机智云SDK集成文档

1概述

1.1 SDK功能概述

机智云sdk主要帮助开发者通过sdk接口调用的方式维护用户系统，用户与设备的绑定关系，设备的配置上线以及设备状态的获取和控制指令的发送。

1.2 SDK基本操作流程

GOKIT:

登录账号 —> 配置设备上网 —> 搜索设备 —> 绑定 —> 获取设备绑定列表 —> 控制设备

虚拟设备:

登录账号 —> 绑定虚拟设备 —> 获取设备绑定列表 —> 控制设备

账号操作基本流程

普通实名账号:

注册 —> 登录 —> 注销
登录 —> 修改密码

手机账号

注册 —> 登录 —> 注销
登录 —> 修改密码
注册 —> 忘记密码

第三方方账号

第三方SDK —> 获取 uid、token —> 登录 —> 注销 —>第三方SDK注销

配置设备上网流程

SOFT-AP 模式

设备进入 Soft-AP 模式 —> 手机连上设备的热点 —> 发送配置请求 —> 配置结果

AIR-LINK 模式

手机连上 Wi-Fi —> 设备进入 Air-Link 模式 —> 发送配置请求 —>配置结果

绑定实体设备基本流程

设备上大循环 —> 手机与设备保持局域网 —> 发现设备 —> 设备按下获取 Passcode 按钮（此流程可选） —> 绑定结果

绑定虚拟设备基本流程

登录 site.gizwits.com 启动一台虚拟设备 —> 显示该设备的二维码 —> 手机扫描该二维码 —> 绑定结果

1.3 基础对象

概述

机制云sdk共有2个基本对象。下面分别介绍它们:

XPGWifiSDK

简介

机智云sdk控制类。机智云sdk中提供了用户注册登录，设备配置，设备连接绑定等操作的接口，该类是对这些接口的抽象。该类是一个单例。它提供了如下接口:
1.sdk初始化的接口。包括appid的指定，日志的设定。
2.用户账号相关的接口。包括用户的注册，登录，密码重置等。
3.设备配置相关的接口。包括设备的配置入网，绑定，获取设备列表等。
4.设备连接相关的接口。包括设备的登录等。

API

初始化SDK

功能描述	初始化sdk		
方法	startWithAppID		
请求参数	参数名称	类型	描述
	context	**Context**	上下文环境
	appID	String	appID
示例	`XPGWifiSDK.sharedInstance().startWithAppID(getApplicationContext(),"ecb16888bb794c68b15606f8247f3e31");`		

初始化日志

功能描述	初始化日志		
方法	setLogLevel		
请求参数	参数名称	类型	描述
	logLevel	**XPGWifiLogLevel**	日志级别
	logFile	String	日志文件的相对路径
	bPrintDataInDebug	**boolean**	是否打印数据
示例	`XPGWifiSDK.sharedInstance().setLogLevel(XPGWifiLogLevel.XPGWifiLogLevelAll,"BassApp.log", true);`		

设置SDK通用监听器

功能描述	指定sdk监听器		
方法	setListener()		
请求参数	参数名称	类型	描述
	sdkListener	**XPGWifiSDKListener**	sdk通用监听器
示例	`XPGWifiSDK.sharedInstance().setListener(new XPGWifiSDKListener());`		

用户注册

功能描述	通过自定义账号注册		
方法	registerUser()		
请求参数	参数名称	类型	描述
	userName	**String**	账户名
	password	**String**	账户密码
回调接口	`XPGWifiSDKListener.didRegisterUser();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	`errorMessage`	String	结果信息(出现异常)
	uid	String	用户ID
	token	String	授权令牌
示例	`XPGWifiSDK.sharedInstance().registerUser("gizwits", "12345678");`		

请求手机验证码

功能描述	请求发送手机验证码短信到指定手机号		
方法	requestSendVerifyCode()		
请求参数	参数名称	类型	描述
	phone	**String**	电话号码
回调接口	`XPGWifiSDKListener.didRequestSendVerifyCode();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	`errorMessage`	String	结果信息(出现异常)
示例	`XPGWifiSDK.sharedInstance().requestSendVerifyCode("13800000000");`		

手机号注册

功能描述	通过手机号和验证码注册用户		
方法	registerUserByPhoneAndCode()		
请求参数	参数名称	类型	描述
	phone	**String**	电话号码
	password	**String**	账户密码
	code	**String**	手机验证码
回调接口	`XPGWifiSDKListener.didRegisterUser();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	`errorMessage`	String	结果信息(出现异常)
	uid	String	用户ID
	token	String	授权令牌
示例	`XPGWifiSDK.sharedInstance().registerUserByPhoneAndCode("13800000000", "password", "123456");`		

电子邮箱注册

功能描述	通过电子邮箱地址注册用户		
方法	registerUserByEmail()		
请求参数	参数名称	类型	描述
	email	**String**	电子邮箱地址
	password	**String**	账户密码
回调接口	`XPGWifiSDKListener.didRegisterUser();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	errorMessage	String	结果信息(出现异常)
	uid	String	用户ID
	token	String	授权令牌
示例	`XPGWifiSDK.sharedInstance().registerUserByPhoneAndCode(“example@gizwits.com”, "password");`		

实名用户登录

功能描述	通过账户\手机号\邮箱注册的用户通过该接口进行登录		
方法	userLoginWithUserName()		
请求参数	参数名称	类型	描述
	userName	**String**	账号名(账号\电话\邮箱)
	password	**String**	账户密码
回调接口	`XPGWifiSDKListener.didUserLogin();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	errorMessage	String	结果信息(出现异常)
	uid	String	用户ID
	token	String	授权令牌
示例	`XPGWifiSDK.sharedInstance().userLoginWithUserName(“example@gizwits.com”, "password");`		

匿名用户登录

功能描述	不注册直接进行匿名登录		
方法	userLoginAnonymous()		
请求参数	无		
回调接口	`XPGWifiSDKListener.didUserLogin();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	`errorMessage`	String	结果信息(出现异常)
	uid	String	用户ID
	token	String	授权令牌
示例	`XPGWifiSDK.sharedInstance().userLoginAnonymous();`		

第三方用户登录

功能描述	使用新浪、百度、腾讯的账号进行登录		
方法	userLoginWithThirdAccountType()		
请求参数	参数名称	类型	描述
	thirdAccountType	XPGWifiThirdAccountType	第三方账号类型枚举
	uid	String	用户ID
	token	String	授权令牌
回调接口	`XPGWifiSDKListener.didUserLogin();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	`errorMessage`	String	结果信息(出现异常)
	uid	String	用户ID
	token	String	授权令牌
示例	`XPGWifiSDK.sharedInstance().userLoginWithThirdAccountType(XPGWifiThirdAccountType.XPGWifiThirdAccountTypeSINA, "12341234", "ACCESS_TOKEN");`		

账户注销

功能描述	账户注销		
方法	userLogout()		
请求参数	参数名称	类型	描述
	uid	**String**	账号ID
回调接口	`XPGWifiSDKListener.didUserLogout();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	`errorMessage`	String	结果信息(出现异常)
示例	`XPGWifiSDK.sharedInstance().userLogout("8e1253f8e430407e0bfaa01de7d60000");`		

手机号重置密码

功能描述	通过手机号和验证码重置密码		
方法	changeUserPasswordByCode()		
请求参数	参数名称	类型	描述
	phone	**String**	已注册手机号
	code	**String**	验证码
	newPassword	**String**	新密码
回调接口	`XPGWifiSDKListener.didChangeUserPassword();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	`errorMessage`	String	结果信息(出现异常)
示例	`XPGWifiSDK.sharedInstance().changeUserPasswordByCode("13800000000","123456","hello");`		

电子邮箱重置密码

功能描述	通过发送验证邮件重置密码		
方法	changeUserPasswordByEmail()		
请求参数	参数名称	类型	描述
	email	**String**	已注册邮箱
回调接口	`XPGWifiSDKListener.didChangeUserPassword();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	`errorMessage`	String	结果信息(出现异常)
示例	`XPGWifiSDK.sharedInstance().changeUserPasswordByCode("example@gizwits.com");`		

修改密码

功能描述	通过token和旧密码修改新密码		
方法	changeUserPassword()		
请求参数	参数名称	类型	描述
	token	**String**	授权令牌
	oldPassword	**String**	旧密码
	newPassword	**String**	新密码
回调接口	`XPGWifiSDKListener.didChangeUserPassword();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	`errorMessage`	String	结果信息(出现异常)
示例	`XPGWifiSDK.sharedInstance().changeUserPassword("e5a37a73e8ad4fcc8753a5099d125e08","oldpsw","newpsw");`		

配置设备上线

功能描述	通过airlink或softap模式配置设备上线		
方法	setDeviceWifi()		
请求参数	参数名称	类型	描述
	ssid	**String**	Wi-Fi名称
	key	**String**	Wi-Fi密码
	mode	**XPGWifiConfigureMode**	配置模式
	timeout	**int**	配置模式超时时间
回调接口	`XPGWifiSDKListener.didSetDeviceWifi();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	`device`	XPGWifiDevice	设备对象

示例	airlink配置模式：
	`XPGWifiSDK.sharedInstance().setDeviceWifi("wifi_home", "123321", XPGWifiConfigureMode.XPGWifiConfigureModeAirLink, 60);`
	
	softap配置模式：
	`XPGWifiSDK.sharedInstance().setDeviceWifi("wifi_home", "123321", XPGWifiConfigureMode.XPGWifiConfigureModeSoftAP, 60);`		

获取WI-FI列表

功能描述	获取手机搜索到的SSID列表		
方法	getSSIDList()		
请求参数	无		
回调接口	`XPGWifiSDKListener.didGetSSIDList();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	`ssidInfoList`	**List<XPGWifiSSID>**	ssid列表
示例	`XPGWifiSDK.sharedInstance().getSSIDList();`		

获取设备列表

功能描述	搜索本地局域网内设备和账号下绑定设备		
方法	getBoundDevices()		
请求参数	参数名称	类型	描述
	uid	**String**	用户ID
	token	**String**	授权令牌
	specialProductKey	**String...**	指定的productkey
回调接口	`XPGWifiSDKListener.didDiscovered();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	`devicesList`	**List<XPGWifiDevice>**	设备列表
示例	`XPGWifiSDK.sharedInstance().getBoundDevices("8e1253f8e430000ea5e5a01de7d60ed9", "e5a37a73e8ad40000753a5099d125e08", "e3cf7332b7834a03a92d9e14a3f6d352");`		

绑定设备

功能描述	把用户账号与设备进行绑定		
方法	bindDevice()		
请求参数	参数名称	类型	描述
	uid	**String**	用户ID
	token	**String**	授权令牌
	did	**String**	设备ID
	passcode	**String**	设备授权码
	remark	**String**	设备备注
回调接口	`XPGWifiSDKListener.didDiscovered();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	`errorMessage`	**String**	设备列表
示例	`XPGWifiSDK.sharedInstance().bindDevice("8e1253f8e430400005e5a01de7d60ed9", "e5a37a73e8ad40000753a5099d125e08", "nKnSj25QiSt55x7ExxxxYX", "JDQQFBHHHH", "light_in_room");`		

解除绑定设备

功能描述	把用户账号与设备进行解除绑定		
方法	unbindDevice()		
请求参数	参数名称	类型	描述
	uid	**String**	用户ID
	token	**String**	授权令牌
	did	**String**	设备ID
	passcode	**String**	设备授权码
回调接口	`XPGWifiSDKListener.didDiscovered();`		
回调接口参数	参数名称	类型	描述
	error	int	结果码
	`errorMessage`	**String**	设备列表
	did	**String**	设备ID
示例	`XPGWifiSDK.sharedInstance().unbindDevice("8e1253f8e430400005e5a01de7d60ed9", "e5a37a73e8ad40000753a5099d125e08", "nKnSj25QiSt55x7ExxxxYX", "JDQQFBHHHH");`		

XPGWifiDevice

简介

机智云sdk设备类。机智云的设备包含了许多属性，该类是对这些属性的抽象。它提供了如下接口：
1.设备操作。包括设备的登录，控制，断开等。
2.设备基础信息获取。包括设备的did，passcode,硬件信息等。
3.设备实时状态获取。例如热水器的水温等，因不同设备而异。

API

设置设备监听器

功能描述	指定sdk监听器		
方法	setListener()		
请求参数	参数名称	类型	描述
	deviceListener	**XPGWifiDeviceListener**	sdk设备监听器
示例	`XPGWifiDevice.setListener(new XPGWifiSDKListener());`		

设备登录

功能描述	登录设备，登录后可进行控制		
方法	login()		
请求参数	参数名称	类型	描述
	uid	**String**	用户ID
	token	**String**	授权令牌
回调接口	`XPGWifiDeviceListener.didLogin();`		
回调接口参数	参数名称	类型	描述
	device	XPGWifiDevice	设备对象
	`result`	**int**	登录结果
示例	`XPGWifiDevice.login("8e1253f8e430400005e5a01de7d60ed9", "e5a37a73e8ad40000753a5099d125e08");`		

发送控制指令

功能描述	发送控制指令		
方法	write()		
请求参数	参数名称	类型	描述
	jsonData	**String**	指令json
回调接口	`XPGWifiDeviceListener.didReceiveData();` 注意：该回调需要mcu能接收到指令后主动上报状态才会回调。否则需要app发送查询指令才能回调。		
回调接口参数	参数名称	类型	描述
	device	XPGWifiDevice	设备对象
	`dataMap`	**ConcurrentHashMap< String, Object>**	状态数据表
	`result`	**int**	登录结果

示例	`JSONObject json = new JSONObject();`
	`try`
	`{`
	`  json.put("cmd", 2);`
	`} catch (JSONException e) {`
	`  e.printStackTrace();`
	`}`
	`XPGWifiDevice.write(json.toString())`		

断开设备

功能描述	断开已连接的设备		
方法	disconnect()		
请求参数	无		
回调接口	`XPGWifiDeviceListener.didDisconnected();`		
回调接口参数	参数名称	类型	描述
	device	XPGWifiDevice	设备对象
示例	`XPGWifiDevice.disconnect();`		

获取设备硬件参数

功能描述	获取设备模块协议版本号，固件版本号等信息		
方法	getHardwareInfo()		
请求参数	无		
回调接口	`XPGWifiDeviceListener.didQueryHardwareInfo();`		
回调接口参数	参数名称	类型	描述
	device	XPGWifiDevice	设备对象
	result	int	登录结果
	hardwareInfo	**ConcurrentHashMap< String, String>**	硬件信息数据表
示例	`XPGWifiDevice.getHardwareInfo();`		

获取设备DID

功能描述	获取设备id	
方法	getDid()	
请求参数	无	
返回参数	参数类型	描述
	String	设备ID
示例	`XPGWifiDevice.getDid();`	

获取设备产品识别码

功能描述	获取设备产品识别码	
方法	getProductKey()	
请求参数	无	
返回参数	参数类型	描述
	String	设备产品识别码
示例	`XPGWifiDevice.getProductKey();`	

获取设备产品名称

功能描述	获取设备名称	
方法	getProductName()	
请求参数	无	
返回参数	参数类型	描述
	String	设备产品名称
示例	`XPGWifiDevice.getProductName();`	

获取设备密码

功能描述	获取设备密码	
方法	getPasscode()	
请求参数	无	
返回参数	参数类型	描述
	String	设备密码
示例	`XPGWifiDevice.getPasscode();`	

获取设备WI-FI模块物理地址

功能描述	获取设备Wi-Fi模块物理地址	
方法	getMacAddress()	
请求参数	无	
返回参数	参数类型	描述
	String	物理地址
示例	`XPGWifiDevice.getMacAddress();`	

获取设备IP

功能描述	获取设备模块在路由器中的IP地址	
方法	getIPAddress()	
请求参数	无	
返回参数	参数类型	描述
	String	IP地址
示例	`XPGWifiDevice.getIPAddress();`	

获取设备别名

功能描述	获取设备别名	
方法	getRemark()	
请求参数	无	
返回参数	参数类型	描述
	String	设备别名
示例	`XPGWifiDevice.getRemark();`	

判断是否已绑定

功能描述	判断设备是否已经跟账号绑定，绑定后可进行大循环控制		
方法	isBind()		
请求参数	参数名称	类型	描述
	uid	String	用户ID
返回参数	参数类型	描述	
	boolean	是否已绑定	
示例	`XPGWifiDevice.isBind("8e1253f8e430407e0bfaa01de7d60000");`		

判断是否局域网内设备

功能描述	判断设备是否与手机同一局域网内	
方法	isLAN()	
请求参数	无	
返回参数	参数类型	描述
	boolean	是否同一局域网
示例	`XPGWifiDevice.isLAN();`	

判断设备是否已注销

功能描述	判断设备已在云端注销	
方法	isDisabled()	
请求参数	无	
返回参数	参数类型	描述
	boolean	是否已在云端注销
示例	`XPGWifiDevice.isDisabled();`	

判断设备是否已连上云端

功能描述	判断设备已连上云端	
方法	isOnline()	
请求参数	无	
返回参数	参数类型	描述
	boolean	是否已连接云端
示例	`XPGWifiDevice.isOnline();`	

判断设备是否已连接

功能描述	判断设备已与app连接，已连接的设备可收发指令	
方法	isConnected()	
请求参数	无	
返回参数	参数类型	描述
	boolean	是否已连接app
示例	`XPGWifiDevice.isConnected();`	

XPGWifiSSID

简介

机智云SDK Wi-Fi类。该类包含了sdk搜索到的Wi-Fi信号的名称和信号强度。

API

获取信号源名称

功能描述	获取Wi-Fi信号源的名称（SSID）	
方法	getSsid()	
请求参数	无	
返回参数	参数类型	描述
	String	名称
示例	`XPGWifiSSID.getSsid();`	

获取信号源强度

功能描述	获取Wi-Fi信号源的信号强度	
方法	getRssi()	
请求参数	无	
返回参数	参数类型	描述
	short	信号强度（0-100）
示例	`XPGWifiSSID.getRssi();`	

1.3专有名词解释

机智云

机智云（Gizwits）是国内第一个专门为智能硬件提供后台支持的云服务平台。成立于2010年，已为过百家国内外智能硬件开发商累计超过200万台设备提供云服务。2014.9月发布机智云2.0，为国内开发者以及企业团队提供智能硬件自助开发及设备云服务。

GAgent

全称Gizwits Agent，运行于Wi-Fi模块中，设备通过GAgent接入机智云服务器。目前已兼容国内主流的Wi-Fi模块，开发者也可以通过获取GAgent二次开发包实现自定义的模块接入机智云

小循环

智能设备与手机、智能设备与智能设备之间，通过连接同一个路由器实现局域网内部的通信（查看状态或控制），我们称之为小循环。

大循环

智能设备通过路由器或直接接入互联网以实现用户的远程监测与控制，我们称为大循环。

ProductKey

产品标识码，开发者通过机智云后台创建新产品后，自动生成的一个32位字符串。在机智云的数据库中是一个唯一的号码，开发者完成开发写入设备主控MCU后，机智云通过此标识码对设备进行识别并自动完成注册。

did

设备号，当一个设备初次接入机智云时，机智云自动根据ProductKey以及设备Wi-Fi模块MAC地址为此设备注册一个did，此did全网唯一，用于与用户的绑定及后续操作。

passcode

设备通行证，用于校验用户的绑定/控制权限。设备初次上线时，云端会分配一个passcode存入设备，但用户发起设备绑定时，只要是合法操作即可拿到此通行证，以成功绑定设备并对设备进行有效期内的查看、控制等操作。

AppID

应用标识码，当开发者需要为一款智能产品开发应用（包括iOS、Android、Web应用等）时，后台会自动生成一个AppID，并与此设备进行关联。应用开发时需要填入此AppID。

Onboarding

用户将一款基于Wi-Fi的物联网设备配置连接上路由器的过程称为Onboarding。新设备第一次使用时需要知道路由器的账号和密码，以通过路由器连接互联网。由于大多数的物联网设备没有自带的屏幕和键盘，所以需要通过智能手机向设备发送路由器的SSID和密码，这个过程机智云称为Onboarding。机智云提供的Wi-Fi设备接入SDK中已经内置了此配置的功能。

AirLink

机智云推出的实现Onboarding的一套技术名称，兼容了多个Wi-Fi模块厂商的Smart-Config协议以及一套良好用户体验的标准Onboarding操作流程，机智云的Wi-Fi设备接入SDK已经内置AirLink技术。

SoftAP

由于目前各个Wi-Fi模块厂商的Smart Config协议均未完全成熟，也不支持5G路由器信号。机智云在提供了AirLink配置模式的同时也支持SoftAP模式配置设备接入路由器。当设备进入SoftAP配置模式时，设备本身将成为一个AP，智能手机可直接与设备进行连接，然后在手机上的界面上输入路由器的SSID和密码，设备接收到信息的时候会自动尝试连接路由器，连接成功则自动切换到正常使用的模式。

2集成准备

2.1注册机智云账号

在使用机智服务前，你需要通过site.gizwits.com 注册一个开发者账号。请完整填写你的注册信息。

2.2新建设备接入

（参考快速开始文档）

2.3获得app ID和产品标识码（productkey）

相关信息：如何通过SDK和Product Key，获取相应的配置文件

新建设备》设备列表》点击设备名》设备详细信息

2.4下载SDK

新建设备—>设备列表—>左侧产品管理—>产品开发资源—>下载SDK

2.5导入SDK

新建android工程—>把下载后的sdk解压—>解压后的libs目录下的所有内容复制到工程的libs目录下

3基本功能集成

3.1配置AndroidManifest.xml

3.1.1添加权限

请将下面权限配置代码复制到 AndroidManifest.xml 文件中：

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
```

3.1.2权限说明

权限	用途
ACCESS_NETWORK_STATE	允许程序访问有关GSM网络信息
ACCESS_WIFI_STATE	允许程序访问Wi-Fi网络状态信息
READ_PHONE_STATE	允许程序访问手机状态信息
ACCESS_COARSE_LOCATION	允许一个程序访问CellID或WiFi热点来获取粗略的位置
ACCESS_FINE_LOCATION	允许一个程序访问精良位置(如GPS)
WRITE_EXTERNAL_STORAGE	允许程序写入外部sd卡
INTERNET	允许程序打开网络接口
CHANGE_WIFI_STATE	允许程序改变Wi-Fi连接状态

3.2初始化sdk

概述

使用sdk之前，需要先执行SDK的初始化，初始化分为两个步骤，首先需要指定sdk的APP ID，其次再设定日志的参数，如输出级别，保存路径等。

使用前须知

- 开发者需要先获取到相关的APP ID（参考名词解释APP ID一节）。例如文档中使用的是机智云实验室内的智能云空调所用的APP ID(ecb16888bb794c68b15606f8247f3e31)。
- 获取步骤请参考本文档2.3节

基本功能

3.2.1填写APP ID

描述

- 请务必在应用的Application或者第一个启动的Activity的onCreate中调用该方法，该方法指定sdk的APP ID,指定后使用注册登录等的所有用户关系都将保存在该APP ID中，替换别的APP ID以后，需要重新注册用户。

方法

```
XPGWifiSDK.sharedInstance().startWithAppID(Context context,String appid);
```

参数

参数	描述
context	上下文对象
appid	APP ID

代码范例

```
import com.xtremeprog.xpgconnect.XPGWifiSDK;
…
public void onCreate() {
           super.onCreate();

XPGWifiSDK.sharedInstance().startWithAppID(getApplicationContext(),
                       "ecb16888bb794c68b15606f8247f3e31");

     }
```

注意事项

- APP ID只能填写一个
- 该方法只需要调用一次

3.2.2日志设定

描述

该方法指定了应用在logcat中输出的日志级别，以及保存的日志文件路径。

方法

```
XPGWifiSDK.setLogLevel(XPGWifiLogLevel logLevel, String logFile,
boolean bPrintDataInDebug);
```

参数

参数	描述
logLevel	日志级别分为 XPGWifiLogLevel.XPGWifiLogLevelAll 所有级别 XPGWifiLogLevel.XPGWifiLogLevelWarning 警告级别 XPGWifiLogLevel.XPGWifiLogLevelError 错误级别
logFile	日志文件路径 相对于/sdcard/
bPrintDataInDebug	是否在后台输出二进制数据

代码范例

```
import com.xtremeprog.xpgconnect.XPGWifiSDK;
import com.xtremeprog.xpgconnect.XPGWifiSDK.XPGWifiLogLevel;
…
public void onCreate() {
          super.onCreate();
          XPGWifiSDK.sharedInstance().setLogLevel(
                    XPGWifiLogLevel.XPGWifiLogLevelAll,
"XpgAirCondition.log",
                    false);

     }
```

注意事项

- 应该在指定了APP ID以后就调用该方法
- 日志文件可以为.log或者.txt格式
- 该方法只需要调用一次

3.3注册监听器

描述

注册监听器是为了能收到模块及云端发回的响应指令。

使用前须知

监听器分两种：
一、XPGWifiSDKListener通用监听器，包含了注册、登录、配置设备、绑定设备等回调接口。
二、XPGWifiDeviceListener设备监听器，包含了单个设备的登录、控制、状态上报等接口。

3.3.1注册 XPGWifiSDKListener

描述

SDK通用监听器，该监听器是SDK使用中十分重要的一个监听器，与SDK相关的操作都会在这里会回调，如果没有正确注册通用监听器，将无法正常使用SDK。

方法

```
XPGWifiSDK.sharedInstance().setListener(new XPGWifiSDKListener());
```

回调介绍

- didBindDevice：绑定设备结果回调
- didChangeUserEmail：更换用户Email结果回调
- didChangeUserPhone：更换用户手机号结果回调
- didChangeUserPassword：更换用户密码结果回调
- didDiscovered：获取设备列表结果回调
- didGetSSIDList：获取模块周围Wi-Fi热点列表结果回调
- didRegisterUser：用户注册结果回调
- didRequestSendVerifyCode：发送手机验证码结果回调
- didSetDeviceWifi：配置模块结果回调
- didUnbindDevice：设备解除绑定结果回调
- didUserLogin：用户登录结果回调
- didUserLogout：用户注销结果回调

代码范例

该回调建议的设置方式有两种：

一、在每一个使用到的Activity中都实例化一次监听器并注册一次，且只实现需要的回调接口。该种方式比较灵活，可在service中使用。但要注意必须每次打开activity都监听一次，且无法多个Activity同时收到回调。如：

```
import com.xtremeprog.xpgconnect.XPGWifiSDK;
import com.xtremeprog.xpgconnect.XPGWifiSDKListener;
…
// 实例化监听器
XPGWifiSDKListener xpgWifiSDKListener = new XPGWifiSDKListener() {

          // 注册用户回调
          public void didRegisterUser(int result, String errorMessage,
String uid,String token) {
               if (result==XPGWifiErrorCode.XPGWifiError_NONE)
               {
                    // 注册成功，处理注册成功的逻辑
                    …
               } else {
                    // 注册失败，处理注册失败的逻辑
                    …
               }
          };
     };

          public void onCreate() {
               super.onCreate();
               …
               // 注册监听器
          XPGWifiSDK.sharedInstance().setListener(xpgWifiSDKListener);
               // 调用SDK注册接口
          XPGWifiSDK.sharedInstance().registerUser("HelloGizwits",
"12345678");
     }
```

二、在一个基类中实例化一次监听器，并把回调抛出，子类继承基类，这要就不需要每个子类都实例化一次监听器。该种方式通过继承的方式，可以多个Activity都收到回调。但该种方式无法在Service中使用。如无特别说明，文档中的范例都是使用该方法注册监听器。

1.创建基类，在基类中实例化和注册监听器

```
import com.xtremeprog.xpgconnect.XPGWifiSDK;
import com.xtremeprog.xpgconnect.XPGWifiSDKListener;

public class BaseActivity extends Activity {
	//注册监听器
	private XPGWifiSDKListener sdkListener = new XPGWifiSDKListener() {
		@Override
		public void didRegisterUser(int error, String errorMessage, String
	uid,String token)
				{
				//
				BaseActivity.this.didRegisterUser(error, errorMessage, uid,
		token);
				}
		};

	/**
	 * 注册用户结果回调接口.
	 *
	 * @param error
	 *            结果代码
	 * @param errorMessage
	 *            错误信息
	 * @param uid
	 *            the 用户id
	 * @param token
	 *            the 授权令牌
	 */
	protected void didRegisterUser(int error, String errorMessage, String uid,
				String token) {
		// TODO Auto-generated method stub
	}

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		…
		// 每次启动activity都要注册一次sdk监听器，保证sdk状态能正确回调
		XPGWifiSDK.sharedInstance().setListener(sdkListener);

	}

}
```

2.子类继承基类，实现基类的回调接口。

```
import com.gizwits.framework.activity.BaseActivity;
import com.xtremeprog.xpgconnect.XPGWifiSDK;

public class TestActivity extends BaseActivity {


      protected void onCreate(android.os.Bundle savedInstanceState) {
            //调用父类方法
            super.onCreate(savedInstanceState);
            //调用注册方法
            XPGWifiSDK.sharedInstance().registerUser("HelloGizwits", "12345678");
      };

      @Override
      protected void didRegisterUser(int result, String errorMessage, String
uid,String token) {
            if (result==XPGWifiErrorCode.XPGWifiError_NONE)
            {
                  // 注册成功，处理注册成功的逻辑
                  …
            } else {
                  // 注册失败，处理注册失败的逻辑
                  …
            }

      }
}
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 每次回到Activity都应该注册一次监听器，确保能正确回调

3.3.2设置XPGWifiDeviceListener

描述

SDK设备监听器，该监听器是SDK使用中十分重要的一个监听器，与设备相关的方法都会在该监听器中回调。

方法

```
XPGWifiDevice.setListener(new XPGWifiSDKListener());
```

回调介绍

- didQueryHardwareInfo：设备硬件信息回调
- didDeviceOnline：设备上下线状态回调
- didDisconnected：设备断开连接回调
- didLogin：设备登录回调
- didReceiveData：接收到设备状态上报回调

代码范例

1.先从设备列表等接口获取到设备列表

```
@Override
protected void didDiscovered(int error, List<XPGWifiDevice> devicesList) {
      super.didDiscovered(error, devicesList);
      //获取设备列表
      List<XPGWifiDevice> xpgWifiDeviceList = devicesList;

      }
```

```
XPGWifiDevice xpgWifiDevice = null;
for (int i = 0; i < xpgWifiDeviceList.getsize(); i++) {
      XPGWifiDevice device = xpgWifiDeviceList.get(i);
      if (device != null) {
            if (device != null && device.getMacAddress().equals("c89346abcd"))
            {
                  xpgWifiDevice = device;
                  break;
            }
      }
}
```

2.遍历设备列表，从设备列表中获取某个指定的device

```
xpgWifiDevice.setListener(new XPGWifiDeviceListener(){
      @Override
      public void didLogin(XPGWifiDevice device, int result) {
            super.didLogin(device, result);
            if(result==XPGWifiErrorCode.XPGWifiError_NONE){
                  //登录设备成功，可控制
                  …
            }else{
                  //登录设备失败，弹出对话框
                  …
            }
      }

});
```

3.注册设备监听器

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 每连接一个新的设备，都应该执行一次注册监听器方法

3.4混淆打包配置

如果的项目使用了 Proguard 混淆打包，为了避免 SDK 被二次混淆导致无法正常使用 SDK，请务必在 proguard-project.txt 中添加以下代码：

```
-libraryjars libs/XPGWIFISDK.jar

-dontwarn com.xtremeprog.**
-keep class com.xtremeprog.**{
      *;
 }
```

并在 project.properties 中指向Android混淆文件

```
proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt
```

4用户系统

概述

机智云的用户系统包含了用户的注册，登录等部分，以APP ID区分用户系统。APP ID的用户系统相互独立。

使用前须知

开发者需要先获取到相关的APP ID并已经正确的初始化了SDK和注册了XPGWifiSDKListener。以下涉及到的监听器注册方法都使用文档中3.3.1节的第二种方法，即使用子类继承基类的方式实现回调。

基本功能

4.1用户注册

概述

机智云的用户注册，分为实名注册和匿名注册两种。实名注册适用于有登录界面，必须用户注册登录以后才能使用的APP。匿名注册适用于没有登录界面，由后台管理用户账号的APP。

4.1.1普通账号注册

描述

普通账号注册属于实名注册的一种。使用用户自定义的账号和密码进行注册。

方法

```
XPGWifiSDK.sharedInstance().registerUser(String userName, String password);
```

输入参数

参数	描述
userName	用户名（只是用户名，跟手机号、邮箱、第三方账号无关）
password	密码

回调

```
XPGWifiSDKListener.didRegisterUser(int error, String errorMessage,
String uid, String token)
```

代码范例

1.注册监听器

参考3.3.1

2.调用用户注册方法

```
XPGWifiSDK.sharedInstance().registerUser("HelloGizwits", "12345678");
```

3.注册结果回调

```
@Override
protected void didRegisterUser(int result, String errorMessage, String uid,
            String token)
{
      if(result==XPGWifiErrorCode.XPGWifiError_NONE)
      {
            //注册成功，获取到uid和token

      }else
      {
            //注册失败，弹出错误信息

      }
}
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI

- 除了普通用户注册，还可以使用手机号、电子邮箱、匿名注册。
- 使用这些API，需保证已经正确初始化SDK和监听器

4.1.2手机号注册

描述

手机号注册属于实名注册的一种。使用用户的手机号码进行注册，注册之前需要先获取到手机验证码。

方法

```
//获取手机验证码
XPGWifiSDK.sharedInstance().requestSendVerifyCode(String phone);
//手机号注册
XPGWifiSDK.sharedInstance().registerUserByPhoneAndCode(String phone, String password, String code);
```

输入参数

1.获取手机验证码

参数	描述
phone	电话号码

2.手机号注册用户

参数	描述
phone	电话号码
password	密码
code	手机验证码

回调

1.获取手机验证码

```
XPGWifiSDKListener.didRequestSendVerifyCode(int error, String errorMessage)
```

2.手机号注册用户

```
XPGWifiSDKListener.didRegisterUser(int error, String errorMessage, String uid, String token)
```

代码范例

1.注册监听器

参考3.3.1

2.调用请求手机验证码接口

```
XPGWifiSDK.sharedInstance().requestSendVerifyCode("13800138000");
```

3. 请求验证码接口回调

```
@Override
protected void didRequestSendVerifyCode(int result, String errorMessage) {
      if(result==XPGWifiErrorCode.XPGWifiError_NONE)
      {
            //请求成功，等待包含验证码的手机短信

      }else
      {
            //请求失败，弹出错误信息

      }
}
```

4.手机收到包含验证码的短信

如“［机智云］您的验证码是012345”

5.调用手机注册接口

```
XPGWifiSDK.sharedInstance().registerUserByPhoneAndCode("13800138000", "admin",
"012345");
```

6.注册结果回调

```
@Override
protected void didRegisterUser(int result, String errorMessage, String uid,
            String token)
{
      if(result==XPGWifiErrorCode.XPGWifiError_NONE)
      {
            //注册成功，获取到uid和token

      }else
      {
            //注册失败，弹出错误信息

      }
}
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 除了手机号注册，还可以使用用户名、电子邮箱、匿名注册。

- 使用这些API，需保证已经正确初始化SDK和监听器

4.1.3邮箱注册

描述

邮箱注册属于实名注册的一种。使用用户的电子邮箱地址进行注册。

方法

```
//手机号注册
XPGWifiSDK.sharedInstance().registerUserByEmail(String email, String password);
```

输入参数

参数	描述
email	电子邮箱地址
password	密码

回调

```
XPGWifiSDKListener.didRegisterUser(int error, String errorMessage,
String uid, String token)
```

代码范例

1.注册监听器

参考3.3.1

2.调用邮箱注册接口

```
XPGWifiSDK.sharedInstance().registerUserByEmail("my_email@163.com", "admin");
```

6.注册结果回调

```
@Override
protected void didRegisterUser(int result, String errorMessage, String uid,
            String token)
{
      if(result==XPGWifiErrorCode.XPGWifiError_NONE)
      {
            //注册成功，获取到uid和token

      }else
      {
            //注册失败，弹出错误信息

      }
}
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 除了邮箱注册，还可以使用手机号、电子邮箱、匿名注册。
- 使用这些API，需保证已经正确初始化SDK和监听器

4.1.4匿名注册

描述

- 匿名注册接口与匿名登录为同一个接口。
- 该种方式一般适用于不需要用户登录注册的APP。
- 用户调用后直接登录，每次匿名登录获取到的did是一样的。
- 该接口原理使用Android ID进行注册和登录，每个Android系统都有一个独立的Android ID，系统刷机后将改变。因此，系统刷机后匿名注册的用户信息将没办法保留。

方法

```
//匿名注册
XPGWifiSDK.sharedInstance().userLoginAnonymous();
```

输入参数

无

回调

```
XPGWifiSDKListener.didUserLogin(int result, String errorMessage,
String uid,String token)
```

代码范例

1.注册监听器

参考3.3.1

2.调用匿名注册接口

```
XPGWifiSDK.sharedInstance().userLoginAnonymous();
```

3.注册结果回调

```
@Override
     protected void didUserLogin(int result, String errorMessage, String uid,
                String token) {
          if(result==XPGWifiErrorCode.XPGWifiError_NONE)
          {
                //登录成功，获取到uid和token

          }else
          {
                //登录失败，弹出错误信息

          }
     }
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 除了匿名注册，还可以使用用户名、手机号、电子邮箱等实名注册。
- 使用这些API，需保证已经正确初始化SDK和监听器

4.2用户登录

概述

- 机智云的用户登录，分为实名登录和匿名登录两种。
- 实名登录适用于设计了登录界面，必须用户注册登录以后才能使用的APP。
- 匿名登录适用于没有设计登录界面，由后台自动生成用户账号的APP。
- 登录后能获取到最新的token，通过该token能进行获取绑定设备列表，大循环控制设备等操作。
- token的有效期为7天，在7天内token不变，可直接使用。

4.2.1实名登录

描述

通过用户名、手机号、邮箱地址登录的过程称为实名登录。

方法

```
XPGWifiSDK.sharedInstance().userLoginWithUserName(String userName, String password);
```

输入参数

参数	描述
userName	用户名(普通用户、电子邮箱、电话号码)
password	密码

回调

```
XPGWifiSDKListener.didUserLogin(int result, String errorMessage, String uid,String token)
```

代码范例

1.注册监听器

参考3.3.1

2.调用实名用户登录接口

```
XPGWifiSDK.sharedInstance().userLoginWithUserName("13800138000", "12345678");
```

3.登录结果回调

```
@Override
    protected void didUserLogin(int result, String errorMessage, String uid,
              String token) {
        if(result==XPGWifiErrorCode.XPGWifiError_NONE)
        {
            //登录成功，获取到uid和token

        }else
        {
            //登录失败，弹出错误信息

        }
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 除了实名注册，还可以使用匿名登录
- 使用这些API，需保证已经正确初始化SDK和监听器
- 登录的用户名必须是已经成功注册的，且登录时的APP ID应与注册时的一致

4.2.2匿名登录

与匿名注册一致

4.2.3第三方用户登录

描述

除了使用在机智云实名或匿名注册的账号登录意外，还能使用第三方合作账号来实现登录。
目前支持的第三方账号有:

- 百度
- 新浪
- 腾讯(服务器未支持)

用户可以使用这三者的API获取到uid和token进行机智云登录，具体获取uid和token的方法请参考这三个平台的开发文档。

方法

```
XPGWifiSDK.sharedInstance().userLoginWithThirdAccountType(XPGWifiThirdAccountType thirdAccountType, String uid, String token);
```

输入参数

参数	描述
thirdAccountType	账号类型（区分新浪、腾讯、百度）
uid	第三方api获取的uid
token	第三方api获取的token

回调

```
XPGWifiSDKListener.didUserLogin(int result, String errorMessage,
String uid,String token)
```

代码范例

1.通过新浪（百度、腾讯）api获取uid和token

具体方法请参考各第三方平台的开发者文档

2.注册监听器

参考3.3.1

3.调用第三方账号登录接口（以新浪为例）

```
XPGWifiSDK.sharedInstance().userLoginWithThirdAccountType(XPGWifiThirdAccountTy
pe.XPGWifiThirdAccountTypeSINA, "12341234", "ACCESS_TOKEN");
```

4.登录结果回调

```
@Override
    protected void didUserLogin(int result, String errorMessage, String uid,
              String token) {
        if(result==XPGWifiErrorCode.XPGWifiError_NONE)
        {
            //登录成功，获取到uid和token

        }else
        {
            //登录失败，弹出错误信息

        }
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 使用第三方账号登录，无需在机智云上注册即可直接登录

4.3用户注销

描述

已登录的用户可通过该方法注销，注销后原token失效（该接口未生效）

方法

```
XPGWifiSDK.sharedInstance().userLogout(String uid);
```

输入参数

参数	描述
uid	用户ID

回调

```
XPGWifiSDKListener.didUserLogout(int result, String errorMessage)
```

代码范例

1.注册监听器

参考3.3.1

2.调用用户注销接口

```
XPGWifiSDK.sharedInstance().userLogout(XPGWifiThirdAccountType.XPGWifiThirdAccountTypeSINA, "12341234", "ACCESS_TOKEN");
```

3.注销结果回调

```
    @Override
    protected void didUserLogout(int result, String errorMessage) {
        if(result==XPGWifiErrorCode.XPGWifiError_NONE){
            //用户注销成功，返回登录界面

        }else{
            //用户注销失败，弹出错误信息

        }
    }
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 注销以后旧token无法正常登录

4.4重置密码

概述

- 如果忘记了用户密码，可通过重置的方式设置新的密码
- 手机号重置需要接收验证码
- 邮箱重置需要进入邮箱，根据链接提示进行充值

4.4.1手机号重置密码

描述

- 通过手机号和验证码重置账户登录密码
- 重置前需要先获取到手机验证码

方法

```
XPGWifiSDK.sharedInstance().changeUserPasswordByCode(String phone,String code,String newPassword);
```

输入参数

参数	描述
phone	已注册手机号
code	手机验证码
newPassword	新密码

回调

```
XPGWifiSDKListener.didChangeUserPassword(int result, String errorMessage)
```

代码范例

1.注册监听器

参考3.3.1

2.获取手机验证码

与 4.1.2手机号注册 中获取手机验证码调用过程一致

3.调用手机号重置密码接口

```
XPGWifiSDK.sharedInstance().changeUserPasswordByCode("13800138000", "012345", "newPsw");
```

4.重置密码接口回调

```
@Override
    protected void didChangeUserPassword(int result, String errorMessage) {
        if(result==XPGWifiErrorCode.XPGWifiError_NONE){
            //用户重置密码成功，返回登录界面

        }else{
            //用户重置密码失败，弹出错误信息

        }
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 手机号重置密码，只供用手机号注册的用户使用

4.4.2邮箱重置密码

描述

- 通过发送到指定邮箱的邮件中的安全链接重置账户登录密码
- 重置需要到注册邮箱中查收邮件，并按邮件指示执行重置操作

方法

```
XPGWifiSDK.sharedInstance().changeUserPasswordByEmail(String email);
```

输入参数

参数	描述
email	已注册电子邮箱

回调

```
XPGWifiSDKListener.didChangeUserPassword(int result, String
errorMessage)
```

代码范例

1.注册监听器

参考3.3.1

2.调用手机号重置密码接口

```
XPGWifiSDK.sharedInstance().changeUserPasswordByEmail("example@gizwits.com");
```

3.重置密码接口回调

```
@Override
    protected void didChangeUserPassword(int result, String errorMessage) {
        if(result==XPGWifiErrorCode.XPGWifiError_NONE){
            //重置密码邮件发送成功，提示用户查收

        }else{
            //重置密码邮件发送失败，弹出错误信息

        }
```

4.到邮箱查收邮件，按邮件提示执行重置操作，如下：

点击如下链接重置您的【智能云空调】密码：http://api.gizwits.com:80/web/reset_password/ecb16888bb794c68b1xxx6f8247fxxxx/2OFCJY

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 邮箱重置密码，只供用邮箱注册的用户使用
- 邮件发送成功回调与密码修改成功回调一致，因此需要注意在回调的时候区分
- 重置密码邮件有可能进入用户的邮箱的垃圾箱中，需提醒用户

4.5密码修改

描述

用户使用有效的token和旧密码修改新密码

方法

```
XPGWifiSDK.sharedInstance().changeUserPassword(String token,String oldPassword,String newPassword);
```

输入参数

参数	描述
token	授权令牌
oldPassword	旧密码
newPassword	新密码

回调

```
XPGWifiSDKListener.didChangeUserPassword(int result, String errorMessage)
```

代码范例

1.注册监听器

参考3.3.1

2.调用修改密码接口

```
XPGWifiSDK.sharedInstance().changeUserPassword("e5a37a73e8ad40000753a5099d125e08", "oldPassword", "newPassword");
```

3.重置密码接口回调

```
@Override
      protected void didChangeUserPassword(int result, String errorMessage) {
            if(result==XPGWifiErrorCode.XPGWifiError_NONE){
                  //密码修改成功，返回登录界面

            }else{
                  //密码修改失败，弹出错误信息

            }
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 使用的token需保证是真实有效且未过期的
- 重置密码邮件发送成功回调与密码修改成功回调一致，因此需要注意在回调的时候区分

4.6匿名用户改实名用户

概述

- 匿名注册的用户，可以通过更改为实名用户的方法使用自定义的账号密码进行登录。
- 转换后原匿名账号失效

4.6.1匿名用户改普通用户

描述

匿名用户通过token和自定义的用户名和密码进行更改为实名用户。

方法

```
XPGWifiSDK.sharedInstance().transAnonymousUserToNormalUser(String token,String
userName,String password);
```

输入参数

参数	描述
token	授权令牌
userName	自定义用户名
password	密码

回调

```
XPGWifiSDKListener.didTransUser(int result, String errorMessage)
```

代码范例

1.注册监听器

参考3.3.1

2.调用匿名改实名普通用户接口

```
XPGWifiSDK.sharedInstance().transAnonymousUserToNormalUser("e5a37a73e8ad40000753a5099d125e08", "myGizwits", "1234567890");
```

3.重置密码接口回调

```
@Override
public void didTransUser(int result, String errorMessage) {
      if(result==XPGWifiErrorCode.XPGWifiError_NONE){
            //用户更改成功，返回登录界面

      }else{
            //用户更改失败，弹出错误信息

      }
};
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 使用的token需保证是真实有效且未过期的

4.6.2匿名用户改手机实名用户

描述

匿名用户通过token和手机验证码进行更改为实名用户。

方法

```
XPGWifiSDK.sharedInstance().transAnonymousUserToPhoneUser(String token,String userName,String password,String code);
```

输入参数

参数	描述
token	授权令牌
userName	自定义用户名
password	密码
code	手机验证码

回调

```
XPGWifiSDKListener.didTransUser(int result, String errorMessage)
```

代码范例

1.注册监听器

参考3.3.1

2.获取手机验证码

与 4.1.2手机号注册 中获取手机验证码调用过程一致

3.调用匿名改手机实名用户接口

```
XPGWifiSDK.sharedInstance().transAnonymousUserToPhoneUser("e5a37a73e8ad40000753a5099d125e08", "myGizwits", "1234567890", "213451");
```

4.重置密码接口回调

```
@Override
public void didTransUser(int result, String errorMessage) {
      if(result==XPGWifiErrorCode.XPGWifiError_NONE){
            //用户更改成功，返回登录界面

      }else{
            //用户更改失败，弹出错误信息

      }
};
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 使用的token需保证是未过期的

4.7实名用户更改用户名

概述

- 已经通过手机或邮箱实名注册的用户，可以调用改接口修改用户名
- 修改的用户名必须是未被使用的
- 该功能适用于用户更换手机号或更换电子邮箱以后仍想保留设备列表
- 修改后两个账号都保留

4.7.1更改邮箱

描述

用邮箱注册的用户可以通过该方法更改邮箱地址，更改后使用新邮箱登录，原邮箱的设备列表等信息将转移到邮箱上

方法

```
XPGWifiSDK.sharedInstance().changeUserEmail(String token,String email);
```

输入参数

参数	描述
token	授权令牌
email	新邮箱地址

回调

```
XPGWifiSDKListener.didChangeUserEmail(int result, String
errorMessage)
```

代码范例

1.注册监听器

参考3.3.1

2.调用修改邮箱地址接口

```
XPGWifiSDK.sharedInstance().changeUserEmail("e5a37a73e8ad40000753a5099d
125e08", “example2@gizwits.com”);
```

3.修改邮箱地址接口回调

```
@Override
public void didChangeUserEmail(int result, String errorMessage) {
      if(result==XPGWifiErrorCode.XPGWifiError_NONE){
            //邮箱更改成功，返回登录界面

      }else{
            //用户更改失败，弹出错误信息

      }
};
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 使用的token需保证是未过期的
- 欲修改的邮箱需保证是未注册过的

4.7.2更改手机号

描述

- 用手机号用户可以通过该方法更改手机号，更改后使用手机号登录，原手机号的设备列表等信息将转移到新手机号上
- 修改前需获取新手机号的短信验证码

方法

```
XPGWifiSDK.sharedInstance().changeUserPhone(String token,String phone,String code);
```

输入参数

参数	描述
token	授权令牌
phone	新电话号码
code	短信验证码

回调

```
XPGWifiSDKListener.didChangeUserPhone(int result, String errorMessage)
```

代码范例

1.注册监听器

参考3.3.1

2.获取手机验证码

与 4.1.2手机号注册 中获取手机验证码调用过程一致

3.调用修改手机号码接口

```
XPGWifiSDK.sharedInstance().changeUserPhone("e5a37a73e8ad40000753a5099d125e08", "13800123456", "112233");
```

4.修改手机号码回调

```
@Override
public void didChangeUserPhone(int result, String errorMessage) {
      if(result==XPGWifiErrorCode.XPGWifiError_NONE){
            //邮箱更改成功，返回登录界面

      }else{
            //用户更改失败，弹出错误信息

      }
};
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 使用的token需保证是未过期的
- 欲修改的手机号码需保证是未注册过的

5设备管理相关

概述

机智云设备是指使用机智云GAgent模块和使用烧写了机智云串口协议的mcu的智能设备。该API主要包括设备的配置上线，绑定和控制等方法。

使用前须知

- 开发者需要先获取到相关的APP ID并已经正确的初始化了SDK和注册了XPGWifiSDKListener。
- 以下涉及到的监听器注册方法都使用文档中3.3.1节的第二种方法，即使用子类继承基类的方式实现回调。
- 智能设备需正确烧写了GAgent和机智云串口通讯协议。

基本功能

5.1设备配置上线（onBoarding）

概述

- 控制设备前，需要先配置模块连上路由器
- 连上路由器的设备，如果能练上万维网，会自动连接机智云进行注册

5.1.1airlink一键配置

描述

- airlink使用dup广播方式，由手机端发出含有目标路由器名称和密码的广播，Wi-Fi模块接收到广播后自动连接目标路由器，连上路由器以后发出配置成功广播，通知手机配置完成。

方法

```
XPGWifiSDK.sharedInstance().setDeviceWifi(String ssid,String
key,XPGWifiConfigureMode mode,int timeout);
```

输入参数

参数	描述
ssid	Wi-Fi名称
key	Wi-Fi密码
mode	配置模式
timeout	超时时间(秒)

回调

```
XPGWifiSDKListener.didSetDeviceWifi(int result, XPGWifiDevice
device)
```

代码范例

1.注册监听器

参考3.3.1

2.手机连上目标Wi-Fi并获取到Wi-Fi的SSID

```
/**
 * 获取当前WIFI的SSID.
 *
 * @param context 上下文
 * @return ssid
 */
public static String getCurentWifiSSID(Context context){
      String ssid = "";
      if(context!=null)
      {
            WifiManager wifiManager =
(WifiManager)context.getSystemService(Context.WIFI_SERVICE);
            WifiInfo wifiInfo = wifiManager.getConnectionInfo();
            ssid = wifiInfo.getSSID();
            if (ssid.substring(0, 1).equals("\"")&&ssid.substring(ssid.length()
- 1).equals("\""))
                  {
                        ssid = ssid.substring(1, ssid.length() - 1);
                  }
      }
            return ssid;
}
```

3.mcu发出开启airlink串口指令，通知模块开启airlink模式。

详情请参考《智能云空调-机智云接入串口通信协议文档》

4.调用设置Wi-Fi接口，发出udp广播

```
XPGWifiSDK.sharedInstance().setDeviceWifi("my_home", "abc123abc",
XPGWifiConfigureMode.XPGWifiConfigureModeAirLink, 60);
```

5.等待配置完成或超时，回调配置完成接口

```
@Override
protected void didSetDeviceWifi(int result, XPGWifiDevice device) {

      if(result==XPGWifiErrorCode.XPGWifiError_NONE){
            //设备配置成功，返回设备列表界面进行绑定操作

      }else{
            //用户配置失败，弹出错误信息，重试或进行softap配置

      }
}
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- airlink配置如果超时时间还未结束，无法进行下一次配置
- 模块开启airlink模式，一分钟内未收到airlink广播或无法正确连上路由器，将进入soft ap模式

5.1.2softap配置

描述

- soft ap配置是模块连接不上路由器时或没有路由器信息时自动转换为热点模式，手机连接热点后发送配置信息，模块根据配置信息自动连接路由器的过程

方法

```
XPGWifiSDK.sharedInstance().setDeviceWifi(String ssid,String
key,XPGWifiConfigureMode mode,int timeout);
```

输入参数

参数	描述
ssid	Wi-Fi名称
key	Wi-Fi密码
mode	配置模式
timeout	超时时间(秒)

回调

```
XPGWifiSDKListener.didSetDeviceWifi(int result, XPGWifiDevice
device)
```

代码范例

1.注册监听器

参考3.3.1

2.手机连上目标Wi-Fi并获取到Wi-Fi的SSID

```
/**
 * 获取当前WIFI的SSID.
 *
 * @param context 上下文
 * @return ssid
 */
public static String getCurentWifiSSID(Context context){
     String ssid = "";
     if(context!=null)
     {
          WifiManager wifiManager =
(WifiManager)context.getSystemService(Context.WIFI_SERVICE);
           WifiInfo wifiInfo = wifiManager.getConnectionInfo();
           ssid = wifiInfo.getSSID();
           if (ssid.substring(0,
1).equals("\"")&&ssid.substring(ssid.length() - 1).equals("\""))
                {
                     ssid = ssid.substring(1, ssid.length() - 1);
                }
     }
```

3.mcu使用airlink配置失败（密码错或udp广播无法正常收发），Wi-Fi模块转为热点模式

详情请参考《智能云空调-机智云接入串口通信协议文档》

4.手机连接Wi-Fi热点，热点名”XPG-GAgent-xxxx”，连接时需要输入密码123456789

5.调用设置Wi-Fi接口，设置设备Wi-Fi

```
XPGWifiSDK.sharedInstance().setDeviceWifi("my_home", "abc123abc",
XPGWifiConfigureMode.XPGWifiConfigureModeSoftAP, 60);
```

6.模块收到配置信息，尝试连接路由器并自动关闭热点

7.指引用户连接回目标路由器

8.模块连上路由器以后，发出配置成功广播，sdk回调配置接口，或超时回调

```
@Override
protected void didSetDeviceWifi(int result, XPGWifiDevice device) {

      if(result==XPGWifiErrorCode.XPGWifiError_NONE){
            //设备配置成功，返回设备列表界面进行绑定操作

      }else{
            //用户配置失败，弹出错误信息，重试或进行softap配置

      }
}
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- soft ap模式只有模块无法正常连接路由器时才会开启
- 因为配置成功的广播包只有app连上同一路由器才能收取，因此应该预留app连接路由器的时间

5.2搜索设备

描述

- 搜索设备会先获取云端已绑定设备然后获取局域网内设备
- 局域网搜索的过程是先由app发出udp广播，模块接收后将回复包含ip，mac等信息的数据帧给app，app收到一次响应则回调一次接口
- 由于每个设备响应的时间不同，因此该方法不建议同时调用多次
- 该方法可通过指定productkey的方式获取特定类型的设备

方法

```
XPGWifiSDK.sharedInstance().getBoundDevices(String uid,String token,String…
specialProductKey);
```

输入参数

参数	描述
uid	用户ID
token	授权令牌
specialProductKey	指定productkey（可变参数）,系统会自自动下载对应的配置文文件，如果配置存在则不会覆盖。如不指定则返回所有类型设备。

回调

```
XPGWifiSDKListener.didDiscovered(int result, List<XPGWifiDevice>
devicesList)
```

代码范例

1.注册监听器

参考3.3.1

2.调用获取设备列表接口(使用时需要替换为自己的uid、token、productkey)

```
XPGWifiSDK.sharedInstance().getBoundDevices("8e1253f8e430000ea5e5a01de7d60ed9",
"e5a37a73e8ad40000753a5099d125e08", "e3cf7332b7834a03a92d9e14a3f6d352");
```

3.设备列表回调（每搜到一个设备会回调一次）

```
private List<XPGWifiDevice> xpgWifiDeviceList;
…
@Override
      protected void didDiscovered(int result, List<XPGWifiDevice> devicesList)
{
            if(result==XPGWifiErrorCode.XPGWifiError_NONE&&devicesList.size()>0)
{
                  //获取设备列表
                  xpgWifiDeviceList = devicesList;
            }else{
                  //获取失败或未发现设备，重试
            }
      }
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 因为每个设备响应的时间不一样，因此回调接口可能会回调多次，建议在一定时间内不要重复多次调用搜索设备方法，避免与UI线程冲突

5.3绑定设备

描述

- 设备信息与账号信息绑定后，可通过获取列表的方式获取到该设备
- 绑定后的设备在不同的手机上登录帐号都可获取到
- 绑定可通过本地列表返回的设备类进行，也可以通过扫描二维码或其他特定方式获取did和passcode来进行
- 设备无法被同一个帐号绑定多次，除非是修改remark

方法

```
XPGWifiSDK.sharedInstance().bindDevice(String uid, String token, String did,
String passCode, String remark);
```

输入参数

参数	描述
uid	用户ID
token	授权令牌
did	设备ID
passcode	设备密码
remark	设备别名

回调

```
XPGWifiSDKListener.didDiscovered(int result, List<XPGWifiDevice>
devicesList)
```

代码范例

1.注册监听器

参考3.3.1

2.调用绑定设备方法 (实际开发中请将uid、token、did、passcode、remark替换成实际的值)

```
 XPGWifiSDK.sharedInstance().bindDevice("uid", "token", "did", "passCode",
"remark");
```

3.绑定结果回调

```
@Override
      protected void didBindDevice(int result, String errorMessage, String did)
{
            if(result==XPGWifiErrorCode.XPGWifiError_NONE){
                  //绑定设备成功，登录设备进行控制

            }else{
                  //绑定设备失败，弹出错误信息

            }
      }
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 使用的token必须是真实有效且未过期的
- 如设备未能连接云端，可能获取不到did

5.4解除绑定设备

描述

- 解除绑定的设备将无法再次通过获取绑定列表获取，只能从本地未绑定列表中进行重新绑定

方法

```
XPGWifiSDK.sharedInstance().unbindDevice(String uid, String token, String did,
String passCode);
```

输入参数

参数	描述
uid	用户ID
token	授权令牌
did	设备ID
passcode	设备密码

回调

```
XPGWifiSDKListener.didUnbindDevice(int result, String
errorMessage, String did)
```

代码范例

1.注册监听器

参考3.3.1

2.调用解除设备绑定方法 (实际开发中请将uid、token、did、passcode替换成实际的值)

```
 XPGWifiSDK.sharedInstance().unbindDevice("uid", "token", "did", "passCode");
```

3.绑定结果回调

```
@Override
     protected void didUnbindDevice(int result, String errorMessage, String
did) {
           if(result==XPGWifiErrorCode.XPGWifiError_NONE){
                 //解除绑定设备成功，返回设备列表

           }else{
                 //解除绑定设备失败，弹出错误信息

           }
     }
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 使用的token必须是真实有效且未过期的

5.5设备登录

描述

- 登录设备后可对设备进行控制以及状态获取
- 登录分为大循环登录和小循环登录，大小循环由sdk内部自行判断，开发者无需特别处
- sdk会优先进行小循环登录，小循环登录失败后才会尝试进行大循环登录
- 登录设备成功后，sdk会一直和设备保持连接
- 登录设备之前，需要先获取到XPGWifiDevice类，且该设备是在线的

方法

```
XPGWifiDevice.login(String uid, String token);
```

输入参数

参数	描述
uid	用户ID
token	授权令牌

回调

```
XPGWifiDeviceListener.didLogin(XPGWifiDevice device, int result)
```

代码范例

1.从设备列表中获取设备类以及注册监听器

参考3.3.2

2.调用设备登录方法(实际开发中请将uid、token替换成实际的值)

```
//已从设备列表中获取到XPGWifiDevice实体mXpgWifiDevice
mXpgWifiDevice.login("uid", "token");
```

3.登录结果回调

```
@Override
protected void didLogin(XPGWifiDevice device, int result)
{
	if(result==XPGWifiErrorCode.XPGWifiError_NONE)
	{
		//设备登录成功，可进行控制

	}else{
		//设备登录失败，弹出错误信息

	}
}
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 使用的token必须是真实有效且未过期的
- 建议每次只登录一台设备，要登录另一台设备前先断开已连接设备
- 如果app进入后台，连接可能因为Android系统资源回收而断开连接

5.6设备硬件信息

描述

- 获取模块协议版本号，mcu固件版本号等硬件信息
- 只有小循环连接设备后才能获取到硬件信息

方法

```
XPGWifiDevice.getHardwareInfo();
```

输入参数

无

回调

```
XPGWifiDeviceListener.didQueryHardwareInfo(XPGWifiDevice device,
int result, ConcurrentHashMap<String,String> hardwareInfo)
```

代码范例

1.从设备列表中获取设备类以及注册监听器

参考3.3.2

2.小循环登录设备

参考5.5

3.调用获取设备硬件信息方法

```
//已从设备列表中获取到XPGWifiDevice实体mXpgWifiDevice
mXpgWifiDevice.getHardwareInfo();
```

4.硬件信息回调

```
@Override
public void didQueryHardwareInfo(XPGWifiDevice device, int result,
ConcurrentHashMap<String,String> hardwareInfo) {
      if (result==XPGWifiErrorCode.XPGWifiError_NONE) {
                        //获取硬件信息成功，刷新UI
            StringBuilder sb = new StringBuilder();
            sb.append("Wifi Hardware
Version:"+hardwareInfo.get(XPGWifiDevice.XPGWifiDeviceHardwareWifiHardVerKey)
+"\r\n");
            sb.append("Wifi Software
Version:"+hardwareInfo.get(XPGWifiDevice.XPGWifiDeviceHardwareWifiSoftVerKey)
+"\r\n");
            sb.append("MCU Hardware
Version:"+hardwareInfo.get(XPGWifiDevice.XPGWifiDeviceHardwareMCUHardVerKey)
+"\r\n");
            sb.append("MCU Software
Version:"+hardwareInfo.get(XPGWifiDevice.XPGWifiDeviceHardwareMCUSoftVerKey)
+"\r\n");
            sb.append("Firmware
Id:"+hardwareInfo.get(XPGWifiDevice.XPGWifiDeviceHardwareFirmwareIdKey)+"\r
\n");
            sb.append("Firmware
Version:"+hardwareInfo.get(XPGWifiDevice.XPGWifiDeviceHardwareFirmwareVerKey)
+"\r\n");
            sb.append("Product
Key:"+hardwareInfo.get(XPGWifiDevice.XPGWifiDeviceHardwareProductKey)+"\r\n");
            sb.append("Device id:"+device.getDid()+"\r\n");

            Message msg = new Message();
            msg.what = "HARD_INFO";
            msg.obj = sb.toString();
            handler.sendMessage(msg);
      }else{
            //获取硬件信息失败，弹出错误信息
      }
};
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 硬件信息只能在小循环连接后才能获取

5.7设备断开

描述

- 断开与已登录设备的连接
- 建议登录一个新的设备前，先断开前一个设备的连接

方法

```
XPGWifiDevice.disconnect();
```

输入参数

无

回调

```
XPGWifiDeviceListener.didDisconnected(XPGWifiDevice device)
```

代码范例

1.设备已登录

参考5.5

3.调用设备断开方法

```
//已从设备列表中获取到XPGWifiDevice实体mXpgWifiDevice
mXpgWifiDevice.disconnect();
```

4.断开设备回调

```
@Override
	protected void didDisconnected(XPGWifiDevice device) {
		//设备断开，弹出提示
	}
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 设备被动断开(网络异常、系统资源回收)也会回调didDisconnected接口，需要在逻辑上做好主动断开和被动断开的区别

6控制设备和接收设备信息

概述

SDK通过JSON的方式进行设备的控制和状态接受。SDK接受到APP传入JSON串后解析为二进制数据，发送给模块。反之同理。

使用前须知

- 开发者需要先获取到相关的APP ID并已经正确的初始化了SDK和注册了XPGWifiSDKListener。
- 以下涉及到的监听器注册方法都使用文档中3.3.1节的第二种方法，即使用子类继承基类的方式实现回调。
- 智能设备需正确烧写了GAgent和机智云串口通讯协议。
- 发送的指令必须是数据点上已定义的指令

基本功能

6.1发送控制指令

描述

根据定义的数据点向设备发送控制指令

方法

```
XPGWifiDevice.write(String jsonData);
```

输入参数

参数	描述
jsonData	JSON数据串，具体定义可看数据点协议文档 http://site.gizwits.com/document/m2m/datapoint/

回调

机智云要求mcu在自身状态改变后都上报一次全状态到Wi-Fi模块，Wi-Fi模块上报云端及已连接手机端。
sdk收到状态后回调

```
XPGWifiDeviceListener.didReceiveData(XPGWifiDevice
device,ConcurrentHashMap<String, Object> dataMap, int result)
```

代码范例

1.设备已登录

参考5.5

2.发送控制指令

```
/**
 * 发送指令.
 *
 * @param xpgWifiDevice
 *            机智云设备对象
 * @param key
 *            数据点标识名
 * @param value
 *            数据点数值
 */
public void cWrite(XPGWifiDevice xpgWifiDevice, String key, Object value)
{

      try {
            //创建JSONObject 对象，用于封装所有数据
            final JSONObject jsonsend = new JSONObject();
            //写入命令字段（所有产品一致）
            jsonsend.put("cmd", 1);
            //创建JSONObject 对象，用于封装数据点
            JSONObject jsonparam = new JSONObject();
            //写入数据点字段
            jsonparam.put(key, value);
            //写入产品字段（所有产品一致）
            jsonsend.put("entity0", jsonparam);
            //调用发送指令方法
            xpgWifiDevice.write(jsonsend.toString());
      } catch (JSONException e) {
            e.printStackTrace();
      }

}

//例如发送开源空调
cWrite(xpgWifiDevice, "switch", true);
```

3.mcu改变状态后上报状态，sdk收到状态回调

```
@Override
public void didReceiveData(XPGWifiDevice device,
                     ConcurrentHashMap<String, Object> dataMap, int result)
{
            //收到状态信息，改变UI
}
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 具体的状态信息请参考6.2 接受设备状态说明

命令（CMD）映射表

功能	值
向设备发送控制指令	1
向设备请求设备状态	2
设备返回请求的设备状态	3
设备推送当前设备状态	4

6.2接收设备状态

描述

- 机智云要求mcu在自身状态改变后都上报一次全状态到Wi-Fi模块，Wi-Fi模块把状态上报云端及已连接手机端。
- 可通过发送｛”cmd”:2｝的指令查询状态

回调

机智云要求mcu在自身状态改变后都上报一次全状态到Wi-Fi模块，Wi-Fi模块上报云端及已连接手机端。
sdk收到状态后回调

```
XPGWifiDeviceListener.didReceiveData(XPGWifiDevice
device,ConcurrentHashMap<String, Object> dataMap, int result)
```

代码范例

1.设备已登录

参考5.5

2.mcu改变状态后上报状态，sdk收到状态回调

```
      @Override
      public boolean didReceiveData(XPGWifiDevice device,
ConcurrentHashMap<String,Object> dataMap, int result) {
            //普通数据点类型，有布尔型、整形和枚举型数据，该种类型一般为可读写
            if (dataMap.get("data") != null) {
                  Log.i("info", (String)dataMap.get("data"));
                  Message msg = new Message();
                  msg.obj = dataMap.get("data");
                  msg.what = RESP;
                  //收到后通知主线程UI刷新
                  handler.sendMessage(msg);
            }
            //设备报警数据点类型，该种数据点只读，设备发生报警后该字段有内容，没有发生报警则
没内容
            if (dataMap.get("alters") != null) {
                  Log.i("info", (String)dataMap.get("alters"));
                  Message msg = new Message();
                  msg.obj = dataMap.get("alters");
                  msg.what = LOG;
                  //收到后通知主线程UI刷新
                  handler.sendMessage(msg);
            }
            //设备错误数据点类型，该种数据点只读，设备发生错误后该字段有内容，没有发生报警则
没内容
            if (dataMap.get("faults") != null) {
                  Log.i("info", (String)dataMap.get("faults"));
                  Message msg = new Message();
                  msg.obj = dataMap.get("faults");
                  msg.what = LOG;
                  //收到后通知主线程UI刷新
                  handler.sendMessage(msg);
            }
            //二进制数据点类型，适合开发者自行解析二进制数据
            if (dataMap.get("binary") != null) {
                  Log.i("info", "Binary data:" +
bytesToHex((byte[])dataMap.get("binary")));
                  //收到后自行解析
            }

            return true;
      };
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 因为每次mcu上报状态，都会回调一次设备状态回调接口，因此应该做好UI的逻辑，避免多个状态上报时冲突。

7其他

7.1下载产品配置文件

描述

- 通过服务器下载 JSON 配置文文件。
- 配置文件，是定义 APP 与指定设备通信的规则。比如智能云空调，下载了智能空调的配置文件，就可以通过 6 控制设备和接收设备信息 里面定义的方法，才可以正常的发送和接收对应的控制指令。如果缺少该产品的配置文文件，APP 就不能控制该类产品的任何设备，不管代码怎么写都是没有作用的。
- 配置文件会由sdk自行下载，sdk发现已有该产品的配置文件以后，不会重新下载
- 调用下载方法以后会强制下载并覆盖原文件

方法

```
XPGWifiSDK.sharedInstance().updateDeviceFromServer(String productKey);
```

输入参数

参数	描述
productKey	产品标识码

回调

```
XPGWifiSDKListener.didUpdateProduct(int error, String productKey)
```

代码范例

1.注册监听器

参考3.3.1

2.调用下载配置文件方法(实际开发中请将productkey替换成实际的值)

```
XPGWifiSDK.sharedInstance().updateDeviceFromServer(String productKey);
```

3.下载配置文件结果回调

```
@Override
    protected void didUpdateProduct(int result, String productKey) {
        if(result==XPGWifiErrorCode.XPGWifiError_NONE){
            //更新配置文件成功，返回设备列表

        }else{
            //更新配置文件失败，弹出错误信息

        }
    }
```

注意事项

- 避免在回调接口中执行耗时任务，应使用Handler发消息等方式控制UI
- 使用这些API，需保证已经正确初始化SDK和监听器
- 下载配置文件必须保证手机能正常接入互联网
- 每个设备对应的文件命名：Product Key 加后缀 “.json”。除了通过 SDK 下载配置文文件外，开发者也可以编程把文件放到指定的位置，效果跟调用下载配置的 API 相同。具体做法，可以参考 https://github.com/gizwits/gokit-android 里面的写法。

8错误代码解释

标准错误代码

错误代码	描述
XPGWifiError_NONE	没有错误
XPGWifiError_GENERAL	一般错误
XPGWifiError_NOT_IMPLEMENTED	写入入数据的操作未执行
XPGWifiError_PACKET_DATALEN	读取或写入数据时，数据长度不在 0-65535 的范围
XPGWifiError_CONNECTION_ID	错误的连接 ID
XPGWifiError_CONNECTION_CLOSED	连接已关闭
XPGWifiError_PACKET_CHECKSUM	数据包的校验和不正确
XPGWifiError_LOGIN_FAIL	控制设备时，发现该设备没有登录过
XPGWifiError_MQTT_FAIL	执行MQTT 相关操作时出错
XPGWifiError_DISCOVERY_MISMATCH	发现小循环设备或者配置 AirLink 时，收到的数据包不能正确解析相关的内容
XPGWifiError_SET_SOCK_OPT	调用 setsockopt() 失败
XPGWifiError_THREAD_CREATE	线程创建失败
XPGWifiError_CONNECTION_POOL_FULLED	建立太多的连接，导致连接池满了。最大允许建立 255 个 TCP 连接
XPGWifiError_NULL_CLIENT_ID	大循环操作时，使用了空的 Client ID
XPGWifiError_CONNECTION_ERROR	连接出现错误
XPGWifiError_INVALID_PARAM	传入了错误的参数
XPGWifiError_CONNECT_TIMEOUT	连接超时。默认超时 1 分钟
XPGWifiError_INVALID_VERSION	数据包版本号错误
XPGWifiError_INSUFFIENT_MEM	不能分配内存
XPGWifiError_THREAD_BUSY	当前线程在使用中
XPGWifiError_HTTP_FAIL HTTP	操作失败
XPGWifiError_GET_PASSCODE_FAIL	获取 Passcode 失败
XPGWifiError_CONFIGURE_TIMEOUT	配置 on-boarding 超时

错误代码	描述
XPGWifiError_CONFIGURE_SENDFAILED	配置 on-boarding 时，发送失败
XPGWifiError_NOT_IN_SOFTAPMODE	错误，执行行Soft-AP 方法但不在 Soft-AP 模式
XPGWifiError_UNRECOGNIZED_DATA	接收到了不可识别的数据
XPGWifiError_CONNECTION_NO_GATEWAY	不能连接，无法获取到网关
XPGWifiError_CONNECTION_REFUSED	连接被拒绝
XPGWifiError_DESCRIPTOR_FAIL	没用到
XPGWifiError_PATH	没用到
XPGWifiError_DOMAIN_NAME	没用到
XPGWifiError_NULL_MAC	没用到

9相关资源

开发资源

GoKit demo Android： https://github.com/gizwits/gokit-android
智能云空调 iOS： https://github.com/gizwits/airconditioner-android

第三方方账号连接

BaiduSDK http://developer.baidu.com/wiki/index.php?title=%E5%B8%AE%E5%8A%A9%E6%96%87%E6%A1%A3%E9%A6%96%E9%A1%B5/%E7%99%BE%E5%BA%A6%E5%B8%90%E5%8F%B7%E8%BF%9E%E6%8E%A5

GOKIT 智能设备开发套件网站相关

GoKit 智能设备开发套件网网网站：site.gizwits.com

机智云数据点协议文文档：http://site.gizwits.com/document/m2m/datapoint/

通过产品识别码下载产品配置文文件（替换自己的product_key）：
http://site.gizwits.com/v2/datapoint?product_key=6f3074fe43894547a4f1314bd7e3ae0b&format=json

常见见问题汇总：

发布时链到一个页面即可

10联系方式

如果有什么其他问题，大家可以在 QQ 群和论坛上交流。
论坛的网网网址：http://club.gizwits.com/ 。
除此之外，您还可以给机智云的技术支持人人员发送邮件，反馈您在使用过程中遇到的任何问题。
邮箱：janel@gizwits.com
电话：4006525488