UC优视

UC 优视游戏 SDK

开发参考说明书

— 服务器接口

2012-07-17

手机上网用UC，就是比你快

修订记录

归档日期	版本	说明	作者	审批人
2012-07-17	1.0.7	增加支付通道代码（1.3.3）		
2012-07-20	1.0.8	补充加密方式和支付通道的说明（1.3.3）		
2012-08-09	1.0.9	增加新支持的支付通道说明（1.3.3）		
2012-08-24	1.0.10	增加游戏老账号登录与 UC 账号进行绑定的说明（2.游戏服务器开发要点）		
2012-08-27	1.0.11	1.新增角色查询接口说明（1.3.4）； 2.充值结果回调接口增加角色编号字段（1.3.3）。		
2012-11-09	1.0.12	1.充值结果回调接口增加充值场景可选字段，及 PC 反充说明（1.3.3）； 2.增加新支持的支付通道说明（1.3.3）。		
2013-07-02	1.0.13	更新支付通道代码（1.3.3）		
2014-07-14	1.0.14	更新充值结果回调接口的说明（1.3.3）		
2014-07-23	1.0.15	1.更新用户会话验证的说明（1.3.1） 2.更新充值结果回调接口的说明（1.3.3）		
2014-08-4	1.0.16	1.去除过期接口“角色查询接口” 2.新增“扩展数据提交接口”接口说明		
2014-08-5	1.0.17	修改各接口中 channelId 的说明		
2014-09-02	1.1.0	1.去掉请求参数 game 中 cpId,serverId,channelId 2.用户会话验证接口修改为 account.verifySession		
2014-10-14	1.2.0	签名规则去掉 cpId 值		
2014-11-21	1.2.1	更新充值结果回调接口的说明（1.3.3）		
2014-11-21	1.2.2	更新充值结果回调接口的说明（1.3.3）		
2014-11-24	1.2.3	更新充值结果回调接口的说明（1.3.3）		

2015-03-18	1.2.4	1.更新协议说明（1.2.2） 2.更新登录会话验证接口说明（1.3.1）		
2015-06-10	1.2.5	1.更新协议说明（1.22) 2.更新充值结果回调接口（1.3.3） 3.删除原“登录状态同步接口”（1.3.2）		

目录

1. 服务器端接口说明

1.1. 概要

本部分主要提供 UC 游戏中心“SDK 服务器”和 UC 游戏合作商“游戏服务器”的交互的接口规范。

1.2. 协议说明

1.2.1 通信协议

“SDK 服务器”采用 HTTP 协议作为通信协议，“游戏服务器”通过构造 HTTP 请求（POST 方式）向“SDK 服务器”发起接口请求。

“游戏服务器” HTTP 请求数据示例：

```
POST http://sdk.g.uc.cn/cp/account.verifySession HTTP1.1
Content-Type: application/json
Content-Length: 136
{
    "id":1332406591685,
    "game":{"gameId":5},
    "data":{
        "sid":"110adf4c-f2d3-4be5-8a9c-3741a83e5853"
    },
    "sign":"bb926c2a9944e9b4f2f6639d928dc95c"
}
```

“SDK 服务器” 响应数据示例：

```
200 OK
Content-Type:  application/json:charset=utf-8
Content-Length:  155

{
    "id":1332406591685,
    "state":{"code":1, "msg":"操作已完成"},
    "data":{
       "creator":"JY",
        "accountId":"U11626774a4e39c16cf7mmsnz5002une",
        "nickName":"昵称"
    }
}
```

1.2.2 数据协议

1) 数据格式

请求消息和响应消息的内容都使用 json 格式表示数据，具体请参考下文的例子。

2) 字符编码

请求与响应内容（json 格式数据）须采用 utf-8 字符编码。

3) 签名规则

使用 md5 哈希对请求内容进行签名，算法如下：

```
md5(签名内容+apiKey)
```

说明：

MD5 使用 RFC 1321 标准，编码后需要转换回全小写(本文提到的所有 md5 结果，都需要转换成全小写)。

表达式中的“+”号表示字符串接，并不存在。

签名内容是指请求数据（data 字段）中，各字段名及其字段值的拼接，字段名与字段值之间使用等号（=）连接。拼接时需对字段名排序，排序方式是按字段名进行升序

排列。

注意：签名内容不应包含“&”符号，拼接签名内容时需把“&”符号剔除。计算MD5 签名时，取签名内容的字节时，应以 UTF-8 编码取字符串的字节值。

apiKey 由 UC 分配，游戏在审核通过后可通过开放平台查看。

4) 签名示例

假设请求内容请求数据（data 字段）为 ：

```
"data":{
    "personid":value1,
    "code":"value2",
    "name":"value3"
    }
}
```

排序拼接后得出要签名的内容串为：

```
code=value2name=value3personid=value1
```

假定 apiKey=202cb962234w4ers2aa，

要进行 MD5 哈希的字符串为：

```
echo -n code=value2name=value3personid=value1202cb962234w4ers2aa|md5sum
```

执行 MD5 哈希，下面为在 linux shell 中执行命令取得 MD5 值：

```
8f468d03c2c42fbe294bdd49f038c031
```

得出签名结果是：

```
8f468d03c2c42fbe294bdd49f038c031
```

最后发送的请求内容为：

```
{
    "id":1330395827,
    "service":"aa.aa.aa",
    "data":{
        "personid":value1,
        "code":"value2",
        "name":"value3"
    },
    "sign":"8f468d03c2c42fbe294bdd49f038c031"
}
```

1.3. 接口说明

1.3.1 用户会话验证

1) 请求地址

测试环境： http://sdk.test4.g.uc.cn/cp/account.verifySession

正式环境：http://sdk.g.uc.cn/cp/account.verifySession

2) 调用方式：HTTP POST

3) 接口描述：

验证 sid 是否为有效的登录用户会话，若有效则返回其账号标识、账号创建者和昵称。

“游戏客户端”通过“SDK 客户端”获取到 sid（详细参考对应 API 分册），传到“游戏服务器”，“游戏服务器”到“SDK 服务器”验证用户会话 sid 的有效性，获取用户的账号标识、账号创建者和昵称，供游戏使用。

注意：进行接口调用前请确认 sid 是否具备值，如 sid 值为空时请勿调用此接口。

4) **请求方：**游戏服务器

5) **响应方：**SDK 服务器

6) **请求内容（json 格式）：**

字段名称	字段说明	类型	必填	备注
id	请求的唯一标识	long	Y	Unix 时间戳，例：1330395827
data	请求数据	json	Y	json 格式

game	game 参数	json	Y	json 格式 ,字段值均为整数。格式如下：
				{
				“gameId” ：游戏编号
				}

sign	签名参数	string	Y	MD5(签名内容+apiKey) 签名内容： sid=……
请求数据(对应 data,采用 json 格式)				
sid	当前用户会话标识	string	Y	游戏客户端登录后从 SDK 取得

例子：

HTTP 请求的 body 内容：

```
{
    "id":1330395827,
    "data":{"sid":"abcdefg123456"},
    "game":{"gameId":12345},
    "sign":"6e9c3c1e7d99293dfc0c81442f9a9984"
}
```

假定 apiKey=202cb962234w4ers2aaa

sign 的签名规则：MD5(sid=...+apiKey)（去掉+；替换...为实际值）

签名原文：

```
sid=abcdefg123456202cb962234w4ers2aaa
```

MD5 加密：

```
echo -n sid=abcdefg123456202cb962234w4ers2aaa|md5sum
```

加密结果：091391c3613711383d4d631318674ac8

7） 返回内容（json 格式）：

字段名称	字段说明	类型	必填	备注
id	请求的唯一标识	long	Y	Unix 时间戳，与请求内容中 id 值相同。例：1330395827
state	响应状态	json	Y	
data	响应数据	json	Y	
响应状态（对应 state,采用 json 格式）				
code	响应码	int	Y	
msg	结果描述	string	Y	
响应数据(对应 data,采用 json 格式)				
accountId	账号标识	string	Y	最长为 32 个字符
creator	账号创建者	string	Y	JY：九游

				PP：PP 助手
nickName	用户昵称	string	Y	

例子：

```
{
    "id":1330395827,
    "state":{"code":1, "msg":"操作成功"},
   "data":{
          "accountId":"U11626774a4e39c16cf7mmsnz5002une",
          "creator":"JY",
          "nickName":"九游玩家"
    }
}
```

响应码说明（state.code）：

响应码	说明	错误原因
1	成功	--
10	请求参数错误	请求内容格式有误、gameID 有误、签名校验失败等
11	用户未登录	sid 不存在，请求地址有误等
99	服务器内部错误	接口名有误、请求地址有误等

1.3.2 充值结果回调接口

1) **请求地址：**即充值结果通知地址，由游戏合作商提供。游戏接入时，由游戏合作商可在 SDK 客户端设置或开放平台录入，优先取客户端设置的地址。

2) **调用方式：**HTTP POST

3) **接口描述：**

用户在游戏中提交充值请求后，UC 游戏平台会异步执行充值，在充值操作完成后，UC 游戏平台通过该接口将充值结果给“游戏服务器”。

此处定义本接口的规范，游戏合作商需根据此规范在“游戏服务器”实现本接口。

此接口用于接收通过游戏 SDK 充值的结果通知， 和通过 PC 页面进行直接充值的结果通知。

注： 支付回调异步信息是通过把回调信息 json 串放在 http 的 body 里，post 给游戏服务器的回调地址上，游戏接收回调时请注意需获取 body 内容。

4) 请求方：SDK 服务器

5) 响应方：游戏服务器

6) 请求内容（json 格式）：

字段名称	字段说明	类型	必填	备注
ver	版本号	string	Y	接口版本号，2.0
data	支付结果数据	json	Y	
sign	签名参数	string	Y	MD5(签名内容+apiKey); 签名内容为 data 所有子字段按字段名升序拼接（剔除&符号及回车和换行符）
支付结果数据(对应 data,采用 json 格式)				
orderId	充值订单号	string	Y	此订单号由 UC 游戏 SDK 生成 ,游戏客户端在进行充值时从 SDK 获得。
gameId	游戏编号	string	Y	由 UC 分配
accountId	账号标识	string	Y	
creator	账号的创建者	string	Y	JY：九游 PP：PP 助手

payWay	支付通道代码	string	Y	统一费率“999”，不提供详细充值方式
amount	支付金额	string	Y	单位：元。**游戏服务端必须根据 UC 回调的金额值进行校验并下发相应价值的虚拟货币。**
callbackInfo	游戏合作商自定义参数	string	Y	游戏客户端在充值时传入，SDK 服务器不做任何处理，在进行充值结果回调时发送给游戏服务器，该字段建议使用字母或数字形式传入，建议不要传入特殊字符。
orderStatus	订单状态	string	Y	S-成功支付 F-支付失败 ,游戏需判断 S 的订单再下发道具
failedDesc	订单失败原因详细描述	string	Y	如果是成功支付，则为空串。
cpOrderId	Cp 订单号	int	N	仅当客户端调用支付方法传入了 transactionNumCP 参数时，才会将原内容通过 cpOrderId 参数透传回游戏服务端。该参数有传递时，才需加入签名，如无，则不需要加入签名

请求示例:

HTTP 请求的 body 内容：

```
{
    "ver": "2.0",
    "data":{
        "orderId":"abcf1330",
        "gameId":123,
        "accountId":"12221222211123",
        "creator":"JY",
        "payWay":1,
        "amount":"100.00",
        "callbackInfo":"custominfo=xxxxx#user=xxxx",
        "orderStatus":"S",
        "failedDesc":"",
        "cpOrderId":"1234567"
    },
    "sign":"6362e564f832d2e8bbcbd50e75409d47"
}
```

假定 apiKey=202cb962234w4ers2aaa

sign 的签名规则：

MD5(accountId=...+amount=...+callbackInfo=...+cpOrderId=...+creator=...+failedDesc=...+gameId=...+orderId=...+orderStatus=...+payWay=...+apiKey)（去掉+；替换...为实际值）

注意：cpOrderId 仅当回调时具备该参数时，才需加入签名，如无，则不需要加入签名，下同

签名原文：

```
accountId=12221222211123amount=100.00callbackInfo=custominfo=xxxxx
#user=xxxxcpOrderId=1234567creator=JYfailedDesc=gameId=123orderId=
abcf1330orderStatus=SpayWay=1202cb962234w4ers2aaa
```

MD5 加密：

```
echo -n
accountId=12221222211123amount=100.00callbackInfo=custominfo=xxxxx
#user=xxxxcpOrderId=1234567creator=JYfailedDesc=gameId=123orderId=
abcf1330orderStatus=SpayWay=1202cb962234w4ers2aaa|md5sum
```

加密结果：6362e564f832d2e8bbcbd50e75409d47

7） 返回内容（文本）：

响应内容	响应内容描述
SUCCESS	成功，表示游戏服务器成功接收了该次充值结果通知，对于充值结果为失败的，只要能成功接收，也应返回 SUCCESS。
FAILURE	失败，表示游戏服务器无法接收或识别该次充值结果通知，如：签名检验不正确、游戏服务器接收失败

结果示例：

```
SUCCESS
```

8) 通知机制：

充值操作完成后，不论是否充值成功，“SDK 服务器”都会将充值结果通过“充值结果回调接口”发送到“游戏服务器”。“游戏服务器”收到“SDK 服务器”的充值通知后，根据处理结果返回字符串 SUCCESS 或 FAILURE。如果返回 SUCCESS，则“SDK 服务器”结束通知任务；如果返回 FAILURE 或由于网络延时导致“SDK 服务器”没有收到任何返回，SDK 服务器将会在周期内进行重复通知。

“游戏服务器”在接收“SDK 服务器”的充值结果通知时，不管订单**是否成功，只要业务逻辑正常验证签名成功，都应该返回 SUCCESS**，表示不需要“SDK 服务器”再次发起通知。当业务逻辑异常（如：收到的 SDK 服务器的充值结果通知内容的签名不正确、充值结果与提交的充值请求不符等），认为需要再次通知，才返回 FAILURE。

【**注意**】：

由于存在多次发送通知的情况，因此“游戏服务器”必须能够正确处理重复的通知。当接收到通知时，需要检查系统内对应业务数据的状态，以判断该通知是否已经处理过。

在对业务数据进行状态检查或处理之前，需要采取数据加锁或时间戳判断等方式进行并发控制。

由于支付网关的通知机制原因，偶尔会发生通知支付失败后又通知支付成功的现象。基于这个情况，“游戏服务器”在处理充值结果通知时，对同一个订单，如果先接收到支付失败，再接收到支付成功的通知，应以成功的支付结果为准，替换原接收到的失败的支付结果。一旦通知支付成功，不会再发生通知支付失败的情形。

1.3.3 扩展数据提交接口

1) **请求地址：**http://sdk.g.uc.cn/ss/

2) **调用方式：**HTTP POST

3) **接口描述：**

收集玩家游戏的数据，与“SDK 客户端“中提供的 submitExtendData 接口作用一致，如已使用服务端接口，可不接入客户端接口，二者必须选其一，建议优先使用服务端接口，放在角色登录后调用

4) **请求方：**游戏服务器

5) **响应方：**SDK 服务器

6) **请求内容（json 格式）：**

字段名称	字段说明	类型	必填	备注
id	请求的唯一标识	long	Y	Unix 时间戳，例：1330395827
service	接口服务名称	string	Y	ucid.game.gameData
data	请求数据	json	Y	json 格式

game	game 参数	json	Y	json 格式 ,字段值均为整数。格式如下：
				{
				“gameId” ：游戏编号
				}

sign	签名参数	string	Y	MD5(签名内容+apiKey) 签名内容： gameData=...sid=.....
请求数据(对应 data,采用 json 格式)				
sid	当前用户会话标识	string	Y	游戏客户端登录后从 SDK 取得
gameData	游戏数据	string	Y	1.gameData 原本是 json 对象 2.gameData 经过 json encode 成为一个字符串 3.该字符串再经过 urlencode 请看示例
category	游戏数据类别	string	Y	
content	游戏数据内容	json	Y	如果玩家数据存在换行符，则要替换成空串

例子：

HTTP 请求的 body 内容：

```
{
    "id":1330395827,
    "service":"ucid.game.gameData",
    "data":{ "sid":"17903ff6-fa8c-4218-85fd-c4a3aa3f3c89177522",
"gameData":"%7B%22category%22%3A%22loginGameRole%22%2C%22content%22%3A%7B%22
roleLevel%22%3A%2288%22%2C%22roleName%22%3A%22%E8%AF%B7%E2%88%9D%E5%86%8D%E7
%BB%99%E6%88%91%E4%B8%80%E6%94%AF%E7%83%9F%22%2C%22zoneName%22%3A%22%E7%BB%8
8%E5%8D%97%E5%B1%B1%E4%B8%8B-%E5%85%B5%E4%B8%B4%E5%9F%8E%E4%B8%8B%22%2C%22ro
leId%22%3A%2253568193%22%2C%22zoneId%22%3A2705%7D%7D"
},
    "game":{"gameId":12345},
    "sign":"bb84860e2812118a88375f9ee4ed931a"
}
```

假如 gameData 的数据为：

```
{"category":"loginGameRole","content":{"roleLevel":"88","roleName":"请∝再给我一支烟","zoneName":"终南山下-兵临城下","roleId":"53568193","zoneId":2705}}
```

那么经过 UrlEncode 后的字符串为（注：该字符串也是作为计算 MD5 签名的原文内容）

```
%7B%22category%22%3A%22loginGameRole%22%2C%22content%22%3A%7B%22roleLevel%22
%3A%2288%22%2C%22roleName%22%3A%22%E8%AF%B7%E2%88%9D%E5%86%8D%E7%BB%99%E6%88
%91%E4%B8%80%E6%94%AF%E7%83%9F%22%2C%22zoneName%22%3A%22%E7%BB%88%E5%8D%97%E
5%B1%B1%E4%B8%8B-%E5%85%B5%E4%B8%B4%E5%9F%8E%E4%B8%8B%22%2C%22roleId%22%3A%2
253568193%22%2C%22zoneId%22%3A2705%7D%7D
```

假定 apiKey=202cb962234w4ers2aaa

sign 的签名规则：MD5(gameData=...sid=...+apiKey)（去掉+；替换...为实际值）

签名原文：

```
gameData=%7B%22category%22%3A%22loginGameRole%22%2C%22content%22%3A%7B%22rol
eLevel%22%3A%2288%22%2C%22roleName%22%3A%22%E8%AF%B7%E2%88%9D%E5%86%8D%E7%BB
%99%E6%88%91%E4%B8%80%E6%94%AF%E7%83%9F%22%2C%22zoneName%22%3A%22%E7%BB%88%E
5%8D%97%E5%B1%B1%E4%B8%8B-%E5%85%B5%E4%B8%B4%E5%9F%8E%E4%B8%8B%22%2C%22roleI
d%22%3A%2253568193%22%2C%22zoneId%22%3A2705%7D%7Dsid=abcdefg123456202cb96223
4w4ers2aaa
```

MD5 加密：

```
echo                                                                    -n
gameData=%7B%22category%22%3A%22loginGameRole%22%2C%22content%22%3A%7B%22rol
eLevel%22%3A%2288%22%2C%22roleName%22%3A%22%E8%AF%B7%E2%88%9D%E5%86%8D%E7%BB
%99%E6%88%91%E4%B8%80%E6%94%AF%E7%83%9F%22%2C%22zoneName%22%3A%22%E7%BB%88%E
5%8D%97%E5%B1%B1%E4%B8%8B-%E5%85%B5%E4%B8%B4%E5%9F%8E%E4%B8%8B%22%2C%22roleI
d%22%3A%2253568193%22%2C%22zoneId%22%3A2705%7D%7Dsid=abcdefg123456202cb96223
4w4ers2aaa|md5sum
```

加密结果：bb84860e2812118a88375f9ee4ed931a

7） 返回内容（json 格式）：

字段名称	字段说明	类型	必填	备注
id	请求的唯一标识	long	Y	Unix 时间戳，与请求内容中 id 值相同。例：1330395827
state	响应状态	json	Y	
data	响应数据	json	Y	
响应状态（对应 state,采用 json 格式）				
code	响应码	int	Y	
msg	结果描述	string	Y	
响应数据(对应 data,采用 json 格式)				

例子：

```
{
    "id":1330395827,
    "state":{"code":1, "msg":"操作已完成"},
    "data":{}
}
```

响应码说明（state.code）：

响应码	说明
1	成功
10	请求参数错误/无效的请求数据,校验签名失败
11	用户未登录(或无效的 sid)

2. 游戏服务器开发要点

“游戏服务器”相关接入开发工作要点如下：

以下 1、2、4 接口为必接

1.实现 sid 验证功能：当用户使用“UC 账号”登录模式时，接收“游戏客户端”提交的 sid，通过调用“SDK 服务器”的“用户会话验证”接口，验证 sid 的合法性，根据需要获取登录用户的账号信息。

2.实现“充值结果回调接口”（包含充值功能时需要实现）：由于充值方式会不断发展，产生新的支付通道，实现“充值结果回调接口”时，应能支持未来新扩展的支付通道。简单地，不应该对接收到的充值结果的支付通道进行限制，只需对用户、游戏、服务器信息时行校验，校验通过则接受收到的充值结果。

3.实现“游戏老账号”登录验证（当游戏需要支持“游戏老账号”登录模式时）：

当用户使用“游戏老账号”登录模式时：

a) “游戏服务器”接收“游戏客户端”提交的游戏老账号、密码，进行登录验证；

b) “游戏服务器”调用“SDK 服务器”的“登录状态同步接口”同步用户登录信息，并获取用户对应的 ucid、sid、用户昵称；

c) “游戏服务器”将获得的 ucid 与游戏老账号进行映射绑定（用户下次使用相应的 ucid 登录时，游戏必须将其识别为同一个用户）；

d) “游戏服务器”返回登录验证结果和 sid 信息给“游戏客户端”。

4.实现“扩展数据提交接口”：收集玩家游戏的数据，与“SDK 客户端“中提供的 submitExtendData 接口作用一致，如已使用服务端接口，可不接入客户端接口，二者必须选其一，建议优先使用服务端接口。