游戏语音 GVoice

SDK接入流程

产品文档

腾讯云

文档目录

SDK接入流程
- C++接口
  - Cocos2D SDK接入流程
  - 基本 API
  - 实时语音API
  - 语音消息API
  - 错误码表
- Unity3D
  - Unity3D SDK接入流程
  - 基本API
  - 实时语音API
  - 语音消息API
  - 错误码表
- JAVA
  - Android平台接入流程
- iOS
  - iOS 接入流程

SDK下载
Demo下载

SDK接入流程
C++接口
Cocos2D SDK接入流程

最近更新时间：2018-08-29 09:50:00

本文档介绍了 GVoice 游戏语音 C++ 接口 SDK 的接入方法，适用于 Cocos2D 引擎，或 iOS、Android 平台直接开发的游戏。

1 下载SDK

Cocos2D SDK 下载
Cocos2D Demo 下载

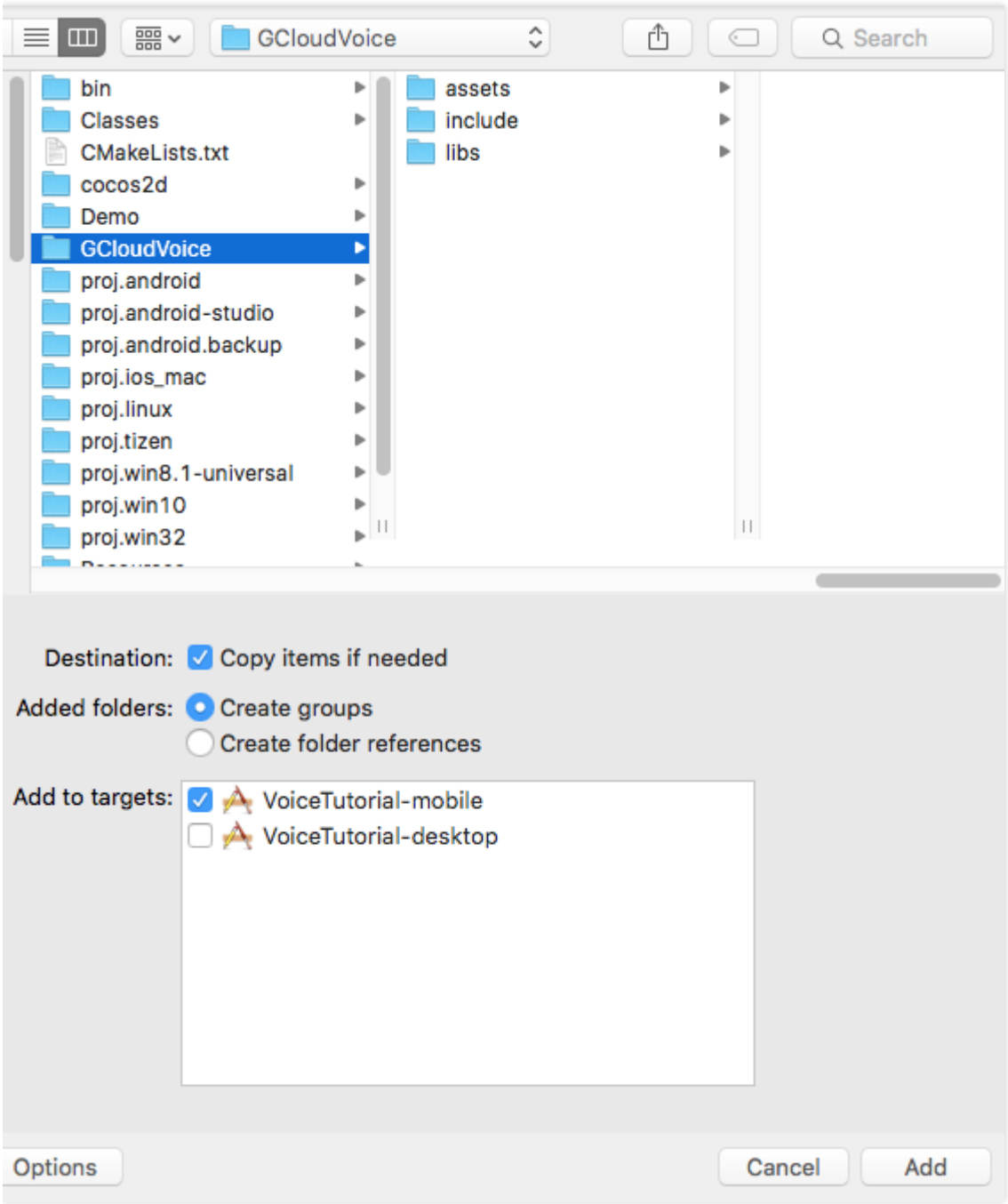
下载 SDK 包解压后，目录结构如下：

```
gcloud_voice_1_1_1_126869_20160826_Cocos
└── dist
    └── Cocos
        ├── assets
        |   └── GCloudVoice
        |       └── config.json
        ├── include
        |   ├── GCloudVoice.h
        |   ├── GCloudVoiceErrno.h
        |   ├── GCloudVoiceExtension.h
        |   └── GCloudVoiceVersion.h
        └── libs
            ├── Android
            |   ├── GCloudVoice.jar
            |   ├── armeabi
            |   |   └── libGCloudVoice.so
            |   ├── armeabi-v7a
            |   |   └── libGCloudVoice.so
            |   └── x86
            |       └── libGCloudVoice.so
            └── iOS
                ├── GCloudVoice.bundle
                |   ├── files
                |   |   └── config.json
                |   └── images
                └── libGCloudVoice.a

14 directories, 11 files
```

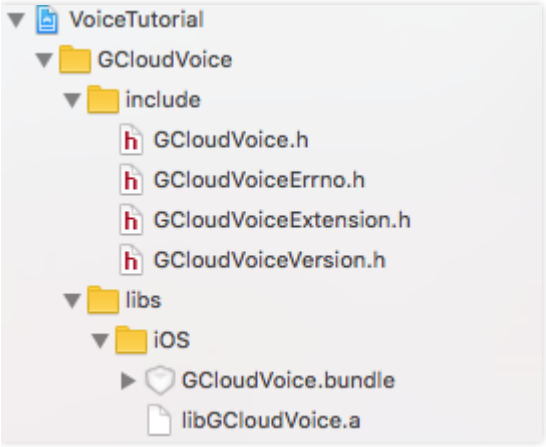
2 系统配置

2.1 iOS系统配置

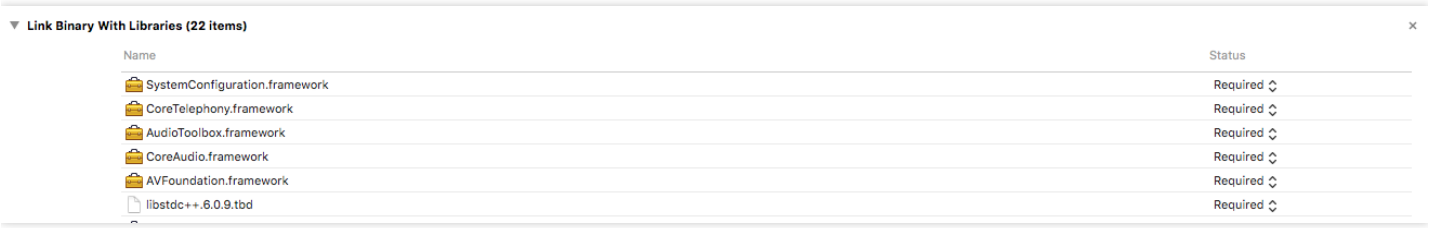
对于 iOS 的 Xcode 工程，只要将 include 目录和 libs/iOS 目录添加到 Xcode 工程中并设置头文件引用位置即可。



在 Xcode 中显示为：



同时需要添加系统库：

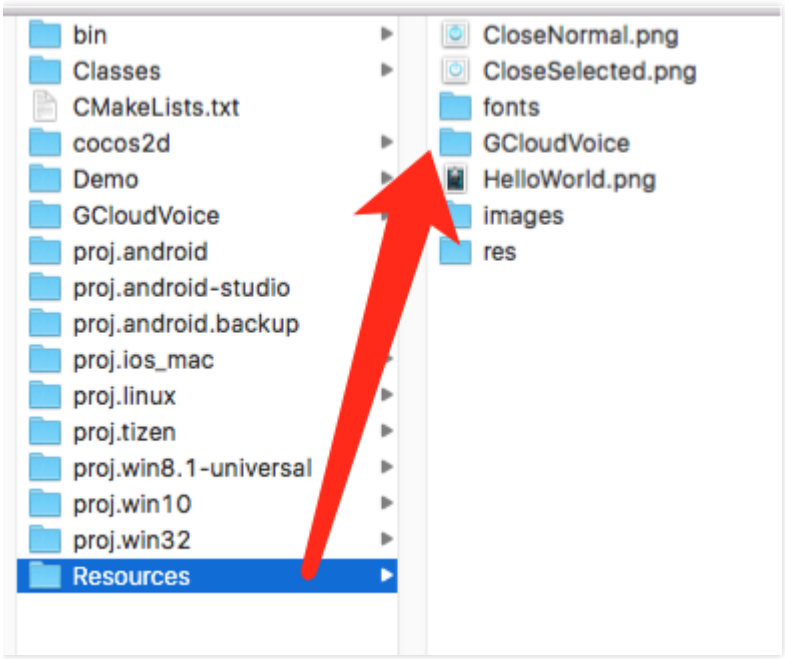


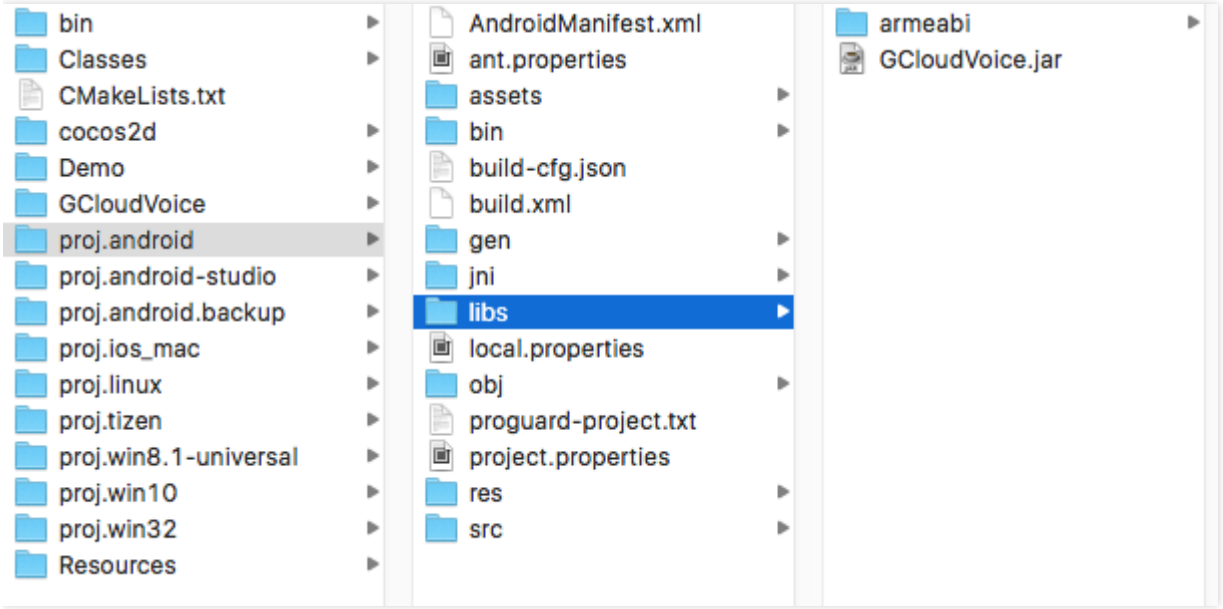
然后按照 Cocos 的编译方式编译就可以了。

2.2 Android 系统

对于 Android 工程，这里以 proj.android 为例，将压缩包中的 assetes 目录中的 GCloudVoice 目录放到 Cocos 工程的Resource 目录下。接着将 libs/Android/GCloudVoice.jar文件放到 proj.android/libs 目录下。最后将

include 和libs/Android 目录放到合适的目录，比如工程下面建一个 GCloudVoice 目录：





在 proj.anroid/jni 的 Android.mk 中添加对库文件和头文件的引用：

```
3 ############################################################
4 include $(CLEAR_VARS)
5 LOCAL_MODULE := libGCloudVoice
6 LOCAL_SRC_FILES := ../../GCloudVoice/libs/Android/$(TARGET_ARCH_ABI)/libGCloudVoice.so
7 include $(PREBUILT_SHARED_LIBRARY)
8 ############################################################
```

```
39 LOCAL_C_INCLUDES := $(LOCAL_PATH)/../../GCloudVoice/include
```

```
59 # _COCOS_LIB_ANDROID_BEGIN
60 LOCAL_SHARED_LIBRARIES += libGCloudVoice
61 # _COCOS_LIB_ANDROID_END
62
63 include $(BUILD_SHARED_LIBRARY)
```

最后在 proj.android/AndroidManifest.xml 添加如下权限即可按照 Cocos 的编译方式编译运行了。

```
android.permission.READ_PHONE_STATE (Uses Permission)
android.permission.RECORD_AUDIO (Uses Permission)
android.permission.MODIFY_AUDIO_SETTINGS (Uses Permission)
android.permission.ACCESS_NETWORK_STATE (Uses Permission)
android.permission.INTERNET (Uses Permission)
```

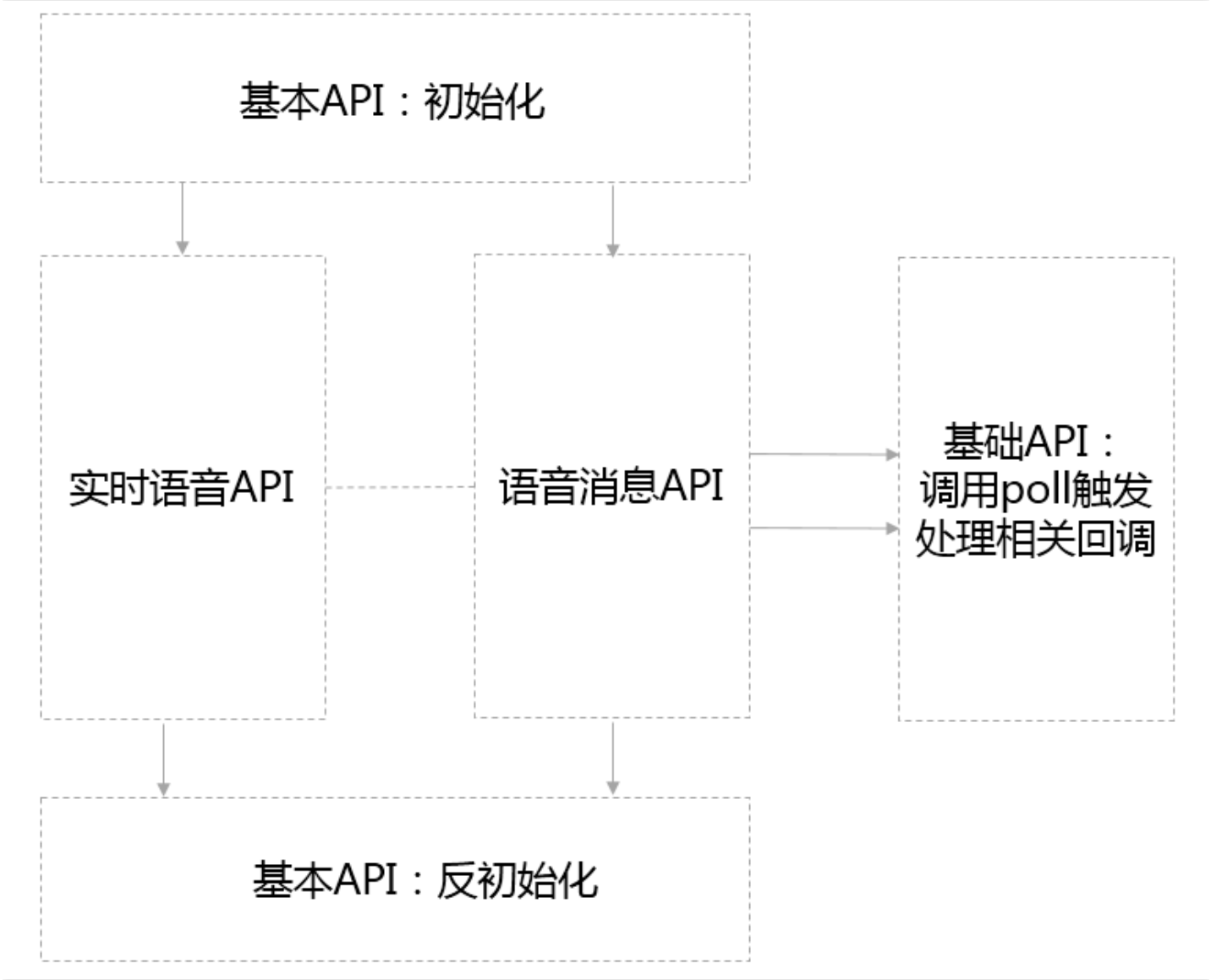
最后需要在 Java 测进行初始化，比如：

```
public class AppActivity extends Cocos2dxActivity {
@Override
protected void onCreate(final Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
GCloudVoiceEngine.getInstance().init(getApplicationContext(), this);
}
}
```

3 接口调用流程

1.基本 API：无论实时语音，还是消息语音功能，都需要调用基本 API,在开始时进行语音的初始化，结束时进行反初始化，以及中间调用 API 时，需要调用 poll 触发处理相关回调，
基本 API 调用 。
2.实时语音 API：实时语音功能调用，
实时语音 API 调用。
3.语音消息 API：消息语音功能调用，

语音消息 API 调用。


基本API：初始化
实时语音API
语音消息API
基础API：
调用poll触发
处理相关回调
基本API：反初始化

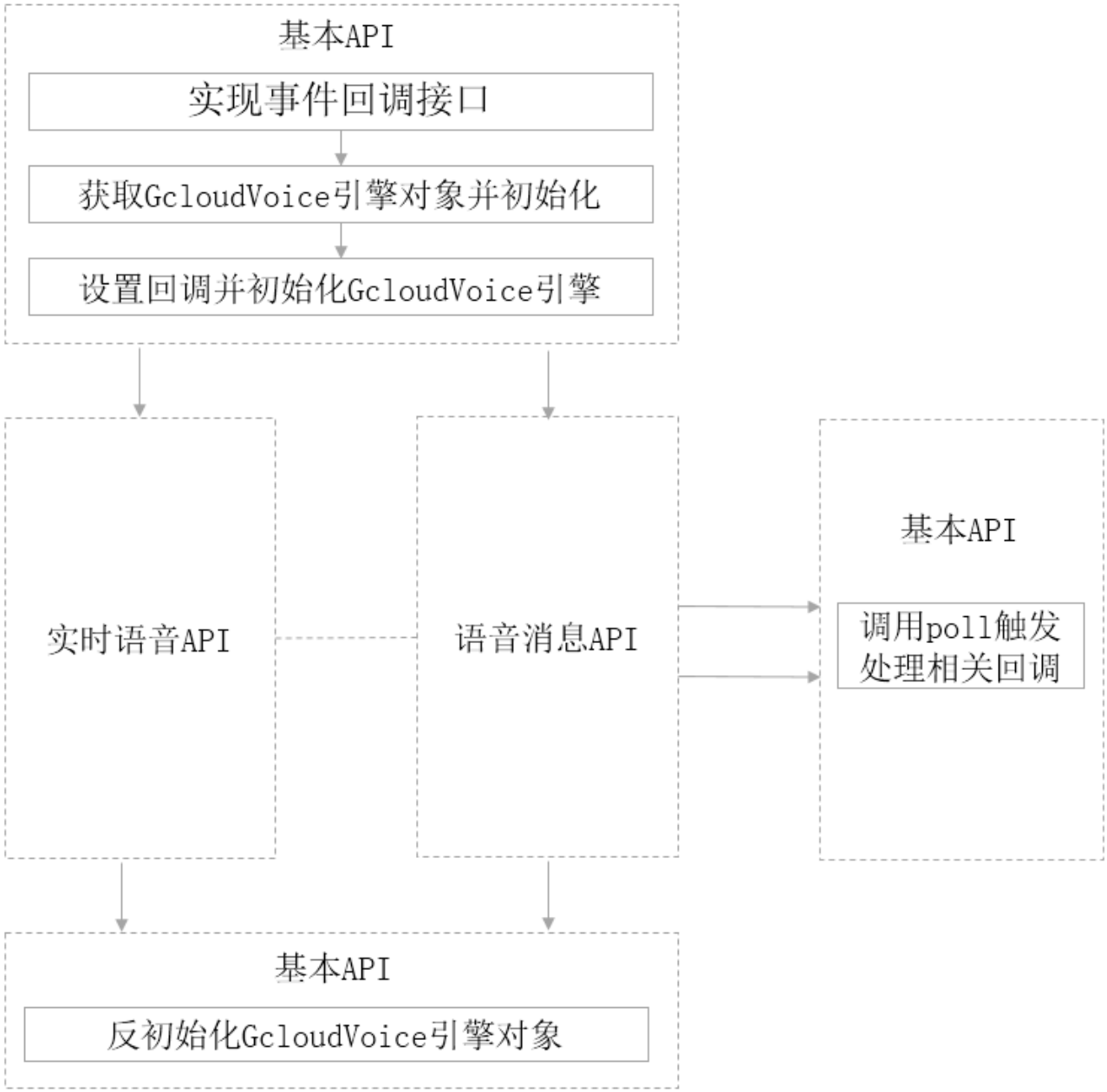
基本 API

最近更新时间：2018-08-29 10:02:12

1 概述

无论使用实时语音，还是语音消息，都需要调用基本 API。

2 基本 API 调用流程



流程说明

1.实现 IGcloudVoiceNotify 回调类。
2.调用 GetVoiceEngine 获取 IGcloudVoiceEngine 对象。
3.对该对象进行初始化操作并设置回调。
4.根据需要调用实时语音接口或语音消息接口。
5.在系统可以 Tick 的地方（如 Unity3D 的 Update ）调用 Poll() 函数驱动程序运行。

3 基本 API

3.1 创建 IGcloudVoiceEngine 对象

接口说明

在使用语音功能时，需要首先获取 VoiceEngine 对象。

函数原型

IGCloudVoiceEngine *GetVoiceEngine()
该函数返回一个IGcloudVoiceEngine对象，通过调用该对象的接口，可以执行相关动作。

示例代码

```
private IGCloudVoiceEngine* m_voiceengine = gcloud_voice::GetVoiceEngine();
```

出错处理

出错时，返回 NULL 对象

3.2 设置业务信息

接口说明

在初始化之前，需要设置之前申请的游戏 ID 和游戏 Key 以及用户的唯一标示 OpenID。

函数原型

GCloudVoiceErrno SetAppInfo(const char *appID,const char *appKey, const char *openID)

参数	类型	意义
appID	const char *	开通业务页面中的游戏ID
appKey	const char *	开通业务页面中的游戏Key
openID	const char *	玩家唯一标示，比如从手Q或者微信获得到的OpenID
返回值	GCloudVoiceErr	成功时返回GCLOUD_VOICE_SUCC

示例代码

`gcloud_voice::GetVoiceEngine()->SetAppInfo("932849489","d94749efe9fce61333121de84123ef9b","E81DCA1782C5CE8B0722A366D7ECB41F");`

3.3 初始化引擎

接口说明

在设置好业务信息后就可以开始初始化引擎了。

函数原型

`GCloudVoiceErr Init();`

示例代码

`gcloud_voice::GetVoiceEngine()->Init();`

出错处理

GCLOUD_VOICE_NEED_SETAPPINFO ： 需要先调用 SetAppInfo

3.4 设置引擎模式

接口说明

在使用语音功能时，根据需要设置成离线、实时还是转文字的模式。如果是小队语音或者国战语音，设置成实时模式；如果是语音消息，设置成离线模式；如果是语音转文字，设置成翻译模式。

函数原型

```
enum GCloudVoiceMode
{
RealTime = 0, // realtime mode for TeamRoom or NationalRoom
Messages, // voice message mode
Translation, // speach to text mode
};
GCloudVoiceErr SetMode(GCloudVoiceMode mode)
```

参数	类型	意义
mode	GCloudVoiceMode	如果是小队语音或者国战语音，设置成实时模式；如果是语音消息，设置成离线模式；如果是语音转文字，设置成翻译模式

示例代码

```
gcloud_voice::GetVoiceEngine()>SetMode(gcloud_voice::IGCloudVoiceEngine::RealTime);
```

出错处理

GCLOUD_VOICE_NEED_SETAPPINFO ：需要先调用 SetAppInfo

3.5 查询触发事件回调

接口说明

通过在update里面周期的调用 Poll 可以触发事件回调

函数原型

`GCloudVoiceErr Poll()`

示例代码

```
// Update is called timer
void TeamRoomSection::update(float delta)
{
gcloud_voice::GetVoiceEngine()->Poll();
}
```

出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化

3.6 系统发生 Pause

接口说明

当系统发生 Pause 事件时，需要同时通知引擎进行 Pause

函数原型

`GCloudVoiceErr Pause()`

示例代码

```
public void OnApplicationPause(bool pauseStatus)
{
gcloud_voice::GetVoiceEngine()->Pause();
}
```

出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化

3.7 系统发生 Resume

接口说明

当系统发生 Resume 事件时，需要同时通知引擎进行 Resume

函数原型

`GCloudVoiceErr Resume()`

示例代码

```
public void OnApplicationPause(bool pauseStatus)
{
gcloud_voice::GetVoiceEngine()->Resume();
}
```

出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化

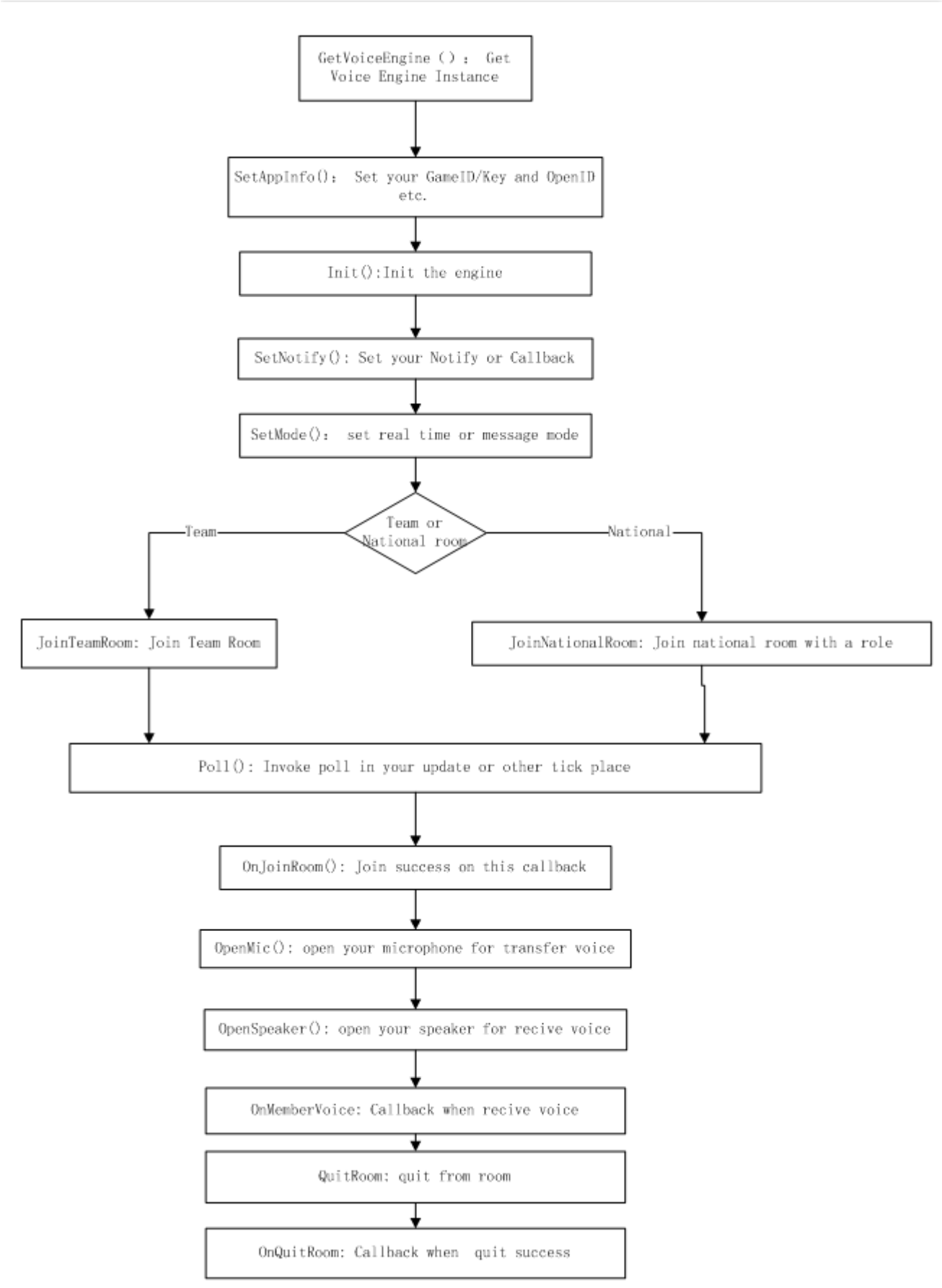
实时语音API

最近更新时间：2017-03-09 10:50:30

1 概述

使用实时语音，需要先调用基本API。

2 实时语音API调用流程


GetVoiceEngine ()： Get Voice Engine Instance
SetAppInfo()： Set your GameID/Key and OpenID etc.
Init():Init the engine
SetNotify(): Set your Notify or Callback
SetMode()： set real time or message mode
Team or National room
Team
National
JoinTeamRoom: Join Team Room
JoinNationalRoom: Join national room with a role
Poll(): Invoke poll in your update or other tick place
OnJoinRoom(): Join success on this callback
OpenMic(): open your microphone for transfer voice
OpenSpeaker(): open your speaker for recive voice
OnMemberVoice: Callback when recive voice
QuitRoom: quit from room
OnQuitRoom: Callback when quit success

流程说明

1.调用 SetMode() 方法设置使用实时语音模式。
2.根据业务需求使用小队语音或国战语音，分别调用 JoinTeamRoom() 或 JoinNationalRoom() 。
3.需要 Tick 的调用 poll 来检查回调，当加入房间成功或者失败时，会回调 OnJoinRoomComplete() 方法。
4.加入房间成功后，就可以调用 OpenMic() 打开麦克风进行采集并发送到网络。
5.调用 OpenSpeaker() 打开扬声器，开始接受网络上的音频流并自动进行播放。
6.当需要退出房间时，调用 QuitRoom() 即可，成功后会回调 OnQuitRoomComplete() 方法。

注意：
对于国战语音，系统要求说话人数不能超过5个人，每个用户多了一个角色信息，在加入房间的时候需要指定是以听众的身份加入还是以主播的身份加入。

3 实时语音API

3.1 加入小队语音

1.接口说明

使用实时语音的小队语音功能时，需要先加入小队语音房间

2.函数原型

GCloudVoiceErrno JoinTeamRoom(const char *roomName, int msTimeout = 10000)

参数	类型	意义
roomName	const char *	要加入的房间名
msTimeout	int	加入房间的超时时间，单位是毫秒

加入房间的结果，需要通过void OnJoinRoom(GCloudVoiceCompleteCode code, const char *roomName, int memberID) ;进行回调

3.示例代码

```
public void JoinRoomBtn_Click()
{
gcloud_voice::GetVoiceEngine()->JoinTeamRoom("cz_test2", 5000);
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 需要先调用SetMode设置成实时模式

GCLOUD_VOICE_PARAM*INVALID* ： *传入的参数不对，比如房间名为空或者超长（最大长度127字节）且由a-z,A-Z,0-9,-,*组成。超时范围5000ms-60000ms。
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如已经加入房间了，需要先调用QuitRoom才能再次加入

3.2 加入国战语音

1.接口说明

使用国战语音功能时，需要先加入国战语音房间

2.函数原型

```
enum GCloudVoiceMemberRole
{
Anchor = 1, // member who can open microphone and say
Audience, // member who can only hear anchor's voice
};
GCloudVoiceErrno JoinNationalRoom(const char *roomName, GCloudVoiceMemberRole role, int ms
Timeout = 10000)
```

参数	类型	意义
roomName	const char *	要加入的房间名
role	GCloudVoiceMemberRole	成员加入的角色，听众只能收听语音不能发送语音，而主播既可以发送语音也可以收听语音
msTimeout	int	加入房间的超时时间，单位是毫秒

加入房间的结果，需要通过void OnJoinRoom(GCloudVoiceCompleteCode code, const char *roomName, int memberID)进行回调

3.示例代码

```
public void AudienceJoin_Click()
{
gcloud_voice::GetVoiceEngine()->JoinNationalRoom("cz_test", gcloud_voice::IGCloudVoiceEngine::Au
dience, 5000);
}
public void AnchorJoin_Click()
{
gcloud_voice::GetVoiceEngine()->JoinNationalRoom("cz_test", gcloud_voice::IGCloudVoiceEngine::An
```

```
chor, 5000);
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 需要先调用SetMode设置成实时模式
GCLOUD_VOICE_PARAM*INVALID* ： *传入的参数不对，比如房间名为空或者超长（最大长度127字节）且由a-z,A-Z,0-9,-,*组成。超时范围5000ms-60000ms。
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如已经加入房间了，需要先调用QuitRoom才能再次加入

3.3 退出实时语音

1.接口说明

当不需要使用实时语音，可以从实时语音（包括小队语音和国战语音）房间中退出。

2.函数原型

```
GCloudVoiceErrno QuitRoom(const char *roomName, int msTimeout = 10000)
```

参数	类型	意义
roomName	const char *	要退出的房间名
msTimeout	int	退出房间的超时时间，单位是毫秒

退出房间的结果，需要通过void OnQuitRoom(GCloudVoiceCompleteCode code, const char *roomName)进行回调

3.示例代码

```
public void QuitRoomBtn_Click()
{
gcloud_voice::GetVoiceEngine()->QuitRoom("cz_test", 5000);
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式
GCLOUD_VOICE_PARAM*INVALID* ： *传入的参数不对，比如房间名为空或者超长（最大长度127字节）且由a-z,A-*

*Z,0-9,-,*组成。超时范围5000ms-60000ms。
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间

3.4 打开麦克风

1.接口说明

在实时语音的模式下，加入房间成功后（包括小队语音和国战语音），需要打开麦克风才能采集音频并发送到网络

2.函数原型

```
GCloudVoiceErr OpenMic();
```

3.示例代码

```
public void OpenMicBtn_Click()
{
gcloud_voice::GetVoiceEngine()->OpenMic();
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间
GCLOUD_VOICE_OPENMIC_NOTANCHOR_ERR ： 当前以听众身份加入的大房间，不能开麦

3.5 关闭麦克风

1.接口说明

在实时语音的模式下，加入房间成功后（包括小队语音和国战语音），当不需要采集音频并发送到网络，可以调用关闭麦克风接口

2.函数原型

```
GCloudVoiceErrno CloseMic();
```

3.示例代码

```
public void CloseMicBtn_Click()
{
gcloud_voice::GetVoiceEngine()->CloseMic ();
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间
GCLOUD_VOICE_OPENMIC_NOTANCHOR_ERR ： 当前以听众身份加入的大房间，不能开麦关麦

3.6 打开扬声器

1.接口说明

在实时语音的模式下，加入房间成功后（包括小队语音和国战语音），需要打开扬声器才能从网络接收数据并播放

2.函数原型

`GCloudVoiceErrno OpenSpeaker();`

3.示例代码

```
public void OpenSpeakerBtn_Click()
{
gcloud_voice::GetVoiceEngine()->OpenSpeaker ();
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间

3.7 关闭扬声器

1.接口说明

在实时语音的模式下，加入房间成功后（包括小队语音和国战语音），当不需要从网络接收数据并播放时，可以调用关闭麦克风接口

2.函数原型

`GCloudVoiceErrno CloseSpeaker();`

3.示例代码

```
public void CloseSpeakerBtn_Click()
{
gcloud_voice::GetVoiceEngine()->CloseSpeaker ();
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间

3.8 加入房间回调

1.接口说明

当加入房间成功或者失败时会通过回调进行通知

2.函数原型

`virtual void OnJoinRoom(GCloudVoiceCompleteCode code, const char *roomName, int memberID) ;`

参数	类型	意义
code	GCloudVoiceCompleteCode	参见GCloudVoiceCompleteCode定义
roomName	const char *	加入的房间名
memberID	int	如果加入成功的话，表示加入后的成员ID

3.示例代码

```
void NationalRoomNotify::OnJoinRoom(gcloud_voice::GCloudVoiceCompleteCode code, const char *
roomName, int memberID)
{
if (code == gcloud_voice::GV_ON_JOINROOM_SUCC) {
_section->setText("Join Team Room Success");
} else {
_section->setText("Join Team Room Error");
}
};
```

3.9 退出房间回调

1.接口说明

当退出房间时，结果通过回调进行通知

2.函数原型

`virtual void OnQuitRoom(GCloudVoiceCompleteCode code, const char *roomName) ;`

参数	类型	意义
code	GCloudVoiceCompleteCode	参见GCloudVoiceCompleteCode定义
roomName	const char *	退出的房间名
memberID	int	如果加入成功的话，表示加入后的成员ID

3.示例代码

```
void NationalRoomNotify::OnQuitRoom(gcloud_voice::GCloudVoiceCompleteCode code, const char * roomName)
{
if (code == gcloud_voice::GV_ON_QUITROOM_SUCC) {
_section->setText("Quit Team Room Success");
} else {
_section->setText("Quit Team Room Error");
}
}
```

3.10 成员状态改变回调

1.接口说明

当房间中的其他成员开始说话或者停止说话的时候，通过该回调进行通知

2.函数原型

```
virtual void OnMemberVoice (const unsigned int *members, int count) ;
```

参数	类型	意义
members	int[]	改变状态的member成员，其值为[memberID | status]这样的对，总共有count对，status有“0”：停止说话 “1”：开始说话 “2”:继续说话
count	int	改变状态的成员的数目

3.示例代码

```
void NationalRoomNotify::OnMemberVoice (const unsigned int *members, int count)
{
for (int i=0; i<count; i++) {
CCLOG("member %d's status is %d", *(members+2*i), *(members+2*i+1));
```

```
}
}
```

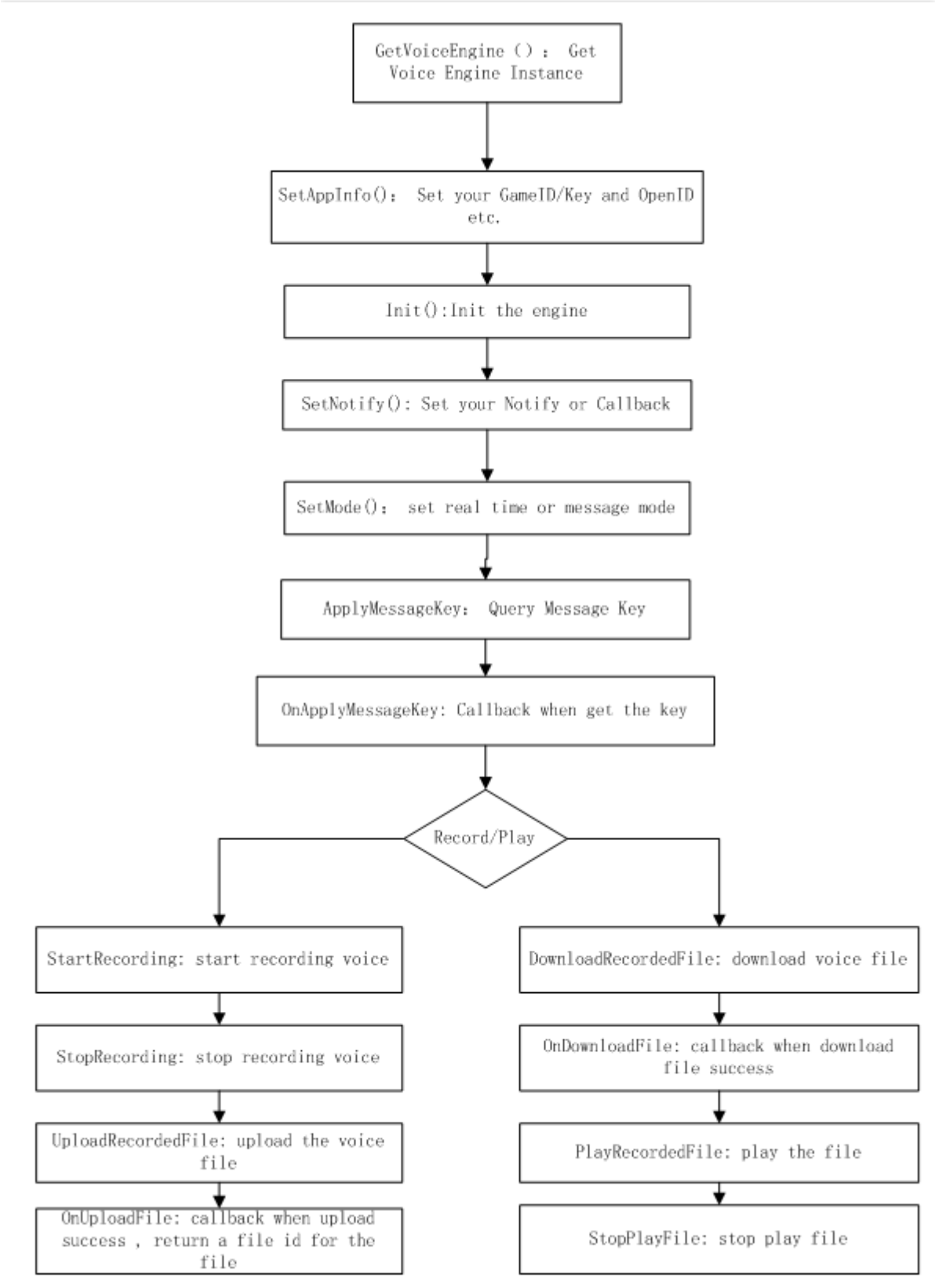
语音消息API

最近更新时间：2018-08-30 16:23:22

1 概述

使用消息语音，需要先调用 基本 API。

2 消息语音API调用流程



流程说明

1.调用 SetMode 方法设置使用语音消息模式。

2.调用 ApplyMessageKey() 获取语音消息安全密钥key信息，当申请成功后会通

过 OnApplyMessageKeyComplete 进行回调。
3.当需要录音时，调用 StartRecording() 录制音频到文件中（文件的路径格式是 /your path ）。
4.如果想取消录制可以调用 StopRecording 接口进行取消。
5.当录制完成后，调用 UploadRecordedFile 将文件上传到 GcloudVoice 的服务器上，该过程会通过 OnUploadReccordFileComplete 回调在上传成功的时候返还一个 ShareFileID .该ID是这个文件的唯一标识符，用于其他用户收听时候的下载。服务器需要对其进行管理和转发。
6.当游戏客户端需要收听其他人的录音时，首先从服务器获取转发的 ShareFileID ，然后调用 DownloadRecordedFile 下载该语言文件，下载结果通过 OnDownloadRecordFileComplete 回调来通知。当下载成功时，就可以调用 PlayRecordedFile 播放下载完成的语音数据了。同样的，如果想取消播放，可以调用 StopPlayFile 进行取消。

3 消息语音详细API

3.1 申请语音消息key

1.接口说明

在语音消息的模式下，需要先申请许可才可以正常使用

2.函数原型

GCloudVoiceErrno ApplyMessageKey(int msTimeout)

参数	类型	意义
msTimeout	itn	超时时间，单位毫秒

申请的结果通过void OnApplyMessageKey(GCloudVoiceCompleteCode code) ;进行回调

3.示例代码

```
void Click_btnApplyMessageKey()
{
gcloud_voice::GetVoiceEngine()->ApplyMessageKey (6000);
}
```

4.出错处理

GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，比如超时范围5000ms-60000ms
GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_AUTHKEY_ERR ： 请求Key的内部错误，此时需要联系GCloud团队，并提供日志进行定位

3.2 限制最大语音消息的长度

1.接口说明

在语音消息的模式下，可以限制最大语音消息的长度，目前默认是2min，最大不超过2min。

2.函数原型

`GCloudVoiceErrno SetMaxMessageLength(int msTime)`

参数	类型	意义
msTimeout	itn	最大语音消息长度，单位毫秒

3.示例代码

`int ret1 = gcloud_voice::GetVoiceEngine()->SetMaxMessageLength (60000);`

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，时间范围1000ms-1000260ms。

3.3 开始录音

1.接口说明

在语音消息的模式下，开始录音时，需要提供一个录音文件存储的地址路径

2.函数原型

`GCloudVoiceErrno StartRecording(const char * filePath)`

参数	类型	意义
filePath	const char *	录音文件存储的地址路径，路径中需要"/"作分隔，不能用"\"

3.示例代码

```
public void Click_btnStartRecord()
{
gcloud_voice::GetVoiceEngine()->StartRecording (m_recordpath);

}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，路径为空。
GCLOUD_VOICE_NEED_AUTHKEY ： 需要先调用GetAuthKey申请许可
GCLOUD_VOICE_PATH_ACCESS_ERR ： 提供的路径不合法或者不可写

3.4 停止录音

1.接口说明

在语音消息的模式下，调用停止录音接口

2.函数原型

`GCloudVoiceErrno StopRecording()`

3.示例代码

```
public void Click_btnStopRecord()
{
gcloud_voice::GetVoiceEngine()->StopRecording ();
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式
GCLOUD_VOICE_NEED_AUTHKEY ： 需要先调用GetAuthKey申请许可

3.5 上传录音的文件

1.接口说明

录音完成后，通过提供一个录音文件存储的地址路径，将已经录音完的文件进行上传

2.函数原型

`GCloudVoiceErrno UploadRecordedFile(const char * filePath, int msTimeout = 60000)`

参数	类型	意义
filePath	const char *	录音文件存储的地址路径，路径中需要"/"作分隔，不能用"\"
msTimeout	int	上传文件超时时间

上传的结果通过void OnUploadFile(GCloudVoiceCompleteCode code, const char *filePath, const char* fileID) 进行回调

3.示例代码

```
public void Click_btnUploadFile()
{
int ret1 = gcloud_voice::GetVoiceEngine()->UploadRecordedFile (m_recordpath, 60000);
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，路径为空
GCLOUD_VOICE_NEED_AUTHKEY ： 需要先调用GetAuthKey申请许可
GCLOUD_VOICE_PATH_ACCESS_ERR ： 提供的路径不合法或者不可读
GCLOUD_VOICE_HTTP_BUSY ： 还在上一次上传或者下载中，需要等待后再尝试

3.6 下载录音的文件

1.接口说明

录音完成后，通过提供一个录音文件存储的地址路径，将已经录音完的文件进行上传

2.函数原型

```
GCloudVoiceErrno DownloadRecordedFile (const char *fileID, const char * downloadFilePath, int msTimeout = 60000);
```

参数	类型	意义
fileID	const char *	要下载文件的文件ID
downloadFilePath	const char *	下载录音文件存储的地址路径，路径中需要"/"作分隔，不能用"\"
msTimeout	int	下载文件超时时间

下载的结果通过void OnDownloadFile(GCloudVoiceCompleteCode code, const char *filePath, const char* fileID) ;进行回调

3.示例代码

```
public void Click_btnDownloadFile()
{
int ret = gcloud_voice::GetVoiceEngine()->DownloadRecordedFile (m_fileid, m_downloadpath, 60000
);
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，路径为空
GCLOUD_VOICE_NEED_AUTHKEY ： 需要先调用GetAuthKey申请许可
GCLOUD_VOICE_PATH_ACCESS_ERR ： 提供的路径不合法或者不可写或者不可读
GCLOUD_VOICE_HTTP_BUSY ： 还在上一次上传或者下载中，需要等待后再尝试

3.7 开始播放下载的音频

1.接口说明

下载下来的音频文件，需要调用相关接口进行播放

2.函数原型

`GCloudVoiceErrno PlayRecordedFile (const char * downloadFilePath)`

参数	类型	意义
filePath	const char*	下载文件存储的地址路径，路径中需要"/"作分隔，不能用"\"

如果正常播放完，会回调void OnPlayRecordedFile(GCloudVoiceCompleteCode code,const char *filePath)

3.示例代码

```
public void Click_btnPlayReocrdFile()
{
int err;
err = gcloud_voice::GetVoiceEngine()->PlayRecordedFile(m_downloadpath);
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式

GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，路径为空
GCLOUD_VOICE_PATH_ACCESS_ERR ： 提供的路径不合法或者不可写
GCLOUD_VOICE_SPEAKER_ERR : 打开麦克风失败

3.8 停止播放下载的音频

1.接口说明

中断播放动作

2.函数原型

`GCloudVoiceErrno StopPlayFile()`

3.示例代码

```
public void Click_btnStopPlayRecordFile()
{
gcloud_voice::GetVoiceEngine()->StopPlayFile ();
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式

3.9 请求语音消息Key回调

1.接口说明

请求语音消息许可的时候会回调

2.函数原型

`void OnApplyMessageKey(GCloudVoiceCompleteCode code) ;`

参数	类型	意义
code	GCloudVoiceCompleteCode	参见GCloudVoiceCompleteCode定义

3.示例代码

```
void MessageNotify::OnApplyMessageKey(gcloud_voice::GCloudVoiceCompleteCode code)
{
std::string msg;
if (code == gcloud_voice::GV_ON_MESSAGE_KEY_APPLIED_SUCC) {
```

```
msg = "OnApplyMessageKey success";
} else {
msg = "OnApplyMessageKey error " + code;
}
_section->setText(msg);
}
```

3.10 上传完成回调

1.接口说明

上传语音文件后的结果通过这个进行回调

2.函数原型

```
void OnUploadFile(GCloudVoiceCompleteCode code, const char *filePath, const char *fileID)
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见GCloudVoiceCompleteCode定义
filepath	const char *	上传的文件路径
fileid	const char *	文件的id

3.示例代码

```
void MessageNotify::OnUploadFile(gcloud_voice::GCloudVoiceCompleteCode code, const char *filePa
th, const char *fileID)
{
if (code == gcloud_voice::GV_ON_UPLOAD_RECORD_DONE) {
_section->setText("OnUploadFile success");
_section->setFileID((char *)fileID);
} else {
_section->setText("OnUploadFile error");
}
}
```

3.11 下载完成回调

1.接口说明

下载语音文件后的结果通过这个进行回调

2.函数原型

```
void OnDownloadFile(GCloudVoiceCompleteCode code, const char *filePath, const char *fileID) ;
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见GCloudVoiceCompleteCode定义
filepath	const char *	下载的路径
fileid	const char *	文件的id

3.示例代码

```
void MessageNotify::OnDownloadFile(gcloud_voice::GCloudVoiceCompleteCode code, const char *fil
ePath, const char *fileID)
{
if (code == gcloud_voice::GV_ON_DOWNLOAD_RECORD_DONE) {
_section->setText("OnDownloadFile success");
} else {
_section->setText("OnDownloadFile error");
}
}
```

3.12 正常播放完成后回调

1.接口说明

如果用户没有暂停播放，而语音文件已经播放完了，通过这个进行回调

2.函数原型

```
void OnPlayRecordedFile(GCloudVoiceCompleteCode code,const char *filePath)
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见GCloudVoiceCompleteCode定义
filepath	const char *	播放的文件路径

3.示例代码

```
void MessageNotify::OnPlayRecordedFile(gcloud_voice::GCloudVoiceCompleteCode code,const char
*filePath)
```

```
{
string str="play file end";
}
```

错误码表

最近更新时间：2017-03-09 01:34:17

1 概述

本文档介绍了GVoice游戏语音C++接口SDK的错误码。

2 错误码表

错误	十六进制值	十进制值	意义
GCLOUD_VOICE_SUCC	0x0	0	Success
GCLOUD_VOICE_PARAM_NULL	0x1001	4097	some param is null
GCLOUD_VOICE_NEED_SETAPPINFO	0x1002	4098	you should call SetAppInfo first before call other api
GCLOUD_VOICE_INIT_ERR	0x1003	4099	Init Erro
GCLOUD_VOICE_RECORDING_ERR	0x1004	4100	now is recording, can't do other operator
GCLOUD_VOICE_POLL_BUFF_ERR	0x1005	4101	poll buffer is not enough or null
GCLOUD_VOICE_MODE_STATE_ERR	0x1006	4102	call some api, but the mode is not correct, maybe you shoud call SetMode first and correct
GCLOUD_VOICE_PARAM_INVALID	0x1007	4103	some param is null or value is invalid for our request, used right param and make sure is value range is correct by our comment
GCLOUD_VOICE_OPENFILE_ERR	0x1008	4104	open a file err

错误	十六进制值	十进制值	意义
GCLOUD_VOICE_NEED_INIT	0x1009	4105	you should call Init before do this operator
GCLOUD_VOICE_ENGINE_ERR	0x100A	4106	you have not get engine instance, this common in use c# api. but not get gcloudvoice instance first
GCLOUD_VOICE_POLL_MSG_PARSE_ERR	0x100B	4107	this common in c# api, parse poll msg err
GCLOUD_VOICE_POLL_MSG_NO	0x100C	4108	poll no msg to update
GCLOUD_VOICE_REALTIME_STATE_ERR	0x2001	8193	call some realtime api, but state err. such as OpenMic but you have not Join Room first
GCLOUD_VOICE_JOIN_ERR	0x2002	8194	join room failed
GCLOUD_VOICE_QUIT_ROOMNAME_ERR	0x2003	8195	quit room err, the quit roomname not equal join roomname
GCLOUD_VOICE_OPENMIC_NOTANCHOR_ERR	0x2004	8196	open mic in bigroom,but not anchor role
GCLOUD_VOICE_AUTHKEY_ERR	0x3001	12289	apply authkey api error
GCLOUD_VOICE_PATH_ACCESS_ERR	0x3002	12290	the path can not access ,may be path file not exists or deny to access
GCLOUD_VOICE_PERMISSION_MIC_ERR	0x3003	12291	you have not right to access micphone in android
GCLOUD_VOICE_NEED_AUTHKEY	0x3004	12292	you have not get authkey, call ApplyMessageKey first
GCLOUD_VOICE_UPLOAD_ERR	0x3005	12293	upload file err

错误	十六进制值	十进制值	意义
GCLOUD_VOICE_HTTP_BUSY	0x3006	12294	http is busy,maybe the last upload/download not finish.
GCLOUD_VOICE_DOWNLOAD_ERR	0x3007	12295	download file err
GCLOUD_VOICE_SPEAKER_ERR	0x3008	12296	open or close speaker tve error
GCLOUD_VOICE_TVE_PLAYSOUND_ERR	0x3009	12297	tve play file error
GCLOUD_VOICE_INTERNAL_TVE_ERR	0x5001	20481	internal TVE err, our used
GCLOUD_VOICE_INTERNAL_VISIT_ERR	0x5002	20482	internal Not TVE err, out used
GCLOUD_VOICE_INTERNAL_USED	0x5003	20483	internal used, you should not get this err num

Unity3D
Unity3D SDK接入流程

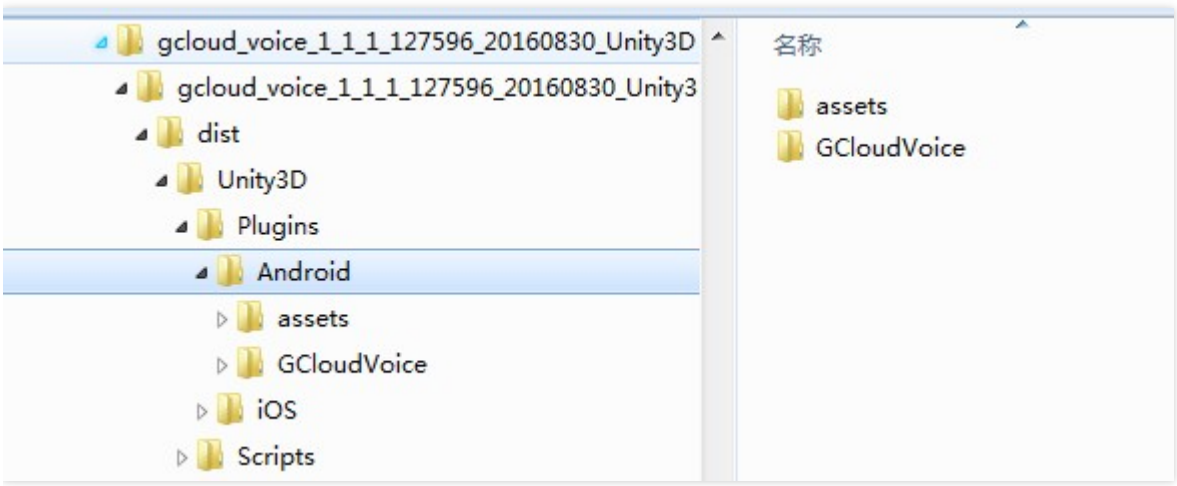
最近更新时间：2017-12-13 17:13:00

本文档介绍了GVoice游戏语音C#接口SDK的接入方法，适用于Unity3D引擎开发的游戏。

1 下载SDK

Unity3D SDK 下载
Unity3D Demo 下载

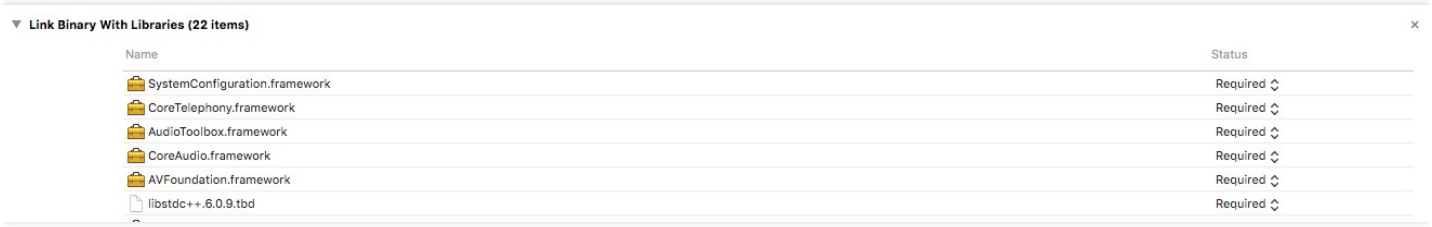
下载SDK包解压后，目录结构如下：



2 系统配置

2.1 iOS系统配置

1.将压缩包中的 dist\Unity3D\Plugins\iOS\ GCloudVoice.bundle 整个文件夹拷贝到自己的工程 MyProj\Assets\Plugins\iOS 目录下。
2.链接发布包中静态库 dist\Unity3D\Plugins\iOS\libGCloudVoice.a ，在 Unity3d 导出的 Xcode 工程中，同时需要链接如下6个系统库：



2.2 Android系统

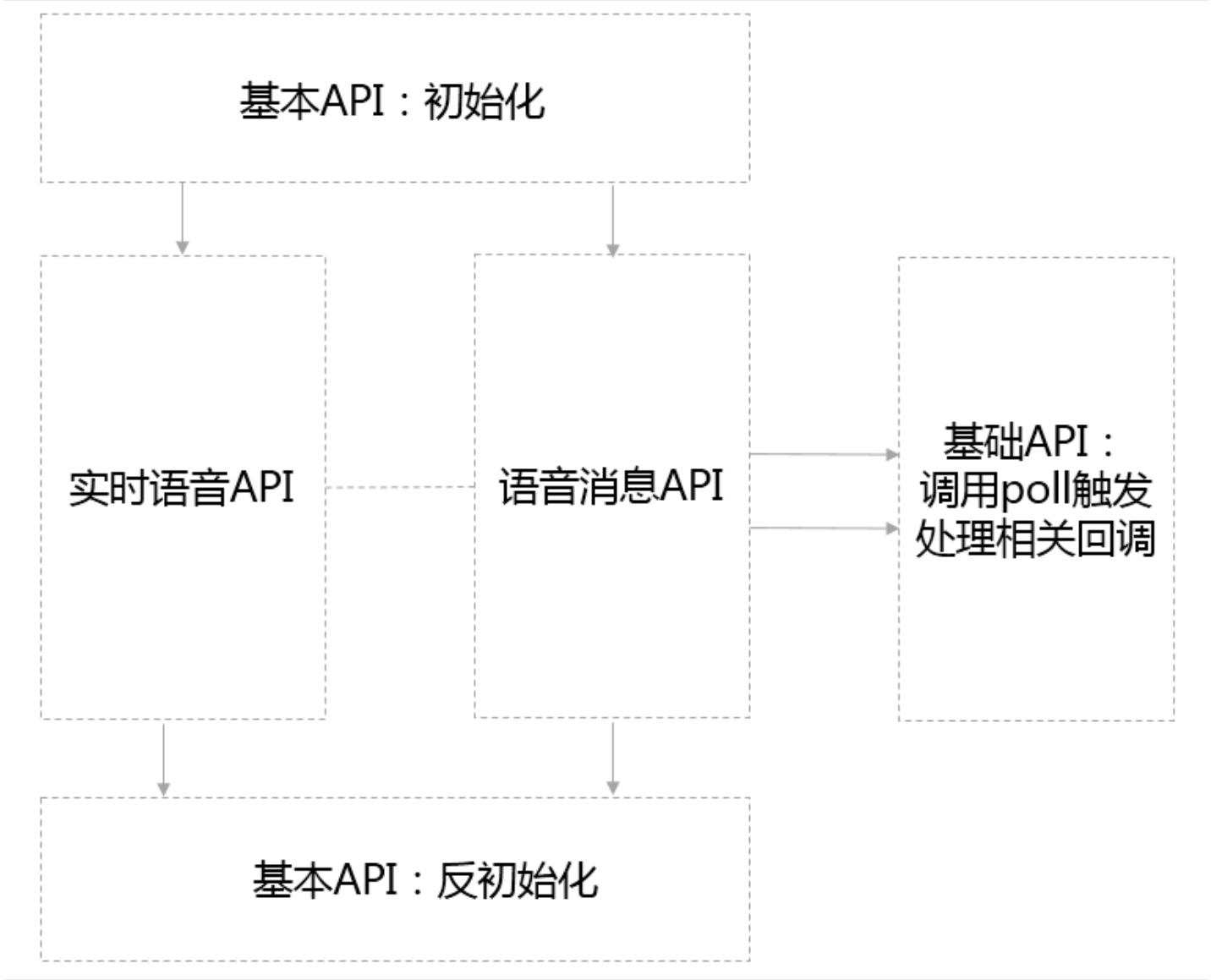
1.将压缩包中的 dist\Unity3D\Plugins\Android 目录下的 GCloudVoice 、 assets 两个文件夹拷贝到自己的工程 MyProj\Assets\Plugins\Android 目录下。
2.将压缩包中的 dist\Unity3D\Scripts\GCloudVoice 目录下的 cs 脚本文件，拷贝到工程 Scripts 文件夹中，使用即可。注意，使用时命名空间为 gcloud_voice 。
3.在游戏主 Activiy 的 OnCreate 函数处，添加 Java 层的 Init 代码。 导入类： import com.tencent.gcloud.voice.GCloudVoiceEngine; 调用 Init 代码：

```
public class MainActivity extends UnityPlayerActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        GCloudVoiceEngine.getInstance().init(getApplicationContext(), this);
    }


}
```

3 接口调用流程

1.基本API：无论实时语音，还是消息语音功能，都需要调用基本API,在开始时进行语音的初始化，结束时进行反初始化，以及中间调用API时，需要调用poll触发处理相关回调，
基本API调用 。
2.实时语音API：实时语音功能调用，
实时语音API调用。
3.语音消息API：消息语音功能调用，

语音消息API调用。


基本API：初始化
实时语音API
语音消息API
基础API：
调用poll触发
处理相关回调
基本API：反初始化

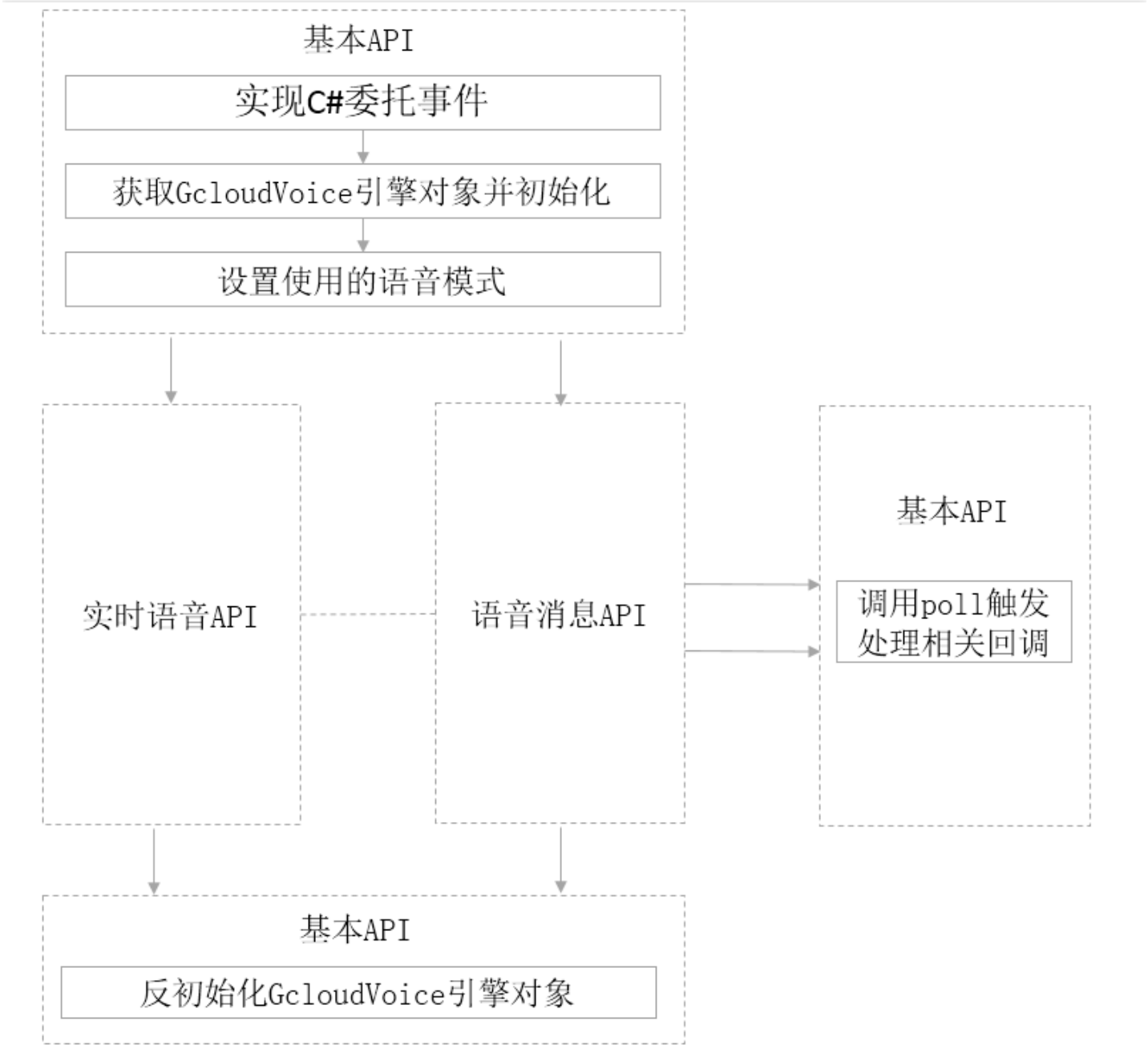
基本API

最近更新时间：2017-12-13 16:54:42

1 概述

无论使用实时语音，还是语音消息，都需要调用基本API。

2 基本API调用流程



流程说明

1.实现相关 Handler 的事件委托。
2.调用 GetEngine 获取 GCloudVoice 对象。
3.对该对象进行初始化操作并设置回调。
4.根据需要调用实时语音接口或语音消息接口。
5.在系统可以 Tick 的地方（如 Unity3D 的 Update ）调用 Poll() 函数驱动程序运行。

3 基本API

3.1 获取VoiceEngine对象

1.接口说明

在使用语音功能时，需要首先获取VoiceEngine对象 。

2.函数原型

```
IGCloudVoice GetEngine()
该函数返回一个实现了IGCloudVoice接口的对象，通过调用该对象的接口，可以执行相关动作。
```

3.出错处理

出错时，返回null对象

4.示例代码

```
//engine have init int mainscene start function
private IGCloudVoice m_voiceengine = GCloudVoice.GetEngine();
```

3.2 设置业务信息

1.接口说明

在初始化之前，需要设置之前申请的游戏ID和游戏Key以及用户的唯一标示OpenID。

2.函数原型

```
GCloudVoiceErr SetAppInfo(string appID, string appKey, string openID)
```

参数	类型	意义
appID	string	开通业务页面中的游戏ID
appKey	string	开通业务页面中的游戏Key
openID	string	玩家唯一标示，比如从手Q或者微信获得到的OpenID
返回值	GCloudVoiceErr	成功时返回GCLOUD_VOICE_SUCC

3.示例代码

m_voiceengine.SetAppInfo("932849489","d94749efe9fce61333121de84123ef9b","E81DCA1782C5CE8B0722A366D7ECB41F");

3.3 初始化引擎

1.接口说明

在设置好业务信息后就可以开始初始化引擎了。

2.函数原型

GCloudVoiceErr Init();

3.示例代码

```
m_voiceengine.SetAppInfo("932849489","d94749efe9fce61333121de84123ef9b","E81DCA1782C5CE8B0722A366D7ECB41F");
m_voiceengine.Init();
```

4.出错处理

GCLOUD_VOICE_NEED_SETAPPINFO ： 需要先调用SetAppInfo

3.4 设置引擎模式

1.接口说明

在使用语音功能时，需要根据需要设置成离线、实时还是转文字的模式。如果是小队语音或者国战语音，设置成实时模式；如果是语音消息，设置成离线模式；如果是语音转文字，设置成翻译模式。

2.函数原型

```
public enum GCloudVoiceMode
{
RealTime = 0, // realtime mode for TeamRoom or NationalRoom
Messages, // voice message mode
Translation, // speach to text mode
};
GCloudVoiceErr SetMode(GCloudVoiceMode mode)
```

参数	类型	意义
mode	GCloudVoiceMode	如果是小队语音或者国战语音，设置成实时模式；如果是语音消息，设置成离线模式；如果是语音转文字，设置成翻译模式

3.示例代码

```
public void RealTimeBtn_Click()
{
Debug.Log ("realtime button click");
m_voiceengine.SetMode (GCloudVoiceMode.RealTime);
Application.LoadLevel ("RealTimeVoice");
}
```

4.出错处理

GCLOUD_VOICE_NEED_SETAPPINFO ：需要先调用SetAppInfo

3.5 查询触发事件回调

1.接口说明

通过在update里面周期的调用Poll可以触发事件回调

2.函数原型

GCloudVoiceErr Poll()

3.示例代码

```
// Update is called once per frame
void Update () {
if (m_voiceengine != null) {
m_voiceengine.Poll();
}

m_logtext.text = s_logstr;
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化

3.6 系统发生Pause

1.接口说明

当系统发生Pause事件时，需要同时通知引擎进行Pause

2.函数原型

```
GCloudVoiceErr Pause()
```

3.示例代码

```
public void OnApplicationPause(bool pauseStatus)
{
Debug.Log("Voice OnApplicationPause: " + pauseStatus);
if (pauseStatus)
{
if (m_voiceengine == null)
{
return;
}
m_voiceengine.Pause();
}
...
}
```

4.出错处理

```
GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
```

3.7 系统发生Resume

1.接口说明

当系统发生Resume事件时，需要同时通知引擎进行Resume

2.函数原型

`GCloudVoiceErr Resume()`

3.示例代码

```
public void OnApplicationPause(bool pauseStatus)
{
Debug.Log("Voice OnApplicationPause: " + pauseStatus);
if (pauseStatus)
{
...
```

```
}
else
{
if (m_voiceengine == null)
{
return;
}
m_voiceengine.Resume();
}
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化

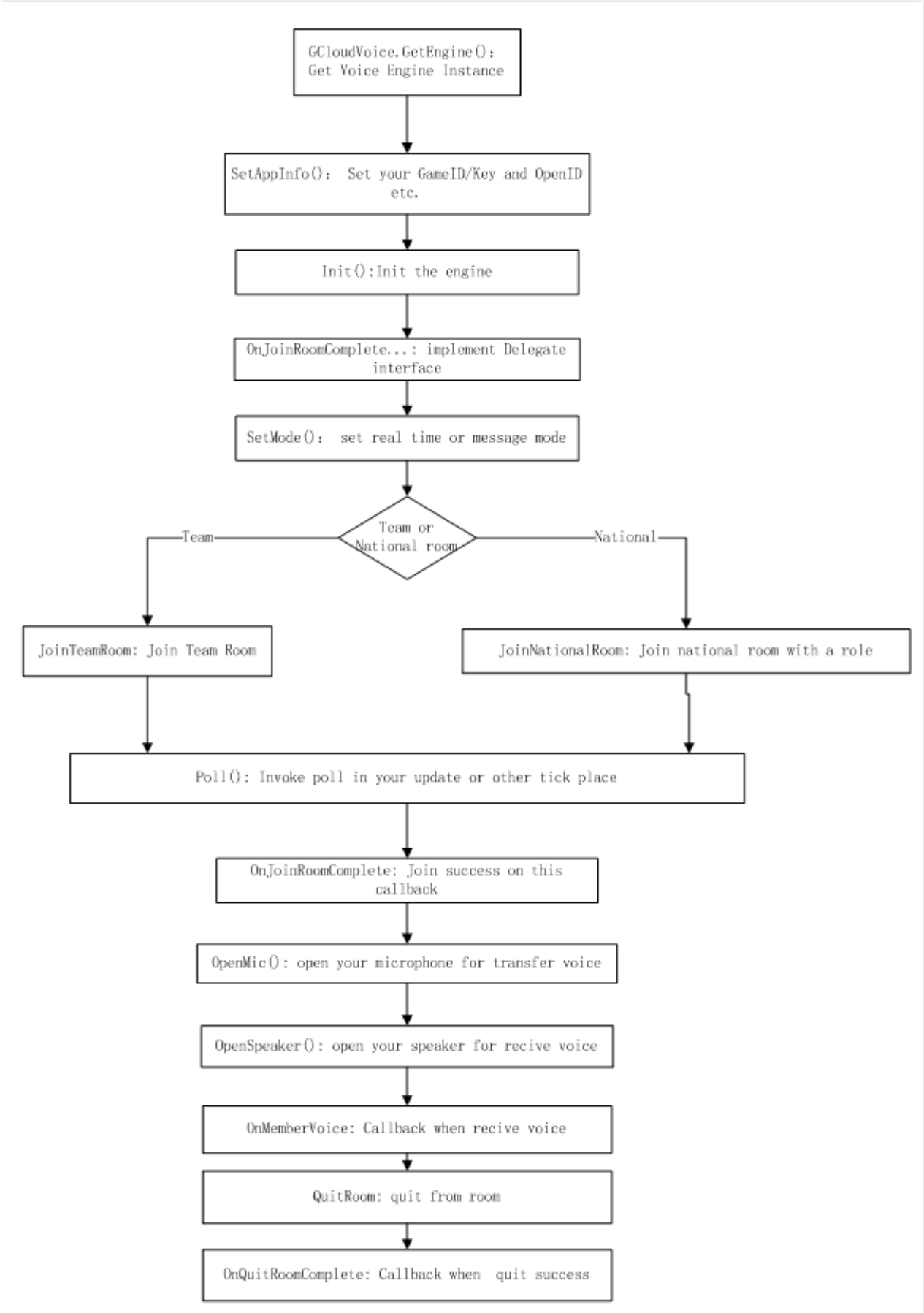
实时语音API

最近更新时间：2017-03-09 10:23:57

1 概述

使用实时语音，需要先调用基本API。

2 实时语音API调用流程


GCloudVoice.GetEngine():
Get Voice Engine Instance
SetAppInfo(): Set your GameID/Key and OpenID etc.
Init():Init the engine
OnJoinRoomComplete...: implement Delegate interface
SetMode(): set real time or message mode
Team or National room
Team
National
JoinTeamRoom: Join Team Room
JoinNationalRoom: Join national room with a role
Poll(): Invoke poll in your update or other tick place
OnJoinRoomComplete: Join success on this callback
OpenMic(): open your microphone for transfer voice
OpenSpeaker(): open your speaker for recive voice
OnMemberVoice: Callback when recive voice
QuitRoom: quit from room
OnQuitRoomComplete: Callback when quit success

流程说明

1.调用 SetMode() 方法设置使用实时语音模式。
2.根据业务需求使用小队语音或国战语音，分别调用 JoinTeamRoom() 或 JoinNationalRoom() 。
3.需要 Tick 的调用 poll 来检查回调，当加入房间成功或者失败时，会回调 OnJoinRoomComplete() 方法。
4.加入房间成功后，就可以调用 OpenMic() 打开麦克风进行采集并发送到网络。
5.调用 OpenSpeaker() 打开扬声器，开始接受网络上的音频流并自动进行播放。
6.当需要退出房间时，调用 QuitRoom() 即可，成功后会回调 OnQuitRoomComplete() 方法。

注意

对于国战语音，系统要求说话人数不能超过5个人，每个用户多了一个角色信息，在加入房间的时候需要指定是以听众的身份加入还是以主播的身份加入。

3 实时语音API

3.1 加入小队语音

1.接口说明

使用实时语音的小队语音功能时，需要先加入小队语音房间

2.函数原型

```
GCloudVoiceErr JoinTeamRoom(string roomName, int msTimeout)
```

参数	类型	意义
roomName	string	要加入的房间名
msTimeout	int	加入房间的超时时间，单位是毫秒

加入房间的结果，需要通过delegate void JoinRoomCompleteHandler(GCloudVoiceCompleteCode code, string roomName, int memberID)进行回调

3.示例代码

```
public void JoinRoomBtn_Click()
{
Debug.Log ("JoinRoom Btn Click");

int ret = m_voiceengine.JoinTeamRoom(m_roomName, 15000);
PrintLog ("joinroom ret=" + ret);
```

```
}
```

4.出错处理

```
GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 需要先调用SetMode设置成实时模式
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，比如房间名为空或者超长（最大长度127字节）且由a-z,A-Z,0-9,-,_组成。超时范围5000ms-60000ms。
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如已经加入房间了，需要先调用QuitRoom才能再次加入
```

3.2 加入国战语音

1.接口说明

使用国战语音功能时，需要先加入国战语音房间

2.函数原型

```
public enum GCloudVoiceRole
{
ANCHOR = 1, // member who can open microphone and say
AUDIENCE, // member who can only hear anchor's voice
}
GCloudVoiceErr JoinNationalRoom(string roomName, GCloudVoiceRole role, int msTimeout)
```

参数	类型	意义
roomName	string	要加入的房间名
role	GCloudVoiceRole	成员加入的角色，听众只能收听语音不能发送语音，而主播既可以发送语音也可以收听语音
msTimeout	int	加入房间的超时时间，单位是毫秒

加入房间的结果，需要通过delegate void JoinRoomCompleteHandler(GCloudVoiceCompleteCode code, string roomName, int memberID)进行回调

3.示例代码

```
public void AudienceJoin_Click()
{
Debug.Log ("AudienceJoin Btn Click");
int ret = m_voiceengine.JoinNationalRoom(m_roomName, GCloudVoiceRole.AUDIENCE, 15000);
PrintLog ("AudienceJoin ret=" + ret);
}
public void AnchorJoin_Click()
{
Debug.Log ("AnchorJoin Btn Click");
int ret = m_voiceengine.JoinNationalRoom(m_roomName, GCloudVoiceRole.ANCHOR, 15000);
PrintLog ("AnchorJoin ret=" + ret);
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 需要先调用SetMode设置成实时模式
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，比如房间名为空或者超长（最大长度127字节）且由a-z,A-Z,0-9,-,_组成。超时范围5000ms-60000ms。
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如已经加入房间了，需要先调用QuitRoom才能再次加入

3.3 退出实时语音

1.接口说明

当不需要使用实时语音，可以从实时语音（包括小队语音和国战语音）房间中退出。

2.函数原型

```
GCloudVoiceErr QuitRoom(string roomName, int msTimeout);
```

参数	类型	意义
roomName	string	要退出的房间名
msTimeout	int	退出房间的超时时间，单位是毫秒

退出房间的结果，需要通过delegate void QuitRoomCompleteHandler(GCloudVoiceCompleteCode code, string roomName, int memberID)进行回调

3.示例代码

```
public void QuitRoomBtn_Click()
{
Debug.Log ("quit room btn click");
m_voiceengine.QuitRoom(m_roomName, 15000);
}
```

4.出错处理

```
GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，比如房间名为空或者超长（最大长度127字节）且由a-z,A-Z,0-9,-,_组成。超时范围5000ms-60000ms。
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间
```

3.4 打开麦克风

1.接口说明

在实时语音的模式下，加入房间成功后（包括小队语音和国战语音），需要打开麦克风才能采集音频并发送到网络

2.函数原型

`GCloudVoiceErr OpenMic();`

3.示例代码

```
public void OpenMicBtn_Click()
{
Debug.Log ("Open Mic btn clieck");
int ret = m_voiceengine.OpenMic ();
PrintLog ("openmic ret=" + ret);
}
```

4.出错处理

```
GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间
GCLOUD_VOICE_OPENMIC_NOTANCHOR_ERR ： 当前以听众身份加入的大房间，不能开麦
```

3.5 关闭麦克风

1.接口说明

在实时语音的模式下，加入房间成功后（包括小队语音和国战语音），当不需要采集音频并发送到网络，可以调用关闭麦克风接口

2.函数原型

`GCloudVoiceErr CloseMic();`

3.示例代码

```
public void CloseMicBtn_Click()
{
Debug.Log ("CoseMic btn click");
m_voiceengine.CloseMic ();
}
```

4.出错处理

```
GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间
GCLOUD_VOICE_OPENMIC_NOTANCHOR_ERR ： 当前以听众身份加入的大房间，不能开麦关麦
```

3.6 打开扬声器

1.接口说明

在实时语音的模式下，加入房间成功后（包括小队语音和国战语音），需要打开扬声器才能从网络接收数据并播放

2.函数原型

`GCloudVoiceErr OpenSpeaker();`

3.示例代码

```
public void OpenSpeakerBtn_Click()
{
Debug.Log ("OpenSpeaker btn click");
int ret = m_voiceengine.OpenSpeaker ();
PrintLog ("OpenSpeaker ret=" + ret);
}
```

4.出错处理

```
GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间
```

3.7 关闭扬声器

1.接口说明

在实时语音的模式下，加入房间成功后（包括小队语音和国战语音），当不需要从网络接收数据并播放时，可以调用关闭麦克风接口

2.函数原型

```
GCloudVoiceErr CloseSpeaker();
```

3.示例代码

```
public void CloseSpeakerBtn_Click()
{
Debug.Log ("Close speaker btn click");
m_voiceengine.CloseSpeaker ();
}
```

4.出错处理

```
GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间
```

3.8 加入房间回调

1.接口说明

当加入房间成功或者失败时会通过delegate进行通知

2.函数原型

```
delegate qvoid JoinRoomCompleteHandler(GCloudVoiceCompleteCode code, string roomName, int memberID)
public abstract event JoinRoomCompleteHandler OnJoinRoomComplete;
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见GCloudVoiceCompleteCode定义
roomName	string	加入的房间名
memberID	int	如果加入成功的话，表示加入后的成员ID

3.示例代码

```
m_voiceengine.OnJoinRoomComplete += (IGCloudVoice.GCloudVoiceCompleteCode code, string roo
mName, int memberID) => {
PrintLog ("On Join Room With " + code);
Debug.Log ("OnJoinRoomComplete ret=" + code + " roomName:" + roomName + " memberID:" + m
emberID);
s_logstr += "\r\n"+"OnJoinRoomComplete ret="+code+" roomName:"+roomName+" memberID:"+m
emberID;
//UIManager.m_Instance.OnJoinRoomDone(code);
};
```

3.9 退出房间回调

1.接口说明

当退出房间时，结果通过delegate进行通知

2.函数原型

```
delegate void QuitRoomCompleteHandler(GCloudVoiceCompleteCode code, string roomName, int
memberID)
public abstract event QuitRoomCompleteHandler OnQuitRoomComplete;
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见GCloudVoiceCompleteCode定义
roomName	string	退出的房间名
memberID	int	如果加入成功的话，表示加入后的成员ID

3.示例代码

```
m_voiceengine.OnQuitRoomComplete += (IGCloudVoice.GCloudVoiceCompleteCode code, string roo
mName, int memberID) => {
PrintLog ("On Quit Room With " + code);
Debug.Log ("OnQuitRoomComplete ret=" + code + " roomName:" + roomName + " memberID:" + m
emberID);
s_logstr += "\r\n"+"OnJoinRoomComplete ret="+code+" roomName:"+roomName+" memberID:"+m
emberID;
//UIManager.m_Instance.OnJoinRoomDone(code);
};
```

3.10 成员状态改变回调

1.接口说明

当房间中的其他成员开始说话或者停止说话的时候，通过该回调进行通知

2.函数原型

```
delegate void MemberVoiceHandler(int[] members, int count) ;
public abstract event MemberVoiceHandler OnMemberVoice;
```

参数	类型	意义
members	int[]	改变状态的member成员，其值为[memberID | status]这样的对，总共有count对，status有“0”：停止说话; “1”：开始说话; “2”:继续说话。
count	int	改变状态的成员的数目

3.示例代码

```
m_voiceengine.OnMemberVoice += (int[] members, int count) =>
{
//PrintLog ("OnMemberVoice");
//s_logstr +="\r\ncount:"+count;
for(int i=0; i < count && (i+1) < members.Length; ++i)
{
//s_logstr +="\r\nmemberid:"+members[i]+" state:"+members[i+1];
++i;
}
```

```
//UIManager.m_Instance.UpdateMemberState(members, length, usingCount);
};
```

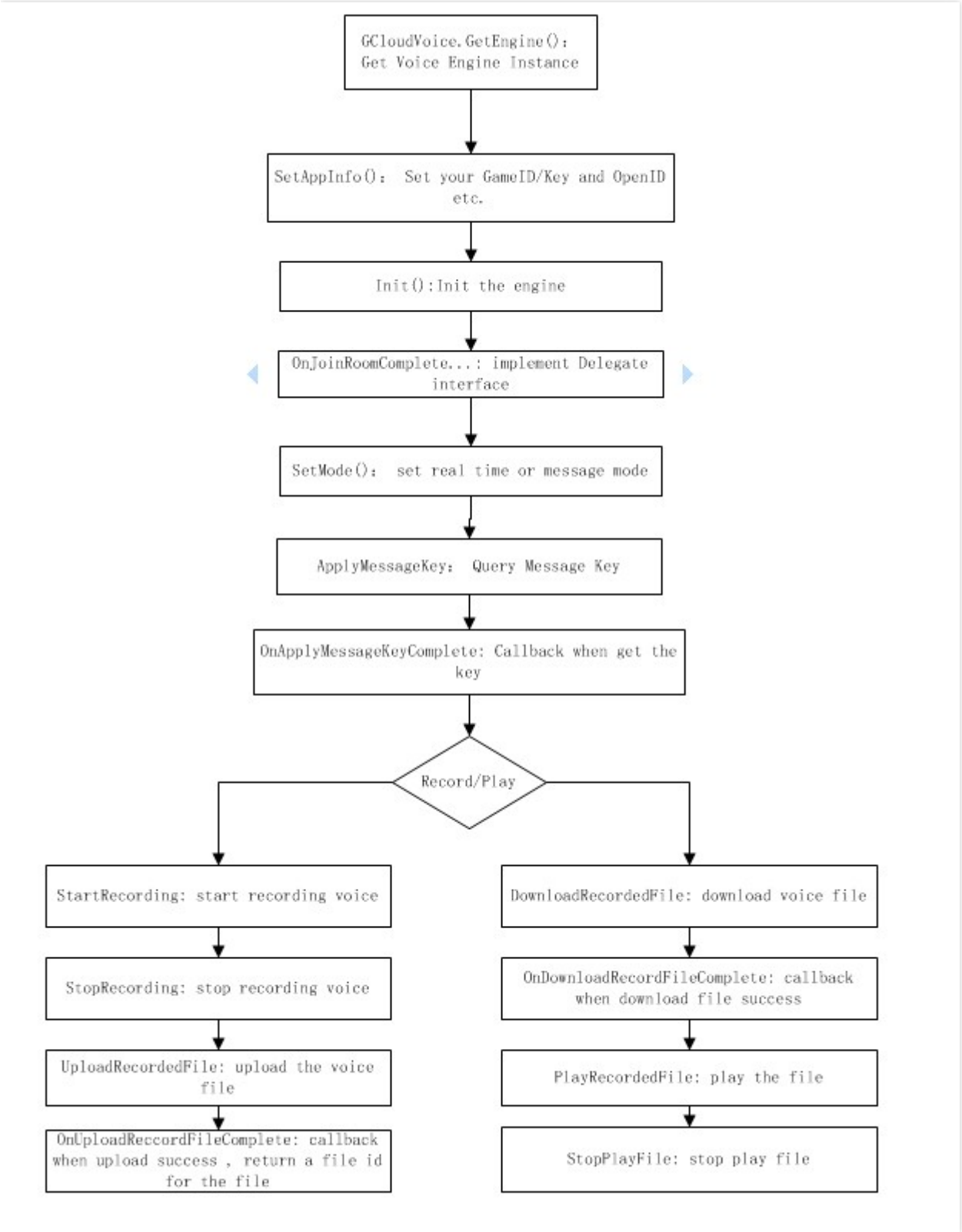
语音消息API

最近更新时间：2018-08-30 16:24:53

1 概述

使用消息语音，需要先调用基本API。

2 语音消息API调用流程



流程说明

1.调用 SetMode 方法设置使用语音消息模式。

2.调用 ApplyMessageKey() 获取语音消息安全密钥key信息，当申请成功后会通过 OnApplyMessageKeyComplete 进行回调。

3.当需要录音时，调用 StartRecording() 录制音频到文件中（文件的路径格式是 /your path ）。
4.如果想取消录制可以调用 StopRecording 接口进行取消。
5.当录制完成后，调用 UploadRecordedFile() 将文件上传到 GcloudVoice 的服务器上，该过程会通过 OnUploadReccordFileComplete() 回调在上传成功的时候返还一个 ShareFileID .该ID是这个文件的唯一标识符，用于其他用户收听时候的下载。服务器需要对其进行管理和转发。
6.当游戏客户端需要收听其他人的录音时，首先从服务器获取转发的 ShareFileID ，然后调用 DownloadRecordedFile 下载该语言文件，下载结果通过 OnDownloadRecordFileComplete 回调来通知。当下载成功时，就可以调用 PlayRecordedFile 播放下载完成的语音数据了。同样的，如果想取消播放，可以调用 StopPlayFile 进行取消。

3 语音消息详细API

3.1 申请语音消息key

1.接口说明

在语音消息的模式下，需要先申请许可才可以正常使用

2.函数原型

```
GCloudVoiceErr ApplyMessageKey(int msTimeout)
```

参数	类型	意义
msTimeout	itn	超时时间，单位毫秒

申请的结果通过delegate void ApplyMessageKeyCompleteHandler(GCloudVoiceCompleteCode code);进行回调

3.示例代码

```
m_voiceengine.OnApplyMessageKeyComplete += (IGCloudVoice.GCloudVoiceCompleteCode code) =
> {
Debug.Log ("OnApplyMessageKeyComplete c# callback");
s_strLog += "\r\n"+"OnApplyMessageKeyComplete ret="+code;
if (code == IGCloudVoice.GCloudVoiceCompleteCode.GV_ON_MESSAGE_KEY_APPLIED_SUCC) {
Debug.Log ("OnApplyMessageKeyComplete succ11");
} else {
Debug.Log ("OnApplyMessageKeyComplete error");
}
```

```
};
```

4.出错处理

```
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，比如超时范围5000ms-60000ms。
GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_AUTHKEY_ERR ： 请求Key的内部错误，此时需要联系GCloud团队，并提供日志进行定位
```

3.2 限制最大语音消息的长度

1.接口说明

在语音消息的模式下，可以限制最大语音消息的长度，目前默认是2min，最大不超过2min。

2.函数原型

```
GCloudVoiceErr SetMaxMessageLength(int msTime)
```

参数	类型	意义
msTimeout	itn	最大语音消息长度，单位毫秒

3.示例代码

```
int ret1 = m_voiceengine.SetMaxMessageLength (60000);
Debug.Log ("SetMaxMessageLength ret==" + ret1);
```

4.出错处理

```
GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，时间范围1000ms-1000260ms。
```

3.3 开始录音

1.接口说明

在语音消息的模式下，开始录音时，需要提供一个录音文件存储的地址路径

2.函数原型

```
GCloudVoiceErr StartRecording(string filePath)
```

参数	类型	意义
filePath	string	录音文件存储的地址路径，路径中需要"/"作分隔，不能用"\"

3.示例代码

```
public void Click_btnStartRecord()
{
Debug.Log("startrecord btn click, recordpath="+m_recordpath);
m_voiceengine.StartRecording (m_recordpath);

}
```

4.出错处理

```
GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，路径为空。
GCLOUD_VOICE_NEED_AUTHKEY ： 需要先调用GetAuthKey申请许可
GCLOUD_VOICE_PATH_ACCESS_ERR ： 提供的路径不合法或者不可写
```

3.4 停止录音

1.接口说明

在语音消息的模式下，调用停止录音接口

2.函数原型

```
GCloudVoiceErr StopRecording()
```

3.示例代码

```
public void Click_btnStopRecord()
{
```

```
Debug.Log("stoprecord btn click");
m_voiceengine.StopRecording ();
}
```

4.出错处理

```
GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式
GCLOUD_VOICE_NEED_AUTHKEY ： 需要先调用GetAuthKey申请许可
```

3.5 上传录音的文件

1.接口说明

录音完成后，通过提供一个录音文件存储的地址路径，将已经录音完的文件进行上传

2.函数原型

```
GCloudVoiceErr UploadRecordedFile(string filePath, int msTimeout)
```

参数	类型	意义
filePath	string	录音文件存储的地址路径，路径中需要"/"作分隔，不能用"\"
msTimeout	int	上传文件超时时间

上传的结果通过delegate void UploadReccordFileCompletehandler(GCloudVoiceCompleteCode code, string filepath, string fileid)进行回调

3.示例代码

```
public void Click_btnUploadFile()
{
int ret1 = m_voiceengine.UploadRecordedFile (m_recordpath, 60000);
Debug.Log ("Click_btnUploadFile file with ret==" + ret1);
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，路径为空。
GCLOUD_VOICE_NEED_AUTHKEY ： 需要先调用GetAuthKey申请许可
GCLOUD_VOICE_PATH_ACCESS_ERR ： 提供的路径不合法或者不可读
GCLOUD_VOICE_HTTP_BUSY ： 还在上一次上传或者下载中，需要等待后再尝试

3.6 下载录音的文件

1.接口说明

录音完成后，通过提供一个录音文件存储的地址路径，将已经录音完的文件进行上传

2.函数原型

```
GCloudVoiceErr DownloadRecordedFile(string fileID, string downloadFilePath, int msTimeout);
```

参数	类型	意义
fileID	string	要下载文件的文件ID
downloadFilePath	string	下载录音文件存储的地址路径，路径中需要"/"作分隔，不能用"\"
msTimeout	int	下载文件超时时间

下载的结果通过delegate void DownloadRecordFileCompletehandler(GCloudVoiceCompleteCode code, string filepath, string fileid)进行回调

3.示例代码

```
public void Click_btnDownloadFile()
{
int ret = m_voiceengine.DownloadRecordedFile (m_fileid, m_downloadpath, 60000);
s_strLog += "\r\n download file with ret=="+ret+" fileid="+m_fileid+" downpath"+m_downloadpath;
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式

GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，路径为空。
GCLOUD_VOICE_NEED_AUTHKEY ： 需要先调用GetAuthKey申请许可
GCLOUD_VOICE_PATH_ACCESS_ERR ： 提供的路径不合法或者不可写或者不可读
GCLOUD_VOICE_HTTP_BUSY ： 还在上一次上传或者下载中，需要等待后再尝试

3.7 开始播放下载的音频

1.接口说明

下载下来的音频文件，需要调用相关接口进行播放

2.函数原型

```
GCloudVoiceErr PlayRecordedFile (string downloadFilePath)
```

参数	类型	意义
filePath	string	下载文件存储的地址路径，路径中需要"/"作分隔，不能用"\"

如果正常播放完，会回调delegate void PlayRecordFilCompletehandler(GCloudVoiceCompleteCode code, string filepath)

3.示例代码

```
public void Click_btnPlayReocrdFile()
{

int err;
if (m_ShareFileID == null) {
err = m_voiceengine.PlayRecordedFile(m_recordpath);
PrintLog ("downloadpath is nill, play local record file with ret=" + err);
return;
}
err = m_voiceengine.PlayRecordedFile(m_downloadpath);
PrintLog ("playrecord file with ret=" + err);

//m_voiceengine.PlayRecordedFile (m_downloadpath);
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，路径为空。
GCLOUD_VOICE_PATH_ACCESS_ERR ： 提供的路径不合法或者不可写
GCLOUD_VOICE_SPEAKER_ERR : 打开麦克风失败

3.8 停止播放下载的音频

1.接口说明

中断播放动作

2.函数原型

`GCloudVoiceErr StopPlayFile()`

3.示例代码

```
public void Click_btnStopPlayRecordFile()
{
m_voiceengine.StopPlayFile ();
}
```

4.出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式

3.9 请求语音消息Key回调

1.接口说明

请求语音消息许可的时候会回调

2.函数原型

```
delegate void ApplyMessageKeyCompleteHandler(GCloudVoiceCompleteCode code)
public abstract event ApplyMessageKeyCompleteHandler OnApplyMessageKeyComplete
```

参数	类型	意义

参数	类型	意义
code	GCloudVoiceCompleteCode	参见GCloudVoiceCompleteCode定义

3.示例代码

```
m_voiceengine.OnApplyMessageKeyComplete += (IGCloudVoice.GCloudVoiceCompleteCode code) =
> {
Debug.Log ("OnApplyMessageKeyComplete c# callback");
s_strLog += "\r\n"+"OnApplyMessageKeyComplete ret="+code;
if (code == IGCloudVoice.GCloudVoiceCompleteCode.GV_ON_MESSAGE_KEY_APPLIED_SUCC) {
Debug.Log ("OnApplyMessageKeyComplete succ11");
} else {
Debug.Log ("OnApplyMessageKeyComplete error");
}
};
```

3.10 上传完成回调

1.接口说明

上传语音文件后的结果通过这个进行回调

2.函数原型

```
delegate void UploadReccordFileCompletehandler(GCloudVoiceCompleteCode code, string filepat
h, string fileid)
public abstract event UploadReccordFileCompletehandler OnUploadReccordFileComplete
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见GCloudVoiceCompleteCode定义
filepath	string	上传的文件路径
fileid	string	文件的id

3.示例代码

```
m_voiceengine.OnUploadReccordFileComplete += (IGCloudVoice.GCloudVoiceCompleteCode code, s
tring filepath, string fileid) => {
```

```
Debug.Log ("OnUploadReccordFileComplete c# callback");
s_strLog += "\r\n"+" fileid len="+fileid.Length+"OnUploadReccordFileComplete ret="+code+" filepath:"+filepath+" fielid:"+fileid;
if (code == IGCloudVoice.GCloudVoiceCompleteCode.GV_ON_UPLOAD_RECORD_DONE) {
m_fileid = fileid;
s_strLog+="OnUploadReccordFileComplete succ, filepath:" +" fileid len="+fileid.Length+ filepath + " fileid:" + fileid+" fileid len="+fileid.Length;
Debug.Log ("OnUploadReccordFileComplete succ, filepath:" + filepath +" fileid len="+fileid.Length+ " fileid:" + fileid+" fileid len="+fileid.Length);
} else {
s_strLog+="OnUploadReccordFileComplete err, filepath:" + filepath + " fileid:" + fileid;
Debug.Log ("OnUploadReccordFileComplete error");
}
};
```

3.11 下载完成回调

1.接口说明

下载语音文件后的结果通过这个进行回调

2.函数原型

```
delegate void DownloadRecordFileCompletehandler(GCloudVoiceCompleteCode code, string filepath, string fileid)
public abstract event DownloadRecordFileCompletehandler OnDownloadRecordFileComplete
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见GCloudVoiceCompleteCode定义
filepath	string	下载的路径
fileid	string	文件的id

3.示例代码

```
m_voiceengine.OnDownloadRecordFileComplete += (IGCloudVoice.GCloudVoiceCompleteCode code, string filepath, string fileid) => {
Debug.Log ("OnDownloadRecordFileComplete c# callback");
s_strLog += "\r\n"+"OnDownloadRecordFileComplete ret="+code+" filepath:"+filepath+" fielid:"+fileid;
```

```
if (code == IGCloudVoice.GCloudVoiceCompleteCode.GV_ON_DOWNLOAD_RECORD_DONE) {
Debug.Log ("OnDownloadRecordFileComplete succ, filepath:" + filepath + " fileid:" + fileid);
} else {
Debug.Log ("OnDownloadRecordFileComplete error");
}
};
```

3.12 正常播放完成后回调

1.接口说明

如果用户没有暂停播放，语音文件已经播放完了，通过这个进行回调

2.函数原型

```
delegate void PlayRecordFilCompletehandler(GCloudVoiceCompleteCode code, string filepath)
public abstract event PlayRecordFilCompletehandler OnPlayRecordFilComplete;
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见GCloudVoiceCompleteCode定义
filepath	string	播放的文件路径

3.示例代码

```
m_voiceengine.OnPlayRecordFilComplete += (IGCloudVoice.GCloudVoiceCompleteCode code, string
filepath) => {
Debug.Log ("OnPlayRecordFilComplete c# callback");
s_strLog += "\r\n"+"OnPlayRecordFilComplete ret="+code+" filepath:"+filepath;
if (code == IGCloudVoice.GCloudVoiceCompleteCode.GV_ON_PLAYFILE_DONE) {
Debug.Log ("OnPlayRecordFilComplete succ, filepath:" + filepath);
} else {
Debug.Log ("OnPlayRecordFilComplete error");
}
};
```

错误码表

最近更新时间：2017-03-09 08:29:17

1 概述

本文档介绍了GVoice游戏语音C++接口SDK的错误码。

2 错误码表

错误	十六进制值	十进制值	意义
GCLOUD_VOICE_SUCC	0x0	0	Success
GCLOUD_VOICE_PARAM_NULL	0x1001	4097	some param is null
GCLOUD_VOICE_NEED_SETAPPINFO	0x1002	4098	you should call SetAppInfo first before call other api
GCLOUD_VOICE_INIT_ERR	0x1003	4099	Init Erro
GCLOUD_VOICE_RECORDING_ERR	0x1004	4100	now is recording, can't do other operator
GCLOUD_VOICE_POLL_BUFF_ERR	0x1005	4101	poll buffer is not enough or null
GCLOUD_VOICE_MODE_STATE_ERR	0x1006	4102	call some api, but the mode is not correct, maybe you shoud call SetMode first and correct
GCLOUD_VOICE_PARAM_INVALID	0x1007	4103	some param is null or value is invalid for our request, used right param and make sure is value range is correct by our comment
GCLOUD_VOICE_OPENFILE_ERR	0x1008	4104	open a file err

错误	十六进制值	十进制值	意义
GCLOUD_VOICE_NEED_INIT	0x1009	4105	you should call Init before do this operator
GCLOUD_VOICE_ENGINE_ERR	0x100A	4106	you have not get engine instance, this common in use c# api. but not get gcloudvoice instance first
GCLOUD_VOICE_POLL_MSG_PARSE_ERR	0x100B	4107	this common in c# api, parse poll msg err
GCLOUD_VOICE_POLL_MSG_NO	0x100C	4108	poll no msg to update
GCLOUD_VOICE_REALTIME_STATE_ERR	0x2001	8193	call some realtime api, but state err. such as OpenMic but you have not Join Room first
GCLOUD_VOICE_JOIN_ERR	0x2002	8194	join room failed
GCLOUD_VOICE_QUIT_ROOMNAME_ERR	0x2003	8195	quit room err, the quit roomname not equal join roomname
GCLOUD_VOICE_OPENMIC_NOTANCHOR_ERR	0x2004	8196	open mic in bigroom,but not anchor role
GCLOUD_VOICE_AUTHKEY_ERR	0x3001	12289	apply authkey api error
GCLOUD_VOICE_PATH_ACCESS_ERR	0x3002	12290	the path can not access ,may be path file not exists or deny to access
GCLOUD_VOICE_PERMISSION_MIC_ERR	0x3003	12291	you have not right to access micphone in android
GCLOUD_VOICE_NEED_AUTHKEY	0x3004	12292	you have not get authkey, call ApplyMessageKey first
GCLOUD_VOICE_UPLOAD_ERR	0x3005	12293	upload file err

错误	十六进制值	十进制值	意义
GCLOUD_VOICE_HTTP_BUSY	0x3006	12294	http is busy,maybe the last upload/download not finish.
GCLOUD_VOICE_DOWNLOAD_ERR	0x3007	12295	download file err
GCLOUD_VOICE_SPEAKER_ERR	0x3008	12296	open or close speaker tve error
GCLOUD_VOICE_TVE_PLAYSOUND_ERR	0x3009	12297	tve play file error
GCLOUD_VOICE_INTERNAL_TVE_ERR	0x5001	20481	internal TVE err, our used
GCLOUD_VOICE_INTERNAL_VISIT_ERR	0x5002	20482	internal Not TVE err, out used
GCLOUD_VOICE_INTERNAL_USED	0x5003	20483	internal used, you should not get this err num

JAVA
Android平台接入流程

最近更新时间：2018-08-29 10:15:14

1 使用简述

GcloudVoice 客户端 SDK 可以运行在 iOS 和 Android 两个平台并支持 Cocos/Unreal/Unity 等主流游戏引擎。其在Cocos 和 Unreal 上提供 C++ 接口。本文档描述了 JAVA 接口使用方法（Android 也可以使用 C++ 接口）。

GcloudVoice 客户端 SDK 目前主要有实时语音(Real-Time)、离线语音(Message)两大功能。实时语音主要应用于游戏对局中的实时交流，如 MOBA 类游戏的实时指挥（王者荣耀、全民超神）、FPS类游戏的指挥（穿越火线-枪战王者）用于弥补手机上输入困难不实时的问题。离线语音主要用于留言消息场景，如大厅的全局聊天（三国之刃）、语音留言（邮件通知）。

同时实时语音还分为国战语音（比如 MMORGP 里面的国战，御龙国战）以及小队语音（MOBA 类里面的对局，王者荣耀的 5V5）,如果一个房间里面成员小于 20 人，建议使用小队语音，否则建议使用国战语音，在国战语音中，同时说话人数不得超过 5 人。

GcloudVoice 客户端 SDK 接口主要分成三个部分：基本 API、实时语音 API 以及离线语音 API。

1.1 系统配置和基本使用

Android SDK 下载
Android Demo 下载

下载 Android SDK 包后，导入 Jar 包和 SO 文件到 android 工程后。按如下流程即可接入。 主要是导入包

```
import com.tencent.gcloud.voice.GCloudVoiceEngine;
```

然后通过

```
GCloudVoiceEngine.getInstance().init(getApplicationContext(), this);
```

初始化 java. 再完成初始化以及对应的模式设置

```
engine = GCloudVoiceEngine.getInstance();
engine.SetAppInfo(appID, appKey, openID);
engine.Init();
engine.SetMode(0);
```

实现和设置回调： 导入回调接口：

```
import com.tencent.gcloud.voice.IGCloudVoiceNotify;
```

实现回调类：

```
 class Notify implements IGCloudVoiceNotify
```

设置回调：

```
Notify notify = new Notify();
engine.SetNotify(notify);
```

加入房间：

```
int ret = engine.JoinTeamRoom(_roomName, 10000);
```

在回调中查看进房的回调结果：

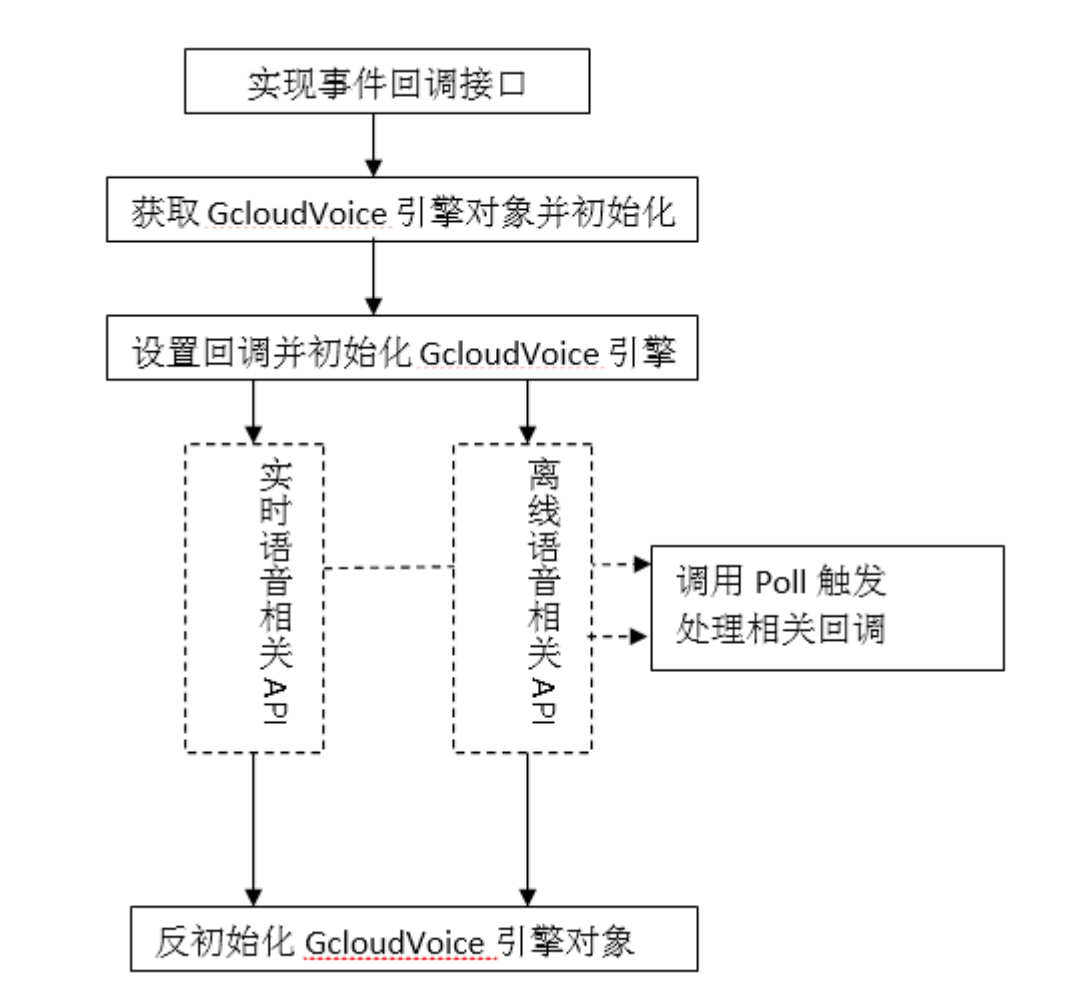
```
class Notify implements IGCloudVoiceNotify {
    /*
    String TAG = "[cz]";
    @Override
    public void onJoinRoomComplete(int code) {
        // TODO Auto-generated method stub
        Log.i(TAG, "onJoinRoomComplete:" + code);
        _logTV.setText( "onJoinRoomComplete with "+ code);
    }*/
```

成功后则可以 OpenMic OpenSpeaker：

```
int ret = engine.OpenMic();
int ret = engine.OpenSpeaker();
```

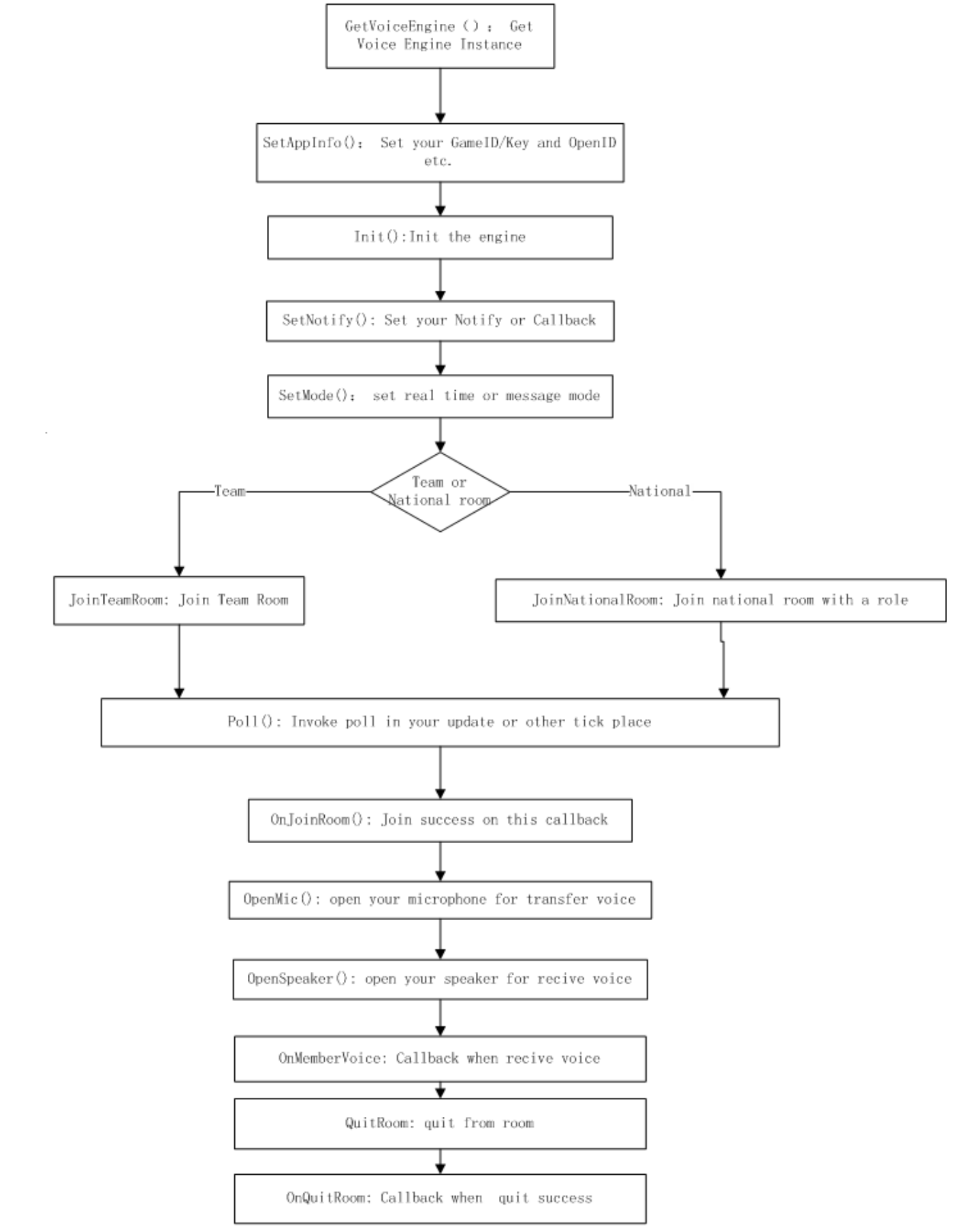
具体流程接口可以照以下文档的详细说明。

1.2 接口调用流程



首先实现 IGCloudVoiceNotify 回调，然后调用 GetVoiceEngine 获取 IGCloudVoiceNotify 对象。然后对该对象进行初始化操作并设置回调。接着再根据需要调用实时语音接口或者离线语音接口，然后在系统可以 Tick 的地方（如 Unity3D 的 Update）调用 Poll 驱动处理事务中的回调信息。

1.3 实时语音接口调用流程


GetVoiceEngine（）： Get Voice Engine Instance
SetAppInfo(): Set your GameID/Key and OpenID etc.
Init():Init the engine
SetNotify(): Set your Notify or Callback
SetMode(): set real time or message mode
Team or National room
Team
National
JoinTeamRoom: Join Team Room
JoinNationalRoom: Join national room with a role
Poll(): Invoke poll in your update or other tick place
OnJoinRoom(): Join success on this callback
OpenMic(): open your microphone for transfer voice
OpenSpeaker(): open your speaker for recive voice
OnMemberVoice: Callback when recive voice
QuitRoom: quit from room
OnQuitRoom: Callback when quit success

在使用实时语音的时候，首先需要调用 SetMode 方法设置使用实时语音模式。然后根据业务逻辑判断是需要加入小队语音房间或者国战语音房间，分别调用 JoinTeamRoom 和 JoinNationalRoom。

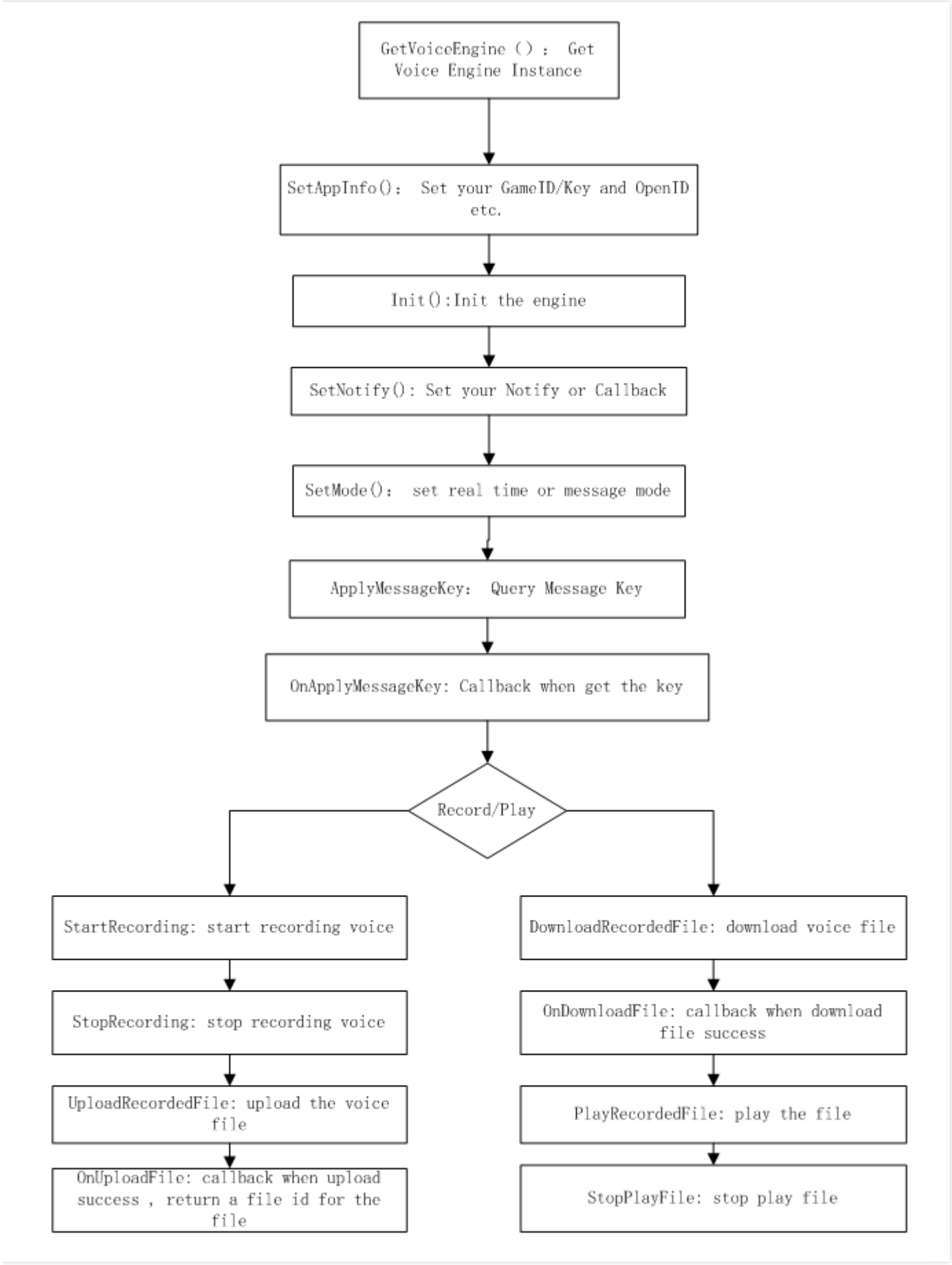
然后需要 tick 的调用 poll 来检查回调，当加入房间成功或者失败时，会回调 OnJoinRoom 方法。

加入房间成功后就可以调用 OpenMic 打开麦克风进行采集并发送到网络，调用 OpenSpeaker 打开扬声器开始接受网络上的音频流并自动进行播放。

当需要退出房间时，调用 QuitRoom 即可，调用成功后会回调 OnQuitRoom 方法。

注意：
对于国战语音，系统要求说话人数不能超过 5 个人，每个用户多了一个角色信息，在加入房间的时候需要指定是以听众的身份加入还是以主播的身份加入。

1.4 离线语音接口调用流程


GetVoiceEngine () : Get Voice Engine Instance
SetAppInfo(): Set your GameID/Key and OpenID etc.
Init():Init the engine
SetNotify(): Set your Notify or Callback
SetMode(): set real time or message mode
ApplyMessageKey: Query Message Key
OnApplyMessageKey: Callback when get the key
Record/Play
StartRecording: start recording voice
StopRecording: stop recording voice
UploadRecordedFile: upload the voice file
OnUploadFile: callback when upload success , return a file id for the file
DownloadRecordedFile: download voice file
OnDownloadFile: callback when download file success
PlayRecordedFile: play the file
StopPlayFile: stop play file

在使用语音消息的时候，首先需要调用 SetMode 方法设置使用语音消息模式。然后调用 ApplyMessageKey 申请用于离线语音的相关的 key 信息，当申请成功后会通过 OnApplyMessageKey 进行回调。

当需要录音时，调用 StartRecording 接口将音频文件录制到文件中（文件的路径格式是file://your path）。如果想取消录制可以调用 StopRecording 接口进行取消。当录制完成后，调用 UploadRecordedFile 将文件上传到 GcloudVoice 的服务器上，该过程会通过 OnUploadFile 回调在上传成功的时候返还一个 ShareFileID.该 ID 是这个文件的唯一标识符，用于其他用户收听时候的下载。服务器需要对其进行管理和转发

当游戏客户端需要收听其他人的录音时，首先从服务器获取转发的 ShareFileID，然后调用 DownloadRecordedFile 下载该语言文件，下载结果通过 OnDownloadFile 回调来通知。当下载成功时，就可以调用 PlayRecordedFile 播放下载完成的语音数据了。同样的，如果想取消播放，可以调用 StopPlayFile 进行取消。

2 接口说明

2.1 基本 API

2.1.1 获取 IGcloudVoiceEngine 对象

接口：IGCloudVoiceEngine *GetVoiceEngine()
参数：无
返回值：IGcloudVoiceEngine对象,出错时返回NULL

2.1.2 设置应用信息

接口：GCloudVoiceErrno SetAppInfo(const char appID,const char appKey, const char *openID)
参数： appID : 应用的GameID appKey: 应用的GameKey openID: 游戏从QQ、微信等获得到的openID，也可以是游戏侧能标示的唯一玩家的角色ID
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.1.3 设置应用信息初始化

接口：GcloudVoiceErrno Init()
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.1.4 反初始化

接口： GcloudVoiceErrno Deinit()
参数：无
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.1.5 反初始化

接口： GCloudVoiceErrno SetNotify(IGCloudVoiceNotify *notify)
参数：notify: 实现的IGCloudVoiceNotify对象
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.1.6 设置通话模式

接口：GcloudVoiceErrno SetMode(GcloudVoiceMode mode)
参数：mode： 通话模式 enum GcloudVoiceMode

```
{
RealTime, // 实时语音
Messages, // 离线语音
};
```

返回值：成功时返回 GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.1.7 系统 Pause 中断

接口：GcloudVoiceErrno Pause()
参数：无
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.1.8 中断恢复Resume

接口：GcloudVoiceErrno Resume()
参数： 无
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.1.9 回调驱动Poll

接口：GcloudVoiceErrno Poll()
参数：无
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.2 实时语音 API

2.2.1 加入小队房间

接口：GcloudVoiceErrno JoinTeamRoom(const char *roomName, int msTimeout)
参数： roomName: 服务器获得的房间的名称，如果不存在服务器会自动创建，长度限制在511以下。 msTimeout: 加入房间的超时时间
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码。这里返回成功不一定标示加入房间已经成功，加入房间是否成功会在OnJoinRoom中回调

2.2.2 国战房间

接口：GcloudVoiceErrno JoinNationalRoom(const char *roomName, GCloudVoiceMemberRole role, int msTimeout)
参数： roomName: 服务器获得的房间的名称，如果不存在服务器会自动创建 role: 用户的角色，是听众还是主播 msTimeout: 加入房间的超时时间
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码。这里返回成功不一定标示加入房间已经成功，加入房间是否成功会在OnJoinRoom中回调

2.2.3 退出房间

接口：GcloudVoiceErrno QuitRoom()
参数：无
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码。当前加入的是那种房间（小队或者国战）就退出哪个房间

2.2.4 打开麦克风开始录音

接口：GcloudVoiceErrno OpenMic()
参数：无
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.2.5 关闭麦克风，停止录音

接口：GcloudVoiceErrno CloseMic()
参数：无
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.2.6 打开扬声器开始播放声音

接口：GcloudVoiceErrno OpenSpeaker()
参数：无
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.2.7 关闭扬声器停止播放声音

接口：GcloudVoiceErrno CloseSpeaker()
参数：无
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.3 离线语音 API

2.3.1 申请 AuthKey

接口：GCloudVoiceErrno ApplyMessageKey(int msTimeout)
参数：msTimeout超时时间

返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码。这里成功不代表已经成功申请了香港key，需要在OnApplyMessageKey中回调

2.3.2 开始录音

接口：GcloudVoiceErrno StartRecording (const char * filePath)
参数：filePath:保存文件的位置，格式为 保存至文件"you_dir/your_filename" ：文件路径需要用反斜杠"/"做分隔
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.3.3 取消录音

接口：GcloudVoiceErrno StopRecording ()
参数：无
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.3.4 发送音频文件

接口：GcloudVoiceErrno UploadRecordedFile (const char * filePath)
参数：filePath：为录制时给的路径
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.3.5 下载音频文件

接口：GcloudVoiceErrno DownloadRecordedFile (const char fileID, const char downloadFilePath, int msTimeout)
参数： fileID： 要下载的文件的ID downloadFilePath: 下载的位置，格式如上 msTimeout: 超时时间
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.3.6 播放下载的音频文件

GcloudVoiceErrno PlayRecordedFile (const char * downloadFilePath)
参数：downloadFilePath： 下载的文件的位置
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.3.7 取消播放

接口：GcloudVoiceErrno StopPlayFile ()
参数：无
返回值：成功时返回GcloudVoiceErrno::GCLOUD_VOICE_SUCC,否则返回其他错误码

2.4 回调接口

2.4.1 加入房间结果

接口：void OnJoinRoom(GCloudVoiceCompleteCode code, char *roomName, int memberID)
参数：code: 加入房间的结果 enum GcloudVoiceCompleteCode
{
GV_ON_JOINROOM_SUCC, // 加入房间成功
GV_ON_JOINROOM_TIMEOUT, // 加入房间超时
GV_ON_JOINROOM_FAIL, // 加入房间其他错误
};
roomName: 加入房间的名字 memberID： 成员的ID
返回值：无

2.4.2 退出房间结果

接口：void OnQuitRoom(GCloudVoiceCompleteCode code, const char *roomName)
参数：code: 加入房间的结果 enum GcloudVoiceCompleteCode

{
GV_ON_JOINROOM_SUCC, // 加入房间成功
GV_ON_JOINROOM_TIMEOUT, // 加入房间超时
GV_ON_JOINROOM_FAIL, // 加入房间其他错误
};
roomName: 加入房间的名字
返回值：无

2.4.3 其他成员发声状态的变化

接口：void OnMemberVoice (const int members, int length)
参数：members: 成员及状态，格式为 memberID,status,memberID,status 成对出现，status值为 0：从发声变成没有发声 1： 从不发生变成发声 2： 从发声再发声 length: member的个数，2就是member数组的长度。
返回值：无

2.4.4 发送文件状态回调

接口：void OnUploadFile(GCloudVoiceCompleteCode code, const char filePath, const char fileID)
参数： filePath： 文件存储的位置，与send的时候一致 fileID： 文件唯一标示的ID code: 如果出错时候的错误码
返回值：无

2.4.5 下载文件状态回调

接口：void OnDownloadFile(GCloudVoiceCompleteCode code, const char filePath, const char fileID)
参数： filePath： 文件存储的位置，与down的时候一致 fileID： 文件唯一标示的ID code: 如果出错时候的错误码
返回值：无

2.4.6 申请Key的回调

接口：void OnApplyMessageKey(GCloudVoiceCompleteCode code)

参数：code: 如果出错时候的错误码

返回值：无

2.4.7 文件播放结束后的回调

接口：void OnPlayRecordedFile(GCloudVoiceCompleteCode code,const char *filePath)

参数： code: 如果出错时候的错误码 filePath: 播放的文件的位置

返回值：无

iOS
iOS 接入流程

最近更新时间：2018-09-04 15:26:12

使用简述

本文档介绍了 GVoice 游戏语音 Objective-C 接口 SDK 的接入方法，适用于 iOS 平台直接开发的游戏或者 App。

GVoice 游戏语音目前提供了实时语音（Real-Time）、语音消息（Message）两大功能。实时语音能够让多名玩家进行实时语音聊天。语音消息为用户提供快速录制并向其他玩家发送一段语音消息的能力。

实时语音分为小队语音、国战语音两种模式。如果一个房间内同时聊天的玩家小于 20 人，则使用小队语音模式，这时玩家可以自由发言（该模式常用于队友间沟通）。当一个房间内语音聊天的玩家大于 20 人时，则使用国战语音模式，在国战语音中，允许开发者控制发言权限，同时说话人数不得超过 5 人（该模式常用于团战指挥、主播）。

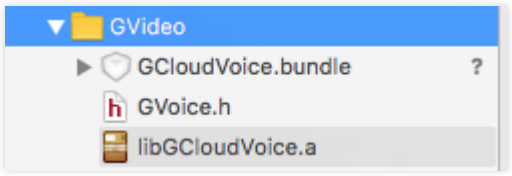
GVoice 客户端 SDK 接口主要分成三个部分：基本 API、实时语音 API 以及语音消息 API。

系统配置和基本使用

iOS SDK 下载
iOS Demo 下载

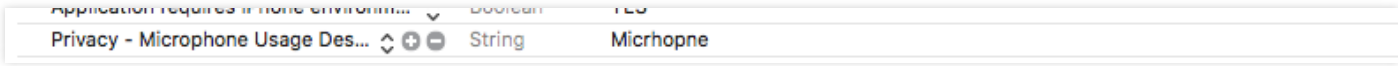
下载iOS SDK包后，解压得到 .a 文件、 .h 文件以及 bundle 文件。按如下流程即可接入：

1. 导入库文件和 Bundle 文件到 Xcode 工程中。



并导入以下库文件：

在iOS 10 以后还需要在 Info 中配置麦克风权限。



2. 在合适的地方进行包含头文件 GVoice.h 并进行初始化
   GVoice 通过单例方法 [GVGCloudVoice sharedInstance] 来获得示例对象并进行相关的操作。首先要设置应用的信息，然后初始化引擎：

```
[[GVGCloudVoice sharedInstance] setAppInfo:"your_appid" withKey:"your_appkey" andOpenID:"your_open_id"];
[[GVGCloudVoice sharedInstance] initEngine];
```

3. 在使用语音前先进行语音模式以及回调 delegate 的实现等操作。

```
[[GVGCloudVoice sharedInstance] setMode:RealTime];

@interface TeamRoomViewController : UIViewController
@end

#pragma mark delegate

- (void) onJoinRoom:(enum GCloudVoiceCompleteCode) code withRoomName: (const char * _Nullable)roomName andMemberID:(int) memberID {
}

- (void) onStatusUpdate:(enum GCloudVoiceCompleteCode) status withRoomName: (const char * _Nullable)roomName andMemberID:(int) memberID {

}
```

```
- (void) onQuitRoom:(enum GCloudVoiceCompleteCode) code withRoomName: (const char * _Nullable)roomName {
}

- (void) onMemberVoice: (const unsigned int * _Nullable)members withCount: (int) count {
}

- (void) onUploadFile: (enum GCloudVoiceCompleteCode) code withFilePath: (const char * _Nullable)filePath andFileID:(const char * _Nullable)fileID {

}

- (void) onDownloadFile: (enum GCloudVoiceCompleteCode) code withFilePath: (const char * _Nullable)filePath andFileID:(const char * _Nullable)fileID {

}

- (void) onPlayRecordedFile:(enum GCloudVoiceCompleteCode) code withFilePath: (const char * _Nullable)filePath {

}

- (void) onApplyMessageKey:(enum GCloudVoiceCompleteCode) code {

}

- (void) onSpeechToText:(enum GCloudVoiceCompleteCode) code withFileID:(const char * _Nullable)fileID andResult:( const char * _Nullable)result {

}

- (void) onRecording:(const unsigned char* _Nullable) pAudioData withLength: (unsigned int) nDataLength {

}
```

4. 驱动回调。需要您定时调用 Poll 函数才会触发回调。例如在 UI 线程里面通过定时器来调用：

```
_pollTimer = [NSTimer scheduledTimerWithTimeInterval:1.000/15 repeats:YES block:^(NSTimer * _Nonnull timer) {
[[GVGCloudVoice sharedInstance] poll];
}];
```

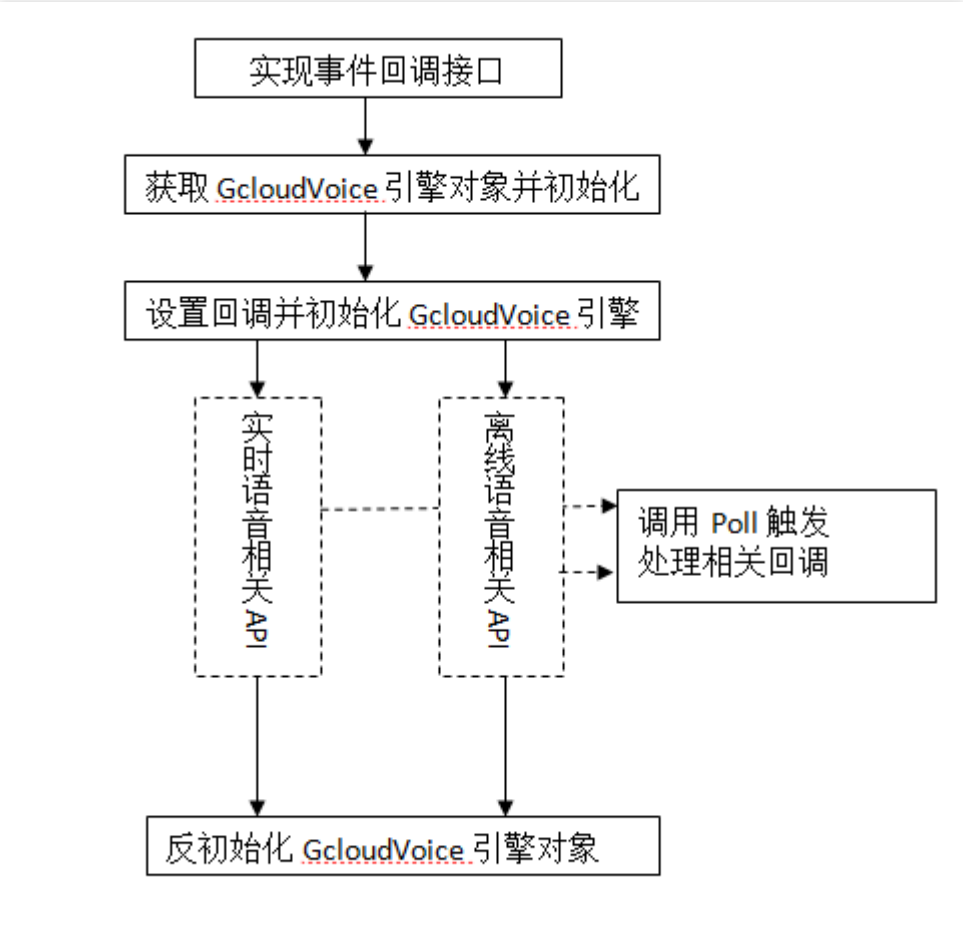
5. 加入房间（请注意在进房回调中才能确定是否进房成功）。

```
[[GVGCloudVoice sharedInstance] joinTeamRoom:[_roomID cStringUsingEncoding:NSUTF8StringEncoding] timeout:18000];
```

6. 如果回调中成功进房了，就可以打开麦克风和扬声器，进行通话了。

```
[[GVGCloudVoice sharedInstance] openSpeaker];
```

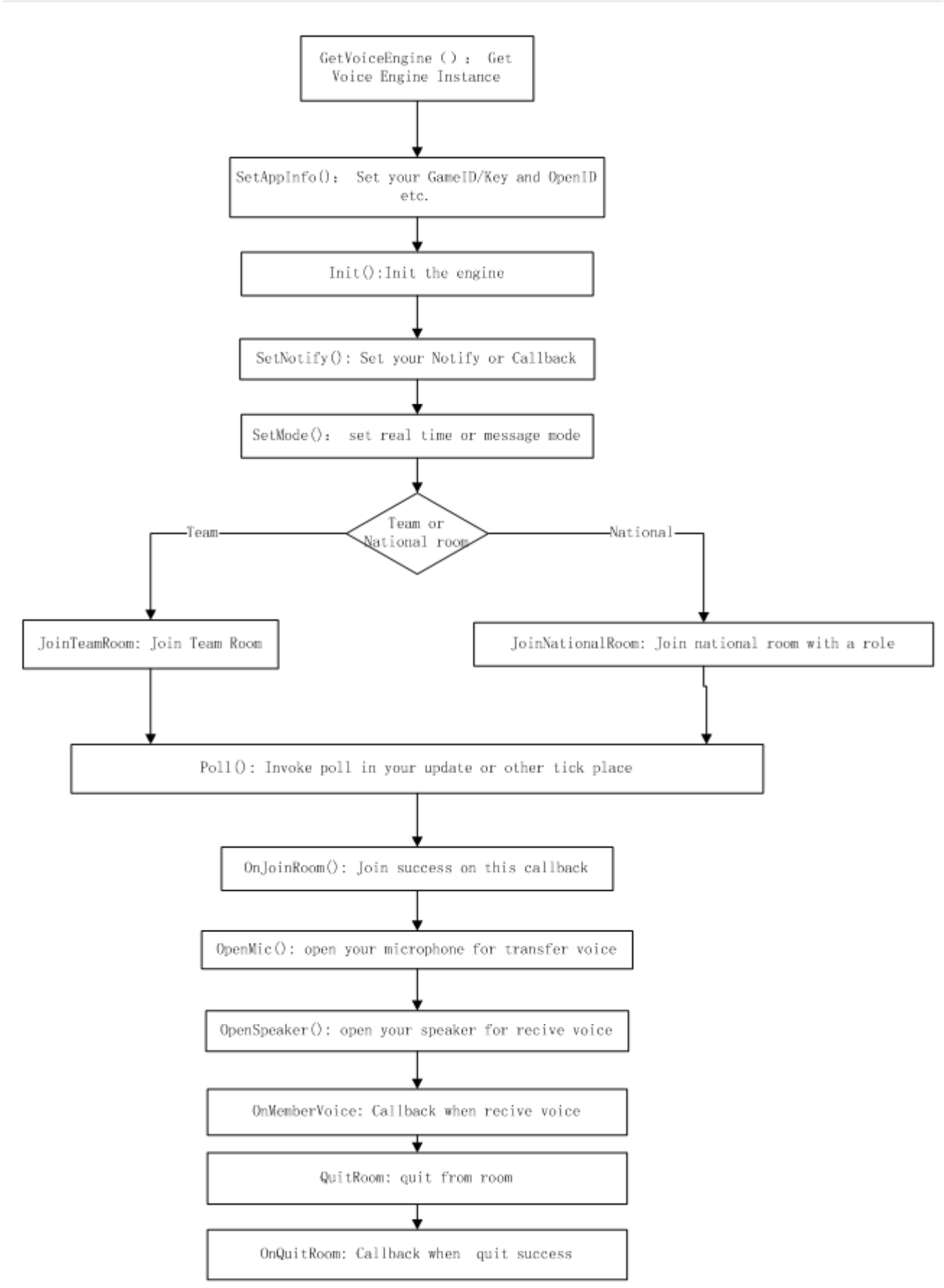
接口调用流程



- 实现 GVGCloudVoiceDelegate 回调类。
- 调用 GVGCloudVoice 的 [GVGCloudVoice sharedInstance] 方法获取 GVGCloudVoice 对象。
- 对该对象进行初始化操作并设置回调。

- 根据需要调用实时语音接口或语音消息接口。
- 在系统可以 Tick 的地方（如 Unity3D 的 Update）调用 poll() 函数驱动程序运行。

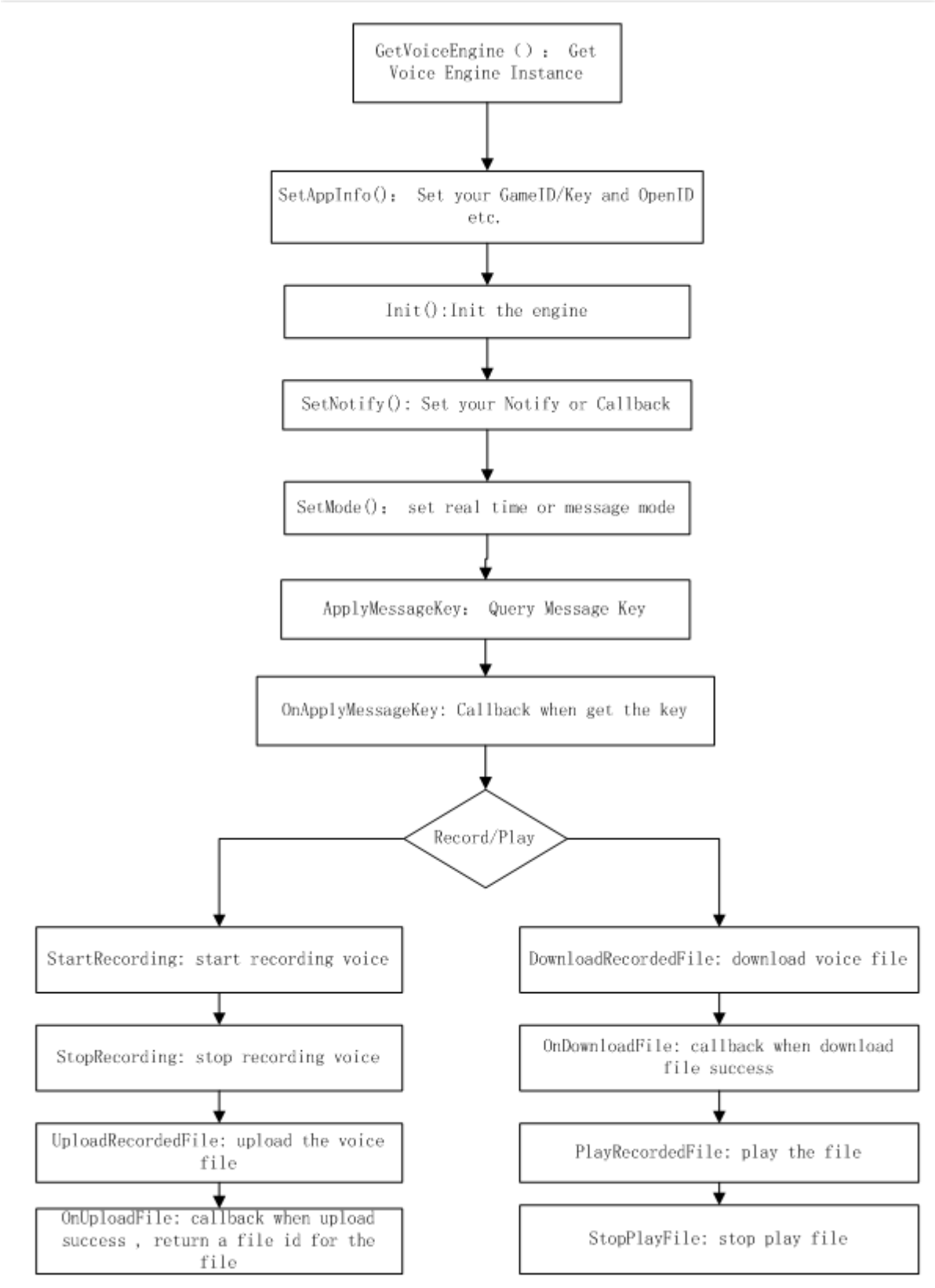
实时语音接口调用流程


GetVoiceEngine ()： Get Voice Engine Instance
SetAppInfo()： Set your GameID/Key and OpenID etc.
Init():Init the engine
SetNotify(): Set your Notify or Callback
SetMode()： set real time or message mode
Team or National room
Team
National
JoinTeamRoom: Join Team Room
JoinNationalRoom: Join national room with a role
Poll(): Invoke poll in your update or other tick place
OnJoinRoom(): Join success on this callback
OpenMic(): open your microphone for transfer voice
OpenSpeaker(): open your speaker for recive voice
OnMemberVoice: Callback when recive voice
QuitRoom: quit from room
OnQuitRoom: Callback when quit success

- 调用 setMode() 方法设置使用实时语音模式。
- 根据业务需求使用小队语音或国战语音，分别调用 joinTeamRoom() 或 joinNationalRoom()。
- 需要 Tick 的调用 poll 来检查回调，当加入房间成功或者失败时，会回调 onJoinRoomComplete() 方法。
- 加入房间成功后，就可以调用 openMic() 打开麦克风进行采集并发送到网络。
- 调用 openSpeaker() 打开扬声器，开始接受网络上的音频流并自动进行播放。
- 当需要退出房间时，调用 quitRoom() 即可，成功后会回调 onQuitRoomComplete() 方法。

注意：
对于国战语音，系统要求说话人数不能超过 5 个人，每个用户多了一个角色信息，在加入房间的时候需要指定是以听众的身份加入还是以主播的身份加入。

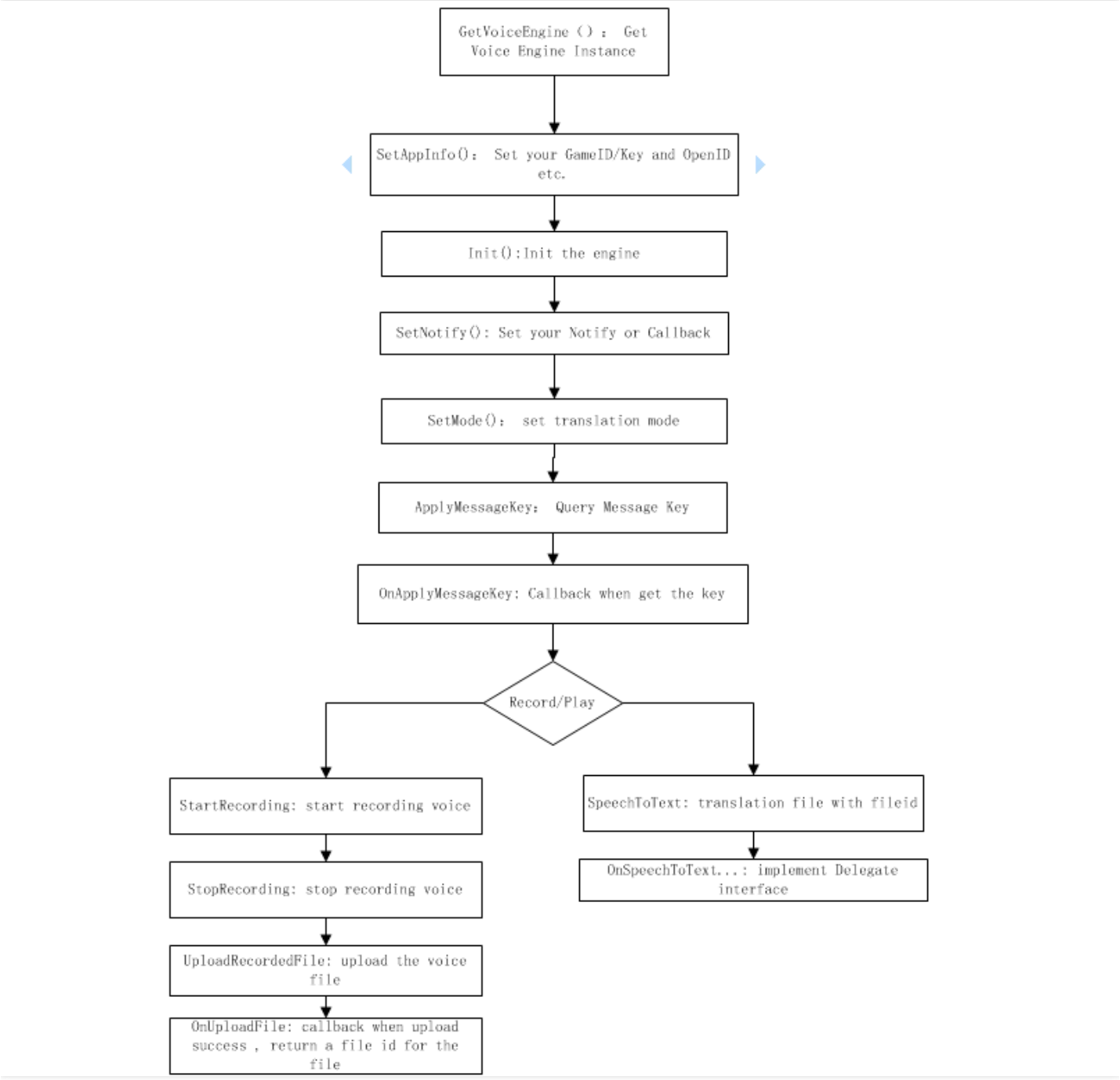
离线语音接口调用流程



- 调用 setMode 方法设置使用语音消息模式。

- 调用 applyMessageKey() 获取语音消息安全密钥 key 信息，当申请成功后会通过 onApplyMessageKeyComplete 进行回调。
- 当需要录音时，调用 startRecording() 录制音频到文件中（文件的路径格式是 `/your path` ）。
- 如果想取消录制可以调用 stopRecording 接口进行取消。
- 当录制完成后，调用 uploadRecordedFile 将文件上传到 GcloudVoice 的服务器上，该过程会通过 onUploadReccordFileComplete 回调在上传成功的时候返还一个 ShareFileID。该 ID 是这个文件的唯一标识符，用于其他用户收听时的下载。服务器需要对其进行管理和转发。
- 当游戏客户端需要收听其他人的录音时，首先从服务器获取转发的 ShareFileID，然后调用 downloadRecordedFile 下载该语言文件，下载结果通过 onDownloadRecordFileComplete 回调来通知。当下载成功时，就可以调用 playRecordedFile 播放下载完成的语音数据了。同样的，如果想取消播放，可以调用 stopPlayFile 进行取消。

语音转文字调用流程



- 调用 setMode 方法设置使用翻译（Translation）模式。
- 调用 applyMessageKey() 获取语音消息安全密钥 key 信息，当申请成功后会通过 onApplyMessageKeyComplete 进行回调。
- 当需要录音时，调用 startRecording() 录制音频到文件中（文件的路径格式是 `/your path` ）。
- 如果想取消录制可以调用 stopRecording 接口进行取消。

- 当录制完成后，调用 uploadRecordedFile 将文件上传到 GcloudVoice 的服务器上，该过程会通过 onUploadReccordFileComplete 回调在上传成功的时候返还一个 ShareFileID。该 ID 是这个文件的唯一标识符，用于其他用户收听时的下载。服务器需要对其进行管理和转发。
  翻译时，调用 speechToText，其中一个重要参数是上一步骤中的 fileid，调用该函数后，会在 onSpeechToText 回调中通知翻译结果。

> **注意：**
> 翻译的 fileid，需要是在 translation 模式下录制得来的，不是 message 模式录制得来的。

接口说明

基本 API

创建 IGcloudVoiceEngine 对象

1. 接口说明
   在使用语音功能时，需要首先获取 IGcloudVoiceEngine 对象。

2. 函数原型

```
+ (GVGCloudVoice* _Nullable) sharedInstance;
```

   该函数返回一个 GVGCloudVoice 对象，通过调用该对象的接口，可以执行相关动作。

3. 出错处理
   出错时，返回 NULL 对象。

4. 示例代码

```
[[GVGCloudVoice sharedInstance] closeMic];
```

设置业务信息

（1）接口说明
在初始化之前，需要设置之前申请的游戏 ID 和游戏 Key 以及用户的唯一标示 OpenID。

（2）函数原型

```
- (enum GCloudVoiceErrno) setAppInfo:(const char *_Nullable) appID withKey: (const char *_Nullable)appKey andOpenID:(const char *_Nullable)openID;
```

参数	类型	意义
appID	const char *	开通业务页面中的游戏 ID
appKey	const char *	开通业务页面中的游戏Key
openID	const char *	玩家唯一标识，比如从手 Q 或者微信获得到的 OpenID

（3）返回值
GCloudVoiceErr 成功时返回 GCLOUD_VOICE_SUCC 。

（4）示例代码

```
[[GVGCloudVoice sharedInstance] setAppInfo:"93223849489" withKey:"d94749efe9dddfce61333121de84123ef9b" andOpenID:[_openID cStringUsingEncoding:NSUTF8StringEncoding]];
```

初始化引擎

1. 接口说明
   在设置好业务信息后就可以开始初始化引擎了。

2. 函数原型

```
- (enum GCloudVoiceErrno) initEngine;
```

3. 示例代码

```
[[GVGCloudVoice sharedInstance] initEngine];
```

4. 出错处理
   GCLOUD_VOICE_NEED_SETAPPINFO ：需要先调用 SetAppInfo。

设置引擎模式

（1）接口说明
在使用语音功能时，需要根据需要设置成离线、实时还是转文字的模式。

- 如果是小队语音或者国战语音，设置成实时模式；
- 如果是语音消息，设置成离线模式；
- 如果是语音转文字，设置成翻译模式。

（2）函数原型

```
enum GCloudVoiceMode
{
RealTime = 0, // realtime mode for TeamRoom or NationalRoom
Messages, // voice message mode
Translation, // speach to text mode
};
(enum GCloudVoiceErrno) setMode:(enum GCloudVoiceMode) mode;
```

参数	类型	意义
mode	GCloudVoiceMode	如果是小队语音或者国战语音，设置成实时模式；

如果是语音消息，设置成离线模式；
如果是语音转文字，设置成翻译模式 |

（3）示例代码

```
[[GVGCloudVoice sharedInstance] setMode:RealTime];
```

（4）出错处理
GCLOUD_VOICE_NEED_SETAPPINFO ：需要先调用 SetAppInfo。

查询触发事件回调

1. 接口说明
   通过在 update 里面周期的调用 Poll 可以触发事件回调

2. 函数原型

```
- (enum GCloudVoiceErrno) poll;
```

3. 示例代码

```
_pollTimer = [NSTimer scheduledTimerWithTimeInterval:1.000/15 repeats:YES block:^(NSTimer * _N
onnull timer) {
```

```
[[GVGCloudVoice sharedInstance] poll];
}];
```

4. 出错处理
   GCLOUD_VOICE_NEED_INIT ：需要先调用 Init 进行初始化。

系统发生 Pause

（1）接口说明
当系统发生 Pause 事件时，需要同时通知引擎进行 Pause。

（2）函数原型

```
- (enum GCloudVoiceErrno) pause;
```

（3）出错处理
GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。

系统发生 Resume

1. 接口说明
   当系统发生 Resume 事件时，需要同时通知引擎进行 Resume。

2. 函数原型

```
- (enum GCloudVoiceErrno) resume;
```

3. 出错处理
   GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。

实时语音 API

加入小队语音

（1）接口说明
使用实时语音的小队语音功能时，需要先加入小队语音房间。

（2）函数原型

```
- (enum GCloudVoiceErrno) joinTeamRoom:(const char * _Nullable)roomName timeout: (int) msTimeout;
```

参数	类型	意义
roomName	const char *	要加入的房间名
msTimeout	int	加入房间的超时时间，单位是毫秒

加入房间的结果，需要通过 void OnJoinRoom(GCloudVoiceCompleteCode code, const char *roomName, int memberID)；进行回调。

（3）示例代码

```
[[GVGCloudVoice sharedInstance] joinTeamRoom:[_roomID cStringUsingEncoding:NSUTF8StringEncoding] timeout:18000];
```

（4）出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。

GCLOUD_VOICE_MODE_STATE_ERR ： 需要先调用 SetMode 设置成实时模式 。

GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，比如房间名为空或者超长（最大长度 127 字节）且由 a-z 、 A-Z 、 0-9 、 - 组成。超时范围 5000 ms ~ 60000 ms。

GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如已经加入房间了，需要先调用 QuitRoom 才能再次加入。

加入国战语音

（1）接口说明

使用国战语音功能时，需要先加入国战语音房间。

（2）函数原型

```
enum GCloudVoiceMemberRole
{
Anchor = 1, // member who can open microphone and say
Audience, // member who can only hear anchor's voice
};
- (enum GCloudVoiceErrno) joinNationalRoom:(const char *_Nullable)roomName role: (enum GCloudVoiceMemberRole) role timeout: (int) msTimeout;
```

参数	类型	意义
roomName	const char *	要加入的房间名
role	GCloudVoiceMemberRole	成员加入的角色，听众只能收听语音不能发送语音，而主播既可以发送语音也可以收听语音

参数	类型	意义
msTimeout	int	加入房间的超时时间，单位是毫秒

加入房间的结果，需要通过 `void OnJoinRoom(GCloudVoiceCompleteCode code, const char *roomName, int memberID)` ; 进行回调。

（3）示例代码

```
[[GVGCloudVoice sharedInstance] joinNationalRoom:[_roomID cStringUsingEncoding:NSUTF8StringEncoding] role:role timeout:18000];
```

（4）出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。

GCLOUD_VOICE_MODE_STATE_ERR ： 需要先调用 SetMode 设置成实时模式。

GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，比如房间名为空或者超长（最大长度 127 字节）且由 `a-z` 、 `A-Z` 、 `0-9` 、 `-` 组成。超时范围 5000 ms ~ 60000 ms。

GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如已经加入房间了，需要先调用 QuitRoom 才能再次加入。

退出实时语音

（1）接口说明

当不需要使用实时语音，可以从实时语音（包括小队语音和国战语音）房间中退出。

（2）函数原型

```
- (enum GCloudVoiceErrno) quitRoom:(const char * _Nullable)roomName timeout:(int) msTimeout ;
```

参数	类型	意义
roomName	const char *	要退出的房间名
msTimeout	Int	退出房间的超时时间，单位是毫秒

退出房间的结果，需要通过 `void OnQuitRoom(GCloudVoiceCompleteCode code, const char *roomName)` 进行回调

（3）示例代码

```
[[GVGCloudVoice sharedInstance] quitRoom:[_roomID cStringUsingEncoding:NSUTF8StringEncoding] timeout:18000];
```

（4）出错处理
GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式。
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，比如房间名为空或者超长（最大长度 127 字节）且由 a-z 、 A-Z 、 0-9 、 - 组成。超时范围 5000 ms ~ 60000 ms。
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间。

打开麦克风

（1）接口说明
在实时语音的模式下，加入房间成功后（包括小队语音和国战语音），需要打开麦克风才能采集音频并发送到网络。

（2）函数原型

```
- (enum GCloudVoiceErrno) openMic;
```

（3）出错处理
GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式。
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间。
GCLOUD_VOICE_OPENMIC_NOTANCHOR_ERR ： 当前以听众身份加入的大房间，不能开麦。

关闭麦克风

（1）接口说明
在实时语音的模式下，加入房间成功后（包括小队语音和国战语音），当不需要采集音频并发送到网络，可以调用关闭麦克风接口。

（2）函数原型

```
- (enum GCloudVoiceErrno) closeMic;
```

（3）出错处理
GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式。
GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间。
GCLOUD_VOICE_OPENMIC_NOTANCHOR_ERR ： 当前以听众身份加入的大房间，不能开麦关麦。

打开扬声器

（1）接口说明
在实时语音的模式下，加入房间成功后（包括小队语音和国战语音），需要打开扬声器才能从网络接收数据并播放。

（2）函数原型

```
- (enum GCloudVoiceErrno) openSpeaker;
```

（3）出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。

GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式。

GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间。

关闭扬声器

（1）接口说明

在实时语音的模式下，加入房间成功后（包括小队语音和国战语音），当不需要从网络接收数据并播放时，可以调用关闭麦克风接口。

（2）函数原型

```
- (enum GCloudVoiceErrno) closeSpeaker;
```

（3）出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。

GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是实时语音模式。

GCLOUD_VOICE_REALTIME_STATE_ERR ： 实时语音状态不对，比如还没有加入房间。

加入房间回调

（1）接口说明

当加入房间成功或者失败时会通过回调进行通知。

（2）函数原型

```
- (void) onJoinRoom:(enum GCloudVoiceCompleteCode) code withRoomName: (const char * _Nullable)roomName andMemberID:(int) memberID;
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见 GCloudVoiceCompleteCode 定义
roomName	const char *	加入的房间名
memberID	Int	如果加入成功的话，表示加入后的成员 ID

（3）示例代码

```
#pragma mark delegate

- (void) onJoinRoom:(enum GCloudVoiceCompleteCode) code withRoomName: (const char * _Nullable)roomName andMemberID:(int) memberID {
NSString *msg;
if (GV_ON_JOINROOM_SUCC == code) {
msg = [NSString stringWithFormat:@"Join Room Success"];
} else {
msg = [NSString stringWithFormat:@"Join Room Failed with code: %d", code];
}
[self warnning:msg];
}
```

退出房间回调

（1）接口说明

当退出房间时，结果通过回调进行通知。

（2）函数原型

```
- (void) onQuitRoom:(enum GCloudVoiceCompleteCode) code withRoomName: (const char * _Nullable)roomName
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见 GCloudVoiceCompleteCode 定义
roomName	const char *	退出的房间名
memberID	Int	如果加入成功的话，表示加入后的成员 ID

（3）示例代码

```
- (void) onQuitRoom:(enum GCloudVoiceCompleteCode) code withRoomName: (const char * _Nullable)roomName {
[_pollTimer invalidate];
}
```

成员状态改变回调

（1）接口说明

当房间中的其他成员开始说话或者停止说话的时候，通过该回调进行通知。

（2）函数原型

```
- (void) onMemberVoice:(const unsigned int * _Nullable)members withCount: (int) count
```

参数	类型	意义
members	Int[]	改变状态的 member 成员，其值为 [memberID status] 这样的对，总共有 count 对，status 有三种状态：0：停止说话；1：开始说话；2：继续说话。
count	Int	改变状态的成员的数目。

（3）示例代码

```
- (void) onMemberVoice: (const unsigned int * _Nullable)members withCount: (int) count {
for (int i=0; i
```

离线语音 API

申请语音消息 Key

（1）接口说明

在语音消息的模式下，需要先申请许可才可以正常使用。

（2）函数原型

```
- (enum GCloudVoiceErrno) applyMessageKey:(int) msTimeout;
```

参数	类型	意义
msTimeout	Int	超时时间，单位毫秒

申请的结果通过 void OnApplyMessageKey(GCloudVoiceCompleteCode code) ；进行回调。

（3）出错处理

GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，比如超时范围 5000 ms ~ 60000 ms。

GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。

GCLOUD_VOICE_AUTHKEY_ERR ： 请求 Key 的内部错误，此时请联系 GCloud 团队，并提供日志进行定位。

限制最大语音消息的长度

（1）接口说明

在语音消息的模式下，可以限制最大语音消息的长度，目前默认是 2 min，最大不超过 2 min。

（2）函数原型

```
- (enum GCloudVoiceErrno) setMaxMessageLength:(int) msTime;
```

参数	类型	意义
msTimeout	Int	最大语音消息长度，单位毫秒

（3）出错处理
GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，时间范围 1000ms-1000260ms。

开始录音

（1）接口说明
在语音消息的模式下，开始录音时，需要提供一个录音文件存储的地址路径。

（2）函数原型

```
- (enum GCloudVoiceErrno) startRecording:(const char *_Nullable) filePath;
```

参数	类型	意义
filePath	const char *	录音文件存储的地址路径，路径中需要 / 作分隔，不能用 \

（3）出错处理
GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式。
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，路径为空。
GCLOUD_VOICE_NEED_AUTHKEY ： 需要先调用 GetAuthKey 申请许可。
GCLOUD_VOICE_PATH_ACCESS_ERR ： 提供的路径不合法或者不可写。

停止录音

（1）接口说明
在语音消息的模式下，调用停止录音接口。
（2）函数原型

```
- (enum GCloudVoiceErrno) stopRecording;
```

（3）出错处理
GCLOUD_VOICE_NEED_INIT ： 需要先调用Init进行初始化。
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式。
GCLOUD_VOICE_NEED_AUTHKEY ： 需要先调用 GetAuthKey 申请许可。

上传录音的文件

（1）接口说明

录音完成后，通过提供一个录音文件存储的地址路径，将已经录音完的文件进行上传。

（2）函数原型

```
- (enum GCloudVoiceErrno) uploadRecordedFile:(const char *_Nullable) filePath timeout: (int) msTimeout ;
```

参数	类型	意义
filePath	const char *	录音文件存储的地址路径，路径中需要 / 作分隔，不能用 \
msTimeout	Int	上传文件超时时间

上传的结果通过 `void OnUploadFile(GCloudVoiceCompleteCode code, const char *filePath, const char *fileID)` 进行回调。

（3）出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式。
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，路径为空。
GCLOUD_VOICE_NEED_AUTHKEY ： 需要先调用 GetAuthKey 申请许可。
GCLOUD_VOICE_PATH_ACCESS_ERR ： 提供的路径不合法或者不可读。
GCLOUD_VOICE_HTTP_BUSY ： 还在上一次上传或者下载中，需要等待后再尝试。

下载录音的文件

（1）接口说明

录音完成后，通过提供一个录音文件存储的地址路径，将已经录音完的文件进行上传。

（2）函数原型

```
- (enum GCloudVoiceErrno) downloadRecordedFile:(const char *_Nullable)fileID filePath:(const char *_Nullable) downloadFilePath timeout: (int) msTimeout ;
```

参数	类型	意义
fileID	const char *	要下载文件的文件 ID
downloadFilePath	const char *	下载录音文件存储的地址路径，路径中需要 / 作分隔，不能用 \
msTimeout	Int	下载文件超时时间

下载的结果通过 `void OnDownloadFile(GCloudVoiceCompleteCode code, const char *filePath, const char *fileID)` ;进行回调。

（3）出错处理
GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式。
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，路径为空。
GCLOUD_VOICE_NEED_AUTHKEY ： 需要先调用GetAuthKey 申请许可。
GCLOUD_VOICE_PATH_ACCESS_ERR ： 提供的路径不合法或者不可写或者不可读。
GCLOUD_VOICE_HTTP_BUSY ： 还在上一次上传或者下载中，需要等待后再尝试。

开始播放下载的音频

（1）接口说明
下载下来的音频文件，需要调用相关接口进行播放。

（2）函数原型

```
- (enum GCloudVoiceErrno) playRecordedFile:(const char *_Nullable) downloadFilePath;
```

参数	类型	意义
filePath	const char *	下载录音文件存储的地址路径，路径中需要 / 作分隔，不能用 \

如果正常播放完，会回调 `void OnPlayRecordedFile(GCloudVoiceCompleteCode code,const char *filePath)` 。

（3）出错处理
GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。
GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式。
GCLOUD_VOICE_PARAM_INVALID ： 传入的参数不对，路径为空。
GCLOUD_VOICE_PATH_ACCESS_ERR ： 提供的路径不合法或者不可写。
GCLOUD_VOICE_SPEAKER_ERR：打开麦克风失败。

停止播放下载的音频

（1）接口说明
中断播放动作。

（2）函数原型

```
(enum GCloudVoiceErrno) stopPlayFile;
```

（3）出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。

GCLOUD_VOICE_MODE_STATE_ERR ： 当前模式不是离线语音模式。

请求语音消息 Key 回调

（1）接口说明

请求语音消息许可的时候会回调

（2）函数原型

```
- (void) onApplyMessageKey:(enum GCloudVoiceCompleteCode) code
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见 GCloudVoiceCompleteCode 定义

（3）示例代码

```
- (void) onApplyMessageKey:(enum GCloudVoiceCompleteCode) code {
NSString *msg;
msg = @"Apply AuthKey Success";
[self warnning:msg];
}
```

上传完成回调

（1） 接口说明

上传语音文件后的结果通过这个接口进行回调。

（2）函数原型

```
- (void) onUploadFile: (enum GCloudVoiceCompleteCode) code withFilePath: (const char * _Nullable)filePath andFileID:(const char * _Nullable)fileID
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见 GCloudVoiceCompleteCode 定义
filepath	const char *	上传的文件路径
fileid	const char *	文件的 ID

（3）示例代码

```
- (void) onUploadFile: (enum GCloudVoiceCompleteCode) code withFilePath: (const char * _Nullable)filePath andFileID:(const char * _Nullable)fileID {
_fileID = [NSString stringWithFormat:@"%s", fileID];
[self warnning:@"Upload Success"];
}
```

下载完成回调

（1）接口说明

下载语音文件后的结果通过这个进行回调。

（2）函数原型

```
- (void) onDownloadFile: (enum GCloudVoiceCompleteCode) code withFilePath: (const char * _Nullable)filePath andFileID:(const char * _Nullable)fileID
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见 GCloudVoiceCompleteCode 定义
filepath	const char *	下载的路径
fileid	const char *	文件的 ID

（3）示例代码

```
- (void) onDownloadFile: (enum GCloudVoiceCompleteCode) code withFilePath: (const char * _Nullable)filePath andFileID:(const char * _Nullable)fileID {
NSString *msg;
msg = @"Download File Success";
[self warnning:msg];
}
```

语音转文字

进行语音转文字

（1）接口说明

使用该函数，进行语音转文字。

（2）函数原型

```
- (enum GCloudVoiceErrno) speechToText:(const char *_Nullable)fileID timeout:(int) msTimeout language:(enum GCloudLanguage) language ;
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见 GCloudVoiceCompleteCode 定义
vfileID	String	需要翻译的文件 ID
result	String	翻译的文字结果

（3）出错处理

GCLOUD_VOICE_NEED_INIT ： 需要先调用 Init 进行初始化。

GCLOUD_VOICE_STTING ： 正在进行上一次的语音转文字。

（4）示例代码

```
[[GVGCloudVoice sharedInstance] speechToText:[_fileID cStringUsingEncoding:NSUTF8StringEncoding] timeout:18000 language:China];
```

语音转文字完成后回调

（1）接口说明

语音转文字的结果通过这个回调进行通知。

（2）函数原型

```
- (void) onSpeechToText:(enum GCloudVoiceCompleteCode) code withFileID:(const char * _Nullable) fileID andResult:( const char * _Nullable)result
```

参数	类型	意义
code	GCloudVoiceCompleteCode	参见 GCloudVoiceCompleteCode 定义
fileID	const char *	翻译文件的 fileid
result	const char *	翻译的文字结果

（3）示例代码

```
- (void) onSpeechToText:(enum GCloudVoiceCompleteCode) code withFileID:(const char * _Nullable) fileID andResult:( const char * _Nullable)result {
_resultTV.text = [NSString stringWithUTF8String: result];
}
```

SDK下载

最近更新时间：2018-08-23 15:48:16

v1.1.19

Unity
Unity(bitcode)
Cocos
Cocos(bitcode)
Win64
Win32
iOS
iOS(bitcode)
Android
UE4
symbol

更新说明：
1、新增功能

- 音质优化，流畅性优化;
- 蓝牙耳机兼容性优化；
- 第三方 app 中断响应（QQ，微信，系统闹钟，电话等）优化;
- 服务器动态调整算法参数支持。

2、新增接口
增加开关麦成功或失败回调（见 OnEvent 回调接口，枚举值详见 GCloudVoiceErrno.h）

v1.1.17

C++\Cocos2D SDK 1.1.17 版本

Unity3D SDK 1.1.17 版本

Android SDK 1.1.17 版本

iOS SDK 1.1.17 版本

iOS(bitcode) SDK 1.1.17 版本

Win64 1.1.17 版本

Win32 1.1.17 版本

UE4 1.1.17 版本

1. OC 接口修复设置音效功能。
2. 修复部分数据上报异常（openID+url）为空。
3. OC 增加 setvoiceeffict 接口。

v1.1.17.beta

C++\Cocos2D SDK 1.1.17.beta 版本

Unity3D SDK 1.1.17.beta 版本

Android SDK 1.1.17.beta 版本

iOS SDK 1.1.17.beta 版本

iOS(bitcode) SDK 1.1.17.beta 版本

Win64 1.1.17.beta 版本

Win32 1.1.17.beta 版本

UE4 1.1.17.beta 版本

1. 增强抗丢包和网络抖动能力，提升上下行语音数据的可靠性。
2. 此版本为公测版本，请根据需求选择接入。

v1.1.16

C++\Cocos2D SDK 1.1.16 版本

Unity3D SDK 1.1.16 版本

Android SDK 1.1.16 版本

iOS SDK 1.1.16 版本

iOS(bitcode) SDK 1.1.16 版本

Win64 1.1.16 版本

Win32 1.1.16 版本

UE4 1.1.16 版本

1. 修复部分 BUG。
2. 在使用变声功能的时候，提高语音识别的翻译准确率。

v1.1.15

C++\Cocos2D SDK 1.1.15 版本

Unity3D SDK 1.1.15 版本

Android SDK 1.1.15 版本

iOS SDK 1.1.15 版本

iOS(bitcode) SDK 1.1.15 版本

Win64 1.1.15 版本

Win32 1.1.15 版本

UE4 1.1.15 版本

1. 提供全局范围语音功能，基于游戏中玩家坐标范围来决定语音通话参与者数量（可参考 PUBG 中的全局语音功能）。
2. 支持加入多语音房间，现在一个玩家最多可以参与 2 个小队语音，或参与 1 个小队语音和 1 个范围语音。

v1.1.14（不发布）

1. 代码合入

v1.1.13

C++\Cocos2D SDK 1.1.13 版本

Unity3D SDK 1.1.13 版本

Android SDK 1.1.13 版本

iOS SDK 1.1.13 版本

iOS(bitcode) SDK 1.1.13 版本

Win64 1.1.13 版本

Win32 1.1.13 版本

1. 噪音优化
2. 回声优化

v1.1.12 （2017-10-18）

C++\Cocos2D SDK 1.1.12 版本

Unity3D SDK 1.1.12 版本

Android SDK 1.1.12 版本

iOS SDK 1.1.12 版本

iOS(bitcode) SDK 1.1.12 版本

Win64 1.1.12 版本

Win32 1.1.12 版本

1. 语音增强扛丢包和网络抖动的能力
2. 语音消息最长时间从 3 分钟提升至 5 分钟
3. 支持每个用户（按 OpenID 区别）保存一条永久语音消息

v1.1.11（2017-9-13）

1. 支持识别男、女声音
2. 支持变声
3. 解决部分小运营商域名劫持的问题
4. 解决高质量语音模式下，小队语音出现声音卡顿的问题
5. 修复接入蓝牙耳机，耳机低电量时，声音从扬声器出来的缺陷
6. 修复接入蓝牙耳机，录制语音消息时，结束录音后声音从扬声器出来的缺陷
7. 修复接入蓝牙耳机，实时语音中，拨打电话后，声音从扬声器出来的缺陷

v1.1.8（2017-06-23）

C++\Cocos2D SDK 1.1.8 版本

Unity3D SDK 1.1.8 版本

Android SDK 1.1.8 版本

iOS SDK 1.1.8 版本

Win64 1.1.8 版本

Win32 1.1.8 版本

1. GVoice 海外服务地区新增日本、马来西亚两个国家
2. 新增高质量语音模式，语音更清晰，同时会增加流量消耗，专门应用于狼人杀等以语音为核心玩法的游戏。
3. 在调用 SetMode() 的时候，传入 HIGHQUALITY 枚举值即可使用高质量语音模式
4. iOS 平台的 SDK，由 C++ 变为 Objective-C
5. 由于关闭了 iOS bitcode 选项，SDK 包变小是正常现象

Demo下载

最近更新时间：2018-01-08 15:52:48

Cocos2D demo 1.1.8 下载

Unity3D demo 1.1.8 下载

Android demo 1.1.8 下载

IOS demo 1.1.8 下载