FLYFISH
TECHNOLOGIES

***Document:*** **hiDBLUE - APIs Software Interface Reference**, Android

| | |
|---|---|
| *Audience:* | **Partners, Designers, Developers - APIs Integrators** |
| *Confidentiality Level:* | **Public** |
| *Document Owner:* | Ivan Zilic |
| *Version:* | 1.2 |
| *Date:* | January 16, 2014 |
| *Number of pages:* | 12 |
| *File name:* | Z:\Dev\Doc\hiDBLUE\hiDBLUE-APIs_SIR_Android.docx |

*Changes log:*

| *Version* | *Date* | *Author* | *Reason for update* |
|---|---|---|---|
| 1.0 | Apr 17, 2013 | Peter Kelpec | Initial document |
| 1.1 | May 13, 2013 | Ivan Zilic | Review, minor clarifications, ownership change |
| 1.2 | Jan 16, 2014 | Martina Krpan | Fixed codename text |

# Table of Contents

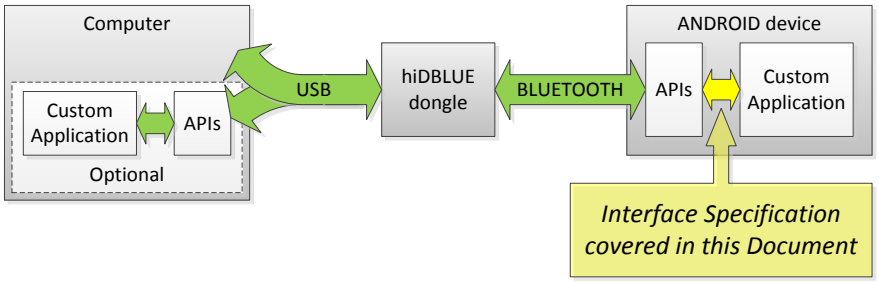
# 1. Scope

This is a reference document intended for Developers implementing their applications using hiDBLUE APIs, on Android platform.

Additionally, this document can give proper hiDBLUE features overview to Products Designers, during their products initial brainstorming and design phases.

# 2. General Overview

The APIs library is middleware between Custom Application(s) and Android native Bluetooth driver. A block scheme below points to the Interface under scope here:




APIs library wraps all hiDBLUE-specific communication details. As a result, it is sufficient to call only API functions covered in this document for mastering complete set of hiDBLUE features.

*Note: "Computer" stands for any device with Operating System supporting HID USB devices (like Apple OS X, MS Windows, Linux, BSD, Android, etc.) and USB host port present.*

# 3. Deployed Environment and Prerequisites

The APIs are packed in a single jar binary file. Package name is com.flyfish_tech.hidblue.api.

The following table summarizes requirements:

| Parameter | Value |
|---|---|
| Android version | 2.0 or later |
| Bluetooth | Not absolutely required.<br>However, if not present, most functions will fail, setting adequate description. |
| Permissions required | android.permission.BLUETOOTH<br>android.permission.BLUETOOTH_ADMIN |
| Automatic update | Not implemented.<br>Maintainers of applications are in charge to incorporate updates processes. |
| Minimum SDK version | 5 |
| Target SDK version | 10 |

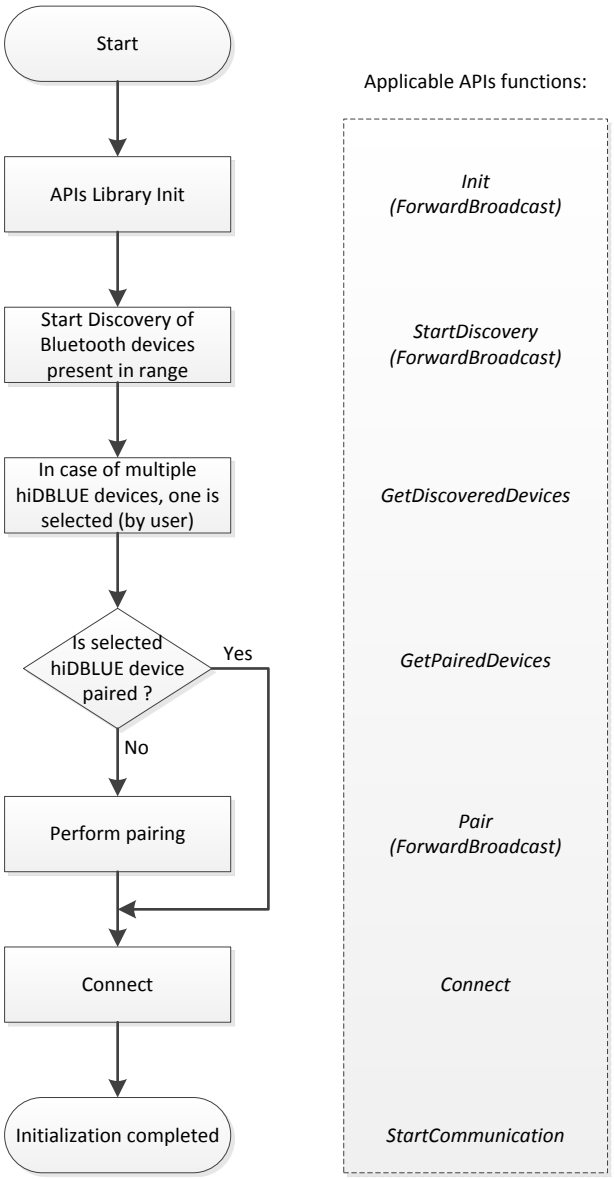
# 4. APIs Usage

Make sure that single instance of APIs is simultaneously run in all your applications.

APIs behavior and usage is based on Internal State Machine model. The state information is available via APIs query calls. Additionally, the state might be forced via a dedicated APIs function call, but it makes sense to do this only during development, in few rare debugging isolated cases.

## 4.1 Initial procedure

Initial procedure is straight-forward. The workflow is the following:



## 4.2 Data exchange

The following four data paths are implemented:

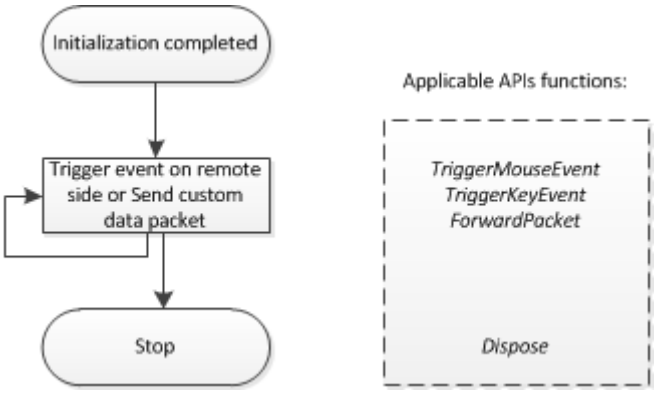
| Purpose | Source | Destination | Note |
|---|---|---|---|
| Mouse | Android device | Computer | |
| Keyboard | Android device | Computer | |
| Custom | Computer | Android device | Dedicated Application is mandatory on Computer side |
| Custom | Android device | Computer | Dedicated Application is expected on Computer side |

All listed paths can be used simultaneously. Each data packet is internally buffered on its path, with periodic retries performed, until successfully delivered. Buffering mechanism is FIFO, which guarantees unchanged data order.

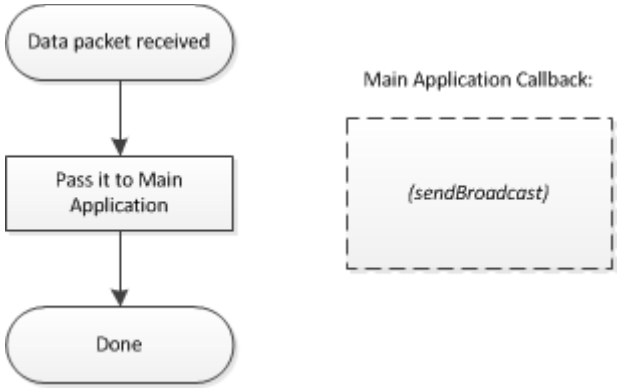
Mouse and Keyboard paths offer one-way data flow only. Destination side doesn't need any additional custom code provided by you.

hiDBLUE offers also two custom paths, typically combined into a single custom duplex channel. Format and content of this data can be freely chosen and is not predefined by APIs in any way. It is up to applications on both ends to compose and interpret this data.

After successfully performed Initial procedure, the workflow when Android device is data source is the following:




When data is initiated on Computer side, the Main application on Android side receives data packet from APIs via local broadcast:



# 5. API Methods

## 5.1 Init

| | |
|---|---|
| Syntax | `boolean Init(Context context, BroadcastReceiver broadcast)` |
| Parameters | • context – content of the Application, *this*<br>• broadcast – broadcast receiver in the Application intended for receiving Bluetooth broadcasts and passing them to API |
| Return value | boolean, indicates success/fail result of the method |
| Discussion | This method performs various initial internal actions.<br>If Bluetooth radio is off, this method will turn it on.<br>The following broadcast filters are set for given broadcast receiver:<br>• BluetoothAdapter.ACTION_STATE_CHANGED<br>• BluetoothDevice.ACTION_FOUND<br>• BluetoothAdapter.ACTION_DISCOVERY_FINISHED<br>• BluetoothDevice.ACTION_BOND_STATE_CHANGED<br><br>Typical Method fail result would be caused by absent Bluetooth. Call `GetStateDetails` to get actual error code. |

## 5.2 StartDiscovery

| | |
|---|---|
| Syntax | `boolean StartDiscovery()` |
| Parameters | None |
| Return value | boolean, indicates success/fail result of the method |
| Discussion | This method initiates scan of Bluetooth devices present in range.<br>The scan typically takes about 12 seconds to complete. This method returns immediately, leaving APIs in scan (busy) mode.<br>During scan mode, the state returned by `GetState` is `STATE_BUSY` and `GetStateDetails` returns `DETAIL_DEVICES_SCAN_IN_PROGRESS`.<br>The scan procedure is completed when state changes to `STATE_DEVICES_SCAN_COMPLETED`. |

## 5.3 GetDiscoveredDevices

| | |
|---|---|
| Syntax | `String[][] GetDiscoveredDevices()` |
| Parameters | None |
| Return value | 2D strings array.<br>One row per discovered device (first array dimension).<br>The array contains two columns (second array dimension). Columns are Name of the device and related MAC address. |
| Description | Returned data is collected during bluetooth devices scan, initiated by `StartDiscovery` call.<br>Returned list is limited to hiDBLUE discovered devices only. |

## 5.4 GetPairedDevices

| | |
|---|---|
| Syntax | `String[][] GetPairedDevices()` |
| Parameters | None |
| Return value | 2D strings array.<br>One row per paired device (first array dimension).<br>The array contains two columns (second array dimension). Columns are Name of the device and related MAC address. |
| Description | Returned list is limited only to hiDBLUE paired devices which were just found during discovery scan. |

## 5.5 Pair

| | |
|---|---|
| Syntax | `boolean Pair(String MAC)` |
| Parameters | • MAC – MAC address of the device. String should contain six groups of two hexadecimal digits, separated by colons. Example: *00:06:66:4D:54:86* |
| Return value | boolean, indicates success/fail result of the method |
| Description | This method invokes Android's native pairing UI.<br>Given MAC address must belong to any hiDBLUE device found during discovery scan.<br><br>Pairing PIN code is "ff". |

## 5.6 Connect

| | |
|---|---|
| Syntax | `boolean Connect(String MAC)` |
| Parameters | • MAC – MAC address of the device. String should contain six groups of two hexadecimal digits, separated by colons. Example: *00:06:66:4D:54:86* |
| Return value | boolean, indicates success/fail result of the method |
| Description | Given MAC address must belong to any hiDBLUE device found during discovery scan.<br>Additionally, the device already has to be paired. |

## 5.7 StartCommunication

| | |
|---|---|
| Syntax | `boolean StartCommunication()` |
| Parameters | None |
| Return value | boolean, indicates success/fail result of the method |
| Description | This method launches communication threads.<br>At this point, the Main application has to be prepared for receiving data packet (if applicable). |

## 5.8 Dispose

| | |
|---|---|
| Syntax | `void Dispose()` |
| Parameters | None |
| Return value | None |
| Description | This method must be executed at the end of the APIs usage. It performs cleanup procedure.<br>APIs can be re-initialized afterwards with `Init` method and complete Initial procedure run. |

## 5.9 TriggerMouseEvent

| | |
|---|---|
| Syntax | `boolean TriggerMouseEvent(byte xDiff, byte yDiff, byte wheelDiff, boolean button1Pressed, boolean button2Pressed, boolean button3Pressed, boolean holdButtons)` |
| Parameters | • xDiff – relative mouse pointer movement on X axis. Valid values are -127 to 127<br>• yDiff – relative mouse pointer movement on Y axis. Valid values are -127 to 127<br>• wheelDiff – relative wheel movement. Valid values are -127 to 127<br>• button1Pressed – set left button state to either clicked (true) or not (false)<br>• button2Pressed – set right button state to either clicked (true) or not (false)<br>• button3Pressed – set middle button state to either clicked (true) or not (false)<br>• holdButtons – when set to true, buttons set above remain in pressed position, else they are immediately released |
| Return value | boolean, indicates success/fail result of the method |
| Description | This method controls mouse behavior on remote side. Mouse data from hiDBLUE is not |

| | |
|---|---|
| | exclusive, resulting in cumulative effect in case of additional mouse existence. |

## 5.10 TriggerKeyEvent

| | |
|---|---|
| Syntax | `boolean TriggerKeyEvent(byte codeKeyMain, byte codeKeyAlternate, boolean clickOnly)` |
| Parameters | • codeKeyMain – USB keyboard key code, see Description below<br>• codeKeyAlternate – USB keyboard alternate code, see Description below<br>• clickOnly – when set to false, key set above remains in pressed position, else it is immediately released |
| Return value | boolean, indicates success/fail result of the method |
| Description | This method controls keyboard behavior on remote side. Keyboard data from hiDBLUE is not exclusive, resulting in cumulative effect in case of additional keyboard existence.<br><br>Codes to be used as parameters value are listed in "Universal Serial Bus HID Usage Tables" document, Chapter 10. The document is available for download from USB.org webpage (*http://www.usb.org/developers/devclass_docs/Hut1_12v2.pdf*) |

## 5.11 ForwardPacket

| | |
|---|---|
| Syntax | `boolean ForwardPacket(byte dataPacket[])` |
| Parameters | • dataPacket – array of bytes to be delivered to Computer side |
| Return value | boolean, indicates success/fail result of the method |
| Description | This method is used when data packet is to be delivered to Computer's Custom Application. Maximum parameter array size is 8 bytes. When larger amount of data needs to be sent, it shell be split into smaller chunks size up to 8 bytes and send in a sequence. The same chunks order appearance is guaranteed on receiver side. |

## 5.12 ForwardBroadcast

| | |
|---|---|
| Syntax | `void ForwardBroadcast(Intent intent)` |
| Parameters | • intent – facility for processing Bluetooth broadcasts |
| Return value | None |
| Description | This method is called on receiving event of broadcasts. If the broadcast is not Bluetooth-related, it is silently ignored by APIs. |

## 5.13 GetState

| | |
|---|---|
| Syntax | `int GetState()` |
| Parameters | None |
| Return value | Integer value representing internal APIs state. Values set is listed in Appendix section below. |
| Description | Main application should call this method to check APIs internal state prior to performing any step during Initial procedure. |

## 5.14 GetStateDetails

| | |
|---|---|
| Syntax | `int GetStateDetails()` |
| Parameters | None |
| Return value | Integer value representing internal APIs substate. Values set is listed in Appendix section below. |
| Description | Main application should call this method to gather additional information about State. Typical usages are in case of errors, when this method returns type of error. |

## 5.15 GetLastException

| | |
|---|---|
| Syntax | `Exception GetLastException()` |
| Parameters | None |
| Return value | Exception containing error details. |
| Description | This method is applicable when `GetStateDetails` method returns `ERROR_EXCEPTION`. In this case further information is available by calling this method. |

## 5.16 ResetErrorState

| | |
|---|---|
| Syntax | `void ResetErrorState()` |
| Parameters | None |
| Return value | None |
| Description | After the error reason is solved, this method is to be called to restore pre-error state. |

## 5.17 ForceState

| | |
|---|---|
| Syntax | `void ForceState(int state)` |
| Parameters | • state – numeric state of internal APIs state |
| Return value | None |
| Description | This function is intended only for debugging purpose of Main application, during development phase.<br><br>Make sure that your production code does not include any call to this method. |

# 6. Main application skeleton

Two Broadcast Receivers are to be implemented in the Main application to serve APIs. First one is mandatory, involved during initial procedure, and the second one during Data exchange.

- Initial procedure expects some broadcasts containing Bluetooth events data. This data needs to be grabbed with Broadcast Receiver implemented in the Main application and forward to the APIs. Example of sufficient related code in the Main application is the following:

```
private final BroadcastReceiver BluetoothEventsReceiver = new BroadcastReceiver() {
	@Override
	public void onReceive(Context context, Intent intent) {
		API.ForwardBroadcast(intent);
	}
};
```

In this case, `BluetoothEventsReceiver` is to be passed as a second parameter to Init APIs method call.

- Second Broadcast Receiver acts as entry point for data packets received from Computer side. Example of this receiver is the following:

```
private final BroadcastReceiver APIDataReceiver = new BroadcastReceiver() {
	@Override
	public void onReceive(Context context, Intent intent) {
		Bundle bundle = intent.getBundleExtra("Packet");
		if (bundle.getString("Content") == "Raw") {
			int datalen = bundle.getInt("DataLen");
			byte[] data = new byte[datalen];
			data = bundle.getByteArray("Data");
			ProcessReceivedAPIData(data);
		}
	}

};
```

In this example, `ProcessReceivedAPIData` is to be implemented in the Main application to interpret received data.

# 7. Appending

## A. API Constants

Internal State Machine constants:

| Name | Value |
|---|---|
| STATE_LIB_INIT | 0 |
| STATE_DEVICES_SCAN_REQUIRED | 1 |
| STATE_DEVICES_SCAN_COMPLETED | 2 |
| STATE_DEVICE_PAIRED | 3 |
| STATE_CONNECTED | 4 |
| STATE_ONLINE | 5 |
| STATE_DISPOSED | 6 |
| STATE_ERROR | 7 |
| STATE_BUSY | 8 |

Internal State Machine Details constants:

| Name | Value |
|---|---|
| DETAIL_NONE | 0 |
| DETAIL_TURNING_BLUETOOTH_ON | 1 |
| DETAIL_DEVICES_SCAN_IN_PROGRESS | 2 |
| DETAIL_DEVICE_PAIRING_IN_PROGRESS | 3 |
| ERROR_BLUETOOTH_NOT_PRESENT | 4 |
| ERROR_NO_DEVICES | 5 |
| ERROR_PAIRING_FAILED | 6 |
| ERROR_PEER_NOT_FOUND | 7 |
| ERROR_DEVICE_NOT_PAIRED | 8 |
| ERROR_SOCKET_NOT_AVAILABLE | 9 |
| ERROR_STREAM_NOT_OPENED | 10 |
| ERROR_IMPROPER_USAGE | 11 |
| ERROR_BUFFER_OVERFLOW | 12 |
| ERROR_EXCEPTION | 13 |

# Codename :: Wheel

WHO INVENTED THE WHEEL ?

Nobody knows.

It is believed that the first wheels were used in Mesopotamia in the 4th millennium B.C. The wheels supposedly spread all over the world from there.

Some ascribe the invention of the wheel to prehistoric Europe.

*World oldest wooden wheel with axle*
*City Museum Ljubljana, photo Matevž Paternoster*

The oldest wooden wheel in the world, which is over 5000 years old according to the analyses, was found while researching the crannog settlement at location Stara gmajna pri Vrhniki, Slovenia.

In the spring of 2002, a team from the Slovenian Institute of Archaeology continued with the project of wood sampling at the mentioned location. A surprise awaited them in one of the drainage ditches. Besides rich findings and two dugouts, they also found the remains of a wooden wheel at the bottom of the ditch that had already been partially damaged by construction machines when they were deepening the ditch. The ditch was widened at the site of the discovery so that they also found the axle that had become separated from the wheel.

The wheel was composed of two ash wood plates that were connected by four oak wedges and had a rectangular aperture in the center, where the axle was mounted. Its diameter was 72cm (28 inch) and it had a thickness of approximately 5cm (2 inch). According to the dendrochronological research the wheel was made from ash wood that comprises both plates, a trunk with the diameter of at least 40cm (16 inch) and was made from a tree that was approximately 80 years old. The choice of ash was not coincidental, because of its strength and because it grew in the vicinity of the crannogs and because it can grow to the dimensions that were needed for large boards without any knots. The axle was constructed from one piece of oak wood and was 124cm (49 inch) long. The ending of the axle was rectangular and fitted into the opening of the wheel. The axle was attached to the wheels with oak wood wedges, which meant that the axle rotated together with the wheels.

According to the opinion of the experts, the wheel and the axle were a part of a two-wheel cart – a pushcart.

The wheel and the axle were dated on the basis of stratigraphic data with dendrochronological research and with the radiocarbon method. The wheel is approximately 5,150 years old and is contemporary with the settlement of Stara gmajna, where it was discovered. Radiocarbon dating was performed in the VERA laboratory (Vienna Environmental Research Accelerator) in Vienna, Austria.

(Source: http://www.koliscar.si)

***FLYFISH TECHNOLOGIES headquarters is located about 20 kilometers (12 miles) north-east from the location where this world oldest wooden wheel has been found.***