

## WM\_W600\_OneShotConfig2.0(Android)SDK 用户手册

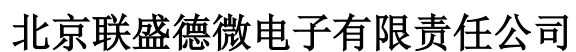
**V1.0**

北京联盛德微电子有限责任公司 (Winner Micro)

地址：北京市海淀区阜成路 67 号银都大厦 18 层

电话：+86-10-62161900

公司网址：[www.winnermicro.com](http://www.winnermicro.com)

[illegible]

目录

1	适用范围.....	4
2	接口定义.....	4
2.1	ISmartConfig 接口.....	4
2.1.1	startConfig 方法.....	4
2.1.2	stopConfig 方法.....	4
2.1.3	sendData 方法.....	5
2.2	IOneShotConfig 接口.....	5
2.2.1	start 方法.....	5
2.2.2	stop 方法.....	6
2.3	OneShotException 异常类.....	6
2.3.1	getErrorID 方法.....	6
2.4	ConfigType 枚举.....	7
2.5	SmartConfigFactory 类.....	7
2.5.1	createSmartConfig 方法.....	7
3	ISmartConfig 接口使用方法.....	7
3.1	获取 ISmartConfig 接口实例.....	7
3.2	启动后台线程，重复调用 startConfig 方法.....	8
4	IOneShotConfig 接口使用方法.....	9
4.1	获取 IOneShotConfig 接口实例.....	9
4.2	启动后台线程，调用 start 方法.....	9

## 1 适用范围

本文档设计描述了 W600 芯片一键配置（Android）SDK 的接口定义、使用方法等，为 Android 设备端开发 APP 配置 Wi-Fi 设备入网提供参考。

## 2 接口定义

### 2.1 ISmartConfig 接口

本接口定义了开始一键配置和结束一键配置的方法，并且定义了支持发送用户数据的方法。

#### 2.1.1 startConfig 方法

本方法开始一键配置，将用户设备连接的 Wi-Fi 名称和密码进行编码加密后，通过 UDP 组播报文发送出来。

原型：

`boolean startConfig(String password) throws OneShotException;`

参数：

**password**：用户设备所连接的 Wi-Fi 网络的密码；

返回值：

布尔类型返回值，true 代表成功；false 代表失败

异常：

具体参照 [OneShotException](#) 异常类

注意：

本方法调用了 UDP socket 发包接口，因此本方法需要在后台线程中调用；本方法首先发送信道同步包再发送将 Wi-Fi 名称和密码编码的数据包，通常在 5~10 秒后返回，为保证 Wi-Fi 设备端收全信息并正确解析，本方法需要被循环调用，直到配网成功。如果在此方法抛出 OneShotException 异常，表明配网失败，请通过调用 stopConfig 方法结束配网并释放资源。

#### 2.1.2 stopConfig 方法

本方法停止一键配置，释放配置过程中的资源。

原型:

`void stopConfig();`

参数: 无

返回值: 无

异常: 无

注意:

本方法停止一键配置并释放资源, 为保证资源释放, 无论配网是否成功, 在结束时到必须调用此方法。

## 2.1.3 sendData 方法

本方法发送用户数据, 将用户自定义数据通过编码加密后, 通过 UDP 组播报文发送出来。

原型:

`boolean sendData(String data) throws OneShotException;`

参数:

**data:** 用户自定义数据

返回值:

布尔类型返回值, true 代表成功; false 代码失败

异常:

具体参照 [OneShotException 异常类](#)

注意:

本方法调用了 UDP socket 发包接口, 因此本方法需要在后台线程中调用; 本方法首先发送信道同步包再发送将 Wi-Fi 名称和密码编码的数据包, 通常在 5~10 秒后返回, 为保证 Wi-Fi 设备端收全信息并正确解析, 本方法需要被循环调用。如果在此方法抛出 OneShotException 异常, 表明发送数据失败, 请通过调用 stopConfig 方法结束发送数据并释放资源。

## 2.2 IOneShotConfig 接口

本接口定义了另外一套开始一键配置和结束一键配置的方法, 实现和 ISmartConfig 接口相同的功能, 同时增加了配网超时功能。

### 2.2.1 start 方法

本方法开始一键配置, 将用户设备连接的 Wi-Fi 名称和密码进行编码加密后, 通过 UDP 组播报文发送出来。

原型:

`void start(String ssid, String key, int timeout, Context context);`

参数:

`ssid`: Wi-Fi 网络名称;

`password`: 用户设备所连接的 Wi-Fi 网络的密码;

`timeout`: 超时时间, 单位秒;

`context`: 是 Activity 的 context 对象

返回值: 无

异常: 无

注意:

本方法调用了 UDP socket 发包接口, 因此本方法需要在后台线程中调用。本方法会在设定的 `timeout` 超时时间后自动返回, 如果没有到超时时间即返回, 表明配网过程中有异常情况终止了配网。如果想停止配网, 可以通过调用 `stop` 方法终止本方法, 使其直接返回。

## 2.2.2 stop 方法

本方法停止一键配置, 释放配置过程中的资源。

原型:

`void stop();`

参数: 无

返回值: 无

异常: 无

注意:

本方法停止一键配置并释放资源, 为保证资源释放, 无论配网是否成功, 在结束时到必须调用此方法。

## 2.3 OneShotException 异常类

本类定义了在一键配置过程中, `startConfig` 方法和 `sendData` 方法可能发生的异常类型。

### 2.3.1 getErrorID 方法

本方法返回具体异常的类型, 用户可根据异常类型提示用户具体配网失败原因。本异常类定义的异常类型有如下几种:

`ERROR_WIFI_DISABLED = 101`;表示设备没有连接 Wi-Fi 网络

`ERROR_NETWORK_DISCONNECTED = 102`;表示网络断开

`ERROR_NETWORK_NOT_SUPPORT = 103`;表示不支持该网络, 例如 5G 网络

`ERROR_TIMEOUT = 104`;表示配网超时错误

原型:

```
public int getErrorID();
```

参数：无

返回值：

Integer 类型返回值，表示具体异常类型。

异常：无

## 2.4 ConfigType 枚举

本枚举定义了具体一键配置的配网方式，目前只有 UDP 组播方式一种。

## 2.5 SmartConfigFactory 类

本类是一个工厂类，提供获取 ISmartConfig 接口实例的接口方法。

### 2.5.1 createSmartConfig 方法

本方法获取 ISmartConfig 接口实例。

原型：

```
public ISmartConfig createSmartConfig(ConfigType configType, Context context);
```

参数：

configType：是一个 ConfigType 枚举类型，目前仅支持 UDP

context：是 Activity 的 context 对象

返回值：

ISmartConfig 类型返回值，返回 ISmartConfig 接口实例。

异常：无

## 3 ISmartConfig 接口使用方法

### 3.1 获取 ISmartConfig 接口实例

首先通过下面代码获取 ISmartConfig 接口实例，这里的 factory 是 SmartConfigFactory 的实例，this 是你的 activity。

```
ISmartConfig smartConfig = factory.createSmartConfig(ConfigType.UDP, this);
```

## 3.2启动后台线程，重复调用 startConfig 方法

然后可以参考 OneShotConfig Demo App 创建后台线程，在线程中反复调用 startConfig 接口，如下代码可以参考。

```
class UDPReqThread implements Runnable {
    public void run() {
        WifiManager wifiManager = null;
        try {
            wifiManager = (WifiManager) getSystemService(WIFI_SERVICE);
            if(wifiManager.isWifiEnabled() || isWifiApEnabled())
            {
                while(isStart){
                    if(smartConfig.startConfig(psw) == false)
                    {
                        break;
                    }
                }
            }
        }
        catch (OneShotException oe) {
            oe.printStackTrace();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        finally{
            smartConfig.stopConfig();
            runOnUiThread(confPost);
        }
    }
}
```



## 4 IOneShotConfig 接口使用方法

### 4.1 获取 IOneShotConfig 接口实例

首先通过下面代码获取 IOneShotConfig 实例，这里的 factory 是 SmartConfigFactory 的实例。

```
IOneShotConfig oneshotConfig = factory.createOneShotConfig(ConfigType.UDP);
```

### 4.2 启动后台线程，调用 start 方法

然后可以参考 OneShotConfig Demo App 创建后台线程，在线程中调用 start 接口，如下代码可以参考。

```
class UDPReqThread implements Runnable {
    public void run() {
        WifiManager wifiManager = null;
        try {
            wifiManager = (WifiManager) getSystemService(WIFI_SERVICE);
            if(wifiManager.isWifiEnabled() || isWifiApEnabled())
            {
                int timeout = 60;//miao
                oneshotConfig.start(ssid, psw, timeout,
OneShotActivity.this);
            }
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        finally{
            oneshotConfig.stop();
            runOnUiThread(confPost);
        }
    }
}
```