1. 购买流程

客户端提交购买信息到服务器游戏服务器，游戏服务器把数据“格式化”之后发到支付服务器，支付服务器生成订单，到支付宝或银联“备案”。之后二者都会有反馈数据，支付服务器把反馈的数据拆分，再组合之后发到游戏服务器，游戏服务器再把数据发到客户端，客户端根据数据调起客户端对应的插件，开始支付过程。

1. 协议

C:客户端 GS：游戏服务器 OS：支付服务器 DS:数据服务器

1、C -> GS

购买道具：BuyItemsReq(C->GS)

itemsId：Short 道具编号（见附录）

itemCnt：Short 购买数量

payType:String 支付类型{0:支付宝，1银联}

2、GS –> OS

sumbitTelegram数据格式如下

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <interactiontimesgame version="1.0">  <gameName> gameName </gameName>  <gameVersion>1.0.1</gameVersion>  <user>LoginName</user>  <itemId> itemId </ itemId >  <itemCnt> itemCnt </ itemCnt >  <payType>{0:支付宝，1银联}</payType>  </ interactiontimesgame > |
| user:用户的登录名 |
| **调用方式：**  OrderInfoHttpTransferconnectUnionServer = new OrderInfoHttpTransfer (PayServerUrl,timeOut);  connectUnionServer.sendMsg(sumbitTelegram);  接收回馈：xmlSign  connectUnionServer.getReMeg() |
| **由于在数据传送中使用URLEncoder重新编码，导致从支付服务器获取的信息需要做特别处理。**  **初始方式：**  String recvMsg = connectPayServer.getReMeg();  recvMsg = LanchPayXmlTelegram.lanchPayXml(recvMsg);  这里的recvMsg就是给客户端的调起插件所需的参数。使用的jar包有： **OrderInfoHttpTransfer-1.0.7.jar**  **jdom-2.0.3.jar**  **lanchpay-1.0.jar** |

3、OS -> GS

xmlSign：String

* "85858" ：操作超时，由客户端判断xmlSign
* null:见三，查看错误代码

4、GS -> C

BuyItemsRsp: (GS->C)

\*buyInfo:String {3中xmlSign的值}

说明：在C中根据BuyResult（值为“85858”）来处理购买还是给出超时提示。

5、OS->DS

报文：

|  |
| --- |
| <?xml version='1.0' encoding='UTF-8'?>  <interactiontimes version="1.0">  <processType>00</process>  <gameName>gameName</gameName>  <gameVersion>gameVersion/gameVersion>  <user>user</user>  <itemId>itemId</itemId>  <itemCnt>itemCnt</itemCnt>  </interactiontimes> |

processType:

00: data update ,01 data query , 02 data delete , 03 data insert

1. 在支付宝、银联购买道具中的错误代码ErrorCode和错误信息ErrorMsg

注：xmlSign=null时，GS与OS之间传输的信息检查

|  |  |  |
| --- | --- | --- |
| ErrorCode | ErrorMsg | Reason |
| -21 | 服务器拒绝连接 | OS服务器为打开 |
| -31 | httpState=[respCode] | 服务器没有正常响应数据 |
| -32 | 传输数据异常 | 报文异常 |

获取错误代码： getErrorCode()

获取错误信息： getErrorMsg()