Tencent. 腾讯科技有限公司	产品版本	密级
	V2.3.2	请输入密级：公开
	项目名称： 设备接入接口协议	共 页

设备接入接口协议

（公开使用）

文档历史记录

部门	微信产品部\开放平台中心\平台开发组\架构优化组		
起始人员	**koukoutan**		
文档版本	**描述**	**撰写人员**	**更新时间**
V1.0Beta	初稿	koukoutan	2013/12/16
V1.1Beta	二维码增加设备接入类型(ble、wifi)	koukoutan	2014/1/7
v1.2Beta	加入 Q&A	koukoutan	2014/2/12
v1.3Beta	修改设备发消息、绑定、解绑协议格式，采用 xml 格式替换以前的 json 格式	koukoutan	2014/2/13
v1.4Beta	二维码 connect_protocol 字段含义调整 增加设备授权 api	koukoutan	2014/2/14
v1.5Beta	设备授权 api：明确 mac 和 auth_key 字段格式，第三方的服务 url 不再增加 device 路径	koukoutan	2014/2/19
v1.6Beta	设备授权新增： crypt_method 和 auth_ver 字段	koukoutan	2014/2/24
v1.7Beta	增加设备状态状态查询 api	koukoutan	2014/2/27
v1.8Beta	修改二维码响应结果，不再提供二维码图片下载，仅提供二维码生成的 ticket	koukoutan	2014/2/28
v1.9Beta	调用设备授权 api 时，通信采用不加密方式，字段的取值定义 ;所有的 api 使用 https 协议，并且放到外网环境，加入 ios 优化连接弹窗	koukoutan	2014/3/11
v1.9.1Beta	1.6 节增加“更新设备属性”功能	koukoutan	2014/4/10
v1.9.2Beta	1.6 节修改“conn_strategy”字段定义	koukoutan	2014/4/11
v2.0Beta	增加 1.8 章，消息接口接入社交功能 增加 1.9 章，二维码验证 QA 增加 api 频率说明，常见错误排查	jashuang koukoutan	2014/4/30
v2.1	设备授权 api 增加 conn strategy 和 close strategy 类型 授权接口，二维码获取接口，约束第三方批量操作的 device id 数的上限	koukoutan	2014/8/22
V2.2	增加 wifi 状态查询消息及 api（见 1.10 章节）	louisliang	2014/10/17

V2.3	1. 增加设备状态查询 api 的返回值 ,status=3 表示设备状态未设置（见 1.7 章节） 2. 增加新的设备授权 api（见 1.11 章节） 3. 增加新的绑定解绑 api（见 1.12 章节） 4. 增加通过 openid 获取用户绑定设备的 api（见 1.13 章节） 5. 增加 api 错误码说明（见第 3 章） 6. 第三方可在设备二维码中追加自定义数据（见 1.5 章节或 1.11.1 章节的【第三方厂商在二维码中追加自定数据】部分） 7. 用户扫描绑定时会把二维码中的自定义数据推送给第三方（见 1.2 章节的【请求参数说明】）	louisliang	2014/11/14
V2.3.1	删除 conn_strategy 的策略 8 删除 close_strategy 的策略 3	lawsonhuang	2015/11/19
V2.3.2	型号码设备授权时，新增 product_id 字段	careykliu	2015/12/15

目　录

插 图

表 格

设备接入接口协议

关键词：设备接入，微信，公众平台，http，json，post，get，xml

摘要： 协议分为两类：消息接口、API，消息接口是指公众平台将用户发送的消息发送给第三方并接收第三方的回复，API 是指公众平台提供给第三方主动调用的接口。

1. 第三方协议

第三方与微信公众平台之间的协议

1.1. 消息接口：设备通过微信发消息给第三方

【接口说明】

设备通过微信同第三方通信，并且接收第三方的响应

【请求方式】

http 请求的 post 方式

【请求 url】

由第三方设定，可在公众平台“高级功能==》开发模式”下查看和设置，举例如下：

图 第三方服务 url

公众平台会在 device url 上追加三个参数：signature=SIGNATURE& timestamp=12345678& nonce=NONCE，最终的请求 url 如下：

http://10.148.150.116:22341/cgi-bin/uly_test?signature=SIGNATURE& timestamp=12345678& nonce=NONCE

【请求内容】

<xml>
<ToUserName><![CDATA[%s]]></ToUserName>
<FromUserName><![CDATA[%s]]></FromUserName>
<CreateTime>%u</CreateTime>
<MsgType><![CDATA[%s]]></MsgType>
<DeviceType><![CDATA[%s]]></DeviceType>
<DeviceID><![CDATA[%s]]></DeviceID>
<Content><![CDATA[%s]]></Content>
<SessionID>%lu</SessionID>
<MsgID>%lu</MsgID>
<OpenID><![CDATA[%s]]></OpenID>
</xml>

【请求参数说明】

字段	是否必须	说明
signature	是	认证签名，公众平台通过第三方 appid 可以拿到第三方的认证 token，signature 为 token、timestamp、随机数组成
timestamp	是	请求发送的时间戳
nonce	是	随机数
ToUserName	是	接收方（公众号）的 user name
FromUserName	是	发送方（微信用户）的 user name
CreateTime	是	消息创建时间，消息后台生成
MsgType	是	消息类型：device_text
DeviceType	是	设备类型，目前为“公众账号原始 ID”
DeviceID	是	设备 ID，第三方提供
Content	是	消息内容，BASE64 编码
SessionID	是	微信客户端生成的 session id，用于 request 和 response 对应，因此响应中该字段第三方需要原封不变的带回

MsgId	是	消息 id，微信后台生成
OpenID	是	微信用户账号的 OpenID

设备通过微信发送消息给第三方请求协议描述

【响应】：

```
<xml>
<ToUserName><![CDATA[%s]]></ToUserName>
<FromUserName><![CDATA[%s]]></FromUserName>
<CreateTime>%u</CreateTime>
<MsgType><![CDATA[%s]]></MsgType>
<DeviceType><![CDATA[%s]]></DeviceType>
<DeviceID><![CDATA[%s]]></DeviceID>
<SessionID>%u</SessionID>
<Content><![CDATA[%s]]></Content>
</xml>
```

【响应参数说明】

字段	是否必须	说明
ToUserName	是	接收方（微信用户）的 user name
FromUserName	是	发送方（公众号）的 user name
CreateTime	是	消息创建时间，第三方自己取当前秒级时间戳
MsgType	是	消息类型（同请求参数）：device_text
DeviceType	是	设备类型（同请求参数）
DeviceID	是	设备 ID（同请求参数）
SessionID	是	微信客户端生成的 session id（同请求参数）
Content	是	消息内容，BASE64 编码

设备通过微信发送消息给第三方响应协议描述

【注意】

公众平台会将 Content 对应的 base64 编码的数据发送给微信终端，微信终端会进行解码，并将解码后的数据发送给设备

1.2. 消息接口：“绑定/解绑”设备

【接口说明】

微信用户绑定设备后，设备会通过微信给第三方发送消息

【请求方式】

http 请求的 post 方式

【请求 url】

同 1.1.章【请求 url】

【请求内容】

```
<xml>
<ToUserName><![CDATA[%s]]></ToUserName>
<FromUserName><![CDATA[%s]]></FromUserName>
<CreateTime>%u</CreateTime>
<MsgType><![CDATA[%s]]></MsgType>
<Event><![CDATA[%s]]></Event>
<DeviceType><![CDATA[%s]]></DeviceType>
<DeviceID><![CDATA[%s]]></DeviceID>
<Content><![CDATA[%s]]></Content>
<SessionID>%u</SessionID>
<OpenID><![CDATA[%s]]></OpenID>
</xml>
```

【请求参数说明】

字段	是否必须	说明
signature	是	认证签名，公众平台通过第三方 appid 可以拿到第三方的认证 token，signature 为 token、timestamp、随机数组成
timestamp	是	请求发送的时间戳
nonce	是	随机数

ToUserName	是	接收方（公众号）的 user name
FromUserName	是	发送方（微信用户）的 user name
CreateTime	是	消息创建时间，消息后台生成
MsgType	是	消息类型：device_event
Event	是	事件类型，取值为 bind/unbind bind：绑定设备 unbind：解除绑定
DeviceType	是	设备类型，目前为“公众账号原始 ID”
DeviceID	是	设备 ID，第三方提供
Content	是	当 Event 为 bind 时 ,Content 字段存放二维码中**第三方追加的自定义的数据** （详情见 1.5 章节 获取设备二维码 或 1.11.1 章节 API：获取 deviceid 和二维码）
SessionID	是	微信客户端生成的 session id，用于 request 和 response 对应，因此响应中该字段第三方需要原封不变的带回
OpenID	是	微信账号的 OpenID

用户“绑定/解绑”设备请求协议描述

【响应】：

```
<xml>
<ToUserName><![CDATA[%s]]></ToUserName>
<FromUserName><![CDATA[%s]]></FromUserName>
<CreateTime>%u</CreateTime>
<MsgType><![CDATA[%s]]></MsgType>
<Event><![CDATA[%s]]></Event>
<DeviceType><![CDATA[%s]]></DeviceType>
<DeviceID><![CDATA[%s]]></DeviceID>
<SessionID>%u</SessionID>
<Content><![CDATA[%s]]></Content>
</xml>
```

【响应参数说明】

字段	是否必须	说明
ToUserName	是	接收方（微信用户）的 user name
FromUserName	是	发送方（公众号）的 user name
CreateTime	是	消息创建时间，第三方自己取当前秒级时间戳
MsgType	是	消息类型（同请求参数）：device_event
Event	是	事件类型（同请求参数）
DeviceType	是	设备类型（同请求参数）
DeviceID	是	设备 ID （同请求参数）
SessionID	是	微信客户端生成的 session id（同请求参数）
Content	是	消息内容，BASE64 编码

用户“绑定/解绑”设备响应协议描述

【注意】

公众平台会将响应的 Content 字段对应的 base64 编码的数据发送给微信终端，微信终端会进行解码，并将解码后的数据发送给设备

1.3. API：第三方主动发送消息给设备

【接口说明】

第三方发送消息给设备主人的微信终端，并最终送达设备

【请求方式】

http 请求方式: POST

【请求 url】

https://api.weixin.qq.com/device/transmsg?access_token=ACCESS_TOKEN

【请求内容】

{

"device_type":"DEVICETYPE",

"device_id":"DEVICEID",

"open_id": " OPEN_ID",

"content": " BASE64 编码内容",

}

【参数说明】

字段	是否必须	说明
access_token	是	调用接口凭证
device_type	是	设备类型 ,目前为 “公众账号原始 ID”
device_id	是	设备 ID
content	是	消息内容，BASE64 编码
open_id	是	微信用户账号的 openid

第三方主动发送消息给设备协议描述

【请求示例】

curl

https://api.weixin.qq.com/device/transmsg?access_token=ACCESS_TOKEN -d "{\"device_type\":\"DEVICE_TYPE\",\"device_id\":\"DEVICE_ID\",\"open_id\":\"OPEN_ID\",\"content\":\"CENTENT\"}"

【响应】

正确的 Json 返回结果：

{"ret":0,"ret_info":"this is ok"}

错误的 Json 返回示例:

{"errcode":-1,"errmsg":""}

【注意】

第三方调用该服务，需要向公众平台申请权限，权限的申请需要向公众平台提供第三方的 appid

1.4. API：获取设备绑定openID

【接口说明】

通过 device type 和 device id 获取设备主人的 openid；

【请求方式】

http 请求方式: GET

【请求 url】

https://api.weixin.qq.com/device/get_openid?access_token=ACCESS_TOKEN&device_type=DEVICE_TYPE&device_id=DEVICE_ID

【参数说明】

字段	是否必须	描述
access_token	是	调用接口凭证
device_type	是	设备类型，目前为“公众账号原始 ID”
device_id	是	设备的 deviceid

获取设备绑定 openID 协议描述

【请求示例】

curl "https://api.weixin.qq.com/device/get_openid?access_token=ACCESS_TOKEN& device_type=DEVICE_TYPE&device_id=DEVICE_ID"

【响应】

返回头：

HTTP/1.1 200 OK

Connection: close

Date: Mon, 16 Dec 2013 07:48:31 GMT

Content-Type: application/json; encoding=utf-8

Content-Length: 98

正确返回:

{"open_id":["omN7ljrpaxQgK4NW4H5cRzFRtfa8","omN7ljtqrTZuvYLkjPEX_t_Pmmlg",],"resp_msg":{"ret_code":0,"error_info":"get open id list OK!"}}

【注意】

第三方调用该服务，需要申请权限，权限的申请需要向公众平台提供第三方的 appid

1.5. API：获取设备二维码

【接口说明】

第三方公众账号通过设备 id 从公众平台批量获取设备二维码

【请求方式】:

http 请求的 post 方式

【请求 url】:

https:// api.weixin.qq.com/device/create_qrcode?access_token=ATOKEN

【请求内容】(json 格式):

```
{
    "device_num":"2",
    "device_id_list":["01234","56789"]
}
```

【字段说明】:

字段	是否必须	描述
access_token	是	调用接口凭证
device_num	是	设备 id 的个数
device_id_list	是	设备 id 的列表，json 的 array 格式，其 size 必须等于 device_num

批量获取二维码请求协议描述

【请求示例】

curl https://api.weixin.qq.com/device/create_qrcode?access_token=ATOKEN -d "{\"device_num\":\"2\", \"device_id_list\":[\"ID1\",\"ID2\"]}"

【响应】：

成功：json 方式返回二维码的生成 ticket，举例如下：

```
{
    "errcode":0,
    "errmsg":"succ",
    "device_num":1,
    "code_list":[{"device_id":"id1","ticket":"t1"}]
}
```

失败：返回失败的错误码和错误信息，譬如：

{"errcode":42001,"errmsg":""}

【响应参数说明】

字段	是否必须	说明
errcode	是	错误码，0 表示设置成功，非 0 表示失败
errmsg	是	错误信息（同 errcode 对应）
device_num	是	成功生成二维码的数量
code_list	否	二维码列表（json 的数组形式），当 errcode 为 0 且 device_num 不为 0 时数组才有内容
device_id	是	设备 id
ticket	是	设备 id 对应的二维码

批量获取二维码响应协议描述

【第三方厂商在二维码中追加自定数据】

公众平台返回的二维码生成串形式如：**http://we.qq.com/d/QRCODE_TICKET(其中 QRCODE_TICKET 是微信生产的二维码 ticket)**，第三方可以自行选择是否在公众平台返回的二维码的基础之上追加自定义的数据，**是否追加自定义数据是可选的**，由第三方自行决定。

追加自定义数据的方法：在公众平台的二维码后**追加#3RD_DEFINE_DATA（其中，#作为分隔符，3RD_DEFINE_DATA 是第三方自定义数据）**，追加后的二维码形式如：**http://we.qq.com/d/QRCODE_TICKET#3RD_DEFINE_DATA** 。

在用户扫描绑定设备的时候，**公众平台会把二维码中的 3RD_DEFINE_DATA 使用 base64 编码，放到 bind 消息中**，推送给第三方（详情参考 1.2 章节 消息接口：“绑定/

解绑”设备）。

字段	字段的值
微信返回的设备二维码	http://we.qq.com/d/QRCODE_TICKET
第三方自定义数据	3RD_DEFINE_DATA
追加自定义数据后的二维码	http://we.qq.com/d/QRCODE_TICKET#3RD_DEFINE_DATA
扫描绑定时 bind 消息推送给第三方的数据	Base64 编码后的 3RD_DEFINE_DATA

【注意】

1、第三方调用该服务，需要申请权限，权限的申请需要向公众平台提供第三方的 appid

2、建议 deviceid 为英文字母、下划线、数字三类字符的串或者组合，不带其他标点符号，以免 json 串解析失败

3、二维码的生成有可能失败，因此请求的 devcie num 和响应的 devcie num 不一定相等；如果不相等，第三方需要核对下请求中哪些 device id 没有生成成功

4、响应中的 ticket 为二维码的生成串，第三方需要用这些串来生成二维码（点阵图），为了提高二维码的扫码成功率，我们建议第三方：使用 qrencode 库，QR 码版本 5，纠错等级为 Q 级，容错率不低于 20%

5、返回的 ticket 字符串，会带有 json 的敏感字符，因此，公众平台对于敏感字符做了转义（如：/字符会被转义成\/），第三方需要将这些敏感字符转义回来

6、设备二维码 ticket 生成需要耗费系统资源，因此，建议公众号开发者一次操作不超过 **5** 个

1.6. API：设备授权（厂商需要提供deviceid）

【接口说明】

第三方公众账号将 device id 及其属性信息提交公众平台进行授权

【请求方式】：

http 请求的 post 方式

【请求 url】：

https://api.weixin.qq.com/device/authorize_device?access_token=ATOKEN

【请求内容】(json 格式):

{
"device_num":"2",
"device_list":[
{"id":"dev1","mac":"mac","connect_protocol":"1|2","auth_key":"","close_strategy":"1","conn_strategy":"1","crypt_method":"0","auth_ver":"1","manu_mac_pos":"-1","ser_mac_pos":"-2"}
],
"op_type":"0"
"product_id":"12222"
}

【字段说明】:

字段	是否必须	描述
access_token	是	调用接口凭证
device_num	是	设备 id 的个数
device_list	是	设备 id 的列表，json 的 array 格式，其 size 必须等于 device_num
id	是	设备的 deviceid
mac	是	设备的 mac 地址（48bit）格式采用 16 进制串的方式（长度为 12 字节），不需要 0X 前缀，如：1234567890AB
connect_protocol	是	支持以下四种连接协议： android classic bluetooth – 1 ios classic bluetooth – 2 ble – 3 wifi -- 4 **一个设备可以支持多种连接类型，用符号"|"做分割，客户端优先选择靠前的连接方式（优先级按|关系的排序依次降低）**，举例： 1：表示设备仅支持 andiod classic bluetooth 1|2：表示设备支持 andiod 和 ios 两种 classic

		bluetooth，但是客户端优先选择 andriod classic bluetooth 协议，如果 andriod classic bluetooth 协议连接失败，再选择 ios classic bluetooth 协议进行连接 （注：安卓平台不同时支持 BLE 和 classic 类型）
auth_key	是	auth 及通信的加密 key，第三方需要将 key 烧制在设备上（128bit），格式采用 16 进制串的方式（长度为 32 字节），不需要 0X 前缀，如：1234567890ABCDEF1234567890ABCDEF
close_strategy	是	断开策略，目前支持： 1：退出公众号页面时即断开连接 2：退出公众号之后保持连接不断开
conn_strategy	是	连接策略，32 位整型，按 bit 位置位，目前仅第 1bit 和第 3bit 位有效（bit 置 0 为无效，1 为有效；第 2bit 已被废弃），且 bit 位可以按或置位（如 1|4=5），各 bit 置位含义说明如下： 1：(第 1bit 置位)在公众号对话页面，不停的尝试连接设备 4：(第 3bit 置位)处于非公众号页面(如主界面等)，微信自动连接。当用户切换微信到前台时，可能尝试去连接设备，连上后一定时间会断开
crypt_method	是	auth 加密方法，目前支持两种取值： 0：不加密 1：AES 加密（CBC 模式，PKCS7 填充方式）
auth_ver	是	auth version，设备和微信进行 auth 时，会根据该版本号来确认 auth buf 和 auth key 的格式（各 version 对应的 auth buf 及 key 的具体格式可以参看“客户端蓝牙外设协议”），该字段目前支持取值： 0：不加密的 version 1：version 1

manu_mac_pos	是	表示 mac 地址在厂商广播 manufature data 里含有 mac 地址的偏移，取值如下： -1：在尾部、 -2：表示不包含 mac 地址 其他：非法偏移
ser_mac_pos	是	表示 mac 地址在厂商 serial number 里含有 mac 地址的偏移，取值如下： -1：表示在尾部 -2：表示不包含 mac 地址 其他：非法偏移
op_type	否	请求操作的类型，限定取值为： 0：设备授权（缺省值为 0） 1：设备更新（更新已授权设备的各属性值）
product_id	否	设备的产品编号（由微信硬件平台分配）。可服务号设备功能管理页面查询。 当 op_type 为‘0’，product_id 为‘1’时，不要填写 product_id 字段（会引起不必要错误）； 当 op_typy 为‘0’，product_id 不为‘1’时，必须填写 product_id 字段； 当 op_type 为 1 时，不要填写 product_id 字段。

设备授权请求协议描述

【请求示例】

curl https://api.weixin.qq.com/device/authorize_device?access_token=ATOKEN

-d

'{"device_num":"1","device_list":[{"id":"dev1","mac":"123456789ABC","connect_protocol":"1|2","auth_key":"","close_strategy":"1","conn_strategy":"1","crypt_method":"0","auth_ver":"0","manu_mac_pos":"-1","ser_mac_pos":"-2"}],"op_type":"0"}'

【响应】：

成功：以 json 格式返回每个 device id 对应的授权状态：

{"resp":[{"base_info":{"device_type":"your_devcie_type","device_id":"id"},"errcode":0,"errmsg":"ok"}]}

失败：返回失败的错误码和错误信息，譬如：

{"errcode":42001,"errmsg":""}

【响应参数说明】

字段	是否必须	说明
resp	是	设备 id 授权的 response（json 数组形式）
base_info	是	设备基本信息（包括 device typ 和 device id ,目前 device type 为用户的原始 id）
errcode	是	错误码，0 表示设置成功，非 0 表示失败
errmsg	是	错误信息（同 errcode 对应）

设备授权响应协议描述

上表中参数描述为服务请求成功时的响应描述

【注意】

1、第三方调用该服务，需要申请权限，权限的申请需要向公众平台提供第三方的 appid

2、建议 id 字段为英文字母、下划线、数字三类字符的串或者组合，不带其他标点符号，以免 json 串解析失败

3、connec_protocol 为设备连接的协议类型，目前有四种连接方式（见字段说明），可以支持四种连接方式的任意组合，并且可以设置客户端优先选择的连接方式，客户端会优先选择该连接方式进行连接，若制定的优先协议无法连接成功，客户端回尝试指定的其他协议方式连接；其他类型可以后续再添加，请第三方同学确保值有效

4、conn_strategy 连接策略，按位进行定义取值（第 2bit 不能置位；所有 bit 均不置位也不支持，即取值为 0），譬如手环类产品，可能需要及时同步数据，可以填 5，表示在公众号对话页面 ,不停的尝试连接设备(取值 1),并且处于非公众号页面时，微信有机会去连接设备，保证数据能及时同步（取值 4）

5、crypt_method 目前支持取值为 0 和 1，对于计算能力弱的设备可以设置为 0（不进行加密处理，此时 auth_ver 也需要为 0），目前的加密方法只支持 AES

6、auth_ver 目前只支持取值为 0 或 1，不同的取值会影响“设备---微信---后台”的

auth 过程的数据包的格式，具体的取值请参看“客户端蓝牙外设协议”，并且，如果不需要加密，则 crypt_method 和 auth_ver 都需要为 0，

7、 对于 ios 蓝牙 2.0 和 4.0 设备，微信客户端做了连接建立的弹框优化操作：对于 ios 蓝牙 **4.0** 协议，广播包必须带上 mac 地址，即：manu_mac_pos 必须设置（且为 -1，非 ios 蓝牙 4.0 设备才可以设置为-2）；对于 ios 蓝牙 **2.0** 协议，iap 的 accessory Info 的 serial number 可以不带 mac 地址，ser_mac_pos 设置为-2，也可以在尾部带上 mac 地址，设置有效（-1），对于除以上两种协议以外的其他协议，该两个值的设置均无效，可以设置为 0

8、 op_type 为请求操作类型，字段缺省值为 0，即：设备授权，字段取值为 1，则将请求中的设备属性更新已有的设备属性（若设备不存在，则更新失败）

9、 授权接口的处理逻辑比较复杂，因此，约束公众号分开发者：一次批量授权的 device id 不能超过 **5** 个

10、product_id 为产品编号，微信硬件平台为在平台上添加的设备分配了产品编号，可在设备详情页中找到，同一型号的设备有相同的产品编号。如果设备支持使用型号二维码，在授权是必须填入产品编号字段。

1.7. API：设备状态查询

【接口说明】

第三方公众账号通过设备 id 查询该 id 的状态（三种状态：未授权、已授权、已绑定）

【请求方式】：

http 请求的 get 方式

【请求 url】：

https://api.weixin.qq.com/device/get_stat?access_token=ATOKEN&device_id=DEVICE_ID

【字段说明】：

字段	是否必须	描述
access_token	是	调用接口凭证
device_id	是	设备 id

查询设备 id 状态请求协议描述

【请求示例】

curl

https://api.weixin.qq.com/device/get_stat?access_token=ATOKEN&device_id=DEVICE_ID

【响应】：

成功：以 json 格式返回 device id 的状态，举例如下：

{"errcode":0,"errmsg":"ok","status":2,"status_info":"bind"}

失败：返回失败的错误码和错误信息，譬如：

{"errcode":42001,"errmsg":""}

【响应参数说明】

字段	是否必须	说明
errcode	是	错误码（0 服务请求成功，非 0 为失败）
errmsg	是	错误信息
status	是	设备状态，目前取值如下： 0：未授权 1：已经授权（尚未被用户绑定） 2：已经被用户绑定 3：属性未设置
status_info	是	status 对应的描述

查询设备 id 状态响应协议描述

上表中参数描述为服务请求成功时的响应描述

【注意】

1、第三方调用该服务，需要申请权限，权限的申请需要向公众平台提供第三方的 appid

1.8. 消息接口：接入社交功能消息

【接入说明】

当用户与第三方进行交互（消息、自定义菜单等）时，第三方可通过消息接口返回特定 xml 结构，可以让用户收取到社交功能消息（如排行版消息等）

【请求方式】：

消息接口返回串

【返回格式】:

```
<xml>
    <ToUserName><![CDATA[toUser]]></ToUserName>
    <FromUserName><![CDATA[fromUser]]></FromUserName>
    <CreateTime>123456789</CreateTime>
    <MsgType><![CDATA[hardware]]></MsgType>
    <HardWare>
        <MessageView><![CDATA[myrank]]></MessageView>
        <MessageAction><![CDATA[ranklist]]></MessageAction>
    </HardWare>
    <FuncFlag>0</FuncFlag>
</xml>
```

【字段说明】:

字段	是否必须	描述
ToUserName	是	发送方帐号（一个 OpenID）
FromUserName	是	开发者微信号
CreateTime	是	创建时间 （整型）
MsgType	是	填写 hardware
HardWare.MessageView	是	消息展示，目前支持 myrank（排行版）
HardWare.MessageAction	是	消息点击动作，目前支持 ranklist（点击跳转排行版）

回复社交功能消息

1.9. API：验证二维码

【接口说明】

第三方公众账号通过获取设备二维码的 api 得到 ticket 后，可以通过该 api 拿到对应的设备属性

【请求方式】:

http 请求的 post 方式

【请求 url】:

https://api.weixin.qq.com/device/verify_qrcode?access_token=ATOKEN

【请求内容】

```
{
    "ticket":"QRCODE_TICKET",
}
```

【字段说明】:

字段	是否必须	描述
access_token	是	调用接口凭证
ticket	是	设备二维码的 ticket

二维码验证请求协议

【请求示例】

curl https://api.weixin.qq.com/device/verify_qrcode?access_token=ATOKEN

【响应】:

成功：以 json 格式返回 device id 的状态，举例如下：

{"errcode":0,"errmsg":"ok","device_type":"gh_xxxxxx","device_id":"DEVICE_ID","mac":"MAC"}

失败：返回失败的错误码和错误信息，譬如：

{"errcode":-1,"errmsg":""}

【响应参数说明】

字段	是否必须	说明
errcode	是	错误码（0 服务请求成功，非 0 为失败）
errmsg	是	错误信息
device_type	是	设备类型
device_id	是	设备 id
mac	是	设备的 mac 地址

验证二维码响应协议描述

上表中参数描述为服务请求成功时的响应描述

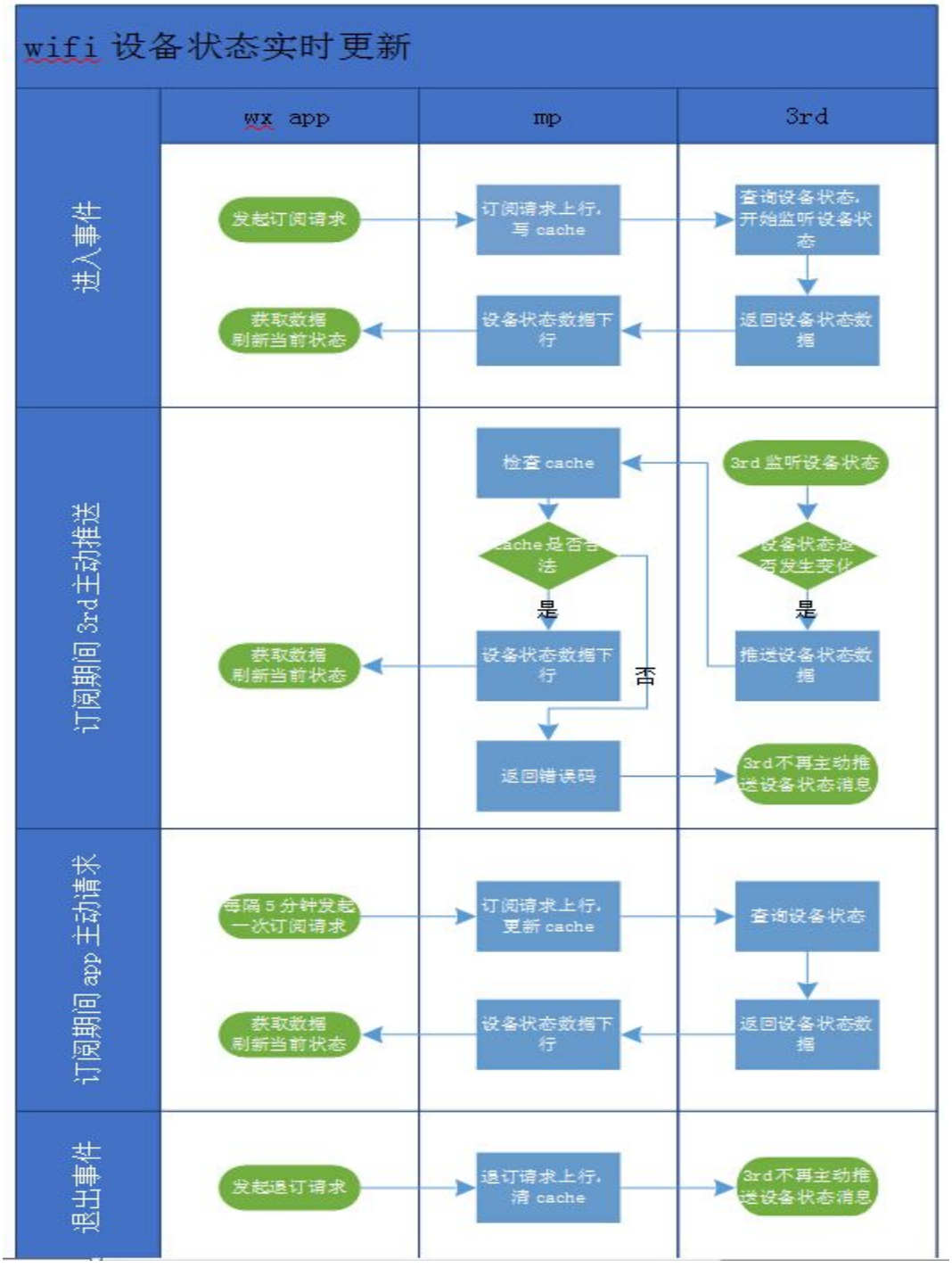
【注意】

1、第三方调用该服务，需要申请权限，权限的申请需要向公众平台提供第三方的appid

1.10. WiFi设备连接状态实时查询功能

Wifi 设备没有和客户端直连，客户端无法实时感知 wifi 设备的连接状态。为了让用户了解到 wifi 设备的连接状态，客户端会通过微信的消息通道向第三方请求设备的连接状态。另外，第三方感知到设备状态变化时可以主动通过微信消息通道把状态推送给客户端。

1.10.1 处理逻辑



1.10.2 消息接口：用户订阅/退订设备状态，消息推送给第三方

【接口说明】

用户订阅/退订设备状态，微信把消息推送给第三方，接收第三方的设备状态信息回包

【设备状态订阅/退订请求内容】

```
<xml>
    <ToUserName><![CDATA[%s]]></ToUserName>
    <FromUserName><![CDATA[%s]]></FromUserName>
    <CreateTime>%ld</CreateTime>
    <MsgType><![CDATA[%s]]></MsgType>
    <Event><![CDATA[%s]]></Event>
    <DeviceType><![CDATA[%s]]></DeviceType>
    <DeviceID><![CDATA[%s]]></DeviceID>
    <OpType>%u</OpType>
    <OpenID><![CDATA[%s]]></OpenID>
</xml>
```

【请求参数说明】

字段	是否必须	说明
signature	是	认证签名，公众平台通过第三方 appid 可以拿到第三方的认证 token，signature 为 token、timestamp、随机数组成
timestamp	是	请求发送的时间戳
nonce	是	随机数
ToUserName	是	接收方（公众号）的 user name
FromUserName	是	发送方（微信用户）的 user name
CreateTime	是	消息创建时间，消息后台生成
MsgType	是	消息类型：device_event
Event	是	事件类型，取值为 subscribe_status/unsubscribe_status subscribe_status：订阅设备状态

		unsubscribe_status：退订设备状态
DeviceType	是	设备类型，目前为“公众账号原始 ID”
DeviceID	是	设备 ID，第三方提供
OpType	是	请求类型： 0：退订设备状态； 1：心跳；(心跳的处理方式跟订阅一样) 2：订阅设备状态;
OpenID	是	微信用户账号的 OpenID

微信发送订阅设备状态消息给第三方请求协议描述

【设备状态订阅的响应内容】：

```
<xml>
    <ToUserName><![CDATA[%s]]></ToUserName>
    <FromUserName><![CDATA[%s]]></FromUserName>
    <CreateTime>%u</CreateTime>
    <MsgType><![CDATA[%s]]></MsgType>
    <DeviceType><![CDATA[%s]]></DeviceType>
    <DeviceID><![CDATA[%s id]]></DeviceID>
    <DeviceStatus>%u</DeviceStatus>
</xml>
```

【响应参数说明】

字段	是否必须	说明
ToUserName	是	接收方（微信用户）的 user name
FromUserName	是	发送方（公众号）的 user name
CreateTime	是	消息创建时间，第三方自己取当前秒级时间戳
MsgType	是	消息类型：device_status
DeviceType	是	设备类型（同请求参数）
DeviceID	是	设备 ID（同请求参数）
DeviceStatus	是	设备状态： 0--未连接； 1--已连接

1.10.3 API：第三方主动发送设备状态消息给微信终端

【接口说明】

第三方发送设备状态消息给设备主人的微信终端

【请求方式】

http 请求方式: POST

【请求 url】

https://api.weixin.qq.com/device/transmsg?access_token=ACCESS_TOKEN

【请求内容】

```
{
    "device_type":"DEVICETYPE",
    "device_id":"DEVICEID",
    "open_id": " OPEN_ID",
    "msg_type": " MSG_TYPE",
    "device_status": " DEVICE_STATUS",
}
```

【参数说明】

字段	是否必须	说明
access_token	是	调用接口凭证
device_type	是	设备类型 ,目前为 “公众账号原始 ID”
device_id	是	设备 ID
open_id	是	微信用户账号的 openid
msg_type	是	消息类型：2--设备状态消息
device_status	是	设备状态：0--未连接， 1--已连接

第三方主动发送消息给设备协议描述

【请求示例】

curl

https://api.weixin.qq.com/device/transmsg?access_token=ACCESS_TOKEN -d "{\"device_type\":\"DEVICE_TYPE\",\"device_id\":\"DEVICE_ID\",\"open_id\":\"OPEN_ID\",\"msg_type\":\"MSG_TYPE\",\"device_status\":\"DEVICE_STATUS\"}"

【响应】

正确的 Json 返回结果：

{"ret":0,"ret_info":"this is ok"}

错误的 Json 返回示例，用户已经退订设备状态，第三方不再主动推送设备状态消息:

{" ret ":-1," ret_info ":" get subscribe cache fail"}

其他错误：

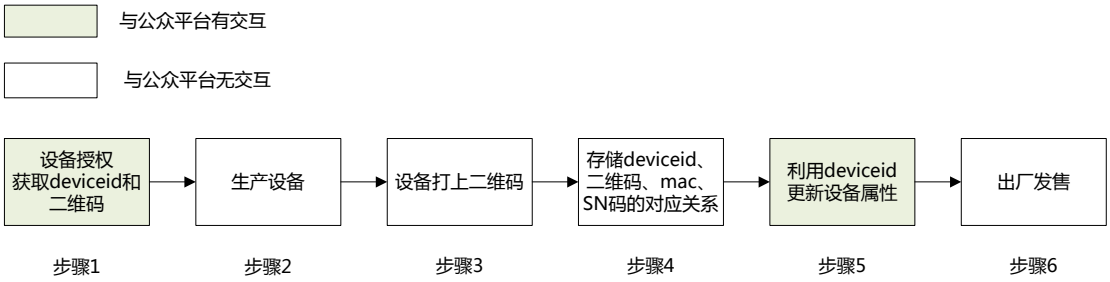
{" ret ":-1," ret_info ":" "}

1.11. 设备授权：厂商不需要提供deviceid

【摘要】

为了简化第三方厂商的生产流程，降低接入微信的门槛，公众平台提供了新的设备授权接口。新接口中，deviceid 由微信生成，作为设备在微信平台的唯一标识，第三方不需要在固件中烧入 deviceid。

设备授权流程如下：



1.11.1 API：获取 deviceid 和二维码

【接口说明】

第三方获取 deviceid 和设备二维码

【请求方式】

http 请求方式: GET

【请求 url】

https://api.weixin.qq.com/device/getqrcode?access_token=ACCESS_TOKEN&product_id=PRODUCT_ID

【请求示例】

https://api.weixin.qq.com/device/

getqrcode?access_token=ACCESS_TOKEN&product_id=PRODUCT_ID

【参数说明】

字段	是否必须	说明
product_id	否	设备的产品编号（由微信硬件平台分配）。可服务号设备功能管理页面查询。 当 product_id 为‘1’时，不要填写 product_id 字段（会引起不必要错误）； 当 product_id 不为‘1’时，必须填写 product_id 字段；

【响应】

正确的 Json 返回结果：

{resp_msg:{"ret_code":0," error_info":"ok"},"deviceid":"XXX","qrticket":"XXX"}

错误的 Json 返回示例：

{resp_msg:{"ret_code":-1," error_info":"system error"}}

返回字段说明：

字段	描述
device_id	设备 id
qrticket	设备二维码生产串

【第三方厂商在二维码中追加自定数据】

公众平台返回的二维码生成串形式如：**http://we.qq.com/d/QRCODE_TICKET(其中 QRCODE_TICKET 是微信生产的二维码 ticket)**，第三方可以自行选择是否在公众平台返回的二维码的基础之上追加自定义的数据，**是否追加自定义数据是可选的**，由第三方自行决定。

追加自定义数据的方法：在公众平台的二维码后**追加#3RD_DEFINE_DATA（其中，#作为分隔符，3RD_DEFINE_DATA 是第三方自定义数据）**，追加后的二维码形式如：**http://we.qq.com/d/QRCODE_TICKET#3RD_DEFINE_DATA** 。

在用户扫描绑定设备的时候，**公众平台会把二维码中的 3RD_DEFINE_DATA 使用 base64 编码，放到 bind 消息中**，推送给第三方（详情参考 1.2 章节 消息接口：“绑定/解绑”设备）。

字段	描述
微信返回的设备二维码	http://we.qq.com/d/QRCODE_TICKET
第三方自定义数据	3RD_DEFINE_DATA
追加自定义数据后的二维码	http://we.qq.com/d/QRCODE_TICKET#3RD_DEFINE_DATA
扫描绑定时 bind 消息推送给第三方的数据	Base64 编码后的 3RD_DEFINE_DATA

【注意】

device_id 由公众平台生成，是设备的唯一标识，且与二维码（qrticket 字段）存在对应关系。

1.11.2 API：利用 deviceid 更新设备属性

【接口说明】

第三方公众账号将 device id 及其属性信息提交公众平台进行授权

【请求方式】：

http 请求的 post 方式

【请求 url】：

https://api.weixin.qq.com/device/authorize_device?access_token=ATOKEN

【请求内容】（json 格式）：


```
{
    "device_num":"1",
    "device_list":[
    {"id":"dev1","mac":"mac","connect_protocol":"1|2","auth_key":"","close_strategy":"1","conn_strategy":"1","crypt_method":"0","auth_ver":"0","manu_mac_pos":"-1","ser_mac_pos":"-2"}
    ],
    "op_type":"1"
}
```


【字段说明】：

字段	是否必须	描述
access_token	是	调用接口凭证
device_num	是	设备 id 的个数

device_list	是	设备 id 的列表，json 的 array 格式，其 size 必须等于 device_num
id	是	设备的 deviceid
mac	是	设备的 mac 地址（48bit）格式采用 16 进制串的方式（长度为 12 字节），不需要 0X 前缀，如：1234567890AB
connect_protocol	是	支持以下四种连接协议： android classic bluetooth – 1 ios classic bluetooth – 2 ble – 3 wifi -- 4 **一个设备可以支持多种连接类型，用符号"|"做分割，客户端优先选择靠前的连接方式（优先级按|关系的排序依次降低）**，举例： 1：表示设备仅支持 andiod classic bluetooth 1|2：表示设备支持 andiod 和 ios 两种 classic bluetooth，但是客户端优先选择 andriod classic bluetooth 协议，如果 andriod classic bluetooth 协议连接失败，再选择 ios classic bluetooth 协议进行连接 （注：安卓平台不同时支持 BLE 和 classic 类型）
auth_key	是	auth 及通信的加密 key，第三方需要将 key 烧制在设备上（128bit），格式采用 16 进制串的方式（长度为 32 字节），不需要 0X 前缀，如：1234567890ABCDEF1234567890ABCDEF
close_strategy	是	断开策略，目前支持： 1：退出公众号页面时即断开连接 2：退出公众号之后保持连接不断开
conn_strategy	是	连接策略，32 位整型，按 bit 位置位，目前仅第 1bit 和第 3bit 位有效（bit 置 0 为无效，1 为有效；第 2bit 已被废弃），且 bit 位可以按或置位

		（如 1|4=5），各 bit 置位含义说明如下： 1 :（第 1bit 置位）在公众号对话页面，不停的尝试连接设备 4 :（第 3bit 置位）处于非公众号页面（如主界面等），微信自动连接。当用户切换微信到前台时，可能尝试去连接设备，连上后一定时间会断开
crypt_method	是	auth 加密方法，目前支持两种取值： 0：不加密 1：AES 加密（CBC 模式，PKCS7 填充方式）
auth_ver	是	auth version，设备和微信进行 auth 时，会根据该版本号来确认 auth buf 和 auth key 的格式（各 version 对应的 auth buf 及 key 的具体格式可以参看“客户端蓝牙外设协议”），该字段目前支持取值： 0：不加密的 version 1：version 1
manu_mac_pos	是	表示 mac 地址在厂商广播 manufature data 里含有 mac 地址的偏移，取值如下： -1：在尾部、 -2：表示不包含 mac 地址 其他：非法偏移
ser_mac_pos	是	表示 mac 地址在厂商 serial number 里含有 mac 地址的偏移，取值如下： -1：表示在尾部 -2：表示不包含 mac 地址 其他：非法偏移
op_type	否	请求操作的类型，限定取值为： 1：设备更新（更新已授权设备的各属性值）

设备授权请求协议描述

【请求示例】

curl https://api.weixin.qq.com/device/authorize_device?access_token=ATOKEN -d '{"device_num":"1","device_list":[{"id":"dev1","mac":"123456789ABC","connect_protocol":"1|2","auth_key":"","close_strategy":"1","conn_strategy":"1","crypt_method":"0","auth_ver":"0","manu_mac_pos":"-1","ser_mac_pos":"-2"}],"op_type":"1"}'

【响应】：

成功：以 json 格式返回每个 device id 对应的授权状态：

{"resp":[{"base_info":{"device_type":"your_devcie_type","device_id":"id"},"errcode":0,"errmsg":"ok"}]}

失败：返回失败的错误码和错误信息，譬如：

{"errcode":42001,"errmsg":""}

【响应参数说明】

字段	是否必须	说明
resp	是	设备 id 授权的 response（json 数组形式）
base_info	是	设备基本信息（包括 device typ 和 device id ,目前 device type 为用户的原始 id）
errcode	是	错误码，0 表示设置成功，非 0 表示失败
errmsg	是	错误信息（同 errcode 对应）

更新设备属性响应协议描述

上表中参数描述为服务请求成功时的响应描述

【注意】

10、 第三方调用该服务，需要申请权限，权限的申请需要向公众平台提供第三方的 appid

11、 建议 id 字段为英文字母、下划线、数字三类字符的串或者组合，不带其他标点符号，以免 json 串解析失败

12、 connec_protocol 为设备连接的协议类型，目前有四种连接方式（见字段说明），可以支持四种连接方式的任意组合，并且可以设置客户端优先选择的连接方式，客户端会优先选择该连接方式进行连接，若制定的优先协议无法连接成功，客户端回尝试指定的其他协议方式连接；其他类型可以后续再添加，请第三方同学确

保值有效

13、 conn_strategy 连接策略，按位进行定义取值（第 2bit 不能置位；所有 bit 均不置位也不支持，即取值为 0），譬如手环类产品，可能需要及时同步数据，可以填 5，表示在公众号对话页面，不停的尝试连接设备（取值 1），并且处于非公众号页面时，微信有机会去连接设备，保证数据能及时同步（取值 4）

14、 crypt_method 目前支持取值为 0 和 1，对于计算能力弱的设备可以设置为 0（不进行加密处理，此时 auth_ver 也需要为 0），目前的加密方法只支持 AES

15、 auth_ver 目前只支持取值为 0 或 1，不同的取值会影响“设备---微信---后台”的 auth 过程的数据包的格式，具体的取值请参看“客户端蓝牙外设协议”，并且，如果不需要加密，则 crypt_method 和 auth_ver 都需要为 0，

16、 对于 ios 蓝牙 2.0 和 4.0 设备，微信客户端做了连接建立的弹框优化操作：对于 ios 蓝牙 **4.0** 协议，广播包必须带上 mac 地址，即：manu_mac_pos 必须设置（且为-1，非 ios 蓝牙 4.0 设备才可以设置为-2）；对于 ios 蓝牙 **2.0** 协议，iap 的 accessory Info 的 serial number 可以不带 mac 地址，ser_mac_pos 设置为-2，也可以在尾部带上 mac 地址，设置有效（-1），对于除以上两种协议以外的其他协议，该两个值的设置均无效，可以设置为 0

1.12. 设备绑定/解绑：设备绑定/解绑结果由第三方控制

【摘要】

针对不同类型的设备，可能会有不同的绑定/解绑策略，为了让第三方厂商灵活地根据自身产品定制绑定/解绑的策略，通过新的接口，微信把设备绑定/解绑处理的决定权交给第三方厂商。

目前绑定/解绑有 2 种体验：

1. 用户通过第三方 H5 直连第三方后台绑定/解绑设备：在第三方后台处理完成后,若绑定/解绑成功，调用 API 把绑定/解绑的结果推送给公众平台。

2. 第三方强制绑定/解绑：第三方调用 API 主要用于客服场景，帮助用户解决无法绑定/解绑的问题。

1.12.1 API：绑定成功通知

【接口说明】

第三方后台绑定操作处理成功，通知公众平台

【请求方式】：

http 请求的 post 方式

【请求 url】:

https://api.weixin.qq.com/device/bind?access_token=ACCESS_TOKEN

【请求内容】

{

"ticket":"TICKET",

"device_id":"DEVICEID",

"openid": " OPENID",

}

【字段说明】:

字段	是否必须	描述
access_token	是	调用接口凭证
ticket	是	绑定操作合法性的凭证（由微信后台生成，第三方 H5 通过客户端 jsapi 获得）
device_id	是	设备 id
openid	是	用户对应的 openid

确认绑定操作请求协议描述

【请求示例】

curl https://api.weixin.qq.com/device/bind?access_token=ATOKEN -d '{"ticket":"TICKET","device_id":"DEVICEID","openid":"OPENID"}' -v

【响应】:

成功：以 json 格式返回 device id 的状态，举例如下：

{base_resp:{"errcode": 0,"errmsg":"ok"}}

失败：返回失败的错误码和错误信息，如：

{base_resp:{"errcode": -1,"errmsg":"system error"}}

1.12.2 API：解绑成功通知

【接口说明】

第三方确认用户和设备的解绑操作

【请求方式】:

http 请求的 post 方式

【请求 url】:

https://api.weixin.qq.com/device/unbind?access_token=ACCESS_TOKEN

【请求内容】

```
{
    "ticket":"TICKET",
    "device_id":"DEVICEID",
    "openid": " OPENID",
}
```

【字段说明】:

字段	是否必须	描述
access_token	是	调用接口凭证
ticket	是	确认解绑操作的凭证
device_id	是	设备 id
openid	是	用户对应的 openid

确认解绑操作请求协议描述

【请求示例】

curl https://api.weixin.qq.com/device/unbind?access_token=ATOKEN -d '{"ticket":"TICKET","device_id":"DEVICEID","openid":"OPENID"}' -v

【响应】:

成功：以 json 格式返回 device id 的状态，举例如下：

{base_resp:{"errcode": 0,"errmsg":"ok"}}

失败：返回失败的错误码和错误信息，如：

{base_resp:{"errcode": -1,"errmsg":"system error"}}

1.12.3 API：强制绑定用户和设备

【接口说明】

第三方强制绑定用户和设备

【请求方式】：

http 请求的 post 方式

【请求 url】：

https://api.weixin.qq.com/device/compel_bind?access_token=ACCESS_TOKEN

【请求内容】

```
{
    "device_id":"DEVICEID",
    "openid": " OPENID",
}
```

【字段说明】：

字段	是否必须	描述
access_token	是	调用接口凭证
device_id	是	设备 id
openid	是	用户对应的 openid

强制绑定请求协议描述

【请求示例】

curl https://api.weixin.qq.com/device/compel_bind?access_token=ATOKEN -d '{"ticket":"TICKET","device_id":"DEVICEID","openid":"OPENID"}' -v

【响应】：

成功：以 json 格式返回 device id 的状态，举例如下：

{base_resp:{"errcode": 0,"errmsg":"ok"}}

失败：返回失败的错误码和错误信息，如：

{base_resp:{"errcode": -1,"errmsg":"system error"}}

1.12.4 API：强制解绑用户和设备

【接口说明】

第三方强制解绑用户和设备

【请求方式】：

http 请求的 post 方式

【请求 url】:

https://api.weixin.qq.com/device/compel_unbind?access_token=ATOKEN

【请求内容】

{

"device_id":"DEVICEID",

"openid": " OPENID",

}

【字段说明】:

字段	是否必须	描述
access_token	是	调用接口凭证
device_id	是	设备 id
openid	是	用户对应的 openid

强制解绑请求协议描述

【请求示例】

curl https://api.weixin.qq.com/device/compel_unbind?access_token=ATOKEN

-d '{"ticket":"TICKET","device_id":"DEVICEID","openid":"OPENID"}' -v

【响应】:

成功：以 json 格式返回 device id 的状态，举例如下：

{base_resp:{"errcode": 0,"errmsg":"ok"}}

失败：返回失败的错误码和错误信息，如：

{base_resp:{"errcode": -1,"errmsg":"system error"}}

1.13. API：通过openid获取用户绑定的deivceid

【接口说明】

通过 openid 获取用户在当前 devicetype 下绑定的 deviceid 列表；

【请求方式】

http 请求方式: GET

【请求 url】

https://api.weixin.qq.com/device/get_bind_device?access_token=ACCESS_TOKEN&openid=OPENID

【参数说明】

字段	是否必须	描述
access_token	是	调用接口凭证
openid	是	要查询的用户的 openid

获取设备绑定 openID 协议描述

【请求示例】

curl "https://api.weixin.qq.com/device/get_bind_device?access_token=ACCESS_TOKEN& openid=OPENID"

【响应】

正确返回:

{"resp_msg":{"ret_code":0,"error_info":"ok"},"openid":"OPENID","device_list":[{"device_type":"dt1","device_id":"di1"}]}

错误返回：

{"resp_msg":{"ret_code":-1,"error_info":"ERR_MSG"},"openid":"OPENID"}

2. Q&A

2.1. 流程

Q：第三方进行设备相关功能的开发和调试，需要哪些步骤进行？

A：主要按照以下步骤进行：

1、需要先熟悉公众平台已有功能

2、详细阅读本文档的“第三方协议”相关章节

3、向公众平台后台提交“设备功能”的 API 使用权限申请，否则无法使用相关 API（申请方法见“权限申请”章节）

4、向公众平台提交 device id 的授权，否则 devcie id 无法被用户绑定（申请方法见“权限申请”章节）

5、 确保第三方服务 url 可用，第三方服务 url 的修改：登录公众平---功能---高级功能---开发模式---URL---修改

6、 获取二维码，开发、调试

2.2. 权限申请

Q：如何申请“设备功能” API 的使用权限？

A :第三方提供该公众账号的 appid 给到公众平台产品或者后台负责人 ,appid 的获取：登录公众平---功能---高级功能---开发模式---appid

Q：如何申请 device id 的使用权限？

A：向公众平台提交 device id 的授权（需要提供设备类型（目前支持蓝牙、WIFI 两种设备）、device id、原始 id，原始 id 的获取：登录公众平台---设置---账号信息---原始 ID）

Q：api 调用的频率控制：

目前，公众平台的 api 调用是有频率控制的，设备 api 的频率控制如下：

api	功能	频率
transmsg	第三方主动发送消息给设备	20w/day
get_openid	获取设备绑定 openID	2k/day
create_qrcode	获取设备二维码	2w/day
authorize_device	设备授权	2w/day
get_stat	设备状态查询	2k/day
verify_qrcode	验证二维码	50w/day

api 频率控制

每个 api 调用超过频率限制后，需要等到第二天凌晨 0 点，才可以恢复服务

2.3. 错误处理

Q：调用设备功能相关的 API，返回错误信息” user unauthorized”

A：第三方没有使用 API 的权限，需要向公众平台后台提交“设备功能”的 API 使用权限（申请方法见“权限申请章节”）

Q :为什么设备发给微信的数据和第三方云端接收到的公众平台平台发送的请求数据不一样？

A：公众平台目前对第三方的协议采用的是“文本协议”方式，因此，对于设备消息数据是经过 base64 加密后的数据，因此：

1、对于设备给第三方发数据，数据流是：设备数据---微信---公众平台 base64 加密数据---第三方云；因此，第三方收到的数据是对设备原始数据进行 base64 加密的数据，第三方需要 base64 解密，才能得到原始数据

2、对于第三方给设备发送响应，数据流是：第三方 base64 加密数据---公众平台---微信终端 base64 解密得到原始数据---设备 ;因此第三方云发送给设备的数据一定是经过 base64 加密的，而设备收到的数据则是 base64 解密后的原始数据

Q：为什么扫描设备二维码，提示设备不存在？

A：确认 device id 是否已经申请授权（授权方式见“权限申请”章节）

Q：为什么调用“主动发消息给设备”的 API 提示“get device_id error”？

A：请确认公众账号是否申请授权了该 device id，并且详细确认调用 API 中的 device_id，device_type，opend_id 是否正确？device_type 目前为公众账号的原始 ID，open_id 必须绑定了该 device_id

Q：为什么调用“主动发消息给设备”的 API 提示成功，但是设备没有收到消息？

A：确保发送的消息中 content 字段是经过 base64 加密的数据，确保 openid 对应的用户已经扫描且绑定了该 device id，确保该公众号账号拥有且授权了该 device id，

Q：为什么第三方收到的 base64 解码后的数据和设备发送的原始数据不一样？

A：base64 算法有很多变种：被编码字符长度不是 3 的倍数的时候，则都用 0 代替，对应的输出字符为=，当然，这个输出字符是可以定制为其他符号，公众平台平台采用的是原始默认的=作为补充字符

此外，很多 framework 在 http 的包体中将英文=字符识别为特殊字符，因此用到相关 framework 的第三方开发人员需要做好兼容处理

Q：发现文档中示例，最终返回失败

A：目前连调发现 api 调用失败的主要原因有：

1、 请求的 url 中带有空格，导致取 ulr 参数出错

2、 post 请求包的 json 串中有多余空格、有中文标点（引号，冒号等），json 的字段顺序和文档描述不符

2.4. 特殊逻辑

Q：对于绑定、解绑定、设备通信三类请求，第三方是否可以不回给公众平台回包？

A：这三类请求，都需要第三方回包，因为公众平台后台给第三方发包后，会超时等待第三方的回包，如果第三方不回包，会严重影响公众平台后台性能，一经发现，我们将会踢掉该公众账号。

对于“绑定”和“解绑定”请求，第三方可以回一个空包，即：post 响应只包含 http 包头，不包含数据，对于空数据，公众平台后台会直接屏蔽掉该消息，而不会下发给微信客户端，也到达不了设备。对于“设备通信”请求，是需要回复非空的符合消息协议的 http 的 post 响应

Q：我有两个公众账号，可否用一个公众账号来给另一个账号的 device id 绑定的用户发消息？

A：不行！不少第三方用户混淆了两个账号，导致消息无法送达设备，用户绑定失败等错误，因此出现错误，请先确保<公众账号，device id，open id>之间的关联是完全正确的，然后再进行下一步排查

3. 错误码说明

错误码	错误信息	描述
40003	invalid openid	openid 不合法
40013	invalid appid	appid 不合法
41009	missing openid	缺少 openid 参数
43001	require GET method	要求使用 GET 请求
43002	require POST method	要求使用 POST 请求
43003	require https	要求使用 https

43005	require friend relations	要求是好友关系
44002	empty post data	post 的数据为空
47001	data format error	数据格式有误
0	ok	成功
-1	system error	系统错误
100001	not exist	查询请求不存在
100002	already exist	新增请求已经存在
100003	list size invalid	请求中的数据大小不合法
100004	qrcode invalid	二维码不合法
100005	device type invalid	device type 不合法
100006	device id invalid	device id 不合法
100007	device status invalid	设备状态不合法
100008	mac invalid	mac 地址不合法
100009	connect protocol invalid	connect protocol 不合法
100010	auth key invalid	auth key 不合法
100011	close strategy invalid	close strategy 不合法
100012	connect strategy invalid	connect strategy 不合法
100013	crypt method invalid	crypt method 不合法
100014	auth version invalid	auth version 不合法
100015	manufature mac position invalid	manufature mac position 不合法
100016	serial number mac position invalid	serial number mac position 不合法
100017	batch process size too big	批量处理请求数量不合法
100018	op_type invalid	optype 不合法
100019	account status invalid	账号状态不合法
100020	account quota not enough	账号 quota 已用完
100021	bind relation invalid	用户和设备的绑定关系不存在
100022	msg type invalid	消息类型不合法
100023	msg content invalid	消息内容不合法

100024	user not subscribe device status	用户当前没有订阅 wifi 设备的状态
100025	device attr not set	设备属性未设置
100026	ticket invalid	票据不合法