3.1

  Exceptions:

  WebAPIException

  with error type TypeMismatchError, if any input attribute is not compatible with the expected type for this attribute.

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *unregister*

Unregisters a service record from the Bluetooth services record database and stops listening for new connections to this service.

```
void unregister(optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

The ErrorCallback is launched with these error types:

- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.spp (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- successCallback *[optional] [nullable]*: Callback function that is called when the record is removed successfully from the service records database
- errorCallback *[optional] [nullable]*: Callback function that is called in case of failure (to unregister)

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type for that parameter.
  - with error type SecurityError, if the application does not have the privilege to call this method.

Code example:

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

function chatServiceSuccessCb(handler) {
  console.log("Chat service registration was successful!");

  chatServiceHandler = handler;
  handler.onconnect = function(socket) {
    console.log("Client is connected: " + socket.peer.name + "," + socket.peer.address);
    socket.onmessage = function() {
      var data = socket.readData();
      // Handle message code goes here
      //....
    };

    // Expected close
    socket.onclose = function() {
      console.log('The socket is closed.');
    };
  };
}

function publishChatService() {
  var CHAT_SERVICE_UUID = "5BCE9431-6C75-32AB-AFE0-2EC108A30860";
  adapter.registerRFCOMMServiceByUUID(CHAT_SERVICE_UUID, "Chat service", chatServiceSuccessCb,
    // Error handler
    function(e) {
      console.log( "Could not register service record, Error: " + e.message);
    });
}
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.24. BluetoothProfileHandler

The BluetoothProfileHandler interface represents the Bluetooth profile handler.

```
[NoInterfaceObject] interface BluetoothProfileHandler {

  readonly attribute BluetoothProfileType profileType;
};
```

Since: 2.3.1

Attributes

- **readonly** BluetoothProfileType **profileType**
  The Bluetooth profile type.
  Since: 2.3.1

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.25. BluetoothHealthProfileHandler

This interface represents the handler of Bluetooth health device profile. The BluetoothHealthProfileHandler object is created by BluetoothAdapter.getBluetoothProfileHandler().

```
[NoInterfaceObject] interface BluetoothHealthProfileHandler : BluetoothProfileHandler {

  void registerSinkApplication(unsigned short dataType, DOMString name, BluetoothHealthApplicationSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback)
raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

  void connectToSource(BluetoothDevice peer, BluetoothHealthApplication application, BluetoothHealthChannelSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

};
```

Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *registerSinkApplication*

Registers an application for the Sink role.

```
void registerSinkApplication(unsigned short dataType, DOMString name, BluetoothHealthApplicationSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

The ErrorCallback is launched with these error types:

- ServiceNotAvailableError - If a Bluetooth device is turned off
- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.health (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- dataType: MDEP data type used for communication, which is referenced in the ISO/IEEE 11073-20601 spec. For example, pulse oximeter is 4100 and blood pressure monitor is 4103.
- name: Friendly name associated with sink application
- successCallback: Callback function that is called when a sink application is registered successfully
- errorCallback *[optional] [nullable]*: Callback function that is called when an error occurs

Exceptions:

- WebAPIException
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type TypeMismatchError, if any of the input parameter is not compatible with the expected type for that parameter.

Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
var healthProfileHandler = adapter.getBluetoothProfileHandler("HEALTH");

function healthRegisterSuccess(app) {
  console.log("Registered application: " + app.name);
}

function healthRegisterError(e) {
  console.log("Failed to register application: " + e.message);
};

healthProfileHandler.registerSinkApplication(4100, "testSinkApp", healthRegisterSuccess, healthRegisterError);
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *connectToSource*

Connects to the health device which acts as the Source role.

```
void connectToSource(BluetoothDevice peer, BluetoothHealthApplication application, BluetoothHealthChannelSuccessCallback successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback);
```

Since: 2.3.1

The ErrorCallback is launched with these error types:

- ServiceNotAvailableError - If a Bluetooth device is turned off
- InvalidValuesError - If any of the input parameters contain an invalid value
- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.health (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- peer: Remote device which acts as the Source role
- application: Registered application for the Sink role
- successCallback: Callback function that is called when a connection is established successfully
- errorCallback *[optional] [nullable]*: Callback function that is called when an error occurs

Exceptions:

- WebAPIException
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type TypeMismatchError, if any of the input parameter is not compatible with the expected type for that parameter.

Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
var healthProfileHandler = adapter.getBluetoothProfileHandler("HEALTH");
var registerHealthApp = null;

function healthConnectSuccess(channel) {
  console.log("Health device is connected");
}

function healthConnectError(e) {
  console.log("Failed to connect to source: " + e.message);
};

function gotDeviceInfo(device) {
  healthProfileHandler.connectToSource(device, registerHealthApp, healthConnectSuccess, healthConnectError);
}

function healthRegisterSuccess(app) {
  console.log("Registered application: " + app.name);
  registerHealthApp = app;
  adapter.getDevice("35:F4:59:D1:7A:03", gotDeviceInfo);
}

function healthRegisterError(e) {
  console.log("Failed to register application: " + e.message);
};

healthProfileHandler.registerSinkApplication(4100, "testSinkApp", healthRegisterSuccess, healthRegisterError);
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.26. BluetoothHealthApplication

The BluetoothHealthApplication interface represents the Bluetooth health application.

```
[NoInterfaceObject] interface BluetoothHealthApplication {

  readonly attribute unsigned short dataType;

  readonly attribute DOMString name;

  [TreatNonCallableAsNull] attribute BluetoothHealthChannelSuccessCallback? onconnect raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

  void unregister(optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)? successCallback, optional ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)? errorCallback)
raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

};
```

Since: 2.3.1

Attributes

- **readonly** unsigned short **dataType**
  The MDEP data type used for communication, which is referenced in the ISO/IEEE 11073-20601 spec.
  For example, pulse oximeter is 4100 and blood pressure monitor is 4103.
  Since: 2.3.1
- **readonly** DOMString **name**
  The friendly name associated with sink application.
  Since: 2.3.1
- BluetoothHealthChannelSuccessCallback **onconnect** *[nullable]*
  Called when a health device is connected successfully through this application.
  By default, this attribute is set to null.
  Since: 2.3.1

  Exceptions:

  WebAPIException

  with error type TypeMismatchError, if an input attribute is not compatible with the expected type for this attribute.

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *unregister*

Unregisters this application.

void unregister(optional SuccessCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#SuccessCallback)**? successCallback, optional** ErrorCallback (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#ErrorCallback)**? errorCallback);**

Since: 2.3.1

The ErrorCallback is launched with these error types:

- ServiceNotAvailableError - If a Bluetooth device is turned off
- UnknownError - If any other error occurs

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.health (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- successCallback *[optional] [nullable]*: Callback function that is called when a sink application is registered successfully
- errorCallback *[optional] [nullable]*: Callback function that is called when an error occurs

Exceptions:

- WebAPIException
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type TypeMismatchError, if any of the input parameter is not compatible with the expected type for that parameter.

Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
var healthProfileHandler = adapter.getBluetoothProfileHandler("HEALTH");
var healthApp = null;

function healthRegisterSuccess(app) {
   console.log("Registered application: " + app.name);
   healthApp = app;
   healthApp.onconnect = function(channel) {
      console.log("Connected!!");
   };
}

function healthRegisterError(e) {
   console.log("Failed to register application: " + e.message);
};

function startSink() {
   try {
```

```
        healthProfileHandler.registerSinkApplication(4100, "testSinkApp", healthRegisterSuccess, healthRegisterError);
    } catch(e) {
        console.log("Error: " + e.message);
    }
}

function stopSink() {
    try {
        if (healthApp != null) {
            healthApp.unregister(function() {
                healthApp = null;
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.27. BluetoothHealthChannel

The BluetoothHealthChannel interface represents the Bluetooth health channel.

```
[NoInterfaceObject] interface BluetoothHealthChannel {

    readonly attribute BluetoothDevice peer;

    readonly attribute BluetoothHealthChannelType channelType;

    readonly attribute BluetoothHealthApplication application;

    readonly attribute boolean isConnected;

    void close() raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

    unsigned long sendData(byte[] data) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

    void setListener(BluetoothHealthChannelChangeCallback listener) raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

    void unsetListener() raises(WebAPIException (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#WebAPIException));

};
```

Since: 2.3.1

Attributes

- **readonly** BluetoothDevice **peer**
  The remote device to which this channel is connected.
  Since: 2.3.1
- **readonly** BluetoothHealthChannelType **channelType**
  The type of this channel.
  Since: 2.3.1
- **readonly** BluetoothHealthApplication **application**
  The health application which is used to communicate with the remote device.
  Since: 2.3.1
- **readonly** boolean **isConnected**
  The flag indicating whether any remote device is connected.
  Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *close*

Closes the connected channel. BluetoothHealthChannel.isConnected is changed to *false* and BluetoothHealthChannelChangeCallback.onclose is invoked when this channel is closed successfully.

```
void close();
```

Since: 2.3.1

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.health (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Exceptions:

- WebAPIException
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type ServiceNotAvailableError, if a Bluetooth device is turned off.
  - with error type UnknownError, if any other error occurs.

Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
var healthProfileHandler = adapter.getBluetoothProfileHandler("HEALTH");

function healthRegisterSuccess(app) {
  console.log("Registered application: " + app.name);
  app.onconnect = function(channel) {;
    console.log("Health device is connected");
    channel.close();
  }
}

function healthRegisterError(e) {
  console.log("Failed to register application: " + e.message);
};

healthProfileHandler.registerSinkApplication(4100, "testSinkApp", healthRegisterSuccess, healthRegisterError);
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*sendData*

Sends data and returns the number of bytes actually written.

```
unsigned long sendData(byte[] data);
```

Since: 2.3.1

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.health (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- data: Data to send

Return value:

unsigned long Number of bytes actually sent

Exceptions:

- WebAPIException
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type ServiceNotAvailableError, if a Bluetooth device is turned off.
  - with error type TypeMismatchError, if any of the input parameter is not compatible with the expected type for that parameter.
  - with error type UnknownError, if any other error occurs.

Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
var healthProfileHandler = adapter.getBluetoothProfileHandler("HEALTH");

function healthRegisterSuccess(app) {
  console.log("Registered application: " + app.name);
  app.onconnect = function(channel) {
    console.log("Health device is connected");
    channel.sendData(dataToSend);
```

```
    }
}

function healthRegisterError(e) {
    console.log("Failed to register application: " + e.message);
};

healthProfileHandler.registerSinkApplication(4100, "testSinkApp", healthRegisterSuccess, healthRegisterError);
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*setListener*

Sets the listener to receive notifications.

```
void setListener(BluetoothHealthChannelChangeCallback listener);
```

Since: 2.3.1

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.health (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Parameters:

- listener: Event listener of Bluetooth health channel

Exceptions:

- WebAPIException
  - with error type TypeMismatchError, if the input parameter is not compatible with the expected type.
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type UnknownError, if any other error occurs.

Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
var healthProfileHandler = adapter.getBluetoothProfileHandler("HEALTH");

var channelCallback = {
    onmessage: function(data) {
        console.log("data is received")
    },
    onclose: function() {
        console.log("channel is closed");
    }
};

function healthRegisterSuccess(app) {
    console.log("Registered application: " + app.name);
    app.onconnect = function(channel) {;
        console.log("Health device is connected");
        channel.setListener(channelCallback);
    }
}

function healthRegisterError(e) {
    console.log("Failed to register application: " + e.message);
};

healthProfileHandler.registerSinkApplication(4100, "testSinkApp", healthRegisterSuccess, healthRegisterError);
```

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*unsetListener*

Unsets the listener. This stops receiving notifications.

```
void unsetListener();
```

Since: 2.3.1

Privilege level: public

Privilege: http://tizen.org/privilege/bluetooth

Warning: http://tizen.org/privilege/bluetooth.health (public level) has been deprecated since 2.4. Instead, use http://tizen.org/privilege/bluetooth.

Exceptions:

- WebAPIException
  - with error type SecurityError, if the application does not have the privilege to call this method.
  - with error type UnknownError, if any other error occurs.

Code example:

```
var adapter = tizen.bluetooth.getDefaultAdapter();
var healthProfileHandler = adapter.getBluetoothProfileHandler("HEALTH");
var connectedChannel = null;

var channelCallback = {
  onmessage: function(data) {
    console.log("data is received")
  },
  onclose: function() {
    console.log("channel is closed");
    connectedChannel.unsetListener();
  }
};

function healthRegisterSuccess(app) {
  console.log("Registered application: " + app.name);
  app.onconnect = function(channel) {;
    console.log("Health device is connected");
    connectedChannel = channel;
    connectedChannel.setListener(channelCallback);
  }
}

function healthRegisterError(e) {
  console.log("Failed to register application: " + e.message);
};

healthProfileHandler.registerSinkApplication(4100, "testSinkApp", healthRegisterSuccess, healthRegisterError);
```

[(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)](/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.28. BluetoothAdapterChangeCallback

The BluetoothAdapterChangeCallback interface specifies a set of methods to be invoked when the changes of Bluetooth adapter occur.

```
[Callback, NoInterfaceObject] interface BluetoothAdapterChangeCallback {
  void onstatechanged(boolean powered);
  void onnamechanged(DOMString name);
  void onvisibilitychanged(boolean visible);
};
```

Since: 2.3.1

Methods

[(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)](/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)***onstatechanged***

Called when the power state is changed.

```
void onstatechanged(boolean powered);
```

Since: 2.3.1

Parameters:

- powered: Flag indicating power state of local Bluetooth: *true* means power-on, *false* means power-off

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *onnamechanged*

Called when the name is changed.

```
void onnamechanged(DOMString name);
```

Since: 2.3.1

Parameters:

- name: Name of local Bluetooth

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *onvisibilitychanged*

Called when the visibility is changed.

```
void onvisibilitychanged(boolean visible);
```

Since: 2.3.1

Parameters:

- visible: Flag indicating visibility of local Bluetooth: *true* means that local Bluetooth is discoverable, *false* means that local Bluetooth is hidden from other devices

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.29. BluetoothDeviceSuccessCallback

The BluetoothDeviceSuccessCallback interface implements the success callback BluetoothAdapter.getDevice() and BluetoothAdapter.createBonding().

```
[Callback=FunctionOnly, NoInterfaceObject] interface BluetoothDeviceSuccessCallback {
  void onsuccess(BluetoothDevice device);
};
```

Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *onsuccess*

Called on success.

```
void onsuccess(BluetoothDevice device);
```

Since: 2.3.1

Parameters:

- device: BluetoothDevice object

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.30. BluetoothDeviceArraySuccessCallback

The BluetoothDeviceArraySuccessCallback interface that defines the success callback for BluetoothAdapter.getKnownDevices().

```
[Callback=FunctionOnly, NoInterfaceObject] interface BluetoothDeviceArraySuccessCallback {
  void onsuccess(BluetoothDevice[] devices);
};
```

Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *onsuccess*

Called when device information is ready.

```
void onsuccess(BluetoothDevice[] devices);
```

Since: 2.3.1

Parameters:

- devices: List of devices known to local Bluetooth adapter. Each element is a BluetoothDevice.

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.31. BluetoothDiscoverDevicesSuccessCallback

The BluetoothDiscoverDevicesSuccessCallback interface that defines the success callback for BluetoothAdapter.discoverDevices().

```
[Callback, NoInterfaceObject] interface BluetoothDiscoverDevicesSuccessCallback {
  void onstarted();
  void ondevicefound(BluetoothDevice device);
  void ondevicedisappeared(BluetoothAddress address);
  void onfinished(BluetoothDevice[] foundDevices);
};
```

Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *onstarted*

Called at the beginning of a device discovery process for finding the nearby Bluetooth device.

```
void onstarted();
```

Since: 2.3.1

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *ondevicefound*

Called when a new device is discovered in the process of inquiry/discovery.

```
void ondevicefound(BluetoothDevice device);
```

Since: 2.3.1

Parameters:

- device: Device that is found

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *ondevicedisappeared*

Called when a device is lost from proximity. After that, this device is no longer visible.

```
void ondevicedisappeared(BluetoothAddress address);
```

Since: 2.3.1

Parameters:

- address: Address of the device that is no longer in range or visible

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/) *onfinished*

Called when the device discovery process has finished.

```
void onfinished(BluetoothDevice[] foundDevices);
```

Since: 2.3.1

Parameters:

- foundDevices: Array of devices found in this discovery session

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.32. BluetoothSocketSuccessCallback

The BluetoothSocketSuccessCallback interface that defines the success method for BluetoothDevice.connectToServiceByUUID().

```
[Callback=FunctionOnly, NoInterfaceObject] interface BluetoothSocketSuccessCallback {
  void onsuccess(BluetoothSocket socket);
};
```

Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)***onsuccess***

Called when the connection to a service is ready.

```
void onsuccess(BluetoothSocket socket);
```

Since: 2.3.1

Parameters:

- socket: Socket to connect to the specified service on a remote device

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.33. BluetoothServiceSuccessCallback

The BluetoothServiceSuccessCallback interface implements the success callback for BluetoothAdapter.registerRFCOMMServiceByUUID().

```
[Callback=FunctionOnly, NoInterfaceObject] interface BluetoothServiceSuccessCallback {
  void onsuccess(BluetoothServiceHandler handler);
};
```

Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)***onsuccess***

Called when registering a service with the local device is successful.

```
void onsuccess(BluetoothServiceHandler handler);
```

Since: 2.3.1

Parameters:

- handler: Bluetooth service handler

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.34. BluetoothHealthApplicationSuccessCallback

The BluetoothHealthApplicationSuccessCallback interface that defines the success method for BluetoothHealthProfileHandler.registerSinkApplication().

```
[Callback=FunctionOnly, NoInterfaceObject] interface BluetoothHealthApplicationSuccessCallback {
  void onsuccess(BluetoothHealthApplication application);
};
```

Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*onsuccess*

Called when the application is registered successfully.

```
void onsuccess(BluetoothHealthApplication application);
```

Since: 2.3.1

Parameters:

- application: Registered health application

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.35. BluetoothHealthChannelSuccessCallback

The BluetoothHealthChannelSuccessCallback interface that defines the success method for BluetoothHealthProfileHandler.connectToSource() and the event callback for BluetoothHealthApplication.onconnect().

```
[Callback=FunctionOnly, NoInterfaceObject] interface BluetoothHealthChannelSuccessCallback {
  void onsuccess(BluetoothHealthChannel channel);
};
```

Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*onsuccess*

Called when a connection is established.

```
void onsuccess(BluetoothHealthChannel channel);
```

Since: 2.3.1

Parameters:

- channel: Connected health channel

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)

2.36. BluetoothHealthChannelChangeCallback

The BluetoothHealthChannelChangeCallback interface specifies a set of methods to be invoked when changes to heath channel occur.

```
[Callback, NoInterfaceObject] interface BluetoothHealthChannelChangeCallback {
  void onmessage(byte[] data);

  void onclose();
};
```

Since: 2.3.1

Methods

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*onmessage*

Called when the message is received.

```
void onmessage(byte[] data);
```

Since: 2.3.1

Parameters:

- data: Received data

(/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/)*onclose*

Called when the health channel is closed.

```
void onclose();
```

Since: 2.3.1

3. Related Feature

You can check if this API is supported with tizen.systeminfo.getCapability() and decide enable/disable codes that need this API.

To guarantee that the Bluetooth application runs on a device with Bluetooth feature, declare the following feature requirements in the config file:

- http://tizen.org/feature/network.bluetooth

To guarantee that the Bluetooth healthcare application runs on a device with Bluetooth health profile feature, declare the following feature requirements in the config file:

- http://tizen.org/feature/network.bluetooth.health

To guarantee that the Bluetooth Low Energy application runs on a device with Bluetooth Low Energy feature, declare the following feature requirements in the config file:

- http://tizen.org/feature/network.bluetooth.le

For more information, see Application Filtering. (https://developer.tizen.org/development/getting-started/web-application/understanding-tizen-programming/application-filtering)

4. Full WebIDL

```
module Bluetooth {
  typedef DOMString BluetoothAddress;

  typedef DOMString BluetoothUUID;

  enum BluetoothSocketState { "CLOSED", "OPEN" };

  enum BluetoothProfileType { "HEALTH" };

  enum BluetoothHealthChannelType { "RELIABLE", "STREAMING" };

  typedef DOMString BluetoothLESolicitationUUID;

  enum BluetoothAdvertisePacketType { "ADVERTISE", "SCAN_RESPONSE" };

  enum BluetoothAdvertisingState { "STARTED", "STOPPED" };

  enum BluetoothAdvertisingMode { "BALANCED", "LOW_LATENCY", "LOW_ENERGY" };

  [NoInterfaceObject] interface BluetoothManagerObject {
    readonly attribute BluetoothManager bluetooth;
  };
  Tizen (?redirect=/dev-guide/2.4/org.tizen.web.apireference/html/device_api/wearable/tizen/tizen.html#Tizen) implements
BluetoothManagerObject;

  [Constructor(DOMString uuid, DOMString data)]
  interface BluetoothLEServiceData {
    attribute BluetoothUUID uuid;
    attribute DOMString data;
  };
```