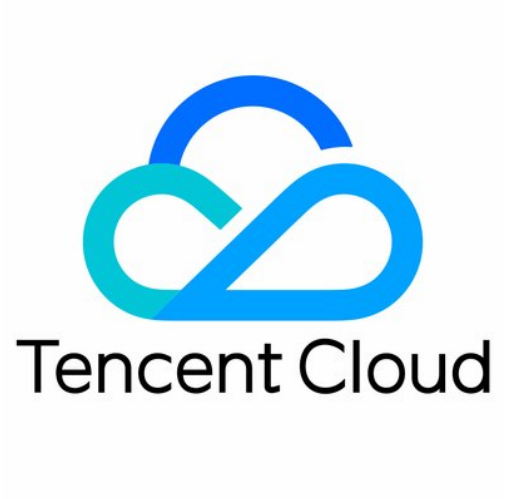
Game Multimedia Engine

Advanced Features

Product Documentation



Contents

Advanced Features
Range Voice

Last updated : 2020-10-28 16:41:53

This document describes how to integrate and debug the GME APIs for range voice.

GME's range voice service is specially developed for games similar to PUBG. Different from a team voice room, a range voice room has the following core capabilities:

1. It provides the "Team only" and "Everyone" voice modes unique to **games like PUBG and Battle Royale**.
2. Relying on its range determination capability, it enables **high numbers of users** to voice chat using their mic in the same voice room.

Concepts

Range voice room

Before using range voice, you need to call the `SetRangeAudioTeamID` API to set the team ID (`teamID`) and then call the `EnterRoom` API to enter the room. If the specified `teamID` is not set to `0` before room entry, a range voice room will be used.

Voice mode

When the user enters a range voice room, the user can choose from two voice modes:

Voice Mode	Parameter Name	Feature
Everyone	RANGE_AUDIO_MODE_WORLD	In this mode, every one within a certain range can hear the user as he or she chats with his or her teammates
Team only	RANGE_AUDIO_MODE_TEAM	Only teammates can hear the user

In different voice modes, the specific sound reachability is as follows:

a. Players in the same team can hear each other regardless of how far apart they are or what voice modes they use.

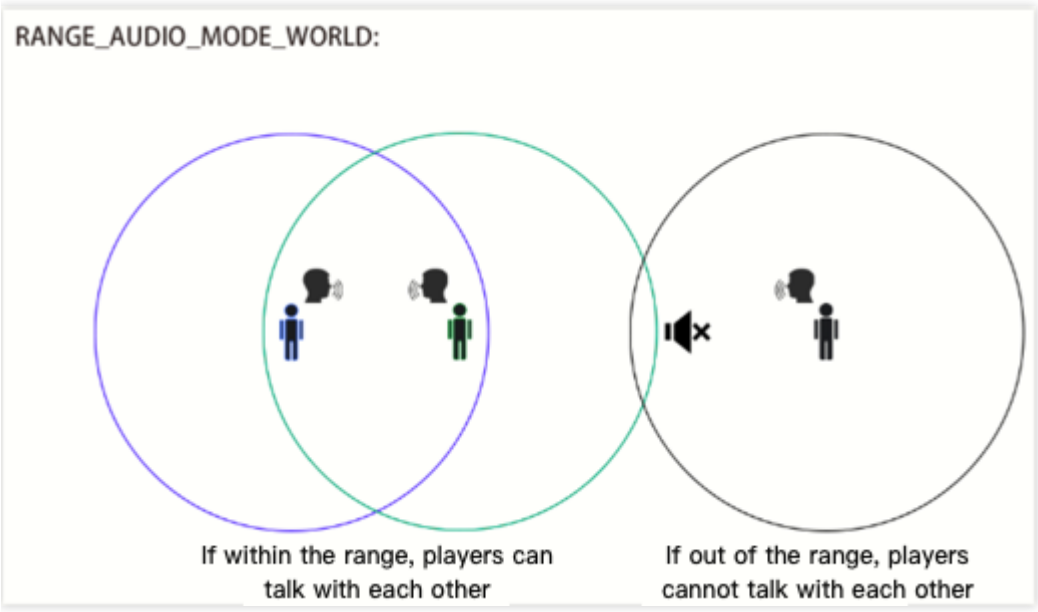
b. Players in different teams can hear each other only if they are all in the "Everyone" voice mode and within a certain distance.

> **Note :**
>
> For more information on the specific sound reachability of players, please see “Appendix” in Range Voice.

Voice reception range

If the voice mode is set to **Everyone** (`RANGE_AUDIO_MODE_WORLD`), the voice reachability will be subject to the `UpdateAudioRecvRange` API.

- Regardless of whether teammates are within the voice range, they can always talk with each other.
- To set the voice reception range, use the `UpdateAudioRecvRange` API.
- Voice modes can be switched in real time in the range voice room. However, changing the `teamID` , which must be specified before room entry, in the range voice room is not supported.
- To use range voice, room entry must be performed with smooth sound quality (i.e., RoomType = 1).



Directions

Different from a team voice room, a range voice room must use smooth sound quality, and `SetRangeAudioTeamID` and `SetRangeAudioMode` must be called before `EnterRoom`. Call `UpdateAudioRecvRange` after successful room entry and call `UpdateSelfPosition` once per each frame.

If you want to use the 3D sound effect at the same time, please do so as instructed in “Simultaneous Use of the 3D Sound Effect” in Range Voice after completing steps 1 and 2 below.

1. Set `teamID`

The team ID (`teamID`) must be set by calling the method `SetRangeAudioTeamID` before `EnterRoom`. Otherwise, GME will return the error code `AV_ERR_ROOM_NOT_EXITED(1202)`. If you re-enter a room you exited, please wait for the `RoomExitComplete` callback notification before setting the team ID.

> ⚠ **Note :**
>
> This parameter will not be automatically reset to `0` upon room exit. Therefore, once you decide to use range voice, please call this method to set `teamID` before calling `EnterRoom` each time.

Function prototype

```
ITMGContext SetRangeAudioTeamID(int teamID)
```

Parameter	Type	Description
teamID	int	Team ID, which is used for audio upstream/downstream control in range voice. When it is set to `0` (default), the team voice room is used.

2. Set the voice mode

- The voice mode can be modified by calling the method `SetRangeAudioMode` either before or after room entry.
- Calling this method before room entry will affect the next room entry.
- Calling this method after room entry will directly change the current voice mode.
- This parameter will not be automatically reset to `MODE_WORLD` upon room exit. Therefore, once you decide to call this method, please do so before each call of `EnterRoom`.

Function prototype

```
ITMGRoom int SetRangeAudioMode(RANGE_AUDIO_MODE rangeAudioMode)
```

Parameter	Type	Description

rangeAudioMode	int	`0` (MODE_WORLD): everyone; `1` (MODE_TEAM): team only

3. Enter a voice room

Before entering a voice room by calling `EnterRoom`, you need to call the following two APIs: `SetRangeAudioTeamID` and `SetRangeAudioMode`.

```
ITMGContext.GetInstance(this).EnterRoom(roomId,ITMG_ROOM_TYPE_FLUENCY, authBuffer);
```

Be sure to use smooth sound quality for the voice room, and monitor and process the callback for room entry.

```
public void OnEvent(ITMGContext.ITMG_MAIN_EVENT_TYPE type, Intent data) {
if (ITMGContext.ITMG_MAIN_EVENT_TYPE.ITMG_MAIN_EVENT_TYPE_ENTER_ROOM == type)
{
//Analyze the returned result
int nErrCode = data.getIntExtra("result" , -1);
String strErrMsg = data.getStringExtra("error_info");

if (nErrCode == AVError.AV_OK)
{
//Return a success response for room entry. You can proceed with your operation
ScrollView_ShowLog("EnterRoom success");
Log.i(TAG,"EnterRoom success!");
}
else
{
//Failed to enter the room. You need to analyze the returned error message
ScrollView_ShowLog("EnterRoom fail :" + strErrMsg);
Log.i(TAG,"EnterRoom fail!");
}
}
}
```

Once you enter the room, call `UpdateAudioRecvRange` at least once, and call `UpdateSelfPosition` once per frame.

4. Set the voice reception range

- This method is used to set the voice reception range (subject to your own game engine) and can be called **only after successful room entry**.
- This method must be used in conjunction with `UpdateSelfPosition` to update the sound source position.
- This method only needs to be called once.

Function prototype

```
ITMGRoom int UpdateAudioRecvRange(int range)
```

Parameter	Type	Description
range	int	Maximum voice reception range in the distance unit used by the game engine

Sample code

```
ITMGContext.GetInstance().GetRoom().UpdateAudioRecvRange(300);
```

5. Update the sound source position

To update the sound source position is to inform the server of the local player's position. The system implements range voice by checking the **local coordinates and the voice reception range** against the **remote coordinates and the voice reception range**.

- This function is used to update the sound source position information. It can be called only **after successful room entry** and needs to be called **once per each frame**. Taking the Unity engine as an example, this API needs to be called in `Update` .
- If range voice is enabled, the sound source position must be updated. Even if range determination is not required, **this API still needs to be called once after room entry**.
- If you want to enable the 3D sound effect at the same time, set the `axisForward` , `axisRight` , and `axisUp` parameters as specified in the “Simultaneous Use of the 3D Sound Effect” section below.

Function prototype

```
public abstract int UpdateSelfPosition(int position[3], float axisForward[3], float axisRight[3],
float axisUp[3])
```

Parameter	Type	Description
position	int[]	Local position (forward, right and up) in the world coordinate system
axisForward	float[]	This parameter can be ignored in this product
axisRight	float[]	This parameter can be ignored in this product
axisUp	float[]	This parameter can be ignored in this product

Simultaneous Use of the 3D Sound Effect

You can enable both the 3D sound effect and the range voice at the same time by following the steps below:

The following steps need to be performed after steps 1, 2 and 3 above are completed.

1. Initialize the 3D sound effect engine

This function is used to initialize the 3D sound effect engine and needs to be called after room entry. You must call this API before using the 3D sound effect. Even if you want to enable only the 3D sound effect reception and not playback, you still need to call this API.

Function prototype

```
public abstract int InitSpatializer(string modelPath)
```

Parameter	Type	Description
modelPath	string	Path of the 3D sound effect resource file. Please download the 3D sound effect model file here (MD5 checksum: d0b76aa64c46598788c2f35f5a8a8694) and store it locally. Its storage path is passed in to the SDK through this parameter. You must use this official file

2. Enable/disable the 3D sound effect

This function is used to enable/disable the 3D sound effect. You can hear the 3D sound after enabling it.

Function prototype

```
public abstract int EnableSpatializer(bool enable, bool applyToTeam)
```

Parameter	Type	Description
enable	bool	After enabling the 3D sound effect, you can hear the 3D sound effect
applyToTeam	bool	Specifies whether the 3D sound effect is enabled within the team and takes effect only when `enable` is `true`

The `IsEnableSpatializer` API can be used to get the 3D sound effect status.

3. Set the voice reception range (for the 3D sound effect)

- This method is used to set the voice reception range (subject to the game engine) and can be called **only after successful room entry**.
- This method must be used in conjunction with `UpdateSelfPosition` to update the sound source position.
- This method only needs to be called once.
- In the 3D sound effect, the volume level of the sound source attenuates as the distance to the sound source increases. If the unit distance exceeds the range, the volume level will attenuate to almost zero.
- For more information on the relationship between the distance and sound attenuation, please see the appendix.

Function prototype

```
ITMGRoom int UpdateAudioRecvRange(int range)
```

Parameter	Type	Description
range	int	Maximum voice reception range in the distance unit used by the game engine

Sample code

```
ITMGContext.GetInstance().GetRoom().UpdateAudioRecvRange(300);
```

4. Update the sound source position (for the 3D sound effect)

To update the sound source position is to inform the server of the local player's position. The system implements range voice by checking the **local coordinates and the voice reception range** against the **remote coordinates and the voice reception range**.

- This function is used to update the sound source position information. It can be called only **after successful room entry** and needs to be called **once per frame**. Taking the Unity engine as an example, this API needs to be called in `Update` .
- **Please directly copy the sample code to use this feature.**

Function prototype

```
public abstract int UpdateSelfPosition(int position[3], float axisForward[3], float axisRight[3],
float axisUp[3])
```

Parameter	Type	Description

position	int[]	Local position (forward, right and up) in the world coordinate system
axisForward	float[]	Forward vector in the local coordinate system
axisRight	float[]	Right vector in the local coordinate system
axisUp	float[]	Up vector in the local coordinate system

Sample code

Unreal

```
FVector cameraLocation = UGameplayStatics::GetPlayerCameraManager(GetWorld(), 0)->GetCameraLocati
on();
FRotator cameraRotation = UGameplayStatics::GetPlayerCameraManager(GetWorld(), 0)->GetCameraRotat
ion();
int position[] = {
(int)cameraLocation.X,
(int)cameraLocation.Y,
(int)cameraLocation.Z };
FMatrix matrix = ((FRotationMatrix)cameraRotation);
float forward[] = {
matrix.GetColumn(0).X,
matrix.GetColumn(1).X,
matrix.GetColumn(2).X };
float right[] = {
matrix.GetColumn(0).Y,
matrix.GetColumn(1).Y,
matrix.GetColumn(2).Y };
float up[] = {
matrix.GetColumn(0).Z,
matrix.GetColumn(1).Z,
matrix.GetColumn(2).Z};
ITMGContextGetInstance()->GetRoom()->UpdateSelfPosition(position, forward, right, up);
```

Unity

```
Transform selftrans = currentPlayer.gameObject.transform;
Matrix4x4 matrix = Matrix4x4.TRS(Vector3.zero, selftrans.rotation, Vector3.one);
int[] position = new int[3] {
selftrans.position.z,
selftrans.position.x,
selftrans.position.y};
float[] axisForward = new float[3] {
matrix.m22,
matrix.m02,
matrix.m12 };
```

```
float[] axisRight = new float[3] {
matrix.m20,
matrix.m00,
matrix.m10 };
float[] axisUp = new float[3] {
matrix.m21,
matrix.m01,
matrix.m11 };
ITMGContext.GetInstance().GetRoom().UpdateSelfPosition(position, axisForward, axisRight, axisUp);
```

Appendix

Voice modes

Player voice reachability in different voice modes is as follows:

- The table below lists the possible cases of sound reachability for player B in different voice modes if player A selects the "Everyone" mode:

In the Same Team	Within the Range	Voice Mode	A Can Hear B	B Can Hear A
Yes	Yes	MODE_WORLD	Yes	Yes
Yes	Yes	MODE_TEAM	Yes	Yes
Yes	No	MODE_WORLD	Yes	Yes
Yes	No	MODE_TEAM	Yes	Yes
No	Yes	MODE_WORLD	Yes	Yes
No	Yes	MODE_TEAM	No	No
No	No	MODE_WORLD	No	No
No	No	MODE_TEAM	No	No

- The table below lists the possible cases of sound reachability for player B in different voice modes if player A selects the "Team only" mode:

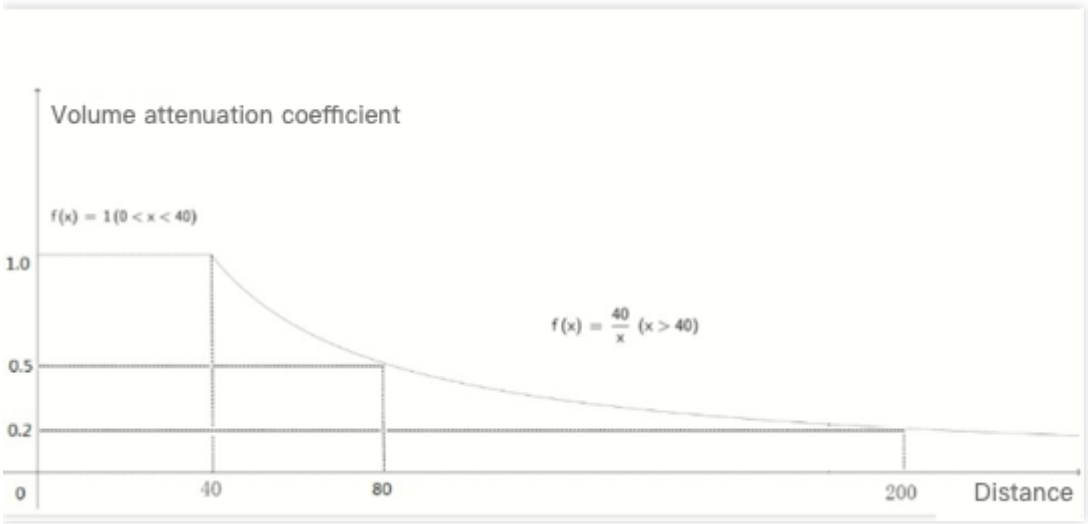
In the Same Team	Within the Range	Voice Mode	A Can Hear B	B Can Hear A
Yes	Yes	MODE_WORLD	Yes	Yes

Yes	Yes	MODE_TEAM	Yes	Yes
Yes	No	MODE_WORLD	Yes	Yes
Yes	No	MODE_TEAM	Yes	Yes
No	Yes	MODE_WORLD	No	No
No	Yes	MODE_TEAM	No	No
No	No	MODE_WORLD	No	No
No	No	MODE_TEAM	No	No

The relationship between distance and sound attenuation

In the 3D sound effect, the sound will begin to attenuate to almost zero as the distance to the sound source exceeds a specified threshold (range/10).

Distance (Unit in Engine)	Attenuation Coefficient
0 < N < range/10	1.0 (no attenuation)
N ≥ range/10	range/10/N



3D Sound Effect

Last updated : 2021-02-26 17:00:23

This document provides a detailed description that makes it easy for developers to debug and integrate GME APIs for 3D sound effect.

Initialize the 3D sound effect engine

This function is used to initialize the 3D sound effect engine and needs to be called after room entry. You must call this API before using the 3D sound effect. Even if you want to enable only the 3D sound effect reception and not playback, you still need to call this API.

Function prototype

```
public abstract int InitSpatializer(string modelPath)
```

Parameter	Type	Description
modelPath	String	Path of 3D sound effect resource file

The 3D sound effect resource file needs to be downloaded to your local disk, please select one according to the SDK version:

- For SDKs below v2.8, please click here to download (md5: d0b76aa64c46598788c2f35f5a8a8694).
- For SDKs above V2.8, please click here to download (md5: 3d4d04b3949e267e34ca809e8a0b9243).

For GME SDK release updates, please see Product Updates.

Enable or disable the 3D sound effect

This function is used to enable or disable the 3D sound effect. You can hear the 3D sound after enabling it.

Function prototype

```
public abstract int EnableSpatializer(bool enable, bool applyToTeam)
```

Parameter	Type	Description
enable	Bool	You can hear the 3D sound after enabling it.
applyToTeam	Bool	Indicates whether 3D sound applies to the team. This parameter takes effect only when `enable` is "true".

Obtain status of 3D sound effect

This function is used to obtain the status of 3D sound effect.

Function prototype

```
public abstract bool IsEnableSpatializer()
```

Returned Value	Description
true	It is enabled.
false	It is disabled.

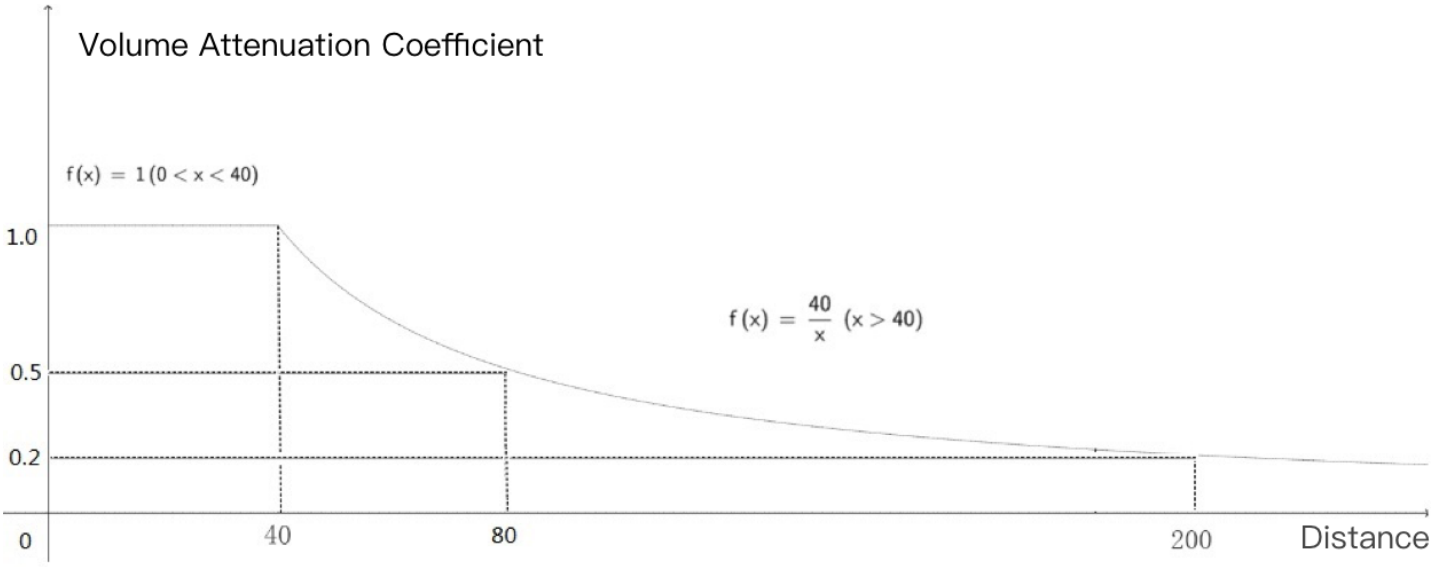
Update sound source azimuth (including orientation)

This function is used to update the sound source azimuth information. The 3D sound effect can be achieved by calling this function for each frame.

The relationship between distance and sound attenuation

In the 3D sound effect, the sound will begin to attenuate to almost zero as the distance to the sound source exceeds a specified threshold (range/10).

Distance (Unit in Engine)	Attenuation Coefficient
0 < N < range/10	1.0 (no attenuation)
N ≥ range/10	range/10/N



Function prototype

```
public abstract void UpdateAudioRecvRange(int range)
```

Parameter	Type	Description
range	Int	Sets the range within which the sound can be received

```
public abstract int UpdateSelfPosition(int position[3], float axisForward[3], float axisRight[3],
float axisUp[3])
```

Parameter	Type	Description
position	Int[]	The user's X-Y-Z coordinates in the world coordinate system
axisForward	Float[]	The unit vector of x axis in the coordinate system
axisRight	Float[]	The unit vector of y axis in the coordinate system
axisUp	Float[]	The unit vector of z axis in the coordinate system

Sample code

Unreal

```
FVector cameraLocation = UGameplayStatics::GetPlayerCameraManager(GetWorld(), 0)->GetCameraLocati
on();
FRotator cameraRotation = UGameplayStatics::GetPlayerCameraManager(GetWorld(), 0)->GetCameraRotat
ion();
int position[] = { (int)cameraLocation.X,(int)cameraLocation.Y, (int)cameraLocation.Z };
FMatrix matrix = ((FRotationMatrix)cameraRotation);
float forward[] = { matrix.GetColumn(0).X,matrix.GetColumn(1).X,matrix.GetColumn(2).X };
float right[] = { matrix.GetColumn(0).Y,matrix.GetColumn(1).Y,matrix.GetColumn(2).Y };
float up[] = { matrix.GetColumn(0).Z,matrix.GetColumn(1).Z,matrix.GetColumn(2).Z};
ITMGContextGetInstance()->GetRoom()->UpdateSelfPosition(position, forward, right, up);
```

Unity

```
Transform selftrans = currentPlayer.gameObject.transform;
Matrix4x4 matrix = Matrix4x4.TRS(Vector3.zero, selftrans.rotation, Vector3.one);
int[] position = new int[3] { selftrans.position.z, selftrans.position.x, selftrans.position.y };
float[] axisForward = new float[3] { matrix.m22, matrix.m02, matrix.m12 };
float[] axisRight = new float[3] { matrix.m20, matrix.m00, matrix.m10 };
float[] axisUp = new float[3] { matrix.m21, matrix.m01, matrix.m11 };
ITMGContext.GetInstance().GetRoom().UpdateSelfPosition(position, axisForward, axisRight, axisUp);
```

Custom Audio Forwarding Routing

Last updated : 2021-02-02 16:26:55

This document provides a detailed description of using custom audio forwarding routing that makes it easy for developers to debug and integrate the APIs for Game Multimedia Engine (GME).

Setting Audio Forwarding Rules

This API is used to set audio forwarding rules, and it is called in the callback of successful room entry. After being called, this API will take effect for this room entry and will be invalid after room exit.

API Prototype

```
//iOS API
-(int)SetServerAudioRouteSendOperateType:(ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE) Sendtype SendList:(NSArray *)OpenIDForSend RecvOperateType:(ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE) Recvtype RecvList:(NSArray *)OpenIDForRecv;
//Unity API
public abstract int SetServerAudioRouteSendOperateType(ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE Sendtype, string[] OpenIDforSend, ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE Recvtype, string[] OpenIDforRecv);
```

Type Descriptions

ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE

Set audio sending rules. There will be different sending effects for the different sending rules.

Sending Types	Effects
AUDIO_ROUTE_NOT_SEND_TO_ANYONE	The local audio is sent upstream to the backend, but the backend does not forward to anyone, which is equivalent to mute the local. At this time, the parameter OpenIDForSend is invalid. You just need to enter null.
AUDIO_ROUTE_SEND_TO_ALL	The local audio is sent upstream to everyone. At this time, the parameter OpenIDForSend is invalid. You just need to enter null.
AUDIO_ROUTE_SEND_BLACK_LIST	The local audio is sent upstream and will not be forwarded to people in the blocklist, which is provided by the parameter OpenIDForSend.

AUDIO_ROUTE_SEND_WHITE_LIST	The local audio is sent upstream and is forwarded to people in the allowlist, which is provided by the parameter OpenIDForSend.

Note :

- If the passed type is AUDIO_ROUTE_NOT_SEND_TO_ANYONE or AUDIO_ROUTE_SEND_TO_ALL, then the parameter OpenIDForSend does not take effect. You just need to enter null.
- If the passed type is AUDIO_ROUTE_SEND_BLACK_LIST, the parameter OpenIDForSend will be the blocklist. Up to 10 people are supported.
- If the passed type is AUDIO_ROUTE_SEND_WHITE_LIST, the parameter OpenIDForSend will be the allowlist. Up to 10 people are supported.

ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE

Set audio receiving rules. There will be different receiving effects for the different receiving rules.

Receiving Types	Effects
AUDIO_ROUTE_NOT_RECV_FROM_ANYONE	The local does not receive any audio, which is equivalent to disable the speaker in the room. At this time, the parameter OpenIDForSend is invalid. You just need to enter null.
AUDIO_ROUTE_RECV_FROM_ALL	The local receives everyone’s audio. At this time, the parameter OpenIDForSend is invalid. You just need to enter null.
AUDIO_ROUTE_RECV_BLACK_LIST	The local does not receive the audio of people in the blocklist. The blocklist is provided by the parameter OpenIDForSend.
AUDIO_ROUTE_RECV_WHITE_LIST	The local will only receive the audio of people in the allowlist. The allowlist is provided by the parameter OpenIDForSend.

Note :

- If the passed type is AUDIO_ROUTE_NOT_RECV_FROM_ANYONE or AUDIO_ROUTE_RECV_FROM_ALL, the parameter OpenIDForSend does not take effect.
- If the passed type is AUDIO_ROUTE_RECV_BLACK_LIST, the parameter OpenIDForSend will be the blocklist. Up to 10 people are supported.
- If the passed type is AUDIO_ROUTE_RECV_WHITE_LIST, the parameter OpenIDForSend will be the allowlist. Up to 10 people are supported.

Response

A returned value of QAV_OK indicates the call is successful.

- If the callback returns 1004, it means the parameter is incorrect. Please check the parameter.
- If the callback returns 1004, it means the operation is repeated.
- If the callback returns 1201, it means the room does not exist. Please check whether the room number is correct.
- If the callback returns 10001 and 1005, please call the API again.

For more explanations of the returned result, please see Error Codes.

Sample code

Executed statements

```
@synthesize _sendListArray;
@synthesize _recvListArray;

int ret = [[[ITMGContext GetInstance] GetRoom] SetServerAudioRouteSendOperateType:SendType SendLi
st:_sendListArray RecvOperateType:RecvType RecvList:_recvListArray];
if (ret != QAV_OK) {
UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Failed to update audioroute list" messa
ge:[NSString stringWithFormat:@"error code:%d",ret] delegate:NULL cancelButtonTitle:@"OK" otherBu
ttonTitles:nil];
[alert show];
}
```

Callback

```
-(void)OnEvent:(ITMG_MAIN_EVENT_TYPE)eventType data:(NSDictionary *)data{
NSString* log =[NSString stringWithFormat:@"OnEvent:%d,data:%@", (int)eventType, data];
switch (eventType) {
case ITMG_MAIN_EVENT_TYPE_SERVER_AUDIO_ROUTE_EVENT:{
```

```
{
UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Update audioroute" message:[NSString st
ringWithFormat:@"Result:%@,sub_type: %@ errorinof: %@", data[@"result"],data[@"sub_type"],data[@
"error_info"]] delegate:NULL cancelButtonTitle:@"OK" otherButtonTitles:nil];
[alert show];
}
}
default:
break;
}
}
```

Obtaining Audio Forwarding Rules

This API is used to obtain audio forwarding rules. After being called, the API returns the rule, and the passed array parameter will return the openId of the corresponding rule.

API Prototype

```
//iOS API
-(ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE)GetCurrentSendAudioRoute:(NSMutableArray *) OpenIDForSend;
-(ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE)GetCurrentRecvAudioRoute:(NSMutableArray *)OpenIDForRecv;
//Unity API
public abstract ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE GetCurrentSendAudioRoute(List<string> OpenIDfor
Send);
public abstract ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE GetCurrentRecvAudioRoute(List<string> OpenIDfor
Recve);
```

Returning rules

ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE

Sending Types	Effects
AUDIO_ROUTE_NOT_SEND_TO_ANYONE	The local audio is sent upstream to the backend, but the backend does not forward to anyone, which is equivalent to mute the local.
AUDIO_ROUTE_SEND_TO_ALL	The local audio is sent upstream to everyone.
AUDIO_ROUTE_SEND_BLACK_LIST	The local audio is sent upstream and will not be forwarded to people in the blocklist.
AUDIO_ROUTE_SEND_WHITE_LIST	The local audio is sent upstream and is forwarded to

	people in the allowlist.
AUDIO_ROUTE_RECV_INQUIRE_ERROR	An error occurred during obtaining. Check whether the local has entered the room and whether the SDK has been initialized.

ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE

Receiving Types	Effects
AUDIO_ROUTE_NOT_RECV_FROM_ANYONE	The local does not receive any audio, which is equivalent to disable the speaker in the room.
AUDIO_ROUTE_RECV_FROM_ALL	The local receives everyone’s audio.
AUDIO_ROUTE_RECV_BLACK_LIST	The local does not receive the audio of people in the blocklist.
AUDIO_ROUTE_RECV_WHITE_LIST	The local will only receive the audio of people in the allowlist.
AUDIO_ROUTE_RECV_INQUIRE_ERROR	An error occurred during obtaining. Check whether the local has entered the room and whether the SDK has been initialized.

⚠ Note :

Please do not use `AUDIO_ROUTE_RECV_INQUIRE_ERROR` in the `SetServerAudioRouteSendOperateType` API.

Accompaniment in Voice Chat

Last updated : 2020-10-10 10:09:01

This document describes the GME APIs for accompaniment in voice chat so that developers can easily integrate and debug them.

APIs for Accompaniment in Voice Chat

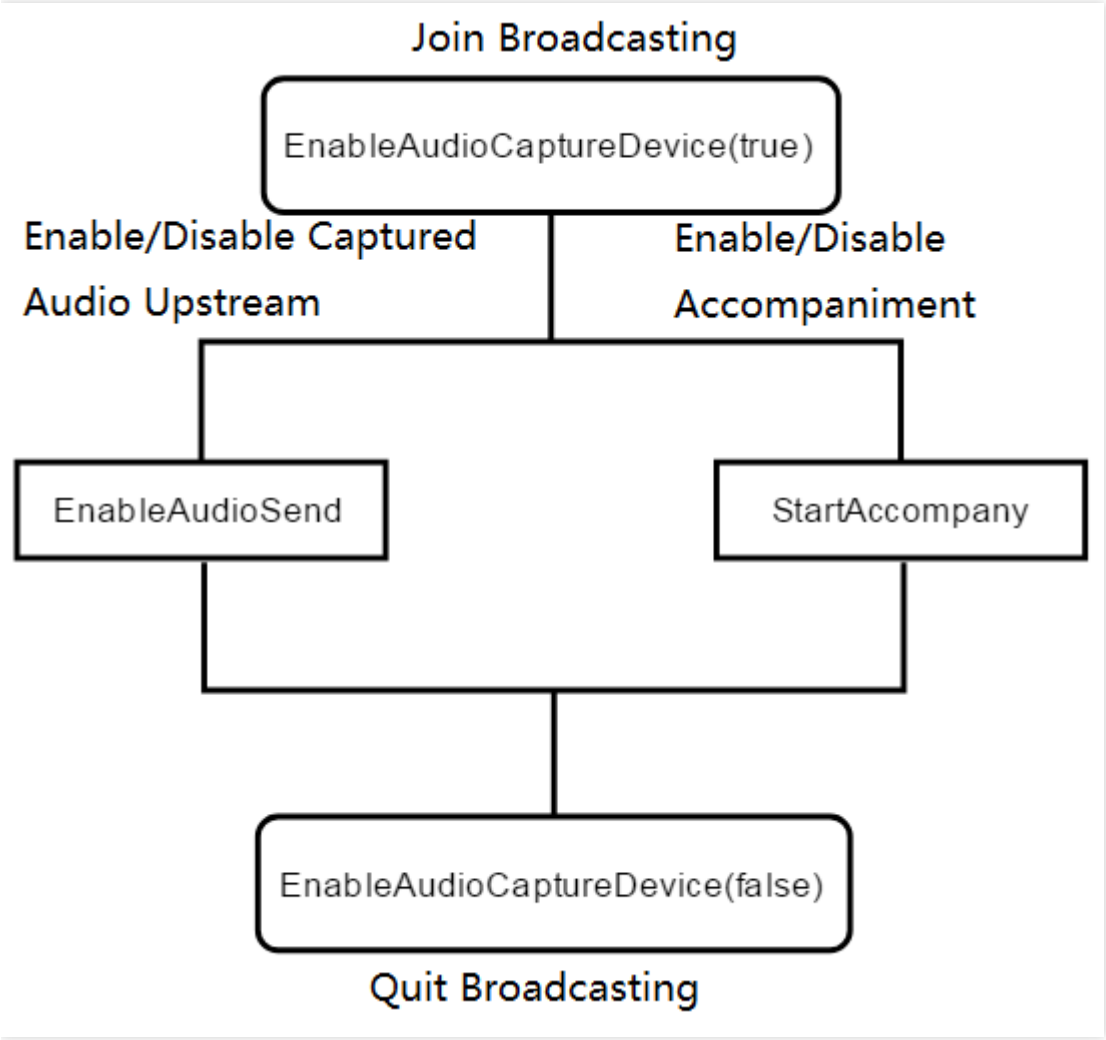
API	Description
StartAccompany	Starts playing back the accompaniment.
StopAccompany	Stops playing back the accompaniment.
IsAccompanyPlayEnd	Indicates whether the accompaniment is over.
PauseAccompany	Pauses playing back the accompaniment.
ResumeAccompany	Resumes playing back the accompaniment.
SetAccompanyVolume	Sets the accompaniment volume.
GetAccompanyVolume	Obtains the accompaniment volume.
SetAccompanyFileCurrentPlayedTimeByMs	Sets the playback progress.

Note :

To use accompaniment in voice chat, you must integrate the GME SDK and enable real-time voice chat.

How to call the APIs for social networking apps

The diagram below shows how to call the APIs for social networking apps:



How to use with EnableAudioCaputreDevice

After you enter a voice chat room, call `EnableAudioCaputreDevice` to enable the capturing device, and call `StartAccompany` to play back the accompaniment. To capture human voices via the microphone, you should call `EnableAudioSend` to enable the microphone first.

Starting playing back the accompaniment

This API (StartAccompany) is used to start playing back the accompaniment in M4A, WAV, or MP3 format. Calling this API resets the volume.

Function prototype

```
ITMGAudioEffectCtrl virtual int StartAccompany(const char* filePath, bool loopBack, int loopCount, int msTime)
```

Parameter	Type	Description

filePath	char*	Path of the accompaniment file.
loopBack	bool	Indicates whether to output the accompaniment in a mixing mode. This is generally set to `true`, indicating that the audience can also hear the accompaniment.
loopCount	int	Indicates the number of loops. The value `-1` means an infinite loop.
msTime	int	Delay time

Sample code

```
// Code for Windows
ITMGContextGetInstance()->GetAudioEffectCtrl()->StartAccompany(filePath,true,-1,0);
// Code for Android
ITMGContext.GetInstance(this).GetAudioEffectCtrl().StartAccompany(filePath,true,loopCount,0);
// Code for iOS
[[[ITMGContext GetInstance] GetAudioEffectCtrl] StartAccompany:path loopBack:isLoopBack loopCoun
t:loopCount msTime:0];
```

Starting the playback of the accompaniment being downloaded

This API (StartAccompanyDownloading) is used to start playing back the accompaniment that is being downloaded.
When you are downloading the accompaniment using code, you can pass the path of your accompaniment file as a parameter to `StartAccompanyDownloading`, which plays back the accompaniment as it is downloaded. `fileSize` is the estimated size of the complete file.
To pass a file being downloaded into this API, please make sure that the file size is at least 10,000 KB.

Function prototype

```
ITMGAudioEffectCtrl virtual int StartAccompany(const char* filePath, bool loopBack, int loopCount
, int msTime, int fileSize)
```

Note :

For the iOS client, follow the steps below:

1. To use this feature on iOS, you need to [download the mp3 library](https://picture-1256313114.cos.ap-beijing.myqcloud.com/mp3_codec.zip?_ga=1.162366908.1422691217.1594629603) and import it into your project.

2. Import the downloaded library and add it through “Link Binary With Libraries”.
3. Add the header file “TMGEngine_adv.h” to the same directory as the other SDK header files in your project.

Callback for the accompaniment playback

After the accompaniment is over, call the function `OnEvent`, and the event message ITMG_MAIN_EVENT_TYPE_ACCOMPANY_FINISH will be returned.
The returned parameter `data` includes “result” and “file_path”.

Sample code

```
void TMGTestScene::OnEvent(ITMG_MAIN_EVENT_TYPE eventType,const char* data){
switch (eventType) {
case ITMG_MAIN_EVENT_TYPE_ENTER_ROOM:
{
// Process
break;
}
...
case ITMG_MAIN_EVENT_TYPE_ACCOMPANY_FINISH:
{
// Process
break;
}
}
}
```

Stopping the accompaniment playback

This API (StopAccompany) is used to stop playing back the accompaniment.

Function prototype

```
ITMGAudioEffectCtrl virtual int StopAccompany(int duckerTime)
```

Parameter	Type	Description
duckerTime	int	Ducking time

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->StopAccompany(0);
```

Indicating whether the accompaniment is over

If it is over, `true` is returned. If it is not, `false` is returned.

Function prototype

```
ITMGAudioEffectCtrl virtual bool IsAccompanyPlayEnd()
```

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->IsAccompanyPlayEnd();
```

Pausing the accompaniment playback

This API (PauseAccompany) is used to pause the accompaniment playback.

Function prototype

```
ITMGAudioEffectCtrl virtual int PauseAccompany()
```

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->PauseAccompany();
```

Resuming the accompaniment playback

This API (ResumeAccompany) is used to resume the accompaniment playback.

Function prototype

```
ITMGAudioEffectCtrl virtual int ResumeAccompany()
```

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->ResumeAccompany();
```

Specifying whether the speaker can hear the accompaniment

This API is used to specify whether the speaker can hear the accompaniment.

Function prototype

```
ITMGAudioEffectCtrl virtual int EnableAccompanyPlay(bool enable)
```

Parameter	Type	Description
enable	bool	Indicates whether the audience can hear the accompaniment.

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->EnableAccompanyPlay(false);
```

Specifying whether the audience can hear the accompaniment

This API is used to specify whether the audience can hear the accompaniment.

Function prototype

```
ITMGAudioEffectCtrl virtual int EnableAccompanyLoopBack(bool enable)
```

Parameter	Type	Description
enable	bool	Indicates whether the audience can hear the accompaniment.

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->EnableAccompanyLoopBack(false);
```

Setting the accompaniment volume

This API (SetAccompanyVolume) is used to set the accompaniment volume. Value range: 0 - 200. The default value is 100. A value greater than 100 means volume up, while a value less than 100 means volume down.

Function prototype

```
ITMGAudioEffectCtrl virtual int SetAccompanyVolume(int vol)
```

Parameter	Type	Description
vol	int	Specifies the volume value.

Sample code

```
int vol=100;
ITMGContextGetInstance()->GetAudioEffectCtrl()->SetAccompanyVolume(vol);
```

Obtaining the accompaniment volume

This API (GetAccompanyVolume) is used to obtain the accompaniment volume.

Function prototype

```
ITMGAudioEffectCtrl virtual int GetAccompanyVolume()
```

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->GetAccompanyVolume();
```

Obtaining the accompaniment playback progress

This action requires using both of these two APIs: `GetAccompanyFileTotalTimeByMs` and `GetAccompanyFileCurrentPlayedTimeByMs`. Please note that Current/Total = current loop times, and Current % Total = current loop playback position.

Function prototype

```
ITMGAudioEffectCtrl virtual int GetAccompanyFileTotalTimeByMs()
ITMGAudioEffectCtrl virtual int GetAccompanyFileCurrentPlayedTimeByMs()
```

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->GetAccompanyFileTotalTimeByMs();
ITMGContextGetInstance()->GetAudioEffectCtrl()->GetAccompanyFileCurrentPlayedTimeByMs();
```

Setting the playback progress

This API (SetAccompanyFileCurrentPlayedTimeByMs) is used to set the playback progress.

Function prototype

```
ITMGAudioEffectCtrl virtual int SetAccompanyFileCurrentPlayedTimeByMs(unsigned int time)
```

Parameter	Type	Description
time	int	Specifies the playback progress in milliseconds

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->SetAccompanyFileCurrentPlayedTimeByMs(time);
```

Setting the accompaniment key

This API (SetAccompanyKey) is used to specify the accompaniment key, and should be called before starting the accompaniment playback.

Function prototype

```
ITMGAudioEffectCtrl virtual int SetAccompanyKey(int nKey)
```

Parameter	Type	Description
nKey	int	Key(s) to adjust by. Value range (recommended): -4 to 4, where `0` indicates using the original key.

Error Codes

Error Message	Error Code	Description	Solution
QAV_ERR_ACC_OPENFILE_FAILED	4001	Failed to open the file	Checks whether the file or its path exists, and whether you have access to the file.
QAV_ERR_ACC_FILE_FORAMT_NOTSUPPORT	4002	Invalid file format	Checks whether the file format is correct.
QAV_ERR_ACC_DECODER_FAILED	4003	Decoding failure	Checks whether the file format is correct.
QAV_ERR_ACC_BAD_PARAM	4004	Invalid parameter	Checks whether the parameters in the code are correct.
QAV_ERR_ACC_MEMORY_ALLOC_FAILED	4005	Memory allocation failed	System resources have run out. If this error persists, please submit a ticket for assistance.
QAV_ERR_ACC_CREATE_THREAD_FAILED	4006	Failed to create a	System resources have run out. If this error

		thread	persists, please submit a ticket for assistance.
QAV_ERR_ACC_STATE_ILLIGAL	4007	Invalid state	This error occurs if an API is called in a state that does not allow calling.

Real-time Sound Effect

Last updated : 2019-11-01 11:18:59

This document describes the integration for real-time sound effect in details to help developers debug and integrate APIs for Tencent Cloud's Game Multimedia Engine (GME).

Real-time sound effect APIs

API	Description
PlayEffect	Play sound effect
PauseEffect	Pause sound effect
PauseAllEffects	Pause all sound effects
ResumeEffect	Resume sound effect
ResumeAllEffects	Resume all sound effects
StopEffect	Stop sound effect
StopAllEffects	Stop all sound effects
SetVoiceType	Set voice change effect
SetKaraokeType	Set karaoke effect
GetEffectsVolume	Obtain volume of sound effect
SetEffectsVolume	Set volume of sound effect

Play sound effect

The PlayEffect API is used to play sound effect. The sound ID, which represents an independent playback event, should be managed in the App. The playback can be controlled by this ID. The file format supports m4a, wav, and mp3.

Function prototype

```
ITMGAudioEffectCtrl virtual int PlayEffect(int soundId, const char* filePath, bool loop, double pitch, double pan, double gain)
```

Parameter	Type	Description
soundId	int	Sound ID
filePath	char*	Sound path
loop	bool	Enable sound loop or not
pitch	double	Playback frequency. The default value is 1.0. Smaller value means slower playback speed and longer time.
pan	double	The channel, with a value ranging from -1.0 to 1.0. -1.0 means that only left channel is enabled.
gain	double	Gain volume, with a value ranging from 0.0 to 1.0. The default value is 1.0.

Sample code

```
double pitch = 1.0;
double pan = 0.0;
double gain = 0.0;
//Windows
ITMGContextGetInstance()->GetAudioEffectCtrl()->PlayEffect(soundId,filepath,true,pitch,pan,gain);
//Android
ITMGContext.GetInstance(this).GetAudioEffectCtrl().PlayEffect(soundId,filePath,loop);
//iOS
[[[ITMGContext GetInstance] GetAudioEffectCtrl] PlayEffect:soundId filePath:path loop:isLoop];
```

Pause sound effect

The PauseEffect API is used to pause sound effect.

Function prototype

```
ITMGAudioEffectCtrl virtual int PauseEffect(int soundId)
```

Parameter	Type	Description
soundId	int	Sound ID

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->PauseEffect(soundId);
```

Pause all sound effects

The PauseAllEffects API is used to pause all sound effects.

Function prototype

```
ITMGAudioEffectCtrl virtual int PauseAllEffects()
```

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->PauseAllEffects();
```

Resume sound effect

The ResumeEffect API is used to resume sound effect.

Function prototype

```
ITMGAudioEffectCtrl virtual int ResumeEffect(int soundId)
```

Parameter	Type	Description
soundId	int	Sound ID

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->ResumeEffect(soundId);
```

Resume all sound effects

The ResumeAllEffects API is used to resume all sound effects.

Function prototype

```
ITMGAudioEffectCtrl virtual int ResumeAllEffects()
```

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->ResumeAllEffects();
```

Stop sound effect

The StopEffect API is used to stop sound effect.

Function prototype

```
ITMGAudioEffectCtrl virtual int StopEffect(int soundId)
```

Parameter	Type	Description
soundId	int	Sound ID

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->StopEffect(soundId);
```

Stop all sound effects

The StopAllEffects API is used to stop all sound effects.

Function prototype

```
ITMGAudioEffectCtrl virtual int StopAllEffects()
```

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->StopAllEffects();
```

Set voice effect

The SetVoiceType API is used to set voice change effect.

Function prototype

```
TMGAudioEffectCtrl int setVoiceType(int type)
```

Parameter	Type	Description
type	int	Indicates the voice change type of the audio in local end

Parameter type	Value	Description
ITMG_VOICE_TYPE_ORIGINAL_SOUND	0	Original
ITMG_VOICE_TYPE_LOLITA	1	Lolita
ITMG_VOICE_TYPE_UNCLE	2	Uncle

Parameter type	Value	Description
ITMG_VOICE_TYPE_INTANGIBLE	3	Soul
ITMG_VOICE_TYPE_DEAD_FATBOY	4	Homebody
ITMG_VOICE_TYPE_HEAVY_MENTA	5	Heavy metal
ITMG_VOICE_TYPE_DIALECT	6	Foreign
ITMG_VOICE_TYPE_INFLUENZA	7	Influenza
ITMG_VOICE_TYPE_CAGED_ANIMAL	8	Animal
ITMG_VOICE_TYPE_HEAVY_MACHINE	9	Machine
ITMG_VOICE_TYPE_STRONG_CURRENT	10	Strong current
ITMG_VOICE_TYPE_KINDER_GARTEN	11	Kid
ITMG_VOICE_TYPE_HUANG	12	Minions

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->setVoiceType(0);
```

Set karaoke effect

The SetKaraokeType API is used to set karaoke effect.

Function prototype

```
TMGAudioEffectCtrl int SetKaraokeType(int type)
```

Parameter	Type	Description
type	int	Indicates the voice change type of the audio in local end

Parameter type	Value	Description
ITMG_KARAOKE_TYPE_ORIGINAL	0	Original
ITMG_KARAOKE_TYPE_POP	1	Popular
ITMG_KARAOKE_TYPE_ROCK	2	Rock

Parameter type	Value	Description
ITMG_KARAOKE_TYPE_RB	3	Hip hop
ITMG_KARAOKE_TYPE_DANCE	4	Dance
ITMG_KARAOKE_TYPE_HEAVEN	5	Soul
ITMG_KARAOKE_TYPE_TTS	6	Voice synthesis

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->SetKaraokeType(0);
```

Obtain volume of sound effect

The GetEffectsVolume API is used to obtain volume of sound effect. It is linear volume with a default value of 100. If the value is greater than 100, the volume increases; and if the value is less than 100, the volume is decreases.

Function prototype

```
ITMGAudioEffectCtrl virtual int GetEffectsVolume()
```

Sample code

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->GetEffectsVolume();
```

Set volume of sound effect

The SetEffectsVolume API is used to set volume of sound effect.

Function prototype

```
ITMGAudioEffectCtrl virtual int SetEffectsVolume(int volume)
```

Parameter	Type	Description
volume	int	Sound effect volume

Sample code

```
int volume=1;
ITMGContextGetInstance()->GetAudioEffectCtrl()->SetEffectsVolume(volume);
```