CONNECT IQ MOBILE SDK FOR ANDROID

A Guide To Building Companion Mobile Applications for Monkey C Applications

Table of Contents

Getting Started

Introduction

The Mobile SDK allows you to create companion applications that run on a user's phone and interact with your application on their wearable device. This allows for a more feature rich user experience where doing certain tasks on the wearable device might be tedious or resource intensive.

Adding the Mobile SDK to a project

The Mobile SDK is distributed as part of the Connect IQ SDK and can be found in the android/lib directory. To begin using the mobile SDK, copy the connectiq.jar file into the libs folder of your Android project.

Additional Requirements

In order to communicate with a Connect IQ device via your companion application the user must also install Garmin Connect Mobile onto their phone. All communication for companion applications running on Android goes through a Garmin Connect Mobile service to reach the device. When initializing the SDK this requirement is checked and initialization will fail if Garmin Connect Mobile is not installed. If true is passed to the auto UI parameter of initialize, a message is displayed to the user that they need to either install or upgrade Garmin Connect Mobile and provides them a way to go directly to the application in the Google Play Store. See "*Displaying an UI message automatically when initialization fails*" for more information.

Interacting with the SDK

All interactions between the companion application and the Connect IQ application are done via the ConnectIQ **class. To use this class you must first obtain an instance of the class and then initialize it.**

```
ConnectIQ connectIQ = ConnectIQ.getInstance(ConnectIQ.IQConnectType.<protocol>);
```

ConnectIQ.IQConnectType **provides two options:**

WIRELESS	**For communicating with the Connect IQ simulator or real device via BLE. This is the default.**
TETHERED	**For communicating with the Connect IQ simulator over the Android Debug Bridge.**

Initializing the SDK

Initializing the SDK is an asynchronous process and requires a ConnectIQListener **to handle returned states of the SDK. You must wait for the** onSdkReady() **call to be made before calling any additional API methods. Doing so beforehand will result in an** InvalidStateException**.**

```
connectIQ.initialize(context, true, new ConnectIQListener() {
        // Called when the SDK has been successfully initialized
    @Override
    public void onSdkReady() {

        // Do any post initialization setup.

        }

        // Called when initialization fails.
    @Override
    public void onInitializationError(IQSdkErrorStatus status) {

        // A failure has occurred during initialization.  Inspect
      // the IQSdkErrorStatus value for more information regarding
      // the failure.

        }

        ...
});
```

Displaying an UI message automatically when initialization fails

If initialization fails due to Garmin Connect Mobile not being installed on the user's phone or if it needs to be upgraded, a message can be displayed prompting the user to take action. You can tell the SDK to display this message automatically by passing true as the second parameter to the initialize() **method. By default, the UI will display a dialog message to the user asking them to take action. The strings that make up the dialogs are by default English only strings. These strings are fully customizable by simply adding some predefined strings to your projects** strings.xml **file.**

Customizable Strings

install_needed_title	Dialog title for when Garmin Connect Mobile needs to be installed.
install_needed_message	Dialog message for when Garmin Connect Mobile needs to be installed.
install_needed_yes	Button text for user to confirm they want to visit the Google Play Store to install Garmin Connect Mobile.
install_needed_cancel	Button text for user to cancel the dialog and not install Garmin Connect Mobile.
upgrade_needed_title	Dialog title for when Garmin Connect Mobile needs to be upgraded to a version that supports the SDK.
upgrade_needed_message	Dialog message for when Garmin Connect Mobile needs to be upgraded to a version that supports the SDK.
upgrade_needed_yes	Button text for user to confirm they want to visit the Google Play Store to upgrade Garmin Connect Mobile.
upgrade_needed_cancel	Button text for a user to cancel the dialog and not upgrade Garmin Connect Mobile.

Working With Devices

Finding Connect IQ-compatible devices

Before you can interact with a Connect IQ device, you must obtain a reference to an IQDevice object instance representing it. This is done by one of two methods.

getKnownDevices() will return a list on any Connect IQ device that has been paired within Garmin Connect Mobile. These devices may or may not be connected at the time the API is called.

```
List<IQDevice> paired = connectIQ.getKnownDevices();

if (paired != null && paired.size() > 0) {
    // get the status of the devices
    for (IQDevice device : paired) {
        IQDeviceStatus status = connectIQ.getStatus(device);
        if (status == IQDeviceStatus.CONNECTED) {
                // Work with the device
        }
    }
}
```

getAvailableDevices() will return a list of currently connected devices. Because these devices could become disconnected at any time, it is good practice to register to receive a notification when the device connects or disconnects. See "*Registering to receive a notification on each device status change*" for more information.

```
List<IQDevice> devices = connectIQ.getAvailableDevices();

if (devices != null && devices.size() > 0) {
    // Work with devices.

}
```

Listening for device events

You can request to be notified when the status of a device changes by calling registerForDeviceEvents(IQDevice, IQDeviceEventListener). Once registered any device status change will call the IQDeviceEventListener.onDeviceStatusChanged() with the new status. When you no longer need to receive updates for a device, you should call unregisterForDeviceEvents(IQDevice) to release any associated resources.

```
// Register to receive status updates
connectIQ.registerForDeviceEvents(device, new IQDeviceEventListener() {

        @Override
        public void onDeviceStatusChanged(IQDevice device, IQDeviceStatus newStatus) {

                // Handle new status

        }

});

// Get the current status
IQDeviceStatus current = connectIQ.getStatus(device);

// Unregister when we not longer need status updates
connectIQ.unregisterForDeviceEvents(device);
```

Possible Device statuses

CONNECTED	The device is connected and can be communicated with.
NOT_CONNECTED	The device is paired with Garmin Connect Mobile but is not currently connected and cannot be communicated with.
NOT_PAIRED	The device is not paired with Garmin Connect Mobile and cannot be communicated with.

Working With Apps

Obtaining an instance of IQApp

Apps are represented in the Mobile SDK as instances of the IQApp class. While you can create an IQApp instance on your own, it is recommended to obtain a fully populated IQApp instance via the getApplicationInfo() method. You can determine if your Connect IQ application is installed on the user's device by calling getApplicationInfo() passing the IQDevice and the Application UUID Identifier for the application you want to check. This will return an IQApp object that can be inspected via the getStatus() method to determine the status of your application on the device. If the status is INSTALLED, the version number will also be populated so you can determine if the user has the latest version of your application.

```
IQApp app = connectIQ.getApplicationInfo(device, MY_APPLICATION_ID);

if (app != null) {
    if (app.getStatus() == INSTALLED) {
        if (app.getVersion() < MY_CURRENT_VERSION) {
            // Prompt the user to upgrade
        }
    }
}
```

Possible Application Statuses

INSTALLED	The application is installed on the device and the version information has been populated.
NOT_INSTALLED	The application is not currently installed on the device but is supported.
NOT_SUPPORTED	The application is not installed on the device and is not supported by the device.

Opening the Connect IQ store

If the user does not have your applications installed (status of NOT_INSTALLED), or needs to upgrade to the latest version, you can open the Connect IQ store directly to your application. Simply call openStore() passing the Application Identifier for your application.

```
connectIQ.openStore(MY_APPLICATION_ID);
```

Sending messages

You can send messages to your Connect IQ application on a connected device using any of the Java equivalent Monkey C data types (see *Supported Data Types* table below). Calling sendMessage() will deliver the message to your applications mailbox.

```
List<Object> message = new ArrayList<String>() {“hello pi”, 3.14159};

IQMessageStatus status = connectIQ.sendMessage(device, app, message);

if (status != IQMessageStatus.SUCCESS) {

       // Evaluate status for the cause of the failure.

}
```

Receiving messages

In order to receive data messages from a Connect IQ application, you must first register to receive application events. Once registered via registerForAppEvents(), when a new message arrives from the Connect IQ application, it will be delivered to the onMessageReceived() method of the listener passed when registering. When you no longer wish to receive incoming messages, you should call unregisterForAppEvents() to release any associated resources.

A companion app may register to receive messages from multiple apps across many devices. However, multiple companion apps cannot be registered to receive messages from the same ConnectIQ application. The SDK will override any previous registrations with each call to registerForAppEvents().

```
// Register to receive messages from our application
connectIQ.registerForAppEvents(device, app, new IQApplicationEventListener() {

       @Override
       public void onMessageReceived(IQDevice device, IQApp app,
                                     List<Object> messageData,
                                     IQMessageStatus status) {
              // First inspect the status to make sure this
              // was a SUCCESS.  If not then the status will indicate why there
              // was an issue receiving the message from the Connect IQ application.
              if (status == IQMessageStatus.SUCCESS) {
                     // Handle the message.
              }
       }

});

// unregister when we no longer care about messages coming from our app.
connectIQ.unregisterForAppEvents(device, app);
```

Supported Data Types

Java Data Type	Monkey C Type	Notes
int, Integer	Integer	
float, Float	Float	
boolean, Boolean	Boolean	
String	String	
List<?>	Array[]	List must contain only supported data types. If the list contains unsupported data types, an exception will be thrown.
Map<?,?>	Dictionary	Map keys and values must be supported data types. If the map contains unsupported data types, an exception will be thrown.