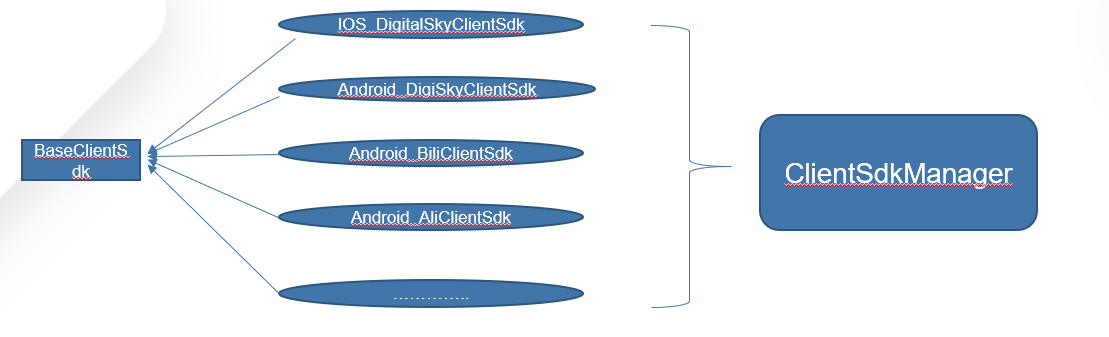
# 项目版本管理之多渠道打包

茜色项目组之小耗子

项目版本管理是游戏开发中必不可少的一部分，也是保证游戏平稳上线的基础保证。针对版本管理，本次主要分享一下平台管理和多渠道打包两个部分。

## 一．平台管理

游戏发布一定会有很多平台的包，比如百度，华为，阿里等等，而每一个平台的SDK的方法又不尽相同，对此，我们需要对所有的平台SDK进行统一管理。BaseClientSdk，作为所有平台SDK的基类，包括所有平台的基础属性以及共同方法。在此基类上，我们可以拓展出任一个平台的SDK，比如Android\_BaiduClientSdk、Android\_BiliClientSdk、Android\_HuaweiClientSdk等等，每个具体的SDK类中具体实现共同方法，也可以实现自己独有的私有方法等等， 这么多平台的SDK，必须有一个管理类ClientSdkManager，管理类统一处理SDK初始化，



我们需要自定义平台宏标签，来区分该包是哪个平台的，具体如何定义后面再讲，在ClientSdkManager管理类中有统一的初始化SDK方法，初始化的同时就把具体当前SDK的类就实例化保存在管理类中，下次要获取当前SDK的话，直接获取管理类中的SDK实例。

## 二．多渠道打包

基于上面的平台管理，我们引出一个渠道概念，平台其实记录的是SDK，SDK是唯一的，而渠道不一定唯一，平台与渠道的关系不是一对一，可能是一对多，比如官网SDK，运维可能需要老用户渠道包，微博渠道，QQ群渠道的包等等。因此每个渠道所需要的数据来源也不同，比如Appid，热更新地址，公告地址等等，有的渠道需要GM，各个渠道支付方式也不一样，所以考虑这些都走配置，数据驱动开发，这样代码也看起来干净很多，减少很多不需要if else。

既然所有渠道数据都要走配置，那么就需要区分渠道配置，需要唯一的键，而我们项目组游戏又是同区同服，因此，我们的包定位是只有一个渠道和一个服，渠道加服务器类型的键，这样生成的配置，能准确有效地包含大部分的需求。我们在游戏中定义了三个重要枚举，平台类型eSdkPlatformType、渠道类型eChannelId和服务器类型eServerType，如下：



图1 平台类型枚举

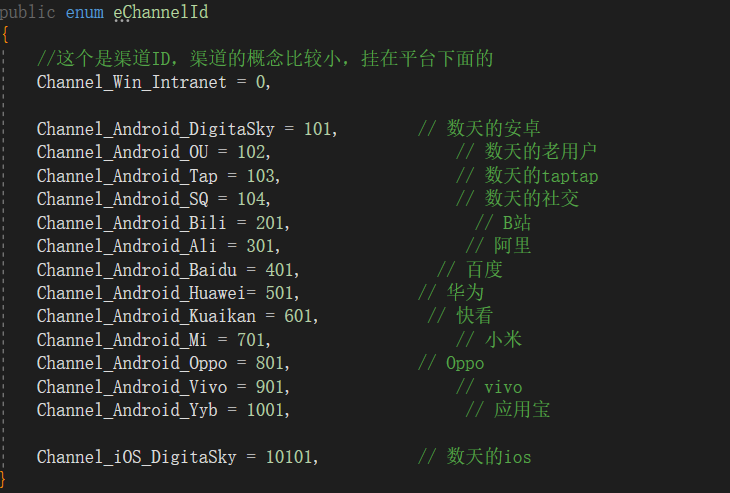


图2 渠道类型枚举

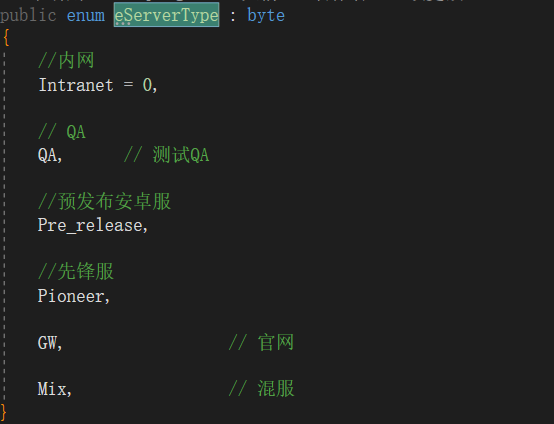


图3 服务器类型枚举

在游戏启动脚本上定义三个序列化变量，记录当前版本的状态，这个状态对于打包也很重要，如下：



图4 序列化变量记录版本状态

我们通过上面的变量切换平台时，在Editor中同时更新平台宏标签，代码如下：

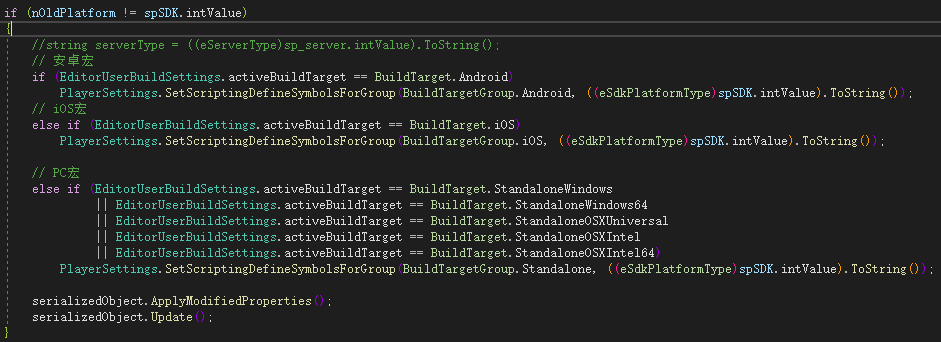
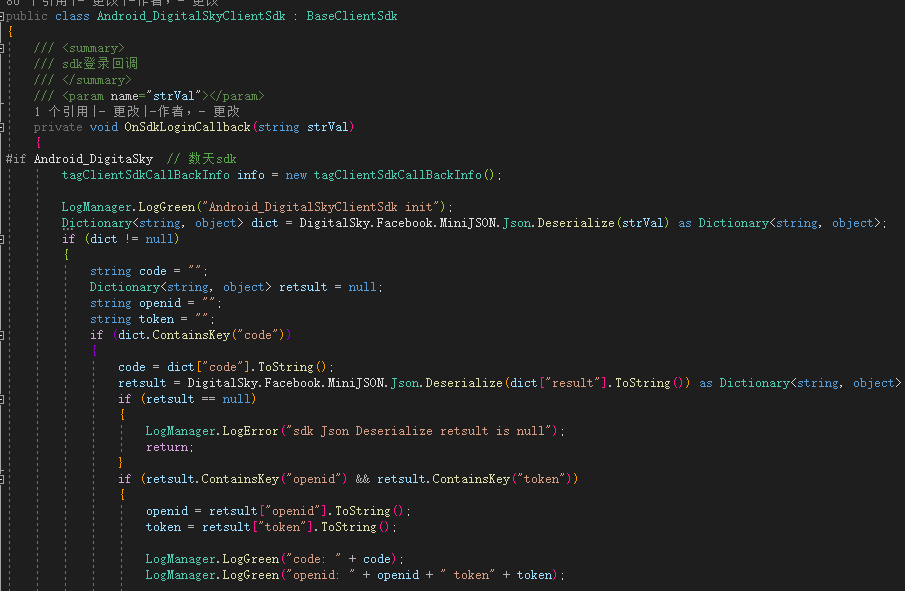


图5 更新宏标签代码

通过宏标签定义，可以灵活动态地切换管理平台代码，打开某平台宏标签后，但是还没有导入平台SDK的UnityPackage包，需要报错提醒，比如：



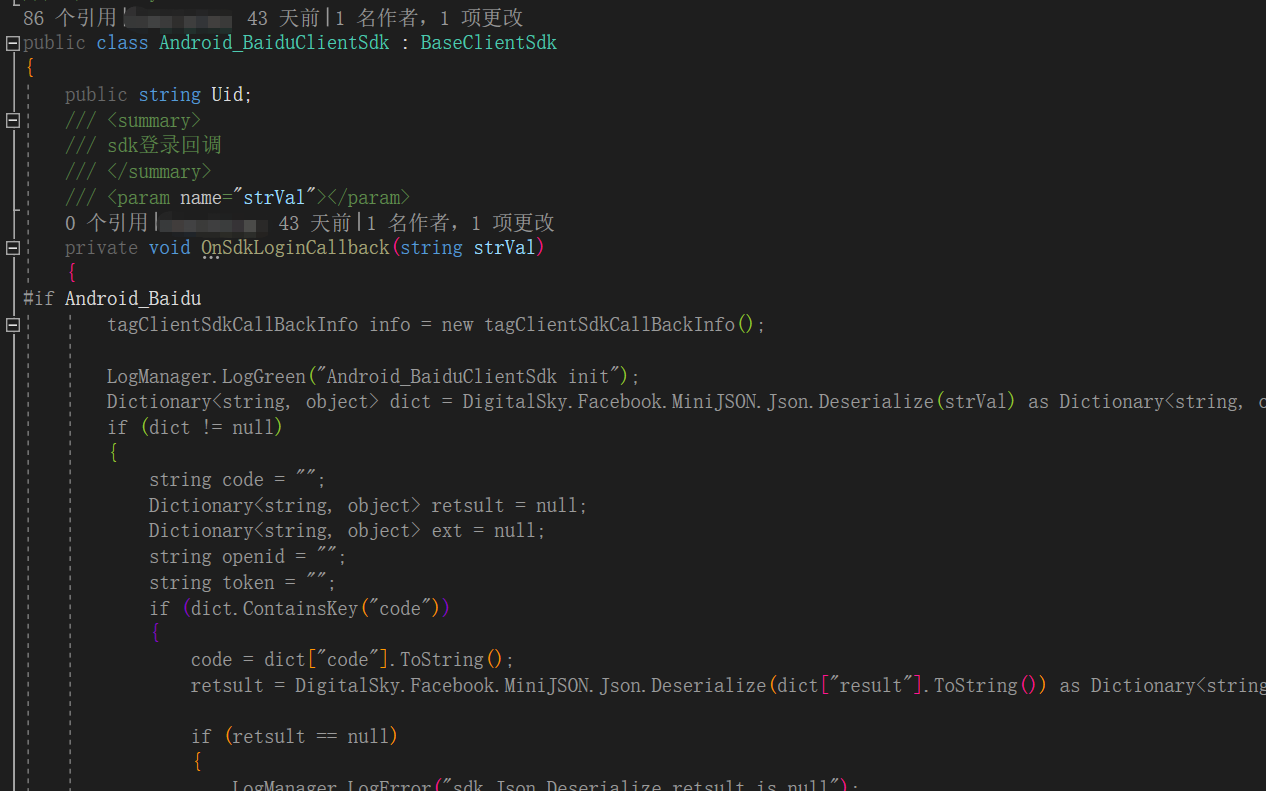


图6,7 宏标签管理平台代码

渠道和服务器的切换，是为了获取不同的配置数据，表的键是，打包方式配置数据表如下:

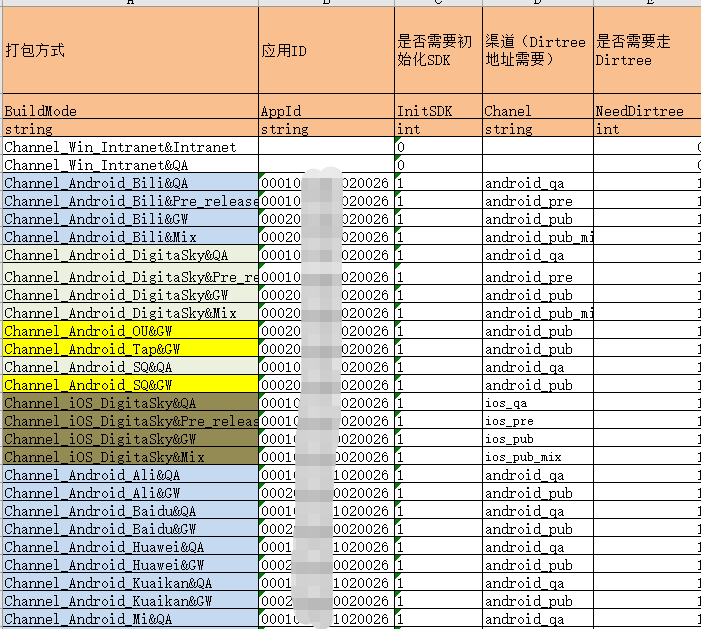


图8 配置表部分截图

可以把使用到的数据尽量都写在这个配置表里，后期维护，如果有问题可以更少地减少修改代码，只需更新配置。这个excel表，我们打表成Lua代码，如下：



图9 配置表Lua部分代码

这样在Lua中可以准确知道当前版本是哪个配置，查错也方便很多，而且可以走热更新。

有了上面的基础，由于渠道众多，所以必须有一键打包这个工具，需要一键打多渠道包，还得解决两个问题。一.由于每个平台SDK中，Plugins文件都不一样，如何切换替换？二.一键打多渠道包过程中，如何更改游戏启动脚本定义的那三个序列化变量值，并更改宏标签呢？对于这两个问题，打每一个渠道包之前，首先把当前Plugins目录下的Android文件夹拷贝一份到Assets同级目录下，其次把平台SDK的UnityPackage包导入进来，在导入完成回调中更新宏标签，以及更新序列化变量值，然后再打包，最后打完包后再把Android文件夹拷贝进来还原之前的目录文件，至此一个渠道包就打好了，多个渠道包依次循环执行一遍就好了。在打包前，还需要做一个可视化的编辑器窗口，需要记录每个渠道打包配置数据，截图如下：

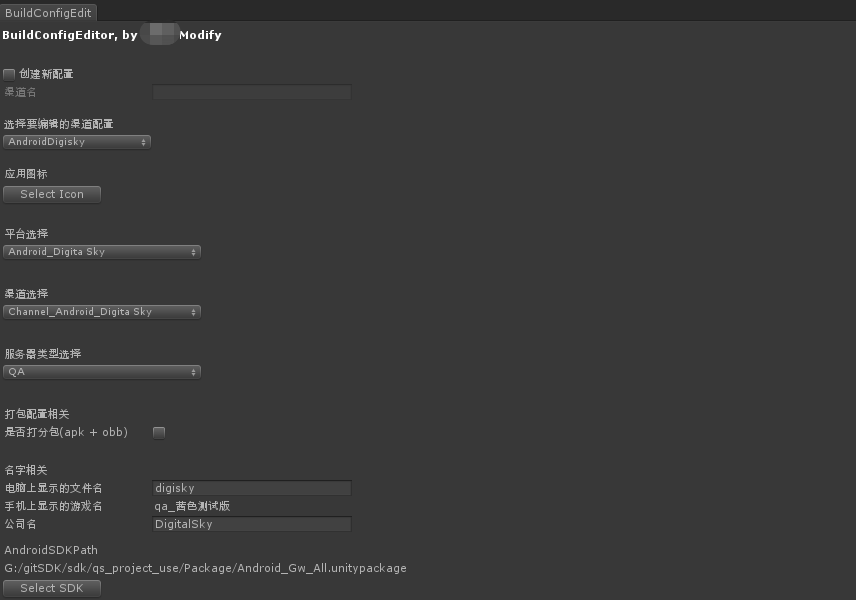


图10 打包配置编辑器部分窗口截图

Build打包窗口如下：



图11 打包编辑器窗口

打包主要难点在于导入UnityPackage包时，必须是在完成回调里面进行打包，而导入UnityPackage包的回调的注册有一个坑，经测试需要注意的是得放在OnEnable里(Unity5.6.6f2有这个问题), 具体实现代码如下:

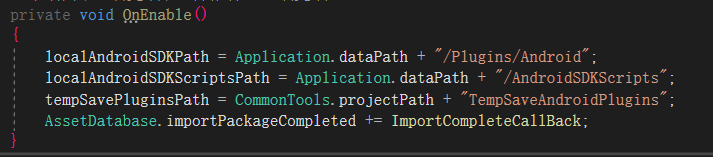


图12 演示代码

在回调里面需要首先处理切换平台宏标签，并且更新序列化值，代码如下：

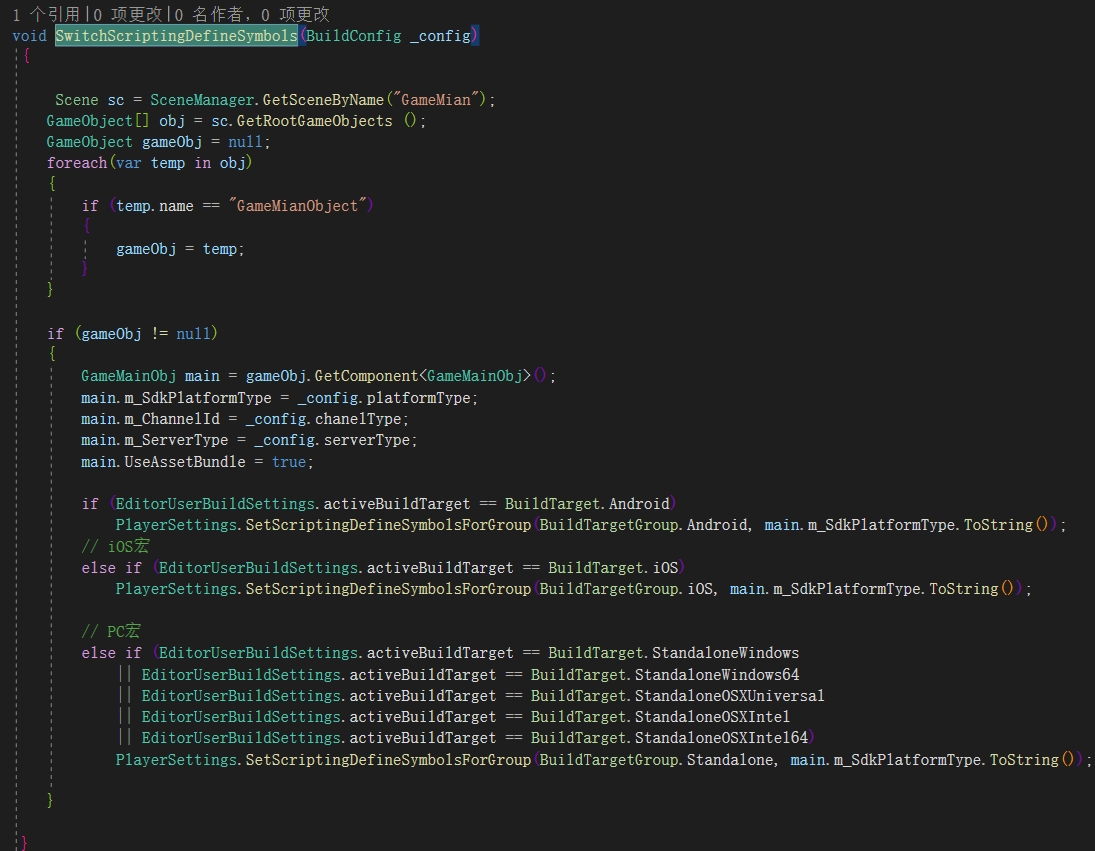


图13 更新序列化值以及宏标签代码

其次，从打包配置表里，读取配置，进行打包设置，代码如下：

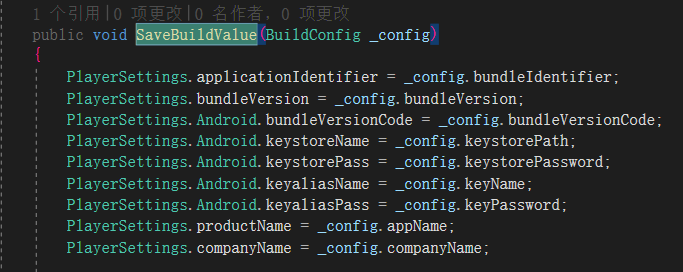


图14 打包设置代码

最后，进行打包，打包完成后再还原所有设置，序列化值以及文件目录，代码如下：

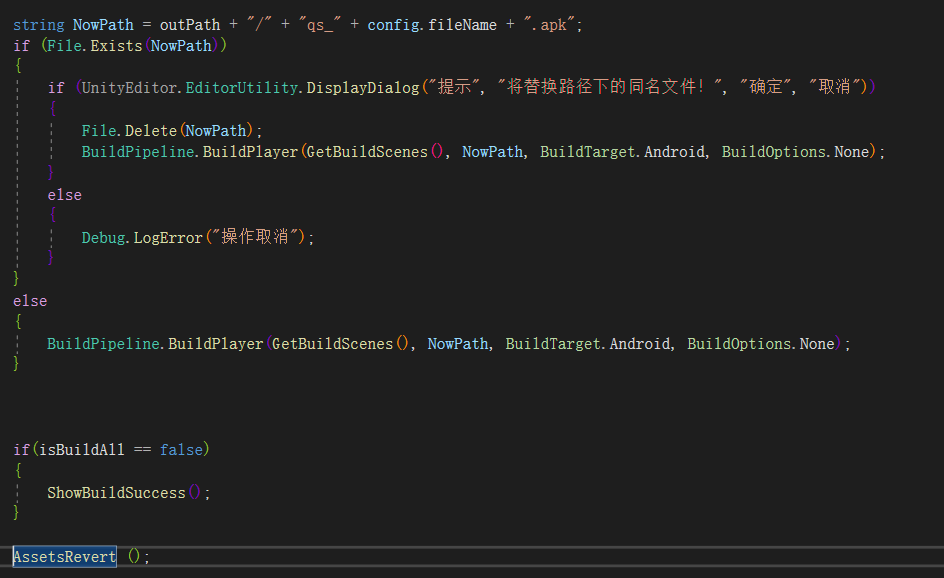


图15 打包代码

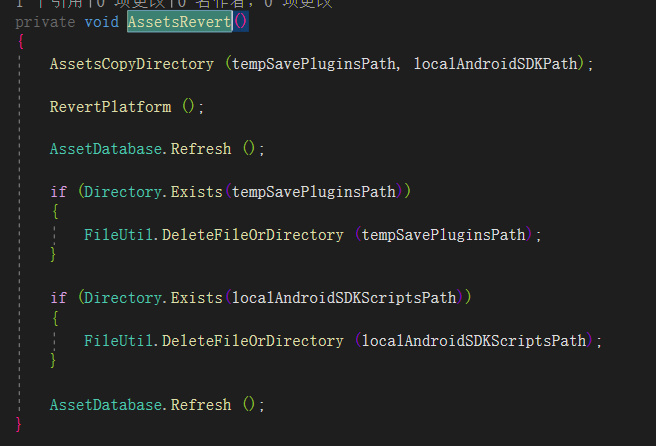


图15 还原代码

后续还可以更好的优化一键打包，用批处理或者python直接自动拉取最新的git仓数据，然后进行命令行打包，这样才是真正意义上的一键打包。

## 三．总结

本次主要分享了版本管理中的多渠道打Apk包，多渠道打包做好了可以节省大量的工作时间，后续会继续分享加解包，热更新流程等等。