小游戏道具直购开发指南

1. 道具直购时序图 & 状态图

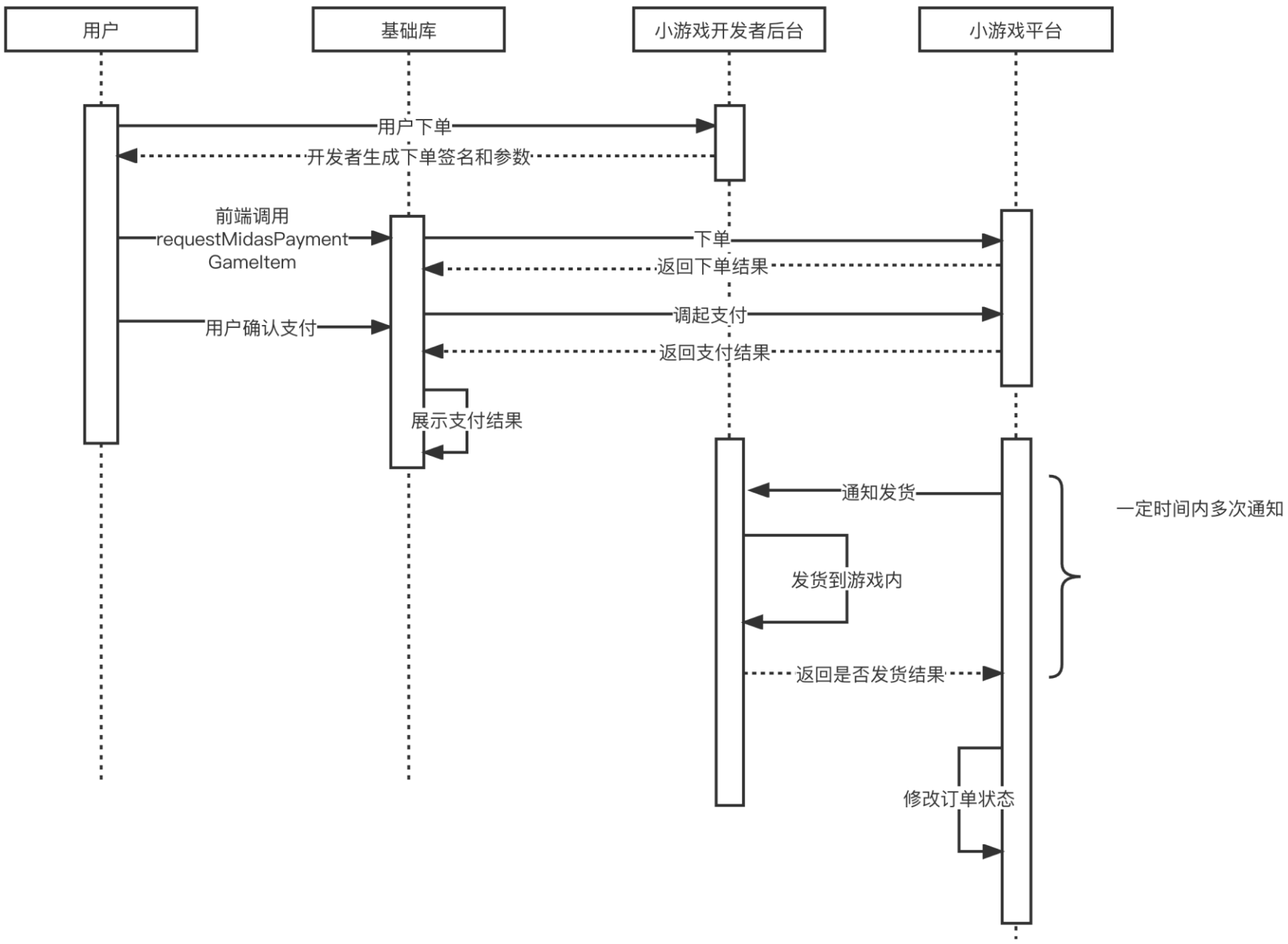

图 1：道具直购时序图

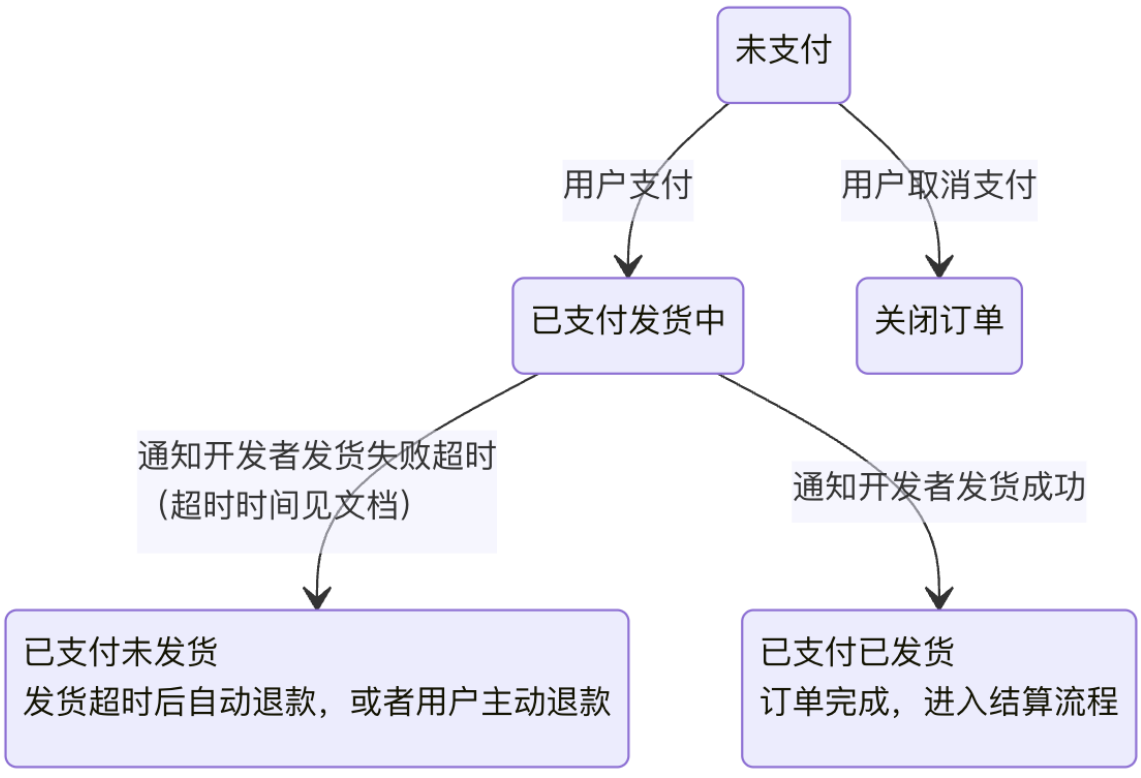

图 2：道具直购状态图

2. 发起米大师支付

wx.requestMidasPaymentGameItem(Object object)

属性	类型	必填	说明
signData	string	是	**支付原串** 具体支付参数见 signData，将数据以 json 格式传递 signData 例子: '{"mode":"goods","offerId":"123","buyQuantity":1,"env":0,"currencyType":"CNY","platform":"android","zoneId":"1","productId":"testproductId","goodsPrice":10,"outTradeNo":"xxxxxx","attach":"testdata"}',
paySig	string	是	**支付签名** pay_sig 参数的签名算法，使用 **“mp-支付基础配置”** 中的 **AppKey** 对支付的请求进行签名，代表请求经过开发者服务端的支付模块发起。签名算法伪代码为： paySig = to_hex(hmac_sha256(appKey,'requestMidasPaymentGameItem' + '&' + signData)) 具体可见 支付签名代码实现
signature	string	是	**用户态签名** signature 参数签名算法参考用户登录态签名 可参考 calc_signature 小游戏虚拟支付 2.0 开发指南（对外）
success	function	否	**接口调用成功的回调函数**，对齐 wx.requestMidasPayment
fail	function	否	**接口调用失败的回调函数**，对齐 wx.requestMidasPayment
complete	function	否	**接口调用结束的回调函数**（调用成功、失败都会执行），对齐 wx.requestMidasPayment

SignData

属性	类型	必填	说明
mode	string	是	**支付的类型**，不同的支付类型有各自额外要传的附加参数。 goods 道具直购
offerId	number	是	**在米大师侧申请的应用 id** mp-支付基础配置中的 offerid
buyQuantity	number	是	**购买数量**
env	number	否	**环境配置** 0 米大师正式环境 1 米大师沙箱环境 默认为 0
currencyType	string	是	**币种** CNY 人民币
platform	string	否	**平台** 默认 android

zoneId	string	否	**分区 ID** 默认分区 id 1
productId	string	是	**道具 ID**
goodsPrice	number	是	**道具单价（分）** 用来校验价格与后台道具价格是否一致，避免用户在业务商城页看到的价格与实际价格不一致导致投诉
outTradeNo	number	是	**业务订单号** 每个订单号只能使用一次，重复使用会失败（极端情况不保证唯一，不建议业务强依赖唯一性）。 要求 32 个字符内，只能是数字、大小写字母、符号 _-|*@组成，不能以下划线(_)开头。 若没有传入，则平台会自动填充一个，并以下划线开头。
attach	string	否	**透传数据** 发货通知时会透传给开发者

示例

```
wx.requestMidasPaymentGameItem({
    signData:
'{"mode":"goods","offerId":"123","buyQuantity":1,"env":0,"currencyType":"CNY","platform":"android","zone
Id":"1","productId":"testproductId","goodsPrice":10,"outTradeNo":"xxxxxx","attach":"testdata"}',
    paySig: 'd0b8bbccbe34ed11549bcfd6602b08711f4acc0965253a949cd6a2b895152f9d',
    signature: 'd0b8bbccbe34ed11549bcfd6602b08711f4acc0965253a949cd6a2b895152f9d',
    success(res, errCode) {
        console.log('pay', res, errCode);
    },
    fail({
        errMsg,
        errCode
    }) {
        console.error(errMsg, errCode)
    }
})
```

第二步：验证消息来自微信服务器

填写完相关配置后点击模拟推送，微信服务器将发送 POST 请求到填写的服务器地址 URL 上，POST 请求 URL 携带参数如下表所示：

参数	描述
signature	微信加密签名，signature 结合了开发者填写的 token 参数和请求中的 timestamp 参数、nonce 参数。
timestamp	时间戳
nonce	随机数
encrypt_type (安全模式特有)	表示加密类型，值固定为 aes
msg_signature (安全模式特有)	消息体签名，用于验证消息体的正确性

- 选择**明文模式**时 POST 请求的 payload 符合道具发货消息协议 JSON 格式，内容参考下文的 JSON 格式的请求示例。
- 选择**安全模式**时 PSOT 请求的 payload 格式实例如下：

```
{            "Encrypt":"XyDesAXZTAZXiZ7/yYhJlp1KARpTBHH8m/k01EetaDe0+46X35la5klYG2D5GlwJb4TDu4brcRDHs64R
b+HMFOWOBsYCGqsOmC58PYcGzZ1Dkrd/ZXFHYLN5AOe4v0q6MkQLraptghbSyoHdtAVAdQe9SExvD8WZGDmUd6HHmCFvJj+0BRH57U6Q
uIEEahgFYSgsTFCLzILvZCxVT4NgzJwDqWv0xNxVPfDZNopqFPTu7XYYAxbDBx2rQHN0lUhfq0NfRXRa5CWMIgJGfxRjv9Nn92buxwxn
dyuqGZsCHu0kCfbxi0ssWY2qCy3gwOFJplEZp+225EgDjqygojAzV75oSkL7iRRybLDS/sOd0dBf9Vt42iTKZYlKxfTE9bQIDBYN8oLN
Iaptz1ButY/k4sJP7KTXvtLf1UZV2Xv4BYYLckVm7HfGiqnEY7T/R4+ApJn4hOZoUqBSRtY7c92NpNFLSl9S+MOgTCQ0PTRnEAqnSrQ9
uyk7MkIldQ23UKyFDnpSz1oQO2JZGf9Ep4ljlOI3P+zcxAGQ9oNDww37FcIWjrnOc51LOE8IUkWBZXHCmzwm82qCmkK/G9l+iY6oEXoB
ekFs3UolKKL/MVwffLqhbGYLjvlKznoPFi72pb4C7uG+9/ldLcOXAYGTRwcMhMEHNw7cMcdp9G3zBwtjzUEgBuPXnGNxh9YG0d5vpAim
5K+oQU5avnSEcuegfKbEcdkNTzYs207Efkma2/wU+jeUgNlc5gO8yCdbwH5EQUhQE+pYS4co4Dt5HDzJTDHEZcF6X+SdC3d+VpVRPrp8
BnG5BE0dnLzD6tVUzgqWyhWJaQAfXlNDXgjQ5arhV28QsC96J7IsTyL2rxxiyN2rBRLabGnmqYtEluijAR86glOUkjmQhxcPl3nDLRKP
Zq/fV+e68x11l5aFlJwj9RSD4aY=",
    "ToUserName":"gh_31b2a1c7e78a"
}
```


开发者需要做以下几点校验：

- 校验签名 signature（校验方法参考：消息推送 | 微信开放文档）。
- 校验 payload 是否符合发货消息协议。

如果选择了加密方式，还需要验证以下两点（校验方法参考：概述 | 微信开放文档）。

- 校验消息体签名 msg_signature。
- 对 Encrypt 字符串进行解密，解密后的明文应该是符合道具发货消息协议的 JSON 字符串，验证所使用的解密函数是否正确。

验证通过后开发者返回如下示例的 JSON 数据，则接入生效，此时页面的测试状态会显示为已通过，否则说明接入失败。

```
{"ErrCode":0,"ErrMsg":"Success"}
```

第三步：配置测试链接

如果开发者有测试需求，可以点击配置测试链接，在该页面填写测试 URL 和 openid 白名单。

- openid 白名单：多个 openid 用;进行分隔，**属于 openid 白名单的发货消息都会发送到所填写的测试 URL。**
- 测试 url：**必须以 http:// 或 https:// 开头，分别支持 80 端口和 443 端口。**

请求参数

字段	类型	说明
ToUserName	String	小游戏原始 ID
FromUserName	String	该事件消息的 openid，道具发货场景固定为微信官方的 openid
CreateTime	Number	消息发送时间
MsgType	String	消息类型，道具发货场景固定为：event
Event	String	事件类型 道具直购（游戏外）场景固定为： minigame_deliver_h5_pay_products 道具直购（游戏内）场景固定为： minigame_deliver_game_pay_products
MiniGame	Object	道具直购发货参数

MiniGame

字段	类型	说明
Payload	String	携带的具体内容，格式为 json，具体内容如下表格 Payload（因为这里需要对消息内容统一签名，所以统一把消息内容设计成 json 格式）
PayEventSig	String	见支付请求签名算法说明 (PayEventSig)

Payload（JSON）

字段	类型	说明
OpenId	String	接收道具的玩家 openid

ErrCode	Number	是	发送状态。0：成功，其他：失败 todo
ErrMsg	String	否	错误原因，用于调试。在 errcode 非 0 的情况下可以返回

XML 格式示例

请求

```
<xml>
    <CreateTime>1583202606</CreateTime>
    <MsgType><![CDATA[event]]></MsgType>
    <Event><![CDATA[minigame_deliver_h5_pay_products]]></Event>
    <MiniGame>

<Payload>{"OpenId":"to_user_openid","OutTradeNo","xxxxxxx","Env":0,"GoodsInfo":{"ProductId":"id_100001",
"ZoneId":"1","OrigPrice":10,"ActualPrice":10,"Quantity":1}}</Payload>
        <PayEventSig>f749f67b751fa80f27ddc0b7c8d2821aeda162ea22b323cd64a2c8056c2736f0</PayEventSig>
    </MiniGame>
</xml>
```

成功返回

```
<xml>
    <ErrCode>0</ErrCode>
    <ErrMsg>Success</ErrMsg>
</xml>
```

失败返回

```
<xml>
    <ErrCode>99999</ErrCode>
    <ErrMsg>internal error</ErrMsg>
</xml>
```

JSON 格式示例

请求

```
{
        "ToUserName": "gh_31b2a1c7e78a",
        "FromUserName": "oUrsf0TSXNtiZjP7JL9UUFiGJzmQ",
         "CreateTime": 1583202606,
    "MsgType": "event",
    "Event": "minigame_deliver_h5_pay_products",
    "MiniGame": {
       "Payload":
"{\"OpenId\":\"to_user_openid\",\"Env\":0,\"GoodsInfo\":{\"ProductId\":\"id_100001\",\"ZoneId\":\"1\",\"
OrigPrice\":10,\"ActualPrice\":10,\"Quantity\":1}}",
       "PayEventSig": "f749f67b751fa80f27ddc0b7c8d2821aeda162ea22b323cd64a2c8056c2736f0"
    }
}
```

成功返回

errmsg	string	错误信息
product_id	string	道具 id
pay_status	number	支付状态 1 未支付 2 已支付
deliver_status	number	发货状态 1 未发货 2 已发货
pay_finish_time	number	支付完成时间
out_trade_no	string	用户订单号
mch_order_no	string	微信支付商户单号
transaction_id	string	交易单号（微信支付订单号）

errcode 的合法值

值	说明
0	请求成功
-1	系统繁忙，此时请开发者稍候再试
90000	订单不存在，即 out_trade_no 对应的订单不存在
90010	signature 签名错误或用户登录态（session_key）已过期
90011	pay_sig 签名错误
90018	参数错误，具体参数见 errmsg 描述

示例

cURL 请求

```
curl -d '{
  "openid": "oUrsfxxxxxxxxx",
  "ts": 1668512806,
  "zone_id": "1",
  "env": 0,
  "out_trade_no": "test_queryorderinfo_1668512806"
}' \
  -H "Content-Type: application/json" \
  -X POST \

'https://api.weixin.qq.com/wxa/game/notifydelivery?access_token=62_so84Zyl5MuPCjXGiR3eb1ysa1lr6aRpcprEnNpZ9ds8676ivjqRn5Zroi2Rxxx9-Yvh2zIl9oEt1hIzK0x2OrMCT5zk8nB_TG98obD_ad3tFPftfFmB7xXtcZv4PVPcbFATZT&signature=a5ec09b5677b5004495ac78eecc0aa78a4aa53c1bfa82850e4dcf1650c8a5d69&pay_sig=e654d5725976945f738caf3485c260b129dc586c797d7616ca15176fd31b5e8b&sig_method=hmac_sha256'
```

成功返回

```
{
  "errcode": 0,
  "errmsg": "ok",
  "out_trade_no": "test_queryorderinfo_1668512806",
  "pay_finish_time":1669364790,
  "product_id":"id_100001",
  "deliver_state":1,
  "pay_state": 1,
  "mch_order_no":"1217752501201407033233368018",
  "transaction_id":"1217752501201407033233368018"
}
```