TypeSDK

星渠互联

TypeSDK

产品接入说明书

更新 2017/3/3

目录

TypeSDK 接入介绍

文档适用范围

本文适用于游戏开发者接入 TypeSDK 开发文档。TypeSDK 接入需要完成游戏客户端和游戏服务端两部分接入。需要开发团队协同完成。

涉及技术范围

TypeSDK 客户端在 Unity3D 环境完成接入，虽然 TypeSDK 渠道接入逻辑使用 Andorid 原生环境开发，但若用户无二次开发需求，可不用关心此部分技术实现。
TypeSDK 服务端使用标准 http 协议作为接口协议。
TypeSDK 打包工具使用 C#、nodejs 开发。用户若无二次开发需求，可不用关心此部分技术实现。

TypeSDK 接入指引

本文将指引用户完成整个 TypeSDK 接入工作

获取渠道参数

完成游戏渠道签约

您需要完成您的游戏与至少一家渠道的签约工作。获取渠道后台登录账号及渠道参数，请务必对您的渠道账号及参数保密，不可泄露，在整个 TypeSDK 使用和支持过程中，TypeSDK 工作人员不会询问您的渠道账号和密码。

TypeSDK 安装

完成 TypeSDK 打包工具安装

您需要在你的开发环境完成 TypeSDK 打包工具安装，并完成了渠道打包测试。
了解如何安装 TypeSDK 打包工具

完成 TypeSDK 服务端安装

您需要完成 TypeSDK 服务端在外网环境的安装，并整理服务端 URL 接口列表，以备服务端接入使用。
了解如何安装 TypeSDK 服务端

同步渠道参数

同步打包工具和服务端的配置参数

您需要再次确认打包工具中项目和渠道参数配置正确，之后利用打包工具同步渠道参数至服务端。
了解如何同步渠道参数

配置渠道管理后台

在您需要测试的渠道商提供的后台内，填写支付回调地址。如有服务器白名单限制也请配置 TypeSDK 服务端 IP 地址。
了解如何配置渠道后台

客户端接入

接入准备

请客户端接入程序员准备好 TypeSDK Unity 客户端 Package 文件、客户端接入文档。
了解如获取 TypeSDK

进行客户端 TypeSDK 接入开发工作

按客户端接入文档要求进行 TypeSDK 客户端的接入工作，接入前务必确认游戏服务端接入开发人员与游戏客户端开发人员，已理解个接口流程，并已经协商好客户端与服务端之间传送 SDK 请求的通讯协议。
了解如接入 TypeSDK 客户端

服务端接入

接入准备

请服务端接入程序员准备好 TypeSDK 服务端 URL 地址列表、服务端接入文档。
了解如获取 TypeSDK 服务端列表

调试

进行游戏服务端的 TypeSDK 接入开发工作

按客户端接入文档要求进行 TypeSDK 客户端的接入工作，接入前务必确认游戏服务端接入开发人员与游戏客户端开发人员，已理解个接口流程，并已经协商好客户端与服务端之间传送 SDK 请求的通讯协议。
了解如何接入 TypeSDK 服务端

导出客户端 Android 项目

客户端完成接入后，需将 Unity 导出 Andorid 项目。以供打包工具作为游戏基础版本。
了解如获得 Andorid 项目

将 Android 项目导入 TypeSDK 打包工具

将 Andorid 导入到 TypeSDK 打包工具中，打包工具可对每次导入的游戏项目建档管理。
了解如何导入 Andorid 项目

利用 TypeSDK 打包工具出渠道 APK

使用 TypeSDK 打包工具，将选择上传的打包工具，打出渠道 APK，下载安装至手机准备测试。
了解如何使用 TypeSDk 打包工具出 Andorid 渠道 APK

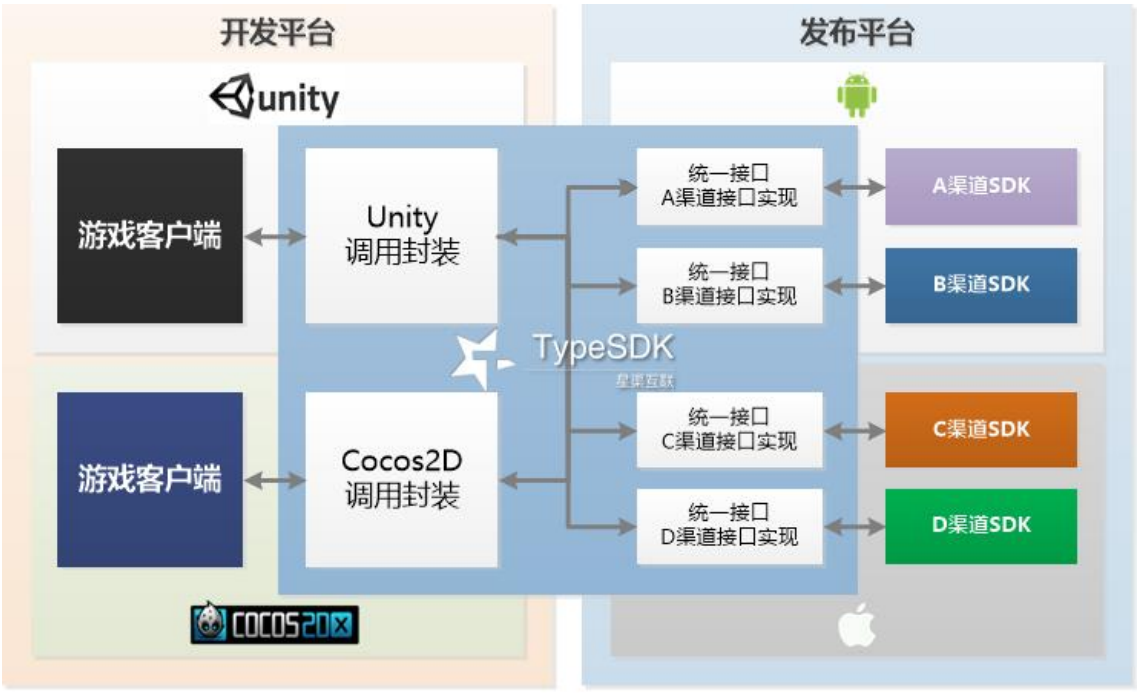
运行 APK 测试

在手机上运行 APK，按照 TypeSDK 测试用例进行渠道 APK 测试。
了解如何进行 APK 测试

客户端接入

渠道 SDK 统一接口原理

使用 TypeSDK 的技术人员都很关心一个问题，TypeSDK 是如何做到统一渠道 SDK 接口，并完成 Andorid 和 iOS 双平台发布的。
图：TypeSDK 聚合发布原理



通过上图，我们可用发现 Unity 游戏开发者如需使用 TypeSDK，只需要实现 TypeSDK Unity 部分的统一调用。而错综复杂的多渠道跨平台逻辑都交由 TypeSDK 进行处理。可用很大程度上节省开发者的接入时间，并且能使得开发者能跟关注聚焦游戏内容，而无需为无数 SDK 接入费神。以上内容与接入无直接关系。

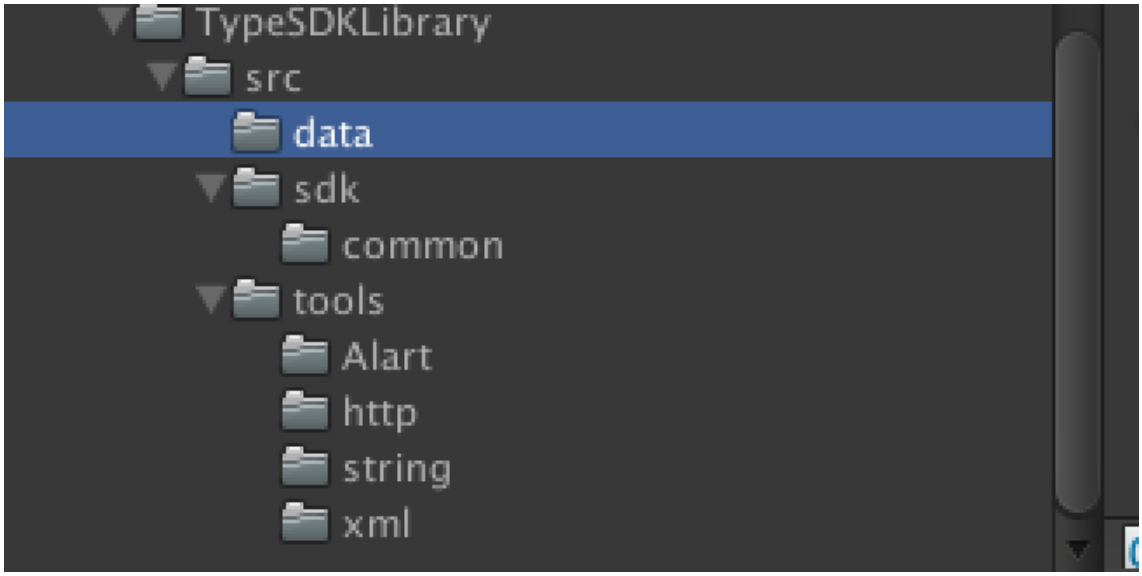
客户端接入准备工作

在客户端接入 TypeSDK 前，你需要从 TypeSDK 官网下载 TypeSDK 提供的 Unity 部分接入资源。如果您的开发环境没有外网，还建议您将接入文档及 Unity 项目 Demo 下载并复制到您的开发环境，文档和 Demo 中提供了大量的范例代码，您可更具需要直接复制和修改，帮助您快速完成编码工作。
了解如何下载 TypeSDK

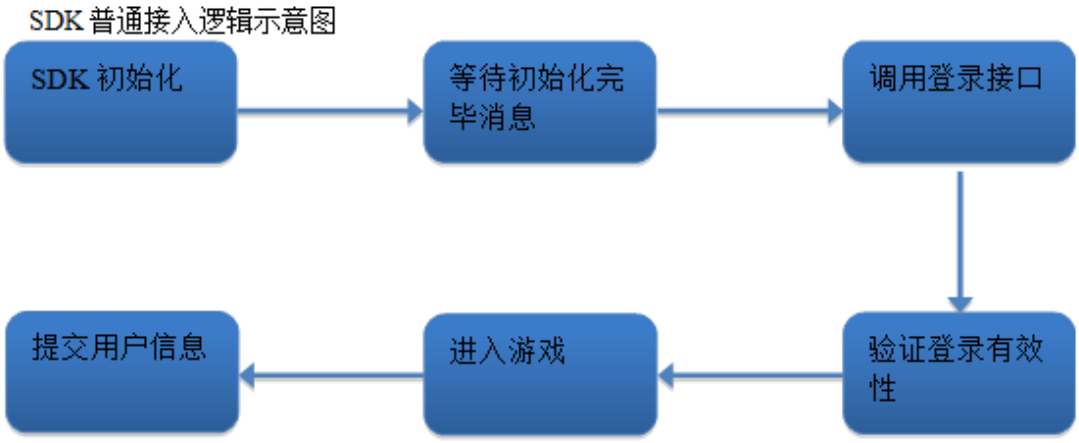
导入资源

将 TypeSDKLibrary 文件夹整个添加到 Unity 工程中，注意结构需要与截图统一。

如出现脚本命名冲突的文件，请修改游戏中冲突的类名
添加后的目录结构如下所示



客户端启动后流程描述



1. 在调用其它 API 前需先调用初始化接口对 SDK 进行初始化。
2. 在等待完成初始化接口调用后方可调用登录接口
3. 登录成功选择角色进入游戏后需要调用提交用户信息接口。

登录成功 TypeSDK 服务器会将关键用户数据返回游戏服务器。请务必将此信息通过提交用户信息反馈至 TypeSDK 客户端。TypeSDK 需要此信息完成用户信息补全，否则可能出现用户无法支付等情况。另外请在用户登录成功进入服务器、创建角色、充值下单、角色升级时，调用提交用户信息，部分渠道需要在游戏过程中实时采集用户信息，如此接口遗漏渠道会做拒包处理。

4. 登录成功后如果需要切换账号，需调用登出接口，并非所有渠道都支持用户账号切换，故不建议在游戏设置按钮主动进行账号切换。
5. 登录成功后需要监听登录成功回调、重新登录成功回调。收到回调说明用户在渠道 SDK 内操作了用户切换。需要游戏完成游戏账号操作。
6. 在期望退出时游戏需要调用退出接口。

调用接口和回调函数

1. 调用接口

函数名称	功能说明
InitSDK	SDK 初始化
Login	显示登录界面
Logout	执行渠道登出，清理渠道用户缓存
PayItem	显示支付界面
ShowPersonCenter	显示用户中心（如果渠道支持）
ExitGame	大退游戏（等于杀进程）
IsHasRequest	判断渠道是否支持某功能（例如：用户中心）

UpdatePlayerInfo	提交用户信息到渠道（渠道要求在进入游戏后调用）

2. 回调函数

回调侦听的事件类型	事件说明
TypeEventType.EVENT_INIT_FINISH	SDK 初始化完毕
TypeEventType.EVENT_LOGIN_SUCCESS	登录成功回调
TypeEventType.EVENT_PAY_RESULT	支付结果回调
TypeEventType.EVENT_LOGOU	登出完毕回调
TypeEventType.EVENT_RELOGIN	重新登录成功回调
TypeEventType.EVENT_CANCEL_EXIT_GAME	取退出应用回调

typesdkbaselib 中已经提供了预先注册完毕所有回调侦听的 cs 文件：U3DTypeEventListener.cs 接入开发者需要在 U3DTypeEventListener 监听类中，根据不同回调事件实现相应的游戏逻辑。
也可以根据自己需要，在游戏内合适的地方，参考 demo 自行注册相关的函数侦听。

注：部分需要界面响应的监听，应判断当时场景，避免造成游戏逻辑混乱。如，在 PVP 时监听到用户切换账号回调。

TypeSDK 基础数据类型

U3DTypeBaseData

TypeSDK 用到的数据类型继承于 U3DTypeBaseData,您可用使用此对象及其方法，获取或创建 TypeSDK 传递数据对象。

1. 创建 U3DTypeBaseData 对象

```
1. U3DTypeBaseData  baseData = new U3DTypeBaseData();
```

2. 设置一条属性

函数名	public void SetData(string attName,string attValue)	
参数列表	string attName,	U3DTypeAttName 中定义的字段
	string attValue	目前支持 int，string，boolean 三种类型基本数据
返回值	Void	

示例：

```
1. baseData.SetData(“键名”,”值”); （键名使用 U3DTypeAttName 定义的字段）
```

3. 获得一个 string 类型的属性 attname 为标识

函数名	public string GetData(string attName)	
参数列表	string attName,	U3DTypeAttName 中定义的字段
返回值	String	指定 key 值的 string 类型 value

4. 获得一个 int 类型的属性 attname 为标识

函数名	public int GetInt (string attName)	
参数列表	string attName,	U3DTypeAttName 中定义的字段
返回值	Int	指定 key 值的 int 类型 value

5. 获得一个 bool 类型的属性 attname 为标识

函数名	public bool GetBool(string attName)	
参数列表	string attName,	U3DTypeAttName 中定义的字段
返回值	Bool	指定 key 值的 boolean 类型 value

使用例子

//新建一个对象

```
1. U3DTypeBaseData egData = new U3DTypeBaseData ();
2. //给该对象赋值
3. egData.SetData (U3DTypeAttName.APP_KEY, "123456789");
4. //读取一个 string 类型数据 readStr = "123456789
5. string readStr = egData.GetData (U3DTypeAttName.APP_KEY);
6. //读取一个 int 类型数据 readInt
7. int readInt = egData.GetInt(U3DTypeAttName.APP_KEY);  = 123456789
```

响应消息传递类型

U3DTypeEvent

TypeSDK 通过 U3DTypeEvent 对象传递响应消息传递的消息类型

```
1. public delegate void U3DTypeEventDelegate( U3DTypeEvent evt);
```

事件对象例子

获取 SDK 用户登录成功的消息。

```
1. void LoginResult(U3DTypeEvent evt)
2. {
3. U3DTypeBaseData data = evt.evtData;
4. string userID = data.GetData(U3DTypeAttName.USER_ID);
5. string userToken = data.GetData(U3DTypeAttName.USER_TOKEN);
6. }
```

详细接口

1. SDK 初始化接口

接口函数：

```
1. public void InitSDK()
```

sdk 的初始化接口，再调用其他 sdk 功能前，请务必先执行该接口，所有渠道都要求在应用启动开始就调用此接口。
调用例子

```
1. U3DTypeSDK.Instance.InitSDK();
```

回调函数：

```
1. //初始化完成后回调函数
2. void NotifyInitFinish(U3DTypeEvent evt)
3. {
4. //游戏需要等待此回调出现后才允许在游戏逻辑内调用登录接口。
5. }
```

```
1. //更新渠道更新检测完成后回调
2. void NotifyUpdateFinish(U3DTypeEvent evt)
3. {
4. //建议等待此更新完成后，再进行游戏的更新逻辑，否则会造成渠道更新和游戏自身更新冲突。
5. }
```

2. 调用登录窗口

接口函数：

```
1. public void Login ()
```

显示登录界面，若登录成功则会发送 TypeEventType.EVENT_LOGIN_SUCCESS 消息。
请在登录界面自动执行调用，不可出现需要点击按钮才显示的情况。
当用户登录失败时需要，再次调用此接口。
调用例子

```
1. U3DTypeSDK.Instance.Login();
```

回调函数：

```
1. //登录操作完成后的回调函数
2. void NotifyLogin(U3DTypeEvent evt){
3. //解析渠道登录成功返回的信息，一般有 user_token,user_id...
4. //此时返回的结果不能作为登录依据，需要进过服务端验证，取的服务器返回的最终数据。
5. //CP 方需要将信息解析为 CP 服务器约定的数据格式转发给游戏服务器，并由游戏服务器转发至 TypeSDK Server 以完成游戏的登录验证
6. string userId = evt.evtData.GetData(U3DTypeAttName.USER_ID);
7. }
```

部分渠道返回不提供 USER_ID，并且此处的返回属于不可信范围，真正 USER_ID 需要通过游戏服务端完成用户验证获取。

3. 提交用户信息

接口函数：

```
1. public void UpdatePlayerInfo ()
```

在有些指定事件，需要设置用户相关信息并且提交。登录完成进入游戏、用户升级、建角。该函数的所有内容，不能使用 SDK 客户端本地缓存的数据，建议从服务端获得

需要设置的属性如下，当没有该属性时，请传空字符串

```
1. string ROLE_TYPE = "create_role";//角色统计信息类型即调用时机，（createRole:创建角色，levelUp:角色升级，enterGame:选定角色进入游戏，不能为空字符串）
2. string SAVED_BALANCE = "0";//当前角色余额（RMB 购买的游戏币），默认为 0
3. string USER_NAME = "user_name"; //用户名
4. string USER_TOKEN = "user_token"; //游戏服务端在完成登录 token 验证后，可能会收到渠道返回的授权 token。需要将授权 token 返回游戏客户端，并通过此参数提交 TypeSDK 客户端。
5. string USER_ID = "user_id"; //用户 id
6. string USER_HEAD_ID = "user_head_id"; //用户头像 id
7. string USER_HEAD_URL = "user_head_url"; //用户头像 url
8. string VIP_LEVEL = "vip_level";//VIP 等级，没有传 0
9. string PARTY_NAME = "party_name";//工会名称,如：天下第一
10. string ROLE_ID = "role_id"; //角色 id
11. string ROLE_NAME = "role_name"; //角色名字
12. string ROLE_LEVEL = "role_level"; //角色等级
13. string ROLE_CREATE_TIME = "role_create_time";//角色创建时间，一定要服务器时间（单位/秒）
14. string ROLE_LEVELUP_TIME = "role_levelup_time";//角色升级时间（单位/秒）
15. string ZONE_ID = "zone_id"; //所在大区 id
16. string ZONE_NAME= "zone_name"; //所在大区名称
17. string SEVER_ID = "server_id"; //所在服务器 id
18. string SERVER_NAME = "server_name";//所在服务器名称
```

调用例子

```
1. //设置所有的数据
2. U3DTypeSDK.Instance.GetUserData().SetData(U3DTypeAttName.USER_ID,"123");
```

```
U3DTypeSDK.Instance.GetUserData().SetData(U3DTypeAttName.USER_NAME,"user1342137");
U3DTypeSDK.Instance.GetUserData().SetData(U3DTypeAttName.ROLE_ID,"c1276481");
U3DTypeSDK.Instance.GetUserData().SetData(U3DTypeAttName.ROLE_NAME,"无敌二三");
U3DTypeSDK.Instance.GetUserData().SetData(U3DTypeAttName.ROLE_LEVEL,"32");
// .....其他需要设置的信息
// 提交数据
U3DTypeSDK.Instance.UpdatePlayerInfo();
```

回调函数：
无

4. 支付

接口函数：

```
public string PayItem (U3DTypeBaseData _in_pay);
```

在获取服务器生成订单号，并在 TypeSDK Server 服务器提交订单信息后，调用此接口启动渠道的支付界面，进行用户支付行为。
调用例子
//创建一个订单信息

```
U3DTypeBaseData payData = new U3DTypeBaseData();
//用户 ID，渠道返回，没有填空字符串
payData.SetData(U3DTypeAttName.USER_ID,userData.GetData(U3DTypeAttName.USER_ID));
//用户 token，登录验签完成后由游戏服务端返回，没有填空字符串
payData.SetData(U3DTypeAttName.USER_TOKEN,userData.GetData(U3DTypeAttName.USER_TOKEN));
//商品支付价格（单位：分）
payData.SetData(U3DTypeAttName.REAL_PRICE,"100");
//商品名称，不要出现空格和特殊字符。
payData.SetData(U3DTypeAttName.ITEM_NAME,"skbi");
//商品数量
payData.SetData(U3DTypeAttName.ITEM_COUNT,"1");
//所在服务器 id（如果没有填“0”）
payData.SetData(U3DTypeAttName.SERVER_ID,"1");
//所在服务器名字（如果没有填“server_name”）
```

```
payData.SetData(U3DTypeAttName.SERVER_NAME,"安卓一区");
//所在大区 id（如果没有填“0”），注意应用宝要求：账户分区 ID_角色 ID。每个应用都有一个分区 ID 为 1 的默认分区，分区可以在 cpay.qq.com/mpay 上自助配置。如果应用选择支持角色，则角色 ID 接在分区 ID 号后用"_"连接，角色 ID 需要进行 urlencode。payData.SetData(U3DTypeAttName.ZONE_ID,"1");
//所在大区名字（如果没有填“server_name”）
payData.SetData(U3DTypeAttName.ZONE_NAME,"华北一区");
//内部订单号（必须填写，并保证多区情况下，订单号唯一）
payData.SetData(U3DTypeAttName.BILL_NUMBER,"NO_123456");
//商品 id（需和 TypeSDK Server 商品列表保持一致）
payData.SetData(U3DTypeAttName.ITEM_SEVER_ID,"id");
//传递的额外参数（建议传入需要用来做订单标识的信息）
payData.SetData(U3DTypeAttName.EXTRA,"extra");
//商品描述，不要出现空格和特殊字符串
payData.SetData(U3DTypeAttName.ITEM_DESC,"1000 元宝，送 1000 元宝");
//玩家在游戏中的角色 ID
payData.SetData(U3DTypeAttName.ROLE_ID,"role_1234");
//玩家在游戏中的角色名字
payData.SetData(U3DTypeAttName.ROLE_NAME,"玩家编号 001");
U3DTypeSDK.Instance.PayItem (U3DTypeBaseData _in_pay);
```

回调函数：

```
 //支付结果通知回调，CP 需根据支付返回结果完成相应逻辑。
void NotifyPayResult(U3DTypeEvent evt){
if (evt.evtData.GetData(U3DTypeAttName.PAY_RESULT).Equals("1"))
{//支付成功
Debug.Log("pay finished:" + evt.evtData.GetData(U3DTypeAttName.PAY_RESULT_DATA));
}
else
{//支付失败，或取消。
Debug.Log("pay failed:" + evt.evtData.GetData(U3DTypeAttName.PAY_RESULT_REASON));
}
}
```

此处为客户端返回结果，不可作为支付到账依据，游戏需等待 TypeSDK Server 的支付成功回调。

部分渠道没有客户端支付回调，所以请不要在回调内写关键逻辑代码。否则您的游戏会被卡在等待支付回调上。

event 中 data 的参数包括以下内容

PAY_RESULT//支付结果（1/0/2）成功 / 失败(除取消)/支付取消

说明：客户端收到“失败或支付取消”状态，建议客户端可以使用户直接发起下笔充值。

PAY_RESULT_REASON//支付结果的原因（失败原因）
PAY_RESULT_DATA//支付结果的返回数据

5. 登出账号

接口函数：

```
1. public void Logout ();
```

调用渠道的登出账号逻辑，不会有界面显示，但是会把渠道账户注销
调用例子：

```
1. U3DTypeSDK.Instance.Logout()
```

回调函数：
注意：用户使用渠道浮标进行登出操作，也会触发此回调函数，需要清理当前用户数据并返回游戏登录界面，调用登录界面等待用户重新登录。

```
1. //登出结果通知回调，说明用户已经做了退出账户操作，
2. void NotifyLogout(U3DTypeEvent evt){
3. //需要返回登录界面，并重新调用登录
4. }
```

6. 请求关闭应用

接口函数：

```
1. public void ExitGame ();
```

用户需要退出游戏时,如按退出键，游戏不应直接退出游戏，需调用渠道退出接口，将退出确认行为交由渠道执行。游戏的真实退出需要执行的 逻辑可放在 OnDestroy()中执行。
调用例子：

```
1. //查询当前渠道是否有退出确认窗口
2. if (U3DTypeSDK.Instance.IsHasRequest(“support_exit_window”))
3. {
4. //如渠道有退出确认窗口，则直接调用 ExitGame
5. U3DTypeSDK.Instance.ExitGame();
6. }else{
7. //如渠道无退出确认窗口，建议弹出游戏的确认界面
8. //
9. if (GUI.Button(new Rect(50,250,200,30), "我要退出"))
10.{
11.//用户确认要退出后，执行 ExitGame
12.U3DTypeSDK.Instance.ExitGame();
13.}
14.}
```

回调函数：

```
1. //取消退出游戏通知回调，如果用户在渠道确认退出界面取消了退出，则触发此回调。
2. void NotifyCancelExit(U3DTypeEvent evt){
3. }
```

7. 获取渠道参数

接口函数：

```
1. public U3DTypeBaseData GetPlatformData()
```

初始化后可随时调用，已获得渠道相关数据。
CHANNEL_ID//渠道的 id，
CP_ID//应用开发者自己的 id（用作内部多游戏辨识）
渠道参数客户端无需关系，但游戏服务端需要渠道参数，完成与 TypeSDK Server 的通信。

调用范例:

```
1. U3DTypeBaseData platformData = U3DTypeSDK.Instance.GetPlatformData()
2. String channelID = platformData..GetData(U3DTypeAttName.CHANNEL_ID))
3. String cpID = platformData..GetData(U3DTypeAttName.CP_ID))
```

8. 获取用户参数

接口函数：

```
1. public U3DTypeBaseData GetUserData()
```

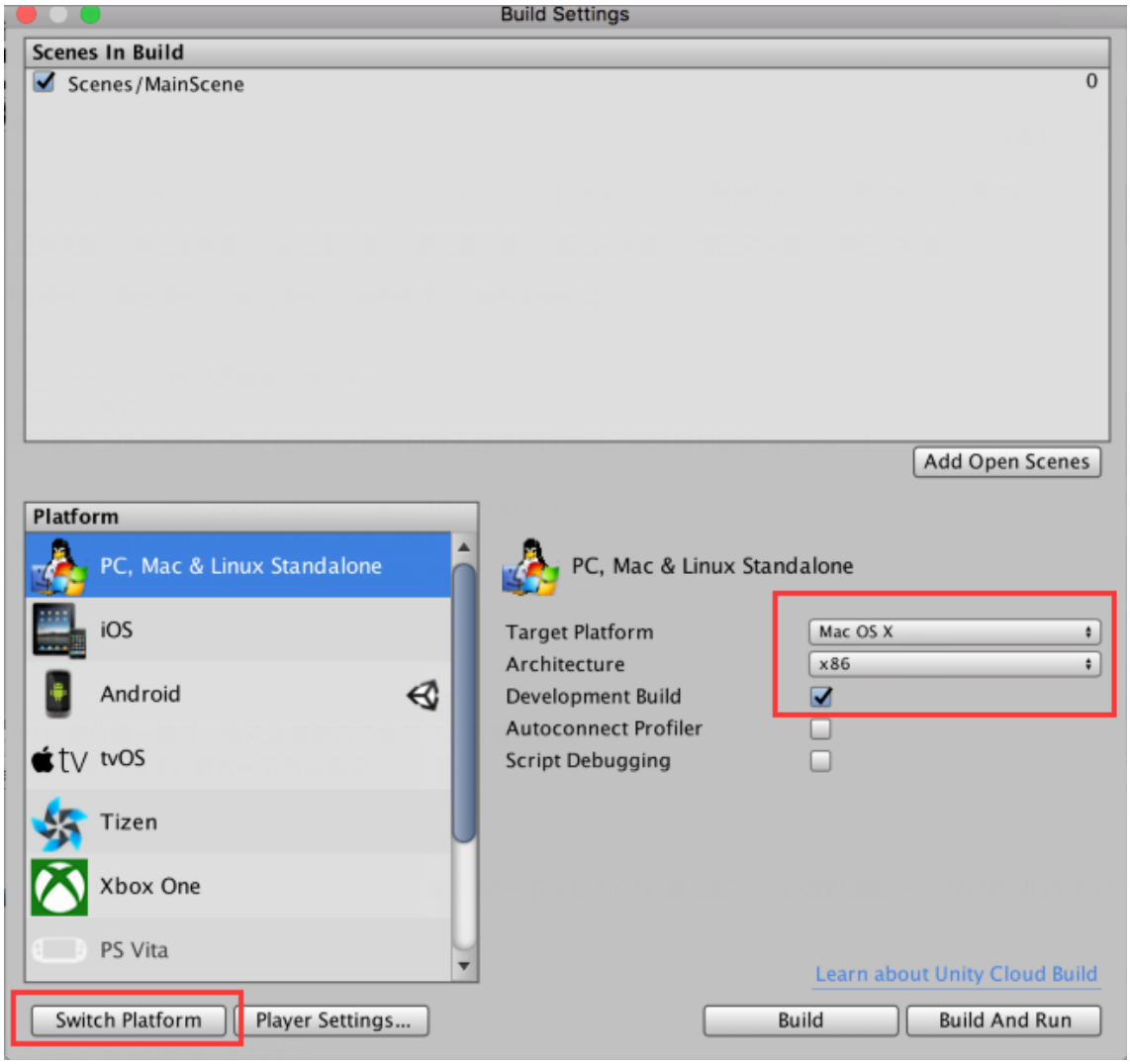
收到用户登录成功回调后可随时调用，已随时获取用户相关数据。
获取用户信息数据 其中包含了
string USER_TOKEN = "user_token"; //用户渠道验证用 token
string USER_ID = "user_id"; //用户 id
登录成功回调接口已经包含了用户参数，在处理登录回调接口时无需再使用此接口获取用户参数

调用范例:

```
1. U3DTypeBaseData userData = U3DTypeSDK.Instance.GetUserData()
2. String userId = userData..GetData(U3DTypeAttName.USER_ID))
3. String userToken = userData..GetData(U3DTypeAttName.USER_TOKEN))
```

运行调试方法

应开发资源只包含 TypeSDK Unity 部分统一接口，故无法直接在 Unity 开发环境进行渠道调试。
为了放方便用户进行调试，TypeSDK Unity 资源内集成了一套测试逻辑可帮助用户跑通 SDK 流程测试。



切换至 Winodws 平台后，即可直接 Unity 使用开发界面的播放按钮进行测试。

导出 Android 项目

打包工具所使用的项目

TypeSDK 打包工具所使用的游戏项目与其他产品有所不同，为了满足开源需求并同时降低用户使用时的技术成本，TypeSDK 没有使用对 APK 母包反编译后替换文件方式进行渠道包编译。而是使用 Unity 导出的 Android 项目和 iOS 项目整合渠道 SDK 原始代码进行编译。

不使用反编译的原因

渠道说 NO

如果渠道知道游戏开发商没有使用渠道官方提供的 SDK，而使用了由第三方提供进行了反编译和修改的渠道 SDK 后。其后果可想而知，游戏下架公司拉黑，如果因此造成渠道用户手机被黑造成经济损失，开发商还会摊上官司。所以为了提现 TypeSDK 产品诚意，让游戏开发商安心使用 TypeSDK 产品，我们坚持完全开源、且对渠道 SDK 反编译说 No。

程序员说 NO

使用编译方法能简化聚合 SDK 发布复杂度，但会给使用者造成技术门槛。反编译项目如遇问题，通常会出现闪退、黑屏等状况，而程序员无法判断问题原因，只能等待聚合 SDK 工具开发者来解决。试想临近上线提包，却因为包有问题毫无办法，只能期望第三方能及时响应解决问题。

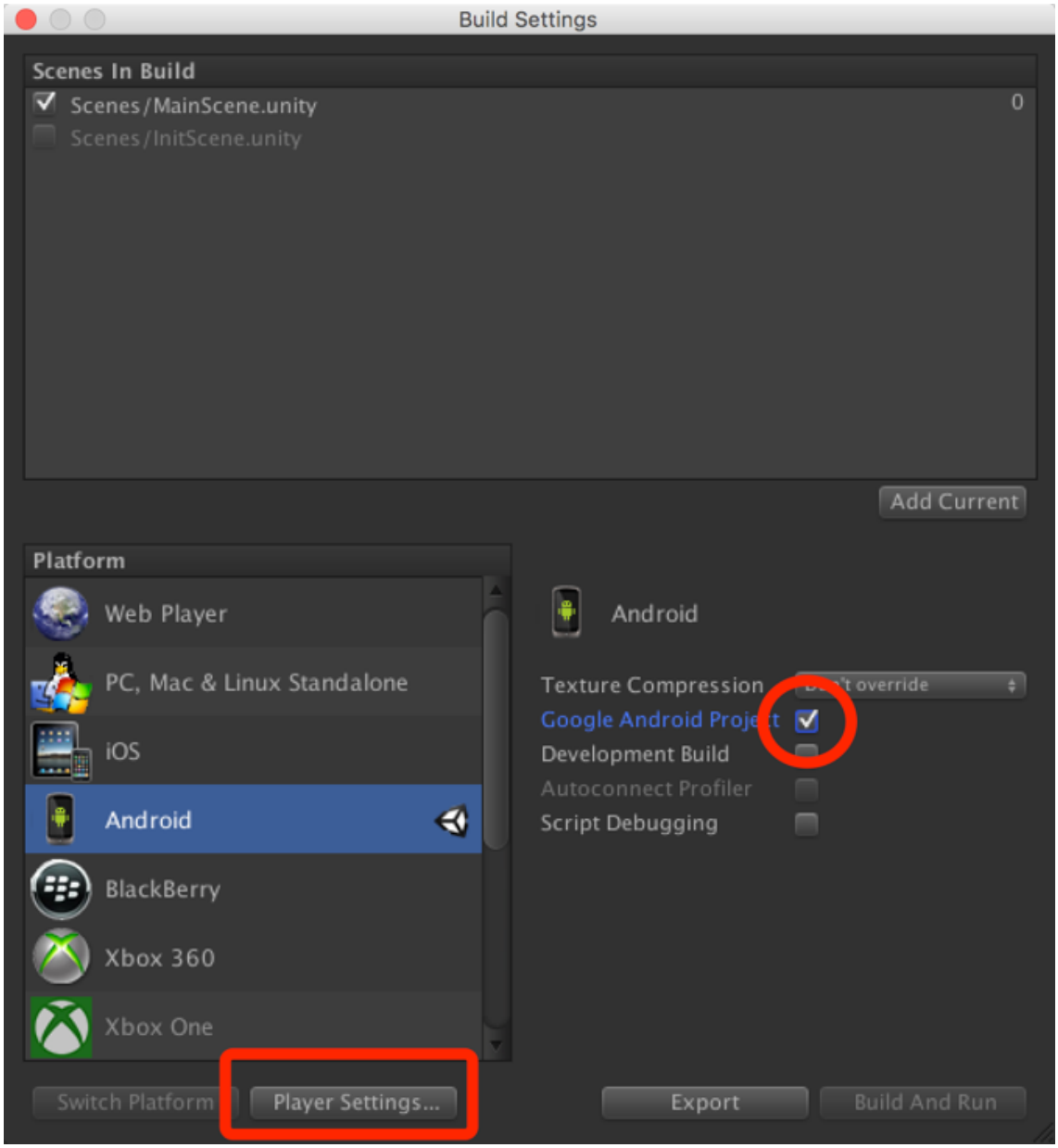
老板说 NO

反编译方案无法做到真正开源，即使号称服务端和工具代码开源，用户也必须依赖聚合 SDK 开发者提供的渠道反编译文件合成渠道包。用户使用时间再久也只能依靠第三方，无法做到真正掌握系统。试想公司即将上市，技术审计时被发现有关键系统还掌握在别人手里，自己无法维护更新。这时老板一定想劈了 CTO，而 TypeSDK 使用开源常规方案,使用者可轻易的掌握其工作原理，并可自行维护更新，消化整合成随心所欲。这也是多家上市公司或有计划上市公司选择 TypeSDK 产品的原因。

Unity 导出 Android 项目

1.点击菜单栏的 Unity 按钮（mac os 系统请点击 file 按钮），会出现如下菜单

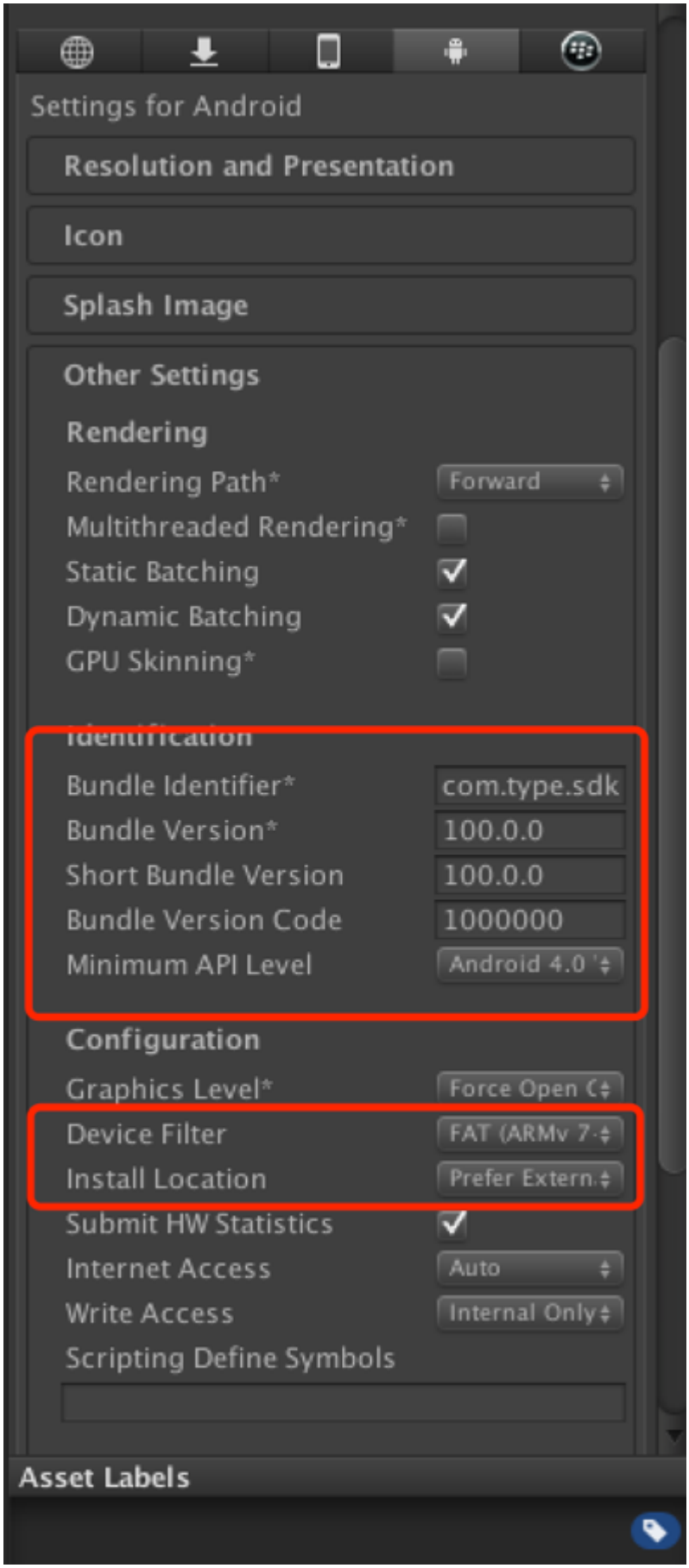
2.点击 buildSettings 选项，会看到如下图的界面



导出安卓项目需要做以下几步

- 先点击安卓平台，
- 点击 switch platform 切换到安卓平台的配置，然后会在界面右侧看到安卓的相关选项。
- 在右侧右侧安卓选项中 google android project 选项不勾选只能导出 apk，勾选后可以导出 Andorid 项目工程。如若要使用 typesdk 打包平台一键式打包，请务必勾选该选项导出 Android 项目工程。

3.点击 buildsettings 的 Playsettings 按钮，会在系统的 inspector 分栏中看到如下界面

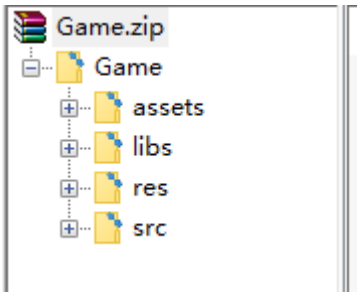

Settings for Android
Resolution and Presentation
Icon
Splash Image
Other Settings
Rendering
Rendering Path*
Forward
Multithreaded Rendering*
Static Batching
Dynamic Batching
GPU Skinning*
Bundle Identifier*
com.type.sdk
Bundle Version*
100.0.0
Short Bundle Version
100.0.0
Bundle Version Code
1000000
Minimum API Level
Android 4.0
Configuration
Graphics Level*
Force Open
Device Filter
FAT (ARMv 7
Install Location
Prefer Extern
Submit HW Statistics
Internet Access
Auto
Write Access
Internal Only
Scripting Define Symbols
Asset Labels

有关需要注意和修改的地方，已经用红框标示出

- **a.bundle identifier** 俗称包名，应用的身份标识，使用默认的 unity 包名无法导出，如需导出 apk，请使用自定义包名，如需使用 typesdk 一键式打包平台，对 unity 中包名没有特殊需求，每个渠道的包名会在打包平台上做相应配置，并最终替换。
- **bundle version** 应用版本号，玩家可以使用第三方工具看到的游戏 apk 版本号，例如 1.0.0.a
- **shortbundleversion**，安卓设置中同上，ios 中该版本号要求不能使用超过 3 位数字的设定，例如 a.b.c(正确)，a.b.c.d(错误)。
- **bundle version Code** 应用版本序列号（必须是 int 类型），该版本号玩家看不到，渠道对应用是否有更新版本判断的主要依据，如果 version code 不同，就会认为是不同版本的应用，渠道会要求用户做更新。
- **Minimum API Level** 支持的最低安卓版本（建议使用 4.0 级以上，4.0 以下版本系统将会有很多功能无法使用，例如：推送）
- **Device Filter** 设备兼容性（armv7,x86）请相关开发者自行决定，x86 架构设备主要都是使用英特尔的 cpu，与 arm 架构芯片在图形渲染方面会有很多不同。
- **Install Location** 安装位置，sd 卡 / 内部存储 / 自动（那里有空间安装在那里，优先安装 sd 卡）建议设置成 auto

压缩项目

将导出项目的总目录名改成 Game。并使用 zip 压缩这个 Game 目录。完成后获得 Game.zip 文件待用。
完成后的压缩文件如下图。

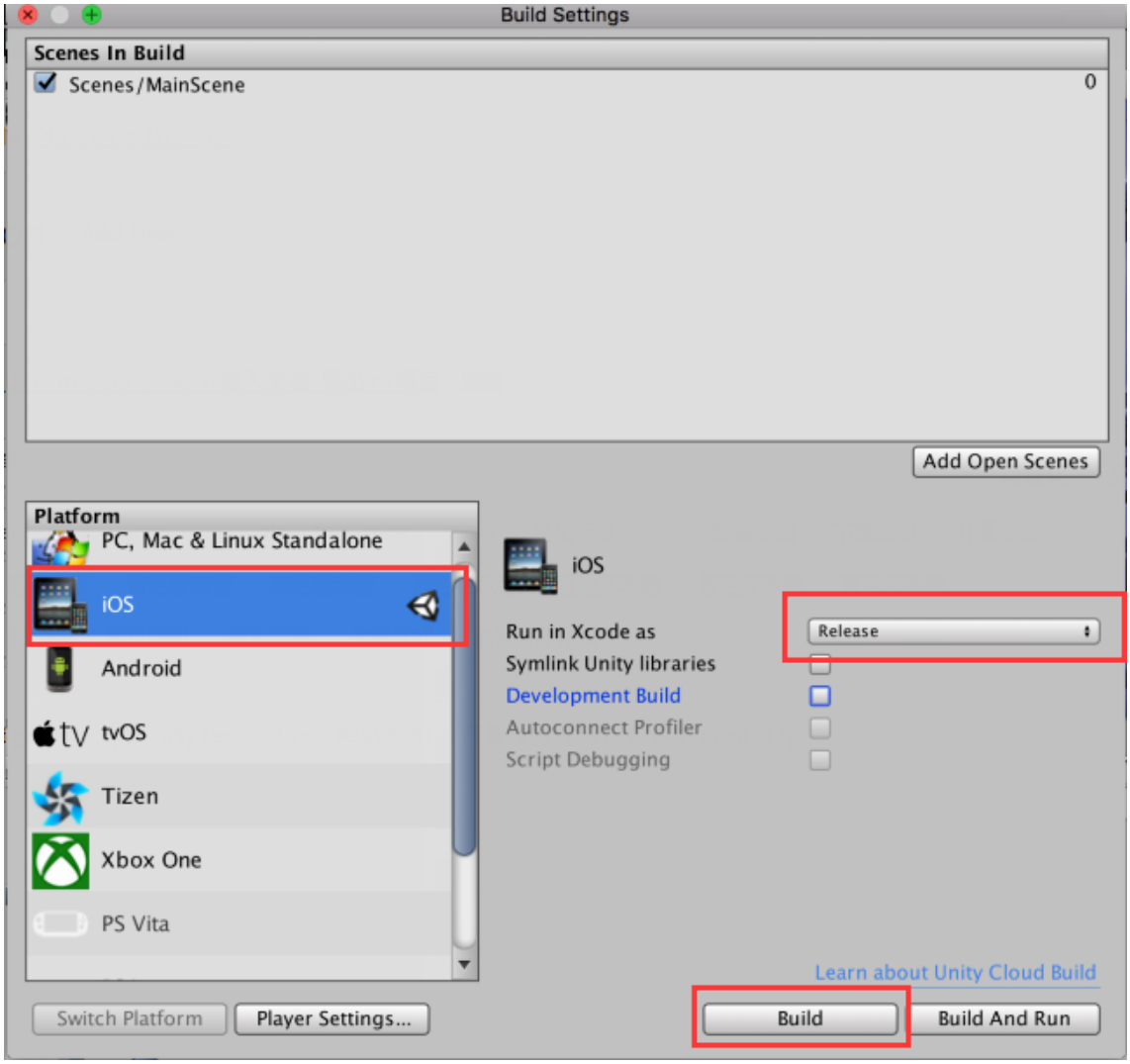


导出 iOS 项目

导出操作

iOS 渠道项目导出相对较为简单。
1.点击菜单栏的 Unity 按钮（mac os 系统请点击 file 按钮），会出现如下菜单

2.Platform 选择 iOS，Run in Xcode as 选择 Release，最后点击 Build。



压缩项目

将导出项目的总目录名改成 Game。并使用 zip 压缩这个 Game 目录。完成后获得 Game.zip 文件待用。

服务端接入

TypeSDK 服务端聚合原理

相对 TypeSDK 客户端和打包工具，TypeSDK 服务器端聚合原来相对简单，即使用一套标准接口完成与游戏服务端的对接，并判读每个请求的渠道标识完成与各个渠道服务端的对接。因为服务端不区分 Andorid 与 iOS，故只每款游戏只需要一套服务端环境。

服务端接入准备工作

TypeSDK 服务端接入需要服务端与客户端配合完成，请与接入前与客户端 SDK 接入程序员做好充分沟通工作。

协议说明

一、通讯协议

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 JSON 方式）向 SDK 服务器发起接口请求。

二、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 表示数据。

2. 字符编码

请求与相应内容均采用 UTF-8 字符编码。

3. 签名规则

请求和响应中的签名均使用 md5 哈希进行，算法如下：

MD5(签名内容 + ”|” + apiKey)

说明：

MD5 使用 RFC1321 标准，编码后需转换成全小写。
描述签名的表达式中，”+”表示做字符串连接，实际产生的待签名字符串中并不存在。
签名内容指各接口请求数据中若干字段的拼接。基本格式为各字段值以 ”|” 符号分隔后直接连接。注意，由于”|”符号用作分隔字段，签名内容中需避免出现该符号，换行符(回车或换行)等特殊符号也需要预先剔除。如果对应字段为空，仍然需要保留“|”符号占位。
计算 MD5 签名时，应以 UTF8 编码取字符串的字节值。
appid 及 apiKey 由打包工具分配，打包工具使用方法请参考使用文档。

范例：
假设请求数据为：

“data:{

“id” : 123,

“name” : “test”

“value” : “something”

“other” : “blarblar”

}

要求的签名内容为:

id + name + value

则拼接后得出要签名的内容串为

123|test|something

假定 apiKey=aabbcc，则需要进行 MD5 哈希的字符串为：

123|test|something|aabbcc

appid、channelid

appid

TypeSDK 所使用的 appid 与渠道分配的 appid 不是通一个，请勿混淆。

TypeSDK 所用 appid 用来在多游戏公用 TypeSDK 的情况下区别游戏。设置于打包工具各个渠道参数中 cp_id 字段，建议一款游戏分配一个 id。游戏客户端可用使用获取游戏信息接口获取到 cp_id，并在请求时传递给游戏服务端。

channelid

channelid 被用来区别当前渠道，已经默认配置于打包工具各个渠道参数页面内 channel_id 字段中，属于固定字请勿进行修改。
游戏客户端可用使用获取游戏信息接口获取到 channel_id，并传在请求时传递给游戏服务端。

在服务端配置渠道参数

如何同步打包工具和服务端参数

接口说明

一、用户会话验证

1.请求地址：http://192.168.0.1:3000/{appid}/{channelid}/Login/

说明：URL 中的{appid}代表游戏代码，由打包工具生成，{channelid}代表渠道代码，渠道代码列表可以参考打包工具说明，可以从客户端提交的参数中获取当前渠道代码。例：http://192.168.0.1:3000/1000/1/Login/

2.**调用方式**：HTTP POST

3.**接口描述**：

验证用户登录结果。

1. 游戏客户端通过 SDK 客户端的登录动作获取用户登录信息。
2. 游戏客户端将获取的用户登录信息传送至游戏服务端。
3. 游戏服务端通过本请求将用户登录信息传送到 SDK 服务端，验证该登录信息是否有效。
4. SDK 服务端返回验证结果及其他信息，供游戏服务器使用。

4.**请求方：游戏服务端**

5.**响应方：SDK 服务端**

6.**请求内容（JSON 格式）：**

字段名称	字段说明	类型	备注
id	用户唯一标识	string	对应渠道的用户 ID。并非必传，未作说明的情况下传空字符串。
token	用户登录会话标识	string	本次登录标识。并非必传，未作说明的情况下传空字符串。
data	附加信息	JSON	附加信息。并非必传，根据渠道不同，该字段含义不同，未作说明的情况下传空字符串。
sign	签名参数	string	MD5(签名内容 + ”|” + apiKey)签名内容： Id + ”|” + token + ”|” + data

7.**返回内容（JSON 格式）：**

字段名称	字段说明	类型	备注
code	响应码	int	本次请求结果标志
id	用户唯一标识	string	对应渠道的用户 ID
nick	用户在渠道的昵称	string	对应渠道的用户昵称
token	用户登录会话标识	string	本次登录标识
msg	响应信息	string	如果请求出错，描述错误信息。

value	渠道返回信息	JSON	渠道返回的原始结果信息。

响应码说明：0：渠道正常返回，结果成功

1：渠道正常返回，结果失败

2：渠道服务端请求错误

-1：提交的请求参数错误

-2：提交的请求转换成渠道参数错误

-3：提交的请求参数签名错误

-99：未知错误

id,nick,token 说明：

根据不同渠道定义的返回字段不同，此三个字段不一定有值。渠道未返回对应字段时，该字段值为空字符串。

要求服务端务必将此三个字段，传递至客户端，并要求客户端在 SDK 上传用户信息时，传入此三个字段。否则可能造成部分渠道无法支付

二、支付结果回调

1.**请求地址：**

该地址为充值结果通知地址，由游戏服务端在下文的 SaveOrder 接口中通过 notifyurl 字段提交至 SDK 服务端。

2.**调用方式：**HTTP POST

3.**接口描述：**

通知用户充值结果。

1. 用户在游戏中向 SDK 客户端提交充值请求。
2. SDK 客户端将充值请求转发渠道方
3. 渠道方异步执行充值。
4. 渠道方将充值结果发送给 SDK 服务端
5. SDK 服务端通过该接口将充值结果发送给游戏服务端。
6. 游戏服务端处理充值逻辑。
7. 游戏服务端向 SDK 服务端返回处理结果。
8. SDK 服务端向渠道方返回处理结果。

4.**请求方：**SDK 服务端

5.响应方：游戏服务端

6.请求内容（JSON 格式）：

字段名称	字段说明	类型	备注
code	响应码	int	渠道返回的充值结果。
id	用户唯一标识	string	对应渠道的用户 ID。
order	渠道订单号	string	渠道返回的订单号。
cporder	CP 订单号	string	游戏客户端在提交订单时传送的内部订单号。如果该渠道未接收该参数，则该字段为空字符串。
info	订单附加信息	string	游戏客户端在提交订单时传送的附加信息。如果该渠道未接收该参数，则该字段为空字符串。
sign	签名参数	string	MD5(签名内容 + ”|” + apiKey)签名内容： code + ”|” + id + ”|” + order+ ”|” + cporder + ”|” + info
amount	订单金额	string	该笔订单价值折算为人民币的金额（以分为单位）供服务端校验使用，不参与签名。

7.返回内容（JSON 格式）：

字段名称	字段说明	类型	备注
code	响应码	int	本次请求结果标志
msg	响应信息	string	如果请求出错，描述错误信息。

响应码说明：0：正常返回，结果成功

1：正常返回，结果失败

-99：未知错误

8.推荐处理方式

游戏服务端收到该请求后可保存该次请求参数，随即返回 code=0，表明成功收到消息。之后异步处理充值或发放道具逻辑。

三、充值信息提交

1.请求地址：http://192.168.0.1:3000/{appid}/{channelid}/SaveOrder/

说明：URL 中的{appid}代表游戏代码，由打包工具生成，{channelid}代表渠道代码，渠道代码列表可以参考打包工具文档，可以从客户端提交的参数中获取当前渠道代码。

例： http://192.168.0.1:3000/1000/1/SaveOrder/

2.调用方式：HTTP POST

3.接口描述：

充值信息存档待查。

1. 用户在游戏中向游戏服务端提交充值请求。
2. 游戏服务端生成内部充值订单号及相关充值信息
3. 游戏服务端将内部充值订单号及相关充值信息返回游戏客户端，供其提交给渠道方。
4. 游戏服务端异步将内部充值订单号，该笔订单回调 url 及相关充值信息发送给 SDK 服务端。
5. SDK 服务端将充值信息存档待查并返回处理结果。

4.请求方：游戏服务端

5.响应方：SDK 服务端

6.请求内容（JSON 格式）：

字段名称	字段说明	类型	备注
cporder	CP 订单号	string	游戏客户端在提交订单时传送的内部订单号。
data	订单信息	string	由 CP 生成订单时自定义的附加信息,不能为空及空字符串。

sign	签名参数	string	MD5(签名内容 + ”|” + apiKey)签名内容： cporder + ”|” + data
notifyurl	订单回调 url	string	该笔订单回调通知游戏服务端的 url，不参与签名。
verifyurl	订单查询 url	string	该笔订单向游戏服务端查询详情的 url，不参与签名。

7.返回内容（JSON 格式）：

字段名称	字段说明	类型	备注
code	响应码	int	本次请求结果标志
msg	响应信息	string	如果请求出错，描述错误信息。

响应码说明：0：正常返回，存档成功

1：正常返回，存档失败

-1：系统错误

-2：参数错误

-3：签名校验错误

-99：未知错误

注意：SaveOrder 接口在服务端生成内部订单号时请求。只有获取到该请求成功返回，才能允许客户端作后续充值动作。

四、充值信息确认

1.请求地址：http://192.168.0.1:3000/{appid}/{channelid}/CheckOrder/

说明：URL 中的{appid}代表游戏代码，由打包工具生成，{channelid}代表渠道代码，渠道代码列表可以参考打包工具文档，可以从客户端提交的参数中获取当前渠道代码。

例： http://192.168.0.1:3000/1000/1/CheckOrder/

2.调用方式：HTTP POST

3.接口描述：

充值信息查询。

1. SDK 服务端向游戏服务端转发充值结果回调。
2. 游戏服务端向 SDK 发送充值信息查询请求，目的是确认该充值结果有效性。
3. SDK 服务端根据提交的内部充值订单号查询充值信息并返回游戏服务端。
4. 游戏服务端根据查询结果进行逻辑处理。

说明：部分渠道提供信息查询接口，本接口将优先使用渠道的信息查询接口处理请求。如果该渠道没有提供信息查询接口，则查询 3.3.3 充值信息提交 接口中保存的充值信息，如果创建充值信息时没有调用该接口，或者没有查询到目标订单对应的充值信息，则会返回未查询到相应充值信息。

4.请求方：游戏服务端

5.响应方：SDK 服务端

6.请求内容（JSON 格式）：

字段名称	字段说明	类型	备注
cporder	CP 订单号	string	游戏客户端在提交订单时传送的内部订单号。
sign	签名参数	string	MD5(签名内容 + ”|” + apiKey)签名内容： cporder

7.返回内容（JSON 格式）：

字段名称	字段说明	类型	备注
code	响应码	int	本次请求结果标志
msg	响应信息	string	如果请求出错，描述错误信息。
value	订单详细信息	JSON	根据渠道不同，返回相应订单信息。

响应码说明：

0：正常返回，获取订单信息成功

1：正常返回，没有获取到订单信息

2：正常返回，获取订单信息错误

-1：系统错误

-2：参数错误

-3：签名校验错误

-99：未知错误

五、游戏订单查询

1.请求地址：

该地址为订单查询地址，在 SaveOrder 接口中通过 verifyurl 字段提交至 SDK 服务端。

2.调用方式：HTTP POST

3.接口描述：

SDK 服务端向游戏服务端查询收到的订单信息。
用户在游戏中向 SDK 客户端提交充值请求。
SDK 客户端将充值请求转发渠道方
渠道方异步执行充值。
渠道方将充值结果发送给 SDK 服务端
SDK 服务端通过该接口向游戏服务端查询该充值请求是否合法。
游戏服务端处理查询逻辑。
游戏服务端向 SDK 服务端返回查询结果。
SDK 服务端比对订单信息并依此确定下一步处理流程。

4.请求方：SDK 服务端
5.响应方：游戏服务端

6.请求内容（JSON 格式）：

字段名称 字段说明 类型 备注
code 操作类型 string 预留字段，区分本次查询操作类型。目前统一传 0
id 用户唯一标识 string 对应渠道的用户 ID。
order 渠道订单号 string 渠道返回的订单号。
cporder CP 订单号 string 游戏客户端在提交订单时传送的内部订单号。如果该渠道未接收该参数，则该字段为空字符串。
info 订单附加信息 string 游戏客户端在提交订单时传送的附加信息。如果该渠道未接收该参数，则该字段为空字符串。

sign 签名参数 string MD5(签名内容 + ”|” + apiKey)
签名内容：

code + ”|” + id + ”|” + order+ ”|” + cporder + ”|” + info

7.返回内容（JSON 格式）：

字段名称 字段说明 类型 备注
code 响应码 int 本次请求结果标志
msg 响应信息 string 如果请求出错，描述错误信息。
id 用户唯一标识 string 对应渠道的用户 ID。
order 渠道订单号 string 渠道返回的订单号。
cporder CP 订单号 string 游戏客户端在提交订单时传送的内部订单号。如果该渠道未接收该参数，则该字段为空字符串。
amount 订单金额 string 该笔订单价值折算为人民币的金额（以分为单位）。
createtime 订单创建时间 string 该笔订单创建时间。
Itemid 道具 id string 该笔订单的道具 id，如果没有传空字符串。
(该字段标识订单中的商品 ID，需要与客户端下订单时对应的字段匹配，验证对比不符时不发货)

Itemquantity 道具数量 Int 该笔订单道具数量，没有传 0。
（该字段标识客户端提交的订单中的道具数量，渠道不接受该字段时，客户端提交的订单没有该字段，此时直接传 0 或 1 均可）

status 订单状态 int 订单状态码。
info 其他信息 string 备用字段，传送其他可供比对的信息。

响应码说明：
0：正常返回，结果成功

1：正常返回，结果失败

-99：未知错误

8.推荐处理方式

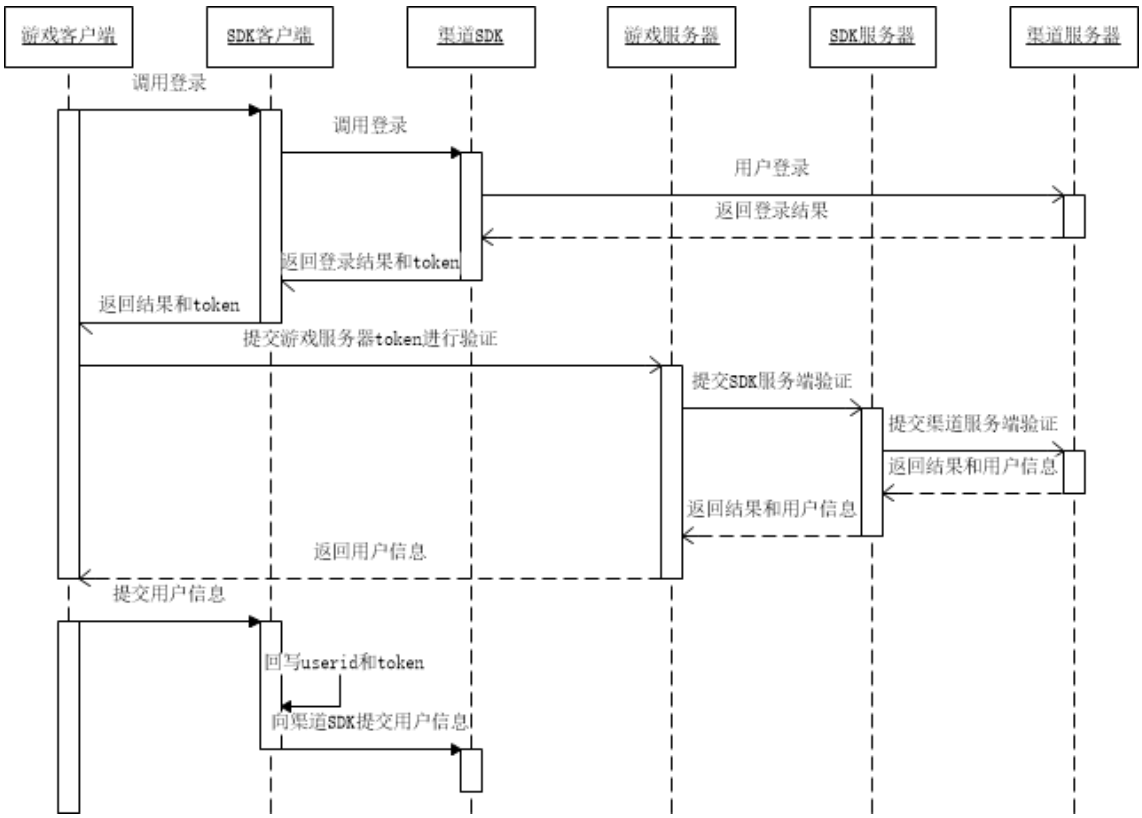
游戏服务端收到该请求后优先以 CP 订单号为条件查询，查询不到或请求中没有 CP 订单号时以渠道订单号为条件查询，找到匹配的订单信息并同步返回 SDK 服务端。

登录流程

游戏客户端调用 TypeSDK 登录接口，TypeSDK 调起渠道 SDK 支付窗口。
用户完成渠道登录后，渠道 SDK 返回一组以供服务端验证用以验证的数据，通常为 user_id、token。
游戏客户端自 TypeSDK 登录成功回调中获取，此组验证数据后，提交至游戏服务端期望进行渠道端验证。

游戏服务端利用 TypeSDK 服务端封装的会话验证接口完成数据验证，数据同时返回用户资料，游戏利用此资料进行用户登录逻辑。
服务端获得返回信息中的 id、token。需要通过游戏客户端用户信息提交接口回写到 TypeSDK 中，否则部分渠道无法进行支付。

登录流程图



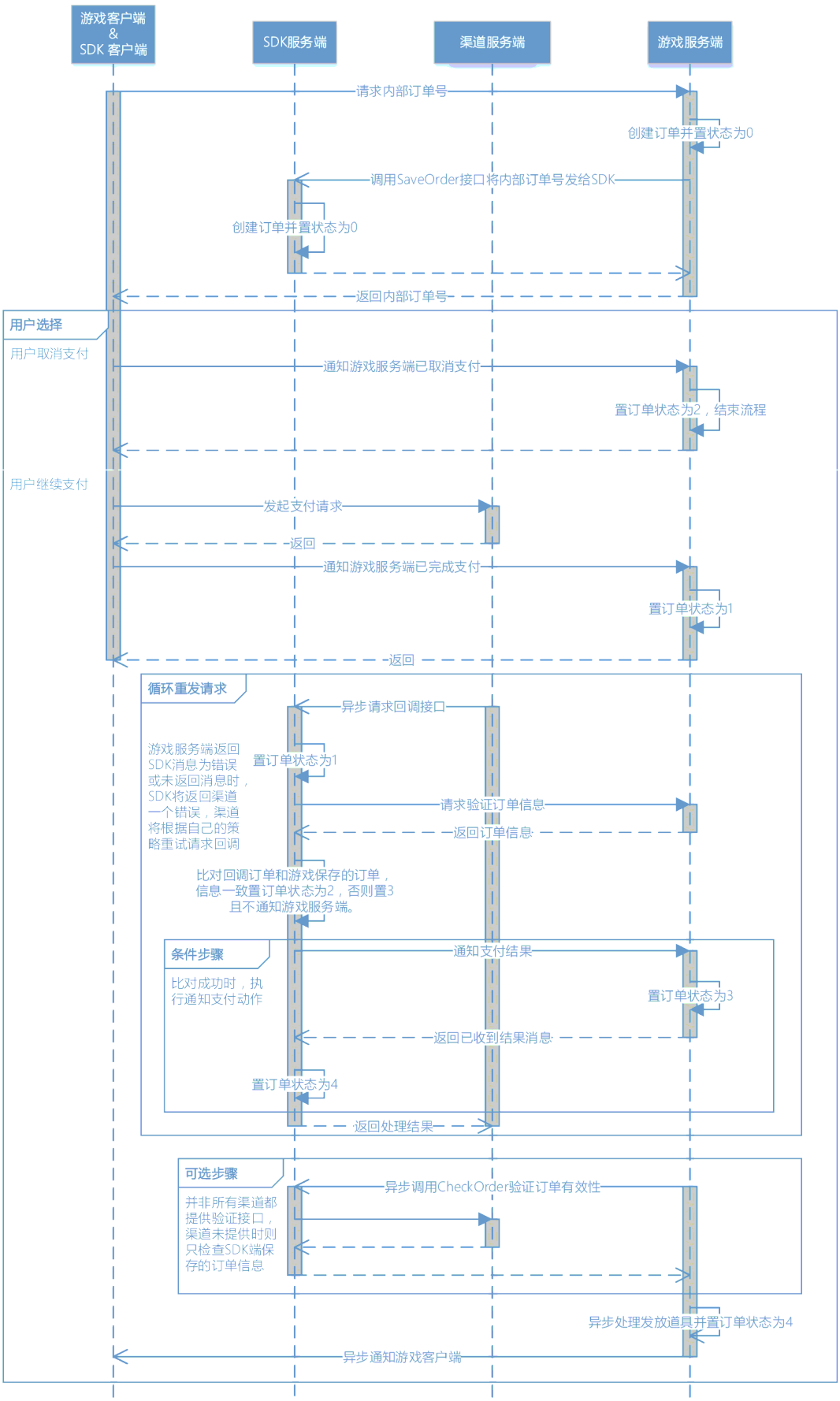
支付流程

游戏客户端在每次用户点击购买时向服务端请求生成内部订单。并需要采用特定机制（例如一定时间内禁止连续点击购买）防止用户频繁操作对服务器造成过高负载。

游戏服务端生成的所有内部订单需要存储待查。并在得到渠道返回的外部订单后异步处理发货操作并以特定机制通知客户端更新数据显示。

渠道支付接口负责完成货币交易操作，生成并存储外部订单，供对账查询使用。

SDK 服务端转发请求时额外存储一份订单日志数据，存储内部订单号，外部订单号及订单状态，供对账及查找 BUG 时作为参考。


游戏客户端
&
SDK 客户端
SDK服务端
渠道服务端
游戏服务端
请求内部订单号
创建订单并置状态为0
调用SaveOrder接口将内部订单号发给SDK
创建订单并置状态为0
返回内部订单号
用户选择
用户取消支付
通知游戏服务端已取消支付
置订单状态为2，结束流程
用户继续支付
发起支付请求
返回
通知游戏服务端已完成支付
置订单状态为1
返回
循环重发请求
异步请求回调接口
游戏服务端返回SDK消息为错误或未返回消息时，SDK将返回渠道一个错误，渠道将根据自己的策略重试请求回调
置订单状态为1
请求验证订单信息
返回订单信息
比对回调订单和游戏保存的订单，信息一致置订单状态为2，否则置3且不通知游戏服务端。
条件步骤
比对成功时，执行通知支付动作
通知支付结果
置订单状态为3
返回已收到结果消息
置订单状态为4
返回处理结果
可选步骤
并非所有渠道都提供验证接口，渠道未提供时则只检查SDK端保存的订单信息
异步调用CheckOrder验证订单有效性
异步处理发放道具并置订单状态为4
异步通知游戏客户端

约定

- 支付相关接口内部订单号字段长度不能超过 10 位，格式使用英文字母和数字的组合，需要能够区分区服。不可重复。
- 渠道返回的用户 id 用于用户唯一标识。单区服内不可重复。
- 支付接口返回的 amount 是当次支付产生的实际金额。

安全设计

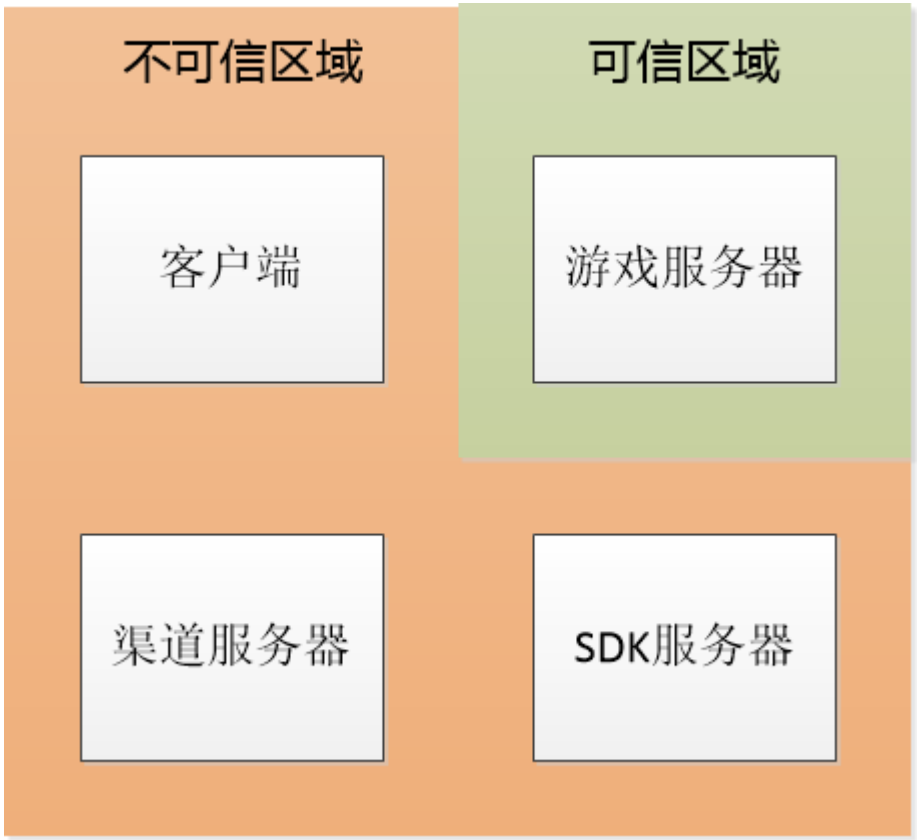
作为以盈利为目的游戏，安全设计极为重要，虽然开发者利用了大量客户端加密、通讯加密方法加密游戏逻辑和协议，但在集成第三方 SDK 过程中往往会疏忽安全设计。此文将介绍开发商在接入 TypeSDK 过程中所需要注意的安全设计。

可能存在的风险

1. 客户端被破解修改。
2. 手机内存被修改。
3. 网络通讯被劫持修改。

可信区域和不可信区域

可信区域：游戏服务端
不可信区域：游戏客户端、渠道服务器、TypeSDK 服务器



客户端

1. 用户登录
完成登录后作为由第三方客户端直接返回的结果数据，存在被串改的可能，故不可直接使用，需要进行验证。

2. 支付
完成支付后 TypeSDK 通常会进行支付结果回调(部分渠道不提供回调，不可依赖客户端回调进行刷新数据逻辑，否则可能出现流程卡死)，但回调仅表示支付完成，不能做为支付成功结果依据。返回结果也可能被篡改。需要等待游戏服务器接受到支付成功回调后才能确认。

服务端

1. 用户登录验证
客户端提供的用户信息可能被篡改，需要通过 TypeSDK 服务器对登录 token 进行验证，因为是同步请求，并且有签名校验，游戏服务端可用相信验证返回获取用户信息。

2. 支付结果回调
在支付成功后 TypeSDK 服务端将支付结果回调给游戏服务器，虽然 TypeSDK 服务端会进行渠道签名、金额验证核对，但游戏服务器可能收到伪造的 TypeSDK 回调，故游戏服务器需要利用 TypeSDK 服务端提供的验证接口进行订单核实，并再次核对金额。