# 智器智能手表应用开发指南

## 版权声明

版权所有 © 2013，合肥华恒电子科技有限责任公司，保留所有权利。

# 1. 概述

该 SDK 旨在使第三方应用便利的集成和使用手表和手机端的通信服务，以及智能手表表盘开发的介绍与代码示例

## 1.1. 目的

本文档对手表和手机端的通信服务以及表盘的研发进行介绍。

## 1.2. 范围

本文档定义通信服务的使用说明、体系结构、API 接口。

# 2. 使用说明

## 2.1. 开发说明

在开发应用程序时，仅需关注文档中所提供的接口函数而不用了解具体实现。

## 2.2. 支持的平台

支持手机为 android 系统平台，手表为智器智器系列手表。

## 2.3. 开发环境以及配置

开发环境：1.推荐 ubuntu 9.10 及以上或 windows 平台。

2.java 1.6 及以上。

3.android sdk 开发环境，推荐版本 4.1。

开发工具：推荐 eclipse3.5 及以上，ADT20 以上，或 google 集成开发环境，下载地址：http://developer.android.com/sdk/index.html#linux-bundle

手机端：android2.3 及以上系统。预装（智器）ZWatch 手机助手，下载地址：

http://www.smartdevices.com.cn/support/download/Z1/

手表端：智器 ZWatch 智能手表。

4.具体使用介绍

手表端：同步程序使用方法：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

手机端：安装智器 Zwatch 手机助手，在一同提供的包内有该程序，也可上智器的官网进行下载安装，下载地址：http://www.smartdevices.com.cn/support/download/Z1/

# 3. 交互方式

通信服务模块为全局的后台常驻 Service 服务，建立蓝牙连接进行通信。在应用程序需要时，主动通过 bind 连接上通信 service 服务即可，不需要启动该 Service，在退出应用时应主动与通信 Service 解邦，减小开销。

## 3.1 启动连接发送数据

手机与手表蓝牙打开，配对完成后，启动手机端 Zwatch 手机助手（必须）后运行应用程序发送数据。注：建议发送应用有效数据前，先发送连接测试数据进行通信握手，验证连接已连接上以后再进行有效数据传输。

函数原型

```
boolean send(SyncData data);
```

参数说明

| 参数名 | 参数解释 |
| --- | --- |
| SyncData | 数据通信的实体类，键值对，类似bundle |

| | |
|---|---|

### 返回值

boolean： 是否发送成功

### 说明

如开始所说，发送正式数据之前，建议发送一些测试数据进行通信握手确保连接状态，例如：

```
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();
    }
```

## 3.2 数据完成回调

### 函数原型

```
void onRetrive(SyncData data)
```

### 返回值

无

### 说明

数据完成通信后，回调函数，返回通信数据。手表端和手机端共用该函数，在数据处理前，应先判断当前是在哪一端。

## 3.3 连接状态变化

### 函数原型

**void** onConnectionStateChanged(**boolean** connect)

### 参数说明

| 参数名 | 参数解释 |
| --- | --- |
| connect | 当前连接状态，`true`:连接上，`false`：断开 |
| | |

### 返回值

无

### 说明

当连接断开后程序需做处理，进行重新连接或者提示用户

## 3.4 模式变化

### 函数原型

**void** onModeChanged(**int** mode)

### 参数说明

| 参数名 | 参数解释 |
| --- | --- |
| mode | mode:SAVING_POWER_MODE = 0;RIGHT_NOW_MODE = 1;代表省电状态与正常状态 |

### 返回值

无

## 3.5 当前连接状态

### 函数原型

**boolean** isConnected()

返回值

**boolean：true，连接。false：断开**

说明

主动判断当前连接状态

## 3.6 发送文件

函数原型

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

参数说明

| 名称 | 可否为空 | 值 | 描述 |
| --- | --- | --- | --- |
| file | NO | File | 文件对象 |
| name | NO | String | 文件名称 |
| length | NO | **int** | 文件大小 |

## 3.7 文件发送完成回调

函数原型

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

参数说明

| 参数名 | 参数解释 |
| --- | --- |
| fileName | 文件名称 |
| success | 是否发送完成 |

返回值

无

### 说明

发送文件完成回调函数

## 3.8 文件接收完成回调函数

### 函数原型

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

### 参数说明

| 参数名 | 参数解释 |
| --- | --- |
| fileName | 文件名称 |
| success | 是否接收完成 |

### 返回值

无

### 说明

接收文件完成回调函数

## 3.9 通道建立完成

### 函数原型

```
void onChannelCreateComplete(ParcelUuid uuid, boolean success,
        boolean local)
```

### 参数说明

| 名称 | 可否为空 | 值 | 描述 |
| --- | --- | --- | --- |
| uuid | NO | ParcelUuid | 蓝牙通信通道的识别号 |
| success | NO | **boolean** | 是否成功 |

| | | | |
|---|---|---|---|
| local | NO | **boolean** | 是否本机 |

返回值

无

说明

通道建立完成回调函数

## 3.10 清理蓝牙配对信息

函数原型

**void** onClear(String address)

参数说明

| 名称 | 可否为空 | 值 | 描述 |
|---|---|---|---|
| address | NO | String | 设备地址 |
| | | | |
| | | | |

返回值

无

说明

通道建立完成回调函数

## 3.11 判断当前平台

函数原型

**boolean** isWatch()

返回值

boolean: true, 手表，false：手机。

说明

发送文件完成回调函数

## 3.12 建立自定义蓝牙通道

函数原型

**void** createChannel(ParcelUuid uuid)

参数

| 名称 | 可否为空 | 值 | 描述 |
| --- | --- | --- | --- |
| uuid | NO | ParcelUuid | 蓝牙通信通道的识别号 |
| | | | |
| | | | |

返回值

说明

建立用户自己的蓝牙通道进行通信，需要在程序退出时销毁该通道。

## 3.13 销毁自定义蓝牙通道

函数原型

void destroyChannel(ParcelUuid uuid)

参数

同 3.11

返回值

无

说明

蓝牙通道销毁

## 3.14 通过制定蓝牙通道发送数据

函数原型

**void** sendOnChannel(SyncData data, ParcelUuid uuid)

参数

| 名称 | 可否为空 | 值 | 描述 |
|---|---|---|---|
| data | NO | SyncData | 通信数据 |
| uuid | | ParcelUuid | 蓝牙通信通道的识别号 |
| | | | |

返回值

无

说明

通过自定义蓝牙通道发送数据

# 4. 通信数据类说明

在手机与手表通信服务支持一种定义数据，为 **SyncData**，该数据支持各种类型的键值对数据存储，作为通信数据的主体，类似与 bundle 数据。下面列出该数据类支持的数据接口。

**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()
```

# 5. 表盘开发指南

## 5.1 说明

表盘是为智器手表独特定制的钟表样式插件，利用统一命名规则进行管理，可在手表中随意切换。

## 5.2 开发环境

表盘开发环境同应用开发环境一致，表盘为 **android** 标准 **widget** 插件，可依附与应用中，也可单独成一个 **apk**。

## 5.3 命名规则

表盘命名必须以 WatchFace 为类名前缀，例如 WatchFaceClockProvider，以达到直接被 **laucher** 管理添加到桌面的效果，不以此前缀命名则需要在手表上点击插件添加按钮来添加至桌面。

## 5.4 示例

表盘插件为标准 **android widget** 插件，下面以 **AnalogClock** 插件为例，展示一个时钟表盘。安装表盘 **apk** 至手表上后，重启手表则可在表盘序列中找到该表盘，后期调试则可直接运行表盘无需

重启。

**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文件：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>
```

AndroidManifest中添加表盘

```
<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>
```

## 5.5 天气信息

表盘中可添加天气信息模块，具体信息由手表上天气软件通过广播更新，注册接收该广播则可收到天气的信息。天气广播的**action**为：String *ACTION_WEATHER_CHANGE* = "cn.indroid.action.weather.freshwidget";

广播中附带以json为数据格式的天气信息，具体格式为

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

conditions：当前天气状态

码　描述

0　龙卷风

1　热带风暴

2　飓风

3　强雷暴

4　雷暴

5　雨雪混合

6　混合雨和雨夹雪

7　混合雪和雨夹雪

8　冷冻小雨

9　细雨

10　冻雨

11　淋浴

12　淋浴

13　雪飘雪

14　光阵雪

15　吹雪

16　雪

17　冰雹

18　雨雪

19　灰尘

20 模糊

21 阴霾

22 烟

23 坏天气的

24 有风

25 冷

26 多云

27 晴间多云（夜）

28 （日）晴间多云

29 晴间多云（夜）

30 （日）晴间多云

31 清除（夜）

32 晴朗

33 公平（夜）

34 公平（日）

35 混合雨和冰雹

36 热

37 局部地区性雷暴

38 分散的雷暴

39 分散的雷暴

40 零星阵雨

41 大雪

42　零星阵雪

43　大雪

44　晴间多云

45　雷阵雨

46　阵雪

47　孤立的雷阵雨

3200　　不可用

high：最高温度

low：最低温度

**temp_unit**：温度单位

code：天气代码

具体代码代表天气为

| 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 |

| **码** | **描述** |
|---|---|
| 0 | 龙卷风 |
| 1 | 热带风暴 |
| 2 | 飓风 |
| 3 | 强雷暴 |
| 4 | 雷暴 |
| 5 | 雨雪混合 |

6 混合雨和雨夹雪

7 混合雪和雨夹雪

8 冷冻小雨

9 细雨

10 冻雨

11 淋浴

12 淋浴

13 雪飘雪

14 光阵雪

15 吹雪

16 雪

17 冰雹

18 雨雪

19 灰尘

20 模糊

21 阴霾

22 烟

23 坏天气的

24 有风

25 冷

26 多云

27 晴间多云（夜）

28 （日）晴间多云

29 晴间多云（夜）

30 （日）晴间多云

31 清除（夜）

32 晴朗

33 公平（夜）

34 公平（日）

35 混合雨和冰雹

36 热

37 局部地区性雷暴

38 分散的雷暴

39 分散的雷暴

40 零星阵雨

41 大雪

42 零星阵雪

43 大雪

44 晴间多云

45 雷阵雨

46 阵雪

47 孤立的雷阵雨

3200 不可用