Spadegaming E-Games Common Wallet Specification Document (Chinese Version)

Version 5.2

修订历史

日期	修订版本	修订说明
2013-11-01	Version0.1	初稿
2013-12-12	Version0.2	修改 4.4 transfer, 1: ticketId 改为可选字段, 2: 删除 referenceId 字段;
2013-12-12	Version0.3	增加 4.7 查询用户有效 sessionid，tokenid;
2013-12-12	Version0.4	增加 4.8 查询 Jackpot 彩池;
2014-02-24	Version0.5	1. 附录 4 游戏代码增加新游戏代码 2. API 里所有 acctId 字段长度由 20 改成 50
2014-04-30	Version0.6	附录 4 游戏增加新游戏代码
2014-05-05	Version0.7	增加 4.9 查询游戏列表信息
2014-08-14	Version0.8	1. 增加 4.10 强制退出当前在线用户 2. Transfer Request 增加“cancelId” 3. BetInfo 增加”sequence”
2014-11-01	Version 0.9	1. 删除 4.7 查询用户有效 sessionid，tokenid; 2. 修改“3 API 接口概况”，增加 serialNo 的说明 3. 增加“4.8 查询游戏列表信息”的说明 4. 修改“附录 2 语言代码定义”，增加语言定义 5. 将示例代码移到附录
2014-12-15	Version 1.0	4.1 游戏大厅页面跳转，参数 minigame = true/false 4.6 查询转帐记录，参数 merchantTxId
2014-01-26	Version 1.1	4.4 拿走派彩取消(cancel payout)和派彩取消撤销 (uncancel payout)
2015-03-13	Version 1.2	附录 4 游戏增加新游戏代码, 4.7 查询下注记录, result 加长到 varchar (1024)
2015-05-04	Version 1.3	4.1 游戏大厅页面跳转，参数 mobile= true/false 4.5 查询下注记录，参数 Channel = Mobile/Web 4.8 查询游戏列表信息，参数 mthumbnail
2015-11-01	Version 1.4	1. 增加 18 个新游戏 2. 4.14 查询 Progressive Jackpot 彩池
2015-11-17	Version 1.5	4.11 安装版用户登录
2015-12-07	Version 1.6	1. 增加 3 个新游戏
2015-12-17	Version 1.7	1. 增加 4 个新游戏
2016-01-20	Version 1.8	1.增加 16 个新游戏 2. 查询下注记录，参数 balance
2016-04-10	Version 1.9	4.5 查询下注记录，参数 jpWin = -100.000000
2016-05-30	Version 2.0	1. 4.4 转帐，参数 referenceId = acahsdhas 2. 4.5 查询下注记录，参数 referenceId = acahsdhas 3. 增加 11 个新游戏
2016-10-27	Version 2.1	1. 4.1 增加 menumode 参数 2. 4.8 增加 jackpotCode, jackpotName 参数 3. 增加 11 个新游戏
2017-01-09	Version 2.2	1.增加 17 个新游戏
2017-03-01	Version 2.3	1.增加 8 个新游戏
2017-04-19	Version 2.4	1.增加 2 个新游戏
2017-08-07	Version 2.5	1. 4.5 查询用户下注记录 - categoryId 添加 ‘BN’ – 专用于红包功能 2. 4.4 转账 （request） – 添加 Channel (Web, Mobile, APP-i, APP-A,PC)

		3. 4.6 查询转账记（response）– 添加 Channel (Web, Mobile, APP-i, APP-A,PC) 4. 语言代码定义– 添加 109 Reference No 不存在 5. 4.4 转账 （request）– 添加 Type =7 (Bonus)
2017-08-21	2.6	4.1 游戏大厅页面跳转 参数 lobby=SG
2017-12-11	2.7	4.2 用户授权 AcctInfo 添加'siteId'
2018-01-04	2.8	1. 附录 1 响应代码添加 '112,113' 2. 4.2 用户授权 – 添加注意事项 Acct ID 格式 3. 4.4 转账– Transfer request 添加 'specialGame(type, count, sequence)', 'refTicketIds'
2018-08-31	2.9	1. 添加 4.10 创建 token API 和 4.11 单号详细 2. 去除 4.10 安装板用户登入 3. getGames 请求信息添加 'currency'
2019-03-27	3.0	添加响应消息代码
2019-05-14	3.1	4.3 添加 gameCode 参数
2019-06-18	3.2	1. 4.15 添加 createAcct 接口 2. 4.16 添加 checkAcct 接口 3. 添加响应消息代码 4. 添加汇率 INR 和 BDT
2019-09-12	3.3	4.6 转账记录 request 添加 transferId
2019-09-17	3.4	4.5 Bet Info 添加 sequence 4
2019-10-02	3.5	1.附录 3 增加新货币代码 ID2 和 VN2 2.更改 4.5 BetIp 数据类型为 Varchar(50)
2019-12-16	3.6	1. 去除 createAcct 接口 2. 添加 C playerDailySumByGame 接口
2020-01-10	3.7	1. getBetHistory-控制查询限制
2020-02-27	3.8	1. 移除内文附录 4 2. 4.6 查询转账记 （response） – 添加 Type 类别
2020-06-10	3.9	1.添加 4.1 exitUrl
2020-09-14	4.0	1. 添加 4.5 查询用户下注记录备注
2020-10-20	4.1	1. 添加 Russia 和 Turkey 语言 2.移除 Digest 在 1 协议 (Protocol) 3.删除 minigame 参数 在 4.1 游戏大厅页面跳转
2021-02-16	4.2	1. 更新 status 参数说明 2. 添加 fullScreen 功能在游戏大厅页面跳转
2021-05-10	4.3	1. 在 header 中 添加 Digest (强化) 2. 更新货币清单 附录 3 货币代码定义
2021-11-25	4.4	1. 优化响应代码定义 1. 移除不必要的 code 2. 添加有使用的 Code 2. Protocol 调整 1. Digest 改去 Mandatory 3. 添加货币-AZN
2021-12-10	4.5	1. 將 Transfer 分成兩個部分 1.1 4.4 Transfer 與 4.5 Bonus Transfer
2022-03-18	4.6	添加新的 API：4.9 jackpotPoolAll
2022-06-02	4.7	1.添加新的 API 4.16 设置 Youtuber

2022-09-12	4.8	1. getGames - 附加新参数: language 2. Transfer -附加新参数: gameFeature
2022-10-03	4.9	1. 添加新的货币：Appendix 3
2022-11-01	5.0	1. getBetHistory -附加新参数: gameFeature
2022-12-14	5.1	1. 添加语言 ：法文, 西班牙文 2. 添加新的货币：Appendix 3 3. 移除不必要的参数 categoryId :BC
2023-02-22	5.2	1. 添加新的货币：Appendix 3 2. checkAcct - 附加新参数: includeType, acctType 3. 优化响应代码定义 - 移除不必要的 Code

目录

1. 协议 (Protocol)

Merchant API 使用 HTTP(s)来发送数据和返回数据，目前支持的数据格式包括 XML 和 JSON。关于 JSON 的信息可参考(http://www.json.org/)

在 HTTP Header 部分，除了 HTTP 协议必须的 Header 外，还有使用如下的 Header：

HTTP Header	是否必须	示例	说明
API	是	authorize	服务接口名称
DataType	是	JSON	数据格式 XML 或 JSON
Digest	是	MD5(Data+securityKey)	Digest 计算时所需的数据 - securityKey 将由供应商(SG)提供
Accept-Encoding	否	gzip, deflate	标识是否支持返回数据 zip 压缩
Content-Encoding	否	gzip	标识提交的数据已使用 zip 压缩
Accept-Language	否	en	默认返回消息的国际化的语言

关于字符编码， 在对所有涉及 text <=> byte[] 进行 encode/decode 处理时，请显式指定编码格式为 UTF-8，请注意不能有空格与换行符号

```
Public abstract class DigestUtils
{
public static String digest(byte[] input) {
byte[] digest = Digest.md5().compute( input );
return Hex.encodeHexString( digest );
  }

public static String digest(byte[] input, byte[]secretKey) {
int bodySize=input.length,keySize=secretKey.length;
byte[] buffer= new byte[bodySize+keySize];
System.arraycopy(input,0,buffer,0,bodySize);
System.arraycopy(secretKey,0,buffer,bodySize,keySize);
byte[] digest = Digest.md5().compute( input );
return Hex.encodeHexString(digest);
  }
}
```

2. 名词解释

1. Game Provider：游戏提供者，提供游戏的平台，管理玩家和商户的数据
2. Merchant：商户，每个注册的商户，都有其各自的配置和数据保存在游戏提供者的系统中
3. 接口使用者：API 接口的调用者
4. 接口提供者：API 接口的服务提供者
5. Acct ID：用户唯一标识 ID，游戏玩家

3. API 接口概况

在 API 的使用时，由调用者按接口定义的消息，生成相应的数据，发送到接口提供者提供的 URL 上，接口提供者处理完成后，将相应的消息返回回来。

一般的接口定义为：xxxResponse xxxx(xxxRequest request)
xxxRequest 为 API 相应的请求消息，xxxResponse 为 API 相应的响应消息

所有的请求消息，都包括：

名称	数据类型	示例	说明
serialNo	Varchar(50)	20120722224255982841	用于标识消息的序列，由调用者生成
merchantCode	Varchar(10)	SPADE	标识商户 ID

所有的响应消息，都包括：

名称	数据类型	示例	说明
serialNo	Varchar(50)	20120722224255982841	用于标识消息的序列，由调用者生成
code	Integer	0	处理结果代码定义 code
msg	Varchar(128)	success	处理结果描述
merchantCode	Varchar(10)	SPADE	标识商户 ID

其中，code 用来定义接口调用的处理结果，详细的 code 的定义，请参照附录 1 “响应消息代码定义”。

特别注意：serialNo 是消息标识，同一个 serialNo 只会有一个 request/response，请确保每次生成的 serialNo 是唯一的值。建议使用 GUID/UUID，或是日期+时间+随机数的字符串。

4.API 接口定义(Service Interface)

4.1 游戏大厅页面跳转

使用场景：当游戏玩家在进入 merchant 网站后，通过网页上的 Goto Game 或 Real Money 按钮，将玩家页面转到游戏提供商的游戏。

接口名：Redirect To Game Lobby
接口提供者：Game Provider
接口使用者：Merchant

名称	数据类型	必填	示例	说明
acctId	Varchar(50)	是	TESTPLAYER1	游戏玩家 ID
token	Varchar(80)	是	fe1a85adc54545d2963b661a22d09c9e	
language	Varchar(10)	否	en_US	语言
game	Varchar(10)	是	S-DG02	游戏代码
fun	Varchar(10)	否	true	是否试玩
mobile	Varchar(10)	否	true/false	是否手机
menumode	Varchar(10)	否	On/off	目录开关
exitUrl	Varchar(150)	否	http://www.baidu.com	将网址重定为退出按钮
fullScreen	Varchar(10)	否	on/off	fullscreen 开关

1. 详细的 language 的定义，请参照附录 2“语言代码定义”。
2. 如果 play for fun，需要传入 acctId & token， fun=true
3. 如果 mobile，需要传入 mobile=true
4. 如果不要 game menu，需要传入 menumode=off
5. 如果不要 full screen，需要传入 fullScreen=off\
6. 如果只需进入要大厅，不需要带入游戏代码 game=S-XXXX，直接带入 lobby=SG

提醒 ：此功能并不是用来调用的，是介绍游戏连结里面的参数功用

以下为游戏连结参考:

Game Lobby Redirect URL and Parameters(real money):

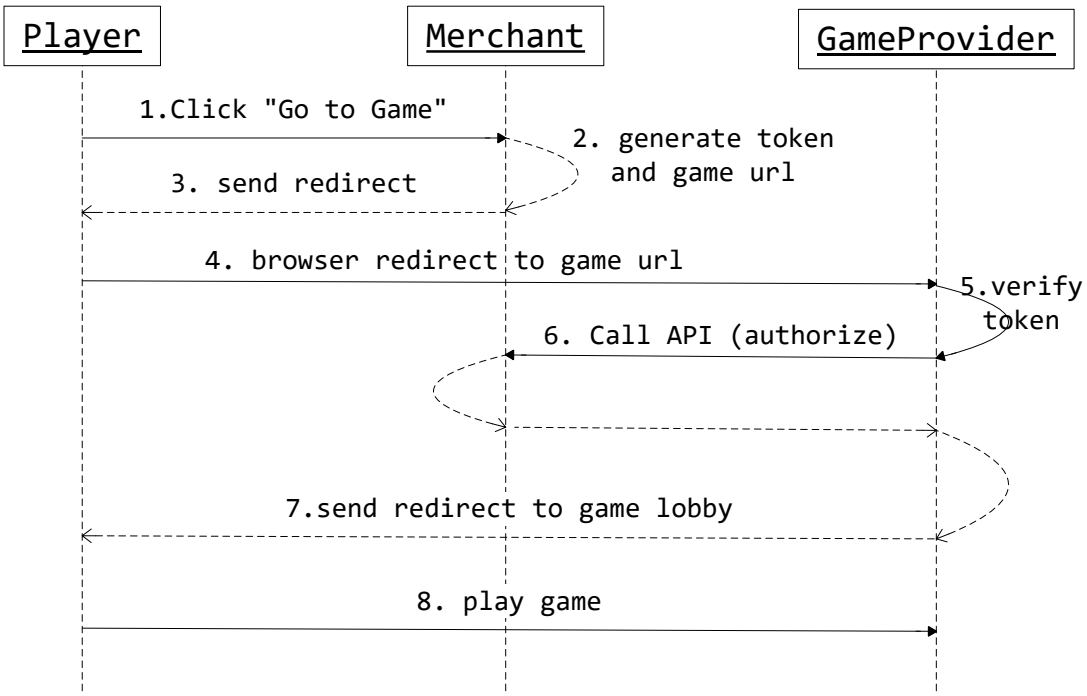
http://portal.e-games.com/auth/?acctId=TESTPLAYER1&language=en_US
&token=fe1a85adc54545d2963b661a22d09c9e&game=S-DG02&mobile=true&menumode=on
&fullScreen=on&exitUrl=http://www.baidu.com

Game Lobby Redirect URL and Parameters(play for fun):

http://portal.e-games.com/auth/?acctId=TESTPLAYER1&language=en_US&
token=fe1a85adc54545d2963b661a22d09c9e&game=S-DG02&fun=true&mobile=true
&menumode=on&exitUrl=http://www.baidu.com

Game Lobby Redirect URL and Parameters(Lobby):

http://portal.e-games.com/auth/?acctId=TESTPLAYER1&language=en_US
&token=fe1a85adc54545d2963b661a22d09c9e&lobby=SG&mobile=true



4.2 用户授权

使用场景：当游戏提供商收到从 merchant 网站跳转过来的请求后，通过本 API 验证请求是否合法。
接口名：authorize
接口提供者：Merchant
接口使用者：Game Provider

请求消息定义 `AuthorizeRequest`：

名称	数据类型	必填	示例	说明
acctId	Varchar(50)	是	TESTPLAYER1	游戏玩家 ID
token	Varchar(80)	是	fe1a85adc54545d2963b661a22d09c9e	由 Merchant 端生成
language	Varchar(10)	否	en_US	语言
merchantCode	Varchar(10)	是	SPADE	标识商户 ID
serialNo	Varchar(50)	是	20120722224255982841	用于标识消息的序列，由调用者生成

响应消息定义 `AuthorizeResponse`：

名称	数据类型	示例	说明
acctInfo	AcctInfo		AcctInfo 类

AcctInfo 定义 `AcctInfo`：

名称	数据类型	必填	示例	说明
acctId	Varchar(50)	是	TESTPLAYER1	用户标识 ID
userName	Varchar(15)	否	TestPlayer	用户名称
currency	Char(3)	是	USD	货币的 ISO 代码
balance	Decimal(20,4)	是	0	用户当前余额
siteId	Varchar (10)	否	SITE_USD1	商户的站点

别注意：

1. 详细的货币代码定义，请参照附录 3“货币代码定义”
2. Acct ID 格式：[a-zA-Z0-9_-@]{1,50}. 例子: TestPlayer_1

Authorize 请求和响应消息示例

JSON AuthorizeRequest：

```
{
   "acctId": "TESTPLAYER1",
   "language": "en_US",
   "merchantCode": "SPADE",
   "token": "fe1a85adc54545d2963b661a22d09c9e",
   "serialNo": "20120722224255982841"
}
```

XML AuthorizeRequest：

```
<AuthorizeRequest>
   <serialNo>20120723161423158480</serialNo>
   <token>fe1a85adc54545d2963b661a22d09c9e</token>
   <merchantCode>SPADE</merchantCode>
   <acctId>TESTPLAYER1</acctId>
   <language>en</language>
</AuthorizeRequest>
```

JSON AuthorizeResponse：

```
{
   "acctInfo": {
      "acctId": "TESTPLAYER1",
      "balance": 0,
      "userName": "TestPlayer",
      "currency": "USD"
      "siteId": "SITE_USD1"
   },
   "merchantCode": "SPADE",
   "msg": "success",
   "code": 0,
   "serialNo": "20120722224255982841"
}
```

XML AuthorizeResponse:

```
<AuthorizeResponse>
   <serialNo>20120722224255982841</serialNo>
   <code>0</code>
   <msg>success</msg>
   <merchantCode>SPADE</merchantCode>
   <acctInfo>
      <acctId>TESTPLAYER1</acctId>
      <userName>TestPlayer</userName>
      <currency>USD</currency>
      <balance>0</balance>
      <siteId>SITE_USD1</siteId>
   </acctInfo>
</AuthorizeResponse>
```

4.3 查询用户余额

使用场景：查询用户余额
接口名：getBalance
接口提供者：Merchant
接口使用者：Game Provider

请求消息定义 `GetBalanceRequest`：

名称	数据类型	必填	示例	说明
acctId	Varchar(50)	是	TESTPLAYER1	游戏玩家 ID
gameCode	Varchar(10)	否	S-DG02	游戏代码

响应消息定义 `GetBalanceResponse`：

名称	数据类型	必填	示例	说明
acctInfo	AcctInfo	是	AcctInfo 类	acctInfo

AcctInfo 定义 `AcctInfo`：

名称	数据类型	必填	示例	说明
acctId	Varchar(50)	是	TESTPLAYER1	用户标识 ID
userName	Varchar(15)	否	TestPlayer	用户名称
currency	Char(3)	是	USD	货币的 ISO 代码
balance	Decimal(20,4)	是	0	用户当前余额
siteId	Varchar (10)	否	SITE_USD1	商户的站点

GetBalance 请求和响应消息示例:

JSON GetBalanceRequest：

```
{
    "acctId":"TESTPLAYER1"
    "merchantCode":"SPADE",
    "serialNo":"20120802152140143938"
    "gameCode":"S-DG02"
}
```

XML GetBalanceRequest:

```
<GetBalanceRequest>
    <serialNo>20120802152140143938</serialNo>
    <merchantCode>TEST</merchantCode>
    <acctId>TESTPLAYER1</acctId>
    <gameCode>S-DG02</gameCode>
</GetBalanceRequest>
```

JSON GetBalanceResponse：

```
{
    "acctInfo": {
        "userName": "TestPlayer1",
        "currency": "USD",
        "acctId": "TESTPLAYER1",
        "balance": 1000
    },
    "merchantCode": "SPADE",
    "msg": "success",
    "code": 0,
    "serialNo": "20120802152140143938"
}
```

XML GetBalanceResponse:

```
<GetBalanceResponse>
    <serialNo>20120802152140143938</serialNo>
    <code>0</code>
    <msg>0</msg>
    <merchantCode>TEST</merchantCode>
    <acctInfo>
          <acctId>TESTPLAYER1</acctId>
          <userName>TestPlayer1</userName>
          <currency>USD</currency>
          <balance>1000.0000</balance>
    </ acctInfo >
</GetBalanceResponse>
```

4.4 转帐

使用场景：

1. 用户 place bet 时，由 Game Provider 将金额信息传入 merchant，merchant 将相应的金额从用户的余额中扣除；
2. 系统 payout 时，由 Game Provider 将金额信息传入 merchant，merchant 将相应的金额存入用户的余额

接口名：transfer
接口提供者：Merchant
接口使用者：Game Provider

请求消息定义 `TransferRequest`：

名称	数据类型	必填	示例	说明
transferId	Varchar(50)	是	a3b0c9dd1ab041	标识转帐流水
acctId	Varchar(50)	是	TESTPLAYER1	用户标识 ID
currency	Char(3)	是	USD	货币的 ISO 代码
amount	Decimal(20,9)	是	1000	转帐金额,无论何种操作，金额为大于等于 0
type	Int	是	1	1 = 下注(place bet) 2 = 取消下注(cancel bet) 4 = 派彩(payout)
channel	Varchar(10)	是	Mobile/Web/APP-i/APP-A/PC	注单来自手机或网
gameCode	Varchar(10)	是	S-DG02	游戏代码
ticketId	Varhcar(20)	否	641482277	
referenceId	Varchar(50)	否	a8ab9cc8f1a277de	Transfer id 的参考 ID
specialGame	SpecialGame		SpecialGame Type	只有在获得 bonus, free, bonusfree, freebonus 时会加入 request 发送
refTicketIds	Varchar(2048)	否	[120001, 120002, 120003]	Reference ticket ids 专属有爆炸功能的游戏使用
gameFeature	Varchar(50)	否	BUY-8	仅在玩家触发“购买功能”时可用。 建议：判断前缀 BUY，只要一个 gameFeature 包含 BUY 前缀，就是 “Buy Feature”

SpecialGame type 定义:

Name	Data Type	Mandatory	Example	Description
type	Varchar(20)	否	Free	分辨特别游戏种，例子: bonus, free, bonusfree, freebonus
count	Int	否	10	玩家所得特别游戏的总数量
sequence	Int	否	1	玩家已完成的特别游戏数量

specialGame & refTicketIds only will pass during type = 4, payout

特别注意：

Type 4: Payout

1. 如果 merchant 收到转帐流水发现之前已经处理成功，则直接回應 code 0 让此交易状态成功即可，并不需要做任何的动作(调整玩家的钱包金额)

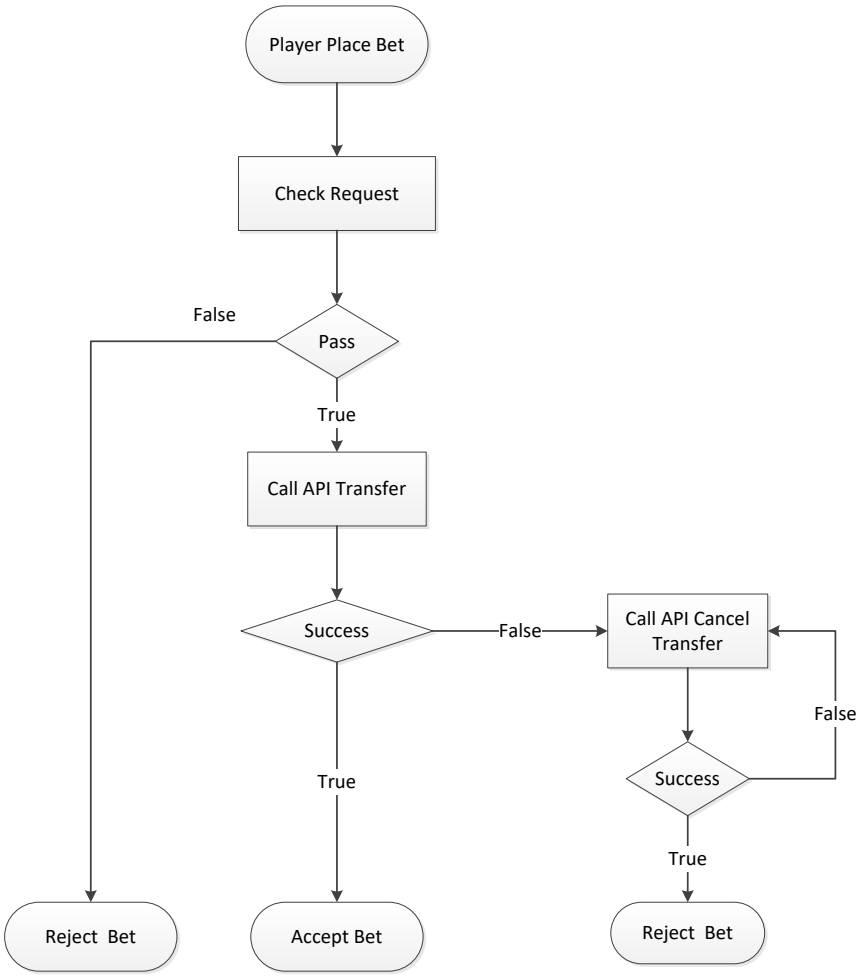
Type 2: Cancel Bet

1. 如果参考 ID 重复或没有找到，商家应该返回我们错误代码：关于此部分，回应 0 或 109. 除了此错误代码，提供商将继续重新发送参考 ID。
2. 如果我司没有收到 placeBet 的回应亦即 time out 状态时，会发送 cancel bet
3. 当玩家进入鱼机游戏的鱼场时，没有进行任何的下注就离开游戏，届时我司会发送 cancel bet

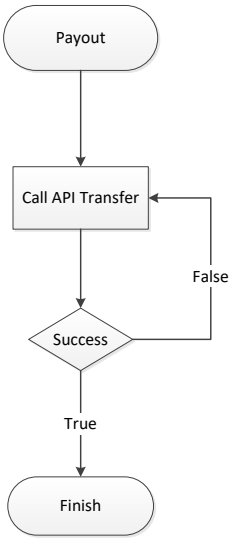
响应消息定义 `TransferResponse`：

名称	数据类型	必填	示例	说明
transferId	Varchar(50)	是	a3b0c9dd1ab041	请求转帐流水
merchantTxId	Varchar(20)	否	498036704	Merchant 处理的交易流水号
acctId	Varchar(50)	是	TESTPLAYER1	用户标识 ID
balance	Decimal(20,4)	是	1000	处理完成后用户余额

Place Bet Flow:



Payout Flow:



Transfer 请求和响应消息示例:

JSON TransferRequest：

```
{
   "acctId": "TESTPLAYER1",
   "transferId": "0ab9bdca06c14811b24653468e60988",
   "currency": "USD",
   "amount": 10,
   "type": 4,
   "ticketId":"234950357",
   "channel": "Web",
   "gameCode": "S-DG02",
   "merchantCode": "SPADE",
   "serialNo": "20120722231413699735",
   "referenceId": "dfffdca06c14811b24653468e60",
   "specialGame":
   {
     "type": "Free",
     "count": 10,
     "sequence": 1
   },
   "refTicketIds": ["120001, 120002, 120003"]
}
```

XML TransferRequest：

```
<TransferRequest>
    <serialNo>20120722231413699735</serialNo>
    <merchantCode>SPADE</merchantCode>
    <transferId>0ab9bdca06c14811b24653468e60988</transferId>
   <acctId>TESTPLAYER1</acctId>
   <currency>USD</currency>
   <amount>10</amount>
   <type>4</type>
   <ticketId>234950357</ticketId>
   <channel>Web</channel>
   <gameCode>S-DG02</gameCode>
   <referenceId>ca06c14811b24653468e60</referenceId>
   <specialGame>
          <type>Free</type>
          <count>10</count>
          <sequence>1</sequence>
   </specialGame>
   <ref_ticket_ids>[120001, 120002, 120003]</ref_ticket_ids>
</TransferRequest>
```

JSON TransferResponse：

```
{
    "transferId": "0ab9bdca06c14811b24653468e609838",
    "merchantCode": "SPADE",
     "merchantTxId": "20130813014319279367",
    "acctId": "TESTPLAYER1",
    "balance": 1050,
    "msg": "success",
    "code": 0,
    "serialNo": "20120722231413699735"
}
```

XML TransferResponse：

```
<TransferResponse>
    <serialNo>20120722231413699735</serialNo>
    <code>0</code>
    <msg>success</msg>
    <merchantCode>SPADE</merchantCode>
    <transferId>0ab9bdca06c14811b24653468e60988</transferId>
  <merchantTxId>20130813014319279367</merchantTxId>
  <acctId>TESTPLAYER1</acctId>
  <balance>1050</balance>
</TransferResponse>
```

4.5 Bonus 转帐

使用场景：

1. 使用于活动 promotion，当商户举办活动的时候，供应商就会传送 type 7
2. 当供应商传送 type 7 时，商户必须调整(增加)玩家的钱包金额
3. 商户必须调整系统来接收 type 7

接口名：transfer
接口提供者：Merchant
接口使用者：Game Provider

请求消息定义 `TransferRequest`：

名称	数据类型	必填	示例	说明
transferId	Varchar(50)	是	a3b0c9dd1ab041	标识转帐流水
acctId	Varchar(50)	是	TESTPLAYER1	用户标识 ID
currency	Char(3)	是	USD	货币的 ISO 代码
amount	Decimal(20,9)	是	1000	转帐金额,无论何种操作，金额为大于等于 0
type	Int	是	7	7=红包（Bonus）
channel	Varchar(10)	是	Mobile/Web/APP-i/APP-A/PC	注单来自手机或网
gameCode	Varchar(10)	是	S-DG02	游戏代码
ticketId	Varhcar(20)	否	641482277	
roundId	Varchar(20)	否	432	Promotion’s 的 Id
siteId	Varchar(20)	否	DEMO1	客戶的 site Id

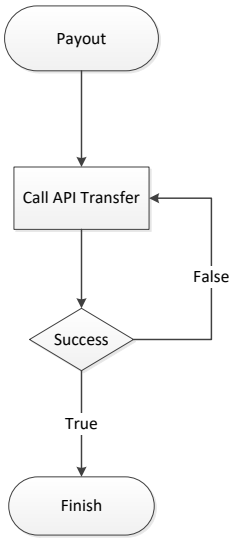
特别注意：

1. Type 7 的部分因為沒有傳送 placebet 因此不會傳送 referenceId
2. roundId 与 siteId 的部分会根据商户的需求传送

响应消息定义 `TransferResponse`：

名称	数据类型	必填	示例	说明
transferId	Varchar(50)	是	a3b0c9dd1ab041	请求转帐流水
merchantTxId	Varchar(20)	否	498036704	Merchant 处理的交易流水号
acctId	Varchar(50)	是	TESTPLAYER1	用户标识 ID
balance	Decimal(20,4)	是	1000	处理完成后用户余额

Bonus flow



Transfer 请求和响应消息示例:

JSON TransferRequest：

```
{
    "acctId": "TESTPLAYER1",
    "transferId": "a3b0c9dd1ab041",
    "currency": "USD",
    "amount": 10,
    "type": 7,
    "ticketId": "234950357",
    "channel": "Web",
    "gameCode": "B-FS02",
    "merchantCode": "SPADE",
    "serialNo": "20120722231413699735",
    "referenceId": ""
}
```

XML TransferRequest：

```
<TransferRequest>
   <serialNo>20120722231413699735</serialNo>
   <merchantCode>SPADE</merchantCode>
   <transferId>a3b0c9dd1ab041</transferId>
   <acctId>TESTPLAYER1</acctId>
   <currency>USD</currency>
   <amount>10</amount>
   <type>7</type>
   <ticketId>234950357</ticketId>
   <channel>Web</channel>
   <gameCode>B-FS02</gameCode>
   <referenceId />
</TransferRequest>
```

JSON TransferResponse：

```
{
    "transferId": "a3b0c9dd1ab041",
    "merchantCode": "SPADE",
    "merchantTxId": "20130813014319279367",
    "acctId": "TESTPLAYER1",
    "balance": 1050,
    "msg": "success",
    "code": 0,
    "serialNo": "20120722231413699735"
}
```

XML TransferResponse：

```
<TransferResponse>
   <serialNo>20120722231413699735</serialNo>
   <code>0</code>
   <msg>success</msg>
   <merchantCode>SPADE</merchantCode>
   <transferId>a3b0c9dd1ab041</transferId>
   <merchantTxId>20130813014319279367</merchantTxId>
   <acctId>TESTPLAYER1</acctId>
   <balance>1050</balance>
</TransferResponse>
```

4.6 查询用户下注记录

使用场景：查询下注记录
接口名：getBetHistory
接口提供者：Game Provider
接口使用者：Merchant

请求消息定义 GetBetHistoryRequest：

名称	数据类型	必填	示例	说明
beginDate	Char(15)	是	20120720T230043	开始时间，格式为 yyyymmdd’T’24hhmmss
endDate	Char(15)	是	20120722T225043	截止时间,格式为 yyyymmdd’T’24hhmmss
pageIndex	Integer	是	1	页码
merchantCode	Varchar(10)	是	SPADE	标识商户 ID
serialNo	Varchar(50)	是	201207222242559 82841	用于标识消息的序列，由调用者生成

响应消息定义 GetBetHistoryResponse：

名称	数据类型	必填	示例	说明
resultCount	Integer	是	100	总共的记录数
pageCount	Integer	是	1	总共的页数
list	BetInfo[]	是		BetInfo 的数组

BetInfo 定义 `BetInfo`：

名称	数据类型	必填	示例	说明
ticketId	Varchar(20)	是	641482277	下注单号
acctId	Varchar(50)	是	TESTPLAYER1	用户标识 ID
ticketTime	Char(15)	是	20120722T225417	下注时间
categoryId	Varchar(10)	是	SM or TB or AD or BN or FH	游戏种类
gameCode	Varchar(10)	是	SS-GD02	游戏代码
currency	Char(3)	是	USD	货币 ISO 代码
betAmount	Decimal(20,4)	是	20	下注金额
result	Varchar(1024)	否	23415	结果
winLoss	Decimal(20,4)	是	-20	用户输赢
jackpotAmount	Decimal(20,4)	是	0	
betIp	Varchar(50)	是	8.8.8.8	用户下注 IP
luckyDrawId	Long	否	10	
roundId	Integer	是	1823	游戏 log ID
sequence	Integer	是	0,1,2,3,4	0 = No jackpot win
channel	Varchar(10)	是	Mobile/Web	注单来自手机或网
Balance	Decimal(18,6)	是	234.000000	上轮余额
jpWin	Decimal(18,6)	是	-100.000000	积宝赢额
referenceId	Varchar(50)	是	a8ab9cc8f1a277de	相关联的转帐流水 payout 关联 place bet 的转帐 cancel 关联 place bet 的转帐
gameFeature	Varchar(50)	否	BUY-20	只在玩家触发“购买功能”时返回

特别注意：

1. 注单是以游戏派奖时间为准；拉取当前时间 3 分钟之前 数据；建议拉取区间为 1-5 分钟，最大不能超过 2 小时。

2. 该 API 具有拉取注单限制，每次拉取注单只可以单一请求。当前请求完成后才可以拉取下一个注单。否则，它将返回代码: 101 USER_CALL_LIMITED

GetBetHistory 请求和响应消息示例:

JSON GetBetHistoryRequest：

```
{
   "beginDate": "20120721T175139",
   "endDate": "20120723T174139",
   "pageIndex": 1,
   "merchantCode": "SPADE",
   "serialNo": "20120723175139440637"
}
```

XML GetBetHistoryRequest：

```
<GetBetHistoryRequest>
   <serialNo>20120723175139440637</serialNo>
   <merchantCode>SPADE</merchantCode>
   <beginDate>20120721T175139</beginDate>
   <endDate>20120723T174139</endDate>
   <pageIndex>1</pageIndex>
</GetBetHistoryRequest>
```

JSON GetBetHistoryResponse：

```
{
   "list": [{
      "ticketId": 633161396,
      "acctId": "TESTPLAYER1",
      "categoryId": "SM",
      "gameCode": "SS-GD02",
      "ticketTime": "20130908T161044",
      "betIp": "8.8.8.8",
      "betAmount": 20,
      "winLoss": 30,
      "result": "23412",
      "jackpotAmount": 1.5,
      "luckyDrawId": 0,
      "completed": True,
      "roundId": 1879,
      "sequence": 0,
      "channel": web,
      "balance": 234.000000,
      "jpWin": 0.0,
      "referenceId": "arwwew232ghfg4dfsdfsdwr2",
      "gameFeature": "BUY-20"
   }],
   "resultCount": 1,
   "pageCount": 1,
   "merchantCode": "SPADE",
   "msg": "success",
   "code": 0,
   "serialNo": "20120723175139440637"
}
```

XML GetBetHistoryResponse：

```
<GetBetHistoryResponse>
  <serialNo>20120723175139440637</serialNo>
  <code>0</code>
  <msg>success</msg>
  <merchantCode>SPADE</merchantCode>
  <resultCount>1</resultCount>
  <pageCount>1</pageCount>
  <list>
    <BetInfo>
      <ticketId>633161396</ticketId>
      <acctId>TESTPLAYER1</acctId>
      <categoryId>SM</categoryId>
      <gameCode>SS-GD02</gameCode>
      <ticketTime>20130908T161044</ticketTime>
      <betIp>8.8.8.8</betIp>
      <betAmount>20</betAmount>
      <winLoss>30</winLoss>
      <jackpotAmount>1.5</jackpotAmount>
      <result>23412</result>
      <luckyDrawId>0</luckyDrawId>
      <completed>True</completed>
      <sequence>0</sequence>
      <roundId>1387</roundId>
      <channel>Web</channel>
      <balance>234.000000</balance>
      <jpWin>0.0</jpWin>
      <referenceId>arwwew232ghfg4dfsdfsdwr2</referenceId>
      <gameFeature>BUY-20</gameFeature>
    </BetInfo>
  </list>
</GetBetHistoryResponse>
```

4.7 查询转帐记录

使用场景：查询转帐状态，供核对转帐数据
接口名：transferHistory
接口提供者：Game Provider
接口使用者：Merchant

请求消息定义 `TransferHistoryRequest`：

名称	数据类型	必填	示例	说明
beginDate	Char(15)	是	20120720T230043	开始时间，格式为 yyyymmdd'T'24hhmmss
endDate	Char(15)	是	20120722T225043	截止时间,格式为 yyyymmdd'T'24hhmmss
pageIndex	Integer	是	1	页码
transferId	Varchar(50)	否	a3b0c9dd1ab041	标识转帐流水

响应消息定义 `TransferHistoryResponse`：

名称	数据类型	必填	示例	说明
resultCount	Integer	是	100	总共的记录数
pageCount	Integer	是	1	总共的页数
list	TransferInfo[]	是		TransferInfo 的数组

TransferInfo 定义 `TransferInfo`：

名称	数据类型	必填	示例	说明
transferId	Varchar(50)	是	a3b0c9dd1ab041	标识转帐流水
acctId	Varchar(50)	是	TESTPLAYER1	用户标识 ID
ticketId	Varchar(20)	是	641482277	
transferTime	Char(15)	是	20120722T225043	转帐时间
merchantTxid	Varchar(20)	是	D141215030051fmowg	商家转帐 Id
type	Int	是	1	1 = 下注(place bet) 2 = 取消下注(cancel bet) 4 = 派彩(payout) 7=红包（Bonus）
amount	Decimal(20,4)	是	1000	转帐金额
status	Int	是	0	状态，收到的响应代码 1=成功; 0= 失败
channel	Varchar(10)	是	Mobile/Web/APP-i/APP-A/PC	注单来自手机或网

TransferHistory 请求和响应消息示例:

JSON TransferHistoryRequest：

```
{
   "beginDate": "20120721T175139",
   "endDate": "20120723T174139",
   "pageIndex": 1,
   "merchantCode": "SPADE",
   "serialNo": "20120723175139440637",
   "transferId": "667706462"
}
```

XML TransferHistoryRequest：

```
<TransferHistoryRequest>
   <serialNo>20120723175139440637</serialNo>
   <merchantCode>SPADE</merchantCode>
   <beginDate>20120721T175139</beginDate>
   <endDate>20120723T174139</endDate>
   <pageIndex>1</pageIndex>
   <transferId>667706462</transferId>
</TransferHistoryRequest>
```

JSON TransferHistoryResponse：

```
{
   "list": [{
      "transferId": "667706462",
      "acctId": "TESTPLAYER1",
      "ticketId": "749084635",
      "type": 1,
      "amount": 20,
      "transferTime": "20130908T151100",
      "merchantTxId": "D141215030051fmowg",
      "status": 0,
      "channel": "Web"
   }],
   "resultCount": 1,
   "pageCount": 1,
   "merchantCode": "SPADE",
   "code": 0,
   "msg": "success",
   "serialNo": "20120726214956563472"
}
```

XML TransferHistoryResponse：

```
<TransferHistoryResponse>
   <serialNo>20120726214956563472</serialNo>
   <code>0</code>
   <msg>success</msg>
   <merchantCode>SPADE</merchantCode>
   <resultCount>1</resultCount>
   <pageCount>1</pageCount>
   <list>
     <TransferInfo>
       <transferId>667706462</transferId>
       <acctId>TESTPLAYER1</acctId>
       <ticketId>749084635</ticketId>
       <type>1</type>
       <amount>20</amount>
       <transferTime>20130908T151100</transferTime>
       <merchantTxId>D141215030051fmowg</merchantTxId>
       <status>0</status>
       <channel>Web</channel>
     </TransferInfo>
   </list>
</TransferHistoryResponse>
```

4.8 查询 Jackpot 彩池

使用场景：查询 Jackpot 金额信息
接口名：jackpotPool
接口提供者：Game Provider
接口使用者：Merchant

请求消息定义 `JackpotPoolRequest`：

名称	数据类型	必填	示例	说明
currency	Char(3)	是	USD	货币的 ISO 代码

详细的货币代码定义，请参照附录 3“货币代码定义”
如果货币是空，那就会显示所有货币的积宝

响应消息定义 `JackpotPoolResponse`：

名称	数据类型	必填	示例	说明
list	JackpotPoolInfo[]	是		JackpotPoolInfo 类

JackpotPoolInfo 定义 `JackpotPoolInfo`：

名称	数据类型	必填	示例	说明
code	Varchar(20)	是	Grand	Jackpot Code
name	Varchar(50)	否	Grand Jackpot	Jackpot 名称
amount	Decimal(20,2)	是	3844.08	彩池金额

JackpotPool 请求和响应消息示例:

JSON JackpotPoolRequest:

```
{
   "currency": "USD",
   "merchantCode": "SPADE",
   "serialNo": "20120802152140143938"
}
```

XML JackpotPoolRequest:

```
<JackpotPoolRequest>
   <serialNo>20120802152140143938</serialNo>
   <merchantCode>SPADE</merchantCode>
   <currency>USD</currency>
</JackpotPoolRequest>
```

JSON JackpotPoolResponse：

```
{
   "list": [{
      "code": "Grand",
      "name": "Grand Jackpot",
      "amount": 3844.08,
      "currency": USD
   }],
   "merchantCode": "SPADE",
   "msg": "success",
   "code": 0,
   "serialNo": "20120802152140143938"
}
```

XML JackpotPoolResponse:

```
<JackpotPoolResponse>
   <serialNo>20120802152140143938</serialNo>
   <code>0</code>
   <msg>0</msg>
   <merchantCode>TEST</merchantCode>
   <resultCount>1</resultCount>
   <pageCount>1</pageCount>
   <list>
    <JackpotPoolInfo>
      <code>Grand</code>
      <name>Grand Jackpot</name>
      <amount>3844.08</amount>
      <currency>USD</currency>
    </JackpotPoolInfo>
  </list>
</JackpotPoolResponse>
```

4.9 查询商号的 Jackpot 彩池（以货币为准）

使用场景：查询用户信息（如余额）

接口名：jackpotPoolAll

接口提供者：Game Provider
接口使用者：Merchant

请求消息定义 `JackpotPoolRequest`：

名称	数据类型	必填	示例	说明
merchantCode	Varchar(10)	是	SPADE	标识商户 ID
serialNo	Varchar(50)	是	20120722224255982841	用于标识消息的序列，由调用者生成

响应消息定义 `JackpotPoolResponse`：

名称	数据类型	必填	示例	说明
list		是		JackpotPoolInfo 类

JackpotPoolInfo 定义 `JackpotPoolInfo`：

名称	数据类型	必填	示例	说明
code	Varchar(20)	是	Grand	Jackpot Code
amoutMap	Varchar(50)	是	"CNY":103418.588499	Jackpot 金额

JackpotPoolAll 请求和响应消息示例：

JSON JackpotPoolRequest：

```
{
   "merchantCode": "SPADE",
   "serialNo": "20120802152140143938"
}
```

XML JackpotPoolRequest:

```
<JackpotPoolAllRequest>
   <serialNo>20120802152140143938</serialNo>
   <merchantCode>SPADE</merchantCode>
</JackpotPoolAllRequest>
```

JSON JackpotPoolAllResponse：


```
{
   "list": [
      {
         "code": "Grand",
         "amountMap":
         {
            "CNY":103418.588499,
            "IDR":3991172.548579,
            "HKD":2175666.803645
         }
      },
      {
         "code":"GJ",
         "amountMap":
         {
            "CNY":16494.54591,
            "IDR":30333469.927946,
            "HKD":1944453.200509
         }
      }
   ]
   "status": 0,
   "merchantCode": "SPADE",
   "msg": "success",
   "code": 0,
   "serialNo": "20120726214956563472"
}
```

XML JackpotPoolAllResponse:

```
<JackpotPoolAllResponse>
   <serialNo>20120802152140143938</serialNo>
   <code>0</code>
   <msg>0</msg>
   <merchantCode>SPADE</merchantCode>
   <resultCount>1</resultCount>
   <pageCount>1</pageCount>
   <list>
       <code>Grand</code>
       <amountMap>
          <CNY>103418.588499</CNY>
          <IDR>3991172.548579</IDR>
          <HKD>2175666.803645</HKD>
       </amountMap>
       <code>GJ</code>
       <amountMap>
          <CNY>16494.54591</CNY>
          <IDR>30333469.927946</IDR>
          <HKD>1944453.200509</HKD>
       </amountMap>

   </list>
</JackpotPoolAllResponse>
```

4.10 查询游戏列表信息

使用场景：查询当前商家的用户游戏列表信息
接口名：getGames
接口提供者：Game Provider
接口使用者：Merchant

请求消息定义 `MerchantGamesRequest`：

名称	数据类型	必填	示例	说明
serialNo	Varchar(50)	是	20120722224255982841	用于标识消息的序列，由调用者生成
merchantCode	Varchar(10)	是	SPADE	标识商户 ID
currency	Char(3)	否	USD	货币
language	Varchar(10)	否	th_TH (Appendix 2)	Language code

响应消息定义 `MerchantGamesResponse`：

名称	数据类型	必填	示例	说明
games	GameInfo[]	是		GameInfo 类

GameInfo 定义 `GameInfo`：

名称	数据类型	必填	示例	说明
gameCode	Varchar(10)	是	S-GD02	游戏代码
gameName	Varchar(30)	是	DerbyNight	游戏名称
jackpot	boolean	是	True	是否有 jackpot
thumbnail	Varchar(100)	是	/thumbnail /aa.jpg	游戏图片
screenshoot	Varchar(100)	是	/screenshoot /bb.jpg	游戏快照
jackpotCode	Varchar(50)	否	Holy	只 jackpot 游戏
jackpotName	Varchar(50)	否	Holy Jackpot	只 jackpot 游戏

特别注意：

1. 图片的完整路径为 http://$host/images/aa.jpg, http://$host/images/bb.jpg. 其中$host 为 API URL 的 host 部分. Example: API 的 URL 为 http://demoapi.egame.spadegaming.com/api, 则游戏图片路径为 http://demoapi.egame.spadegaming.com/ thumbnail/Gamecode.jpg

2. 使用本接口时，建议使用定时任务，每天或是每周调用一次，然后下载图片。由于 api 的访问是 server-to-server, 并非 public IP 可以访问的，所以不能直接在网站上使用 API 返回的图片路径作为显示，必须下载图片到服务器上，然后显示在网站里面。

getGames 请求和响应消息示例：

JSON MerchantGamesRequest：

```
{
   "merchantCode": "SPADE",
   "serialNo": "20130502191551906534"
}
```

XML MerchantGamesRequest:

```
<MerchantGamesRequest>
  <serialNo>20130502191551906534</serialNo>
  <merchantCode>SPADE</merchantCode>
</MerchantGamesRequest>
```

JSON MerchantGamesResponse：

```
{
   "games": [{
      "gameCode": "SS-DG02",
      "gameName": "DerbyNight",
      "jackpot": true,
      "thumbnail": "http://xxxx.com/aa.jpg",
      "screenshot": "http://xxxx.com/aa.jpg",
      "jackpotCode ": "Holy",
       "jackpotName ": "Holy Jackpot"

   }],
   "merchantCode": "SPADE",
   "code": 0,
   "msg": "success",
   "serialNo": "20130502191551906534"
}
```

XML MerchantGamesResponse:

```
<MerchantGamesResponse>
  <serialNo>20130502191551906534</serialNo>
  <code>0</code>
  <msg>success</msg>
  <merchantCode>SPADE</merchantCode>
  <games>
    <GameInfo>
      <gameCode>SS-DG02</gameCode>
      <gameName>DerbyNight</gameName>
      <jackpot>true</jackpot>
      <thumbnail>http://xxxx.com/aa.jpg</thumbnail>
      <screenshot>http://xxxx.com/aa.jpg </screenshot>
      <jackpotCode>Holy</jackpotCode >
      <jackpotName>Holy Jackpot</jackpotName>
    </GameInfo>
  </games>
</MerchantGamesResponse>
```

4.11 强制退出当前在线用户

使用场景：当 merchant 网站进行维护时,如果不希望当前在游戏中的玩家继续游戏,则调用本接口,将所属的在线用户强制退出
接口名：kickAcct
接口提供者：Game Provider
接口使用者：Merchant

请求消息定义 KickAcctRequest：

名称	数据类型	必填	示例	说明
acctId	Varchar(50)	否	TESTPLAYER1	玩家 ID，如果为空则退出全部

响应消息定义 KickAcctResponse：

名称	数据类型	必填	示例	说明
List	String[]	是	[TESTPLAYER1]	退出的用户列表

KickAcct 请求和响应消息示例:

JSON KickAcctRequest：

```
{
   "acctId": "TESTPLAYER1",
   "merchantCode": "SPADE",
   "serialNo": "20130430164644935780"
}
```

XML KickAcctRequest：

```
<KickAcctRequest>
  <serialNo>20130430164644935780</serialNo>
  <merchantCode>SPADE</merchantCode>
  <acctId>TESTPLAYER1</acctId>
</KickAcctRequest>
```

JSON KickAcctResponse：

```
{
   "list": ["Test10001"],
   "merchantCode": "SPADE",
   "code": 0,
   "msg": "success",
   "serialNo": "20130430164644935780"
}
```

XML KickAcctResponse：

```
<KickAcctResponse>
  <serialNo>20130430164644935780</serialNo>
  <code>0</code>
  <msg>success</msg>
  <merchantCode>SPADE</merchantCode>
  <list>
    <string>Test10001</string>
  </list>
</KickAcctResponse>
```

4.12 创建 Token

使用场景：Merchant 通过此 API 获取 token 来请求单号详细连结
接口名： createToken
接口提供者：Game Provider
接口使用者：Merchant

请求消息定义 `createTokenRequest:`

名称	数据类型	必填	示例	说明
acctId	Varchar(50)	是	TESTPLAYER1	用户标识 ID
merchantCode	Varchar(50)	是	SPADE	商号
action	Varchar(50)	是	ticketLog	客户行径
serialNo	Varchar(50)	是	20120723175139440637	用于标识消息的序列，由调用者生成

响应消息定义 `createTokenResponse`：

名称	数据类型	必填	示例	说明
token	Varchar(128)	是	fe1a85adc54545d29	
merchantCode	Varchar(50)	是	SPADE	商号
serialNo	Varchar(50)	是	20120723175139440637	用于标识消息的序列，由调用者生成

特别注意：

此功能并非用来產生 authorize 所需的 token 使用

createToken 请求和响应消息示例

JSON createTokenRequest：

```
{
   "acctId": "TESTPLAYER1",
    "merchantCode": "SPADE",
    "action": "ticketLog",
    "serialNo": "20120722224255982841"
}
```

XML createTokenRequest：

```
<createTokeneRequest>
    <acctId>TESTPLAYER1</acctId>
    <merchantCode>SPADE</merchantCode>
   <action>ticketLog</action>
   <serialNo>20120723161423158480</serialNo>
</createTokenRequest>
```

JSON createTokenResponse：

```
{
    "token": "fela3ret56dhhT999",
    "merchantCode": "SPADE",
    "serialNo": "20120722224255982841"
}
```

XML createTokenResponse:

```
<createTokenResponse>
   <token>fela3ret56dhhT999</token>
    <merchantCode>SPADE</merchantCode>
    <serialNo>20120722224255982841</serialNo>
</createTokenResponse>
```

4.13 单号详细页面跳转

使用场景：当游戏玩家在进入 merchant 网站后，通过网页上的 View ticket details 按钮，将玩家页面转到单号详细的页面。

接口名：Redirect To Ticket Details Page
接口提供者：Game Provider
接口使用者：Merchant

名称	数据类型	必填	示例	说明
ticketId	Varchar(20)	是	641482277	单号
acctId	Varchar(50)	是	TESTPLAYER1	玩家 ID
action	Varchar(50)	是	ticketLog	商家行径
token	Varchar(80)	是	fe1a85adc54545d29	
language	Varchar(10)	否	en_US	显示单号详细页面的语言.

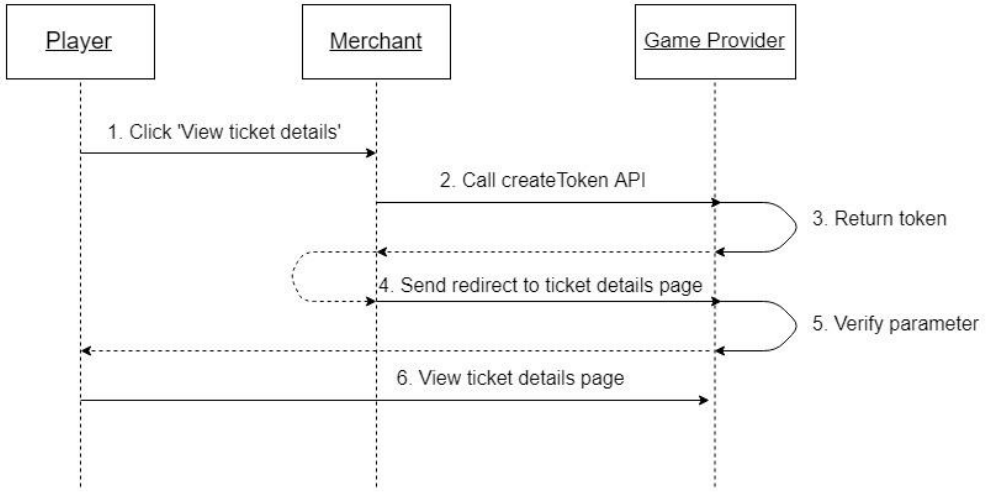
特别注意：

1. 详细的 language 的定义，请参照附录 2“语言代码定义”。
2. Action 使用‘ticketLog’

Ticket Details Redirect URL and Parameters:

```
http://portal.e-games.com/merchantCode/createToken/?ticketId=248114533&acctId=TESTPLAYER1&action=ticketLog&token=fe1a85adc54545d2963b661a22d09c9e&language=en_US
```

Get Ticket Details Flow



4.14 查询玩家账号

使用场景：查询玩家账号是否存在

接口名：checkAcct

接口提供者：Game Provider

接口使用者：Merchant

请求消息定义 checkAcct Request

名称	数据类型	必填	示例	说明
acctId	Varchar(50)	是	TESTPLAYER1	玩家账号
includeType	Boolean	否	True	确认是否为 youtuber 玩家
merchantCode	Varchar(50)	是	SPADE	标识商户 ID
serialNo	Varchar(50)	是	2012072223141359560 2	用于标识消息的序列，由调用者生成

响应消息定义 checkAcct Request

名称	数据类型	必填	示例	说明
serialNo	Varchar(50)	是	20120722224255982841	用于标识消息的序列，由调用者生成
code	Integer	是	0	处理结果代码定义 code
acctType	Integer	否	0	处理结果代码定义 code 0:normal 1:youtuber
msg	Varchar(128)	是	success	处理结果描述
merchantCode	Varchar(10)	是	SPADE	标识商户 ID

checkAcct 请求和响应消息示列:

JSON checkAcct Request：

```
{
       "acctId": " TESTPLAYER1",
       "includeType": true,
       "merchantCode": "SPADE",
       "serialNo": "20180722231413699735"
}
```

XML checkAcct Request：

```
<CheckAcctRequest>
      <acctId>TESTPLAYER1</acctId>
      <includeType>true</includeType>
      <merchantCode>SPADE</merchantCode>
      <serialNo>20180722231413699735</serialNo>
</CheckAcctRequest>
```

JSON checkAcct Response：

```
{
      "serialNo":"20180722231413699735",
      "code":0,
      "acctType":0,
      "msg":"Success",
      "merchantCode":"SPADE"
}
```

XML checkAcct Response：

```
<CheckAcctResponse>
  <serialNo>20180722231413699735</serialNo>
  <code>0</code>
  <acctType>0</acctType>
  <msg>Success</msg>
  <merchantCode>SPADE</merchantCode>
</CheckAcctResponse>
```

4.15 查询玩家下注总结

使用场景：查询玩家下注总结

接口名：playerDailySumByGame

接口提供者：Game Provider
接口使用者：Merchant

请求消息定义 playerDailySumByGameRequest：

名称	数据类型	必填	示例	说明
beginDate	Char(15)	是	20120721T000000	开始时间，格式为 yyyymmdd'T'24hhmmss
endDate	Char(15)	是	20120721T010000	截止时间，格式为 yyyymmdd'T'24hhmmss
pageIndex	Integer	是	1	页码
merchantCode	Varchar(10)	是	SPADE	标识商户 ID
acctId	Varchar(50)	否	TESTPLAYER1	用户标识 ID

响应消息定义 playerDailySumByGameResponse：

名称	数据类型	必填	示例	说明
resultCount	Integer	是	100	总共的记录数
pageCount	Integer	是	1	总共的页数
list	PlayerBetInfo[]	是		PlayerBetInfo 的数组

PlayerBetInfo 定义 `PlayerBetInfo`：

名称	数据类型	必填	示例	说明
acctId	Varchar(50)	是	TESTPLAYER1	用户标识 ID
list	BetInfo[]	是		BetInfo 的数组

BetInfo 定义 `BetInfo`：

名称	数据类型	必填	示例	说明
categoryId	Varchar(10)	是	SM or TB or AD or BN	游戏种类
gameCode	Varchar(10)	是	S-GD02	游戏代码
bets	Integer	是	1	下注次数
betAmount	Decimal(18,6)	是	20	下注金额
jackpotAmount	Decimal(18,6)	是	0	
jpWin	Decimal(18,6)	是	-100.000000	积宝赢额
winLoss	Decimal(18,6)	是	-20	用户输赢

特别注意：

1. 拉取区间最大不能超过 1 天
2. 拉取数据时间必须是当前时间 12 小时前
3. 数据是以小时查询

playerDailySumByGame 请求和响应消息示例:

JSON playerDailySumByGameRequest：

```
{
   "beginDate": "20120721T000000",
   "endDate": "20120721T010000",
   "pageIndex": 1,
   "merchantCode": "SPADE",
   "serialNo": "20120723175139440637"
}
```

XML playerDailySumByGameRequest：

```
<GetBetHistoryRequest>
   <serialNo>20120723175139440637</serialNo>
   <merchantCode>SPADE</merchantCode>
   <beginDate>20120721T000000</beginDate>
   <endDate>20120721T010000</endDate>
   <pageIndex>1</pageIndex>
</GetBetHistoryRequest>
```

JSON playerDailySumByGameResponse：


```
{
   "resultCount": 2,
   "pageCount": 1,
   "list": [
    {
       "acctId": "TESTPLAYER1",
       "list": [{
           "bets": 34,
           "betAmount": 20,
           "winLoss": 30,
           "jackpotAmount": 1.5,
           "jpWin": -100.000000,
           "categoryId": "SM",
           "gameCode": "S-GD02"
         }]
    },
    {
       "acctId": "TESTPLAYER2",
       "list": [{
           "bets": 34,
           "betAmount": 20,
           "winLoss": 30,
           "jackpotAmount": 1.5,
           "jpWin": -100.000000,
           "categoryId": "SM",
           "gameCode": "S-GD02"
         }]
    }

   }],
   "merchantCode": "SPADE",
   "msg": "success",
   "code": 0,
   "serialNo": "20120723175139440637"
}
```

XML playerDailySumByGameResponse：

```
<PlayerDailySumByGameResponse>
   <serialNo>20120723175139440637</serialNo>
   <code>0</code>
   <msg>success</msg>
   <merchantCode>SPADE</merchantCode>
   <resultCount>1</resultCount>
   <list>
    <PlayerBetInfo>
      <acctId>TESTPLAYER1</acctId>
      <list>
      <BetInfo>
         <bets>20</bets>
         <betAmount>20</betAmount>
         <winLoss>30</winLoss>
         <jackpotAmount>0</jackpotAmount>
         <jpWin>-100.000000</jpWin>
         <categoryId>SM</categoryId>
         <gameCode>S-GD02</gameCode>
      </BetInfo>
      </list>
    </PlayerBetInfo>
    </list>
</PlayerDailySumByGameResponse>
```

4.16 设置 Youtuber

使用场景：用于商户设置某特定玩家为 youtuber 形式

接口名：setYoutuber

接口提供者：Game Provider
接口使用者：Merchant

请求消息定义 setYoutuberRequest：

名称	数据类型	必填	示例	说明
acctId	Varchar(50)	是	TESTPLAYER1	游戏玩家 ID
currency	Char(3)	是	USD	Currency Code
youtuber	Boolean	是	True	True: Youtuber 模式 False: 一般模式
merchantCode	Varchar(50)	是	SGDEMO	由 Merchant 端生成
serialNo	Varchar(50)	是	20120722231413595602	用于标识消息的序列，由调用者生成

响应消息定义 setYoutuberResponse：

名称	数据类型	必填	示例
serialNo	Varchar(50)	20120722224255982841	用于标识消息的序列，由调用者生成.
code	Integer	0	处理结果代码定义 code
msg	Varchar(128)	success	处理结果描述
merchantCode	Varchar(10)	SGDEMO	标识商户 ID

setYoutuber 请求和响应消息示例:

JSON setYoutuberRequest：

```
{
    "acctId": "TESTPLAYER1",
    "currency": "USD",
    "youtuber": true,
    "merchantCode": "SPADE",
    "serialNo": "20120722231413699735"
}
```

XML setYoutuberRequest：

```
<setYoutuberRequest>
    <acctId>TESTPLAYER1</acctId>
    <currency>USD</currency>
    <merchantCode>SPADE</merchantCode>
    <serialNo>20120722231413699735</serialNo>
    <youtuber>true</youtuber>
</setYoutuberRequest>
```

JSON setYoutuberResponse：

```
{
    "merchantCode": "SPADE",
    "msg": "success",
    "code": 0,
    "serialNo": "20120722231413699735"
}
```

XML setYoutuberResponse：

```
<setYoutuber>
   <code>0</code>
   <merchantCode>SPADE</merchantCode>
   <msg>success</msg>
   <serialNo>20120722231413699735</serialNo>
</setYoutuber>
```

附录 1 响应代码定义

使用场景：Merchant 会收到的响应代码定义通过 API。

code	中文描述	Message in English	说明
0	成功	Success	接口调用成功
1	系统错误	System Error	系统内部出错，程序 bug、数据库连接不上等
2	非法请求	Invalid Request	不是合法的请求
3	服务暂时不可用	Service Inaccessible	服务不可用
100	请求超时	Request Timeout	
101	用户调用次数超限	Call Limited	
104	请求被禁止	Request Forbidden	
105	缺少必要的参数	Missing Parameters	
106	非法的参数	Invalid Parameters	
107	批次号重复	Duplicated Serial NO.	
109	关联 id 找不到	Related id not found	这个通常用于单一钱包的 Transfer API
110	批次号不存在	Record ID Not Found	
111	重复请求	Duplicated request	
112	API 调用次数超限	API Call Limited	调用 API 接口的次数已超限
113	Acct ID 不正确	Invalid Acct ID	Acct ID 格式不正确
118	格式不正确	Invalid Format	Parse Json Data Failed
120	ip 不在白名单内	IP no whitelisted	
5003	游戏正在维护中	System Maintenance	
10113	Merchant 不存在	Merchant Not Found	
10116	Merchan 被暂停	merchant suspend	
50099	账号已存在	Acct Exist	
50100	账号不存在	Acct Not Found	
50101	帐号未激活	Acct Inactive	
50102	帐号已锁	Acct Locked	
50103	帐号 suspend	Acct Suspend	
50104	Token 验证失败	Token Validation Failed	
50110	帐号余额不足	Insufficient Balance	
50111	超过帐号交易限制	Exceed Max Amount	
50112	币种不支持	Currency Invalid	Deposit/withdraw 的币种与用户的币种不一致，或者 merchant 没

			有相应的币种
50113	金额不合法	Amount Invalid	Deposit/withdraw 金额必须>0
50115	日期格式错误	Date Format Invalid	时间格式不正确

附录 2 语言代码定义

ISO Language Code	Description
en_US	英文
zh_CN	简体中文
th_TH	泰文
id_ID	印尼文
vi_VN	越南文
ko_KR	韩文
ja_JP	日文
ru_RU	俄文
tr_TR	土耳其文
es_ES	西班牙文
fr_FR	法文

附录 3 货币代码定义

ISO 货币代码	Description	叙述	ISO 货币代码	Description	叙述
BDT	Bangladeshi Taka	孟加拉塔卡	MXN	Mexican Peso	墨西哥比索
CNY	Chinese Yuan (Renminbi)	人民币	NZD	New Zealand Dollar	纽西兰元
USD	US Dollar	美元	PEN	Peruvian New Sol	秘鲁新索尔
EUR	Euro	欧元	PLN	Polish Zloty	波兰兹罗提
EPV	Euro	欧元	PYG	Paraguayan Guarani	巴拉圭瓜拉尼
GBP	British Pound	英镑	RSD	Serbian Dinar	塞尔维亚第纳尔
HKD	Hong Kong Dollar	港币	UYU	Uruguayan Peso	乌拉圭比索
SGD	Singapore Dollar	新加坡元	ZAR	South African Rand	南非兰特
KRW	South Korean Won	韩元	UAH	Ukrainian hryvnia	乌克兰格里夫纳
JPY	Japanese Yen	日元	TND	Tunisian Dinar	突尼西亚第纳尔
THB	Thai Baht	泰铢	RUB	Russian ruble	俄罗斯卢布
IDR	Indonesia Rupiah	印尼盾	TRY	Turkish Lira	土耳其里拉
ID2	Indonesia Rupiah	印尼盾	SEK	Swedish Krona	瑞典克朗
INR	Indian Rupee	印度卢比	NOK	Norwegian Krone	挪威克朗
VND	Viet Nam Dong	越南盾	AED	Dirham	迪拉姆
VN2	Viet Nam Dong	越南盾	AMD	Armenian Dram	亚美尼亚德拉姆
MYR	Malaysia Ringgit	马来西亚零吉	CAD	Canadian Dollar	加拿大元
MMK	Myanmar Kyat	缅元	CHF	Swiss franc	瑞士法郎
MMV	Myanmar Kyat	缅元	CLP	Chilean Peso	智利披索
AUD	Australian Dollar	澳大利亚元	CZK	Czech Koruna	捷克克朗
ALL	Albania	阿尔巴尼亚	DKK	Danish Krone	丹麦克朗
BRL	Brazilian Real	巴西雷亚尔	ARS	Argentine Peso	阿根廷比索
BGN	Bulgarian Lev	保加利亞列弗	COP	Colombian Peso	哥伦比亚的比索
IRR	IranianRial	伊朗里亚尔	CO2	Colombian Peso	哥伦比亚的比索
IR2	IranianRial	伊朗里亚尔	PKR	Pakistan Rupee	巴基斯坦卢比
KZT	Kazakhstani Tenge	哈萨克斯坦坚戈	HUF	Hungarian Forint	匈牙利福林
XAF	Central African CFA franc	中非法郎	XOF	West African CFA franc	西非法郎
AZN	Azerbaijan Manat	新马纳特	NGN	Nigerian Nairas	尼日利亚奈拉
ILS	Israeli Shekel	以色列新谢克尔	LBP	Lebanese pound	黎巴嫩镑
MDL	Moldovan leu	摩尔多瓦列伊	ETB	Ethiopian birr	埃塞俄比亚比尔
TJS	Tajikistani somoni	塔吉克斯坦索莫尼	BAM	Bosnia-Herzegovina Convertible Marka	马克
KGS	Kyrgyzstani som	吉尔吉斯斯坦索姆	TMT	Turkmenistani manat	土库曼斯坦马纳特
BND	Bruneian Dollars	汶莱元	DEM	Deutsche Mark	德国马克
KSH	Kenyan shilling	肯雅先令	RON	Romanian leu	罗马尼亚列伊

BYN	Belarusian ruble	白俄罗斯卢布	NPR	Nepalese rupee	尼泊尔卢比
GEL	Georgian lari	格鲁吉亚拉里	KHR	Cambodian Riels	柬埔寨瑞尔
LAK	Lao Kip	老挝基普	BIF	Burundian Francs	布隆迪法郎

备注：

1. IRR , COP ,VND, IDR, MMV, LAK, KHR 比率 1:1000
2. IR2, CO2,VN2 和 ID2 和 MMK 用比率 1:1

附录 4 API 调用 Java 示例(HTTP)

Constants.java

```
public interface Constants {
    public static final Charset Charset = Charsets.UTF_8;
    public static final String  DateFormat = "yyyyMMdd'T'HHmmss";
    public static final String GZIP = "gzip";
    public static final int PageSize = 100;

    public static interface HttpHeaders {
        public static final String ACCEPT_ENCODING =
        "Accept-Encoding";
        public static final String ACCEPT_LANGUAGE =
        "Accept-Language";
        public static final String CONTENT_LENGTH =
        "Content-Length";
        public static final String CONTENT_TYPE = "Content-Type";
        public static final String CONTENT_ENCODING =
        "Content-Encoding";
        public static final String API = "API";
        public static final String DATA_TYPE = "DataType";
        public static final String DIGEST = "Digest";
    }

    public static interface DataType {
        public static final String Xml = "XML";
        public static final String Json = "JSON";
    }
}
```

Api Post.java

```
    public static byte[] post(String target, String api, String
    dataType, byte[] data) throws Exception {
    String requestHash = DigestUtils.digest( data );
    HttpURLConnection conn = null;
    OutputStream out = null;
    InputStream in = null;
    try {
        String dataLength = Integer.toString( data.length );
```

```
        //Create connection
        URL url = new URL( target );
        conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod( "POST" );
        conn.setRequestProperty( API, api );
        conn.setRequestProperty( DATA_TYPE, dataType );
        conn.setRequestProperty( DIGEST, requestHash );
        conn.setRequestProperty( CONTENT_LENGTH, dataLength );
        conn.setRequestProperty( ACCEPT_ENCODING, GZIP );
        conn.setUseCaches( false );
        conn.setDoInput( true );
        conn.setDoOutput( true );
        conn.setReadTimeout( timeout );

        //Send request & handle Response
        out = conn.getOutputStream();
        out.write( data );
        out.flush();
        in = conn.getInputStream();
        return handleResponse( in, conn );
    } finally {
        IOUtils.closeQuietly( out );
        IOUtils.closeQuietly( in );
        if ( conn != null ) {
            conn.disconnect();
        }
    }
}

    private static byte[] handleResponse(final InputStream in,
    final HttpURLConnection conn) throws IOException {
    final int code = conn.getResponseCode();
    if ( code != 200 ) {
        throw new RuntimeException( "server return error! status "
        + code );
    }
    InputStream is = in;
    String encoding = conn.getHeaderField( CONTENT_ENCODING );
    if ( StringUtils.notEmpty( encoding ) &&
    encoding.indexOf( GZIP ) >= 0 ) {
        int contentLength = conn.getContentLength();
        is = contentLength > 0 ? new GZIPInputStream( in,
        contentLength ) : new GZIPInputStream( in );
```

```
    }
    byte[] data = IOUtils.toByteArray( is );
    String responseHash = conn.getHeaderField( DIGEST );
    if ( StringUtils.notEmpty( responseHash ) ) {
        String computedHash = DigestUtils.digest( data );
        if ( !responseHash.equals( computedHash ) ) {
            String msg = String.format( "received digest data
            invalid! response: %s", new String( data, Charset ) );
            throw new RuntimeException( msg );
        }
    }
    return data;
}
```

附录 5 API 调用 c#示例(HTTP)

HttpHeader.cs

```
public class HttpHeaders {
    public const String AcceptEncoding = "Accept-Encoding";
    public const String AcceptLanguage = "Accept-Language";
    public const String ContentEncoding =
    "Content-Encoding";
    public const String Api = "API";
    public const String DataType = "DataType";
    public const String Disgest = "Digest";
}
```

HttpPost.cs

```
public class HttpPost
{
    public event MessageEventHandler OnMessageReceived;

    private static readonly HashAlgorithm DigestProvider =
    new MD5CryptoServiceProvider();
    private static readonly Encoding Charset = Encoding.UTF8;

    public void Send(String target, String api, String
    dataType, String data)
    {
        Byte[] body = Charset.GetBytes(data);
        String hash = ComputeHash(body);

        HttpWebRequest request =
        (HttpWebRequest)WebRequest.Create(target);
        request.Method = "POST";
        request.ContentLength = body.Length;
        request.Headers[HttpHeaders.Api] = api;
        request.Headers[HttpHeaders.DataType] = dataType;
        request.Headers[HttpHeaders.AcceptEncoding] =
        "gzip";   //support gzip
        request.AutomaticDecompression =
        DecompressionMethods.GZip;
        request.Headers[HttpHeaders.Disgest] = hash;
```

```
        using (Stream os = request.GetRequestStream())
        {
            os.Write(body, 0, body.Length);
        }

        IAsyncResult result = request.BeginGetResponse(new
        AsyncCallback(AsyncReceive), request);
    }


    private void AsyncReceive(IAsyncResult result)
    {
        HttpWebRequest request =
        (HttpWebRequest)result.AsyncState;
        HttpWebResponse response =
        (HttpWebResponse)request.EndGetResponse(result);
        String message = null;
        using (Stream stream = response.GetResponseStream())
        {
            StreamReader sr = new StreamReader(stream,
            Charset);
            message = sr.ReadToEnd().Trim();

            if (response.StatusCode != HttpStatusCode.OK)
            {
                message += "err: " + response.StatusCode +
                Environment.NewLine;
            }
        }

        string hash = response.GetResponseHeader("hash");
        if (!String.IsNullOrEmpty(hash)
        && !"err".Equals(message))
        {
            Byte[] body = Charset.GetBytes(message);
            String computed = ComputeHash(body);
            if (!computed.Equals(hash))
            {
                message += String.Format("hash not equal:
                \nreceived: {0}, computed: {1}", hash,
                computed);
            }
            else
            {
```

```
                hash += Environment.NewLine + "Hash check
                passed";
            }
        }

        if (OnMessageReceived != null)
        {
            MessageEventArgs e = new MessageEventArgs();
            e.Data = message;
            OnMessageReceived(this, e);
        }
    }

    private String ComputeHash(Byte[] body)
    {
        Byte[] hashBytes = DigestProvider.ComputeHash(body);
            String hash =
            BitConverter.ToString(hashBytes).Replace("-",
            String.Empty);
        return hash;
    }

}
```

附录 6 Authorize 示例(merchant)

```
public class AuthorizeAction implements ResponseCodes {
    //dependent inject from config
    private String merchant;

    //dependent inject
    private TokenManager tokenManager;
    private AcctDao acctDao;

    public AuthorizeResponse execute(AuthorizeRequest request) {
        AuthorizeResponse response = new AuthorizeResponse();
        if ( !merchant.equals( request.getMerchantCode() ) ) {
            response.setCode( MERCHANT_NOT_FOUND );
            return response;
        }
        final String acctId = request.getAcctId();
        final String reqToken = request.getToken();
        String token = tokenManager.findTokenById( acctId );
        if ( isEmpty( token ) || !token.equals( reqToken ) ) {
            response.setCode( TOKEN_INVALID );
            return response;
        }
        //expire token and enhance security
        tokenManager.expireToken( acctId );

        Acct acct = acctDao.findById( acctId );
        if(acct == null){
            response.setCode( ACCT_NOT_FOUND );
            return response;
        }

        AcctInfo info = new AcctInfo();
        info.setAcctId( acct.getId() );
        info.setUserName( acct.getName() );
        info.setCurrency( acct.getCurrency() );
        response.setAcctInfo( info );

        response.setCode( SUCCESS );
        return response;
    }
}
```

附录 7 RSA 解密示例

```
import java.security.Key;
import java.security.KeyFactory;
import java.security.Provider;
import java.security.interfaces.RSAPrivateKey;
import java.security.spec.PKCS8EncodedKeySpec;
import javax.crypto.Cipher;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

public static void decrypt(String privateKey,String
encryptedData ) throws Exception {
	Provider provider = new BouncyCastleProvider( );
	byte[] input =
Hex.decodeHex( encryptedData.toCharArray( ) );
	Cipher cipher = Cipher.getInstance( "RSA", provider );
	byte[] hex = Hex.decodeHex( privateKey.toCharArray( ) );
	PKCS8EncodedKeySpec pkcs8KeySpec = new
PKCS8EncodedKeySpec( hex );
	KeyFactory keyFactory = KeyFactory.getInstance( "RSA",
provider );
	Key privateK = keyFactory.generatePrivate( pkcs8KeySpec );

	cipher.init( 2, ( RSAPrivateKey ) privateK );

	byte[] raw = cipher.doFinal( input );

	String s = new String( raw, "UTF-8" );
	String result = new
StringBuilder( s ).reverse( ).toString( );
	System.out.println( result );
   }
```