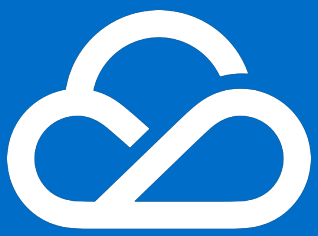
腾讯云游戏更新

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++）

程序&资源更新

1 准备工作

在android平台和ios平台，程序更新和资源更新的整体流程都一致，只有android有少量特殊配置，android的特殊配置在情况中标明。

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

1.2 更新说明

- 类型支持：我们支持程序apk更新(android)，ios版本更新检查，资源更新（android、ios）。
  一次初始化更新，只能做一种类型的更新，对象尽量不重用。
  重试，删除之前的更新对象，重新创建。

- 资源更新
  ：在页面上配置时，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。

- 更新逻辑：请仔细阅读更新逻辑
  https://cloud.tencent.com/document/product/595/10555

2 SDK下载与导入

2.1 下载dolphin服务客户端sdk和demo

c++ sdk下载地址

点击下载示例dem

o（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）

2.2 库文件引入到项目

将发布包Libs下的文件放到游戏项目对应的库目录中并引用

Android: libgcoud.so

IOS: gcloud.framework

Windows: gcloud.dll或者gcloud.lib

Mac: gcloud.bundle

2.3 头文件引用

将发布包Include目录下的头文件放到对应的目录并引用。

3 开始更新

3.1 初始化GCloud服务

未初始化GCloud服务先初始化GCloud服务

```
m_uiGCloudGameId = 1167737824; //替换成您的游戏id
m_strGCloudGameKey = "72558ea6f31512ded57d213a8f3293ac";//替换成您的游戏key

GCloud::InitializeInfo initInfo;
initInfo.GameId = m_uiGCloudGameId;
initInfo.GameKey = m_strGCloudGameKey.c_str();
```

```
IGCloud::GetInstance().Initialize(initInfo);
```

Android平台gcloud初始化有额外操作，参考文档《Android平台操作指南》

3.2 实现更新回调

```
class dolphinCallback :public GCloud::GCloudDolphinCallBack
{
public:
virtual ~dolphinCallback(){}
virtual void OnDolphinVersionInfo(GCloud::dolphinVersionInfo& newVersinInfo)
{
  //获取到新版本，更新流程等待，newVersinInfo中有更新目标版本对应的内容

  //IOS:
  //程序更新不针对ios的程序下载更新，可以通过我们的更新模块检测新版本，
  //通过页面创建版本时，将appstore对应的下载地址填写在版本创建的用户自定义字符串部分；

//发布新版本后，会回调OnDolphinVersionInfo，调用CancelUpdate退出更新，再自行跳转到appstore下载更新。
}
virtual void OnDolphinProgress(GCloud::dolphinUpdateStage curVersionStage, cu_uint64 totalSize, cu_uint64 nowSize)
{
  //更新进度，curVersionStage是当前状态，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
}
virtual void OnDolphinError(GCloud::dolphinUpdateStage curVersionStage, cu_uint32 errorCode)
{
  //更新失败，更新流程结束
```

```
}
virtual void OnDolphinSuccess()
{
  //更新成功，更新流程结束
}
virtual void OnDolphinNoticeInstallApk(char* apkurl)
{
  //apk更新成功，可拉起apk安装，安装完成后游戏根据需求删除或保留apk
  //程序更新不针对ios的ipa下载更新，可以通过我们的更新模块检测新版本，
  //通过页面创建版本时，将appstore对应的下载地址填写在版本创建的用户自定义字符串部分；

//发布新版本后，会回调OnDolphinVersionInfo，调用CancelUpdate退出更新，再自行跳转到appstore下载更新。
}
virtual void OnDolphinFirstExtractSuccess()
{
  //首包解压通知成功
}
};
```

3.3 初始化更新

请项目根据自身情况修改红色部分

```
GCloud::GCloudDolphinInterface * mDolphin = GCloud::CreateDolphin();
GCloud::dolphinInitInfo initInfo;
initInfo.updateType = GCloud::UpdateInitType_OnlySource;//更新类型
initInfo.channelId = channelid;//渠道id,与更新控制台上的渠道分类id对应
initInfo.isGrayUpdate = false;//是否灰度更新
initInfo.openDebugLog = true;//是否开启debug日志
initInfo.openErrorLog = true;//是否开启error日志
```

```
//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，参照下面的版本号说明
_snprintf(initInfo.appVersion, DOLPHIN_VERSIONLENGTH, "%s", "当前程序版本号");
//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，参照下面的版本号说明
_snprintf(initInfo.srcVersion, DOLPHIN_VERSIONLENGTH, "%s", "当前资源版本号");
//更新环境地址，对应渠道页面下的正式环境地址或预发布环境地址
_snprintf(initInfo.updateUrl, DOLPHIN_UPDATEURLLENGTH, "%s", "更新环境地址");

GCloud::dolphinPathInfo pathInfo;

//当前apk路径,当前正在运行apk的绝对路径
_snprintf(pathInfo.curApkPath, DOLPHIN_PATHLENGTH, "%s", "当前apk路径");
//更新信息存储路径，传入时存在并可写，不要删除修改此目录下的内容
_snprintf(pathInfo.dolphinPath, DOLPHIN_PATHLENGTH, "%s", "更新信息存储路径");
//更新路径，资源存放路径，传入时存在并可写，资源从此目录读取
_snprintf(pathInfo.updatePath, DOLPHIN_PATHLENGTH, "%s", "更新路径，资源存放路径");

GCloud::dolphinGrayInfo grayInfo;
//用户id，例如游戏微信登录的openid或QQ号等
_snprintf(grayInfo.userID, DOLPHIN_USERIDLENGTH, "%s", "用户id");
//用户区服id, 用户登录到游戏区服对应的id
_snprintf(grayInfo.worldID, DOLPHIN_WORLDIDLENGTH, "%s", "用户区服id");
GCloud::dolphinFirstExtractInfo feInfo;

//首包解压文件路径,参照第10点说明
_snprintf(feInfo.ifsPath, DOLPHIN_PATHLENGTH, "%s", "首包解压文件路径");
dolphinCallback callback;

//init初始化参数三grayInfo，可以为NULL,在initInfo.isGrayUpdate为true不能是NULL
//init初始化参数四feInfo，为NULL时不做首包解压，不为NULL时先解压再更新
mDolphin->Init(&initInfo, &pathInfo, &grayInfo, & feInfo, &callback);
```

版本号说明： 更新时，需要游戏当前的版本号（资源或程序），初始化时传入。 这里的版本号与更新控制台创建的版本号对应。比如游戏第一个程序版本，在控制台上创建第一个版本，版本号为0.0.0.1，0.0.0.1程序版本运行时初始化更新，这里就要传入0.0.0.1。当在控制台上创建0.0.0.2版本后，0.0.0.1版本就可以获取到0.0.0.2的更新，更新之后就变成了0.0.0.2版本，这时再运行时初始化更新，这里就需要传入0.0.0.2

3.4 启动更新

```
mDolphin->CheckAppUpdate();
```

3.5 更新流程控制

示例：

```
//定期调用，建议在更新阶段插入到游戏的逻辑帧循环中
{
  if (dolphin_success || dolphin_error || dolphin_noticeinstall)
  {
  //在更新成功失败，或通知安装apk时
  break;
  }
  if (dolphin_getversion)
  {
  //获取到新版本需要更新
  dolphin_getversion = false;

  //不更新，ContinueUpdate(false)，并退出循环
  mDolphin->ContinueUpdate(true);//继续更新
  }
  mDolphin->PollCallback(); //定期驱动更新
  Sleep(1);
}
```

3.5 更新过程中取消更新

```
mDolphin->CancelUpdate();
```

3.6 反初始化更新模块

```
mDolphin->Uninit();
GCloud::ReleaseDolphin(&mDolphin);
```

4 其他说明

4.1 资源更新-首包解压说明

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。
首包解压的操作在用户初始化更新模块的时候指定，如果用户指定，会先解压资源之后在连接版本服务器获取更新 打包资源成ifs，命名为x.png，打包到apk或ipa中

初始化首包解压路径如何传？
（1）Android：apk://apkpath?ifsinapkpath

- apkpath：当前运行apk在手机上的绝对路径
- ifsinapkpath：x.png在apk中的路径

示例：x.png在apk的assets目录下 传入路径：apk:///data/app/x.apk?assets/x.png

（2）其他平台：游戏安装之后，在x.png在系统磁盘的绝对路径

4.2 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

4.3 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。

C++ API手册

1 更新回调接口

请继承实现

Namespace GCloud

Class GCloudDolphinCallBack

(1) virtual void OnDolphinVersionInfo(dolphinVersionInfo& newVersinInfo) = 0;

回调时机：查询版本服务后返回

参数：newVersinInfo 版本信息

newVersinInfo. isAppUpdating是否是程序更新

newVersinInfo.isNeedUpdating是否需要更新

newVersinInfo.isForcedUpdating是否强制更新

newVersinInfo.versionNumberOne新版本版本号第一位（左起）

newVersinInfo.versionNumberTwo新版本版本号第二位（左起）

newVersinInfo.versionNumberThree新版本版本号第三位（左起）

newVersinInfo.versionNumberFour新版本版本号第四位（左起）

newVersinInfo.needDownloadSize版本更新需要的下载大小

newVersinInfo.versionDescrition新版本版本描述

newVersinInfo.userDefineStr新版本用户自定义信息

(2) virtual void OnDolphinProgress(dolphinUpdateStage curVersionStage, cu_uint64 totalSize, cu_uint64 nowSize) = 0;

回调时机：进度，定期回调

参数：curVersionStage当前更新状态

dolphinUpdateStage.VS_GetVersionInfo(3)正在获取版本信息

dolphinUpdateStage.VS_FirstExtract(10)首包解压

dolphinUpdateStage.VS_ApkUpdate(70)apk更新

dolphinUpdateStage.VS_ApkUpdateDownConfig(71)apk更新下载配置文件

dolphinUpdateStage.VS_ApkUpdateDownDiffFile(72)apk更新下载差异文件

dolphinUpdateStage.VS_ApkUpdateDownFullApk(73)apk更新下载全量文件

dolphinUpdateStage.VS_ApkUpdateCheckCompletedApk(74) apk更新校验完成文件

dolphinUpdateStage.VS_ApkUpdateCheckLocalApk(75)apk更新校验本地apk
dolphinUpdateStage.VS_ApkUpdateCheckConfig(76)apk更新校验配置文件
dolphinUpdateStage.VS_ApkUpdateCheckDiff(77)apk更新校验差异文件
dolphinUpdateStage.VS_ApkUpdateMergeDiff(78)apk更新合并差异
dolphinUpdateStage.VS_ApkUpdateCheckFull(79)apk更新校验全量文件
dolphinUpdateStage.VS_SourceUpdateCures(90)资源更新
dolphinUpdateStage.VS_SourceUpdateDownloadList(91)资源更新下载配置
dolphinUpdateStage.VS_SourcePrepareUpdate(92)资源更新准备更新
dolphinUpdateStage.VS_SourceAnalyseDiff(93)资源更新分析差异
dolphinUpdateStage.VS_SourceDownload(94)资源更新下载资源
dolphinUpdateStage.VS_SourceExtract(95)资源更新解压资源
totalSize总进度
nowSize当前进度
(3) virtual void OnDolphinError(dolphinUpdateStage curVersionStage, cu_uint32 errorCode) = 0;
回调时机：出错时
参数：
curVersionStage 出错是状态，参照OnDolphinProgress
errorCode 错误码，定位问题

(4) virtual void OnDolphinSuccess() = 0;
回调时机：更新流程执行成功

(5) virtual void OnDolphinNoticeInstallApk(char* apkurl) = 0;
回调时机：当需要apk更新时，更新成功，通知安装apk，游戏收到后退出更新模块拉起apk安装

(6) virtual void OnDolphinFirstExtractSuccess() = 0;
回调时机：当需要首包解压时，解压成功

2 更新对象接口

Namespace GCloud

(1) GCloudDolphinInterface* CreateDolphin();
功能：创建更新对象

(2) void ReleaseDolphin(GCloudDolphinInterface** dolphin);
功能：删除更新对象

3 更新功能接口

Namespace GCloud
Class GCloudDolphinInterface

(1)virtual bool Init(const dolphinInitInfo* initInfo, const dolphinPathInfo* pathInfo,const dolphinGrayInfo* grayInfo,
const dolphinFirstExtractInfo* feInfo,
GCloudDolphinCallBack* callBack) = 0;
功能：初始化更新对象
参数：
initInfo 初始化信息
initInfo. channelId;渠道号
initInfo. updateType;更新类型
initInfo. isGrayUpdate;是否是灰度更新
initInfo. openDebugLog;是否打开debug日志
initInfo. openErrorLog;是否打开error日志
initInfo. updateUrl;更新地址
initInfo. appVersion;当前程序版本
initInfo. srcVersion当前资源版本
pathInfo更新路径信息
pathInfo.updatePath;更新路径，可写并存在
pathInfo.dolphinPath;更新模块信息路径，可写并存在

pathInfo.curApkPath当前apk的路径，apk更新使用

grayInfo灰度信息

grayInfo.userID;用户id

char worldID用户登录区服id

feInfo首包解压信息，传null，不需要首包解压

feInfo. ifsPath首包解压资源包绝对路径

返回：是否执行成功

(2)virtual bool Uninit() = 0;

功能：反初始化

返回：是否执行成功

(3)virtual void ContinueUpdate(bool bContinue) = 0;

功能：继续更新，在回调OnDolphinVersionInfo之后，如果需要更新调用

(4)virtual void CheckAppUpdate() = 0;

功能：启动版本更新流程

(5)virtual void CancelUpdate() = 0;

功能：取消更新

(6)virtual cu_uint32 GetLastError() = 0;

功能：获取错误码，暂时未完善

(7)virtual bool PollCallback() = 0;

功能：驱动更新，定期调用，在此函数中执行回调

返回：是否执行成功，失败退出更新