Z Watch Programming Guide

Rev. 1.0

October 9, 2013

chenee@smartdevices.com.cn

This Document help you to develop Application for Z Watch™ with the Z Watch SDK V1.0. Including the steps to establish communication between Z Watch and Android mobile phone through Bluetooth, Useful API, As well as the introduction of the watch faces development and code examples.

Setting up a development environment

Before you start developing for Z Watch, there are a few tools that you need.

1\ Ubuntu 9.10 or later (Windows XP or later)

2\ Java version 1.6 or later

3\ Android SDK (Version 4.1 is preferred)

4\ Eclipse 3.5+ ADT20, or Google Android Studio

5\ Android phone running at least Android 2.3, and Z Watch phone assistant APK installed

(you can download it at: http://www.smartdevices.com.cn/zh.html)

6\ SmartQ Z Watch (http://shop.smartdevices.com.cn/goods.php?id=54)

How to use the assistant APK:

1. adb push 手表端同步程序.apk /sdcard/

2. adb shell;su;mount -o remount,rw /system;busybox cp /sdcard/手表端同步助手.apk /system/app/IndroidSyncWatch.apk

3. Install Z Watch phone assistant APK, you can find it in the ZIP file attached with this document. Or download it here:http://www.smartdevices.com.cn/support/download/Z1/

Connection API Description

The Communication service module function as a global Damon Service to establish a Bluetooth connection for communication. Your application don't need to "create" the service, instead, "bind" the communication service before your application need to send/receive data, and "unbind" the service when your application exiting to reduce the overhead of system.

1 send data

First your should pair the Z Watch with your Android mobile phone through Bluetooth, then start the Z Watch assistant APK on your phone,after that you can run the application to send data.

Note : It is recommended to test the connection before send useful data, for example, sending some handshake testing data.

Declaration

boolean send(SyncData data);

Parameters

name	Description

SyncData	data communication entity class , Key-Value pair , similar to bundle

Return

boolean： is data send OK.

Sample Code

As mentioned at the beginning,before the useful data is sent, you'd better send some test data for the handshake to ensure connection status.

```
SyncData data = new SyncData();
data.putString(DemoUtil.KEY_CONNECT, "connect... ");
try {
    mDemoModule.send(data);
    while (!mDemoModule.isConnected()) {
        Thread.sleep(100);
    }
} catch (Exception e) {
    e.printStackTrace();

}
```

2 receive data call back

Synopsis

void onRetrive(SyncData data)

Return

void

Description

This function is called back when data communication completed, with the returned data stored in “SyncData”. Applications on both Z Watch and Phone share this function, so you should detect witch side you are before processing the data.

3 Connection status changed

Synopsis

void onConnectionStateChanged(**boolean** connect)

Parameters

name	Description
connect	Current connection status. true:connected ;false : disconnected

Return

void

Discription

When disconnect detected ,your application should re-connect or notify the user.

4 Mode changed

Synopsis

void onModeChanged(**int** mode)

Parameters

name	Description
mode	mode: SAVING_POWER_MODE = 0; RIGHT_NOW_MODE = 1(normal

	mode);

Return

void

5 Current connection status

Synopsis

boolean isConnected()

Return

boolean : true , connected ,false : disconnected

Discription

You can call this function to get current connection status.

6 Send file

Synopsis

boolean sendFile(File file, String name, **int** length)

Parameters

name	Could be NULL	value	description
file	NO	File	File object
name	NO	String	File name
length	NO	**int**	File size

7 File send completion call back

Synopsis

void onFileSendComplete(String fileName, **boolean** success)

Parameters

name	Description
fileName	File name
success	Is send success

Return

void

Description

This function will be called back when file sent finished.

8 File receive completion call back

Synopsis

void onFileRetriveComplete(String fileName, **boolean** success)

Parameters

name	Description
fileName	File name
success	Is received success

Return

Void

Description

This function will be called back when file sent finished.

9 Channel create call back

Synopsis

void onChannelCreateComplete(ParcelUuid uuid, **boolean** success,

boolean local)

Parameters

name	Could be null	value	description
uuid	NO	ParcelUuid	Bluetooth communication channel ID
success	NO	**boolean**	Is success
local	NO	**boolean**	Is local

Return

Void

Description

This function will be call back when communication channel created.

10 Bluetooth pair information clean call back

Synopsis

void onClear(String address)

Parameters

name	Could be NULL	value	Description
address	NO	String	Device address

Return

Void

Description

This function will be called back when Bluetooth pair information cleaned

11 Platform detect

Synopsis

boolean isWatch()

Return

boolean: true: Z Watch; false:Android Phone。

Description

Call this function to detect which side your application running on

12 Create custom channel

Synopsis

void createChannel(ParcelUuid uuid)

Parameters

Name	Could be NULL	Value	Description
uuid	NO	ParcelUuid	Bluetooth communication channel ID

Return

void

Description

Create custom channel,you need destroy this channel when your application exit.

13 Destroy custom channel

Synopsis

void destroyChannel(ParcelUuid uuid)

Parameter

Name	Could be NULL	Value	Description
uuid	NO	ParcelUuid	Bluetooth communication channel ID

Return

Void

Description

Destroy Bluetooth channel

14 send data through custom channel

Synopsis

void sendOnChannel(SyncData data, ParcelUuid uuid)

parameters

name	Could be NULL	Value	description
data	NO	SyncData	Communication data
uuid		ParcelUuid	Channel ID

Return

Void

Description

Send data through custom channel

Communication Data Class Description

We define class “SyncData” as data-interchange class to transfer the data between Android Phone and Z Watch. SyncData Class support various types of Key-Value pair data storage, like bundle data, used as communication entity . The following lists the supported data type of the data interface.

```
public void writeToParcel(Parcel dest, int flags)

public void putBoolean(String key, boolean b)

public void putBooleanArray(String key, boolean[] array)

public void putByte(String key, byte value)

public void putByteArray(String key, byte[] value)

public void putChar(String key, char value)

public void putCharArray(String key, char[] value)

public void putDouble(String key, double value)

public void putDoubleArray(String key, double[] value)

public void putFloat(String key, float value)

public void putFloatArray(String key, float[] value)

public void putInt(String key, int value)

public void putIntArray(String key, int[] value)

public void putLong(String key, long value)

public void putLongArray(String key, long[] value)

public void putShort(String key, short value)

public void putShortArray(String key, short[] value)

public void putString(String key, String value)

public void putStringArray(String key, String[] value)

private void typeWarning(String key, Object value, String className,

        ClassCastException e)

private void typeWarning(String key, Object value, String className,
```

Object defaultValue, ClassCastException e)

public boolean getBoolean(String key, **boolean** defaultValue)

public boolean[] getBooleanArray(String key)

public byte getByte(String key)

public Byte getByte(String key, **byte** defaultValue)

public byte[] getByteArray(String key)

public char getChar(String key)

public char getChar(String key, **char** defaultValue)

public char[] getCharArray(String key)

public double getDouble(String key)

public double getDouble(String key, **double** defaultValue)

public double[] getDoubleArray(String key)

public float getFloat(String key)

public float getFloat(String key, **float** defaultValue)

public float[] getFloatArray(String key)

public int getInt(String key)

public int getInt(String key, **int** defaultValue)

public int[] getIntArray(String key)

public long getLong(String key) {

public long getLong(String key, **long** defaultValue)

public long[] getLongArray(String key)

public short getShort(String key)

```
public short getShort(String key, short defaultValue)

public short[] getShortArray(String key)

public String getString(String key, String defaultValue)

public void put(String key, Object obj)

public Object get(String key)

public Set<String> keySet()
```

Watch Faces Developer Guide

Description

Watch faces application are customed theme plugins for Z Watch,managed by uniform naming convention. It allowed the user to switch Watch Faces dynamically in Z Watch.
The development environment of Watch faces consistent with the normal applications. Watch face app is a standard Android widget ,could be build in Application or separate as an APK.

To be add to the Desktop automatically by launcher, the class name of watch face application must be prefixed with “WatchFace” , such as “WatchFaceClockProvider”. if no, it should be add manually by click “Add Plugin” button on Z Watch.

Example

Watch face plugin is standard Android widget,we take the "AnalogClock" for example as below to show you how to develop a new watch face.

Compile and install this watch face application, restart Z Watch, then you can find it among the Watch Faces.It does not need to restart Z Watch in the later debug process.

Java.class

```
import android.appwidget.AppWidgetManager;

import android.appwidget.AppWidgetProvider;

import android.content.Context;

import android.widget.RemoteViews;


/**

 * Simple widget to show analog clock.

 */

public class WatchFaceClockProvider extends AppWidgetProvider {


    public void onUpdate(Context context, AppWidgetManager appWidgetManager,

            int[] appWidgetIds) {

        super.onUpdate(context, appWidgetManager, appWidgetIds);

        RemoteViews views = new RemoteViews(context.getPackageName(),

                R.layout.analog_appwidget);
```

```
            appWidgetManager.updateAppWidget(appWidgetIds, views);
        }
}
```

layout xml：analog_appwidget.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="240px"
    android:layout_height="240px" >

    <AnalogClock
        android:id="@+id/analog_appwidget_clock4"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</RelativeLayout>
```

Xml file：widget_provider.xml

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:initialLayout="@layout/analog_appwidget"
    android:minHeight="240px"
```

```
    android:minWidth="240px"
    android:updatePeriodMillis="0" >

</appwidget-provider>
```

Add watch face to AndroidManifest

```
        <receiver
            android:name=".WatchFaceClockProvider"
            android:enabled="true"
            android:exported="true"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
            </intent-filter>

            <meta-data
                android:name="android.appwidget.provider"
                android:resource="@xml/widget_provider" />
        </receiver>
```

Weather Information

You can add weather module to watch face,the weather information is broadcast by application on the phone. To receive this broadcast you need register. Weather action is :

String *ACTION_WEATHER_CHANGE* = "cn.indroid.action.weather.freshwidget";

The broadcast use JSON as data format:

weather_full:

{conditions:"Cloudy",high:"22",low:"14",code:"26",temp_unit:"℃"}

conditions：current weather

high：high temperature

low：low temperature

temp_unit: temperature unit

code：weather code

Wether code description:

Code	Description
0	tornado
1	tropical storm
2	hurricane
3	severe thunderstorms
4	thunderstorms
5	mixed rain and snow
6	mixed rain and sleet
7	mixed snow and sleet
8	freezing drizzle

9	drizzle
10	freezing rain
11	showers
12	showers
13	snow flurries
14	light snow showers
15	blowing snow
16	snow
17	hail
18	sleet
19	dust
20	foggy
21	haze
22	smoky
23	blustery
24	windy
25	cold
26	cloudy
27	mostly cloudy (night)
28	mostly cloudy (day)
29	partly cloudy (night)
30	partly cloudy (day)
31	clear (night)
32	sunny
33	fair (night)
34	fair (day)

35	mixed rain and hail
36	hot
37	isolated thunderstorms
38	scattered thunderstorms
39	scattered thunderstorms
40	scattered showers
41	heavy snow
42	scattered snow showers
43	heavy snow
44	partly cloudy
45	thundershowers
46	snow showers
47	isolated thundershowers
3200	not available