GIZWIFI IOS SDK 2.0 API REFERENCE MANUAL

1. GizWifiSDK Class

1.1. Introduction

Gizwits Wi-Fi base class, which provides base APIs of SDK initialization, user management, device management and configuration. It inherites from NSObject, and follows rule of Cocoa. If the symbol before method is “+”, it is static fucntion. If the symbol is “-”, it is instance function.

1.2. Member Functions

Member Function	Description
delegate	Using delegate to get corresponding event. The corresponding callback API of GizWifiSDK is defined on GizWifiSDK delegate, callback the API which you want to use.
deviceList	NSArray type, consists of GizWifiDevice objects, the cache of device list. APP would get the device list by accessing this variable

1.3. Callback API

Here are all the callback API of GizWifiSDK, can see details on following API Definition:
• didNotifyEvent: SDK system event nofitication callback
• didGetCurrentCloudService: stand-alone deployment cloud service domain call back
• didDiscovered: device list change notification callback
• didGetSSIDList: Wi-Fi list of device around callback
• didSetDeviceOnboarding: device Wi-Fi configuration callback
• didBindDevice: device binding callback
• didUnbindDevice: device unbinding callback
• didUpdateProduct: device definition change notification callback
• didGetCaptchaCode: image captcha callback
• didRequestSendPhoneSMSCode: SMS verify-code callback

- didRegisterUser: user registration callback
- didUserLogin: user login callback
- didTransAnonymousUser: anonymous user conversion callback
- didChangeUserPassword: user password change callback
- didChangeUserInfo: user info change callback
- didGetUserInfo: user info callback

1.4. API Definition

【sharedInstance】

Definition	+ (instancetype)sharedInstance;
Description	Get single instance of GizWifiSDK
Returns	The single instance of SDK. if SDK is uninitialized or initialization failed, it would return nil.
Sample code	GizWifiSDK mGizWifiSDKInstance = [GizWifiSDK sharedInstance];

【startWithAppID】

Definition	+ (void)startWithAppID:(NSString *)appID appSecret:(NSString*)appSecret specialProductKeys:(NSArray *)specialProductKeys cloudServiceInfo:(NSDictionary *)cloudSeviceInfo autoSetDeviceDomain:(BOOL)autoSetDeviceDomain;	
Description	This uses to initialize SDK. Only after this API is executed, can other APIs do. If delegate has been set, SDK will report discoverable devices immediately by didDiscovered callback. If App want to switch cloud service domain and filter devices by productKey, it should specify domain and productKey while initializing SDK. If you want to set domain of device, can enable auto-setting when this API is called, SDK would let all devices which support setting domain and App connect to the same cloud service domain, but auto-setting is disabled by default. Note: If auto-setting is enable, it will effective and remain in force, you can call setDeviceServerInfo API to stop auto-setting.	
Parameters	appid	On Gizwits Developer Zone dev.gizwits.com, each registered device can find its appID on its corresponding “application setting”. This parameter appID doesn’t have default value, developer must send correct value.

	appSecret	On Gizwits Developer Zone dev.gizwits.com, can find appSecret corresponding to appID in “application setting”. This parameter appSecret doesn’t have default value, developer must pass correct value.
	specialProductKeys	This parameter is device productKeys which you want to filter, it is NSString array. The default value of specialProductKeys is nil. When it is default value, and then SDK would return all devices. If you want SDK return devices which have been filtered, specialProductKeys should be special productKey.

	cloudServiceInfo	This parameter is the domain info of service which you want to connect.
		The default value is nil. When it is default value, SDK would base on location of mobile to set service domain as Gizwits general cloud service domain.
		If App want to set cloud service domain as stand-alone deployment, it should pass value as the following format Dictionary{key: value}
		{
		"openAPIInfo": "xxx", // NSString, cloud API service
		"siteInfo": "xxx" // NSString, site service domain
		"pushInfo": "xxx" // NSString, push service domain
		}
		App must pass value of openAPIInfo and siteInfo, but pushInfo is optional. App can pass value without specifying port number such as api.gizwits.com. If App want to specify port number, it should specify port number of http and https at the same time, like xxx.gizwits.com:81&8443.

	autoSetDeviceDomain	This parameter is auto-setting of device domain, the default value is NO. If App pass value is YES, it would enable auto-setting of device domain, device in LAN

		would connect to cloud service domain which is using by App
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didNotifyEvent:(GizEventType)eventType eventSource:(id)eventSource eventID:(GizWifiErrorCode)eventID eventMessage:(NSString *)eventMessage;	
Callback description	When the events enumerating in GizEventType happen, SDK would trigger this callback, it would notify some exception event, and the event 8316 which is SDK start successful.	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	eventType	Event type. It points out which event happens, can see details on GizEventType enum definition
	eventSource	Event source, it points out who triggers event. If eventType is GizEventSDK, eventSource is niil. If eventType is GizEventDevice, eventSource must be cast to GizWifiDevice. If eventType is GizEventM2Mservice or GizEventToken, eventSource must be cast to NSString.
	eventID	Event ID, it points out what happens to eventSource, can see details on GizWifiErrorCode enum definition
	eventMessage	the descriptipn for eventID

Sample code	// Set produceKey list which you want to filter. No need to define this variable if you don't want to filter, it would pass nil by default
	
	NSArray *specialProductKeys = [NSArray arrayWithObjects: @"your_product_key", nil];
	// Specify domain info which you want to connect. No need to define this variable while using Gizwits general cloud service domain, it would pass nil by default
	NSDictionary* cloudServiceInfo = @{@"openAPIInfo": @"your_api_domain", @"siteInfo": @"your_site_domain"};
	// call SDK start API
	[GizWifiSDK startWithAppID:@"your_appid" appSecret:@"your_app_secret" specialProductKeys:specialProductKeys cloudServiceInfo:cloudServiceInfo autoSetDeviceDomain:NO];
	// Callback implement callback of system event notification
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK	

	didNotifyEvent:(GizEventType)eventType eventSource:(id)eventSource eventID:(GizWifiErrorCode)eventID eventMessage:(NSString *)eventMessage {
	if(eventType == GizEventSDK) {
	// notification while SDK gets exception
	NSLog(@"SDK event happened: [%@] = %@", @(eventID), eventMessage);
	} else if(eventType == GizEventDevice) {
	// possible notification when device disconnect
	GizWifiDevice* mDevice = (GizWifiDevice*)eventSource;
	NSLog(@"device mac %@ disconnect caused by %@", mDevice.macAddress, eventMessage);
	} else if(eventType == GizEventM2MService) {
	// exception notification returns from M2M service
	NSLog(@"M2M domain %@ exception happened: [%@] = %@", (NSString*)eventSource, @(eventID), eventMessage);
	} else if(eventType == GizEventToken) {
	// notification when token is invalid
	NSLog(@"token %@ expired: %@", (NSString*)eventSource, eventMessage);
	}
	}

【getCurrentCloudService】

Definition	+ (void) getCurrentCloudService	
Description	Get current cloud service domain info	
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didGetCurrentCloudService:(NSError *)result cloudServiceInfo:(NSDictionary *)cloudServiceInfo;	
Callback description	Query result	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition, GIZ_SDK_SUCCESS is success, others are failures. When result is failure, cloudServiceInfo would be nil.
	cloudServiceInfo	Current cloud service domain info, following is format for dictionary{key: value}: { "openAPIDomain" : "xxx", // NSString

		"openAPIPort" : xxx, // int "siteDomain" : "xxx", // NSString "sitePort" : xxx, // int }

Sample cope	[GizWifiSDK sharedInstance].delegate = self;
	[GizWifiSDK getCurrentCloudService];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK
	didGetCurrentCloudService:(NSError *)result
	cloudServiceInfo:(NSDictionary *)cloudServiceInfo {
	if(result.code == GIZ_SDK_SUCCESS) {
	// success
	} else {
	// failure
	}
	}	

【getVersion】

Definition	+ (NSString *)getVersion;
Description	get SDK version
Returns	SDK version
Sample code	[[GizWifiSDK sharedInstance] getVersion];

【setLogLevel】

Definition	+ (void)setLogLevel:(GizLogPrintLevel)logPrintLevel;	
Description	Sets printing log level. This is the printing log level when debug in terminal, it would print all log by default. Printing log level would not affect log file print, SDK would write all log into log file no matter what printing log level is. Log file is stored in directory of Documents: GizWifiSDK/GizSDKLog/	
Parameters	logLevel	Printing log level, see details on GizLogPrintLevel definition
Sample code	[[GizWifiSDK sharedInstance] setLogLevel: GizLogPrintAll];	

【getSSIDList】

Definition	- (void)getSSIDList;
Description	Get SSID list of device in Soft-AP mode, SSID list would return from

	asynchronous callback.	
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didGetSSIDList:(NSError *)result ssidList:(NSArray *)ssidList;	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition, GIZ_SDK_SUCCESS is success, others are failures. When result is failure, ssidList would be nil.
	ssidList	SSID list consist of GizWifiSSID object

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] getSSIDList];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didGetSSIDList:(NSError *)result ssidList:(NSArray *)ssidList {
	if(result.code == GIZ_SDK_SUCCESS) {
	// get list success
	} else {
	// get list failure
	}
	}	

【setDeviceOnboarding】

Definition	- (void)setDeviceOnboarding:(NSString *)ssid key:(NSString *)key configMode:(GizWifiConfigureMode)mode softAPSSIDPrefix:(NSString *)softAPSSIDPrefix timeout:(int)timeout wifiGAgentType:(NSArray *)types;	
Description	Configure device to LAN Wi-Fi network. When device in softap mode, module will create a hotspot, mobile can configure after connecting to this hotspot. While using firmware of Gizwits, prefix name of module hotspot should be “XPG-GAgent-” and password should be “123456789”. When device in airlink mode, mobile can configure at any time. No matter choosing which configuration mode, when device is online, only after mobile connects to configuration LAN Wi-Fi, can confirm that device has configured success. When device has configured successfully, it would callback MAC address of device. If device has been reset, maybe you can get DID of device in callback of device discovering.	
Parameters	ssid	ssid of router
	key	password of router

	mode	Configuration mode, see details on GizWifiConfigureMode enum definition
	softAPSSIDPrefix	Prefix name or full name of SoftAP hotspot in SoftAPMode mode, default prefix should be “XPG-GAgent-”, SDK would base on this parameter to judge wether mobile has connected to SoftAP hotspot of device or not. In AirLink mode, this parameter is ignored, just pass nil.
	timeout	Configuration timeout, default is 30s.
	wifiGAgentType	Module type waiting for configuration, this is GizWifiGAgentType enum array, default value is ESP module. GizWifiGAgentType has defined all types which SDK support, it aslo defines an enum GizGAgentOther, developers can use any other configuration library to configure device.
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didSetDeviceOnboarding:(NSError *)result mac:(NSString *)mac did:(NSString *)did productKey:(NSString *)productKey;	
Callback description	Note: If you specify filter productKey while calling getBoundDevices API, when device has been configured to router, it would return configure success, but it would not be added into device list.	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	Configuration success or failure. If configuration failure, other parameters should be nil.
	mac	MAC of device
	did	DID of device, after configuring successfully, the value of DID depends on whether the device has report
	productKey	productKey of device

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	// airlink configuration
	[[GizWifiSDK sharedInstance] setDeviceOnboarding:@"your_ssid" key:@"your_key" mode:GizWifiAirLink softAPSSIDPrefix:nil timeout:60 wifiGAgentType:[NSArray arrayWithObjects: @(GizGAgentESP), nil]];
	// softap configuration
	[[GizWifiSDK sharedInstance] setDeviceOnboarding:@"your_ssid" key:@"your_key" mode:GizWifiSoftAP softAPSSIDPrefix: @"your_gagent_hotspot_prefix" timeout:60 wifiGAgentType:nil]];
	// implement callback	

	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didSetDeviceOnboarding:(NSError *)result mac:(NSString *)mac did:(NSString *)did productKey:(NSString *)productKey {
	if(result.code == GIZ_SDK_SUCCESS) {
	// configure success
	} else {
	// configure failure
	}
	}

【getBoundDevices】

Definition	- (void)getBoundDevices:(NSString *)uid token:(NSString *)token specialProductKeys:(NSArray *)specialProductKeys;	
Description	Get list of binding devices, in different network, it would have different operations: When mobile can access internet, this API would send request to cloud service to get binding devices. When mobile couldn't access internet, it would find devices of LAN in real time, but it would remain devices which have been bound before. When mobile is out of network, unbinding devices of LAN would disappear, but it would remain binding devices which you have got before. Note: In this API, if the length of UID or token is error, SDK would use previous UID or token	
Parameters	uid	UID get from user login or registration
	token	token get from user login or registration
	specialProductKeys	Specify product key of device to search, it is NSString array, you can specify one or more product key of device, it would return all devices without specifying any product key.
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didDiscovered:(NSError *)result deviceList:(NSArray *)deviceList;	
Callback description	Maybe the changing of device list would trigger this callback, the error code is GIZ_SDK_SUCCESS. Calling getBoundDevices API would trigger this callback, error code means request status of cloud, list of device means the collection merging binding devices and LAN devices.	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition.

		GIZ_SDK_SUCCESS means success, others mean failures. When result is failure, deviceList is not nil.
	deviceList	It is an array consist of GizWifiDevice object. This parameter would only return devices filter by special productKey.

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] getBoundDevices:@"your_uid" token:@"your_token" specialProductKeys:[NSArray arrayWithObjects: @"your_product_key", nil]];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didDiscovered:(NSError *)result deviceList:(NSArray *)deviceList {
	// show error reasom
	if(result.code != GIZ_SDK_SUCCESS) {
	NSLog(@"result: %@", result.localizedDescription);
	}
	// show list of devices
	NSLog(@"discovered deviceList: %@", deviceList);
	}	

【bindRemoteDevice】

Definition	- (void)bindRemoteDevice:(NSString *)uid token: (NSString *)token mac:(NSString *)mac productKey:(NSString *)productKey productSecret:(NSString *)productSecret;	
Description	Bind remote device to server	
Parameters	uid	UID get from user login or registration
	token	token get from user login or registration
	mac	MAC of device waiting for binding
	productKey	productKey of device waiting for binding
	productSecret	productSecret of device waiting for binding
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didBindDevice:(NSError *)result did:(NSString *)did;	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition. GIZ_SDK_SUCCESS means success, others mean failures.

	did	DID of binding device

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] bindRemoteDevice:@"your_uid" token:@"your_token" mac:@"your_mac" productKey:@"your_product_key" productSecret:@"your_product_secret"];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didBindDevice:(NSError *)result did:(NSString *)did {
	if(result.code == GIZ_SDK_SUCCESS) {
	// binding success
	} else {
	// binding failure
	}
	}	

【unbindDevice】

Definition	- (void)unbindDevice:(NSString *)uid token:(NSString *)token did:(NSString *)did;	
Description	unbind device from server	
Parameters	uid	UID get from user login or registration
	token	token get from user login or registration
	did	DID of device waiting for unbinding
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didUnbindDevice:(NSError *)result did:(NSString *)did;	
Callback Parameters	wifiSDK	The single instance of GizWifiSDK
	result	See details on GizWifiErrorCode enum definition. GIZ_SDK_SUCCESS means success, others mean failures. When result is failure, other callback parameters are nil.
	did	DID of unbinded devices

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] unbindDevice:@"your_uid" token:@"your_token" did:@"your_did"];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didUnbindDevice:(NSError *)result did:(NSString *)did {
	if(result.code == GIZ_SDK_SUCCESS) {	

	// unbinding success
	} else {
	// unbinding failure
	}
	}

【getCaptchaCode】

Definition	- (void)getCaptchaCode:(NSString *)appSecret;	
Description	Get image captcha. Developers login on site.gizwits.com, getting App Secret from user's application setting, then use App Secret to get image captcha.	
Parameters	appSecret	secret of app, can see this on site.gizwits.com
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didGetCaptchaCode:(NSError *)result token:(NSString *)token captchaId:(NSString *)captchaId captchaURL:(NSString *)captchaURL;	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition. GIZ_SDK_SUCCESS means success, others mean failures. When result is failure, other callback parameters would be nil.
	token	token of image captcha, it would be invalid an hour later.
	captchaId	ID of image captcha, it would be invalid 5 minutes later.
	captchaURL	URL of image captcha, it would be invalid after captcha code has been used.

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] getCaptchaCode:@"your_app_secret"];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didGetCaptchaCode:(NSError *)result token:(NSString *)token captchaId:(NSString *)captchaId captchaURL:(NSString *)captchaURL {
	if(result.code == GIZ_SDK_SUCCESS) {
	// success
	} else {
	// failure
	}
	}	

【requestSendPhoneSMSCode】

Definition	- (void)requestSendPhoneSMSCode:(NSString *)appSecret phone:(NSString *)phone;	
Description	Get SMS verify code by phone	
Parameters	appSecret	secret of app, can see it on site.gizwits.com
	phone	phone number
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didRequestSendPhoneSMSCode:(NSError *)result token:(NSString *)token	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition. GIZ_SDK_SUCCESS means success, others mean failures. When result is failure, token is nil.
	token	get token while request message captcha

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] requestSendPhoneSMSCode:@"your_app_secret" phone:@"your_phone_number"];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didRequestSendPhoneSMSCode:(NSError *)result token:(NSString *)token {
	if(result.code == GIZ_SDK_SUCCESS) {
	// success
	} else {
	// failure
	}
	}	

【requestSendPhoneSMSCode】

Definition	- (void)requestSendPhoneSMSCode:(NSString *)token captchaId:(NSString *)captchaId captchaCode:(NSString *)captchaCode phone:(NSString *)phone;	
Description	Get SMS verify code by image captcha	
Parameters	token	get token from getCaptchaCode API
	captchaId	get captchaId from getCaptchaCode API

	captchaCode	code of image captcha
	phone	phone number
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didRequestSendPhoneSMSCode:(NSError *)result token:(NSString *)token;	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition. GIZ_SDK_SUCCESS means success, others mean failures. When result is failure, other callback parameters are nil.
	token	the token by getCaptchaCode API

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] requestSendPhoneSMSCode:@"your_token" captchaId:@"your_captcha_id" captchaCode:@"your_captcha_code" phone:@"your_phone_number"];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didRequestSendPhoneSMSCode:(NSError *)result token:(NSString *)token {
	if(result.code == GIZ_SDK_SUCCESS) {
	// success
	} else {
	// failure
	}
	}	

【verifyPhoneSMSCode】

Definition	- (void)verifyPhoneSMSCode:(NSString *)token verifyCode:(NSString *)code phone:(NSString *)phone;	
Description	Verify SMS verify code. Note: if SMS verify code has been verified, it would become invalid, and couldn't use to registration again.	
Parameters	token	the token of captcha, can get it by getCaptchaCode API
	code	SMS verify code
	phone	Phone number
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didVerifyPhoneSMSCode:(NSError *)result;	

Callback parameters	wifiSDK	The sample instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition. GIZ_SDK_SUCCESS means success, others mean failures. When result is failure, other callback parameters are nil.

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] verifyPhoneSMSCode:@"your_token" verifyCode:@"your_verify_code" phone:@"your_phone_number"];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didVerifyPhoneSMSCode:(NSError *)result {
	if(result.code == GIZ_SDK_SUCCESS) {
	// success
	} else {
	// failure
	}
	}	

【registerUser】

Definition	- (void)registerUser:(NSString *)username password:(NSString *)password verifyCode:(NSString *)code accountType:(GizUserAccountType)accountType;	
Description	Registration should specify account type, username of GizUserPhone is phone number, username of GizUserEmail is email address, username of GizUserNormal is normal username.	
Parameters	username	username (phone number, email address or normal username)
	password	password
	code	SMS verify code, it would be invalid after registration, and it couldn't be used again.
	accountType	account type, see details on GizUserAccountType enum definition. If username is phone number, this parameter should be GizUserPhone. If username is email address, this parameter should be GizUserEmail. If username is normal username, this parameter should be GizUserNormal.
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didRegisterUser:(NSError *)result uid:(NSString *)uid token:(NSString *)token;	

Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition. GIZ_SDK_SUCCESS means success, others mean failures. When result is failure, other callback parameters are nil.
	uid	get UID after finishing registration
	token	get token after finishing registration

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] registerUser:@"your_phone_number" password:@"your_password" verifyCode:@"your_verify_code" accountType:GizUserPhone];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didRegisterUser:(NSError *)result uid:(NSString *)uid token:(NSString *)token {
	if(result.code == GIZ_SDK_SUCCESS) {
	// success
	} else {
	// failure
	}
	}	

【userLoginAnonymous】

Definition	- (void)userLoginAnonymous;	
Description	Login with anonymous user, no need to register account.	
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didRegisterUser:(NSError *)result uid:(NSString *)uid token:(NSString *)token;	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition. GIZ_SDK_SUCCESS means success, others mean failures. When result is failure, other callback parameters are nil.
	uid	Get UID after finishing registration
	token	Get token after finishing registration

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] userLoginAnonymous];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didUserLogin:(NSError *)result uid:(NSString *)uid token:(NSString *)token {
	if(result.code == GIZ_SDK_SUCCESS) {	

	// login success
	} else {
	// login failure
	}
	}

【userLogin】

Definition	- (void)userLogin:(NSString *)username password:(NSString *)password;	
Description	User login, using username and password which have been registered successfully. username can be phone number, email address and normal username.	
Parameters	username	username
	password	password
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didUserLogin:(NSError *)result uid:(NSString *)uid token:(NSString *)token;	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition. GIZ_SDK_SUCCESS means success, others mean failures. When result is failure, UID and token are nil.
	uid	Get UID after login success
	token	Get token after login success

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] userLogin:@"your_user_name" password:@"your_user_password"];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didUserLogin:(NSError *)result uid:(NSString *)uid token:(NSString *)token {
	if(result.code == GIZ_SDK_SUCCESS) {
	// login success
	} else {
	// login failure
	}
	}	

【changeUserPassword】

Definition	- (void)changeUserPassword:(NSString *)token oldPassword:(NSString

	*)oldPassword newPassword:(NSString *)newPassword;	
Description	Change user password	
Parameters	token	token fron user login or registration
	oldPassword	old password
	newPassword	new password
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didChangeUserPassword:(NSError *)result;	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition. GIZ_SDK_SUCCESS means success, others mean failures. When result is failure, other callback parameters are nil.

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] changeUserPassword:@"your_token" oldPassword:@"your_old_password" newPassword:@"your_new_password"];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didChangeUserPassword:(NSError *)result {
	if(result.code == GIZ_SDK_SUCCESS) {
	// success
	} else {
	// failure
	}
	}	

【resetPassword】

Definition	- (void)resetPassword:(NSString *)username verifyCode:(NSString *)code newPassword:(NSString *)newPassword accountType:(GizUserAccountType)accountType;	
Description	Reset password. GizUserPhone user would reset password by using SMS verify code. GizUserEmail user would reset password by uing reset-link of email	
Parameters	username	It should be phone number or email address.
	code	SMS verify code is required to reset password by phone. If reset by email, it can be set to nil

	newPassword	New password. It can be set to nil if reset password by email.
	accountType	Account type, see details on GizThirdAccountType enum definition. If username is phone number, this parameter should be GizUserPhone. If username is email address, this parameter should be GizUserEmail.
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didChangeUserPassword:(NSError *)result;	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition. GIZ_SDK_SUCCESS means success, others mean failures. When result is failure, other callback parameters are nil.

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] resetPassword:@"your_phone_number" verifyCode:@"your_verify_code" newPassword:@"your_new_password" accountType:GizUserPhone];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didChangeUserPassword:(NSError *)result {
	if(result.code == GIZ_SDK_SUCCESS) {
	// success
	} else {
	// failure
	}
	}	

【changeUserInfo】

Definition	- (void)changeUserInfo:(NSString *)token username:(NSString *)username SMSVerifyCode:(NSString *)code accountType:(GizUserAccountType)accountType additionalInfo:(GizUserInfo *)additionalInfo;
Description	Change user info, include username and personal additional info. username can only be changed to phone number or email address which has been registered before. This API can be used by the following case: Only change phone number Only change email address Only change personal additional info of normal user

	Change phone number and additional info Change email address and additional info If you only change personal additional info, accountTyoe should be GizUserNormal. If you change phone number, accountType should be GizUserPhone. If you change email address, accountType should be GizUserEmail.	
Parameters	token	Token from user login or registration
	username	Username which want to change, it should be phone number or email address.
	code	SMS verify code which uses to change phone number
	accountType	account type, see details on GizThirdAccountType enum definition. If you change phone number, accountType should be GizUserPhone. If you want to change email address, accountType should be GizUserEmail. If you only want to change additional info, accountType should be GizUserNomal. If you want to change username and additional info, accountType will depend on what username is.
	additionalInfo	Additional info wait for changing, see details on GizUserInfo class definition. If you only change additional info, need to set value of token, username and code as nil.
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didChangeUserInfo:(NSError *)result;	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition. GIZ_SDK_SUCCESS means success, others mean failures.

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	// change phone number
	[[GizWifiSDK sharedInstance] changeUserInfo:@"your_token" username:@"your_phone_number" SMSVerifyCode:@"your_verify_code" userType:GizUserPhone additionalInfo:nil];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didChangeUserInfo:(NSError *)result {
	if(result.code == GIZ_SDK_SUCCESS) {
	// change info success	

	} else {
	// change info failure
	}
	}

【getUserInfo】

Definition	- (void)getUserInfo:(NSString *)token;	
Description	Get user info	
Parameters	token	Token from user login or registration
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didGetUserInfo:(NSError *)result userInfo:(GizUserInfo*)userInfo;	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition. GIZ_SDK_SUCCESS means success, others mean failures.
	userInfo	User info, see details of GizUserInfo class

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] getUserInfo:@"your_token"];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didGetUserInfo:(NSError *)result userInfo:(GizUserInfo *)userInfo {
	if(result.code == GIZ_SDK_SUCCESS) {
	// get info success
	} else {
	// get info failure
	}
	}	

【transAnonymousUser】

Definition	- (void)transAnonymousUser:(NSString *)token username:(NSString *)username password:(NSString *)password verifyCode:(NSString *)code accountType:(GizUserAccountType)accountType;	
Description	Anonymous user convert to GizUserPhone or GizUserNormal. Note: username should not be registered before.	
Parameters	token	Token fron user login or registration
	username	Username, it should be normal username or phone number

	password	password
	code	SMS verify code
	accountType	Account type, see details on GizThirdAccountType enum definition. If username is phone number, this parameter should be GizUserPhone. If username is normal username, this parameter should be GizUserNormal.
Callback	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didTransAnonymousUser:(NSError *)result;	
Callback parameters	wifiSDK	The single instance of GizWifiSDK for callback
	result	See details on GizWifiErrorCode enum definition. GIZ_SDK_SUCCESS means success, others mean failures.

Sample code	[GizWifiSDK sharedInstance].delegate = self;
	[[GizWifiSDK sharedInstance] transAnonymousUser:@"your_token" username:@"your_phone_number" password:@"your_password" verifyCode:@"your_verify_code" accountType:GizUserPhone];
	// implement callback
	- (void)wifiSDK:(GizWifiSDK *)wifiSDK didTransAnonymousUser:(NSError *)result {
	if(result.code == GIZ_SDK_SUCCESS) {
	// transform success
	} else {
	// transform failure
	}
	}	

2. GizWifiDevice Class

2.1. Introduction

This is the device class of Gizwits Wi-Fi. GizWifiDevice class provide developers with device subscriptions, data notices, real-time status updates, device control, with applications in products such as controlling the water temperature of water heater. The device object is allocated by GizWifiDevice class, cannot be self-created.

2.2. Member function

Definition	Description

Definition	Description
delegate	Use the delegate to get the corresponding event. GizWifiDevice corresponds to the callback interface defined in GizWifiDeviceDelegate. Need to use which interface, callback can be.
macAddress	NSString, the mac address of device. If it is “Virtual: Site”, then it is a virtual device.
did	NSString, DID of the device on cloud
ipAddress	NSString, ip address of device. If device is a WLAN, the ip address is the domain of cloud server.
productKey	NSString, the product type identifier of the device
productName	NSString, the product name of the device
productType	GizWifiDeviceType, product type, it should be standard or gateway device.
remark	NSString, remark of device. After the device is bound, the remark can be changed, default is nil
alias	NSString, the alias of the device. After the device is bound, the alias can be changed, default is nil.
netStatus	GizWifiDeviceNetStatus, the network status of the device.
isLAN	BOOL, determines whether the device is a LAN device or WLAN device
isBind	BOOL, determines whether the device has been bound
isDisabled	BOOL, determines whether the device has been disabled by the cloud.
isSubscribed	BOOL, determines whether the device has been subscribed.
isProductDefined	BOOL, determines whether the device has defined datapoint
sharingRole	GizDeviceSharingUserRole, binding device user’s role

2.3. Callback API

Below are the callback interfaces provided by the GizWifiDevice class. Afterwards, the definitions of the API will be specified:

- didUpdateNetStatus: Notification of network status change
- didSetSubscribe: Callback when device is subscribed or unsubscribed
- didReceiveData: Callback when device receives a status report
- didSetCustomInfo: Callback when binding information of device has been set
- didGetHardwareInfo: Device hardware information callback

2.4. API

【didUpdateNetStatus】

Callback	- (void)device:(GizWifiDevice *)device didUpdateNetStatus:(GizWifiDeviceNetStatus)netStatus;	
Callback description	The callback initiates device reports on the network status changes. When the device is reconnected to power on, when it is disconnected, or when it is controllable, this callback will trigger.	
Callback parameters	device	The GizWifiDevice object which trigger callback
	netStatus	The net status of the device (Offline, online or controllable)

Sample code	// mDevice is the entity object obtained from list of device
	mDevice.delegate = self;
	// implement callback
	- (void)device:(GizWifiDevice *)device didUpdateNetStatus:(GizWifiDeviceNetStatus)netStatus {
	}	

【setSubscribe】

Definition	- (void)setSubscribe:(NSString*)productSecret subscribed:(BOOL)subscribed;	
Description	On whether the device is subscribed or unsubscribed. If the device is subscribed, it means that the user is interested in the push messages of the device. After subscription, the SDK will automatically sign in and bound the device. After being unsubscribed, the device will automatically disconnect, but will not automatically be unbound. Usually, subscriptions will always succeed and the SDK will remember whether the device is subscribed.	
Parameters	productSecret	The product secret of device. On the product information section found on GizWits Developer Zone dev.gizwits.com, the corresponding product key of the product secret can be found. There is no default value for this parameters, so developers must input the correct productSecret.
	subscribed	Whether it is subscribed or unsubscribed. YES means subscribed, while NO means the subscription is cancelled.
Callback	- (void)device:(GizWifiDevice *)device didSetSubscribe:(NSError *)result	

	isSubscribed:(BOOL)isSubscribed;	
Callback parameters	device	The single instance of GizWifiDevice for callback
	result	See the definition for GizWifiErrorCode. GIZ_SDK_SUCCESS means success, all other values are failures. When failed, subscription status will not be changed.
	isSubscribed	Whether the device is subscribed or unsubscribed. YES means it has been subscribed, NO means it has been unsubscribed.

Sample code	// mDevice is the device object obtained from list of device
	mDevice.delegate = self;
	[mDevice setSubscribe:@"your_product_secret" subscribed:YES]; // Subscribed Device
	[mDevice setSubscribe:@"your_product_secret" subscribed:NO]; // Unsubscribed device
	// implement callback
	- (void)device:(GizWifiDevice *)device didSetSubscribe:(NSError *)result isSubscribed:(BOOL)isSubscribed {
	if(result.code == GIZ_SDK_SUCCESS) {
	// When subscription or unsubscription has succeeded.
	} else {
	// Failure
	}
	}	

【getDeviceStatus】

Definition	- (void)getDeviceStatus:(NSArray*) attrs;	
Description	Acquires device status. For devices that are subscribed, the device must be controllable before the status can be acquired. If the device has variable-length datapoints, you can also search for status of special datapoint.	
Parameters	attrs	special datapoint names, using a NSString array. The default is nil. By default, SDK will return all datapoint status of a device. If only check special datapoint status, the parameter should be a NSString array.
Callback	- (void)device:(GizWifiDevice *)device didReceiveData:(NSError *)result data:(NSDictionary *)dataMap withSN:(NSNumber *)sn;	
Callback descriptiom	The response or report data of device, which couldn't be analysed by	

	SDK, will be processed as transparent data, and the error code will be GIZ_SDK_SUCCESS.	
Callback parameters	device	The GizWifiDevice object which trigger callback
	result	See the definition for GizWifiErrorCode. GIZ_SDK_SUCCESS means it has succeeded, while all others are failures. When in failure, dataMap is empty.

	data	The data content reported by the device, in dictionary format:
		{
		"data": [value],
		// The value is of the NSDictonary type, with the content being status key pair of the device. [Datapoint Name: Value], with the datapoint value type being the same as the definition on the site
		"alerts": [value],
		// The value is of the NSDictionary type, with the content being alert key pair of the device. [Datapoint Name: Value], with the datapoint value type being the same as the definition on the site
		"faults": [value],
		// The value is of the NSDictionary type, with the content being fault key pair of the device. [Datapoint Name: Value], with the datapoint value type being the same as the definition on the site
		"binary": [value],
		// The value is of the NSData type, with the content being binary data, which is not defined on site.
		}

	sn	The responded serial number of the command, which should be the same as the serial number sent by the APP. When the device answering the query command or reporting data, the serial number is 0.

Sample code	// mDevice is the device object obtained from list of device
	mDevice.delegate = self;
	[mDevice getDeviceStatus];
	// implement callback
	- (void)device:(GizWifiDevice *)device didReceiveData:(NSError *)result data:(NSDictionary *)data withSN:(NSNumber *)sn {
	if(result.code == GIZ_SDK_SUCCESS) {
	// Qurey success
	} else {
	// Query failure
	}
	}	

【write】

Definition	- (void)write:(NSDictionary *)data withSN:(int)sn;	
Description	Gives a controlled command to the device. Commands can only be sent after a subscribed device is in a controllable status.	
Parameters	data	The parameters are the commands given to the device. This is in dictionary format, the key-value pair can be input with the following ways: • If the device has definitions for its datapoints, one single sending can be given to multiple datapoints. Keys within a dictionary should be named its datapoints, and values should be the datapoint values. Value types have to be the same as datapoint definitions: （1）If the datapoint is boolean, input NSNumber as the value type; （2）If the datapoint is of the numerical type, input NSNumber as the value type; （3）If the datapoint is of the enum type, input an enumerated serial number(NSNumber type) or an enumerated string(NSString type); （4）If the datapoint is an extended type, input value as NSData type; If the device is operated in a transparent format, transparent commands can only be sent one at a time. The key in the dictionary is binary, and the value is of NSData type.
	sn	The serial number of the control command, corresponding to the responded data of the command. When device has confirmed the command, this SN will be returned.
Callback	- (void)device:(GizWifiDevice *)device didReceiveData:(NSError *)result data:(NSDictionary *)dataMap withSN:(NSNumber *)sn;	
Callback description	The response or report data of device, which couldn't be analysed by SDK, will be treated as transparent data and processed. The error code will be GIZ_SDK_SUCCESS	
Callback parameters	device	The GizWifiDevice object which trigger callback
	result	See the definition for GizWifiErrorCode. GIZ_SDK_SUCCESS means it has succeeded, while all others are failures. When in failure, dataMap is empty.
	data	The data content uploaded by the device, in dictionary format:

		{
		"data": [value], // The value is of the NSDictionary type, with the content being a status key pair of the device. [Datapoint Name: Value], with the datapoint value type being the same as the definition on the site
		"alerts": [value], // The value is of the NSDictionary type, with the content being a alert key pair of the device. [Datapoint Name: Value], with the datapoint value type being the same as the definition on the site
		"faults": [value], // The value is of the NSDictionary type, with the content being a fault key pair of the device. [Datapoint Name: Value], with the datapoint value type being the same as the definition on the site
		"binary": [value], // The value is of the NSData type, with the content being binary data, which is not defined on site.
		}

	sn	The responded serial number of the command, which should be the same as the serial number sent by the APP. When the device answering the query command or reporting data, the serial number is 0.

Sample code		// mDevice is the device object obtained from list of device
		mDevice.delegate = self;
		// open light
		int sn = 5;
		[mDevice write: @{@"LED_OnOff": @(YES)} sn:@(sn)];
		// implement callback
		- (void)device:(GizWifiDevice *)device didReceiveData:(NSError *)result data:(NSDictionary *)data withSN:(NSNumber *)sn {
		if(result.code == GIZ_SDK_SUCCESS) {
		if (sn == 5) {
		// Command serial number matches, the light opening sequence
		} else {
		// ack of other commands or data is uploaded
		}
		} else {
		// Light has failed to open
		}
		}

【setCustomInfo】

Definition	- (void)setCustomInfo:(NSString *)remark alias:(NSString *)alias;	
Description	Changes the remarks and alias. Can only be set after the device is bound.	
Parameters	remark	Changes the remarks. nil means not being changed, @“” means an empty string.
	alias	Changes the alias. nil means not being changed, @“” means an empty string.
Callback	- (void)device:(GizWifiDevice *)device didSetCustomInfo:(NSError *)result;	
Callback parameters	device	The device object of the targeted alias or remark change.
	result	See the definition for GizWifiErrorCode. GIZ_SDK_SUCCESS means success, all other values are failures.

Sample code	// mDevice is the entity object obtained from list of device
	mDevice.delegate = self;
	[mDevice setCustomInfo:@"your_remark" alias:@"your_alias"];
	// implement callback
	- (void)device:(GizWifiDevice *)device didSetCustomInfo:(NSError *)result {
	if(result.code == GIZ_SDK_SUCCESS) {
	// change success
	} else {
	// change failure
	}
	}	

【getHardwareInfo】

Definition	- (void) getHardwareInfo;	
Description	Obtains hardware information	
Callback	- (void)device:(GizWifiDevice *)device didGetHardwareInfo:(NSError *)result hardwareInfo:(NSDictionary *)hardwareInfo;	
Callback parameters	device	The device object which returns the hardware information.
	result	See the definition for GizWifiErrorCode. GIZ_SDK_SUCCESS means success, all other values are failures. When in failure, hardwareInfo is nil

	hardwareInfo	Hardware information. Corresponding hardware key-value pair are:
		{
		"wifiHardVersion": [value], // values are of the NSString type, and is the devices Wi-Fi hardware version number
		"wifiSoftVersion": [value], // values are of the NSString type, and is the device Wi-Fi software version number
		"wifiFirmwareId": [value], // ID values are of the NSString type, and is the device Wi-Fi firmware ID number
		"wifiFirmwareVer": [value], // values are of the NSString type, and is the device Wi-Fi firmware version number
		"mcuHardVersion": [value], // values are of the NSString type, and is the device hardware version number
		"mcuSoftVersion": [value], // values are of the NSString type, and is the device software version number
		"productKey": [value], // values are of the NSString type, and is the product unique identifier of device
		}

Sample code		//// mDevice is the device object obtained from list of device
		mDevice.delegate = self;
		[mDevice getHardwareInfo];
		// implement callback
		- (void)device:(GizWifiDevice *)device didGetHardwareInfo:(NSError *)result hardwareInfo:(NSDictionary *)hardwareInfo {
		if(result.code == GIZ_SDK_SUCCESS) {
		// Success
		} else {
		// Failure
		}
		}

3. GizUserInfo Class

3.1. Introduction

GizUserInfo class is provided for developers to get and modify user info.

3.2. Member function

Member function	Description
uid	Type: NSString. Get UID after user logs in. Provide get method.
username	Type: NSString. Username: phone number or email address. Provide get method
email	Type: NSString. User email address. Provide get method.
phone	Type: NSString. User phone number. Provide get method.
isAnonymous	Type: BOOL. Judge whether it is anonymous user. Provide get method.
lang	Type: NSString. User's language environment. Provide get method.
name	Type: NSString. User's nickname. Provide get and set methods.
userGender	Type: GizUserGenderType. User's gender. Provide get and set methods.
birthday	Type: NSString. User's birthday. Provide get and set methods.
address	Type: NSString. User's home address. Provide get and set methods.
remark	Type: NSString. User's remark. Provide get and set methods.
deviceBindTime	Type: NSString. This variable means the time user binds the device. Provide get methods.

4. GizWifiSSID Class

4.1. Introduction

SSID info class of the route, includes the signal name SSID and signal strength of the Wi-Fi.

4.2. Member function

Member function	Description
ssid	SSID name: Name could be searched when we connect to a Wi-Fi hotspot.
rssi	Signal strength of the corresponding hotspot. Ranges: 0-100

5. Enumeration Class Definition

5.1. Introduction

This introduces all enumeration class definition that we use in GizWifiSDK.

5.2. Definition

【GizLogPrintLevel】

Description: printing log level

Enum ID	Enum Definition	Description
0	GizLogPrintNone	Not print any log
1	GizLogPrintI	Print error log
2	GizLogPrintII	Print debug log
3	GizLogPrintAll	Print data log

【GizEventType】

Description: the type of event notificationv

Enum ID	Enum Definition	Description
0	GizEventSDK	SDK system event

Enum ID	Enum Definition	Description
1	GizEventDevice	device exception event
2	GizEventM2MService	M2M exception event
5	GizEventToken	Token invalid event

【GizWifiConfigureMode】

Description：device configuration mode

Enum ID	Enum Definition	Description
0	GizWifiSoftAP	SoftAP configuration mode
1	GizWifiAirLink	AirLink configuration mode

【GizWifiDeviceType】

Description：device type

Enum ID	Enum Definition	Description
0	GizDeviceNormal	Normal device
1	GizDeviceCenterControl	Center control device

【GizUserAccountType】

Description：account type

Enum ID	Enum Definition	Description
0	GizUserNormal	Normal user
1	GizUserPhone	Phone user
2	GizUserEmail	Email user
3	GizUserOther	Other user (include anonymous user)

【GizWifiDeviceNetStatus】

Description：the type of device net status

Enum ID	Enum Definition	Description
0	GizDeviceOffline	Offline
1	GizDeviceOnline	Online
2	GizDeviceControlled	Controlled

【GizWifiGAgentType】

Description：the type of module

Enum ID	Enum Definition	Description
0	GizGAgentMXCHIP	MXChip 3162 module
1	GizGAgentHF	HF module
2	GizGAgentRTK	RTK module
3	GizGAgentWM	WM module
4	GizGAgentESP	ESP module
5	GizGAgentQCA	Qualcomm module
6	GizGAgentTI	TI module
7	GizGAgentFSK	FSK module
8	GizGAgentMXCHIP3	MXChip V3 module
9	GizGAgentBL	BL module
10	GizGAgentAtmelEE	Atmel module
11	GizGAgentOther	Other module

【GizUserGenderType】

Description：user’s gender

Enum ID	Enum Definition	Description
0	GizUserGenderMale	Male
1	GizUserGenderFemale	Female
2	GizUserGenderUnknown	Unknow

【GizWifErrorCode】

Description：error code definition

Enum ID	Enum Definition	Descriptiom
0	GIZ_SDK_SUCCESS	SDK runs successfully
8001	GIZ_SDK_PARAM_FORM_INVALID	The format of SDK internal param is invalid
8002	GIZ_SDK_CLIENT_NOT_AUTHEN	SDK is not started yet

Enum ID	Enum Definition	Descriptiom
8003	GIZ_SDK_CLIENT_VERSION_INVALID	SDK version is invalid
8004	GIZ_SDK_UDP_PORT_BIND_FAILED	UDP port binding failed
8005	GIZ_SDK_DAEMON_EXCEPTION	Catch exception when execute SDK daemon
8006	GIZ_SDK_PARAM_INVALID	SDK param is invalid
8007	GIZ_SDK_APPID_LENGTH_ERROR	AppID length error
8008	GIZ_SDK_LOG_PATH_INVALID	SDK log file path is invalid
8009	GIZ_SDK_LOG_LEVEL_INVALID	The log level is invalid
8020	GIZ_SDK_NO_AVAILABLE_DEVICE	There‘s no available device to set server info
8021	GIZ_SDK_DEVICE_CONFIG_SEND_FAILED	Device's Wi-Fi config info sending failed
8022	GIZ_SDK_DEVICE_CONFIG_IS_RUNNING	Device's Wi-Fi config is running
8023	GIZ_SDK_DEVICE_CONFIG_TIMEOUT	Device's Wi-Fi config timeout
8024	GIZ_SDK_DEVICE_DID_INVALID	DID is invalid
8025	GIZ_SDK_DEVICE_MAC_INVALID	MAC is invalid
8026	GIZ_SDK_SUBDEVICE_DID_INVALID	The sub device is invalid
8027	GIZ_SDK_DEVICE_PASSCODE_INVALID	The passcode is invalid
8028	GIZ_SDK_DEVICE_NOT_CENTERCONTROL	The device is not central
8029	GIZ_SDK_DEVICE_NOT_SUBSCRIBED	Device is not subscribed yet
8030	GIZ_SDK_DEVICE_NO_RESPONSE	No response from device
8031	GIZ_SDK_DEVICE_NOT_READY	Device is not ready
8032	GIZ_SDK_DEVICE_NOT_BINDED	Device is not bound yet
8033	GIZ_SDK_DEVICE_CONTROL_WITH_INVALID_COMMAND	Device control command is with invalid command
8034	GIZ_SDK_DEVICE_CONTROL_FAILED	Failed to controll device
8035	GIZ_SDK_DEVICE_GET_STATUS_FAILED	Failed to get device status
8036	GIZ_SDK_DEVICE_CONTROL_VALUE_TYPE_ERROR	The command param type is error
8037	GIZ_SDK_DEVICE_CONTROL_VALUE_OUT_OF_RANGE	The command param value is out of range

Enum ID	Enum Definition	Descriptiom
8038	GIZ_SDK_DEVICE_CONTROL_NOT_WRITABLE_COMMAND	Device control command is with not writable command
8039	GIZ_SDK_BIND_DEVICE_FAILED	Device binding failed
8040	GIZ_SDK_UNBIND_DEVICE_FAILED	Device unbinding failed
8041	GIZ_SDK_DNS_FAILED	DNS parsing failed
8042	GIZ_SDK_M2M_CONNECTION_SUCCESS	Connect to M2M successfully
8043	GIZ_SDK_SET_SOCKET_NON_BLOCK_FAILED	Socket non-blocking setting failed
8044	GIZ_SDK_CONNECTION_TIMEOUT	Connection timeout
8045	GIZ_SDK_CONNECTION_REFUSED	Connection is refused
8046	GIZ_SDK_CONNECTION_ERROR	Connection error occurred
8047	GIZ_SDK_CONNECTION_CLOSED	Connection is cloesed by peer
8048	GIZ_SDK_SSL_HANDSHAKE_FAILED	SSL handshake failed
8049	GIZ_SDK_DEVICE_LOGIN_VERIFY_FAILED	Device login verifying failed
8050	GIZ_SDK_INTERNET_NOT_REACHABLE	The Internet is unreachable
8095	GIZ_SDK_HTTP_SERVER_NOT_SUPPORT_API	Cloud Service does not support the API
8096	GIZ_SDK_HTTP_ANSWER_FORMAT_ERROR	HTTP response data format error
8097	GIZ_SDK_HTTP_ANSWER_PARAM_ERROR	HTTP response parameter error
8098	GIZ_SDK_HTTP_SERVER_NO_ANSWER	No response from HTTP server
8099	GIZ_SDK_HTTP_REQUEST_FAILED	HTTP request failed
8100	GIZ_SDK_OTHERWISE	Reserved error
8101	GIZ_SDK_MEMORY_MALLOC_FAILED	Memory allocation failed
8102	GIZ_SDK_THREAD_CREATE_FAILED	Thread creation failed
8201	GIZ_SDK_DATAPOINT_NOT_DOWNLOAD	The config file of device datapoint is not yet downloaded
8202	GIZ_SDK_DATAPOINT_SERVICE_UNAVAILABLE	Config service of device datapoint is unavailable
8203	GIZ_SDK_DATAPOINT_PARSE_FAILED	Device datapoint parsing failed
8300	GIZ_SDK_SDK_NOT_INITIALIZED	SDK is not initialized yet

Enum ID	Enum Definition	Descriptiom
8301	GIZ_SDK_APK_CONTEXT_IS_NULL	Android context is null, unable to start SDK
8302	GIZ_SDK_APK_PERMISSION_NOT_SET	The permission for SDK is not set
8303	GIZ_SDK_CHMOD_DAEMON_REFUSED	Refused to change permission of SDK daemon
8304	GIZ_SDK_EXEC_DAEMON_FAILED	Failed to execute SDK daemon
8305	GIZ_SDK_EXEC_CATCH_EXCEPTION	Catch exception when execute SDK daemon
8306	GIZ_SDK_APPID_IS_EMPTY	AppID is null, unable to use SDK
8307	GIZ_SDK_UNSUPPORTED_API	This API has been discarded and no longer provide support
8308	GIZ_SDK_REQUEST_TIMEOUT	request timeout
8309	GIZ_SDK_DAEMON_VERSION_INVALID	SDK daemon version is invalid
8310	GIZ_SDK_PHONE_NOT_CONNECT_TO_SOFTAP_SSID	Phone do not connect the SoftAp SSID
8311	GIZ_SDK_DEVICE_CONFIG_SSID_NOT_MATCHED	The current Wi-Fi network is not matched the device onboarding SSID, unable to connect device with network
8312	GIZ_SDK_NOT_IN_SOFTAPMODE	Device is not in SoftAP mode
8313	GIZ_SDK_CONFIG_NO_AVAILABLE_WIFI	The phone Wi-Fi is unavailable
8314	GIZ_SDK_RAW_DATA_TRANSMIT	The current mode is raw data transparent tranmission
8315	GIZ_SDK_PRODUCT_IS_DOWNLOADING	Downloading config file of device datapoint
8316	GIZ_SDK_START_SUCCESS	SDK started successfully
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

Enum ID	Enum Definition	Descriptiom
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!
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!

Enum ID	Enum Definition	Descriptiom
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!
9046	GIZ_OPENAPI_REGISTER_IS_BUSY	Register already in progress!
9080	GIZ_OPENAPI_CANNOT_SHARE_TO_SELF	can not share device to self!
9081	GIZ_OPENAPI_ONLY_OWNER_CAN_SHARE	guest or normal user can not share device!
9082	GIZ_OPENAPI_NOT_FOUND_GUEST	guest user not found!
9083	GIZ_OPENAPI_GUEST_ALREADY_BOUND	guest user alread bound!
9084	GIZ_OPENAPI_NOT_FOUND_SHARING_INFO	sharing record not found!
9085	GIZ_OPENAPI_NOT_FOUND_THE_MESSAGE	message record not found!
9087	GIZ_OPENAPI_SHARING_IS_WAITING_FOR_ACCEPT	sharing alread created, waiting for the guest to accept!
9088	GIZ_OPENAPI_SHARING_IS_EXPIRED	sharing record expired!
9089	GIZ_OPENAPI_SHARING_IS_COMPLETED	sharing record status is not

Enum ID	Enum Definition	Descriptiom
		unaccept!
9090	GIZ_OPENAPI_INVALID_SHARING_BECAUSE_UNBINDING	owner binding disabled!
9092	GIZ_OPENAPI_ONLY_OWNER_CAN_BIND	owner exist, guest can not bind!
9093	GIZ_OPENAPI_ONLY_OWNER_CAN_OPERATE	permission denied, you are not owner!
9094	GIZ_OPENAPI_SHARING_ALREADY_CANCELLED	sharing already canceled!
9095	GIZ_OPENAPI_OWNER_CANNOT_UNBIND_SELF	can not unbind self!
9096	GIZ_OPENAPI_ONLY_GUEST_CAN_CHECK_QRCODE	permission denied, you are not guest!
9098	GIZ_OPENAPI_MESSAGE_ALREADY_DELETED	notify delele binding failed!
9099	GIZ_OPENAPI_BINDING_NOTIFY_FAILED	notify delele binding failed!
9100	GIZ_OPENAPI_ONLY_SELF_CAN_MODIFY_ALIAS	permission denied, you are not owner or guest!
9101	GIZ_OPENAPI_ONLY_RECEIVER_CAN_MARK_MESSAGE	permission denied, you are not the receiver!
9999	GIZ_OPENAPI_RESERVED	reserved