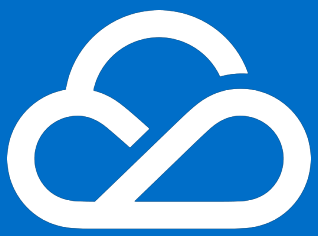
腾讯云游戏更新

SDK接入（C# ）

产品文档



腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

SDK接入（C# ）

C#SDK下载与导入

1 准备工作

1.1 服务配置

开通服务：https://cloud.tencent.com/document/product/595/10557

新建渠道：https://cloud.tencent.com/document/product/595/10558

程序更新：https://cloud.tencent.com/document/product/595/10559

资源更新：https://cloud.tencent.com/document/product/595/10560

1.3 更新业务逻辑

请仔细阅读游戏更新的业务逻辑
https://cloud.tencent.com/document/product/595/10555

2 SDK与demo下载

c# sdk下载

示例demo下载（请勿直接应用demo工程中的sdk库文件到开发者自己的工程，请直接下载正确的sdk使用）

3 开始接入更新

程序和资源更新，安卓、IOS略有差别，请分别参与接入文档：
程序&资源更新-Android接入、程序&资源更新-IOS接入

程序&资源更新-Android

1 导入SDK和初始化

1.1 导入sdk到unity工程

在Unity3D IDE中选择 asset-->import package-->custom package-->select
导入GCloudx.x.x.x.unitypackage

1.2 使用游戏id和游戏key初始化游戏云服务

GCloud.InitializeInfoinitinfo = new GCloud.InitializeInfo(gameid, gamekey);

GCloud.IGCloud.Instance.Initialize(initinfo);

Android平台请详细参考：《Android平台操作指南》

2 Android程序更新

每个渠道有两套更新环境：” 预发布环境 “和” 正式环境 “，两套环境仅更新地址不同。下面以正式环境为例，示例sdk接入

2.1 实现DolphinData接口

```
class TestUpdateData:GCloud.Dolphin.DolphinDateInterface
{
  public string GetUpdateTempPath()
   {
  return "更新过程相关的存储路径，不要删除修改此目录的部分内容，目录必须存在并可写";
  }
  public string GetUpdateSourceSavePath()
```

```
 {
 return "资源存储解压路径，游戏从此目录可读取游戏资源，目录必须存在并可写";
 }
 public string GetUpdateApolloPath()
 {
 return "更新模块相关的信息的存储路径，不要删除修改此目录的内容，目录必须存在并可写";
 }
 public string GetUserDateString()
 {
 return "用户自定义上报内容，可以为空，可通过这个内容在后台查询到相应的上报日志，定位问题";
 }
 public string GetCurrentSourceVersion()
 {
 return "";
//当前资源版本号，形式x.x.x.x，x对应一个数字（short长度），属于更新服务控制台上创建的资源版本号集合，程序更新此处返回”“ 即可;
 }
 public string GetCurrentProgramVersion()
 {
 return
"0.0.0.1";
//当前程序版本号，形式x.x.x.x，x对应一个数字（short长度），属于更新服务控制台上创建的程序版本号集合;
 }
}
```

2.2 实现更新回调

```
public class TestCallBack : GCloud.Dolphin.DolphinCallBackInterface
{
 public void OnNoticeNewVersionInfo(GCloud.Dolphin.NewVersionInfo info)
```

```
{
//获取版本信息，更新流程等待，info中有更新目标版本对应的内容
}

public void OnUpdateProgressInfo(string msg, System.UInt32 nowSize,
System.UInt32 totalSize,bool isDownloading)
{
//更新进度，msg是当前状态对应的提示信息，一次更新会有多种状态，对应
//不同的提示信息，每一种状态都会有0-100%的进度，isDownloading表示当
//前进度是一个下载状态，在使用网路
}

public void OnUpdateMessageBoxInfo(string msg,
GCloud.Dolphin.MessageBoxType msgBoxType,
bool isError,
System.UInt32 errorCode)
{
//更新信息提示，实现一个messagebox显示内容，更新流程等待
}
public void OnNoticeInstallApk(string apkPath)
{
//通知安装apk，收到此回调后游戏可以拉起apk安装
}
public void OnNoticeUpdateSuccess()
{
//更新成功
}
public void OnNoticeChangeSourceVersion(string newVersionStr)
{
//please change source version to newVersionStr
//通知本地资源已更新，游戏需更新本地资源版本号，以便于下次更新传入
}
public void OnNoticeFirstExtractSuccess()
{
```

```
 //通知首包解压完成
 }
}
```

2.3 初始化更新

```
GCloud.Dolphin.DolphinFactory factory = new GCloud.Dolphin.DolphinFactory();
TestCallBack callback = new TestCallBack();
TestUpdateData data = new TestUpdateData();

GCloud.Dolphin.DolphinMgrInterface mgr = factory.CreateDolphinMgr(callback, data);
GCloud.Dolphin.UpdateInitInfo info = new GCloud.Dolphin.UpdateInitInfo();

info.updateInitType = GCloud.Dolphin.UpdateInitType.UpdateInitType_OnlyProgram; //更新类型
info.gameUpdateUrl = "download.1.1075077702.gcloudcs.com"; //正式环境地址或预发布环境地址
info.updateChannelId = 23979; //在更新控制台上渠道分类id对应
info.userId = "用户id";
//用户id，灰度更新使用，例如游戏微信登录的openid或QQ号等,非灰度更新，传入空字符即可
info.worldId = "区服id ";
//用户区服id，灰度更新使用，用户登录到游戏服对应的id，非灰度更新，传入空字符即可
info.grayUpdate = false; //是否灰度更新

mgr.InitUpdateMgr(info, false);
mgr.StartUpdateService(); //启动更新
```

2.4 更新流程控制

在游戏帧循环中定期调用

mgr.DriveUpdateService();

2.5 回调处理

收到回调
OnNoticeNewVersionInfo()和OnUpdateMessageBoxInfo()回调之后更新流程等待；有下面两条流程：
1． 调用Continue()继续更新
2． 调用StopUpdateService()退出更新。

2.6 更新过程中停止流程

mgr.StopUpdateService();
调用StopUpdateService
之后若再次重启更新，需要重新创建更新对象（包括GCloud.Dolphin.DolphinFactory
对象），重新开始；因为更新的特殊性，sdk
不提供暂停重启功能；内部下载会断点续传，无重复下载带来的流量消耗

2.7 Android程序APK拉起安装说明

可选择使用我们的发布包GCloud.jar中的方法：com.tencent.gcloud.dolphin.CuIIPSMobile.installAPK方法，或者自行实现apk的安装

示例：

```
public bool InstallApk(string path)
{
AndroidJavaClass jc = new AndroidJavaClass ("com.unity3d.player.UnityPlayer");
if(jc == null)
{
return false;
}
AndroidJavaObject m_jo = jc.GetStatic<AndroidJavaObject> ("currentActivity");
if(m_jo == null)
{
return false;
}
AndroidJavaObject jo = new AndroidJavaObject("com.tencent.gcloud.dolphin.CuIIPSMobile");
if(jo == null)
{
return false;
}
int result = jo.Call<int>("installAPK",path,m_jo);
if (result != 0)
{
return false;
}
else
{
return true;
}
}
```

3 资源更新

游戏资源指游戏配置文件，美术图片，脚本，unity的assetbundle文件等资源，不需要重新安装游戏以及不需要苹果审核，下载后游戏直接使用的文件。通过资源更新功能，可极大的提高游戏的可运营性。

特别说明
每次打包的资源必须是游戏对应的全量资源，在更新的过程中，游戏更新dolphin的sdk会自动和玩家本地的资源对比，下载差异资源； 以打包到zip文件中的单个文件为单位做差异，所以单个资源文件不应过大，避免大文件少量数据区修改导致大文件下载；
资源更新只做文件替换和增加，不做文件删除，游戏应避免和删除相关的游戏逻辑。

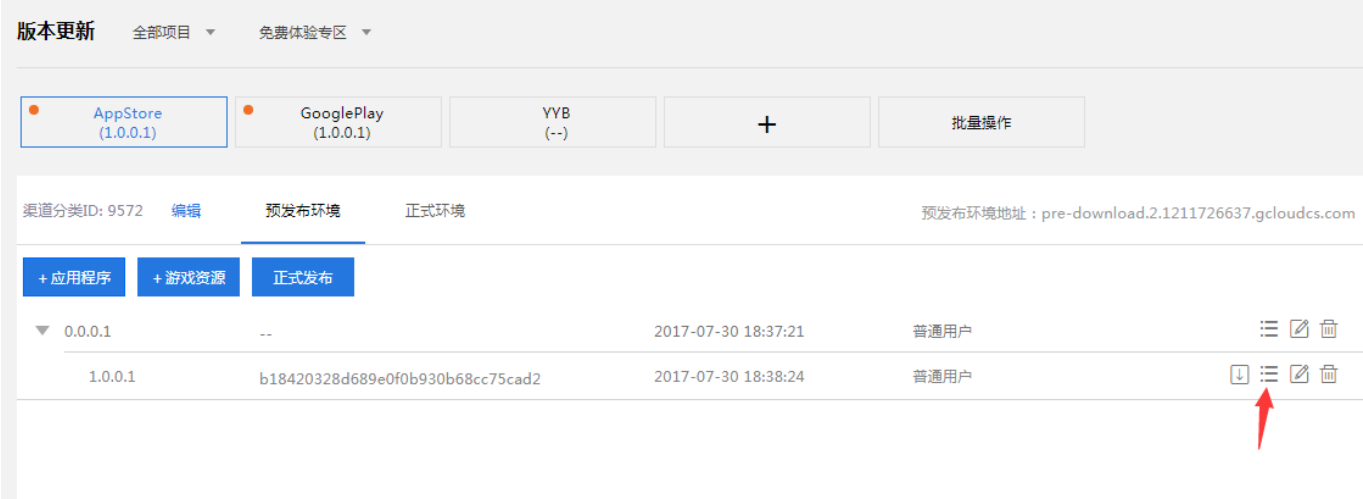
3.1 首包解压

首包解压是将打包在应用安装包中的资源解压到一个应用的磁盘目录，这样才能对这个目录中的资源进行资源更新，本更新方案的资源更新也是针对这个磁盘目录的，应用都从这个磁盘目录读取资源。

示例：资源版本1.0.0.1就是程序版本0.0.0.1的首包资源，向市场发布0.0.0.1的程序包时，玩家从市场下载安装0.0.0.1的程序包之后，第一次运行就需要做首包解压，将1.0.0.1的资源解压到磁盘上。

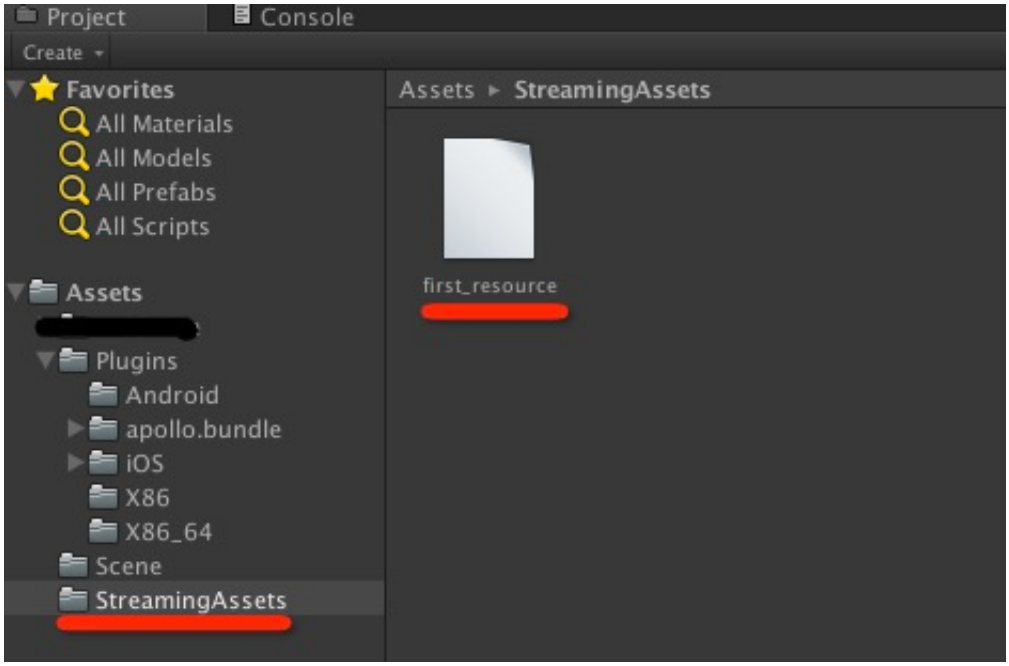
3.1.1 第一步：下载首包

第一步：下载已经创建好的1.0.0.1版本的资源包到本地。如图，下载红箭头处url对应的ifs文件。



3.1.2 第二步：ifs重命名

第二步：将下载的对应ifs文件重命名成first_source.png，存放到unity游戏工程的StreamingAssets目录。



3.1.3 第三步：开启首包解压

第三步：游戏运行后游戏判断需要首包解压时，在上文程序更新流程中开启首包解压功能（也可以在下文资源更新流程），开启首包解压后，会先做首包解压，然后执行更新流程。

...

mgr.InitUpdateMgr(info, true);//第二个参数，是否开启首包解压

...

说明：首包解压操作执行完成会回调OnNoticeFirstExtractSuccess()

3.2 资源更新客户端sdk接入

示例：程序版本0.0.0.1上创建了1.0.0.1 和 1.0.0.2资源版本，下面示例，从1.0.0.1更新到1.0.0.2

3.2.1 实现DolphinData接口

```
class TestUpdateData:GCloud.Dolphin.DolphinDateInterface
{
public string GetUpdateTempPath()
{
return "更新过程相关的存储路径，不要删除修改此目录的部分内容，目录必须存在并可写";
}
public string GetUpdateSourceSavePath()
{
return "资源存储解压路径，游戏从此目录可读取游戏资源，目录必须存在并可写";
}
public string GetUpdateApolloPath()
{
return "更新模块相关的信息的存储路径，要删除修改此目录的内容，目录必须存在并可写";
}
public string GetUserDateString()
{
return "用户自定义上报内容，可以为空，可通过这个内容在后台查询到相应的上报日志，定位问题";
}
public string GetCurrentSourceVersion()
{
return "1.0.0.1"; //当前资源版本号，示例当前的资源版本应该是1.0.0.1;
}
```

```
public string GetCurrentProgramVersion()
{
return "0.0.0.1";//当前程序版本号;
}
}
```

3.2.2 实现更新回调

```
public class TestCallBack : GCloud.Dolphin.DolphinCallBackInterface
{
public void OnNoticeNewVersionInfo(GCloud.Dolphin.NewVersionInfo info)
{
//获取版本信息，更新流程等待，info中有更新目标版本对应的内容，示例这里会提示有1.0.0.2的资源更新
}
public void OnUpdateProgressInfo(string msg,
 System.UInt32 nowSize,
 System.UInt32 totalSize,bool isDownloading)
{
//更新进度，msg是当前状态对应的提示信息，一次更新会有多种状态，对应
//不同的提示信息，每一种状态都会有0-100%的进度，isDownloading表示当
//前进度是一个下载状态，在使用网路
}
public void OnUpdateMessageBoxInfo(string msg,
  GCloud.Dolphin.MessageBoxType msgBoxType,
  bool isError,
  System.UInt32 errorCode)
{
//更新信息提示，实现一个messagebox显示内容，更新流程等待
}
public void OnNoticeInstallApk(string apkPath)
{
```

```
//通知安装apk，收到此回调后游戏可以拉起apk安装
}
public void OnNoticeUpdateSuccess()
{
//更新成功
}
public void OnNoticeChangeSourceVersion(string newVersionStr)
{
//please change source version to newVersionStr
//通知本地资源已更新，游戏需更新本地资源版本号，以便于下次更新传入
//这里记录当前资源的版本号newVersionStr（示例newVersionStr应是1.0.0.2）
}
public void OnNoticeFirstExtractSuccess()
{
//通知首包解压完成
}
}
```

3.2.3 初始化更新

```
GCloud.Dolphin.DolphinFactory factory = new GCloud.Dolphin.DolphinFactory();
TestCallBack callback = new TestCallBack();
TestUpdateData data = new TestUpdateData();

GCloud.Dolphin.DolphinMgrInterface mgr = factory.CreateDolphinMgr(callback, data);
GCloud.Dolphin.UpdateInitInfo info = new GCloud.Dolphin.UpdateInitInfo();

info.updateInitType = GCloud.Dolphin.UpdateInitType.UpdateInitType_OnlySource; //资源更新类型

info.gameUpdateUrl = "download.1.1075077702.gcloudcs.com";
info.updateChannelId = 23979;
```

```
info.userId = "用户id";
info.worldId = "区服id ";
info.grayUpdate = false;

mgr.InitUpdateMgr(info, false);
mgr.StartUpdateService();
```

3.2.4 更新流程控制

在游戏帧循环中定期调用

```
mgr.DriveUpdateService();
```

3.2.5 回调处理

收到回调

OnNoticeNewVersionInfo()和OnUpdateMessageBoxInfo()回调之后更新流程等待；有下面两条流程：

1． 调用Continue()继续更新

2． 调用StopUpdateService()退出更新。

3.2.6 更新过程中停止流程

```
mgr.StopUpdateService();
```

调用StopUpdateService

之后若再次重启更新，需要重新创建更新对象（包括GCloud.Dolphin.DolphinFactory对象），重新开始；

因为更新的特殊性，sdk不提供暂停重启功能；内部下载会断点续传，无重复下载带来的流量消耗

3.2.7 更新判断

收到回调OnNoticeNewVersionInfo之后，服务器返回的更新信息GCloud.Dolphin.NewVersionInfo对象中

info.isCurrentNewest -> 当前版本是否是最新的，是否有新版本可以更新
info.isForce -> 是否强制更新
true：强制更新，则必须更新之后才能进入游戏,对应客户端当前的版本已经在gcloud控制台上变成了不可用
false：推荐更新，表示当前版本在gcloud上仍然可用，但是不是最新可用版本
info.needDownloadSize -> 更新需要的下载大小，单位byte
info.versionStr -> 新版本版本号
info.userDefineStr -> info.versionStr对应版本配置的用户自定义和版本描述

3.2.8 更新判断流程

isCurrentNewest（是否有新版本） -> isForce(有新版本，是否强制)

4 android安装7.0兼容说明（使用installAPK方法）

Android7.0强制启用了被称作
StrictMode的策略，带来的影响就是你的App对外无法暴露file://类型的URI了。 如果你使用Intent携带这样的URI去打开外部App(比如：通过url安装apk)，那么会抛出FileUriExposedException异常。

接入时需要通过下面两个步骤：

manifest修改 --- AndroidManifest.xml添加FileProvider
path文件添加 --- res/xml目录下添加apollo_file_paths.xml
第一步：manifest修改 AndroidManifest.xml添加FileProvider，如下:

```
<!-- 7.0 fileShare for targeSdkVersion>=24 注意:
1. authorities这里格式为应用包名packageName+".ApolloFileprovider"
2. resource属性：这里需要定义apollo_file_paths.xml文件放到工程res/xml下面-->
<provider
android:name="android.support.v4.content.FileProvider"
android:authorities="包名.ApolloFileprovider"
android:exported="false"
android:grantUriPermissions="true" >
<meta-data
android:name="android.support.FILE_PROVIDER_PATHS"
android:resource="@xml/apollo_file_paths" />
</provider>
```

第二步：path文佳添加 添加apollo_file_paths.xml文件到res/xml/目录下，文件内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<paths>
<external-path path="." name="external_storage_root" />
<files-path path="." name="file_patch_root" />
<cache-path path="." name="cache_patch_root" />
</paths>
```

这两个步骤缺一不可，不然会导致应用在7.0手机安装apk的时候crash。

程序&资源更新-IOS

1 导入SDK和初始化

1.1 导入sdk到unity工程

在Unity3D IDE中选择 asset-->import package-->custom package-->select
导入GCloudx.x.x.x.unitypackage

1.2 使用游戏id和游戏key初始化游戏云服务

GCloud.InitializeInfoinitinfo = new GCloud.InitializeInfo(gameid, gamekey);

GCloud.IGCloud.Instance.Initialize(initinfo);

2 IOS程序更新sdk接入

IOS不能进行程序更新，有新版本可以提示跳转至app store进行更新。

2.1 实现DolphinData接口

```
class TestUpdateData:GCloud.Dolphin.DolphinDateInterface
{
  public string GetUpdateTempPath()
   {
  return "更新过程相关的存储路径，不要删除修改此目录的部分内容，目录必须存在并可写";
  }
  public string GetUpdateSourceSavePath()
   {
  return "资源存储解压路径，游戏从此目录可读取游戏资源，目录必须存在并可写";
```

```
 }
 public string GetUpdateApolloPath()
  {
 return "更新模块相关的信息的存储路径，不要删除修改此目录的内容，目录必须存在并可写";
 }
 public string GetUserDateString()
  {
 return "用户自定义上报内容，可以为空，可通过这个内容在后台查询到相应的上报日志，定位问题";
 }
 public string GetCurrentSourceVersion()
  {
 return "";
//当前资源版本号，形式x.x.x.x，x对应一个数字（short长度），属于更新服务控制台上创建的资源版本号集合，程序更新此处返回”“ 即可;
 }
 public string GetCurrentProgramVersion()
  {
 return
"0.0.0.1";
//当前程序版本号，形式x.x.x.x，x对应一个数字（short长度），属于更新服务控制台上创建的程序版本号集合;
 }
}
```

2.2 实现更新回调

```
public class TestCallBack : GCloud.Dolphin.DolphinCallBackInterface
{
 public void OnNoticeNewVersionInfo(GCloud.Dolphin.NewVersionInfo info)
  {
 //获取版本信息，更新流程等待，info中有更新目标版本对应的内容
```

```
}

public void OnUpdateProgressInfo(string msg, System.UInt32 nowSize,
System.UInt32 totalSize,bool isDownloading)
 {
//更新进度，msg是当前状态对应的提示信息，一次更新会有多种状态，对应
//不同的提示信息，每一种状态都会有0-100%的进度，isDownloading表示当
//前进度是一个下载状态，在使用网路
}

public void OnUpdateMessageBoxInfo(string msg,
GCloud.Dolphin.MessageBoxType msgBoxType,
bool isError,
System.UInt32 errorCode)
 {
//更新信息提示，实现一个messagebox显示内容，更新流程等待
}
public void OnNoticeInstallApk(string apkPath)
 {
//通知安装apk，收到此回调后游戏可以拉起apk安装
}
public void OnNoticeUpdateSuccess()
 {
//更新成功
}
public void OnNoticeChangeSourceVersion(string newVersionStr)
 {
//please change source version to newVersionStr
//通知本地资源已更新，游戏需更新本地资源版本号，以便于下次更新传入
}
public void OnNoticeFirstExtractSuccess()
 {
//通知首包解压完成
}
```

```
}
```

2.3 初始化更新

```
GCloud.Dolphin.DolphinFactory factory = new GCloud.Dolphin.DolphinFactory();
TestCallBack callback = new TestCallBack();
TestUpdateData data = new TestUpdateData();

GCloud.Dolphin.DolphinMgrInterface mgr = factory.CreateDolphinMgr(callback, data);
GCloud.Dolphin.UpdateInitInfo info = new GCloud.Dolphin.UpdateInitInfo();

info.updateInitType = GCloud.Dolphin.UpdateInitType.UpdateInitType_OnlyProgram; //更新类型
info.gameUpdateUrl = "download.1.1075077702.gcloudcs.com"; //正式环境地址或预发布环境地址
info.updateChannelId = 23979; //在更新控制台上渠道分类id对应
info.userId = "用户id";
//用户id，灰度更新使用，例如游戏微信登录的openid或QQ号等,非灰度更新，传入空字符即可
info.worldId = "区服id ";
//用户区服id，灰度更新使用，用户登录到游戏服对应的id，非灰度更新，传入空字符即可
info.grayUpdate = false; //是否灰度更新

mgr.InitUpdateMgr(info, false);
mgr.StartUpdateService(); //启动更新
```

2.4 更新流程控制

在游戏帧循环中定期调用

mgr.DriveUpdateService();

2.5 回调处理

收到回调OnNoticeNewVersionInfo之后判断是否需要更新，调用mgr.StopUpdateService()停止更新，若需要更新跳转到appstore下载，若不需要更新可进入游戏后续流程。

```
public class TestCallBack : GCloud.Dolphin.DolphinCallBackInterface
{
public void OnNoticeNewVersionInfo(GCloud.Dolphin.NewVersionInfo info)
{
//jump to appstore
}
...
}
```

2.6 更新过程中停止流程

mgr.StopUpdateService();
调用StopUpdateService
之后若再次重启更新，需要重新创建更新对象（包括GCloud.Dolphin.DolphinFactory
对象），重新开始；因为更新的特殊性，sdk
不提供暂停重启功能；内部下载会断点续传，无重复下载带来的流量消耗

3 资源更新

游戏资源指游戏配置文件，美术图片，脚本，unity的assetbundle文件等资源，不需要重新安装游戏以及不需要苹果审核，下载后游戏直接使用的文件。通过资源更新功能，可极大的提高游戏的可运营性。

特别说明
每次打包的资源必须是游戏对应的全量资源，在更新的过程中，游戏更新dolphin的sdk会自动和玩家本地的资源对比，下载差异资源； 以打包到zip文件中的单个文件为单位做差异，所以单个资源文件不应过大，避免大文件少量数据区修改导致大文件下载；
资源更新只做文件替换和增加，不做文件删除，游戏应避免和删除相关的游戏逻辑。

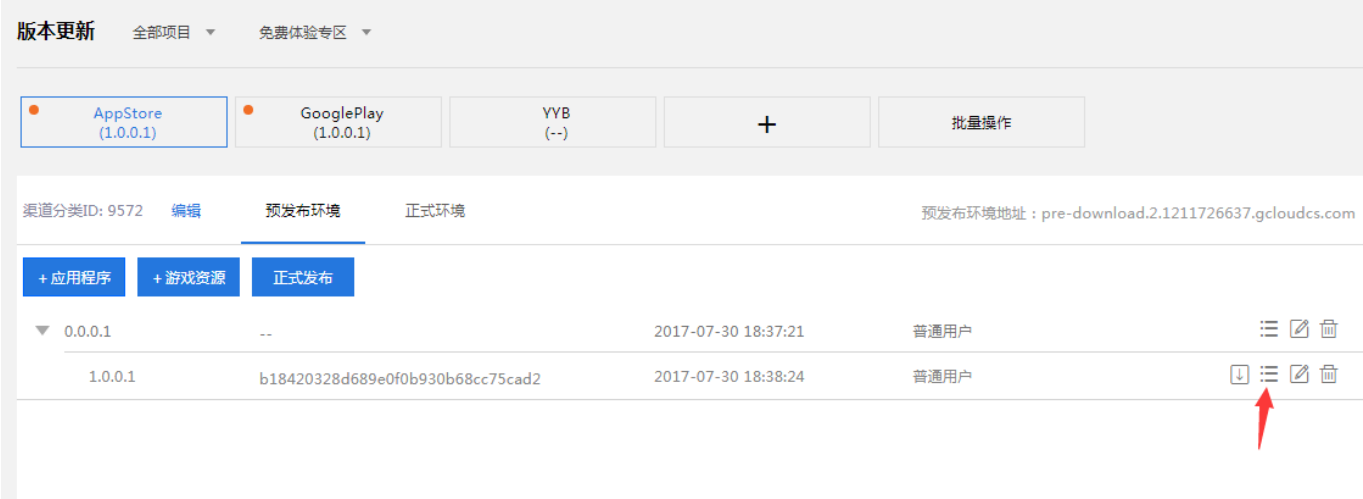
3.1 首包解压

首包解压是将打包在应用安装包中的资源解压到一个应用的磁盘目录，这样才能对这个目录中的资源进行资源更新，本更新方案的资源更新也是针对这个磁盘目录的，应用都从这个磁盘目录读取资源。

示例：资源版本1.0.0.1就是程序版本0.0.0.1的首包资源，向市场发布0.0.0.1的程序包时，玩家从市场下载安装0.0.0.1的程序包之后，第一次运行就需要做首包解压，将1.0.0.1的资源解压到磁盘上。

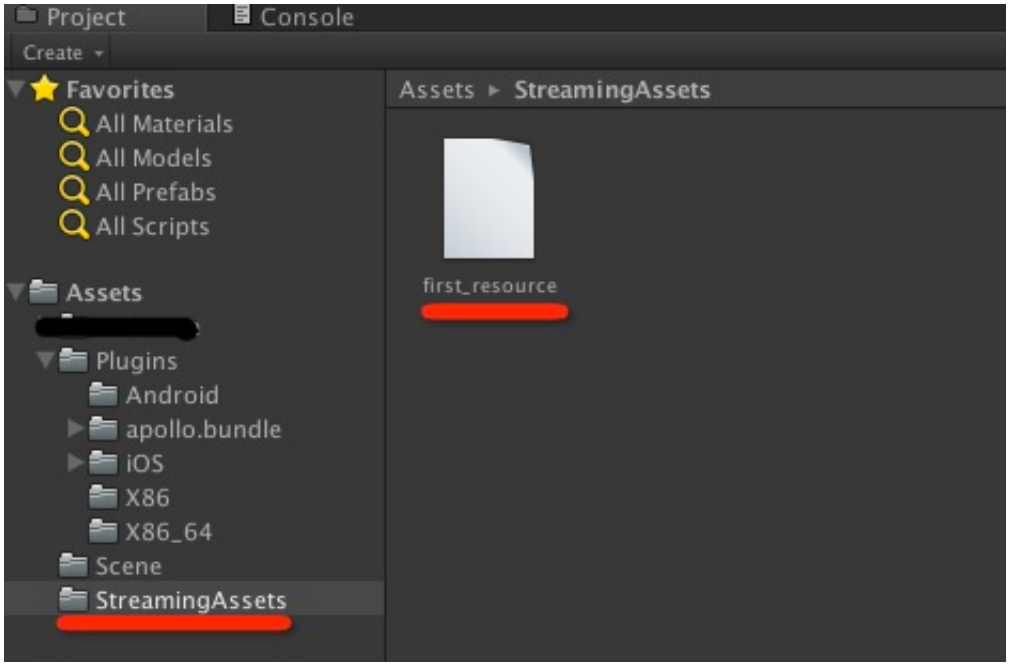
3.1.1 第一步：下载首包

第一步：下载已经创建好的1.0.0.1版本的资源包到本地。如图，下载红框中url对应的ifs文件。



3.1.2 第二步：ifs重命名

第二步：将下载的对应ifs文件重命名成first_source.png，存放到unity游戏工程的StreamingAssets目录。



3.1.3 第三步：开启首包解压

第三步：游戏运行后游戏判断需要首包解压时，在上文程序更新流程中开启首包解压功能（也可以在下文资源更新流程），开启首包解压后，会先做首包解压，然后执行更新流程。

...

mgr.InitUpdateMgr(info, true);//第二个参数，是否开启首包解压

...

说明：首包解压操作执行完成会回调OnNoticeFirstExtractSuccess()

3.2 资源更新客户端sdk接入

示例：程序版本0.0.0.1上创建了1.0.0.1 和 1.0.0.2资源版本，下面示例，从1.0.0.1更新到1.0.0.2

3.2.1 实现DolphinData接口

```
class TestUpdateData:GCloud.Dolphin.DolphinDateInterface
{
public string GetUpdateTempPath()
{
return "更新过程相关的存储路径，不要删除修改此目录的部分内容，目录必须存在并可写";
}
public string GetUpdateSourceSavePath()
{
return "资源存储解压路径，游戏从此目录可读取游戏资源，目录必须存在并可写";
}
public string GetUpdateApolloPath()
{
return "更新模块相关的信息的存储路径，要删除修改此目录的内容，目录必须存在并可写";
}
public string GetUserDateString()
{
return "用户自定义上报内容，可以为空，可通过这个内容在后台查询到相应的上报日志，定位问题";
}
public string GetCurrentSourceVersion()
{
return "1.0.0.1"; //当前资源版本号，示例当前的资源版本应该是1.0.0.1;
}
```

```
public string GetCurrentProgramVersion()
{
return "0.0.0.1";//当前程序版本号;
}
}
```

3.2.2 实现更新回调

```
public class TestCallBack : GCloud.Dolphin.DolphinCallBackInterface
{
public void OnNoticeNewVersionInfo(GCloud.Dolphin.NewVersionInfo info)
{
//获取版本信息，更新流程等待，info中有更新目标版本对应的内容，示例这里会提示有1.0.0.2的资源更新
}
public void OnUpdateProgressInfo(string msg,
 System.UInt32 nowSize,
 System.UInt32 totalSize,bool isDownloading)
{
//更新进度，msg是当前状态对应的提示信息，一次更新会有多种状态，对应
//不同的提示信息，每一种状态都会有0-100%的进度，isDownloading表示当
//前进度是一个下载状态，在使用网路
}
public void OnUpdateMessageBoxInfo(string msg,
  GCloud.Dolphin.MessageBoxType msgBoxType,
  bool isError,
  System.UInt32 errorCode)
{
//更新信息提示，实现一个messagebox显示内容，更新流程等待
}
public void OnNoticeInstallApk(string apkPath)
{
```

```
//通知安装apk，收到此回调后游戏可以拉起apk安装
}
public void OnNoticeUpdateSuccess()
{
//更新成功
}
public void OnNoticeChangeSourceVersion(string newVersionStr)
{
//please change source version to newVersionStr
//通知本地资源已更新，游戏需更新本地资源版本号，以便于下次更新传入
//这里记录当前资源的版本号newVersionStr（示例newVersionStr应是1.0.0.2）
}
public void OnNoticeFirstExtractSuccess()
{
//通知首包解压完成
}
}
```

3.2.3 初始化更新

```
GCloud.Dolphin.DolphinFactory factory = new GCloud.Dolphin.DolphinFactory();
TestCallBack callback = new TestCallBack();
TestUpdateData data = new TestUpdateData();

GCloud.Dolphin.DolphinMgrInterface mgr = factory.CreateDolphinMgr(callback, data);
GCloud.Dolphin.UpdateInitInfo info = new GCloud.Dolphin.UpdateInitInfo();

info.updateInitType = GCloud.Dolphin.UpdateInitType.UpdateInitType_OnlySource; //资源更新类型

info.gameUpdateUrl = "download.1.1075077702.gcloudcs.com";
info.updateChannelId = 23979;
```

```
info.userId = "用户id";
info.worldId = "区服id ";
info.grayUpdate = false;

mgr.InitUpdateMgr(info, false);
mgr.StartUpdateService();
```

3.2.4 更新流程控制

在游戏帧循环中定期调用

```
mgr.DriveUpdateService();
```

3.2.5 回调处理

收到回调

OnNoticeNewVersionInfo()和OnUpdateMessageBoxInfo()回调之后更新流程等待；有下面两条流程：

1． 调用Continue()继续更新

2． 调用StopUpdateService()退出更新。

3.2.6 更新过程中停止流程

```
mgr.StopUpdateService();
```

调用StopUpdateService

之后若再次重启更新，需要重新创建更新对象（包括GCloud.Dolphin.DolphinFactory对象），重新开始；

因为更新的特殊性，sdk不提供暂停重启功能；内部下载会断点续传，无重复下载带来的流量消耗

3.2.7 更新判断

收到回调OnNoticeNewVersionInfo之后，服务器返回的更新信息GCloud.Dolphin.NewVersionInfo对象中

info.isCurrentNewest -> 当前版本是否是最新的，是否有新版本可以更新
info.isForce -> 是否强制更新
true：强制更新，则必须更新之后才能进入游戏,对应客户端当前的版本已经在gcloud控制台上变成了不可用
false：推荐更新，表示当前版本在gcloud上仍然可用，但是不是最新可用版本
info.needDownloadSize -> 更新需要的下载大小，单位byte
info.versionStr -> 新版本版本号
info.userDefineStr -> info.versionStr对应版本配置的用户自定义和版本描述

3.2.8 更新判断流程

isCurrentNewest（是否有新版本） -> isForce(有新版本，是否强制)

unityAPI手册

1 更新回调接口

用户需继承实现

Namespace：GCloud.Dolphin
Interface：DolphinCallBackInterface

(1) void OnNoticeNewVersionInfo(NewVersionInfo newVersionInfo);
回调时机：查询版本服务后返回
参数：NewVersionInfo newVersionInfo
newVersionInfo. versionStr最新版本号
newVersionInfo. needDownloadSize更新需要下载的大小
newVersionInfo. isForce是否强制更新
newVersionInfo. updateType更新类型（资源、程序）
newVersionInfo. userDefineStr用户自定义信息，json格式，对应改版本下用户配置的版本描述和

自定义字符串
newVersionInfo.
isCurrentNewest当前版本是否是最新的，如果是最新的表示不需要更新，否则看isForce，判断是强

制还是可选

(2) void OnUpdateProgressInfo(string msg, System.UInt32 nowSize, System.UInt32 totalSize, bool isDownloading);
回调时机：进度定期回调
参数：msg提示信息
Nowsize当前进度
Totalsize总进度
isDownloading 当前是否是下载，如果是size的单位是B；不是size仅仅为数值，没有单位

(3) void OnUpdateMessageBoxInfo(string msg, MessageBoxType msgBoxType, bool isError, System.UInt32 errorCode);

回调时机：更新服务需要提示用户的时候，如果内部错误
参数：msg提示信息
msgBoxType提示框类型
isError是否是错误提示
errorCode错误码，定位问题使用

(4) void OnNoticeInstallApk(string apkPath);
回调时机：当需要更新apk的时候，apk下载完成通知，可等价于更新成功，用户收到此回调后退出更新模块安装游戏
参数：apkPathapk所在路径

(5) void OnNoticeUpdateSuccess();
回调时机：更新成功

(6) void OnNoticeChangeSourceVersion(string newVersionStr);
回调时机：资源更新成功，通知修改当前资源版本号
参数：newVersionStr更新完成新的资源版本号

(7) void OnNoticeFirstExtractSuccess();
回调时机：初始化需要首包解压，首包解压完成时

2 更新数据接口

用户需继承实现

Namespace：GCloud.Dolphin
Interface：DolphinCallBackInterface

(1) string GetUpdateTempPath();
返回一个存在并可写目录，当资源被清空的时候，需要清空此目录，此目录中有一个flist文件记录本地的资源

列表

(2) string GetUpdateSourceSavePath();
返回一个存在并可写的目录，资源读取目录，游戏更新的资源全在此目录，资源更新依赖此目录更新

(3) string GetUserDateString();
返回用户自定义信息，可返回用户的机器信息等

(4) string GetCurrentSourceVersion();
返回当前的资源版本号

(5) string GetCurrentProgramVersion();
返回当前的程序版本号

(6) string GetUpdateApolloPath();
返回一个存在并可写的目录，不要删除修改此目录中的内容

3 更新模块创建接口

Namespace：GCloud.Dolphin
Interface：DolphinFactory

CreateDolphinMgr(DolphinCallBackInterface callBackImp, DolphinDateInterface dateMgr)
功能：创建更新对象，更新对象不能重复使用
参数：callBackImp 更新回调对象
dateMgr更新数据对象
返回：更新对象

4 更新功能接口

Namespace：GCloud.Dolphin

Interface：DolphinFactory

(1) bool InitUpdateMgr(UpdateInitInfo updateInfo, bool needFirstExtract);

功能：初始化更新对象

参数：updateInfo初始化参数

needFirstExtract是否需要首包解压

返回：执行是否成功

(2) bool UninitUpdateMgr();

功能：反初始化更新对象

返回：是否执行成功

(3) bool StartUpdateService();

功能：启动更新流程

返回：是否执行成功

(4) bool Continue();

功能：继续更新流程，在回调了OnNoticeNewVersionInfo和OnUpdateMessageBoxInfo之后，调用继续更新或重试更新

返回：是否执行成功

(5) bool StopUpdateService();

功能：停止更新流程

返回：是否执行成功

(6) void DriveUpdateService();

功能：驱动更新，定期调用，此函数中执行回调

(7) System.UInt32 GetCurrentDownSpeed();

功能：获取当前的下载速度

(8) void SetNeedFirstExtract(bool needFirstExtract);

功能：设置是否需要首包解压