Weightless Starter Kit
Java Sample Code

Revision History

Revision Code	Date	Description	Comments
1.0.0	Oct 5 2017	Firmware 1.0.0	First Release
1.0.5	Sept 5 2017	Firmware 1.0.5	
1.0.7	Nov 11 2017	Firmware 1.0.7	Added Sample Code location
1.0.8	Jan 8 2018	Firmware 1.0.8	
1.0.9	May 9 2018	Firmware 1.0.9	

Table Of Contents

Sample Code

All sample code mentioned in this guide can be found in the *Downloads* section of Ubiik Cloud.
https://wpkit.ubiik.com/analytics/downloads/

Lower Level Example

If you want to parse the messages yourself and/or store them yourself, you can manage the connection directly using the class *PSConnectionManager*.

```
BlockingQueue<byte[]> inMessagesQueue = new LinkedBlockingQueue<byte[]>();
this.connectionManager = new PSConnectionManager(host, port, this, inMessagesQueue);
this.msgProcessor = new MessagesProcessor(inMessagesQueue);

this.msgProcessorThread = new Thread(this.msgProcessor);
this.msgProcessorThread.start();

// Connect
try {
        // The application will connect and authenticate.
        // We will be notified when the authentication has finished
        this.connectionManager.connect();
} catch(PSConnectionException e) {
        this.authenticationFinished(false);
}
```

Incoming messages will be stored in *inMessagesQueue*
You need to implement the interface *PSConnectionManagerDelegate*, which exposes the methods
public void connectionFailed(PSConnectionErrorCode errorCode);
Method called if the connection fails
public void authenticationFinished(boolean autheticated);
Method called after authentication finished. It sends true if authentication succeeded or false if it failed

Please see wp-ps-sample-code

Higher Level Example

If you don't want to parse the messages and/or store them yourself, you can use the classes and interfaces described next.

Connect To Protocol Stack

Connection to the Protocol Stack is managed via an instance of *MessageController*.

```
DevicesController devicesController = new DevicesController();
MessageController = new MessageController(devicesController);
messageController.connect(host, port); // port = 7979
```

Get Notified on Connection Status Changes

You need to implement the interface *PSConnectionListener* which exposes the following methods:

public void didConnect();

Method called after the Application connected to the protocol stack Authentication is still pending

public void didAuthenticate(boolean success);

Method called after the authentication response has been received. If Authentication was successful, you may start sending messages.

public void didDisconnect();

Method called after the Application disconnected from the protocol stack

Add an instance of your class as a connection observer:

```
messageController.addConnectionObserver(this);
```

Send Message

Sending messages to the Protocol Stack is also managed via the same instance of *MessageController*.

Create an instance of a subclass of PSMessage and send it via sendMessage():

```
…
PSDownlink downlink = new PSDownlink(data, target, endpoint);
messageController.sendMessage(downlink);
```

Get Notified when New Messages are Received

To get Notified when new messages are received, your class must implement the interface *MessagesObserver* and needs to be added to the listeners.

This interface exposes the method:

public void update();

Method called when a change in the messages model happened

```
PSMessageStorage.getInstance().addObdserver(this);
```

How to Get Stored Messages

From the Message Storage you can retrieve a copy of the currently stored Uplinks, Downlinks and Commands:

```
PSMessageStorage ms = PSMessageStorage.getInstance();
List<PSUplink> uplinks = ms.getUplinks();
List<PSDownlink> downlinks = ms.getDownlinks();
List<PSCommand> commands = ms.getCommands();
```

Get Notified When The Response For A Command Is Received

To get Notified when a response to a command is received, your class must implement the interface *PSCommandDelegate,* which exposes the method:

public void commandandReceivedResponse(PSCommand command);

When creating an instance of PSCommand, you must set the instance as the PSCommandDelegate.

```
PSCommandGetBaseFrequency command = new PSCommandGetBaseFrequency(this);
messageController.sendMessage(command);
```

Get Notified on Device Changes

The interface *DevicesObserver* exposes the method:

public void updateDevices();

You will get notified when:

- A New End Device is detected
- An End Device status changed (Connection Status, Uplinks and Downlinks count)
- Base Station ID is received
- Devices added or removed from Multicast Groups
- SIB changed
- Base Station parameter changed
- Multicast Groups Changed
- Base Station Run Mode Changed

In order to get notified, you must add an instance of your class to your instance of devicesController:

```
devicesController.addObdserver(this);
```

Uplinks Demos

Uplinks Demos includes two demo applications that process different kinds of data.
The purpose of these applications is to show how to receive and process uplinks

The base class *UplinksDemoViewController* implements the interface *MessagesObserver*. Every time it is notified that a new Message is received, the method *refreshData* will be called. If there are new uplinks, *processUplink* will be called. This method is redefined by each application in order to look a Temperature/Humidity sample or a RSSI/Packet count sample.

The main difference between both applications is in the method *getSampleFromUplink* in each *ViewController*.
Please see wp-ps-config-tool-demos