



# 微信公众号支付接口文档

## V2.2

**( 内部文档，请勿外传 )**



1.微信支付简介.....	4
1.1 功能简介.....	4
1.2 支付账户.....	4
1.3 支付方式.....	5
1.4 测试帐号.....	6
2.JS API 支付接口.....	6
2.1 支付场景.....	6
2.2 功能交互.....	8
2.3 获取当前微信版本号.....	8
2.4 显示微信安全支付标题.....	8
2.5JS API 支付接口 ( getBrandWCPayRequest ) 定义.....	9
2.6 订单详情 ( package ) 扩展字符串定义.....	11
2.7 支付签名 ( paySign ) 生成方法.....	15
2.8 接口使用示例.....	16
3.Native ( 原生 ) 支付接口.....	17
3.1 支付场景.....	17
3.2 基本交互.....	18
3.3Native ( 原生 ) 支付接口描述.....	18
3.4Native ( 原生 ) 支付 URL 定义.....	18
3.5Native ( 原生 ) 支付 URL 签名方式.....	19
3.6Native ( 原生 ) 支付回调商户后台获取 package.....	20
4.通知接口说明.....	22



4.1 通知接口简介.....	22
4.2 补单机制.....	22
4.3 通知接口参数.....	23
4.4 后台通知结果返回.....	27
4.5 后台通知签名方式.....	27
5.API 接口说明.....	30
5.1API 接口简介.....	30
5.2API 使用方式.....	30
5.3API 列表.....	30
5.3.1 发货通知 delivernotify.....	30
5.3.2 订单查询 orderquery.....	32



# 1.微信支付简介

## 1.1 功能简介

微信支付，是基于微信客户端提供的支付服务功能。同时向商户提供销售经营分析、账户和资金管理的技术支持。用户通过扫描二维码、点击图文消息进入商品页面购买等多种方式调起微信支付模块完成支付。

目前微信支持公众号内支付。其中支付方式，可以分为 JS API 支付、Native(原生)支付。商户可以结合业务场景，自主选择支付形式。

本文将全面介绍公众号支付技术解决方案。

## 1.2 支付账户

商户向微信公众平台提交企业信息以及银行账户资料，审核通过并签约后，可以获得以下帐户（包含财付通的相关支付资金账户），用于公众号支付。

帐号	作用
appId	公众号身份标识。
appSecret	公众平台 API(参考文档 API 接口部分)的权限获取所需密钥 Key，在使用所有公众平台 API 时，都需要先用它去换取 access_token，然后再进行调用。
paySignKey	公众号支付请求中用于加密的密钥 Key，可验证商户唯一身份，PaySignKey 对应于支付场景中的 appKey 值。
partnerId	财付通商户身份标识。
partnerKey	财付通商户权限密钥 Key。



注意：appSecret、paySignKey、partnerKey 是验证商户唯一性的安全标识，请妥善保管。

对于 appSecret 和 paySignKey 的区别，可以这样认为：appSecret 是 API 使用时的登录密码，会在网络中传播的；而 paySignKey 是在所有支付相关数据传输时用于加密并进行身份校验的密钥，仅保留在第三方后台和微信后台，不会在网络中传播。

## 1.3 支付方式

公众号支付有 2 种方式：

**JS API 支付**：是指用户打开图文消息或者扫描二维码，在微信内置浏览器 **打开网页进行支付**。商户网页前端通过使用微信提供的 JS API，调用微信支付模块。这种方式，适合需要在商户网页进行选购下单的购买流程。

**Native（原生）支付**：是指商户组成符合 Native（原生）支付规则的 URL 链接，用户可通过点击该链接或者扫描对应的二维码直接进入微信支付模块（微信客户端界面），即可进行支付。这种方式，适合无需选购直接支付的购买流程。

以上两种支付方式，**最大的差别在于是否需要经过网页调起支付**。以下是两种支付方式的基本交互：



## 1.4 测试帐号

名称	取值
appId	wxf8b4f85f3a794e77
appSecret	4333d426b8d01a3fe64d53f36892df
paySignKey	2Wozy2aksie1puXUBpWD8oZxiD1DfQuEaiC7KcRATv1Ino3mdopKaPGQQ7TtkNySuAmCaDCrw4xhPY5qKTBl7Fzm0RgR3c0WaVYIXZARsxzHV2x7iwPPzOz94dnwPWSn
partnerId	1900000109
partnerKey	8934e7d15453e97507ef794cf7b0519d

# 2.JS API 支付接口

## 2.1 支付场景

以下是支付场景的交互细节，请认真阅读，并设计商家页面的逻辑：

(1) 用户打开商户网页选购商品，发起支付，在网页通过 JavaScript 调用



getBrandWCPayRequest 接口，发起微信支付请求，用户进入支付流程。

(2) 商户设定订单轮询频率，前端开始轮询商户后台是否收到交易成功通知。若某时刻收到通知，即刷新前端页面。订单查询请参考下文的订单查询的 API 接口。

(3) 用户成功支付点击完成按钮后，商户的前端会收到 JavaScript 的返回值。此时，若商家已收到后台成功的通知，则向用户展示支付成功；若未收到支付成功的后台通知，需要后台再主动发起查询订单状态请求，前端显示结果，不能只依靠前端的 JavaScript 返回值。简单地说，**支付成功务必以后台状态为准，前端判断为辅。**

注：(3) 中的 **JS API 返回值只在支付成功时返回**，但**并不表示** JS API 无返回或者有其他返回值时就一定失败。鉴于以上极少数的情况，微信团队建议商户在调起微信支付后，前端页面能够提供用户主动查询订单状态的入口，待用户结束支付流程回到商户网页前端界面时能够主动触发商户后台主动查询订单结果。**以上会影响用户的支付成功体验，请仔细阅读并设计好网页处理逻辑。**



## 2.2 功能交互



## 2.3 获取当前微信版本号

由于微信 5.0 版本后才加入微信支付模块，**低版本用户调用微信公众号支付将无效**。因此，微信团队建议商户通过 user agent 来确定用户当前的版本号后再调用支付接口。以 iPhone 版本为例，可以通过 user agent 可获取如下微信版本示例信息：

"Mozilla/5.0(iphone;CPU iphone OS 5\_1\_1 like Mac OS X) AppleWebKit/534.46(KHTML,like Gecko) Mobile/9B206 MicroMessenger/5.0"

其中 5.0 为用户安装的微信版本号，商户可以判定版本号是否高于或者等于 5.0。

## 2.4 显示微信安全支付标题

**对于商户具有支付权限且需要调用微信支付的页面，为了让用户增加购买信心，确认交易环境安全，微信强烈建议商户使用“微信安全支付”标题。**

微信安全支付标题的具体显示如下：





显示支付安全标题，除了需要商户具有支付权限，还需将原始链接添加上 "showwxpaytitle=1" 的尾串。通过这种方式，商户的页面将出现微信支付安全的标识。

例如，原始 url 为：<http://weixin.qq.com>，显示安全支付标题的 url 为：  
<http://weixin.qq.com?showwxpaytitle=1>。

当用户在微信里打开 <http://weixin.qq.com> 不会直接出现微信支付安全的标题，而打开 <http://weixin.qq.com?showwxpaytitle=1> 后将出现微信支付安全标题。

## 2.5 JS API 支付接口 ( getBrandWCPayRequest ) 定义

微信 JS API 只能在微信内置浏览器中使用，其他浏览器调用无效。微信提供 getBrandWCPayRequest 接口供商户前端网页调用，调用之前微信会鉴定商户支付权限，若商户具有调起支付的权限，则将开始支付流程。这里主要介绍支付前的接口调用规则，支付状态消息通知机制请参加下文。

接口需要注意：**所有传入参数都是字符串类型！**

getBrandWCPayRequest 参数以及返回值定义

内部文档，请勿外传

参数列表

参数	必填	说明
appId	是	字段名称：公众号 id；字段来源：商户注册具有支付权限的公众号成功后即可获得；传入方式：由商户直接传入。  参数类型：字符串类型
timeStamp	是	字段名称：时间戳；字段来源：商户生成从 1970 年 1 月 1 日 00:00:00 至今的秒数，即当前的时间，且最终需要转换为字符串形式；由商户生成后传入。  参数类型：字符串类型；参数长度：32 个字节以下。
nonceStr	是	字段名称：随机字符串；字段来源：商户生成的随机字符串。由商户生成后传入。  参数类型：字符串类型；参数长度：32 个字节以下。
package	是	字段名称：扩展字符串；参数类型：字符串类型；字段来源：商户将订单信息组成该字符串，具体组成方案参见接口使用说明中 package 组包帮助；由商户按照规范拼接后传入。  参数类型：字符串类型；参数长度：4096 个字节以下。
signType	是	字段名称：签名方式；参数类型：字符串类型；字段来源：按照文档中所示填入，目前仅支持 SHA1；  参数类型：字符串类型；参数取值："SHA1"。
paySign	是	字段名称：签名；字段来源：商户将接口列表中的参数按照指定方式进行签名，签名方式使用 signType 中标示的签名方式，



		具体签名方案参见接口使用说明中签名帮助；由商户按照规范签名后传入。  参数类型：字符串类型；参数长度：40 个字符。
--	--	--

#### 返回结果

返回值	说明
err_msg	get_brand_wcpay_request:ok 支付成功

注：JS API 的返回结果 get\_brand\_wcpay\_request:ok 仅在用户成功完成支付时返回，其他情况下函数将无返回或返回其他无意义的结果，商户可以不予理会，更不应该以其他返回值作为判定支付成功或失败的条件。

## 2.6 订单详情 ( package ) 扩展字符串定义

在商户调起 JS API 时，商户需要此时确定该笔订单详情，并将该订单详情通过一定的方式进行组合放入 package。JS API 调用后，微信将通过 package 的内容生成预支付单。下面将定义 package 的所需字段列表以及签名方法。

#### package 所需字段列表

参数	必填	说明
bank_type	是	银行通道类型，由于这里是使用的微信公众号支付，因此这个字段固定为 WX，注意大写。参数取值："WX"。
body	是	商品描述。参数长度：128 字节以下。
attach	否	附加数据，原样返回。128 字节以下。
partner	是	商户号,即注册时分配的 partnerId。

out_trade_no	是	商户系统内部的订单号,32 个字符内、可包含字母,确保在商户系统唯一。  参数取值范围：32 字节以下。
total_fee	是	订单总金额，单位为分。
fee_type	是	现金支付币种,取值：1( 人民币 ),默认值是 1，暂只支持 1。
notify_url	是	通知 URL,在支付完成后,接收微信通知支付结果的 URL,需给绝对路径,255 字符内，格式如:http://wap.tenpay.com/tenpay.asp。  取值范围：255 字节以内。
spbill_create_ip	是	订单生成的机器 IP，指用户浏览器端 IP，不是商户服务器 IP，格式为 IPV4 整型。  取值范围：15 字节以内。
time_start	否	交易起始时间，也是订单生成时间，格式为 yyyyMMddHHmmss，如 2009 年 12 月 25 日 9 点 10 分 10 秒表示为 20091225091010。时区为 GMT+8 beijing。该时间取自商户服务器。  取值范围：14 字节。
time_expire	否	交易结束时间，也是订单失效时间，格式为 yyyyMMddHHmmss，如 2009 年 12 月 27 日 9 点 10 分 10 秒表示为 20091227091010。时区为 GMT+8 beijing。该时间取自商户服务器。

		取值范围：14 字节。
transport_fee	否	物流费用，单位为分。如果有值，必须保证 $\text{transport\_fee} + \text{product\_fee} = \text{total\_fee}$ 。
product_fee	否	商品费用，单位为分。如果有值，必须保证 $\text{transport\_fee} + \text{product\_fee} = \text{total\_fee}$ 。
goods_tag	否	商品标记，优惠券时可能用到。
input_charset	是	传入参数字符编码。取值范围："GBK"、"UTF-8"。默认："GBK"

#### package 生成方法

由于 package 中携带了生成订单的详细信息，因此在微信将对 package 里面的内容进行鉴权，确定 package 携带的信息是真实、有效、合理的。因此，这里将定义生成 package 字符串的方法。

a. 对所有传入参数按照字段名的 ASCII 码从小到大排序（字典序）后，使用 URL 键值对的格式（即  $\text{key1}=\text{value1}\&\text{key2}=\text{value2}\dots$ ）拼接成字符串 string1；

b. 在 string1 最后拼接上  $\text{key}=\text{paternerKey}$  得到 stringSignTemp 字符串，并对 stringSignTemp 进行 md5 运算，再将得到的字符串所有字符转换为大写，得到 sign 值 signValue。

c. 对 string1 中的所有键值对中的 value 进行 urlencode 转码，按照 a 步骤重新拼接成字符串，得到 string2。对于 js 前端程序，一定要使用函数 encodeURIComponent 进行 urlencode 编码（**注意！进行 urlencode 时要将空格转化为 %20 而不是 +**）。

d. 将  $\text{sign}=\text{signValue}$  拼接到 string1 后面得到最终的 package 字符串。

下面定义了一段生成 package 字符串的示范过程：

假设以下为 package 传入参数：

bank\_type=WX;

fee\_type=1 ,

body=XXX ,

input\_charset=GBK ,

partner=1900000109 ,

total\_fee=1 ,

spbill\_create\_ip=127.0.0.1

out\_trade\_no=16642817866003386000 ,

notify\_url=http://www.qq.com

i：经过 a 过程 url 键值对字典序排序后的字符串 string1 为：

```
bank_type=WX&body=XXX&fee_type=1&input_charset=GBK&notify_url=http://www.qq.com
&out_trade_no=16642817866003386000&partner=1900000109&spbill_create_ip=127.0.0.1&total_fee=1
```

ii：经过 b 过程后得到 sign 为：

sign

=

```
md5(string1&key=8934e7d15453e97507ef794cf7b0519d).toUpperCase
```

=

```
md5(
```

```
bank_type=WX&body=XXX&fee_type=1&input_charset=GBK&notify_url=http://www.qq.com
&out_trade_no=16642817866003386000&partner=1900000109&spbill_create_ip=127.0.0.1&total_fee=1&key=8934e7d15453e97507ef794cf7b0519d).toUpperCase()
```

=

```
"beef37ad19575d92e191c1e4b1474ca9".toUpperCase()
```

=

```
" BEEF37AD19575D92E191C1E4B1474CA9"
```

iii：再对 string1 中的每一个键值对中的 value 进行 urlencode 编码后得到：

```
bank_type=WX&body=XXX&fee_type=1&input_charset=GBK&notify_url=http%3a%2f%2fww
```



```
w.qq.com&out_trade_no=16642817866003386000&partner=1900000109&spbill_create_ip=127.0.0.1&total_fee=1
```

iv : 拼接上 sign 后得到最终 package 结果 :

```
bank_type=WX&body=XXX&fee_type=1&input_charset=GBK&notify_url=http%3a%2f%2fwww.qq.com&out_trade_no=16642817866003386000&partner=1900000109&spbill_create_ip=127.0.0.1&total_fee=1&sign=BEEF37AD19575D92E191C1E4B1474CA9
```

## 2.7 支付签名 ( paySign ) 生成方法

paySign 字段是对本次发起 JS API 的行为进行鉴权, 只有通过了 paySign 鉴权, 才能继续对 package 鉴权并生成预支付单。这里将定义 paySign 的生成规则。

参与 paySign 签名的字段包括: appid、timestamp、noncestr、package 以及 appkey。这里 signType 并不参与签名。

对所有待签名参数按照字段名的 ASCII 码从小到大排序 ( 字典序 ) 后, 使用 URL 键值对的格式 ( 即 key1=value1&key2=value2... ) 拼接成字符串 string1。这里需要注意的是所有参数名均为小写字符, 例如 appId 在排序后字符串则为 appid ;

对 string1 作签名算法, 字段名和字段值都采用原始值 ( 此时 package 的 value 就对应了使用 2.6 中描述的方式生成的 package ), 不进行 URL 转义。具体签名算法为 paySign = SHA1(string)。

这里给出生成 paySign 的具体示例如下 :

假设参数如下 :

```
"appId":"wxf8b4f85f3a794e77",
"timeStamp":"189026618",
"nonceStr":"adssdasssd13d",
"package":"bank_type=WX&body=XXX&fee_type=1&input_charset=GBK&notify_url=http%3a%2f%2fwww.qq.com&out_trade_no=16642817866003386000&partner=1900000109&spbill_create_ip=127.0.0.1&total_fee=1&sign=BEEF37AD19575D92E191C1E4B1474CA9"
```

```
"appKey": "2Wozy2aksie1puXUBpWD8oZxiD1DfQuEaiC7KcRATv1Ino3mdopKaPGQQ7TtkNySuAmCaDCrw4xhPY5qKTBI7Fzm0RgR3c0WaVYIXZARsxzHV2x7iwPPzOz94dnwPWSn"
```

i : 经过 a 过程键值对排序后得到 string1 为 :

```
appid=wx8b4f85f3a794e77&appkey=2Wozy2aksie1puXUBpWD8oZxiD1DfQuEaiC7KcRATv1Ino3mdopKaPGQQ7TtkNySuAmCaDCrw4xhPY5qKTBI7Fzm0RgR3c0WaVYIXZARsxzHV2x7iwPPzOz94dnwPWSn&noncestr=adssdasssd13d&package=bank_type=WX&body=XXX&fee_type=1&input_charset=GBK&notify_url=http%3a%2f%2fwww.qq.com&out_trade_no=16642817866003386000&partner=1900000109&spbill_create_ip=127.0.0.1&total_fee=1&sign=BEEF37AD19575D92E191C1E4B1474CA9&timestamp=189026618
```

ii : 经过 b 过程签名后可得到 :

```
paySign  
SHA1(appid=wx8b4f85f3a794e77&appkey=2Wozy2aksie1puXUBpWD8oZxiD1DfQuEaiC7KcRATv1Ino3mdopKaPGQQ7TtkNySuAmCaDCrw4xhPY5qKTBI7Fzm0RgR3c0WaVYIXZARsxzHV2x7iwPPzOz94dnwPWSn&noncestr=adssdasssd13d&package=bank_type=WX&body=XXX&fee_type=1&input_charset=GBK&notify_url=http%3a%2f%2fwww.qq.com&out_trade_no=16642817866003386000&partner=1900000109&spbill_create_ip=127.0.0.1&total_fee=1&sign=BEEF37AD19575D92E191C1E4B1474CA9&timestamp=189026618)  
=7717231c335a05165b1874658306fa431fe9a0de
```

## 2.8 接口使用示例

接口需要注意：**所有传入参数都是字符串类型！**

示例代码如下：

```
WeixinJSBridge.invoke('getBrandWCPayRequest',{  
  "appId": "wx8b4f85f3a794e77", //公众号名称，由商户传入  
  
  "timeStamp": "189026618", //时间戳 这里随意使用了一个值  
  
  "nonceStr": "adssdasssd13d", //随机串  
  
  "package":  
    "bank_type=WX&body=XXX&fee_type=1&input_charset=GBK&notify_url=http%3a%2f%2fwww.qq.com&out_trade_no=16642817866003386000&partner=1900000109&spbill_create_ip=127.0.0.1&total_fee=1&timestamp=189026618" :
```





```
p=127.0.0.1&total_fee=1&sign=BEEF37AD19575D92E191C1E4B1474CA9",  
  
    //扩展字段，由商户传入  
  
    "signType" : "SHA1", //微信签名方式:sha1  
  
    "paySign" : "7717231c335a05165b1874658306fa431fe9a0de" //微信签名  
  
    },function(res){  
  
    // 返回 res.err_msg,取值  
  
    // get_brand_wcpay_request:cancel 用户取消  
  
    // get_brand_wcpay_request:fail 发送失败  
  
    // get_brand_wcpay_request:ok 发送成功  
  
    WeixinJSBridge.log(res.err_msg);  
    alert(res.err_code+res.err_desc);  
});
```

## 3.Native（原生）支付接口

### 3.1 支付场景

（1）商户如果想具有 Native（原生）支付的权限，除了需要填写申请支付权限的必须信息，还需要提供一个获取订单 package 的回调 URL，以便微信后台通过 Post 的方式去获取该笔支付的订单信息。

（2）商户根据微信指定的规则制定用于微信支付的 URL 字符串，当用户点击该字符串，或者通过扫描该字符串对应的二维码时，微信后台开始进入预支付流程。

（3）微信后台鉴定权限后，通过 Post 方式向商户后台获取生成订单的必要信息，再次鉴定获取的信息成功后，客户端将进入支付流程。

（4）支付成功后，微信后台将通过 Post 机制通知商户后台该笔交易已成功。

## 3.2 基本交互

商品类型图文消息



商品二维码



直接支付，不经过网页



输入支付密码



支付成功



## 3.3 Native（原生）支付接口描述

Native（原生）支付过程中，首先需要商户定义符合 Native（原生）支付规范的 URL，也就是 Native（原生）支付 URL，同时需要在微信后台 Post 商户后台时需要提供 package 内容。

因此这里将重点介绍支付前的接口调用，支付通知等信息请查看下文。

## 3.4 Native（原生）支付 URL 定义

Native（原生）支付 URL 是一系列具有 weixin://wmpay/bizpayurl?前缀的 url，同时后面紧跟着一系列辨别商家的键值对。Native（原生）支付 URL 的规则如下：

**weixin://wmpay/bizpayurl?sign=XXXXXX&appid=XXXXXX&productid=XXXXXX&timestamp=XXXXXX&noncestr=XXXXXX**

其中 XXXXXX 为商户需要填写的内容，具体参数定义如下：

参数	必填	说明
appid	是	字段名称：公众号 id；字段来源：商户注册具有支付权限的公众号成功后即可获得；传入方式：由商户直接传入。
timestamp	是	字段名称：时间戳；字段来源：商户生成从 1970 年 1 月 1 日 00:00:00 至今的秒数，即当前的时间；由商户生成后传入。取值范围：32 字符以下
noncestr	是	字段名称：随机字符串；字段来源：商户生成的随机字符串；取值范围：长度为 32 个字符以下。由商户生成后传入。取值范围：32 字符以下
productid	是	字段名称：商品唯一 id；字段来源：商户需要定义并维护自己的商品 id，这个 id 与一张订单等价，微信后台凭借该 id 通过 Post 商户后台获取交易必须信息。由商户生成后传入。取值范围：32 字符以下
sign	是	字段名称：签名；字段来源：对前面的其他字段与 appKey 按照字典序排序后，使用 SHA1 算法得到的结果。由商户生成后传入。

### 3.5 Native（原生）支付 URL 签名方式

参与 sign 签名的字段包括：appid、timestamp、noncestr、productid 以及 appkey。

a. 对所有待签名参数按照字段名的 ASCII 码从小到大排序（字典序）后，使用 URL 键值对的格式（即 key1=value1&key2=value2...）拼接成字符串 string1。这里需要注意的是所有参数名均为小写字母，即 appId 则排序后的字符串则为 appid；



b.对 string1 作签名算法，字段名和字段值都采用原始值，不进行 URL 转义。具体签名算法为  $sign = SHA1(string)$ 。

这里给出生成 sign 的具体示例如下：

```
假设参数如下：

"appid":"wxf8b4f85f3a794e77",
"timestamp":"189026618",
"noncestr":"adssdasssd13d",
"productid": "123456"

"appkey" :

"2Wozy2aksie1puXUBpWD8oZxiD1DfQuEaiC7KcRATv1Ino3mdopKaPGQQ7TtkNySuAmCaDCrw4xhPY5qKTB17Fzm0RgR3c0WaVYIXZARsxzHV2x7iwPPzOz94dnwPWSn"

则经过 a 过程后得到：

string1=
"appid=wxf8b4f85f3a794e77&appkey=2Wozy2aksie1puXUBpWD8oZxiD1DfQuEaiC7KcRATv1Ino3mdopKaPGQQ7TtkNySuAmCaDCrw4xhPY5qKTB17Fzm0RgR3c0WaVYIXZARsxzHV2x7iwPPzOz94dnwPWSn&noncestr=adssdasssd13d&productid=123456&timestamp=189026618"

则经过 SHA1 处理后得到：

sign= 18c6122878f0e946ae294e016eddda9468de80df
```

### 3.6Native（原生）支付回调商户后台获取 package

在公众平台接到用户点击上述特殊 Native（原生）支付的 URL 之后，会调用注册时填写的商家获取订单 Package 的回调 URL。

假设回调 URL 为 <https://www.outdomain.com/cgi-bin/bizpaygetpackage>

微信公众平台调用时会使用 Post 方式，推送 xml 格式的 PostData，形如：

```
<xml>
  <AppId><![CDATA[wxf8b4f85f3a797777]]></AppId>
  <OpenId><![CDATA[111222]]></OpenId>
```

```
<IsSubscribe>1</IsSubscribe>
<ProductId><![CDATA[777111666]]></ProductId>
<TimeStamp> 1369743908</TimeStamp>
<NonceStr><![CDATA[YvMZOX28YQkoU1i4NdOnlXB1]]></NonceStr>
<AppSignature><![CDATA[a9274e4032a0fec8285f147730d88400392acb9e]]></AppSignature>
<SignMethod><![CDATA[sha1]]></SignMethod>
</xml>
```

#### 参数说明：

- ( 1 ) AppId , 公众帐号的 appid
- ( 2 ) OpenId , 点击链接准备购买商品的用户 openid
- ( 3 ) IsSubscribe , 标记用户是否订阅该公众帐号 , 1 为关注 , 0 为未关注
- ( 4 ) ProductId , 第三方的商品 ID 号
- ( 5 ) TimeStamp , 时间戳
- ( 6 ) NonceStr , 随机串
- ( 7 ) AppSignature , 参数的加密签名 , 是根据 2.7 支付签名 ( paySign ) 生成方法中所讲的签名方式生成的签名

- ( 8 ) SignMethod , 签名方式 , 目前只支持 “SHA1” 。该字段不参与签名。

第三方为了确保是来自于微信公众平台的合法请求 , 需要使用同样的方式生成签名 , 并与 AppSignature 的值进行对比。

参与签名的字段为 :appid、appkey、productid、timestamp、noncestr、openid、issubscribe。

为了返回 Package 数据 , 回调 URL 必须返回一个 xml 格式的返回数据 , 形如 :

```
<xml>
  <AppId><![CDATA[wwwwb4f85f3a797777]]></AppId>
  <Package><![CDATA[a=1&url=http%3A%2F%2Fwww.qq.com]]></Package>
  <TimeStamp> 1369745073</TimeStamp>
  <NonceStr><![CDATA[iuytxA0cH6PyTAVISB28]]></NonceStr>
  <RetCode>0</RetCode>
```

```
<RetErrMsg><![CDATA[ok]]></ RetErrMsg>
<AppSignature><![CDATA[53cca9d47b883bd4a5c85a9300df3da0cb48565c]]>
</AppSignature>
<SignMethod><![CDATA[sha1]]></ SignMethod >
</xml>
```

其中，AppSignature 依然是根据前文 paySign 所讲的签名方式生成的签名，参与签名的字段为：appid、appkey、package、timestamp、noncestr、retcode、reterrmsg。

package 的生成规则请参考 JS API 所定义的 package 生成规则。这里就不再赘述了。

其中，对于一些第三方觉得商品已经过期或者其他错误的情况，可以在 RetCode 和 RetErrMsg 中体现出来，RetCode 为 0 表明正确，可以定义其他错误；当定义其他错误时，可以在 RetErrMsg 中填上 UTF8 编码的错误提示信息，比如“该商品已经下架”，客户端会直接提示出来。

## 4.通知接口说明

### 4.1 通知接口简介

用户在成功完成支付后，微信后台通知（post）商户服务器（notify\_url）支付结果。商户可以使用 notify\_url 的通知结果进行个性化页面的展示。

### 4.2 补单机制

对后台通知交互时，如果微信收到商户的应答不是 success 或超时，微信认为通知失败，微信会通过一定的策略（如 30 分钟共 8 次）定期重新发起通知，尽可能提高通知的成功率，但微信不保证通知最终能成功。

由于存在重新发送后台通知的情况，因此同样的通知可能会多次发送给商户系统。商户



系统必须能够正确处理重复的通知。

微信推荐的做法是，当收到通知进行处理时，首先检查对应业务数据的状态，判断该通知是否已经处理过，如果没有处理过再进行处理，如果处理过直接返回 success。在对业务数据进行状态检查和处理之前，要采用数据锁进行并发控制，以避免函数重入造成的数据混乱。

目前补单机制的间隔时间为：8s、10s、10s、30s、30s、60s、120s、360s、1000s。

### 4.3 通知接口参数

后台通知通过请求中的 notify\_url 进行，采用 post 机制。返回通知中的参数一致，url 包含如下内容：

字段名	变量名	必填	类型	说明
协议参数				
签名方式	sign_type	否	String(8)	签名类型，取值：MD5、RSA，默认：MD5
接口版本	service_version	否	String(8)	版本号，默认为 1.0
字符集	input_charset	否	String(8)	字符编码，取值：GBK、UTF-8，默认：GBK。
签名	sign	是	String(32)	签名
密钥序号	sign_key_index	否	Int	多密钥支持的密钥序号，默认 1
业务参数				
交易模式	trade_mode	是	Int	1-即时到账



				其他保留
交易状态	trade_state	是	Int	支付结果：  0—成功  其他保留
支付结果信息	pay_info	否	String(64)	支付结果信息，支付成功时空
商户号	partner	是	String(10)	商户号，也即之前步骤的 partnerid,由微信统一分配的 10 位正整数(120XXXXXXX)号
付款银行	bank_type	是	String(16)	银行类型，在微信中使用 WX
银行订单号	bank_billno	否	String(32)	银行订单号
总金额	total_fee	是	Int	支付金额，单位为分，如果 discount 有值 通知的 total_fee + discount = 请求的 total_fee
币种	fee_type	是	Int	现金支付币种,目前只支持人民币,默认值是 1-人民币
通知 ID	notify_id	是	String(128)	支付结果通知 id，对于某些特定商户，只返回通知 id，要求商户据此查询交易结果
订单号	transaction_id	是	String(28)	交易号，28 位长的数值，其中





				前 10 位为商户号，之后 8 位为订单产生的日期，如 20090415，最后 10 位是流水号。
商户订单号	out_trade_no	是	String(32)	商户系统的订单号，与请求一致。
商家数据包	attach	否	String(127)	商家数据包，原样返回
支付完成时间	time_end	是	String(14)	支付完成时间，格式为 yyyyMMddhhmmss，如 2009 年 12 月 27 日 9 点 10 分 10 秒表示为 20091227091010。时区为 GMT+8 beijing。
物流费用	transport_fee	否	Int	物流费用，单位分，默认 0。 如果有值，必须保证 $\text{transport\_fee} + \text{product\_fee} = \text{total\_fee}$
物品费用	product_fee	否	Int	物品费用，单位分。如果有值，必须保证 $\text{transport\_fee} + \text{product\_fee} = \text{total\_fee}$
折扣价格	discount	否	Int	折扣价格，单位分，如果有值，通知的 $\text{total\_fee} + \text{discount} =$ 请求的 $\text{total\_fee}$
买家别名	buyer_alias	否	String(64)	对应买家账号的一个加密串



同时，在 postData 中还将包含 xml 数据。数据如下：

```
<xml>
  <OpenId><![CDATA[111222]]></OpenId>
  <AppId><![CDATA[www4f85f3a79777]]></AppId>
  <IsSubscribe>1</IsSubscribe>
  <TimeStamp> 1369743511</TimeStamp>
  <NonceStr><![CDATA[jALldRTHAFd5Tgs5]]></NonceStr>
  <AppSignature><![CDATA[bafe07f060f22dcda0bfdb4b5ff756f973aecffa]]>
  </AppSignature>
  <SignMethod><![CDATA[sha1]]></ SignMethod >
</xml>
```

各字段定义如下：

参数	必填	说明
AppId	是	字段名称：公众号 id；字段来源：商户注册具有支付权限的公众号成功后即可获得；传入方式：由商户直接传入。
TimeStamp	是	字段名称：时间戳；字段来源：商户生成从 1970 年 1 月 1 日 00：00：00 至今的秒数，即当前的时间；由商户生成后传入。取值范围：32 字符以下
NonceStr	是	字段名称：随机字符串；字段来源：商户生成的随机字符串；取值范围：长度为 32 个字符以下。由商户生成后传入。取值范围：32 字符以下
OpenId	是	支付该笔订单的用户 ID，商户可通过公众号其他接口为付款用户服务。
AppSignature	是	字段名称：签名；字段来源：对前面的其他字段与 appKey 按照字典序排序后，使用 SHA1 算法得到的结果。由商户生成后传入。



IsSubscribe	是	用户是否关注了公众号。1 为关注，0 为未关注。
-------------	---	--------------------------

AppSignature 依然是根据前文 paySign 所述的签名方式生成,参与签名的字段为 :appid、appkey、timestamp、noncestr、openid、issubscribe。

从以上信息可以看出，url 参数中携带订单相关信息，postData 中携带该次支付的用户相关信息，这将便于商家拿到 openid，以便后续提供更好的售后服务。

## 4.4 后台通知结果返回

微信后台通过 notify\_url 通知商户，商户做业务处理后，需要以字符串的形式反馈处理结果，内容如下：

返回结果	结果说明
success	处理成功，微信系统收到此结果后不再进行后续通知
fail 或其它字符	处理不成功，微信收到此结果或者没有收到任何结果，系统通过补单机 制再次通知

## 4.5 后台通知签名方式

对于 url 中签名原始串按以下方式组装成字符串：

a.除 sign 字段外，所有参数按照字段名的 ascii 码从小到大排序后使用 QueryString 的格式（即 key1=value1&key2=value2...）拼接而成字符串 string1，空值不传递，不参与签名组串。

b.在 string1 最后拼接上 key=paternerKey 得到 stringSignTemp 字符串，并对 stringSignTemp 进行 md5 运算，再将得到的字符串所有字符转换为大写，得到 sign 值



signValue。

c.对 string1 进行 urlencode 转码，得到 string2。

d.将 sign=signValue 拼接到 string1 后面得到最终的 notifyargs 字符串。

e.所有参数是指通信过程中实际出现的所有非空参数，即使是接口中无描述的字段，也需要参与签名组串。

f.签名原始串中，字段名和字段值都采用原始值，不进行 URL Encode。

g.微信通知消息可能会由于升级增加参数，请验证应答签名时注意允许这种情况。

下面定义了一段生成 notifyargs 字符串的示范过程：

假设以下为 notifyargs 传入参数：

bank\_billno=206064184488，

bank\_type=0，

discount=0，

fee\_type=1，

input\_charset=GBK，

notify\_id=WE37gwCoFBcAKdkH34Y1nW94r\_vao2ljwtE3oAHEeAP690xSVhRleOMfhsg  
jwVGDpluT-vdS79kbDbkDnjYg4qsmTdSjuJxl，

out\_trade\_no=843254536943809900，

partner=1900000109，

product\_fee=1，

sign\_type=MD5，

time\_end=20130606015331，

total\_fee=1，

trade\_mode=1 ,

trade\_state=0 ,

transaction\_id=1900000109201306060282555397 ,

transport\_fee=0

i : 经过 a 过程 url 键值对字典序排序后的字符串 string1 为 :

```
bank_billno=206064184488&bank_type=0&discount=0&fee_type=1&input_charset=GBK&notify_id=WE37gwCoFBcAKdkH34Y1nW94r_vao2ljwtE3oAHEeAP690xSVhRleOMfhsgjwVGDpluT-vdS79kbDbkDnjYg4qsmTdSjuJxl&out_trade_no=843254536943809900&partner=1900000109&product_fee=1&sign_type=MD5&time_end=20130606015331&total_fee=1&trade_mode=1&trade_state=0&transaction_id=1900000109201306060282555397&transport_fee=0
```

ii : 经过 b 过程后得到 sign 为 :

sign

=

```
md5(string1&key=8934e7d15453e97507ef794cf7b0519d).toUpperCase
```

=

```
md5(bank_billno=206064184488&bank_type=0&discount=0&fee_type=1&input_charset=GBK&notify_id=WE37gwCoFBcAKdkH34Y1nW94r_vao2ljwtE3oAHEeAP690xSVhRleOMfhsgjwVGDpluT-vdS79kbDbkDnjYg4qsmTdSjuJxl&out_trade_no=843254536943809900&partner=1900000109&product_fee=1&sign_type=MD5&time_end=20130606015331&total_fee=1&trade_mode=1&trade_state=0&transaction_id=1900000109201306060282555397&transport_fee=0&key=8934e7d15453e97507ef794cf7b0519d).toUpperCase()
```

=

```
" 8ef1f69d5d9d4ec39d3787526f27924e".toUpperCase()
```

=

```
" 8EF1F69D5D9D4EC39D3787526F27924E"
```

iii : 再对 string1 经过 c 过程 urlencode 编码后得到 :

```
bank_billno=206064184488&bank_type=0&discount=0&fee_type=1&input_charset=GBK&notify_id=WE37gwCoFBcAKdkH34Y1nW94r_vao2ljwtE3oAHEeAP690xSVhRleOMfhsgjwVGDpluT-vdS79kbDbkDnjYg4qsmTdSjuJxl&out_trade_no=843254536943809900&partner=1900000109&product_fee=1&sign_type=MD5&time_end=20130606015331&total_fee=1&trade_mode=1&trade_state=0&transaction_id=1900000109201306060282555397&transport_fee=0
```

iv : 拼接上 sign 后得到最终 package 结果 :



```
bank_billno=206064184488&bank_type=0&discount=0&fee_type=1&input_charset=GBK&notify_id=WE37gwCoFBcAKdkH34Y1nW94r_vao2ljwtE3oAHEeAP690xSVhRleOMfhsgjwVGDpluT-vdS79kbDbkDnjYg4qsmTdSjuJxl&out_trade_no=843254536943809900&partner=1900000109&product_fee=1&sign_type=MD5&time_end=20130606015331&total_fee=1&trade_mode=1&trade_state=0&transaction_id=1900000109201306060282555397&transport_fee=0&sign=8EF1F69D5D9D4EC39D3787526F27924E
```

## 5.API 接口说明

### 5.1 API 接口简介

为了更好地接入支付的整个流程，包括购买、通知、发货等，微信提供了一系列的支付相关 API，以供第三方调用。

### 5.2 API 使用方式

在使用 API 之前，需要拥有 API 使用过程中用到的具有时效性的凭证 `access_token`。这个获取方式就是使用前文提到的 `appid` 和 `appsecret` 调用 `token` 这个 api 获取。更详细的文档请参考：

<http://mp.weixin.qq.com/wiki> 说明文档中的通用接口。

只有拥有了有效 `access_token`（即相当于具有了一定时间内的 API 登录态），后续的所有 API 调用才会成功。

### 5.3 API 列表

#### 5.3.1 发货通知 `delivernotify`

为了更好地跟踪订单的情况，需要第三方在收到最终支付通知之后，调用发货通知 API



告知微信后台该订单的发货状态。**请在收到支付通知发货后，一定调用发货通知接口，否则可能影响商户信誉和资金结算。**

Api 的 url 为：[https://api.weixin.qq.com/pay/delivernotify?access\\_token=xxxxxx](https://api.weixin.qq.com/pay/delivernotify?access_token=xxxxxx)

Url 中的参数只包含目前微信公众平台凭证 access\_token，而发货通知的真正的数据是放在 PostData 中的，格式为 json，如下：

```
{
  "appid" : "wwwb4f85f3a797777",
  "openid" : "oX99MDgNcgwnz3zFN3DNmo8uwa-w",
  "transid" : "111112222233333",
  "out_trade_no" : "555666uuu",
  "deliver_timestamp" : "1369745073",
  "deliver_status" : "1",
  "deliver_msg" : "ok",
  "app_signature" : "53cca9d47b883bd4a5c85a9300df3da0cb48565c",
  "sign_method" : "sha1"
}
```

其中，

appid 是公众平台账户的 AppId；

openid 是购买用户的 OpenId，这个已经放在最终支付结果通知的 PostData 里了；

transid 是交易单号；

out\_trade\_no 是第三方订单号；

deliver\_timestamp 是发货时间戳，这里指得是 linux 时间戳；

deliver\_status 是发货状态，1 表明成功，0 表明失败，失败时需要在 deliver\_msg 填上失败原因；

deliver\_msg 是发货状态信息，失败时可以填上 UTF8 编码的错误提示信息，比如“该商品已退款”；



app\_signature 依然是根据 1 中所讲的签名方式生成的签名，参加签名字段为：appid、appkey、openid、transid、out\_trade\_no、deliver\_timestamp、deliver\_status、deliver\_msg；

sign\_method 是签名方法（不计入签名生成）；

微信公众平台在校验 ok 之后，会返回数据表明是否通知成功，例如：

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

如果有异常，会在 errcode 和 errmsg 描述出来，如果成功 errcode 就为 0。

### 5.3.2 订单查询 orderquery

因为某一方技术的原因，可能导致商家在预期时间内都收不到最终支付通知，此时商家可以通过该 API 来查询订单的详细支付状态。

Api 的 url 为：[https://api.weixin.qq.com/pay/orderquery?access\\_token=xxxxxx](https://api.weixin.qq.com/pay/orderquery?access_token=xxxxxx)

Url 中的参数只包含目前微信公众平台凭证 access\_token，而发货通知的真正的数据是放在 PostData 中的，格式为 json，如下：

```
{
  "appid" : "wwwb4f85f3a797777",
  "package" : "out_trade_no=11122&partner=1900090055&sign=4e8d0df3da0c3d0df38f",
  "timestamp" : "1369745073",
  "app_signature" : "53cca9d47b883bd4a5c85a9300df3da0cb48565c",
  "sign_method" : "sha1"
}
```

其中，

appid 是公众平台账户的 AppId；

package 是查询订单的关键信息数据，包含第三方唯一订单号 out\_trade\_no、财付通商户身份标识 partner（即前文所述的 partnerid）、签名 sign，其中 sign 是对参数字典序排序并





使用&联合起来，最后加上&key=partnerkey（唯一分配），进行md5运算，再转成全大写，最终得到sign，对于示例，就是：

```
sign=md5 ( out_trade_no=11122&partner=1900090055&key=xxxxxx ) .toupper ;
```

timestamp 是 linux 时间戳；

app\_signature 依然是根据 1 中所讲的签名方式生成的签名，参加签名字段为：appid、appkey、package、timestamp；

sign\_method 是签名方法（不计入签名生成）；

微信公众平台在校验 ok 之后，会返回数据表明是否通知成功，例如：

```
{"errcode":0,"errmsg":"ok", ..... }
```

如果有异常，会在 errcode 和 errmsg 描述出来，如果成功 errcode 就为 0。

如果查询成功，会返回详细的订单数据，如下：

```
{
  "errcode":0,
  "errmsg":"ok",
  "order_info":
  {
    "ret_code":0,
    "ret_msg": "",
    "input_charset":"GBK",
    "trade_state":"0",
    "trade_mode":"1",
    "partner":"1900000109",
    "bank_type":"CMB_FP",
    "bank_billno":"207029722724",
    "total_fee":"1",
    "fee_type":"1",
    "transaction_id":"1900000109201307020305773741",
    "out_trade_no":"2986872580246457300",
    "is_split":"false",
    "is_refund":"false",
    "attach": "",
    "time_end":"20130702175943",
    "transport_fee":"0",
```

```
"product_fee": "1",  
"discount": "0",  
"rmb_total_fee": ""  
}  
}
```

对于详细的订单信息，放在 order\_info 中的 json 数据中，各个字段的含义如下：

ret\_code 是查询结果状态码，0 表明成功，其他表明错误；

ret\_msg 是查询结果出错信息；

input\_charset 是返回信息中的编码方式；

trade\_state 是订单状态，0 为成功，其他为失败；

trade\_mode 是交易模式，1 为即时到帐，其他保留；

partner 是财付通商户号，即前文的 partnerid；

bank\_type 是银行类型；

bank\_billno 是银行订单号；

total\_fee 是总金额，单位为分；

fee\_type 是币种，1 为人民币；

transaction\_id 是财付通订单号；

out\_trade\_no 是第三方订单号；

is\_split 表明是否分账，false 为无分账，true 为有分账；

is\_refund 表明是否退款，false 为无退款，ture 为退款；

attach 是商家数据包，即生成订单 package 时商家填入的 attach；

time\_end 是支付完成时间；

transport\_fee 是物流费用，单位为分；

product\_fee 是物品费用，单位为分；



discount 是折扣价格，单位为分；

rmb\_total\_fee 是换算成人民币之后的总金额，单位为分，一般看 total\_fee 即可。