GME for Wwise Plug-in

Contents

Version information

GME for Wwise Plug-in

The current document covers the following GME plug-in versions:

GME Plug-in version	Compatible with Wwise Major Version
2019.2.6	2019.2
2021.1.6	2021.1

GME plug-ins are compatible with the corresponding Wwise major version, including any minor version (i.e., versions are compatible if they share the same *year.major*).

GME SDK

The GME SDK version used with this version is **2.4.10**.

The library was built against the following platform SDKs:

Platform	Platform SDK version
Windows	10.0.18362.1
Android	NDK r16b, android-14
macOS	Xcode 10.1
iOS	Xcode 10.1
Xbox One	XDK 10.0.17134.5063 (July 2018 QFE9)
PS4	8.5
Switch	10.4.1 (Ask representative for libraries)

Release note

GME V6: Plug-in versions 2019.2.6 and 2021.1.6

- Updated to GME SDK 2.4.10
- Converted the bus effect plug-in “Tencent GME Session” to an Audio Device Effect plug-in for 2021.1.
- Updated packaged demos to make use of new features
- Support dropped for 2019.1
- New API functions:
    - Connection management:
        - GMEWWisePlugin_Pause
        - GMEWWisePlugin_Resume
    - Network audio stream monitoring
        - GMEWWisePlugin_GetAudioSendStreamLevel
        - GMEWWisePlugin_GetAudioRecvStreamLevel
    - User blocklist
        - GMEWWisePlugin_AddAudioBlockList
        - GMEWWisePlugin_RemoveAudioBlockList
    - Audio message with speech-to-text
        - GMEWWisePlugin_StartRecording
        - GMEWWisePlugin_StopRecording
        - GMEWWisePlugin_PlayRecordFile
        - GMEWWisePlugin_StopPlayFile
        - GMEWWisePlugin_GetVoiceFileDuration
        - GMEWWisePlugin_SpeechToText
- Changes to existing functions:
    - Added error codes to **GMEWWisePlugin_SetUserID** and **GMEWWisePlugin_SetRoomID**
    - Defined events provided by **GMEWWisePlugin_GetMessage**
- Bug fixes
    - **WG-54324** Fixed: Profiling clears authentication information.

Known Issues

The following issues are known to affect the current version of the plug-in. Before using the plug-in, make sure you are aware of these issues to evaluate how they can affect your product.

- **WG-57103:** Support by request for Switch platform
- **WG-56778:** No prompt for microphone in Unreal 4.27
- **WG-56723:** Apple M1 unsupported by GME
- **WG-56553**: Some network exchanges still occur after GMEWWisePlugin_Pause
- **WG-46843:** Wwise profiler interrupts GME on macOS
- **WG-46754:** GME does not work with Xbox One's proprietary controller headset
  Note: *GME works with any standard headset connected to the jack port of the* Xbox *One controller*

Previous releases

GME V5: Plug-in version 2019.1.5 and 2019.2.5

- Updated to GME SDK **2.4.5**
- Replaced the Audio Device (sink) plug-in by an effect plug-in "Tencent GME Session"
- Updated demo project IntegrationDemoGME to sources from latest Wwise versions
- Support dropped for Wwise 2018.1
- API changes:
    - GMEWWisePlugin_SetLogLevel can now set a different logging level for writing to files and printing to console
- Bugfixes:
    - **WG-51224** Fixed: Unreal Blueprint have implicit int32 to bool conversions

GME V4: Plug-in version 2018.1.4, 2019.1.4 and 2019.2.4

- New API functions:
    - GMEWWisePlugin_SetRangeAudioTeamID
    - GMEWWisePlugin_SetRangeAudioTeamMode
    - GMEWWisePlugin_SetRangeAudioRecvRange
    - GMEWWisePlugin_SetSelfPosition
- Updated to GME SDK **2.4.4.2a03146b**
- Bugfixes
    - **WG-46845** Fixed: TencentGME plug-in not automatically added to Xcode project on iOS in Unity

GME V3: Plug-in version 2018.1.0, 2019.1.0 and 2019.2.0

- Initial release
- Includes GME SDK **2.4.0.324.**

Migration Notes

Migrating to GME V6: Plug-in versions 2019.2.6 and 2021.1.6

Wwise Project

If you are *upgrading a project that used a previous version of GME for Wwise to* ***2021.1***, you need to replace the GME Session effect plug-in (see Figure 2a) by its Audio Device Effect counterpart in the System Audio Device (see Figure 2b). The plug-in must be inserted in an Effect slot of the System Audio Device. The interface of the new Audio Device Effect plug-in is the same as the previous effect plug-in and serves the same purpose, with the exception that it allows proper handling of an Audio Object pipeline. *This change is not required with Wwise* ***2019.2****,* as that version of Wwise does not support the Audio Object pipeline.

Copy the Authentication Key from the Audio Device Effect plug-in and paste it in the same field of the GME Session effect plug-in interface. No other changes are required in the Wwise Project.

API

The functions **GMEWWisePlugin_SetUserID** and **GMEWWisePlugin_SetRoomID** now return an error code in the event of invalid argument formatting: you may want to use this to add validation to your code.

Error and status events are now reported by the previously available **GMEWWisePlugin_GetMessage** function. See defines with prefixes:

- **"GMESDK_MESSAGETYPE_*"**: Defines the type of message
- **"GMESDK_Code_Error_*"**: Specific error codes
- **"GMESDK_Code_RoomStatus_*"**: Specific room status codes

Refer to the API documentation for more details regarding **GMEWWisePlugin_GetMessage**.

Unity Integration

The GMESDK asset provided as part of the sample Unity project found in <Wwise>/SDK/Plugins/TencentGME/Unity must be manually updated. This is because it is provided separately from the plug-in during Wwise Unity Integration. Remove the **GMESDK** directory from the **Assets** directory of your Unity project. Then replace it with a copy of the **GMESDK** directory that is installed with the new version of GME, found in the following directory:

```
<Wwise>/SDK/Plugins/TencentGME/Unity/GMEWwiseDemo/Assets
```

Unreal Integration

The TencentGME_Wwise Unreal plug-in that is provided as part of the sample Unreal project found in <Wwise>/SDK/Plugins/TencentGME/Unreal must be manually updated. This is because it is provided separately from the plug-in during Wwise Unreal Integration. Remove the **TencentGME_Wwise** directory from the **Plugins** directory of your Unreal project. Then replace it with a copy of the **TencentGME_Wwise** directory that is installed with the new version of GME, found in the following directory:

```
<Wwise>/SDK/Plugins/TencentGME/Unreal/GMEWwiseDemo/
Plugins
```

If you are building the Unreal Engine from source, you will need to regenerate your Visual Studio or Xcode solution and rebuild the plug-in.

Overview

Product Introduction

GME (Game Multimedia Engine) is a real-time game audio SDK that has been specifically designed to support many types of games, including casual, competitive (such as MOBA, MMORPG, and FPS games), and large-scale commander games. It provides functionality such as real-time multi-person voice chat, voice messages, and speech-to-text conversion.

For projects using Wwise, GME provides a turnkey solution for game developers to integrate in-game voice in creative and immersive ways.

Platform Support

In its integration with Wwise, the GME plugin supports the following platforms:

- Windows
- Android
- iOS
- macOS
- Xbox One
- PS4
- Switch

For more information on Wwise Platform SDK requirements, consult:

https://www.audiokinetic.com/library/edge/?source=SDK&id=reference_platform.html

Using GME Voice Communication Service

The GME voice communication service is cloud-based technology, so the developer's information needs to be authenticated in order to use the service. Developers need to have their own account registered on Tencent Cloud to obtain the corresponding Authentication Key. Audiokinetic provides this information to developers when they start their integration of GME. The Authentication Key can be retrieved from the Audiokinetic customer portal. If your project doesn't display the Authentication Key, please contact WwiseGME@audiokinetic.com.

GME for Wwise plug-in Installation

Download and install the GME for Wwise plug-in using the Wwise Launcher. GME is available for Wwise 2019.2 and above. All the files needed by the Wwise authoring tool are copied to the Wwise installation directory and are ready to be used in your project.

For projects developed with UE4 or Unity, please refer to the related section of this documentation to get the additional steps to run GME in your project.

GME Integration in Wwise

The following diagram illustrates the integration of GME in Wwise and provides an idea of how the project is set up and how it is executed at runtime.

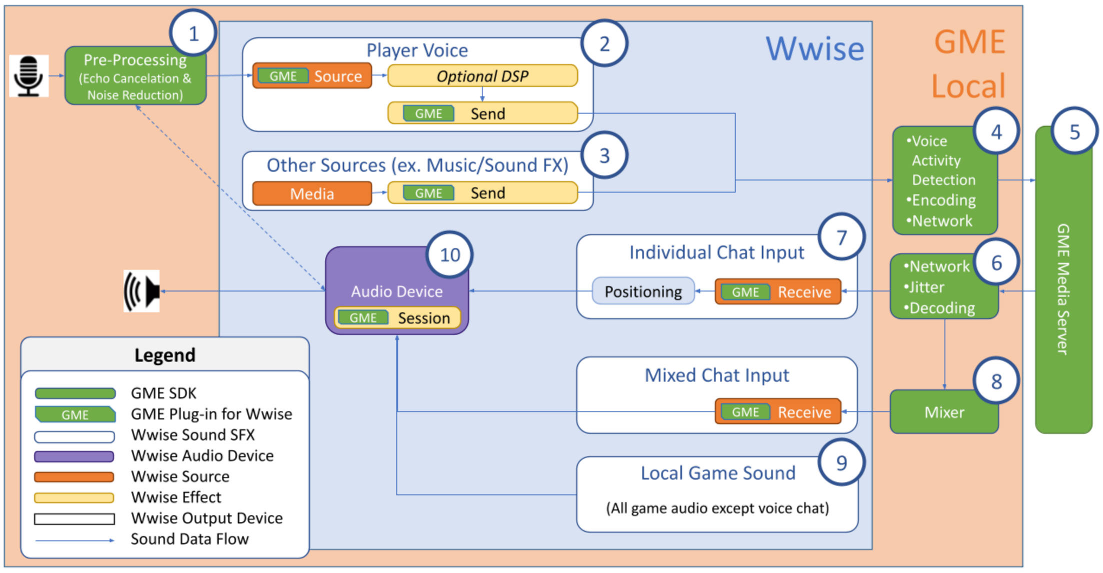


Figure 1 – Overview of GME integration in Wwise and runtime execution flow

1. GME captures the microphone input from the OS and applies noise reduction and echo cancellation. Echo cancellation needs the final rendered audio to operate. This is provided by the Tencent GME Session effect plug-in (#10) from the audio router to the Master Bus.
2. The local player's voice chat source is sent to teammates through GME servers. Audio effects can optionally be applied.
3. Any other game audio source such as music, ambient sounds, SFX, etc. can also be sent to GME servers.
4. Voice activity detection and encoding optimizes network bandwidth.
5. The GME media server receives voice chat and redirects individual streams to the proper receivers.
6. The incoming voices from the GME servers are received and managed, prior to their decompression into PCM audio.
7. Incoming voice chat is assigned to independent sound instances, which allows for additional DSP, distance attenuation, positioning, etc.
8. Voice chat can be mixed as a single audio input stream when desired.
9. All sounds, except audio received from GME servers, are locally triggered by the game.
10. The audio streams sent to the Audio Device are observed by the Tencent GME Session effect plug-in, which is necessary for echo cancellation (#1).

Authoring Tool

GME Plugin Introduction

To connect to the GME real-time voice service and support voice chat in your game, the following four plug-ins must be added to your project:

- **Tencent GME source plug-in:** Developed as a Wwise source plug-in. Its main function is to record the player's voice and pre-process the captured signals.
- **Tencent GME Session audio effect effect plug-in:** Developed as a Wwise audio device effect plug-in inserted on the System Audio Device. Its main function is to observe the audio routed to the Audio Device for echo cancellation and to set server authentication information.
- **Tencent GME Send plug-in:** Developed as a Wwise effect plug-in. Its main function is to send local recorded voice to the GME server.
- **Tencent GME Receive plug-in:** Developed as a Wwise source plug-in. The main function is to receive the voice chats from the same room from the GME server.

These plug-ins allow GME to capture, send and receive voice chat and other audio streams, such as sound effects and music between players registered to the same room.

Migration Notice

In the 2021.1 version of the plug-in, the Tencent GME Session effect plug-in that shipped with previous versions has been converted to an Audio Device Effect plug-in. When updating a project, remove the Tencent GME Session effect plug-in from the Master Audio Bus and follow the steps in the following section to add the Tencent GME Session effect plug-in as an Audio Device Effect plug-in instead.

Tencent GME Session plug-in Setup

In order to operate, GME has to observe the final signal so that echo cancellation can be applied. An effect plug-in is placed at the very end of the audio pipeline for that reason. This plug-in is also responsible for setting the authentication information for the server, as discussed in the next section.
Follow these steps to setup the Tencent GME Session effect plug-in for your project:

2019.2: As a Master Audio Bus Effect

In the Master Mixer Hierarchy, on the Master Audio Bus, add the Tencent GME Session effect plug-in in the last effect slot, as shown in Figure 2a. This ensures that the audio used for echo cancellation is as close as possible to what would be output by the system and could leak into the captured feed from the microphone.

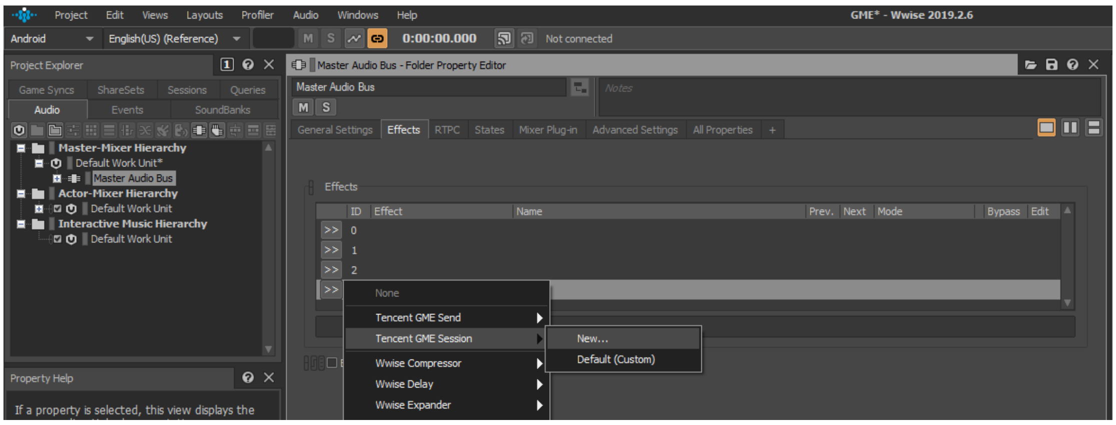


Figure 2a – Setting the Tencent GME Session effect plug-in in 2019.2

2021.1: As an Audio Device Effect

In the Audio Devices hierarchy, on the System Audio Device, add the Tencent GME Session Audio Device Effect as the last element in the effect chain or before the Mastering Suite, as shown in Figure 2b. This ensures that the audio used for echo cancellation is as close as possible to what would be output by the system and could leak into the captured feed from the microphone.

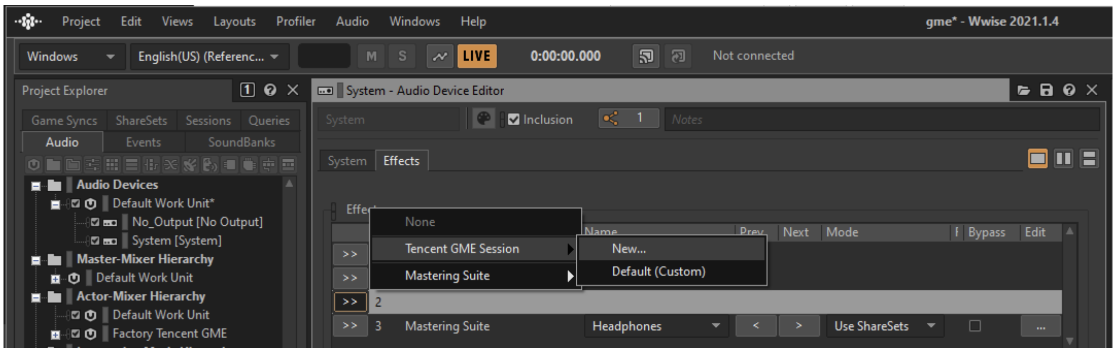


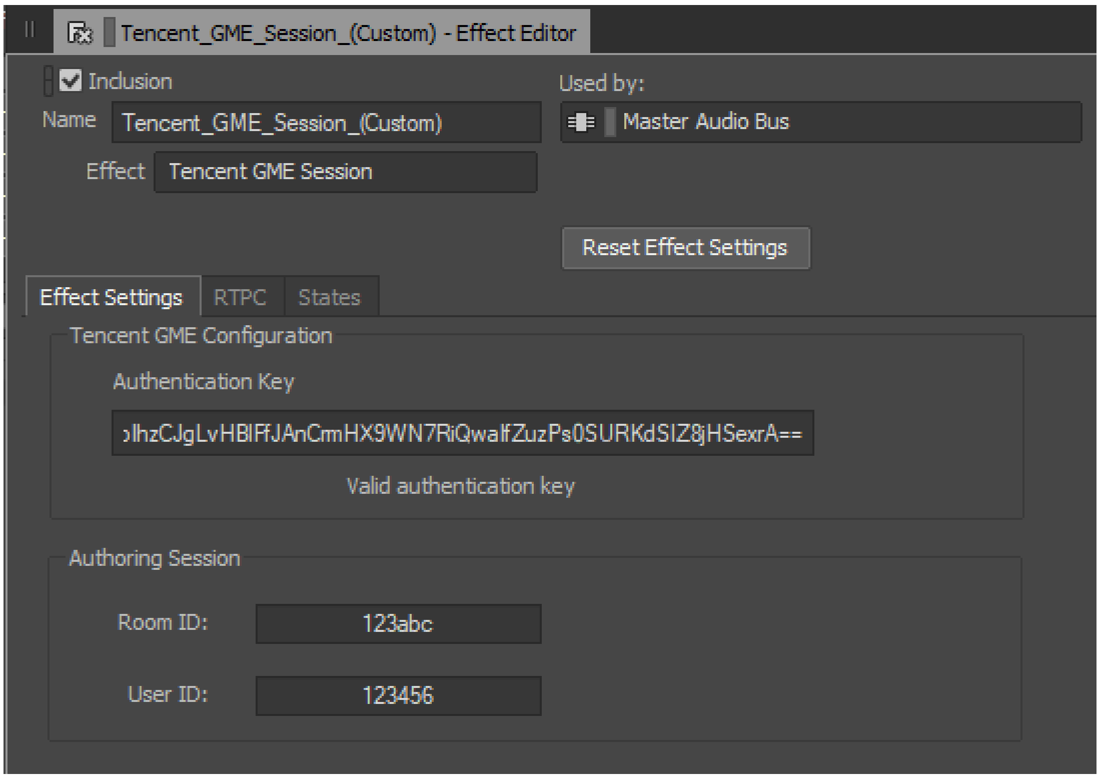
Figure 2b – Setting the Tencent GME Session Audio Device effect plug-in in 2021.1

GME Authentication Information and Chat Room Parameters Setup

GME needs to authenticate the game application during the voice chat room setup. Authentication requires a unique Authentication Key for the service to work. The Authentication Key is available to project leaders on the GME section of the Audiokinetic customer portal. If the information cannot be found on the customer portal under your project, please contact WwiseGME@audiokinetic.com.

The Authentication Key and various IDs are set in the Effects Settings of the Tencent GME Session effect plug-in. Simply double-click the Tencent GME Session effect plug-in to modify the settings accordingly.

- Each project has its unique Authentication Key, provided by Audiokinetic. The Authentication key is required to enable the system.
- The Room ID is used to create "chat rooms" in the game, so that all the participants can hear each other. Use different Room IDs if you need to separate players into different teams.
  - Room IDs can use up to 127 characters.
- Each user should have his/her own unique ID.
  - User IDs should be a 64 bit integer that is greater than 10,000.



Tencent_GME_Session_(Custom) - Effect Editor

Inclusion

Name: Tencent_GME_Session_(Custom)

Effect: Tencent GME Session

Used by: Master Audio Bus

Reset Effect Settings

Effect Settings | RTPC | States

Tencent GME Configuration

Authentication Key: ›lhzCJgLvHBlFfJAnCmHX9WN7RiQwalfZuzPs0SURKdSIZ8jHSexrA==

Valid authentication key

Authoring Session

Room ID: 123abc

User ID: 123456

Figure 3 – Tencent GME Session effect settings to establish connection with GME servers

Setting up GME for Voice Chat – Sending and Receiving

Setting up voice chat in Wwise simply requires setting up sending and receiving points in your project to enable the basic functionality of the system. This section explains the steps required to set up sending and receiving points in the Wwise project.

There are two ways to send audio from a player to other players, and the method used is determined by a simple question: Do you need to send discrete audio (e.g. the player's voice, a specific sound effect), or do you need to send an audio mixdown of multiple sound sources?

Sending Discreet Audio Signal

Sending a player's voice, in isolation from the rest of the game audio, is probably the most common scenario when integrating voice chat into a game. To send discreet audio, follow the instructions below:

1. Create a new Sound SFX in the Default Work Unit. under the Actor-Mixer Hierarchy, and name it: GME Send.
2. Select the newly created GME Send sound, click the Add Source >> button in the Content Editor, and select the GME Send Source plug-in as shown in Figure 4.

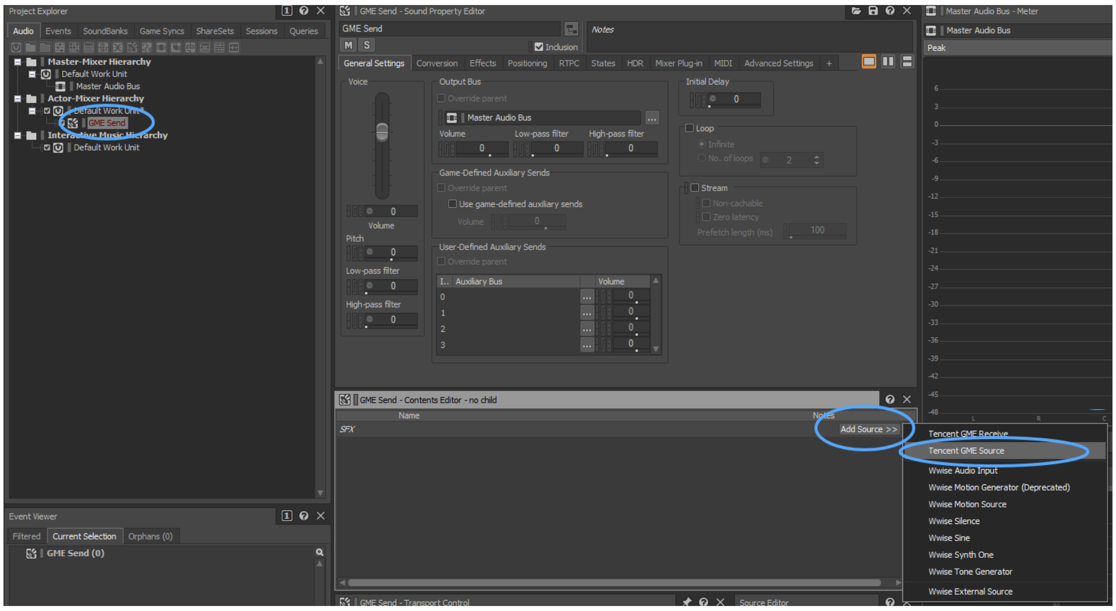


Figure 4 – Creating a new sound to send to GME servers

The GME Send source plug-in receives its audio from the device's microphone, as set in the operating system of your computer. There are no settings associated with this source plug-in. Now, in order to have the GME Send sound send its audio to the other players in the same room using the GME servers, we must configure the GME Send effect plug-in.

3. On the GME Send sound, select the Effects tab, and insert the GME Send effect plug-in in the last effect slot. Adding the effect on the last slot is not mandatory, but it allows us to insert other effects prior to sending the acquired audio to the server, if necessary.

   Note that the GME Send effect plug-in "steals" the audio from Wwise and sends it to the other players. The speaking player never hears his or her own voice in the final game mix.

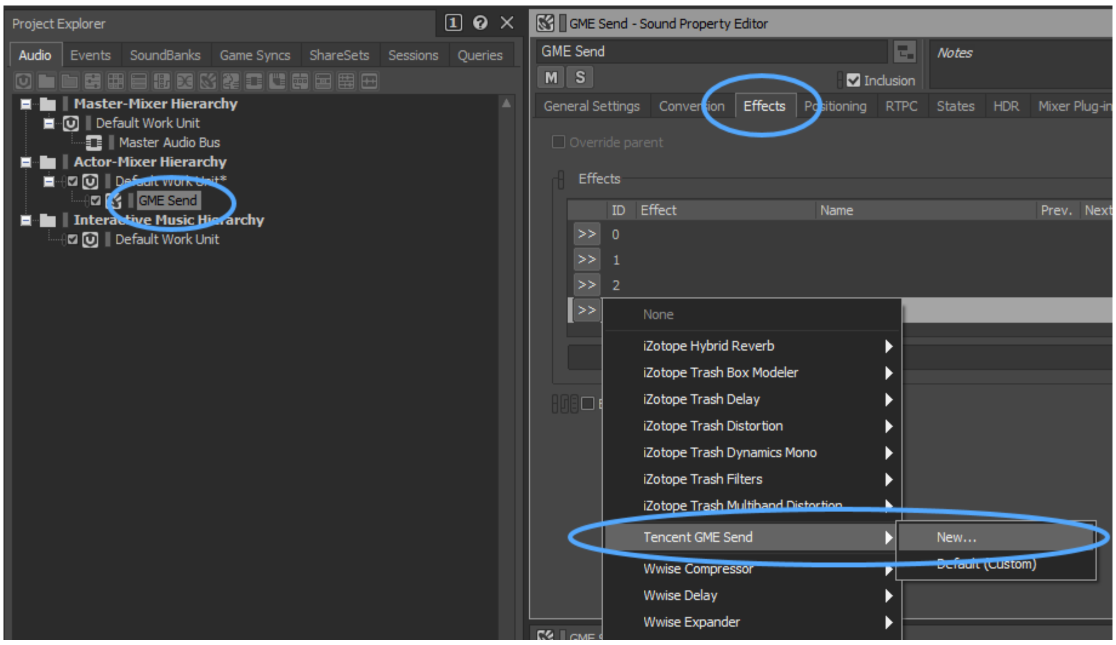


Figure 5 – Inserting GME Send effect on the sound

That's it! You're now set up to send audio from your system to other players in the same chat room (assuming you set the same Authentication Key, Room ID, but a different User ID on the Tencent GME effect plug-in, as previously described).

Sending Mixed Audio as a Single Chat Stream

There are situations where various sources may need to be mixed into a single "chat" signal before being sent to the other players in the chat room. For instance, if a game supports having one of the players act as the DJ for the rest of the group. In this situation, the music may come from many different sources, so it is best to set the GME Send effect plug-in to a bus instead of to each individual music file. Here's an easy way to achieve this:

1. Create a new "top" bus in the Master-Mixer Hierarchy, by right-clicking on the Default Work Unit, and selecting New Child – Bus. Name it: GME Send Bus.

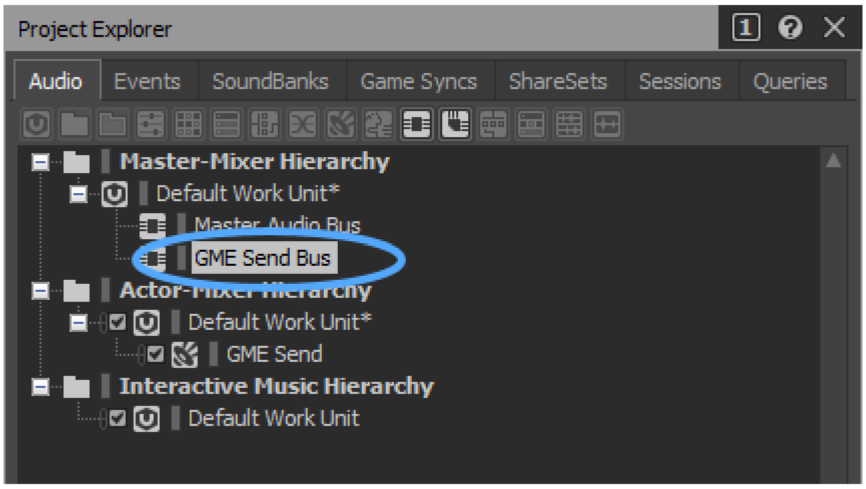


Figure 6 – Creating a new top bus named GME Send Bus

2. Select the GME Send Bus, click the Effects tab in the Property Editor, and insert the GME Send effect plug-in on the last effect slot, as shown in Figure 7. Adding the effect on the last slot is not mandatory, but it allows us to insert other effects prior

to sending the acquired audio to the server, if necessary.

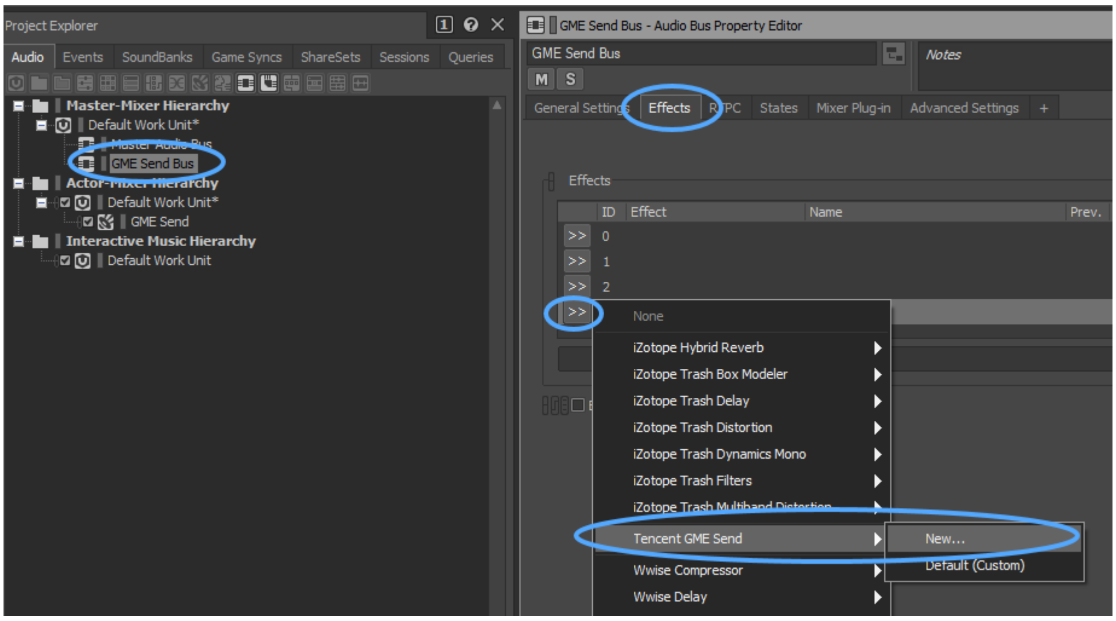


Figure 7 – Inserting GME Send effect on the bus

3. Any audio object routed to the GME Send Bus (or to a child of this bus) is mixed with the other concurrent sounds, which are routed to the same bus before being sent to GME and received by the other players. To send audio to the GME Send Bus, simply modify the Output Bus routing to the GME Send Bus on your sounds or containers from the General Settings tab, as shown on Figure 8.

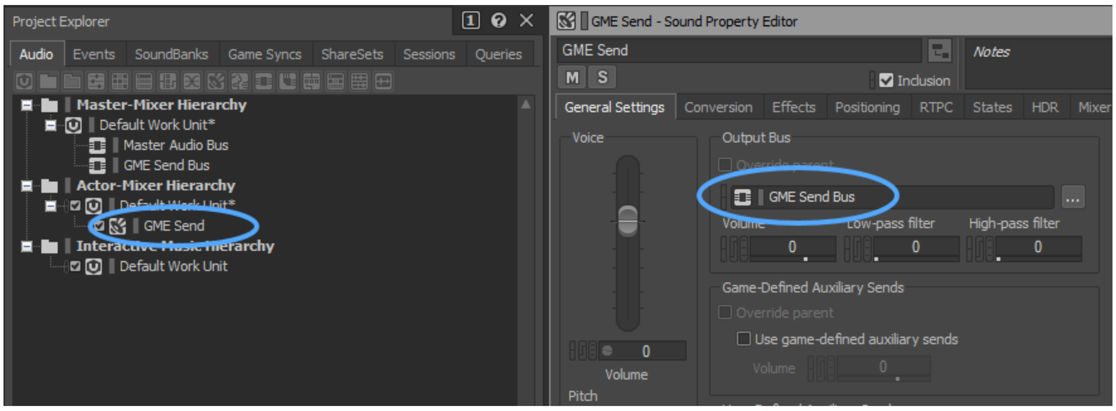


Figure 8 – Re-routing GME Send sound to send to GME Send Bus

Using the GME for Wwise plug-in to Receive Voice

Now that your project is configured to send audio, you simply need to add a receiving point for the audio. To create a receiving point for the voice chat, you must create a new sound that uses the Tencent GME Receive source plug-in:

1. Create a new Sound SFX in the Default Work Unit under the Actor-Mixer Hierarchy. Set the name to GME Receive, for example.
2. Select the newly created GME Receive sound, click the Add Source >> button in the Content Editor, and select GME Receive, as shown in Figure 9.

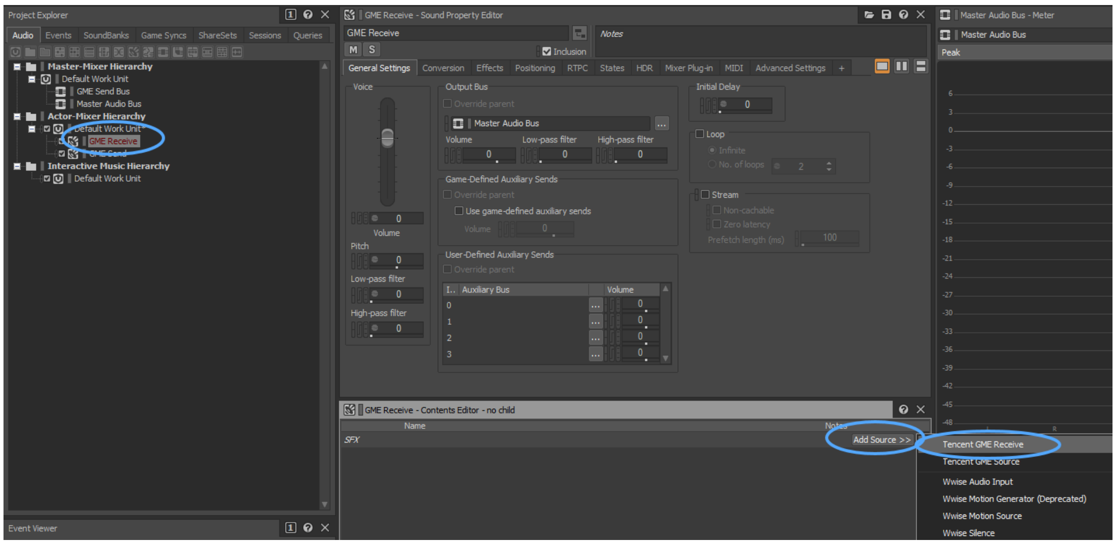


Figure 9 – Creating a new sound to receive audio from GME servers

3. Select the GME Receive sound to load it in the Transport and click the play button. If there are other players in the room, you should hear them as well.
[Add Snapshot]

Closing the Loop - Listening and Talking Simultaneously

To be able to speak and listen to everybody in the chat room, you need to have both GME Send and GME Receive sounds (or events targeting these sounds, as shown in Figure 10). The easiest way to accomplish this is by dropping and playing both sounds in a Soundcaster session.

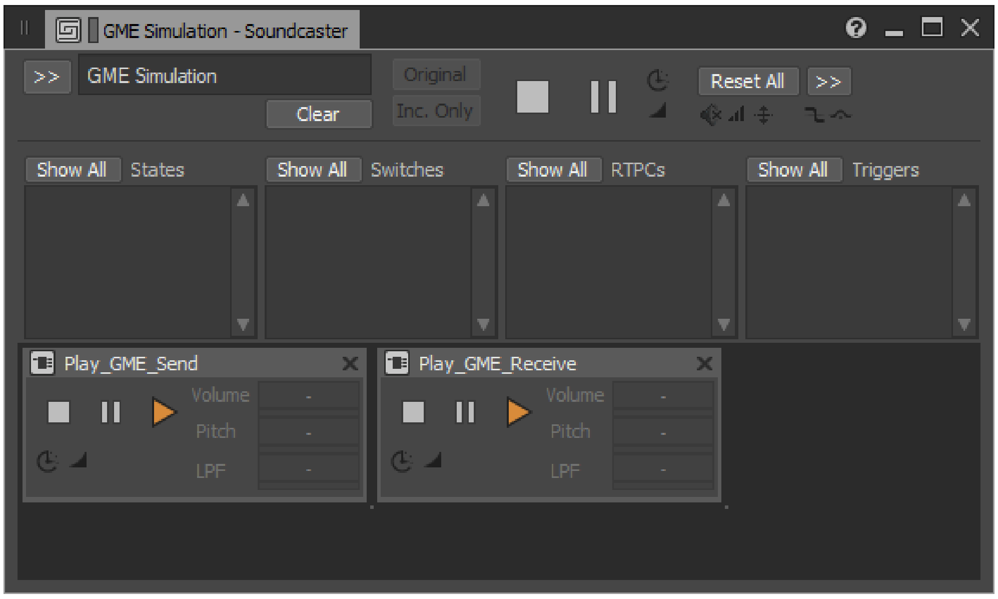


Figure 10 – Use the Soundcaster to send and receive audio concurrently

Note: For this setup to produce sound, a second player must use the same Room ID, but use a different User ID. This can be achieved locally by opening a second instance of the Wwise Authoring application and setting the User ID and Room ID inside the Tencent GME Session effect plug-in using the Effect Editor (see the "Authoring Session" section of the user interface in Figure 3).

A second instance of Wwise with a different User ID is necessary because, by design, sound is not routed from a GME Send to a GME Receive instance that belongs to the same User ID.

Creative Voice Chat Integration Possibilities

GME voice chat in Wwise has a much more elaborate feature set than a simple "conference call system" between players. The fact that GME is integrated in Wwise as input sources, similar to any pre-recorded audio assets, means that you can be creative in how you integrate voice chat into your game. Here are a few examples of how a developer may want to leverage typical Wwise features and apply them to voice chat:

- **Positioning**: It may be helpful in certain games to position other teammates around the listener when they speak. This way, you get an instant cue about where your teammates are in the environment, without seeing them. For this to work, select the GME Receive sound, click on the Positioning tab, and set the 3D Spatialization dropdown menu to "Position". Do not set any attenuation.
- **Distance Attenuation:** Similar to hearing your teammates around you, some games can leverage Wwise's distance attenuation in certain situations within the game. For example, if communication between players in the game emulates radio transmission, that game could create more interesting storytelling moments, by, for example, disabling radio communication in certain areas in the game, forcing a squad of players to stay within speaking distance of each other until they move to another area.
- **Environmental Reverb:** Similar to positioning and distance attenuation, processing the room's reverb in the player's chat can be used to enhance realism and make the experience more immersive.
- **"Voice leakage" across teams:** In many games, discussions are meant to remain private within teams when they use radio communication. However, one could create a situation where all communications are private, except when players are close enough to each other to hear each other, as would happen in real life.
- **Creative effects:** DSP effects can be added to voice chat, on the sending or receiving end, to customize the "voice profile" of certain players so they sound more fun, or to reflect certain game states. A game can be configured to add an effect (distortion, delay, flanger, etc.) to the voice of a teammate who just received damage or a special bonus.

These are just a few examples of how the combination of GME and the standard feature set of Wwise can be used to enhance certain game mechanics.

Prepping the Wwise Project to Run in your Game

Now that you have set the sending and receiving sounds, and you've configured the Tencent GME Session effect in your project with the necessary key and IDs, all that is left is to create a few events and package them into a SoundBank, so the game can use the service.

Create the Events

For each GME-related sound, corresponding Play and Stop events should be created in order to provide transport control to the game. Pause and Resume events can also be created, insofar as they make sense in the context of your game.

For voice receiving operations, the following four events may be created:

Event name	Description
Play_GME_Receive	To start receiving voice
Stop_GME_Receive	To stop receiving voice
Pause_GME_Receive	To pause receiving voice
Resume_GME_Receive	To resume receiving voice

For voice sending operations, the following four events may be created:

Event name	Description
Play_GME_Send	To start sending voice
Stop_GME_Send	To stop sending voice
Pause_GME_Send	To pause sending voice
Resume_GME_Send	To resume sending voice

Generate the SoundBank

Like any other object in your project, GME sounds and events (such as GME_Send and GME Receive) need to be added to a SoundBankfor them to be loaded and played by the game.

Using the GME for Wwise plug-in SDK

In its integration with the Wwise audio engine, GME provides support for all the mainstream platforms. After installing the GME for Wwise plug-in via the Wwise Launcher, the GME SDK is installed in the corresponding platform folders, under the Wwise default SDK directory. The naming rules of the GME SDK are consistent with those used by Wwise (%WWISEROOT%\SDK).

In this directory, the GME SDK provides support for the following target platforms. Static libraries and dynamic libraries corresponding to each platform are provided. The SDK directory also includes the GME related header files.

SDK Directory	**Description**
x64_	Windows: 64 bits
Win32_	Windows: 32 bits
Android_arm64-v8a	Android: ARM 64 bits, Android 4.2 or above
Android_armeabi-v7a	Android: ARM 32 bits, Android 4.2 or above
Android_x86	Android: x86 32 bits, Android 4.2 or above
Android_x86_64	Android: x86 64 bits, Android 4.2 or above
iOS	iOS: Device includes ARM 64 bits and 32 bits, simulator includes x86_64 and x86
Mac	macOS: x86 64 bits
Xbox One	Xbox One SDK
PS4	PS4 SDK
NX64	Nintendo Switch 64 bits
NX32	Nintendo Switch 32 bits
Include	GME for Wwise plug-in header file and GME API header file

Windows Platform Project Configuration

The GME SDK for Windows includes header files, libraries and dependency dynamic libraries:

- Header: TencentGMEFactory.h, TencentGMEPlugin.h
- Plug-in library: TencentGMEPlugin.lib
- Plug-in dynamic library: TencentGME.dll, gmesdk.dll

Copy the header files and libraries into the Visual Studio project directory. The dynamic libraries need to stay in the same directory as the executable file. On the Visual Studio project property page, adjust the settings for the Include directories and dependency libraries of the SDK.

Android Platform Project Configuration

The GME SDK for Android includes header files, libraries, dependency dynamic libraries, and a Java jar library package:

- Header: TencentGMEFactory.h, TencentGMEPlugin.h
- Plug-in library: libTencentGMEPlugin.a
- Dynamic libraries: libgmesdk.so, libTencentGME.so
- Jar class library: gmesdk.jar

Copy the header files, the dynamic libraries and the Java library to the Android project directory. The gmesdk.jar should be placed in the libs folder, which is generated after the Android native code is compiled. Figure 11 shows the reference structure of the libs folder:

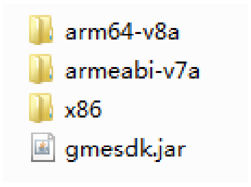

arm64-v8a
armeabi-v7a
x86
gmesdk.jar

Figure 11

In the Android.mk file, include the directories and dependency libraries of the GME SDK. Use the following pseudo Android.mk content for reference:

```
include $(CLEAR_VARS)
LOCAL_MODULE := libgmesdk
LOCAL_SRC_FILES :=
$(LOCAL_PATH)/../GME_SDK/Android_$(APP_ABI)/$(CONFIGURATION)/bin/GME_PLUGIN/libgmes
dk.so
include $(PREBUILT_SHARED_LIBRARY)

include $(CLEAR_VARS)
LOCAL_MODULE := TencentGME
LOCAL_SRC_FILES :=
$(LOCAL_PATH)/../GME_SDK/Android_$(APP_ABI)/$(CONFIGURATION)/lib/libTencentGME.a
include $(PREBUILT_STATIC_LIBRARY)
…
include ($CLEAR_VARS)
…
LOCAL_C_INCLUDES += $(LOCAL_PATH)/../GME_SDK/include
LOCAL_SHARED_LIBRARIES += libgmesdk
LOCAL_STATIC_LIBRARIES += TencentGME
…
include $(BUILD_SHARED_LIBRARY)
```

In the Android project AndroidManifest.xml file, add the following permissions:

```
<!-- INTERNET is needed to use communication -->
<uses-permission android:name=”android.permission.RECORD_AUDIO“ />
<uses-permission android:name=”android.permission.INTERNET“ />
<uses-permission android:name=”android.permission.BLUETOOTH“ />
<uses-permission android:name=”android.permission.WRITE_EXTERNAL_STORAGE“ />
```

From the location of the Application Java source code, call the following Java codes to pass Application Context to GME:

```
protected void onCreate(android.os.Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      com.tencent.GME.GMESDK.setApplicationContext(this);
}
```

If Jave code needs to be obfuscated, make sure to add the following configuration to keep the GME related classes unchanged.

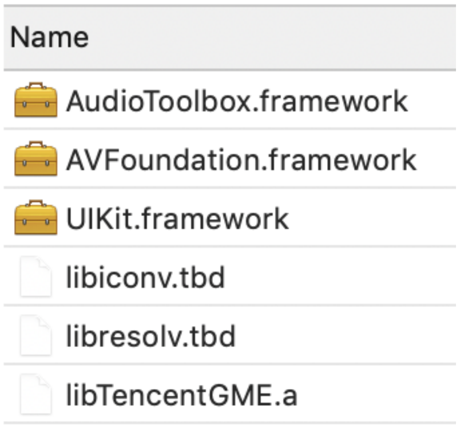
```
-dontwarn com.tencent.**
-keep class com.tencent.** { *;}
-keepclassmembers class com.tencent.**{*;}
```

iOS Platform Project Configuration

The GME SDK for iOS includes header files and plugin libraries.

- Header: TencentGMEFactory.h, TencentGMEPlugin.h
- Plug-in library: libTencentGMEPlugin.a, libGMESDK.a

Copy the header files and libraries to the Xcode project directory. From the Xcode project build settings, adjust the settings for the Include directories and dependency libraries of the SDK. Figure 12 (below) shows the dependency libraries to add:



Name
AudioToolbox.framework
AVFoundation.framework
UIKit.framework
libiconv.tbd
libresolv.tbd
libTencentGME.a

Figure 12

The Bitcode setting should be disabled in the Xcode project, as it's not supported by the GME SDK. To disable Bitcode, look for Bitcode under Targets -> Build Settings and set the corresponding option to NO.

The following permission on iOS is required:

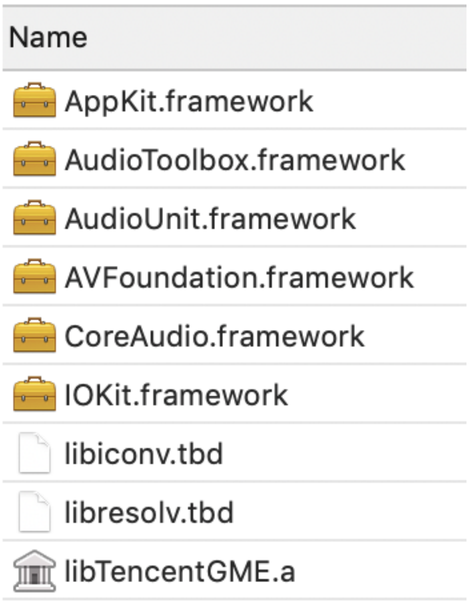
Key	Description
Microphone Usage Description	Allows microphone permission

macOS Platform Project Configuration

The GME SDK for MacOS includes header files and plugin libraries.

- Header: TencentGMEFactory.h, TencentGMEPlugin.h
- Plug-in library: libTencentGMEPlugin.a, libGMESDK.a
- Dynamic library: libTencentGME.dylib

Copy the header files and libraries to the Xcode project directory. From the Xcode project build settings, make the corresponding settings for the Include directories and dependency libraries of SDK. Figure 13 (below) shows the dependency libraries to add



Name
AppKit.framework
AudioToolbox.framework
AudioUnit.framework
AVFoundation.framework
CoreAudio.framework
IOKit.framework
libiconv.tbd
libresolv.tbd
libTencentGME.a

Figure 13

The following permission on Mac is required:

Key	Description
Microphone Usage Description	Allows microphone permission

Xbox One Platform Project Configuration

GME SDK for Xbox One includes header files, libraries and dependency dynamic libraries:

- Header: TencentGMEFactory.h, TencentGMEPlugin.h
- Plug-in library: TencentGMEPlugin.lib
- Plug-in dynamic library: TencentGME.dll, gmesdk.dll

Copy the header files and libraries into the Visual Studio project directory. The dynamic library needs to stay in the same directory as the executable file. In the Visual Studio project property page, adjust the settings for the Include directories and dependency libraries of the SDK.

The following capabilities should be added in the file of Package.appxmanifest

```
<Capabilities>
        <Capability Name=”internetClient” />
        <mx:Capability Name=”kinectAudio” />
        <mx:Capability Name=”kinectGamechat” />
</Capabilities>
```

PS4 Platform Project Configuration

GME SDK for PS4 includes header files and plugin libraries.

- Header: TencentGMEFactory.h, TencentGMEPlugin.h
- Plug-in library: TencentGMEPlugin.a, GME.a
- Dynamic library: TencentGME.prx

Those libraries are available for different versions of the PS4 SDK. You **must use the corresponding libraries**.

There are provided libraries built against the following PS4 SDKs:

- PS4 SDK 6.0: <Wwise>/PS4/SDK_6.0
- PS4 SDK 6.5: <Wwise>/PS4/SDK_6.5
- PS4 SDK 7.0: <Wwise>/PS4/SDK_7.0

Simply copy the content of the SDK_<version> needed into <Wwise>/SDK/PS4, overwriting existing files.

- Copy the header files and libraries into the Visual Studio project directory. In the Visual Studio project property page, adjust the settings for the Include directories and dependency libraries of the SDK. Specify the size of the heap in the project code, by adding the following lines :

```
unsigned int sceLibcHeapExtendedAlloc = 1;  /* Switch to dynamic allocation */
size_t sceLibcHeapSize = 100*1024*1024;     /* Must be larger than 100MB */
```

Switch Platform Project Configuration

GME SDK for Switch includes header files and plugin libraries.

- Header: TencentGMEFactory.h, TencentGMEPlugin.h
- Plug-in library: TencentGMEPlugin.a, libGMESDK.a

Those libraries are available for different versions of the Switch SDK. You **must use the corresponding libraries**.

There are provided libraries built against the following Switch SDKs:

- Switch SDK 8.3.0
  - <Wwise>/NX32/SDK_8.3.0
  - <Wwise>/NX64/SDK_8.3.0
- Switch SDK 9.3.1
  - <Wwise>/NX32/SDK_9.3.1
  - <Wwise>/NX64/SDK_9.3.1
- Switch SDK 9.3.1
  - <Wwise>/NX32/SDK_10.4.1
  - <Wwise>/NX64/SDK_10.4.1

Simply copy the content of the SDK_<version> needed into the respective <Wwise>/SDK/NX32 and <Wwise>/SDK/NX64 directories, overwriting existing files.

- Copy the header files and libraries into the Visual Studio project directory. In the Visual Studio project properties, under Linker > Input > Additional Dependencies, add the two static libraries.
- Add the following code to init the network module which is needed by GME

```
nn::nifm::Initialize();
nn::ssl::Initialize();
static nn::socket::ConfigDefaultWithMemory g_SocketConfigWithMemory;
```

```
nn::socket::Initialize(g_SocketConfigWithMemory);
CURLcode res = CURLE_OK;
res = curl_global_init(CURL_GLOBAL_DEFAULT);
```

If you are using the Communication module from Wwise, make sure that this initialization occurs before Wwise is initialized and that you disable the initialization of system libraries in the Communication initialization settings on Switch:

```
AkCommSettings commSettings;
AK::Comm::GetDefaultInitSettings( commSettings );
#ifdef AK_NX
       commSettings.bInitSystemLib = false;
#endif
AK::Comm::Init( commSettings );
```

Omitting to disable system library initialization will cause the Communication module to attempt a double initialization, which will trigger an exception.

GME for Wwise plug-in API

In order to integrate the GME voice service for Wwise, GME implements a total of four plug-ins. By themselves, the four Wwise plug-ins are not able to handle all of the information needed by GME, so additional API functions have been added to the GME for Wwise plug-in code.

The game code needs to call extra API functions in order to pass all of the necessary data to the GME for Wwise plug-ins. These API functions are declared in the header file TencentGMEPlugin.h.

The following section provides a description of the GME for Wwise plug-in API for Wwise.

General Settings

```
/**
* @brief Set the local user ID for the local user. Each GME user must have a unique identifier.
*           This function must be called before posting events sending or receiving to
*           GME servers.
* @param[in] userID The identifier of the local GME user.
*           The value is a 64-bit integer data and should be greater than 10000. It needs to be
*           converted into character type.
* @return Code to indicate if UserID is a valid value.
*           0: UserID is set successfully.
*           -1: NULL or empty string, -2: non-digit character, -3: not greater than 10000
*/
int GMEWWisePlugin_SetUserID(const char* userID);
```

```
/**
* @brief Set the GME chat room ID.
*           This function must be called before posting events sending or receiving to
*           GME servers.
*           Setting a new roomID will not affect already playing voices.
* @param[in] roomID Alphanumeric character string of up to 127 characters identifying a
*           GME chatting room.
* @return Code to indicate if UserID is a valid value.
*           0: RoomID is set successfully.
*           -1: NULL or empty string
*/
int GMEWWisePlugin_SetRoomID(const char* roomID);
```

```
/**
* @brief Set the level of logging for the GME SDK library.
*         For log files, default values are GMESDK_LOGLEVEL_INFO in debug builds and
*         GMESDK_LOGLEVEL_ERROR in release builds.
*         For console print, default values are GMESDK_LOGLEVEL_INFO in debug builds and
*         GMESDK_LOGLEVEL_NONE in release builds.
* @param[in] levelWrite Logging level for saved log file.
* @param[in] levelPrint Logging level for console printing.
* @sa GMEWWisePlugin_LogLevel
*/
void GMEWWisePlugin_SetLogLevel(
         GMEWWisePlugin_LogLevel levelWrite
         GMEWWisePlugin_LogLevel levelPrint
);
```

```
/**
* @brief Get GME specific messages.
* @param[out] localUTCTime Unix time of the message.
* @param[out] messageType Message type, ref to the following detailed description.
* @param[out] code Message code according to the message type, ref to the
*         following detailed description.
* @param[out] message1 Content of Message1
* @param[in] len1 Length of the message1 array.
* @param[out] message2 Content of Message2
* @param[in] len2 Length of the message2 array.
* @return Flag to indicate if there are any new messages.
*         0: There are new messages to be received.
*         1: No messages to be received.
*/
int GMEWWisePlugin_GetMessage(
         int* localUTCTime,
         int* messageType,
         int* code,
         char* message1,
         int len1,
         char* message2,
         int len2
);
```

```
/**
* @brief Suspend GME related activities.
*/
void GMEWWisePlugin_Pause();
```

```
/**
* @brief Resume GME related activities.
*/
void GMEWWisePlugin_Resume();
```

```
/**
* @brief Get version information of GME plugin lib
* @return GME plugin version.
*/
const char* GMEWWisePlugin_GetVersion();
```

Blocklist Settings

```
/**
* @brief Add a member specified by the targetID into the blocklist,
*           so as to mute the voice from this member.
* @param[in] targetID The identifier of the GME user.
*/
void GMEWWisePlugin_AddAudioBlockList(const char* targetID);
```

```
/**
* @brief Remove a member specified by the targetID from the blocklist,
*           so as to unmute the voice from this member.
* @param[in] targetID The identifier of the GME user.
*/
void GMEWWisePlugin_RemoveAudioBlockList(const char* targetID);
```

Receiving Stream Management

- Set the mapping between a Wwise game object ID and a user's GME user ID.
    - To receive an individual voice stream specified by userID, configure the gameObjectID and the userID to bind together.
    - To receive a mix of all voice streams within the current room, simply **do not** set a mapping (i.e., there is no need to call this function). If a mapping to a gameObjectID has been previously set, it can be cleared by using NULL or "" (empty string) as the userID.

```
/**
* @brief Set the mapping between a Wwise game object ID and a GME user ID
*         corresponding to a user. A mapping between gameObjectID and userID is used
*         by GME to decide whether a GME Receive plug-in instance should play a
*         specific userID or, if there is no mapping, the mix of all voice streams within
*         the currently set roomID.
* @param[in] gameObjectID The gameObjectID allocated in the game.
* @param[in] userID The user ID of the GME user. Passing NULL or "" (empty string)
*         clears a previously set mapping.
*/
void GMEWWisePlugin_ReceivePlugin_SetReceiveOpenIDWithGameObjectID(
      AkUInt64 gameObjectID,
      const char* userID
);
```

```
/**
* @brief Get the mapping between a Wwise game object ID and a GME user ID
*         corresponding to a user.
* @param[in] gameObjectID The gameObjectID allocated in the game.
* @param[out] userID User-allocated character buffer of maximum size specified by maxlen.
*         The userID mapped to the gameObjectID will be copied to the buffer if one exists.
* @param[in] maxlen The maximum size of the character buffer userID.
*/
void GMEWWisePlugin_ReceivePlugin_GetReceiveOpenIDWithGameObjectID(
      AkUInt64 gameObjectID,
      char* userID,
      int maxlen
);
```

```
/**
* @brief Get real-time level of a receiving voice stream specified by the targetID.
* @param[in] targetID The identifier of the GME user.
* @return Normalized level values from 0 to 100.
*/
int GMEWWisePlugin_GetAudioRecvStreamLevel(const char* targetID);
```

Sending Stream Management

```
/**
* @brief Enable or disable audio loopback. Loopback controls if sound related to the
*          specified game object is routed back to Wwise to be played on the local device.
*          The audio is always sent to the server, but will only be played locally if loopback
*          is enabled.
* @param[in] gameObjectID The gameObjectID allocated in the game.
* @param[in] enableLoopback Loopback flag value.
*/
void GMEWWisePlugin_SendPlugin_EnableLoopbackWithGameObjectID(
      AkUInt64 gameObjectID,
      bool enableLoopback
);
```

```
/**
* @brief Get the loopback status. Retrieve whether loopback is enabled or disabled for
*          a given game object. Loopback controls if sound posted on the specified game
*          object is routed back to Wwise to be played on the local device.
*          The audio is always sent to the server, but will only be played locally if loopback
*          is enabled.
* @param[in] gameObjectID The gameObjectID allocated in the game.
* @return The loopback status.
*/
bool GMEWWisePlugin_SendPlugin_GetEnableLoopbackWithGameObjectID(
      AkUInt64 gameObjectID
);
```

```
/**
* @brief Get real-time level of the sending voice stream.
* @return Normalized level values from 0 to 100.
*/
int GMEWWisePlugin_GetAudioSendStreamLevel();
```

Proximity Settings

The proximity settings allow fine control of the voice routing that occurs on the server. These settings are used to optimize bandwidth usage in cases where many users are in the same room, but only a small subset is heard by any given player. Games in the open world and battle royale genres are good examples of this scenario.

In most team-based games, where members of a team can only speak amongst themselves, using a room ID per team would be simpler than using the proximity settings.

By default, all users have a team ID of 0, which indicates they are not part of any team. In this case, the server does not apply any filtering when routing voices to this user.

```
/**
* @brief Set the team ID for the current player.
*           Players with the same team ID can always talk to each other,
*           regardless of the distance between them in any team mode.
*           A team is a group of players sharing the same team ID within the same room.
* @param[in] teamID Integer value identifying a team ID.
*           A team ID of 0 indicates “No team”, and will deactivate any filtering done on the
*           server when routing voices to this user.
* @sa GMEWWisePlugin_SetRangeAudioTeamMode
*/
void GMEWWisePlugin_SetRangeAudioTeamID(int teamID);
```

```
/**
* @brief Set the team mode to use for the current player.
* @param[in] teamMode The communication mode for the current player
*           * TEAMMODE_GLOBAL: Sound from players of any team is received if their position
*                     is within the receiving range.
*                     Sound from players in the same team is always received.
*           * TEAMMODE_TEAM: Sound is only received from teammates, i.e., players sharing
*                     the same team ID as the current player.
*                     Filtering based on the distance range is not applied.
*/
void GMEWWisePlugin_SetRangeAudioTeamMode(
          GMEWWisePlugin_TeamMode teamMode
);
```

```
/**
* @brief Set the hearing distance between players in global team mode.
*           In global team mode, sound from players with a different team ID
*           will be received when their position is within range.
*           The range is the 3D distance between player positions, as set by
*           GMEWWisePlugin_SetSelfPosition.
*           The filtering applied by this range is done on the server.
* @param[in] range Integer value represents the hearing distance.
* @sa GMEWWisePlugin_SetSelfPosition
*/
void GMEWWisePlugin_SetRangeAudioRecvRange(int range);
```

```
/**
* @brief Set the position of the current player.
* @param[in] positionX The X-coordinate of the current player
* @param[in] positionY The Y-coordinate of the current player
* @param[in] positionZ The Z-coordinate of the current player
* @sa GMEWWisePlugin_SetRangeAudioRecvRange
*/
void GMEWWisePlugin_SetSelfPosition(int positionX, int positionY, int positionZ);
```

Recording and Speech-To-Text

```
/**
* @brief Start recording speech, which will be recognized and uploaded to the server.
*         The maximum duration of speech is 60 seconds. Recording will be stopped
*         automatically if GMEWWisePlugin_StopRecording() is not called.
*         After calling GMEWWisePlugin_StopRecording(), periodically call
*         GMEWWisePlugin_GetMessage() to obtain the fileID of the recorded speech,
*         together with the speech recognition result in UTF-8 format.
*         The event type will be GMESDK_MESSAGETYPE_PTT_RECORD_COMPLETE.
* @param[in] speechLanguage Language tag for the speech.
*         Refer to the user manual for the language tags supported.
*         If the language tag is set to an empty string, only uploading will be done.
*         See https://intl.cloud.tencent.com/document/product/607/30260 for an
*         up-to-date list of supported languages.
* @return Return code:
*         0: Success.
*         Other values: Ref to Errorcode section in the user manual.
*/
int GMEWWisePlugin_StartRecording(const char* speechLanguage);
```

```
/**
* @brief Stop recording speech.
* @return Return code:
*         0: Success.
*         Other values: Ref to PTT Errorcode section in the user manual.
*/
int GMEWWisePlugin_StopRecording();
```

```
/**
* @brief Download and play the recording specified by fileID, which is stored on the server.
*         Periodically call GMEWWisePlugin_GetMessage() to obtain the result as a
*         GMESDK_MESSAGETYPE_PTT_PLAYOUT_COMPLETE event.
* @param[in] fileid fileID of the recording to be downloaded and played.
* @return Return code:
*         0: Success.
*         Other values: Ref to PTT Errorcode section in the user manual.
*/
int GMEWWisePlugin_PlayRecordFile(const char* fileid);
```

```
/**
* @brief Stop playing speech.
* @return Return code:
*         0: Success.
*         Other values: Ref to PTT Errorcode section in the user manual.
*/
int GMEWWisePlugin_StopPlayFile();
```

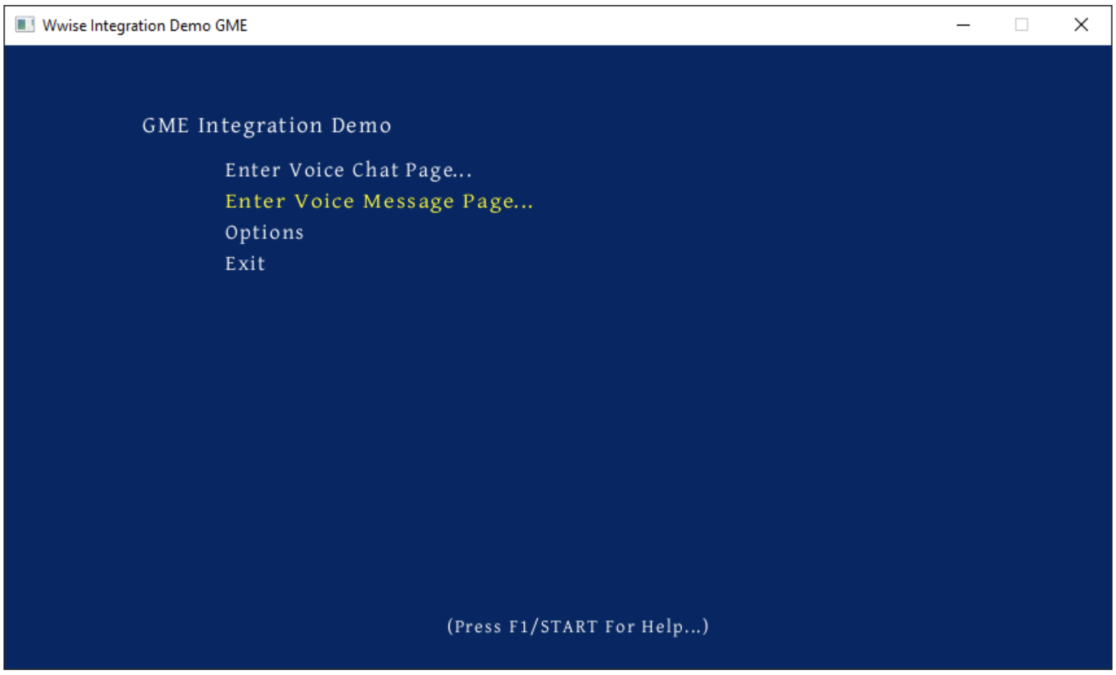
```
/**
* @brief Recognize the speech specified by the fileID.
*           Periodically call GMEWWisePlugin_GetMessage() to obtain the result as a
*           GMESDK_MESSAGETYPE_PTT_ASR_COMPLETE event.
*           The result will be in UTF-8 format.
* @param[in] fileid fileID of the speech to be recognized.
* @param[in] speechLanguage Language tag for the speech.
*           Ref to the user manual all for the language tags supported.
* @param[in] translateLanguage Not supported now.
* @return Return code:
*           0: Success.
*           Other values: Ref to PTT Errorcode section in the user manual.
*/
int GMEWWisePlugin_SpeechToText(
            const char* fileid,
            const char* speechLanguage,
            const char* translateLanguage
);
```

```
/**
* @brief Get speech duration specified by the fileID.
*           Periodically call GMEWWisePlugin_GetMessage() to obtain the result as a
*           GMESDK_MESSAGETYPE_PTT_GETVOICEDUARTION_COMPLETE event.
* @param[in] fileid fileID of the speech.
* @return Return code:
*           0: Success.
*           Other values: Ref to PTT Errorcode section in the user manual.
*/
int GMEWWisePlugin_GetVoiceFileDuration(const char* fileid);
```

GME Integration Demo

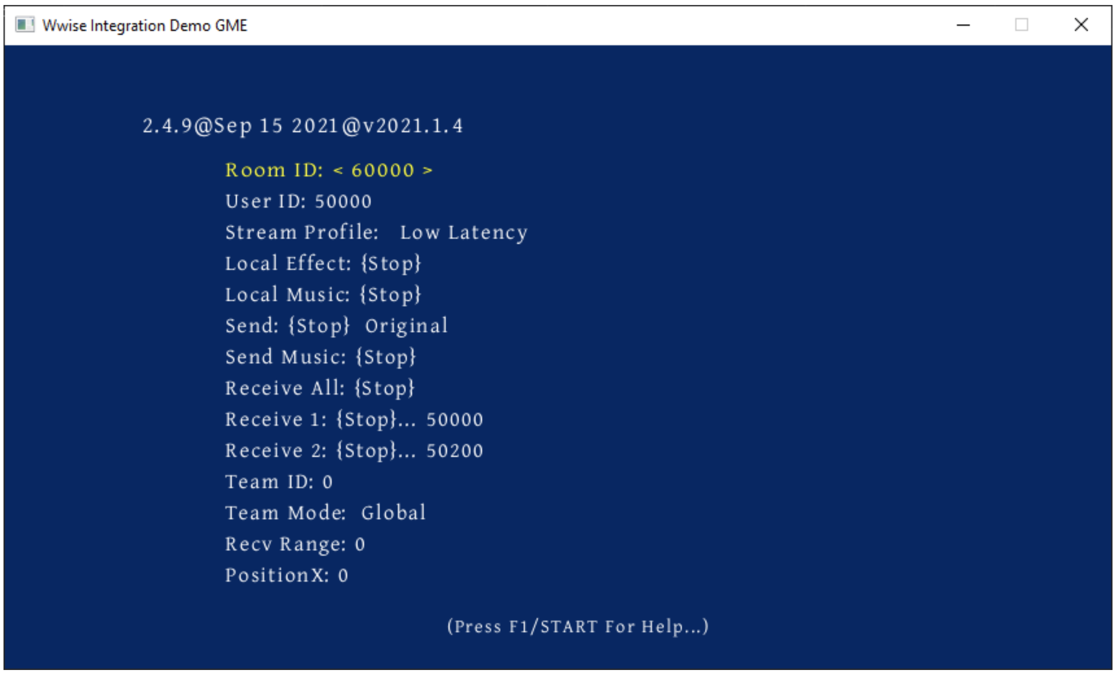
The GME demo is implemented using a copy of the Wwise Integration Demo.

The screenshots below are the two main user interfaces of the GME demo running on Windows. In this document, only the controls for the Windows platform are described. However, controls for other platforms are similar.



```
Wwise Integration Demo GME

GME Integration Demo
      Enter Voice Chat Page...
      Enter Voice Message Page...
      Options
      Exit

                    (Press F1/START For Help...)
```

Figure 13 - Landing page, showing general information about the demo build



```
Wwise Integration Demo GME

2.4.9@Sep 15 2021@v2021.1.4
      Room ID: < 60000 >
      User ID: 50000
      Stream Profile:  Low Latency
      Local Effect: {Stop}
      Local Music: {Stop}
      Send: {Stop}  Original
      Send Music: {Stop}
      Receive All: {Stop}
      Receive 1: {Stop}... 50000
      Receive 2: {Stop}... 50200
      Team ID: 0
      Team Mode:  Global
      Recv Range: 0
      PositionX: 0

                    (Press F1/START For Help...)
```

Figure 14 - Main page of the demo, allowing setting the Room ID and User ID

Building the GME Wwise Integration Demo

After installing the GME for Wwise plug-in via the Wwise Launcher, the Wwise Integration Demo can be found using the following path:

```
%WWISEROOT%\SDK\samples\Plugins\TencentGME\IntegrationDemo
```

Platform-specific implementation files are provided in dedicated directories. Details for building the demo for each platform are provided below.

Windows

The following solutions are provided:

- Visual Studio 2015 (*IntegrationDemoGME_Windows_vc140.sln*)
- Visual Studio 2017 (*IntegrationDemoGME_Windows_vc150.sln*)
- Visual Studio 2019 (*IntegrationDemoGME_Windows_vc160.sln*)

You may need to change the Windows SDK version you wish to target with one you have installed. To do so, in Visual Studio, right-click on the project in the Solution Explorer and in the “General” section, set the “Windows SDK Version” to your available version.
The minimum required version is 10.0.17763.0.

Mac and iOS

Xcode projects (*IntegrationDemoGME.Xcodeproj*) are provided in the Mac and iOS directories.

Android

A script “*Build.cmd*” is provided in the Android directory. Its usage is as follows:

```
Build.cmd [armeabi-v7a|x86|arm64-v8a|x86_64] [Debug|Profile|Release]
```

The script must be run in the same working directory.
Note that the output directory for this sample is:

```
%WWISEROOT%\SDK\Android_<ARCH>\<CONFIG>\bin\IntegrationDemoGME
```

PS4

A Visual Studio 2015 solution (*IntegrationDemoGME_PS4_vc140.sln*) is provided in the PS4 directory.

Xbox One

A Visual Studio 2015 solution (*IntegrationDemoGME_XboxOne_vc140.sln*) is provided in the XboxOne directory.

Switch

A Visual Studio 2015 solution (*IntegrationDemoGME_NX_vc140.sln*) is provided in the NX directory.

GME Demo Keyboard Controls

The following lists the various commands that can be assigned to keyboard shortcuts:

- **Up/Down or W/S:** Move vertically between items on the page.
- **Left/Right or A/D:** Used to change values for items displayed inside angle brackets:
  - Room ID: Set the Room ID. The step size is 100.
  - User ID: Set the User ID. Each user inside the same room should have a unique User ID.
  - Send: Set the DSP effects (3rd party reverb and voice effects from Wwise) used to process the voice being sent.
  - Send Route: Enable or disable the loopback for the voice being sent.
  - Receive 1: Specify the user ID that you want to receive. The default step size is 100.
  - Receive 2: Specify the user ID that you want to receive. The default step size is 100.
  - Team ID: Specify the team ID to which the user ID belongs.
  - Team Mode: Set the team mode between Global and Team.
  - Recv Range: Set the receiving range for use in the global mode of the proximity settings. The default step size is 100.
  - PositionX: Set the position of the user on a single axis. The default step size is 30.
- **Enter:** Activates the selected item or enters the settings subpage of the selected item. On game consoles, use the A button instead:
  - Enter Main Page: Enters the main GME function (send and receive) page, loading the GME sound bank.
  - Send: Enters the route settings subpage for the voice being sent.
  - Send Music: Enters the route settings subpage for the music being sent.
  - Receive 1: Enters the position settings subpage for the voice being received.
  - Receive 2: Enters the position settings subpage for the voice being received.
- **Space or Esc:** Go back to the parent page. On game consoles, use the B button instead.
- **Q:** Start or stop the selected GME operation. On game consoles, use the X button instead.
  - Local Effect: Start or stop the game object sound playback.
  - Local Music: Start or stop the BGM playback.
  - Send: Start or stop sending the voice captured by the microphone to the chat room.
  - Send Music: Start or stop sending the music to the chat room.
  - Receive All: Start or stop receiving all the voices from the chat room.
  - Receive 1: Start or stop receiving the voice of the specified ID from the chat room.
  - Receive 2: Start or stop receiving the voice of the specified ID from the chat room.
- **E:** Pause or resume the selected operation. On game consoles, use the Y button instead.
  - Local Effect: Pause or resume the game object sound playback.
  - Local Music: Pause or resume the BGM playback.
  - Send: Pause or resume sending the voice captured by the microphone to the chat room.
  - Send Music: Pause or resume sending the music to the chat room.
  - Receive All: Pause or resume receiving all the voice from the chat room.
  - Receive 1: Pause or resume receiving the voice of the specified ID from the chat room.
  - Receive 2: Pause or resume receiving the voice of the specified ID from the chat room.
- **Arrow keys:** Set 3D positioning of the voice sound object using Wwise 3D spatialization.

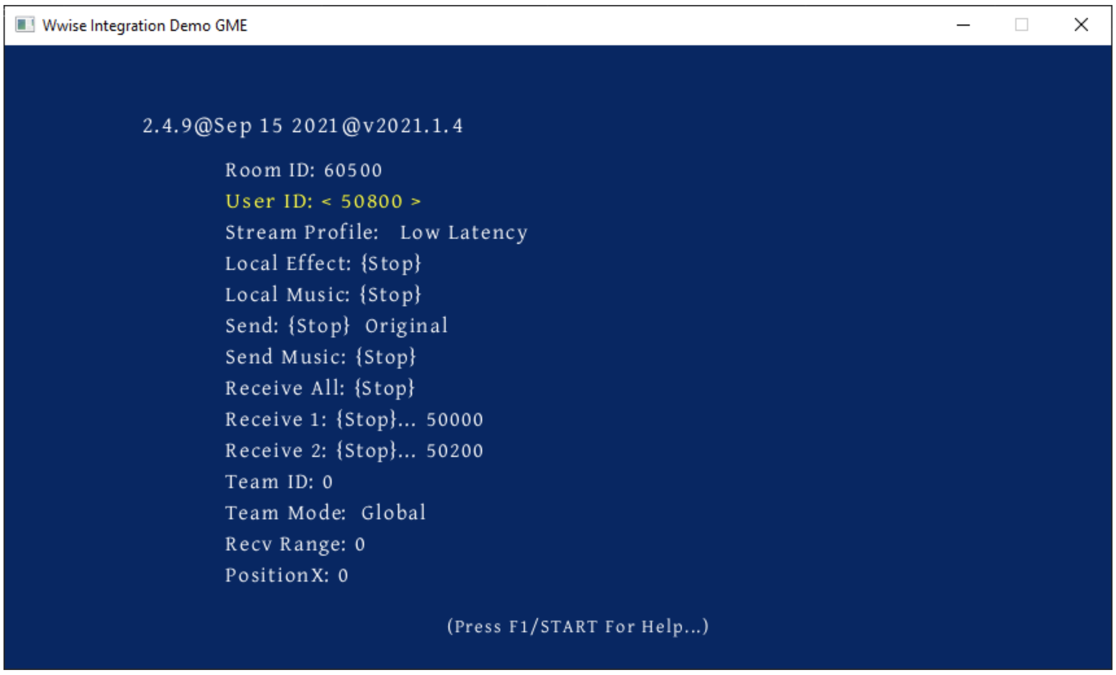
Sending Voice

To send the voice stream to the server, follow these steps:

1. Enter the Main page from the GME Demo starting screen.

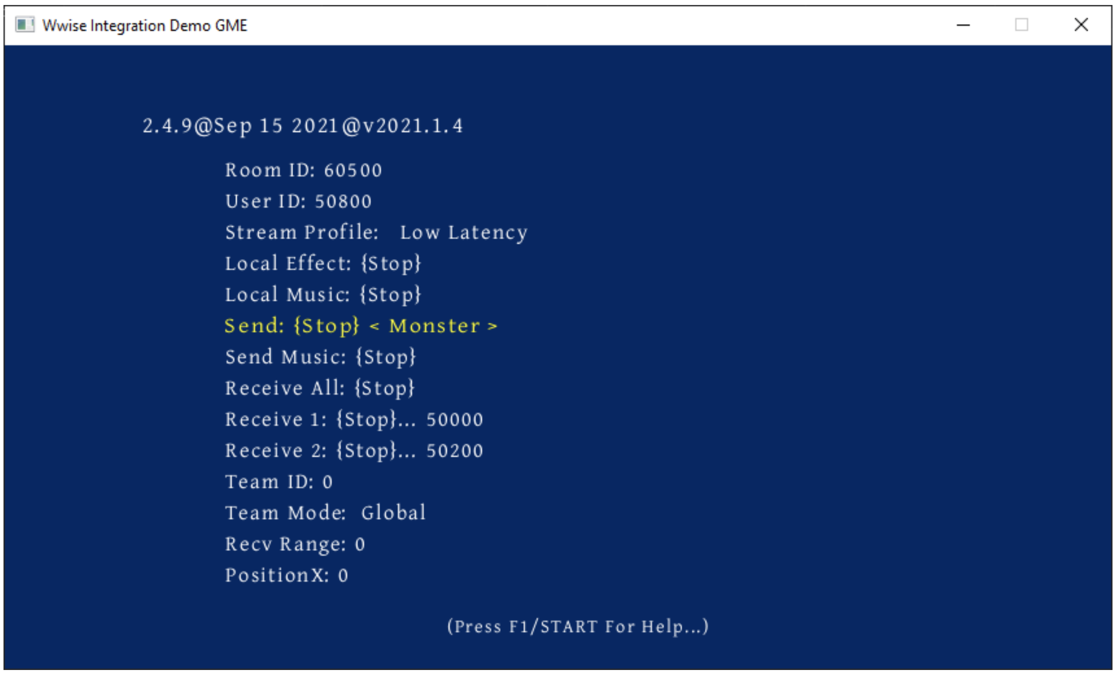
2. Set the Room ID and the User ID, as shown in Figure 16.
3. To send...
   a. the **microphone stream**, select the "Send" option.
      - Select the effect to be applied to the captured voice. The default is Original, which means no effects will be processed. Figure 17 shows a selection of the "Monster" effect, which is a pitch shift to a lower pitch.

   b. a **music stream**, select the "Send Music" option.
4. Press Q to start or stop sending the voice.
5. Press E to pause or resume sending the voice.



```
2.4.9@Sep 15 2021@v2021.1.4

    Room ID: 60500
    User ID: < 50800 >
    Stream Profile:  Low Latency
    Local Effect: {Stop}
    Local Music: {Stop}
    Send: {Stop}  Original
    Send Music: {Stop}
    Receive All: {Stop}
    Receive 1: {Stop}... 50000
    Receive 2: {Stop}... 50200
    Team ID: 0
    Team Mode:  Global
    Recv Range: 0
    PositionX: 0

                    (Press F1/START For Help...)
```

Figure 15



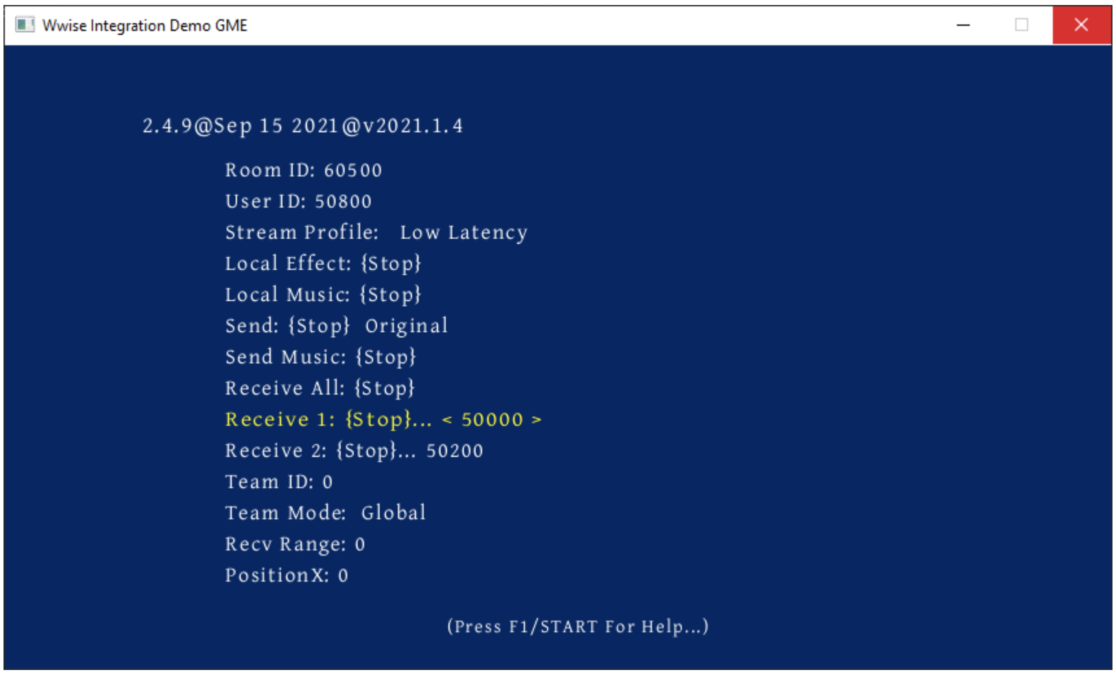
```
2.4.9@Sep 15 2021@v2021.1.4

    Room ID: 60500
    User ID: 50800
    Stream Profile:  Low Latency
    Local Effect: {Stop}
    Local Music: {Stop}
    Send: {Stop} < Monster >
    Send Music: {Stop}
    Receive All: {Stop}
    Receive 1: {Stop}... 50000
    Receive 2: {Stop}... 50200
    Team ID: 0
    Team Mode:  Global
    Recv Range: 0
    PositionX: 0

                    (Press F1/START For Help...)
```

Figure 16

Receiving Voice

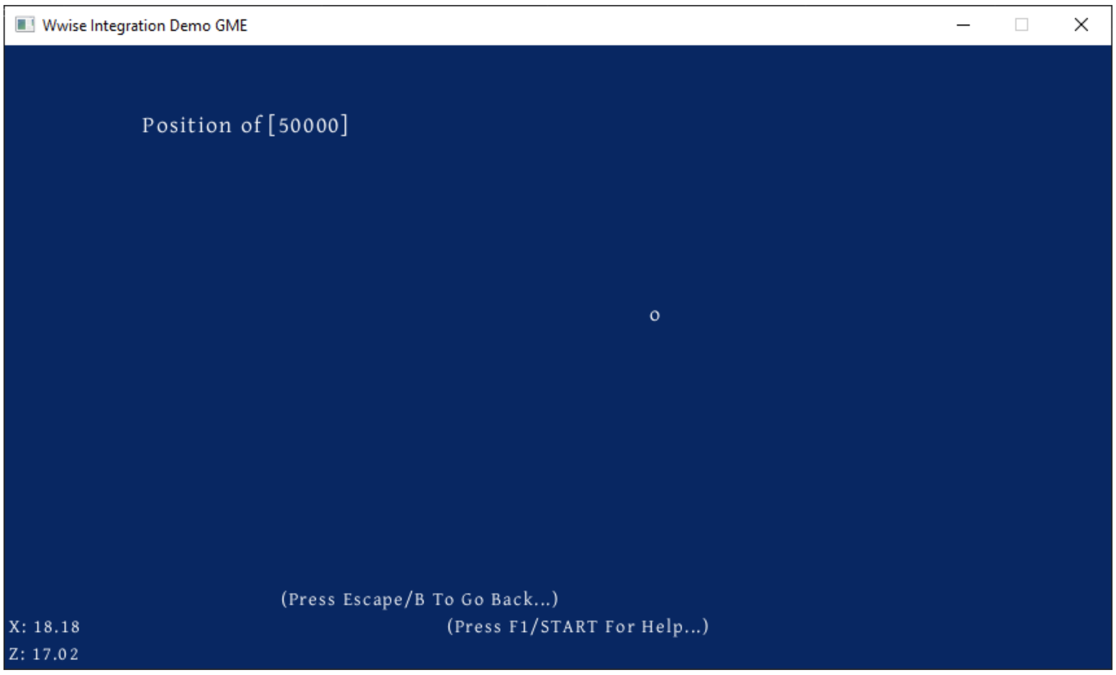
To receive a mix (*Receive All*) or individual voices (*Receive 1* and *Receive 2*), follow these steps:

1. Enter the Main page from the GME Demo starting screen.
2. Set the Room ID and the User ID.
3. To receive...
    a. **all voices from the chat room**, navigate to Receive All.
    b. the **voice of the specified user ID**, navigate to Receive 1 or Receive 2.
        i. You can enable 3D spatialization for the received voice by entering the position subpage using Enter on the "Receive 1" or "Receive 2" option.
        ii. Use the arrow keys to move the game object's position, as shown in Figure 19.
4. Click Q to start or stop receiving the voice.
5. Click E to pause or resume receiving the voice.



```
Wwise Integration Demo GME

2.4.9@Sep 15 2021@v2021.1.4

        Room ID: 60500
        User ID: 50800
        Stream Profile:  Low Latency
        Local Effect: {Stop}
        Local Music: {Stop}
        Send: {Stop}  Original
        Send Music: {Stop}
        Receive All: {Stop}
        Receive 1: {Stop}... < 50000 >
        Receive 2: {Stop}... 50200
        Team ID: 0
        Team Mode:  Global
        Recv Range: 0
        PositionX: 0

                        (Press F1/START For Help...)
```

Figure 17



Position of [50000]

o

(Press Escape/B To Go Back...)

X: 18.18

(Press F1/START For Help...)

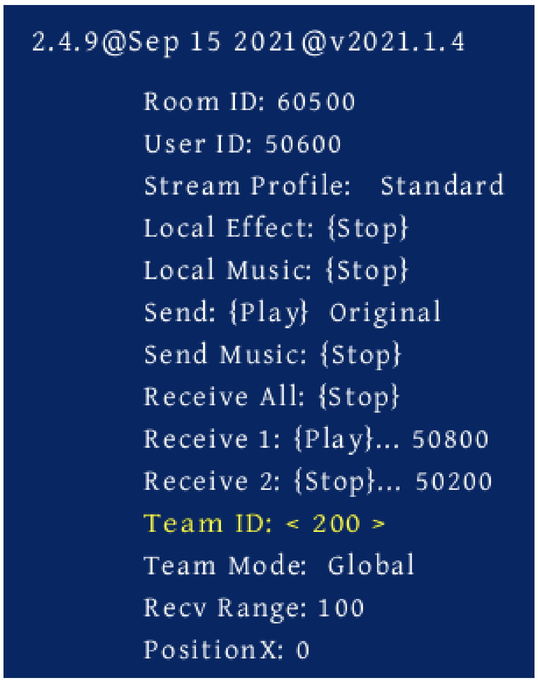
Z: 17.02

Figure 18

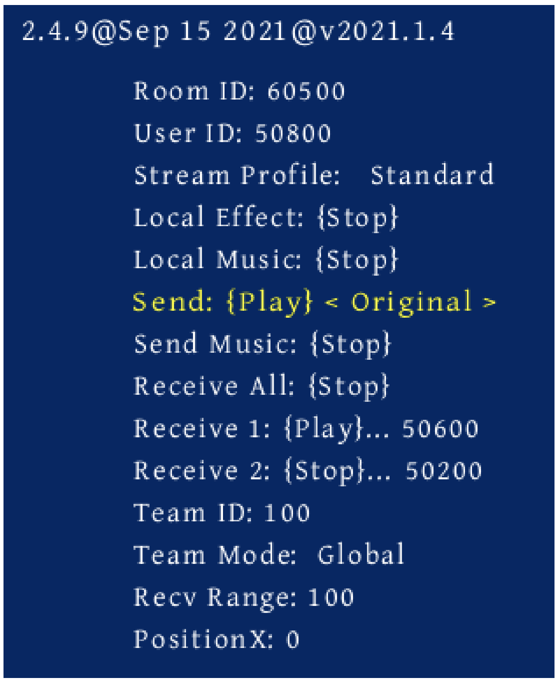
Using Proximity Settings

All the settings corresponding to the proximity feature are available as the last options on the Main page. To use the proximity settings, follow these steps (refer to Figure 20 for the final setup):

1. Enter the Main page from the GME Demo starting screen.
2. Set the Room ID and the User ID.
3. Set the proximity settings as follows:
   a. Team Mode to Global.
   b. Recv Range to 100.
   c. PositionX to 0.
4. In a **second instance** of the Integration Demo: use a different User ID and the same Room ID and proximity settings.
5. In the **first instance**, set the Team ID to 100 (it must be non-zero).
6. In the **second instance**, set the Team ID to 200 (it must be non-zero).
7. Set the Receive 1 ID in each of the Integration Demo instances to the User ID of the **other instance**. Press Q to play the Receive 1 voice.
8. Navigate to the "Send" option and play it by pressing Q.
9. Change the value of PositionX in either of the instances to create a **position difference** of 100 units. When this occurs, sound should stop being routed between the two instances.



```
2.4.9@Sep 15 2021@v2021.1.4

        Room ID: 60500
        User ID: 50600
        Stream Profile:  Standard
        Local Effect: {Stop}
        Local Music: {Stop}
        Send: {Play}  Original
        Send Music: {Stop}
        Receive All: {Stop}
        Receive 1: {Play}... 50800
        Receive 2: {Stop}... 50200
        Team ID: < 200 >
        Team Mode:  Global
        Recv Range: 100
        PositionX: 0
```



```
2.4.9@Sep 15 2021@v2021.1.4

        Room ID: 60500
        User ID: 50800
        Stream Profile:  Standard
        Local Effect: {Stop}
        Local Music: {Stop}
        Send: {Play} < Original >
        Send Music: {Stop}
        Receive All: {Stop}
        Receive 1: {Play}... 50600
        Receive 2: {Stop}... 50200
        Team ID: 100
        Team Mode:  Global
        Recv Range: 100
        PositionX: 0
```

Figure 19: Proximity settings applied to two instances of Integration Demo GME

Unity Integration

Prerequisite

Before integrating GME functionality into a game using Unity and Wwise, the Wwise integration must first be completed by the Wwise launcher. You can find more information on integrating Wwise to Unity, by following this link:
https://www.audiokinetic.com/library/edge/?source=Unity&id=main.html

In the Wwise project, create new send and receive points, as described in the GME Wwise Integration Demo section of this document , and add the relevant GME events to a SoundBank.

SDK Package

The GME integration SDK is a collection of resources, which include prebuilt plug-in libraries for each platform, the GME C# layer API, and reference build scripts. To help with the GME integration, a sample game is available to download to help you understand how to use GME in a Unity project. After installing the Tencent GME for Wwise plug-in through the Wwise Launcher, the sample game is found in <WWISE_INSTALL_PATH>/SDK/plugins/TencentGME/Unity/GMEWwiseDemo. Note that it is necessary to integrate Wwise into the Unity project, using the Wwise Launcher.

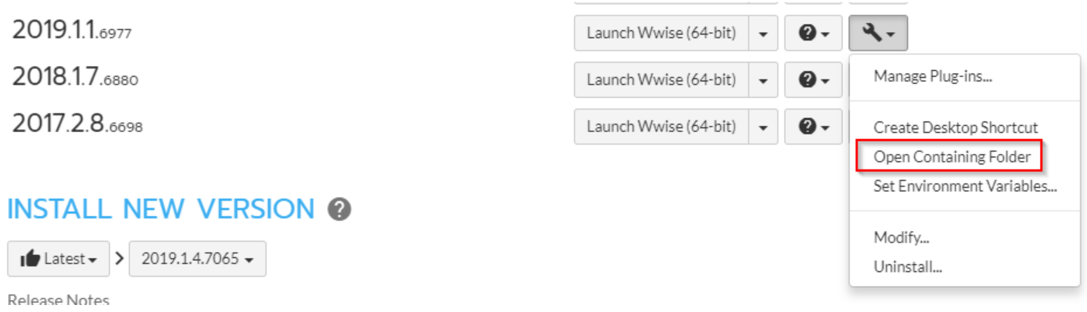
The GME SDK package for Unity is installed when installing the Tencent GME for Wwise plug-in from the Wwise Launcher. The SDK has been tested on Windows, Android, iOS, macOS, Xbox One, PS4, and Switch.

Integration into an existing Unity project

After installing the Tencent GME for Wwise plug-in from the Wwise Launcher, integrate Wwise into your Unity project, using the Unity tab in the Wwise Launcher. If you have already integrated Wwise to your Unity project, use the Wwise Launcher to modify your Unity integration. This copies the newly installed GME plugins to your Unity project.

To integrate the GME plug-in for Wwise into your Wwise project, refer to the steps in the Authoring Tool section of this document.

Once the integration process is complete , navigate to the Wwise installation folder:



From there, copy the folder:

```
<Wwise>/SDK/plugins/TencentGME/Unity/GMEWwiseDemo/Assets/GMESDK
```

into the “Assets” folder of your Unity project. This folder contains the C# layer API files and the scripts required to build your game for specific platforms.

The various native plugins for each platform supported by the GME integration should have been copied in the platform's DSP folder. For all platforms except iOS and Switch, the plug-in is distributed as a dynamic library. The following table shows the expected paths for each platform. Ensure the various GME libraries reside at the appropriate locations.

Platform	GME libraries	Path
Windows	TencentGME.dll gmesdk.dll	Assets\Wwise\Deployment\Plugins\Windows\x86_64\DSP Assets\Wwise\Deployment\Plugins\Windows\x86\DSP
Android	libTencentGME.so	Assets\Wwise\Deployment\Plugins\Android\arm64-v8a\DSP Assets\Wwise\Deployment\Plugins\Android\armeabi-v7a\DSP Assets\Wwise\Deployment\Plugins\Android\x86\DSP
iOS	libGMESDK.a libTencentGMEPlugin.a TencentGMEPluginFactory.h TencentGMEPlugin.h	Assets\Wwise\Deployment\Plugins\iOS\DSP
macOS	libTencentGME.bundle	Assets\Wwise\Deployment\Plugins\Mac\DSP
Xbox One	TencentGME.dll gmesdk.dll	Assets\Wwise\Deployment\Plugins\XboxOne\DSP
PS4	TencentGME.prx TencentGME_stub.a TencentGME_stub_weak.a	Assets\Wwise\Deployment\Plugins\PS4\DSP
Switch	libGMESDK.a libTencentGMEPlugin.a TencentGMEPluginFactory.h TencentGMEPlugin.h	Assets\Wwise\Deployment\Plugins\Switch\NX64\DSP

The Wwise-GME Unity integration requires specific settings for Android and iOS:

Android: in the Assets\GMESDK\Plugins\Android folder, there are files specifically required for the Android platform. The folder structure hierarchy must be kept as is in order to pack everything needed into the final APK. Below are some important files.

- AndroidManifest.xml: Adds the related permissions for GME. This file is combined with the final Application AndroidManifest.xml generated by Unity.
- gmesdk.jar: Dependency Java class library for GME. This library must be packed in the final APK.
- libs folder: Contains the gmesdk.so dependency. These libraries must be packed in the final APK.

As GME native code needs to call the Java API, the Application context should be passed before the initialization of GME for Wwise plug-in. It is recommended to do this as early as possible in the game code to avoid a null pointer error during the GME for Wwise plug-in initialization. The sample code for doing this can be found in Assets\Scripts\GMEInit.cs:

```
#if UNITY_ANDROID && !UNITY_EDITOR
Debug.Log("GME.setApplicationContext begin");
AndroidJavaClass jcUnityPlayer =
       new AndroidJavaClass("com.unity3d.player.UnityPlayer");
AndroidJavaObject curActivity =
       jcUnityPlayer.GetStatic<AndroidJavaObject>("currentActivity");

AndroidJavaClass jcGMESDK = new AndroidJavaClass("com.tencent.GME.GMESDK");
jcGMESDK.CallStatic<int>("setApplicationContext", curActivity);
Debug.Log("GME.setApplicationContext end");
```

```
#endif
```

iOS: When building for iOS, Unity generates an Xcode project. In order to link the GME for Wwise plug-in library, some settings must be changed in the Xcode project. The reference code for doing these configurations can be found in the following file: Assets/Editor/iosProjectScript.cs.

The settings can be divided into three categories, the pseudo code is as follows:

- Library dependency: *proj.AddFrameworkToProject (target, "libresolv.9.tbd", false);*
- Disable Bitcode: *proj.SetBuildProperty (target, "ENABLE_BITCODE", "NO");*
- Apply permission: rootDict.SetString ("NSMicrophoneUsageDescription", "GMEDemo").

For the other platforms, there are no specific settings to configure.

GME C# API

There are some specific APIs that must be called before using the GME service. The C# layer APIs exposed are defined in the file: Assets\GMESDK\GMEWWisePluginNative.cs. These APIs are called by the game's C# code directly. For a description of each function, please refer to the GME for Wwise plug-in API section of this document.

```
public static extern IntPtr GMEWWisePlugin_GetVersion();
public static extern void GMEWWisePlugin_SetUserID(string userID);
public static extern void GMEWWisePlugin_SetRoomID(string roomID);

public static extern void GMEWWisePlugin_ReceivePlugin_SetReceiveOpenIDWithGameObjectID(
    ulong gameObjectID, string userID);
public static extern void GMEWWisePlugin_ReceivePlugin_GetReceiveOpenIDWithGameObjectID(
    ulong gameObjectID, StringBuilder userID);
public static extern void GMEWWisePlugin_SendPlugin_EnableLoopbackWithGameObjectID(
    ulong gameObjectID, bool enableLoopback);
public static extern bool GMEWWisePlugin_SendPlugin_GetEnableLoopbackWithGameObjectID(
    ulong gameObjectID);

public static extern int GMEWWisePlugin_GetMessage(
    ref int localUTCTime, ref int messageType, ref int code,
    byte[] message1, int len1
    byte[] message2, int len2);
public static extern void GMEWWisePlugin_SetLogLevel(int logLevel);

public static extern void GMEWWisePlugin_Pause();
public static extern void GMEWWisePlugin_Resume();

public static extern void GMEWWisePlugin_AddAudioBlockList(string targetID);
public static extern void GMEWWisePlugin_RemoveAudioBlockList(string targetID);

public static extern void GMEWWisePlugin_SetRangeAudioTeamID(int teamID);
public static extern void GMEWWisePlugin_SetRangeAudioTeamMode(int teamMode);
public static extern void GMEWWisePlugin_SetRangeAudioRecvRange(int range);
public static extern void GMEWWisePlugin_SetSelfPosition(
    int positionX, int positionY, int positionZ);

public static extern int GMEWWisePlugin_StartRecording(string speechLanguage);
public static extern int GMEWWisePlugin_StopRecording();
public static extern int GMEWWisePlugin_PlayRecordFile(string fileid);
public static extern int GMEWWisePlugin_StopPlayFile();
public static extern int GMEWWisePlugin_SpeechToText(
    string fileid, string speechLanguage, string translateLanguage);
public static extern int GMEWWisePlugin_GetVoiceFileDuration(string fileid);
```

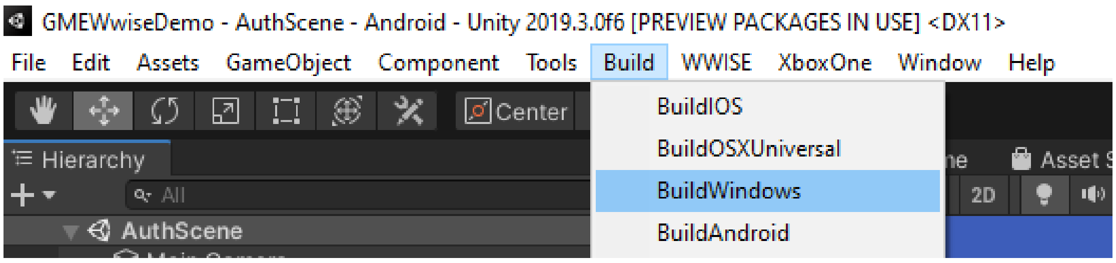
Public functions exposed in GMEWWisePluginNative.cs

Build and Run

In addition to the Tencent GME for Wwise plug-in library, a dynamic library (gmesdk.dll or libgmesdk.so) is required on some platforms. This library is automatically included in the packaging when using the build script (Assets\GMESDK\Editor\BuildScript.cs) by implementing the OnPreprocessBuild and OnPostprocessBuild functions. These will be called during the Unity build process.

This script is also responsible for adding the static libraries as link dependencies for iOS and Switch.

You can use the script through the Build menu in the top bar:



Reference Sample Game

Demo game setup

The GME Unity SDK is bundled with a sample game to help you understand how to use GME in a Unity project. It is located in:

```
<WWISE_INSTALL_PATH>/SDK/plugins/TencentGME/Unity/GMEWwiseDem
o
```

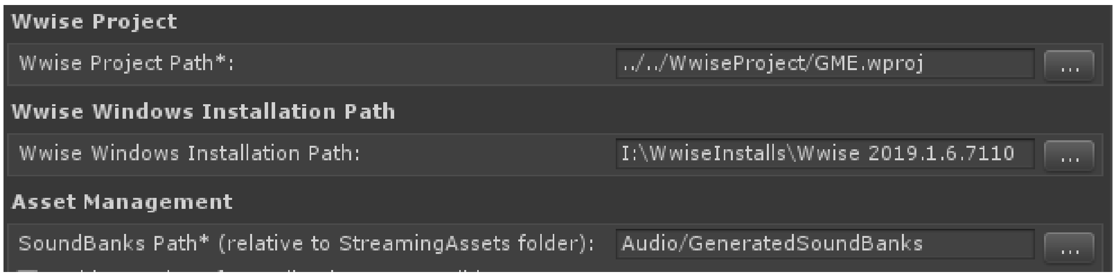
Make a copy of the project to a location where your user has write access, e.g., C:/Users/<YourUser>. The demo game requires Wwise to be integrated using the Launcher to copy the Wwise SDK with the Tencent GME for Wwise plug-in. Refer to the Wwise Unity Integration documentation.

Note: The Wwise project provided was created with Unity 2018.4.31f1, which is the oldest supported version of Wwise. You will need to first upgrade the project to your Unity version prior to integrating Wwise in it. The upgrade has been tested with Unity 2021.1.10f1.

First, copy the Wwise project provided for the GME Integration Demo Wwise project next to the demo game. It is located in:

```
<WWISE_INSTALL_PATH>/SDK/samples/Plugins/TencentGME/
IntegrationDemo/WwiseProject
```

Next, set the paths to the Wwise project and your Wwise installation in the Wwise settings (Edit > Wwise Settings…). The settings should look something like this:



Wwise Project

Wwise Project Path*: ../../WwiseProject/GME.wproj

Wwise Windows Installation Path

Wwise Windows Installation Path: I:\WwiseInstalls\Wwise 2019.1.6.7110

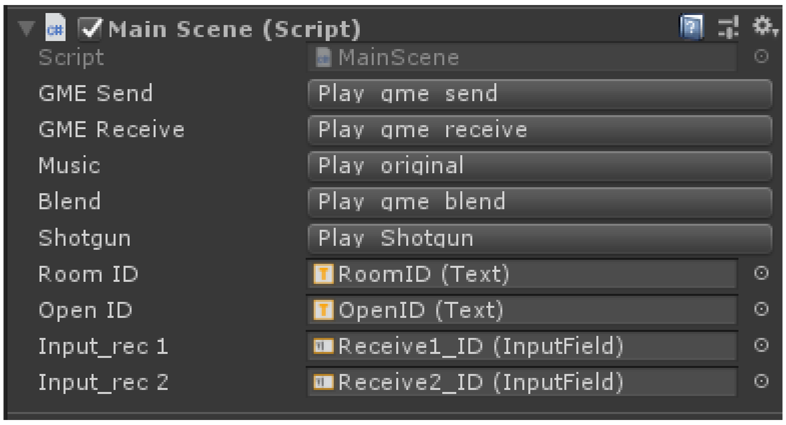
Asset Management

SoundBanks Path* (relative to StreamingAssets folder): Audio/GeneratedSoundBanks

Then, copy the GeneratedSoundBanks directory from the GME Integration Demo Wwise project to the specified location in the SoundBanks Path field. The previous figure assumed the following path as the location of the banks:

<UNITY_PROJECT>/Assets/StreamingAssets/Audio

Finally, set events and SoundBank elements from the Wwise project into Unity components. The Main Scene has a set of checkboxes for posting events. They are controlled by the Main Scene script component on the Canvas object: set them as follows:

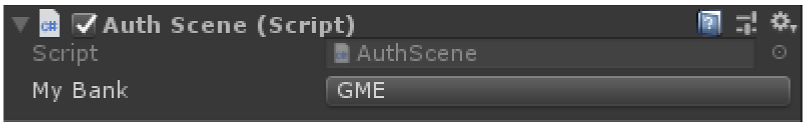


Main Scene (Script)

Script	MainScene
GME Send	Play_gme_send
GME Receive	Play_gme_receive
Music	Play_original
Blend	Play_gme_blend
Shotgun	Play_Shotgun
Room ID	RoomID (Text)
Open ID	OpenID (Text)
Input_rec 1	Receive1_ID (InputField)
Input_rec 2	Receive2_ID (InputField)

Note: If the events do not appear, the Wwise project may not have been read by Unity yet. You can force a refresh by clicking the "Refresh Project" button in the Wwise Picker window (Window > Wwise Picker).

The WwiseGlobal object in the AuthScene has a script allowing you to set the SoundBank to load at initialization: set to to the GME bank:



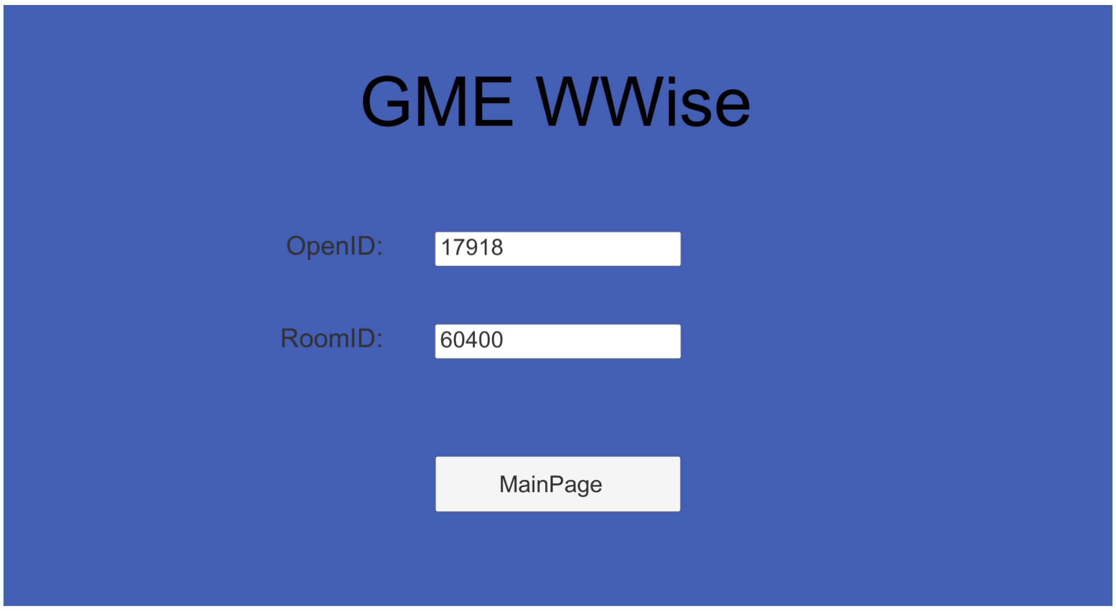
Auth Scene (Script)

Script	AuthScene
My Bank	GME

There are two scenes, the first one shows how to set the GME initial parameters, the second one shows the GME features. The related source files can be found in the following locations:

- Demo scenes: Assets\Scenes
- Demo scripts: Assets\Scripts

AuthScene.cs provides a facility to specify the UserID and the RoomID for the session.

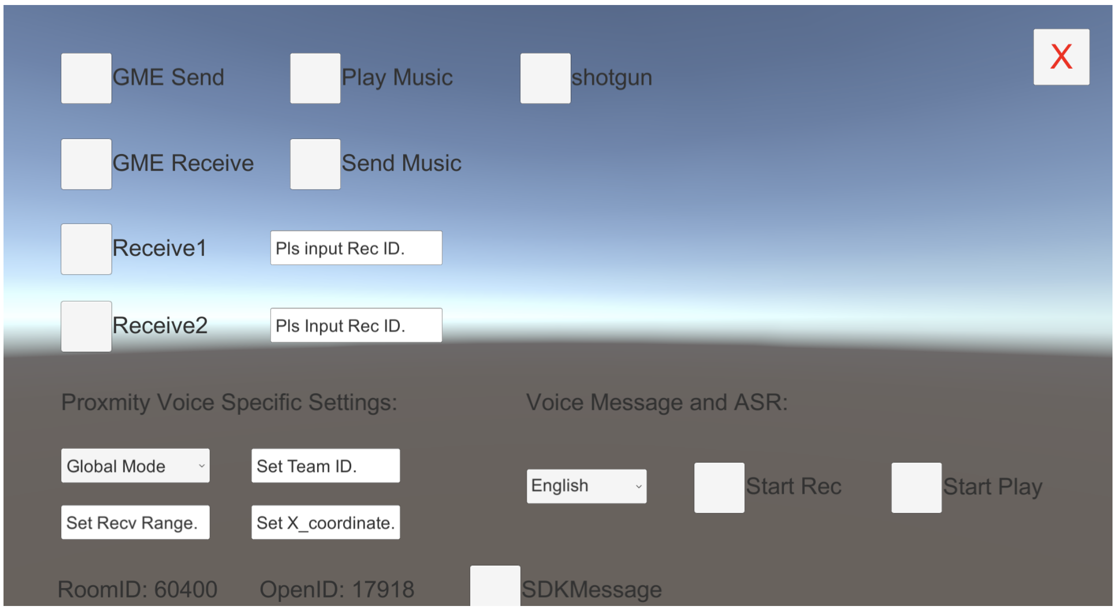


GME WWise

OpenID: 17918

RoomID: 60400

MainScene.cs contains the main script to post GME related Wwise events to enable voice chat.



RoomID: 60400 OpenID: 17918

UE4 Integration

Prerequisite

Before integrating GME functionality into a game using Unreal and Wwise, the Wwise integration must first be completed by the Wwise launcher. You can find more information on integrating Wwise to Unreal by following this link: https://www.audiokinetic.com/zh/library/edge/?source=UE4&id=index.html

In the Wwise project, create new send and receive points as described in the GME Wwise Integration Demo section of this document, and add the relevant GME events to a SoundBank.

SDK Package

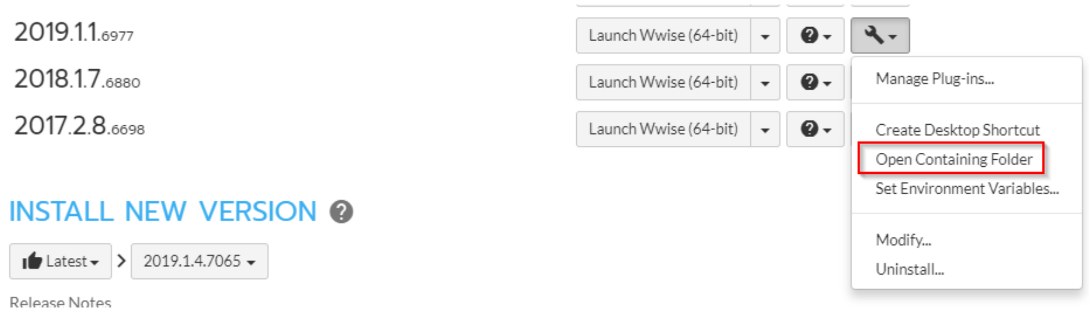
The GME integration SDK acts both as an Unreal plug-in and as a Wwise plug-in at the same time. It includes prebuilt plug-in libraries for each platform, build scripts and the API functions. To facilitate the GME integration, a sample game is available to download to help you understand how to use GME in an Unreal project. After installing the TencentGME for Wwise plug-in through the Wwise Launcher, the sample game is found in <WWISE_INSTALL_PATH>/SDK/plugins/TencentGME/Unreal/GMEWwiseDemo. Note that it is necessary to integrate Wwise into the Unreal project, using the Wwise Launcher.

The GME SDK package for Unreal is installed when installing the Tencent GME for Wwise plug-in from the Wwise Launcher. The SDK has been tested on Windows, Android, iOS, macOS, Xbox One, PS4 and Switch.

Integration

After installing the Tencent GME for Wwise plug-in from the Wwise Launcher, integrate Wwise to your Unreal project using the Unreal Engine tab in the Wwise Launcher. If you have already integrated Wwise to your Unreal project, you can skip this step.

Once the integration process is done, navigate to the Wwise installation folder:



From there, copy the folder:

```
<Wwise>/SDK/plugins/TencentGME/Unreal/GMEWwiseDemo/Plugins/TencentGME_Wwise
```

to the "Plugins" folder of your Unreal project. This folder contains everything required for the GME integration.

GME prebuilt plug-in libraries for different platform targets must be placed in the specified locations of the TencentGME_Wwise plugin-in "ThirdParty" folder. The naming style and the folder structure are exactly the same as those of the Wwise SDK. The following table shows the expected location paths for each platform.

Platform	GME libraries	Path
Windows	\lib\TencentGMEPlugin.lib \bin\gmesdk.dll	Plugins\TencentGME_Wwise\ThirdParty\x64_vc150\(CONFIG) Plugins\TencentGME_Wwise\ThirdParty\Win32_vc150\(CONFIG)
Android	Base: Android_arch\(CONFIG) \lib\libTencentGMEPlugin.a \bin\libgmesdk.so gmesdk.jar	Plugins\TencentGME_Wwise\ThirdParty\Android_arm64-v8a Plugins\TencentGME_Wwise\ThirdParty\Android_armeabi-v7a Plugins\TencentGME_Wwise\ThirdParty\Android_x86
iOS	\lib\libTencentGMEPlugin.a \lib\libGMESDK.a	Plugins\TencentGME_Wwise\ThirdParty\iOS\(CONFIG)
macOS	\lib\libTencentGMEPlugin.a \lib\libGMESDK.a	Plugins\TencentGME_Wwise\ThirdParty\Mac\(CONFIG)
Xbox One	\lib\TencentGMEPlugin.lib \bin\gmesdk.dll	Plugins\TencentGME_Wwise\ThirdParty\XboxOne_vc140\(CONFIG) Plugins\TencentGME_Wwise\ThirdParty\XboxOne_vc150\(CONFIG)
PS4	\lib\libTencentGMEPlugin.a \lib\GME.a	Plugins\TencentGME_Wwise\ThirdParty\PS4\(CONFIG)
Switch	Base: NX{arch}\(CONFIG) \lib\libTencentGMEPlugin.a \lib\libGMESDK.a	Plugins\TencentGME_Wwise\ThirdParty\NX{arch}\(CONFIG)

The TencentGME_Wwise module must be linked by the game C++ project. In the game project module file (eg. GMEWWiseDemo.Build.cs), add the GME for Wwise plug-in dependency as follows:

```
public class GMEWWiseDemo : ModuleRules
{
        public GMEWWiseDemo(ReadOnlyTargetRules Target) : base(Target)
        {
                PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;

                PublicDependencyModuleNames.AddRange(new string[]
                {
                        "Core",
                        "CoreUObject",
                        "Engine",
                        "InputCore",
                        "HeadMountedDisplay",
                        "AkAudio",
                        "TencentGME_Wwise"
                });
        }
}
```

Removing the TencentGME Library From the Wwise Plug-in

Starting with V6 of the GME plug-in, there must be only one copy of the TencentGME libraries and it must reside in the TencentGME_Wwise Unreal plug-in. As such, **you must delete all** libTencentGME **and** TencentGME **files from the <u>Wwise</u> plug-in** Plugins/Wwise/ThirdParty/ folder. Keep the header file from the Include folder (Plugins/Wwise/ThirdParty/include/AK/Plugin/TencentGMEFactory.h). Failure to do so will result in communication failures.

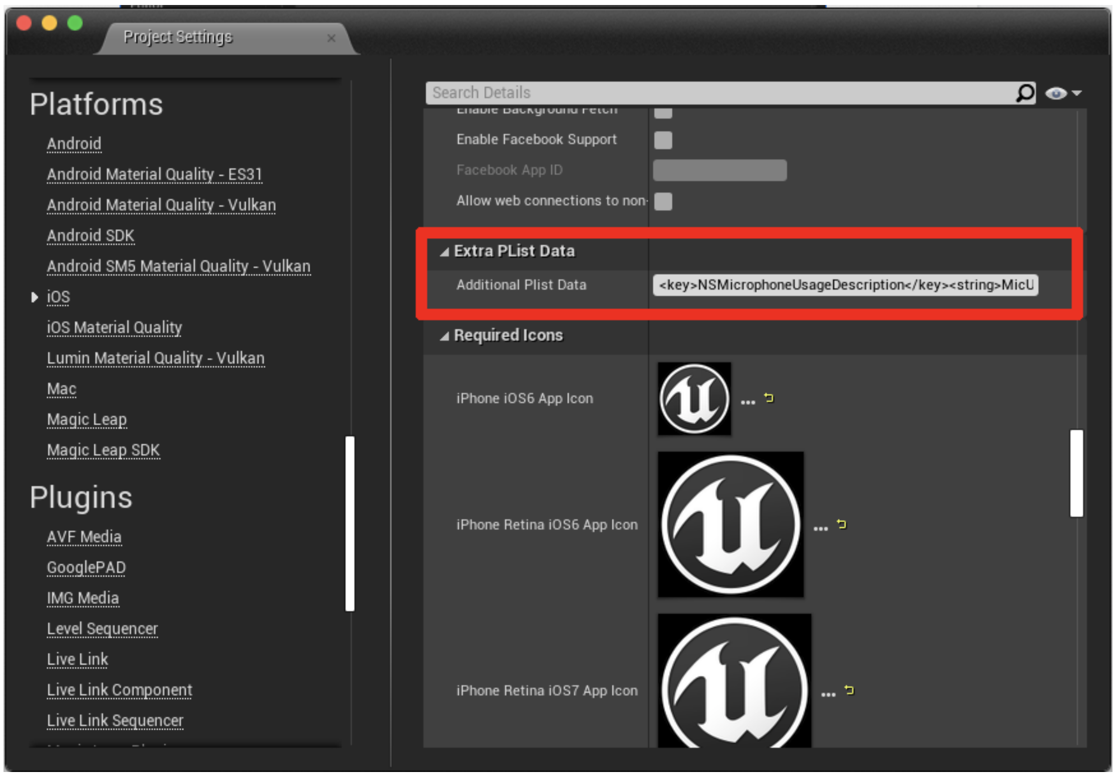
You must also remove the TencentGME plug-in registration from the AkAudio plug-in. To do so, remove any lines written as `#include <AK/Plugin/TencentGMEFactory.h>` from the `Plugins/Wwise/Source/AkAudio/Private/Generated` folder. This file inclusion is added automatically at integration.

Platform-specific configuration

iOS

Because the integration process copies the entire Wwise SDK, there will be a copy of libGMESDK.a in both the TencentGME_Wwise and the Wwise plug-ins for Unreal. **You must delete all of the libGMESDK.a files from the various subdirectories of the Wwise plug-in directory** in your Unreal project. Otherwise, duplicate libraries will be reported and the build will fail.

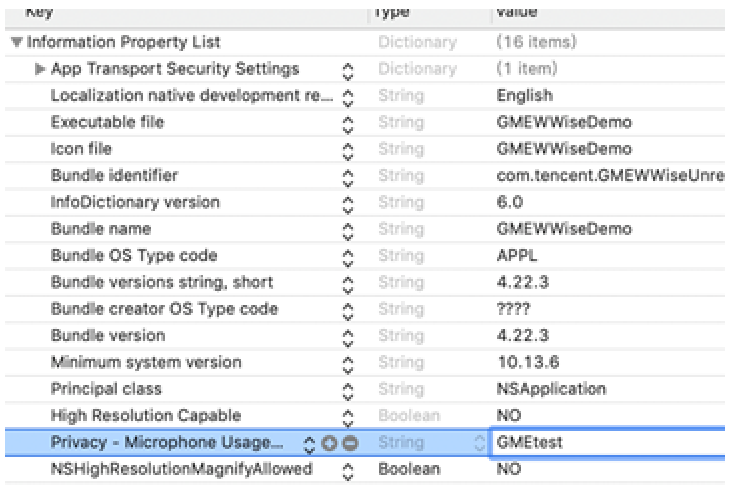
Also, in the Unreal project's Project Settings Window (Edit > Project Settings), go to the iOS platform settings. Add the following microphone permissions for the Plist Data:
`<key>NSMicrophoneUsageDescription</key><string>MicUse</string>`



Extra PList Data
Additional Plist Data
<key>NSMicrophoneUsageDescription</key><string>MicU

macOS

In the current versions of Unreal Editor, microphone permissions cannot be set for the macOS platform: you will need to manually add the microphone permissions in the final App package in order to do the recording. Open the Info.plist file in the App package with a text editor, and then add microphone permissions as follows:



Key	Type	Value
▼ Information Property List	Dictionary	(16 items)
▶ App Transport Security Settings	Dictionary	(1 item)
Localization native development re...	String	English
Executable file	String	GMEWWiseDemo
Icon file	String	GMEWWiseDemo
Bundle identifier	String	com.tencent.GMEWWiseUnre
InfoDictionary version	String	6.0
Bundle name	String	GMEWWiseDemo
Bundle OS Type code	String	APPL
Bundle versions string, short	String	4.22.3
Bundle creator OS Type code	String	????
Bundle version	String	4.22.3
Minimum system version	String	10.13.6
Principal class	String	NSApplication
High Resolution Capable	Boolean	NO
Privacy - Microphone Usage...	String	GMEtest
NSHighResolutionMagnifyAllowed	Boolean	NO

GME API for C++ Project

There are specific APIs that must be called before using the GME service. The exposed C++ API calls are declared in the TencentGMEDevice.h file. These API calls are available to any game with functionality that depends on theTencentGME_Wwise module. For a description of each function, see the GME for Wwise plug-in API section of this document.

```
class TENCENTGME_WWISE_API FTencentGMEDevice
{
public:
    static FString GetGMEVersion();
    static void SetUserID(const FString& userID);
    static void SetRoomID(const FString& roomID);

    static void ReceivePlugin_SetReceiveOpenIDWithGameObjectID(
        AkGameObjectID gameObjectID, const FString& userID);
    static void ReceivePlugin_GetReceiveOpenIDWithGameObjectID(
        AkGameObjectID gameObjectID, FString& userID);
    static void SendPlugin_EnableLoopbackWithGameObjectID(
        AkGameObjectID gameObjectID, bool enableLoopback);
    static bool SendPlugin_GetEnableLoopbackWithGameObjectID(
        AkGameObjectID gameObjectID);

    static void GMEPause();
    static void GMEResume();
    static int GetAudioSendStreamLevel();
    static int GetAudioRecvStreamLevel(const FString& targetID);
    static void AddAudioBlockList(const FString& targetID);
    static void RemoveAudioBlockList(const FString& targetID);

    static void SetLogLevel(
        GMEWWisePlugin_LogLevel levelWrite,
        GMEWWisePlugin_LogLevel levelPrint);
    static void SetRangeAudioTeamID(int teamID);
    static void SetRangeAudioTeamMode(GMEWWisePlugin_TeamMode teamMode);
    static void SetRangeAudioRecvRange(int range);
    static void SetSelfPosition(int positionX, int positionY, int positionZ);

    static int StartRecording(const FString& fileid);
    static int StopRecording();
    static int PlayRecordFile(const FString& fileid);
    static int StopPlayFile();
    static int SpeechToText(
        const FString& fileid,
        const FString& speechLanguage,
        const FString& translateLanguage);
    static int GetVoiceFileDuration(const FString& fileid);

    static int GetMessage(
              int& localUTCTime,
              int& messageType,
              int& code,
              FString& message1,
              int len1,
              String& message2,
              int len2);
};
```

Public functions exposed in TencentGMEDevice.h

GME API for Blueprint Project

The API is also exposed to the Blueprint project. The functions are declared in the TencentGMEBlueprintLibrary.h file and map to the functions in TencentGMEDevice.h in a one-to-one fashion. They are listed under the category “TencentGME”.

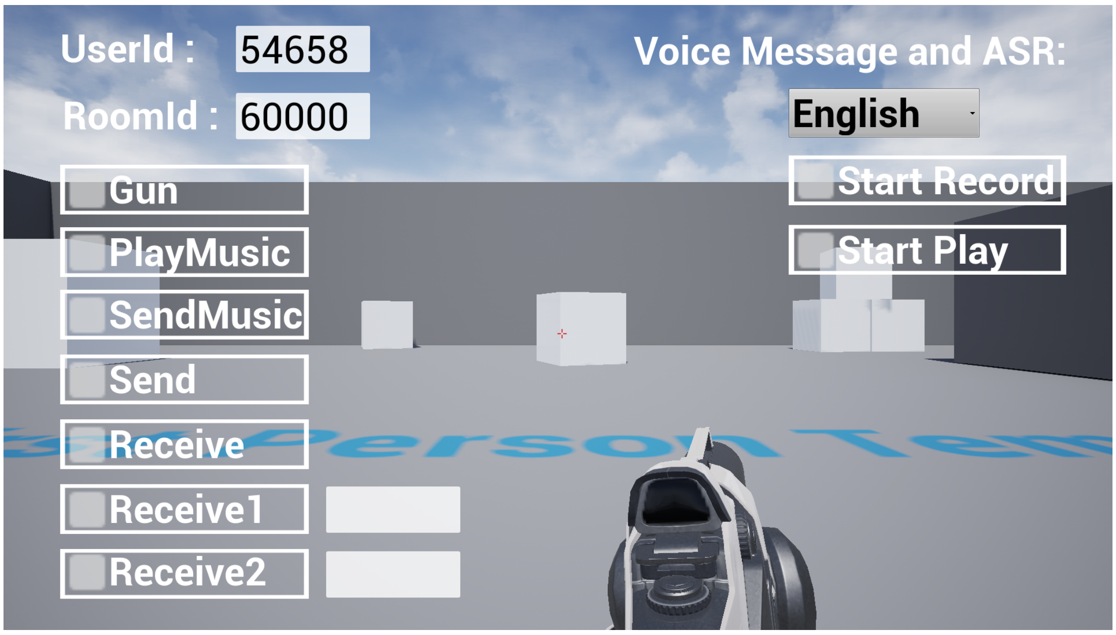
Reference Sample Game

The GME Unreal SDK is bundled with a sample game to help you understand how to use GME in an Unreal C++ project. You must integrate Wwise using the Wwise Launcher prior to using it. Go to the Unreal Engine tab in the Wwise Launcher and select “Browse for project…” from the menu button next to “Recent Unreal Engine Projects”. Once added, click on the blue “Integrate Wwise into Project…” button for that project and follow the steps.

The related source files for the sample game can be found in the following locations:

- C++ code: GMEWWiseDemo\Source
- UI: GMEWWiseDemo\Content\UI
- Soundbank and Event assets: GMEWWiseDemo\Content

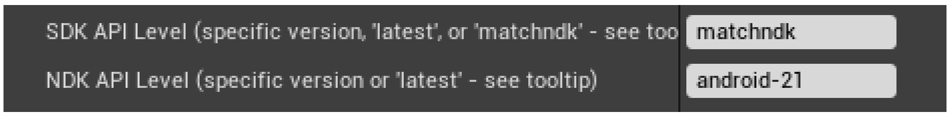
HomeViewController.cpp is the main C++ file to trigger the GME related Wwise events to enable for voice chat.



Interface of the Unreal Engine demo game provided as sample

Android

When targeting Android, make sure that you have matching SDK and NDK versions in your Unreal project settings.



SDK API Level (specific version, 'latest', or 'matchndk' - see too	matchndk
NDK API Level (specific version or 'latest' - see tooltip)	android-21

Android SDK and NDK settings in the Unreal project settings page

An XML manifest file is provided for Android that sets the correct permissions and copies additional files such as the libgmesdk.so library at build time. This manifest file is found in:

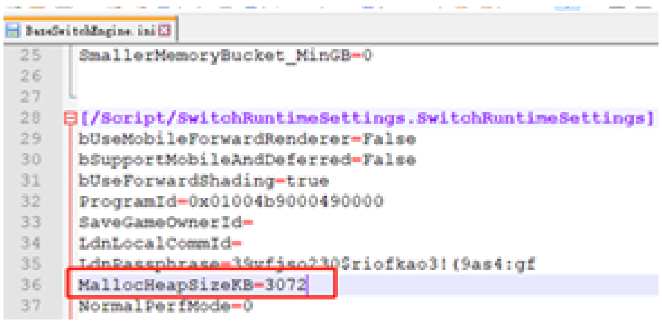
```
GMEWwiseDemo/Plugins/TencentGME_Wwise/Source/TencentGME_Wwise/GMESDK_APL.xml
```

If you are targeting a different architecture than the default *armeabi-v7a*, you will need to modify this file to copy the appropriate library for the target architecture:

```
<copyFile
src="$S(PluginDir)/../../ThirdParty/Android/armeabi_v7a/Release/bin/libgmesdk.so"
dst="$S(BuildDir)/libs/armeabi-v7a/libgmesdk.so"/>
<!-- Uncomment the appropriate line for the target architecture -->
<!--<copyFile
src="$S(PluginDir)/../../ThirdParty/Android/arm64-v8a/Release/bin/libgmesdk.so"
dst="$S(BuildDir)/libs/arm64-v8a/libgmesdk.so"/> -->
<!-- <copyFile
src="$S(PluginDir)/../../ThirdParty/Android/x86/Release/bin/libgmesdk.so"
dst="$S(BuildDir)/libs/x86/libgmesdk.so"/> -->
```

Switch

The heap size needs to be sufficiently large to ensure the sample project’s resources are successfully allocated. This is defined in the Switch-specific config file (Config\Switch\BaseSwitchEngine.ini). The default size is 3072, but should be set to a larger value such as 9216.



```
SmallerMemoryBucket_MinGB=0


[/Script/SwitchRuntimeSettings.SwitchRuntimeSettings]
bUseMobileForwardRenderer=False
bSupportMobileAndDeferred=False
bUseForwardShading=true
ProgramId=0x01004b9000490000
SaveGameOwnerId=
LdnLocalCommId=
LdnPassphrase=39vfjso230$riofkao3!(9as4:gf
MallocHeapSizeKB=3072
NormalPerfMode=0
```