GizWifiSDK API Android 参考手册

修订记录

修改时间	修改内容	版本	修改人	备注
2016.5.10	更新目录	0.9.0	Pomia	
2016.6.14	校对	1.0.0	Pomia	
2016.9.7	增加定时任务接口	1.0.1	Pomia	
2016.9.27	增加新接口说明	1.0.2	Pomia	
2016.10.19	启动接口中增加域名、pk 过滤参数 设备配置时的模组类型增加一个自定义枚举值 旧的启动接口仍然兼容，但不推荐使用 旧的切换域名接口仍然兼容，但不推荐使用 定时任务接口已废弃，不推荐使用	1.0.3	Pomia	
2016.11.7	启动接口参数使用变更 增加设备全球域名部署接口	1.0.4	Pomia	
2016.11.30	startWithAppID 接口增加开启设备域名自动设置参数 setDeviceServerInfo 接口 mac 参数使用变更	1.0.5	Pomia	

1. GizWifiSDK 类

1.1. 简介

机智云 Wifi SDK 的基础类，为 APP 开发者提供设备配置和发现、设备分组、用户登录和注册等功能。

1.2. 属性方法

属性方法名	定义
setListener	`public void setListener(GizWifiSDKListener listener)`
getDeviceList	`public List<GizWifiDevice> getDeviceList()`

1.3. 回调接口

以下是 GizWifiSDK 提供的所有回调接口，将在在后续 API 定义中详细介绍：

- didNotifyEvent：SDK 系统事件通知
- didGetCurrentCloudService：服务域名独立部署的回调接口
- didDisableLAN：小循环是否禁用的回调接口
- didDiscovered：设备列表上报的回调接口
- didGetSSIDList：获取设备周围 Wi-Fi 热点列表的回调接口
- didSetDeviceOnboarding：设备配置结果的回调接口
- didBindDevice：设备绑定结果的回调接口
- didUnbindDevice：设备解除绑定结果的回调接口
- didUpdateProduct：设备配置文件上报的回调接口
- didCreateScheduler：创建定时任务的回调接口
- didDeleteScheduler：删除定时任务的回调接口
- didGetSchedulers：获取定时任务列表的回调接口
- didGetSchedulerStatus：查询定时任务执行状态的回调接口
- didGetCaptchaCode：获取图片验证码的回调接口
- didRequestSendPhoneSMSCode：请求手机短信验证码的回调接口
- didVerifyPhoneSMSCode：验证手机短信验证码的回调接口
- didRegisterUser：用户注册结果的回调接口
- didUserLogin：用户登录结果的回调接口
- didTransAnonymousUser：匿名用户转换的回调接口
- didChangeUserPassword：更换用户密码结果的回调接口

- didChangeUserInfo：修改用户信息结果的回调接口
- didGetUserInfo：获取用户信息的回调接口
- didGetGroups：获取用户设备分组列表的回调接口

1.4. API 定义

【sharedInstance】

定义	public static synchronized GizWifiSDK sharedInstance()
功能描述	获取 GizWifiSDK 单例的实例。
返回值	SDK 唯一的实例。
代码示例	GizWifiSDK mSDKInstance = GizWifiSDK.sharedInstance();

【setListener】

定义	public void setListener(GizWifiSDKListener listener)	
功能描述	设置 SDK 通用监听器	
参数	listener	GizWifiSDKListener 回调对象

代码示例	GizWifiSDK.sharedInstance().setListener(new GizWifiSDKListener() {
	  // app 根据自己的需要实现回调函数
	});	

【startWithAppID】

定义	public void startWithAppID(Context context, String appID, List<String> specialProductKeys, ConcurrentHashMap<String, String> cloudServiceInfo, boolean autoSetDeviceDomain)	
功能描述	初始化 SDK。该接口执行后，其他接口功能才能正常执行。如果已经设置了 delegate，SDK 会立即通过 didDiscovered 上报发现的设备。 如果 App 要做域名切换和设备的 productKey 过滤，建议在 SDK 初始化时就指定好要切换的域名和产品 productKey。 如果需要设置设备连接的云服务域名，可以在该接口调用时开启自动设置功能。这时 SDK 会让所有支持域名设置的设备都与 App 连接到同一个云服务域名上。但该接口默认是不开启此功能的。 注意：设备域名自动设置开启后会一直生效，但调用 setDeviceServerInfo 接口将会终止自动设置	
参数	context	上下文对象
	appid	在机智云开发者中心 dev.gizwits.com 中，每个注册的设备在对应的“应用配置”中，都能够查到对应的 appID。此参数无默认

		值，开发者必须传入正确的 appID
	specialProductKeys	要过滤的设备产品类型 productKey 列表，为 String 数组。此参数默认值为 null，此时 SDK 返回所有设备。若希望 SDK 只返回过滤后的设备，则参数应指定为需要的设备产品类型 productKey

	cloudServiceInfo	要切换的服务器域名信息。此参数默认值为 null，此时 SDK 将根据用户手机的地理位置信息为 App 设置机智云统一部署的云服务域名。
		若 App 希望使用独立部署的私有云服务域名，需按照以下字典{key: value}格式传值：
		{
		"openAPIInfo": "xxx", // String类型，api服务域名
		"siteInfo": "xxx" // String类型，site服务域名
		"pushInfo": "xxx" // String类型，推送服务域名
		}
		其中，openAPIInfo 和 siteInfo 必须传值，pushInfo 可选。
		
		可以不指定端口号，SDK 会使用默认的服务端口。此时形如：api.gizwits.com
		指定端口号时，需同时指定 Http 和 Https 端口。此时形如：xxx.gizwits.com:81&8443

	autoSetDeviceDomain	是否要开启设备域名的自动设置功能。此参数默认值为 false，即不开启自动设置。 参数值传 true，则开启设备域名的自动设置功能。如果开启了设备域名的自动设置，小循环设备将被连接到 App 当前使用的云服务域名上
回调	public void didNotifyEvent(GizEventType eventType, Object eventSource, GizWifiErrorCode eventID, String eventMessage)	
回调说明	当发生 GizEventType 中列举的事件类型时，SDK 会主动触发该回调，该回调通知的主要是发生的异常事件	
回调参数	eventType	事件类型。指明发生了哪一类的事件，详细见 GizEventType 枚举定义
	eventSource	事件源，指是谁触发的事件。如果 eventType 是 GizEventSDK，eventSource 为 null；如果是 GizEventDevice，eventSource 需要强制转换为 GizWifiDevice 类型再使用；如果是 GizEventM2Mservice 或者 GizEventToken，eventSource 需要强制转换为 String 类型再使用
	eventID	事件 ID。代表事件编号，详细见 GizWifiErrorCode 枚举定义。该参数指出 eventSource 发生了什么事
	eventMessage	事件 ID 的消息描述
代码示例	// 设置 SDK 监听	

```
GizWifiSDK.sharedInstance().setListener(mListener);
// 设置要过滤的设备 productKey 列表。不需要过滤则不用定义此变量直接传 null
List<String> specialProductKeys = new ArrayList<String> ();
specialProductKeys.add("your_product_key");
// 指定要切换的域名信息。使用机智云生产环境的 App 则不用定义此变量直接传 null
ConcurrentHashMap<String, Object> cloudServiceInfo = new
ConcurrentHashMap<String, Object>();
cloudServiceInfo.put("openAPIInfo", "your_api_domain");
cloudServiceInfo.put("siteInfo", "your_site_domain");
// 调用 SDK 的启动接口
GizWifiSDK.sharedInstance().startWithAppID("your_app_id",     context,
specialProductKeys, cloudServiceInfo, false);

// 实现系统事件通知回调
GizWifiSDKListener mListener = new GizWifiSDKListener() {
    @Override
    public void didNotifyEvent(GizEventType eventType, Object
eventSource, GizWifiErrorCode eventID, String eventMessage) {
        if (eventType == GizEventType.GizEventSDK) {
            // SDK发生异常的通知
            Log.i("GizWifiSDK", "SDK event happened: " + eventID + ", " +
eventMessage);
        } else if (eventType == GizEventType.GizEventDevice) {
            // 设备连接断开时可能产生的通知
            GizWifiDevice mDevice = (GizWifiDevice)eventSource;
            Log.i("GizWifiSDK", "device mac: " + mDevice.getMacAddress()
+ " disconnect caused by eventID: " + eventID + ", eventMessage: " +
eventMessage);
        } else if (eventType == GizEventType.GizEventM2MService) {
            // M2M服务返回的异常通知
            Log.i("GizWifiSDK", "M2M domain " + (String)eventSource + "
exception happened, eventID: " + eventID + ", eventMessage: " +
eventMessage);
        } else if (eventType == GizEventType.GizEventToken) {
            // token失效通知
            Log.i("GizWifiSDK","token "+(String)eventSource+" expired:
" + eventMessage);
        }
    }
};
```

【getCurrentCloudService】

定义	public void getCurrentCloudService()	
功能描述	查询当前使用的云服务域名信息	
回调	public void didGetCurrentCloudService(GizWifiErrorCode result, ConcurrentHashMap<String, String> cloudServiceInfo)	
回调说明	查询结果	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，cloudServiceInfo 为 null

	cloudServiceInfo	当前域名信息，字典{key: value}格式：
		{
		"openAPIDomain" : "xxx", // String类型
		"openAPIPort" : "xxx", // String类型
		"siteDomain" : "xxx", // String类型
		"sitePort" : "xxx", // String类型
		}

代码示例	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().getCurrentCloudService();
	
	// 实现回调
	public void didGetCurrentCloudService(GizWifiErrorCode result, ConcurrentHashMap<String, String> cloudServiceInfo) {
	if(result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
	// 成功
	} else {
	// 失败
	}
	}	

【getVersion】

定义	public String getVersion()
功能描述	获取 SDK 的版本号。
返回值	SDK 的版本号
代码示例	String sdkVersion = GizWifiSDK.sharedInstance().getVersion();

【setLogLevel】

定义	public void setLogLevel(GizLogPrintLevel logLevel)

功能描述	设置日志输出级别。该级别指日志在调试终端的输出级别，默认是全部输出的。 日志输出级别不影响日志文件的输出，无论日志输出级别设成什么，SDK 都会将运行日志写入文件。日志文件存放在 SD 卡目录下：GizWifiSDK/包名/GizSDKLog/	
参数	logLevel	日志输出级别，参考 GizLogPrintLevel 定义
代码示例	GizWifiSDK.sharedInstance().setLogLevel(GizLogPrintLevel.GizLogPrintAll);	

【disableLAN】

定义	public void disableLAN(boolean disabled)	
功能描述	设置是否禁用小循环功能	
参数	disabled	禁用或启用小循环
回调	public void didDisableLAN(GizWifiErrorCode result)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败

代码示例	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().disableLAN();
	
	// 实现回调
	GizWifiSDKListener mListener = new GizWifiSDKListener() {
	@Override
	public void didDisableLAN(GizWifiErrorCode result) {
	if (result == GizWifiErrorCode.*GIZ_SDK_SUCCESS*) {
	// 设置成功
	} else {
	// 设置失败
	}
	}
	};	

【getSSIDList】

定义	public void getSSIDList()	
功能描述	在 Soft-AP 模式时，获得设备的 SSID 列表。SSID 列表通过异步回调方式返回	
回调	public void didGetSSIDList(GizWifiErrorCode result, List<GizWifiSSID> ssidInfoList)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，ssidInfoList 为 null

	ssidInfoList	由 GizWifiSSID 实例组成的 SSID 信号列表

代码示例	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().getSSIDList();
	
	// 实现回调
	GizWifiSDKListener mListener = new GizWifiSDKListener() {
	@Override
	public void didGetSSIDList(GizWifiErrorCode result,
	List<GizWifiSSID> ssidInfoList) {
	if (result == GizWifiErrorCode.*GIZ_SDK_SUCCESS*) {
	// 获取成功
	} else {
	// 获取失败
	}
	}
	};	

【setDeviceOnboarding】

定义	public void setDeviceOnboarding(String ssid, String key, GizWifiConfigureMode mode, String softAPSSIDPrefix, int timeout, List<GizWifiGAgentType> types)	
功能描述	把设备配置到局域网 wifi 上。设备处于 softap 模式时，模组会产生一个热点名称，手机 wifi 连接此热点后就可以配置了。如果是机智云提供的固件，模组热点名称前缀为"XPG-GAgent-"，密码为"123456789"。设备处于 airlink 模式时，手机随时都可以开始配置。但无论哪种配置方式，设备上线时，手机要连接到配置的局域网 wifi 上，才能够确认设备已配置成功。 设备配置成功时，在回调中会返回设备 mac 地址。如果设备重置了，设备 did 可能要在设备搜索回调中才能获取。	
参数	ssid	要配置的路由 SSID
	key	要配置的路由密码
	mode	配置模式，详细见 GizWifiConfigureMode 枚举定义
	softAPSSIDPrefix	SoftAPMode 模式下 SoftAP 的 SSID 前缀或全名。默认前缀为：XPG-GAgent-，SDK 以此判断手机当前是否连上了设备的 SoftAP 热点。AirLink 配置时该参数无意义，传 null 即可
	timeout	配置的超时时间。SDK 默认执行的超时时间为 30 秒
	types	待配置的模组类型，是一个 GizWifiGAgentType 枚举数组。若不指定则默认配置乐鑫模组。GizWifiGAgentType 定义了 SDK 支持的所有模组类型。GizWifiGAgentType 还定义了一个 GizGAgentOther 枚举值，用于开发者使用自己的配置库进行设备配置

回调	public void didSetDeviceOnboarding(GizWifiErrorCode result, String mac, String did, String productKey)	
回调说明	注意：如果调用 getBoundDevices 接口时指定了待筛选的 productKey 集合，如果设备被成功配置到路由上了，会返回配置成功，但不会出现在设备列表中。	
回调参数	result	详细见 GizWifiErrorCode 枚举定义，GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，其他参数为 null
	mac	设备 mac 地址
	did	设备 did。配置成功时，did 的值可能为 null，因为设备刚配置完不一定会马上从云端申请到 did
	productKey	设备的产品类型标识

代码示例	// airlink 配置
	GizWifiSDK.sharedInstance().setListener(mListener);
	List<GizWifiGAgentType> types = new ArrayList();
	types.add(GizWifiGAgentType.*GizGAgentESP*);
	GizWifiSDK.sharedInstance().setDeviceOnboarding("your_ssid", "your_key", GizWifiConfigureMode.*GizWifiAirLink*, null, 60, types);
	
	// softap 配置
	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().setDeviceOnboarding("your_ssid", "your_key", GizWifiConfigureMode.*GizWifiSoftAP*, "XPG-GAgent-DF4A", 60, null);
	
	// 实现回调
	GizWifiSDKListener mListener = new GizWifiSDKListener() {
	@Override
	public void didSetDeviceOnboarding(GizWifiErrorCode result, String mac, String did, String productKey) {
	if (result == GizWifiErrorCode.*GIZ_SDK_SUCCESS*) {
	// 配置成功
	} else {
	// 配置失败
	}
	}
	};	

【getDevicesToSetServerInfo】

定义	public void getDevicesToSetServerInfo();
功能描述	获取可以设置域名的设备列表。该接口返回支持域名设置功能的设备信息列表，App 可以在给设备设置域名前，先调用该接口查看有哪些设备可以设置域名。

回调	public void didGetDevicesToSetServerInfo(GizWifiErrorCode result, List<ConcurrentHashMap<String, String>> devices);	
回调说明	该回调接口只返回设备的 mac、productKey、domain 这三个信息，不返回设备对象	
回调参数	result	获取成功或失败。如果获取失败，其他参数为 null

	devices	设备信息列表。设备信息字典格式如下：
		{
		“mac”: “xxx” // 设备 mac 地址
		“productKey”: “xxx” // 设备的 productKey
		“domain”: “xxx” // 设备的域名信息
		}

代码示例	GizWifiSDK.sharedInstance().setListener(mListener);
	
	// 获取可设置域名的设备列表
	GizWifiSDK.sharedInstance().getDevicesToSetServerInfo();
	
	// 实现回调
	public void didGetDevicesToSetServerInfo(GizWifiErrorCode result, List<ConcurrentHashMap<String, String>> devices) {
	if(result == GizWifiErrorCode.*GIZ_SDK_SUCCESS*) {
	// 获取成功
	} else {
	// 获取失败
	}
	}	

【setDeviceServerInfo】

定义	public void setDeviceServerInfo(String domain, String mac);	
功能描述	此接口为手动设置设备域名接口，可为设备设置对应的云服务域名。 设备和手机都连接到同一个 wifi 路由器后，可以设置设备要连接的云服务域名。可以设置当前已上线的所有小循环设备的域名。也可以单独设置某个设备的域名。如果不知道设备的 MAC 地址，可以先调用 getDevicesToSetServerInfo 接口查看有哪些设备可以设置域名，再调用该接口进行设置。 注意： 1、 只支持可设置域名的设备 2、 调用该接口将关闭已开启的设备域名自动设置功能	
参数	domain	待设置的域名。若该参数为 null，SDK 将根据用户手机的地理位置信息为设备设置机智云统一部署的云服务域名。 若要让设备连接独立部署的私有云域名，该参数为对应的私有云域名字

		符串，格式为：api.xxxxxx.com。这里需保证传入的域名是有效的，否则可能导致设备无法正常工作
	mac	待设置的设备 mac。默认参数为 null，即所有已发现的小循环设备都会被修改域名。如果只设置特定设备的域名，需指定 mac 地址
回调	public void didSetDeviceServerInfo(GizWifiErrorCode result, String mac);	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败
	mac	设置域名的设备 mac

代码示例	GizWifiSDK.sharedInstance().setListener(mListener);
	
	// 给设备设置域名
	GizWifiSDK.sharedInstance().setDeviceServerInfo(null, "your_device_mac");
	
	// 实现回调
	public void didSetDeviceServerInfo(GizWifiErrorCode result, String mac)
	{
	    if(result == GizWifiErrorCode.*GIZ_SDK_SUCCESS*) {
	        // 设置成功
	    } else {
	        // 设置失败
	    }
	}	

【getBoundDevices】

定义	public void getBoundDevices(String uid, String token, List<String> specialProductKey)	
功能描述	获取绑定设备列表。在不同的网络环境下，有不同的处理： 当手机能访问外网时，该接口会向云端发起获取绑定设备列表请求；当手机不能访问外网时，局域网设备是实时发现的，但会保留之前已经获取过的绑定设备；手机处于无网模式时，局域网未绑定设备会消失，但会保留之前已经获取过的绑定设备。用户未登录时，无法获取到绑定设备列表。 请注意：此接口传入的 uid、token，如果长度错误，SDK 会继续使用之前的 uid、token 作处理	
参数	uid	用户登录或注册时得到的 uid
	token	用户登录或注册时得到的 token
	specialProductKeys	指定要搜索的产品类型，为 String 数组。可以指定一个或多个产品类型，如果不指定则返回所有设备

回调	public void didDiscovered(GizWifiErrorCode result, List<GizWifiDevice> deviceList)	
回调说明	以下触发场景触发回调： getBoundDevices 接口调用时触发该回调，错误码代表云端请求状态，设备列表是绑定设备与局域网设备合并之后的集合； 设备列表发生变化时会主动上报时触发该回调，此时错误码 GIZ_SDK_SUCCESS，设备列表仍然是合并过的集合。	
回调参数	result	详细见 GizWifiErrorCode 枚举定义，GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，deviceList 为非 null 集合
	deviceList	GizWifiDevice 实例组成的数组，该参数将只返回根据指定 productKey 筛选过的设备集合。productKey 在 getBoundDevices 接口调用时指定

代码示例

```
GizWifiSDK.sharedInstance().setListener(mListener);
GizWifiSDK.sharedInstance().getBoundDevices("your_uid",
"your_token", null);

// 实现回调
GizWifiSDKListener mListener = new GizWifiSDKListener() {
    @Override
    public void didDiscovered(GizWifiErrorCode result,
    List<GizWifiDevice> deviceList) {
        // 提示错误原因
        if(result != GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            Log.d("", "result: " + result.name());
        }
        // 显示设备列表
        Log.d("", "discovered deviceList: " + deviceList);
    }
};
```

【bindRemoteDevice】

定义	public void bindRemoteDevice(String uid, String token, String mac, String productKey, String productSecret)	
功能描述	绑定远端设备到服务器	
参数	uid	用户登录或注册时得到的 uid
	token	用户登录或注册时得到的 token
	mac	待绑定设备的 mac
	productKey	待绑定设备的 productKey
	productSecret	待绑定设备的 productSecret

回调	public void didBindDevice(GizWifiErrorCode result, String did)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，did 为 null
	did	绑定成功的设备 did

示例	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().bindRemoteDevice ("your_uid", "your_token", "your_device_mac", "your_device_product_key", "your_product_secret");
	
	// 实现回调
	GizWifiSDKListener mListener = new GizWifiSDKListener() {
	@Override
	public void didBindDevice(GizWifiErrorCode result, String did) {
	if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
	// 绑定成功
	} else {
	// 绑定失败
	}
	}
	};	

【unbindDevice】

定义	public void unbindDevice(String uid, String token, String did)	
功能描述	把设备从服务器解绑	
参数	uid	用户登录或注册时得到的 uid
	token	用户登录或注册时得到的 token
	did	待解绑设备的 did
回调	public void didUnbindDevice(GizWifiErrorCode result, String did)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，did 为 null
	did	已解绑的设备 did

示例	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().unbindDevice("your_uid", "your_token", "your_device_did");
	
	// 实现回调
	GizWifiSDKListener mListener = new GizWifiSDKListener() {
	@Override	

```
    public void didUnbindDevice(GizWifiErrorCode result, String did) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 解绑成功
        } else {
            // 解绑失败
        }
    }
};
```

【getCaptchaCode】

定义	public void getCaptchaCode(String appSecret)	
功能描述	获取图片验证码。开发者登录 site.gizwits.com，在自己账户下的应用管理中可以得到 App Secret，通过应用的 App Secret 才能获取到图片验证码。	
参数	appSecret	应用的 secret 信息，从 site.gizwits.com 中可以看到
回调	public void didGetCaptchaCode(GizWifiErrorCode result, String token, String captchaId, String captchaURL)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，其他回调参数为 null
	token	图片验证码 token。图片验证码 token 在 1 小时后过期
	captchaId	图片验证码 id。图片验证码 5 分钟后过期
	captchaURL	图片验证码网址。图片验证码 url 在使用后过期

代码示例

```
GizWifiSDK.sharedInstance().setListener(mListener);
GizWifiSDK.sharedInstance().getCaptchaCode("your_app_secret");

// 实现回调
GizWifiSDKListener mListener = new GizWifiSDKListener() {
    @Override
    public void didGetCaptchaCode(GizWifiErrorCode result, String
    token, String captchaId, String captchaURL) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 获取成功
        } else {
            // 获取失败
        }
    }
};
```

【requestSendPhoneSMSCode】

定义	public void requestSendPhoneSMSCode(String appSecret, String phone)	
功能描述	通过手机号请求短信验证码	
参数	appSecret	应用的 secret 信息，从 site.gizwits.com 中可以看到
	phone	手机号
回调	public void didRequestSendPhoneSMSCode(GizWifiErrorCode result, String token)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，token 为 null
	token	请求短信验证码时得到的 token

代码示例	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().requestSendPhoneSMSCode
	("your_app_secret", "your_phone_number");
	
	// 实现回调
	GizWifiSDKListener mListener = new GizWifiSDKListener() {
	@Override
	public void didRequestSendPhoneSMSCode(GizWifiErrorCode result,
	String token) {
	if (result == GizWifiErrorCode.*GIZ_SDK_SUCCESS*) {
	// 请求成功
	} else {
	// 请求失败
	}
	}
	};	

【requestSendPhoneSMSCode】

定义	public void requestSendPhoneSMSCode(String token, String captchaId, String captchaCode, String phone)	
功能描述	通过图形验证码请求短信验证码	
参数	token	通过 getCaptchaCode 获取到的 token
	captchaId	通过 getCaptchaCode 获取到的 captchaId
	captchaCode	图片验证码的内容
	phone	手机号
回调	public void didRequestSendPhoneSMSCode(GizWifiErrorCode result, String token)	

回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，token 为 null
	token	请求短信验证码时得到的 token

代码示例	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().requestSendPhoneSMSCode ("your_token", "your_captchaId", "your_captchaCode", "your_phone_number");
	
	// 实现回调
	GizWifiSDKListener mListener = new GizWifiSDKListener() {
	@Override
	public void didRequestSendPhoneSMSCode(GizWifiErrorCode result,
	String token) {
	if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
	// 请求成功
	} else {
	// 请求失败
	}
	}
	};	

【verifyPhoneSMSCode】

定义	public void verifyPhoneSMSCode(String token, String phoneCode, String phone)	
功能描述	验证手机短信验证码。注意，验证短信验证码后，验证码就失效了，无法再用于手机号注册	
参数	token	验证码的 token，通过 getCaptchaCode 获取
	phoneCode	手机短信验证码
	phone	手机号码
回调	public void didVerifyPhoneSMSCode(GizWifiErrorCode result)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败

代码示例	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().verifyPhoneSMSCode ("your_token", "your_verify_code", "your_phone_number");
	
	// 实现回调
	GizWifiSDKListener mListener = new GizWifiSDKListener() {
	@Override
	public void didVerifyPhoneSMSCode (GizWifiErrorCode result) {
	if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {	

```
            // 验证成功
        } else {
            // 验证失败
        }
    }
};
```

【registerUser】

定义	public void registerUser(String username, String password, String code, GizUserAccountType accountType)	
功能描述	用户注册。需指定用户类型注册。手机用户的用户名是手机号，邮箱用户的用户名是邮箱、普通用户的用户名可以是普通用户名	
参数	username	注册用户名（可以是手机号、邮箱或普通用户名）
	password	注册密码
	code	手机短信验证码。短信验证码注册后就失效了，不能被再次使用
	accountType	用户类型，详细见 GizUserAccountType 枚举定义。注册手机号时，此参数指定为手机用户，注册邮箱时，此参数指定为邮箱用户，注册普通用户名时，此参数指定为普通用户
回调	public void didRegisterUser(GizWifiErrorCode result, String uid, String token)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，uid、token 为 null
	uid	注册成功后得到的 uid
	token	注册成功后得到的 token

代码示例

```
GizWifiSDK.sharedInstance().setListener(mListener);
GizWifiSDK.sharedInstance().registerUser("your_phone_number",
"your_password", "your_verify_code",
GizUserAccountType.GizUserPhone);

// 实现回调
GizWifiSDKListener mListener = new GizWifiSDKListener() {
    @Override
    public void didRegisterUser(GizWifiErrorCode result, String uid,
    String token) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 注册成功
        } else {
            // 注册失败
        }
```

```
    }
};
```

【userLoginAnonymous】

定义	public void userLoginAnonymous()	
功能描述	匿名登录。匿名方式登录，不需要注册用户账号。	
回调	public void didUserLogin(GizWifiErrorCode result, String uid, String token)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，uid、token 为 null
	uid	注册成功后得到的 uid
	token	注册成功后得到的 token

代码示例

```
GizWifiSDK.sharedInstance().setListener(mListener);
GizWifiSDK.sharedInstance().userLoginAnonymous();

// 实现回调
GizWifiSDKListener mListener = new GizWifiSDKListener() {
    @Override
    public void didUserLogin(GizWifiErrorCode result, String uid,
    String token) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 登录成功
        } else {
            // 登录失败
        }
    }
};
```

【userLogin】

定义	public void userLogin(String username, String password)	
功能描述	用户登录。需使用注册成功的用户名、密码进行登录，可以是手机用户名、邮箱用户名或普通用户名	
参数	username	注册成功的用户名
	password	注册成功的用户密码
回调	public void didUserLogin(GizWifiErrorCode result, String uid, String token)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其

		他为失败。失败时，uid、token 为 null
	uid	登录成功后得到的 uid
	token	登录成功后得到的 token

代码示例	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().userLogin("your_user_name", "your_password");
	
	// 实现回调
	GizWifiSDKListener mListener = new GizWifiSDKListener() {
	@Override
	public void didUserLogin(GizWifiErrorCode result, String uid, String token) {
	if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
	// 登录成功
	} else {
	// 登录失败
	}
	}
	};	

【userLoginWithThirdAccount】

定义	public void loginWithThirdAccount(GizThirdAccountType thirdAccountType, String uid, String token)	
功能描述	第三方账号登录（第三方接口登录方式）	
参数	thirdAccountType	第三方账号类型，详细见 GizThirdAccountType 枚举定义
	uid	通过第三方平台 api 方式登录后得到的 uid
	token	通过第三方平台 api 方式 登录后得到的 token
回调	public void didUserLogin(GizWifiErrorCode result, String uid, String token)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，uid、token 为 null
	uid	登录成功后得到的 uid
	token	登录成功后得到的 token
代码示例	GizWifiSDK.sharedInstance().setListener(mListener); GizWifiSDK.sharedInstance().loginWithThirdAccount(GizThirdAccountType.GizThirdSINA, "your_third_uid", "your_third_token"); // 实现回调	

```
GizWifiSDKListener mListener = new GizWifiSDKListener() {
    @Override
    public void didUserLogin(GizWifiErrorCode result, String uid,
    String token) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 登录成功
        } else {
            // 登录失败
        }
    }
};
```

【changeUserPassword】

定义	public void changeUserPassword(String token, String oldPassword, String newPassword)	
功能描述	修改用户密码	
参数	token	用户登录或注册时得到的 token
	oldPassword	旧密码
	newPassword	新密码
回调	public void didChangeUserPassword(GizWifiErrorCode result)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败

代码示例

```
GizWifiSDK.sharedInstance().setListener(mListener);
GizWifiSDK.sharedInstance().changeUserPassword("your_token",
"your_old_password", "your_new_password");

// 实现回调
GizWifiSDKListener mListener = new GizWifiSDKListener() {
    @Override
    public void didChangeUserPassword(GizWifiErrorCode result) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 修改成功
        } else {
            // 修改失败
        }
    }
};
```

【resetPassword】

定义	public void resetPassword(String username, String code, String newPassword, GizUserAccountType accountType)	
功能描述	重置密码。手机号重置密码时通过手机短信验证码重置，邮箱重置密码时需通过邮箱密码重置链接重置	
参数	username	待重置密码的手机号或邮箱
	code	重置手机用户密码时需要使用手机短信验证码（通过 requestSendPhoneSMSCode 方法获取）
	newPassword	新密码。邮箱重置密码时不需要填充密码，可指定为 null
	accountType	用户类型，详细见 GizThirdAccountType 枚举定义。待重置密码的用户名是手机号时，此参数指定为手机用户，待重置密码的用户名是邮箱时，此参数指定为邮箱用户
回调	public void didChangeUserPassword(GizWifiErrorCode result)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败
代码示例		

```
GizWifiSDK.sharedInstance().setListener(mListener);
GizWifiSDK.sharedInstance().resetPassword("your_phone_number",
"your_verify_code",                                    "your_new_password",
GizUserAccountType.GizUserPhone);

// 实现回调
GizWifiSDKListener mListener = new GizWifiSDKListener() {
    @Override
    public void didChangeUserPassword(GizWifiErrorCode result) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 修改成功
        } else {
            // 修改失败
        }
    }
};
```

【changeUserInfo】

定义	public void changeUserInfo(String token, String username, String code, GizUserAccountType accountType, GizUserInfo additionalInfo)
功能描述	修改用户信息，包括用户名和个人信息。用户名只支持修改手机号或邮箱，并且手机号或邮箱必须是已经注册过的。该接口用于以下场景：只修改手机号、只修改邮箱、只修改普通用户的个人信息、同时修改手机号和补充信息、同时修改邮箱和补充信息。只修改个人信息时，accountType

	可以指定为 GizUserNormal；修改手机号要指定为 GizUserPhone；修改邮箱要指定为 GizUserEmail	
参数	token	用户登录或注册时得到的 token
	username	待修改的手机号或邮箱
	code	修改手机号时要使用的手机短信验证码
	accountType	用户类型，详细见 GizThirdAccountType 枚举定义。修改手机号时，accountType 传 GizUserPhone；修改普通用户名时，accountType 传 GizUserEmail；只修改个人信息时，accountType 传 GizUserNormal；同时修改用户名和个人信息时，可根据待修改的是手机号还是邮箱来指定。
	additionalInfo	待修改的个人信息，详细见 GizUserInfo 类定义。如果只修改个人信息，需要指定 token，username、code 填 null
回调	public void didChangeUserInfo(GizWifiErrorCode result)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败

代码示例

```
GizWifiSDK.sharedInstance().setListener(mListener);
GizWifiSDK.sharedInstance().changeUserInfo("your_token",
"your_phone_number",                                    "your_verify_code",
GizUserAccountType.GizUserPhone, null);

// 实现回调
GizWifiSDKListener mListener = new GizWifiSDKListener() {
    @Override
    public void didChangeUserInfo(GizWifiErrorCode result) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 修改成功
        } else {
            // 修改失败
        }
    }
};
```

【getUserInfo】

定义	public void getUserInfo(String token)	
功能描述	获取用户信息	
参数	token	用户登录或注册时得到的 token
回调	public void didGetUserInfo(GizWifiErrorCode result, GizUserInfo userInfo)	

回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败
	userInfo	用户信息，详细见 GizUserInfo 类

代码示例	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().getUserInfo ("your_token");
	
	// 实现回调
	GizWifiSDKListener mListener = new GizWifiSDKListener() {
	@Override
	public void didGetUserInfo(GizWifiErrorCode result, GizUserInfo
	userInfo) {
	if (result == GizWifiErrorCode.*GIZ_SDK_SUCCESS*) {
	// 获取成功
	} else {
	// 获取失败
	}
	}
	};	

【transAnonymousUser】

定义	public void transAnonymousUser(String token, String username, String password, String code, GizUserAccountType accountType)	
功能描述	匿名用户转换，可转换为手机用户或者普通用户。注意，待转换的帐号必须是还未注册过的	
参数	token	用户登录或注册时得到的 token
	username	待转换的普通账号或手机号
	password	转换后的帐号密码
	code	转换为手机用户时要使用的手机短信验证码
	accountType	指定待转换的用户类型，详细见 GizThirdAccountType 枚举定义。待转换的用户名是手机号时，此参数指定为 GizUserPhone，待转换用户名是普通账号时，此参数指定为 GizUserNormal
回调	public void didTransAnonymousUser(GizWifiErrorCode result)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败

代码示例	// 匿名转手机用户
	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().transAnonymousUser("your_token", "your_phone_number", "your_password", "your_verify_code", GizUserAccountType. GizUserPhone);	

```
// 实现回调
GizWifiSDKListener mListener = new GizWifiSDKListener() {
    @Override
    public  void didTransAnonymousUser(GizWifiErrorCode result) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 转换成功
        } else {
            // 转换失败
        }
    }
};
```

【getGroups】

定义	public void getGroups(String uid, String token, List<String> specialProductKeys)	
功能描述	获取分组列表，可以获取之前创建过的组列表	
参数	uid	用户登录或注册时得到的 uid
	token	用户登录或注册时得到的 token
	specialProductKeys	待筛选的组类型标识， String 数组。不指定则不筛选
回调	public void didGetGroups(GizWifiErrorCode result, List<GizWifiGroup> groupList)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，groupList 为 null
	groupList	为 GizWifiGroup 实例组成的数组

代码示例	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().getGroups("your_uid", "your_token", null);
	
	// 实现回调
	GizWifiSDKListener mListener = new GizWifiSDKListener() {
	@Override
	public void didGetGroups(GizWifiErrorCode result, List<GizWifiGroup> groupList) {
	if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
	// 获取成功
	} else {
	// 获取失败
	}
	}	

	};

【addGroup】

定义	public void addGroup(String uid, String token, String productKey, String groupName, List<ConcurrentHashMap<String, String>> specialDevices)	
功能描述	添加分组。添加分组时，必须指定组类型，组类型应该是一个有效的设备产品类型标识。默认分配的组名称为“Default”，组设备亦可以在创建组之后再添加。但如果添加的组设备是 SDK 无法识别的，也无法被添加到分组中。组设备添加失败不会影响组的创建	
参数	uid	用户登录或注册时得到的 uid
	token	用户登录或注册时得到的 token
	productKey	指定组类型标识
	groupName	指定组名称
	specialDevices	指定加入组内的设备。字典数组，依赖键值 sdid（子设备标识码）、did（父设备标识码）。不加入设备则传 null
回调	public void didGetGroups(GizWifiErrorCode result, List<GizWifiGroup> groupList)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，groupList 为 null
	groupList	为 GizWifiGroup 实例组成的数组

代码示例

```
GizWifiSDK.sharedInstance().setListener(mListener);
GizWifiSDK.sharedInstance().getGroups("your_uid",       "your_token",
null);

// 实现回调
GizWifiSDKListener mListener = new GizWifiSDKListener() {
    @Override
    public          void     didGetGroups(GizWifiErrorCode     result,
    List<GizWifiGroup> groupList) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 添加成功
        } else {
            // 添加失败
        }
    }
};
```

【removeGroup】

定义	public void removeGroup(String uid, String token, String gid)	
功能描述	删除分组。以组 id 为唯一标识	
参数	uid	用户登录或注册时得到的 uid
	token	用户登录或注册时得到的 token
	gid	待删除的组 id
回调	public void didGetGroups(GizWifiErrorCode result, List<GizWifiGroup> groupList)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，groupList 为 null
	groupList	为 GizWifiGroup 实例组成的数组

代码示例	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().removeGroup("your_uid", "your_token", "your_gid");
	
	// 实现回调
	GizWifiSDKListener mListener = new GizWifiSDKListener() {
	@Override
	public void didGetGroups(GizWifiErrorCode result, List<GizWifiGroup> groupList) {
	if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
	// 删除成功
	} else {
	// 删除失败
	}
	}
	};	

【editGroup】

定义	public void editGroup(String uid, String token, String gid, String groupName, List<ConcurrentHashMap<String, String>> specialDevices)	
功能描述	编辑分组，可以修改组名称和组设备。但如果要编辑的组设备，与组类型不匹配时，或是 SDK 无法识别的，设备是不能更新到分组里的	
参数	uid	用户登录或注册时得到的 uid
	token	用户登录或注册时得到的 token
	gid	编辑组时需要指定组 id

	groupName	待修改的组名称
	specialDevices	要编辑的组内设备。字典数组，依赖键值 sdid（子设备标识码）、did（父设备标识码）。不指定设备则传 null
回调	public void didGetGroups(GizWifiErrorCode result, List<GizWifiGroup> groupList)	
回调参数	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，groupList 为 null
	groupList	为 GizWifiGroup 实例组成的数组

代码示例	GizWifiSDK.sharedInstance().setListener(mListener);
	GizWifiSDK.sharedInstance().editGroup("your_uid", "your_token", null);
	
	// 实现回调
	GizWifiSDKListener mListener = new GizWifiSDKListener() {
	@Override
	public void didGetGroups(GizWifiErrorCode result, List<GizWifiGroup> groupList) {
	if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
	// 编辑成功
	} else {
	// 编辑失败
	}
	}
	};	

2. GizWifiDevice 类

2.1. 简介

机智云 Wi-Fi 的设备类。GizWifiDevice 类为 APP 开发者提供设备订阅、设备数据通知、设备实时状态通知，例如热水器的水温等功能。该设备实例是通过 GizWifiDevice 类分配出来的，不能自行创建。

2.2. 属性方法

【setListener】

定义	public void setListener(GizWifiDeviceListener Listener)	
功能描述	设置设备的监听器	
参数	listener	设备监听器

示例	// mDevice 是从设备列表中获取到的设备实体对象
	mDevice.setListener(new GizWifiDeviceListener() {});

【getMacAddress】

定义	public String getMacAddress()
功能描述	获取设备的 Mac 地址。如果是 VIRTUAL:SITE，则是虚拟设备
返回值	返回设备的 Mac 地址
示例	// mDevice 是从设备列表中获取到的设备实体对象 String mac = mDevice.getMacAddress();

【getDid】

定义	public String getDid()
功能描述	设备云端身份标识 DID
返回值	返回设备的 did
示例	// mDevice 是从设备列表中获取到的设备实体对象 String did = mDevice.getDid();

【getIPAddress】

定义	public String getIpAddress()
功能描述	获取设备的 ip 地址。大循环设备的 ip 地址为云端服务器域名
返回值	返回设备的 ip 地址
示例	// mDevice 是从设备列表中获取到的设备实体对象 String ip = mDevice.getIPAddress();

【getProductKey】

定义	public String getProductKey()
功能描述	获取设备的产品类型识别码
返回值	返回设备的产品类型识别码
示例	// mDevice 是从设备列表中获取到的设备实体对象 String productKey = mDevice.getProductKey();

【getProductName】

定义	public String getProductName()

功能描述	获取设备的产品名称
返回值	返回设备的产品名称
示例	// mDevice 是从设备列表中获取到的设备实体对象 String productName = mDevice.getProductName();

【getProductType】

定义	public GizWifiDeviceType getProductType()
功能描述	获取设备分类是中控设备还是普通设备
返回值	返回设备分类
示例	// mDevice 是从设备列表中获取到的设备实体对象 GizWifiDeviceType type = mDevice.getProductType();

【getRemark】

定义	public String getRemark()
功能描述	获取设备的备注信息。设备绑定后可以修改，默认为空
返回值	返回设备的备注信息
示例	// mDevice 是从设备列表中获取到的设备实体对象 String remark = mDevice.getRemark();

【getAlias】

定义	public String getAlias()
功能描述	获取设备的别名。设备绑定后可以修改，默认为空
返回值	返回设备的别名
示例	// mDevice 是从设备列表中获取到的设备实体对象 String alias = mDevice.getAlias();

【getNetStatus】

定义	public GizWifiDeviceNetStatus getNetStatus()
功能描述	获取设备的网络状态，详见 GizWifiDeviceNetStatus 枚举定义
返回值	返回设备的网络状态
示例	// mDevice 是从设备列表中获取到的设备实体对象 GizWifiDeviceNetStatus netStatus = mDevice. getNetStatus ();

【isLAN】

定义	`public boolean isLAN()`
功能描述	判断设备是小循环还是大循环
返回值	返回设备是小循环还是大循环
示例	`// mDevice 是从设备列表中获取到的设备实体对象` `boolean isBind = mDevice.isLAN();`

【isBind】

定义	`public boolean isBind()`
功能描述	判断设备是否已绑定
返回值	返回设备是否已绑定
示例	`// mDevice 是从设备列表中获取到的设备实体对象` `boolean isBind = mDevice.isBind();`

【isDisabled】

定义	`public boolean isDisabled()`
功能描述	判断设备是否已在云端注销
返回值	返回设备是否已注销
示例	`// mDevice 是从设备列表中获取到的设备实体对象` `boolean isDisabled = mDevice.isDisabled();`

【isSubscribed】

定义	`public boolean isSubscribed()`
功能描述	判断设备是否已订阅
返回值	返回设备是否已订阅
示例	`// mDevice 是从设备列表中获取到的设备实体对象` `boolean isSubscribed = mDevice.isSubscribed();`

【isProductDefined】

定义	`public boolean isProductDefined()`
功能描述	判断设备是否定义了产品数据点
返回值	返回设备是否有数据点定义

示例	// mDevice 是从设备列表中获取到的设备实体对象 boolean isProductDefined = mDevice.isProductDefined();

2.3. 回调接口

以下是 GizWifiDevice 类提供的所有回调接口，将在在后续 API 定义中详细介绍：

- didGetHardwareInfo：设备硬件信息的回调
- didSetCustomInfo：设置设备绑定信息的回调
- didExitProductionTesting：设备退出产测的回调
- didSetSubscribe：设备订阅或解除订阅的回调
- didUpdateNetStatus：设备网络状态变化通知
- didReceiveData：接收到设备状态上报的回调

2.4. API

【didUpdateNetStatus】

回调	public void didUpdateNetStatus(GizWifiDevice device,GizWifiDeviceNetStatus netStatus)	
回调说明	该回调主动上报设备的网络状态变化，当设备重上电、断电或可控时会触发该回调	
回调参数	device	回调的 GizWifiDevice 对象
	netStatus	设备是离线、在线还是可控状态

代码示例	// mDevice是从设备列表中获取到的设备实体对象
	mDevice.setListener(deviceListener);
	
	// 实现回调
	GizWifiDeviceListener mListener = new GizWifiDeviceListener() {
	@override
	public void didUpdateNetStatus(GizWifiDevice device,
	GizWifiDeviceNetStatus netStatus) {
	}
	};	

【setSubscribe】

定义	public void setSubscribe(boolean subsribe)	
功能描述	设备订阅或解除订阅。订阅了设备，表示使用者关心这个设备的消息推送。解除订阅，表示使用者不关心这个设备的消息推送。订阅设备后，SDK 将自动登录和自动绑定设备。解除订阅后，设备连接将自动断开，但不会自动解绑。一般来说，设备订阅都会成功的，SDK 会记住设备是否被订阅了。	
参数	isSubscribed	订阅或取消订阅。true 表示订阅，false 表示取消订阅

回调	public void didSetSubscribe(GizWifiErrorCode result, GizWifiDevice device, boolean isSubscribed)	
回调参数	device	回调的 GizWifiDevice 对象
	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，订阅状态无变化
	isSubscribed	设备是被订阅了还是被解除订阅了。true 表示被订阅，false 表示被解除订阅

代码示例	// mDevice是从设备列表中获取到的设备实体对象
	mDevice.setSubscribe(true); // 订阅设备
	mDevice.setSubscribe(false); // 解除订阅
	
	// 实现回调
	GizWifiDeviceListener mListener = new GizWifiDeviceListener() {
	@Override
	public void didSetSubscribe(GizWifiErrorCode result, GizWifiDevice device, boolean isSubscribed) {
	if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
	// 订阅或解除订阅成功
	} else {
	// 失败
	}
	}
	};	

【getHardwareInfo】

定义	public void getHardwareInfo()	
功能描述	获取硬件信息。不订阅设备也可以使用此接口，只要设备连入正常工作模式即可	
回调	public void didGetHardwareInfo(GizWifiErrorCode result, GizWifiDevice device, ConcurrentHashMap<String, String> hardwareInfo)	
回调参数	device	返回硬件信息的设备对象
	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，hardwareInfo 为 null
	hardwareInfo	硬件信息。对应的硬件信息键值对有： { "wifiHardVersion": [value], // value为String 类型，设备的 Wifi 模组硬件版本号 "wifiSoftVersion": [value], // value为String 类型，设备的 Wifi 模组软件版本号

		"wifiFirmwareId": [value], // value为String 类型，设备的 Wifi 固件 ID "wifiFirmwareVer": [value], // value为String 类型，设备的 Wifi 固件版本 "mcuHardVersion": [value], // value为String 类型，设备的硬件版本号 "mcuSoftVersion": [value], // value为String 类型，设备的软件版本号 "productKey": [value], // value为String 类型，设备的产品唯一标识码 }

代码示例		// mDevice是从设备列表中获取到的设备实体对象
		mDevice.getHardwareInfo();
		
		// 实现回调
		GizWifiDeviceListener mListener = new GizWifiDeviceListener() {
		@Override
		public void didGetHardwareInfo(GizWifiErrorCode result,
		GizWifiDevice device, ConcurrentHashMap<String, String>
		hardwareInfo) {
		if (result == GizWifiErrorCode.*GIZ_SDK_SUCCESS*) {
		// 获取成功
		} else {
		// 获取失败
		}
		}
		};

【exitProductionTesting】

定义	public void exitProductionTesting()	
功能描述	让设备退出产测模式。不订阅设备就可以调用此接口，设备进入产测模式后会响应	
回调	public void didExitProductionTesting (GizWifiErrorCode result, GizWifiDevice device)	
回调参数	device	返回硬件信息的设备对象
	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败

代码示例	// mDevice是从设备列表中获取到的设备实体对象
	mDevice.exitProductionTesting();
	
	// 实现回调	

```
GizWifiDeviceListener mListener = new GizWifiDeviceListener() {
    @Override
    public  void didExitProductionTesting(GizWifiErrorCode result,
    GizWifiDevice device) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 执行成功
        } else {
            // 执行失败
        }
    }
};
```

【setCustomInfo】

定义	public void setCustomInfo(String remark, String alias)	
功能描述	修改设备的备注和别名。设备绑定后才能修改	
参数	remark	待修改的备注信息。传 null 表示不修改，传""则会覆盖为空串
	alias	待修改的设备别名。传 null 表示不修改，传""则会覆盖为空串
回调	public void didSetCustomInfo(GizWifiErrorCode result, GizWifiDevice device)	
回调参数	device	修改备注和别名的设备对象
	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败

代码示例

```
// mDevice是从设备列表中获取到的设备实体对象
mDevice.setCustomInfo("your_remark", "your_alias");

// 实现回调
GizWifiDeviceListener mListener = new GizWifiDeviceListener() {
    @Override
    public void didSetCustomInfo(GizWifiErrorCode result, GizWifiDevice
    device) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 修改成功
        } else {
            // 修改失败
        }
    }
};
```

【getDeviceStatus】

定义	public void getDeviceStatus()	
功能描述	获取设备状态。已订阅的设备变为可控状态后才能获取到状态	
回调	public void didReceiveData(GizWifiErrorCode result, GizWifiDevice device, ConcurrentHashMap<String, Object> dataMap, int sn)	
回调说明	设备回复或上报的数据中，当 SDK 遇到无法解析的数据时，会作为透传数据处理，此时错误码为 GIZ_SDK_SUCCESS	
回调参数	device	回复状态的设备对象
	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，dataMap 为空字典

	data	设备上报的数据内容，字典格式：
		{
		"data": [value], // value为ConcurrentHashMap类型，内容为设备状态键值对，[数据点标识名：数据点值]，数据点值的类型与 site 上的定义一致
		"alerts": [value], // value为ConcurrentHashMap类型，内容为设备报警键值对，[数据点标识名：数据点值]，数据点值的类型与 site 上的定义一致
		"faults": [value], // value为ConcurrentHashMap类型，内容为设备故障键值对，[数据点标识名：数据点值]，数据点值的类型与site上的定义一致
		"binary": [value], // value为Byte[]类型，内容为二进制数据，指没有在site上定义数据点的需要透传的数据
		}

	sn	控制指令的应答序号，此应答序号与 APP 发送控制指令的序号一致。设备主动上报数据和回复状态查询时，序号为0
代码示例		

```
// mDevice是从设备列表中获取到的设备实体对象
mDevice.getDeviceStatus();

// 实现回调
GizWifiDeviceListener mListener = new GizWifiDeviceListener() {
    @Override
    public void didReceiveData(GizWifiErrorCode result, GizWifiDevice
    device, ConcurrentHashMap<String, Object> dataMap, int sn) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 查询成功
        } else {
            // 查询失败
        }
    }
```

	};

【write】

定义	public void write(ConcurrentHashMap<String, Object> data, int sn)	
功能描述	给设备发送控制指令。已订阅的设备变为可控状态后才能发送控制指令	
参数	data	该参数为要发给设备的操作指令。为字典格式，字典键值对可按以下方式填充： • 如果设备有数据点定义，操作指令一次可以下发多个数据点。字典中的 key 为数据点名称，value 为数据点的值。value 类型要与数据点定义一致： （1）如果数据点为布尔类型，则 value 为 boolean 类型； （2）如果数据点为数值类型，则 value 为 int 或 float 类型； （3）如果数据点为枚举类型，则 value 为枚举序号（int 类型）或者枚举字符串 （String 类型）； （4）如果数据点为扩展类型，则 value 为 Byte[]类型； • 如果设备操作采用透传方式，透传指令一次只能下发一条。字典中的 key 填充为”binary”，value 为 Byte[]类型。
	sn	控制指令序号，用于对应控制指令应答数据。控制确认回调时会返回这个 sn
回调	public void didReceiveData(GizWifiErrorCode result, GizWifiDevice device, ConcurrentHashMap<String, Object> dataMap, int sn)	
回调说明	设备回复或上报的数据中，当 SDK 遇到无法解析的数据时，会作为透传数据处理，此时错误码为 GIZ_SDK_SUCCESS	
回调参数	device	回复状态的设备对象
	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，dataMap 为空字典

	data	设备上报的数据内容，字典格式：
		{
		"data": [value], // value为ConcurrentHashMap类型，内容为设备状态键值对，[数据点标识名：数据点值]，数据点值的类型与 site 上的定义一致
		"alerts": [value], // value为ConcurrentHashMap类型，内容为设备报警键值对，[数据点标识名：数据点值]，数据点值的类型与 site 上的定义一致
		"faults": [value], // value为ConcurrentHashMap类型，内容为设备故障键值对，[数据点标识名：数据点值]，数据点值的类型与site上的定义一致
		"binary": [value], // value为Byte[]类型，内容为二进制数据，指没有在site上定义数据点的需要透传的数据
		}

	sn	控制指令的应答序号，此应答序号与 APP 发送控制指令的序号一致。设备主动上报数据和回复状态查询时，序号为0

代码示例

```
// mDevice是从设备列表中获取到的设备实体对象，设置监听
mDevice.setListener(mListener);

/*
 * 此代码为使用sn的示例。如果App不使用sn，sn可设为0
 */
// 订阅设备并变为可控状态后，执行开灯动作
int sn = 5;
ConcurrentHashMap command = new ConcurrentHashMap<String, boolean> ();
command.put("LED_OnOff", true);
mDevice.write(command, sn);

// 实现回调
GizWifiDeviceListener mListener = new GizWifiDeviceListener() {
    @Override
    public  void didReceiveData(GizWifiErrorCode result, GizWifiDevice
    device, ConcurrentHashMap<String, Object> dataMap, int sn) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            if (sn == 5) {
                // 命令序号相符，开灯指令执行成功
            } else {
                // 其他命令的ack或者数据上报
            }
        } else {
            // 开灯失败
        }
    }
};
```

3. GizWifiCentralControlDevice 类

3.1. 简介

GizWifiCentralControlDevice 类为 APP 开发者提供中控子设备操作，包括获取子设备列表、添加子设备、删除子设备等功能。

该类继承自 GizWifiDevice 类，除下列属性和方法外，也具备 GizWifiDevice 类的所有属性和方法。

3.2. 属性

属性	描述
subDeviceList	List<GizWifiSubDevice>类型，提供 get 方法。子设备列表

3.3. 回调接口

以下是 GizWifiCentralControlDevice 类提供的所有回调接口，将在后续 API 定义中详细介绍：

- didDiscovered：中控子设备发现

3.4. API

【getSubDevices】

定义	public void getSubDevices()	
功能描述	获取子设备列表，只有中控设备可控后才能获取到。该接口会向中控设备发送获取子设备列表请求，中控设备将子设备列表通过回调返回	
回调	public void didDiscovered(GizWifiErrorCode result, GizWifiCentralControlDevice device, List<GizWifiSubDevice> subDeviceList)	
回调参数	device	触发回调的 GizWifiCentralControlDevice 对象
	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，subDeviceList 大小为 0
	subDeviceList	子设备列表。GizWifiSubDevice 对象数组

代码示例

```
// mDevice是从设备列表中获取到的中控设备实体对象
mDevice.getSubDevices();

// 实现回调
GizWifiDeviceListener mListener = new GizWifiDeviceListener() {
    @Override
    public void didDiscovered(GizWifiErrorCode result,
    GizWifiCentralControlDevice device, List<GizWifiSubDevice>
    subDeviceList) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 获取成功
        } else {
            // 获取失败
        }
    }
};
```

【addSubDevice】

定义	public void addSubDevice()
功能描述	添加子设备，只有中控设备可控后才能执行添加操作。该接口会向中控设备发送添加子设备请求，

	中控设备将添加后的子设备列表通过回调返回	
回调	public void didDiscovered(GizWifiErrorCode result, GizWifiCentralControlDevice device, List<GizWifiSubDevice> subDeviceList)	
回调参数	device	触发回调的 GizWifiCentralControlDevice 对象
	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，subDeviceList 大小为 0
	subDeviceList	子设备列表。GizWifiSubDevice 对象数组

代码示例	// mDevice是从设备列表中获取到的中控设备实体对象
	mDevice.addSubDevice();
	
	// 实现回调
	GizWifiDeviceListener mListener = new GizWifiDeviceListener() {
	@Override
	public void didDiscovered(GizWifiErrorCode result,
	GizWifiCentralControlDevice device, List<GizWifiSubDevice>
	subDeviceList) {
	if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
	// 添加成功
	} else {
	// 添加失败
	}
	}
	};	

【deleteSubDevice】

定义	public void deleteSubDevice(String subDid)	
功能描述	删除子设备，只有中控设备可控后才能执行删除操作。该接口会向中控设备发送删除子设备请求，中控设备将删除后的子设备列表通过回调返回	
参数	subDid	待删除的子设备 id
回调	public void didDiscovered(GizWifiErrorCode result, GizWifiCentralControlDevice device, List<GizWifiSubDevice> subDeviceList)	
回调参数	device	触发回调的 GizWifiCentralControlDevice 对象
	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，subDeviceList 大小为 0
	subDeviceList	子设备列表。GizWifiSubDevice 对象数组
代码示例	// mDevice是从设备列表中获取到的中控设备实体对象	

```
mDevice.deleteSubDevice("your_subDevice_id");

// 实现回调
GizWifiDeviceListener mListener = new GizWifiDeviceListener() {
    @Override
    public void didDiscovered(GizWifiErrorCode result,
    GizWifiCentralControlDevice device, List<GizWifiSubDevice>
    subDeviceList) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 删除成功
        } else {
            // 删除失败
        }
    }
};
```

4. GizWifiSubDevice

4.1. 简介

GizWifiSubDevice 类为 APP 开发者提供子设备的操作函数。子设备不需要订阅和解除订阅，只要中控设备可控，子设备就可以使用。

该类继承 GizWifiDevice 类，除以下列出的属性外，可以访问父类的相关接口和属性变量。但由于子设备具备一定的特殊性，子设备的属性和 API 的使用，需注意以下章节的说明。

4.2. 属性

子设备具备下面已列出的属性：

属性	描述
subDid	String 类型，提供 get 方法。子设备标识 DID
subProductKey	String 类型，提供 get 方法。子设备的产品类型标识码
subProductName	String 类型，提供 get 方法。子设备的产品类型名称
netStatus	boolean 类型，提供 get 方法。子设备的在线状态

4.3. API

不支持以下功能：获取硬件信息、退出产测功能、修改备注和别名。

支持以下 API：

【write】

定义	public String write()

功能描述	给子设备发送控制指令。 接口参数和回调说明，请参见 GizWifiDevice 类

【getDeviceStatus】

定义	`public String getDeviceStatus()`
功能描述	给子设备发送控制指令。 接口参数和回调说明，请参见 GizWifiDevice 类

5. GizWifiGroup 类

5.1. 简介

GizWifiGroup 类为 APP 开发者提供中控子设备的设备分组操作，包括获取分组内的设备、往分组内添加设备、删除分组内的设备等功能。

5.2. 属性方法

【setListener】

定义	`public void setListener(GizWifiGroupListener listener)`	
功能描述	设置设备分组的监听器	
参数	listener	设备分组监听器

代码示例	`// mDevice 是从设备列表中获取到的中控设备实体对象`
	`mDevice.setListener(new GizWifiGroupListener() {`
	`  // ……`
	`});`	

【getGid】

定义	`public String getGid()`
功能描述	获取组标识
返回值	组标识
代码示例	`// mGroup 是从设备分组列表中获取到的组实体对象` `String gid = mGroup.getGid();`

【getGroupName】

定义	`public String getGroupName()`

功能描述	获取组名称
返回值	组名称
代码示例	// mGroup 是从设备分组列表中获取到的组实体对象 String groupName = mGroup.getGroupName();

【getGroupType】

定义	public String getGroupType()
功能描述	获取组类型
返回值	组类型
代码示例	// mGroup 是从设备分组列表中获取到的组实体对象 String groupType = mGroup.getGroupType();

5.3. 回调接口

以下是 GizWifiGroup 类提供的所有回调接口，将在后续 API 定义中详细介绍：

- didGetDevices：分组设备列表回调

5.4. API

【getDevices】

定义	public void getDevices()	
功能描述	获取分组设备列表	
回调	public void didGetDevices(GizWifiErrorCode result, GizWifiGroup group, List<ConcurrentHashMap<String, String>> deviceList)	
回调参数	group	触发回调的设备分组对象
	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，deviceList 大小为 0

	deviceList	分组设备列表，用以下键值对唯一标识一个子设备：
		{
		"did": [value] //父设备标识码
		"sdid": [value], //子设备标识码
		}

代码示例	// mGroup是从分组列表中获取到的组实体对象
	mGroup.getDevices();
	
	// 实现回调
	GizWifiDeviceListener mListener = new GizWifiDeviceListener() {	

```
    @Override
    public void didGetDevices(GizWifiErrorCode result, GizWifiGroup
    group, List<ConcurrentHashMap<String, String>> deviceList) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 获取成功
        } else {
            // 获取失败
        }
    }
};
```

【addDevice】

定义	`public void addDevice(String did, String subdid)`	
功能描述	往设备分组内添加设备。设备的产品类型与组类型相同，才能添加到组里	
参数	did	父设备的设备标识
	subdid	子设备的设备标识
回调	`public void didGetDevices(GizWifiErrorCode result, GizWifiGroup group, List<ConcurrentHashMap<String, String>> deviceList)`	
回调参数	group	触发回调的设备分组对象
	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，deviceList 大小为 0

	deviceList	分组设备列表，用以下键值对唯一标识一个子设备：
		{
		"did": [value] //父设备标识码
		"sdid": [value], //子设备标识码
		}

代码示例

```
// mGroup是从分组列表中获取到的组实体对象
mGroup.addDevice("your_centralControlDevice_did",
"your_subdevice_did");

// 实现回调
GizWifiDeviceListener mListener = new GizWifiDeviceListener() {
    @Override
    public void didGetDevices(GizWifiErrorCode result, GizWifiGroup
    group, List<ConcurrentHashMap<String, String>> deviceList) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 添加成功
        } else {
            // 添加失败
```

```
        }
    }
};
```

【removeDevice】

定义	`public void removeDevice(String did, String subdid)`	
功能描述	从设备分组内删除设备	
参数	did	父设备的设备标识
	subdid	子设备的设备标识
回调	`public void didGetDevices(GizWifiErrorCode result, GizWifiGroup group, List<ConcurrentHashMap<String, String>> deviceList)`	
回调参数	group	触发回调的设备分组对象
	result	详细见 GizWifiErrorCode 枚举定义。GIZ_SDK_SUCCESS 表示成功，其他为失败。失败时，deviceList 大小为 0

	deviceList	分组设备列表，用以下键值对唯一标识一个子设备：
		{
		"did": [value] //父设备标识码
		"sdid": [value], //子设备标识码
		}

代码示例		

```
// mGroup是从分组列表中获取到的组实体对象
mGroup.removeDevice("your_centralControlDevice_did",
"your_subdevice_did");

// 实现回调
GizWifiDeviceListener mListener = new GizWifiDeviceListener() {
    @Override
    public void didGetDevices(GizWifiErrorCode result, GizWifiGroup
    group, List<ConcurrentHashMap<String, String>> deviceList) {
        if (result == GizWifiErrorCode.GIZ_SDK_SUCCESS) {
            // 删除成功
        } else {
            // 删除失败
        }
    }
};
```

6. GizUserInfo 类

6.1. 简介

GizUserInfo 类为开发者提供用户信息修改和获取。

6.2. 属性

属性	描述
uid	String 类型。用户登录后得到的 uid，提供 get 方法
username	String 类型。用户名：手机号或者邮箱，提供 get 方法
email	String 类型。用户邮箱，提供 get 方法
phone	String 类型。用户手机号，提供 get 方法
isAnonymous	boolean 类型。是否为匿名用户，提供 get 方法
lang	String 类型。用户的语言环境，提供 get 方法
name	String 类型。用户昵称，提供 get、set 方法
userGender	GizUserGenderType 类型。用户性别，提供 get、set 方法
birthday	String 类型。用户生日，提供 get、set 方法
address	String 类型。用户家庭住址，提供 get、set 方法
remark	String 类型。用户的备注信息，提供 get、set 方法

7. GizWifiSSID 类

7.1. 简介

路由的 SSID 信息类，包括 Wifi 信号名称 SSID 和信号强度。

7.2. 属性

属性	描述
ssid	SSID 名。我们连接一个 Wi-Fi 热点时，可以搜索到的名字
rssi	热点对应的信号强度。取值范围 0-100

【getSsid】

定义	public String getSsid()
功能描述	获取 wifi 的 SSID 名称。我们连接一个 Wi-Fi 热点时，可以搜索到的名字
返回值	Wifi 的 ssid 名称

代码示例	// mWifiSSID 是 SDK 提供的热点列表中的 ssid 实体对象 String ssid = mWifiSSID.getSsid();

【getRssi】

定义	public int getRssi()
功能描述	热点对应的信号强度。取值范围 0-100
返回值	Wifi 的 ssid 名称
代码示例	// mWifiSSID 是 SDK 提供的热点列表中的 ssid 实体对象 int rssi = mWifiSSID.getRssi();

8. 枚举类定义

8.1. 简介

本节说明 GizWifiSDK 中使用的所有枚举定义。

8.2. 定义

【GizLogPrintLevel】

功能描述：日志打印级别。

枚举 ID	枚举定义	描述
0	GizLogPrintNone	不输出任何日志
1	GizLogPrintI	输出错误日志
2	GizLogPrintII	输出调试日志
3	GizLogPrintAll	输出数据日志

【GizEventType】

功能描述：事件通知类型。

枚举 ID	枚举定义	描述
0	GizEventSDK	SDK 系统事件
1	GizEventDevice	设备异常事件
2	GizEventM2MService	M2M 异常事件
5	GizEventToken	Token 失效事件

【GizWifiConfigureMode】

功能描述：设备配置模式。

枚举 ID	枚举定义	描述
0	GizWifiSoftAP	SoftAP 配置模式
1	GizWifiAirLink	AirLink 配置模式

【GizWifiDeviceType】

功能描述：设备类型。

枚举 ID	枚举定义	描述
0	GizDeviceNormal	普通设备
1	GizDeviceCenterControl	中控设备

【GizThirdAccountType】

功能描述：第三方账号类型。

枚举 ID	枚举定义	描述
0	GizThirdBAIDU	百度账号
1	GizThirdSINA	新浪账号
2	GizThirdQQ	QQ 账号

【GizUserAccountType】

功能描述：机智云用户类型。

枚举 ID	枚举定义	描述
0	GizUserNormal	普通用户
1	GizUserPhone	手机用户
2	GizUserEmail	电子邮箱用户

【GizWifiDeviceNetStatus】

功能描述：设备网络状态类型。

枚举 ID	枚举定义	描述
0	GizDeviceOffline	离线状态
1	GizDeviceOnline	在线状态
2	GizDeviceControlled	可控状态

【GizWifiGAgentType】

功能描述：模组类型。

枚举 ID	枚举定义	描述
0	GizGAgentMXCHIP	庆科 3162 模组
1	GizGAgentHF	汉枫模组
2	GizGAgentRTK	睿昱模组
3	GizGAgentWM	联盛德模组
4	GizGAgentESP	乐鑫模组
5	GizGAgentQCA	高通模组
6	GizGAgentTI	TI 模组
7	GizGAgentFSK	语音天下模组
8	GizGAgentMXCHIP3	庆科 V3 配置
9	GizGAgentBL	古北模组
10	GizGAgentAtmelEE	Atmel 模组
11	GizGAgentOther	其他模组

【GizUserGenderType】

功能描述：用户性别。

枚举 ID	枚举定义	描述
0	GizUserGenderMale	男
1	GizUserGenderFemale	女
2	GizUserGenderUnknown	其他

【GizWifErrorCode】

功能描述：错误码定义。

枚举ID	枚举定义	描述
0	GIZ_SDK_SUCCESS	Client 发出的请求执行成功
8001	GIZ_SDK_PARAM_FORM_INVALID	Client 发给 Daemon 的 json 格式错误
8002	GIZ_SDK_CLIENT_NOT_AUTHEN	Client 与 Daemon 之间如果没有通过握手认证，任何数据交互都无效
8003	GIZ_SDK_CLIENT_VERSION_INVALID	Client 版本号无效
8004	GIZ_SDK_UDP_PORT_BIND_FAILED	udp 端口绑定失败

枚举 ID	枚举定义	描述
8005	GIZ_SDK_DAEMON_EXCEPTION	Daemon 系统错误
8006	GIZ_SDK_PARAM_INVALID	Client 发出的数据请求，Json 格式正确，但参数无效；APP 传入参数无效
8007	GIZ_SDK_APPID_LENGTH_ERROR	appid 长度错误
8008	GIZ_SDK_LOG_PATH_INVALID	日志路径无效
8009	GIZ_SDK_LOG_LEVEL_INVALID	日志级别无效
8021	GIZ_SDK_DEVICE_CONFIG_SEND_FAILED	设备配置信息发送失败
8022	GIZ_SDK_DEVICE_CONFIG_IS_RUNNING	设备正在配置
8023	GIZ_SDK_DEVICE_CONFIG_TIMEOUT	设备配置超时
8024	GIZ_SDK_DEVICE_DID_INVALID	设备 did 无效
8025	GIZ_SDK_DEVICE_MAC_INVALID	设备 mac 无效
8026	GIZ_SDK_SUBDEVICE_DID_INVALID	子设备 did 无效
8027	GIZ_SDK_DEVICE_PASSCODE_INVALID	设备 passcode 无效
8028	GIZ_SDK_DEVICE_NOT_CENTERCONTROL	不是中控设备
8029	GIZ_SDK_DEVICE_NOT_SUBSCRIBED	设备未订阅
8030	GIZ_SDK_DEVICE_NO_RESPONSE	设备未响应
8031	GIZ_SDK_DEVICE_NOT_READY	设备未就绪
8032	GIZ_SDK_DEVICE_NOT_BINDED	设备未绑定
8033	GIZ_SDK_DEVICE_CONTROL_WITH_INVALID_COMMAND	设备控制指令中包含无效指令
8034	GIZ_SDK_DEVICE_CONTROL_FAILED	设备控制指令执行失败
8035	GIZ_SDK_DEVICE_GET_STATUS_FAILED	设备状态查询失败
8036	GIZ_SDK_DEVICE_CONTROL_VALUE_TYPE_ERROR	设备控制指令参数类型错误
8037	GIZ_SDK_DEVICE_CONTROL_VALUE_OUT_OF_RANGE	设备控制指令参数值不在有效范围内
8038	GIZ_SDK_DEVICE_CONTROL_NOT_WRITABLE_COMMAND	设备控制指令中包含不可写指令
8039	GIZ_SDK_BIND_DEVICE_FAILED	设备绑定失败
8040	GIZ_SDK_UNBIND_DEVICE_FAILED	设备解绑失败
8041	GIZ_SDK_DNS_FAILED	域名解析失败
8042	GIZ_SDK_M2M_CONNECTION_SUCCESS	m2m 连接成功
8043	GIZ_SDK_SET_SOCKET_NON_BLOCK_FAILED	socket 设置非阻塞失败
8044	GIZ_SDK_CONNECTION_TIMEOUT	连接超时
8045	GIZ_SDK_CONNECTION_REFUSED	连接被拒绝
8046	GIZ_SDK_CONNECTION_ERROR	连接错误

枚举ID	枚举定义	描述
8047	GIZ_SDK_CONNECTION_CLOSED	连接被关闭
8048	GIZ_SDK_SSL_HANDSHAKE_FAILED	ssl 握手失败
8049	GIZ_SDK_DEVICE_LOGIN_VERIFY_FAILED	设备登录验证失败
8050	GIZ_SDK_INTERNET_NOT_REACHABLE	当前外网不可达
8096	GIZ_SDK_HTTP_ANSWER_FORMAT_ERROR	openapi 应答格式错
8097	GIZ_SDK_HTTP_ANSWER_PARAM_ERROR	http 应答参数错误
8098	GIZ_SDK_HTTP_SERVER_NO_ANSWER	http 服务无响应
8099	GIZ_SDK_HTTP_REQUEST_FAILED	http 请求失败，比如返回 404 等
8100	GIZ_SDK_OTHERWISE	其他错误
8101	GIZ_SDK_MEMORY_MALLOC_FAILED	Daemon 内存分配失败
8102	GIZ_SDK_THREAD_CREATE_FAILED	Daemon 内部线程创建失败
8150	GIZ_SDK_USER_ID_INVALID	用户 ID 无效
8151	GIZ_SDK_TOKEN_INVALID	用户 token 无效
8152	GIZ_SDK_GROUP_ID_INVALID	组 ID 无效
8153	GIZ_SDK_GROUPNAME_INVALID	组名称无效
8154	GIZ_SDK_GROUP_PRODUCTKEY_INVALID	组类型无效
8155	GIZ_SDK_GROUP_FAILED_DELETE_DEVICE	组设备删除失败
8156	GIZ_SDK_GROUP_FAILED_ADD_DEVICE	组设备添加失败
8157	GIZ_SDK_GROUP_GET_DEVICE_FAILED	组设备获取失败
8201	GIZ_SDK_DATAPOINT_NOT_DOWNLOAD	配置文件还未下载
8202	GIZ_SDK_DATAPOINT_SERVICE_UNAVAILABLE	配置文件服务不可用
8203	GIZ_SDK_DATAPOINT_PARSE_FAILED	配置文件解析失败
8300	GIZ_SDK_SDK_NOT_INITIALIZED	SDK 未初始化
8301	GIZ_SDK_APK_CONTEXT_IS_NULL	Context 无效，无法启动
8302	GIZ_SDK_APK_PERMISSION_NOT_SET	app 权限不足
8303	GIZ_SDK_CHMOD_DAEMON_REFUSED	无法修改 daemon 的执行权限
8304	GIZ_SDK_EXEC_DAEMON_FAILED	daemon 程序执行失败
8305	GIZ_SDK_EXEC_CATCH_EXCEPTION	尝试运行 daemon 时发生异常
8306	GIZ_SDK_APPID_IS_EMPTY	APPID 为空
8307	GIZ_SDK_UNSUPPORTED_API	不支持的 API
8308	GIZ_SDK_REQUEST_TIMEOUT	Client 如果等不到 Daemon 的回复，就向 APP 返回操作超时

枚举ID	枚举定义	描述
8309	GIZ_SDK_DAEMON_VERSION_INVALID	Daemon 版本号无效
8310	GIZ_SDK_PHONE_NOT_CONNECT_TO_SOFTAP_SSID	手机没有连接软 AP 热点
8311	GIZ_SDK_DEVICE_CONFIG_SSID_NOT_MATCHED	手机热点和要配置的路由 ssid 不匹配
8312	GIZ_SDK_NOT_IN_SOFTAPMODE	设备不在 softap 模式
8313	GIZ_SDK_CONFIG_NO_AVAILABLE_WIFI	设备配置时无可用 wifi
8314	GIZ_SDK_RAW_DATA_TRANSMIT	设备上报透传数据的标识
8315	GIZ_SDK_PRODUCT_IS_DOWNLOADING	正在下载设备的产品定义
8316	GIZ_SDK_START_SUCCESS	SDK 启动成功
9001	GIZ_OPENAPI_MAC_ALREADY_REGISTERED	mac already registered!
9002	GIZ_OPENAPI_PRODUCT_KEY_INVALID	product_key invalid
9003	GIZ_OPENAPI_APPID_INVALID	appid invalid
9004	GIZ_OPENAPI_TOKEN_INVALID	token invalid
9005	GIZ_OPENAPI_USER_NOT_EXIST	user not exist
9006	GIZ_OPENAPI_TOKEN_EXPIRED	token expired
9007	GIZ_OPENAPI_M2M_ID_INVALID	m2m_id invalid
9008	GIZ_OPENAPI_SERVER_ERROR	server error
9009	GIZ_OPENAPI_CODE_EXPIRED	code expired
9010	GIZ_OPENAPI_CODE_INVALID	code invalid
9011	GIZ_OPENAPI_SANDBOX_SCALE_QUOTA_EXHAUSTED	sandbox scale quota exhausted!
9012	GIZ_OPENAPI_PRODUCTION_SCALE_QUOTA_EXHAUSTED	production scale quota exhausted!
9013	GIZ_OPENAPI_PRODUCT_HAS_NO_REQUEST_SCALE	product has no request scale!
9014	GIZ_OPENAPI_DEVICE_NOT_FOUND	device not found!
9015	GIZ_OPENAPI_FORM_INVALID	form invalid!
9016	GIZ_OPENAPI_DID_PASSCODE_INVALID	did or passcode invalid!
9017	GIZ_OPENAPI_DEVICE_NOT_BOUND	device not bound!
9018	GIZ_OPENAPI_PHONE_UNAVALIABLE	phone unavailable!
9019	GIZ_OPENAPI_USERNAME_UNAVALIABLE	username unavailable!
9020	GIZ_OPENAPI_USERNAME_PASSWORD_ERROR	username or password error!
9021	GIZ_OPENAPI_SEND_COMMAND_FAILED	send command failed!
9022	GIZ_OPENAPI_EMAIL_UNAVALIABLE	email unavailable!
9023	GIZ_OPENAPI_DEVICE_DISABLED	device is disabled!
9024	GIZ_OPENAPI_FAILED_NOTIFY_M2M	fail to notify m2m!

枚举ID	枚举定义	描述
9025	GIZ_OPENAPI_ATTR_INVALID	attr invalid!
9026	GIZ_OPENAPI_USER_INVALID	user invalid!
9027	GIZ_OPENAPI_FIRMWARE_NOT_FOUND	firmware not found!
9028	GIZ_OPENAPI_JD_PRODUCT_NOT_FOUND	JD product info not found!
9029	GIZ_OPENAPI_DATAPOINT_DATA_NOT_FOUND	datapoint data not found!
9030	GIZ_OPENAPI_SCHEDULER_NOT_FOUND	scheduler not found!
9031	GIZ_OPENAPI_QQ_OAUTH_KEY_INVALID	qq oauth key invalid!
9032	GIZ_OPENAPI_OTA_SERVICE_OK_BUT_IN_IDLE	ota upgrade service OK, but in idle or disable!
9033	GIZ_OPENAPI_BT_FIRMWARE_UNVERIFIED	bt firmware unverified, except verify device!
9034	GIZ_OPENAPI_BT_FIRMWARE_NOTHING_TO_UPGRADE	bt firmware is OK, but nothing to upgrade!
9035	GIZ_OPENAPI_SAVE_KAIROSDB_ERROR	Save kairosdb error!
9036	GIZ_OPENAPI_EVENT_NOT_DEFINED	event not defined!
9037	GIZ_OPENAPI_SEND_SMS_FAILED	send sms failed!
9038	GIZ_OPENAPI_APPLICATION_AUTH_INVALID	X-Gizwits-Application-Auth invalid!
9039	GIZ_OPENAPI_NOT_ALLOWED_CALL_API	Not allowed to call deprecated API!
9040	GIZ_OPENAPI_BAD_QRCODE_CONTENT	bad qrcode content!
9041	GIZ_OPENAPI_REQUEST_THROTTLED	request was throttled
9042	GIZ_OPENAPI_DEVICE_OFFLINE	device offline!
9043	GIZ_OPENAPI_TIMESTAMP_INVALID	'X-Gizwits-Timestamp invalid!
9044	GIZ_OPENAPI_SIGNATURE_INVALID	X-Gizwits-Signature invalid!
9045	GIZ_OPENAPI_DEPRECATED_API	API deprecated!
9999	GIZ_OPENAPI_RESERVED	reserved