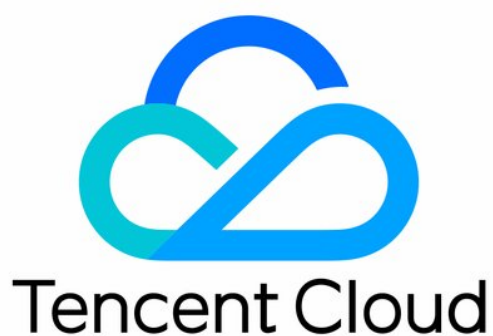


Game Multimedia Engine

拡張機能開発ガイド

製品ドキュメント



Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

カタログ：

拡張機能開発ガイド

サーバー側のレコーディング

全量レコーディング

カスタムレコーディング

レコーディングコールバックの説明

レンジボイス

3Dサウンド

効果音と伴奏

ボイスチェンジ

リアルタイム音声のBGM

リアルタイムなサウンドイコライザ

リアルタイムカラオケ機能

ネットワークオーディオストリーム転送ルーティング

カスタムメッセージチャネル

社内ファイアウォール制限への対応について

Language Parameter Reference List

GMEルーム管理機能の導入

拡張機能開発ガイド

サーバー側のレコーディング

全量レコーディング

最終更新日：2024-01-18 15:47:47

本書では、フルレコーディングで**GMEサーバー側のレコーディング機能**に素早くアクセスする方法を説明します。

運用シーン

GMEはリアルタイム音声ストリーム向けの**サーバー側のレコーディング機能**を提供し、開発者がコンテンツの保存/管理/二次創作などのシーンを実装することに役立ちます。フルレコーディング：アプリケーションにおけるすべての音声ルームに対して、ルームによるミックスストリーミングとユーザーによるシングルストリーミングをレコーディングすることをサポートします。カスタムレコーディング：ユーザーが指定したルームに対して、ルームによるミックスストリーミングとユーザーによるシングルストリーミングをレコーディングすることをサポートします。レコーディングしたオーディオファイルは、ご利用のアカウント配下の**COS**サービスに保存されます。

本書では、**フルレコーディング**の開発・アクセス方法のみを説明します。アプリケーションに対して、カスタムレコーディングを有効にするには、[開発ガイド-カスタムレコーディング](#)をご参照ください。

注意：

GMEサーバー側のレコーディング機能を使用すると、レコーディング中にGMEでレコーディングサービスの使用料金が発生します。GME国際サイトのレコーディングサービスは、2023年4月1日より正式に課金を開始いたします。詳細な料金情報は、事前に[GME購入ガイド](#)で公開いたします。

レコーディングしたファイルは、ご利用のTencent Cloudアカウント配下の**COS**サービスに保存されます。**COS**請求書は、実際のストレージ使用量、保存期間、アクセス頻度などに応じて作成されます。料金情報の詳細については、[COS料金説明](#)をご参照ください。

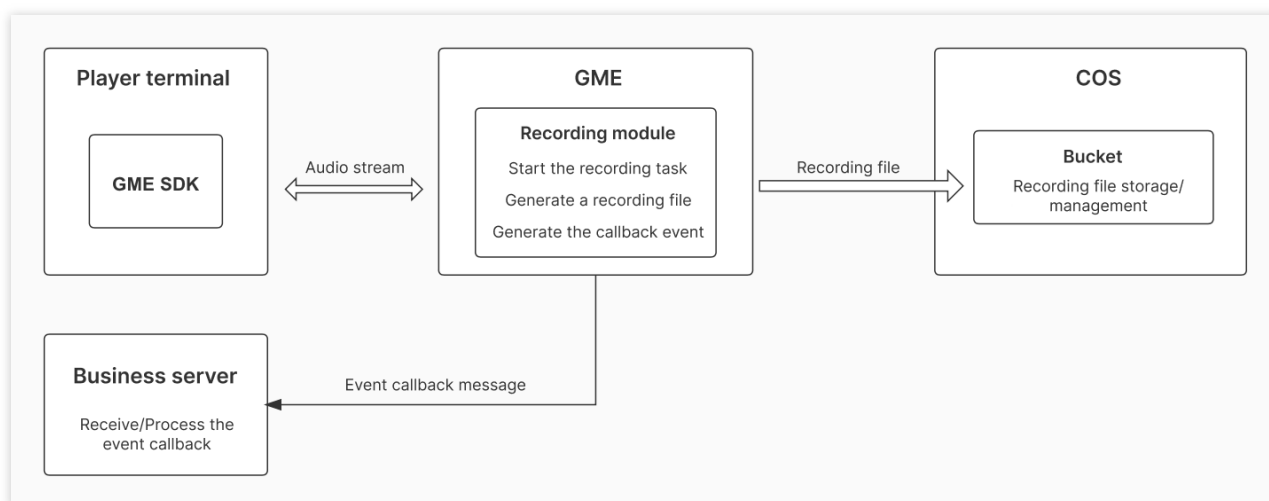
前提条件

**リアルタイム音声サービスが有効になっていること：[サービス有効化ガイド](#)をご参照ください。

サーバー側のレコーディングサービスが有効になっていること：現在、サーバー側のレコーディングサービスは、ホワイトリストユーザーだけに提供しています。ホワイトリストを有効化するには、こちらに連絡してください。

GME SDK導入済み：コアインターフェースとリアルタイム音声インターフェースの導入を含みます。詳細については、[Native SDKクイックスタート](#)、[Unity SDKクイックスタート](#)、[Unreal SDKクイックスタート](#)をご参照ください。

サービスアーキテクチャ



機能説明

1. レコーディング範囲

フルレコーディングを有効にすると、すべてのリアルタイム音声ルームがレコーディングされます。レコーディングは、ルームミックスストリーミングのみ、ユーザーシングルストリーミングのみ、シングルストリーミングとミックスストリーミング両方を設定できます。

2. レコーディングメカニズム

レコーディングタスクの起動メカニズム

最初のユーザーがルームに参加すると、レコーディングタスクが開始します

レコーディングタスクの停止メカニズム

最後のユーザーがルームを退出すると、レコーディングタスクが終了します

レコーディングタスクのマルチパートメカニズム

単一オーディオファイルの長さが2時間になると、オーディオファイルを自動的にマルチパートに分割します

ユーザーがマイクの接続を切断すると、ユーザーシングルストリーミングレコーディングオーディオを自動的にマルチパートに分割します。ユーザーがマイクの接続を確立すると、新しいマルチパートが作成されます。レコーディング中のタスクが異常で中断した場合、タスクが自動的に再接続すると、新しいマルチパートが作成されます。

レコーディングタスクのイベント通知メカニズム

レコーディングタスクのイベントは、コールバックメカニズムにより、設定されたコールバックアドレスに通知されます。**レコーディング開始**、**レコーディング停止**、**レコーディングファイルアップロード完了**、これらのイベントが発生した場合、コールバック通知を受信します。

コールバック情報の詳細については、[レコーディングコールバックの説明](#)をご参照ください。

3. 保存先

GMEサーバー側のレコーディングが完了した後、レコーディングしたオーディオファイルは、ご利用のアカウント配下の**COS**サービスの指定されたバケットに保存されます。バケットのリージョンは指定する必要があります。指定可能なリージョンのリストについては、[COSリージョンの説明](#)をご参照ください。

4. レコーディングファイルのフォーマット

.mp3

5. レコーディングファイルの命名規則

ユーザーシングルレコーディングファイル：bizid_roomid_userid/\${タスクの開始時間}_\${id}_audio.mp3

ルームミックスストリーミングレコーディングファイル：bizid_roomid/\${taskid}_\${タスクの開始時間}_\${id}_audio.mp3

bizid: GMEアプリケーションID。[GMEコンソール](#)から取得できます。

roomid: 音声ルームID。リアルタイム音声サービスを使用する時に定義してGME SDKに渡します。

userid: プレイヤーID。リアルタイム音声サービスを使用する時に定義してGME SDKに渡します。

taskid: レコーディングタスクID。GMEレコーディングサービスにより生成されます。レコーディングタスクごとに1つの特殊なタスクIDがあります。

id: レコーディングタスクのマルチパートのシリアル番号。シリアル番号は0から始まります。

アクセス手順

<dx-steps>

-<dx-tag-link link="#StartRealTimeASR" tag="业务側">コンソールでのレコーディングサービスの設定</dx-tag-link>

-<dx-tag-link link="#callback" tag="业务側">レコーディングタスクのコールバック受信（オプション）</dx-tag-link>

-<dx-tag-link link="#result" tag="业务側">レコーディングファイルの確認/管理</dx-tag-link>

</dx-steps>

ステップ1：コンソールでのレコーディングサービスの設定

レコーディングサービスの有効化/無効化

[GMEコンソール](#)にログインし、【サービス管理】メニューを開き、レコーディングサービスを有効にするアプリケーションの【設定】をクリックし、アプリケーションの詳細ページに進みます。ページで【音声レコーディングサービス】の**変更**をクリックします。

Voice Recording Service

Recording enable/disable ☒ Enable ☐ Disable

Store recording to [Bind](#)

Callback URL [Modify](#)

Recording mode ☐ Custom recording ☒ Full recording

☒ Single-stream recording by user ☒ Mixed-stream recording by room

☐ I have read and agree to [Billing Description for Voice Recording Service](#) [🔗](#).

SaveCancel

レコーディングスイッチに**有効**を設定します。

レコーディングサービスを初めて有効にした場合、GMEはご利用の**COSサービス**にアクセスする許可を要求します。ポップアップしたダイアログで許可すると、サーバー側のレコーディングサービスが有効になります。

Service Authorization ×

Other cloud service features will be used when performing operations related to this service.
Please create Service-Linked roles for **GME** and authorize it to use other cloud services. The relevant information is as follows:

Role Name	GME_QCSLinkedRoleInGameMedia (Service-Linked roles)
Role Description	The current role is the GME service linked role, which will access your other service resources within the scope of the permissions of the associated policy.
(Preset) Policy	QcloudAccessForGMELinkedRoleInGameMedia ⓘ

Authorize Cancel

レコーディングファイルのストレージ設定

アプリケーションの詳細ページで、【音声レコーディングサービス】の**変更**をクリックし、**レコーディングファイルのバケット**で**バインディング**をクリックします。

バケットのバインディングダイアログで、既存COSバケット（既存バケットは**COSコンソール**で事前に作成する必要があります）のバインディングまたはバケットの新規作成ができます。

Bind bucket ×

File storage is supported by COS. The created buckets will be synced to COS.

Current bucket

Bucket type ☒ Existing COS bucket ☐ Create

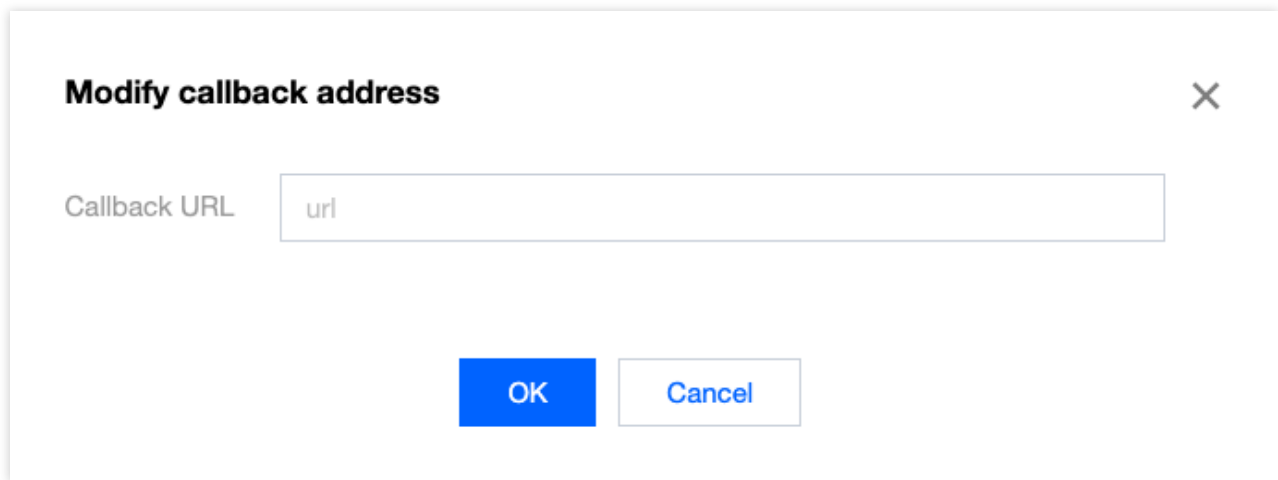
Bucket name

OK Cancel

レコーディングイベントのコールバック設定（オプション）

レコーディングサービスのイベントコールバックを受信するには、コールバックアドレスを設定する必要があります。操作パス：アプリケーションの詳細ページで、【音声レコーディングサービス】の**変更**をクリックし、**コー**

ルバックアドレスで**変更**をクリックし、ポップアップしたダイアログにコールバックを受信するurlアドレスを入力します。現在、レコーディングタスクの完了状態のみに対して、イベントコールバックのメッセージをプッシュします。

A dialog box titled "Modify callback address" with a close button (X) in the top right corner. It contains a label "Callback URL" and a text input field with the placeholder text "url". At the bottom, there are two buttons: "OK" (blue) and "Cancel" (white with blue border).

レコーディング範囲の設定

レコーディング範囲は、カスタムレコーディングとフルレコーディングを選択できます。フルを選択し、チェックボックスでシングルストリーミングのレコーディングとミックスストリーミングのレコーディングを指定してください。

上記のように設定し、**保存**をクリックすると、レコーディングサービスが有効になります。レコーディングサービスを使用する必要がある場合、予定外の費用の発生を防ぐために、コンソールでレコーディングサービスのスイッチを**無効**にしてください。

ステップ2：レコーディングタスクのコールバック受信（オプション）

ステップ1でコールバックアドレスを設定した場合、レコーディングタスクのイベントコールバックを受信することになります。

ステップ3：レコーディングファイルの確認/管理

完全なオーディオファイルはレコーディングタスクが終了して数分以内に作成できます。ご利用の[COSコンソール](#)で、レコーディングファイルを確認、管理できます。

カスタムレコーディング

最終更新日：2024-01-18 15:47:47

本書では、カスタムレコーディングで**GMEサーバー側のレコーディング機能**にアクセスする方法を説明します。

運用シーン

GMEはリアルタイム音声ストリーム向けの**サーバー側のレコーディング機能**を提供し、開発者がコンテンツの保存/管理/二次創作などのシーンを実装することに役立ちます。フルレコーディング：アプリケーションにおけるすべての音声ルームに対して、ルームによるミックスストリーミングとユーザーによるシングルストリーミングをレコーディングすることをサポートします。カスタムレコーディング：ユーザーが指定したルームに対して、ルームによるミックスストリーミングとユーザーによるシングルストリーミングをレコーディングすることをサポートします。レコーディングしたオーディオファイルは、ご利用のアカウント配下の**COS**サービスに保存されます。

本書では、**カスタムレコーディング**の開発・アクセス方法のみを説明します。アプリケーションに対して、フルレコーディングを有効にするには、[開発ガイド-フルレコーディング](#)をご参照ください。

ご注意：

GMEサーバー側のレコーディング機能を使用すると、レコーディング中にGMEでレコーディングサービスの使用料金が発生します。GME国際サイトのレコーディングサービスは、2023年4月1日より正式に課金を開始いたします。詳細な料金情報については、[GME購入ガイド](#)をご参照ください。

レコーディングしたファイルは、ご利用のTencent Cloudアカウント配下の**COS**サービスに保存されます。**COS**請求書は、実際のストレージ使用量、保存期間、アクセス頻度などに応じて作成されます。料金情報の詳細については、[COS料金説明](#)をご参照ください。

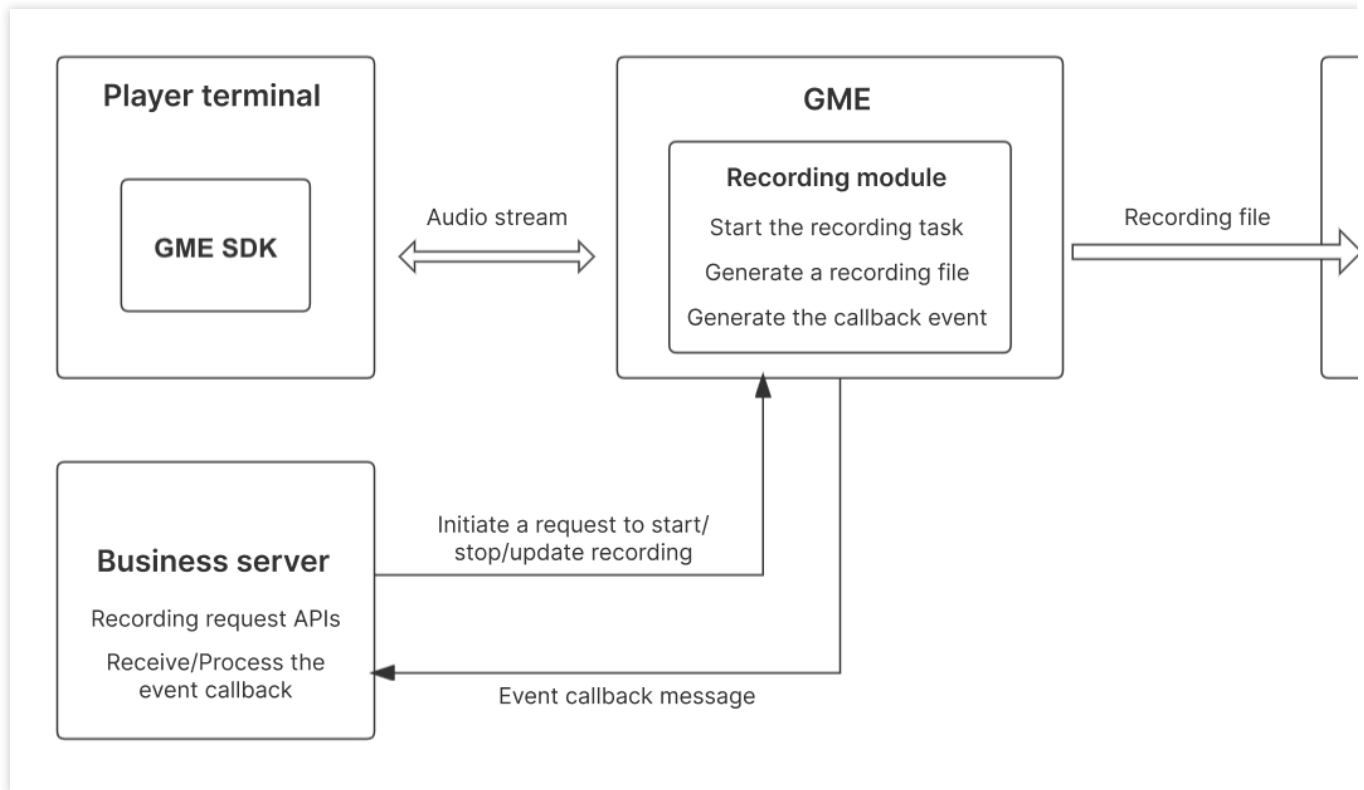
前提条件

リアルタイム音声サービスが有効になっていること：[サービス有効化ガイド](#)をご参照ください。

サーバー側のレコーディングサービスが有効になっていること：現在、サーバー側のレコーディングサービスは、ホワイトリストユーザーだけに提供しています。ホワイトリストを有効化するには、こちらに連絡してください。

GME SDK導入済み：コアインターフェースとリアルタイム音声インターフェースの導入を含みます。詳細については、[Native SDKクイックスタート](#)、[Unity SDKクイックスタート](#)、[Unreal SDKクイックスタート](#)をご参照ください。

サービスアーキテクチャ



機能説明

1. レコーディング範囲

サーバー側のインターフェースで、レコーディングするルームIDのミックスストリーミングまたはルーム内ユーザーのシングルストリーミングを指定できます。

レコーディングを指定したルームIDに対して、サーバー側のインターフェースで、レコーディングするプレイヤーホワイトリストまたはレコーディングしないプレイヤーブラックリストを指定できます。

2. 関連インターフェース

StartRecord()レコーディング起動

このインターフェースのパラメータで、シングルストリーミングまたはミックスストリーミングのレコーディング、レコーディングするRoomID、サブスクリプションのホワイトリストユーザーIDまたはブラックリストユーザーIDを指定できます。

StopRecord()レコーディング停止

このインターフェースを使用し、指定したtaskidのレコーディングを停止できます。taskidを記録しなかった場合、DescribeTaskInfo()を呼び出して、指定したルームで実行しているレコーディングタスクのtaskidを取得する必要があります。

ModifyRecordInfo()レコーディング情報の更新

このインターフェースを使用し、指定したtaskidのレコーディング情報を更新できます。具体的には、レコーディングタイプ、サブスクリプションのホワイトリストユーザーIDまたはブラックリストユーザーIDを更新できます。taskidを記録しなかった場合、DescribeTaskInfo()を呼び出して、指定したルームで実行しているレコーディングタスクのtaskidを取得する必要があります。

DescribeTaskInfo()ルームのレコーディング情報の確認

このインターフェースを使用し、指定したルームのレコーディング情報を確認できます。具体的には、実行中のレコーディングタスクのtaskid、サブスクリプションのホワイトリストユーザーIDまたはブラックリストユーザーIDを確認できます。

DescribeRecordInfo()レコーディングタスク情報の確認

このインターフェースを使用し、指定したtaskidのタスク情報を確認できます。具体的には、レコーディングタイプ、レコーディングしたルームID、レコーディングしたユーザーID、レコーディングファイルの情報を確認できます。

3.レコーディングメカニズム

レコーディングタスクの起動メカニズム

StartRecord() インターフェースを呼び出すと、指定したルームのレコーディングタスクが起動します。

レコーディングタスクの停止メカニズム

StopRecord() インターフェースを呼び出すと、指定したルームのレコーディングタスクが停止します。

録音ファイルの作成タイミング

ルームミックスストリーミングの録音ファイル：ルームのレコーディングタスクが起動した後、ルームにユーザーがすでに存在している場合、ミックスストリーミングオーディオファイルが直ちに作成されます。ルームにユーザーが存在しない場合、最初のユーザーがルームに入った直後に、ミックスストリーミングオーディオファイルが作成されます。

ユーザーシングルミックスストリーミングの録音ファイル：ルームのレコーディングタスクが起動した後、レコーディング範囲内のユーザーがルームに入ると、このユーザーのシングルストリーミングオーディオファイルが作成されます。

レコーディングタスクのマルチパートメカニズム

単一オーディオファイルの長さが2時間になると、オーディオファイルを自動的にマルチパートに分割します。ユーザーがマイクの接続を切断すると、ユーザーシングルストリーミングレコーディングオーディオを自動的にマルチパートに分割します。ユーザーがマイクの接続を確立すると、新しいマルチパートが作成されます。

レコーディング中のタスクが異常で中断した場合、タスクが自動的に再接続すると、新しいマルチパートが作成されます

レコーディングタスクのイベント通知メカニズム

レコーディングタスクのイベントは、コールバックメカニズムにより、設定されたコールバックアドレスに通知されます。**レコーディング開始**、**レコーディング停止**、**レコーディングファイルアップロード完了**、これらのイベントが発生した場合、コールバック通知を受信します

コールバック情報の詳細については、[レコーディングコールバックの説明](#)をご参照ください

4.保存先

GMEサーバー側のレコーディングが完了した後、レコーディングしたオーディオファイルは、ご利用のアカウント配下の**COS**サービスの指定されたバケットに保存されます。バケットのリージョンは指定する必要があります。指定可能なリージョンのリストについては、[COSリージョンの説明](#)をご参照ください。

5.レコーディングファイルのフォーマット

.mp3

6.レコーディングファイルの命名規則

ユーザーシングルレコーディングファイル：bizid_roomid_userid/\${タスクの開始時間}_\${id}_audio.mp3

ルームミックスストリーミングレコーディングファイル：bizid_roomid/\${taskid}/\${タスクの開始時間}_\${id}_audio.mp3

bizid: GMEアプリケーションID。[GMEコンソール](#)から取得できます

roomid: 音声ルームID。リアルタイム音声サービスを使用する時に定義してGME SDKに渡します

userid: プレイヤーID。リアルタイム音声サービスを使用する時に定義してGME SDKに渡します

taskid: レコーディングタスクID。GMEレコーディングサービスにより生成されます。レコーディングタスクを正常に呼び出すたびに、1つ特殊なタスクIDが生成されます

id: レコーディングタスクのマルチパートのシリアル番号。シリアル番号は0から始まります

アクセス手順

1. [コンソールでのレコーディングサービスの設定](#) 業務側
2. [サーバー側のインターフェースの呼出しによるレコーディング範囲の定義](#) 業務側
3. [レコーディングタスクのコールバック受信（オプション）](#) 業務側
4. [レコーディングファイルの確認/管理](#) 業務側

ステップ1：コンソールでのレコーディングサービスの設定

[GMEコンソール](#)にログインし、**サービス管理**メニューを開き、レコーディングサービスを有効にするアプリケーションの**設定**をクリックし、アプリケーションの詳細ページに進みます。

レコーディングサービスの有効化/無効化

Voice Recording Service

Recording enable/disable

☒ Enable ☐ Disable

Store recording to

[Bind](#)

Callback URL

[Modify](#)

Recording mode

☒ Custom recording ☐ Full recording

☐ I have read and agree to [Billing Description for Voice Recording Service](#) [🔗](#).

Save

Cancel

ページで**音声レコーディングサービスの変更**をクリックし、レコーディングスイッチを**有効**に設定します。レコーディングサービスを初めて有効にした場合、GMEはご利用の**COSサービス**にアクセスする許可を要求します。ポップアップしたダイアログで許可すると、サーバー側のレコーディングサービスが有効になります。

Service Authorization

Other cloud service features will be used when performing operations related to this service.

Please create Service-Linked roles for **GME** and authorize the it to use other cloud services. The relevant information is as follows:

Role Name	GME_QCSLinkedRoleInGameMedia (Service-Linked roles)
Role Description	The current role is the GME service linked role, which will access your other service resources within the scope of the permissions of the associated policy.
(Preset) Policy	QcloudAccessForGMELinkedRoleInGameMedia ⓘ

[Authorize](#)[Cancel](#)

レコーディングファイルのストレージ設定

レコーディングファイルのバケットでバインディングをクリックします。

バケットのバインディングダイアログで、既存COSバケット（既存バケットは[COSコンソール](#)で事前に作成する必要があります）のバインディングまたはバケットの新規作成ができます。

Bind bucket

File storage is supported by COS. The created buckets will be synced to COS.

Current bucket

Bucket type ☒ Existing COS bucket ☐ Create

Bucket name

OK

Cancel

レコーディングイベントのコールバック設定（オプション）

レコーディングサービスのイベントコールバックを受信するには、コールバックアドレスを設定する必要があります。操作パス：アプリケーションの詳細ページで、**音声レコーディングサービスの変更**をクリックし、**コールバックアドレスで変更**をクリックし、ポップアップしたダイアログにコールバックを受信するurlアドレスを入力します。現在、レコーディングタスクの完了状態のみに対して、イベントコールバックのメッセージをプッシュします。

Modify callback address

Callback URL

OK

Cancel

レコーディング範囲の設定

レコーディング範囲は、カスタムレコーディングとフルレコーディングを選択できます。この場合、カスタムレコーディングを選択してください。そうしないと、サーバー側に関連するインターフェースの呼出しに失敗します。

上記のように設定し、**保存**をクリックすると、レコーディングサービスが有効になります。レコーディングサービスを使用する必要がある場合、予定外の費用の発生を防ぐために、コンソールでレコーディングサービスのスイッチを**無効**にしてください。

ステップ2：レコーディングタスクのコールバック受信（オプション）

ステップ1でコールバックアドレスを設定すれば、レコーディングタスクのイベントコールバックを受信できます。

ステップ3：サーバー側のインターフェースの呼出しによるレコーディング範囲の定義

業務シーンの必要性に応じて、インターフェースを呼び出してください。呼出し順とプロセスの設計において、以下の点に注意してください：

- （1）既存のRoomIdに対するレコーディング開始リクエストのみをサポートします。RoomIdが存在しなければ、レコーディングタスクの作成に失敗します
- （2）自発的に呼び出してレコーディングを停止しなければ、ルームにユーザーがいる限り、レコーディングプロセスはずっと実行します。ルーム内のユーザーが全部退出した後、元のレコーディングタスクのプロセスは12時間保持されます。保持中に、ユーザーが改めてルームに入れば、レコーディングが自動的に開始します。レコーディングプロセスの保持時間が過ぎた後、ユーザーがルームに入っても、レコーディングは自動的に開始しません。

ステップ4：レコーディングファイルの確認/管理

完全なオーディオファイルはレコーディングタスクが終了して数分以内に作成できます。ご利用の[COSコンソール](#)で、レコーディングファイルを確認、管理できます。

レコーディングコールバックの説明

最終更新日：：2024-01-18 15:47:47

サーバー側レコーディングコールバックの説明

説明

ネットワークの影響を受け、サーバーで通知を受信した順番は、イベントの発生順番と異なる可能性があります。こちらから提供するサービスにはリトライメカニズムはありますが、すべてのメッセージが届く保証はできません。そのため、お客様の重要な業務ロジックがメッセージ通知サービスに依存しないようにすることをお勧めします。

ネットワークプロトコル

コンソールでコールバックアドレス、すなわち、HTTP(S)プロトコルインターフェースのURLを設定した場合、POSTメソッドをサポートし、データ転送にUTF-8エンコードを使用する必要があります。

HTTPヘッダーのパラメータ

名前	タイプ	必須か	説明
Signature	string	はい	署名。詳細については、以下の 署名生成 の説明をご参照ください

署名生成

Signature = HMAC-SH1 (strContent, SecretKey)

strContent：署名そのものの文字列であり、bodyのJSON内容（長さはContent-Lengthと同様）です。

body：業務にコールバックするJSON内容。以下の[コールバックの例](#)におけるすべての内容はbodyです。

SecretKey：キー。アプリケーションの権限キーで、[コンソール > アプリケーションの詳細](#)で参照できます。

HMAC-SH1：署名アルゴリズム。

コールバックのパラメータ

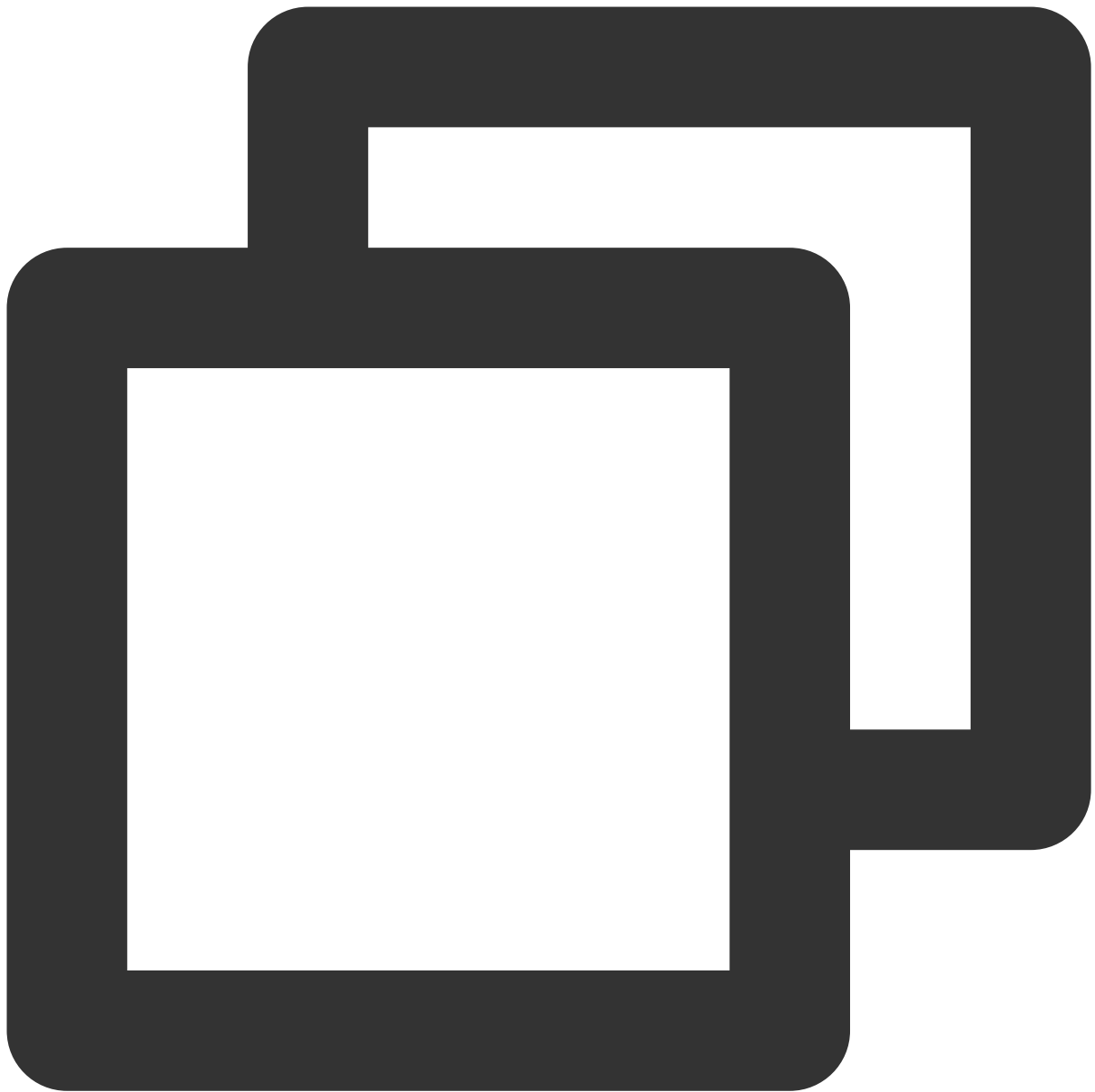
名前	タイプ	説明
BizID	Integer	アプリケーションのAppID。 コンソール > アプリケーションの詳細 で参照できます。
RoomID	String	ルームID
UserID	String	ユーザーID

RecordMode	Integer	レコーディングモード 0: シングルストーリーミング 1: ミックスストーリーミング
Timestamp	Integer	コールバックを送信した時のタイムスタンプ (s)
TaskID	Integer	クラウドレコーディングサービスに割り当てたタスクID。タスクIDは1回のレコーディングライフサイクルの一意の識別子であり、レコーディング終了後に意味がなくなります。カスタムレコーディングモードを使用した場合、タスクIDは、レコーディング開始時にレスポンスパラメータにより取得でき、次回このレコーディングタスクを操作する時のリクエストパラメータとして、業務を保存する必要があります。
EventType	Integer	イベントタイプ
Detail	EventDetail	イベント詳細。EventTypeでフォーマットが決まります

EventDetail イベントの詳細な説明

EventType	説明	Detail
1	オーディオファイルレコーディング開始	SeqNo: Number。マルチパートの番号 FileName: String。ファイル名
2	オーディオファイルレコーディング完了	SeqNo: Number。マルチパートの番号 FileName: String。ファイル名
3	オーディオファイルアップロード完了	SeqNo: Number。マルチパートの番号 FileName: String。ファイル名

コールバックの例



```
{
  "BizID":1400000000,
  "RoomID":"100",
  "UserID":"999",
  "TaskID":446946705284000000,
  "RecordMode":1,
  "Timestamp":1675930605,
  "EventType":1,
  "Detail":{
    "SeqNo":0,
    "FileName":"1400000000_100_999/2023-02-09-16-16-45_446946705284000000_audio
```

```
}  
}
```

レンジボイス

最終更新日：：2024-01-18 15:47:47

機能の説明

音声ルーム内では、ユーザーはある距離内にいる他のユーザーとリアルタイムで音声通話を行うことができます。この機能は、ビジネス層でGMEクライアントインターフェースを呼び出して音源の方位を更新します。サーバーに自端の位置を通知することを目的とします。**自端の世界座標+自端が受信した音声の範囲、他端の世界座標+他端が受信した音声の範囲**によって判断した後、サーバーでプレイヤーの範囲内の音声ストリームを転送し、デフォルトでは、プレイヤーから最も近い20本の音声ストリームに転送します。**最大数万人がルーム内に同時にマイクをオンにすることができます。**

ユースケース

PUBG ゲーム	PUBGゲーム、生存射撃モバイルゲーム 特有の「チームのみ」または「全員」のボイスモードを提供します。ゲームの設定で、プレイヤーが次の項目を選択できます： 1. 「チームのみ」モードを使用すると、チーム内のチームメイトが話している声だけが聞こえます。 2. 「すべての人」モードを使用すると、チーム内のチームメイトの声や、ある範囲内で他の対局者が話している声を聞くことができます。
没入型 バーチャ ルシーン	例えば ゲームのコンサート などのバーチャルシーンでは、レンジボイスを使ってボーカリストがバーチャルルーム内で歌い、聴衆全員が歌声を聞きながら、自分の一定範囲内の他の聴衆とコミュニケーションをとるようになっています。最高で同ルーム内の1万人以上が同時にマイクをオンにすることができます。

体験効果

[Demo体験](#)で体験プログラムをダウンロードし、3Dサウンドやレンジボイスの効果を体験することができます。

基本コンセプト

レンジボイス機能を使用するには、**音声モード**、**音声受信範囲**、**TeamID**という3つのコンセプトがあります。

ボイスモード

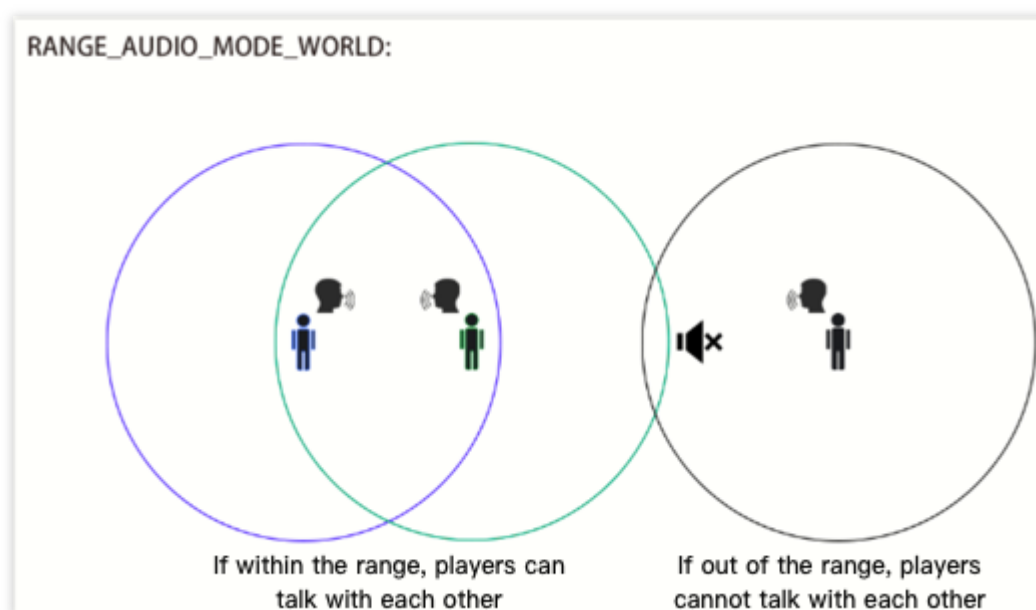
レンジボイスルームに参加した時に、2種類の音声モードを選択することが可能です。

ボイスモード	パラメータ名	機能
すべての人	RANGE_AUDIO_MODE_WORLD	設定されると、プレイヤーの近くにいる一定範囲の人がプレイヤーの話を聞くことができます。その範囲内にこのモードが設定されているプレイヤーがいれば、お互いに話をすることもできます。 チームメンバーがお互いに聞こえます
チームのみ	RANGE_AUDIO_MODE_TEAM	チームメンバーだけがお互いに聞こえます

ご注意：

チームメイト間の通話は、距離や音声モードに左右されません。

音声受信範囲



音声モードが**すべての人（RANGE_AUDIO_MODE_WORLD）**に設定されている場合、この場合の音声受信範囲はUpdateAudioRecvRangeインターフェースの影響を受けます。

次の2人のプレイヤーAとBが異なるチームで、音声モードが**すべての人（RANGE_AUDIO_MODE_WORLD）**に設定されているとします。

プレイヤー座標	音声受信範囲	他のプレイヤーとの音声到達可能状況	チームメンバーとの音声到達可能状況
A	10	プレイヤーBはプレイヤーAから10メートル以内に	同じチームのメンバー同士

(0,0,0)	メートル	るため、プレイヤーBの声を聞くことができます	の通話には影響しません
B (0,8,0)	5メートル	プレイヤーAとプレイヤーBは5メートル以上離れているため、プレイヤーAの声は聞こえません。	同じチームのメンバー同士の通話には影響しません

説明：

プレイヤー音声到達状況の詳細については[付属書](#)をご参照ください。

TeamID

範囲ボイスを使用するには、**SetRangeAudioTeamID**インターフェースを呼び出してチーム番号**TeamID**を設定し、**EnterRoom**インターフェースを呼び出してボイスルームに入る必要があります。

ルームに参加するときに指定された**TeamID != 0**の場合、範囲ボイスルームモードに切り替えます。メンバーが**TeamID=1**を使用してボイスルームに入った場合、そのボイスモードが**RANGE_AUDIO_MODE_TEAM**に設定されると、**TeamID=1**のメンバーだけが彼の声聞くことができます、設定された音声モードが**RANGE_AUDIO_MODE_WORLD**であれば、**TeamID=1**のメンバー以外の一定範囲のプレイヤーにもその音声を聞くことができます。

TeamIDの状況	ボイスモード	範囲	音声到達可能状況
TeamID != 0 で、仮に TeamID=1と します	RANGE_AUDIO_MODE_TEAM	10 メートル	TeamID=1のメンバーと通話できます
	RANGE_AUDIO_MODE_WORLD	10 メートル	TeamID=1のメンバーと、音声モードがRANGE_AUDIO_MODE_WORLDに設定されている同室10メートル以内のメンバーと通話できます

チームID=0を使用して音声ルームに入室したメンバーは、レンジボイスのホストモードとなり、ルーム内の全員（音声モードがすべての人であるか、チームのみであるかにかかわらず）がそのメンバーの音声を聞くことができます。

TeamIDの状況	TeamIDの変更タイミング	範囲	音声到達可能状況
TeamID = 0	ルームに入る前にTeamID!=0、ルームに入ったらTeamID=0に変更します	10メートル	話し声はルーム内のすべての人に聞こえます（音声モードがすべての人かチームのみかを問わない） TeamID=0のメンバーとコミュニケーションが取れます RANGE_AUDIO_MODE_WORLDに設定された同ルーム、10メートルの範囲内でメンバーの声が聞こえます

	ルームに入る前に TeamID=0であり、 TeamID=0でルームに入ります	10メートル	話し声はルーム内のすべての人に聞こえます（音声モードがすべての人かチームのみかを問わない） TeamID=0のメンバーとコミュニケーションが取れます ルーム内の他の人の声が聞こえません
--	---	--------	--

シナリオ例

PUBGゲーム：例えばPUBGタイプのゲームでは、4人でチームを組む場合、この4人に同じチーム番号TeamIDを設定する必要があります。100人ごとに1つの対局ルーム、1つの対局に25チームがある場合、25チームは同じ音声ルームに入る必要があります。対局中、あるプレイヤーが10メートル範囲内の見知らぬ人とコミュニケーションを取りたい場合、音声距離範囲を10に設定し、音声モードをRANGE_AUDIO_MODE_WORLDに設定し、またマイクとスピーカーをオンにします。チーム以外のメンバーでなく、チームメンバーとコミュニケーションを取りたい場合は、音声モードをRANGE_AUDIO_MODE_TEAMに設定するだけでよい。

ゲームコンサート：ゲーム中にコンサートを開催し、歌手とゲームプレイヤーが対話しない場合、ゲームプレイヤーがTeamID=OpenIDを使用してレンジボイスルームに入ることができます。また音声モードをRANGE_AUDIO_MODE_WORLDに設定し、ゲームのプレイ方法に応じて音声距離の範囲を設定します。これにより、ゲームプレイヤーは近くのプレイヤーとコミュニケーションを取ることができます。歌手がTeamIDを0に設定してルームに入ると、歌手の声はルームの人全体に聞こえるが、歌手には他の人の声は聞こえません。

ホストモード：ゲーム中の仮想テーブルゲームのようなシーンでは、ホストの話し声はルーム内のすべての人に聞こえると同時に、範囲内のプレイヤーの話し声も聞こえるようにしなければなりません。ホストはまずTeamID!=0の形でルームに入り、ルームに入ってからTeamIDを0にすると、その時点でホストの話はルーム内のすべての人に聞こえ、ホストも範囲内のプレイヤーの声を聞くことができます。

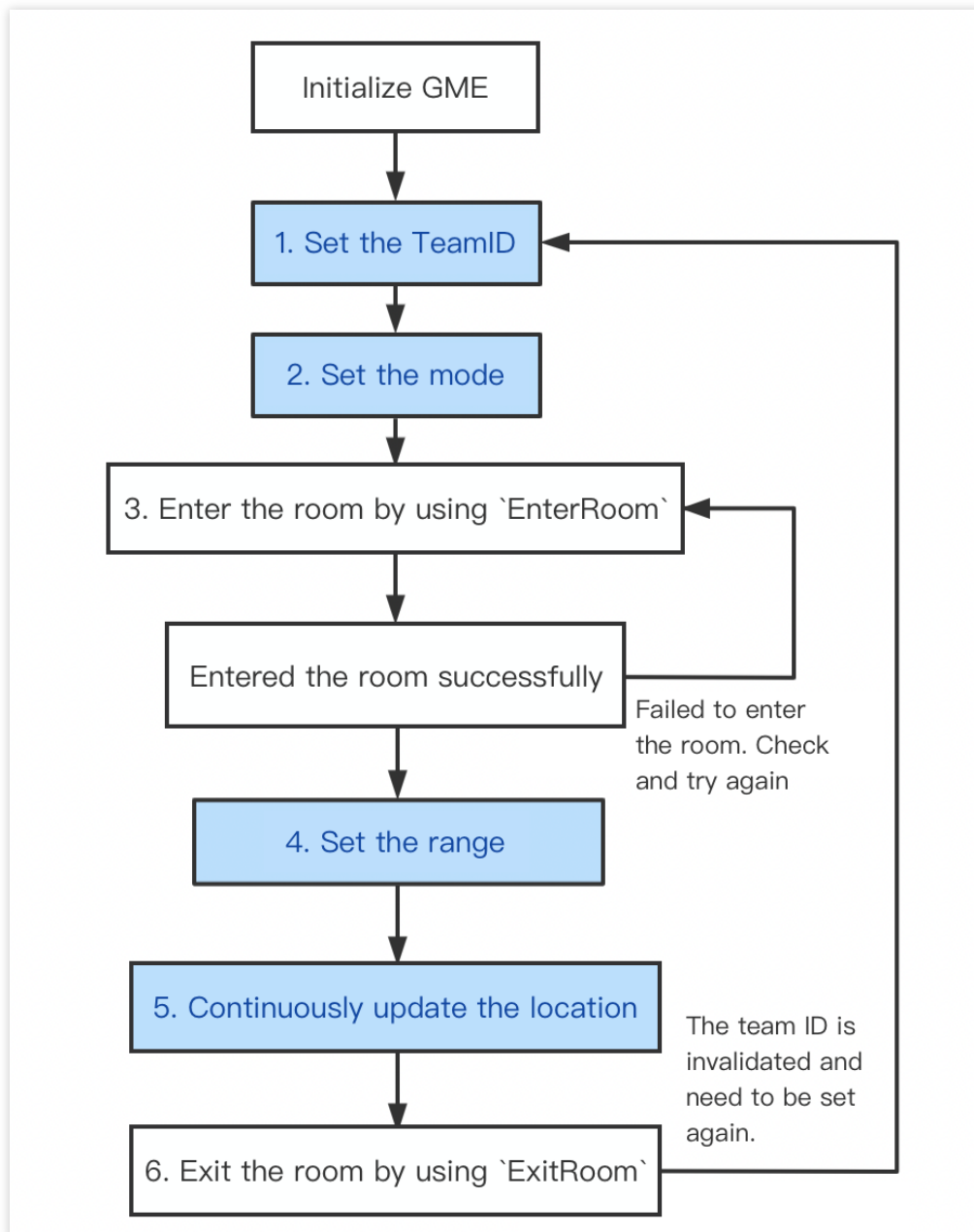
前提条件

リアルタイム音声サービスが有効になっていること：[音声サービス有効化ガイド](#)をご参照ください。

GME SDK導入済み：コアインターフェースとリアルタイム音声インターフェースの導入を含みます。詳細については、[Native SDKクイックスタート](#)、[Unity SDKクイックスタート](#)、[Unreal SDKクイックスタート](#)をご参照ください。

万人レンジボイス：同室でレンジボイスを使用する人数は1000人を超えた場合は、[チケットを提出](#)してGME開発者に連絡してください。

使用手順



ご注意：

この手順に従ってインターフェースを呼び出してください。

フローチャットの青い部分はレンジボイスに必要な手順です。

通常のチーム音声ルームとは異なり、レンジボイス機能を使用しているは、**スムーズな音質でルームに入る必要があります。**

入室が成功した後、UpdateAudioRecvRangeを1回以上呼び出し、フレームごとにUpdateSelfPositionを呼び出します。

1. TeamIDの設定

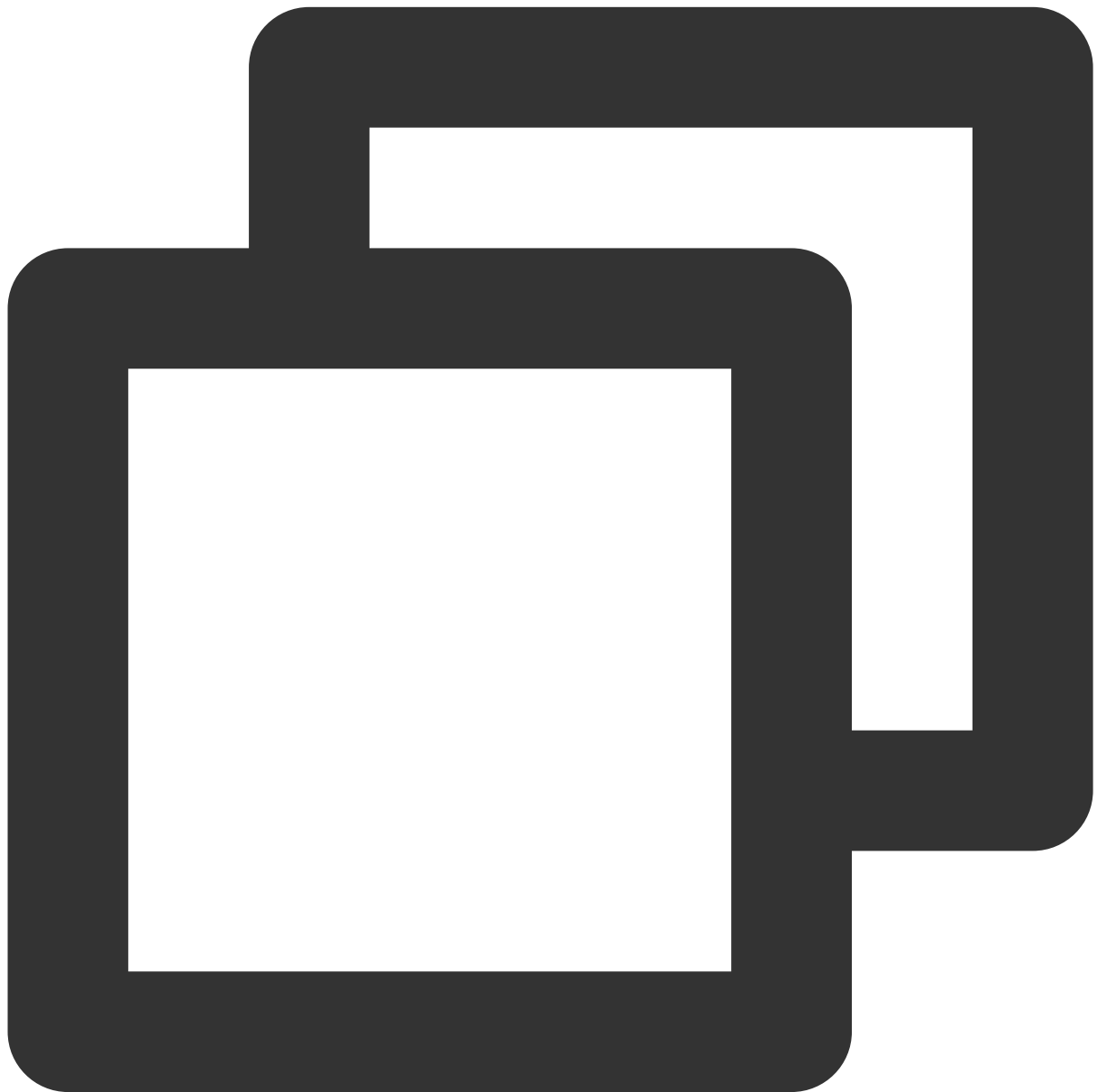
ルームに入る前に、このインターフェースでチーム番号を設定すると、次回の入室に有効です。

入室後、このインターフェースを使用してチーム番号を変更できます。設定後すぐに有効になります。

退室後、TeamIDが自動的に0にリセットされないため、当該ボイスモードを呼び出すことを決めたら、毎回EnterRoomの前にこのメソッドを呼び出しTeamIDを設定してください。

退室してからもう一度入室した場合、退室成功のコールバックを実行した後、チーム番号設定インターフェースを呼び出してください。

関数のプロトタイプ



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

パラメー	タイプ	意味
------	-----	----

タ		
teamID	int	チームナンバーであり、レンジボイスモードの利用中に使われています。 TeamIDが0の場合は、通話モードは一般チームボイスで、デフォルトは0です。

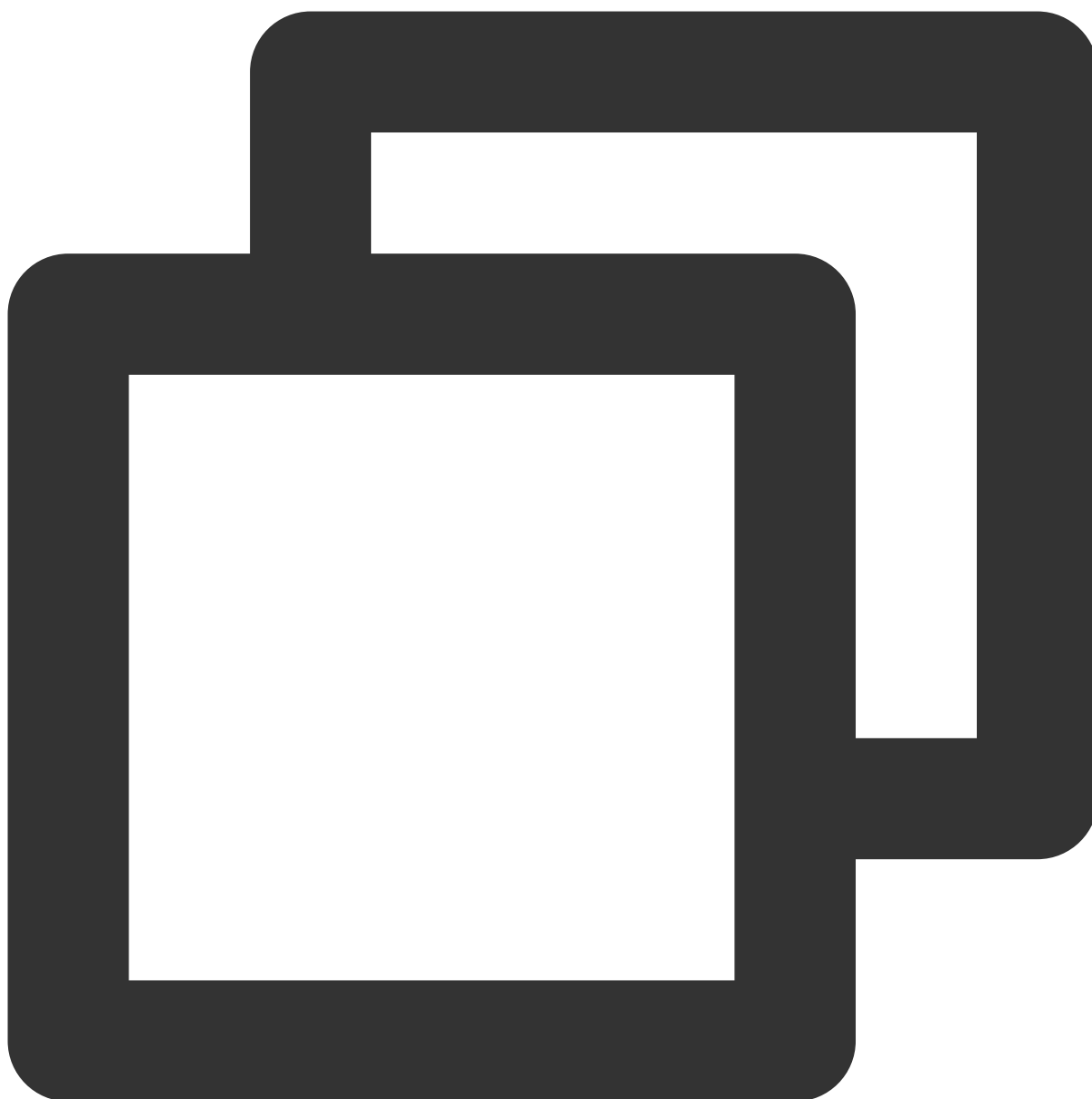
2. 音声モードの設定

入室前に、このインターフェースを呼び出して音声モードを変更します。次回の入室に有効です。

入室後、このインターフェースを呼び出して音声モードを変更すると、現在のユーザーの音声モードが直接変更されます。

退室後、このパラメータは自動的にMODE_WORLDにリセットされないため、このメソッドを呼び出すことを決めたら、毎回はEnterRoomの前にこのメソッドを呼び出し、audioModeを設定してください。

関数のプロトタイプ



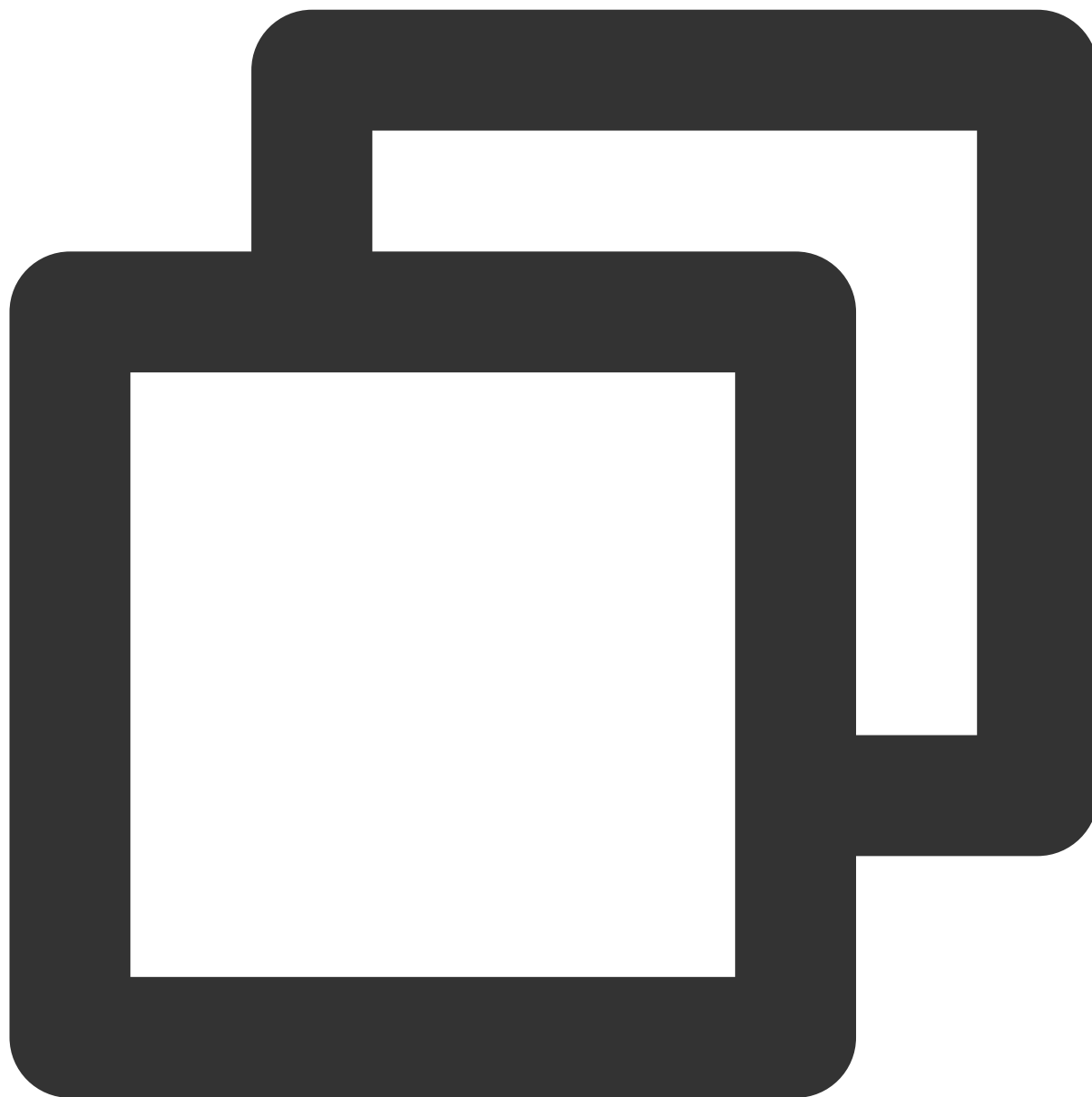
```
ITMGRoom int SetRangeAudioMode (RANGE_AUDIO_MODE rangeAudioMode)
```

パラメータ	タイプ	意味
rangeAudioMode	int	0(MODE_WORLD)が「全ての人」を表し、1(MODE_TEAM)が「チームのみ」を表します

3. 音声ルームに参加します

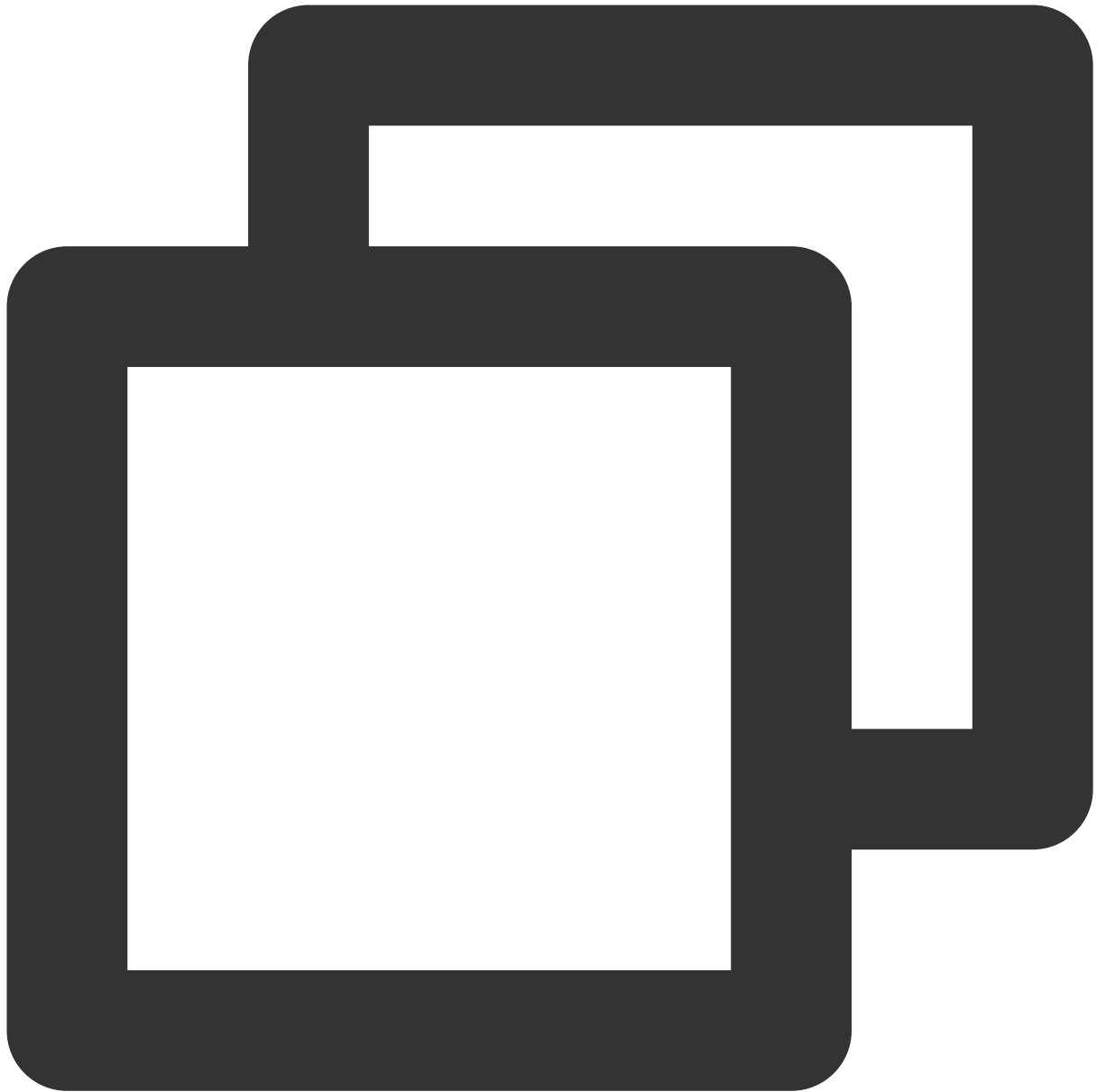
EnterRoomを呼び出す前に、SetRangeAudioTeamIDとSetRangeAudioModeの2つのAPIを呼び出す必要があります。

関数のプロトタイプ



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

音声ルームに入るにはスムーズな音質を使用しなければなりません。その後、入室のコールバックを監視して処理します。



```
public void OnEvent(ITMGContext.ITMG_MAIN_EVENT_TYPE type, Intent data) {
    if (ITMGContext.ITMG_MAIN_EVENT_TYPE.ITMG_MAIN_EVENT_TYPE_ENTER_ROOM == type)
    {
        //イベントから返されたデータの分析
        int nErrCode = data.getIntExtra("result" , -1);
        String strErrMsg = data.getStringExtra("error_info");

        if (nErrCode == AVErrors.AV_OK)
        {
            ///入室シグナリングを受信し、入室が成功し、機器を操作できます
            ScrollView_ShowLog("EnterRoom success");
        }
    }
}
```

```
        Log.i(TAG, "EnterRoom success!");
    }
    else
    {
        //入室に失敗し、返されたエラーメッセージを分析する必要があります
        ScrollView_ShowLog("EnterRoom fail :" + strErrMsg);
        Log.i(TAG, "EnterRoom fail!");
    }
}
}
```

入室が成功すると、UpdateAudioRecvRangeを少なくとも1回呼び出し、フレームごとにUpdateSelfPositionを呼び出します。

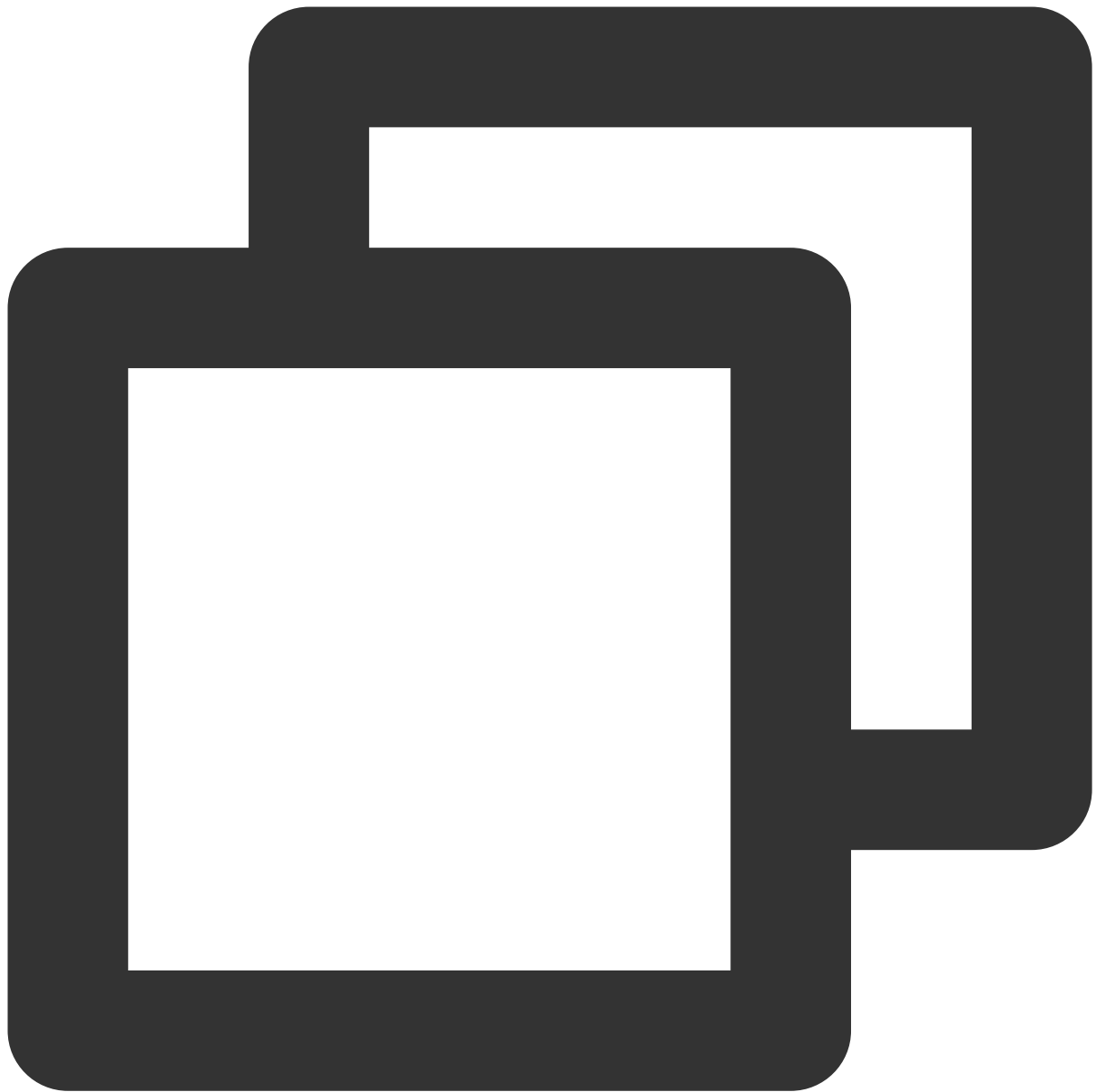
4. ボイス受信距離範囲の設定

このメソッドで設定された音声受信範囲（距離はゲームエンジンによって異なる）は、**正常に入室した後に呼び出すことが可能です。**

このメソッドはUpdateSelfPositionと連携し音源位置を更新することに使われています

このメソッドは一度呼び出すだけで有効になり、変更が可能です。

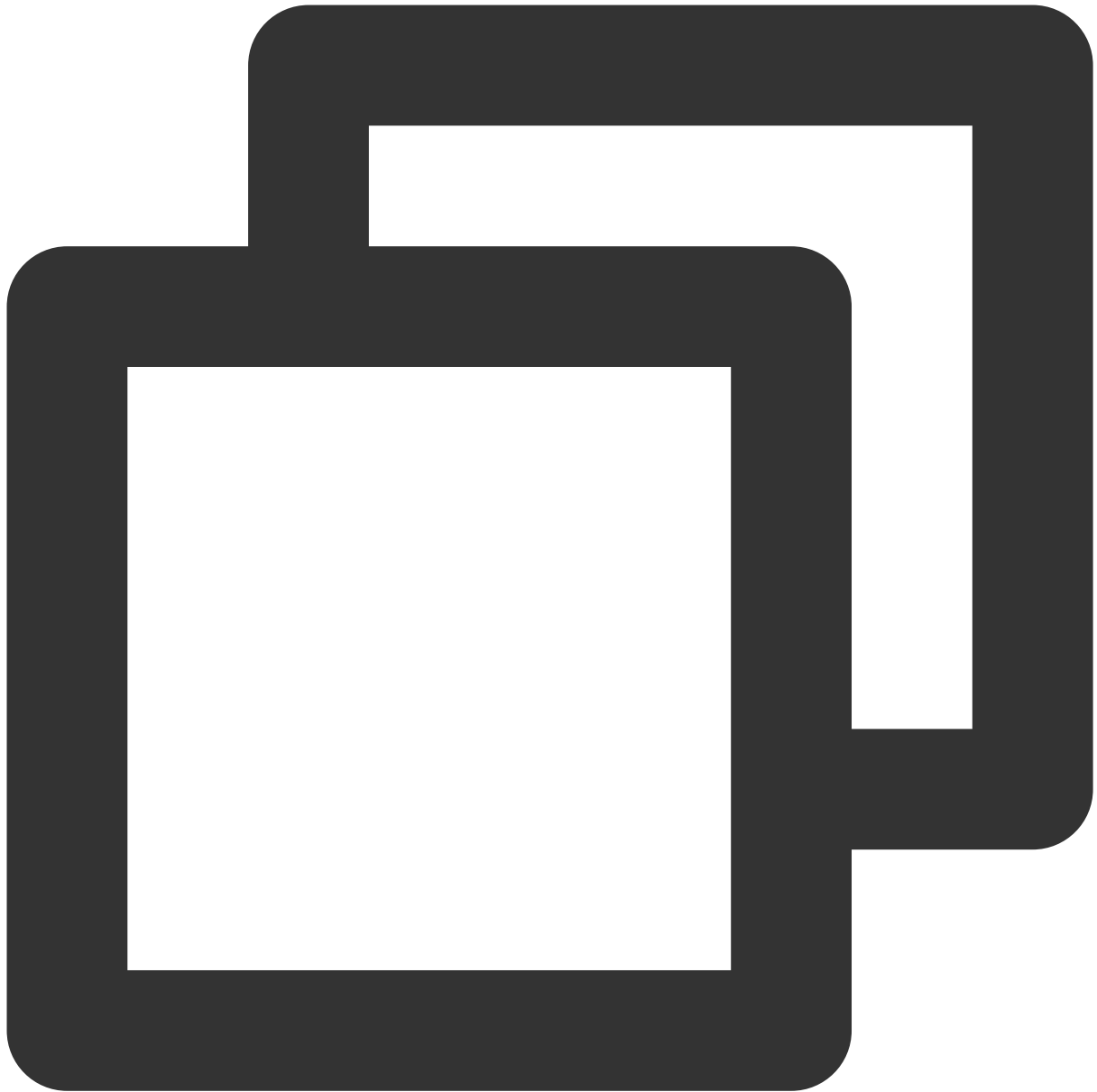
関数のプロトタイプ



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

パラメータ	タイプ	意味
range	int	音声を受信できる最大範囲（エンジン距離単位）

サンプルコード



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

5. 音源方位の更新

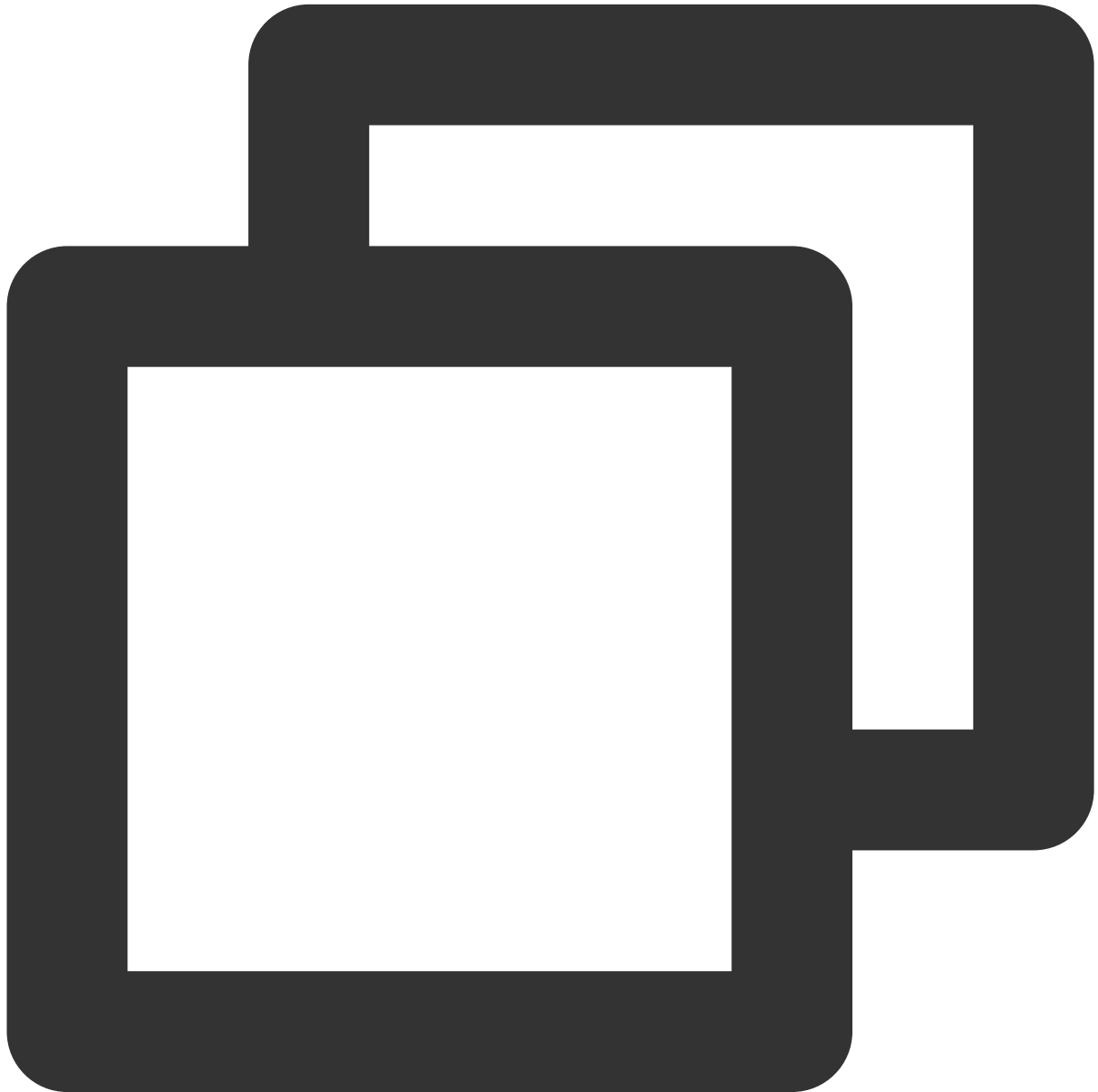
音源方位の更新はサーバーに自端の位置を通知することを目的とします。**自端の世界座標+自端が受信した音声の範囲**と、**他端の世界座標+他端が受信した音声の範囲**で判断し、レンジボイス効果を達成します。

この関数は音源方位の更新に使用されます。**入室が成功した後に呼び出すことができます。またフレームごとに呼び出す必要があります。**Unityエンジンの場合、このインターフェースはUpdateで呼び出される必要があります。

レンジボイスを使用して音源方位を更新する必要があります。範囲判断能力が不要な場合でも、**ルームに入ってから一度このインターフェースを呼び出す必要があります。**

3Dサウンド効果を同時に使用する場合、このインターフェースのパラメータaxisForward、axisRightおよびaxisUpは下文の**3Dボイスセクション**に従って設定する必要があります。

関数のプロトタイプ



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

パラメータ	タイプ	意味
-------	-----	----

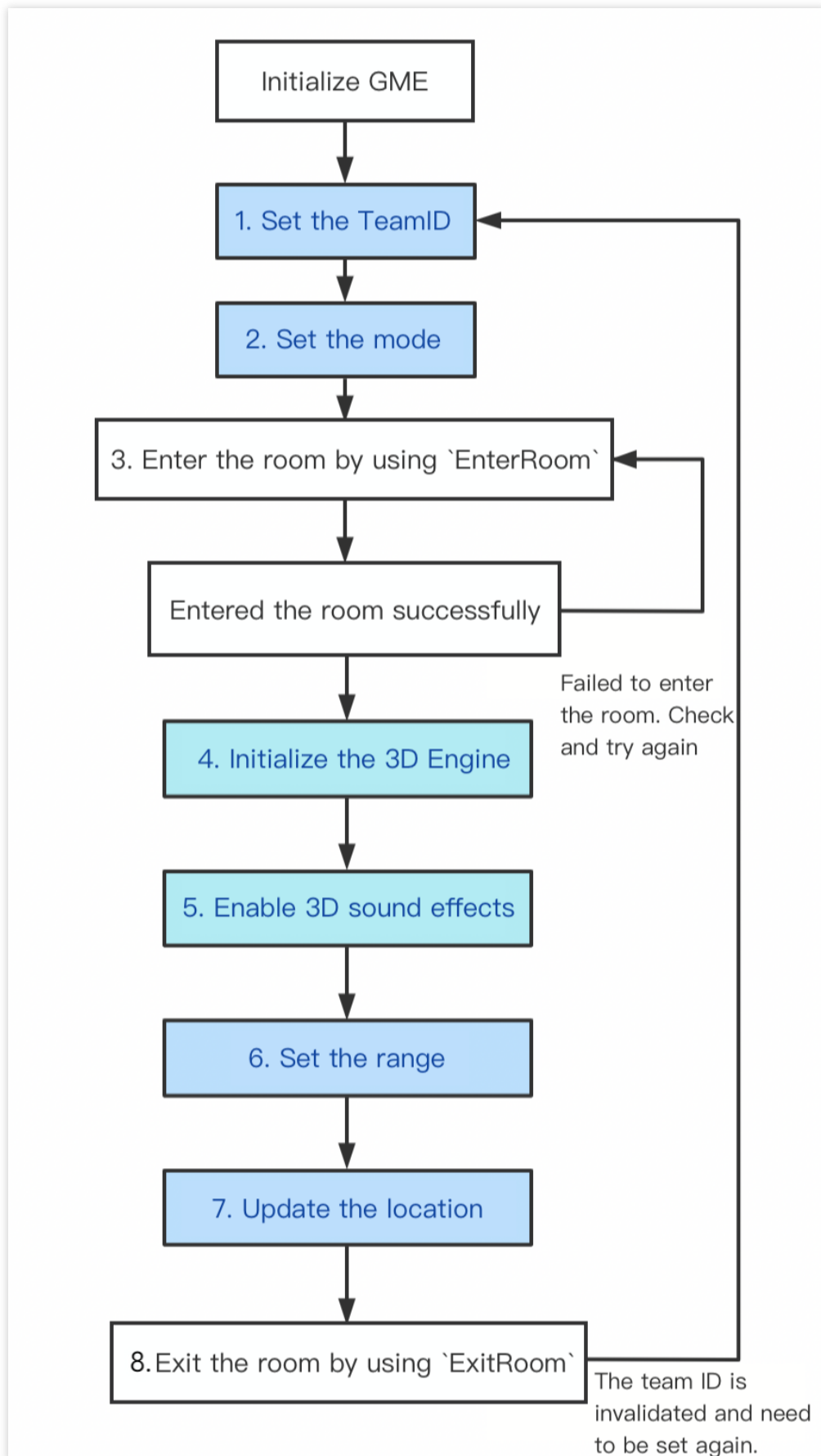
position	int[]	ワールド座標系中の座標で、前、右、上の順序で表示されます
axisForward	float[]	この製品では無視してください
axisRight	float[]	この製品では無視してください
axisUp	float[]	この製品では無視してください

レンジボイスと3D効果音

上文で紹介したレンジボイス機能は、距離によって音声の到達性を制御するもので、より没入感のある体験が必要な場合は、3D効果音と併用することをお勧めします。

使用手順

レンジボイスを使用すると同時に、3D効果音機能を使用する場合は、ルームに入ってから次のレンジボイスステップ1、2、3を完了した後、3Dエンジンを初期化し、3D効果音をオンにします。



説明：

フローチャートの緑の部分は、3D音声に必要な手順です。

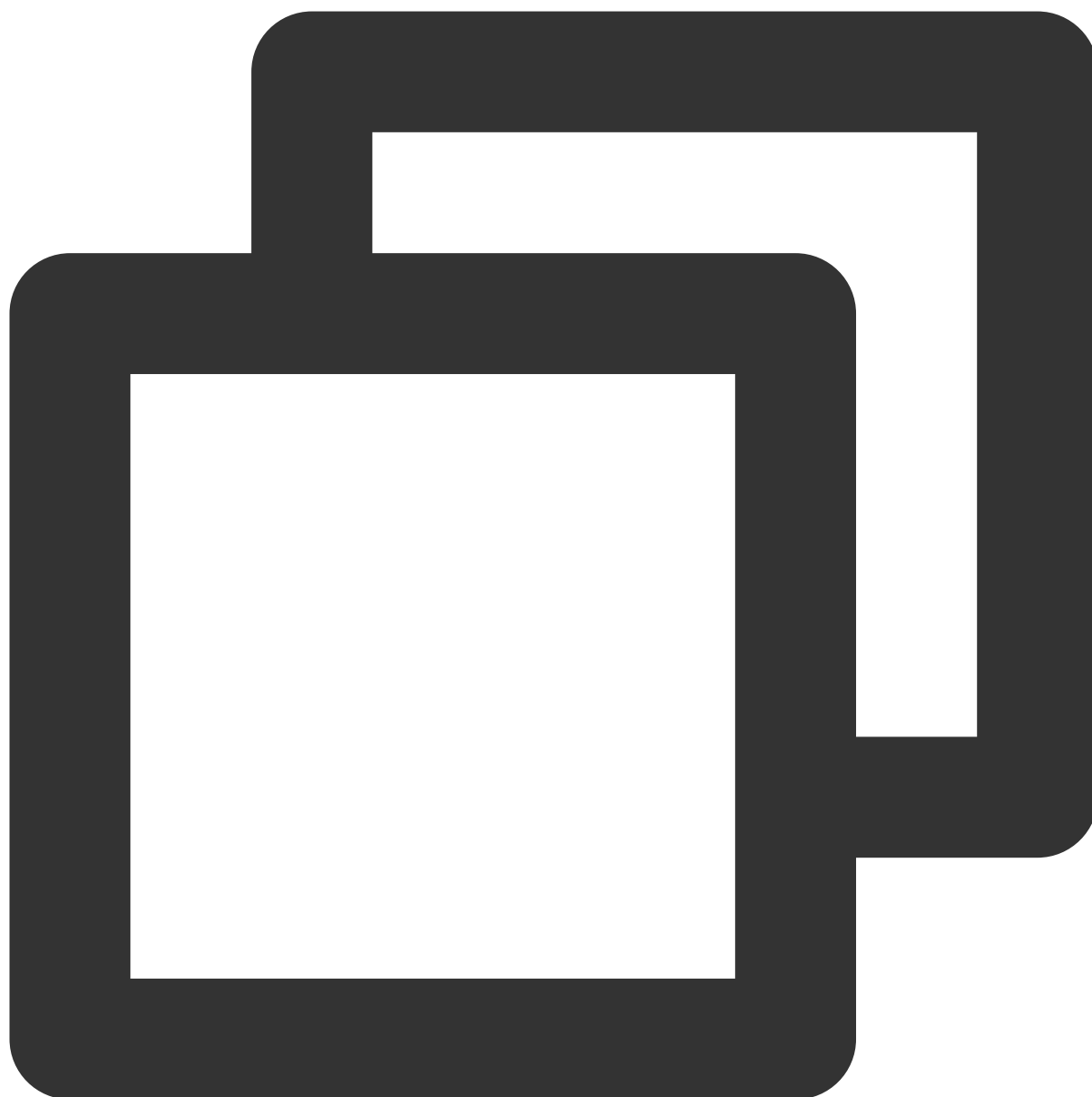
前提条件

[レンジボイスの利用手順](#)を参照して手順1、2、3を完了してください。

4. 3D効果音エンジンの初期化

この関数は、3Dサウンドエフェクトエンジンを初期化するために使用され、入室した後に呼び出します。このインターフェイスは、3Dサウンドエフェクトを使用する前に呼び出す必要があります、3Dサウンドエフェクトを送信せず受信するのみのユーザーでも、このインターフェイスを呼び出す必要があります。

関数のプロトタイプ



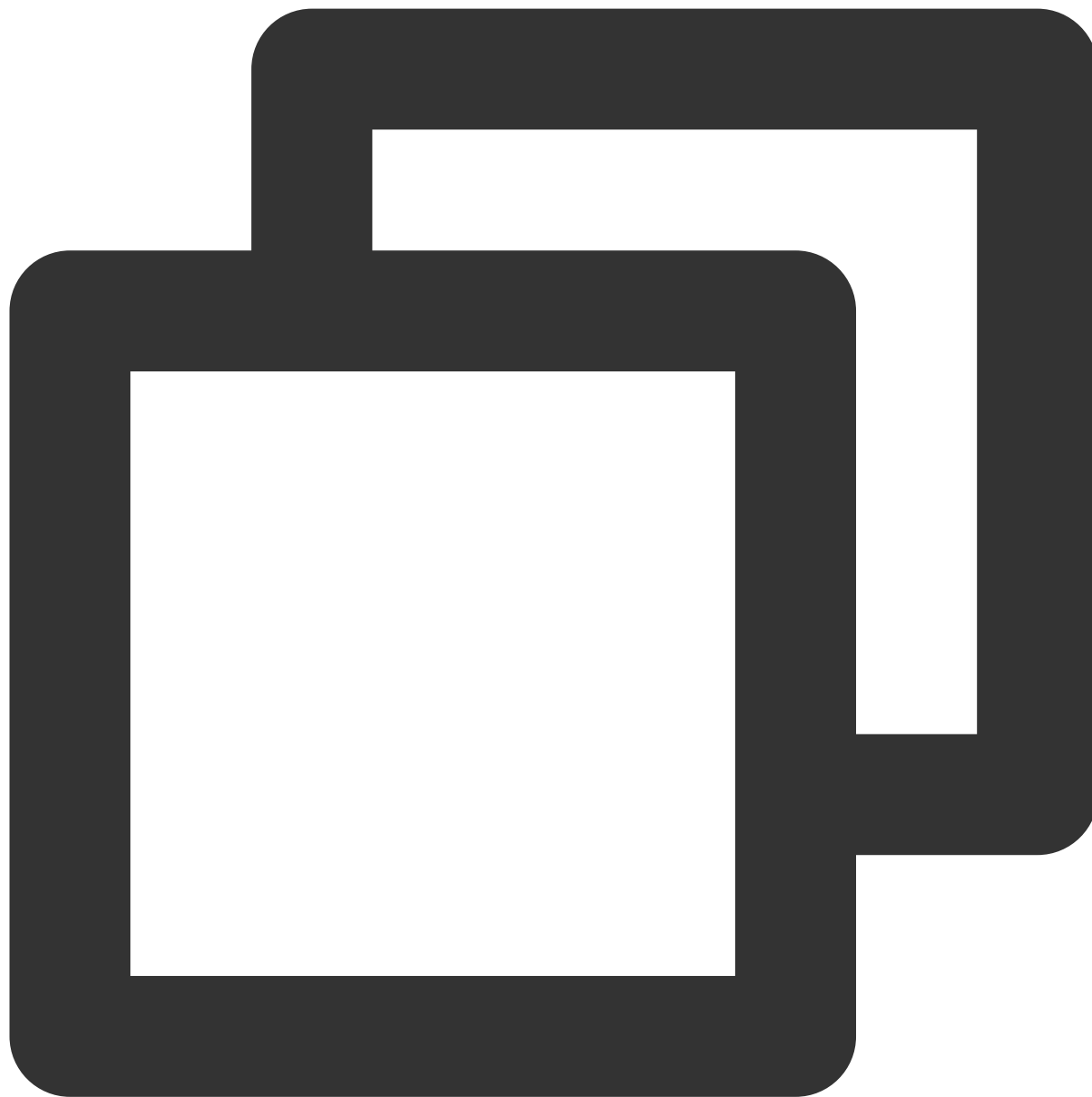
```
public abstract int InitSpatializer(string modelPath)
```

パラメータ	タイプ	意味
modelPath	string	空欄を埋めてください

5. 3D効果音のオン/オフ

この関数は、3Dサウンドエフェクトをオン/オフにするために使用されます。オンにした後、3Dサウンドエフェクトが聞こえます。

関数のプロトタイプ



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

パラメータ	タイプ	意味
enable	bool	オンにすると3D効果音が聞こえます
applyToTeam	bool	3D音声はチーム内で機能するかどうかを示します。enableがtrueである場合にのみ有効です。

IsEnableSpatializer インターフェースを使用して3Dサウンドの状態を取得します。

6. ボイス受信距離範囲（3D）の設定

このメソッドで設定された音声受信範囲（距離はゲームエンジンによって異なる）は、**正常に入室した後に呼び出すことが可能です。**

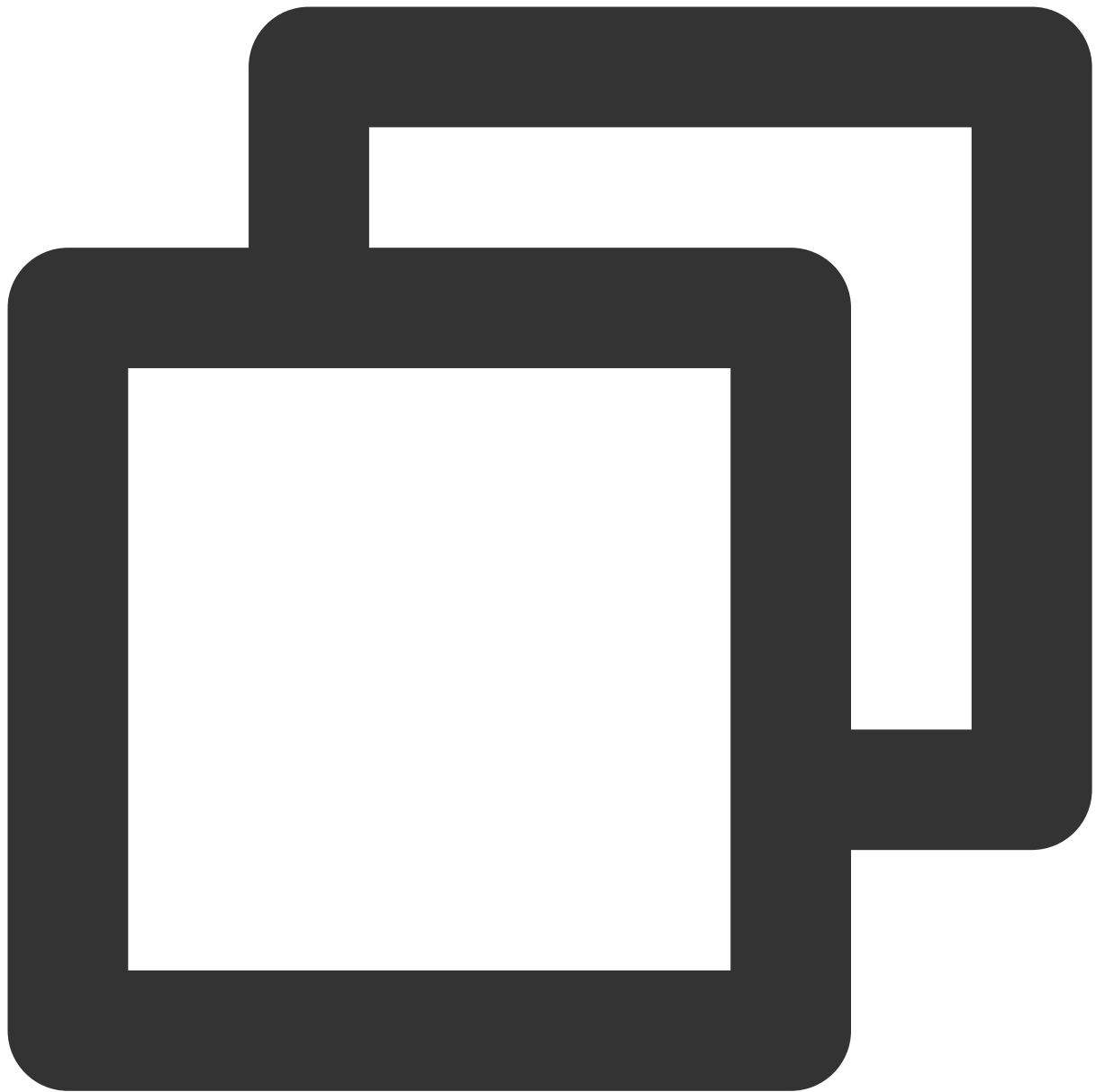
このメソッドはUpdateSelfPositionと連携し音源位置を更新することに使われています

このメソッドは一度呼び出すだけで有効になります。

3Dサウンドエフェクトでは、音源のボリュームは音源の距離と減衰関係にあります。単位距離がrangeを超えた後、ボリュームはほぼゼロまで減衰します。

距離と音声の減衰の関係については、ドキュメント付属書をご参照ください。

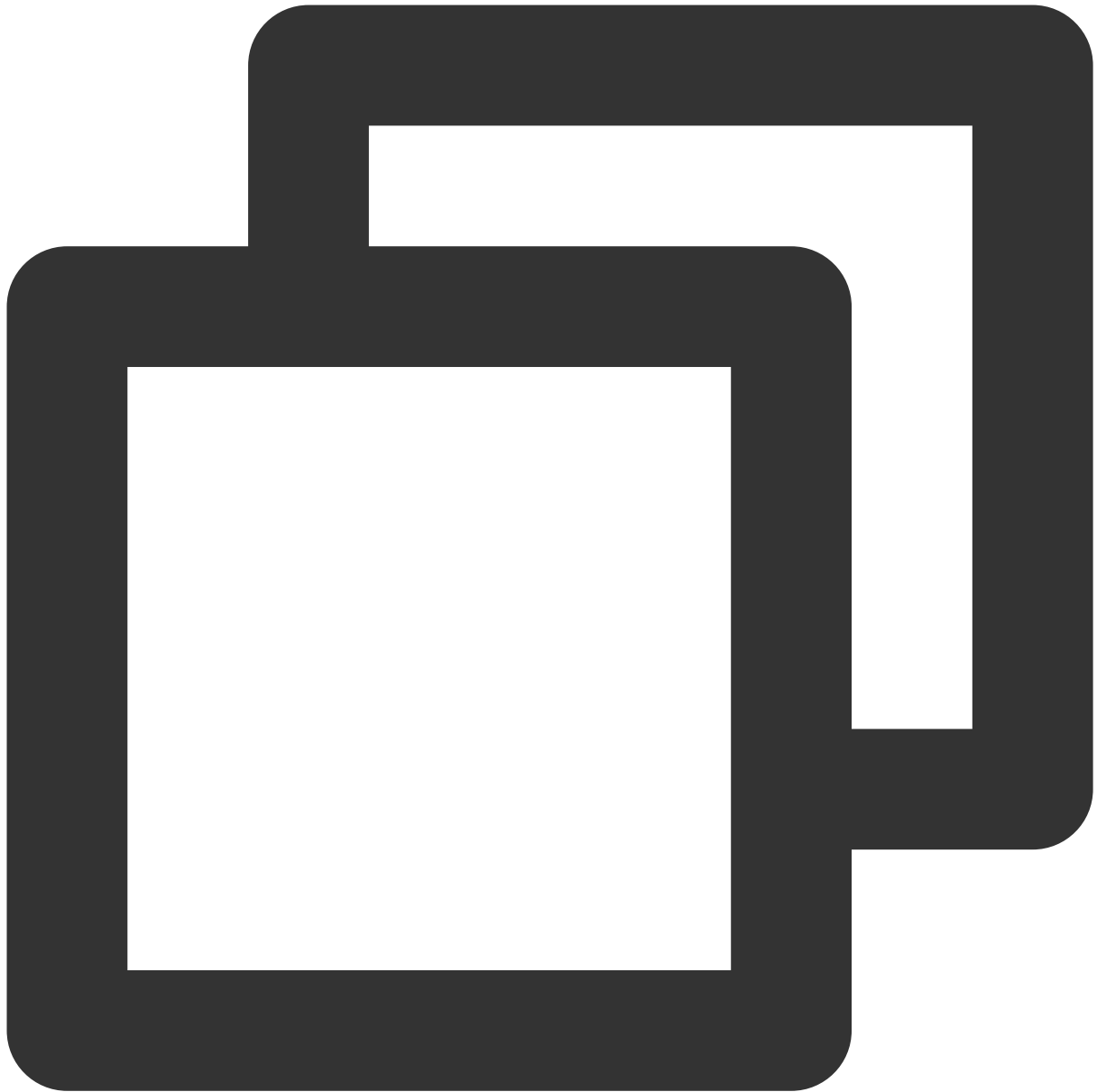
関数のプロトタイプ



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

パラメータ	タイプ	意味
range	int	音声を受信できる最大範囲（エンジン距離単位）

サンプルコード



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

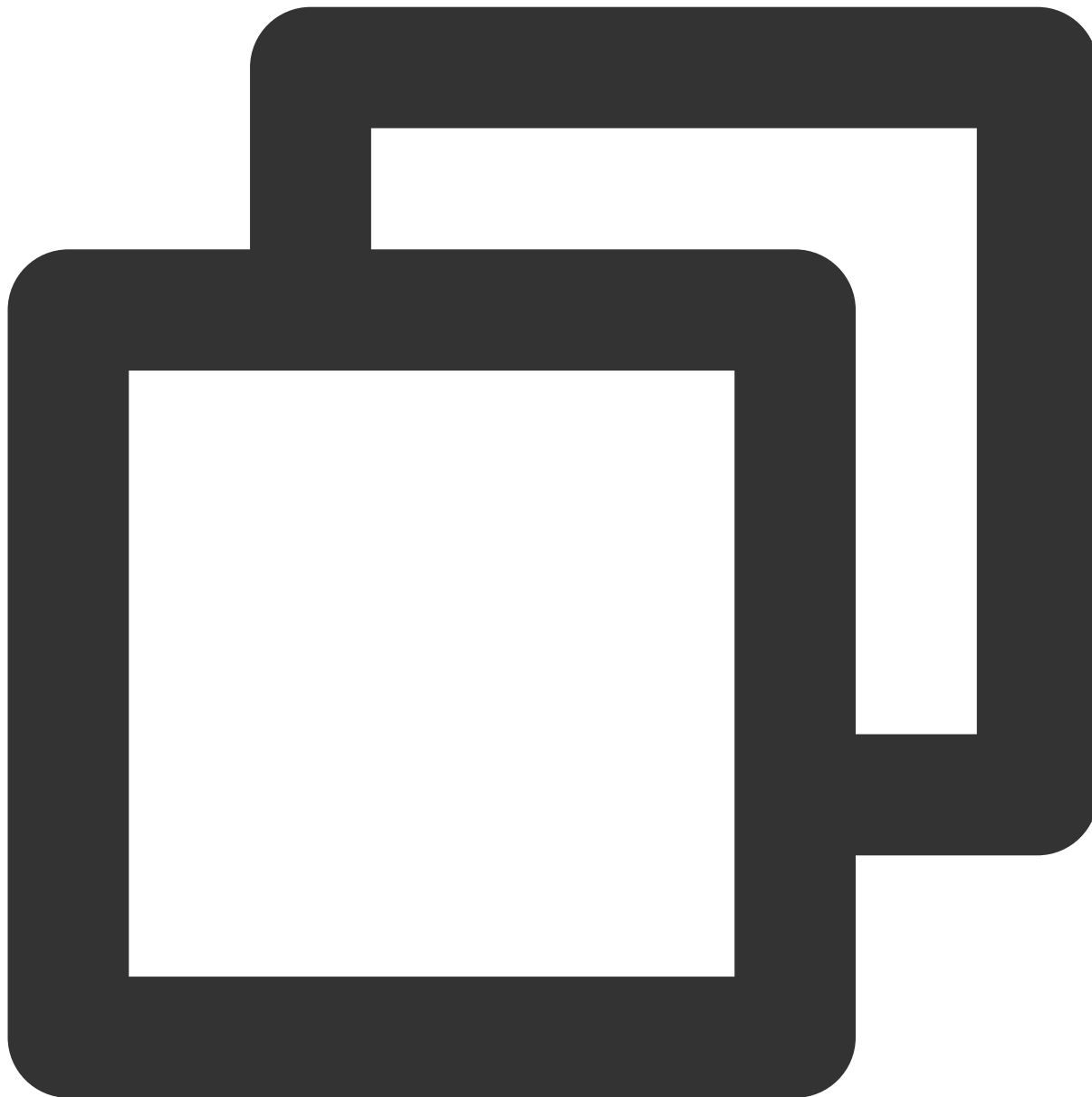
7. 音源方位（3D）の更新

音源方位の更新はサーバーに自端の位置を通知することを目的とします。**自端の世界座標+自端が受信した音声の範囲**と、**他端の世界座標+他端が受信した音声の範囲**で判断し、レンジボイス効果を達成します。

この関数は音源方位の更新に使用されます。**入室が成功した後に呼び出すことができます。またフレームごと**に**呼び出す**必要があります。Unityエンジンの場合、このインターフェースはUpdateで呼び出される必要があります。

この機能を使用するには、サンプルコードを直接コピーして呼び出してください。

関数のプロトタイプ



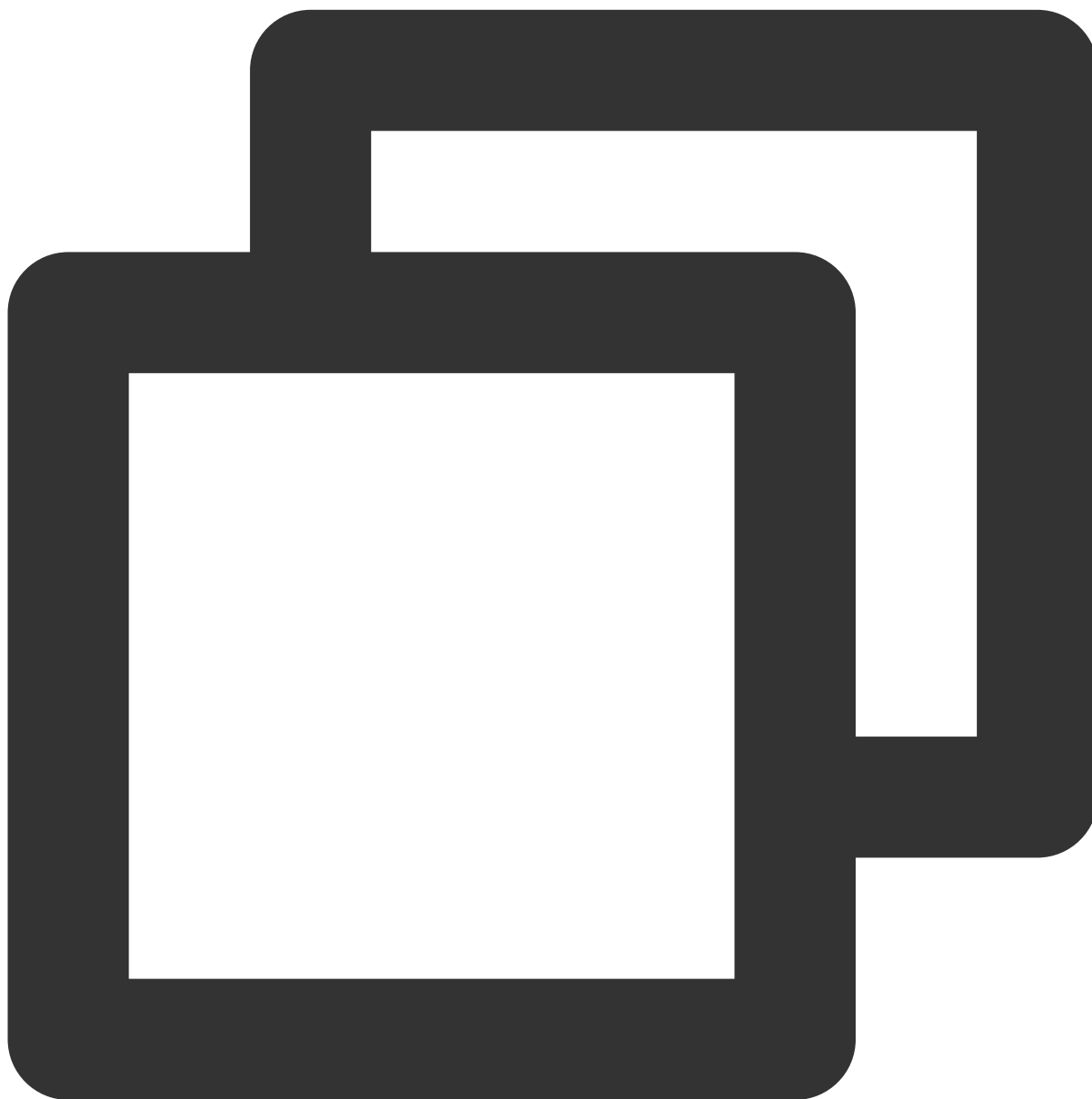
```
public abstract int UpdateSelfPosition(int position[3], float axisForward[3], float
```

パラメータ	タイプ	意味
position	int[]	ワールド座標系中の座標で、前、右、上の順序で表示されます
axisForward	float[]	ローカル座標系前軸の単位ベクトル

axisRight	float[]	ローカル座標系右軸の単位ベクトル
axisUp	float[]	ローカル座標系上軸の単位ベクトル

サンプルコード

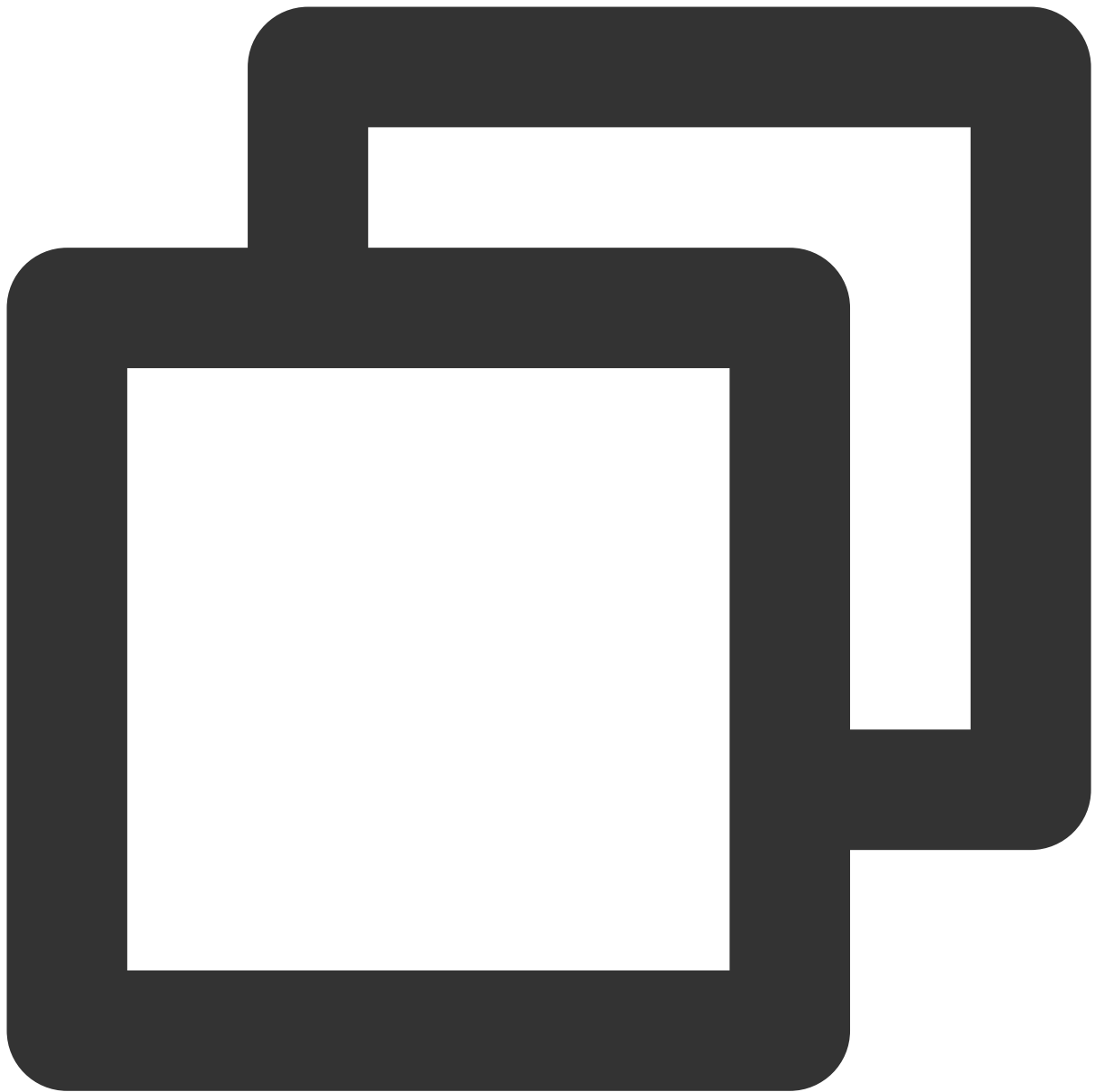
Unreal



```
FVector cameraLocation = UGameplayStatics::GetPlayerCameraManager(GetWorld(), 0)->G
FRotator cameraRotation = UGameplayStatics::GetPlayerCameraManager(GetWorld(), 0)->
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, u
```

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] {
```

```

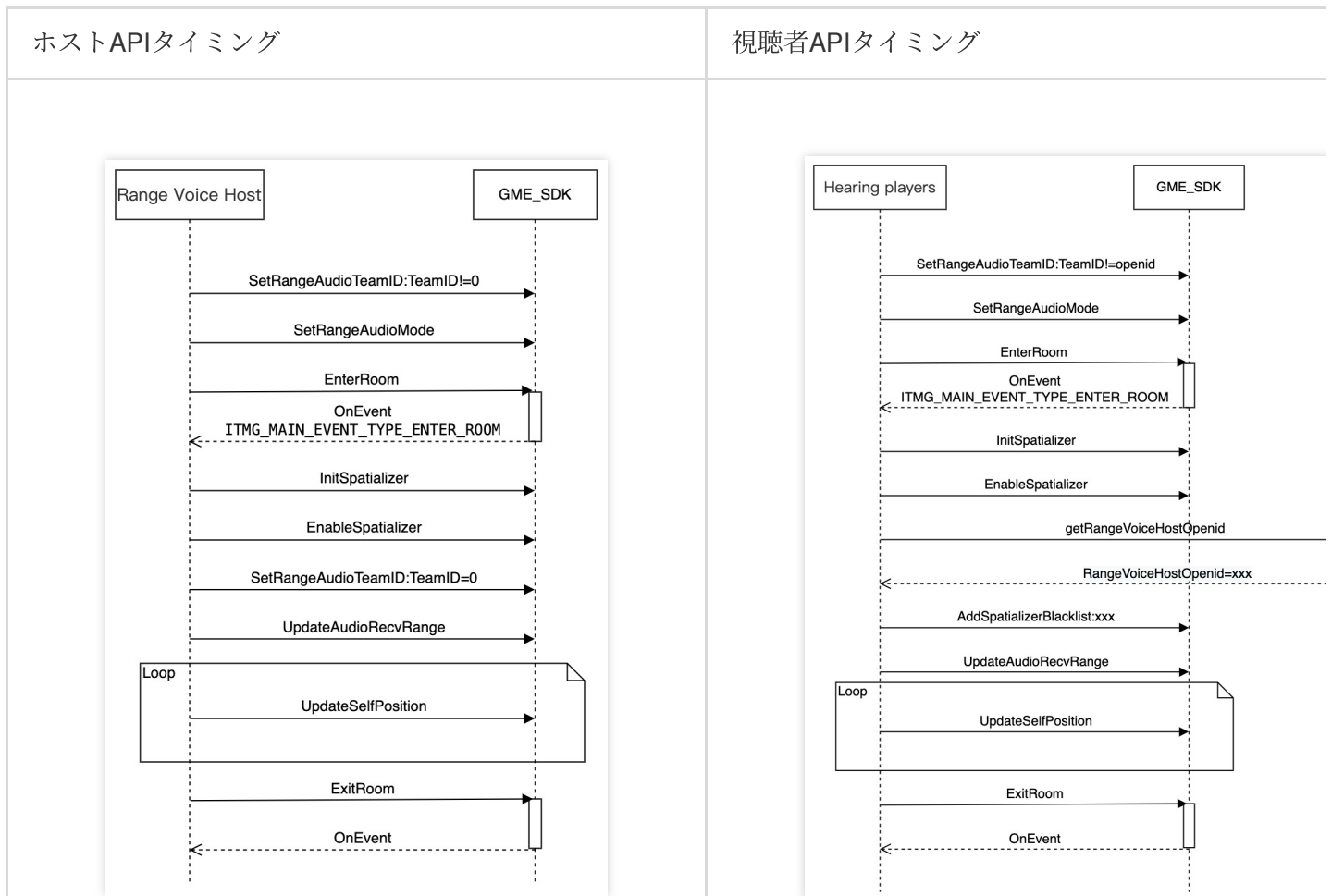
matrix.m20,
matrix.m00,
matrix.m10 };
float[] axisUp = new float[3] {
matrix.m21,
matrix.m01,
matrix.m11 };
ITMGContext.GetInstance().GetRoom().UpdateSelfPosition(position, axisForward, axisR

```

ホストに3D効果音を使用することを禁止する

もしシナリオの中でプレイヤーがレンジボイスのホストモードを使用する場合、つまり彼の声はルーム内のすべてのプレイヤーが聞くようにする場合、[3D音声ブラックリストインターフェース](#)を参照し、すべての聴取側でホストを聴取側の3D音声ブラックリストに追加する必要があります。これで、3D音声機能減衰効果がホストの音声到達性の影響を回避します。

各ロールのAPI呼び出しのタイミングは次のとおりです：



付録

各音声モード

各音声モードでのプレイヤーの音声到達状況：

仮にAプレイヤーの状態を「すべての人」とすると、対応するBプレイヤーの異なる音声モードでの到達可能状況は下記の通りです：

同じチームかどうか	範囲内かどうか	ボイスモード	AとBがお互いの声を聞いているかどうか
同じチーム	はい	MODE_WORLD	はい
		MODE_TEAM	はい
	いいえ	MODE_WORLD	はい
		MODE_TEAM	はい
異なるチーム	はい	MODE_WORLD	はい
		MODE_TEAM	いいえ
	いいえ	MODE_WORLD	いいえ
		MODE_TEAM	いいえ

仮にAプレイヤーの状態を「チームのみ」とすると、対応するBプレイヤーの異なる音声モードでの到達可能状況は下記の通りです：

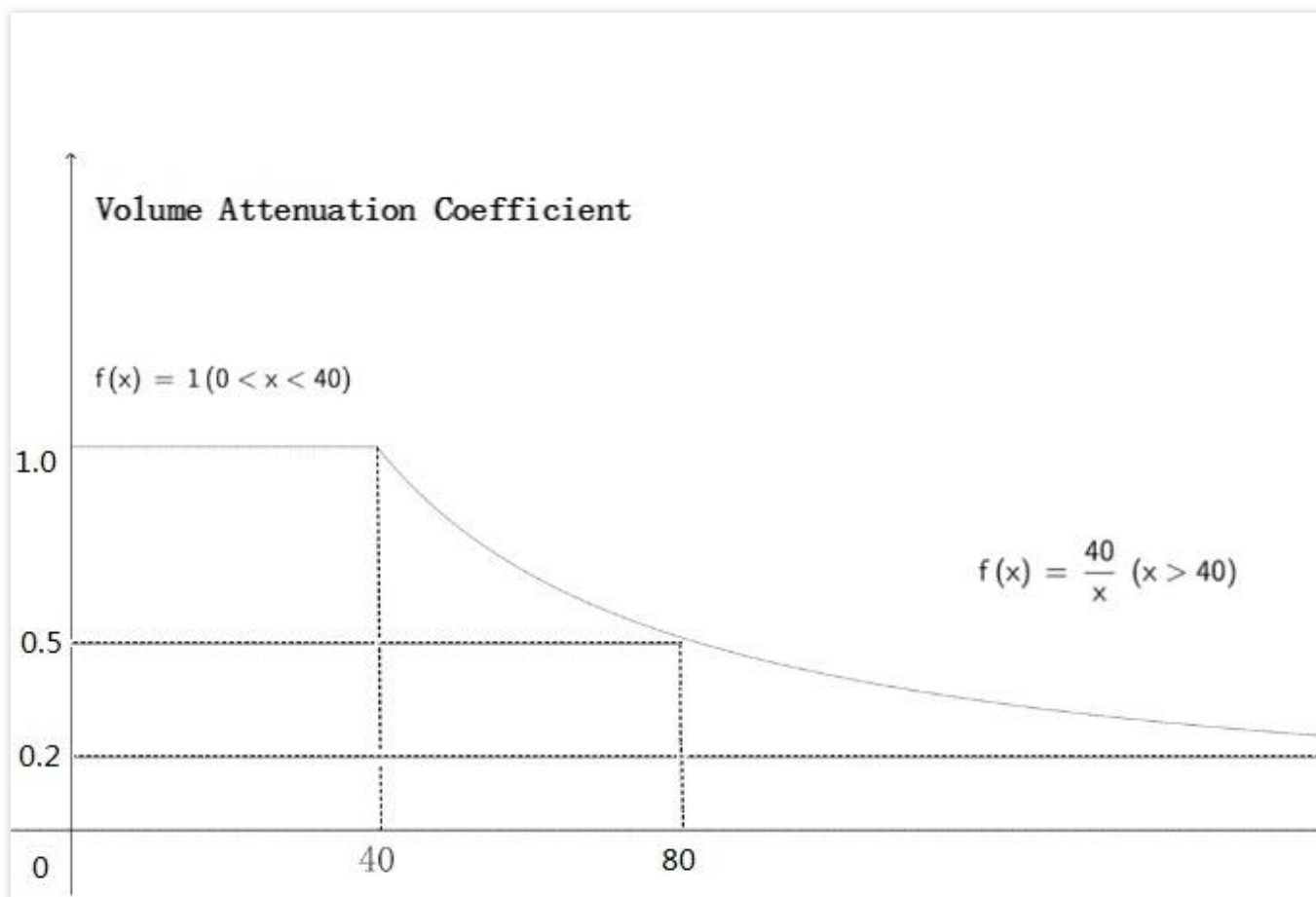
同じチームかどうか	範囲内かどうか	音声状態	AとBがお互いの声を聞いているかどうか
同じチーム	はい	MODE_WORLD	はい
		MODE_TEAM	はい
	いいえ	MODE_WORLD	はい
		MODE_TEAM	はい
異なるチーム	はい	MODE_WORLD	いいえ
		MODE_TEAM	いいえ
	いいえ	MODE_WORLD	いいえ
		MODE_TEAM	いいえ

距離と音の減衰の関係

3Dサウンドエフェクトでは、音源のボリュームは音源の距離と減衰関係にあります。単位距離がrangeを超えた後、ボリュームはほぼゼロまで減衰します。

--	--

距離範囲（エンジン単位）	減衰公式
$0 < N < \text{range}/10$	減衰係数：1.0（音量が減衰しない）
$N \geq \text{range}/10$	減衰係数： $\text{range}/10/N$



3Dサウンド

最終更新日：2024-01-18 15:47:47

開発者がTencent Cloud GME製品のAPIを容易にデバッグして導入するために、このドキュメントではGME 3D効果音の導入技術を紹介します。

シナリオ

通常の入室後のリアルタイム音声では、プレイヤーの音声に3D効果音の効果がなく、プレイヤーの間ではとても簡単なコラボライブを行うことしかできません。一方、3D位置音声を導入した後は、プレイヤーが呼びかけると自分の方位と位置情報を開示し、プレイヤーの音声も位置の変化によってリアルタイムで変化するようにになっています。3D効果音は、「大逃殺」のようなプレイヤー間のコミュニケーションや戦闘体験をよりリアルにし、PUBGのように、より没入的で臨場感のある遊び方を体感できるようになったと言えます。

[demoのダウンロード](#)をクリックして、3D効果音を体験できます。

前提条件

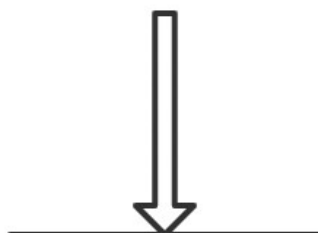
リアルタイム音声サービスが有効になっていること：[音声サービス有効化ガイド](#)をご参照ください。

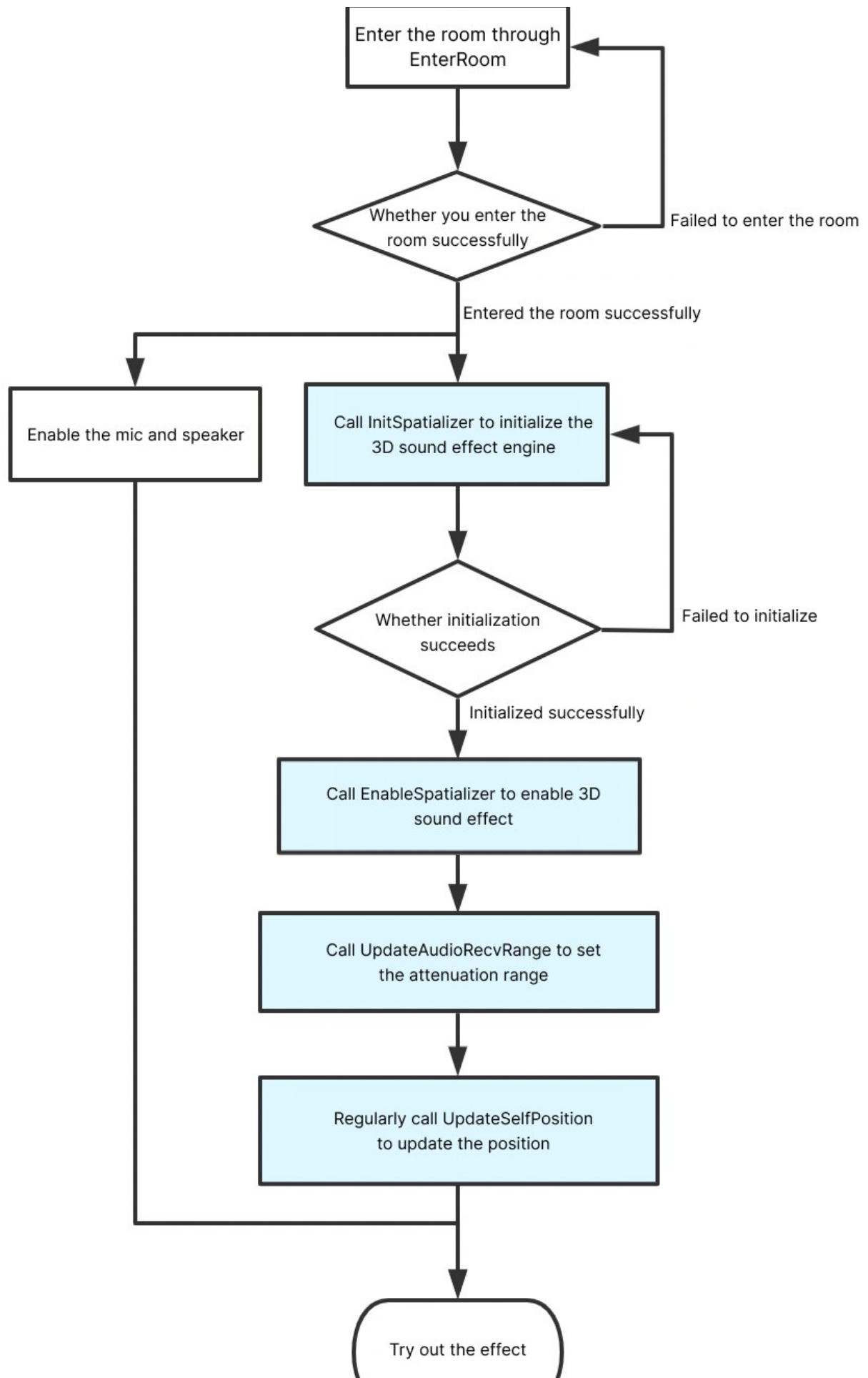
GME SDK導入済み：コアインターフェースとリアルタイム音声インターフェースの導入を含みます。詳細については、[Native SDKクイックスタート](#)、[Unity SDKクイックスタート](#)、[Unreal SDKクイックスタート](#)をご参照ください。

実装プロセス

実装フローチャート

次の図は3D効果音を実装するフローチャートです。青い部分は通常の入室リアルタイム音声と比較した導入ステップです。

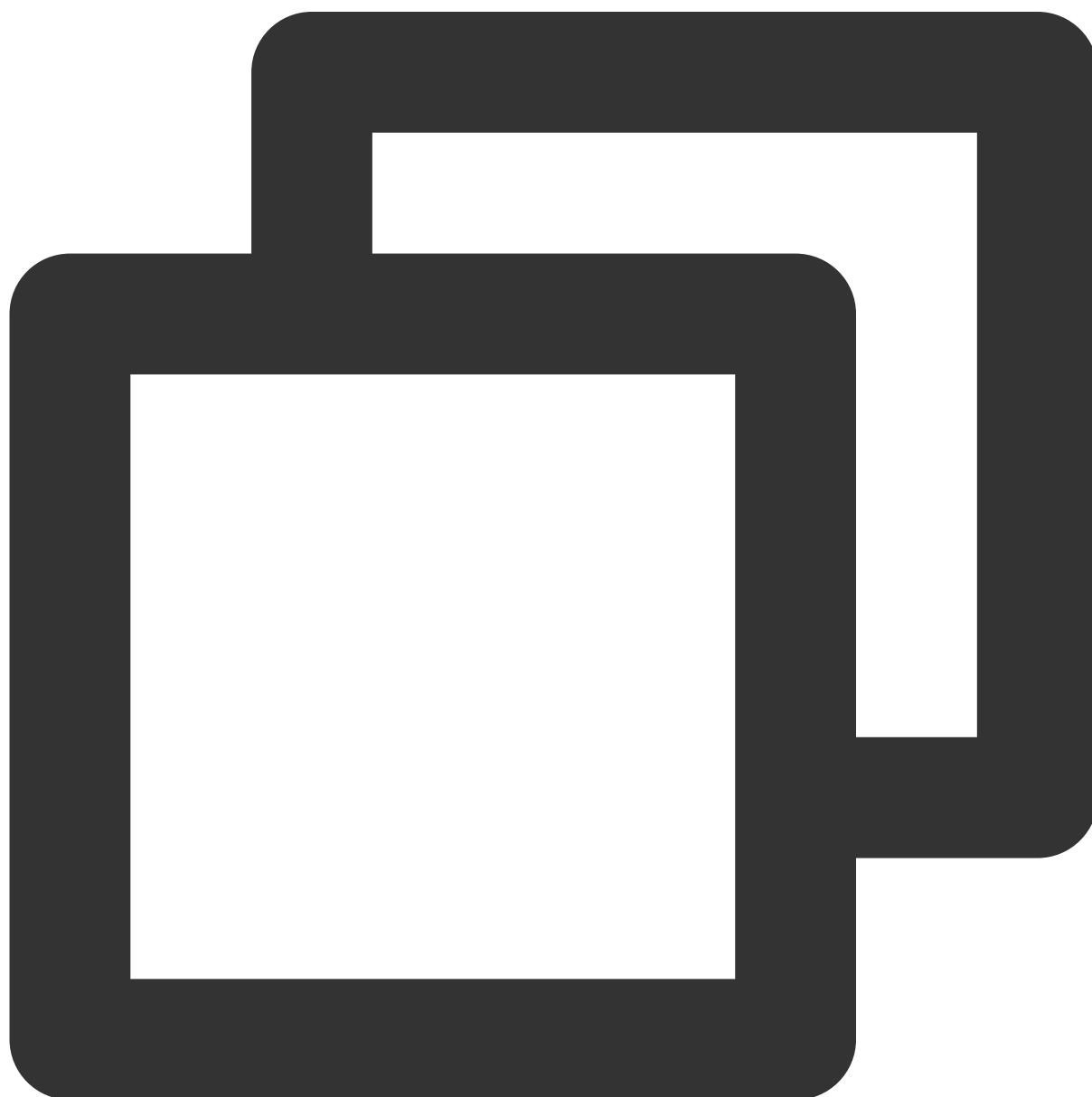




3D効果音エンジンの初期化

この関数は、3Dサウンドエフェクトエンジンを初期化するために使用され、入室した後に呼び出します。このインターフェイスは、3Dサウンドエフェクトを使用する前に呼び出す必要があります、3Dサウンドエフェクトを送信せず受信するのみのユーザーでも、このインターフェイスを呼び出す必要があります。

関数のプロトタイプ



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

パラメータ	タイプ	意味
modelPath	string	3D効果音のリソースファイルの絶対パス

パラメーター内の3D効果音リソースファイルは、別途ローカルにダウンロードしてください。導入するSDKのバージョンによって区別されます。

v2.8以降のバージョンの場合は、[ダウンロード](#)をクリックしてください、。md5:

d0b76aa64c46598788c2f35f5a8a8694。

v2.8～v2.9.5の場合は、[ダウンロード](#)をクリックしてください、。md5: 3d4d04b3949e267e34ca809e8a0b9243。

v2.9.6以降のバージョンの場合は、3D効果音のリソースファイルが組み込まれているので、ここのmodelPathを空にすることができます。

SDKのバージョンリリース履歴については[製品の最新情報](#)をご参照ください。

リソースパスについて

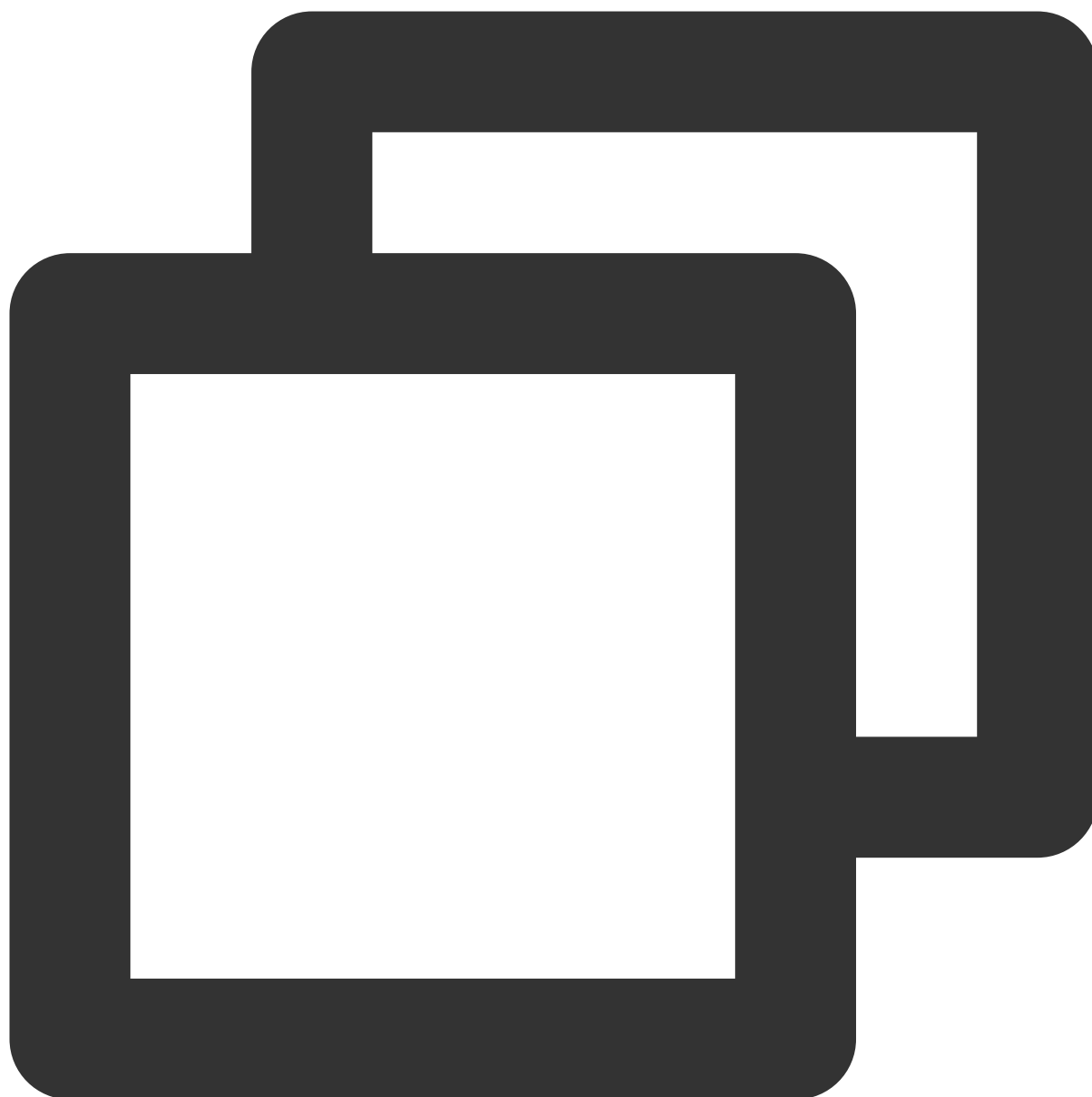
Unityの場合、プロジェクトのStreamingAssetsディレクトリに3Dファイルを配置し、SampleCodeの**copyFileFromAssetsToPersistent**関数を参照して、リソースファイルを各プラットフォームの適切なディレクトリにコピーすることをお勧めします。

Unrealの場合、SampleCodeの**CopyAllAssetsToExternal**関数を参照して、3Dモデルファイルをコピーしてからパスを読み込みます。

3D効果音のオン/オフ

この関数は、3Dサウンドエフェクトをオン/オフにするために使用されます。オンにした後、3Dサウンドエフェクトが聞こえます。

関数のプロトタイプ



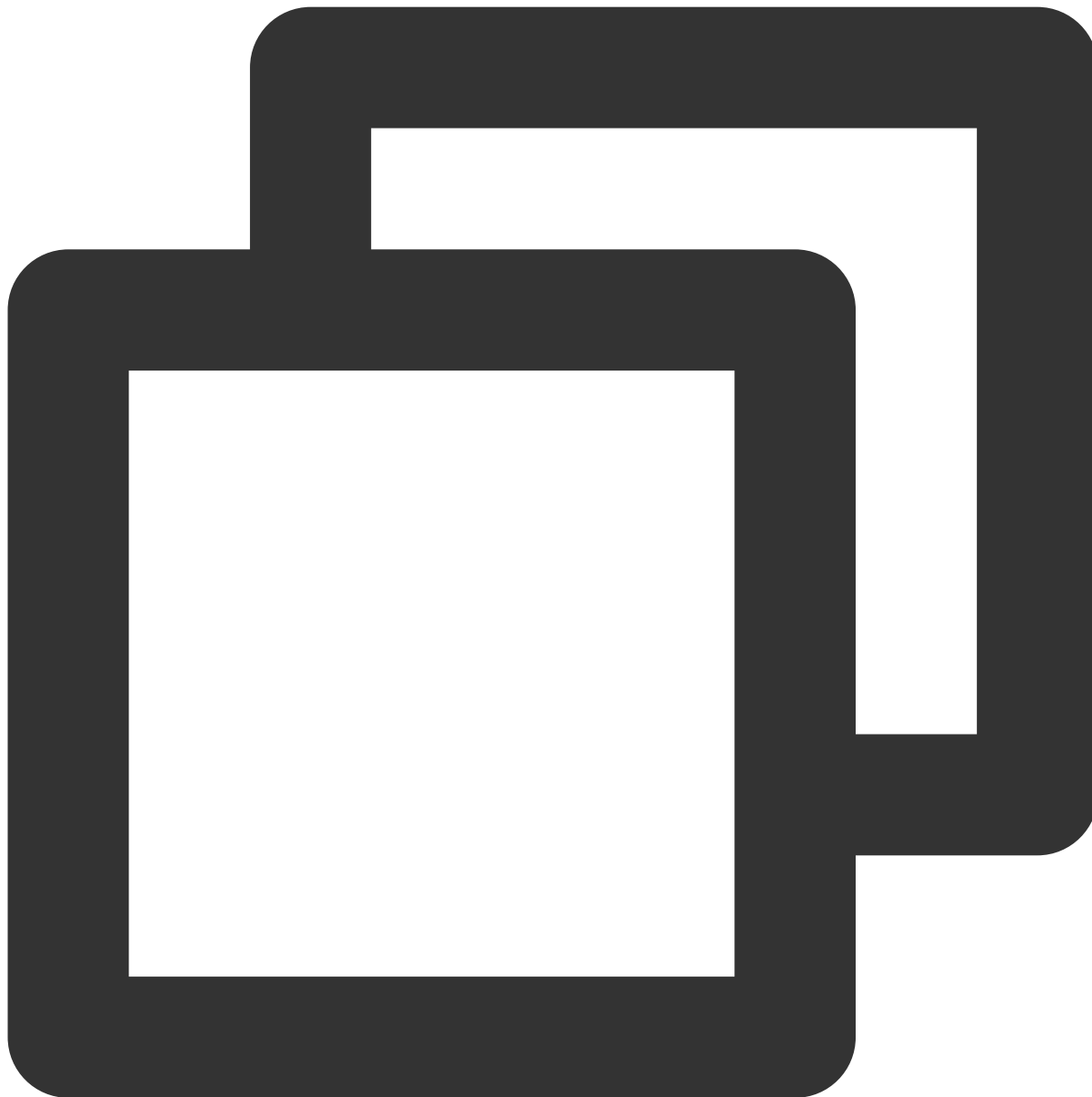
```
public abstract int EnableSpatializer(bool enable, bool applyToTeam)
```

パラメータ	タイプ	意味
enable	bool	オンにすると3D効果音が聞こえます
applyToTeam	bool	3D音声はチーム内で機能するかどうかを示します。enableがtrueである場合にのみ有効です。

現在の3D効果音状態の取得

この関数は、現在の3Dサウンドエフェクトの状態を取得するために使用されます。

関数のプロトタイプ



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

戻り値	意味
true	起動状態
false	オフ状態

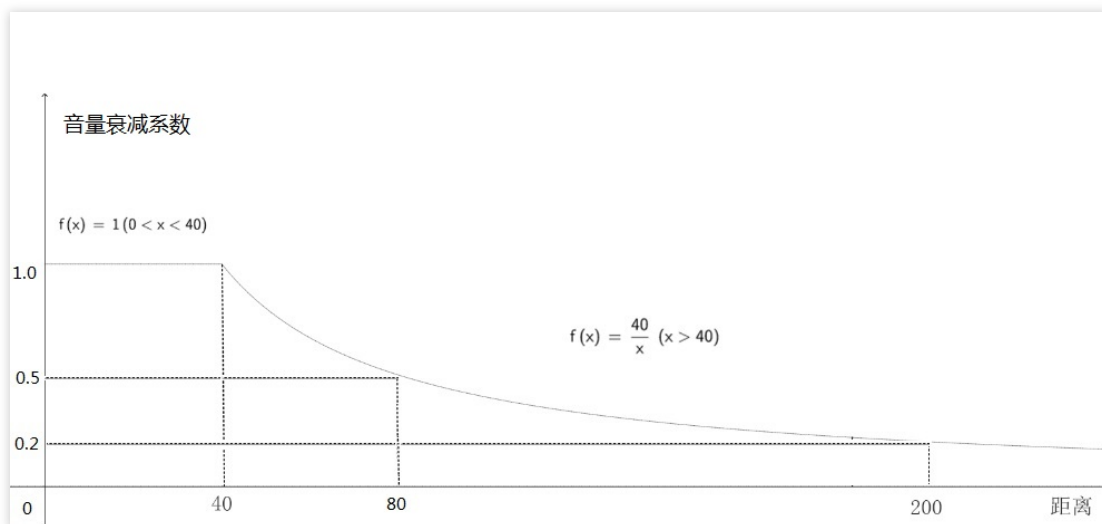
3D効果音の減衰距離の設定

減衰距離を設定する必要があります。推奨値は100です。

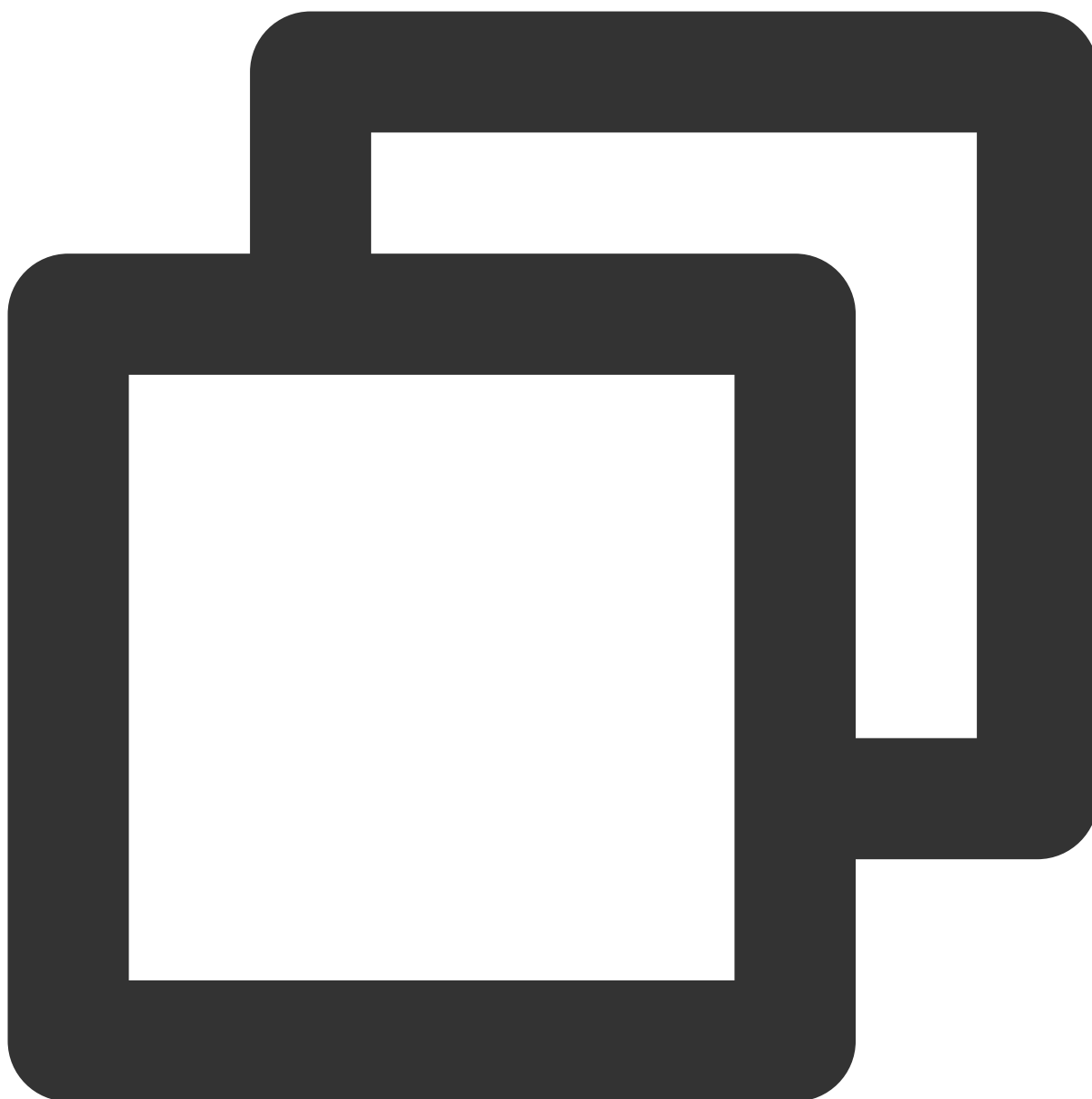
距離と音声減衰の関係

3Dサウンドエフェクトでは、音源のボリュームは音源の距離と減衰関係にあります。単位距離がrangeを超えた後、ボリュームはほぼゼロまで減衰します。

距離範囲（エンジン単位）	減衰公式
$0 < N < \text{range}/10$	減衰係数：1.0（音量が減衰しない）
$N \geq \text{range}/10$	減衰係数： $\text{range}/10/N$



関数のプロトタイプ



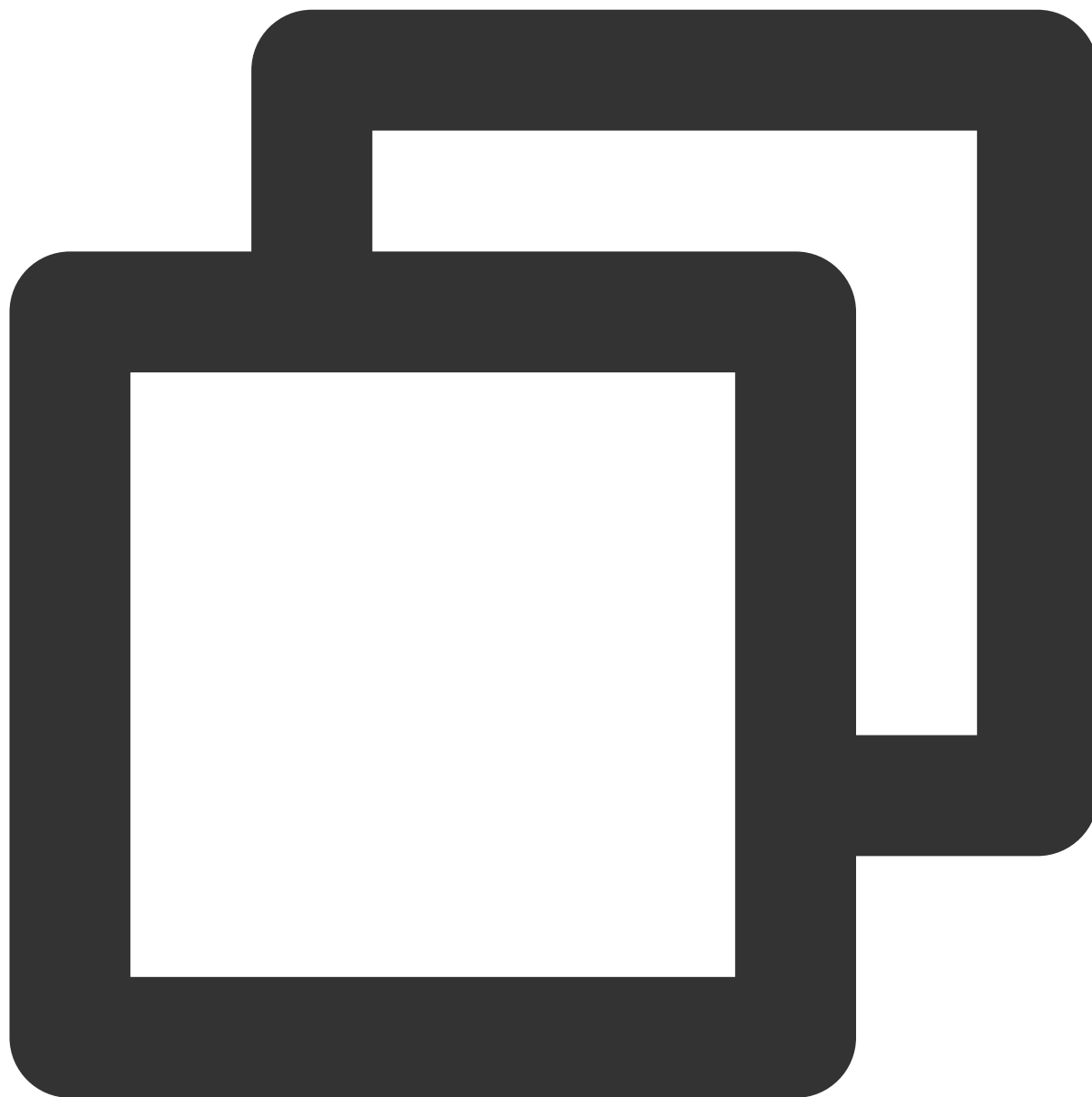
```
public abstract void UpdateAudioRecvRange(int range)
```

パラ メータ	タイ プ	意味
range	int	効果音の受信可能な範囲を設定します。推奨値は100です。この距離単位はゲームエンジン内の距離の単位です

音源の方位を更新する（向きを含む）

この関数は、音源の方位情報を更新するために使用され、フレームごとに呼び出すと、3Dサウンドエフェクトを実現できます。

関数のプロトタイプ



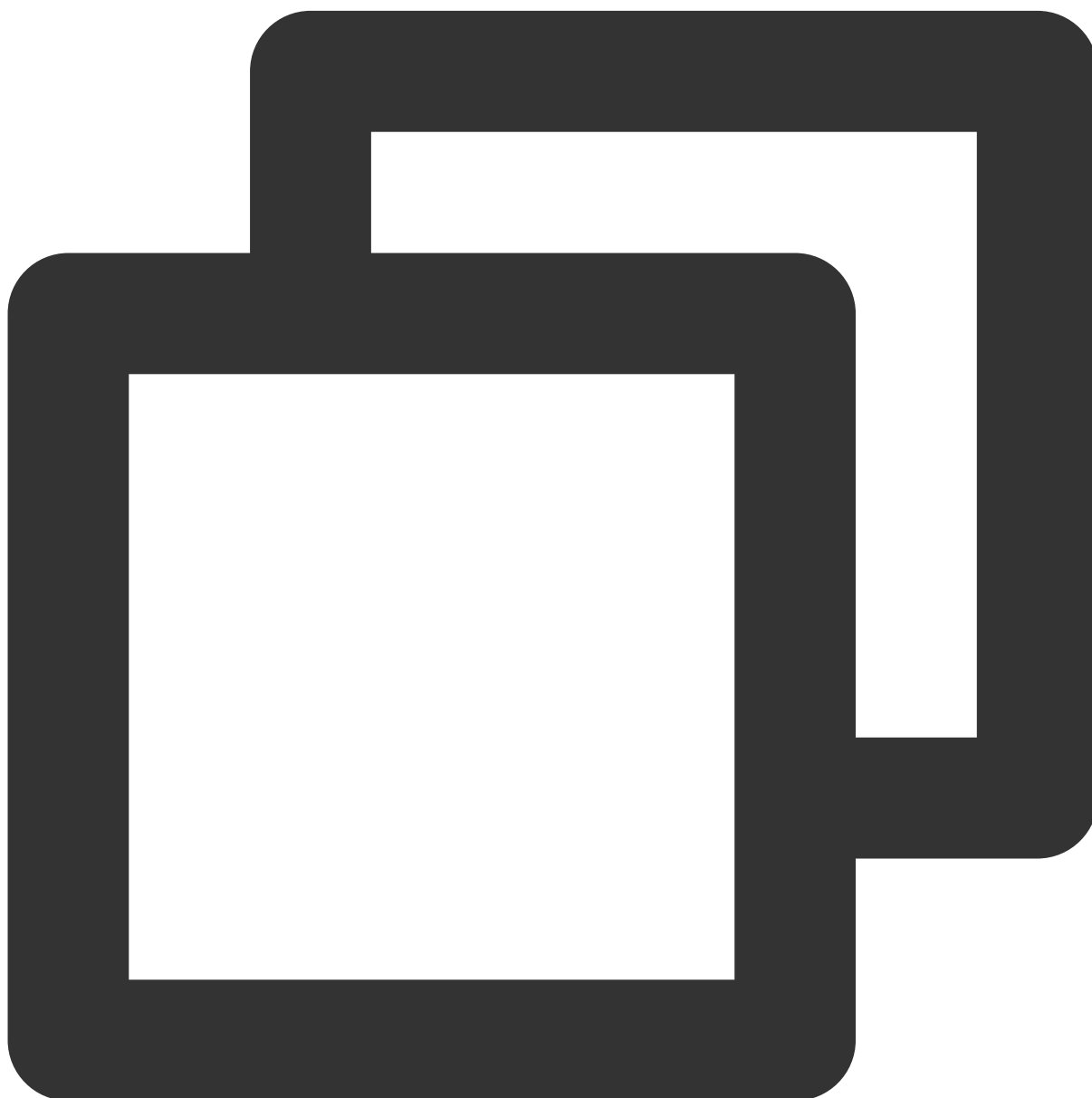
```
public abstract int UpdateSelfPosition(int position[3], float axisForward[3], float
```

パラメータ	タイプ	意味
position	int[]	ワールド座標系中の座標で、前、右、上の順序で表示されます

axisForward	float[]	ローカル座標系前軸の単位ベクトル
axisRight	float[]	ローカル座標系右軸の単位ベクトル
axisUp	float[]	ローカル座標系上軸の単位ベクトル

サンプルコード

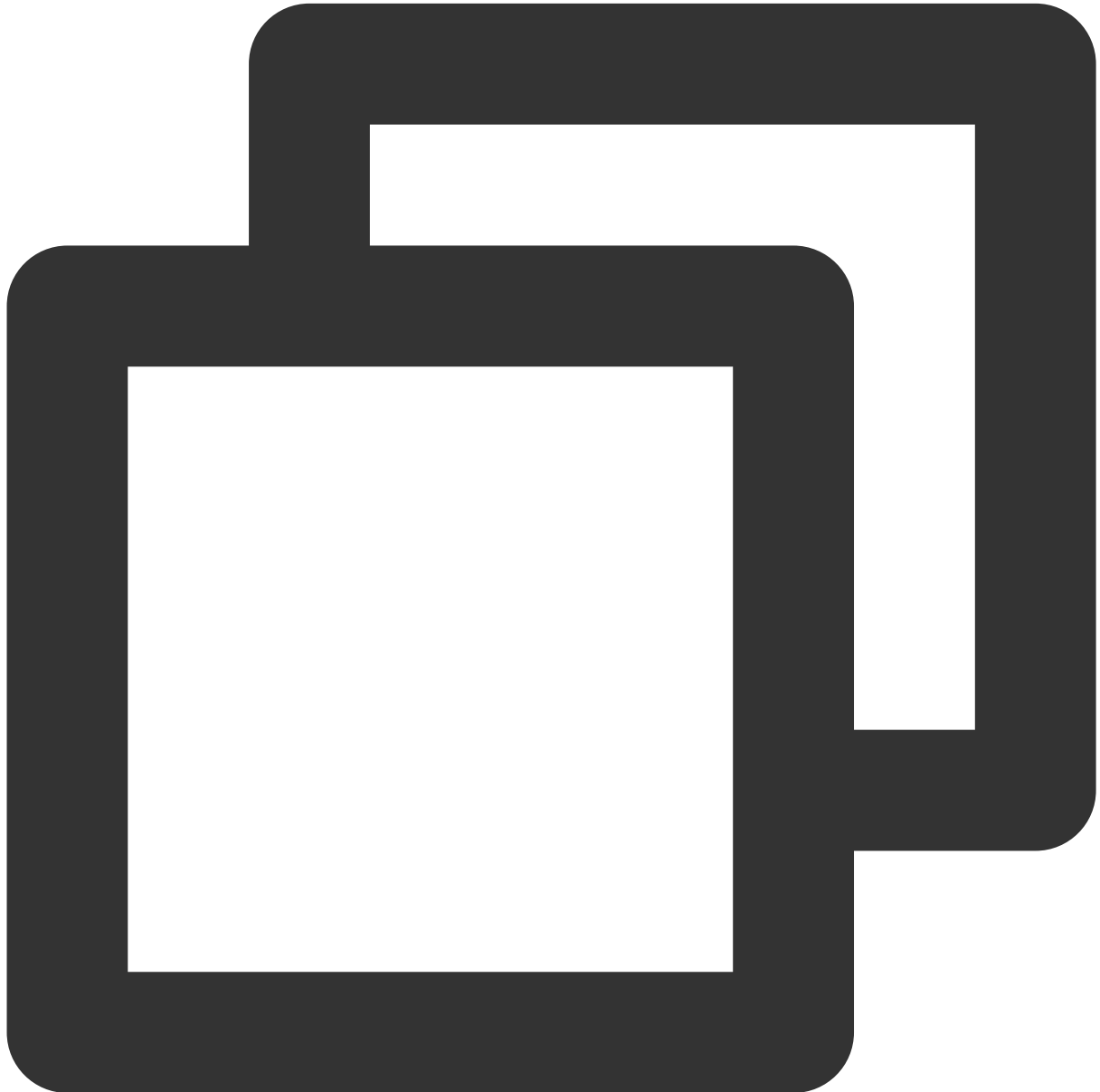
Unreal



```
FVector cameraLocation = UGameplayStatics::GetPlayerCameraManager(GetWorld(), 0)->G  
FRotator cameraRotation = UGameplayStatics::GetPlayerCameraManager(GetWorld(), 0)->
```

```
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, axisR
```

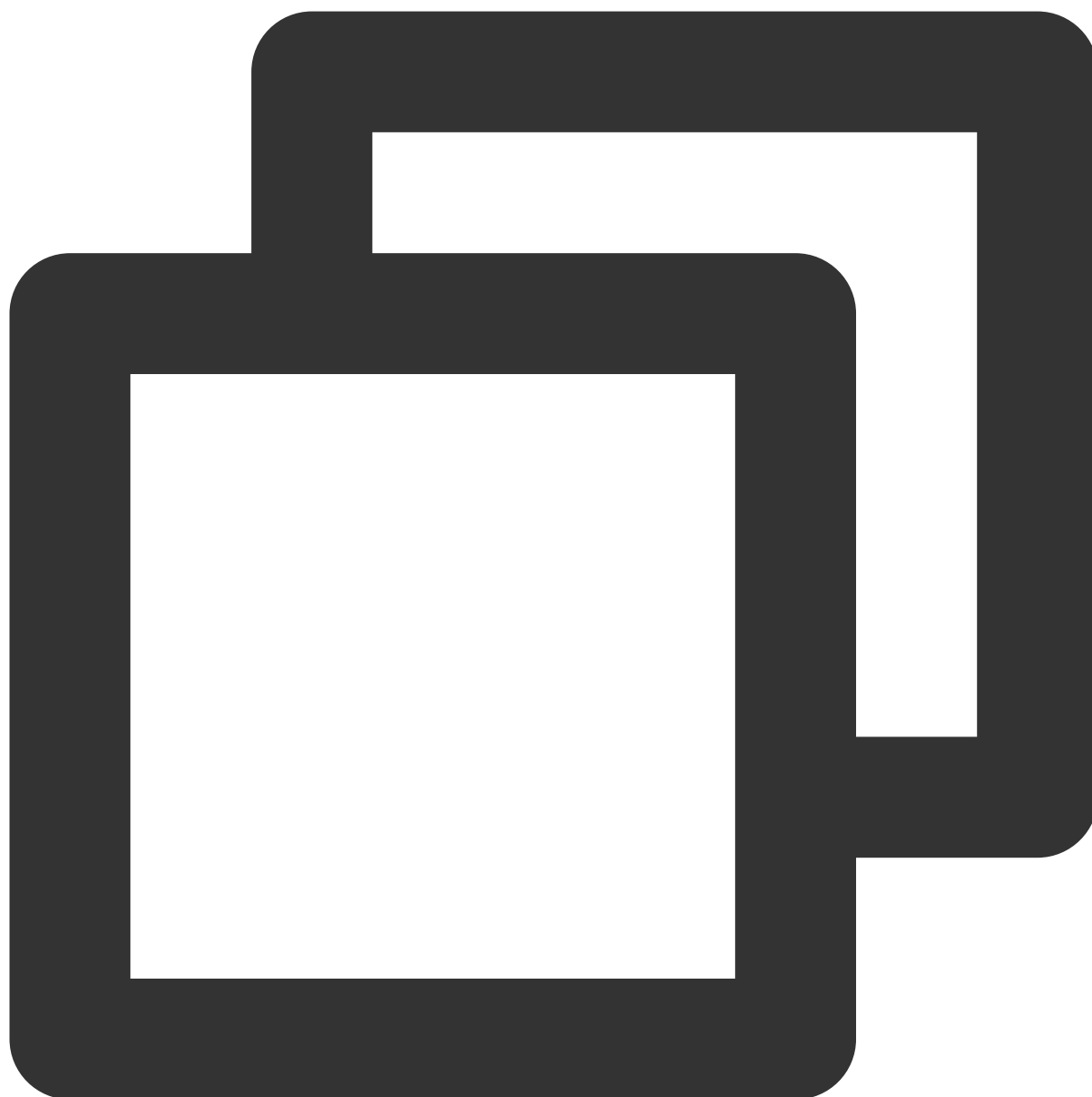
ローカル方位のインターフェース (VRシーン)

このインターフェースはVRデバイスの中でも、3Dの位置変化が非常に求められるシーンに適しています。この機能は高度なインターフェースであり、GME2.9.2以降が必要です。

通常の3Dシーンでは、ユーザーが単にUpdateSelfPosition関数を使用して自分の位置情報を更新し、ネットワークを介して他のユーザーに送信することができます。UpdateOtherPositionは他のプレイヤーの位置情報をネットワークを経由せずに、ローカルから送信することができ、VRゲームシーンに適しています。

リモートで更新された座標との競合を避けるために、UpdateOtherPositionを呼び出すと、リモート座標は破棄され、その影響は再びルームに入るまで残ります。したがって、**プレイヤーの座標をローカルに更新する場合は、すべてのプレイヤーの座標を更新してください。**

関数のプロトタイプ



```
public abstract int UpdateOtherPosition(int position[3])
```

パラメータ	タイプ	意味
position	int[]	世界座標での他のプレイヤーの座標であり、順序は前、右、上です

ご注意：

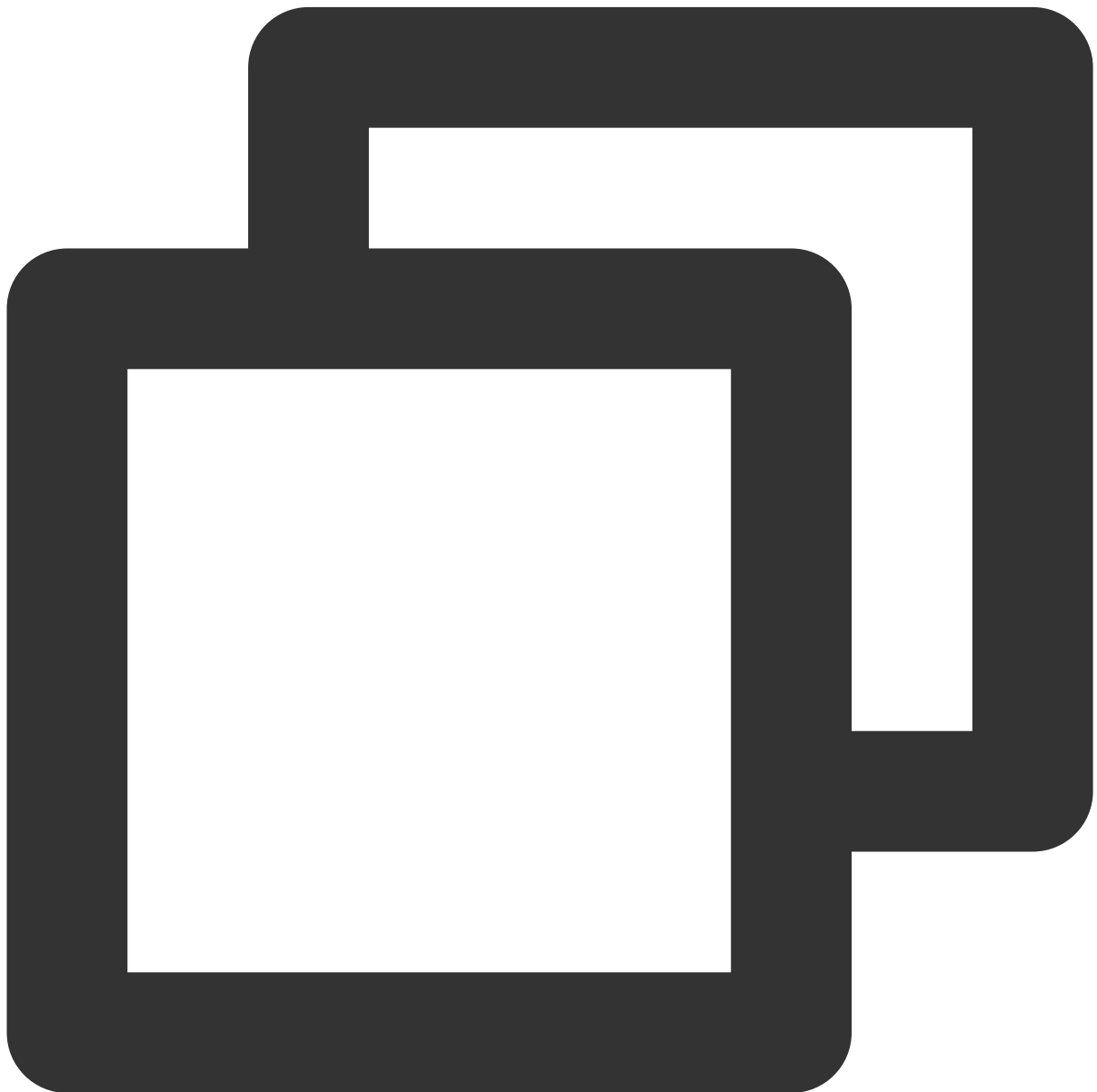
プレイヤーの座標をローカルに更新する場合は、**すべてのプレイヤーの座標**をトラバーサルし、このインターフェースを使用して座標を渡してください。

3D音声のブラックリストインターフェース

ご注意：

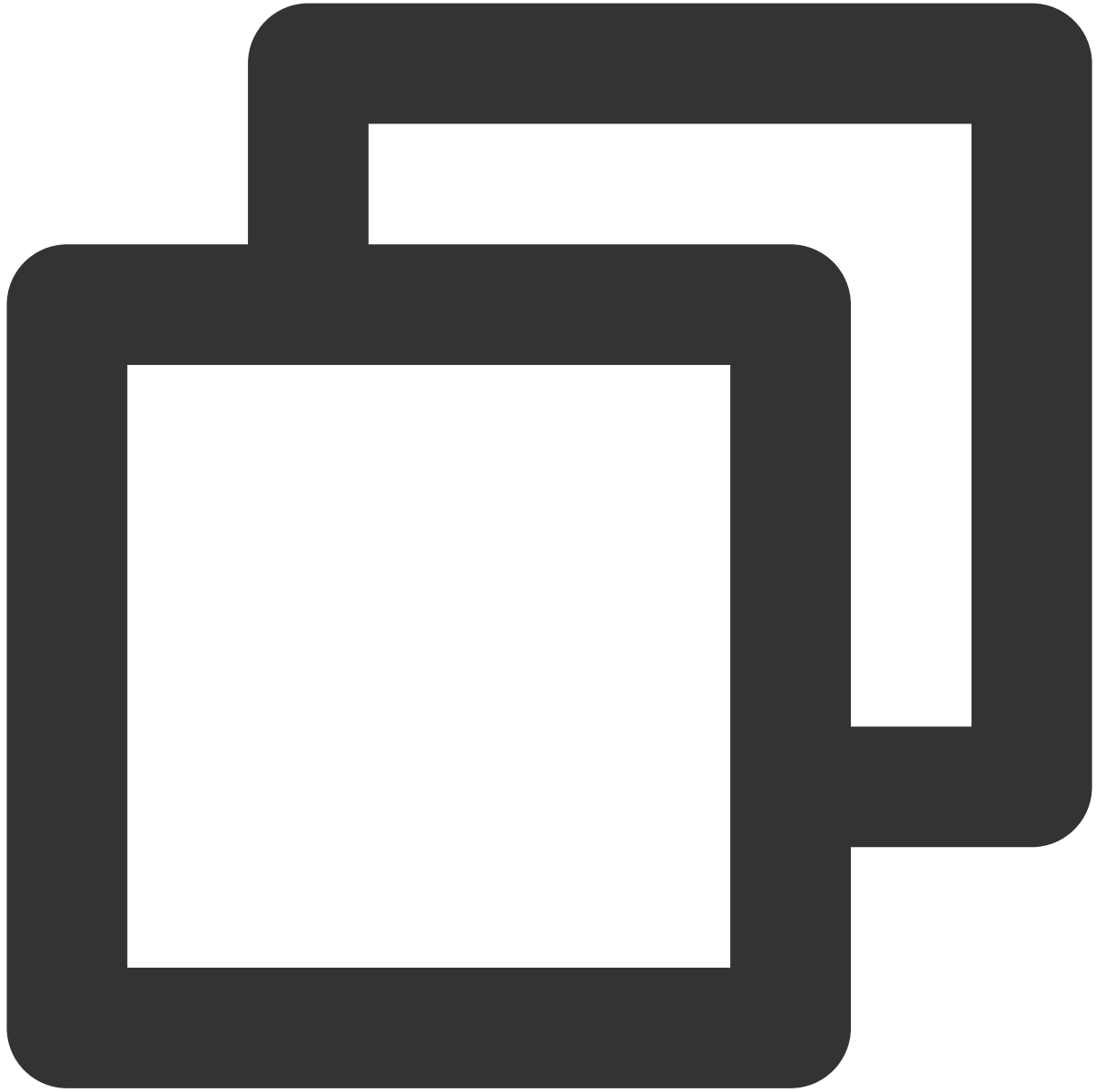
このインターフェースはGME2.9.3以降のSDKで有効になります。

現在、3D効果音を呼び出すとルーム内のすべての人に効果があります。特定のシーンでは、受信した人の音声は3D効果音によって減衰することが望ましくない場合、このインターフェースを呼び出すことで、その人を3D効果音ブラックリストに追加することができます。追加したら、このopenidの音声は3D効果音の影響を受けなくなります。



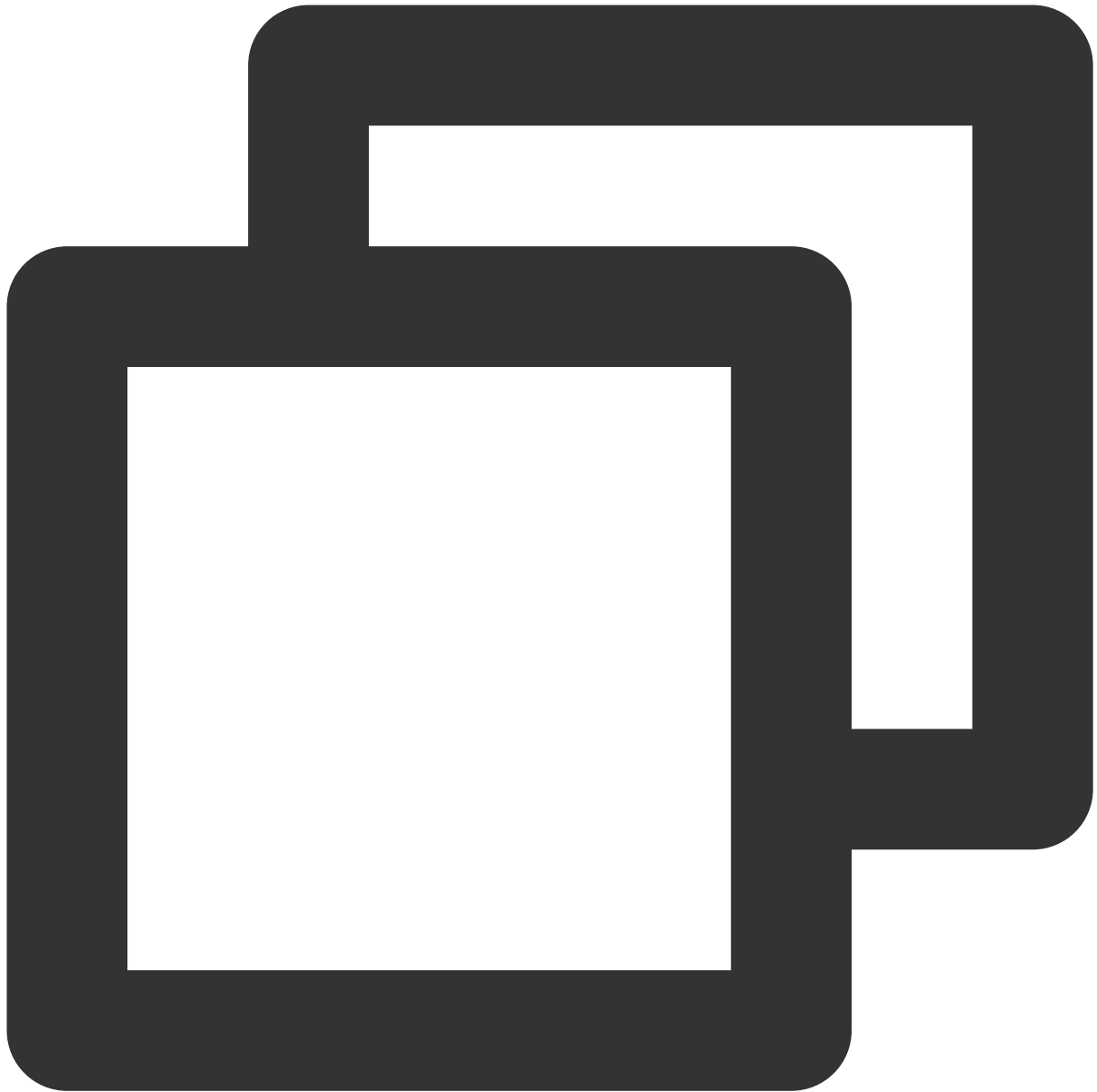
```
virtual int AddSpatializerBlacklist(const char* openId);
```

このopenidをブラックリストから削除する必要がある場合は、次のインターフェースを呼び出してください。



```
virtual int RemoveSpatializerBlacklist(const char* openId);
```

ブラックリストを空にする必要がある場合は、次のインターフェースを呼び出してください。



```
virtual int ClearSpatializerBlacklist();
```

トラブルシューティング

接続後に音声をテストしても3D効果音がない場合は、次の手順に従って確認してください：

1. 正常にルームに参加したか、マイクをオンにしたか？両方が声が聞こえるか？
2. デュアルチャンネルヘッドセットに適しているか？

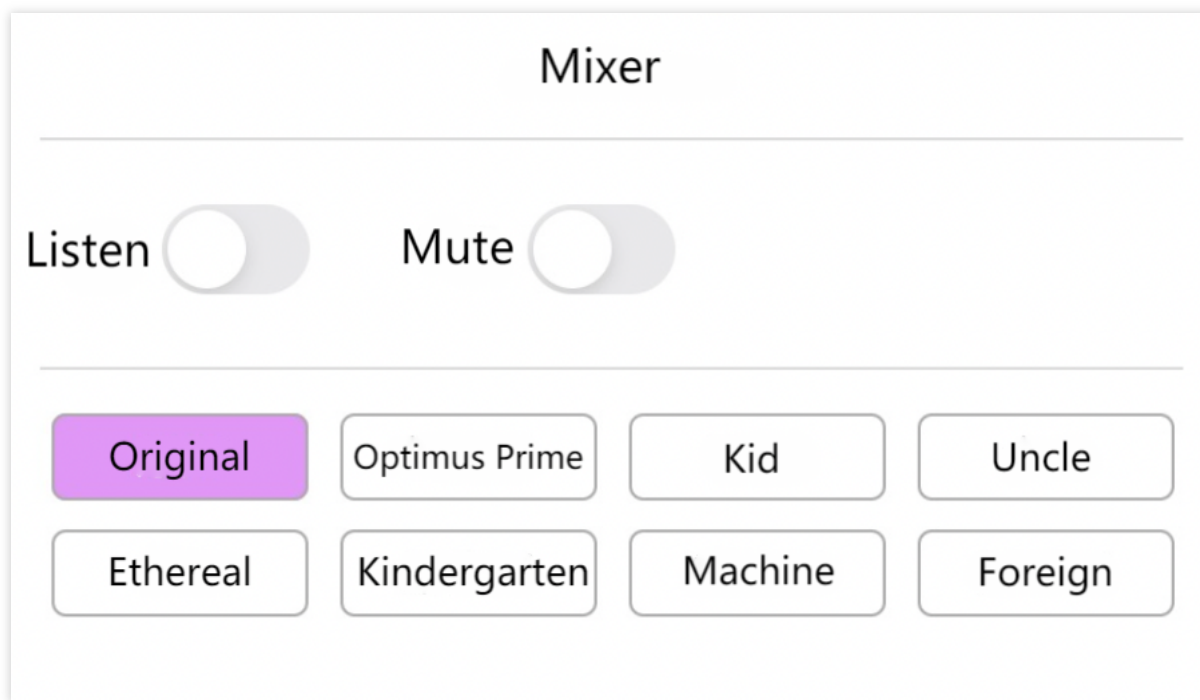
3. InitSpatializer インターフェースの戻り値は0か？
4. UpdateAudioRecvRange の設定が小さすぎないか？
5. UpdateSelfPosition インターフェースが定期的に呼び出されたか？
6. [エラーコードドキュメント](#)を使用して判断、解決します。

効果音と伴奏 ボイスチェンジ

最終更新日：：2024-01-18 15:47:47

開発者がTencent Cloud GME製品のAPIを容易にデバッグして導入するために、このドキュメントではGMEのボイス・チェンジ効果音の導入方法を紹介します。

シナリオ



前提条件

リアルタイムボイスサービスを有効にしました： [サービス有効化ガイド](#)をご参照ください。

ボイスツーテキスト変換サービスを有効にしました： [サービス有効化ガイド](#)をご参照ください。

GME SDK導入済み：コアインターフェースとリアルタイム音声インターフェースの導入が含まれます。詳細については、 [Native SDKのクイック導入](#)、 [Unity SDKクイック導入](#)、 [Unreal SDKクイック導入](#)をご参照ください。

- **GME SDKライブラリファイルlibgmesoundtouchを導入しました**：libgmesoundtouchがプロジェクトライブ

ラリファイルに含まれていることを確認する必要があります。具体的には、[ライブラリファイル対応機能](#)をご参照ください。

リアルタイム音声のボイス・チェンジの導入

ボイス・チェンジインターフェース

入室に成功し、マイクがオンになっている場合、`SetVoiceType`インターフェースを呼び出してボイスオーバー効果を設定します。インターフェースが0を返した場合、呼び出しに成功したことを示します。この場合、ルームにいる人は自端末からボイス・チェンジ効果のある音声を聞くことができます。ボイス・チェンジをセルフテストする場合は、インイヤ・モニタリング機能（インターフェース：`EnableLoopBack`）を使用します。

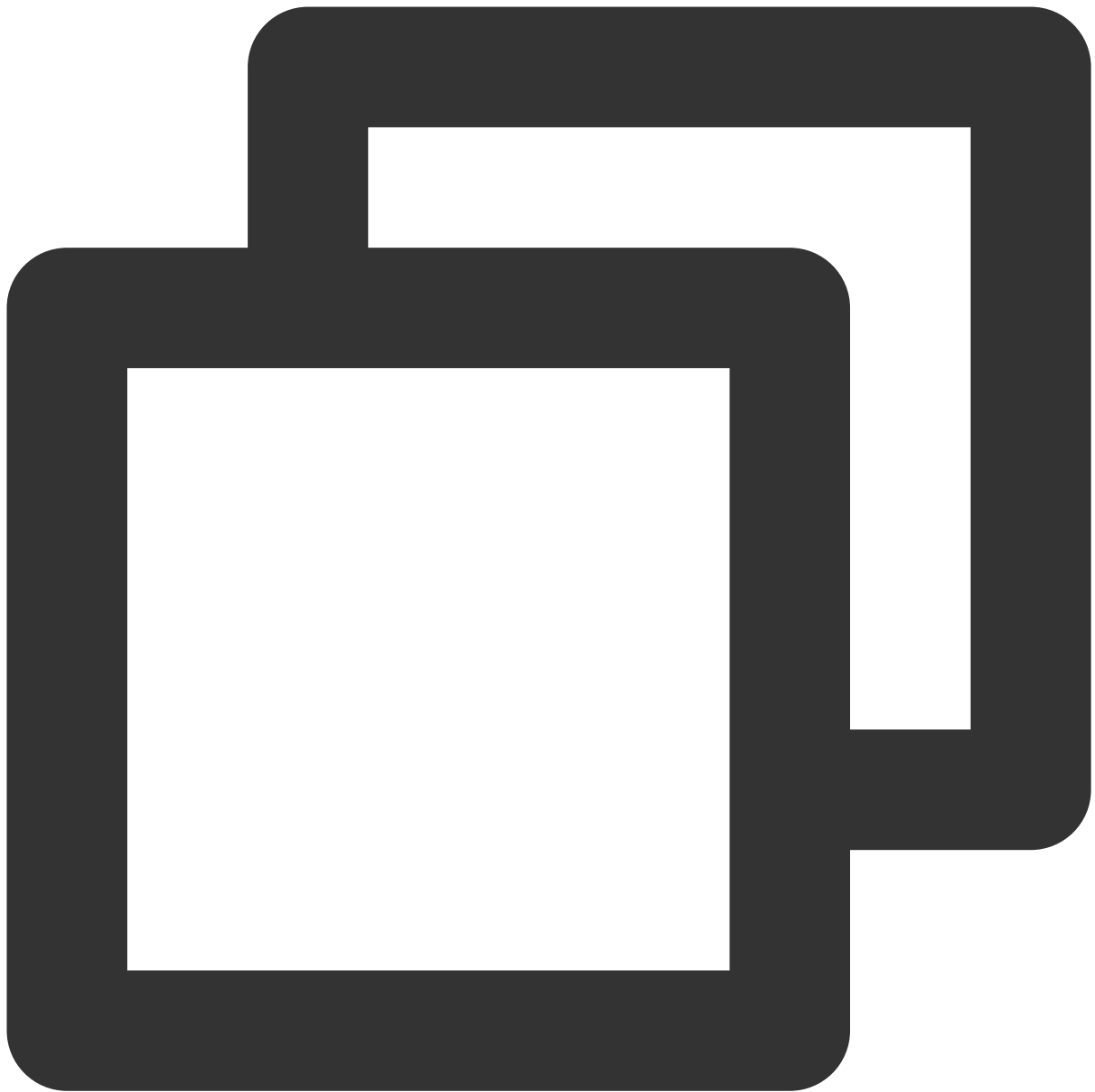
関数のプロトタイプ

Android

iOS

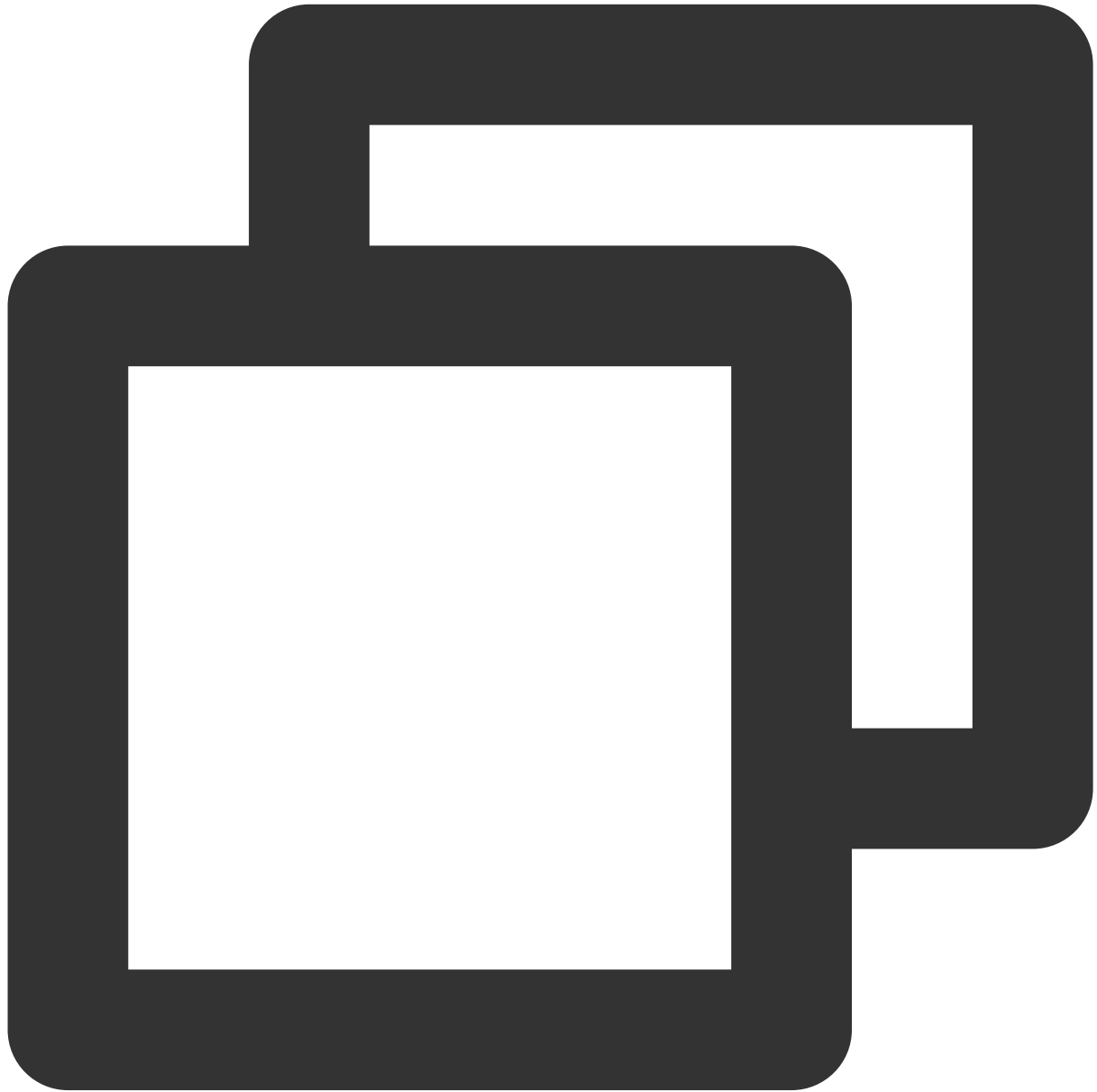
Unity

C++

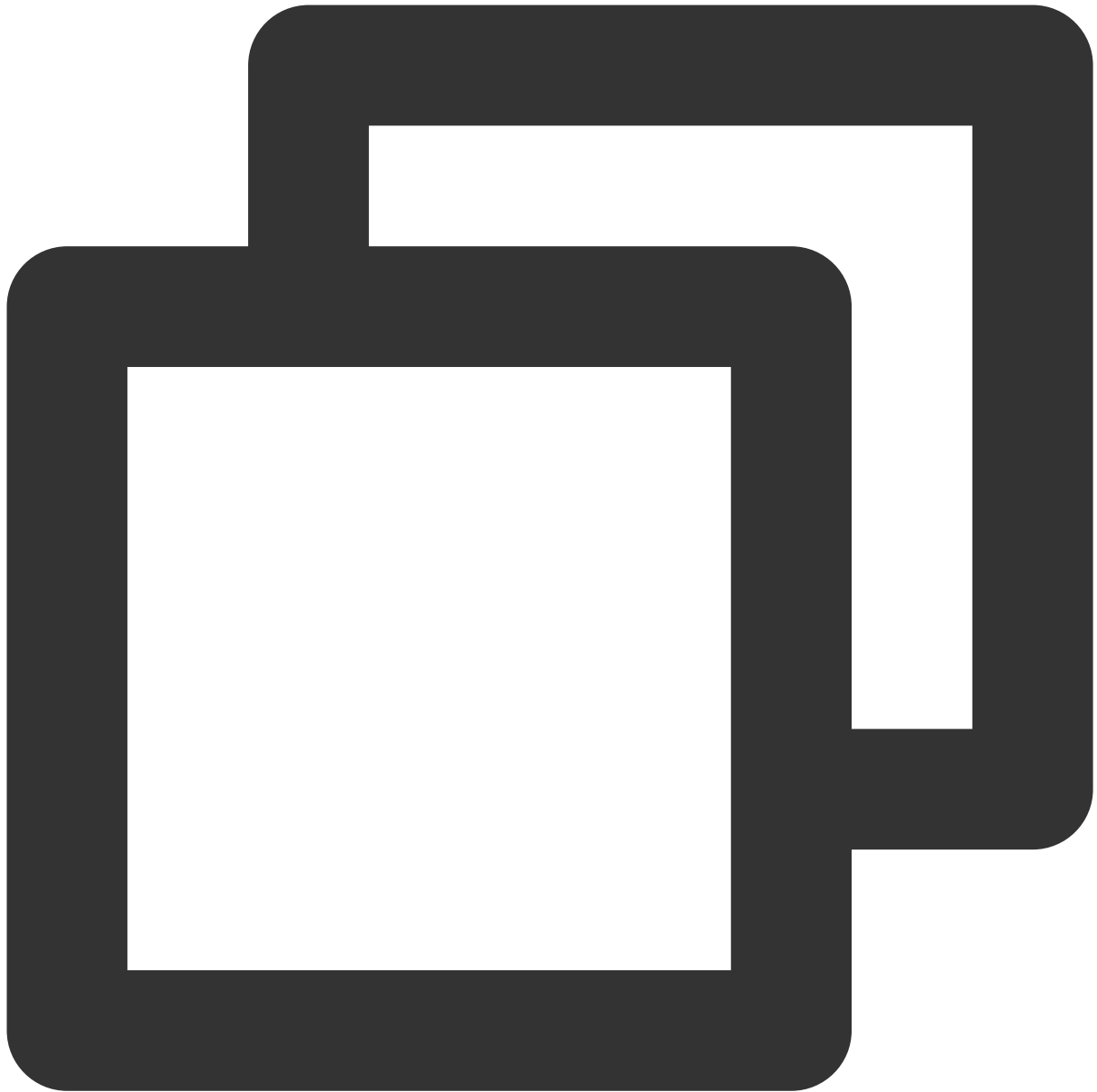


```
public static class ITMG_VoiceType {  
    public static final int ITMG_VOICE_TYPE_ORIGINAL_SOUND = 0;  
    public static final int ITMG_VOICE_TYPE_LOLITA = 1;  
    public static final int ITMG_VOICE_TYPE_UNCLE = 2;  
    public static final int ITMG_VOICE_TYPE_INTANGIBLE = 3;  
    public static final int ITMG_VOICE_TYPE_DEAD_FATBOY = 4;  
    public static final int ITMG_VOICE_TYPE_HEAVY_MENTAL = 5;  
    public static final int ITMG_VOICE_TYPE_DIALECT = 6;  
    public static final int ITMG_VOICE_TYPE_INFLUENZA = 7;  
    public static final int ITMG_VOICE_TYPE_CAGED_ANIMAL = 8;  
    public static final int ITMG_VOICE_TYPE_HEAVY_MACHINE = 9;  
}
```

```
public static final int ITMG_VOICE_TYPE_STRONG_CURRENT = 10;  
public static final int ITMG_VOICE_TYPE_KINDER_GARTEN = 11;  
public static final int ITMG_VOICE_TYPE_HUANG = 12;  
};  
public abstract int SetVoiceType(int type);
```

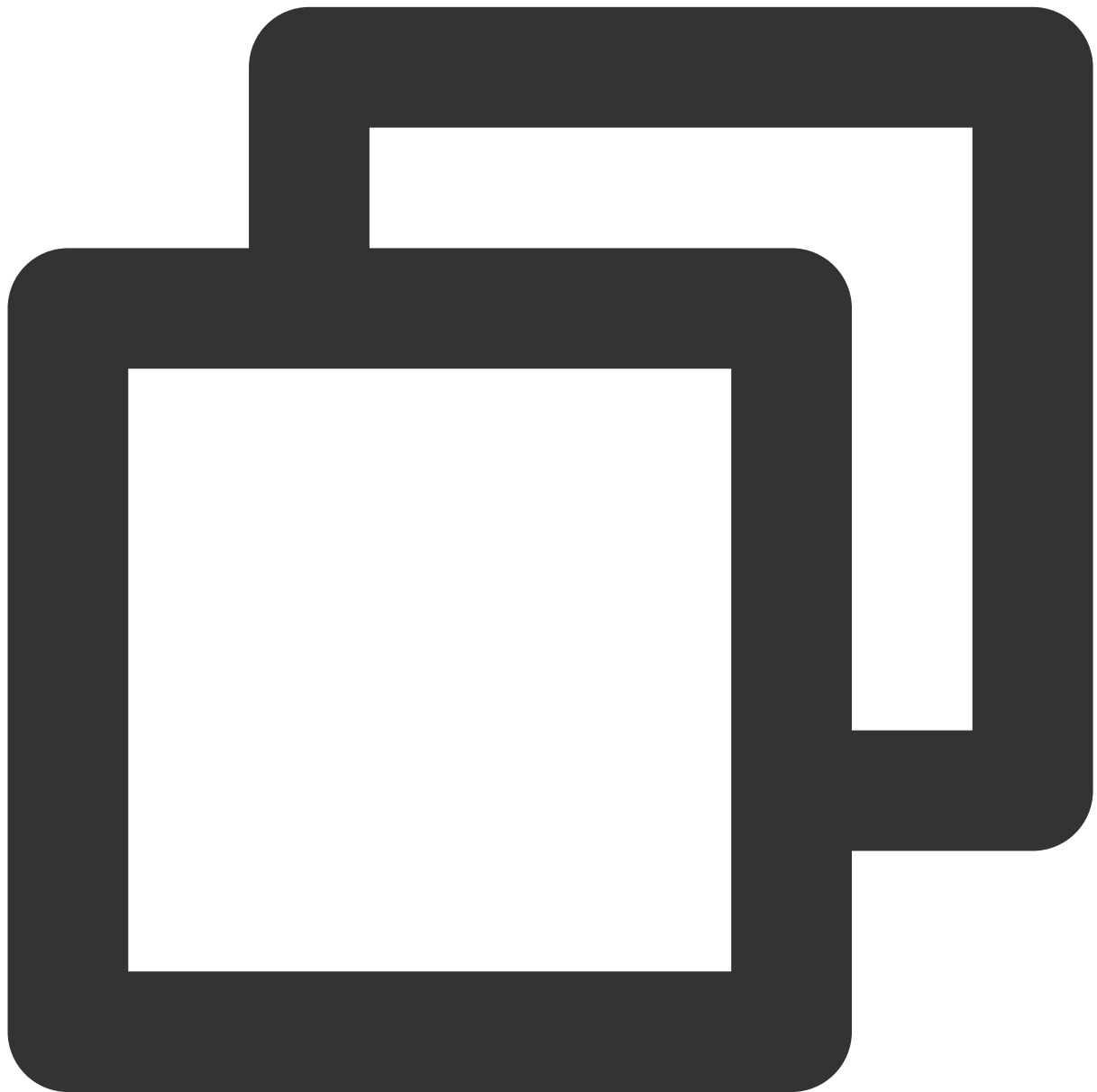


```
-(QAVResult) SetVoiceType:(ITMG_VOICE_TYPE) type
```



```
public abstract class ITMGAudioEffectCtrl{
    public static int VOICE_TYPE_ORIGINAL_SOUND = 0;
    public static int VOICE_TYPE_LOLITA = 1;
    public static int VOICE_TYPE_UNCLE = 2;
    public static int VOICE_TYPE_INTANGIBLE = 3;
    public static int VOICE_TYPE_DEAD_FATBOY = 4;
    public static int VOICE_TYPE_HEAVY_MENTAL = 5;
    public static int VOICE_TYPE_DIALECT = 6;
    public static int VOICE_TYPE_INFLUENZA = 7;
    public static int VOICE_TYPE_CAGED_ANIMAL = 8;
    public static int VOICE_TYPE_HEAVY_MACHINE = 9;
```

```
public static int VOICE_TYPE_STRONG_CURRENT = 10;  
public static int VOICE_TYPE_KINDER_GARTEN = 11;  
public static int VOICE_TYPE_HUANG = 12;  
public abstract int SetVoiceType(int voiceType);  
}
```



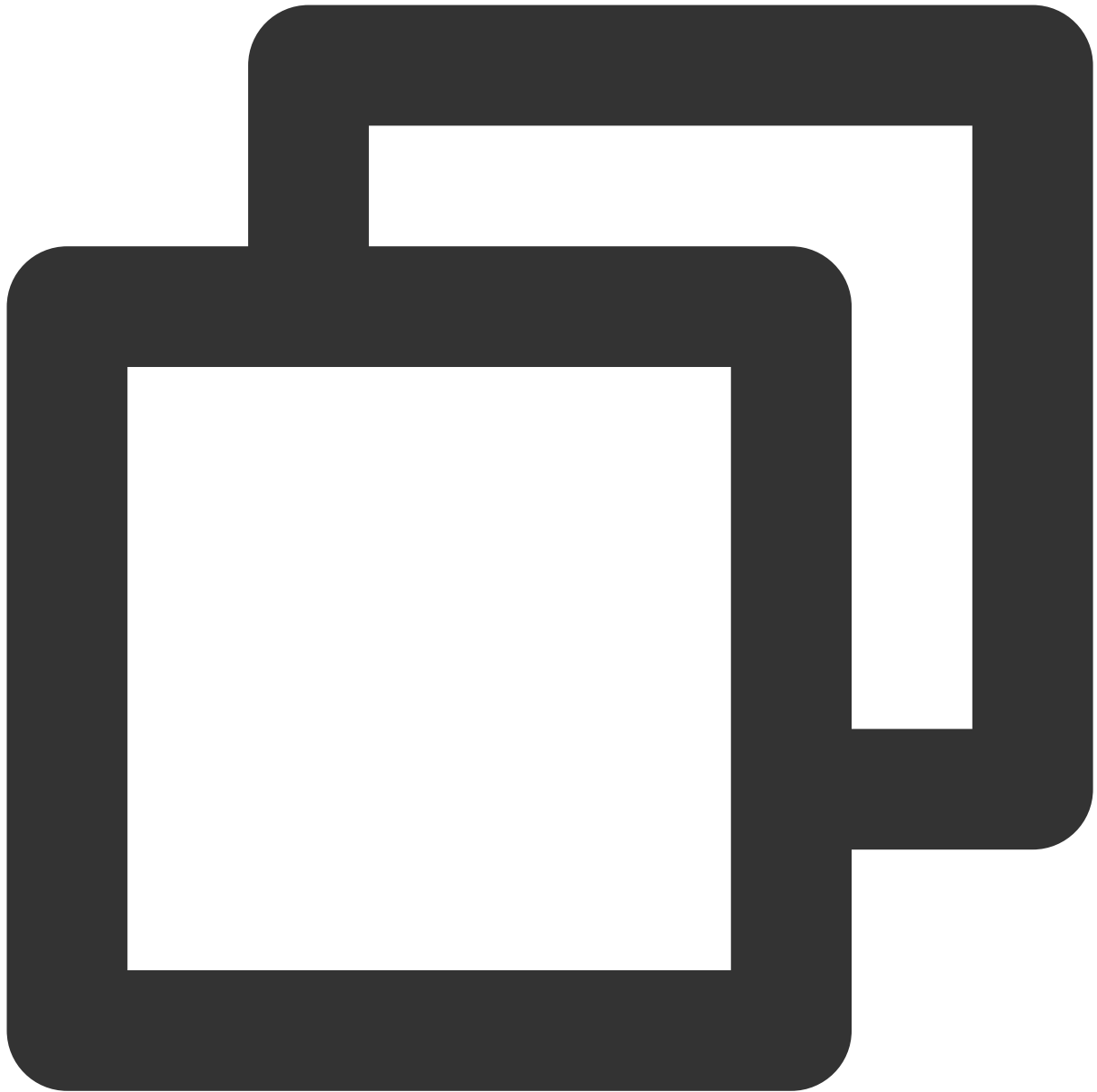
```
class ITMGAudioEffectCtrl {  
public:  
    virtual ~ITMGAudioEffectCtrl(){};  
    virtual int SetVoiceType(ITMG_VOICE_TYPE voiceType) = 0;
```

```
}
```

パラメータ	タイプ	意味
type	int	ローカル側のボイス変更タイプを示す

タイプパラメータ	パラメータ代表	意味
ITMG_VOICE_TYPE_ORIGINAL_SOUND	0	原音
ITMG_VOICE_TYPE_LOLITA	1	ロリ
ITMG_VOICE_TYPE_UNCLE	2	おじさん
ITMG_VOICE_TYPE_INTANGIBLE	3	ファンタジー
ITMG_VOICE_TYPE_DEAD_FATBOY	4	オタク
ITMG_VOICE_TYPE_HEAVY_MENTA	5	ヘビーメタル
ITMG_VOICE_TYPE_DIALECT	6	外国人のようになまりがある声です。
ITMG_VOICE_TYPE_INFLUENZA	7	風邪
ITMG_VOICE_TYPE_CAGED_ANIMAL	8	絶望
ITMG_VOICE_TYPE_HEAVY_MACHINE	9	ヘビーマシン
ITMG_VOICE_TYPE_STRONG_CURRENT	10	強電流
ITMG_VOICE_TYPE_KINDER_GARTEN	11	幼稚園
ITMG_VOICE_TYPE_HUANG	12	ミニオン

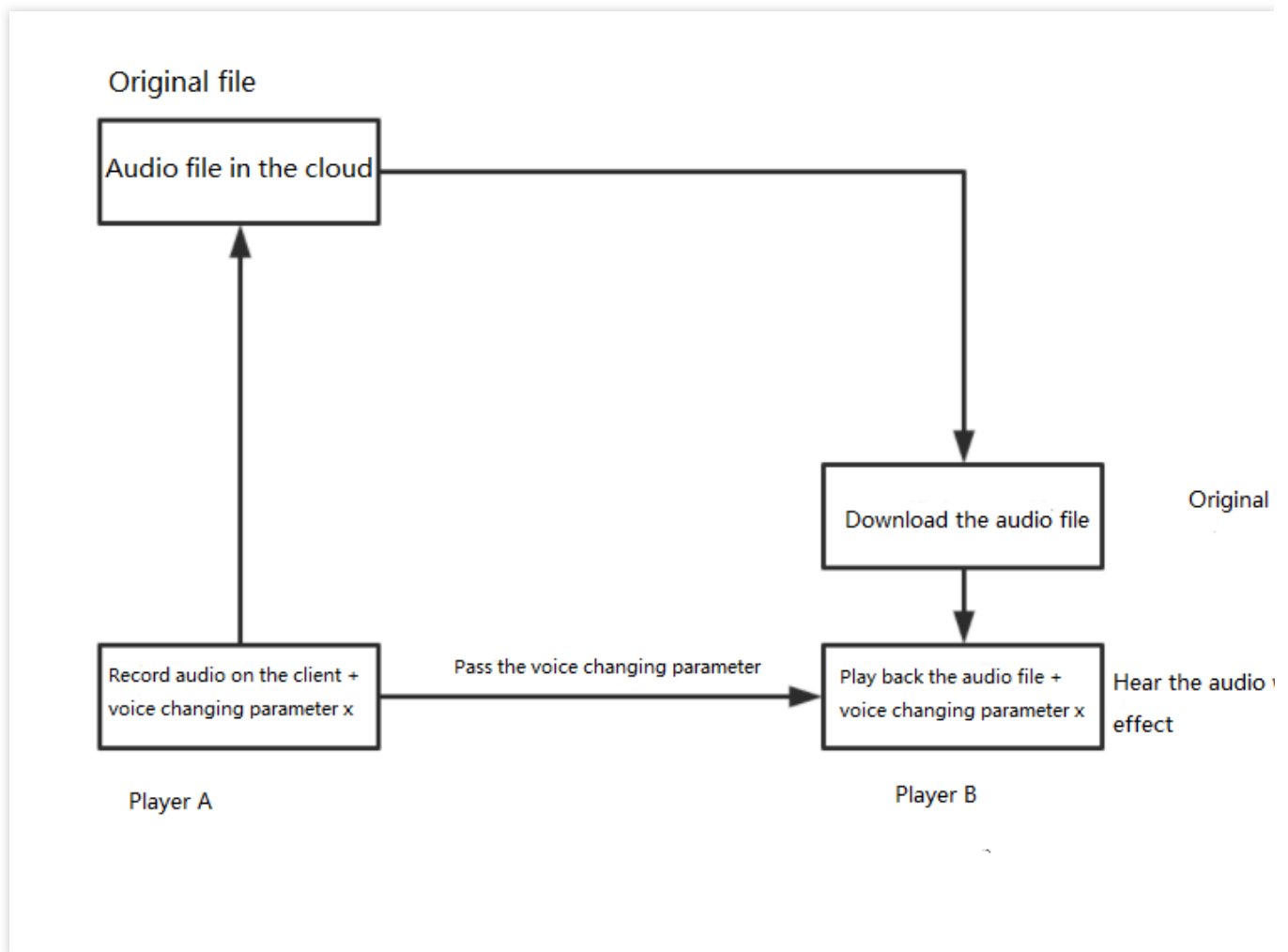
サンプルコード



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

音声メッセージのボイス・チェンジ導入

音声メッセージボイス・チェンジ手順



音声メッセージのボイス・チェンジは元のオーディオ情報に影響を与えず、再生時にボイス・チェンジ効果が反映されます。

音声メッセージ再生

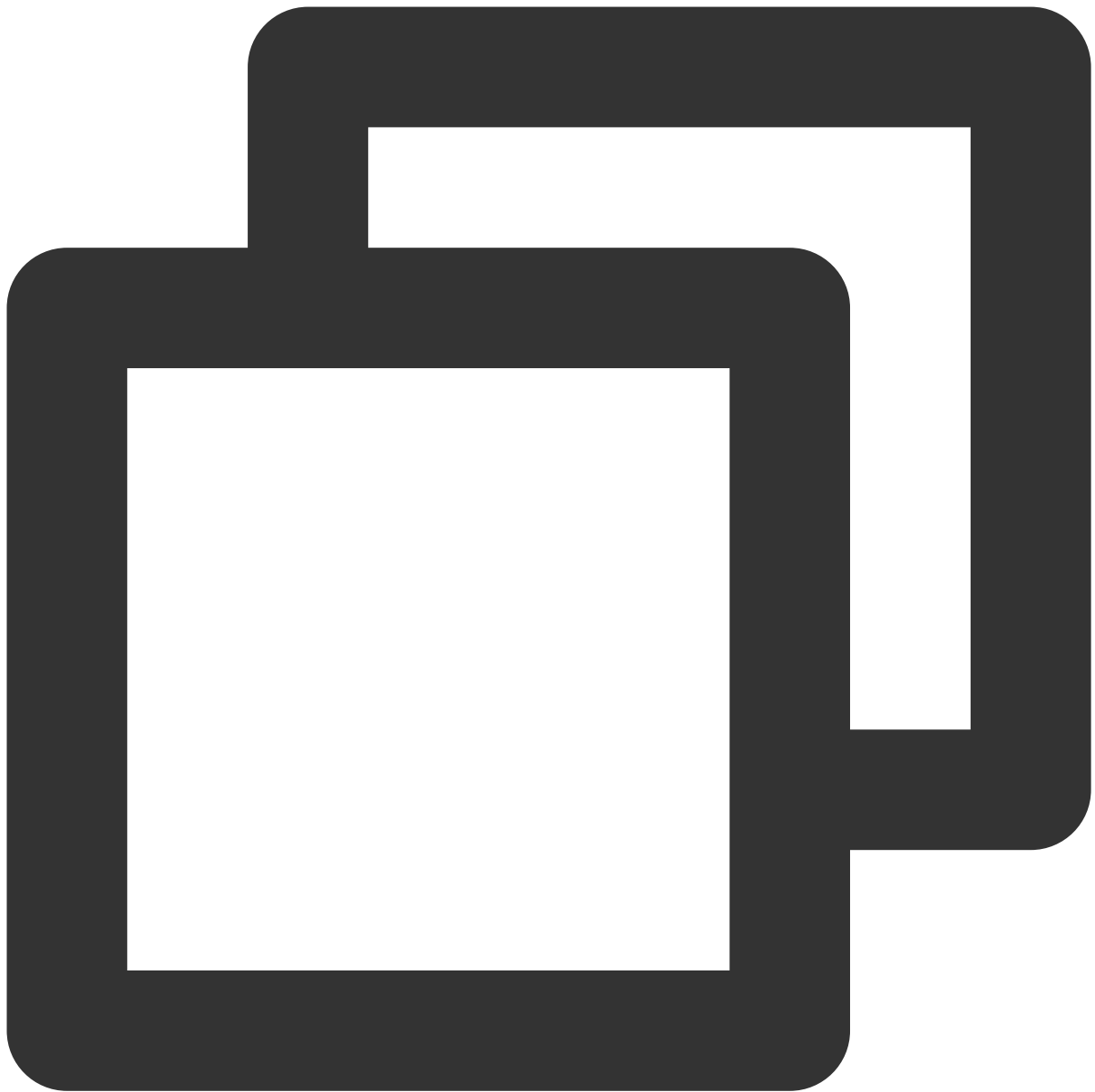
ボイス・チェンジパラメータ付きの音声メッセージ再生インターフェース。

Android

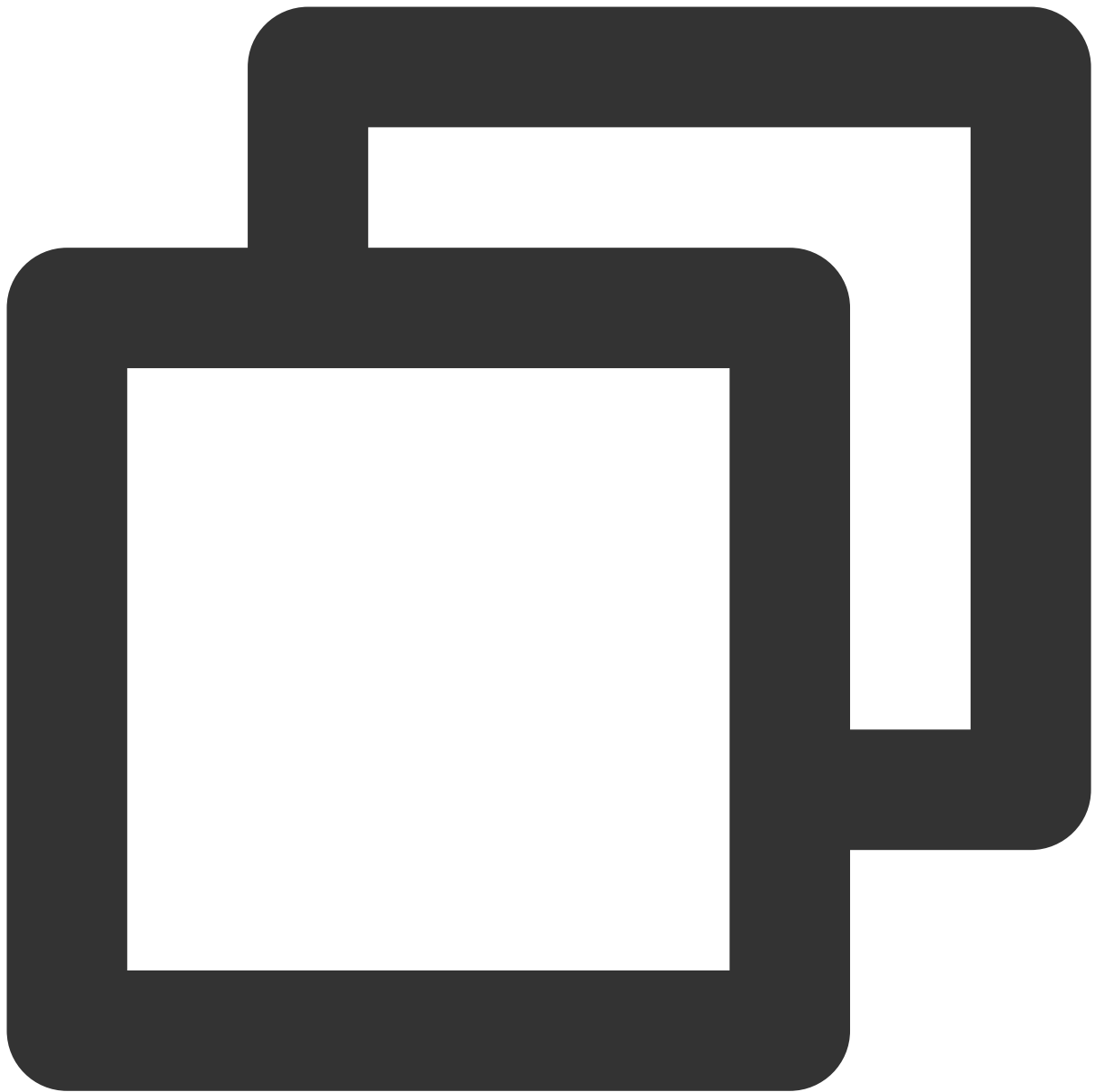
iOS

Unity

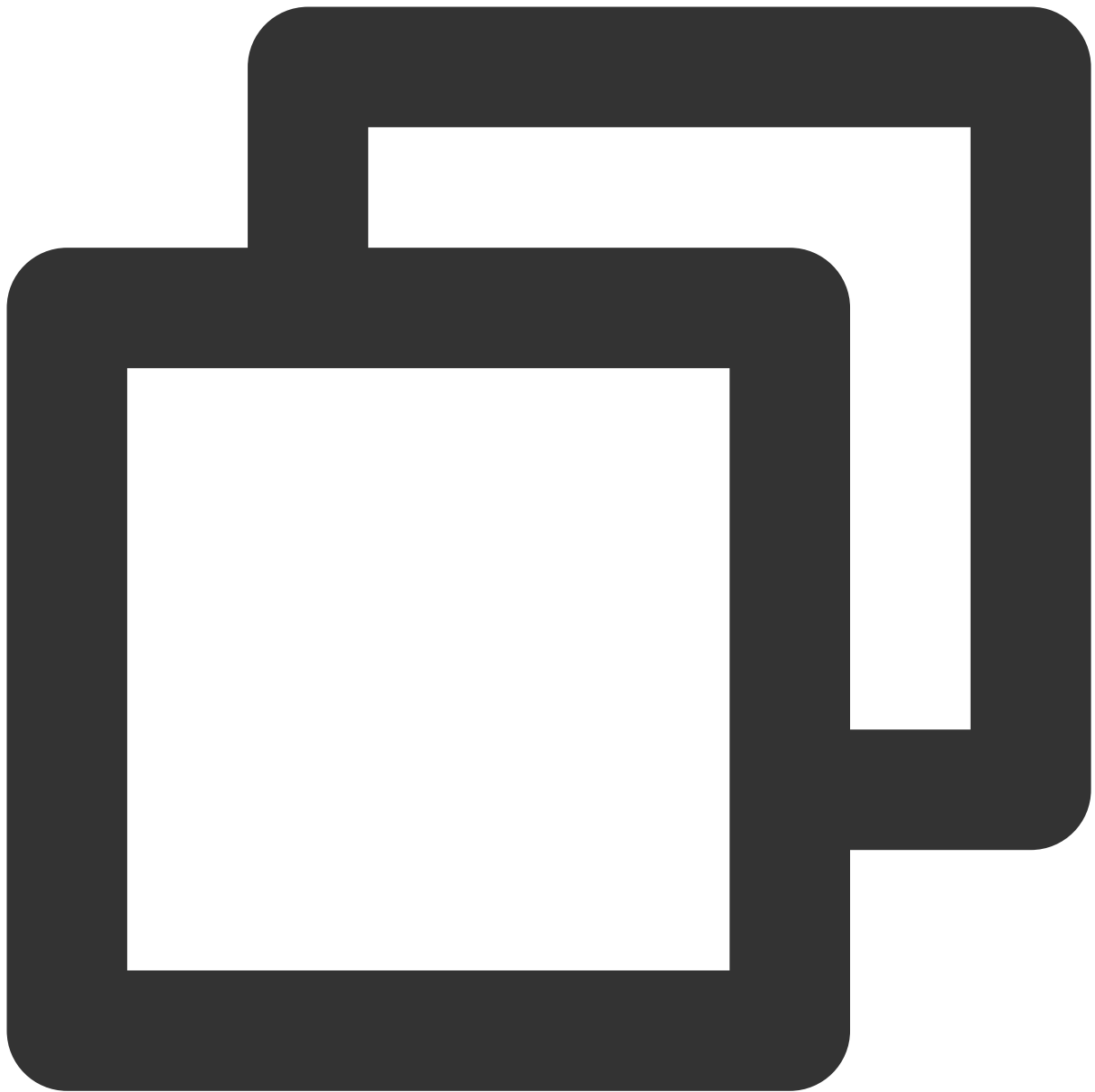
C++



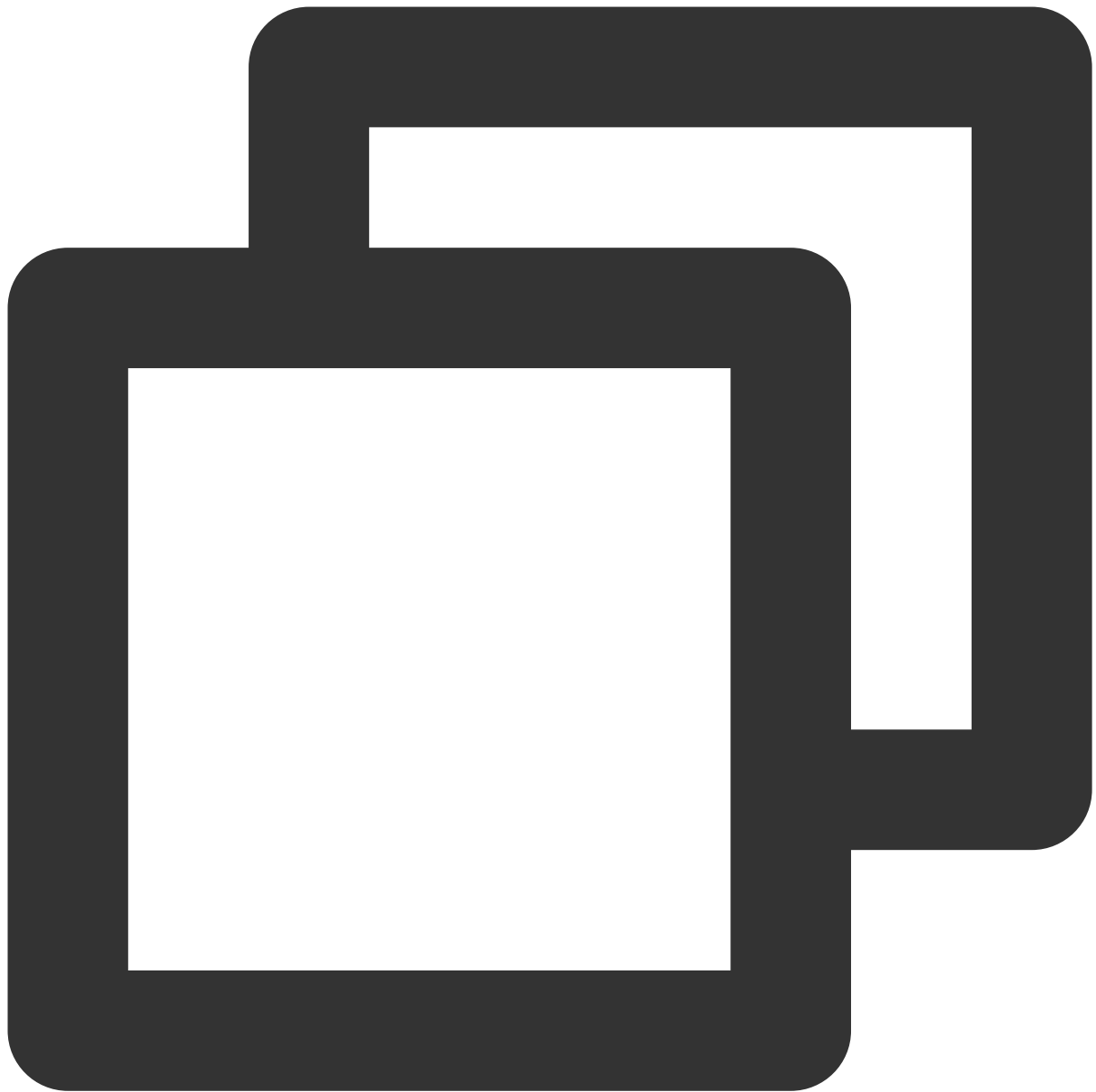
```
public abstract int PlayRecordedFile(String filePath,int voicetype);
```



```
-(int)PlayRecordedFile:(NSString*)filePath VoiceType:(ITMG_VOICE_TYPE) type
```



```
ITMGPTT PlayRecordedFile(string filePath,int voiceType);
```



```
public abstract int PlayRecordedFile(string filePath,int voiceType);
```

パラメータ	タイプ	意味
filePath	string	ローカル音声ファイルのパス
voicetype	int	ボイス・チェンジタイプ

エラーコード

エラーコード	原因	解決策
20485	再生が開始されていません	ファイルがあるかどうか、及びファイルパスの正当性を確認します

リアルタイム音声のBGM

最終更新日：2024-01-18 15:47:47

開発者がTencent Cloud Gaming Multimedia Engine製品APIのデバッグ・アクセスを行いやすいように、ここで、Gaming Multimedia Engineリアルタイム音声伴奏のインポート技術ドキュメントを説明させていただきます。

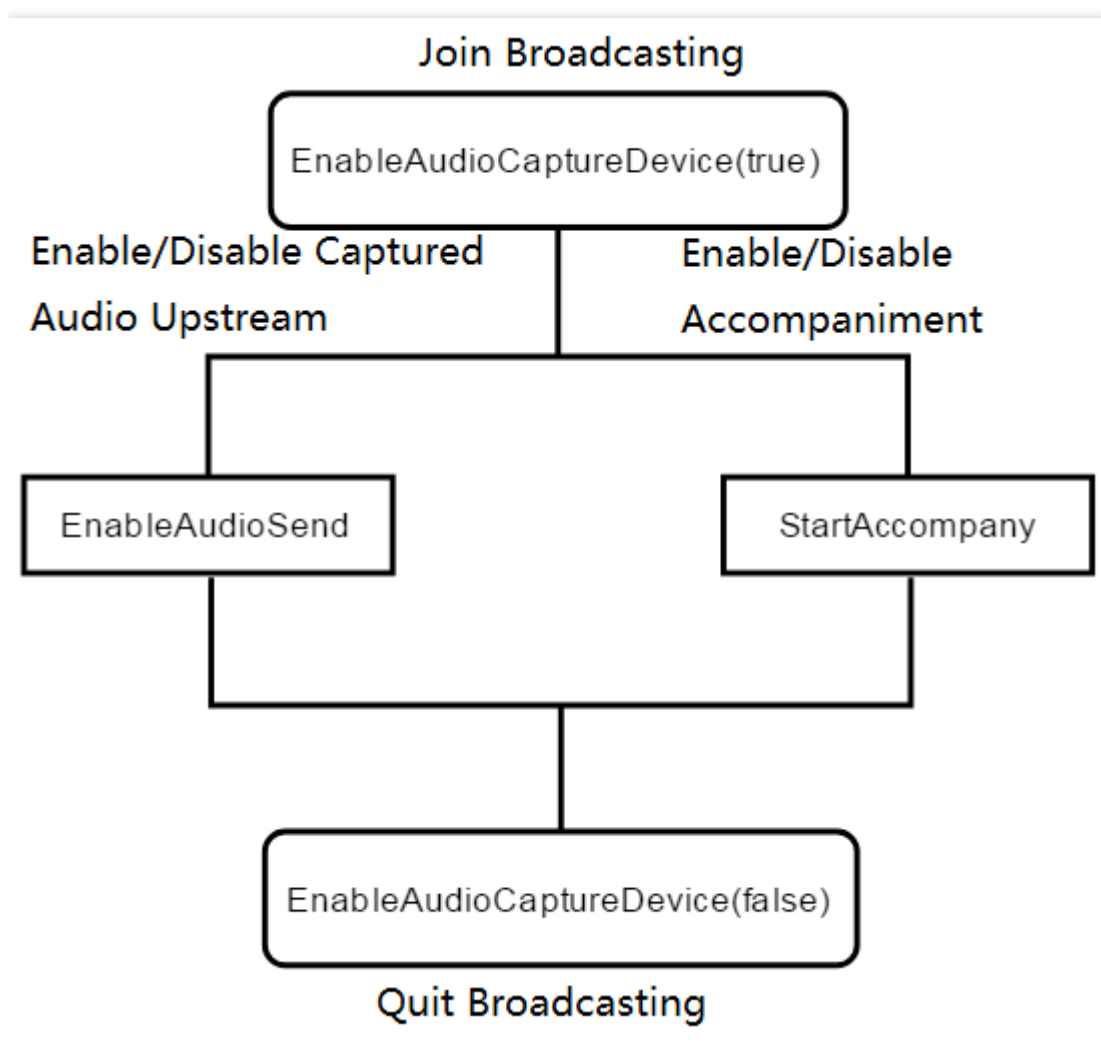
リアルタイム音声伴奏の関連インターフェース

インターフェース	インターフェースの意味
StartAccompany	伴奏再生を開始します。
StopAccompany	伴奏再生を停止します。
IsAccompanyPlayEnd	伴奏再生が終わっているかどうか。
PauseAccompany	伴奏再生を一時停止します。
ResumeAccompany	伴奏再生を再開します。
SetAccompanyVolume	伴奏ボリュームを設定します。
GetAccompanyVolume	伴奏再生ボリュームを取得します。
SetAccompanyFileCurrentPlayedTimeByMs	再生進捗を設定します。

リアルタイム音声伴奏を利用する必要がある場合は、GME SDK をアクセスし、且つリアルタイムの音声通話を行える必要があります。

フローチャート

ソーシャルタイプAPPの呼び出しフローチャートは下図の通りです。



伴奏再生を開始する

StartAccompany インターフェースを呼び出し伴奏再生を開始します。m4a、wav、mp3 の3種類のフォーマットをサポートしています。このAPIを呼び出すと、ボリュームがリセットされます。

関数のプロトタイプ

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

パラメータ	タイプ	意味
filePath	char*	伴奏を再生するパスです。
loopBack	bool	ミキシングで発送するかどうかは、一般的にはtrueにします、即ち、ほかの人たちも伴奏が聞こえることです。

パラメータ	タイプ	意味
loopCount	int	循環回数です。数値が-1の場合、無限循環を示しています。
msTime	int	ディレー時間です。

サンプルコード

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

伴奏再生のコールバック

伴奏再生が終わった後に、コールバック関数が **OnEvent** を呼び出します。イベントメッセージが **ITMG_MAIN_EVENT_TYPE_ACCOMPANY_FINISH** であり、**OnEvent** 関数でイベントメッセージを判断します。渡されるパラメータ **data** には **result** と **file_path** の二つの情報が含まれています。

サンプルコード

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

伴奏再生を停止する

StopAccompany インターフェースを呼び出し、伴奏再生を停止します。

関数のプロトタイプ

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

パラメータ	タイプ	意味
duckerTime	int	フェードアウト時間です。

サンプルコード

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

伴奏再生が終わっているかどうか

再生が終わった場合は、戻り値がtrueです。再生が終わっていない場合は、戻り値がfalseです。

関数のプロトタイプ

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

サンプルコード

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

伴奏再生を一時停止する

PauseAccompany インターフェースを呼び出し伴奏再生を一時停止します。

関数のプロトタイプ

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

サンプルコード

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

伴奏再生を再開する

ResumeAccompany インターフェースは伴奏再生の再開に使われています。

関数のプロトタイプ

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

サンプルコード

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

自分に伴奏が聞こえるかどうかを設定する

このインターフェースは自分に伴奏が聞こえるかどうかを設定するために使われます。

関数のプロトタイプ

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

パラメータ	タイプ	意味
enable	bool	聞こえるかどうか。

サンプルコード

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

他人にも伴奏が聞こえるかどうかを設定します

他人にも伴奏が聞こえるかどうかを設定します。

関数のプロトタイプ

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

パラメータ	タイプ	意味
enable	bool	聞こえるかどうか。

サンプルコード

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

伴奏ボリュームを設定する

SetAccompanyVolume インターフェースを呼び出し伴奏ボリュームを設定します、ボリュームのデフォルト値が100です。数値が100以上の場合はゲインを上げ、数値が100以下の場合はゲインを下げます。数値範囲が0-200で

す。

関数のプロトタイプ

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

パラメータ	タイプ	意味
vol	int	ボリュームの数値です。

サンプルコード

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

伴奏再生のボリュームを取得する

GetAccompanyVolume インターフェースは伴奏ボリュームの取得に使われています。

関数のプロトタイプ

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

サンプルコード

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

伴奏再生進捗を取得する

下記の二つのインターフェースは伴奏再生進捗の取得に使われています。注意事項：**Current / Total** = 現在循環回数、**Current % Total** = 現在再生循環の位置。

関数のプロトタイプ

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

サンプルコード

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

再生進捗を設定する

SetAccompanyFileCurrentPlayedTimeByMs インターフェースは再生進捗の設定に使われています。

関数のプロトタイプ

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

パラメータ	タイプ	意味
time	int	再生進捗はミリ秒を単位とします。

サンプルコード

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

エラーコードリスト

エラーコードの名	エラーコードの値	エラーコードの意味	解決方法
QAV_ERR_ACC_OPENFILE_FAILED	4001	ファイルを開くことに失敗しました	ファイルパス及びファイルが存在しているかどうかを確認し、ファイルにアクセスする権限があるかどうかを確認します。

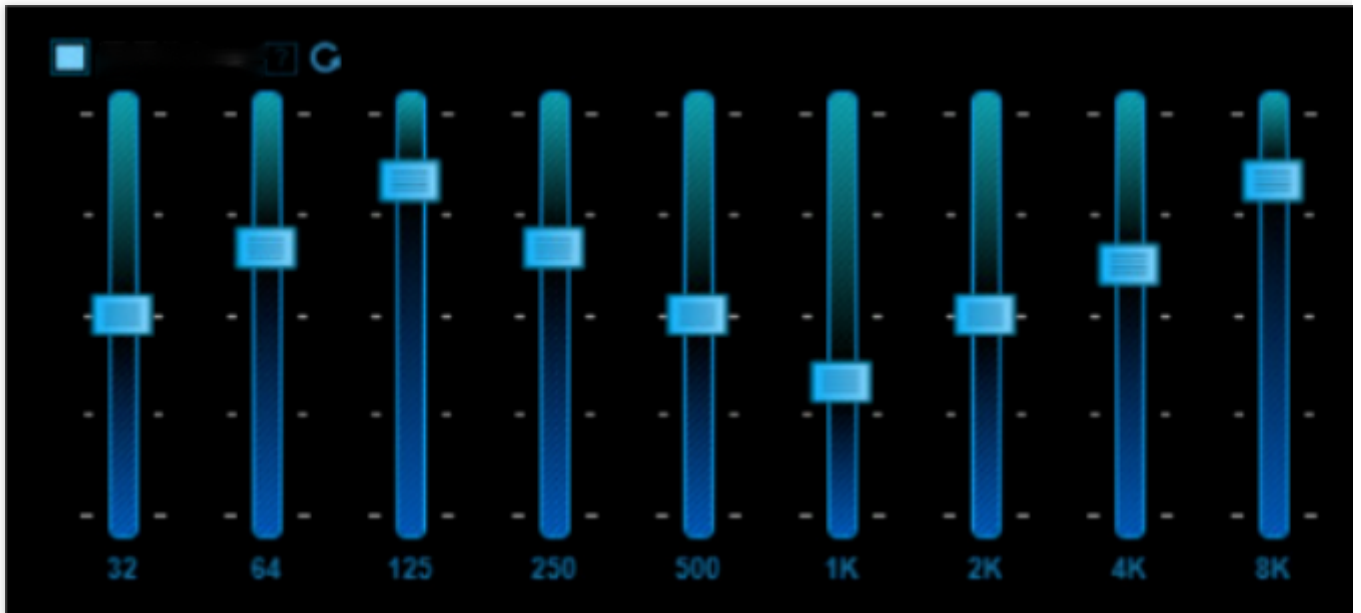
エラーコードの名	エラーコードの値	エラーコードの意味	解決方法
QAV_ERR_ACC_FILE_FORAMT_NOTSUPPORT	4002	未対応のファイル形式です	ファイル形式が正しいかどうかを確認します。
QAV_ERR_ACC_DECODER_FAILED	4003	デコードに失敗しました	ファイル形式が正しいかどうかを確認します。
QAV_ERR_ACC_BAD_PARAM	4004	パラメータエラー	コードにおけるパラメータが正しいかどうかを確認します。
QAV_ERR_ACC_MEMORY_ALLOC_FAILED	4005	メモリの割り当てに失敗しました	システムリソースが使い切られています、このエラーコードが続く場合、開発者に連絡してください。

エラーコードの名	エラーコードの値	エラーコードの意味	解決方法
QAV_ERR_ACC_CREATE_THREAD_FAILED	4006	スレッドの作成は失敗しました	システムリソースが使い切られています、このエラーコードが続く場合、開発者に連絡してください。
QAV_ERR_ACC_STATE_ILLIGAL	4007	不正な状態です	ある状態でないことです。この状態でないと呼び出せないインターフェースを呼び出した場合は、このエラーが発生します。

リアルタイムなサウンドイコライザ

最終更新日：2024-01-18 15:47:47

開発者がTencent Cloud GME製品のAPIを容易にデバッグして導入するために、このドキュメントではGMEリアルタイムなサウンドイコライザの導入を紹介します。



シナリオ

GMEイコライザ機能は、GME SDKで収集したオーディオストリームをリアルタイムでイコライザ調整することができます。この機能はオンラインのカラオケシーンに使用されます。プレイヤーが歌い始めた後、**GME SDKのイコライザ**インターフェースを呼び出して、プレイヤーのリアルタイムの音声ストリームに対し音の美化効果を調整することができます。

前提条件

リアルタイムボイスサービスを有効にしました：[サービス有効化ガイド](#)をご参照ください。

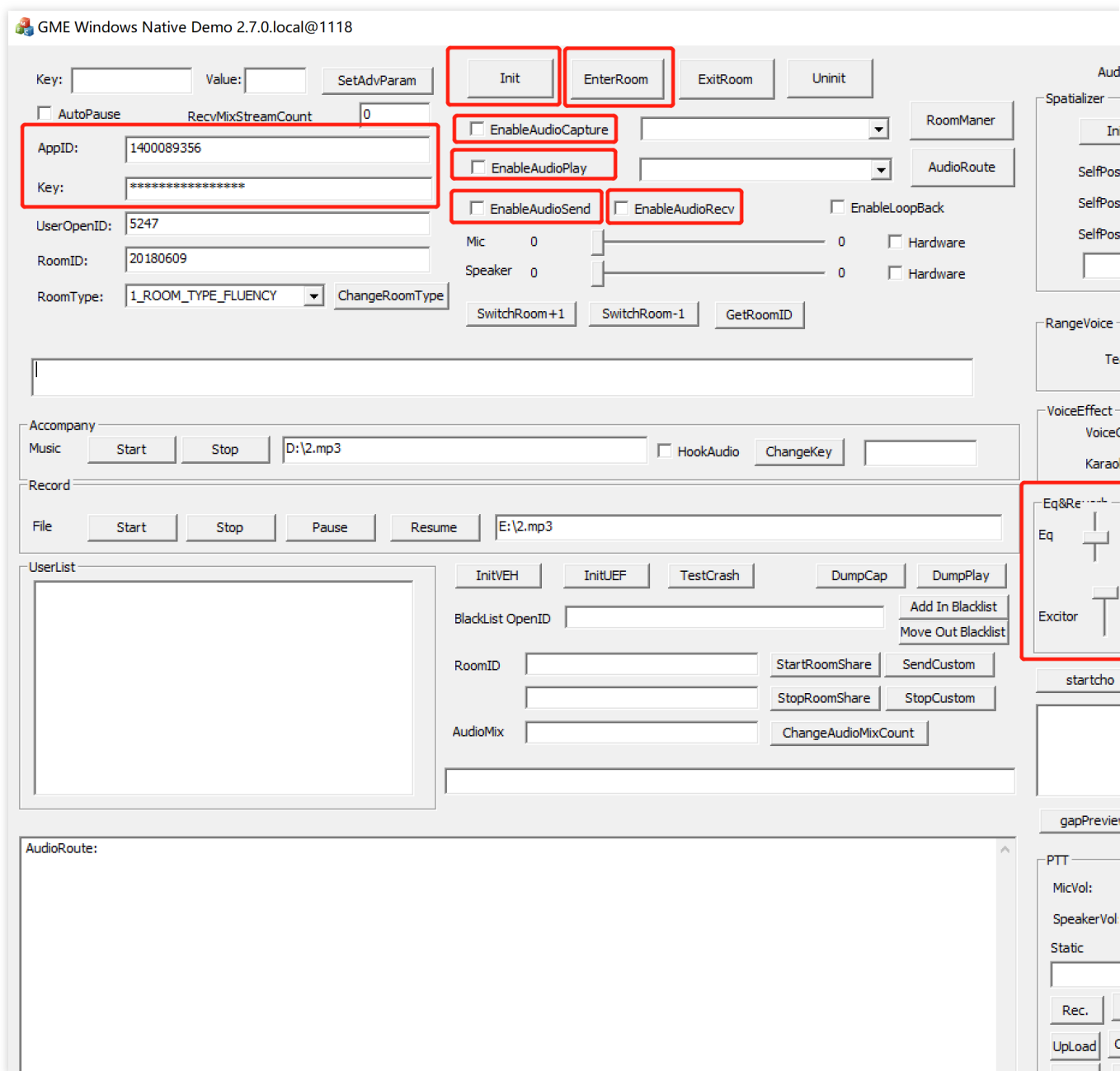
GME SDK導入済み：コアインターフェースとリアルタイム音声インターフェースの導入が含まれます。詳細については、[Native SDKのクイック導入](#)、[Unity SDKクイック導入](#)、[Unreal SDKクイック導入](#)をご参照ください。

Demo体験

Demoのダウンロード

[ダウンロードアドレス>>](#)

この体験Demoは、次のような画面を持つWindows実行可能プログラムです。



構成パラメータ

AppidとKeyの入力ボックスに自分が申請したGME AppIDとKeyを記入します。
必要に応じて、ターゲットのルーム番号とOpenIDを入力することもできます。

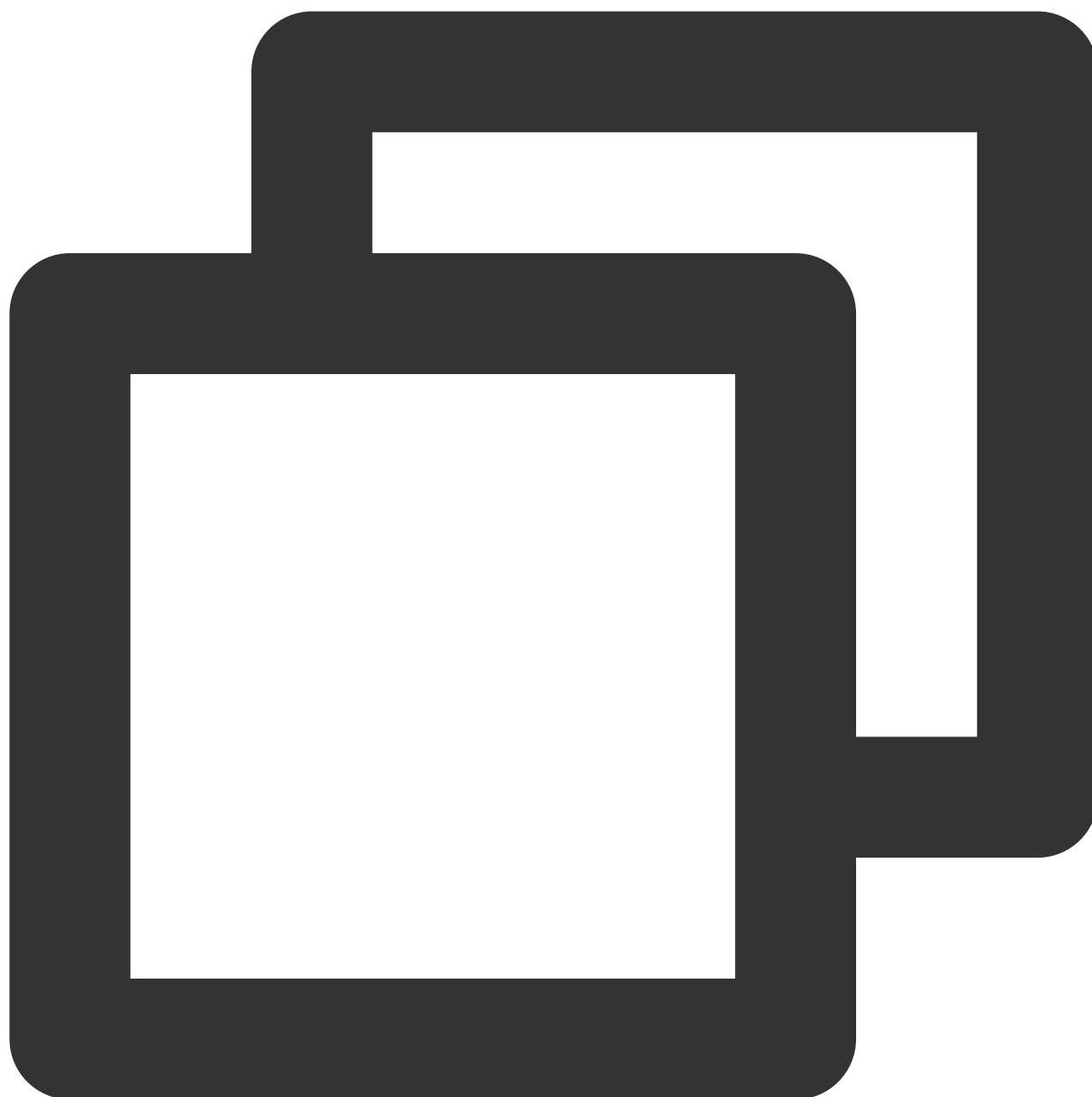
使用方法

1. ルームに入ってマイクスピーカーをオンにする手順は、Init > EnterRoom > EnableCapture > EnablePlay > EnableSend > EnableRecvです。
2. ルームに入ると、**EnableLoopBack**をオンにして自分の声を聞くことができます。
3. 赤枠の**EQ**イコライザ（EQは帯域利得、ExditorとReverbはリバーブ）を調整します。

イコライザ機能の導入

このインターフェースを使用してローカル側で収音されたサウンドのイコライザ調整を行うには、入室が成功している状態が必要です。

関数プロトタイプ



```
int SetKaraokeType(ITMG_VOICE_TYPE_EQUALIZER* pEqualizer, ITMG_VOICE_TYPE_REVERB* p
```

パラメータ	タイプ	意味
pEqualizer	ITMG_VOICE_TYPE_EQUALIZER	帯域利得
pReverb	ITMG_VOICE_TYPE_REVERB	HARMONICとREVERBを含む

構造体の詳細

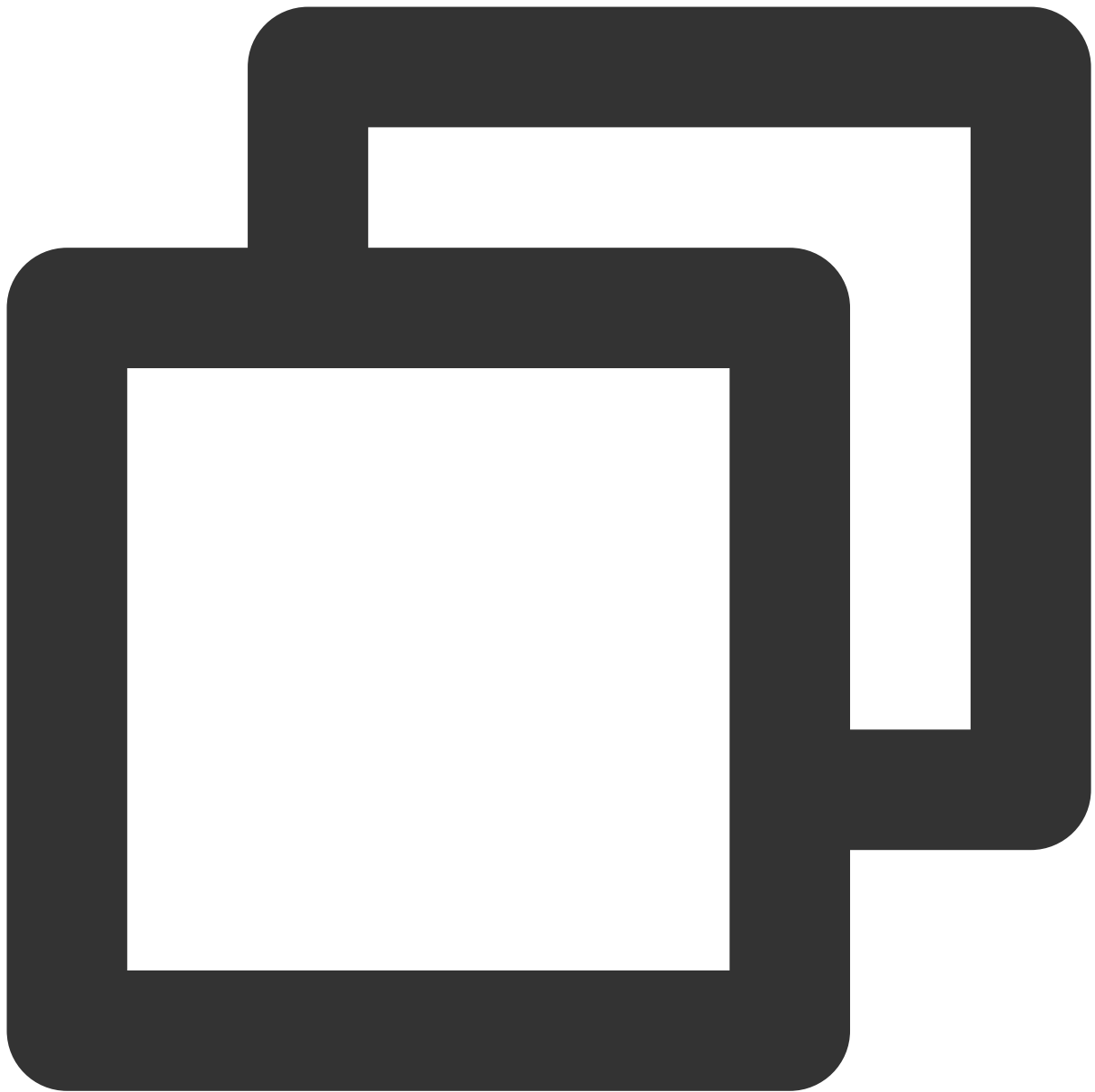
ITMG_VOICE_TYPE_EQUALIZERの構造体メンバーはfloatタイプで、数値の範囲は-12から12です。

ITMG_VOICE_TYPE_EQUALIZER	意味
EQUALIZER_32HZ	32HZ帯域に加えた利得
EQUALIZER_64HZ	64HZ帯域に加えた利得
EQUALIZER_128HZ	128HZ帯域に加えた利得
EQUALIZER_250HZ	250HZ帯域に加えた利得
EQUALIZER_500HZ	500HZ帯域に加えた利得
EQUALIZER_1KHZ	1KHZ帯域に加えた利得
EQUALIZER_2KHZ	2KHZ帯域に加えた利得
EQUALIZER_4KHZ	4KHZ帯域に加えた利得
EQUALIZER_8KHZ	8KHZ帯域に加えた利得
EQUALIZER_16KHZ	16KHZ帯域に加えた利得
EQUALIZER_MASTER_GAIN	全体的な音量

ITMG_VOICE_TYPE_REVERBの構造体メンバーはfloatタイプで、数値の範囲は0~1です。

ITMG_VOICE_TYPE_REVERB
HARMONIC_GAIN
HARMONIC_START_FREQUENCY
HARMONIC_BASS_CONTROL
REVERB_SIZE
REVERB_DEPTH
REVERB_GAIN
REVERB_ECHO_DEPTH

サンプルコード



```
void CTMGSDK_For_AudioDlg::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    if ((CWnd*)pScrollBar == (CWnd*)&m_SliderEQ1 ||
        (CWnd*)pScrollBar == (CWnd*)&m_SliderEQ2 ||
        (CWnd*)pScrollBar == (CWnd*)&m_SliderEQ3 ||
        (CWnd*)pScrollBar == (CWnd*)&m_SliderEQ4 ||
        (CWnd*)pScrollBar == (CWnd*)&m_SliderEQ5 ||
        (CWnd*)pScrollBar == (CWnd*)&m_SliderEQ6 ||
        (CWnd*)pScrollBar == (CWnd*)&m_SliderEQ7 ||
        (CWnd*)pScrollBar == (CWnd*)&m_SliderEQ8 ||
        (CWnd*)pScrollBar == (CWnd*)&m_SliderEQ9 ||
```

```

(CWnd*)pScrollBar == (CWnd*)&m_SliderEQ10 ||
(CWnd*)pScrollBar == (CWnd*)&m_SliderEQ11 ||
(CWnd*)pScrollBar == (CWnd*)&m_SliderExGain ||
(CWnd*)pScrollBar == (CWnd*)&m_SliderExStartFrequency ||
(CWnd*)pScrollBar == (CWnd*)&m_SliderExBaseCtrl ||
(CWnd*)pScrollBar == (CWnd*)&m_SliderReverbSize ||
(CWnd*)pScrollBar == (CWnd*)&m_SliderReverbDepth ||
(CWnd*)pScrollBar == (CWnd*)&m_SliderReverbGain ||
(CWnd*)pScrollBar == (CWnd*)&m_SliderReverbEchoDepth
)
{
    ITMG_VOICE_TYPE_EQUALIZER equalizer = {
        (m_SliderEQ1.GetPos() - 50) * 24.0f / 100,
        (m_SliderEQ2.GetPos() - 50) * 24.0f / 100,
        (m_SliderEQ3.GetPos() - 50) * 24.0f / 100,
        (m_SliderEQ4.GetPos() - 50) * 24.0f / 100,
        (m_SliderEQ5.GetPos() - 50) * 24.0f / 100,
        (m_SliderEQ6.GetPos() - 50) * 24.0f / 100,
        (m_SliderEQ7.GetPos() - 50) * 24.0f / 100,
        (m_SliderEQ8.GetPos() - 50) * 24.0f / 100,
        (m_SliderEQ9.GetPos() - 50) * 24.0f / 100,
        (m_SliderEQ10.GetPos() - 50) * 24.0f / 100,
        (m_SliderEQ11.GetPos() - 50) * 24.0f / 100
    };

    ITMG_VOICE_TYPE_REVERB reverb = {
        (m_SliderExGain.GetPos()) * 1.0f / 100.0f,
        (m_SliderExStartFrequency.GetPos()) * 1.0f / 100.0f,
        (m_SliderExBaseCtrl.GetPos()) * 1.0f / 100.0f,
        (m_SliderReverbSize.GetPos()) * 1.0f / 100.0f,
        (m_SliderReverbDepth.GetPos()) * 1.0f / 100.0f,
        (m_SliderReverbGain.GetPos()) * 1.0f / 100.0f,
        (m_SliderReverbEchoDepth.GetPos()) * 1.0f / 100.0f
    };

    m_pTmgContext->GetAudioEffectCtrl()->SetKaraokeType(&equalizer, &reverb);

}
CDialogEx::OnVScroll(nSBCode, nPos, pScrollBar);
}

```

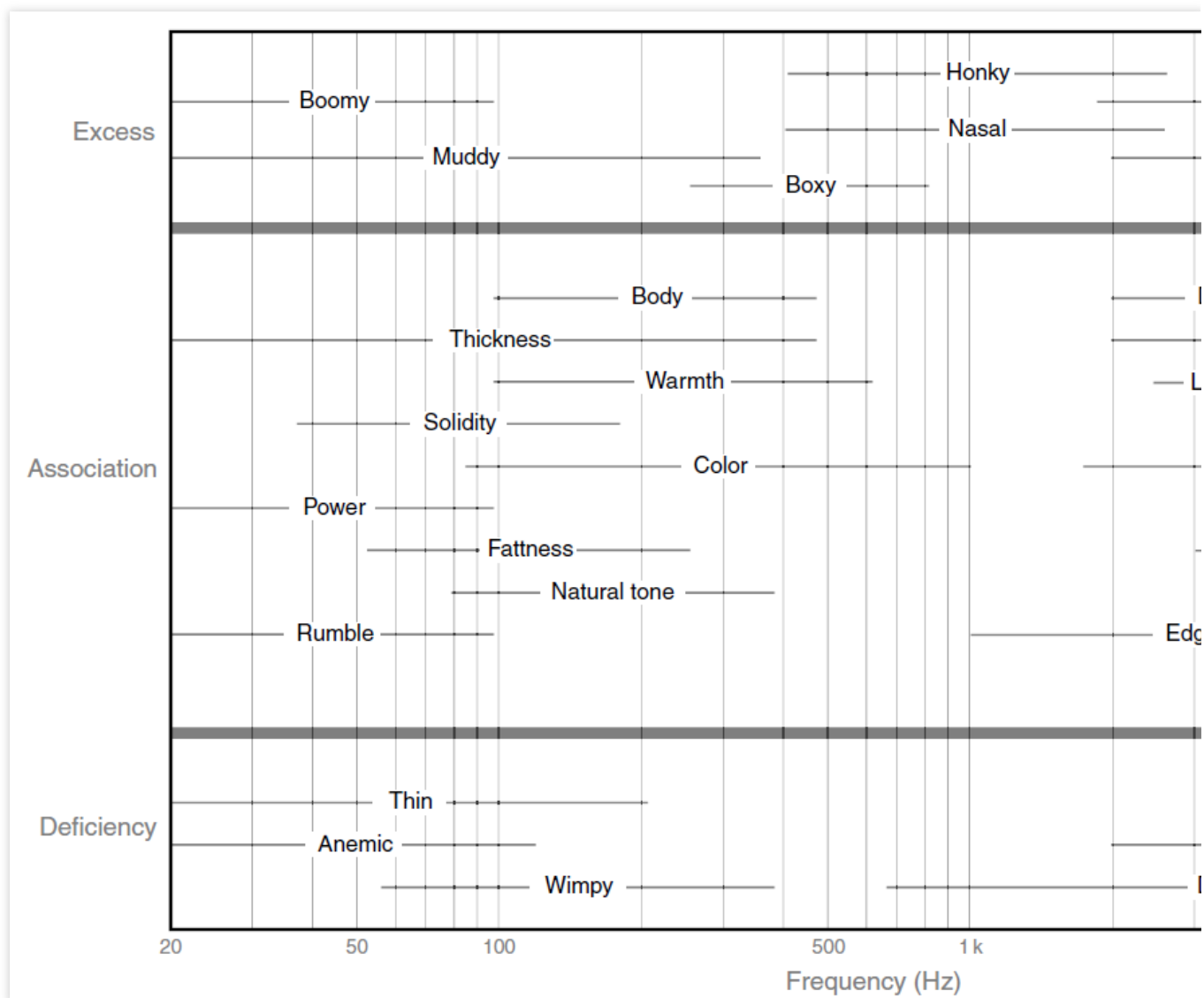
イコライザ使用ガイド

ご注意:

ここでは簡単な使い方を示しています。高級のEQ効果を使用する場合は、専門の調律師に依頼してください。人間の耳に聞こえる音の範囲はだいたい20Hz～20KHzであり、人間の耳は各周波数帯に対して対数関係にあるので、ミキシングプロジェクトでは人間の可聴周波数帯を10オクターブに分けて調整することがよくあり、調律は大きく次のように分けられます。

帯域	領域	説明
20HZ - 32HZ	サブサウンドと超低音域	大部分の周波数帯は人の耳の聴覚の下限を下回って、触覚で感知することが多い。音楽の超大型パイプオルガンと映画の中の爆発と雷の効果音はこの周波数に達することができ、人の声はこの周波数帯に達することができません。一般的なVOIP通話はzの最低に調整し、直流の干渉を除去し、信号のエネルギーを他の周波数帯に残すことをお勧めします。
32HZ - 64HZ	重低音域	主にドラムとベースのダウンを調整するために使用され、音調が目立たないように感じられ、一部のバス音域はこの周波数帯に達することができます。VoIP通話の人の声の調律は一般的に低くし、周波数干渉を除去し、エネルギーを他の周波数帯に残すことをお勧めします。
64HZ - 125HZ	低音域	ほとんどの管弦楽器の基本周波数の範囲です。打楽器の強さも決定します。
125HZ - 250HZ	低音域	人の声の基本周波数の範囲です。人の声のトーンの知覚を決定します。重すぎると音が濁ることがあります。
250HZ - 500HZ	中低音域	人の声色の重要な低次高調波がある周波数帯を含み、男性の声の音色を調整するために使用され、これを適当に強化して人の声色を温かく、重厚になり、強めすぎて音が濁ります。
500HZ - 1KHZ	中音域	女声の音色を調整し、それをより豊かにします。高すぎると鼻音が重くなります。携帯電話などのモバイル再生機器のフォルマントはこの周波数帯にあり、調整が高すぎて構造振動雑音が発生しやすいことを避けることをお勧めします。
1KHZ - 2KHZ	中音域	人の耳の敏感な領域で、響きと臨場感に影響し、適切に増強することができます。
2KHZ - 4KHZ	中高音域	人の耳が最も敏感な領域で、この判断を高めることで音の響きを高め、音声の理解度の相関を高めることができ、調整が高すぎると歯音が重すぎるようになります。
4KHZ - 8KHZ	高音域	チャイナシンバルなどの高周波楽器や弦楽器の摩擦音などの音の細かい部分を表現します。唇歯音、摩擦音など、人の声の高周波数のディテールを決定します。通常増強しないでください。
8KHZ - 16KHZ	超高音や超音波領域	楽器の高周波オーバートーンです。調整が音声にあまり影響しません。

調整の詳細については、次の図をご参照ください。



リアルタイムカラオケ機能

最終更新日：：2024-01-18 15:47:47

開発者がTencent Cloud Gaming Multimedia Engine製品APIのデバッグ・アクセスを行いやすいように、ここで、Gaming Multimedia Engineリアルタイム音声カラオケ機能の導入技術ドキュメントを説明します。

使用前提

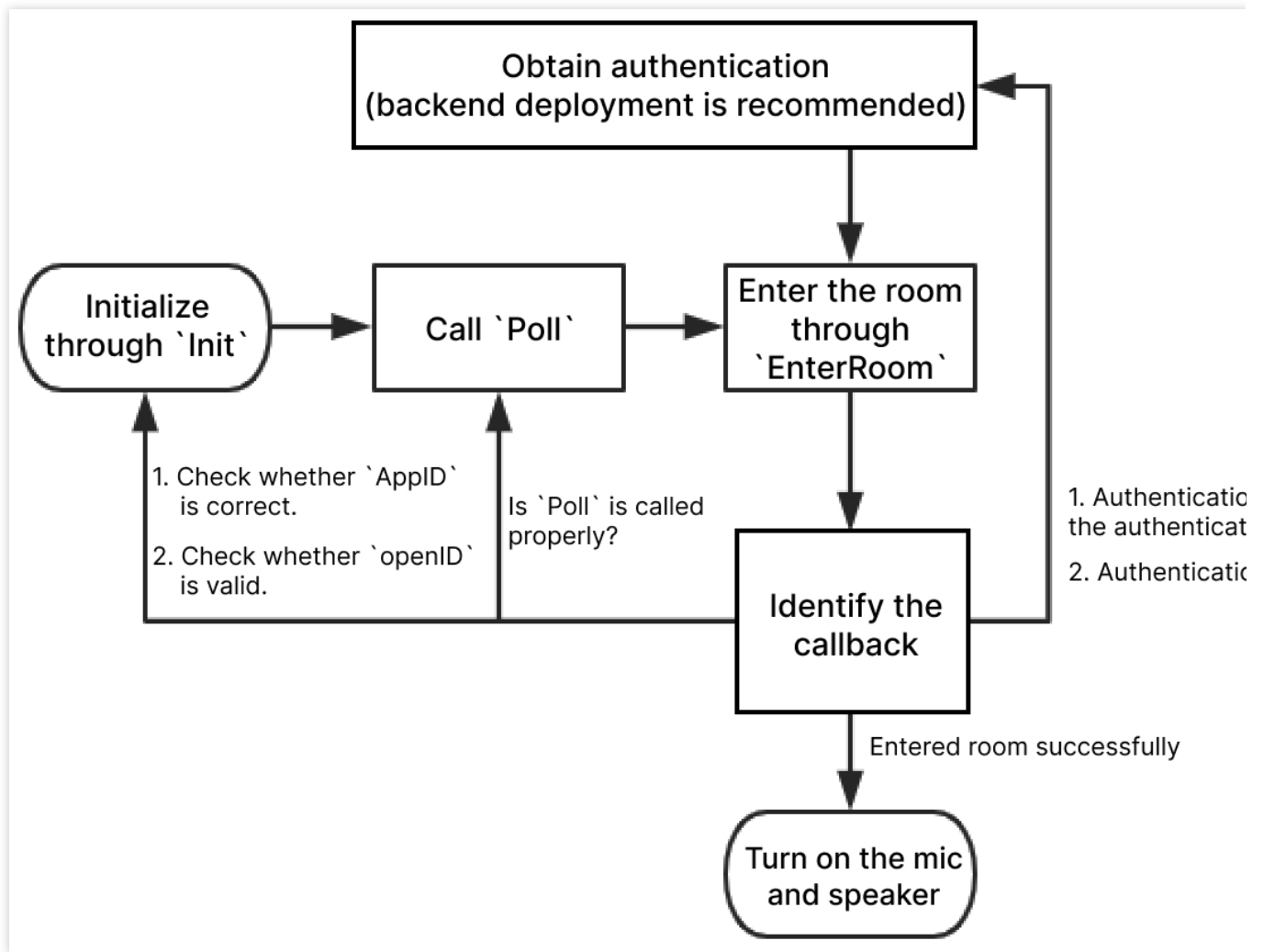
リアルタイム音声カラオケ機能を利用する必要がある場合は、GME SDKへアクセスし、且つリアルタイムの音声通話を行える必要があります。

入室時にルームタイプのパラメータを入力します。RoomType=2でルームに参加することをお勧めします（2または3を使用してください）。

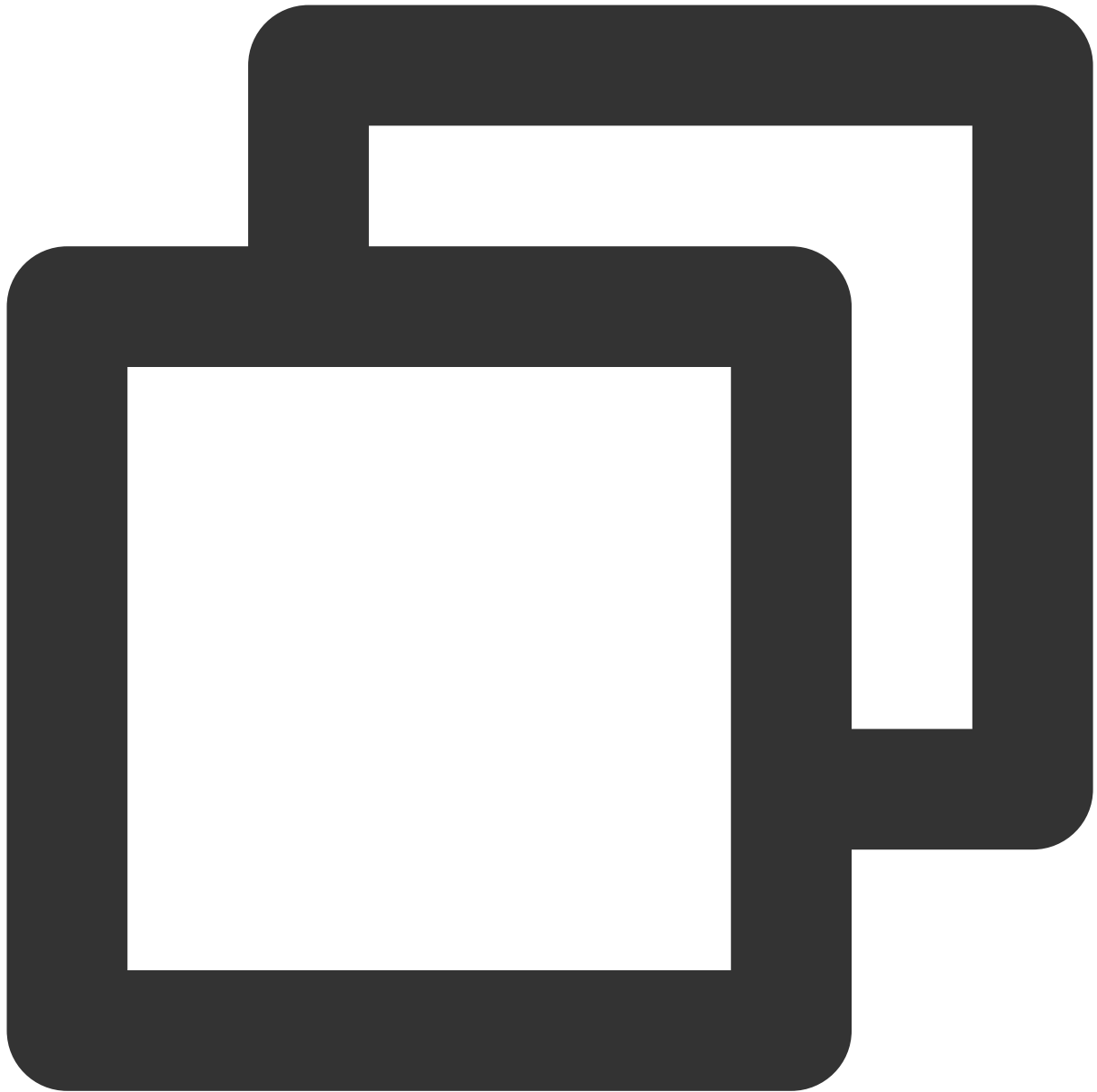
使用中にエラーコードが表示された場合、[エラーコードドキュメント](#)を参照して解決してください。

フローチャート

入室手順の参考図は以下のとおりです：



リアルタイムボイスルーム参加インターフェース：



```
ITMGContext.GetInstance(this).Init(String.valueOf(mAppId), mUserId); // sdkの初期化
ITMGContext.GetInstance(this).SetTMGDelegate(new MyDelegate()); // さまざまなコールバック
EnginePollHelper.createEnginePollHelper(); // 定期的にPoll関数が呼び出され、コールバックがト

byte[] authbuff = AuthBuffer.GetInstance().genAuthBuffer(mAppId, mRoomId, mUserId, m
ITMGContext.GetInstance(this).EnterRoom(mRoomId, 2, authbuff); // 入室
```

説明：

呼び出し手順とインターフェースの詳細については[各端末SDKインターフェースドキュメント](#)をご参照ください。

カラオケインターフェースの取得

1. [ダウンロードガイド](#)から標準SDKファイルをダウンロードします。
2. プラットフォームによっては、[カラオケ機能インターフェース](#)をダウンロードし、適切なインターフェースファイルをインポートします。

説明：

この機能は、mp3とoggの両方の形式をサポートしています。

音楽ファイルがogg形式の場合は、[クリックしてoggダイナミックライブラリをダウンロード](#)してください（iOS側にはoggダイナミックライブラリが含まれているのでインポートは不要）。

音楽ファイルがmp3形式の場合は、[クリックしてmp3ダイナミックライブラリをダウンロード](#)してください（iOS以外のプラットフォームにはこのダイナミックライブラリをインポートする必要があります）。

Android端末の設定

Android対応のインターフェースはすでに標準jarパッケージに含まれており、別途でインターフェースファイルをダウンロードする必要はありません。

iOS端末の設定

1. iOS側でカラオケ機能を使用するには、関連するダイナミックライブラリをプロジェクトに導入する必要があります。 [クリックしてmp3ダイナミックライブラリをダウンロード](#)してください。
2. ダウンロードしたファイルをプロジェクトファイルに取り込みます。このダイナミックライブラリをLink Binary With Librariesに追加します。
3. ヘッダーファイルTMGEngine_adv.hを、他のSDKヘッダーファイルと同じディレクトリにあるプロジェクトに追加します。

Windows端末の設定

Windows側ではカラオケ機能を使用するには、ヘッダーファイルをダウンロードした後、ヘッダーファイルtmg_sdk_adv.h、tmg_type_adv.hをプロジェクトにインポートする必要があります。

Unityエンジンの設定

Unityエンジンでカラオケ機能を使用するには、ヘッダーファイルをダウンロードし、Unityフォルダ下のコードファイルTMGEngine_Adv.cs、ITMGEngine_Adv.csをコピーしてプロジェクトにインポートする必要があります。iOSプラットフォームからエクスポートする場合は、上記を参照してmp3ダイナミックライブラリをインポートしてください。

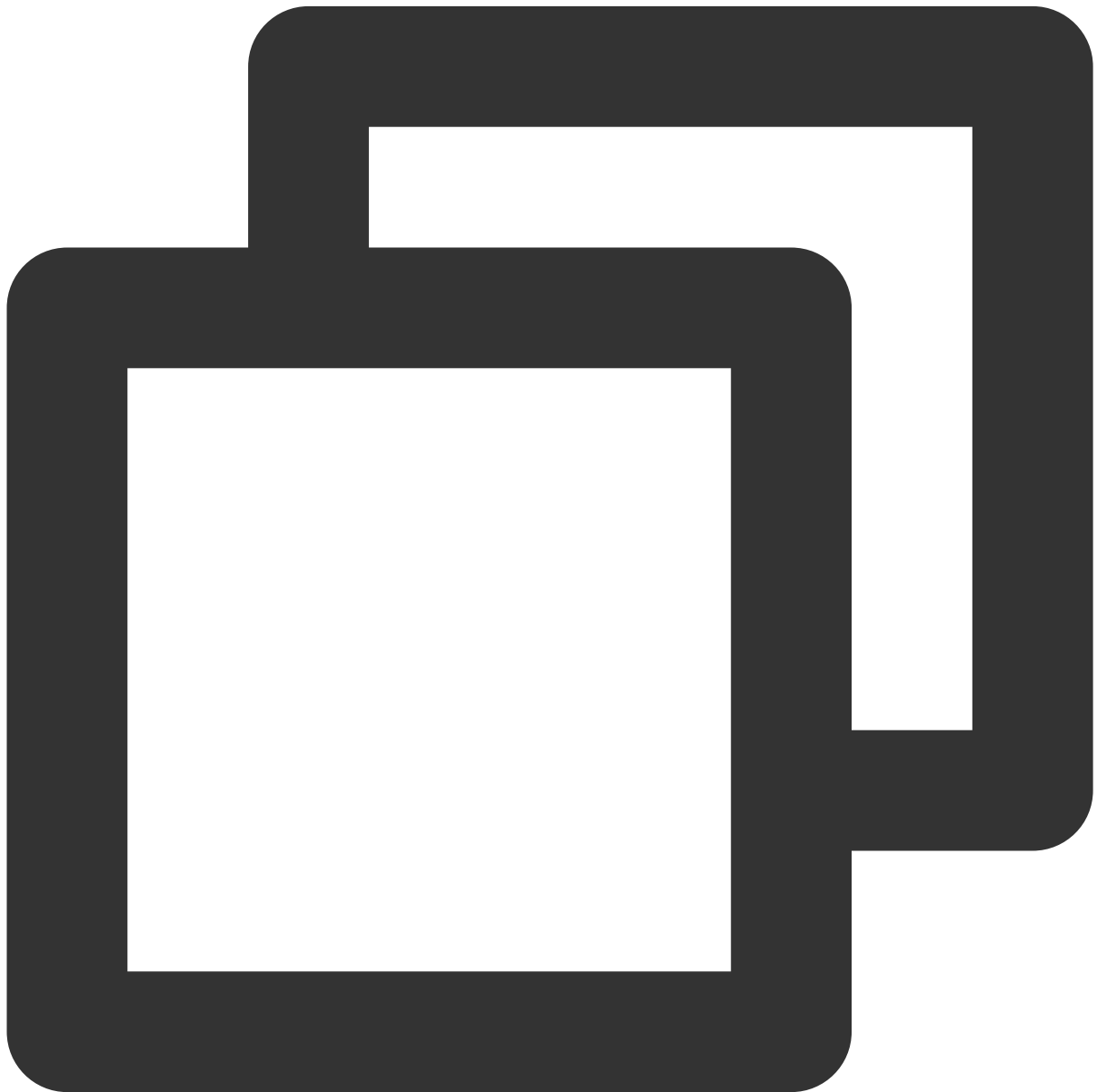
レコーディング関連インターフェース

レコーディング開始

StartRecordインターフェースを呼び出してレコーディングを開始します。レコーディングが完了すると、コールバック関数が呼び出され、ITMG_MAIN_EVENT_TYPE_RECORD_COMPLETEDをリスニングする必要があります。

レコーディングの際にはマイクがオンになっていること（デバイスと上りの両方がオンになっている必要があります）、ファイルパスがアクセス可能であることを確認してください。SDKは自動でフォルダを作成できません。

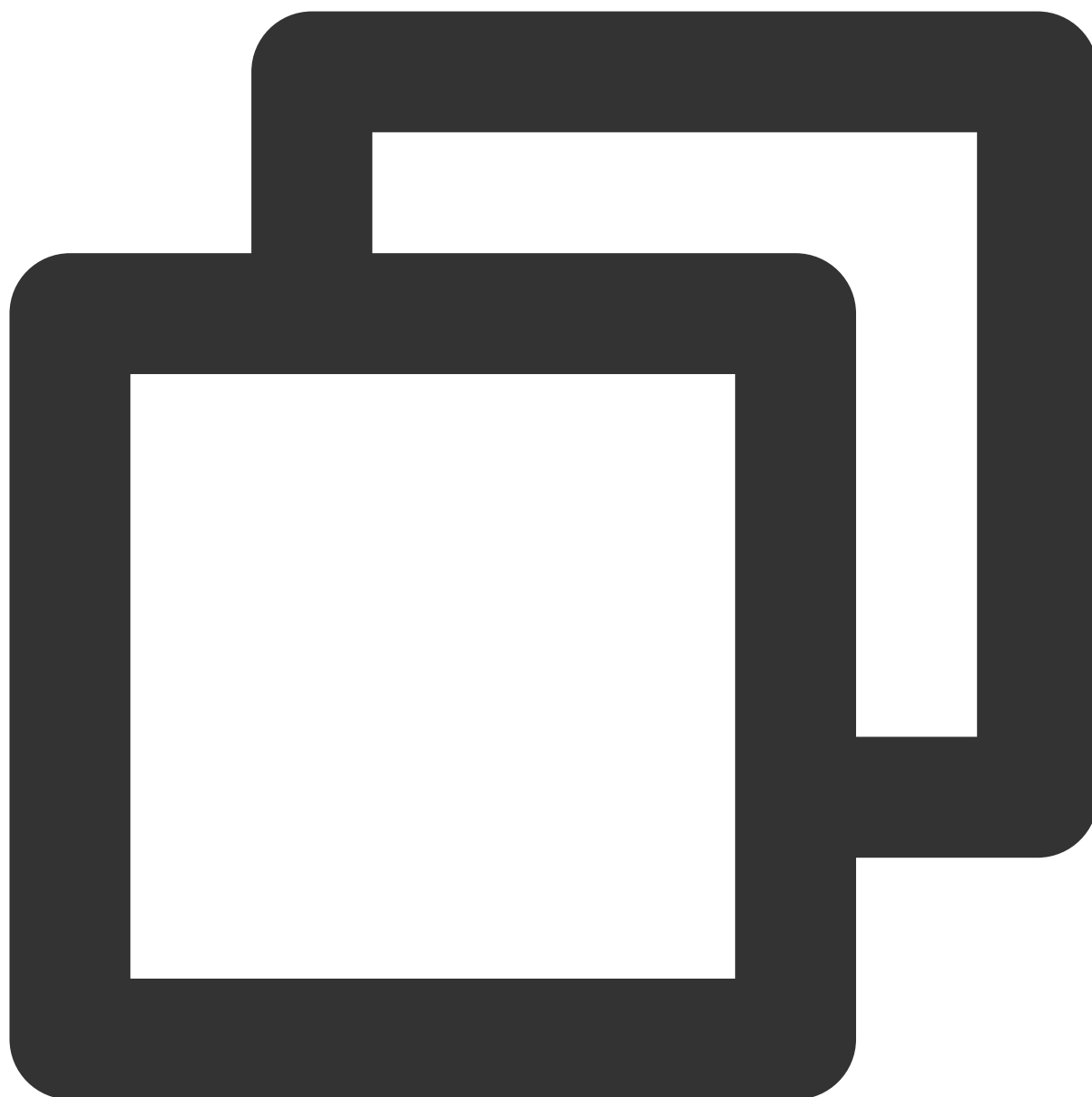
関数のプロトタイプ



```
int StartRecord(int type, String dstFile, String accMixFile, String accPlayFile)
```

パラメータ	タイプ	意味
type	int	カラオケシナリオで、このパラメータはITMG_AUDIO_RECORDING_KTVに渡されます。単にMP3ファイルをレコーディングする場合は、ITMG_AUDIO_RECORDING_SELFを使用してください。
dstFile	String	レコーディングされた音楽を保存するための対象ファイルのパス。
accMixFile	String	人間の声と音楽ファイルを合成するために用いられる通常の歌なしの伴奏。
accPlayFile	String	再生に使用する音楽ファイルであり、通常はaccMixFileと同じファイルです。しかし、ユーザーが曲に慣れていない場合には、歌付きの音楽ファイルのパスを埋め込みます。この場合の再生内容は歌付きの音楽であり、歌なしの伴奏を合成します。

サンプルコード

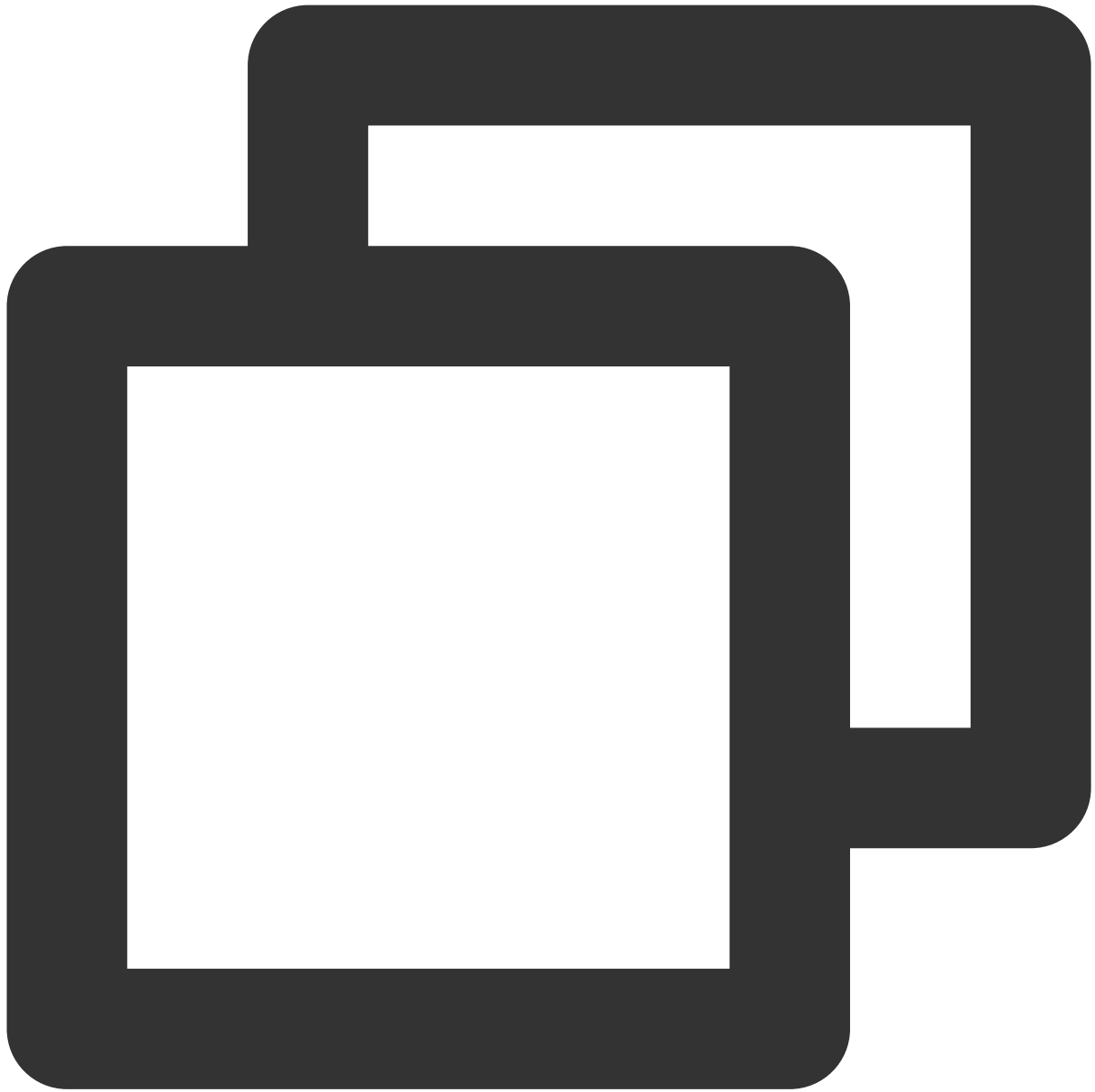


```
//Android
ITMGAudioRecordCtrl.GetInstance().StartRecord(ITMGAudioRecordCtrl.ITMG_AUDIO_RECORD
//iOS
#import "GMESDK/TMGEngine_adv.h"
[[ITMGAudioRecordCtrl GetInstance]StartPreview]
```

レコーディング停止

StopRecordインターフェースを呼び出して記録を停止します。

関数のプロトタイプ

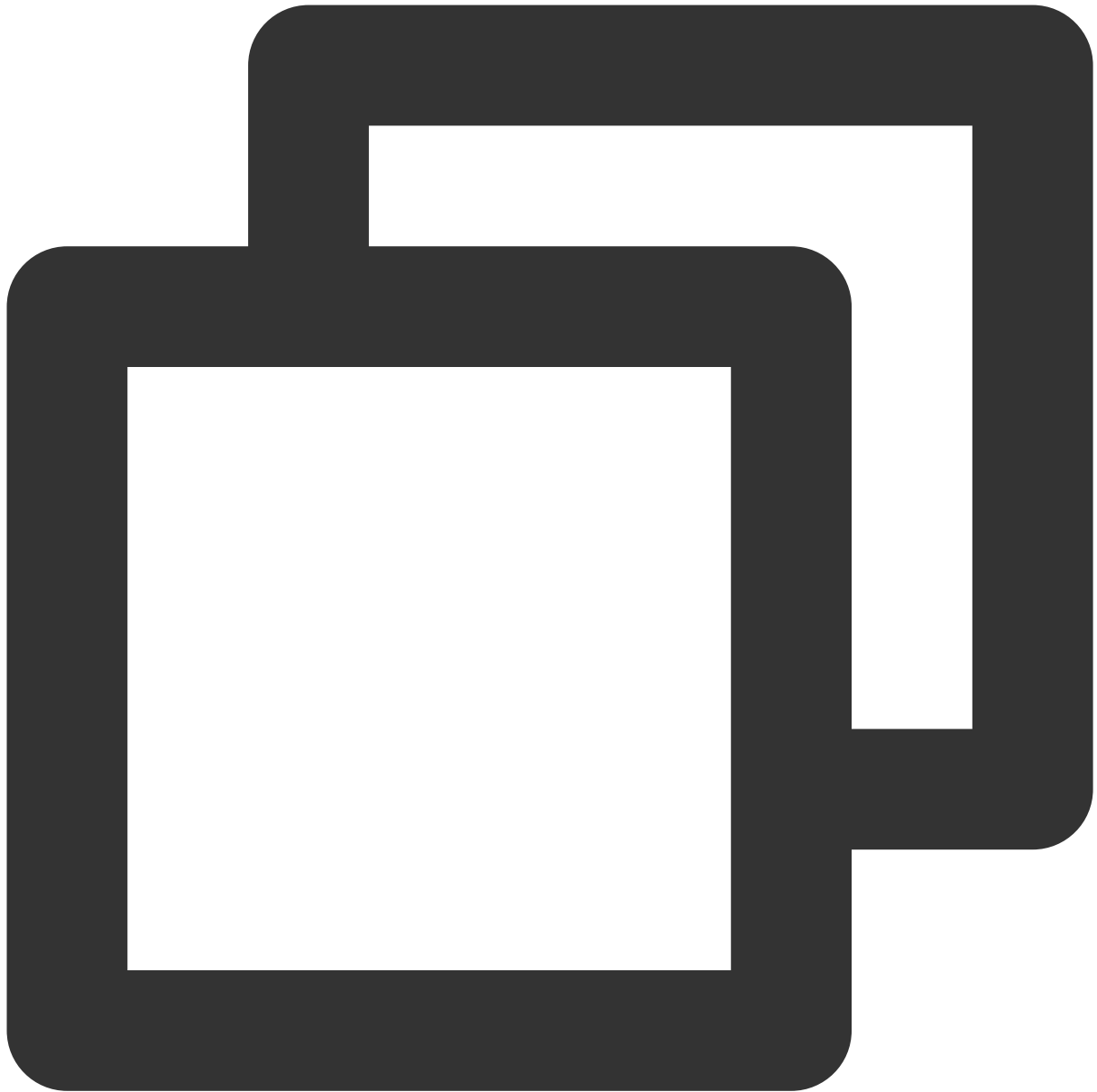


```
int StopRecord()
```

レコーディングの一時停止

PauseRecordインターフェースを呼び出して記録を一時停止します。

関数のプロトタイプ

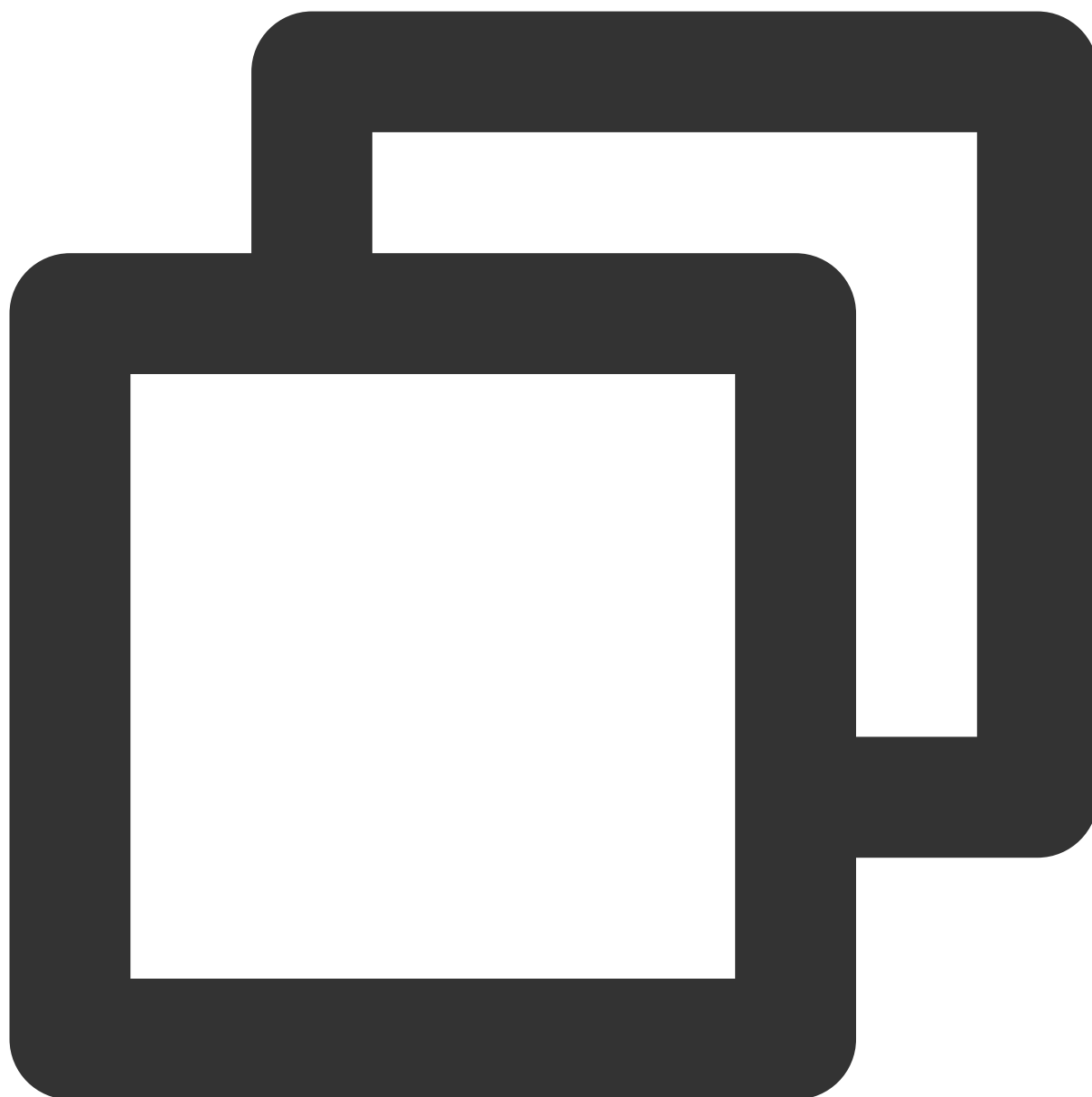


```
int PauseRecord()
```

レコーディングの再開

記録を再開するには、ResumeRecordインターフェースを呼び出します。

関数のプロトタイプ



```
int ResumeRecord()
```

コールバックのレコーディング

ITMG_MAIN_EVENT_TYPE_RECORD_COMPLETEDレコーディングが完了したときのコールバック。このコールバックは伴奏の再生が終了するか、StopRecordが呼び出されるとトリガーされます。

コールバックパラメータ

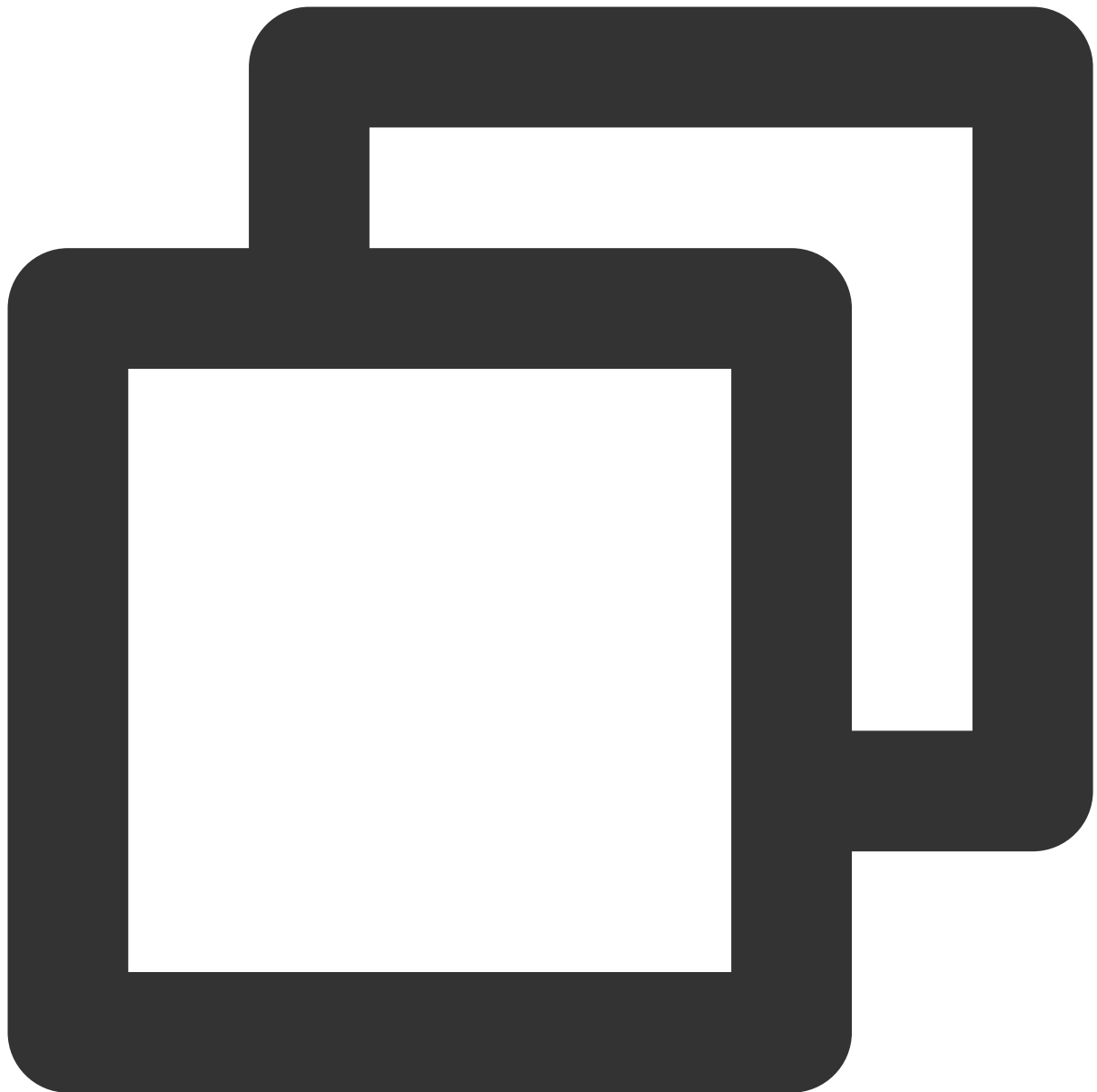
パラメータ	タイプ	意味
-------	-----	----

result	int	記録結果です。0は成功を意味します。他にエラーコードがある場合は、 エラーコードドキュメント を参照して解決してください。
filepath	String	対象ファイルへのパスです。StartRecordから渡される引数dstFile。
duration	String	記録ファイルの長さ（ミリ秒単位）です。

再生ファイルの設定

StartRecordインターフェースを呼び出してレコーディングする際に、再生する音楽ファイルを設定します。再設定したい場合は、このインターフェースを呼び出して再生ファイルを再設定することができます。通常、歌と伴奏を切り替えるのに使われます。

関数のプロトタイプ



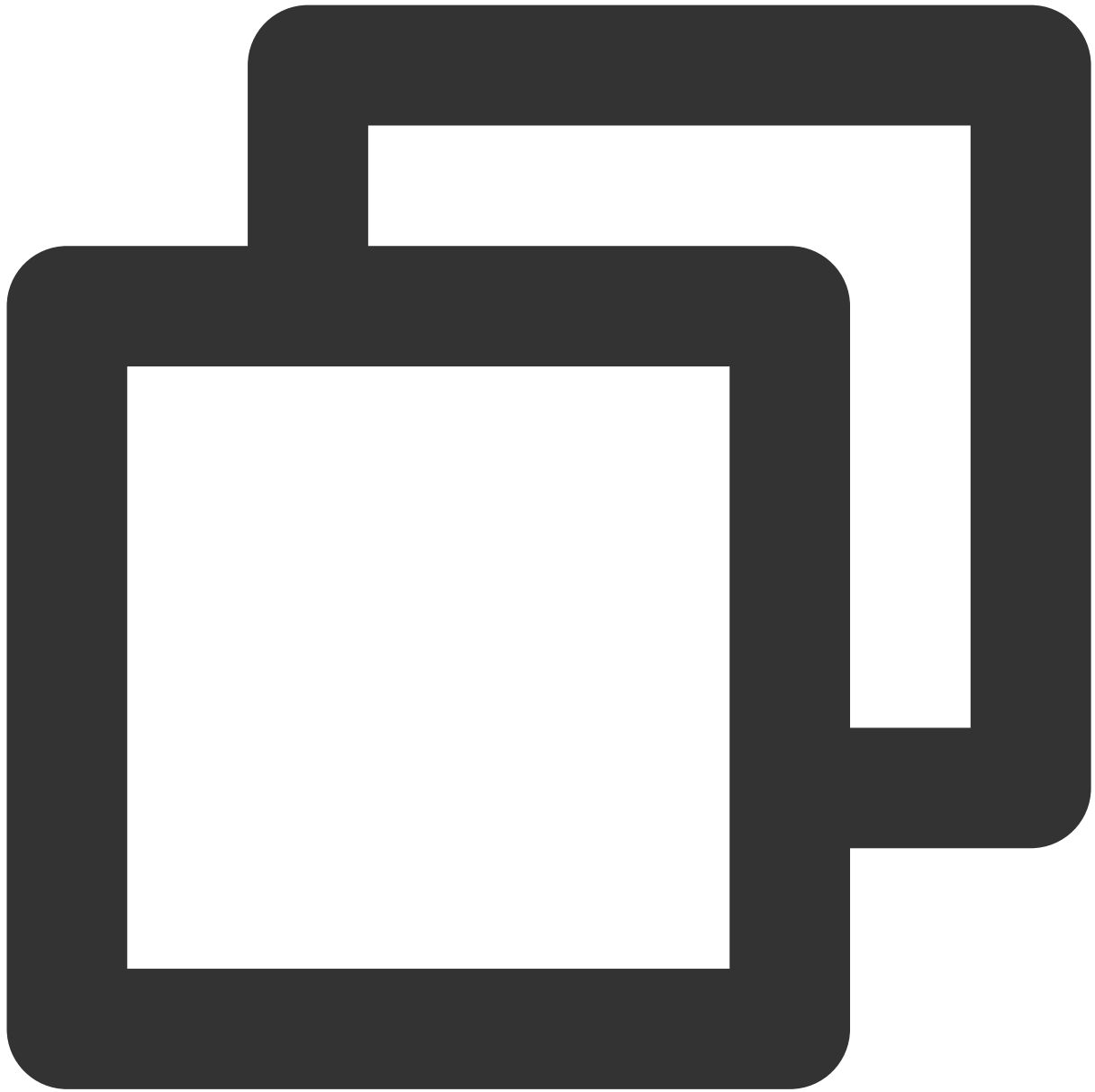
```
int SetAccompanyFile(String accPlayFile)
```

パラメータ	タイプ	意味
accPlayFile	String	再生に使用される音楽ファイル。

伴奏の長さを取得

このパラメータを呼び出し、伴奏ファイルaccMixFileの長さを取得します。ミリ秒単位で返されます。

関数のプロトタイプ

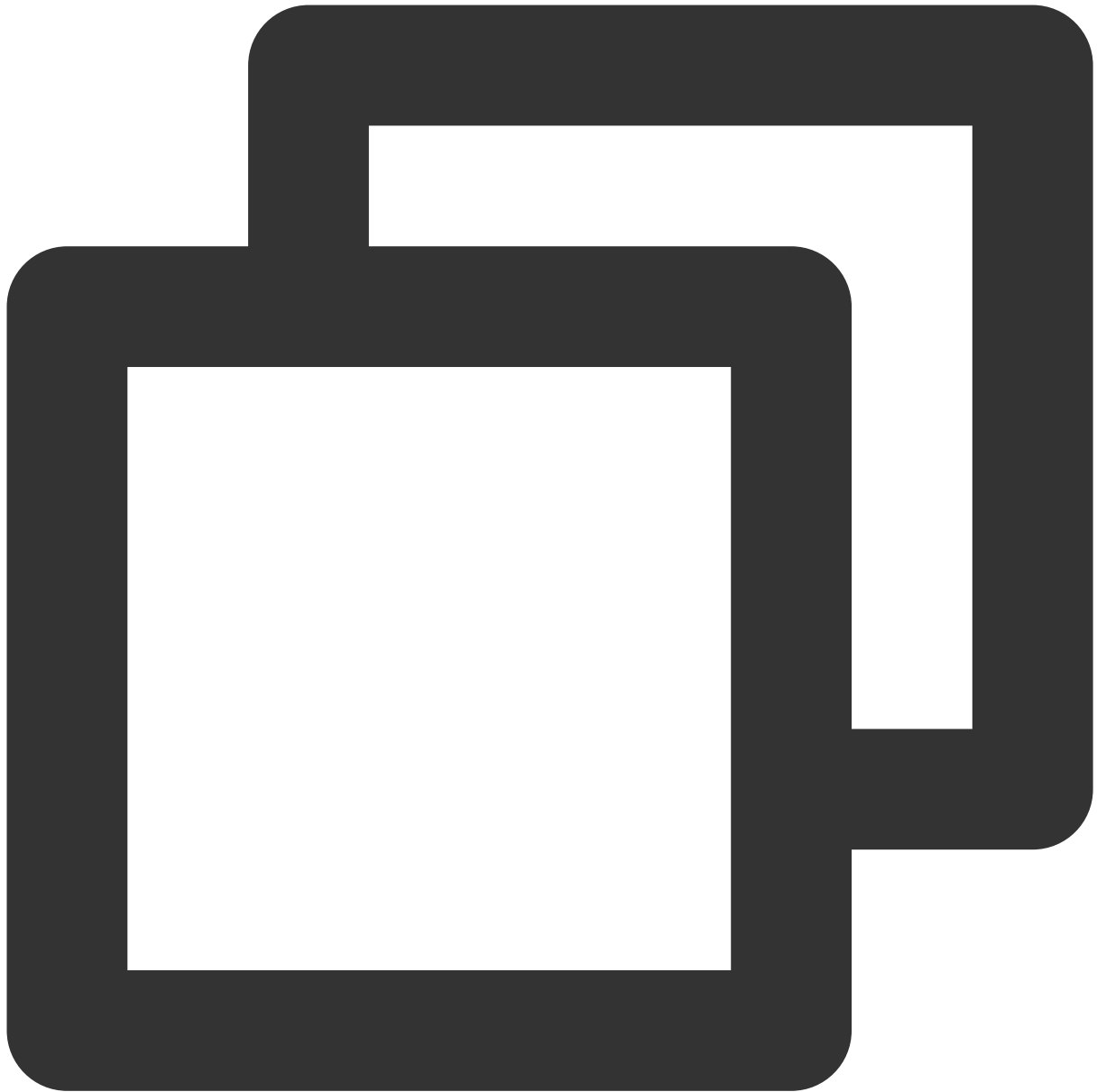


```
int GetAccompanyTotalTimeByMs()
```

現在の記録時間を取得

このパラメータを呼び出し、現在の記録時間をミリ秒単位で取得します。

関数のプロトタイプ

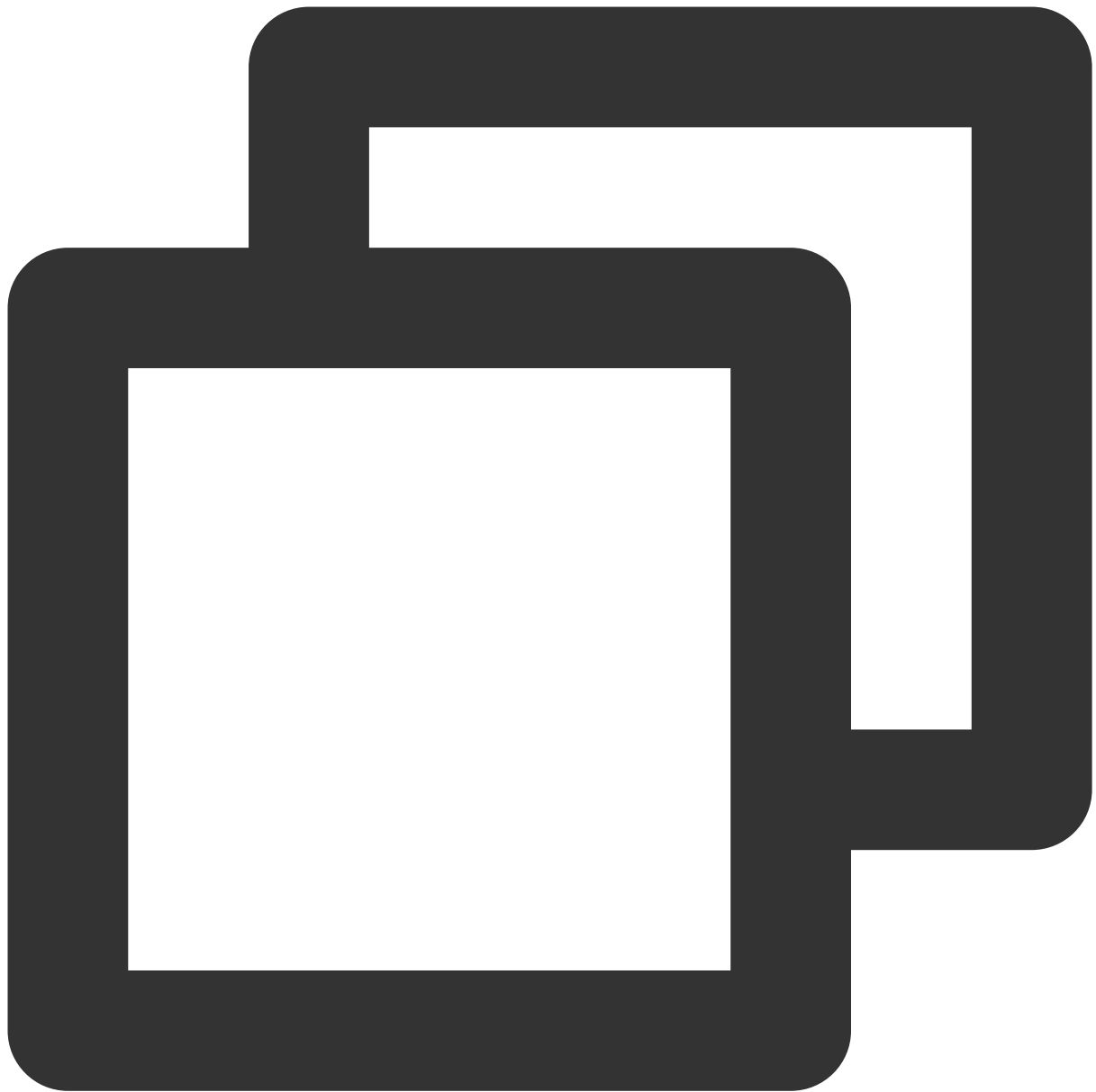


```
int GetRecordTimeByMs ()
```

レコーディングのジャンプ

記録時間を指定した時刻にジャンプします。パラメータが現在の時刻よりも前にある場合、繰り返した箇所は再記録されます。現在時刻よりも後であれば、記録されていない部分をミュートデータで埋めます。

関数のプロトタイプ



```
int SetRecordTimeByMs(int timeMs)
```

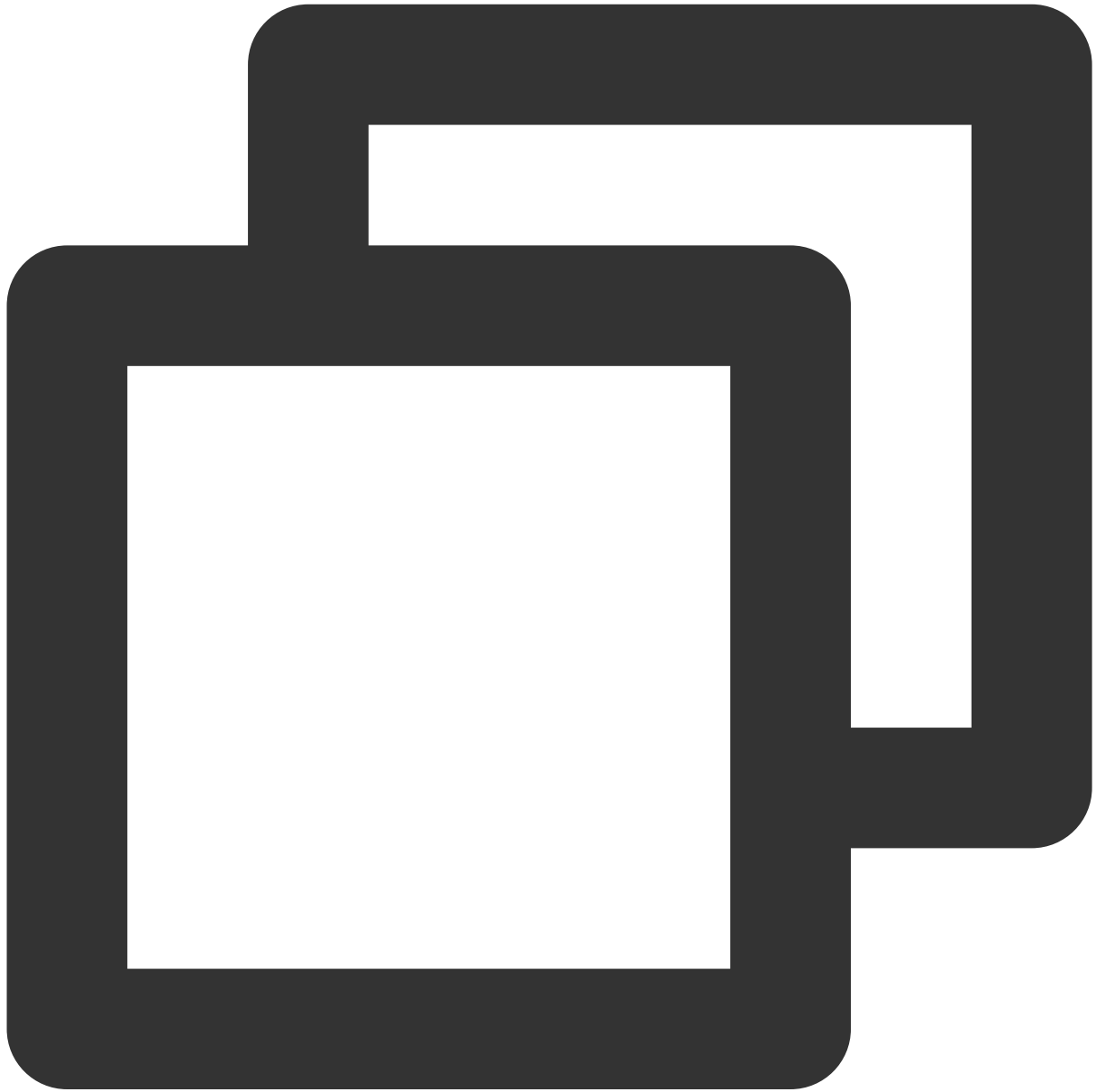
パラメータ	タイプ	意味
timeMs	int	ジャンプする時刻（ミリ秒単位）。

カラオケファイルのプレビュー

レコーディングファイルの長さの取得

このパラメータを呼び出して、記録ファイルの長さを取得します。

関数のプロトタイプ

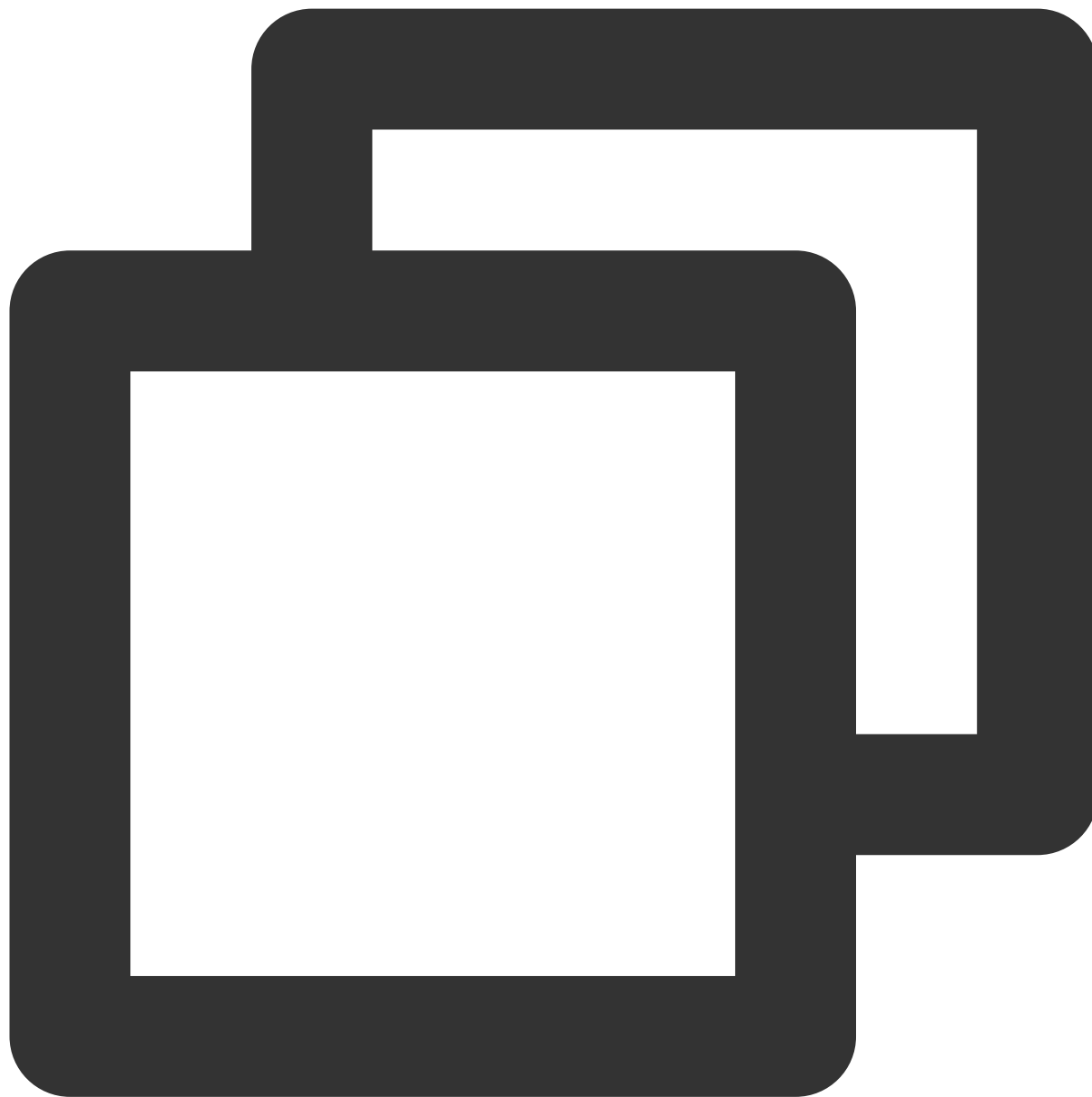


```
int GetRecordFileDurationByMs()
```

記録ファイルのプレビューを開始

このパラメータを呼び出して、記録ファイルのプレビューを開始します。

関数のプロトタイプ

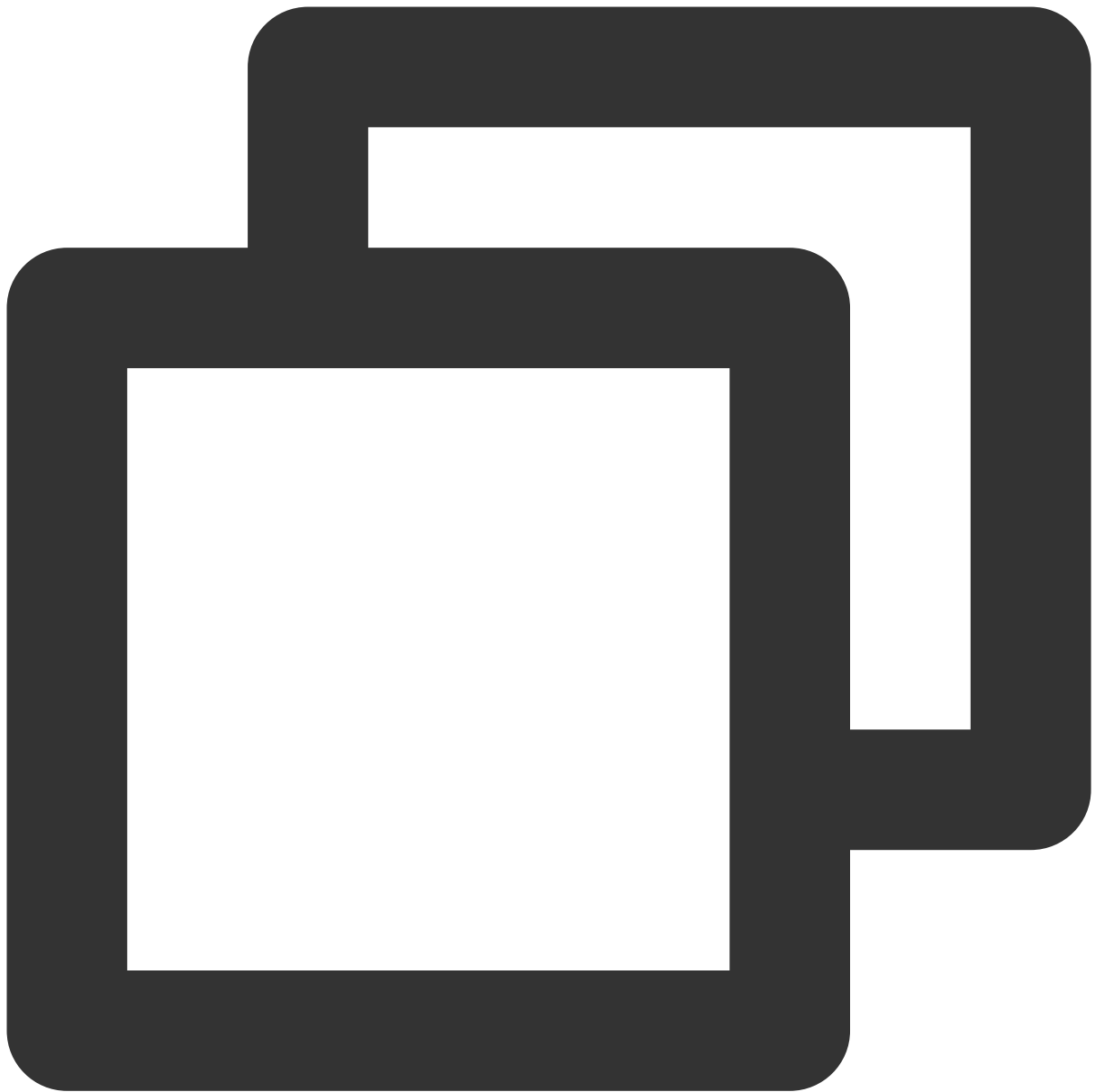


```
int StartPreview()
```

記録ファイルのプレビューを停止

このパラメータを呼び出して、記録ファイルのプレビューを停止します。

関数のプロトタイプ

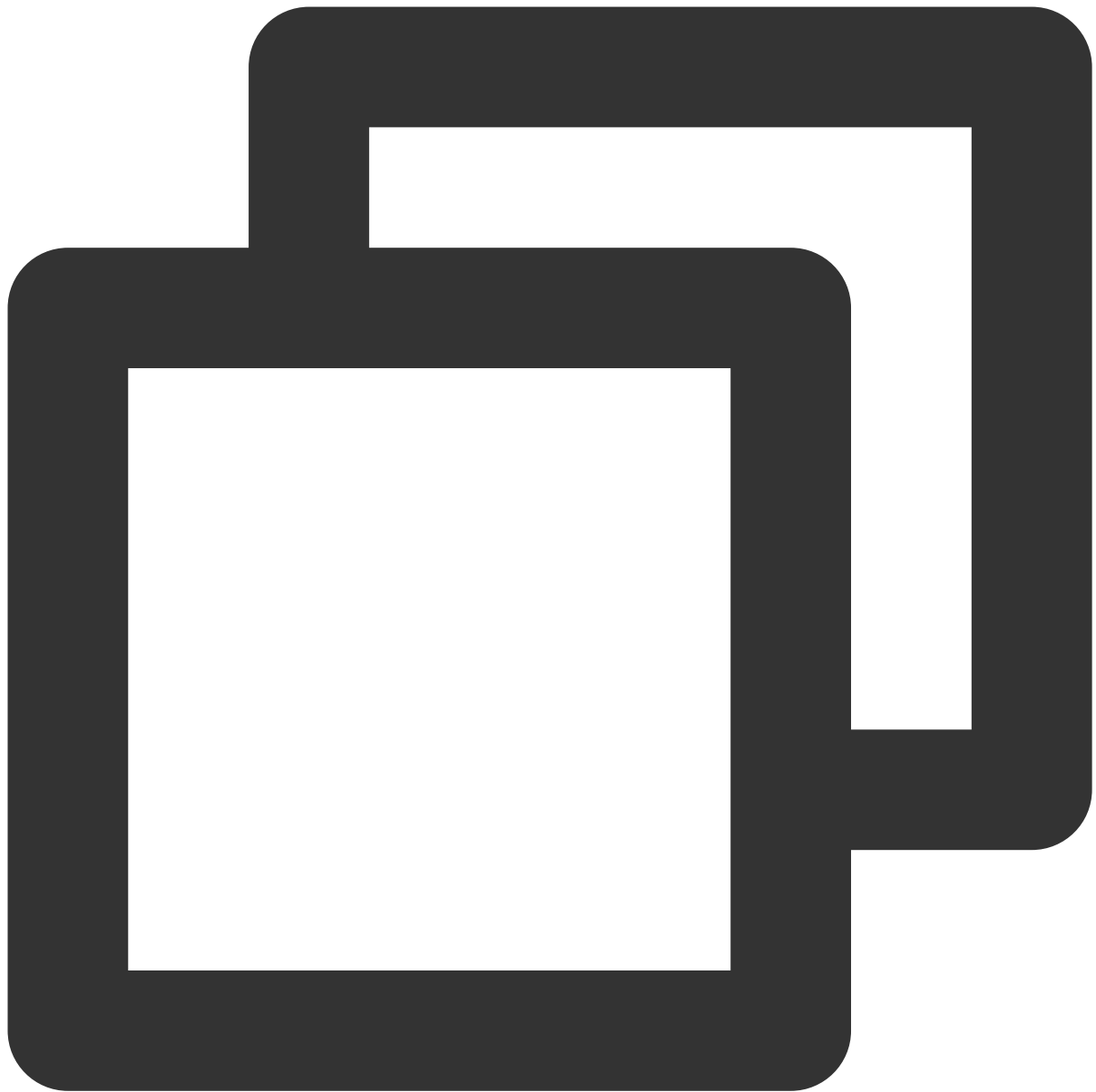


```
int StopPreview()
```

記録ファイルのプレビューを一時停止

このパラメータを呼び出して、記録ファイルのプレビューを一時停止します。

関数のプロトタイプ

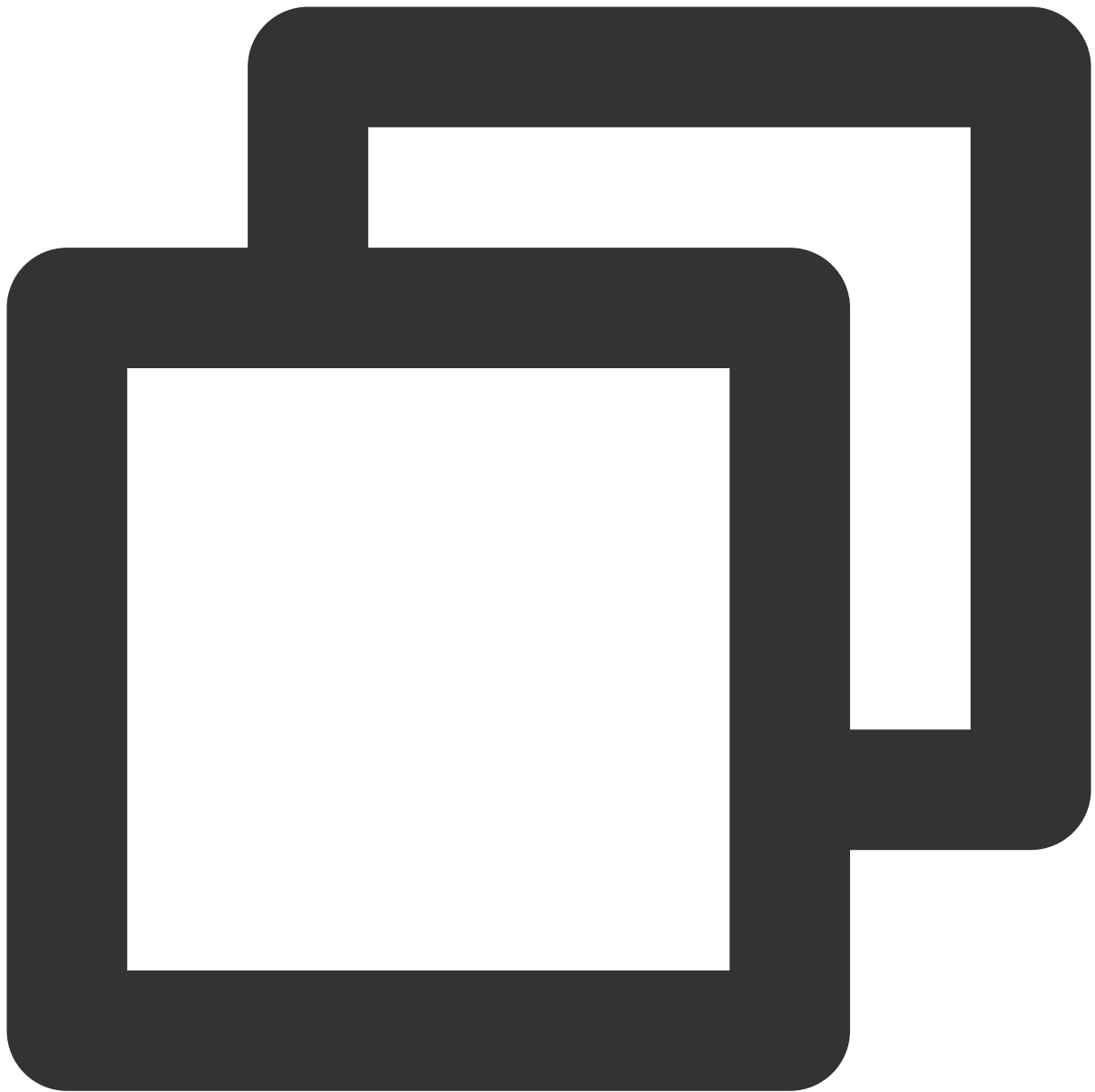


```
int PausePreview()
```

記録ファイルのプレビューを復元

このパラメータを呼び出して、記録ファイルのプレビューを復元します。

関数のプロトタイプ

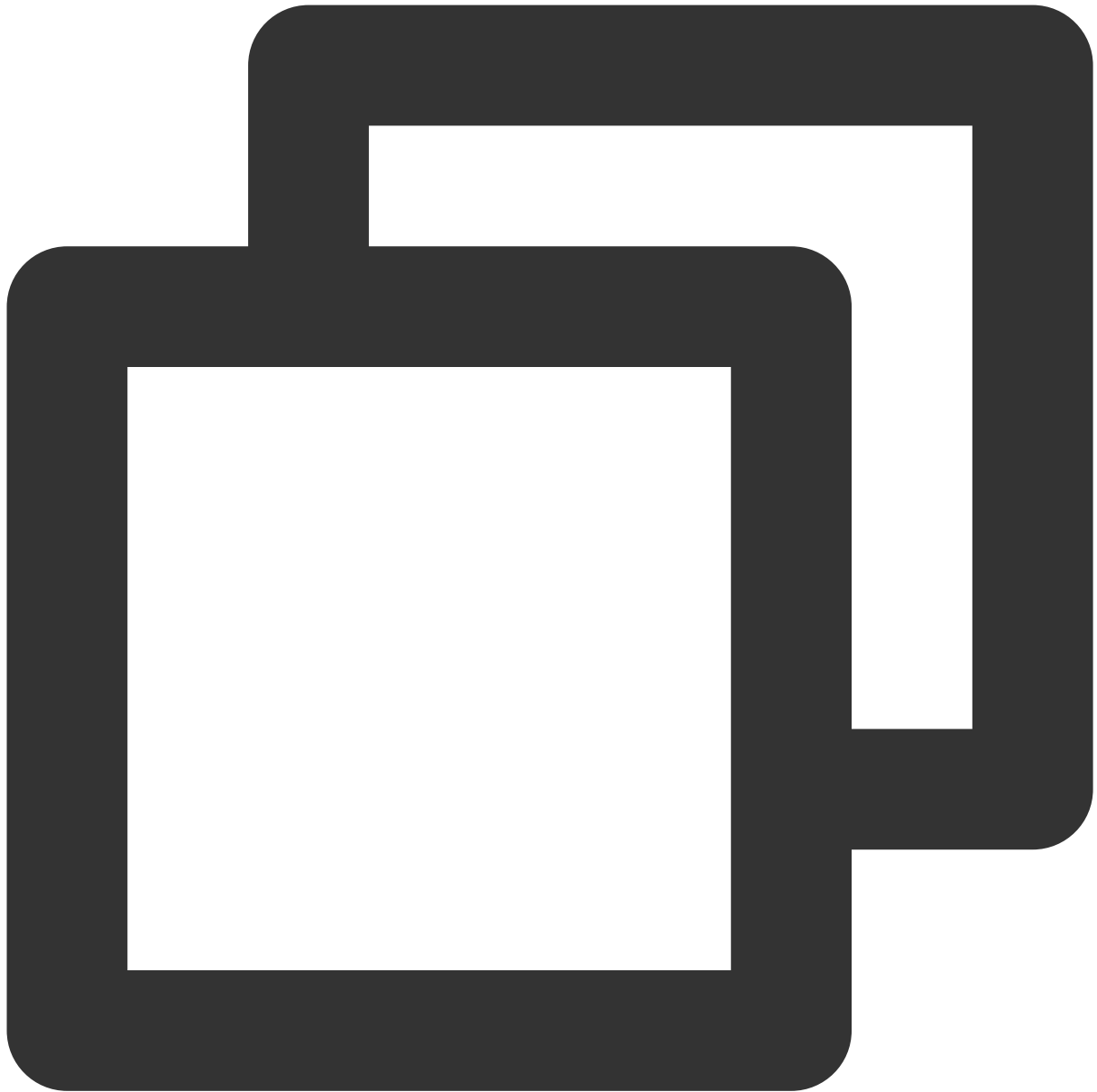


```
int ResumePreview()
```

現在のプレビューの時点を設定

このパラメータを呼び出して、現在のプレビューの時点を設定します。

関数のプロトタイプ



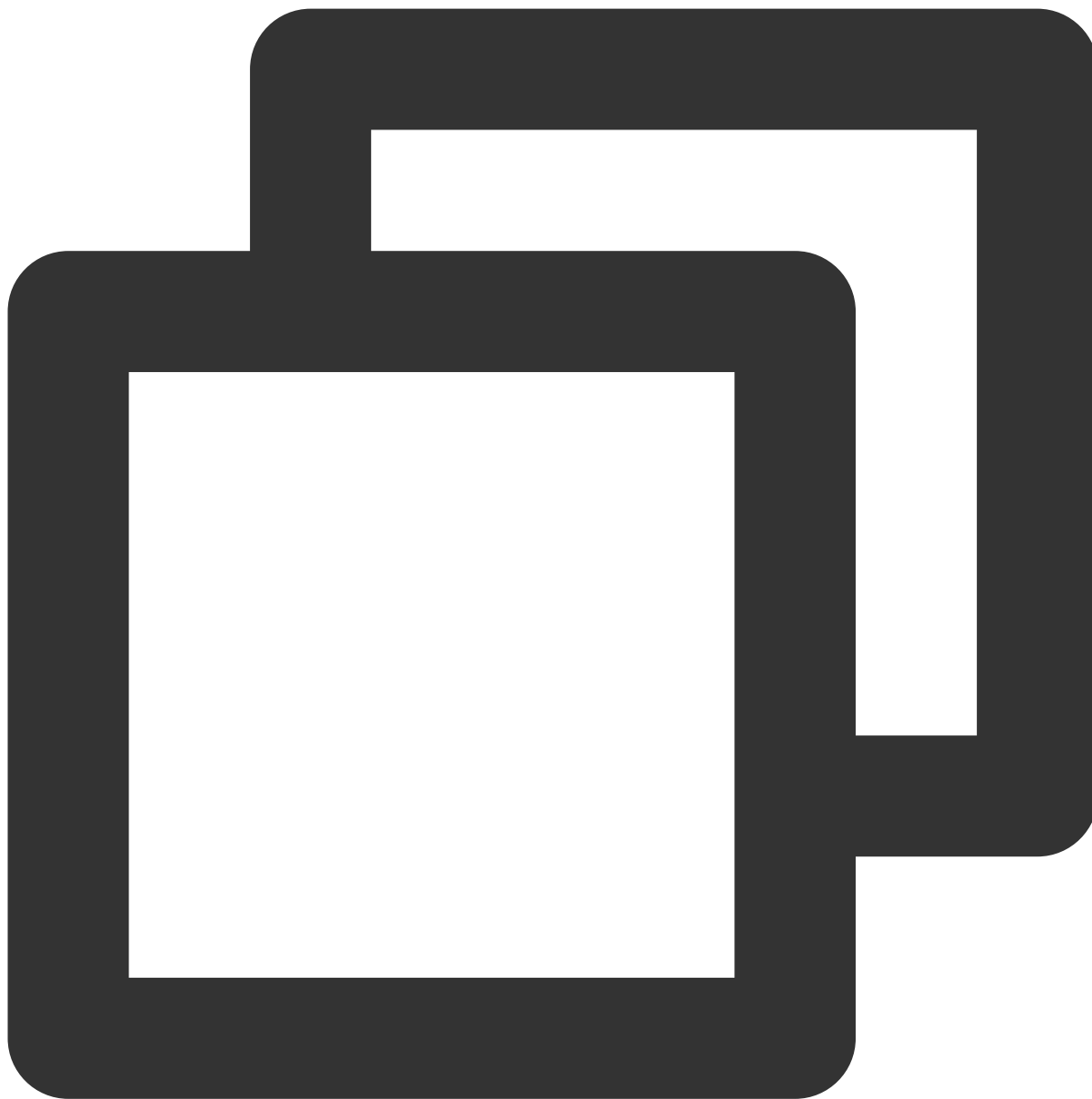
```
int SetPreviewTimeByMs(int time)
```

パラメータ	タイプ	意味
time	int	ファイルがプレビューされる時点（ミリ秒単位）。

現在のプレビューの時点を取得

このパラメータを呼び出して、現在のプレビューの時点を取得します。

関数のプロトタイプ



```
int GetPreviewTimeByMs()
```

プレビュー再生のコールバック

ITMG_MAIN_EVENT_TYPE_RECORD_PREVIEW_COMPLETED プレビューが完了したときのコールバック。プレビューファイルの再生が終了するか、StopPreview インターフェースを呼び出すとトリガーされます

コールバックパラメータ

--	--	--

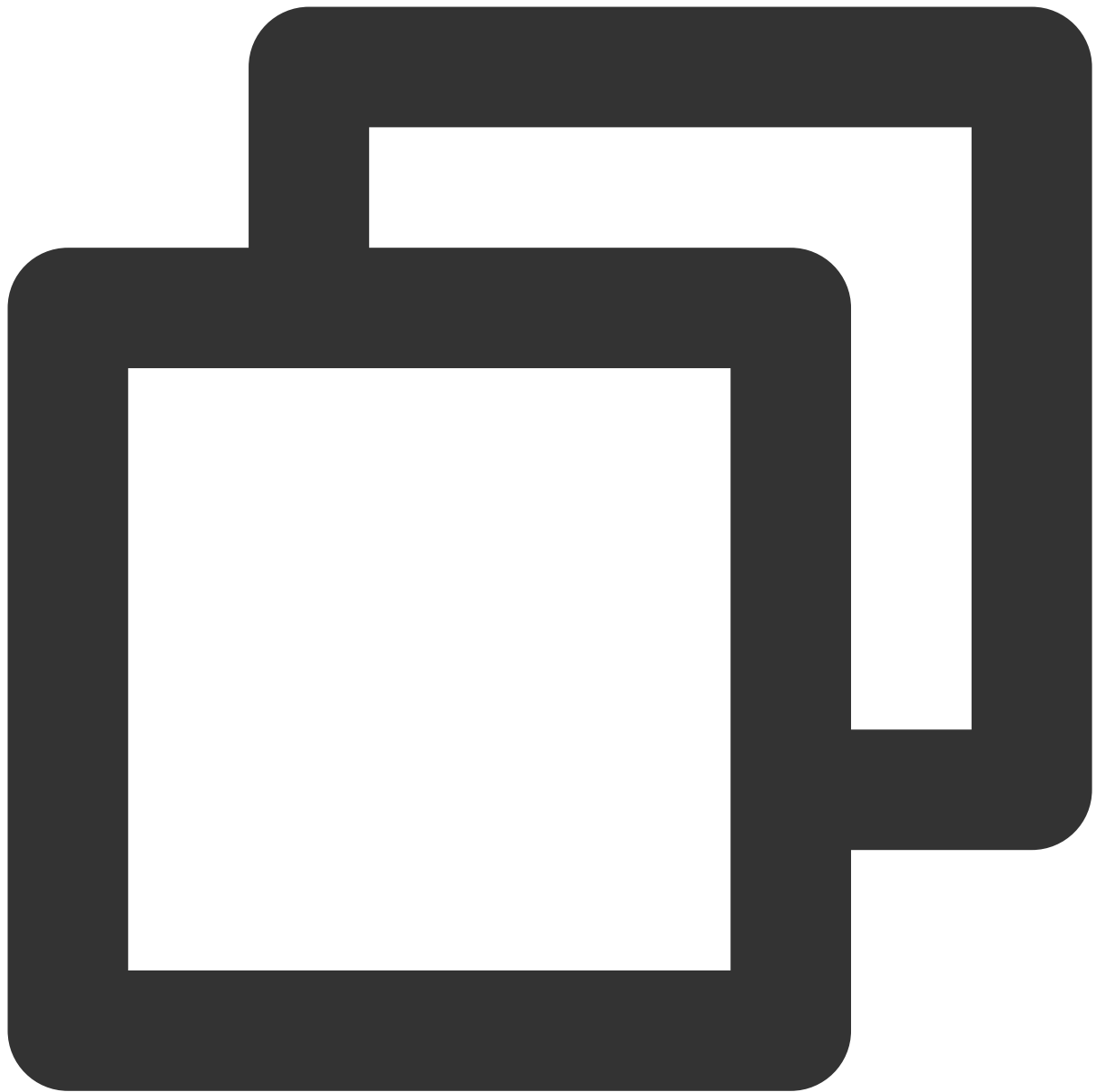
パラメータ	タイプ	意味
result	int	再生結果です。0は成功を意味します。

ファイル合成インターフェース

ファイル結合

このパラメータを呼び出して、録音された人の声と伴奏を1つのファイルにまとめます。

関数のプロトタイプ

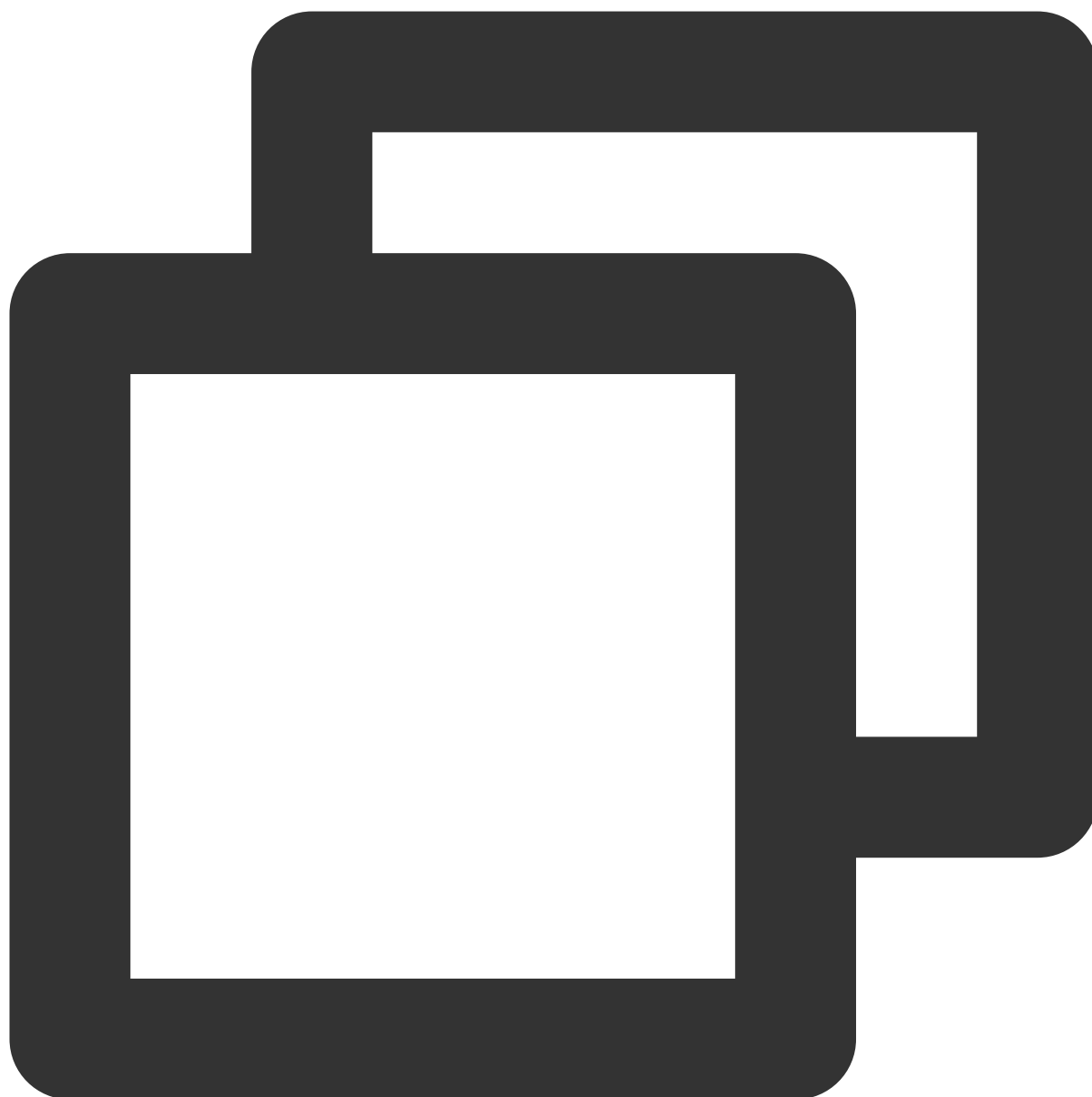


```
int MixRecordFile();
```

合成の解除

このパラメータを呼び出して、合成操作がキャンセルされます。

関数のプロトタイプ



```
int CancelMixRecordFile();
```

ファイル合成のコールバック

ITMG_MAIN_EVENT_TYPE_RECORD_MIX_COMPLETED プレビューが完了したときのコールバック。プレビューファイルの再生が終了するか、StopPreview インターフェースを呼び出してトリガされます。

コールバックパラメータ

パラメータ	タイプ	意味
-------	-----	----

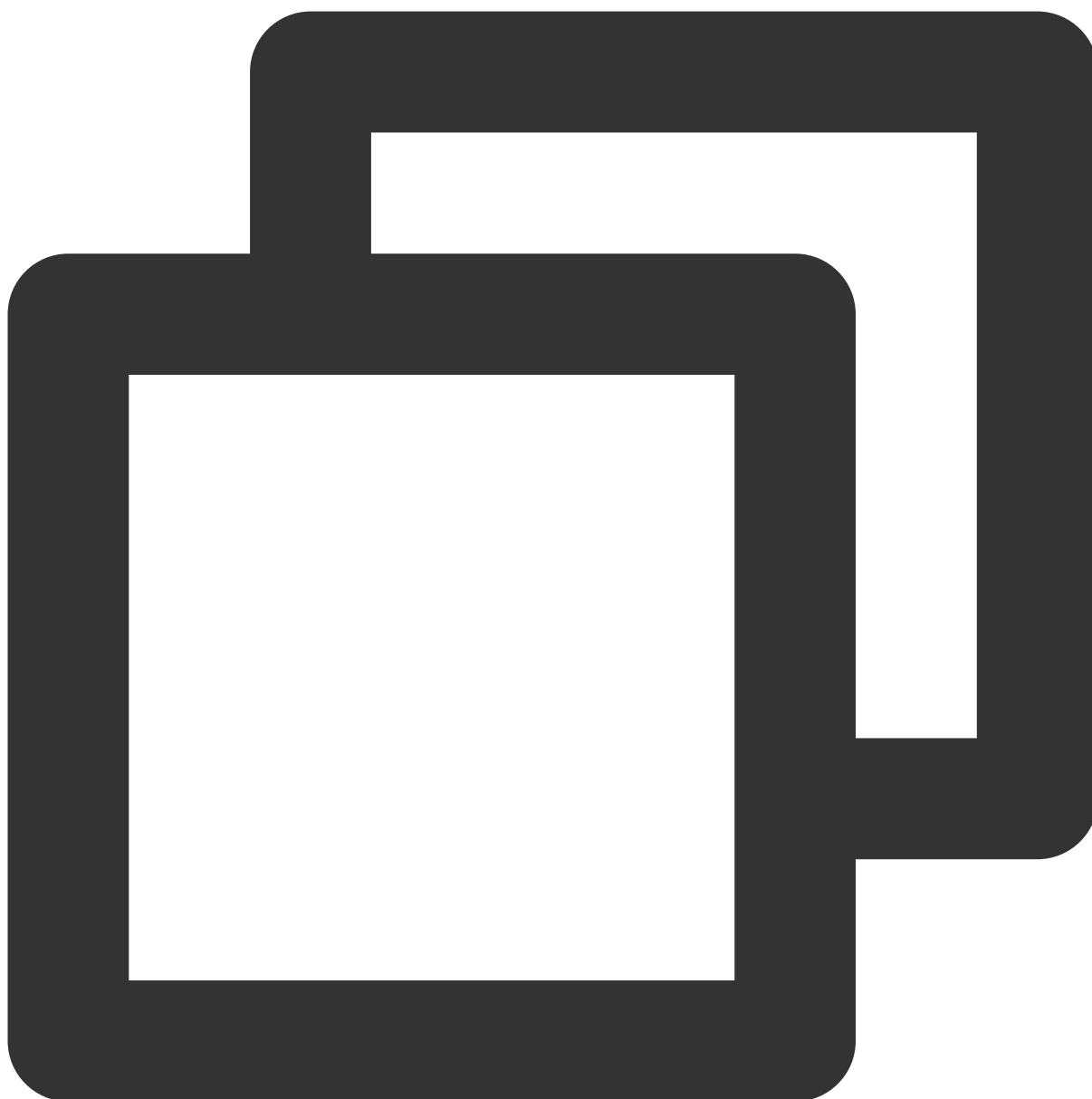
result	int	合成結果です。0は成功を意味します。
filepath	String	対象ファイルのパスであり、StartRecordインターフェースから渡されるdstFile。
duration	String	記録ファイルの長さ（ミリ秒単位）です。

詳細設定

伴奏スケールを設定

このインターフェースを呼び出して、人の声と伴奏のスケールを設定します。記録が完了したら調整できます。

関数のプロトタイプ



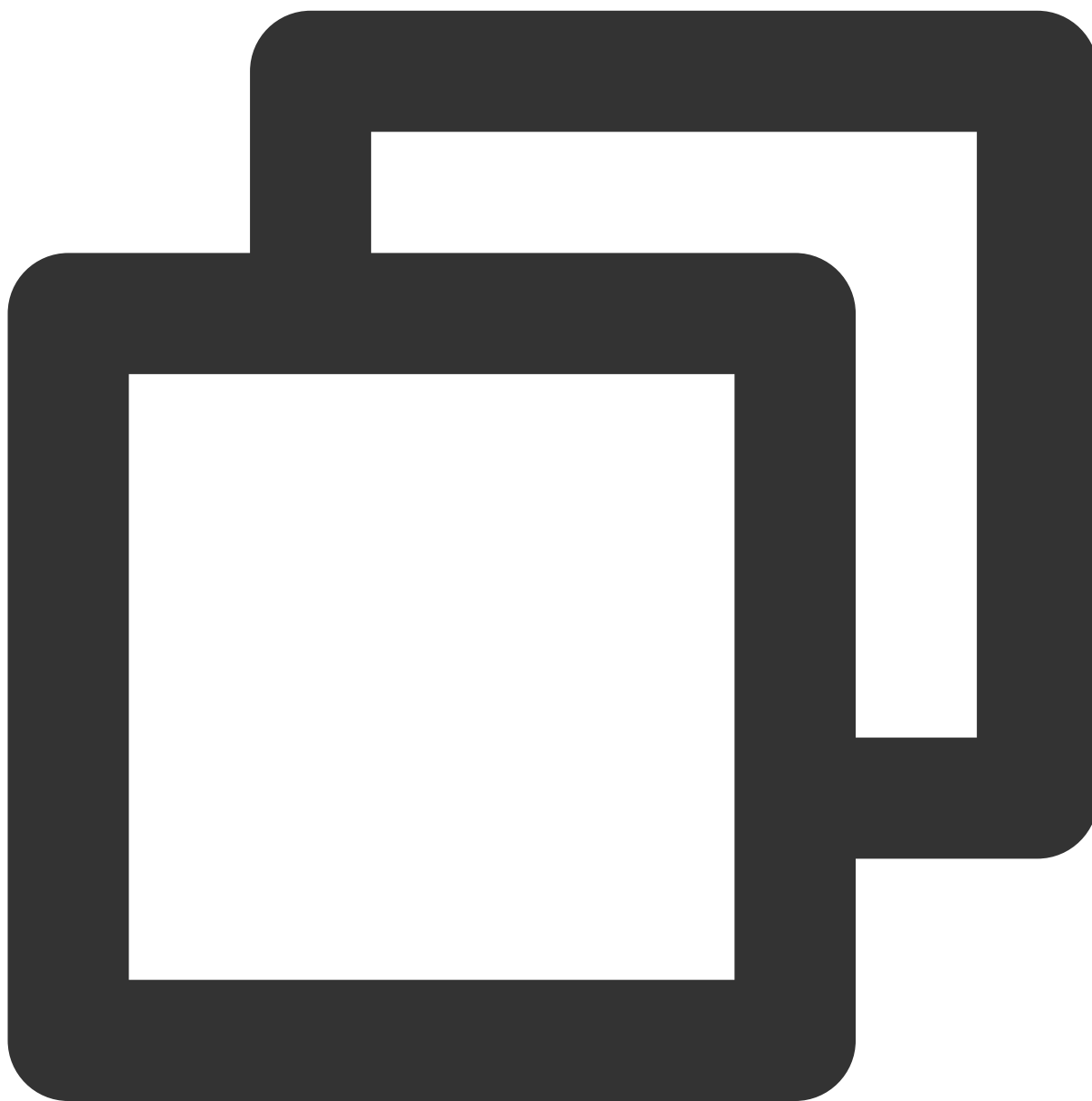
```
int SetMixWieghts(float mic, float acc)
```

パラ メータ	タイプ	意味
mic	float	人の声のスケールです。1.0が元の音量、1.0未満が縮小、1.0を超えると拡大、0～2の範囲で設定されます。
acc	float	伴奏のスケールです。1.0が元のボリューム、1.0未満が縮小、1.0を超えると拡大、0～2の範囲で設定されます。

オフセットを設定

このインターフェースを呼び出して伴奏に対する人の声のオフセットを設定します。通常、声がテンポに追いつかない問題を調整するために使用されます。記録が完了したら調整できます。

関数のプロトタイプ



```
int AdjustAudioTimeByMs(int time)
```

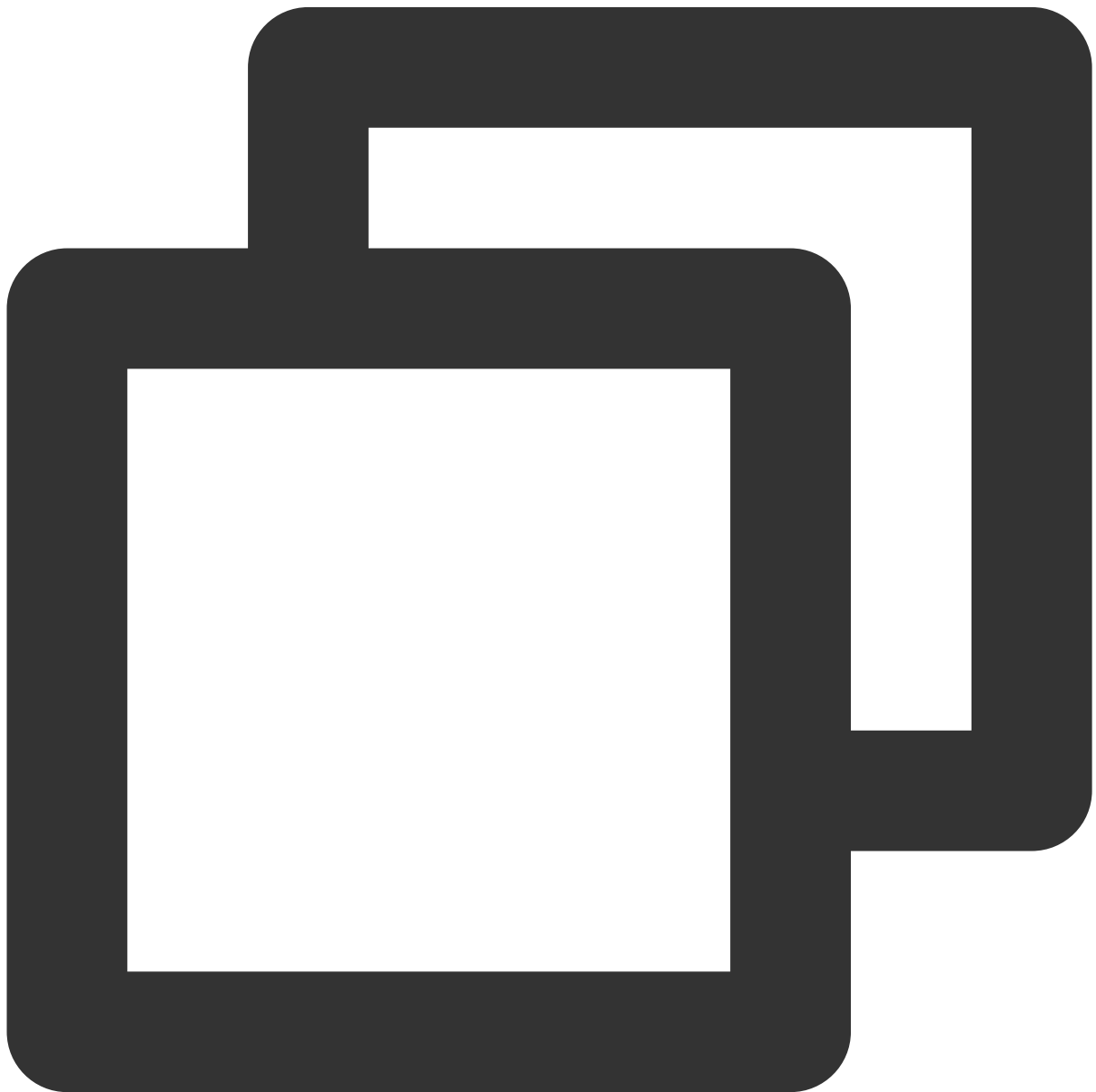
パラメータ	タイプ	意味
-------	-----	----

time	int	伴奏に対する人の声のオフセット時間（単位ms）。0より大きい場合は後方に移動し、0より小さい場合は前方に移動します。
------	-----	--

効果音の設定

効果音の種類を設定します。カラオケ効果音については[リアルタイム音声効果音ドキュメント](#)をご参照ください。調整は、レコーディングが完了した後または、レコーディング中にも行われることが可能です。

関数のプロトタイプ



```
int SetRecordKaraokeType(int type)
```

パラメータ	タイプ	意味
type	int	このタイプは、リアルタイム音声効果のカラオケ効果音のタイプと同様します。詳細については、 カラオケ効果音 をご参照ください。

ネットワークオーディオストリーム転送ルーティング

最終更新日：：2024-01-18 15:47:47

GME開発者がTencent Cloud GME製品APIのデバッグと導入を容易にするために、このドキュメントではGMEカスタマイズオーディオ転送ルーティング機能に適している使用参考ドキュメントを紹介します。

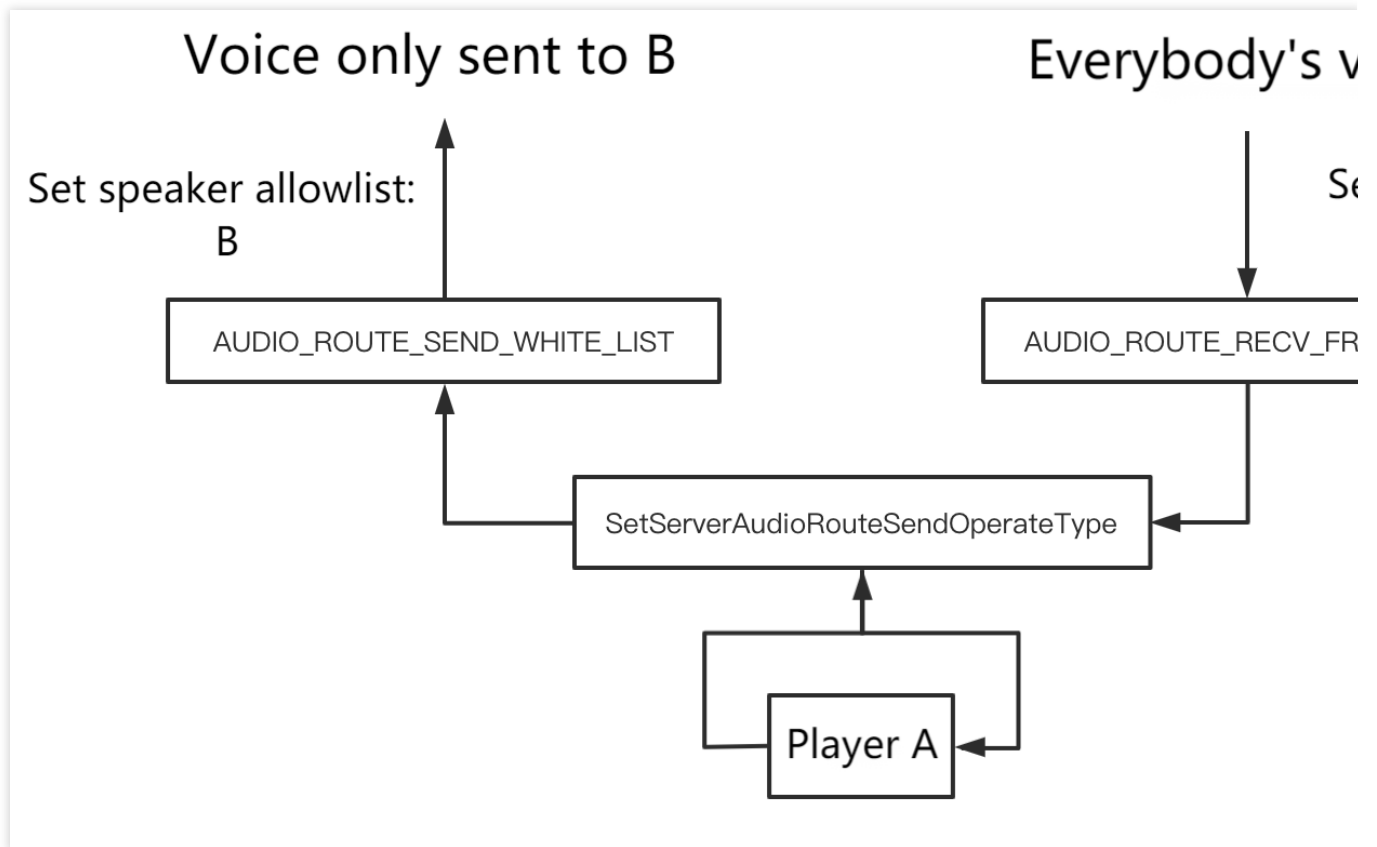
シナリオ

場面説明：2人の友達がチームを組んだ後、3人の知らない人をマッチングして大きなチームを組み、チーム全員の声を聞いて、チームの友達と話すような機能が必要です。

カスタムオーディオルーティング機能でそれを実装することができます。ここでは5人全員が同じ音声ルームに入り、音声ルーティングのインターフェース設定を行い、2人チームの音声のみまたは全ルームの音声聞こえるように設定したり、2人チームの者のみに話が聞こえるように設定したり、全ルームの者に話が聞こえるように設定したりすることができます。

オーディオルール距離：`SetServerAudioRouteSendOperateType(AUDIO_ROUTE_SEND_WHITE_LIST,"2人チームのlist",ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE,"2人チームのlist");`

これにより、音声はlistに含まれる人にも送信されるとともに、3人チームの音声のみが受信されます。



前提条件

リアルタイム音声サービスが有効になっていること：[サービス有効化ガイド](#)をご参照ください。

GME SDK導入済み：コアインターフェースとリアルタイム音声インターフェースの導入を含みます。詳細については、[Native SDKクイックスタート](#)、[Unity SDKクイックスタート](#)、[Unreal SDKクイックスタート](#)をご参照ください。

GMEリアルタイム音声機能を使用して音声ルームへの参加に成功し、マイク（EnableMic）、スピーカー（EnableSpeaker）をオンにしました。

音声転送ルーティング機能の導入

オーディオ転送ルールの設定

このインターフェースを呼び出して音声転送ルールを設定します。このインターフェースは、入室のコールバックに成功したときに呼び出され、呼び出し後にこの入室が有効になり、退室後に無効になります。

ご注意：

発言禁止機能AddBlackListはネイティブに有効で、カスタムオーディオルーティングよりも優先されます。例えば、AはSetServerAudioRouteSendOperateTypeでBの発話のみを聞くように設定したが、AddBlackListを呼び出し

てBの発言を禁止した場合、AはBの声を聞くことができなくなります。

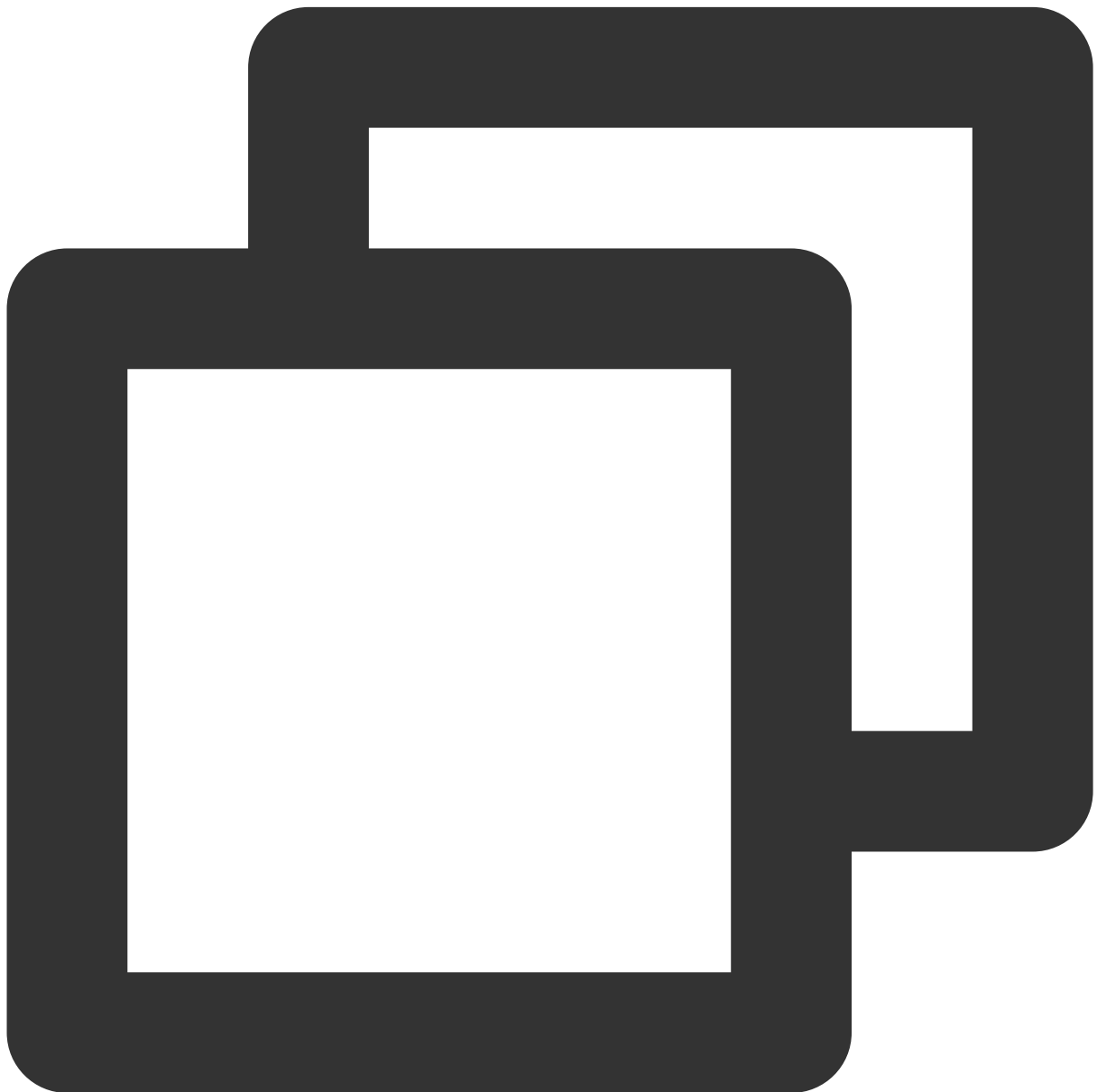
インターフェースのプロトタイプ

Unity

C++

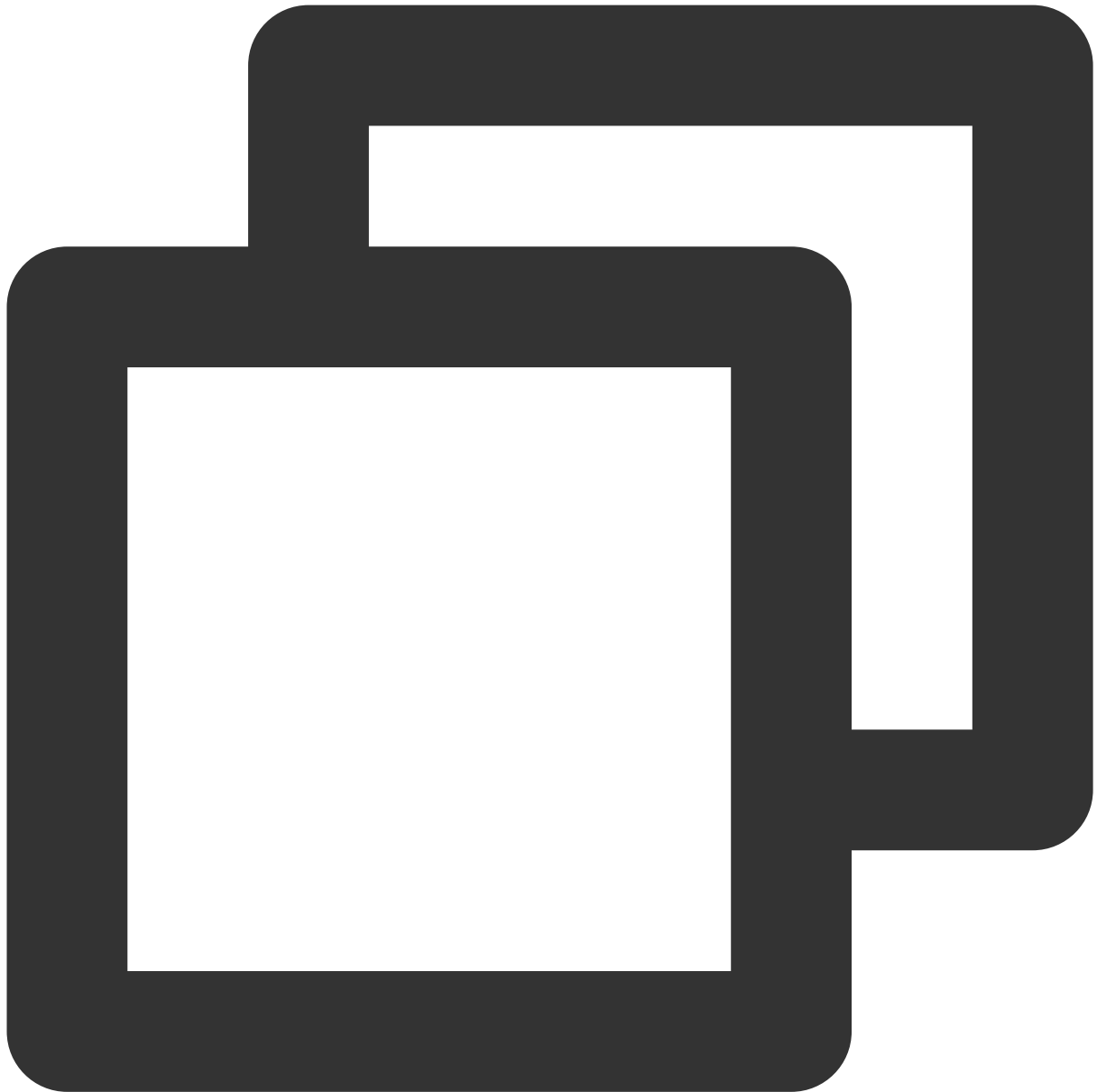
Android

iOS

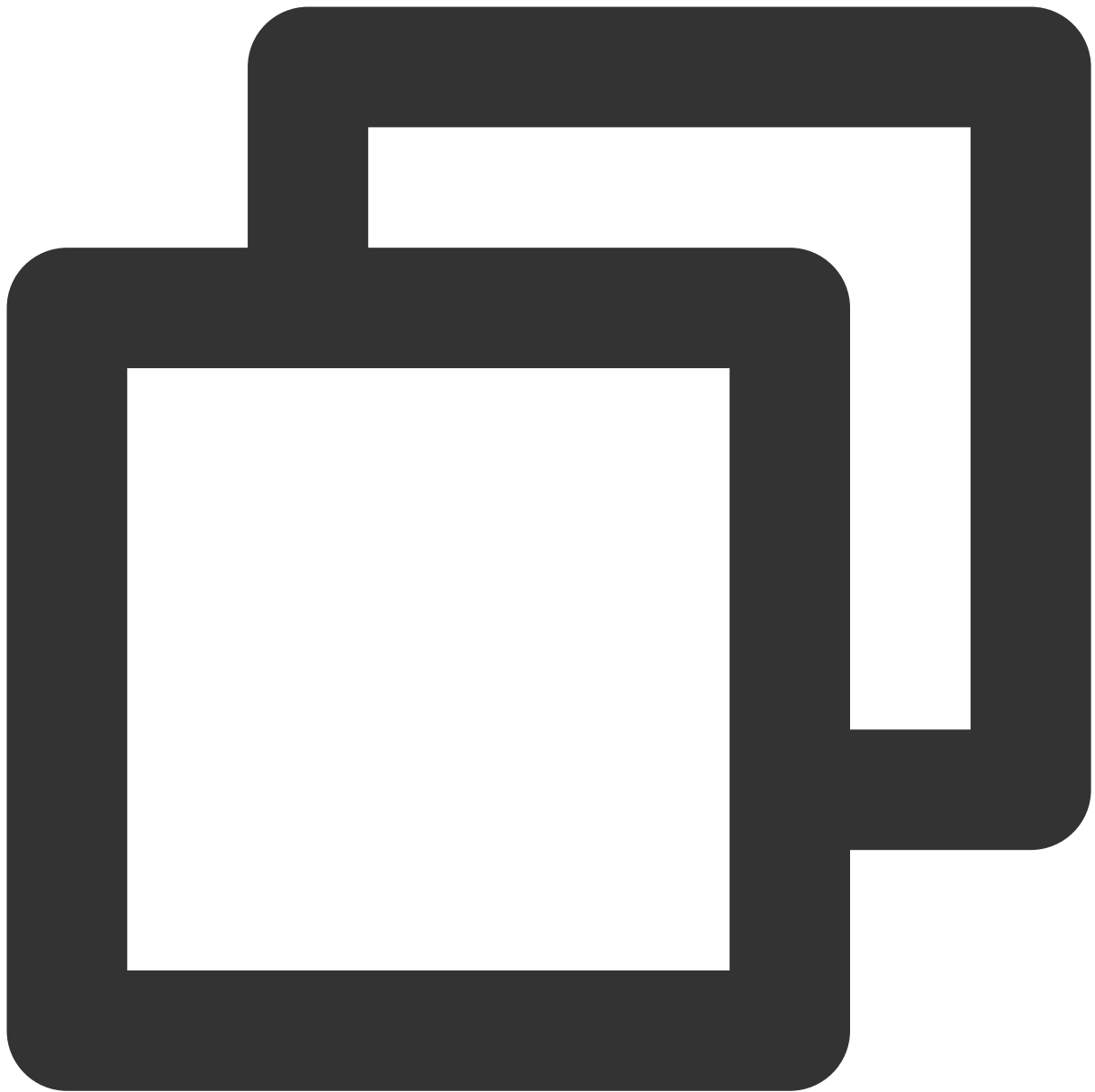


```
public abstract class ITMGRoom{  
    public abstract int SetServerAudioRouteSendOperateType(ITMG_SERVER_AUDIO_ROUTE_
```

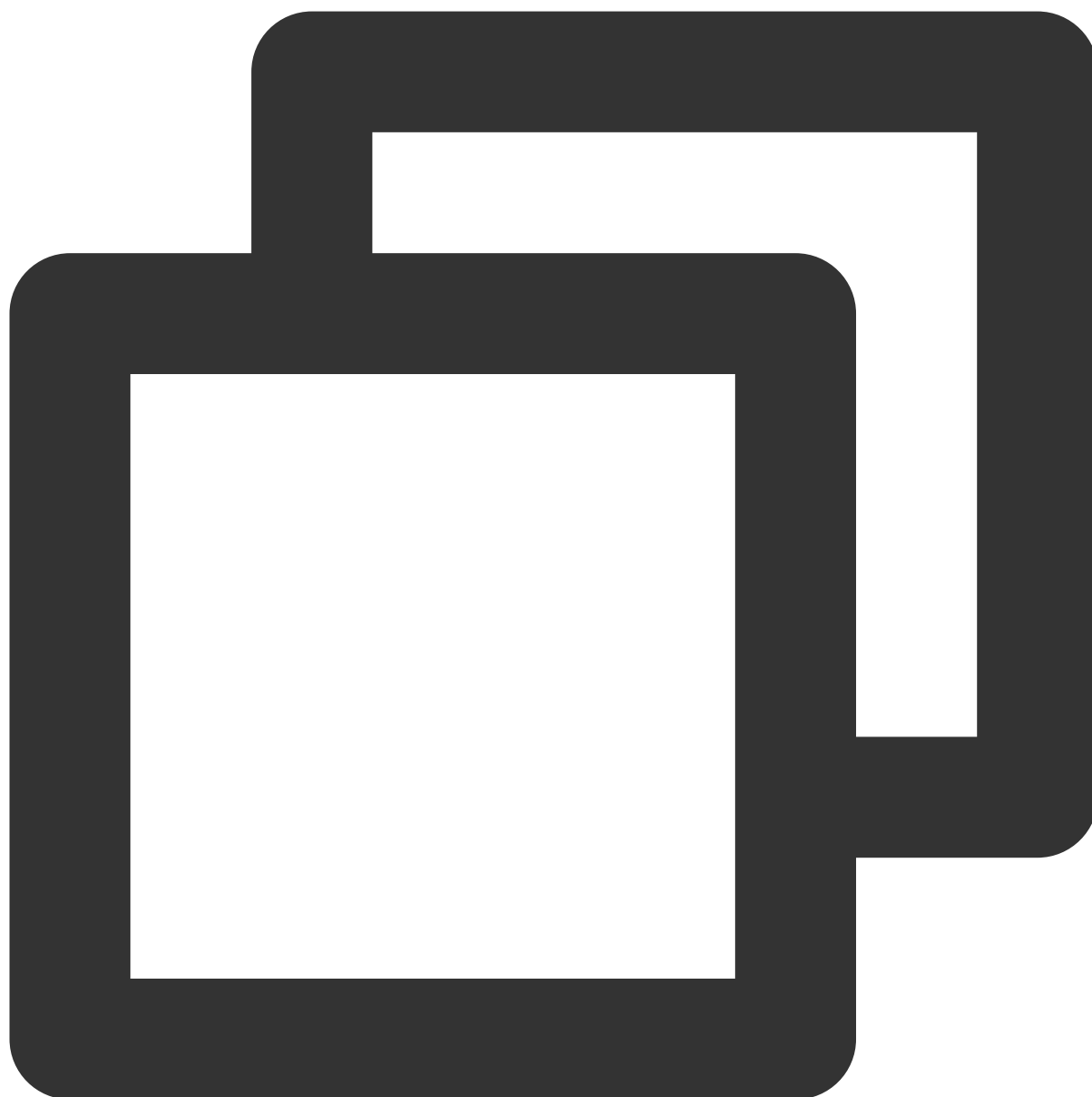
```
}
```



```
virtual int SetServerAudioRoute(ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE SendType, const c
```



```
public abstract int SetServerAudioRoute(ITMGContext.ITMG_SERVER_AUDIO_ROUTE_SEND_TY
```



```
-(int)SetServerAudioRouteSendOperateType:(ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE) Sendty
```

タイプの説明

ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE

音声送信ルールを設定し、異なるルールを入力すると、異なる送信ルールが設定されます。

受信タイプ	効果
AUDIO_ROUTE_NOT_SEND_TO_ANYONE	ローカルからのアップリンクオーディオはバックグラウンドに送信されますが、バックグラウンドはそれをだれにも転送

	しません。これはご自身をミュートすることと同じです。このとき、パラメータOpenIDForSendは無効で、nullを入力すればよいです。
AUDIO_ROUTE_SEND_TO_ALL	ローカルからのアップリンクオーディオは全員に転送されます。このとき、パラメータOpenIDForSendは無効で、nullを入力すればよいです。
AUDIO_ROUTE_SEND_BLACK_LIST	ローカルからのアップリンクオーディオは、パラメータOpenIDForSendによって提供されるブラックリスト内の人には転送されません。
AUDIO_ROUTE_SEND_WHITE_LIST	ローカルからのアップリンクオーディオは、パラメータOpenIDForSendによって提供されるホワイトリスト内の人だけに転送されます。

説明：

タイプにAUDIO_ROUTE_NOT_SEND_TO_ANYONEおよびAUDIO_ROUTE_SEND_TO_ALLが渡された場合、パラメータOpenIDForSendは有効でなく、nullを入力すればよいです。

タイプにAUDIO_ROUTE_SEND_BLACK_LISTが渡された場合、パラメータOpenIDForSendはブラックリストで、最大10個までサポートされます。

タイプにAUDIO_ROUTE_SEND_WHITE_LISTが渡された場合、パラメータOpenIDForSendはホワイトリストで、最大10個までサポートされます。

ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE

音声受信ルールを設定します。異なるルールが入力されると、異なる受信ルールが設定されます。

受信タイプ	効果
AUDIO_ROUTE_NOT_RECV_FROM_ANYONE	ローカルでは、すべてのオーディオを受け入れません。これはルーム内のスピーカー効果をオフにすることと同じです。このとき、パラメータOpenIDForSendは無効で、nullを入力すればよいです。
AUDIO_ROUTE_RECV_FROM_ALL	ローカルでは、全員のオーディオを受け入れます。このとき、パラメータOpenIDForSendは無効で、nullを入力すればよいです。
AUDIO_ROUTE_RECV_BLACK_LIST	ローカルでは、パラメータOpenIDForSendによって提供されるブラックリスト内の人々のオーディオを受け入れません。
AUDIO_ROUTE_RECV_WHITE_LIST	ローカルでは、パラメータOpenIDForSendによって提供されるホワイトリスト内の人々のオーディオのみ受け入れます。

説明：

タイプにAUDIO_ROUTE_NOT_RECV_FROM_ANYONEおよびAUDIO_ROUTE_RECV_FROM_ALLが渡された場合、パラメータOpenIDForSendは有効になりません。

タイプにAUDIO_ROUTE_RECV_BLACK_LISTが渡された場合、パラメータOpenIDForSendはブラックリストで、最大10個までサポートされます。

タイプにAUDIO_ROUTE_RECV_WHITE_LISTが渡された場合、パラメータOpenIDForSendはホワイトリストで、最大10個までサポートされます。

戻り値

インターフェースの戻り値がQAV_OKの場合、成功したことを示します。

コールバックが1004を返した場合、パラメータが間違っていることを示します。パラメータが正しいかどうかを再確認することをお勧めします。

コールバックが1001を返した場合は、動作が繰り返されることを示します。

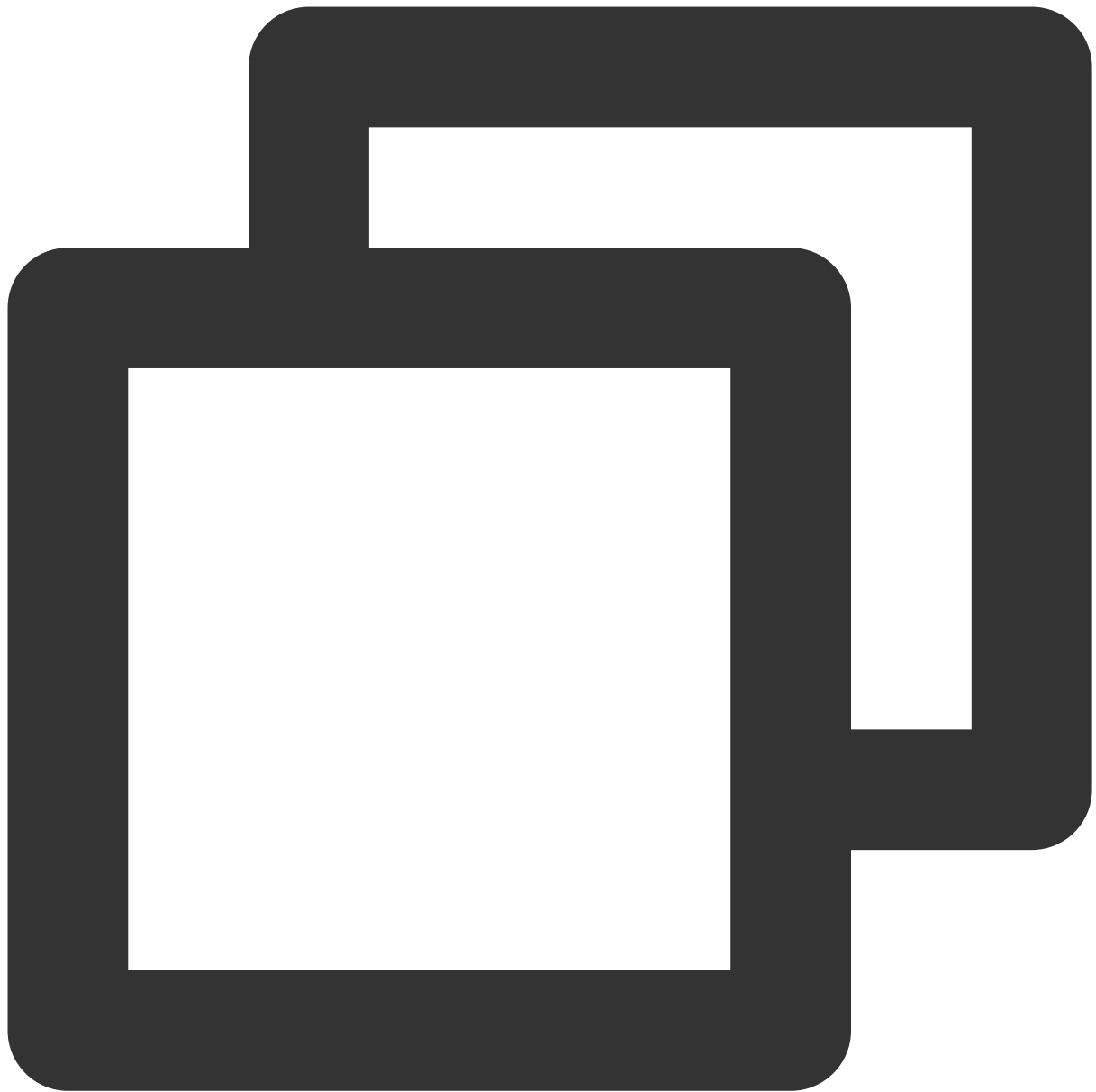
コールバックが1201を返した場合は、ルームが存在しないことを示します。ルーム番号が正しいかどうかを確認することをお勧めします。

コールバックが10001と1005を返した場合は、インターフェースをもう一度呼び出すことをお勧めします。

返された結果の詳細について、[エラーコード](#)をご参照ください。

サンプルコード

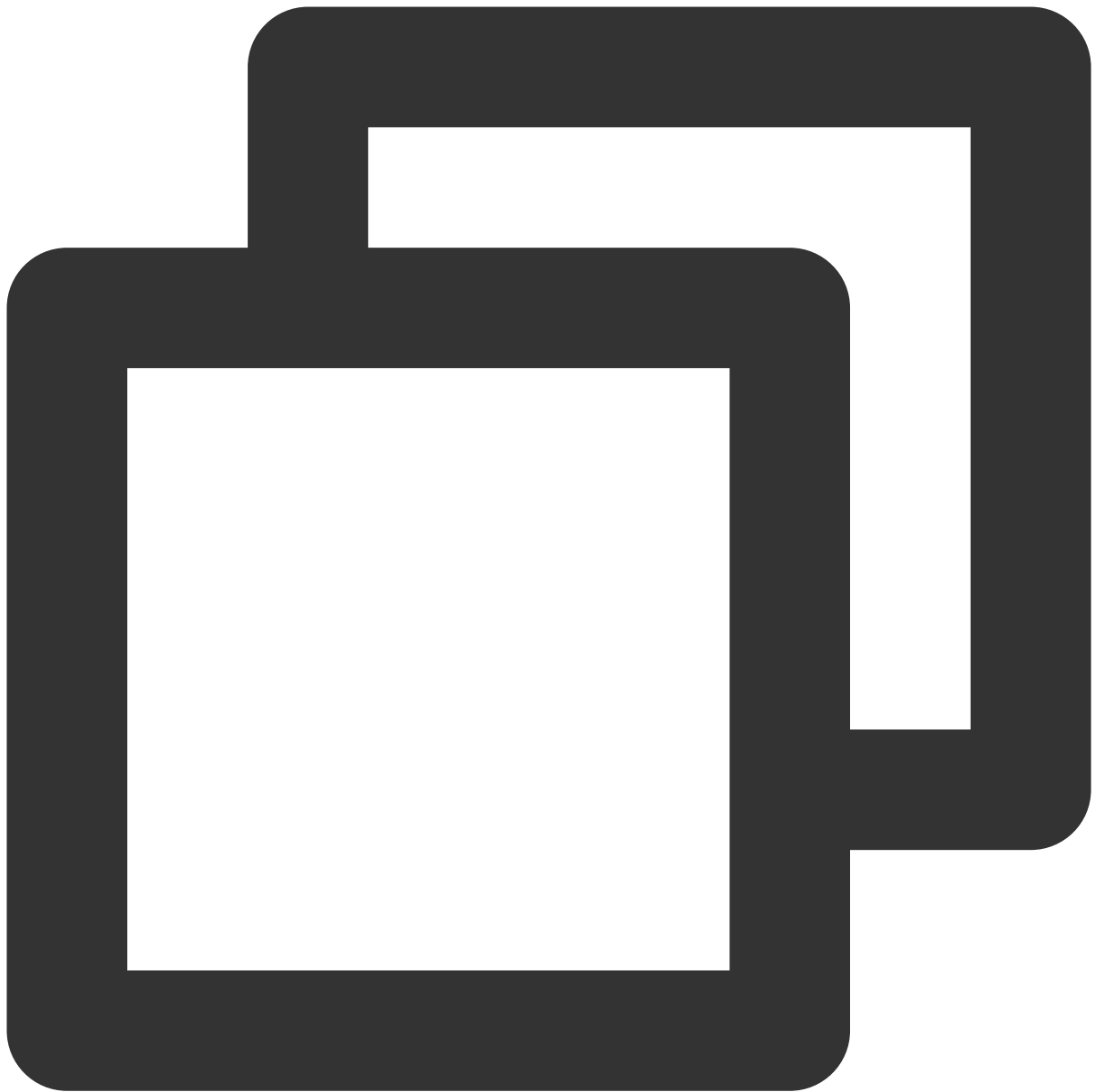
実行ステートメント



```
@synthesize _sendListArray;
@synthesize _recvListArray;

int ret = [[ITMGContext GetInstance] GetRoom] SetServerAudioRouteSendOperateType:
if (ret != QAV_OK) {
    UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"audiorouteリストの更新に
    [alert show];
}
```

コールバック



```
-(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:@"audiorout"
                                                                message:log
                                                                delegate:nil
                                                                cancelButtonTitle:@"OK"
                                                                otherButtonTitles:nil];
            [alert show];
        }
        default:
            break;
    }
}
```

```
}  
}
```

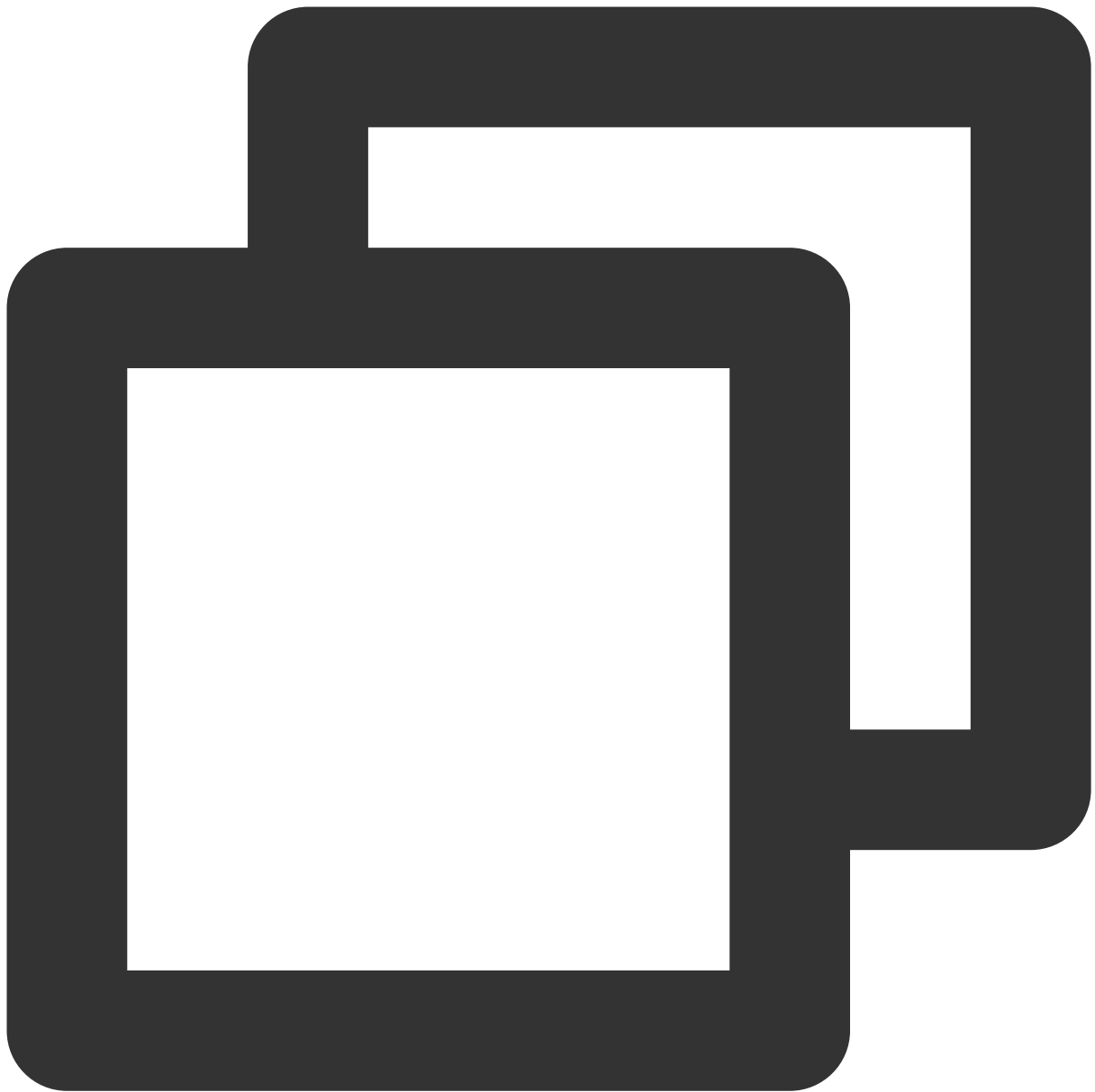
オーディオ設定転送ルールの取得

このインターフェースを呼び出すとオーディオ転送ルールを取得します。呼び出し後、インターフェースはルールを返します。渡された配列パラメータは、対応するルールのopenIdを返します。

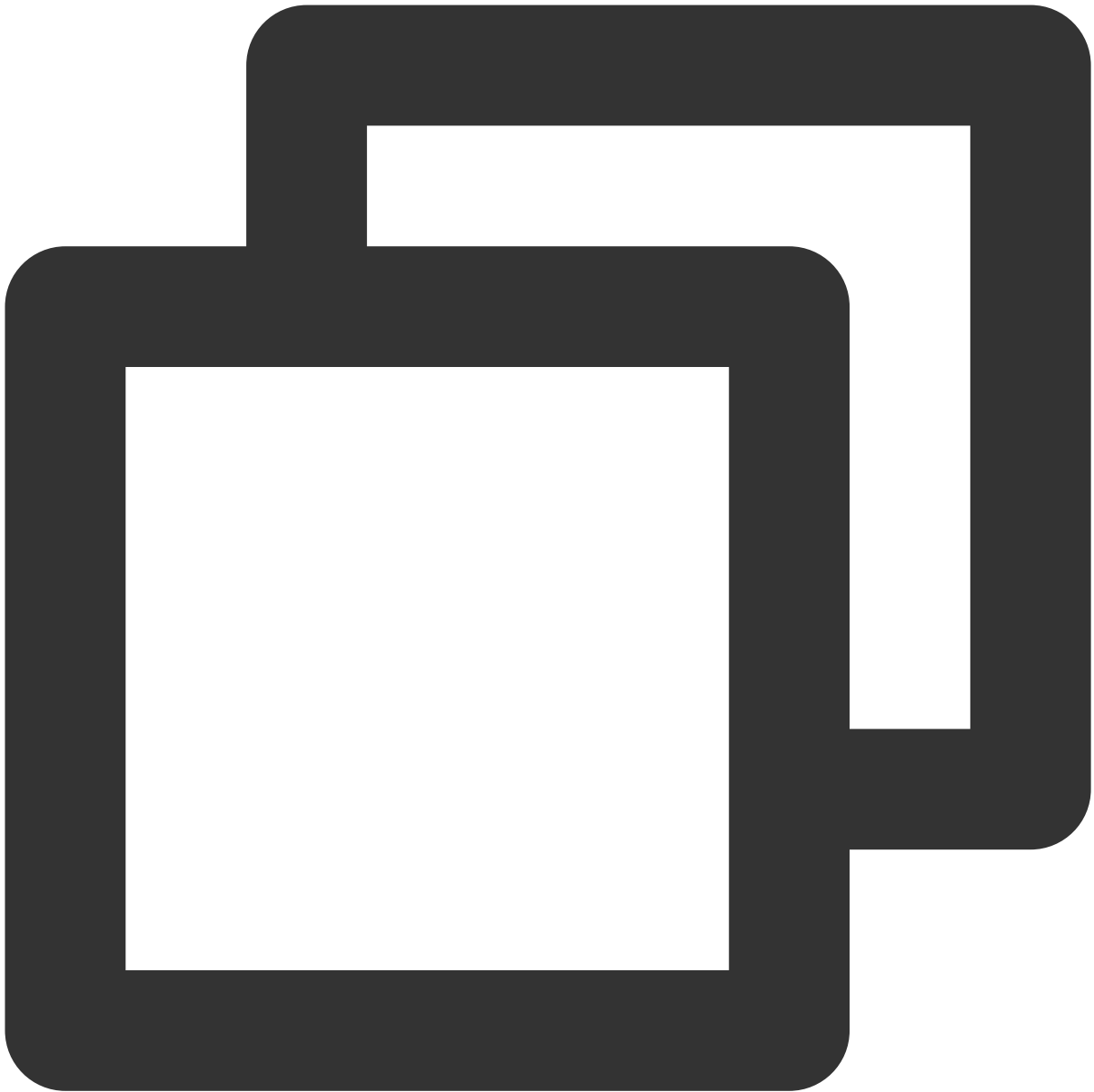
インターフェースのプロトタイプ

Unity

iOS



```
public abstract ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE GetCurrentSendAudioRoute(List<str  
public abstract ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE GetCurrentRecvAudioRoute(List<str
```



```
- (ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE)GetCurrentSendAudioRoute:(NSMutableArray *) Open
- (ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE)GetCurrentRecvAudioRoute:(NSMutableArray *) Open
```

戻りルール

ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE

受信タイプ	効果
AUDIO_ROUTE_NOT_SEND_TO_ANYONE	ローカルからのアップリンクオーディオはバックグラウンドに送信されますが、バックグラウンドはそれをだれにも転送

	しません。これはご自身をミュートすることと同じです
AUDIO_ROUTE_SEND_TO_ALL	ローカルからのアップリンクオーディオは全員に転送されます
AUDIO_ROUTE_SEND_BLACK_LIST	ローカルからのアップリンクオーディオはブラックリスト内の人には転送されません
AUDIO_ROUTE_SEND_WHITE_LIST	ローカルからのアップリンクオーディオはホワイトリスト内の人にはのみ転送されます
AUDIO_ROUTE_RECV_INQUIRE_ERROR	取得にエラーが発生しました。ルームに参加したか、SDKが初期化されているかを確認します

ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE

受信タイプ	効果
AUDIO_ROUTE_NOT_RECV_FROM_ANYONE	ローカルでは、すべてのオーディオを受け入れません。これはルーム内のスピーカー効果をオフにすることと同じです
AUDIO_ROUTE_RECV_FROM_ALL	ローカルでは、全員のオーディオを受け入れます
AUDIO_ROUTE_RECV_BLACK_LIST	ローカルでは、ブラックリスト内の人々のオーディオを受け入れません
AUDIO_ROUTE_RECV_WHITE_LIST	ローカルでは、ホワイトリスト内の人々のオーディオのみ受け入れます
AUDIO_ROUTE_RECV_INQUIRE_ERROR	取得にエラーが発生しました。ルームに参加したか、SDKが初期化されているかを確認します

ご注意：

`SetServerAudioRouteSendOperateType` インターフェースで `AUDIO_ROUTE_RECV_INQUIRE_ERROR` を使用しないでください。

カスタムメッセージチャネル

最終更新日：2024-01-18 15:47:47

GME開発者がTencent Cloud GME製品APIのデバッグと導入を容易にするために、このドキュメントではGMEユーザーカスタマイズオーディオパックにメッセージが付属される機能の使用について紹介します。

シナリオ

GMEユーザーカスタムオーディオパックにメッセージ機能が付属することにより、開発者はGMEオーディオパックにカスタムメッセージを持ち運び、同室の人にブロードキャストするためのシグナリングとして機能することができます。

前提条件

リアルタイムボイスサービスを有効にしました：[サービス有効化ガイド](#)をご参照ください。

GME SDK導入済み：コアインターフェースとリアルタイム音声インターフェースの導入が含まれます。詳細については、[Native SDKのクイック導入](#)、[Unity SDKクイック導入](#)、[Unreal SDKクイック導入](#)をご参照ください。

使用制限

このインターフェースを呼び出すには、**Standard**および**High-Definition**（ITMG_ROOM_TYPE_STANDARDおよびITMG_ROOM_TYPE_HIGHQUALITY）のルームタイプが必要で、また送信側でマイクをオンにし、受信側でスピーカーをオンにする必要があります。

カスタムメッセージ機能の導入

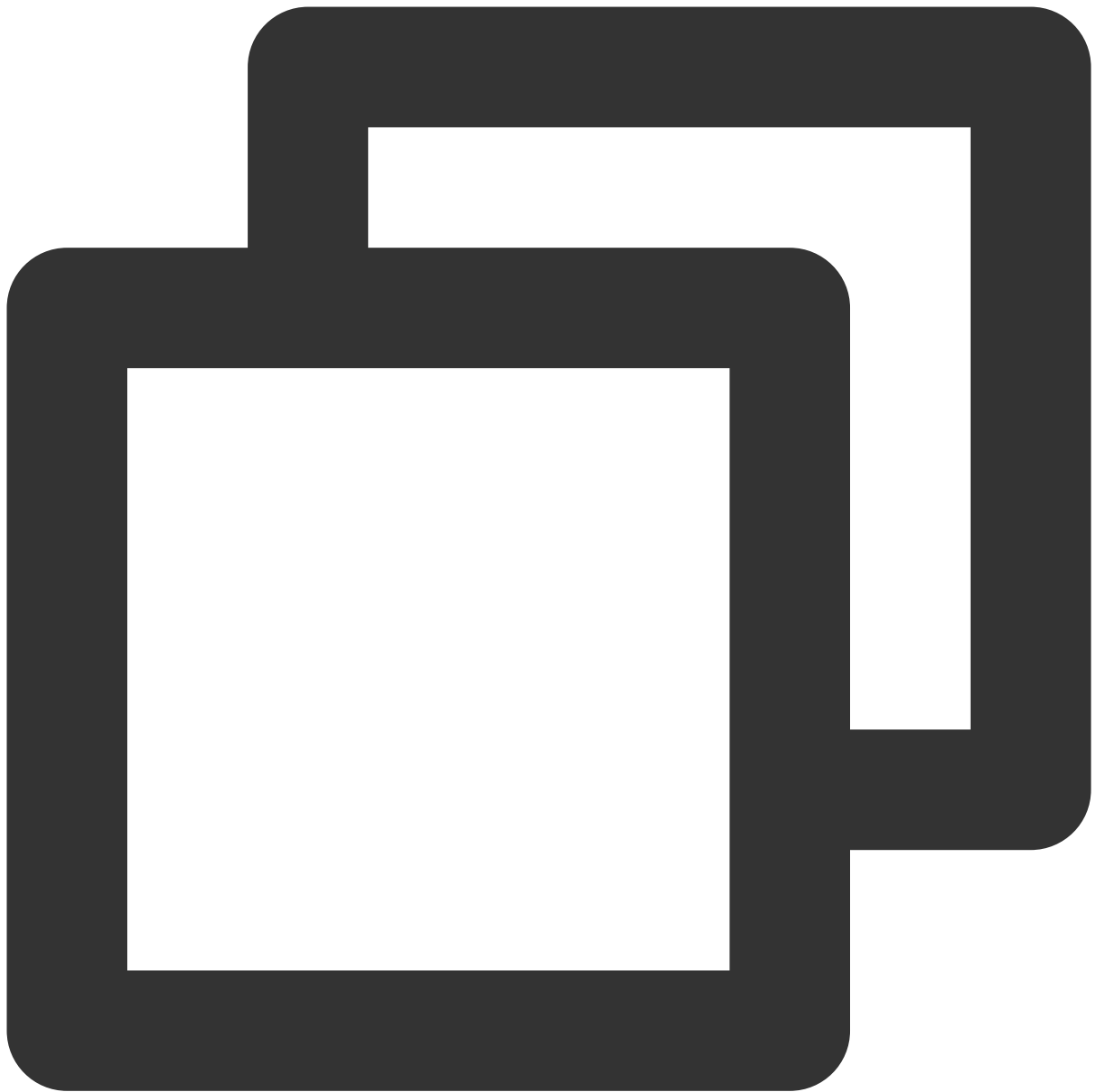
カスタムメッセージの送信

インターフェースのプロトタイプ

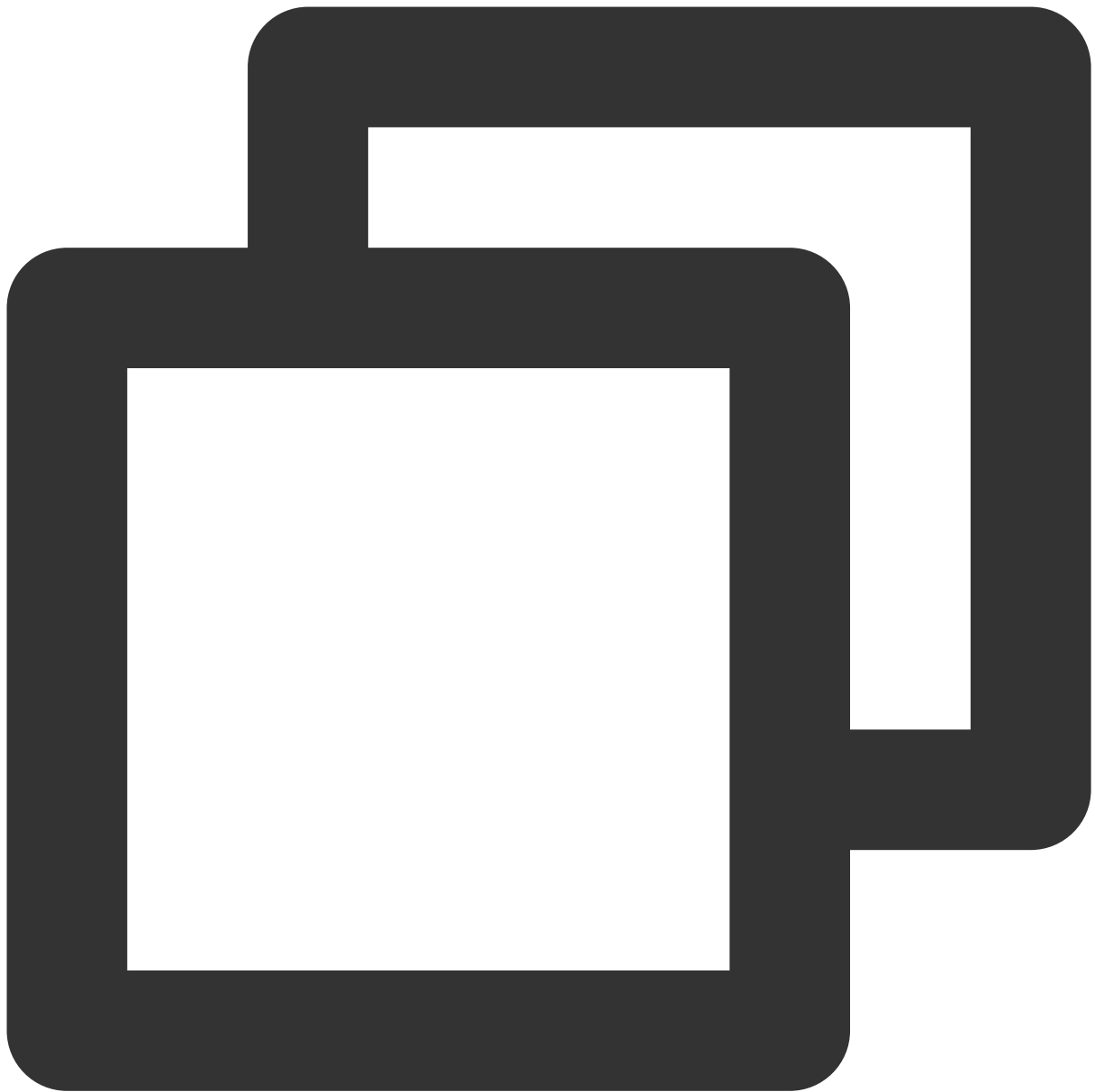
iOS

Android

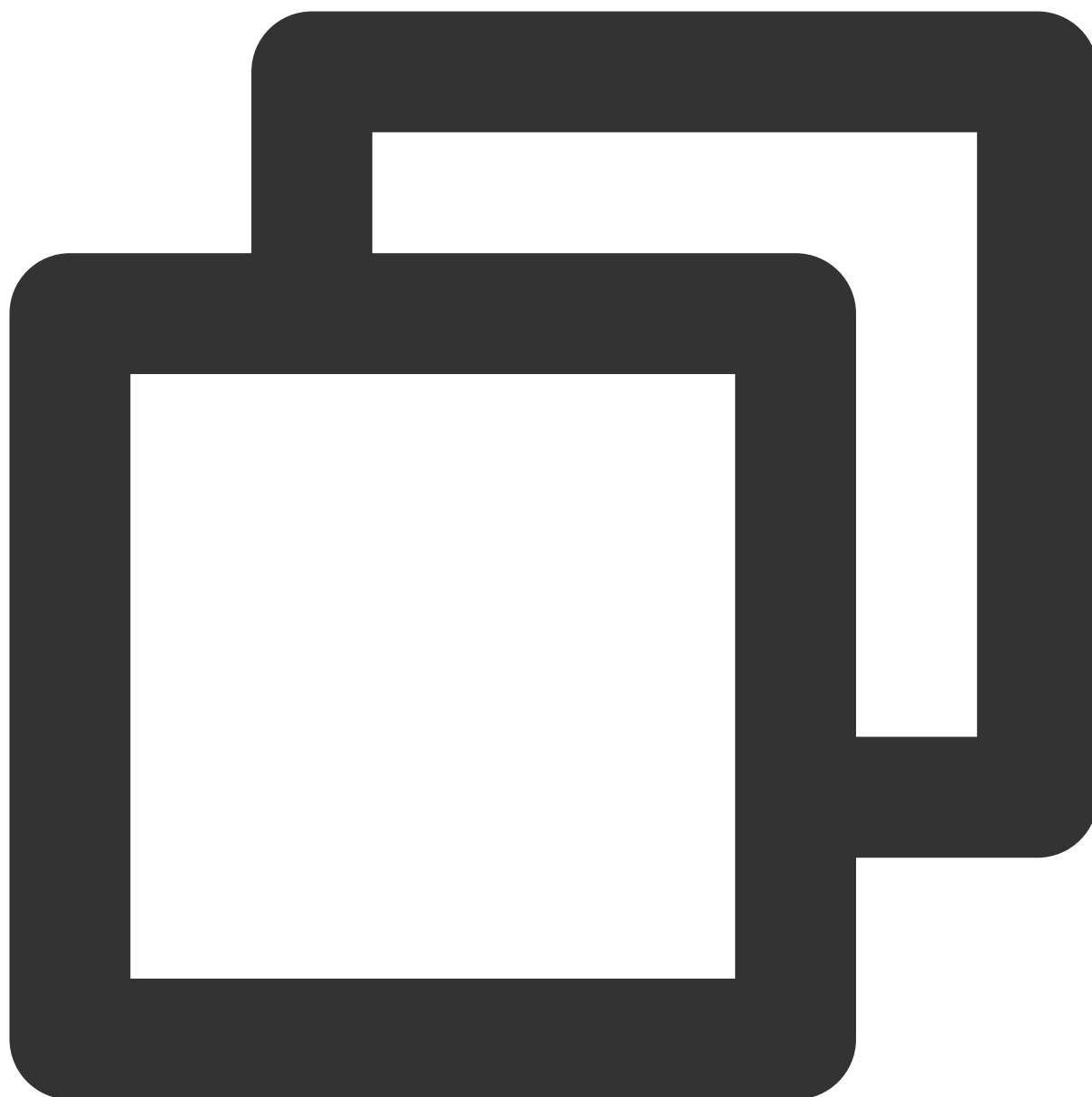
Unity



```
-(int) SendCustomData:(NSData *)data repeatCout:(int)reaptCout;
```



```
public abstract int SendCustomData(byte[] data,int repeatCout);
```



```
public abstract int SendCustomData(byte[] customdata,int repeatCout);
```

パラメータの説明

パラメータ	タイプ	意味
data	NSData *、byte[]	渡す情報
reaptCout	int	繰返し回数です。-1を入力すると無限回繰返し送信となります

戻り値

インターフェースの戻り値がQAV_OKの場合、成功したことを示します。

コールバックが1004を返した場合、パラメータが間違っていることを示します。パラメータが正しいかどうかを再確認することをおすすめします。1201が返された場合はルームが存在しないことを示します。ルーム番号が正しいかどうかを確認することをお勧めします。

エラーコードの詳細については、[エラーコード](#)ドキュメントをご参照ください。

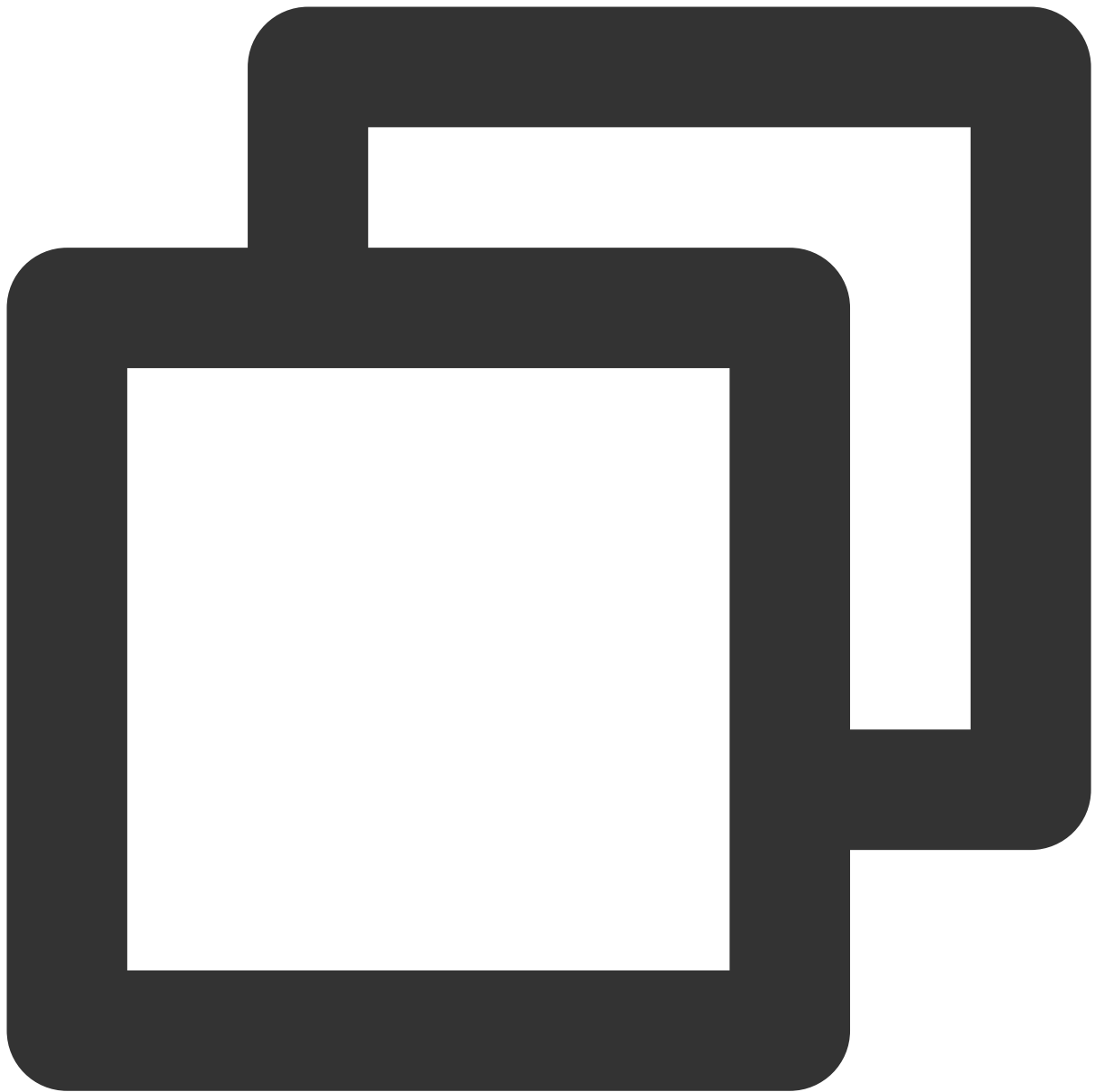
サンプルコード

実行ステートメント

iOS

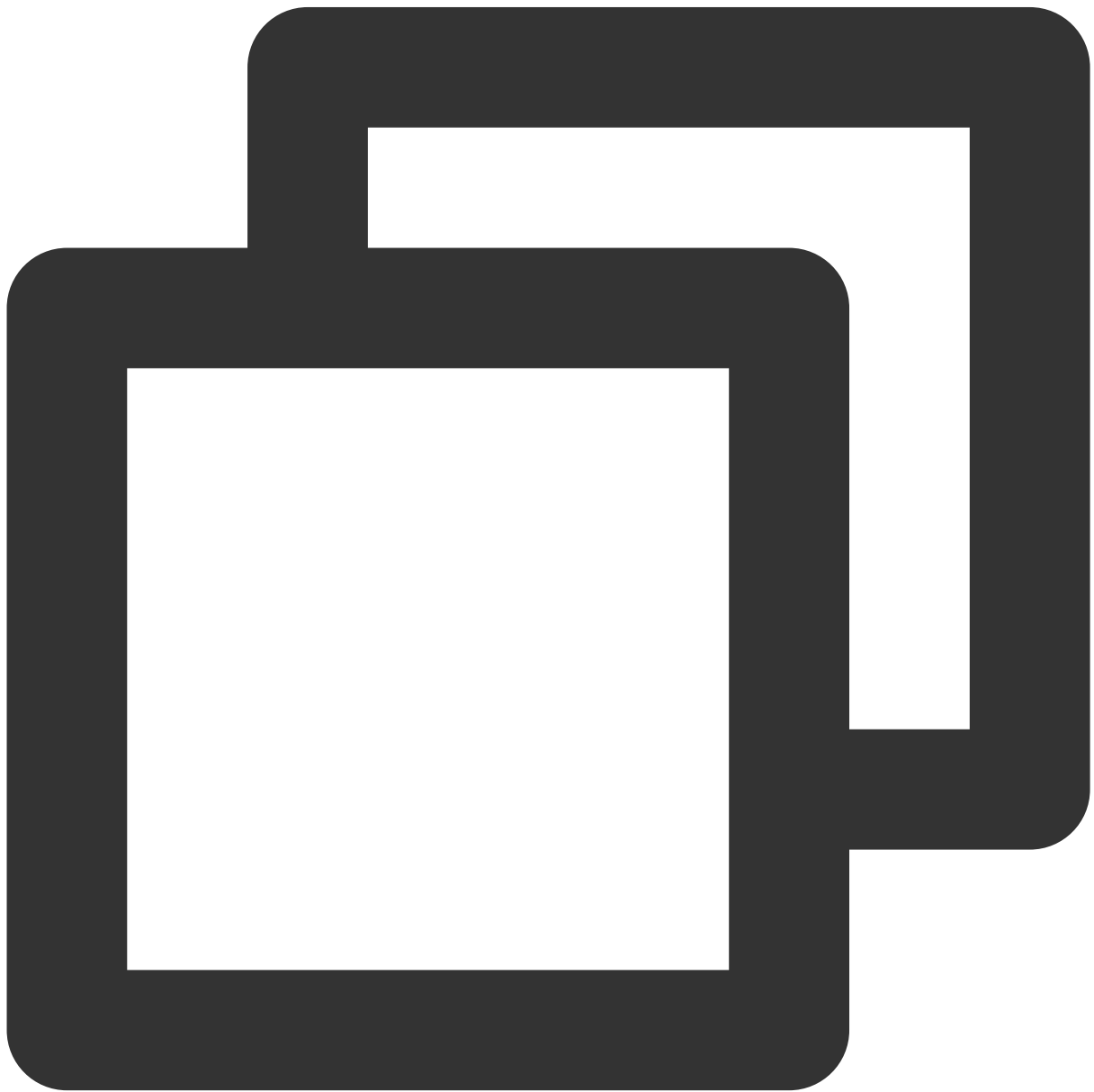
Android

Unity

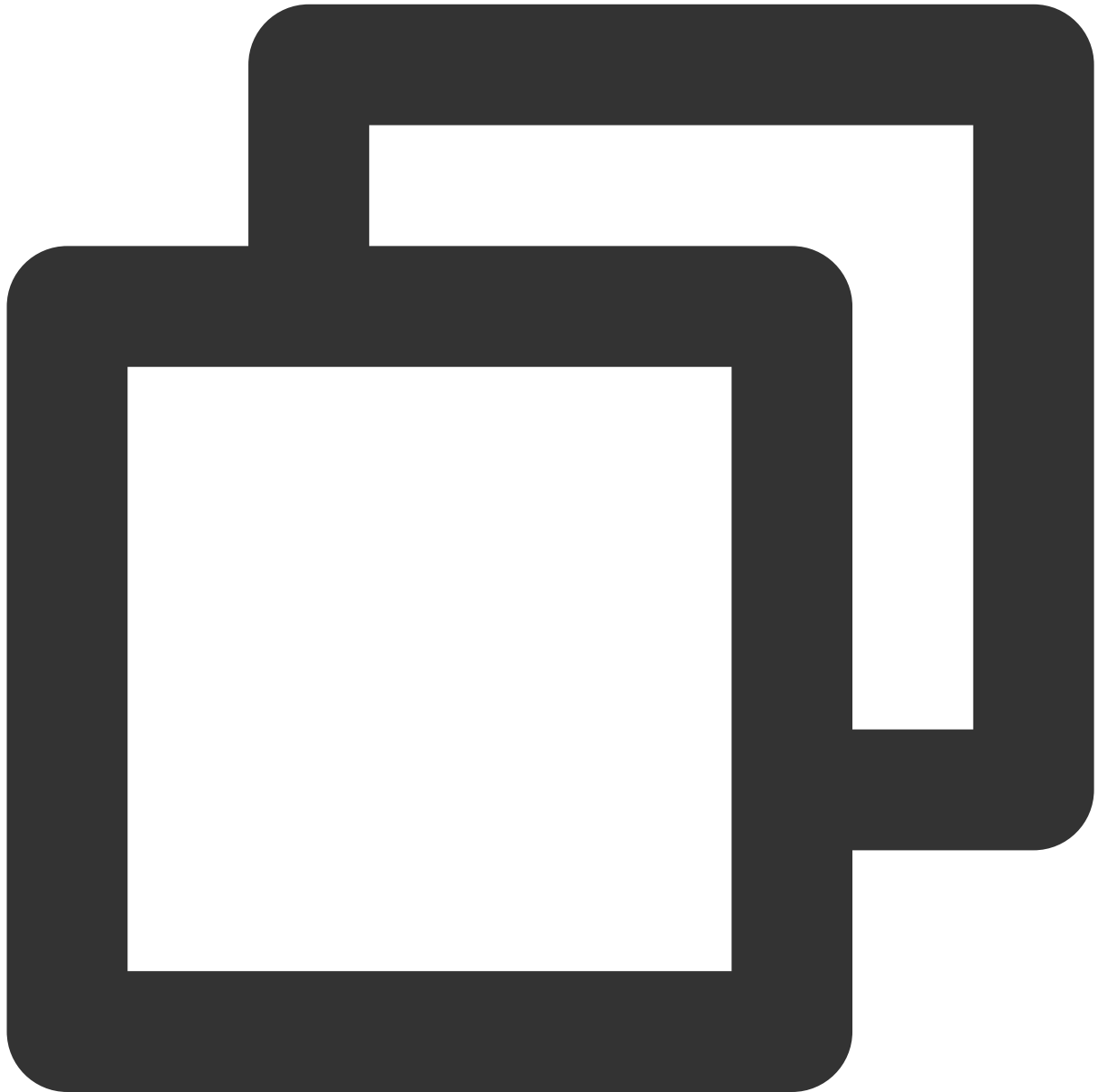


```
-(IBAction)SendCustData:(UIButton*)sender {
    int ret = 0;
    NSString *typeString;
    switch (sender.tag) {
        case 1:
            ret = [[[ITMGContext GetInstance] GetRoom] SendCustomData:[NSData dataW
            typeString = @"sendCustData";
            break;
        case 2:
            ret = [[[ITMGContext GetInstance] GetRoom] StopSendCustomData];
            typeString = @"recvCustData";
```

```
        break;
    default:
        break;
}
if(ret != 0) {
    UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"set fail" message
[alert show];
}
}
```



```
String strData = mEditData.getText().toString();
String repeatCount = mEditRepeatCount.getText().toString();
int nRet = ITMGContext.GetInstance(getActivity()).GetRoom().SendCustomData(strData.
```



```
InputField SendCustom_Count_InputField = transform.Find("inroomPanel/imPanel/SendCu
InputField SendCustom_Data_InputField = transform.Find("inroomPanel/imPanel/SendCus

transform.Find("inroomPanel/imPanel/SendCustom_Btn").GetComponent<Button>().onClick
{
```

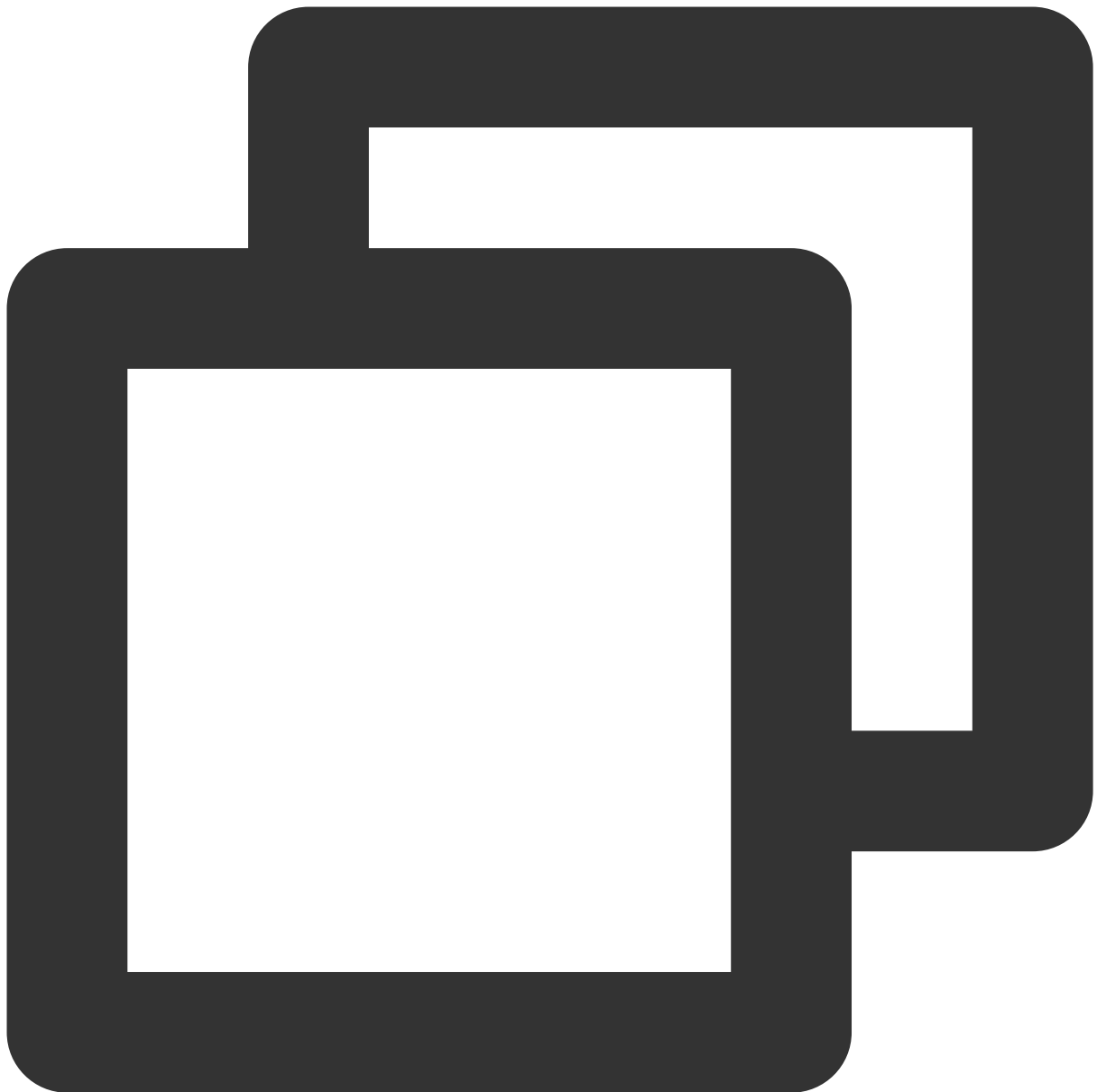
```
        string data = SendCustom_Data_InputField.text;
        string str_count = SendCustom_Count_InputField.text;
        int count = 0;
        if (int.TryParse(str_count, out count)) {
            Debug.Log(data+ count.ToString());
            byte[] byteData = Encoding.Default.GetBytes(data);
            int ret = ITMGContext.GetInstance().GetRoom().SendCustomData(byteData);
            if(ret != 0) {
                ShowWarnning(string.Format("send customdata failed err:{0}",ret));
            }
        }
    });
}
```

コールバック

iOS

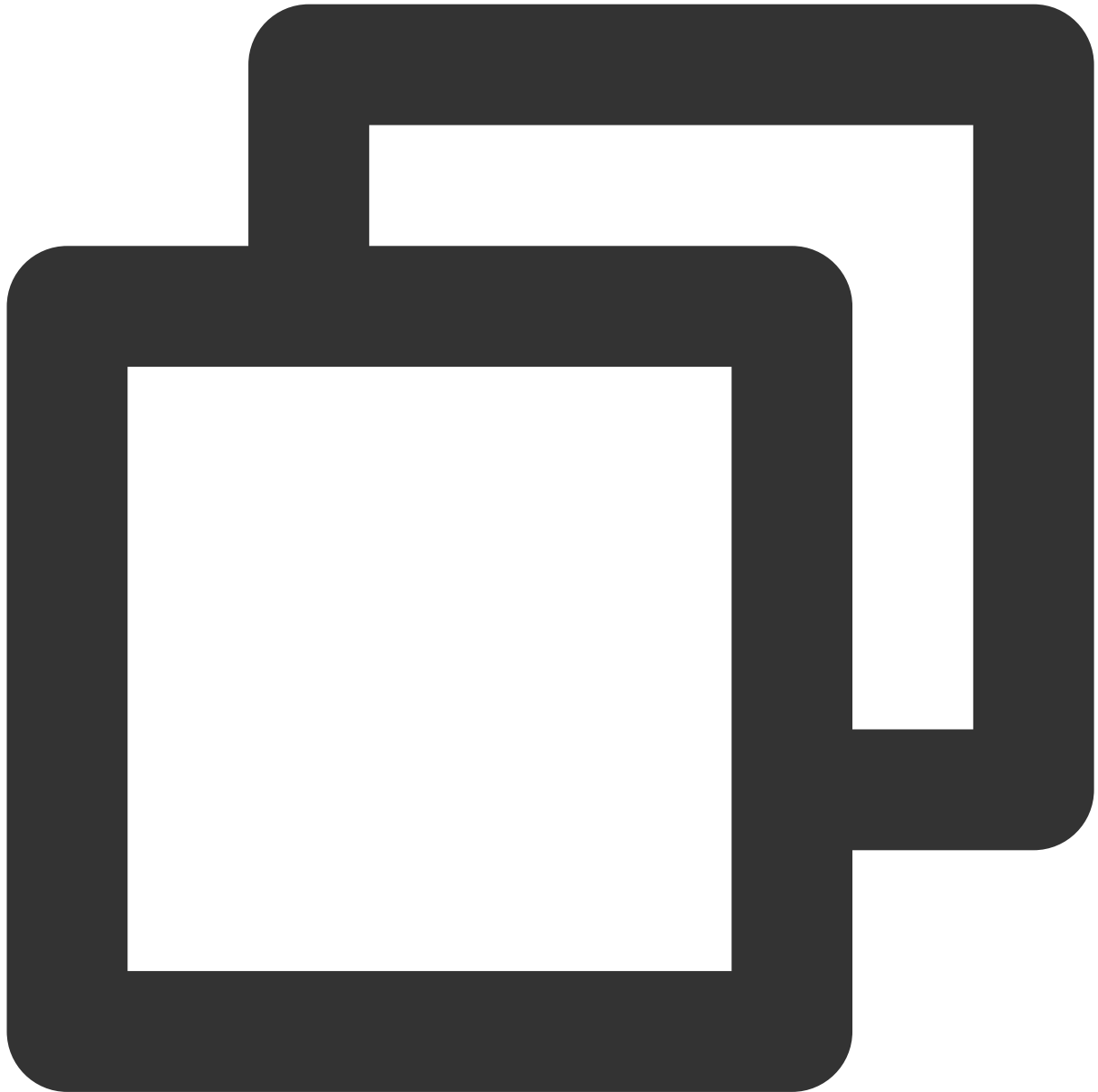
Android

Unity

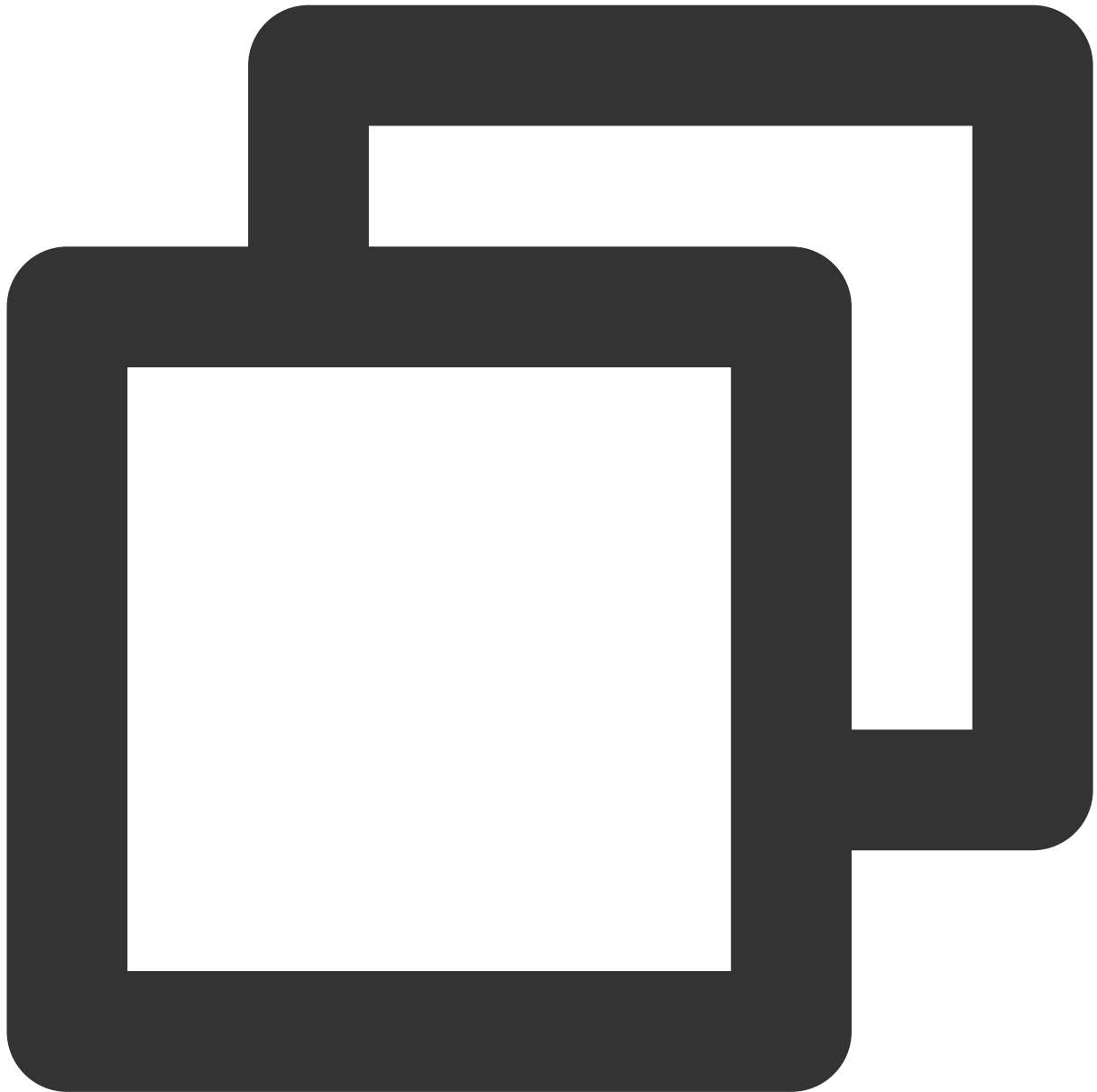


```
-(void)OnEvent:(ITMG_MAIN_EVENT_TYPE)eventType data:(NSDictionary *)data{
    NSString *log = [NSString stringWithFormat:@"OnEvent:%d,data:%@", (int)eventType, data];
    switch (eventType) {
        case ITMG_MAIN_EVENT_TYPE_CUSTOMDATA_UPDATE: {
            if (ITMG_CUSTOMDATA_AV_SUB_EVENT_UPDATE == ((NSNumber *)data[@"sub_type"])->intValue) {
                UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"customdata update" message:log delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil];
                [alert show];
            }
        }
        break;
    }
}
```

```
}
```



```
if (ITMGContext.ITMG_MAIN_EVENT_TYPE.ITMG_MAIN_EVENT_TYPE_CUSTOMDATA_UPDATE == type) {
    int subtype = data.getIntExtra("sub_event", -1);
    if (subtype == 0) {
        String content = data.getStringExtra("content");
        String sender = data.getStringExtra("senderid");
        Toast.makeText(getActivity(), String.format("recv content =%s, from:%s", content, sender), Toast.LENGTH_SHORT).show();
    }
}
```



```
void OnEvent(int eventType,int subEventType,string data)
{
    Debug.Log (data);
    switch (eventType) {
        case (int)ITMG_MAIN_EVENT_TYPE.ITMG_MAIN_EVENT_TYPE_CUSTOMDATA_UPDATE:
        {
            if(subEventType == (int)ITMG_CUSTOMDATA_SUB_EVENT.ITMG_CUSTOMDATA_AV_SUB_E
                _customData = JsonUtility.FromJson<CustomDataInfo>(data);
                ShowWarnning(string.Format("recve customdata {0}  from {1}",_customData.
```

```
    }  
    }  
    break;  
}
```

カスタムメッセージの送信を停止する

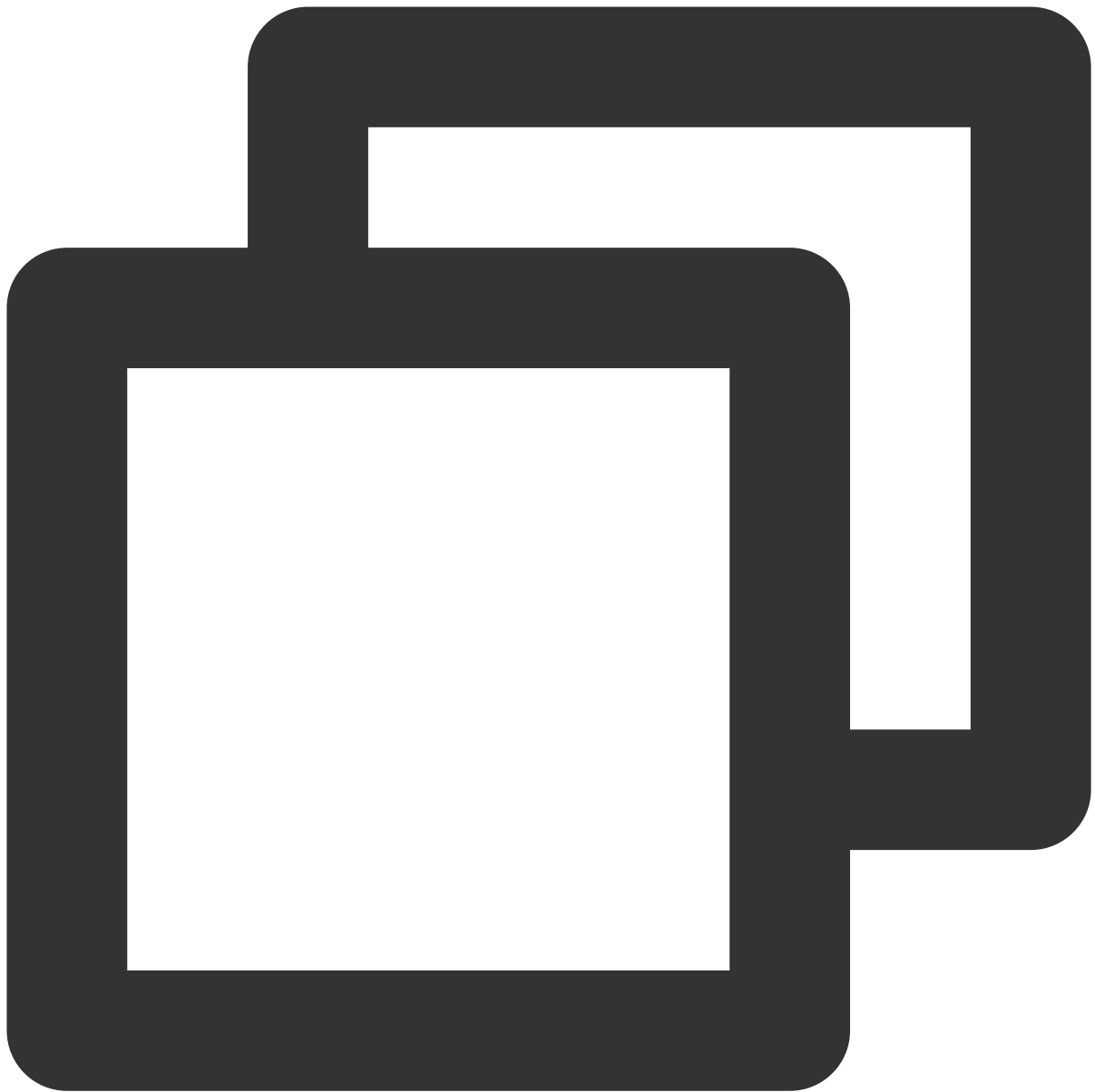
このインターフェースを呼び出すとカスタムメッセージの送信を停止します。

インターフェースのプロトタイプ

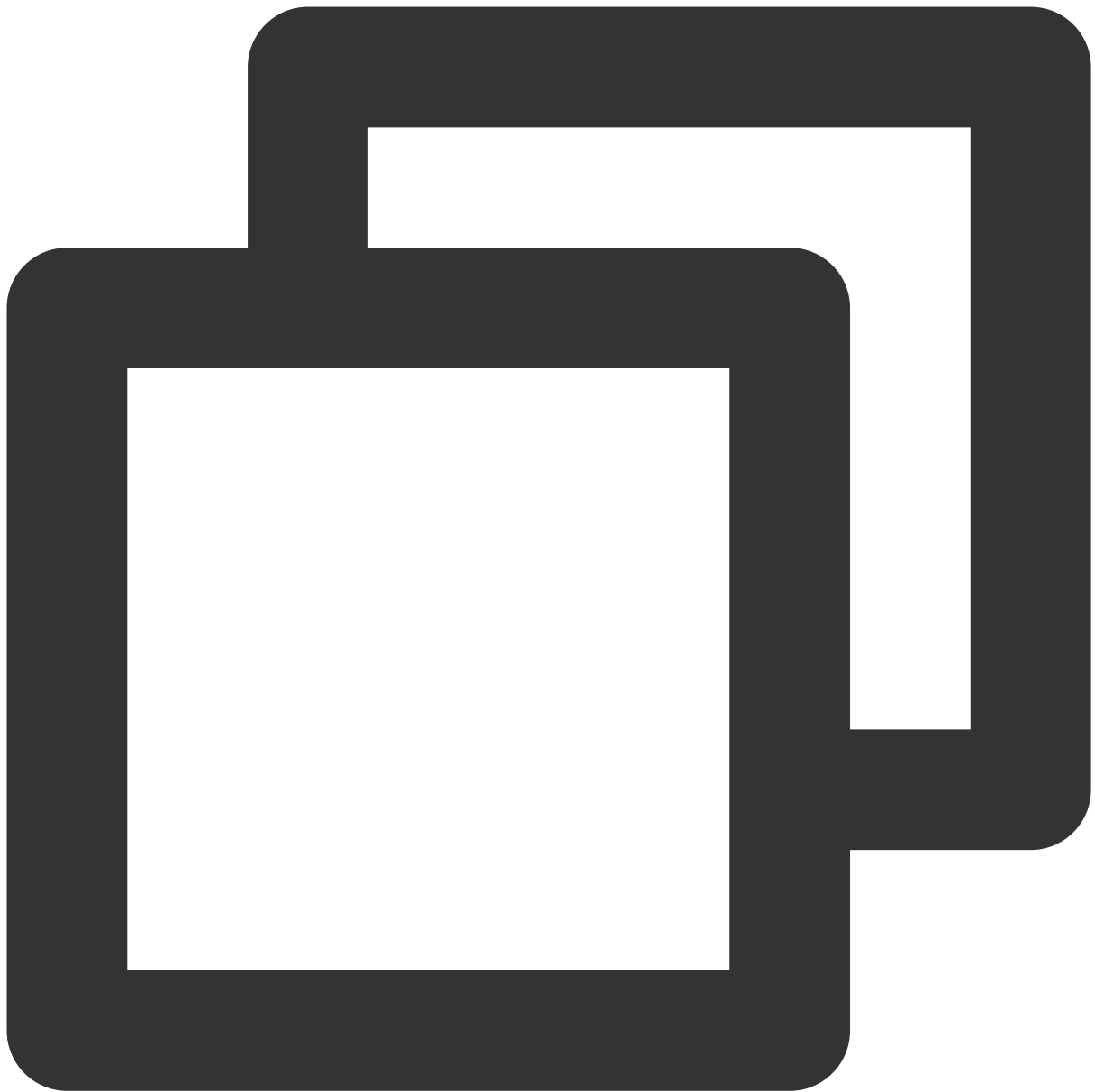
iOS

Android

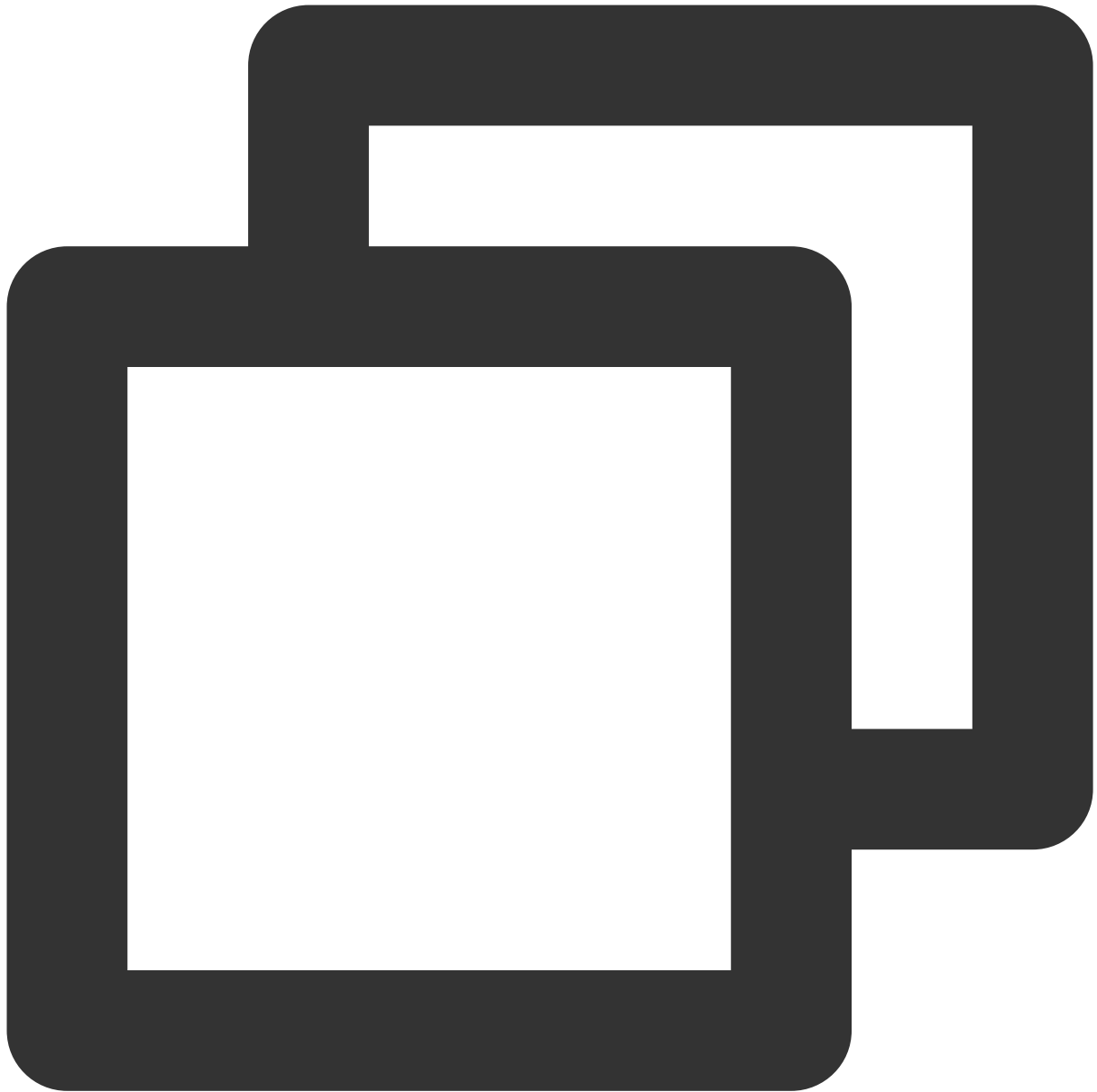
Unity



```
-(int) StopSendCustomData;
```



```
public abstract int StopSendCustomData();
```



```
public abstract int StopSendCustomData();
```

戻り値

インターフェースが1003を返した場合、StopSendCustomDataが操作されたことを示します。SDKはその操作を実行中であり、再度呼び出す必要はありません。

社内ファイアール制限への対応について

最終更新日：：2024-01-18 15:47:47

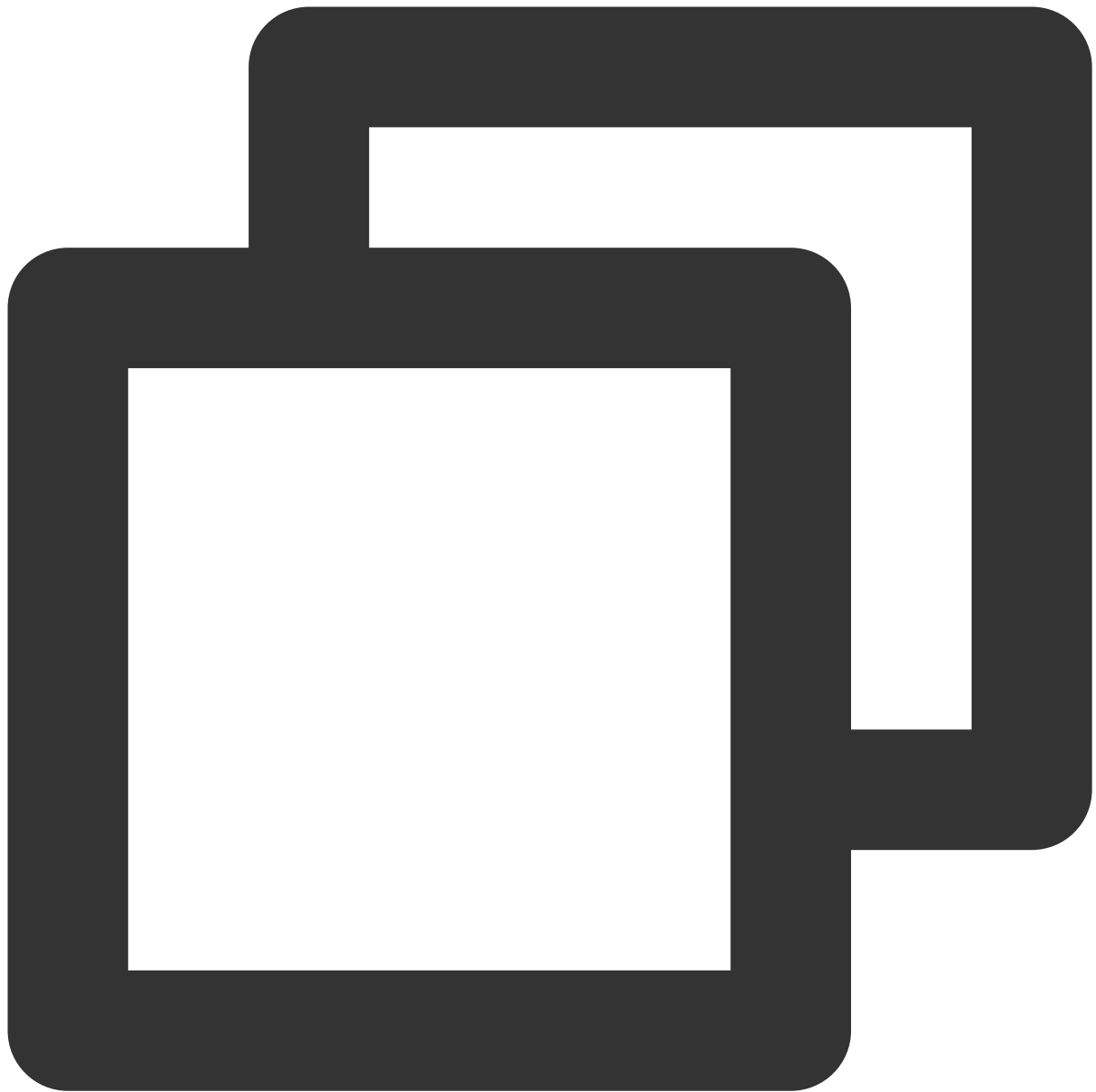
外部ネットワークへのアクセス制限がある場合、ファイアールホワイトリストに追加しないとアクセスできません。具体的なルールは以下のとおりです：

クライアント Native SDK（バージョン2.2以降）

ファイアールポート：

ポートタイプ	ホワイトリスト項目
TCP ポート	443
UDP ポート	8000

ドメイン名ホワイトリスト：



```
tcloud.tim.qq.com  
gmeconf.qcloud.com  
yun.tim.qq.com  
gmeosconf.qcloud.com  
sg.global.gme.qcloud.com
```

ご注意：

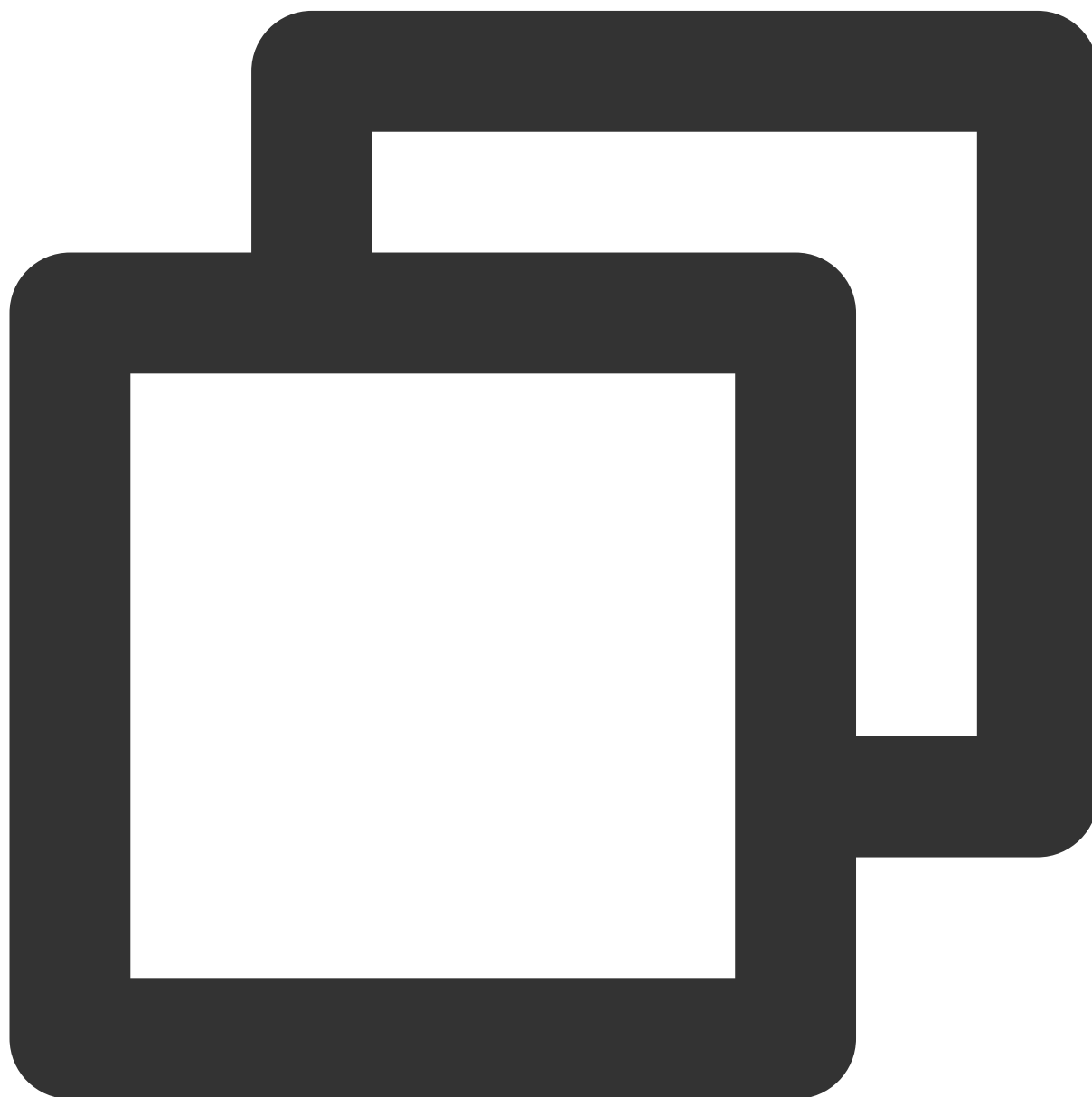
Tencent CloudのサーバーのIPアドレスは、固定IPアドレスではなく動的に更新されるもののため、固定したIPリストを提供することはできません。

Windows XPでGME SDKを使用するには、さらに以下のファイアウォールホワイトリストを追加してください。

ファイアウォールポート：

ポートタイプ	ホワイトリスト項目
TCP ポート	15000

ドメイン名ホワイトリスト：



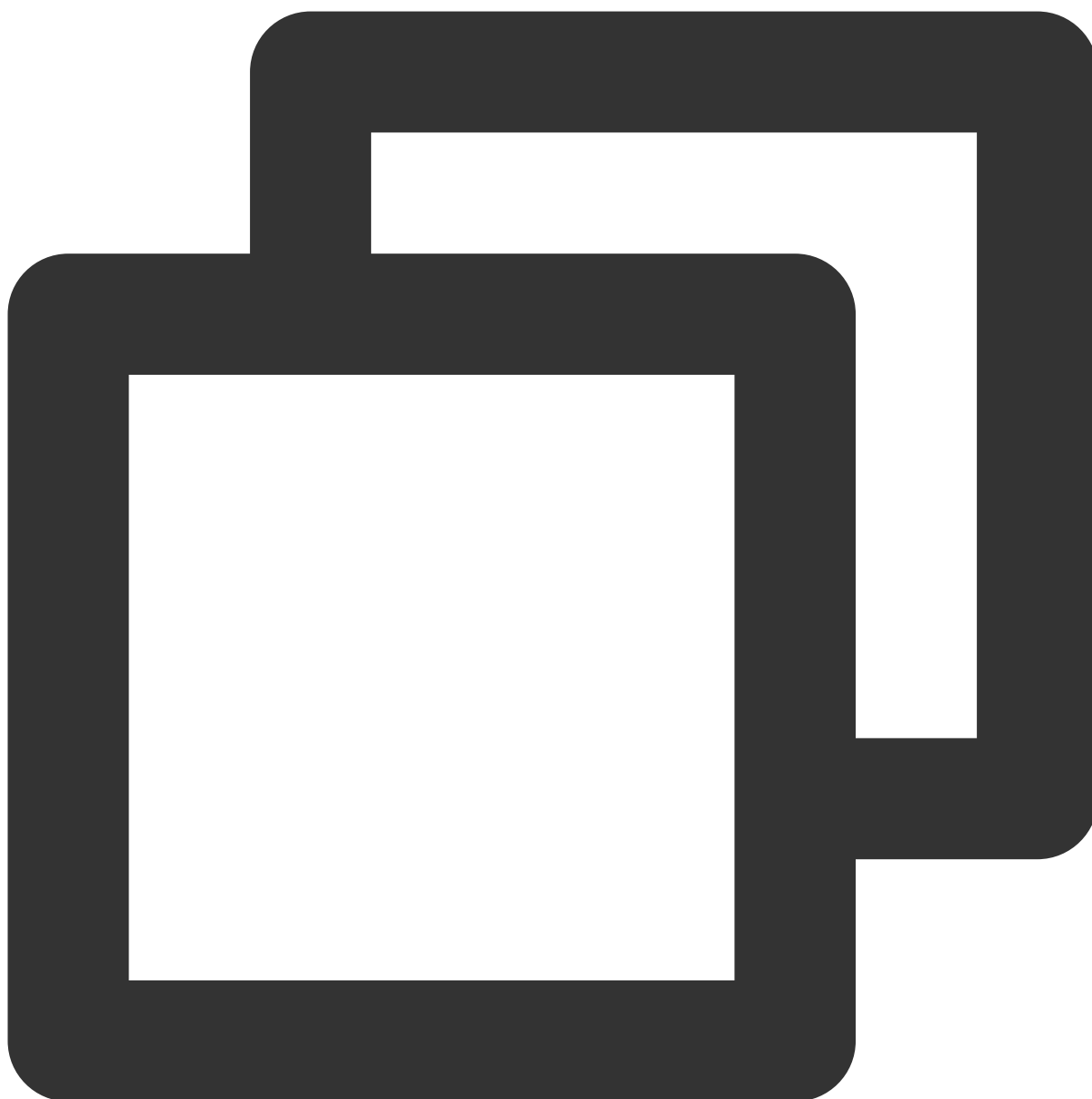
openmsf.3g.qq.com

H5 SDKを使用する場合

ファイアウォールポート：

ポートタイプ	ホワイトリスト項目
TCP ポート	443,8687
UDP ポート	8000 ; 8800 ; 843 ; 443

ドメイン名ホワイトリスト：



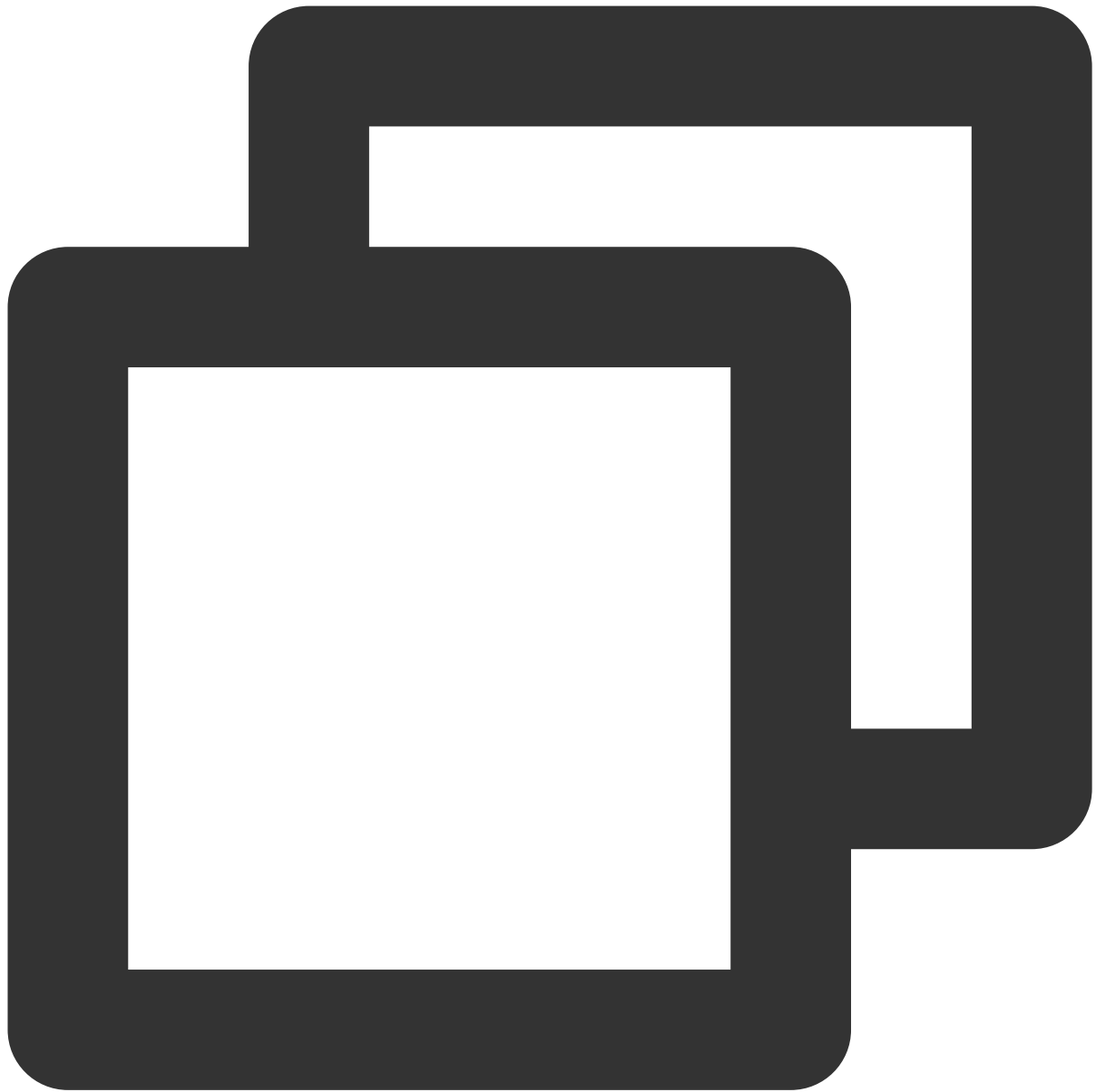
qcloud.rtc.qq.com
rtc.qcloud.qq.com

ボイスメッセージ及びテキスト変換サービスを使用する場合

ファイアウォールポート：

ポートタイプ	ホワイトリスト項目
TCP ポート	80, 443

ドメイン名ホワイトリスト：



gmespeech.qcloud.com
yun.tim.qq.com
gmeconf.qcloud.com

Language Parameter Reference List

最終更新日：：2023-04-04 15:33:25

This document describes the language parameters of GME's speech-to-text, text translation, and text-to-speech services.

Speech-to-text, text translation, and text-to-speech services

Currently, the speech-to-text, text translation, and text-to-speech services support mainstream languages. The text translation API is `TranslateText`, and the text-to-speech API is `TextToSpeech`.

Note

If your application needs to use the text translation or text-to-speech feature, [submit a ticket](#) for application.

Language	Parameter	Description
普通话（中国大陆）	cmn-Hans-CN	Chinese, Mandarin (Simplified, Chinese mainland)
國語（中国台灣）	cmn-Hant-TW	Chinese, Mandarin (Traditional, Taiwan (China))
廣東話（中国香港）	yue-Hant-HK	Chinese, Cantonese (Traditional, Hong Kong (China))
普通話（中国香港）	cmn-Hans-HK	Chinese, Mandarin (Simplified, Hong Kong (China))
Afrikaans (Suid-Afrika)	af-ZA	Afrikaans (South Africa)
አማርኛ (ኢትዮጵያ)	am-ET	Amharic (Ethiopia)
Հայ (Հայաստան)	hy-AM	Armenian (Armenia)
Azərbaycan (Azərbaycan)	az-AZ	Azerbaijani (Azerbaijan)
Bahasa Indonesia (Indonesia)	id-ID	Indonesian (Indonesia)
Bahasa Melayu (Malaysia)	ms-MY	Malay (Malaysia)
বাংলা (বাংলাদেশ)	bn-BD	Bengali (Bangladesh)
বাংলা (ভারত)	bn-IN	Bengali (India)
Català (Espanya)	ca-ES	Catalan (Spain)
Čeština (Česká republika)	cs-CZ	Czech (Czech Republic)

Dansk (Danmark)	da-DK	Danish (Denmark)
Deutsch (Deutschland)	de-DE	German (Germany)
English (Australia)	en-AU	English (Australia)
English (Canada)	en-CA	English (Canada)
English (Ghana)	en-GH	English (Ghana)
English (Great Britain)	en-GB	English (United Kingdom)
English (India)	en-IN	English (India)
English (Ireland)	en-IE	English (Ireland)
English (Kenya)	en-KE	English (Kenya)
English (New Zealand)	en-NZ	English (New Zealand)
English (Nigeria)	en-NG	English (Nigeria)
English (Philippines)	en-PH	English (Philippines)
English (South Africa)	en-ZA	English (South Africa)
English (Tanzania)	en-TZ	English (Tanzania)
English (United States)	en-US	English (United States)
Español (Argentina)	es-AR	Spanish (Argentina)
Español (Bolivia)	es-BO	Spanish (Bolivia)
Español (Chile)	es-CL	Spanish (Chile)
Español (Colombia)	es-CO	Spanish (Colombia)
Español (Costa Rica)	es-CR	Spanish (Costa Rica)
Español (Ecuador)	es-EC	Spanish (Ecuador)
Español (El Salvador)	es-SV	Spanish (El Salvador)
Español (España)	es-ES	Spanish (Spain)
Español (Estados Unidos)	es-US	Spanish (United States)
Español (Guatemala)	es-GT	Spanish (Guatemala)

Español (Honduras)	es-HN	Spanish (Honduras)
Español (México)	es-MX	Spanish (Mexico)
Español (Nicaragua)	es-NI	Spanish (Nicaragua)
Español (Panamá)	es-PA	Spanish (Panama)
Español (Paraguay)	es-PY	Spanish (Paraguay)
Español (Perú)	es-PE	Spanish (Peru)
Español (Puerto Rico)	es-PR	Spanish (Puerto Rico)
Español (República Dominicana)	es-DO	Spanish (Dominican Republic)
Español (Uruguay)	es-UY	Spanish (Uruguay)
Español (Venezuela)	es-VE	Spanish (Venezuela)
Euskara (Espainia)	eu-ES	Basque (Spain)
Filipino (Pilipinas)	fil-PH	Filipino (Philippines)
Français (Canada)	fr-CA	French (Canada)
Français (France)	fr-FR	French (France)
Galego (España)	gl-ES	Galician (Spain)
ქართული (საქართველო)	ka-GE	Georgian (Georgia)
ગુજરાતી (ભારત)	gu-IN	Gujarati (India)
Hrvatski (Hrvatska)	hr-HR	Croatian (Croatia)
IsiZulu (Ningizimu Afrika)	zu-ZA	Zulu (South Africa)
Íslenska (Ísland)	is-IS	Icelandic (Iceland)
Italiano (Italia)	it-IT	Italian (Italy)
Jawa (Indonesia)	jv-ID	Javanese (Indonesia)
()	kn-IN	Kannada (India)
()	km-KH	Khmer (Cambodia)
()	lo-LA	Lao (Laos)

Latviešu (latviešu)	lv-LV	Latvian (Latvia)
Lietuvių (Lietuva)	lt-LT	Lithuanian (Lithuania)
Magyar (Magyarország)	hu-HU	Hungarian (Hungary)
മലയാളം (ഇന്ത്യ)	ml-IN	Malayalam (India)
मराठी (भारत)	mr-IN	Marathi (India)
Nederlands (Nederland)	nl-NL	Dutch (Netherlands)
नेपाली (नेपाल)	ne-NP	Nepali (Nepal)
Norsk bokmål (Norge)	nb-NO	Norwegian Bokmål (Norway)
Polski (Polska)	pl-PL	Polish (Poland)
Português (Brasil)	pt-BR	Portuguese (Brazil)
Português (Portugal)	pt-PT	Portuguese (Portugal)
Română (România)	ro-RO	Romanian (Romania)
සිංහල (ශ්‍රී ලංකාව)	si-LK	Sinhalese (Sri Lanka)
Slovenčina (Slovensko)	sk-SK	Slovak (Slovakia)
Slovenščina (Slovenija)	sl-SI	Slovenian (Slovenia)
Urang (Indonesia)	su-ID	Sundanese (Indonesia)
Swahili (Tanzania)	sw-TZ	Swahili (Tanzania)
Swahili (Kenya)	sw-KE	Swahili (Kenya)
Suomi (Suomi)	fi-FI	Finnish (Finland)
Svenska (Sverige)	sv-SE	Swedish (Sweden)
தமிழ் (இந்தியா)	ta-IN	Tamil (India)
தமிழ் (சிங்கப்பூர்)	ta-SG	Tamil (Singapore)
தமிழ் (இலங்கை)	ta-LK	Tamil (Sri Lanka)
தமிழ் (மலேசியா)	ta-MY	Tamil (Malaysia)
()	te-IN	Telugu (India)

Tiếng Việt (Việt Nam)	vi-VN	Vietnamese (Vietnam)
Türkçe (Türkiye)	tr-TR	Turkish (Turkey)
اردو (پاکستان)	ur-PK	Urdu (Pakistan)
اردو (بھارت)	ur-IN	Urdu (India)
Ελληνικά (Ελλάδα)	el-GR	Greek (Greece)
Български (България)	bg-BG	Bulgarian (Bulgaria)
Русский (Россия)	ru-RU	Russian (Russia)
Српски (Србија)	sr-RS	Serbian (Serbia)
Українська (Україна)	uk-UA	Ukrainian (Ukraine)
עברית(ישראל)	he-IL	Hebrew (Israel)
العربية (إسرائيل)	ar-IL	Arabic (Israel)
العربية (الأردن)	ar-JO	Arabic (Jordan)
العربية (الإمارات)	ar-AE	Arabic (United Arab Emirates)
العربية (البحرين)	ar-BH	Arabic (Bahrain)
العربية (الجزائر)	ar-DZ	Arabic (Algeria)
العربية (السعودية)	ar-SA	Arabic (Saudi Arabia)
العربية (العراق)	ar-IQ	Arabic (Iraq)
العربية (الكويت)	ar-KW	Arabic (Kuwait)
العربية (المغرب)	ar-MA	Arabic (Morocco)
العربية (تونس)	ar-TN	Arabic (Tunisia)
العربية (عمان)	ar-OM	Arabic (Oman)
العربية (فلسطين)	ar-PS	Arabic (Palestine)
العربية (قطر)	ar-QA	Arabic (Qatar)
العربية (لبنان)	ar-LB	Arabic (Lebanon)
العربية (مصر)	ar-EG	Arabic (Egypt)

فارسی (ایران)	fa-IR	Persian (Iran)
हिन्दी (भारत)	hi-IN	Hindi (India)
ไทย (ประเทศไทย)	th-TH	Thai (Thailand)
한국어 (대한민국)	ko-KR	Korean (South Korea)

GMEルーム管理機能の導入

最終更新日：2024-01-18 15:47:47

ご注意：

GME 3.xバージョンは現在、ルーム管理機能をサポートしていません。

ここでは、開発者がルーム管理サービスに素早くアクセスできるように、ルーム管理サービスのユースケースとアクセスの流れについて紹介します。

機能の説明

クライアントルーム管理インターフェースにより、ルームメンバーの管理、ルームメンバーのマイクのオン・オフ管理を簡単に実現することができます。

シナリオ

例えば、人狼ゲームのシナリオでは、ホストとしてEnableMicで他のプレイヤーのマイクオンをコントロールできます。あるプレイヤーが「死亡」し、ルームの音声を聞いたりマイクを操作して話す必要がない場合、ForbidUserOperationインターフェースを介してそのプレイヤーのデバイス操作を禁止します。

前提条件

リアルタイム音声サービスが有効になっていること：[音声サービス有効化ガイド](#)をご参照ください。

GME SDK導入済み：コアインターフェースとリアルタイム音声インターフェースの導入を含みます。詳細については、[Native SDKクイックスタート](#)、[Unity SDKクイックスタート](#)、[Unreal SDKクイックスタート](#)をご参照ください。

説明：

この機能はH5 SDKではサポートされていません。

導入プロセス

クラス名：ITMGRoomManager

GMEルーム管理機能は、ルームに入ってから呼び出し、ルーム内のメンバーの状態のみを変更できます。

すべてのインターフェースの結果は `ITMG_MAIN_EVNET_TYPE_ROOM_MANAGEMENT_OPERATOR` によってコールバックされます。コールバックの詳細については、[コールバック処理](#)をご参照ください。

インターフェースリスト

タイプ	インターフェース
収集制御	EnableMic、EnableAudioCaptureDevice、EnableAudioSend
再生制御	EnableSpeaker、EnableAudioPlayDevice、EnableAudioRecv
機器状態の取得	GetMicState、GetSpeakerState
敏感なインターフェース	ForbidUserOperation

収集管理の関連インターフェース

収集管理インターフェースには、**マイク管理**、**オーディオアップリンク管理**および**収集ハードウェアデバイス管理**が含まれます。その中で、マイク管理は、オーディオアップリンク管理と収集ハードウェアデバイス管理に相当します。

収集管理

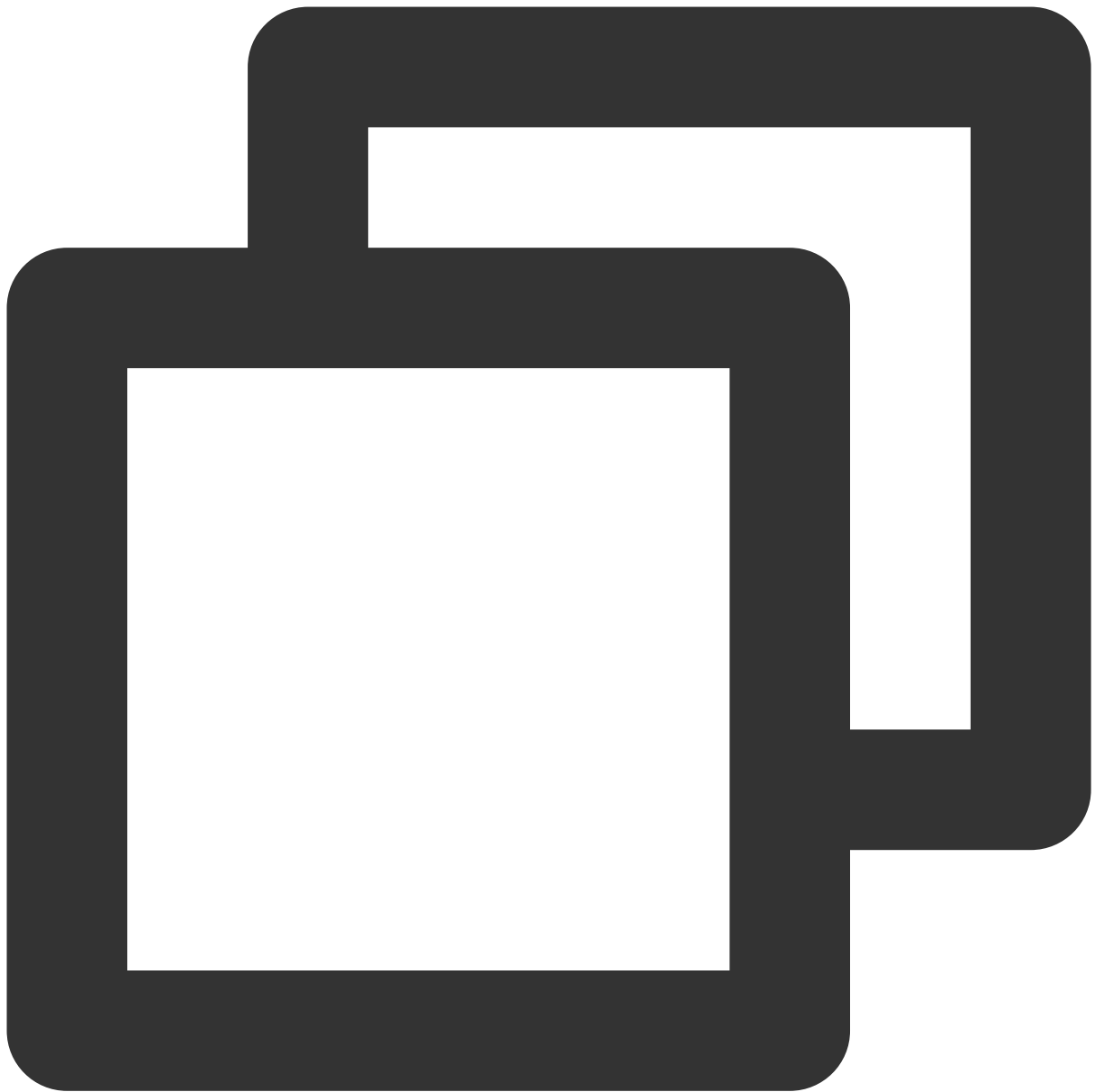
このインターフェースを呼び出して、ルームにいるユーザーのマイクをオンまたはオフにします。呼び出しが成功すると、そのユーザーのマイクはオフまたはオンになります。

EnableMicはEnableAudioSendとEnableAudioCaptureDeviceを同時に呼び出すことに相当します。

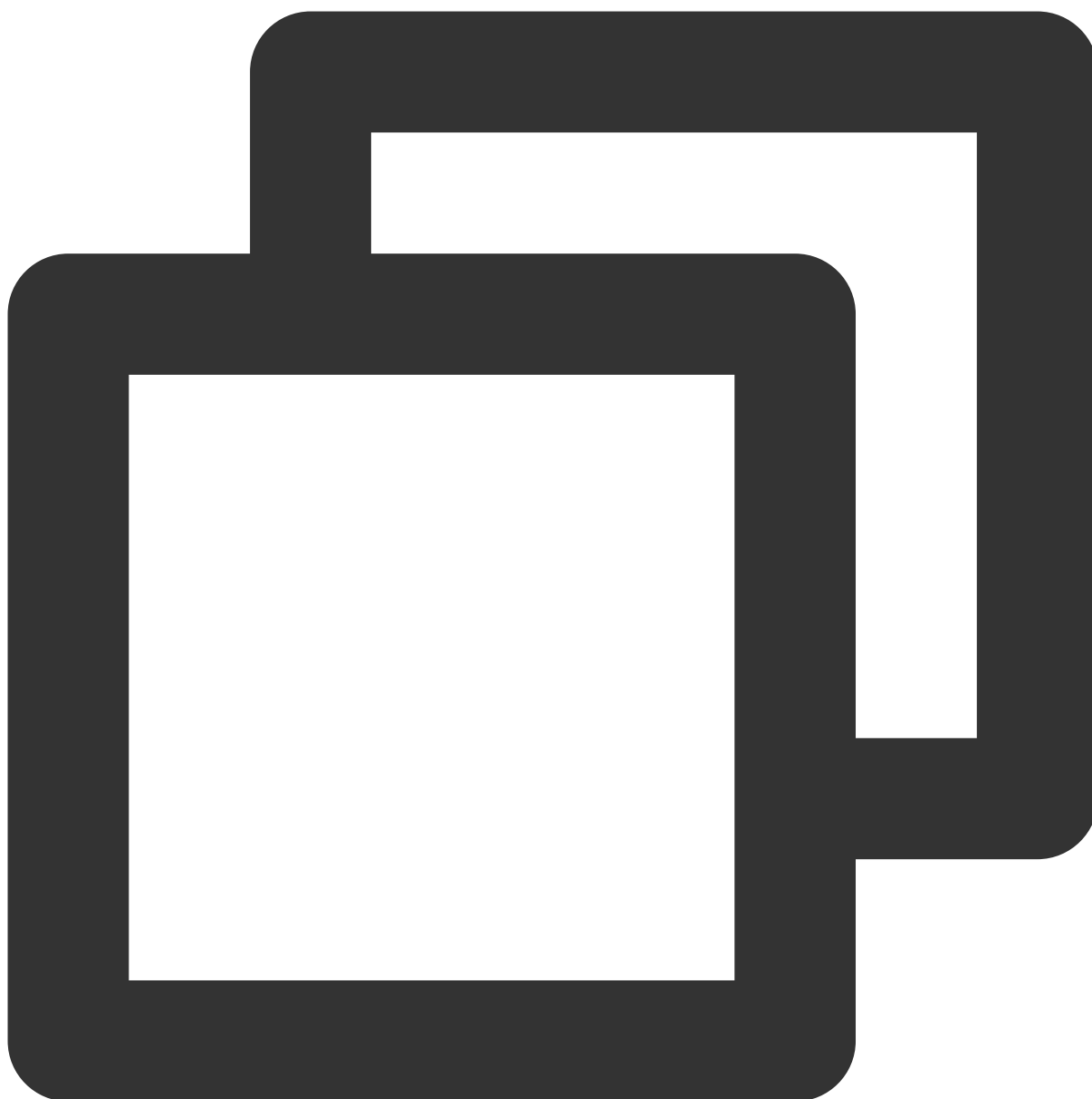
関数のプロトタイプ

Android

iOS



```
public abstract int EnableMic(boolean isEnabled,String receiverID);
```



```
-(QAVResult)EnableMic:(BOOL)enable Receiver:(NSString *)receiverID;
```

パラメータ	タイプ	意味
enable	BOOL	YES：特定のユーザーのマイクをオンにします。NO：特定のユーザーのマイクをオフにします
receiverID	NSString*	ターゲットユーザーOpenIdを入力します

コールバック

コールバックパラメータはITMG_ROOM_MANAGEMENT_MIC_OP。

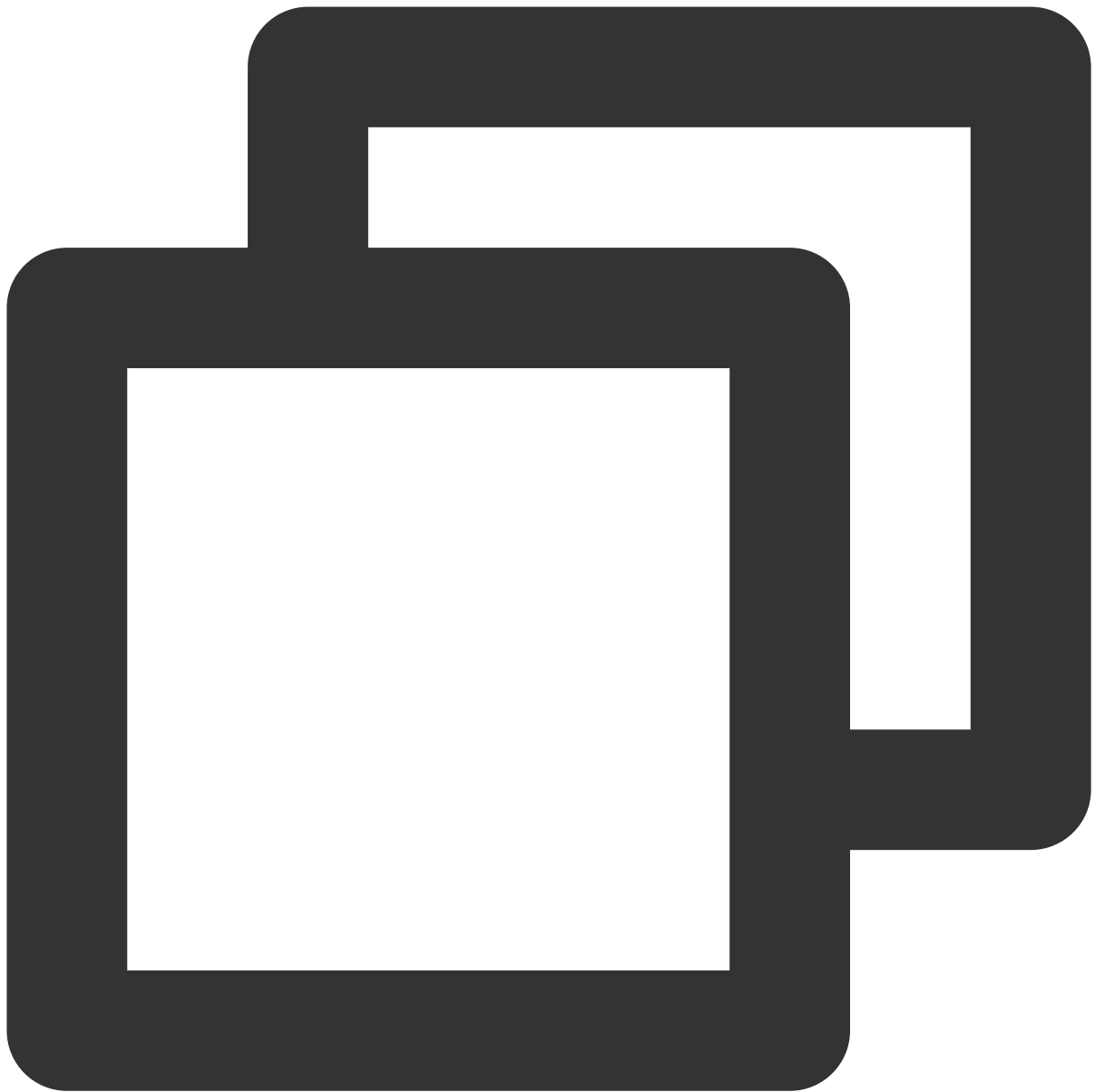
オーディオストリーム送信管理

このインターフェースを呼び出して、ルーム内のユーザーのオーディオ上りをオンまたはオフにします。呼び出しが成功すると、そのユーザーのオーディオ上りはオフまたはオンになりますが、マイクの収集には影響しません。

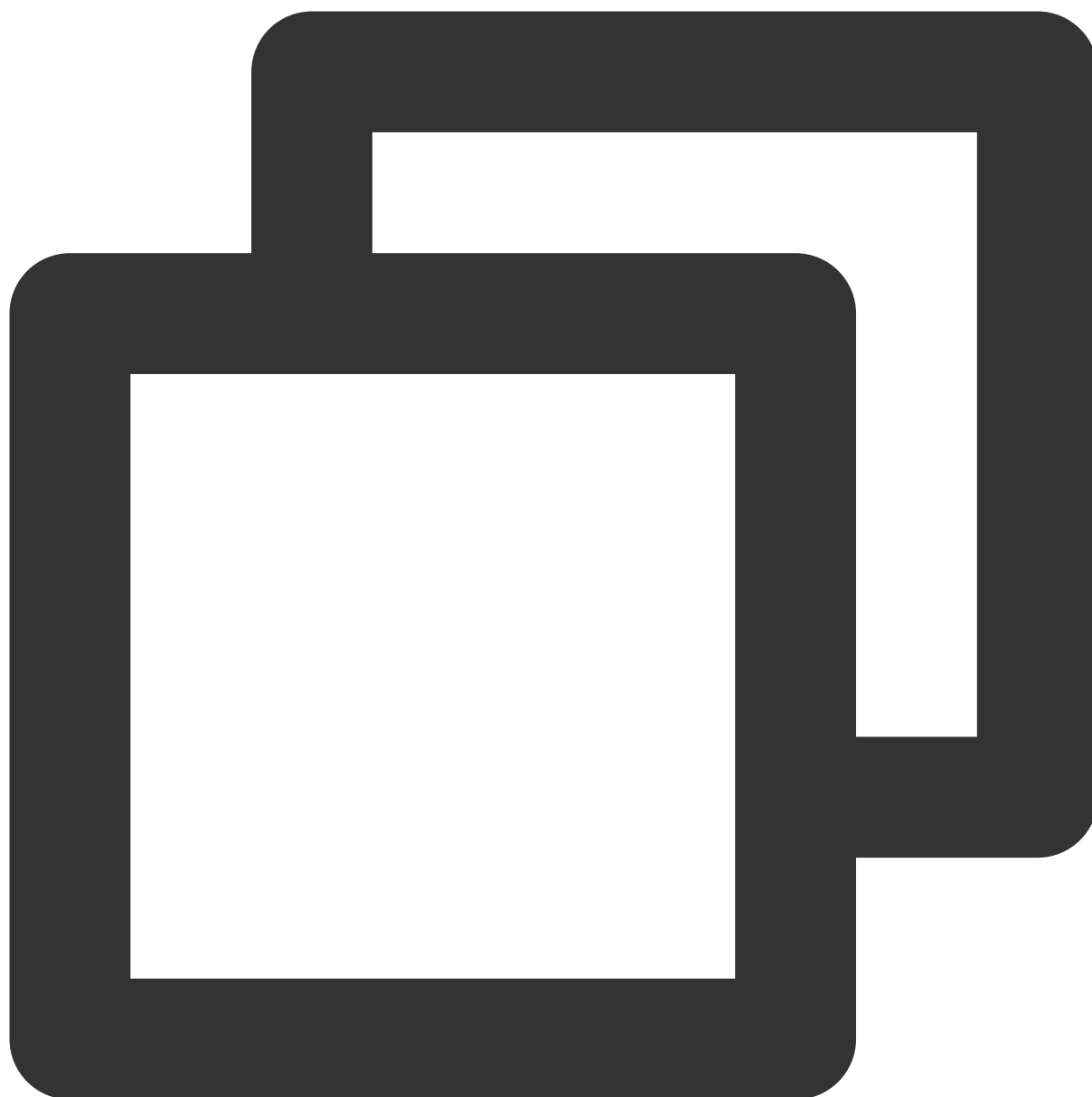
関数のプロトタイプ

Android

iOS



```
public abstract int EnableAudioSend(boolean isEnabled,String receiverID);
```



```
-(QAVResult)EnableAudioSend:(BOOL)enable Receiver:(NSString *)receiverID;
```

パラメータ	タイプ	意味
enable	BOOL	YES：特定のユーザーのアップリンクをオンにします。NO：特定のユーザーのアップリンクをオフにします
receiverID	NSString*	ターゲットユーザーOpenIdを入力します

コールバック

コールバックパラメータはITMG_ROOM_MANAGEMENT_AUDIO_SEND_OP。

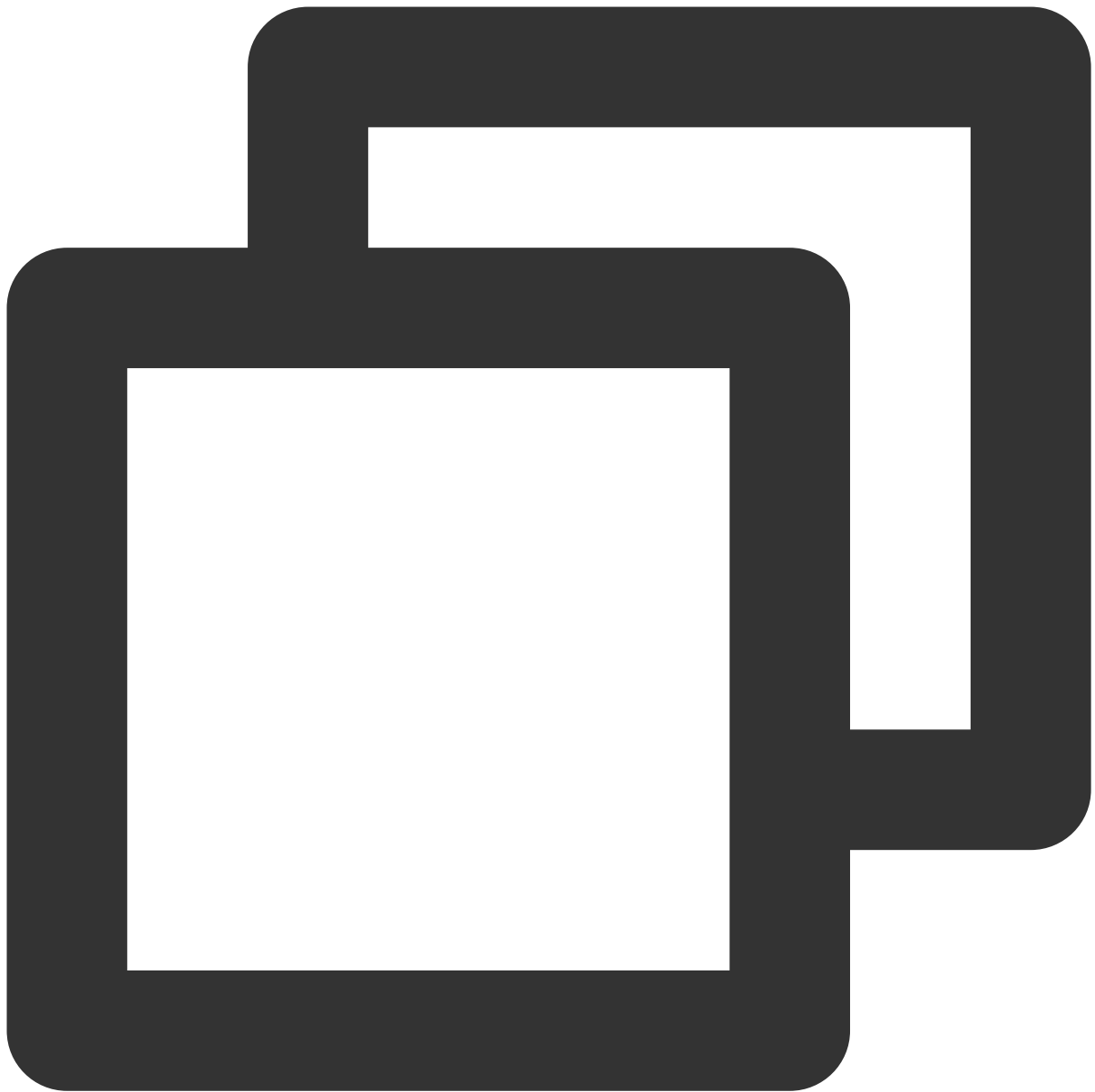
オーディオ収集ハードウェア管理

このインターフェースを呼び出して、ルーム内のユーザーのオーディオ収集ハードウェアデバイスをオンまたはオフにします。呼び出しが成功すると、そのユーザーのオーディオ収集ハードウェアデバイスはオフまたはオンになりますが、上りには影響しません。

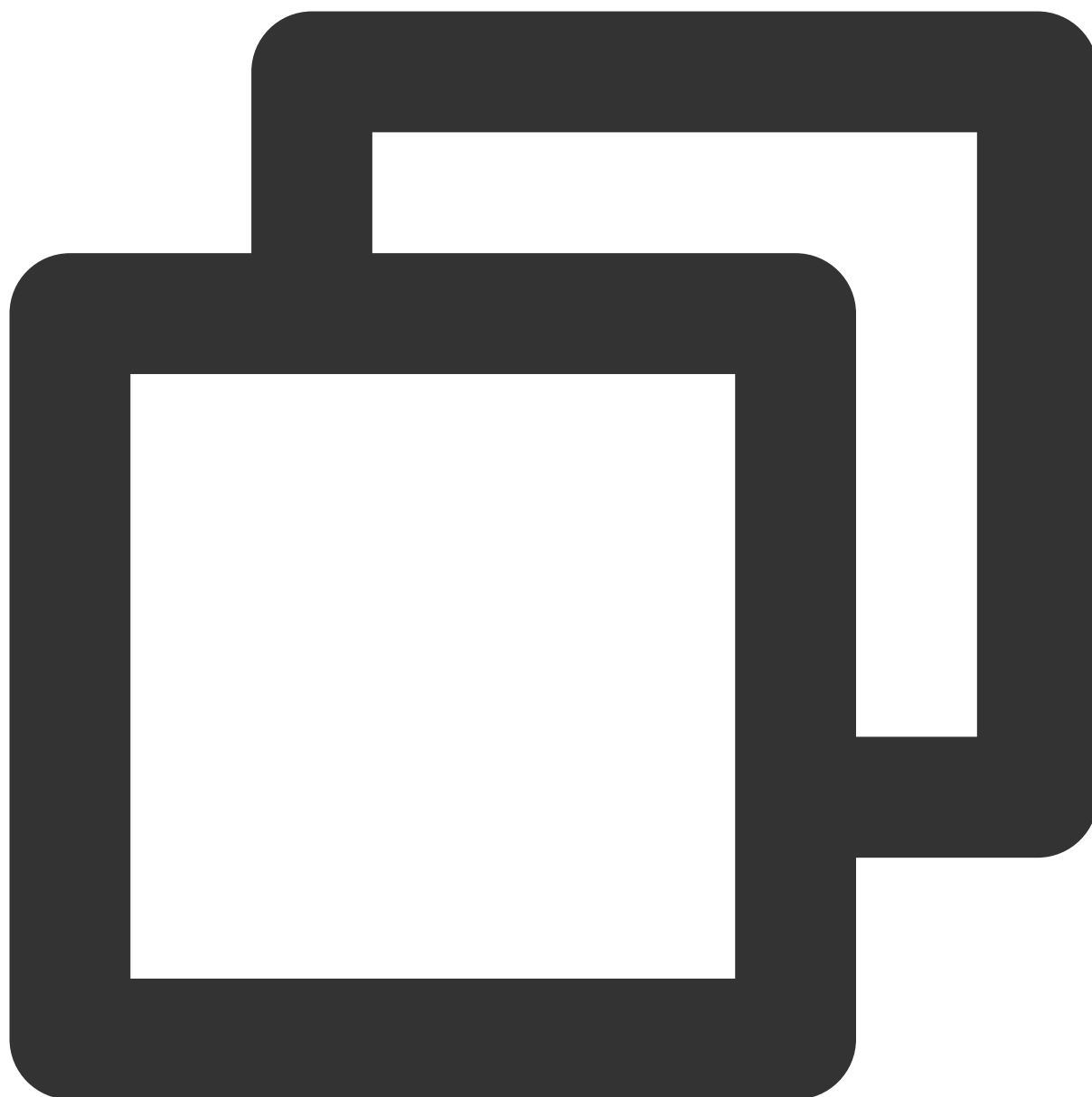
関数のプロトタイプ

Android

iOS



```
public abstract int EnableAudioCaptureDevice(boolean isEnabled,String receiverID);
```



```
-(QAVResult)EnableAudioCaptureDevice:(BOOL)enabled Receiver:(NSString *)receiverID;
```

パラメータ	タイプ	意味
enable	BOOL	YES：特定のユーザーのオーディオ収集ハードウェアデバイスをオンにします。NO：特定のユーザーのオーディオ収集ハードウェアデバイスをオフにします
receiverID	NSString*	ターゲットユーザーOpenIdを入力します

コールバック

コールバックパラメータはITMG_ROOM_MANAGEMENT_CAPTURE_OP。

再生管理の関連インターフェース

再生管理インターフェースには、**スピーカー管理**、**オーディオダウンストリーム管理**および**再生ハードウェアデバイス管理**が含まれます。その中で、スピーカー管理は、オーディオダウンリンク管理と再生ハードウェアデバイス管理に相当します。

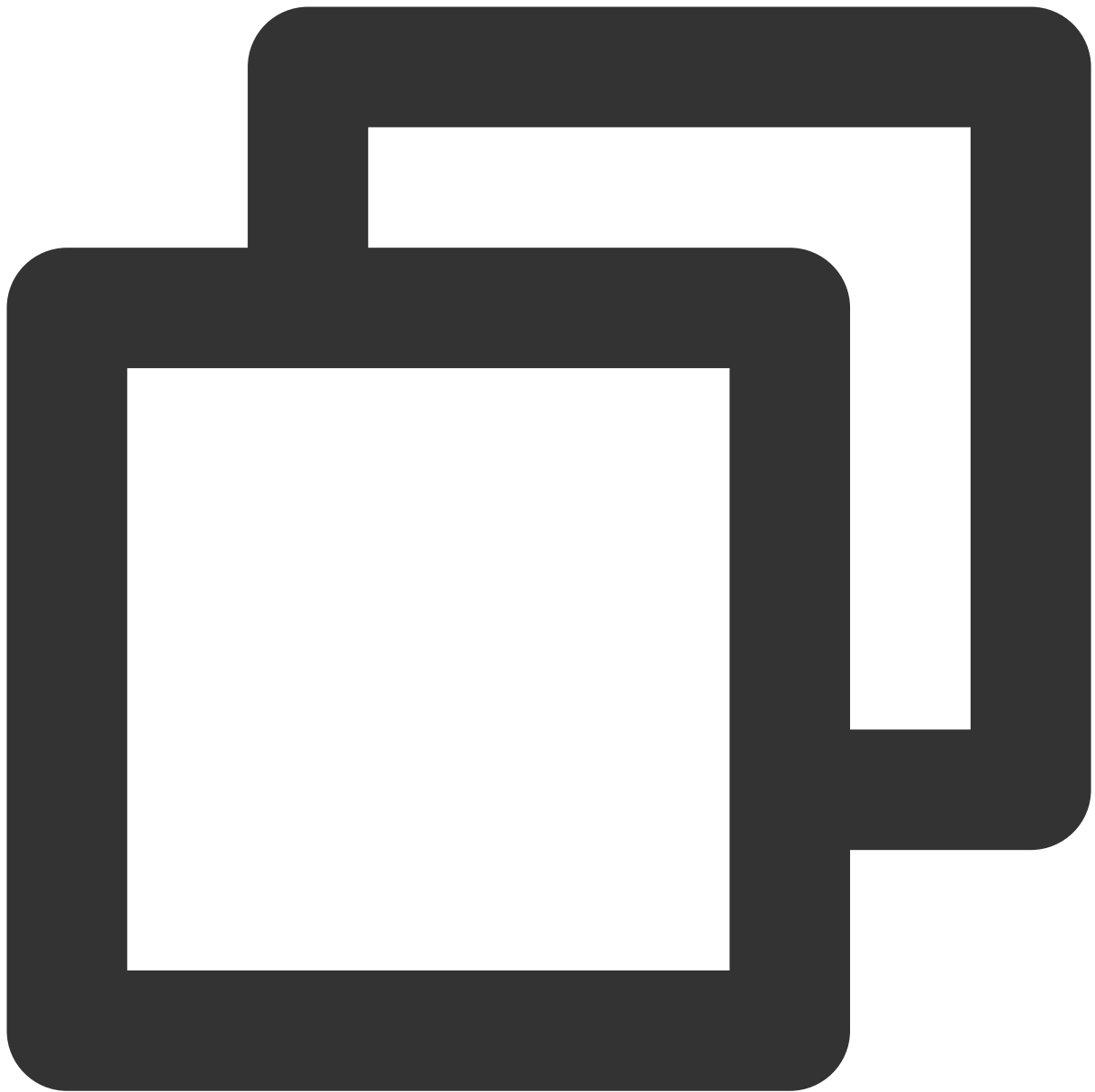
再生管理

このインターフェースを呼び出して、ルーム内のユーザーのスピーカーをオンまたはオフにします。呼び出しが成功すると、そのユーザーのスピーカーがオフまたはオンになり、室内のオーディオ音が聞こえるようになります。**EnableSpeaker**は**EnableAudioRecv**と**EnableAudioPlayDevice**を同時に呼び出すことに相当します。

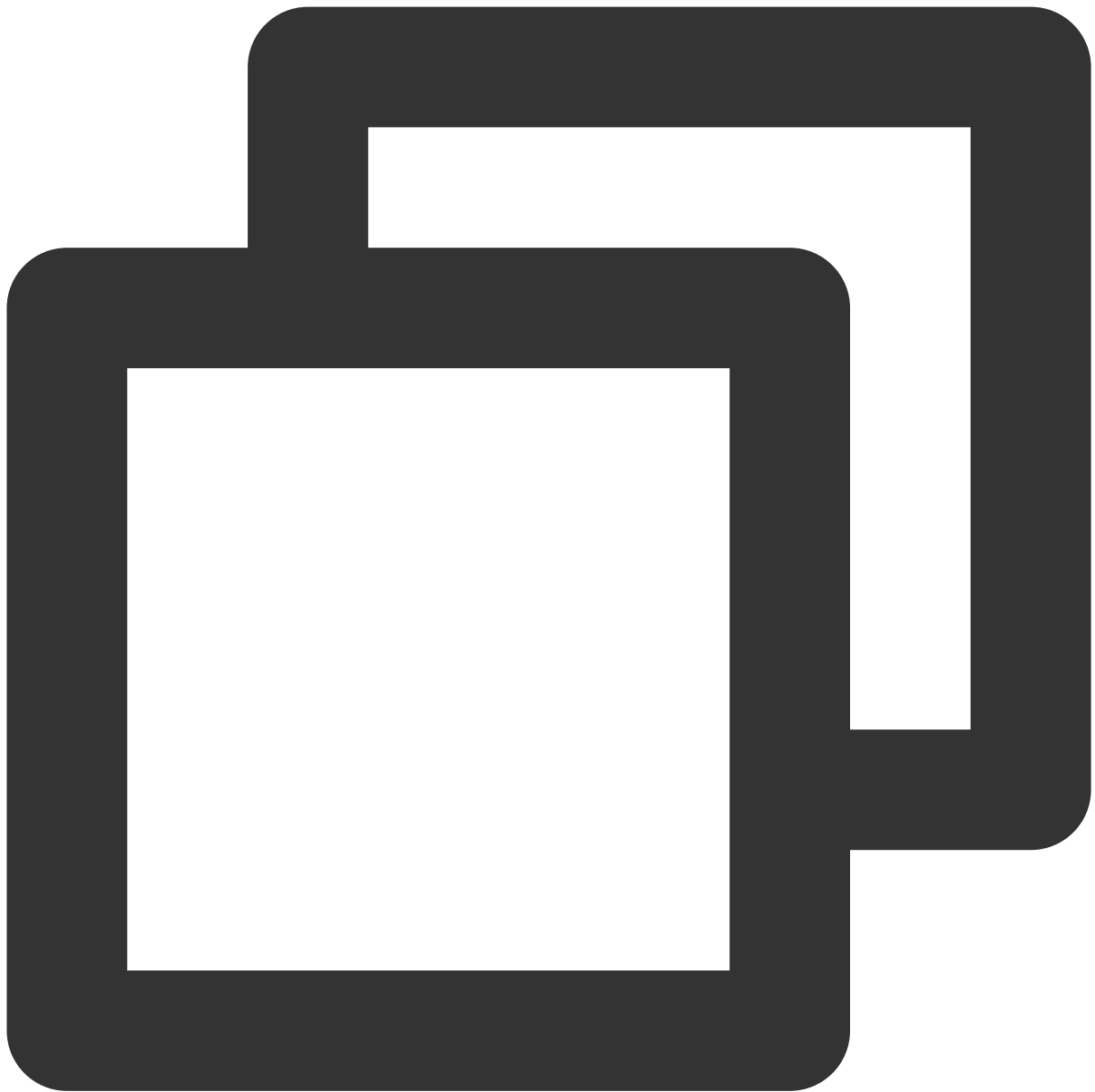
関数のプロトタイプ

Android

iOS



```
public abstract int EnableSpeaker(boolean isEnabled,String receiverID);
```



```
-(QAVResult)EnableSpeaker:(BOOL)enable Receiver:(NSString *)receiverID;
```

パラメータ	タイプ	意味
enable	BOOL	YES：特定のユーザーのスピーカーをオンにします。NO：特定のユーザーのスピーカーをオフにします
receiverID	NSString*	ターゲットユーザーOpenIdを入力します

コールバック

コールバックパラメータはITMG_ROOM_MANAGEMENT_SPEAKER_OP。

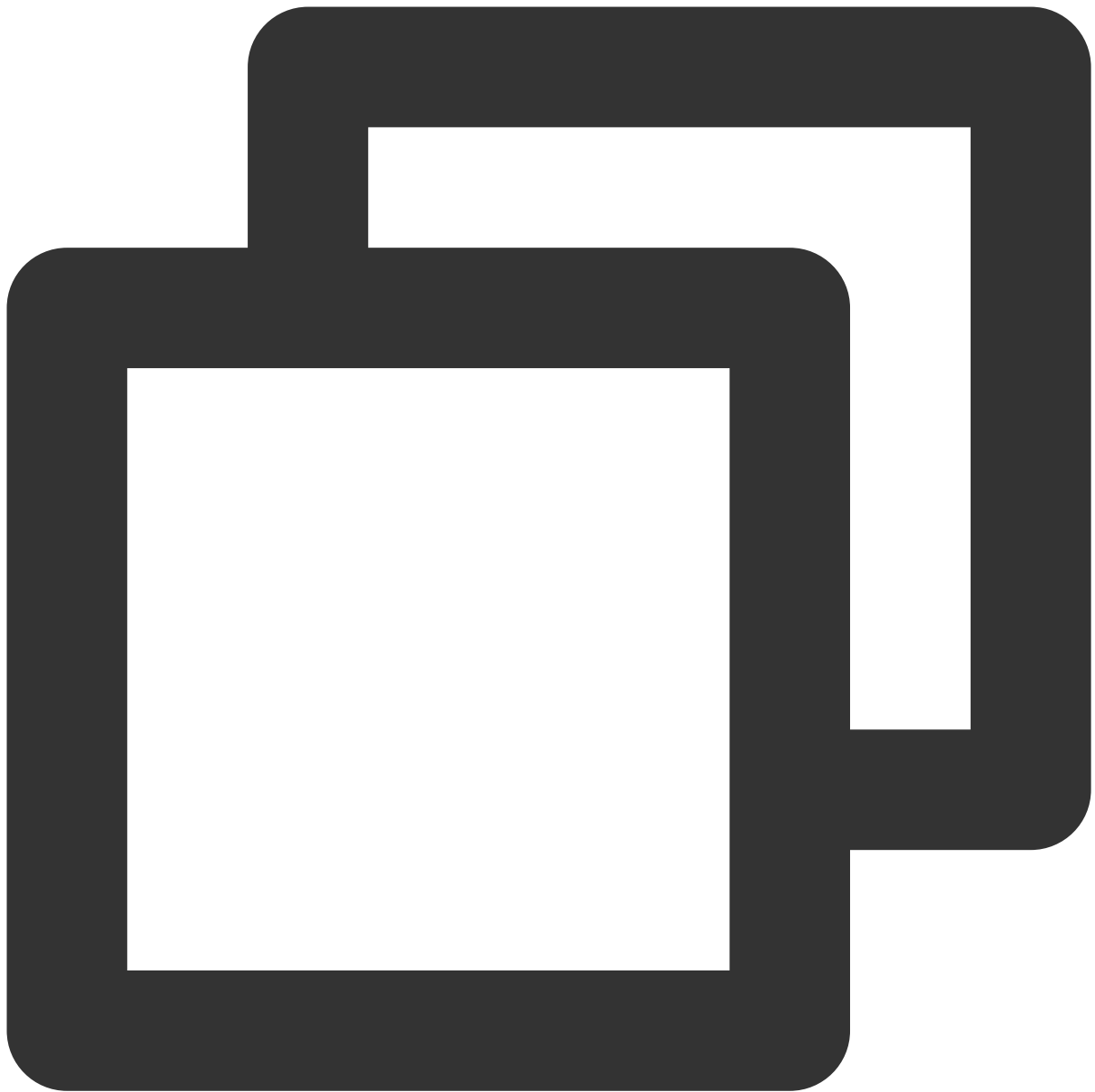
オーディオストリーム受信管理

このインターフェースを呼び出して、ルーム内のユーザーのオーディオ下りをオンまたはオフにします。呼び出しが成功すると、そのユーザーのオーディオ下りはオフまたはオンになりますが、再生デバイスには影響しません。

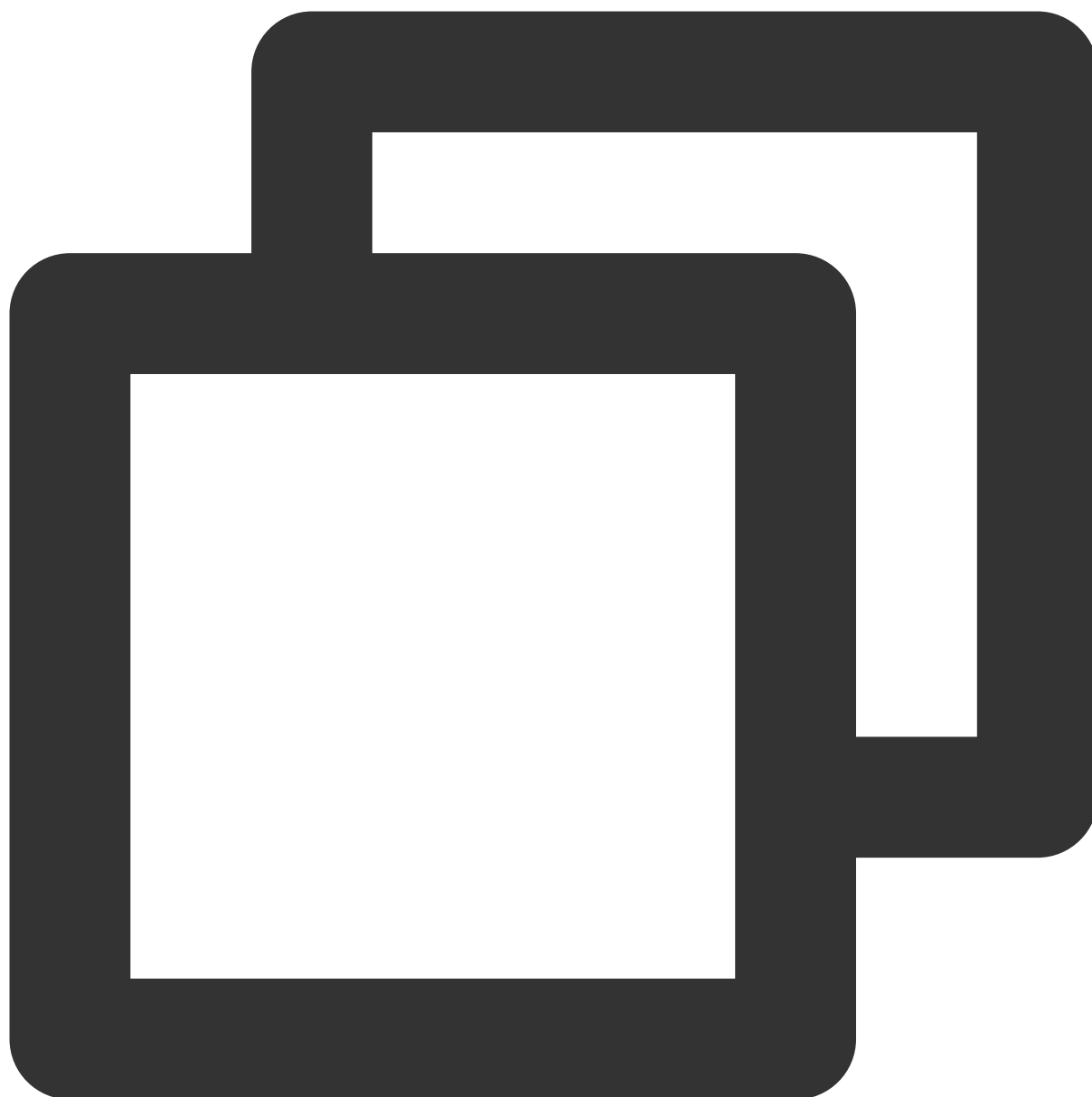
関数のプロトタイプ

Android

iOS



```
public abstract int EnableAudioRecv(boolean isEnabled,String receiverID);
```



```
-(QAVResult)EnableAudioRecv:(BOOL)enabled Receiver:(NSString *)receiverID;
```

パラメータ	タイプ	意味
enable	BOOL	YES：特定のユーザーのダウンリンクをオンにします。NO：特定のユーザーのダウンリンクをオフにします
receiverID	NSString*	ターゲットユーザーOpenIdを入力します

コールバック

コールバックパラメータはITMG_ROOM_MANAGEMENT_AUDIO_REC_OP。

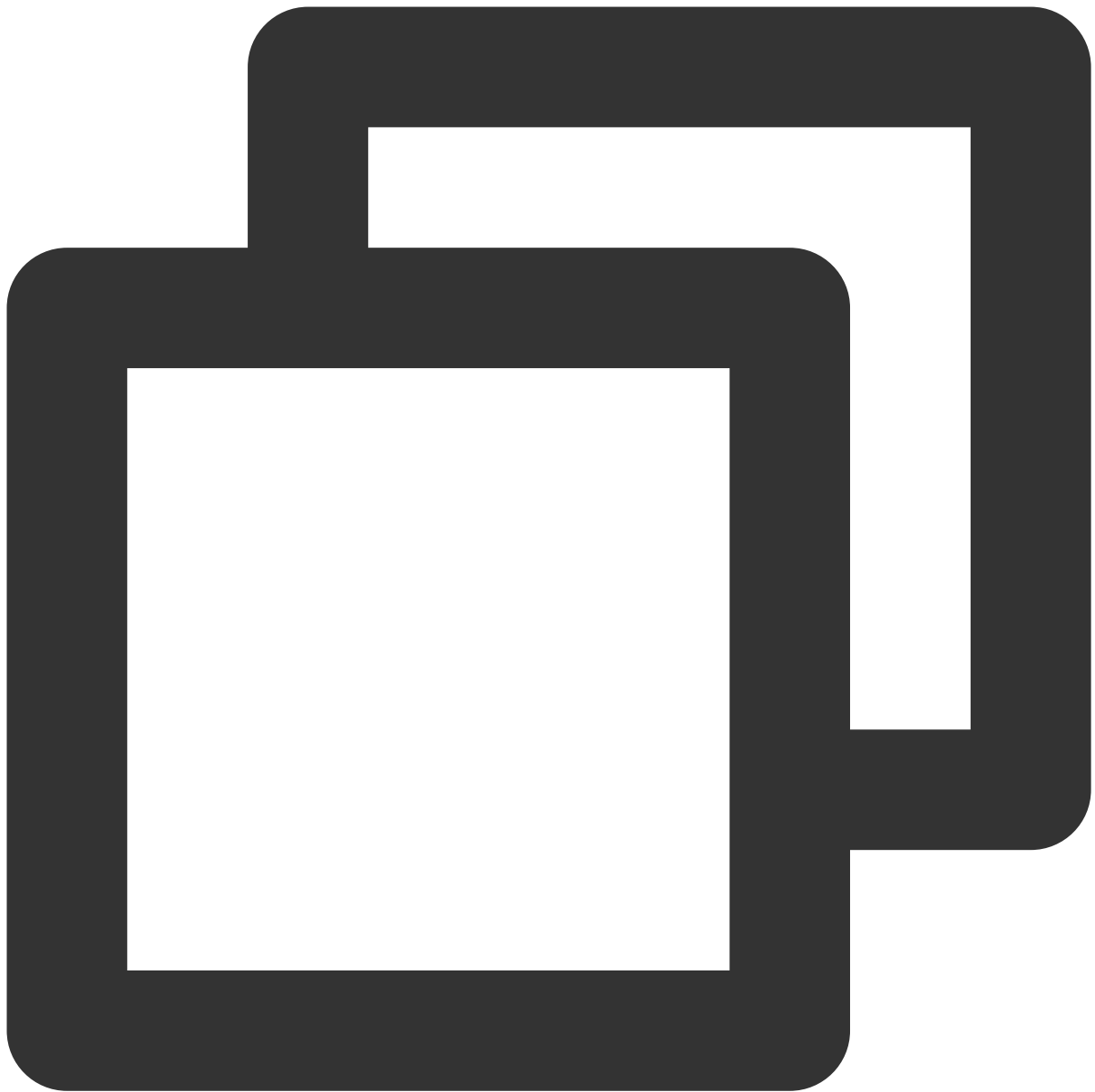
オーディオ再生ハードウェア管理

このインターフェースを呼び出して、ルーム内のユーザーのオーディオ再生ハードウェアデバイスをオンまたはオフにします。呼び出しが成功すると、そのユーザーのオーディオ再生ハードウェアデバイスがオフまたはオンになりますが、下りには影響しません。

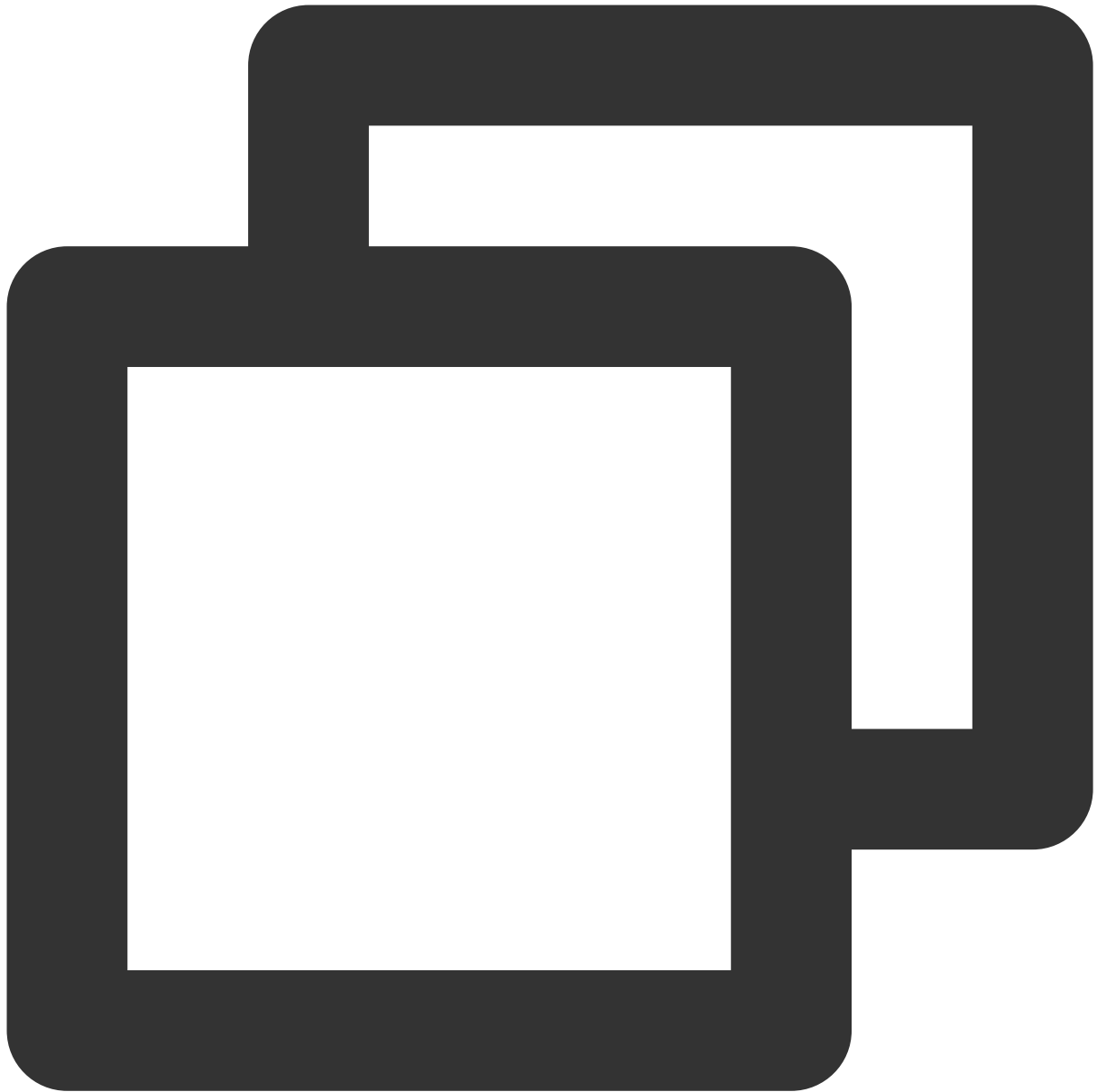
関数のプロトタイプ

Android

iOS



```
public abstract int EnableAudioPlayDevice(boolean isEnabled,String receiverID);
```



```
-(QAVResult)EnableAudioPlayDevice:(BOOL)enabled Receiver:(NSString *)receiverID;
```

パラメータ	タイプ	意味
enable	BOOL	YES：特定のユーザーのオーディオ再生ハードウェアデバイスをオンにします。NO：特定のユーザーのオーディオ再生ハードウェアデバイスをオフにします
receiverID	NSString*	ターゲットユーザーOpenIdを入力します

コールバック

コールバックパラメータはITMG_ROOM_MANAGEMENT_PLAY_OP。

メンバー状態取得インターフェース

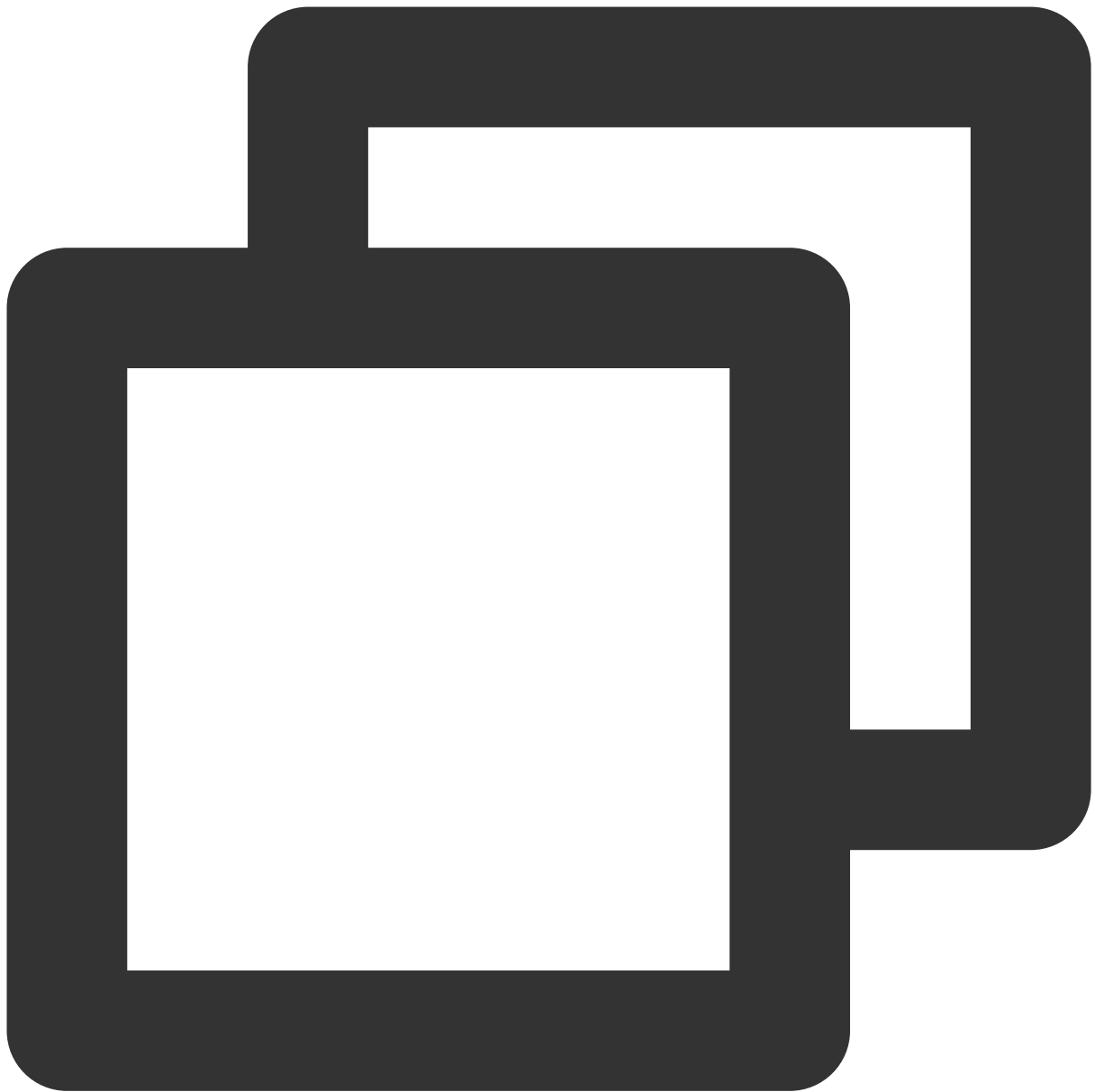
特定ユーザーの収集状態の取得

このインターフェースを呼び出して、ルーム内のメンバーのマイク状態を取得します。

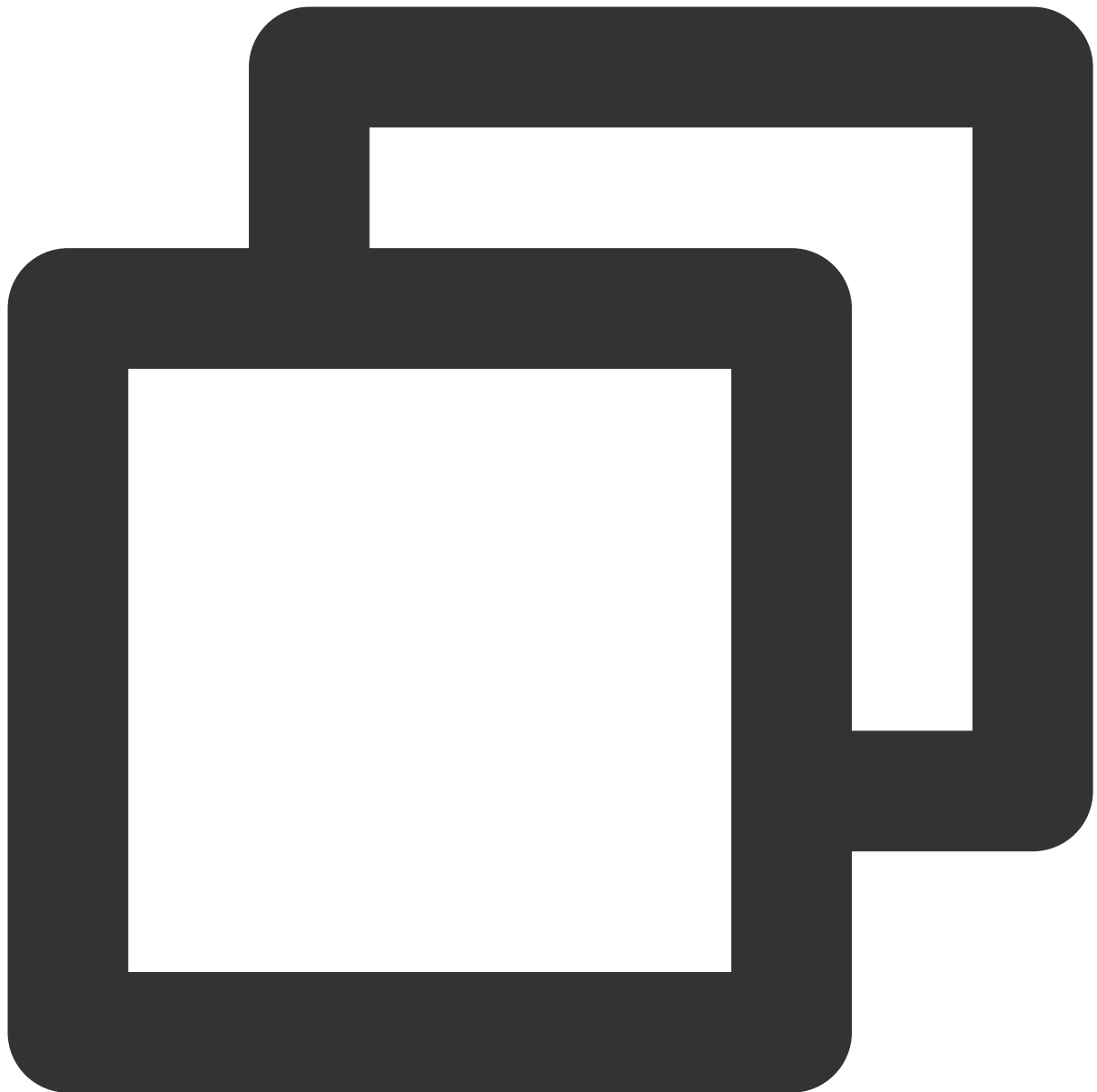
関数のプロトタイプ

Android

iOS



```
public abstract int GetMicState(String receiverID);
```



```
-(QAVResult)GetMicState:(NSString *)receiverID;
```

パラメータ	タイプ	意味
receiverID	NSString*	ターゲットユーザーOpenIdを入力します

コールバック

コールバックパラメータはITMG_ROOM_MANAGEMENT_GET_MIC_STATE。

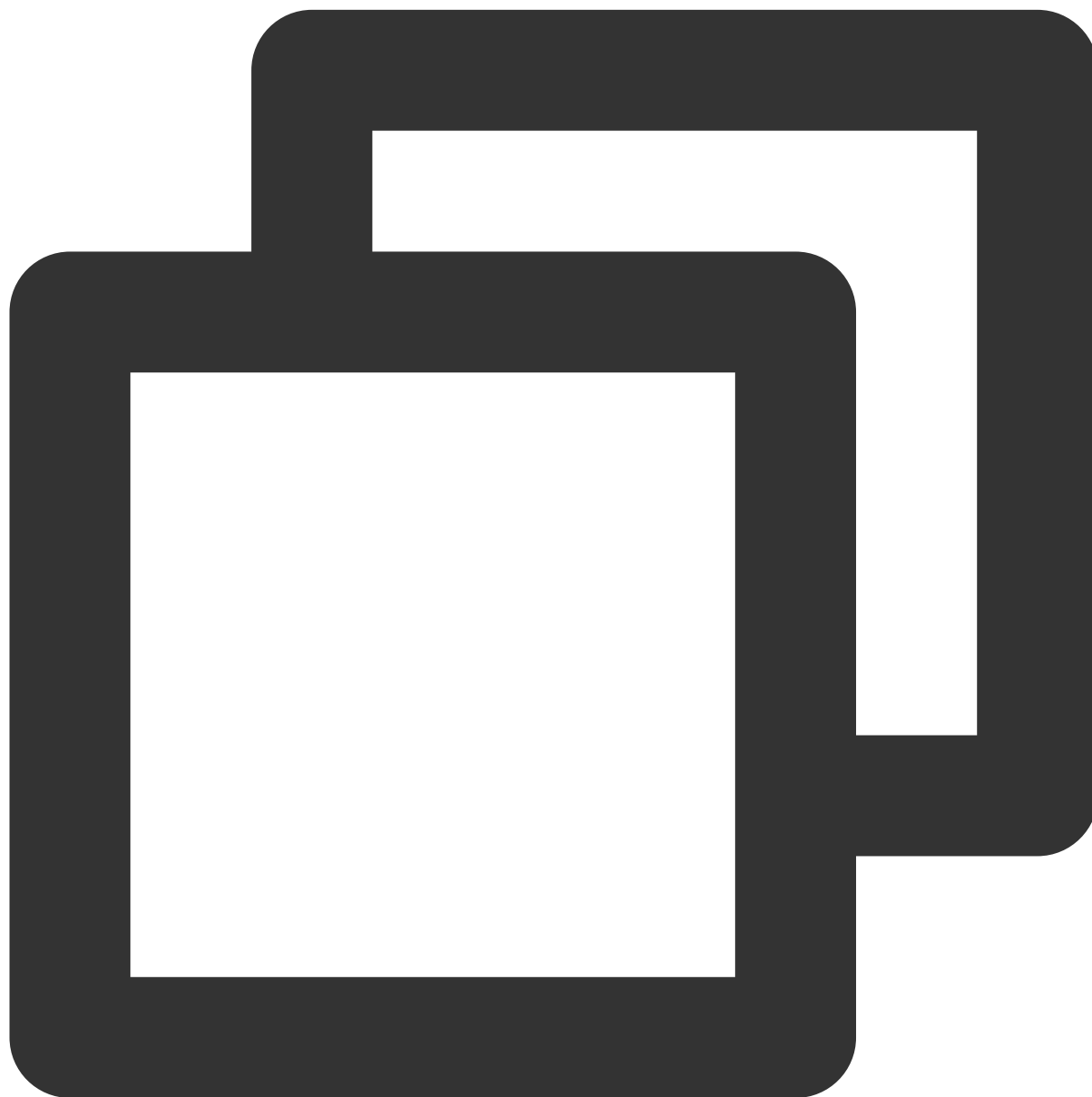
特定ユーザーの再生状態の取得

このインターフェースを呼び出して、ルーム内のメンバーのスピーカー状態を取得します。

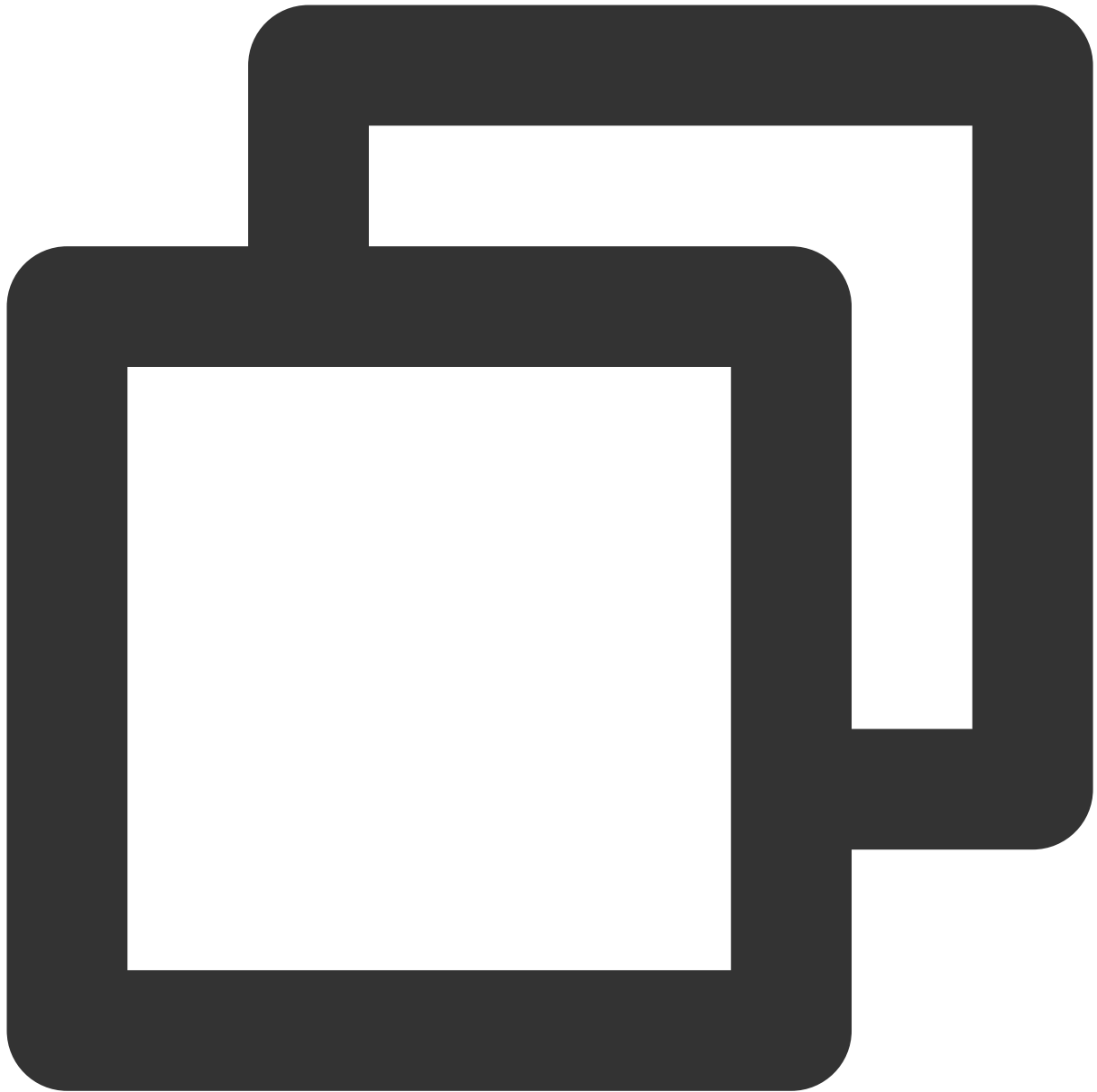
関数のプロトタイプ

Android

iOS



```
public abstract int GetSpeakerState(String receiverID);
```



```
-(QAVResult)GetSpeakerState:(NSString *)receiverID;
```

コールバック

コールバックパラメータはITMG_ROOM_MANAGEMENT_GET_SPEAKER_STATE。

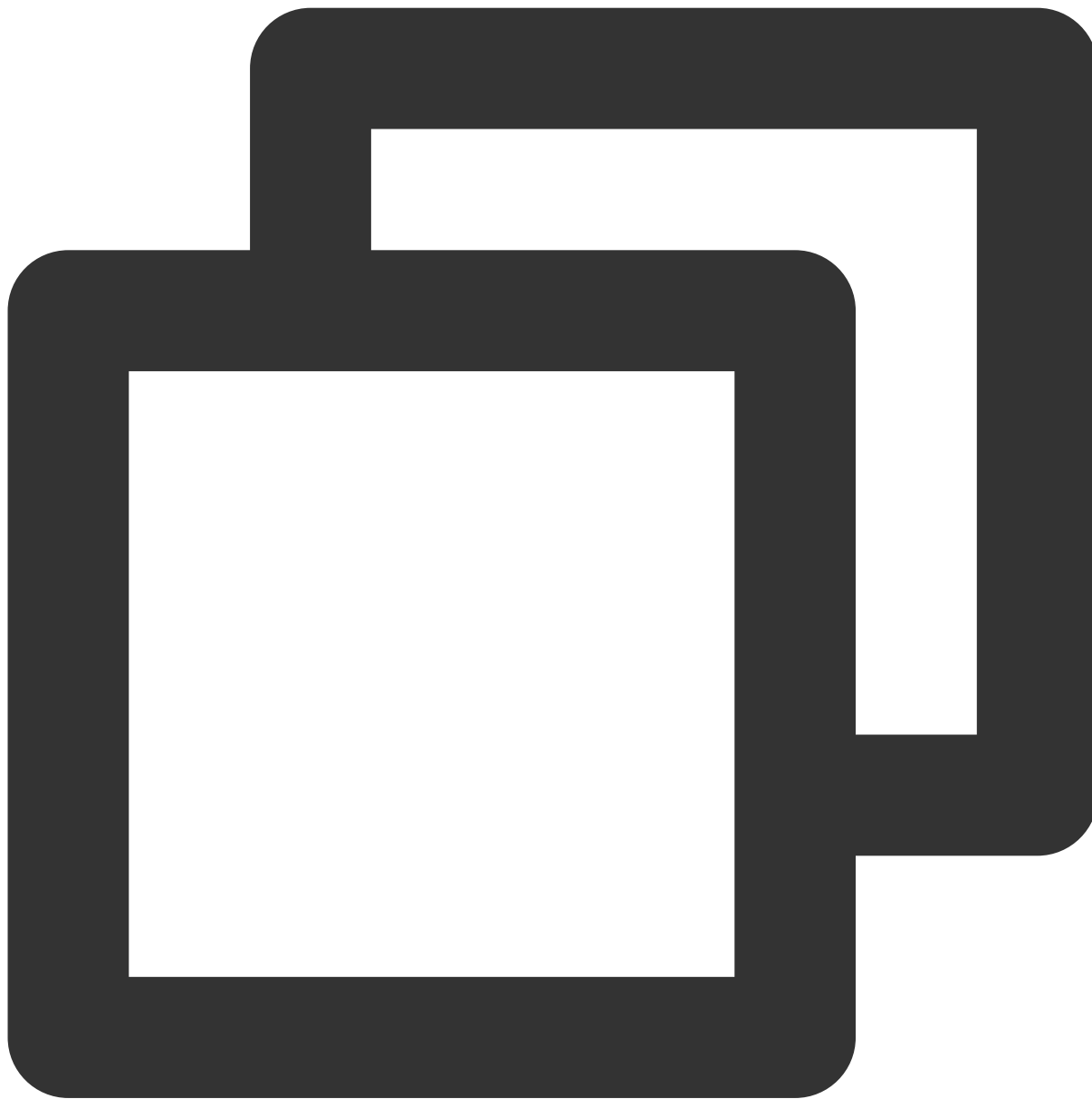
特定のメンバーによる収集および再生操作の禁止

メンバーがルームに入ると、デフォルトでマイクとスピーカーの操作が許可されます。このインターフェースを呼び出すと、ルームのメンバーがマイクとスピーカーを操作できなくなります。この機能は、メンバーがルームを出

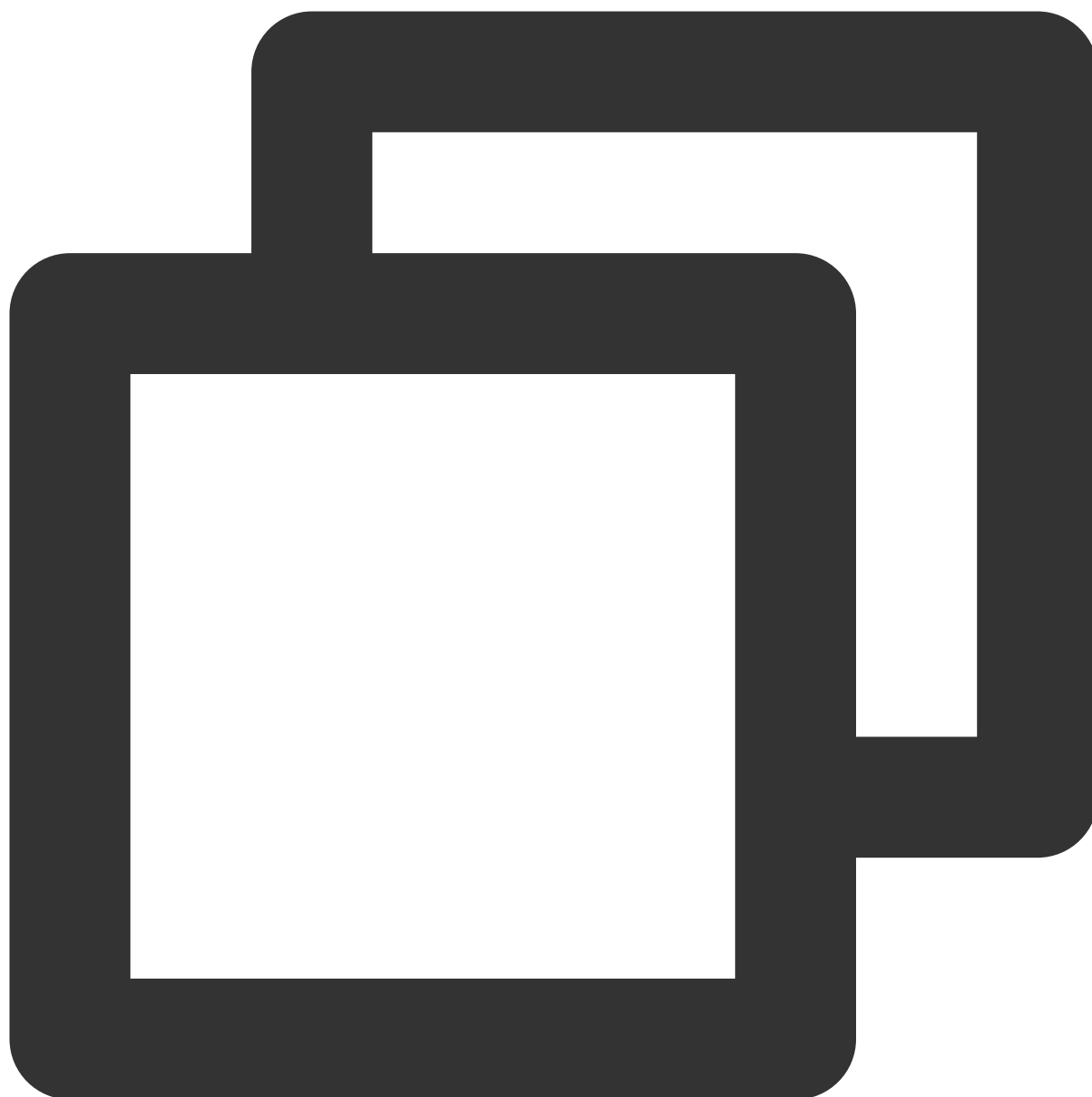
ると無効になります。

Android

iOS



```
public abstract int ForbidUserOperation(boolean isEnabled,String receiverID);
```



```
-(QAVResult)ForbidUserOperation:(BOOL)enable Receiver:(NSString *)receiverID;
```

パラメータ	タイプ	意味
enable	BOOL	YES：特定ユーザーによるデバイスの操作を禁止します。NO：特定ユーザーによるデバイスの操作を許可します
receiverID	NSString*	ターゲットユーザーOpenIdを入力します

コールバック

コールバックパラメータはITMG_ROOM_MANAGEMENT_FOBIN_OP。

コールバック処理

GMEの他のコールバックと同様に、ルーム管理のコールバックもOnEventで処理されます。イベント名は ITMG_MAIN_EVNET_TYPE_ROOM_MANAGEMENT_OPERATOR で、イベントは次のような構造体を返します。

コールバックパラメータ

パラメータ	タイプ	意味
SenderID	NSString	イベント送信者IDです。自身のOpenIdと同じ場合は、ローカルから送信されたコマンドです
ReceiverID	NSString	イベント受信者IDです。自身のOpenIdと同じ場合は、ローカルが受信したコマンドです
OperateType	NSNumber	イベントタイプ
Result	NSNumber	イベント結果です。0は成功を意味します
OperateValue	NSNumber	コマンドの詳細

OperateType

数値	イベントタイプ	意味
0	ITMG_ROOM_MANAGEMENT_CAPTURE_OP	収集デバイスのハードウェアコールバックの制御
1	ITMG_ROOM_MANAGEMENT_PLAY_OP	再生デバイスのハードウェアコールバックの制御
2	ITMG_ROOM_MANAGEMENT_AUDIO_SEND_OP	アップリンクコールバックの制御
3	ITMG_ROOM_MANAGEMENT_AUDIO_REC_OP	ダウンリンクコールバックの制御
4	ITMG_ROOM_MANAGEMENT_MIC_OP	マイクコールバックの制御
5	ITMG_ROOM_MANAGEMENT_PLAY_OP	スピーカーコールバックの制御
6	ITMG_ROOM_MANAGEMENT_GET_MIC_STATE	マイク状態の取得
7	ITMG_ROOM_MANAGEMENT_GET_SPEAKER_STATE	スピーカー状態の取得

8	ITMG_ROOM_MANAGERMENT_FOBIN_OP	マイクイベントとスピーカーイベントの操作を禁止します
---	--------------------------------	----------------------------

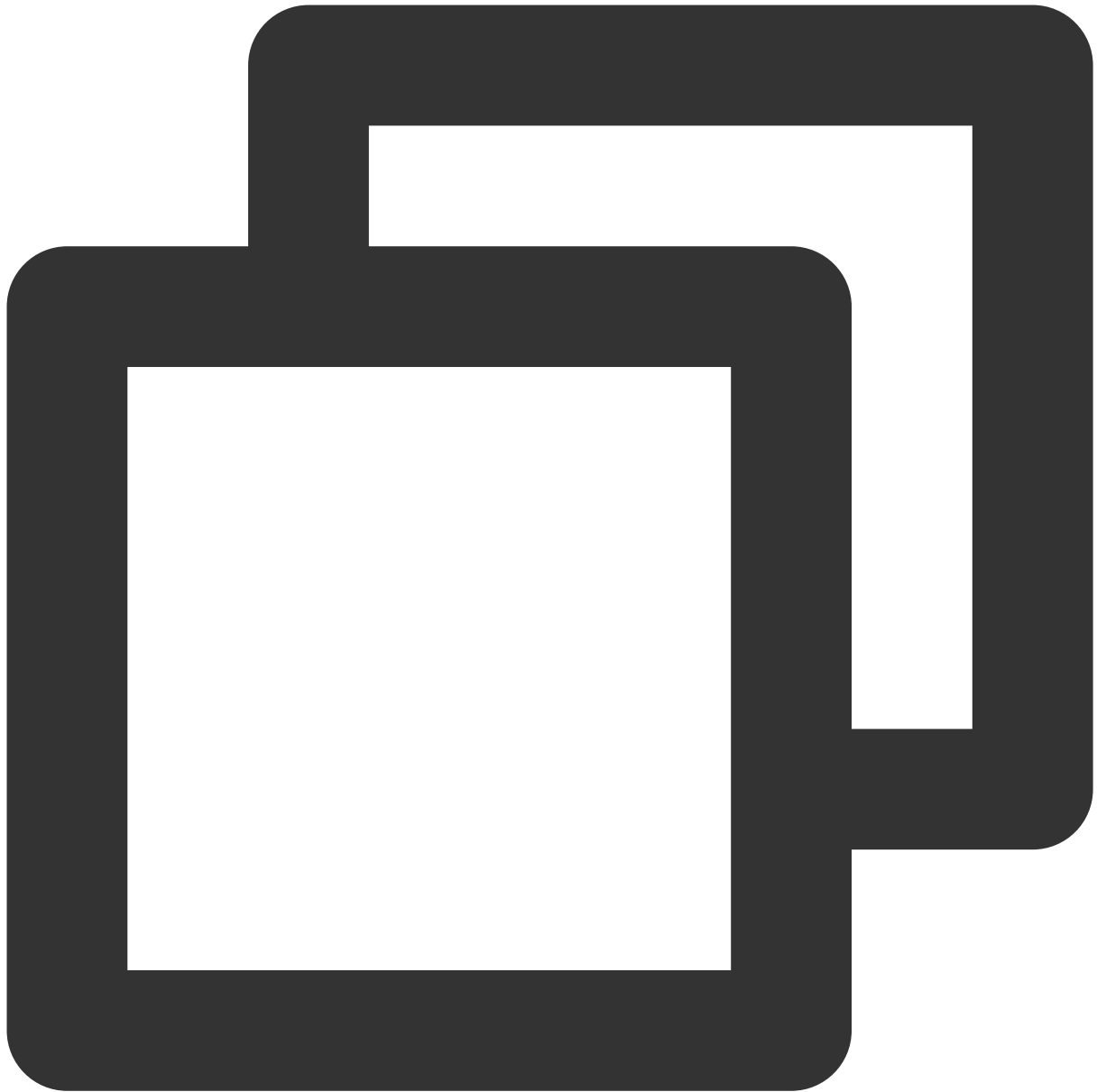
OperateValue

メンバー	意味
boolValue	0：コマンドをオフにします。1：コマンドをオンにします

サンプルコード

Android

iOS



```
public void OnEvent(ITMGContext.ITMG_MAIN_EVENT_TYPE type, Intent data) {  
    if (ITMGContext.ITMG_MAIN_EVENT_TYPE.ITMG_MAIN_EVNET_TYPE_ROOM_MANAGEMENT_OPERATOR  
  
        ArrayList<String> operatorArr = new ArrayList<String>();  
        operatorArr.add("收集");  
        operatorArr.add("再生");  
        operatorArr.add("上り");  
        operatorArr.add("下り");  
        operatorArr.add("上り収集");  
        operatorArr.add("下り再生");  
        operatorArr.add("mic状態");
```

```
operatorArr.add("spk状態");
operatorArr.add("mic/speak操作禁止");

String SenderID = data.getStringExtra("SenderID");
String ReceiverID = data.getStringExtra("ReceiverID");
int OperateType = data.getIntExtra("OperateType",-1000);

int Result =data.getIntExtra("Result",-1000);
boolean OperateValue = data.getBooleanExtra("OperateValue",false);
if (OperateType == -1000 ||Result == -1000) {
    return;
}
if (SenderID.equals(identifier)) {
    if (OperateType == ITMGContext.ITMG_ROOM_MANAGEMENT_GET_MIC_STATE |
        Toast.makeText(getActivity(), String.format("id:%sへの%s操作、結果",
    }else{
        Toast.makeText(getActivity(), String.format("id:%sへの%s%s操作、結果",
    }

} else if (ReceiverID.equals(identifier)||ReceiverID.equals("ALL")) {
    if (Result == 0) {
        switch (OperateType) {
            case ITMGContext.ITMG_ROOM_MANAGEMENT_CAPTURE_OP:
                {
                    if (!OperateValue) {
                        mSwitchCapture.setChecked(OperateValue);
                    }else{
                        AlertDialog.Builder dialog = new AlertDialog.Builder
                        dialog.setTitle("機器収集をオンにしますか");
                        dialog.setMessage("");
                        dialog.setCancelable(false);
                        dialog.setPositiveButton("オン", new DialogInterface
                            //OKボタンのクリックイベントを設定
                            @Override
                            public void onClick(DialogInterface dialog, int
                                mSwitchCapture.setChecked(true);
                                ITMGContext.GetInstance(getActivity()).GetA
                            }
                        });
                        dialog.setNegativeButton("オフ", new DialogInterface
                            //キャンセルボタンのクリックイベントを設定
                            @Override
                            public void onClick(DialogInterface dialog, int
                                }
                        });
                        dialog.show();
                    }
                }
            }
        }
    }
}
```

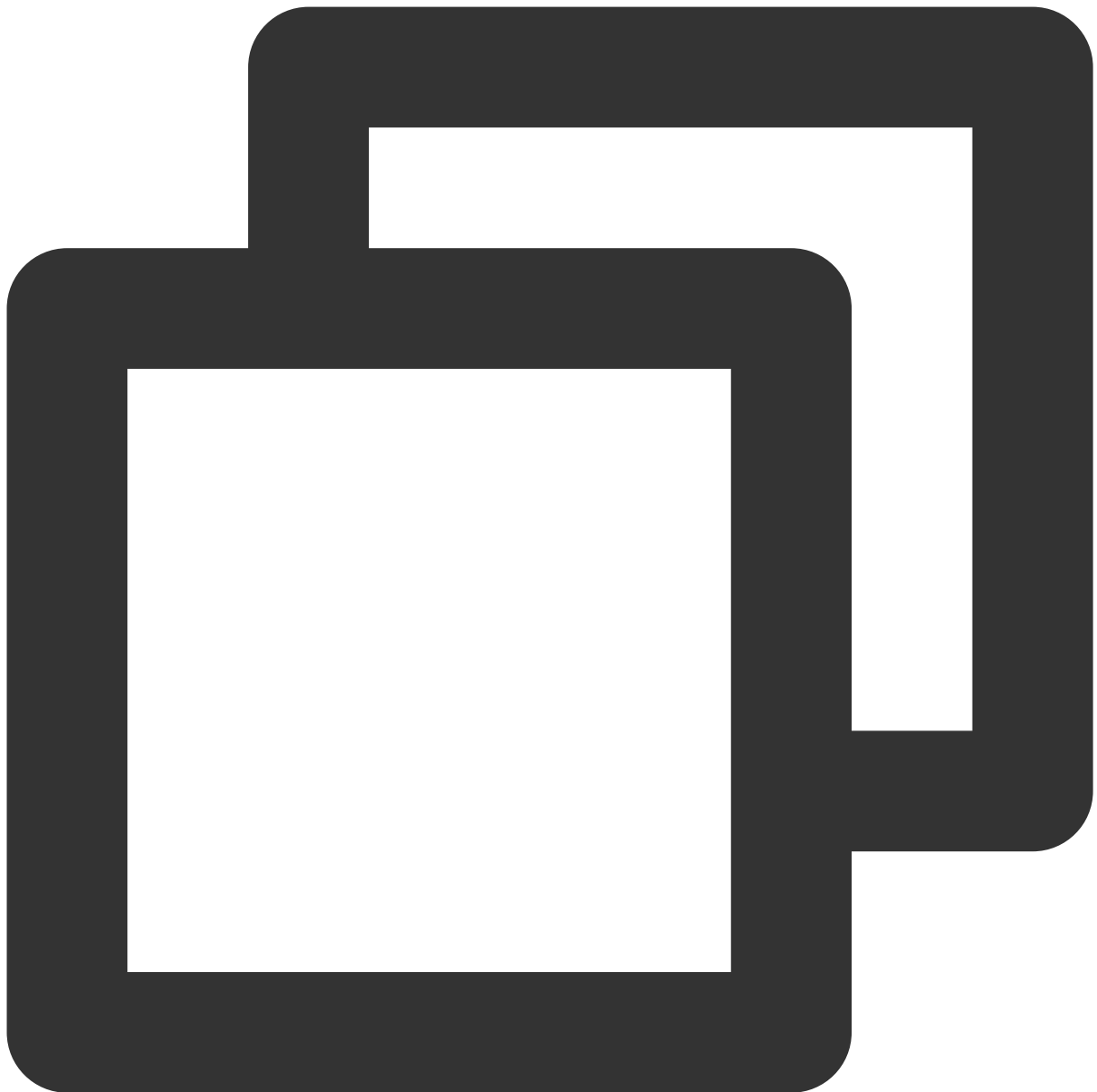
```
    }
    break;
case ITMGContext.ITMG_ROOM_MANAGEMENT_PLAY_OP:
{
    mSwitchPlayDevice.setChecked(OperateValue);
}
    break;
case ITMGContext.ITMG_ROOM_MANAGEMENT_AUDIO_SEND_OP:
{
    if (!OperateValue) {
        mSwitchSend.setChecked(OperateValue);
    }else{
        AlertDialog.Builder dialog = new AlertDialog.Builder
        dialog.setTitle("上りをオンにしますか");
        dialog.setMessage("");
        dialog.setCancelable(false);
        dialog.setPositiveButton("オン", new DialogInterface
            //OKボタンのクリックイベントを設定
            @Override
            public void onClick(DialogInterface dialog, int
                mSwitchSend.setChecked(true);
                ITMGContext.GetInstance(getActivity()).GetA
            }
        ));
        dialog.setNegativeButton("オフ", new DialogInterface
            //キャンセルボタンのクリックイベントを設定
            @Override
            public void onClick(DialogInterface dialog, int
            }
        ));
        dialog.show();
    }
}
    break;
case ITMGContext.ITMG_ROOM_MANAGEMENT_AUDIO_REC_OP:
{
    mSwitchRecv.setChecked(OperateValue);
}
    break;
case ITMGContext.ITMG_ROOM_MANAGEMENT_MIC_OP:
{
    if (!OperateValue) {
        mSwitchCapture.setChecked(OperateValue);
        mSwitchSend.setChecked(OperateValue);
    }else{
        AlertDialog.Builder dialog = new AlertDialog.Builder
```

```
        dialog.setTitle("収集と上りをオンにしますか");
        dialog.setMessage("");
        dialog.setCancelable(false);
        dialog.setPositiveButton("オン", new DialogInterface.OnClickListener() {
            //OKボタンのクリックイベントを設定
            @Override
            public void onClick(DialogInterface dialog, int which) {
                mSwitchCapture.setChecked(true);
                mSwitchSend.setChecked(true);
                ITMGContext.GetInstance(getActivity()).GetRoomManagement()
            }
        });
        dialog.setNegativeButton("オフ", new DialogInterface.OnClickListener() {
            //キャンセルボタンのクリックイベントを設定
            @Override
            public void onClick(DialogInterface dialog, int which) {
            }
        });
        dialog.show();
    }
}

break;
case ITMGContext.ITMG_ROOM_MANAGEMENT_SPEAKER_OP:
{
    mSwitchPlayDevice.setChecked(OperateValue);
    mSwitchRecv.setChecked(OperateValue);
}
break;

}

}
if (OperateType == ITMGContext.ITMG_ROOM_MANAGEMENT_GET_MIC_STATE) {
{
    Toast.makeText(getActivity(), String.format("id:%sからの%s操作、終了", id, OperateType), Toast.LENGTH_SHORT).show();
}
else if (OperateType == ITMGContext.ITMG_ROOM_MANAGEMENT_SPEAKER_OP) {
    Toast.makeText(getActivity(), String.format("id:%sからの%s%s操作、終了", id, OperateType, OperateValue), Toast.LENGTH_SHORT).show();
} else if (OperateValue == false) {
    Toast.makeText(getActivity(), String.format("id:%sからの%s%s操作、終了", id, OperateType, OperateValue), Toast.LENGTH_SHORT).show();
}
}
}
```



```
-(void)OnEvent:(ITMG_MAIN_EVENT_TYPE)eventType data:(NSDictionary *)data{
    NSString *log = [NSString stringWithFormat:@"OnEvent:%d,data:%@", (int)eventType, data];
    [self showLog:log];
    NSLog(@"====%@====", log);
    switch (eventType) {
        case ITMG_MAIN_EVNET_TYPE_ROOM_MANAGEMENT_OPERATOR:
        {
            NSArray *operatorArr = @[@"收集",@"再生",@"上り",@"下り",@"上り収集",@"下り収集"];
            // _openId
            NSString *SenderID = [data objectForKey:@"SenderID"];
            NSString *ReceiverID = [data objectForKey:@"ReceiverID"];
```

```
NSNumber *OperateType = [data objectForKey:@"OperateType"];
NSNumber *Result = [data objectForKey:@"Result"];
NSNumber *OperateValue = [data objectForKey:@"OperateValue"];

///自分が出したコマンド
if ([SenderID isEqualToString:_openId]) {
    if (OperateType.intValue == ITMG_ROOM_MANAGEMENT_GET_MIC_STATE || O
        NSString *alterString = [NSString stringWithFormat:@"id:%
        UIAlertView *alert = [[UIAlertView all
        [alert show];
    }
    else
    {
        NSString *alterString = [NSString stringWithFormat:@"id:%@への%
        UIAlertView *alert = [[UIAlertView alloc] initWithTi
        [alert show];
    }
}
else if([ReceiverID isEqualToString:_openId] ){ //他人からのコマンド
    if (Result.intValue == 0) {
        switch (OperateType.intValue) {
            case ITMG_ROOM_MANAGEMENT_CAPTURE_OP:{
                [_micSwitch setOn:OperateValue.boolValue animated:true]
            }
            break;
            case ITMG_ROOM_MANAGEMENT_PLAY_OP:{
                [_speakerSwitch setOn:OperateValue.boolValue animated:t
            }
            break;
            case ITMG_ROOM_MANAGEMENT_AUDIO_SEND_OP:{
                [_sendSwitch setOn:OperateValue.boolValue animated:true
            }
            break;
            case ITMG_ROOM_MANAGEMENT_AUDIO_REC_OP:{
                [_recvSwitch setOn:OperateValue.boolValue animated:true
            }
            break;
            case ITMG_ROOM_MANAGEMENT_MIC_OP:{
                [_micSwitch setOn:OperateValue.boolValue animated:true];
                [_sendSwitch setOn:OperateValue.boolValue animated:true];
            }
            break;
            case ITMG_ROOM_MANAGEMENT_SPEAKER_OP:{
                [_speakerSwitch setOn:OperateValue.boolValue animated:true]
                [_recvSwitch setOn:OperateValue.boolValue animated:true];
            }
        }
    }
}
```

```
        }
        break;
    default:
        break;
    }

    if (OperateType.intValue == ITMG_ROOM_MANAGEMENT_GET_MIC_STATE || O
        NSString *alterString = [NSString stringWithFormat:@"id:%@",
            UIAlertView *alert = [[UIAlertView alloc] initWithTit
            [alert show];
        }
    else{
        NSString *alterString = [NSString stringWithFormat:@"id:%@",
            UIAlertView *alert = [[UIAlertView alloc] initWithTit
            [alert show];
        }
    }
}
break;
}
```