Tencent Real-Time Communication

旧バージョンのドキュメント

製品ドキュメント

カタログ：

旧バージョンのドキュメント
- TUIRoom（Web）の統合
- TUIRoom (Android)の統合
- TUIRoom (iOS)の統合
- TUIRoom (Flutter)の統合
- TUIRoom (Windows&Mac)の統合
- TUIRoom (Electron)の統合
- TUIRoom APIのクエリー
  - TUIRoom(Android)
  - TUIRoom API（iOS）
  - TRTCMeeting API（Flutter）
  - TUIRoom(Windows&Mac)
- クラウドレコーディングと再生の実現（旧）
- Legacy Version (TUIVoiceRoom)
  - TUIVoiceRoom (Android)の統合
  - TUIVoiceRoom (iOS)の統合
  - TUIVoiceRoom API照会
    - TRTCVoiceRoom (iOS)
    - TRTCVoiceRoom (Android)

旧バージョンのドキュメント
TUIRoom（Web）の統合

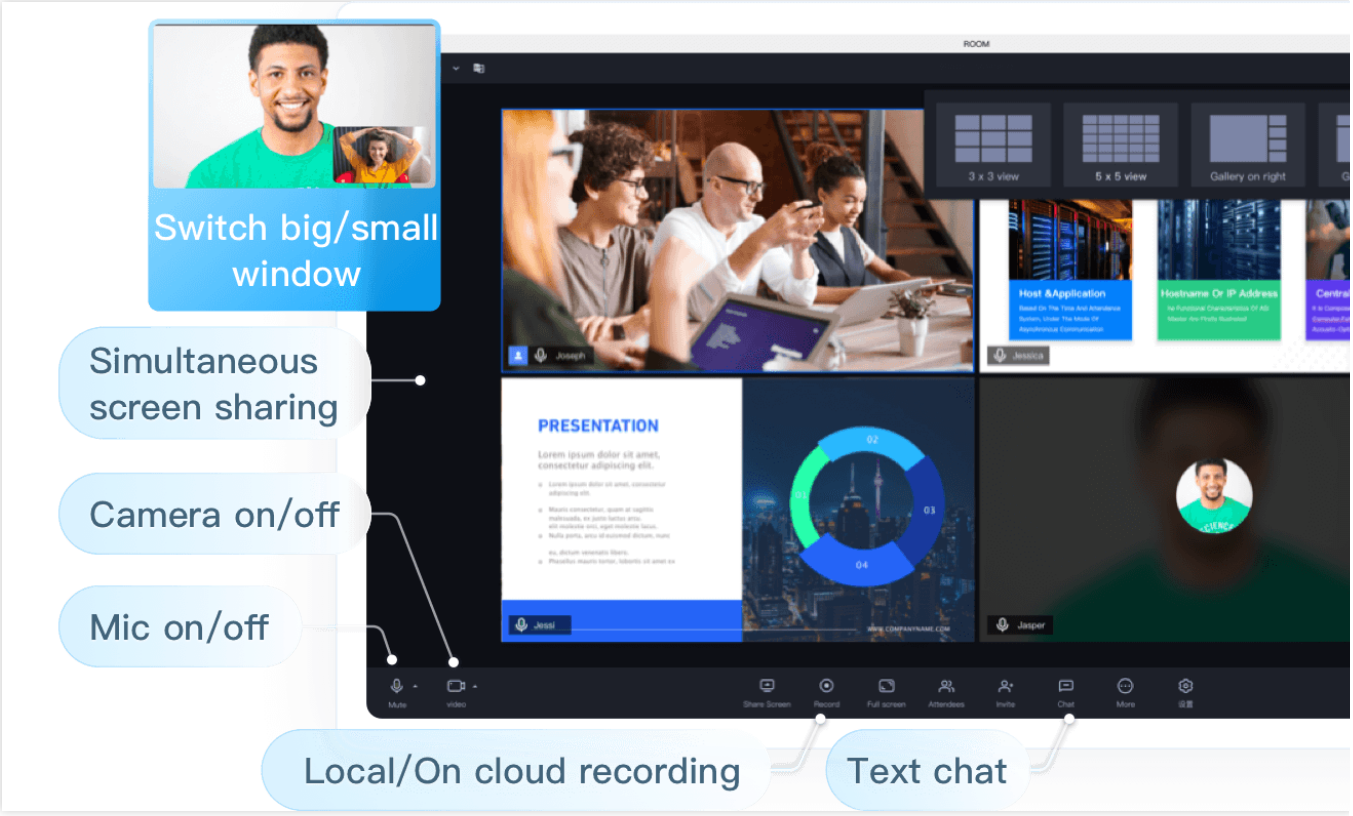
最終更新日：：2024-07-19 15:35:35

コンポーネントの説明

TUIRoomは、UIを含むオープンソースのオーディオビデオコンポーネントであり、TUIRoomの統合により、業務にオーディオビデオルーム、画面共有、チャットなどの機能を速やかに立ち上げることができます。Web側のTUIRoomの基本機能は下図のとおりです。

説明：

TUIKitシリーズコンポーネントはTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期的にアクティブにすることができます。IMサービスの課金ルールの詳細については、Instant Messagingの料金説明をご参照ください。TRTCをアクティブにすると、デフォルトでは、100DAUまでサポートするIM SDK体験版もアクティブになりますs。



TUIRoomオンラインリンクをクリックすると、TUIRoomのその他の機能を体験できます。

GithubをクリックしてTUIRoomコードをダウンロードし、TUIRoom Webデモプロジェクトのクイックスタートド

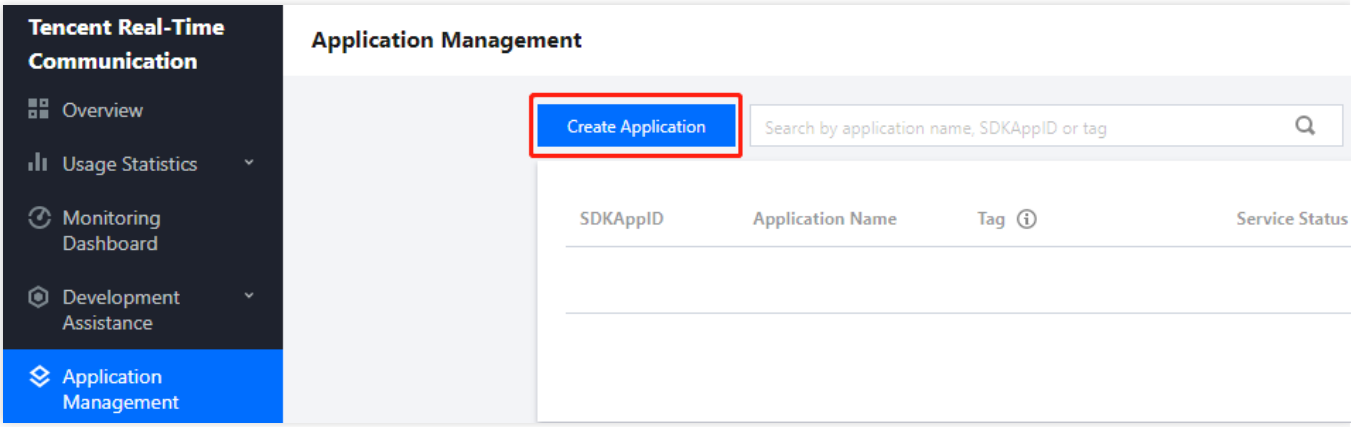
キュメントを参照してTUIRoom Webデモプロジェクトをクイックスタートできます。
Web側のTUIRoomコンポーネントを既存のビジネスに統合する場合は、このドキュメントをご参照ください。

コンポーネントの統合

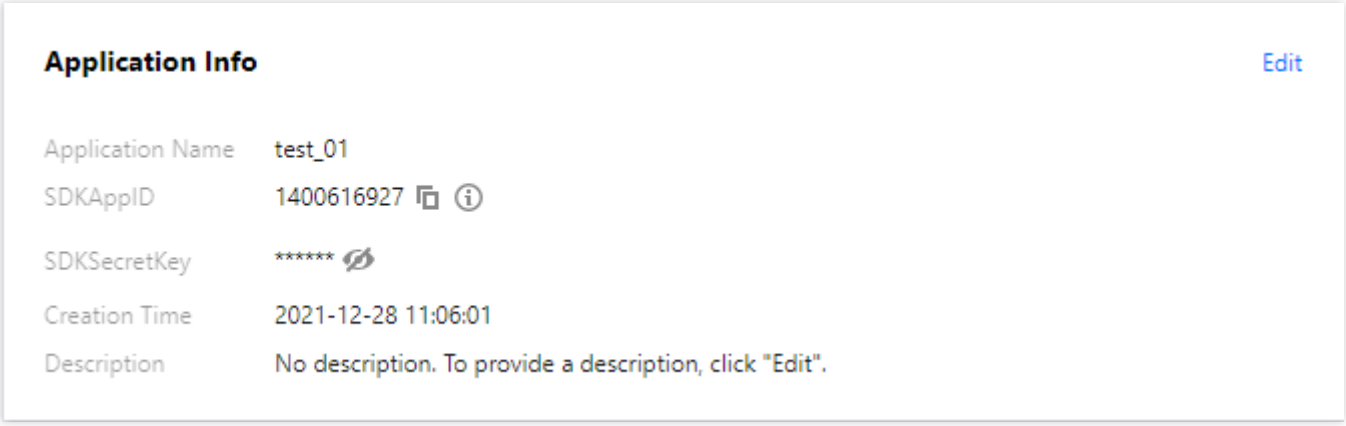
TUIRoomコンポーネントは、Vue3+TS+Pinia+Element Plus+SCSSを使用して開発されており、インポート先のプロジェクトにはVue3+TSテクノロジースタックが必要です。

ステップ1：Tencent Cloud TRTCとInstant Messagingサービスを有効化します

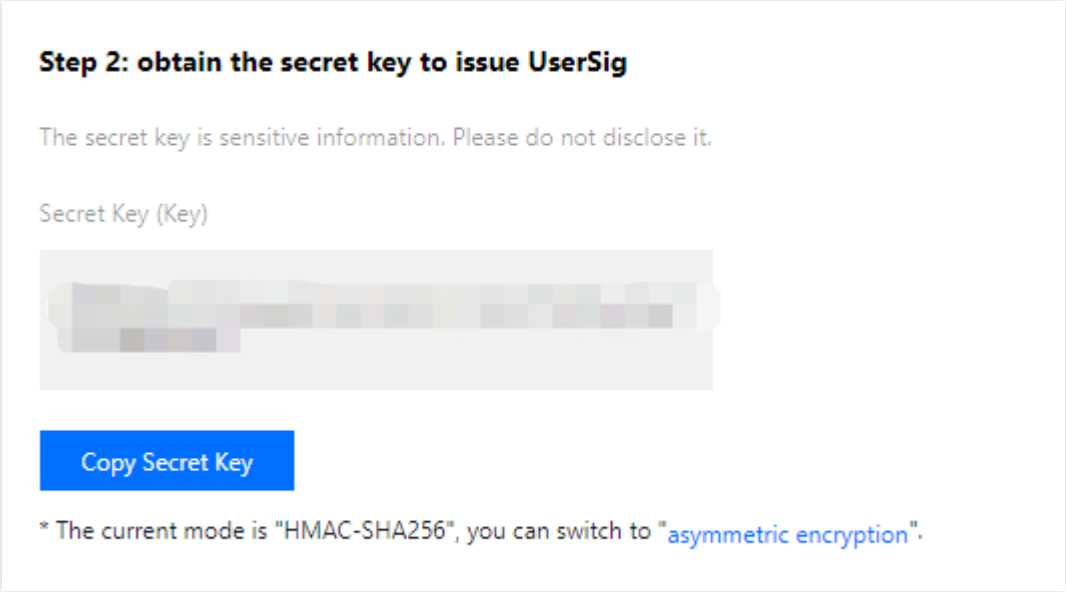
TUIRoomは、Tencent Cloud TRTCとInstant Messagingサービスをベースに開発されたものです。
ステップ1. **TRTCアプリケーションを作成します**
Tencent Cloudアカウントがない場合は、Tencent Cloudアカウントの登録を行ってください。
TRTCコンソール で、**アプリケーション管理>アプリケーションの作成** をクリックし、新たなアプリケーションを作成します。



2.TRTCキー情報およびキー情報の取得
3.アプリケーション管理>アプリケーション情報でSDKAppID情報を取得します。SDKAppIDは、TRTCのアプリケーションIDであり、サービス分離のために使用され、すなわち、異なるSDKAppIDによる通話が相互に通じません。



4.アプリケーション管理>クイックマスターでアプリケーションのsecretKey情報を取得します。SecretKeyはTRTCのアプリケーションキーであり、SDKAppIDとペアで使用してください。TRTCサービスを正当に使用するための認証用チケットUserSigをチェックアウトします。



5.UserSigの発行

UserSigとは、悪意ある攻撃者によるクラウドサービスの使用権の盗用を防ぐために、Tencent Cloudによって設計されたセキュリティ保護された署名です。TUIRoomを初期化するには、UserSigパラメータを提供してください。

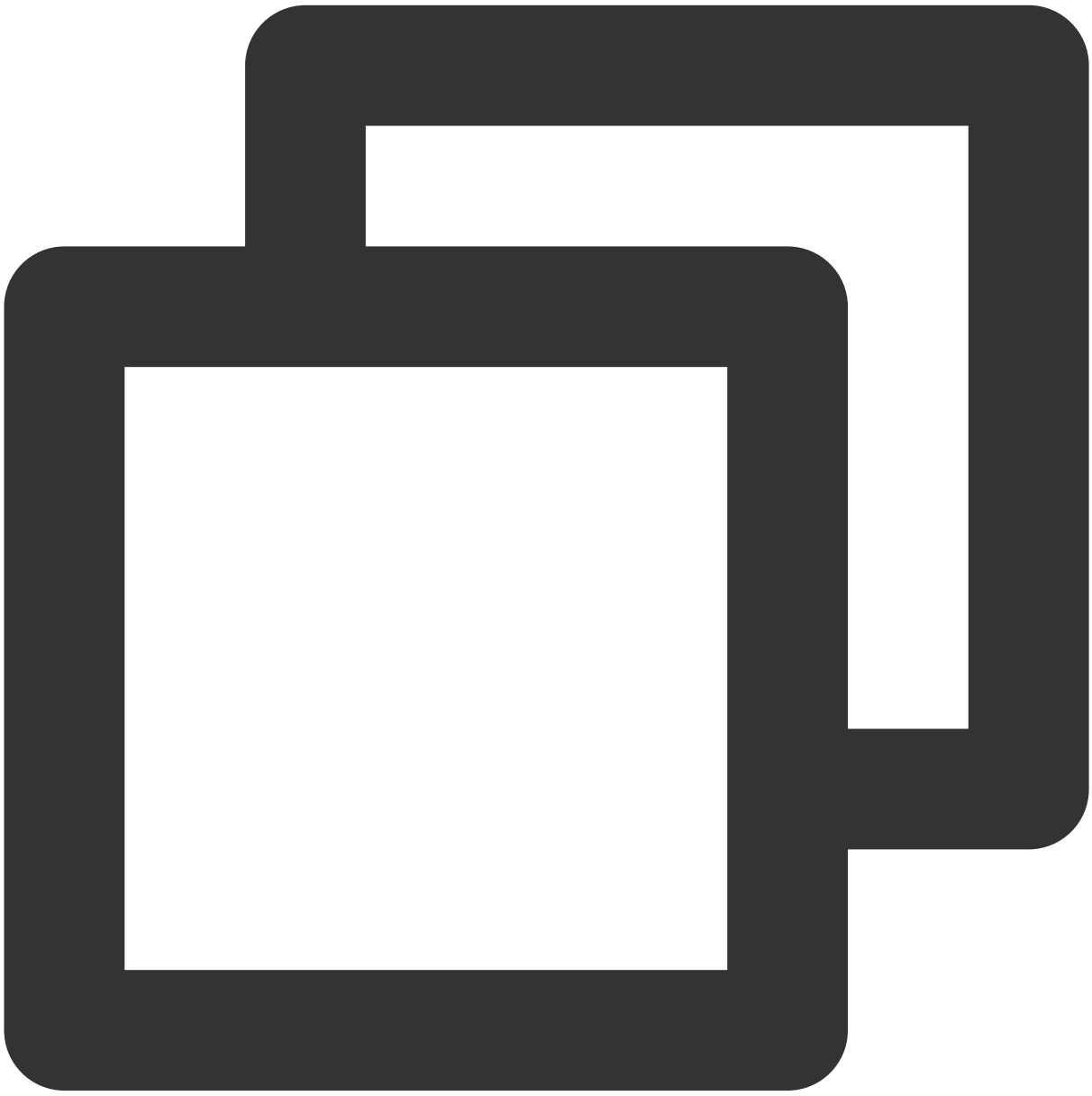
デバッグスタート段階でuserSigを発行する方法については、デバッグスタート段階でのUserSigの計算方法をご参照ください。

本番環境でuserSigを発行する方法については、本番環境でのUserSigの計算方法をご参照ください。

ステップ2：TUIRoomコンポーネントのダウンロードとコピー

1. Githubをクリックして、TUIRoomウェアハウスコードをクローンまたはダウンロードします。

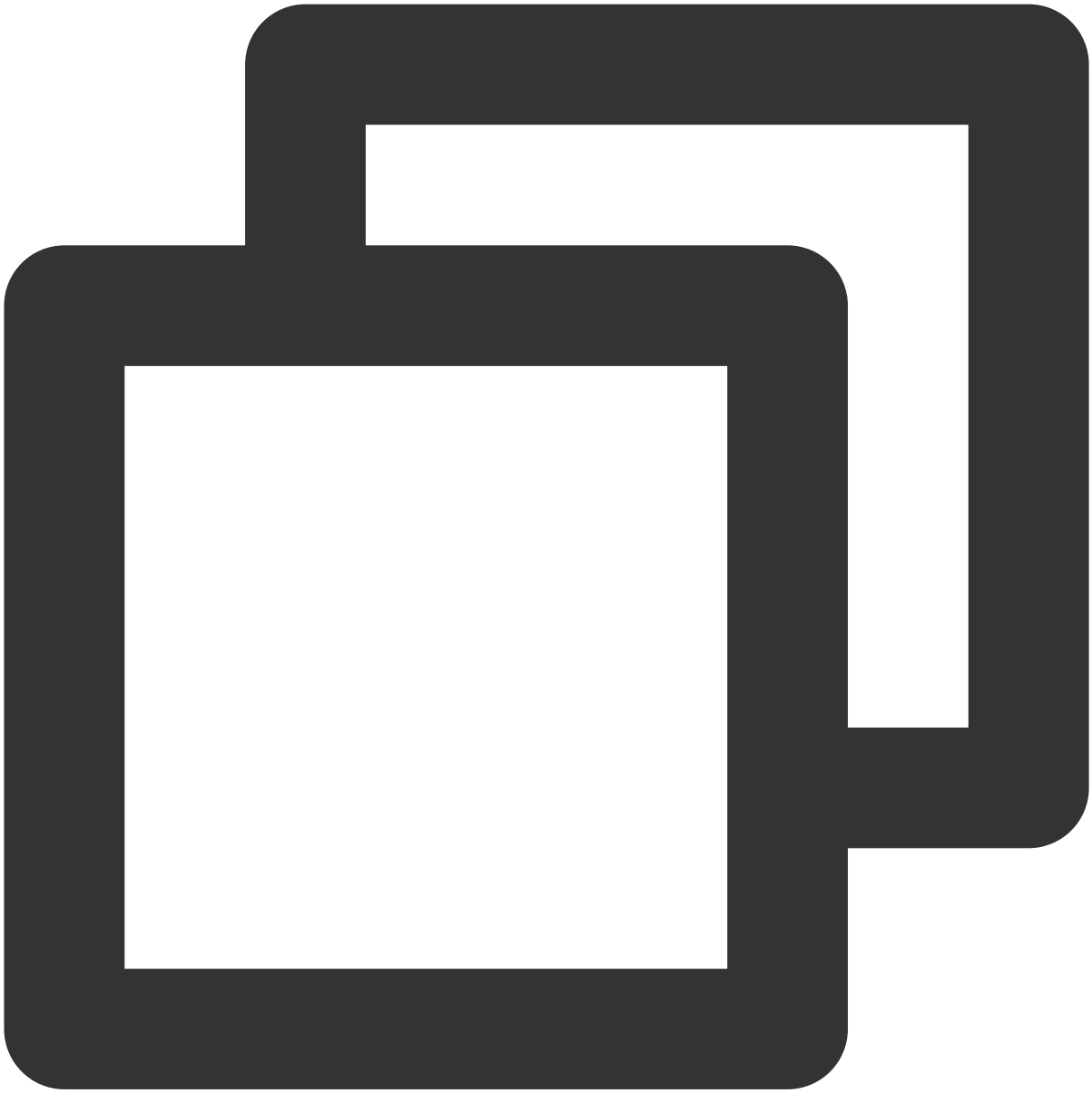
2. ビジネスサイドで既存のVue3 + TSプロジェクトを開き、ViteとWebpackのパッケージ方式をサポートします。Vue3 + TSのプロジェクトがない場合、以下のいずれかの方法を選択して、テンプレートプロジェクトを生成することができます。
Vue3 + Vite + TSのテンプレートプロジェクトの生成
Vue3 + Webpack + TSテンプレートプロジェクトの生成



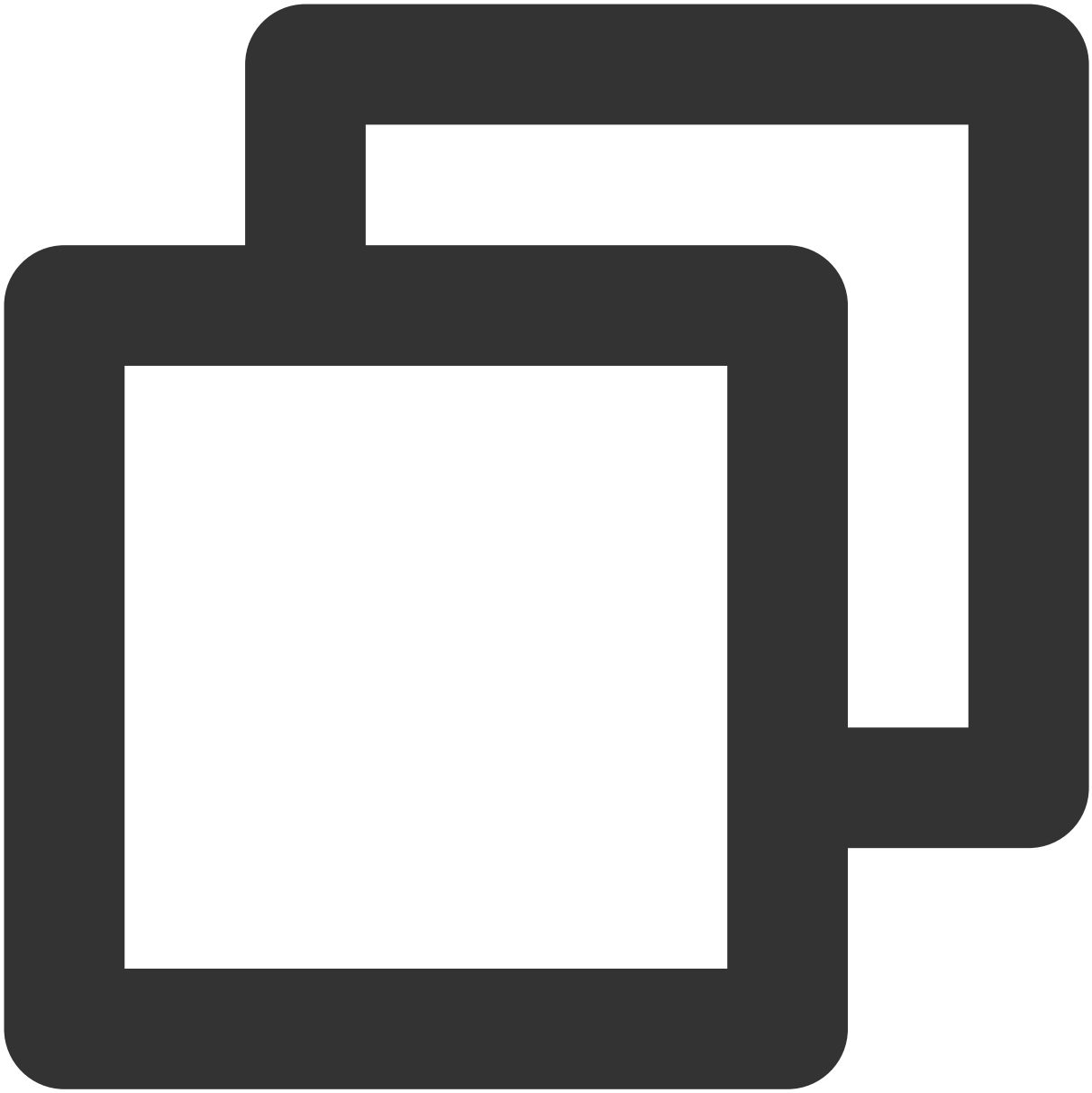
```
npm create vite@latest TUIRoom-demo -- --template vue
```

ご注意：

テンプレートプロジェクトスクリプトの生成を実行する場合、ステップ1で直接リターンし、ステップ2でVueを選択し、ステップ3でvue-tsを選択します。
Vue3 + Vite + TSテンプレートプロジェクトの生成に成功したら、以下のスクリプトを実行します。



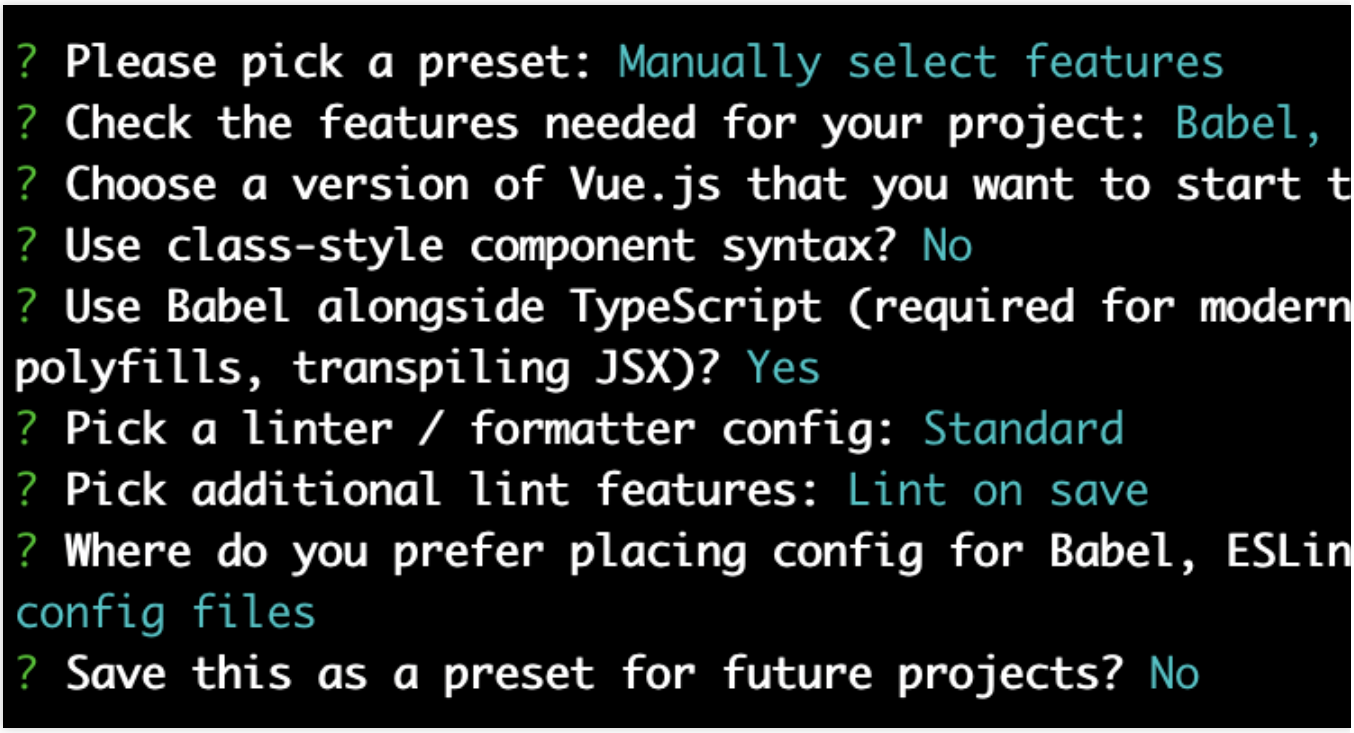
```
cd TUIRoom-demo
npm install
npm run dev
```



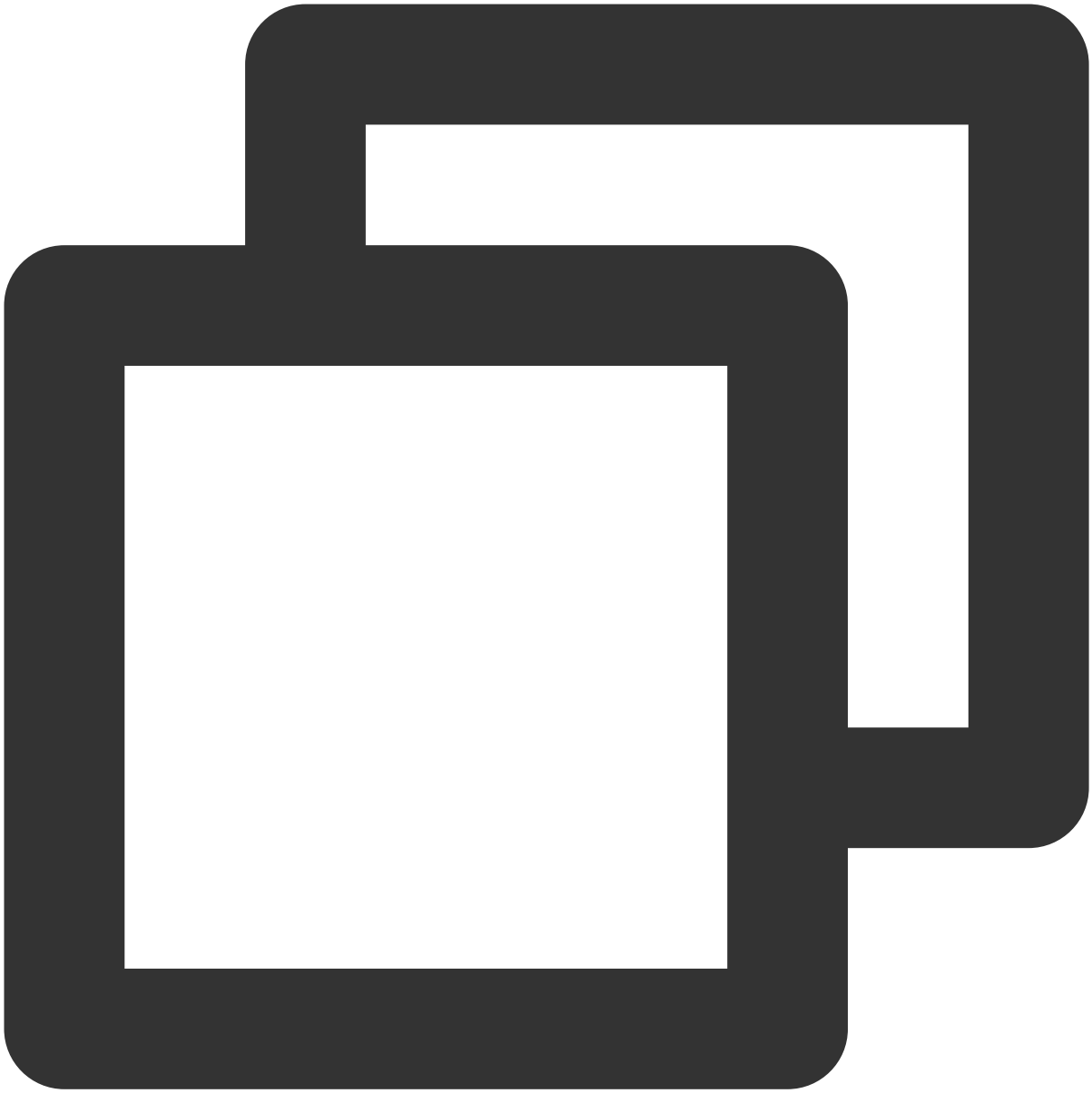
```
// vue-cliをインストールします。Vue CLI 4.xはNode.jsのv10以上のバージョンが必要なのでご注意くだ
npm install -g @vue/cli
::: Vue3 + Webpack + TSテンプレートプロジェクトの作成
vue create TUIRoom-demo
```

ご注意：

テンプレートプロジェクトスクリプトの生成を実行する場合、テンプレートの生成方式としてManually select featuresを選択し、残りの設定オプションは画像を参照します。



Vue3 + Webpack + TSテンプレートプロジェクトの生成に成功したら、以下のスクリプトを実行します。



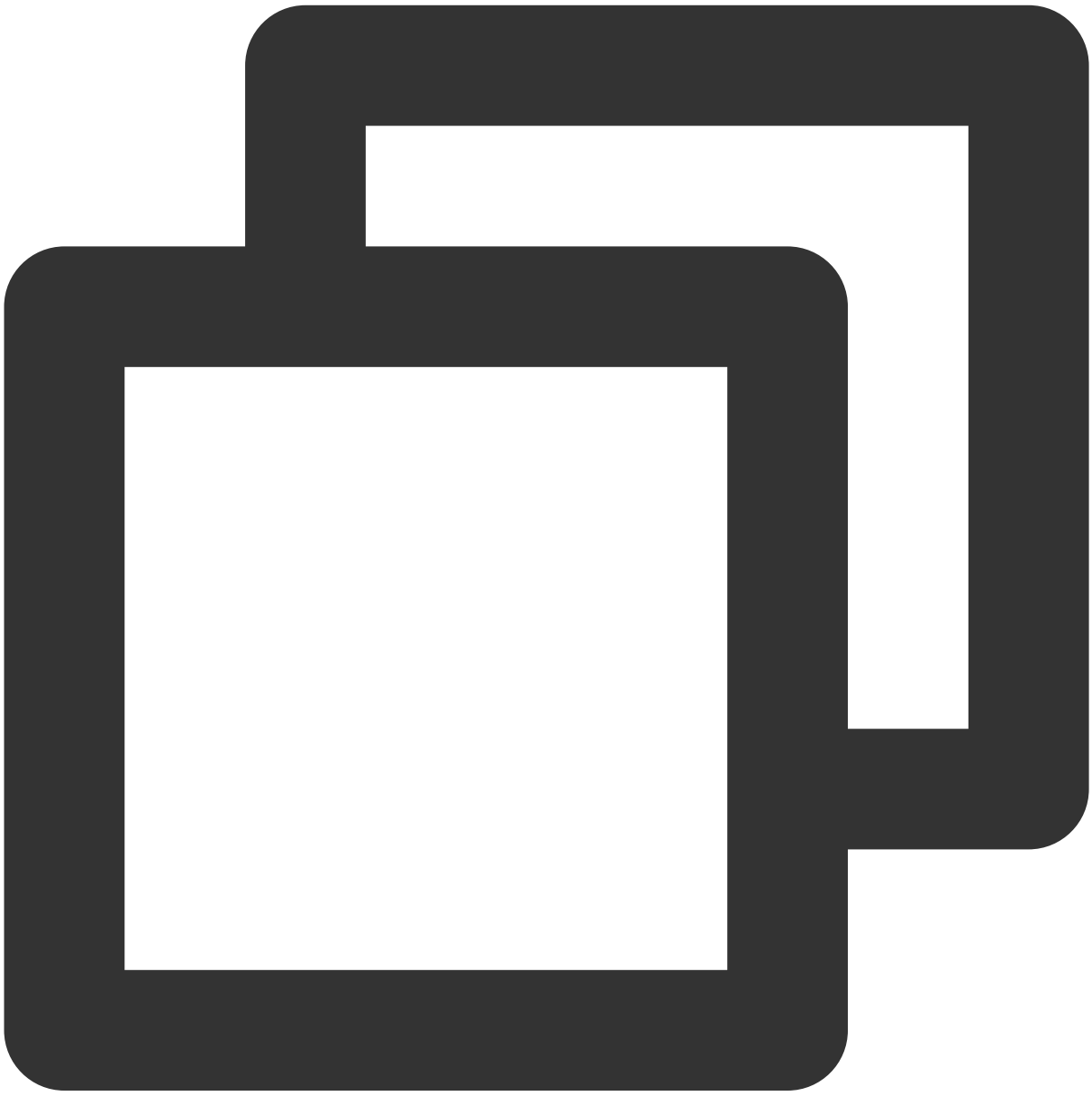
```
cd TUIRoom-demo
npm run serve
```

3. `TUIRoom/Web/src/TUIRoom` フォルダを既存のプロジェクトsrc/ディレクトリにコピーします。

ステップ3：TUIRoomコンポーネントのインポート

1. ページでTUIRoomコンポーネントを参照します。例：TUIRoomコンポーネントを `App.vue` コンポーネントにインポートします。

TUIRoomコンポーネントは、ユーザーをキャスターロールと一般メンバーロールに分けます。コンポーネントは外部にinit、createRoom、enterRoomの各メソッドを提供します。
キャスターと一般メンバーはinitメソッドを使用してTUIRoomコンポーネントの適用とユーザーデータを初期化できます。キャスターはcreateRoomメソッドを使用してルームを作成して参加します。一般メンバーはenterRoomメソッドを使用してキャスターが作成したルームに参加できます。



```
<template>
 <room ref="TUIRoomRef"></room>
</template>
```

```
<script setup lang="ts">
 import { ref, onMounted } from 'vue';
 // TUIRoomコンポーネントをインポートします。インポートパスが正しいことを確認してください
 import Room from './TUIRoom/index.vue';
 // TUIRoomコンポーネントのメソッドを呼び出すために、TUIRoomコンポーネント要素を取得します
 const TUIRoomRef = ref();

  onMounted(async () => {
  // TUIRoomコンポーネントの初期化
  // キャスターはルームを作成する前にTUIRoomコンポーネントを初期化してください
  // 一般メンバーは、ルームに参加する前にTUIRoomコンポーネントを初期化してください
  await TUIRoomRef.value.init({
      // sdkAppIdを取得するにはステップ1をご参照ください
      sdkAppId: 0,
      // ビジネスにおけるユーザーの一意の識別ID
      userId: '',
      // ローカル開発デバッグは、userSigをhttps://console.intl.cloud.tencent.com/trtc/us
      userSig: '',
      // ユーザーがビジネスで使用するニックネーム
      userName: '',
      // ユーザーがビジネスで使用するアバターへのリンク
      userAvatar: '',
      // 画面共有のためにユーザーが使用する一意のID。shareUserId=`share_${userId}`が必要です
      shareUserId: '',
      // このドキュメントのステップ1>3を参照し、sdkAppIdとshareUserIdを使用してshareUserSig
      shareUserSig: '',
  })
   // デフォルトではルームの作成が実行され、実際のインポートはオンデマンドでhandleCreateRoomメ
  await handleCreateRoom();
 })

 // キャスターがルームを作成します。このメソッドはルームが作成されたときにのみ呼び出されます
 async function handleCreateRoom() {
  // roomIdはユーザーが参加するルーム番号です。roomIdタイプはnumber型が必要です
  // roomModeは、'FreeSpeech'（自由発言モード）と'ApplySpeech'（挙手発言モード）の2モードを含
  // roomParamは、ユーザがルームに参加したときのデフォルトの動作を指定します（デフォルトでマイクを
  const roomId = 123456;
  const roomMode = 'FreeSpeech';
  const roomParam = {
      isOpenCamera: true,
      isOpenMicrophone: true,
  }
  await TUIRoomRef.value.createRoom(roomId, roomMode, roomParam);
 }

 // 一般メンバーがルームに参加します。このメソッドは作成されたルームに一般メンバーが参加するときに呼
 async function handleEnterRoom() {
```

```
    // roomIdはユーザーが参加するルーム番号です。roomIdタイプはnumber型が必要です
    // roomParamは、ユーザがルームに参加したときのデフォルトの動作を指定します（デフォルトでマイクを
    const roomId = 123456;
    const roomParam = {
        isOpenCamera: true,
        isOpenMicrophone: true,
    }
    await TUIRoomRef.value.enterRoom(roomId, roomParam);
  }
</script>

<style>
html, body {
  width: 100%;
  height: 100%;
  margin: 0;
}

#app {
  width: 100%;
  height: 100%;
}
</style>
```

ご注意：

ページで上記のコードをコピーした後、TUIRoomインターフェースのパラメータを実際のデータに変更する必要があります。

ステップ4：開発環境の設定

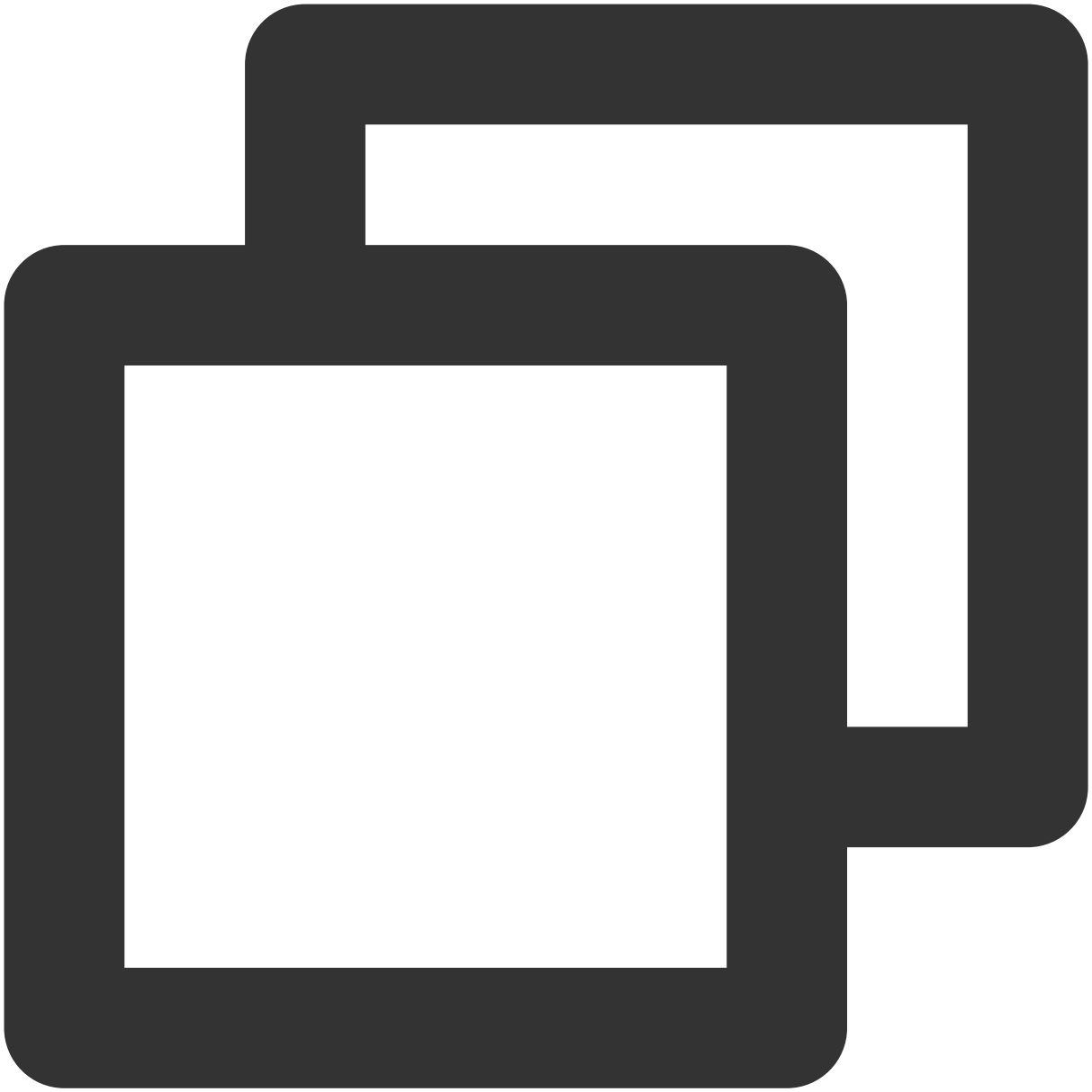
TUIRoomコンポーネントがインポートされたら、プロジェクトが正常に機能するように、次の設定を行ってください

Vue3 + Vite + TSプロジェクトの開発環境の設定

Vue3+Webpack+Tsプロジェクトの開発環境の設定

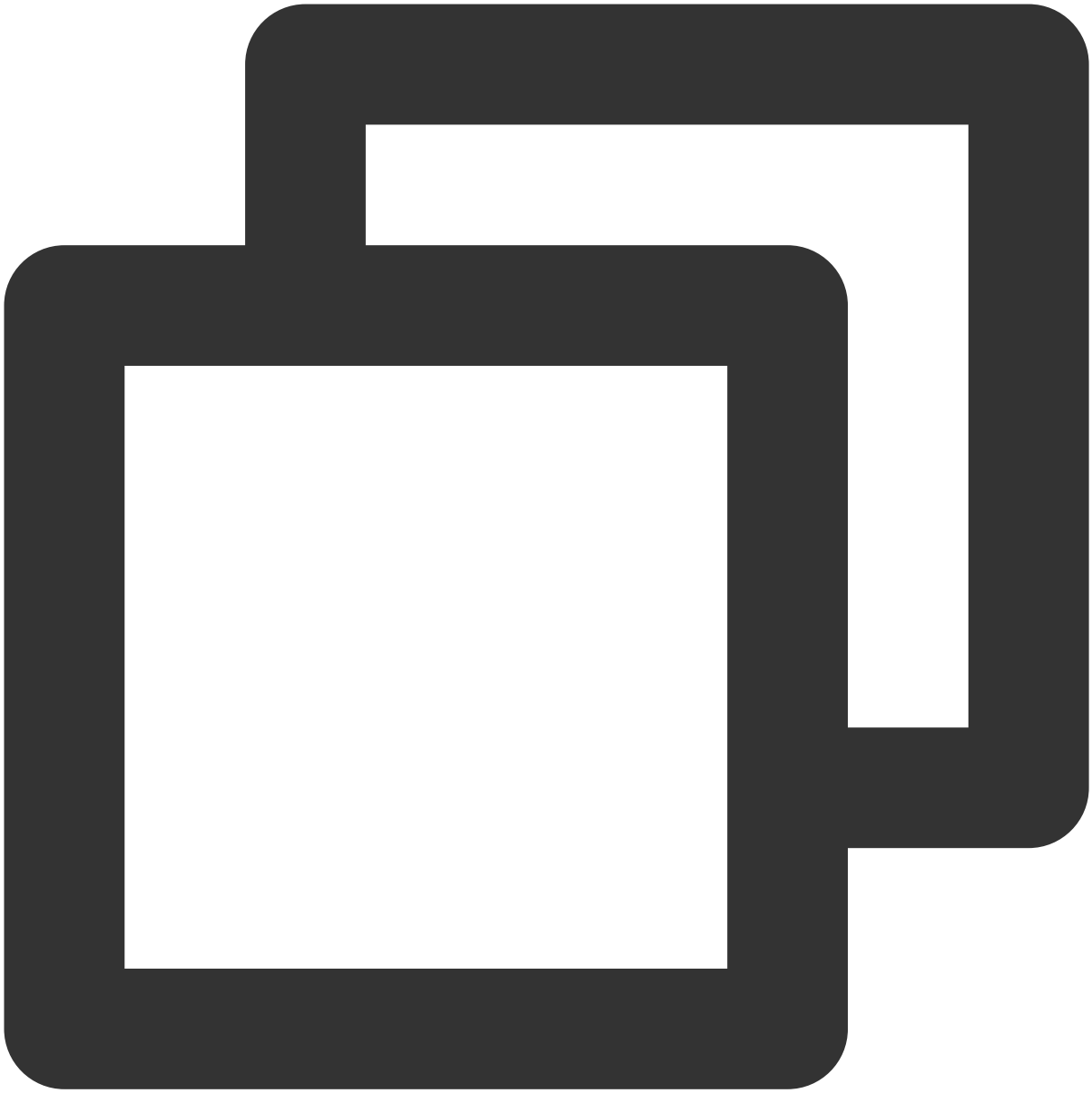
1. 依存関係のインストール

開発環境の依存関係をインストールします。



```
npm install sass typescript unplugin-auto-import unplugin-vue-components -S -D
```

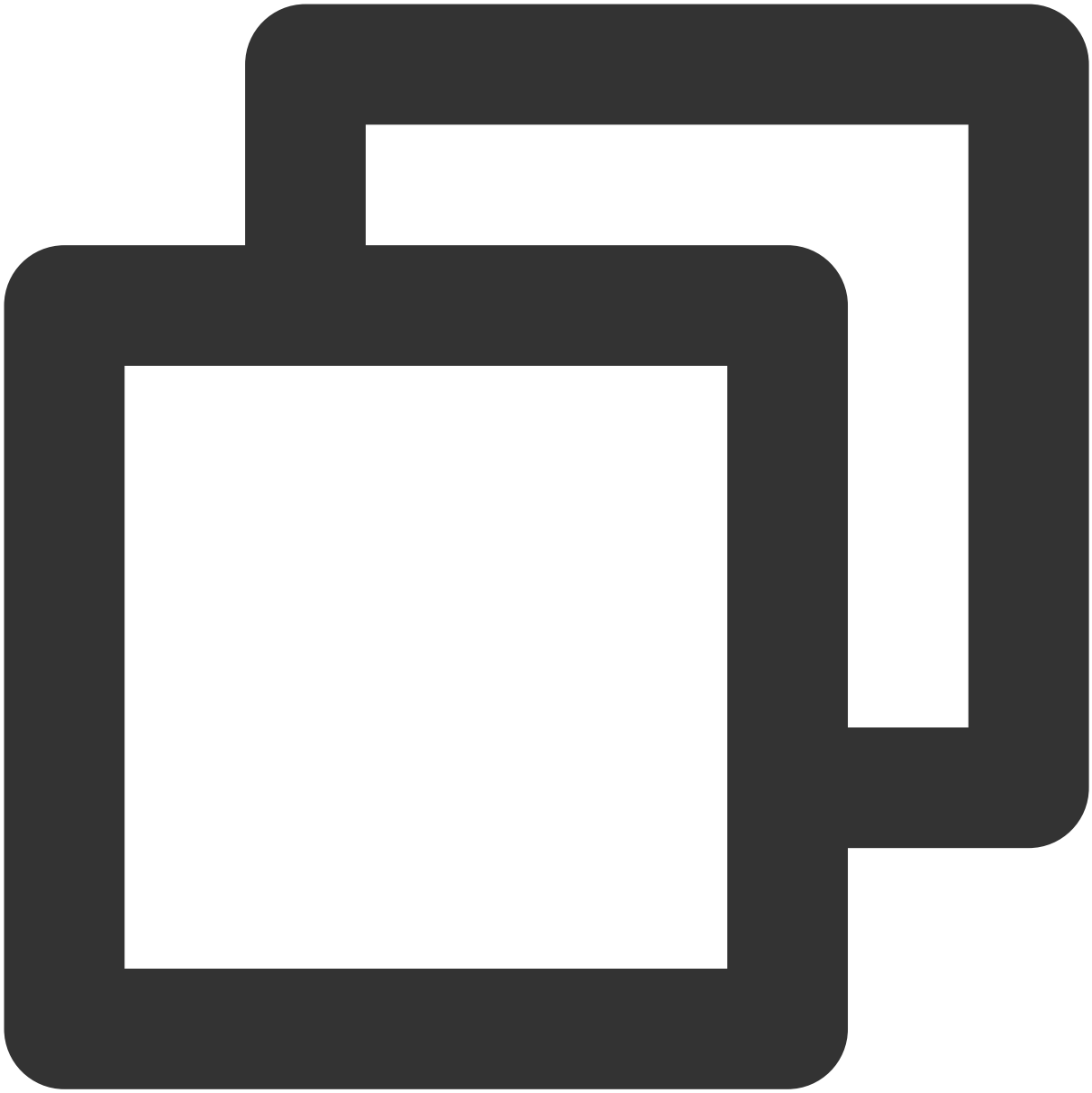
本番環境の依存関係をインストールします。



```
npm install element-plus events mitt pinia rtc-beauty-plugin tim-js-sdk trtc-js-sdk
```

2. **Piniaの登録**

TUIRoomはPiniaを使用してルームのデータ管理を行います。Piniaをプロジェクトエントリファイルに登録してください。プロジェクトエントリファイルは `src/main.ts` ファイルです。



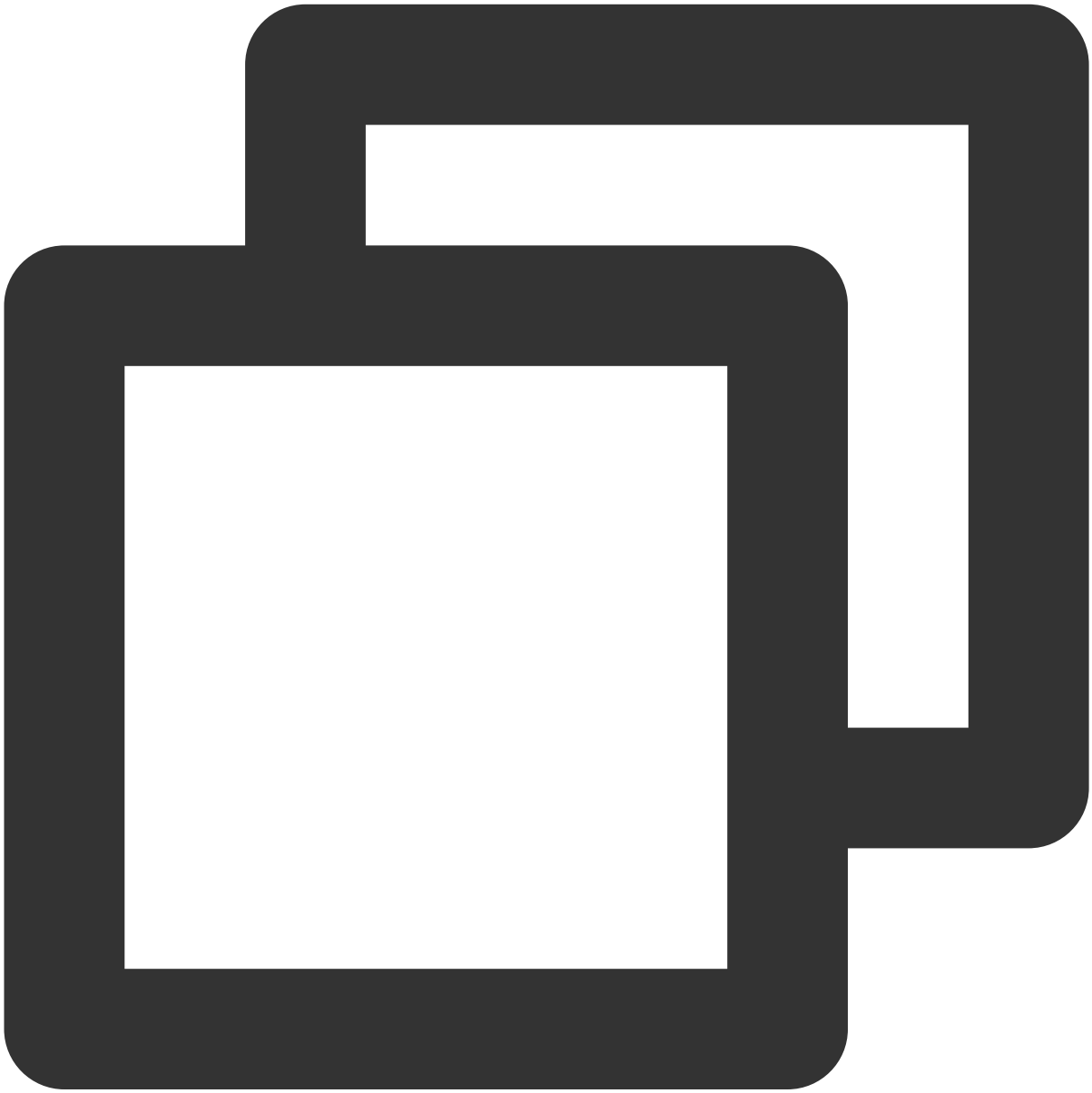
```
// src/main.tsファイル
import { createPinia } from 'pinia';

const app = createApp(App);
// pinia登録
app.use(createPinia());
app.mount('#app');
```

3. **element-plusをオンデマンドでインポートするための設定**

TUIRoomはelement-plus UIコンポーネントを使用します。すべてのelement-plusコンポーネントがインポートされないようにするには、 `vite.config.ts` でelement-plusコンポーネントをオンデマンドでロードするように設定してください。

ご注意：

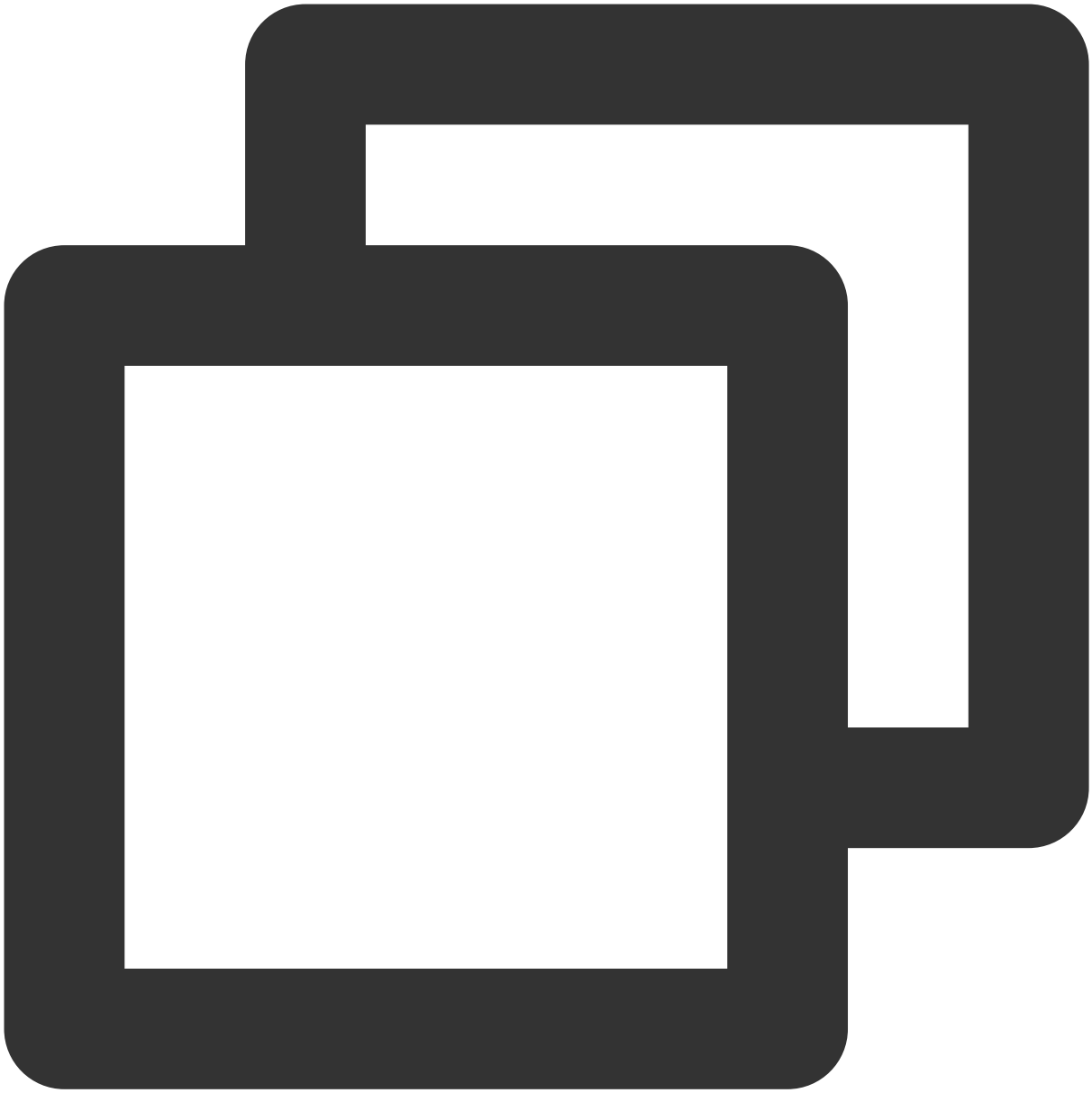
以下の設定項目は増分設定です。すでに存在するVite設定項目を削除しないでください。



```
// vite.config.ts
import AutoImport from 'unplugin-auto-import/vite';
import Components from 'unplugin-vue-components/vite';
import { ElementPlusResolver } from 'unplugin-vue-components/resolvers';
```

```
export default defineConfig({
    // ...
    plugins: [
        AutoImport({
            resolvers: [ElementPlusResolver()],
        }),
        Components({
            resolvers: [ElementPlusResolver({
                importStyle: 'sass',
            })],
        }),
    ],
    css: {
        preprocessorOptions: {
            scss: {
                // ...
                additionalData: `
                    @use "./src/TUIRoom/assets/style/element.scss" as *;
                `,
            },
        },
    },
});
```

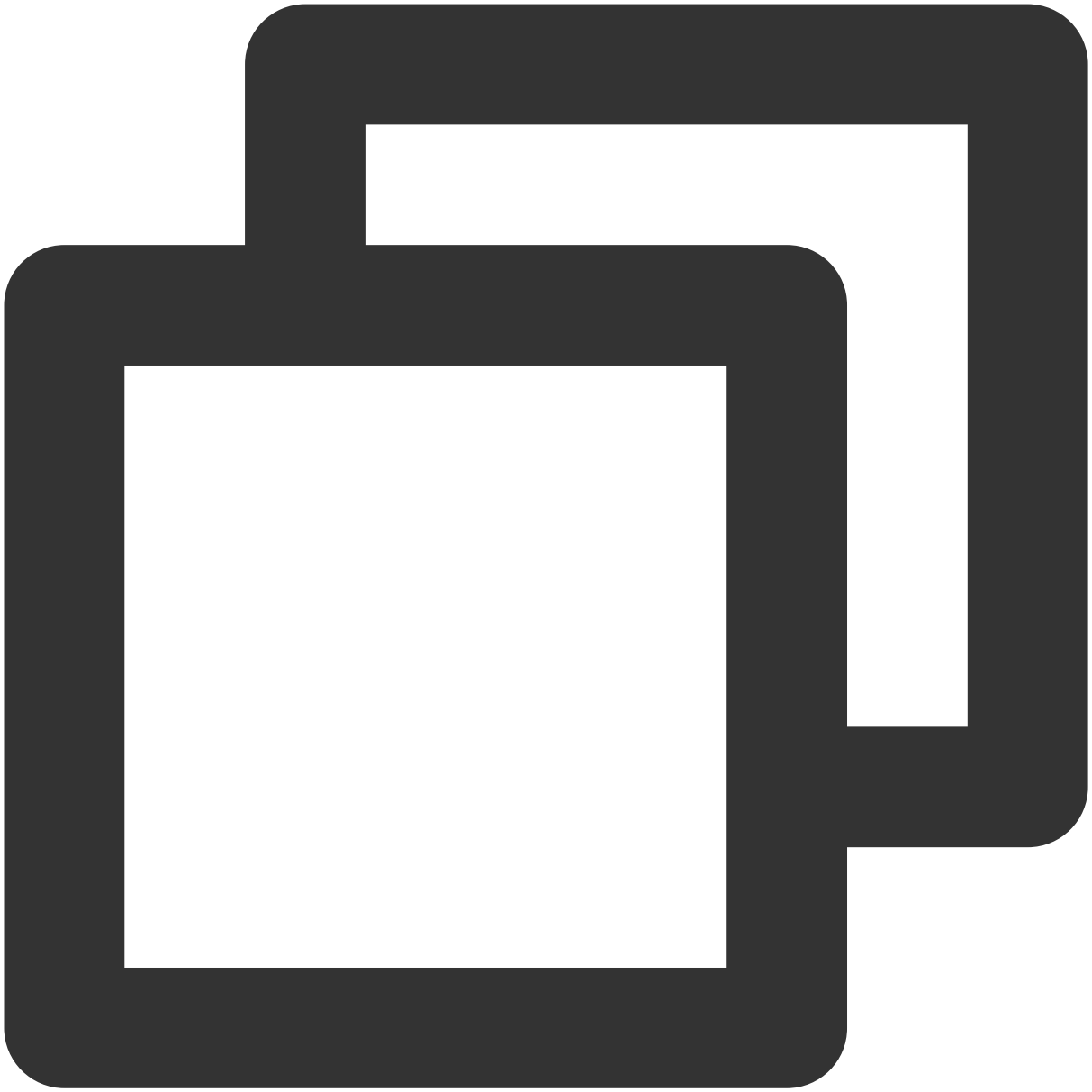
また、UI付きのelement-plus コンポーネントがスタイルを正しく表示するためには、エントリファイル `src/main.ts` にelement-plusコンポーネントスタイルをロードしてください。



```
// src/main.ts
import 'element-plus/theme-chalk/el-message.css';
import 'element-plus/theme-chalk/el-message-box.css';
```

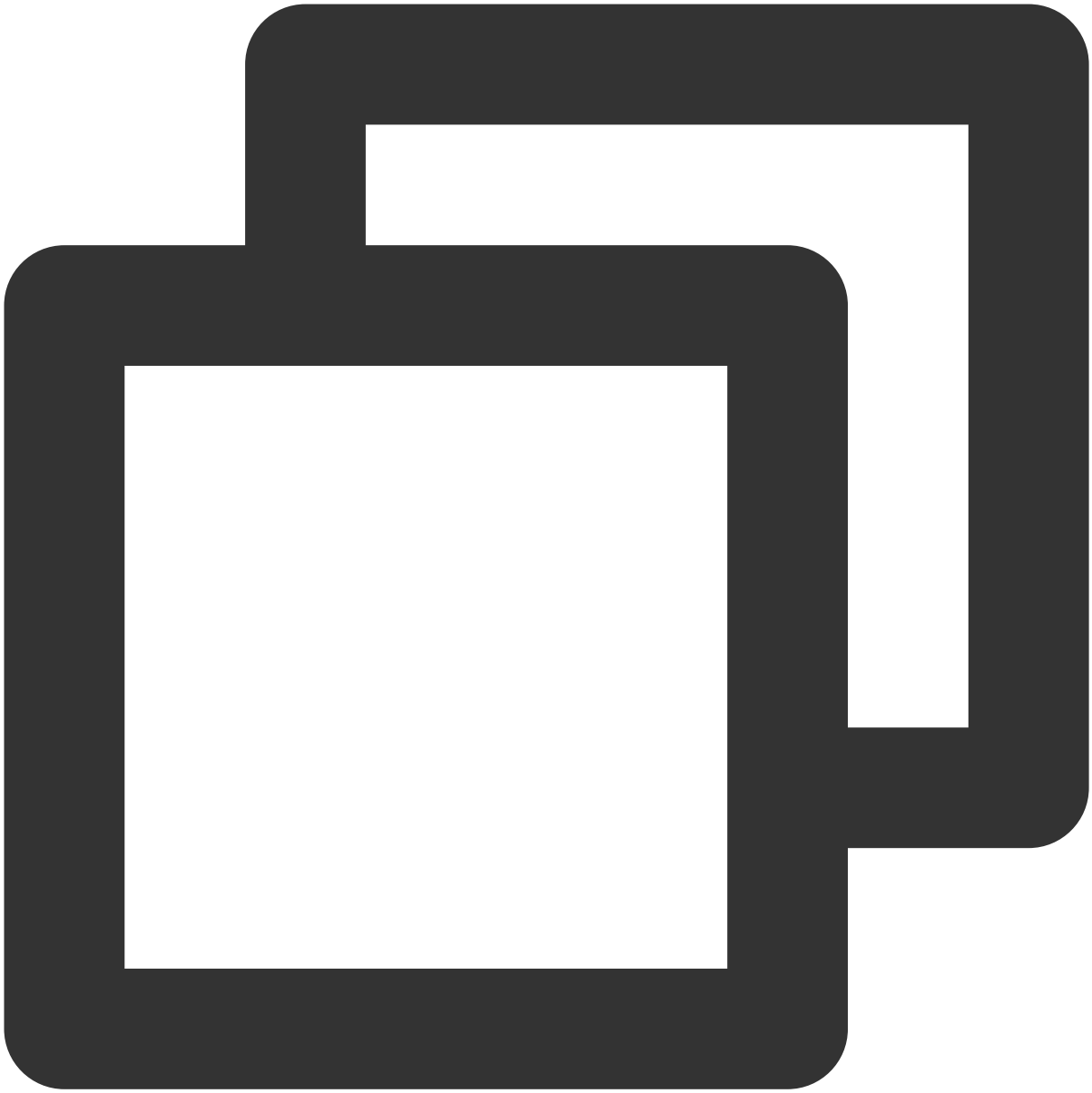
1. **依存関係のインストール**

開発環境の依存関係をインストールします。



```
npm install sass sass-loader typescript unplugin-auto-import unplugin-vue-component
```

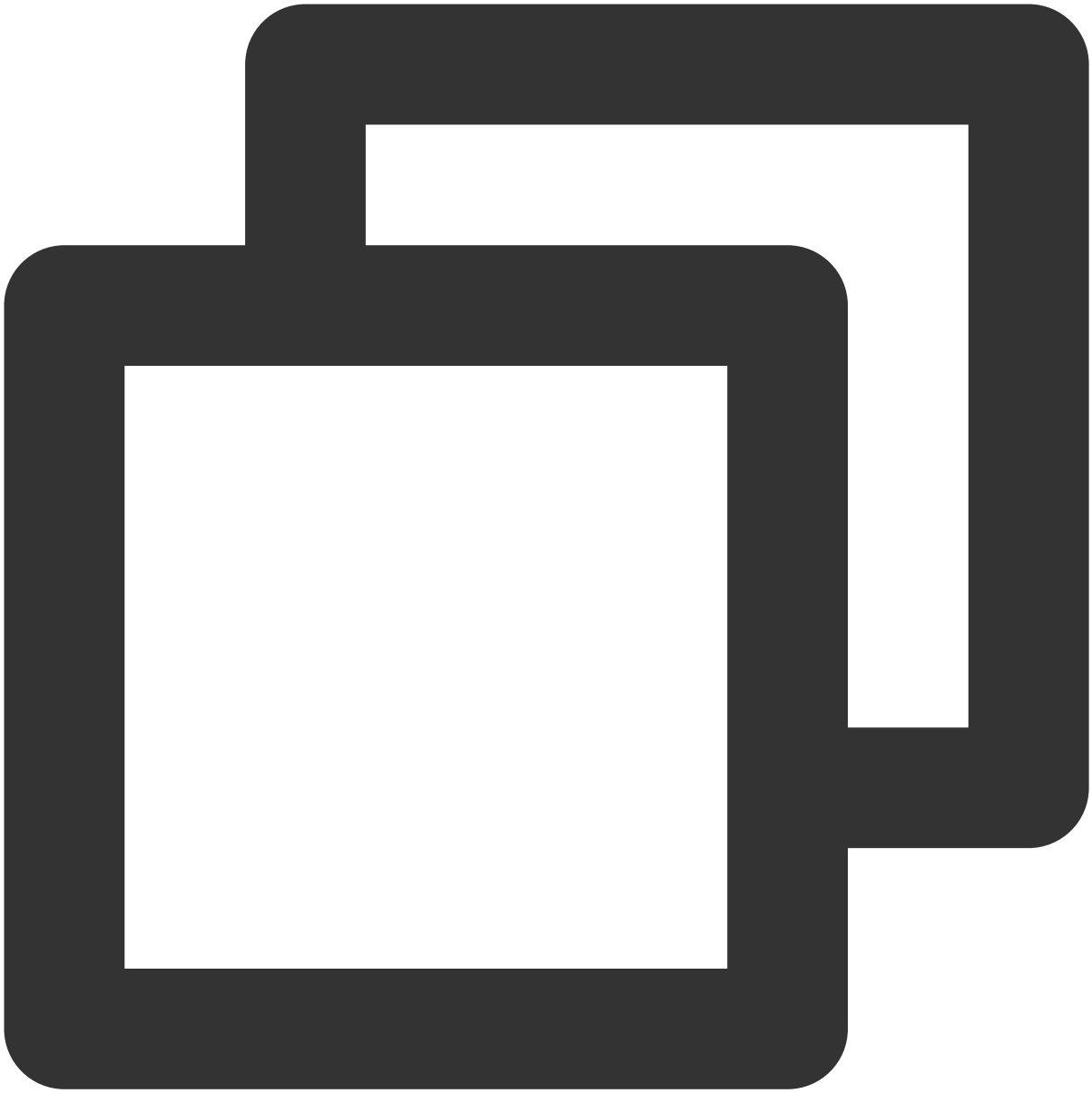
本番環境の依存関係をインストールします。



```
npm install element-plus events mitt pinia rtc-beauty-plugin tim-js-sdk trtc-js-sdk
```

2. **Piniaの登録**

TUIRoomはPiniaを使用してルームのデータ管理を行います。Piniaをプロジェクトエントリファイルに登録してください。プロジェクトエントリファイルは `src/main.ts` ファイルです。



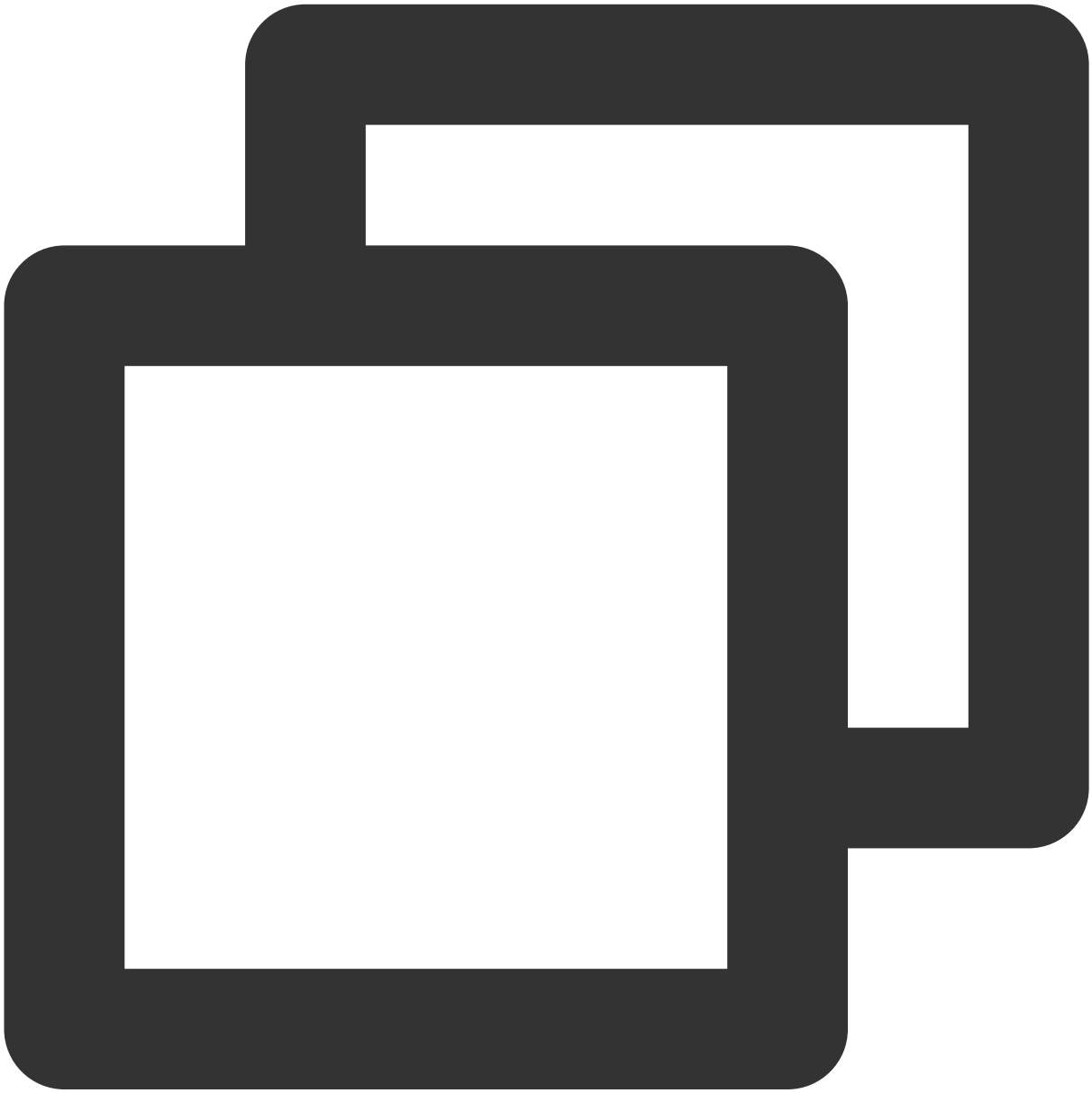
```
// src/main.tsファイル
import { createPinia } from 'pinia';

const app = createApp(App);
// pinia登録
app.use(createPinia());
app.mount('#app');
```

3. **element-plusをオンデマンドでインポートするための設定**

TUIRoomはelement-plus UIコンポーネントを使用します。すべてのelement-plusコンポーネントをインポートしてしまわないように、 `vue.config.js` で、element-plusコンポーネントを必要に応じてロードするように設定しておく必要があります。

ご注意：

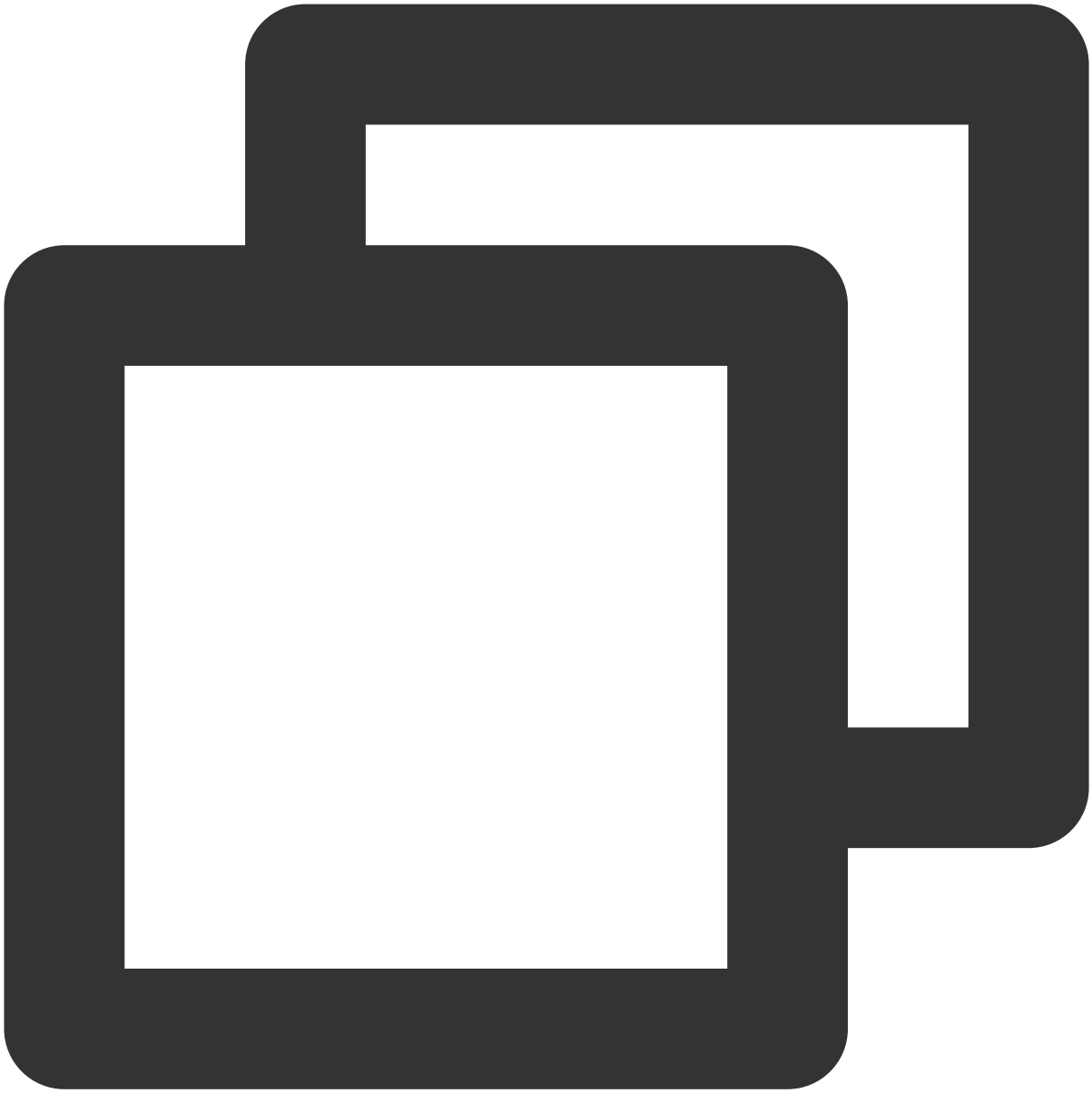
以下の設定項目は増分設定です。すでに存在するvue.config.js設定項目を削除しないでください。



```
// vue.config.js
const { defineConfig } = require('@vue/cli-service')
const AutoImport = require('unplugin-auto-import/webpack')
const Components = require('unplugin-vue-components/webpack')
```

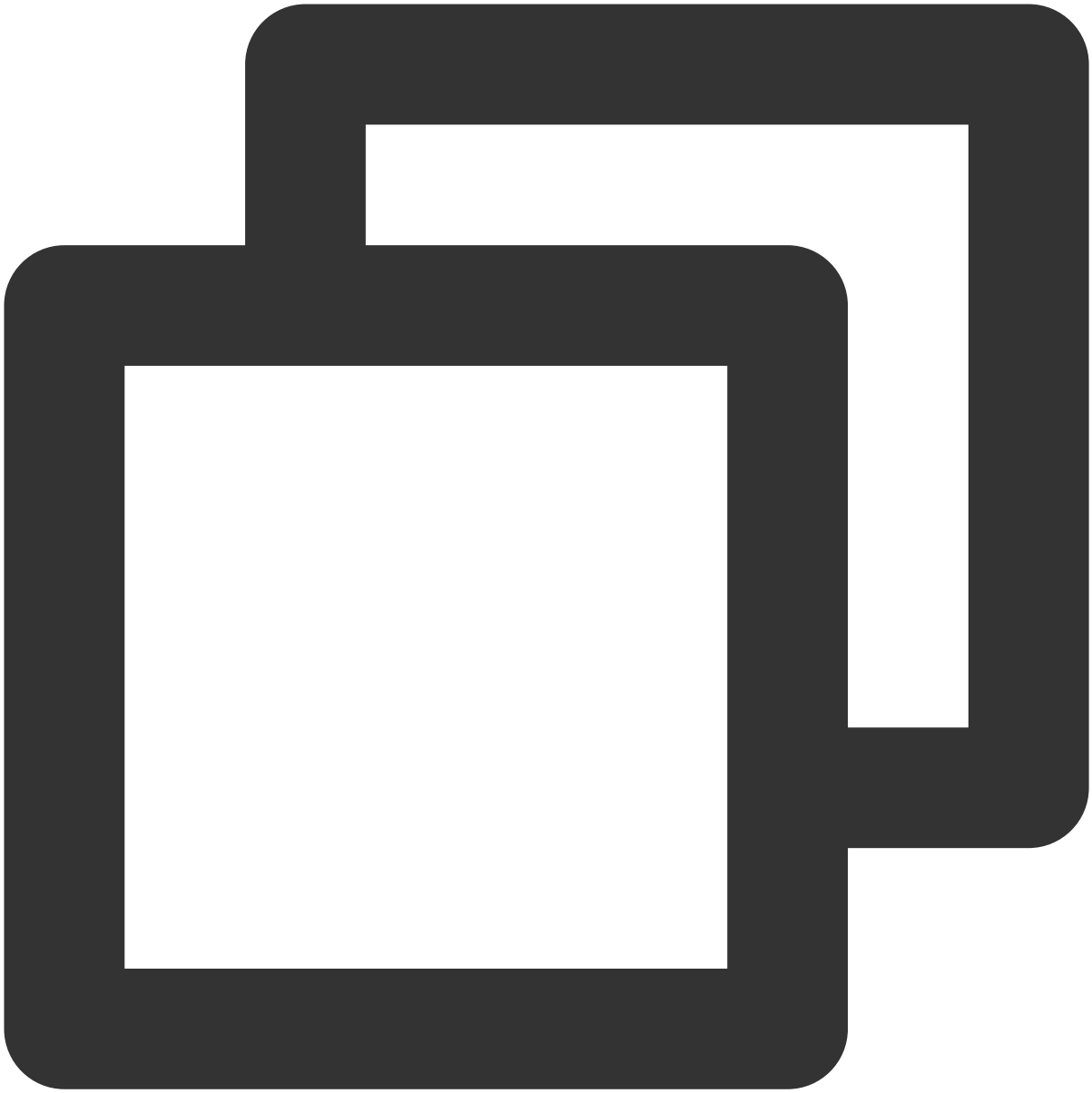
```
const { ElementPlusResolver } = require('unplugin-vue-components/resolvers')
const ElementPlus = require('unplugin-element-plus/webpack')

module.exports = defineConfig({
  transpileDependencies: true,
  css: {
    loaderOptions: {
      scss: {
        additionalData: '@use "./src/TUIRoom/assets/style/element.scss" as *;'
      }
    }
  },
  configureWebpack: {
    plugins: [
      AutoImport({
        resolvers: [ElementPlusResolver({ importStyle: 'sass' })]
      }),
      Components({
        resolvers: [ElementPlusResolver({ importStyle: 'sass' })]
      }),
      // 必要に応じてインポート時のテーマカラーをカスタマイズ
      ElementPlus({
        useSource: true
      })
    ]
  }
})
```

また、UI付きのelement-plus コンポーネントがスタイルを正しく表示するためには、エントリファイル `src/main.ts` にelement-plusコンポーネントスタイルをロードしてください。



```
// src/main.ts
import 'element-plus/theme-chalk/el-message.css';
import 'element-plus/theme-chalk/el-message-box.css';
```

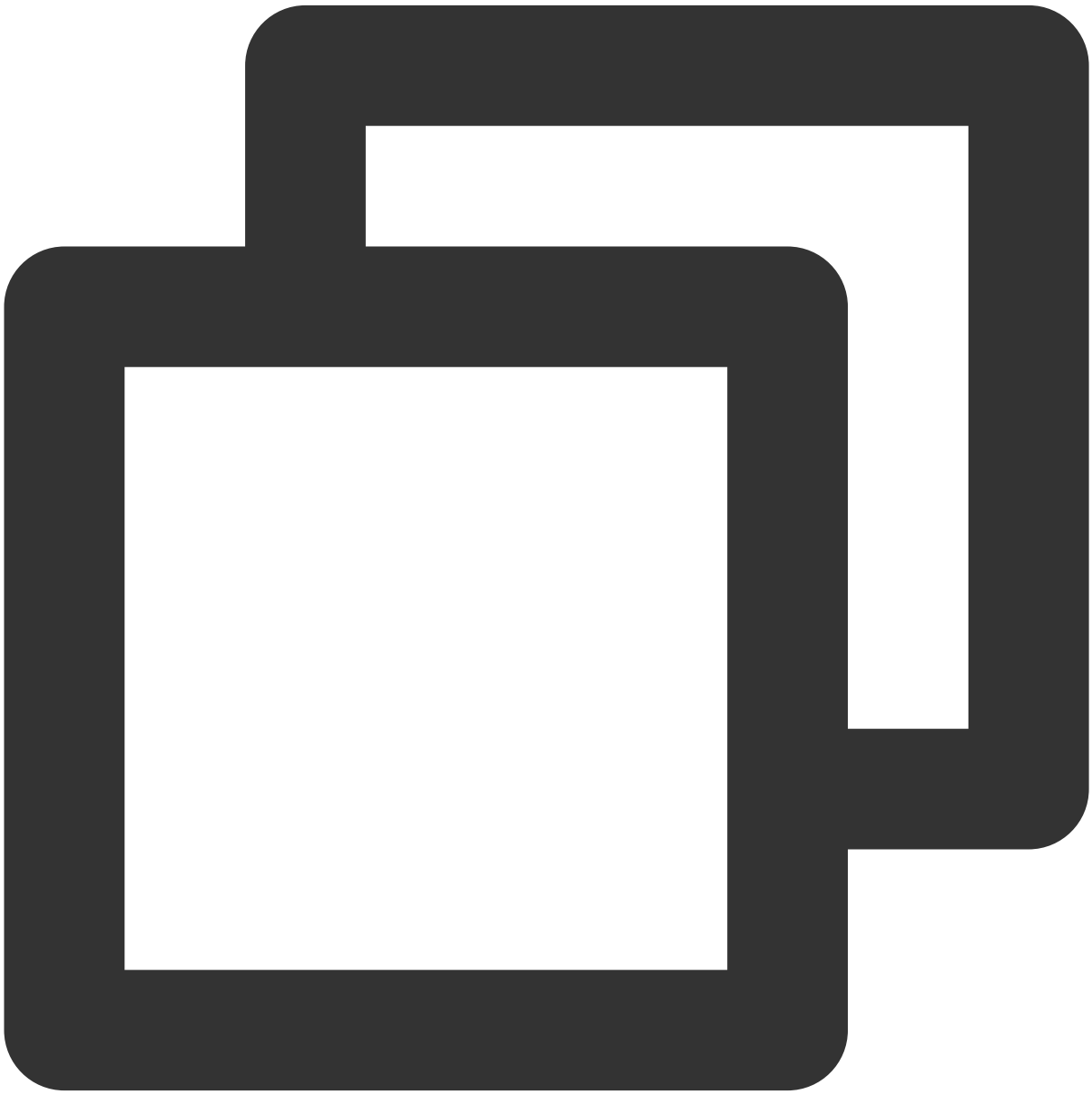
4. **ts宣言ファイルの設定** `src/shims-vue.d.ts` ファイルに以下の設定を追加します。



```
declare module 'tsignaling/tsignaling-js' {
  import TSignaling from 'tsignaling/tsignaling-js';
  export default TSignaling;
}

declare module 'tim-js-sdk' {
  import TIM from 'tim-js-sdk';
  export default TIM;
}

declare const Aegis: any;
```

ステップ5：開発環境の実行

コンソールで開発環境実行スクリプトを実行し、ブラウザを使用してTUIRoomを含むページを開くと、ページ内でTUIRoomコンポーネントを使用できます。
ステップ2のスクリプトを使用してVue + Vite + TSプロジェクトを生成する場合は、次の事項を行う必要があります。
1. 開発環境コマンドを実行します。



```
npm run dev
```

2. ブラウザでページ `http://localhost:3000/` を開きます。

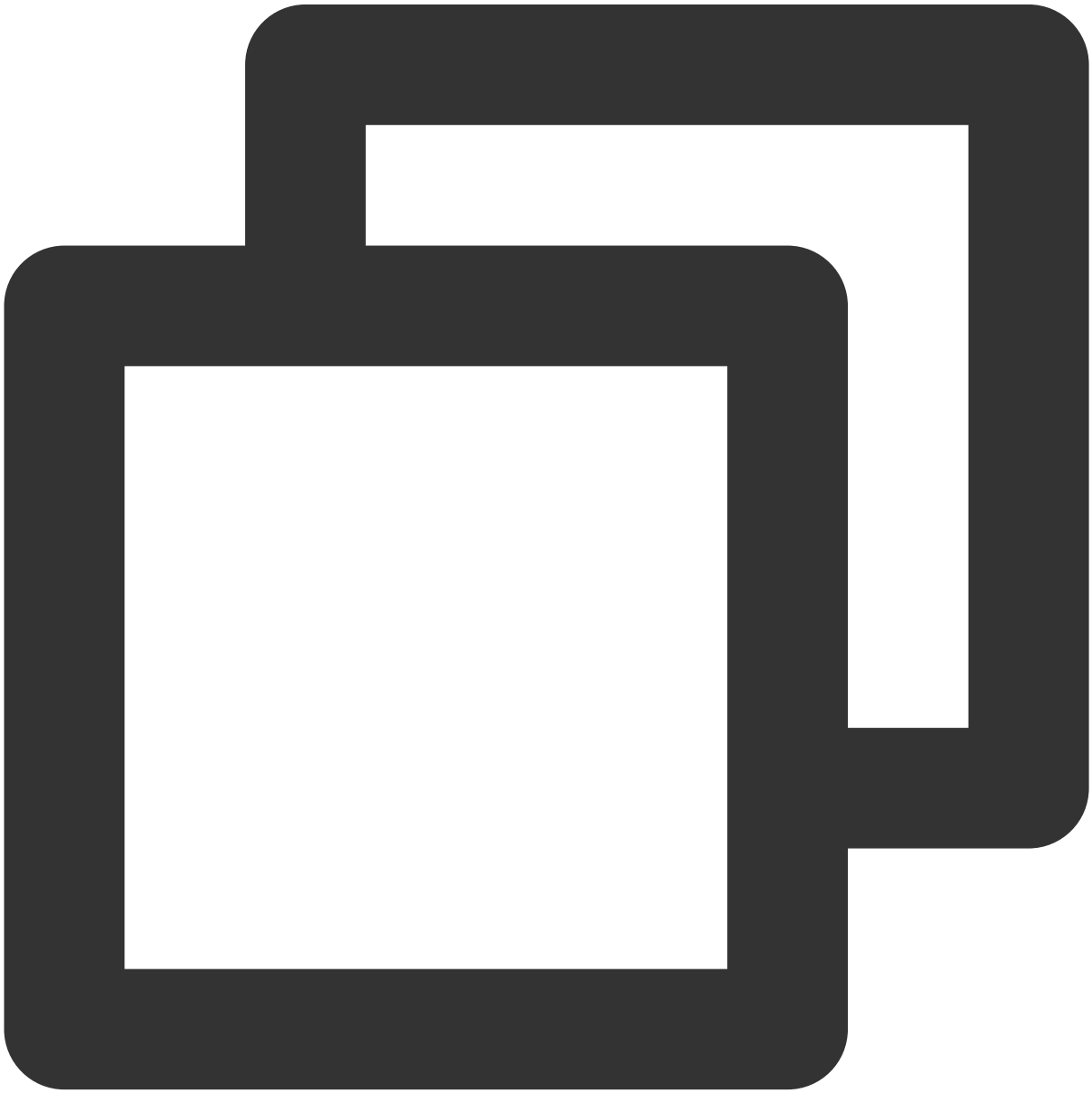
ご注意：

TUIRoomはオンデマンドでelement-plusコンポーネントをインポートするため、開発環境のルーティングページが最初にロードされたときの反応が遅くなり、element-plusがオンデマンドでロードされるのを待つと正常に機能するようになります。element-plusのオンデマンドロードは、パッケージング後のページロードには影響しません。

3. TUIRoomコンポーネント機能を体験します。

ステップ2のスクリプトを使用してVue + Webpack + TSプロジェクトを生成する場合は、次の事項を行う必要があります。

1. 開発環境コマンドを実行します。



```
npm run serve
```

2. ブラウザで `http://localhost:8080/` のページを開きます

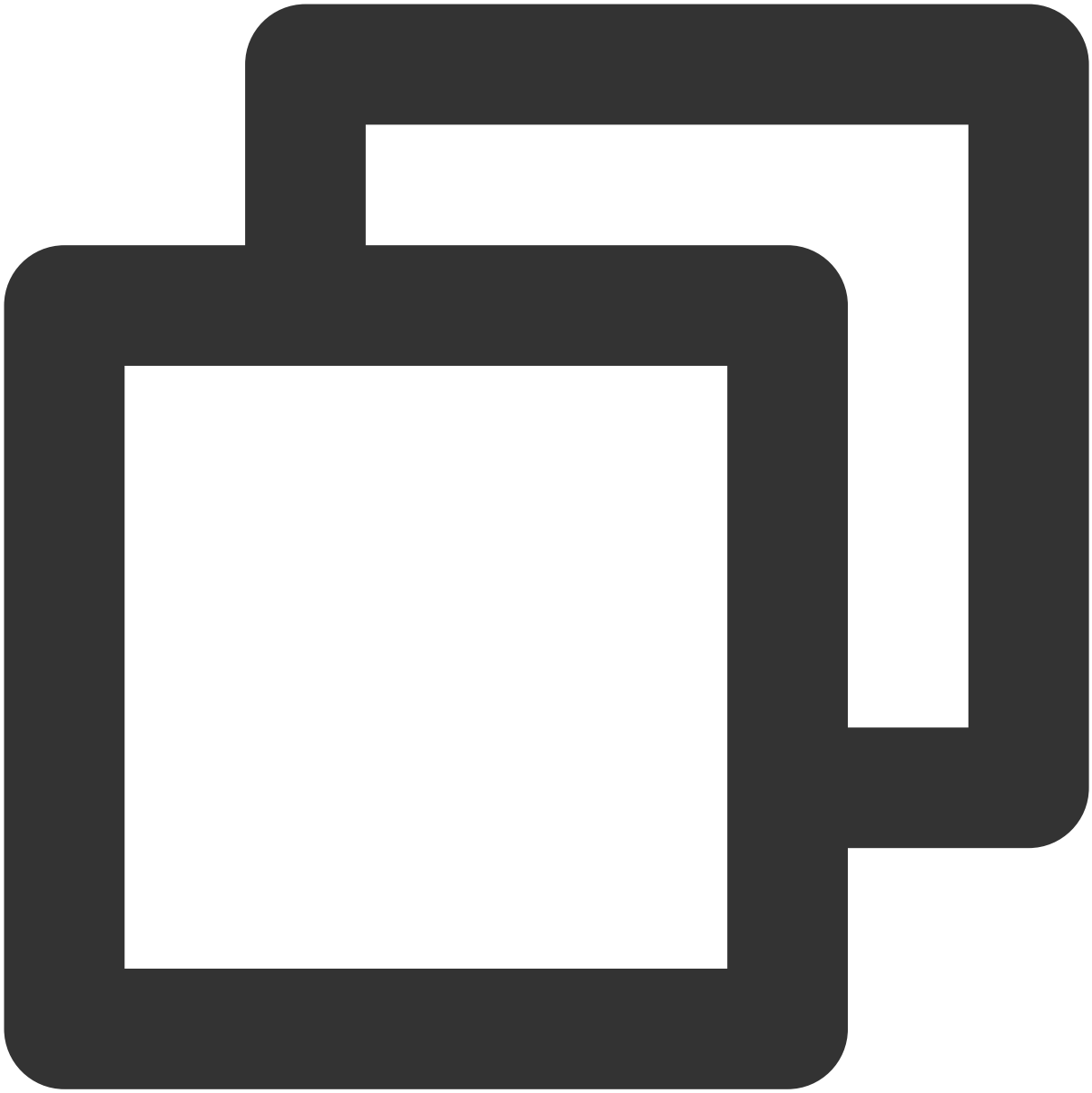
ご注意：

実行中、src/TUIRoomディレクトリでeslintエラーが発生した場合、.eslintignoreファイルに/src/TUIRoomのパスを追加すると、eslintチェックをブロックできます。

3. TUIRoomコンポーネント機能を体験します。

ステップ6：本番環境のデプロイ

1. distファイルのパッケージ



```
npm run build
```

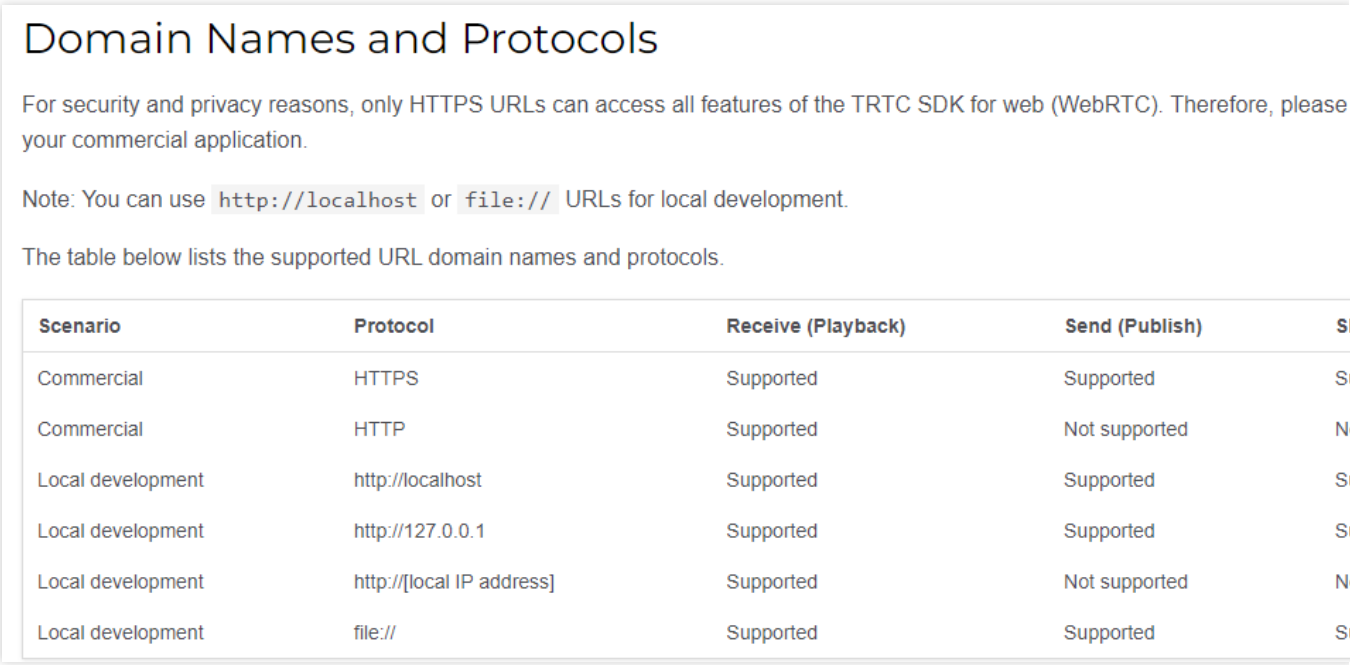
説明：

実際のパッケージコマンドについてはpackage.jsonファイルをご確認ください。

2. distファイルのサーバーへのデプロイ

ご注意：

本番環境ではHTTPSドメイン名を使用する必要があります。



Domain Names and Protocols

For security and privacy reasons, only HTTPS URLs can access all features of the TRTC SDK for web (WebRTC). Therefore, please your commercial application.

Note: You can use `http://localhost` or `file://` URLs for local development.

The table below lists the supported URL domain names and protocols.

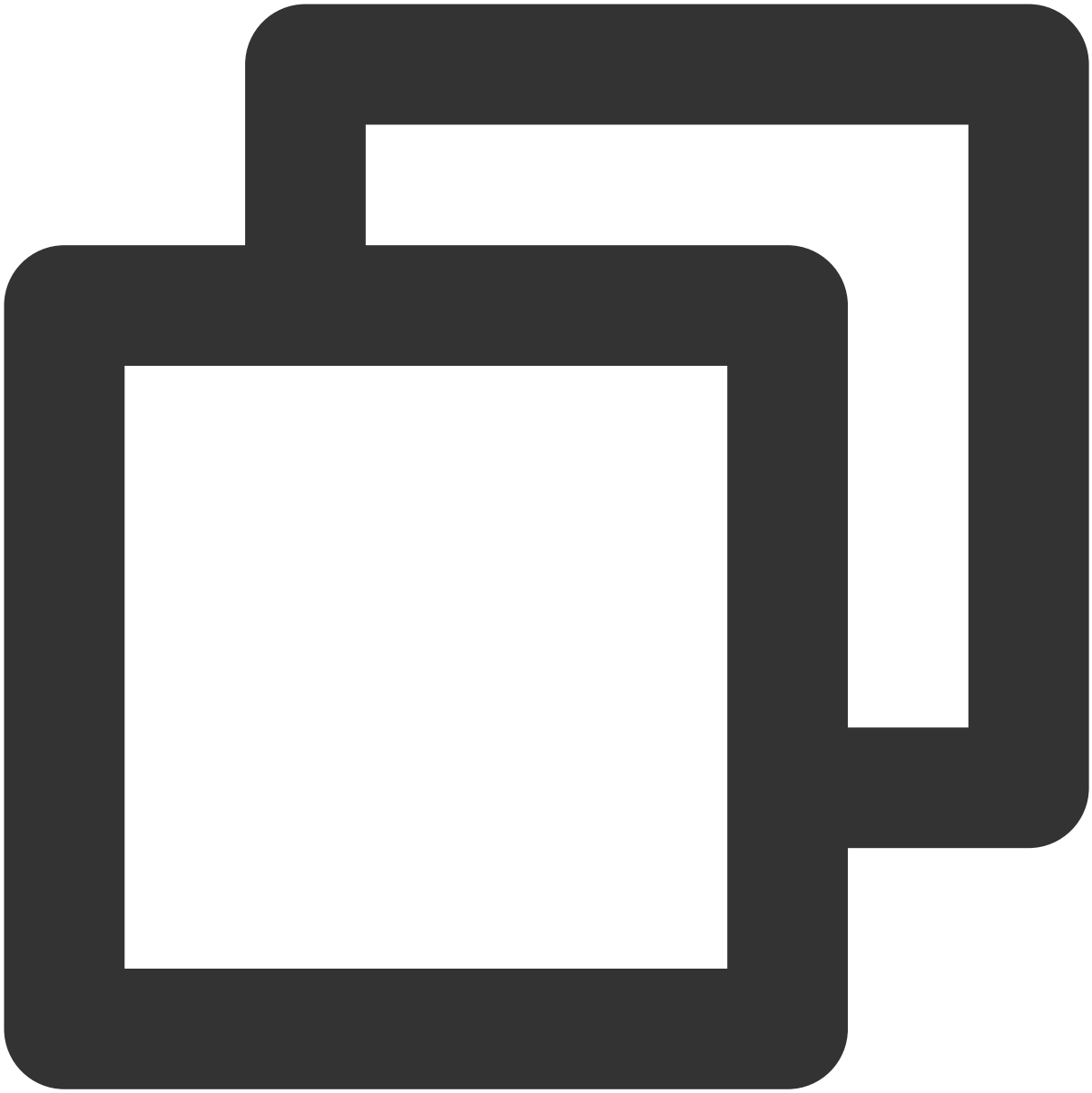
Scenario	Protocol	Receive (Playback)	Send (Publish)	S
Commercial	HTTPS	Supported	Supported	S
Commercial	HTTP	Supported	Not supported	N
Local development	http://localhost	Supported	Supported	S
Local development	http://127.0.0.1	Supported	Supported	S
Local development	http://[local IP address]	Supported	Not supported	N
Local development	file://	Supported	Supported	S

付録：TUIRoom API

TUIRoomインターフェース

init

TUIRoomデータを初期化します。TUIRoomを使用するすべてのユーザーはinitメソッドを呼び出してください。



```
TUIRoomRef.value.init(roomData);
```

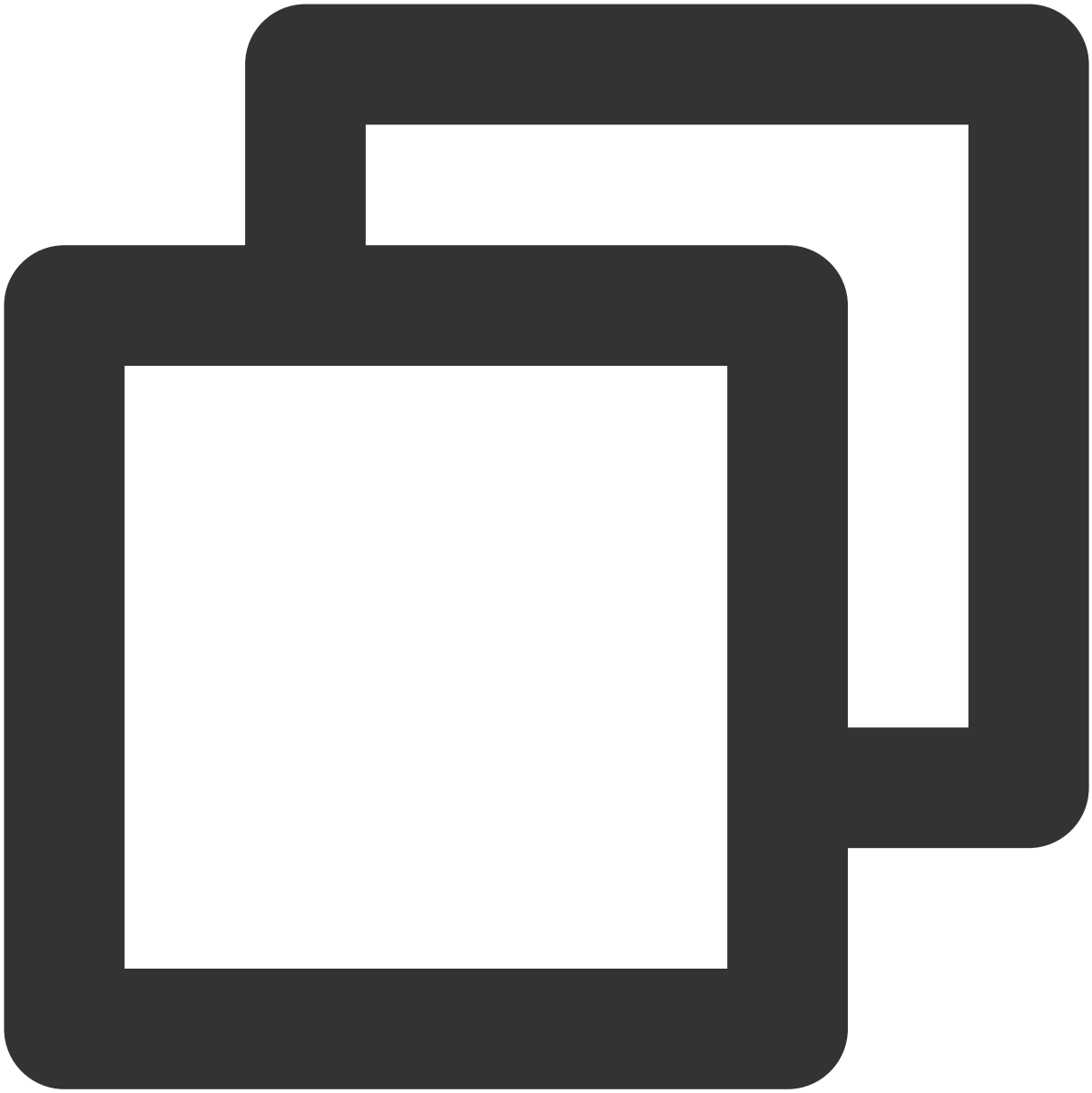
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
roomData	object	
roomData.sdkAppId	number	お客様のSDKAppId
roomData.userId	string	ユーザーの唯一ID

roomData.userSig	string	ユーザーのUserSig
roomData.userName	string	ユーザーのニックネーム
roomData.userAvatar	string	ユーザーのプロフィール写真
roomData.shareUserId	string	非必須。ユーザーが画面共有を行うUserIdです。 shareUserId= `share_${userId}` が要求され、画面共有機能がない場合は渡されません
roomData.shareUserSig	string	非必須。ユーザーが画面共有を行うUserSig

createRoom

キャスターがルームを作成します。



```
TUIRoomRef.value.createRoom(roomId, roomMode, roomParam);
```

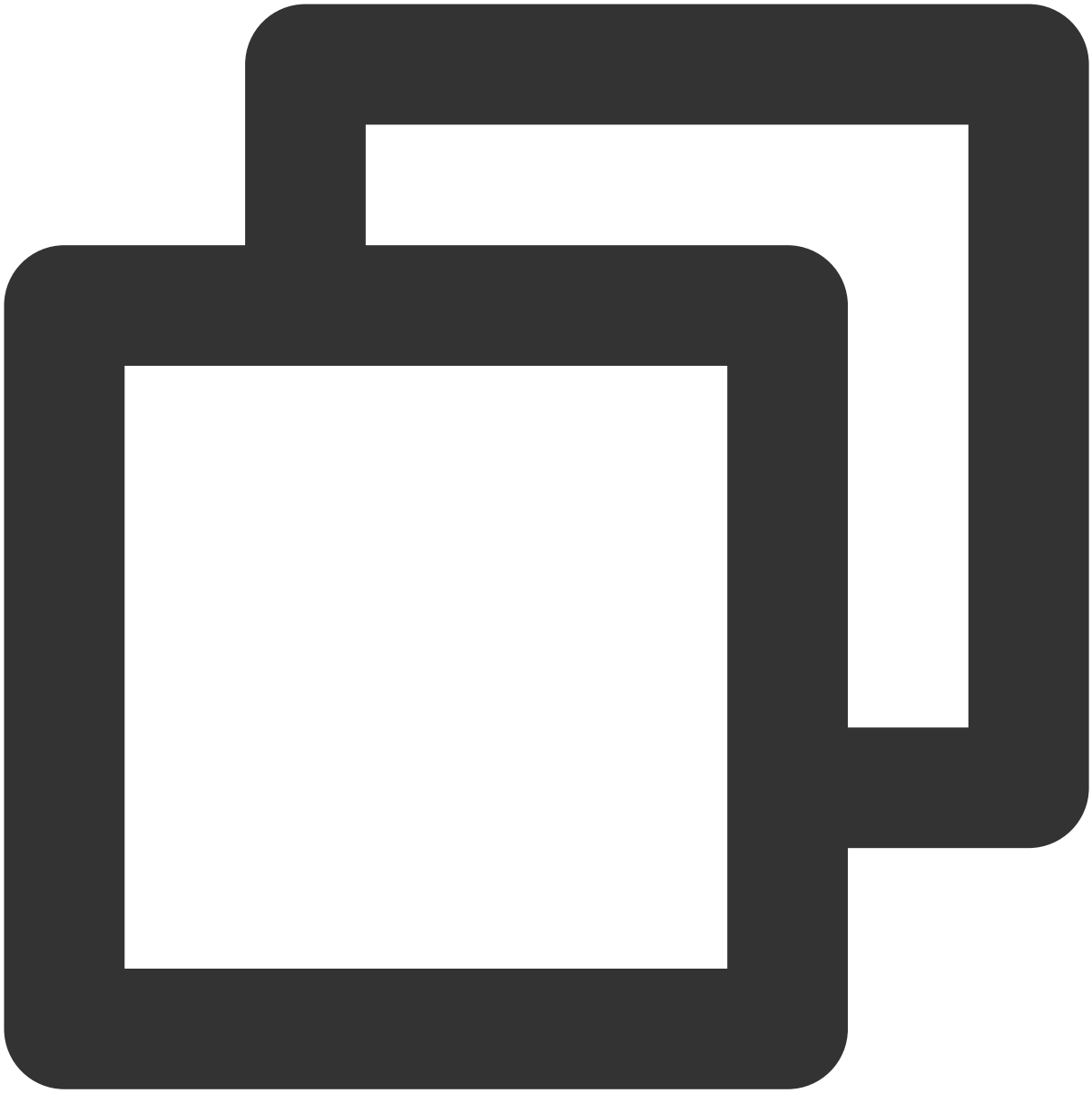
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
roomId	number	ルームID
roomMode	string	ルームモードです。'FreeSpeech'（自由発言モード）と'ApplySpeech'（挙手発言モード）を含み、デフォルト

		は'FreeSpeech'です。現在、自由発言モードだけをサポートします
roomParam	Object	非必須
roomParam.isOpenCamera	string	非必須。入室はカメラをオンにするかどうか。デフォルトはオフです
roomParam.isOpenMicrophone	string	非必須。入室はマイクをオンにするかどうか。デフォルトはオフです
roomParam.defaultCameraId	string	非必須。デフォルトはカメラ装置のID
roomParam.defaultMicrophoneId	string	非必須。デフォルトはマイク装置のID
roomParam.defaultSpeakerId	String	非必須。デフォルトはスピーカー装置のID

enterRoom

一般メンバーがルームに参加します。



```
TUIRoomRef.value.enterRoom(roomId, roomParam);
```

パラメータは下表に示すとおりです。

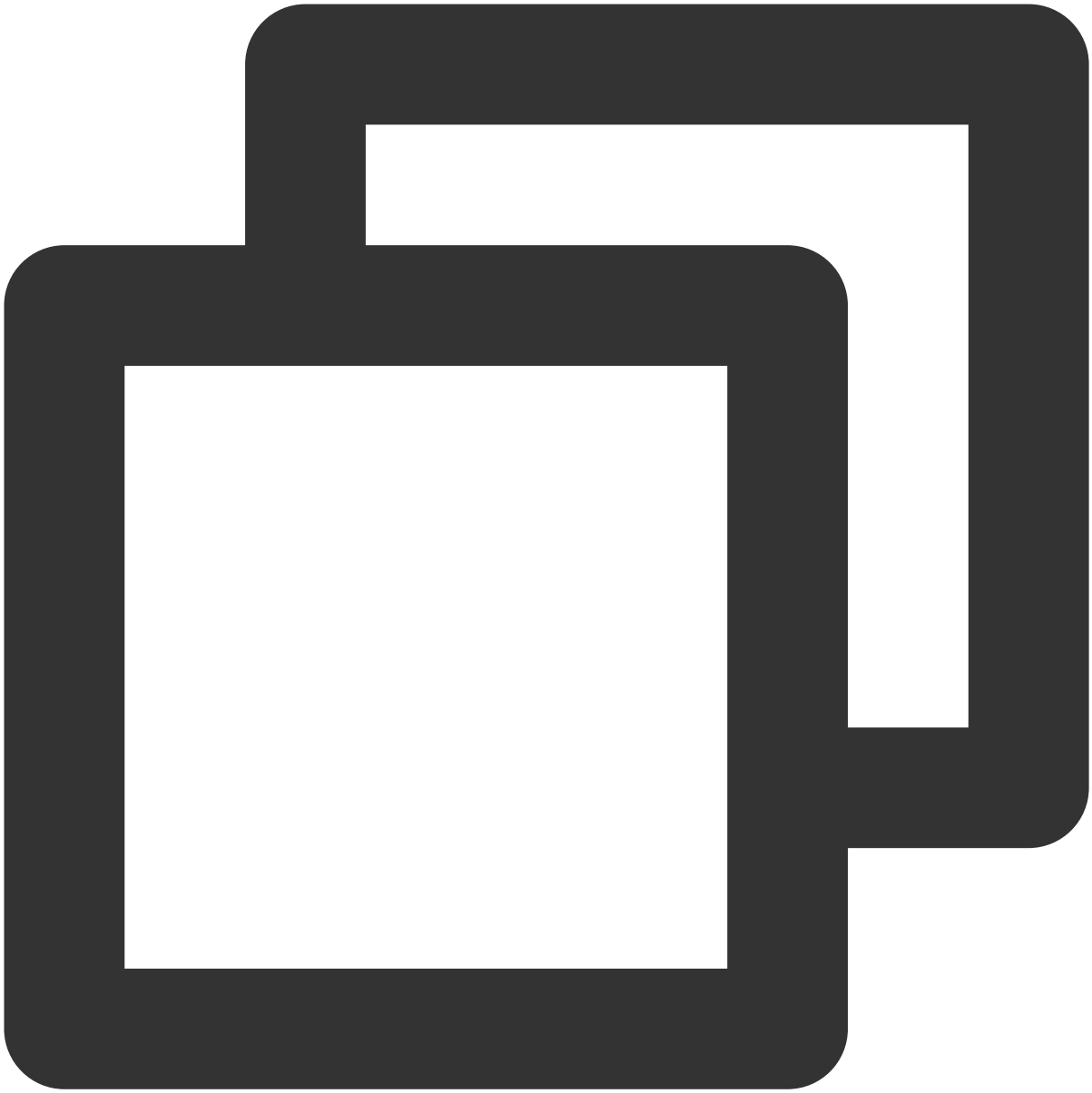
パラメータ	タイプ	意味
roomId	number	ルームID
roomParam	Object	非必須
roomParam.isOpenCamera	string	非必須。入室はカメラをオンにするかどうか。デフォルトは

		オフです
roomParam.isOpenMicrophone	string	非必須。入室はマイクをオンにするかどうか。デフォルトはオフです
roomParam.defaultCameraId	string	非必須。デフォルトはカメラ装置のID
roomParam.defaultMicrophoneId	string	非必須。デフォルトはマイク装置のID
roomParam.defaultSpeakerId	String	非必須。デフォルトはスピーカー装置のID

TUIRoomイベント

onRoomCreate

ルーム作成のコールバック。

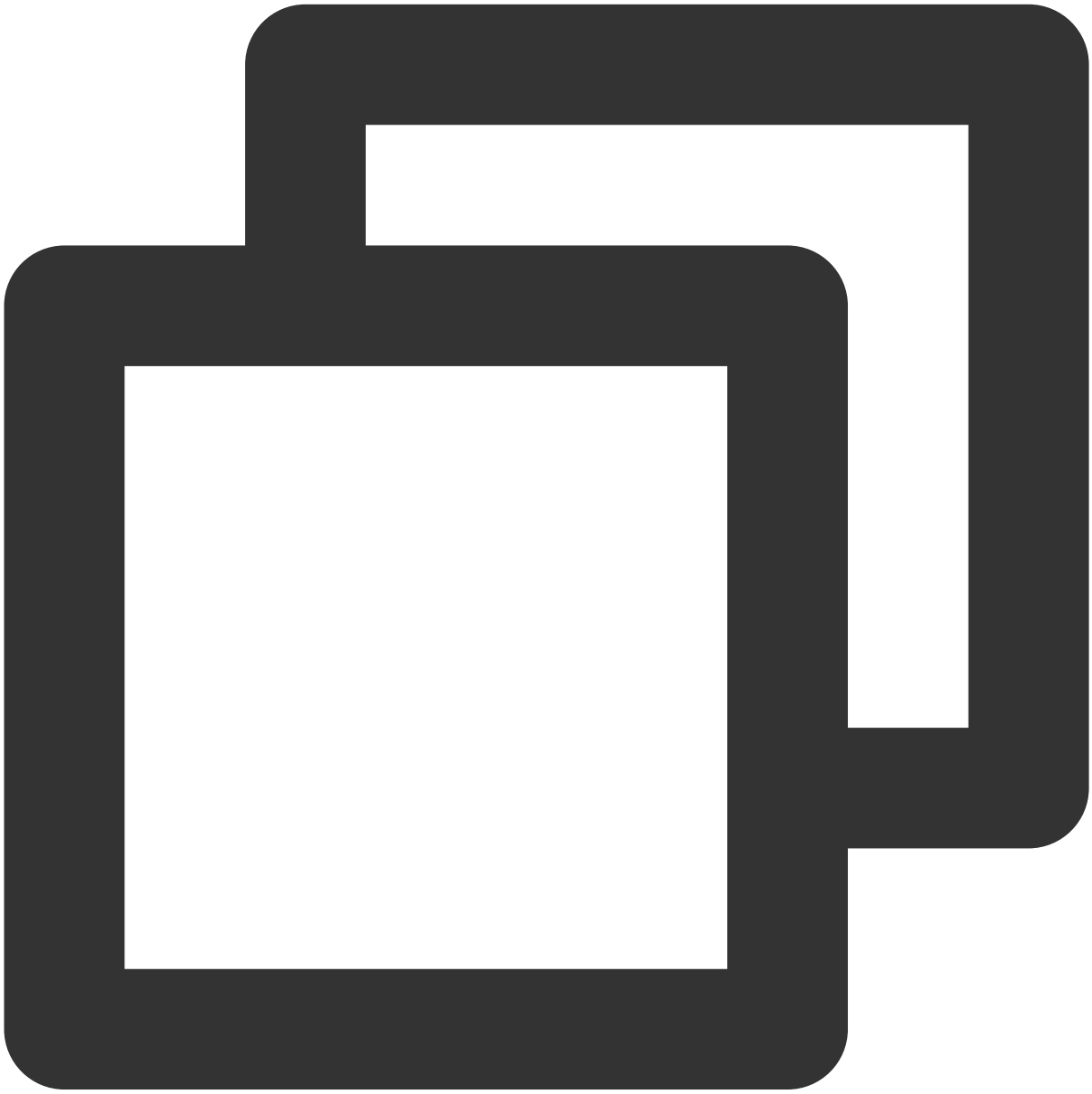


```
<template>
  <room ref="TUIRoomRef" @on-room-create="handleRoomCreate"></room>
</template>

<script setup lang="ts">
  // TUIRoomコンポーネントをインポートします。インポートパスが正しいことを確認してください
  import Room from './TUIRoom/index.vue';

  function handleRoomCreate(info) {
    if (info.code === 0) {
      console.log('ルーム作成成功')
```

```
    }
  }
</script>
```

onRoomEnter

入室コールバック。

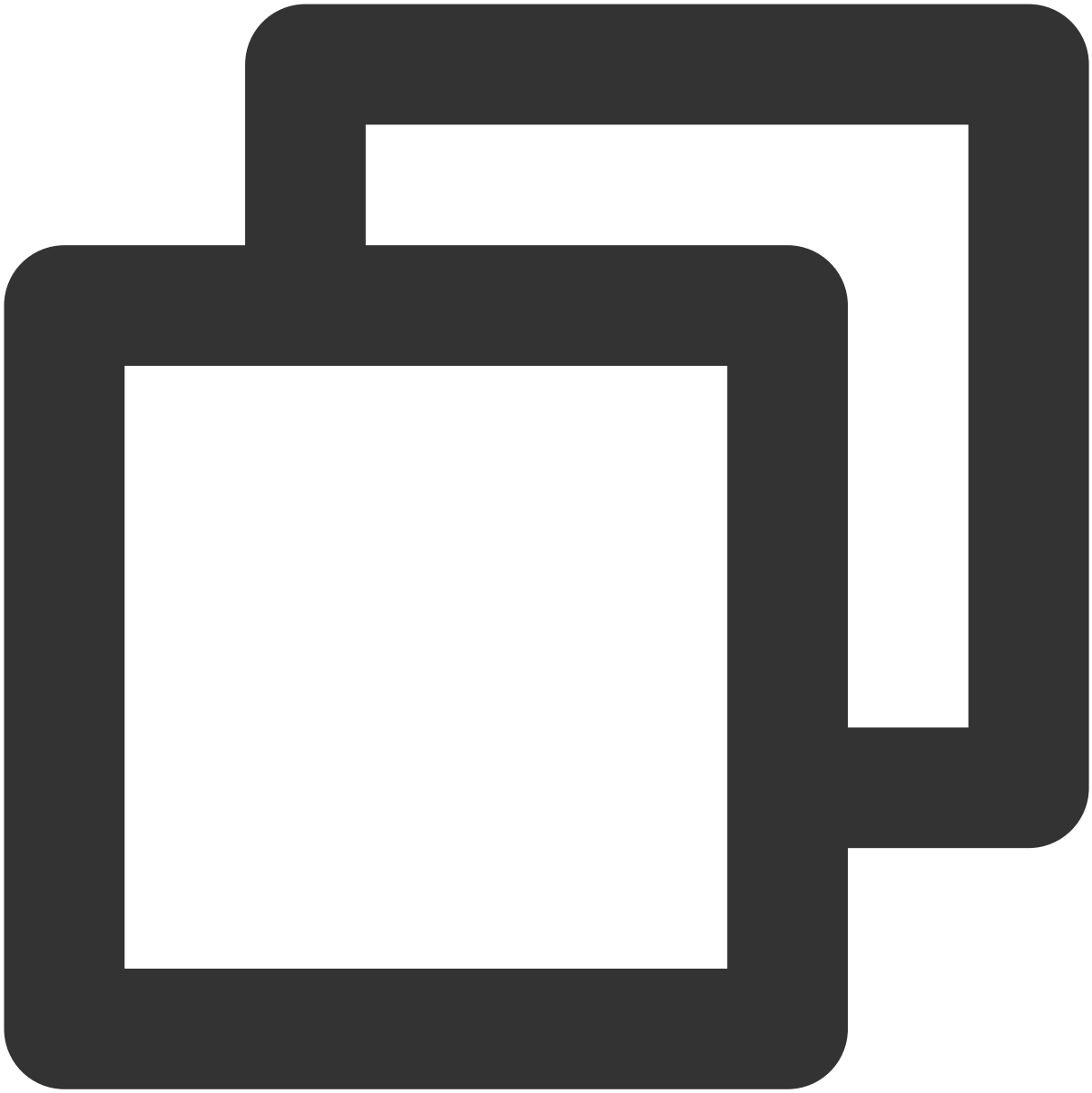


```
<template>
  <room ref="TUIRoomRef" @on-room-enter="handleRoomEnter"></room>
</template>
```

```
<script setup lang="ts">
  // TUIRoomコンポーネントをインポートします。インポートパスが正しいことを確認してください
  import Room from './TUIRoom/index.vue';

  function handleRoomEnter(info) {
    if (info.code === 0) {
      console.log('入室成功')
    }
  }
</script>
```

onRoomDestory

キャスターがルームを廃棄する通知。

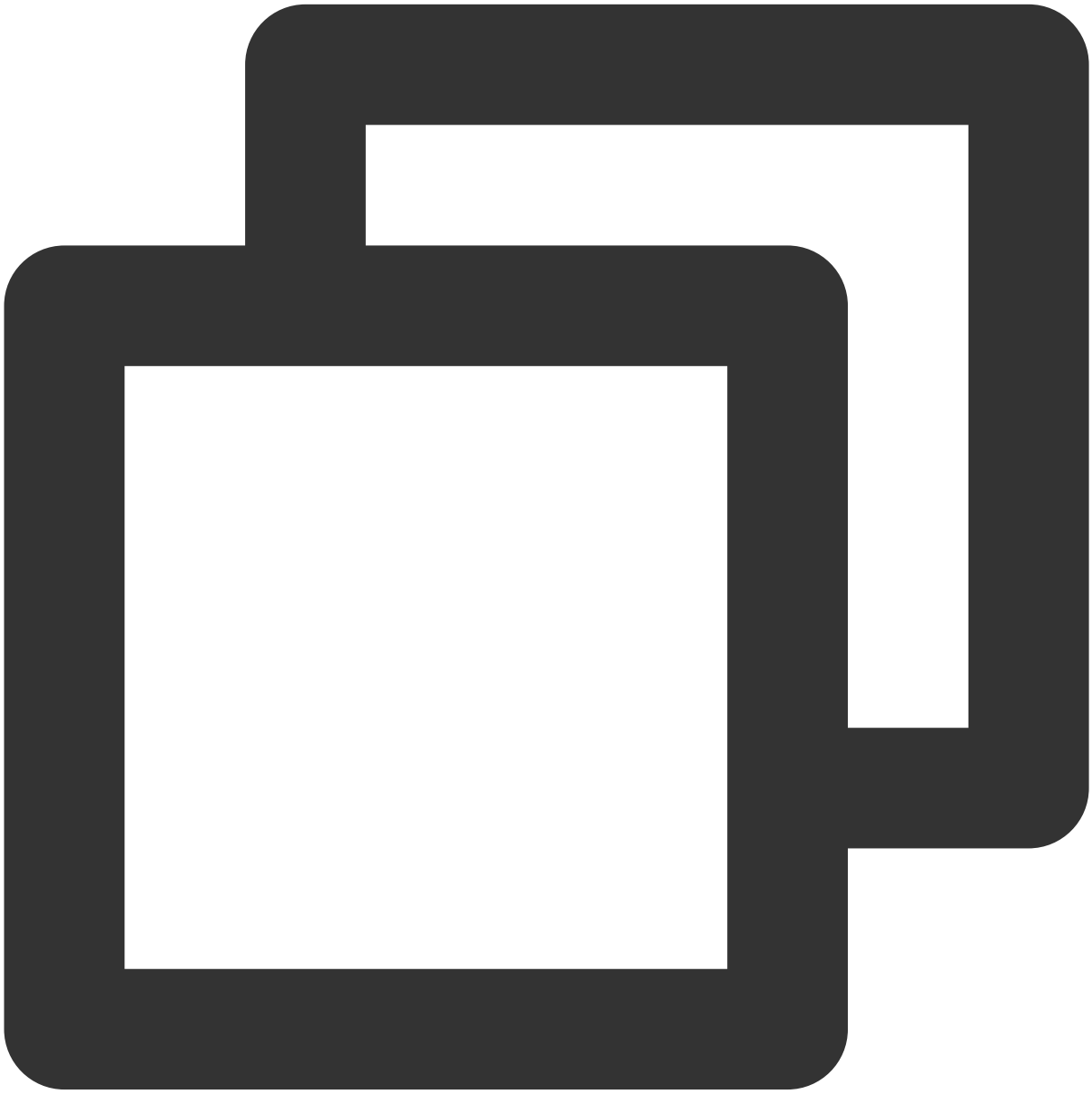


```
<template>
  <room ref="TUIRoomRef" @on-room-destory="handleRoomDestory"></room>
</template>

<script setup lang="ts">
  // TUIRoomコンポーネントをインポートします。インポートパスが正しいことを確認してください
  import Room from './TUIRoom/index.vue';

  function handleRoomDestory(info) {
    if (info.code === 0) {
      console.log('キャスターが廃棄することに成功しました')
```

```
    }
  }
</script>
```

onRoomExit

一般メンバーが退室する通知。

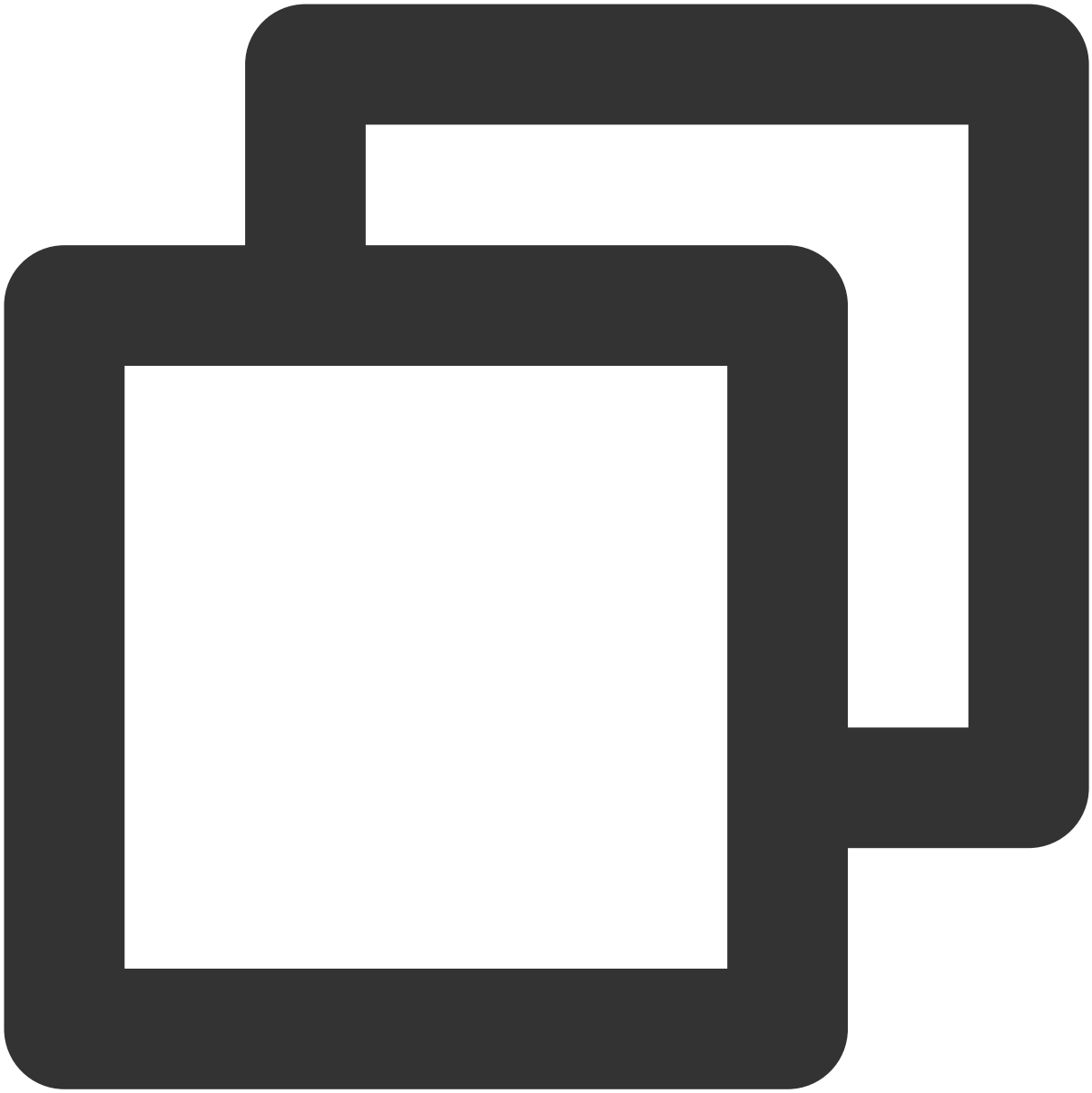


```
<template>
  <room ref="TUIRoomRef" @on-room-exit="handleRoomExit"></room>
</template>
```

```
<script setup lang="ts">
  // TUIRoomコンポーネントをインポートします。インポートパスが正しいことを確認してください
  import Room from './TUIRoom/index.vue';

  function handleRoomExit(info) {
    if (info.code === 0) {
      console.log('一般メンバーが退室することに成功しました')
    }
  }
</script>
```

onKickOff

一般メンバーがキャスターにルームからキックアウトされたときの通知です。



```
<template>
  <room ref="TUIRoomRef" @on-kick-off="handleKickOff"></room>
</template>

<script setup lang="ts">
  // TUIRoomコンポーネントをインポートします。インポートパスが正しいことを確認してください
  import Room from './TUIRoom/index.vue';

  function handleKickOff(info) {
    if (info.code === 0) {
      console.log('一般メンバーがキャスターにルームからキックアウト')
```

```
    }
  }
</script>
```

TUIRoom (Android)の統合

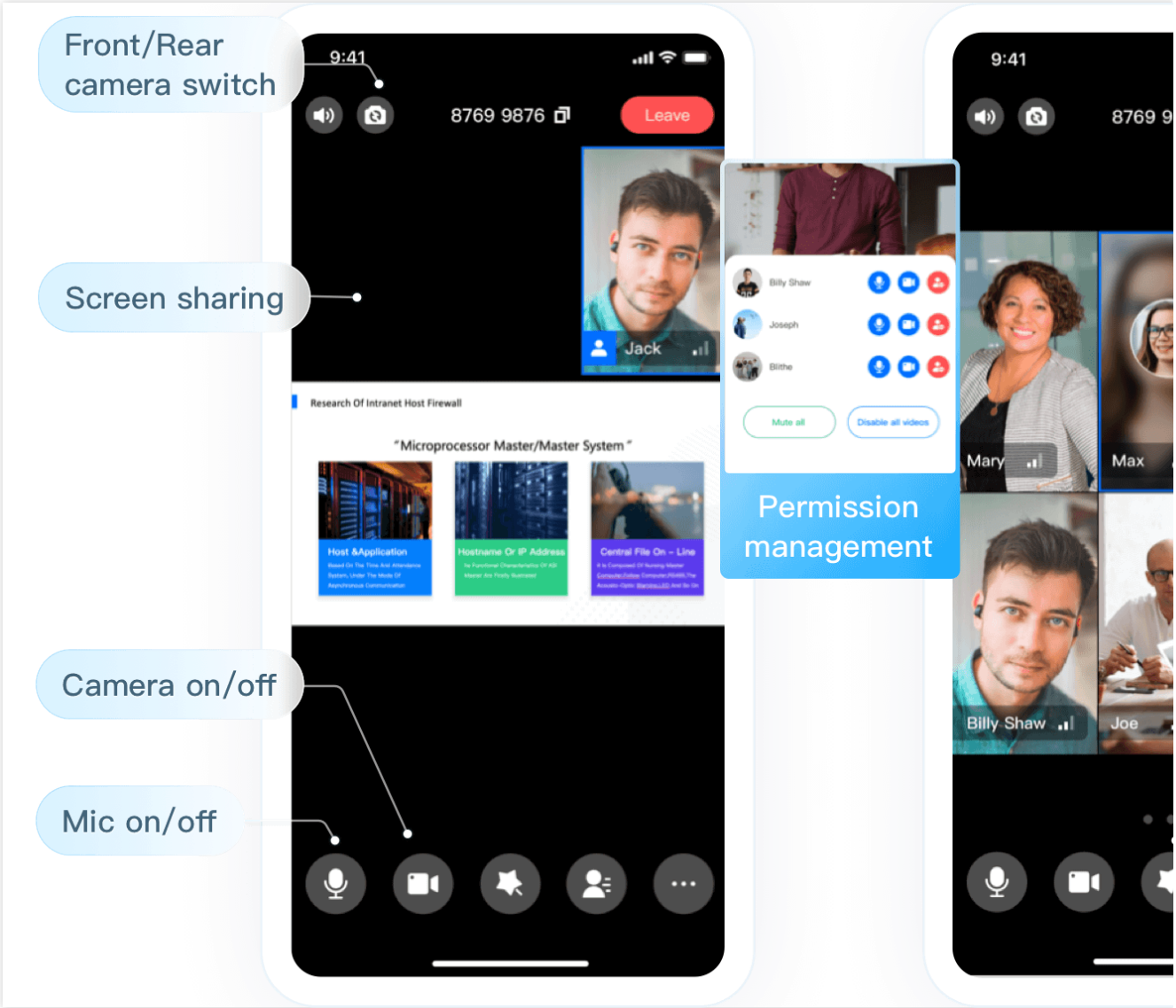
最終更新日：：2024-07-19 15:35:35

コンポーネントの説明

TUIRoomはオープンソースのオーディオビデオUIコンポーネントであり、プロジェクトにTUIRoomコンポーネントを統合することにより、数行のコードを書くだけで、Appに画面共有、美顔、低遅延ビデオ通話などを組み込むことができます。TUIRoomはまた、iOS、Windows、Macなどのプラットフォームでもサポートしています。基本機能は下図のとおりです：

説明：

TUIKitシリーズコンポーネントはTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期的にアクティブにすることができます。IMサービスの課金ルールの詳細については、Instant Messagingの料金説明をご参照ください。TRTCをアクティブにすると、デフォルトでは、100DAUまでサポートするIM SDK体験版もアクティブになります。

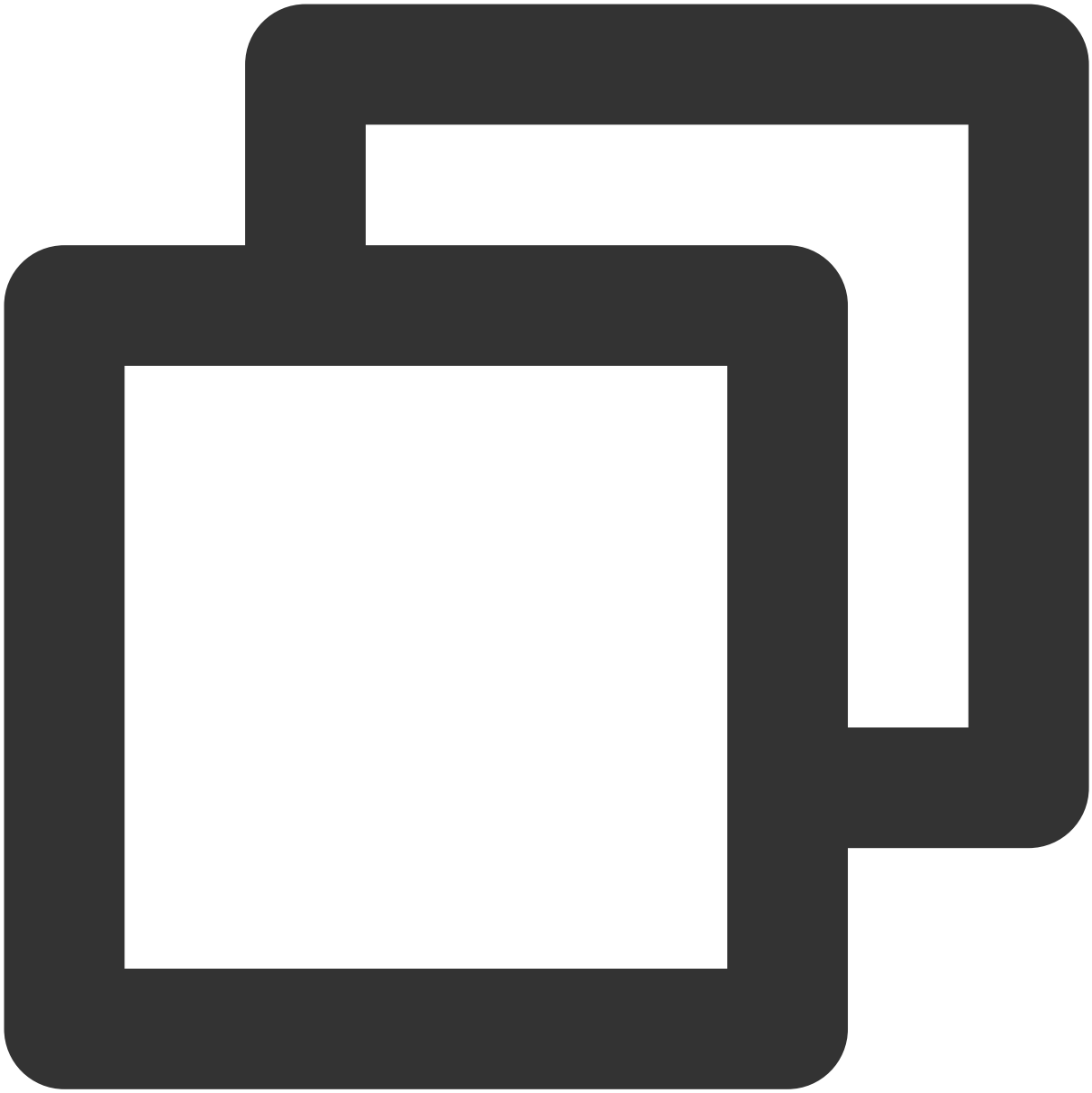


コンポーネントの統合

ステップ1：TUIRoomコンポーネントのダウンロードとインポート

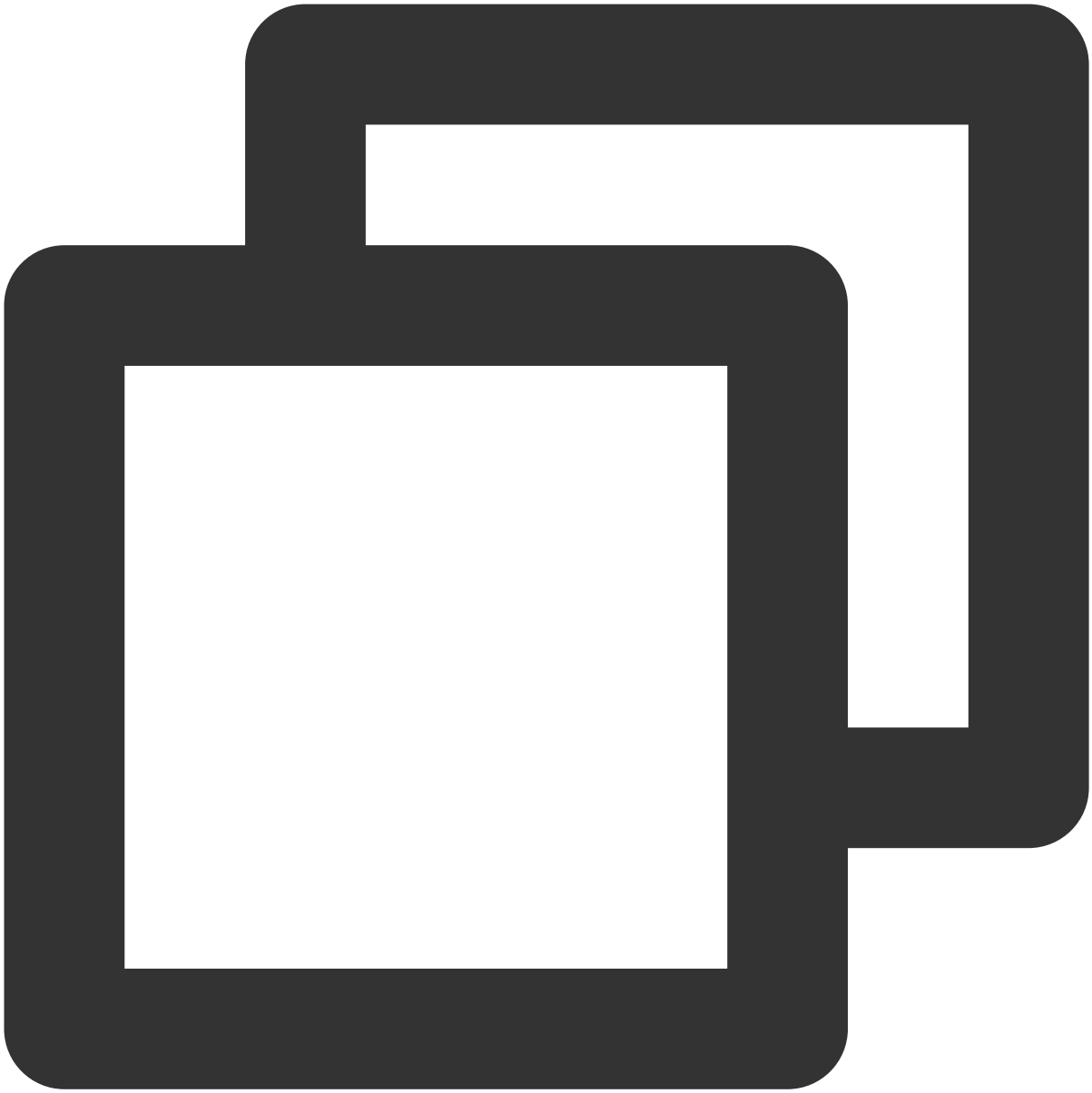
クリックしてGithubに進み、コードのクローン/ダウンロードを選択した後、Android下のtuiroom、debug、tuibeautyディレクトリをプロジェクトにコピーし、次のようにインポート動作を完了します：

1. `setting.gradle` へのインポートを完了します。以下をご参照ください：



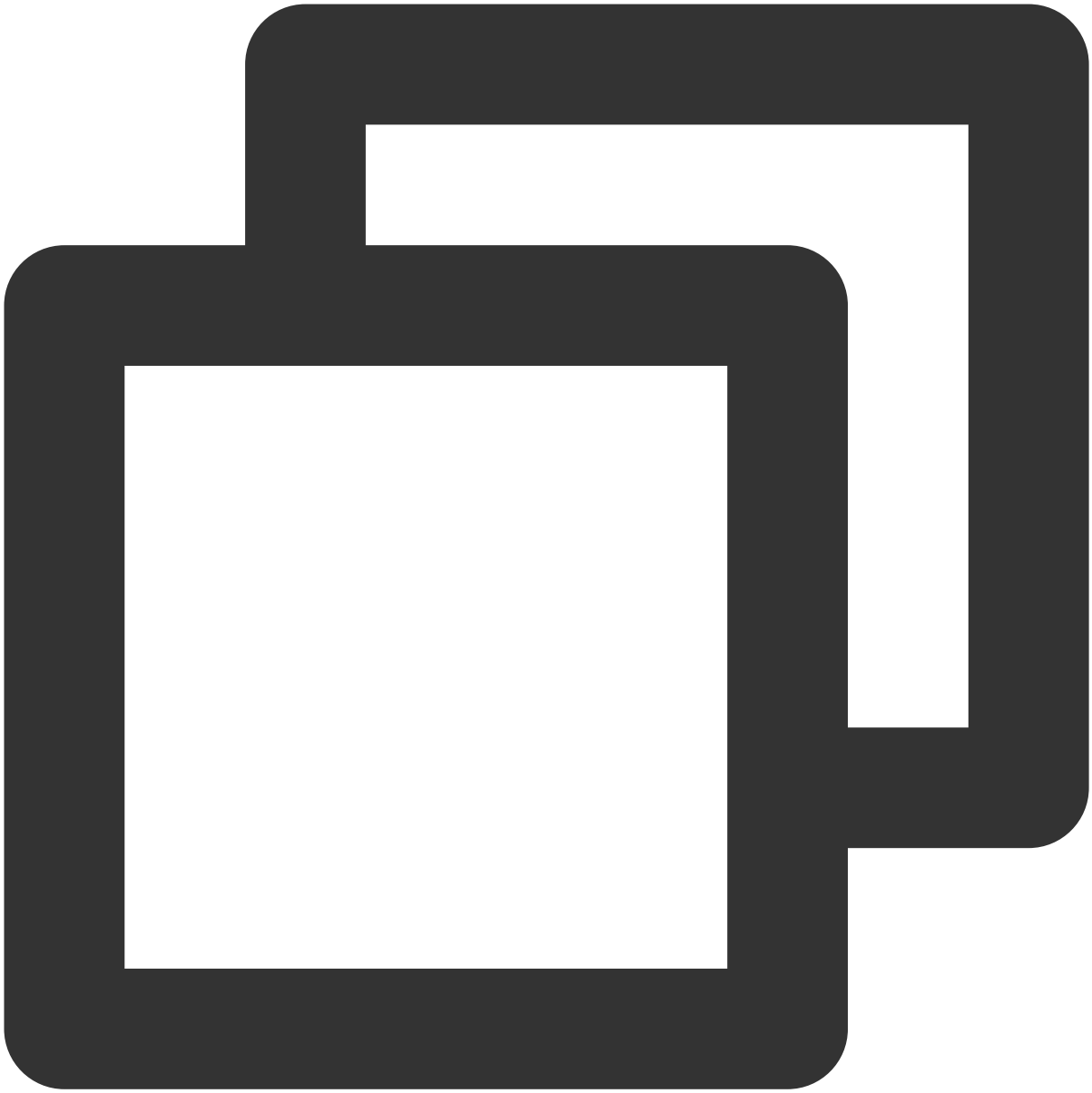
```
include ':tuiroom'
include ':debug'
include ':tuibeauty'
```

2. appの `build.gradle` ファイルにtuiroom、debug、tuibeautyに対する依存関係を追加します：



```
api project(':tuiroom')
api project(':debug')
api project(':tuibeauty')
```

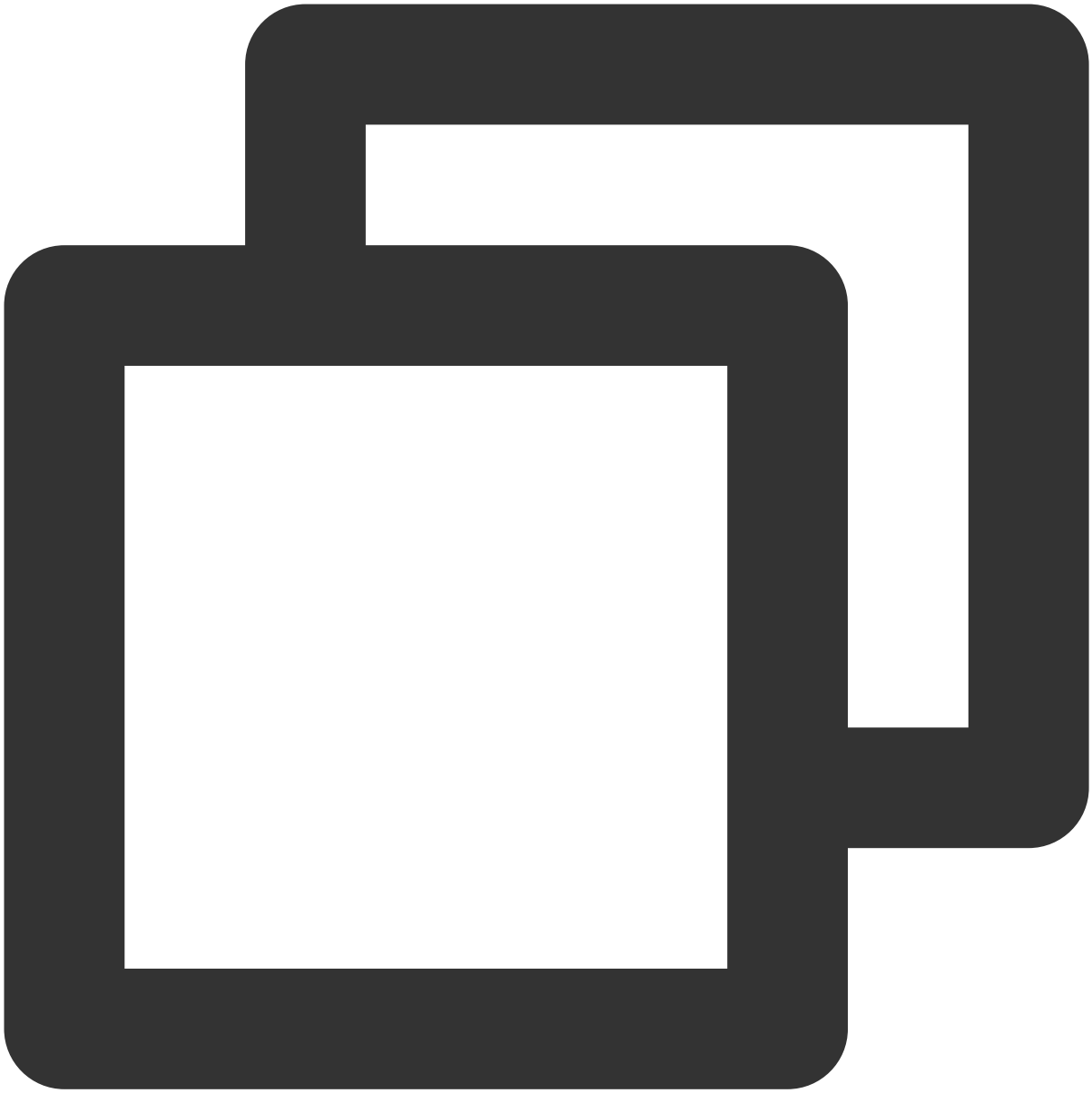
3. ルートディレクトリの `build.gradle` ファイルに `TRTC SDK` および `IM SDK` の依存関係を追加します：



```
ext {
 liteavSdk = "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
 imSdk = "com.tencent.imsdk:imsdk-plus:latest.release"
}
```

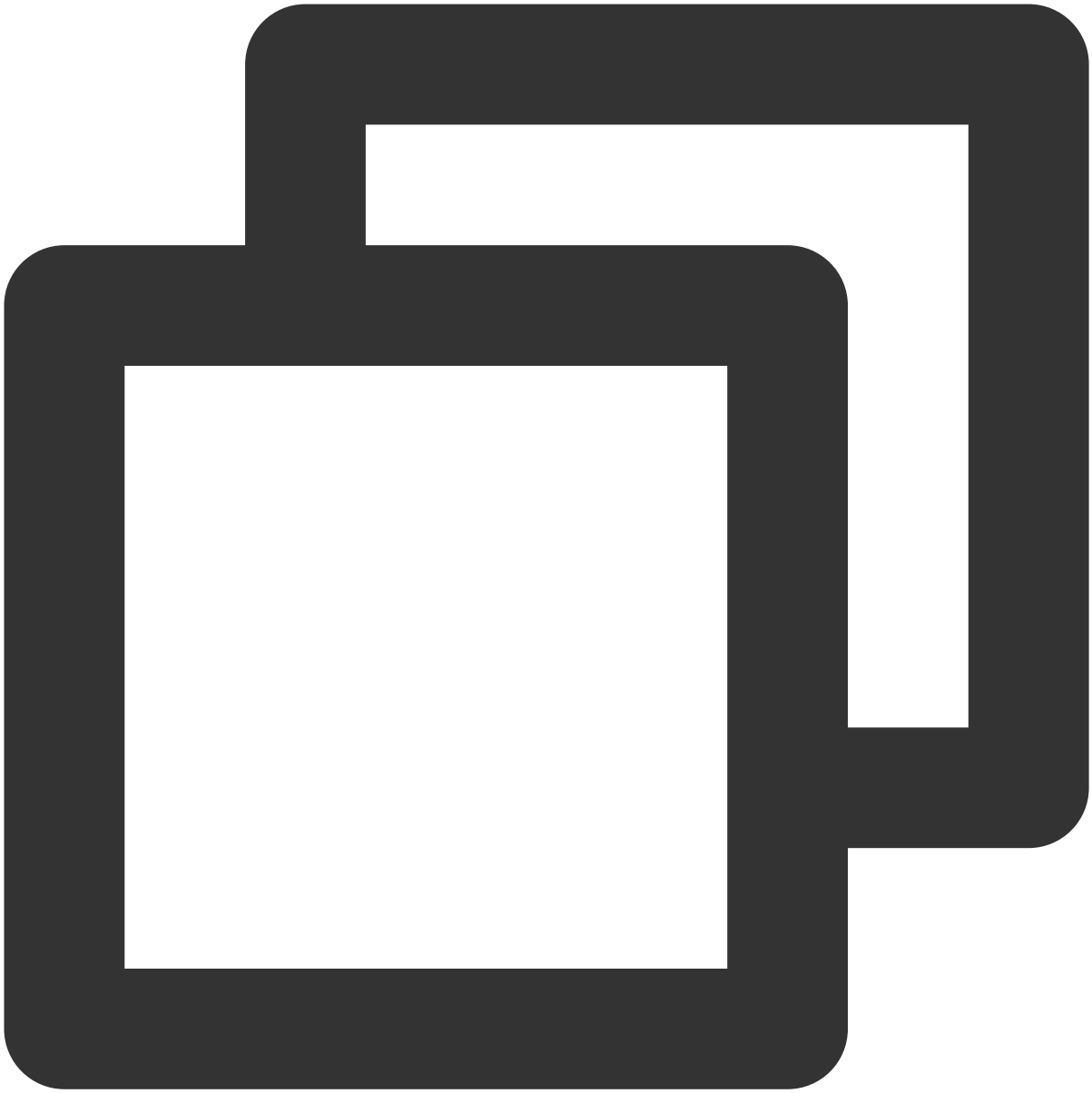
ステップ2：権限の設定および難読化ルール

1. AndroidManifest.xmlにAppの権限を設定します。SDKには次の権限が必要です（6.0以上のAndroidシステムではマイクの権限などを動的に申請してください）：



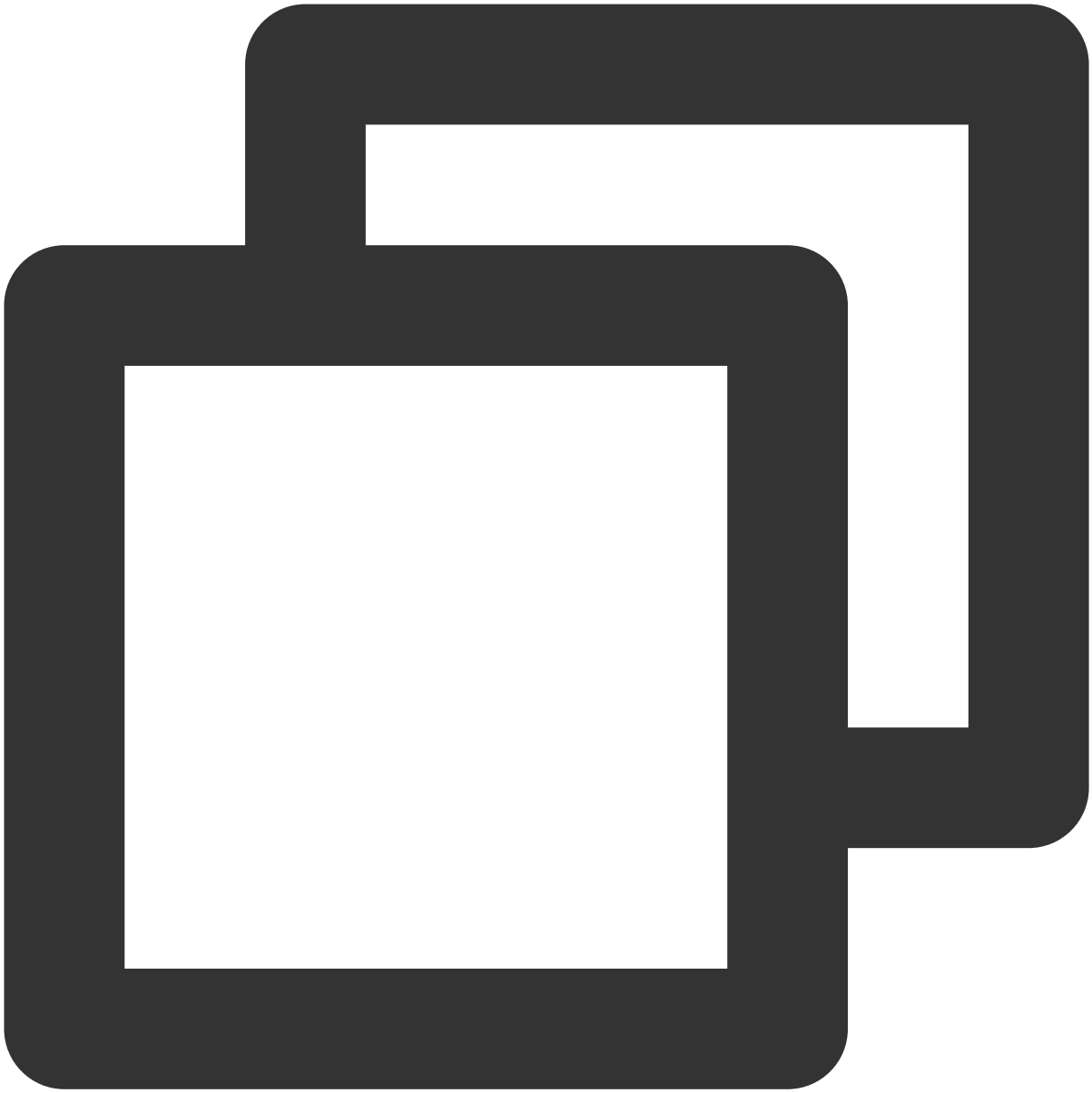
```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera"/>
<uses-feature android:name="android.hardware.camera.autofocus" />
```

2. proguard-rules.proファイルでは、SDK関連クラスを非難読化リストに追加します：



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

ステップ3：TUIコンポーネントリポジトリの作成と初期化

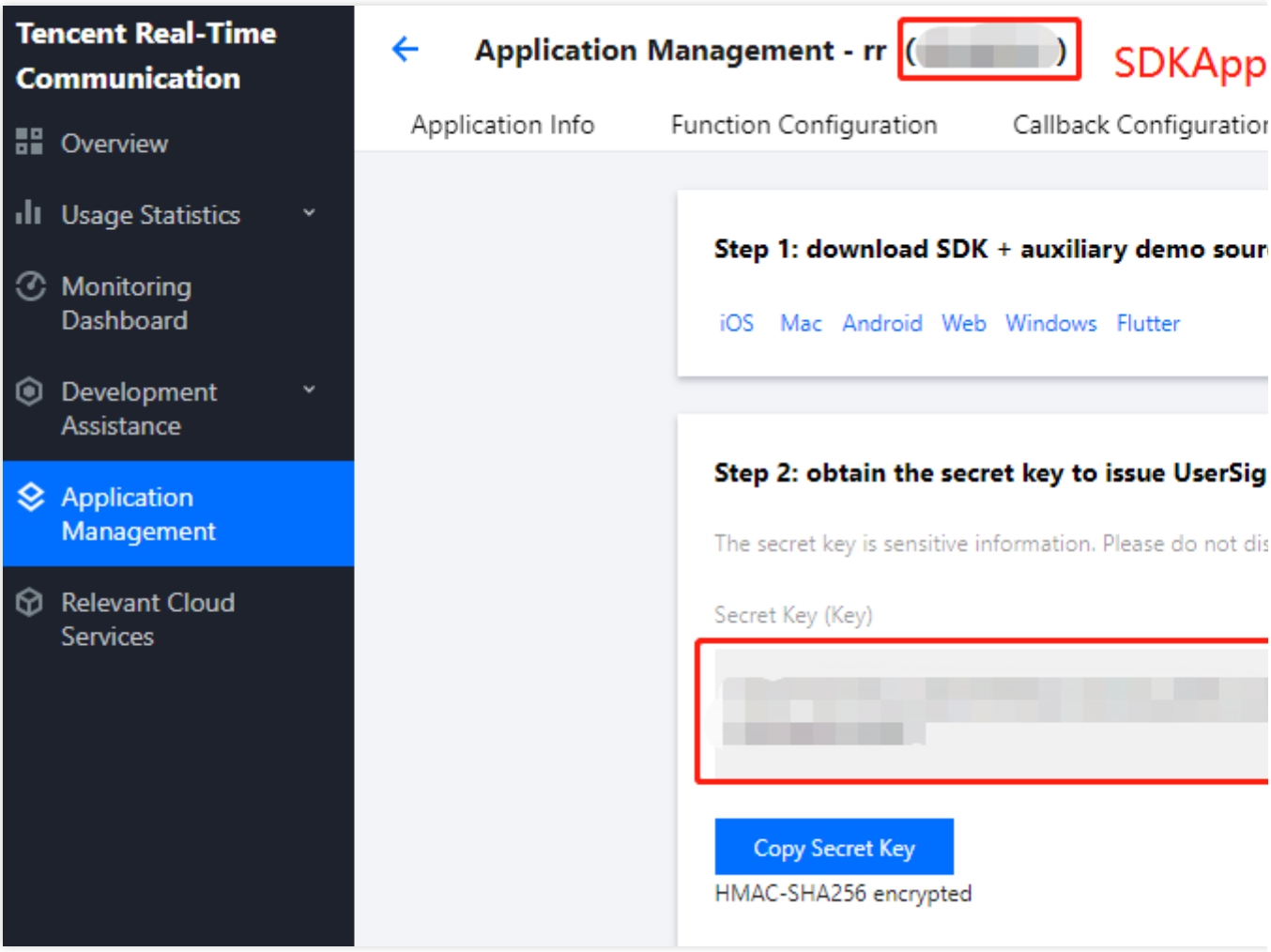


```
// 1.コンポーネントのログイン
TUILogin.addLoginListener(new TUILoginListener() {
    @Override
    public void onKickedOffline() {  //  ログインがキックアウトされたオフライン通知（例：ア
    }

    @Override
    public void onUserSigExpired() { // userSig期限切れ通知
    }
});
```

```
TUILogin.login(context, "あなたのSDKAppId", "あなたのuserId", "あなたのuserSig", null);


// 2.TUIRoomインスタンスの初期化
TUIRoom tuiRoom = TUIRoom.sharedInstance(this);
```

パラメータの説明

SDKAppID：**TRTCアプリケーションID**です。Tencent Cloud TRTCサービスをアクティブ化していない場合は、Tencent Cloud TRTCコンソールに進み、新しいTRTCアプリケーションを作成した後、**アプリケーション情報**をクリックすると、SDKAppID情報が次の図のように表示されます：



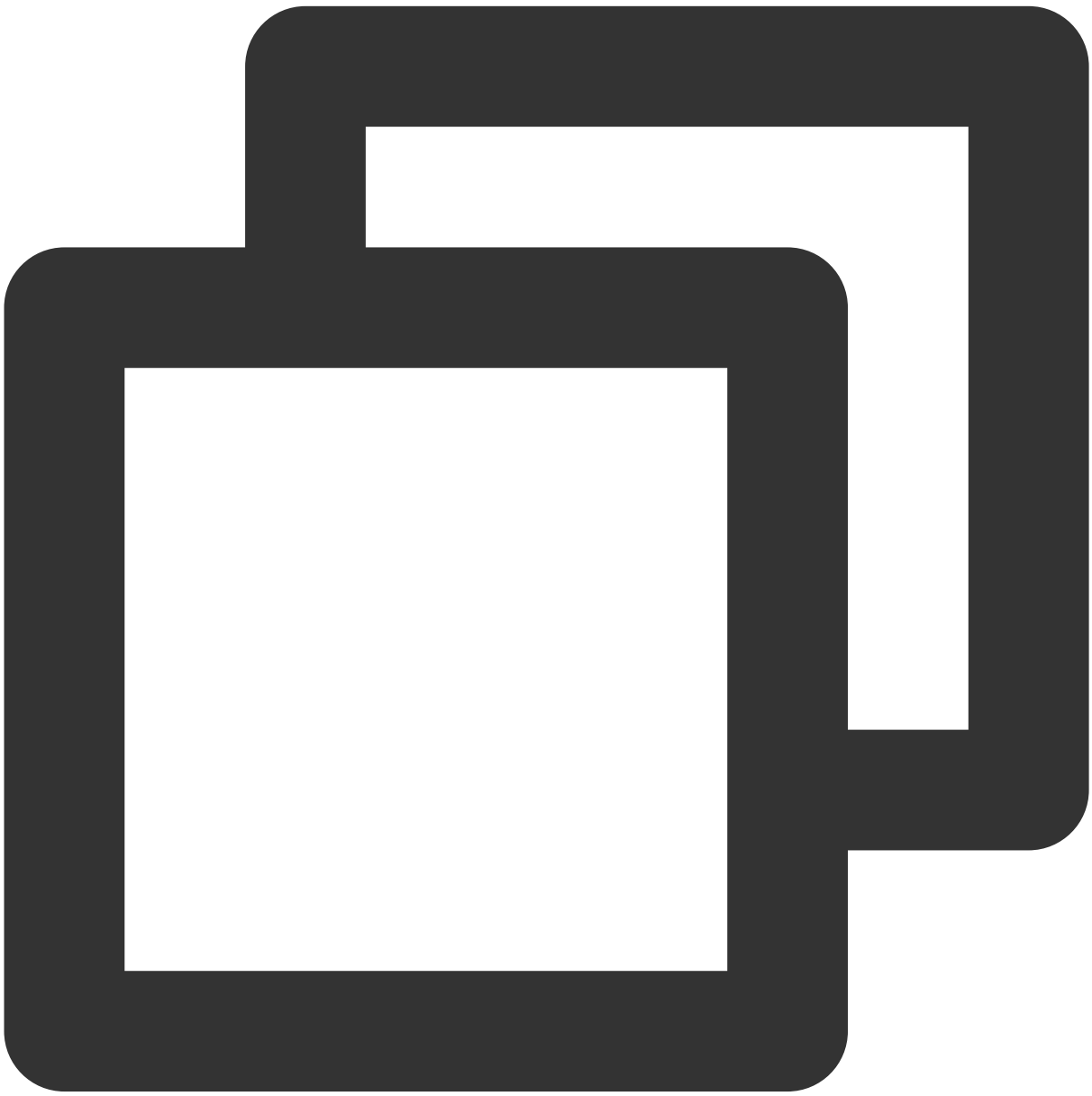
Secretkey：**TRTC アプリケーションキー**であり、SDKAppIdに対応しています。TRTCアプリケーション管理に進むと、SecretKey情報が上の図のように表示されます。

userId：現在のユーザーID。文字列タイプであり、英語のアルファベット（a-zとA-Z）、数字（0-9）、ハイフン（-）とアンダーライン（_）のみ使用できます。業務の実際のアカウントシステムと組み合わせてご自身で設定することをお勧めします。

userSig：SDKAppId、userId，Secretkeyなどの情報に基づく計算によって得られるセキュリティ保護署名です。ここをクリックするとデバッグ用のuserSigがオンラインで直接生成されます。UserSigの計算、使用方法をご参照ください。

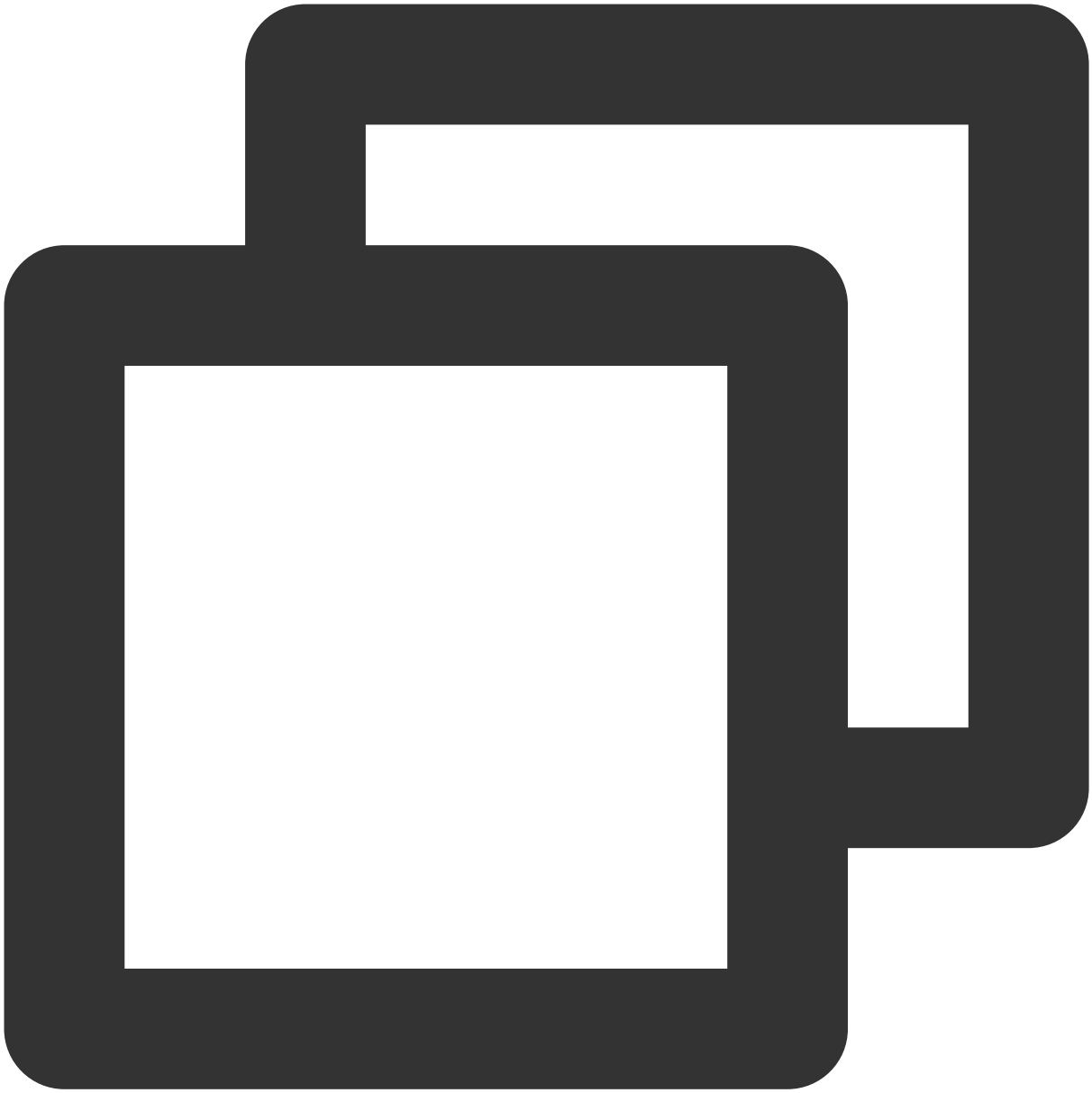
ステップ4：多人数オーディオビデオインタラクションの実装

1. **管理者が多人数オーディオビデオインタラクティブルームを作成できるようにします。**



```
tuiRoom.createRoom(12345, TUIRoomCoreDef.SpeechMode.FREE_SPEECH, true, true);
```

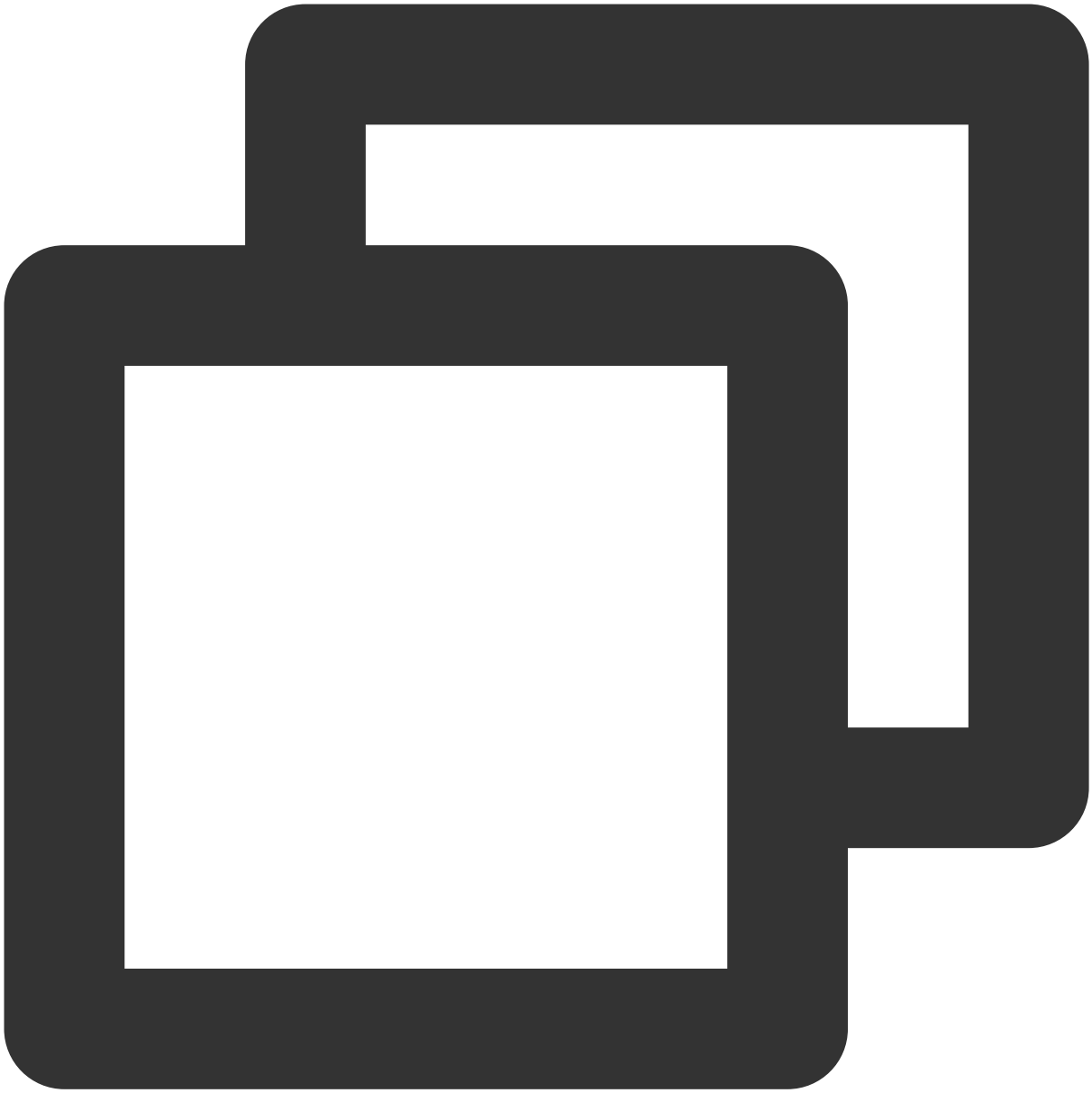
2. **他のメンバーがオーディオビデオルームに参加できるようにします。**



```
tuiRoom.enterRoom(12345, true, true);
```

ステップ5：ルーム管理（オプション）

1. 管理者によるルーム解散 **TUIRoomCore#destroyRoom**。



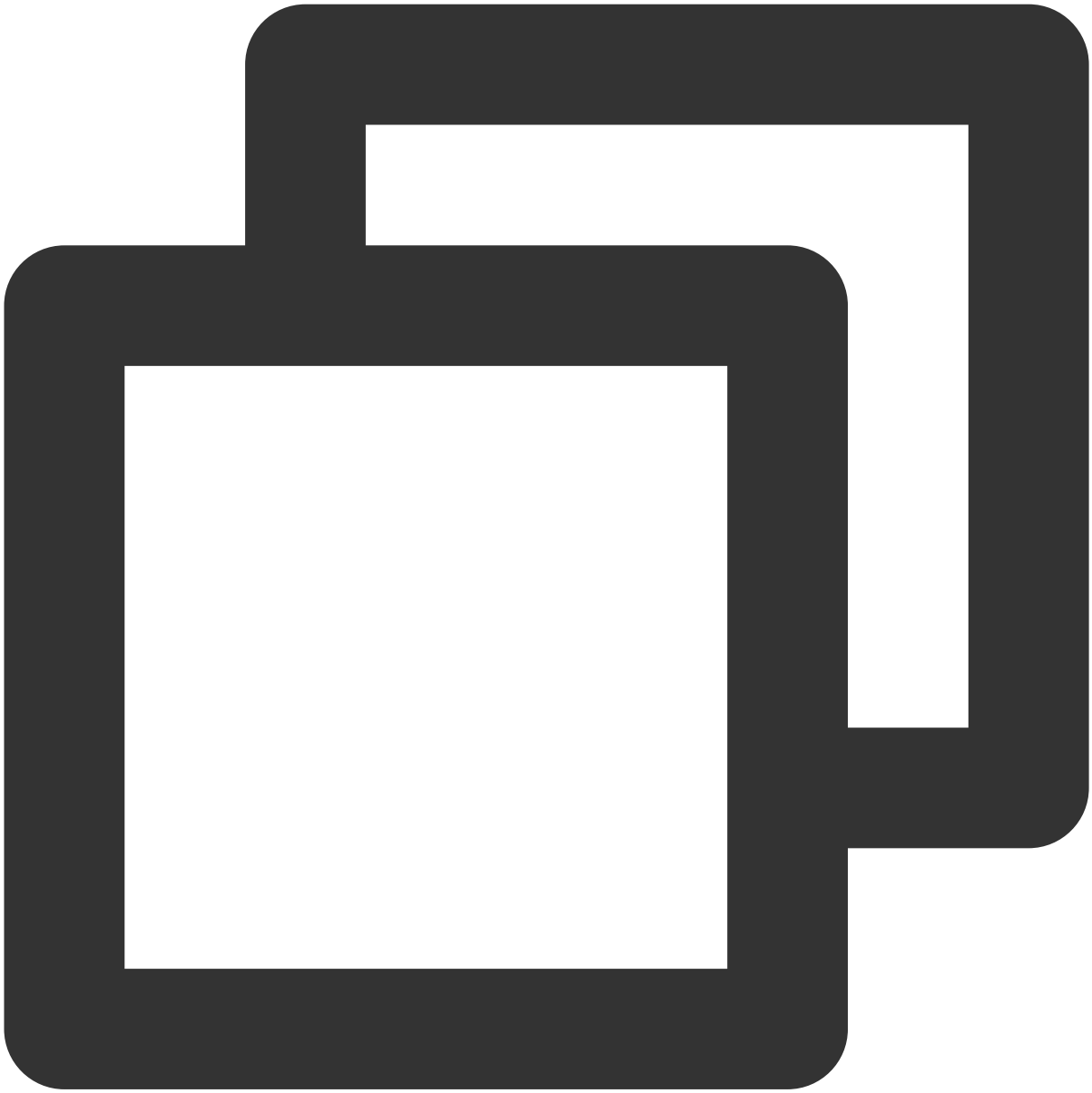
```
// 1.管理者が呼び出してルームを解散
mTUIRoomCore.destroyRoom(new TUIRoomCoreCallback.ActionCallback() {
 @Override
 public void onCallback(int code, String msg) {

 }
});
```

メンバー側は、ルームの解散を通知するonDestroyRoomコールバックメッセージを受信します

```
@Override
public void onDestroyRoom() {
```

```
  //管理者が解散し、ルームから退出
}
```

2. メンバーの退室 **TUIRoomCore#leaveRoom**。

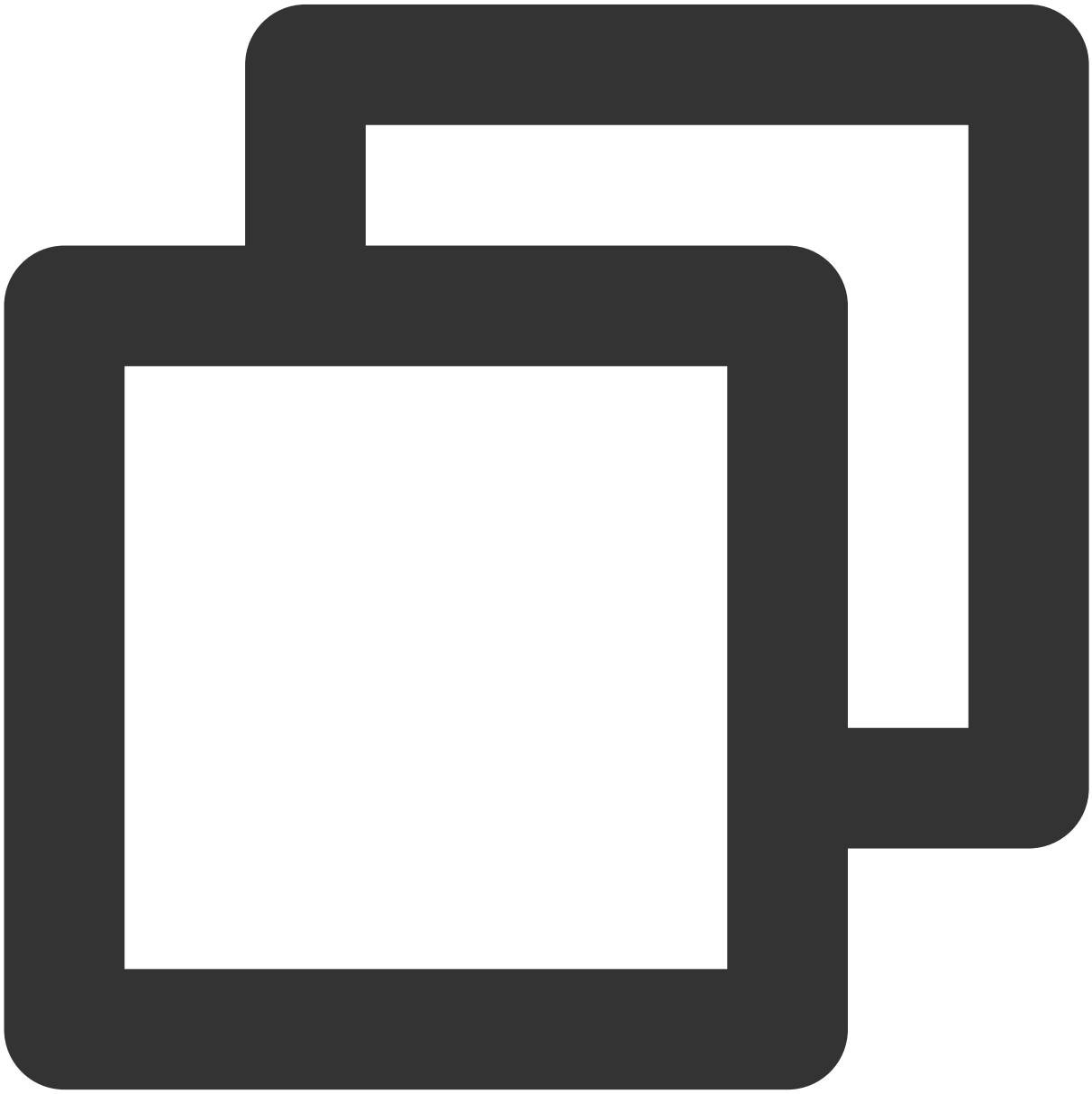


```
// 1.管理者以外による退室の呼び出し
mTUIRoomCore.leaveRoom(new TUIRoomCoreCallback.ActionCallback() {
 @Override
 public void onCallback(int code, String msg) {

 }
});
```

```
//メンバー側は、退室者があったことを通知するonRemoteUserLeaveコールバックメッセージを受信します
@Override
public void onRemoteUserLeave(String userId) {
     Log.d(TAG, "onRemoteUserLeave userId: " + userId);
}
```

ステップ6：画面共有（オプション）

画面共有TUIRoomCore#startScreenCaptureを実装します。



```
// 1.AndroidManifest.xmlのファイルの中にSDKのスクリーンレコーディング機能のactivityと権限を追
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
<application>
    <activity
        android:name="com.tencent.rtmp.video.TXScreenCapture$TXScreenCaptureAssista
        android:theme="@android:style/Theme.Translucent" />
</application>

// 2.自分のインターフェースの中でフローティングウィンドウの権限を申請します
if (Build.VERSION.SDK_INT >= 23) {
    if (!Settings.canDrawOverlays(getApplicationContext())) {
        Intent intent = new Intent(Settings.ACTION_MANAGE_OVERLAY_PERMISSION, Uri.p
        startActivityForResult(intent, 100);
    }else{
        startScreenCapture();
    }
}else{
    startScreenCapture();
}

// 3.システムのコールバックの結果
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == 100) {
        if (Build.VERSION.SDK_INT >= 23) {
            if (Settings.canDrawOverlays(this)) {
                // ユーザーの権限承認が成功
                startScreenCapture();
            }else{
            }
        }
    }
}

// 4.画面共有の起動
private void startScreenCapture() {
        TRTCCloudDef.TRTCVideoEncParam encParams = new TRTCCloudDef.TRTCVideoEncPar
        encParams.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_1280_720;
        encParams.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_POR
        encParams.videoFps = 10;
        encParams.enableAdjustRes = false;
        encParams.videoBitrate = 1200;

        TRTCCloudDef.TRTCScreenShareParams params = new TRTCCloudDef.TRTCScreenShar
        mTUIRoom.stopCameraPreview();
        mTUIRoom.startScreenCapture(encParams, params);
}
```

よくあるご質問

ご要望やフィードバックなどがございましたら、colleenyu@tencent.comまでご連絡ください。

TUIRoom (iOS)の統合

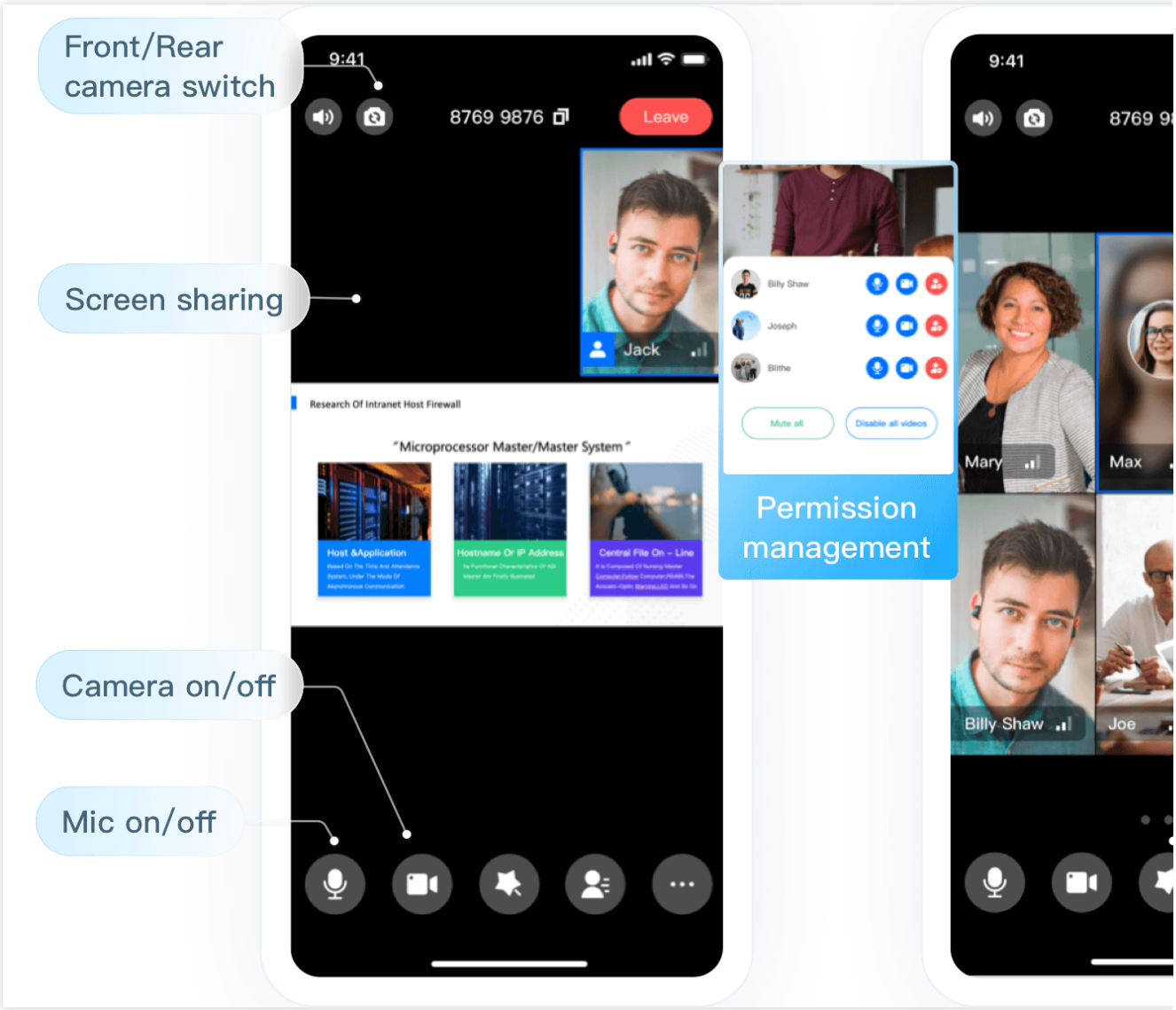
最終更新日：：2024-07-19 15:35:35

コンポーネントの説明

TUIRoomはオープンソースのオーディオビデオUIコンポーネントであり、プロジェクトにTUIRoomコンポーネントを統合することにより、数行のコードを書くだけで、Appに画面共有、美顔、低遅延ビデオ通話などを組み込むことができます。TUIRoomはまた、Android、Windows、Macなどのプラットフォームでもサポートしています。基本機能は下図のとおりです：

説明：

TUIKitシリーズコンポーネントはTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期的にアクティブにすることができます。IMサービスの課金ルールの詳細については、Instant Messagingの料金説明をご参照ください。TRTCをアクティブにすると、デフォルトでは、100DAUまでサポートするIM SDK体験版もアクティブになります。

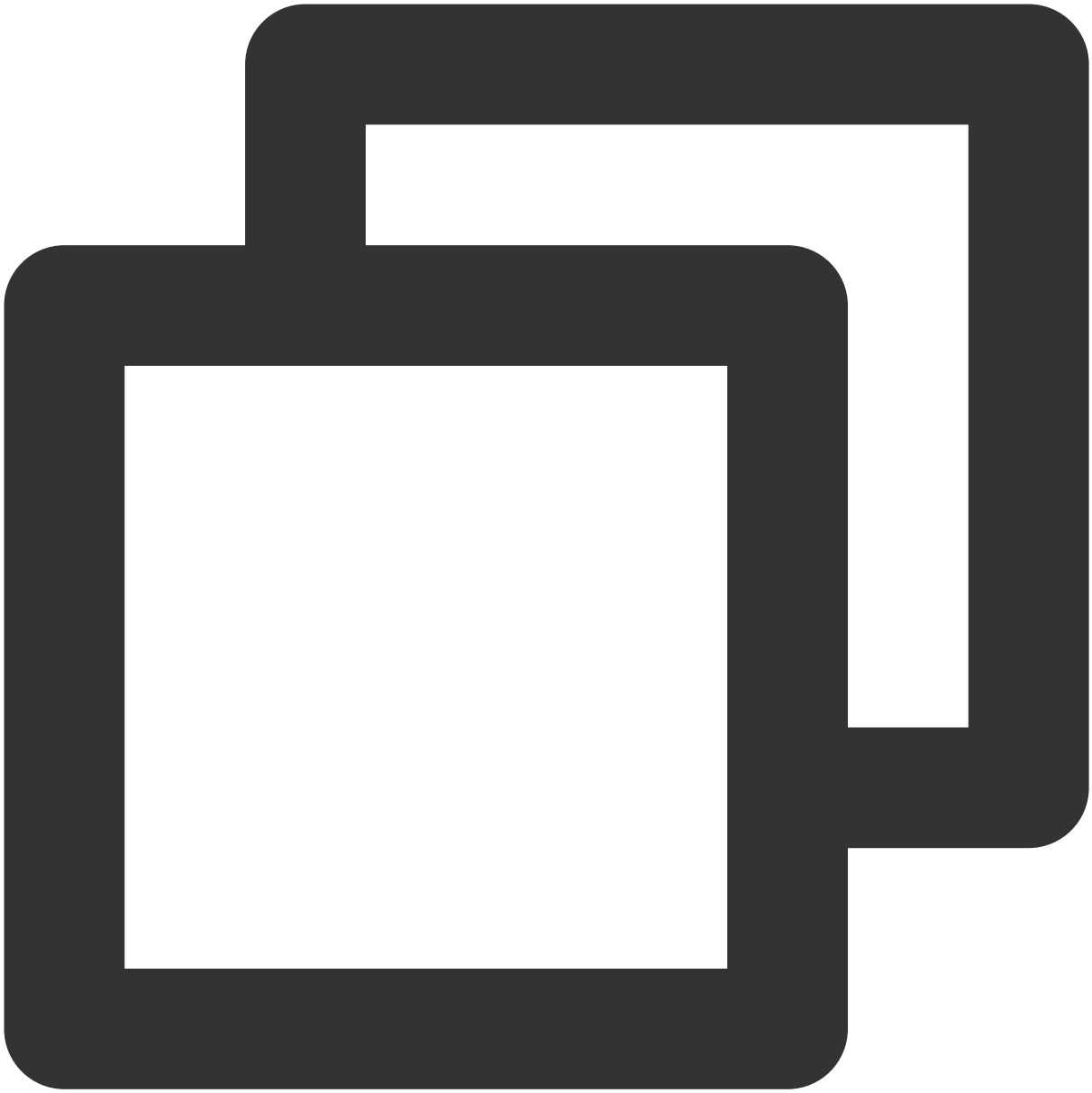


コンポーネントの統合

ステップ1：TUIRoomコンポーネントのインポート

cocoapodsによってコンポーネントをインポートします。具体的な手順については、以下のとおりです：

1. プロジェクトの `Podfile` ファイルと同じ階層のディレクトリ下に `TUIRoom` フォルダを作成します。
2. クリックして**Github/TUIRoom**に進み、コードのクローン/ダウンロードを選択した後、**TUIRoom/iOS/**ディレクトリ下の `Source` 、 `Resources` 、 `TUIBeauty` 、 `TXAppBasic`フォルダ、 TUIRoom.podspec `ファイル`を、 ステップ1`で作成したTUIRoomフォルダ下にコピーします。
3. Podfileファイル内に以下の依存関係を追加します。その後、 `pod install` コマンドを実行すると、インポートが完了します。



```
# :path => "TUIRoom.podspecを指定する相対パス"
pod 'TUIRoom', :path => "./TUIRoom/TUIRoom.podspec", :subspecs => ["TRTC"]
# :path => "TXAppBasic.podspecを指定する相対パス"
pod 'TXAppBasic', :path => "./TUIRoom/TXAppBasic/"
# :path => "TUIBeauty.podspecを指定する相対パス"
pod 'TUIBeauty', :path => "./TUIRoom/TUIBeauty/"
```

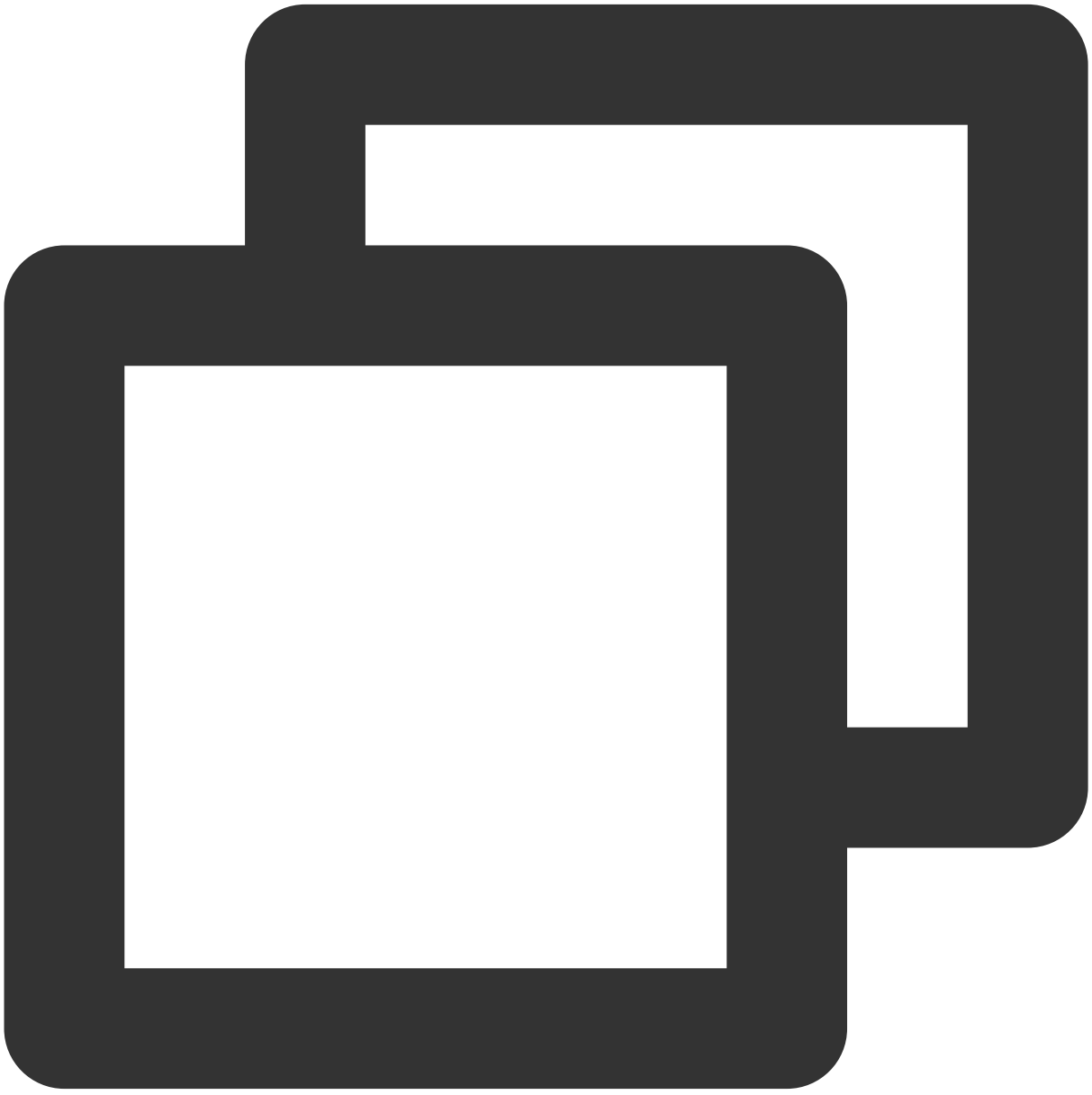
ご注意：

`Source` 、 `Resources` フォルダと `TUIRoom.podspec` ファイルは同一のディレクトリ下にある必要があります。

TXAppBasic.podspecはTXAppBasicフォルダ下にあります。
TUIBeauty.podspecはTCBeautyKitフォルダ下にあります。

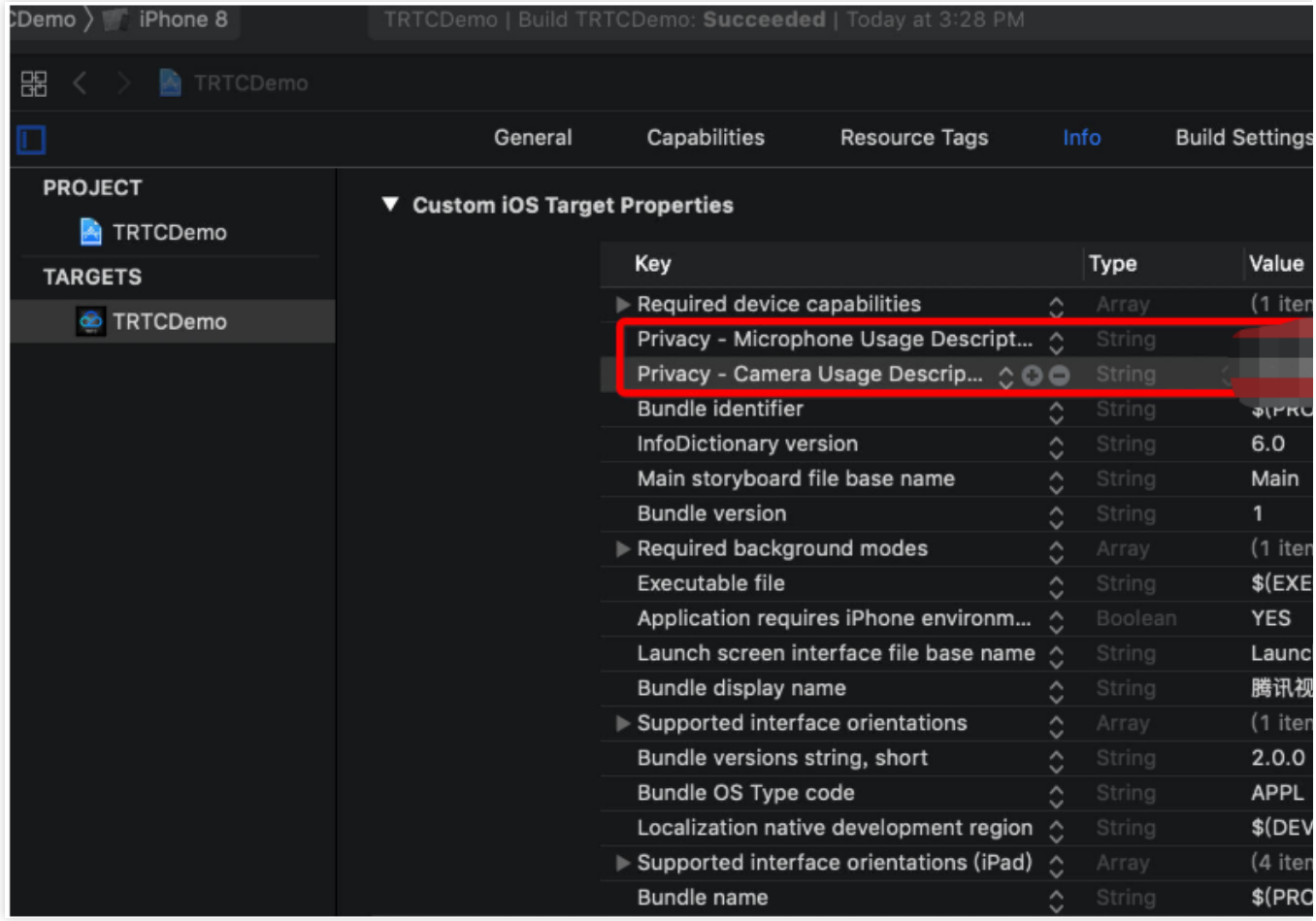
ステップ2：権限の設定

オーディオビデオ機能を使用するには、マイクおよびカメラの使用権限を付与する必要があります。AppのInfo.plistで以下の2項を追加します。これらはシステムが権限付与ダイアログボックスをポップアップする際に表示される、マイクとカメラのメッセージにそれぞれ対応します。



```
<key>NSCameraUsageDescription</key>
<string>RoomAppはカメラへのアクセス権限が必要です。有効にしないとレコーディングしたビデオの画面は
```

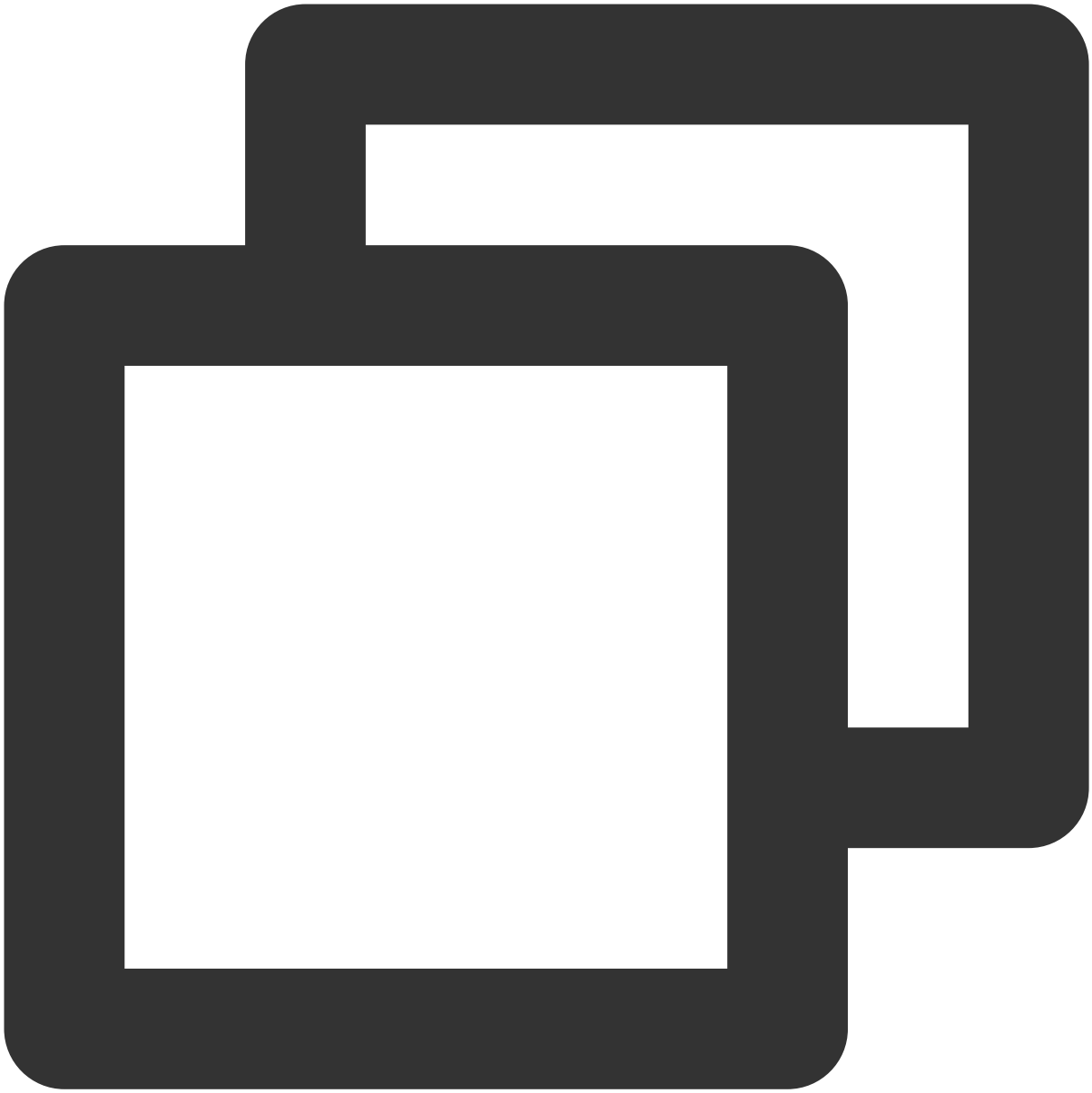
```
<key>NSMicrophoneUsageDescription</key>
<string>RoomAppはマイクへのアクセス権限が必要です。有効にないと、レコーディングしたビデオの音声は
```



ステップ3：TUIコンポーネントリポジトリの作成と初期化

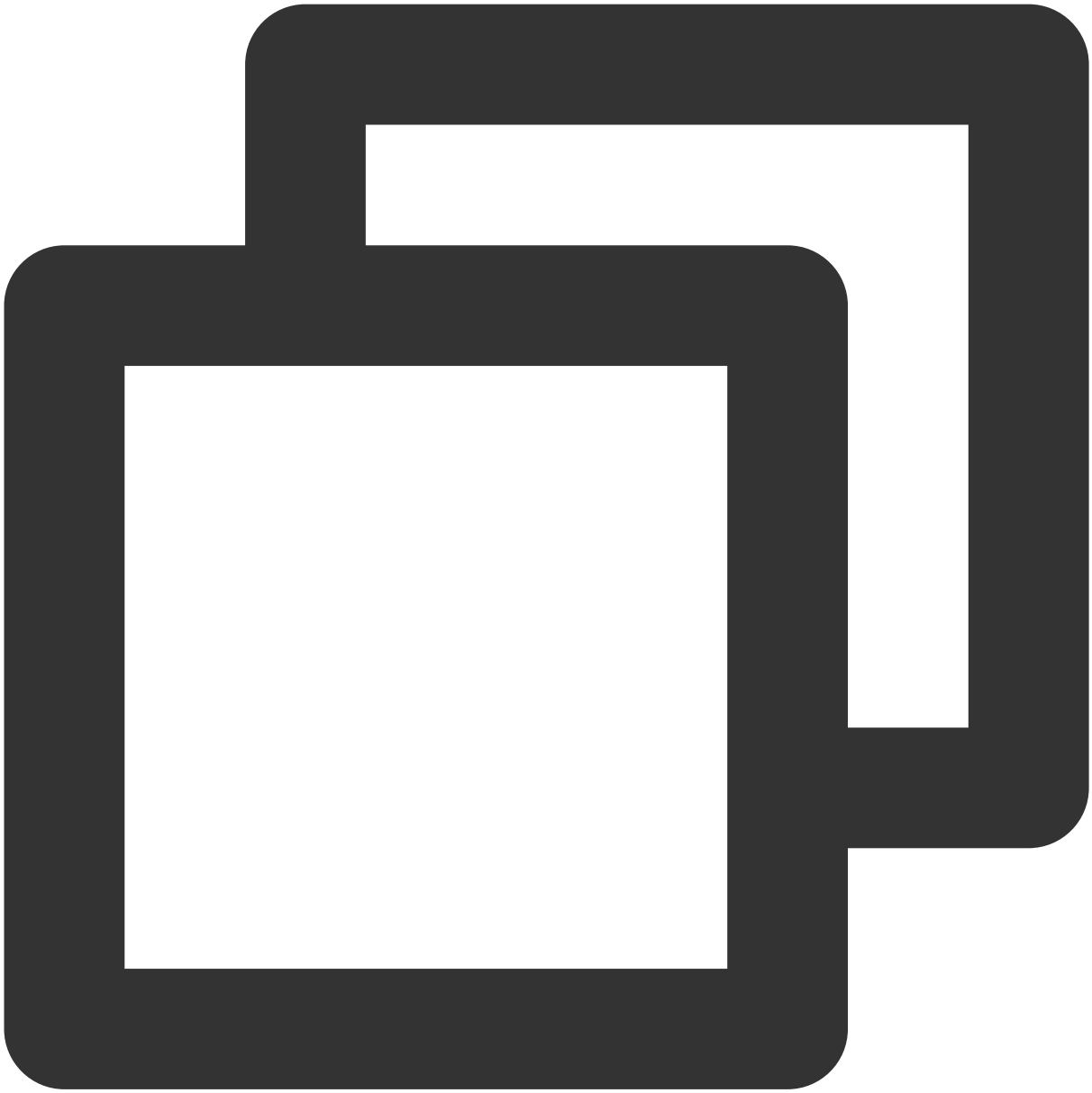
Objective-C

Swift



```
@import TUIRoom;
@import TUICore;

// 1.コンポーネントのログイン
[TUILogin login:@"あなたのSDKAppID" userID:@あなたのUserID" userSig:@"あなたのUserSig"

} fail:^(int code, NSString *msg) {

}];
// 2.TUIRoomインスタンスの初期化
TUIRoom *tuiRoom = [TUIRoom sharedInstance];
```

```
```

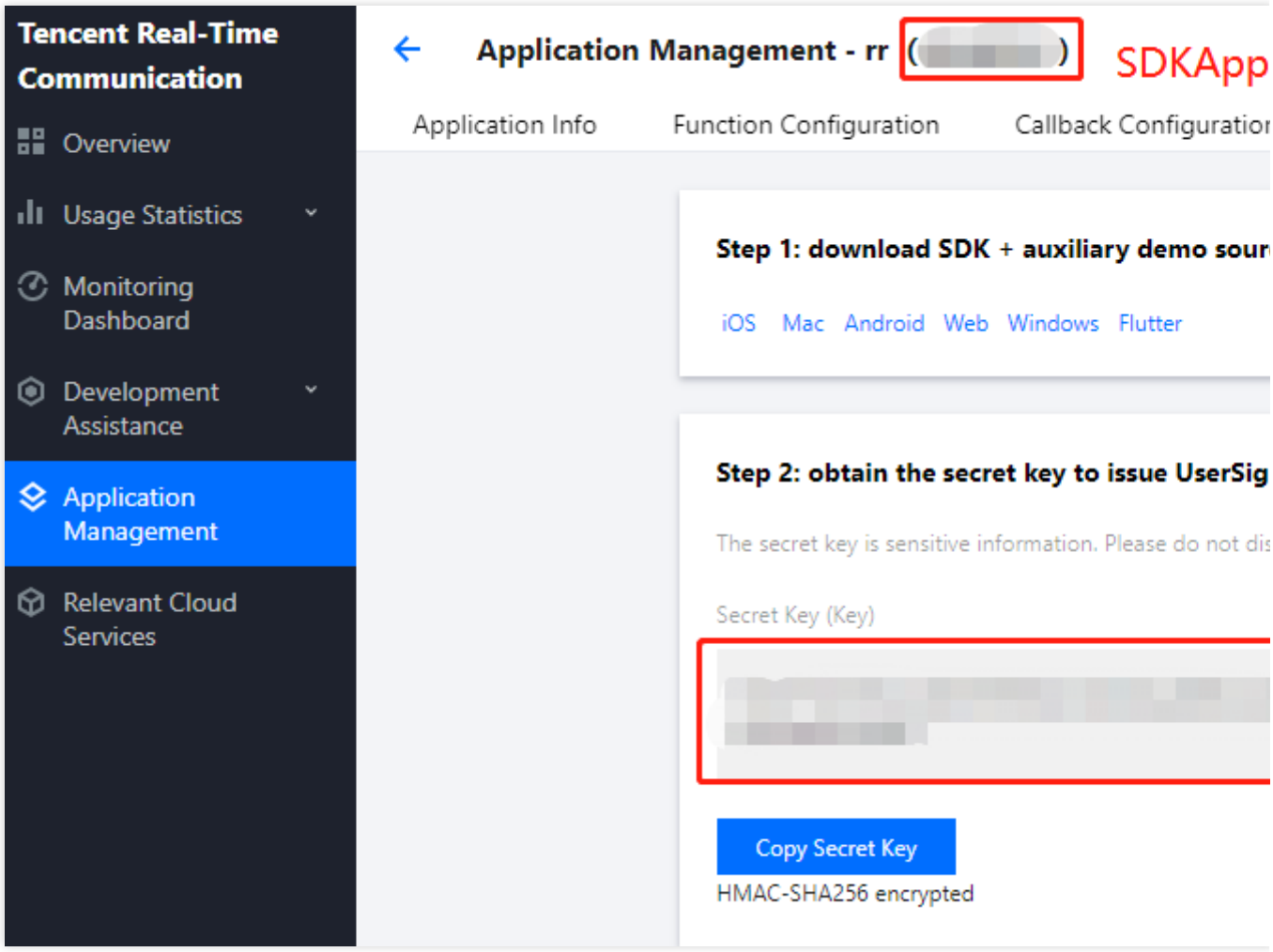


```
import TUIRoom
import TUICore

// 1.コンポーネントのログイン
TUILogin.login("あなたのSDKAppID", userID: "あなたのUserID", userSig: "あなたのUserSig"

} fail: { code, msg in
```

```
}

// 2.TUIRoomインスタンスの初期化
let tuiRoom = TUIRoom.sharedInstance
```

パラメータの説明：

SDKAppID：**TRTCアプリケーションID**です。Tencent Cloud TRTCサービスをアクティブ化していない場合は、Tencent Cloud TRTCコンソールに進み、新しいTRTCアプリケーションを作成した後、**アプリケーション情報**をクリックすると、SDKAppID情報が次の図のように表示されます：



SecretKey：**TRTC アプリケーションキー**であり、SDKAppIDに対応しています。TRTCアプリケーション管理に進むと、SecretKey情報が上図のように表示されます。

UserID：現在のユーザーのIDです。文字列タイプであり、長さは32バイト以内とし、特殊文字の使用はサポートしていません。英語または数字の使用をお勧めします。業務の実際のアカウントシステムと組み合わせてご自身で設定することができます。

UserSig：SDKAppId、UserID，SecretKeyなどの情報に基づく計算によって得られるセキュリティ保護署名です。ここをクリックするとデバッグ用のUserSigがオンラインで直接生成されます。また当社のTUIRoomデモプロジェクトを参照してご自身で計算することもできます。その他の情報については、UserSigの計算、使用方法をご参照ください。

ステップ4：多人数オーディオビデオインタラクションの実装

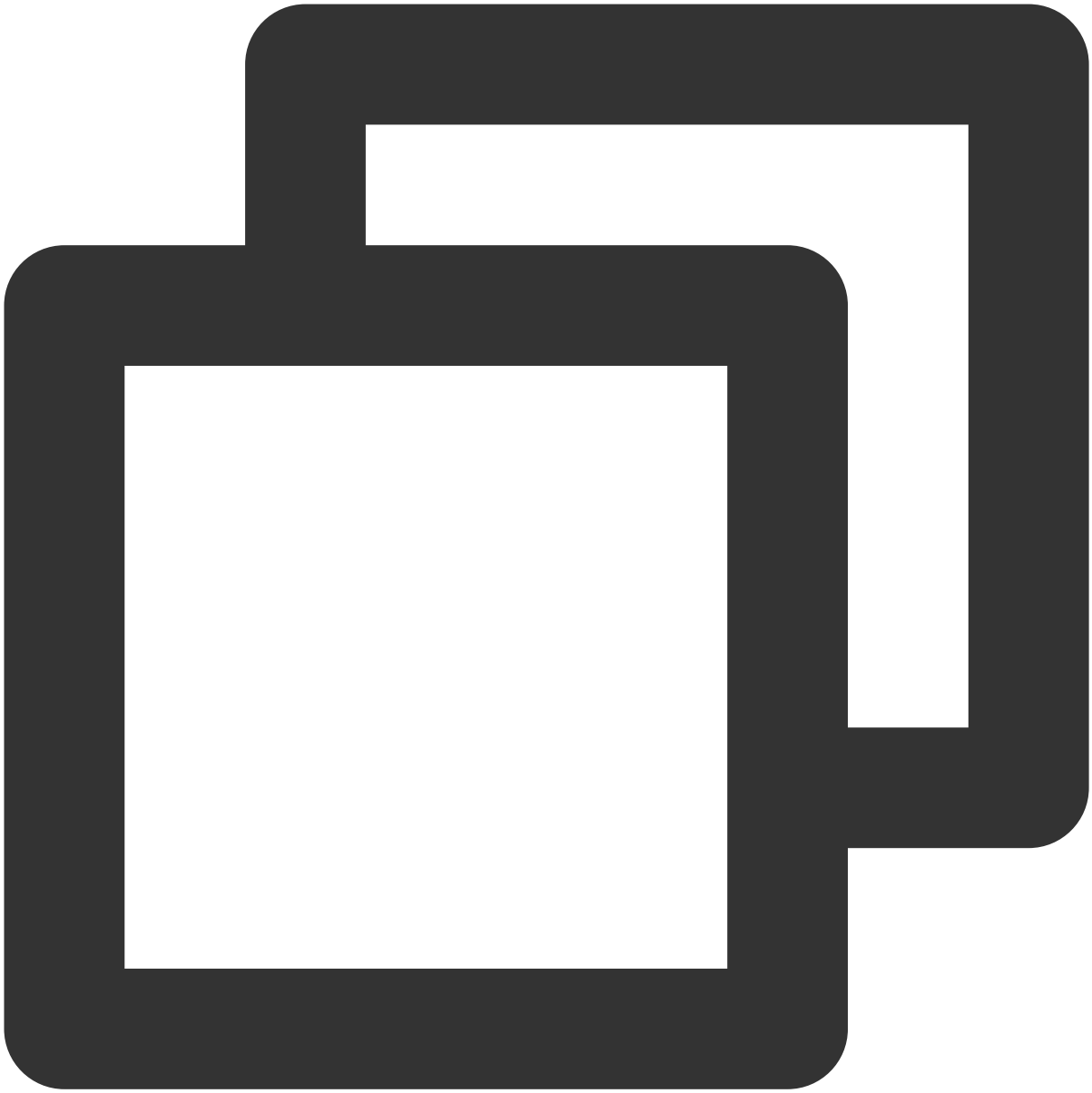
1. **管理者が多人数オーディオビデオインタラクティブルームを作成できるようにします。**

Objective-C

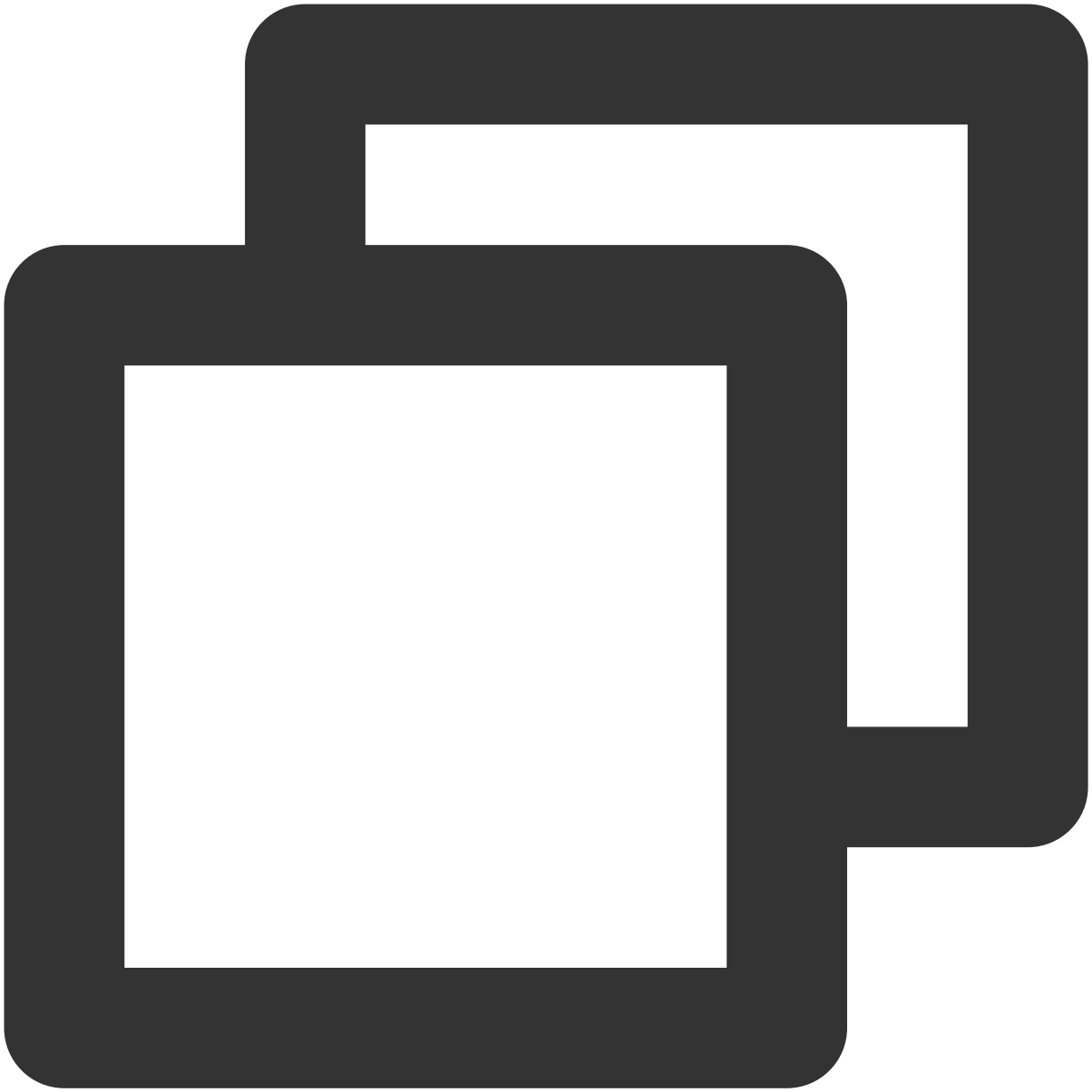
Swift

Objective-C

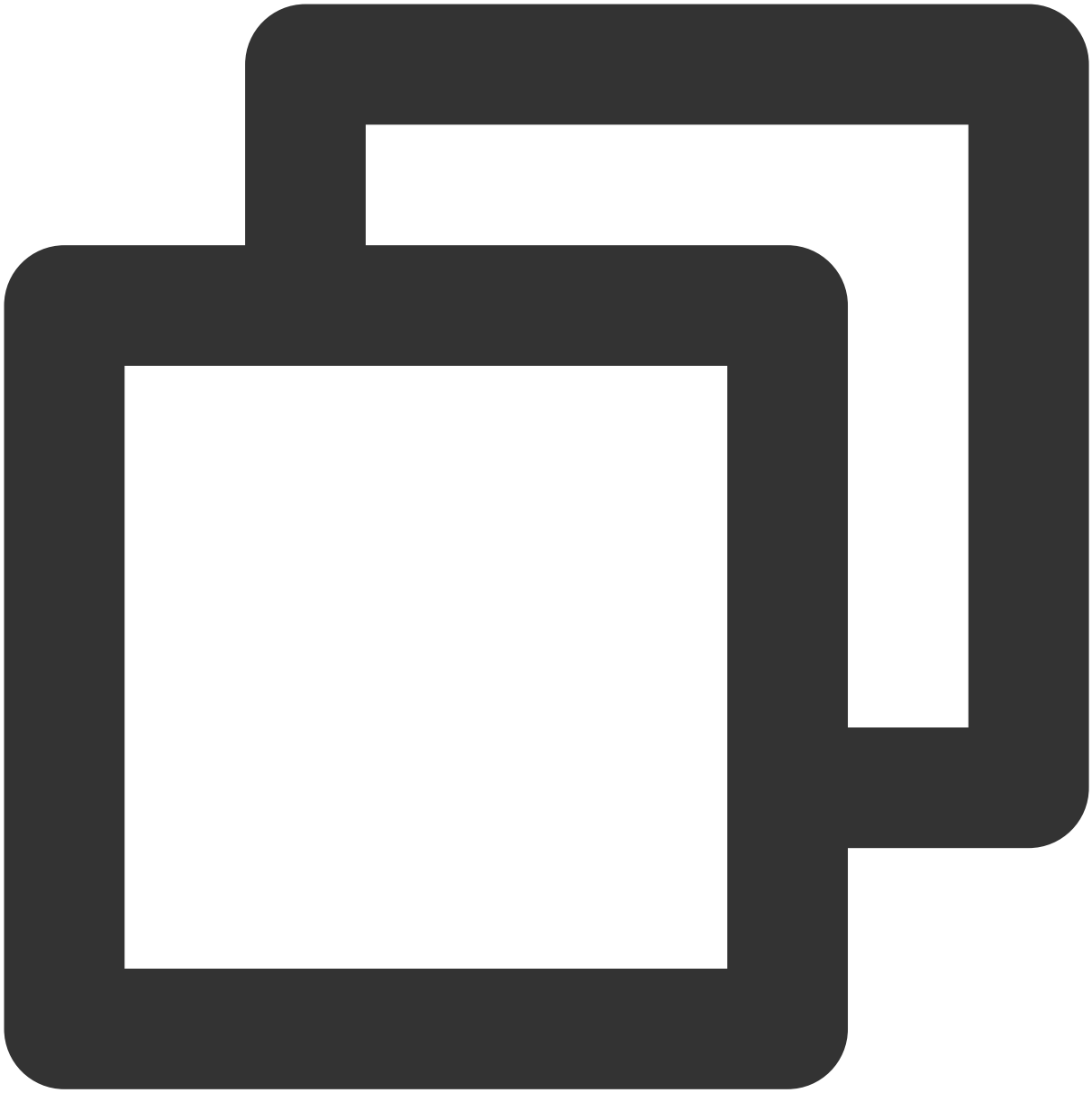
Swift



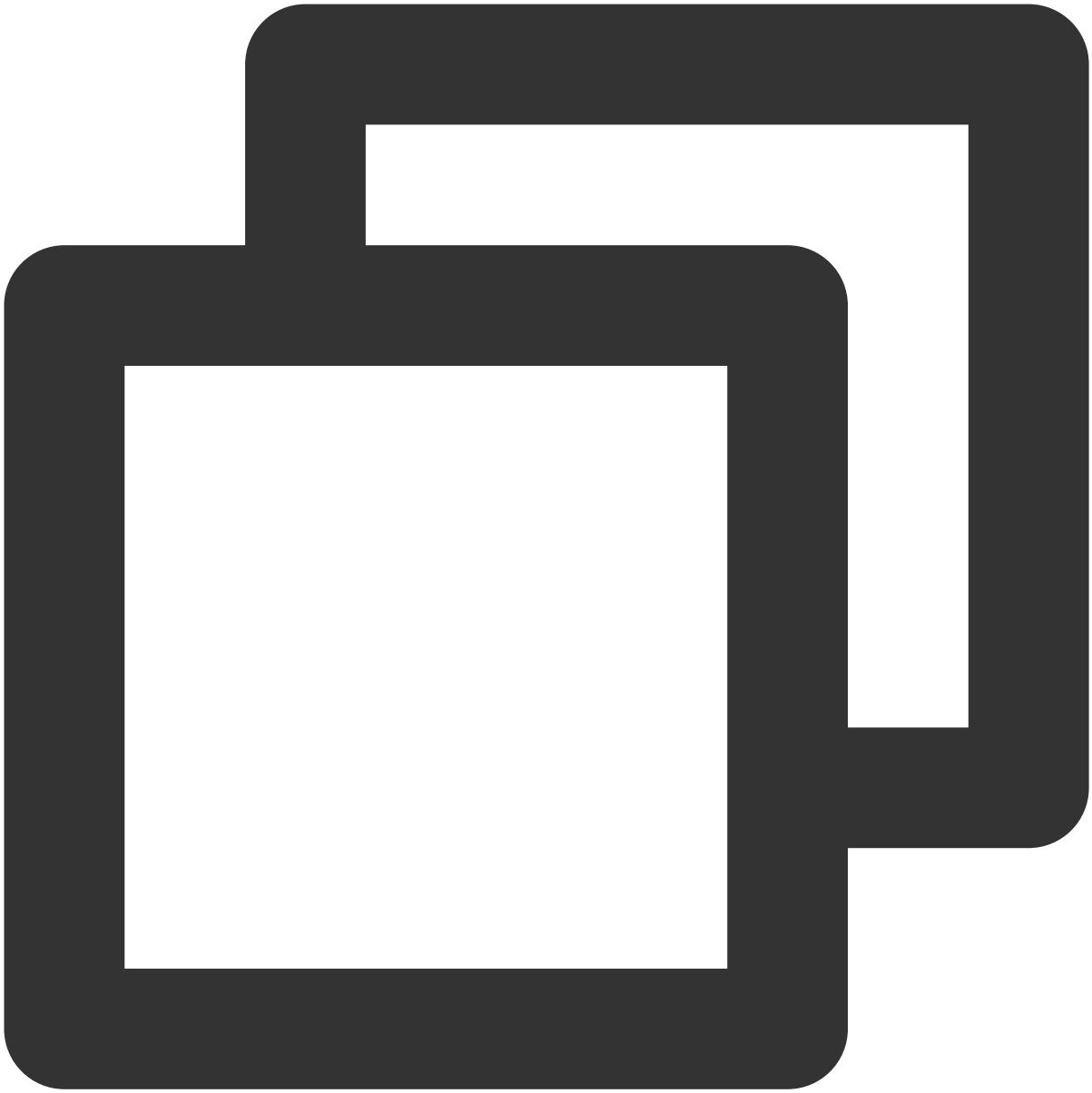
```
@import TUIRoom;

[tuiRoom createRoomWithRoomId:12345 speechMode:TUIRoomFreeSpeech isOpenCamera:YES i
```



```
import TUIRoom

tuiRoom.createRoom(roomId: 12345, speechMode: .freeSpeech, isOpenCamera: true, isOp
```



```
@import TUIRoom;

[tuiRoom enterRoomWithRoomId:12345 isOpenCamera:YES isOpenMicrophone:YES]
```



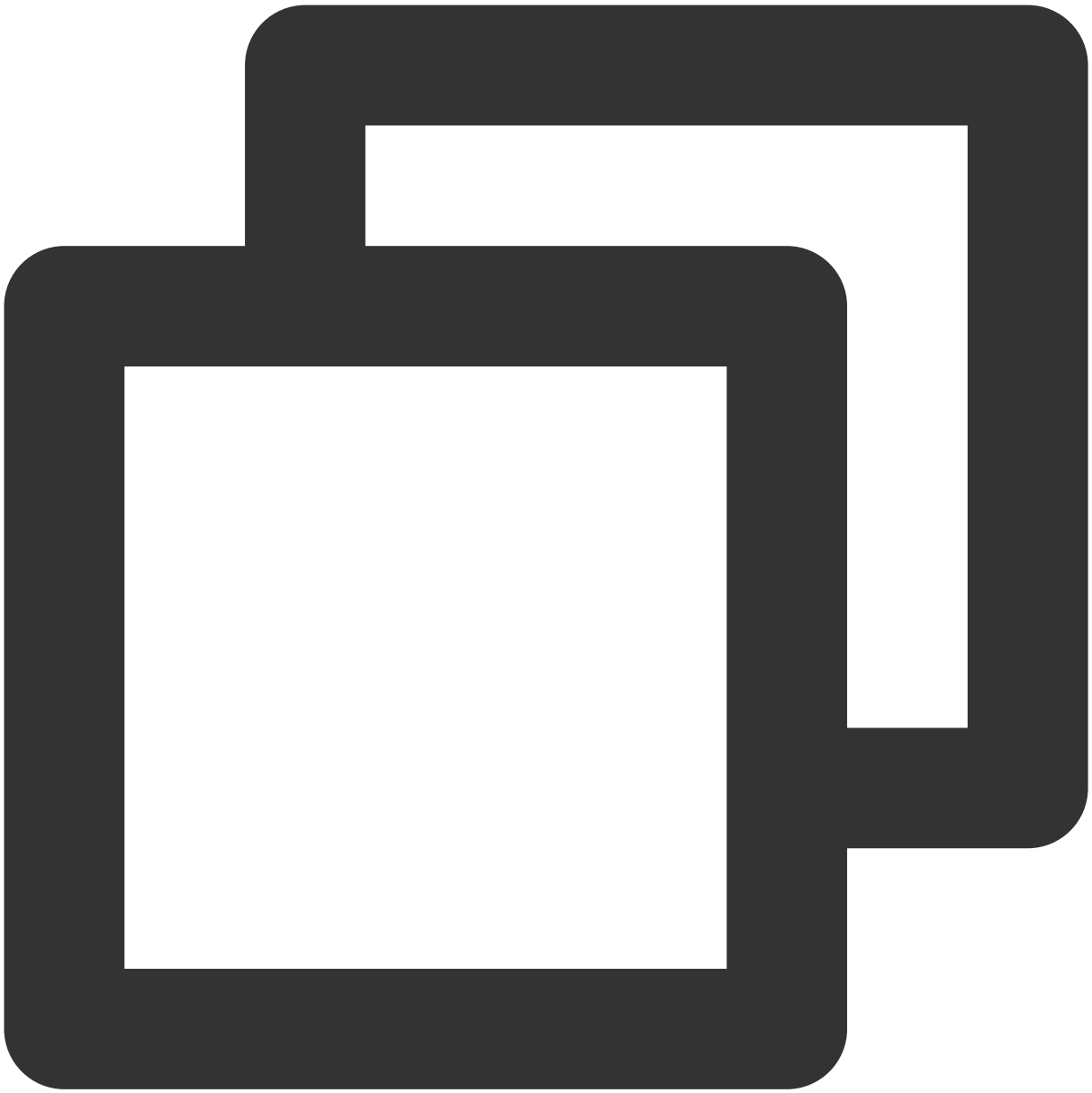
```
import TUIRoom

tuiRoom.enterRoom(roomId: 12345, isOpenCamera: true, isOpenMicrophone: true)
```

ステップ5：ルーム管理（オプション）

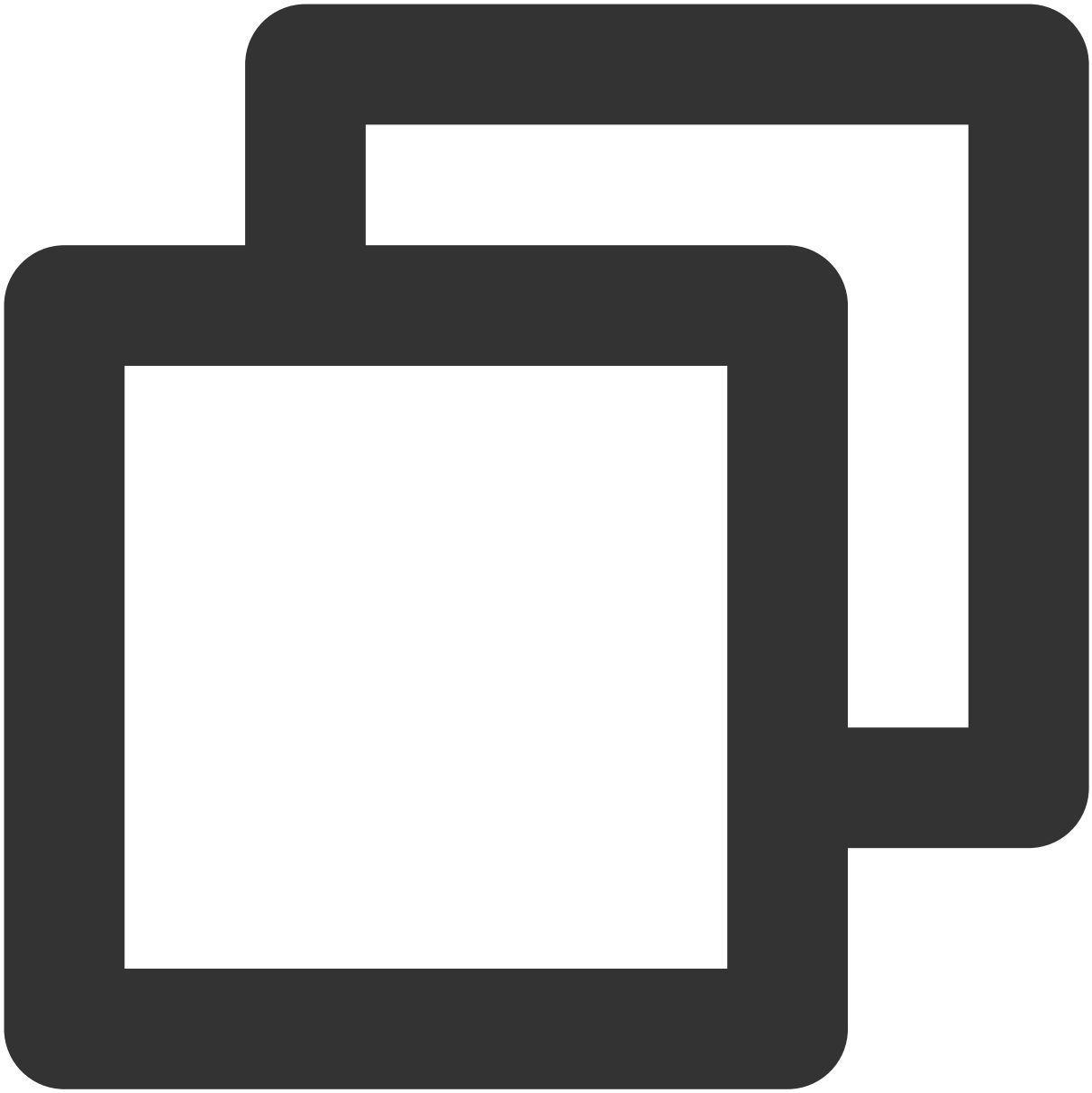
1. 管理者によるルーム解散 TUIRoomCore#destroyRoom。

Objective-C

Swift



```
@import TUIRoom;

[[TUIRoomCore shareInstance] destroyRoom:^(NSInteger code, NSString * _Nonnull mess

}];
```

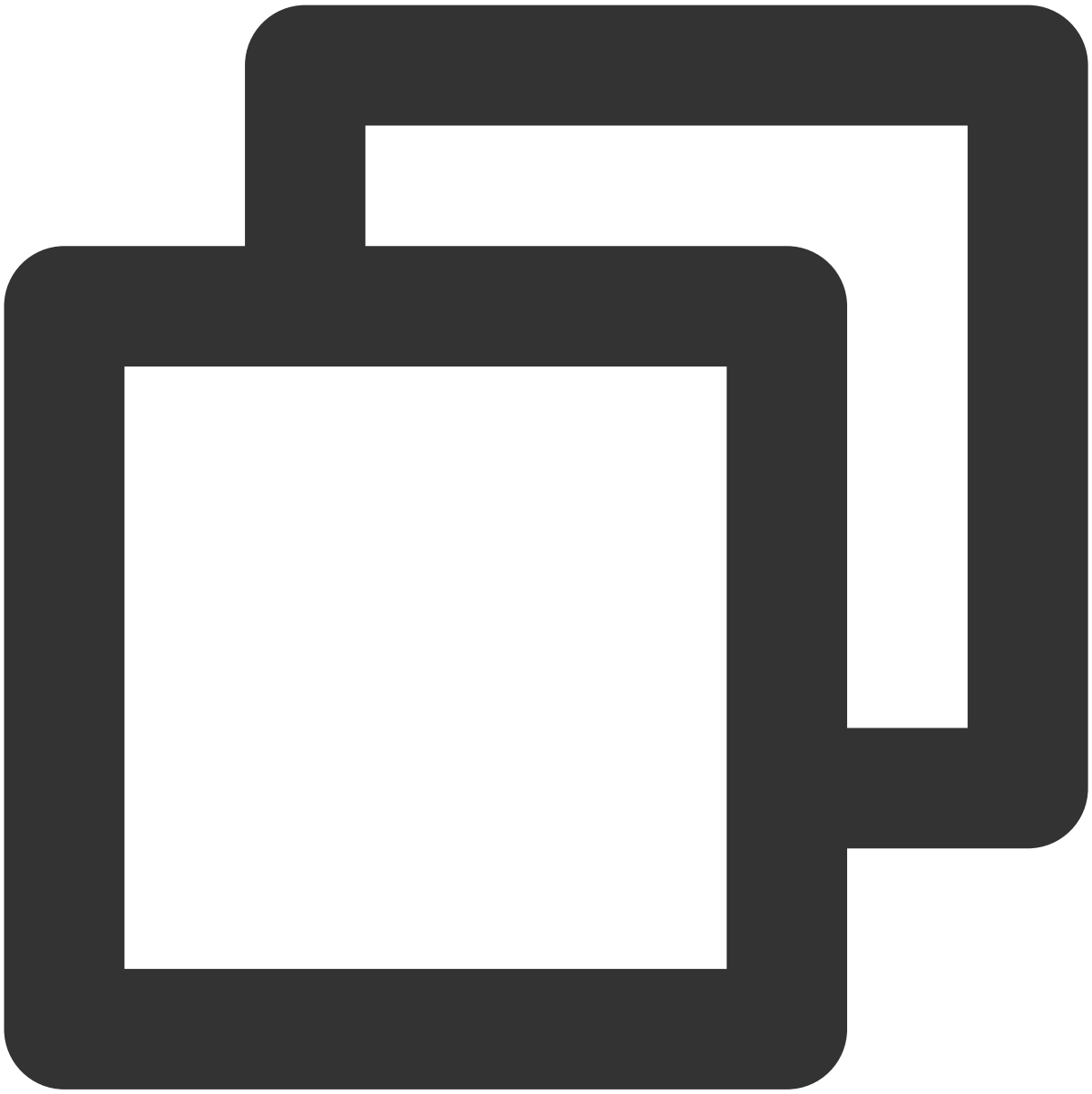


```
import TUIRoom

TUIRoomCore.shareInstance().destroyRoom { [weak self] _, _ in
    guard let self = self else { return }
    self.navigationController?.popViewController(animated: true)
}
```

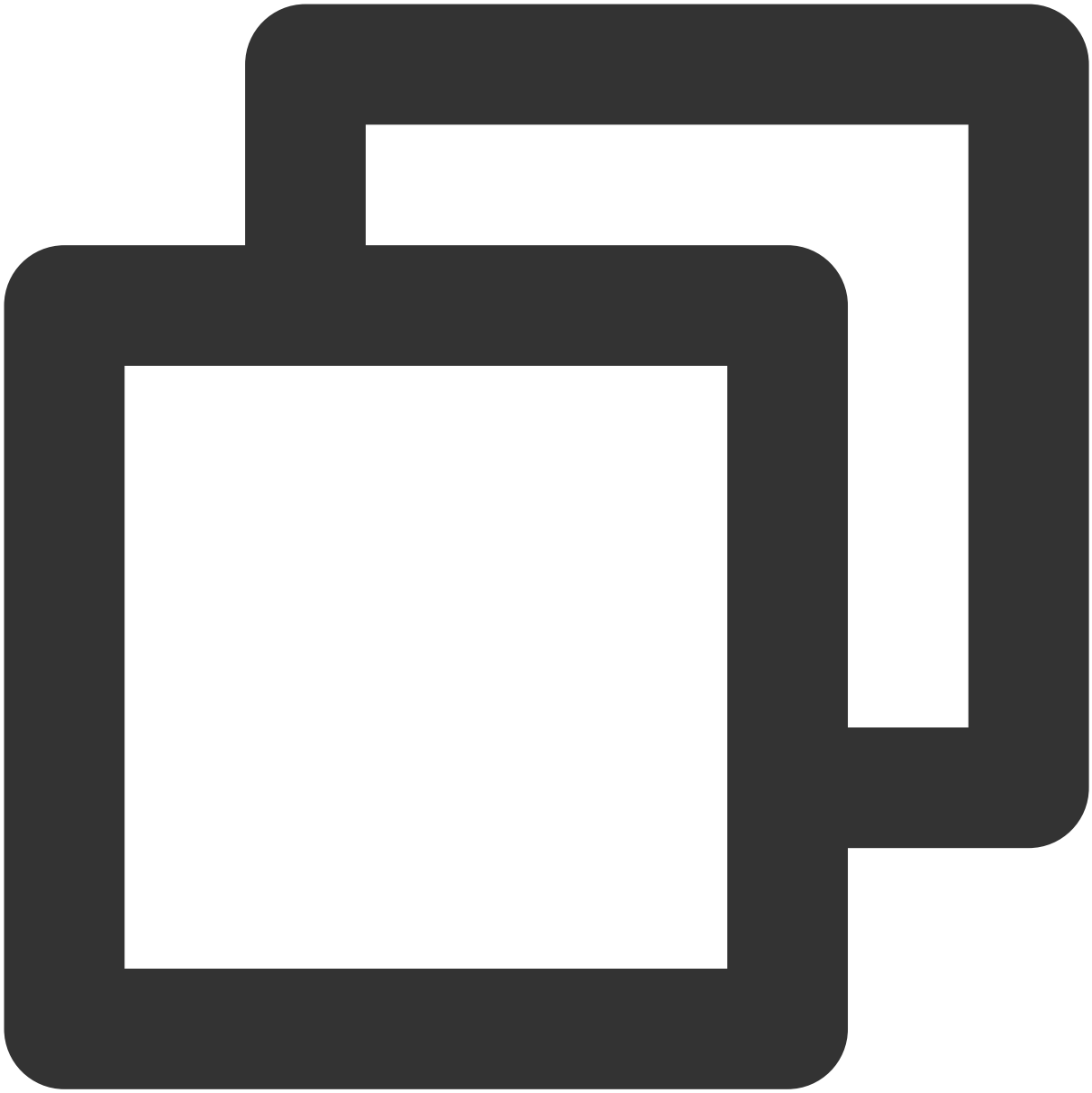
2. メンバーの退室 TUIRoomCore#leaveRoom。

Objective-C

Swift



```
@import TUIRoom;

[[TUIRoomCore shareInstance] leaveRoom:^(NSInteger code, NSString * _Nonnull messag

}];
```



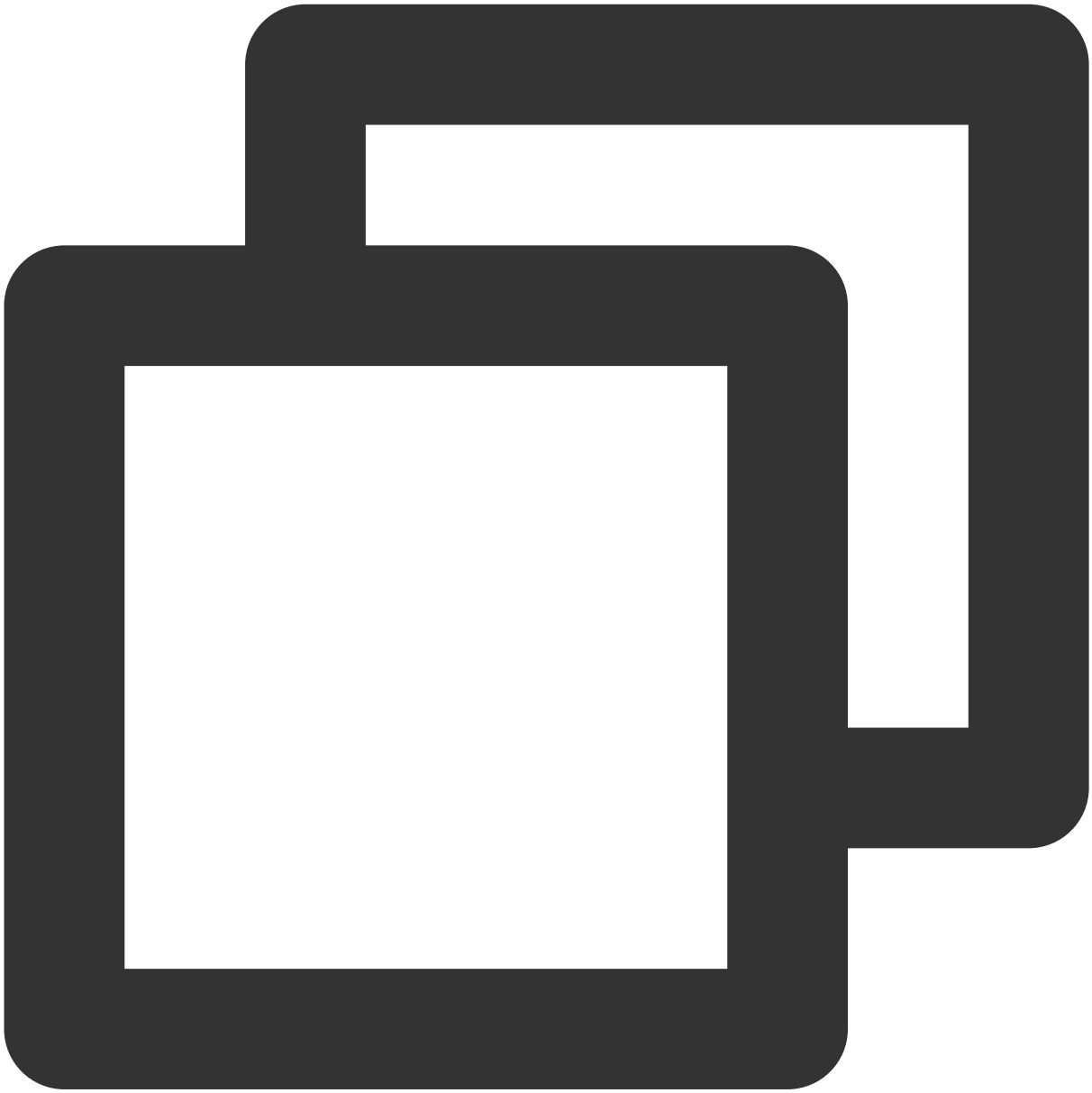
```
import TUIRoom

TUIRoomCore.shareInstance().leaveRoom { [weak self] _, _ in
    guard let self = self else { return }
    self.navigationController?.popViewController(animated: true)
}
```

ステップ6：画面共有（オプション）

画面共有TUIRoomCore#startScreenCaptureを実装します。画面共有プロジェクトの設定については、リアルタイム画面共有(iOS)をご参照ください。

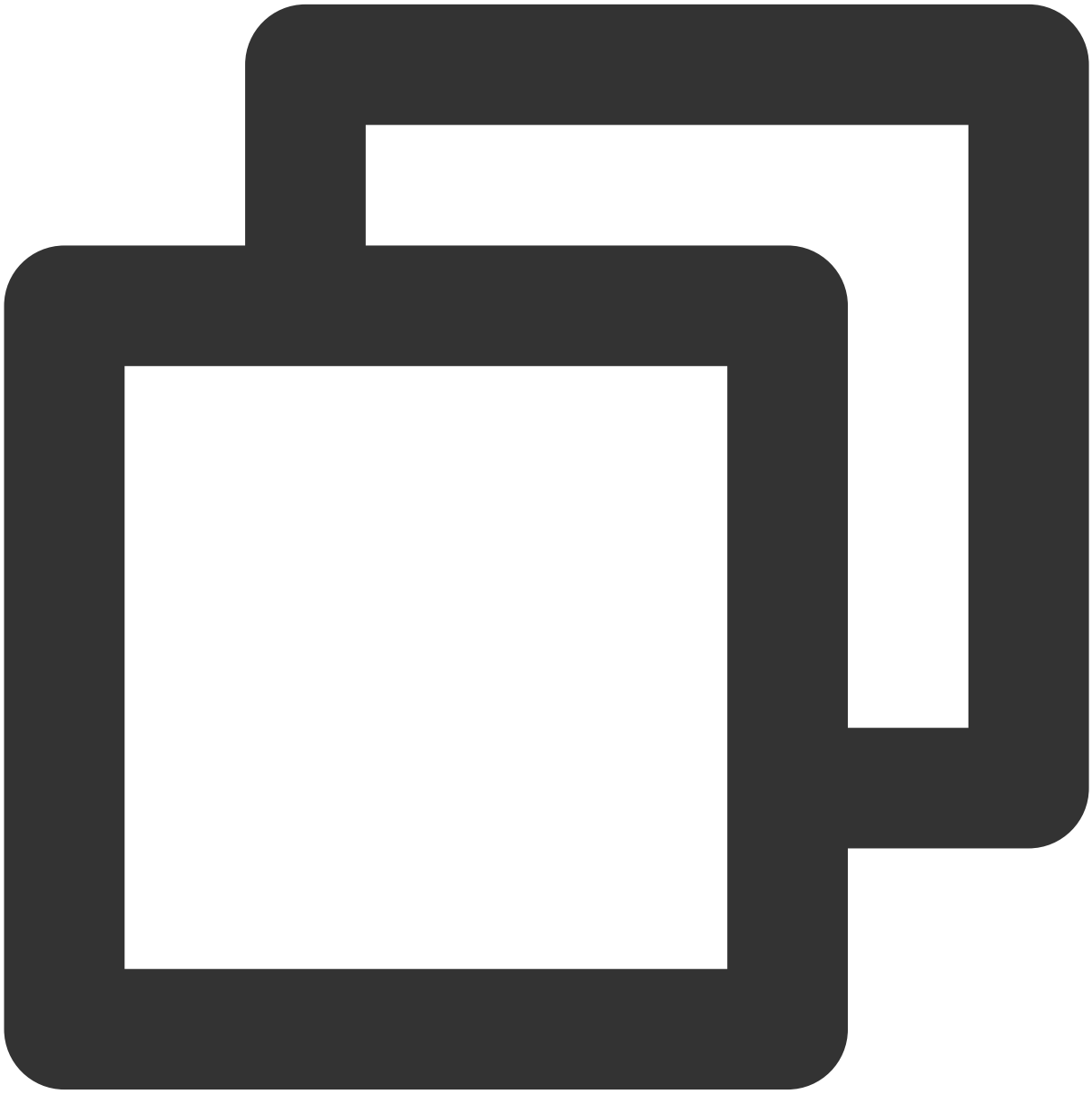
Objective-C

Swift



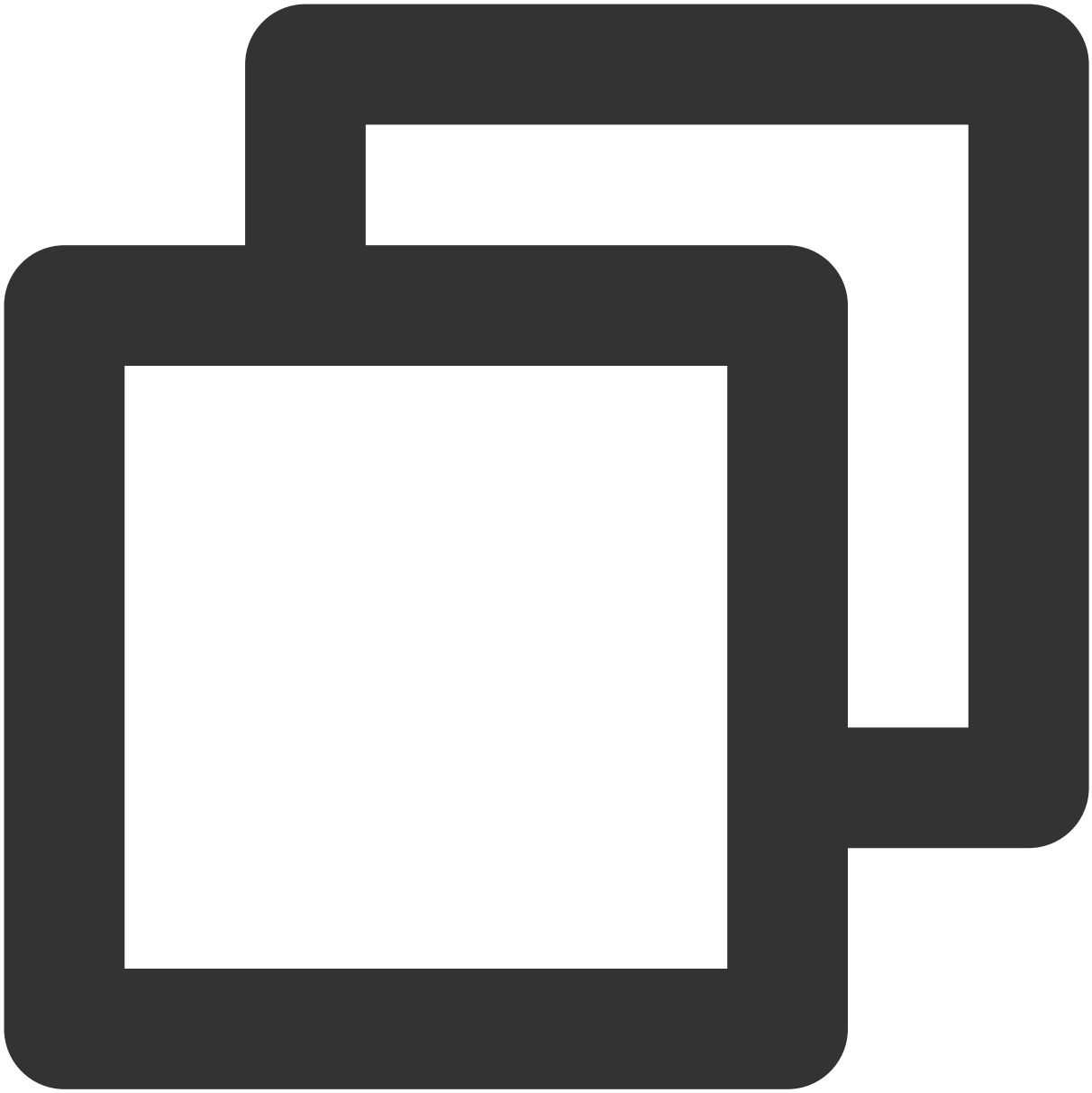
```
@import TUIRoom;
@import TXLiteAVSDK_Professional;

TRTCVideoEncParam *params = [[TRTCVideoEncParam alloc] init];
params.videoResolution = TRTCVideoResolution_1280_720;
params.resMode = TRTCVideoResolutionModePortrait;
```

```
params.videoFps = 10;
params.enableAdjustRes = NO;
params.videoBitrate = 1500;

[[TUIRoomCore shareInstance] startScreenCapture:param];
```



```
import TUIRoom

 // 画面共有
```

```
let params = TRTCVideoEncParam()
params.videoResolution = TRTCVideoResolution._1280_720
params.resMode = TRTCVideoResolutionMode.portrait
params.videoFps = 10
params.enableAdjustRes = false
params.videoBitrate = 1500
TUIRoomCore.shareInstance().startScreenCapture(params)

```
```

よくあるご質問

CocoaPodsをインストールするにはどうすればよいですか？

端末のウィンドウに次のコマンドを入力します（事前にMac にRuby環境をインストールしてください）。
```



```
sudo gem install cocoapods
```

説明：

ご要望やフィードバックなどがございましたら、colleenyu@tencent.comまでご連絡ください。

TUIRoom (Flutter)の統合

最終更新日：：2024-07-19 15:35:35

当社のAppをダウンロードしてインストールし、多人数オーディオビデオルームの効果をご体験いただけます。これには、画面共有、美顔、低遅延ミーティングなど、TRTCの多人数オーディオビデオルームシーン関連の機能が含まれています。

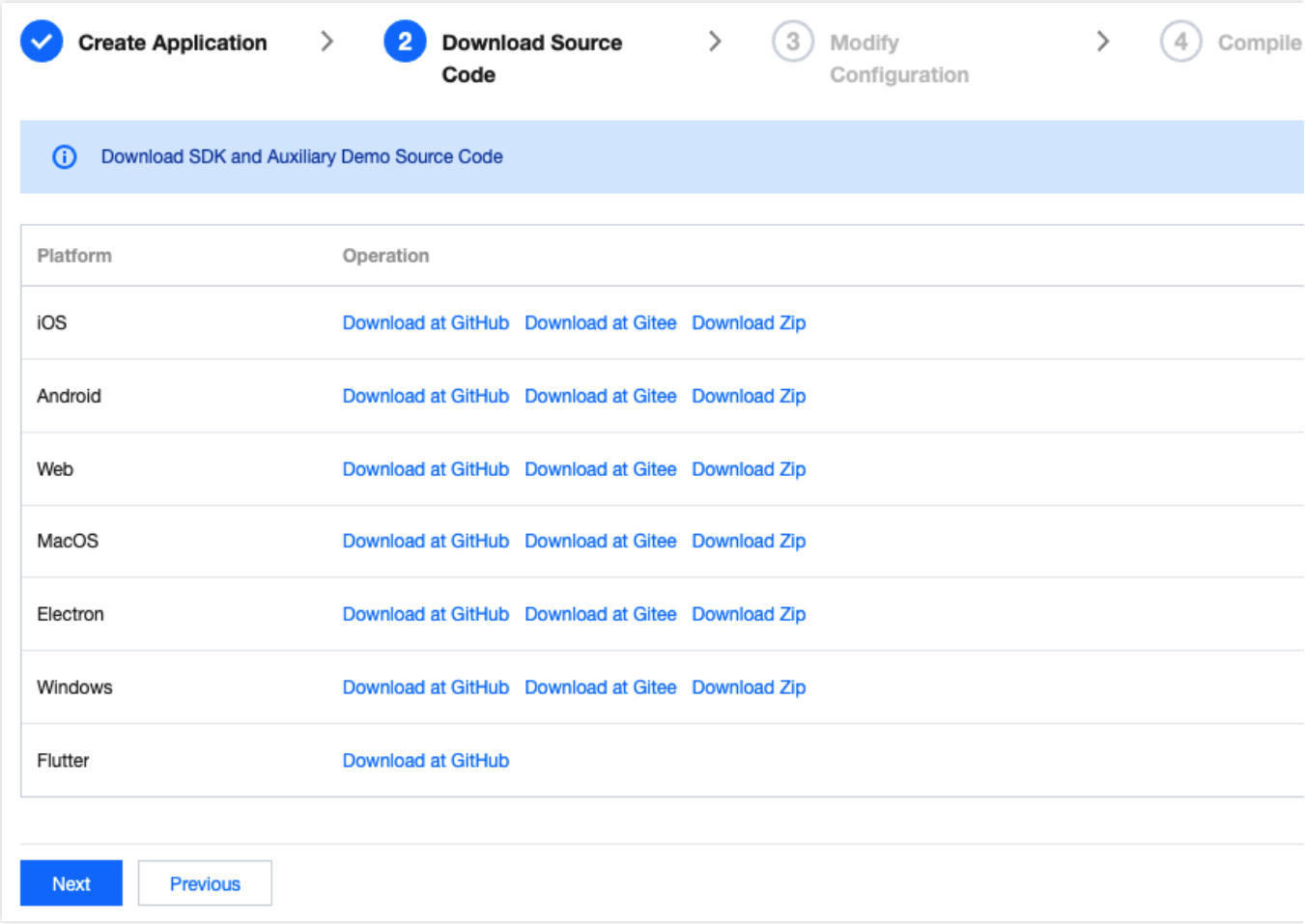
説明：

TUIKitシリーズコンポーネントはTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期的にアクティブにすることができます。IMサービスの課金ルールの詳細については、Instant Messagingの料金説明をご参照ください。TRTCをアクティブにすると、デフォルトでは、100DAUまでサポートするIM SDK体験版もアクティブになります。

AppのUIの再利用

ステップ1：新規アプリケーションの作成

1. TRTCコンソールにログインし、**開発支援** > **Demoクイックスタート**を選択します。
2. アプリケーション名（例： `TestMeetingRoom` ）を入力し、**作成**をクリックします。
3. **ダウンロードしました。次のステップ**をクリックすると、この手順をスキップします。



ご注意：

本機能はTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期的にアクティブにすることができます。IMは付加価値サービスであり、課金ルールの詳細については 、Instant Messagingの料金説明をご参照ください。

ステップ2：Appソースコードのダウンロード

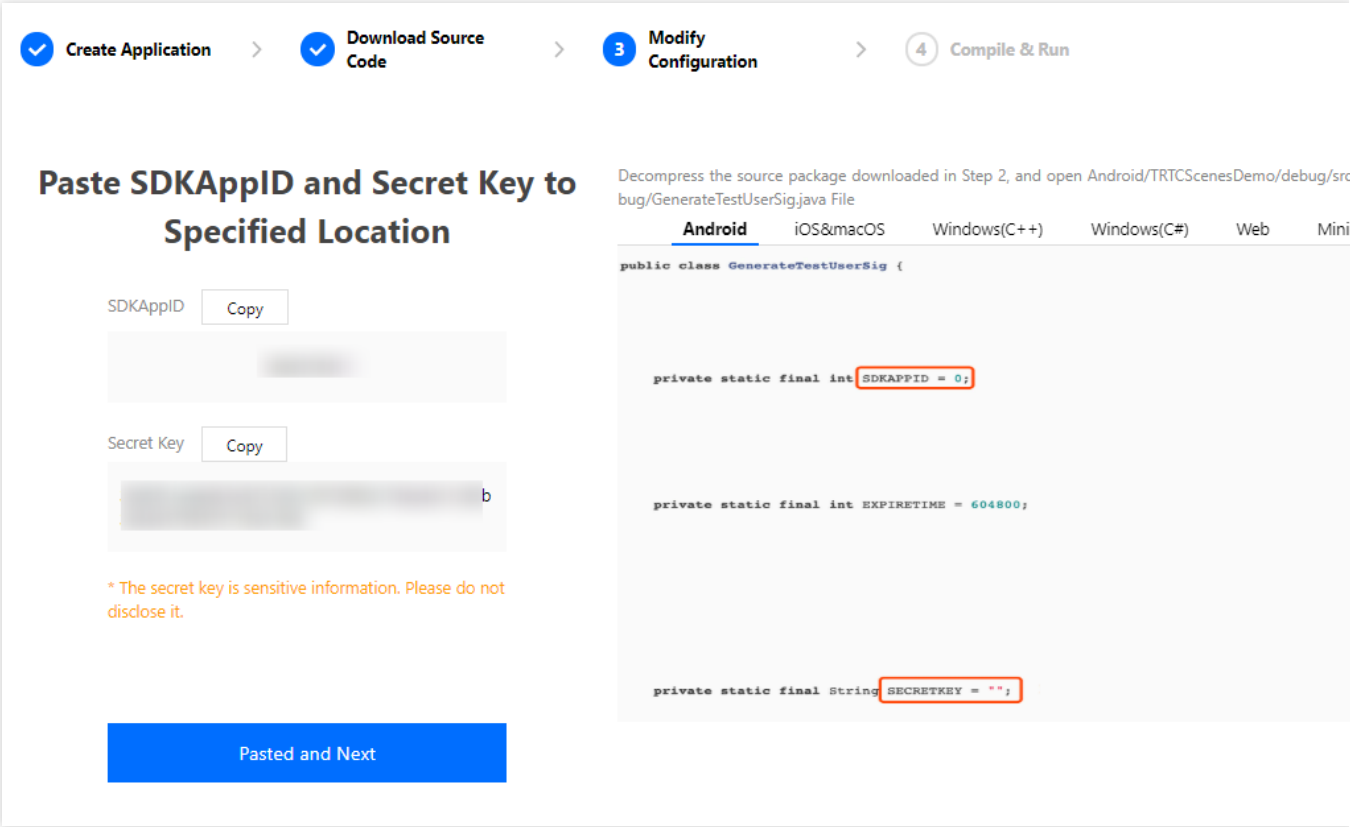
TRTCFlutterScenesDemoクリックして当該ページに入り、ソースコードをCloneまたはダウンロードします。

ステップ3：Appファイルの設定

1. 設定変更画面に進み、ダウンロードしたソースコードパッケージに基づき、対応する開発環境を選択します。
2. `/lib/debug/GenerateTestUserSig.dart` ファイルを見つけて開きます。
3. `GenerateTestUserSig.dart` ファイル内の関連パラメータを設定します：

SDKAPPID: A placeholder by default. Set it to the actual `SDKAppID`.

SECRETKEY: A placeholder by default. Set it to the actual secret key.



4. 貼り付け完了後、**貼り付けました。次のステップ**をクリックすれば、作成が完了します。
5. コンパイル完了後、 **コンソール概要に戻る** をクリックすれば終了です。

ご注意：

ここで言及したUserSigの新規作成ソリューションでは、クライアントコードでSECRETKEYを設定します。この手法のうちSECRETKEYは逆コンパイルによって逆向きにクラッキングされやすく、キーがいったん漏洩すると、攻撃者はTencent Cloudトラフィックを盗用できるようになります。そのため**この手法は、ローカルのDemoクイックスタートおよび機能デバッグにのみ適合します**。

UserSigの正しい発行方法は、UserSigの計算コードをサーバーに統合し、App向けのインターフェースを提供します。UserSigが必要なときは、Appから業務サーバーにリクエストを発出し動的にUserSigを取得します。詳細はサーバーでUserSigを生成するをご参照ください。

ステップ4：コンパイルと実行

ご注意：

Android端末は実際のマシンで実行する必要があり、エミュレーターのデバッグはサポートされていません。

1. `flutter pub get` を実行します。
2. コンパイルを実行し、デバッグを行います：

iOS 端末

Android 端末

1. XCode（11.0およびそれ以降のバージョン）を使用して、ソースコードディレクトリの `/ios`プロジェクトを開きます。

2. コンパイルを行い、Demoプロジェクトを実行します。
1. `flutter run` を実行します。
2. Android Studio（3.5およびそれ以降のバージョン）を使用して、ソースプロジェクトを開き、**実行**をクリックすれば完了です。

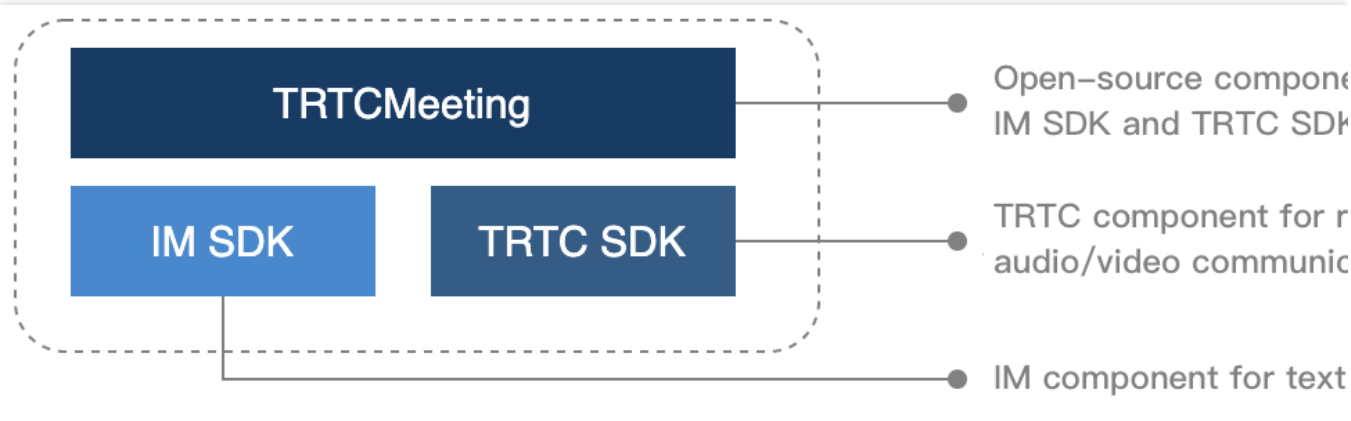
ステップ5：Demoソースコードの修正

ソースコードのTRTCMeetingDemoフォルダには、uiとmodelの2つのサブフォルダが含まれ、uiフォルダの中はいずれもインターフェースコードです。以下の表にはファイルまたはフォルダおよび対応するUIをリストアップし、二次調整に便利です：

ファイルまたはフォルダ	機能説明
TRTCMeetingIndex.dart	ミーティング作成または参加インターフェース
TRTCMeetingRoom.dart	ビデオミーティングのメインインターフェース
TRTCMeetingMemberList.dart	参加者リストインターフェース
TRTCMeetingSetting.dart	ビデオミーティング関連パラメータ設定インターフェース

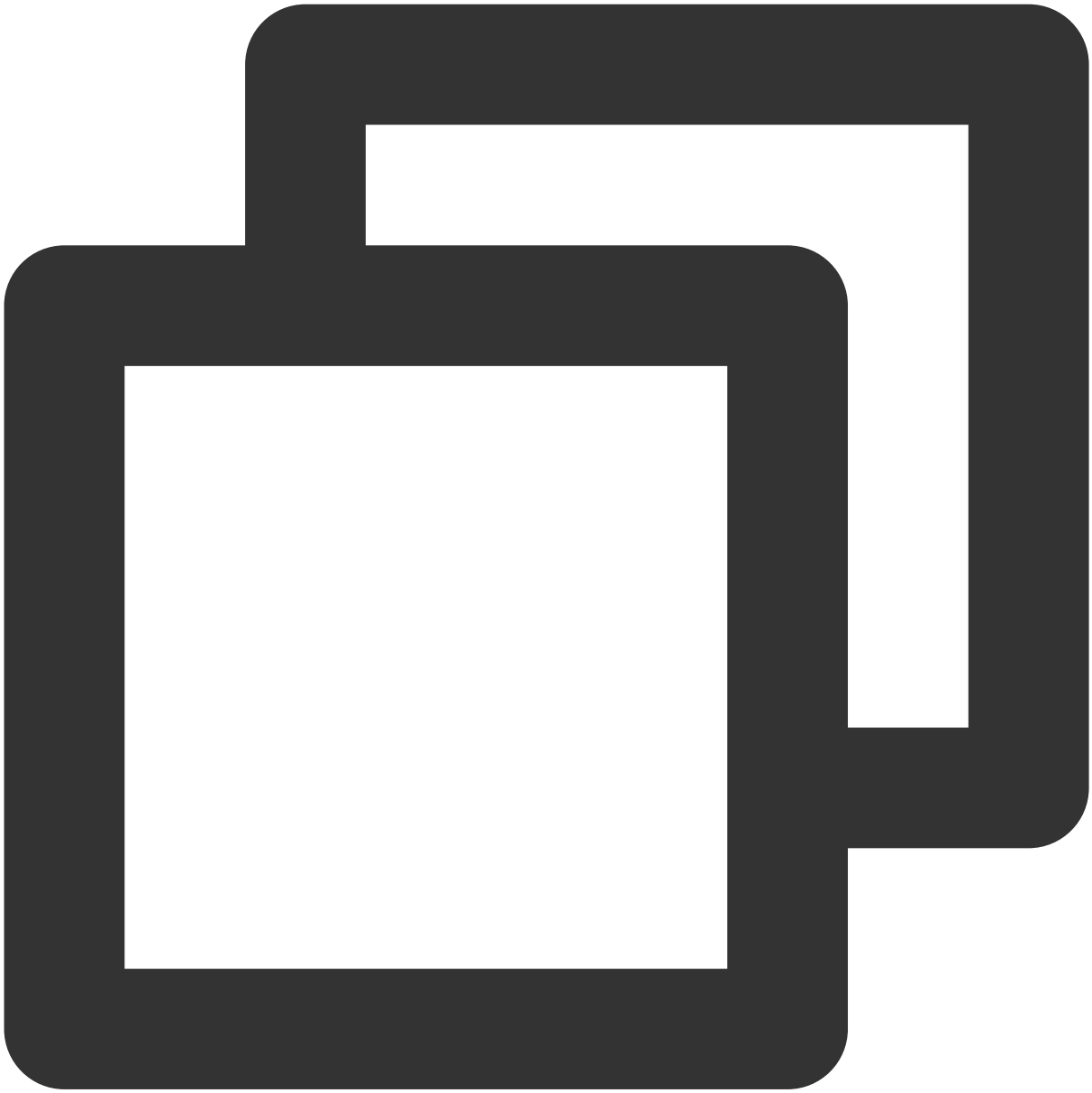
UIカスタマイズの実装

ソースコードの中のTRTCMeetingDemoフォルダにはuiとmodelの2つのサブフォルダが含まれ、modelフォルダには再利用可能なオープンソースコンポーネントTRTCMeetingが含まれています。 `TRTCMeeting.dart` ファイルの中からこのコンポーネントが提供するインターフェース関数を見つけ、対応するインターフェースを使用してカスマイズUIを実装することが可能です。



ステップ1：SDKの統合

インタラクティブライブストリーミングコンポーネントTRTCMeetingは、TRTC SDKとIM SDKに依存しています。 `pubspec.yaml` を設定することで、自動的にダウンロードして更新できます。
プロジェクトの `pubspec.yaml` に次の依存関係を記述します：



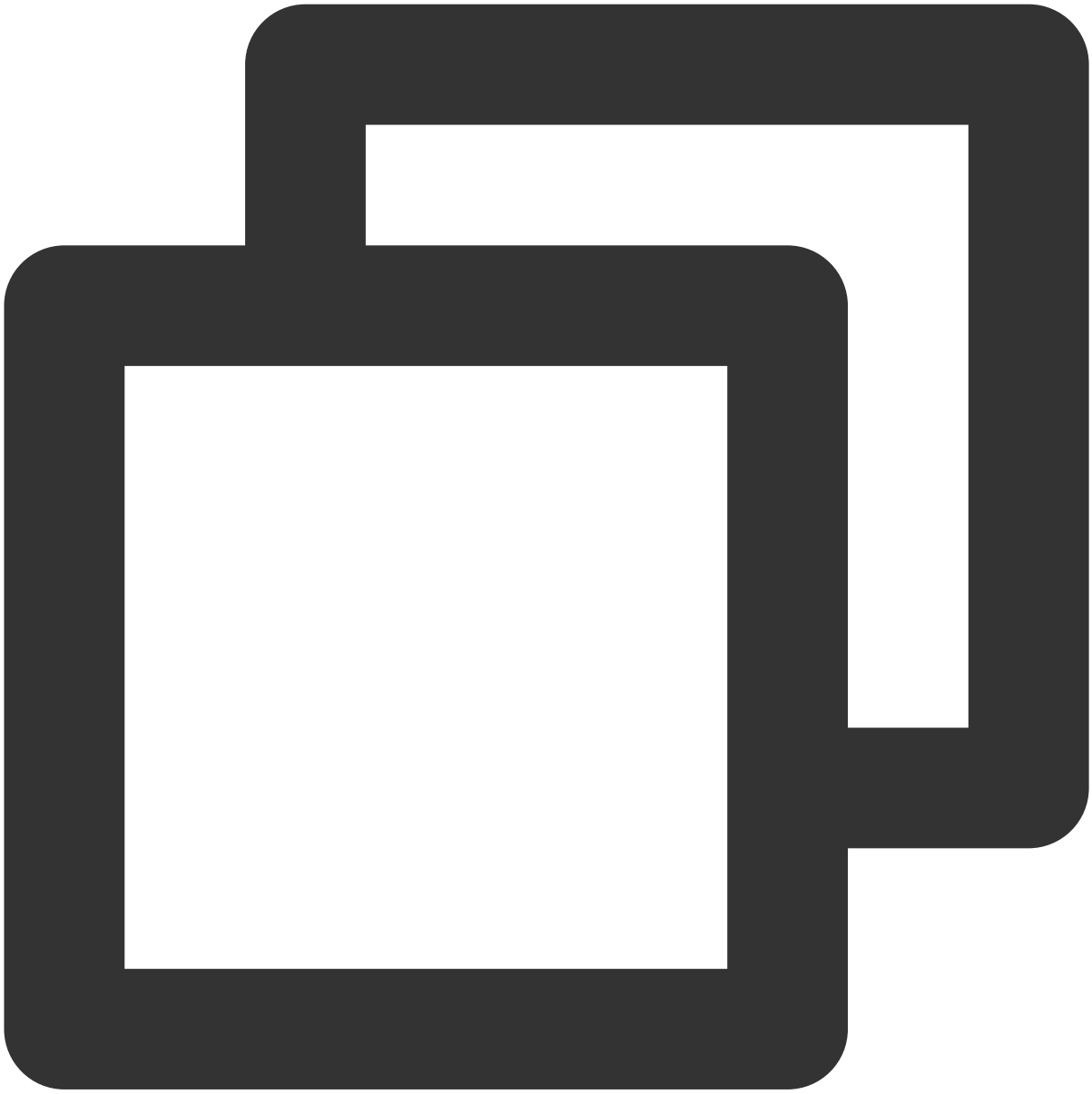
```
dependencies:
  tencent_trtc_cloud: 最新バージョン番号
  tencent_im_sdk_plugin: 最新バージョン番号
```

ステップ2：権限の設定および難読化ルール

iOS 端末

Android 端末

`Info.plist` にカメラとマイクの権限申請を追加してください：

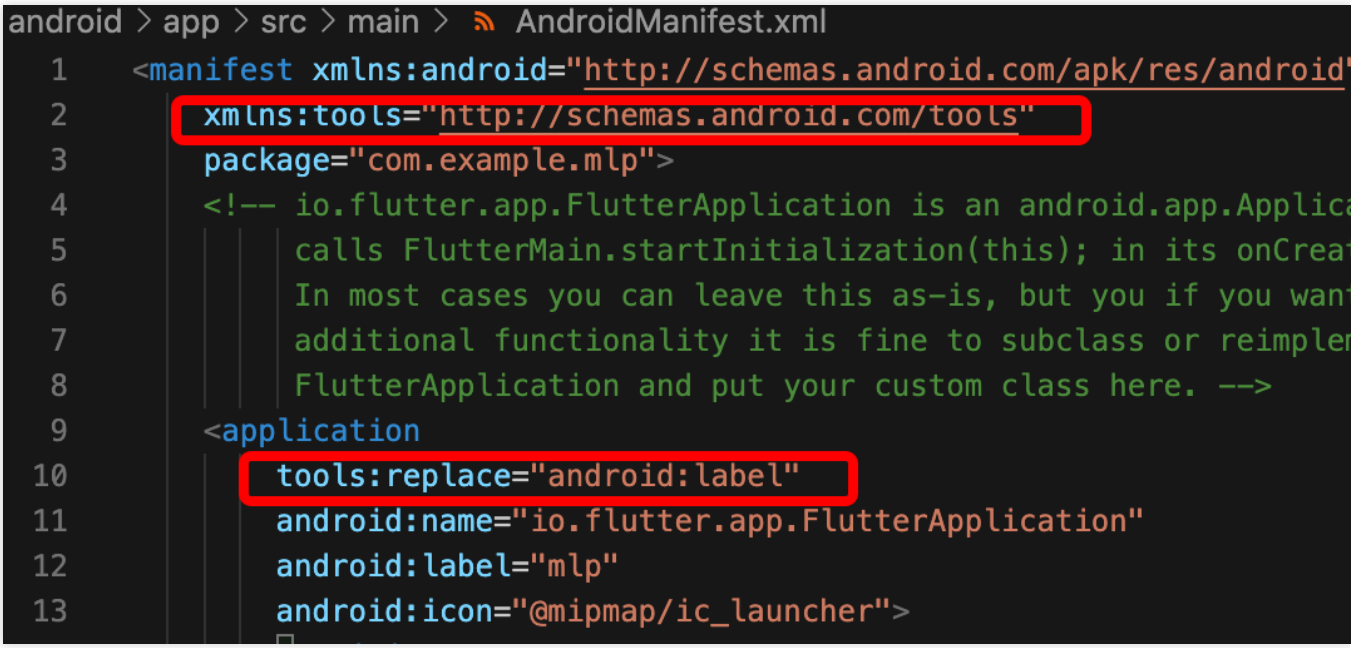


```
<key>NSMicrophoneUsageDescription</key>
<string>通常の音声通話が行えるようにマイクの権限を承認します</string>
```

1. `/android/app/src/main/AndroidManifest.xml` ファイルを開きます。
2. `xmlns:tools="http://schemas.android.com/tools"` をmanifestの中に追加します。
3. `tools:replace="android:label"をapplicationの中に追加します。

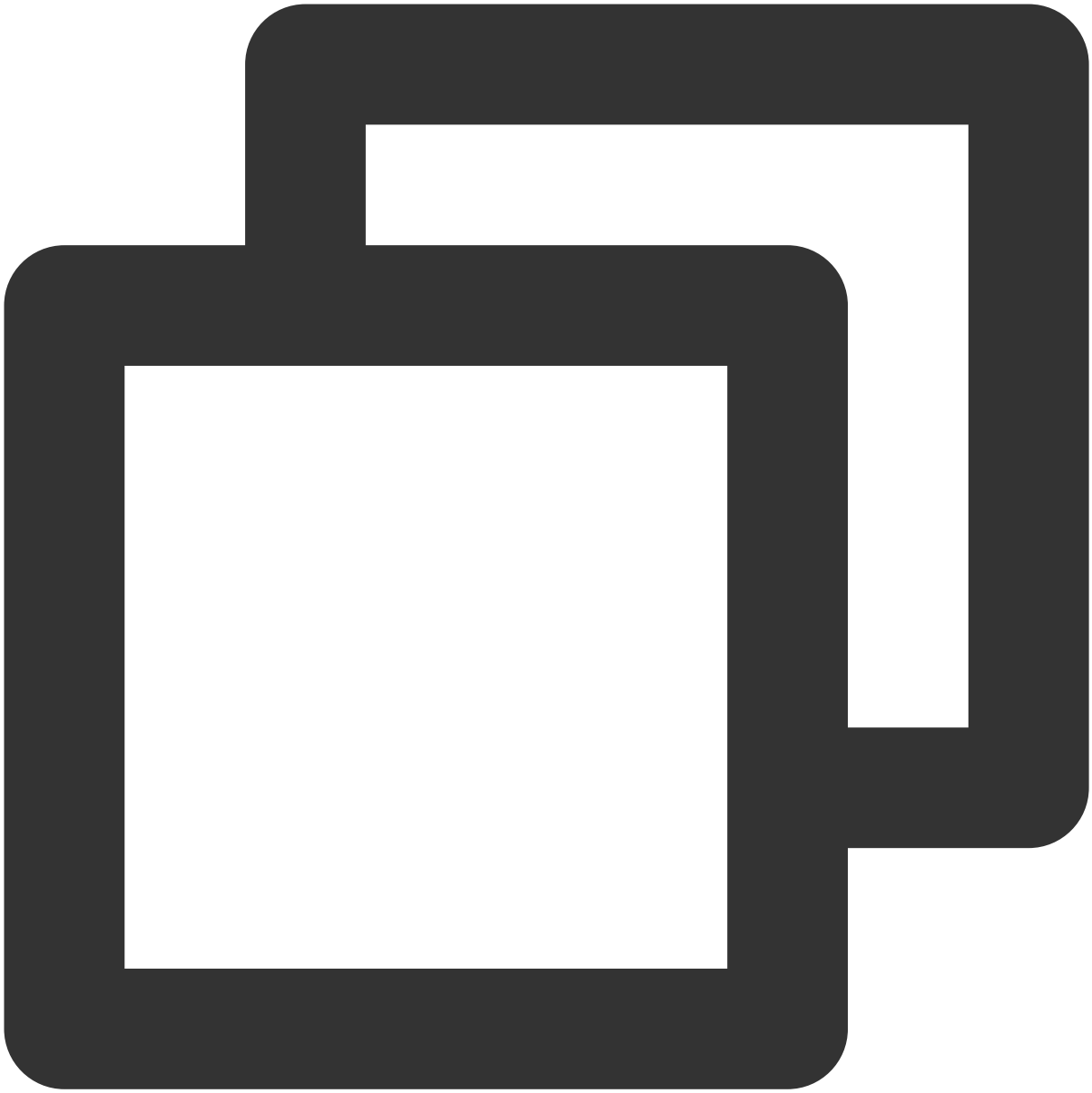
explain

この手順を実行しないと、Android Manifest merge failedコンパイルの失敗という問題が発生します。



ステップ3：TRTCMeetingコンポーネントのインポート

次のディレクトリ内のすべてのファイルをプロジェクトにコピーします：

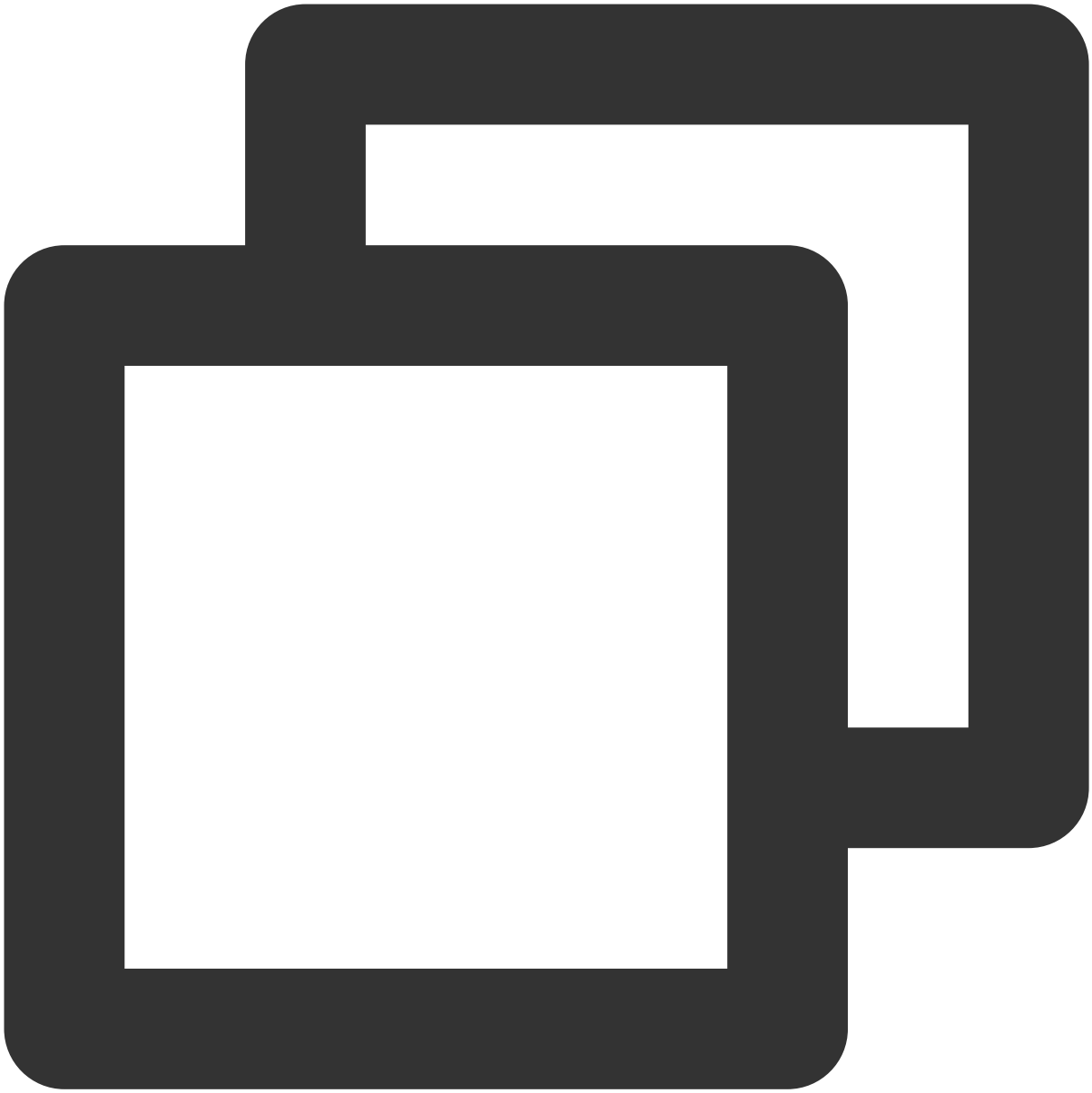


```
lib/TRTCMeetingDemo/model/
```

ステップ4：コンポーネントの作成およびログイン

1. `sharedInstance` インターフェースを呼び出すと、TRTCMeetingコンポーネントのインスタンスオブジェクトを作成できます。
2. `registerListener` 関数を呼び出して、コンポーネントのイベント通知を登録します。
3. `login` 関数を呼び出してコンポーネントのログインを完了します。下表を参考にキーパラメータを入力してください：

パラメータ名	作用
sdkAppId	TRTCコンソール で SDKAppIDを表示できます。
userId	現在のユーザーID。文字列タイプであり、英語のアルファベット（a-z、A-Z）、数字（0-9）、ハイフン（-）とアンダーライン（_）のみ使用できます。業務の実際のアカウントシステムと組み合わせてご自身で設定することをお勧めします。
userSig	Tencent Cloudによって設計されたセキュリティ保護署名。取得方法については、UserSigの計算、使用方法をご参照ください。

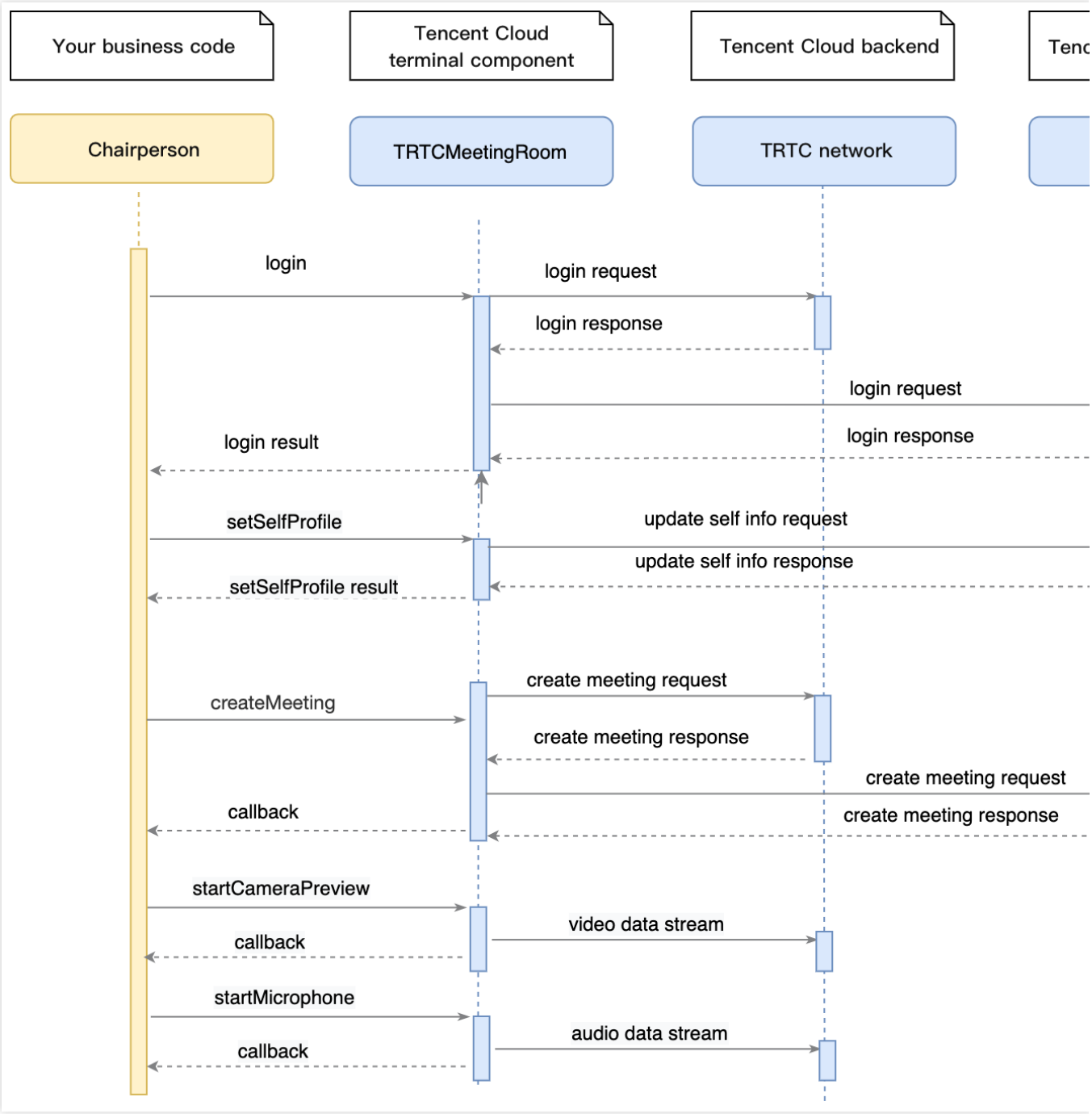


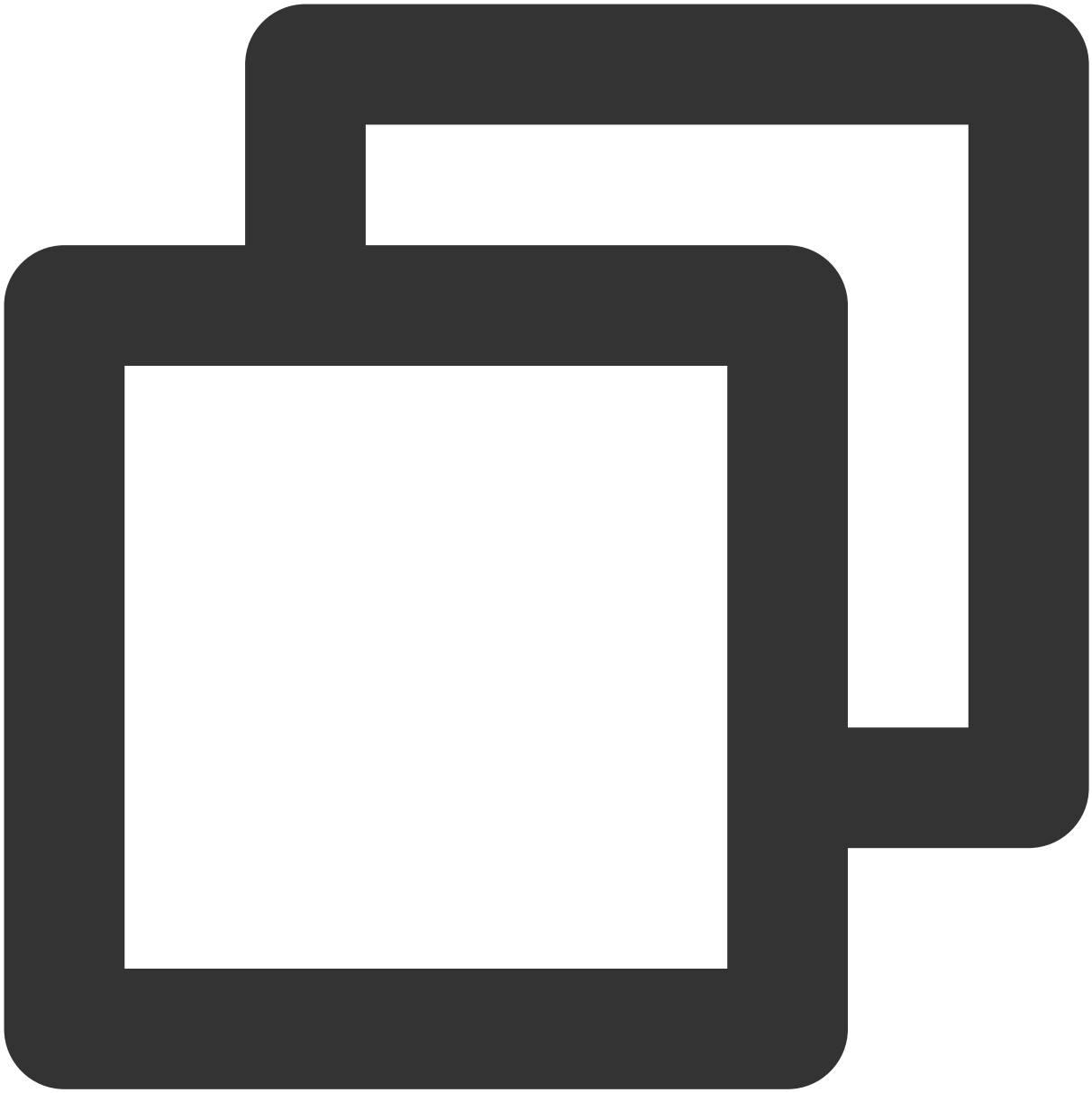
```
TRTCMeeting trtcMeeting = TRTCMeeting.sharedInstance();
trtcMeeting.registerListener(onListener);
ActionCallback res = await trtcMeeting.login(
    GenerateTestUserSig.sdkAppId,
    userId,
    GenerateTestUserSig.genTestSig(userId),
);
if (res.code == 0) {
    // ログイン成功
}
```

ステップ5：多人数ミーティングの新規作成

1. キャスターは、手順4でログインした後、 `setSelfProfile` を呼び出して自身のニックネームおよびプロフィール画像を設定することができます。
2. キャスターがcreateMeetingを呼び出し、新しいミーティングルームを作成します。
3. キャスターは、 `startCameraPreview` を呼び出してビデオ画面をキャプチャすることができ、 `startMicrophone` を呼び出して音声をキャプチャすることもできます。
4. キャスターに美顔のニーズがある場合は、インターフェース上に美顔調節ボタンを設定して呼び出すことができ、getBeautyManager を介して美顔設定を行います。

説明：
エンタープライズ版以外のSDKは変顔やスタンプの機能をサポートしていません。



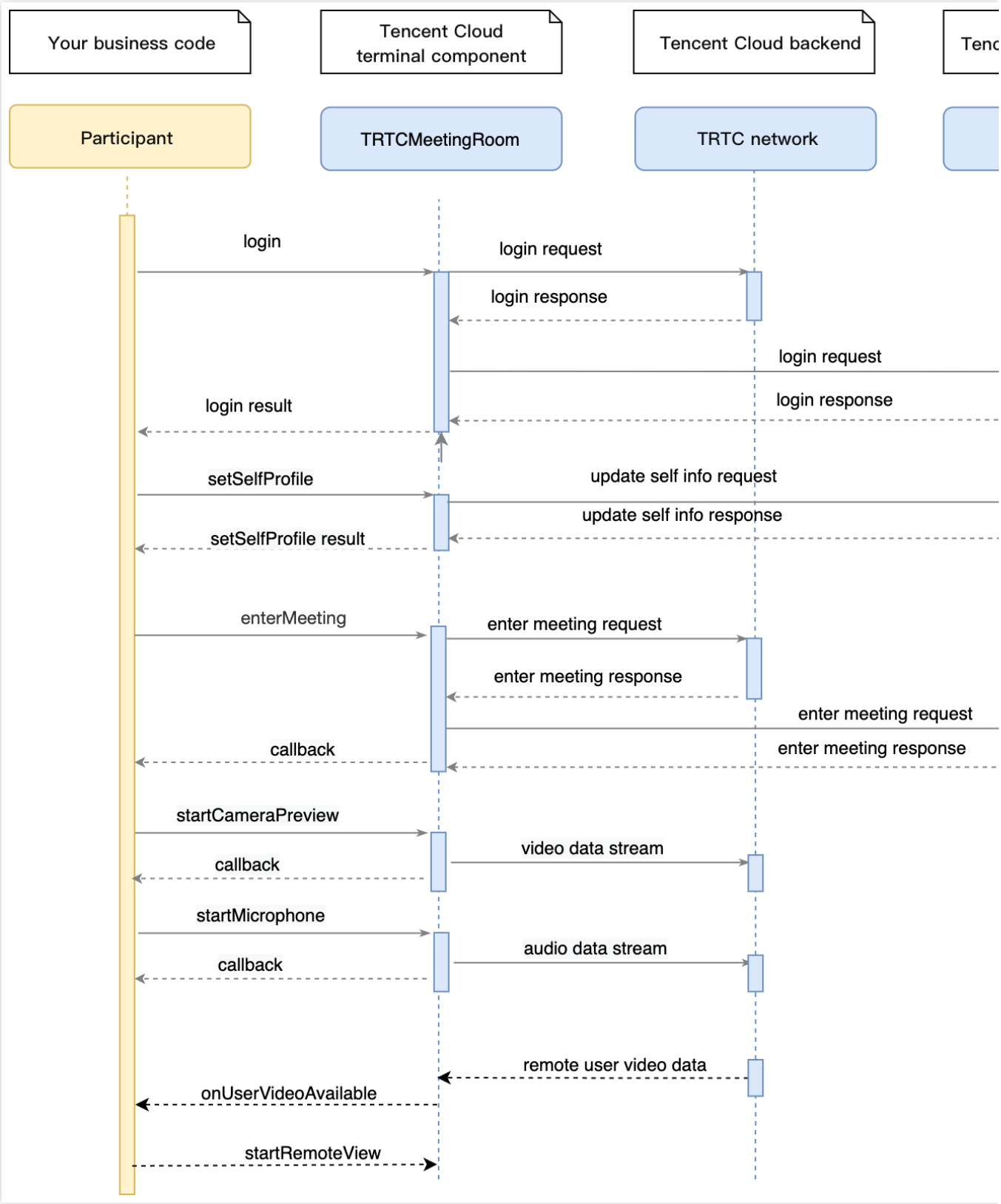


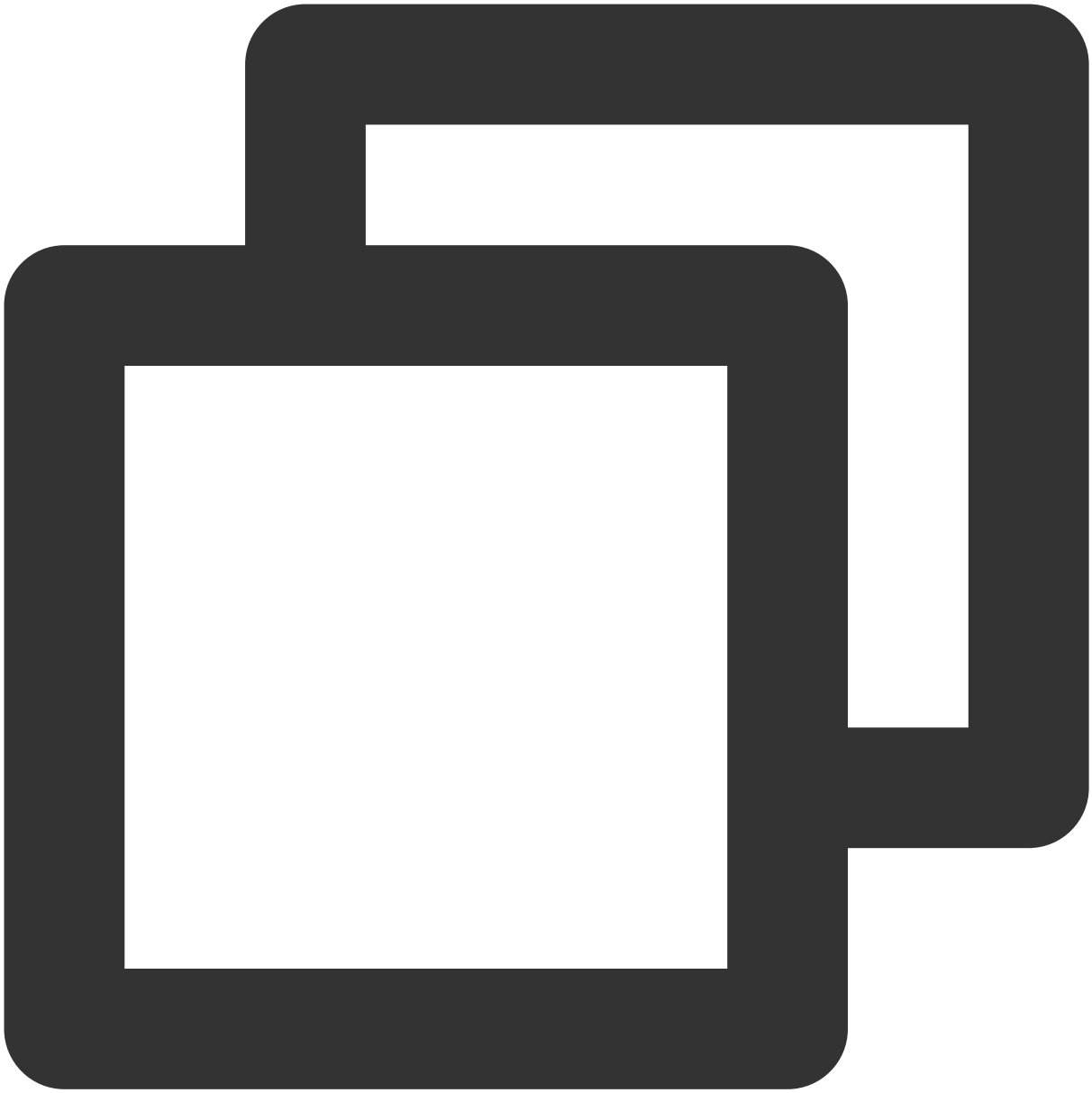
```
// 1. キャスターがニックネームとプロフィール画像を設定
trtcMeeting.setSelfProfile('my_name', 'my_avatar');

// 2. キャスターがミーティングを作成
ActionCallback res = await trtcMeeting.createMeeting(roomId);
if (res.code == 0) {
    // ミーティング作成に成功
    // 3. カメラと音声キャプチャを起動
    trtcMeeting.startCameraPreview(true, TRTCCloudVideoViewId);
    trtcMeeting.startMicrophone();
    // 4. 美顔設定
```

```
    trtcMeeting.getBeautyManager().setBeautyStyle(TRTCCloudDef.TRTC_BEAUTY_STYLE_PI
    trtcMeeting.getBeautyManager().setBeautyLevel(6);
}
```

ステップ6：参加者の多人数ミーティングへの参加

1. 参加者は、手順4でログインした後、setSelfProfileを呼び出して自身のニックネームおよびプロフィール画像を設定することができます。
2. 参加者は、enterMeeting呼び出してミーティングルーム番号を渡すと、ミーティングルームに参加できます。
3. 参加者は、startCameraPreviewを呼び出してビデオ画面をキャプチャすることができ、またstartMicrophoneを呼び出して音声をキャプチャすることもできます。
4. 他の参加者がカメラを起動すると、onUserVideoAvailableのイベントを受け取ります。この時、startRemoteViewを呼び出してuserIdを渡すと再生を開始できます。





```
// 1. 参加者がニックネームとプロフィール画像を設定
trtcMeeting.setSelfProfile('my_name', 'my_avatar');

// 2. 参加者がenterMeetingを呼び出してミーティングルームに参加
ActionCallback res = await trtcMeeting.enterMeeting(roomId);
if (res.code == 0) {
    // ミーティング参加に成功
    // 3. カメラと音声キャプチャを起動
    trtcMeeting.startCameraPreview(true, TRTCCloudVideoViewId);
    trtcMeeting.startMicrophone();
    // 4. 美顔設定
```

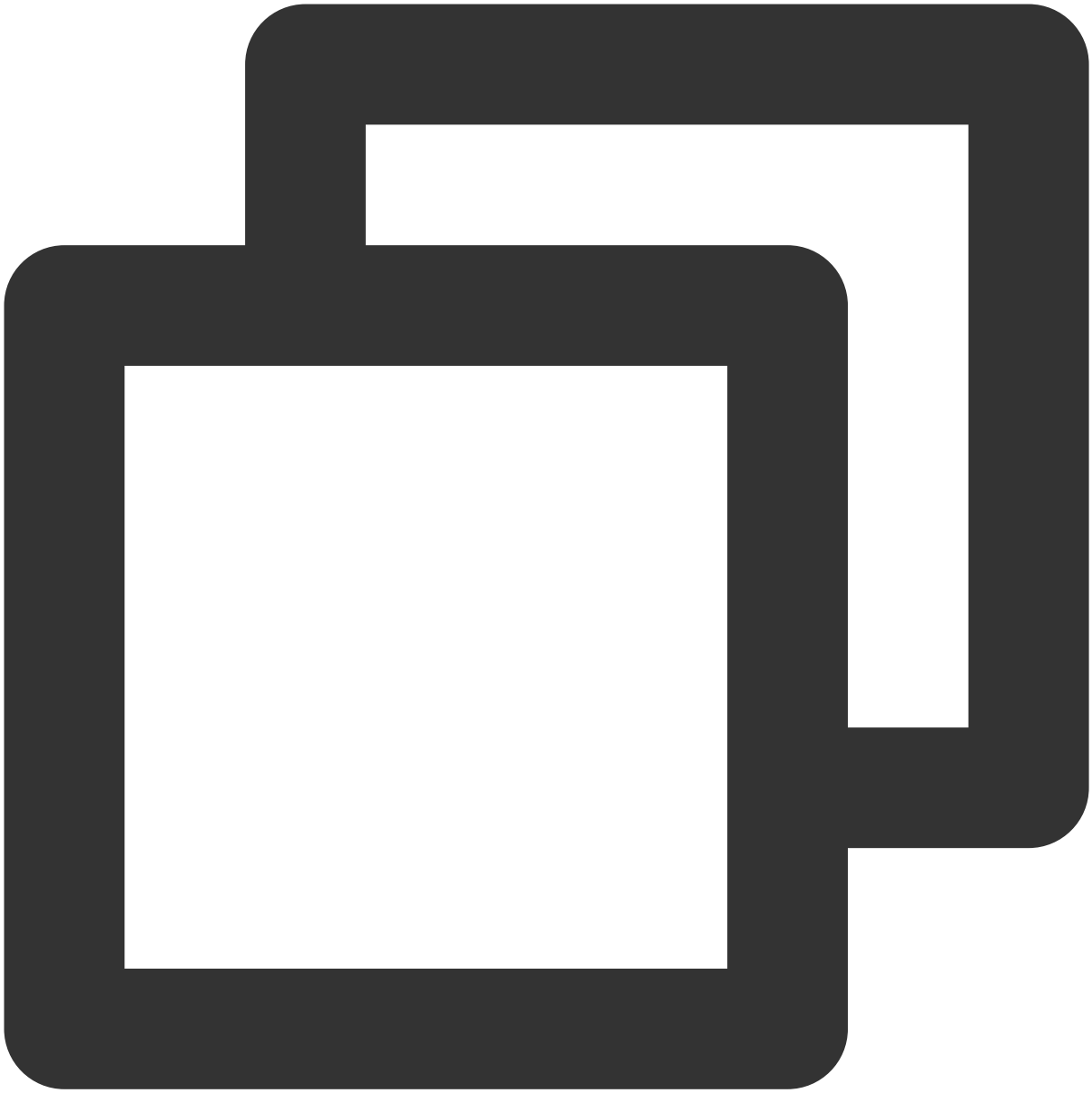
```
    trtcMeeting.getBeautyManager().setBeautyStyle(TRTCCloudDef.TRTC_BEAUTY_STYLE_PI
    trtcMeeting.getBeautyManager().setBeautyLevel(6);
}

// 5. 参加者が他のメンバーのカメラ起動の通知を受信し、再生が開始されます
trtcMeeting.registerListener(onListener);
onListener(TRTCMeetingDelegate type, param) {
    switch (type) {
        case TRTCMeetingDelegate.onUserVideoAvailable:
            if (param['available']) {
                trtcMeeting.startCameraPreview(
                    param['userId'],
                    TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG,
                    TRTCCloudVideoViewId
                );
            }else{
                trtcMeeting.stopRemoteView(
                    param['userId'],
                    TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG
                );
            }
            break;
    }
}
```

ステップ7：画面共有

1. 画面共有機能は、システムにフローティングウィンドウの権限を申請する必要があり、自分でUIの中で特定のロジックを実装してください。
2. startScreenCaptureを呼び出し、エンコードパラメータとスクリーンキャプチャのプロセスのフローティングウィンドウを渡すと、画面共有機能を実装できます。具体的な情報については、TRTC SDKをご参照ください。
3. ミーティング中の他メンバーがonUserVideoAvailableのイベント通知を受け取ります。

ご注意：

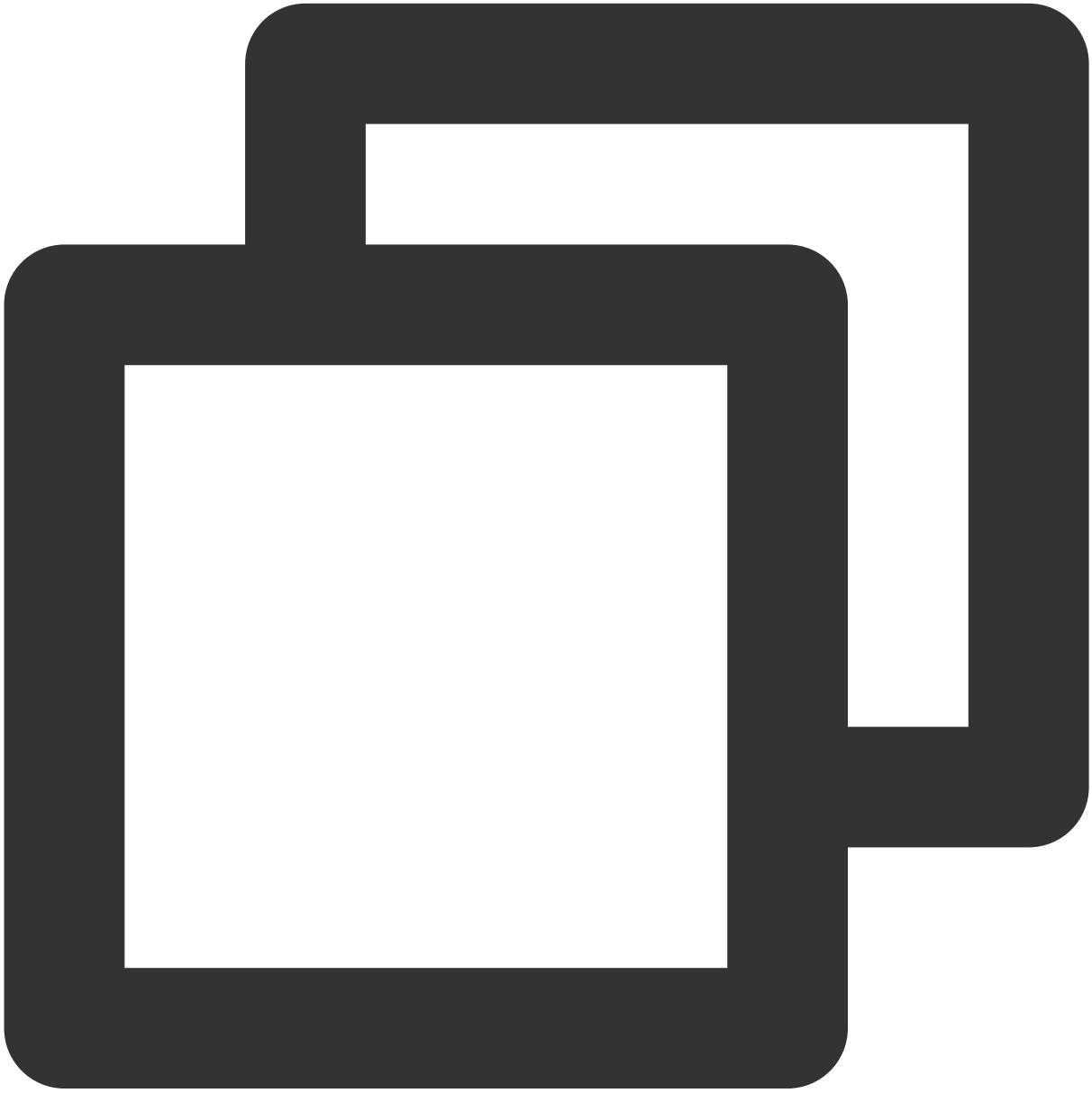
画面共有とカメラキャプチャは相互排他的な操作です。画面共有機能をアクティブにしたい場合は、先にstopCameraPreviewを呼び出してカメラキャプチャを停止してください。詳細については、TRTC SDKをご参照ください。



```
await trtcMeeting.stopCameraPreview();
trtcMeeting.startScreenCapture(
    videoFps: 10,
    videoBitrate: 1600,
    videoResolution: TRTCCloudDef.TRTC_VIDEO_RESOLUTION_1280_720,
    videoResolutionMode: TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT,
    appGroup: iosAppGroup,
);
```

ステップ8：文字チャットとミュートメッセージの実現

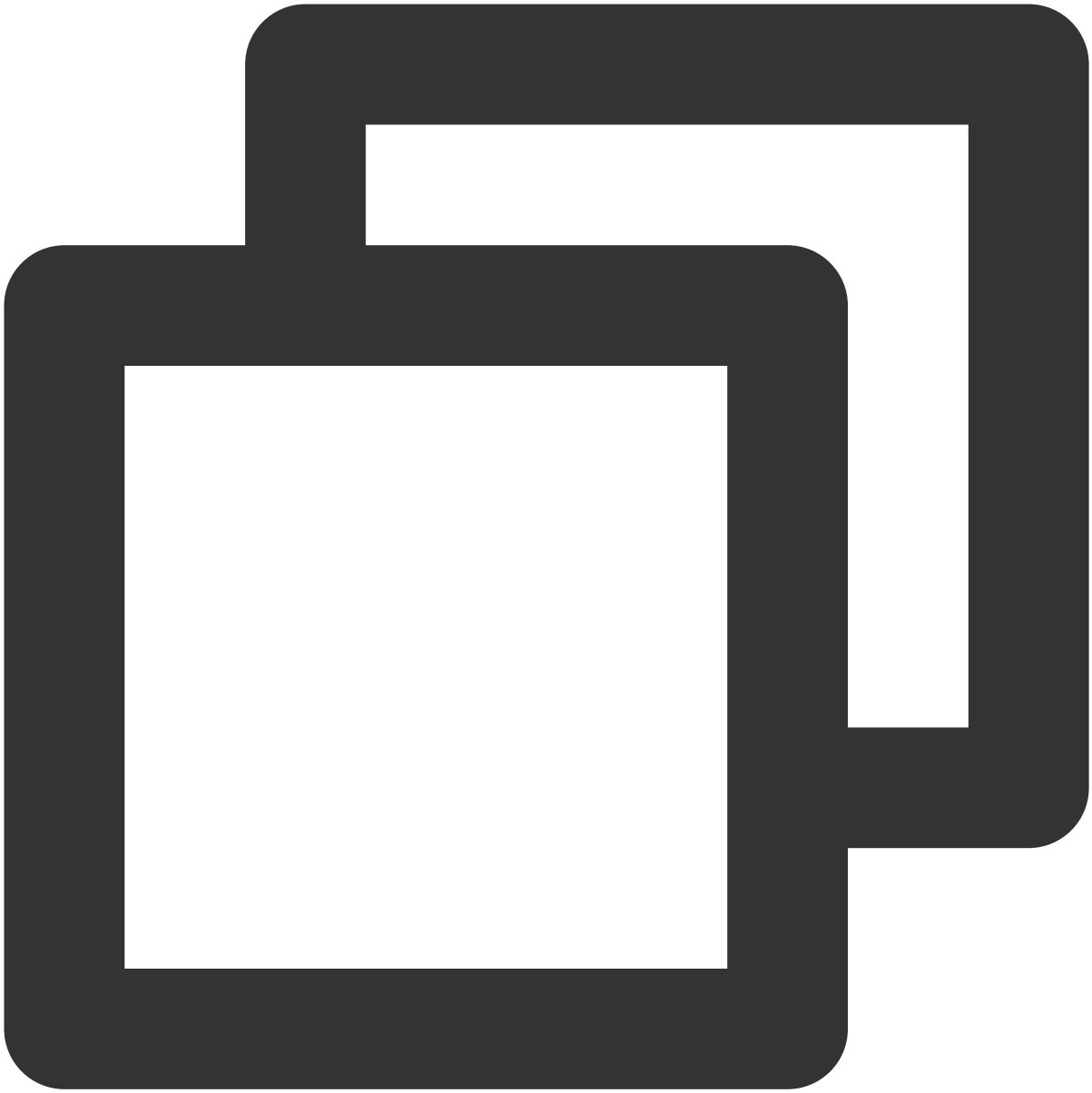
sendRoomTextMsgにより、ノーマルなテキストメッセージを送信でき、そのミーティングのすべての参加者がonRecvRoomTextMsgのコールバックを受信します。IMバックエンドにはデフォルトのセンシティブワードフィルタリングルールが備わっており、センシティブワードと判定されたテキストメッセージがクラウドで転送されることはありません。



```
// 発信側：テキストメッセージの発信
trtcMeeting.sendRoomTextMsg('Hello Word!');
// 受信側：テキストメッセージのモニタリング
trtcMeeting.registerListener(onListener);
```

```
onListener(TRTCMeetingDelegate type, param) {
    switch (type) {
        case TRTCMeetingDelegate.onRecvRoomTextMsg:
            print( param['userName'] + 'からのメッセージ：' + param['message']+'を受信'
            break;
    }
}
```

sendRoomCustomMsgにより、カスタム（シグナリング）メッセージを送信でき、そのミーティングのすべての参加者がonRecvRoomCustomMsgのコールバックを受信します。カスタムメッセージは、通常カスタマイズしたシグナリングの転送に使用されます（例：ミュートの会場内制御など）。



```
// 送信側：カスタマイズしたcmdでミュートの通知を区別できます
// eg: "CMD_MUTE_AUDIO"はミュートの通知を表します
trtcMeeting.sendRoomCustomMsg('CMD_MUTE_AUDIO', '1');
// 受信側：カスタムメッセージのモニタリング
trtcMeeting.registerListener(onListener);
onListener(TRTCMeetingDelegate type, param) {
    switch (type) {
        case TRTCMeetingDelegate.onRecvRoomCustomMsg:
            if (param['command'] == 'CMD_MUTE_AUDIO') {
                // ミュートの通知を受信します
                print(param['userName'] +'から' + 'のミュートの通知：' + param['message
```

```
                trtcMeeting.muteLocalAudio(message == '1');
            }
            break;
        }
    }
```

TUIRoom (Windows&Mac)の統合

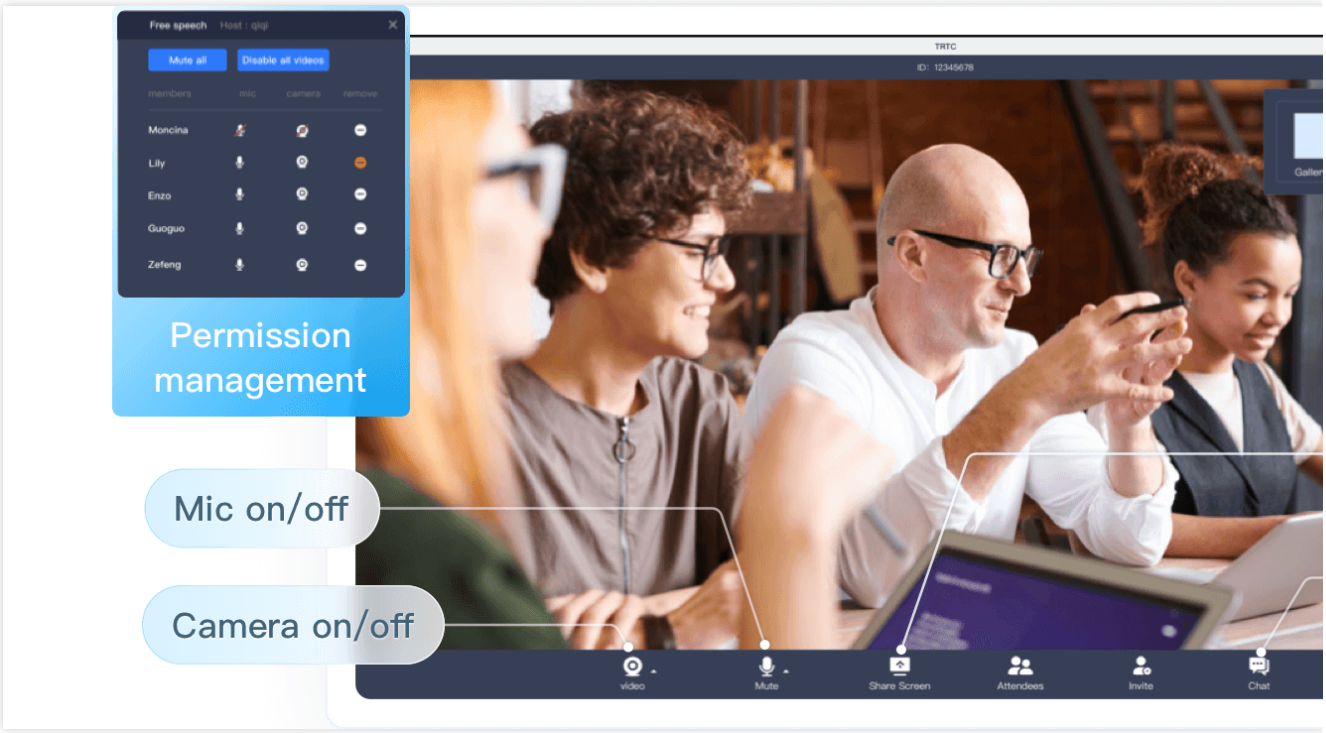
最終更新日：：2024-07-19 15:35:35

ここで紹介するPC版TUIRoomコンポーネントは、柔軟なレイアウトで、適用性の高いオーディオビデオコミュニケーションツールです。コラボレーションオフィス、リモート採用、リモート問診、保険金査定、オンラインカスタマーサービス、ビデオ面接、デジタル行政、金融のデジタル化、オンラインミーティング、eラーニングなどのシーンで使用することができます。 各業界のシナリオと深く結びつき、企業のコスト削減と効率アップを助け、デジタル化のモデルチェンジを加速し、競争力を向上させます。
WindowsまたはMacプラットフォームのAppをダウンロード、インストールして体験できます。

説明

TUIKitシリーズコンポーネントはTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期的にアクティブにすることができます。IMサービスの課金ルールの詳細については、Instant Messagingの料金説明をご参照ください。TRTCをアクティブにすると、デフォルトでは、100DAUまでサポートするIM SDK体験版もアクティブになります。

デモンストレーション



ソリューションの優位性

超低遅延のオーディオビデオ通話、チャットルーム、画面共有、美顔、デバイス検出、データ統計などの機能を統合し、多人数オーディオビデオルームの一般的な機能をカバーします。
二次開発の需要に応じて、カスタムUIやレイアウトを迅速に実装し、ビジネスの迅速な立ち上げに役立ちます。
TRTCとIMの基本SDKをカプセル化し、基本的なロジックコントロールを実装し、呼び出しを容易にするインターフェースを提供します。

統合ガイド

多人数オーディオビデオルーム機能をスピーディーに統合するため、ここでは2種類の統合方法をお勧めしています。ご自身に合った方法を選択して二次開発にお役立てください。
外部プロセス起動
カスタムUIの実装

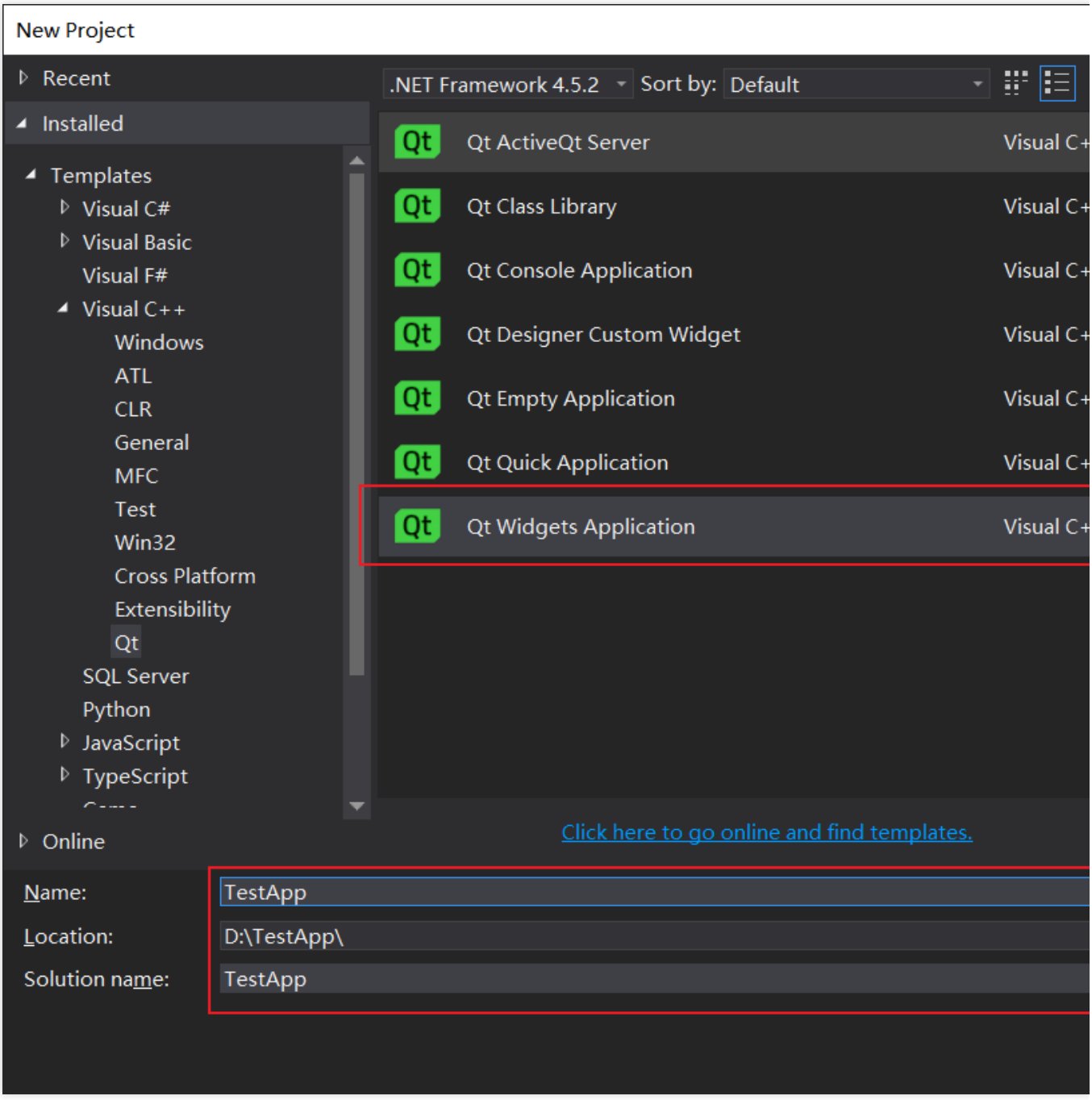
環境の準備

Windows環境 ：
Visual Studio 2015以上の統合開発環境。
QT5.9.1以上のバージョンのQT開発データベース。
VS下のQT開発プラグインQt Visual Studio Tools 2.2.0以上。
サポートする最低システム要件：Windows 8。
正常な開発が行える統合開発環境であることを確実にしてください。
Mac環境：
QT5.9.1およびそれ以上のバージョンのQT開発データベース。
QtCreator統合開発環境。 QTのインストール時に、QtCreatorの同時インストールを選択してください。バージョンはQTの公式インストールパッケージに従います。
QtCreator統合開発環境で正常な開発が行えることを確実にしてください。

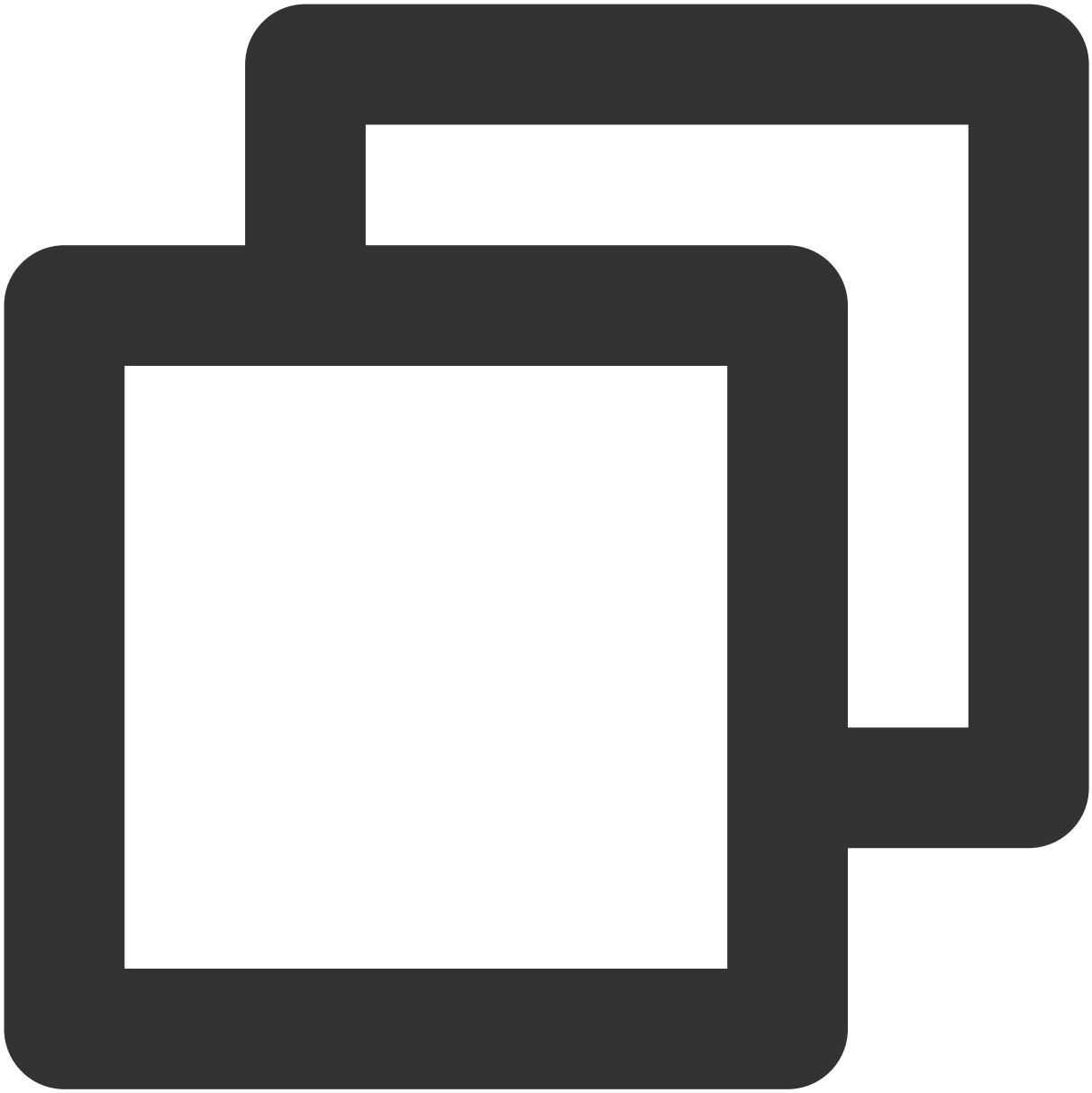
外部プロセス起動

1. **RoomAppプログラムのコンパイル**
外部プロセスによるRoomApp起動方式を使用する場合は、元のRoomAppの実行プログラムに依存するため、事前のコンパイルが必要です。
クリックしてRoomAppに進み、ソースコードをCloneし、プロジェクトを設定し、RoomAppをコンパイルして生成できます。
2. **TestAppプロジェクトの新規作成**
Windowsプラットフォーム
Macプラットフォーム

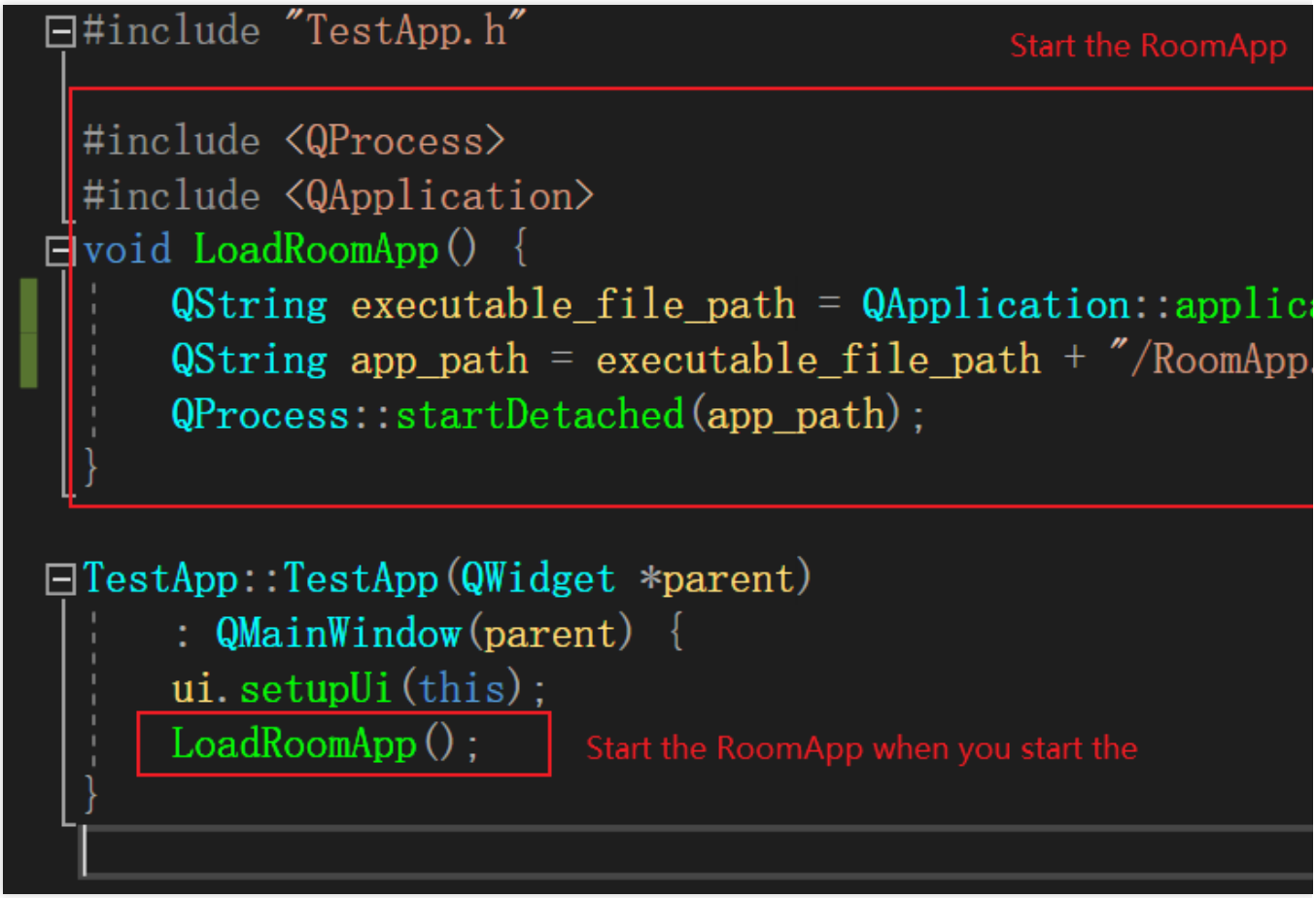
1. VSを開き、Qt Widgets Applicationプロジェクトタイプを選択し、TestAppプロジェクトを作成します。



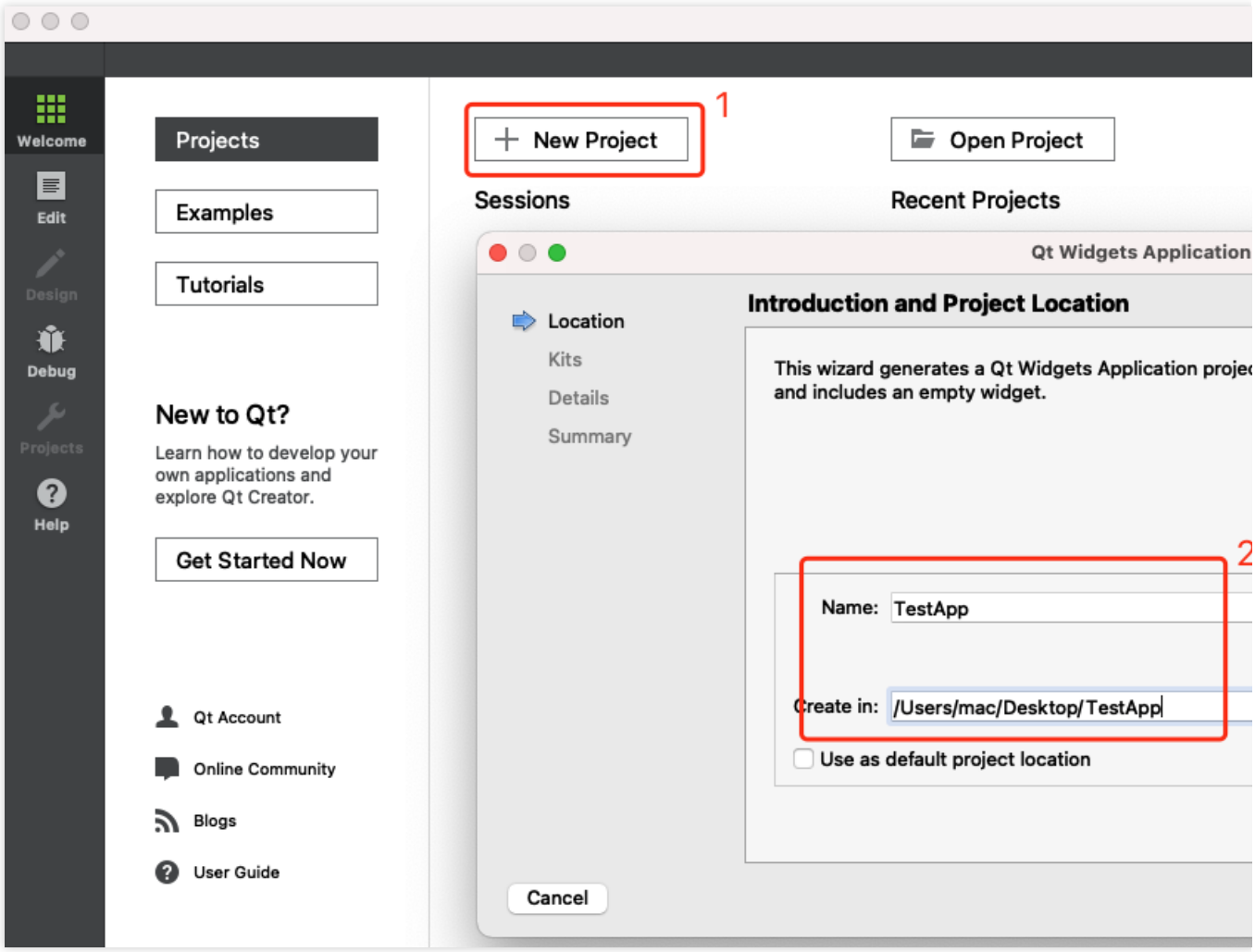
2. プロセスを起動するプログラムを作成し、適切な位置にLoadRoomApp関数を呼び出します。



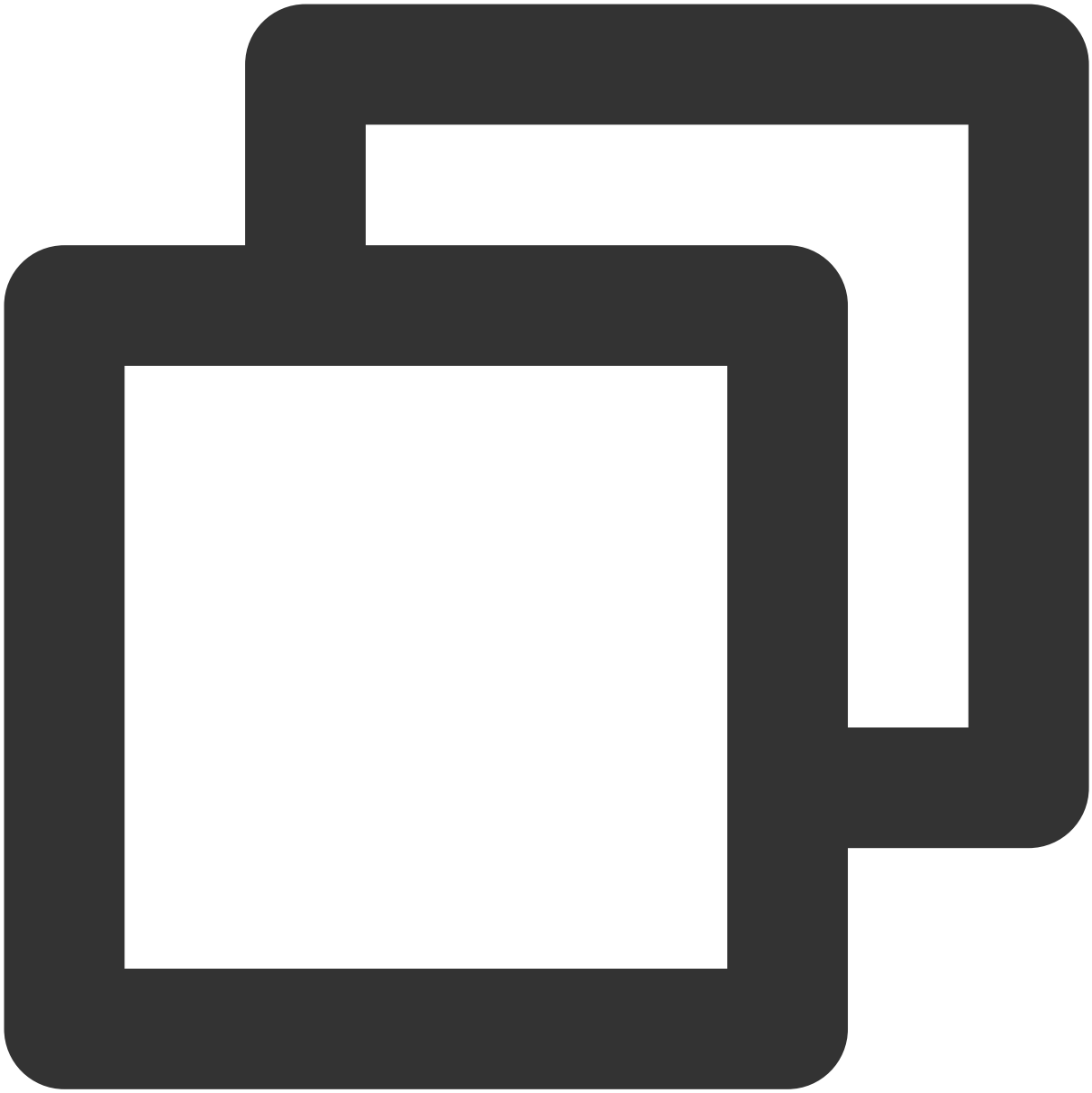
```
#include <QProcess>
#include <QApplication>
void LoadRoomApp() {
 QString executable_file_path = QApplication::applicationDirPath();
 QString app_path = executable_file_path + "/RoomApp.exe";
 QProcess::startDetached(app_path);
}
```



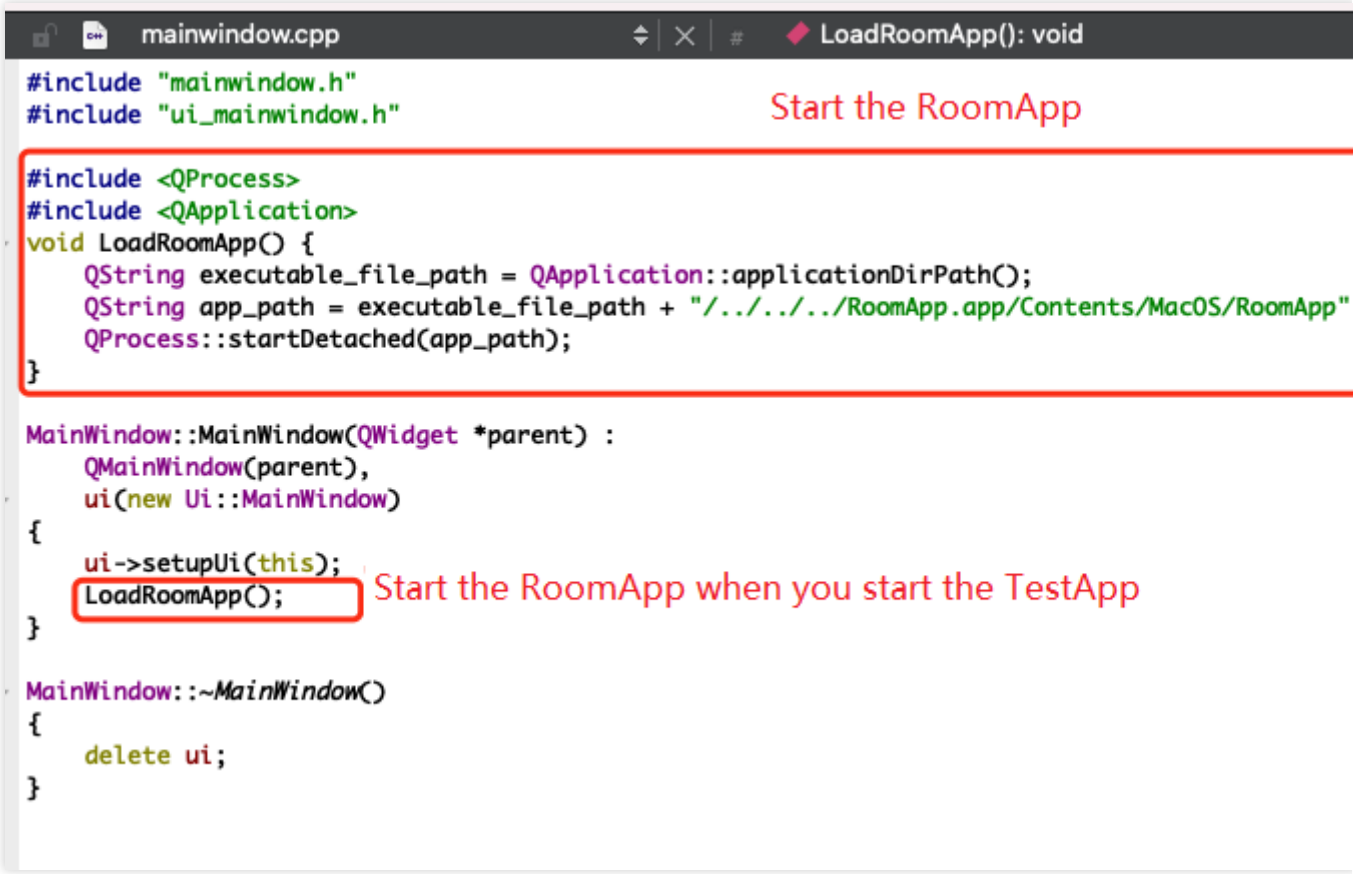
3. プロジェクトをコンパイルし、RoomAppをコンパイルした成果物を現在実行可能なプログラムディレクトリにコピーします。release x86プログラムを例にとると、次のようになります：
`TUIRoom\\Windows-Mac\\RoomApp\\bin\\Win32\\Release` ディレクトリ下のすべてのファイルを現在のプログラムディレクトリ下にコピーします。
4. プログラムを実行します。TestAppの起動と同時にRoomAppを起動します。
1. QtCreatorを開き、Qt Widgets Applicationプロジェクトを選択して新規作成し、TestAppプロジェクトを作成します。



2. プロセスを起動するプログラムを作成し、適切な位置にLoadRoomApp関数を呼び出します。



```
#include <QProcess>
#include <QApplication>
void LoadRoomApp() {
 QString executable_file_path = QApplication::applicationDirPath();
 QString app_path = executable_file_path + "/../../../RoomApp.app/Contents/MacOS/Ro
 QProcess::startDetached(app_path);
}
```



3. プロジェクトをコンパイルし、RoomAppをコンパイルした生成物 `RoomApp.app` を現在のプロジェクトの生成物と同一階層のディレクトリ下にコピーします。上の図で作成したプロジェクトのパスを例にとると、次のようになります：
`RoomApp.app` を `/Users/mac/Desktop/TestApp/build-TestApp-Desktop_Qt_5_9_1_clang_64bit-Release` ディレクトリ下にコピーします。
4. プログラムを実行します。TestAppの起動と同時にRoomAppを起動します。

カスタムUIの実装

当社が提供するAppで直接変更して適用するか、あるいはAppのソースコードのModuleを使用してカスタムUIを実装することができます
ソースコードのModuleには、TRTC SDKおよびIM SDKのカプセル化が含まれています。 `TUIRoomCore.h` 、 `TUIRoomCoreCallback.h` 、 `TUIRoomDef.h` などのファイルでこのモジュールが提供するインターフェース関数およびその他定義を確認し、対応するインターフェースを使用してカスタムUIを実装できます
Appディレクトリには UIに関連する設計とロジックが含まれており、必要に応じてRoomAppソースコードを変更し、二次開発を行うことができます。主な機能ポイントは次のとおりです：

機能ポイント	ファイルディレクトリ
トップページログイン	Windows-Mac\\RoomApp\\App\\LoginViewController.cpp

デバイス検査	Windows-Mac\\RoomApp\\App\\PresetDeviceController.cpp
ホームページ	Windows-Mac\\RoomApp\\App\\MainWindow.cpp
配信側リスト	Windows-Mac\\RoomApp\\App\\StageListController.cpp
メンバーリスト	Windows-Mac\\RoomApp\\App\\MemberListViewController.cpp
設定ページ	Windows-Mac\\RoomApp\\App\\SettingViewController.cpp
チャットルーム	Windows-Mac\\RoomApp\\App\\ChatRoomViewController.cpp
画面共有	Windows-Mac\\RoomApp\\App\\ScreenShareWindow.cpp
下部ツールバー	Windows-Mac\\RoomApp\\App\\BottomBarController.cpp

よくあるご質問

ご要望やフィードバックなどがございましたら、colleenyu@tencent.comまでご連絡ください。

TUIRoom (Electron)の統合

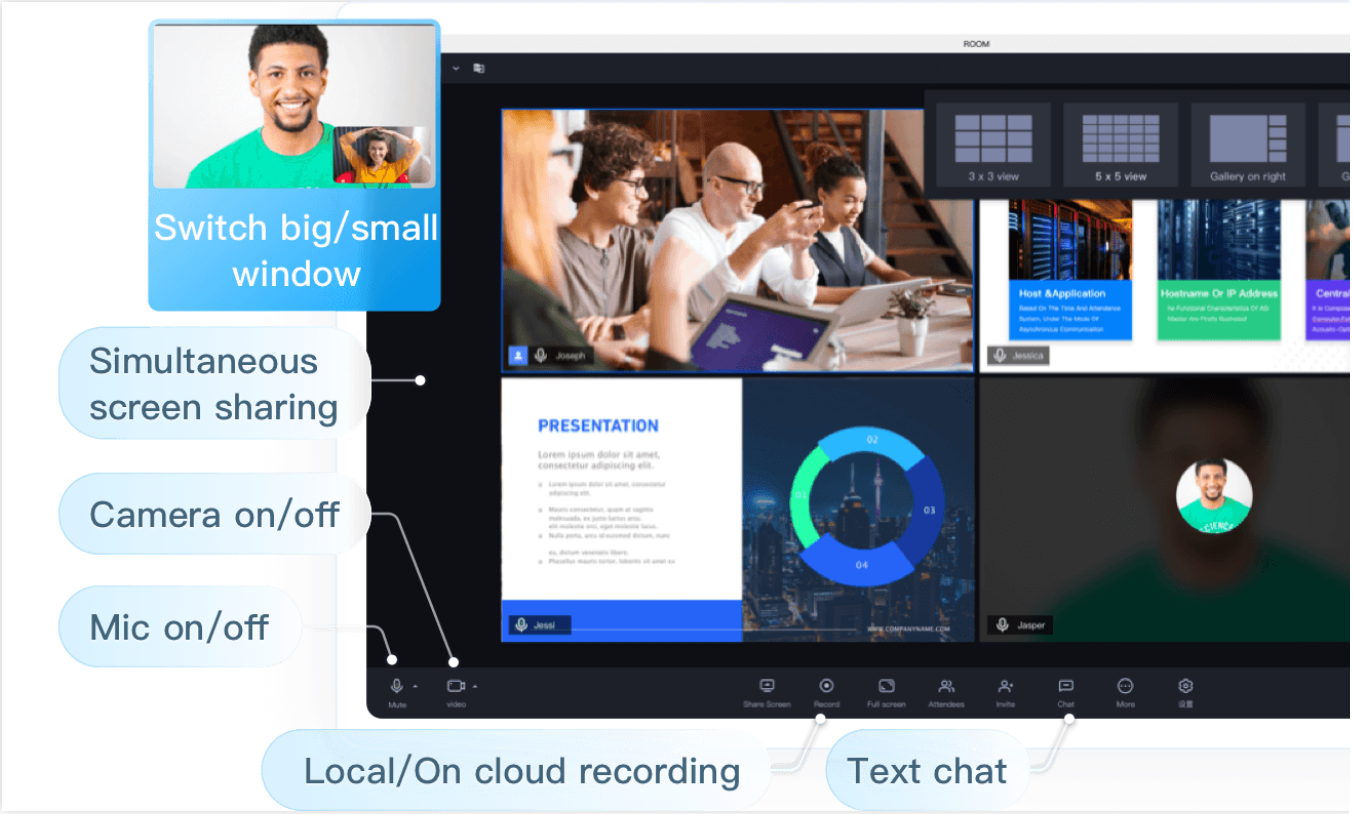
最終更新日：：2024-07-19 15:35:35

コンポーネントの説明

TUIRoomは、UIを含むオープンソースオーディオビデオコンポーネントです。TUIRoomを統合することにより、ビジネスにおいて、オーディオビデオルーム、画面共有、チャットなどの機能を速やかにオンラインにすることができます。Electron端末のTUIRoomの基本機能は下図のとおりです。

説明：

TUIKitシリーズコンポーネントはTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期してアクティブ化することができます。 IMサービスの課金ルールの詳細については、Instant Messagingの料金説明をご参照ください。TRTCをアクティブ化すると、関連するIM SDKの体験版がデフォルトでアクティブ化されます。これは100 DAUのみをサポートします。



オンライン体験リンクMac OS版およびWindows版をクリックしてダウンロードすると、TUIRoom Electronのより多くの機能を体験することができます。

GithubをクリックしてTUIRoomのコードをダウンロードし、TUIRoom Electronデモプロジェクトクイックスタートのドキュメントを参照し、TUIRoom Electronデモプロジェクトをクイックスタートすることもできます。

現在の業務でElectron端末のTUIRoomコンポーネントを統合する必要がある場合は、こちらのドキュメントをご参照ください。

コンポーネントの統合

TUIRoomコンポーネントはVue3 + TS + Pinia + Element Plus + SCSSを使用して開発されています。プロジェクトの統合には、Electron + Vue3 + TSの技術スタックを使用する必要があります。

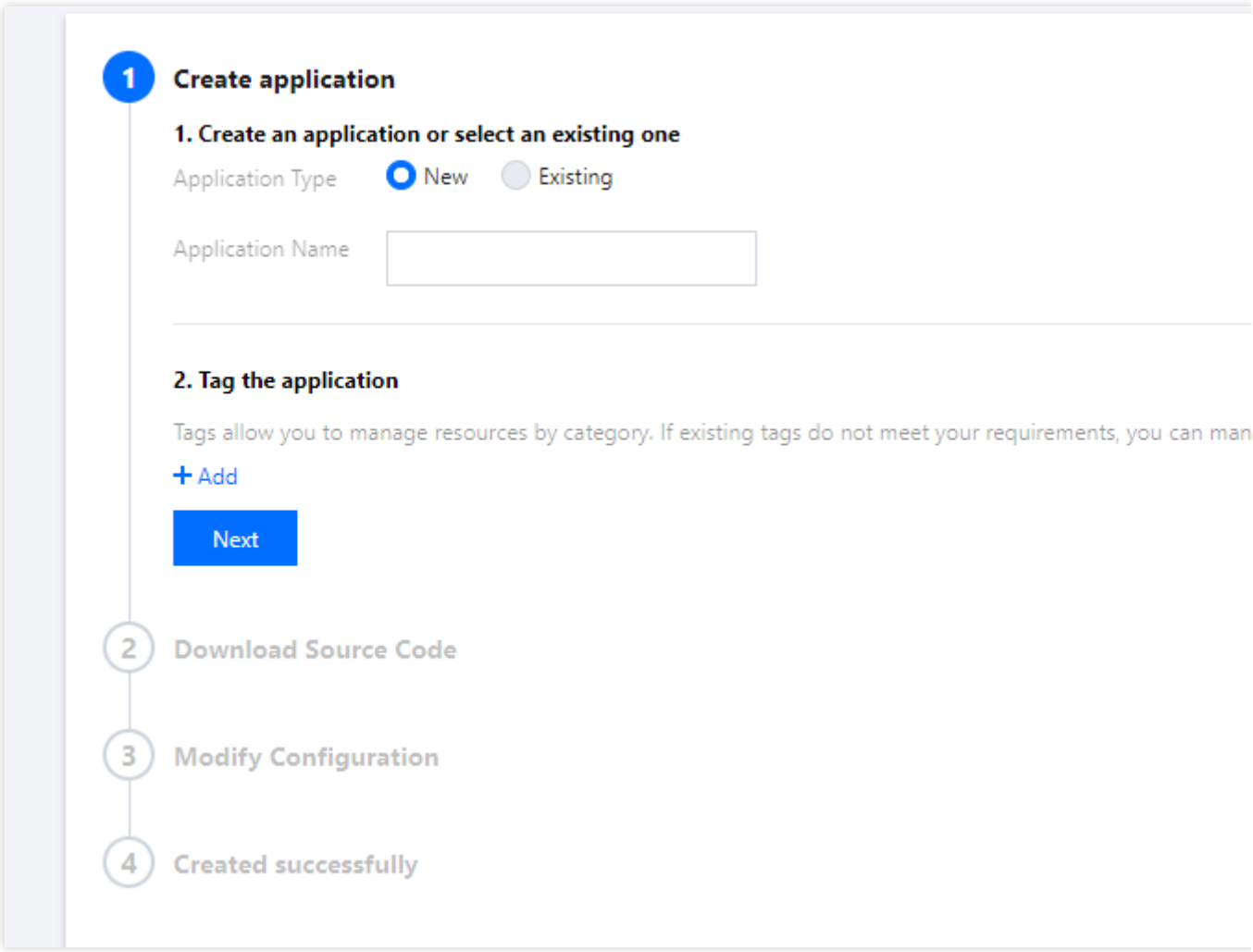
ステップ1： Tencent Cloud TRTCおよびIMサービスのアクティブ化

TUIRoomは、Tencent Cloud TRTCとIMサービスをベースに開発されています。

1. **TRTCアプリケーションの作成**

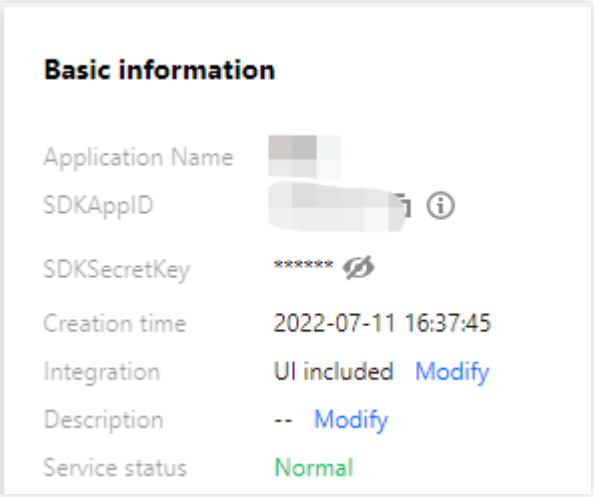
Tencent Cloudアカウントをまだお持ちでない場合は、Tencent Cloudアカウントの登録を行い、実名認証を完了してください。

TRTCコンソール で、**アプリケーション管理>アプリケーションの作成** をクリックし、新たなアプリケーションを作成します。

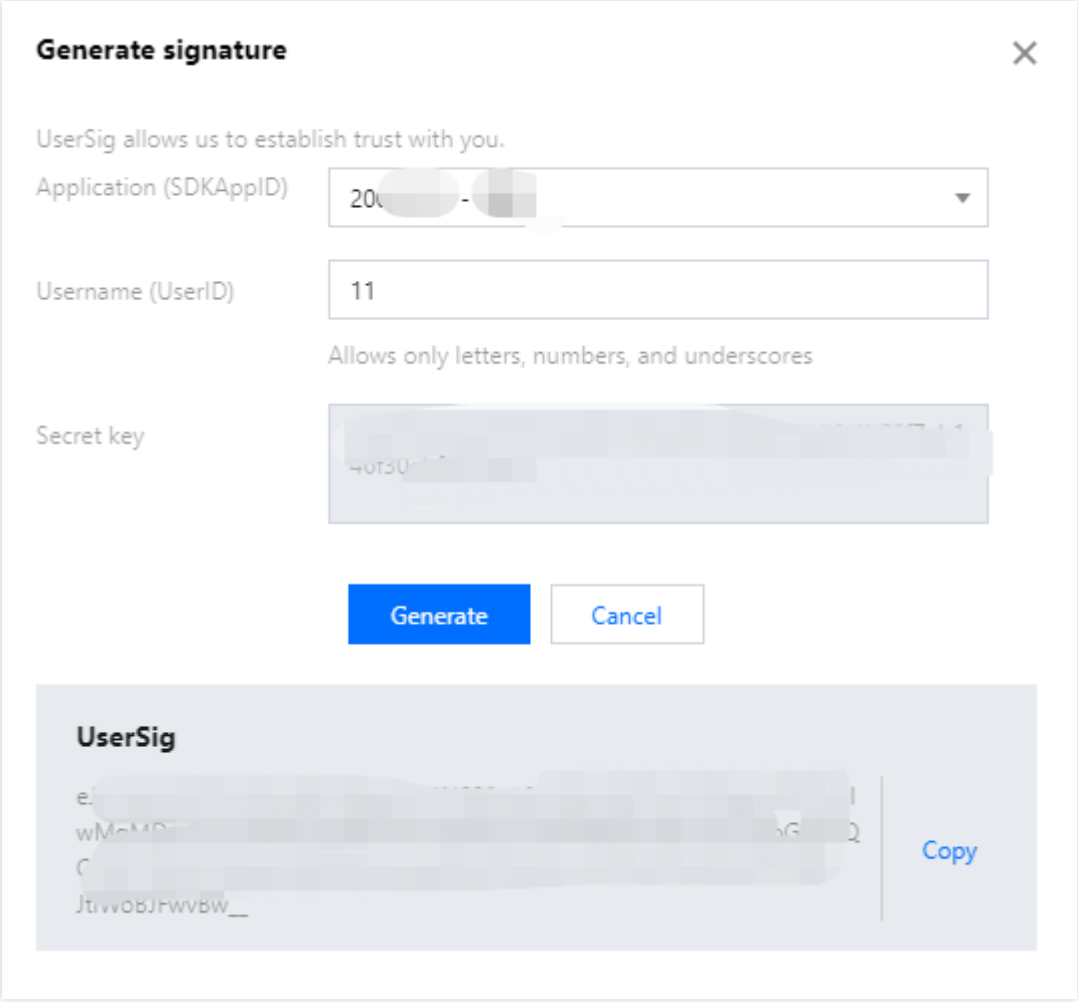


2. **TRTCアプリケーションおよびキー情報の取得**

アプリケーション管理 > アプリケーション情報でSDKAppID の情報を取得します。SDKAppIDはTRTCのアプリケーションIDです。業務の分離、すなわち異なるSDKAppIDの通話が互いに接続できないようにするために用いられます。



アプリケーション管理 > クイックマスターでアプリケーションのsecretKey情報を取得します。SecretKeyはTRTCのアプリケーションキーです。SDKAppIDとペアで使用する必要があり、TRTCサービスを正当に使用するための認証用証明書UserSigの発行に用いられます。



3. **UserSigの発行**

UserSigとは、悪意ある攻撃者によるクラウドサービスの使用権の盗用を防ぐために、Tencent Cloudによって設計された、セキュリティ保護された署名です。TUIRoomの初期化にはUserSigパラメータの提供が必要です。

デバッグスタート段階でuserSigを発行する方法については、デバッグスタート段階でのUserSigの計算方法をご参照ください。

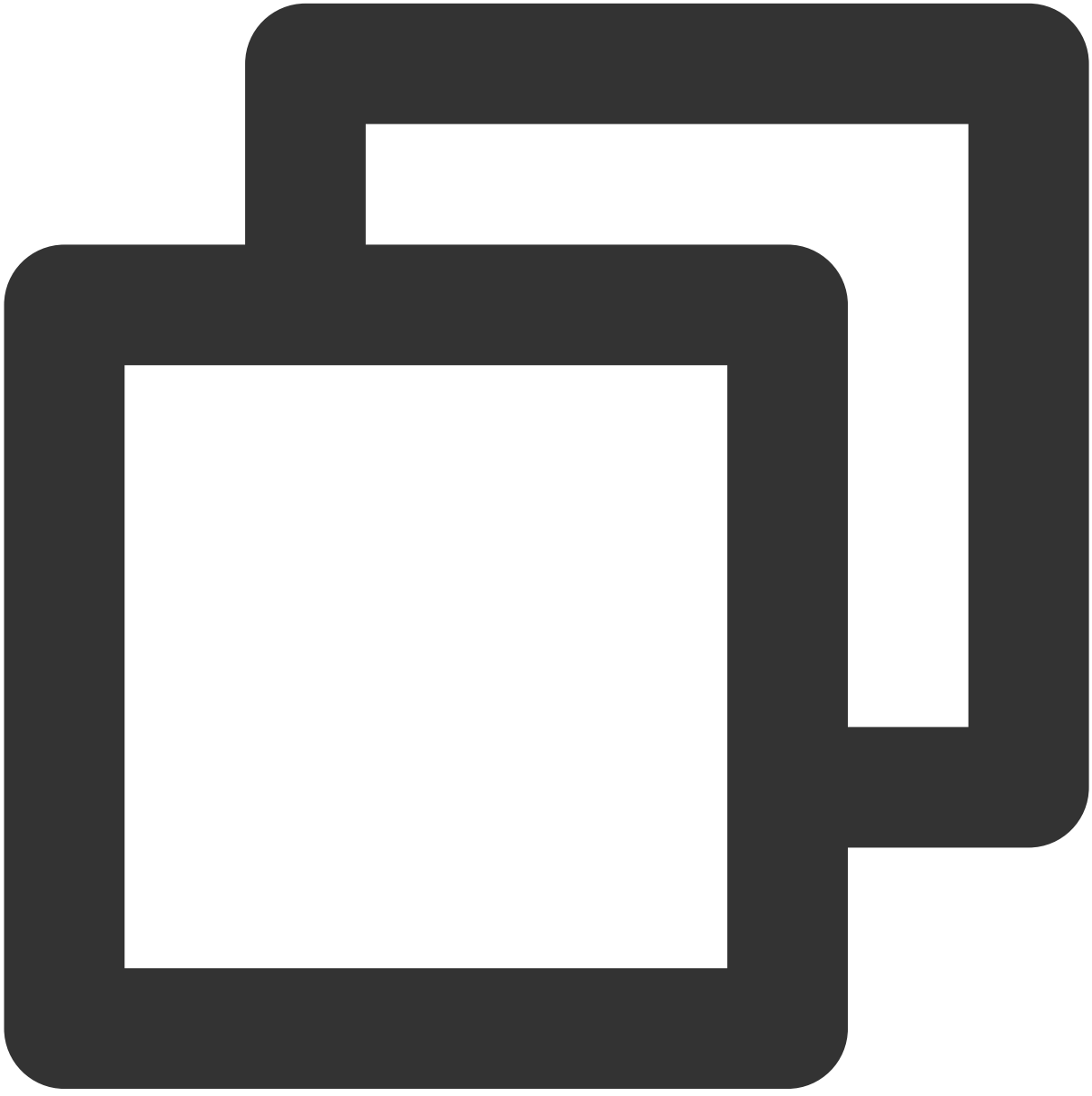
本番環境でuserSigを発行する方法については、本番環境でのUserSigの計算方法をご参照ください。

ステップ2：TUIRoomコンポーネントのダウンロードとコピー

1. 業務側にすでにあるElectron + Vue3 + TSプロジェクトを開きます。Electron + Vue3 + TSプロジェクトがない場合は、このテンプレートGithubによってElectron + Vue3 + TSのテンプレートプロジェクトを生成することができます。

ご注意

このドキュメントで説明する統合手順はelectron-vite-vueテンプレートプロジェクトのバージョン1.0.0をベースにしています。
electron-vite-vueテンプレートプロジェクトの最新バージョンではディレクトリ構造が調整されています。最新バージョンを使用される場合は、このドキュメントを参照してご自身でディレクトリの調整と設定を行うことができます。
2. テンプレートプロジェクトの生成に成功後、次のスクリプトを実行します。

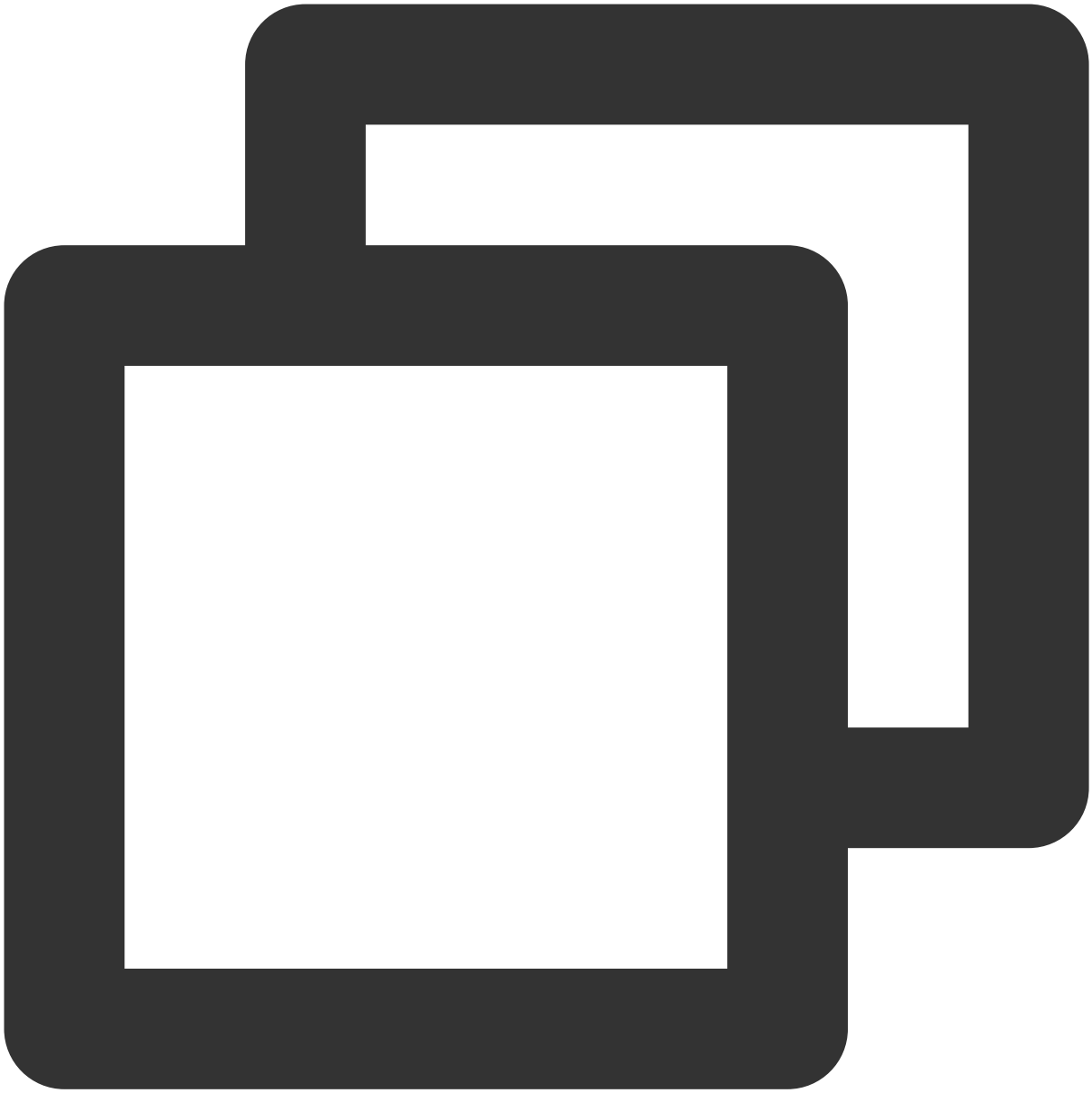


```
cd electron-vite-vue
npm install
npm run dev
```

3. Githubをクリックし、TUIRoomリポジトリコードをクローンまたはダウンロードし、`TUIRoom/Electron/packages/renderer/src/TUIRoom` フォルダを既存のプロジェクトの `packages/renderer/src/` ディレクトリ下にコピーします。

ステップ3：TUIRoomコンポーネントのインポート

1. ページ内にTUIRoomコンポーネントをインポートします。例えば、`App.vue` コンポーネントにTUIRoomコンポーネントをインポートします。
TUIRoomコンポーネントはユーザーをキャスターロールと一般メンバーロールに区分します。コンポーネントは外部に対しinit、createRoom、enterRoomのメソッドを提供します。
キャスターと一般メンバーはinitメソッドによってTUIRoomコンポーネントに対し、アプリケーションとユーザーデータの初期化を行うことができます。キャスターはcreateRoom メソッドによってルームを作成し入室することができます。一般メンバーはenterRoomメソッドによって、キャスターが作成したルームに入室することができます。



```
<template>
 <room ref="TUIRoomRef"></room>
</template>

<script setup lang="ts">
 import { ref, onMounted } from 'vue';
 // TUIRoomコンポーネントをインポートします。インポートパスが正しいかどうかよく確認してください
 import Room from './TUIRoom/index.vue';
 // TUIRoomコンポーネント要素を取得し、TUIRoomコンポーネントの呼び出しメソッドに用います
 const TUIRoomRef = ref();
```

```
onMounted(async () => {
// TUIRoomコンポーネントを初期化します
// キャスターはルームを作成する前にTUIRoomコンポーネントを初期化する必要があります
// 一般メンバーは入室前にTUIRoomコンポーネントを初期化する必要があります
await TUIRoomRef.value.init({
    // sdkAppIdを取得します。ステップ1をご参照ください
    sdkAppId: 0,
    業務// ユーザーの業務における固有の表示IDです
    userId: '',
    // ローカル開発デバッグの場合はhttps://console.intl.cloud.tencent.com/trtc/usersig
    userSig: '',
    // ユーザーが業務で使用するニックネーム
    userName: '',
    // ユーザーが業務で使用するプロフィール画像リンク
    userAvatar: '',
    // ユーザーが画面共有に用いる固有のIdであり、shareUserId = `share_${userId}`である必要
    shareUserId: '',
    // 本文ステップ1 > 第3段階を参照し、sdkAppIdとshareUserIdを使用してshareUserSigを発行
    shareUserSig: '',
})
 // デフォルトではルーム作成が実行されます。実際の統合では必要に応じ、タイミングを見てhandleCre
await handleCreateRoom();
})

// キャスターがルームを作成します。このメソッドはルーム作成時にのみ呼び出します
async function handleCreateRoom() {
// roomIdはユーザーが入室するルームナンバーです。roomIdのタイプはnumberである必要があります
// roomModeには'FreeSpeech'(自由発言モード) と'ApplySpeech'(挙手発言モード)の2種類があり
// roomParamはユーザーの入室時のデフォルト動作(デフォルトでマイクをオンにするかどうか、デフォル
const roomId = 123456;
const roomMode = 'FreeSpeech';
const roomParam = {
    isOpenCamera: true,
    isOpenMicrophone: true,
}
await TUIRoomRef.value.createRoom(roomId, roomMode, roomParam);
}

// 一般メンバーが入室します。このメソッドは一般メンバーが作成済みのルームに入室する際にのみ呼び出し
async function handleEnterRoom() {
// roomIdはユーザーが入室するルームナンバーです。roomIdのタイプはnumberである必要があります
// roomParamはユーザーの入室時のデフォルト動作(デフォルトでマイクをオンにするかどうか、デフォル
const roomId = 123456;
const roomParam = {
    isOpenCamera: true,
    isOpenMicrophone: true,
}
```

```
   await TUIRoomRef.value.enterRoom(roomId, roomParam);
  }
</script>

<style>
html, body {
 width: 100%;
 height: 100%;
 margin: 0;
}

#app {
 width: 100%;
 height: 100%;
}
</style>
```

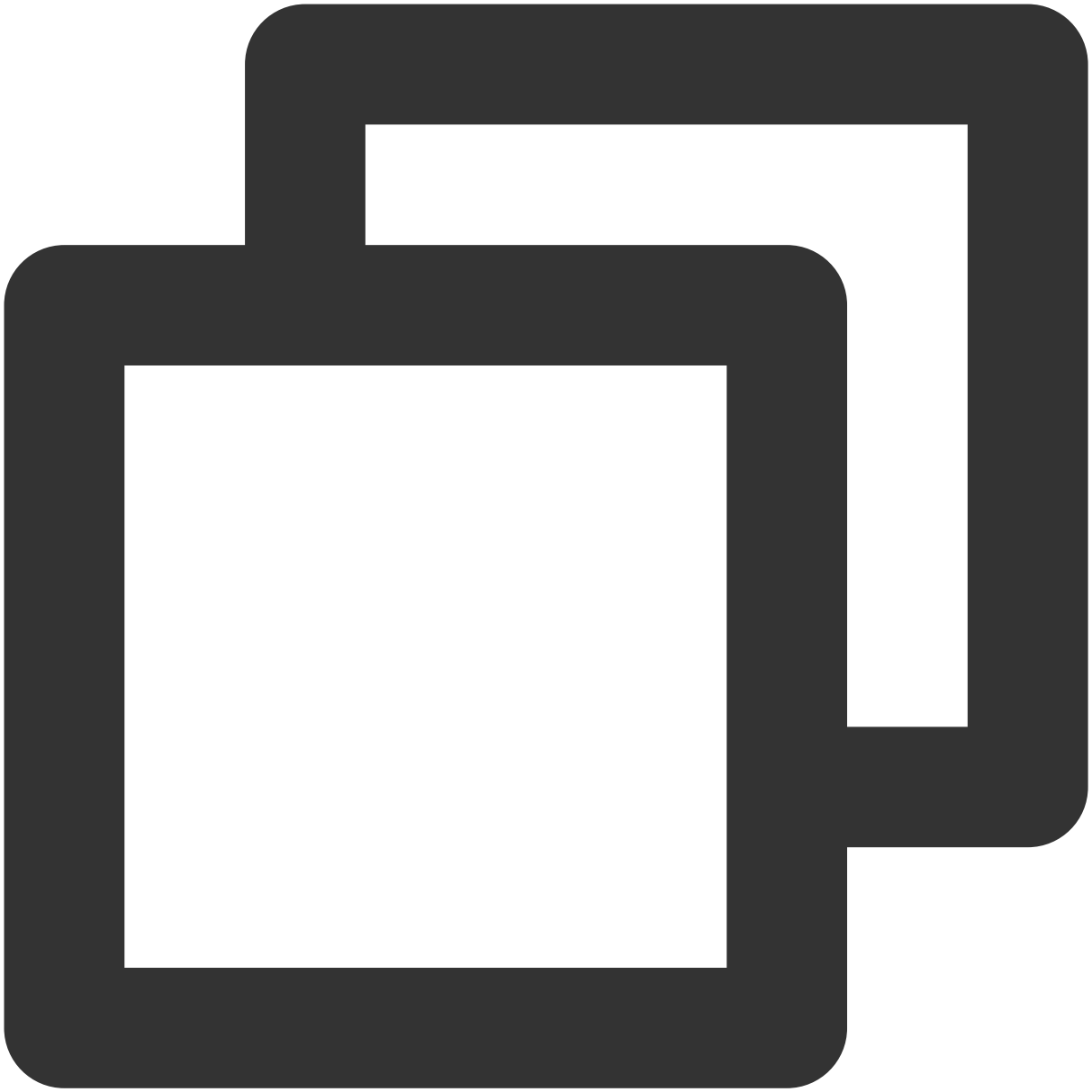
ご注意：
ページ内で上記のコードをコピーした後、TUIRoomインターフェースのパラメータを実際のデータに変更する必要があります。

ステップ4：本番環境の設定

TUIRoomコンポーネントをインポートした後、プロジェクトが正常に実行されることを確認するため、次の設定を行います。

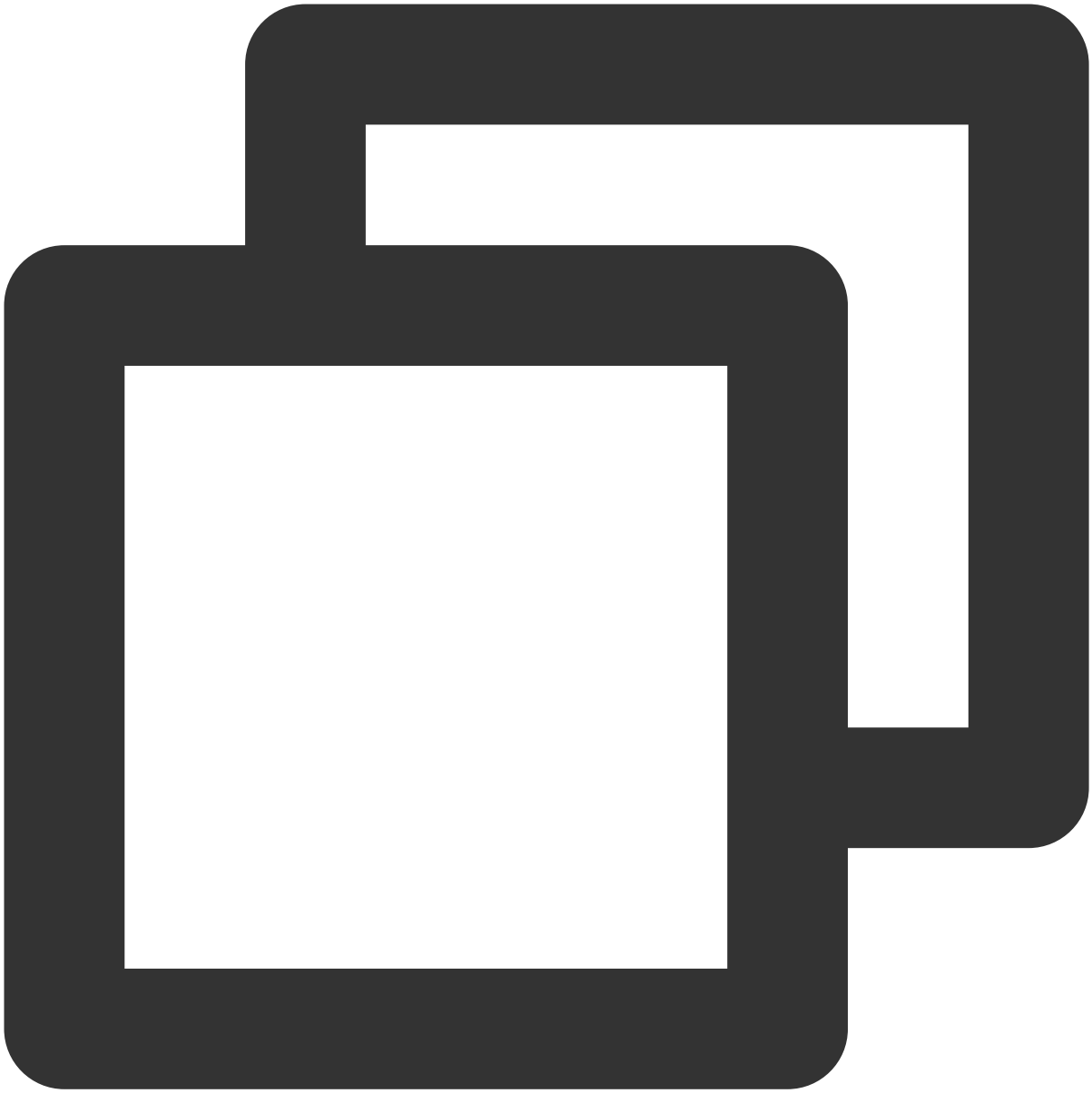
1. **依存のインストール**

開発環境の依存をインストールします。



```
npm install sass typescript unplugin-auto-import unplugin-vue-components -S -D
```

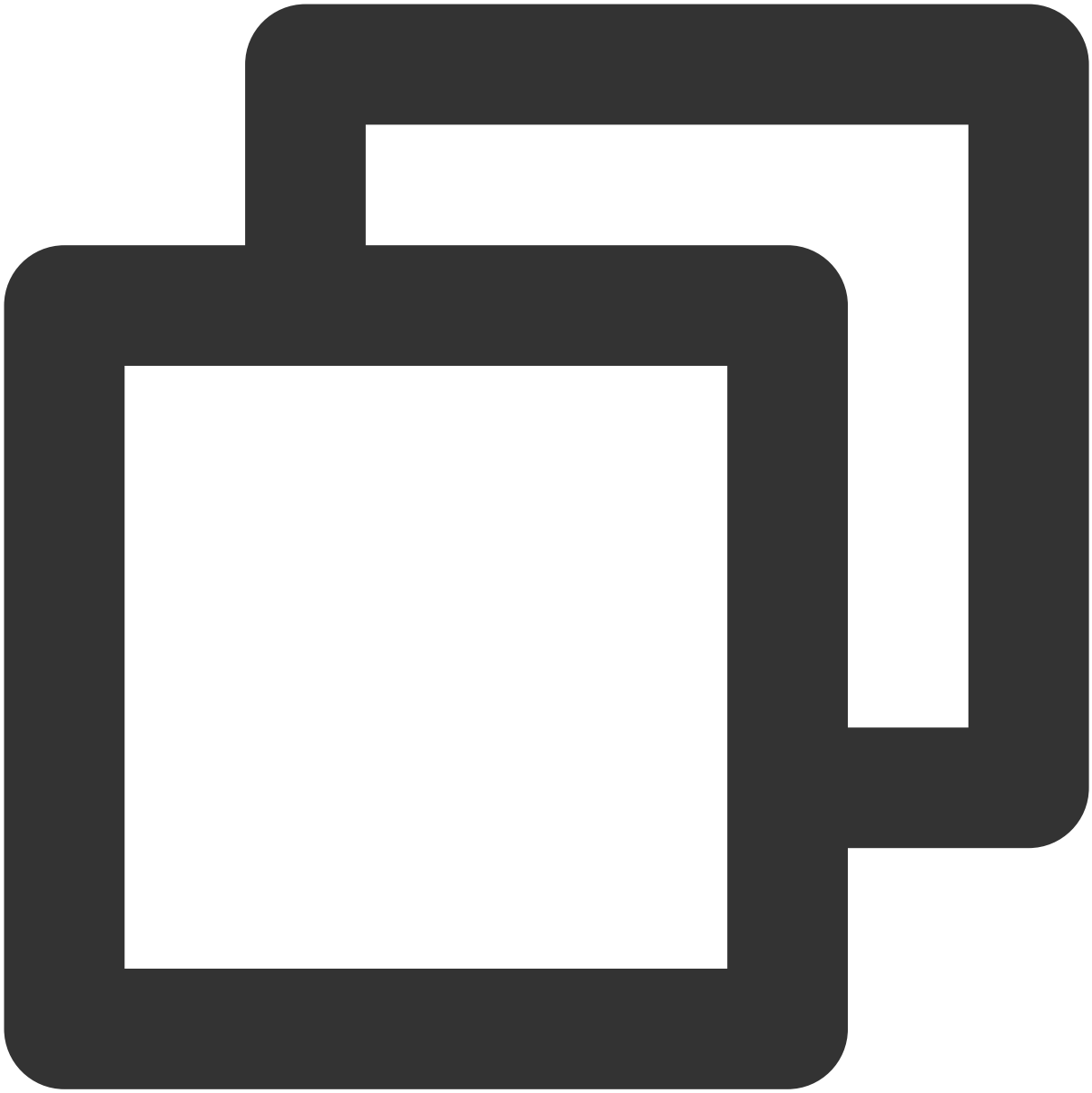
本番環境の依存をインストールします。



```
npm install element-plus events mitt pinia trtc-electron-sdk tim-js-sdk tsignaling
```

2. **Piniaの登録**

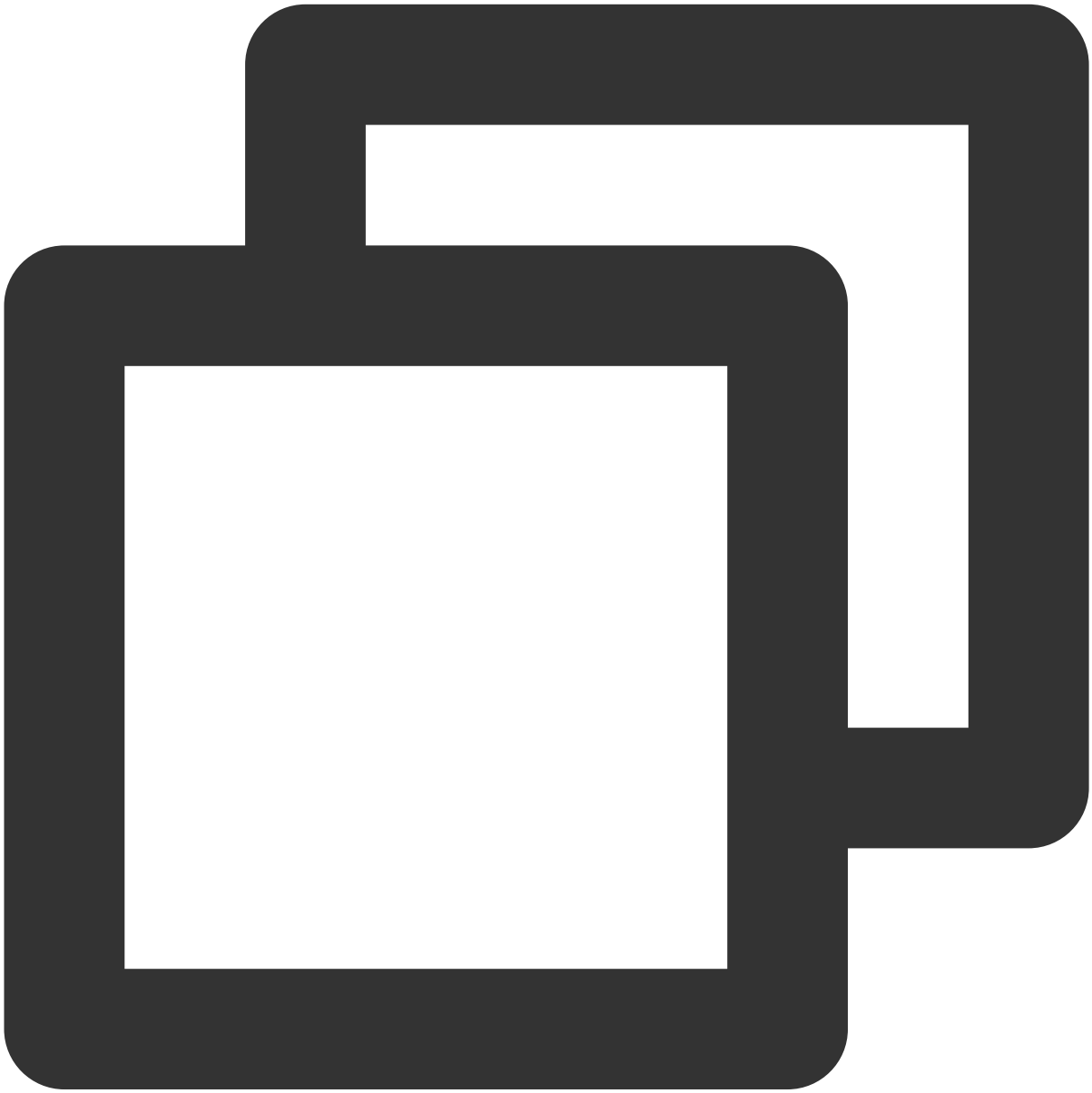
TUIRoomはPiniaを使用してルームデータの管理を行うため、プロジェクトのエントリーファイルでPiniaの登録を行う必要があります。プロジェクトのエントリーファイルは `packages/renderer/src/main.ts` ファイルです。



```
// src/main.tsファイル
import { createPinia } from 'pinia';

const app = createApp(App);
// Piniaの登録
createApp(App)
  .use(createPinia())
  .mount('#app')
  .$nextTick(window.removeLoading)
```

3. **element-plusを必要に応じてインポートするよう設定**

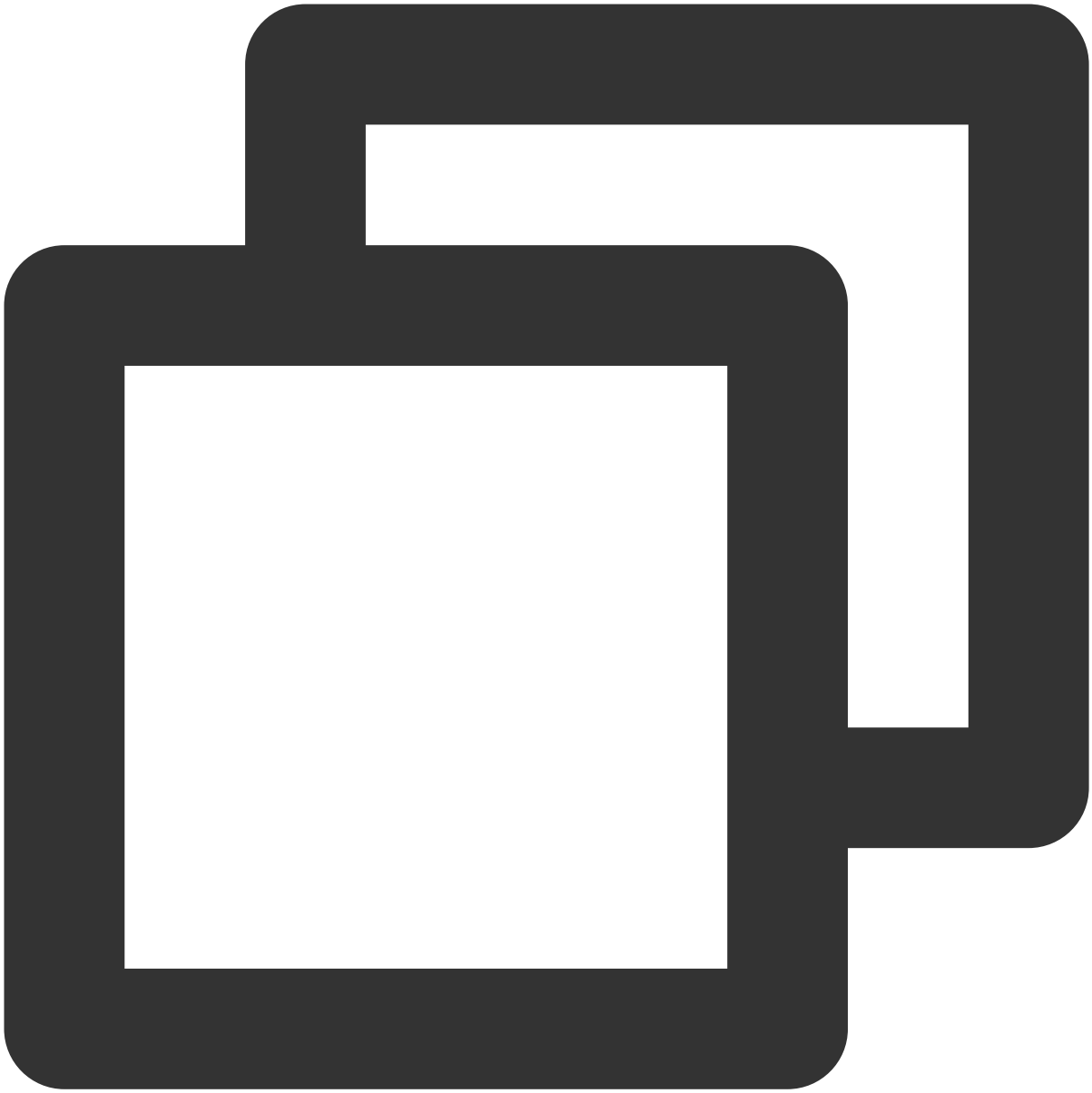
TUIRoomはelement-plus UIコンポーネントを使用します。すべてのelement-plusコンポーネントをインポートしてしまわないように、 `packages/renderer/vite.config.ts` で、element-plusコンポーネントを必要に応じてロードするように設定しておく必要があります。
ご注意：
以下の設定項目は増分設定です。すでに存在するVite設定項目を削除しないでください。



```
// vite.config.ts
import AutoImport from 'unplugin-auto-import/vite';
import Components from 'unplugin-vue-components/vite';
import { ElementPlusResolver } from 'unplugin-vue-components/resolvers';
```

```
const path = require('path');

export default defineConfig({
    // ...
    plugins: [
        AutoImport({
            resolvers: [ElementPlusResolver()],
        }),
        Components({
            resolvers: [ElementPlusResolver({
                importStyle: 'sass',
            })],
        }),
    ],
    css: {
    preprocessorOptions: {
    scss: {
        additionalData: `
          @use '${path.resolve(__dirname, 'src/TUIRoom/assets/style/element.scss')}
        `,
      },
    },
  },
});
```

また、element-plusに付帯するUIコンポーネントがスタイルを正常に表示できるようにするため、エントリーファイル`packages/renderer/src/main.ts`でelement-plusコンポーネントスタイルをロードする必要があります。



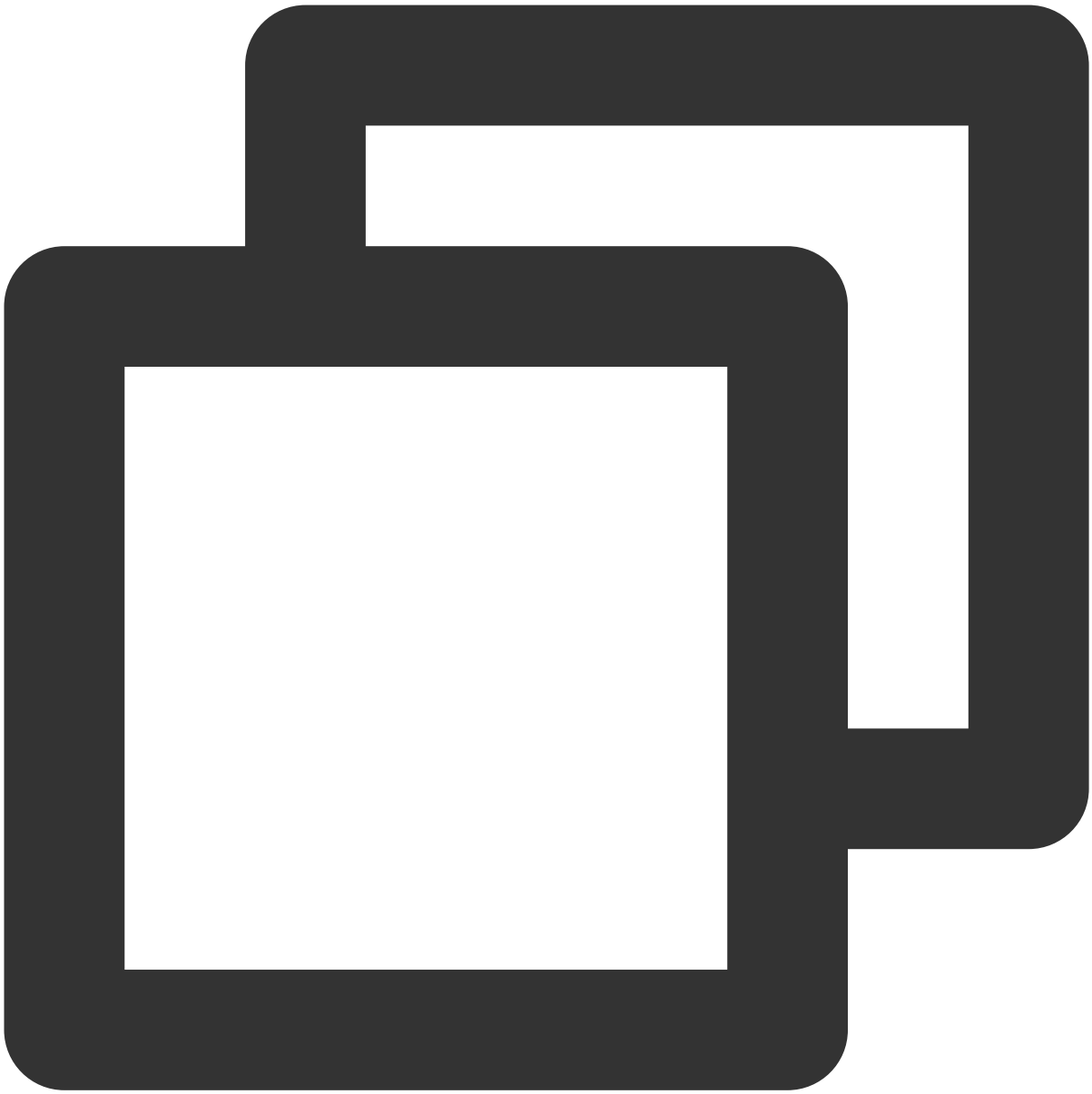
```
// src/main.ts
import 'element-plus/theme-chalk/el-message.css'
import 'element-plus/theme-chalk/el-message-box.css'
```

4. **trtc-electron-sdkのインポート**

UIレイヤーでimport方式により `trtc-electron-sdk` をインポートするためには、コードスタイルを統一します。これを行わない場合はrequire方式でのインポートが必要になります。 `packages/renderer/vite.config.ts` での設定が必要です。

ご注意：

以下の設定項目でresolveの内容を置き換えます。



```
// vite.config.ts

export default defineConfig({
 // ...
 plugins: [
 resolve(
   {
     "trtc-electron-sdk": `
       const TRTCCloud = require("trtc-electron-sdk");
```

```
      const TRTCParams = TRTCCloud.TRTCParams;
      const TRTCAppScene = TRTCCloud.TRTCAppScene;
      const TRTCVideoStreamType = TRTCCloud.TRTCVideoStreamType;
      const TRTCScreenCaptureSourceType = TRTCCloud.TRTCScreenCaptureSourceType;
      const TRTCVideoEncParam = TRTCCloud.TRTCVideoEncParam;
      const Rect = TRTCCloud.Rect;
      const TRTCAudioQuality = TRTCCloud.TRTCAudioQuality;
      const TRTCScreenCaptureSourceInfo = TRTCCloud.TRTCScreenCaptureSourceInfo;
      const TRTCDeviceInfo = TRTCCloud.TRTCDeviceInfo;
      const TRTCVideoQosPreference = TRTCCloud.TRTCVideoQosPreference;
      const TRTCQualityInfo = TRTCCloud.TRTCQualityInfo;
      const TRTCStatistics = TRTCCloud.TRTCStatistics;
      const TRTCVolumeInfo = TRTCCloud.TRTCVolumeInfo;
      const TRTCDeviceType = TRTCCloud.TRTCDeviceType;
      const TRTCDeviceState = TRTCCloud.TRTCDeviceState;
      const TRTCBeautyStyle = TRTCCloud.TRTCBeautyStyle;
      const TRTCVideoResolution = TRTCCloud.TRTCVideoResolution;
      const TRTCVideoResolutionMode = TRTCCloud.TRTCVideoResolutionMode;
      const TRTCVideoMirrorType = TRTCCloud.TRTCVideoMirrorType;
      const TRTCVideoRotation = TRTCCloud.TRTCVideoRotation;
      const TRTCVideoFillMode = TRTCCloud.TRTCVideoFillMode;
      export {
        TRTCParams,
        TRTCAppScene,
        TRTCVideoStreamType,
        TRTCScreenCaptureSourceType,
        TRTCVideoEncParam,
        Rect,
        TRTCAudioQuality,
        TRTCScreenCaptureSourceInfo,
        TRTCDeviceInfo,
        TRTCVideoQosPreference,
        TRTCQualityInfo,
        TRTCStatistics,
        TRTCVolumeInfo,
        TRTCDeviceType,
        TRTCDeviceState,
        TRTCBeautyStyle,
        TRTCVideoResolution,
        TRTCVideoResolutionMode,
        TRTCVideoMirrorType,
        TRTCVideoRotation,
        TRTCVideoFillMode,
      };
      export default TRTCCloud.default;
    `,
  }
```

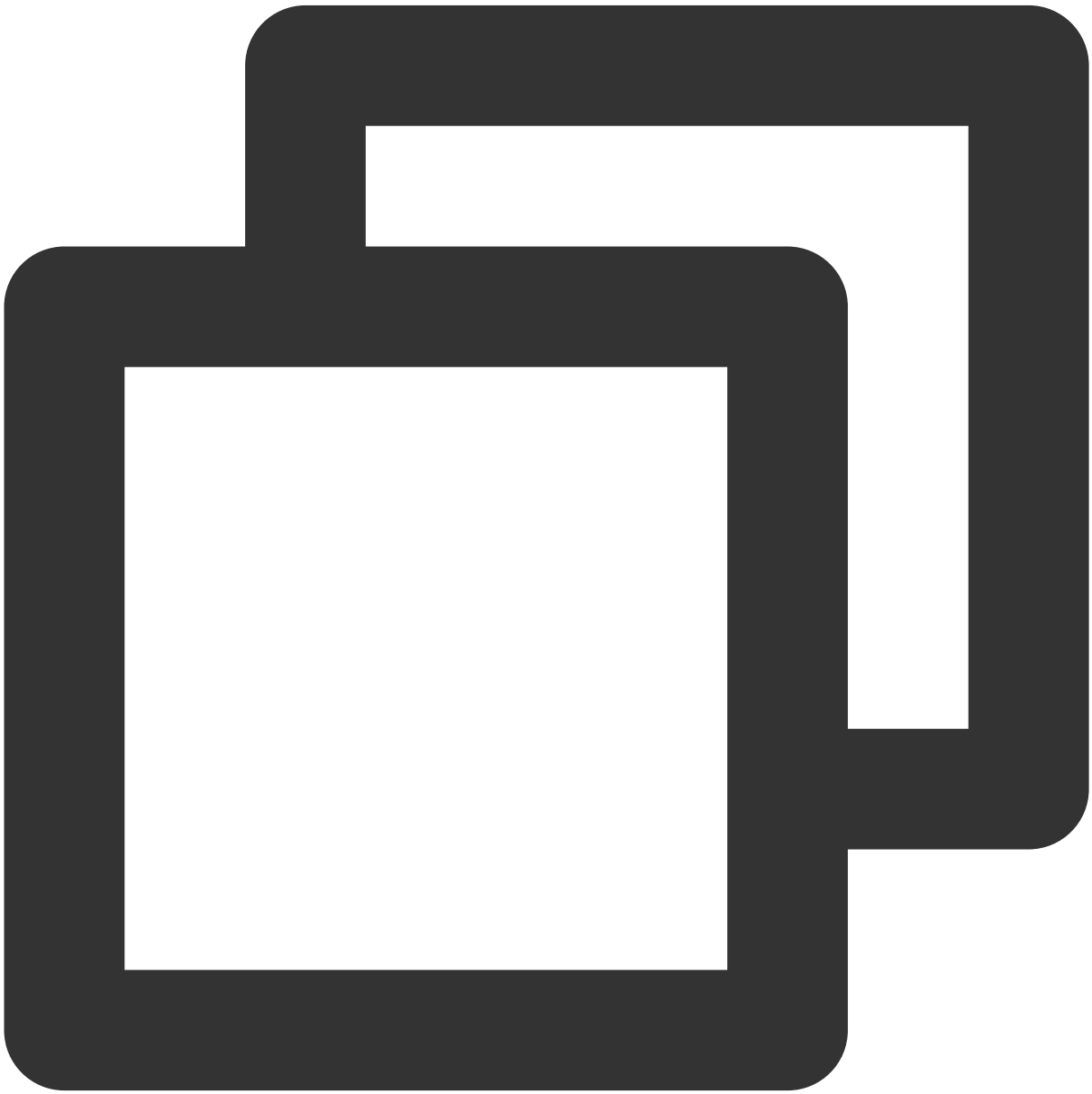
```
  ),
  ]
  // ...
});
```

5. **env.d.tsファイル設定**

env.d.tsファイル設定は `packages/renderer/src/env.d.ts` で行う必要があります。

ご注意：

以下の設定項目は増分設定です。すでに存在するenv.d.tsファイル設定を削除しないでください。



```
// env.d.ts
```

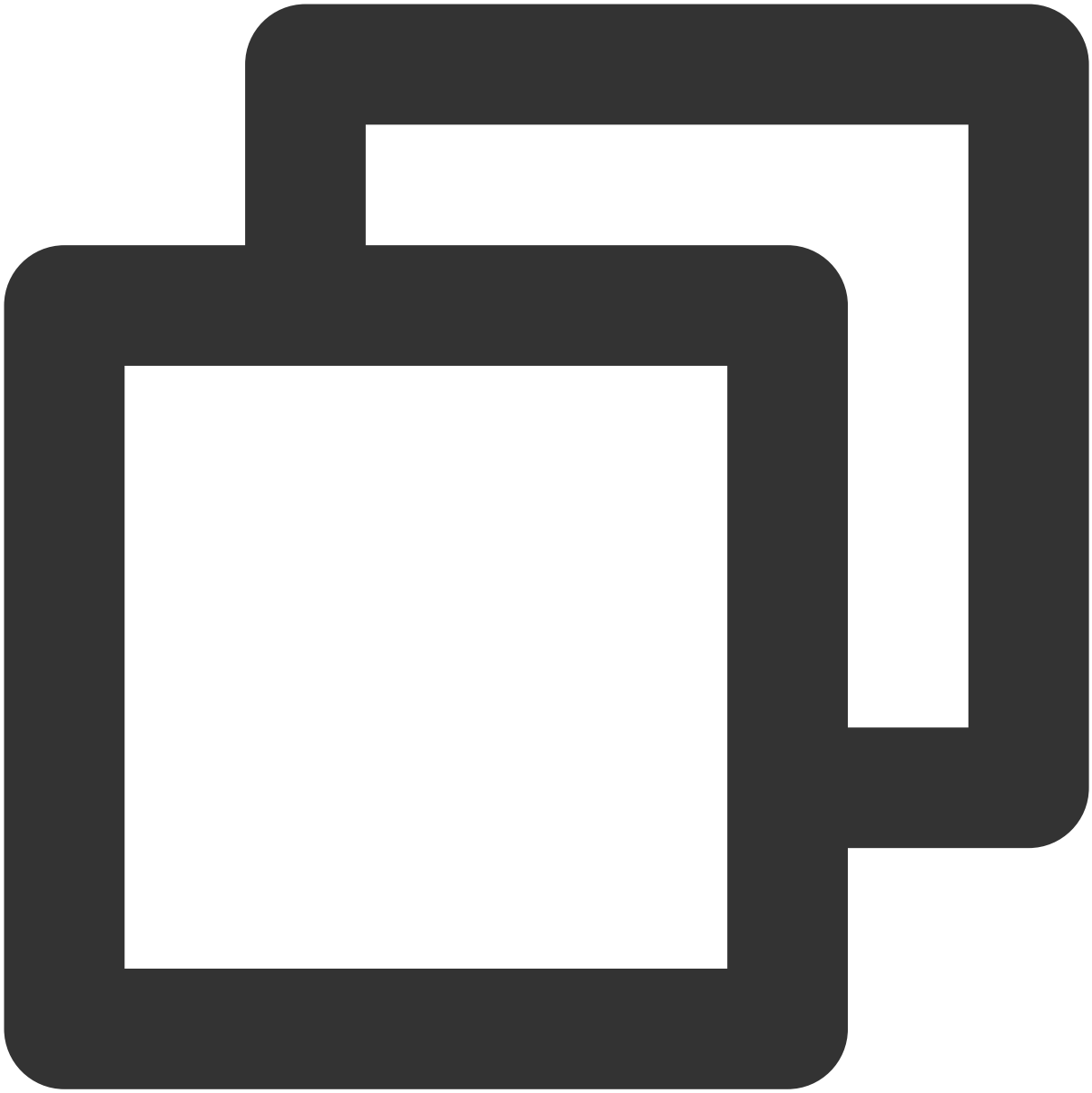
```
declare module 'tsignaling/tsignaling-js' {
  import TSignaling from 'tsignaling/tsignaling-js';
  export default TSignaling;
}

declare module 'tim-js-sdk' {
  import TIM from 'tim-js-sdk';
  export default TIM;
}
```

6. プロジェクト内にimport動的ロードが存在する場合は、アーキテクチャ設定を変更し、esモジュールをパッケージ化して生成する必要があります

esモジュールをパッケージ化して生成するには、`packages/renderer/vite.config.ts` 内で設定を行う必要があります。

注意：

プロジェクト内にimport動的ロードが存在しない場合は、この設定を行わないでください。以下の設定項目は増分設定です。すでに存在するVite設定項目を削除しないでください。

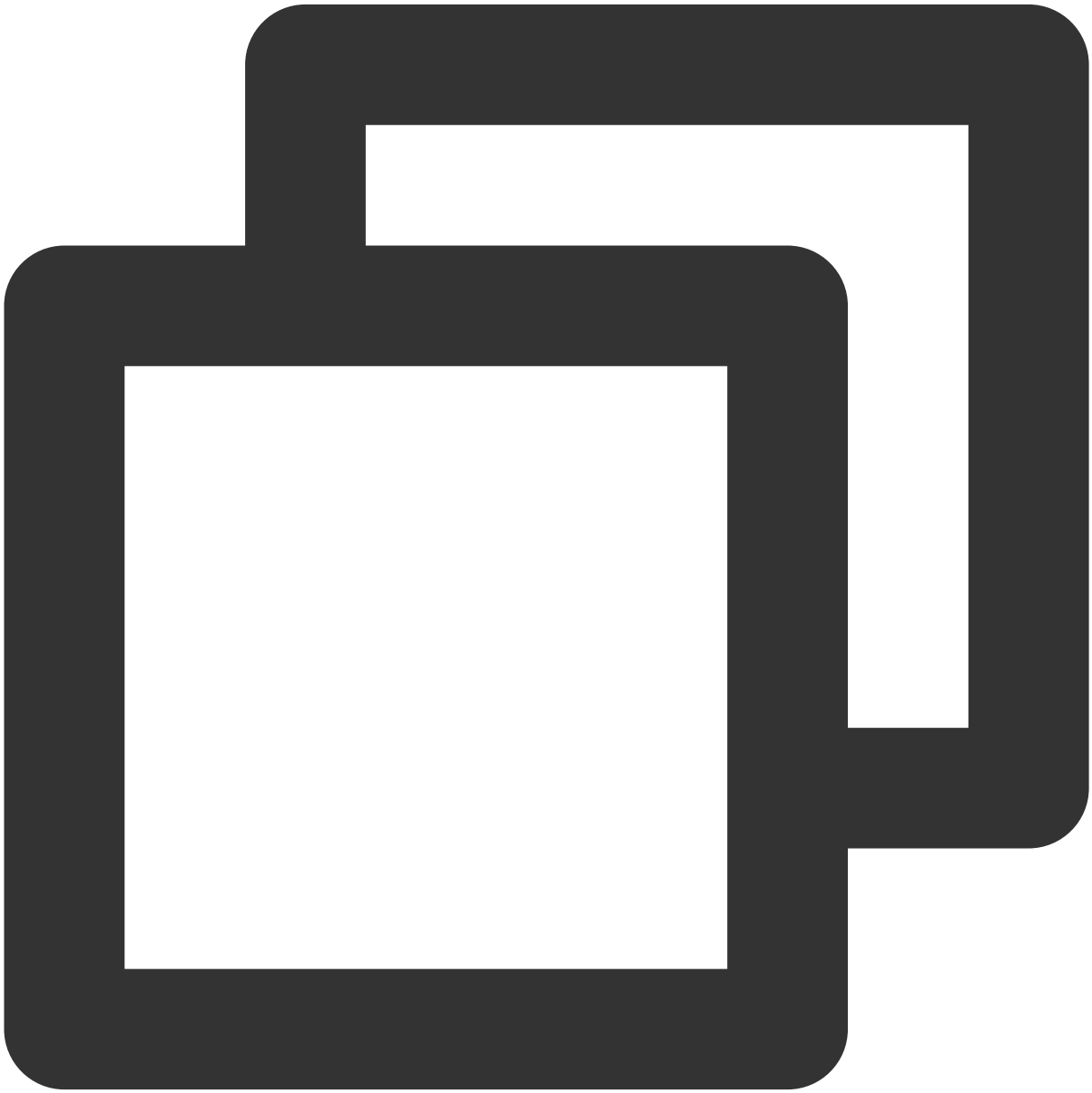


```
// vite.config.ts

export default defineConfig({
    // ...
    build: {
        rollupOptions: {
          output: {
              format: 'es'
          }
        }
    },
```

```
});
```

ステップ5：開発環境の実行

コンソールの実行開発環境でスクリプトを実行し、TUIRoomの含まれるページをブラウザで開くと、そのページでTUIRoomコンポーネントを使用することができます。
ステップ2のスクリプトを使用してElectron + Vue3 + TSプロジェクトを作成する場合は、次の事項を行う必要があります。
1. 開発環境コマンドを実行します。



```
npm run dev
```

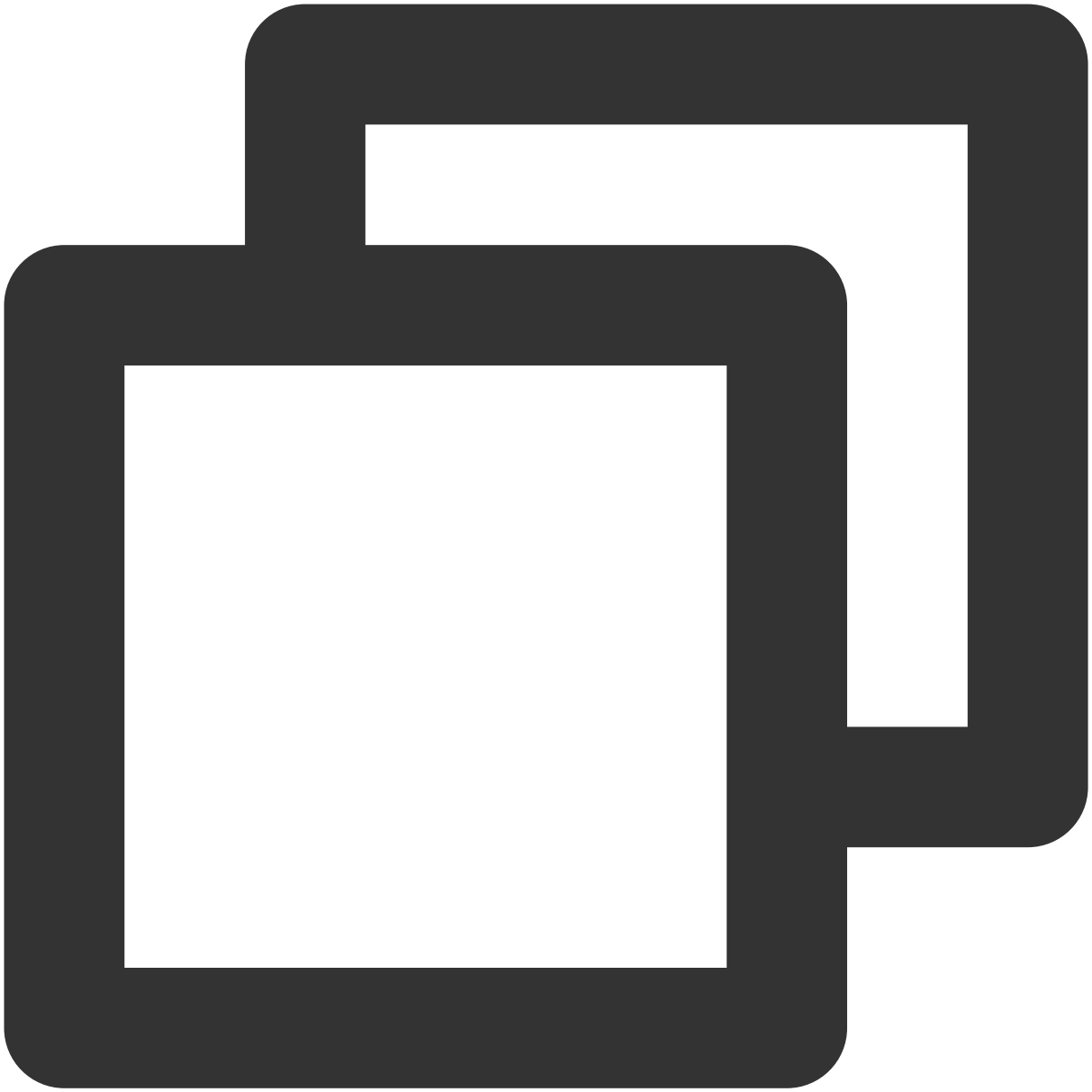
ご注意：

TUIRoomはelement-plusコンポーネントを必要に応じてインポートするため、開発環境のルーティングページでの初回ロード時に反応がやや遅くなりますが、必要なelement-plusのロードが完了すると正常に使用できるようになります。element-plusの必要に応じてのロードがパッケージ化後のページロードに影響することはありません。

2. TUIRoomコンポーネント機能を体験します。

ステップ6：インストールパッケージの作成、実行

コマンドライン端末で、次のコマンドを実行してインストールパッケージを作成します。作成したインストールパッケージは `release` ディレクトリにあり、インストールの実行が可能です。



```
npm run build
```

ご注意：

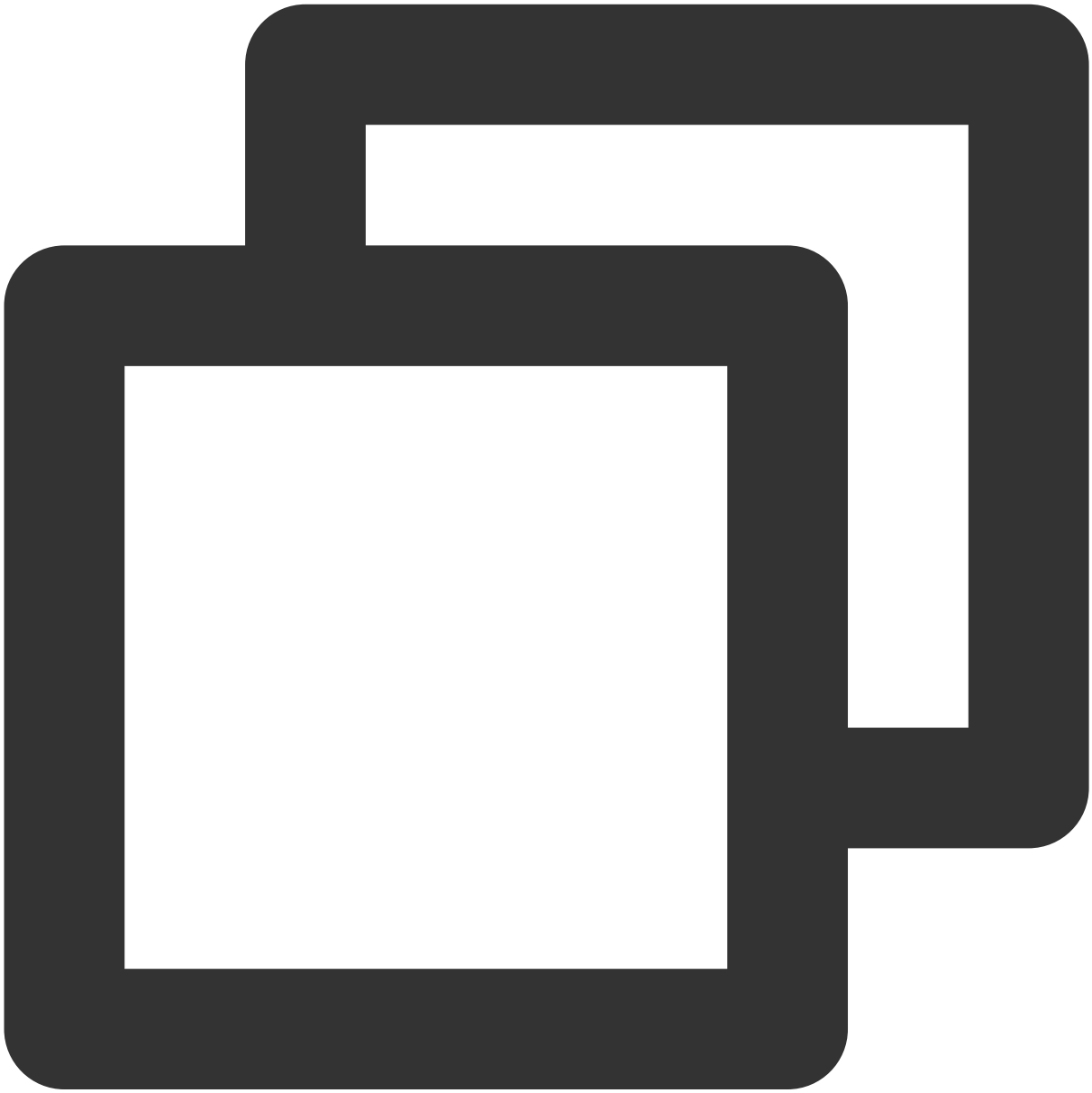
Mac PCを使用したMacインストールパッケージの作成、Windows PCを使用したWindowsインストールパッケージの作成のみ可能です。

付録：TUIRoom API

TUIRoomインターフェース

init

TUIRoomデータを初期化します。TUIRoomを使用するすべてのユーザーはinitメソッドを呼び出す必要があります。



```
TUIRoomRef.value.init(roomData);
```

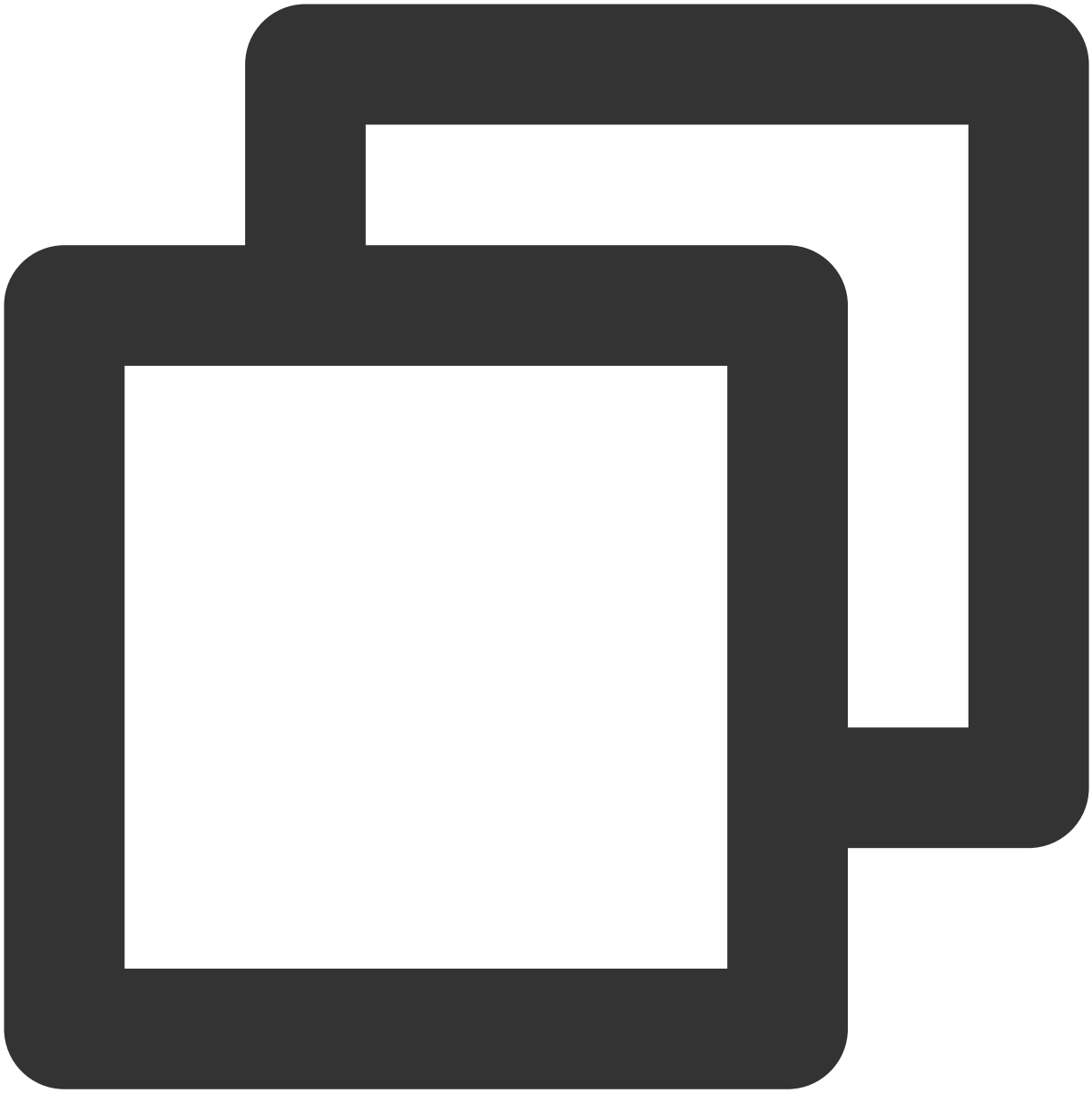
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

roomData	object	
roomData.sdkAppId	number	お客様のSDKAppId
roomData.userId	string	ユーザーの固有ID
roomData.userSig	string	ユーザーのUserSig
roomData.userName	string	ユーザーのニックネーム
roomData.userAvatar	string	ユーザーのプロフィール画像
roomData.shareUserId	string	任意入力。ユーザーが画面共有を行う際のUserIdであり、`shareUserId = share_${userId}` である必要があります。画面共有機能がなければ渡さなくても結構です
roomData.shareUserSig	string	任意入力。ユーザーが画面共有を行う際のUserSig

createRoom

キャスターがルームを作成します。



```
TUIRoomRef.value.createRoom(roomId, roomMode, roomParam);
```

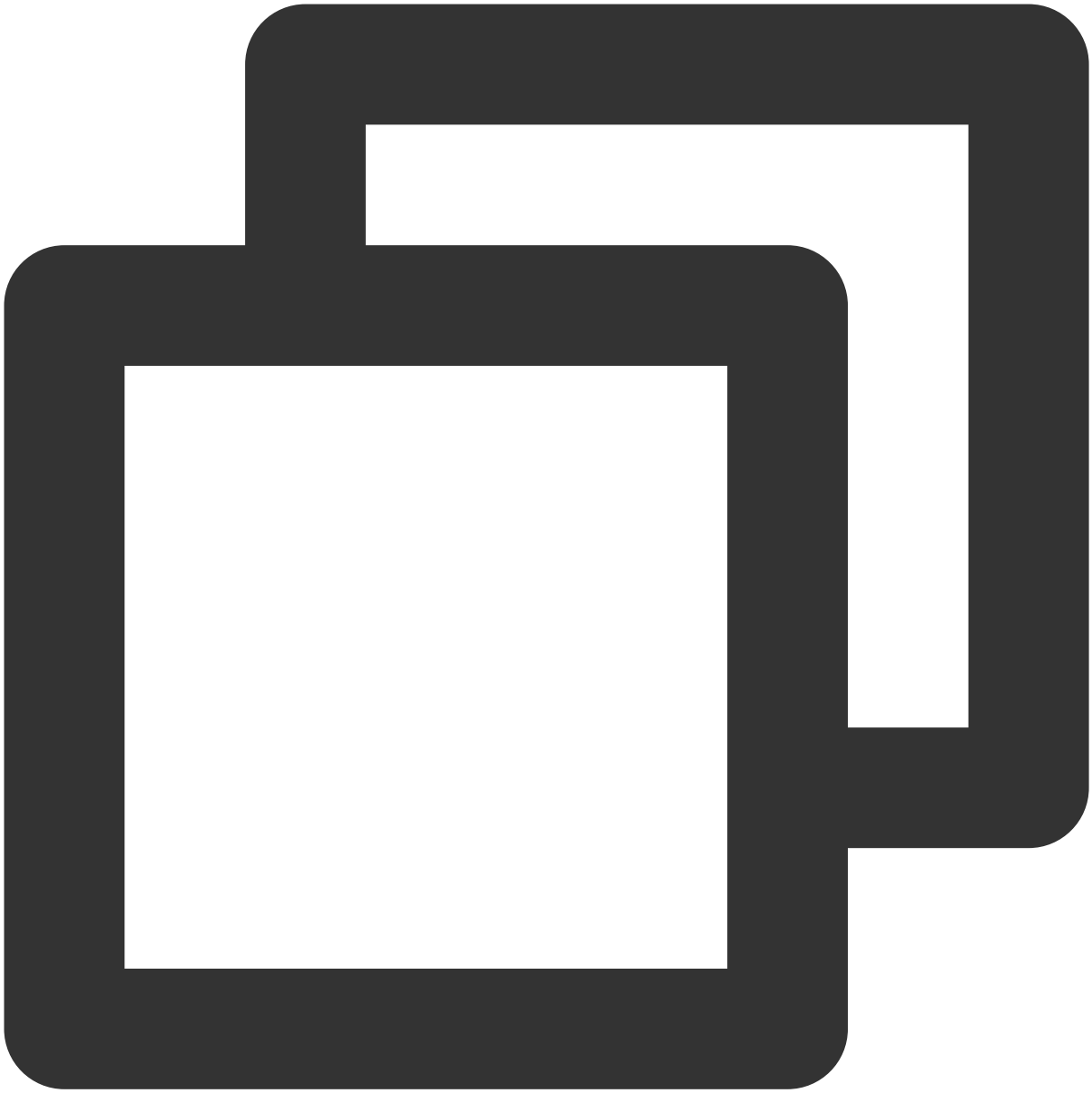
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
roomId	number	ルームID
roomMode	string	ルームモード。'FreeSpeech'（自由発言モード）と'ApplySpeech'（挙手発言モード）があり、デフォルトで

		は'FreeSpeech'です。現在は自由発言モードのみサポートされていますのでご注意ください
roomParam	Object	任意入力
roomParam.isOpenCamera	string	任意入力。入室時にカメラをオンにするかどうか。デフォルトではオフ
roomParam.isOpenMicrophone	string	任意入力。入室時にマイクをオンにするかどうか。デフォルトではオフ
roomParam.defaultCameraId	string	任意入力。デフォルトのカメラデバイスID
roomParam.defaultMicrophoneId	string	任意入力。デフォルトのマイクデバイスID
roomParam.defaultSpeakerId	String	任意入力。デフォルトのスピーカーデバイスID

enterRoom

一般メンバーが入室します。



```
TUIRoomRef.value.enterRoom(roomId, roomParam);
```

パラメータは下表に示すとおりです。

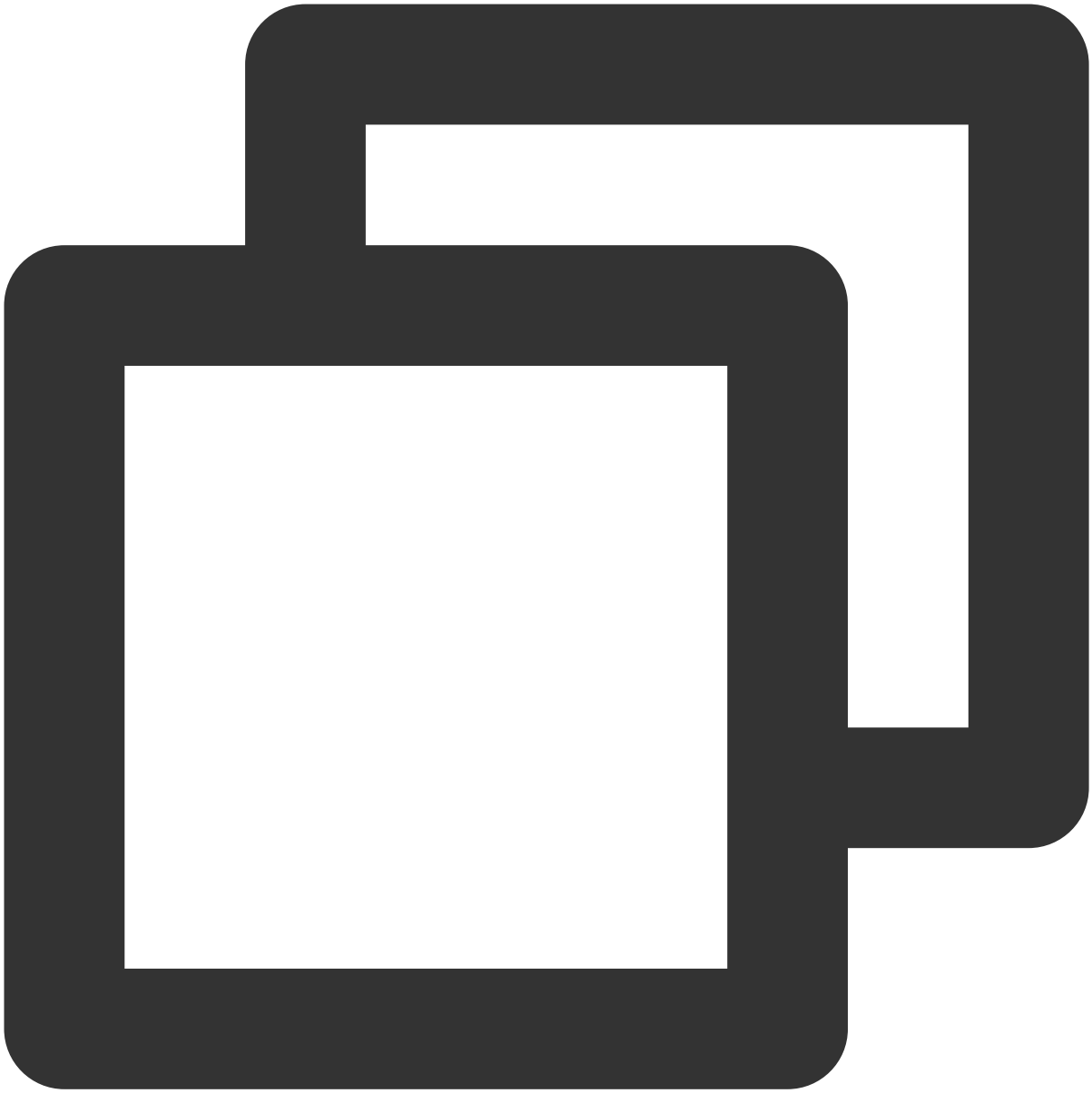
パラメータ	タイプ	意味
roomId	number	ルームID
roomParam	Object	任意入力
roomParam.isOpenCamera	string	任意入力。入室時にカメラをオンにするかどうか。デフォル

		トではオフ
roomParam.isOpenMicrophone	string	任意入力。入室時にマイクをオンにするかどうか。デフォルトではオフ
roomParam.defaultCameraId	string	任意入力。デフォルトではカメラデバイスID
roomParam.defaultMicrophoneId	string	任意入力。デフォルトではマイクデバイスID
roomParam.defaultSpeakerId	String	任意入力。デフォルトではスピーカーデバイスID

TUIRoomイベント

onRoomCreate

ルーム作成のコールバックです。

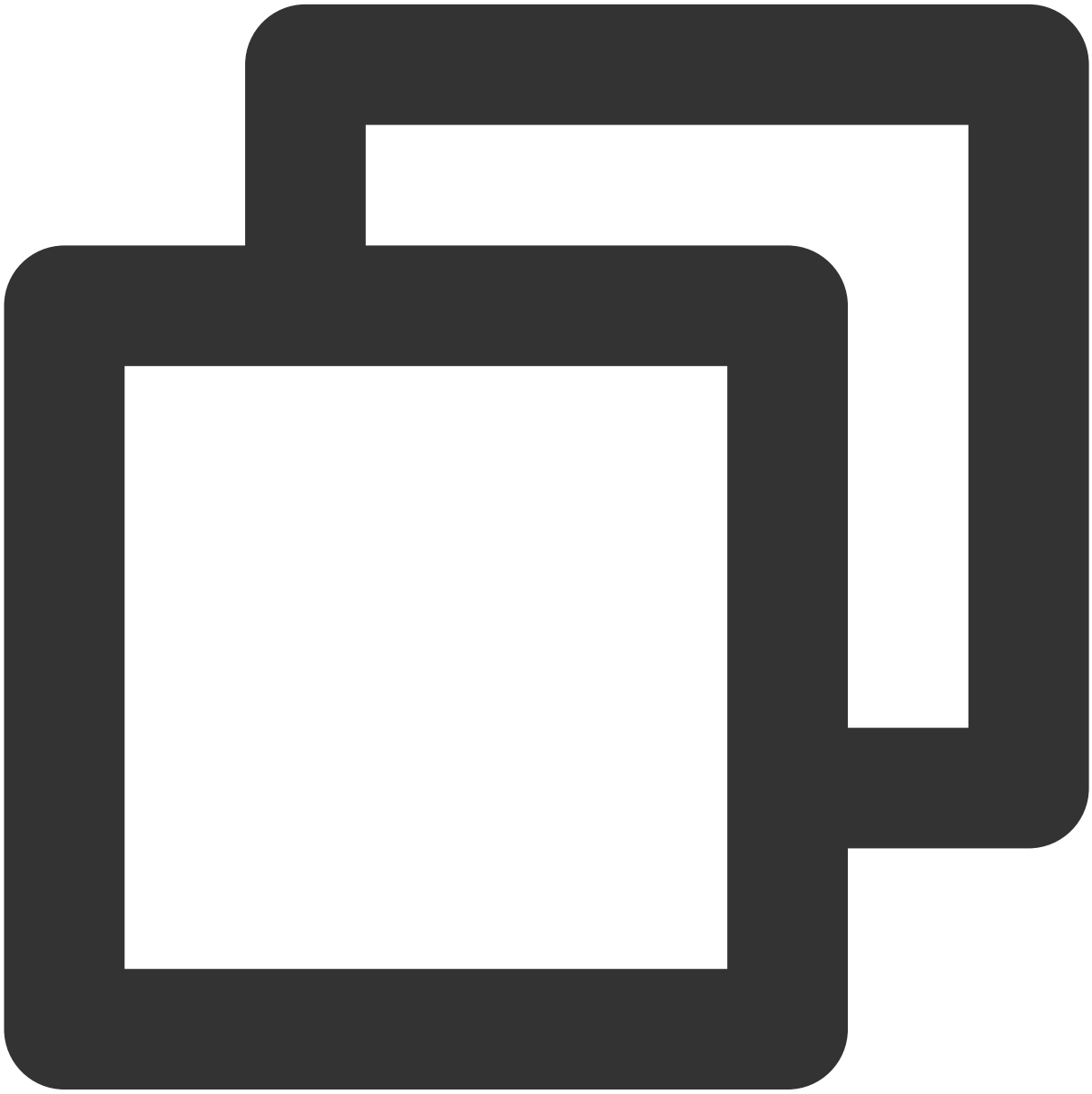


```
<template>
  <room ref="TUIRoomRef" @on-room-create="handleRoomCreate"></room>
</template>

<script setup lang="ts">
  // TUIRoomコンポーネントをインポートします。インポートパスが正しいかどうかよく確認してください
  import Room from './TUIRoom/index.vue';

  function handleRoomCreate(info) {
    if (info.code === 0) {
      console.log('ルーム作成成功')
```

```
    }
  }
</script>
```

onRoomEnter

入室コールバックです。

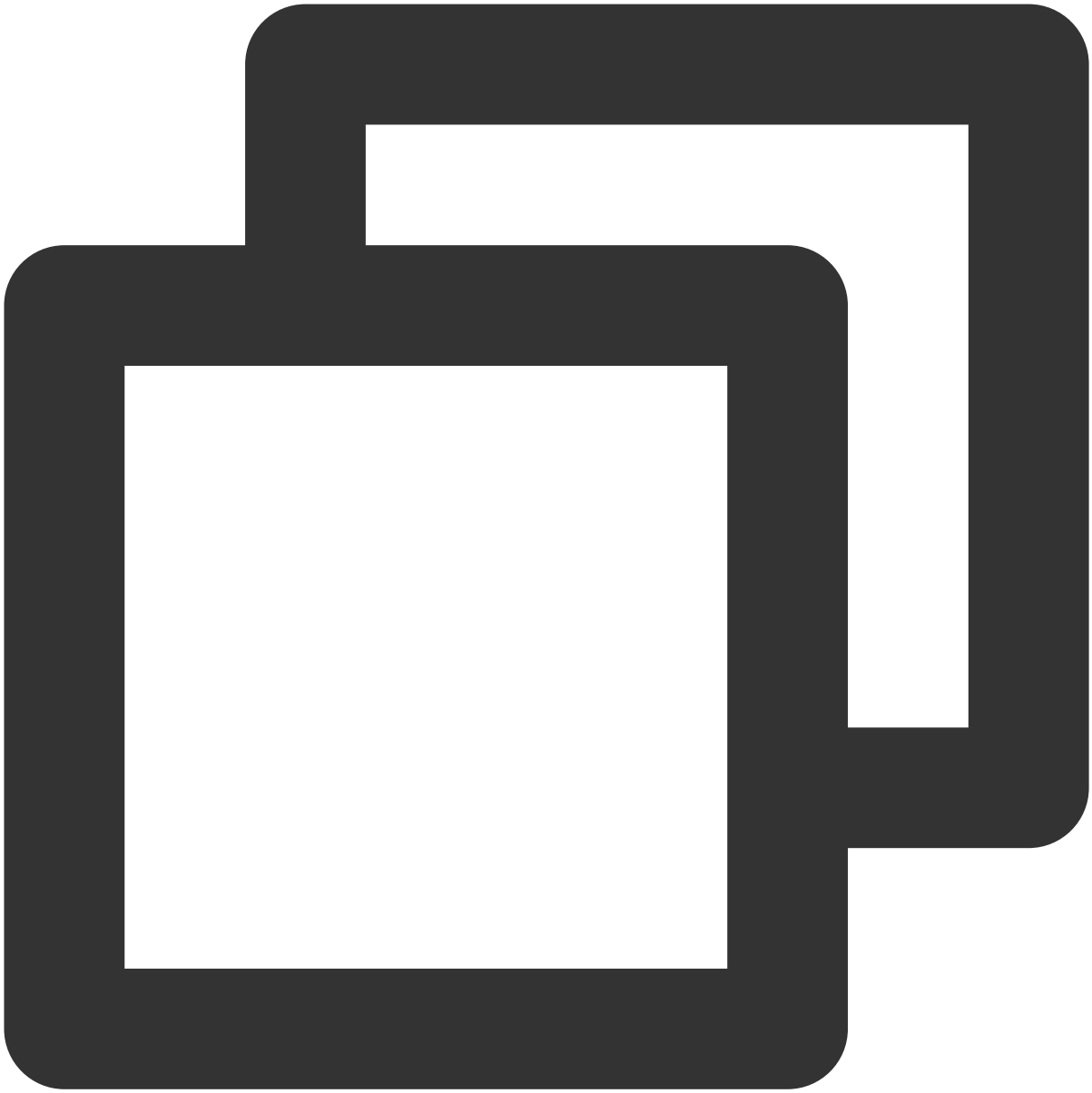


```
<template>
  <room ref="TUIRoomRef" @on-room-enter="handleRoomEnter"></room>
</template>
```

```
<script setup lang="ts">
  // TUIRoomコンポーネントをインポートします。インポートパスが正しいかどうかよく確認してください
  import Room from './TUIRoom/index.vue';

  function handleRoomEnter(info) {
    if (info.code === 0) {
      console.log('入室成功')
    }
  }
</script>
```

onRoomDestory

キャスターのルーム破棄通知。

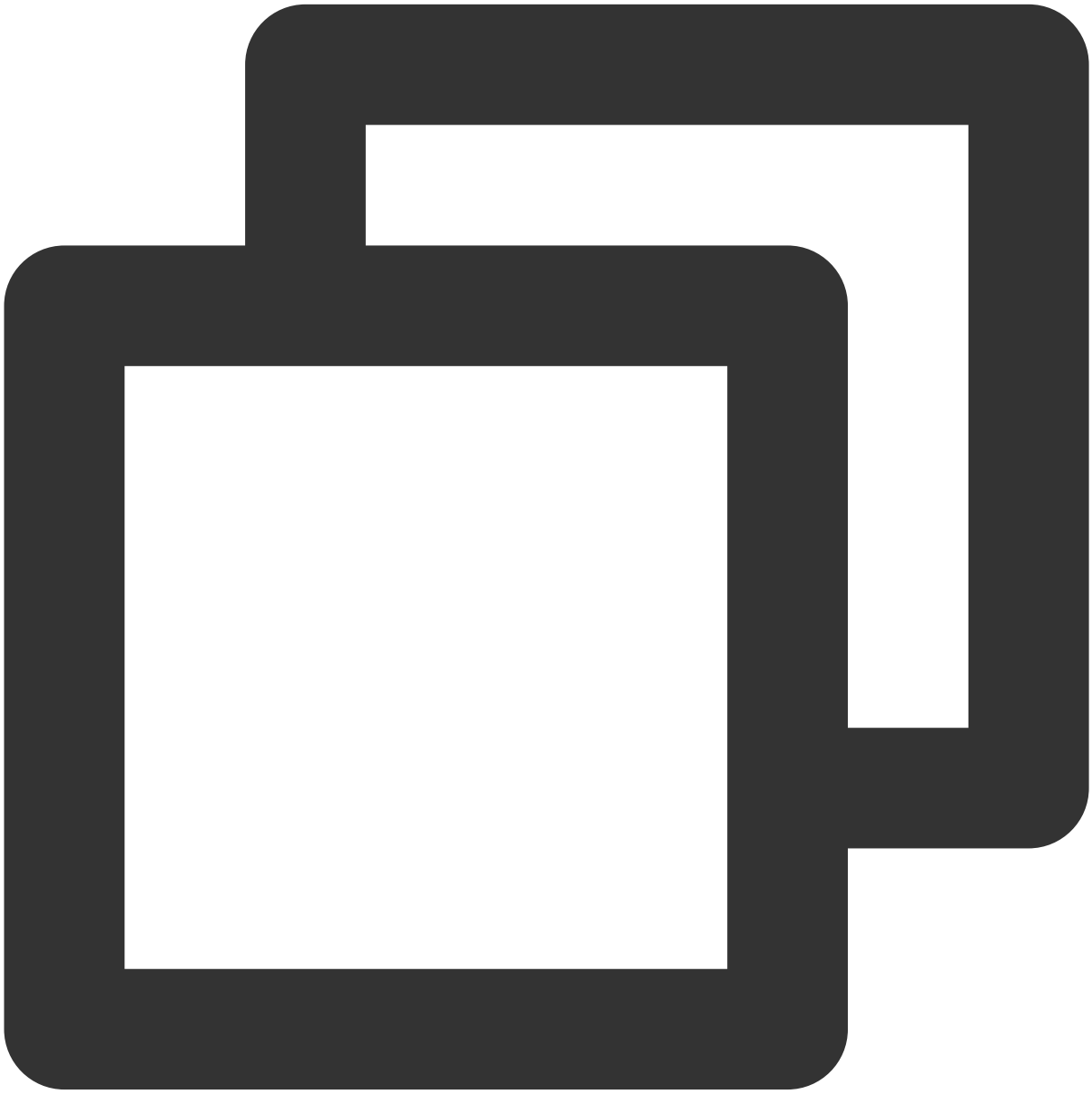


```
<template>
  <room ref="TUIRoomRef" @on-room-destory="handleRoomDestory"></room>
</template>

<script setup lang="ts">
  // TUIRoomコンポーネントをインポートします。インポートパスが正しいかどうかよく確認してください
  import Room from './TUIRoom/index.vue';

  function handleRoomDestory(info) {
    if (info.code === 0) {
      console.log('キャスターが破棄に成功')
```

```
    }
  }
</script>
```

onRoomExit

一般メンバーのルーム退出通知。



```
<template>
  <room ref="TUIRoomRef" @on-room-exit="handleRoomExit"></room>
</template>
```

```
<script setup lang="ts">
  // TUIRoomコンポーネントをインポートします。インポートパスが正しいかどうかよく確認してください
  import Room from './TUIRoom/index.vue';

  function handleRoomExit(info) {
    if (info.code === 0) {
      console.log('一般メンバーのルーム退出成功')
    }
  }
</script>
```

TUIRoom APIのクエリー
TUIRoom(Android)

最終更新日：：2024-07-19 15:35:35

TUIRoomは、Tencent CloudのTencent Real-Time Communication（TRTC）およびIMサービスを基に組み合わせたコンポーネントで、以下の機能をサポートしています。
キャスターがルームを作成し、入室者はルームナンバーを入力した後に入室できます。
入室者の間で画面共有を行います。
各種のテキストメッセージとカスタムメッセージの送信をサポートします。
説明：
TUIKitシリーズコンポーネントはTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期してアクティブ化することができます。 IMサービスの課金ルールの詳細については、Instant Messagingの料金説明をご参照ください。TRTCをアクティブ化すると、関連するIM SDKの体験版がデフォルトでアクティブ化されます。これは100 DAUのみをサポートします。
TUIRoomはオープンソースのClassであり、Tencent Cloudの2つのクローズドソースのSDKに依存しています。具体的な実装プロセスについては、 多人数オーディオビデオルーム(Android)をご参照ください。
TRTC SDK：TRTC SDKを低遅延のオーディオビデオルームコンポーネントとして使用します。
IM SDK：IM SDK を利用してチャットルームの機能（**IM SDKはAndroidバージョンを使用**）を実装します。

TUIRoom API概要

TUIRoomCore基本関数

API	説明
getInstance	シングルトンオブジェクトを取得します。
destroyInstance	シングルトンオブジェクトを破棄します。
setListener	イベントコールバックを設定します。

ルーム関連インターフェース関数

API	説明
createRoom	ルームの作成（キャスターが呼び出し）。
destroyRoom	ルームの破棄（キャスターが呼び出し）。

enterRoom	入室（参加者が呼び出し）。
leaveRoom	退室（参加者が呼び出し）。
getRoomInfo	ルーム情報の取得。
getRoomUsers	ルーム内全メンバー情報の取得。
getUserInfo	特定ユーザーの情報の取得。
transferRoomMaster	キャスター権限の移転（キャスターが呼び出し）。

ローカルのオーディオビデオ操作インターフェース

API	説明
startCameraPreview	ローカルビデオのプレビュー画面を立ち上げます。
stopCameraPreview	ローカルのビデオキャプチャおよびプレビューを停止します。
startLocalAudio	マイクキャプチャを起動します。
stopLocalAudio	マイクキャプチャを停止します。
setVideoMirror	ローカル画面のイメージプレビューモードを設定します。
setSpeaker	スピーカーの起動を設定します。

リモートユーザーに関するインターフェース

API	説明
startRemoteView	指定メンバーのリモートビデオ画面をサブスクリプションし再生します。
stopRemoteView	リモートビデオ画面のサブスクリプションをキャンセルし再生を停止します。

チャットメッセージ送信インターフェース

API	説明
sendChatMessage	チャットメッセージを送信します。
sendCustomMessage	カスタムメッセージを送信します。

フィールドコントロール関連インターフェース

API	説明
muteUserMicrophone	特定ユーザーのマイクを無効化/再有効化します。
muteAllUsersMicrophone	全ユーザーのマイクを無効化/再有効化し、ステータスをルーム情報に同期させます。
muteUserCamera	特定ユーザーのカメラを無効化/再有効化します。
muteAllUsersCamera	全ユーザーのカメラを無効化/再有効化し、ステータスをルーム情報に同期させます。
muteChatRoom	チャットルームのミュートを開始/停止します（キャスターが呼び出し）。
kickOffUser	ルーム内の特定ユーザーをリムーブします（キャスターが呼び出し）。
startCallingRoll	キャスターが点呼を開始します。
stopCallingRoll	キャスターが点呼を終了します。
replyCallingRoll	参加者がキャスターの点呼に応答します。
sendSpeechInvitation	キャスターがメンバーに発言するようインビテーションを送信します。
cancelSpeechInvitation	キャスターがメンバーの発言のためのインビテーションをキャンセルします。
replySpeechInvitation	参加者がキャスターの発言申請に同意/拒否します。
sendSpeechApplication	参加者が発言を申請します。
replySpeechApplication	キャスターが参加者の発言申請に同意/拒否します。
forbidSpeechApplication	キャスターが発言申請を禁止します。
sendOffSpeaker	キャスターが参加者に発言を停止するよう命令します。
sendOffAllSpeakers	キャスターが全員に発言を停止するよう命令します。
exitSpeechState	参加者は発言を停止し、視聴者になります。

画面共有インターフェース

API	説明
startScreenCapture	画面共有を開始。
stopScreenCapture	画面キャプチャの停止。

美顔フィルターに関するインターフェース関数

API	説明
getBeautyManager	美顔管理オブジェクトTXBeautyManager。を取得します。

関連設定インターフェース

API	説明
setVideoQosPreference	ネットワークトラフィックコントロール関連パラメータを設定します。

SDKバージョンインターフェース関数の取得

API	説明
getSDKVersion	SDKバージョンを取得します。

TUIRoomCoreListener API概要

エラーイベントコールバック

API	説明
onError	エラーのコールバック。

基本イベントコールバック

API	説明
onDestroyRoom	ルーム解散のコールバック。
onUserVoiceVolume	音量の大きさのコールバック。
onRoomMasterChanged	キャスター変更のコールバック。

リモートユーザーイベントコールバック

API	説明
onRemoteUserEnter	リモートユーザー入室コールバック。
onRemoteUserLeave	リモートユーザー退室コールバック。

onRemoteUserCameraAvailable	リモートユーザーがカメラビデオを起動するかどうかのコールバック。
onRemoteUserScreenVideoAvailable	リモートユーザーが画面共有を開始するかどうかのコールバック。
onRemoteUserAudioAvailable	リモートユーザーがオーディオのアップストリームを開始するかどうかのコールバック。
onRemoteUserEnterSpeechState	リモートユーザーの発言開始のコールバック。
onRemoteUserExitSpeechState	リモートユーザーの発言終了のコールバック。

メッセージイベントのコールバック

API	説明
onReceiveChatMessage	テキストメッセージ受信のコールバック。
onReceiveRoomCustomMsg	カスタムメッセージ受信のコールバック。

フィールドコントロールイベントコールバック

API	説明
onReceiveSpeechInvitation	ユーザーがキャスターの発言要請を受信した場合のコールバック。
onReceiveInvitationCancelled	ユーザーがキャスターの発言要請キャンセルを受信する場合のコールバック。
onReceiveSpeechApplication	キャスターがユーザーの発言申請を受信する場合のコールバック。
onSpeechApplicationCancelled	ユーザーが発言申請をキャンセルする場合のコールバック。
onSpeechApplicationForbidden	キャスターが発言申請を禁止する場合のコールバック。
onOrderedToExitSpeechState	参加者が発言の停止をリクエストされる場合のコールバック。
onCallingRollStarted	キャスターが点呼を開始し、参加者が受信する場合のコールバック。
onCallingRollStopped	キャスターが点呼を終了し、参加者が受信する場合のコールバック。
onMemberReplyCallingRoll	参加者が点呼に応答し、キャスターが受信する場合のコールバック。
onChatRoomMuted	キャスターがチャットルームのミュートを変更する場合のコールバック。
onMicrophoneMuted	キャスターがマイクの無効化を設定する場合のコールバック。
onCameraMuted	キャスターがカメラの無効化を設定する場合のコールバック。

onReceiveKickedOff	参加者がキャスターからキックアウトされた場合のコールバック。

統計および品質コールバック

API	説明
onStatistics	技術指標統計のコールバック。
onNetworkQuality	ネットワーク品質のコールバック。

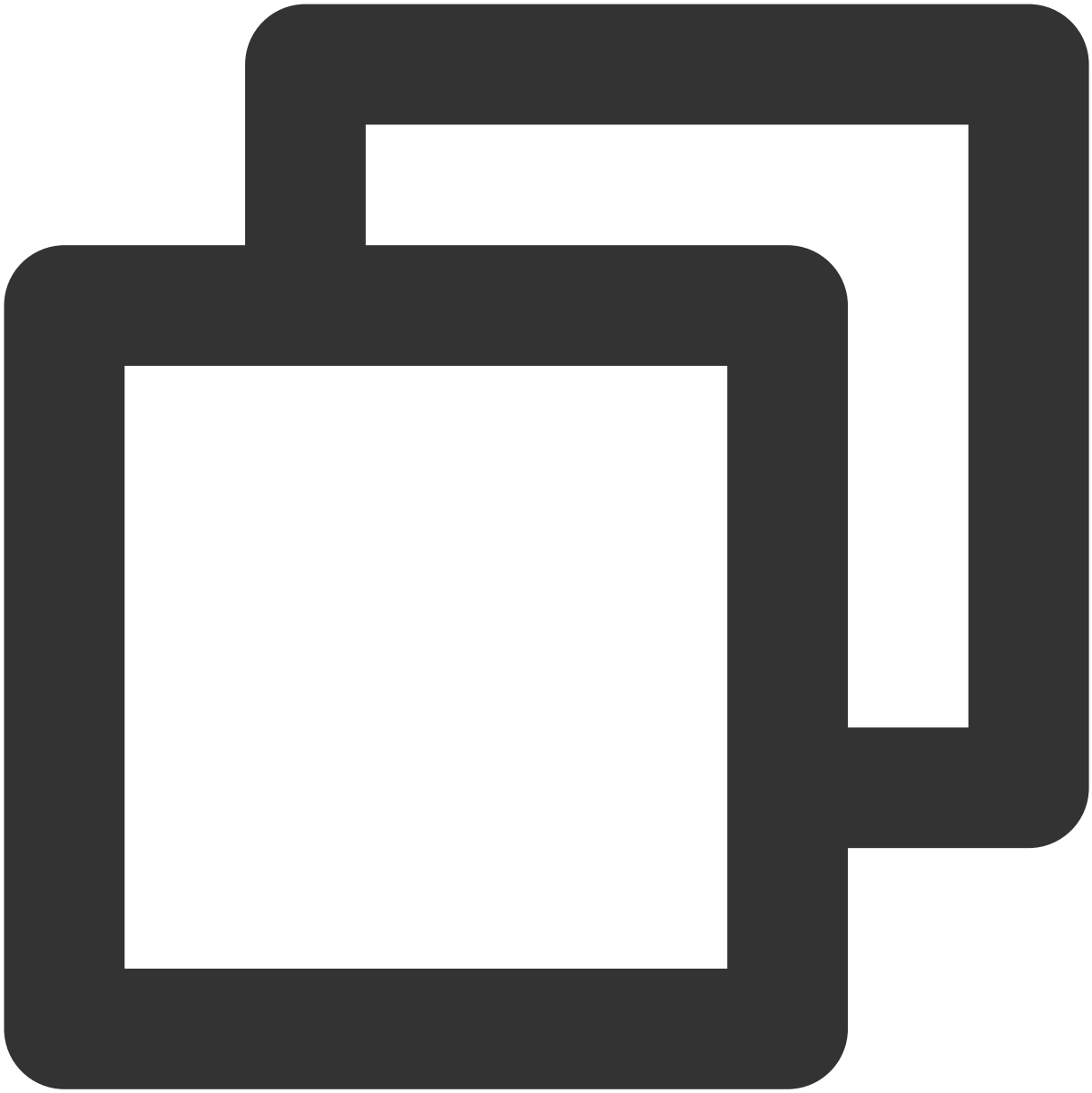
画面共有関連コールバック

API	説明
onScreenCaptureStarted	画面共有開始のコールバック。
onScreenCaptureStopped	画面共有停止のコールバック。

TUIRoomCore基本関数

getInstance

TUIRoomCore シングルトンオブジェクトを取得します。

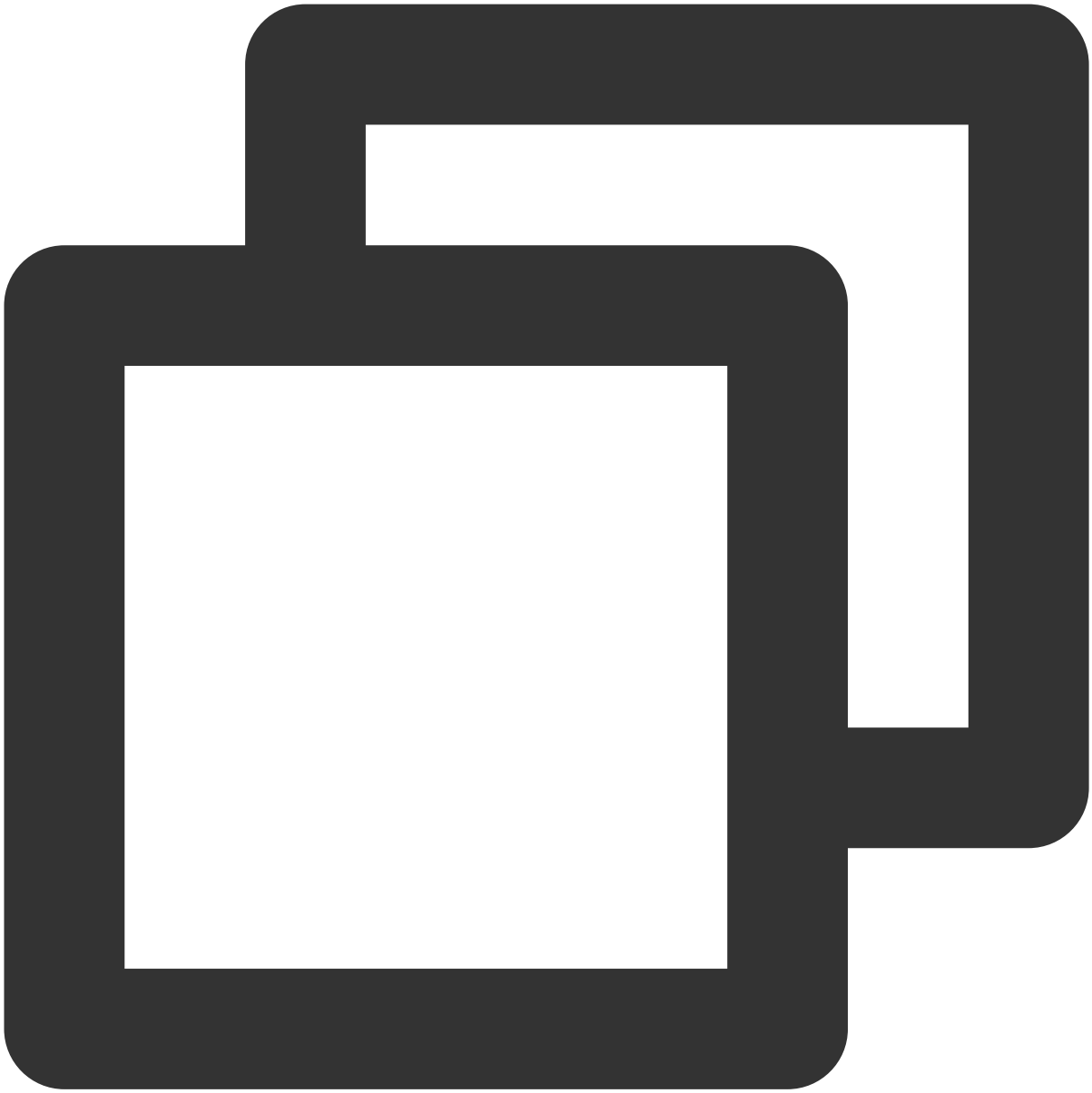


```
public static TUIRoomCore getInstance(Context context);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
context	Context	Androidコンテキスト。内部ではApplicationContextに変換してシステムAPIの呼び出しに使用します。

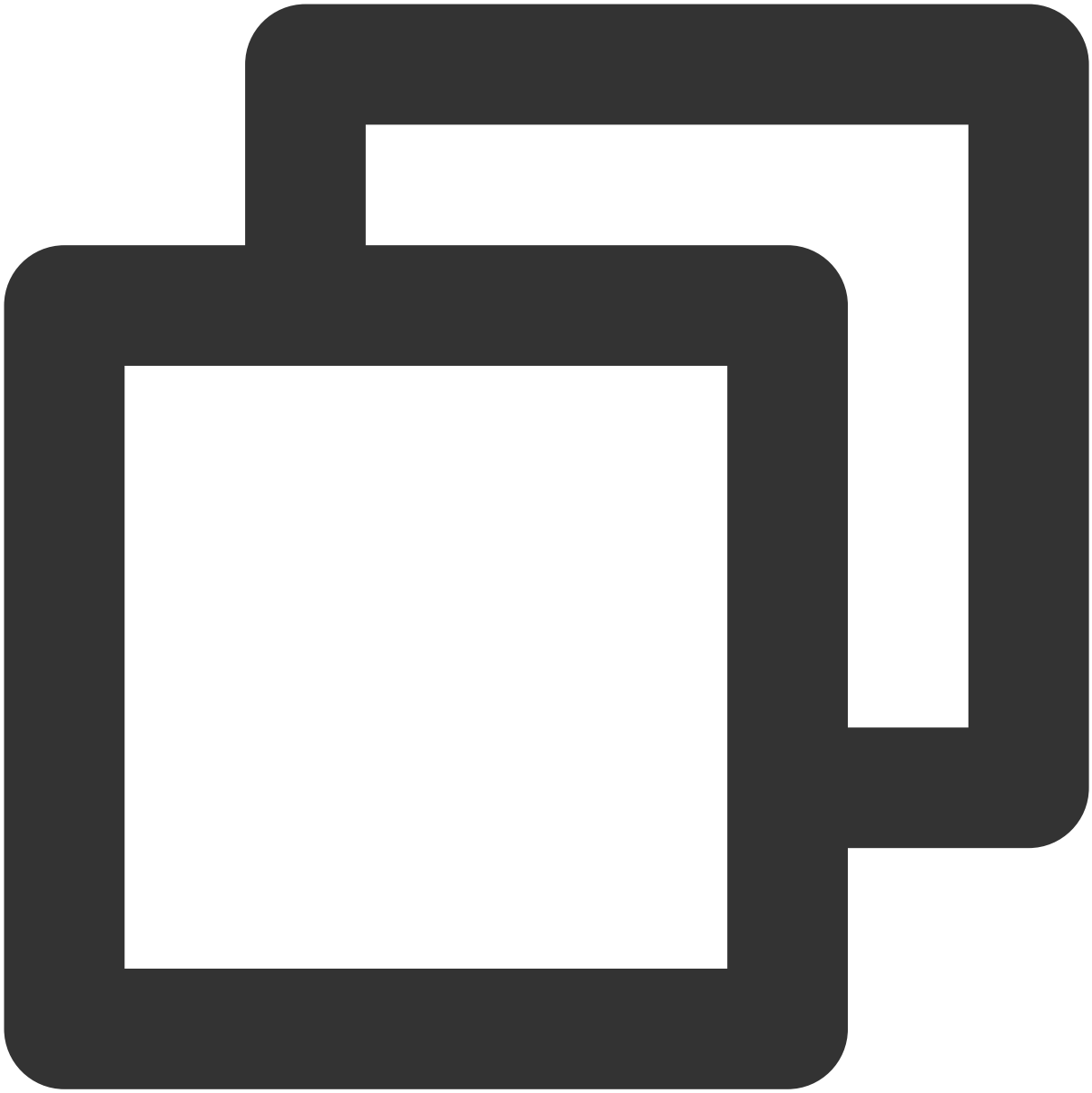
destroyInstance



```
void destroyInstance();
```

setListener

TUIRoomCore イベントコールバック。TUIRoomCoreListener を介してTUIRoomCoreの各種ステータス通知を取得できます。



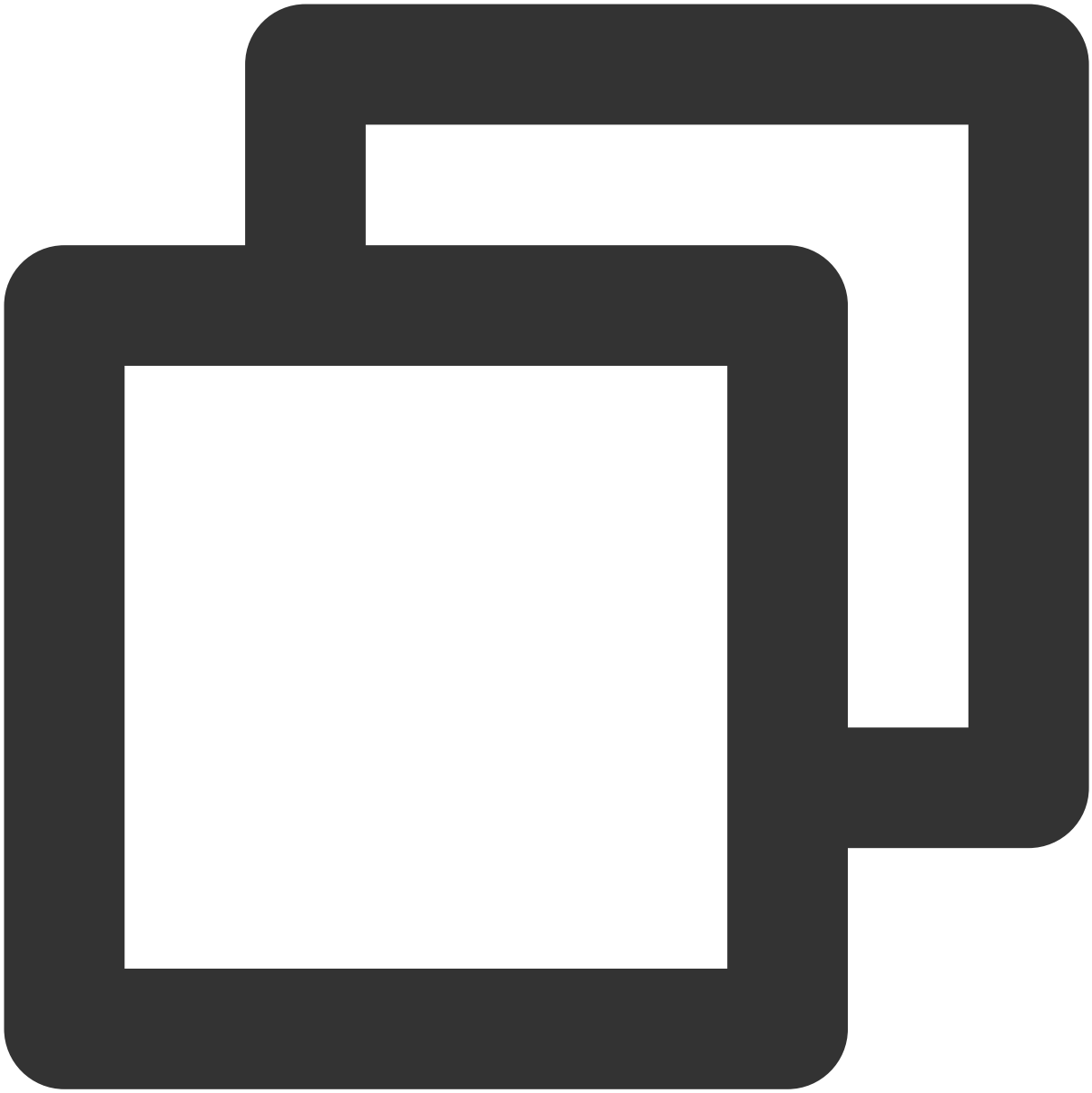
```
void setListener(TUIRoomCoreListener listener);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
listener	TUIRoomCoreListener	イベントコールバッククラスを受信します。

createRoom

ルームの作成（キャスターが呼び出し）。



```
void createRoom(String roomId, TUIRoomCoreDef.SpeechMode speechMode, TUIRoomCoreCal
```

パラメータは下表に示すとおりです。

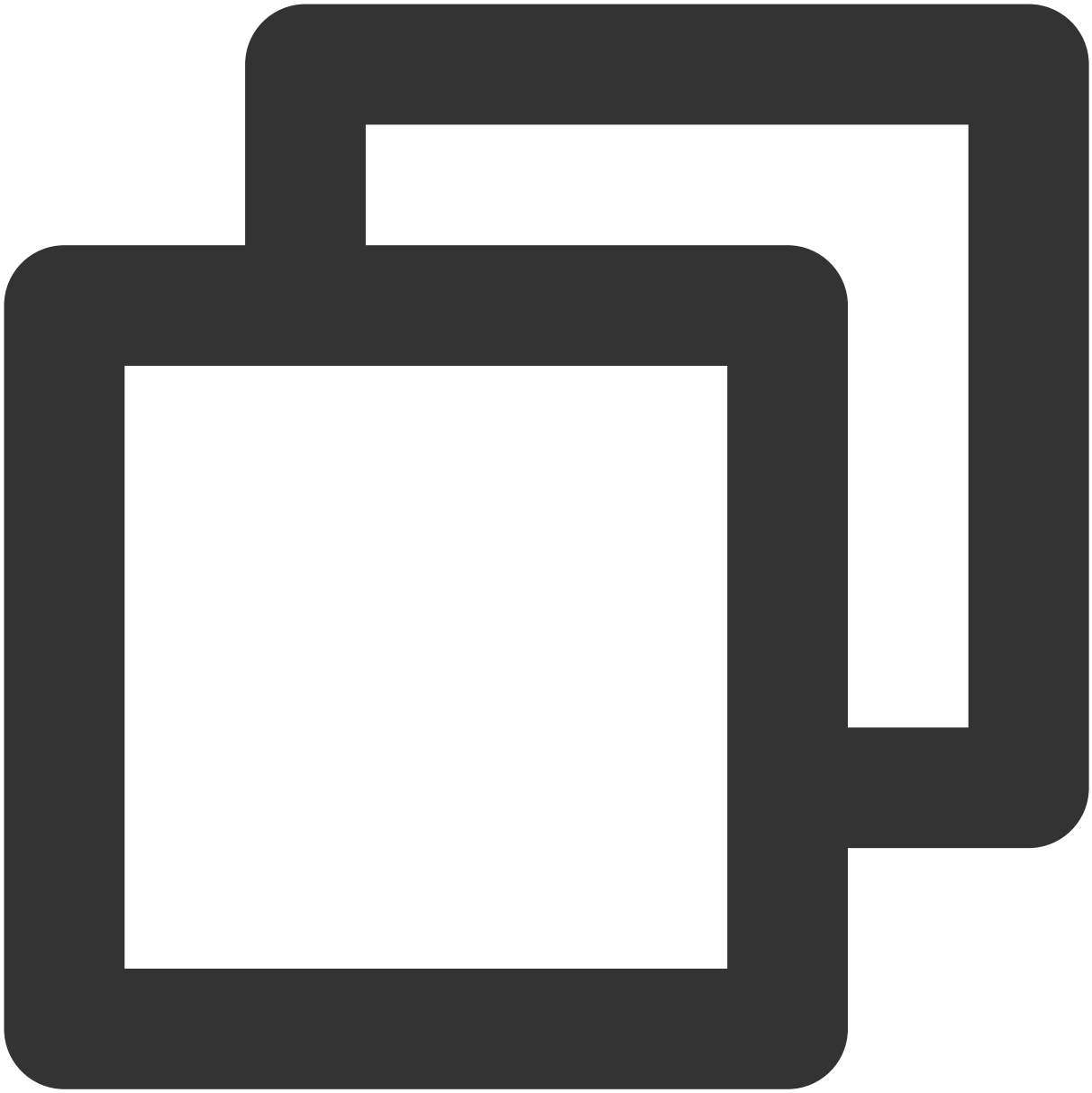
パラメータ	タイプ	意味
roomId	String	ルームID。ご自身でアサインし、一元管理する必要があります。
speechMode	TUIRoomCoreDef.SpeechMode	発言モード。

callback	TUIRoomCoreCallback.ActionCallback	ルームの作成結果のコールバック。

キャスターの通常の呼び出しフローは以下のとおりです。
1. **キャスター**が `createRoom()` を呼び出し、ルームを作成します。ルーム作成の成否は `TUIRoomCoreCallback.ActionCallback` でキャスターに通知されます。
2. **キャスター**が `startCameraPreview()` を呼び出し、カメラキャプチャとプレビューを起動します。
3. **キャスター**が `startLocalAudio()` を呼び出し、ローカルマイクを起動します。

destroyRoom

ルームの破棄（キャスターが呼び出し）。キャスターは、ルームの作成後、この関数を呼び出して、ルームを破棄できます。



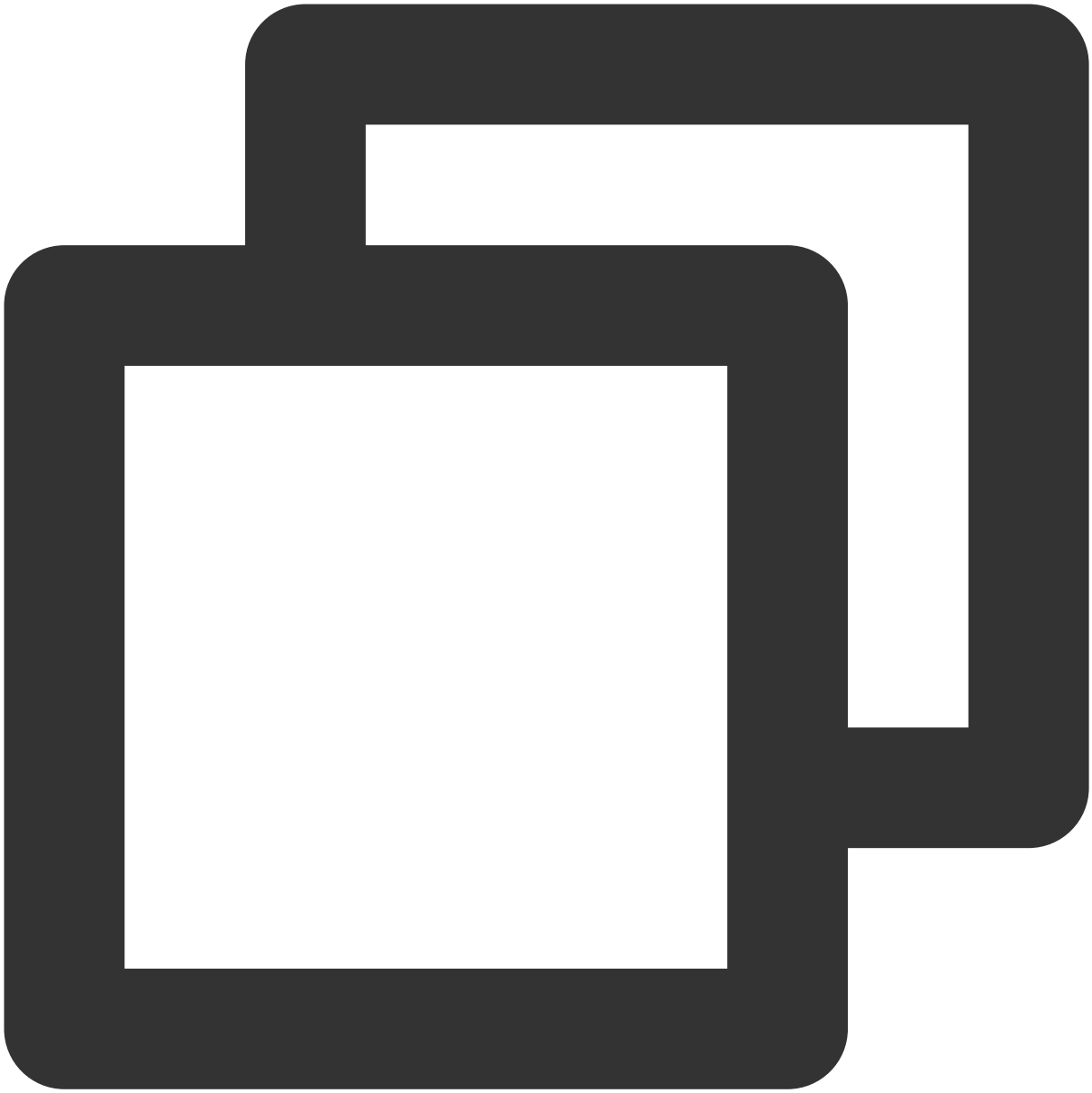
```
void destroyRoom(TUIRoomCoreCallback.ActionCallback callback);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	UIRoomCoreCallback.ActionCallback	ルームの破棄結果のコールバック。

enterRoom

入室（参加者が呼び出し）。



```
void enterRoom(String roomId, TUIRoomCoreCallback.ActionCallback callback);
```

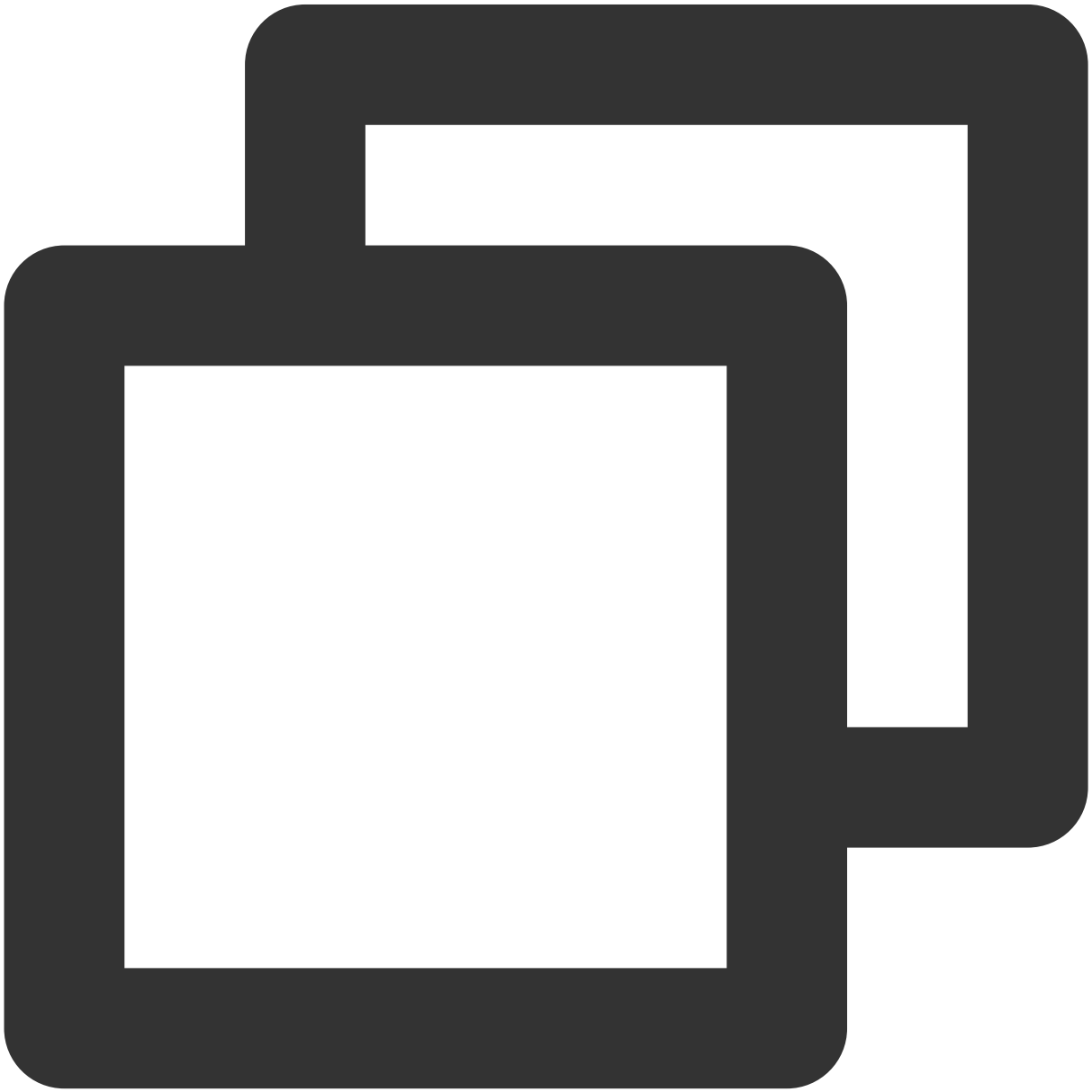
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
roomId	String	ルームID。
callback	UIRoomCoreCallback.ActionCallback	結果のコールバック。

参加者が入室する場合の通常の呼び出し手順は次のとおりです。
1. **参加者**が `enterRoom` を呼び出し、roomIdを渡せば入室できます。
2. **参加者**が `startCameraPreview()` を呼び出して、カメラプレビューを起動し、 `startLocalAudio()` を呼び出して、マイクキャプチャを起動します。
3. **参加者**が `onRemoteUserCameraAvailable` のイベントを受信し、 `startRemoteView()` を呼び出して、ビデオ再生を開始します。

leaveRoom

退室（参加者が呼び出し）。



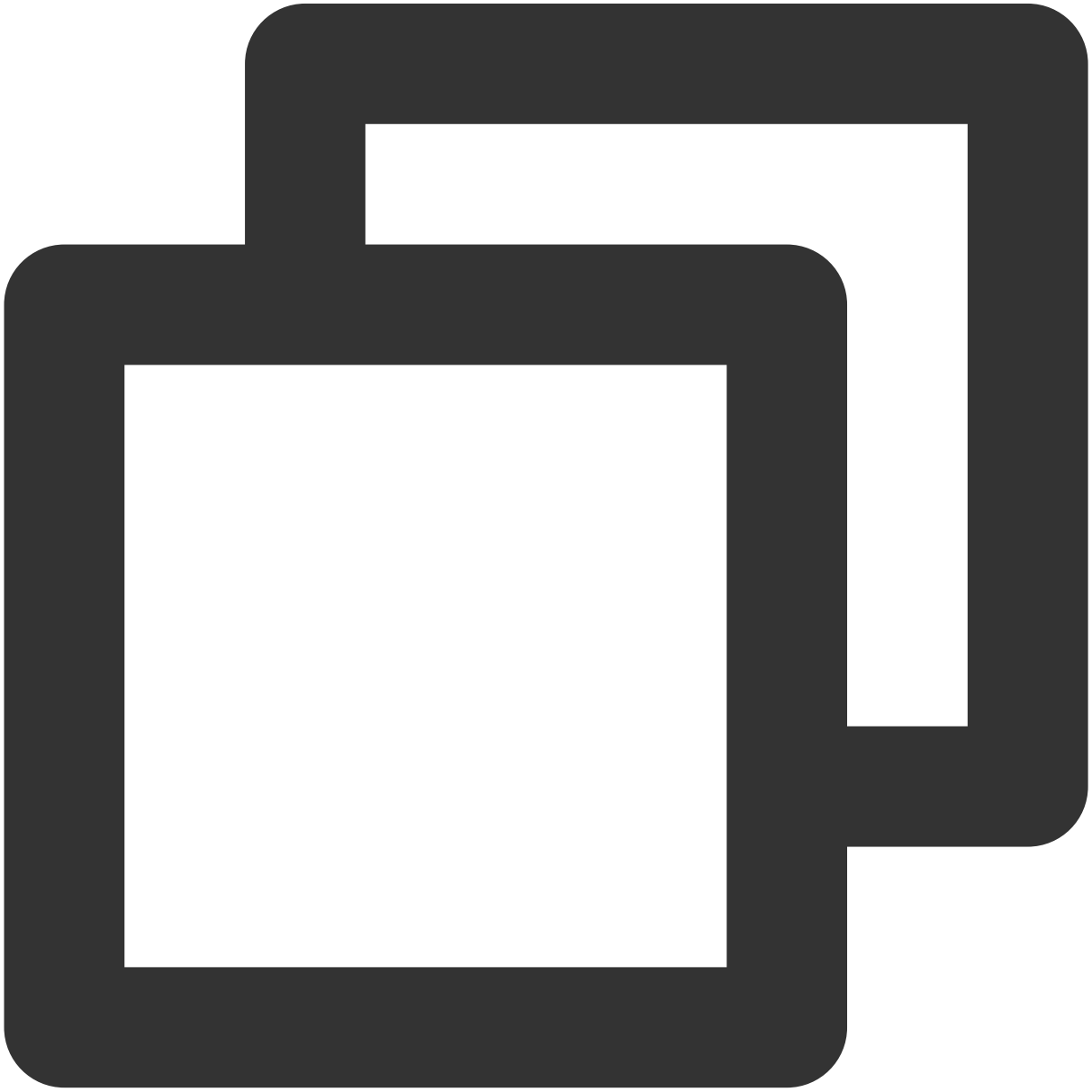
```
void leaveRoom(TUIRoomCoreCallback.ActionCallback callback);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	UIRoomCoreCallback.ActionCallback	結果のコールバック。

getRoomInfo

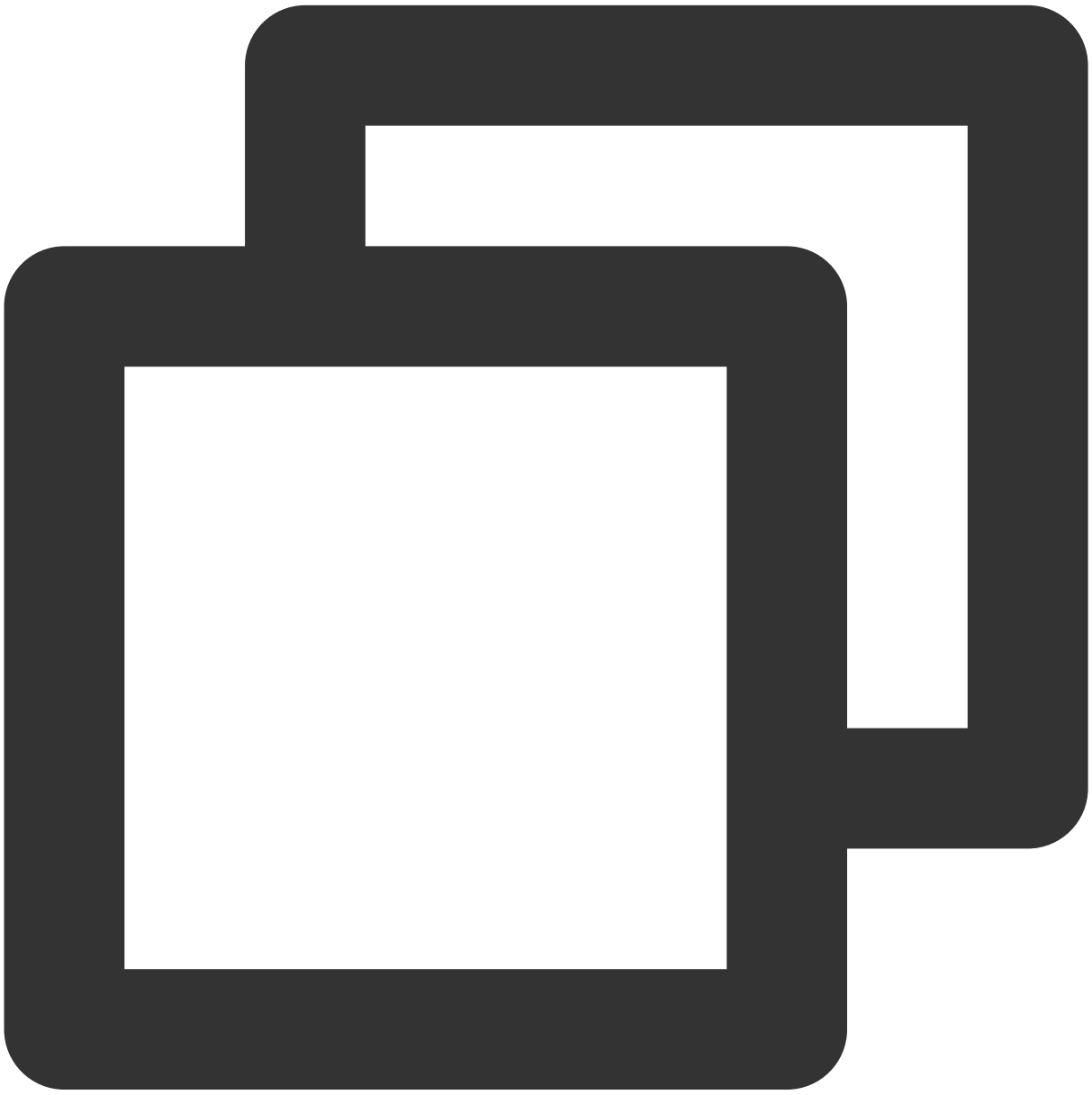
ルーム情報を取得します。



```
TUIRoomCoreDef.RoomInfo getRoomInfo();
```

getRoomUsers

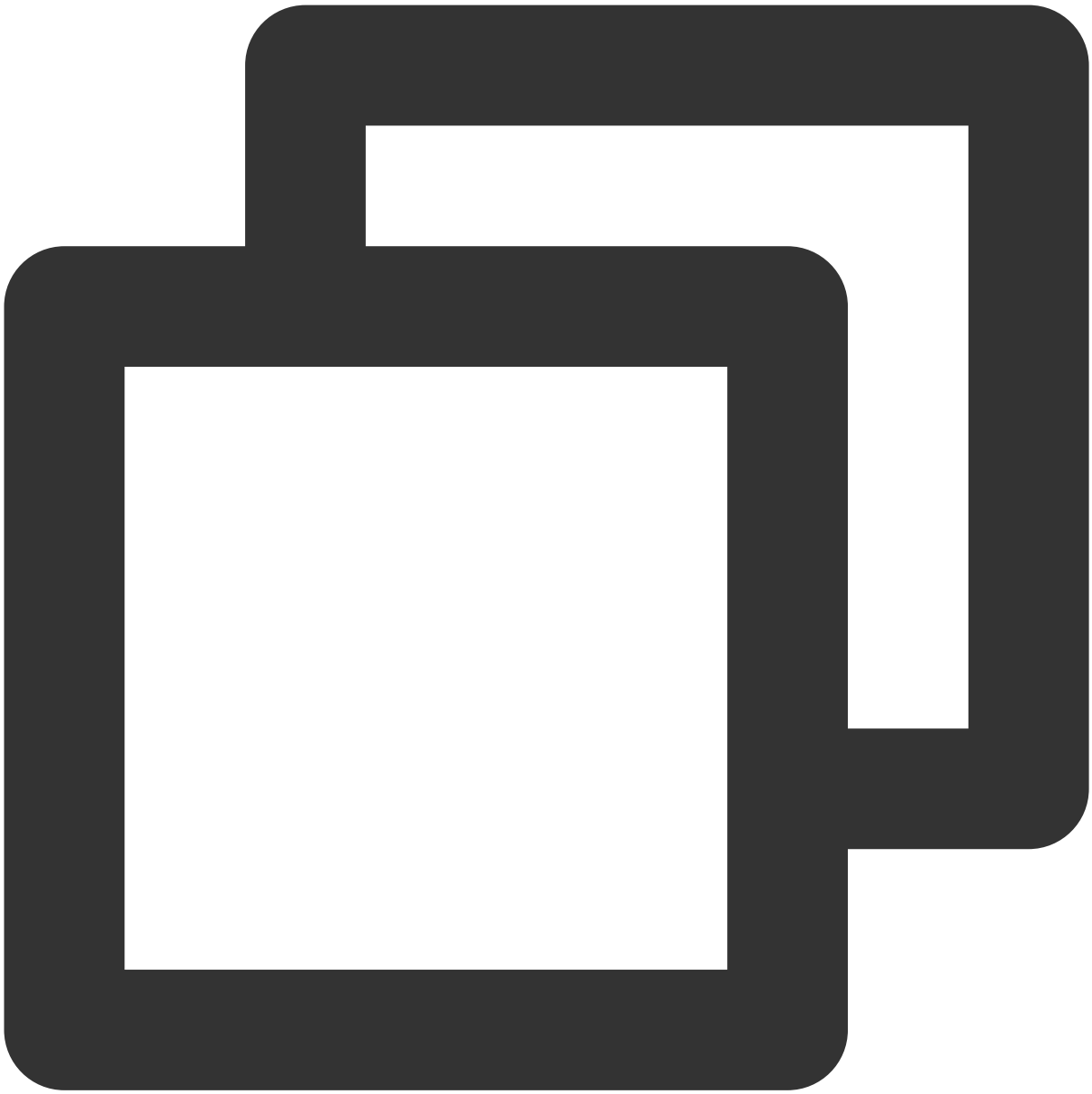
ルームの全メンバー情報を取得します。



```
List<TUIRoomCoreDef.UserInfo> getRoomUsers();
```

getUserInfo

メンバー情報を取得します。



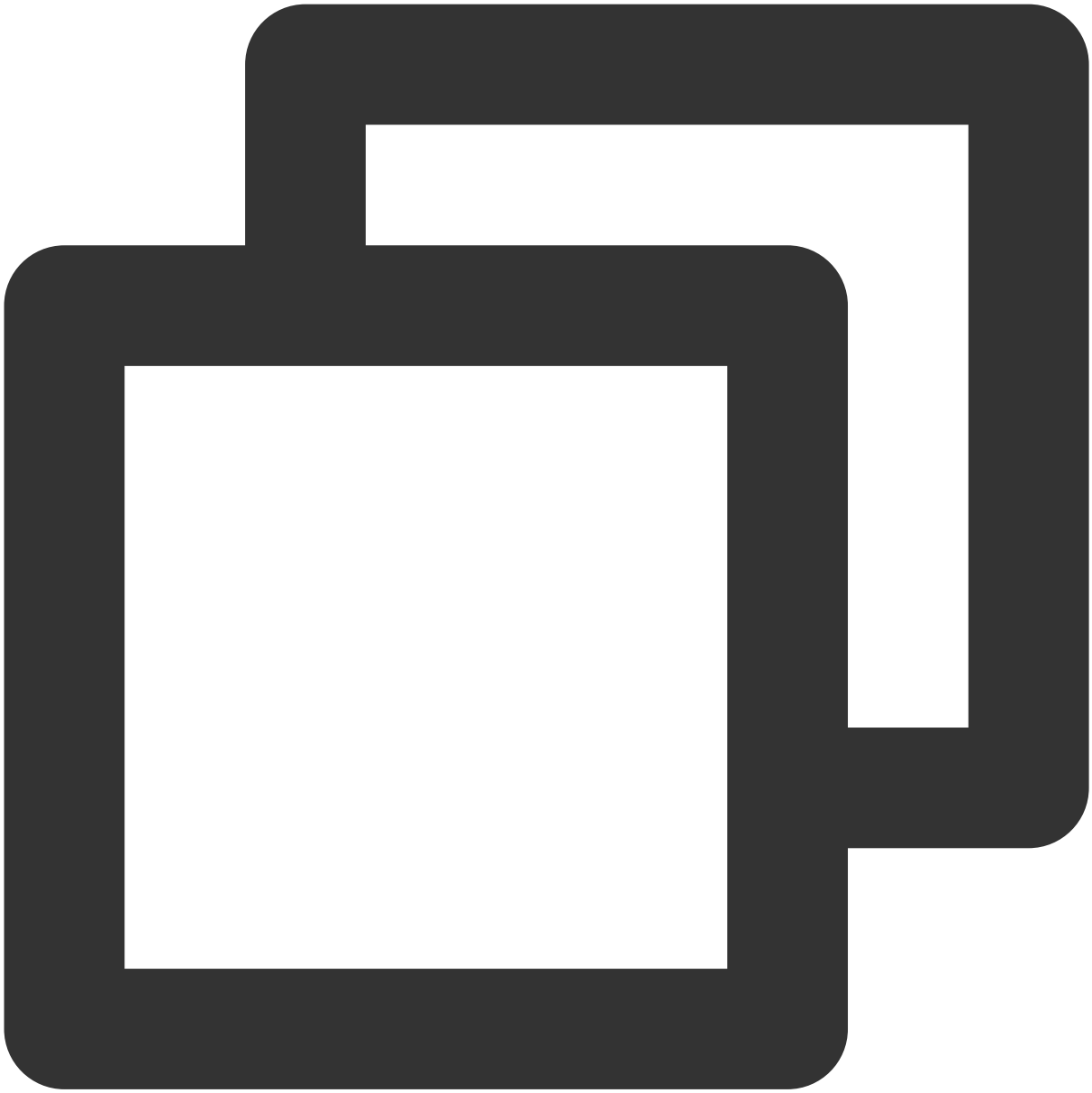
```
void getUserInfo(String userId, TUIRoomCoreCallback.UserInfoCallback callback);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	ユーザーID。
callback	UIRoomCoreCallback.UserInfoCallback	ルームメンバーの詳細情報のコールバック。

setSelfProfile

ユーザー情報を設定します。



```
void setSelfProfile(String userName, String avatarURL, TUIRoomCoreCallback.ActionCa
```

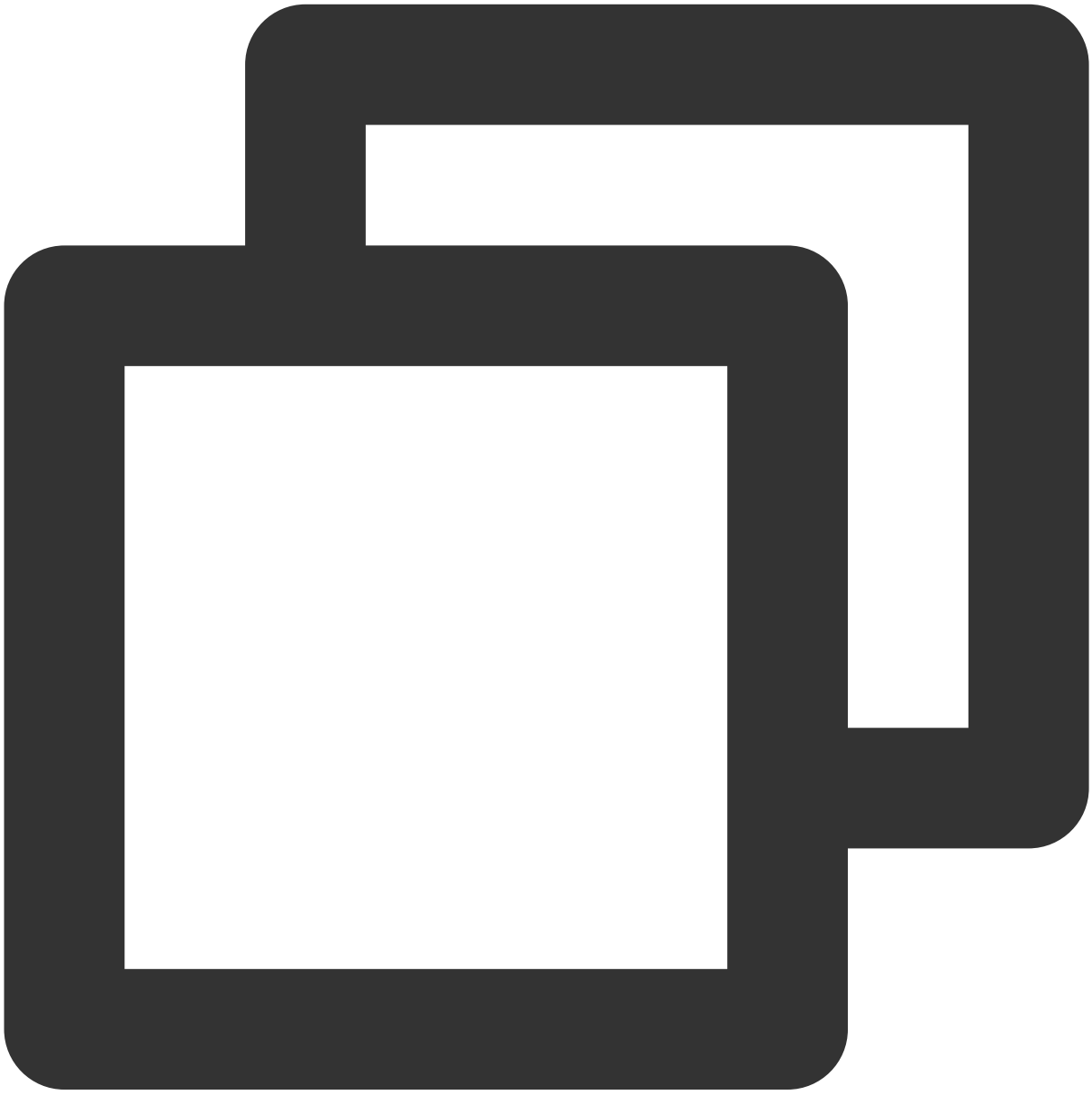
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userName	String	ユーザーの氏名。
avatarURL	String	ユーザーのプロフィール画像URL。

callback	TUIRoomCoreCallback.ActionCallback	設定が成功したかどうかの結果のコールバック。

transferRoomMaster

グループを他のユーザーに引き渡します。



```
void transferRoomMaster(String userId, TUIRoomCoreCallback.ActionCallback callback
```

パラメータは下表に示すとおりです。

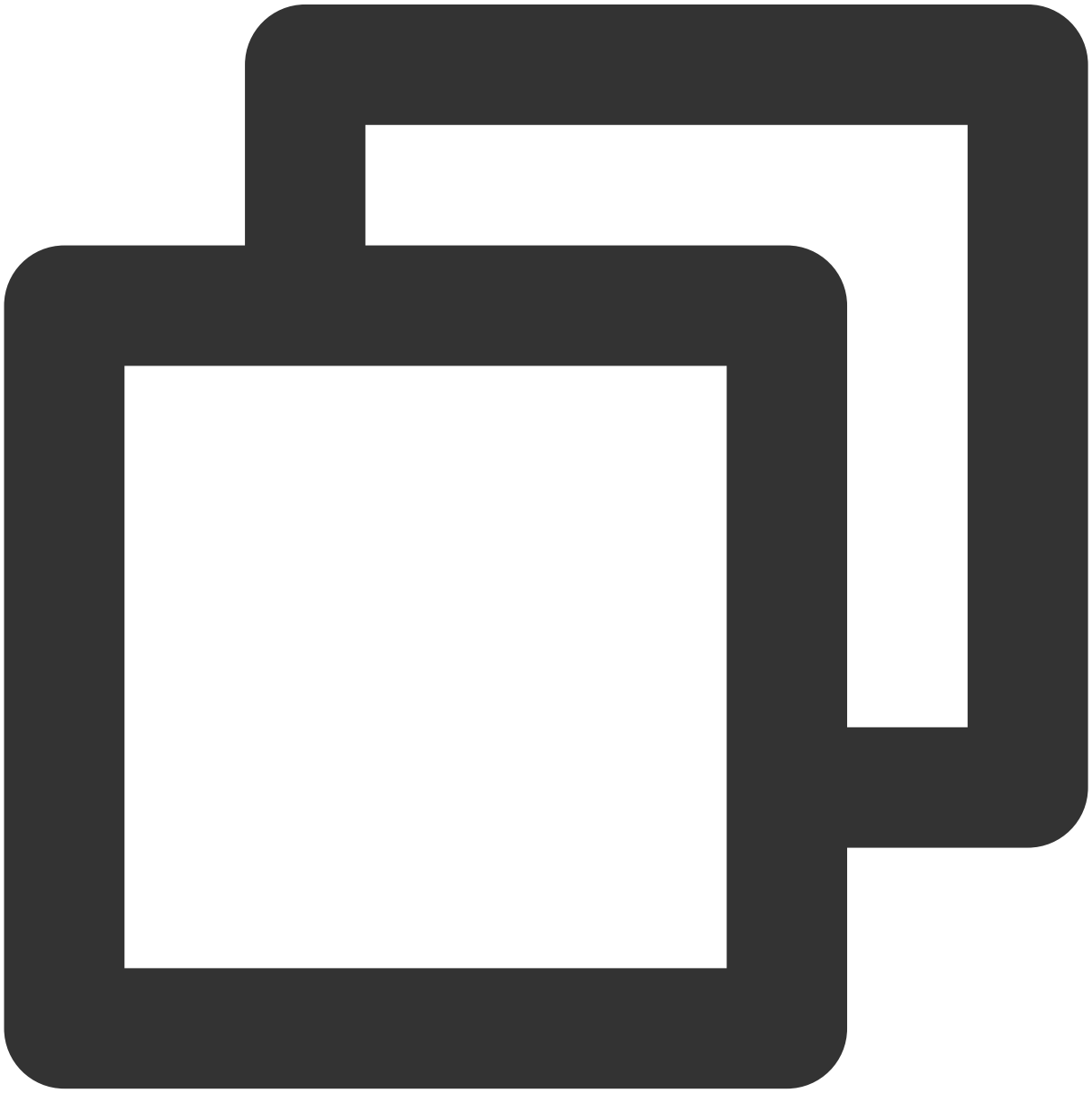
パラメータ	タイプ	意味

userId	String	ユーザーID。
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

ローカルプッシュインターフェース

startCameraPreview

ローカルカメラプレビューを起動します。



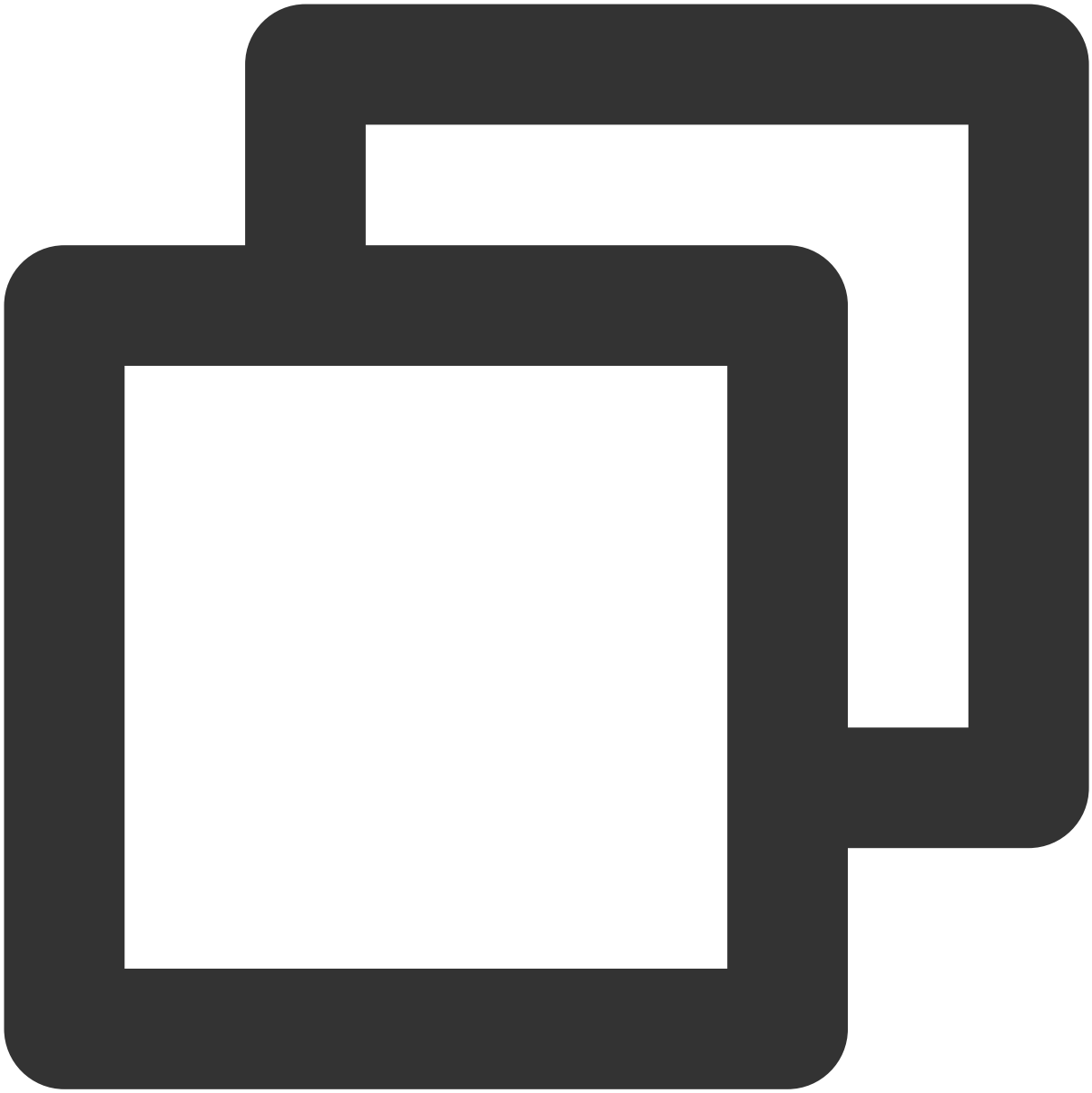
```
void startCameraPreview(boolean isFront, TXCloudVideoView view);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
isFront	boolean	true：フロントカメラ、false：リアカメラ。
view	TXCloudVideoView	ビデオ画像をロードするウィジェット。

stopCameraPreview

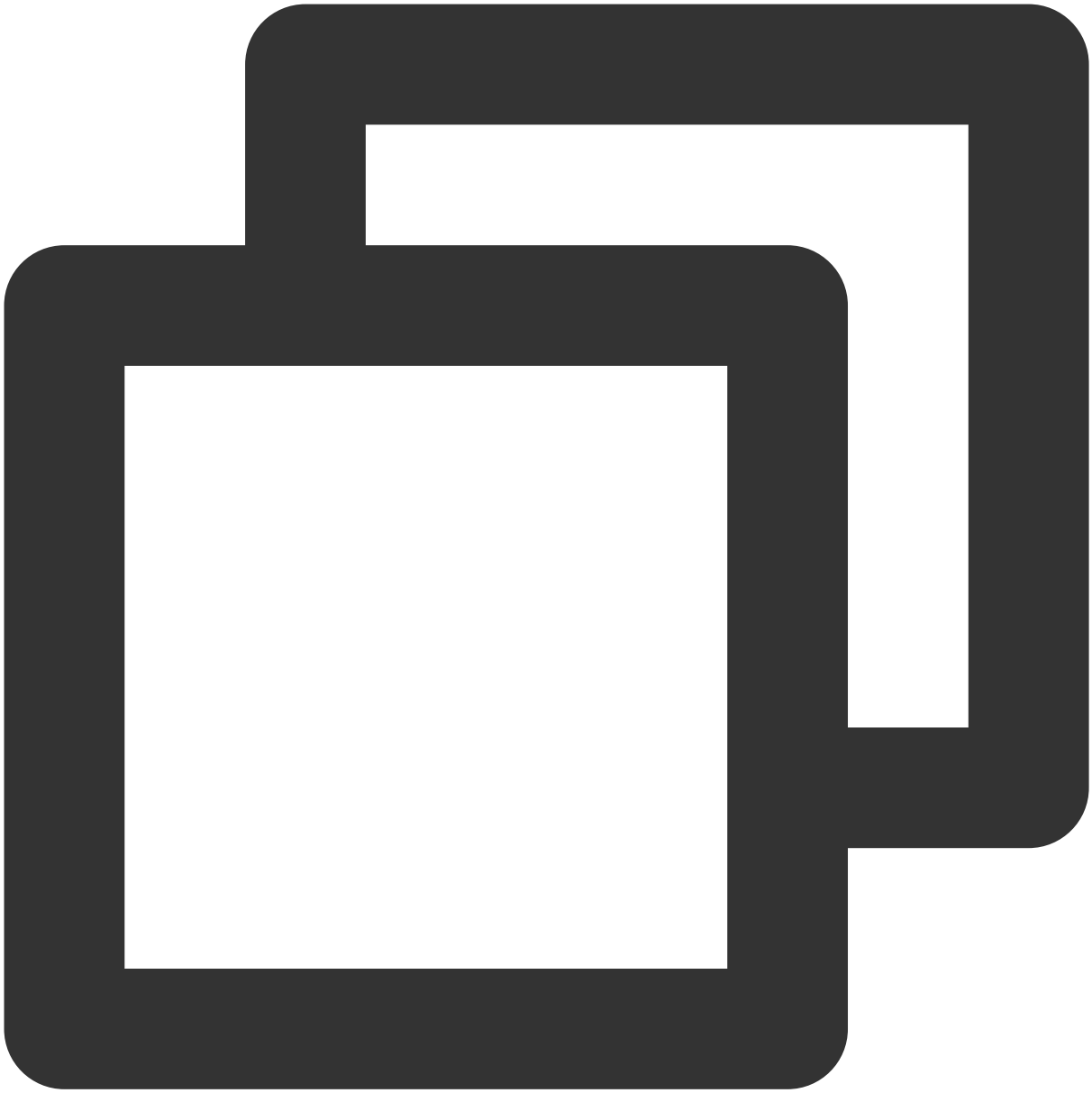
ローカルカメラプレビューを停止します。



```
void stopCameraPreview();
```

startLocalAudio

マイクの集音開始。



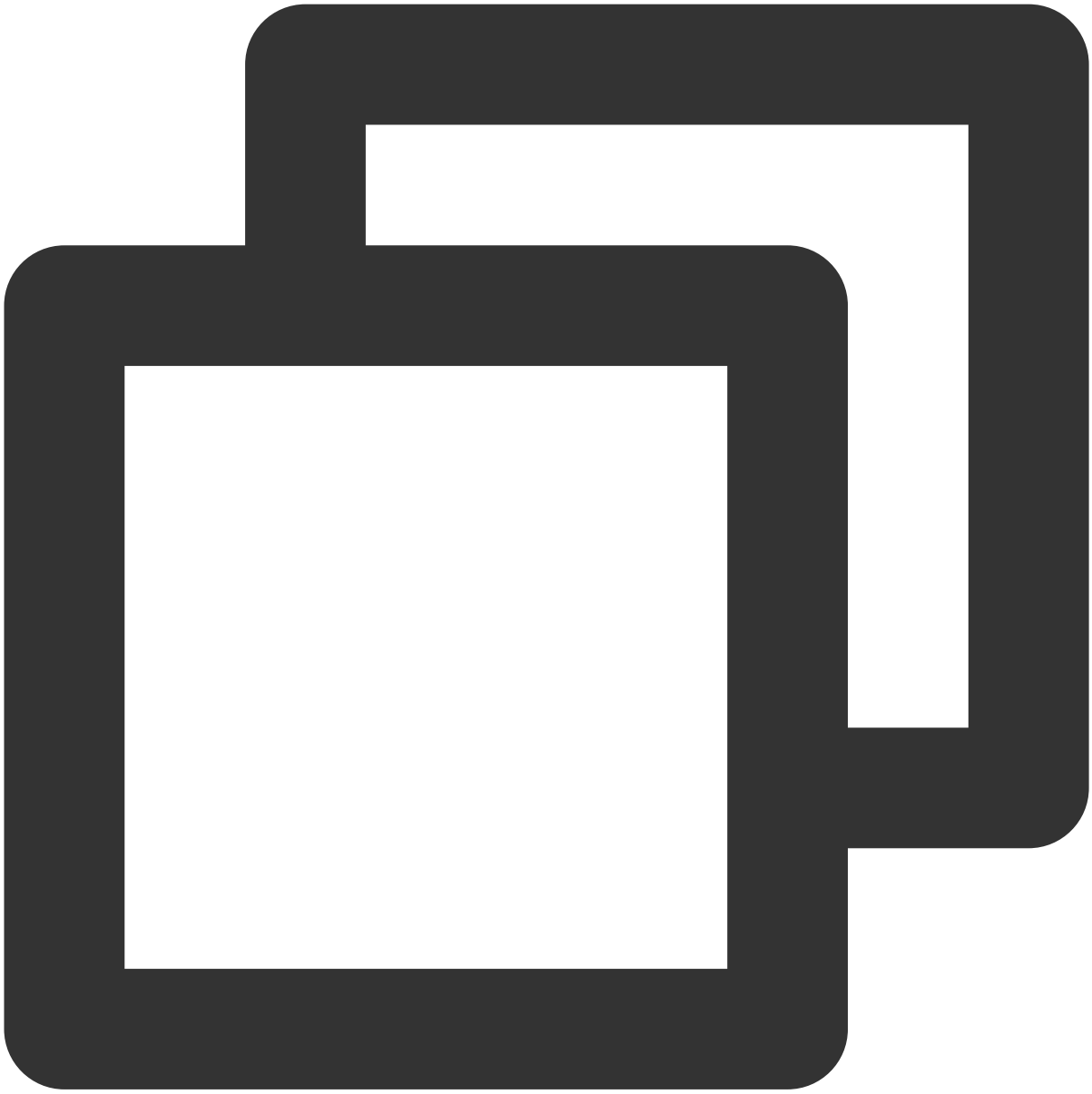
```
void startLocalAudio(int quality);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
quality	int	キャプチャの音質： TRTC_AUDIO_QUALITY_MUSIC TRTC_AUDIO_QUALITY_DEFAULT TRTC_AUDIO_QUALITY_SPEECH

stopLocalAudio

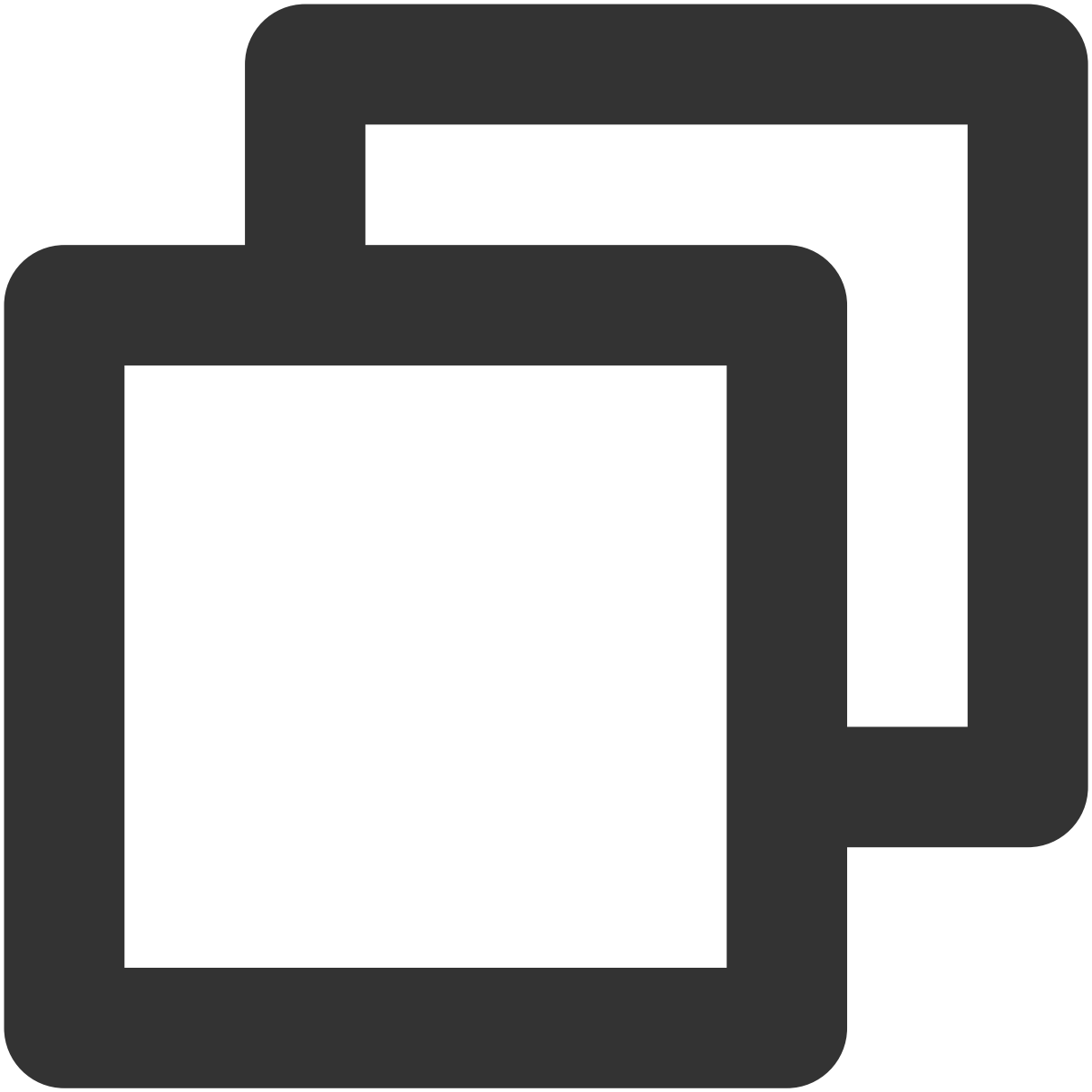
マイクの集音停止



```
void stopLocalAudio();
```

setVideoMirror

ローカル画面のイメージプレビューモードを設定します。



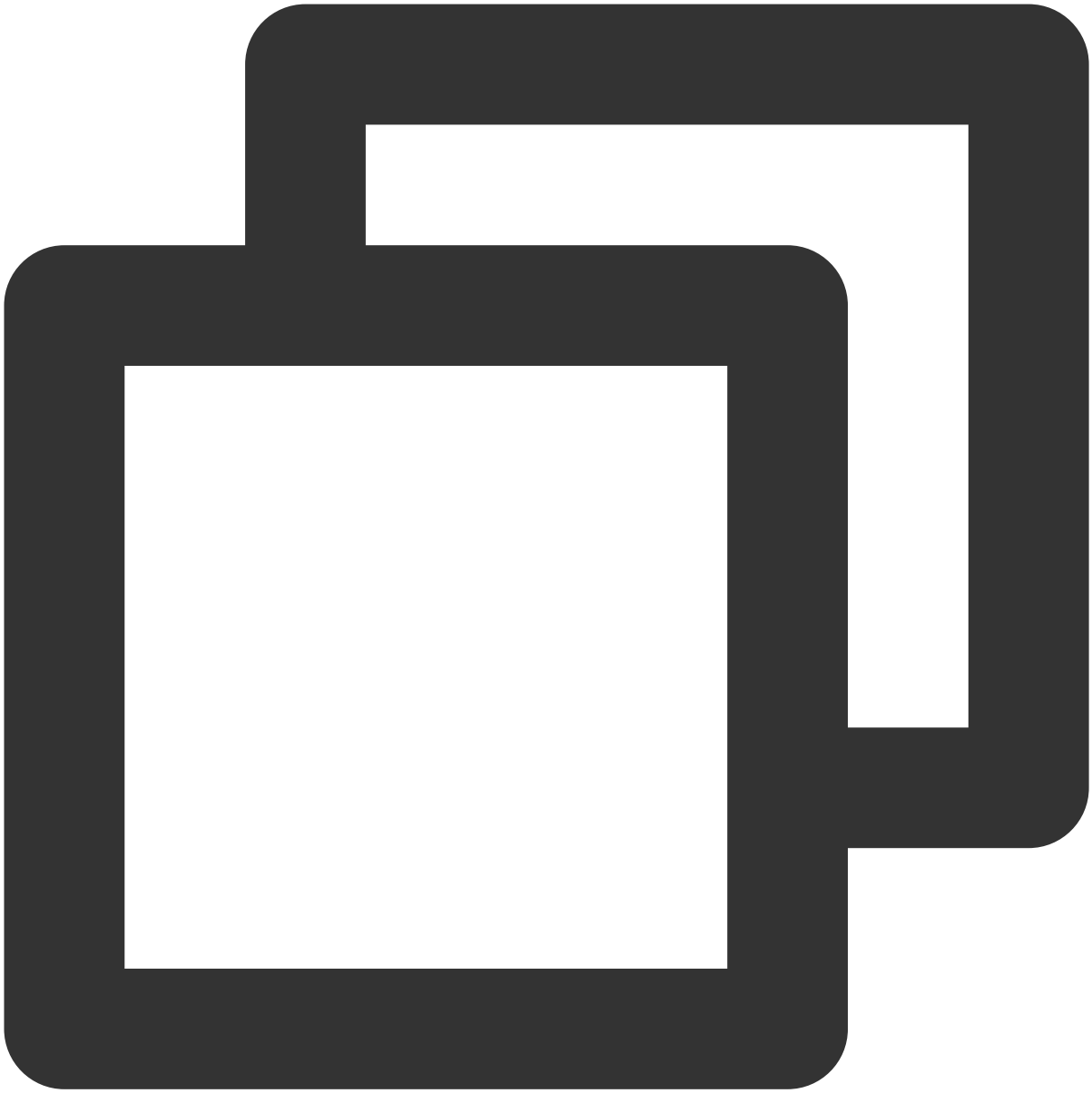
```
void setVideoMirror(int type);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
type	int	イメージタイプ。

setSpeaker

スピーカーの起動設定。



```
void setSpeaker(boolean isUseSpeaker);
```

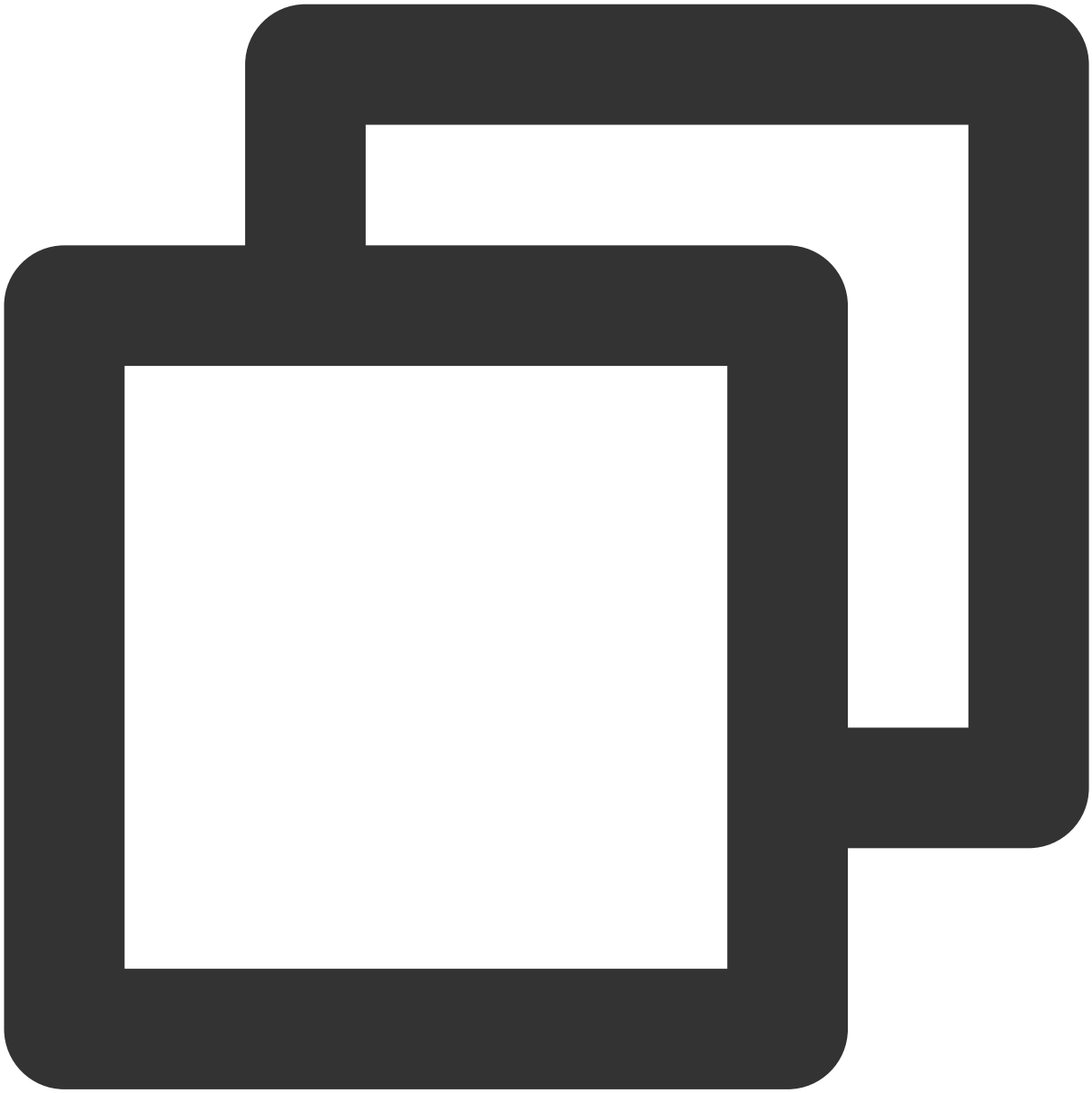
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
isUseSpeaker	boolean	true：スピーカー、false：ヘッドホン。

リモートユーザーに関するインターフェース

startRemoteView

リモートユーザーのビデオストリームのサブスクリプション。



```
void startRemoteView(String userId, TXCloudVideoView view, TUIRoomCoreDef.SteamType
```

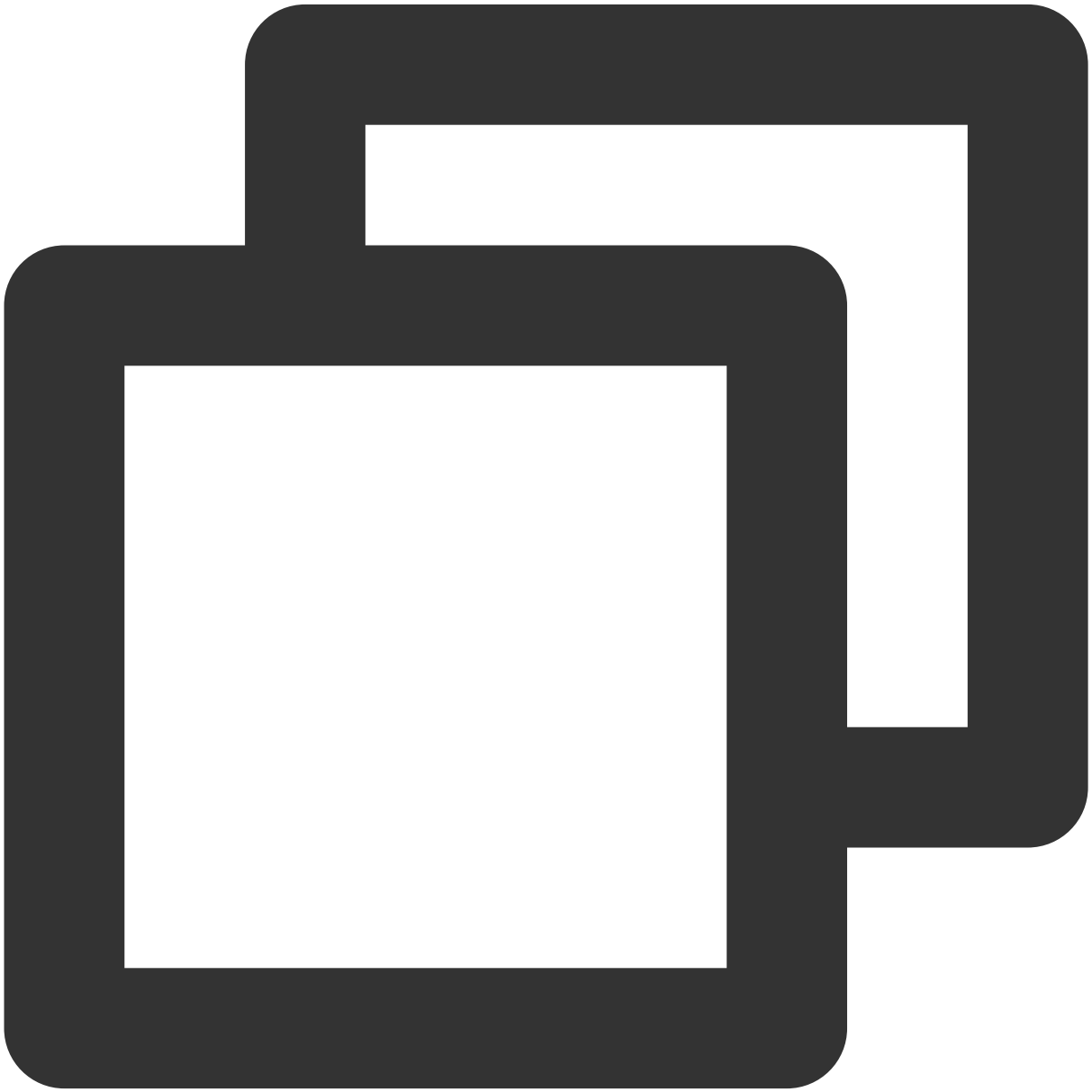
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	再生が必要なユーザーのID。
view	TXCloudVideoView	ビデオ画像をロードするviewウィジェット。

streamType	TUIRoomCoreDef.SteamType	ストリームのタイプ。
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

stopRemoteView

サブスクリプションをキャンセルし、リモートビデオ画面の再生を停止します。



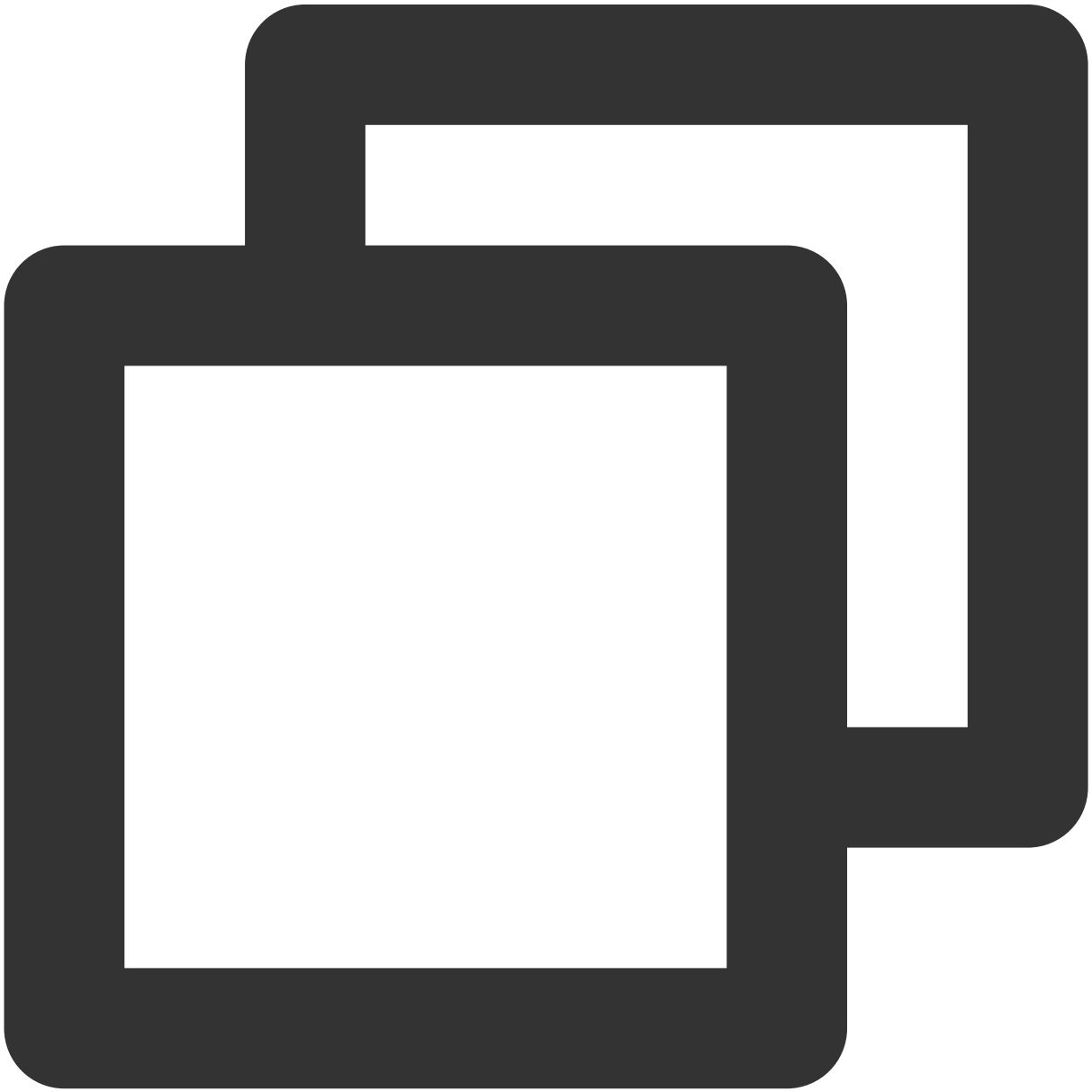
```
void stopRemoteView(String userId, TUIRoomCoreCallback.ActionCallback callback);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	再生停止が必要なユーザーのID。
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

switchCamera

フロント/リアカメラを切り替えます。



```
void switchCamera(boolean isFront);
```

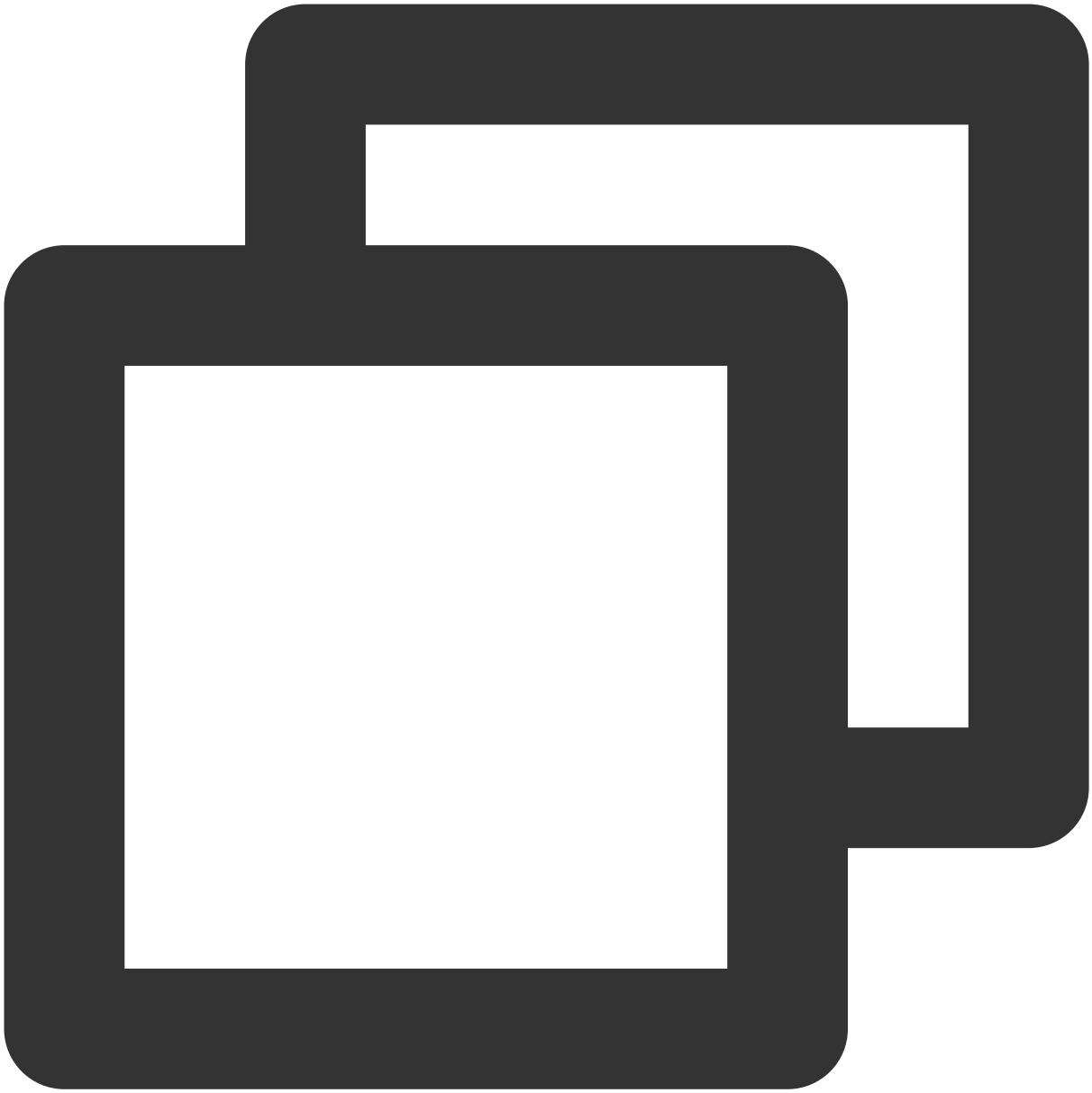
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
isFront	boolean	true：フロントカメラ、false：リアカメラ。

メッセージ送信インターフェース

sendChatMessage

ルーム内でテキストメッセージをブロードキャストします。通常、テキストによるチャットに使用します。



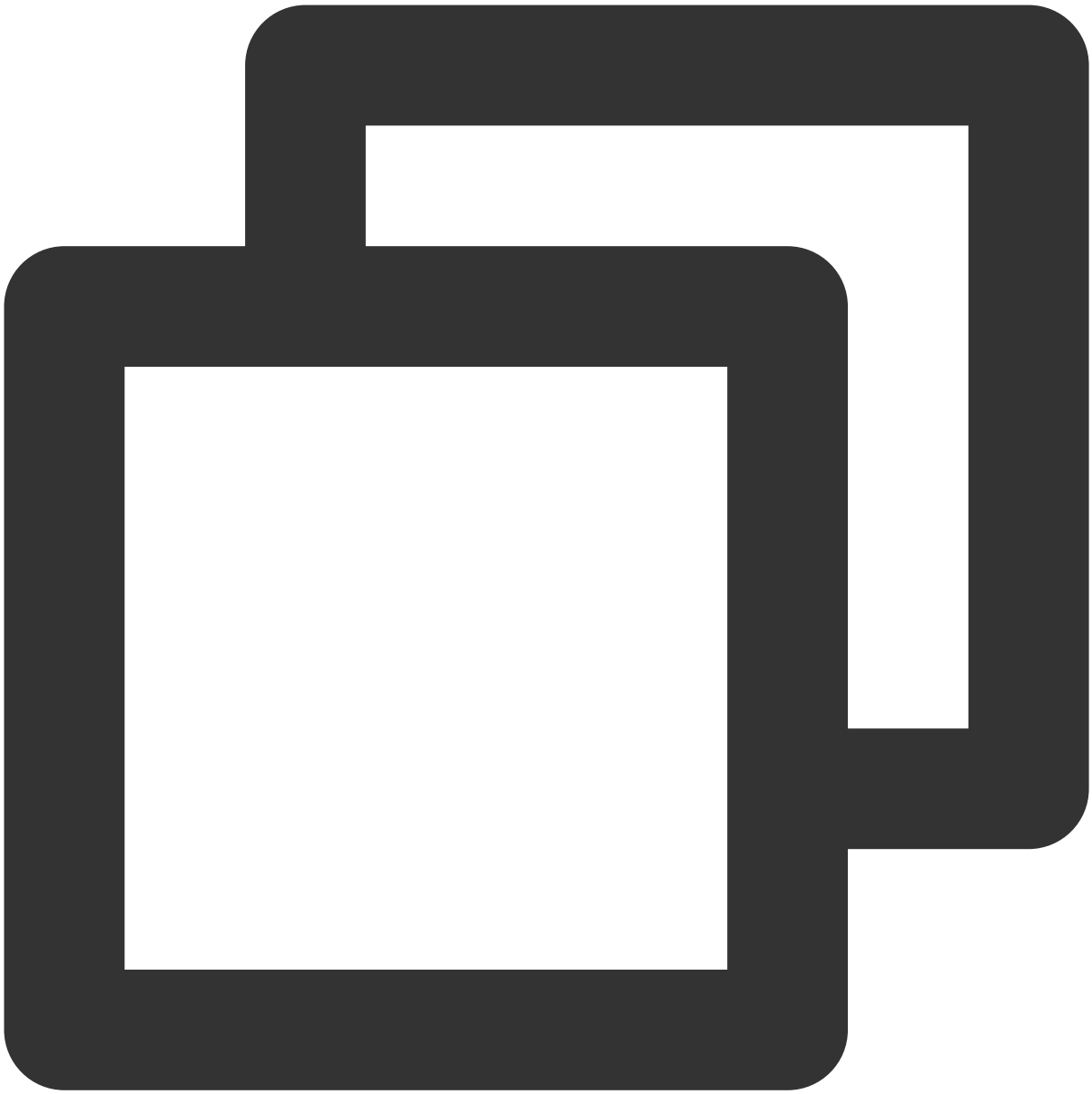
```
void sendChatMessage(String message, TUIRoomCoreCallback.ActionCallback callback);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
message	String	メッセージの内容。
callback	TUIRoomCoreCallback.ActionCallback	送信結果のコールバック。

sendCustomMessage

カスタムメッセージを送信します。



```
void sendCustomMessage(String data, TUIRoomCoreCallback.ActionCallback callback);
```

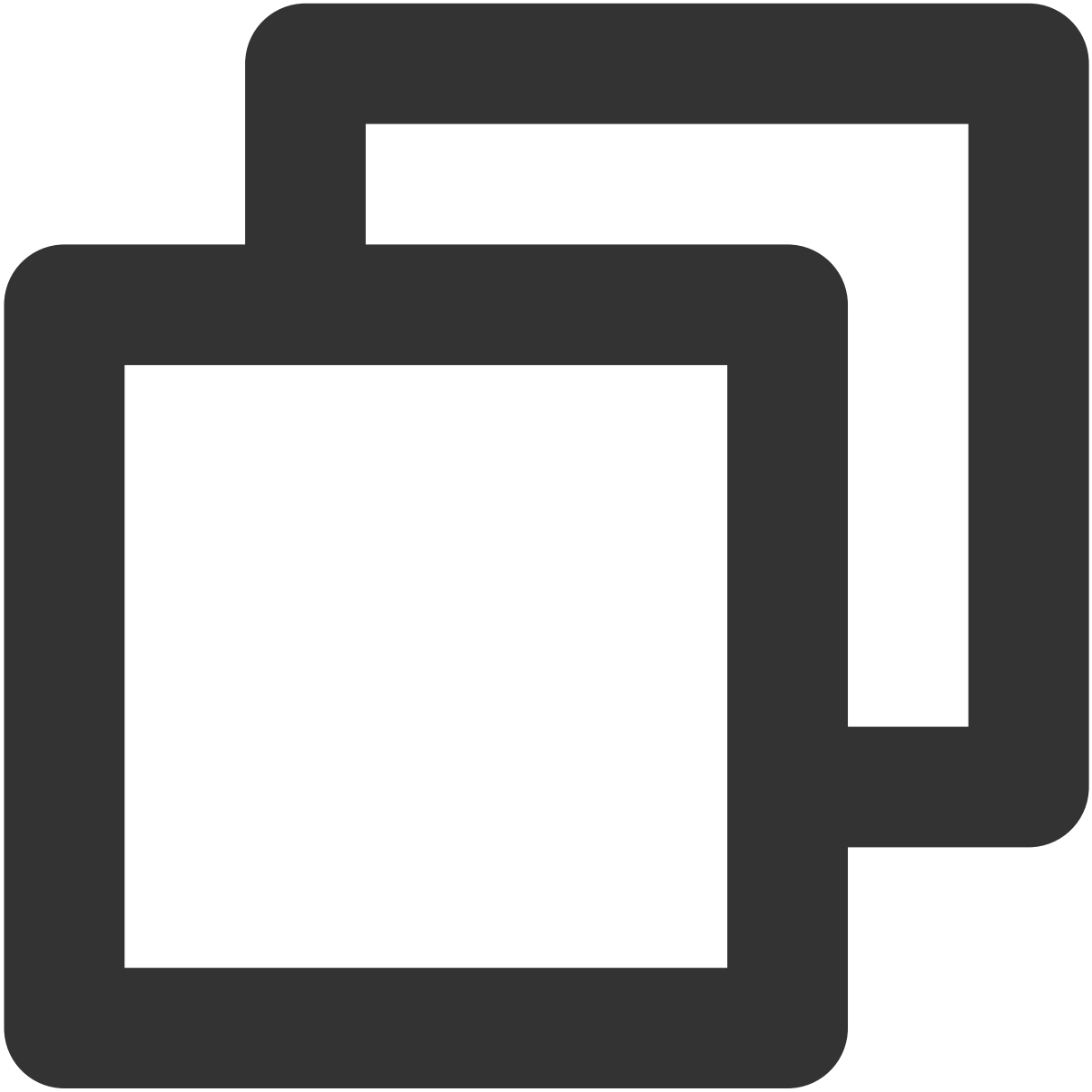
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
data	String	メッセージの内容。
callback	TUIRoomCoreCallback.ActionCallback	送信結果のコールバック。

フィールドコントロール関連インターフェース

muteUserMicrophone

特定ユーザーのマイクを無効化/再有効化します。



```
void muteUserMicrophone(String userId, boolean mute, TUIRoomCoreCallback.ActionCall
```

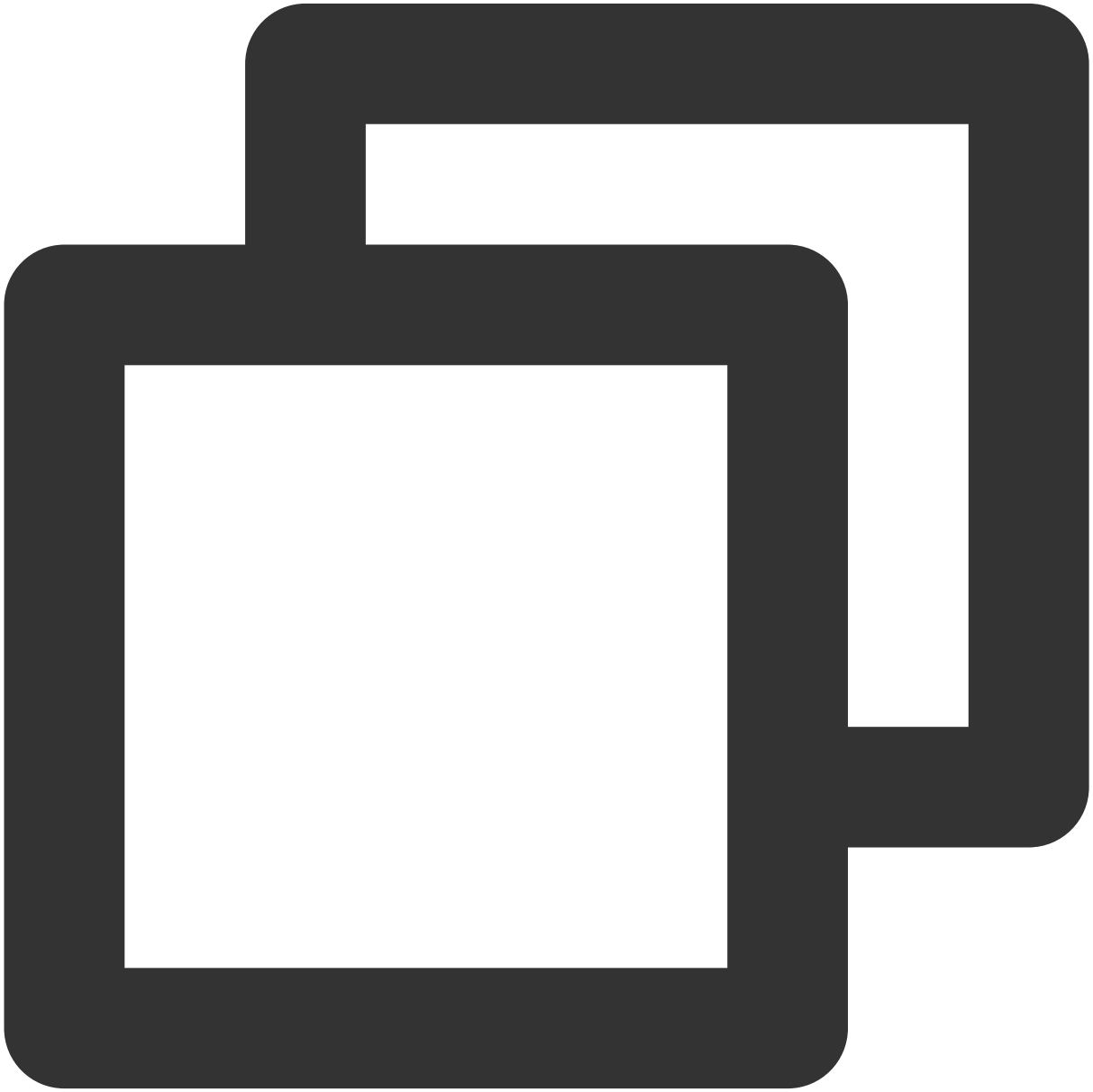
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	String	ユーザーID。
mute	boolean	無効にするかどうか。
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

muteAllUsersMicrophone

全ユーザーのマイクを無効化/再有効化します。



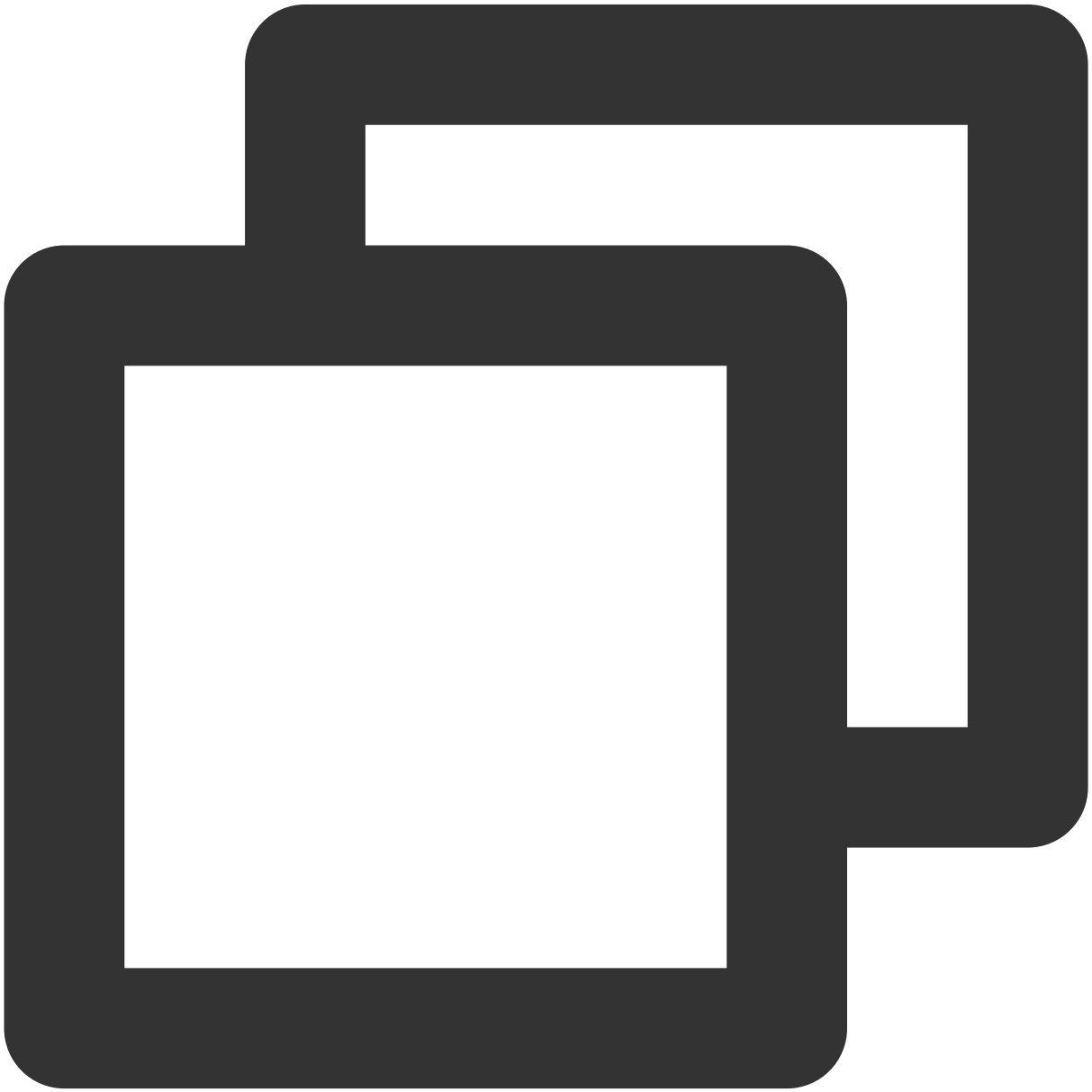
```
void muteAllUsersMicrophone(boolean mute, TUIRoomCoreCallback.ActionCallback callba
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
mute	boolean	無効にするかどうか。
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

muteUserCamera

特定ユーザーのカメラを無効化/再有効化します。



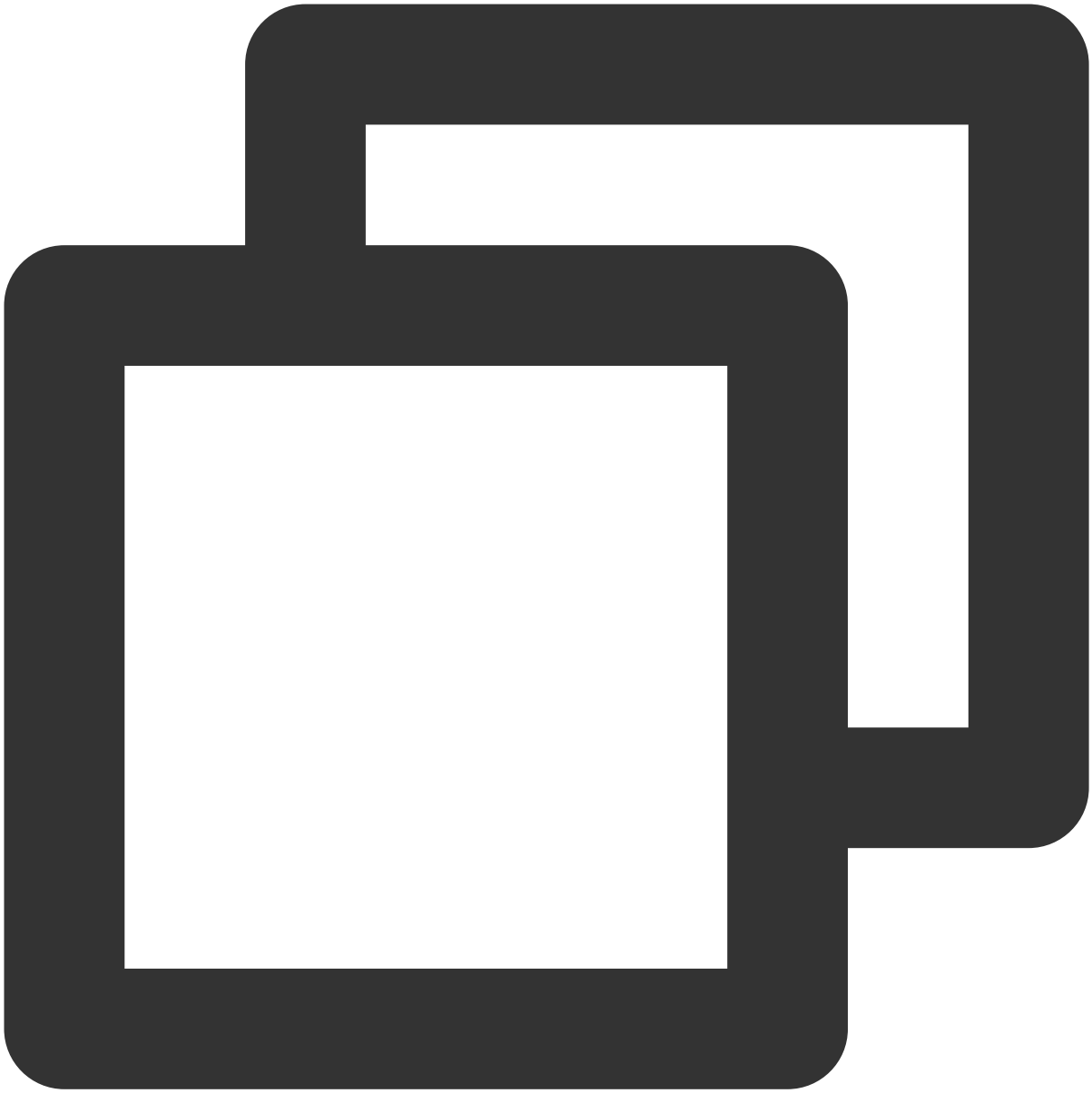
```
void muteUserCamera(String userId, boolean mute, TUIRoomCoreCallback.ActionCallback
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	ユーザーID。
mute	boolean	無効にするかどうか。
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

muteAllUsersCamera

全ユーザーのカメラを無効化/再有効化します。



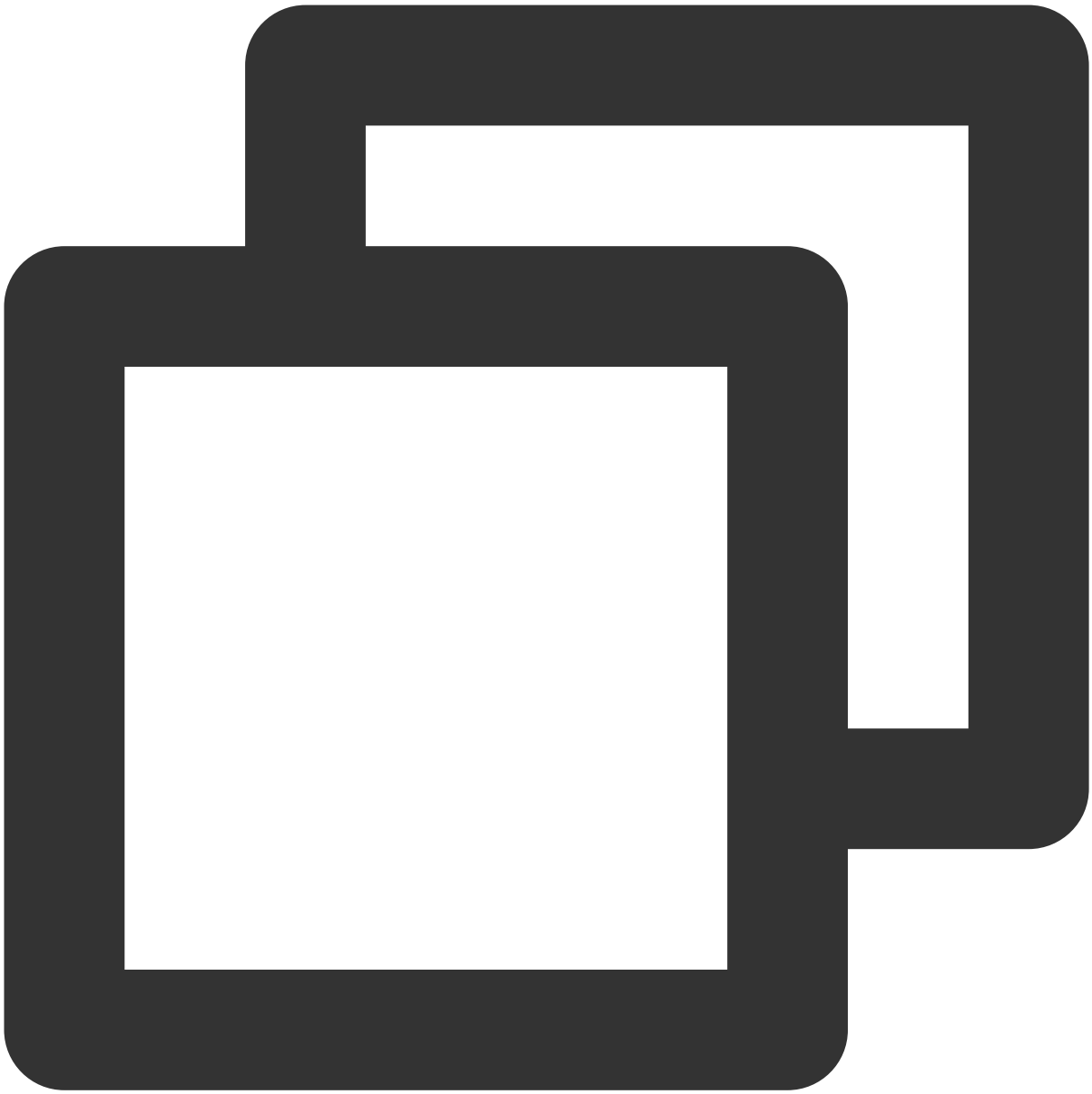
```
void muteAllUsersCamera(boolean mute, TUIRoomCoreCallback.ActionCallback callback);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
mute	boolean	無効にするかどうか。
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

muteChatRoom

テキストチャットのミュート/再有効化。



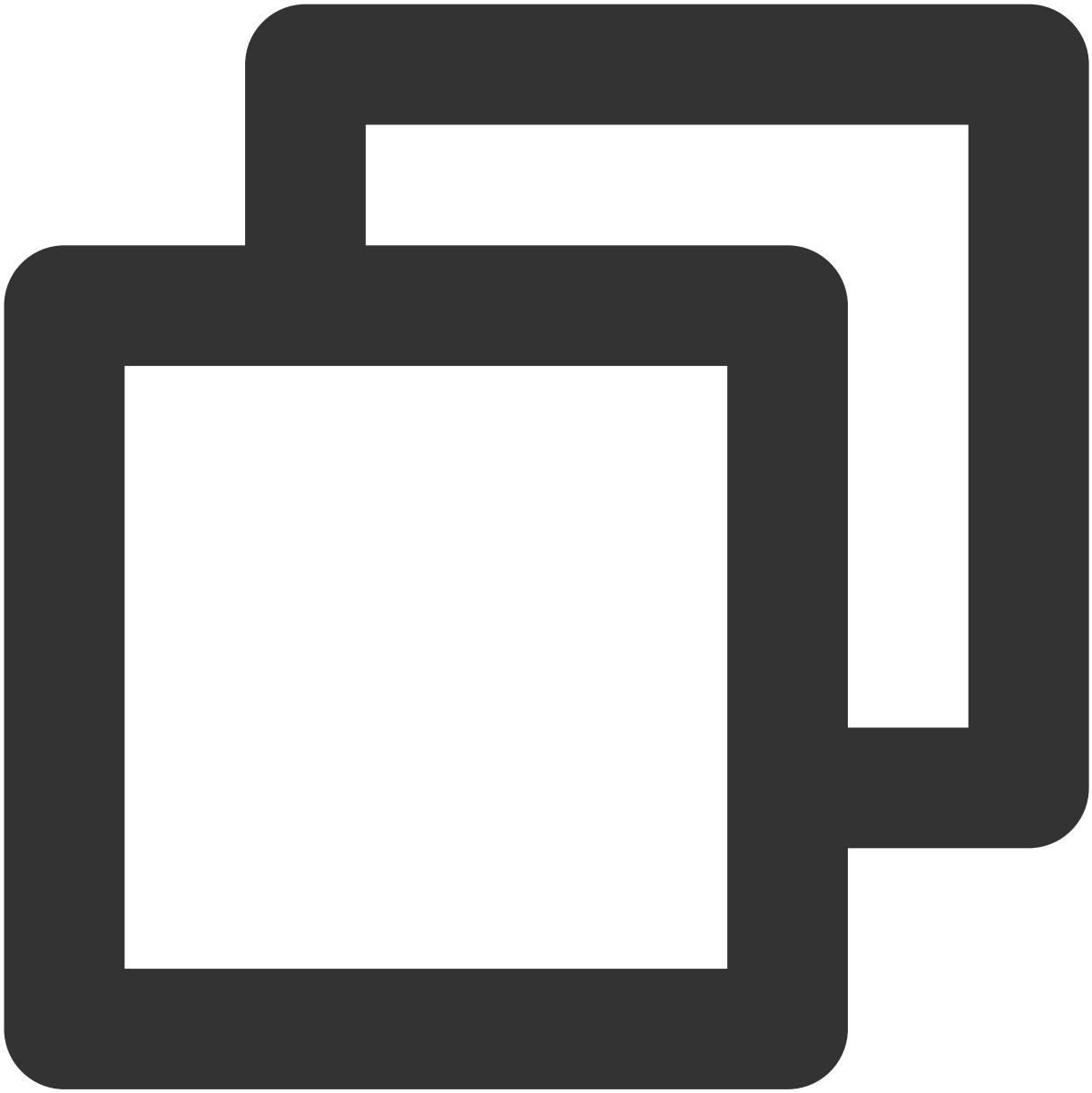
```
void muteChatRoom(boolean mute, TUIRoomCoreCallback.ActionCallback callback);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
mute	boolean	無効にするかどうか。
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

kickOffUser

キャスターがキックアウトします。



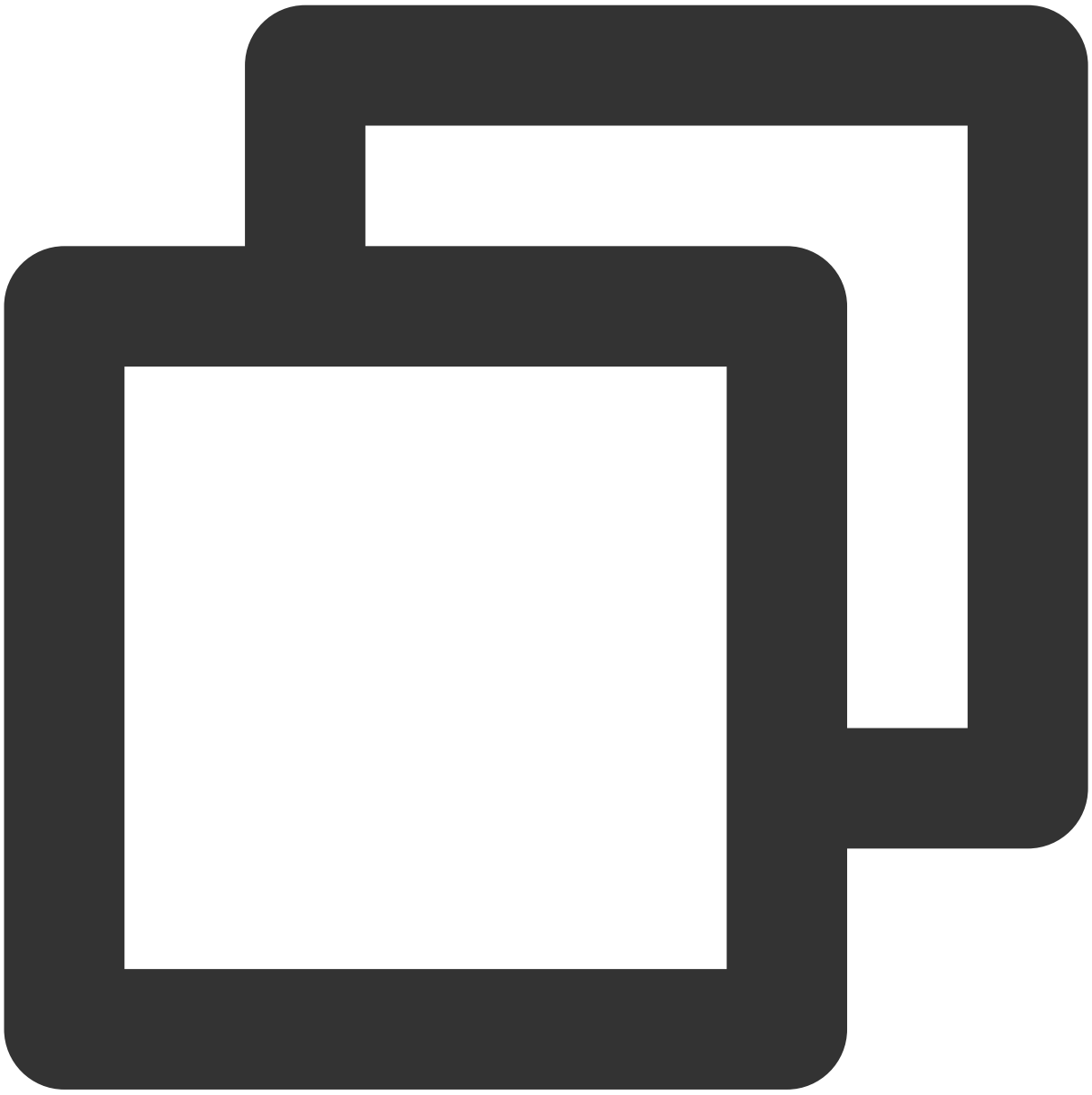
```
void kickOffUser(String userId, TUIRoomCoreCallback.ActionCallback callback);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	ユーザーID。
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

startCallingRoll

キャスターが点呼を開始します。



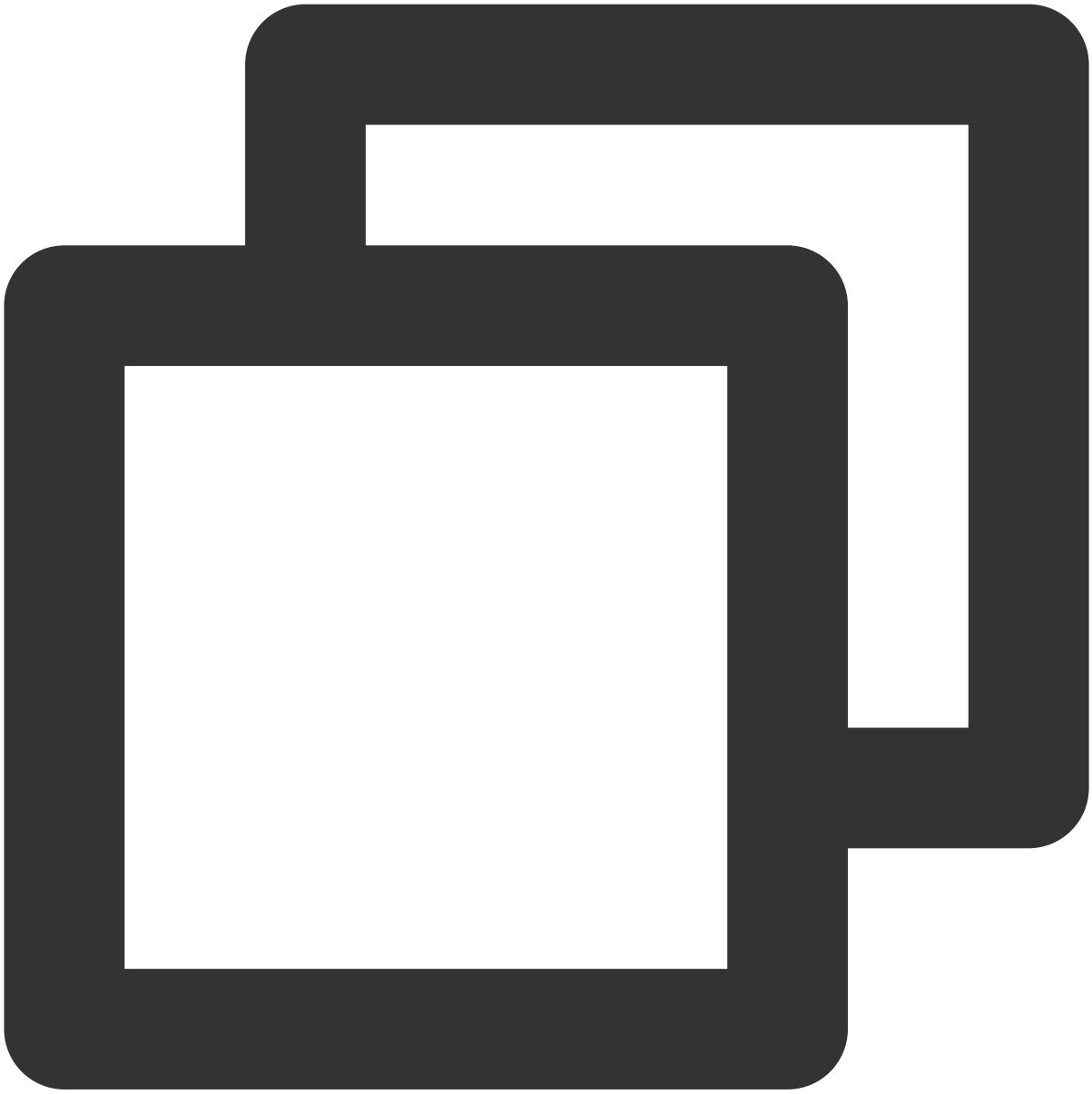
```
void startCallingRoll(TUIRoomCoreCallback.ActionCallback callback);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

stopCallingRoll

キャスターが点呼を終了します。



```
void stopCallingRoll(TUIRoomCoreCallback.ActionCallback callback);
```

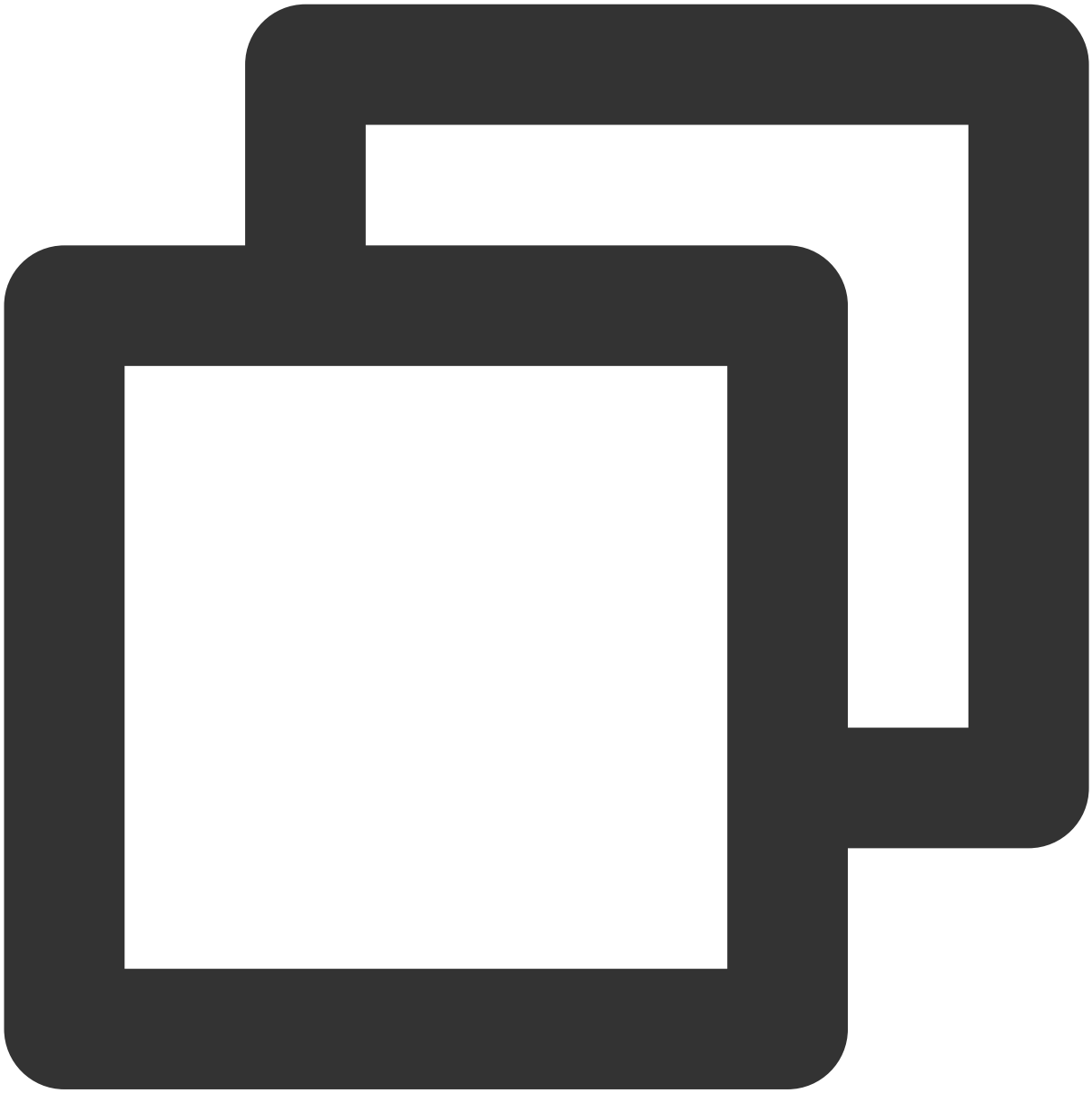
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

replyCallingRoll

参加者がキャスターの点呼に応答します。



```
void replyCallingRoll(TUIRoomCoreCallback.ActionCallback callback);
```

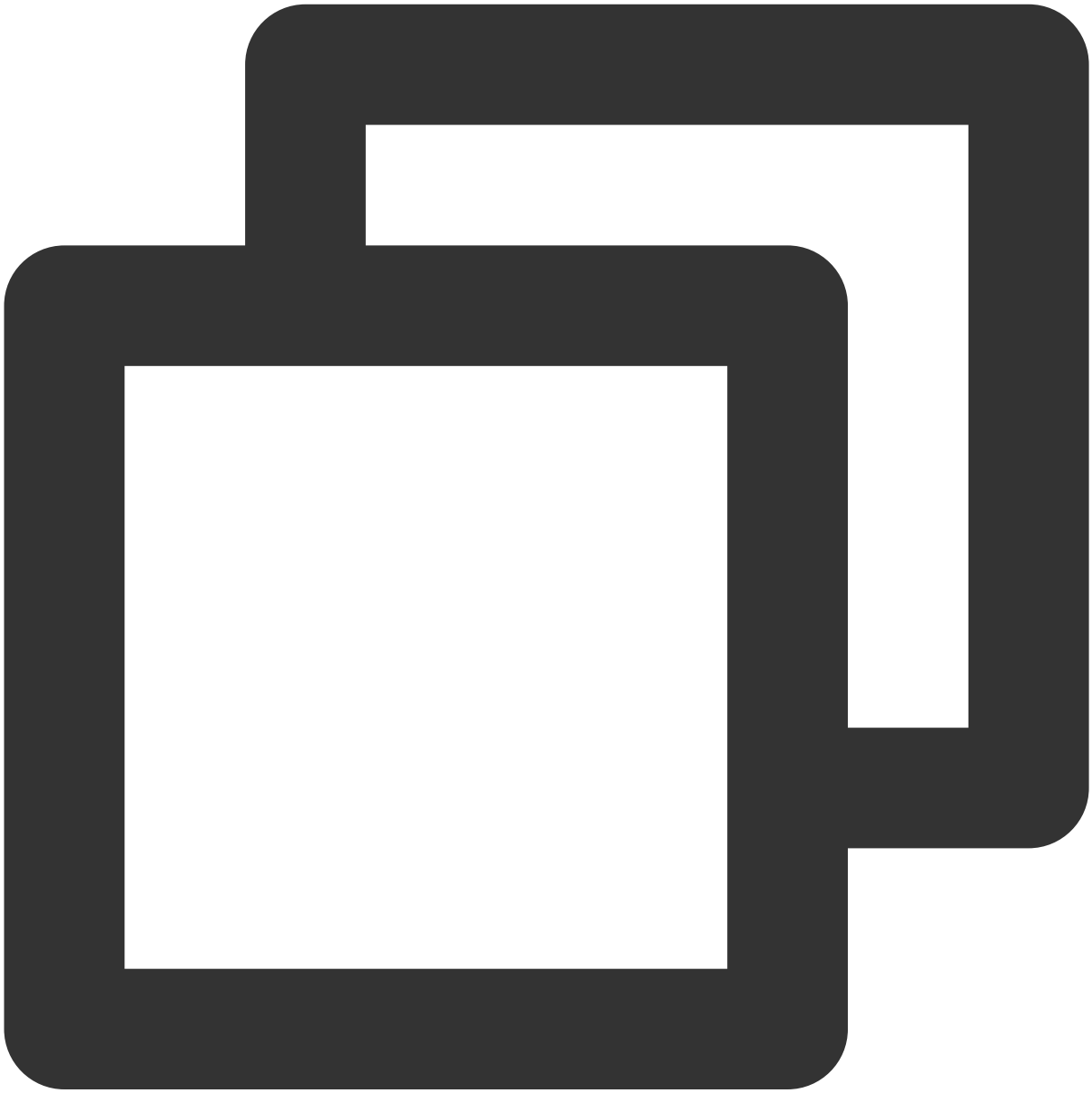
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

sendSpeechInvitation

キャスターが参加者の発言を要請します。



```
void sendSpeechInvitation(String userId, TUIRoomCoreCallback.InvitationCallback cal
```

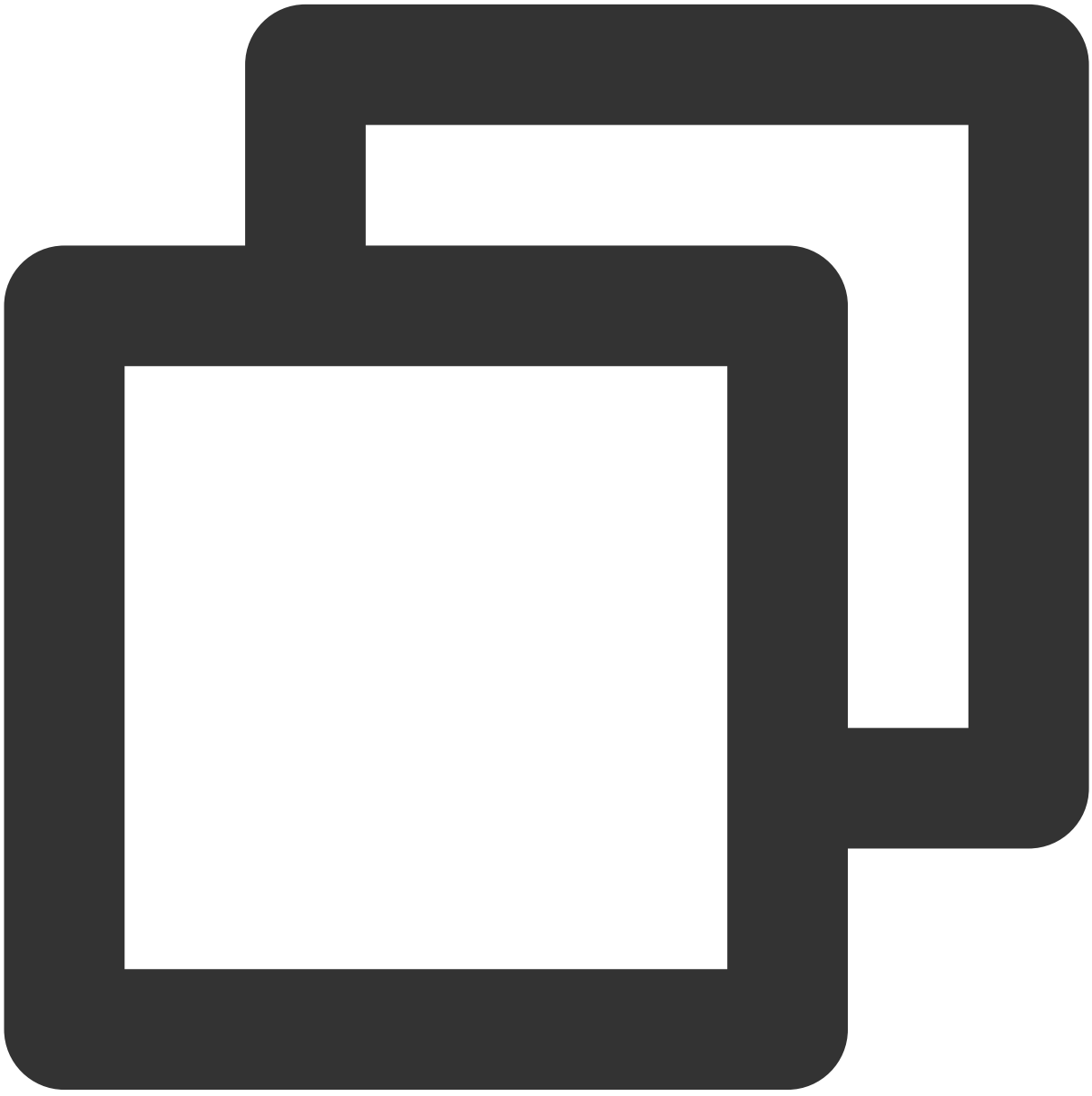
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	String	ユーザーID。
callback	TUIRoomCoreCallback.InvitationCallback	結果のコールバック。

cancelSpeechInvitation

キャスターが参加者の発言要請をキャンセルします。



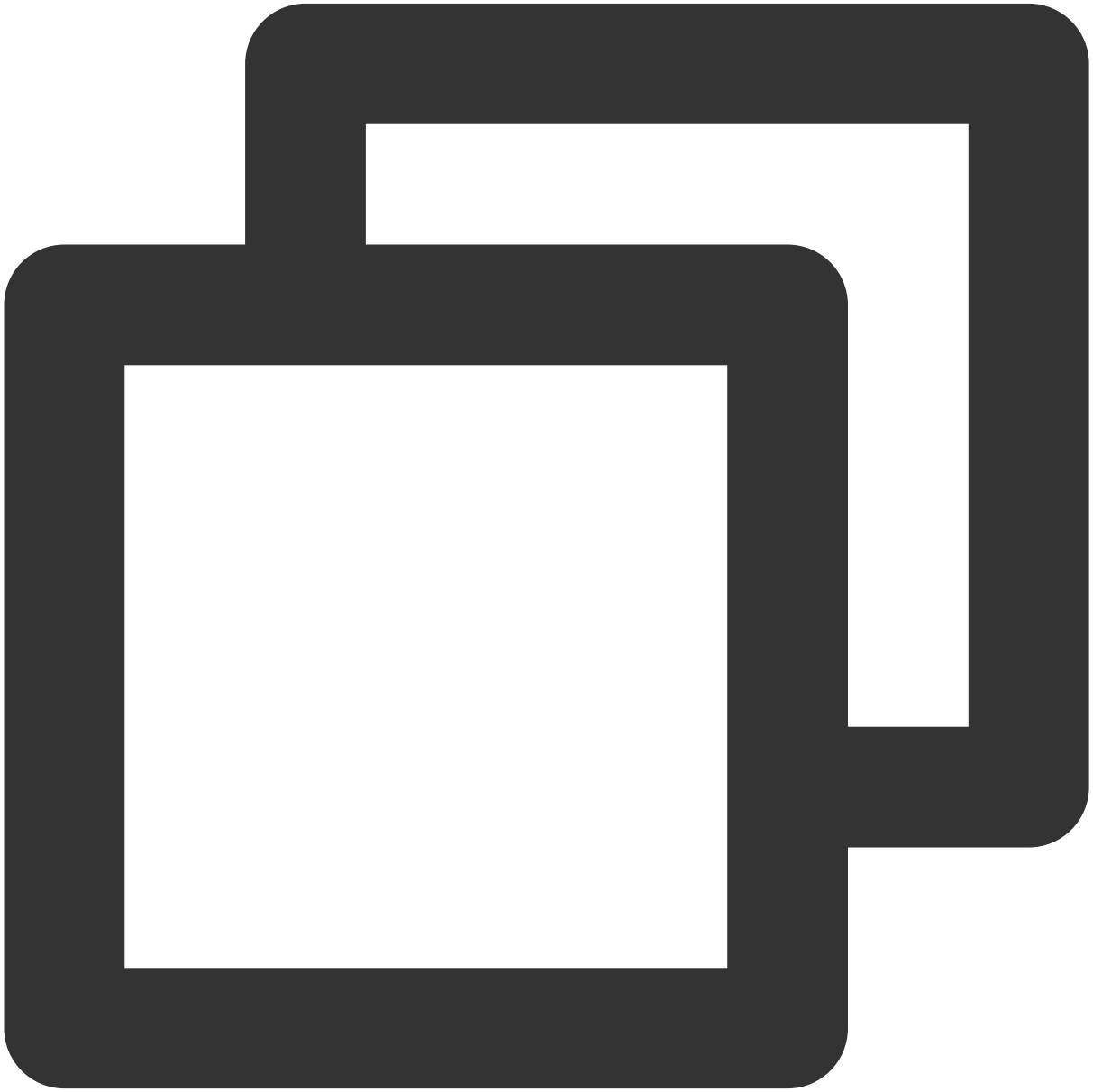
```
void cancelSpeechInvitation(String userId, TUIRoomCoreCallback.ActionCallback call
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	ユーザーID。
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

replySpeechInvitation

参加者がキャスターの発言要請に同意/拒否します。



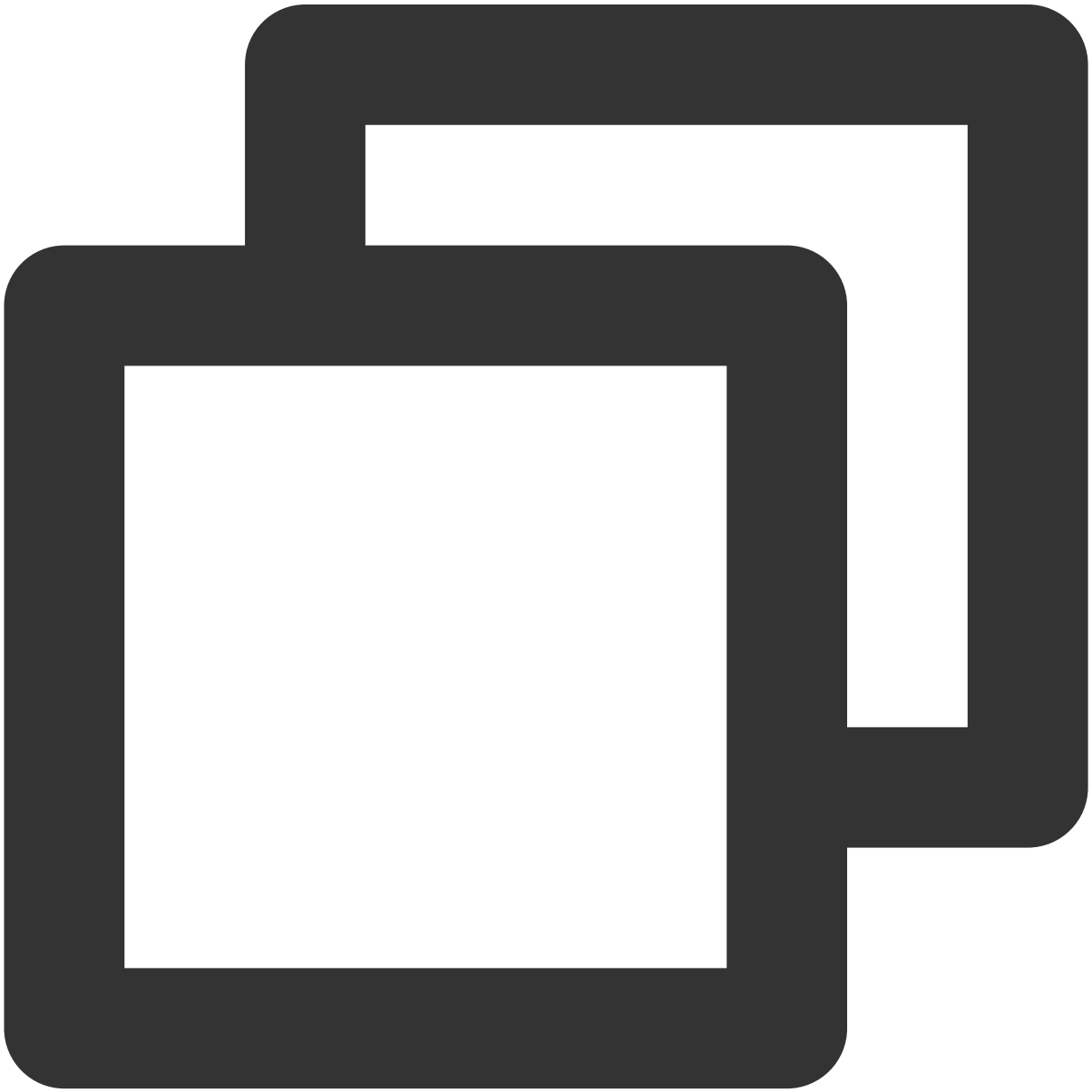
```
void replySpeechInvitation(boolean agree, TUIRoomCoreCallback.ActionCallback callba
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
agree	boolean	同意するかどうか。
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

sendSpeechApplication

参加者が発言を申請します。



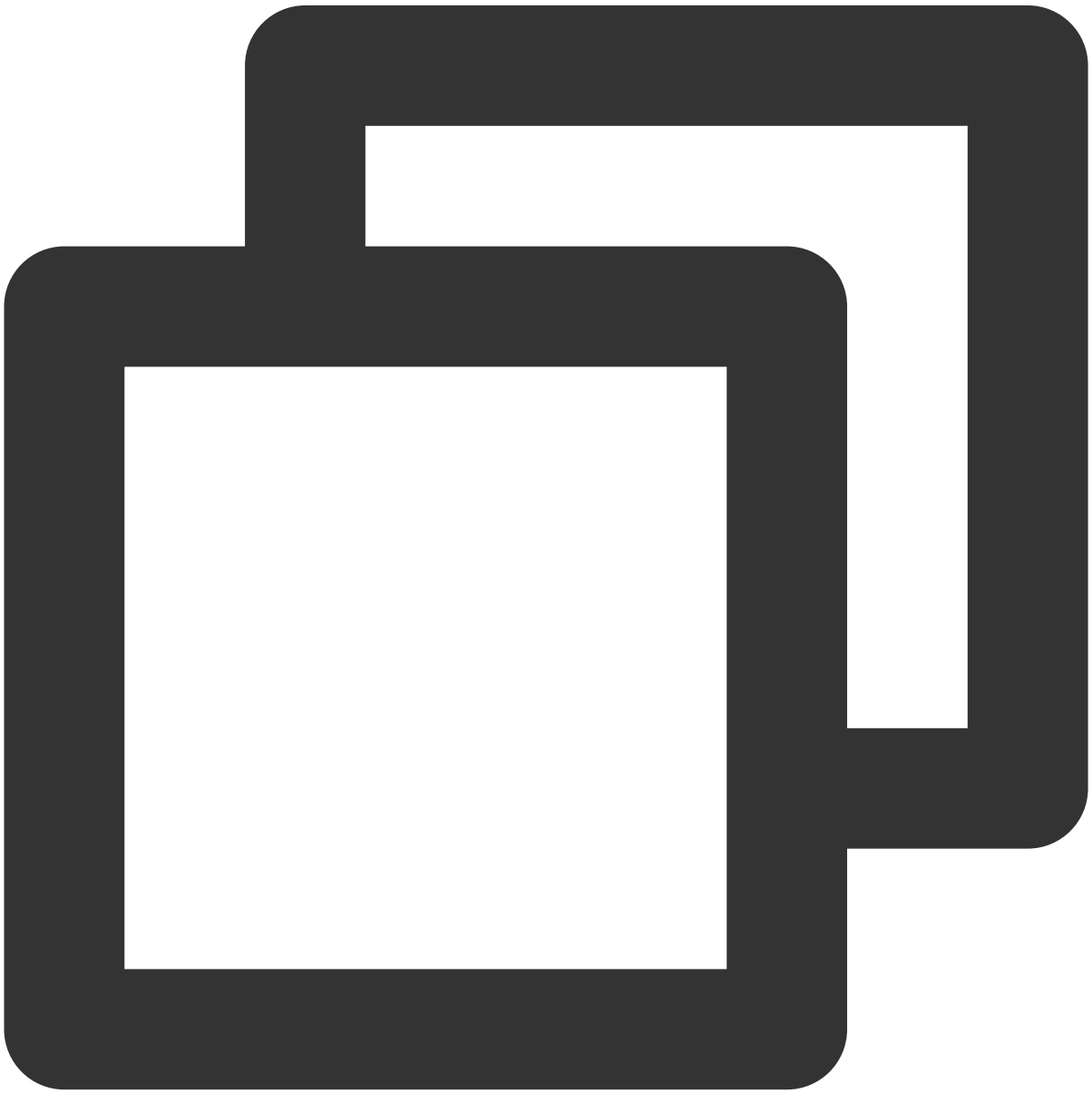
```
void sendSpeechApplication(TUIRoomCoreCallback.InvitationCallback callback);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	TUIRoomCoreCallback.InvitationCallback	結果のコールバック。

cancelSpeechApplication

参加者が発言申請をキャンセルします。



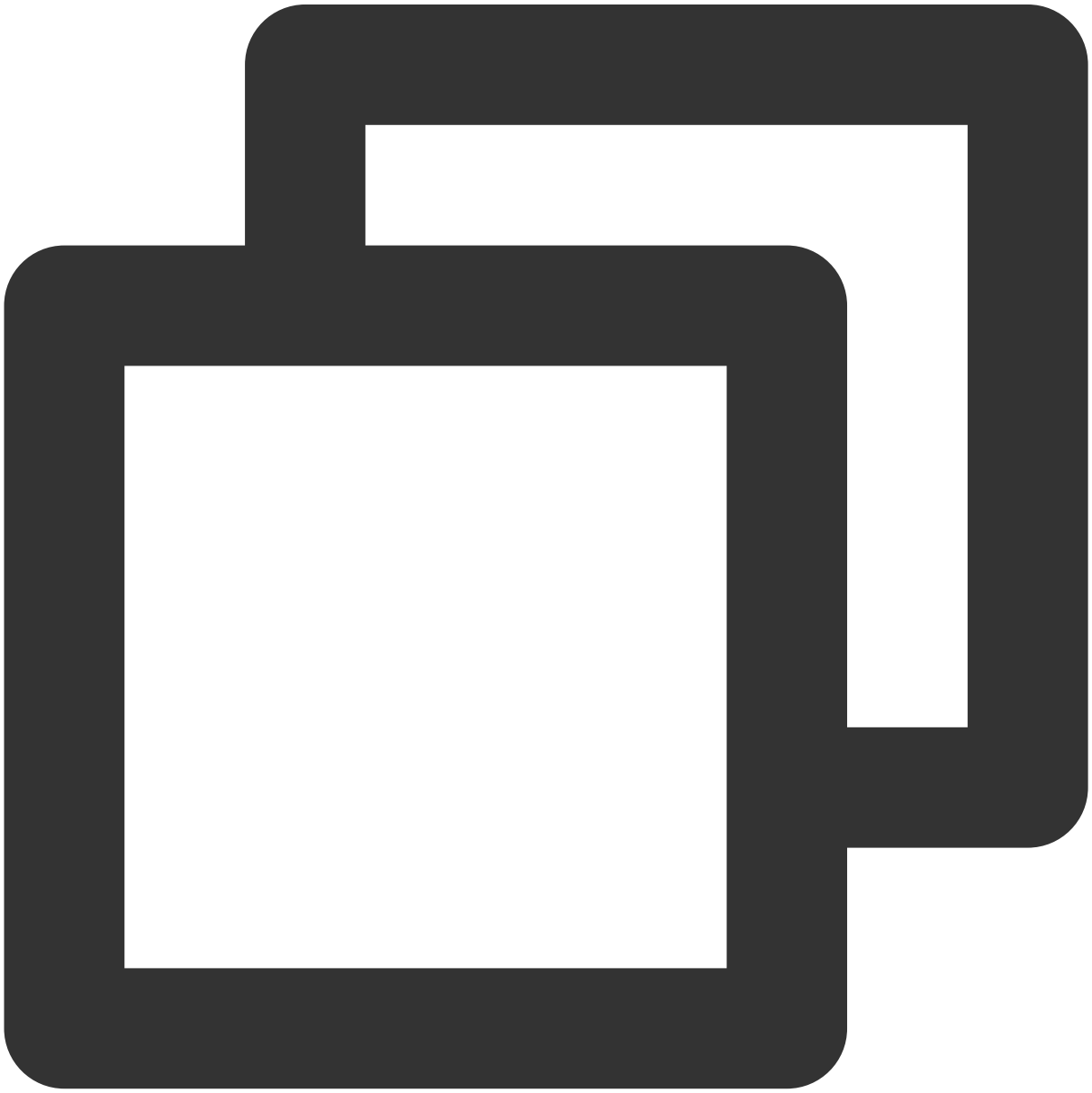
```
void cancelSpeechApplication(TUIRoomCoreCallback.ActionCallback callback);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

replySpeechApplication

キャスターが参加者の発言申請に同意/拒否します。



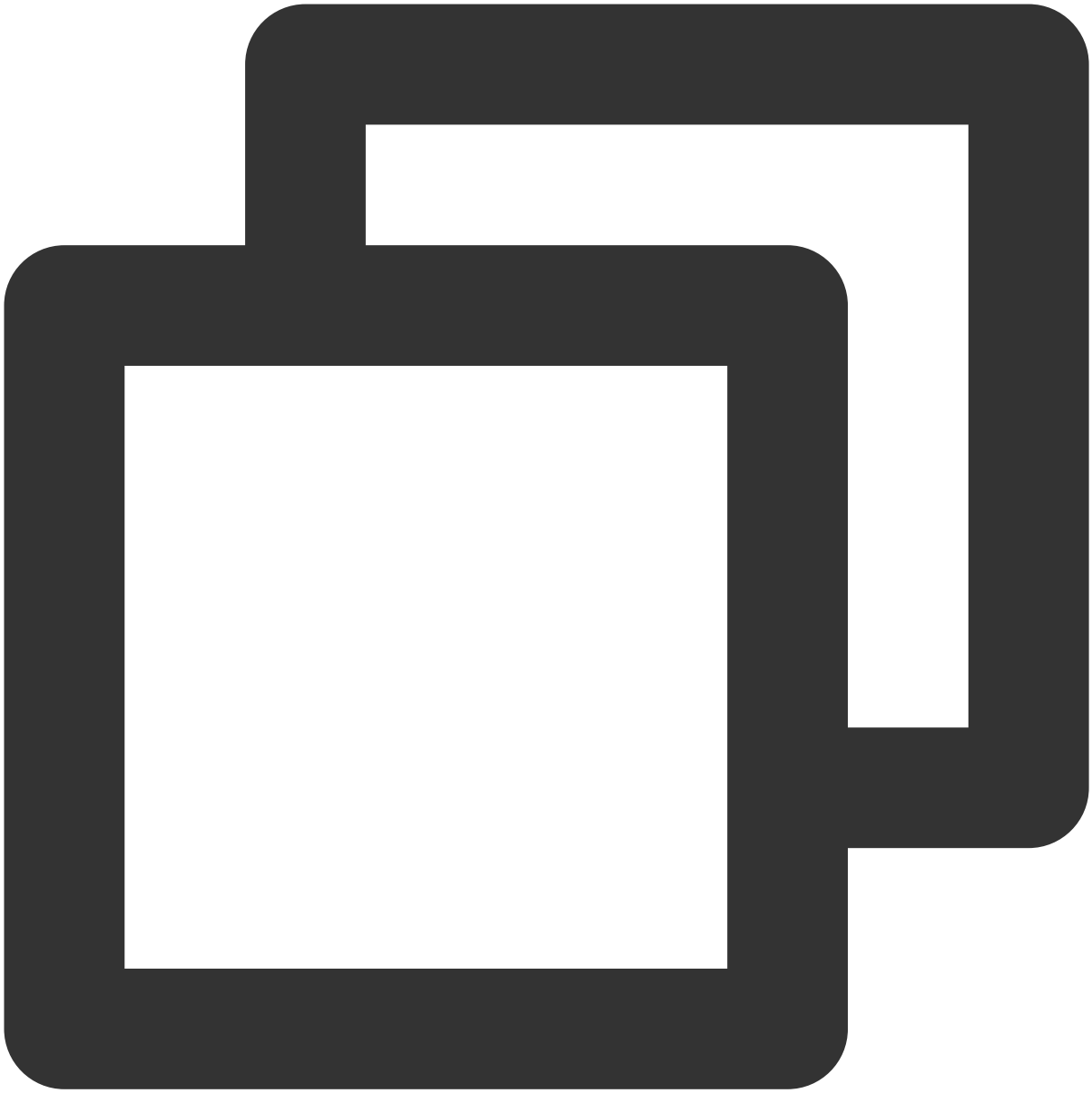
```
void replySpeechApplication(boolean agree, String userId, TUIRoomCoreCallback.Actio
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
agree	boolean	同意するかどうか。
userId	String	ユーザーID。
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

forbidSpeechApplication

キャスターが発言申請を禁止します。



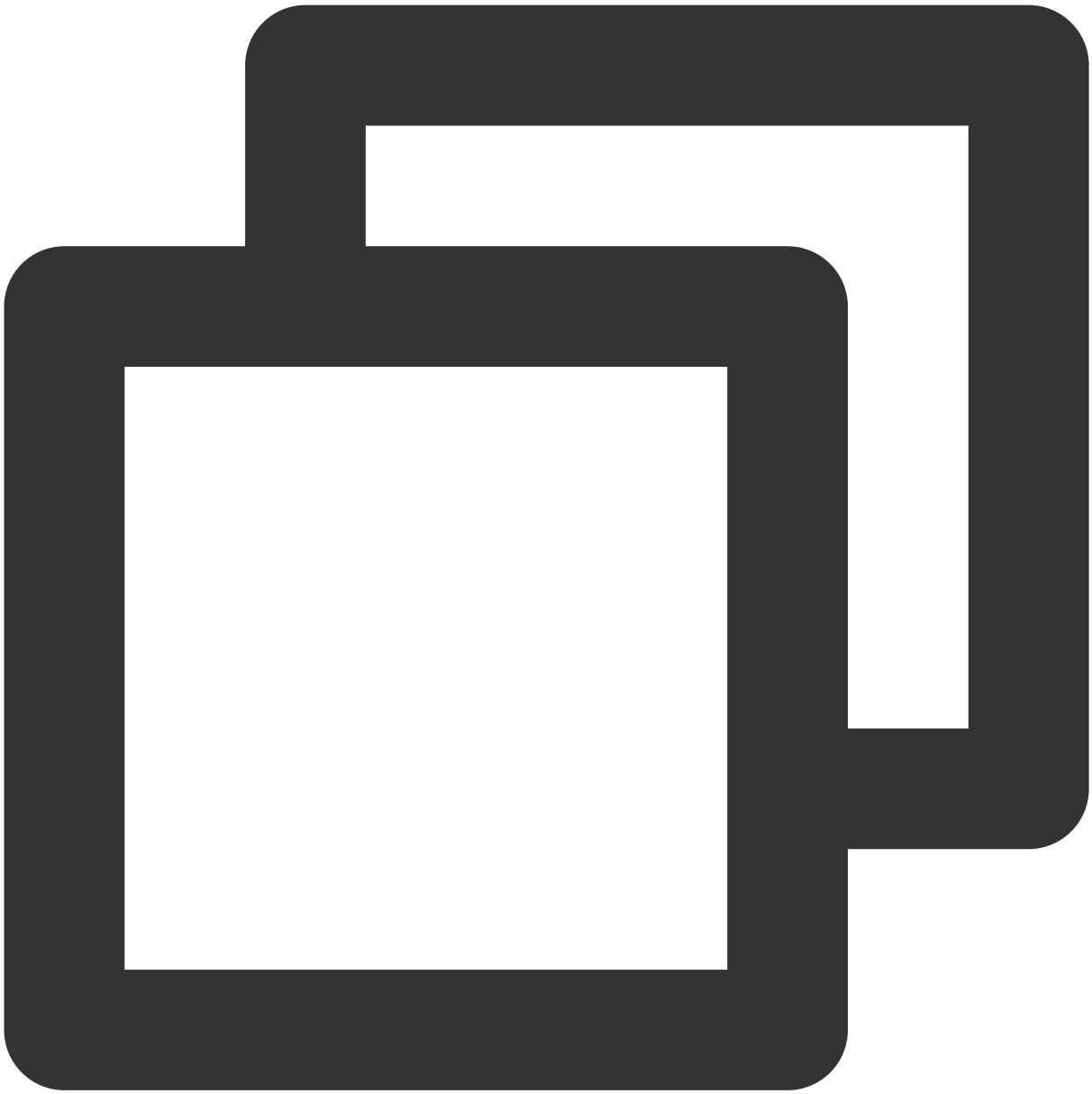
```
void forbidSpeechApplication(boolean forbid, TUIRoomCoreCallback.ActionCallback ca
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
forbid	boolean	禁止するかどうか。
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

sendOffSpeaker

キャスターが参加者に発言の停止を命令します。



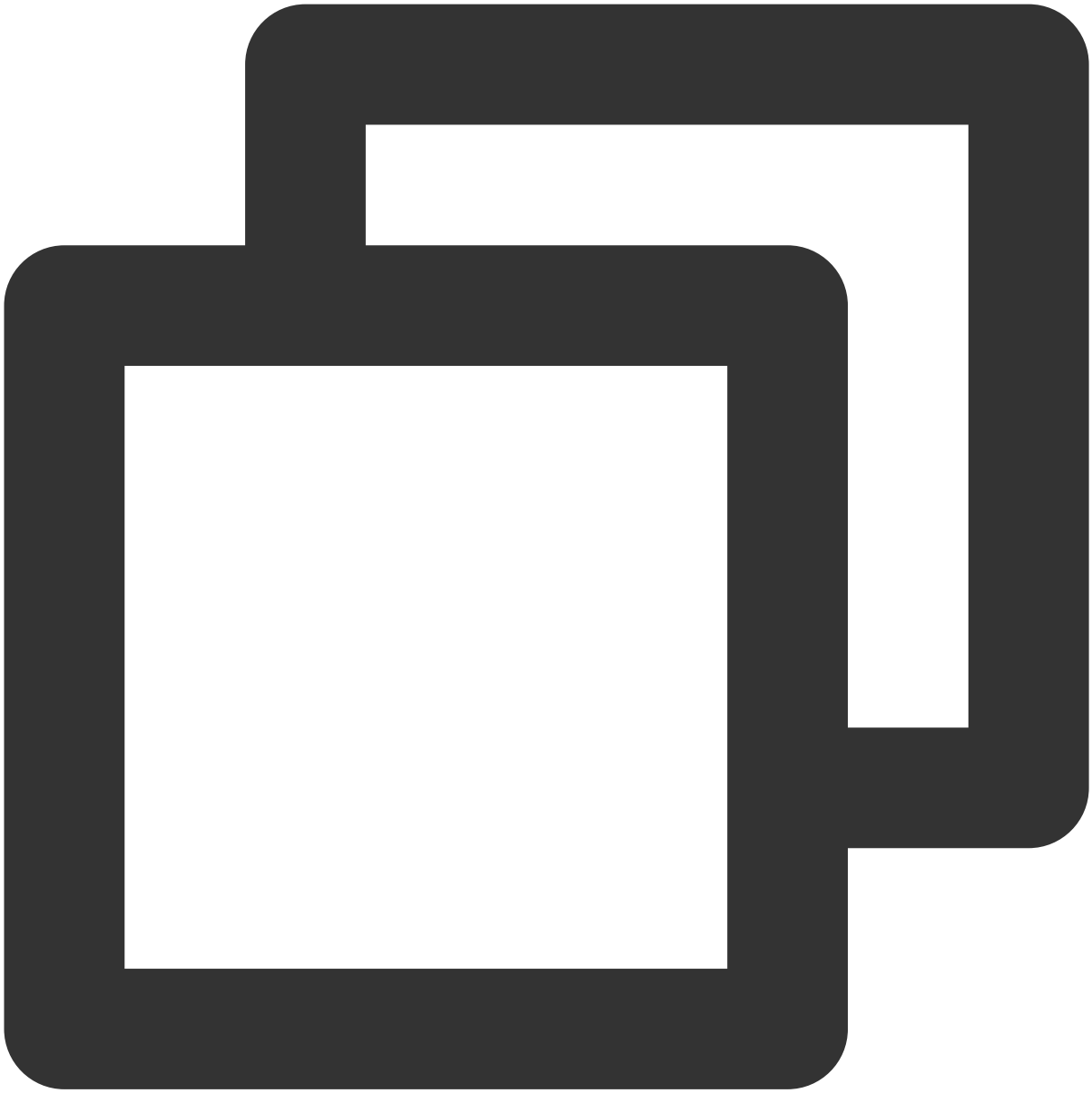
```
void sendOffSpeaker(String userId, TUIRoomCoreCallback.ActionCallback callback);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	ユーザーID。
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

sendOffAllSpeakers

キャスターが全メンバーに発言の停止を命令します。



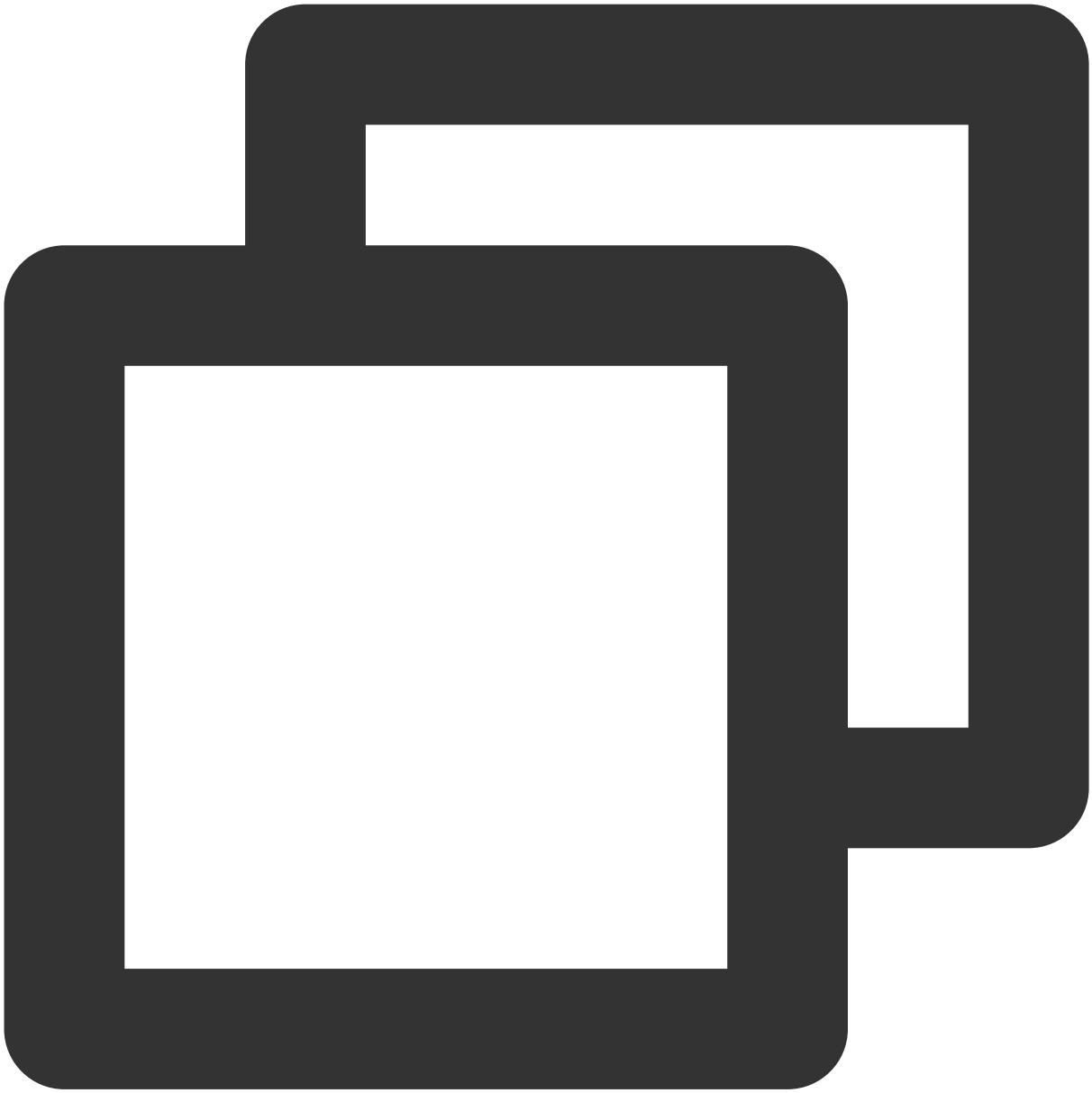
```
void sendOffAllSpeakers(TUIRoomCoreCallback.ActionCallback callback);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

exitSpeechState

参加者が発言を停止し、視聴者になります。



```
void exitSpeechState(TUIRoomCoreCallback.ActionCallback callback);
```

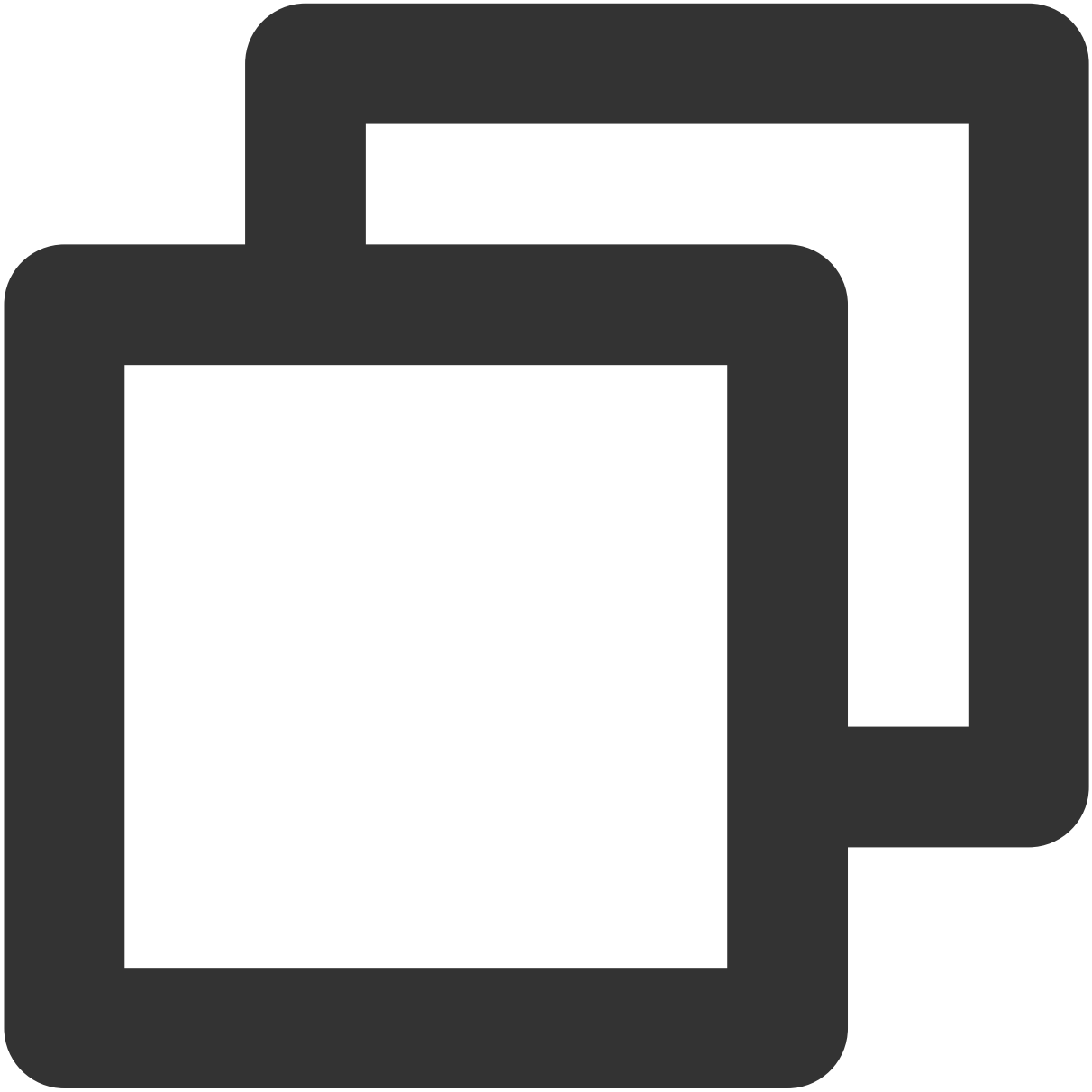
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	TUIRoomCoreCallback.ActionCallback	結果のコールバック。

画面共有インターフェース

startScreenCapture

画面共有を開始。



```
void startScreenCapture(TRTCCloudDef.TRTCVideoEncParam encParams, TRTCCloudDef.TRTC
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

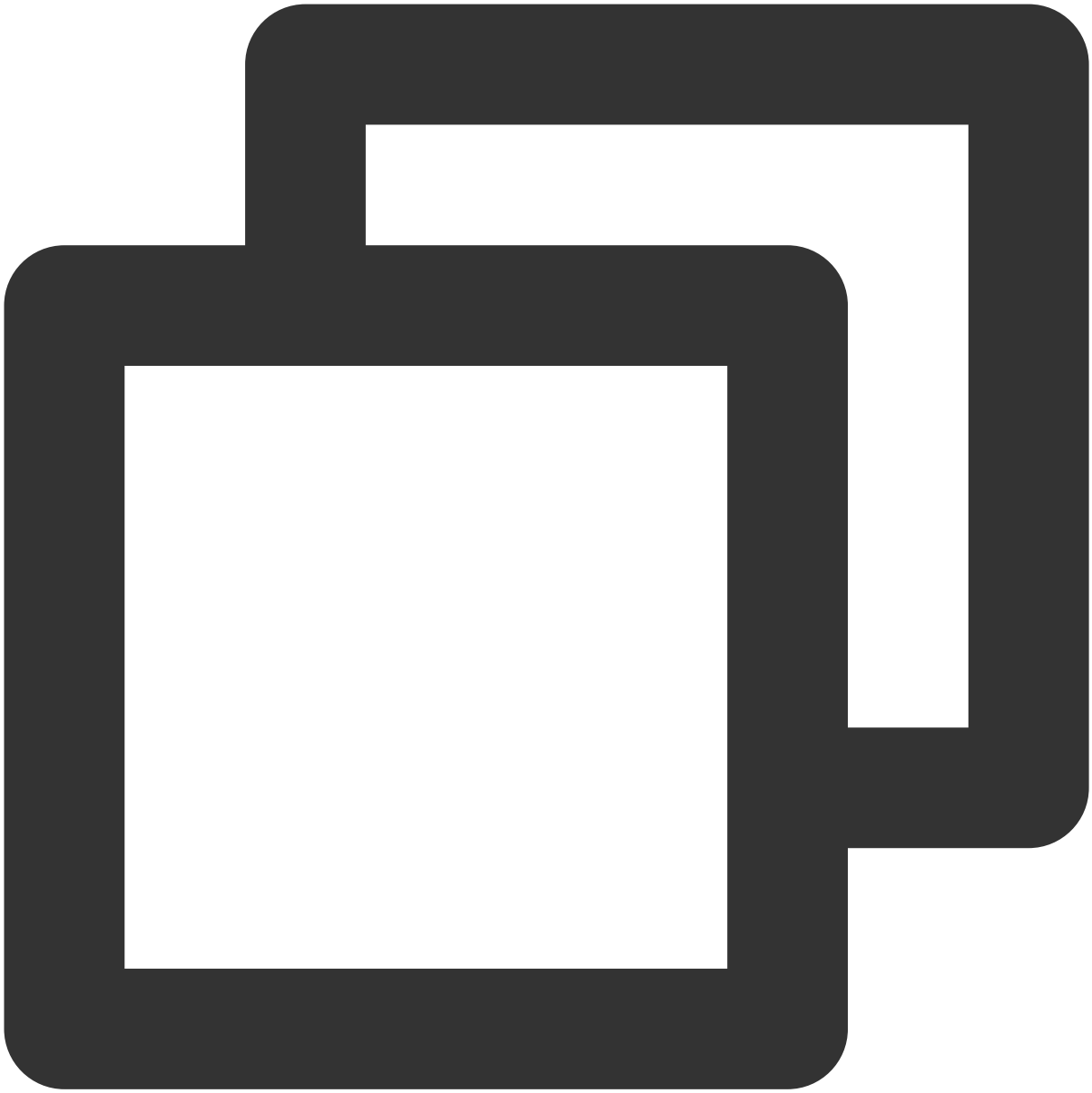
encParams	TRTCCloudDef.TRTCVideoEncParam	画面共有時のエンコードパラメータを設定します。上記の推奨設定を採用することをお勧めします。encParamsにnullを指定した場合、startScreenCaptureを呼び出す前のエンコードパラメータ設定が使用されます。
screenShareParams	TRTCCloudDef.TRTCScreenShareParams	画面共有の特殊なレイアウト設定については、その中のfloatingViewの設定を推奨します。一方で、Appがシステムから強制排除されるのを回避でき、もう一方で、ユーザーのプライバシー保護にも役立ちます。

説明：

詳細については、TRTC SDKをご参照ください。 |

stopScreenCapture

画面キャプチャの停止。

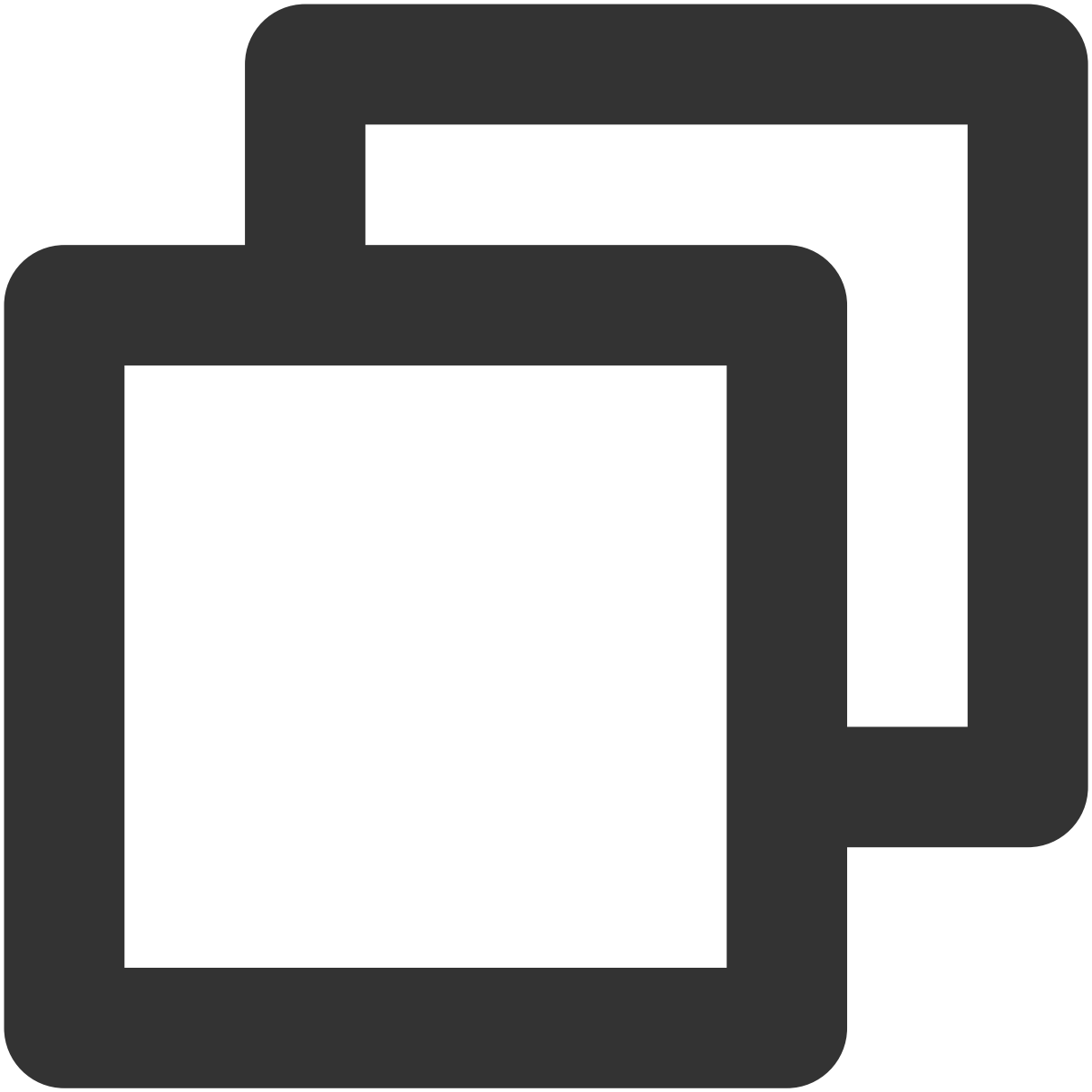


```
void stopScreenCapture();
```

美顔フィルターに関するインターフェース関数

getBeautyManager

美顔管理オブジェクト TXBeautyManagerを取得します。



```
TXBeautyManager getBeautyManager();
```

美顔管理では、次の機能を使用できます。

「美顔のスタイル」、「美白」、「肌色補正（血色・つや感）」、「デカ眼」、「顔痩せ」、「V顔」、「下あご」、「面長補正」、「小鼻」、「キラキラ目」、「白い歯」、「目の弛み除去」、「シワ除去」、「ほうれい線除去」などの美容効果を設定します。

「髪の生え際」、「眼と眼の距離」、「眼の角度」、「唇の形」、「鼻翼」、「鼻の位置」、「唇の厚さ」、「顔の形」を調整します。

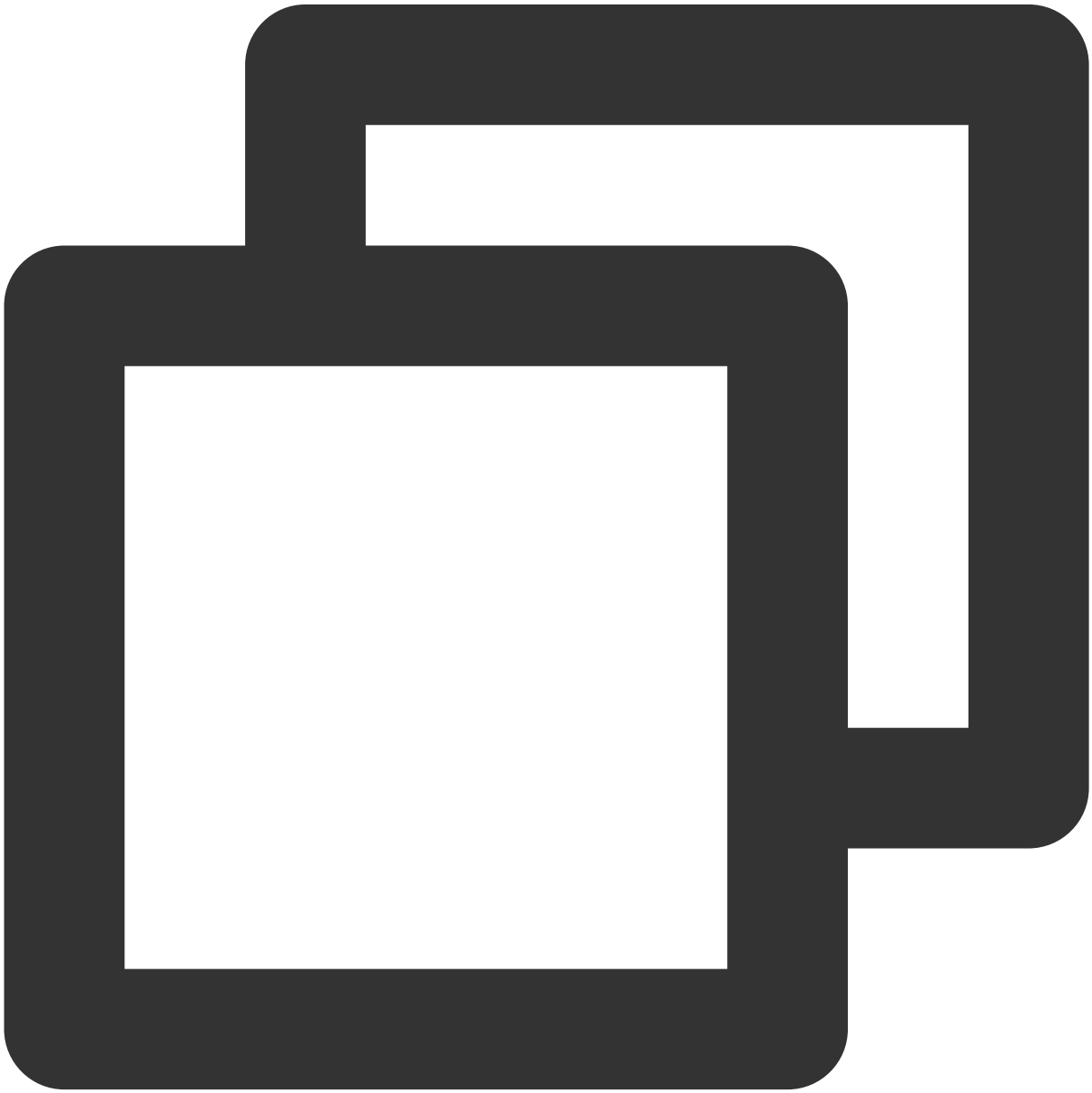
人の顔のスタンプ（素材）等のダイナミック効果を設定します。

メイクアップを追加します。
ジェスチャー認識を行います。

関連設定インターフェース

setVideoQosPreference

ネットワークトラフィックコントロール関連パラメータを設定します。



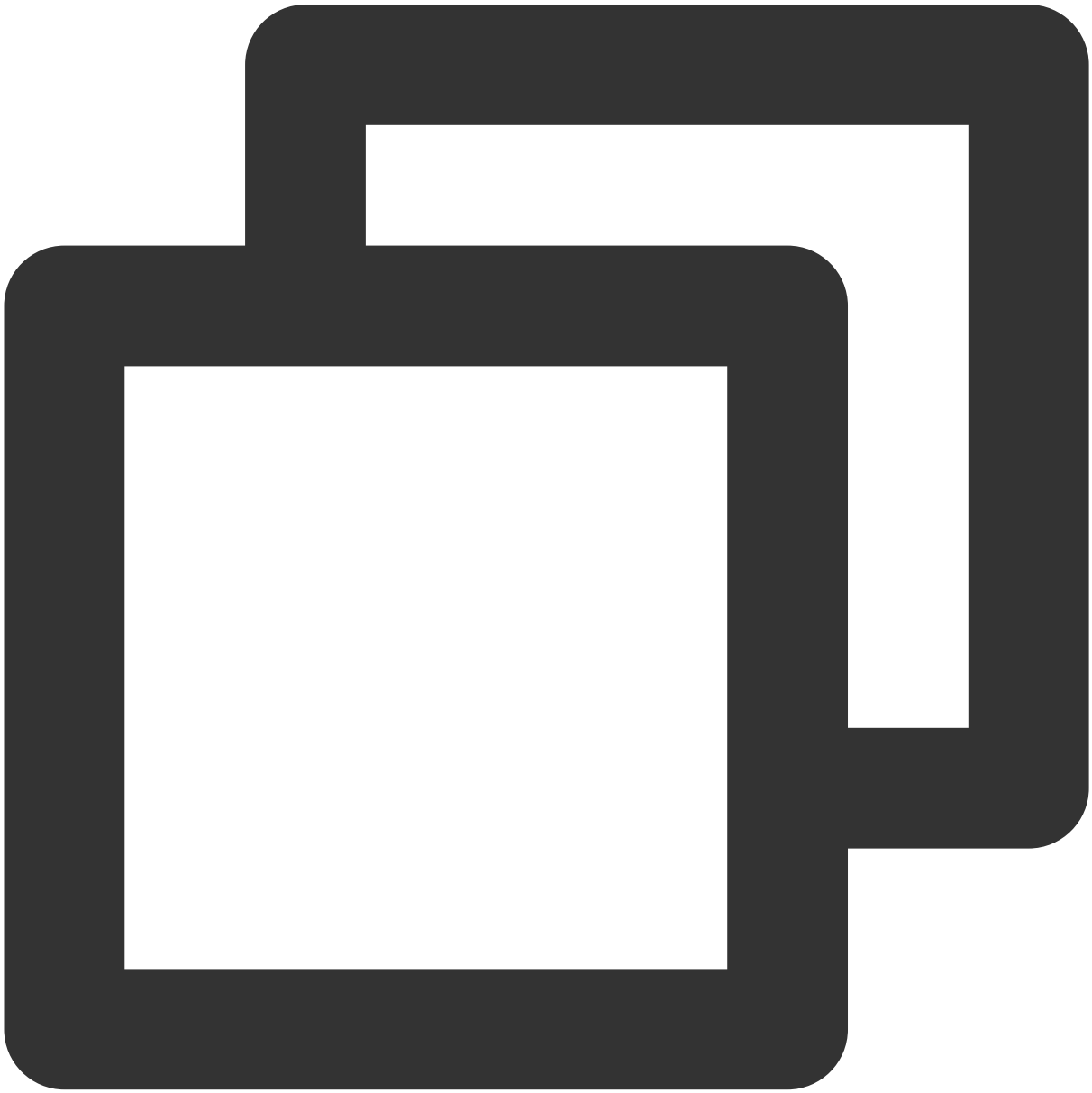
```
void setVideoQosPreference(TRTCCloudDef.TRTCNetworkQosParam preference);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
preference	TRTCCloudDef.TRTCNetworkQosParam	ネットワークトラフィックコントロールポリシー。

setAudioQuality

音質の設定。



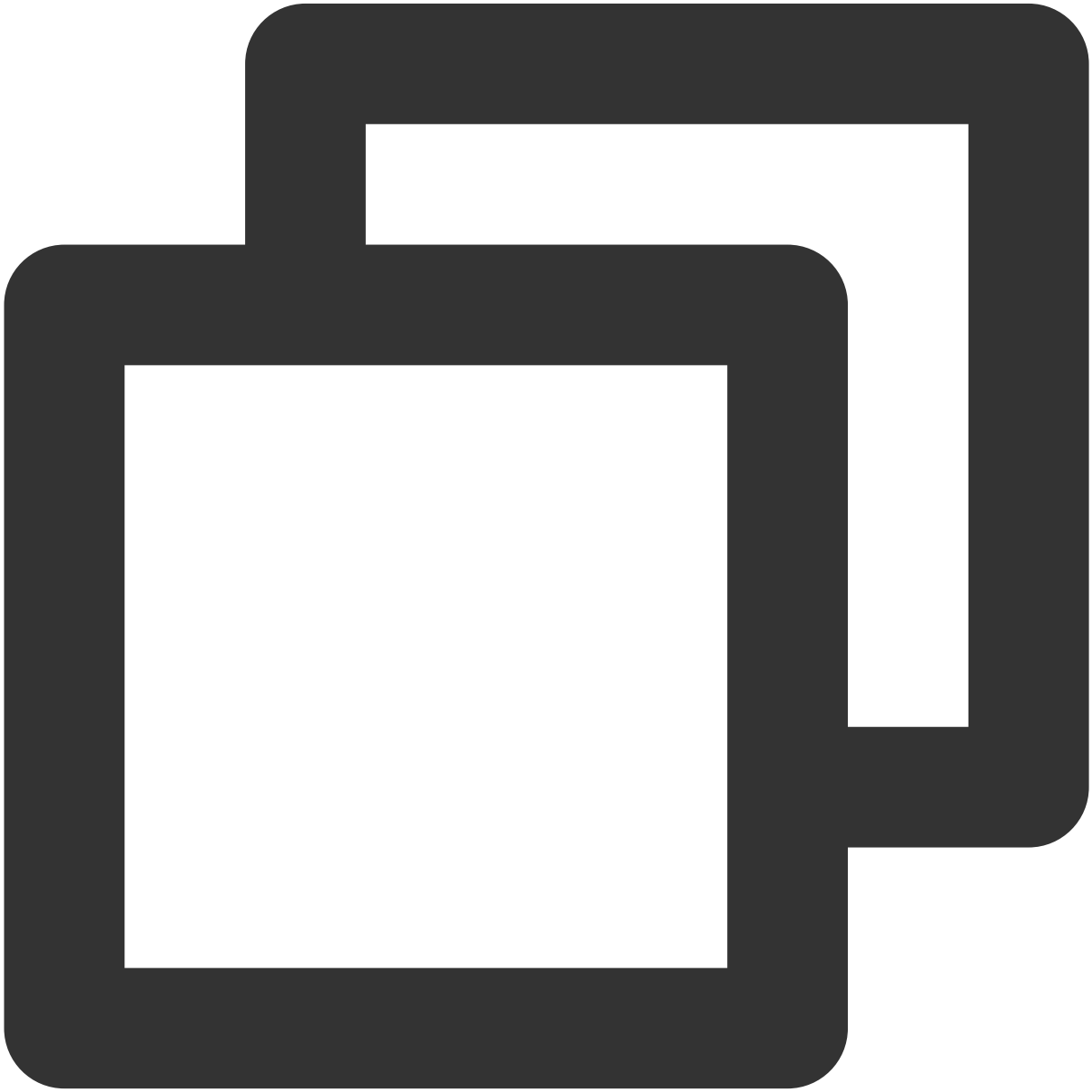
```
void setAudioQuality(int quality);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
quality	int	オーディオ品質。詳細については、TRTC SDKをご参照ください。

setVideoResolution

解像度の設定。



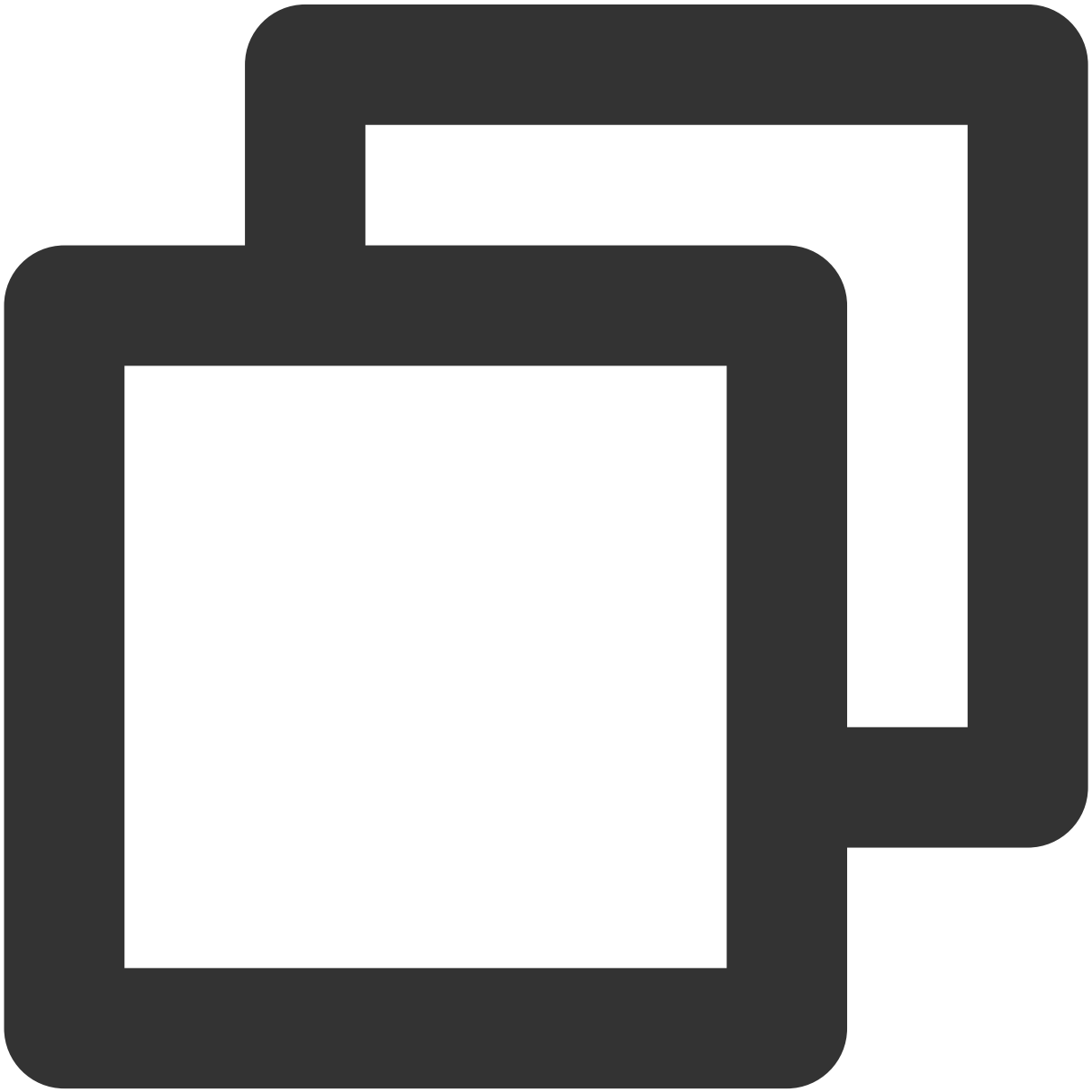
```
void setVideoResolution(int resolution);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
resolution	int	ビデオの解像度。詳細については、TRTC SDKをご参照ください。

setVideoFps

フレームレートの設定。



```
void setVideoFps(int fps);
```

パラメータは下表に示すとおりです。

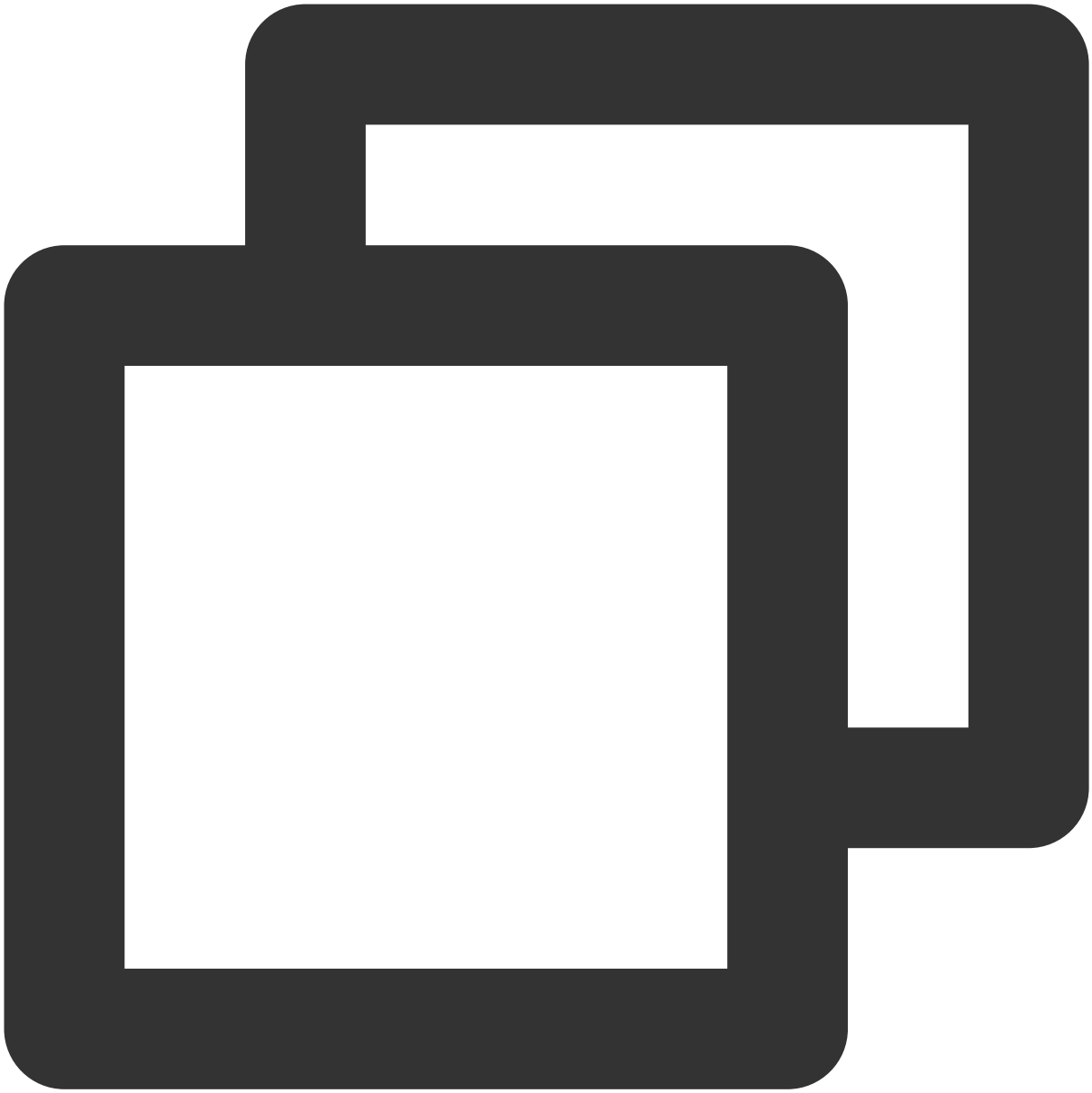
パラメータ	タイプ	意味
fps	int	ビデオキャプチャのフレームレート。

説明：

推奨する値：15fpsまたは20fps。5fps以下ではラグ感が目立ち、10fps以下では軽微なラグ感があります。20fps以上は高すぎて浪費になります（映画のフレームレートは24fps）。

setVideoBitrate

ビットレートの設定。



```
void setVideoBitrate(int bitrate);
```

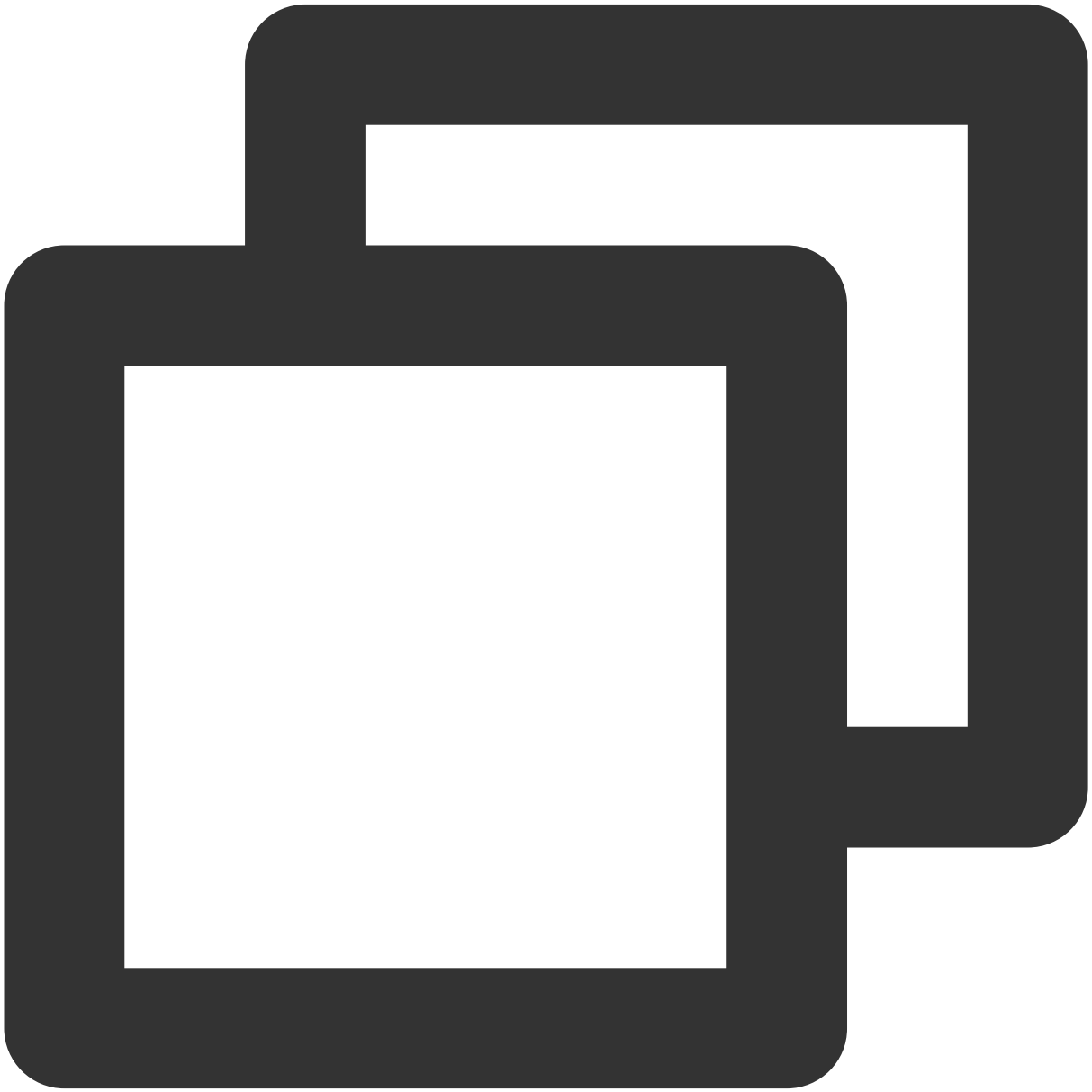
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
bitrate	int	ビットレート。SDKは、目標ビットレートに応じてエンコードを行い、ネットワークの状態が良くない場合のみ、ビデオビットレートを動的に引き下げます。詳細については、TRTC SDKをご参照ください。

説明：
推奨する値：TRTCVideoResolutionの各クラスに注記する最適ビットレートをご参照ください。これをもとにより高いレートに適宜調整することも可能です。例えば、TRTC_VIDEO_RESOLUTION_1280_720に対応する目標ビットレートが1200kbpsであるならば、設定を1500kbpsにし、より鮮明な画像を得ることができます。

enableAudioEvaluation

音量レベルリマインダを有効にします。



```
void enableAudioEvaluation(boolean enable);
```

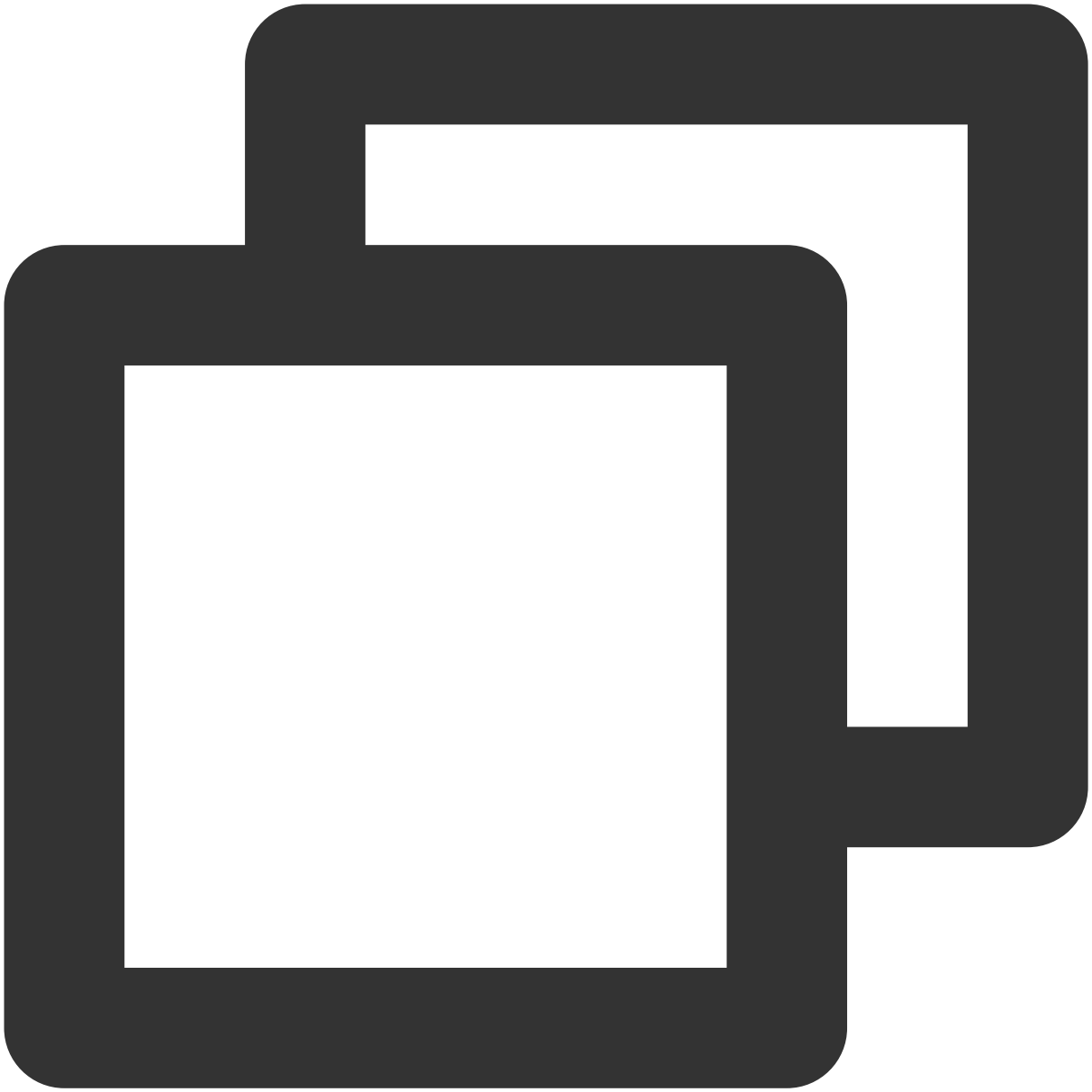
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
enable	boolean	true：オン、false：オフ。

説明：
有効化すると、onUserVolumeUpdateの中でSDKの音量のボリュームに対する評価を取得できます。

setAudioPlayVolume

再生音量の設定。



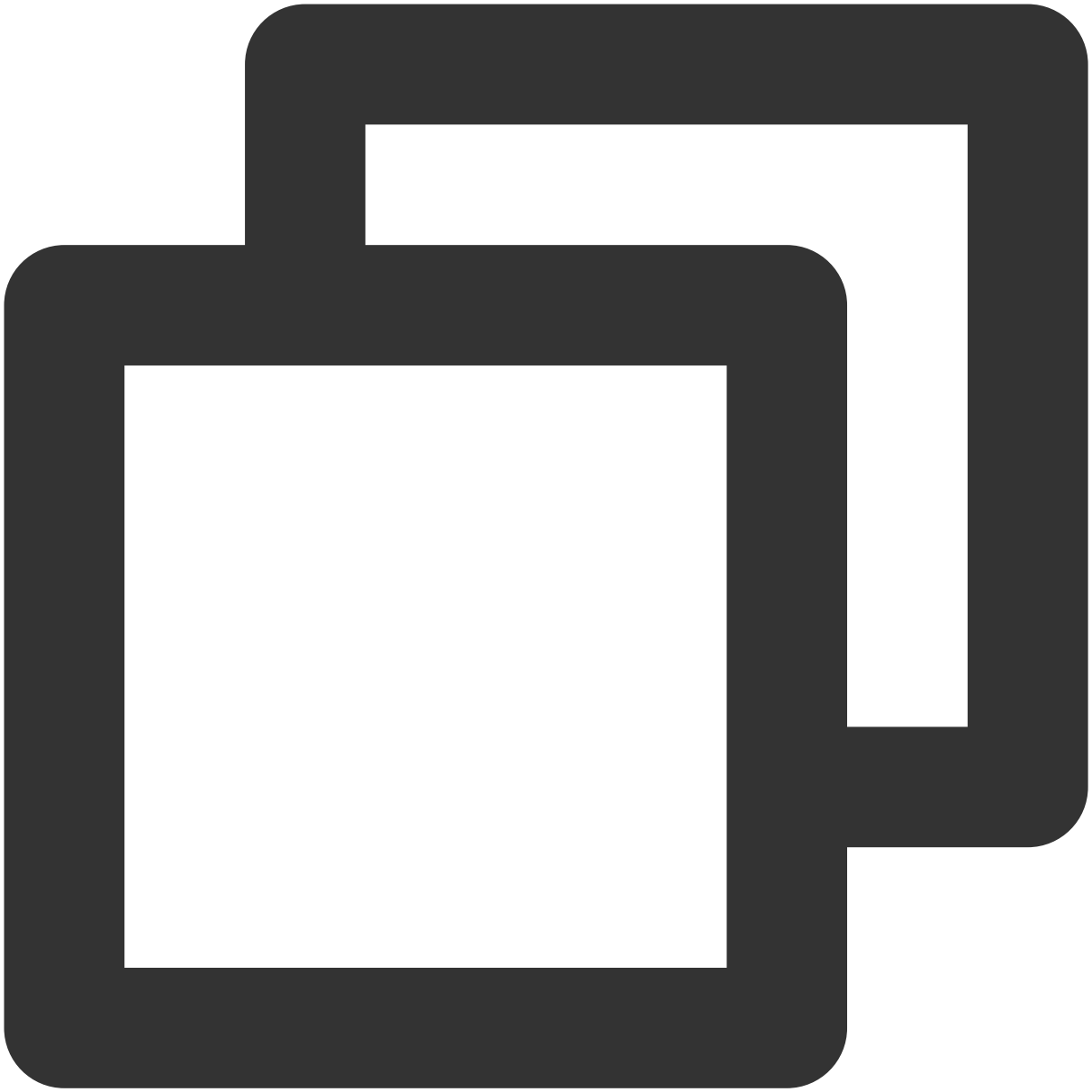
```
void setAudioPlayVolume(int volume);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
volume	int	再生音量、0〜100、 デフォルト100。

setAudioCaptureVolume

マイクの集音音量設定



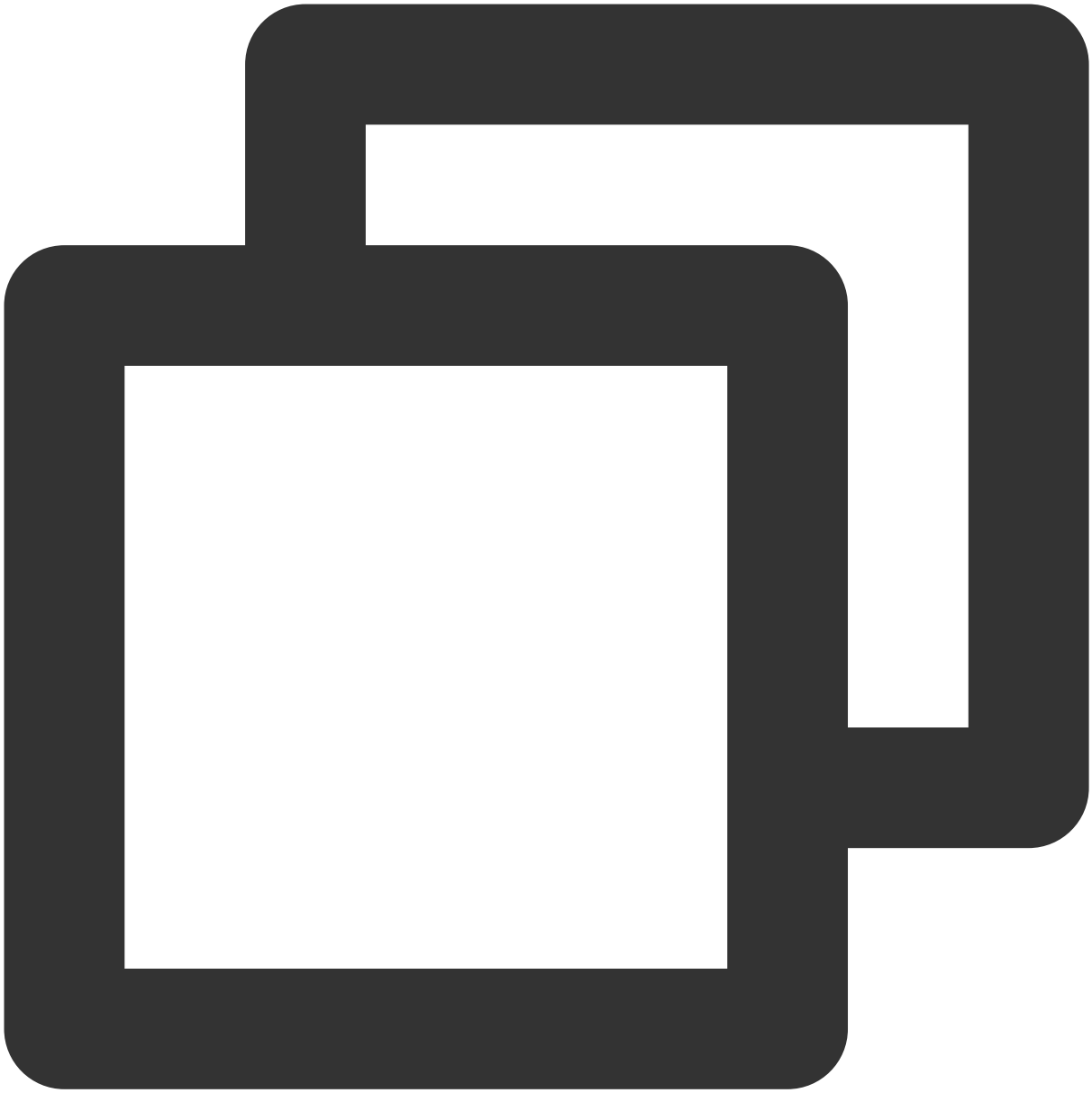
```
void setAudioCaptureVolume(int volume);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
volume	int	集音音量、0〜100、 デフォルト100。

startFileDumping

録音の開始。



```
void startFileDumping(TRTCCloudDef.TRTCAudioRecordingParams trtcAudioRecordingParam
```

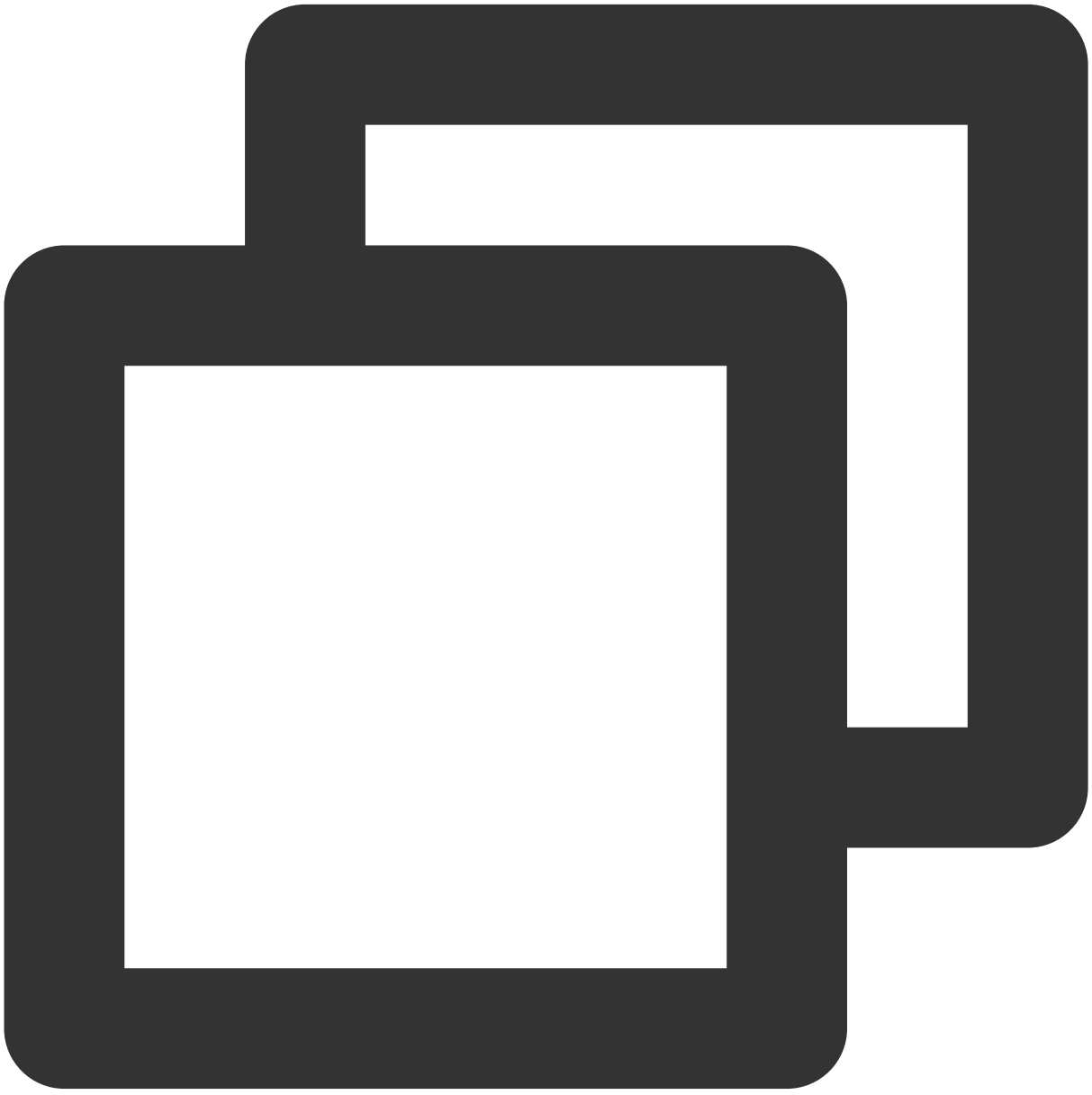
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
trtcAudioRecordingParams	TRTCCloudDef.TRTCAudioRecordingParams	録音パラメータ。詳細については、TRTC SDKをご参照ください。

説明：
この方法で呼び出した後、 SDKは通話プロセスの中のすべての音声（ローカル音声、リモート音声、BGMなど）を1つのファイルにレコーディングします。ルームに参加しているか否かにかかわらず、このインターフェースを呼び出せば有効となります。leaveRoomを呼び出した時に録音中であれば、録音は自動的に停止します。

stopFileDumping

録音の停止。

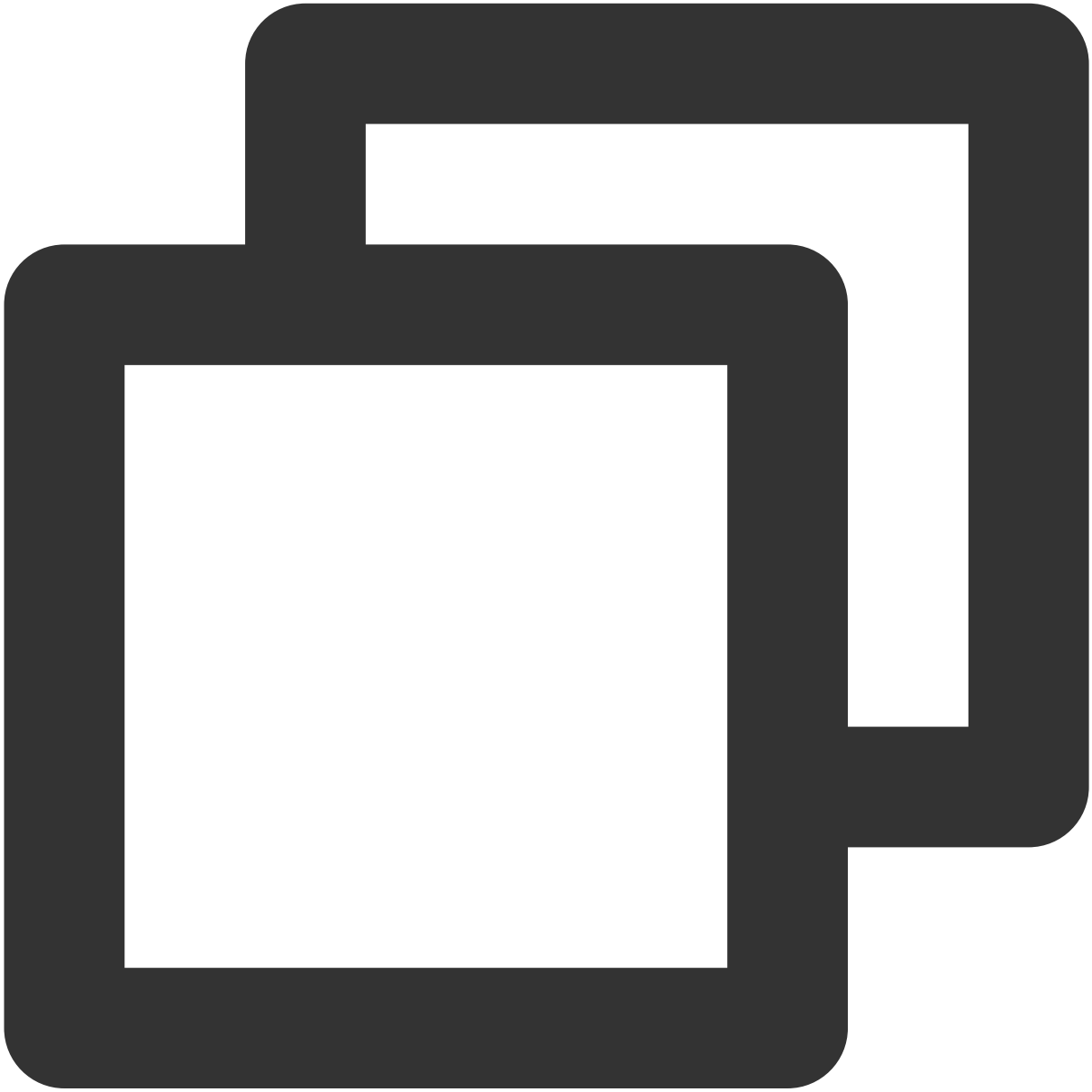


```
void stopFileDumping();
```

SDKバージョンインターフェースの取得

getSdkVersion

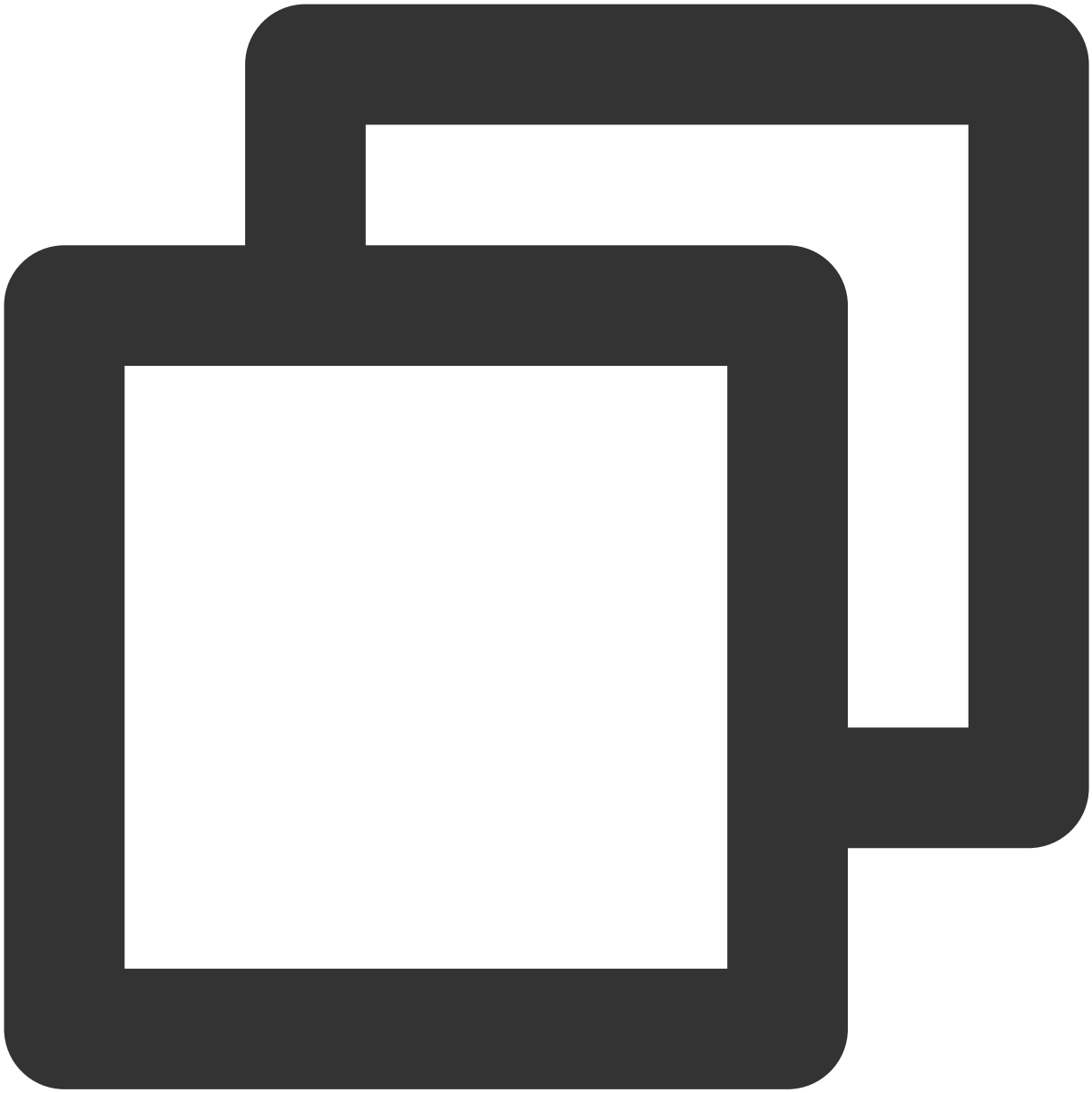
SDKバージョン情報を取得します。



```
int getSdkVersion();
```

エラーイベントコールバック

onError



```
void onError(int code, String message);
```

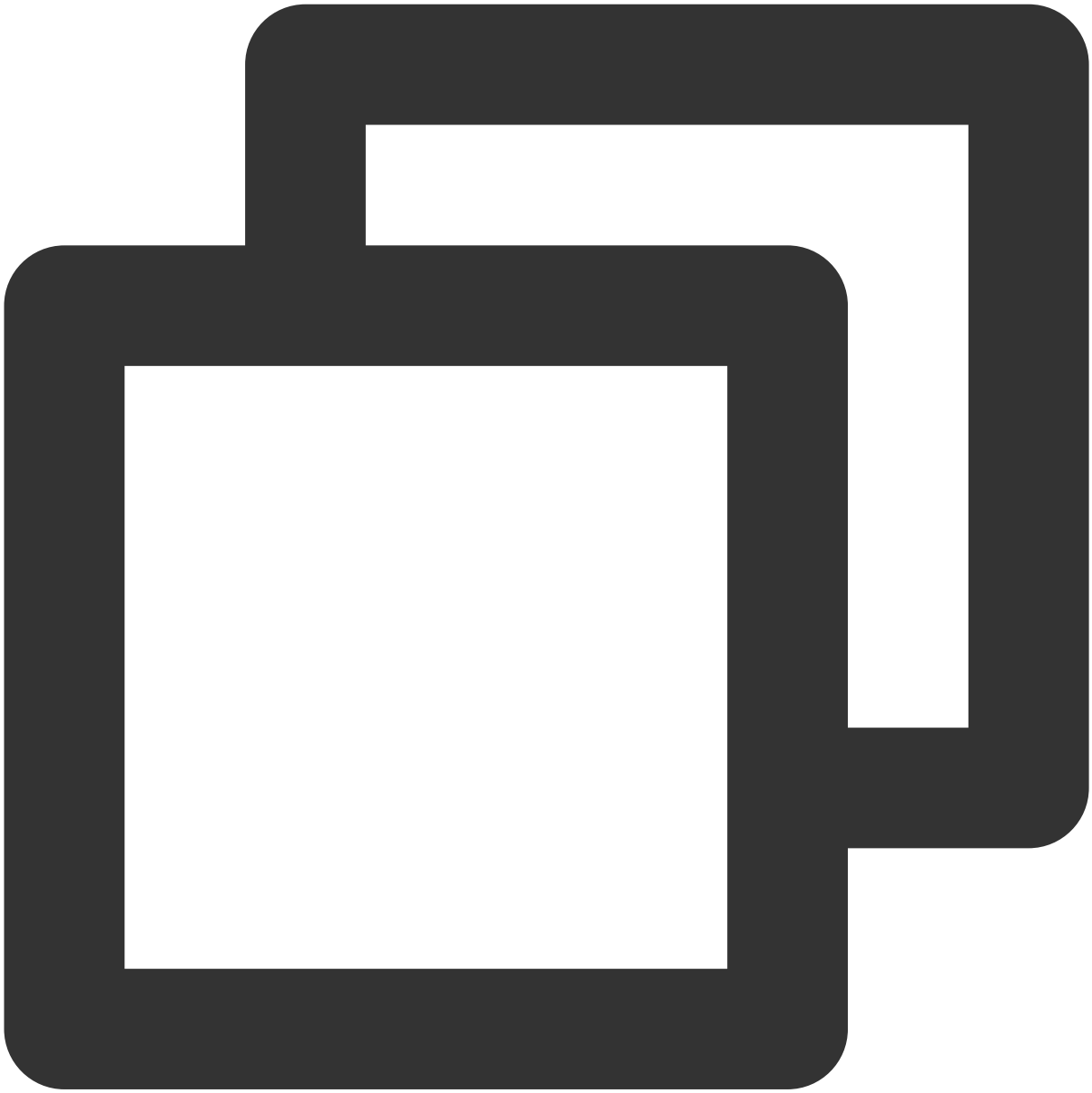
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
code	int	エラーコード。
message	String	エラー情報。

基本イベントコールバック

onDestroyRoom

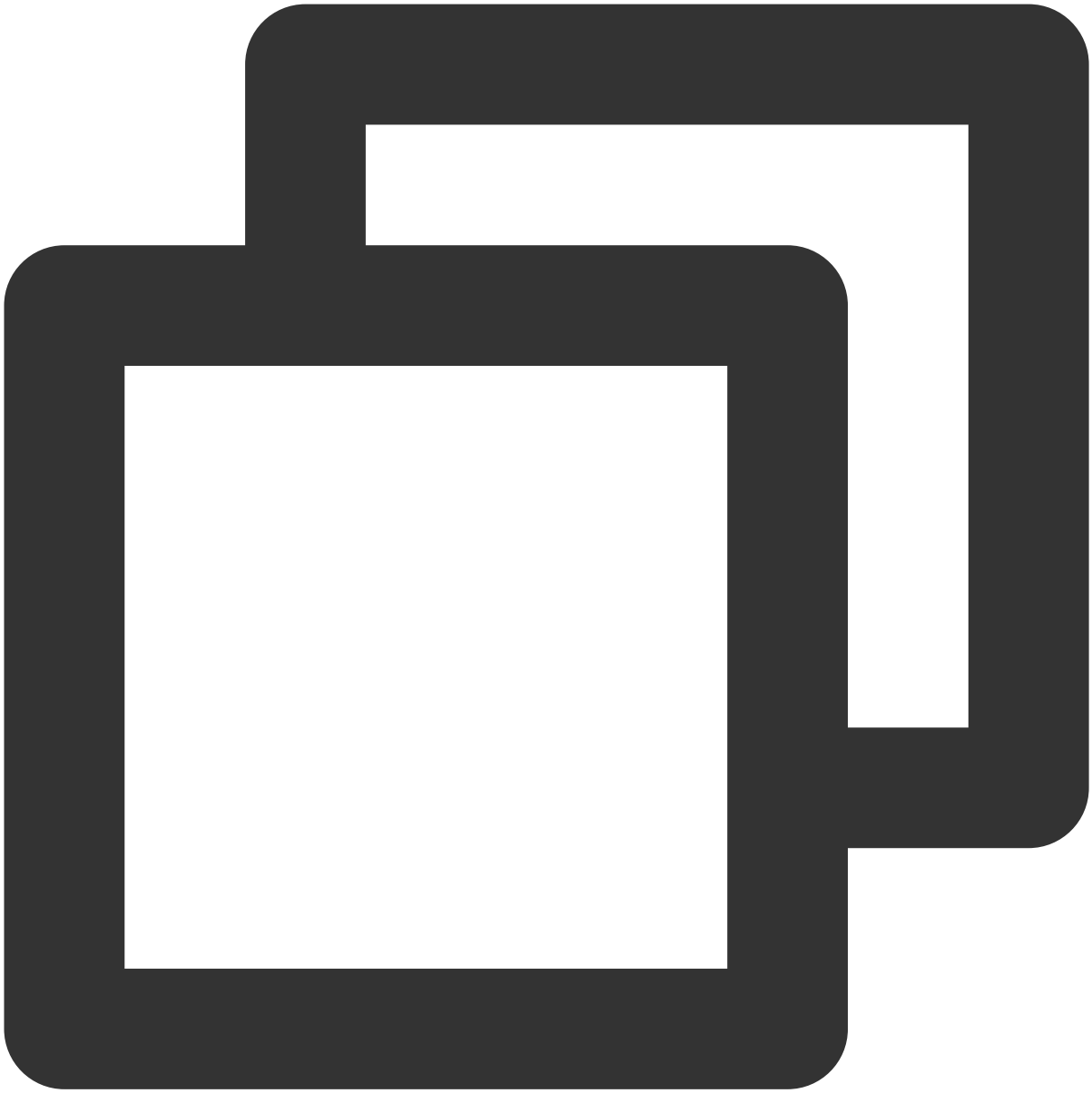
ルーム解散のコールバックです。



```
void onDestroyRoom();
```

onUserVoiceVolume

ユーザー音量の大きさのコールバック。



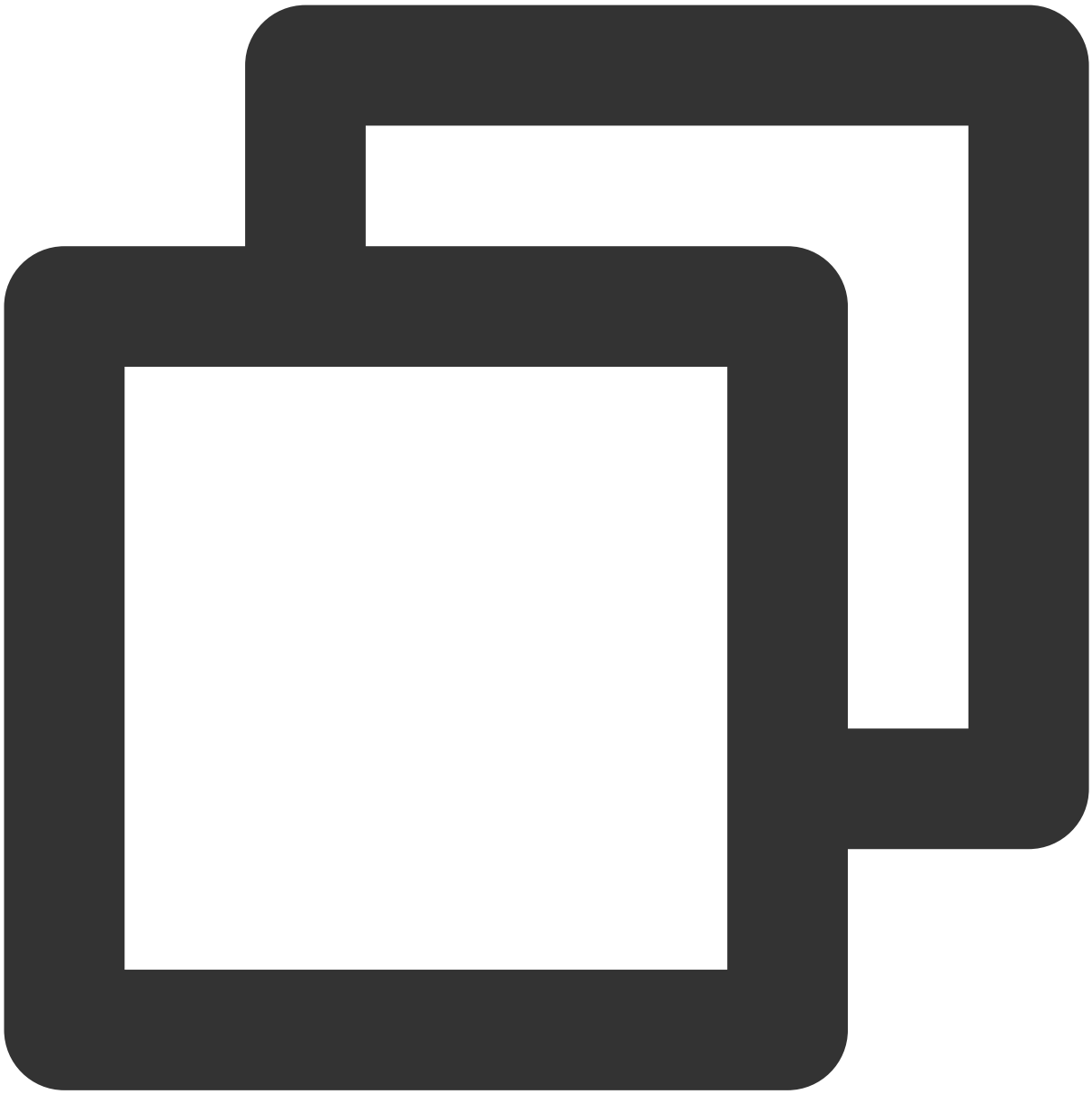
```
void onUserVoiceVolume(String userId, int volume);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	ユーザーID。
volume	int	ユーザーの音量の大きさ、値の範囲0～100。

onRoomMasterChanged

キャスター変更のコールバック。



```
void onRoomMasterChanged(String previousUserId, String currentUserId);
```

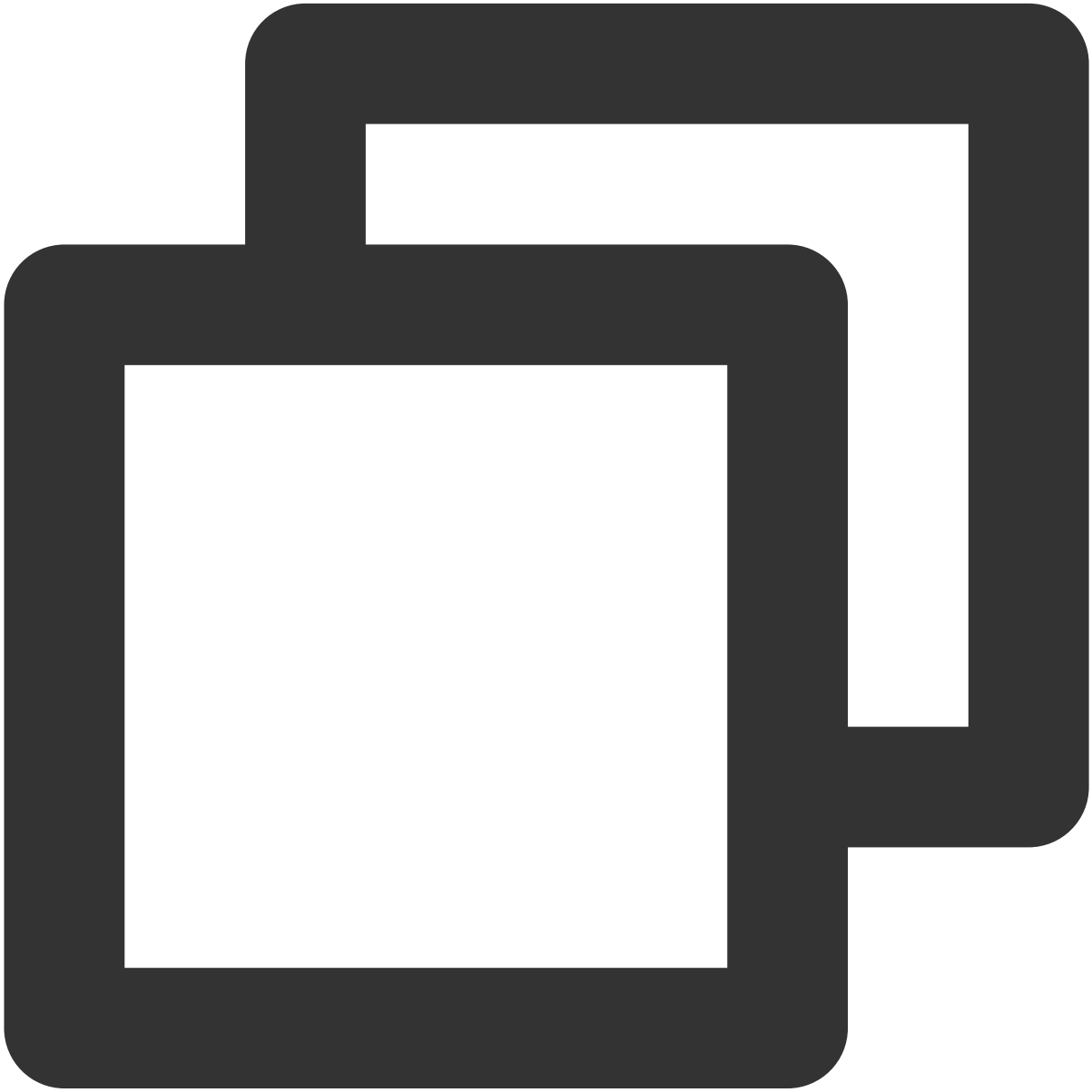
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
previousUserId	String	変更前のキャスターユーザーID。
currentUserId	String	変更後のキャスターユーザーID。

リモートユーザーコールバックイベント

onRemoteUserEnter

リモートユーザー入室コールバック。



```
void onRemoteUserEnter(String userId);
```

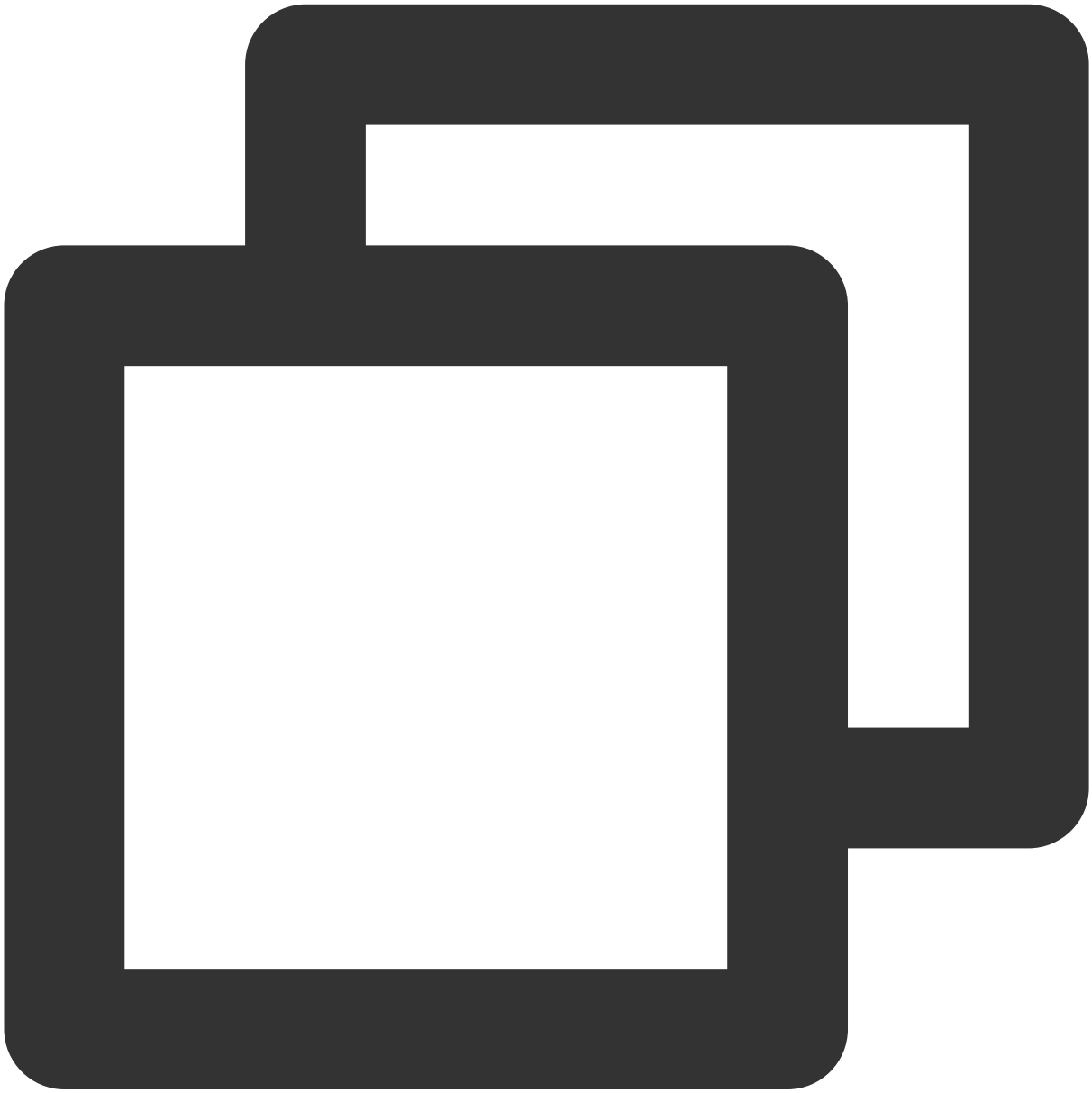
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	String	ユーザーID。

onRemoteUserLeave

リモートユーザー退室コールバック。



```
void onRemoteUserLeave(String userId);
```

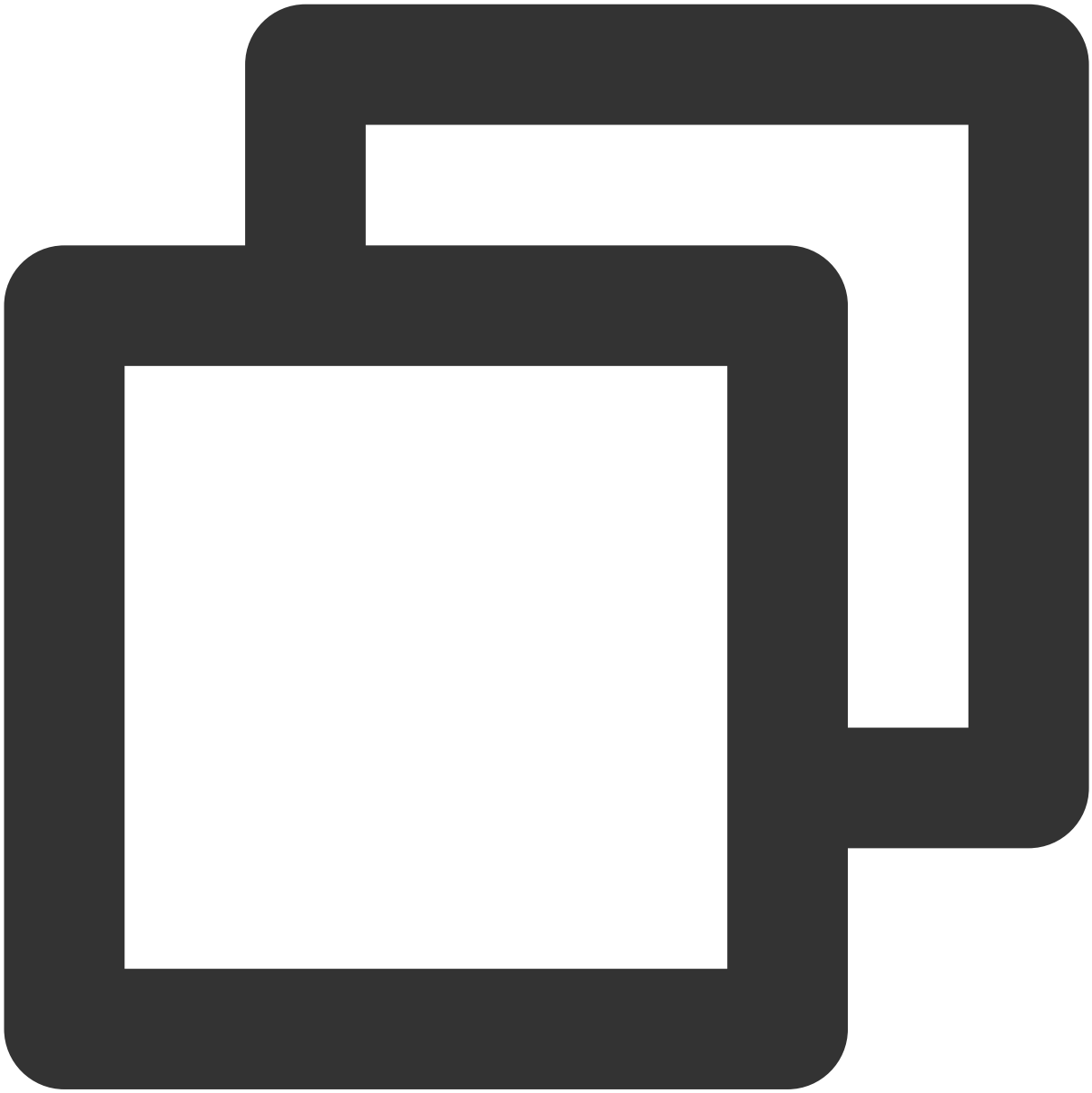
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	String	ユーザーID。

onRemoteUserCameraAvailable

リモートユーザーが、カメラ、ビデオを起動しているかどうか。



```
void onRemoteUserCameraAvailable(String userId, boolean available);
```

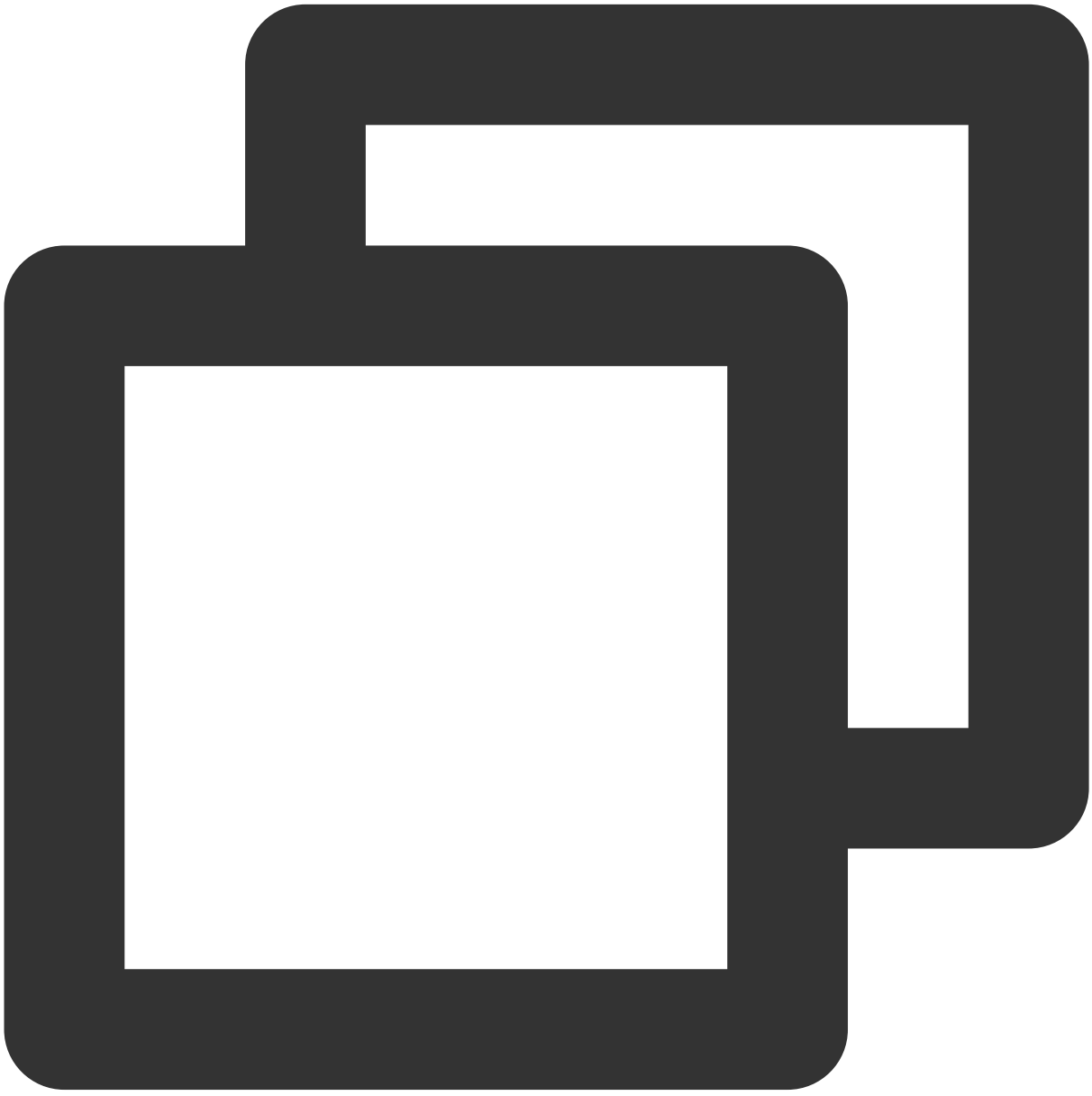
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	String	ユーザーID。
available	boolean	true：ビデオストリームデータあり、false：ビデオストリームデータなし。

onRemoteUserScreenVideoAvailable

メンバーのビデオ共有**オン/オフ**の通知。



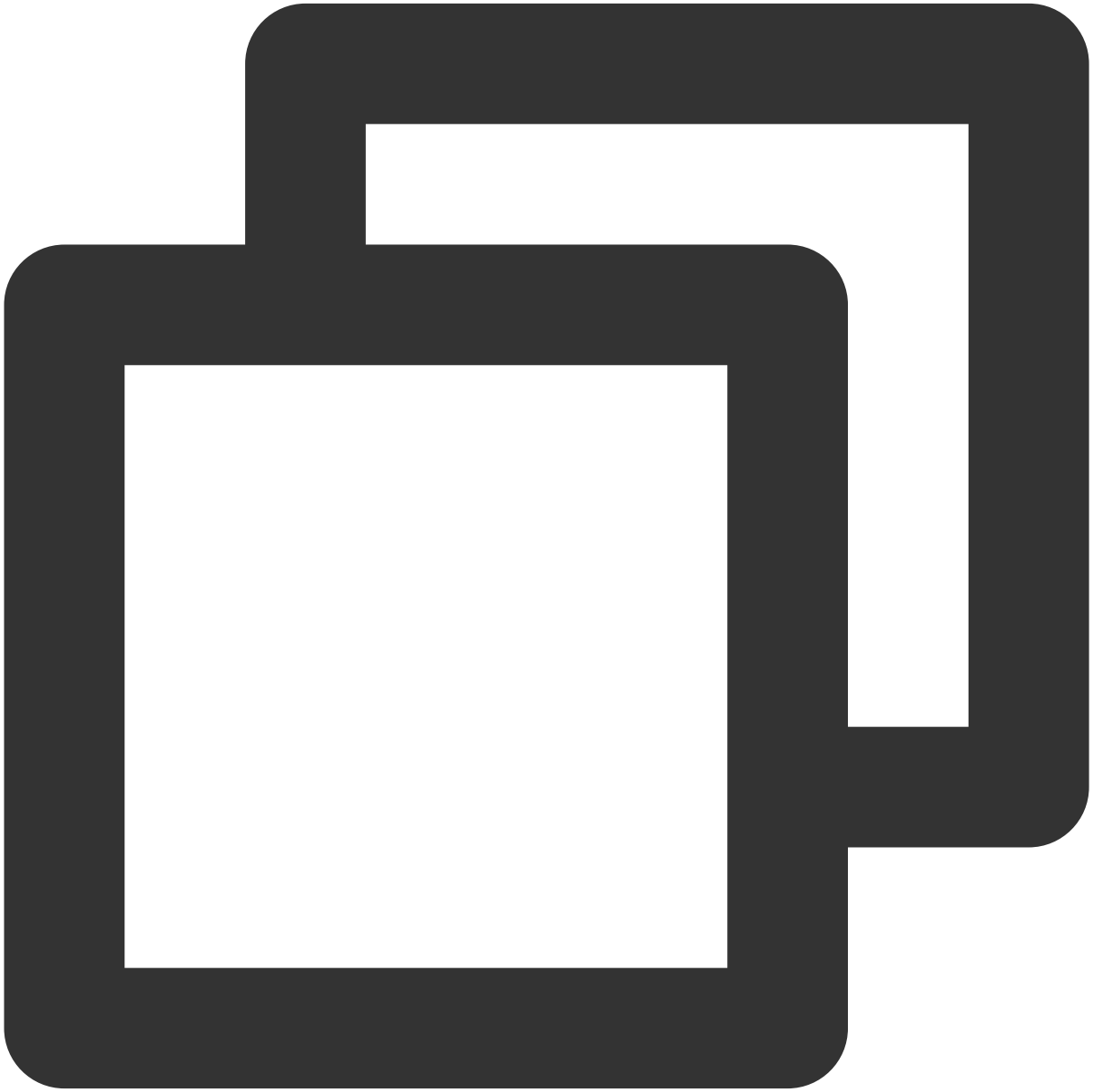
```
void onRemoteUserScreenVideoAvailable(String userId, boolean available);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	ユーザーID。
available	boolean	画面共有ストリームデータの有無。

onRemoteUserAudioAvailable

リモートユーザーがオーディオアップストリームを開始したかどうかのコールバック。



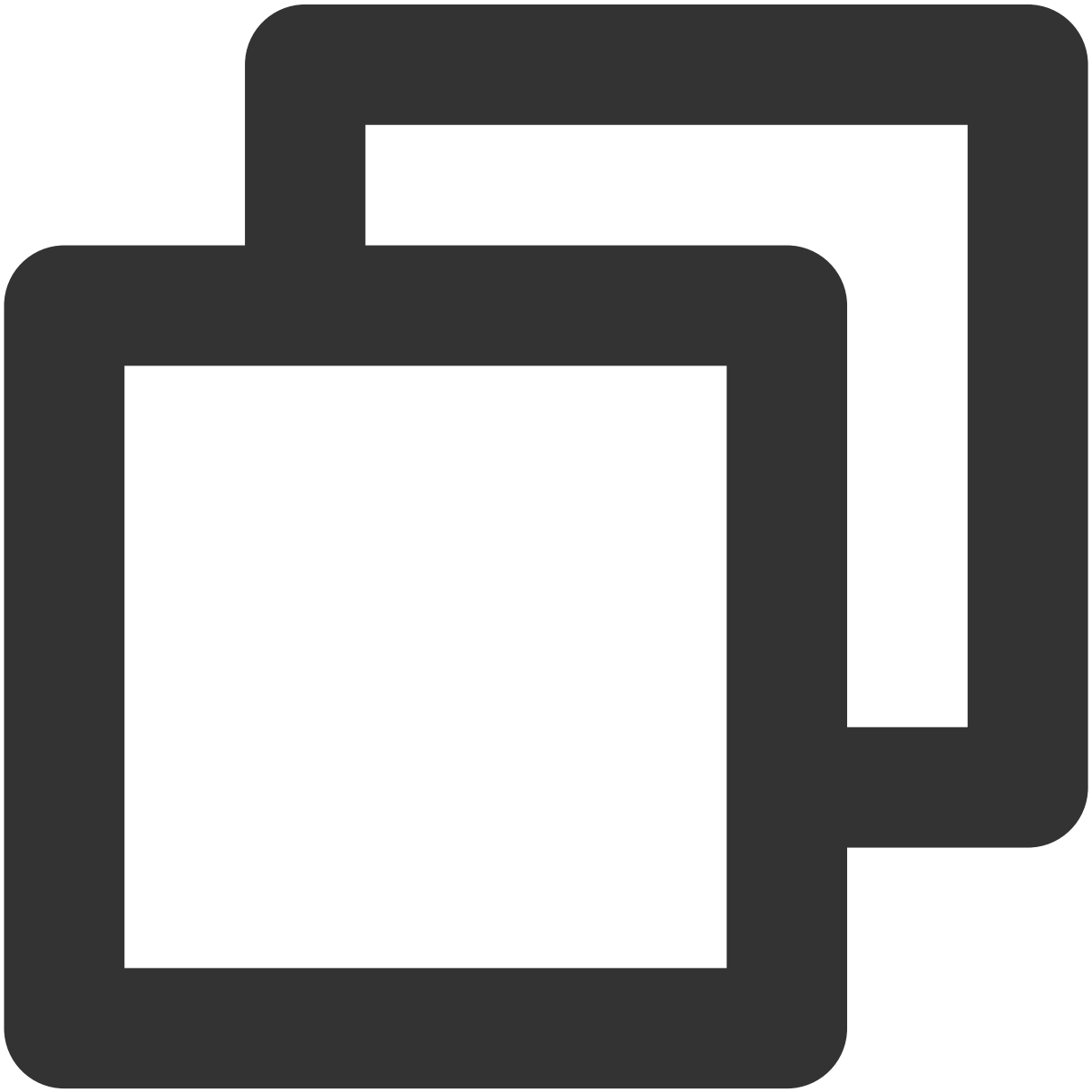
```
void onRemoteUserAudioAvailable(String userId, boolean available);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	ユーザーID。
available	boolean	オーディオデータの有無。

onRemoteUserEnterSpeechState

リモートユーザーが発言を開始します。



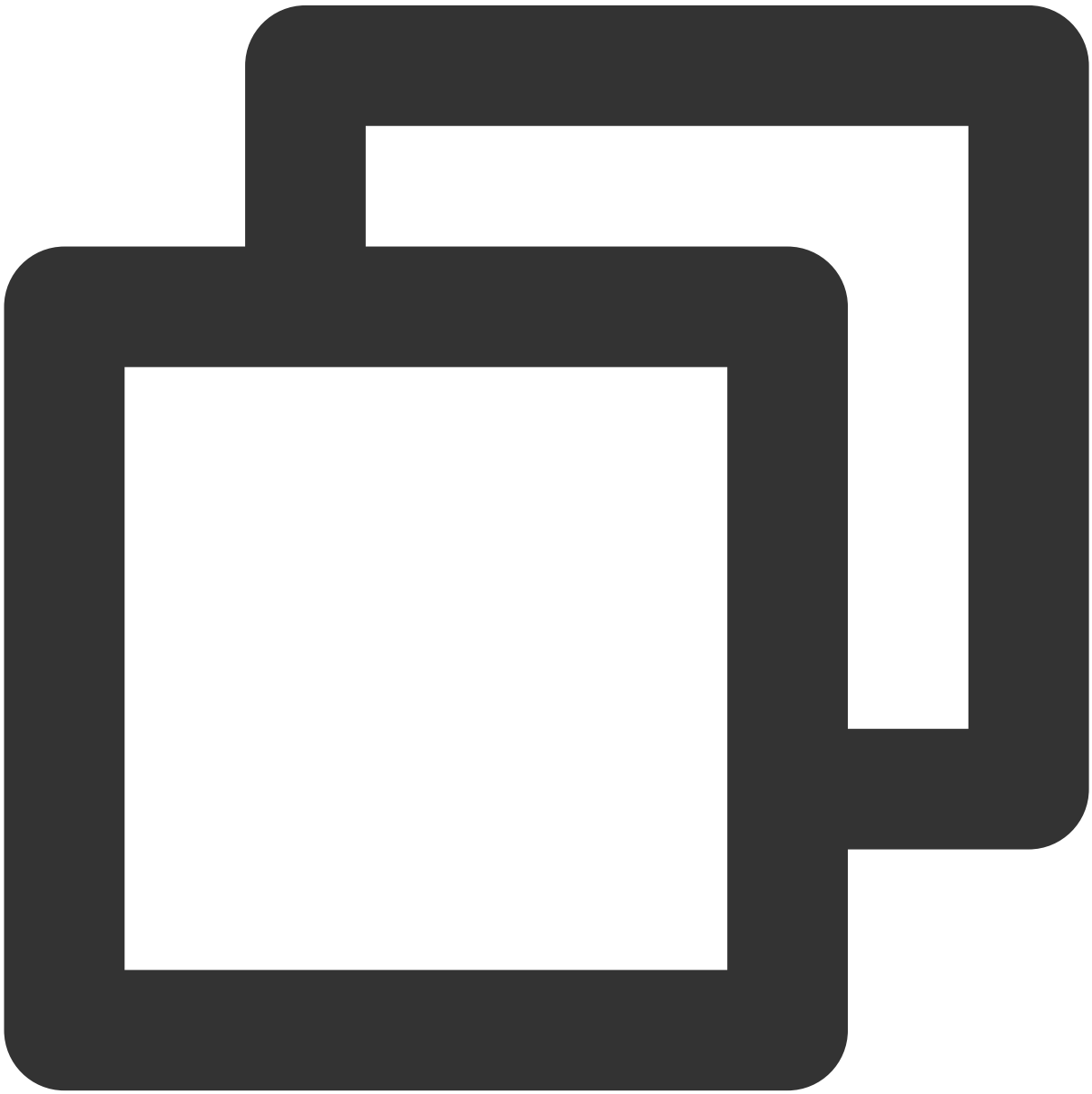
```
void onRemoteUserEnterSpeechState(String userId);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	ユーザーID。

onRemoteUserExitSpeechState

リモートユーザーが発言を終了します。



```
void onRemoteUserExitSpeechState(String userId);
```

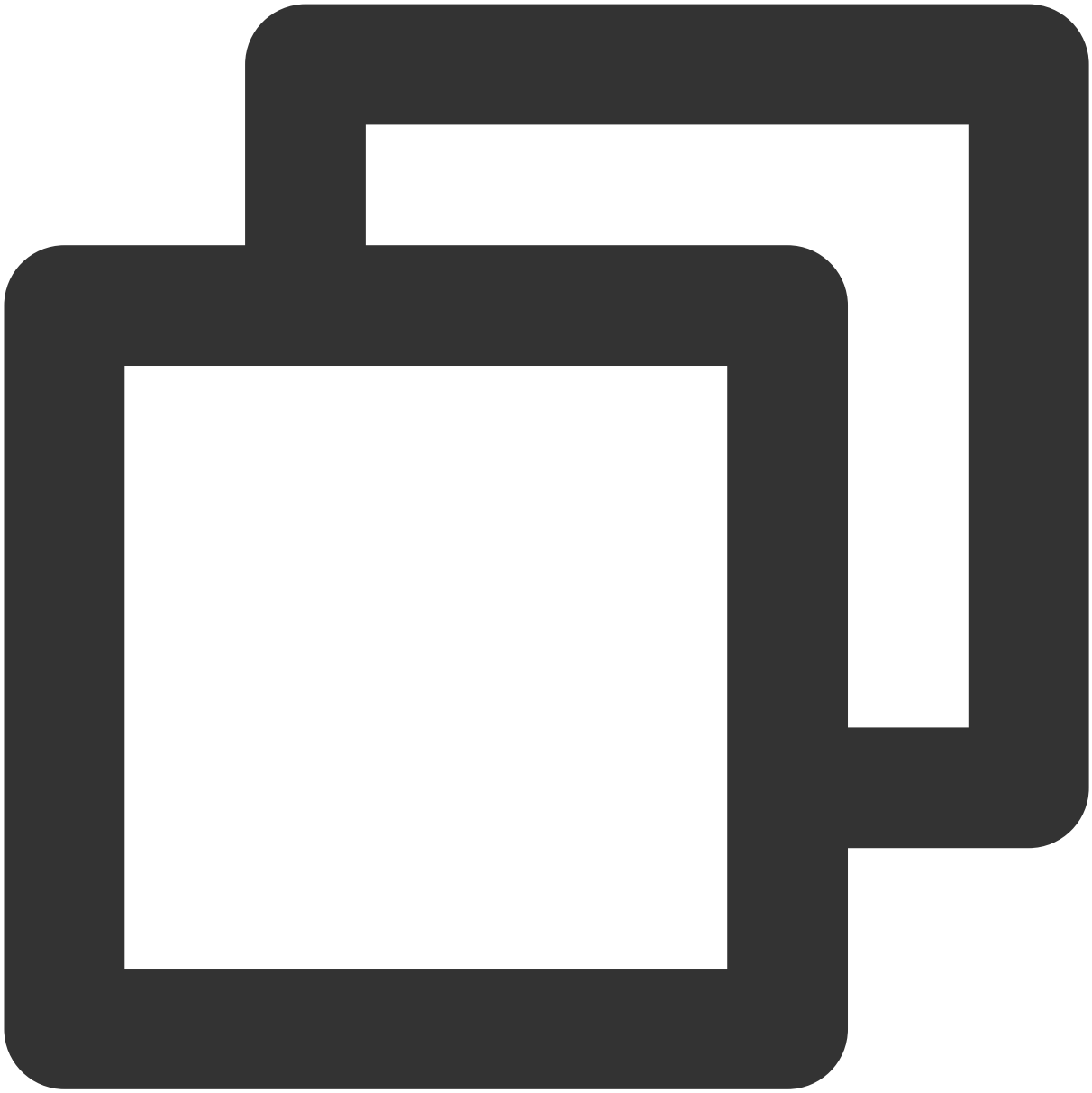
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	ユーザーID。

チャットルームメッセージイベントコールバック

onReceiveChatMessage

テキストメッセージの受信。



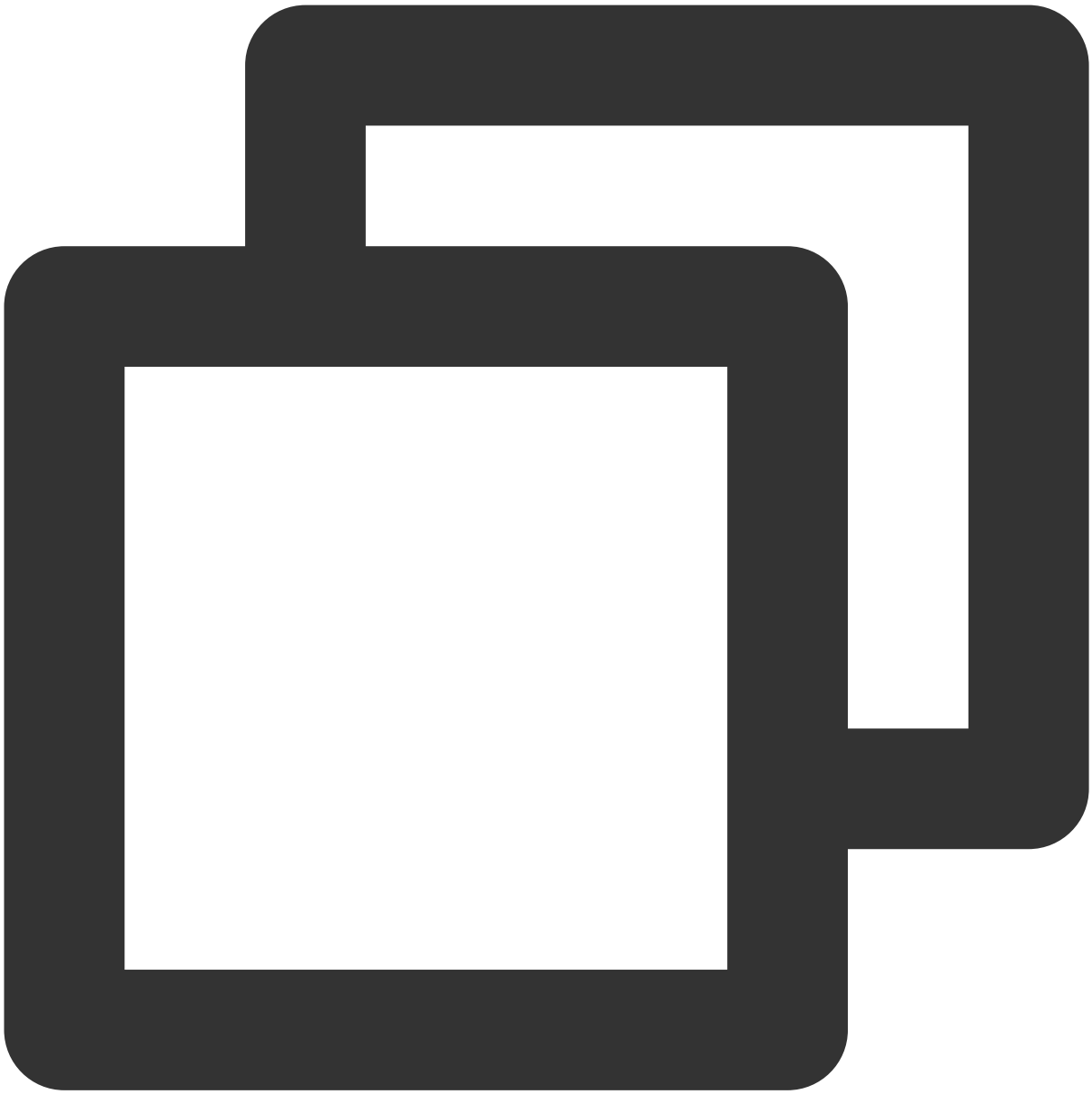
```
void onReceiveChatMessage(String userId, String message);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	ユーザーID。
message	String	テキストメッセージ。

onReceiveRoomCustomMsg

カスタムメッセージの受信。



```
void onReceiveRoomCustomMsg(String userId, String data);
```

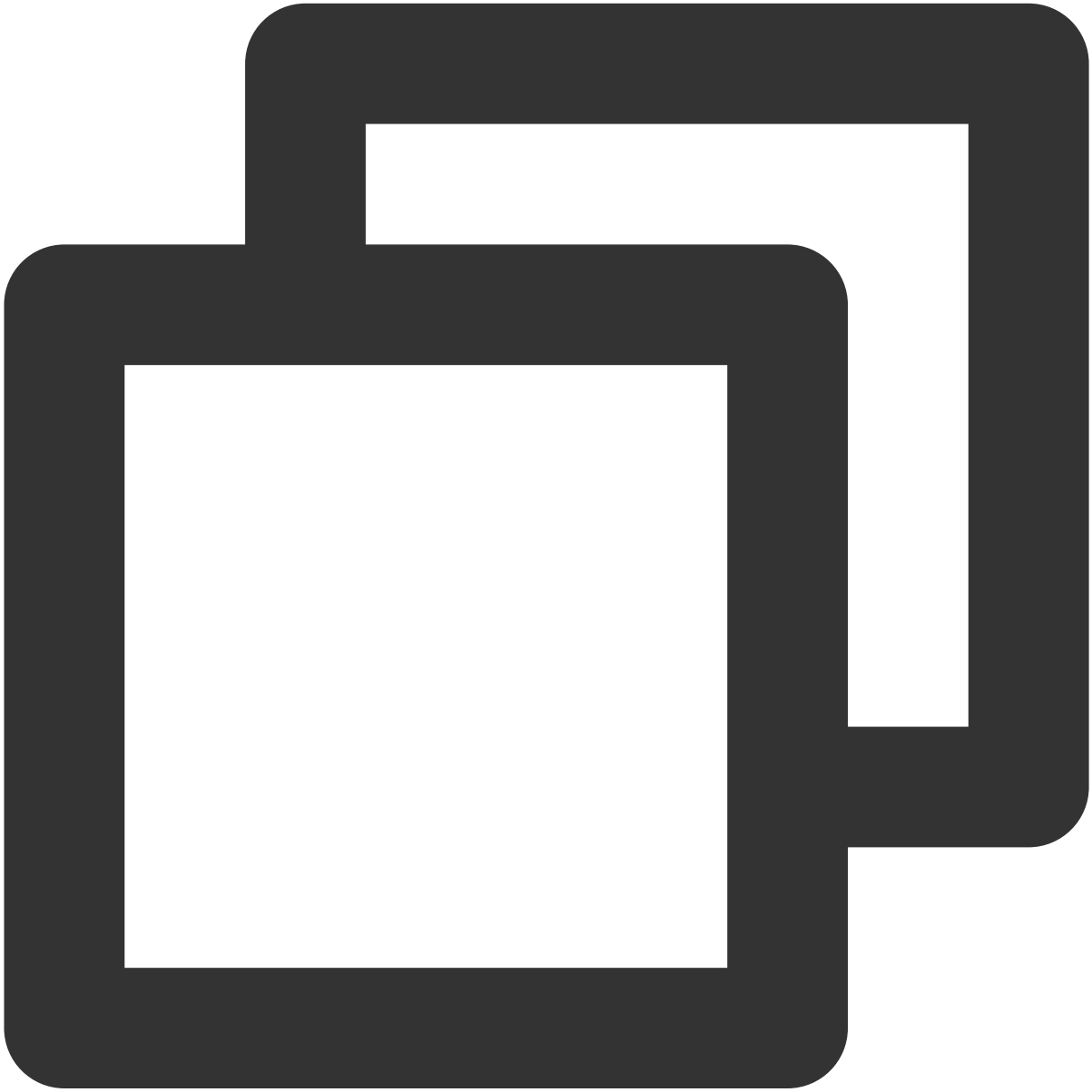
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	ユーザーID。
message	String	カスタムメッセージ。

フィールドコントロールメッセージコールバック

onReceiveSpeechInvitation

ユーザーがキャスターの発言要請を受信する場合のコールバック。



```
void onReceiveSpeechInvitation(String userId);
```

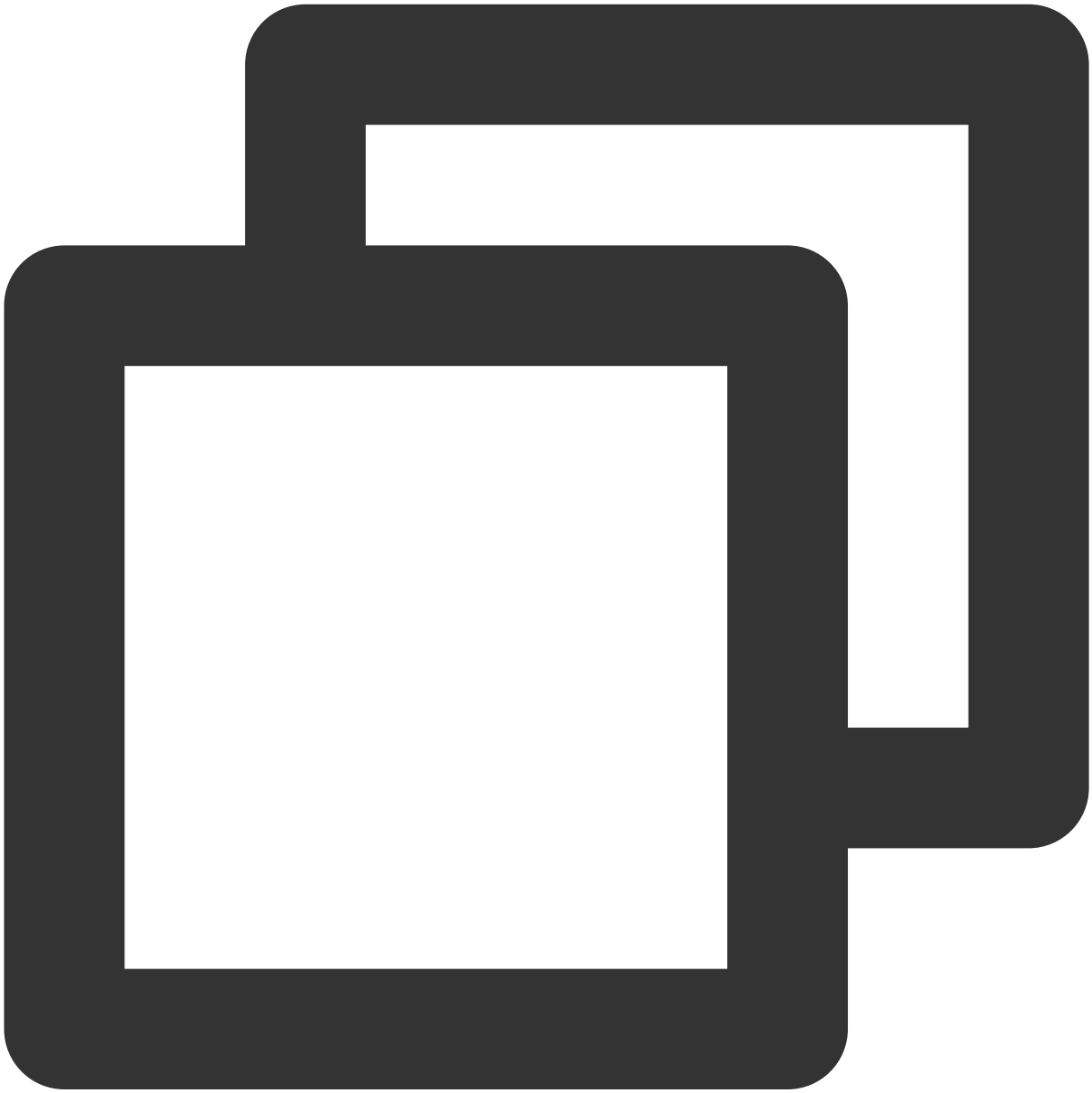
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	String	キャスターユーザーID。

onReceiveInvitationCancelled

ユーザーがキャスターの発言要請キャンセルを受信する場合のコールバック。



```
void onReceiveInvitationCancelled(String userId);
```

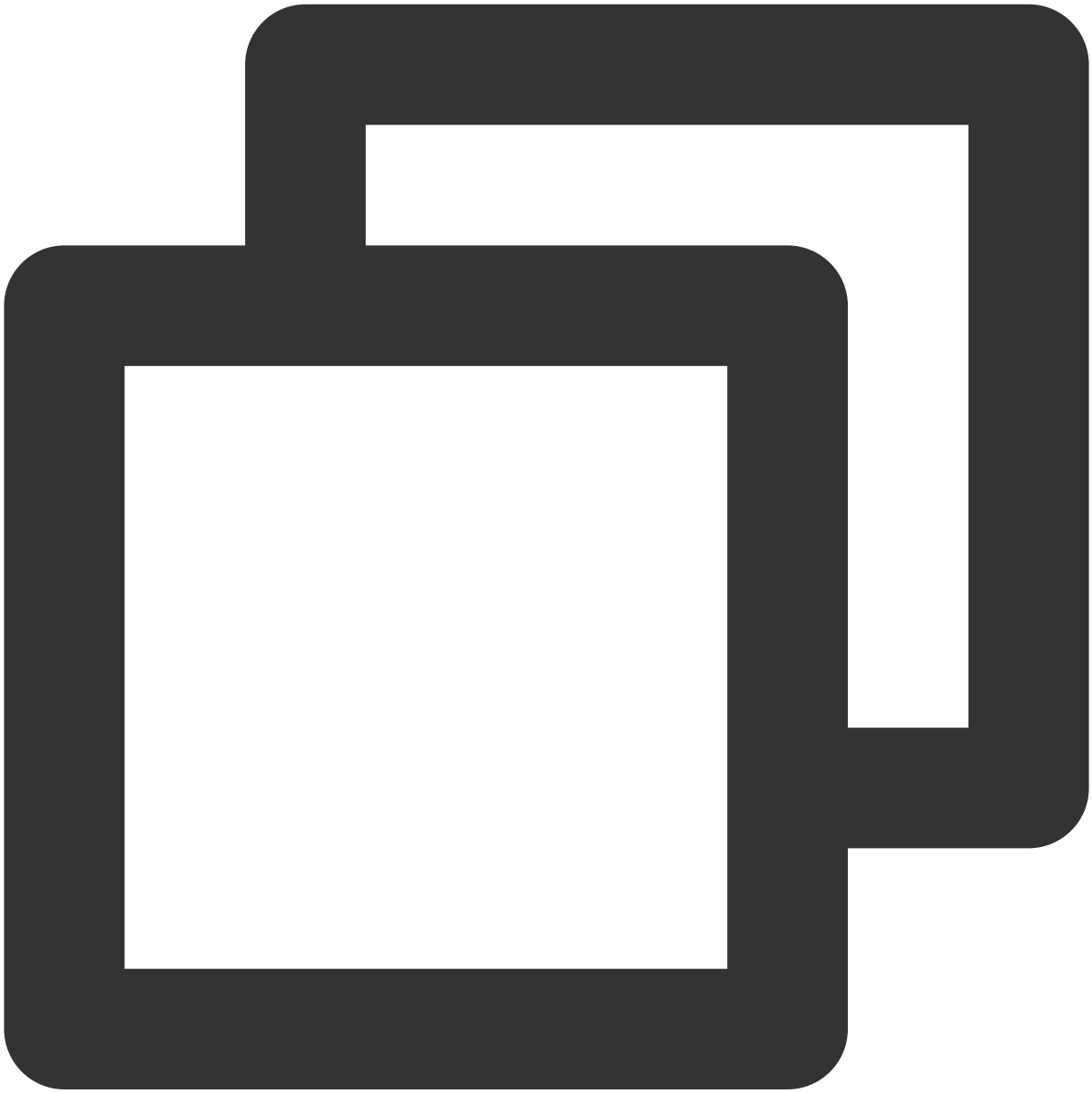
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	String	キャスターユーザーID。

onReceiveSpeechApplication

キャスターがユーザーの発言申請を受信する場合のコールバック。



```
void onReceiveSpeechApplication(String userId);
```

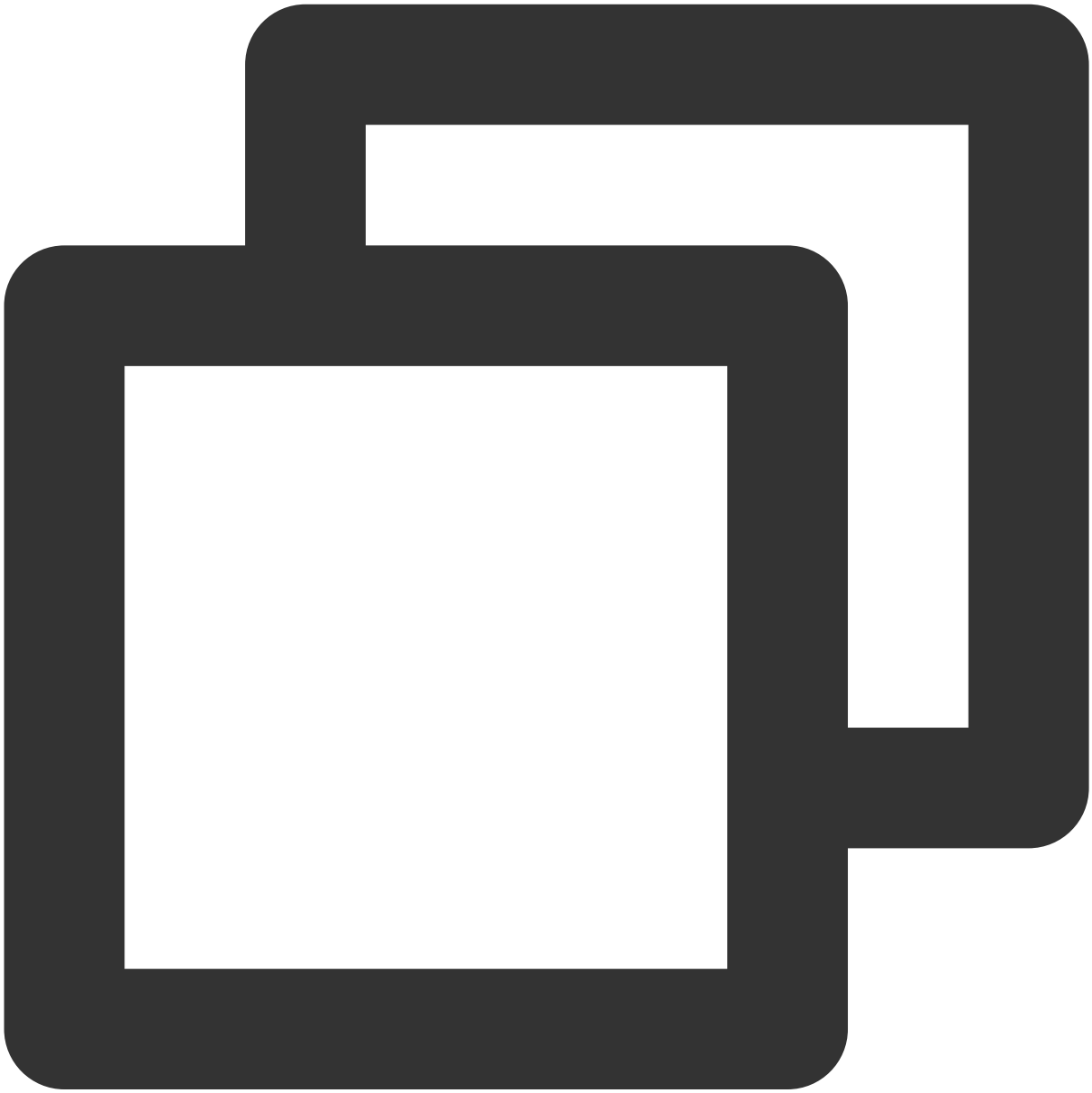
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	String	ユーザーID。

onSpeechApplicationCancelled

ユーザーが発言申請をキャンセルする場合のコールバック。



```
void onSpeechApplicationCancelled(String userId);
```

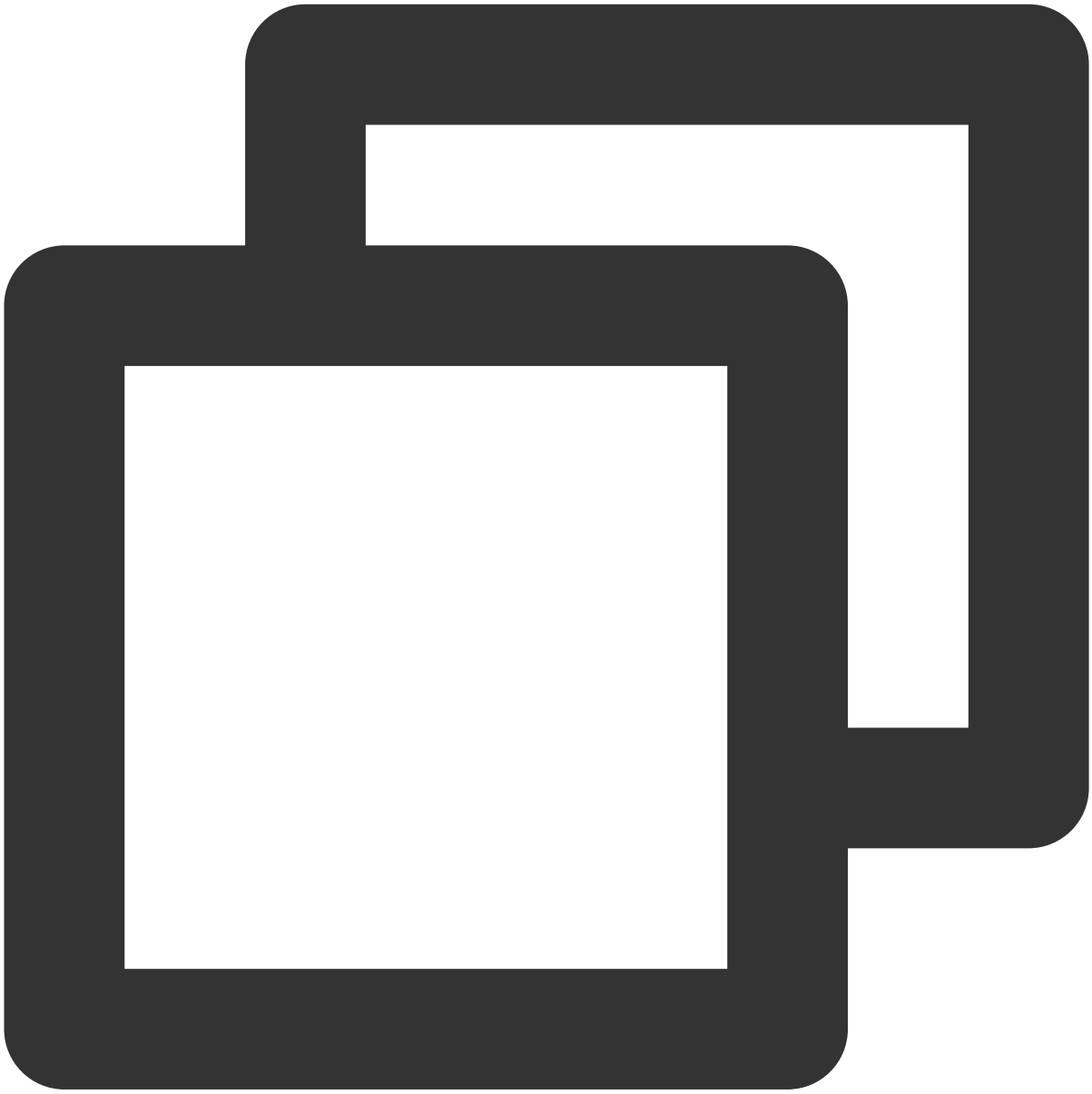
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	String	ユーザーID。

onSpeechApplicationForbidden

キャスターが発言申請を禁止する場合のコールバック。



```
void onSpeechApplicationForbidden(boolean isForbidden);
```

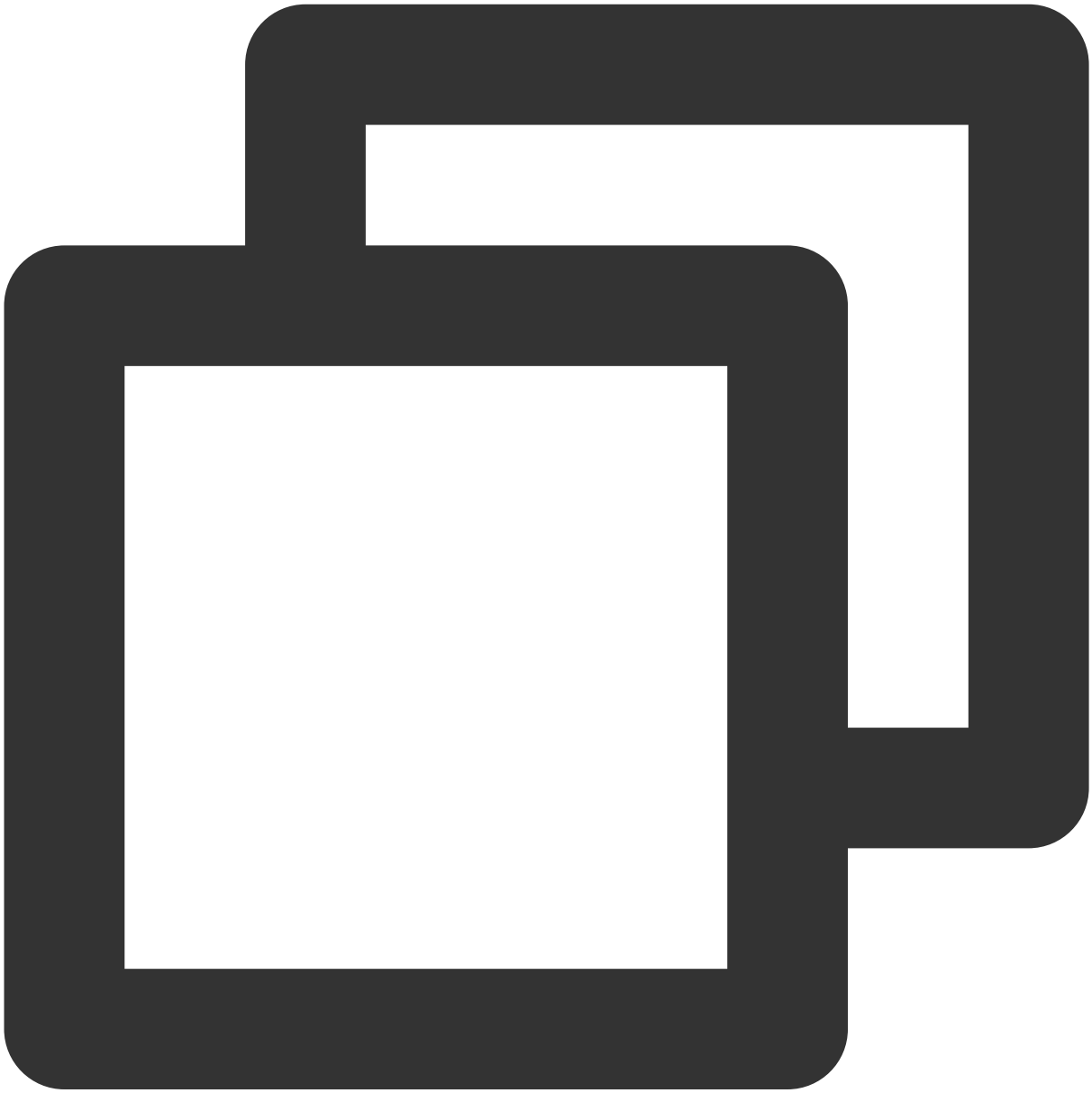
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

isForbidden	boolean	禁止するかどうか。

onOrderedToExitSpeechState

参加者が発言を停止するようリクエストされる場合のコールバック。



```
void onOrderedToExitSpeechState(String userId);
```

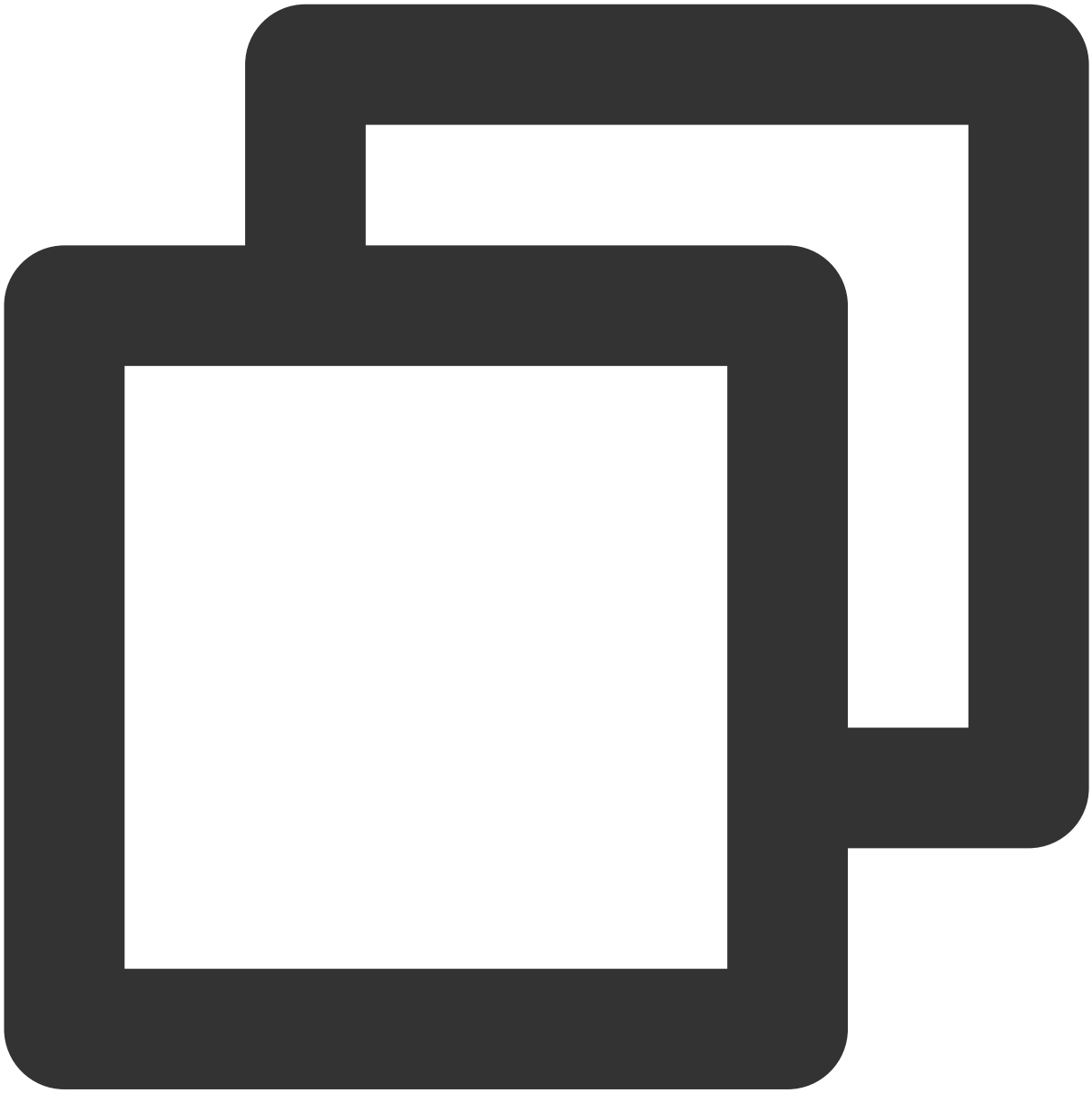
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	String	キャスターユーザーID。

onCallingRollStarted

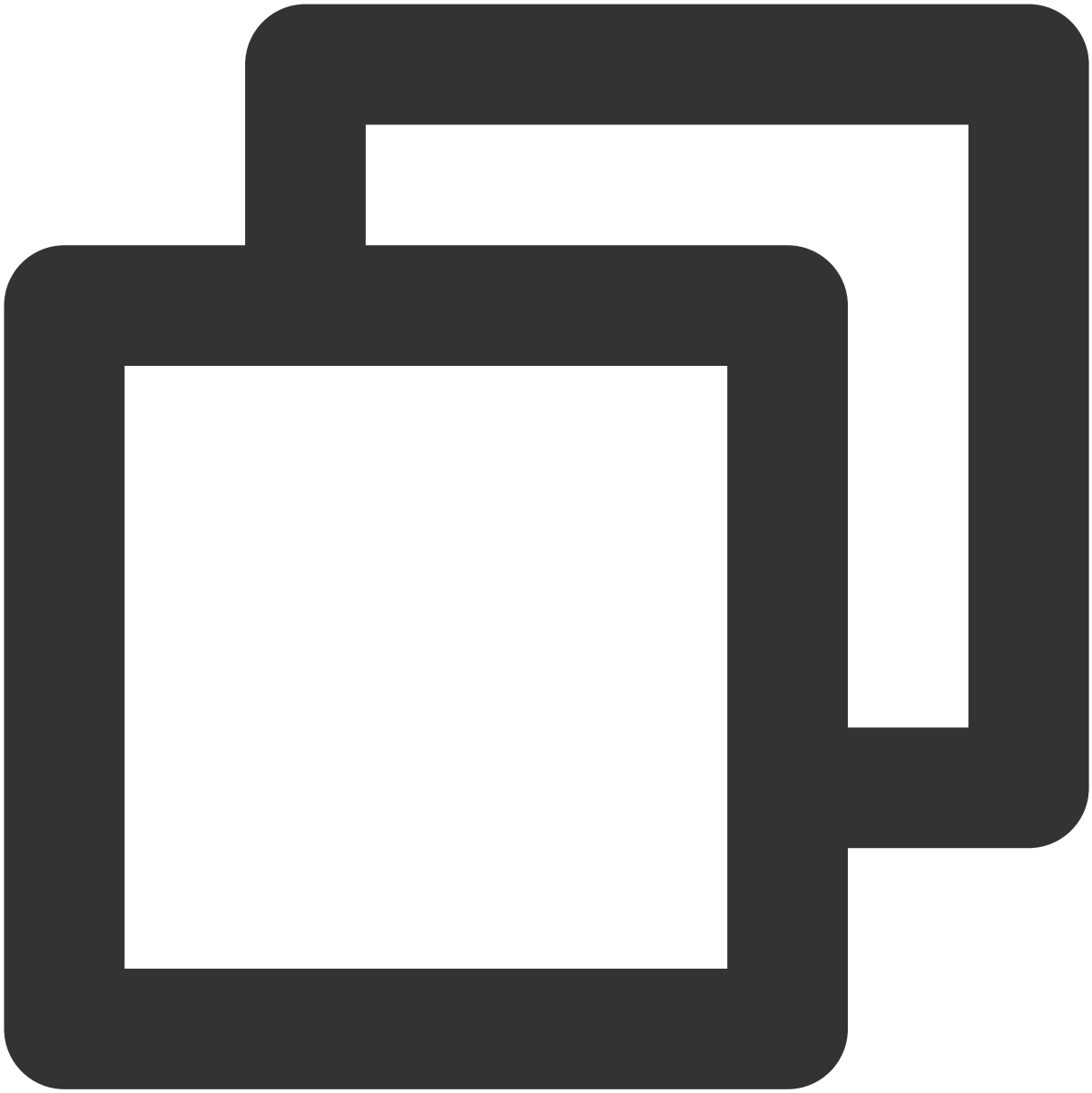
キャスターが点呼を開始し、参加者が受信する場合のコールバック。



```
void onCallingRollStarted(String userId);
```

onCallingRollStopped

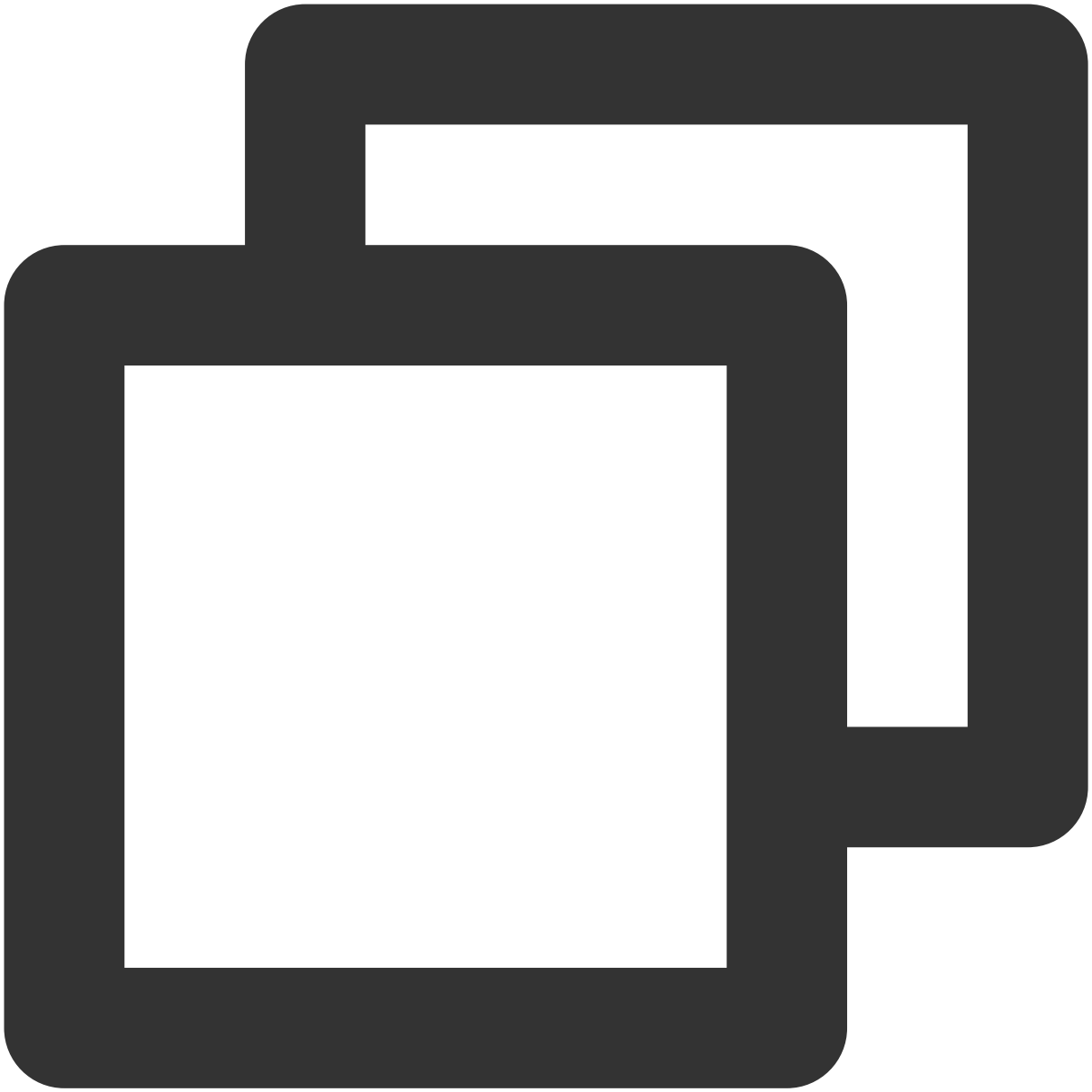
キャスターが点呼を終了し、参加者が受信する場合のコールバック。



```
void onCallingRollStopped(String userId);
```

onMemberReplyCallingRoll

参加者が点呼に応答し、キャスターが受信する場合のコールバック。



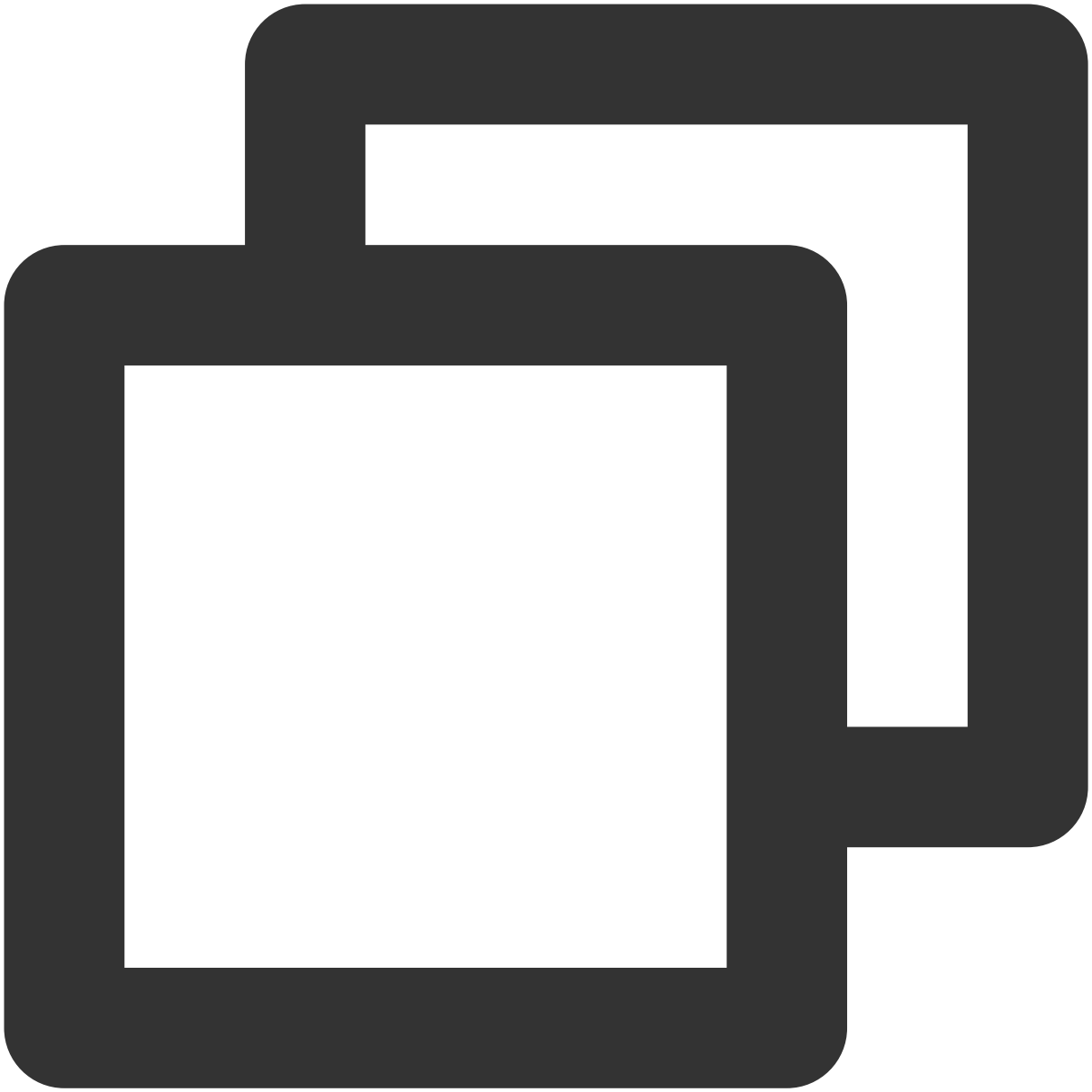
```
void onMemberReplyCallingRoll(String userId);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	ユーザーID。

onChatRoomMuted

キャスターがチャットルームのミュートを変更する場合のコールバック。



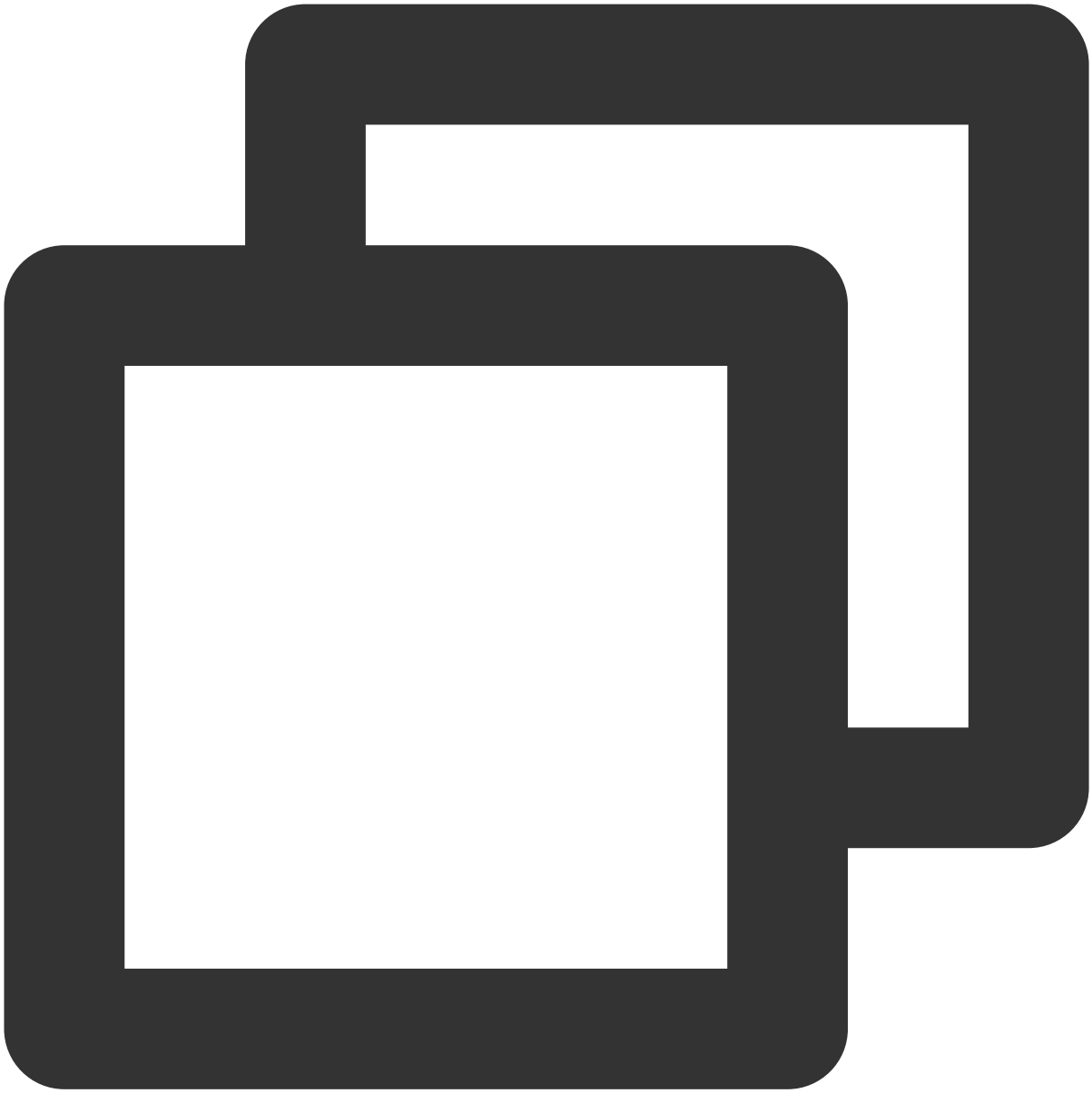
```
void onChatRoomMuted(boolean muted);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
muted	boolean	無効にするかどうか。

onMicrophoneMuted

キャスターがマイクの無効化を設定する場合のコールバック。



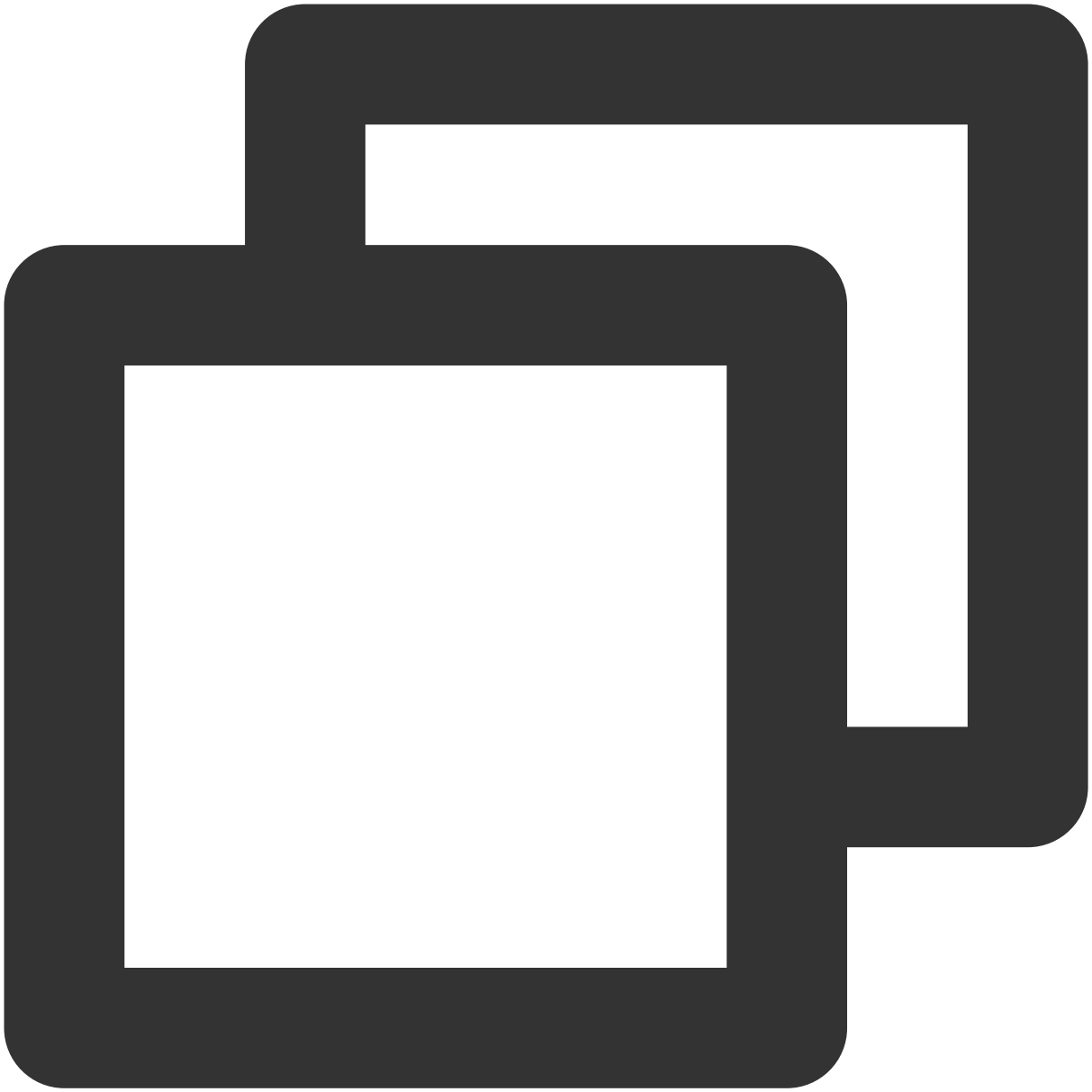
```
void onMicrophoneMuted(boolean muted);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
muted	boolean	無効にするかどうか。

onCameraMuted

キャスターがカメラの無効化を設定する場合のコールバック。



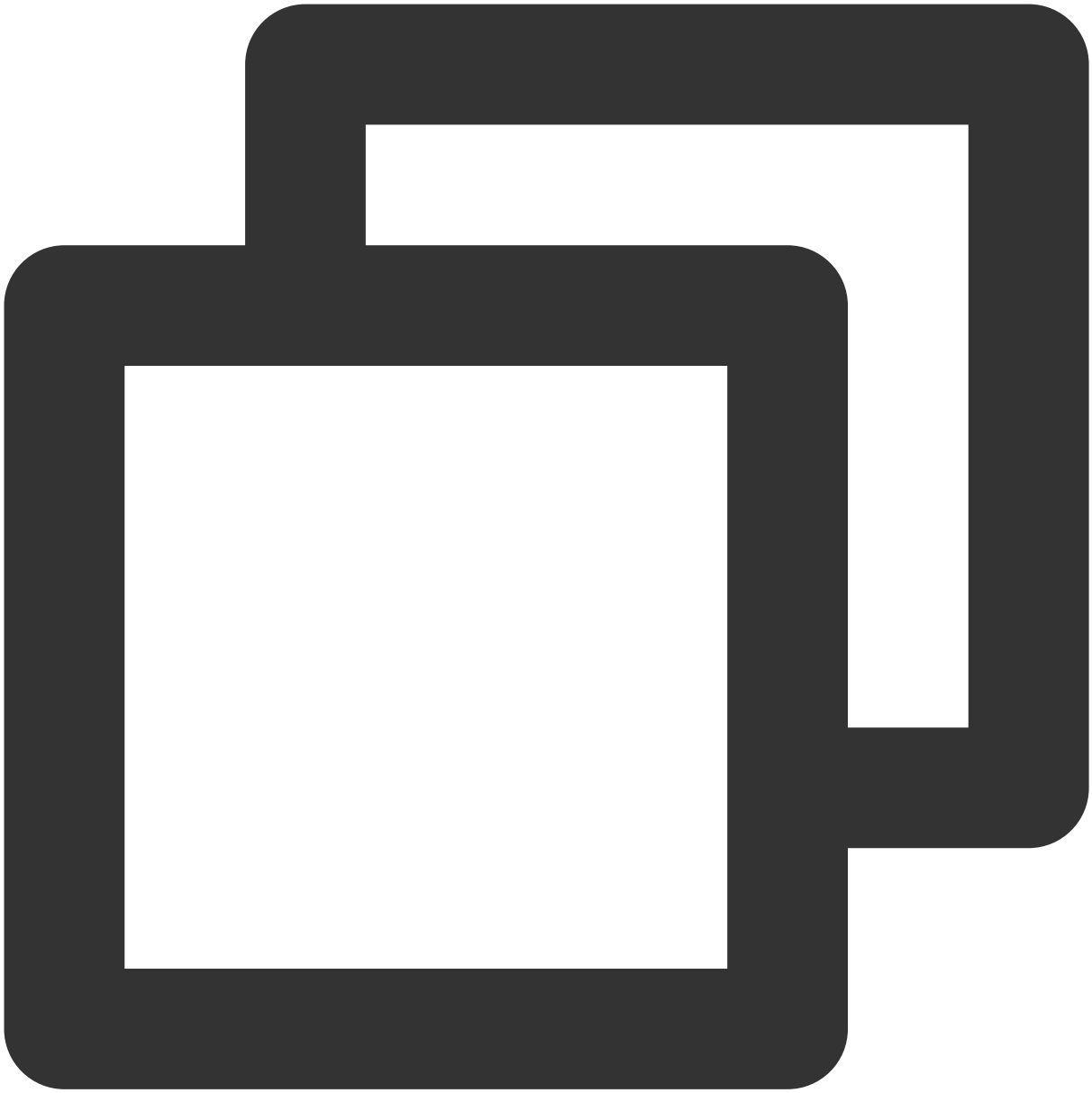
```
void onCameraMuted(boolean muted);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
muted	boolean	無効にするかどうか。

onReceiveKickedOff

キャスターによるキックアウトのコールバック。



```
void onReceiveKickedOff(String userId);
```

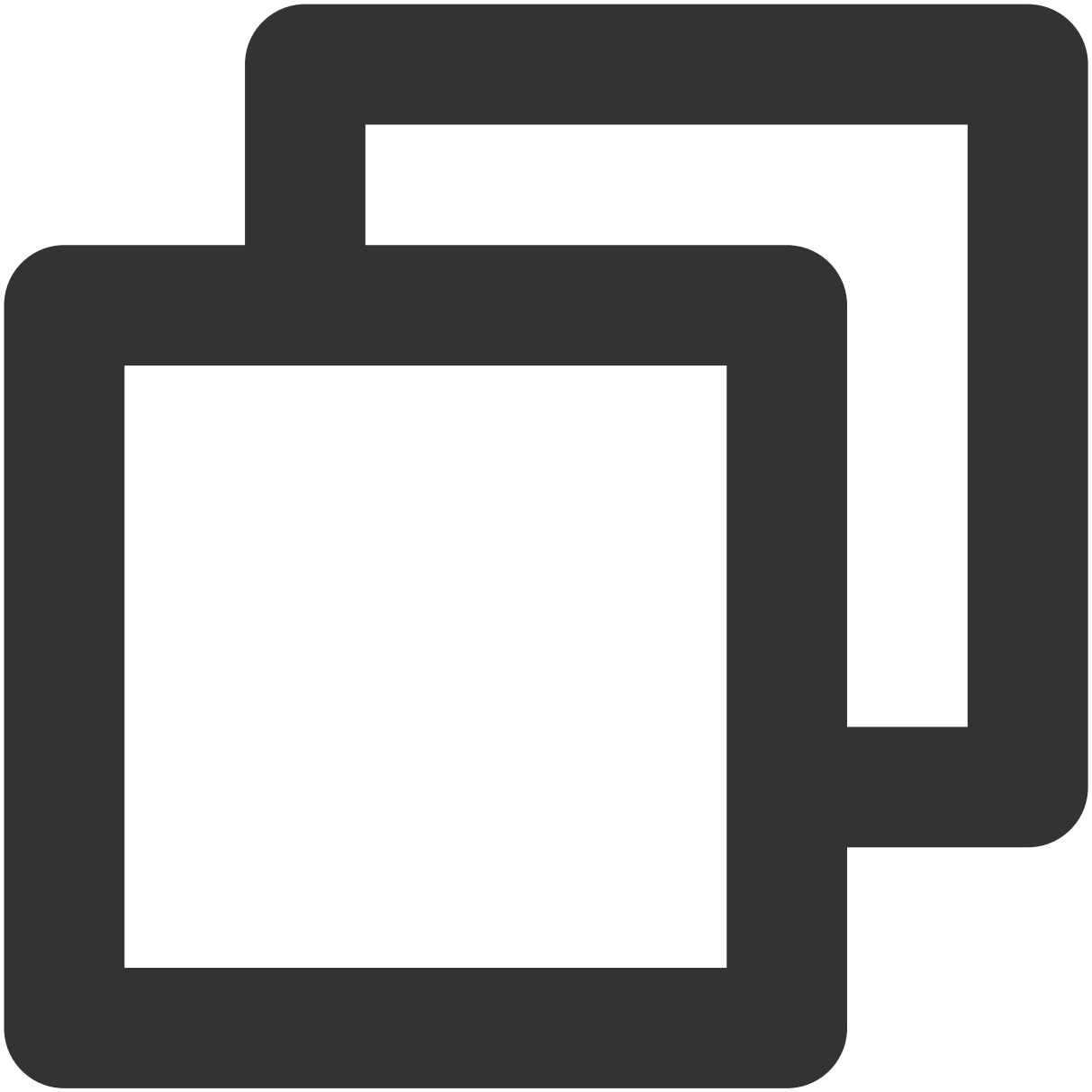
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	キャスター/管理者ユーザーID。

統計および品質コールバック

onStatistics

技術指標統計のコールバック。



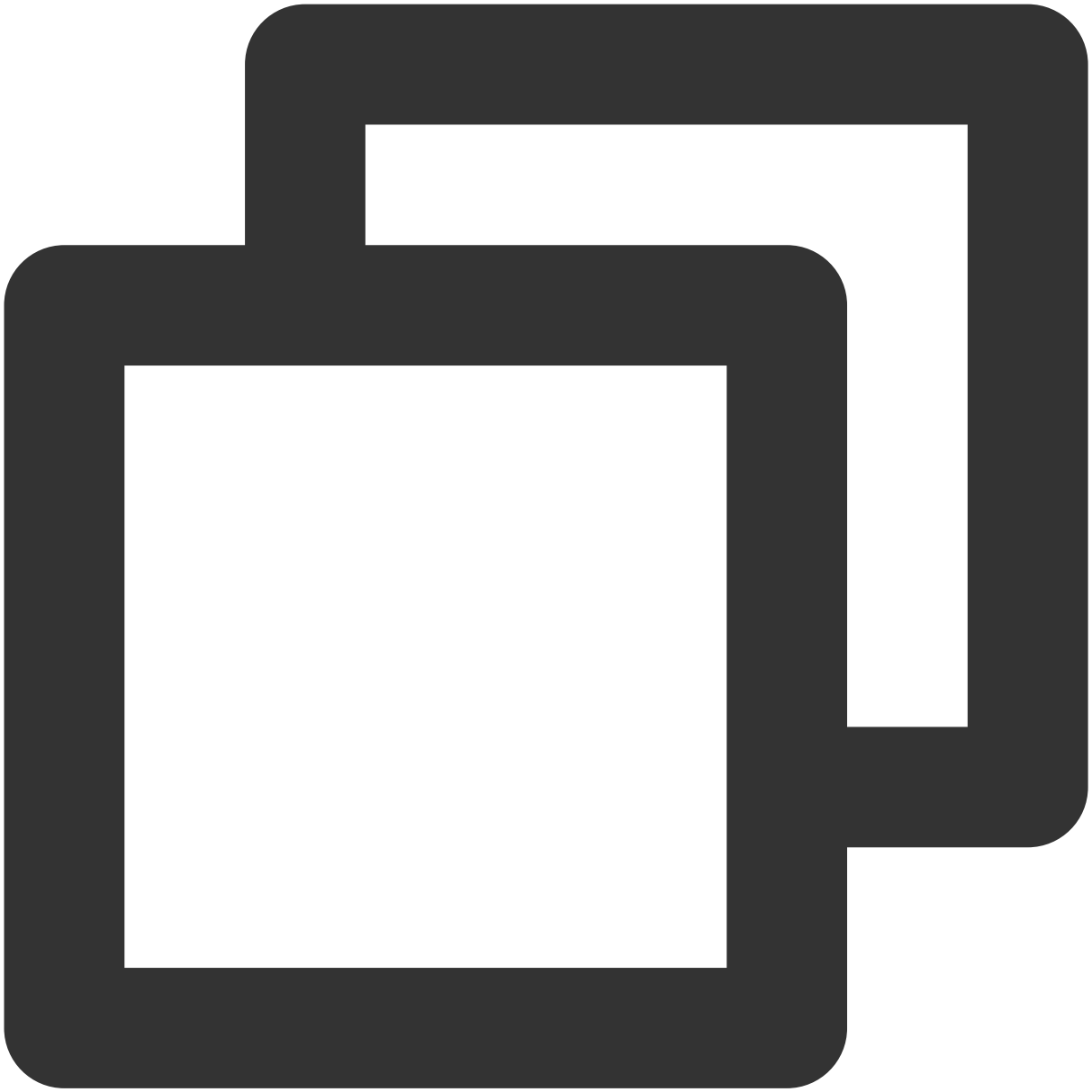
```
void onStatistics(TRTCStatistics statistics);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
statis	TRTCStatistics	統計データ。

onNetworkQuality

ネットワーク状況のコールバック。



```
void onNetworkQuality(TRTCCloudDef.TRTCQuality localQuality, List<TRTCCloudDef.TRTC
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
localQuality	TRTCCloudDef.TRTCQuality	アップストリームネットワークの品

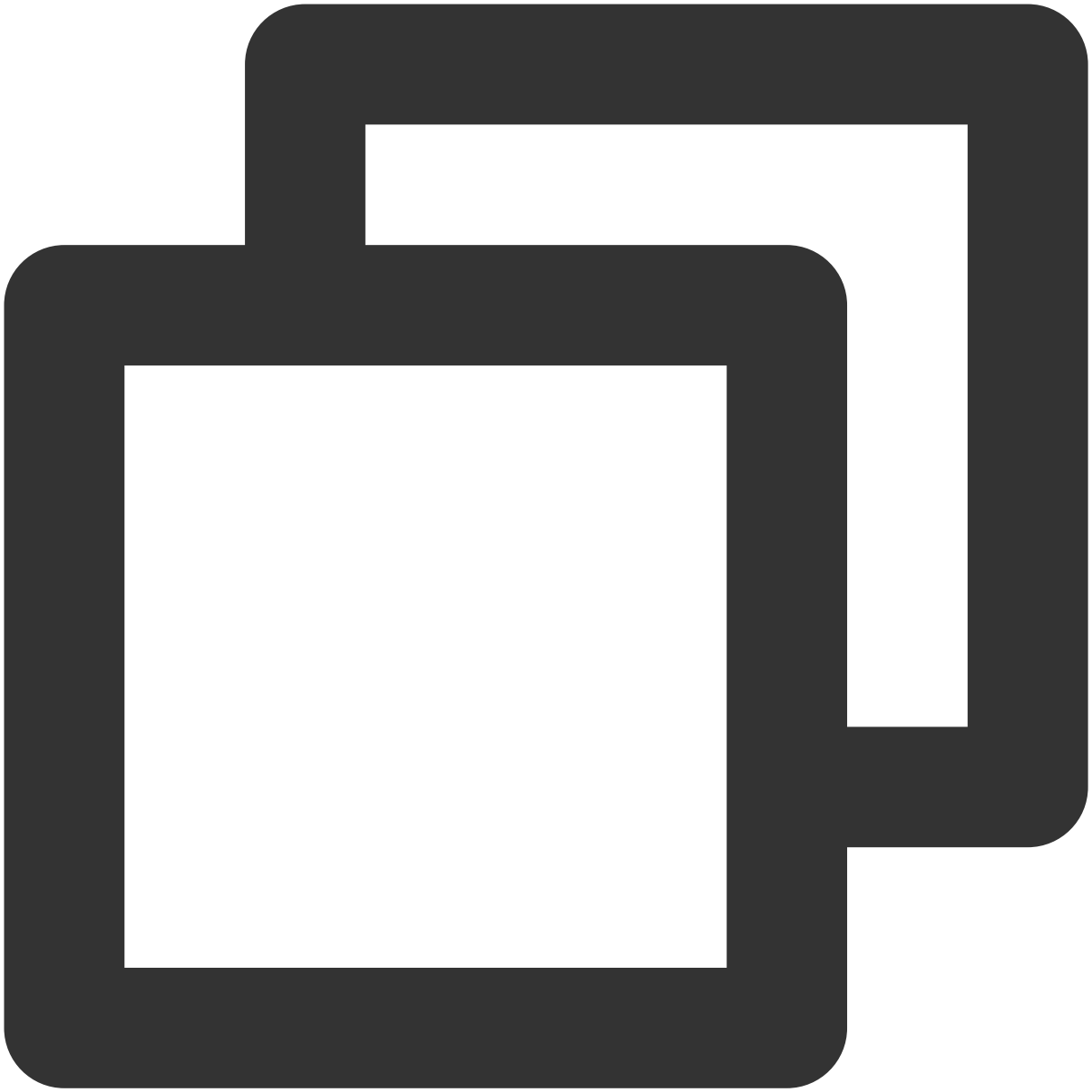
		質。
remoteQuality	List<TRTCCloudDef.TRTCQuality>	ダウンストリームのネットワーク品質。

説明：
詳細については、 TRTC SDKをご参照ください。

画面共有イベントコールバック

onScreenCaptureStarted

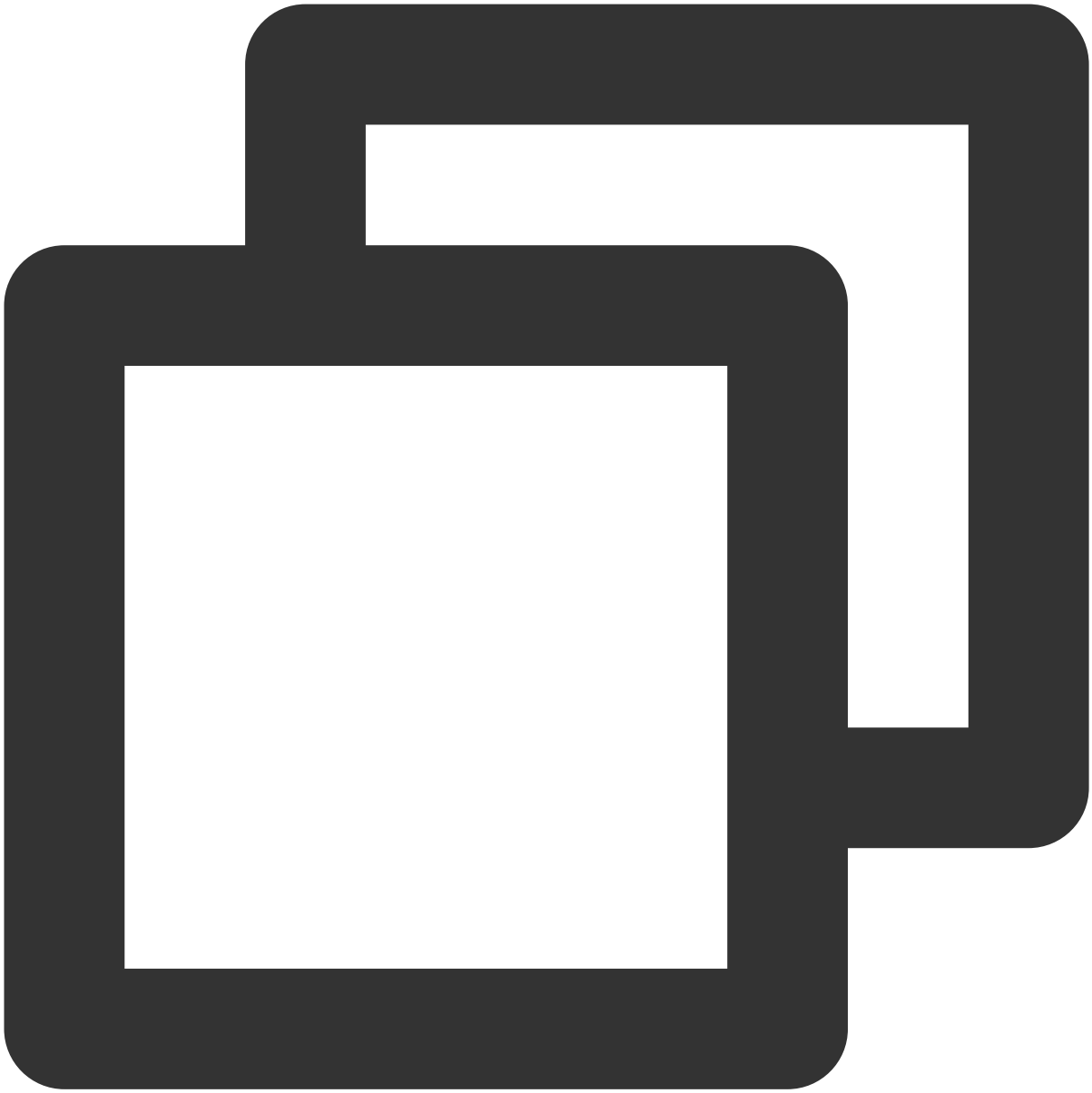
画面共有開始のコールバック。



```
void onScreenCaptureStarted();
```

onScreenCaptureStopped

画面共有停止のコールバック。



```
void onScreenCaptureStopped(int reason);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
reason	int	停止の理由。 0：ユーザーの自発的な停止。 1：その他アプリケーションに占有されたことによる停止。

TUIRoom API（iOS）

最終更新日：：2024-07-19 15:35:35

TUIRoomは、Tencent CloudのTencent Real-Time Communication（TRTC）およびIMサービスを基に組み合わせたコンポーネントで、以下の機能をサポートしています。
キャスターがルームを作成し、入室者はルームナンバーを入力した後に入室できます。
入室者の間で画面共有を行います。
各種のテキストメッセージとカスタムメッセージの送信をサポートします。
説明：
TUIKitシリーズコンポーネントはTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期してアクティブ化することができます。 IMサービスの課金ルールの詳細については、Instant Messagingの料金説明をご参照ください。TRTCをアクティブ化すると、関連するIM SDKの体験版がデフォルトでアクティブ化されます。これは100 DAUのみをサポートします。
TUIRoomはオープンソースのClassであり、Tencent Cloudの2つのクローズドソースのSDKに依存しています。具体的な実装プロセスについては、 多人数オーディオビデオルーム(iOS)をご参照ください。
TRTC SDK：TRTC SDKを低遅延のオーディオビデオルームコンポーネントとして使用します。
IM SDK：IM SDK を利用してチャットルームの機能（**IM SDKはiOSバージョンを使用**）を実装します。

TUIRoom API概要

TUIRoomCore基本関数

API	説明
shareInstance	シングルトンオブジェクトを取得します。
destroyInstance	シングルトンオブジェクトを破棄します。
setDelegate	イベントコールバックを設定します。

ルーム関連インターフェース関数

API	説明
createRoom	ルームの作成（キャスターが呼び出し）。
destroyRoom	ルームの破棄（キャスターが呼び出し）。
enterRoom	入室（参加者が呼び出し）。

leaveRoom	退室（参加者が呼び出し）。
getRoomInfo	ルーム情報の取得。
getRoomUsers	ルーム内全メンバー情報の取得。
getUserInfo	特定ユーザーの情報の取得。
transferRoomMaster	キャスター権限の移転（キャスターが呼び出し）。

ローカルのオーディオビデオ操作インターフェース

API	説明
startCameraPreview	ローカルビデオのプレビュー画面を立ち上げます。
stopCameraPreview	ローカルのビデオキャプチャおよびプレビューを停止します。
startLocalAudio	マイクキャプチャを起動します。
stopLocalAudio	マイクキャプチャを停止します。
setVideoMirror	ローカル画面のイメージプレビューモードを設定します。
setSpeaker	スピーカーの起動を設定します。

リモートユーザーに関するインターフェース

API	説明
startRemoteView	指定メンバーのリモートビデオ画面をサブスクリプションし再生します。
stopRemoteView	リモートビデオ画面のサブスクリプションをキャンセルし再生を停止します。

チャットメッセージ送信インターフェース

API	説明
sendChatMessage	チャットメッセージを送信します。
sendCustomMessage	カスタムメッセージを送信します。

フィールドコントロール関連インターフェース

API	説明

muteUserMicrophone	特定ユーザーのマイクを無効化/再有効化します。
muteAllUsersMicrophone	全ユーザーのマイクを無効化/再有効化し、ステータスをルーム情報に同期させます。
muteUserCamera	特定ユーザーのカメラを無効化/再有効化します。
muteAllUsersCamera	全ユーザーのカメラを無効化/再有効化し、ステータスをルーム情報に同期させます。
muteChatRoom	チャットルームのミュートを開始/停止します（キャスターが呼び出し）。
kickOffUser	ルーム内の特定ユーザーをリムーブします（キャスターが呼び出し）。
startCallingRoll	キャスターが点呼を開始します。
stopCallingRoll	キャスターが点呼を終了します。
replyCallingRoll	参加者がキャスターの点呼に応答します。
sendSpeechInvitation	キャスターが参加者の発言を要請します。
cancelSpeechInvitation	キャスターが参加者の発言要請をキャンセルします。
replySpeechInvitation	参加者がキャスターの発言申請に同意/拒否します。
sendSpeechApplication	参加者が発言を申請します。
replySpeechApplication	キャスターが参加者の発言申請に同意/拒否します。
forbidSpeechApplication	キャスターが発言申請を禁止します。
sendOffSpeaker	キャスターが参加者に発言を停止するよう命令します。
sendOffAllSpeakers	キャスターが全員に発言を停止するよう命令します。
exitSpeechState	参加者は発言を停止し、視聴者になります。

画面共有インターフェース

API	説明
startScreenCapture	画面共有を開始。
stopScreenCapture	画面キャプチャの停止。

美顔フィルターに関するインターフェース関数

API	説明
getBeautyManager	美顔管理オブジェクトTXBeautyManagerを取得します。

関連設定インターフェース

API	説明
setVideoQosPreference	ネットワークトラフィックコントロール関連パラメータを設定します。

SDKバージョンインターフェース関数の取得

API	説明
getSDKVersion	SDKバージョンを取得します。

TUIRoomCoreDelegate API概要

エラーイベントコールバック

API	説明
onError	エラーのコールバック。

基本イベントコールバック

API	説明
onDestroyRoom	ルーム解散のコールバック。
onUserVoiceVolume	音量の大きさのコールバック。
onRoomMasterChanged	キャスター変更のコールバック。

リモートユーザーイベントコールバック

API	説明
onRemoteUserEnter	リモートユーザー入室コールバック。
onRemoteUserLeave	リモートユーザー退室コールバック。
onRemoteUserCameraAvailable	リモートユーザーがカメラビデオを起動するかどうかのコールバッ

	ク。
onRemoteUserScreenVideoAvailable	リモートユーザーが画面共有を開始するかどうかのコールバック。
onRemoteUserAudioAvailable	リモートユーザーがオーディオのアップストリームを開始するかどうかのコールバック。
onRemoteUserEnterSpeechState	リモートユーザーの発言開始のコールバック。
onRemoteUserExitSpeechState	リモートユーザーの発言終了のコールバック。

メッセージイベントのコールバック

API	説明
onReceiveChatMessage	テキストメッセージ受信のコールバック。

フィールドコントロールイベントコールバック

API	説明
onReceiveSpeechInvitation	ユーザーがキャスターの発言要請を受信した場合のコールバック。
onReceiveInvitationCancelled	ユーザーがキャスターの発言要請キャンセルを受信する場合のコールバック。
onReceiveSpeechApplication	キャスターがユーザーの発言申請を受信する場合のコールバック。
onSpeechApplicationCancelled	ユーザーが発言申請をキャンセルする場合のコールバック。
onSpeechApplicationForbidden	キャスターが発言申請を禁止する場合のコールバック。
onOrderedToExitSpeechState	参加者が発言の停止をリクエストされる場合のコールバック。
onCallingRollStarted	キャスターが点呼を開始し、参加者が受信する場合のコールバック。
onCallingRollStopped	キャスターが点呼を終了し、参加者が受信する場合のコールバック。
onMemberReplyCallingRoll	参加者が点呼に応答し、キャスターが受信する場合のコールバック。
onChatRoomMuted	キャスターがチャットルームのミュートを変更する場合のコールバック。
onMicrophoneMuted	キャスターがマイクの無効化を設定する場合のコールバック。
onCameraMuted	キャスターがカメラの無効化を設定する場合のコールバック。
onReceiveKickedOff	参加者がキャスターからキックアウトされた場合のコールバック。

統計および品質コールバック

API	説明
onStatistics	技術指標統計のコールバック。
onNetworkQuality	ネットワーク品質のコールバック。

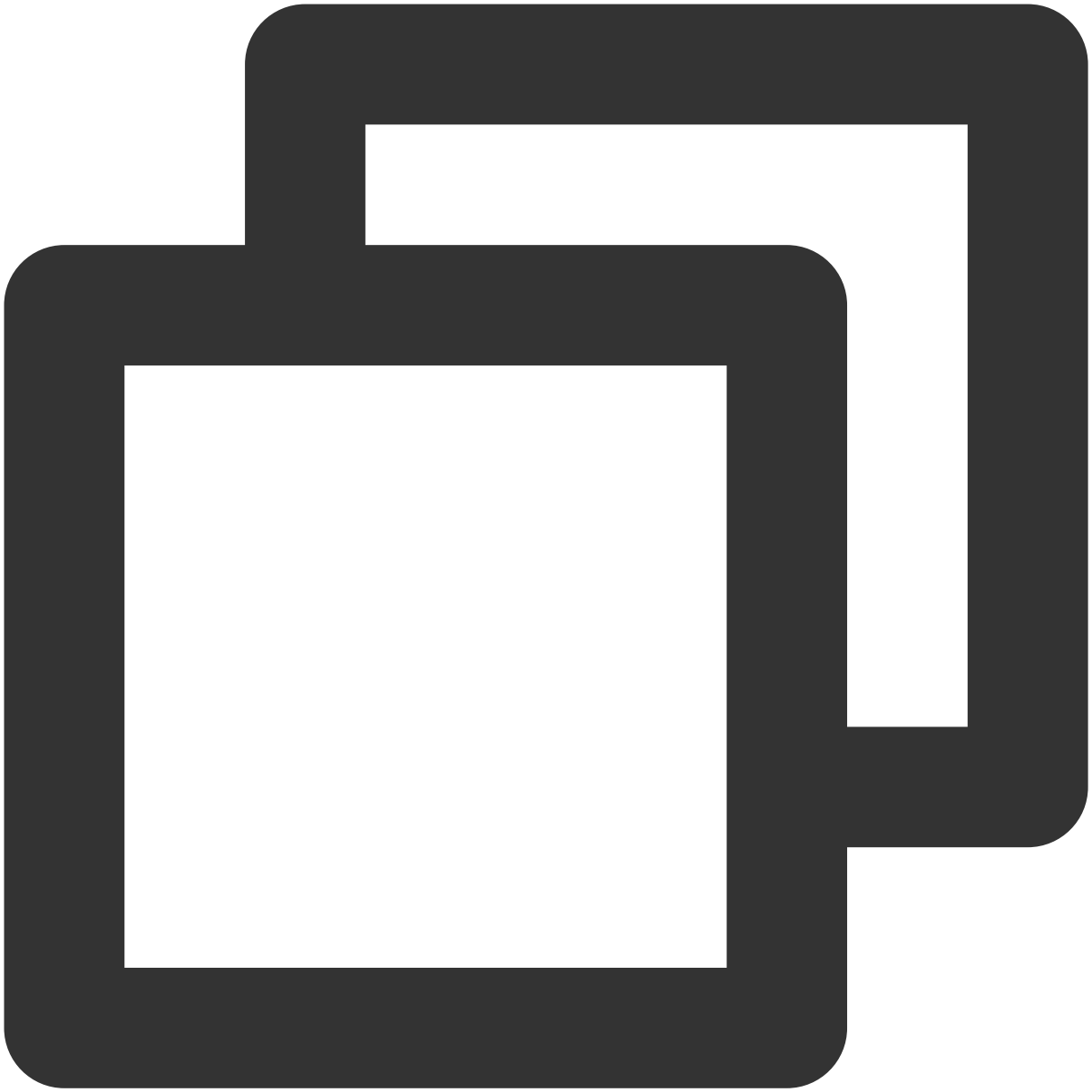
画面共有関連コールバック

API	説明
onScreenCaptureStarted	画面共有開始のコールバック。
onScreenCaptureStopped	画面共有停止のコールバック。

TUIRoomCore基本関数

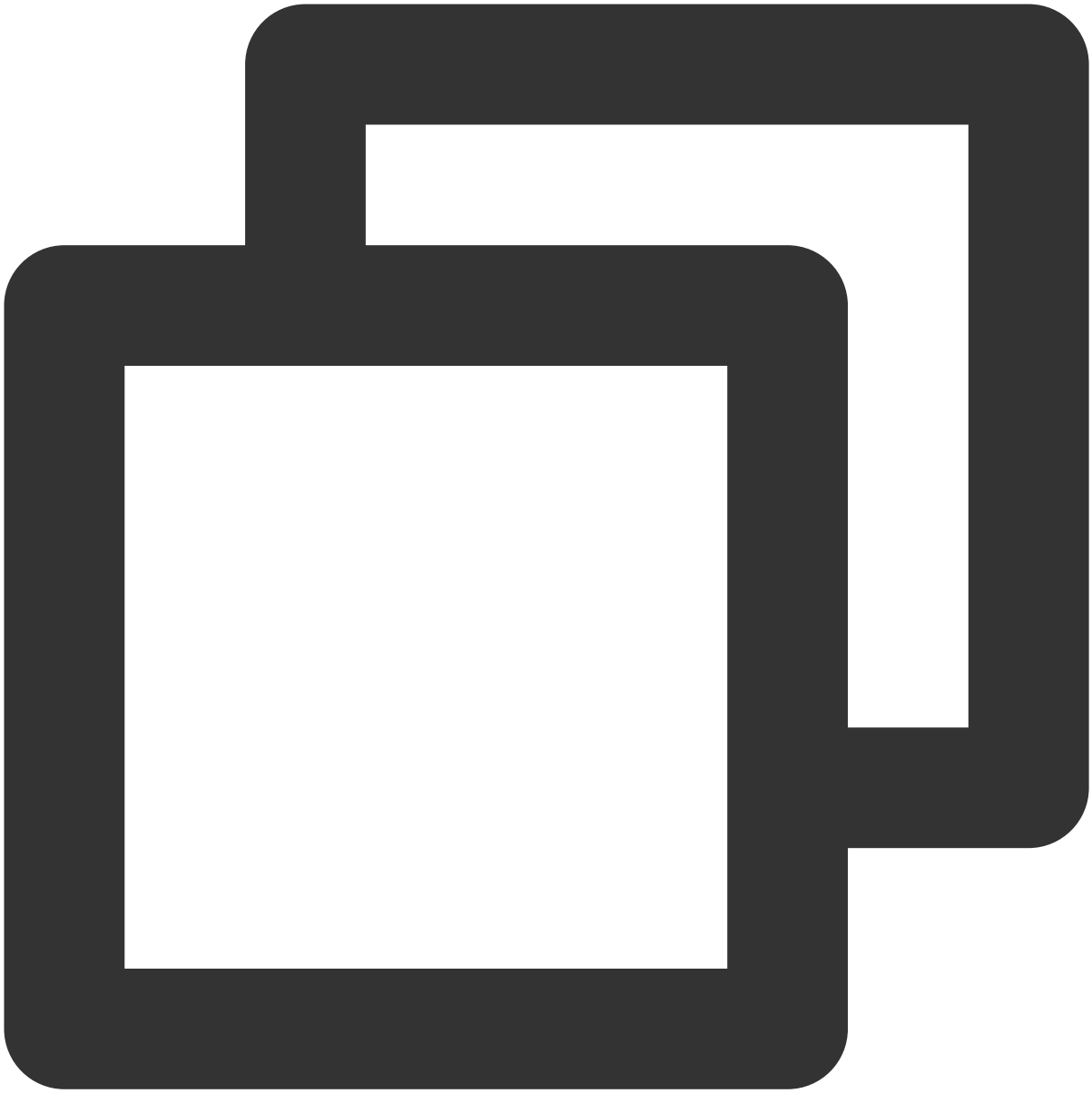
getInstance

TUIRoomCore シングルトンオブジェクトを取得します。



```
+ (instancetype)shareInstance;
```

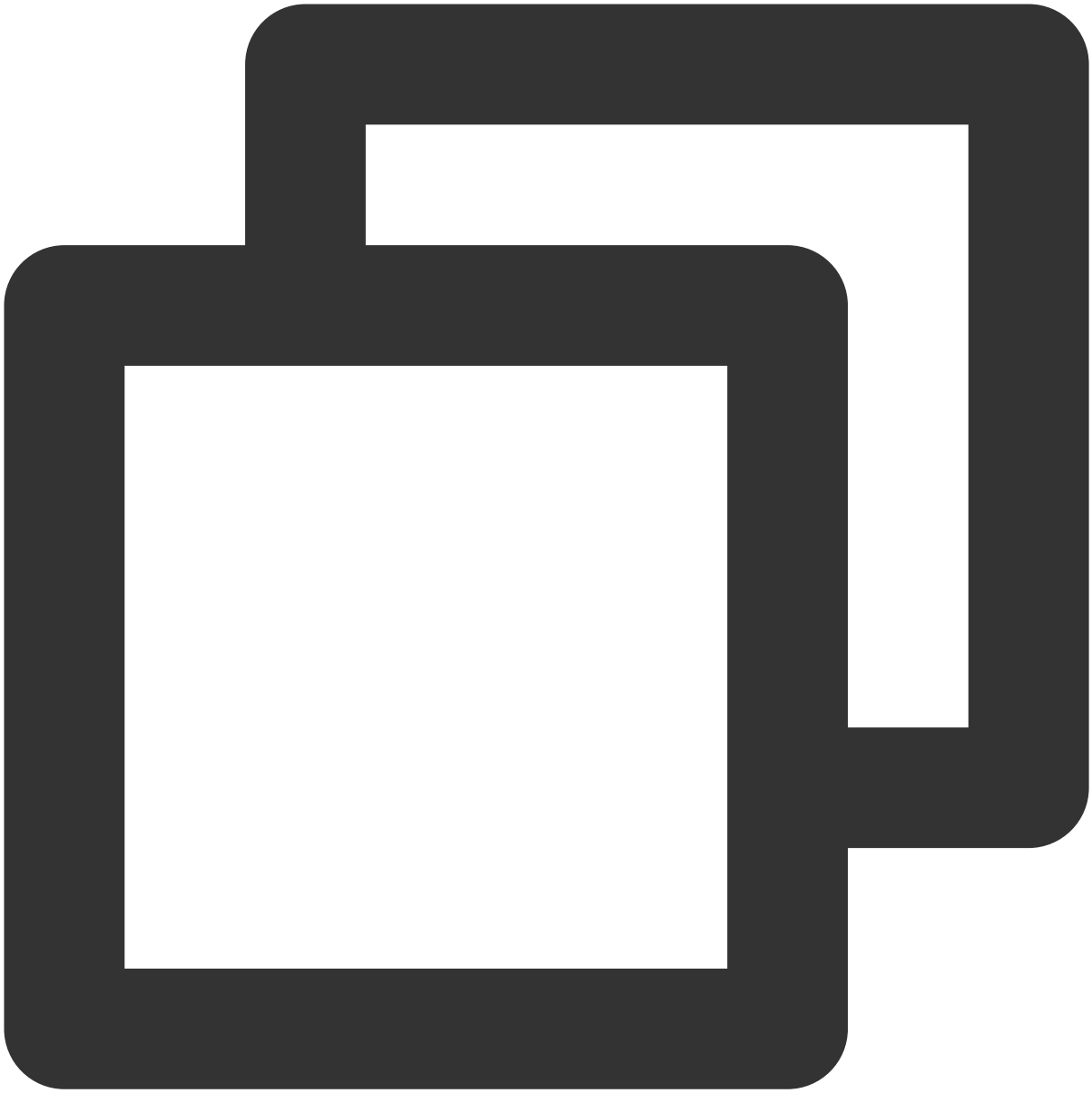
destroyInstance



```
+ (void)destroyInstance;
```

setDelegate

TUIRoomCore イベントコールバック。TUIRoomCoreDelegateを介してTUIRoomCoreの各種ステータス通知を取得できます。



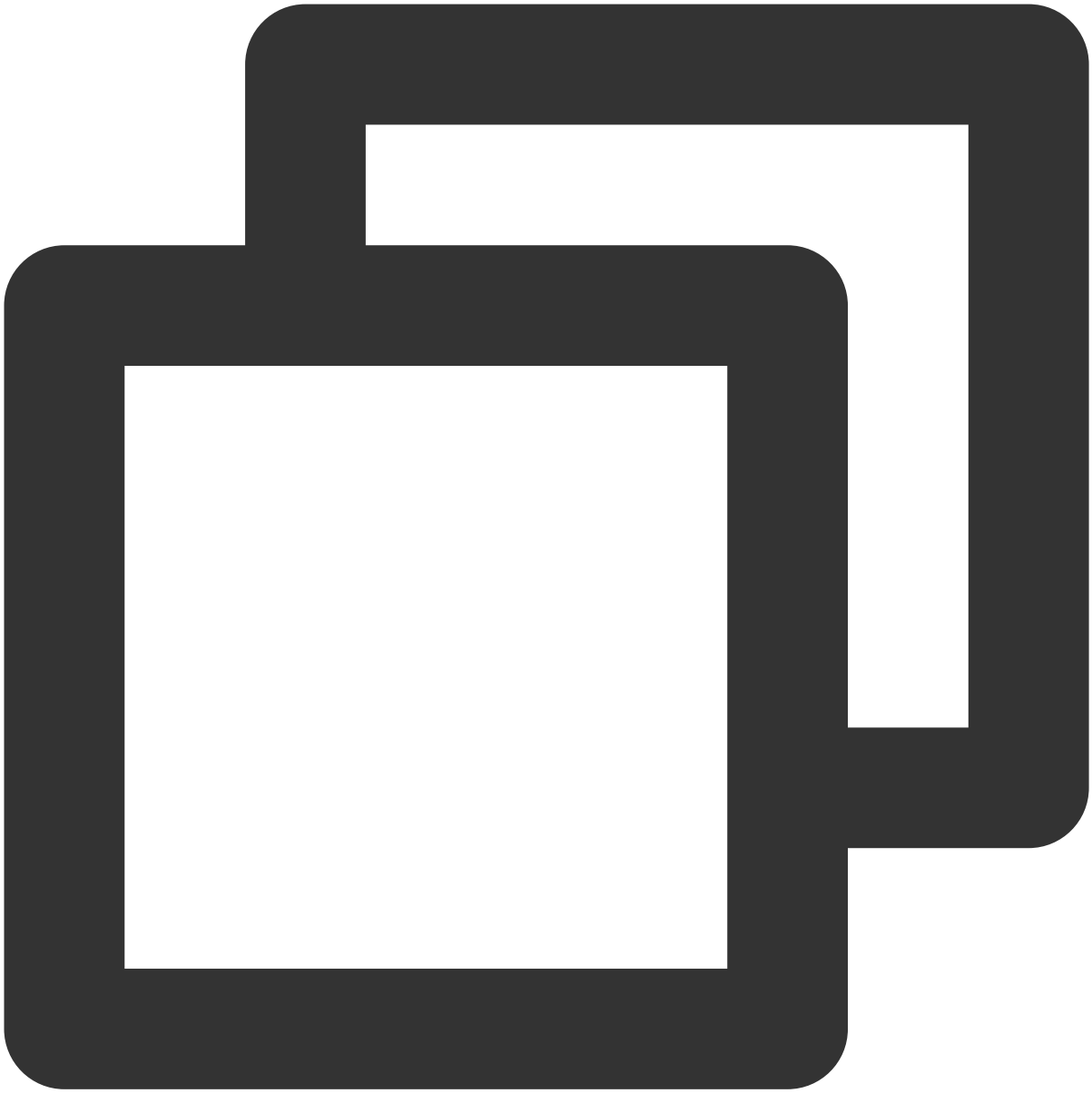
```
- (void)setDelegate:(id<TUIRoomCoreDelegate>)delegate;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
delegate	TUIRoomCoreDelegate	イベントコールバッククラスを受信します。

createRoom

ルームの作成（キャスターが呼び出し）。



```
- (void)createRoom:(NSString *)roomId
        speechMode:(TUIRoomSpeechMode)speechMode
        callback:(TUIRoomActionCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
roomId	NSString	ルームID。ご自身でアサインし、一元管理する必要があります。

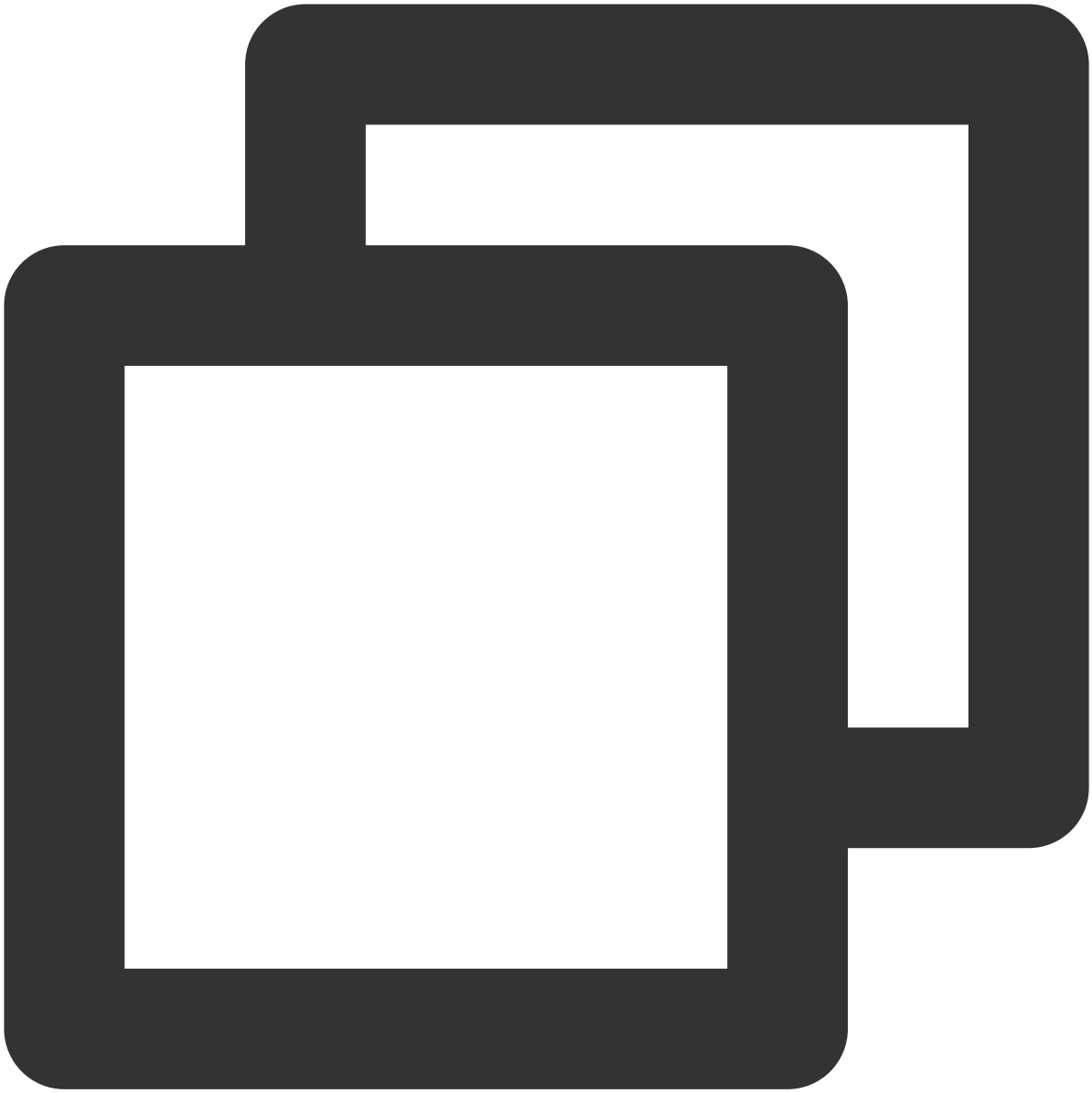
speechMode	TUIRoomSpeechMode	発言モード。
callback	TUIRoomActionCallback	ルームの作成結果のコールバック。

キャスターの通常の呼び出しフローは以下のとおりです。

1. **キャスター**が `createRoom()` を呼び出し、ルームを作成します。ルーム作成の成否はTUIRoomActionCallbackでキャスターに通知されます。
2. **キャスター**が `startCameraPreview()` を呼び出し、カメラキャプチャとプレビューを起動します。
3. **キャスター**が `startLocalAudio()` を呼び出し、ローカルマイクを起動します。

destroyRoom

ルームの破棄（キャスターが呼び出し）。キャスターは、ルームの作成後、この関数を呼び出して、ルームを破棄できます。



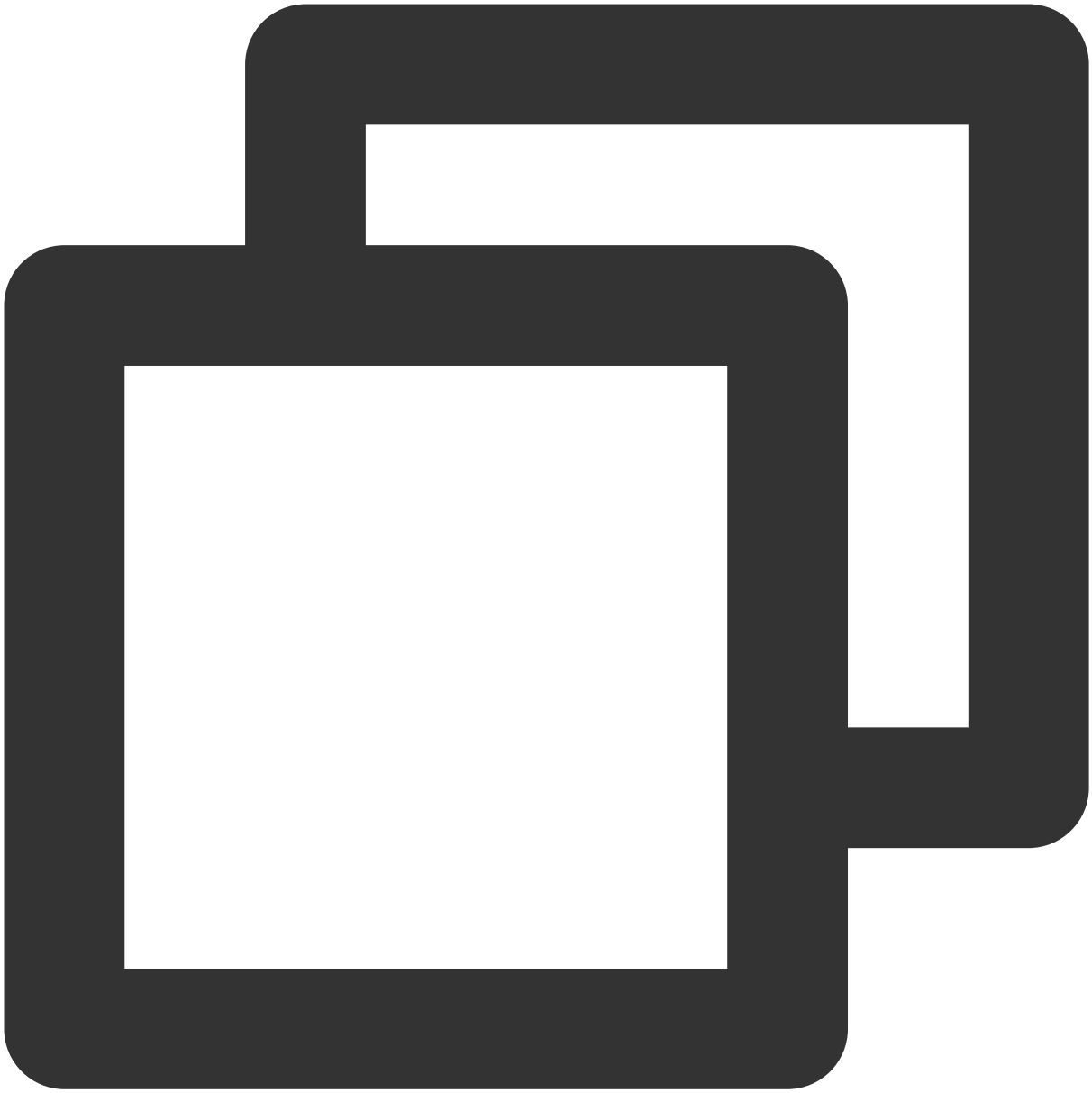
```
- (void)destroyRoom:(TUIRoomActionCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	TUIRoomActionCallback	ルームの破棄結果のコールバック。

enterRoom

入室（参加者が呼び出し）。



```
- (void)enterRoom:(NSString *)roomId
         callback:(TUIRoomActionCallback)callback;
```

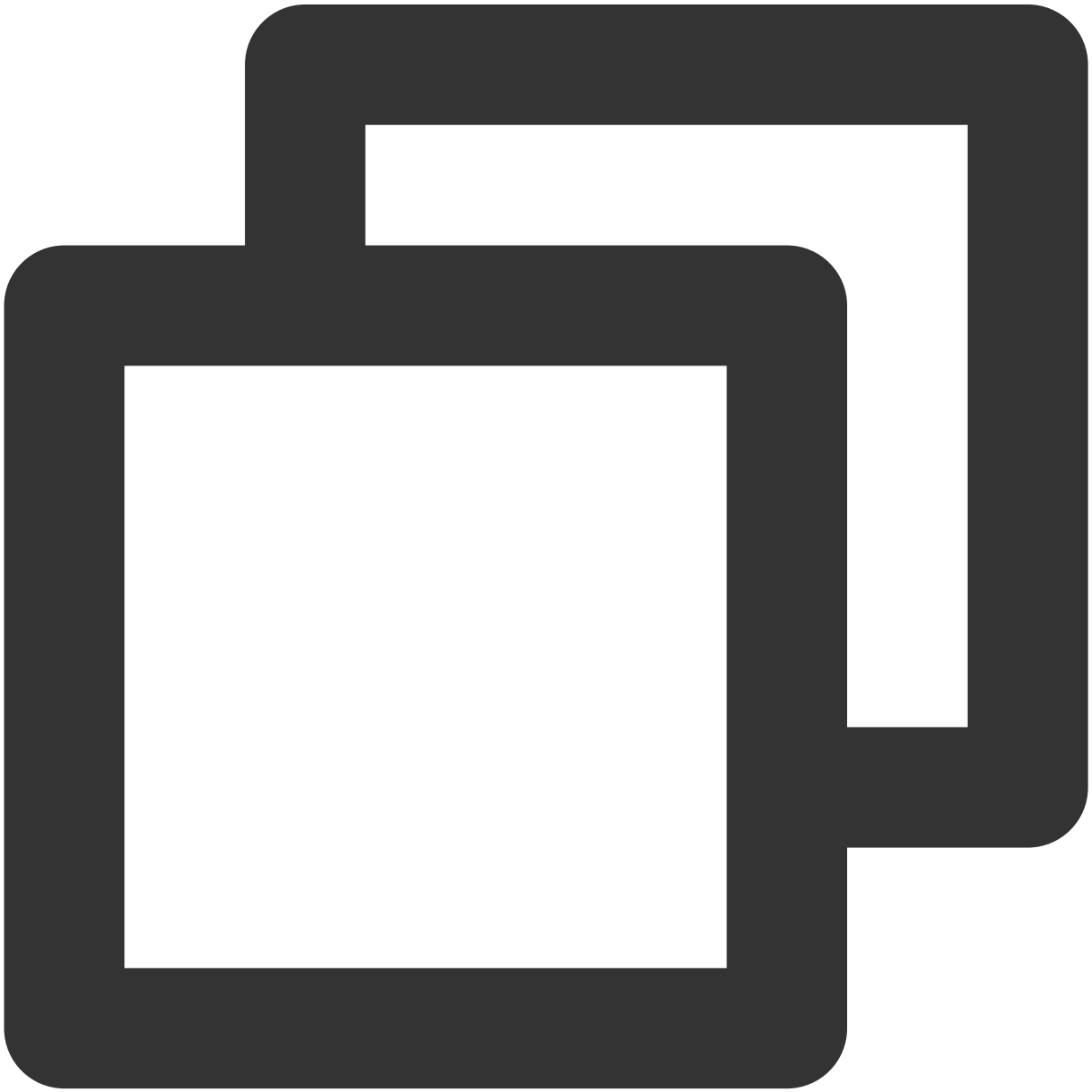
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
roomId	NSString	ルームID。
callback	TUIRoomActionCallback	結果のコールバック。

参加者が入室する場合の通常の呼び出し手順は次のとおりです。
1. **参加者**が `enterRoom` を呼び出し、roomIdを渡せば入室できます。
2. **参加者**が `startCameraPreview()` を呼び出して、カメラプレビューを起動し、 `startLocalAudio()` を呼び出して、マイクキャプチャを起動します。
3. **参加者**が `onRemoteUserCameraAvailable` のイベントを受信し、 `startRemoteView()` を呼び出して、ビデオ再生を開始します。

leaveRoom

退室（参加者が呼び出し）。



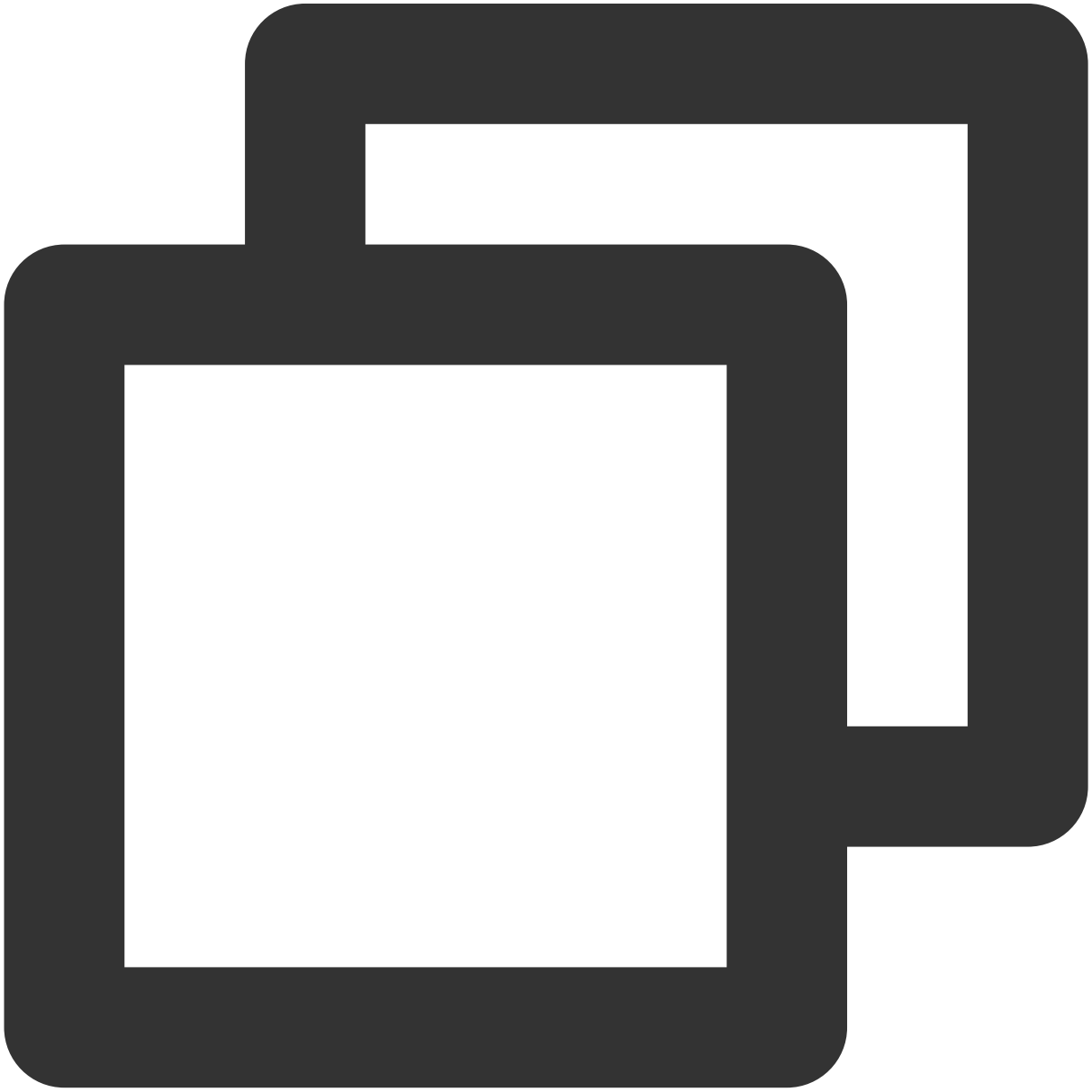
```
- (void)leaveRoom:(TUIRoomActionCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	TUIRoomActionCallback	結果のコールバック。

getRoomInfo

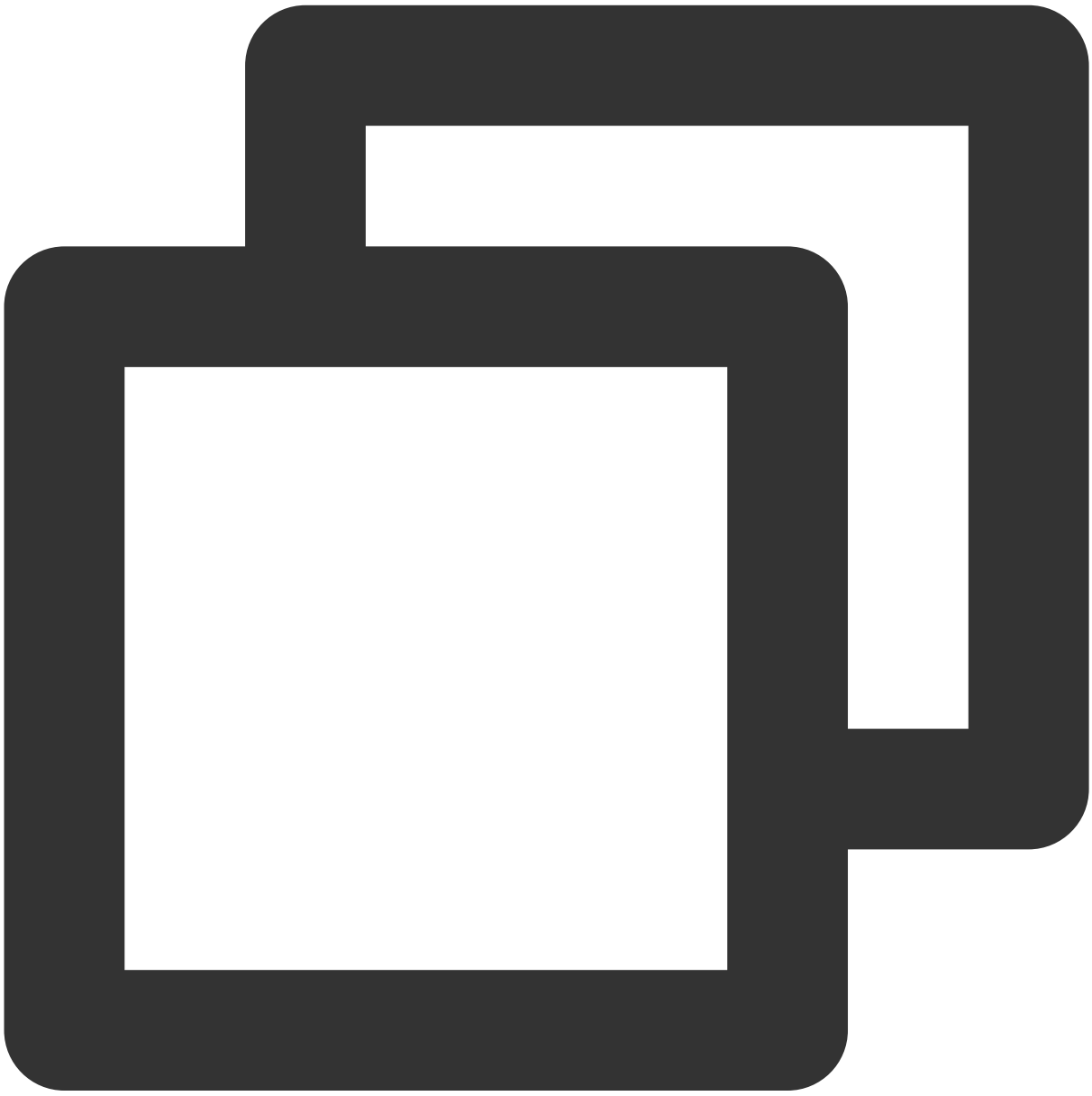
ルーム情報を取得します。



```
- (nullable TUIRoomInfo *)getRoomInfo;
```

getRoomUsers

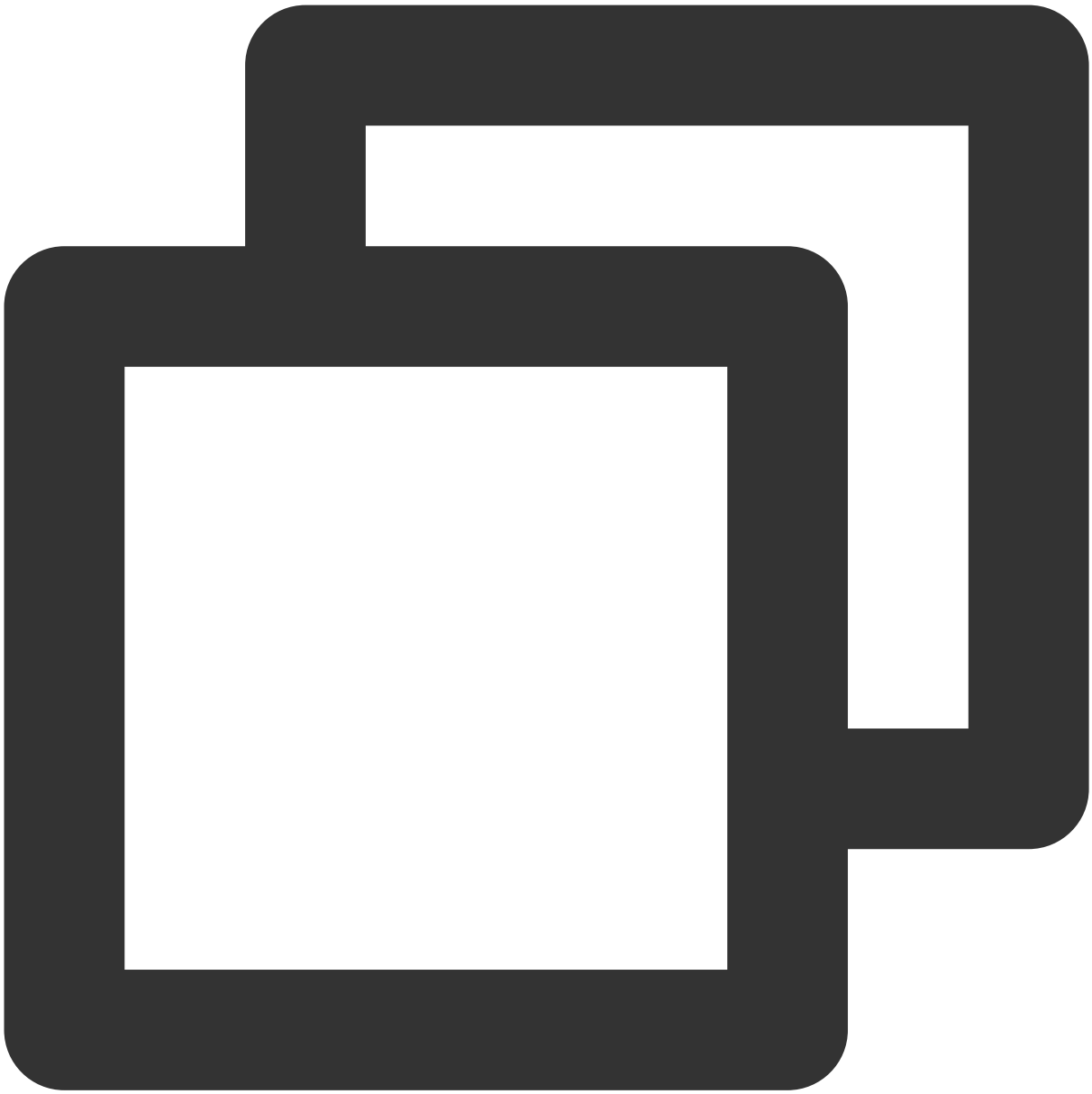
ルームの全メンバー情報を取得します。



```
- (nullable NSArray<TUIRoomUserInfo *> *)getRoomUsers;
```

getUserInfo

メンバー情報を取得します。



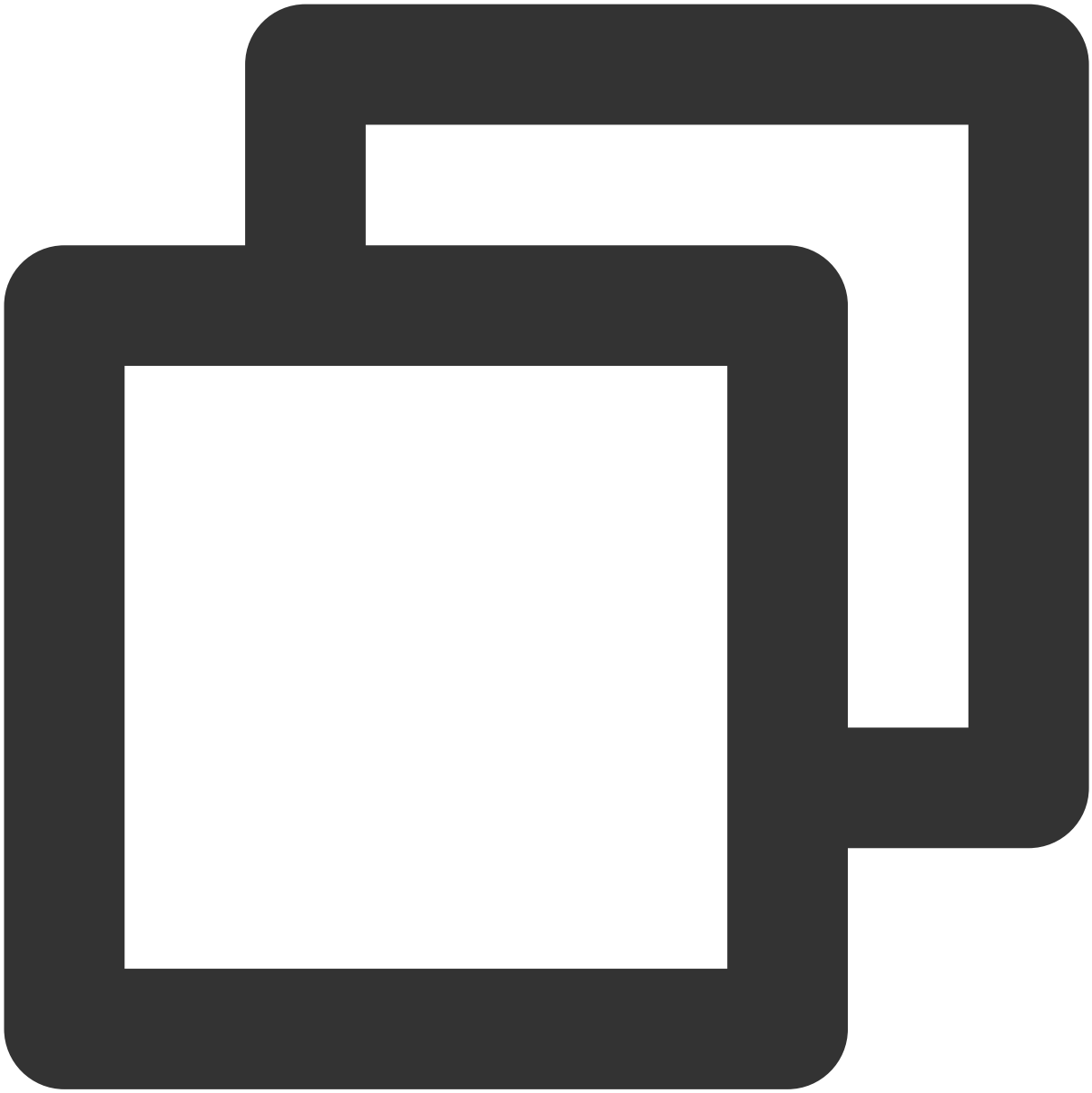
```
- (void)getUserInfo:(NSString *)userId
           callback:(TUIRoomUserInfoCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。
callback	TUIRoomUserInfoCallback	ルームメンバーの詳細情報のコールバック。

setSelfProfile

ユーザー情報を設定します。



```
- (void)setSelfProfile:(NSString *)userName
        avatarURL:(NSString *)avatarURL
        callback:(TUIRoomActionCallback)callback;
```

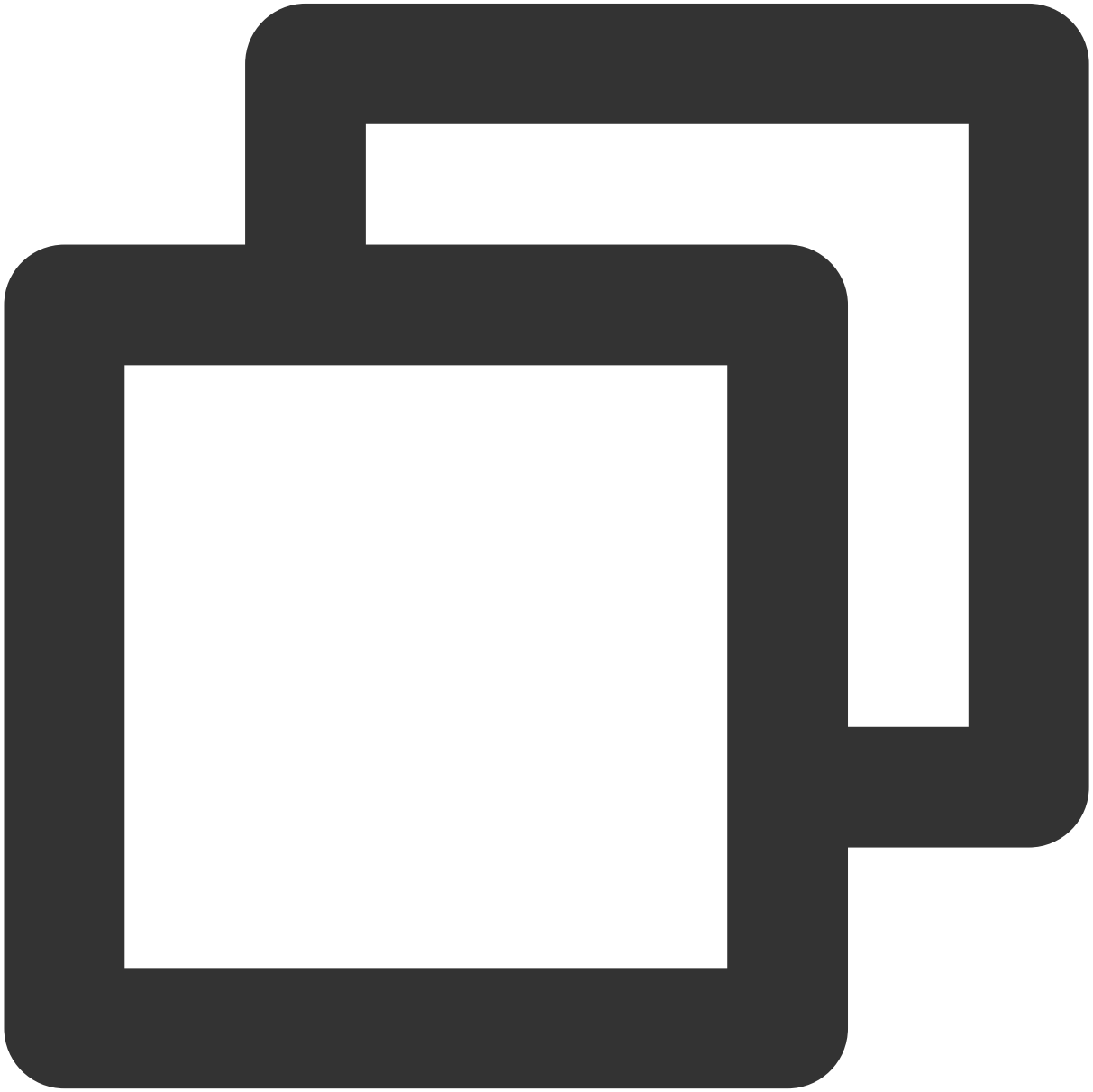
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userName	NSString	ユーザーの氏名。
avatarURL	NSString	ユーザーのプロフィール画像URL。
callback	TUIRoomActionCallback	設定が成功したかどうかの結果のコールバック。

transferRoomMaster

グループを他のユーザーに引き渡します。



```
- (void)transferRoomMaster:(NSString *)userId
```

```
                callback:(TUIRoomActionCallback)callback;
```

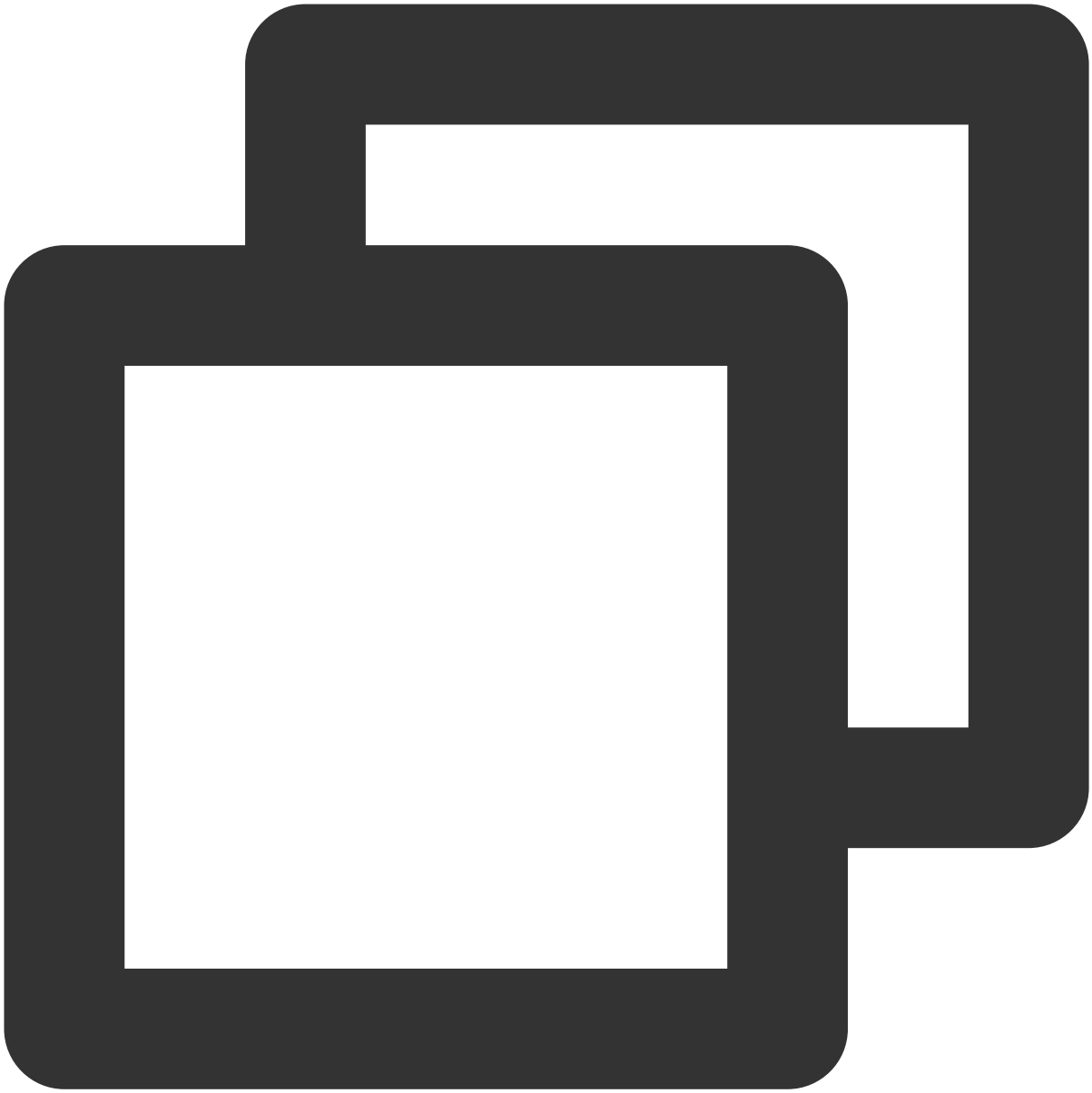
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。
callback	TUIRoomActionCallback	結果のコールバック。

ローカルプッシュインターフェース

startCameraPreview

ローカルカメラプレビューを起動します。



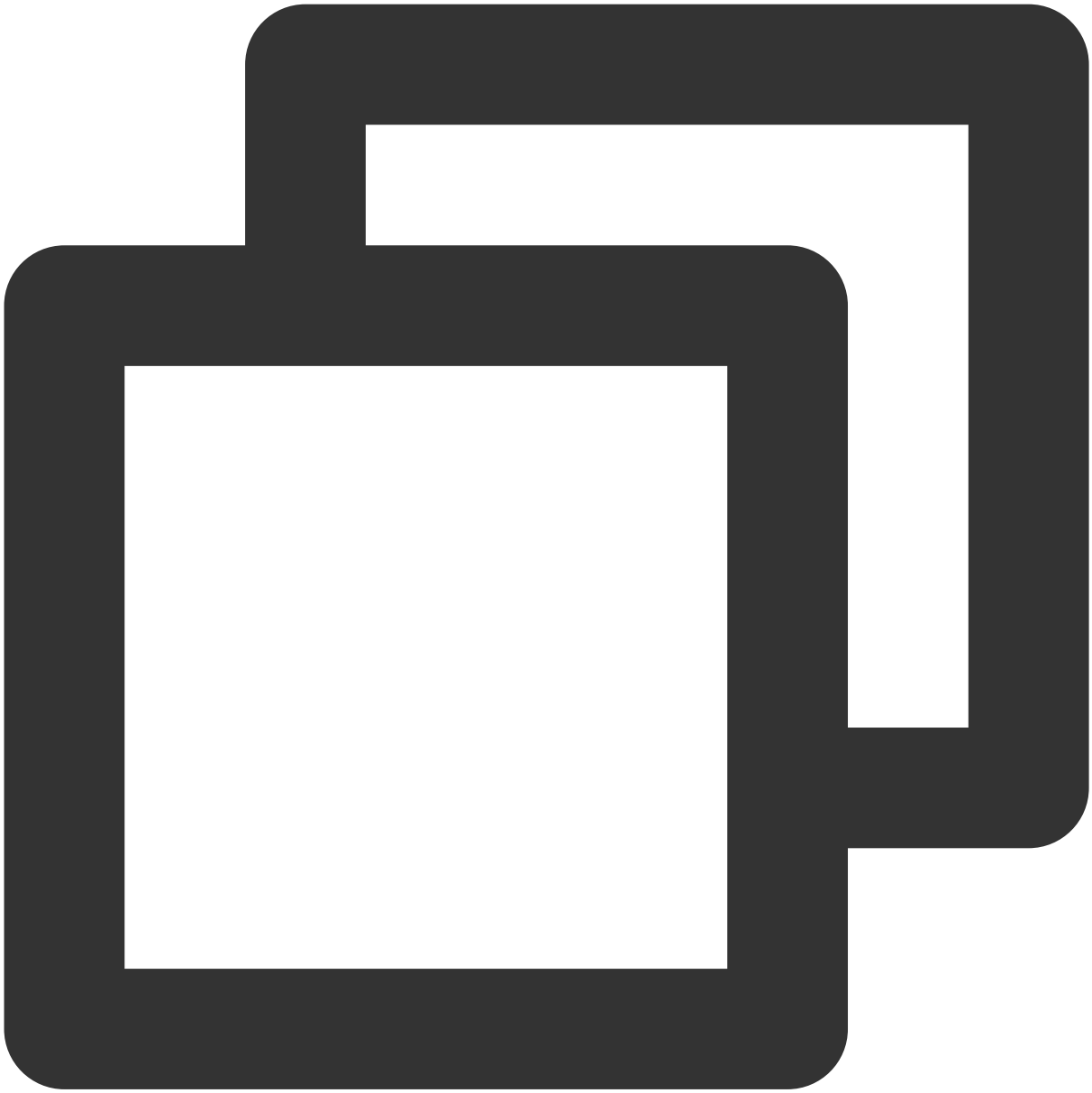
```
- (void)startCameraPreview:(BOOL)isFront
                      view:(UIView *)view;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
isFront	BOOL	YES：フロントカメラ、NO：リアカメラ。
view	UIView	ビデオ画像をロードするウィジェット。

stopCameraPreview

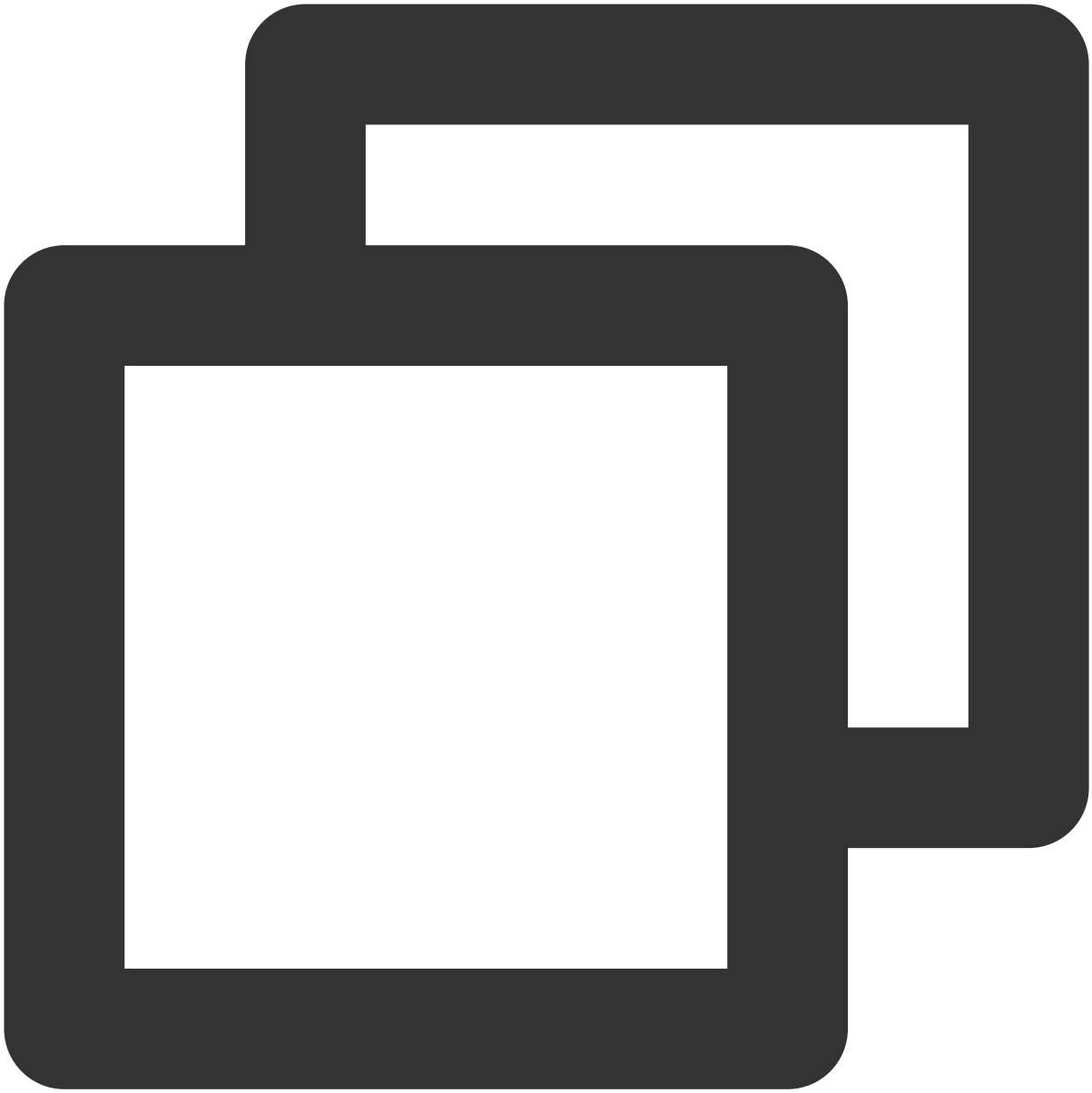
ローカルカメラプレビューを停止します。



```
- (void)stopCameraPreview;
```

startLocalAudio

マイクの集音開始。



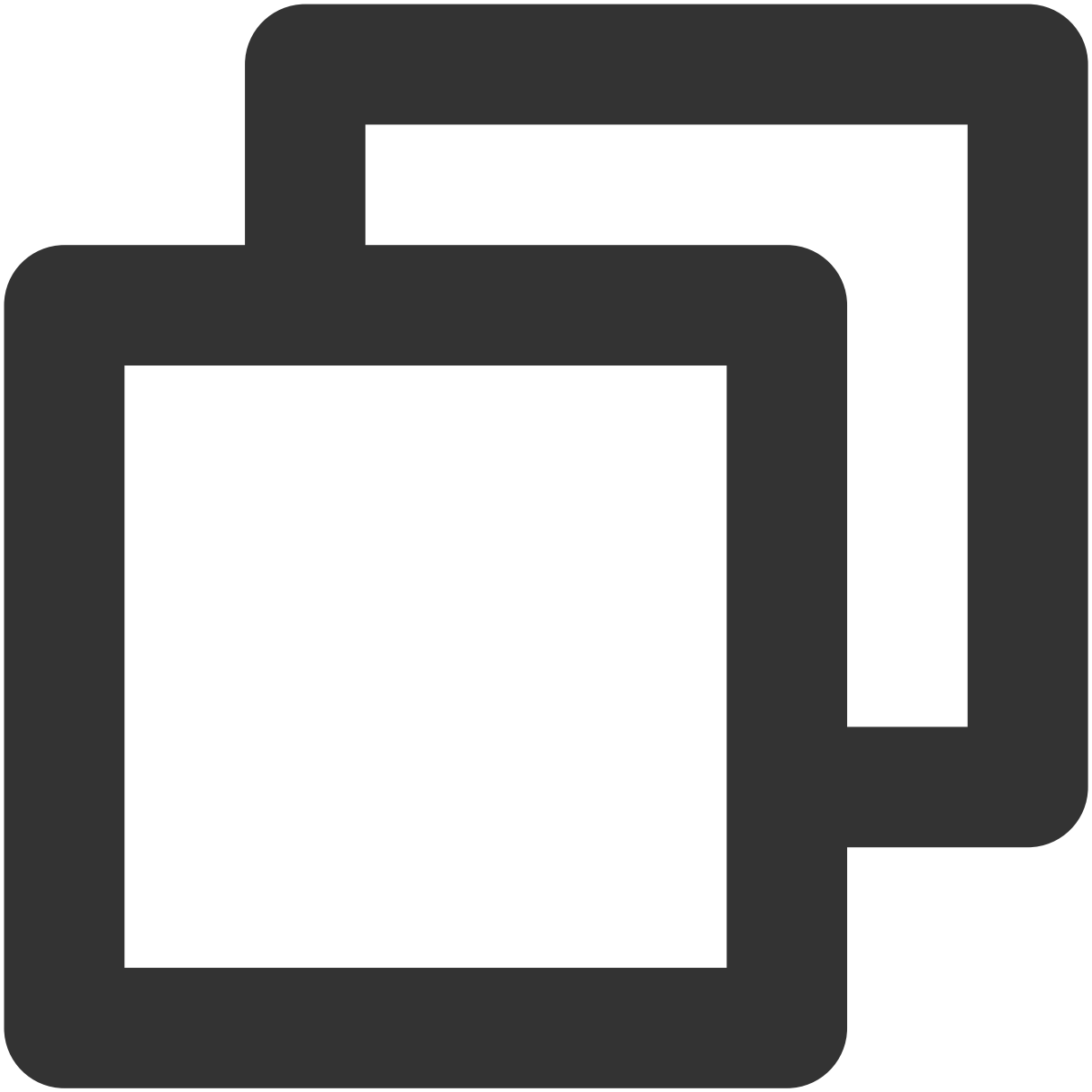
```
- (void)startLocalAudio:(TRTCAudioQuality)quality;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
quality	TRTCAudioQuality	キャプチャの音質。

stopLocalAudio

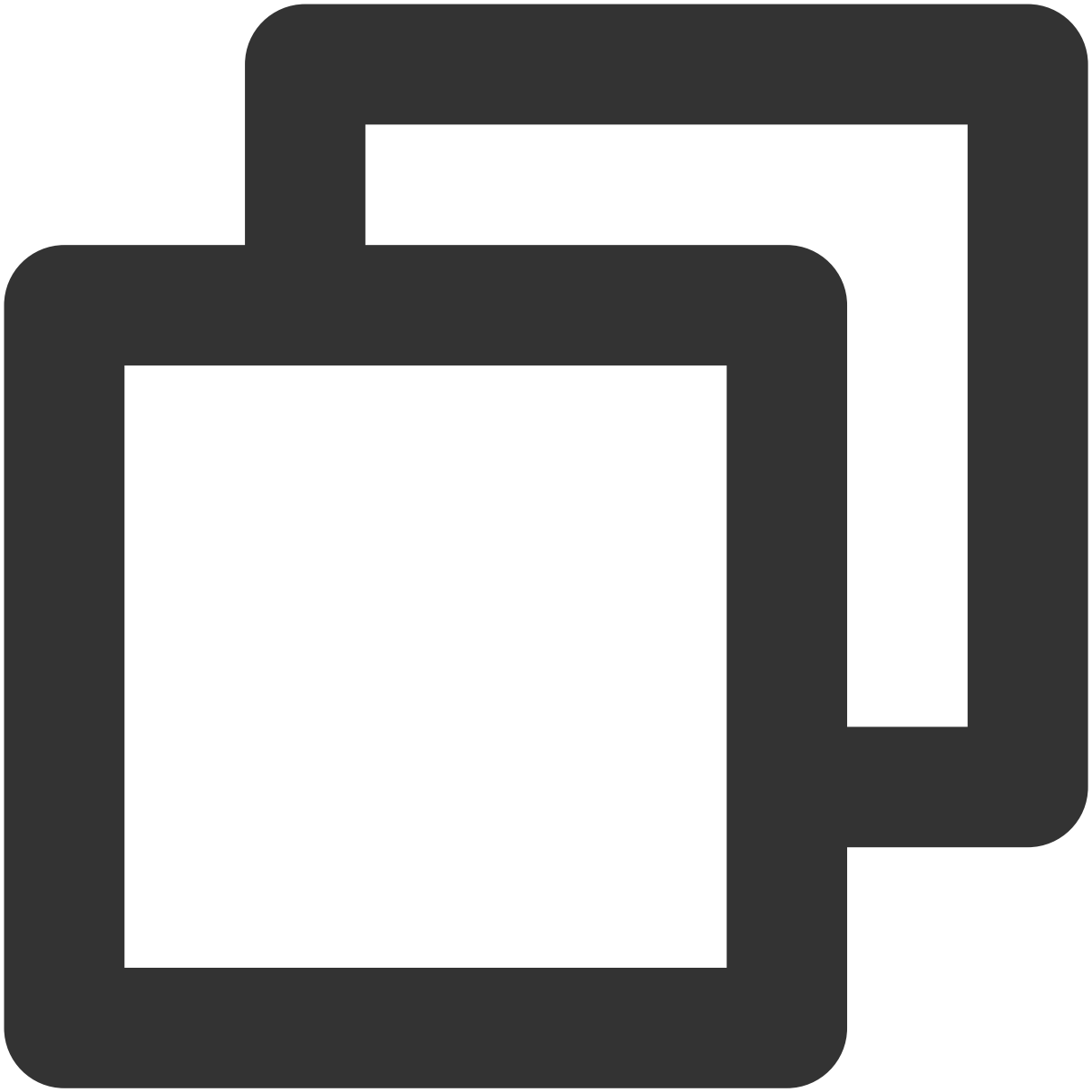
マイクの集音停止



```
- (void)stopLocalAudio;
```

setVideoMirror

ローカル画面のイメージプレビューモードを設定します。



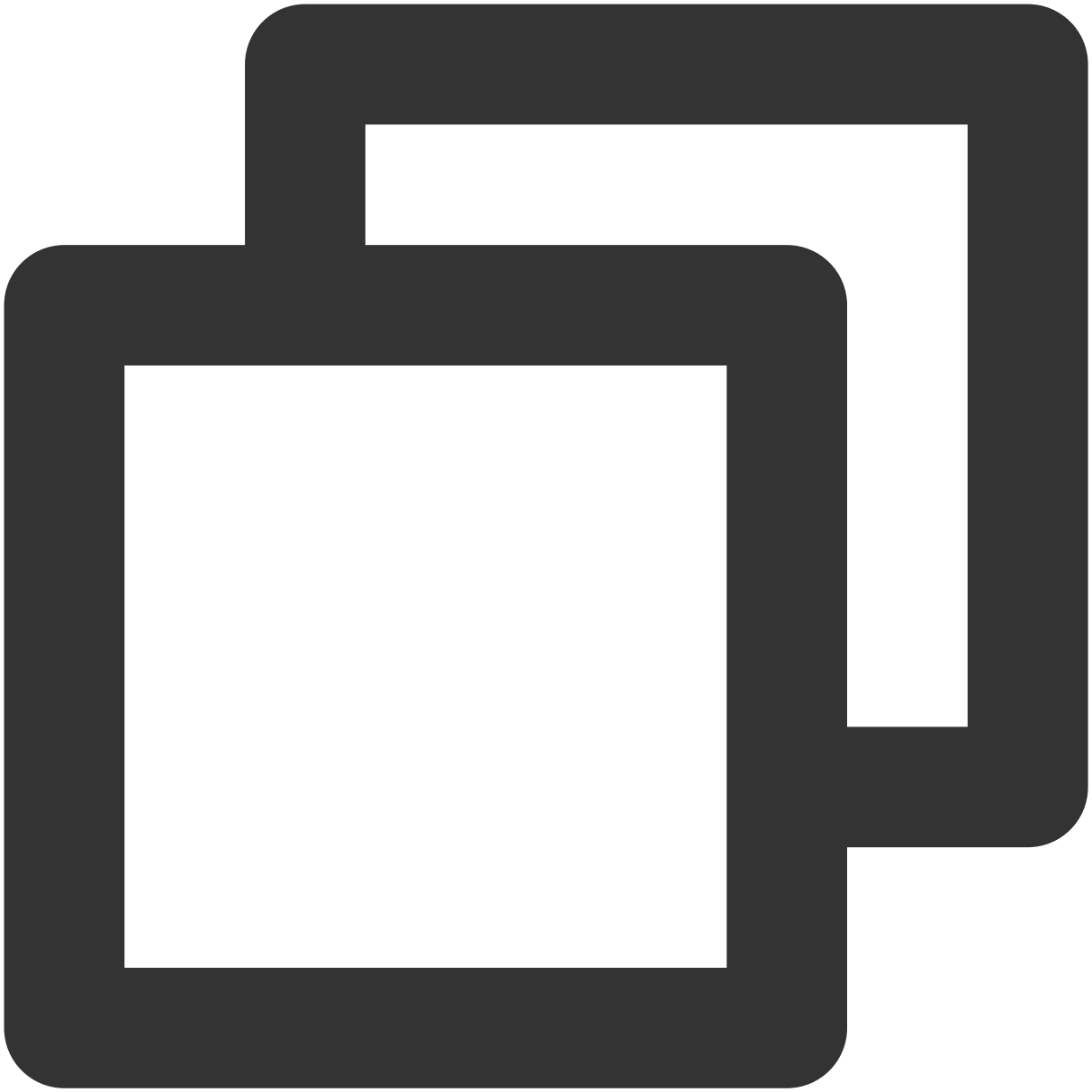
```
- (void)setVideoMirror:(TRTCVideoMirrorType)type;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
type	TRTCVideoMirrorType	イメージタイプ。

setSpeaker

スピーカーの起動設定。



```
- (void)setSpeaker:(BOOL)isUseSpeaker;
```

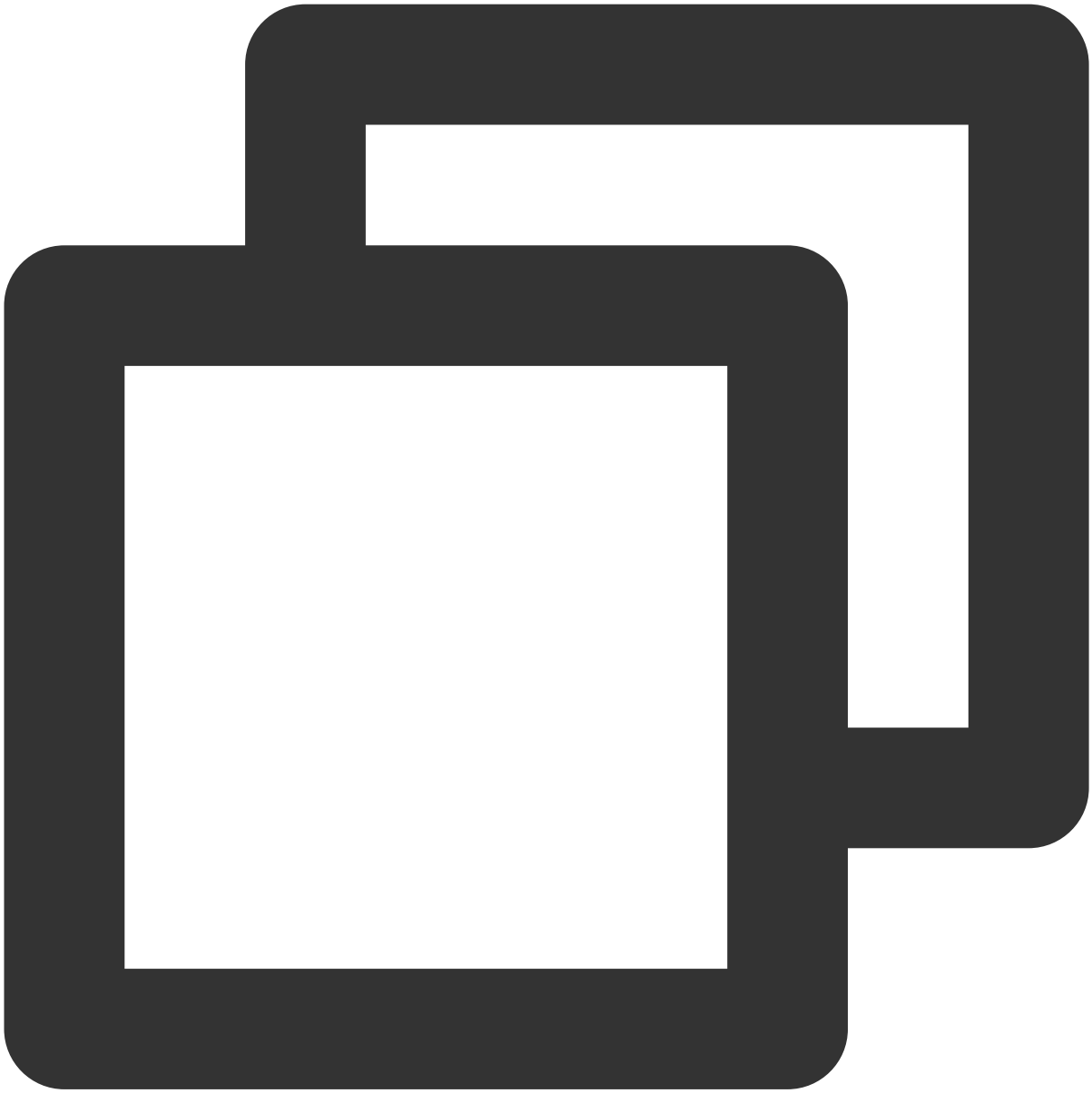
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
isUseSpeaker	BOOL	YES：スピーカー、NO：ヘッドホン。

リモートユーザーに関するインターフェース

startRemoteView

リモートユーザーのビデオストリームのサブスクリプション。



```
- (void)startRemoteView:(NSString *)userId
                   view:(UIView *)view
             streamType:(TUIRoomStreamType)streamType
               callback:(TUIRoomActionCallback)callback;
```

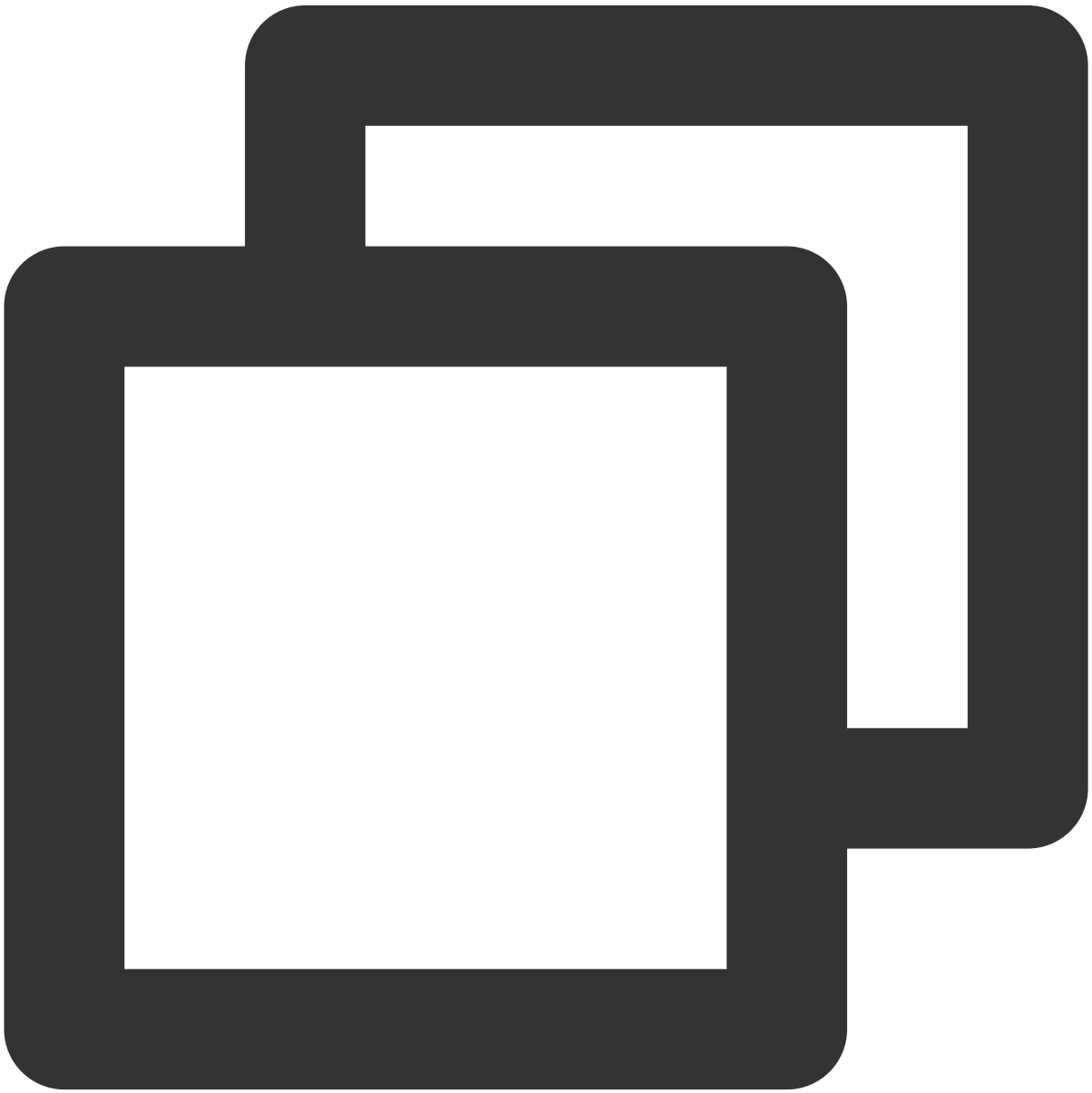
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	NSString	再生が必要なユーザー ID。
view	UIView	ビデオ画像をロードするviewウィジェット。
streamType	TUIRoomStreamType	ストリームのタイプ。
callback	TUIRoomActionCallback	結果のコールバック。

stopRemoteView

サブスクリプションをキャンセルし、リモートビデオ画面の再生を停止します。



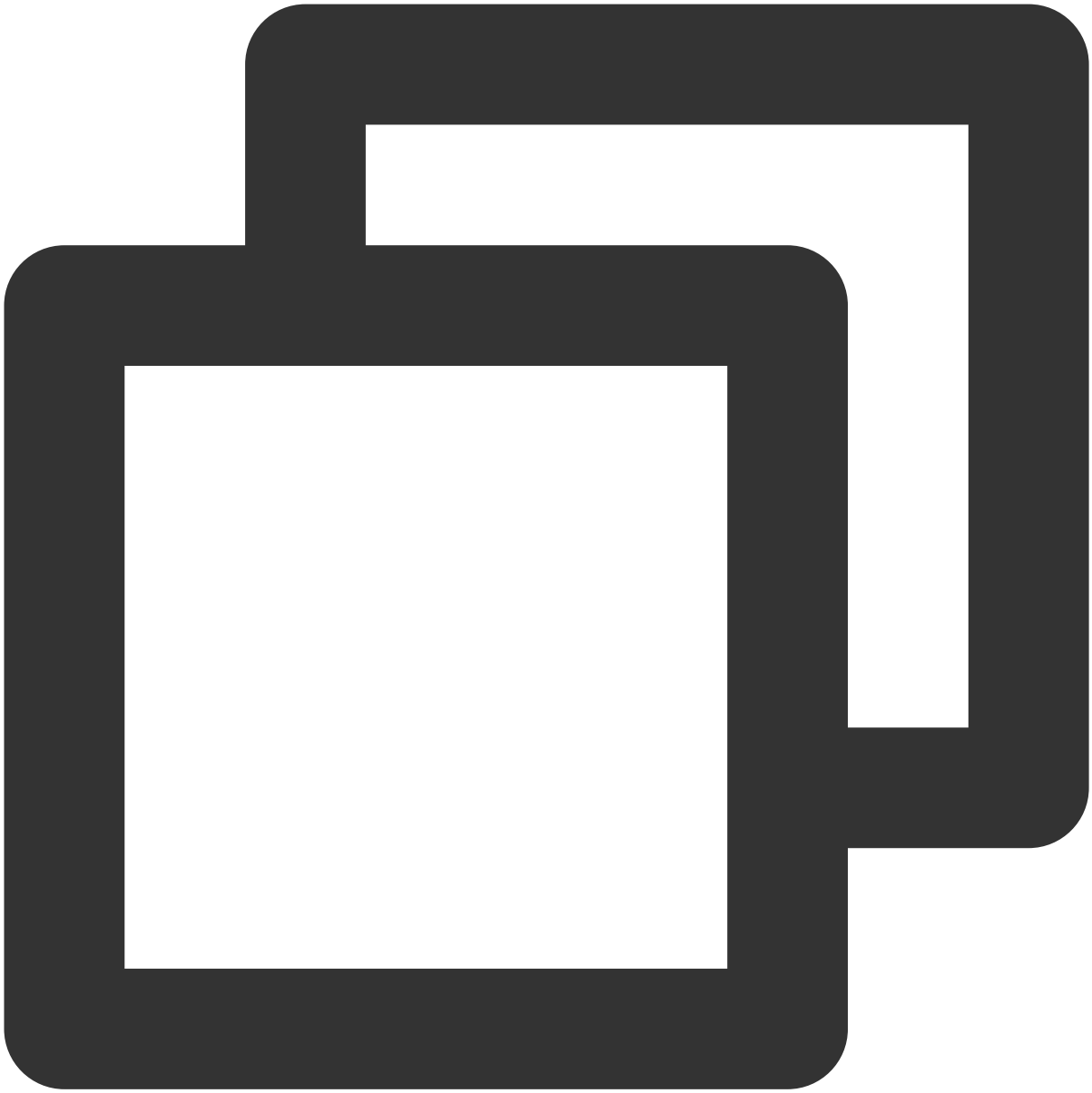
```
- (void)stopRemoteView:(NSString *)userId
            streamType:(TUIRoomStreamType)streamType
              callback:(TUIRoomActionCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	再生停止が必要なユーザーのID。
streamType	TUIRoomStreamType	ストリームのタイプ。
callback	TUIRoomActionCallback	結果のコールバック。

switchCamera

フロント/リアカメラを切り替えます。



```
- (void)switchCamera:(BOOL)isFront;
```

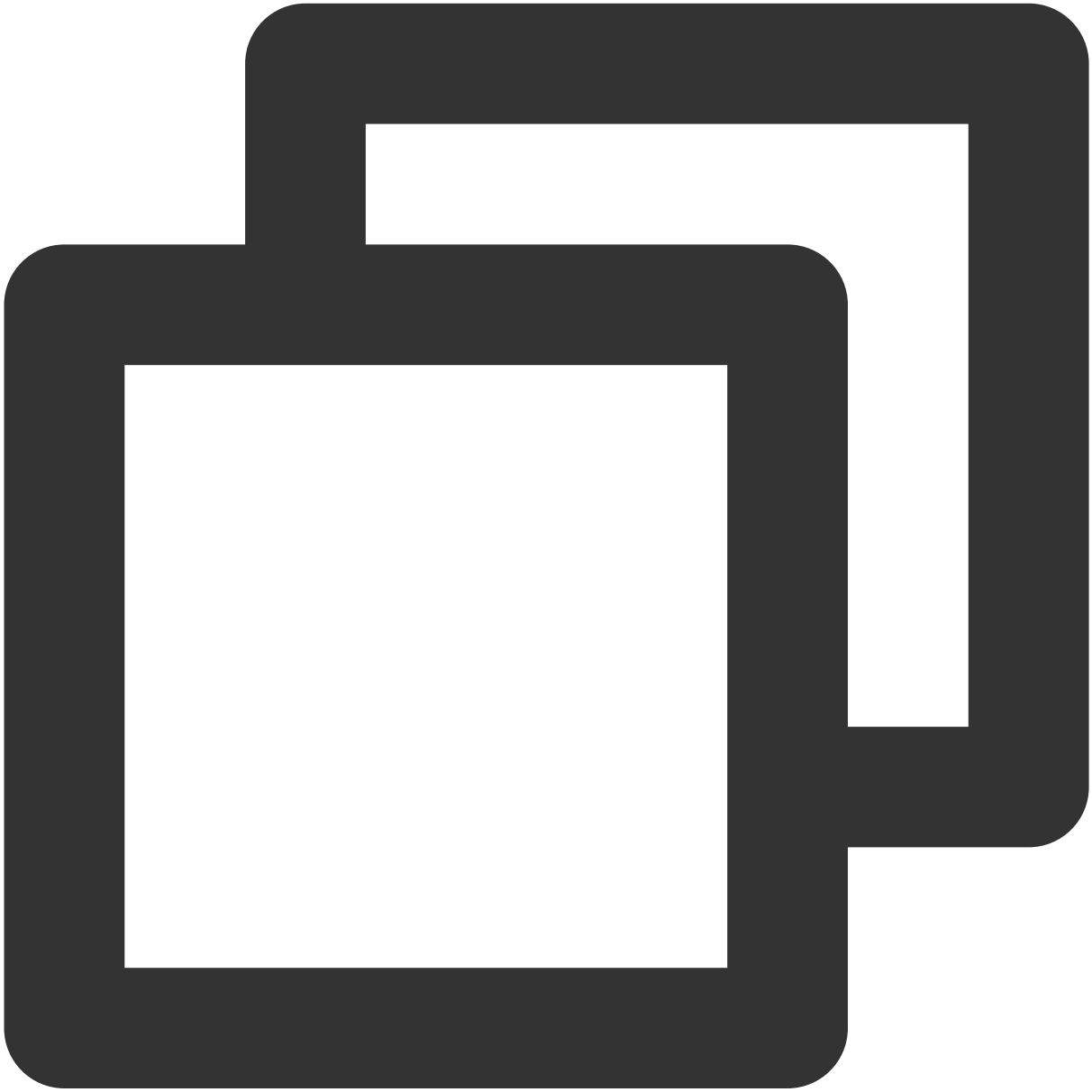
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
isFront	BOOL	YES：フロントカメラ、NO：リアカメラ。

メッセージ送信インターフェース

sendChatMessage

ルーム内でテキストメッセージをブロードキャストします。通常、テキストによるチャットに使用します。



```
- (void)sendChatMessage:(NSString *)message
               callback:(TUIRoomActionCallback)callback;
```

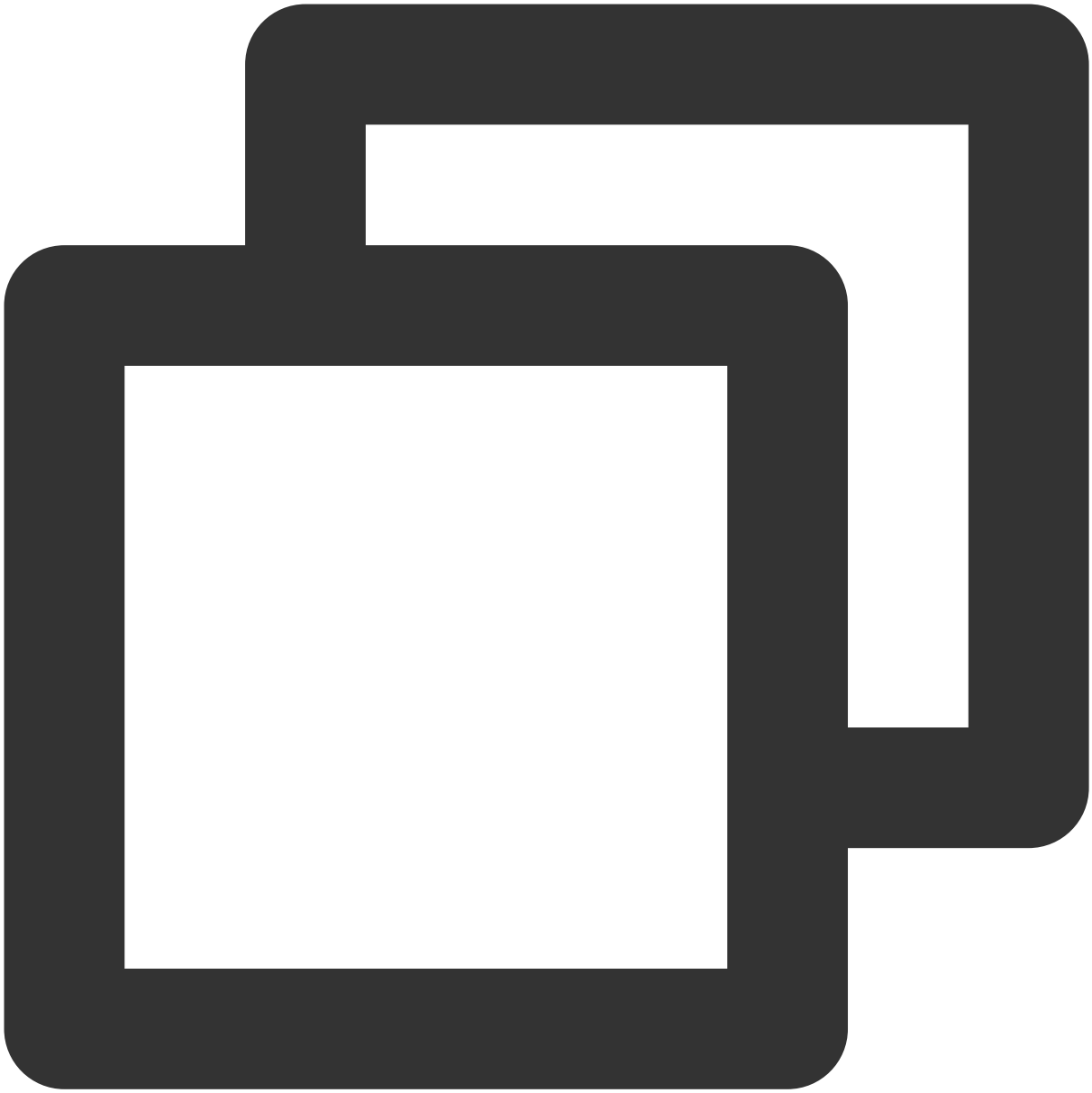
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
message	NSString	メッセージの内容。
callback	TUIRoomActionCallback	送信結果のコールバック。

フィールドコントロール関連インターフェース

muteUserMicrophone

特定ユーザーのマイクを無効化/再有効化します。



```
- (void)muteUserMicrophone:(NSString *)userId
                      mute:(BOOL)mute
                  callback:(TUIRoomActionCallback)callback;
```

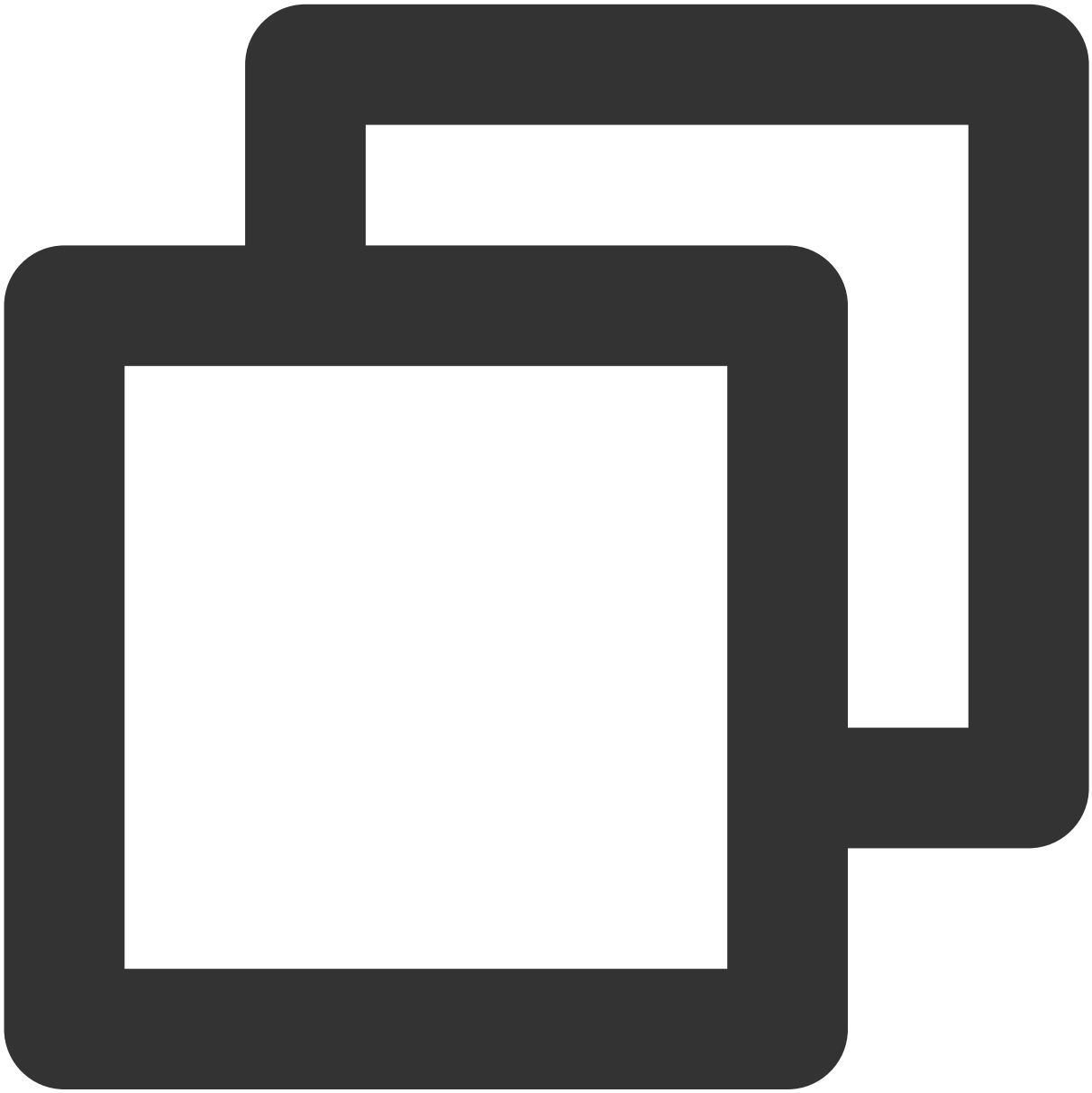
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。
mute	BOOL	無効にするかどうか。

callback	TUIRoomActionCallback	結果のコールバック。

muteAllUsersMicrophone

全ユーザーのマイクを無効化/再有効化します。



```
- (void)muteAllUsersMicrophone:(BOOL)mute
                      callback:(TUIRoomActionCallback)callback;
```

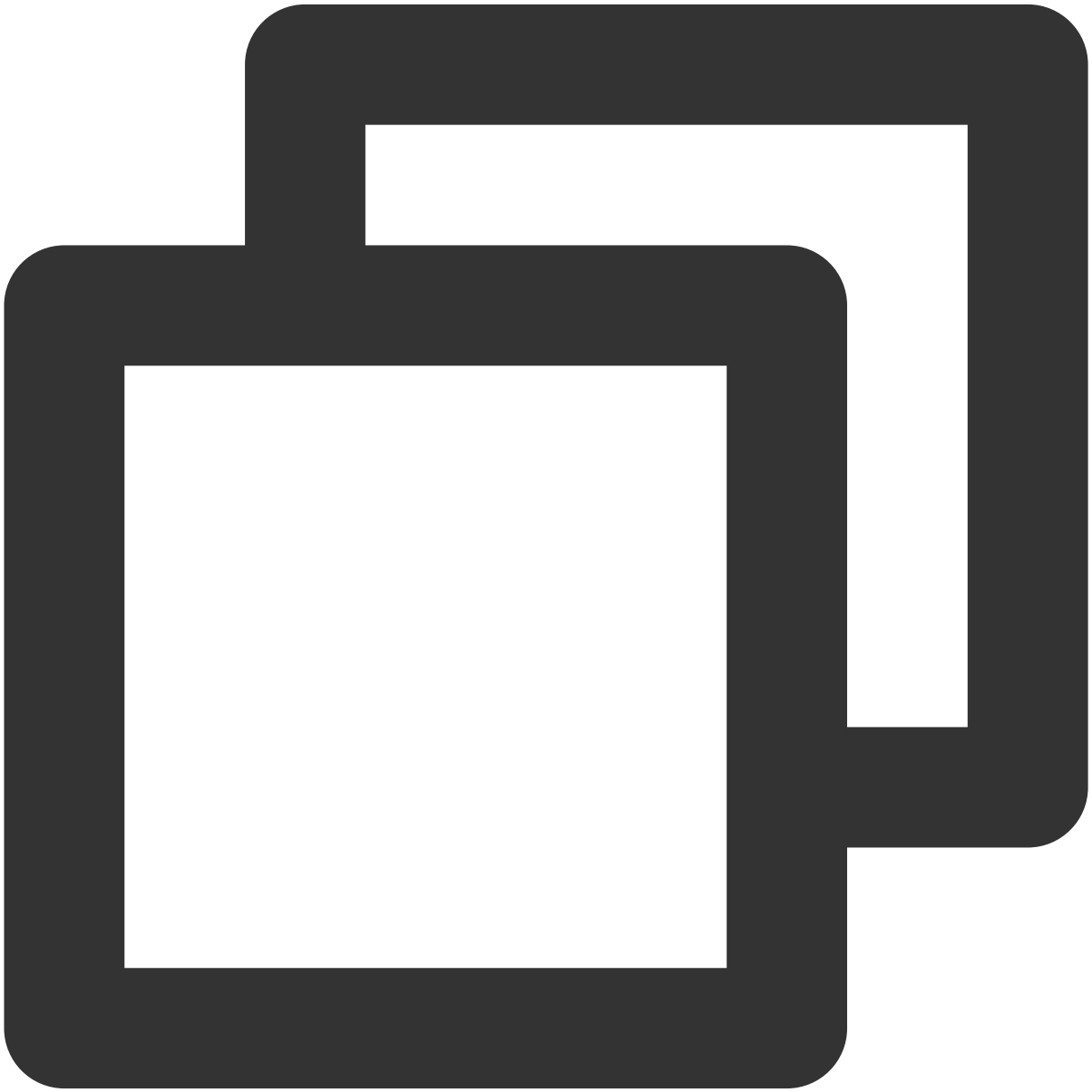
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

mute	BOOL	無効にするかどうか。
callback	TUIRoomActionCallback	結果のコールバック。

muteUserCamera

特定ユーザーのカメラを無効化/再有効化します。



```
- (void)muteUserCamera:(NSString *)userId
                  mute:(BOOL)mute
```

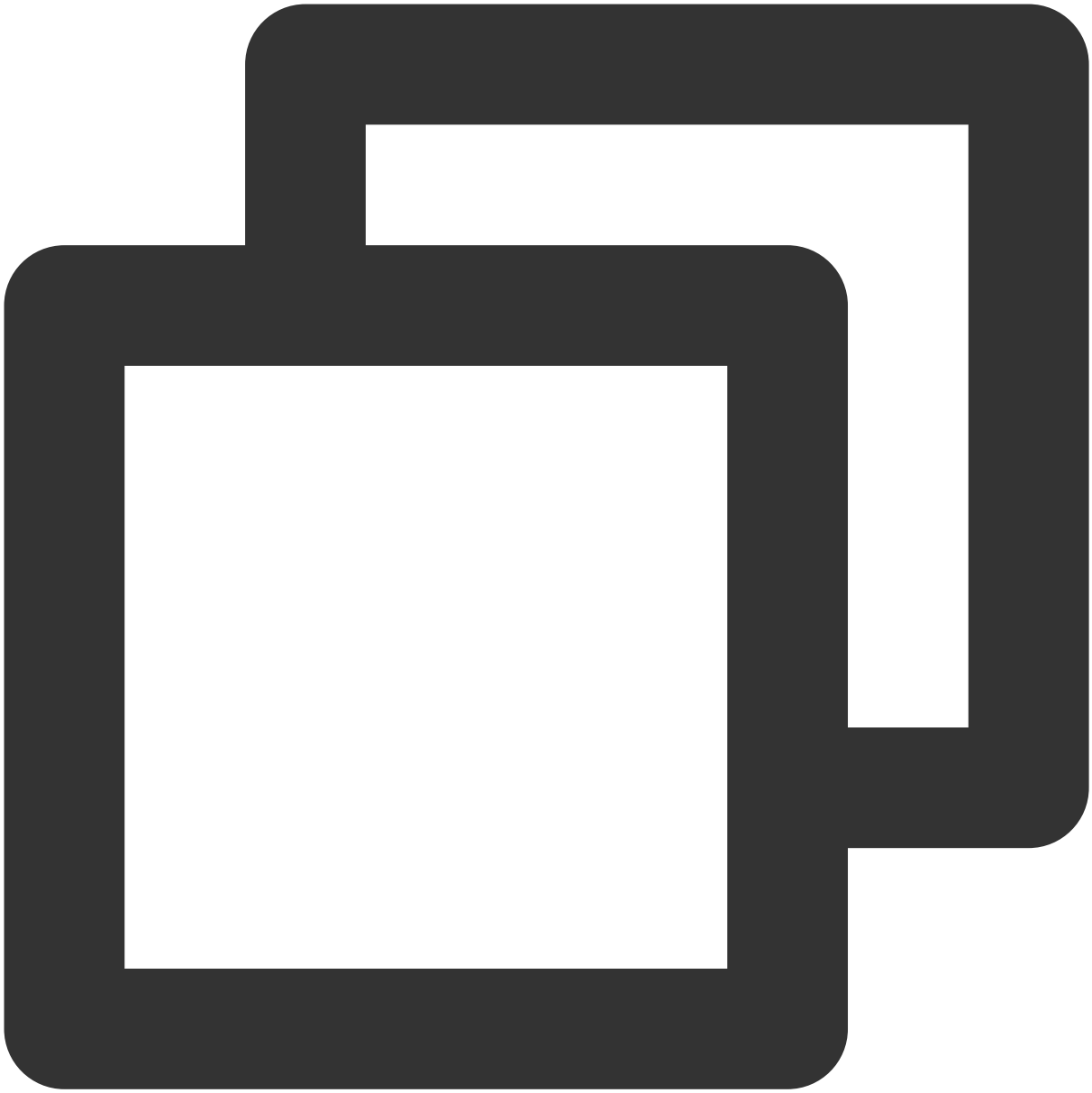
```
                callback:(TUIRoomActionCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。
mute	BOOL	無効にするかどうか。
callback	TUIRoomActionCallback	結果のコールバック。

muteAllUsersCamera

全ユーザーのカメラを無効化/再有効化します。



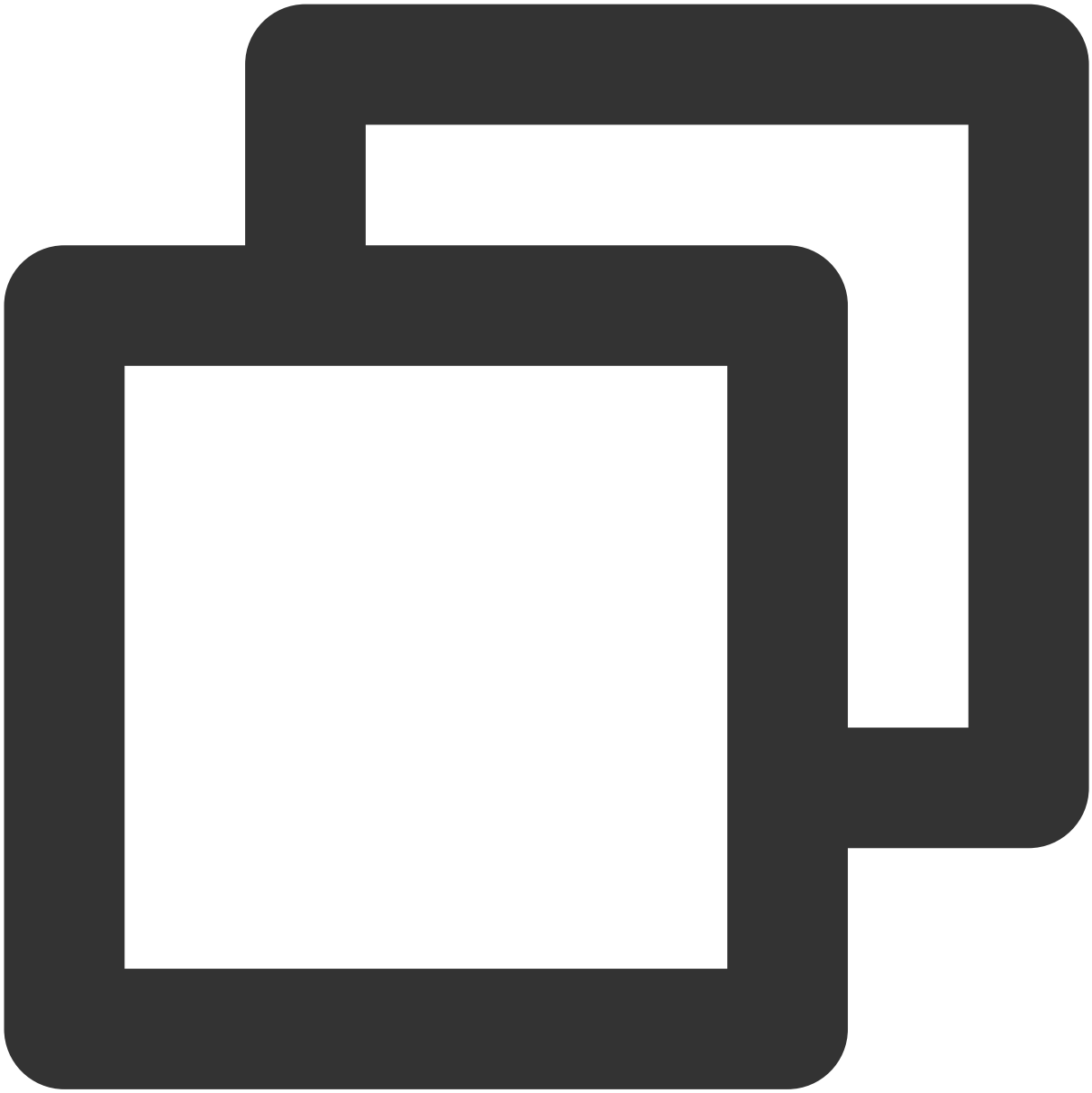
```
- (void)muteAllUsersCamera:(BOOL)mute
                  callback:(TUIRoomActionCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
mute	BOOL	無効にするかどうか。
callback	TUIRoomActionCallback	結果のコールバック。

muteChatRoom

テキストチャットのミュート/再有効化。



```
- (void)muteChatRoom:(BOOL)mute
            callback:(TUIRoomActionCallback)callback;
```

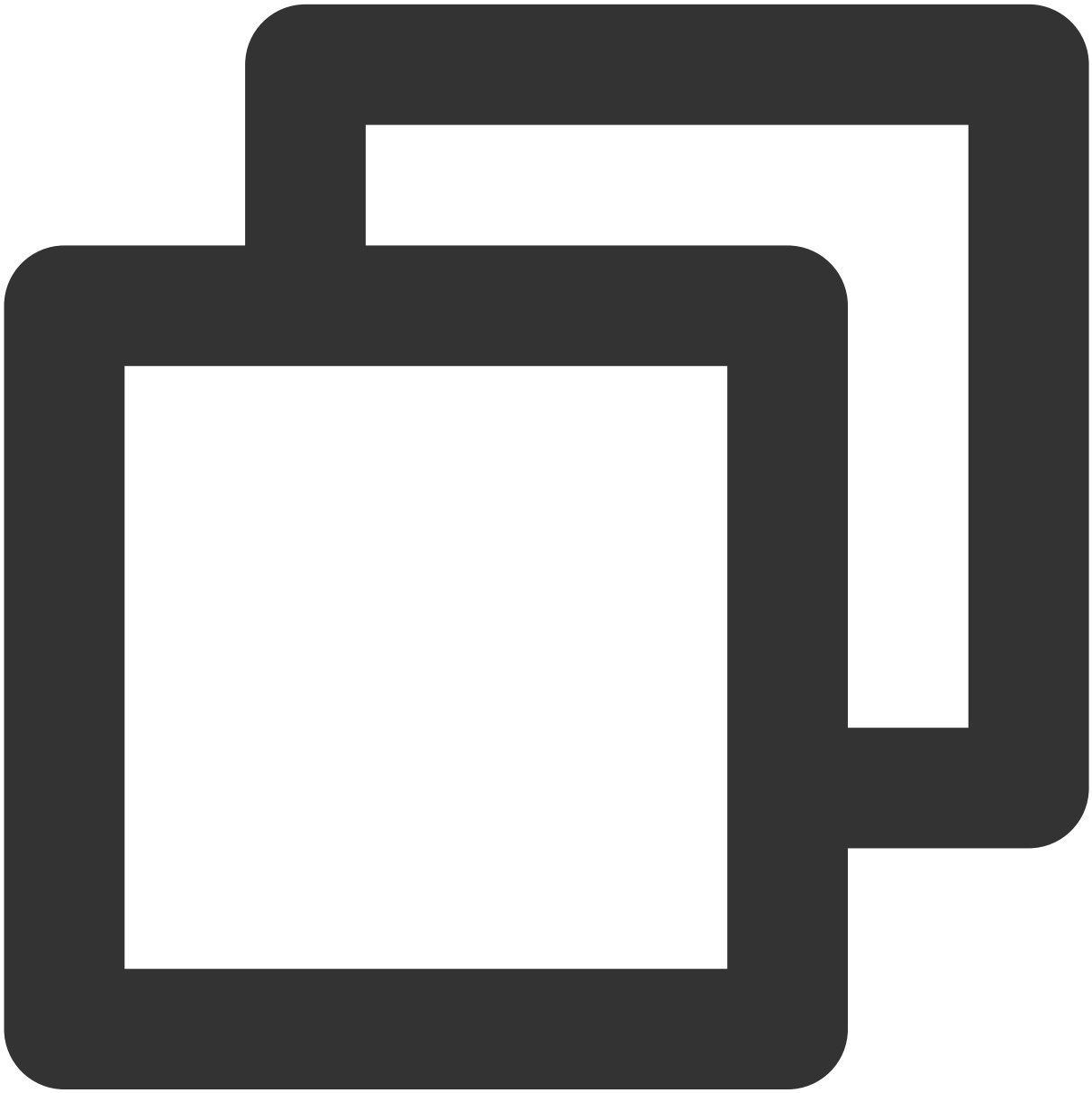
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
mute	BOOL	無効にするかどうか。

callback	TUIRoomActionCallback	結果のコールバック。

kickOffUser

キャスターがキックアウトします。



```
- (void)kickOffUser:(NSString *)userId
           callback:(TUIRoomActionCallback)callback;
```

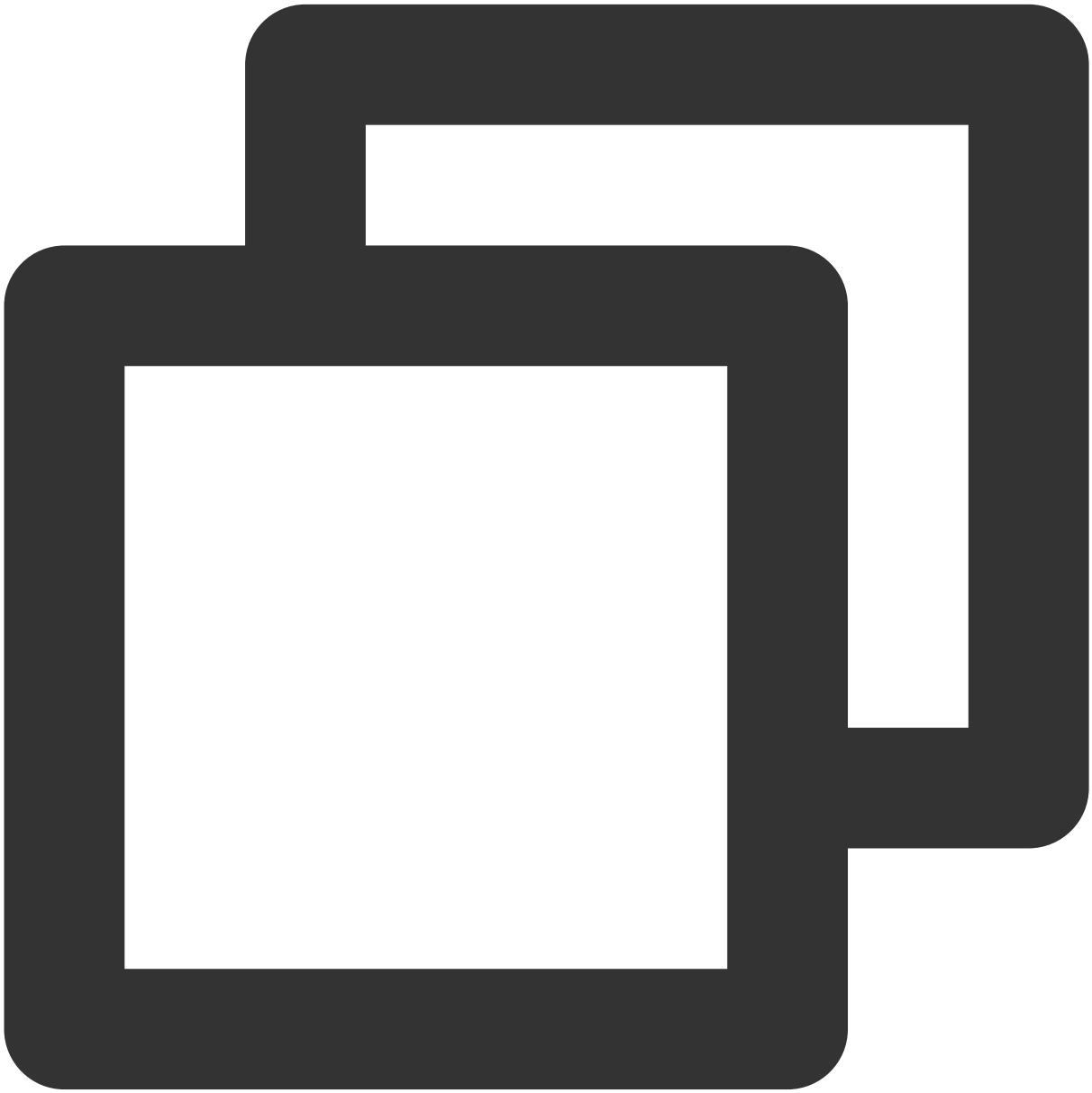
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	NSString	ユーザーID。
callback	TUIRoomActionCallback	結果のコールバック。

startCallingRoll

キャスターが点呼を開始します。



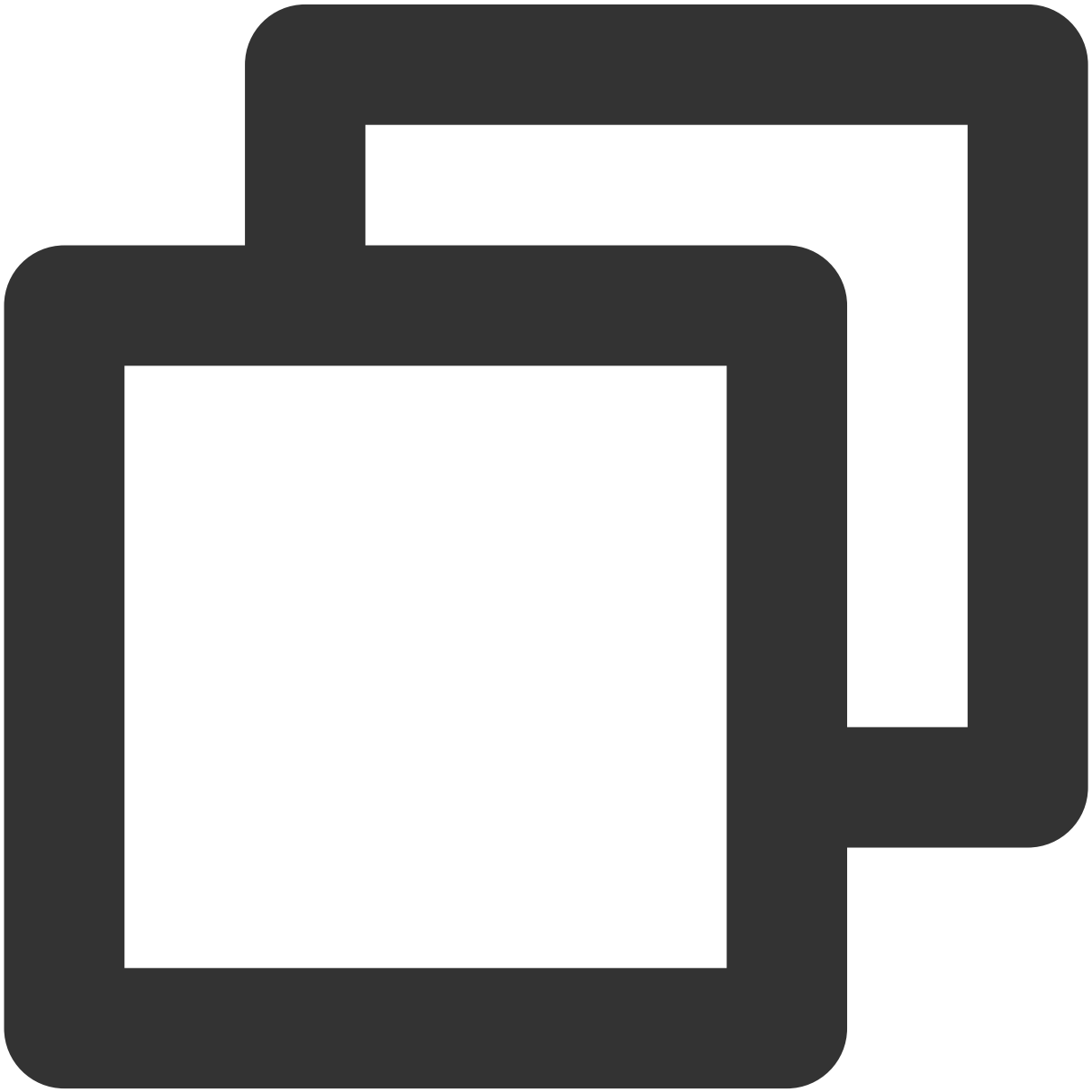
```
- (void)startCallingRoll:(TUIRoomActionCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	TUIRoomActionCallback	結果のコールバック。

stopCallingRoll

キャスターが点呼を終了します。



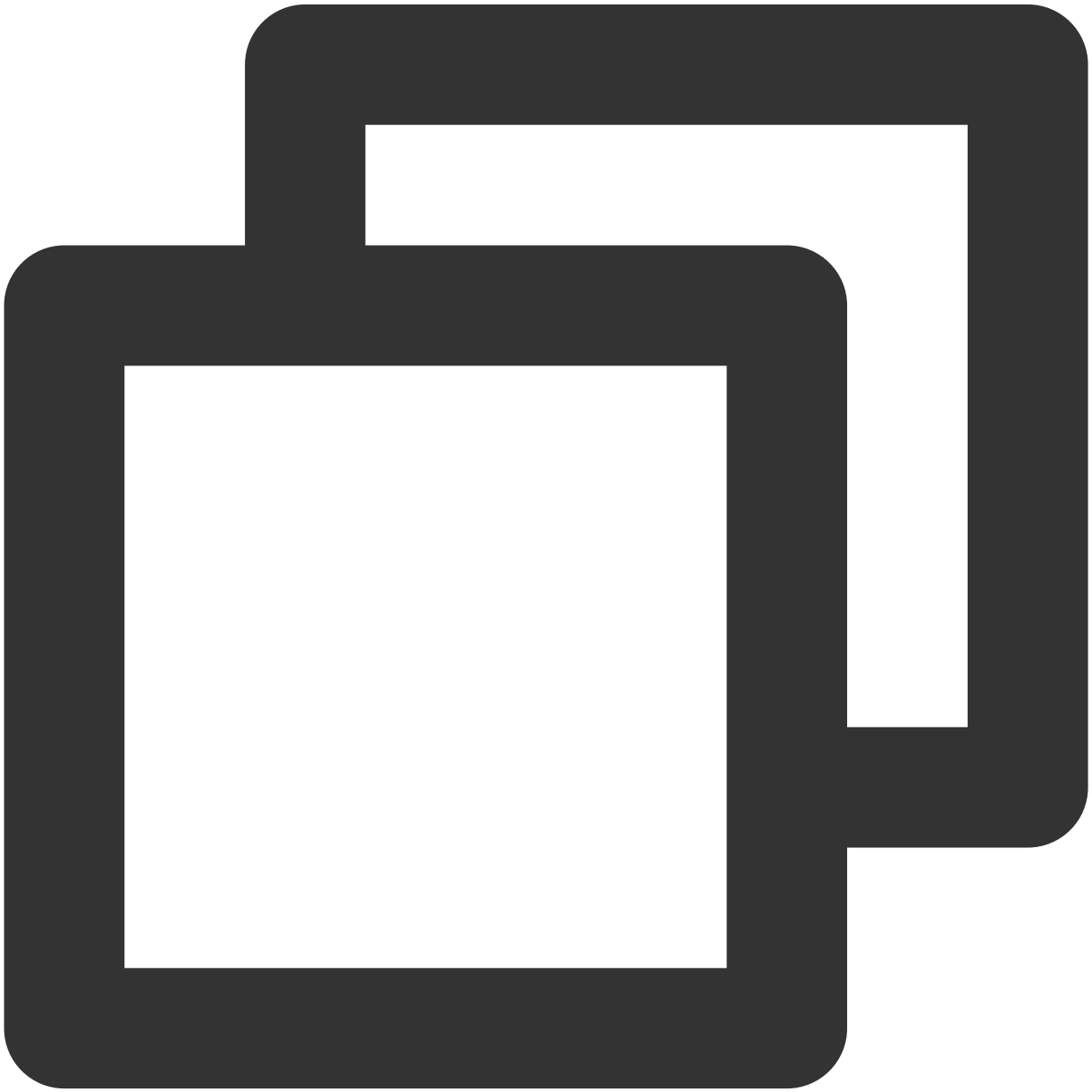
```
- (void)stopCallingRoll:(TUIRoomActionCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	TUIRoomActionCallback	結果のコールバック。

replyCallingRoll

参加者がキャスターの点呼に応答します。



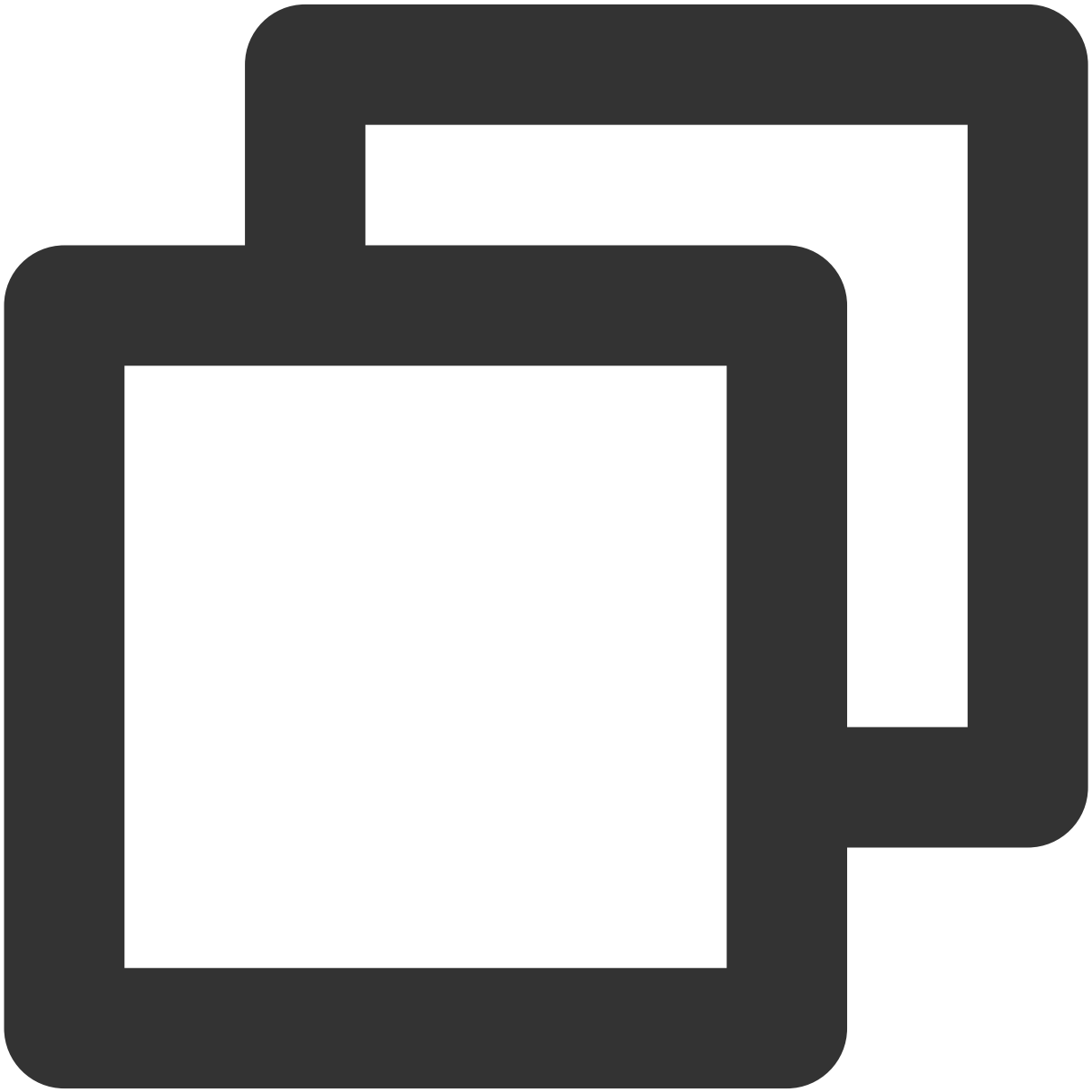
```
- (void)replyCallingRoll:(TUIRoomActionCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	TUIRoomActionCallback	結果のコールバック。

sendSpeechInvitation

キャスターが参加者の発言を要請します。



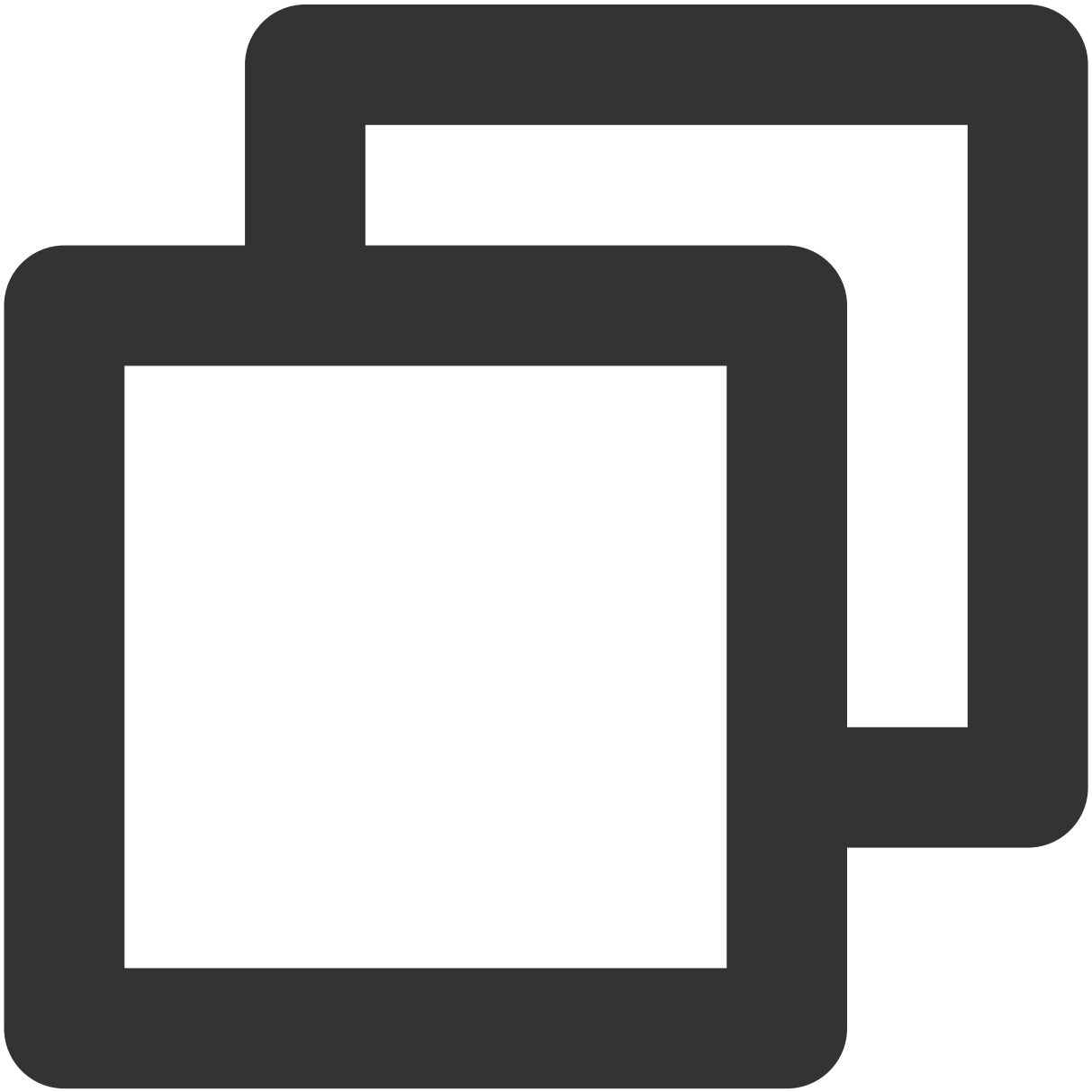
```
- (void)sendSpeechInvitation:(NSString *)userId
                    callback:(TUIRoomInviteeCallback)callback
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。
callback	TUIRoomInviteeCallback	結果のコールバック。

cancelSpeechInvitation

キャスターが参加者の発言要請をキャンセルします。



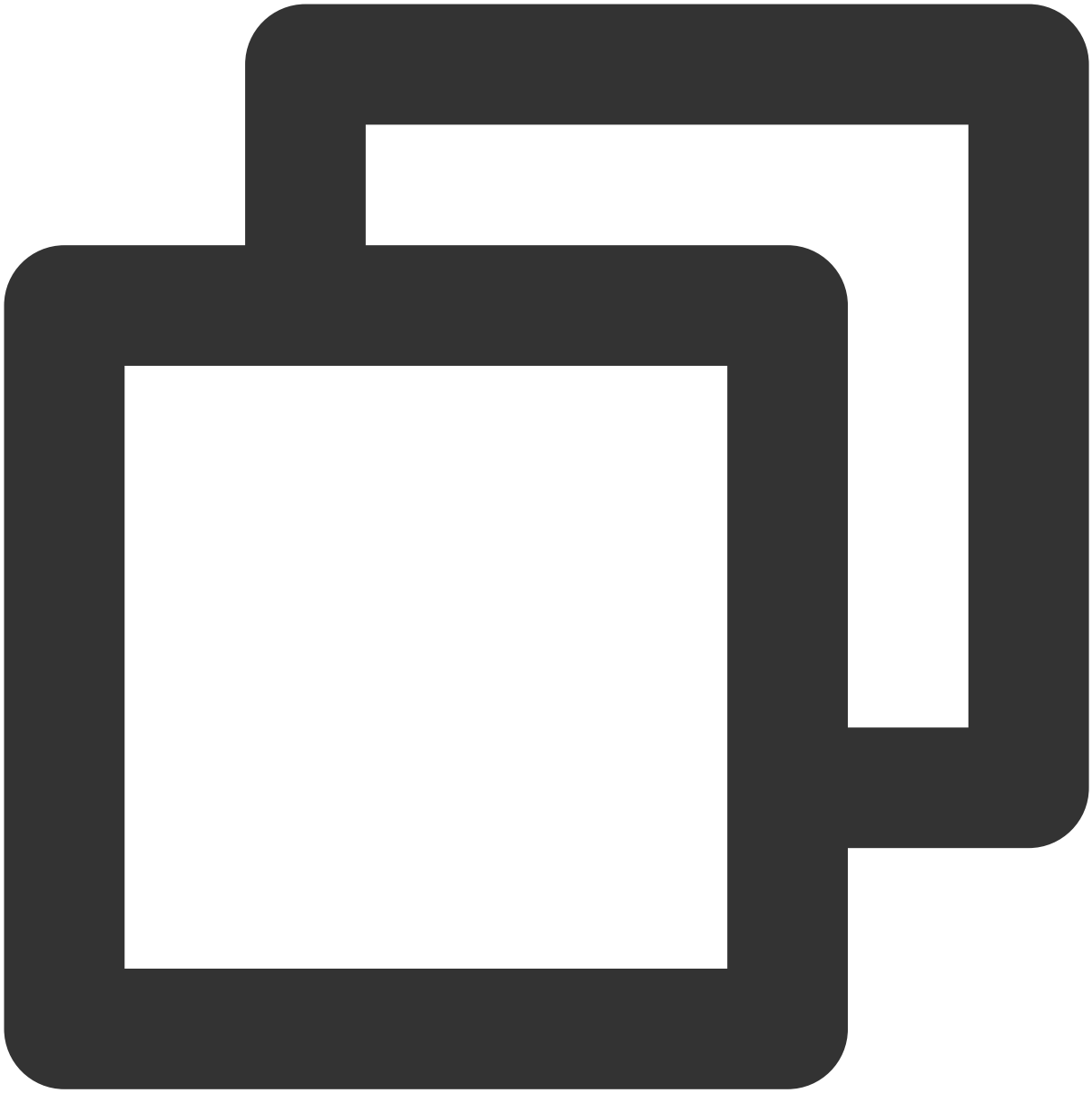
```
- (void)cancelSpeechInvitation:(NSString *)userId
                      callback:(TUIRoomActionCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。
callback	TUIRoomActionCallback	結果のコールバック。

replySpeechInvitation

参加者がキャスターの発言要請に同意/拒否します。



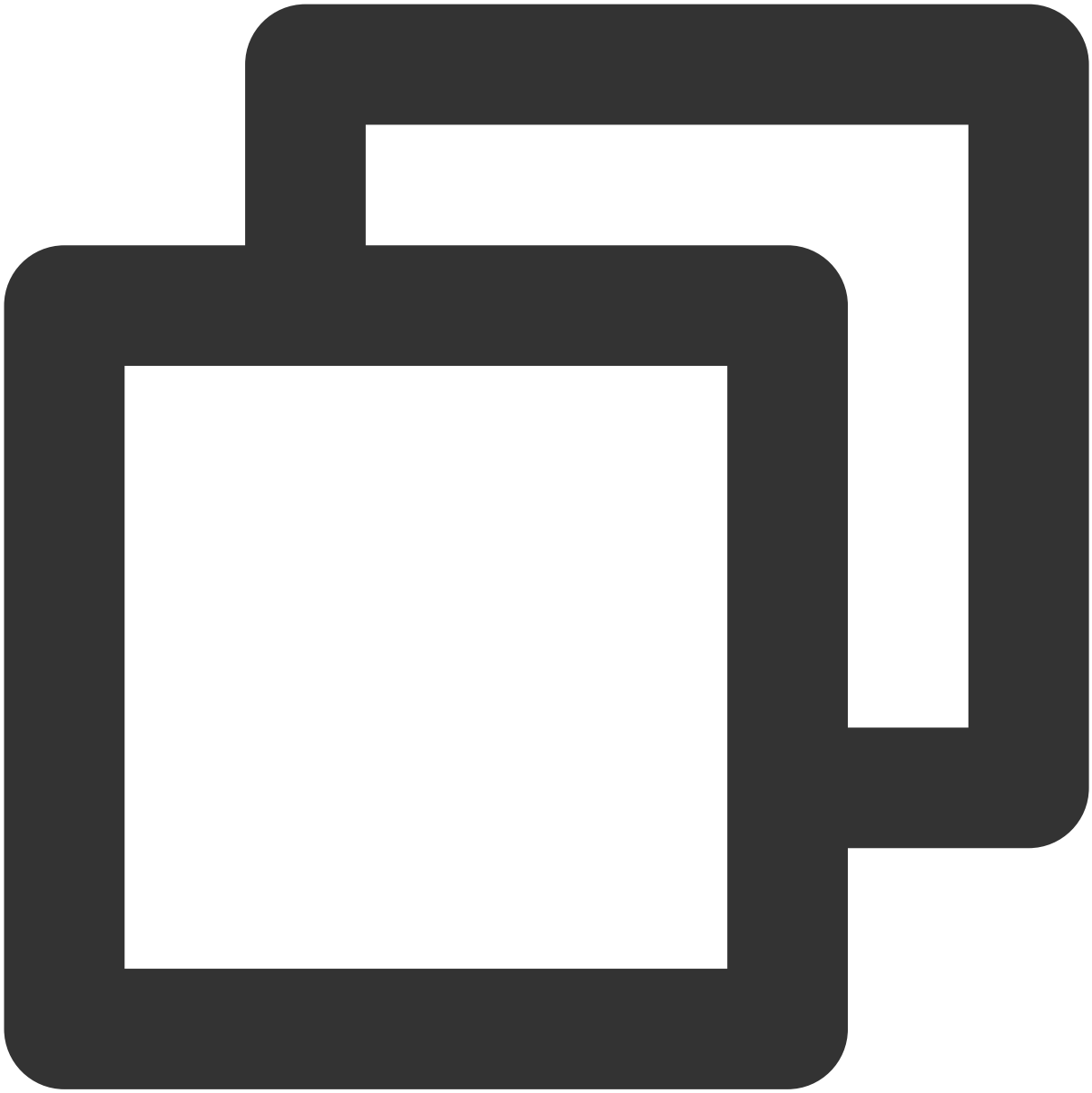
```
- (void)replySpeechInvitation:(BOOL)agree
                     callback:(TUIRoomActionCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
agree	BOOL	同意するかどうか。
callback	TUIRoomActionCallback	結果のコールバック。

sendSpeechApplication

参加者が発言を申請します。



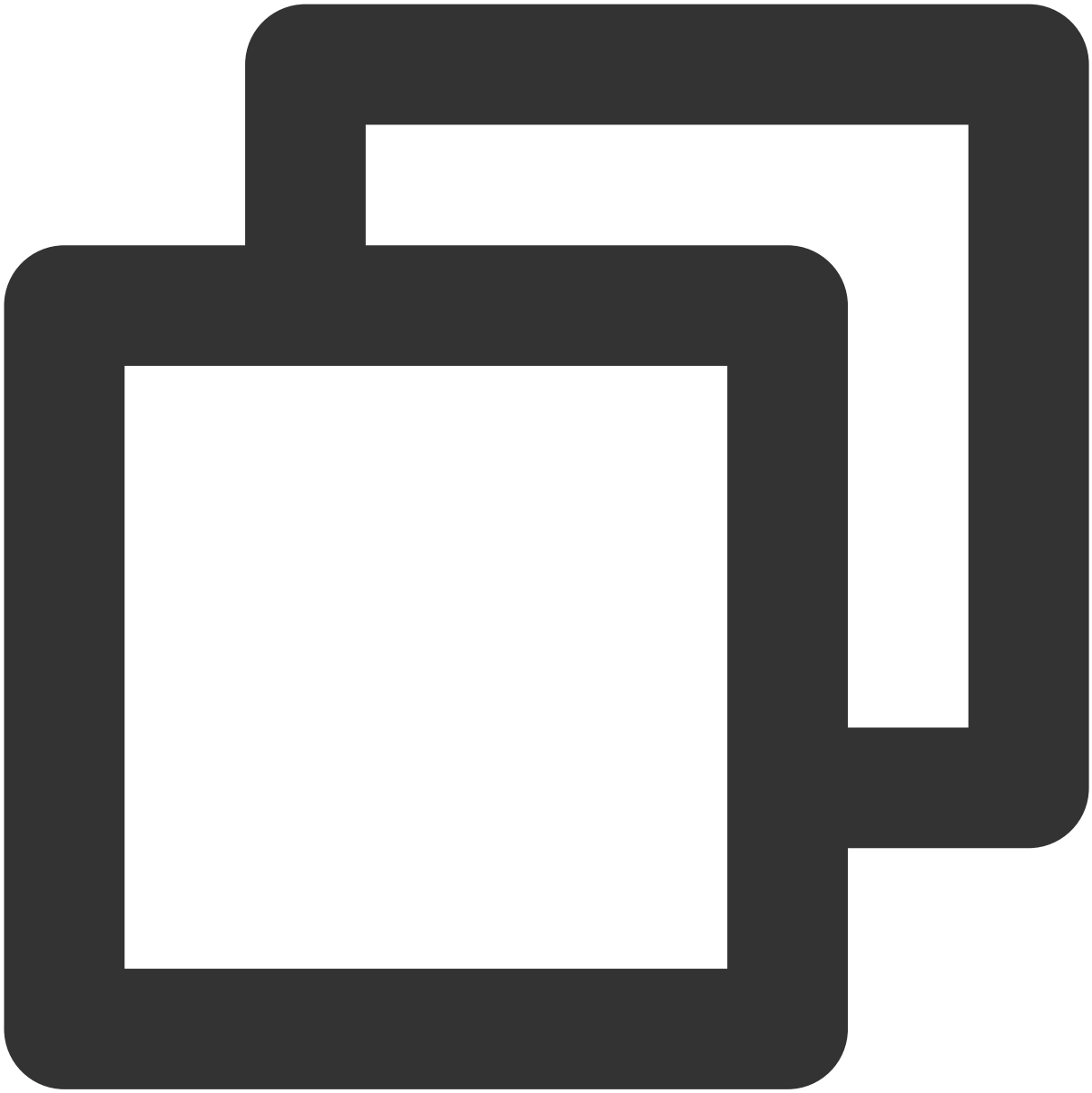
```
- (void)sendSpeechApplication:(TUIRoomInviteeCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	TUIRoomInviteeCallback	結果のコールバック。

cancelSpeechApplication

参加者が発言申請をキャンセルします。



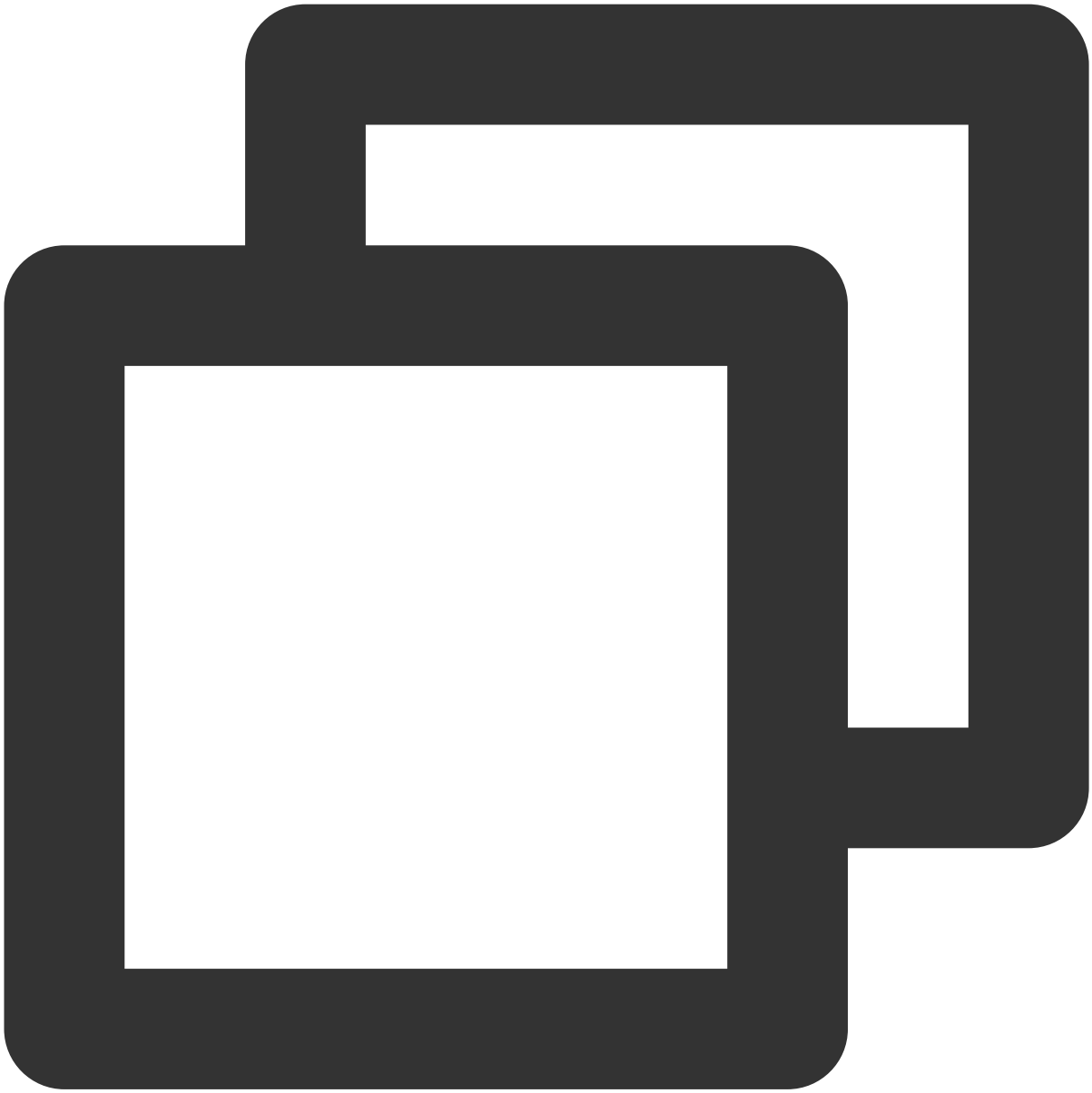
```
- (void)cancelSpeechApplication:(TUIRoomActionCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	TUIRoomActionCallback	結果のコールバック。

replySpeechApplication

キャスターが参加者の発言申請に同意/拒否します。



```
- (void)replySpeechApplication:(BOOL)agree
                        userId:(NSString *)userId
                      callback:(TUIRoomActionCallback)callback;
```

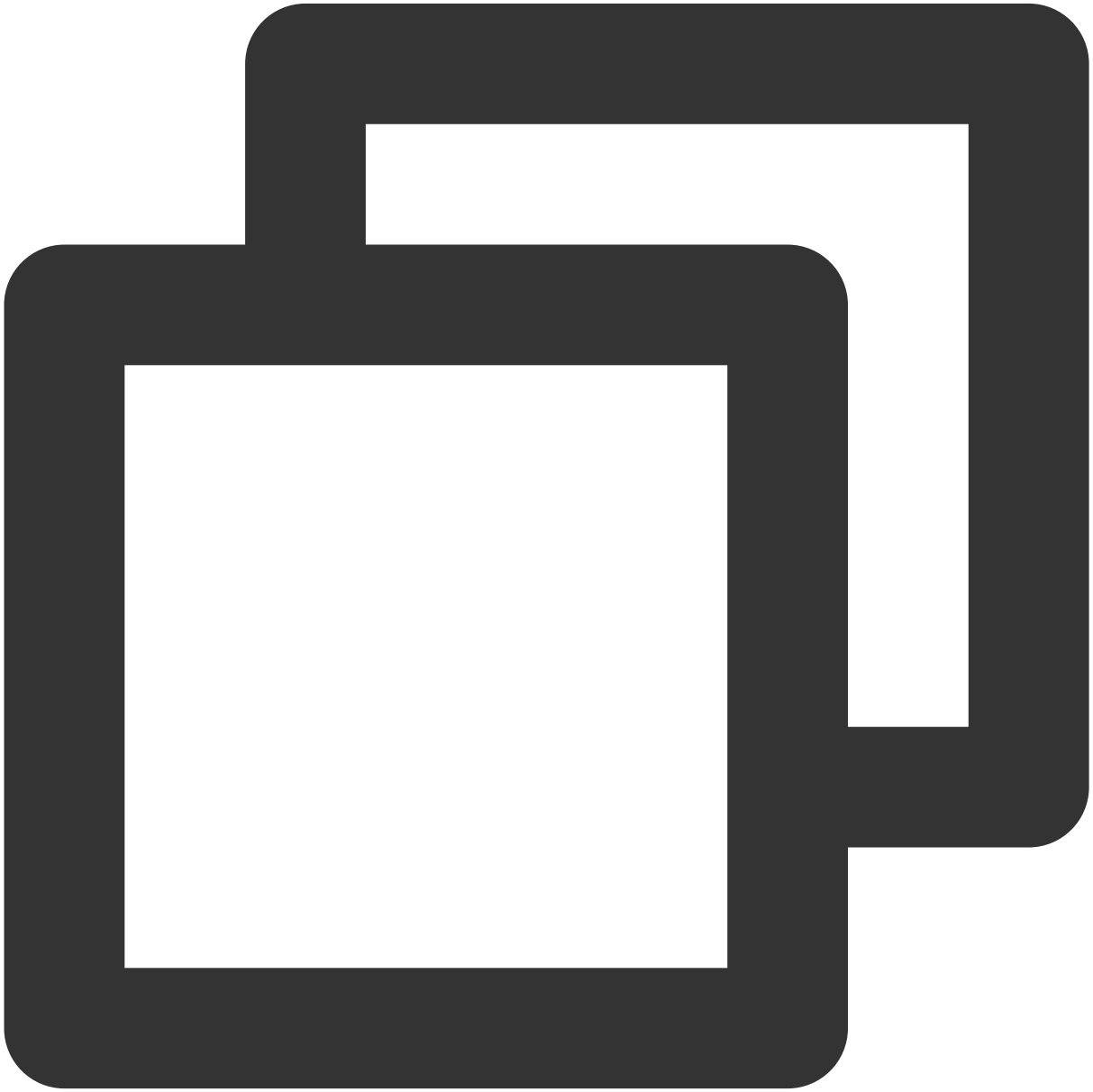
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

agree	BOOL	同意するかどうか
userId	NSString	ユーザーID。
callback	TUIRoomActionCallback	結果のコールバック。

forbidSpeechApplication

キャスターが発言申請を禁止します。



```
- (void)forbidSpeechApplication:(BOOL)forbid
```

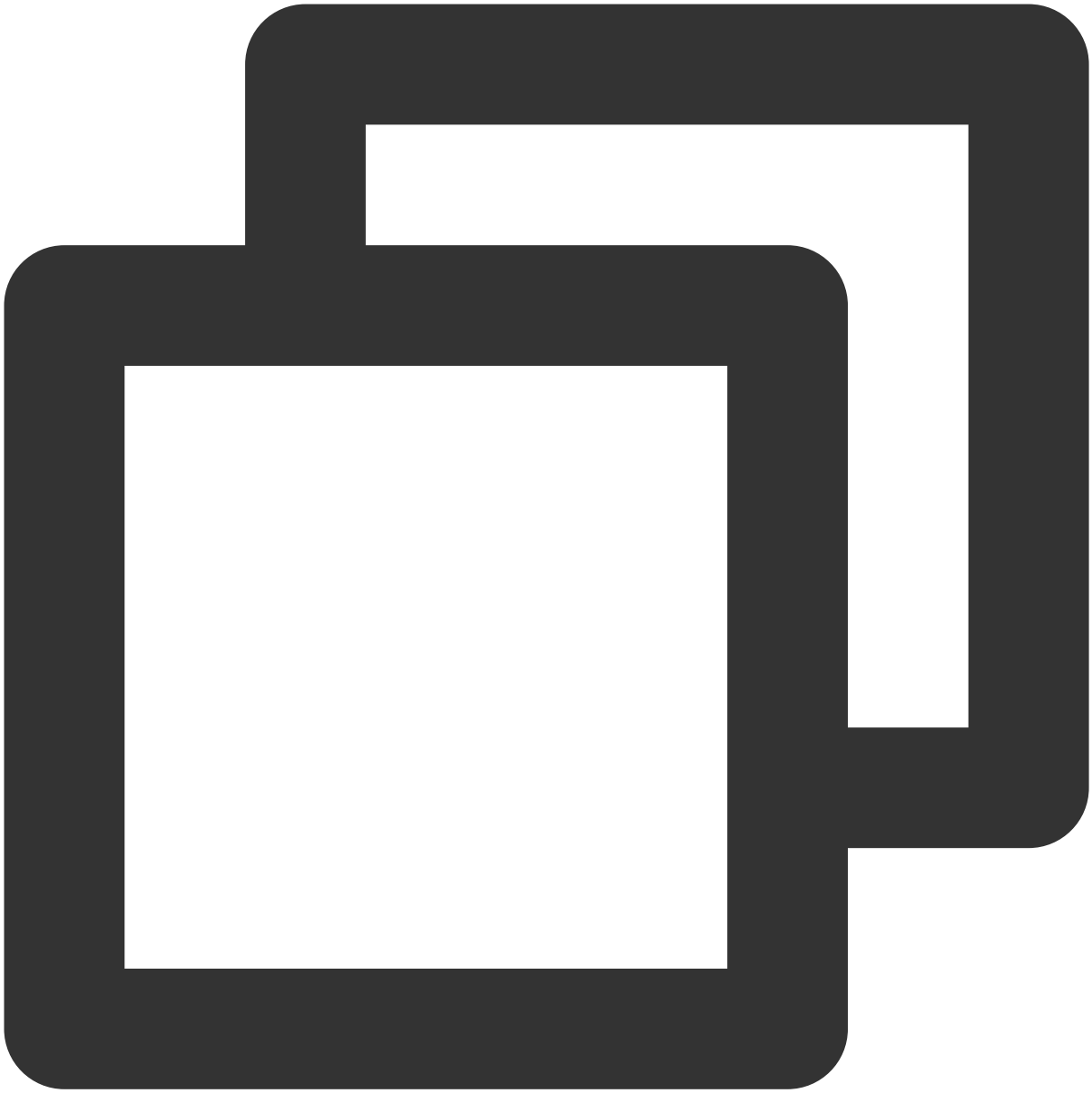
```
callback:(TUIRoomActionCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
forbid	BOOL	禁止するかどうか。
callback	TUIRoomActionCallback	結果のコールバック。

sendOffSpeaker

キャスターが参加者に発言の停止を命令します。



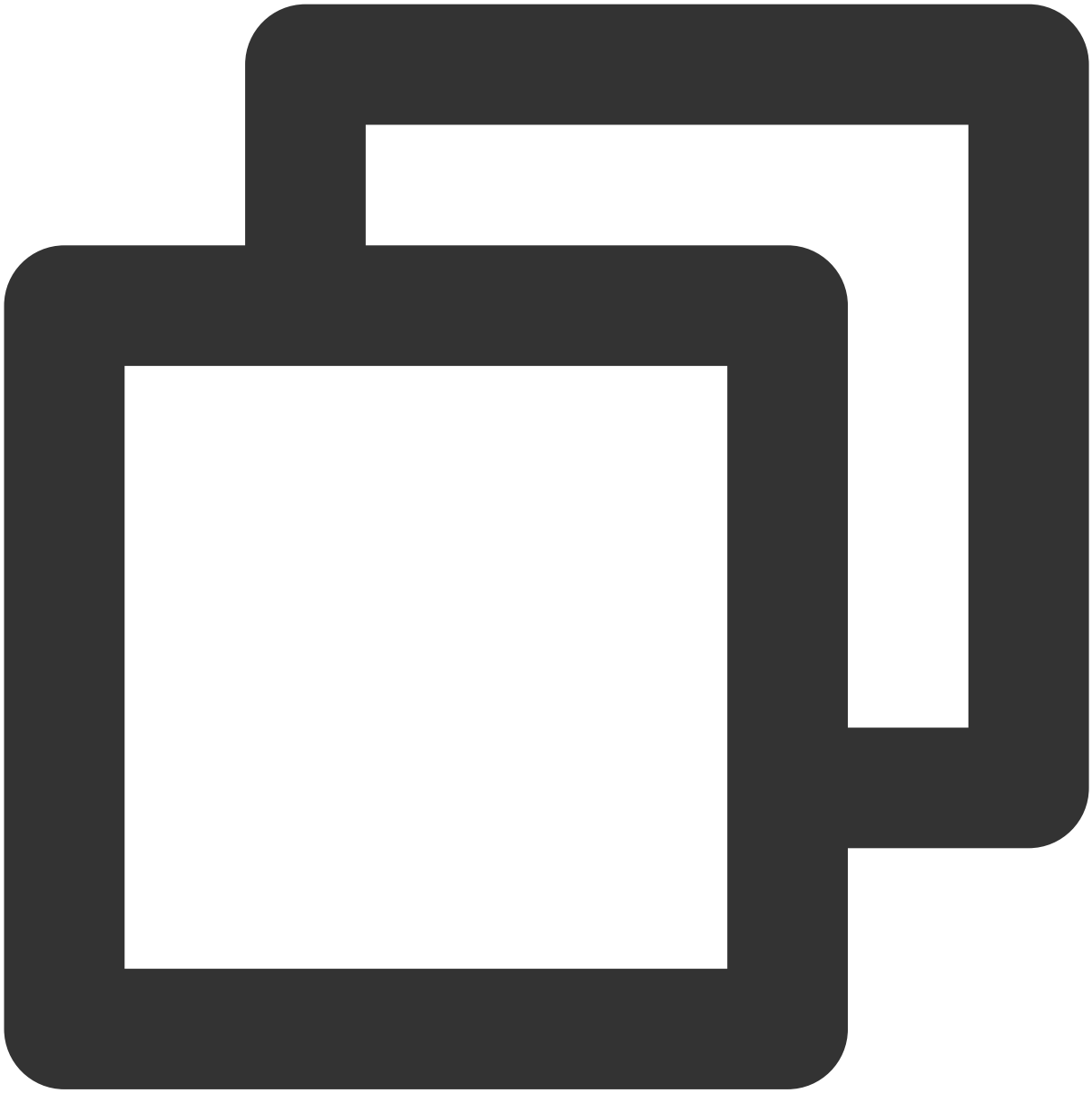
```
- (void)sendOffSpeaker:(NSString *)userId
              callback:(TUIRoomInviteeCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。
callback	TUIRoomInviteeCallback	結果のコールバック。

sendOffAllSpeakers

キャスターが全メンバーに発言の停止を命令します。



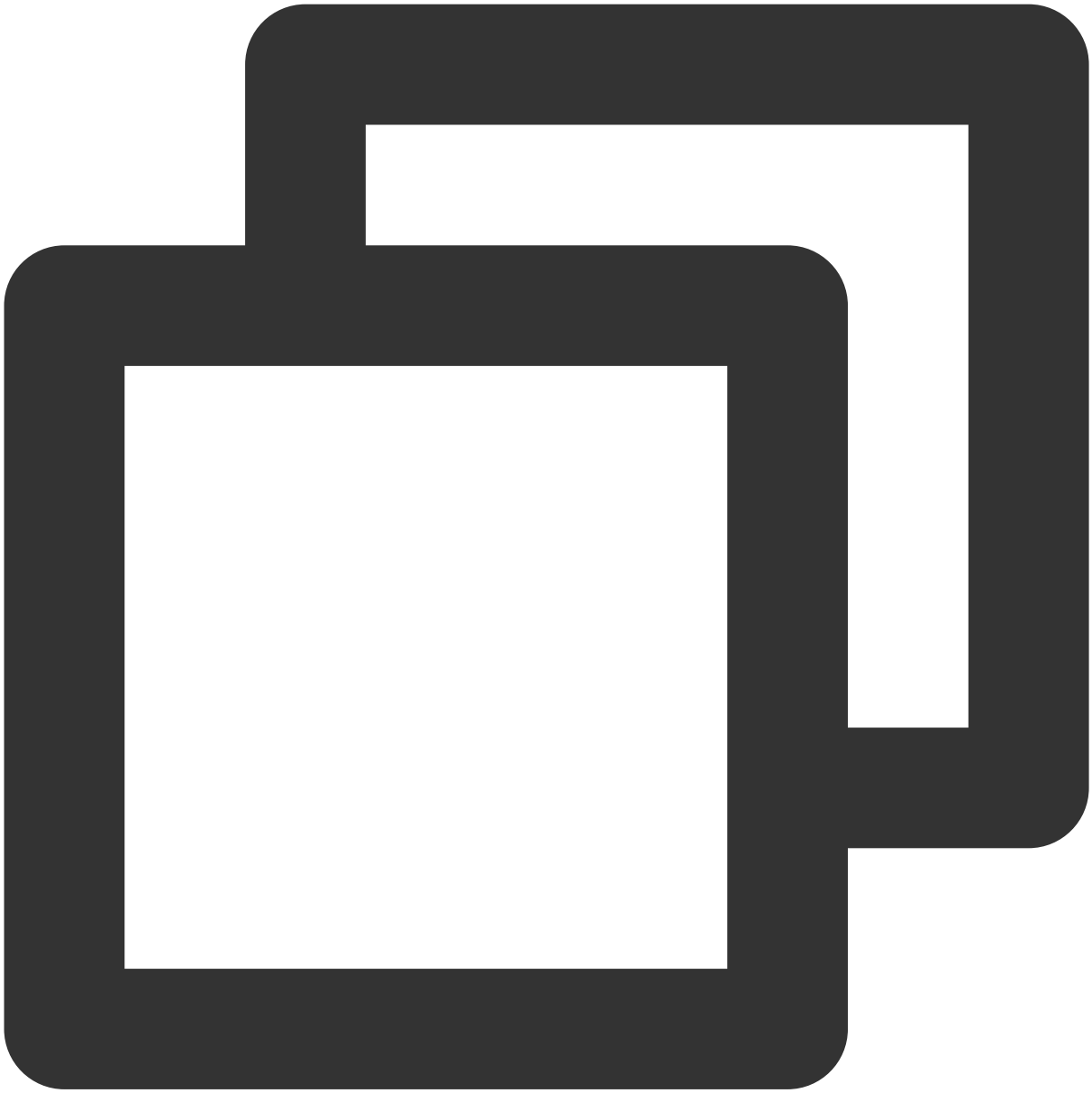
```
- (void)sendOffAllSpeakers:(TUIRoomInviteeCallback)callback;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	TUIRoomInviteeCallback	結果のコールバック。

exitSpeechState

参加者が発言を停止し、視聴者になります。



```
- (void)exitSpeechState:(TUIRoomActionCallback)callback;
```

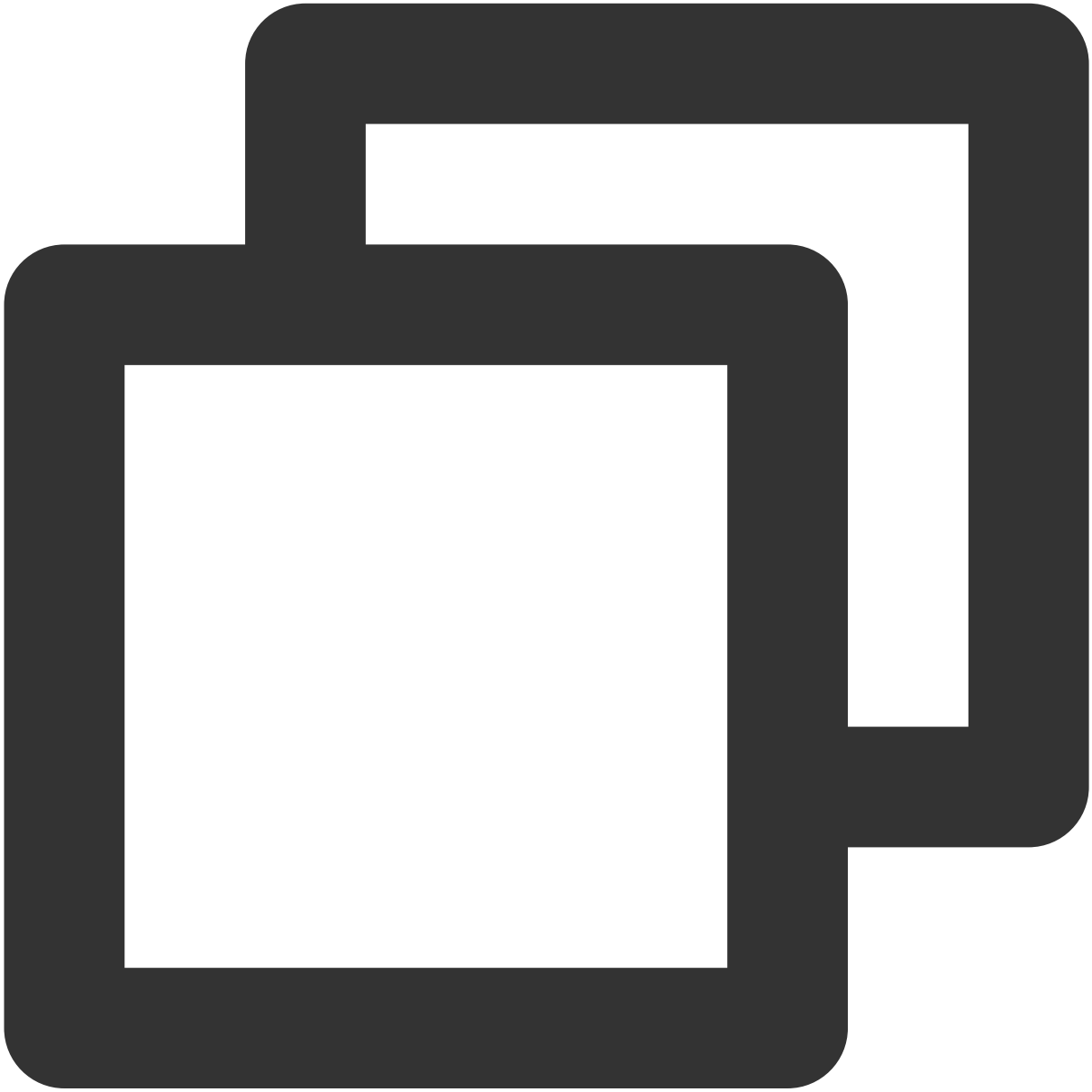
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	TUIRoomActionCallback	結果のコールバック。

画面共有インターフェース

startScreenCapture

画面共有を開始。



```
- (void)startScreenCapture:(TRTCVideoEncParam *)encParam API_AVAILABLE(ios(11.0));
```

パラメータは下表に示すとおりです。

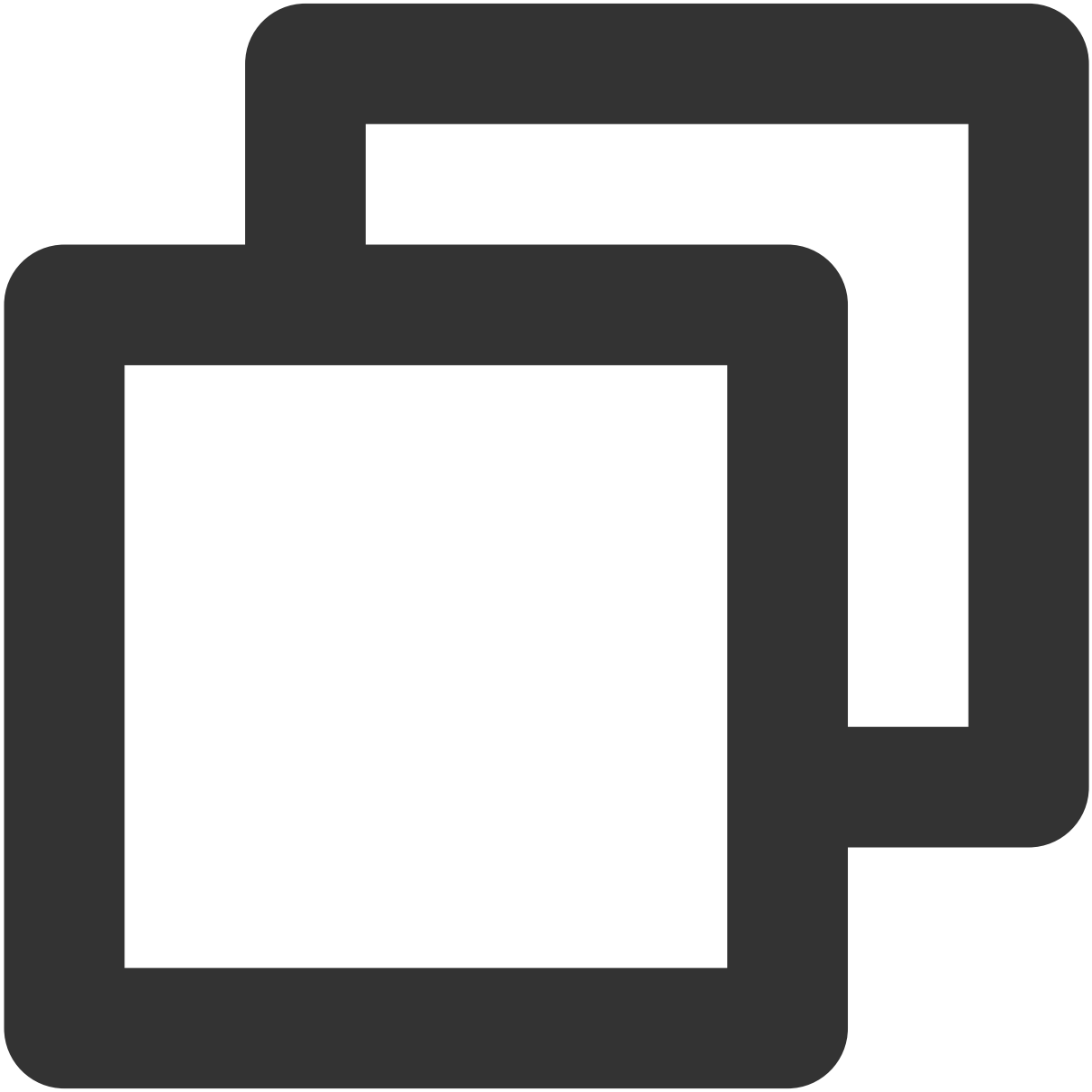
パラメータ	タイプ	意味

encParams	TRTCVideoEncParam	画面共有時のエンコードパラメータを設定します。

説明：
詳細については、TRTC SDKをご参照ください。

stopScreenCapture

画面キャプチャの停止。

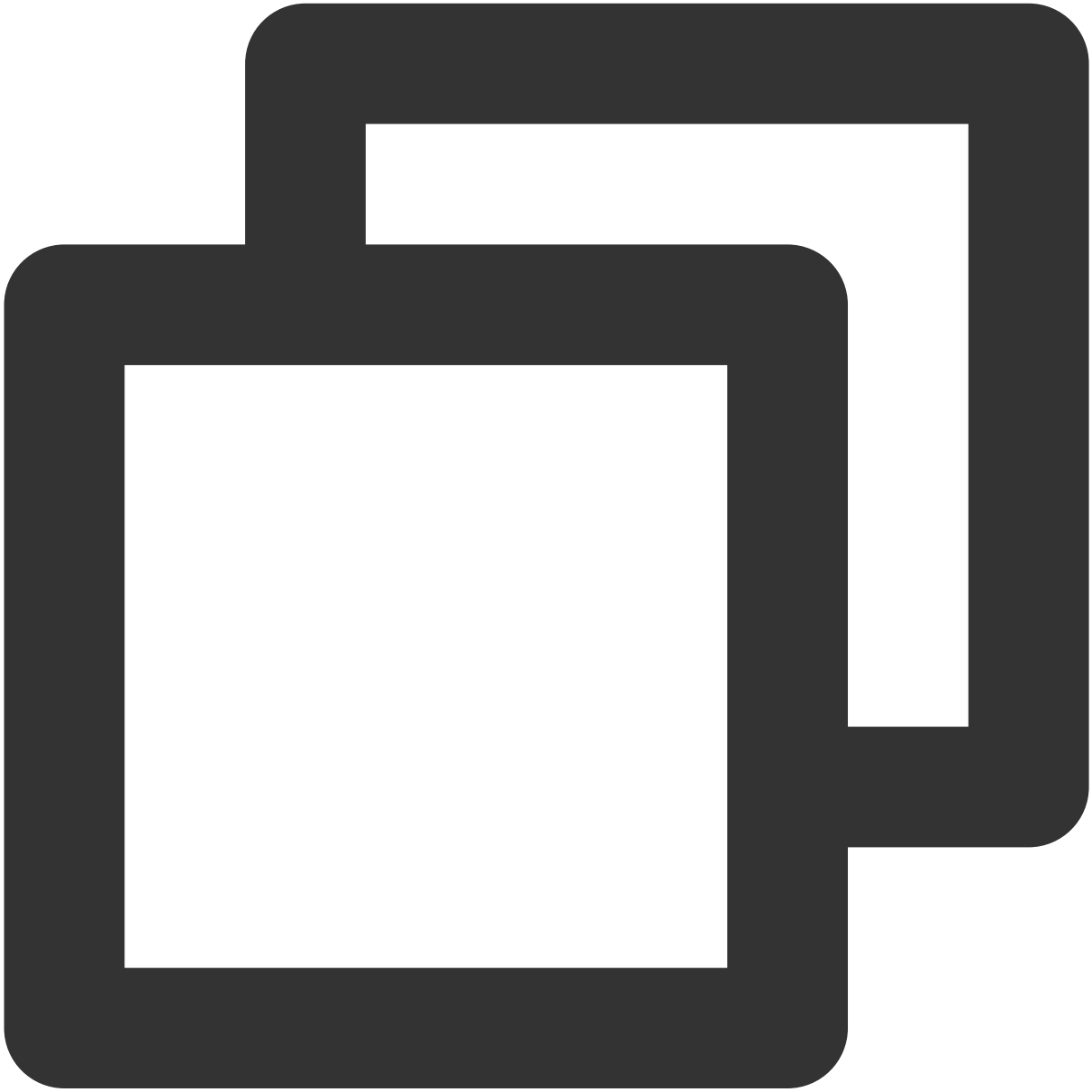


```
- (void)stopScreenCapture API_AVAILABLE(ios(11.0));
```

美顔フィルターに関するインターフェース関数

getBeautyManager

美顔管理オブジェクトTXBeautyManagerを取得します。



```
- (TXBeautyManager *)getBeautyManager;
```

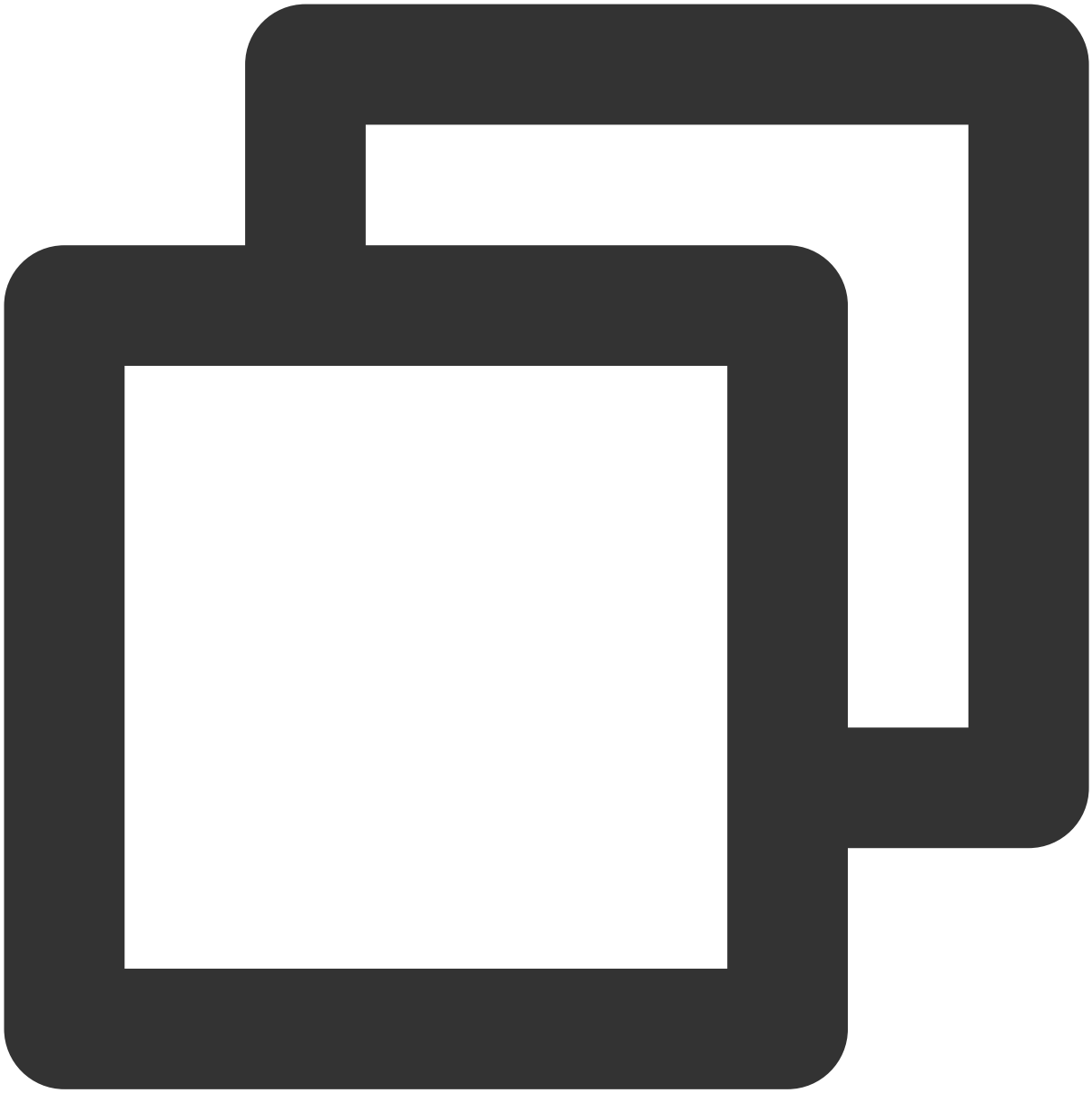
美顔管理では、次の機能を使用できます。

「美顔のスタイル」、「美白」、「肌色補正（血色・つや感）」、「デカ眼」、「顔痩せ」、「V顔」、「下あご」、「面長補正」、「小鼻」、「キラキラ目」、「白い歯」、「目の弛み除去」、「シワ除去」、「ほうれい線除去」などの美容効果を設定します。
「髪の生え際」、「眼と眼の距離」、「眼の角度」、「唇の形」、「鼻翼」、「鼻の位置」、「唇の厚さ」、「顔の形」を調整します。
人の顔のスタンプ（素材）等のダイナミック効果を設定します。
メイクアップを追加します。
ジェスチャー認識を行います。

関連設定インターフェース

setVideoQosPreference

ネットワークトラフィックコントロール関連パラメータを設定します。



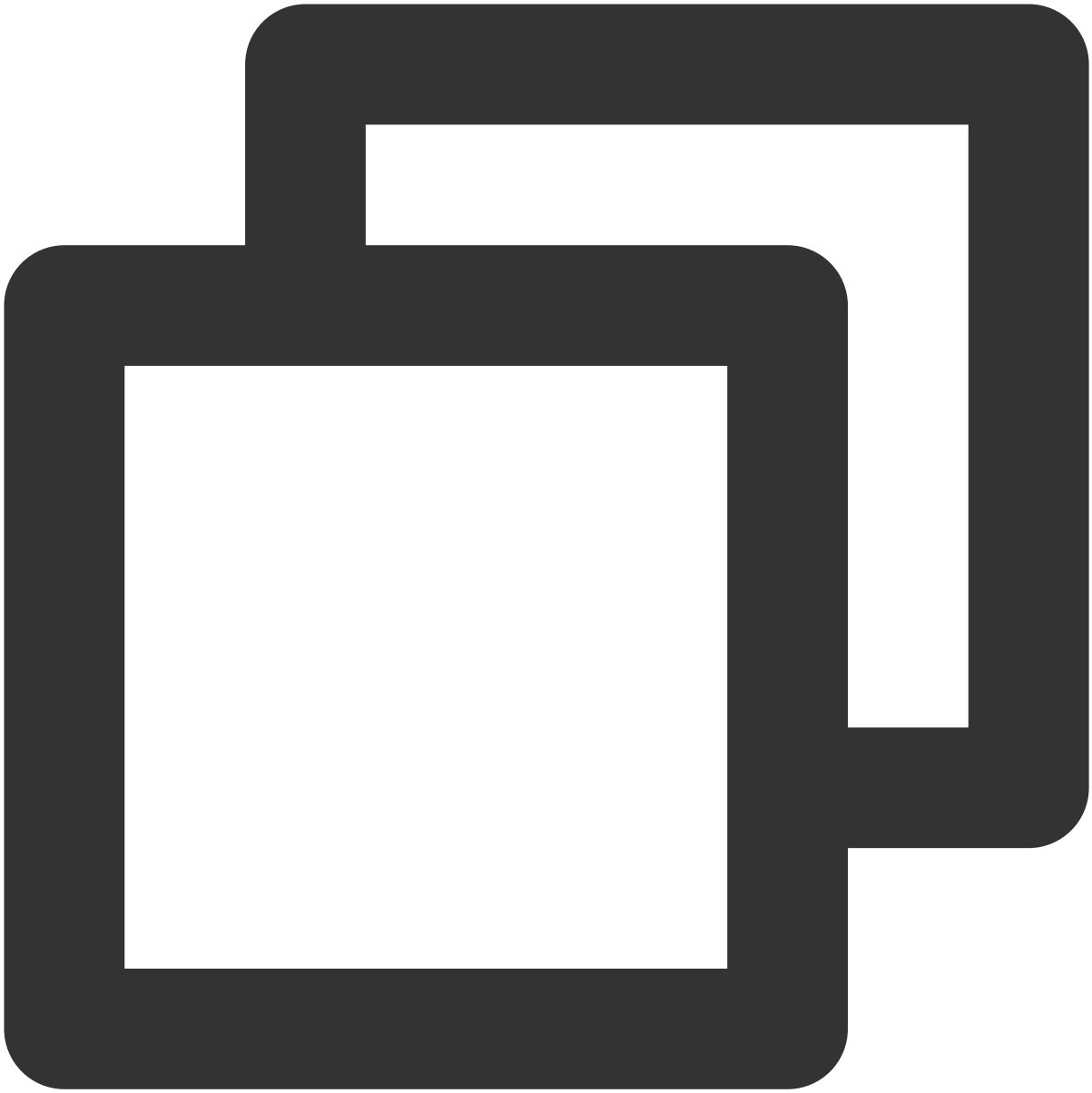
```
- (void)setVideoQosPreference:(TRTCNetworkQosParam *)preference;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
preference	TRTCNetworkQosParam	ネットワークトラフィックコントロールポリシー。

setAudioQuality

音質の設定



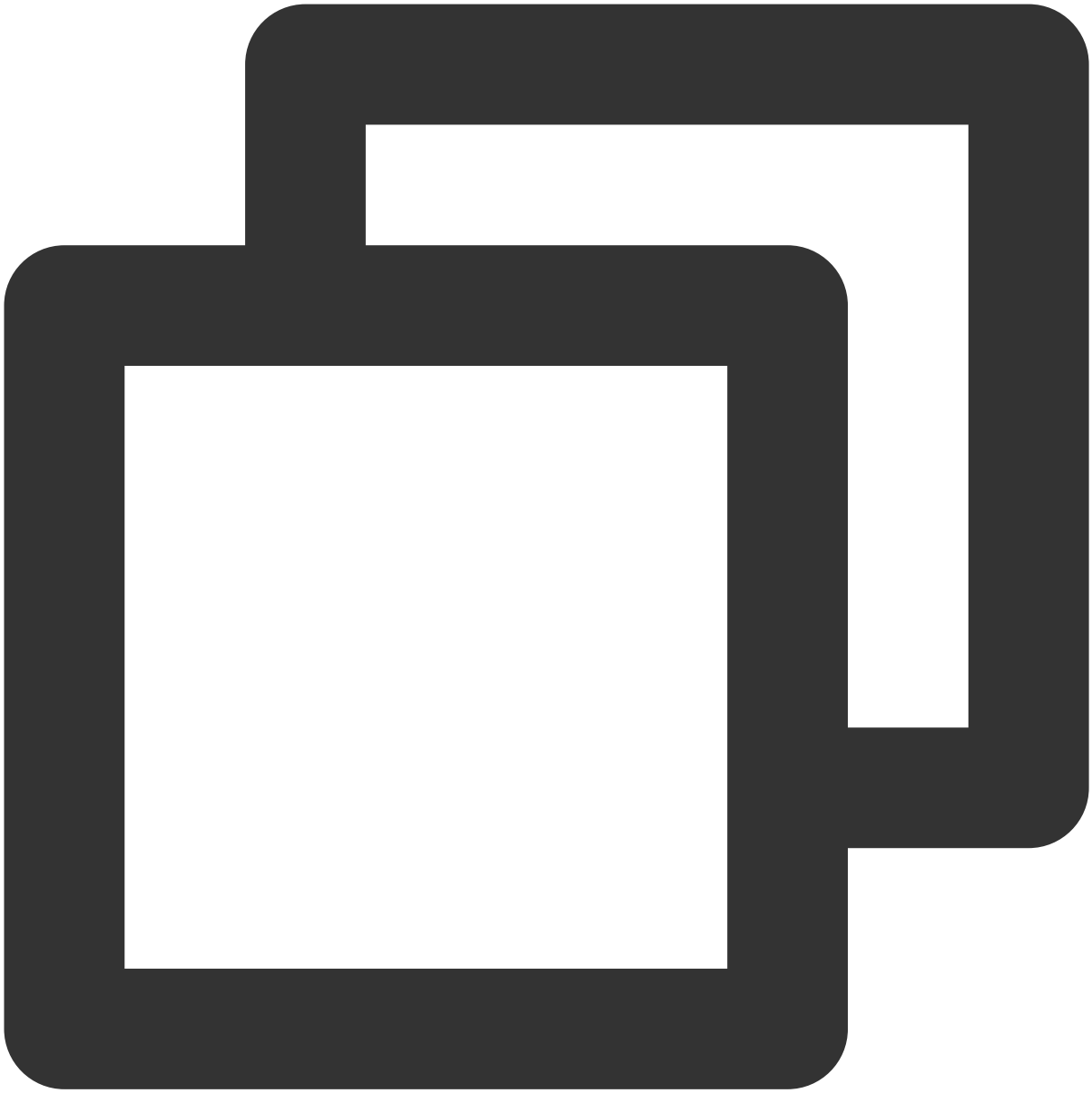
```
- (void)setAudioQuality:(TRTCAudioQuality)quality;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
quality	TRTCAudioQuality	音質。詳細については、TRTC SDKをご参照ください。

setVideoResolution

解像度の設定。



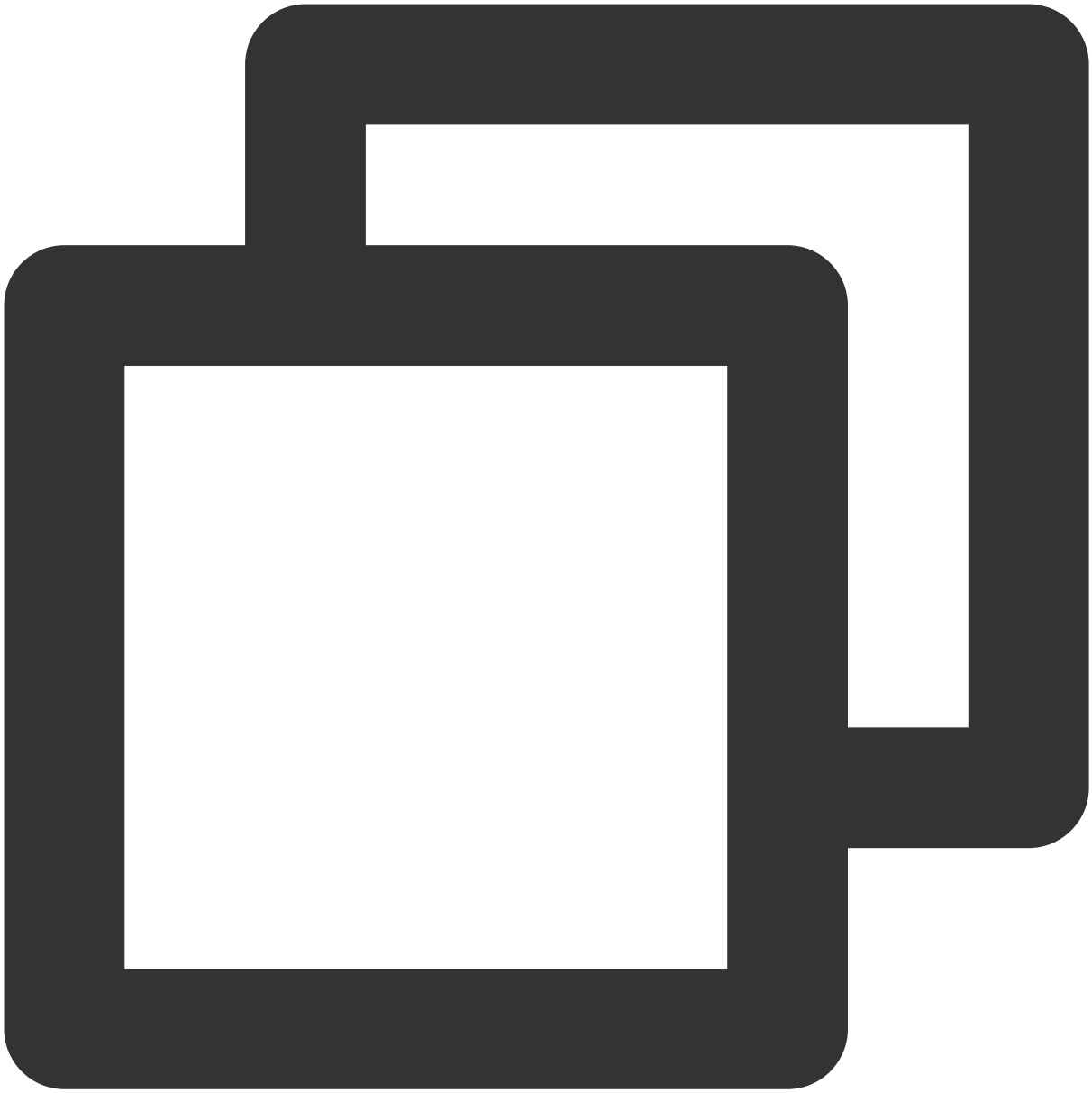
```
- (void)setVideoResolution:(TRTCVideoResolution)resolution;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
resolution	TRTCVideoResolution	ビデオの解像度。詳細については、TRTC SDKをご参照ください。

setVideoFps

フレームレートの設定。



```
- (void)setVideoFps:(int)fps;
```

パラメータは下表に示すとおりです。

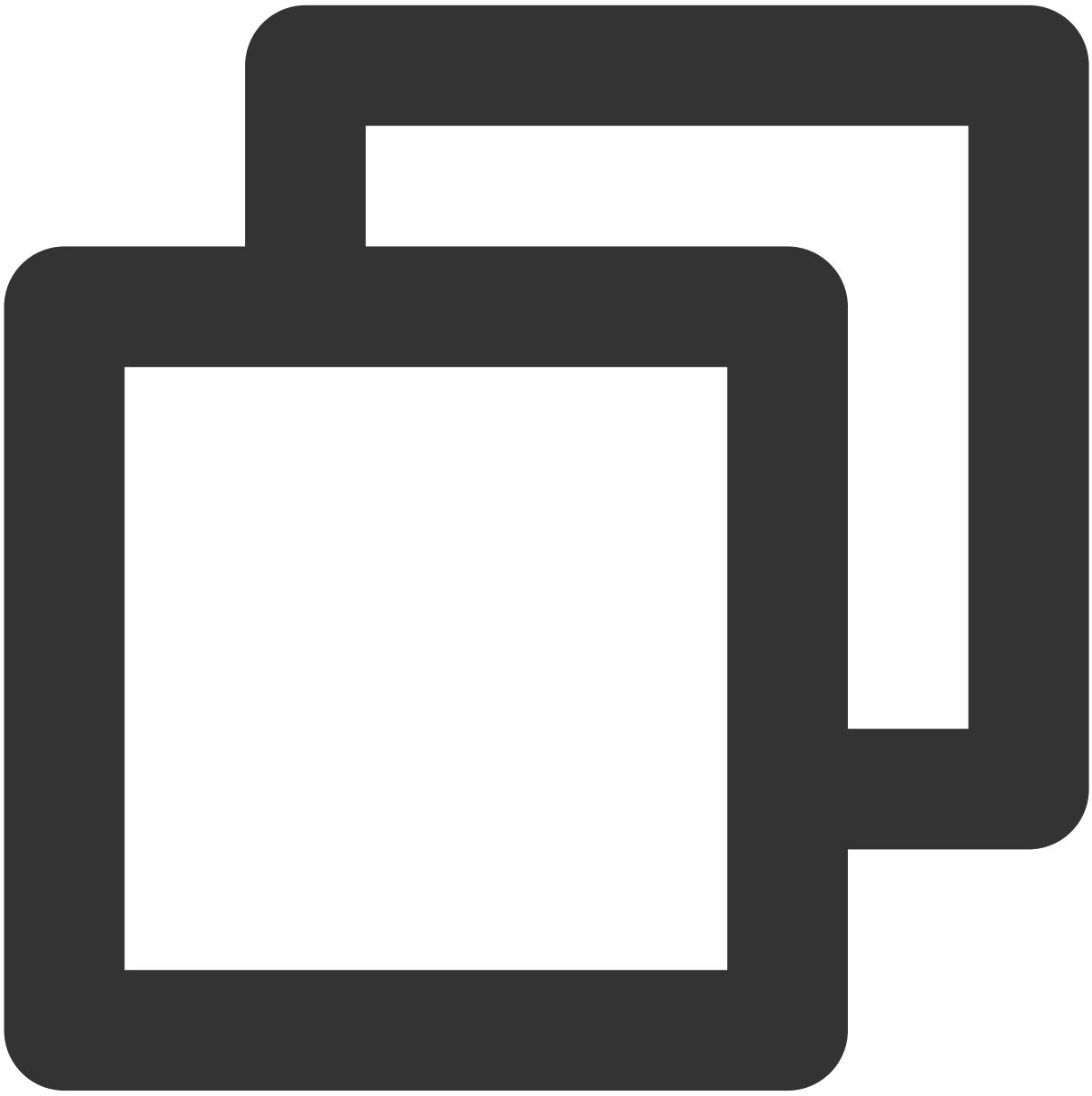
パラメータ	タイプ	意味
fps	int	ビデオキャプチャのフレームレート。

説明：

推奨する値：15fpsまたは20fps。5fps以下ではラグ感が目立ち、10fps以下では軽微なラグ感があります。20fps以上は高すぎて浪費になります（映画のフレームレートは24fps）。

setVideoBitrate

ビットレートの設定。



```
- (void)setVideoBitrate:(int)bitrate;
```

パラメータは下表に示すとおりです。

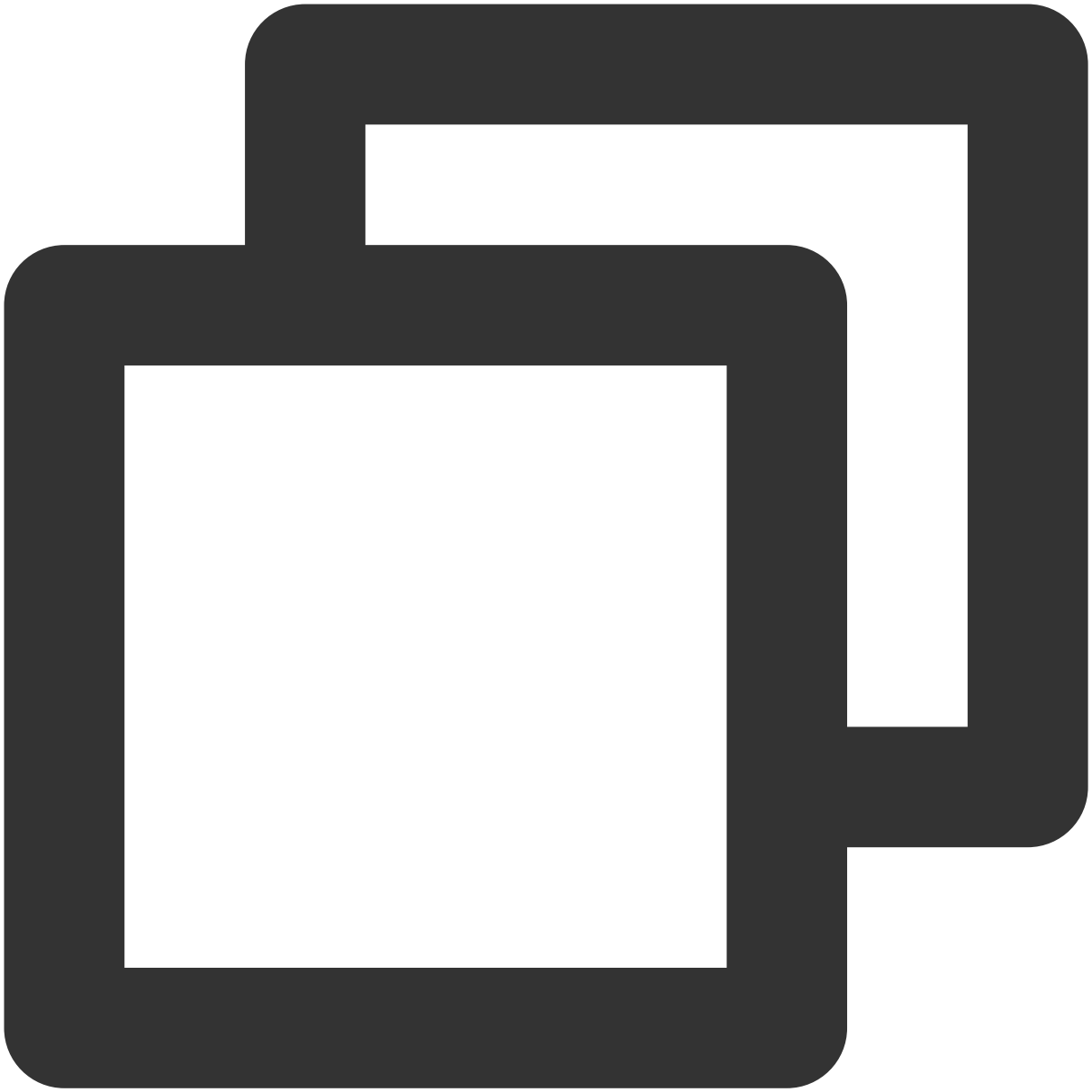
パラ	タ	意味

メータ	イプ	
bitrate	int	ビットレート。SDKは、目標ビットレートに応じてエンコードを行い、ネットワークの状態が良くない場合のみ、ビデオビットレートを動的に引き下げます。詳細については、TRTC SDKをご参照ください。

説明：
推奨する値：TRTCVideoResolutionの各クラスに注記する最適ビットレートをご参照ください。これをもとにより高いレートに適宜調整することも可能です。例えば、TRTC_VIDEO_RESOLUTION_1280_720に対応する目標ビットレートが1200kbpsであるならば、設定を1500kbpsにし、より鮮明な画像を得ることができます。

enableAudioEvaluation

音量レベルリマインダを有効にします。



```
- (void)enableAudioEvaluation:(BOOL)enable;
```

パラメータは下表に示すとおりです。

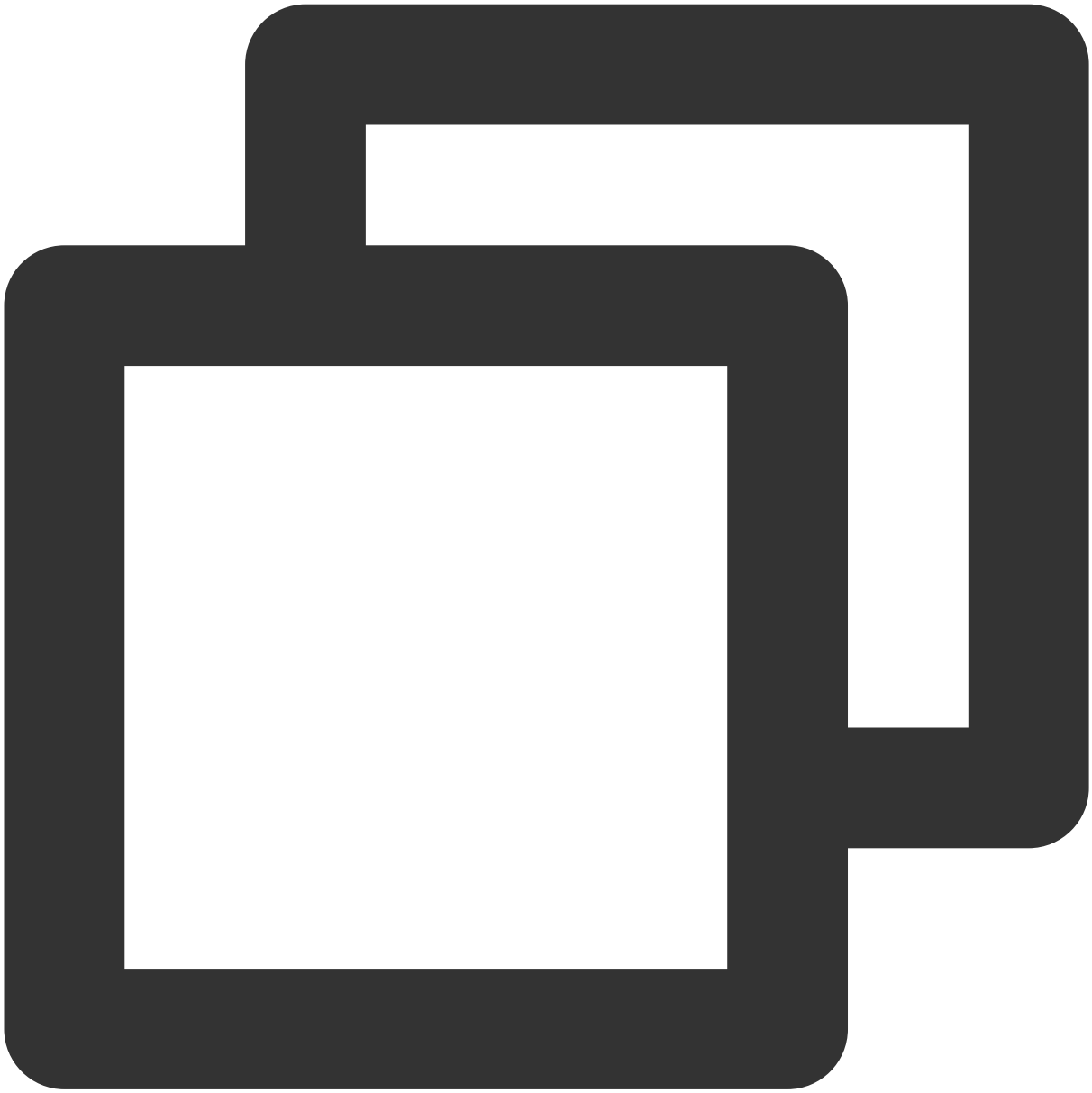
パラメータ	タイプ	意味
enable	BOOL	YES：オン、NO：オフ。

説明：

有効化すると、onUserVolumeUpdateの中でSDKの音量のボリュームに対する評価を取得できます。

setAudioPlayVolume

再生音量の設定。



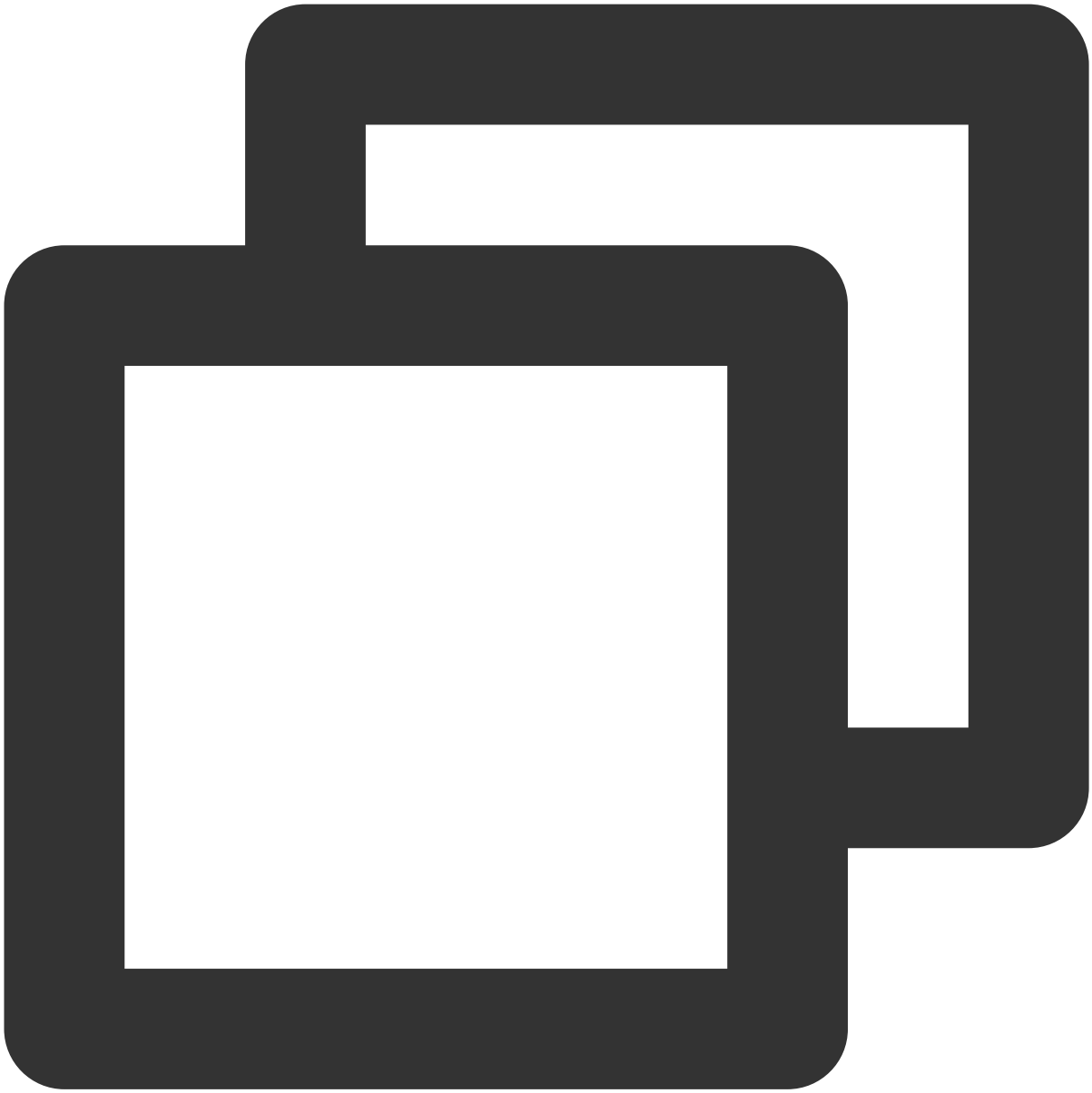
```
- (void)setAudioPlayVolume:(NSInteger)volume;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
volume	int	再生音量、0〜100、 デフォルト100。

setAudioCaptureVolume

マイクの集音音量設定。



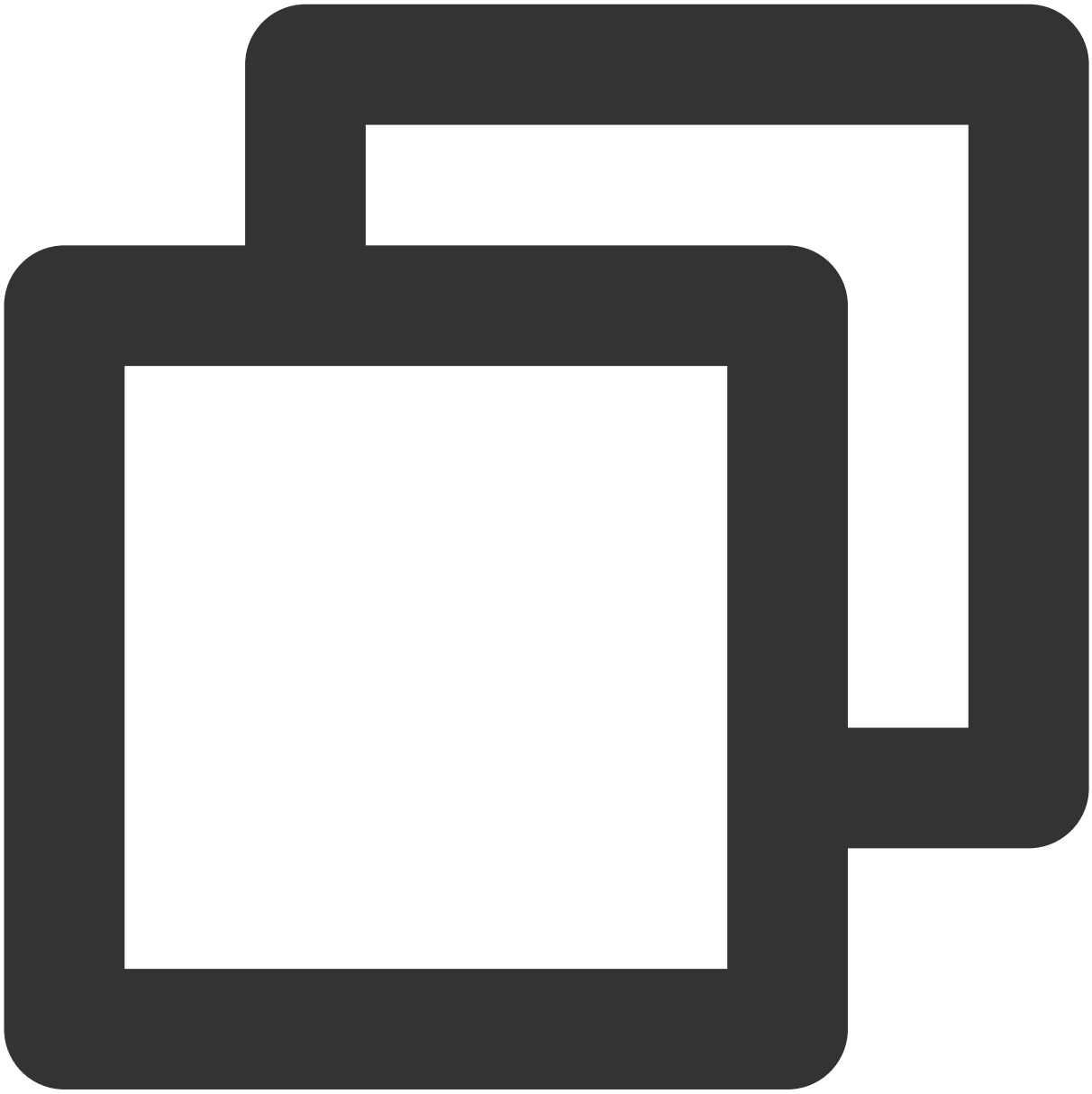
```
- (void)setAudioCaptureVolume:(NSInteger)volume;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
volume	int	集音音量、0〜100、 デフォルト100。

startFileDumping

録音の開始。



```
- (void)startFileDumping:(TRTCAudioRecordingParams *)params;
```

パラメータは下表に示すとおりです。

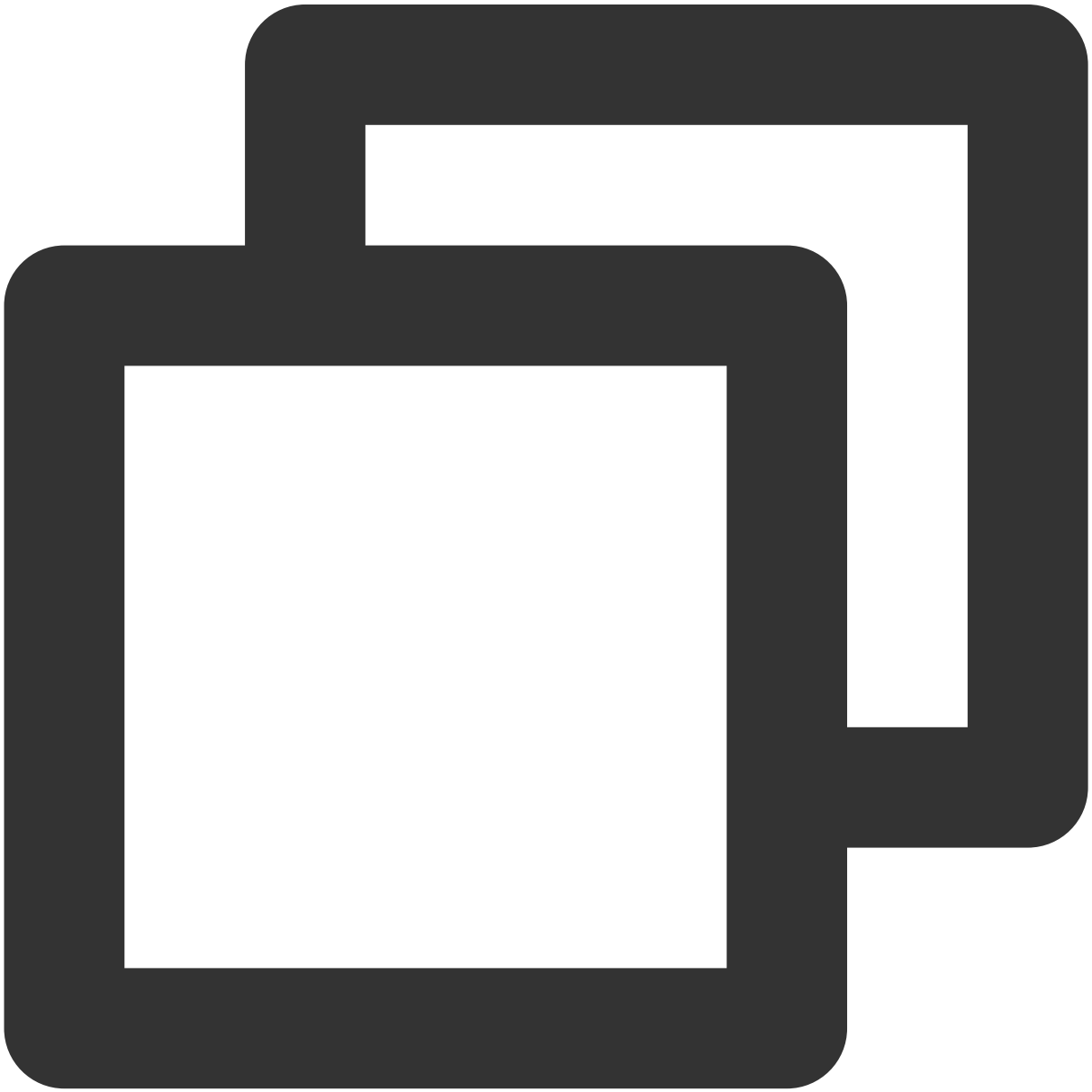
パラメータ	タイプ	意味
params	TRTCAudioRecordingParams	録音パラメータ。詳細については、TRTC SDKをご参照くだ

		さい。

説明：
この方法で呼び出した後、 SDKは通話プロセスの中のすべての音声（ローカル音声、リモート音声、BGMなど）を1つのファイルにレコーディングします。ルームに参加しているか否かにかかわらず、このインターフェースを呼び出せば有効となります。leaveRoomを呼び出した時に録音中であれば、録音は自動的に停止します。

stopFileDumping

録音の停止。

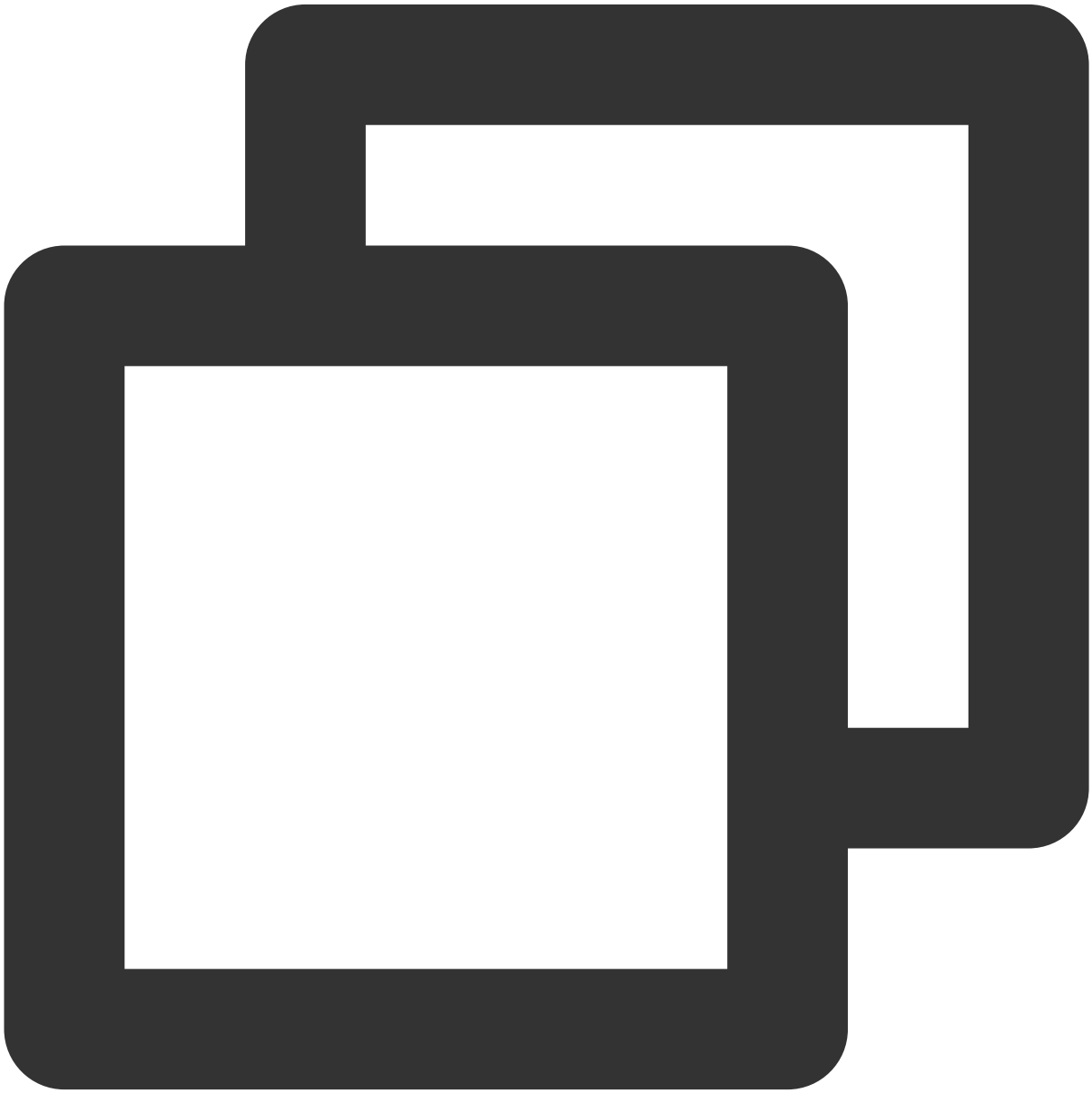


```
- (void)stopFileDumping;
```

SDKバージョンインターフェースの取得

getSdkVersion

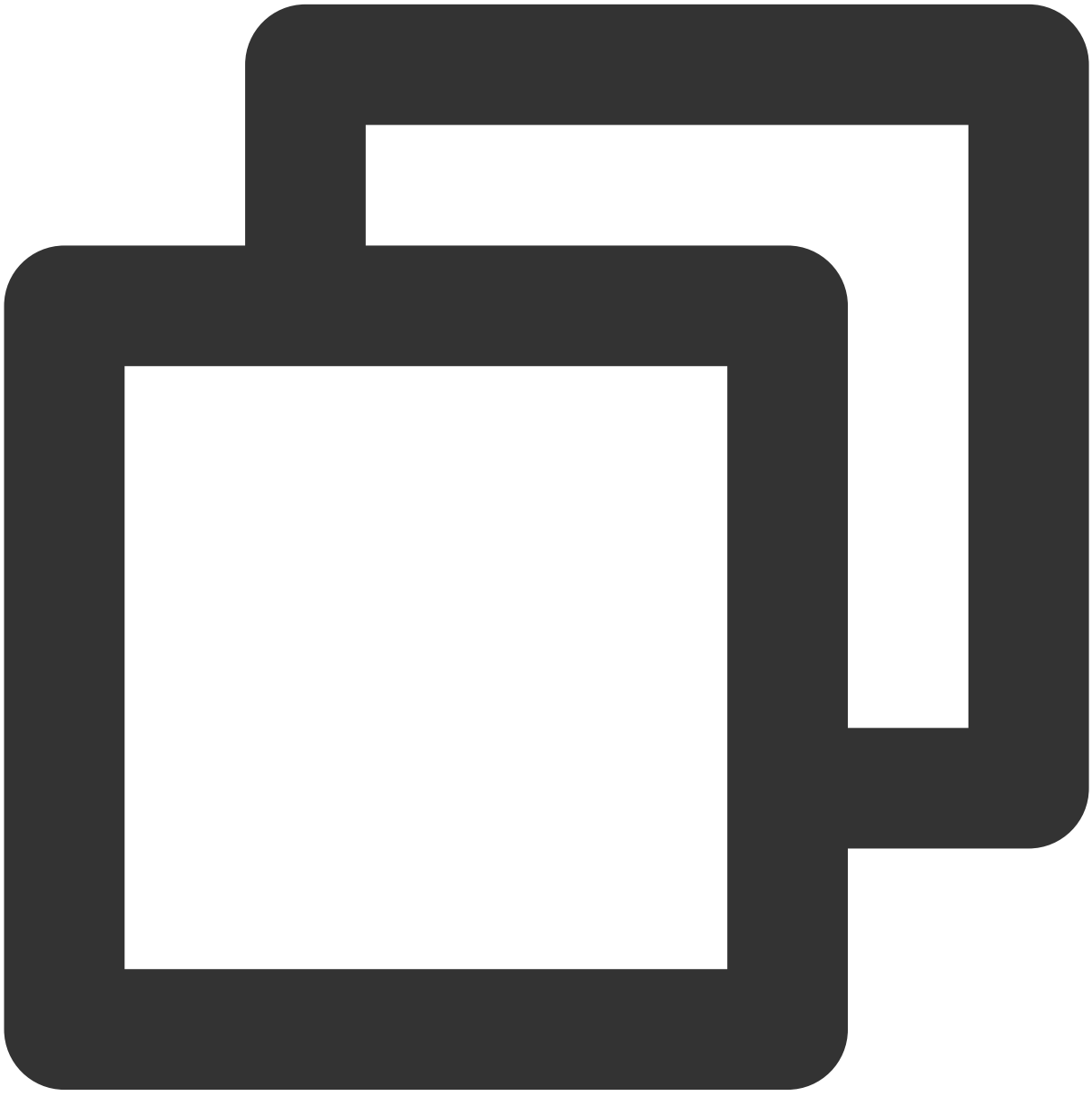
SDKバージョン情報を取得します。



```
- (NSInteger)getSdkVersion;
```

エラーイベントコールバック

onError



```
- (void)onError:(NSInteger)code message:(NSString *)message;
```

パラメータは下表に示すとおりです。

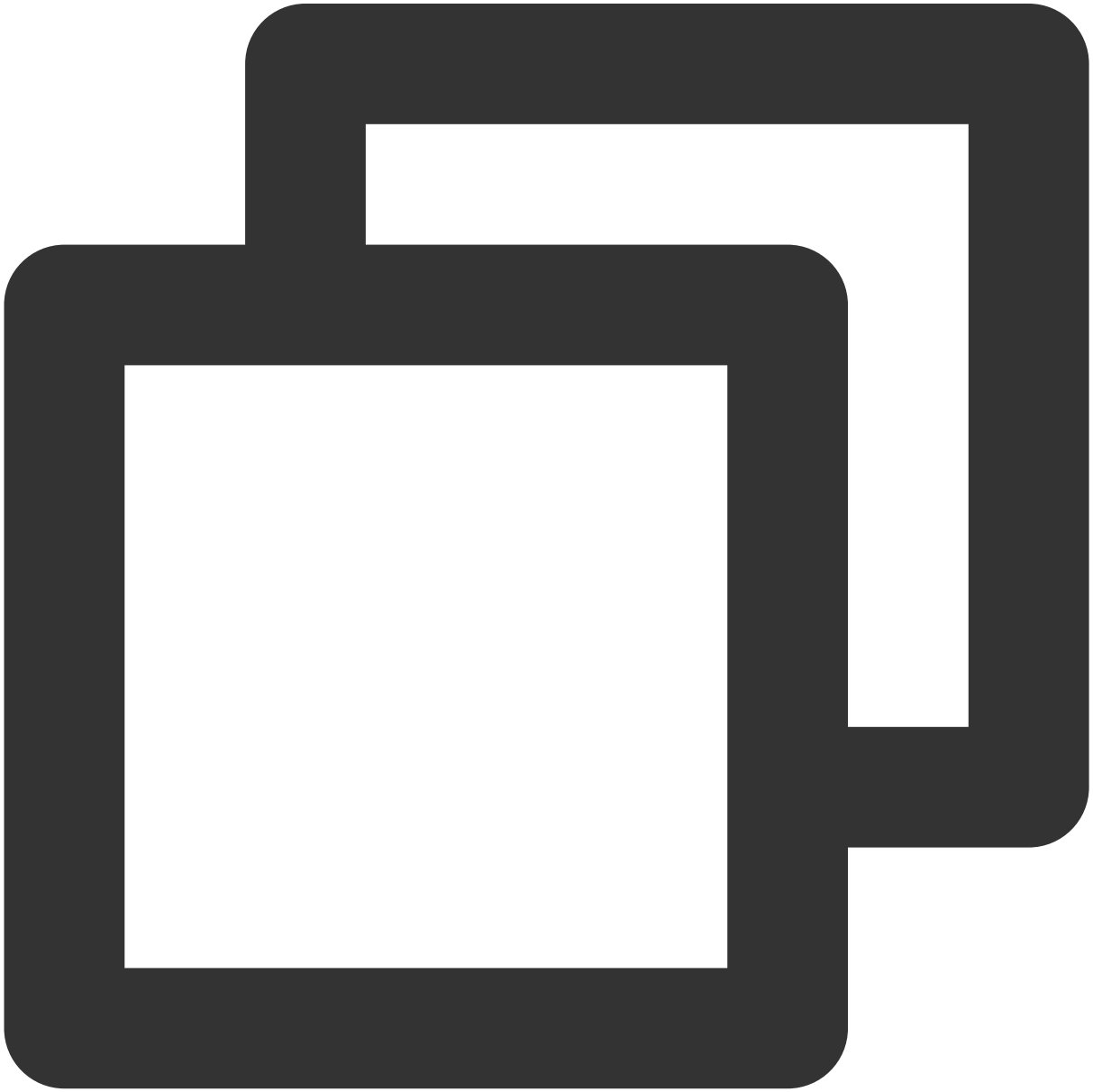
パラメータ	タイプ	意味

code	NSInteger	エラーコード。
message	NSString	エラー情報。

基本イベントコールバック

onDestroyRoom

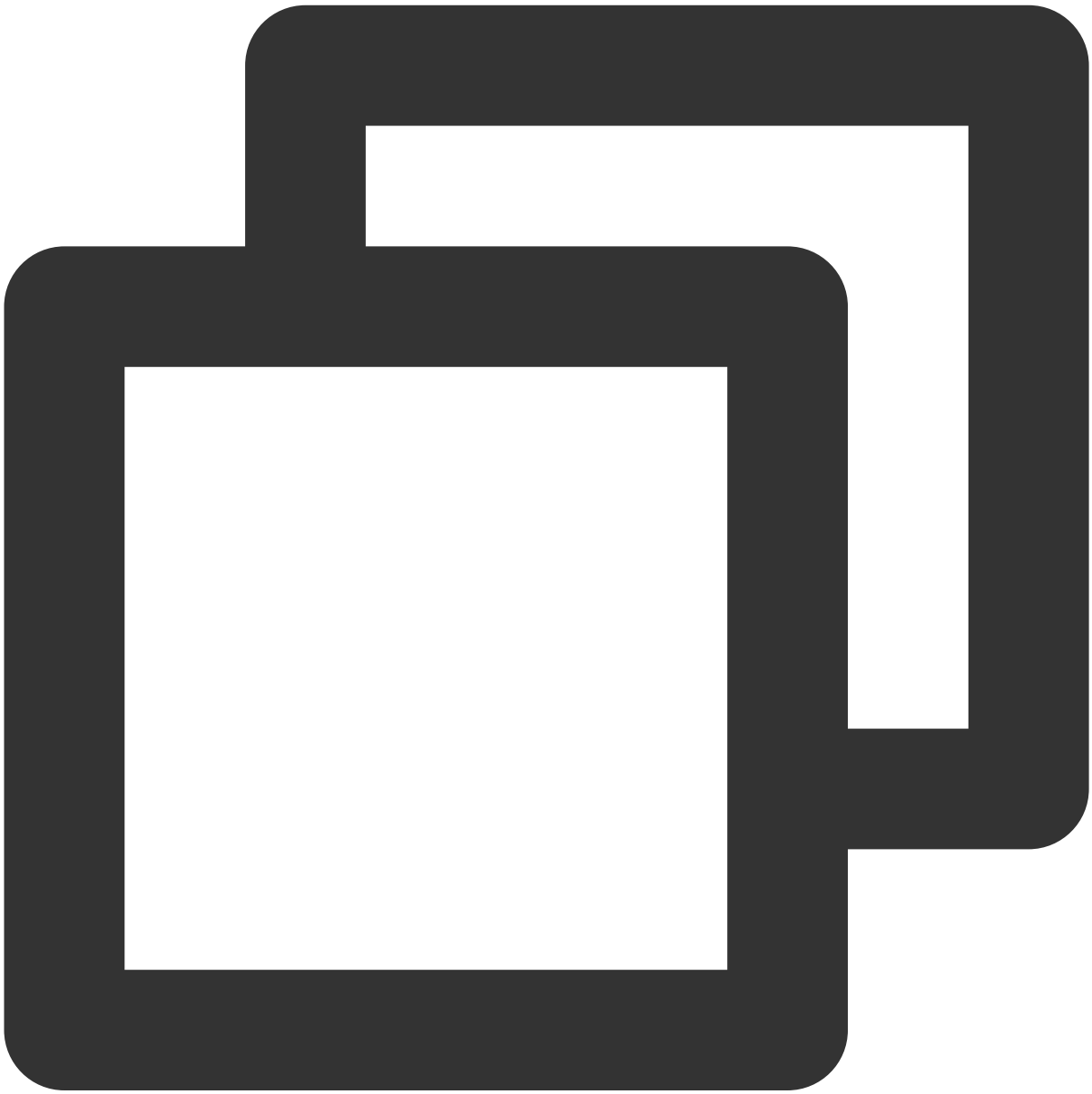
ルーム解散のコールバックです。



```
- (void)onDestroyRoom;
```

onUserVoiceVolume

ユーザー音量の大きさのコールバック。



```
- (void)onUserVoiceVolume:(NSString *)userId volume:(NSInteger)volume;
```

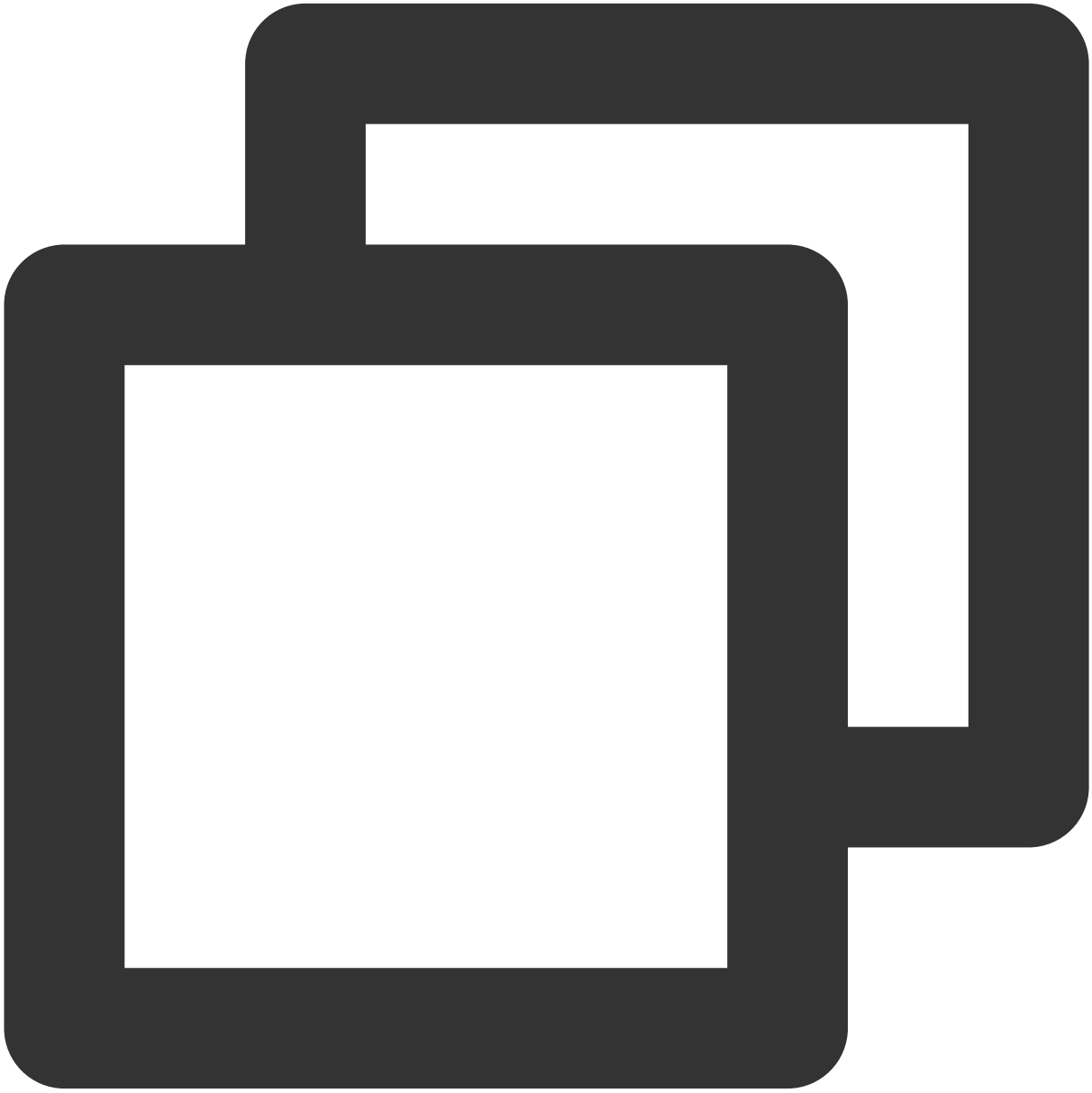
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	NSString	ユーザーID。
volume	NSInteger	ユーザーの音量の大きさ。値の範囲は0～100。

onRoomMasterChanged

キャスター変更のコールバック。



```
- (void)onRoomMasterChanged:(NSString *)previousUserId
                currentUserId:(NSString *)currentUserId;
```

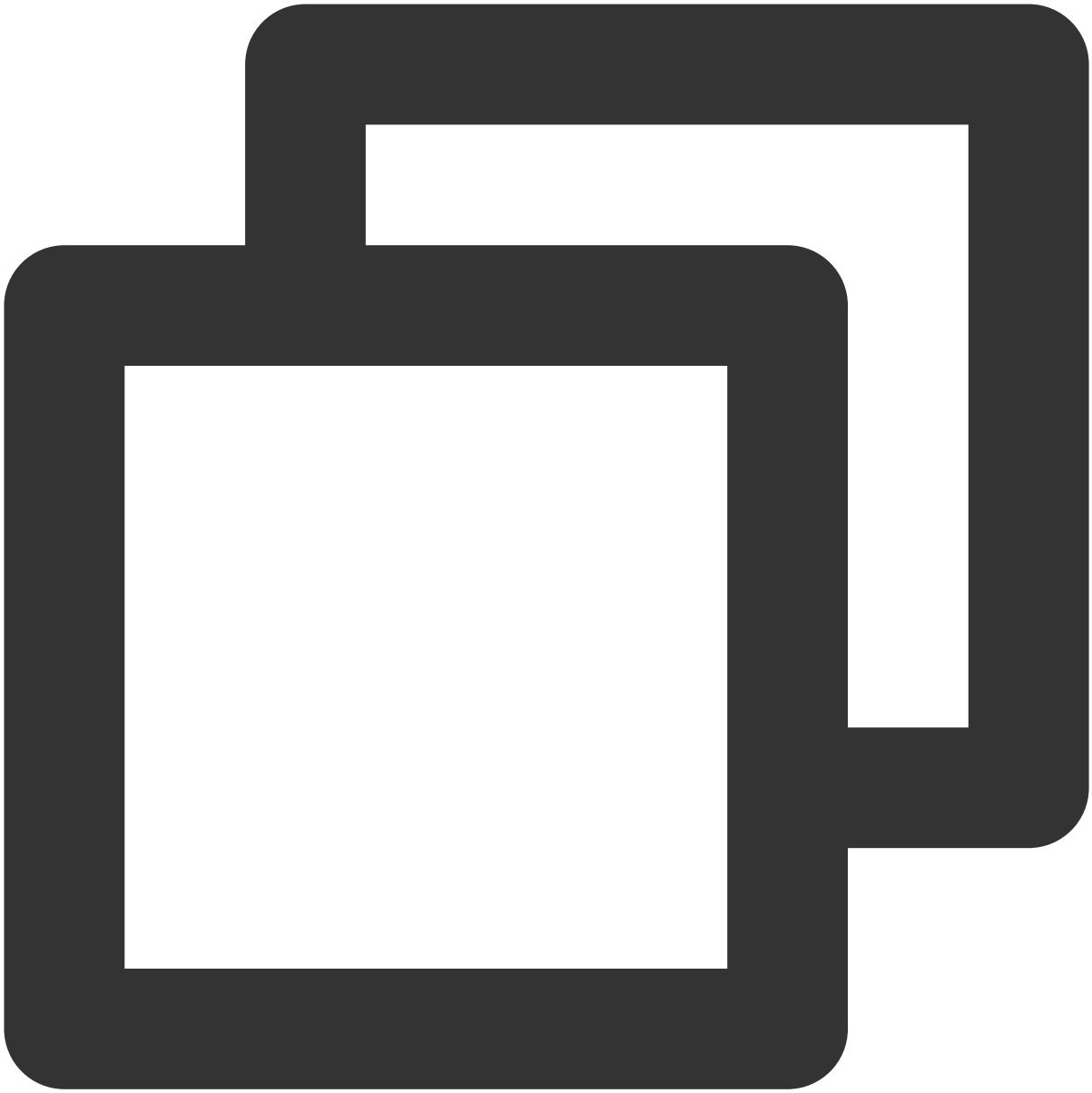
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
previousUserId	NSString	変更前のキャスターユーザーID。
currentUserId	NSString	変更後のキャスターユーザーID。

リモートユーザーコールバックイベント

onRemoteUserEnter

リモートユーザー入室コールバック。



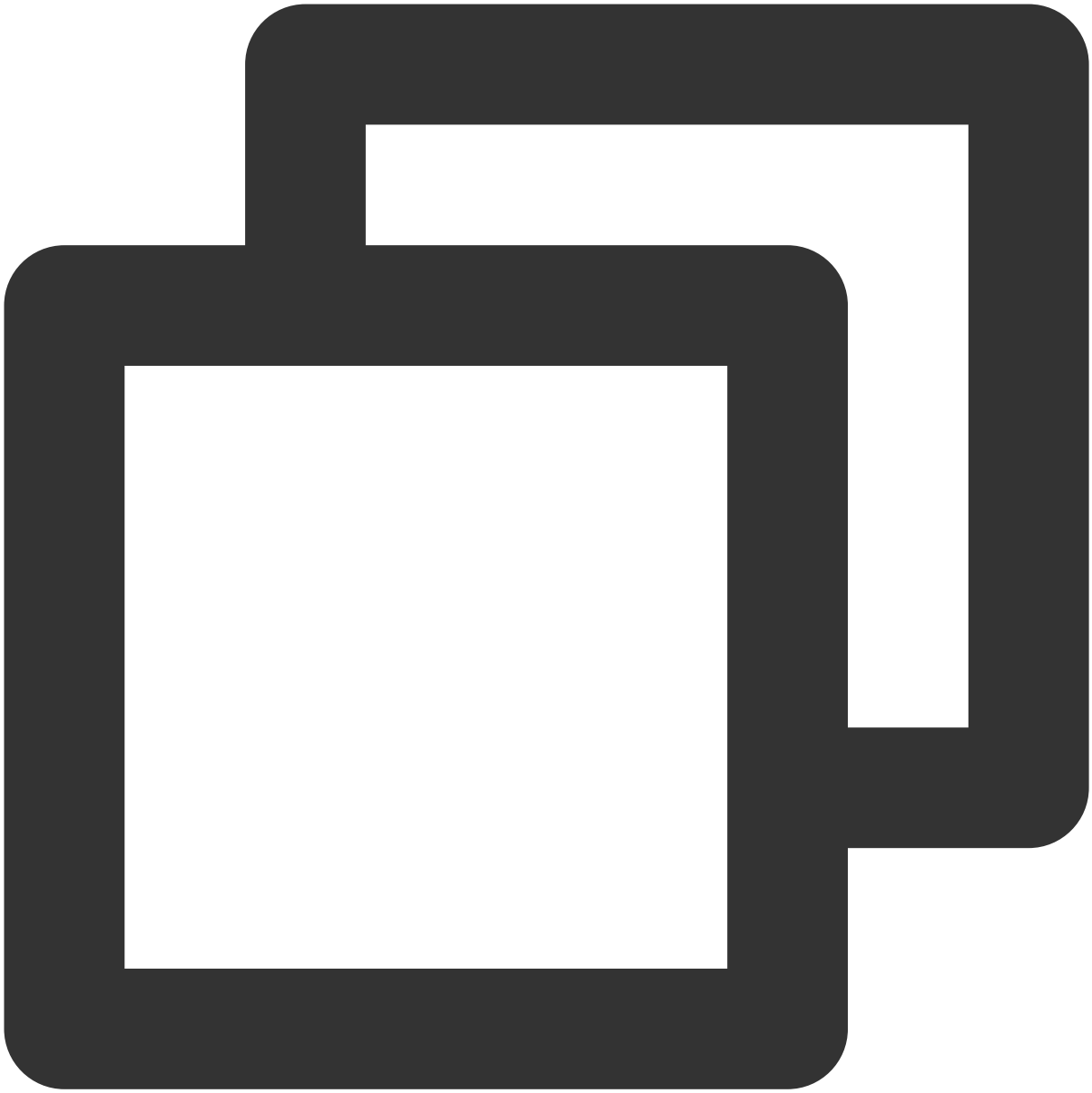
```
- (void)onRemoteUserEnter:(NSString *)userId;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。

onRemoteUserLeave

リモートユーザー退室コールバック。



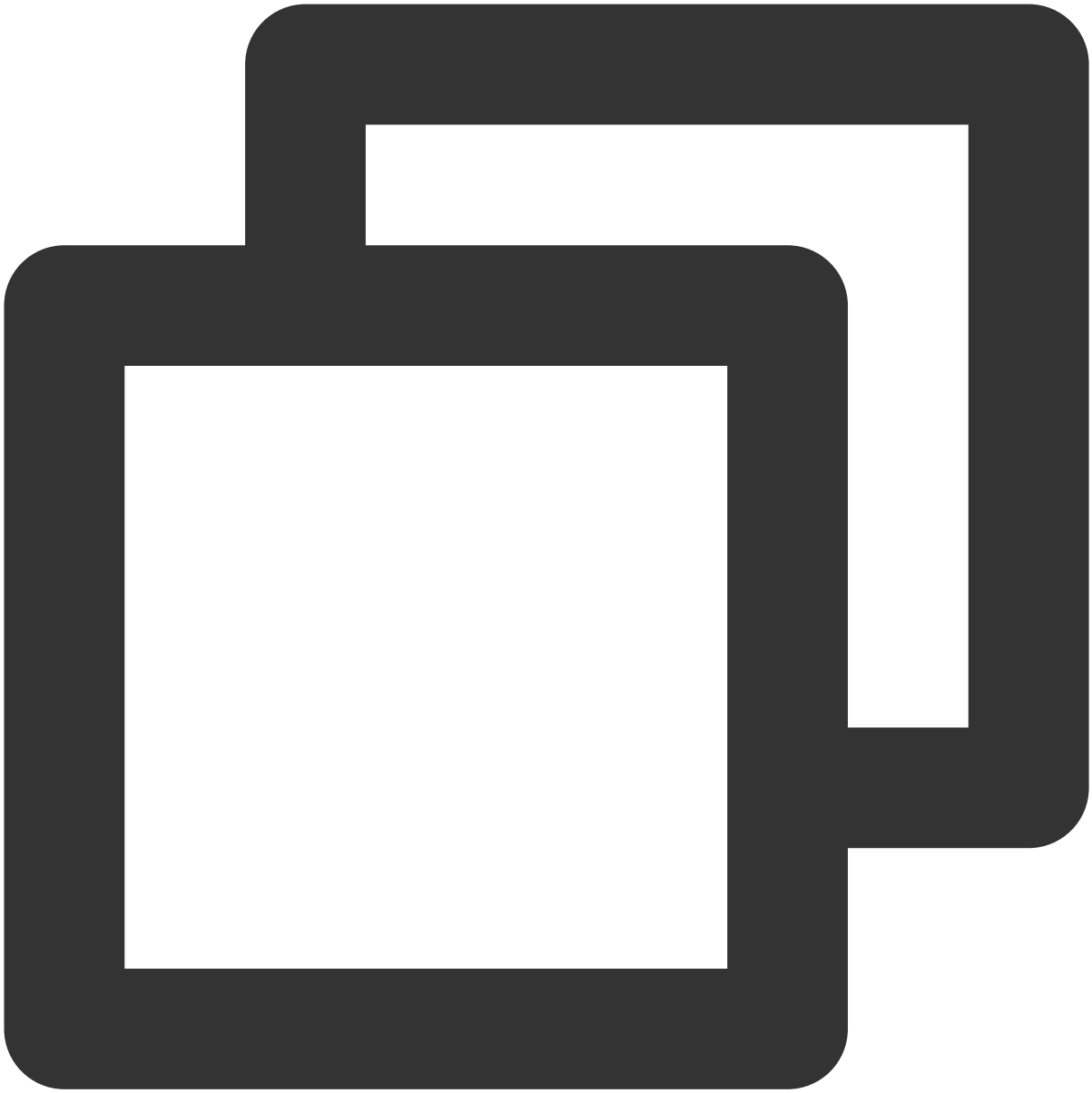
```
- (void)onRemoteUserLeave:(NSString *)userId;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。

onRemoteUserCameraAvailable

リモートユーザーが、カメラ、ビデオを起動しているかどうか。



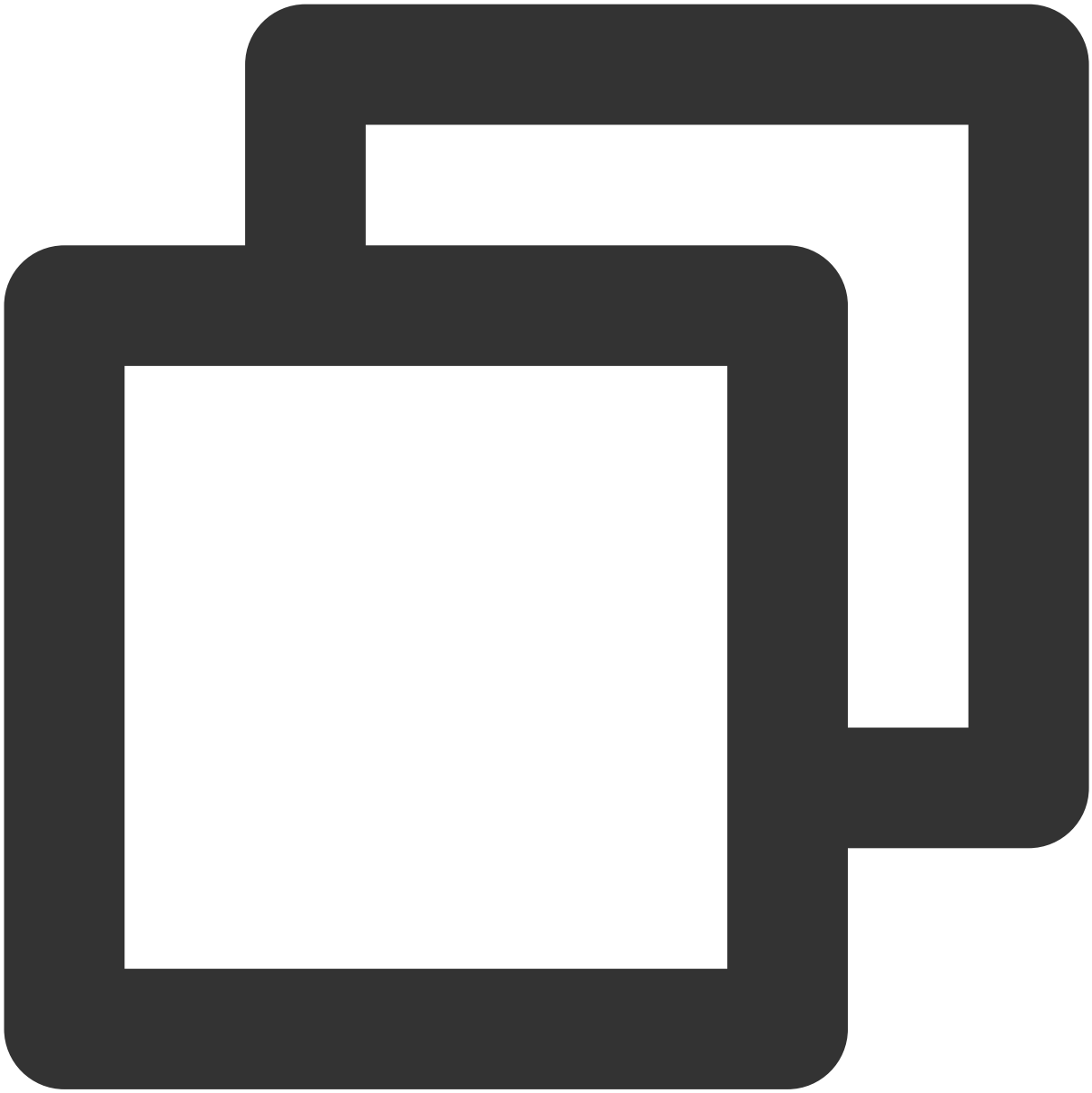
```
- (void)onRemoteUserCameraAvailable:(NSString *)userId
                          available:(BOOL)available;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。
available	BOOL	YES：ビデオストリームデータあり、NO：ビデオストリームデータなし。

onRemoteUserScreenVideoAvailable

メンバーのビデオ共有**オン/オフ**の通知。



```
- (void)onRemoteUserScreenVideoAvailable:(NSString *)userId
                                 available:(BOOL)available;
```

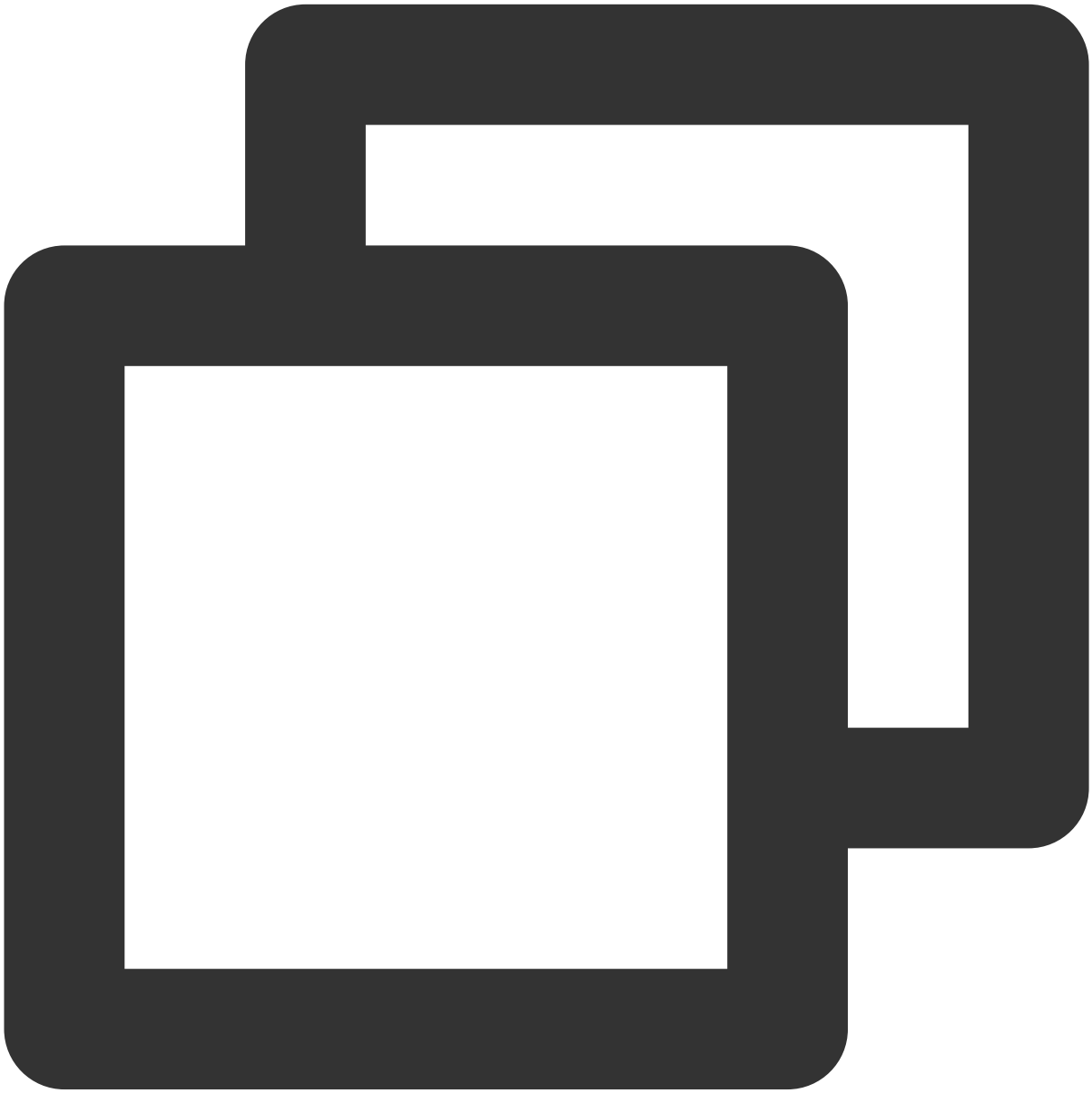
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。

available	BOOL	画面共有ストリームデータの有無。

onRemoteUserAudioAvailable

リモートユーザーがオーディオアップストリームを開始したかどうかのコールバック。



```
- (void)onRemoteUserAudioAvailable:(NSString *)userId
                         available:(BOOL)available;
```

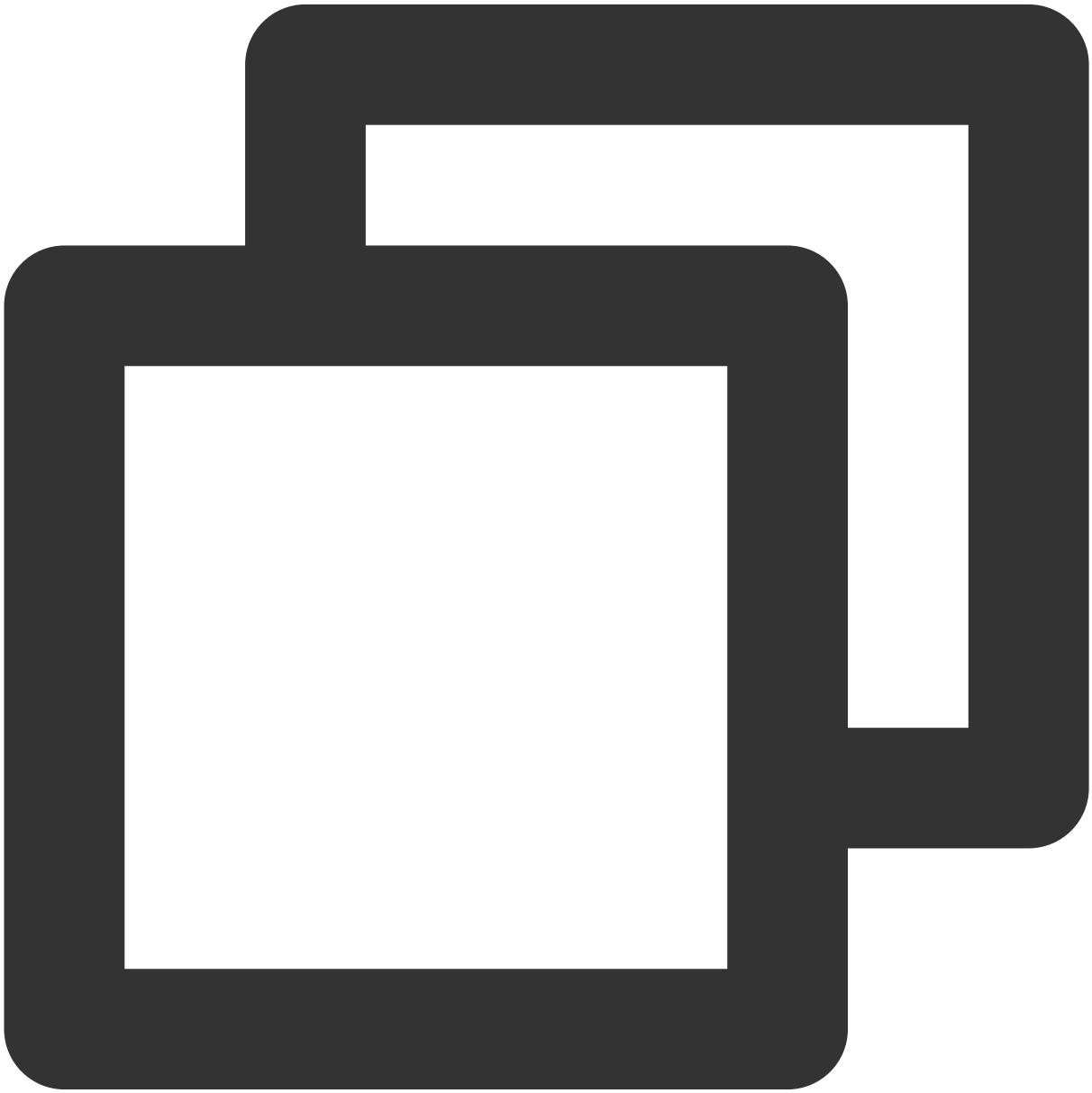
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	NSString	ユーザーID。
available	BOOL	オーディオデータの有無。

onRemoteUserEnterSpeechState

リモートユーザーが発言を開始します。



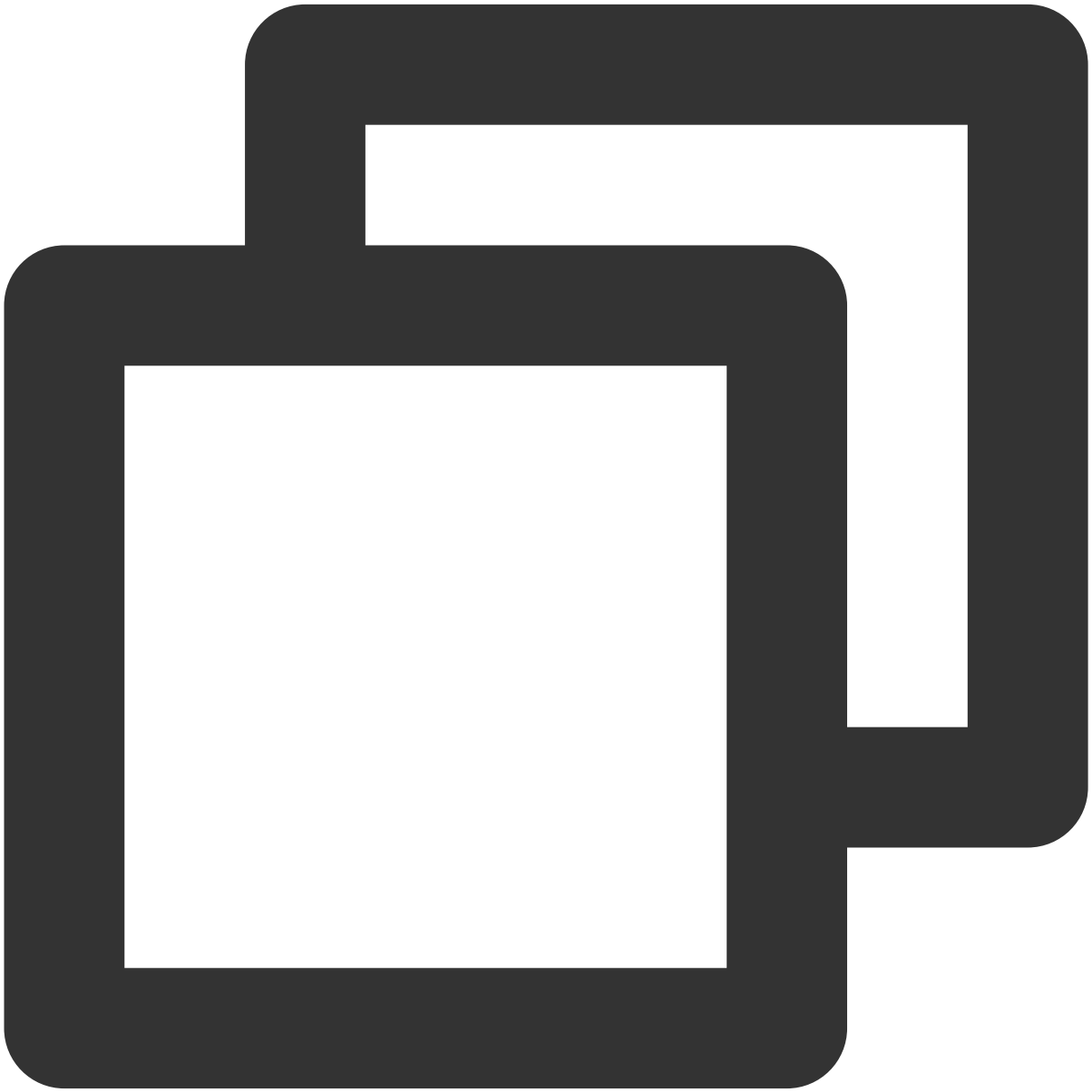
```
- (void)onRemoteUserEnterSpeechState:(NSString *)userId;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。

onRemoteUserExitSpeechState

リモートユーザーが発言を終了します。



```
- (void)onRemoteUserExitSpeechState:(NSString *)userId;
```

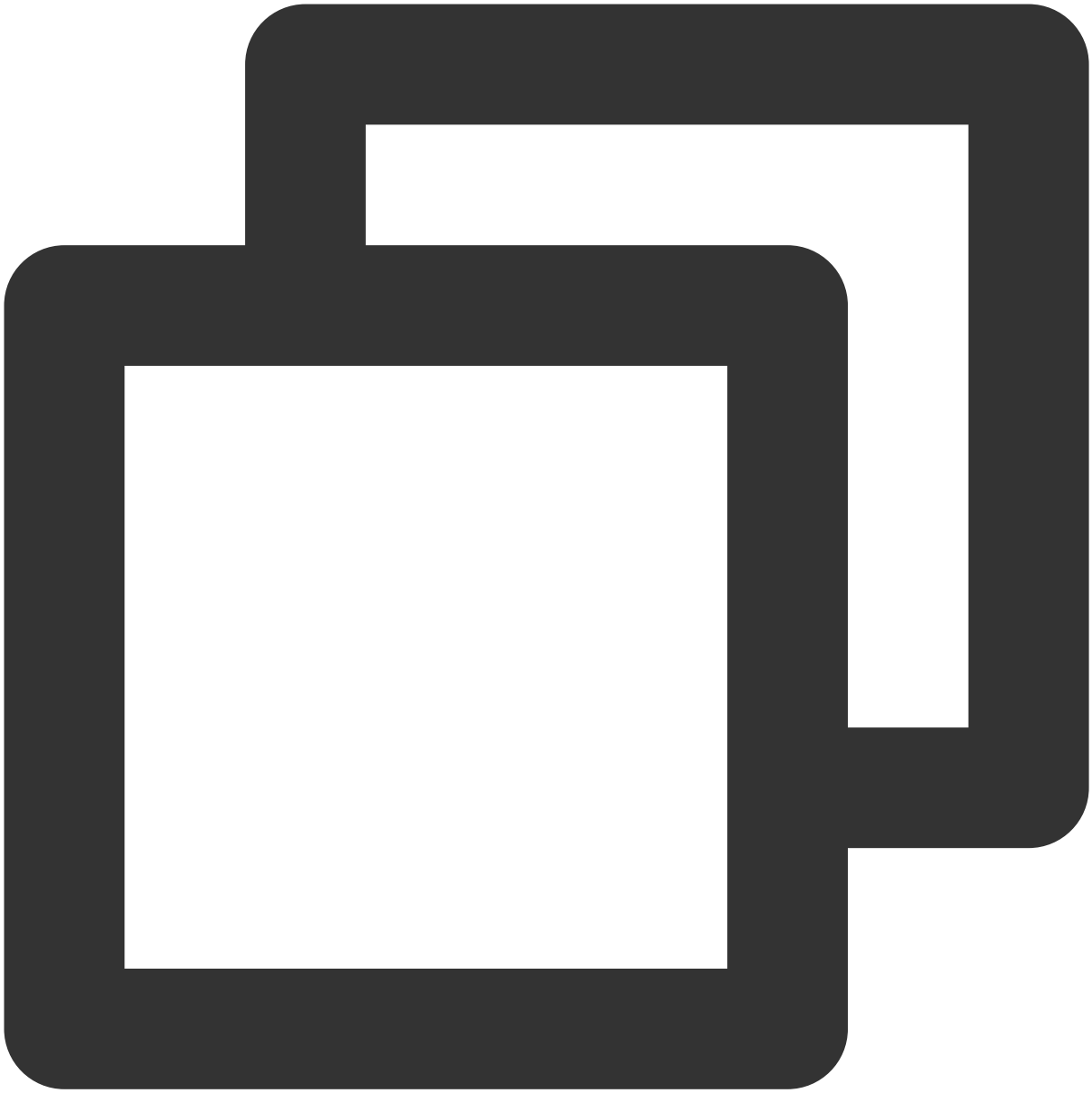
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。

チャットルームメッセージイベントコールバック

onReceiveChatMessage

テキストメッセージの受信。



```
- (void)onReceiveChatMessage:(NSString *)userId message:(NSString *)message;
```

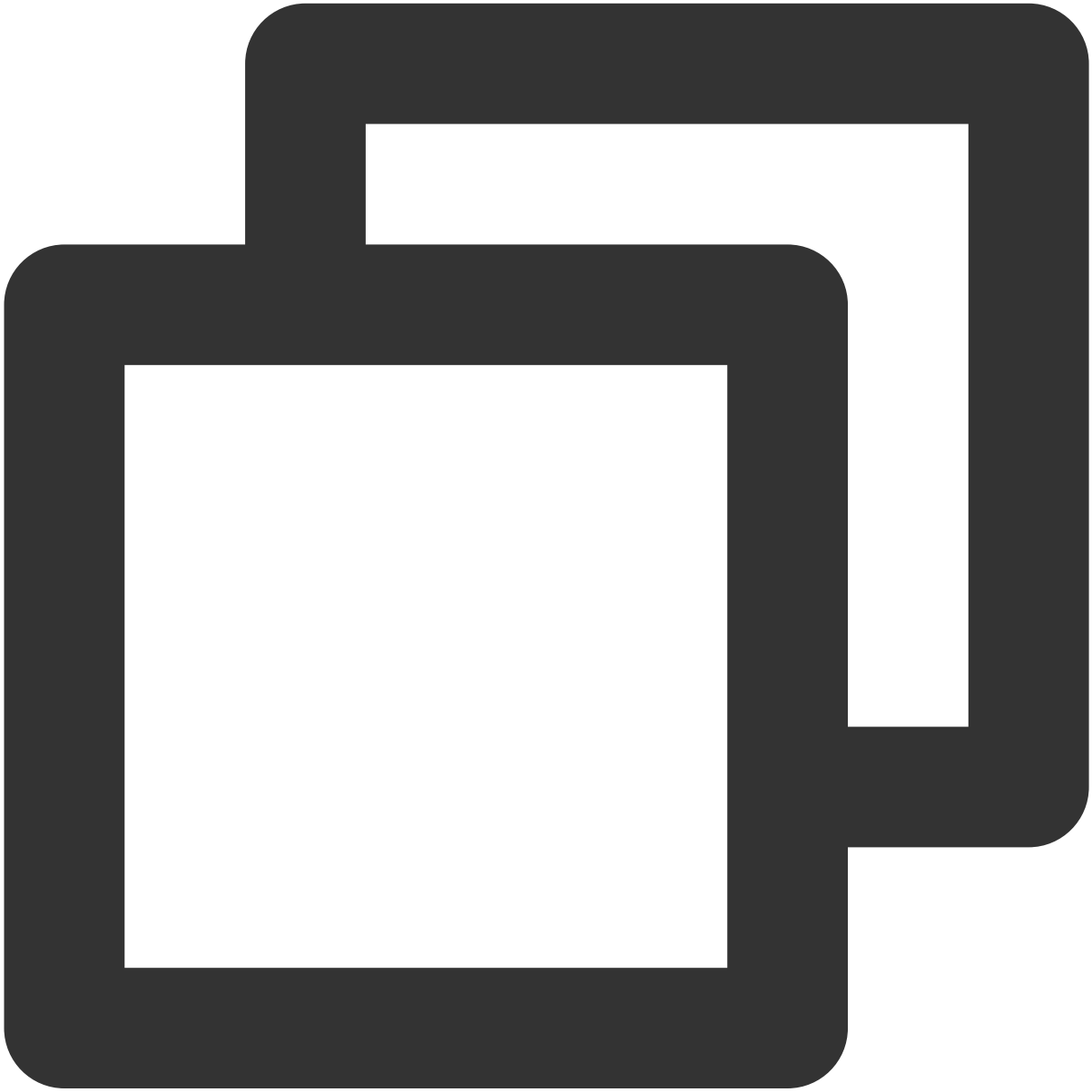
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。
message	NSString	テキストメッセージ。

フィールドコントロールメッセージコールバック

onReceiveSpeechInvitation

ユーザーがキャスターの発言要請を受信する場合のコールバック。



```
- (void)onReceiveSpeechInvitation:(NSString *)userId;
```

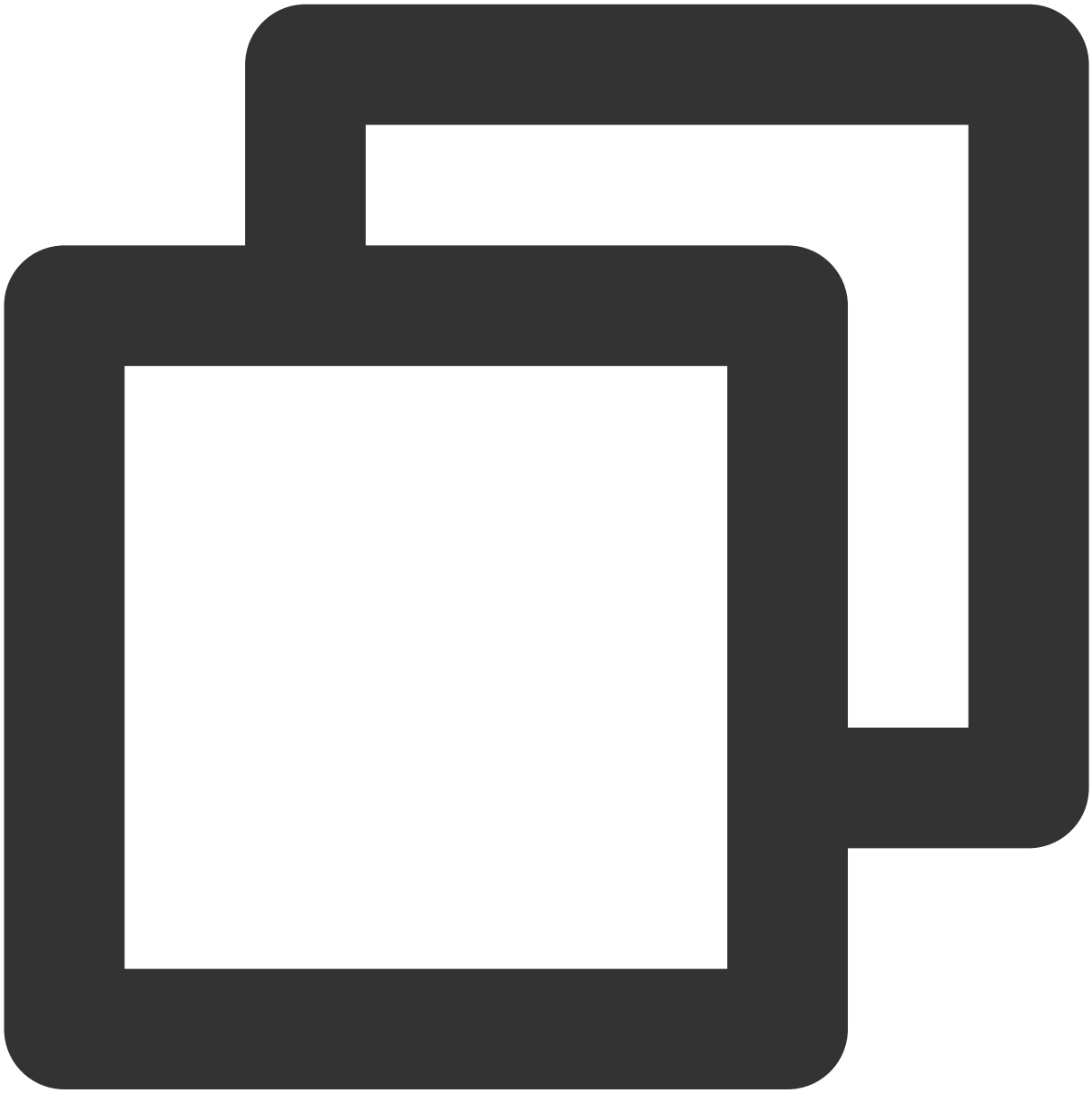
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	NSString	キャスターユーザーID。

onReceiveInvitationCancelled

ユーザーがキャスターの発言要請キャンセルを受信する場合のコールバック。



```
- (void)onReceiveInvitationCancelled:(NSString *)userId;
```

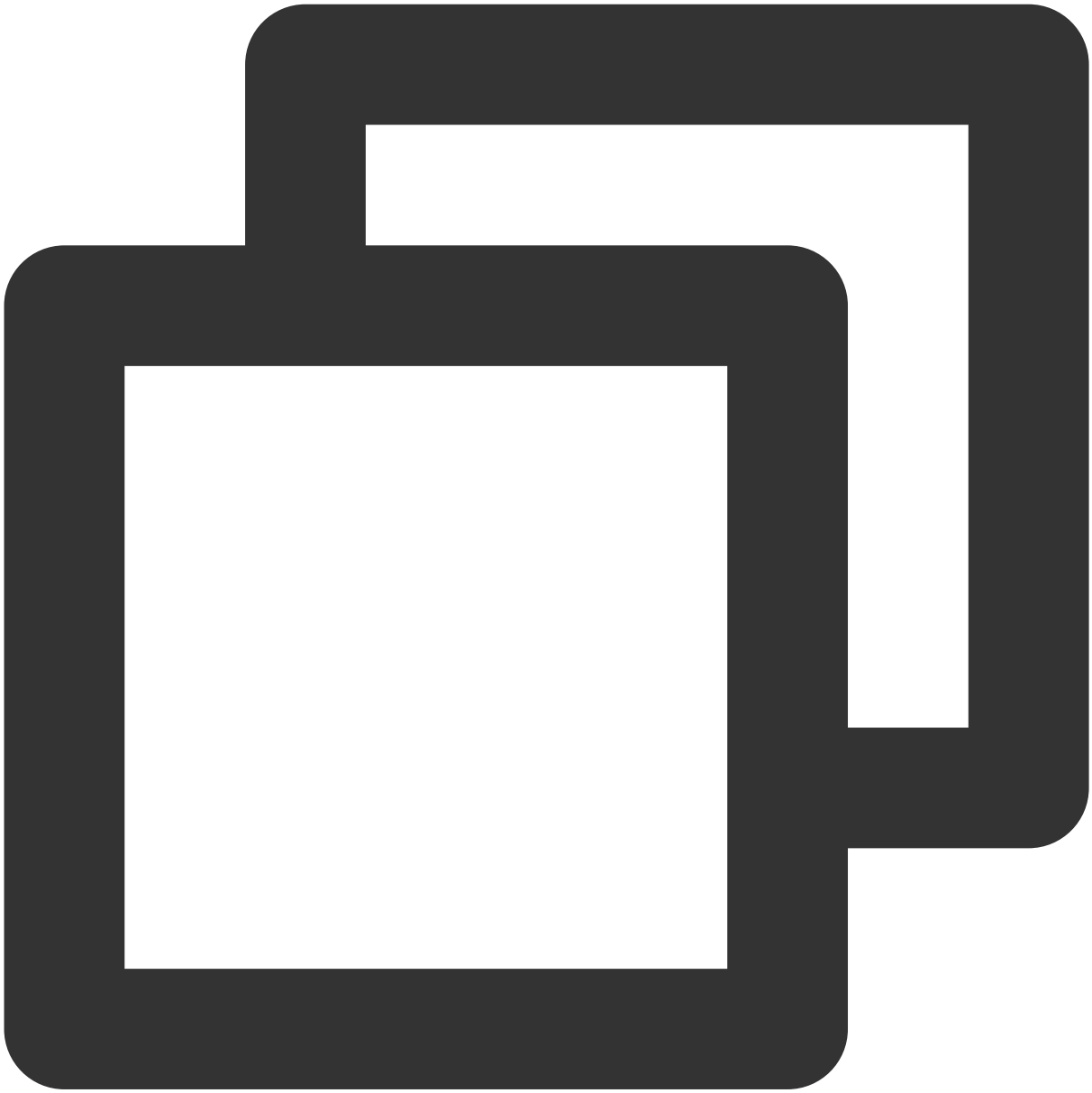
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	NSString	キャスターユーザーID。

OnReceiveSpeechApplication

キャスターがユーザーの発言申請を受信する場合のコールバック。



```
void onReceiveSpeechApplication(String userId);
```

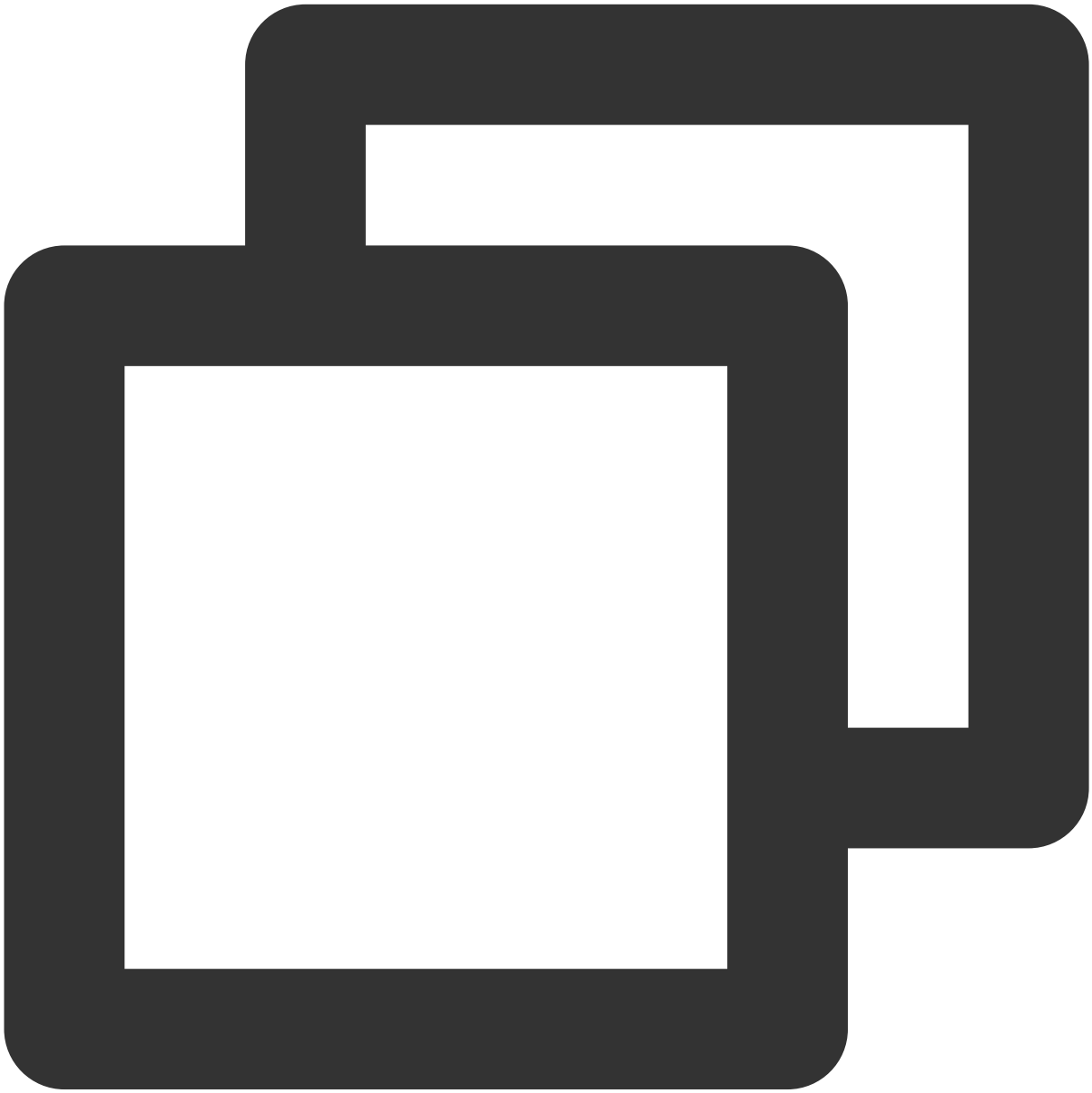
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	NSString	ユーザーID。

onSpeechApplicationCancelled

ユーザーが発言申請をキャンセルする場合のコールバック。



```
- (void)onSpeechApplicationCancelled:(NSString *)userId;
```

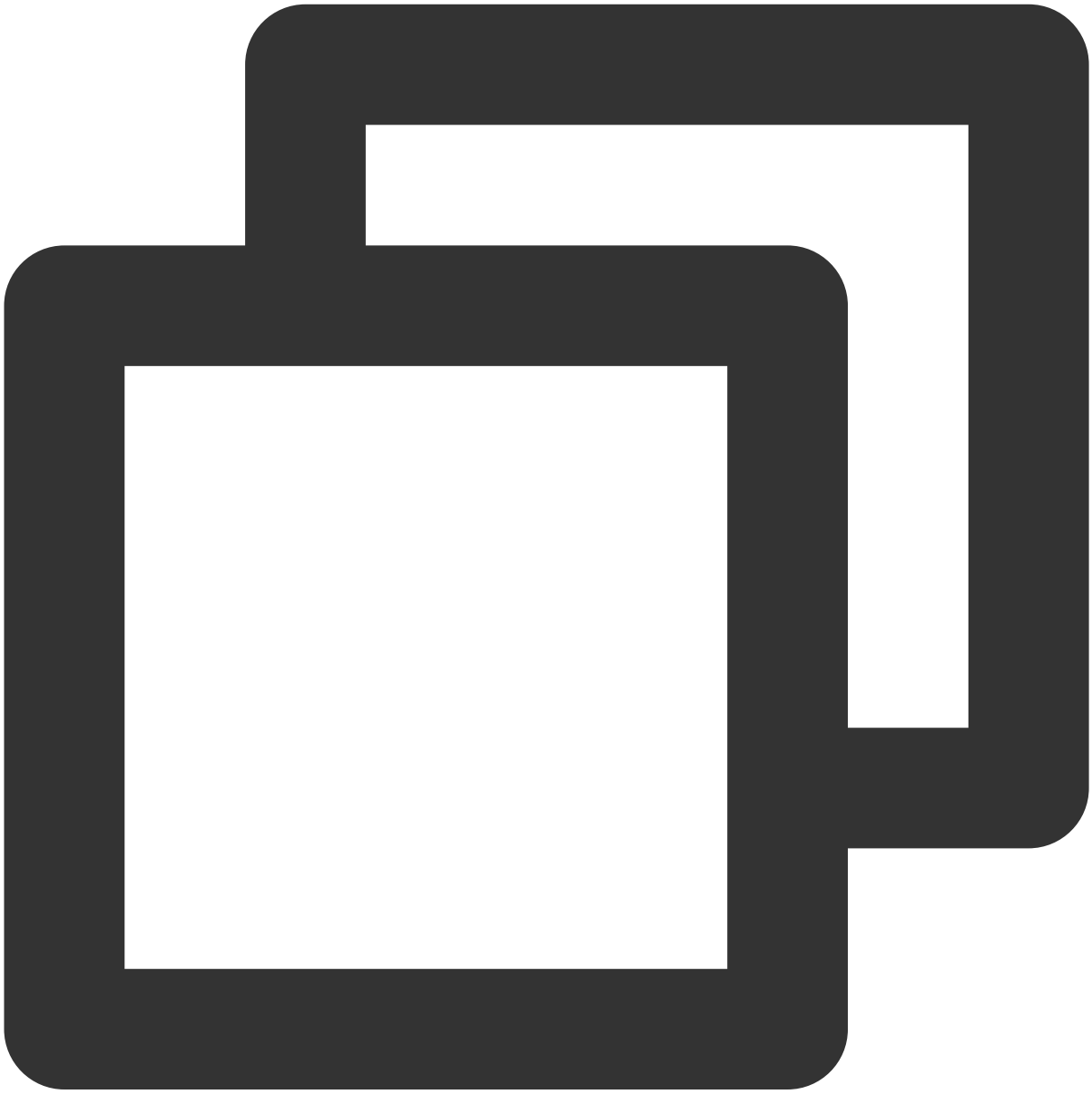
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

userId	NSString	ユーザーID。

onSpeechApplicationForbidden

キャスターが発言申請を禁止する場合のコールバック。



```
- (void)onSpeechApplicationForbidden:(BOOL)isForbidden userId:(NSString *)userId;
```

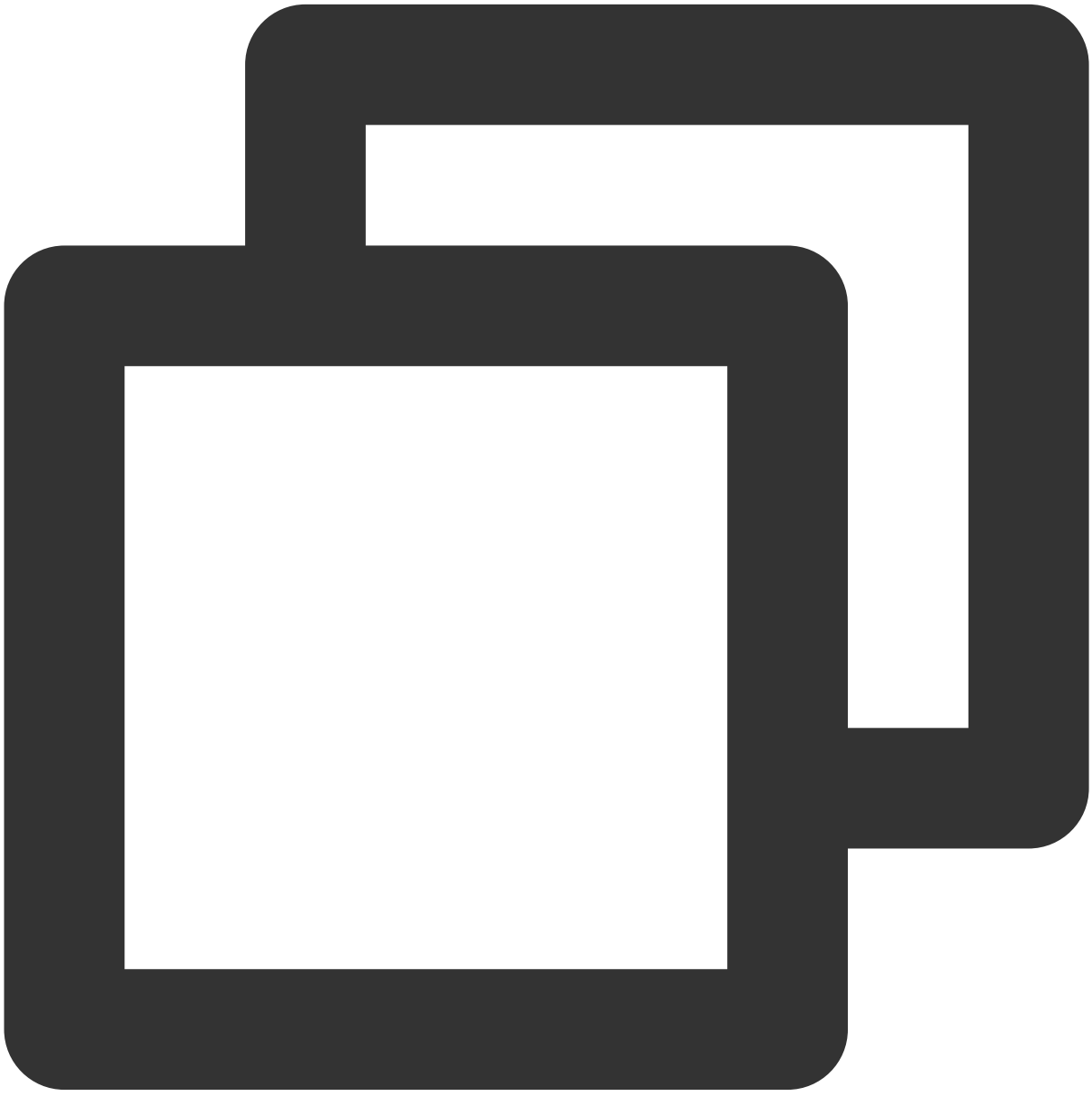
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

isForbidden	BOOL	禁止するかどうか。
userId	NSString	ユーザーID。

onOrderedToExitSpeechState

参加者が発言を停止するようリクエストされる場合のコールバック。



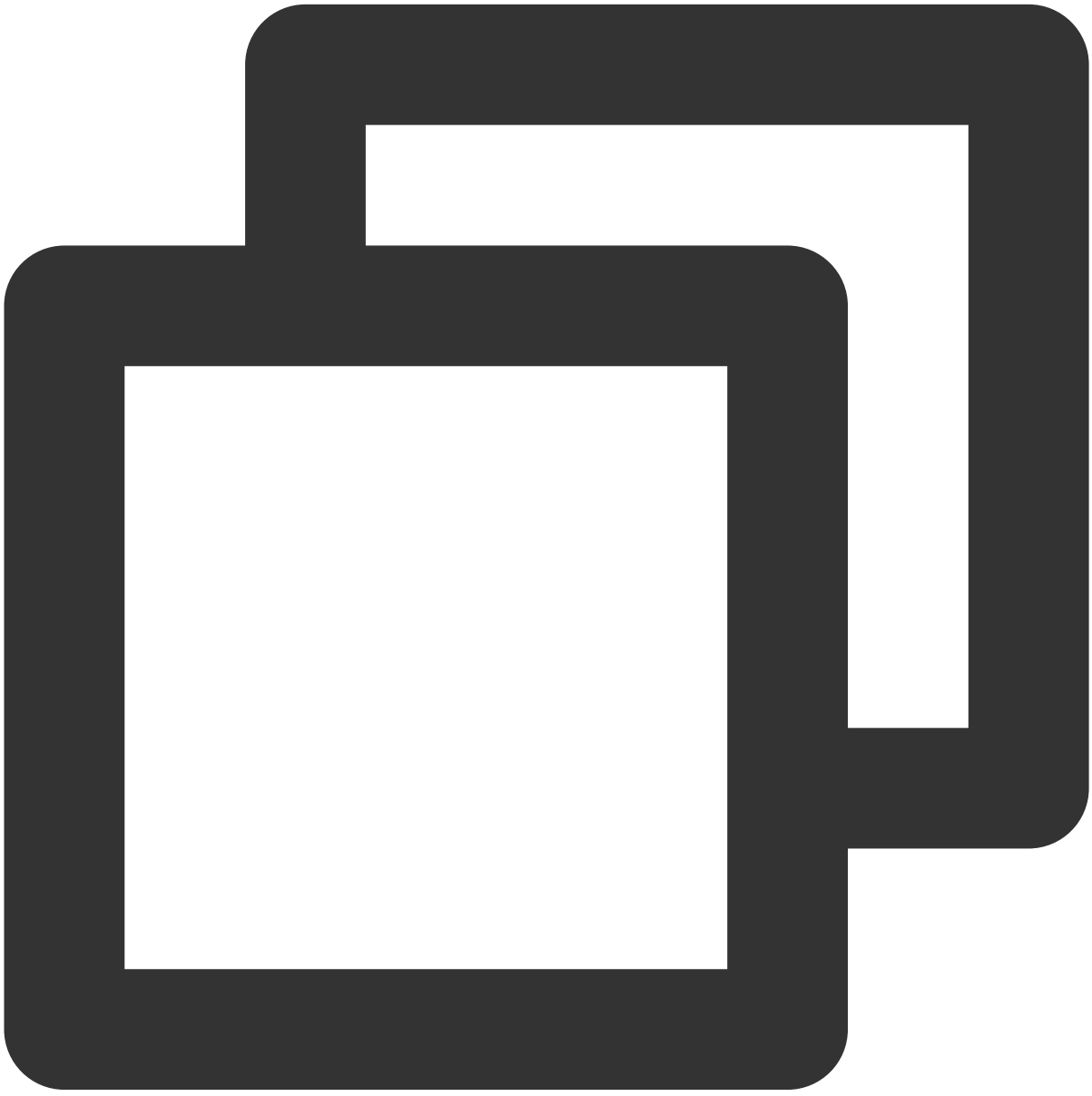
```
- (void)onOrderedToExitSpeechState:(NSString *)userId;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	キャスターユーザーID。

onCallingRollStarted

キャスターが点呼を開始し、参加者が受信する場合のコールバック。



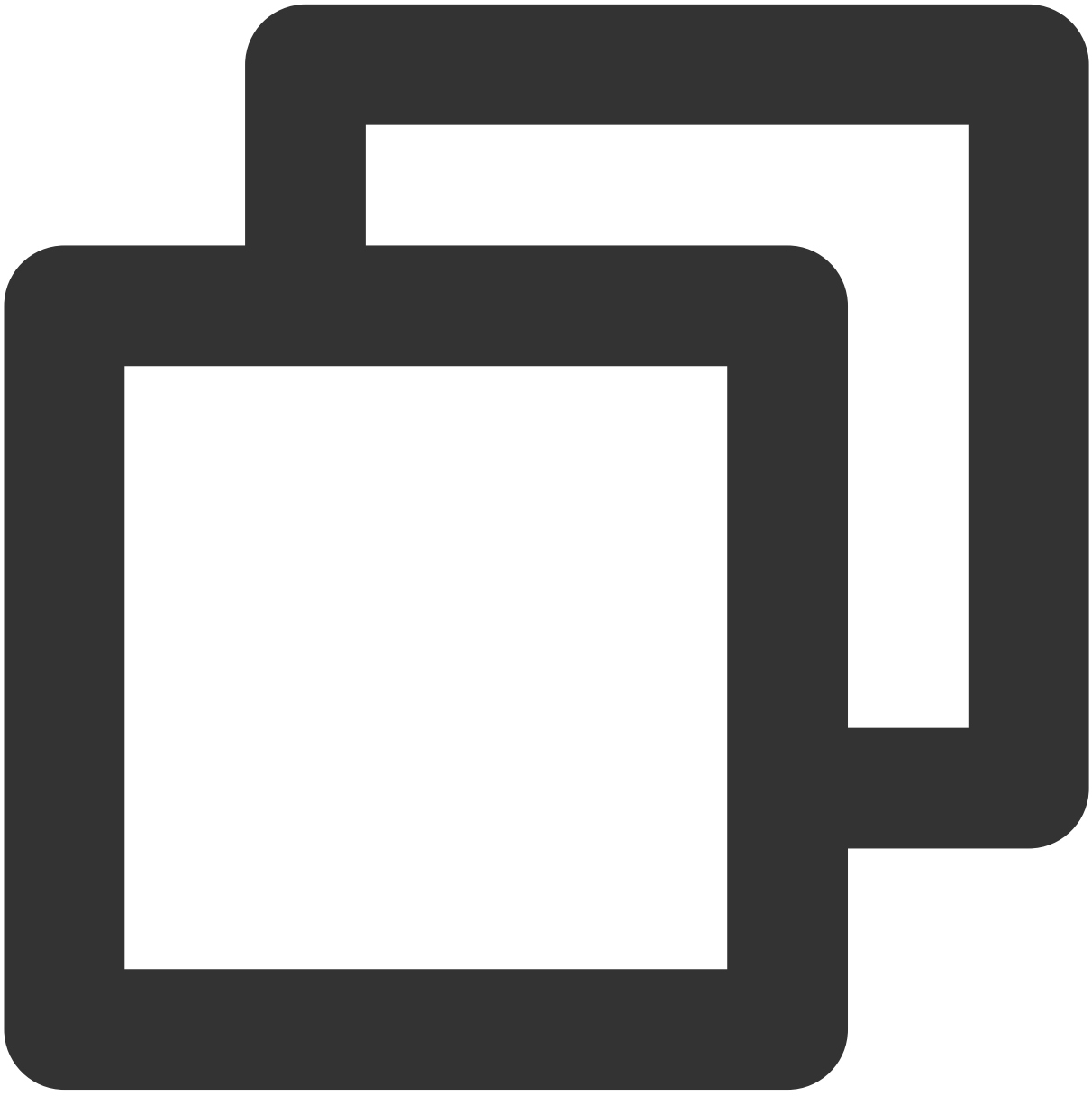
```
- (void)onCallingRollStarted:(NSString *)userId;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	キャスターユーザーID。

onCallingRollStopped

キャスターが点呼を終了し、参加者が受信する場合のコールバック。



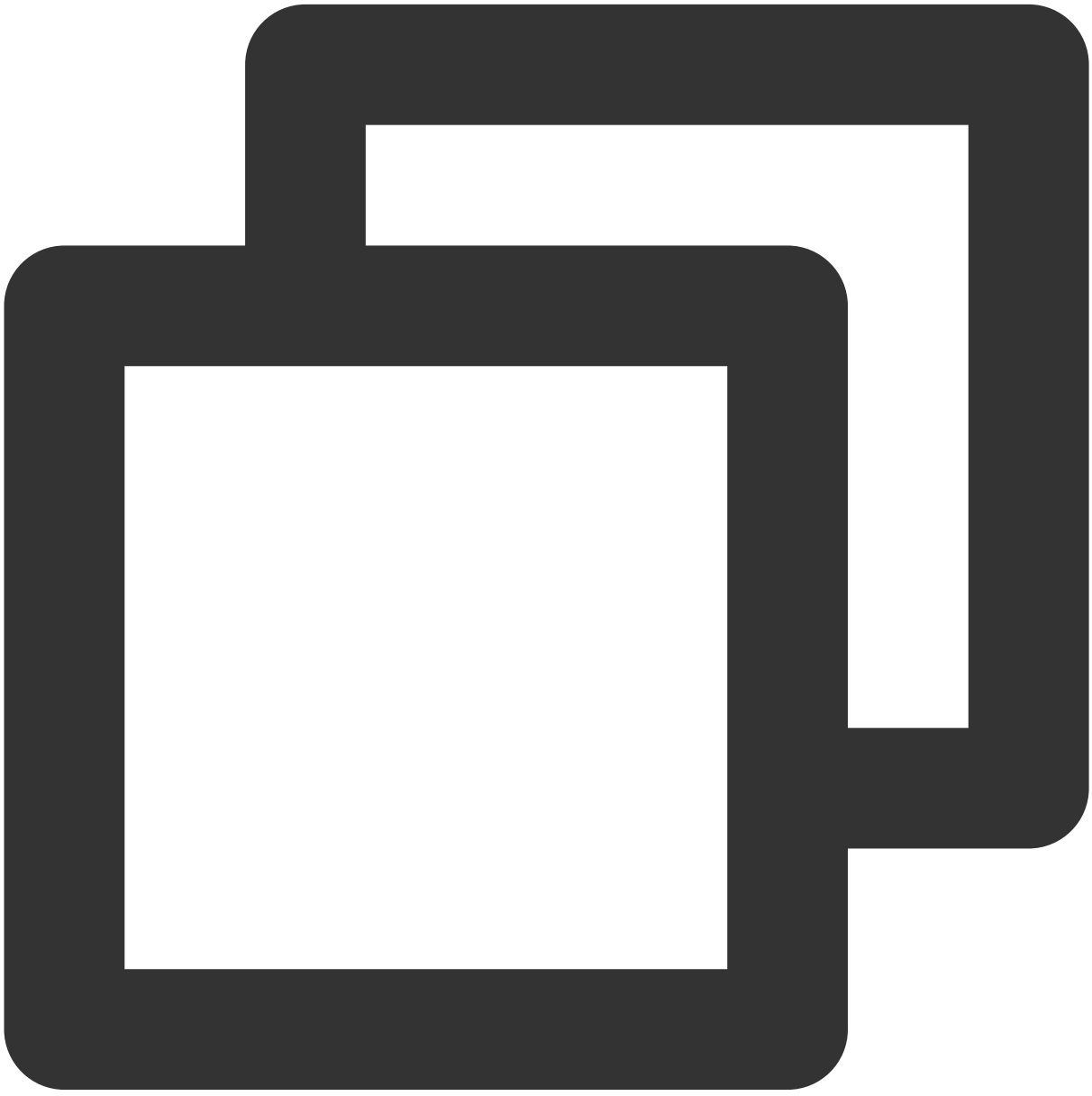
```
- (void)onCallingRollStopped:(NSString *)userId;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	キャスターユーザーID。

onMemberReplyCallingRoll

参加者が点呼に応答し、キャスターが受信する場合のコールバック。



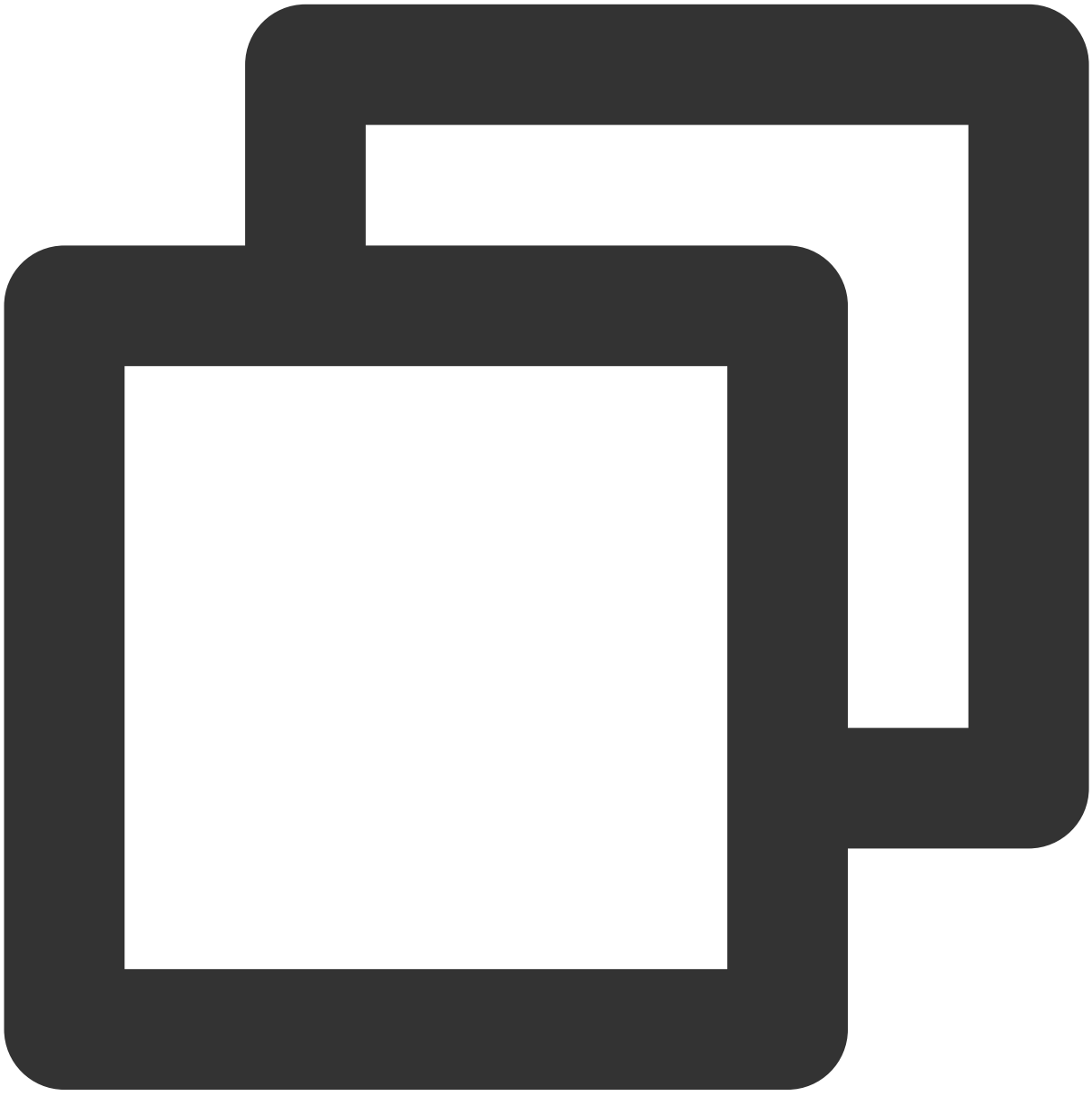
```
- (void)onMemberReplyCallingRoll:(NSString *)userId;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	ユーザーID。

onChatRoomMuted

キャスターがチャットルームのミュートを変更する場合のコールバック。



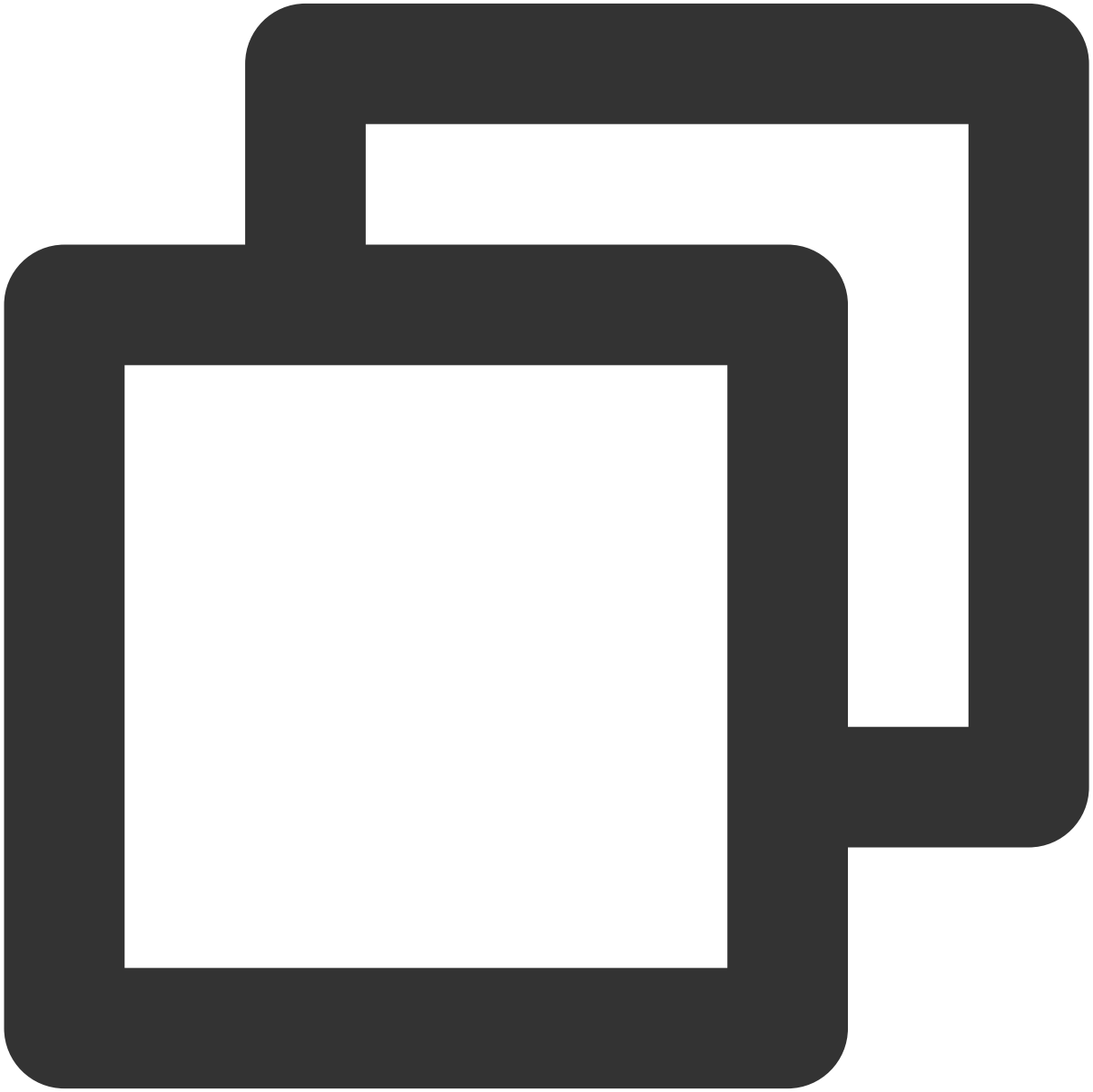
```
- (void)onChatRoomMuted:(BOOL)muted userId:(NSString *)userId;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
muted	BOOL	無効にするかどうか。
userId	NSString	キャスターユーザーID。

onMicrophoneMuted

キャスターがマイクの無効化を設定する場合のコールバック。



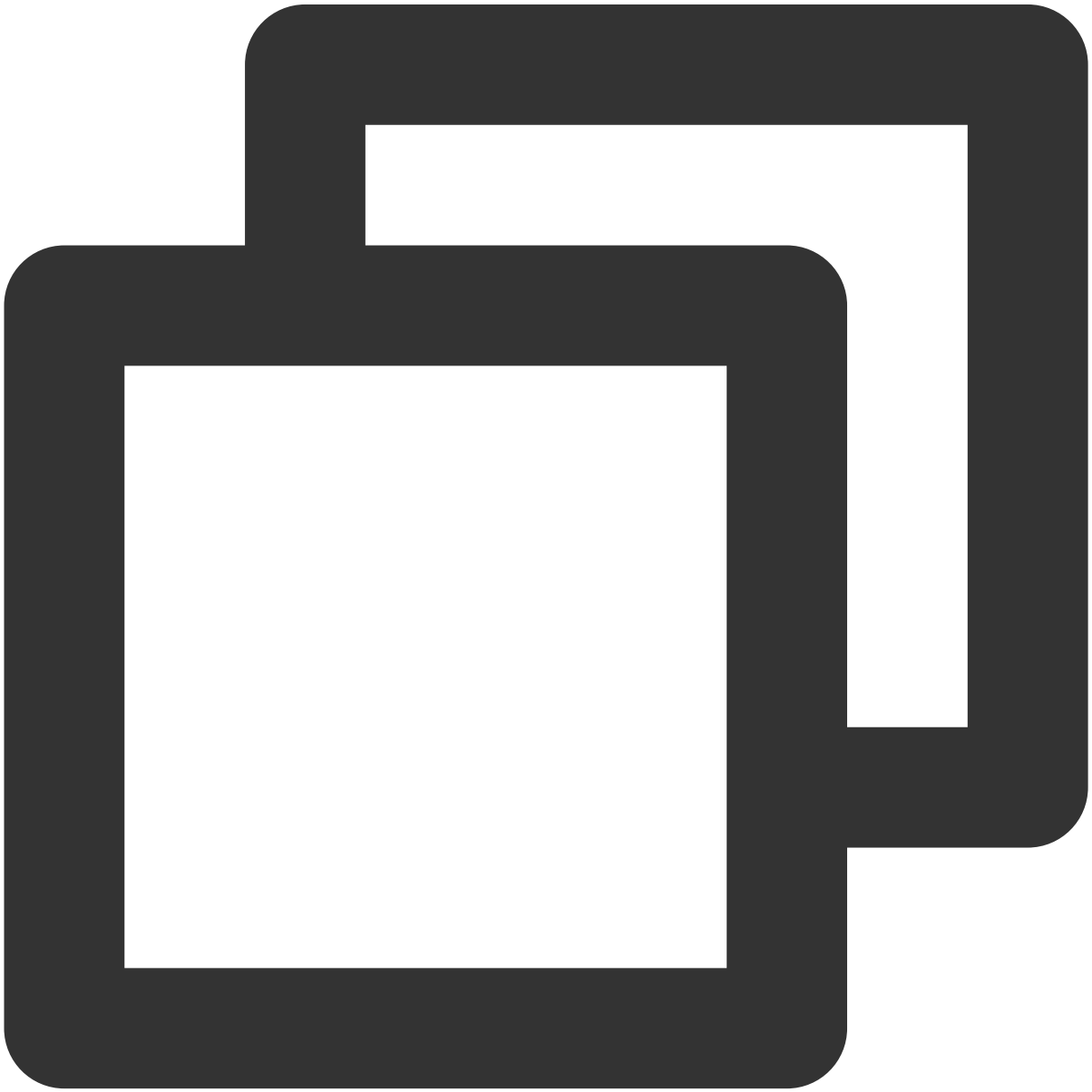
```
- (void)onMicrophoneMuted:(BOOL)muted userId:(NSString *)userId;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
muted	BOOL	無効にするかどうか。
userId	NSString	キャスターユーザーID。

onCameraMuted

キャスターがカメラの無効化を設定する場合のコールバック。



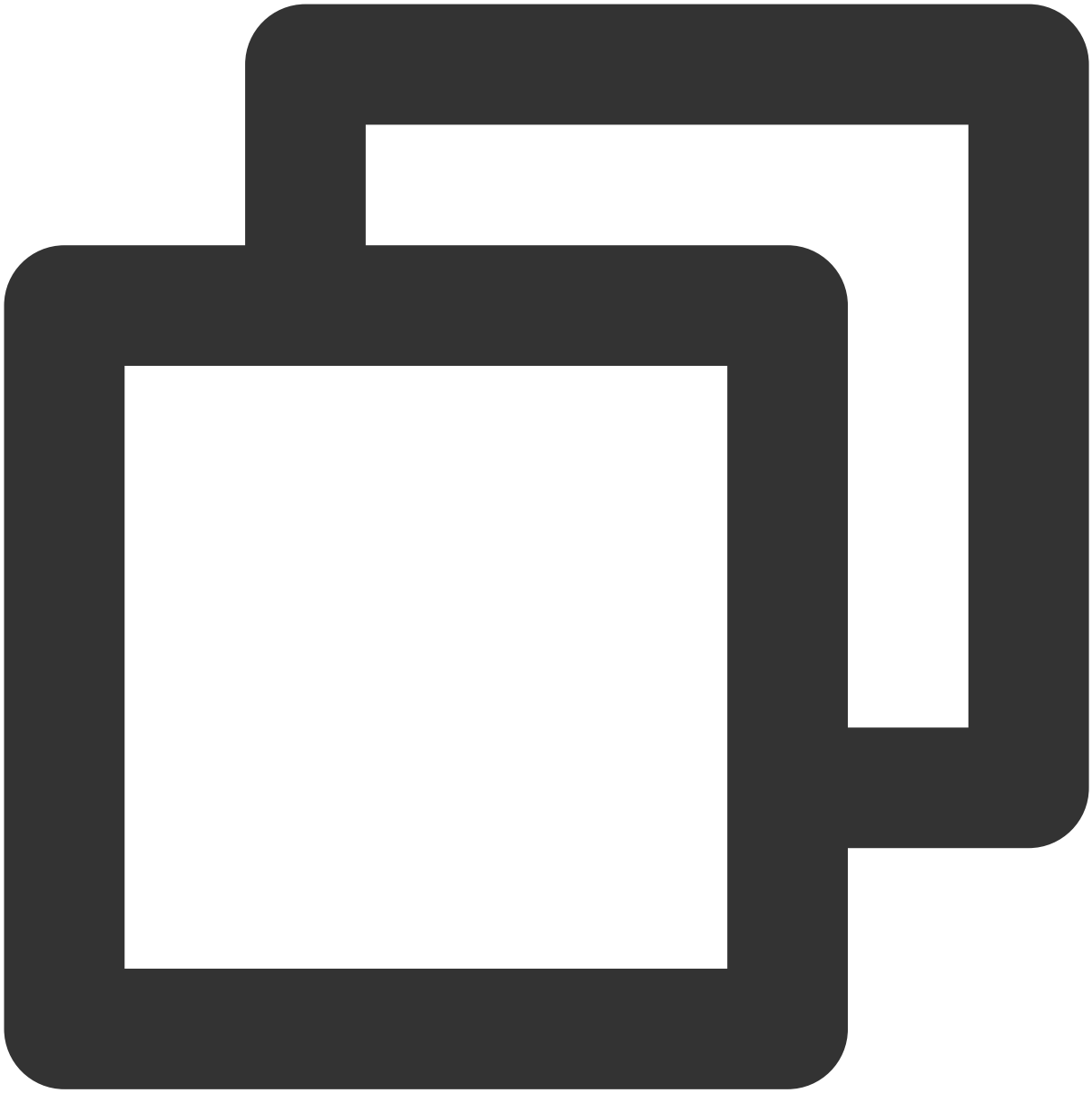
```
- (void)onCameraMuted:(BOOL)muted userId:(NSString *)userId;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
muted	BOOL	無効にするかどうか。
userId	NSString	キャスターユーザーID。

onReceiveKickedOff

キャスターによるキックアウトのコールバック。



```
- (void)onReceiveKickedOff:(NSString *)userId;
```

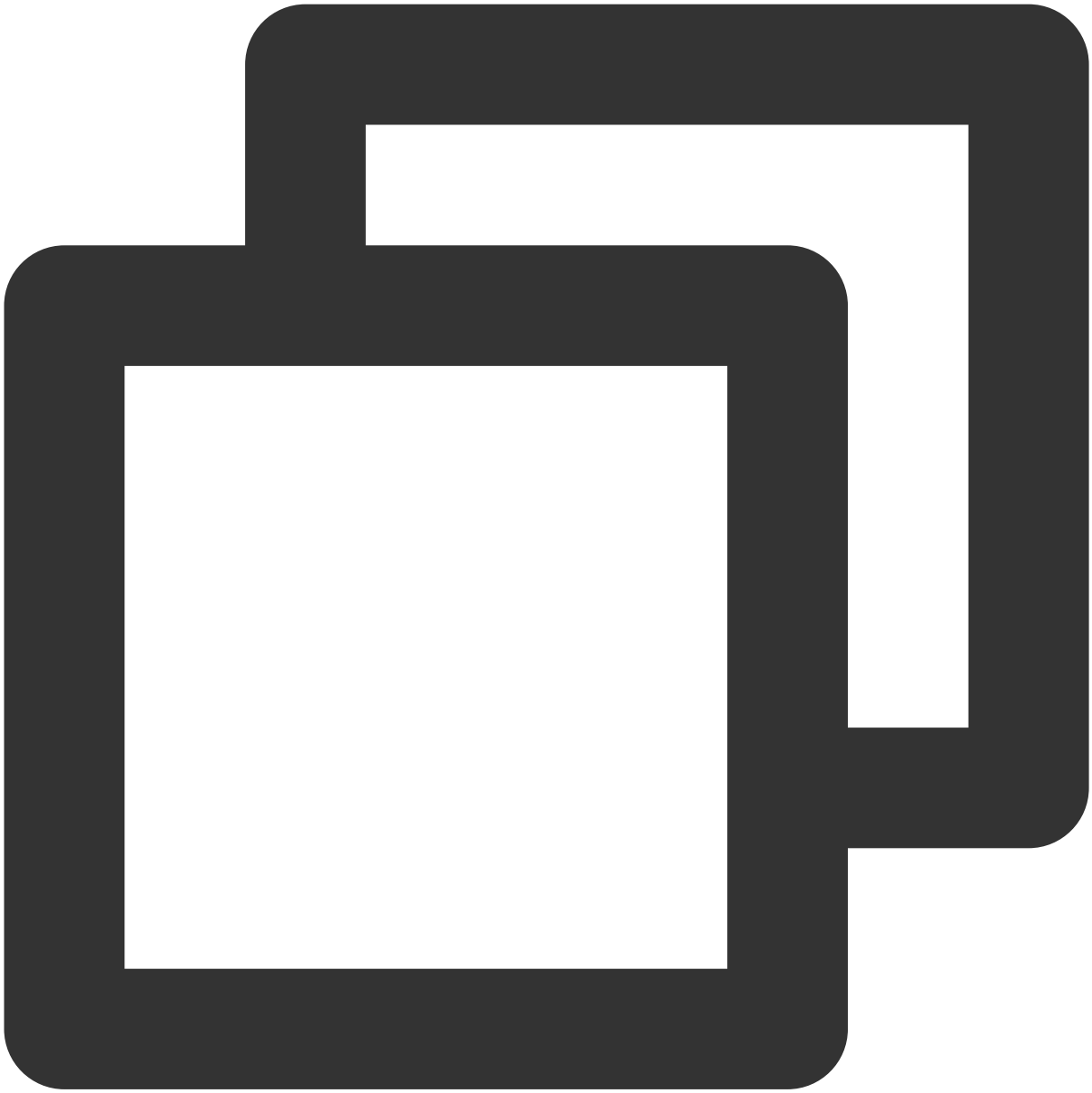
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	NSString	キャスター/管理者ユーザーID。

統計および品質コールバック

onStatistics

技術指標統計のコールバック。



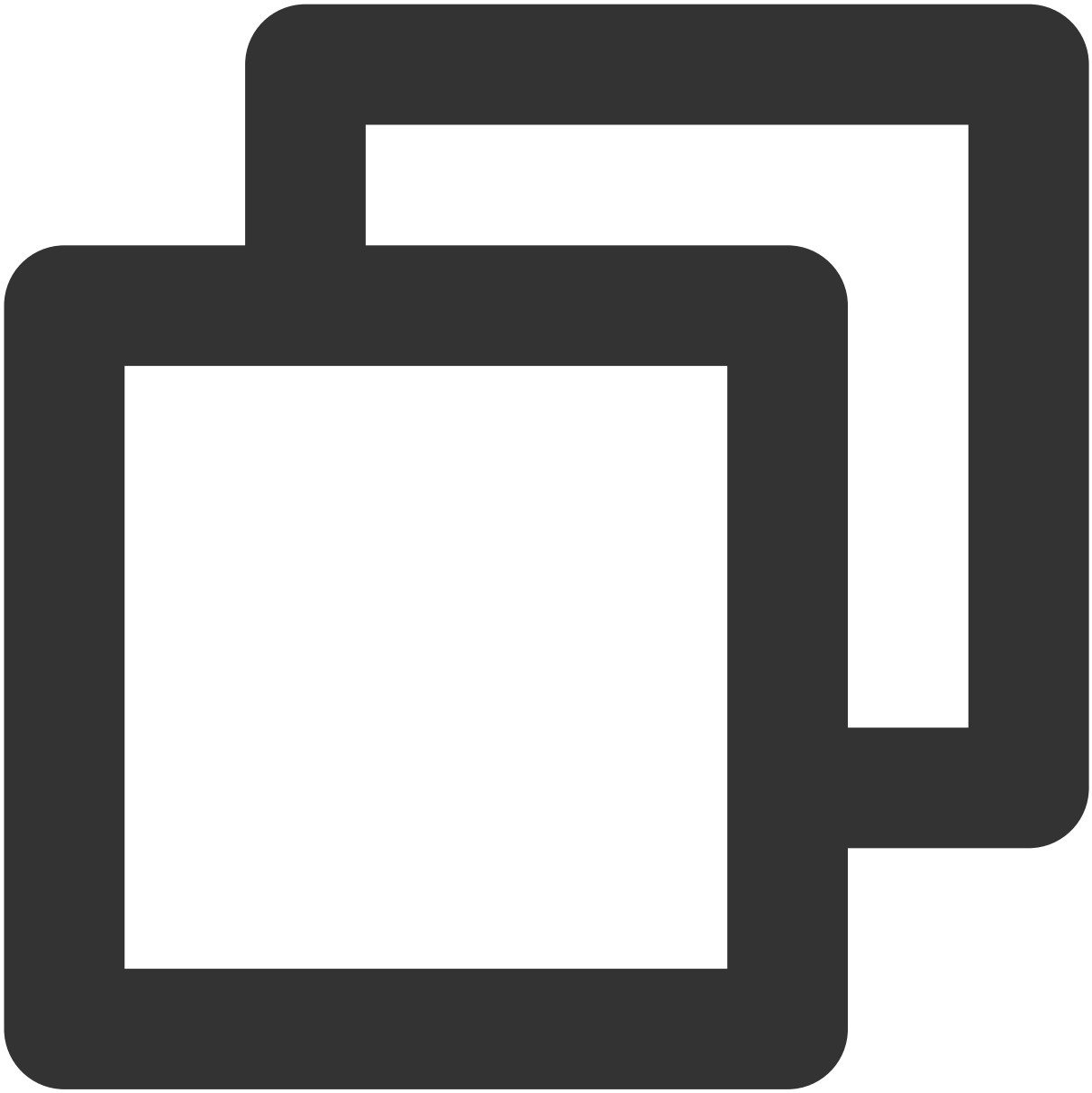
```
- (void)onStatistics:(TRTCStatistics *)statistics;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
statis	TRTCStatistics	統計データ。

onNetworkQuality

ネットワーク状況のコールバック。



```
- (void)onNetworkQuality:(TRTCQualityInfo *)localQuality remoteQuality:(NSArray<TRT
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
localQuality	TRTCQualityInfo	アップストリームネットワークの品質。

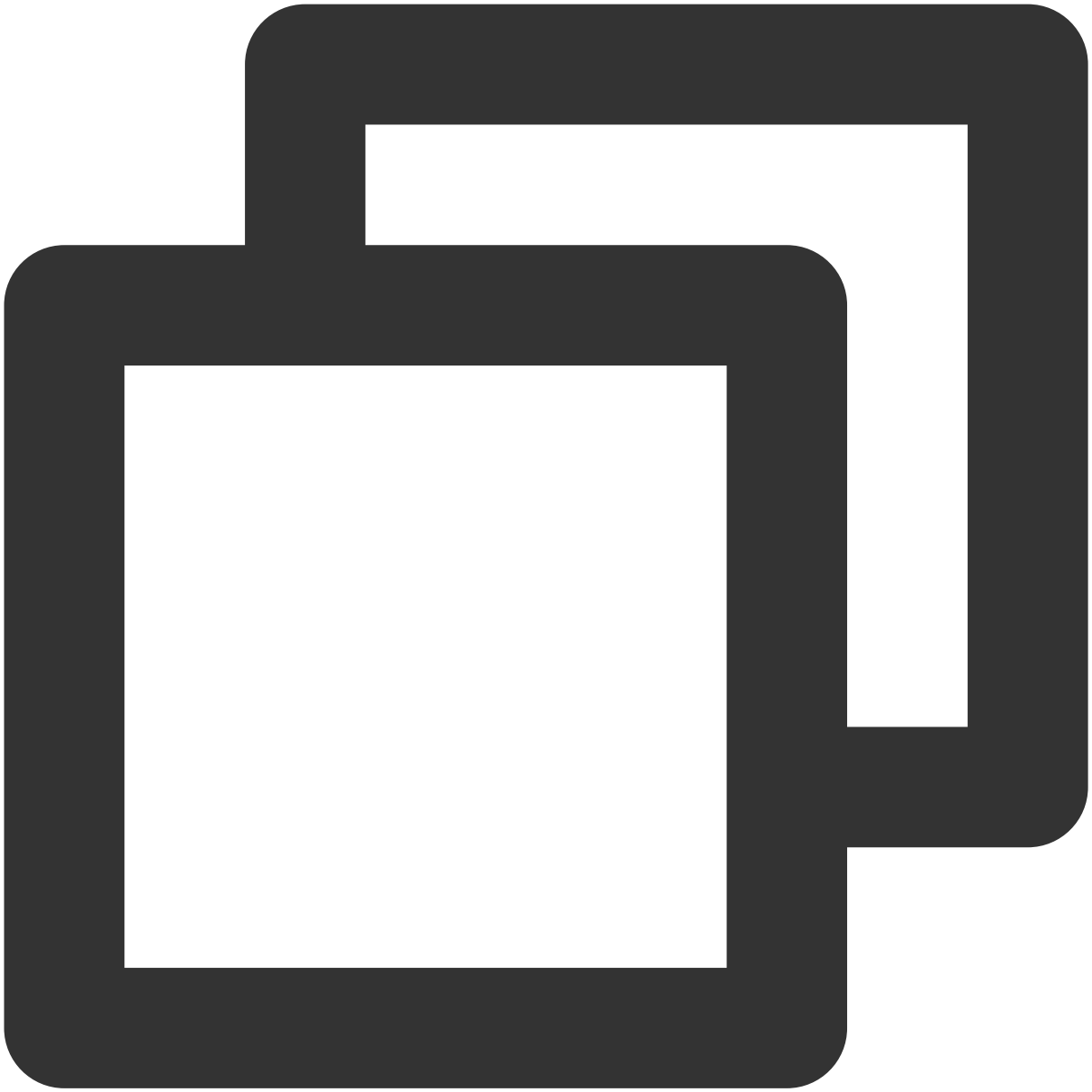
remoteQuality	NSArray<TRTCQualityInfo *>	ダウンストリームネットワークの品質。

説明：
詳細については、TRTC SDKをご参照ください。

画面共有イベントコールバック

onScreenCaptureStarted

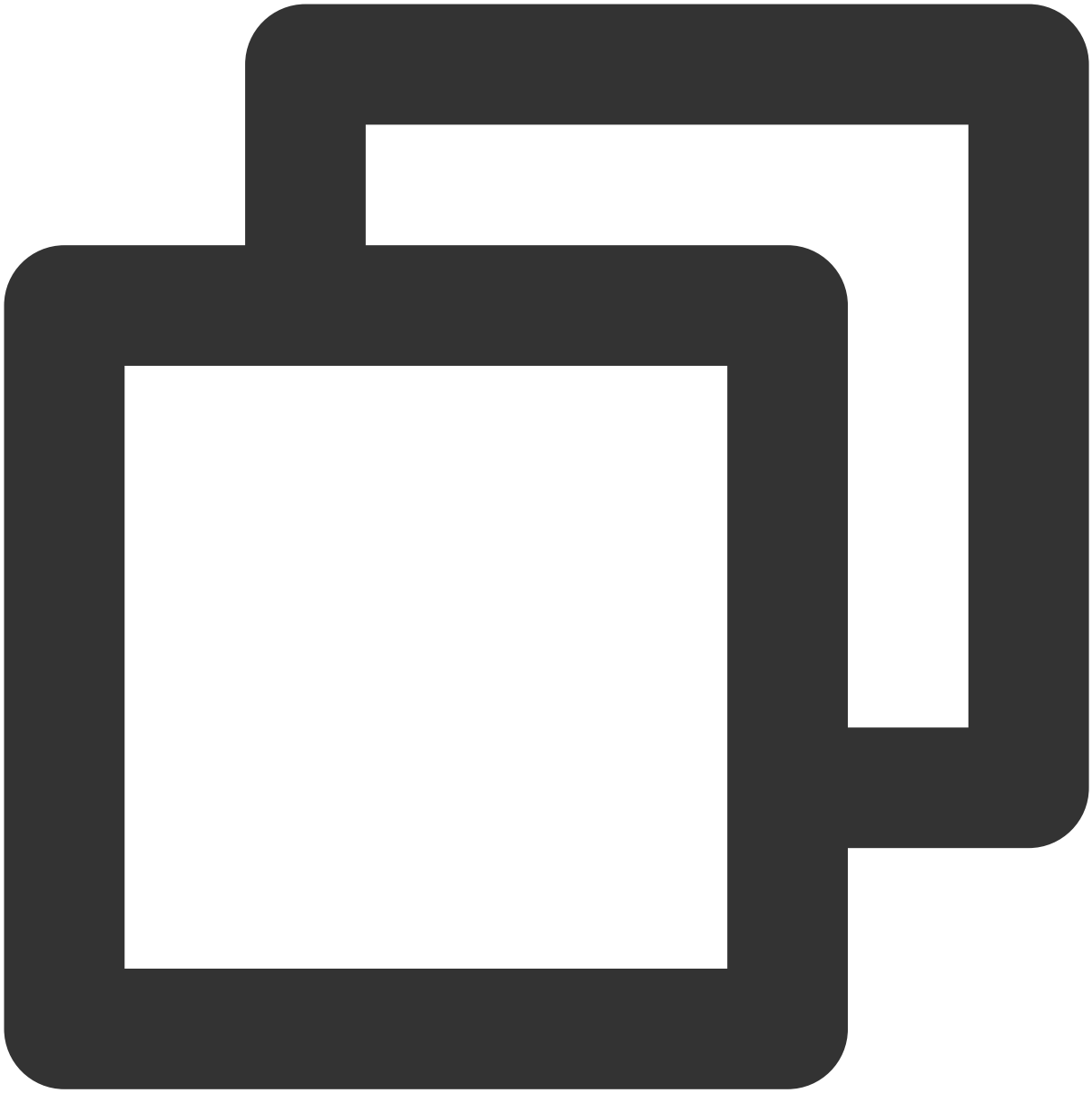
画面共有開始のコールバック。



```
- (void)onScreenCaptureStarted;
```

onScreenCaptureStopped

画面共有停止のコールバック。



```
- (void)onScreenCaptureStopped:(NSInteger)reason;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
reason	NSInteger	停止の理由。0：ユーザーの自発的な停止。1：その他のアプリケーションに占有されたことによる停止。

TRTCMeeting API（Flutter）

最終更新日：：2024-07-19 15:35:35

TRTCMeetingは、Tencent CloudのTRTCおよびIMサービスを基に組み合わせたコンポーネントで、以下の機能をサポートしています。
キャスターがミーティングルームを作成し、参加者はルームナンバーを入力した後にミーティングに参加します。
参加者の間で画面共有を行います。
各種のテキストメッセージとカスタムメッセージの送信をサポートします。
説明：
TUIKitシリーズコンポーネントはTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期的にアクティブにすることができます。IMサービスの課金ルールの詳細については、Instant Messagingの料金説明をご参照ください。TRTCをアクティブにすると、デフォルトでは、100DAUまでサポートするIM SDK体験版もアクティブになります。
TRTCMeetingは、1つのオープンソースのClassであり、Tencent Cloudの2つのクローズドソースのSDKに依存しています。具体的な実現プロセスは、多人数オーディオビデオルーム(Flutter)をご参照ください。
TRTC SDK： TRTC SDK を低遅延のビデオミーティングのコンポーネントとして使用します。
IM SDK：IM SDKのMeetingRoomを利用して、ミーティング中のチャットルームの機能を実現します。

TRTCMeeting API概要

SDK基本インターフェース

API	説明
sharedInstance	シングルトンオブジェクトを取得します。
destroySharedInstance	シングルトンオブジェクトを破棄します。
registerListener	イベント監視を設定します。
unRegisterListener	イベント監視を破棄します。
login	ログイン。
logout	ログアウト。
setSelfProfile	個人情報を修正します。

ミーティングルームに関するインターフェース

API	説明
createMeeting	ルームの作成（キャスターが呼び出し）。
destroyMeeting	ルームの破棄（キャスターが呼び出し）。
enterMeeting	入室（参加者が呼び出し）。
leaveMeeting	退室（参加者が呼び出し）。
getUserInfoList	ルーム内の全メンバーのリストを取得します。enterMeeting()成功後に呼び出しが有効となります。
getUserInfo	ルーム内の指定メンバーの詳細情報を取得します。enterMeeting()成功後に呼び出しが有効となります。

リモートユーザーに関するインターフェース

API	説明
startRemoteView	指定メンバーのリモートビデオ画面を再生します。
stopRemoteView	指定メンバーのリモートビデオ画面の再生を停止します。
setRemoteViewParam	指定メンバーのリモート画像のレンダリングパラメータを設定します。
muteRemoteAudio	指定メンバーのリモート音声をミュート/ミュート取り消しにします。
muteAllRemoteAudio	全メンバーのリモート音声をミュート/ミュート取り消しにします。
muteRemoteVideoStream	指定メンバーのリモートビデオを一時停止/再開します。
muteAllRemoteVideoStream	全メンバーのリモートビデオストリームを一時停止/再開します。

ローカルのビデオ操作インターフェース

API	説明
startCameraPreview	ローカルビデオのプレビュー画面を立ち上げます。
stopCameraPreview	ローカルのビデオキャプチャおよびプレビューを停止します。
switchCamera	フロント/リアカメラを切り替えます。
setVideoEncoderParam	ビデオエンコーダ関連パラメータを設定します。
setLocalViewMirror	ローカル画面のミラーモードのプレビューを設定します。

ローカルのオーディオ操作インターフェース

API	説明
startMicrophone	マイクの集音開始。
stopMicrophone	マイクの集音停止。
muteLocalAudio	ローカル音声のミュート起動/停止。
setSpeaker	スピーカーまたはヘッドホン起動の設定。
setAudioCaptureVolume	マイクの集音音量設定。
setAudioPlayoutVolume	再生音量の設定。
startAudioRecording	録音の開始。
stopAudioRecording	録音の停止。
enableAudioVolumeEvaluation	音声レベルリマインダを有効にします。

スクリーンキャプチャのインターフェース

API	説明
startScreenCapture	画面共有を開始。
stopScreenCapture	画面キャプチャの停止。
pauseScreenCapture	スクリーンキャプチャの一時停止。
resumeScreenCapture	スクリーンキャプチャの再開。

管理オブジェクト取得に関するインターフェース

API	説明
getDeviceManager	デバイス管理オブジェクトTXDeviceManagerを取得します。
getBeautyManager	美顔管理オブジェクトTXBeautyManagerを取得します。

メッセージ送信関連インターフェース

API	説明
sendRoomTextMsg	ミーティング中にテキストメッセージをブロードキャストします。通常、チャット

	に使用します。
sendRoomCustomMsg	カスタマイズしたテキストメッセージを送信します。

TRTCLiveRoomDelegate API概要

一般的なイベントコールバック

API	説明
onError	エラーのコールバック。
onWarning	警告のコールバック。
onKickedOffline	他のユーザーが同じアカウントでログインすると、キックアウトされてオフラインになります。

ミーティングルームのイベントコールバック

API	説明
onRoomDestroy	ミーティングルームが破棄された時のコールバック。
onNetworkQuality	ネットワーク状態のコールバック。
onUserVolumeUpdate	ユーザー通話音量のコールバック。

参加者入退室のイベントコールバック

API	説明
onEnterRoom	ローカルの入室のコールバック。
onLeaveRoom	ローカルの退室のコールバック。
onUserEnterRoom	新しい参加者の入室のコールバック。
onUserLeaveRoom	参加者の退出のコールバック。

参加者の音声・ビデオのイベントコールバック

API	説明
onUserAudioAvailable	参加者がマイクを起動/停止したときのコールバック。

onUserVideoAvailable	参加者がカメラを起動/停止したときのコールバック。
onUserSubStreamAvailable	参加者がサブチャンネル画面を起動/停止したときのコールバック。

メッセージイベントのコールバック

API	説明
onRecvRoomTextMsg	テキストメッセージ受信のコールバック。
onRecvRoomCustomMsg	カスタムメッセージ受信のコールバック。

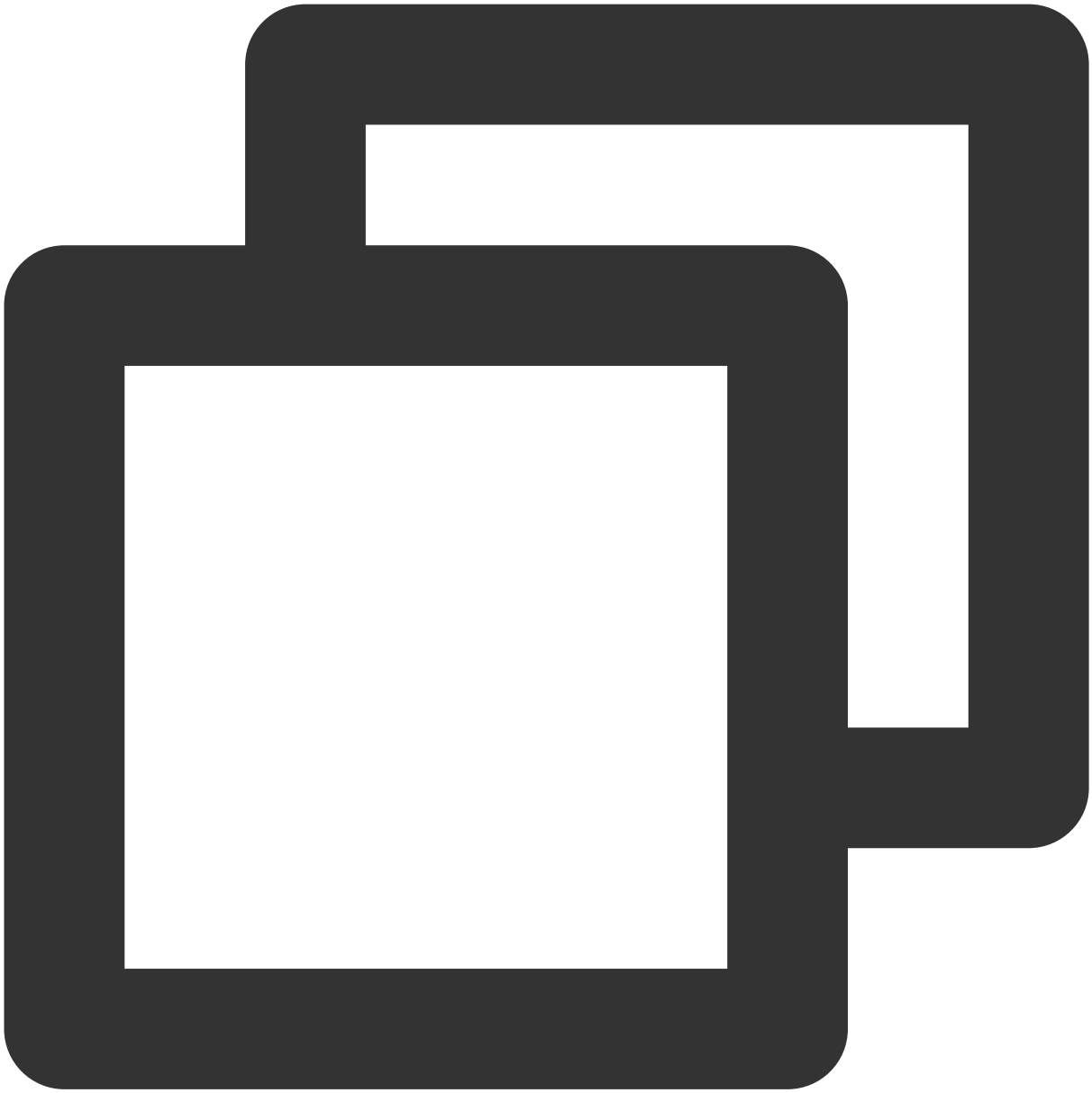
スクリーンキャプチャのイベントコールバック

API	説明
onScreenCaptureStarted	スクリーンキャプチャ開始のコールバック。
onScreenCapturePaused	スクリーンキャプチャ一時停止のコールバック。
onScreenCaptureResumed	スクリーンキャプチャ再開のコールバック。
onScreenCaptureStoped	スクリーンキャプチャ停止のコールバック。

SDK基本インターフェース

sharedInstance

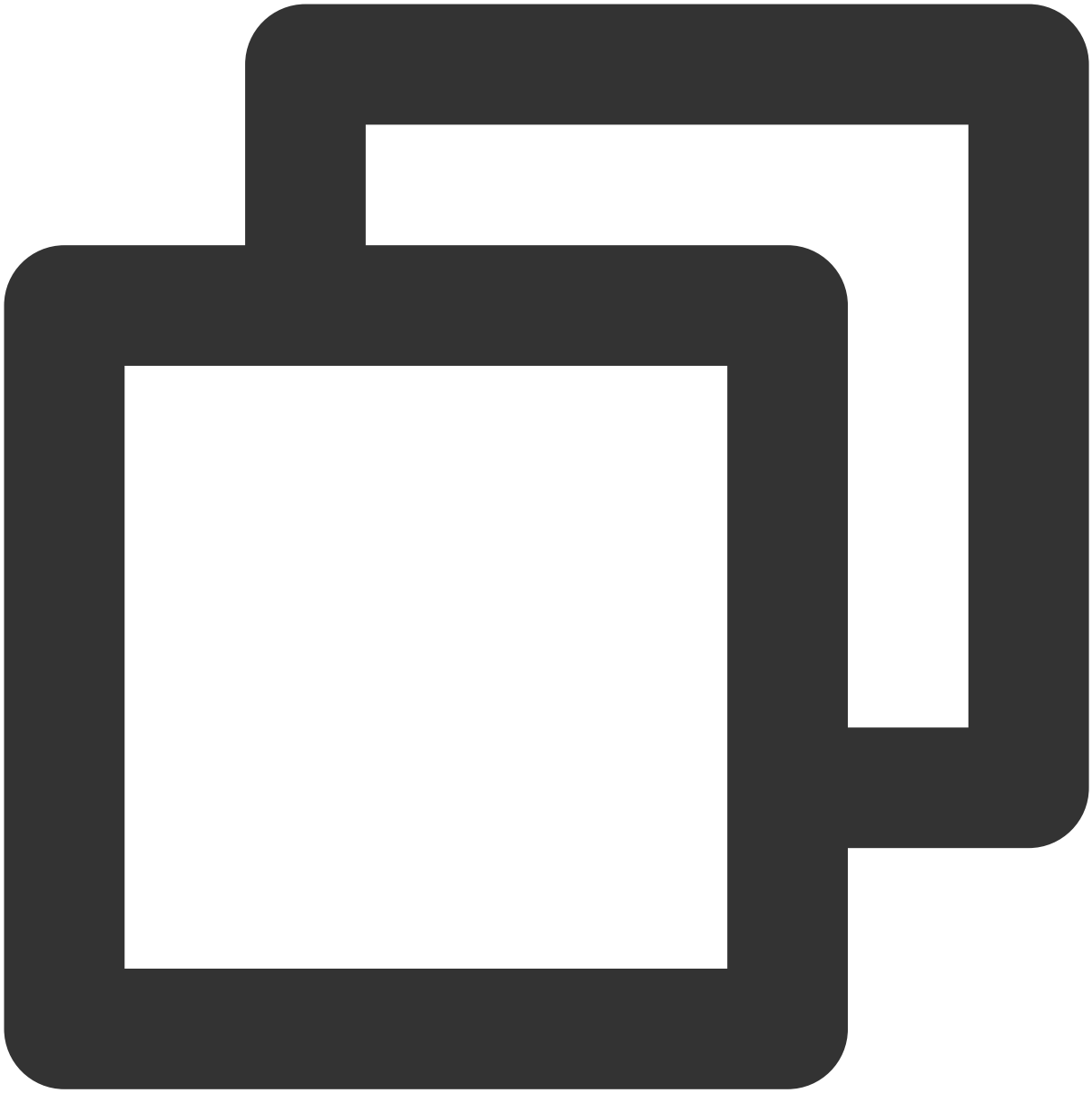
TRTCMeetingシングルトンオブジェクトを取得します。



```
static Future<TRTCMeeting> sharedInstance();
```

destroySharedInstance

TRTCMeetingシングルトンオブジェクトを破棄します。



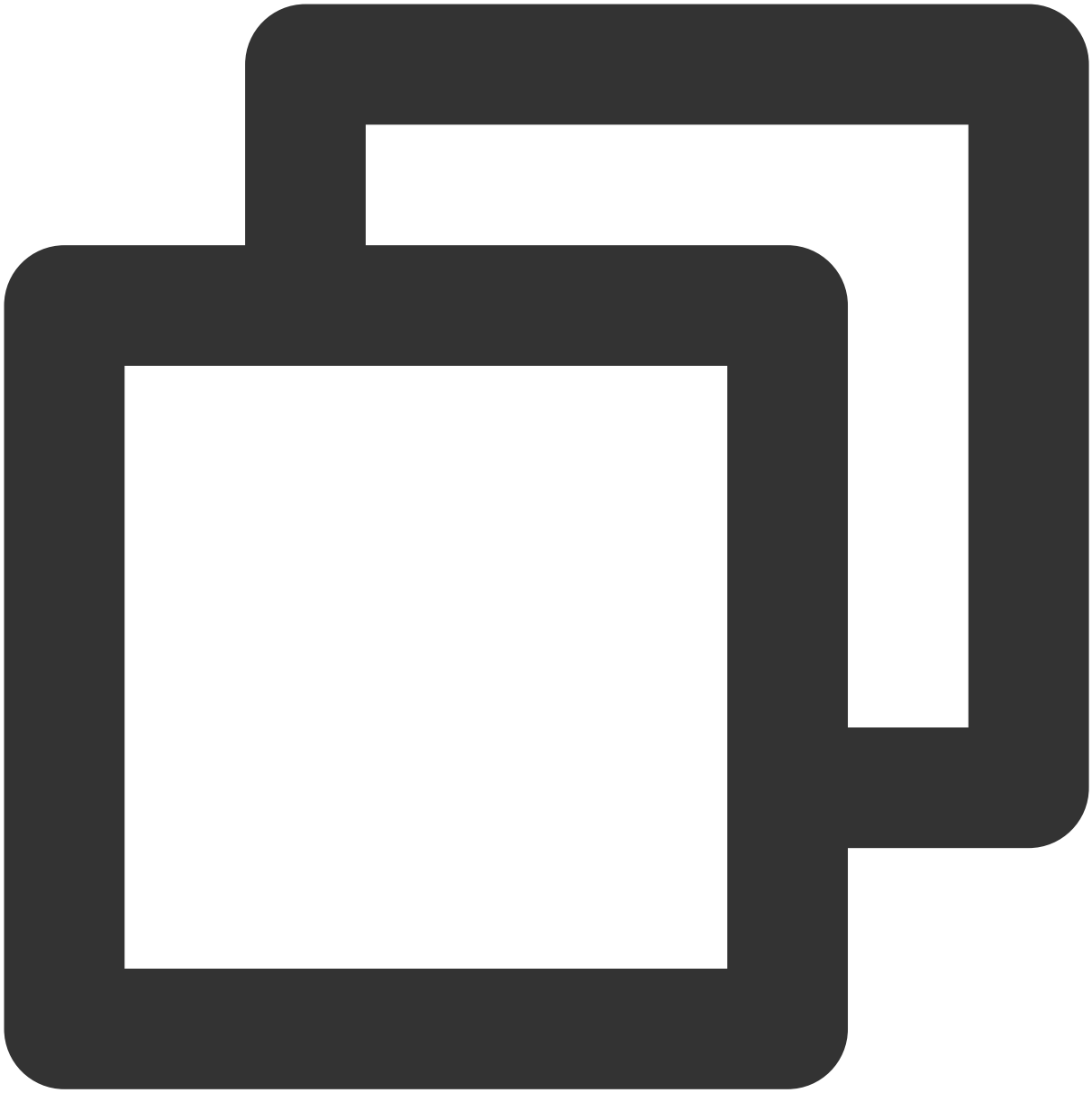
```
static void destroySharedInstance();
```

説明：

インスタンスを破棄すると、外部キャッシュのTRTCMeetingインスタンスは再利用できなくなります。あらためてsharedInstanceを呼び出し、新しいインスタンスを取得してください。

registerListener

イベント監視を設定します。TRTCMeetingDelegateを介して、TRTCMeetingの各種ステータスの通知を取得できます。



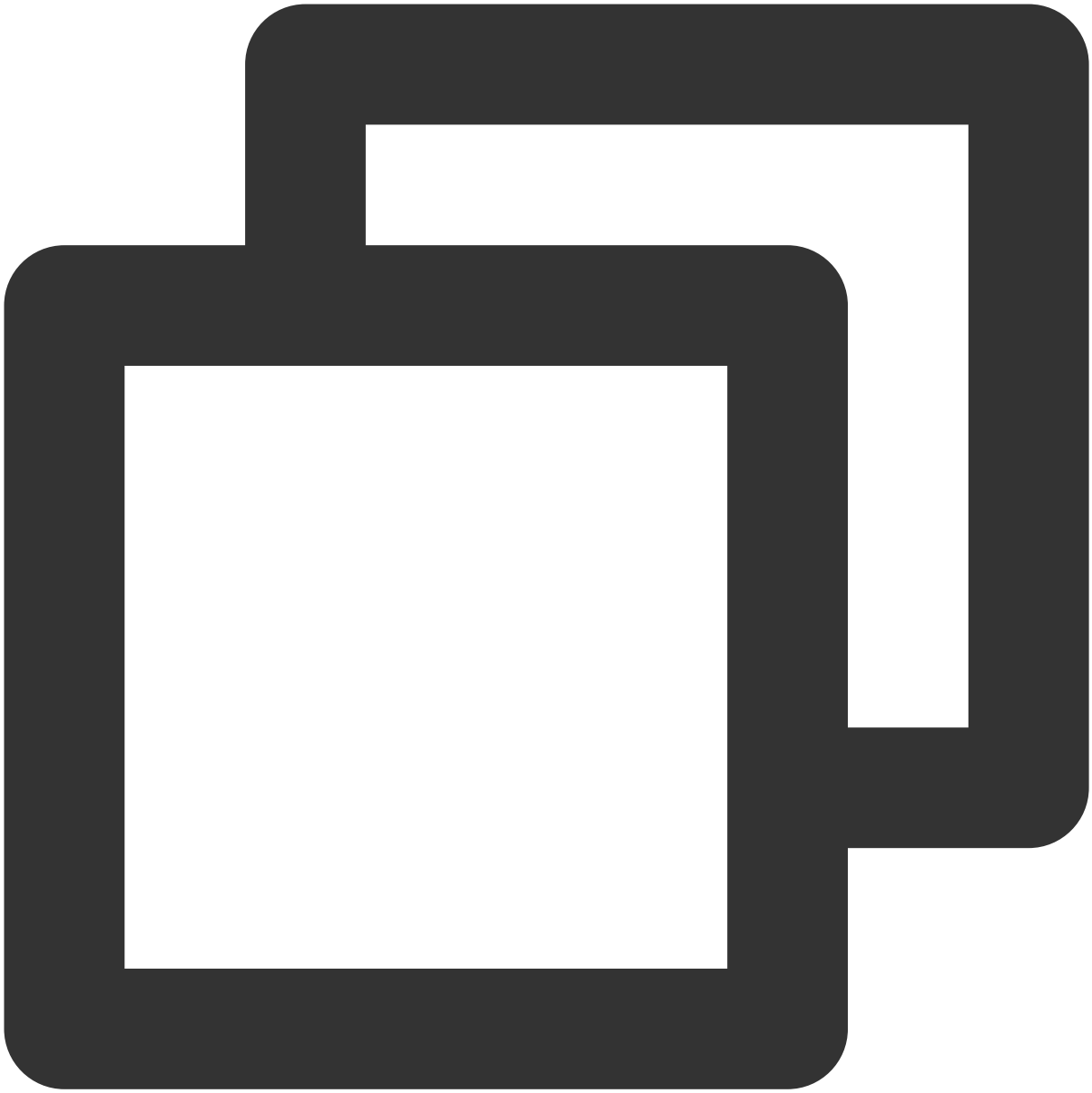
```
void registerListener(MeetingListenerFunc func);
```

説明：

funcはTRTCMeetingのプロキシコールバックです。

unRegisterListener

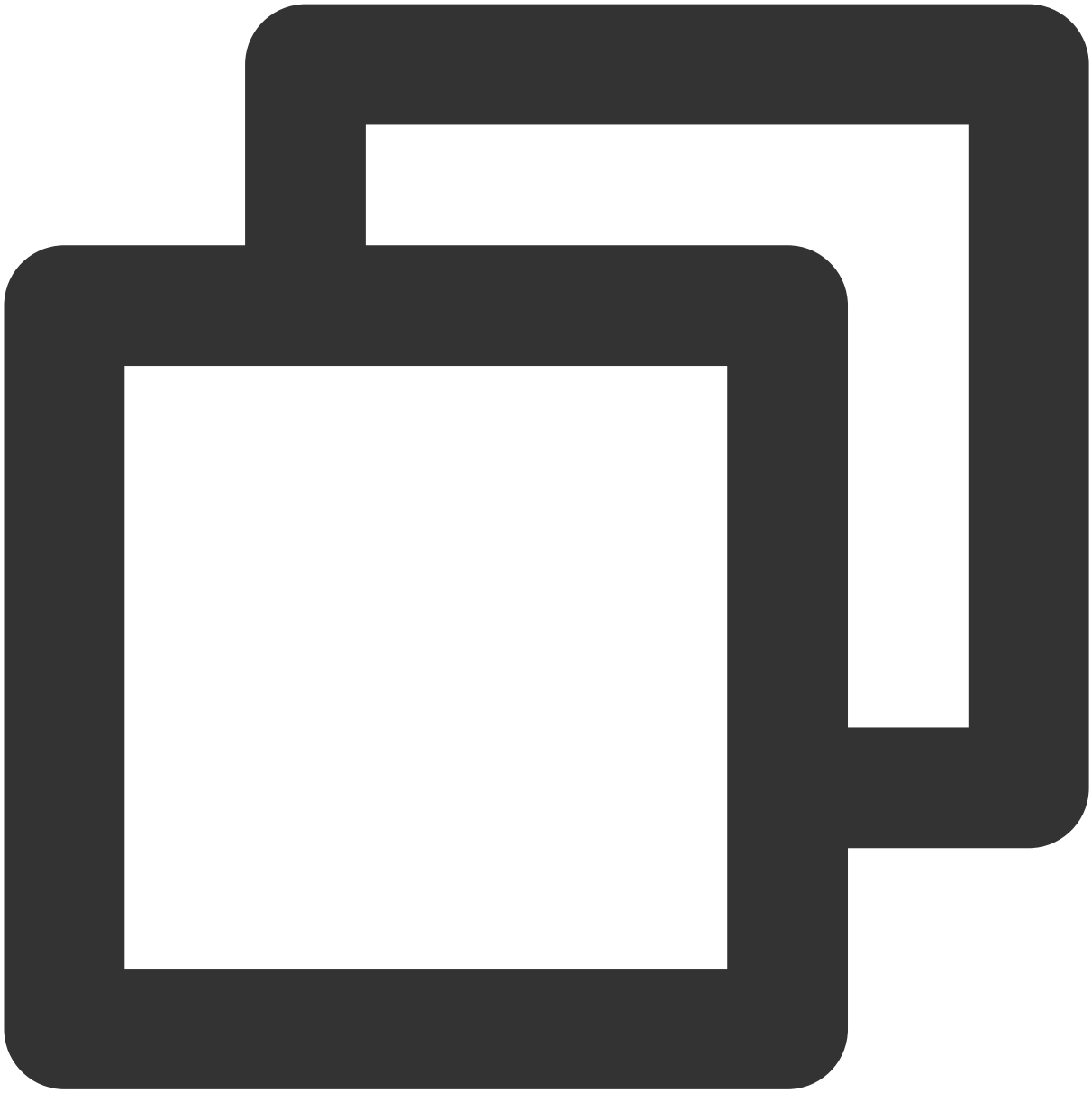
イベント監視を破棄します。



```
void unRegisterListener(MeetingListenerFunc func);
```

login

ログイン。



```
Future<ActionCallback> login(int sdkAppId, String userId, String userSig);
```

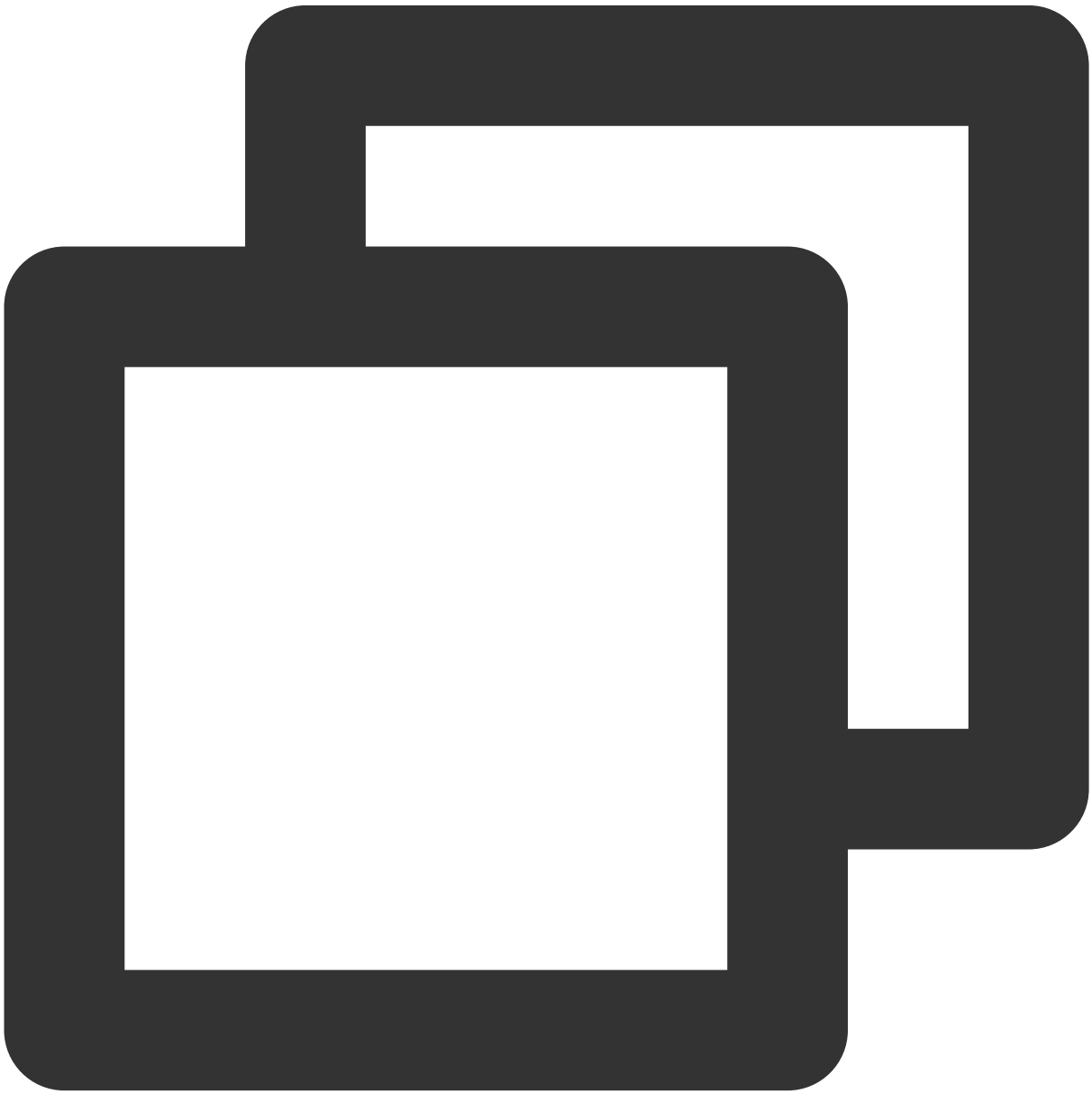
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
sdkAppId	int	TRTCコンソール > 【アプリケーション管理】 > アプリケーション情報の中でSDKAppIDを確認できます。
userId	String	現在のユーザーのID。文字列タイプ。アルファベット（a-z および A-Z）、数字（0-

		9）、ハイフン（-）、アンダーバー（_）のみ使用できます。
userSig	String	Tencent Cloudによって設計されたセキュリティ保護署名。取得方法については、UserSigの計算、使用方法をご参照ください。

logout

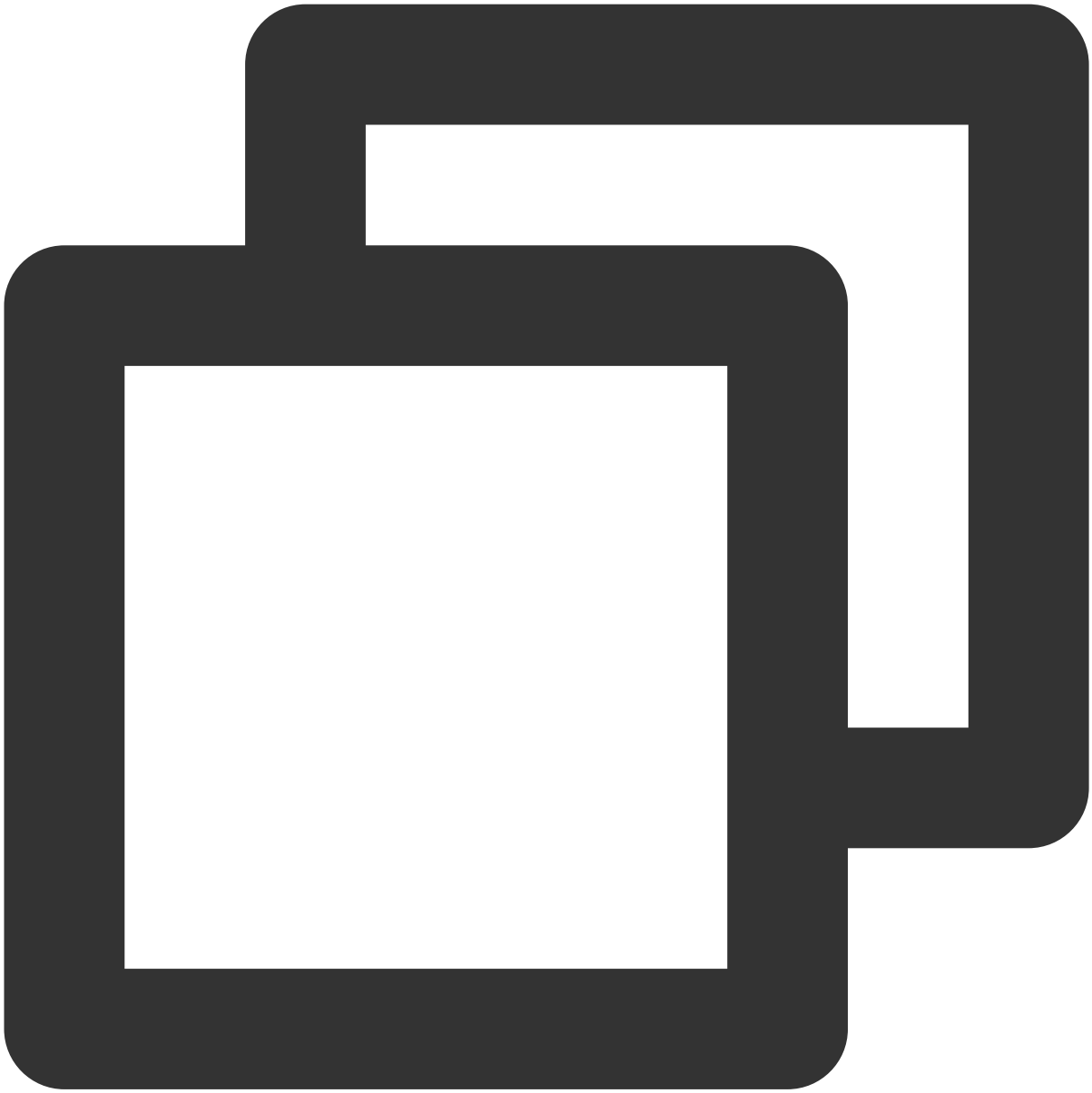
ログアウト。



```
Future<ActionCallback> logout();
```

setSelfProfile

個人情報の修正。



```
Future<ActionCallback> setSelfProfile(String userName, String avatarURL);
```

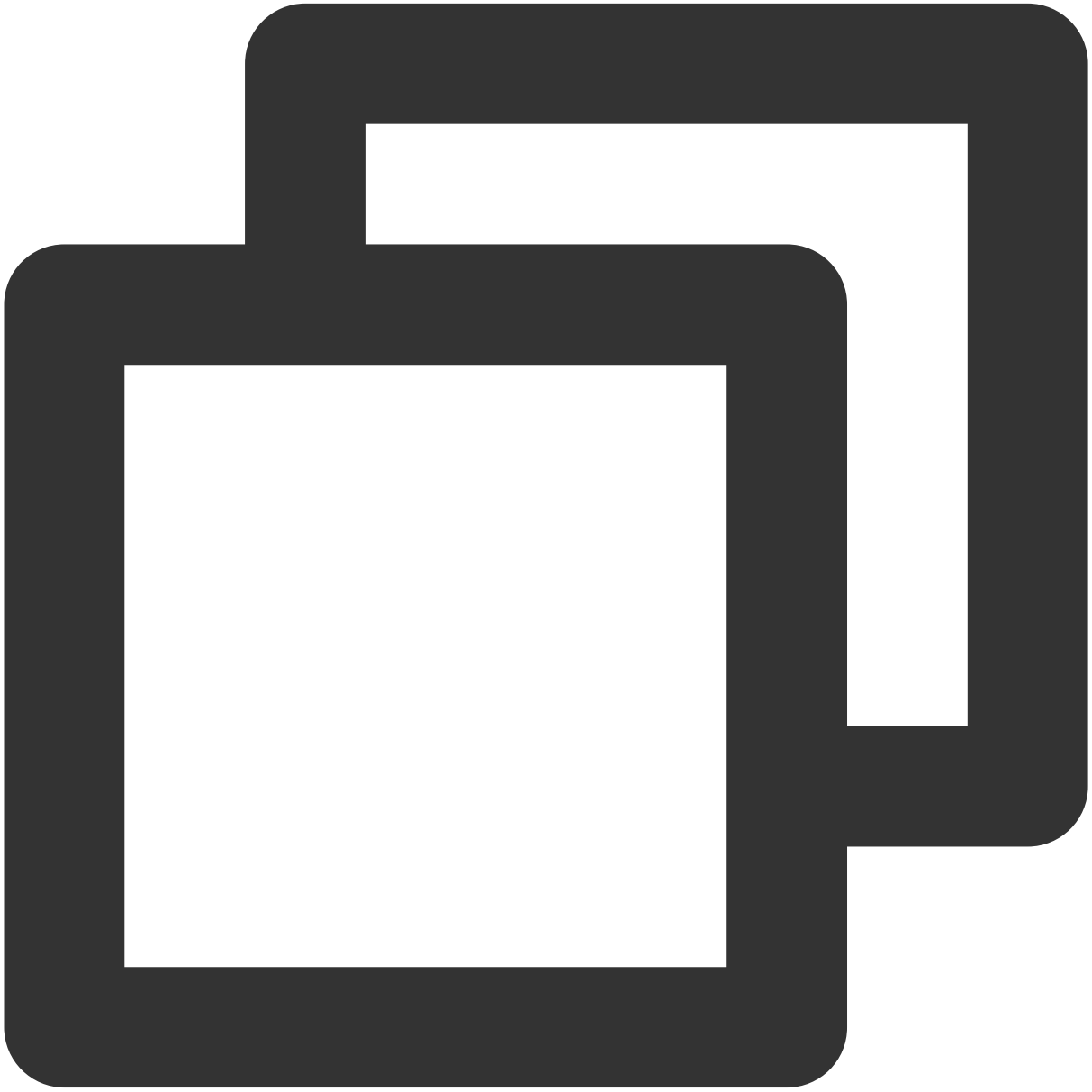
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userName	String	ユーザーニックネーム。
avatarURL	String	ユーザープロフィール画像のアドレス。

ミーティングルーム関連インターフェース

createMeeting

ミーティングの作成（キャスターの呼び出し）。



```
Future<ActionCallback> createMeeting(int roomId);
```

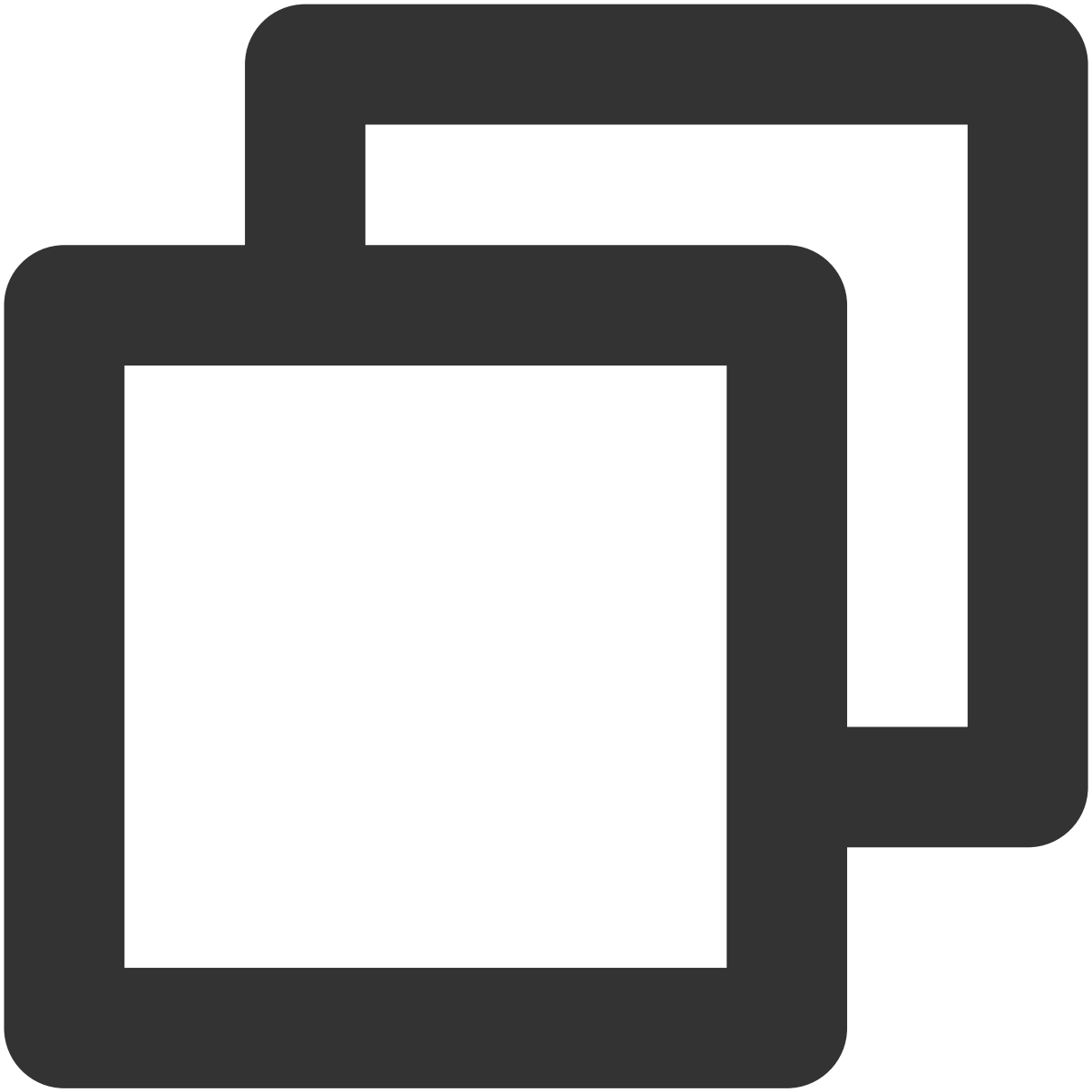
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
roomId	int	ミーティングルームID。ご自身でアサインし、一括管理してください。

キャスターの通常の呼び出しフローは以下のとおりです：
1. 【キャスター】 `createMeeting()` を呼び出し、 `roomId` を渡してミーティングを作成します。ミーティングルーム作成の成功の有無が `ActionCallback` によってキャスターに通知されます。
2. 【キャスター】 `startCameraPreview()` を呼び出し、カメラのプレビューを起動します。この時美顔パラメータを調整できます。
3. 【キャスター】 `startMicrophone()` を呼び出し、マイクキャプチャを起動します。

destroyMeeting

ミーティングルームを破棄します（キャスターが呼び出し）。キャスターは、ミーティング作成後、この関数を呼び出してミーティングを破棄できます。



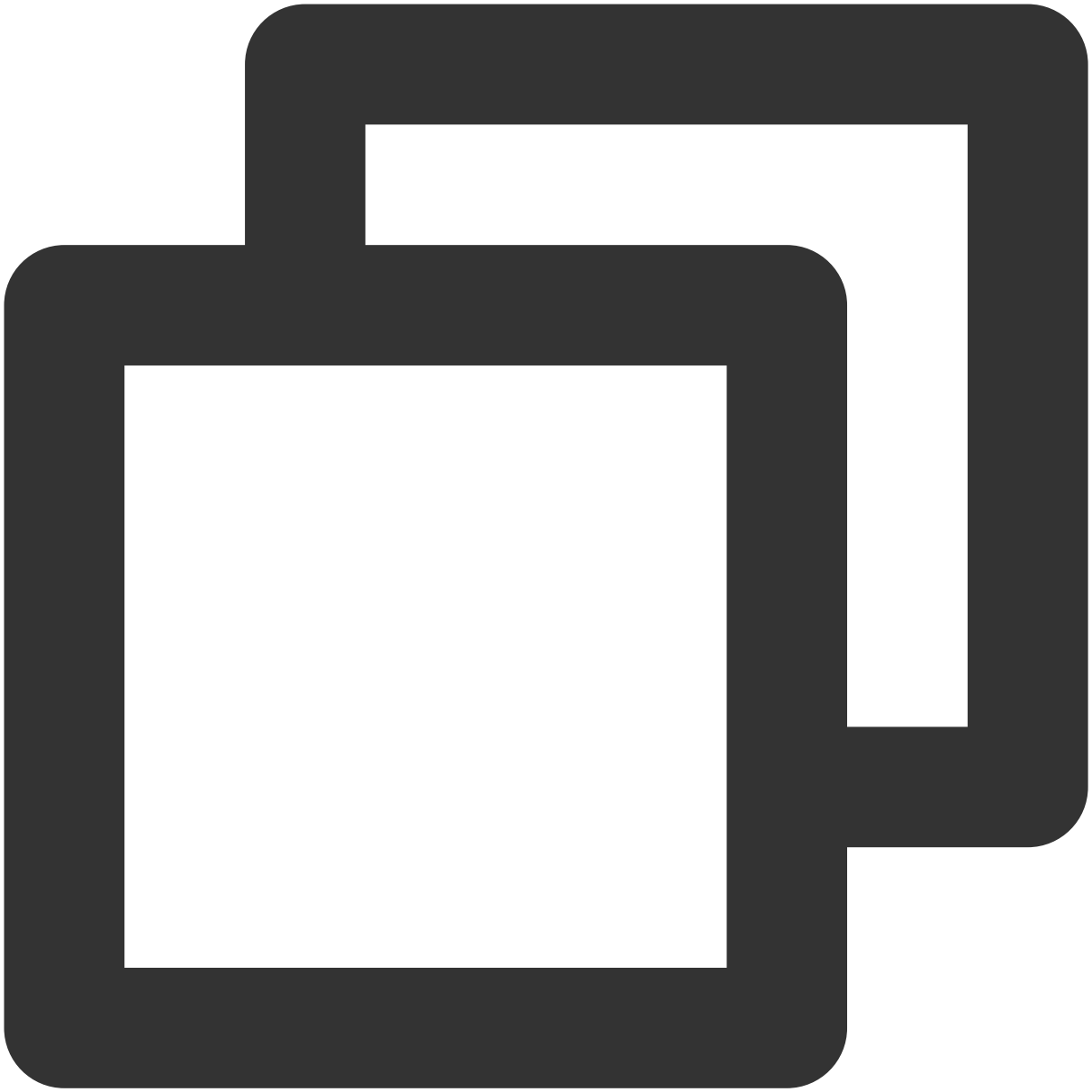
```
Future<ActionCallback> destroyMeeting(int roomId);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
roomId	int	ミーティングルームID。ご自身でアサインし、一括管理してください。

enterMeeting

ミーティングルームに参加します（参加者が呼び出し）。



```
Future<ActionCallback> enterMeeting(int roomId);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
roomId	int	ミーティングルームID。

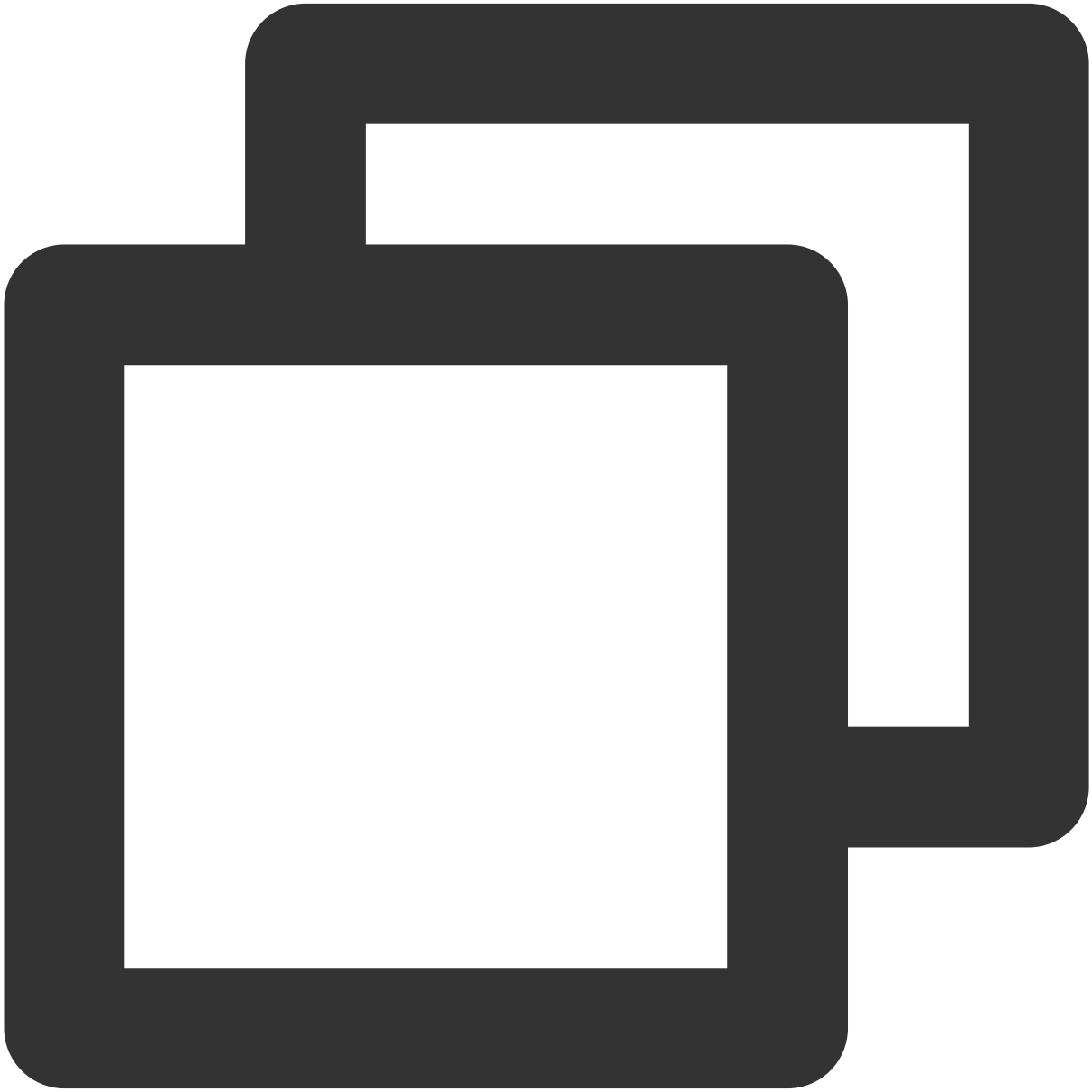
参加者がミーティングに参加する通常の呼び出しフローは以下のとおりです：

1. 【参加者】 `enterMeeting()` を呼び出し、 `roomId` を渡せばミーティングルームに入れます。

2. 【参加者】 `startCameraPreview()` を呼び出して、カメラのプレビューを起動し、 `startMicrophone()` を呼び出し、マイクキャプチャを起動します。
3. 【参加者】 `onUserVideoAvailable` のイベントを受信します。 `startRemoteView()` を呼び出し、メンバーのuserIdを渡して再生を開始します

leaveMeeting

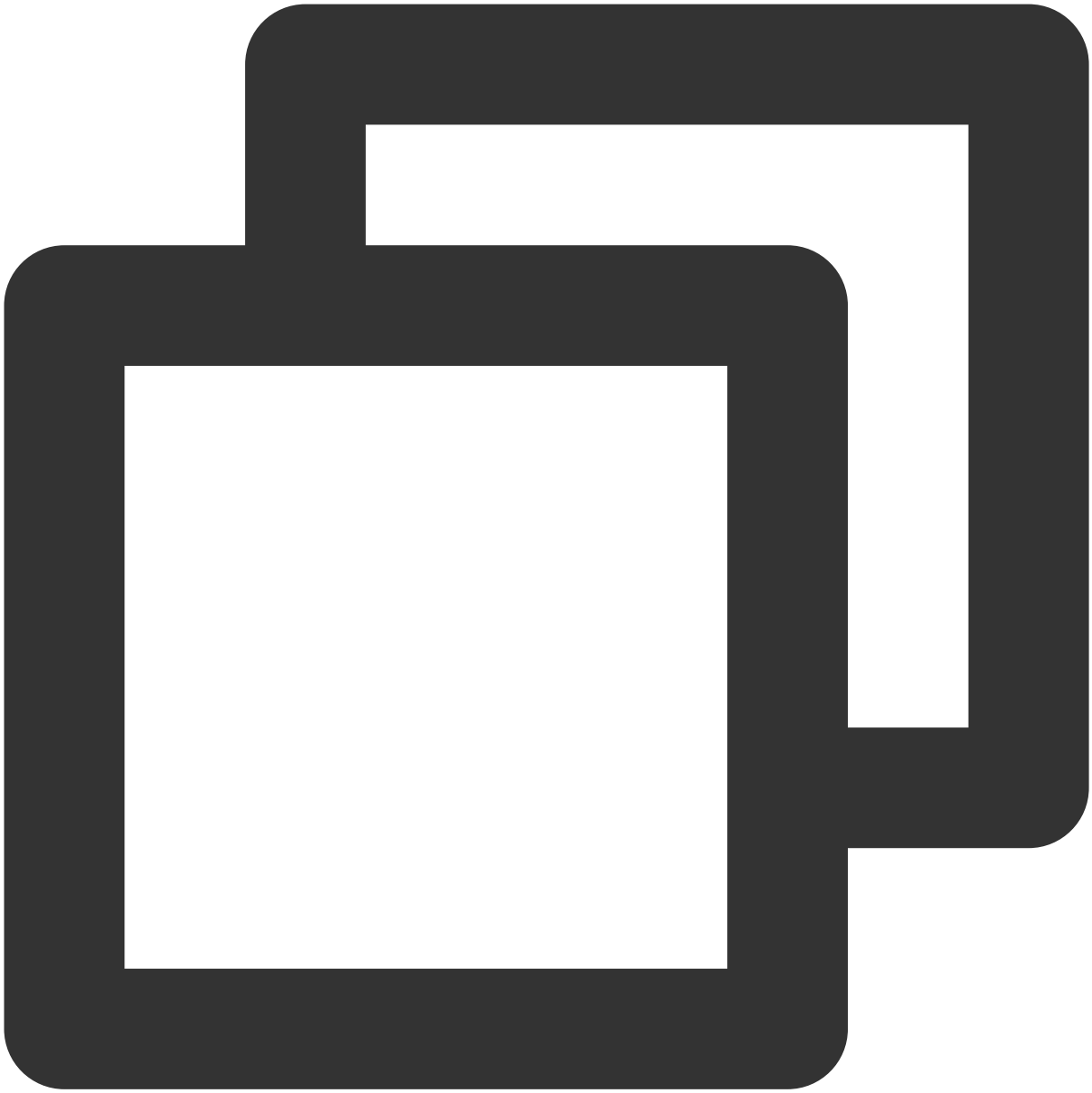
ミーティングルームから退出します（参加者が呼び出し）。



```
Future<ActionCallback> leaveMeeting();
```

getUserInfoList

ルーム内の全メンバーのリストを取得します。enterMeeting()成功後に呼び出しが有効となります。



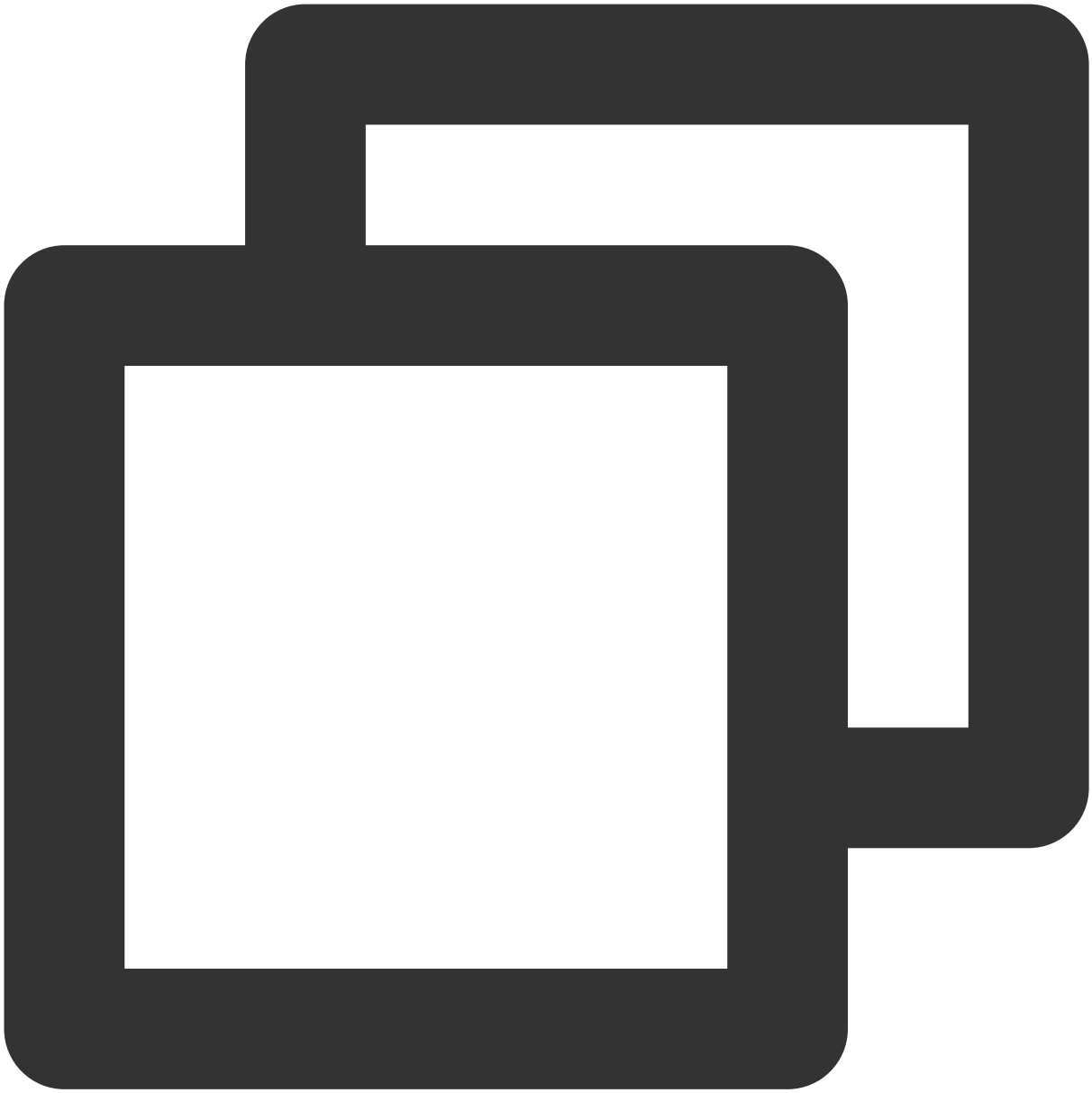
```
Future<UserListCallback> getUserInfoList(List<String> userIdList);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userIdList	List<String>	取得する必要があるuserId リスト。

getUserInfo

ルーム内の指定メンバーの詳細情報を取得します。enterMeeting()成功後に呼び出しが有効となります。



```
Future<UserListCallback> getUserInfo(String userId);
```

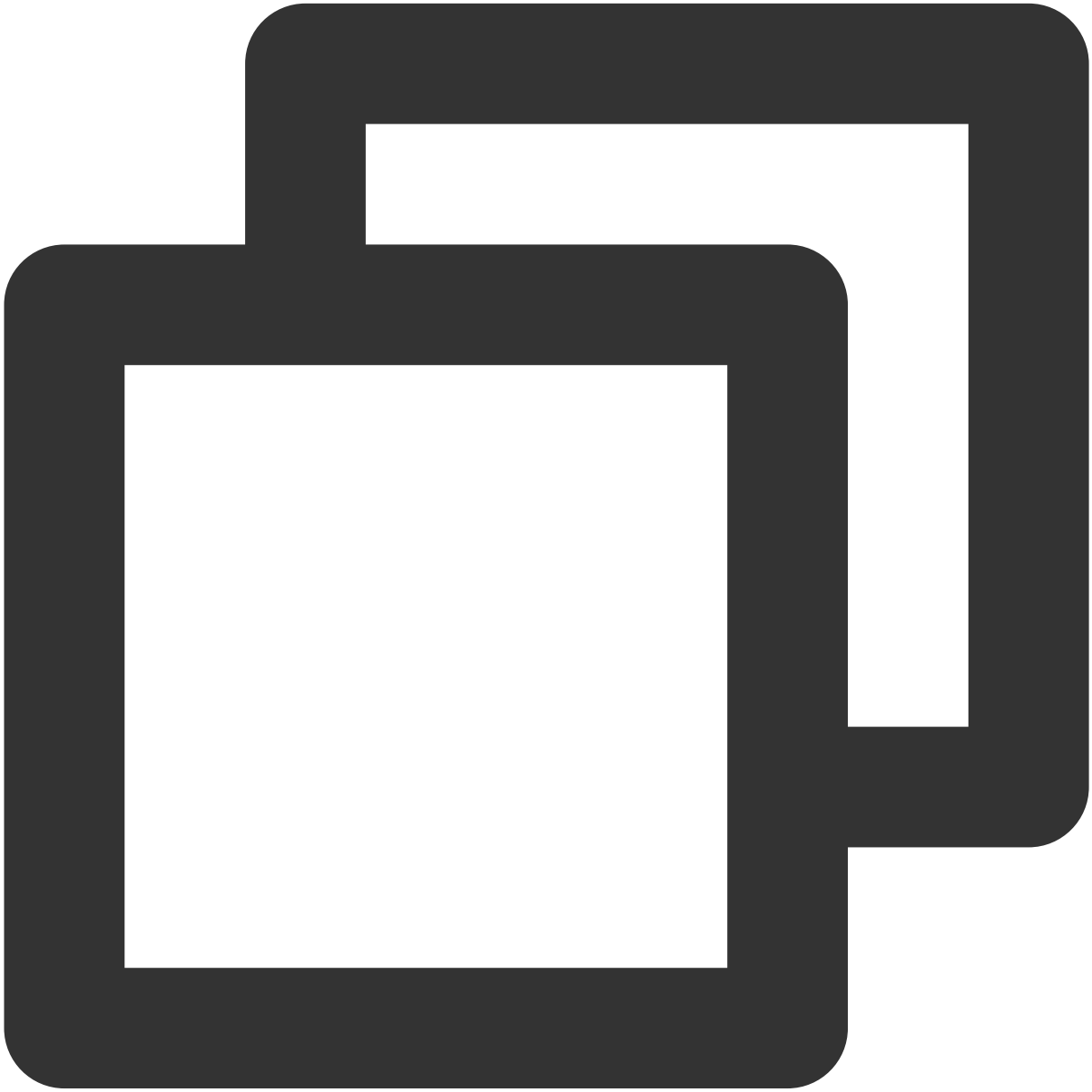
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userId	String	指定メンバーのID。

リモートユーザーに関するインターフェース

startRemoteView

指定メンバーのリモートビデオ画面を再生します。



```
Future<void> startRemoteView(String userId, int streamType, int viewId);
```

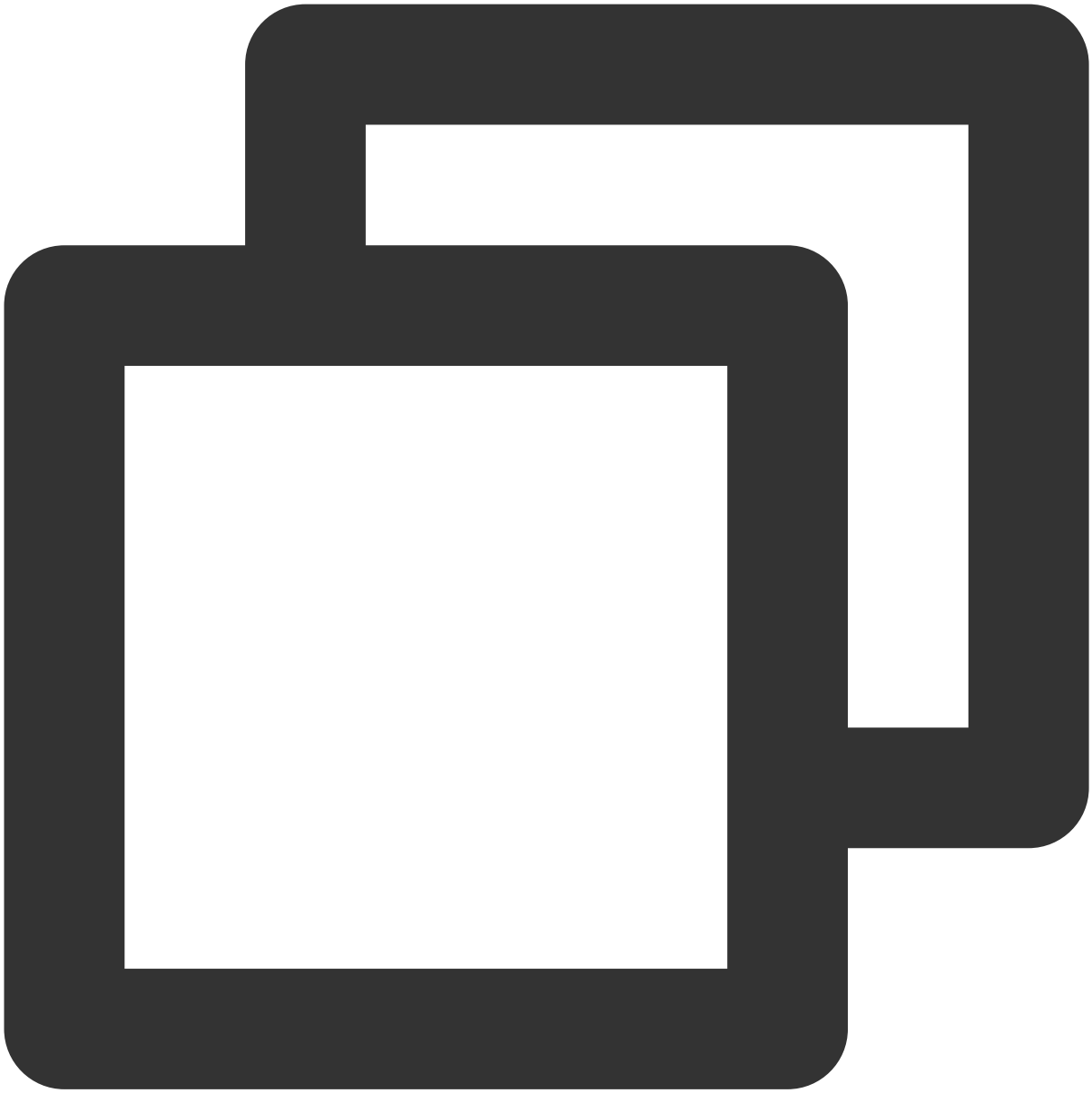
パラメータは下表に示すとおりです：

パラメータ	タイ	意味

	プ	
userId	String	指定メンバーのID。
streamType	int	視聴したいビデオストリームのタイプ。詳細については、TRTC SDKをご参照ください。
viewId	int	TRTCCloudVideoViewが発行するviewId。

stopRemoteView

指定メンバーのリモートビデオ画面の再生を停止します。



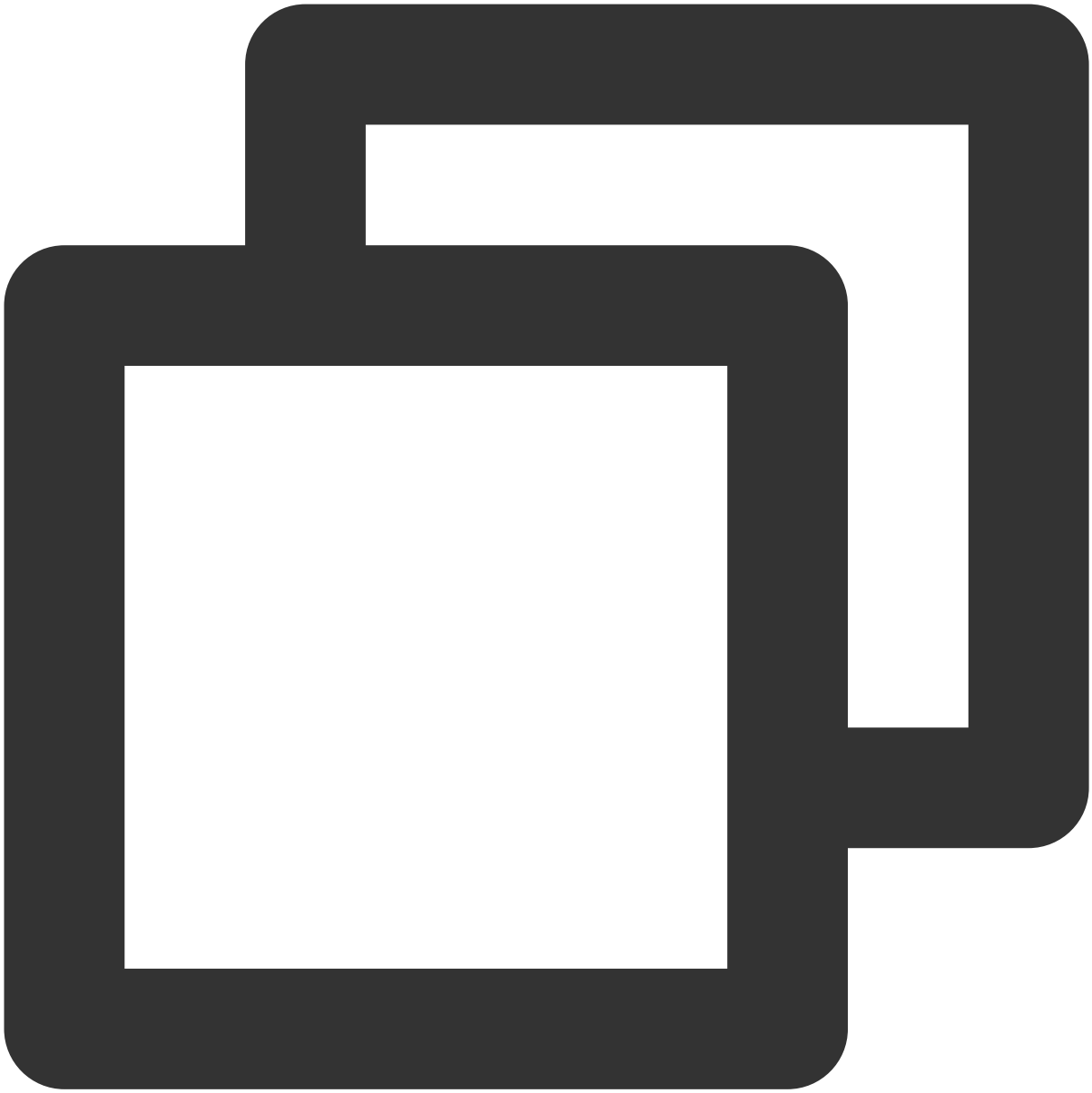
```
Future<void> stopRemoteView(String userId, int streamType);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userId	String	指定メンバーのID。
streamType	int	視聴したいビデオストリームのタイプ。詳細については、TRTC SDKをご参照ください。

setRemoteViewParam

指定メンバーのリモート画像のレンダリングパラメータを設定します。



```
Future<void> setRemoteViewParam(String userId, int streamType,
  {int fillMode, int rotation, int mirrorType});
```

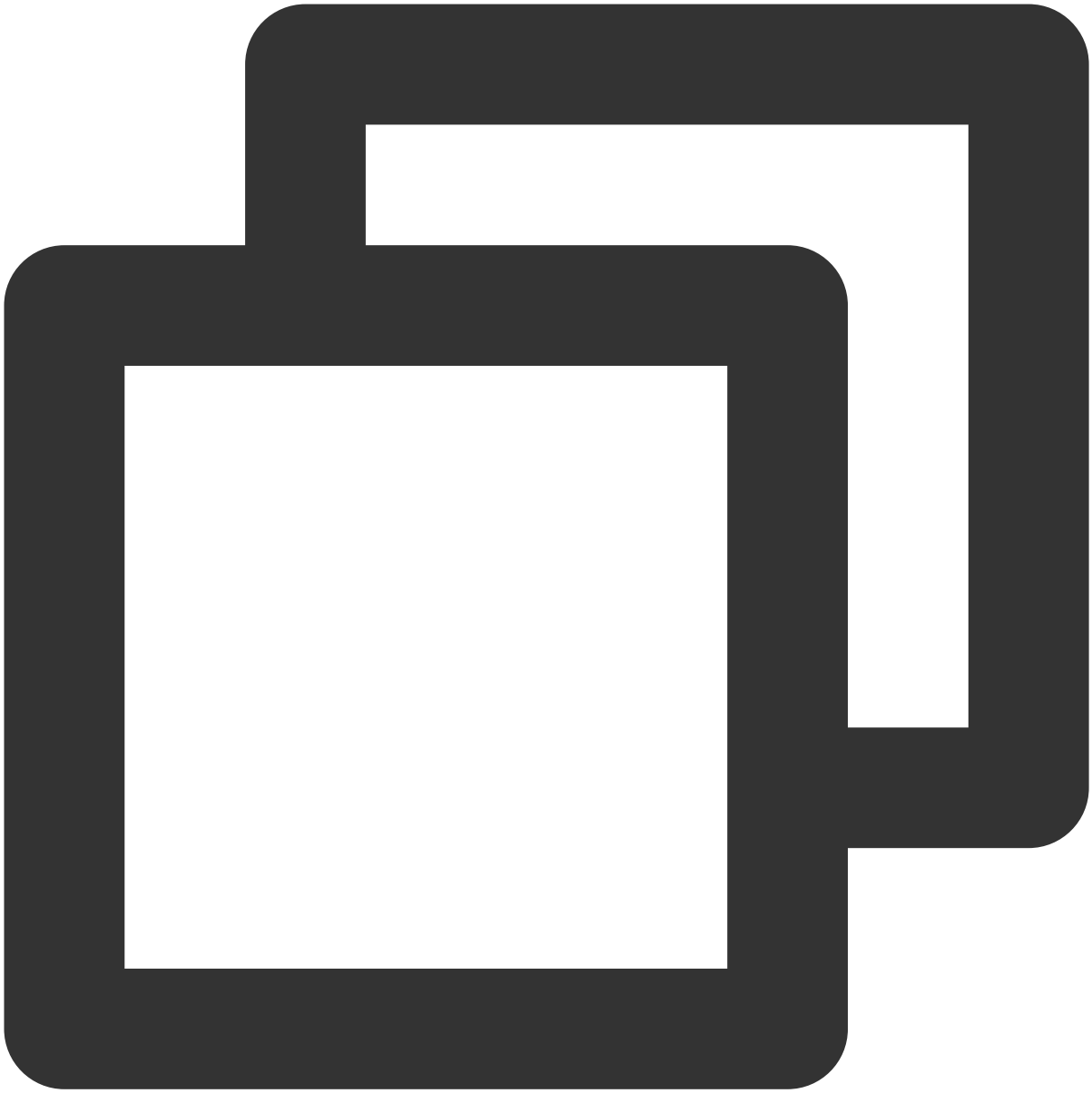
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味

userId	String	指定メンバーのID。
streamType	int	視聴したいビデオストリームのタイプ。詳細については、TRTC SDKをご参照ください。
fillMode	int	画像レンダリングモード。FILL（デフォルト）またはFITモード。詳細については、TRTC SDKをご参照ください。
rotation	int	画像の時計回りの回転角度。詳細については、TRTC SDKをご参照ください。
mirrorType	int	イメージモード。詳細については、 TRTC SDKをご参照ください。

muteRemoteAudio

指定メンバーのリモート音声をミュート/ミュート取り消します。



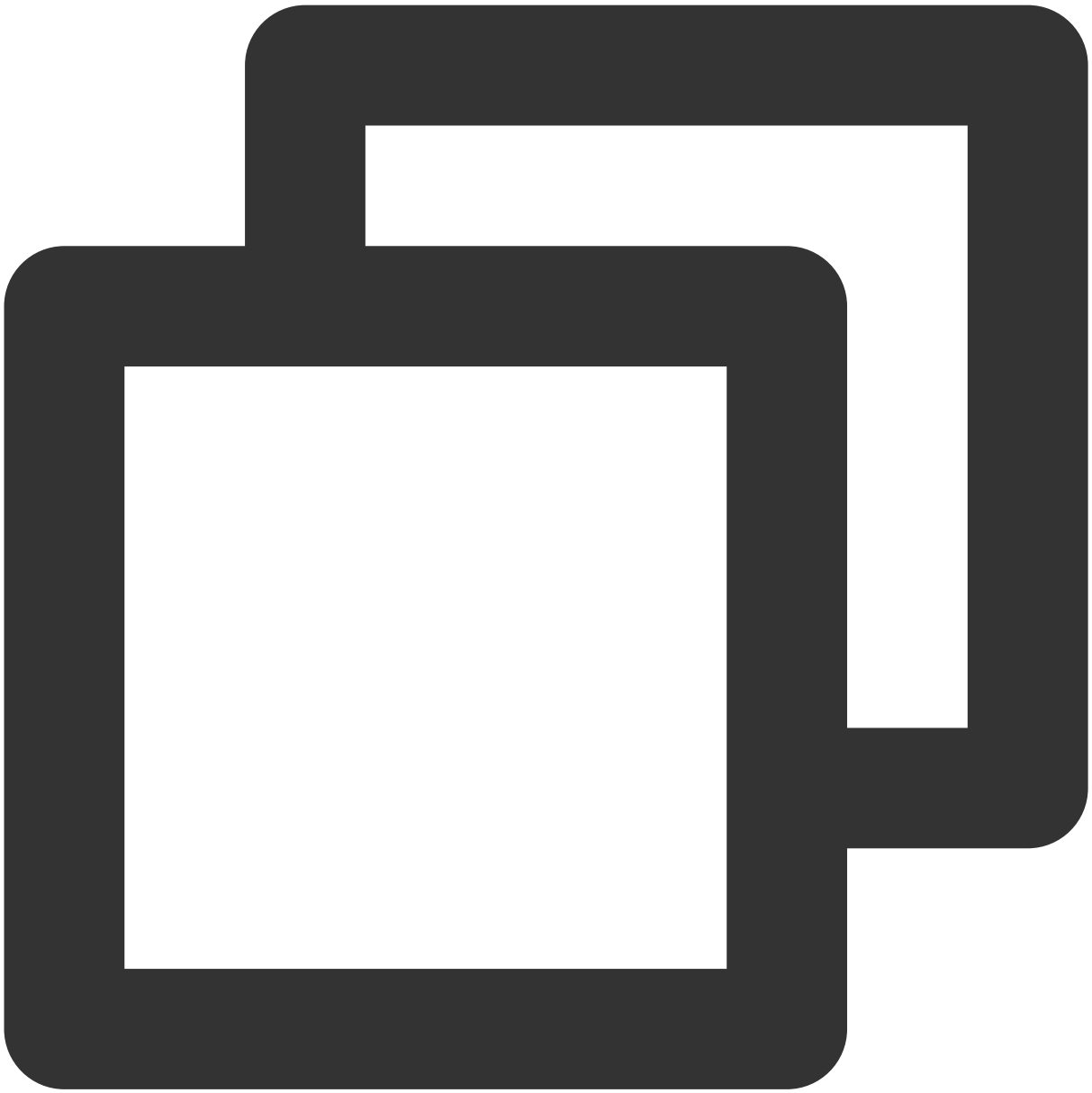
```
Future<void> muteRemoteAudio(String userId, bool mute);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userId	String	指定メンバーのID。
mute	boolean	true：ミュート、false：ミュート停止。

muteAllRemoteAudio

全メンバーのリモート音声をミュート/ミュート取り消します。



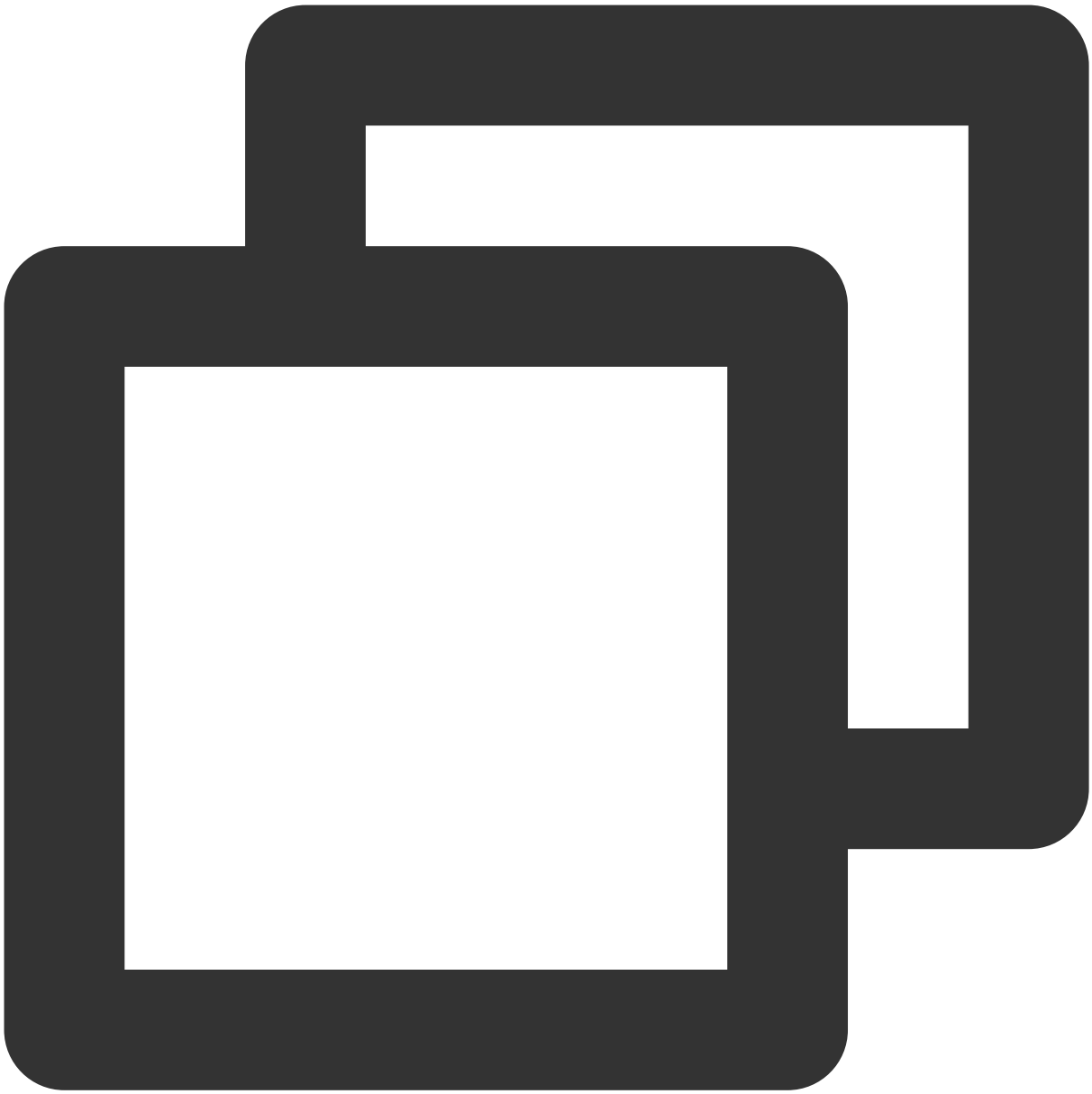
```
Future<void> muteAllRemoteAudio(bool mute);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
mute	boolean	true：ミュート、false：ミュート停止。

muteRemoteVideoStream

指定メンバーのリモートビデオを一時停止/再開します。



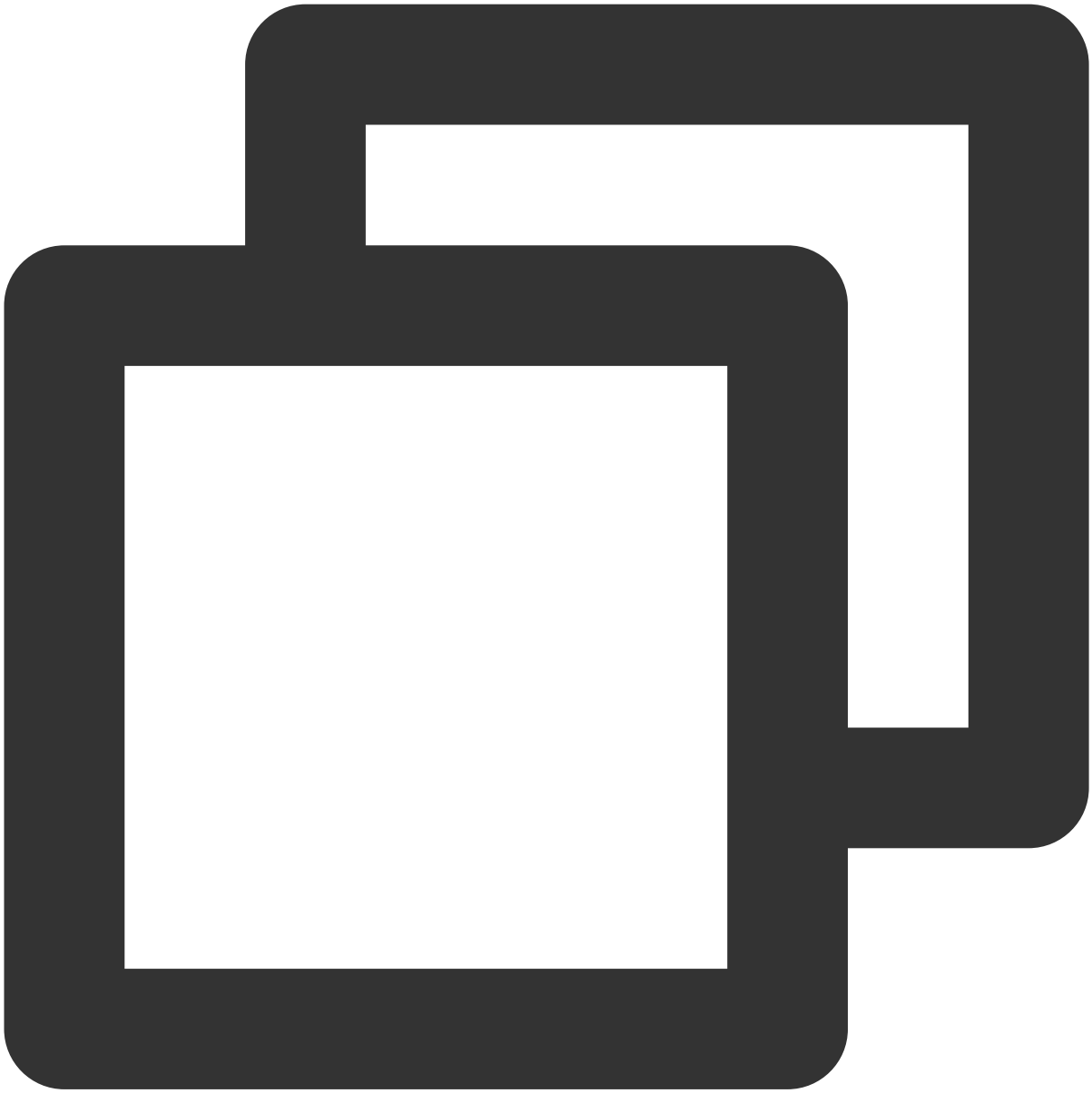
```
Future<void> muteRemoteVideoStream(String userId, bool mute);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userId	String	指定メンバーのID。
mute	boolean	true：一時停止、false：再開。

muteAllRemoteVideoStream

全メンバーのリモートビデオストリームを一時停止/再開します。



```
Future<void> muteAllRemoteVideoStream(bool mute);
```

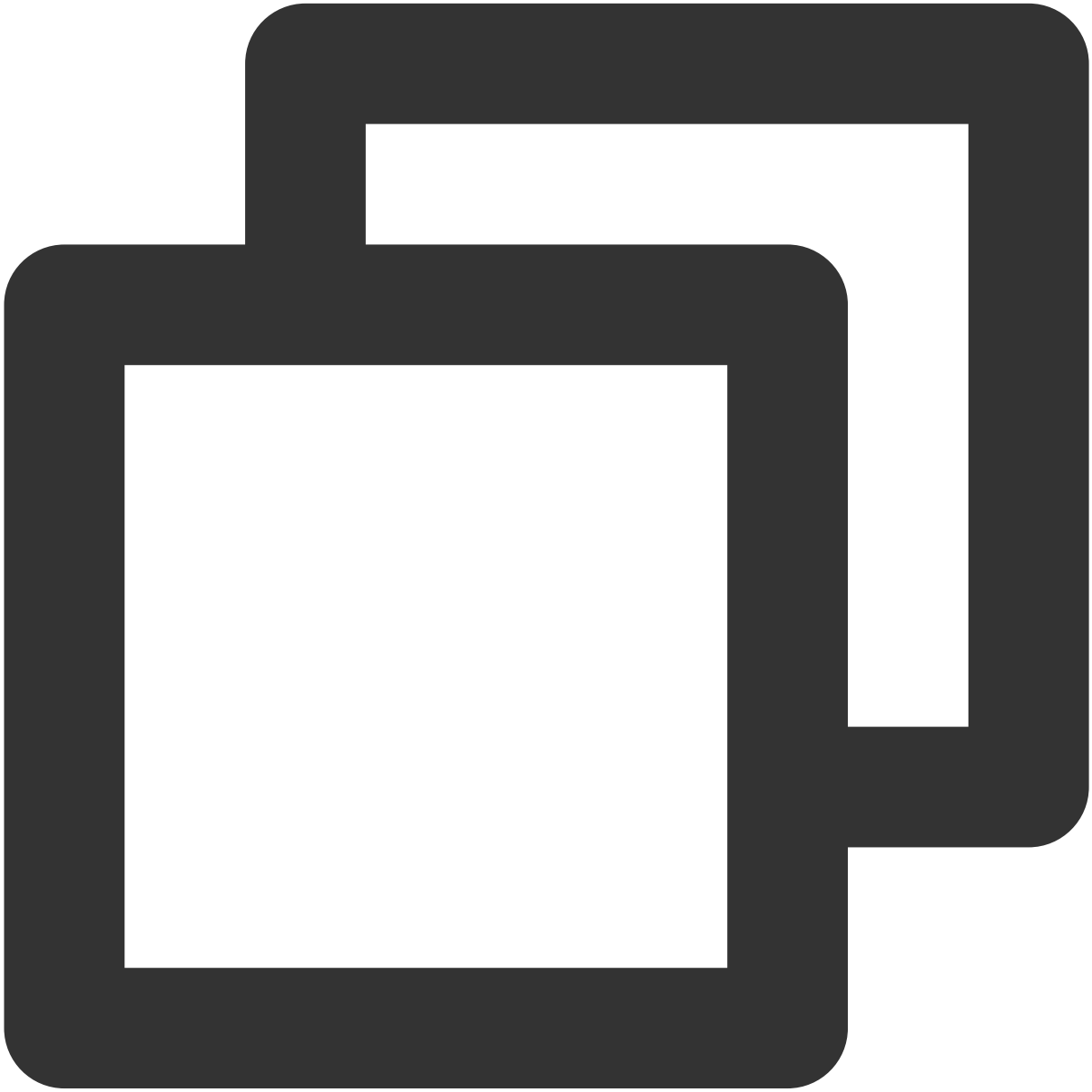
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
mute	boolean	true：一時停止、false：再開。

ローカルのビデオ操作インターフェース

startCameraPreview

ローカルビデオのプレビュー画面を立ち上げます。



```
Future<void> startCameraPreview(bool isFront, int viewId);
```

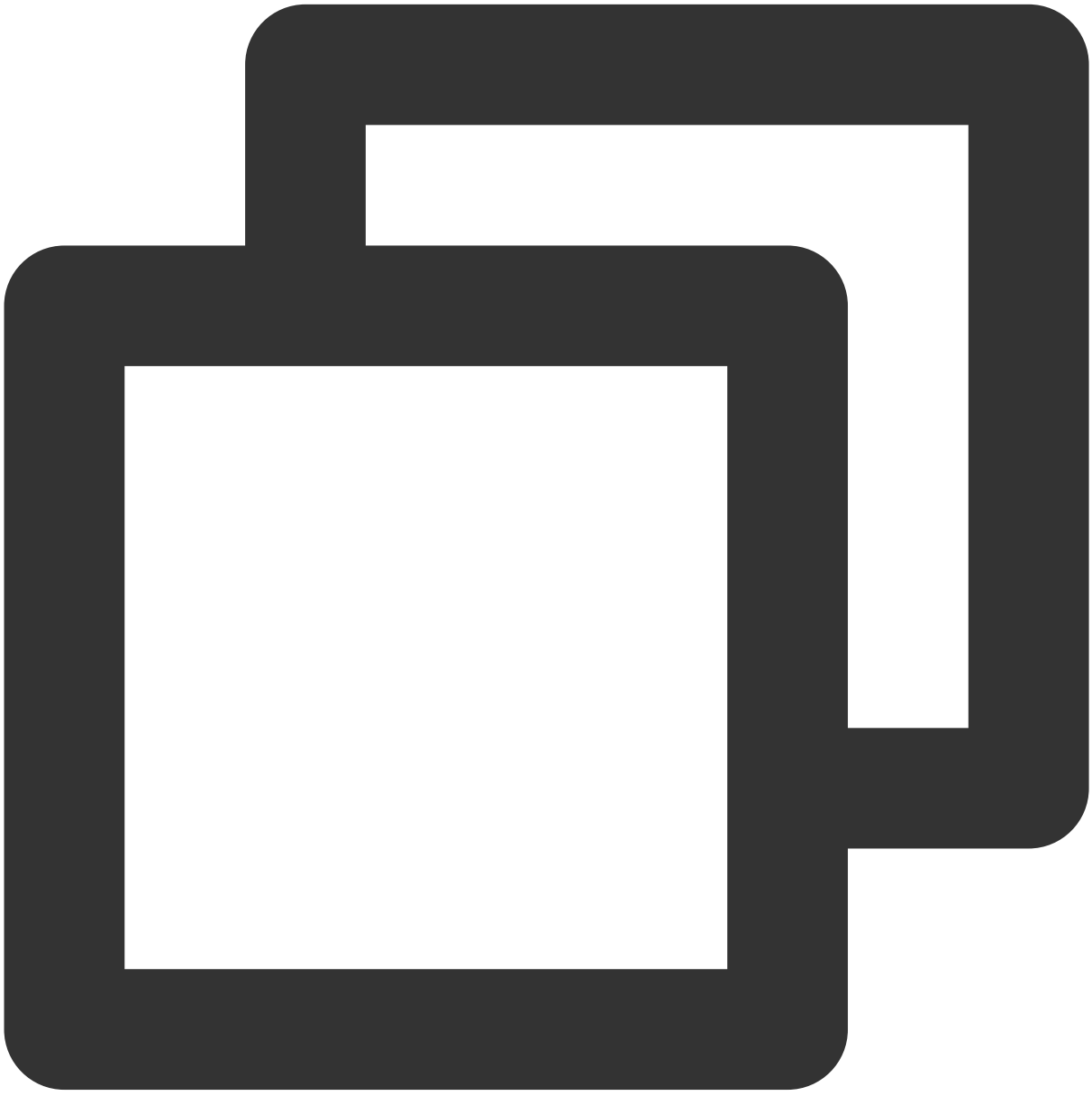
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味

isFront	boolean	true：フロントカメラ、false：リアカメラ。
viewId	int	TRTCCloudVideoViewが発行するviewId。

stopCameraPreview

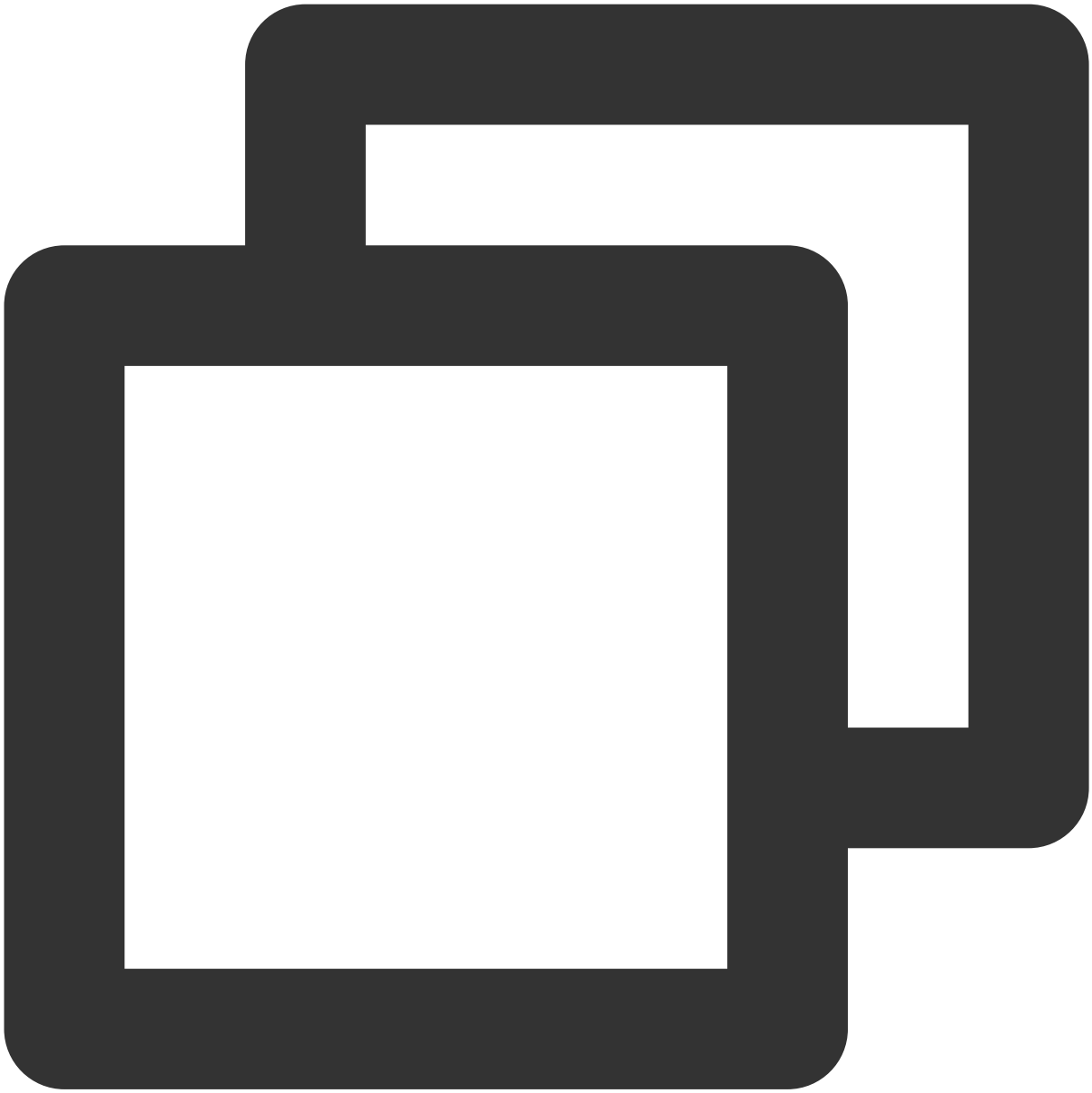
ローカルのビデオキャプチャおよびプレビューを停止します。



```
Future<void> stopCameraPreview();
```

switchCamera

フロント/リアカメラを切り替えます。



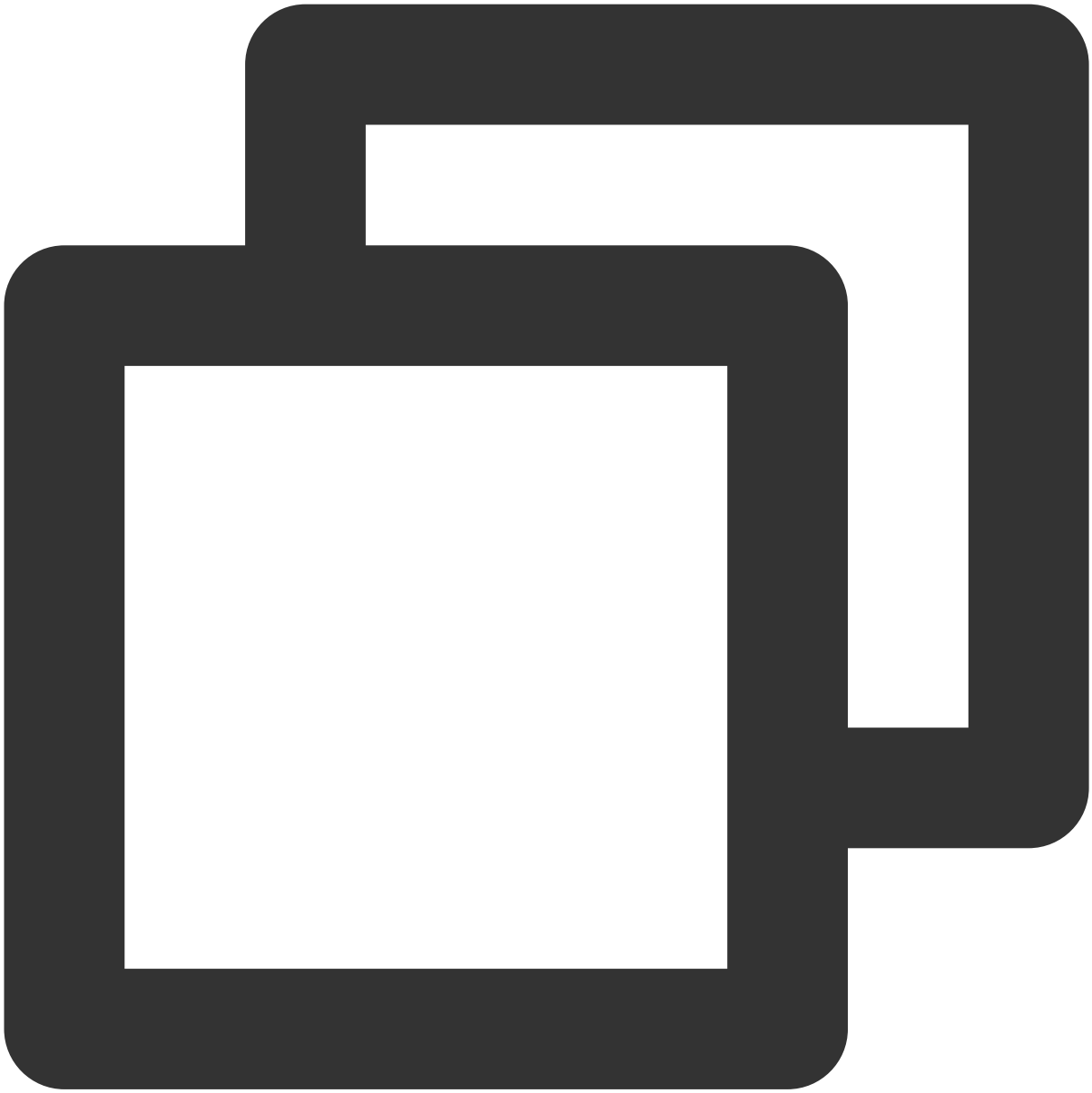
```
Future<void> switchCamera(bool isFront);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
isFront	boolean	true：フロントカメラ、false：リアカメラ。

setVideoEncoderParam

ビデオエンコーダ関連パラメータを設定します。



```
Future<void> setVideoEncoderParam({
  int videoFps,
  int videoBitrate,
  int videoResolution,
  int videoResolutionMode,
});
```

パラメータは下表に示すとおりです：

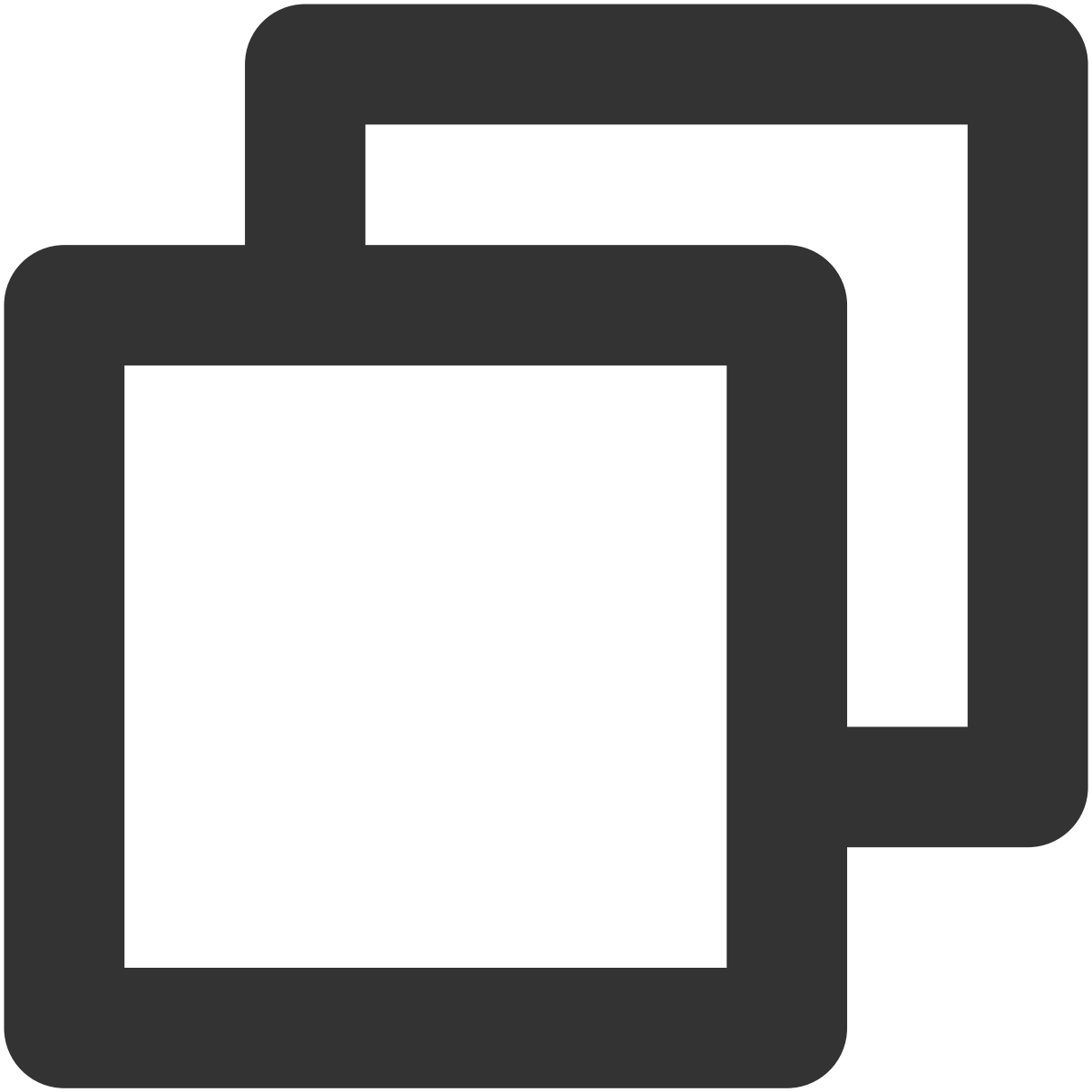
パラメータ	タイプ	意味
videoFps	int	ビデオキャプチャのフレームレート。
videoBitrate	int	ビットレート。SDKは目標ビットレートに応じてエンコードを行い、ネットワークの状態が良くない場合のみ、ビデオビットレートを動的に引き下げます。
videoResolution	int	ビデオ解像度。
videoResolutionMode	int	解像度モード。

説明：

詳細については、TRTC SDKをご参照ください。

setLocalViewMirror

ローカル画面のミラーモードのプレビューを設定します。



```
Future<void> setLocalViewMirror(bool isMirror);
```

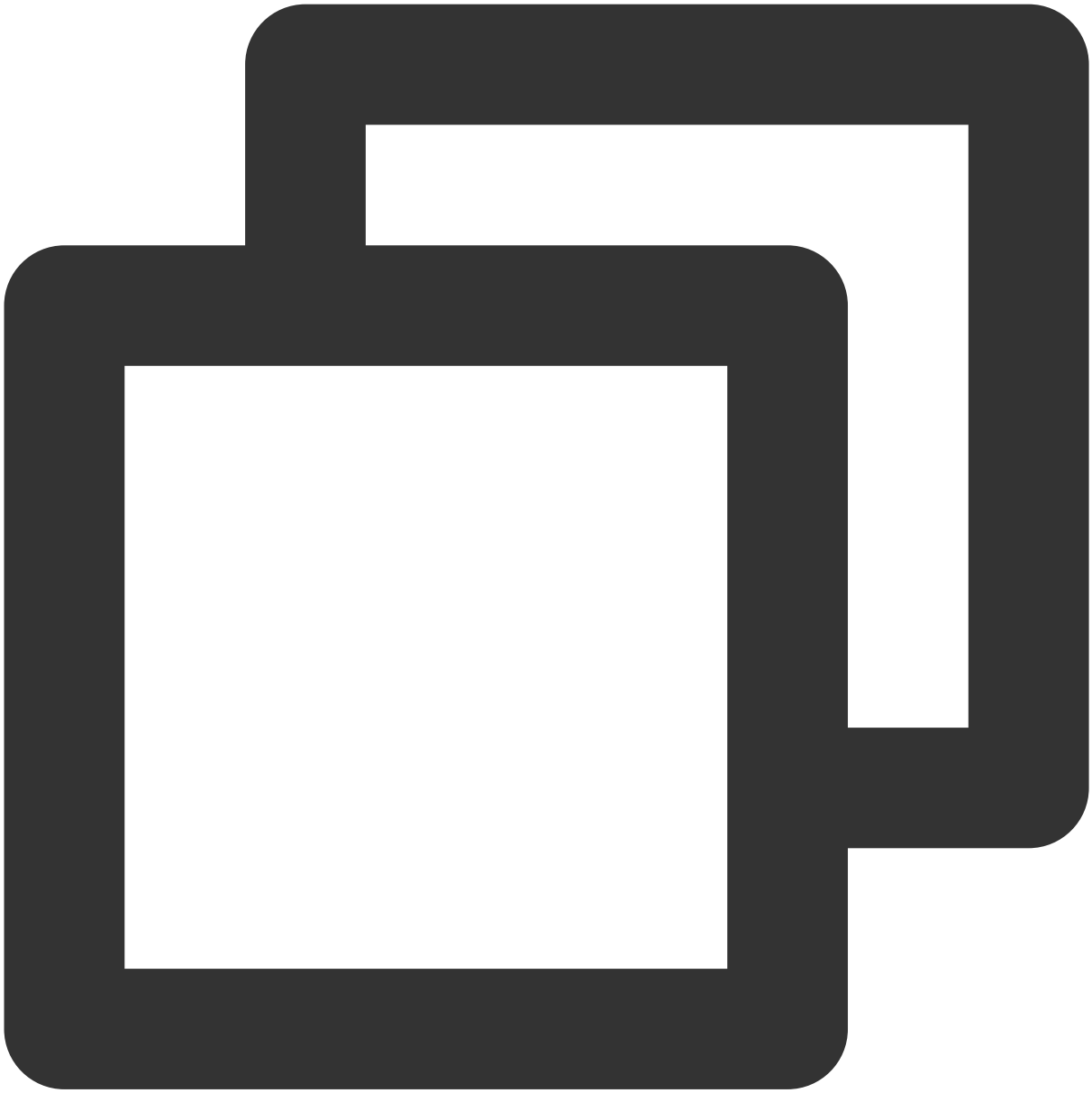
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
isMirror	boolean	イメージプレビューモードの起動の有無。true：オン、false：オフ。

ローカル音声操作のインターフェース

startMicrophone

マイクの集音開始。



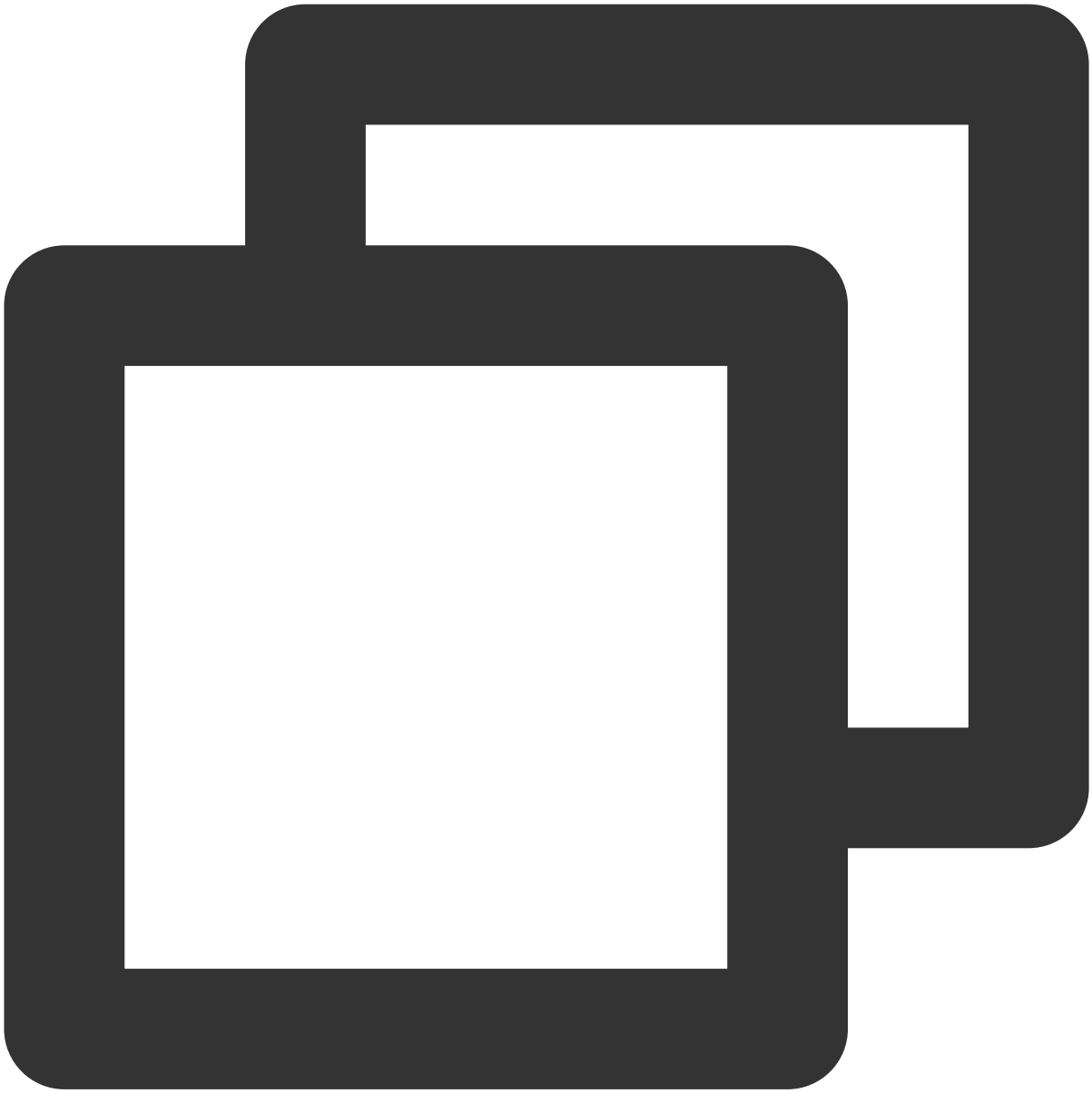
```
Future<void> startMicrophone({int quality});
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
quality	int	オーディオ品質。詳細については、 TRTC SDKをご参照ください。

stopMicrophone

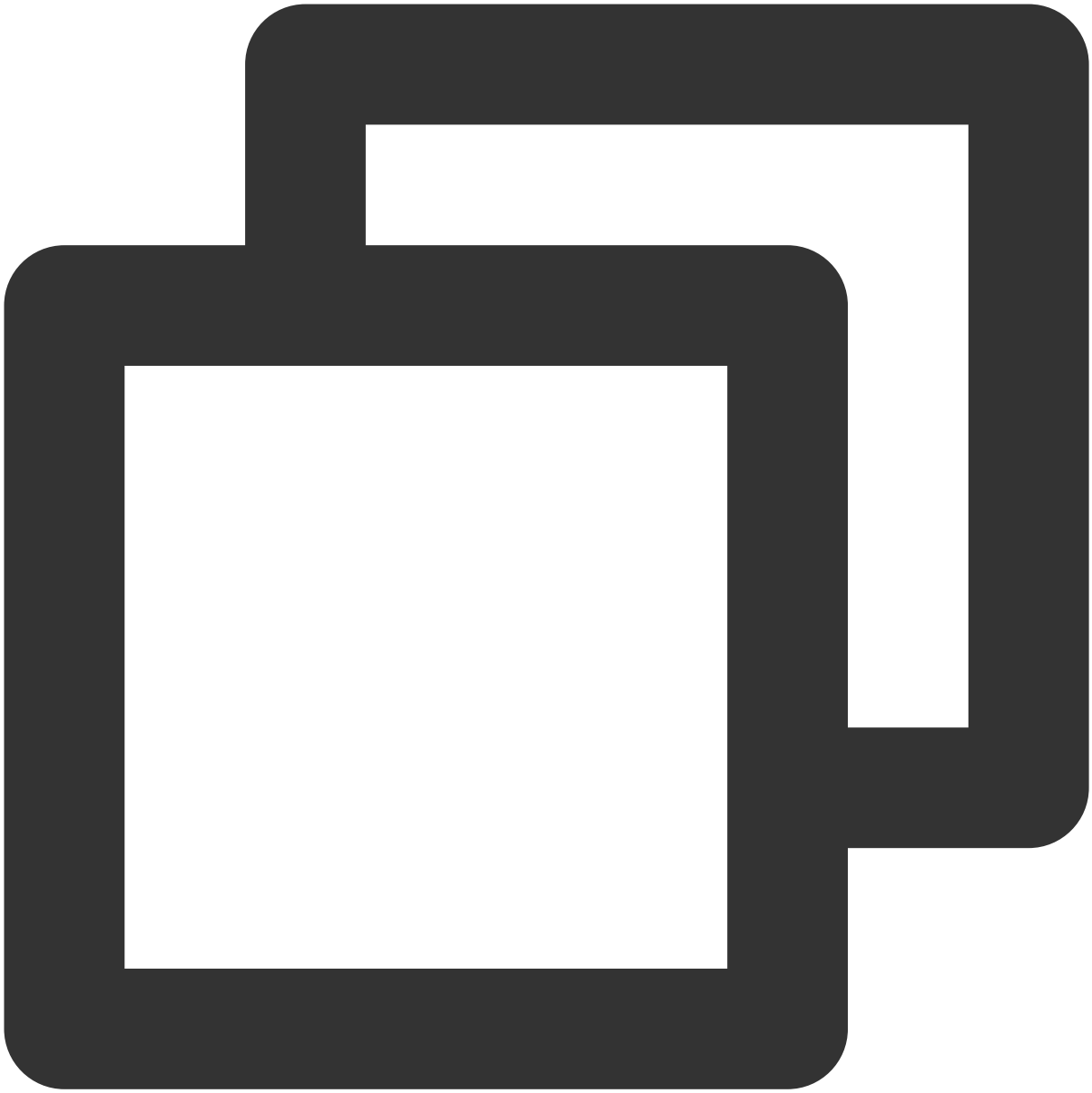
マイクの集音停止。



```
Future<void> stopMicrophone();
```

muteLocalAudio

ローカル音声のミュート起動/停止。



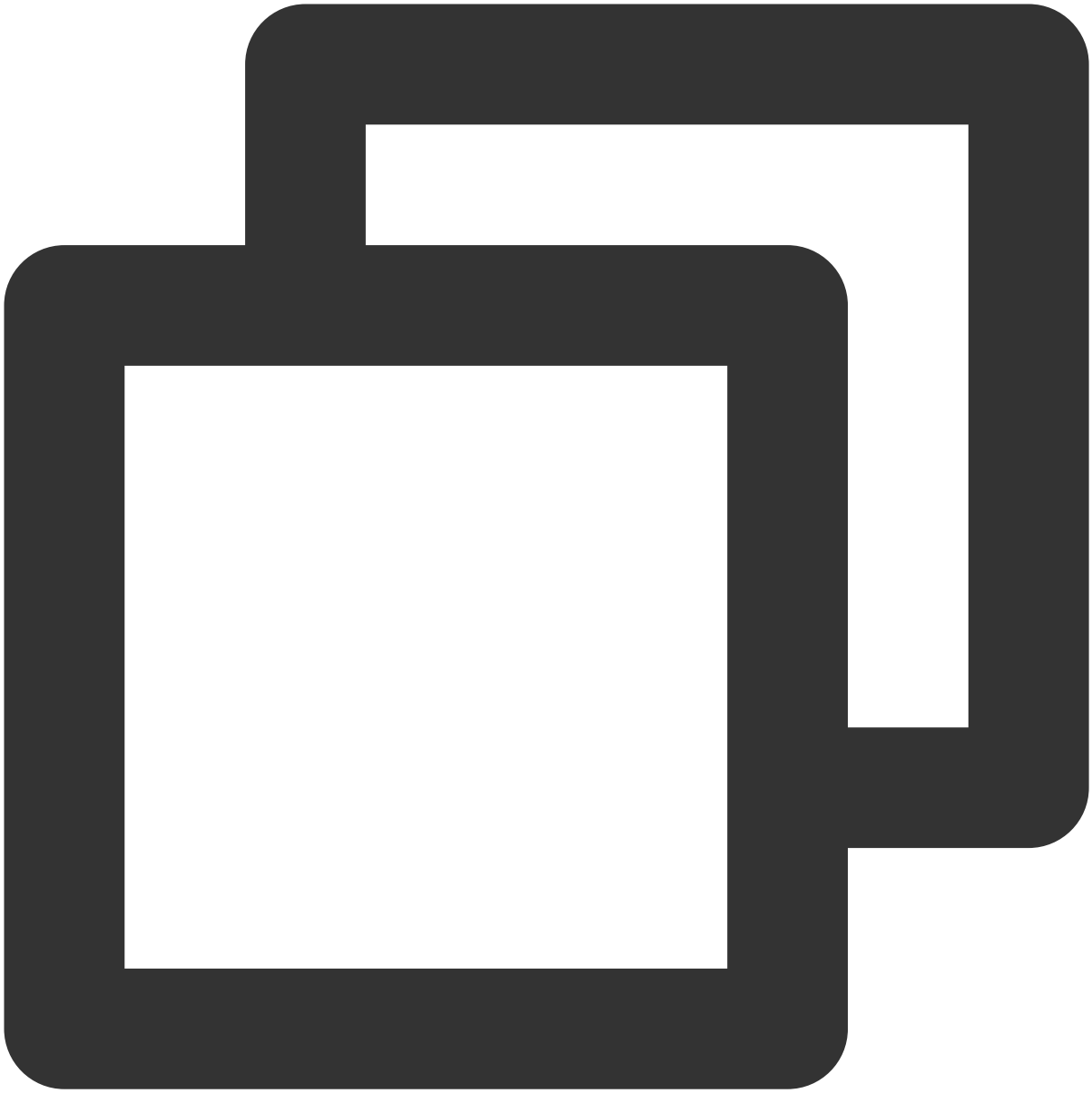
```
Future<void> muteLocalAudio(bool mute);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
mute	boolean	true：ミュート、false：ミュート取り消し。

setSpeaker

スピーカーまたはヘッドホンの起動設定。



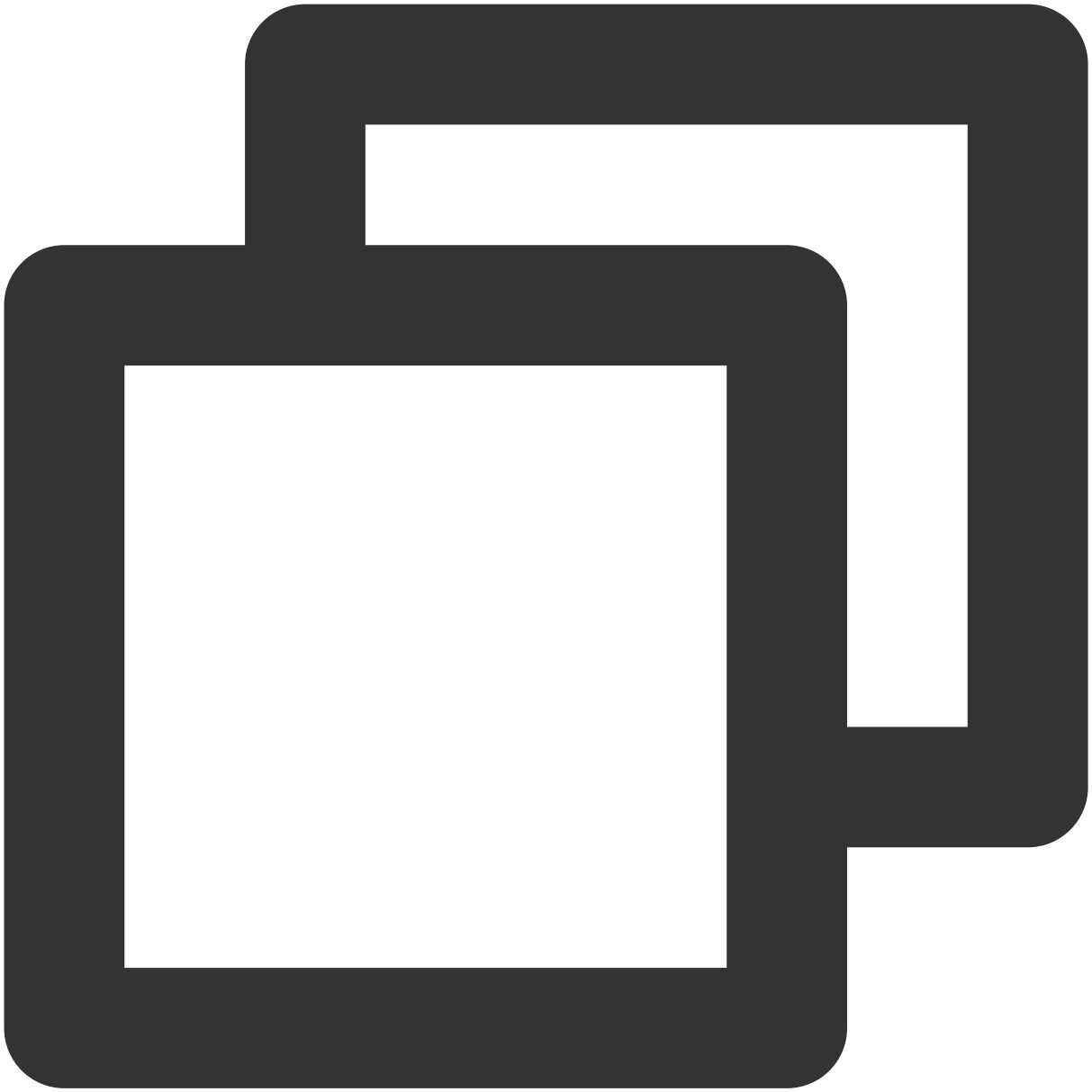
```
Future<void> setSpeaker(bool useSpeaker);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
useSpeaker	boolean	true：スピーカー、false：ヘッドホン。

setAudioCaptureVolume

マイクの集音音量設定。



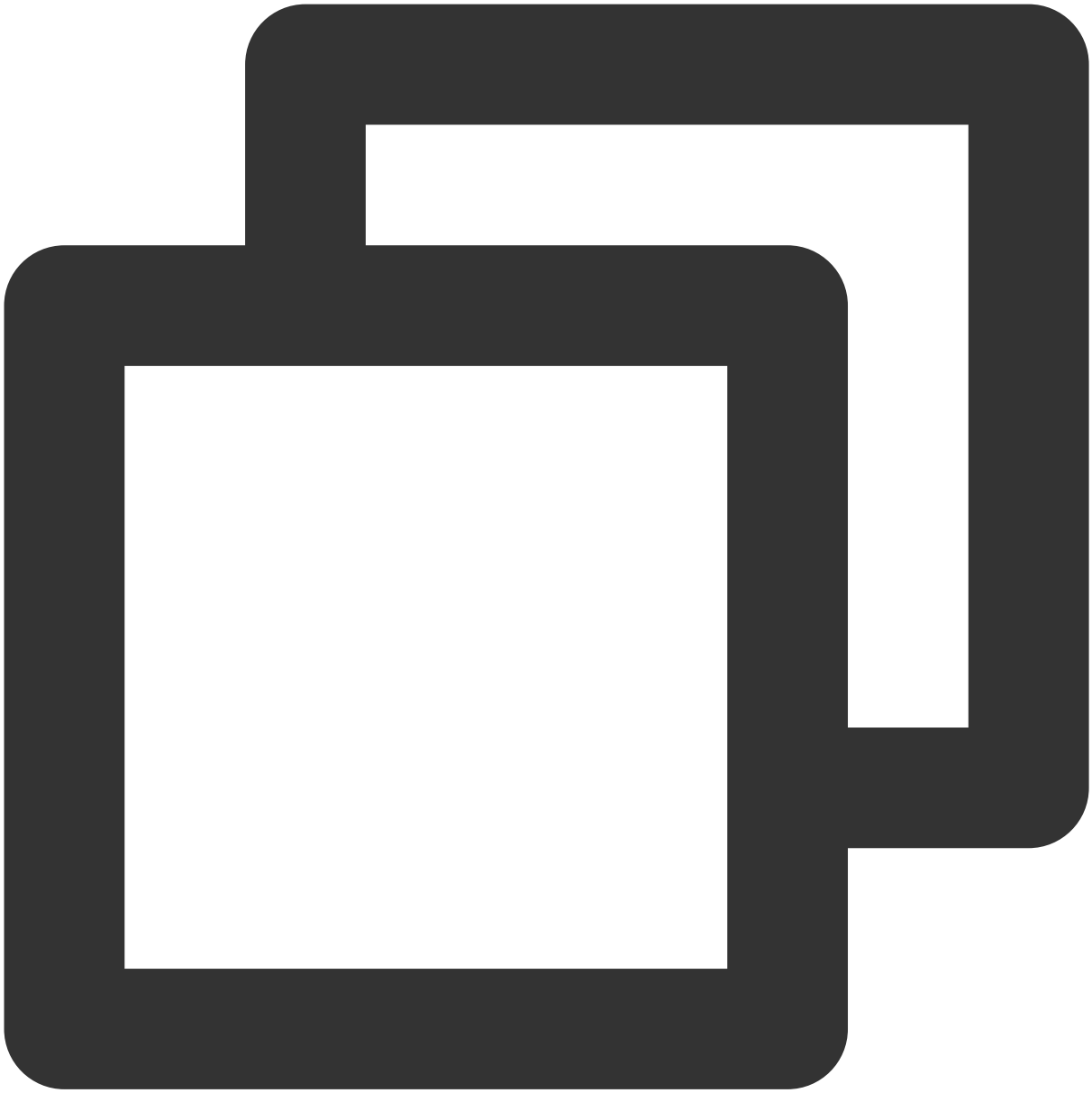
```
Future<void> setAudioCaptureVolume(int volume);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
volume	int	集音音量、値：0 - 100、デフォルト値：100。

setAudioPlayoutVolume

再生音量の設定。



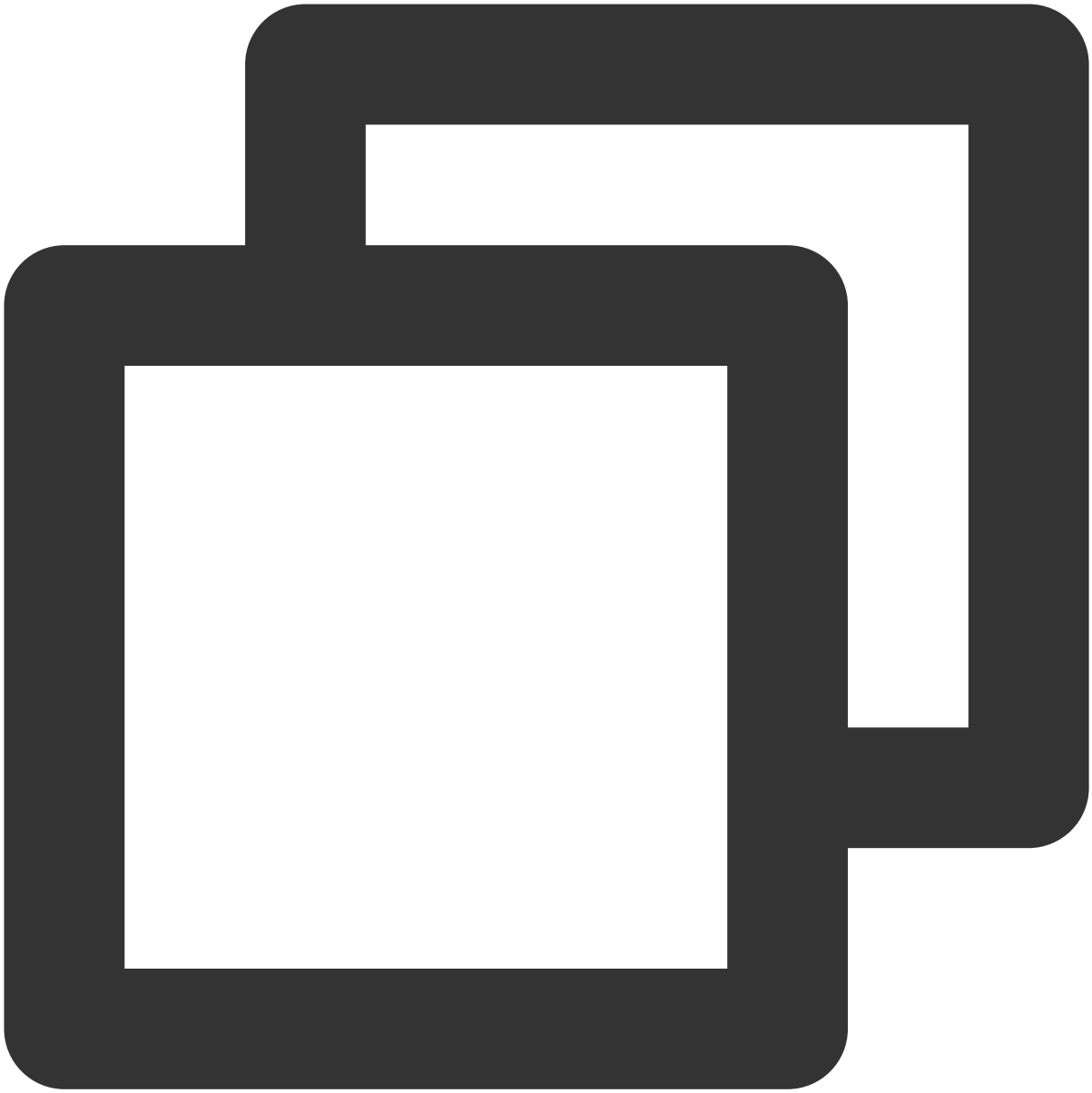
```
Future<void> setAudioPlayoutVolume(int volume);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
volume	int	再生音量、値：0 - 100、デフォルト値：100。

startAudioRecording

録音の開始。



```
Future<int?> startAudioRecording(String filePath);
```

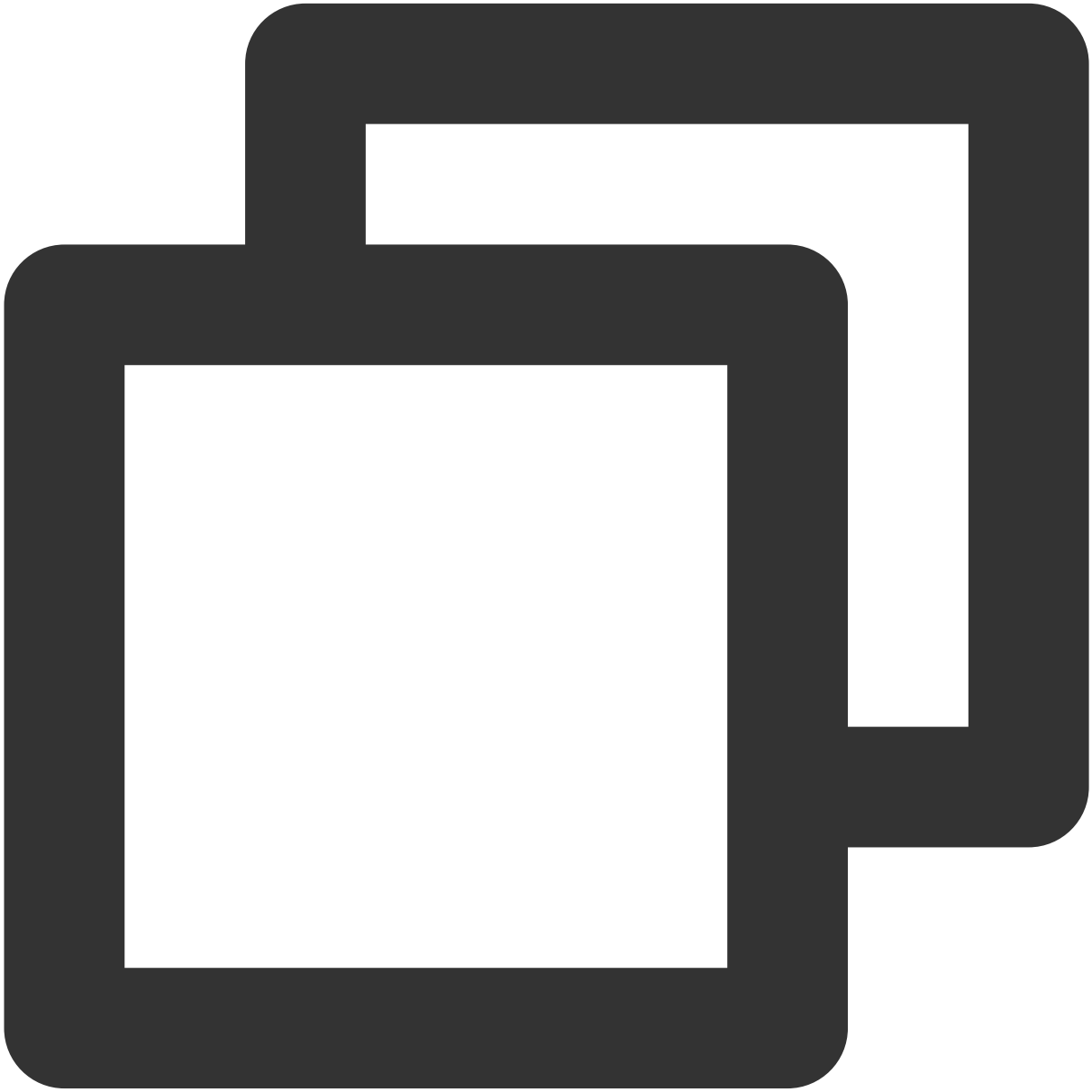
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
filePath	String	録音ファイルの保存パス。このパスはユーザーが自身で指定する必要があります。パスが存在し、記述できることを確認してください。このパスはファイル名およびフォーマットの拡張子まで正確に示す必要があります。フォーマットの拡張子に

		よって録音ファイルの形式が決まります。現在サポートしている形式はPCM、WAV、AACです。

stopAudioRecording

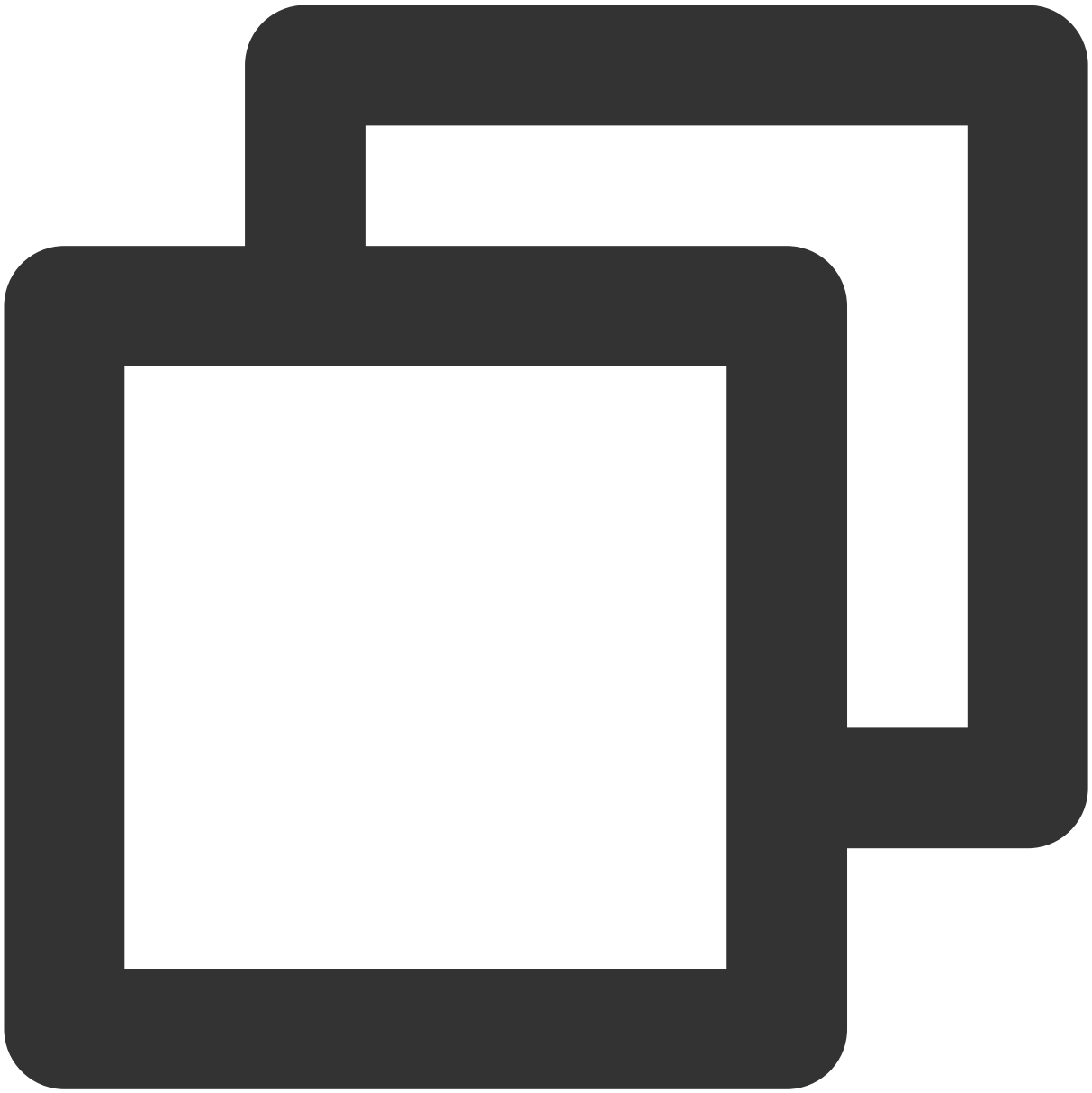
録音の停止。



```
Future<void> stopAudioRecording();
```

enableAudioVolumeEvaluation

音量レベルリマインダを有効にします。



```
Future<void> enableAudioVolumeEvaluation(int intervalMs);
```

パラメータは下表に示すとおりです：

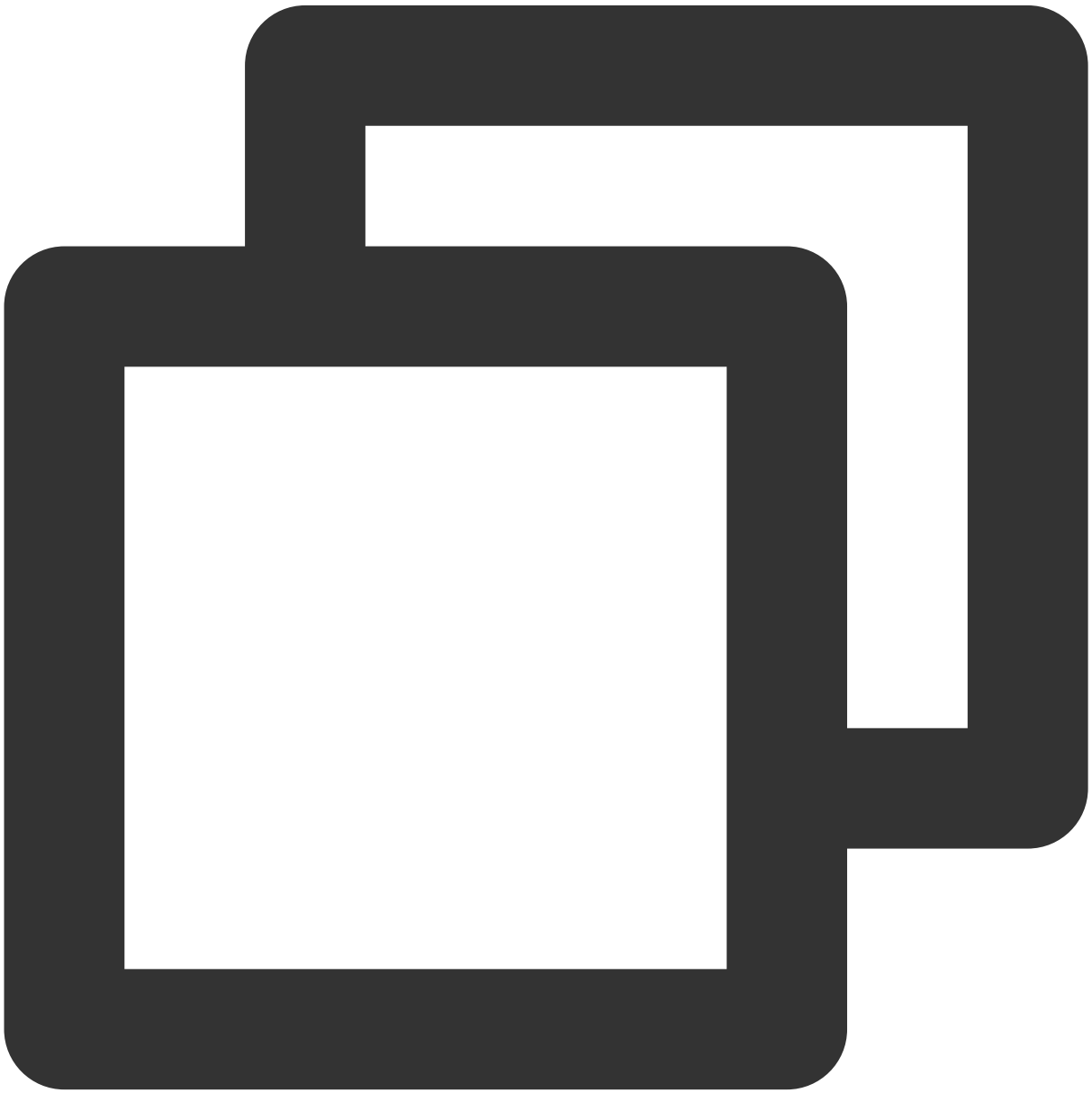
パラメータ	タイプ	意味
intervalMs	int	onUserVoiceVolume コールバックをトリガーするインターバルを決定します。単位はms。最小インターバルは100ms。0以下の場合、コールバックは停止します。設定を

		300msにすることを推奨します。

スクリーンキャプチャのインターフェース

startScreenCapture

画面共有を開始。



```
Future<void> startScreenCapture({
  int videoFps,
```

```
    int videoBitrate,
    int videoResolution,
    int videoResolutionMode,
    String appGroup,
});
```

パラメータは下表に示すとおりです：

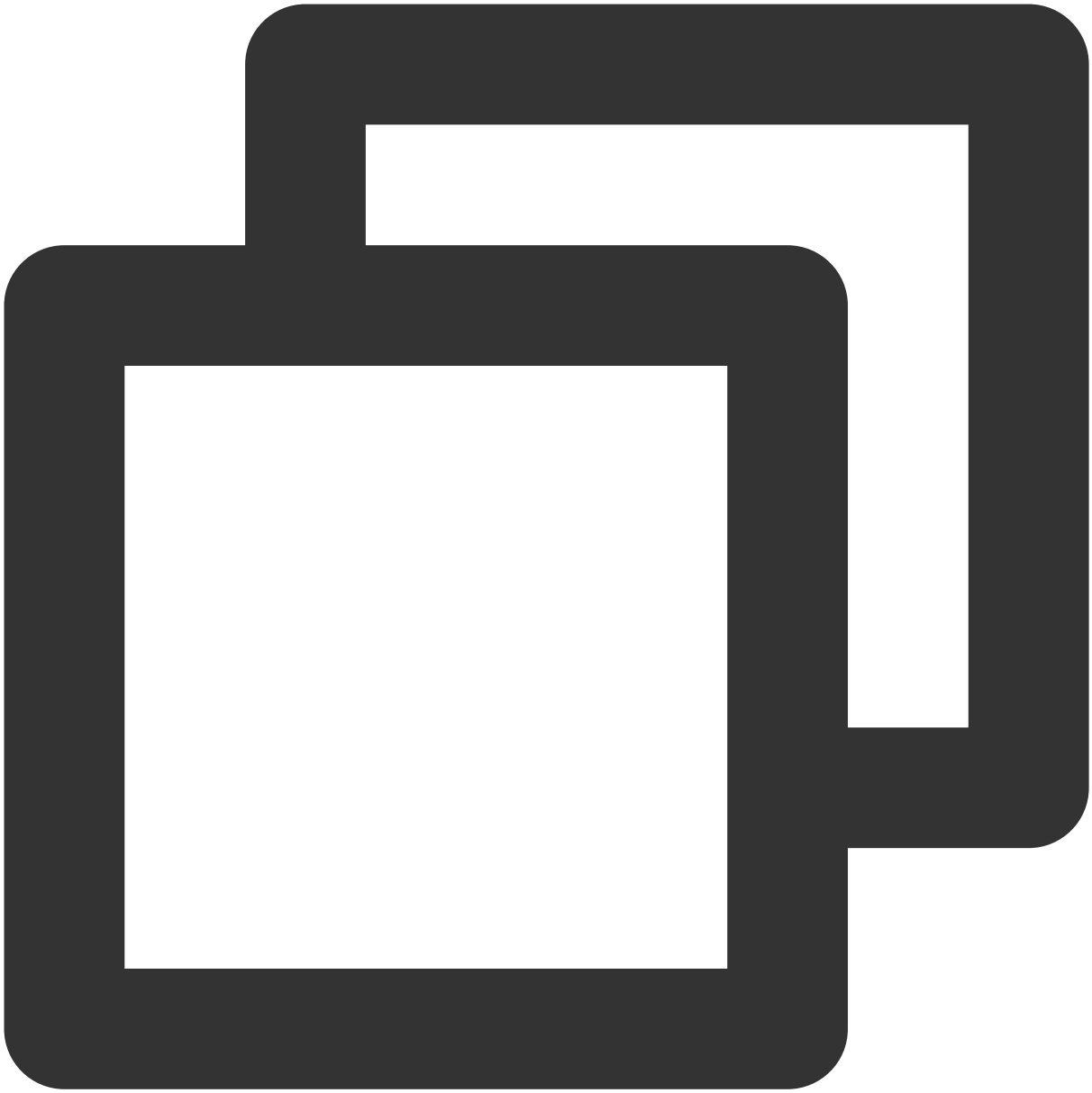
パラメータ	タイプ	意味
videoFps	int	ビデオキャプチャのフレームレート。
videoBitrate	int	ビットレート。SDKは目標ビットレートに応じてエンコードを行い、ネットワークの状態が良くない場合のみ、ビデオビットレートを動的に引き下げます。
videoResolution	int	ビデオ解像度。
videoResolutionMode	int	解像度モード。
appGroup	String	このパラメータはiOS端末でのみ有効となり、Android端末の場合、このパラメータを気にする必要はありません。このパラメータはメインAppとBroadcastが共有するApplication Group Identifierです。

説明：

詳細については、TRTC SDKをご参照ください。

stopScreenCapture

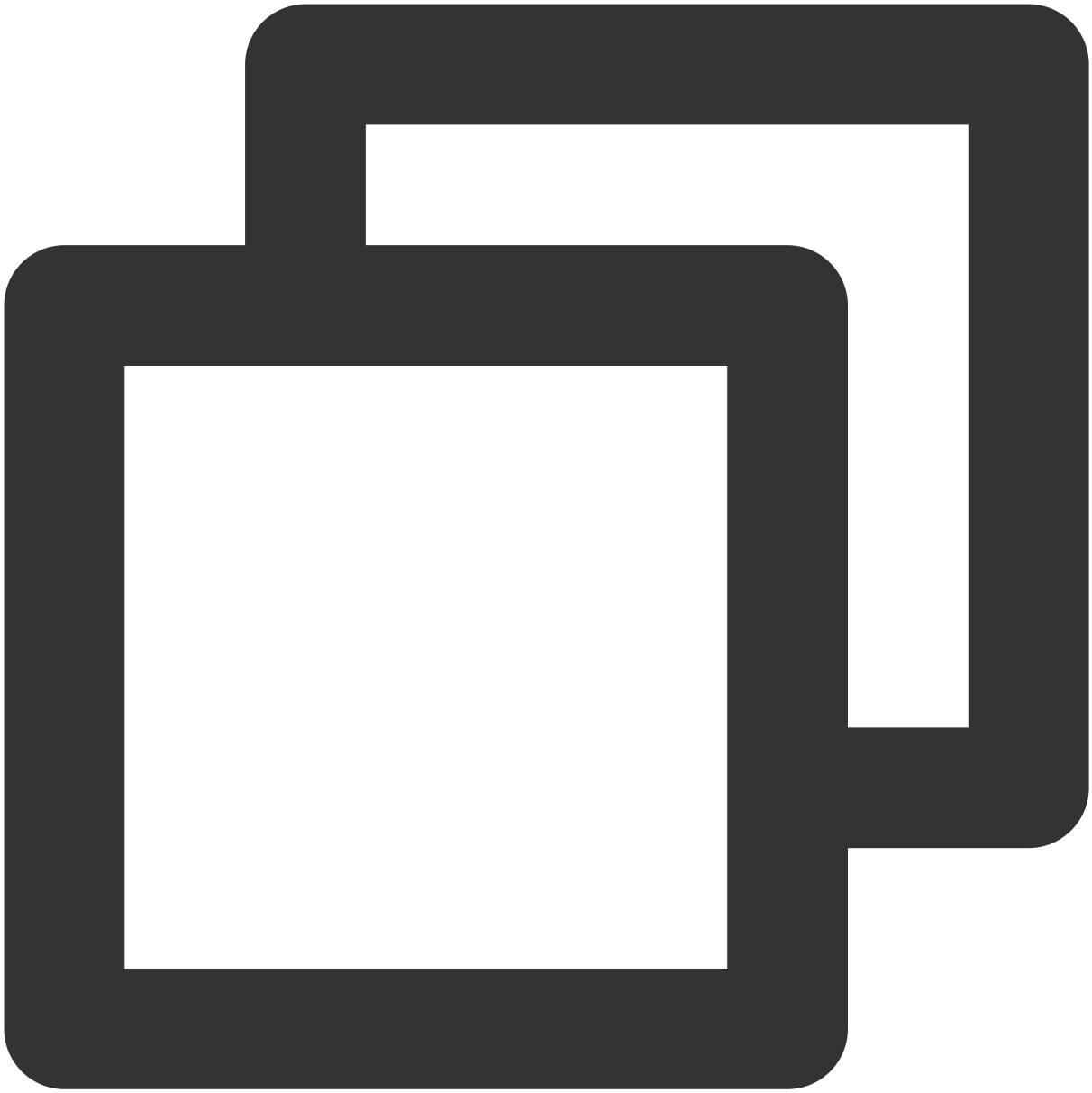
画面キャプチャの停止。



```
Future<void> stopScreenCapture();
```

pauseScreenCapture

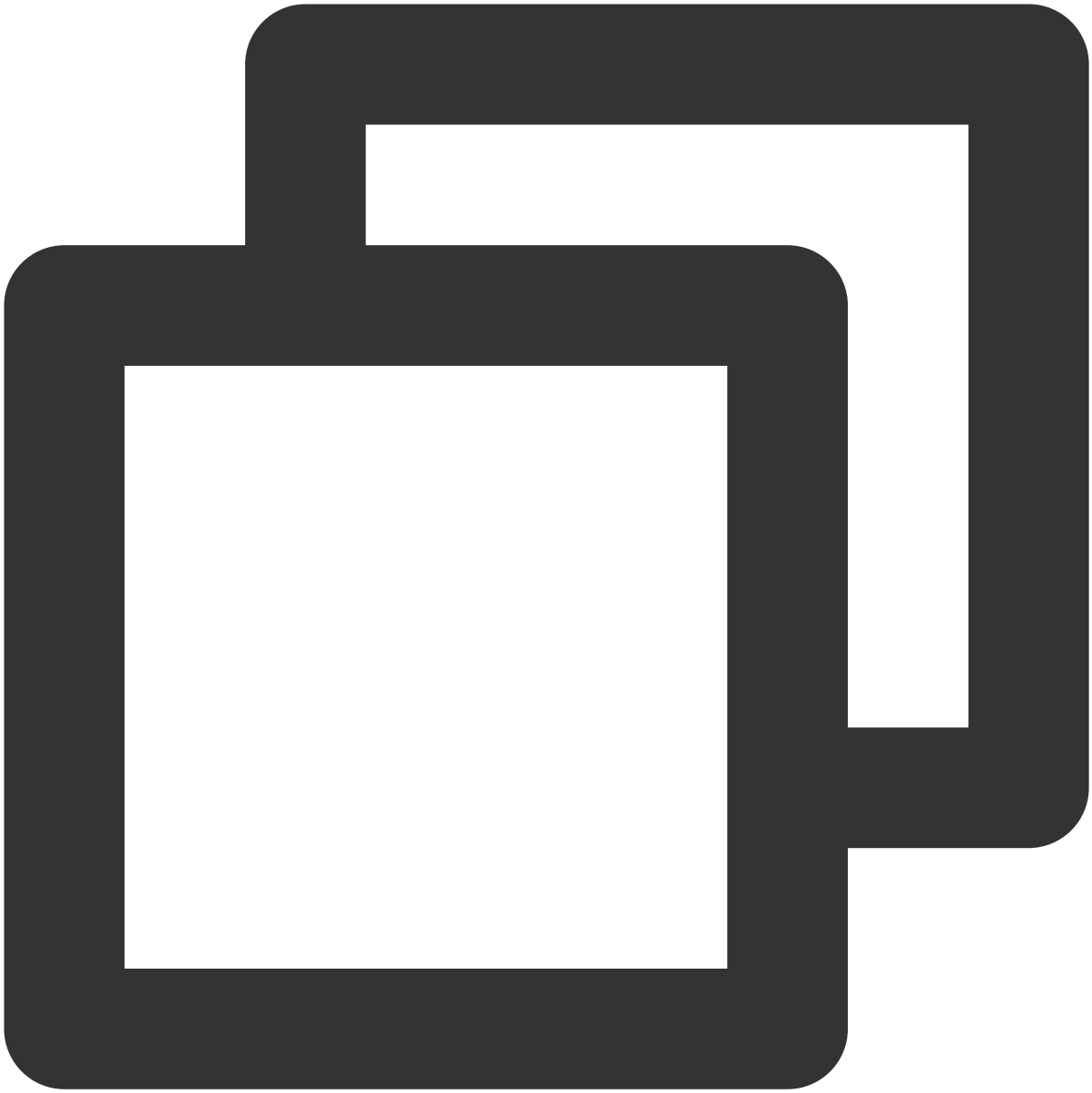
スクリーンキャプチャの一時停止。



```
Future<void> pauseScreenCapture();
```

resumeScreenCapture

スクリーンキャプチャの再開。

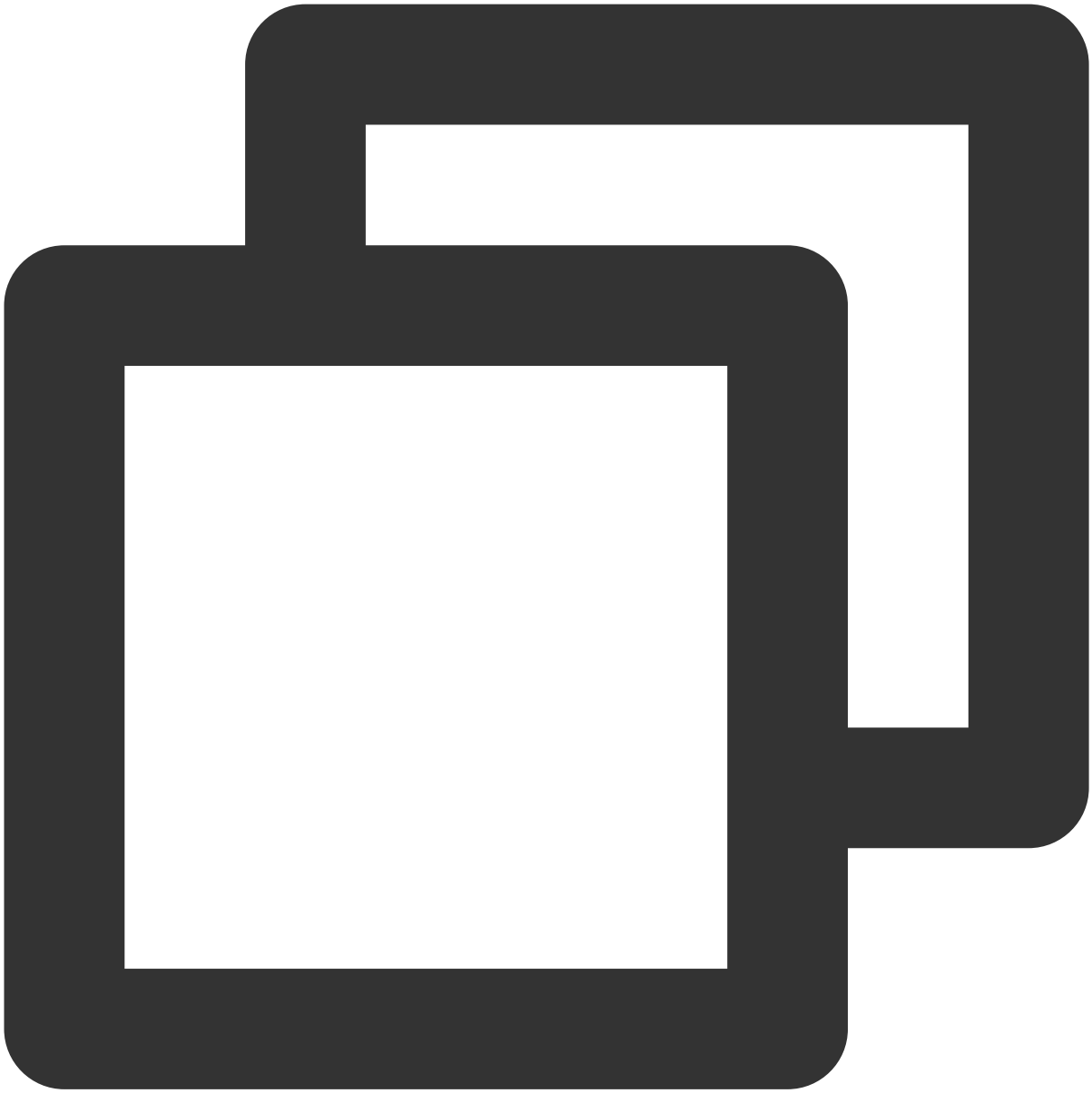


```
Future<void> resumeScreenCapture();
```

管理オブジェクト取得に関するインターフェース

getDeviceManager

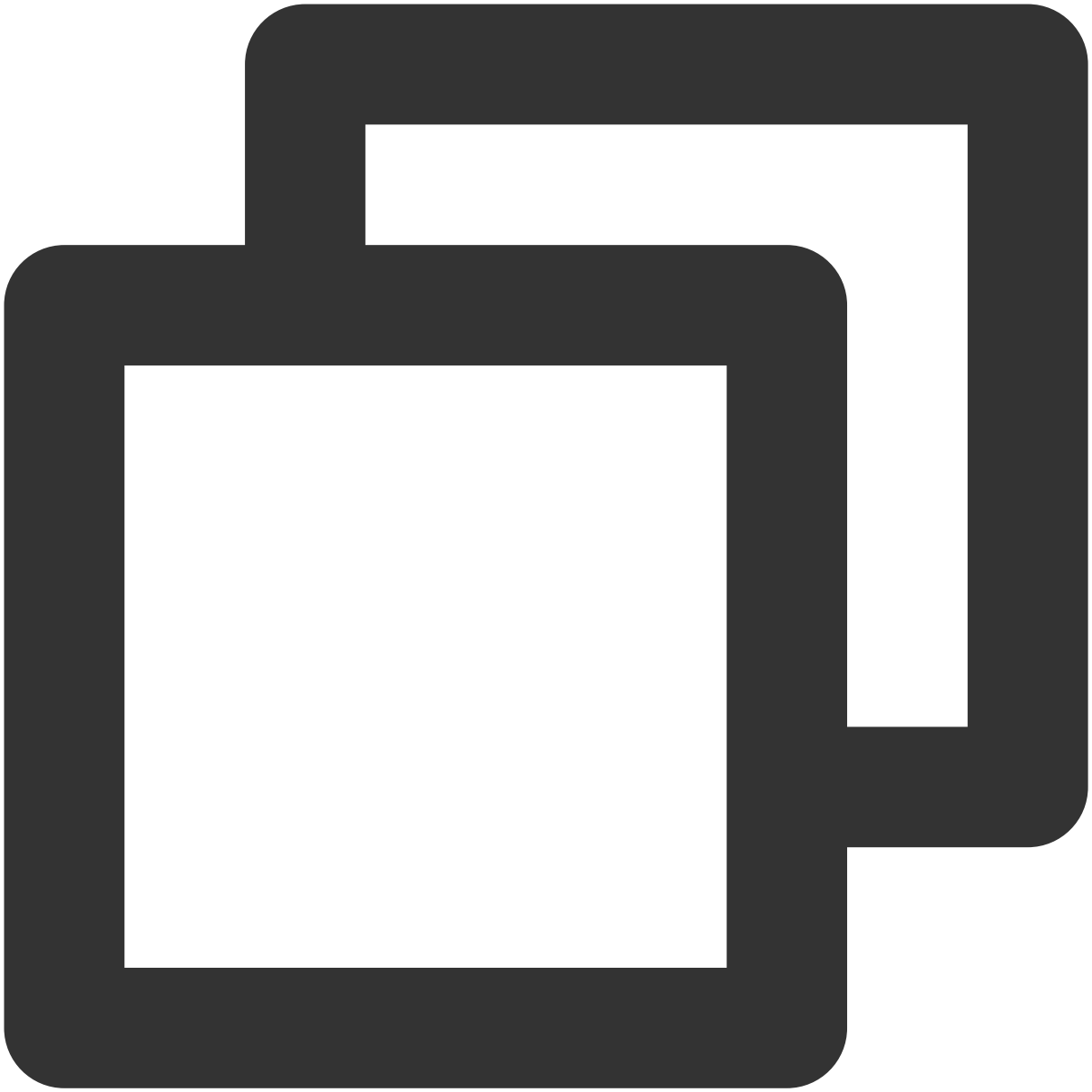
デバイス管理オブジェクト TXDeviceManagerを取得します。



```
getDeviceManager();
```

getBeautyManager

美顔管理オブジェクトTXBeautyManagerを取得します。 |



```
getBeautyManager();
```

美顔管理では、次の機能を使用できます。

「美顔のスタイル」、「美白」、「肌色補正（血色・つや感）」、「デカ眼」、「顔痩せ」、「V顔」、「下あご」、「面長補正」、「小鼻」、「キラキラ目」、「白い歯」、「目の弛み除去」、「シワ除去」、「ほうれい線除去」などの美容効果を設定します。

「髪の生え際」、「眼と眼の距離」、「眼角度」、「唇の形」、「鼻翼」、「鼻の位置」、「唇の厚さ」、「顔の形」を調整します。

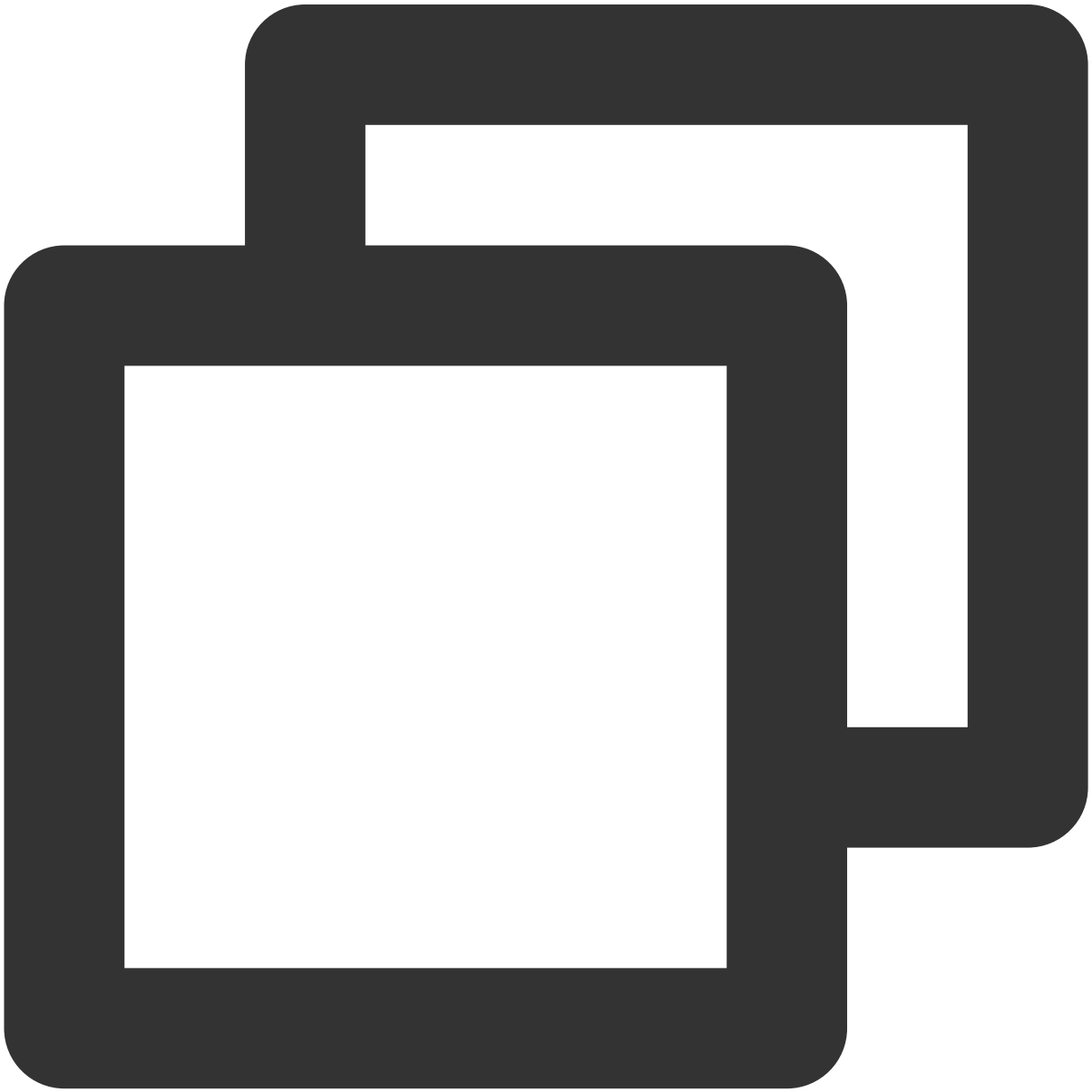
人の顔のスタンプ（素材）等のダイナミック効果を設定します。

メイクアップを追加します。
ジェスチャー認識を行います。

メッセージ送信関連インターフェース

sendRoomTextMsg

ミーティング中にテキストメッセージをブロードキャストします。通常、チャットに使用します。



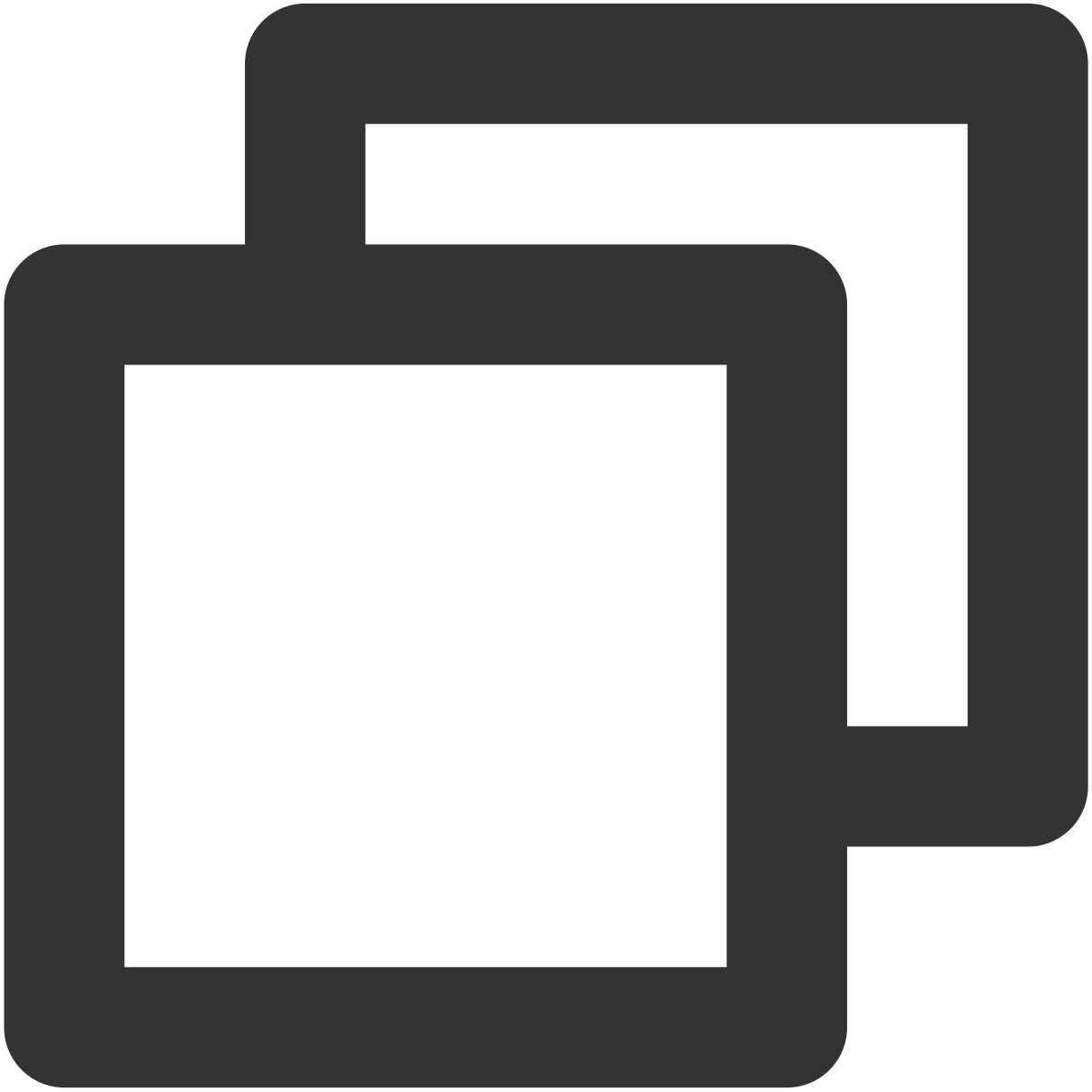
```
Future<ActionCallback> sendRoomTextMsg(String message);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
message	String	テキストメッセージ。

sendRoomCustomMsg

カスタマイズしたテキストメッセージを送信します。



```
Future<ActionCallback> sendRoomCustomMsg(String cmd, String message);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
cmd	String	コマンドワードです。開発者がカスタマイズするものであり、主にさまざまなメッセージタイプを区別するために使用されます。
message	String	テキストメッセージ。

TRTCMeetingDelegateイベントコールバック

一般的なイベントコールバック

onError

エラーのコールバック。

説明：

SDKリカバリー不能なエラーは必ず監視し、状況に応じてユーザーに適切なインターフェースプロンプトを表示します。

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
errCode	int	エラーコード。
errMsg	String	エラー情報。
extraInfo	String	拡張情報フィールド。個別のエラーコードにより問題の特定に役立つ追加情報をもたせることが可能です。

onWarning

警告のコールバック。

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
warningCode	int	エラーコード。
warningMsg	String	警告情報。

extraInfo	String	拡張情報フィールド。個別のエラーコードにより問題の特定に役立つ追加情報をもたせることが可能です。

onKickedOffline

他のユーザーが同じアカウントでログインすると、キックアウトされてオフラインになります。

ミーティングルームのイベントコールバック

onRoomDestroy

ミーティングルームが破棄された時のコールバック。キャスターの退室時、ルーム内の全ユーザーがこの通知を受信します。
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
roomId	String	ミーティングルームID。

onNetworkQuality

ネットワーク状態のコールバック。
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
localQuality	TRTCCloudDef.TRTCQuality	アップストリームのネットワーク品質。
remoteQuality	List<TRTCCloudDef.TRTCQuality>	ダウンストリームのネットワーク品質。

onUserVolumeUpdate

ユーザーの通話音量のコールバック。
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userVolumes	List	発言中のすべてのルーム参加者の音量。値の範囲：0～100。
totalVolume	int	すべてのリモート参加者の総音量。値の範囲：0～100。

参加者入退室のイベントコールバック

onEnterRoom

ローカルの入室のコールバック。

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
result	int	0を上回るときは入室の所要時間(ms)、0未満のときは入室のエラーコードです。

onLeaveRoom

ローカルの退室のコールバック。

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
reason	int	ミーティング退出の原因。0：自主的にleaveMeetingを呼び出して退出。1：サーバーによる現在のミーティングからのキックアウト。2：現在のミーティング全体の解散。

onUserEnterRoom

新しい参加者の入室のコールバック。

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userId	String	新しい参加者のユーザーID。

onUserLeaveRoom

参加者の退出のコールバック。

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userId	String	退出した参加者のユーザーID。

参加者の音声・ビデオのイベントコールバック

onUserAudioAvailable

参加者のマイク起動/停止のコールバック。

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userId	String	ユーザーID。
available	boolean	true：ユーザーがマイクをオンにします。false：ユーザーがマイクをオフにします。

onUserVideoAvailable

参加者によるカメラ起動/停止のコールバック。
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userId	String	ユーザーID。
available	boolean	true：ユーザーがカメラをオンにします。false：ユーザーがカメラをオフにします。

onUserSubStreamAvailable

参加者がサブチャンネル画面を起動/停止したときのコールバック。
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userId	String	ユーザーID。
available	boolean	true：ユーザーがサブチャンネル画面を起動。false：ユーザーがサブチャンネル画面を停止。

メッセージイベントのコールバック

onRecvRoomTextMsg

テキストメッセージ受信のコールバック。
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
message	String	テキストメッセージ。
sendId	String	送信者のユーザーID。

userAvatar	String	送信者のユーザープロフィール画像。
userName	String	送信者のユーザーニックネーム。

onRecvRoomCustomMsg

カスタムメッセージ受信のコールバック。
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
command	String	コマンドワードです。開発者がカスタマイズするものであり、主にさまざまなメッセージタイプを区別するために使用されます。
message	String	テキストメッセージ。
sendId	String	送信者のユーザーID。
userAvatar	String	送信者のユーザープロフィール画像。
userName	String	送信者のユーザーニックネーム。

スクリーンキャプチャのイベントコールバック

onScreenCaptureStarted

スクリーンキャプチャ開始のコールバック。

onScreenCapturePaused

スクリーンキャプチャ一時停止のコールバック。

onScreenCaptureResumed

スクリーンキャプチャ再生のコールバック。

onScreenCaptureStoped

スクリーンキャプチャ停止のコールバック。
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
reason	int	停止の理由。 0：ユーザーの自発的な停止。

		1：スクリーンウィンドウが閉じられたことによるによる停止。

TUIRoom(Windows&Mac)

最終更新日：：2024-07-19 15:35:35

TUIRoomは、Tencent CloudのTencent Real-Time Communication（TRTC）およびIMサービスを基に組み合わせたコンポーネントで、以下の機能をサポートしています。
キャスターがルームを作成し、参加者はルームナンバーを入力した後に入室できます。
参加者の間で画面共有を行います。
各種のテキストメッセージとカスタムメッセージの送信をサポートします。
説明：
TUIKitシリーズコンポーネントはTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期してアクティブ化することができます。 IMサービスの課金ルールの詳細については、Instant Messagingの料金説明をご参照ください。TRTCをアクティブ化すると、関連するIM SDKの体験版がデフォルトでアクティブ化されます。これは100 DAUのみをサポートします。
TUIRoomはオープンソースのClassであり、Tencent Cloudの2つのクローズドソースのSDKに依存しています。具体的な実装プロセスについては、 多人数オーディオビデオルーム(Windows&Mac)をご参照ください。
TRTC SDK： TRTC SDK を低遅延のビデオミーティングのコンポーネントとして使用します。
IM SDK：IM SDK を利用してチャットルームの機能（**IM SDKはC++バージョンを使用**）を実装します。

TUIRoom API概要

TUIRoomCore基本関数

API	説明
GetInstance	シングルトンオブジェクトを取得します。
DestroyInstance	シングルトンオブジェクトを破棄します。
SetCallback	イベントコールバックを設定します。

ルーム関連インターフェース関数

API	説明
login	ログイン。
logout	ログアウト。
CreateRoom	ルームの作成（キャスターが呼び出し）。

DestroyRoom	ルームの破棄（キャスターが呼び出し）。
EnterRoom	入室（参加者が呼び出し）。
LeaveRoom	退室（参加者が呼び出し）。
GetRoomInfo	ルーム情報の取得。
GetRoomUsers	ルーム内全メンバー情報の取得。
GetUserInfo	特定ユーザーの情報の取得。
TransferRoomMaster	キャスター権限の移転（キャスターが呼び出し）。

ローカルのオーディオビデオ操作インターフェース

API	説明
StartCameraPreview	ローカルビデオのプレビュー画面を立ち上げます。
StopCameraPreview	ローカルビデオキャプチャおよびプレビューを停止します。
UpdateCameraPreview	ローカルビデオレンダリングウィンドウを変更します。
StartLocalAudio	マイクキャプチャを起動します。
StopLocalAudio	マイクキャプチャを停止します。
StartSystemAudioLoopback	システム音声のキャプチャを起動/停止します。
StopSystemAudioLoopback	システム音声のキャプチャを起動/停止します。
SetVideoMirror	ローカル画面のイメージプレビューモードを設定します。

リモートユーザーに関するインターフェース

API	説明
StartRemoteView	指定メンバーのリモートビデオ画面をサブスクリプションし再生します。
StopRemoteView	リモートビデオ画面のサブスクリプションをキャンセルし再生を停止します。
UpdateRemoteView	リモートユーザーのビデオレンダリングウィンドウを変更します。

チャットメッセージ送信インターフェース

API	説明

SendChatMessage	チャットメッセージを送信します。
SendCustomMessage	カスタムメッセージを送信します。

フィールドコントロール関連インターフェース

API	説明
MuteUserMicrophone	特定ユーザーのマイクを無効化/再有効化します。
MuteAllUsersMicrophone	全ユーザーのマイクを無効化/再有効化し、ステータスをルーム情報に同期させます。
MuteUserCamera	特定ユーザーのカメラを無効化/再有効化します。
MuteAllUsersCamera	全ユーザーのカメラを無効化/再有効化し、ステータスをルーム情報に同期させます。
MuteChatRoom	チャットルームのミュートを開始/停止します（キャスターが呼び出し）。
KickOffUser	ルーム内の特定ユーザーをリムーブします（キャスターが呼び出し）。
StartCallingRoll	キャスターが点呼を開始します。
StopCallingRoll	キャスターが点呼を終了します。
ReplyCallingRoll	メンバーがキャスターの点呼に応答します。
SendSpeechInvitation	キャスターが参加者の発言を要請します。
CancelSpeechInvitation	キャスターが参加者の発言要請をキャンセルします。
ReplySpeechInvitation	参加者がキャスターの発言申請に同意/拒否します。
SendSpeechApplication	参加者が発言を申請します。
CancelSpeechApplication	参加者が発言申請をキャンセルします。
ReplySpeechApplication	キャスターが参加者の発言申請に同意/拒否します。
ForbidSpeechApplication	キャスターが発言申請を禁止します。
SendOffSpeaker	キャスターが参加者に発言を停止するよう命令します。
SendOffAllSpeakers	キャスターが全員に発言を停止するよう命令します。
ExitSpeechState	参加者は発言を停止し、視聴者になります。

基本コンポーネントインターフェース関数

API	説明
GetDeviceManager	ローカル設定管理オブジェクトITXDeviceManagerを取得します。
GetScreenShareManager	画面共有管理オブジェクトIScreenShareManagerを取得します。

クラウドレコーディングインターフェース関数

API	説明
StartCloudRecord	クラウドレコーディングを開始します 。
StopCloudRecord	クラウドレコーディングを停止します 。

美顔関連インターフェース関数

API	説明
SetBeautyStyle	美顔を設定します。

関連設定インターフェース

API	説明
SetVideoQosPreference	ネットワークトラフィックコントロール関連パラメータを設定します。

SDKバージョンインターフェース関数の取得

API	説明
GetSDKVersion	SDKバージョンを取得します。

TUIRoomCoreCallback API概要

エラーイベントコールバック

API	説明
OnError	エラーのコールバック。

基本イベントコールバック

API	説明
OnLogin	ログインコールバック。
OnLogout	ログアウトコールバック。
OnCreateRoom	ルーム作成のコールバック。
OnDestroyRoom	ルーム解散のコールバック。
OnEnterRoom	入室のコールバック。
OnExitRoom	退室のコールバック。
OnFirstVideoFrame	最初のフレーム画面のコールバック。
OnUserVoiceVolume	音量の大きさのコールバック。
OnRoomMasterChanged	キャスター変更のコールバック。

リモートユーザーイベントコールバック

API	説明
OnRemoteUserEnter	リモートユーザー入室コールバック。
OnRemoteUserLeave	リモートユーザー退室コールバック。
OnRemoteUserCameraAvailable	リモートユーザーがカメラビデオを起動するかどうかのコールバック。
OnRemoteUserScreenAvailable	リモートユーザーが画面共有を開始するかどうかのコールバック。
OnRemoteUserAudioAvailable	リモートユーザーがマイクをオンにしているかどうかのコールバック。
OnRemoteUserEnterSpeechState	リモートユーザーの発言開始のコールバック。
OnRemoteUserExitSpeechState	リモートユーザーの発言終了のコールバック。

メッセージイベントのコールバック

API	説明
OnReceiveChatMessage	テキストメッセージ受信のコールバック。
OnReceiveCustomMessage	テキストメッセージ受信のコールバック。

フィールドコントロールイベントコールバック

API	説明
OnReceiveSpeechInvitation	ユーザーがキャスターの発言要請を受信した場合のコールバック。
OnReceiveInvitationCancelled	ユーザーがキャスターの発言要請キャンセルを受信する場合のコールバック。
OnReceiveReplyToSpeechInvitation	キャスターがユーザーの発言要請への同意を受信する場合のコールバック。
OnReceiveSpeechApplication	キャスターがユーザーの発言申請を受信する場合のコールバック。
OnSpeechApplicationCancelled	ユーザーが発言申請をキャンセルする場合のコールバック。
OnReceiveReplyToSpeechApplication	キャスターが発言申請に同意する場合のコールバック。
OnSpeechApplicationForbidden	キャスターが発言申請を禁止する場合のコールバック。
OnOrderedToExitSpeechState	参加者が発言の停止をリクエストされる場合のコールバック。
OnCallingRollStarted	キャスターが点呼を開始し、参加者が受信する場合のコールバック。
OnCallingRollStopped	キャスターが点呼を終了し、参加者が受信する場合のコールバック。
OnMemberReplyCallingRoll	参加者が点呼に応答し、キャスターが受信する場合のコールバック。
OnChatRoomMuted	キャスターがチャットルームのミュートを変更する場合のコールバック。
OnMicrophoneMuted	キャスターがマイクの無効化を設定する場合のコールバック。
OnCameraMuted	キャスターがカメラの無効化を設定する場合のコールバック。

統計および品質コールバック

API	説明
OnStatistics	技術指標統計のコールバック。
OnNetworkQuality	ネットワーク品質のコールバック。

画面共有関連コールバック

API	説明
OnScreenCaptureStarted	画面共有開始のコールバック。
OnScreenCaptureStopped	画面共有停止のコールバック。

ビデオレコーディングコールバック

API	説明
OnRecordError	レコーディングエラーのコールバック。
OnRecordComplete	レコーディング完了のコールバック。
OnRecordProgress	レコーディング進捗のコールバック。

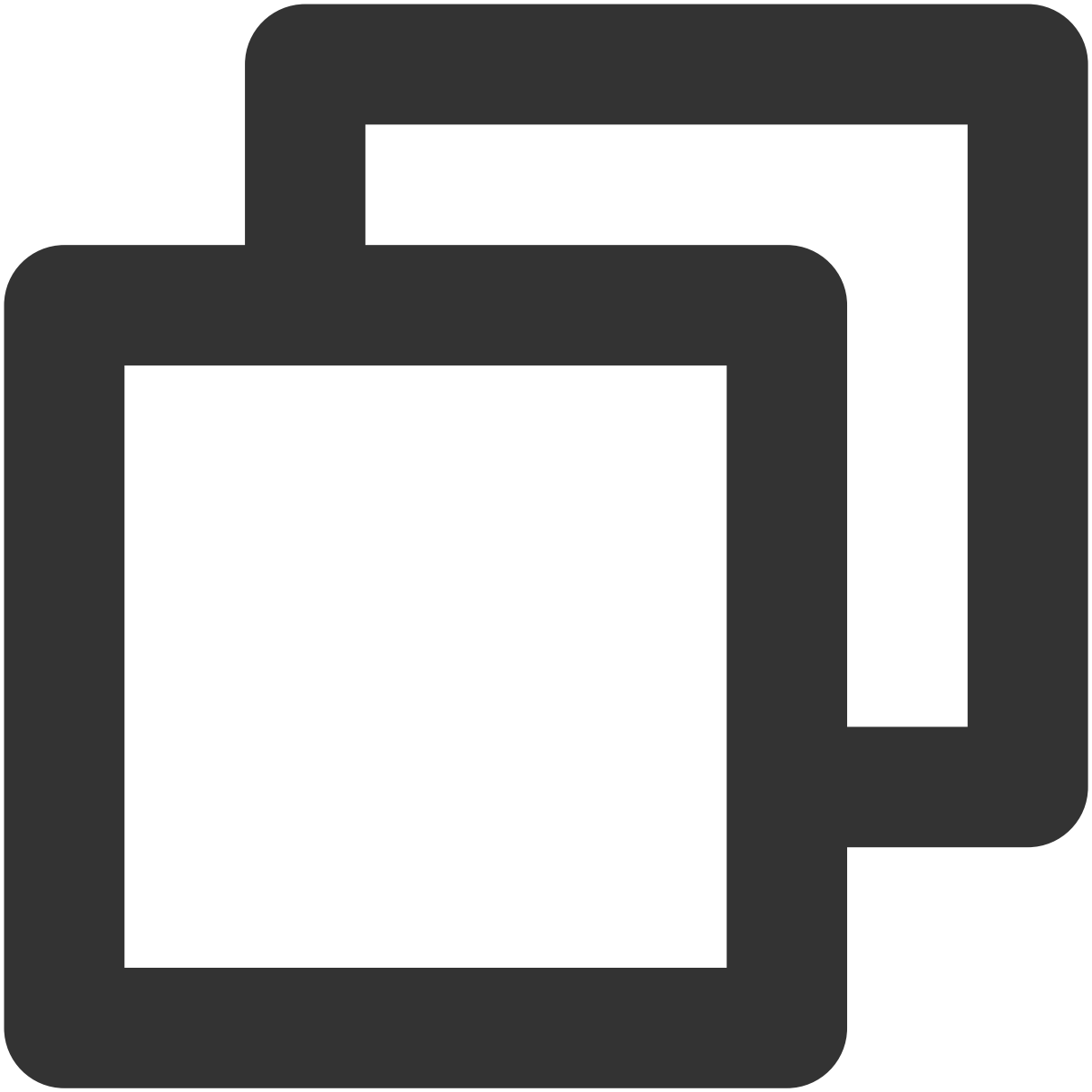
ローカルデバイステストコールバック

API	説明
OnTestSpeakerVolume	スピーカー音量のコールバック。
OnTestMicrophoneVolume	マイク音量のコールバック。
OnAudioDeviceCaptureVolumeChanged	システムキャプチャ音量調節のコールバック。
OnAudioDevicePlayoutVolumeChanged	システム再生音量調節のコールバック。

TUIRoomCore基本関数

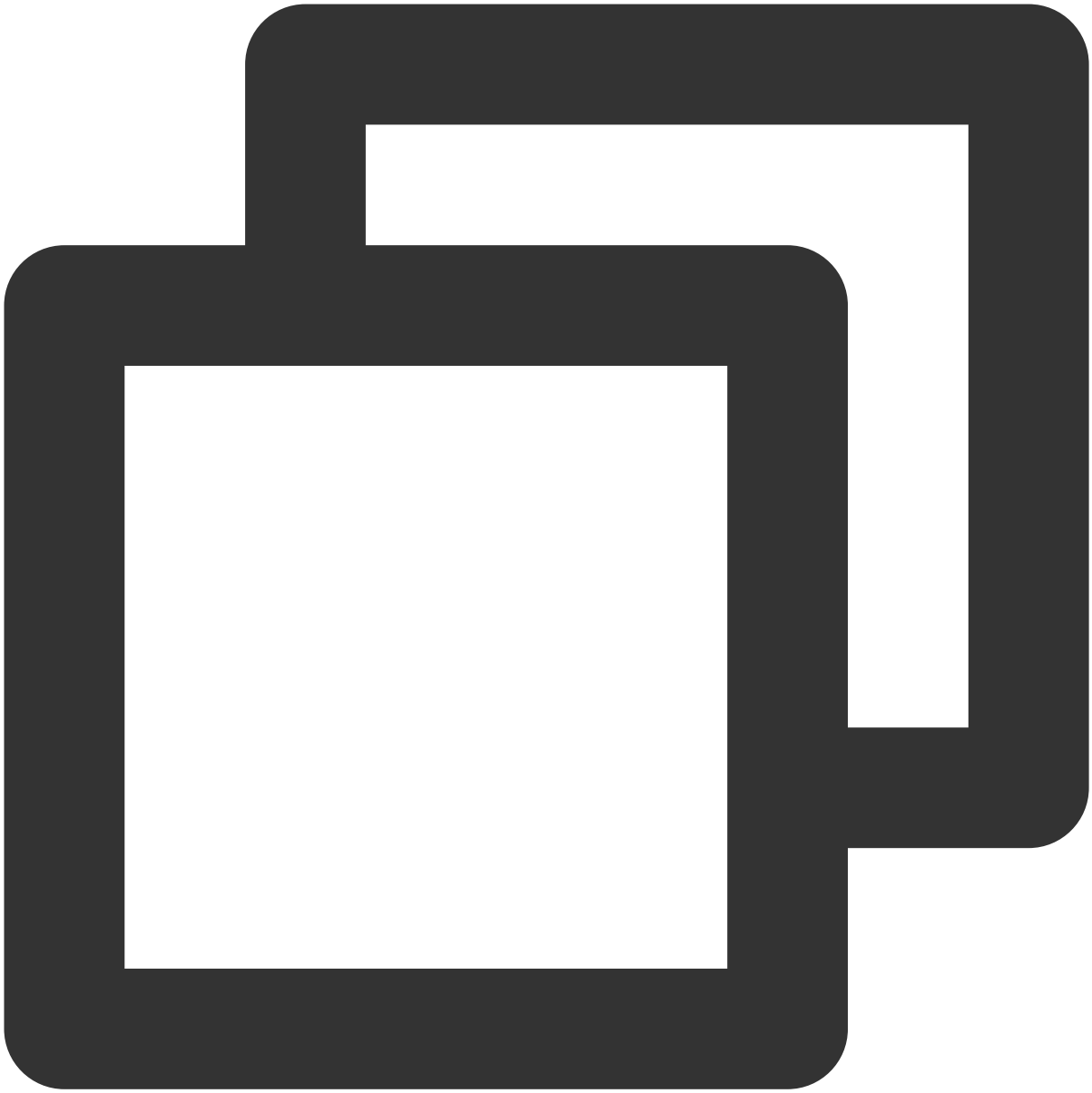
GetInstance

TUIRoomCore シングルトンオブジェクトを取得します。



```
static TUIRoomCore* GetInstance();
```

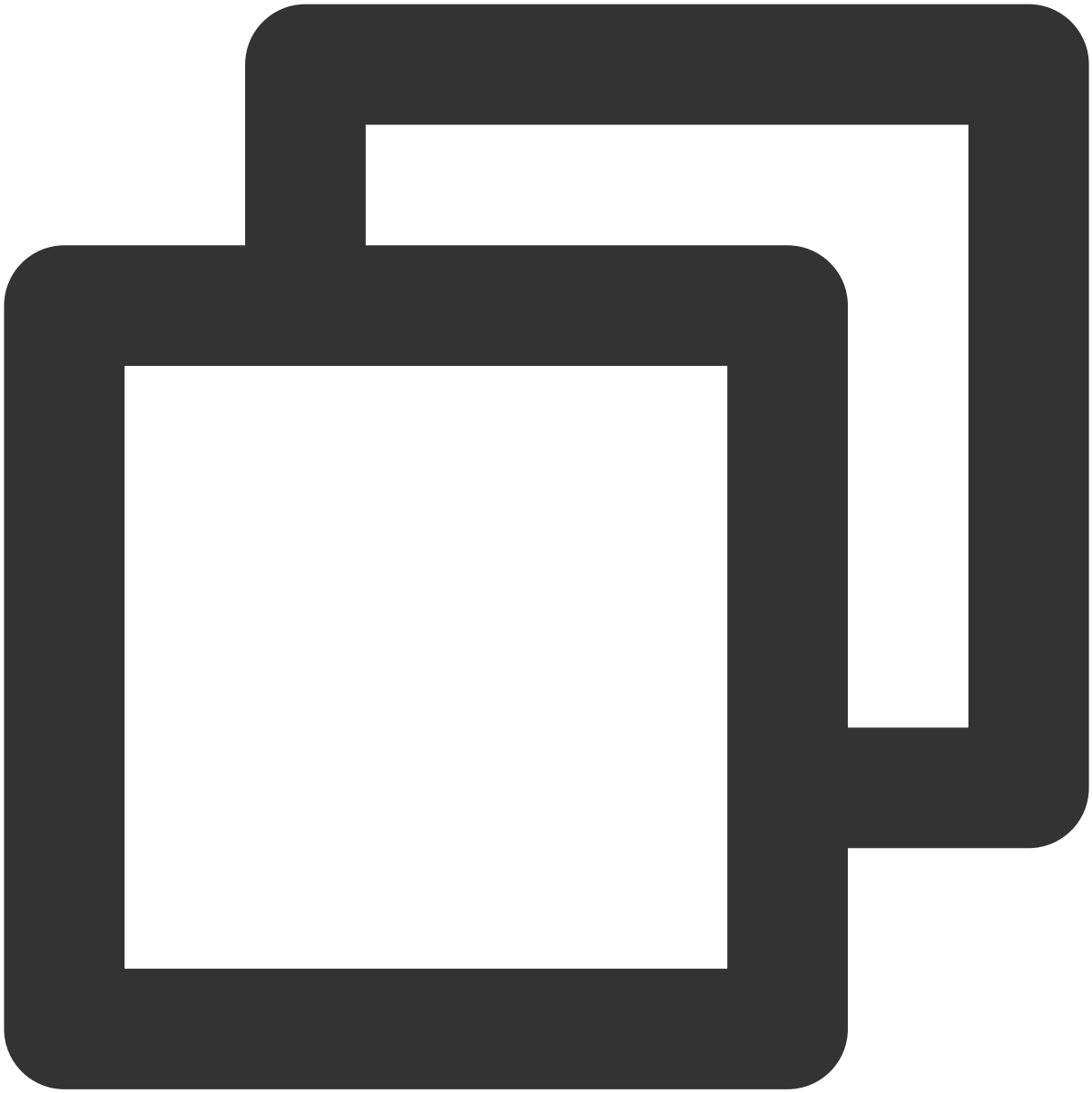
DestroyInstance



```
static void DestroyInstance();
```

SetCallback

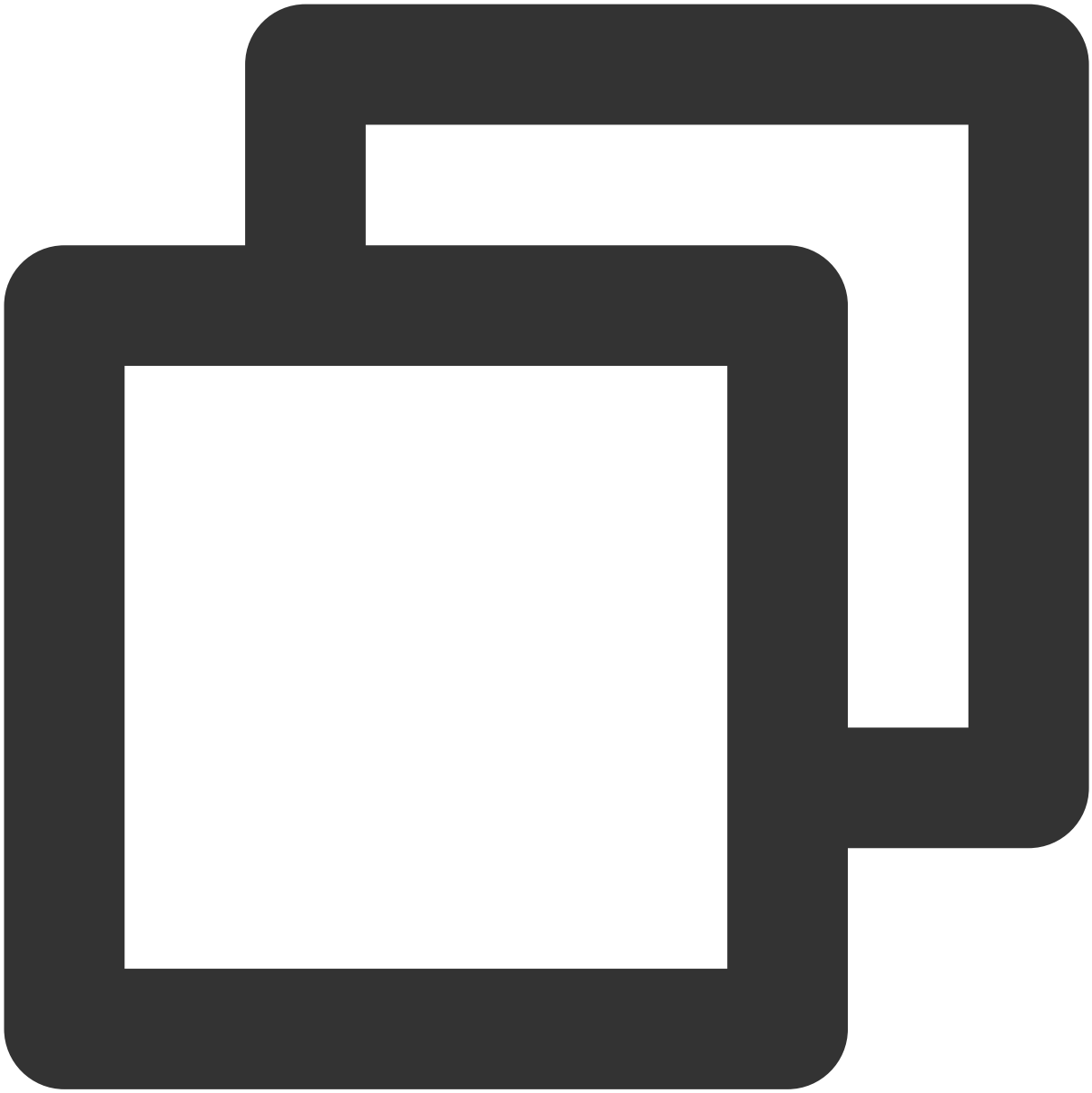
TUIRoomCoreイベントコールバック。TUIRoomCoreCallbackを介してTUIRoomCoreの各種ステータス通知を取得できます。



```
virtual void SetCallback(const TUIRoomCoreCallback* callback) = 0;
```

Login

ログイン。



```
virtual int Login(int sdk_appid, const std::string& user_id, const std::string& use
```

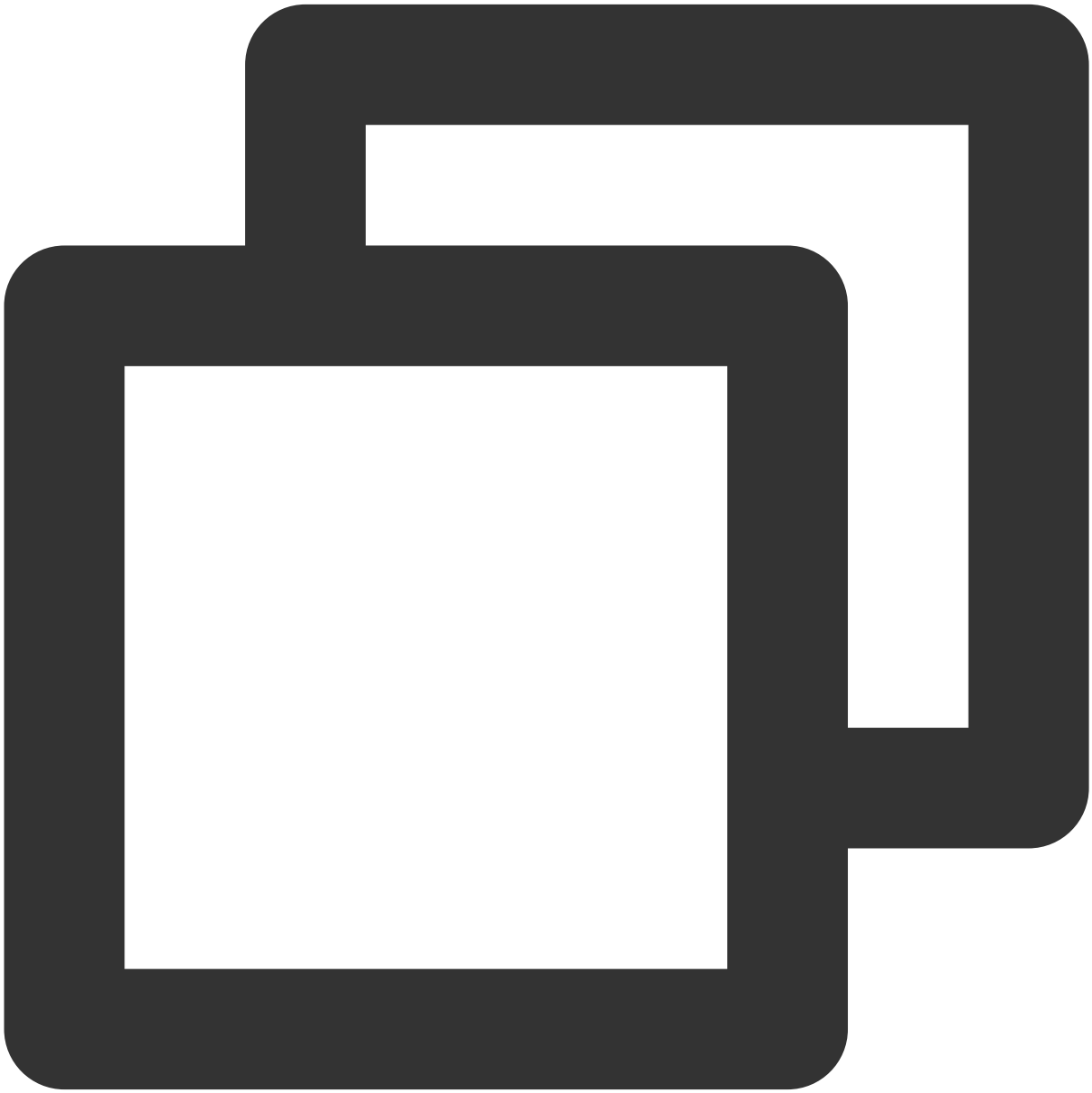
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
sdk_appid	int	TRTCコンソール > アプリケーション管理> アプリケーション情報の中でSDKAppIDを確認できます。
user_id	string	現在のユーザーID。文字列タイプでは、英語のアルファベット（a-z、A-Z）、数字

		（0-9）、ハイフン（-）とアンダーライン（_）のみ使用できます。業務の実際のアカウントシステムと組み合わせてご自身で設定することをお勧めします。
user_sig	string	Tencent Cloudによって設計されたセキュリティ保護署名。取得方法については、UserSigの計算、使用方法をご参照ください。

Logout

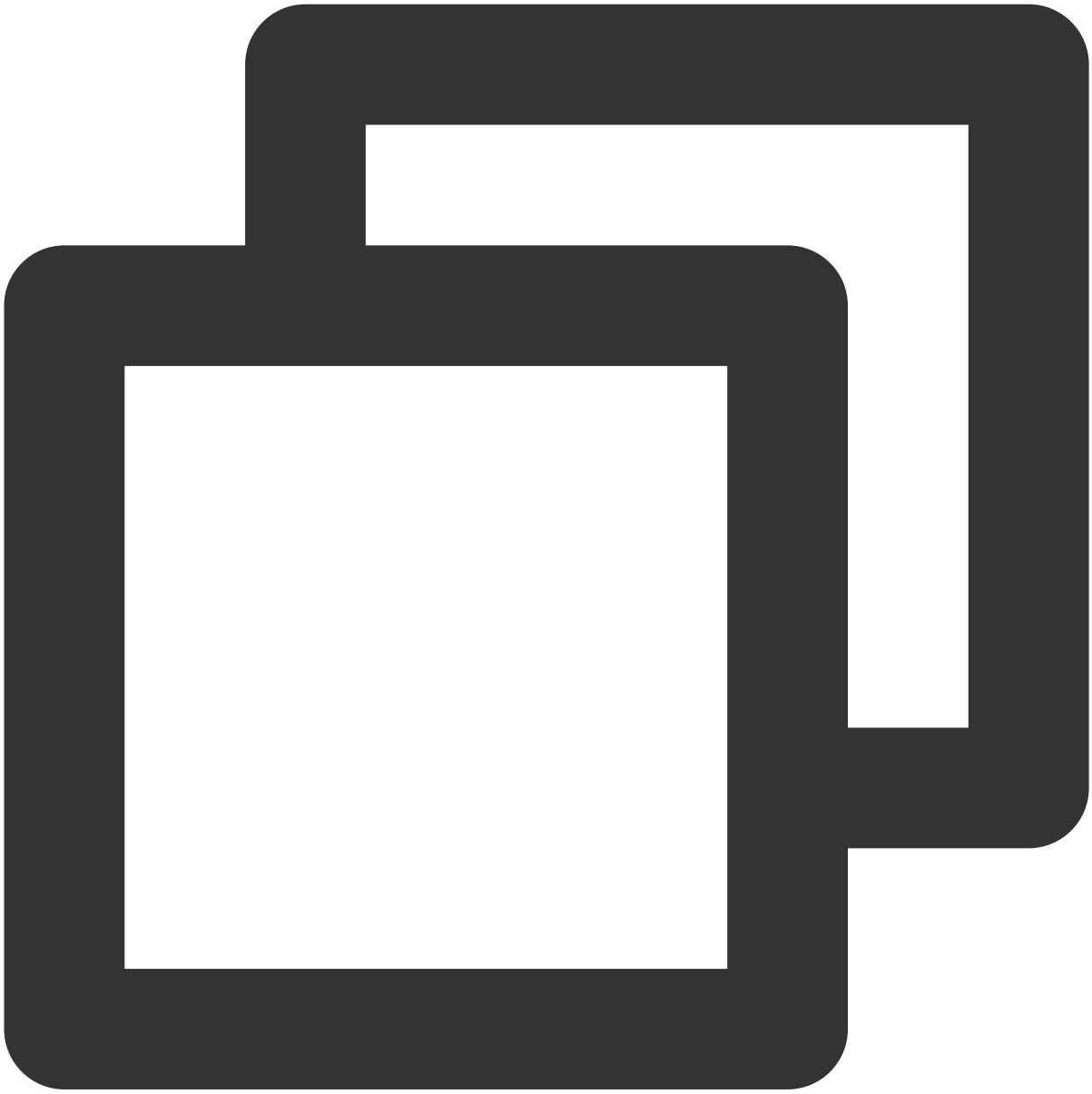
ログアウト。



```
virtual int Logout() = 0;
```

CreateRoom

ルームの作成（キャスターが呼び出し）。



```
virtual int CreateRoom(const std::string& room_id, TUISpeechMode speech_mode) = 0;
```

パラメータは下表に示すとおりです。

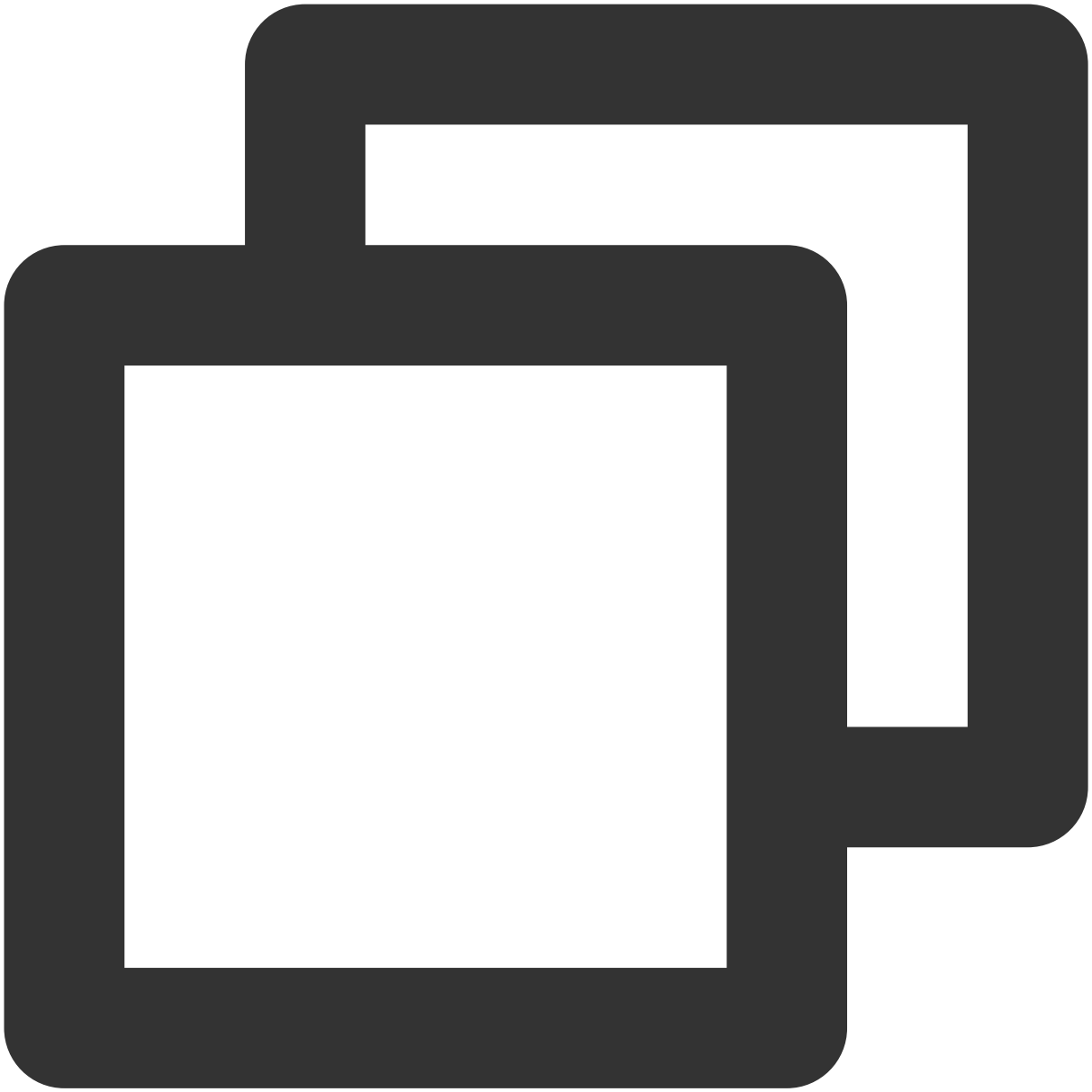
パラメータ	タイプ	意味
room_id	string	ルームIDは、ご自身でアサインし、一元管理する必要があります。

speech_mode	TUISpeechMode	発言モード。

キャスターの通常の呼び出しフローは以下のとおりです。
1. **キャスター**が `CreateRoom()` を呼び出し、ルームを作成します。ルームの作成の成否はOnCreateRoomを介してキャスターに通知されます。
2. **キャスター**が `EnterRoom()` を呼び出し、入室します。
3. **キャスター**が `StartCameraPreview()` を呼び出し 、カメラキャプチャとプレビューを起動します。
4. **キャスター**が `StartLocalAudio()` を呼び出し、ローカルマイクを起動します。

DestroyRoom

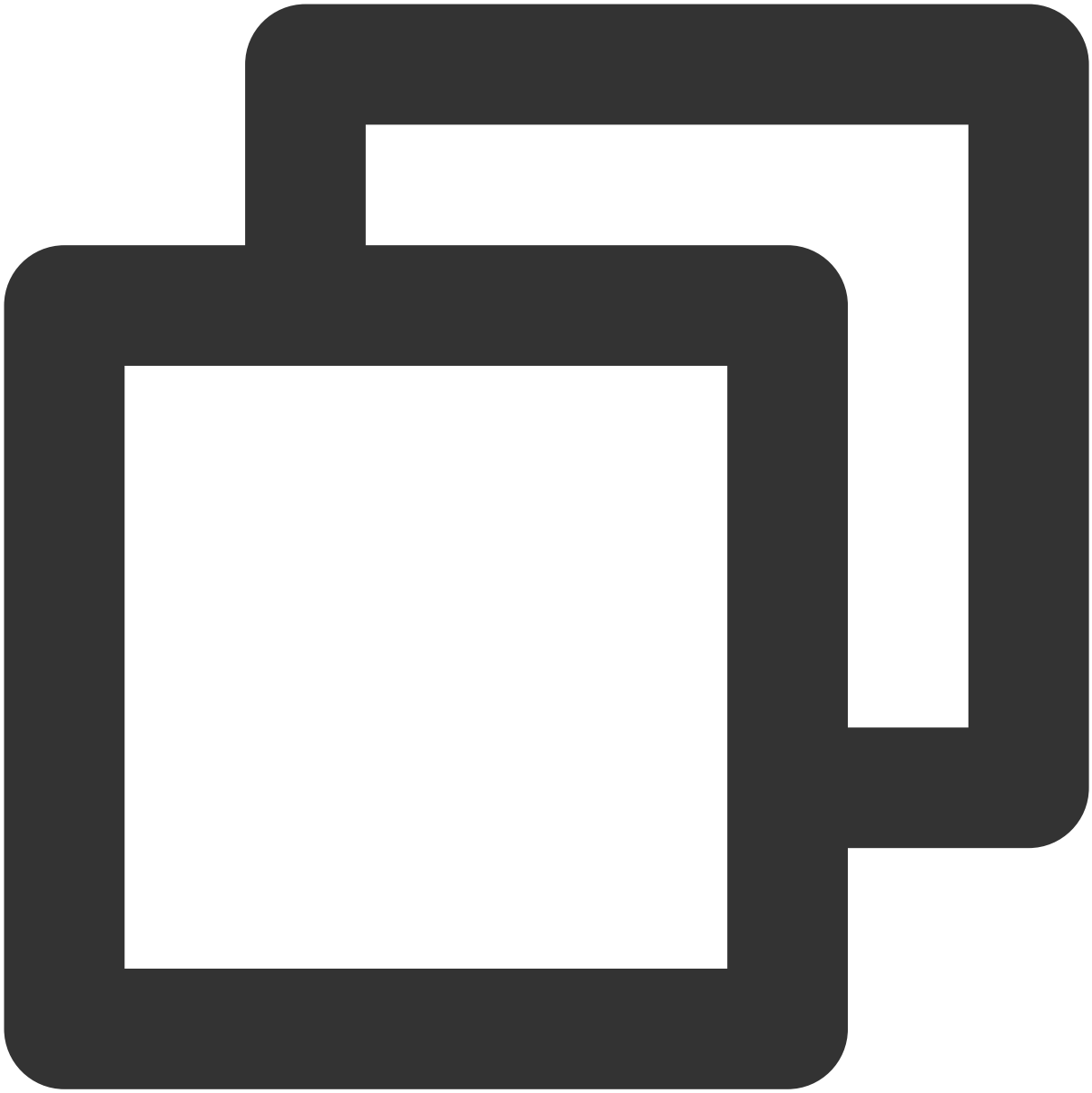
ルームの破棄（キャスターが呼び出し）。キャスターは、ルームの作成後、この関数を呼び出して、ルームを破棄できます。



```
virtual int DestroyRoom() = 0;
```

EnterRoom

入室（参加者が呼び出し）。



```
virtual int EnterRoom(const std::string& room_id) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
room_id	string	ルームID。

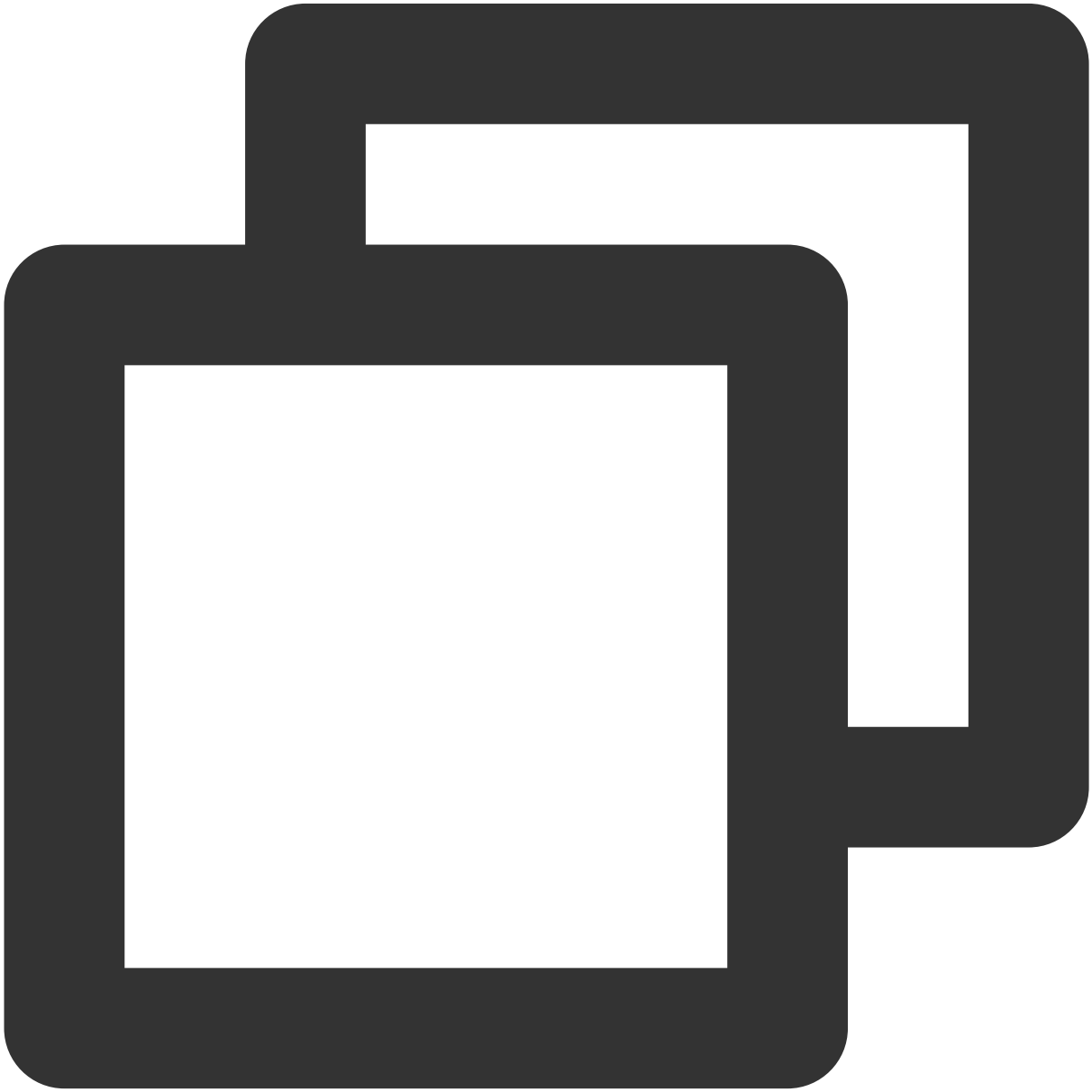
参加者が入室する場合の通常の呼び出し手順は次のとおりです。

1. **参加者**が `EnterRoom` を呼び出し、room_idを渡せば、入室できます。

2. **参加者**が `startCameraPreview()` を呼び出して、カメラプレビューを起動し、 `StartLocalAudio()` を呼び出して、マイクキャプチャを起動します。
3. **参加者**が `OnRemoteUserCameraAvailable` のイベントを受信し、 `StartRemoteView()` を呼び出して、ビデオ再生を開始します。

LeaveRoom

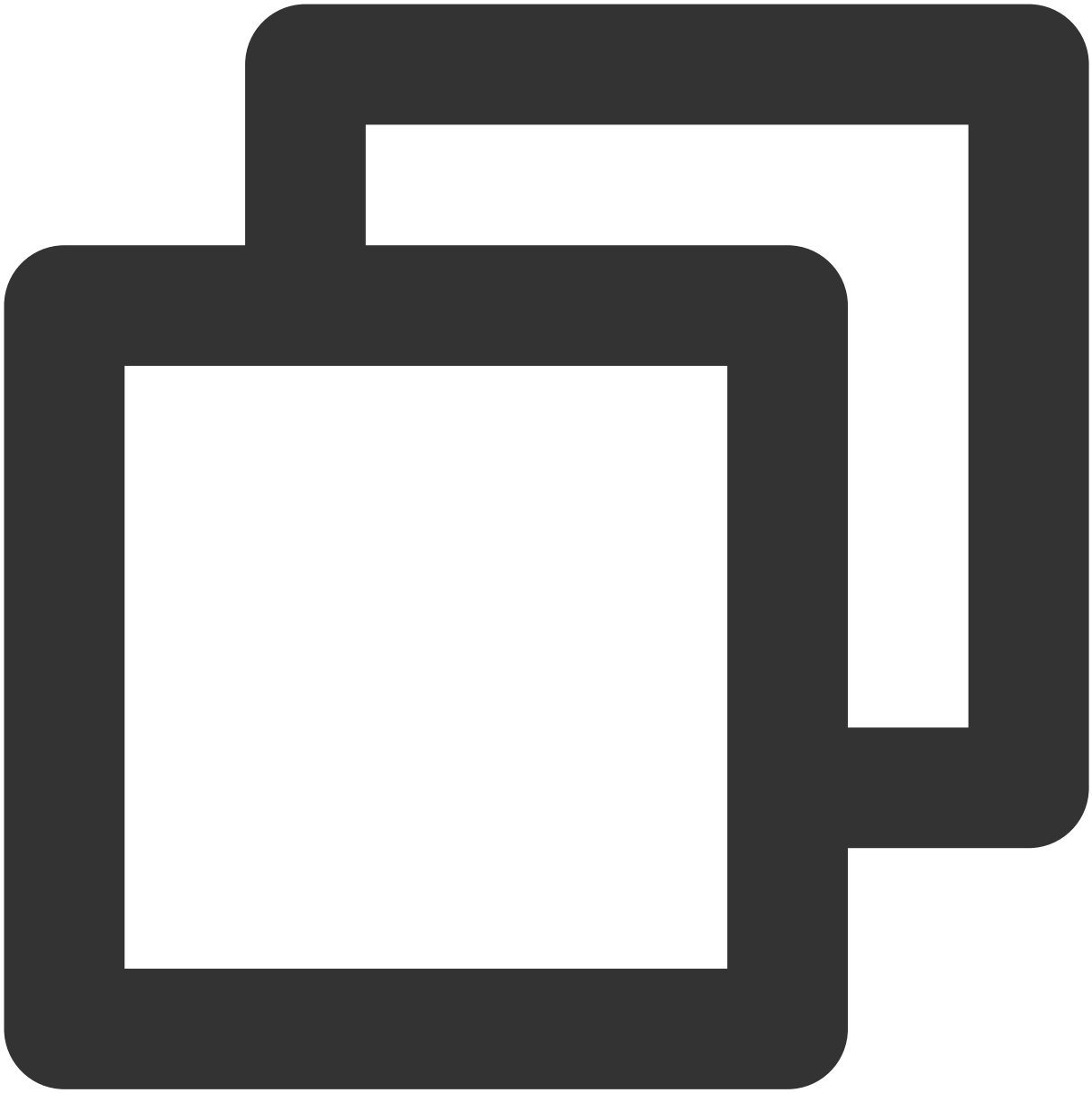
退室（参加者が呼び出し）。



```
virtual int LeaveRoom() = 0;
```

GetRoomInfo

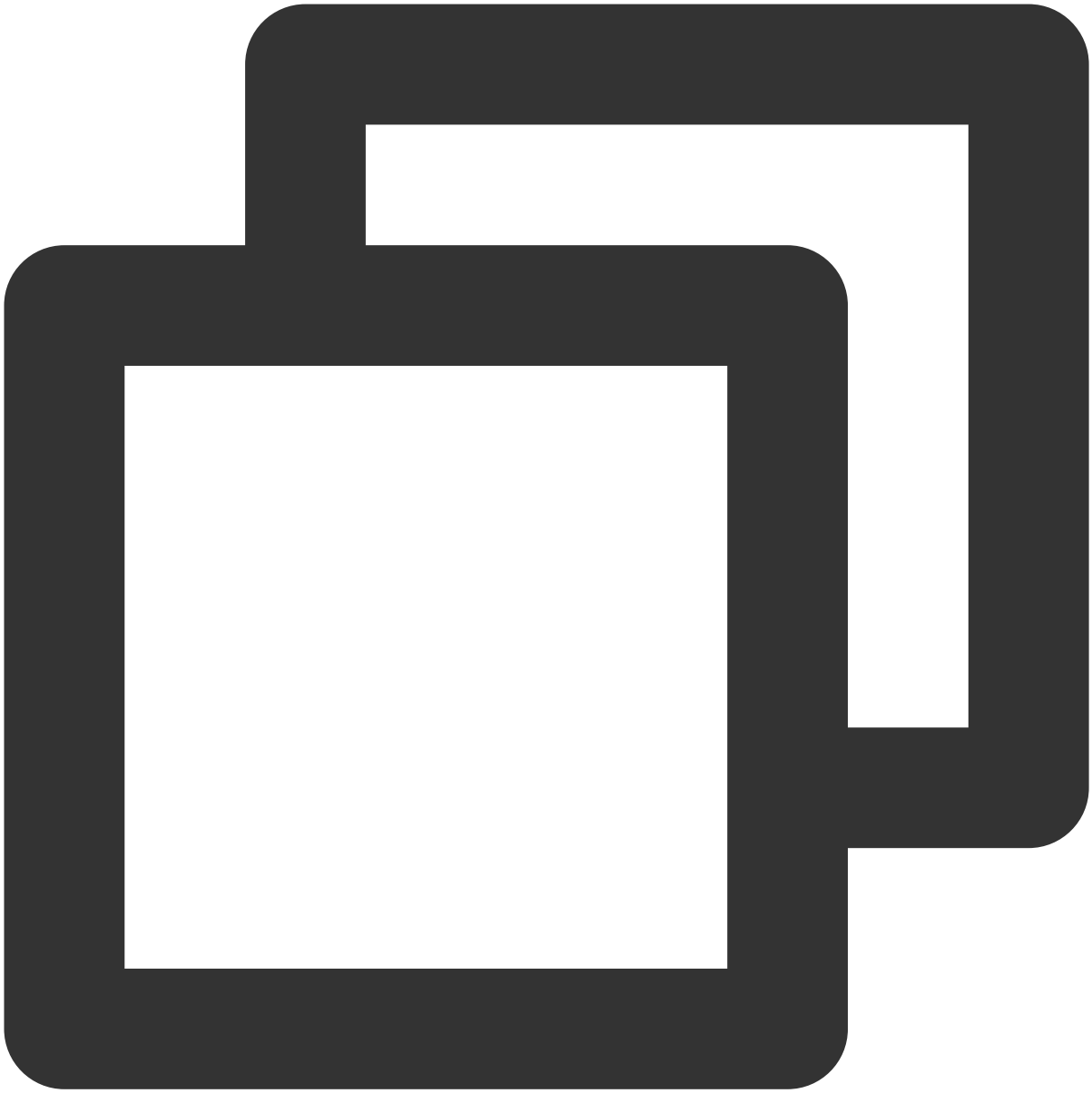
ルーム情報を取得します。



```
virtual TUIRoomInfo GetRoomInfo() = 0;
```

GetRoomUsers

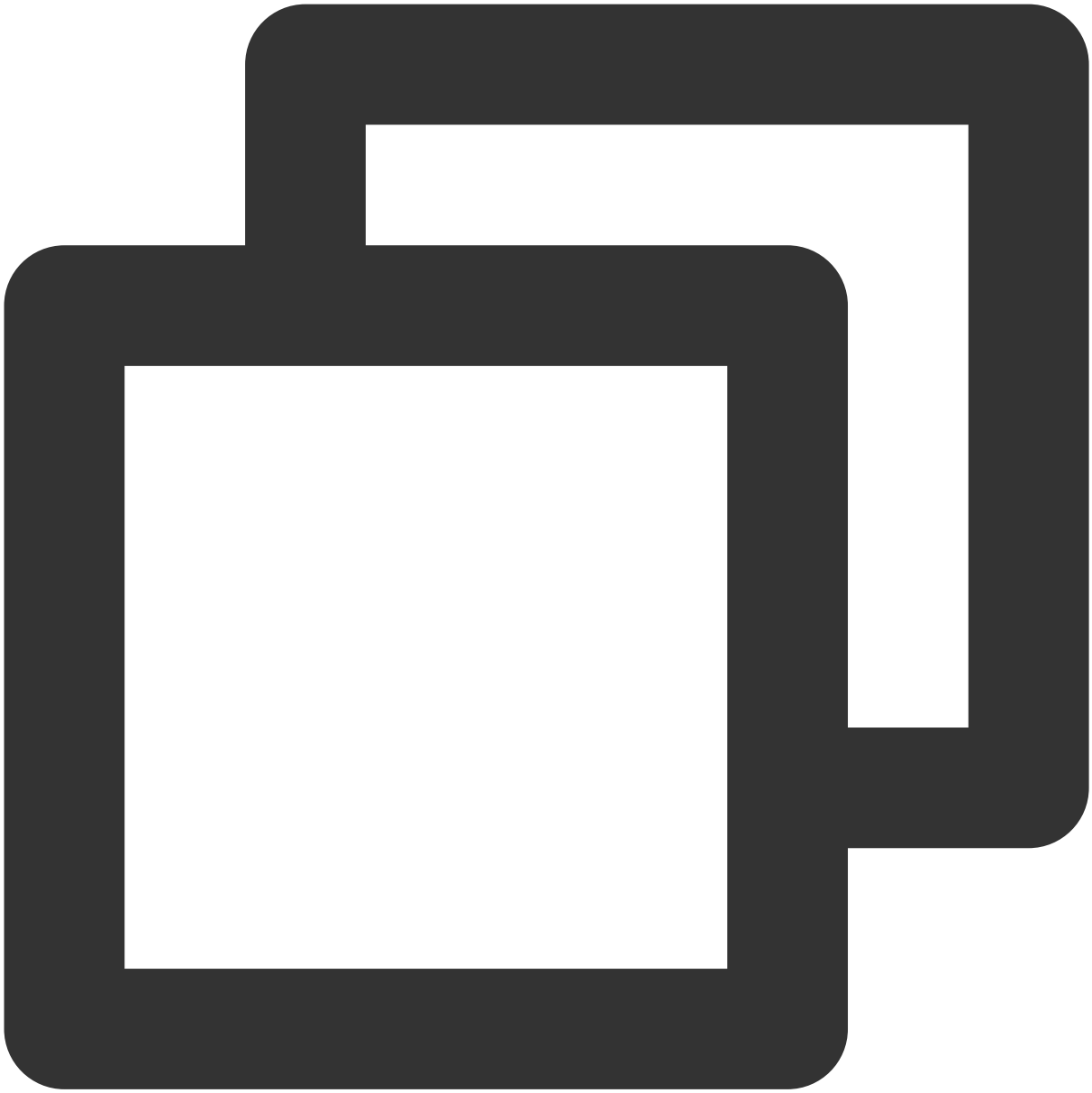
ルームの全メンバー情報を取得します。



```
virtual std::vector<TUIUserInfo> GetRoomUsers() = 0;
```

GetUserInfo

メンバー情報を取得します。



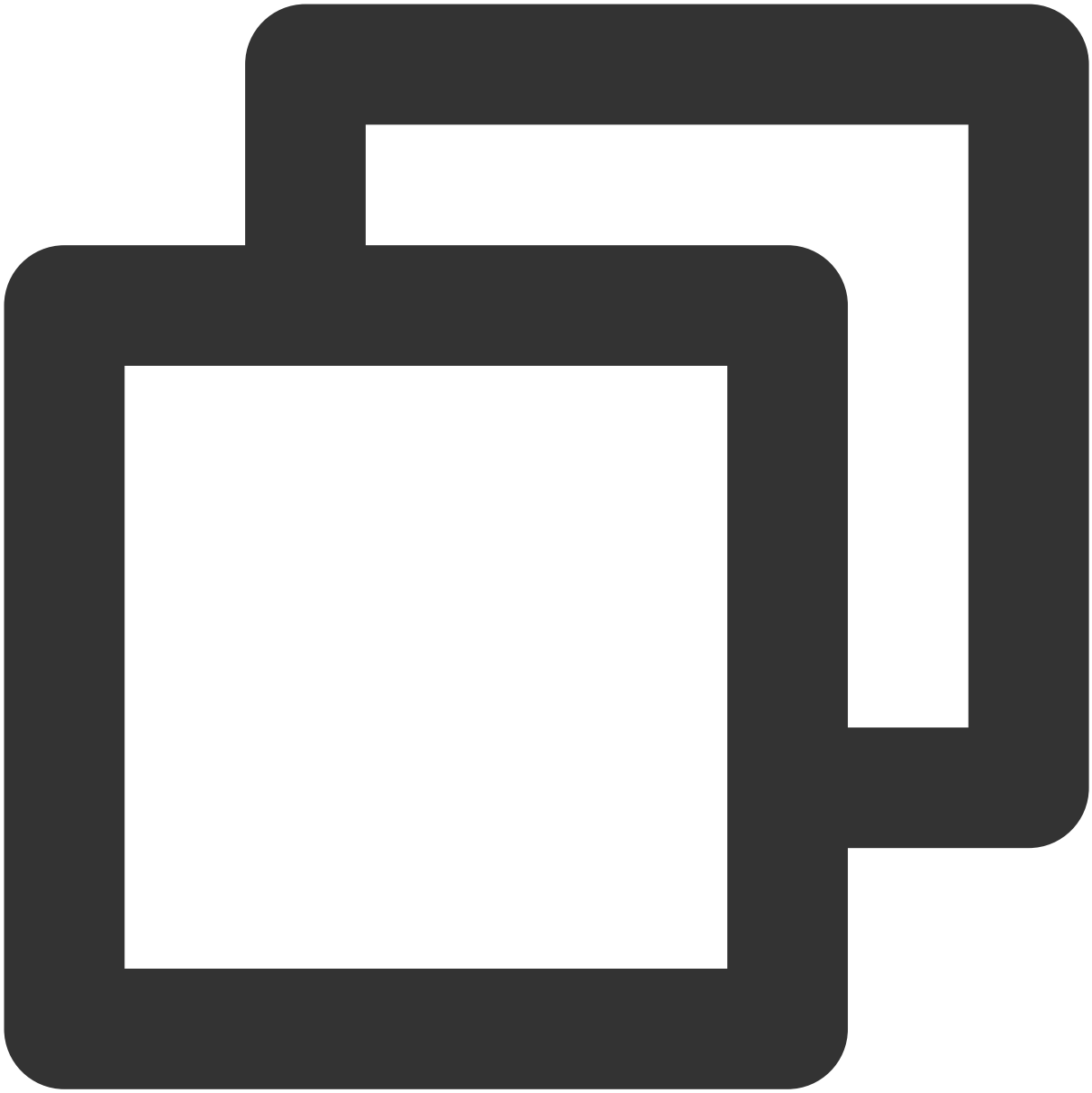
```
virtual const TUIUserInfo* GetUserInfo(const std::string& user_id) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザーID。

SetSelfProfile

ユーザーの属性を設定します。



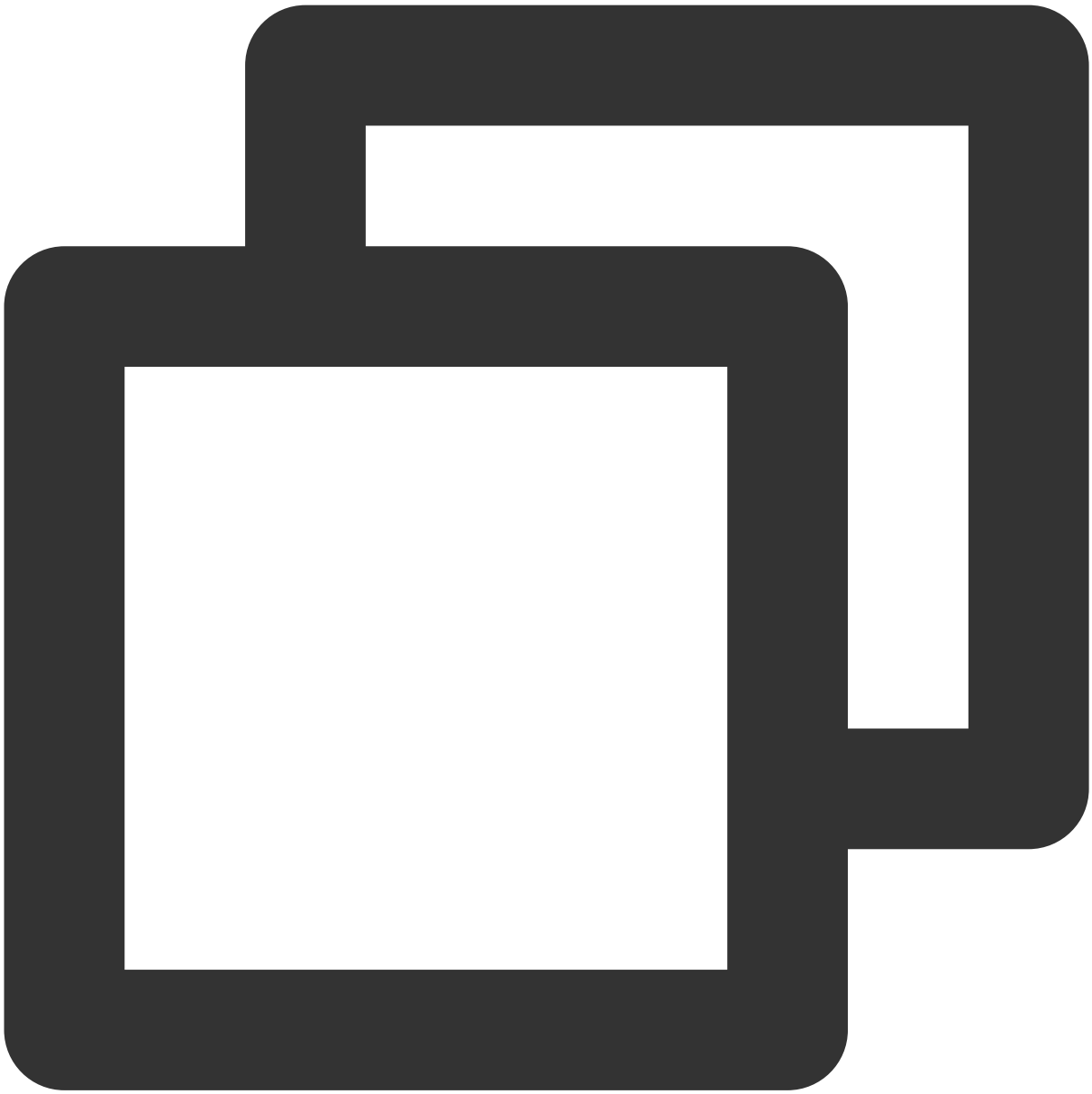
```
virtual int SetSelfProfile(const std::string& user_name, const std::string& avatar_
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_name	string	ユーザー氏名。
avatar_url	string	ユーザーのプロフィール画像URL。

TransferRoomMaster

グループを他のユーザーに引き渡します。



```
virtual int TransferRoomMaster(const std::string& user_id) = 0;
```

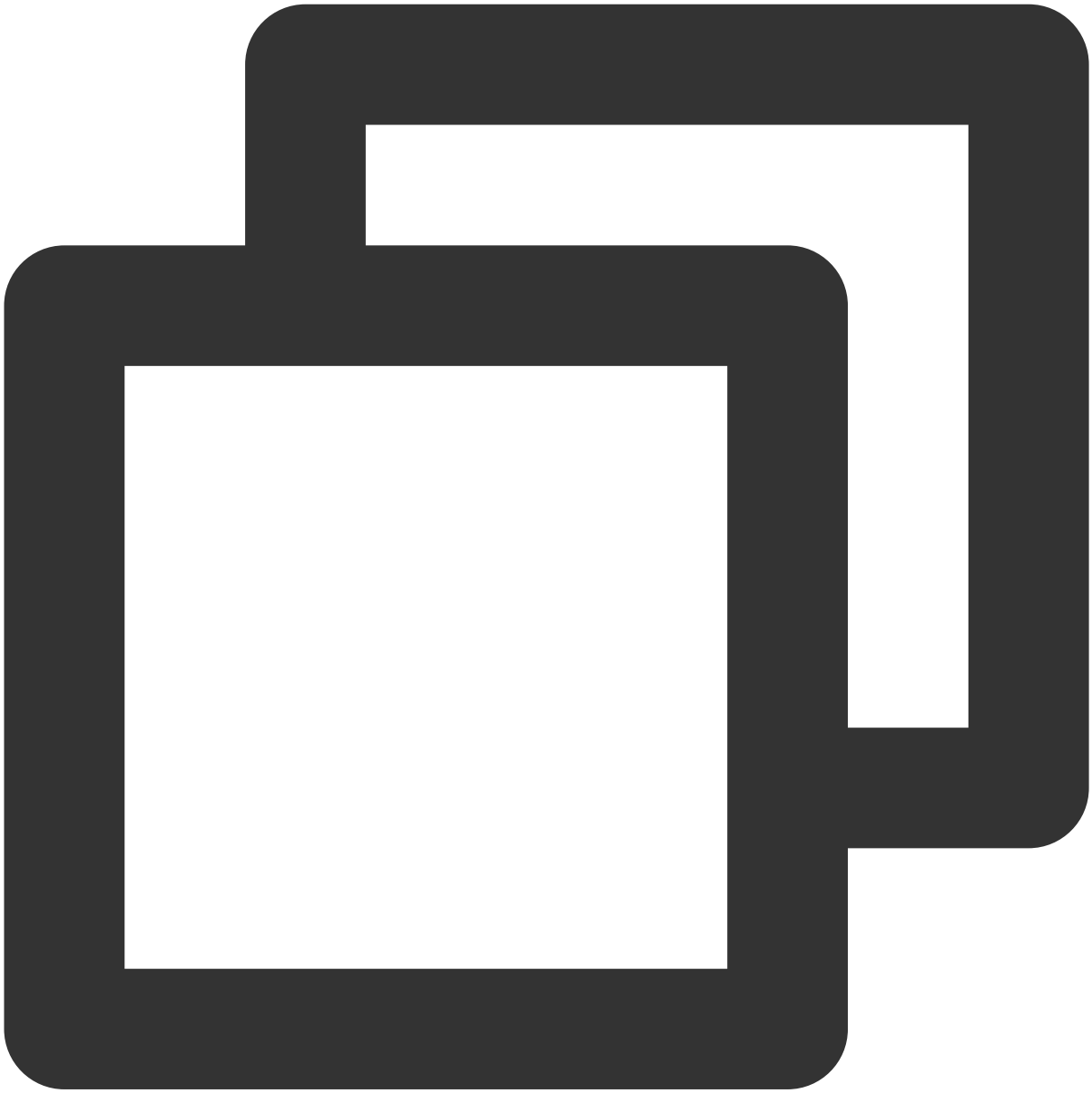
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザーID。

ローカルプッシュインターフェース

StartCameraPreview

ローカルカメラプレビューを起動します。



```
virtual int StartCameraPreview(const liteav::TXView& view) = 0;
```

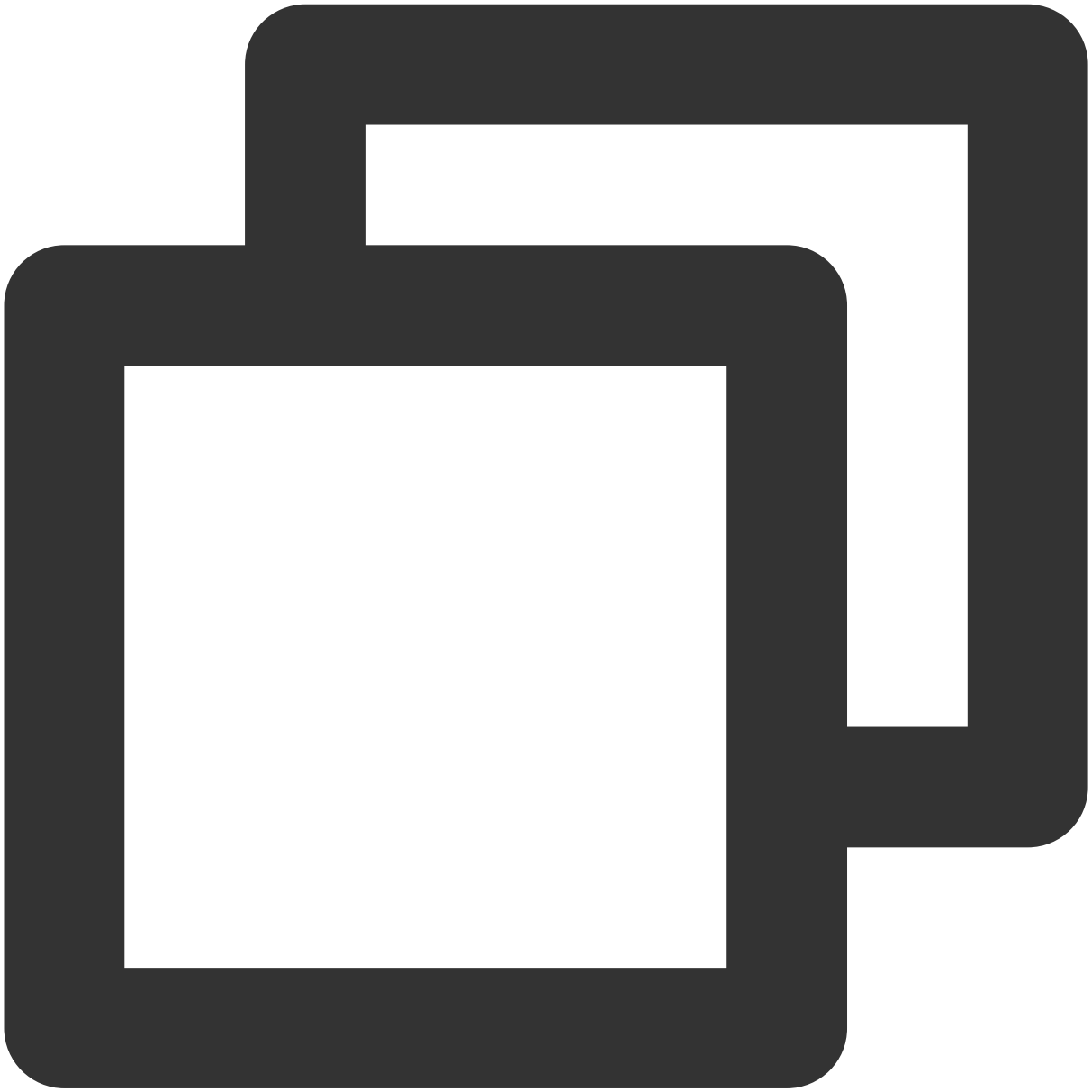
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

view	liteav::TXView	ウィンドウハンドル。

StopCameraPreview

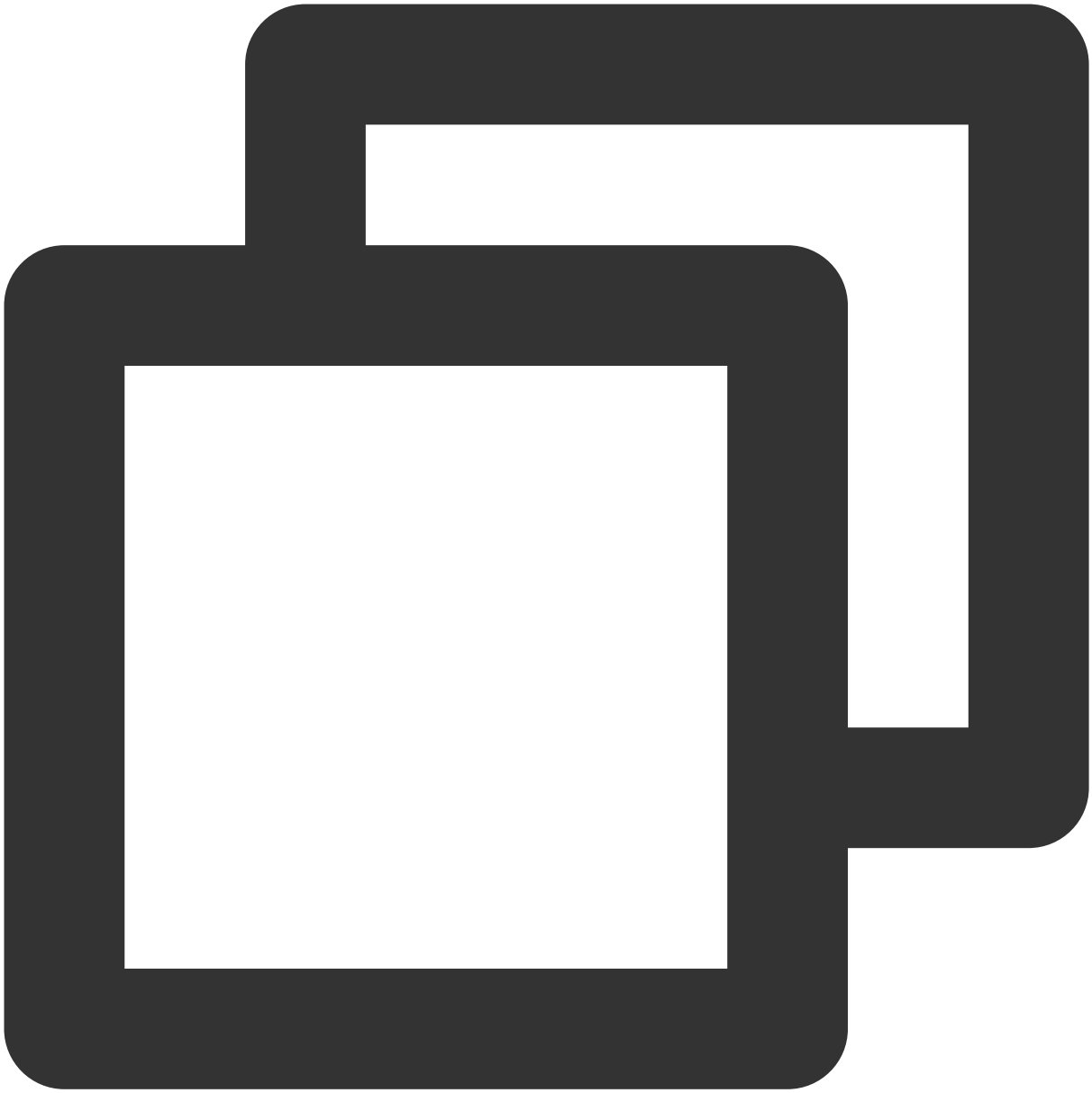
ローカルカメラプレビューを停止します。



```
virtual int StopCameraPreview() = 0;
```

UpdateCameraPreview

ローカルビデオプレビュー画面のウィンドウを更新します。



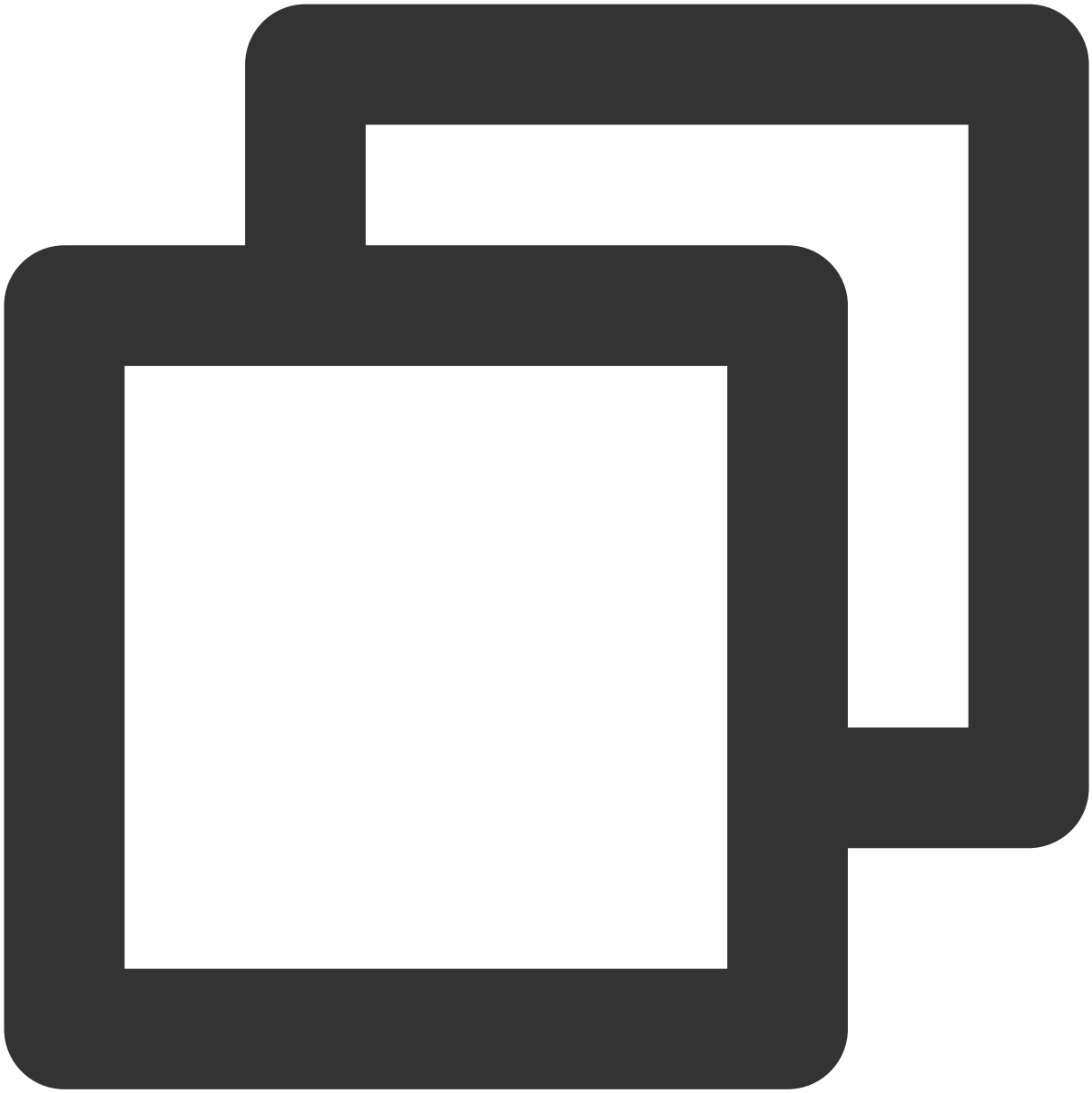
```
virtual int UpdateCameraPreview(const liteav::TXView& view) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
view	liteav::TXView	ウィンドウハンドル。

StartLocalAudio

ローカルオーディオデバイスを起動します。



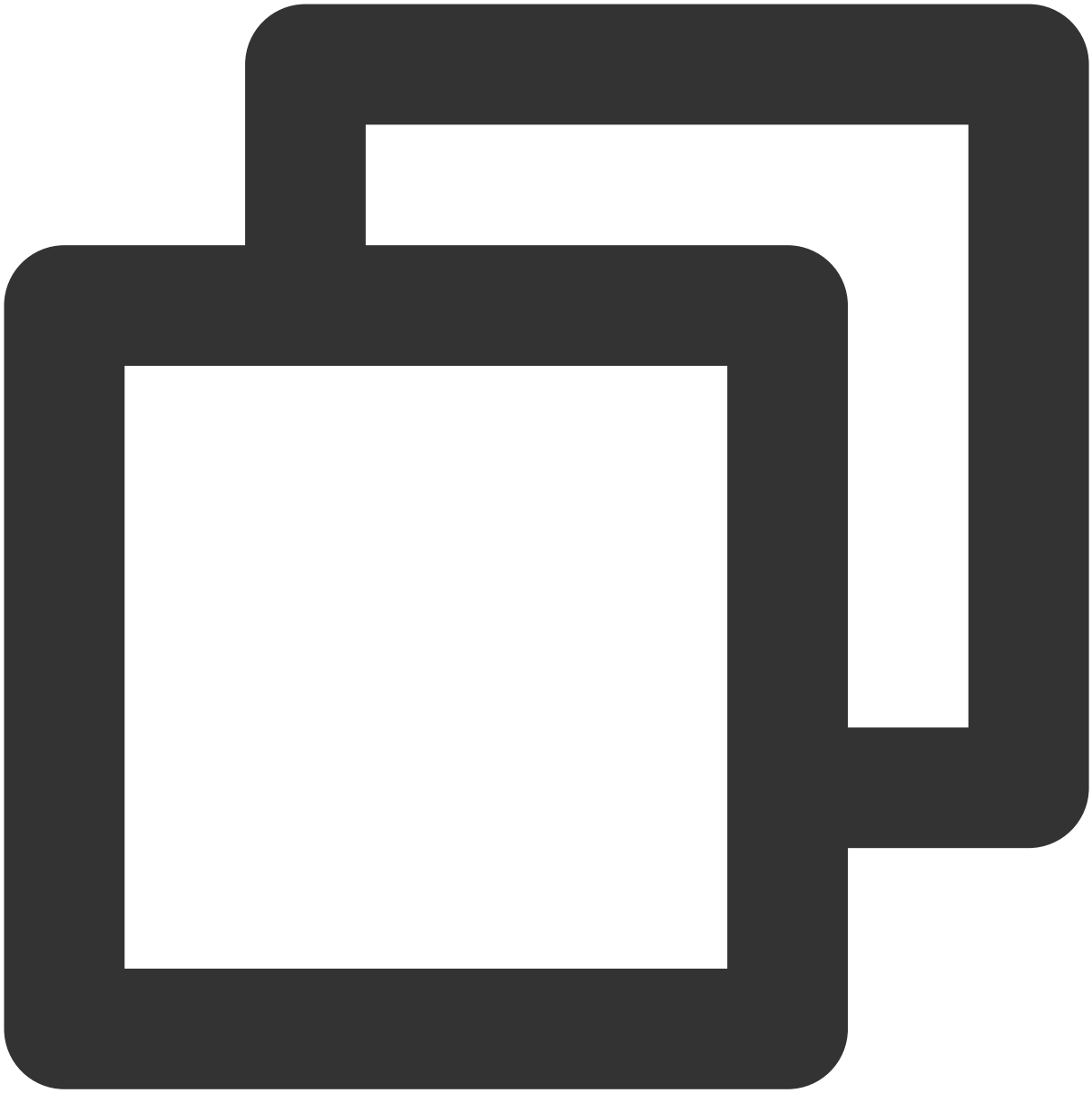
```
virtual int StartLocalAudio(const liteav::TRTCAudioQuality& quality) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
view	liteav::TXView	ウィンドウハンドル。

StopLocalAudio

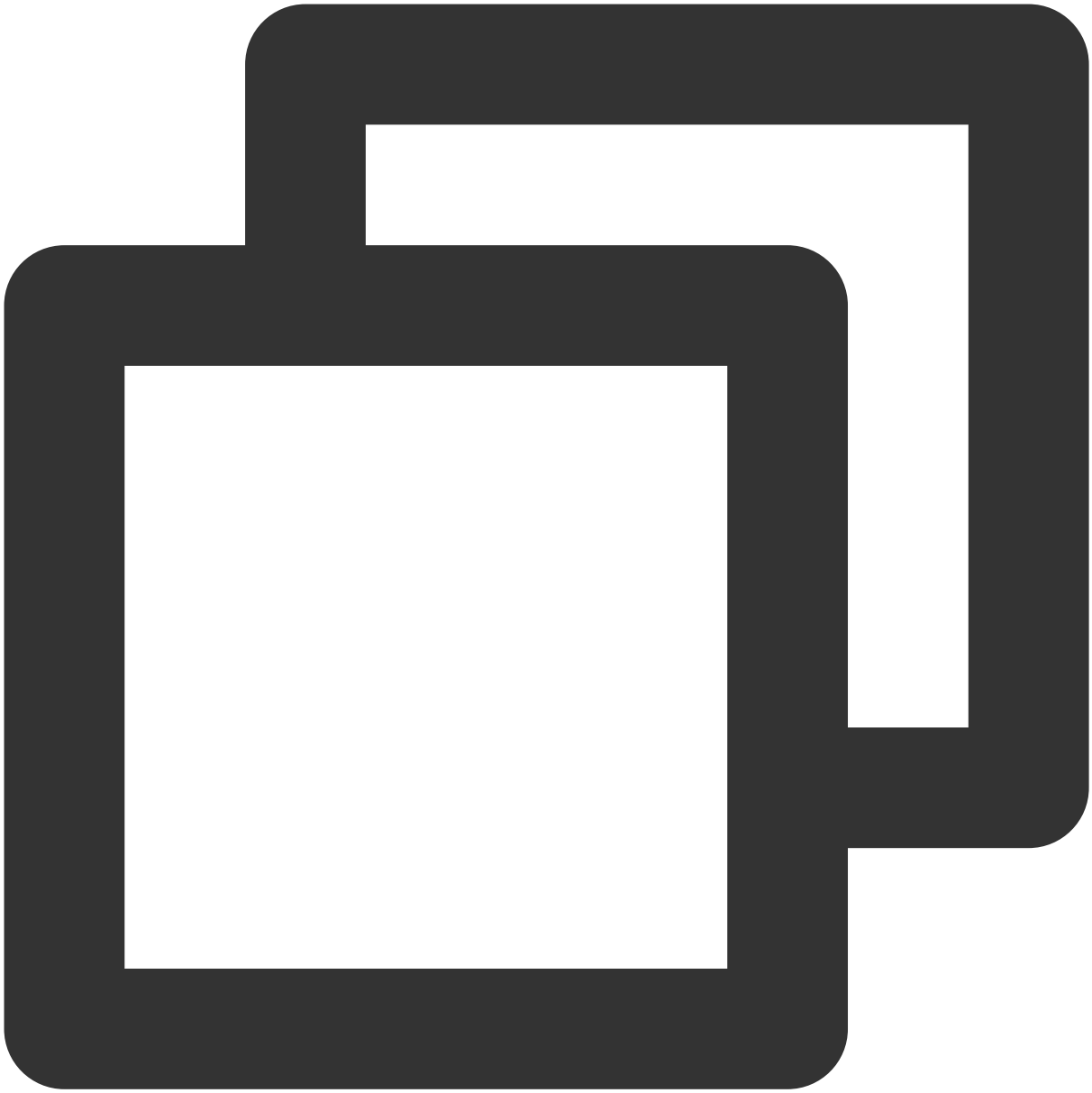
ローカルオーディオデバイスを停止します。



```
virtual int StopLocalAudio() = 0;
```

StartSystemAudioLoopback

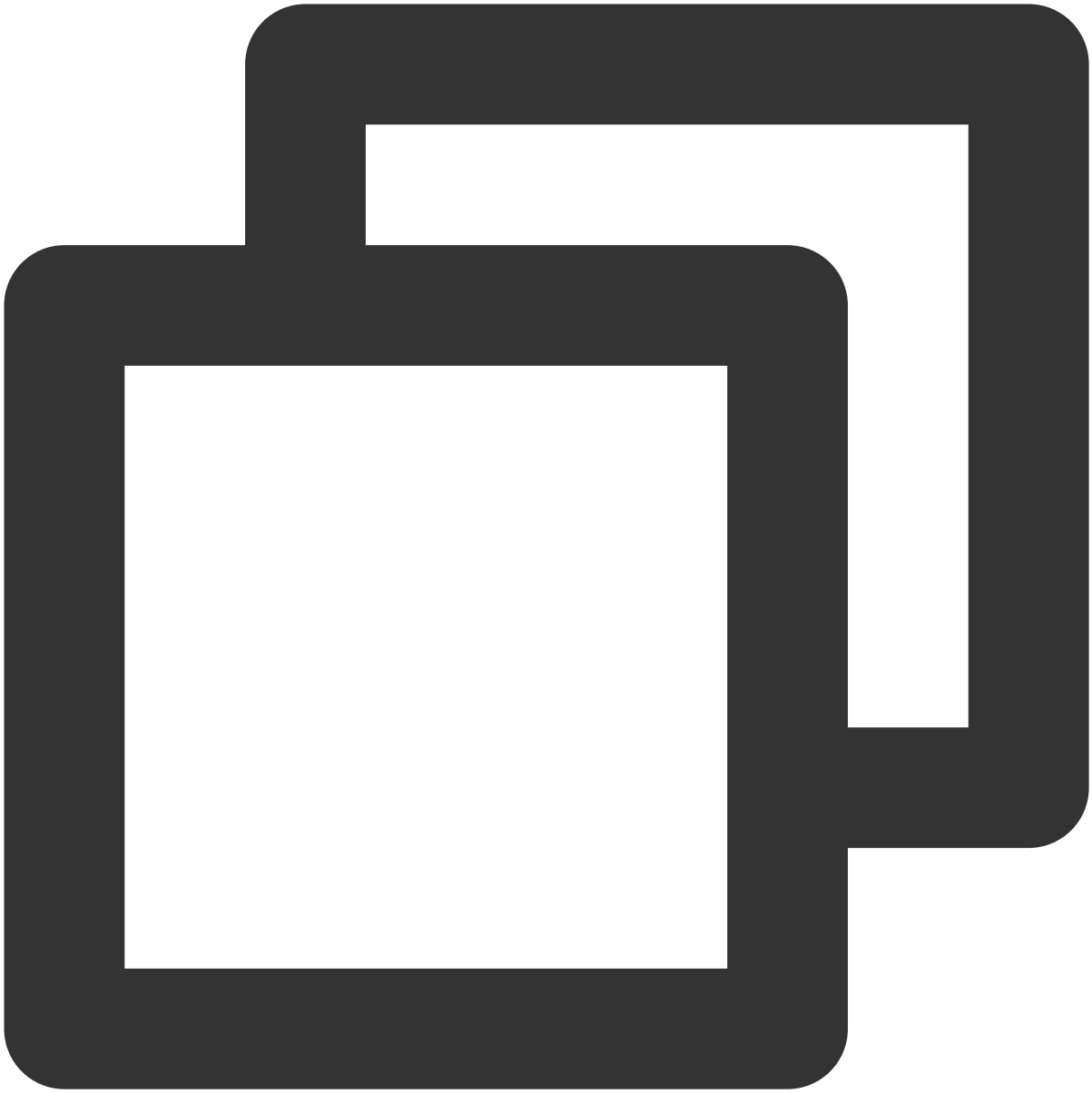
システム音声のキャプチャを開始します。



```
virtual int StartSystemAudioLoopback() = 0;
```

StopSystemAudioLoopback

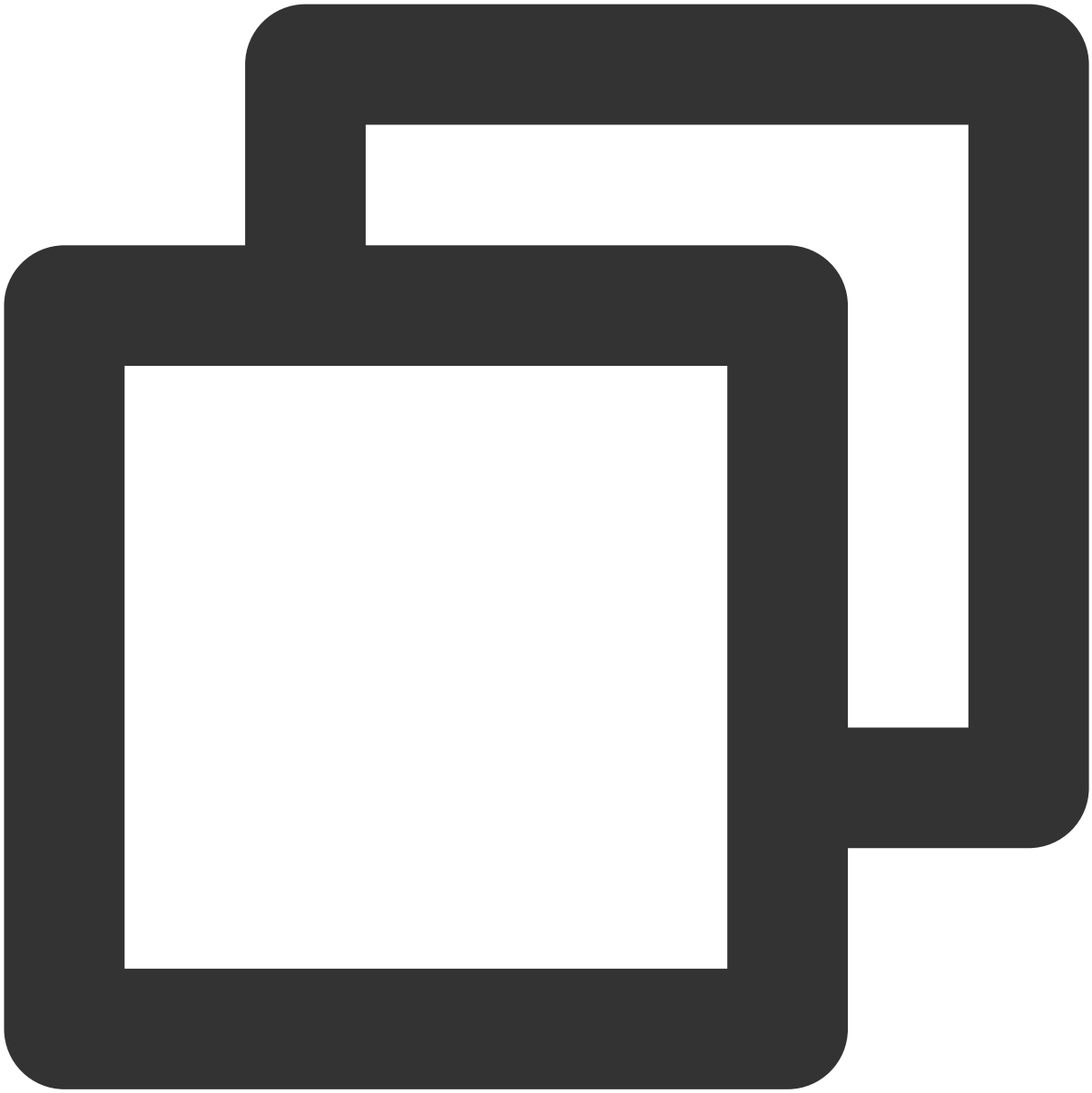
システム音声のキャプチャを停止します。



```
virtual int StopSystemAudioLoopback() = 0;
```

SetVideoMirror

イメージを設定します。



```
virtual int SetVideoMirror(bool mirror) = 0;
```

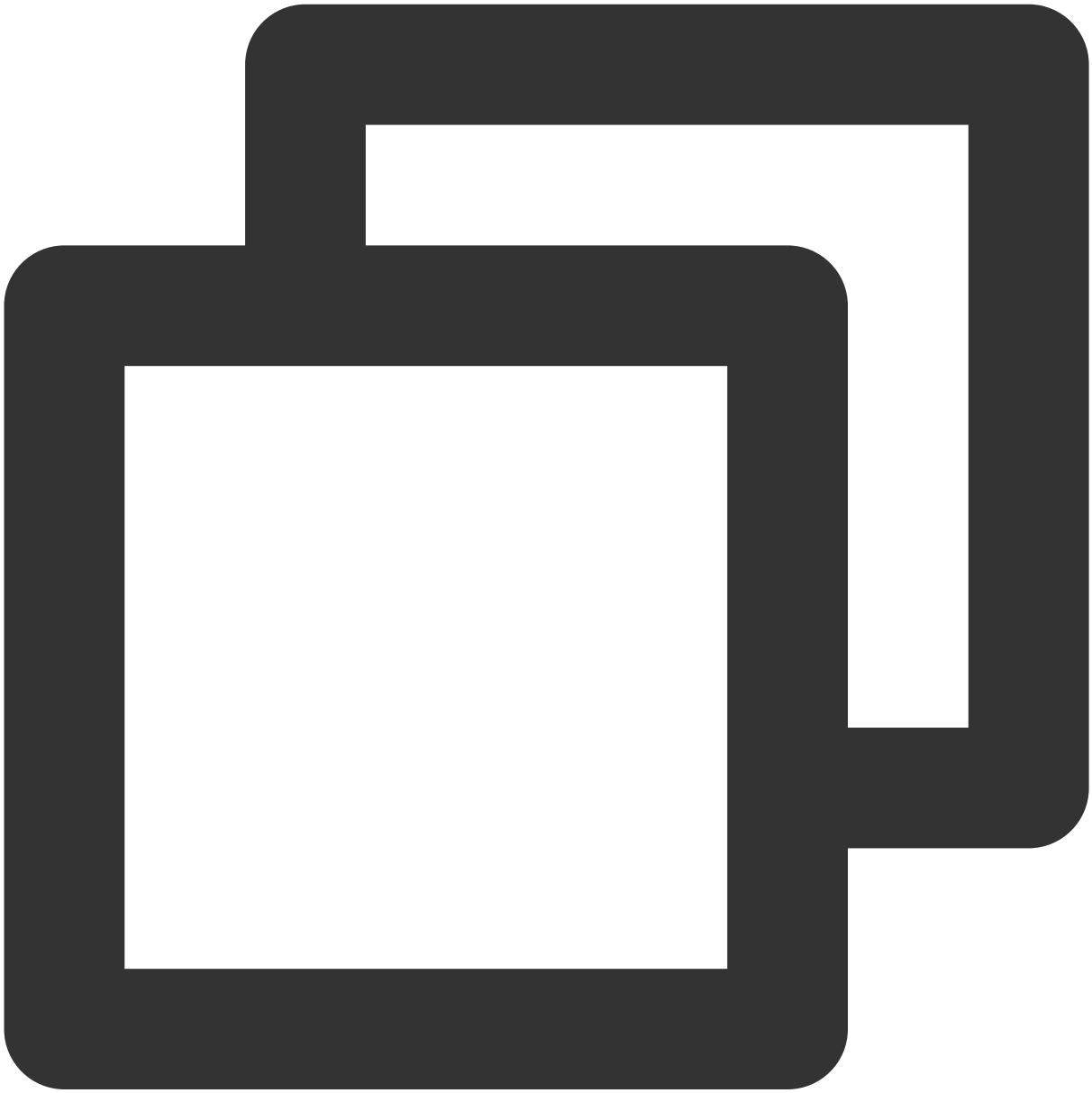
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
mirror	bool	ミラーオン/オフ。

リモートユーザーに関するインターフェース

StartRemoteView

リモートユーザーのビデオストリームのサブスクリプション。



```
virtual int StartRemoteView(const std::string& user_id, const liteav::TXView& view,
        TUIStreamType type = TUIStreamType::kStreamTypeCamera) = 0;
```

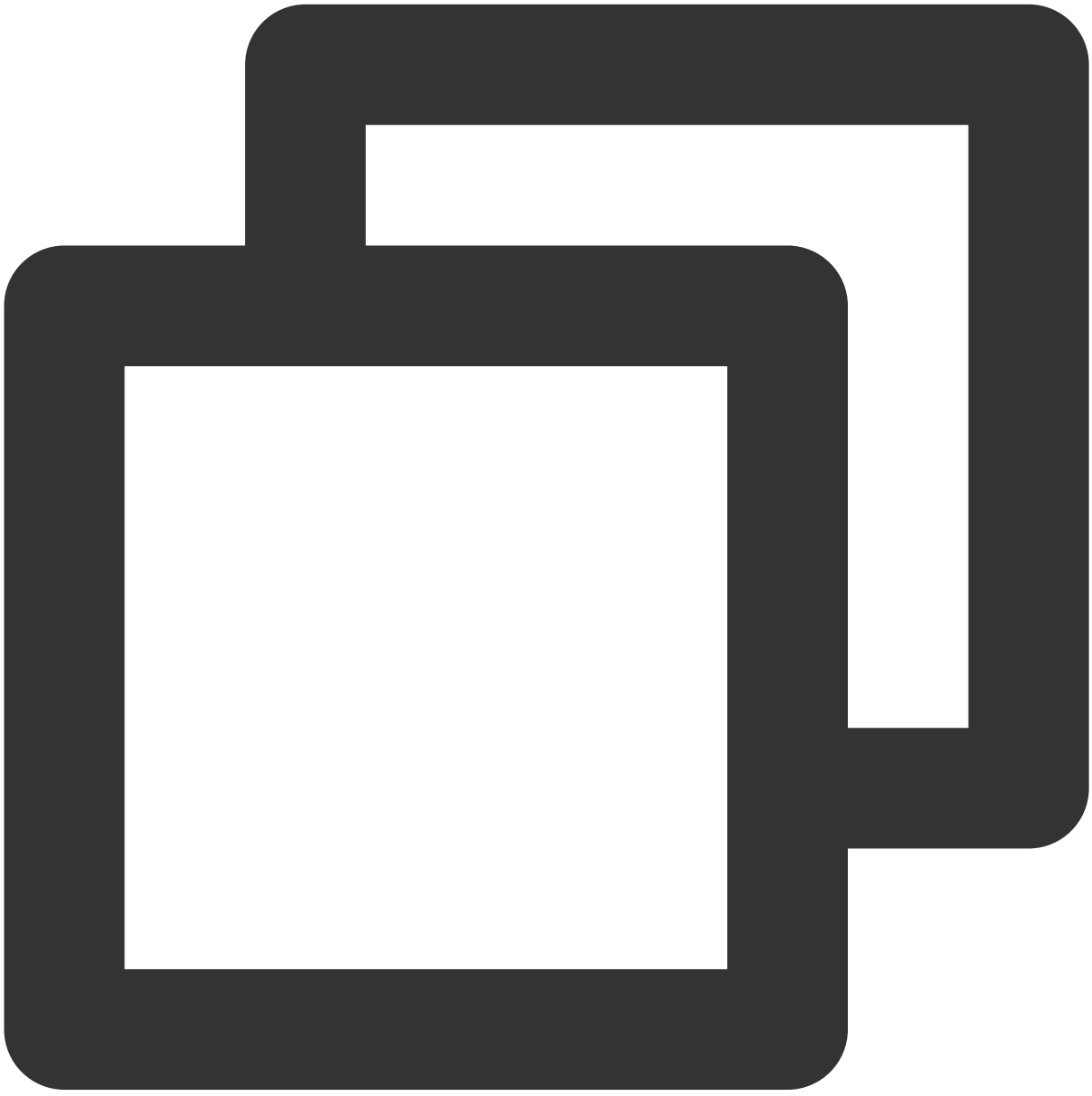
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	再生が必要なユーザー ID。

liteav::TXView	TXView	ビデオ画像をロードするviewウィジェット。
type	TUIStreamType	ストリームのタイプ。

StopRemoteView

サブスクリプションをキャンセルし、リモートビデオ画面の再生を停止します。



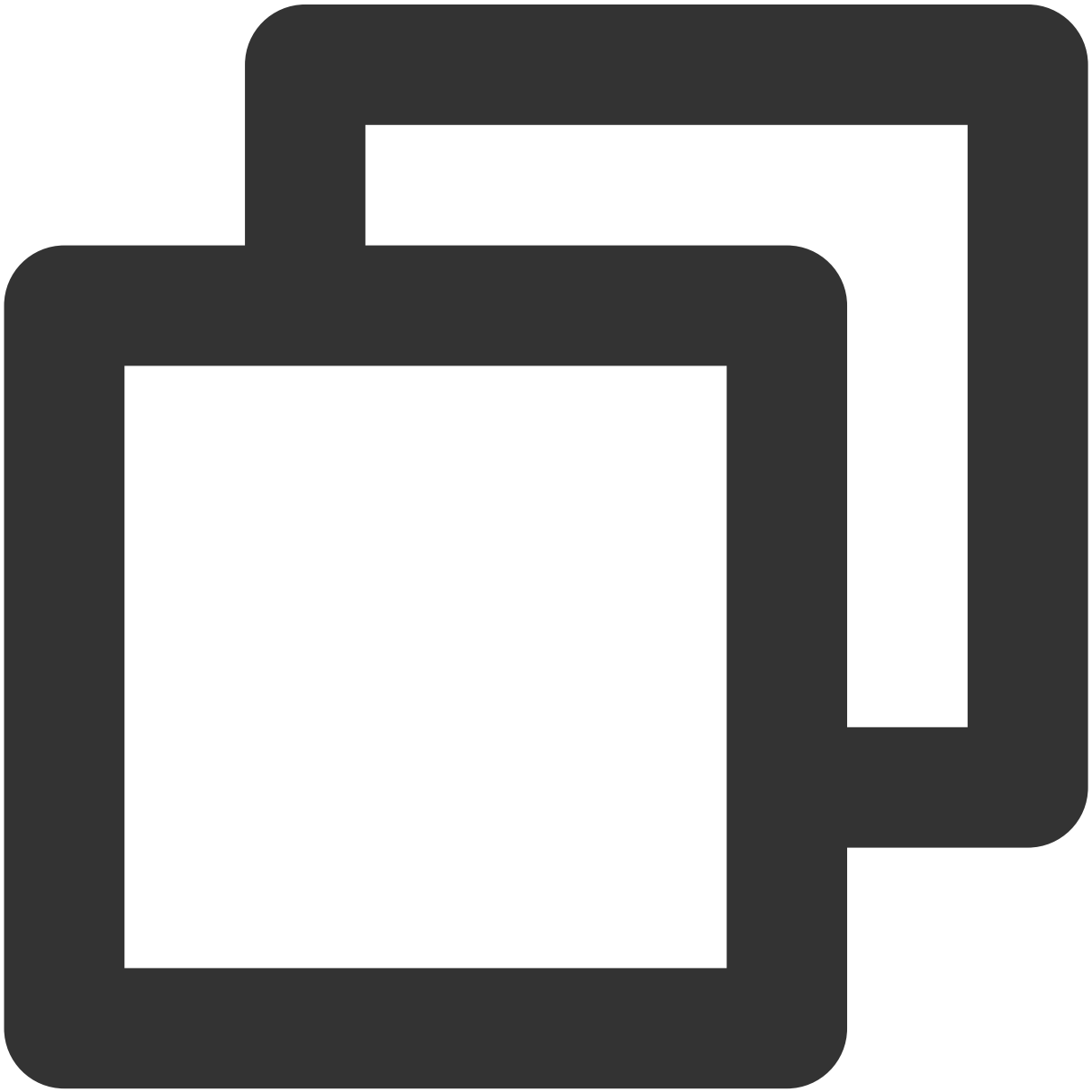
```
virtual int StopRemoteView(const std::string& user_id,
        TUIStreamType type = TUIStreamType::kStreamTypeCamera) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	再生の停止が必要なユーザー ID。
type	TUIStreamType	ストリームのタイプ。

UpdateRemoteView

リモートビデオレンダリングウィンドウを更新します。



```
virtual int UpdateRemoteView(const std::string& user_id, TUIStreamType type, liteav
```

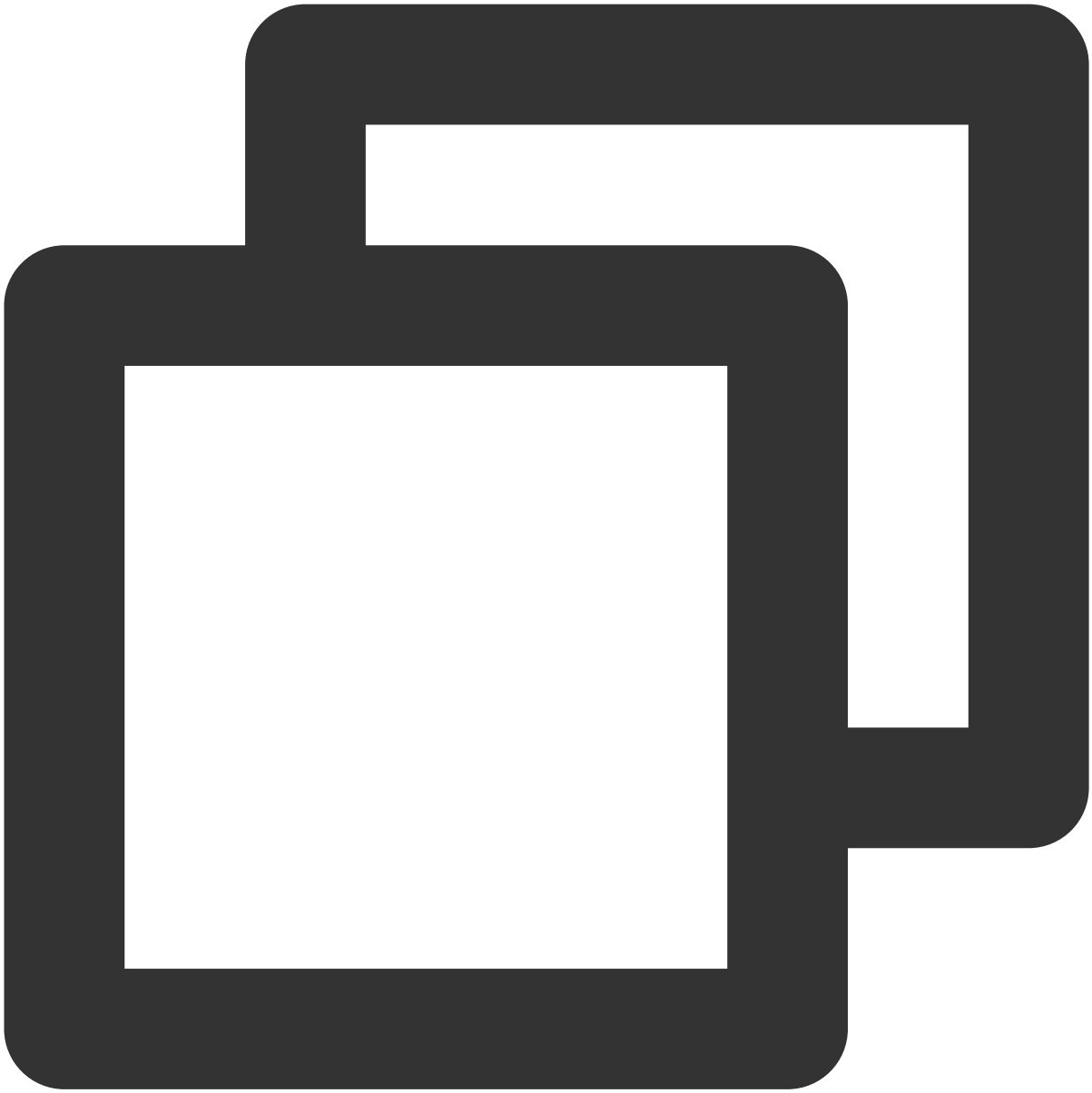
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。
type	TUIStreamType	ストリームのタイプ。
view	liteav::TXView	レンダリングウィンドウハンドル。

メッセージ送信インターフェース

SendChatMessage

テキストメッセージを送信します。



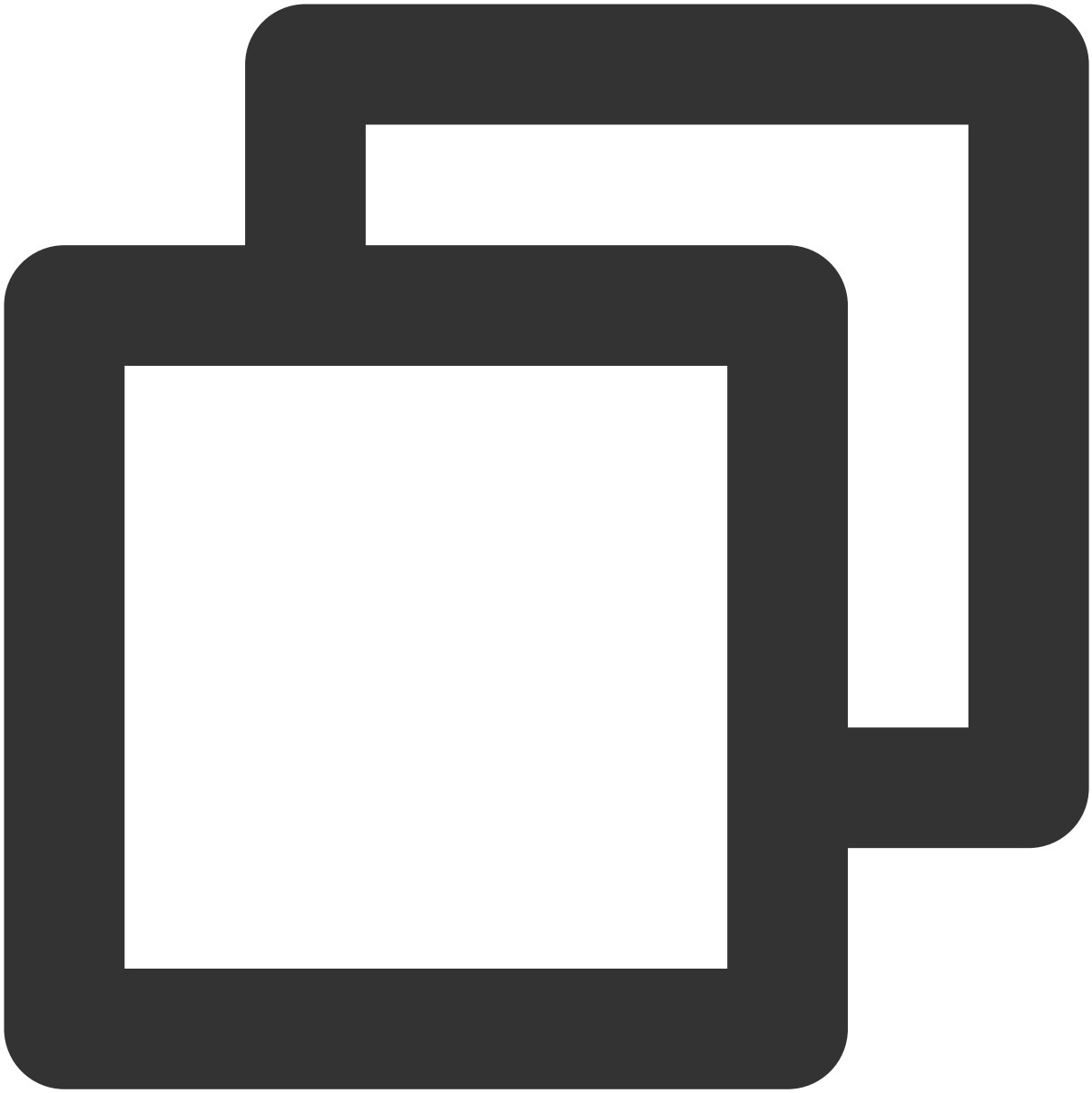
```
virtual int SendChatMessage(const std::string& message) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
message	string	メッセージの内容。

SendCustomMessage

カスタムメッセージを送信します。



```
virtual int SendCustomMessage(const std::string& message) = 0;
```

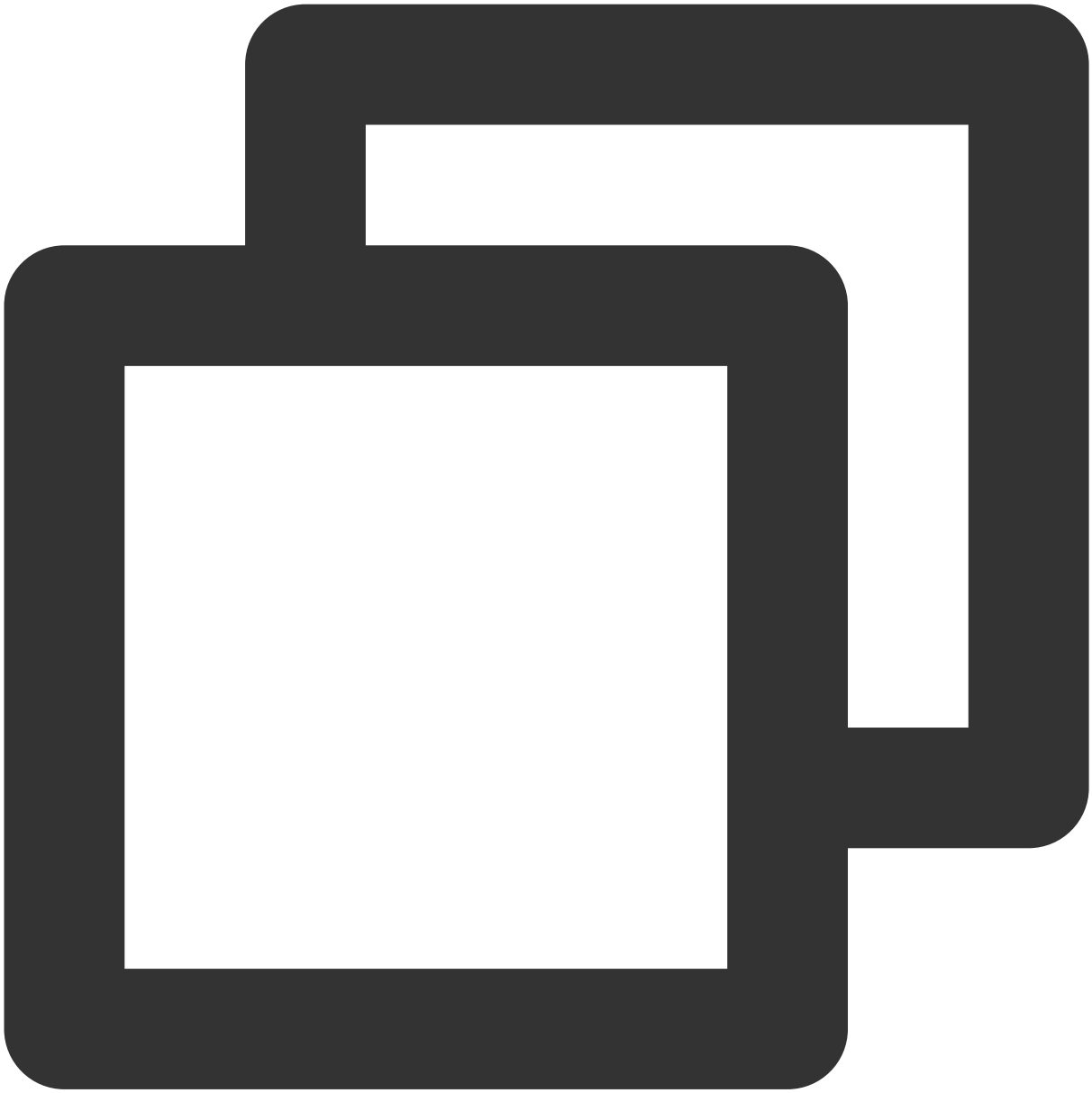
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
message	string	メッセージの内容。

フィールドコントロール関連インターフェース

MuteUserMicrophone

特定ユーザーのマイクを無効化/再有効化します。



```
virtual int MuteUserMicrophone(const std::string& user_id, bool mute, Callback call
```

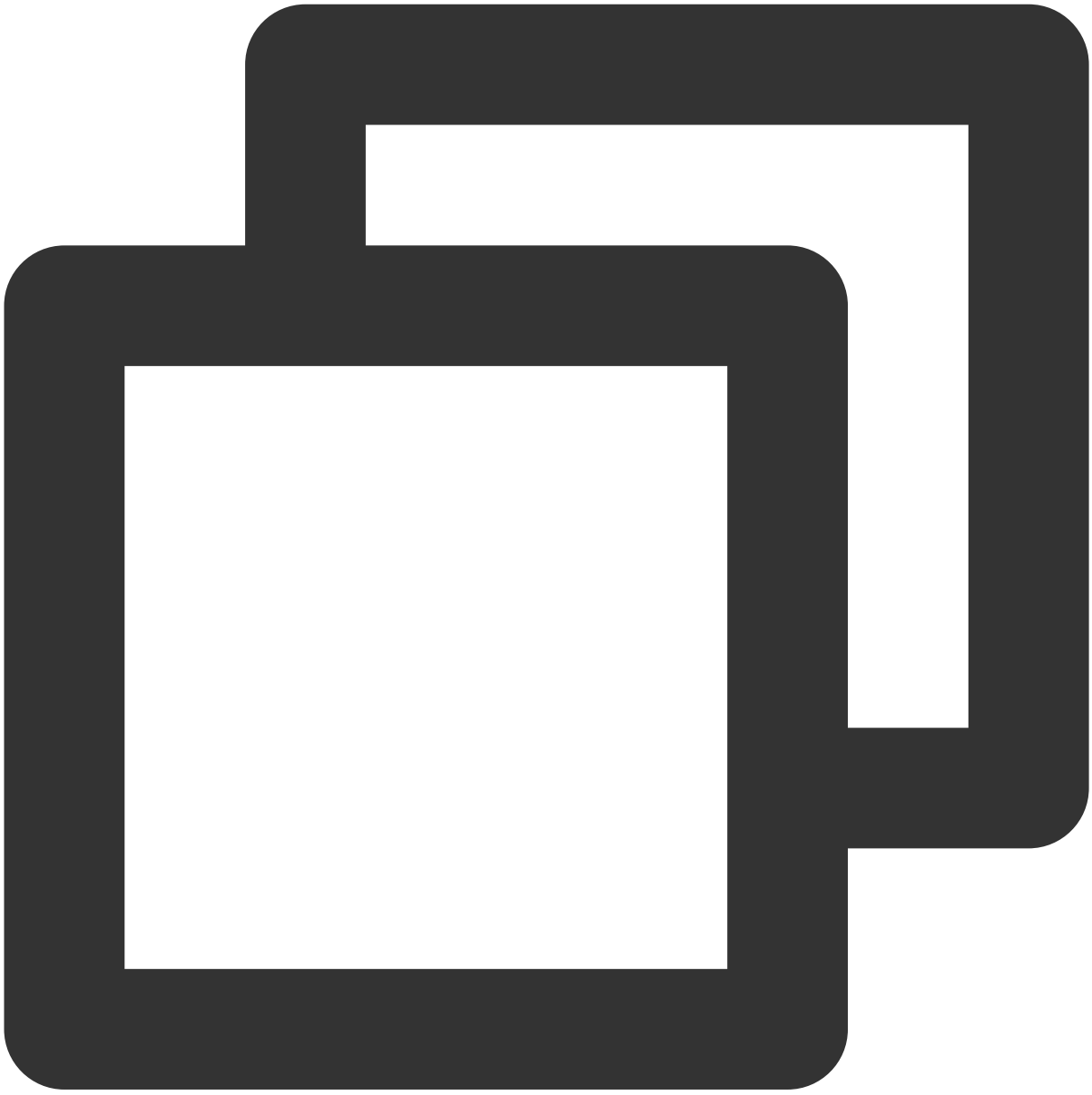
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。
mute	bool	無効にするかどうか。

callback	Callback	インターフェースコールバック。

MuteAllUsersMicrophone

全ユーザーのマイクを無効化/再有効化します。



```
virtual int MuteAllUsersMicrophone(bool mute) = 0;
```

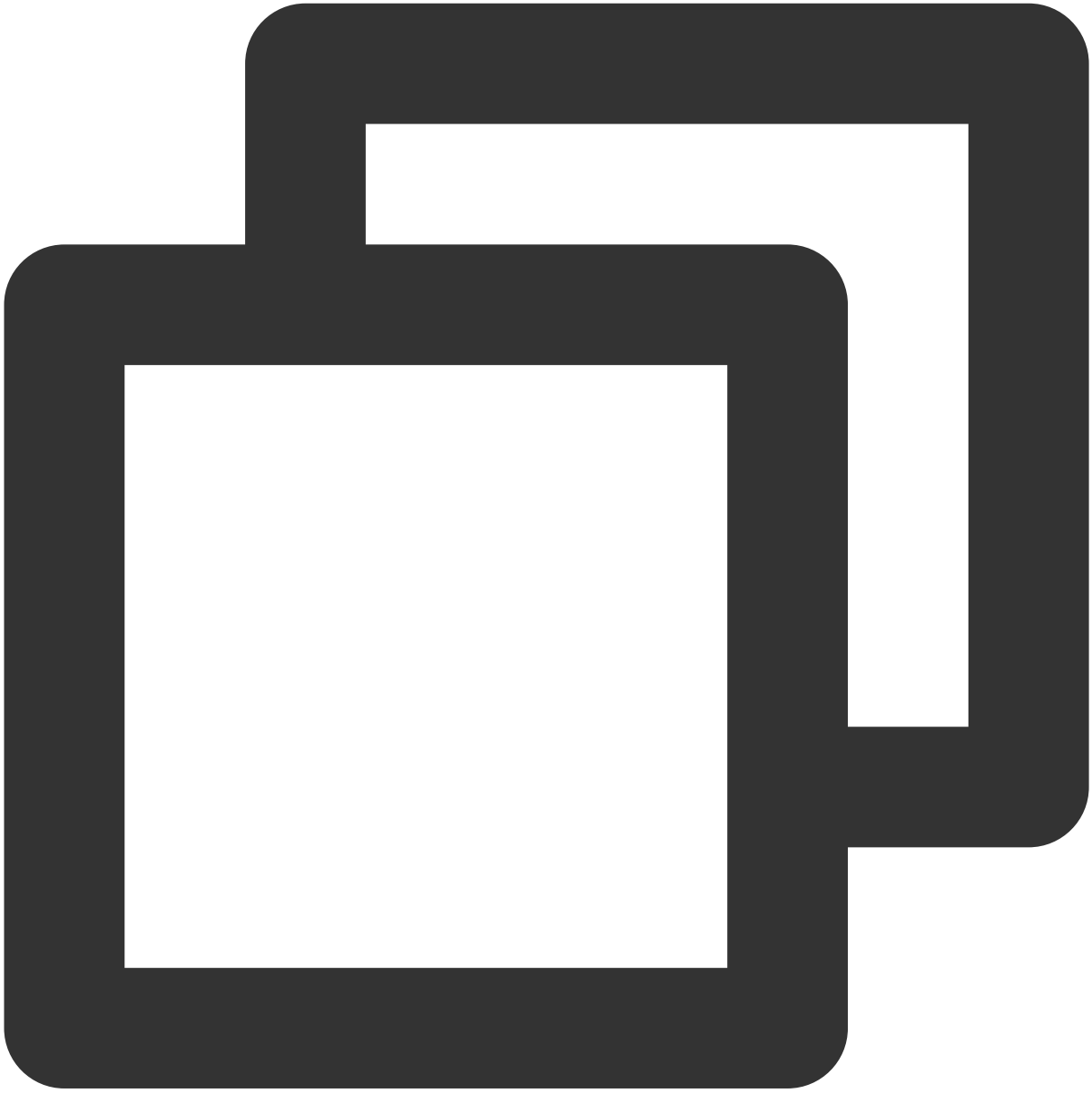
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

mute	bool	無効にするかどうか。

MuteUserCamera

特定ユーザーのカメラを無効化/再有効化します。



```
virtual int MuteUserCamera(const std::string& user_id, bool mute, Callback callback
```

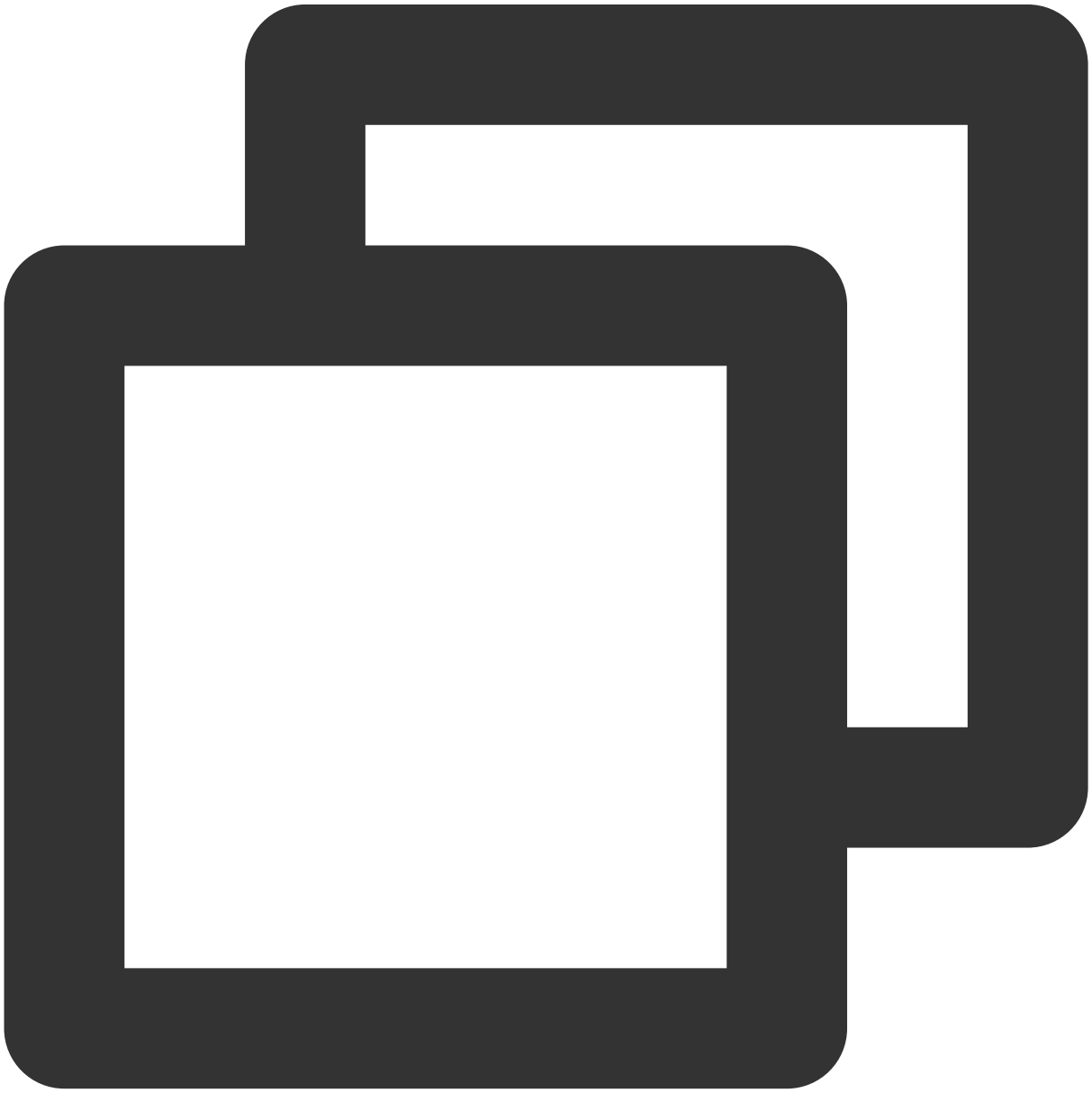
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

user_id	string	ユーザー ID。
mute	bool	無効にするかどうか。
callback	Callback	インターフェースコールバック。

MuteAllUsersCamera

全ユーザーのカメラを無効化/再有効化します。



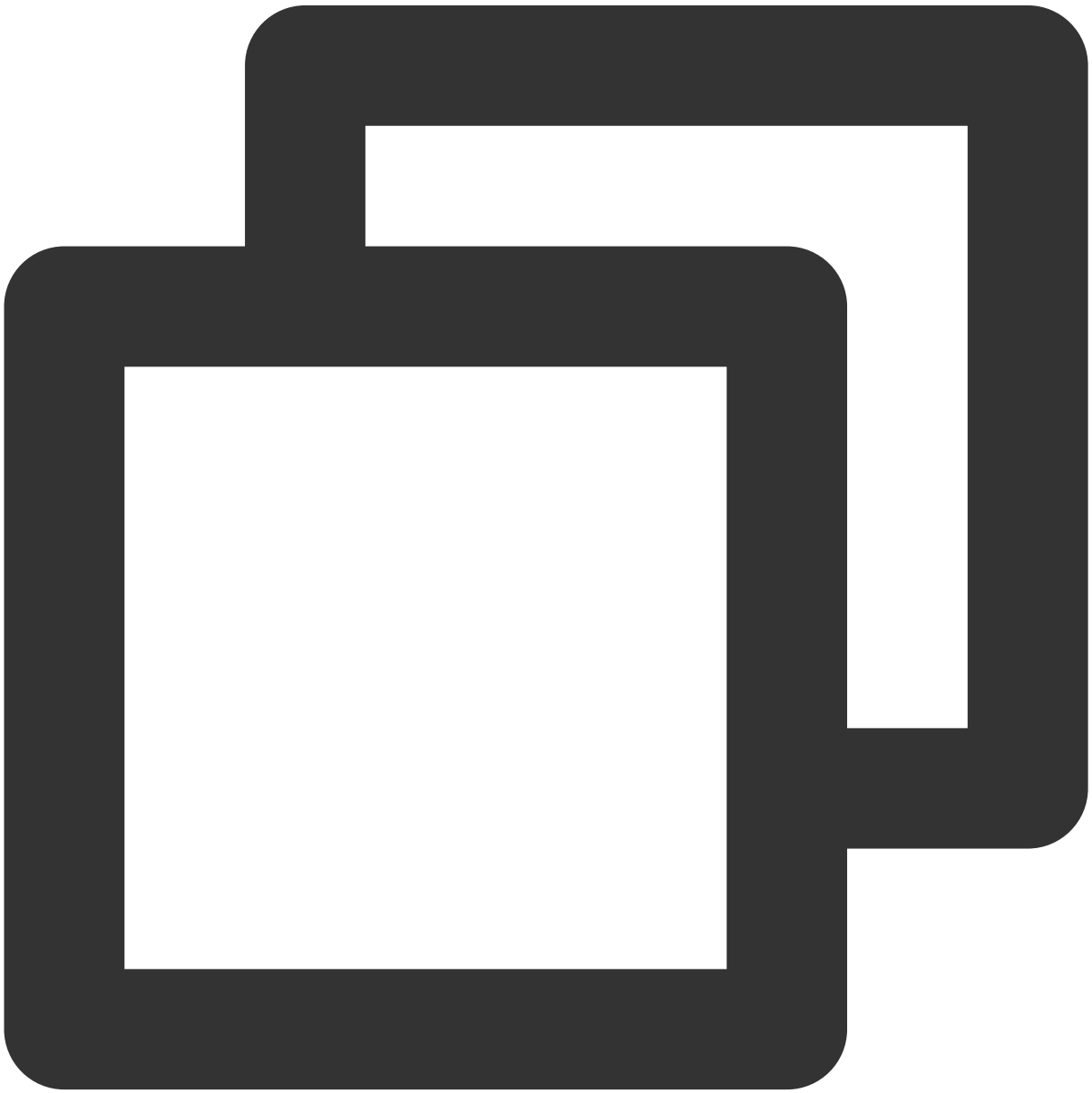
```
virtual int MuteAllUsersCamera(bool mute) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
mute	bool	無効にするかどうか。

MuteChatRoom

チャットルームのミュートを開始/停止します。



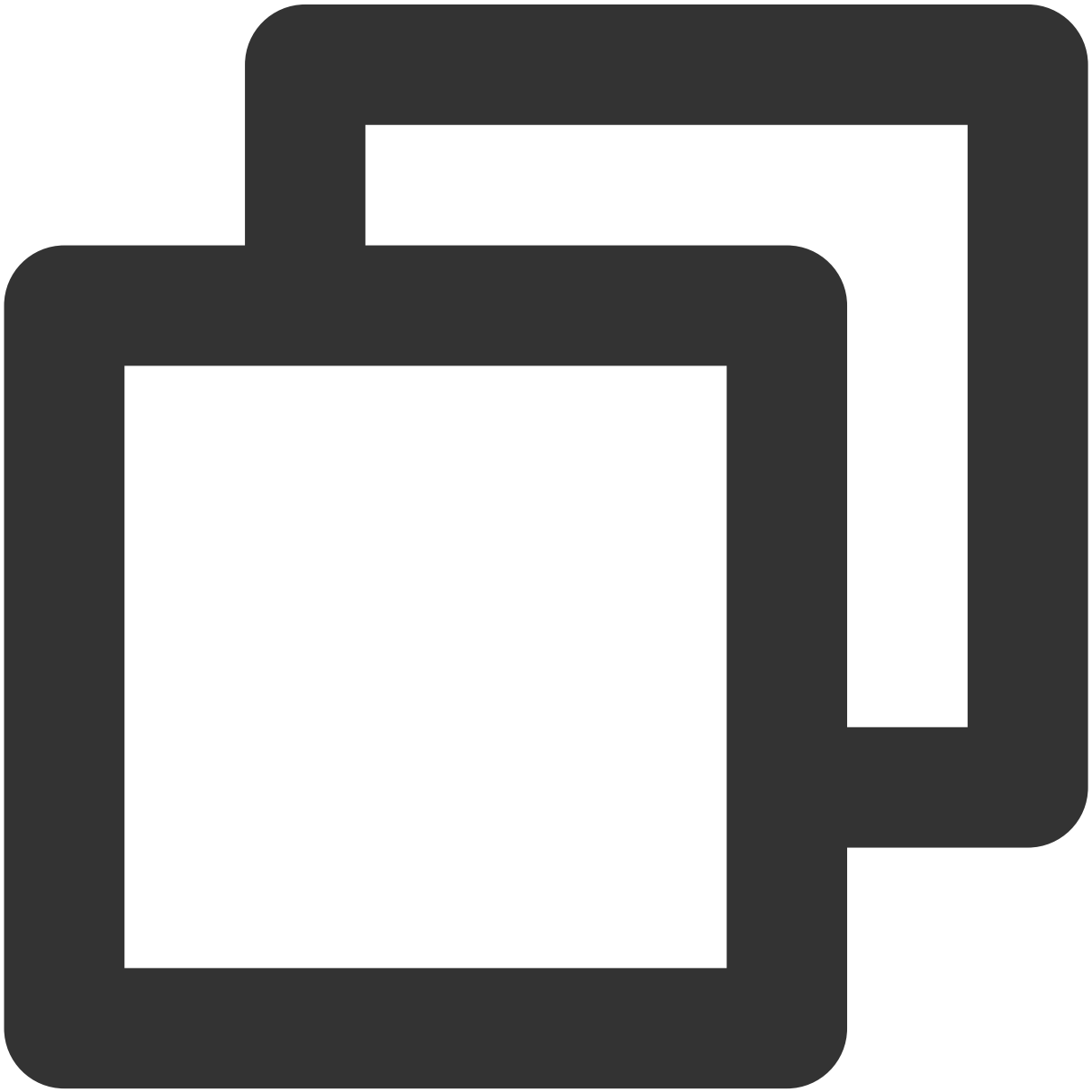
```
virtual int MuteChatRoom(bool mute) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
mute	bool	無効にするかどうか。

KickOffUser

キャスターがキックアウトします。



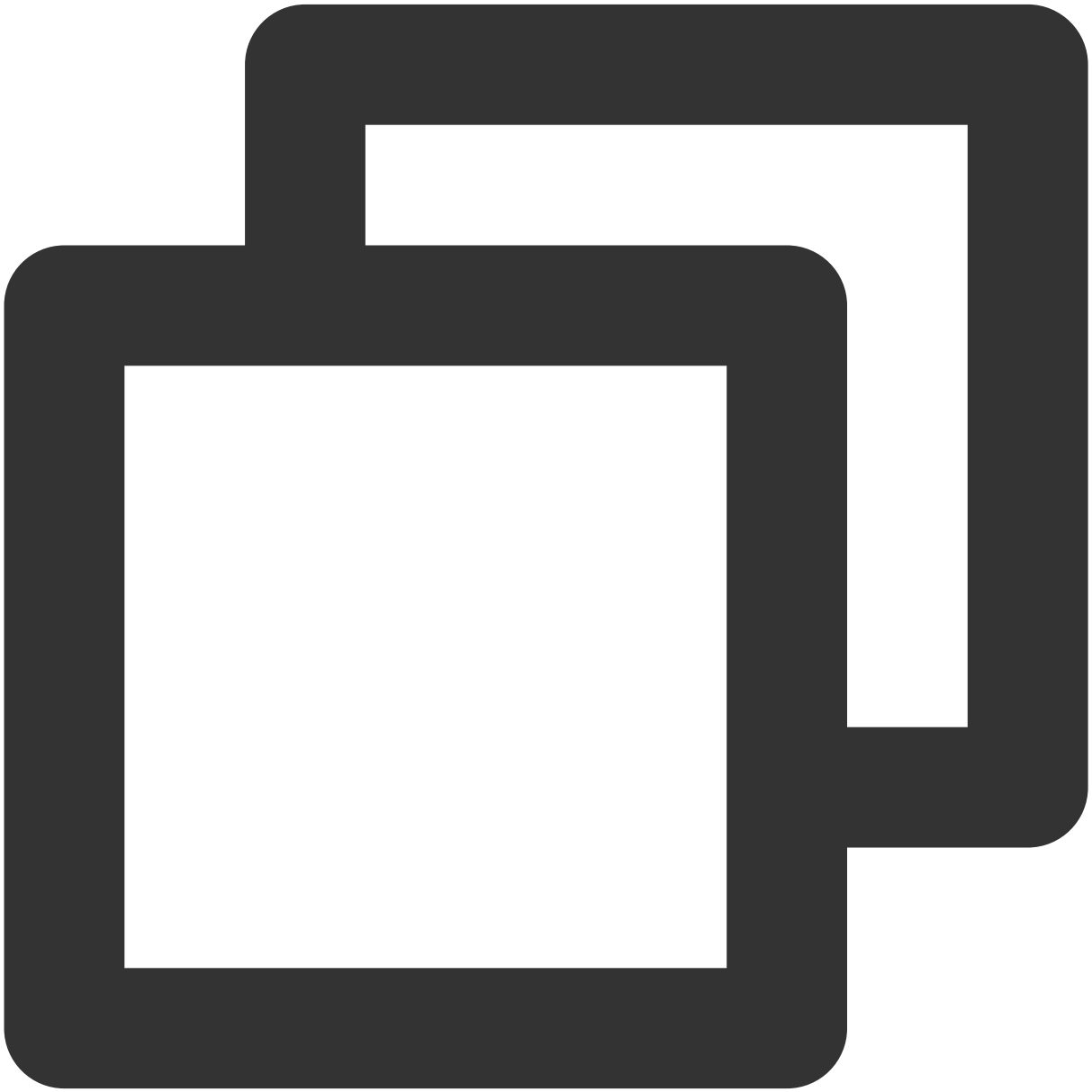
```
virtual int KickOffUser(const std::string& user_id, Callback callback) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。
callback	Callback	インターフェースコールバック。

StartCallingRoll

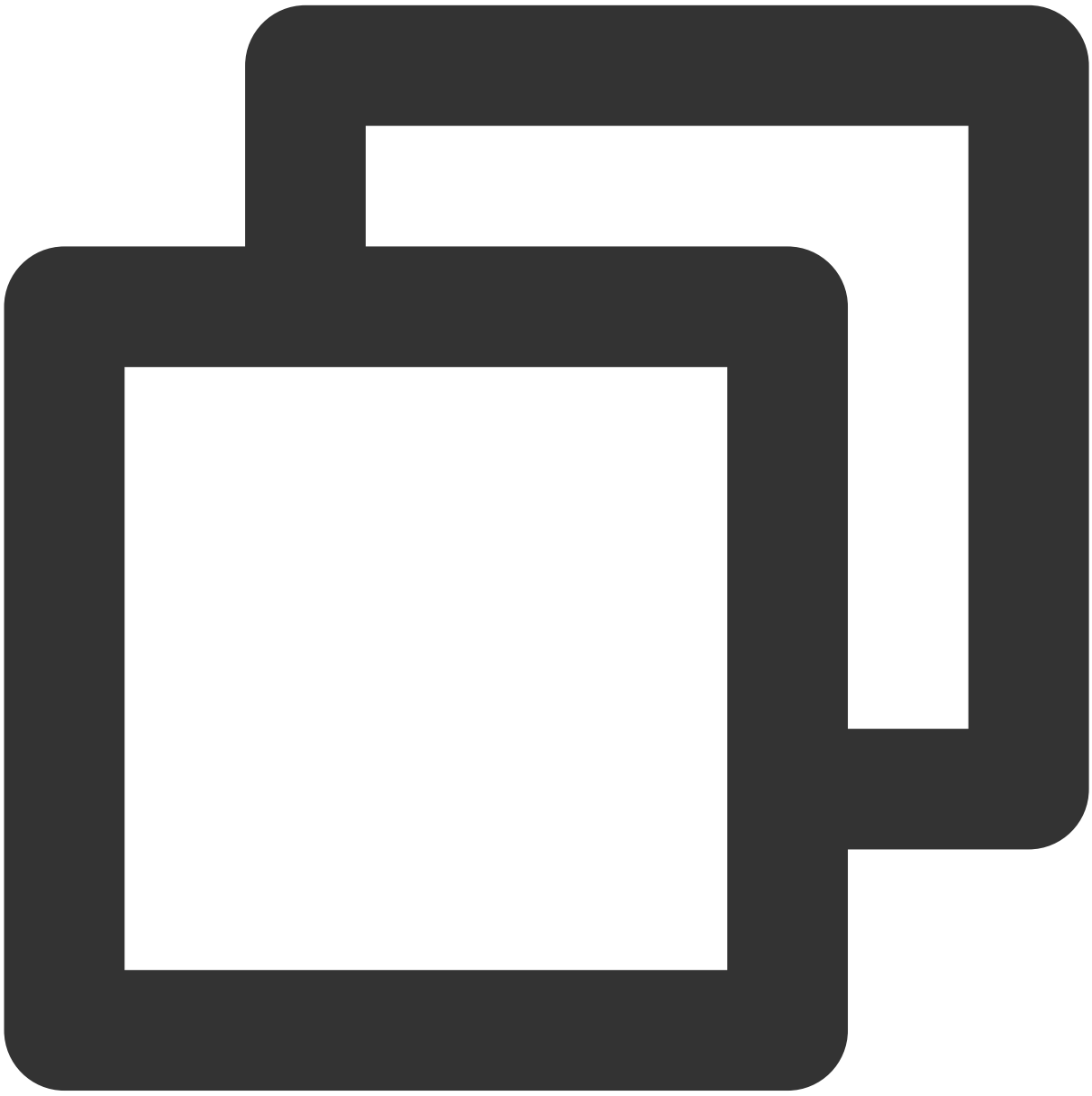
キャスターが点呼を開始します。



```
virtual int StartCallingRoll() = 0;
```

StopCallingRoll

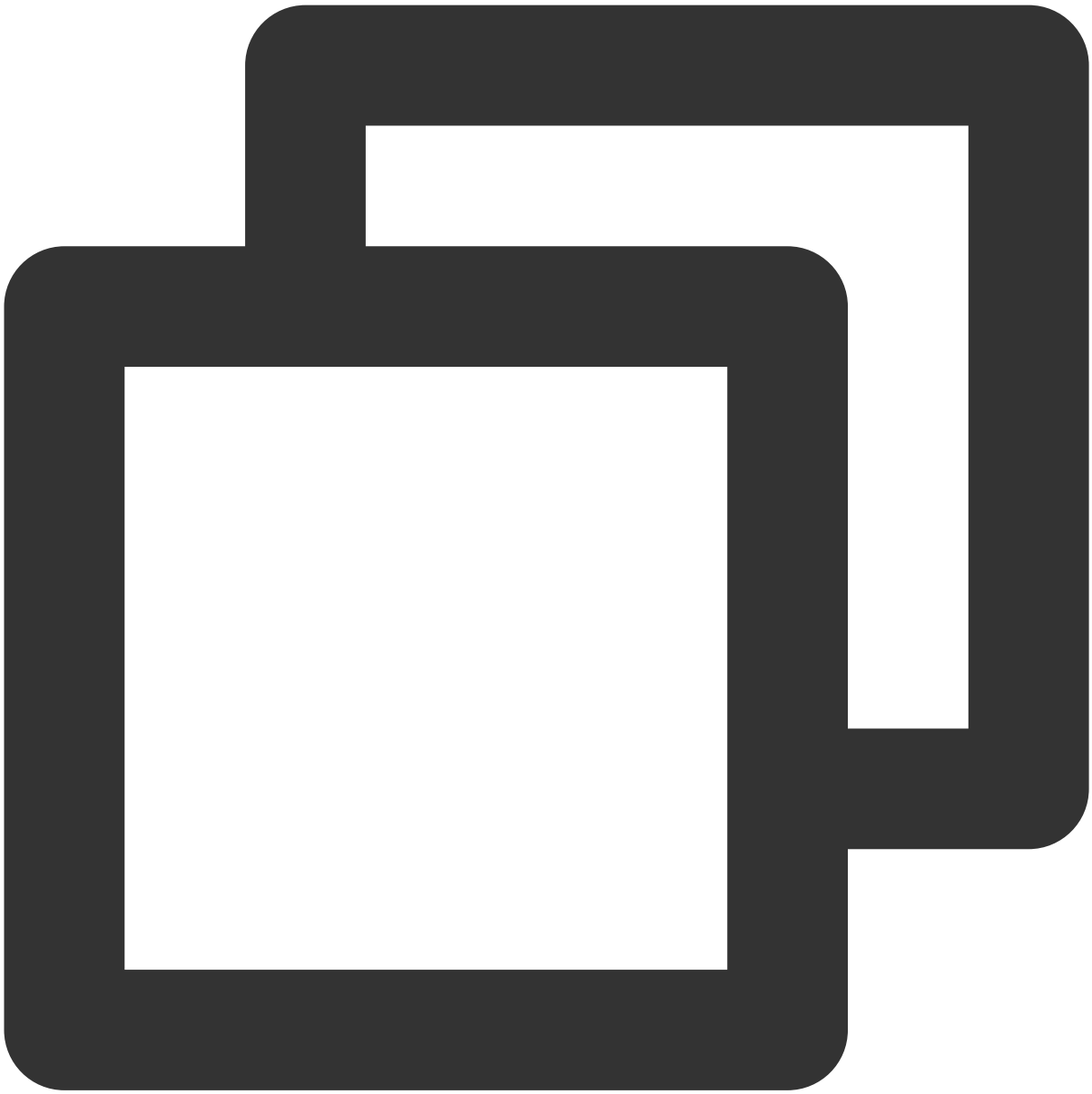
キャスターが点呼を終了します。



```
virtual int StopCallingRoll() = 0;
```

ReplyCallingRoll

参加者がキャスターの点呼に応答します。



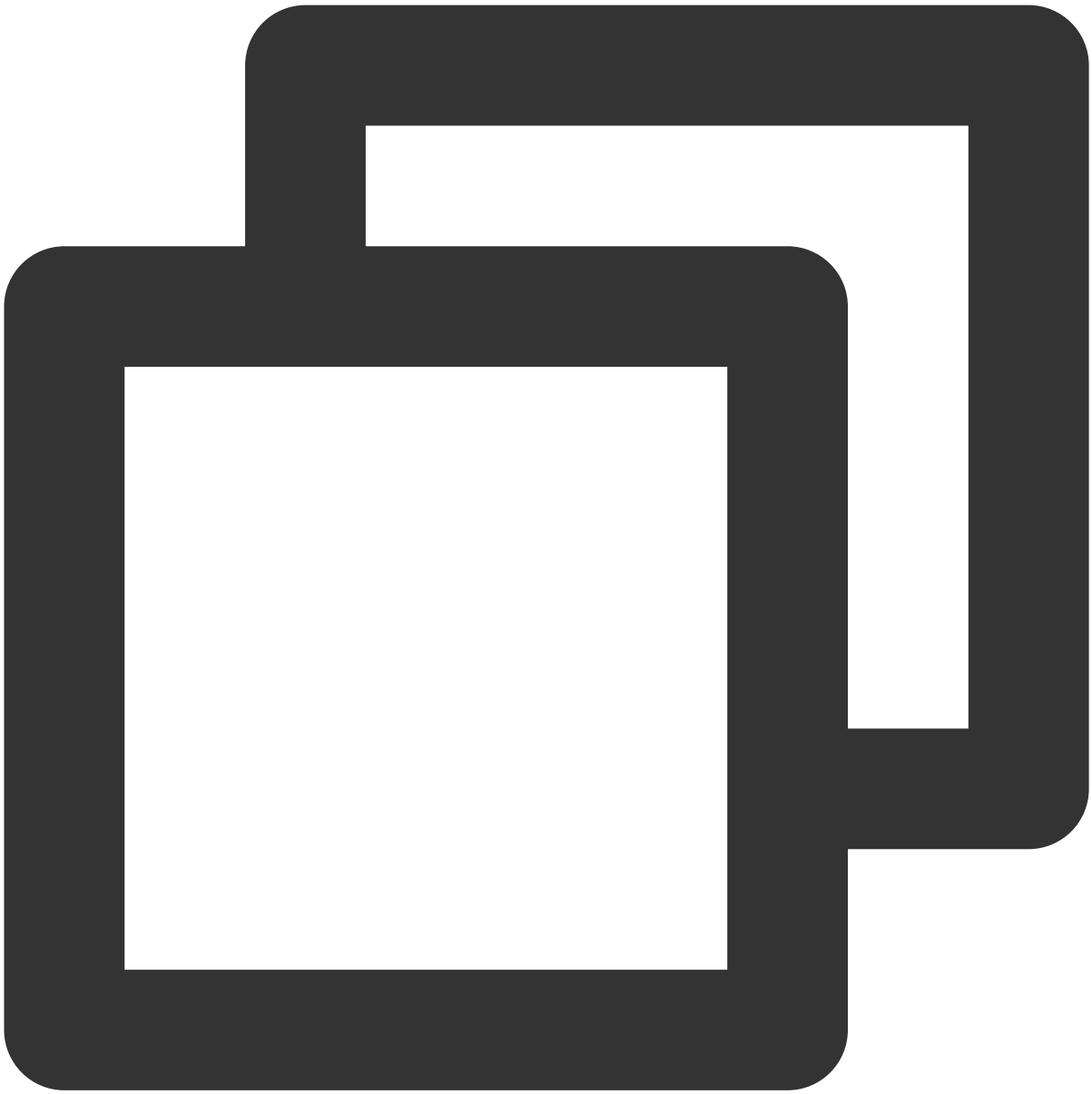
```
virtual int ReplyCallingRoll(Callback callback) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	Callback	インターフェースコールバック。

SendSpeechInvitation

キャスターが参加者の発言を要請します。



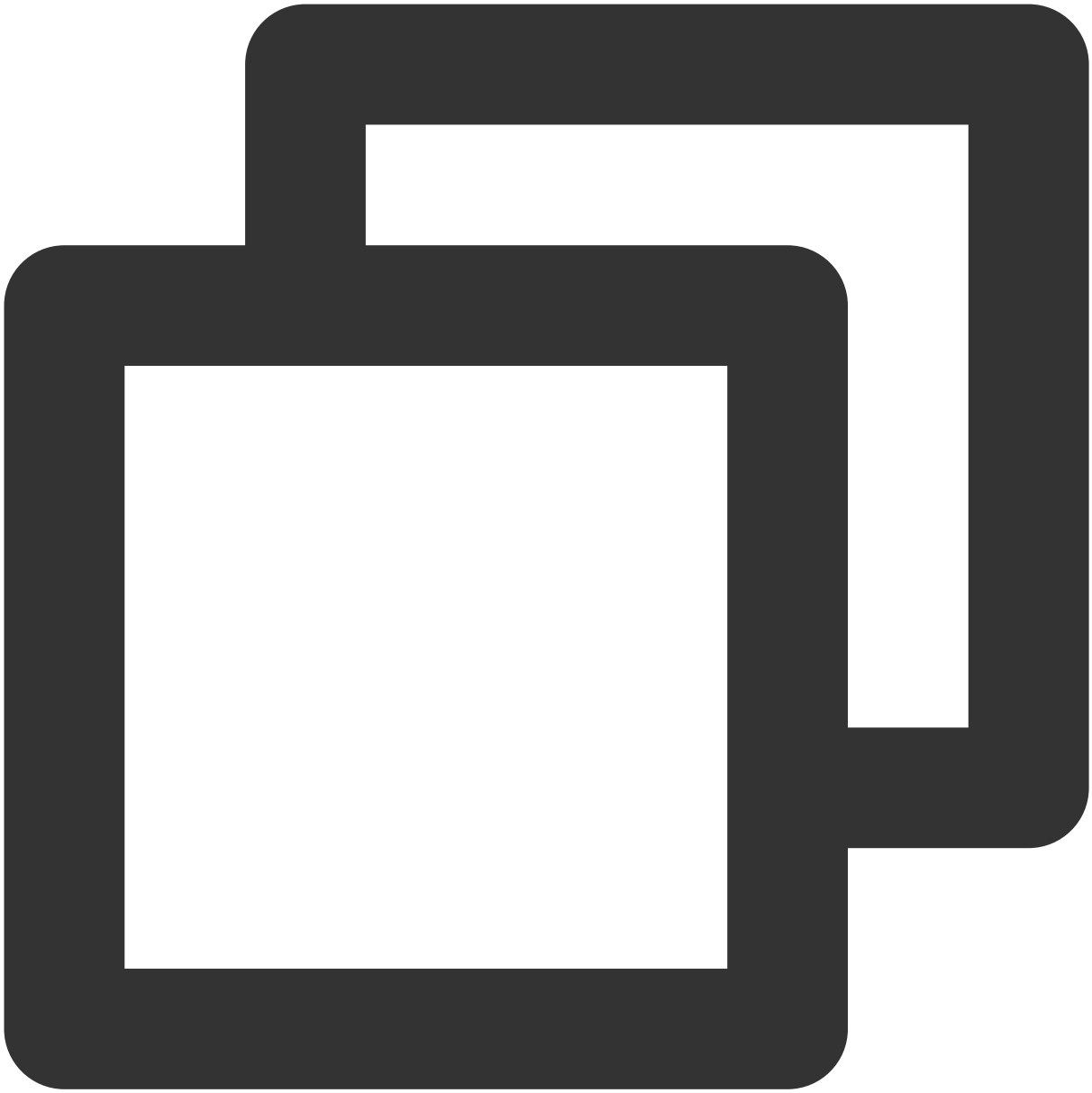
```
virtual int SendSpeechInvitation(const std::string& user_id, Callback callback) = 0
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。
callback	Callback	インターフェースコールバック。

CancelSpeechInvitation

キャスターが参加者の発言要請をキャンセルします。



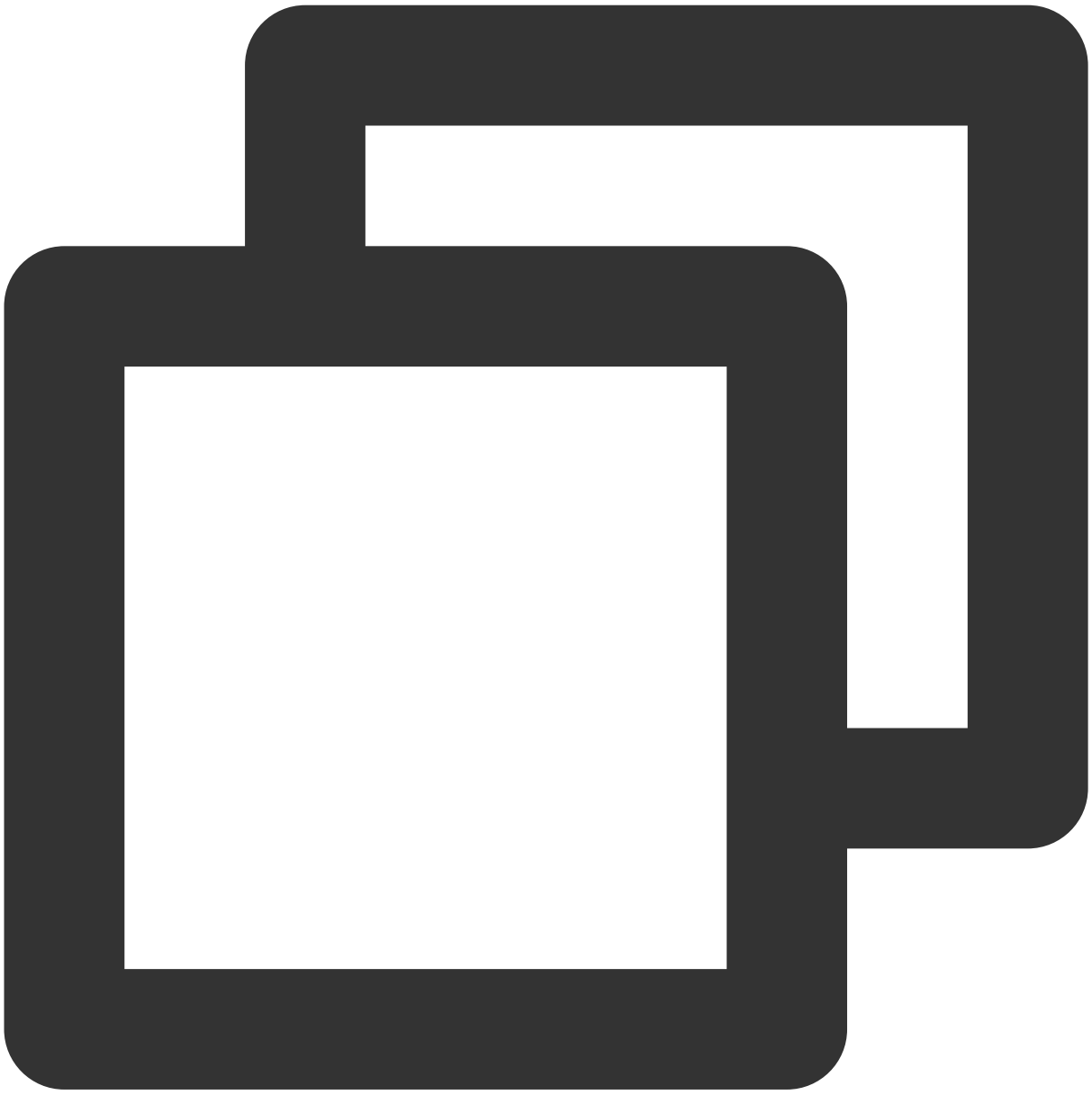
```
virtual int CancelSpeechInvitation(const std::string& user_id, Callback callback) =
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。
callback	Callback	インターフェースコールバック。

ReplySpeechInvitation

参加者がキャスターの発言要請に同意/拒否します。



```
virtual int ReplySpeechInvitation(bool agree, Callback callback) = 0;
```

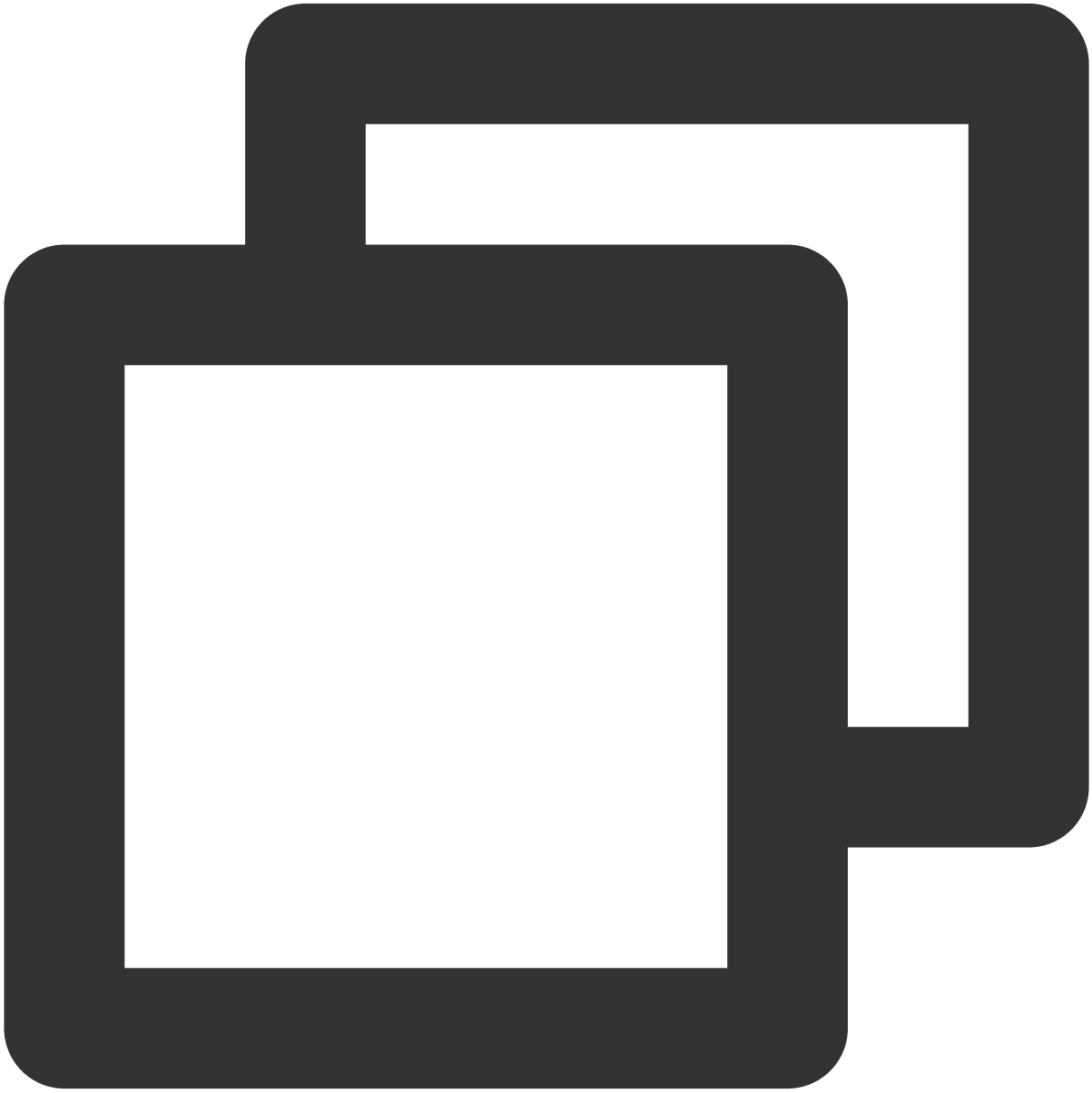
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
agree	bool	同意するかどうか。

callback	Callback	インターフェースコールバック。

SendSpeechApplication

参加者が発言を申請します。



```
virtual int SendSpeechApplication(Callback callback) = 0;
```

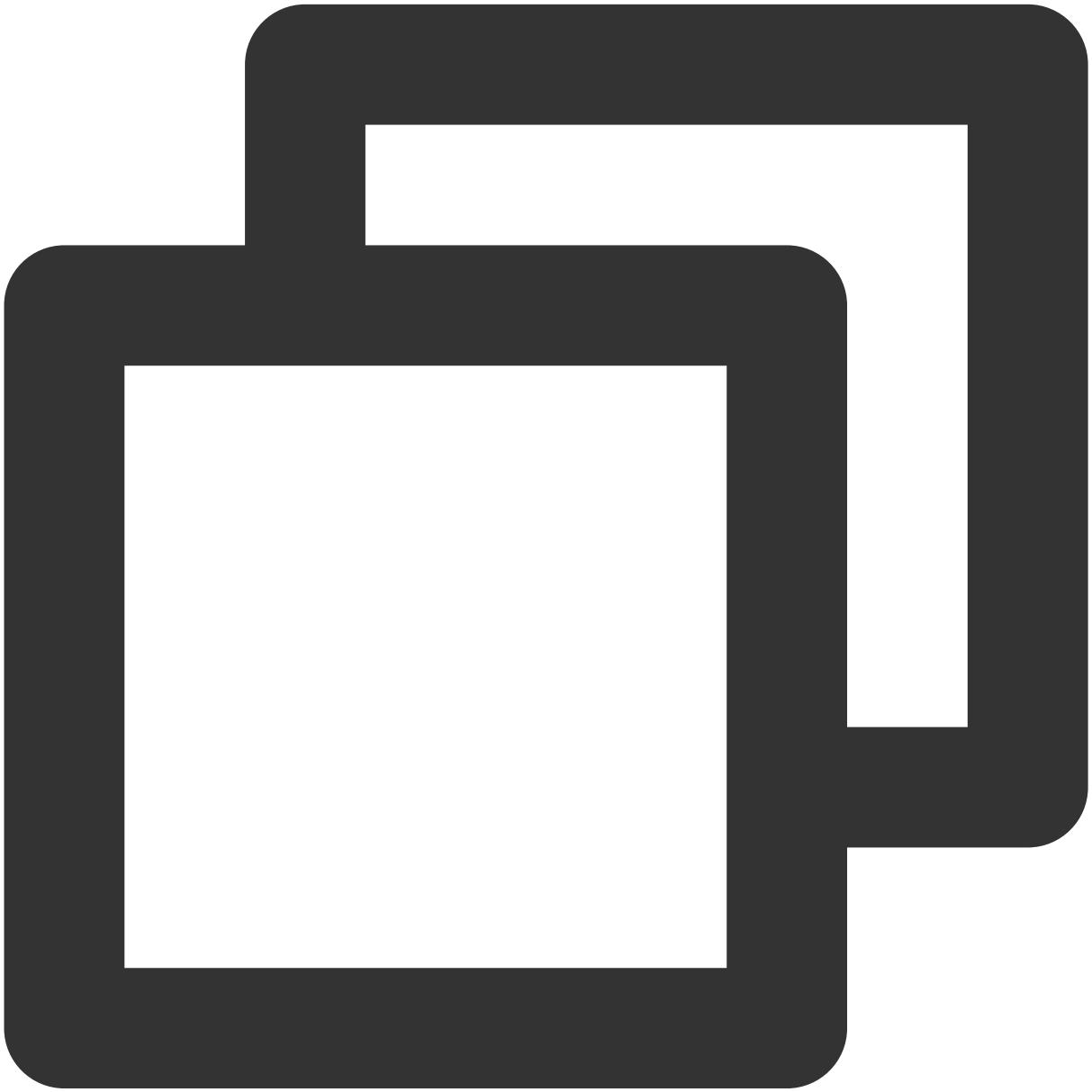
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

callback	Callback	インターフェースコールバック。

CancelSpeechApplication

参加者が発言申請をキャンセルします。



```
virtual int CancelSpeechApplication(Callback callback) = 0;
```

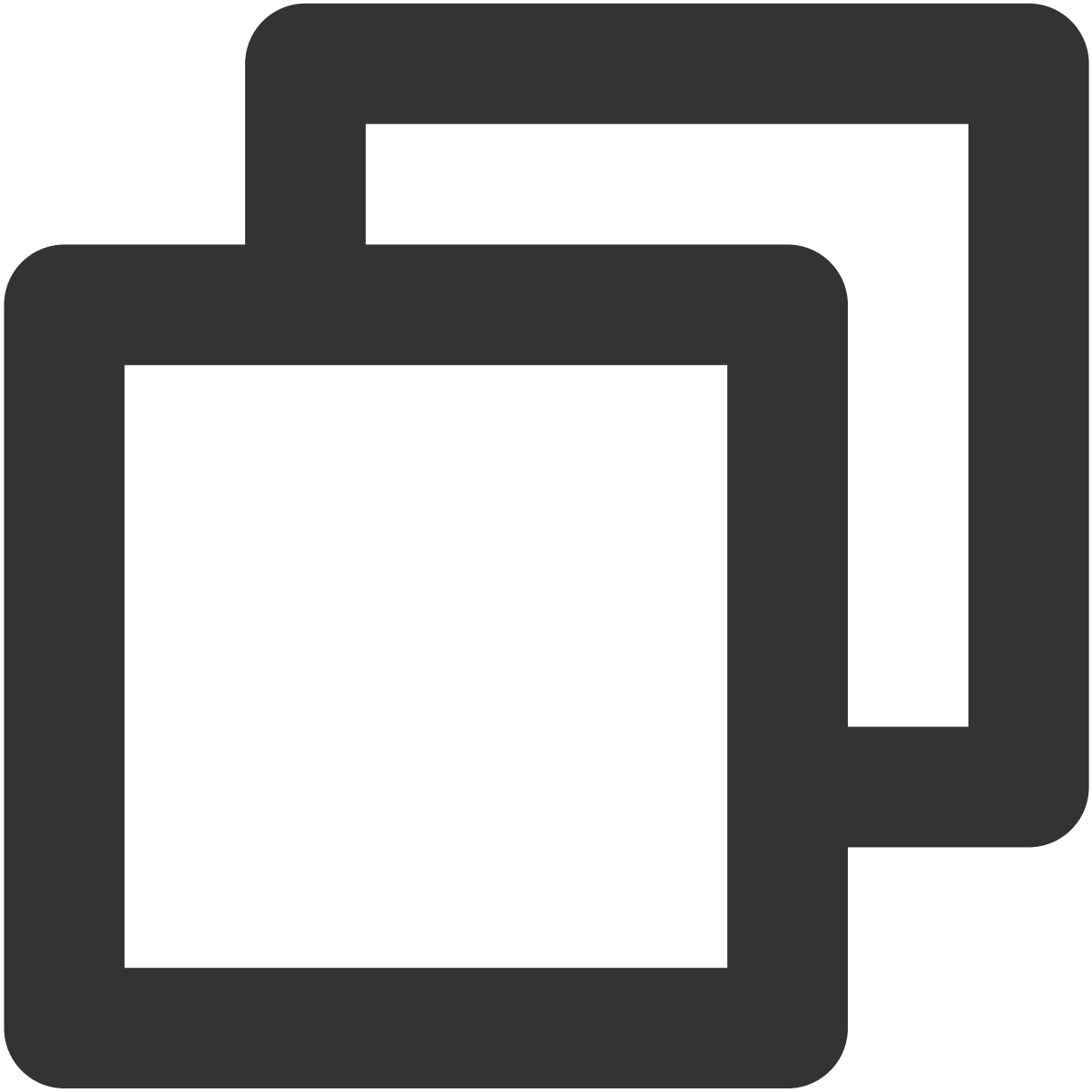
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

callback	Callback	インターフェースコールバック。

ReplySpeechApplication

キャスターが参加者の発言申請に同意/拒否します。



```
virtual int ReplySpeechApplication(const std::string& user_id, bool agree, Callback
```

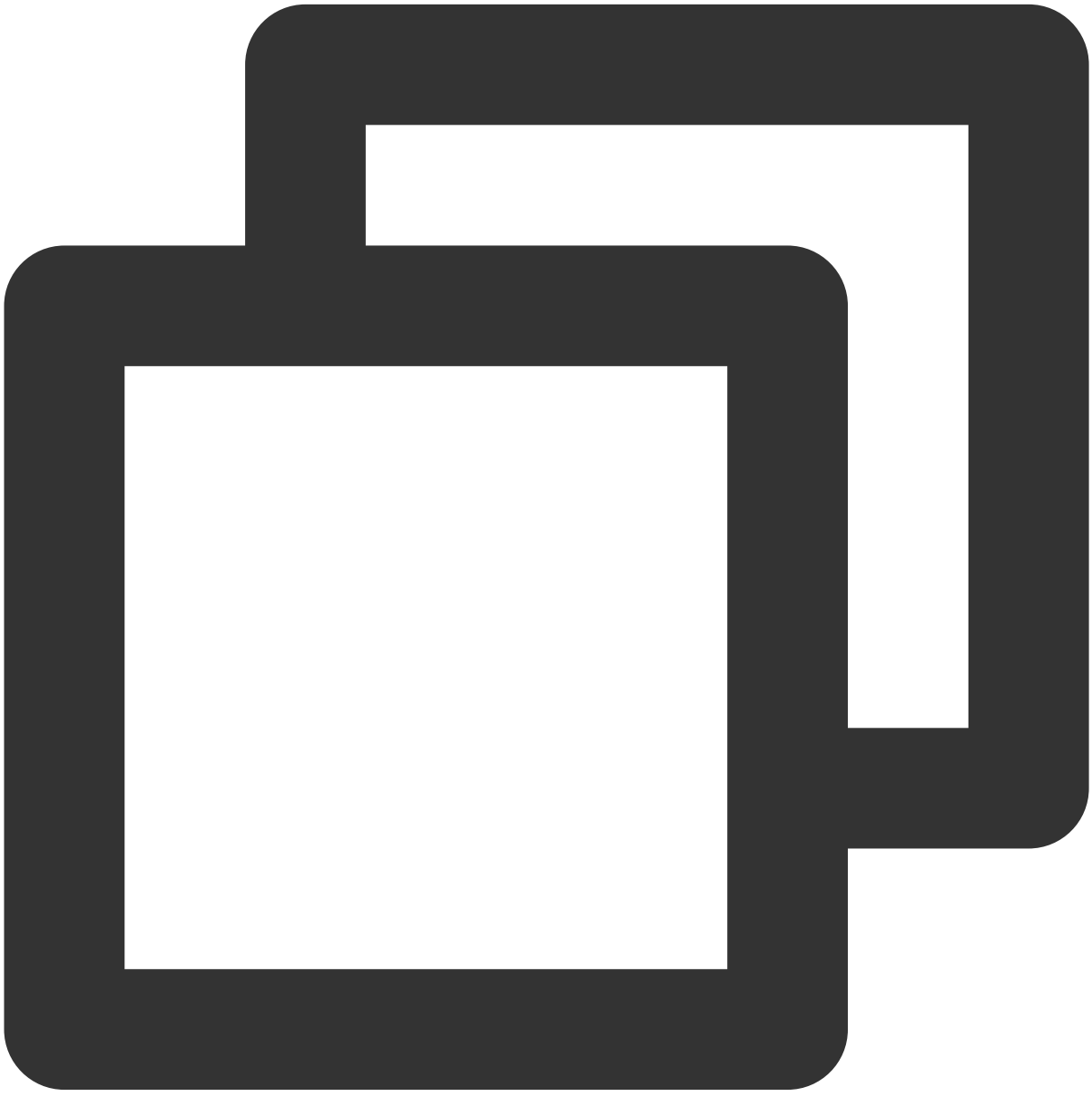
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

user_id	string	ユーザー ID。
callback	Callback	インターフェースコールバック。

ForbidSpeechApplication

キャスターが発言申請を禁止します。



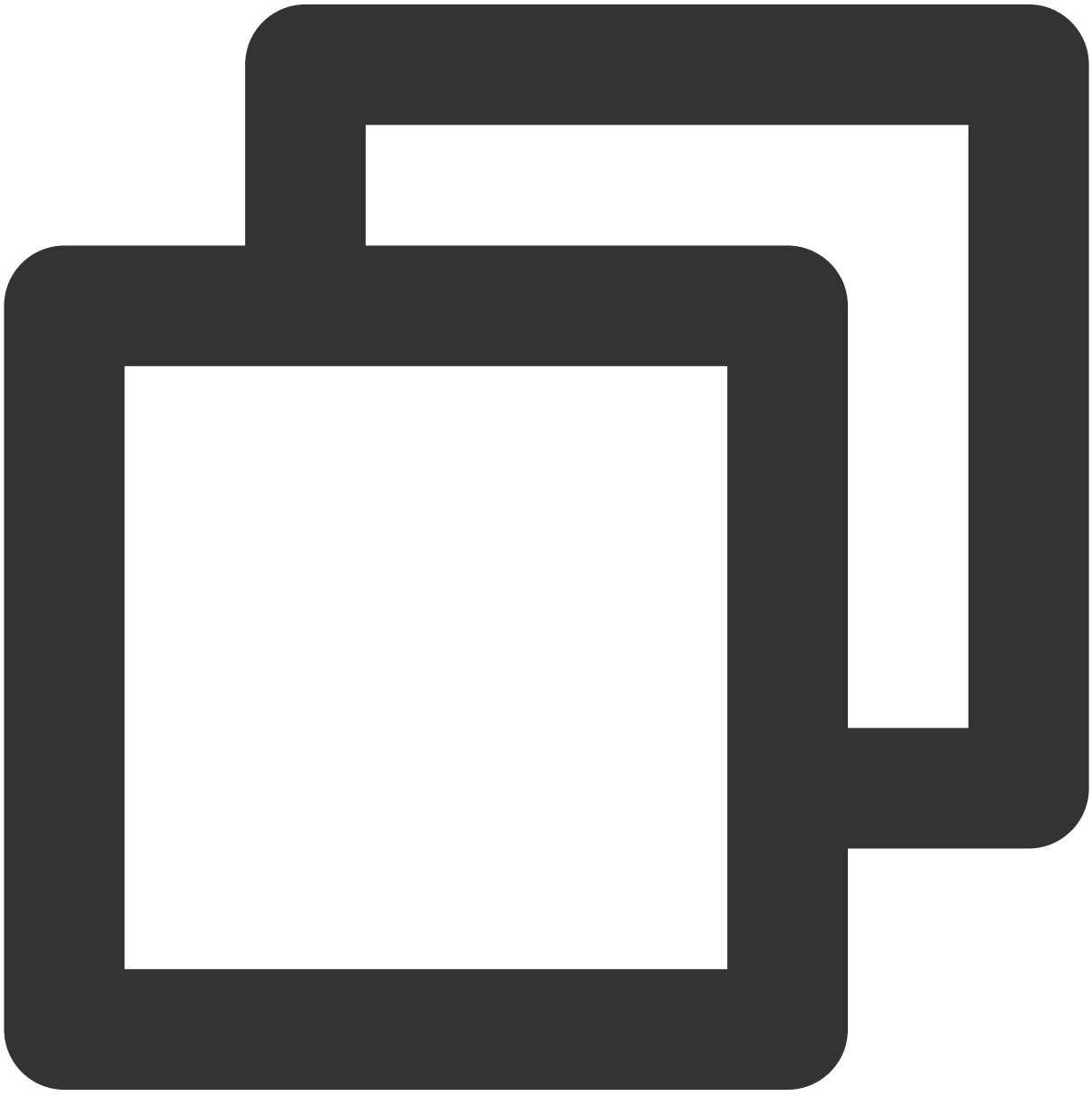
```
virtual int ForbidSpeechApplication(bool forbid) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
forbid	bool	禁止するかどうか。

SendOffSpeaker

キャスターが参加者に発言の停止を命令します。



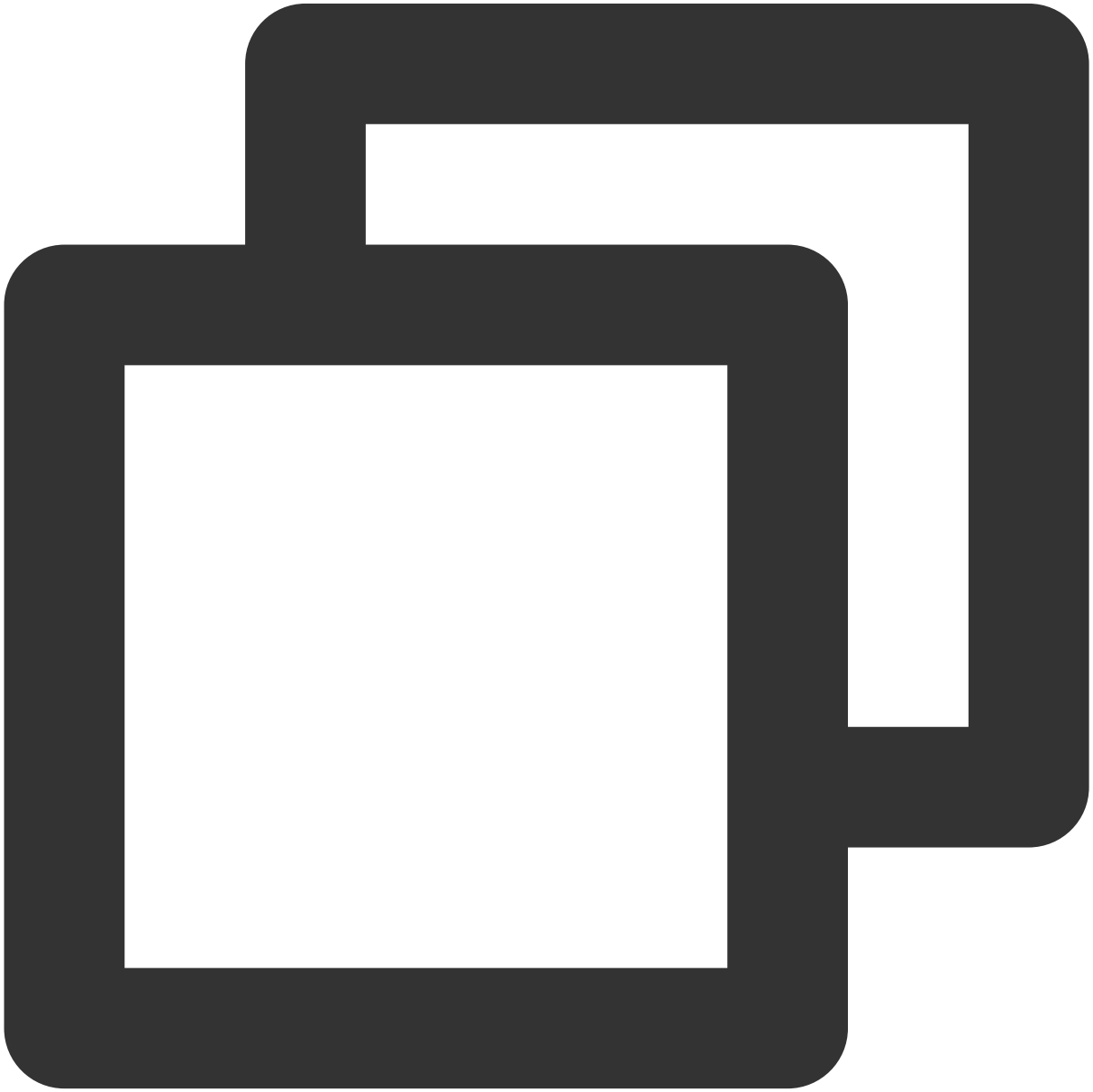
```
virtual int SendOffSpeaker(const std::string& user_id, Callback callback) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。
callback	Callback	インターフェースコールバック。

SendOffAllSpeakers

キャスターが全メンバーに発言の停止を命令します。



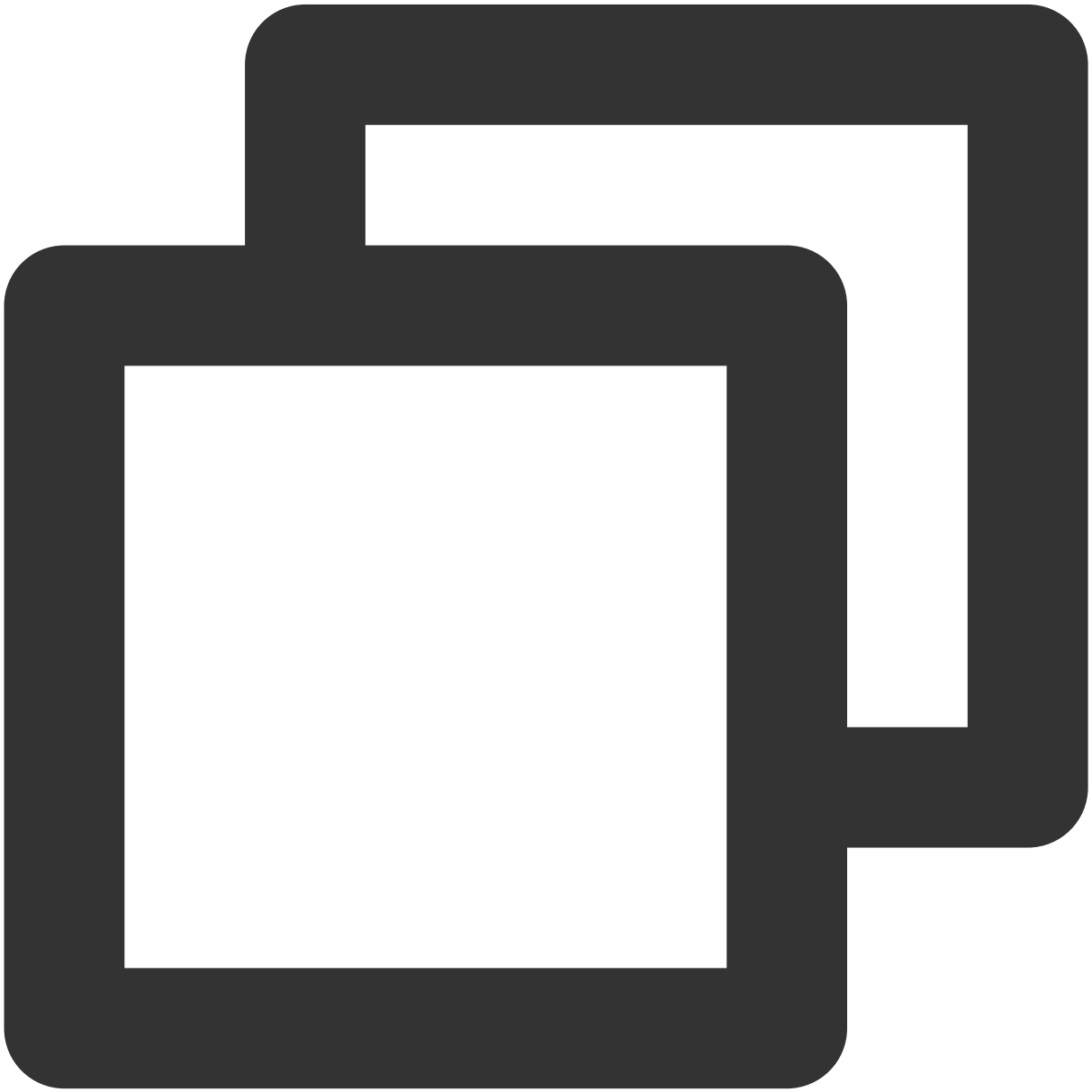
```
virtual int SendOffAllSpeakers(Callback callback) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callback	Callback	インターフェースコールバック。

ExitSpeechState

参加者が発言を停止し、視聴者になります。

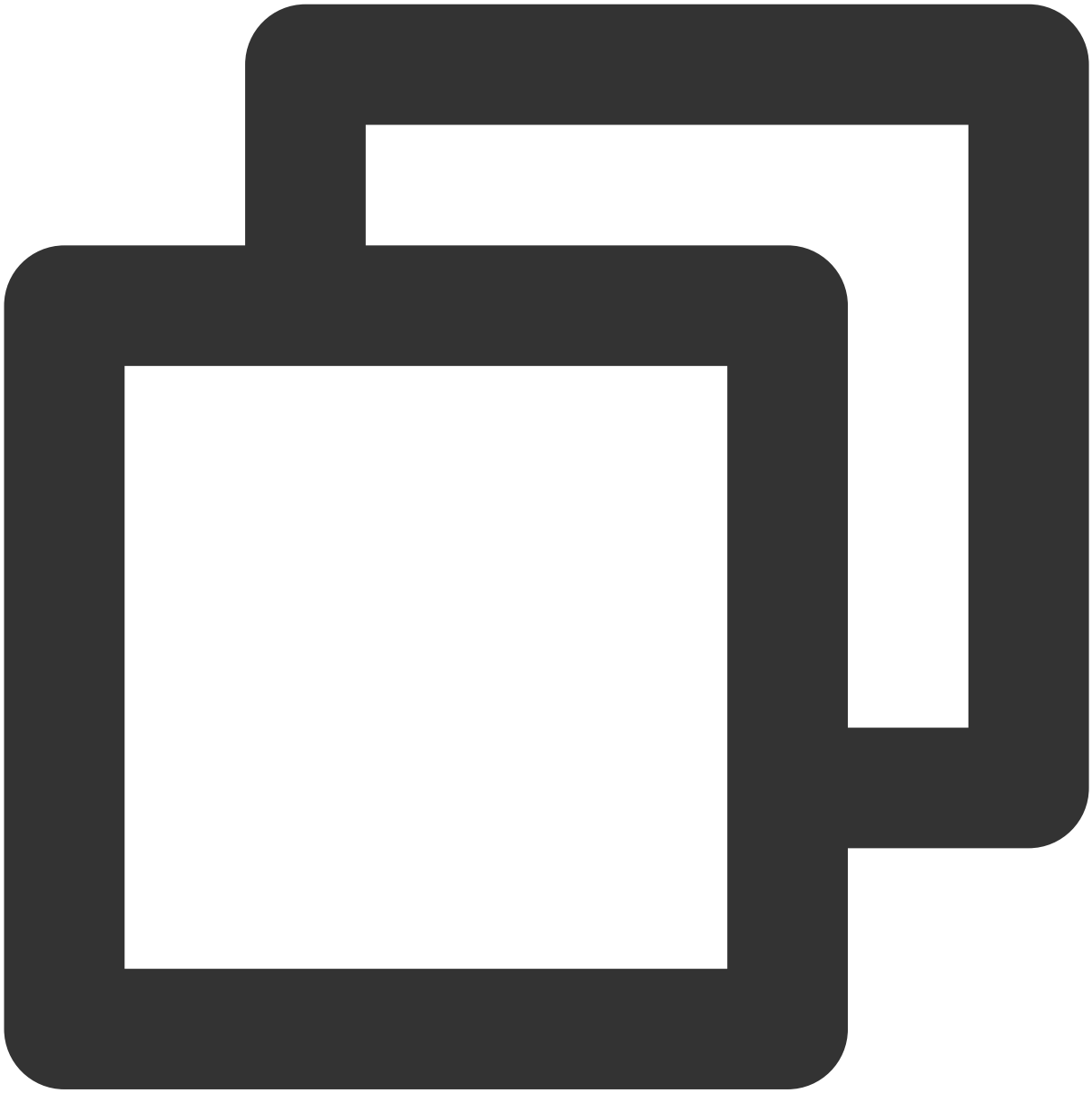


```
virtual int ExitSpeechState() = 0;
```

基本コンポーネントインターフェース

GetDeviceManager

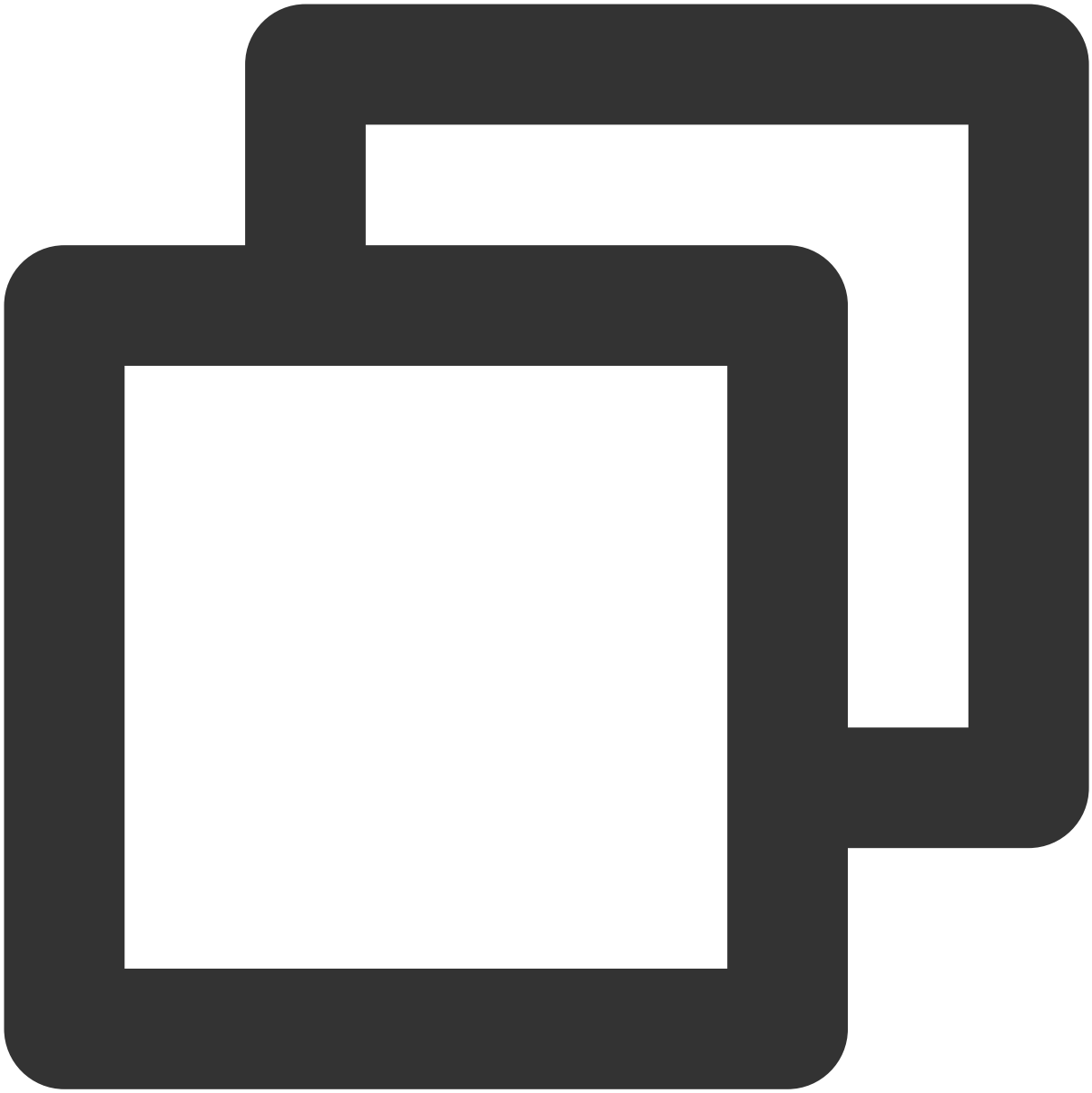
デバイス管理のオブジェクトポインタを取得します。



```
virtual liteav::ITXDeviceManager* GetDeviceManager() = 0;
```

GetScreenShareManager

画面共有管理のオブジェクトポインタを取得します。

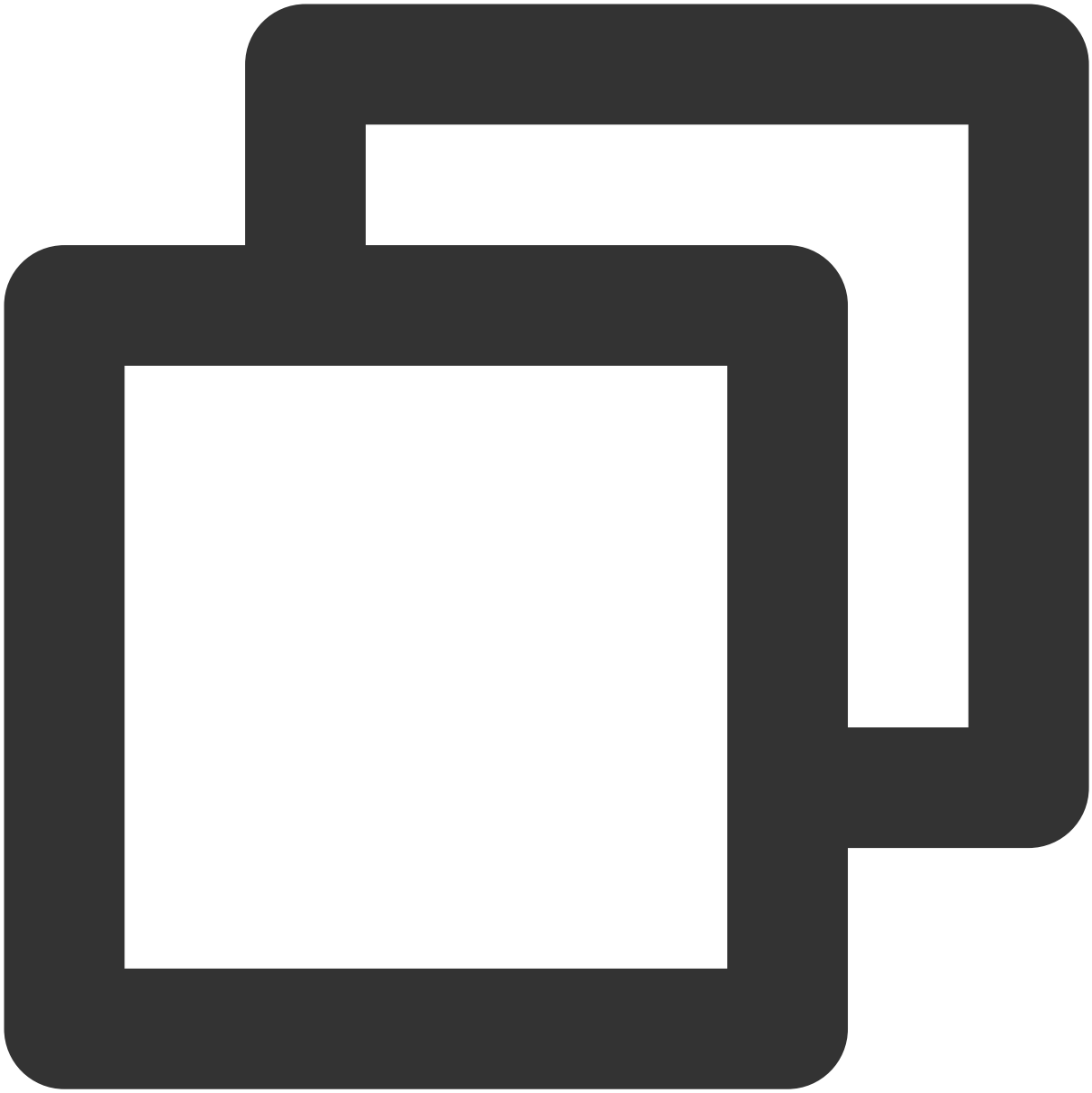


```
virtual IScreenShareManager* GetScreenShareManager() = 0;
```

クラウドレコーディングインターフェース

StartCloudRecord

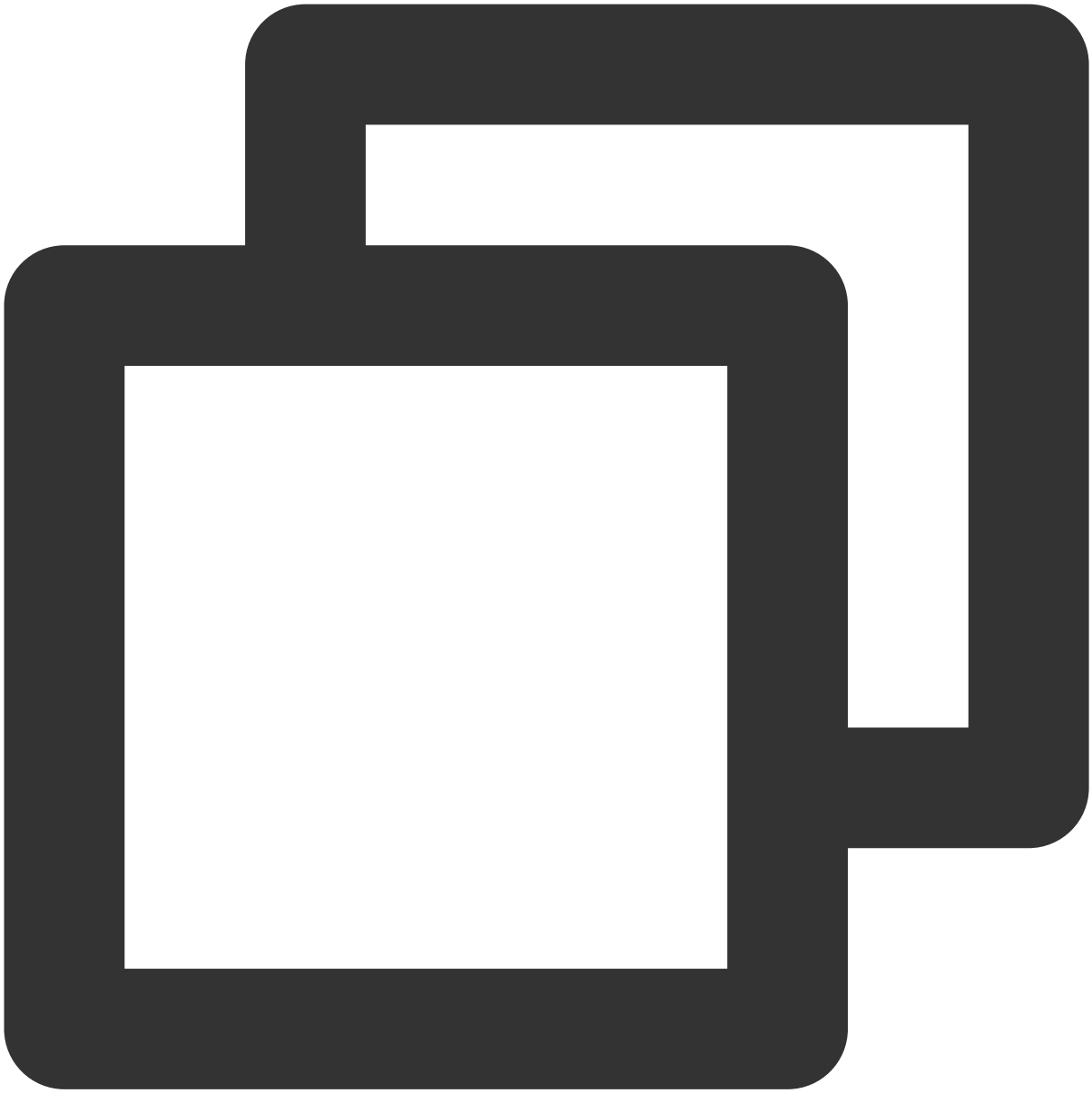
クラウドレコーディングを開始します。



```
virtual int StartCloudRecord() = 0;
```

StopCloudRecord

クラウドレコーディングを停止します。

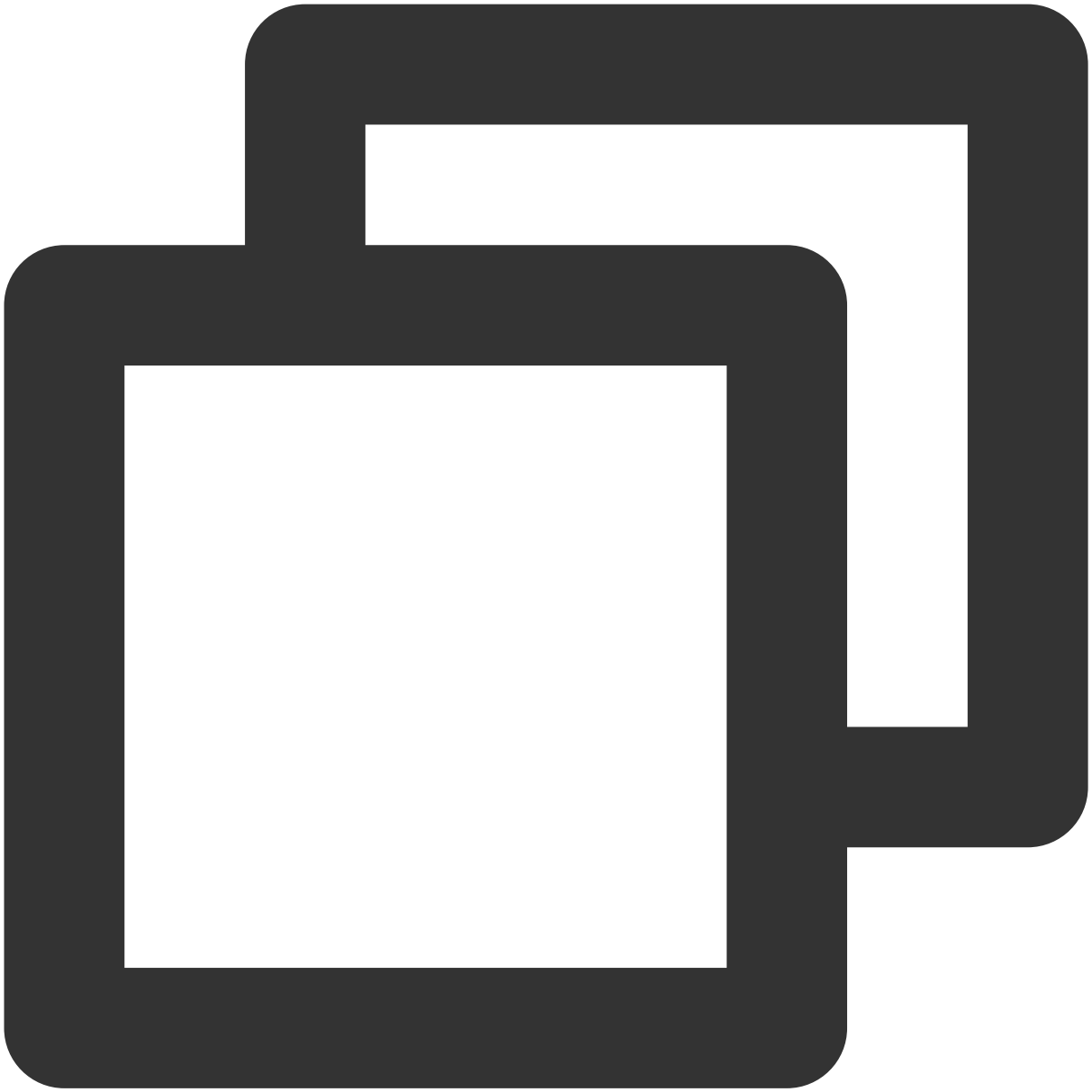


```
virtual int StopCloudRecord() = 0;
```

美顔関連インターフェース関数

SetBeautyStyle

美顔、美白、肌の色調補正効果のランクを設定します。



```
virtual int SetBeautyStyle(liteav::TRTCBeautyStyle style, uint32_t beauty_level,
        uint32_t whiteness_level, uint32_t ruddiness_level) = 0;
```

美顔管理では、次の機能を使用できます。
「美顔スタイル」を「スムース」または「ナチュラル」に設定します。「スムース」では、より強力な美肌補正効果が得られます。
「美顔レベル」を設定します。数値の範囲は0～9で、0はオフ、1～9までは数値が大きくなるほど効果が高くなります。

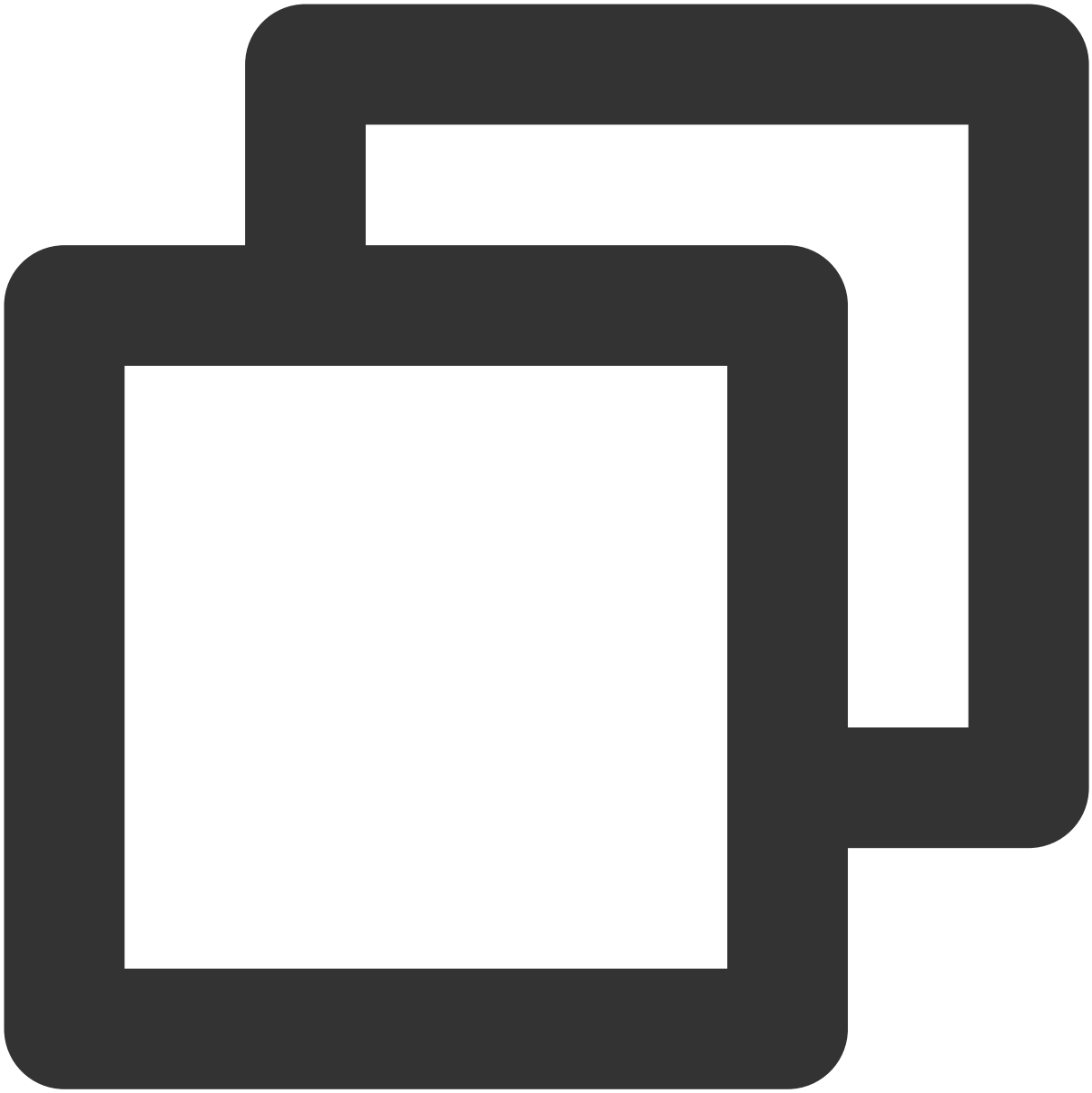
「美白レベル」を設定します。数値の範囲は0〜9で、0はオフ、1〜9までは数値が大きくなるほど効果が高くなります。
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
style	liteav::TRTCBeautyStyle	美顔スタイル。
beauty_level	uint32_t	美顔レベル。
whiteness_level	uint32_t	美白レベル。
ruddiness_level	uint32_t	肌色補正レベル。

関連設定インターフェース

SetVideoQosPreference

ネットワークトラフィックコントロール関連パラメータを設定します。



```
virtual int SetVideoQosPreference(TUIVideoQosPreference preference) = 0;
```

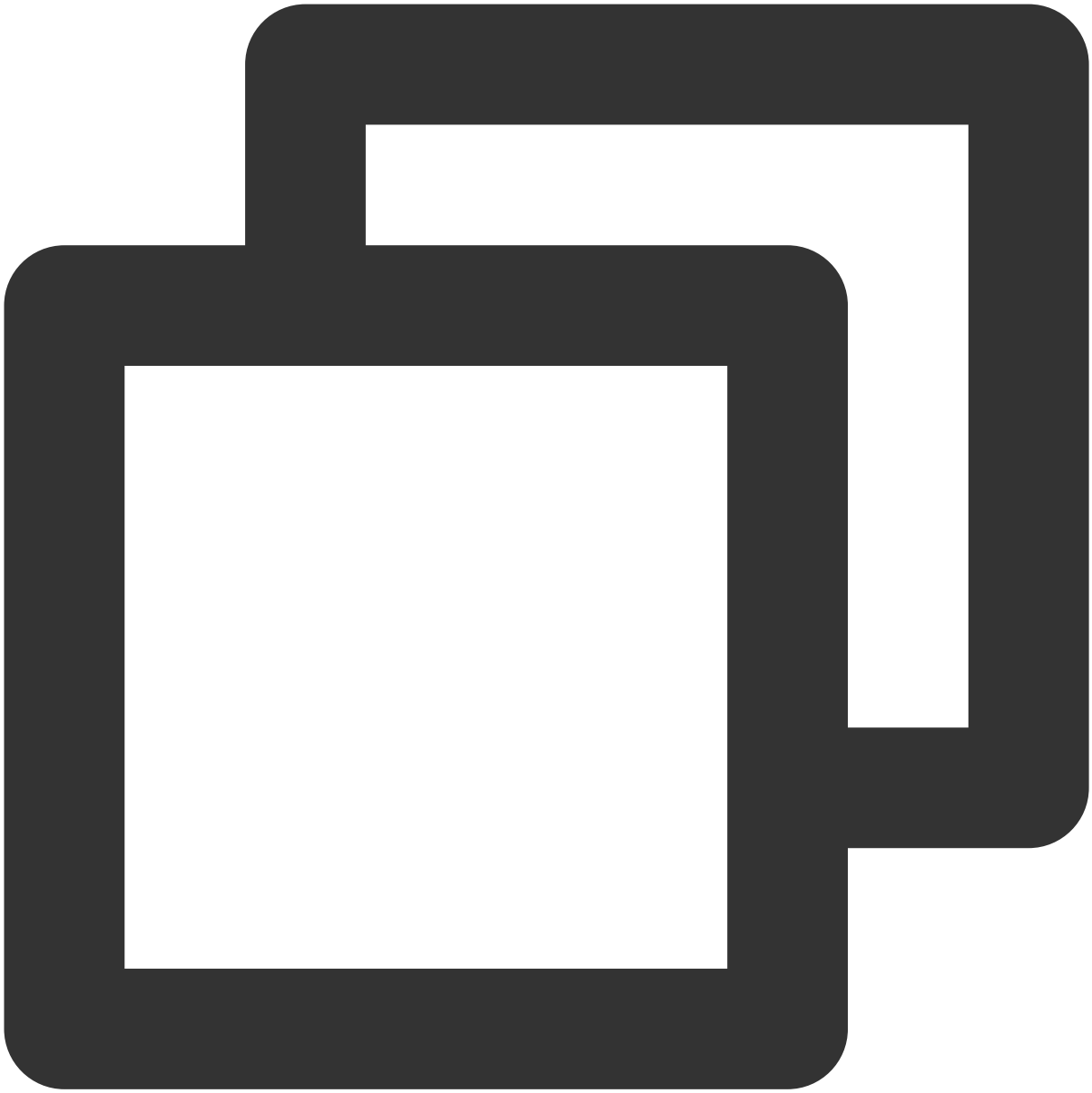
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
preference	TUIVideoQosPreference	ネットワークトラフィックコントロールポリシー。

SDKバージョンインターフェースの取得

GetSDKVersion

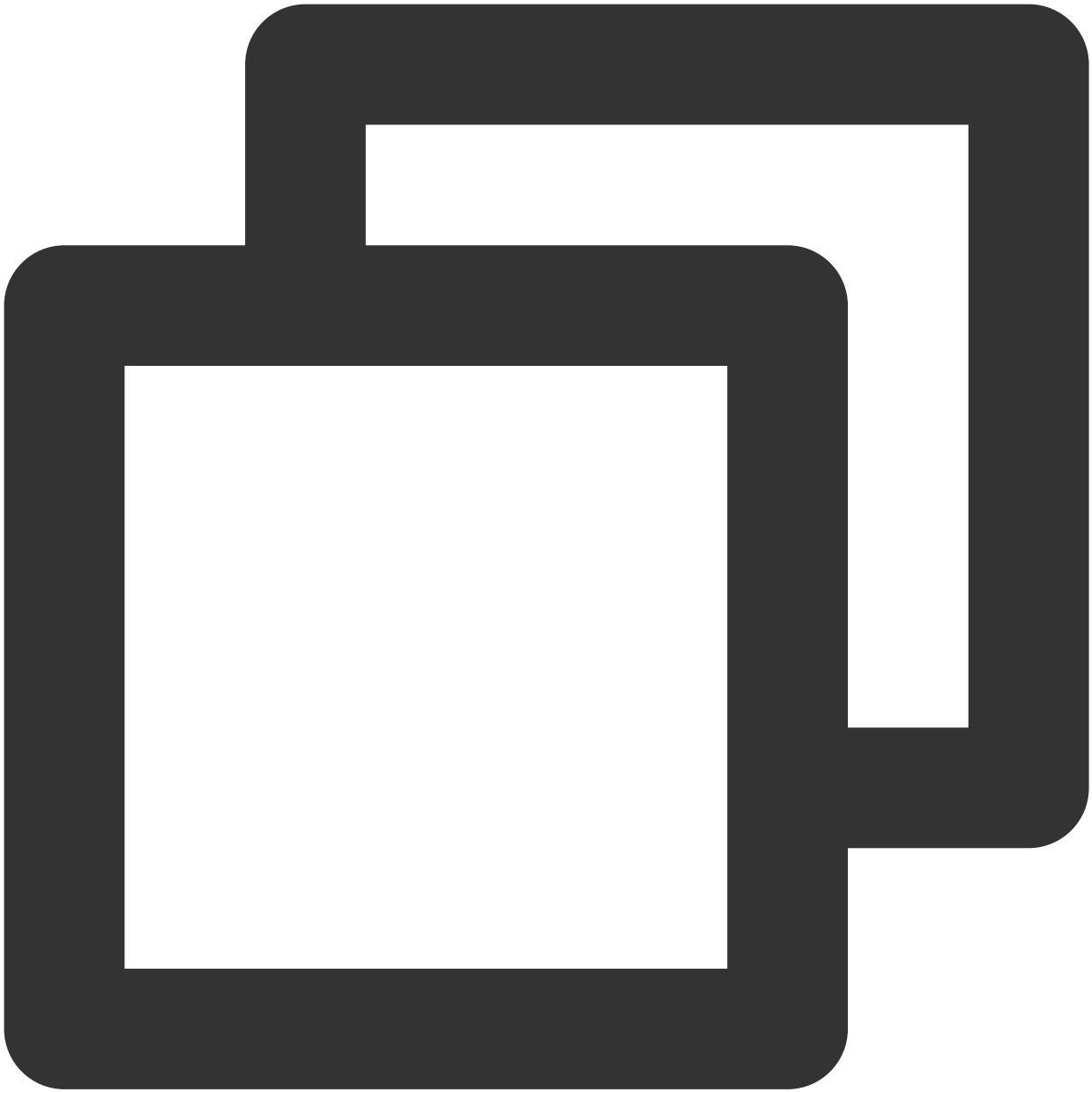
SDKバージョン情報を取得します。



```
virtual const char* GetSDKVersion() = 0;
```

エラーイベントコールバック

OnError



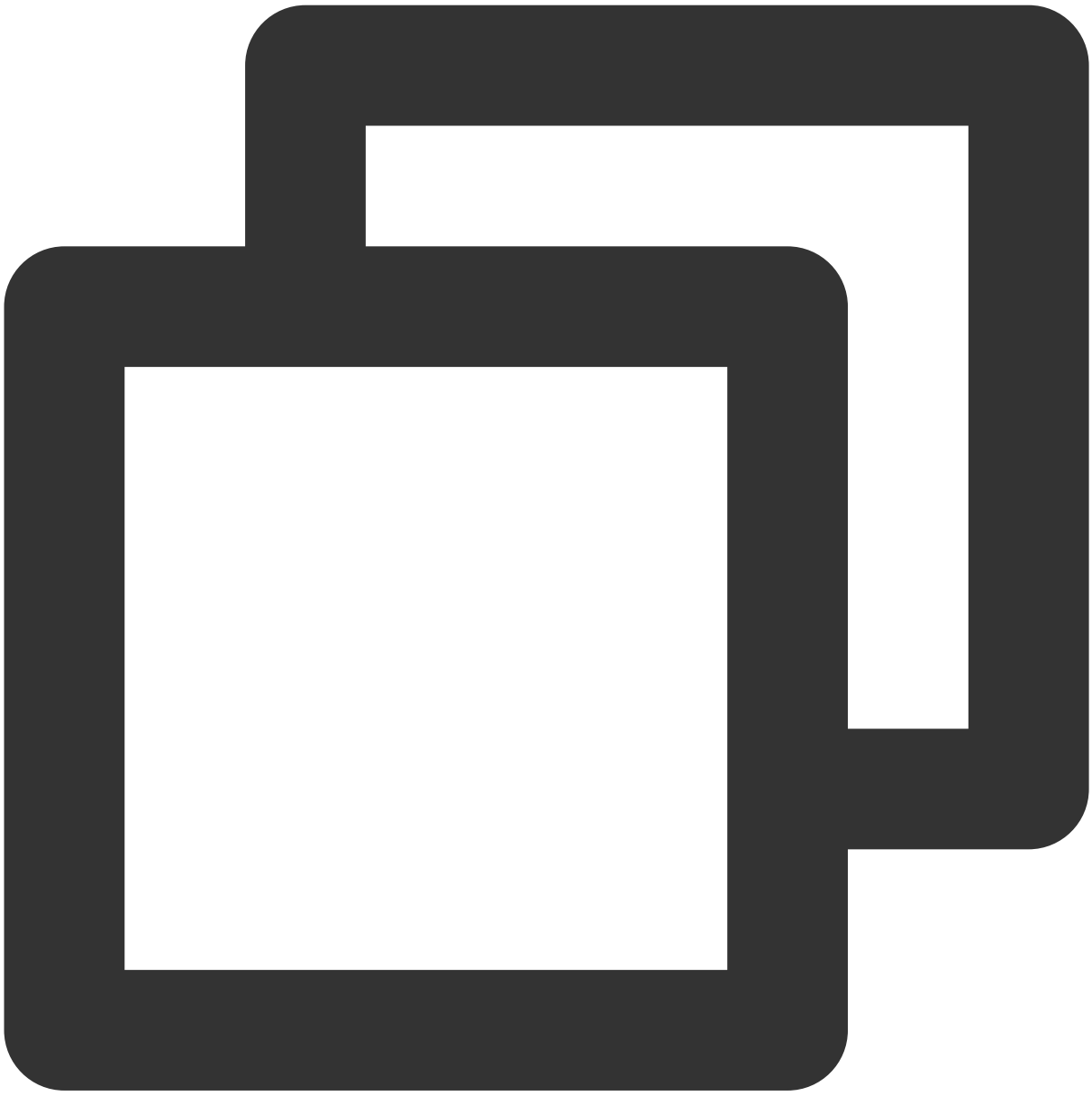
```
void OnError(int code, const std::string& message);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
code	int	エラーコード。
message	string	エラー情報。

基本イベントコールバック

OnLogin



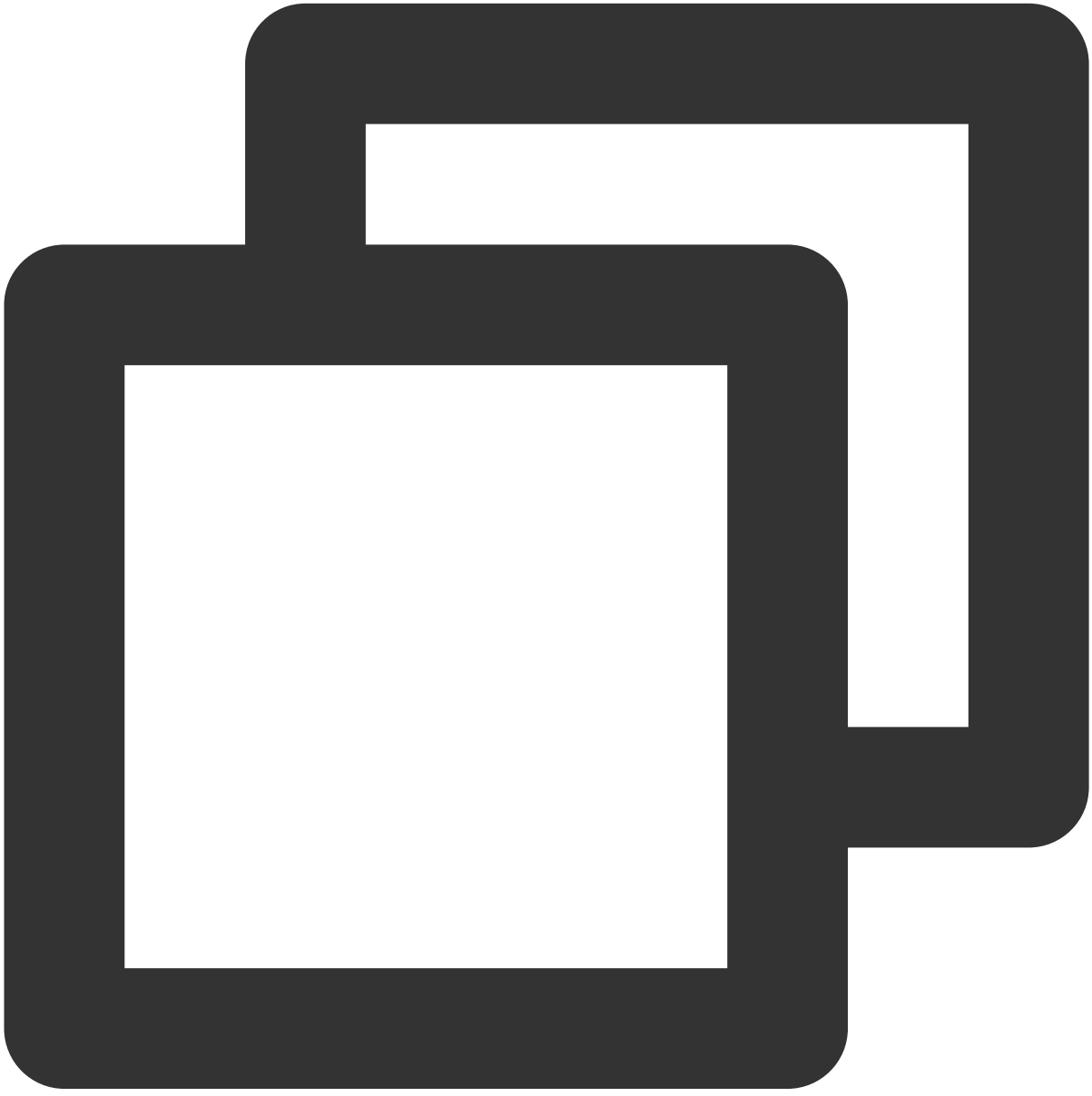
```
virtual void OnLogin(int code, const std::string& message) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
code	int	エラーコード。

message	string	ログイン情報またはログイン失敗のエラー情報。

OnLogout



```
virtual void OnLogout(int code, const std::string& message) = 0;
```

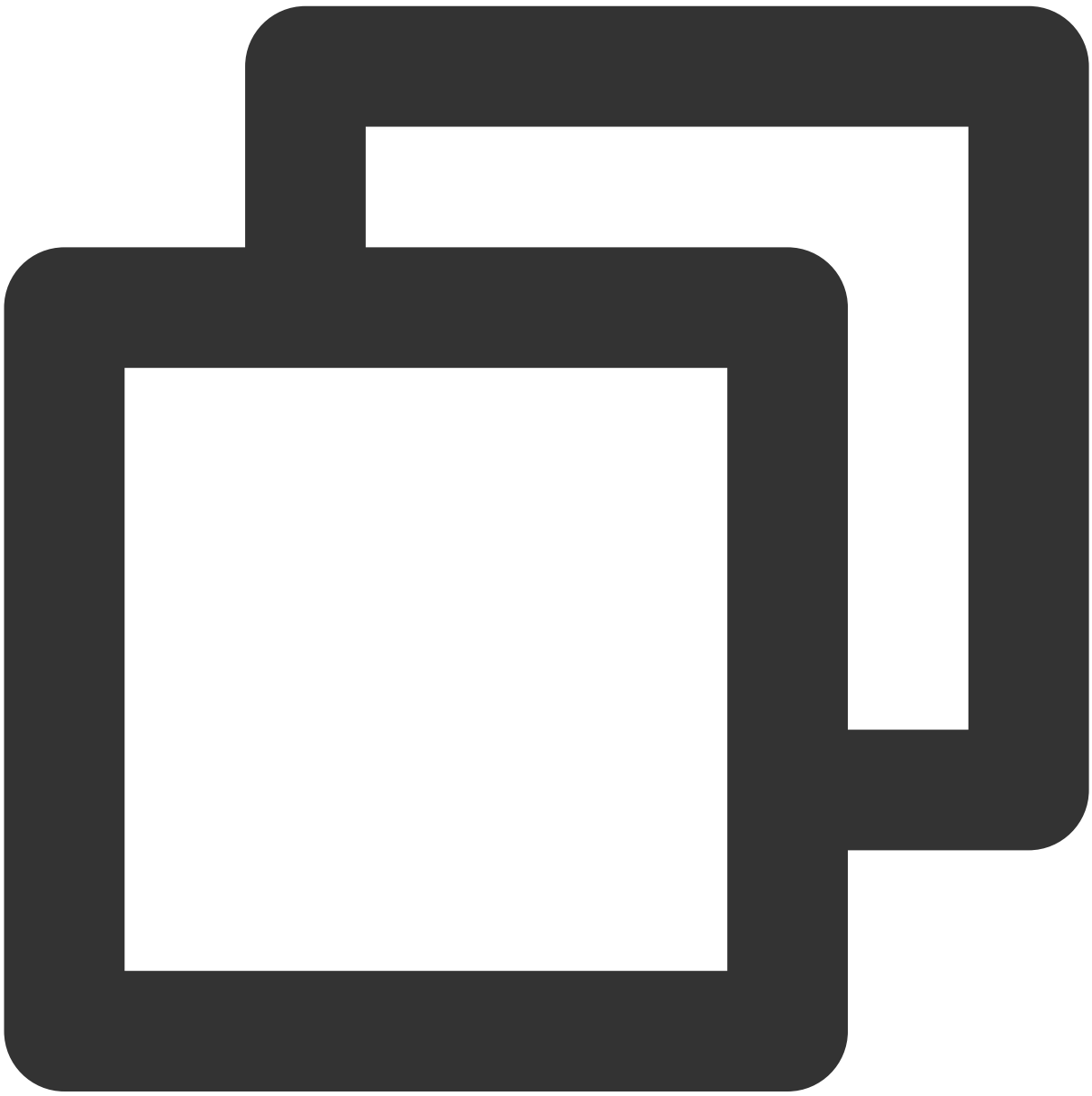
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

code	int	エラーコード。
message	string	エラー情報。

OnCreateRoom

ルーム作成のコールバックです。



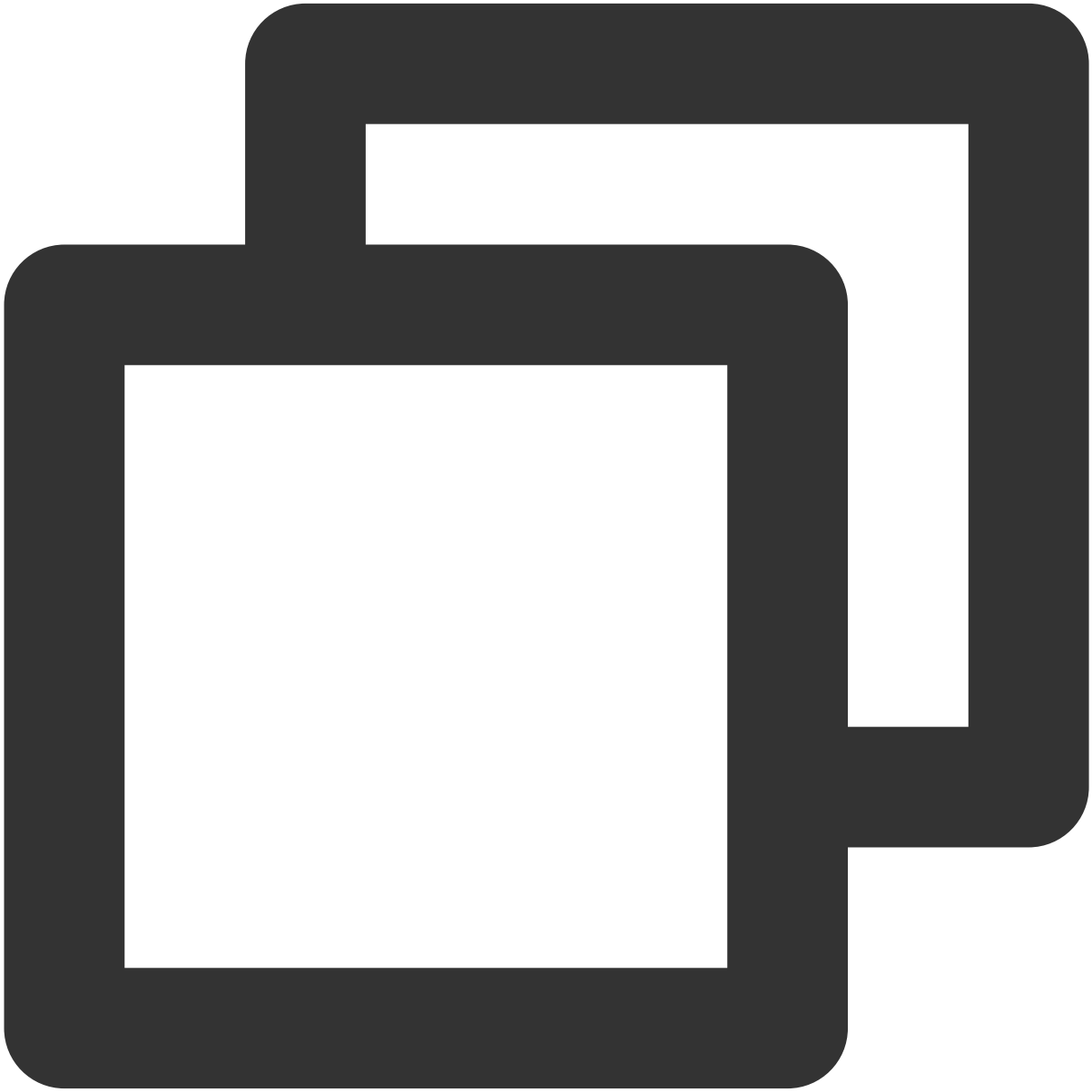
```
virtual void OnCreateRoom(int code, const std::string& message) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
code	int	エラーコード。
message	string	エラー情報。

OnDestroyRoom

ルーム解散のコールバックです。



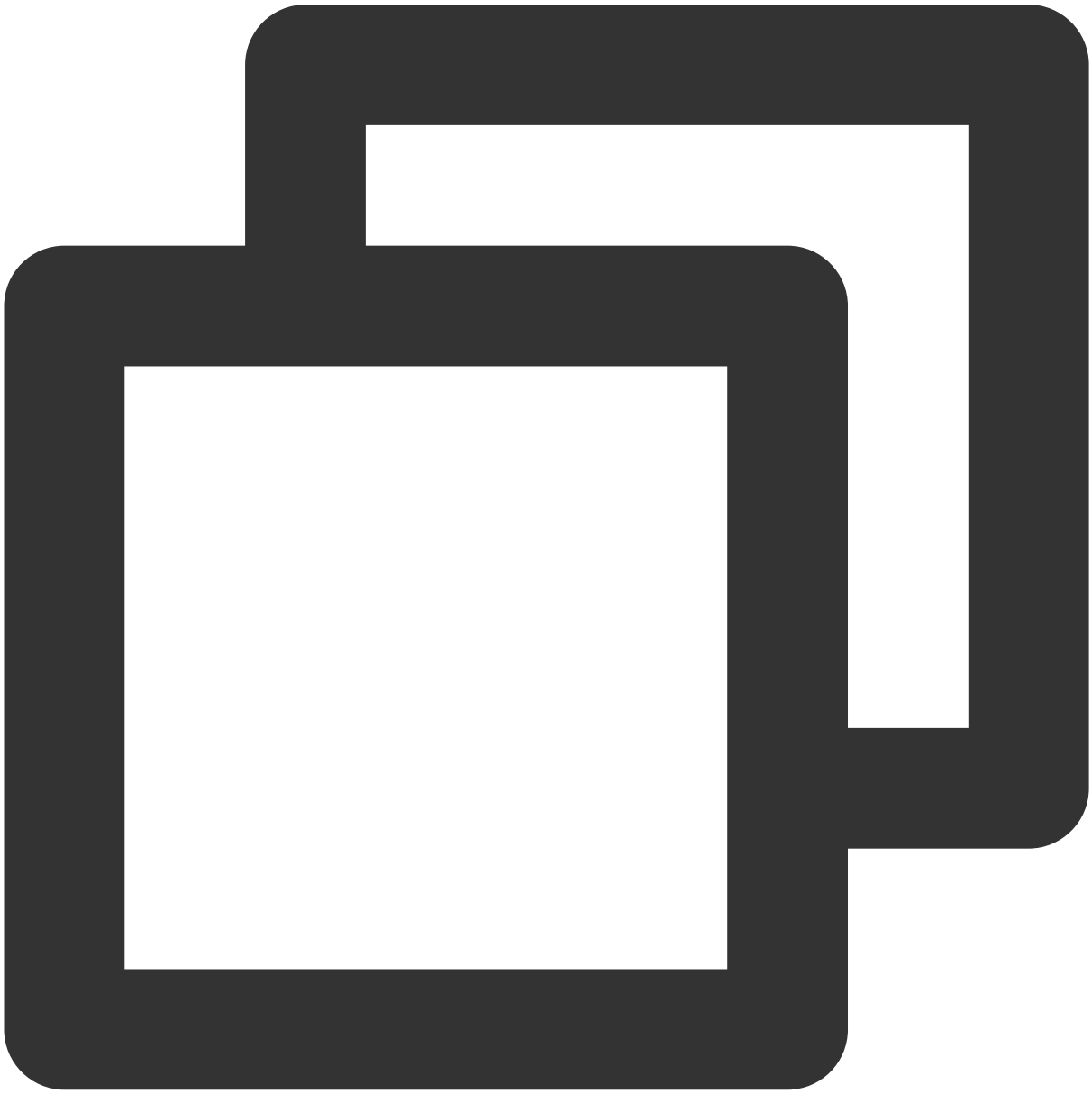
```
virtual void OnDestroyRoom(int code, const std::string& message) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
code	int	エラーコード。
message	string	エラー情報。

OnEnterRoom

入室コールバックです。



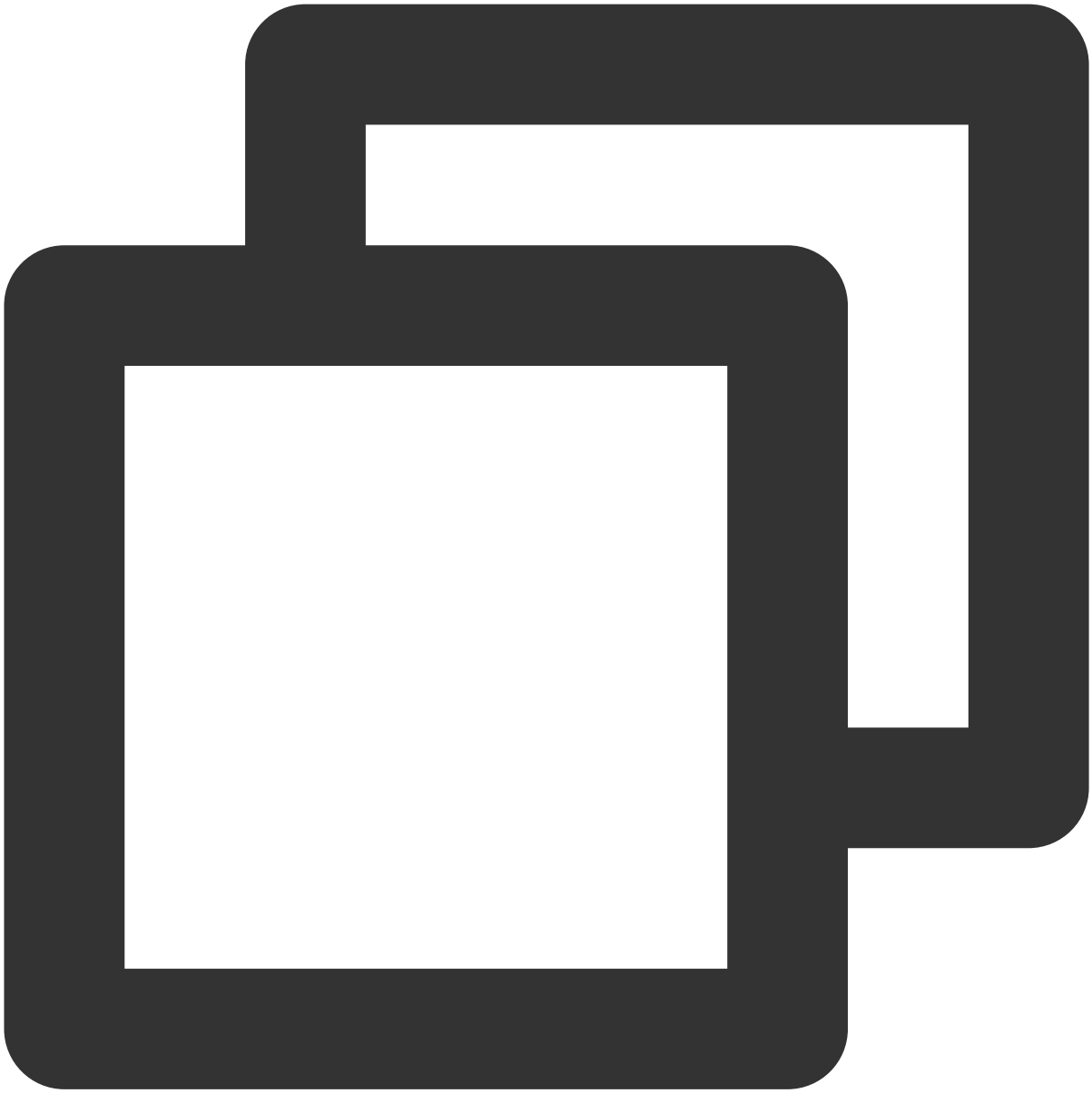
```
virtual void OnEnterRoom(int code, const std::string& message) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
code	int	エラーコード。
message	string	エラー情報。

OnExitRoom

退室コールバックです。



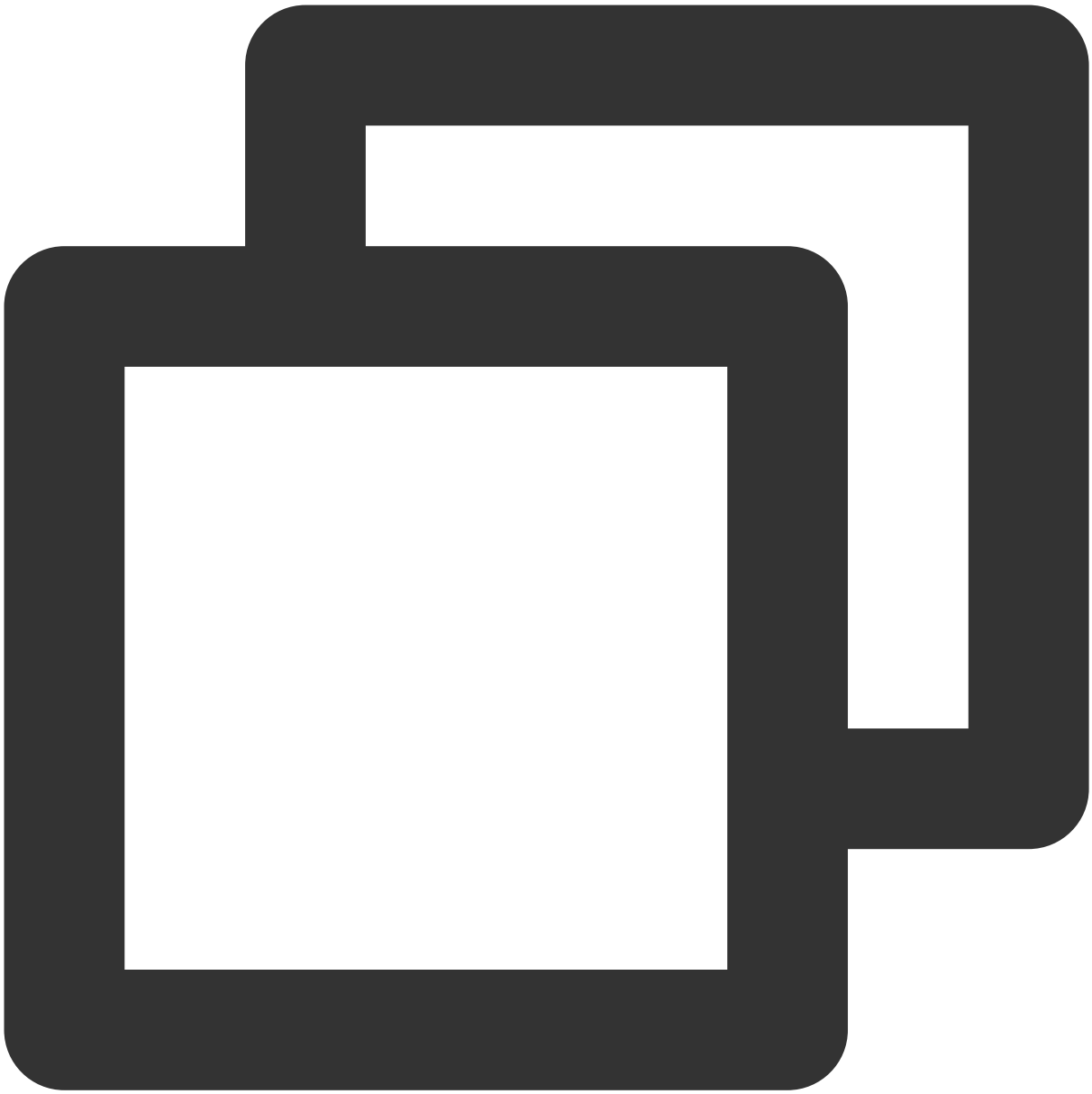
```
virtual void OnExitRoom(TUIExitRoomType type, const std::string& message) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
type	TUIExitRoomType	退室のタイプ。
message	string	エラー情報。

OnFirstVideoFrame

自身のローカルまたはリモートユーザーの最初のフレーム画面のレンダリングを開始します。



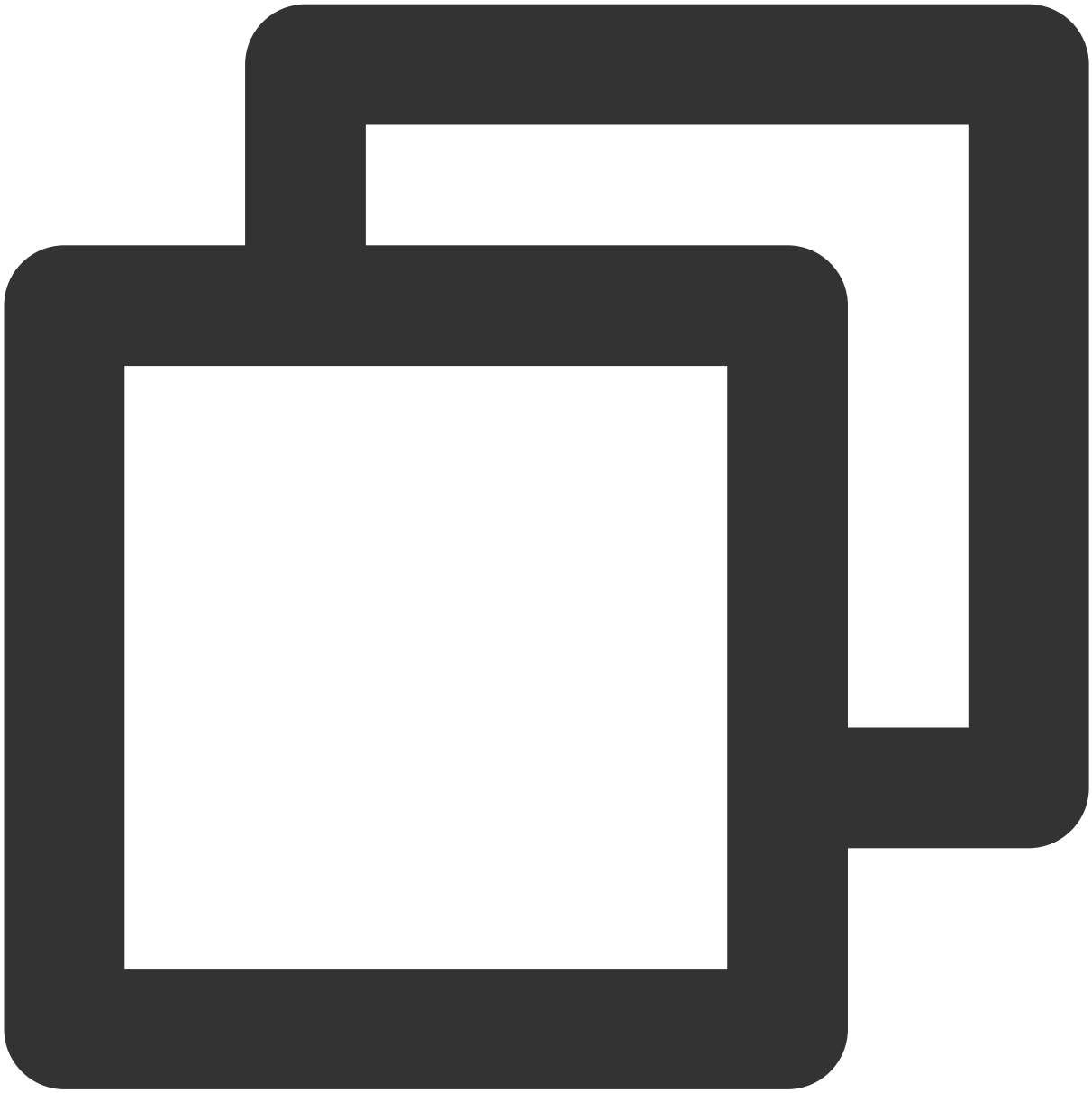
```
virtual void OnFirstVideoFrame(const std::string& user_id, const TUIStreamType stre
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。
stream_type	TUIStreamType	ストリームのタイプ。

OnUserVoiceVolume

ユーザー音量の大きさのコールバック。



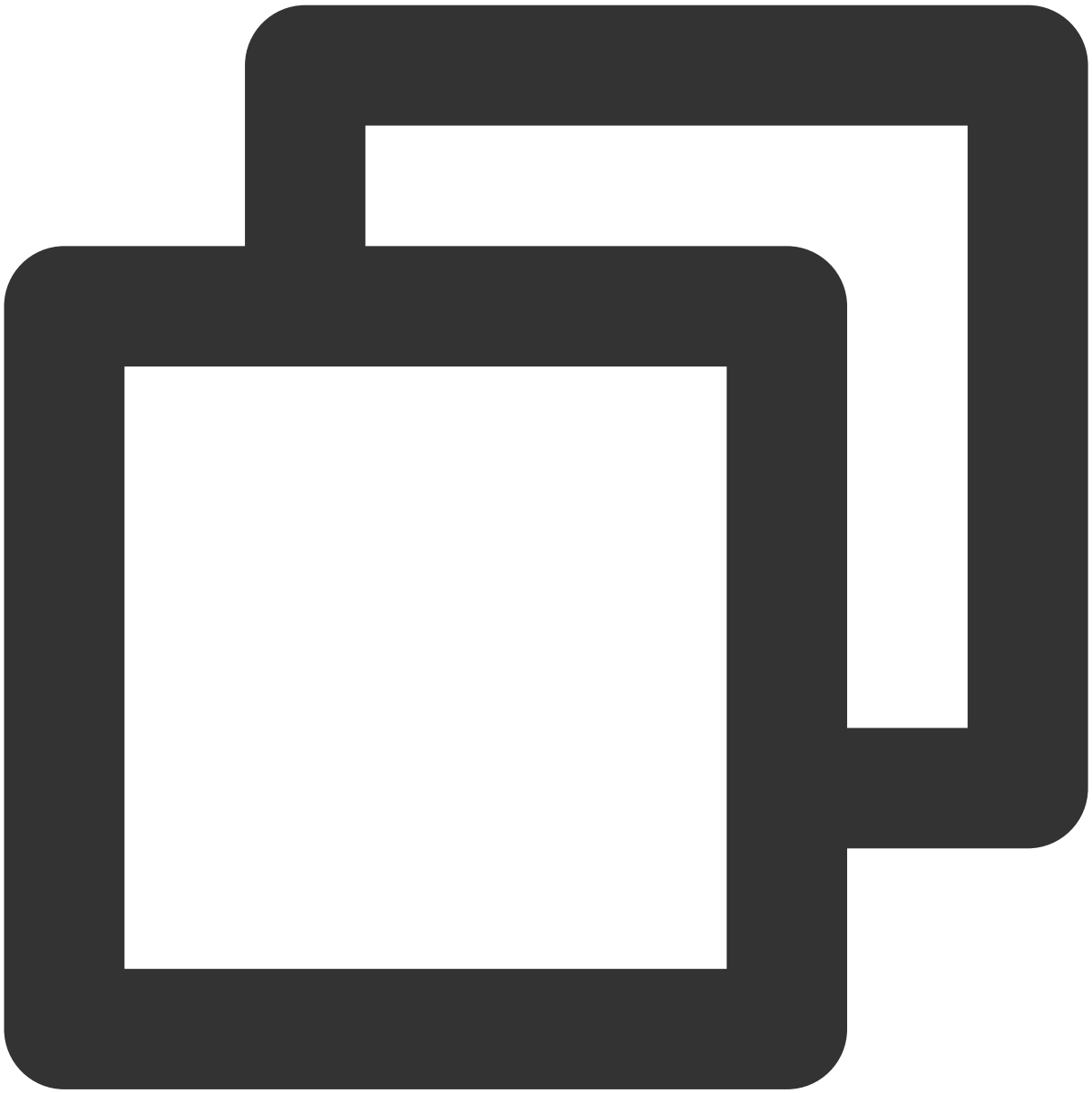
```
virtual void OnUserVoiceVolume(const std::string& user_id, int volume)
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。
volume	int	ユーザーの音量の大きさ、値の範囲0～100。

OnRoomMasterChanged

キャスター変更のコールバック。



```
virtual void OnRoomMasterChanged(const std::string& user_id) = 0;
```

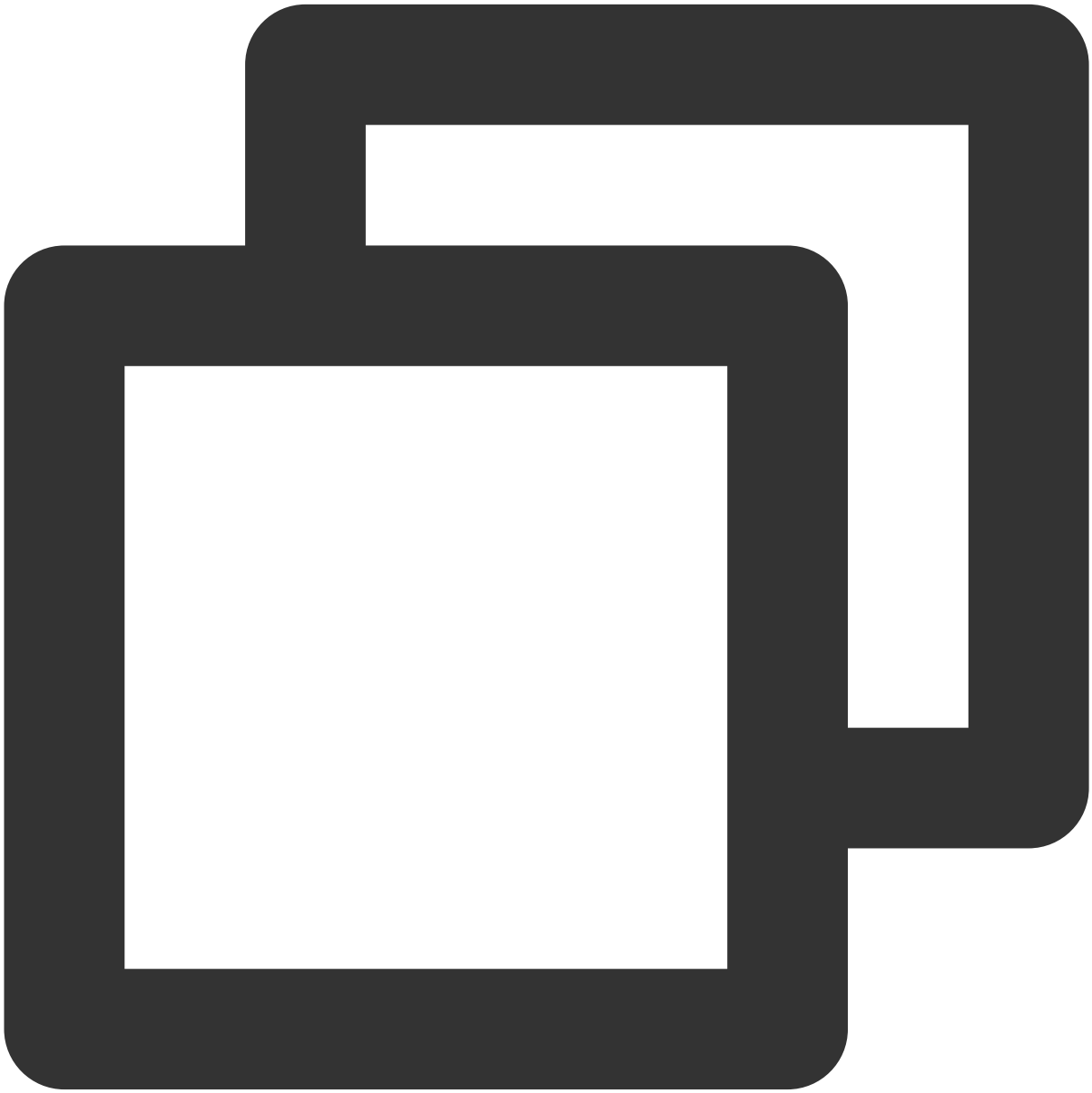
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。

リモートユーザーコールバックイベント

OnRemoteUserEnter

リモートユーザー入室コールバック。



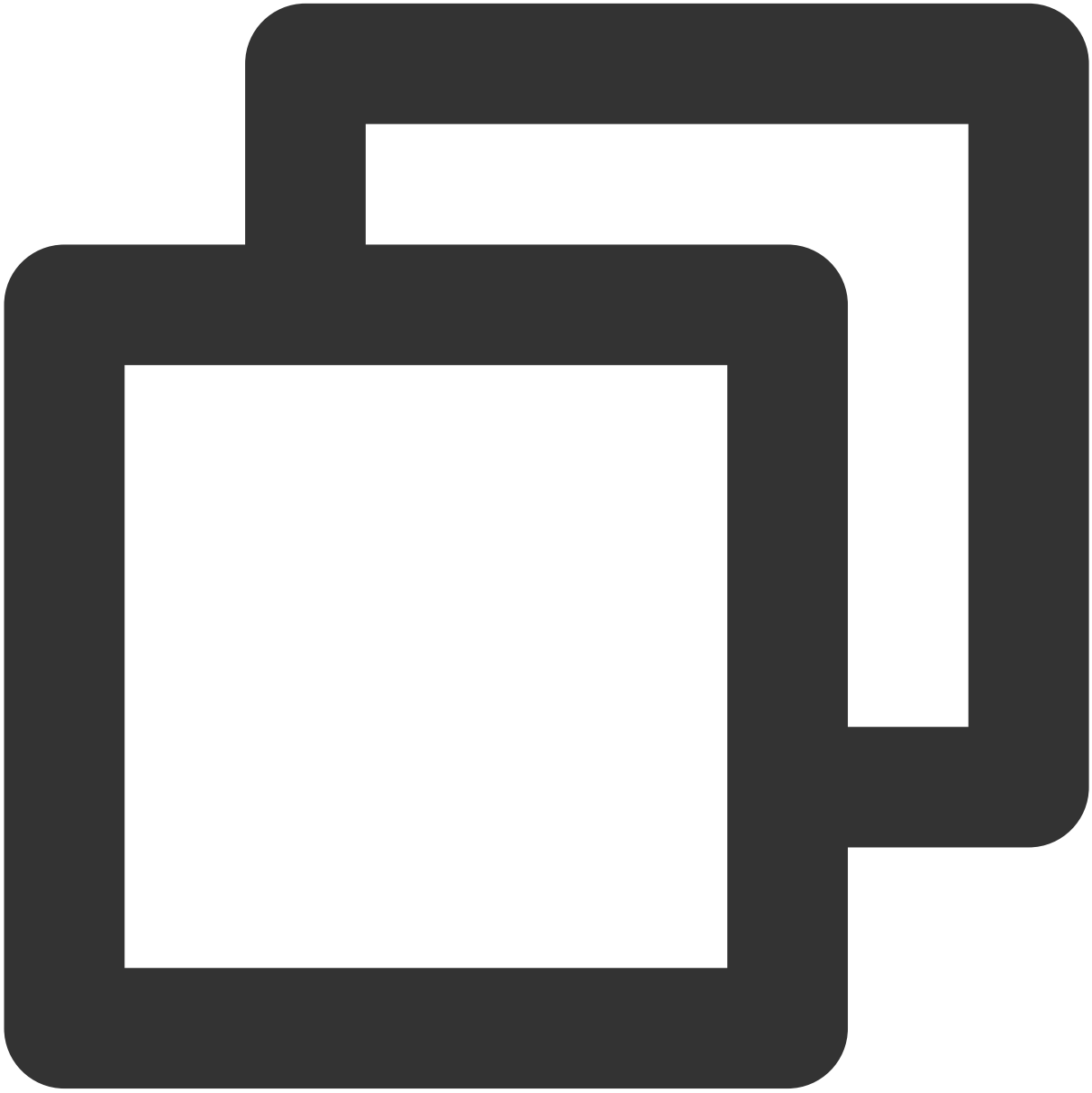
```
virtual void OnRemoteUserEnter(const std::string& user_id) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。

OnRemoteUserLeave

リモートユーザー退室コールバック。



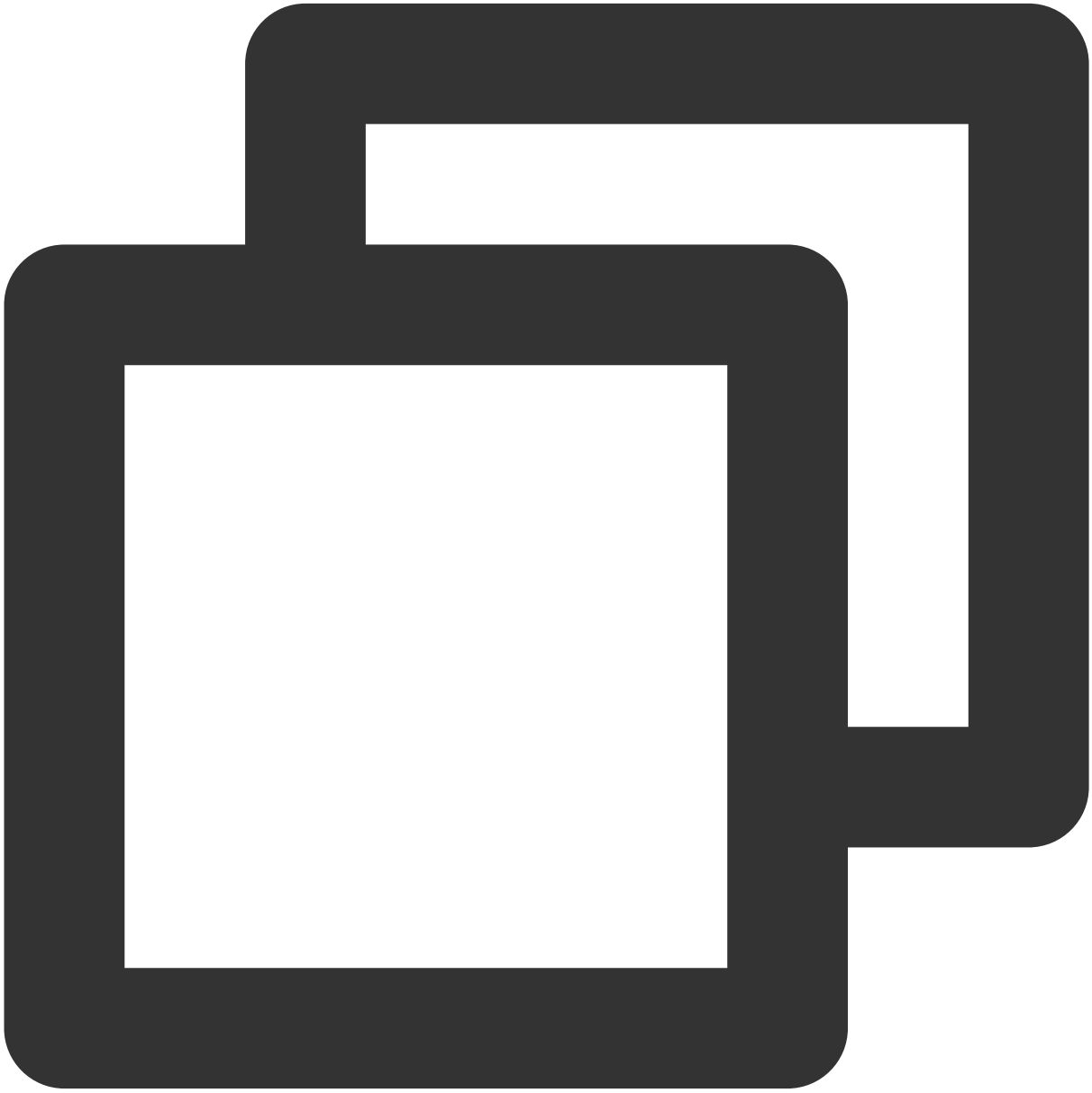
```
virtual void OnRemoteUserLeave(const std::string& user_id) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。

OnRemoteUserCameraAvailable

リモートユーザーが、カメラ、ビデオを起動しているかどうか。



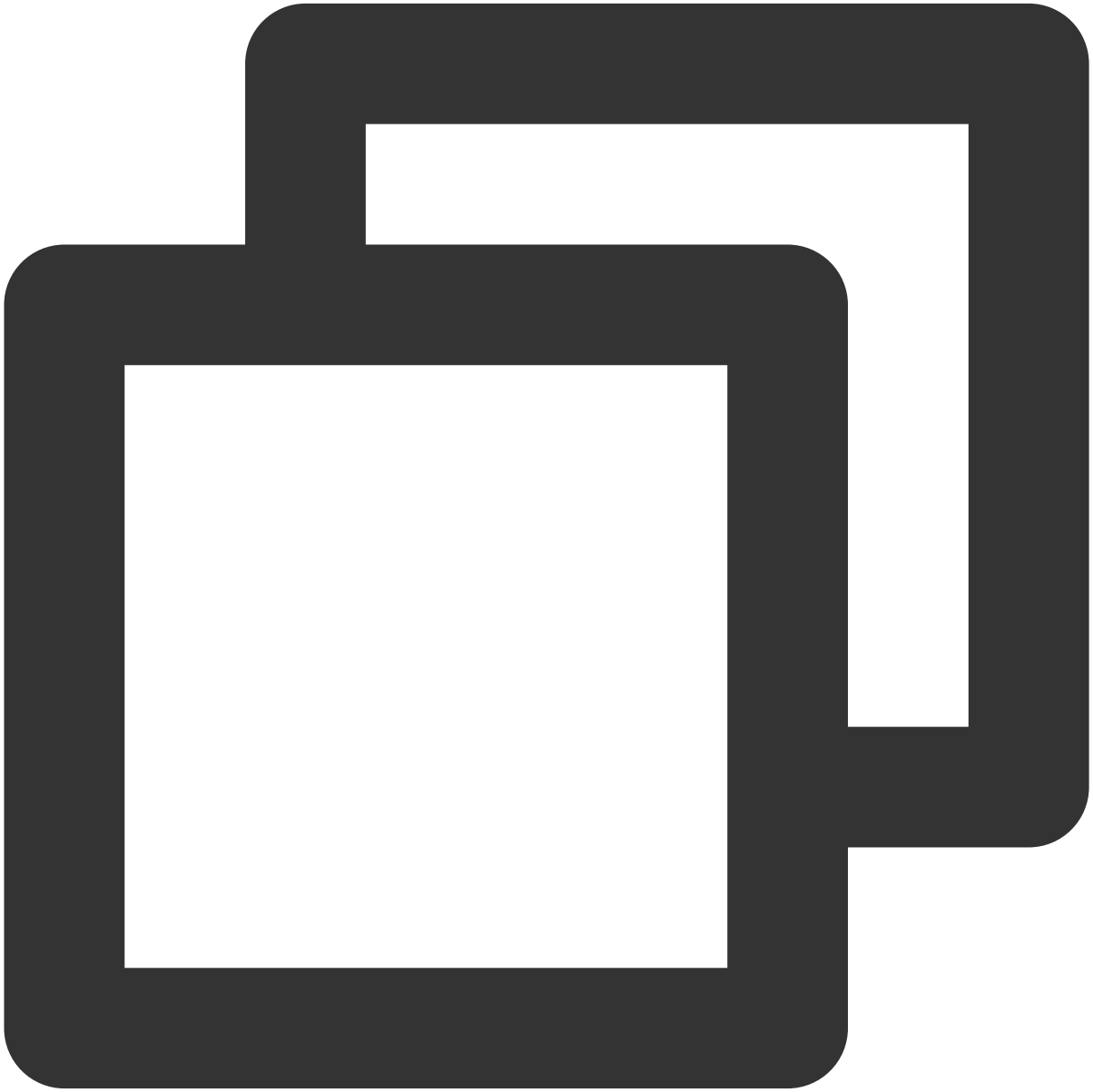
```
virtual void OnRemoteUserCameraAvailable(const std::string& user_id, bool available
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。
available	bool	true：ビデオストリームデータあり；false：ビデオストリームデータなし。

OnRemoteUserScreenAvailable

リモートユーザーが画面共有を開始しているかどうか。



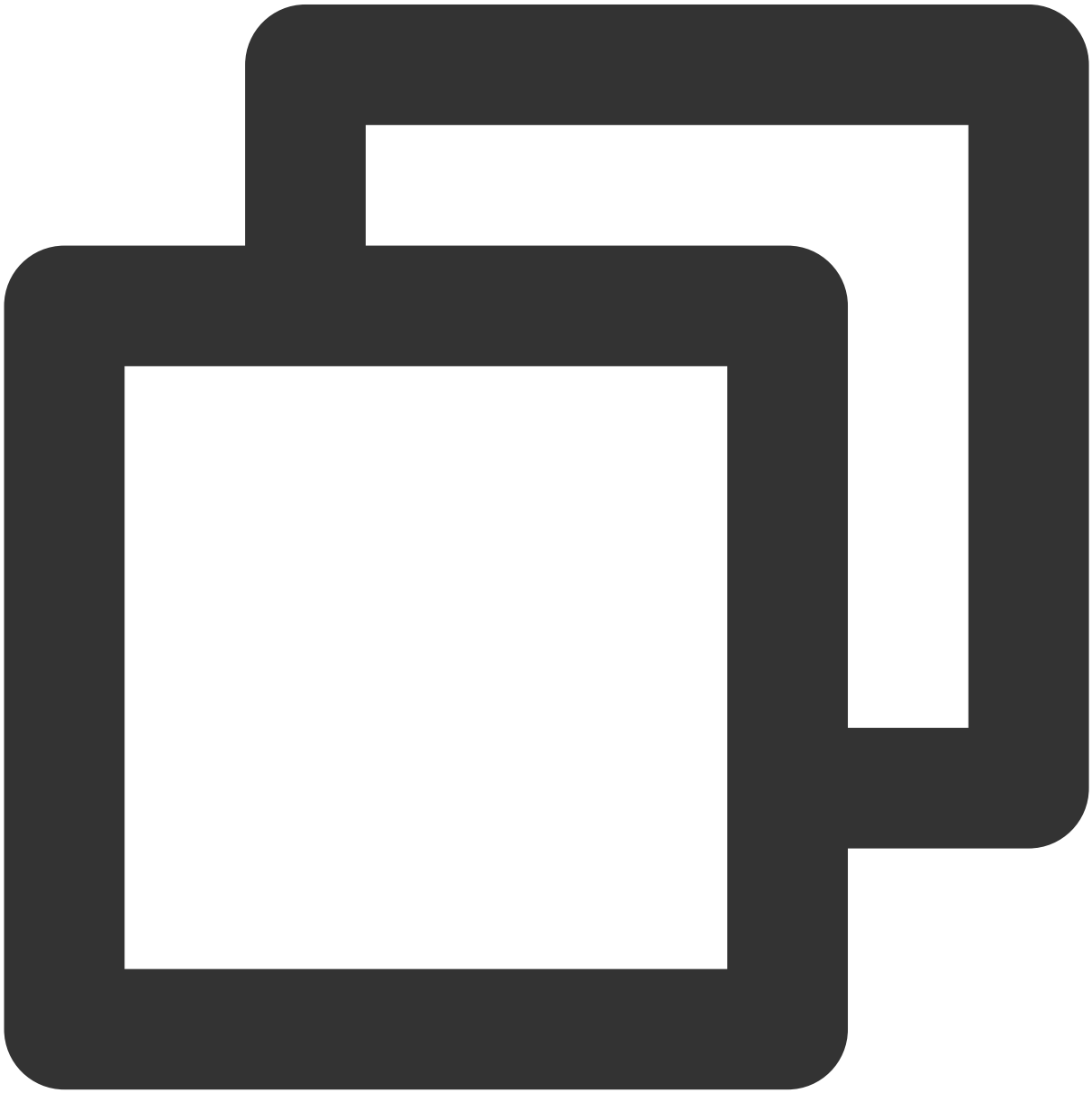
```
virtual void OnRemoteUserScreenAvailable(const std::string& user_id, bool available
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。
available	bool	true：ビデオストリームデータあり；false：ビデオストリームデータなし。

OnRemoteUserAudioAvailable

リモートユーザーがマイクをオンにしているかどうか。



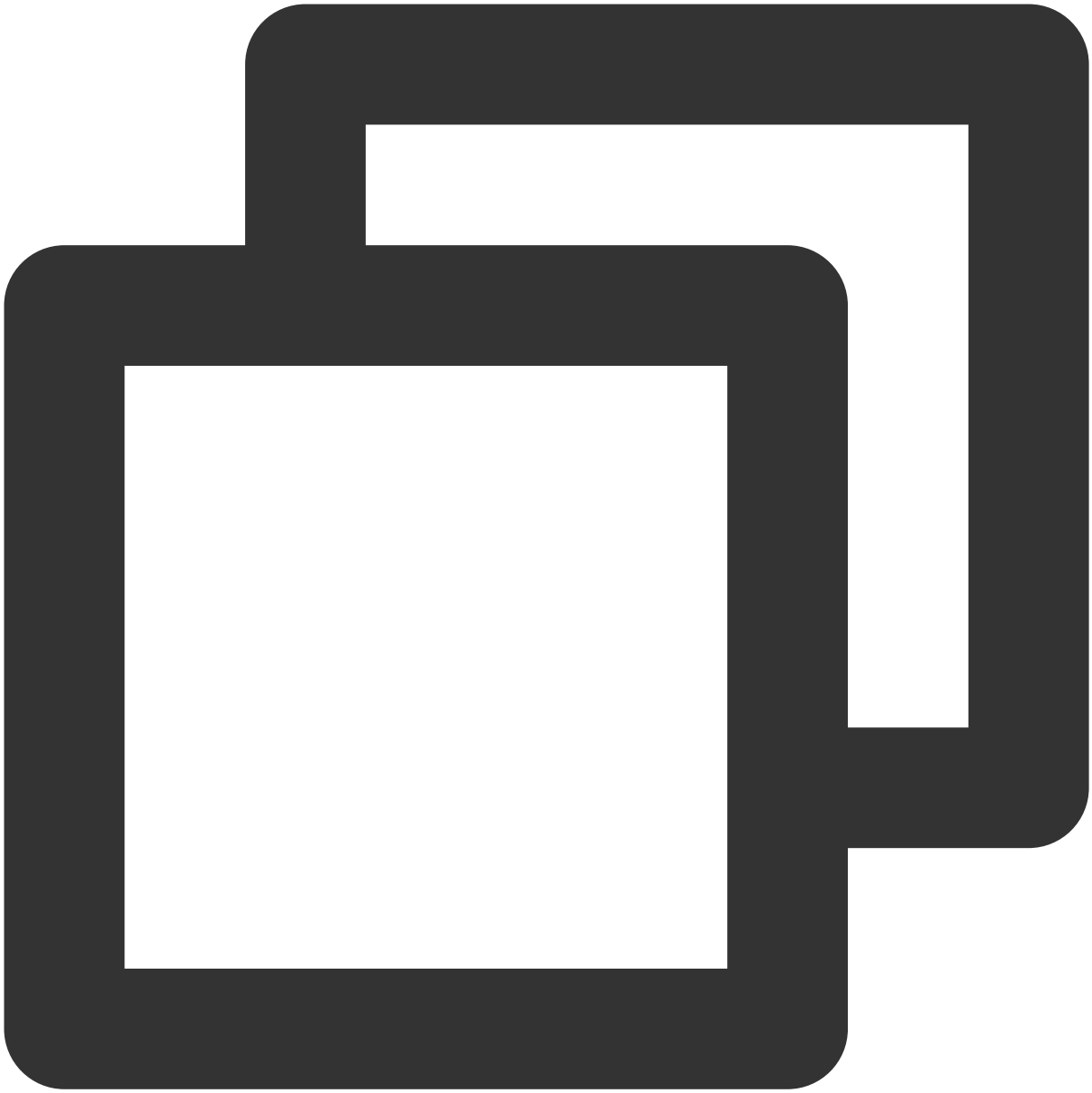
```
virtual void OnRemoteUserAudioAvailable(const std::string& user_id, bool available)
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。
available	bool	true：オーディオストリームデータあり、false：オーディオストリームデータなし。

OnRemoteUserEnterSpeechState

リモートユーザーが発言を開始します。



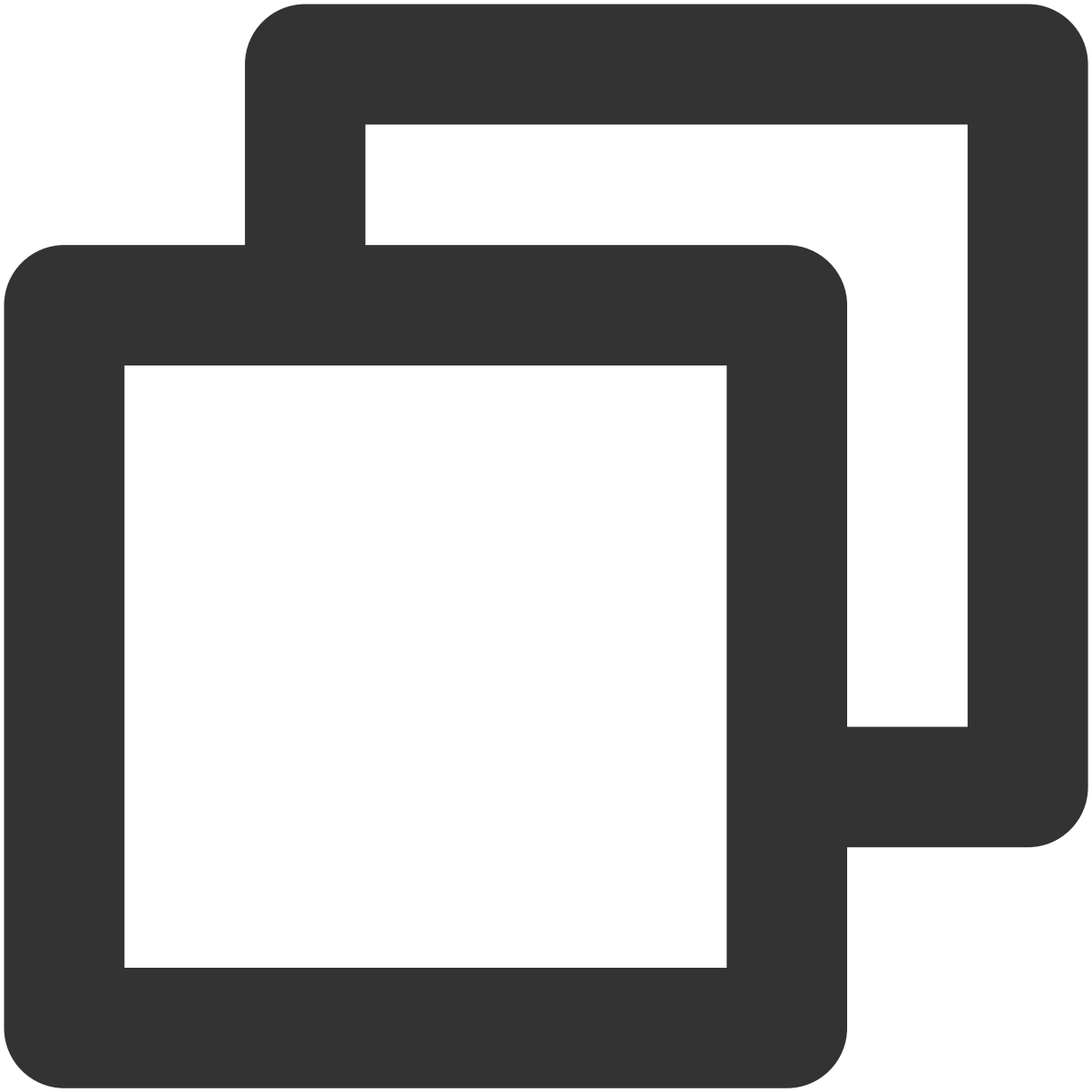
```
virtual void OnRemoteUserEnterSpeechState(const std::string& user_id) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。

OnRemoteUserExitSpeechState

リモートユーザーが発言を終了します。



```
virtual void OnRemoteUserExitSpeechState(const std::string& user_id) = 0;
```

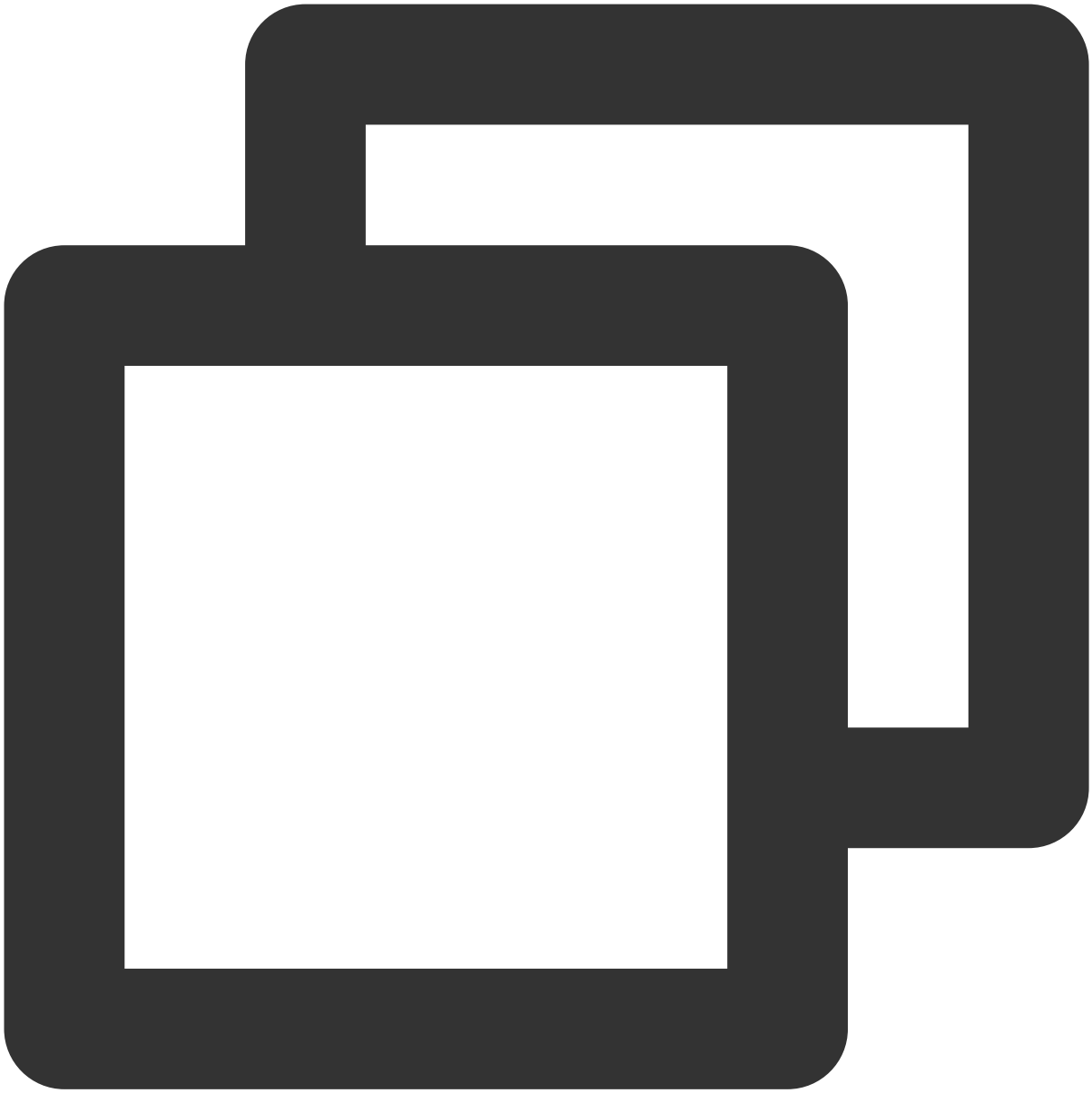
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。

チャットルームメッセージイベントコールバック

OnReceiveChatMessage

テキストメッセージの受信。



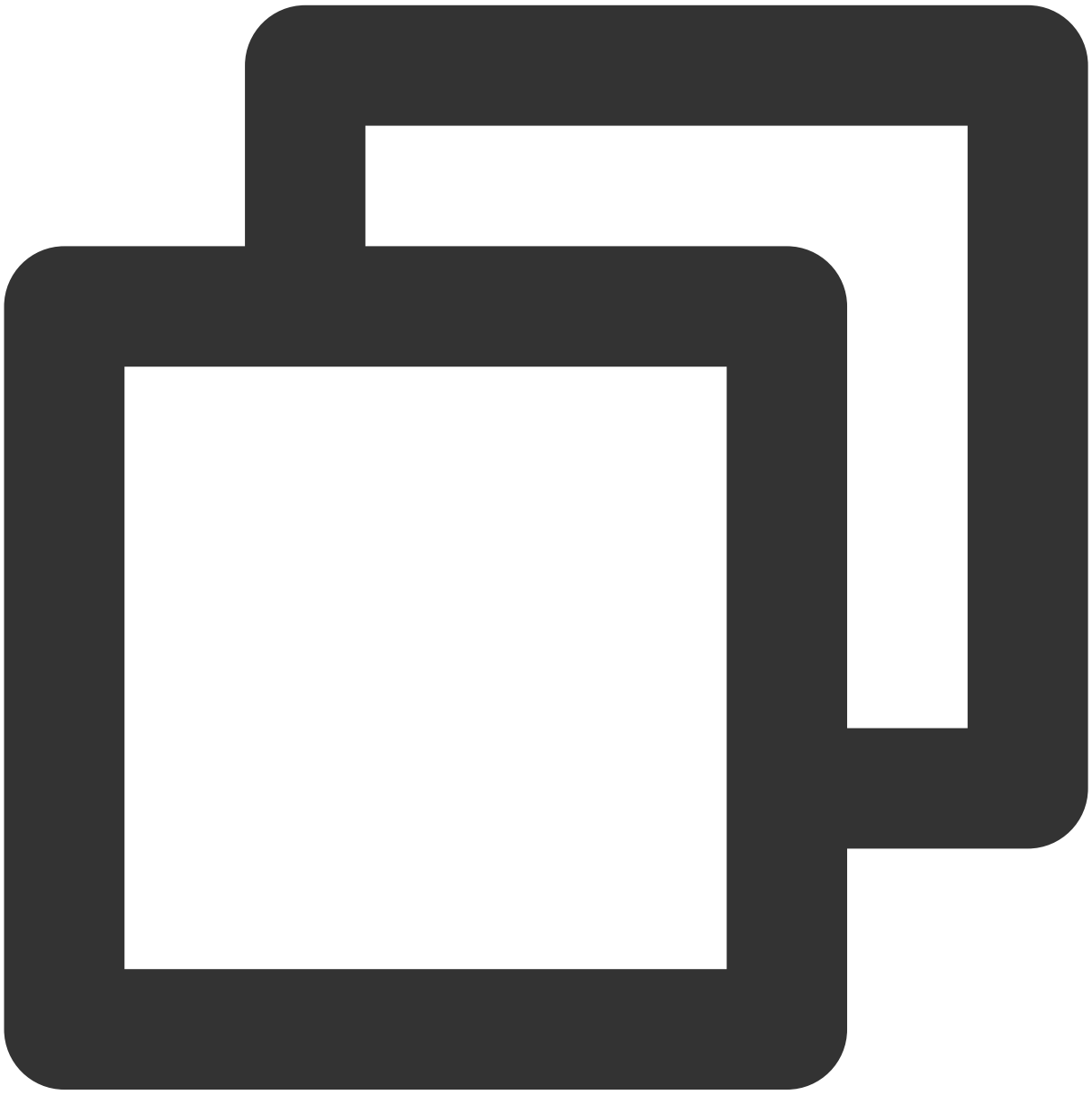
```
virtual void OnReceiveChatMessage(const std::string& user_id, const std::string& me
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。
message	string	テキストメッセージ。

OnReceiveCustomMessage

カスタムメッセージの受信。



```
virtual void OnReceiveCustomMessage(const std::string& user_id, const std::string&
```

パラメータは下表に示すとおりです。

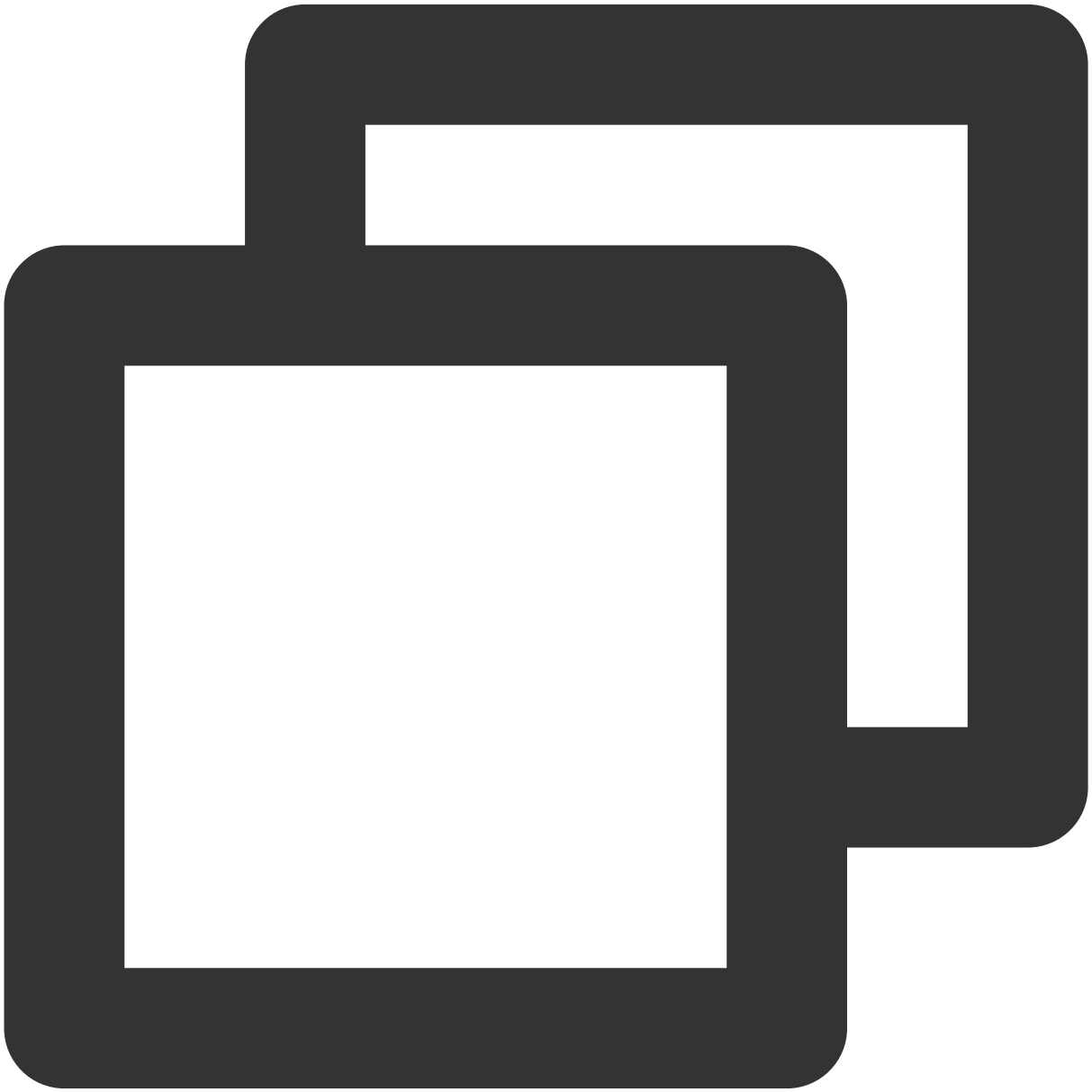
パラメータ	タイプ	意味
user_id	string	ユーザー ID。

message	string	カスタムメッセージ。

フィールドコントロールメッセージコールバック

OnReceiveSpeechInvitation

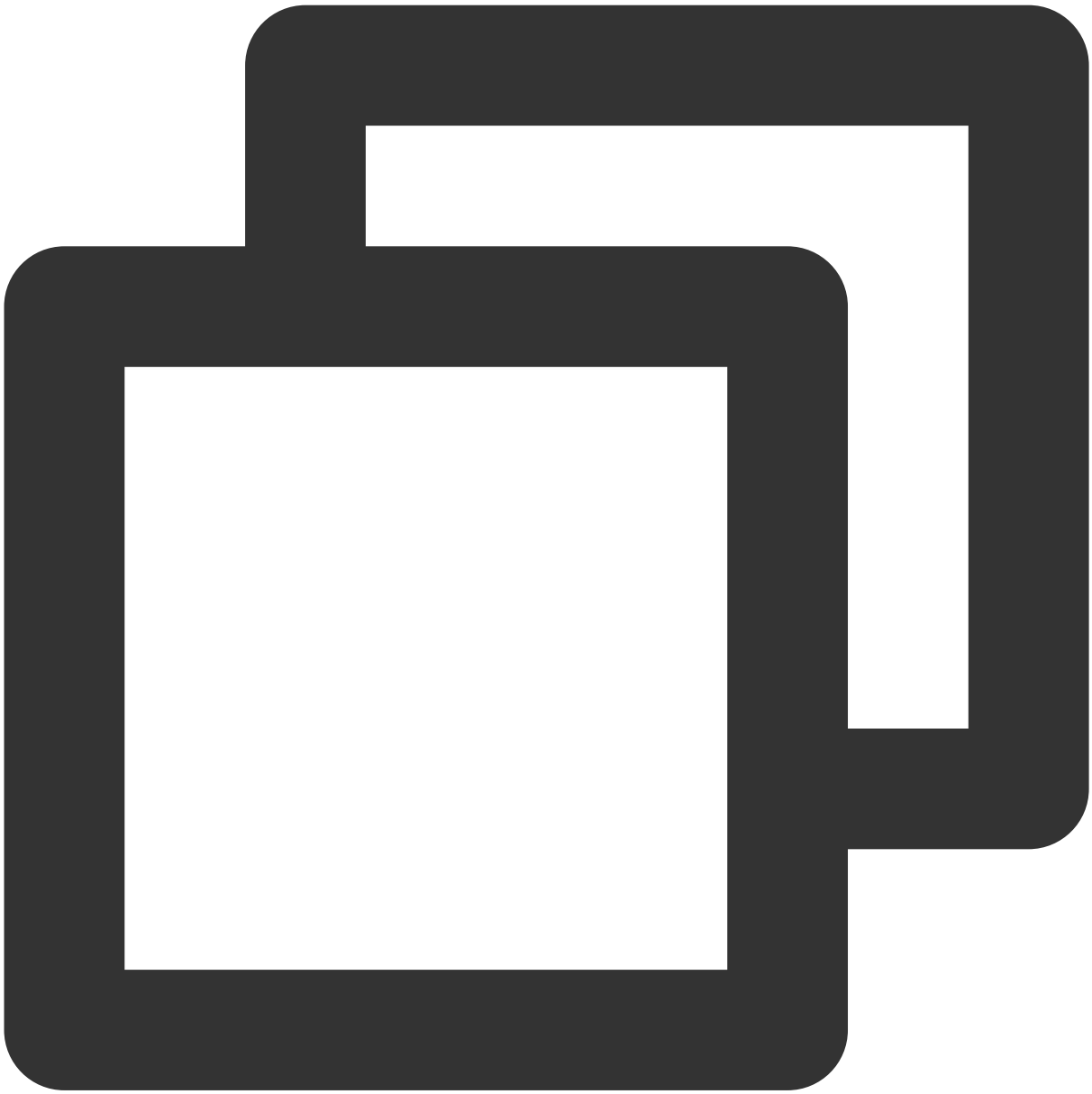
ユーザーがキャスターの発言要請を受信する場合のコールバック。



```
virtual void OnReceiveSpeechInvitation() = 0;
```

OnReceiveInvitationCancelled

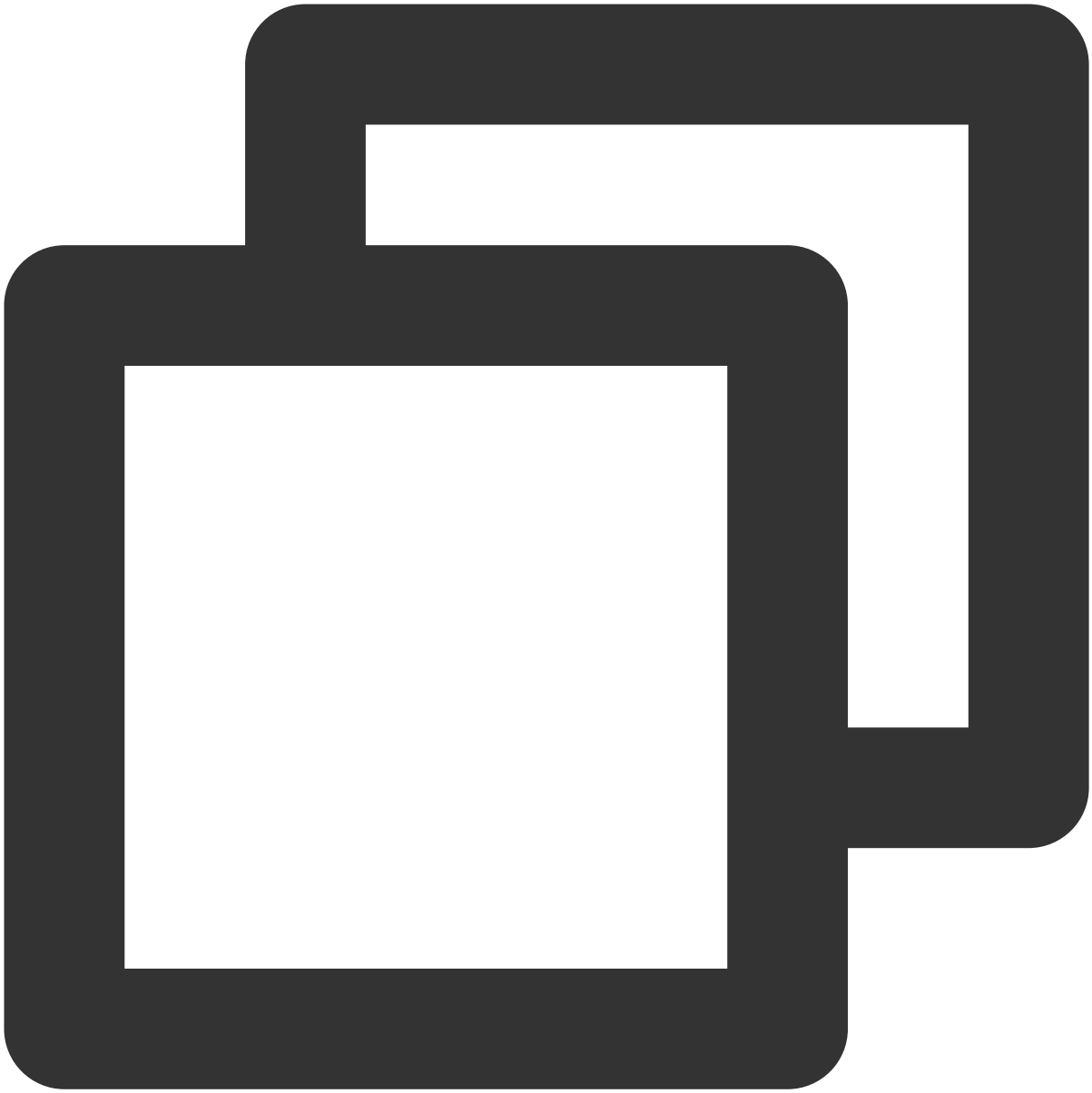
ユーザーがキャスターの発言要請キャンセルを受信する場合のコールバック。



```
virtual void OnReceiveInvitationCancelled() = 0;
```

OnReceiveReplyToSpeechInvitation

キャスターがユーザーの発言要請への同意を受信する場合のコールバック。



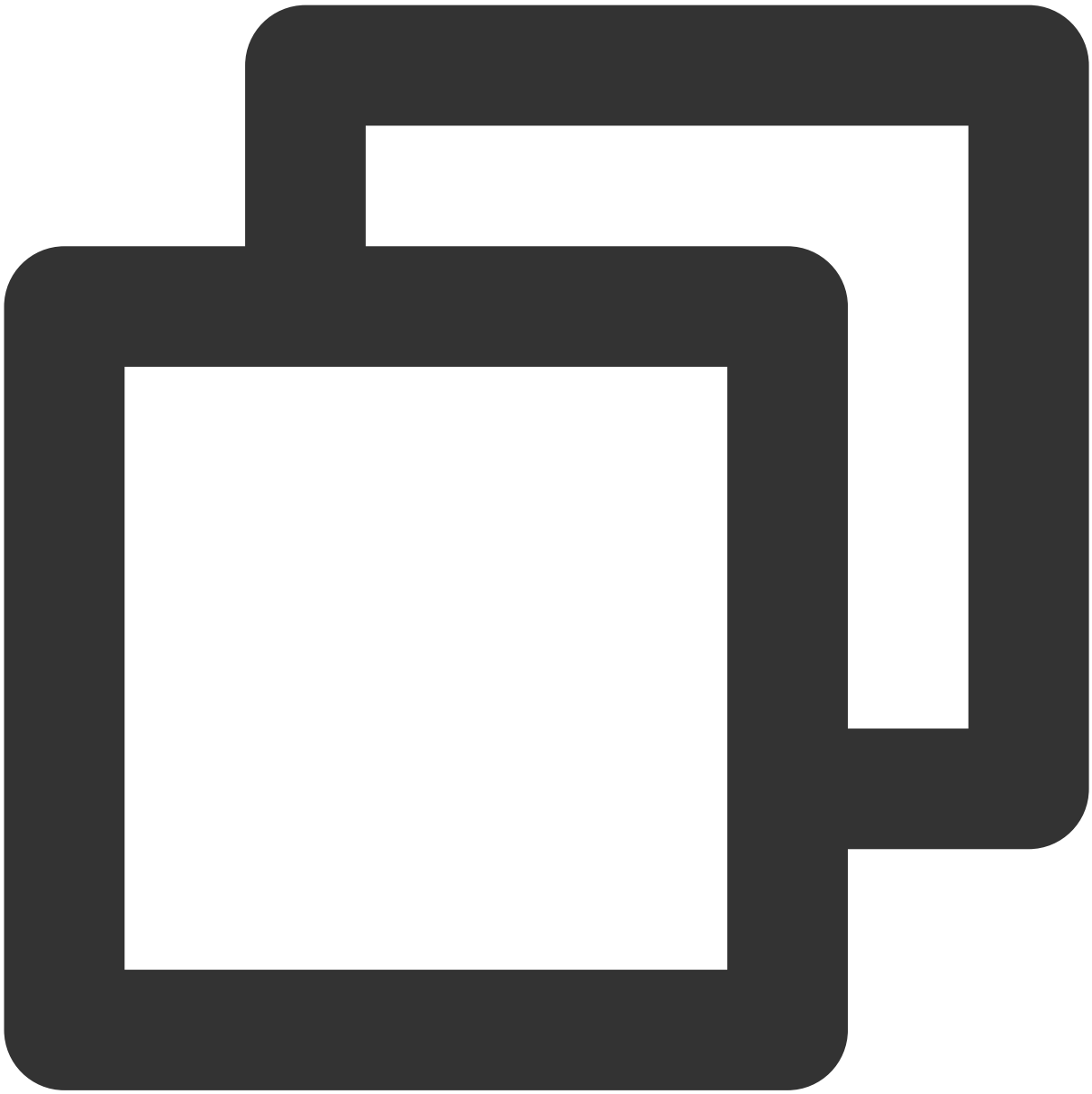
```
virtual void OnReceiveReplyToSpeechInvitation(const std::string& user_id, bool agre
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。
agree	bool	同意するかどうか。

OnReceiveSpeechApplication

キャスターがユーザーの発言申請を受信する場合のコールバック。



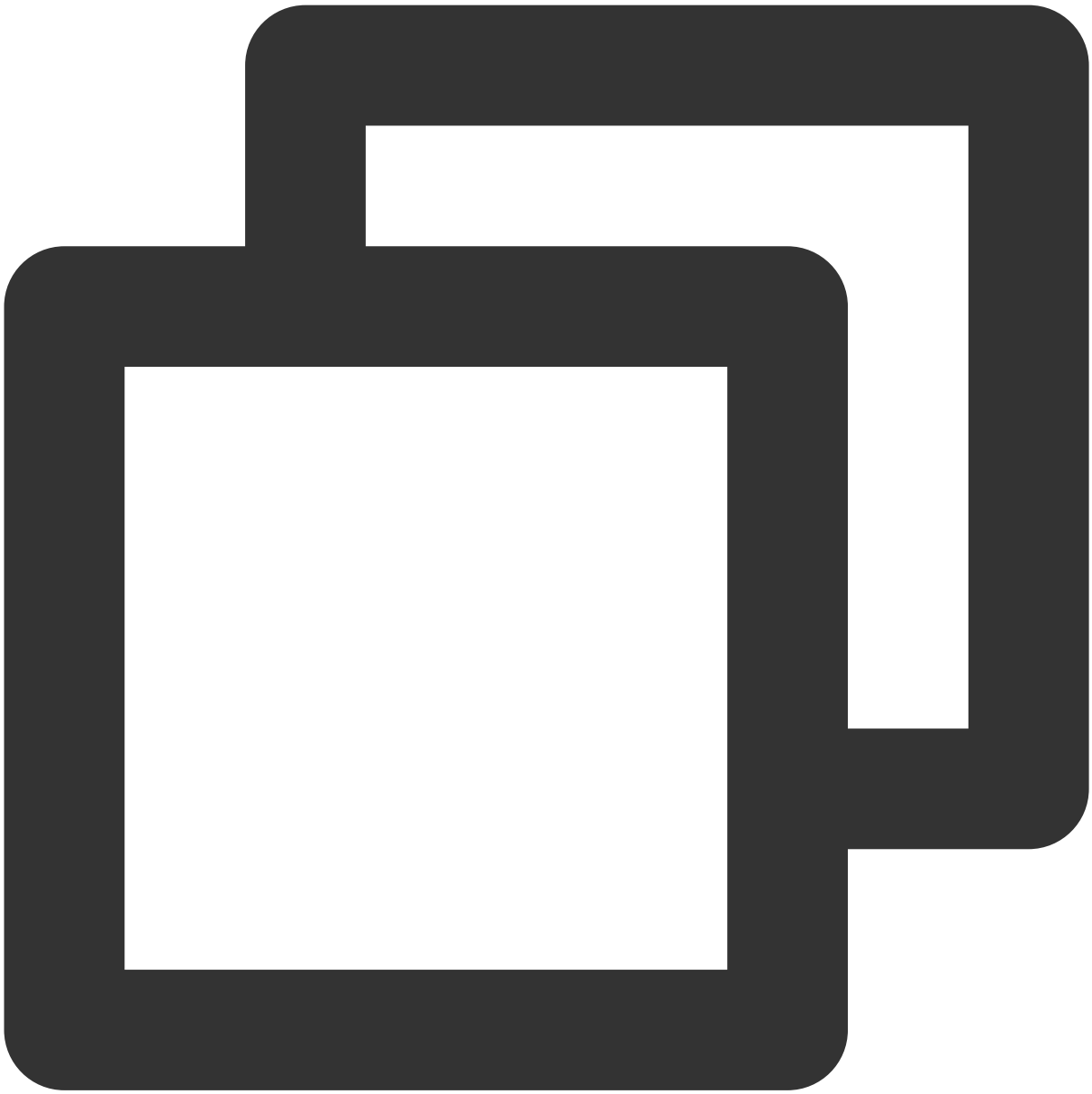
```
virtual void OnReceiveSpeechApplication(const std::string& user_id) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。

OnSpeechApplicationCancelled

ユーザーが発言申請をキャンセルする場合のコールバック。



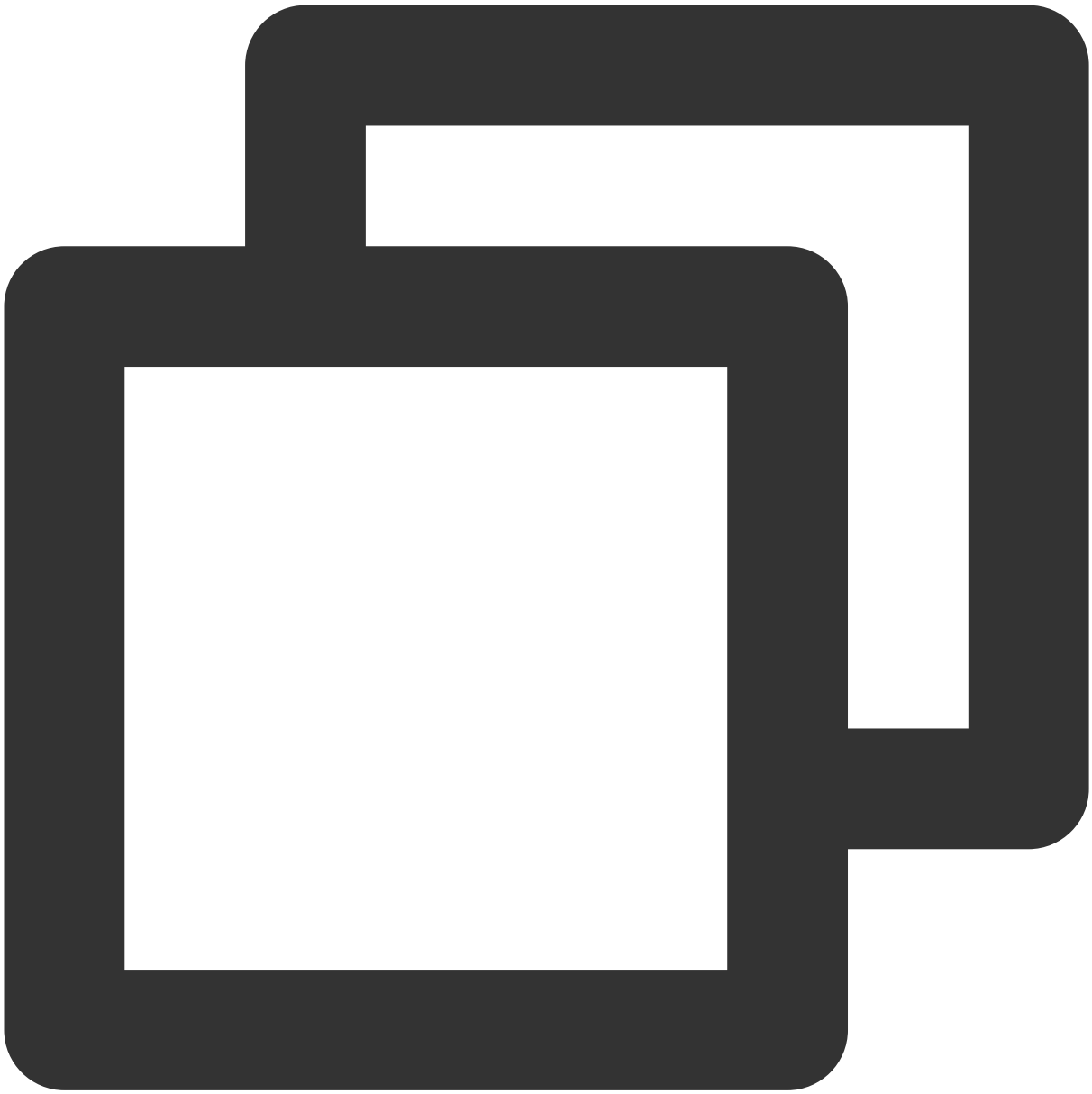
```
virtual void OnSpeechApplicationCancelled(const std::string& user_id) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。

OnReceiveReplyToSpeechApplication

キャスターが発言申請に同意する場合のコールバック。



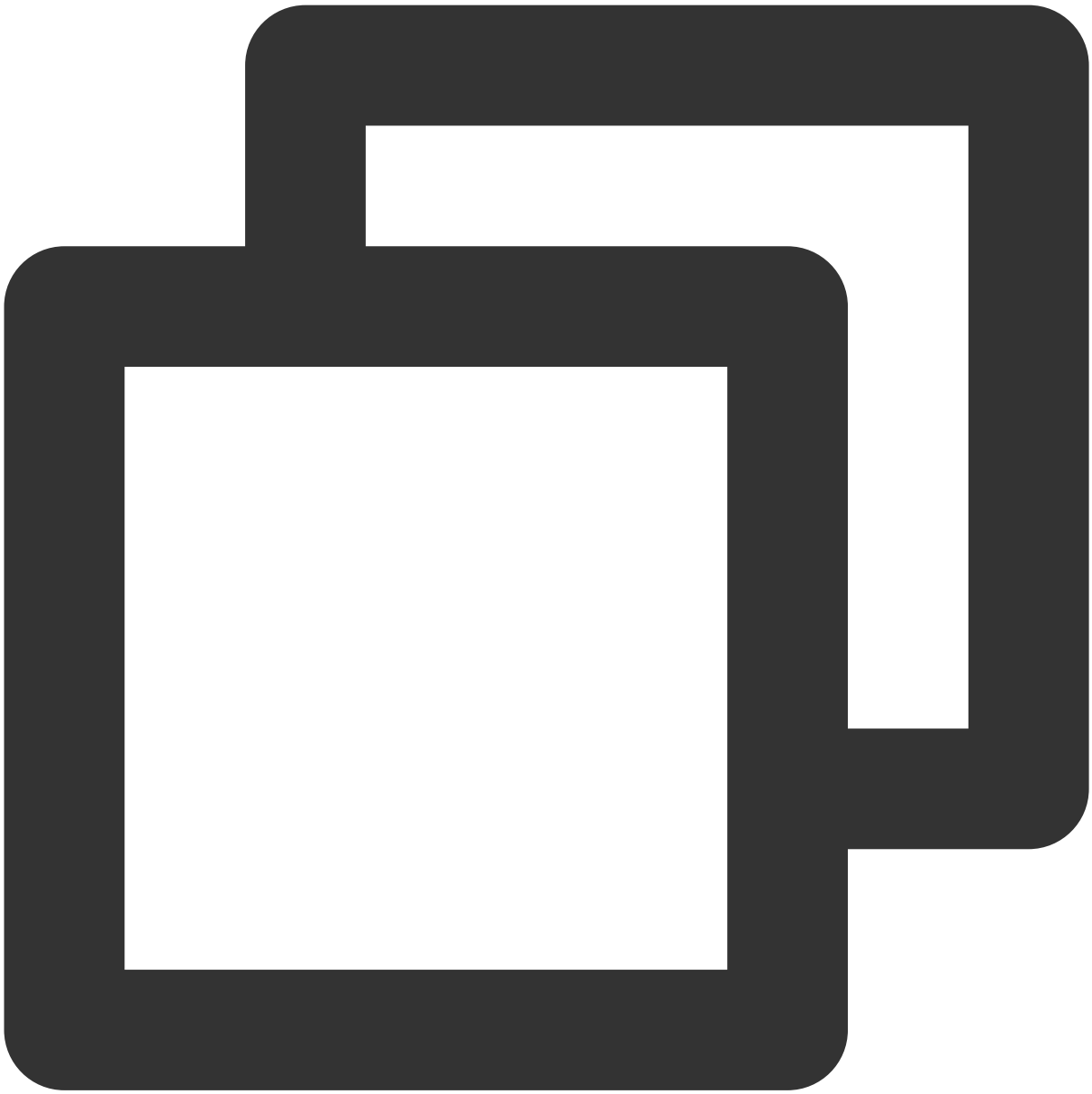
```
virtual void OnReceiveReplyToSpeechApplication(bool agree) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
agree	bool	同意するかどうか。

OnSpeechApplicationForbidden

キャスターが発言申請を禁止する場合のコールバック。



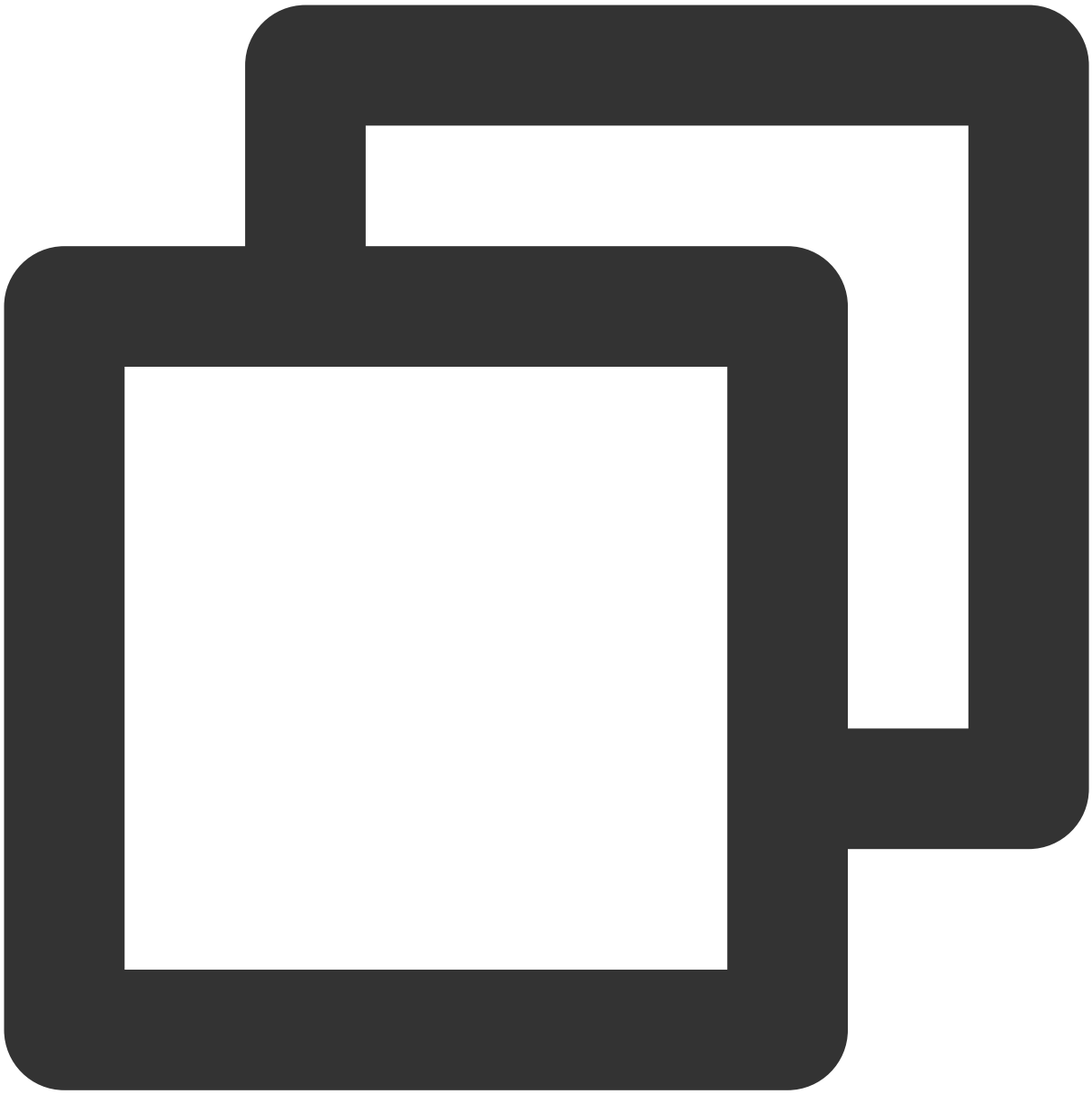
```
virtual void OnSpeechApplicationForbidden(bool forbidden) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
forbidden	bool	禁止するかどうか。

OnOrderedToExitSpeechState

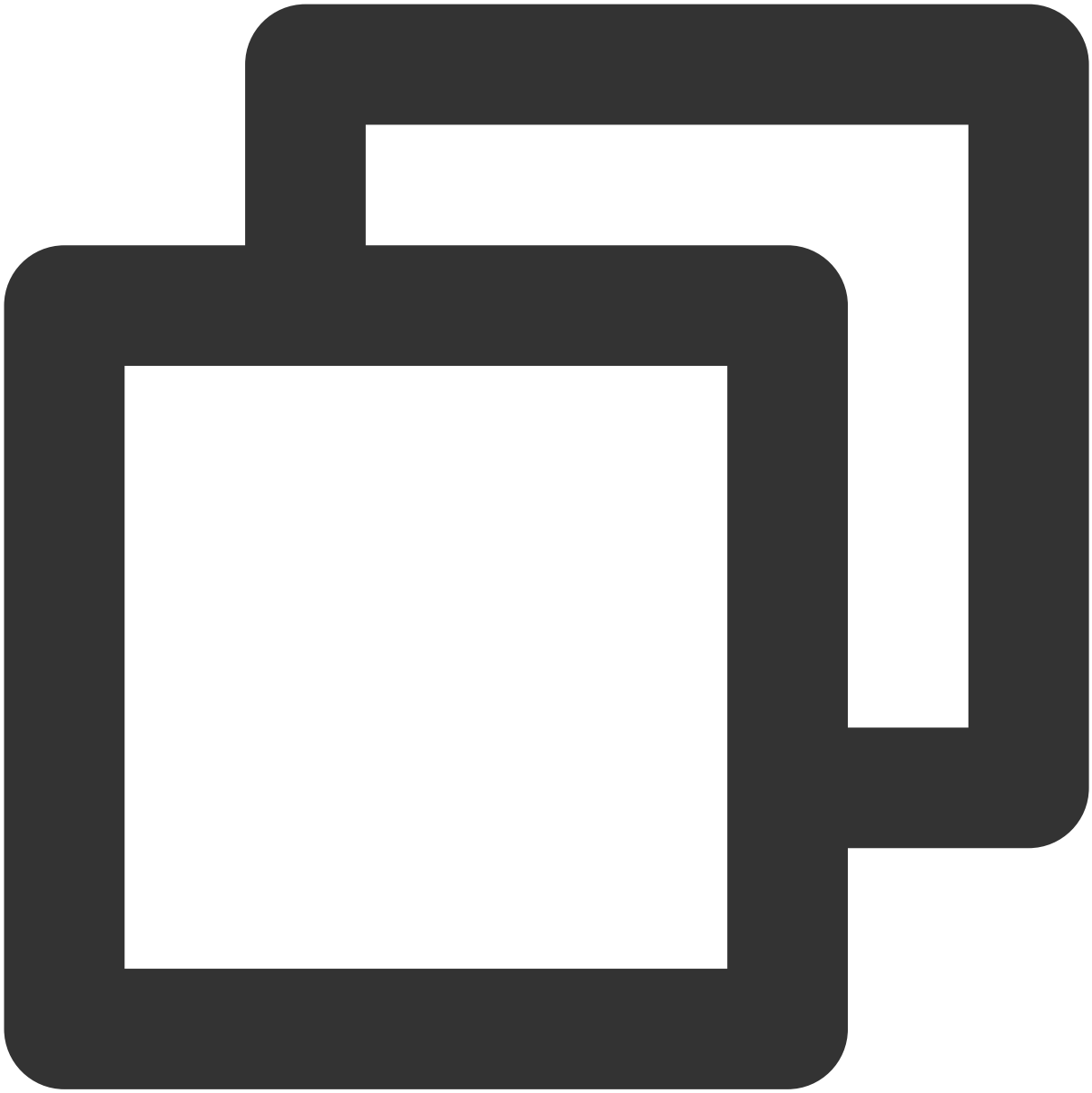
参加者が発言を停止するようリクエストされる場合のコールバック。



```
virtual void OnOrderedToExitSpeechState() = 0;
```

OnCallingRollStarted

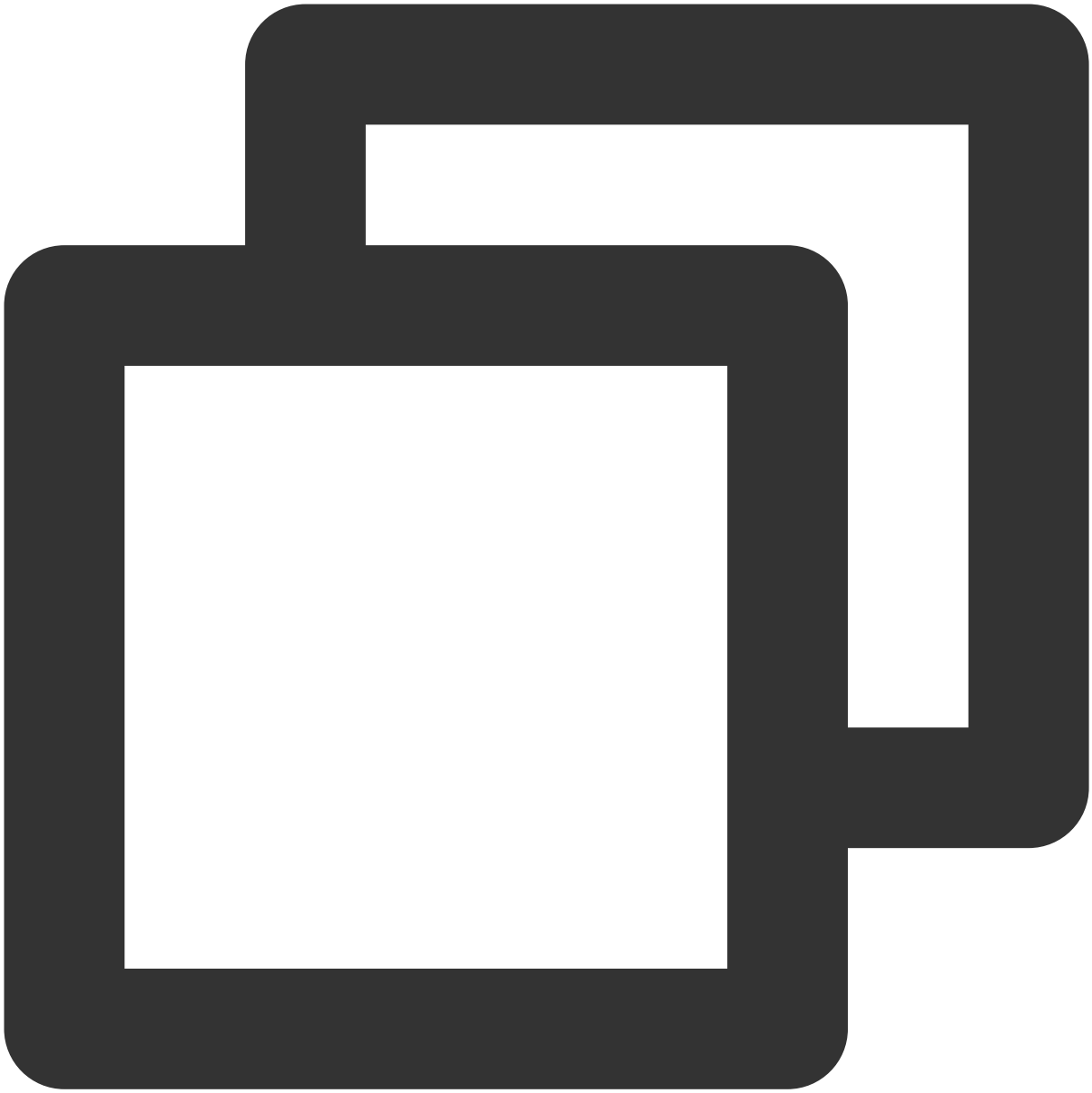
キャスターが点呼を開始し、メンバーが受信する場合のコールバック。



```
virtual void OnCallingRollStarted() = 0;
```

OnCallingRollStopped

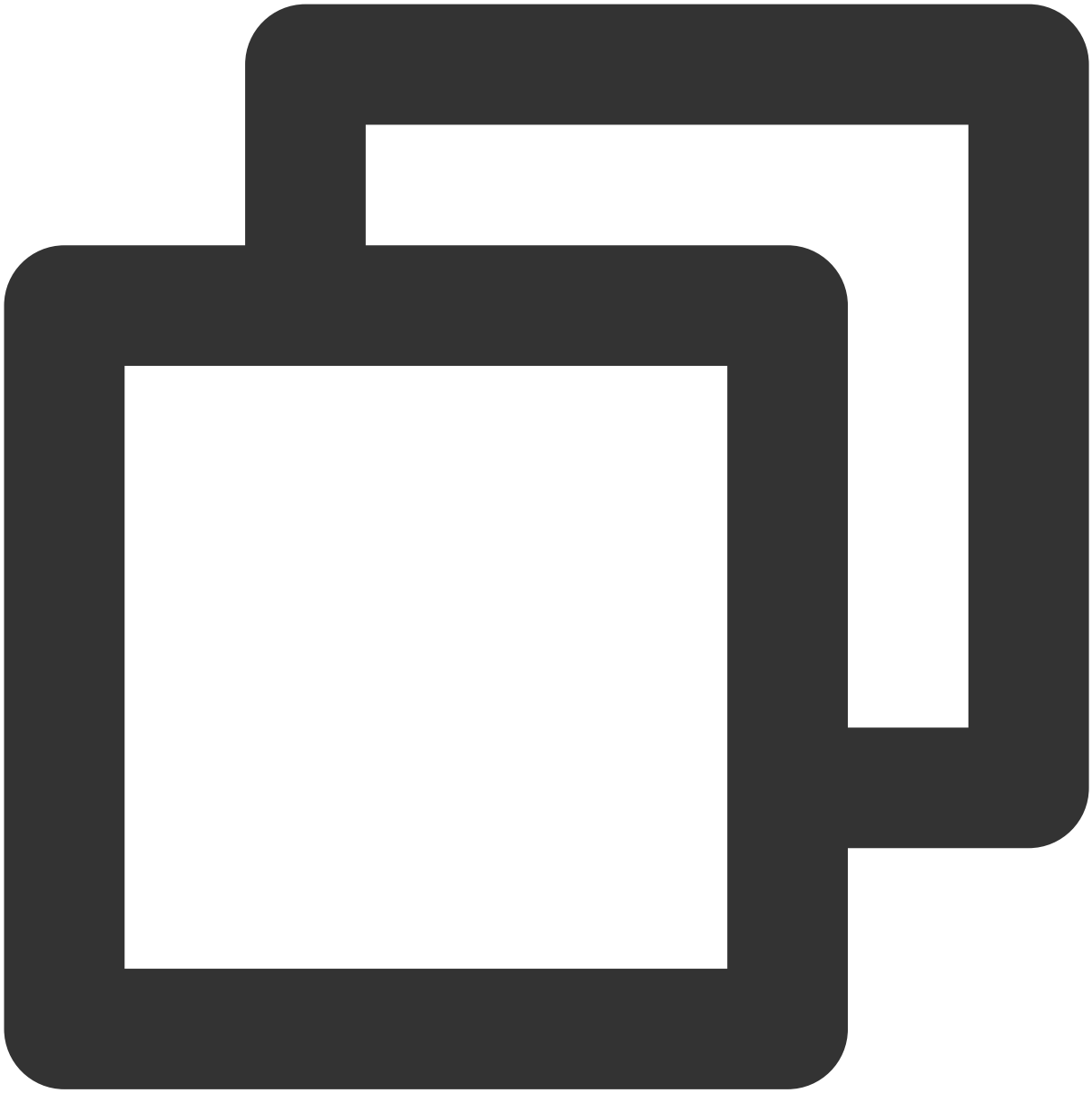
キャスターが点呼を終了し、参加者が受信する場合のコールバック。



```
virtual void OnCallingRollStopped() = 0;
```

OnMemberReplyCallingRoll

参加者が点呼に応答し、キャスターが受信する場合のコールバック。



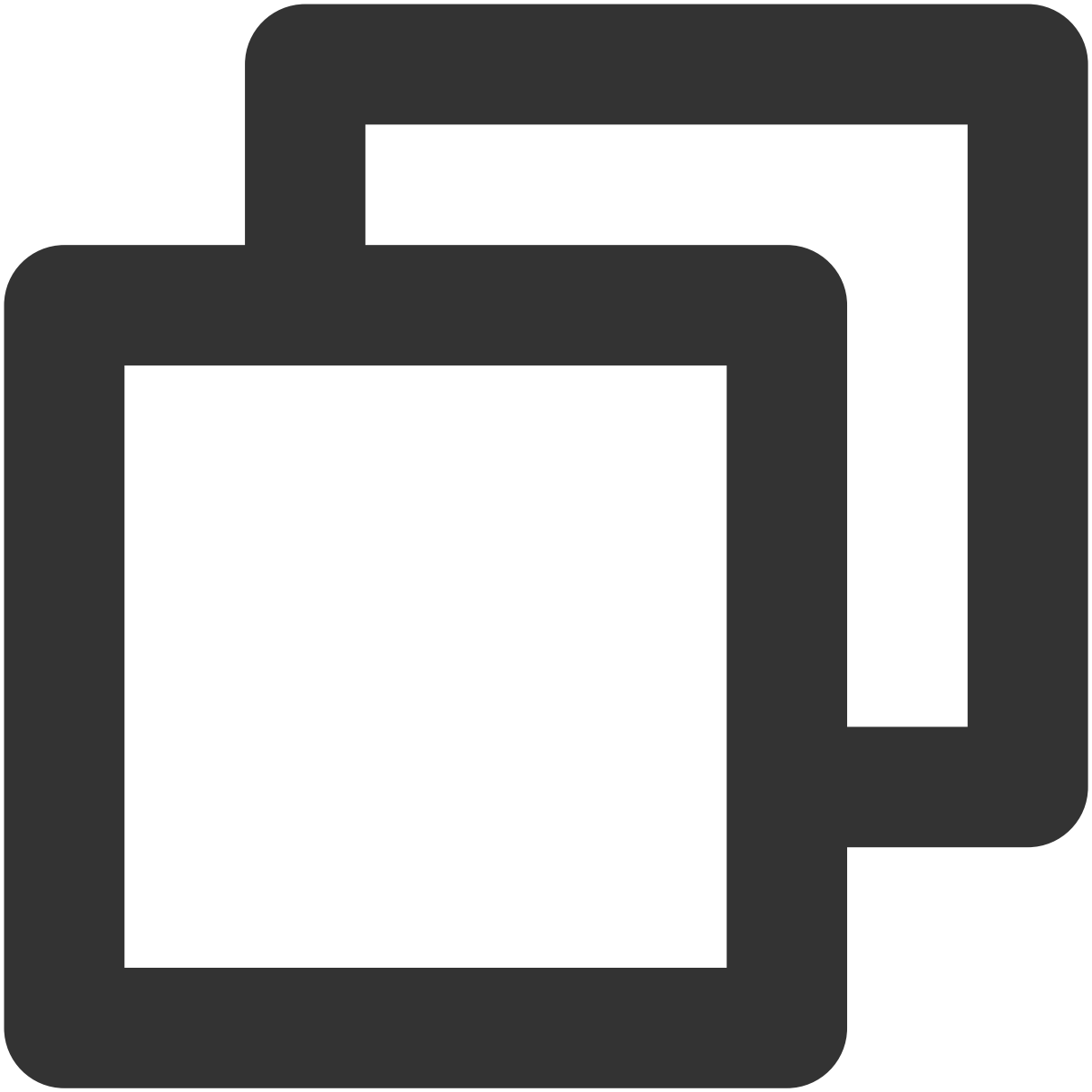
```
virtual void OnMemberReplyCallingRoll(const std::string& user_id) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
user_id	string	ユーザー ID。

OnChatRoomMuted

キャスターがチャットルームのミュートを変更する場合のコールバック。



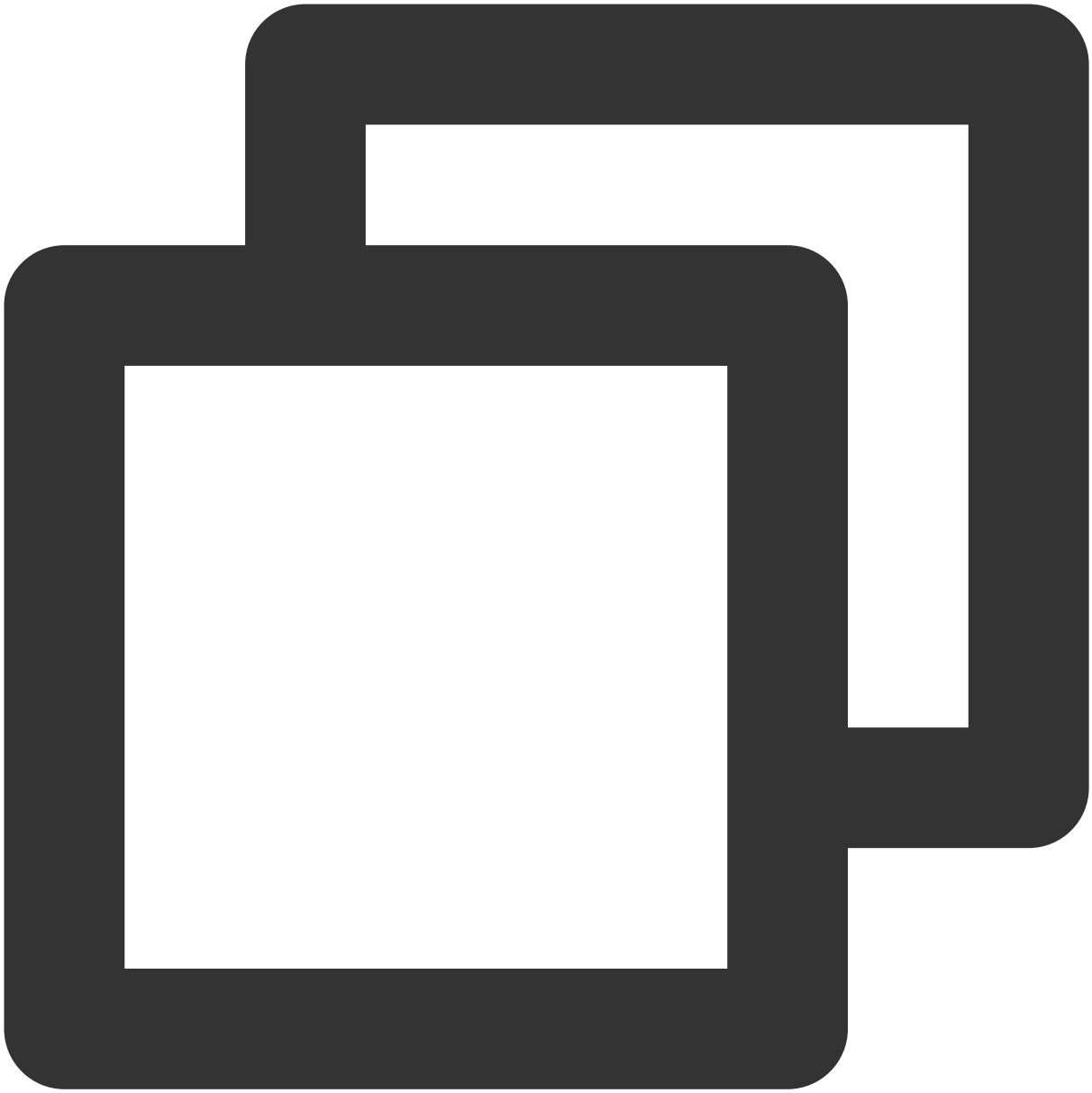
```
virtual void OnChatRoomMuted(bool muted) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
muted	bool	無効にするかどうか。

OnMicrophoneMuted

キャスターがマイクの無効化を設定する場合のコールバック。



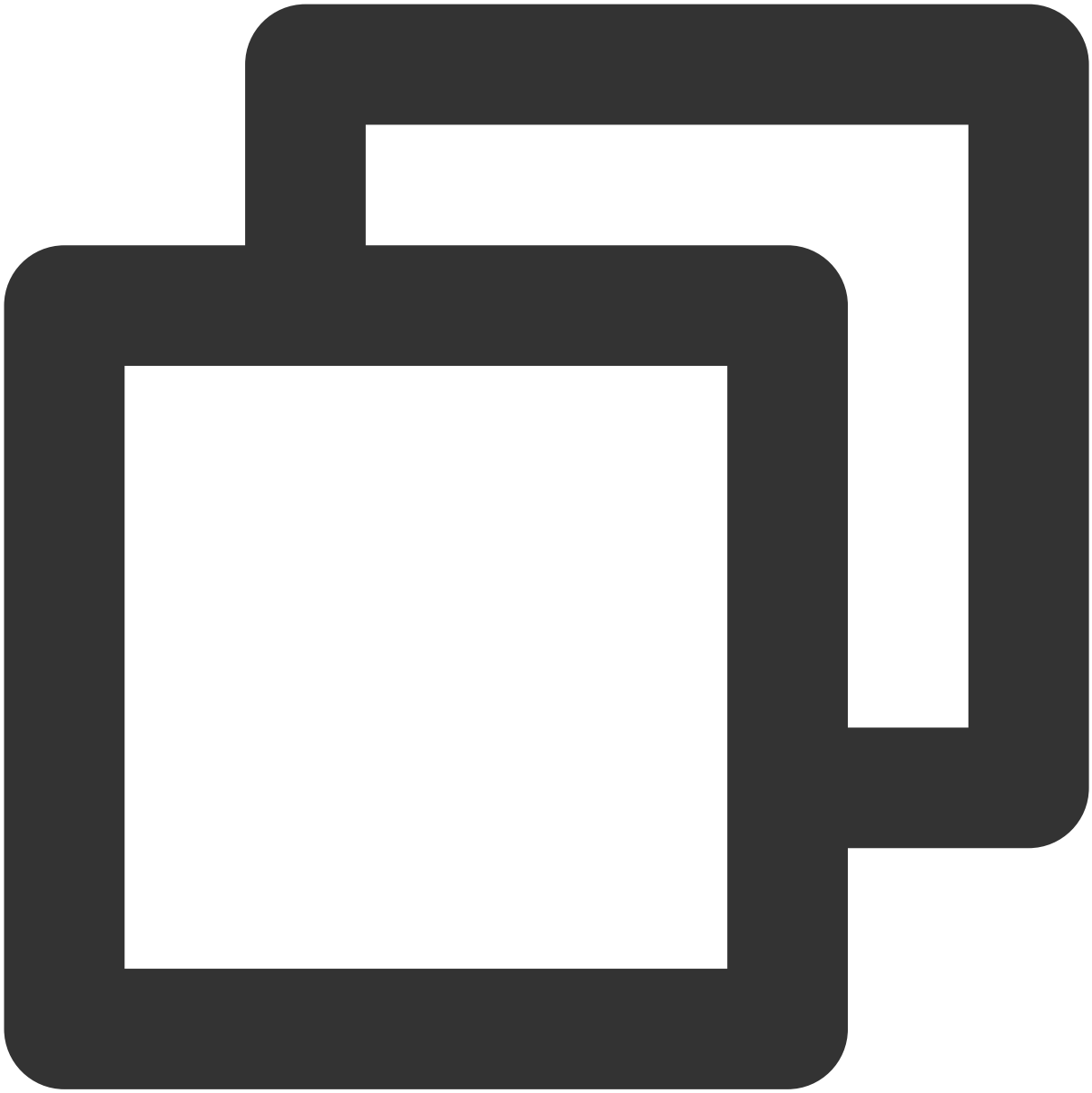
```
virtual void OnMicrophoneMuted(bool muted) = 0;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
muted	bool	無効にするかどうか。

OnCameraMuted

キャスターがカメラの無効化を設定する場合のコールバック。



```
virtual void OnCameraMuted(bool muted) = 0;
```

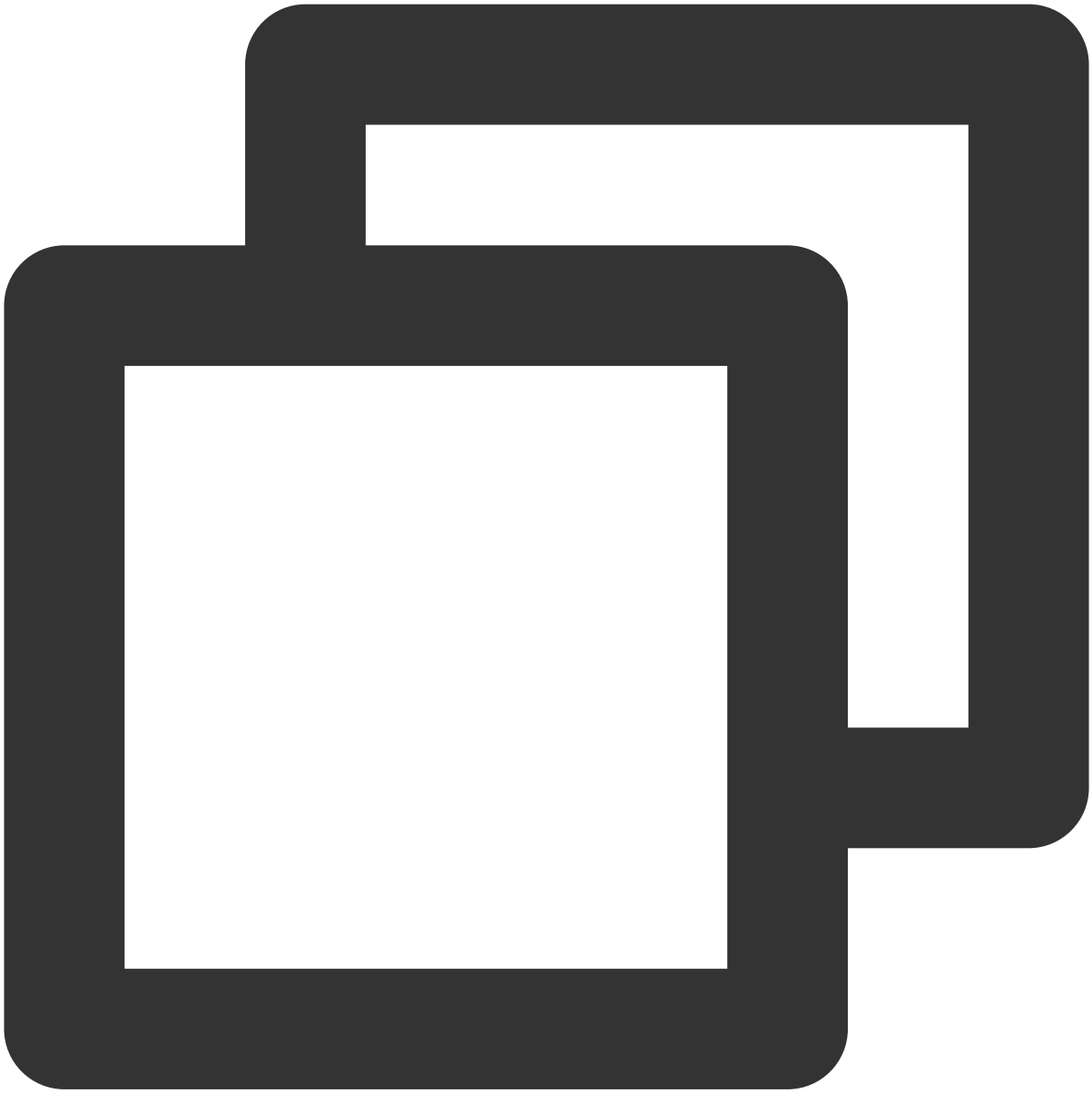
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
muted	bool	無効にするかどうか。

統計および品質コールバック

OnStatistics

技術指標統計のコールバック。



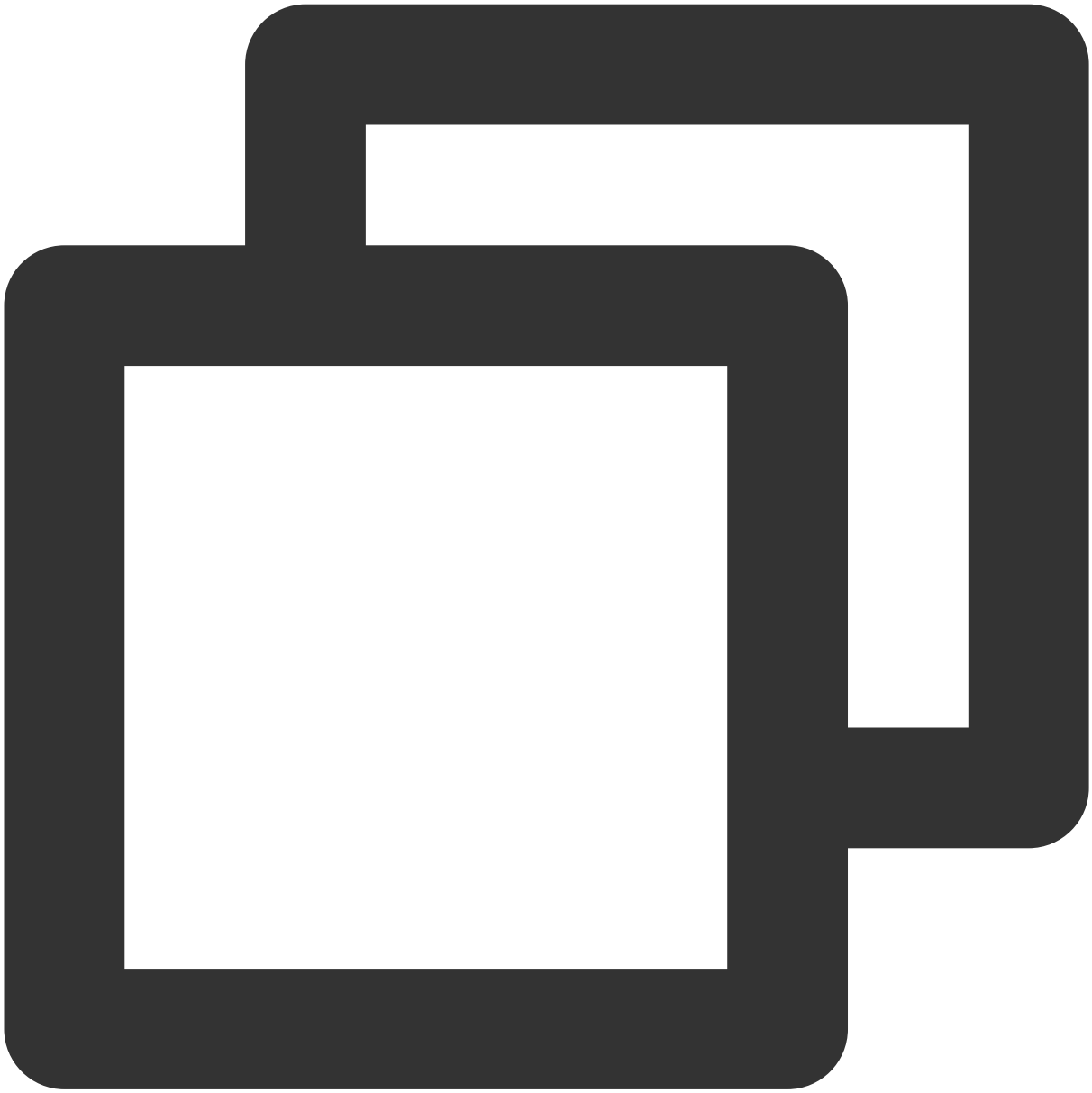
```
virtual void OnStatistics(const liteav::TRTCStatistics& statis) {}
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
statis	liteav::TRTCStatistics	統計データ。

OnNetworkQuality

ネットワーク状況のコールバック。



```
virtual void OnNetworkQuality(const liteav::TRTCQualityInfo& local_quality, liteav:
        uint32_t remote_quality_count) {}
```

パラメータは下表に示すとおりです。

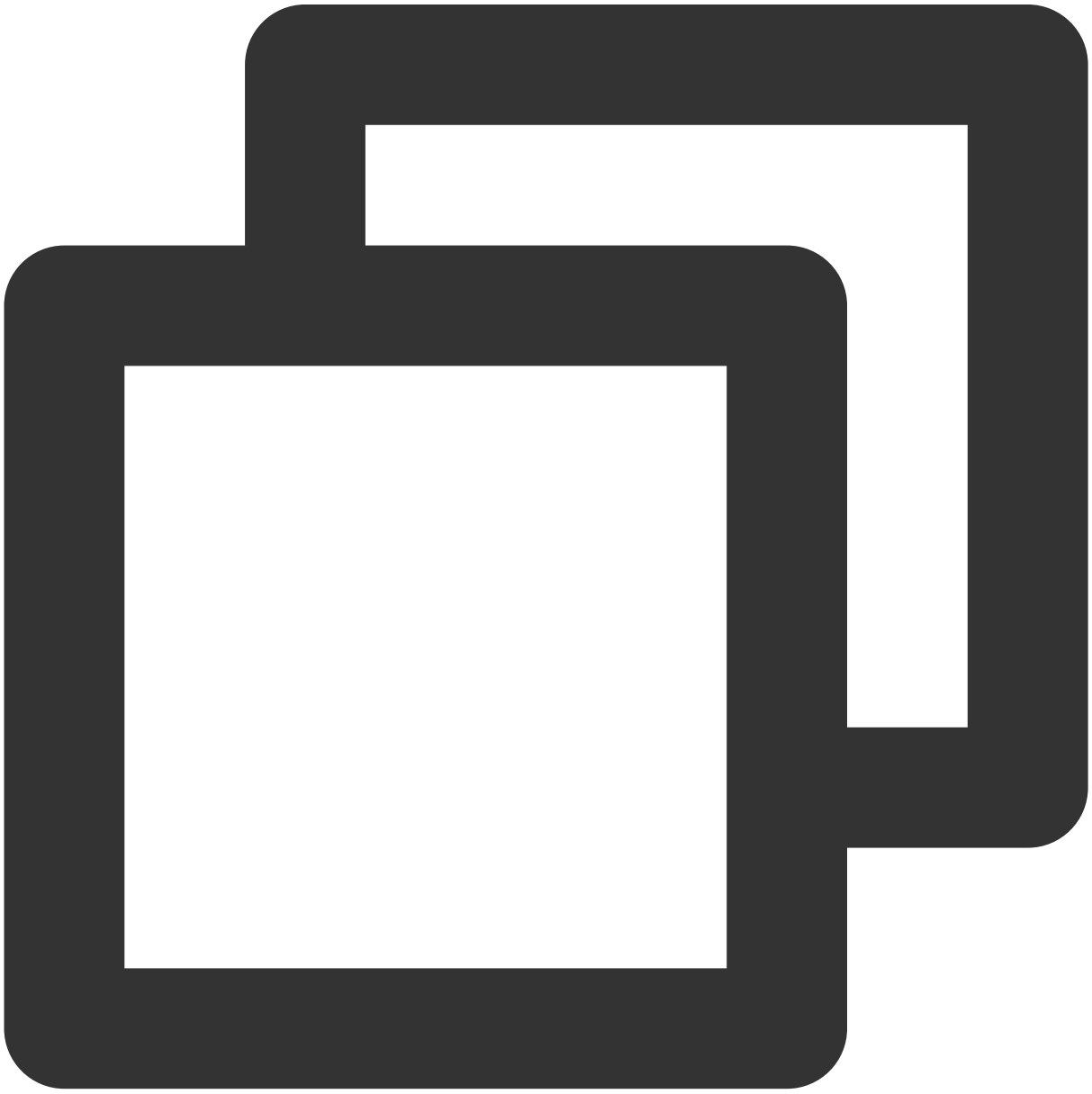
パラメータ	タイプ	意味
local_quality	liteav::TRTCQualityInfo	ローカルユーザー品質情報。

remote_quality	liteav::TRTCQualityInfo*	リモートユーザー品質情報ポインタ。
remote_quality_count	uint32_t	リモートユーザー数。

スクリーンキャプチャのイベントコールバック

OnScreenCaptureStarted

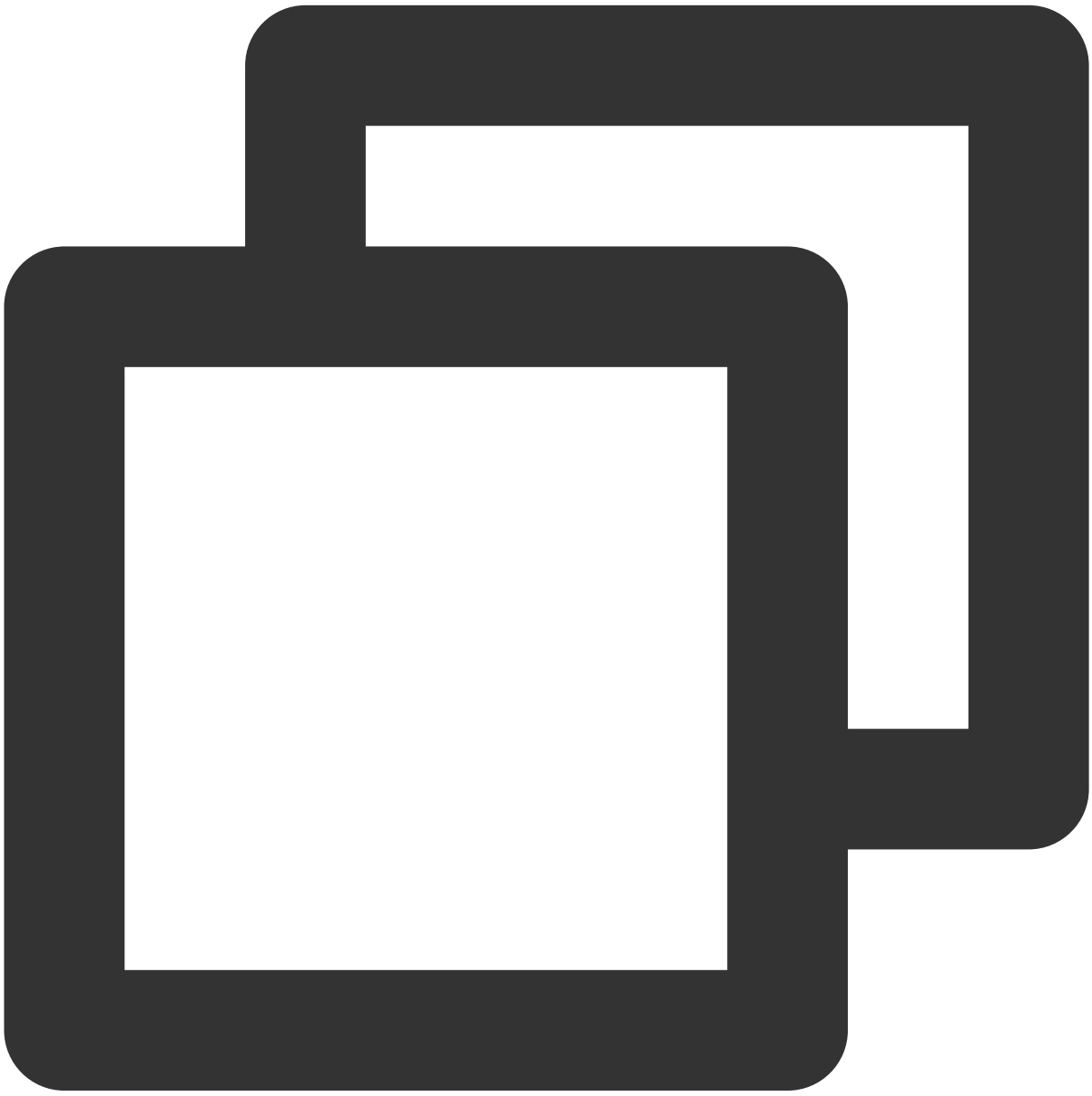
画面共有開始のコールバック。



```
virtual void OnScreenCaptureStarted() {}
```

OnScreenCaptureStopped

画面共有停止のコールバック。



```
void OnScreenCaptureStopped(int reason) {}
```

パラメータは下表に示すとおりです。

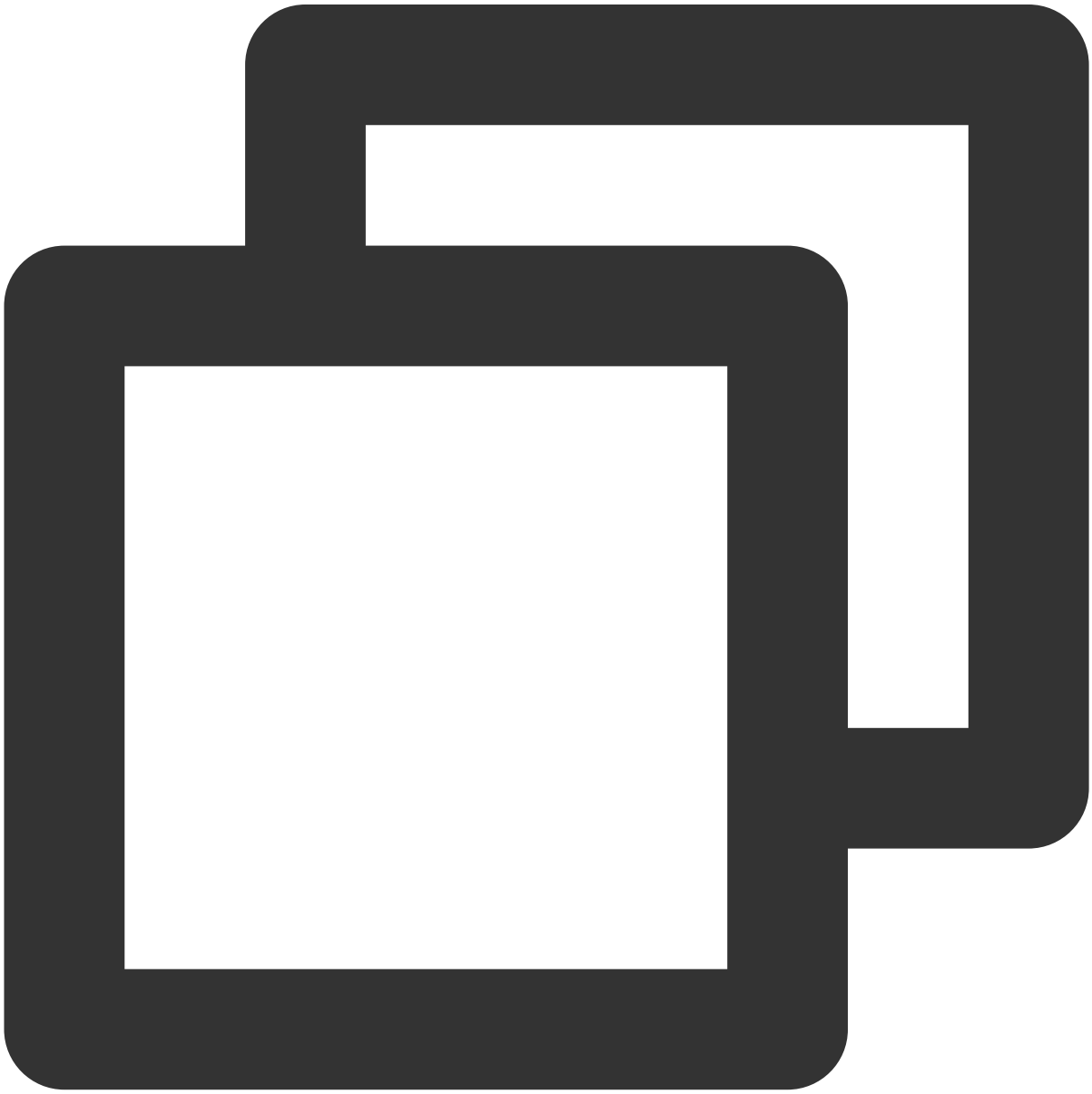
パラ	タイ	意味

メータ	プ	
reason	int	停止の理由。0：ユーザーの自発的な停止。1：その他アプリケーションに占有されたことによる停止。

ビデオレコーディングコールバック

OnRecordError

レコーディングエラーのコールバック。



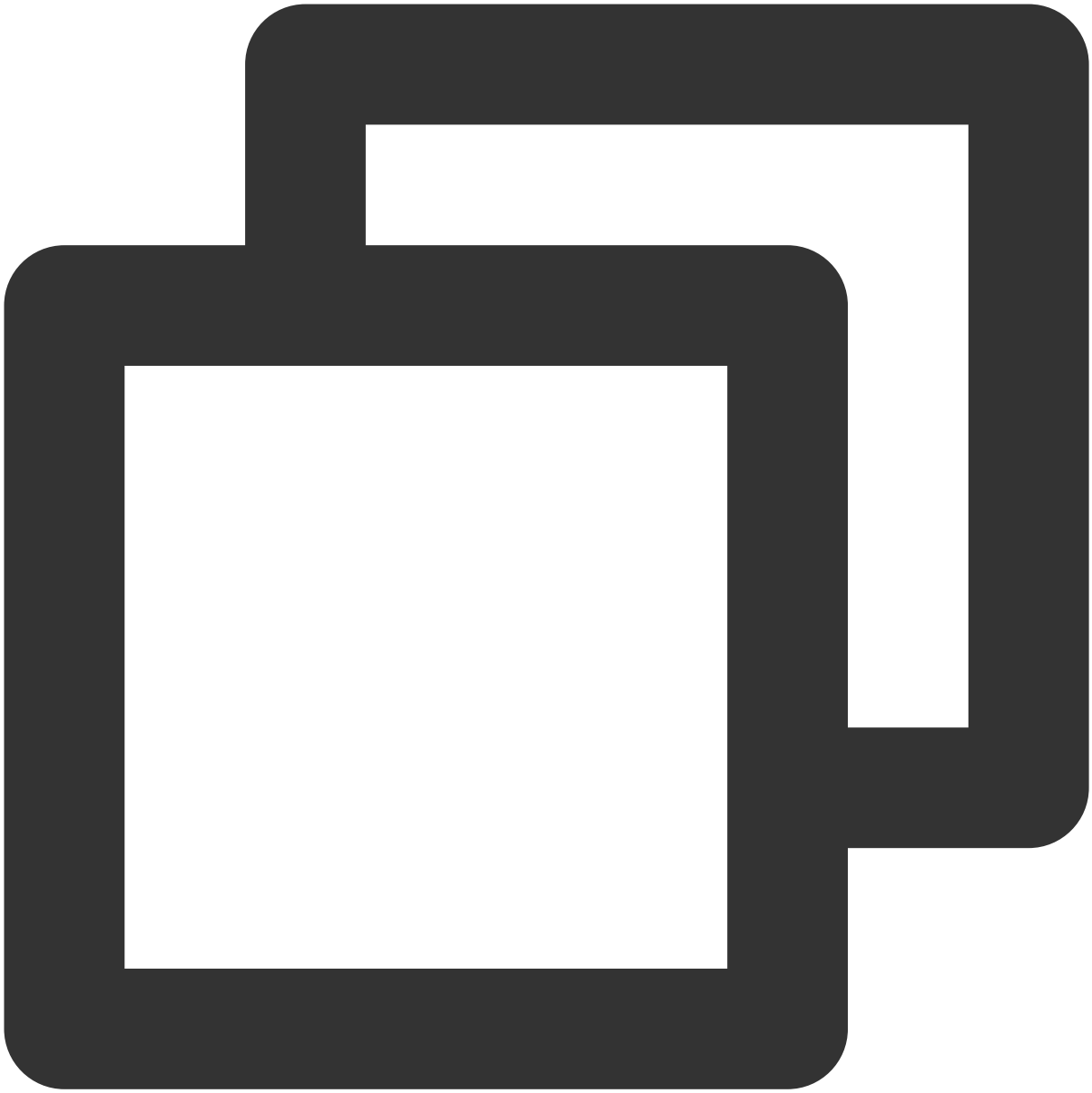
```
virtual void OnRecordError(TXLiteAVLocalRecordError error, const std::string& messg
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
error	TXLiteAVLocalRecordError	エラー情報。
messgae	string	エラー説明。

OnRecordComplete

レコーディング完了のコールバック。



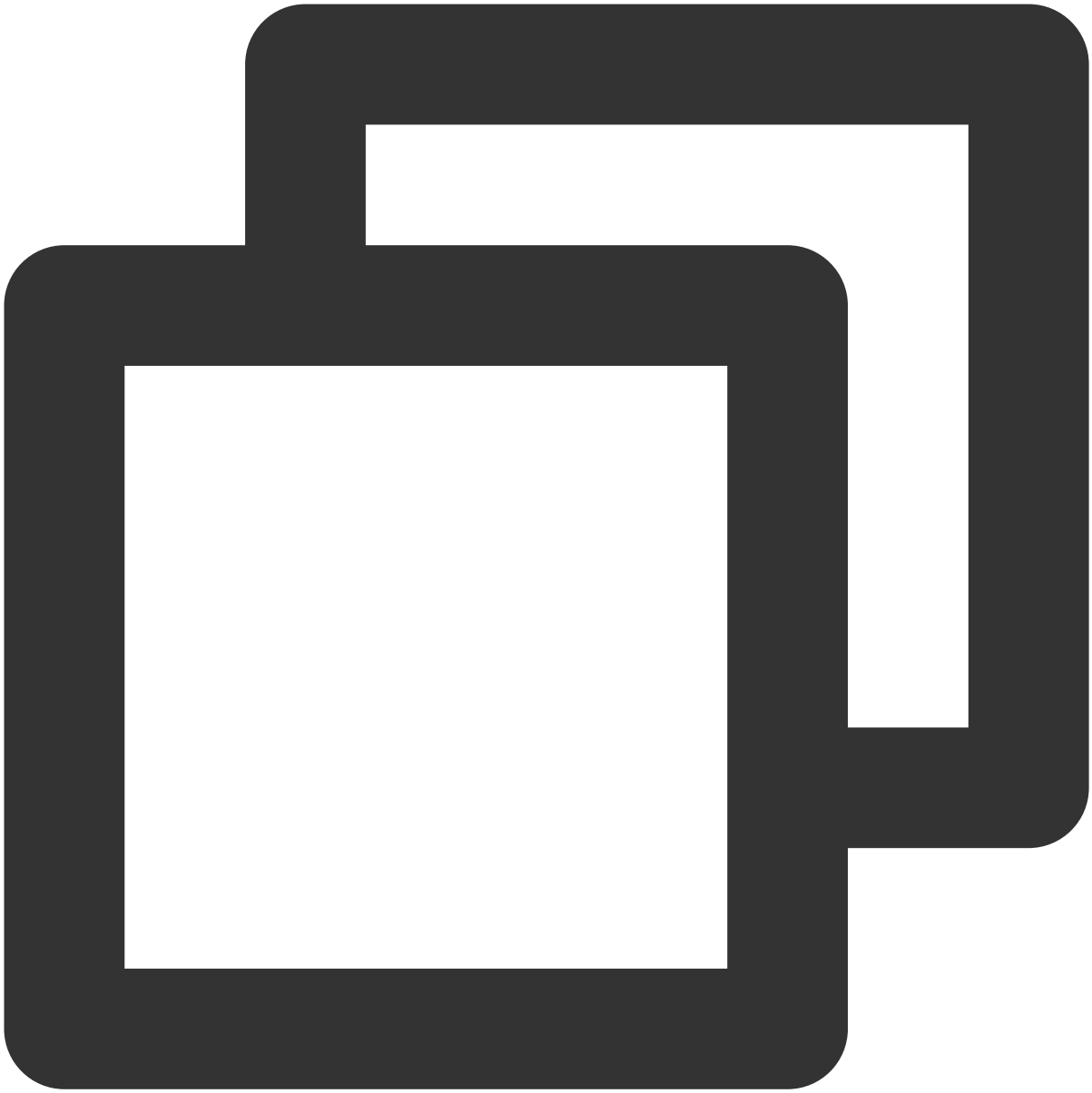
```
virtual void OnRecordComplete(const std::string& path) {}
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
path	string	エラーの説明。

OnRecordProgress

レコーディング進捗のコールバック。



```
virtual void OnRecordProgress(int duration, int file_size) {}
```

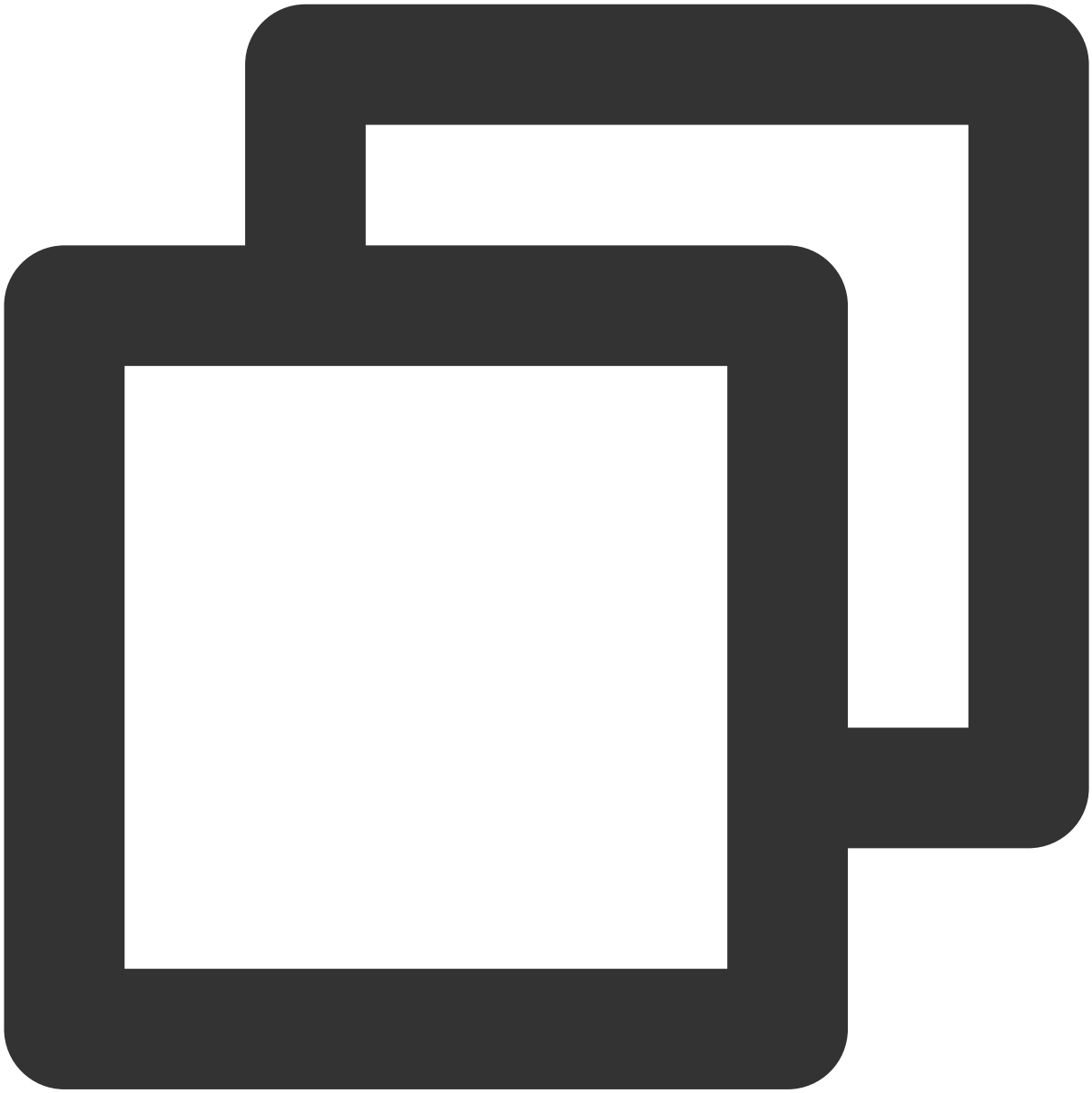
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
duration	int	ファイルの長さ。
file_size	int	ファイルのサイズ。

ローカルデバイステストコールバック

OnTestSpeakerVolume

スピーカー音量の大きさのコールバック。



```
virtual void OnTestSpeakerVolume(uint32_t volume) {}
```

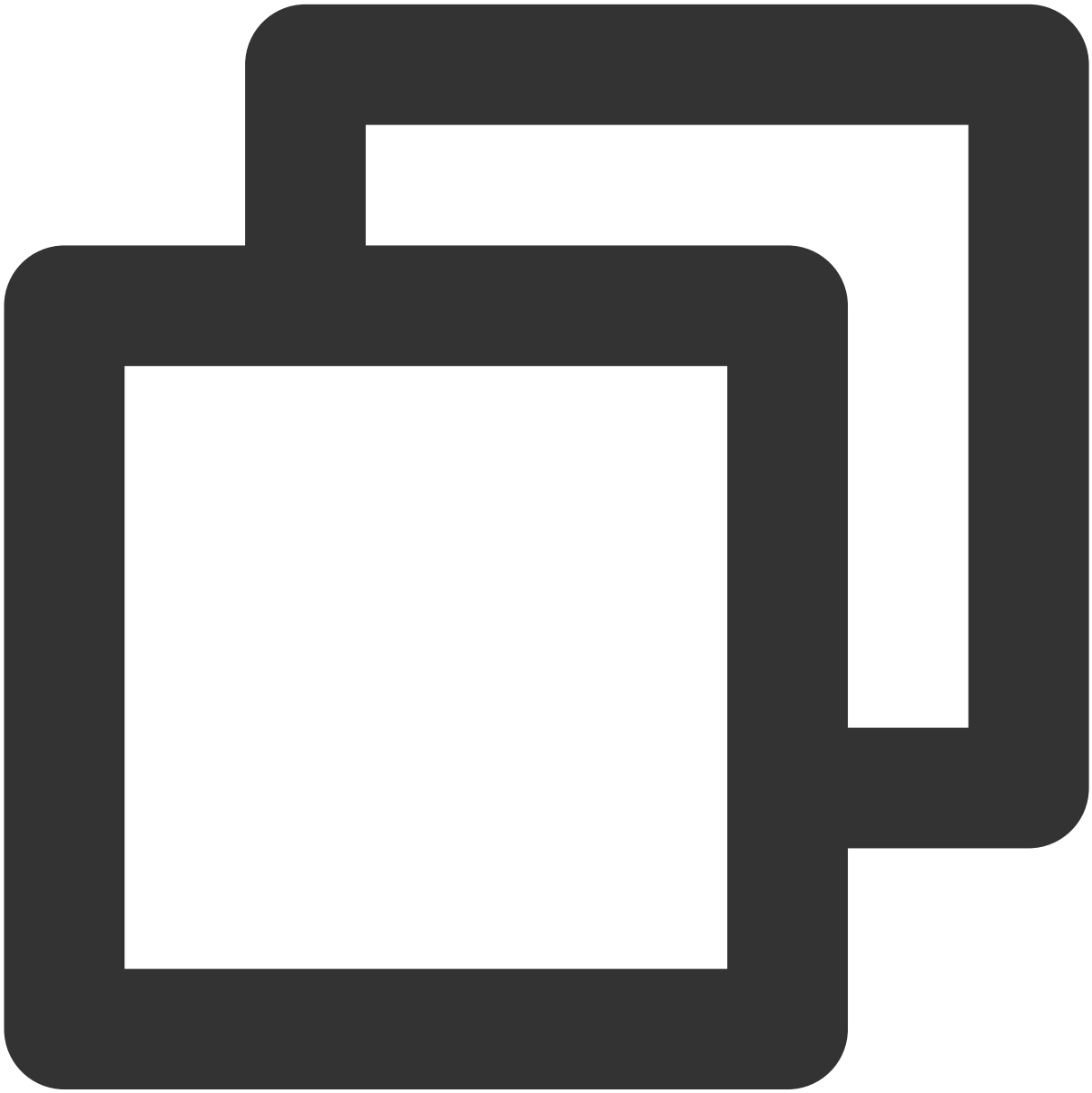
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

volume	uint32_t	音量の大きさ。

OnTestMicrophoneVolume

マイク音量の大きさのコールバック。



```
virtual void OnTestMicrophoneVolume(uint32_t volume) {}
```

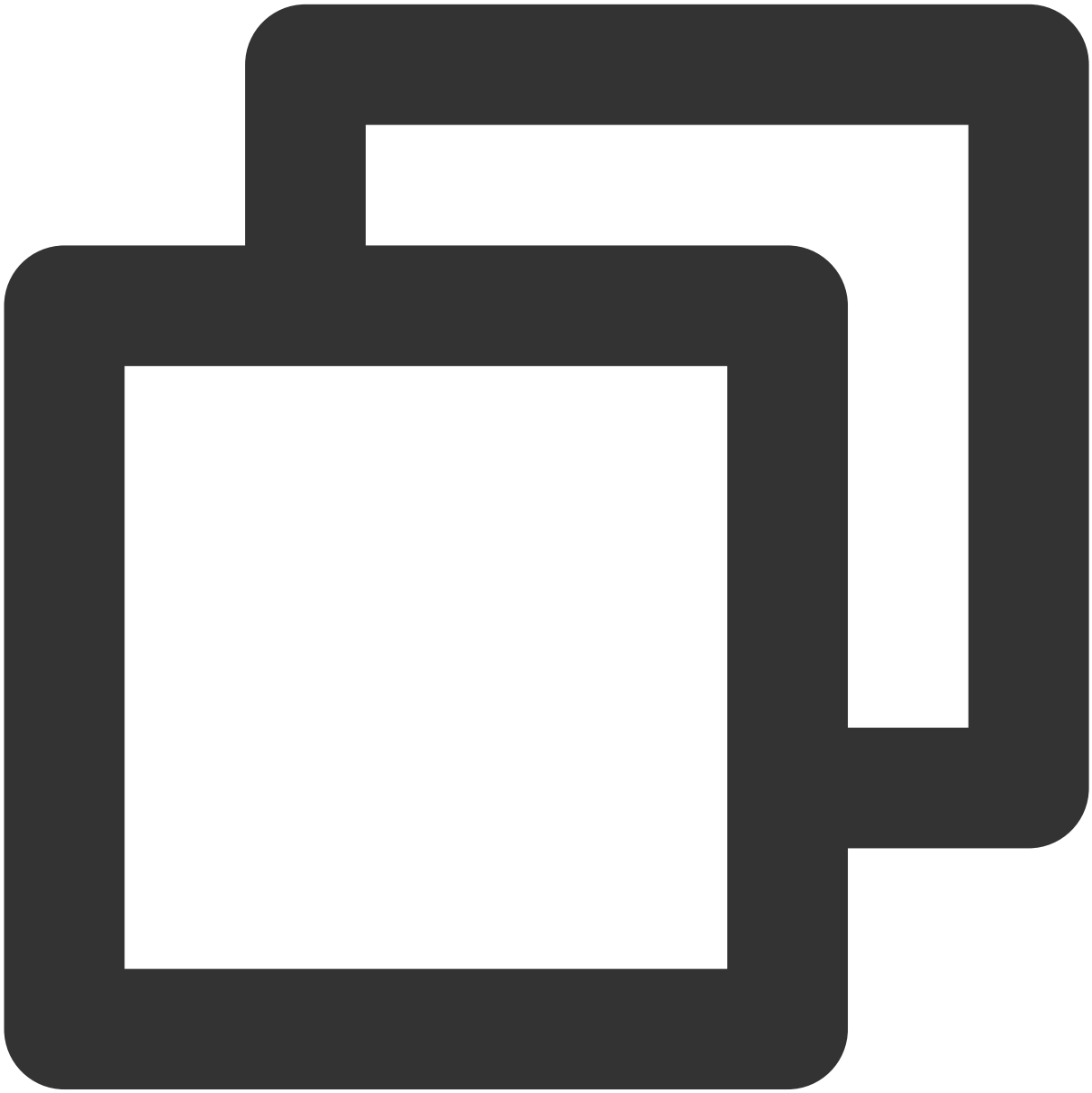
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

volume	uint32_t	音量の大きさ。

OnAudioDeviceCaptureVolumeChanged

システムキャプチャ音量調節のコールバック。



```
virtual void OnAudioDeviceCaptureVolumeChanged(uint32_t volume, bool muted) {}
```

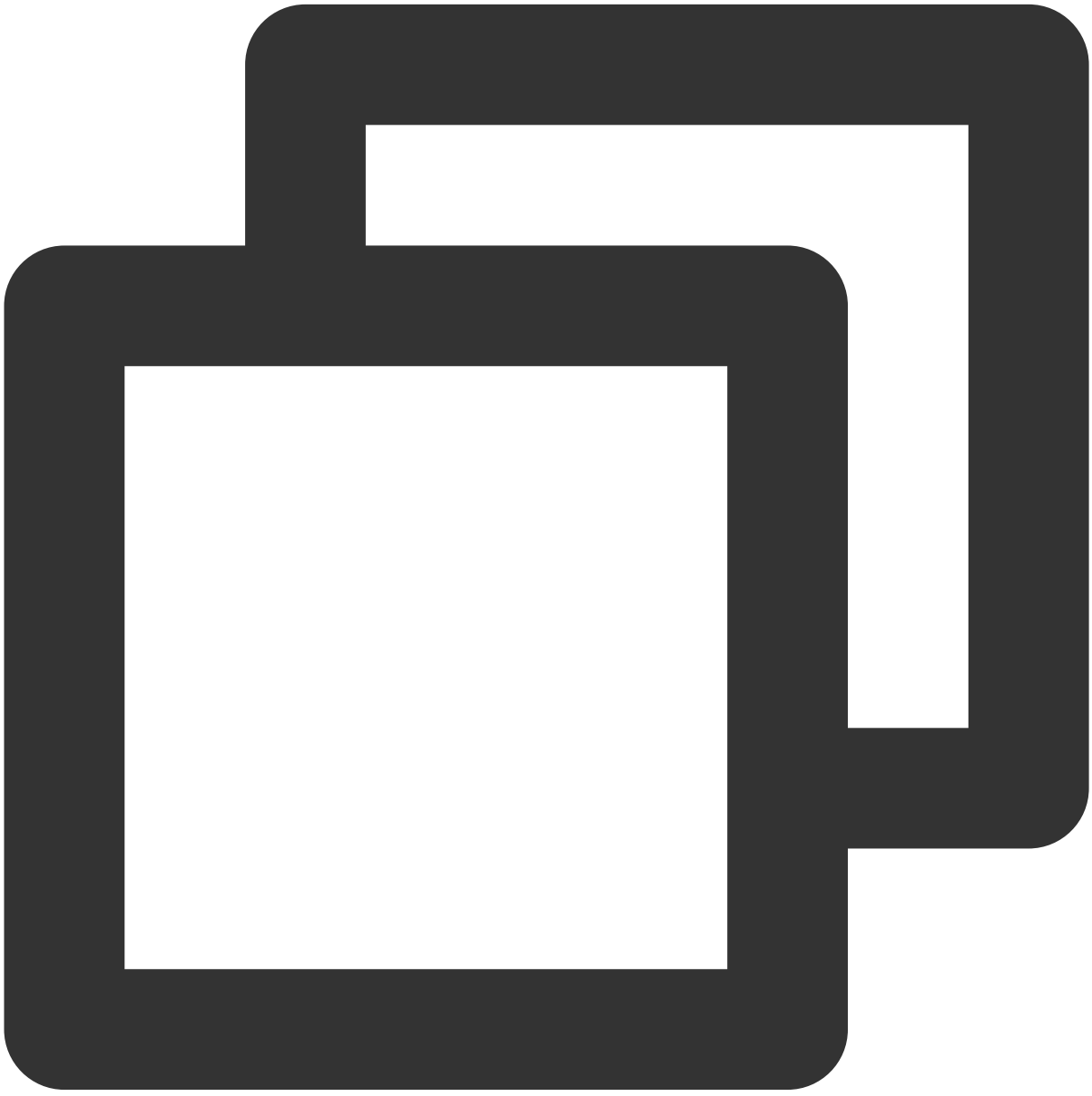
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

volume	uint32_t	音量の大きさ。
muted	bool	無効にされるかどうか

OnAudioDevicePlayoutVolumeChanged

システム再生音量調節のコールバック。



```
virtual void OnAudioDevicePlayoutVolumeChanged(uint32_t volume, bool muted) {}
```

パラメータは下表に示すとおりです。

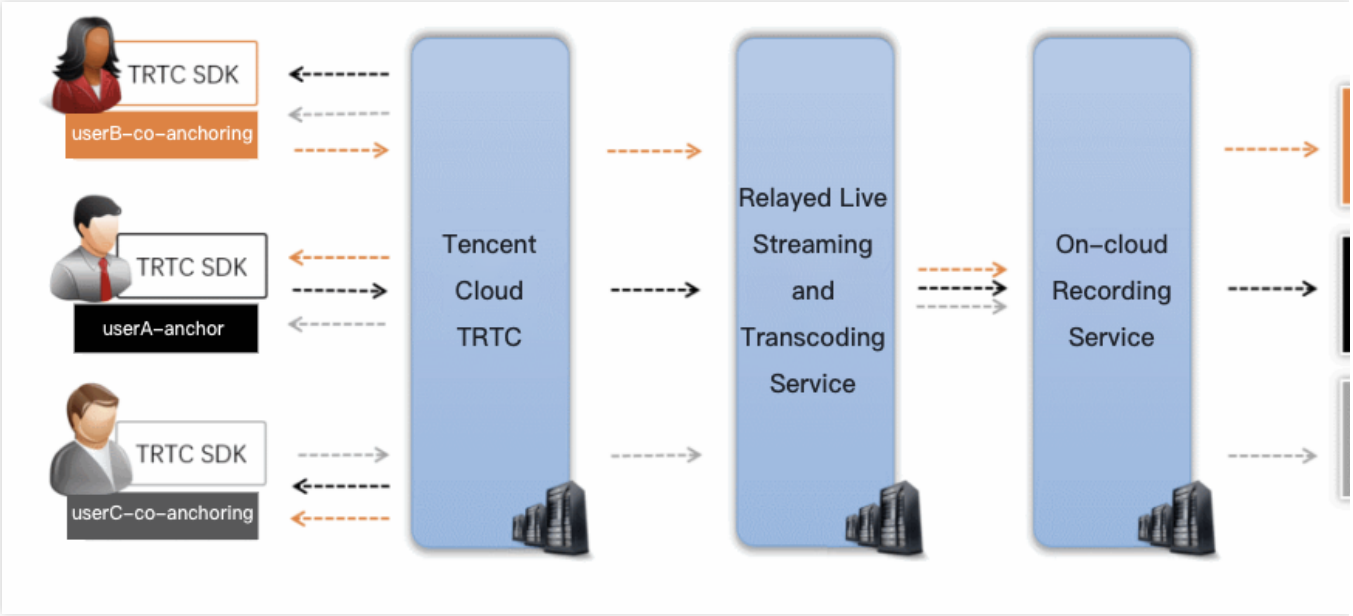
パラメータ	タイプ	意味
volume	uint32_t	音量の大きさ。
muted	bool	無効にされるかどうか

クラウドレコーディングと再生の実現（旧）

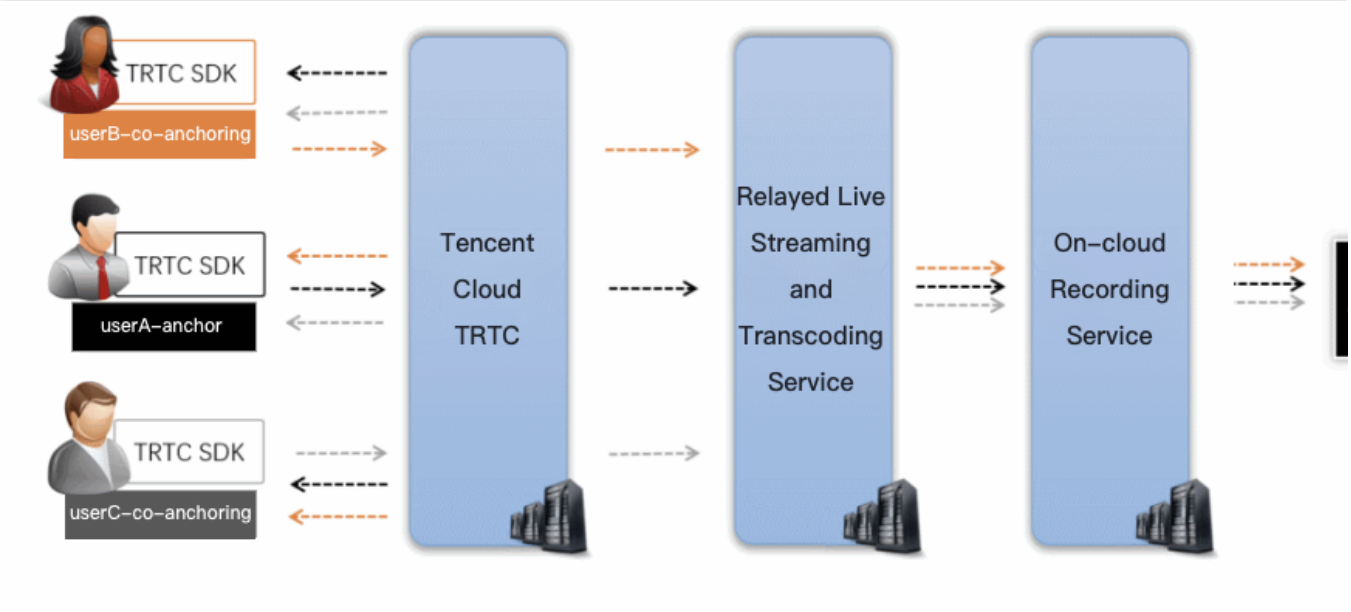
最終更新日：：2024-07-19 15:35:35

ユースケース

リモート教育、ステージライブストリーミング、ビデオミーティング、リモートでの損失評価、金融に関する録音・録画、オンライン医療などのユースケースでは、証明取得、品質検査、審査、アーカイブおよび再生などのニーズを考慮して、ビデオ通話やインタラクティブライブストリーミングプロセスの全体をレコーディングまたは保存する必要が度々出てきます。
TRTCのクラウドレコーディングは、ルームの各ユーザーのオーディオビデオストリーミングを独立した一つのファイルにレコーディングします。



ルーム内のマルチチャネルのオーディオビデオを、まずクラウドミクスストリーミングしてから、ミックス後のオーディオビデオストリーミングを一つのファイルにレコーディングします。



ご注意：

SDKAppIDが140で始まるユーザーは、このクラウド録画および再生操作ガイドを参照して統合および使用してください。アプリケーションのSDKAppIDが200で始まる場合は、新バージョンのクラウドレコーディングを参照して、クラウドレコーディング機能を起動してください。

コンソールガイド

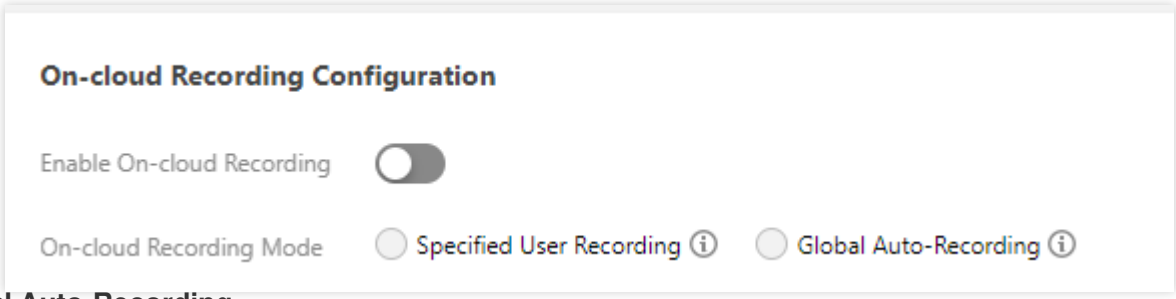
レコーディングサービスのアクティブ化

1. TRTCコンソールにログインして、左側ナビゲーションバーでアプリケーション管理を選択します。
2. ターゲットアプリケーションが所在する行の**機能設定**をクリックして、機能設定ページカードに進みます。アプリケーションを作成したことがない場合は、**アプリケーションの作成**をクリックしてアプリケーション名を入力し、**OK**をクリックして新しいアプリケーションを作成します。
3. **クラウドレコーディングの起動**右側の

をクリックすると、クラウドレコーディングの設定ページがポップアップで表示されます。

レコーディング形式の選択

TRTCのクラウドレコーディングサービスは、「Global Auto-Recording」および「指定ユーザーレコーディング」という2種類の異なるレコーディング形式を提供しています。



Global Auto-Recording

各TRTCルームの各ユーザーのアップストリームのオーディオとビデオは自動的にレコーディングされます。レコーディングタスクの開始と停止は自動的に行われるため、心配する必要はありません。操作は比較的シンプルで使いやすいです。具体的な使用方法については、方法1：Global Auto-Recordingをご参照ください。

指定ユーザーレコーディング

一部ユーザーのオーディオビデオストリーミングのみをレコーディングするように指定できます。これには、クライアントのSDKAPIまたはサーバーのRESTAPIを介して制御する必要があり、追加の開発作業が必要になります。具体的な使用方法については、方法2：指定ユーザーレコーディング（SDK API）および方法3：指定ユーザーレコーディング（REST API）をご参照ください。

ファイルのフォーマットの選択

クラウドレコーディングでは、HLS、MP4、FLV 、AACという4種類の異なるファイルフォーマットをサポートしています。4種類のフォーマットの違いと適用ケースを表形式でリストアップしているので、ご自分の業務のニーズに合わせて選択することができます。

パラメータ	パラメータの説明
ファイルタイプ	以下のファイルタイプをサポートしています。 HLS：このファイルタイプは、ほとんどのブラウザでオンライン再生をサポートしており、ビデオ再生シナリオに適しています。このファイルタイプを選択すると、ブレークポイント記録がサポートされ、単一ファイルの最大長が制限を受けません。 FLV：このファイルタイプはブラウザでのオンライン再生はサポートしていませんが、このフォーマットはシンプルでフォールトトレラント性に優れています。レコーディングファイルをVODプラットフォームに保存する必要がない場合、このファイルタイプを選択すると、レコーディングが完了したら、すぐにレコーディングファイルをダウンロードしてソースファイルを削除することができます。 MP4：このファイルタイプはWebブラウザでのオンライン再生をサポートしていますが、このフォーマットはフォールトトレラント性が低いです。ビデオ通話中は、どのようなパケットロスであっても、最終ファイルの再生品質に影響を与えます。 AAC：オーディオのみをレコーディングする必要がある場合、このファイルタイプを選択できます。
単一ファイルの最長時間（分）	実際の業務上のニーズに応じて、単一のビデオファイルの継続時間の最大制限を設定します。時間制限を超えると、ビデオファイルは自動的に分割されます。単位は分、値の範囲は1～120です。

	ファイルタイプがHLSに設定されている場合、単一ファイルの最大長に制限はないため、このパラメータは無効となります。
ファイルの保存期間（日）	実際の業務ニーズに応じて、ビデオファイルをVODプラットフォームに保存する日数を設定します。日数単位で数値範囲は0～1500です。期限後は、ファイルはVODプラットフォームによって自動的に削除され、復元できなくなります。0は永久保存を表します。
連続レコーディングタイムアウト時間（秒）	デフォルトでは、ネットワークの変動やその他の原因で通話（またはライブストリーミング）が中断した場合、レコーディングファイルは複数のファイルに分割されます。 「1回の通話（またはライブストリーミング）で1つしか再生リンクが生成されない」ようにする必要がある場合、実際の状況に応じて、連続レコーディングのタイムアウト時間を設定できます。中断間隔が設定された連続レコーディングタイムアウト時間を超えない場合、1回の通話（またはライブストリーミング）に対して1ファイルのみ生成されます。ただし、レコーディングファイルを受信する前に、連続レコーディング時間がタイムアウトするのを待つ必要があります。 単位は秒、値の範囲は1～1800で、0はブレークポイント以降、レコーディングを継続しないことを意味します。

説明：
HLSは最長で30分間の連続レコーディングをサポートしており、「1講義に1つしか再生リンクが生成されないように」することができます。また、ほとんどのブラウザのオンライン視聴をサポートしており、eラーニングにおけるビデオ再生のユースケースに大変適しています。

保存場所の選択

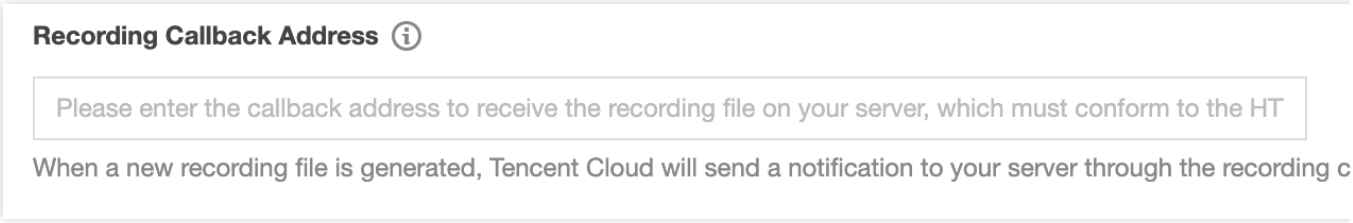
TRTCクラウドレコーディングファイルは、デフォルトではTencent Cloud VODサーバーに保存されます。プロジェクトで多くの業務が一つのTencent Cloud VODアカウントを共有している場合、レコーディングファイルの隔離が必要になることがあります。Tencent Cloud VODの「サブアプリ」機能によって、TRTCのレコーディングファイルをその他の業務エリアと切り離すことができます。

VODのメインアプリとサブアプリとは何ですか。
メインアプリとサブアプリとは、VODの1種類のリソース分割方式です。メインアプリはVODのメインアカウントに相当し、サブアプリは複数作成できます。各サブアプリは、ルートアカウント下の1個のサブアカウントに相当し、独立したリソース管理を備えています。ストレージ領域では、その他のサブアプリと相互に隔離することができます。

VODサービスのサブアプリケーションを有効にする方法とは。
ドキュメント「VODサブアプリケーションを有効にする方法」にもとづき、新規サブアプリケーションを追加することができます。このステップではVODコンソールに移動して操作する必要があります。

レコーディングコールバックの設定

レコーディングコールバックアドレス：
新しいファイルのランディング通知をリアルタイムで受信する必要がある場合は、サーバーがレコーディングファイルのコールバックを受け取るアドレスをここに入力できます。このアドレスは、HTTP（またはHTTPS）プロトコルに適合する必要があります。新しいレコーディングファイルが生成されたら、Tencent Cloudはこのアドレスを介して、サーバーに通知を送信します。



レコーディングコールバックキー：
コールバックキーは、コールバックイベントの受信時に署名認証を生成するために使用します。このキーはアルファベットの大文字・小文字と数字で構成し、32文字を超えないようにしてください。使用については、レコーディングイベントパラメータの説明をご参照ください。
説明：
詳細なレコーディングコールバックの受信と解読方法については、ドキュメント後半部分のレコーディングファイルの受け取りをご参照ください。

レコーディングの制御方法

TRTCでは、Global Auto-Recording、指定ユーザーレコーディング（SDK APIによる制御）、指定ユーザーレコーディング（REST APIによる制御）という3種類のクラウドレコーディング制御方法を提供しています。これらの方法を一つずつ、詳しくご紹介します。
コンソールでこの方式の使用設定をする方法
レコーディングタスクを開始する方法
レコーディングタスクを終了する方法
ルームのマルチチャネル画面を1チャネルにミックスする方法
レコーディングしたファイルの命名方法
ほかにどのような方法でプラットフォームをサポートするのか

方法1：Global Auto-Recording

コンソールでの設定
このレコーディング方法を使用したい場合は、コンソールのレコーディング形式の選択で「Global Auto-Recording」に設定してください。
レコーディングタスクの開始
TRTCルームの各ユーザーのオーディオ・ビデオストリーミングは、すべてファイルに自動レコーディングされる

ので、追加の操作は必要ありません。

レコーディングタスクの終了

自動停止。各キャスターがオーディオビデオのアップストリームを停止すると、このキャスターのクラウドレコーディングは自動停止します。ファイルフォーマットの選択のときに「連続レコーディング時間」を設定すると、レコーディングファイルを受信する前に、連続レコーディング時間がタイムアウトするのを待つ必要が出てきます。

マルチ画面のミックス

Global Auto-Recordingモードでのクラウドミクスストリーミングには、「サーバーREST API方式」と「クライアントSDK API方式」という2つの方式があります。この2つの方式を混合して使用しないでください。

サーバーREST APIミクスストリーミング方式：お客様のサーバーからAPI呼び出しを起動する必要があります。クライアントのプラットフォームバージョンの制限は受けません。

クライアントSDK APIミクスストリーミング方式：直接クライアントからミクスストリーミングを開始できます。現在サポートしているのは、iOS、Android、Windows、Mac、Webなどのプラットフォームであり、WeChat Mini Programは現在サポートしていません。

レコーディングファイルの命名

キャスターが入室時にuserDefineRecordIdパラメータを指定した場合、レコーディングファイルは `userDefineRecordId_streamType_開始時間_終了時間` で命名されます（streamTypeにはmainおよびauxの2つの値があり、mainはメインストリームを、auxはサブストリームを表します。サブストリームは通常、画面共有に利用されます）。

キャスターが入室時にuserDefineRecordIdパラメータを指定していないが、streamIdパラメータを指定している場合は、レコーディングファイルは `streamId_開始時間_終了時間` で命名されます。

キャスターが入室時にuserDefineRecordIdパラメータも、streamIdパラメータも指定していない場合、レコーディングファイルは、 `sdkappid_roomid_userid_streamType_開始時間_終了時間` で命名されます（streamTypeにはmainおよびauxの2つの値があり、mainはメインストリームを、auxはサブストリームを表します。サブストリームは通常、画面共有に利用されます）。

サポート済みのプラットフォーム

お客様側のサーバーで制御し、クライアント側プラットフォームの制限は受けません。

方法2：指定ユーザーレコーディング（SDK API）

TRTC SDKが提供するいくつかのAPIインターフェースとパラメータを呼び出すことで、クラウドミクスストリーミング、クラウドレコーディング、Relayed live streamingの3つの機能を実装できます。

クラウド機能	どのようにして開始しますか。	どのようにして停止しますか。
クラウドレコーディング	入室時にパラメータ `TRTCParams` 内の `userDefineRecordId` フィールドを指定します	キャスターの退室時に自動停止します
クラウドミ	SDK APIsetMixTranscodingConfig()を呼	ミクスストリーミングのキャスターの退室を開

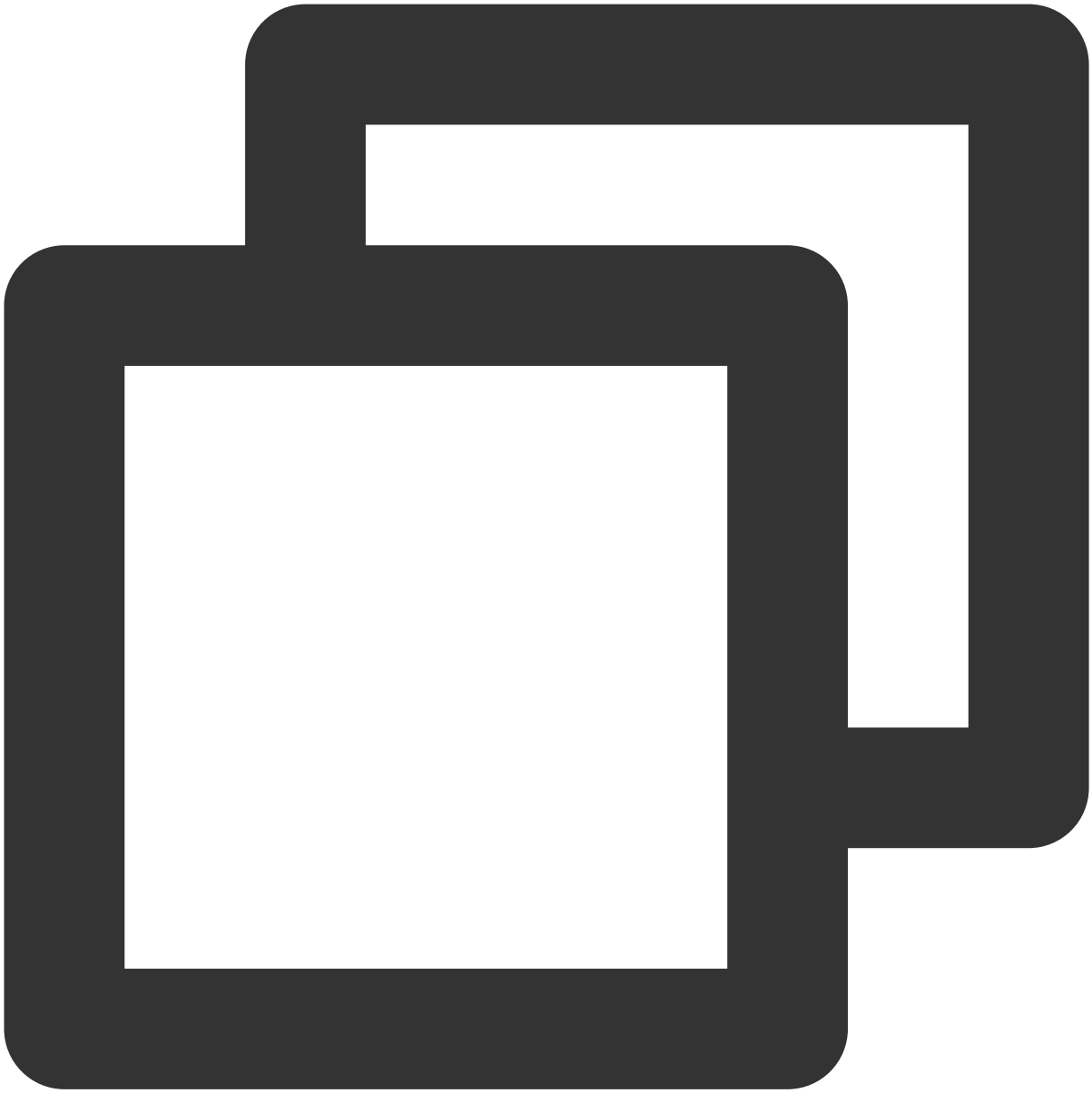
クスストリーミング	び出して、クラウドミクスストリーミングを開始します。	始すると、ミクスストリーミングは自動停止するか、途中でsetMixTranscodingConfig()を呼び出し、パラメータを `null/nil` に設定して手動で停止します
Relayed live streaming	入室時にパラメータ `TRTCParams` の `streamId` フィールドを指定します	キャスターの退室時に自動停止します

コンソールでの設定

このレコーディング方法を使用したい場合は、コンソールのレコーディング形式の選択で「ユーザーレコーディングの指定」に設定してください。

レコーディングタスクの開始

キャスターが入室時に入室パラメータ TRTCParams の userDefineRecordId フィールドを指定すると、その後、同キャスターのアップストリームのオーディオ・ビデオデータがクラウドレコーディングされます。このパラメータを指定していないキャスターはレコーディングタスクを起動することはありません。



```
// サンプルコード：レコーディングユーザーrexchangのオーディオビデオストリーミングを指定し、ファイ
TRTCCloud *trtcCloud = [TRTCCloud sharedInstance];
TRTCParams *param = [[TRTCParams alloc] init];
param.sdkAppId = 1400000123;      // TRTCのSDKAppIDは、アプリケーション作成後に取得できます
param.roomId   = 1001;            // ルームナンバー
param.userId   = @"rexchang";     // ユーザー名
param.userSig  = @"xxxxxxxx";     // ログイン署名
param.role     = TRTCRoleAnchor; // ロール：キャスター
param.userDefineRecordId = @"1001_rexchang";  // レコーディングID。そのユーザーのレコーデ
[trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE]; // LIVEモードを使用してください
```

レコーディングタスクの終了

自動停止。入室時にuserDefineRecordIdパラメータのキャスターを指定してオーディオビデオのアップストリームを停止すると、クラウドレコーディングは自動停止します。ファイルフォーマットの選択のときに「連続レコーディング時間」を設定すると、レコーディングファイルを受信する前に、連続レコーディング時間がタイムアウトするのを待つ必要が出てきます。

マルチ画面のミックス

SDK APIsetMixTranscodingConfig()を呼び出すことによって、ルーム内の他のユーザーの画面や音声をカレントユーザーのチャネルのオーディオビデオストリーミングにミックスすることができます。この部分の詳細については、次のドキュメントCloud MixTranscodingをご参照ください。

ご注意：

1つのTRTCルーム内では、1人のキャスター（配信を開始したキャスターを推奨）のみが `setMixTranscodingConfig` を呼び出せます。複数のキャスターが呼び出すと、ステータスの混乱を生じて、エラーが起きる可能性があります。

レコーディングファイルの命名

レコーディングファイルは、 `userDefineRecordId_開始時間_終了時間` のフォーマットで命名されます。

サポート済みのプラットフォーム

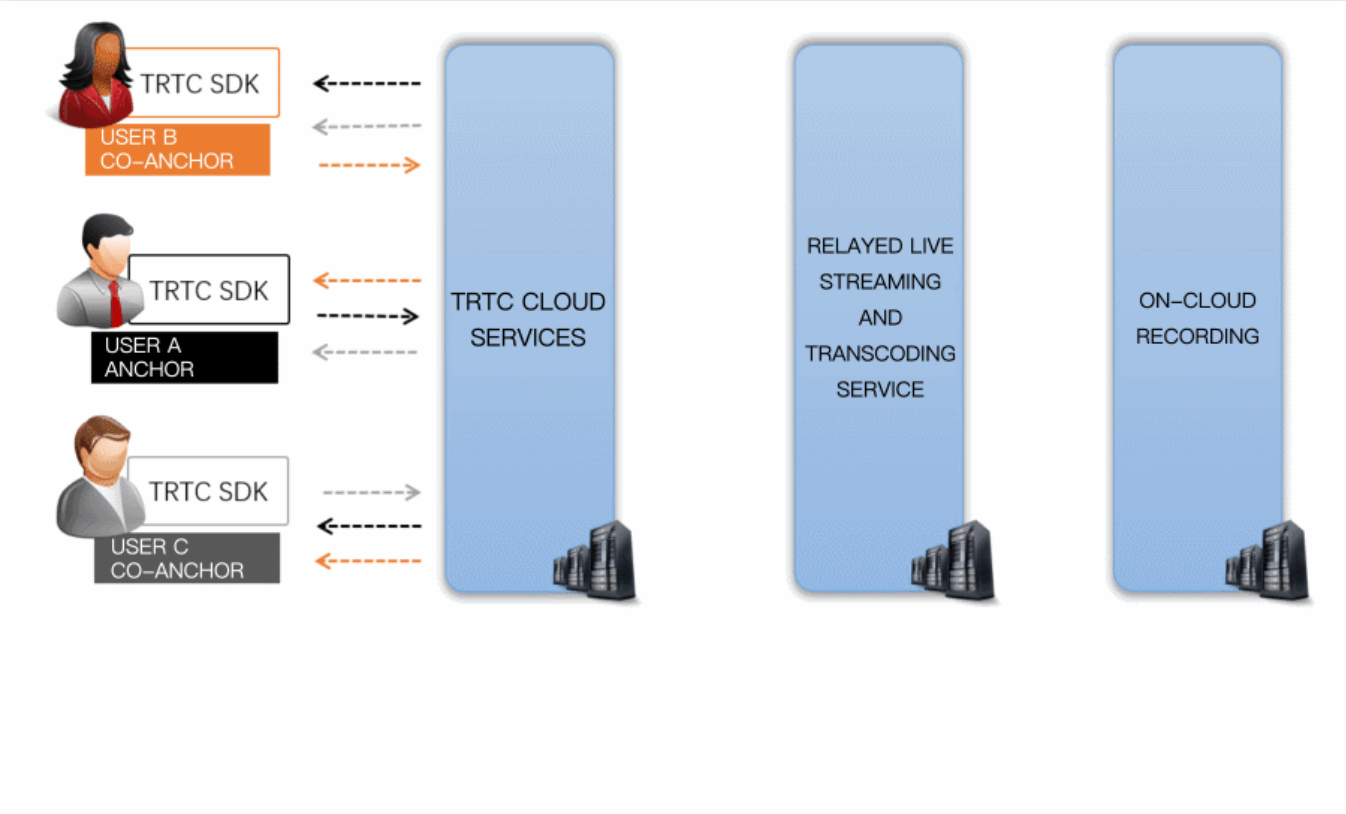
iOS、Android、Windows、Mac、Electron、Webなどの端末からのレコーディング制御の開始をサポートしていますが、現時点ではWeChat Mini Programからの制御はサポートしていません。

方法3：指定ユーザーレコーディング（REST API）

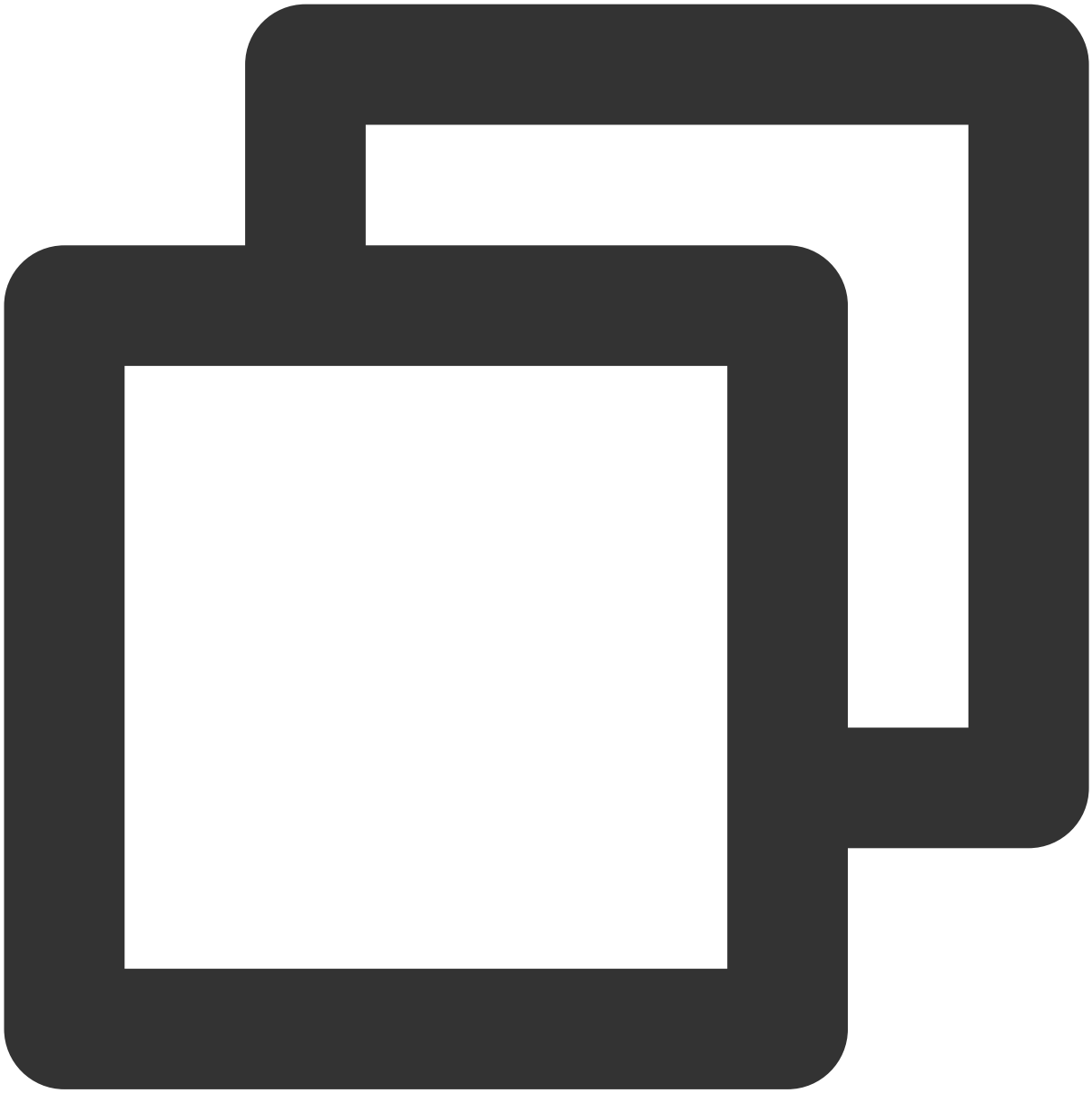
TRTCのサーバーはREST APIのペア（StartMCUMixTranscodeおよびStopMCUMixTranscode）を提供し、クラウドミクスストリーミング、クラウドレコーディング、Relayed live streamingという3つの機能を実装するために使用します。

クラウド機能	どのようにして開始しますか。	どのようにして停止しますか。
クラウドレコーディング	StartMCUMixTranscodeを呼び出して `OutputParams.RecordId` パラメータを指定すると、レコーディングを開始することができます	自動停止。途中でStopMCUMixTranscodeを呼び出して停止することができます
クラウドミクスストリーミング	StartMCUMixTranscodeを呼び出して `LayoutParams` パラメータを指定すると、テンプレートとレイアウトパラメータを設定することができます	すべてのユーザーが入室した後に自動停止します。途中でStopMCUMixTranscodeを呼び出して手動で停止することができます
Relayed live streaming	StartMCUMixTranscodeを呼び出して `OutputParams.StreamId` パラメータを指定すると、CDNのRelayed live streamingを起動することができます	自動停止。途中でStopMCUMixTranscodeを呼び出して停止することができます

説明：
これらのREST APIに対する制御は、TRTCクラウドサービスのコアとなるミクスストリーミングモジュール(MCU)であり、MCUミクスストリーミング後の結果はレコーディングシステムとライブCDNに送信されるため、APIの名前は `Start/StopMCUMixTranscode` と呼ばれます。したがって機能面から言うと、 `Start/StopMCUMixTranscode` はミクスストリーミングの機能を実装できるだけでなく、クラウドレコーディングとRelayed live streaming CDNの機能も実装できます。



コンソールでの設定
このレコーディング方法を使用したい場合は、コンソールのレコーディング形式の選択で「ユーザーレコーディングの指定」に設定してください。
レコーディングタスクの開始
サーバーから StartMCUMixTranscode を呼び出して、 `OutputParams.RecordId` パラメータを指定すると、ミクスストリーミングおよびレコーディングが実行できます。



```
// コードサンプル：REST APIによってクラウドミクスストリーミングおよびクラウドレコーディングのタス
https://trtc.tencentcloudapi.com/?Action=StartMCUMixTranscode
&SdkAppId=1400000123
&RoomId=1001
&OutputParams.RecordId=1400000123_room1001
&OutputParams.RecordAudioOnly=0
&EncodeParams.VideoWidth=1280
&EncodeParams.VideoHeight=720
&EncodeParams.VideoBitrate=1560
&EncodeParams.VideoFramerate=15
&EncodeParams.VideoGop=3
```

```
&EncodeParams.BackgroundColor=0
&EncodeParams.AudioSampleRate=48000
&EncodeParams.AudioBitrate=64
&EncodeParams.AudioChannels=2
&LayoutParams.Template=1
&<パブリックリクエストパラメータ>
```

ご注意：

このREST APIは、ルーム内の少なくとも1人のユーザーが入室（enterRoom）に成功した場合に有効になります。REST APIを使用すると、シングルストリームのレコーディングはサポートされません。シングルストリームをレコーディングしたい場合は、方法1または方法2を選択してください。

レコーディングタスクの終了

自動停止では、途中で StopMCUMixTranscode を呼び出して、ミクスストリーミングおよびレコーディングのタスクを停止することもできます。

マルチ画面のミックス

StartMCUMixTranscodeを呼び出す時に、同時に `LayoutParams` パラメータを指定すれば、クラウドミクスストリーミングを実装できます。このAPIは、ライブストリーミング全時間内での複数回呼び出しをサポートしていますので、必要に応じて `LayoutParams` パラメータを修正し、このAPIを再度呼び出して合成画面のレイアウトを調整することができます。ただし、注意が必要なのは、パラメータ `OutputParams.RecordId` と `OutputParams.StreamId` が複数回呼び出しの中で一致性を維持する必要があることです。維持しない場合、ストリームが切断され、複数のレコーディングファイルが生成されてしまいます。

説明：

クラウドミクスストリーミングのより詳しい紹介については、Cloud MixTranscodingをご参照ください。

レコーディングファイルの命名

レコーディングファイルは、 StartMCUMixTranscode を呼び出すときに指定した `OutputParams.RecordId` パラメータで命名します。命名フォーマットは、 `OutputParams.RecordId_開始時間_終了時間` です。

サポート済みのプラットフォーム

お客様側のサーバーで制御し、クライアント側プラットフォームの制限は受けません。

レコーディングファイルの検索

レコーディング機能を起動させた後、TRTCシステムからレコーディングしたファイルは、Tencent CloudのVODサーバーから探し出せます。VODコンソールから手動で直接探し出したり、バックエンドサーバーからREST APIを使用したりして所定の時間にフィルタリングを行うこともできます。

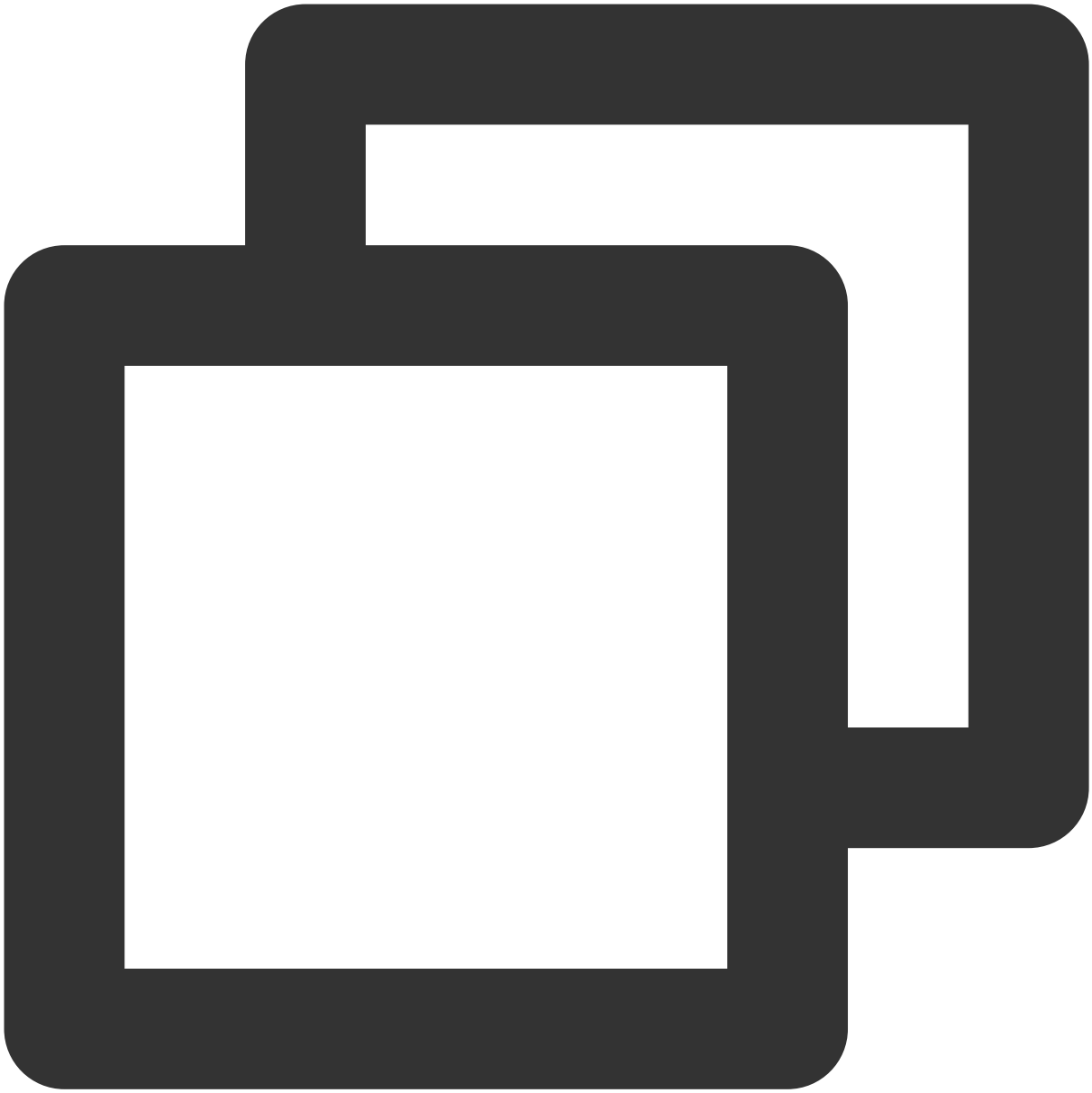
方法1：VODコンソールで手動で検索

1. VODコンソールにログインし、左側ナビゲーションバーで**メディア資産管理**を選択します。

2. リスト上方の**プレフィックス検索**をクリックして、**プレフィックス検索**を選択します。検索ボックスに、例えば `1400000123_1001_rexchang_main` といったキーワードを入力して検索ボタンをクリックすると、ビデオ名がプレフィックスと符合するビデオファイルを表示します。
3. 作成時間に基づき、必要なターゲットファイルをフィルタリングすることができます。

方法2：VOD REST APIによる検索

Tencent CloudのVODシステムは、一連のREST APIを提供してその上のオーディオビデオファイルを管理します。メディア情報検索のこのREST APIによって、VODシステム上のファイルを検索します。パラメータ表の `Text` パラメータをリクエストすることで、あいまい一致や、 `StreamId` パラメータをもとに正確な検索を行うこともできます。
RESTリクエスト例：



```
https://vod.tencentcloudapi.com/?Action=SearchMedia
&StreamId=stream1001
&Sort.Field=CreateTime
&Sort.Order=Desc
&<パブリックリクエストパラメータ>
```

レコーディングファイルの受け取り

レコーディングファイルの検索のほかにも、コールバックアドレスを設定することによって、Tencent Cloudに新たにレコーディングしたファイル情報をサーバーに自動的にプッシュさせることもできます。
ルームの最後の1チャネルのオーディオビデオストリーミングが終了すると、Tencent Cloudはレコーディングを終了し、ファイルをVODプラットフォームに転送して保存します。このプロセスには、デフォルトで約30秒から2分かかります（連続レコーディング時間を300秒に設定した場合、待ち時間はデフォルトの時間に300秒を加えた時間になります）。転送と保存が完了すると、Tencent Cloudは、レコーディングコールバック設定で設定したコールバックアドレス（HTTP/HTTPS）を介してサーバーに通知を送信します。
Tencent Cloudは、レコーディングおよびレコーディング関連のイベントを、設定したコールバックアドレスを介して、サーバーにプッシュします。コールバックメッセージの例を次の図に示します。

下表のフィールドによって、現在のコールバックがどの通話（またはCSS）に対応するかを確定することができます。

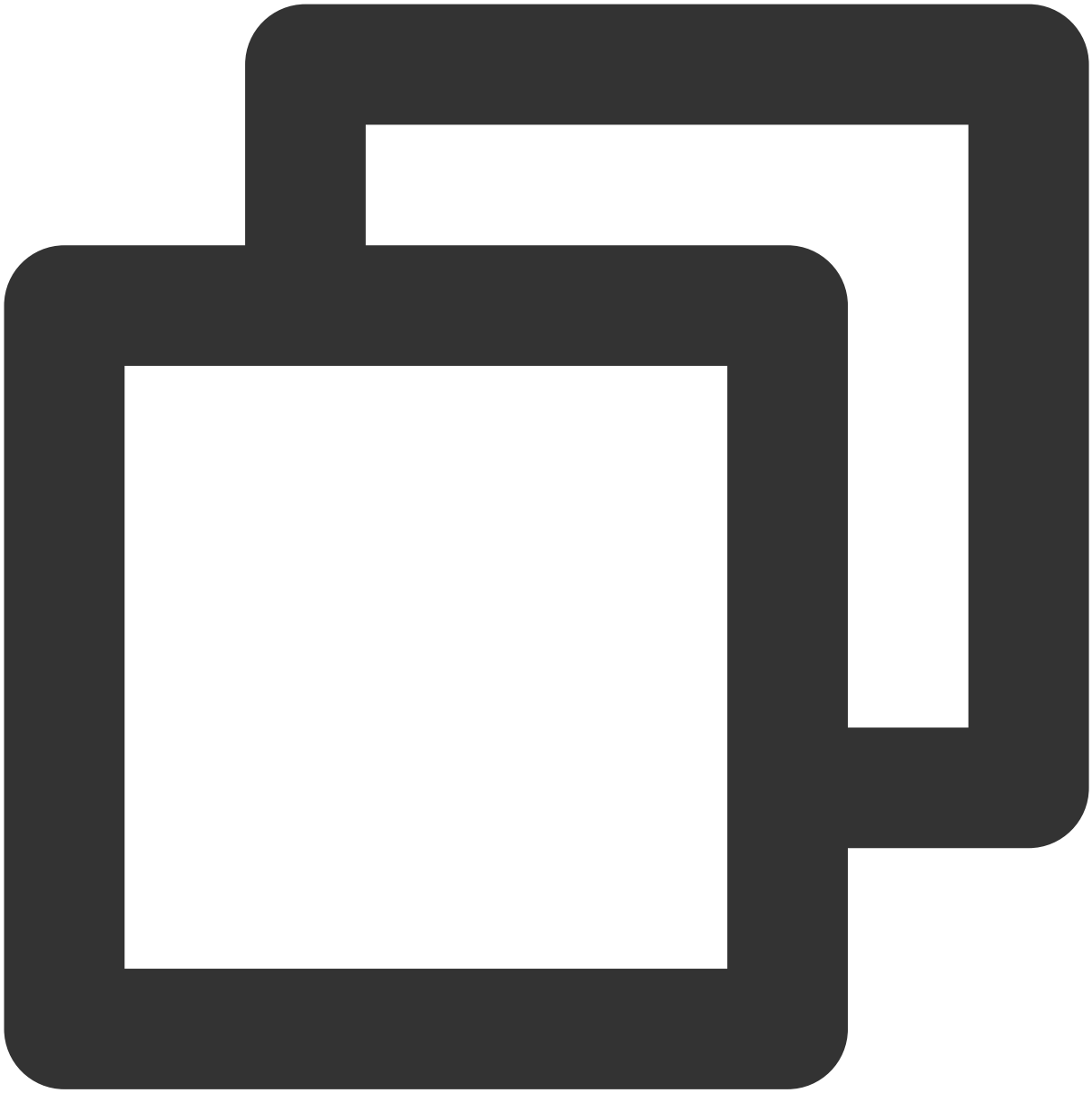
番号	フィールド名	説明
1	event_type	情報形式。event_typeが100のときは、このコールバックメッセージが、レコーディングファイルが生成したメッセージであることを表します。
2	stream_id	ライブCDNのstreamIdです。入室時にTRTCParamsのstreamIdフィールドを設定することで指定でき（推奨）、また、TRTCCloudのstartPublishingを呼び出すときにパラメータstreamIdで指定することもできます。
	stream_param.userid	ユーザー名のBase64エンコード。

3		
4	stream_param.userdefinerecordid	カスタムフィールド。TRTCParamsのuserDefineRecordIdフィールドを設定することで指定できます。</td>
5	video_url	レコーディングファイルの視聴アドレスは、VOD再生に使用できます。

説明：
コールバックフィールドに関するその他の説明については、CSS-レコーディングイベントの通知をご参照ください。

レコーディングファイルの削除

Tencent CloudのVODシステムは、一連のシステムREST APIを提供して、その上のオーディオビデオファイルを管理します。メディアAPIの削除によって任意の指定したファイルを削除することができます。
RESTリクエスト例：



```
https://vod.tencentcloudapi.com/?Action=DeleteMedia
&FileId=52858907988664150587
&<パブリックリクエストパラメータ>
```

レコーディングファイルの再生

eラーニングなどのユースケースでは、教育リソースを最大限に活用するために、通常、ライブストリーミングの終了後、レコーディングファイルを複数回再生する必要があります。

ファイルフォーマットの選択（HLS）

レコーディングフォーマットの設定では、ファイルフォーマットをHLSに選択します。
HLSは最長30分間のブレークポイントでの連続レコーディングをサポートしており、「1つのライブストリーミング（または1回の授業）で1つしか再生リンクが生成されないように」することができます。また、HLSファイルは、ほとんどのブラウザのオンライン再生をサポートしており、ビデオ再生のユースケースに大変適しています。

VODアドレス（video_url）の取得

レコーディングファイルの受け取りのときに、コールバック情報の**video_url**フィールドを取得できます。このフィールドは、現在のレコーディングファイルのTencent CloudにおけるVODアドレスになります。

VODプレーヤーとの接続

使用するプラットフォームに基づいてVODプレーヤーと接続します。具体的な操作は以下をご参照ください。
iOSプラットフォーム
Androidプラットフォーム
Webブラウザ
ご注意：
プロフェッショナル版のTRTC SDKの使用を推奨します。プロフェッショナル版にはSuper Player（Player+）やモバイルライブストリーミングなどの機能が統合され、基盤モジュールを高度に再利用することにより、統合したプロフェッショナル版の容量増加は同時に2つの独立したSDKを統合させた場合よりも小さくなります。さらにシンボル重複（symbol duplicate）の問題も回避できます。

関連料金

クラウドレコーディングと再生機能で使用できる機能には、クラウドサービスリソース（クラウドレコーディングサービス、VODの再生ファイルストレージおよび処理、VODの再生サービスなどを含む）、端末SDKのオンデマンドビデオ再生の機能が含まれます。実際のニーズに応じて次の料金が発生する可能性があります。

クラウドサービス消費

クラウドサービス消費には、クラウドレコーディングによって発生する料金とファイル再生によって発生する消費料金が含まれます。実際のニーズに応じて使用することができます。
説明
ここでの価格は一例であり、あくまでも参考です。価格と実際の状況とが一致しない場合は、クラウドレコーディング課金説明、CSS、VODの料金を基準とします。
レコーディング料金：トランスコーディングまたはカプセル化によって発生した料金の計算
レコーディングは、オーディオ・ビデオストリーミングのトランスコーディングまたはそのパッケージングが必要となるため、サーバーの計算リソースの消費が発生します。このため、レコーディング業務が計算リソースを占有するコストに応じて費用を徴収する必要があります。

ご注意：
2020年7月1日以降、初めてTRTCコンソールでアプリケーションを作成したTencent Cloudアカウントの場合、クラウドレコーディング機能を使用した後に発生するレコーディング料金については、クラウドレコーディング課金説明を基準とします。
2020年7月1日以前にTRTCコンソールでアプリケーションを作成したことのあるTencent Cloudアカウントの場合、2020年7月1日以前またはそれ以降に作成したアプリケーションであっても、クラウドレコーディング機能を使用して発生するレコーディング料金については、デフォルトで、**CSSレコーディング**の課金ルールを引き続き適用します。
CSSレコーディング課金の計算方法は、同時レコーディングのチャネル数に応じて料金を徴収し、同時実行数が多いほどレコーディング料金も高くなります。具体的な課金説明については、CSS > CSSレコーディングをご参照ください。
計算式：
レコーディング有効日数割合=1暦月におけるレコーディング機能を使用する日数/その月の総日数。
レコーディング料金=1暦月の同時レコーディングチャネル数ピーク値×レコーディング有効日数割合×レコーディングチャネル数単価。

例えば、4月には1000名のキャスターがいて、夕方のピーク時に、最多500チャネルのキャスターのオーディオビデオストリーミングを同時レコーディングする必要があると同時に、4月に計6日間、レコーディング機能を使用したとします（有効日数の割合は6/30）。仮にレコーディング単価が5.2941ドル/チャネル/月とすると、4月の合計レコーディング料金は、`500チャネル×5.2941米ドル/チャネル/月=2647.05米ドル/月` となります。
レコーディングフォーマットの設定をする時に同時に2種類のレコーディングファイルを選択した場合、レコーディング料金とストレージ料金はいずれも× 2になります。同様の理屈により、3種類のファイルを選択した時のレコーディング料金とストレージ料金は× 3になります。不要な場合は、必要な1種類のファイルフォーマットのみを選択することをお勧めします。コストを大幅に抑えることができます。
トランスコード料金：ミクスストリーミングレコーディングを有効にすると、この料金が発生します
ミクスストリーミングレコーディングを有効にすると、ミクスストリーミング自体にデコードとエンコードが必要なため、別途ミクスストリーミングトランスコード料金が発生します。ミクスストリーミングトランスコーディングは、解像度のサイズとトランスコーディング時間によって課金され、キャスターが使用する解像度が高く、マイク接続時間が長いほど（通常、マイク接続シナリオにのみミクスストリーミングトランスコーディングが必要）、料金が高くなります。具体的な料金の計算については、CSSトランスコードをご参照ください。
例えば、setVideoEncoderParam()を介してキャスターのビットレート（videoBitrate）を1500kbps、解像度を720Pに設定したとします。1名のキャスターが視聴者と1時間マイク接続し、マイク接続の間にクラウドミクスストリーミングを有効にした場合、発生するトランスコード料金は `0.0057米ドル/分×60分=0.342米ドル` となります。
再生ファイル保存料金：クラウドビデオストレージサービスによって発生する保存料金
レコーディングしたビデオファイルをVODサービスに保存すると、VODのクラウドストレージサービスを利用することになります。保存自体にディスクリソースの消費が発生しますので、保存するリソースの占用状況に応じて料金を徴収する必要があります。保存期間が長いほど料金も高くなるため、特殊なニーズがない場合は、ファイル

の保存期間を短めに設定して費用を節約するか、またはファイルを自分のサーバーに保存することもできます。保存料金は、ビデオ保存（日次決済）価格を選択し、日次決済で計算することができます。
例えば、setVideoEncoderParam()を介してキャスターのビットレート（videoBitrate）を1000kbpsに設定し、そのキャスターのライブストリーミングビデオ（1種類のファイルフォーマットを選択）をレコーディングし、1時間レコーディングすると、おおよそ `(1000/8)KBps×3600秒=450000KB=0.45GB` サイズのビデオファイルが生成されます。このファイルに毎日発生するおおよそのストレージ料金は、 `0.45GB×0.0009米ドル/GB/日=0.000405米ドル` となります。
再生視聴料金：VODビデオの配信と再生によって発生する再生料金
レコーディングしたビデオファイルを再生に用いる場合は、VODのCDN再生機能を利用することになります。視聴自体にCDNトラフィックが消費されるため、VODの価格に基づいて課金する必要があります。デフォルトは、トラフィックによる課金となっています。視聴者数が多いほど料金は高くなります。視聴料金はビデオアクセラレーション（日次決済）価格 を選択し、日次決済で計算できます。
例えば、クラウドレコーディングによって1GBのファイルが生成され、1,000名の視聴者が最初から最後まで完全にそのビデオを視聴すると、約1TBのVOD視聴トラフィックが発生します。この場合は、階層価格表にもとづき、1,000名の視聴で、 `1000×1GB×0.0794米ドル/GB=79.4米ドル` の料金が発生します。トラフィックパッケージの場合、24.5ドルです。
Tencent Cloudからファイルをお客様のサーバーにダウンロードするように選択した場合も、非常に小さなVODトラフィックが消費され、お客様の月次請求書の中に反映されます。

SDKの再生権限承認

Tencent Real-Time Communication（TRTC）の全機能版SDKは全面的な機能と高度なパフォーマンスが可能なビデオ再生機能をご提供し、VODと組み合わせることで簡単にビデオ再生機能を実装できます。モバイルSDKの10.1以降のバージョンでは、指定のLicenseを取得することでビデオ再生機能のロックを解除することができます。
ご注意：
TRTCの再生機能にはLicense承認は必要ありません。

Legacy Version (TUIVoiceRoom)
TUIVoiceRoom (Android)の統合

最終更新日：：2024-07-19 15:35:35

コンポーネントの説明

TUIVoiceRoomはオープンソースのオーディオビデオUIコンポーネントであり、プロジェクトにTUIVoiceRoomコンポーネントを統合することにより、数行のコードを書くだけで、Appに「多人数ボイスチャット」などのシーンを組み込むことができます。TUIVoiceRoomはiOSなどのプラットフォームをサポートしています。基本機能は下図のとおりです：

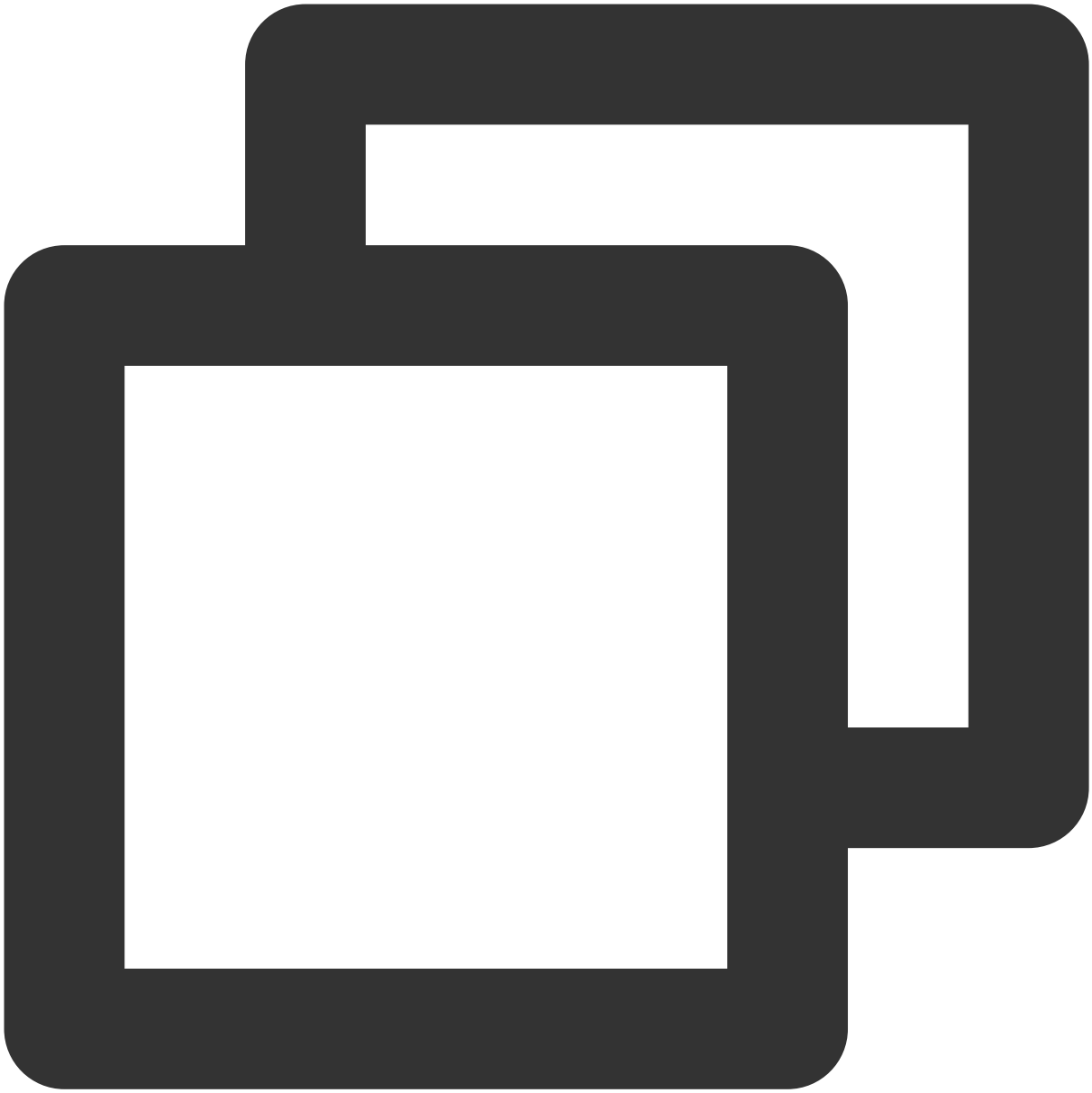
説明：

TUIKitシリーズコンポーネントはTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期的にアクティブにすることができます。IMサービスの課金ルールの詳細については、Instant Messagingの料金説明をご参照ください。TRTCをアクティブにすると、デフォルトでは、100DAUまでサポートするIM SDK体験版もアクティブになります。

コンポーネントの統合

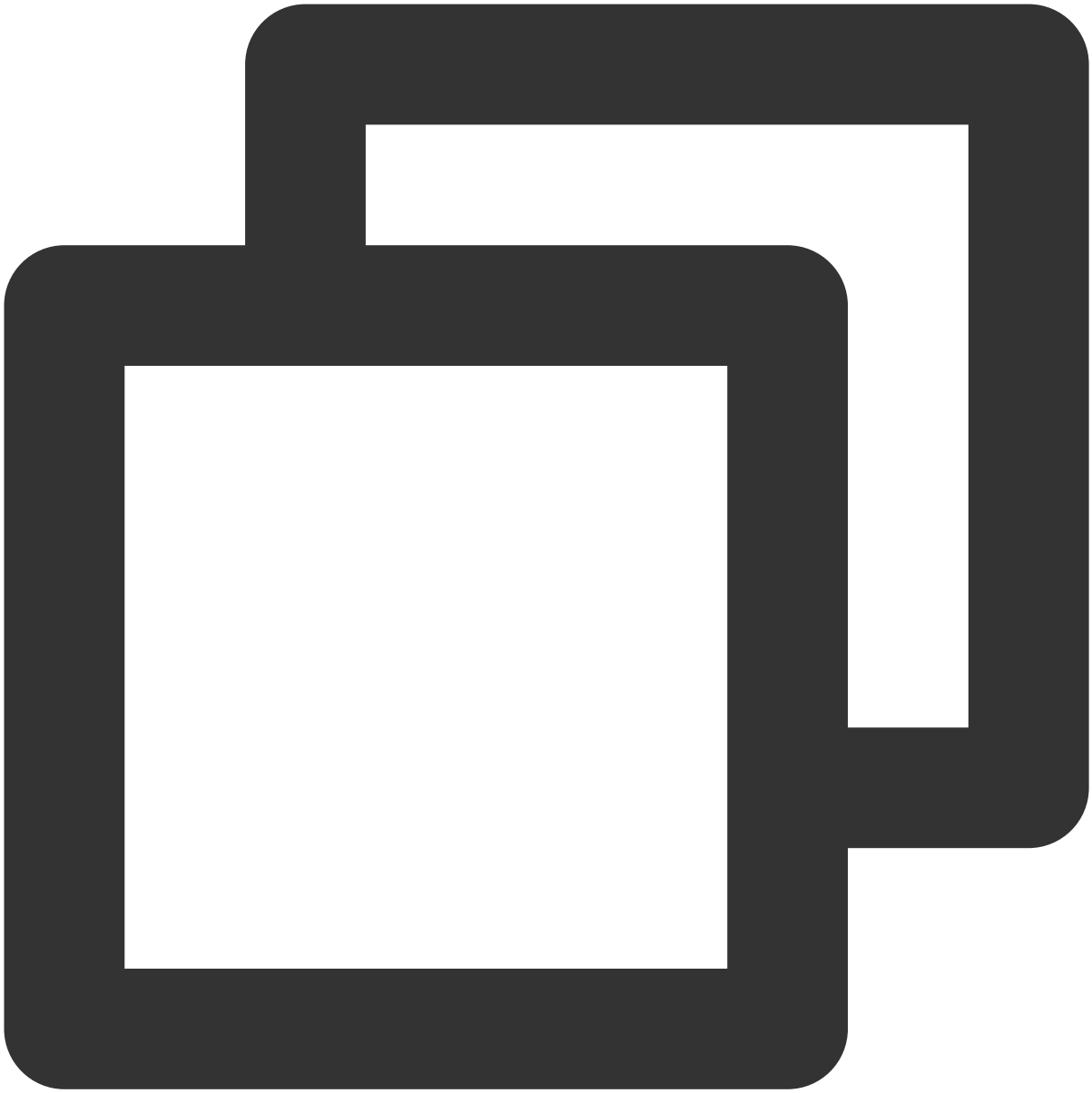
ステップ1：TUIVoiceRoomコンポーネントのダウンロードとインポート

クリックしてGithubに進み、コードのクローン/ダウンロードを選択した後、Android/Sourceディレクトリをプロジェクトにコピーし、次のようにインポート動作を完了します：
`setting.gradle` へのインポートを完了します。以下をご参照ください：



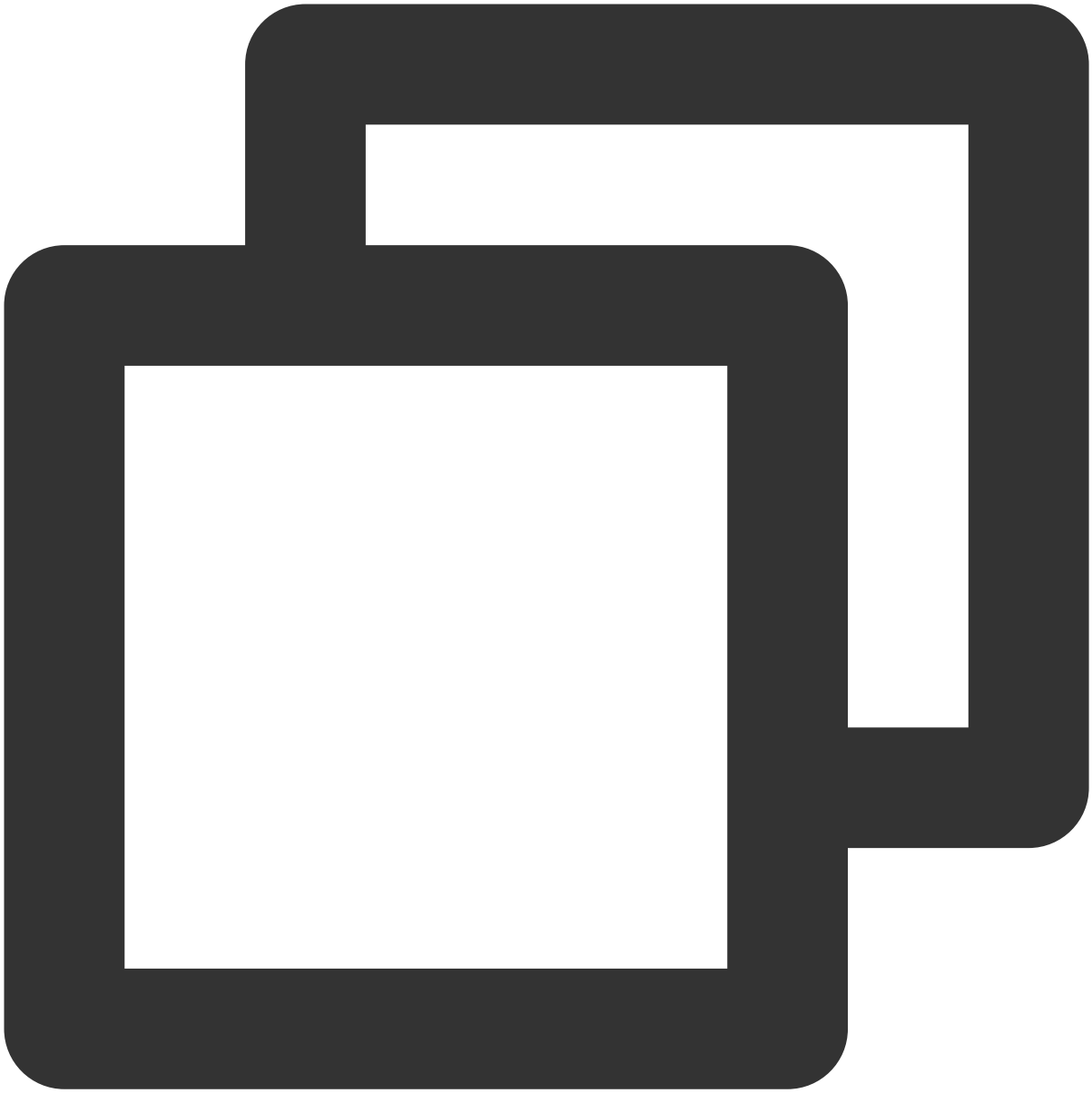
```
include ':Source'
```

appのbuild.gradleファイルにSourceに対する依存関係を追加します：



```
api project(':Source')
```

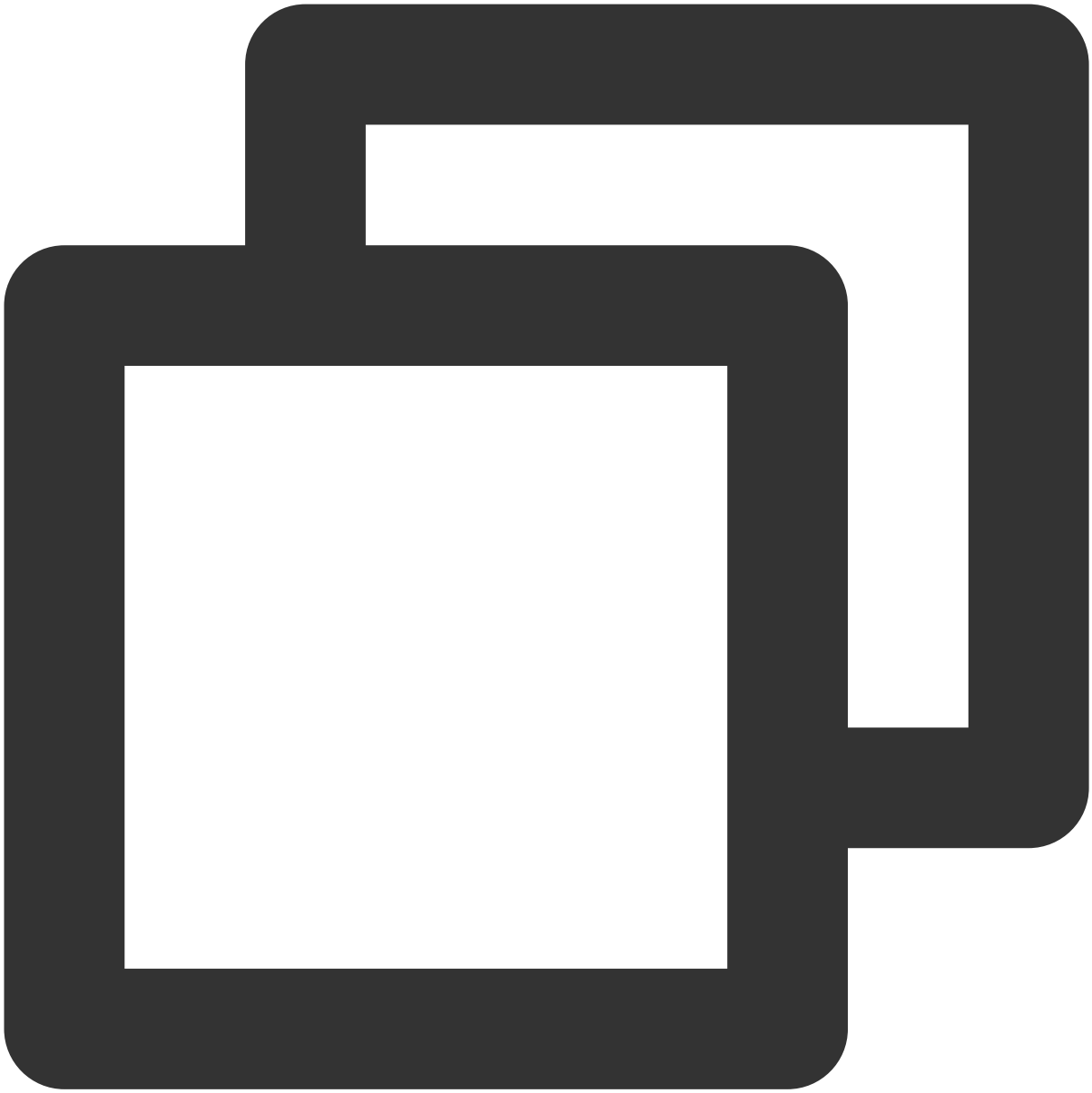
ルートディレクトリの `build.gradle` ファイルに `TRTC SDK` および `IM SDK` の依存関係を追加します



```
ext {
  liteavSdk = "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
  imSdk = "com.tencent.imsdk:imsdk-plus:latest.release"
}
```

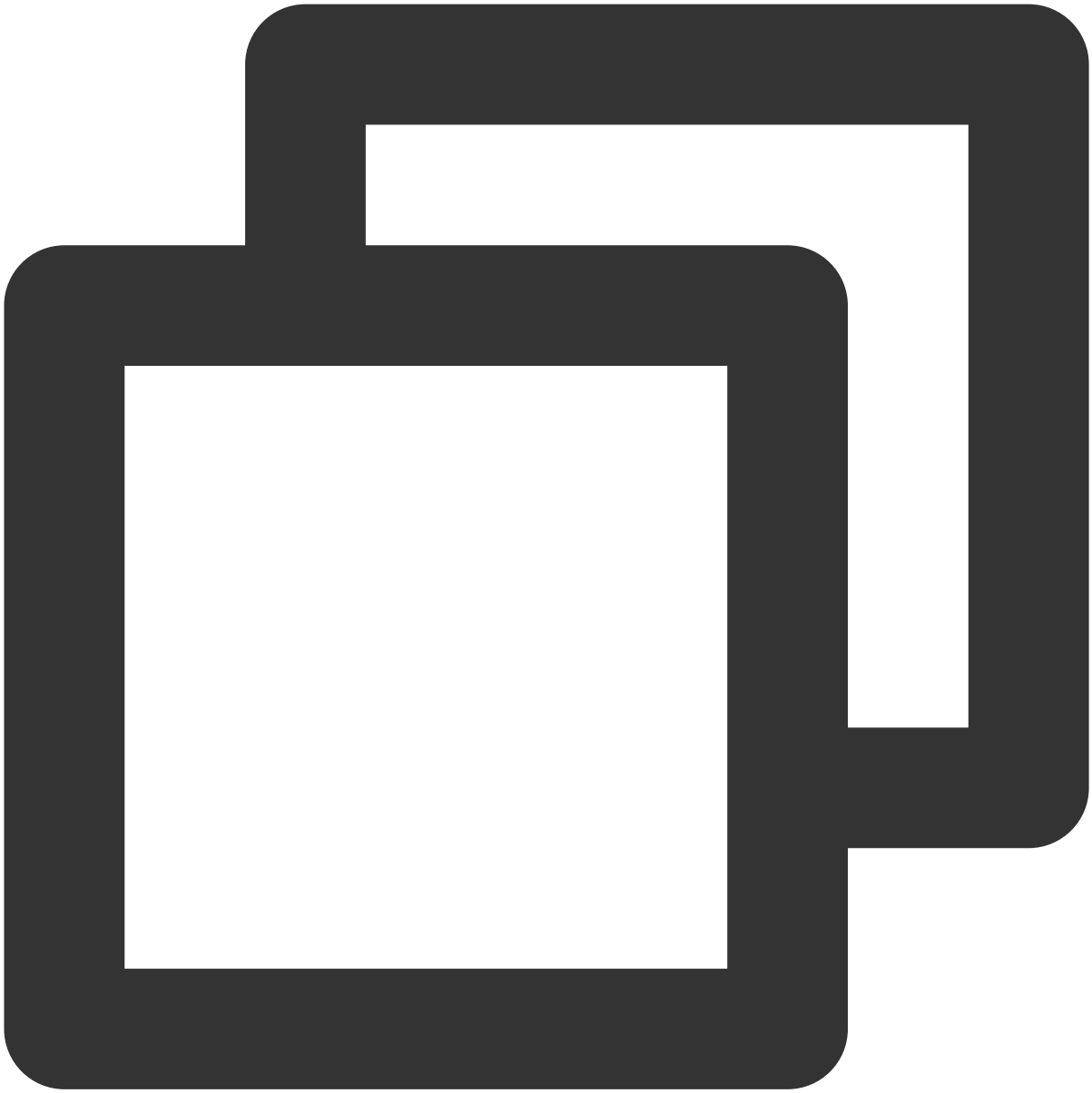
ステップ2：権限の設定および難読化ルール

AndroidManifest.xmlにAppの権限を設定します。SDKには次の権限が必要です（6.0以上のAndroidシステムではマイクの権限などを動的に申請してください）：



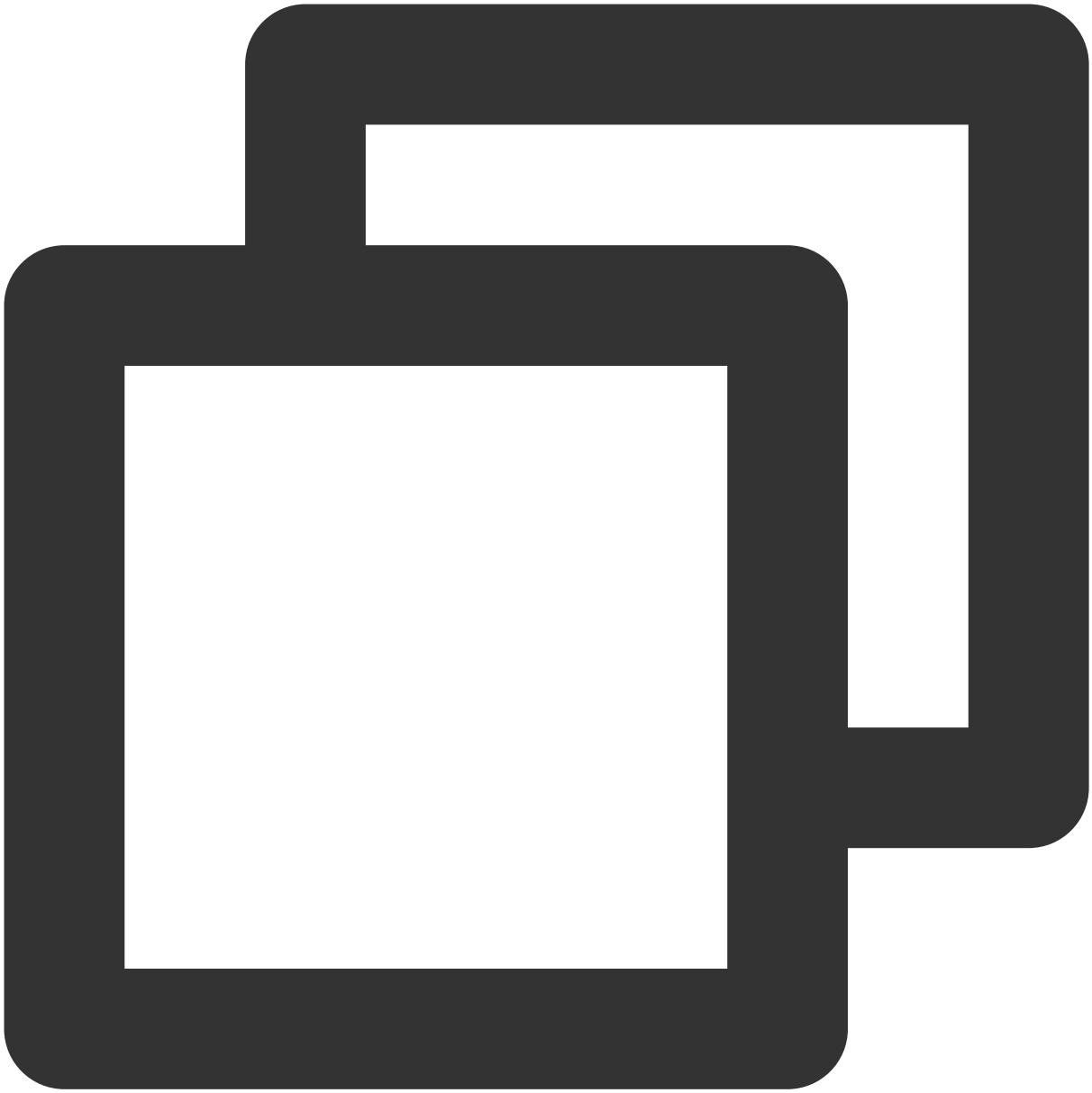
```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

proguard-rules.proファイルでは、SDK関連を非難読化リストに追加します：



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

ステップ3：初期化およびログイン

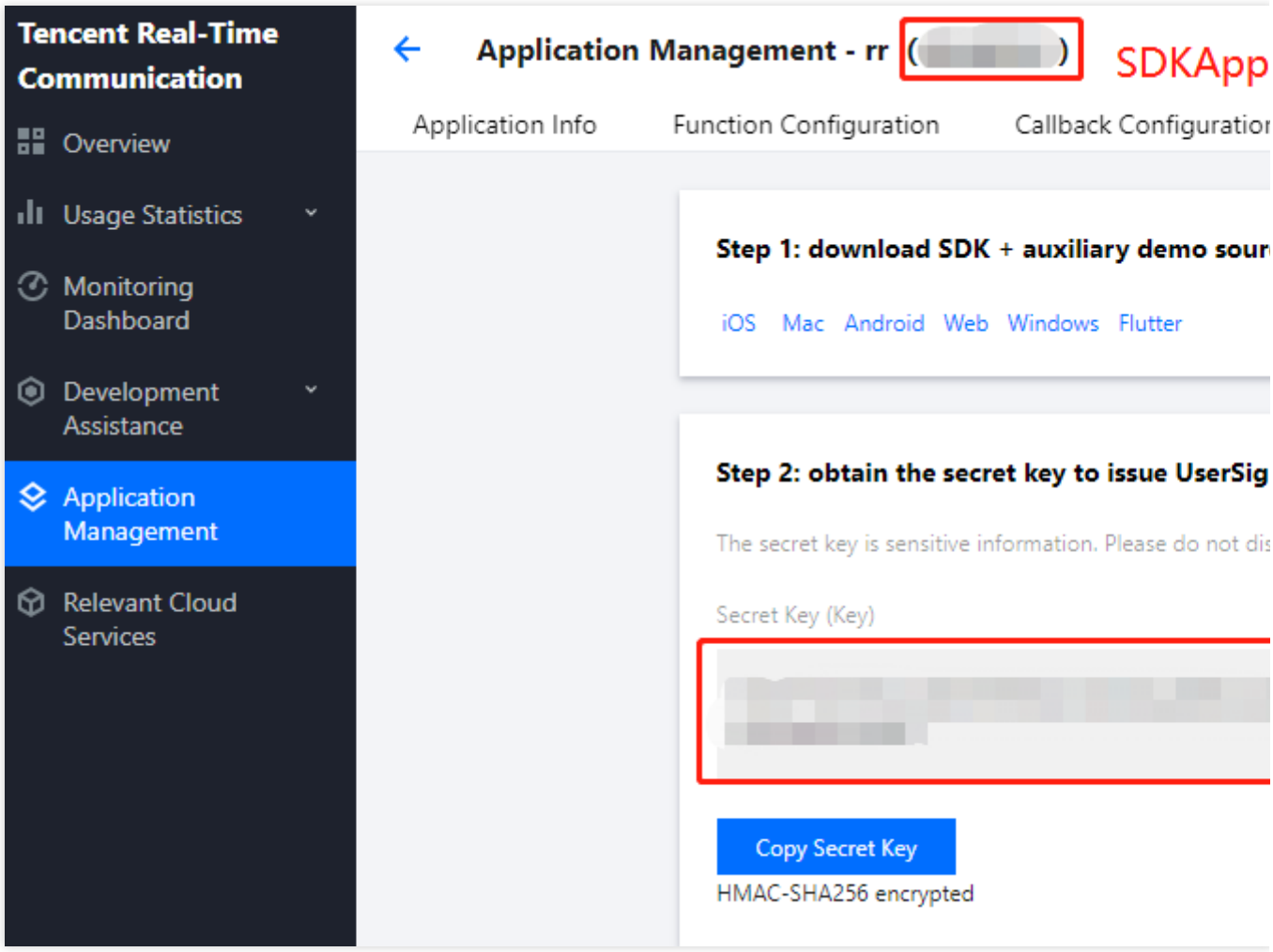


```
// 1.初期化
TRTCVoiceRoom mTRTCVoiceRoom = TRTCVoiceRoom.sharedInstance(this);
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
);
      // 2.ログイン
mTRTCVoiceRoom.login(SDKAppID, userId, userSig, new TRTCVoiceRoomCallback.Actio
    @Override
    public void onCallback(int code, String msg) {
        if (code == 0) {
        //ログイン成功
        }
```

```
            }
        });
```

パラメータの説明：

SDKAppID：**TRTCアプリケーションID**です。Tencent Cloud TRTCサービスをアクティブ化していない場合は、Tencent Cloud TRTCコンソールに進み、新しいTRTCアプリケーションを作成した後、**アプリケーション情報**をクリックすると、SDKAppID情報が次の図のように表示されます：



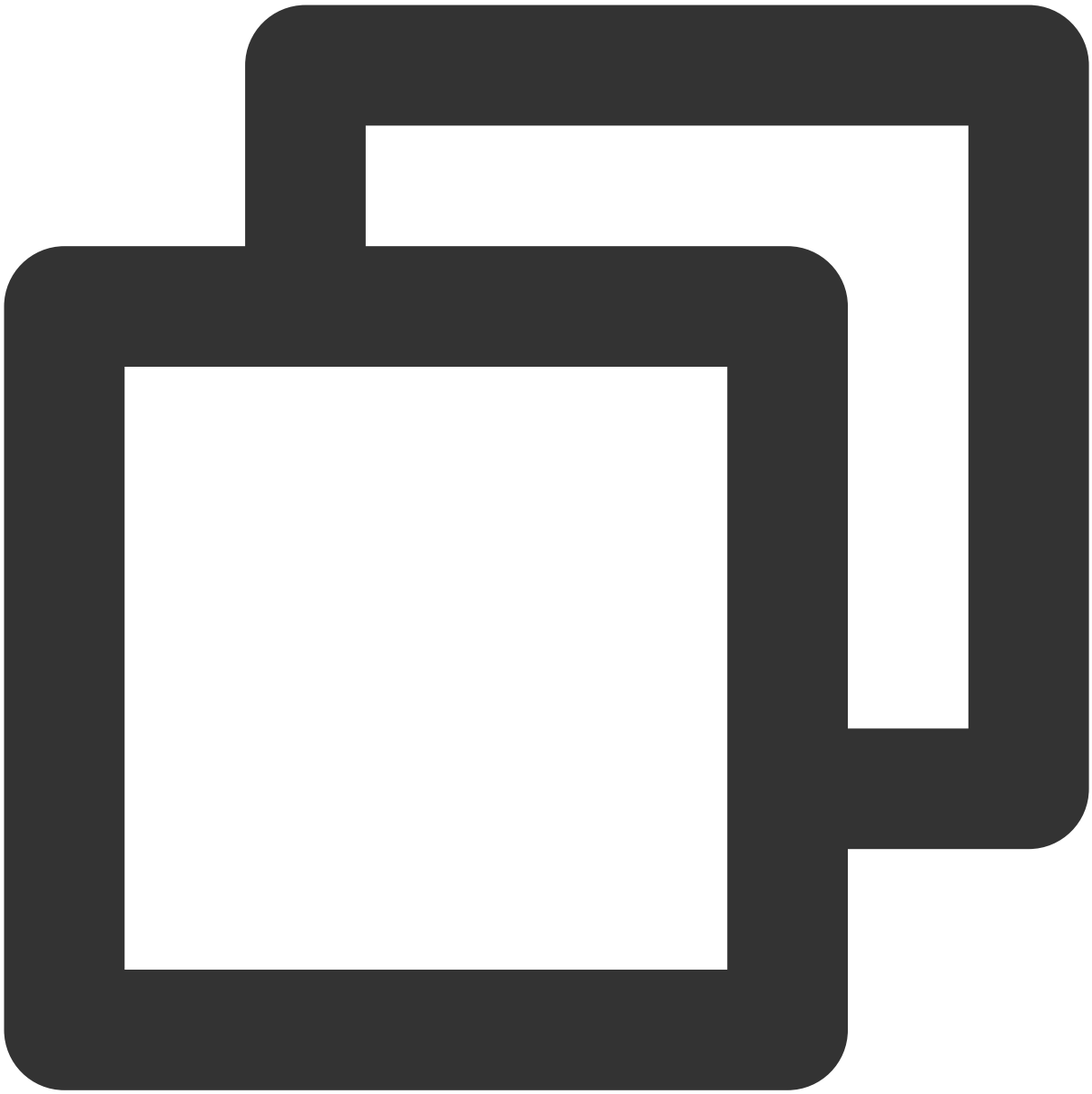
Secretkey：**TRTC アプリケーションキー**であり、SDKAppIdに対応しています。TRTCアプリケーション管理に進むと、SecretKey情報が上の図のように表示されます。

userId：現在のユーザーID。文字列タイプであり、英語のアルファベット（a-zとA-Z）、数字（0-9）、ハイフン（-）とアンダーライン（_）のみ使用できます。業務の実際のアカウントシステムと組み合わせてご自身で設定することをお勧めします。

userSig：SDKAppId、userId，Secretkeyなどの情報に基づく計算によって得られるセキュリティ保護署名です。ここをクリックするとデバッグ用のuserSigがオンラインで直接生成されます。UserSigの計算、使用方法をご参照ください。

ステップ4：ボイスチャットルームの実装

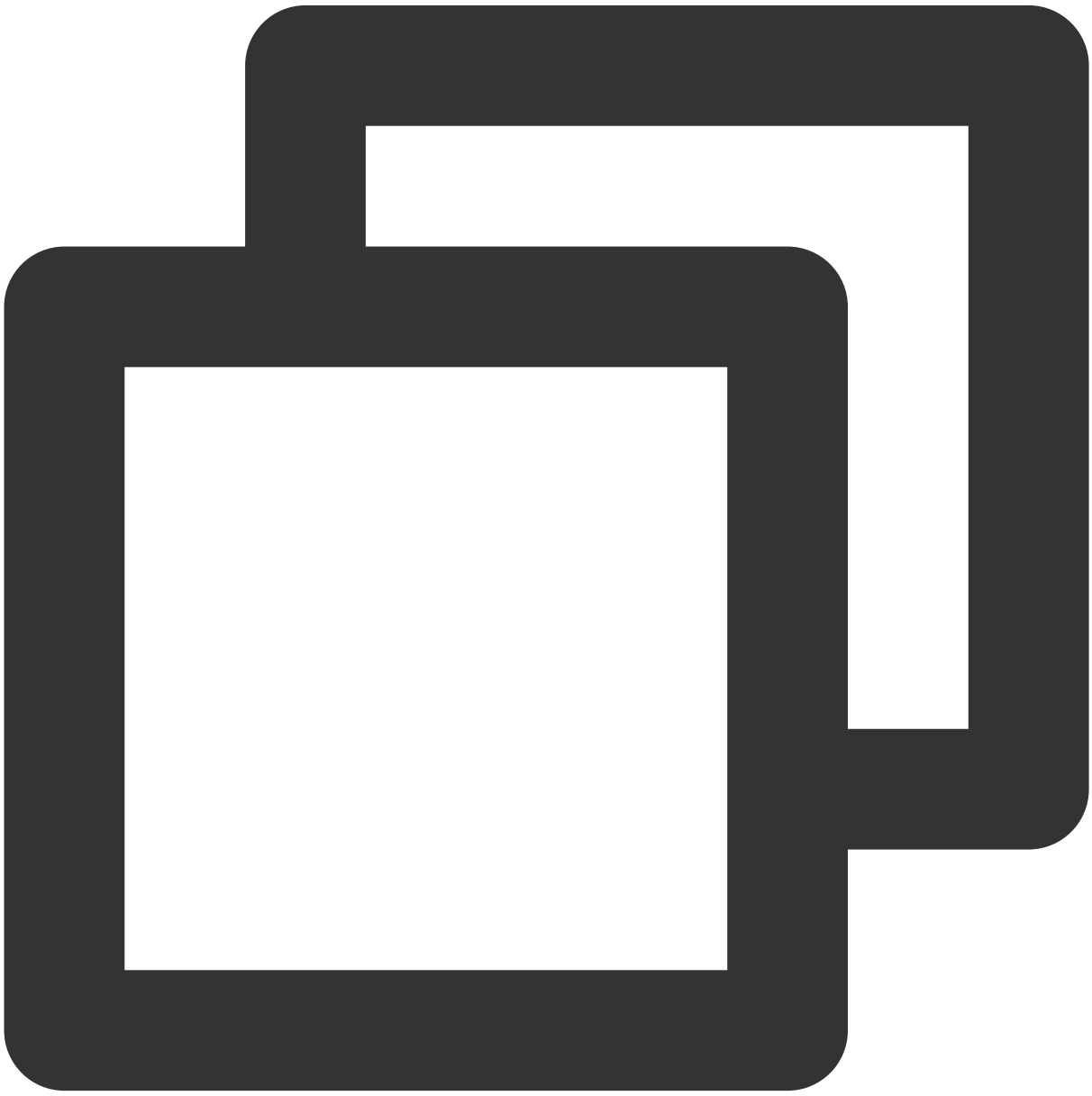
1. 管理者によるボイスチャットルーム作成の実装TRTCVoiceRoom#createRoom



```
// 1.管理者が呼び出してルームを作成
int roomId = 12345; //ルームid
final TRTCVoiceRoomDef.RoomParam roomParam = new TRTCVoiceRoomDef.RoomParam();
roomParam.roomName = "ルーム名";
roomParam.needRequest = false; // マイク・オンに対する管理者の確認の要否
roomParam.seatCount = 7; // ルームの座席数。ここでは計7席あり、管理者が1席を占め、残り6席がリ
roomParam.coverUrl = "ルームカバー図のURL ";
mTRTCVoiceRoom.createRoom(roomId, roomParam, new TRTCVoiceRoomCallback.ActionCallba
 @Override
```

```
 public void onCallback(int code, String msg) {
     if (code == 0) {
             //作成に成功
     }
 }
});
```

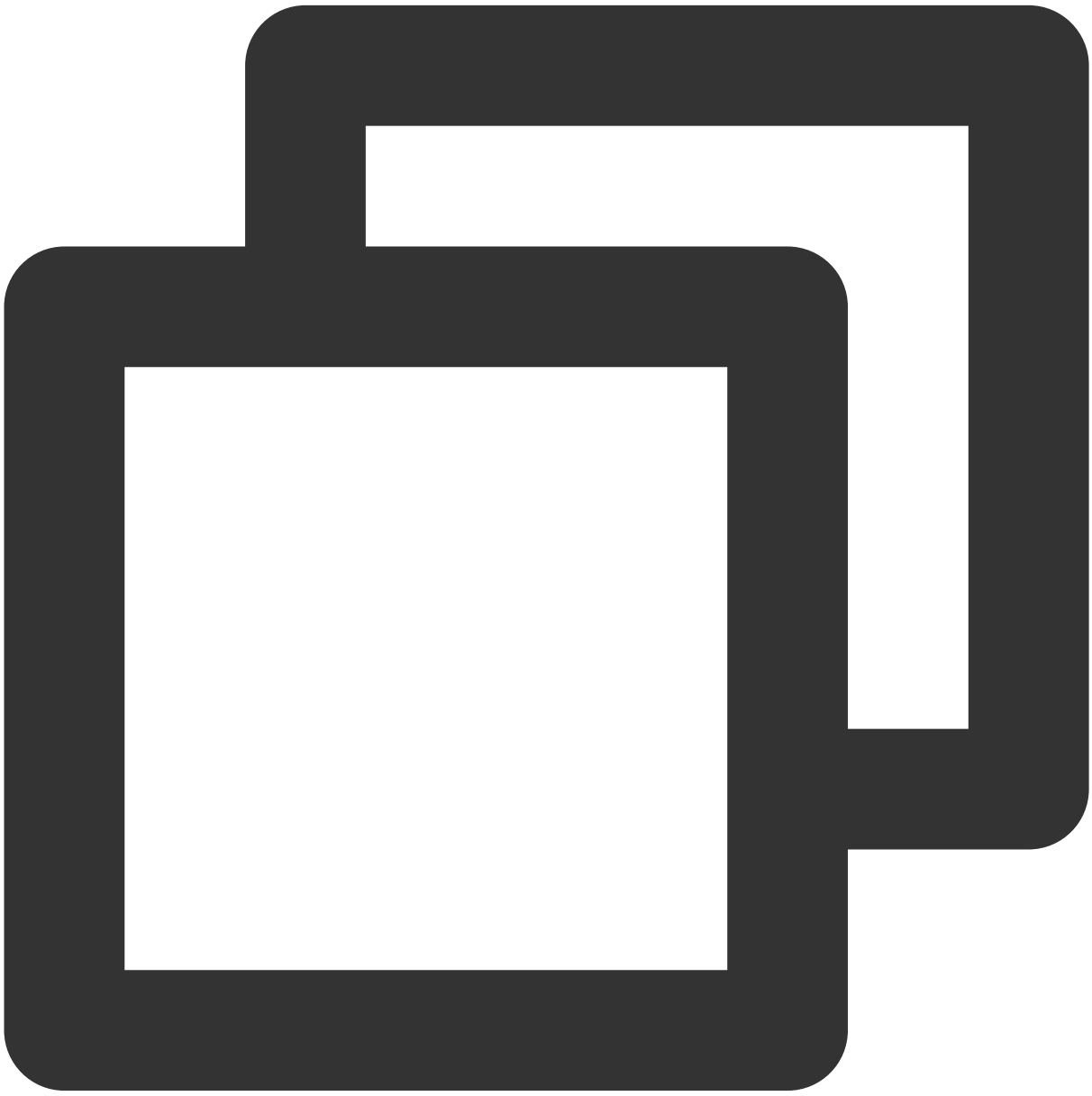
2. リスナーのボイスチャットルーム入室の実装TRTCVoiceRoom#enterRoom



```
// 1.リスナーが呼び出して入室
mTRTCVoiceRoom.enterRoom(roomId, new TRTCVoiceRoomCallback.ActionCallback() {
     @Override
```

```
        public void onCallback(int code, String msg) {
            if (code == 0) {
            //入室に成功
            }
        }
});
```

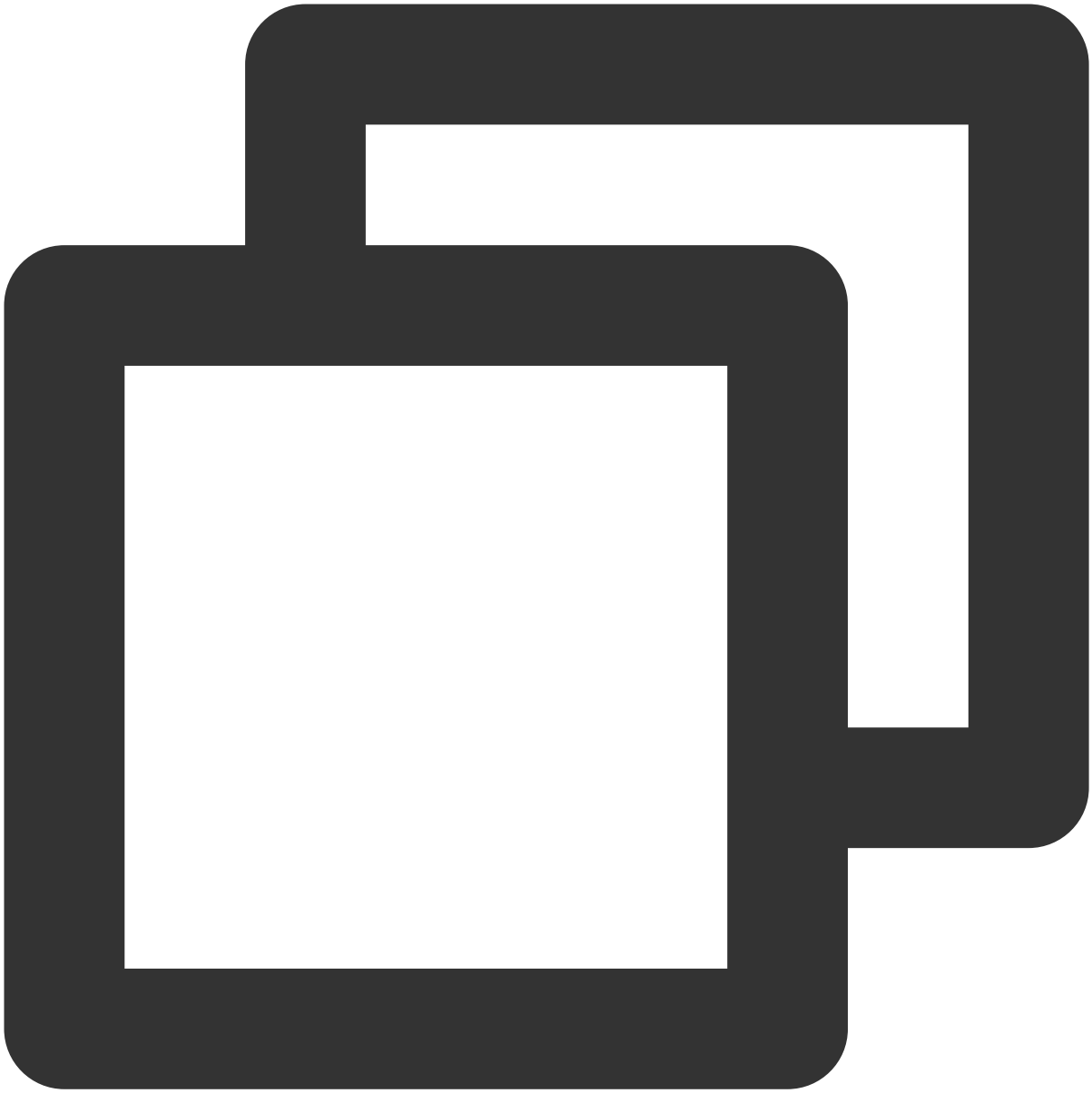
3. リスナーの自主的なマイク・オンの実装TRTCVoiceRoom#enterRoom



```
// 1: リスナーが呼び出してマイク・オン
int seatIndex = 2; //マイクのindex
mTRTCVoiceRoom.enterSeat(seatIndex, new TRTCVoiceRoomCallback.ActionCallback() {
```

```
 @Override
 public void onCallback(int code, String msg) {
     if (code == 0) {
     //操作に成功しました
     }
 }
});

// 2. onSeatListChangeコールバックを受信し、マイクリストを更新します
@Override
public void onSeatListChange(final List<TRTCVoiceRoomDef.SeatInfo> seatInfoList) {
}
```

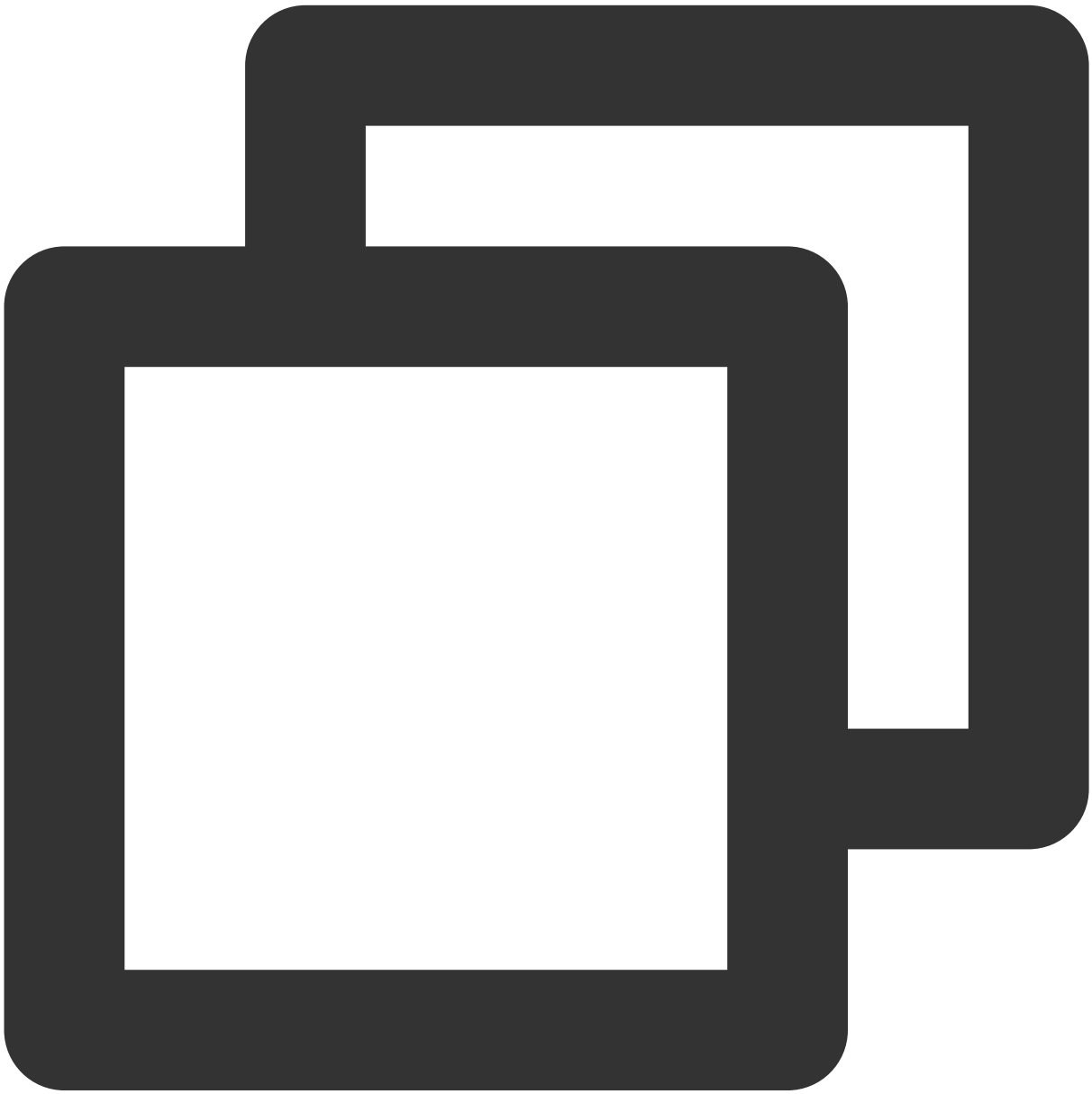
4. **管理者による視聴者発言招待の実装**TRTCVoiceRoom#pickSeat



```
// 1: 管理者が呼び出して、視聴者が発言できるように招待
int seatIndex = 2; //マイクのindex
String userId = "123"; //マイク・オンが必要なユーザーid
mTRTCVoiceRoom.pickSeat(1, userId, new TRTCVoiceRoomCallback.ActionCallback() {
 @Override
 public void onCallback(int code, String msg) {
     if (code == 0) {
             //操作に成功しました
     }
 }
});
```

```
// 2. onSeatListChangeコールバックを受信し、マイクリストを更新します
@Override
public void onSeatListChange(final List<TRTCVoiceRoomDef.SeatInfo> seatInfoList) {
}
```

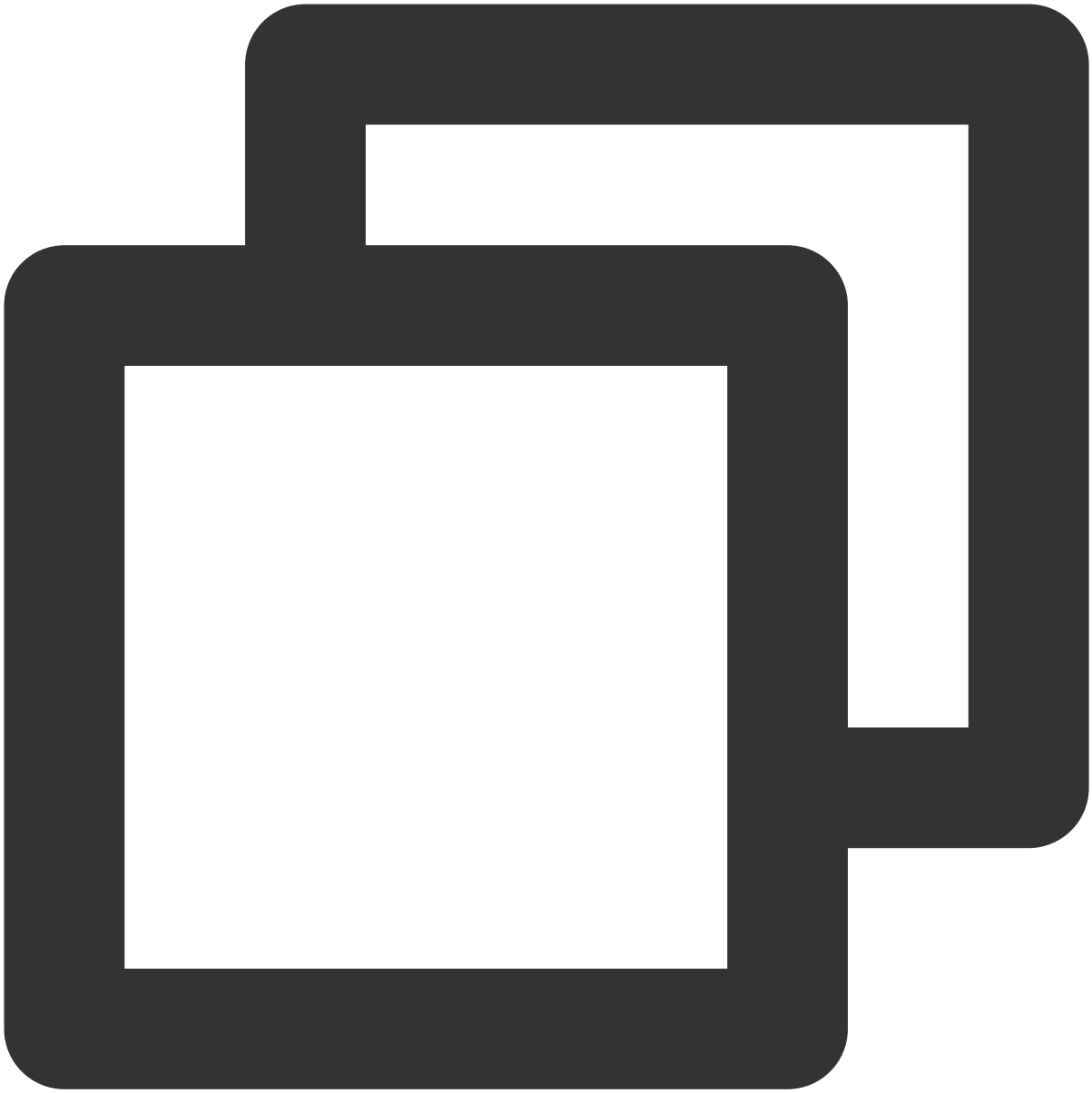
5. **リスナーによるマイク・オン申請の実装** TRTCVoiceRoom#sendInvitation



```
// リスナー側の視点
// 1.リスナーが呼び出してマイク・オンを申請
String seatIndex = "1"; //マイクのindex
String userId = "123"; //ユーザーid
```

```
String inviteId = mTRTCVoiceRoom.sendInvitation("takeSeat", userId, seatIndex, null

// 2.招待のリクエスト同意を受信し、正式にマイク・オンになります
@Override
public void onInviteeAccepted(String id, String invitee) {
 if(id.equals(inviteId)) {
     mTRTCVoiceRoom.enterSeat(index, null);
 }
}

// 管理者側の視点
// 1.管理者がリクエストを受信します
 @Override
public void onReceiveNewInvitation(final String id, String inviter, String cmd, fin
 if (cmd.equals("takeSeat")) {
     // 2.管理者がリスナーのリクエストに同意します
      mTRTCVoiceRoom.acceptInvitation(id, null);
 }
}
```

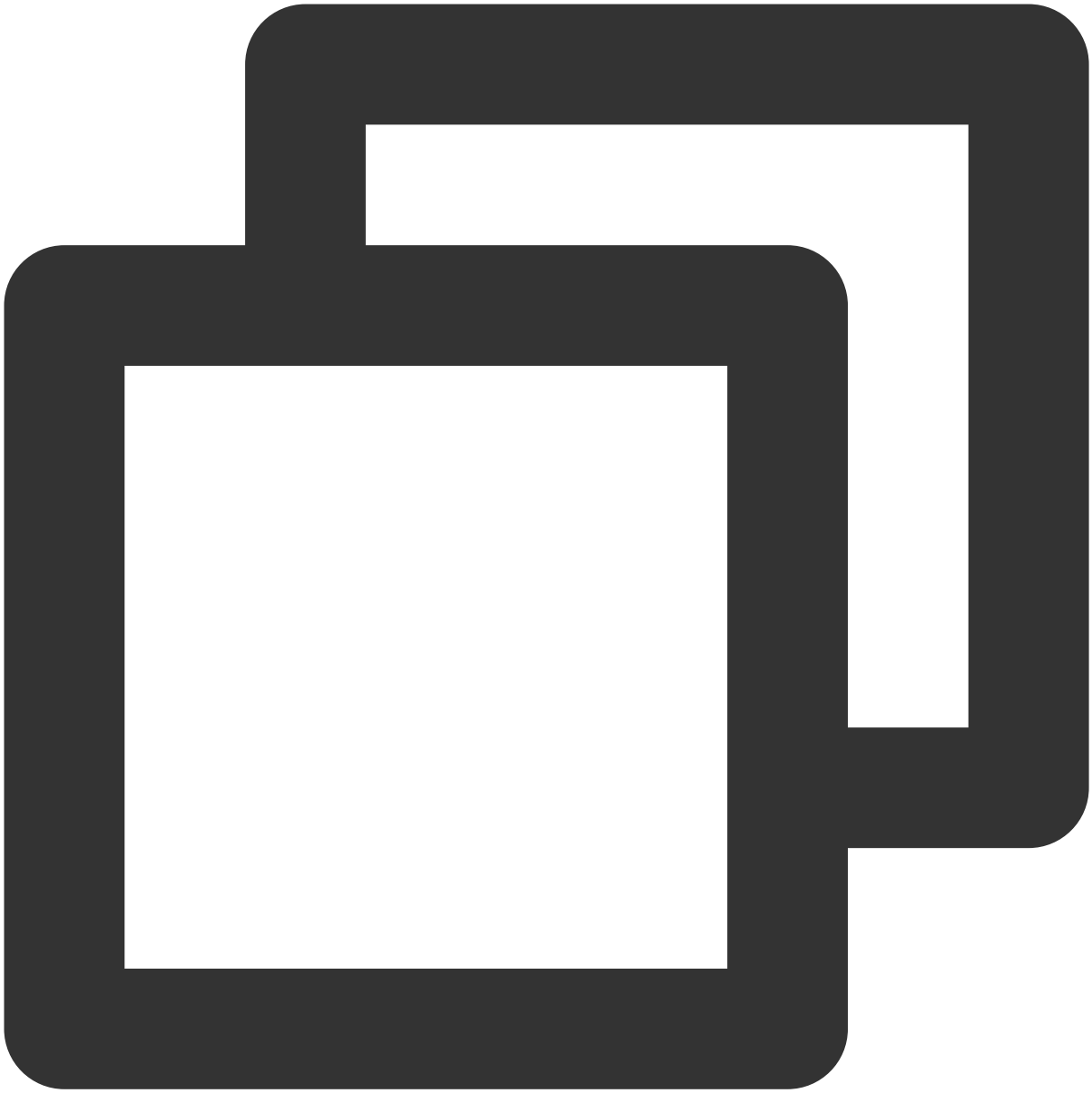
6. **管理者によるマイク・オンへの招待の実装** TRTCVoiceRoom#sendInvitation



```
// 管理者側の視点
// 1.管理者が呼び出して、視聴者が発言できるように招待をリクエスト
String seatIndex = "1"; //マイクのindex
String userId = "123"; //ユーザーid
String inviteId = mTRTCVoiceRoom.sendInvitation("pickSeat", userId, seatIndex, null

// 2.招待のリクエスト同意を受信し、正式にマイク・オンになります
@Override
public void onInviteeAccepted(String id, String invitee) {
 if(id.equals(inviteId)) {
     mTRTCVoiceRoom.pickSeat(index, null);
```

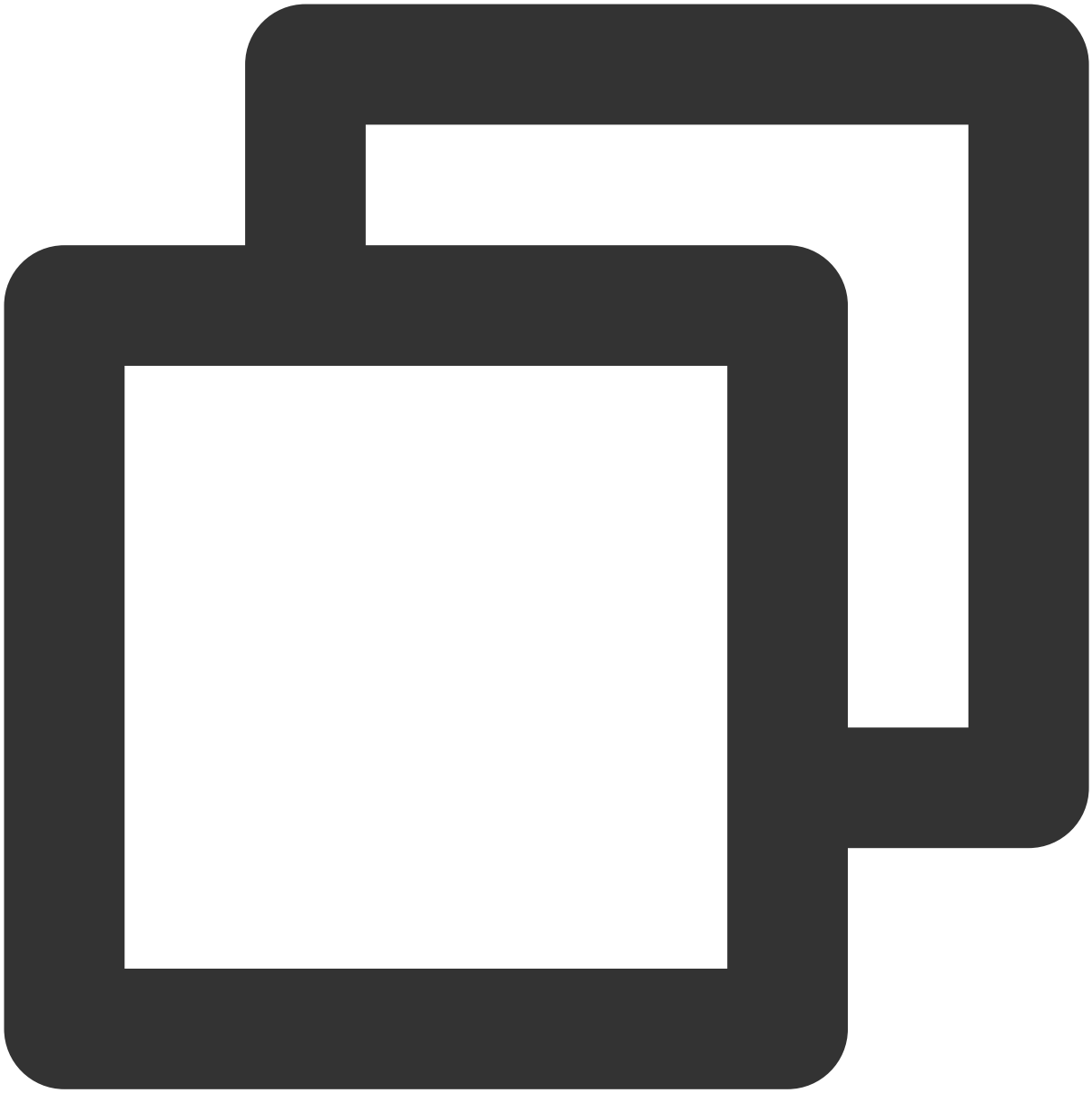
```
    }
}

// リスナー側の視点
// 1.リスナーがリクエストを受信します
 @Override
public void onReceiveNewInvitation(final String id, String inviter, String cmd, fin
 if (cmd.equals("pickSeat")) {
     // 2.リスナーが管理者のリクエストに同意します
      mTRTCVoiceRoom.acceptInvitation(id, null);
 }
}
```

7. **テキストチャットの実装**TRTCVoiceRoom#sendRoomTextMsg



```
// 発信側：テキストメッセージの発信
mTRTCVoiceRoom.sendRoomTextMsg("Hello Word!", null);
// 受信側：テキストメッセージのモニタリング
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
  @Override
  public void onRecvRoomTextMsg(String message, TRTCVoiceRoomDef.UserInfo userInfo)
   Log.d(TAG,"が" + userInfo.userName + "から受信したメッセージ:" + message);
  }
});
```

8. **弾幕コメントの実装** TRTCVoiceRoom#sendRoomCustomMsg



```
// 発信側：カスタマイズCmdによって弾幕と「いいね」情報を区別することができます
// eg:「CMD_DANMU」は弾幕コメントを表し、「CMD_LIKE」は「いいね」情報を表します
mTRTCVoiceRoom.sendRoomCustomMsg("CMD_DANMU", "Hello world", null);
mTRTCVoiceRoom.sendRoomCustomMsg("CMD_LIKE", "", null);
// 受信側：カスタムメッセージのモニタリング
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
 @Override
 public void onRecvRoomCustomMsg(String cmd, String message, TRTCVoiceRoomDef.UserI
     if ("CMD_DANMU".equals(cmd)) {
         // 弾幕コメントの受信
         Log.d(TAG, "が" + userInfo.userName + "から受信した弾幕コメント:" + message);
```

```
        } else if ("CMD_LIKE".equals(cmd)) {
            // 「いいね」情報の受信
            Log.d(TAG, userInfo.userName + "いいねを付けました！");
        }
    }
});
```

よくあるご質問

ご要望やフィードバックなどがございましたら、colleenyu@tencent.comまでご連絡ください。

TUIVoiceRoom (iOS)の統合

最終更新日：：2024-07-19 15:35:35

コンポーネントの説明

TUIVoiceRoomはオープンソースのオーディオビデオUIコンポーネントであり、プロジェクトにTUIVoiceRoomコンポーネントを統合することにより、数行のコードを書くだけで、Appに「多人数ボイスチャット」などのシーンを組み込むことができます。TUIVoiceRoomはAndroidなどのプラットフォームをサポートしています。基本機能は下図のとおりです：

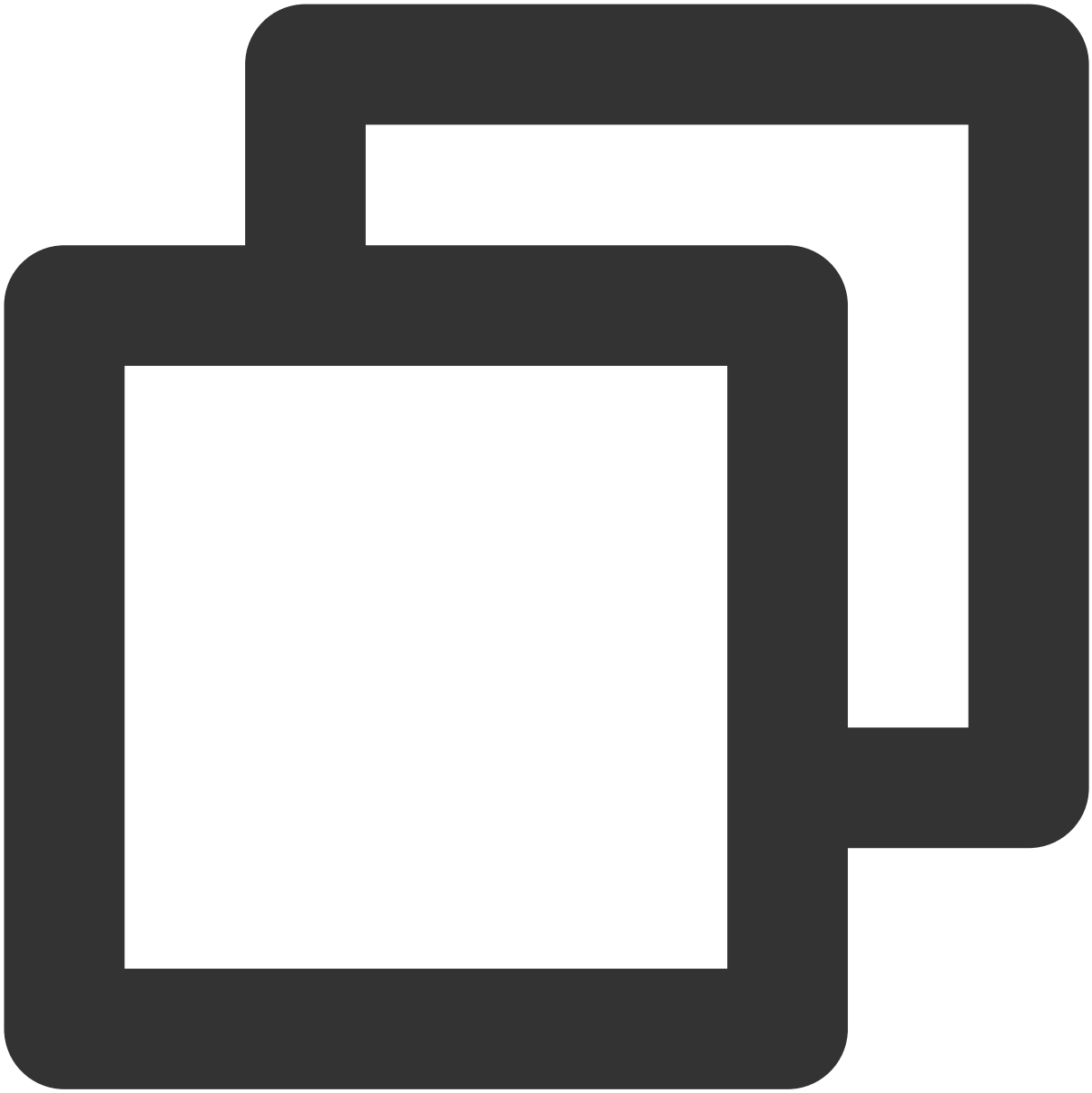
説明：

TUIKitシリーズコンポーネントはTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期的にアクティブにすることができます。IMサービスの課金ルールの詳細については、Instant Messagingの料金説明をご参照ください。TRTCをアクティブにすると、デフォルトでは、100DAUまでサポートするIM SDK体験版もアクティブになりますs。

コンポーネントの統合

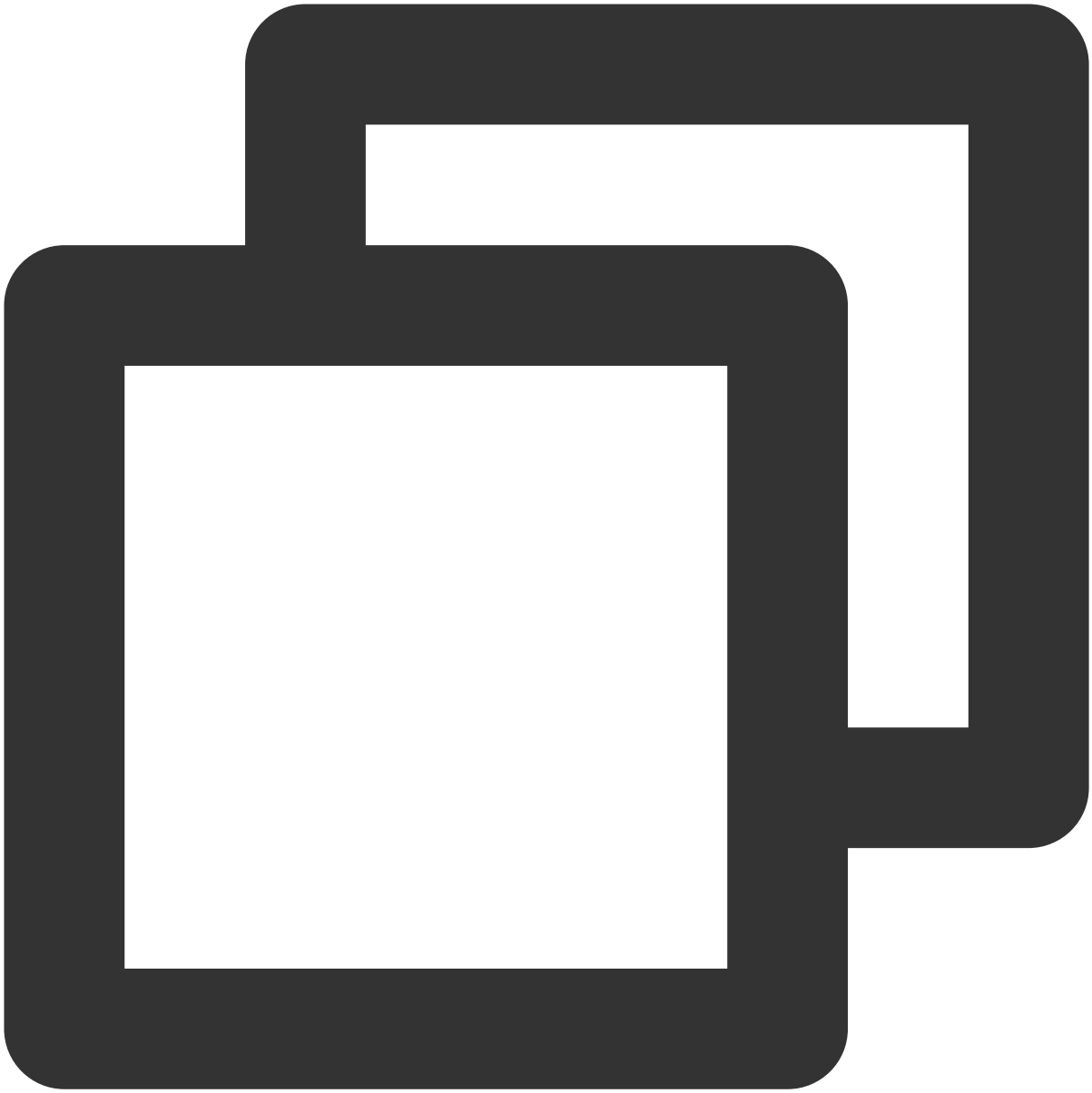
ステップ1：TUIVoiceRoomコンポーネントのダウンロードとインポート

xcodeプロジェクトの `Podfile` ファイルと同一階層のディレクトリ下に `TUIVoiceRoom` フォルダを作成し、GithubリポジトリiOSディレクトリ下のTXAppBasic、Resources、Source、TUIVoiceRoom.podspecなどのファイルを、ご自身のプロジェクトで作成した `TUIVoiceRoom` ディレクトリ下にコピーします。さらに、次のようにインポート動作を完了します：
プロジェクトのPodfileファイルを開き、TUIVocieRoom.podspecをインポートします。次をご参照ください：



```
# pathはTXAppBasic.podspecのPodfileファイルに対する相対パスです
pod 'TXAppBasic', :path => "TUIVoiceRoom/TXAppBasic/"
# pathはTUIVoiceRoom.podspecのPodfileファイルに対する相対パスです
pod 'TUIVoiceRoom', :path => "TUIVoiceRoom/", :subspecs => ["TRTC"]
```

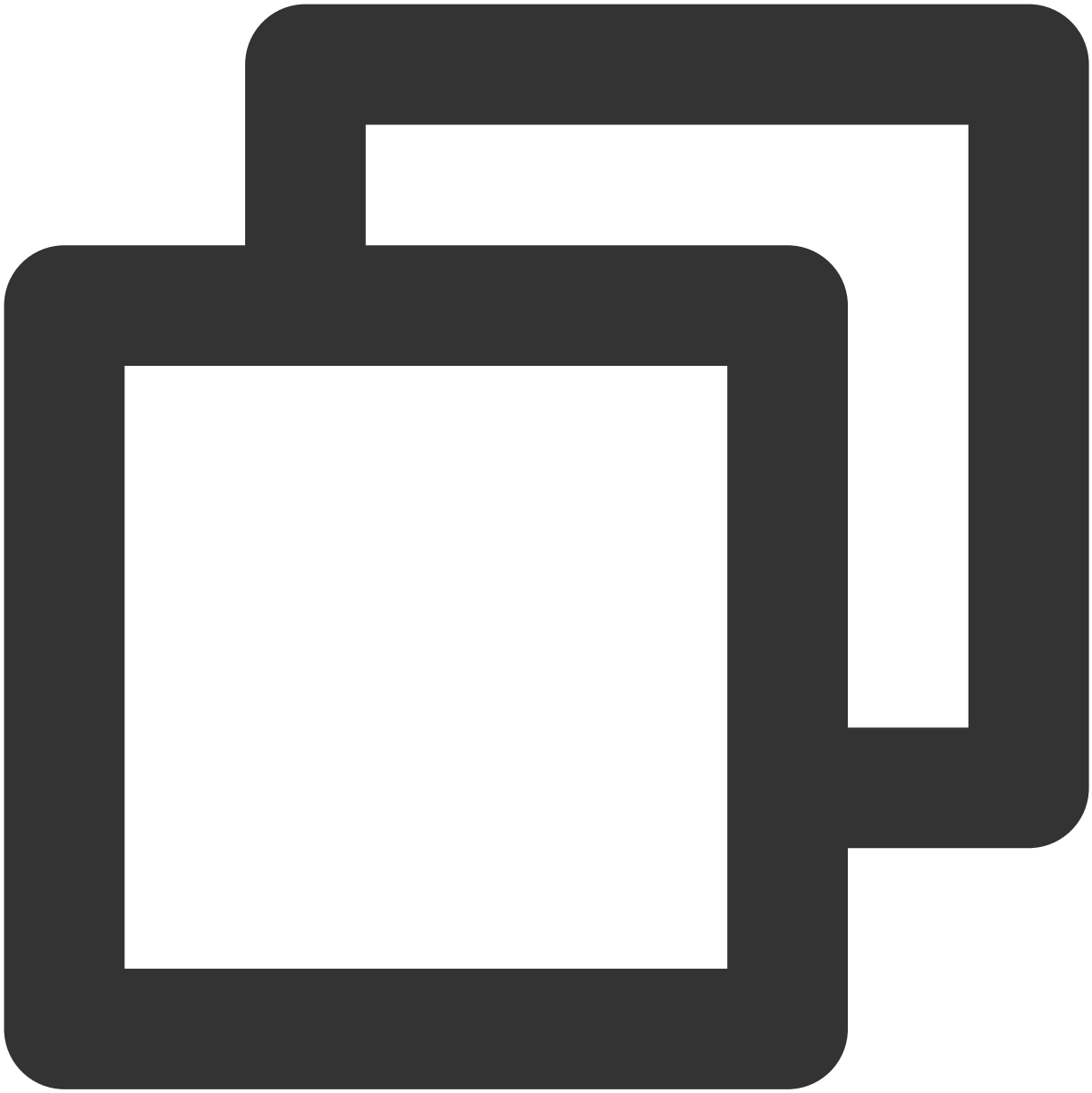
端末でPodfileのあるディレクトリ下に入り、`pod install`を実行します。次をご参照ください：



```
pod install
```

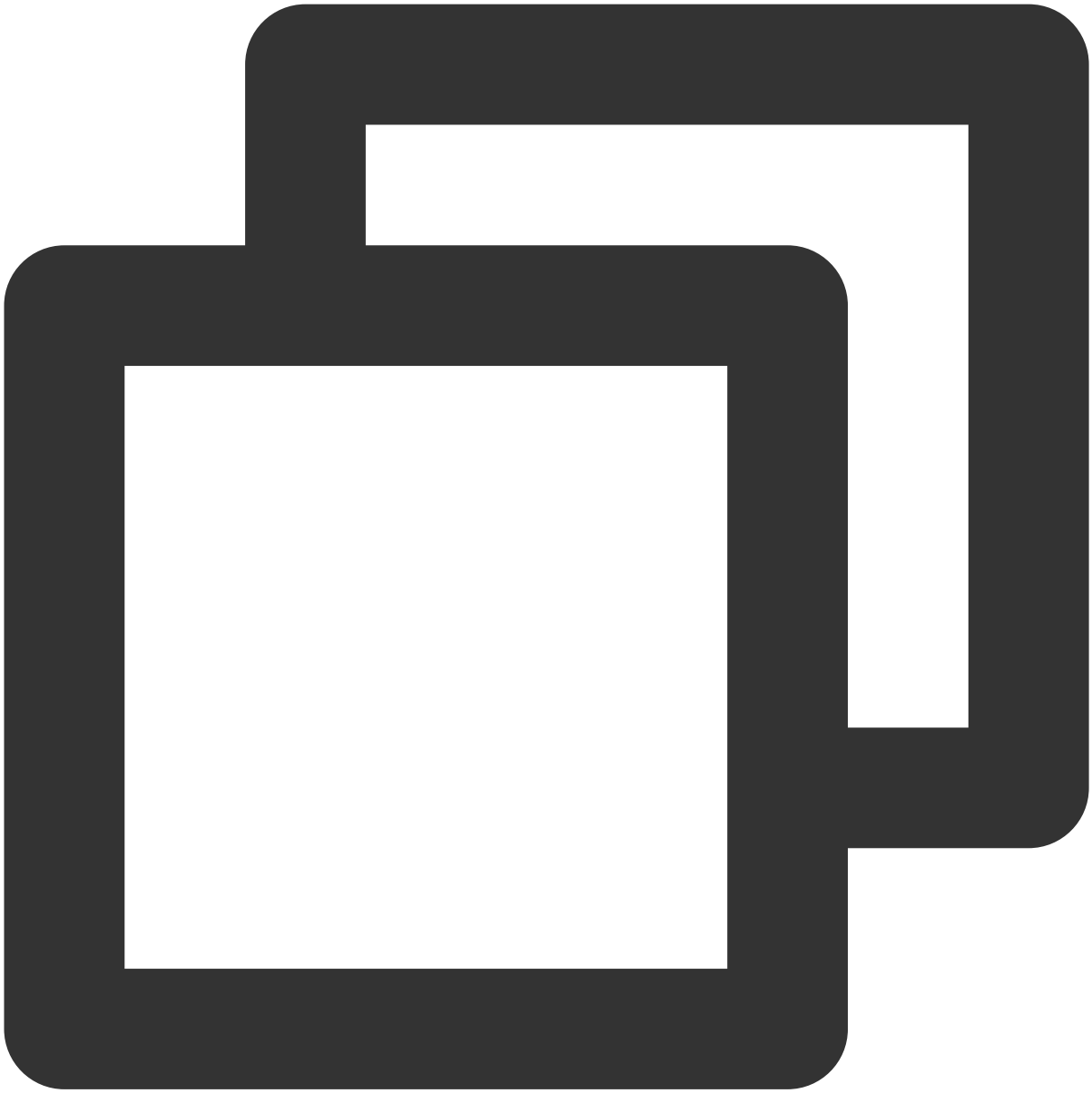
ステップ2：権限の設定および難読化ルール

info.plistファイルにおいて、 `Privacy > Microphone Usage Description` を追加して、マイクの権限を申請してください。



```
<key>NSMicrophoneUsageDescription</key>
<string>VoiceRoomAppはマイクへのアクセス権限が必要です。有効にした後にレコーディングしたビデオで
```

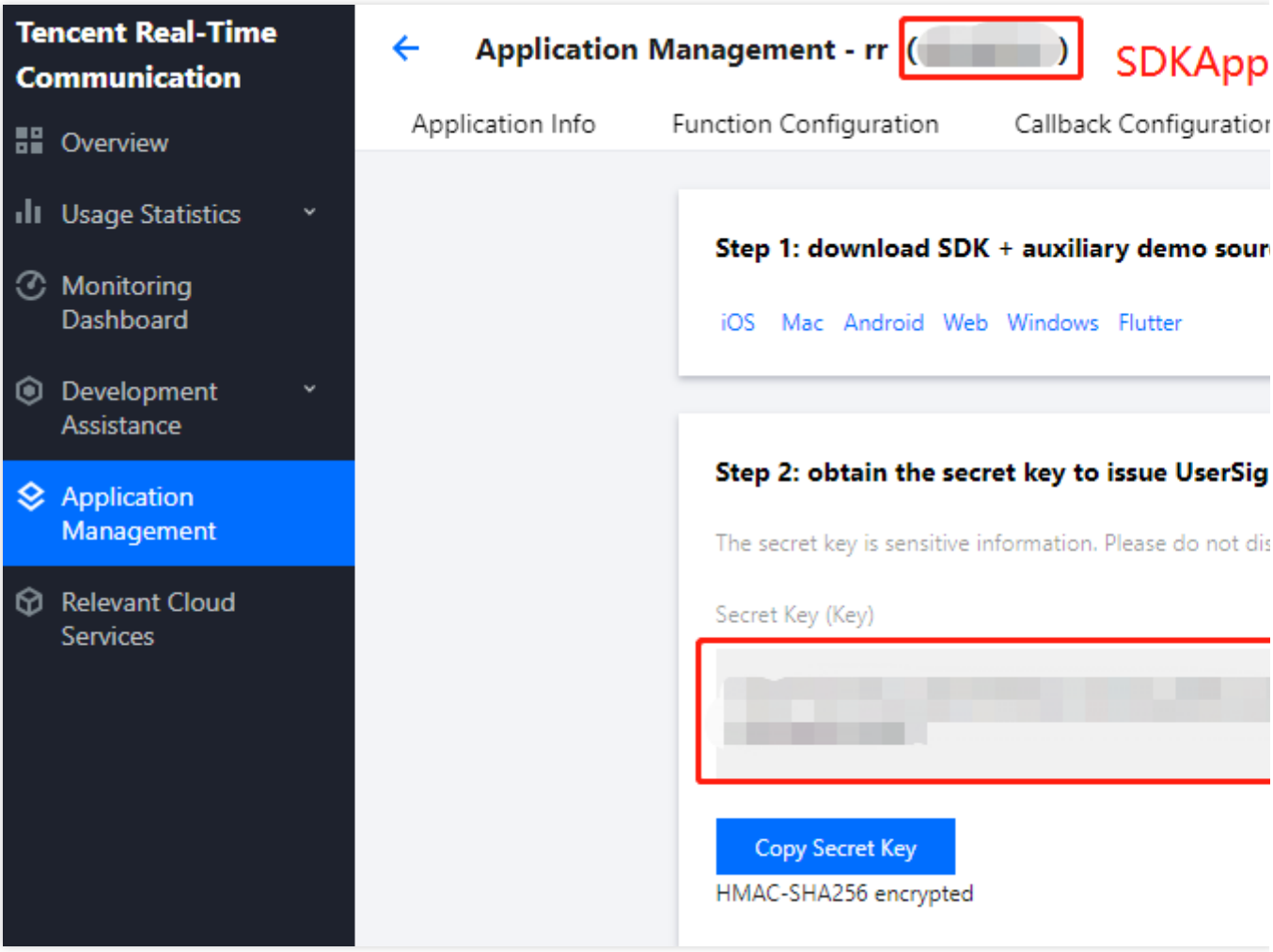
ステップ3：初期化およびログイン



```
// 初期化
let mTRTCVoiceRoom = TRTCVoiceRoom.shared()
// ログイン
mTRTCVoiceRoom.login(sdkAppID: SDKAppID, userId: userId, userSig: userSig) { code,
    if code == 0 {
        //ログイン成功
    }
}
```

パラメータの説明：

SDKAppID：**TRTCアプリケーションID**です。Tencent Cloud TRTCサービスをアクティブ化していない場合は、Tencent Cloud TRTCコンソールに進み、新しいTRTCアプリケーションを作成した後、**アプリケーション情報**をクリックすると、SDKAppID情報が次の図のように表示されます：



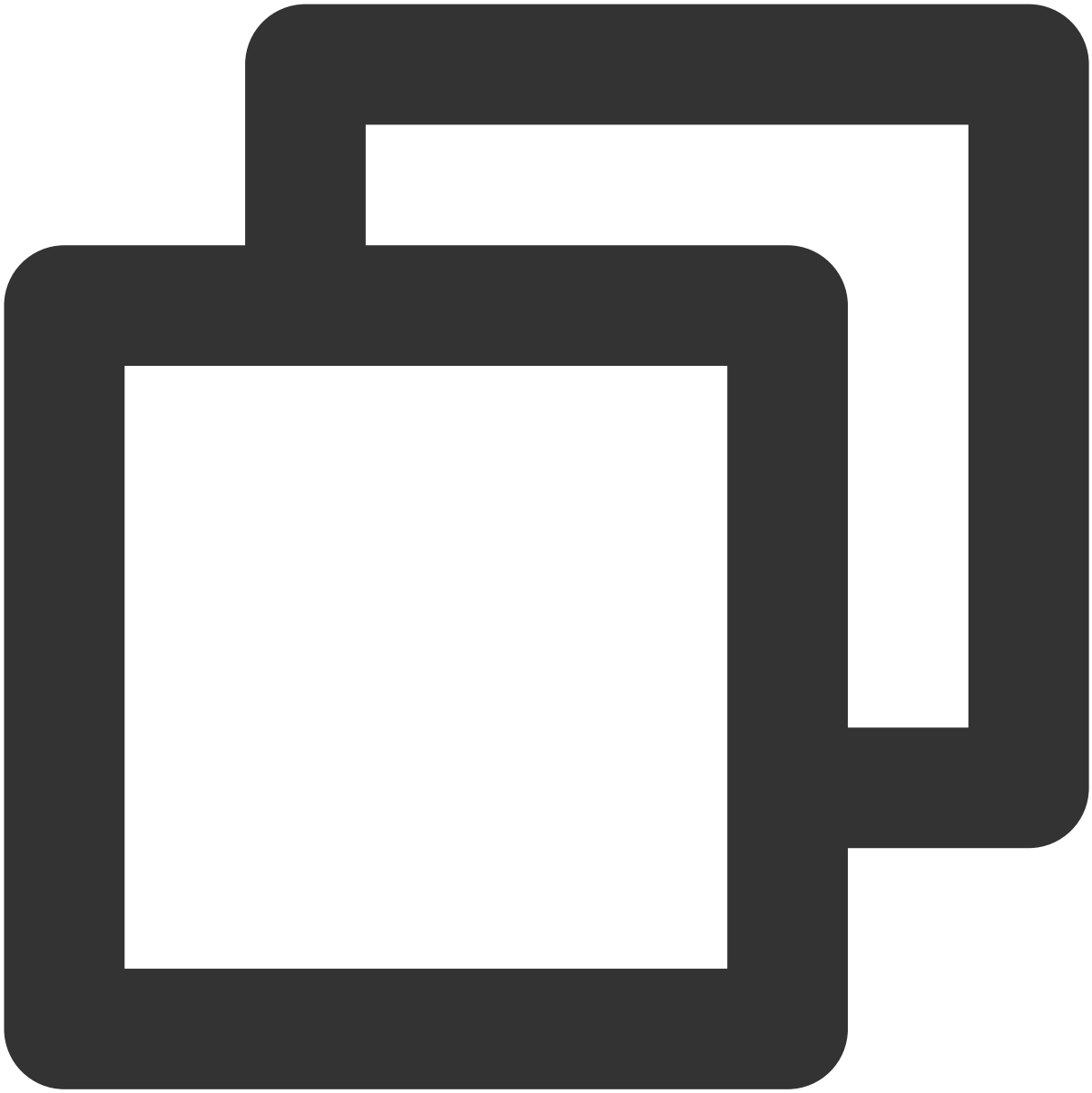
Secretkey：**TRTC アプリケーションキー**であり、SDKAppIdに対応しています。TRTCアプリケーション管理に進むと、SecretKey情報が上の図のように表示されます。

userId：現在のユーザーID。文字列タイプであり、英語のアルファベット（a-zとA-Z）、数字（0-9）、ハイフン（-）とアンダーライン（_）のみ使用できます。業務の実際のアカウントシステムと組み合わせてご自身で設定することをお勧めします。

userSig：SDKAppId、userId，Secretkeyなどの情報に基づく計算によって得られるセキュリティ保護署名です。ここをクリックするとデバッグ用のuserSigがオンラインで直接生成されます。UserSigの計算、使用方法をご参照ください。

ステップ4：ボイスチャットルームの実装

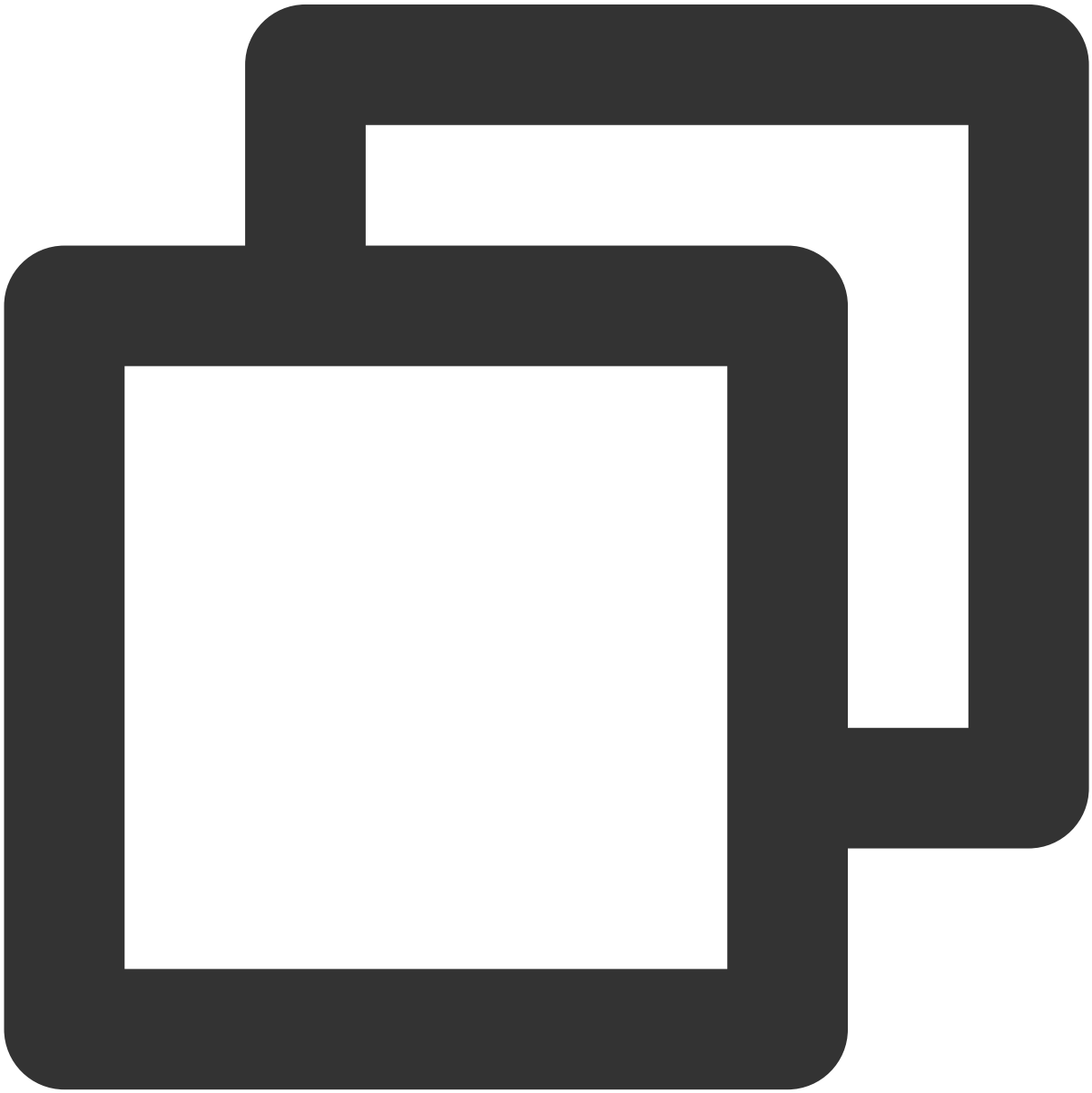
1. **管理者によるボイスチャットルーム作成の実装**TRTCVoiceRoom#createRoom



```
// ボイスチャットルームパラメータの初期化
let roomParam = VoiceRoomParam()
roomParam.roomName = "ルーム名"
roomParam.needRequest = false // リスナーのマイク・オンに対する管理者の同意の要否
roomParam.coverUrl = "ルームカバー図のURL"
roomParam.seatCount = 7 // ルームの座席数。ここでは計7席あり、管理者が1席を占め、残り6席がリス
roomParam.seatInfoList = []
// マイク情報の初期化
for _ in 0..< param.seatCount {
 let seatInfo = VoiceRoomSeatInfo()
 param.seatInfoList.append(seatInfo)
```

```
}
// 管理者側でルームを作成
mTRTCVoiceRoom.createRoom(roomID: yourRoomID, roomParam: roomParam) { (code, messag
 if code == 0 {
     // 作成に成功
 }
}
```

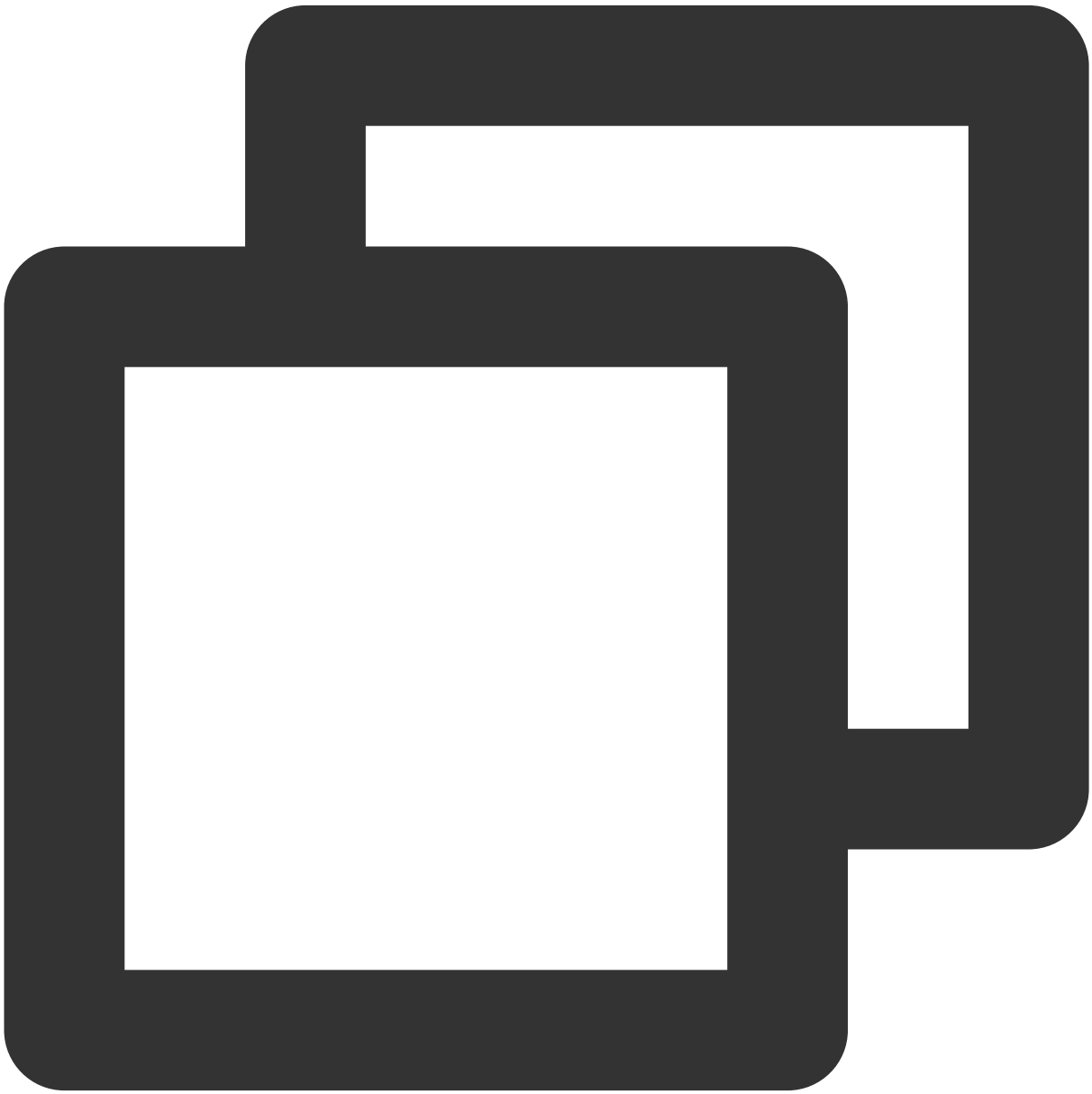
2. **リスナーのボイスチャットルーム入室の実装**TRTCVoiceRoom#enterRoom



```
// 1.リスナーが呼び出して入室
mTRTCVoiceRoom.enterRoom(roomID: roomID) { (code, message) in
```

```
 // 入室結果コールバック
 if code == 0 {
    // 入室に成功
 }
}
```

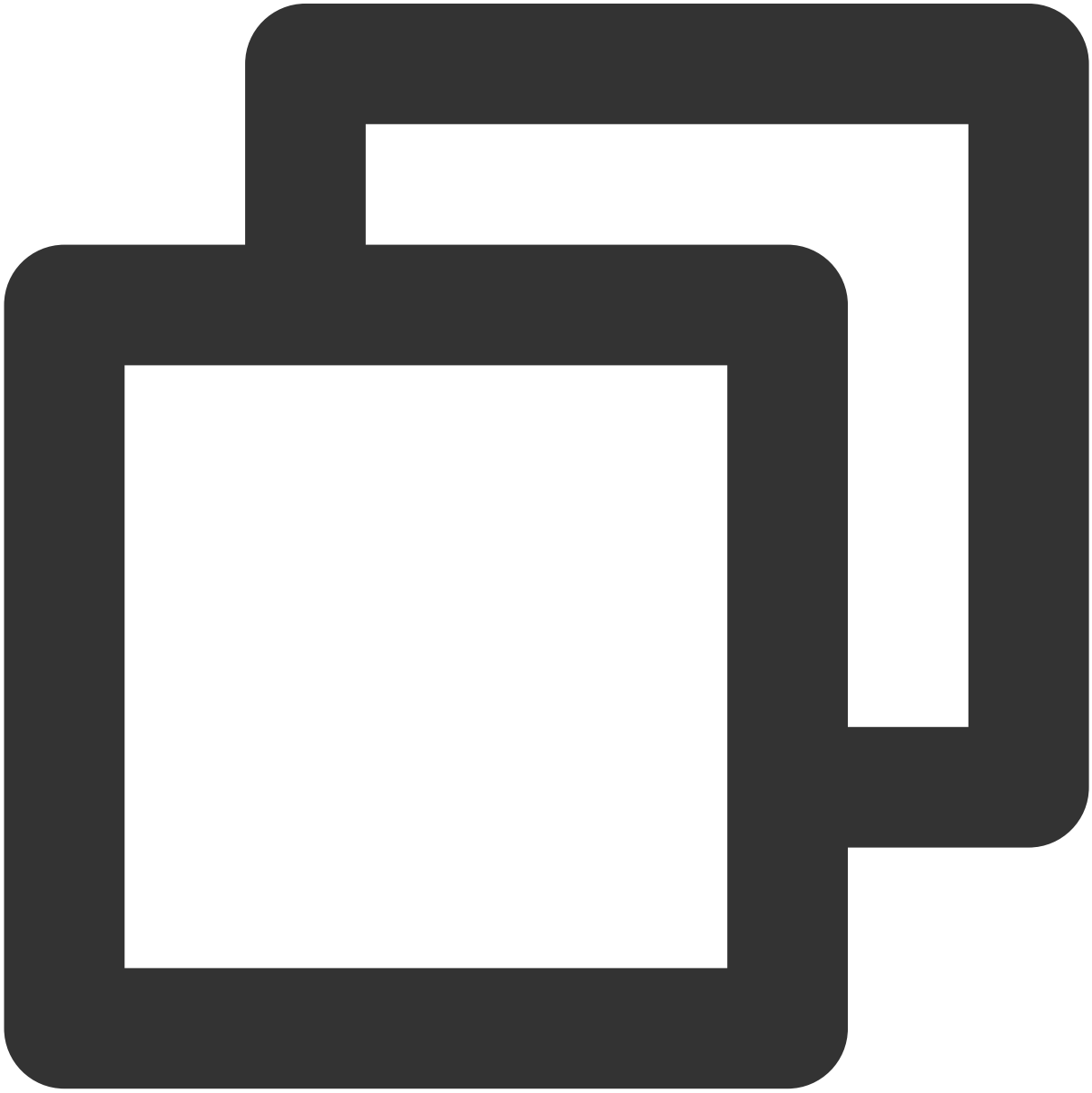
3. リスナーの自主的なマイク・オンの実装TRTCVoiceRoom#enterRoom



```
// 1: リスナーが呼び出してマイク・オン
let seatIndex = 2; //マイクのindex
mTRTCVoiceRoom.enterSeat(seatIndex: 2) { (code, message) in
 if code == 0 {
```

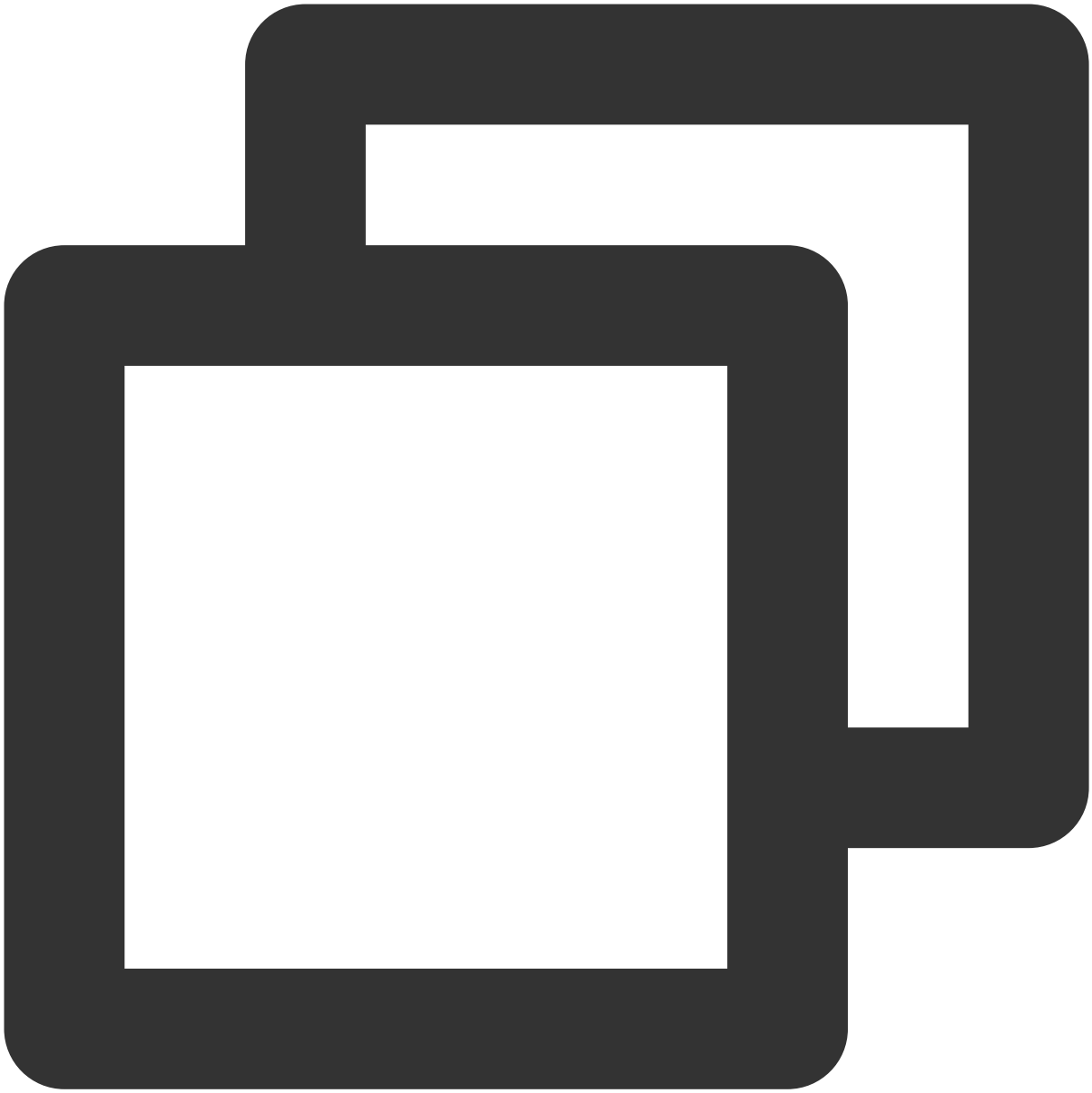
```
        // マイク・オン成功
    }
}

// 2. onSeatListChangeコールバックを受信し、マイクリストを更新します
@Override
func onSeatListChange(seatInfoList: [VoiceRoomSeatInfo]) {
    // 更新したマイクリスト
}
```

4. **管理者による視聴者発言招待の実装** TRTCVoiceRoom#pickSeat



```
// 1: 管理者が呼び出して、視聴者が発言できるように招待
let seatIndex = 2; //マイクのindex
let userId = "123"; //マイク・オンが必要なユーザーid
mTRTCVoiceRoom.pickSeat(seatIndex: 1, userId: "123") { (code, message) in
 if code == 0 {
 }
}

// 2. onSeatListChangeコールバックを受信し、マイクリストを更新します
func onSeatListChange(seatInfoList: [VoiceRoomSeatInfo]) {
 // 更新したマイクリスト
}
```

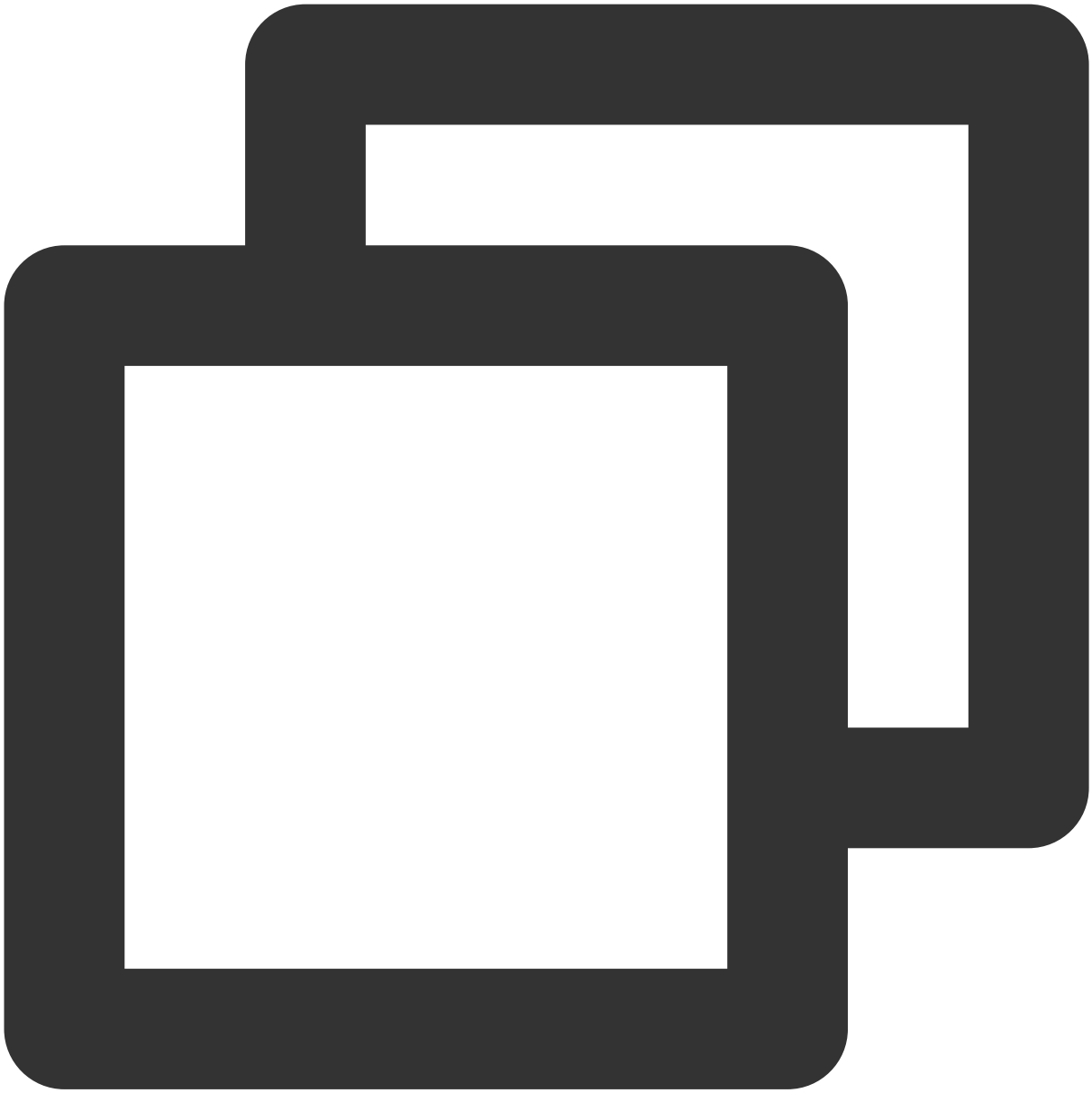
5. **リスナーによるマイク・オン申請の実装** TRTCVoiceRoom#sendInvitation



```
// リスナー側の視点
// 1.リスナーが呼び出してマイク・オンを申請
let seatIndex = "1"; //マイクのindex
let userId = "123"; //ユーザーid
let inviteId = mTRTCVoiceRoom.sendInvitation(cmd: "takeSeat", userId: ownerUserId,
 // 発信結果のコールバック
}

// 2.招待のリクエスト同意を受信し、正式にマイク・オンになります
func onInviteeAccepted(identifier: String, invitee: String) {
 if identifier == selfID {
```

```
        self.mTRTCVoiceRoom.enterSeat(seatIndex: ) { (code, message) in
            // マイク・オン結果のコールバック
        }
    }
}

// 管理者側の視点
// 1.管理者がリクエストを受信します
func onReceiveNewInvitation(identifier: String, inviter: String, cmd: String, conte
    if cmd == "takeSeat" {
        // 2.管理者がリスナーのリクエストに同意します
        self.mTRTCVoiceRoom.acceptInvitation(identifier: identifier, callback: nil)
    }
}
```

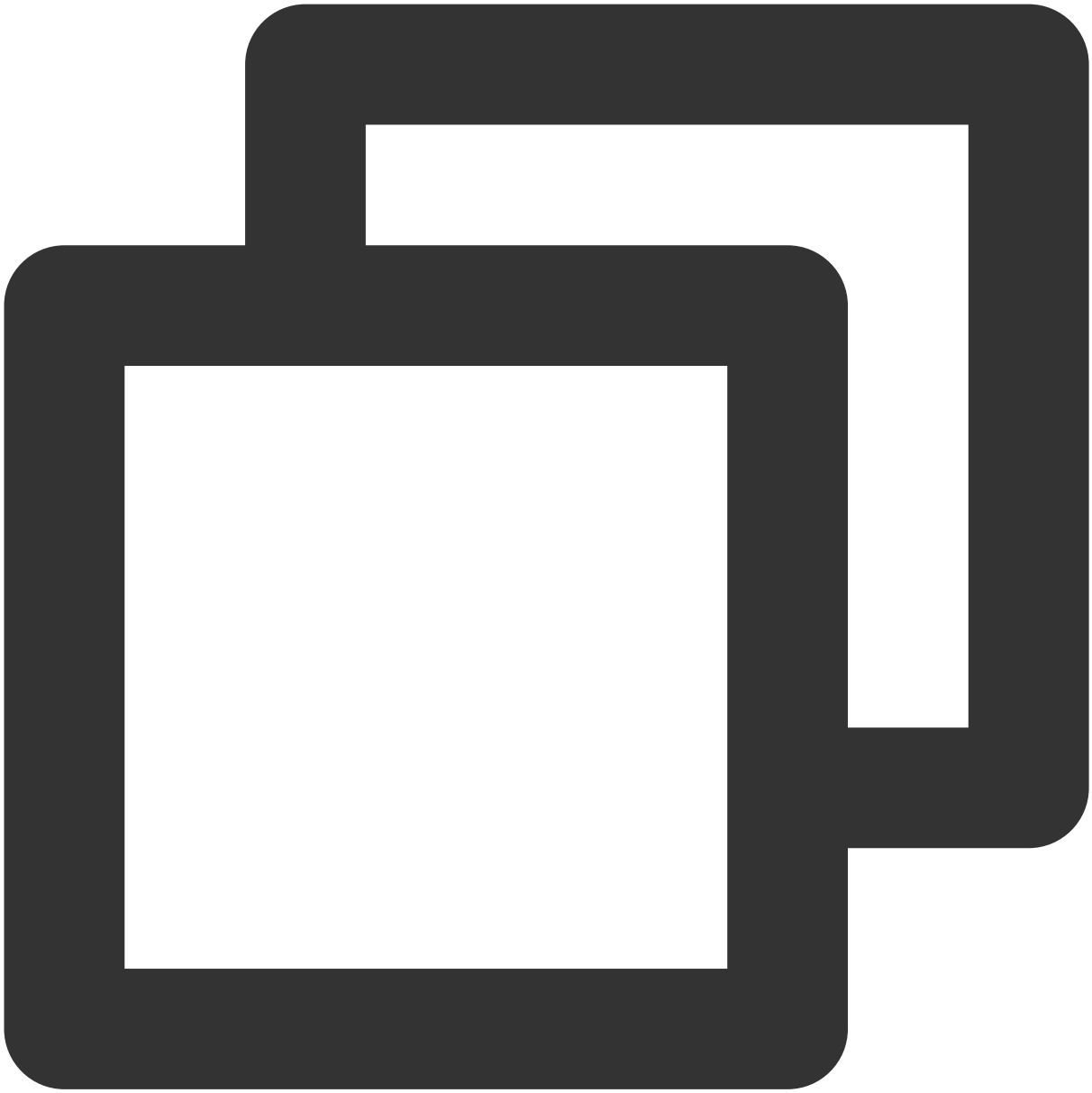
6. **管理者によるマイク・オンへの招待の実装** TRTCVoiceRoom#sendInvitation



```
// 管理者側の視点
// 1.管理者はsendInvitationを呼び出して、リスナー「123」をピックして2号マイクのマイク・オンをリ
let inviteId = self.mTRTCVoiceRoom.sendInvitation(cmd: "pickSeat", userId: ownerUse
 // 発信結果のコールバック
}

// 2.招待のリクエスト同意を受信し、正式にマイク・オンになります
func onInviteeAccepted(identifier: String, invitee: String) {
 if identifier == selfID {
     self.mTRTCVoiceRoom.pickSeat(seatIndex: ) { (code, message) in
         // マイク・オン結果のコールバック
```

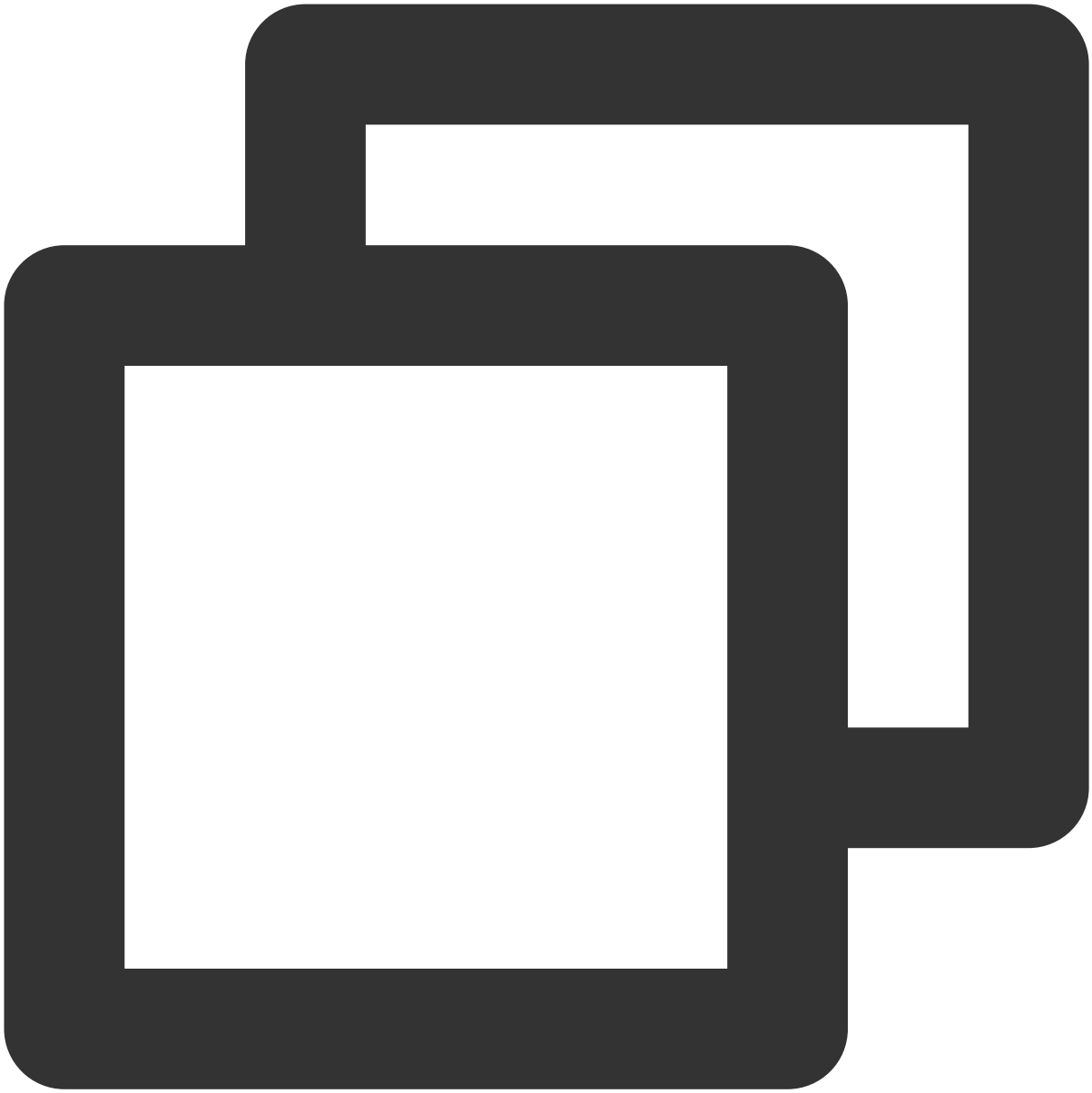
```
        }
    }
}

// リスナー側の視点
// 1.リスナーがリクエストを受信します
func onReceiveNewInvitation(identifier: String, inviter: String, cmd: String, conte
    if cmd == "pickSeat" {
        // 2.リスナーが管理者のリクエストに同意します
        self.mTRTCVoiceRoom.acceptInvitation(identifier: identifier, callback: nil)
    }
}
```

7. **テキストチャットの実装** TRTCVoiceRoom#sendRoomTextMsg



```
// 発信側：テキストメッセージの発信
self.mTRTCVoiceRoom.sendRoomTextMsg(message: message) { (code, message) in

}

// 受信側：テキストメッセージのモニタリング
func onRecvRoomTextMsg(message: String, userInfo: VoiceRoomUserInfo) {
 // 受信したmessage情報の処理方法
}
```

8. **弾幕コメントの実装** TRTCVoiceRoom#sendRoomCustomMsg



```
// 例：発信側：カスタマイズCmdによって、弾幕と「いいね」情報を区分することができます
// eg:「CMD_DANMU」は弾幕コメントを表し、「CMD_LIKE」は「いいね」情報を表します
self.mTRTCVoiceRoom.sendRoomCustomMsg(cmd: "CMD_DANMU", message: "hello world", cal
self.mTRTCVoiceRoom.sendRoomCustomMsg(cmd: "CMD_LIKE", message: "", callback: nil)

// 受信側：カスタムメッセージのモニタリング
func onRecvRoomCustomMsg(cmd: String, message: String, userInfo: VoiceRoomUserInfo)
 if cmd == "CMD_DANMU" {
     // 弾幕コメントの受信
 }
 if cmd == "CMD_LIKE" {
```

```
        // 「いいね」情報の受信
    }
}
```

よくあるご質問

ご要望やフィードバックなどがございましたら、colleenyu@tencent.comまでご連絡ください。

TUIVoiceRoom API照会
TRTCVoiceRoom (iOS)

最終更新日：：2024-07-19 15:35:35

TRTCVoiceRoomは、Tencent CloudのTencent Real-Time Communication (TRTC)およびIMサービスを基に組み合わせたコンポーネントで、以下の機能をサポートしています：
管理者が新しいボイスチャットルームを作成して配信を開始し、リスナーがボイスチャットルームに参加して視聴/インタラクティブなコミュニケーションを行います。
管理者は、リスナーを招待してマイク・オンにしたり、マイク・オンのキャスターをキックアウトしたりすることもできます。
管理者はまた、座席をクローズすることができ、その他のリスナーはマイク・オンを申請することができなくなります。
リスナーはマイク・オンを申請して、マイク・オンのキャスターになり、他者と音声インタラクションを行うことができます。また、いつでもマイク・オフにして、通常のリスナーになることも可能です。
各種のテキストメッセージやカスタマイズメッセージの発信をサポートし、カスタムメッセージは弾幕、「いいね」、ギフトなどに使用することときます。
説明：
TUIKitシリーズコンポーネントはTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期的にアクティブにすることができます。IMサービスの課金ルールの詳細については、Instant Messagingの料金説明をご参照ください。TRTCをアクティブにすると、デフォルトでは、100DAUまでサポートするIM SDK体験版もアクティブにします。
TRTCVoiceRoomは、オープンソースのClassであり、Tencent Cloudの2つのクローズソースであるSDKに依存しています。具体的な実装プロセスは ボイスチャットルーム (iOS)をご参照ください。
TRTC SDK：TRTC SDKを低遅延のボイスチャットコンポーネントとして使用しています。
IM SDK：IM SDKのAVChatroomを使用してチャットルーム機能を実装します。同時にIMの属性インターフェースによって、マイクリストなどのルーム情報を保存し、招待シグナリングはマイク・オン/ピックのリクエストに用いることができます。

TRTCVoiceRoom API 概要

SDK基本関数

API	説明
sharedInstance	シングルトンオブジェクトを取得します。
destroySharedInstance	シングルトンオブジェクトを廃棄します。

setDelegate	イベントコールバックを設定します。
setDelegateHandler	イベントのコールバックが配置されているスレッドを設定します。
login	ログイン。
logout	ログアウト。
setSelfProfile	個人情報を修正します。

ルーム関連インターフェース関数

API	説明
createRoom	ルームの作成（管理者が呼び出し）。ルームが存在しない場合は、システムが新しいルームを自動的に作成します。
destroyRoom	ルームの破棄（管理者が呼び出し）。
enterRoom	入室（リスナーが呼び出し）。
exitRoom	退室（リスナーが呼び出し）。
getRoomInfoList	ルームリストの詳細情報を取得します。
getUserInfoList	指定されたuserIdのユーザー情報を取得します。nilの場合は、ルーム内全員の情報を取得します。

マイク管理インターフェース

API	説明
enterSeat	ユーザーが発言者になります（リスナー側/管理者ともに呼び出し可）。
moveSeat	マイクの移動 (マイク・オンするキャスター側で呼び出せます)。
leaveSeat	ユーザーが視聴者になります（キャスターが呼び出し）。
pickSeat	視聴者が発言できるように招待（管理者が呼び出し）。
kickSeat	キックアウトしてマイク・オフ（管理者が呼び出し）。
muteSeat	任意のマイクのミュート/ミュート解除（管理者が呼び出し）。
closeSeat	任意のマイクのクローズ/解除（管理者が呼び出し）。

ローカルのオーディオ操作インターフェース

API	説明
startMicrophone	マイクの集音開始。
stopMicrophone	マイクの集音停止。
setAudioQuality	音質の設定。
muteLocalAudio	ローカルオーディオミュートの開始/停止。
setSpeaker	スピーカーの起動設定。
setAudioCaptureVolume	マイクの集音音量設定。
setAudioPlayoutVolume	再生音量の設定。
setVoiceEarMonitorEnable	インイヤーモニタリングのオン/オフ。

リモートユーザー音声操作インターフェース

API	説明
muteRemoteAudio	指定メンバーをミュート/ミュート解除。
muteAllRemoteAudio	全メンバーをミュート/ミュート解除。

BGMサウンドエフェクト関連インターフェース

API	説明
getAudioEffectManager	BGMサウンドエフェクト管理オブジェクトTXAudioEffectManagerを取得します。

メッセージ送信関連インターフェース

API	説明
sendRoomTextMsg	ルーム内でのテキストメッセージのブロードキャスト。通常、弾幕によるチャットに使用します。
sendRoomCustomMsg	カスタマイズしたテキストメッセージを送信します。

招待シグナリング関連インターフェース

API	説明

sendInvitation	ユーザーに招待を送信。
acceptInvitation	招待の同意。
rejectInvitation	招待の辞退。
cancelInvitation	招待の取り消し。

TRTCVoiceRoomDelegate API概要

一般的なイベントコールバック

API	説明
onError	エラーのコールバック。
onWarning	警告のコールバック。
onDebugLog	Logコールバック。

ルームイベントのコールバック

API	説明
onRoomDestroy	ルームが廃棄された時のコールバック。
onRoomInfoChange	ボイスチャットルーム情報変更のコールバック。
onUserVolumeUpdate	ユーザー通話音量のコールバック。

マイク変更コールバック

API	説明
onSeatListChange	全量のマイクリストの変更。
onAnchorEnterSeat	発言者のメンバーがいます（ユーザーが発言者になります/管理者が視聴者を発言できるように招待）。
onAnchorLeaveSeat	視聴者のメンバーがいます（ユーザーが視聴者になる/管理者がキックアウトしてマイク・オフ）。
onSeatMute	管理者のマイクミュート。
onUserMicrophoneMute	ユーザーのマイクがミュートされているかどうか。

onSeatClose	管理者のマイククローズ。

リスナーの入退室イベントのコールバック

API	説明
onAudienceEnter	リスナー入室通知の受信。
onAudienceExit	リスナー退室通知の受信。

メッセージイベントのコールバック

API	説明
onRecvRoomTextMsg	テキストメッセージを受信します。
onRecvRoomCustomMsg	カスタムメッセージを受信します。

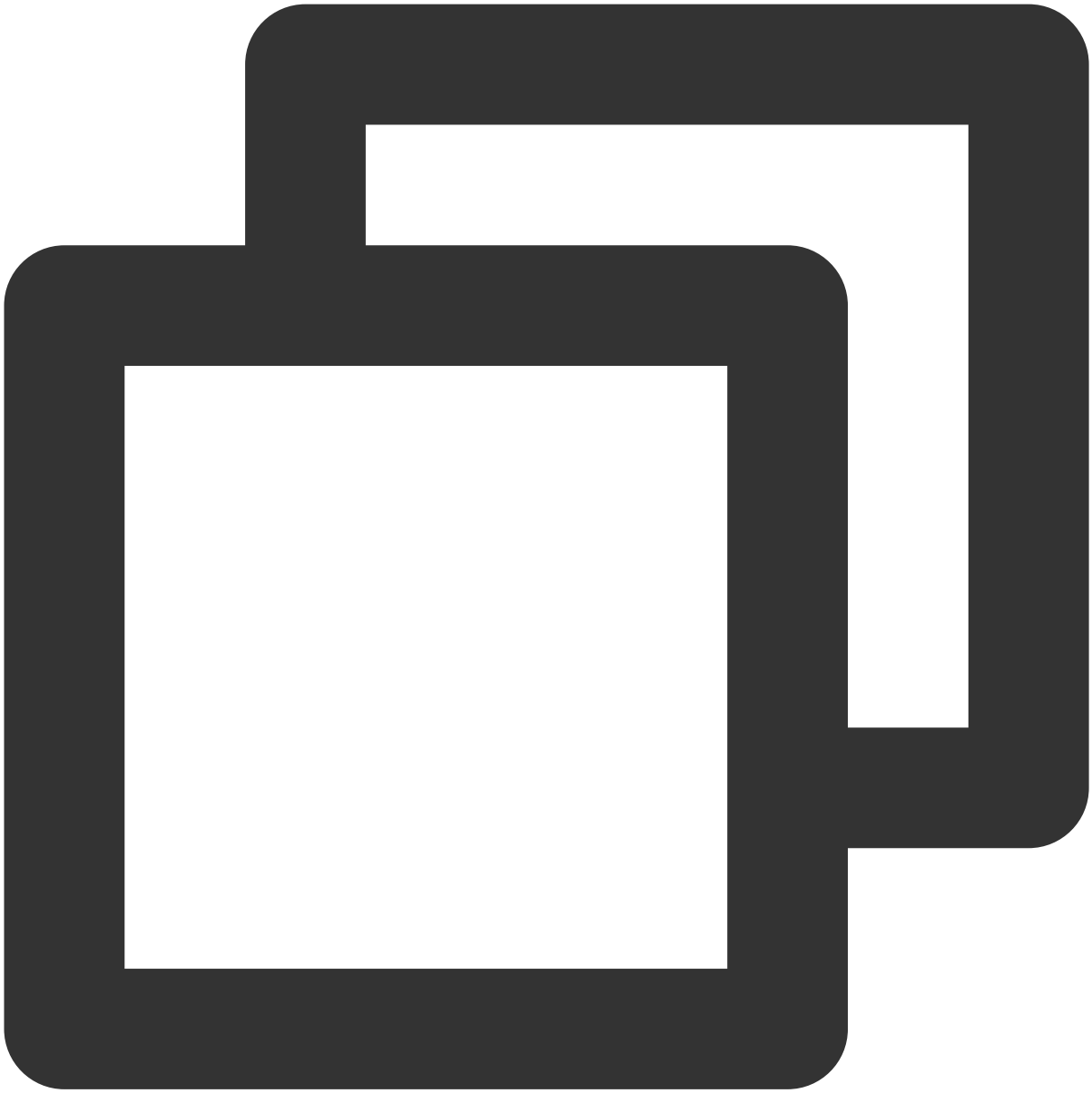
シグナリングイベントのコールバック

API	説明
onReceiveNewInvitation	新規招待リクエストを受信。
onInviteeAccepted	被招待者が招待に同意。
onInviteeRejected	被招待者が招待を拒否。
onInvitationCancelled	招待者が招待を取り消し。

SDK基本関数

sharedInstance

TRTCVoiceRoom シングルトンオブジェクトを取得します。



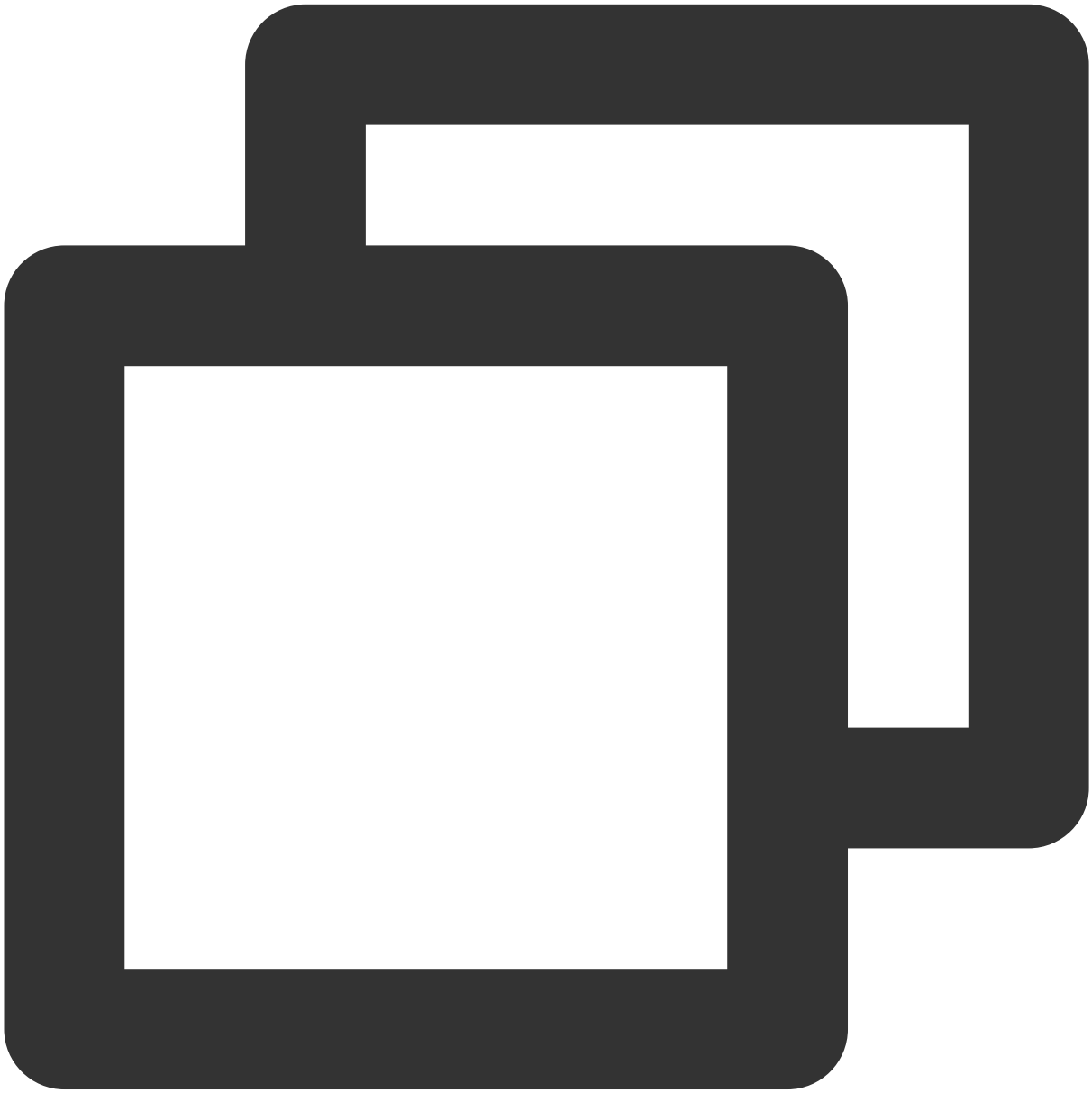
```
/**
* TRTCVoiceRoom シングルトンオブジェクトの取得
*
* - returns: TRTCVoiceRoom インスタンス
* - note: {@link TRTCVoiceRoom#destroySharedInstance()}を呼び出してシングルトンオブジェク
*/
+ (instancetype)sharedInstance NS_SWIFT_NAME(shared());
```

destroySharedInstance

TRTCVoiceRoom シングルトンオブジェクトを破棄します。

説明：

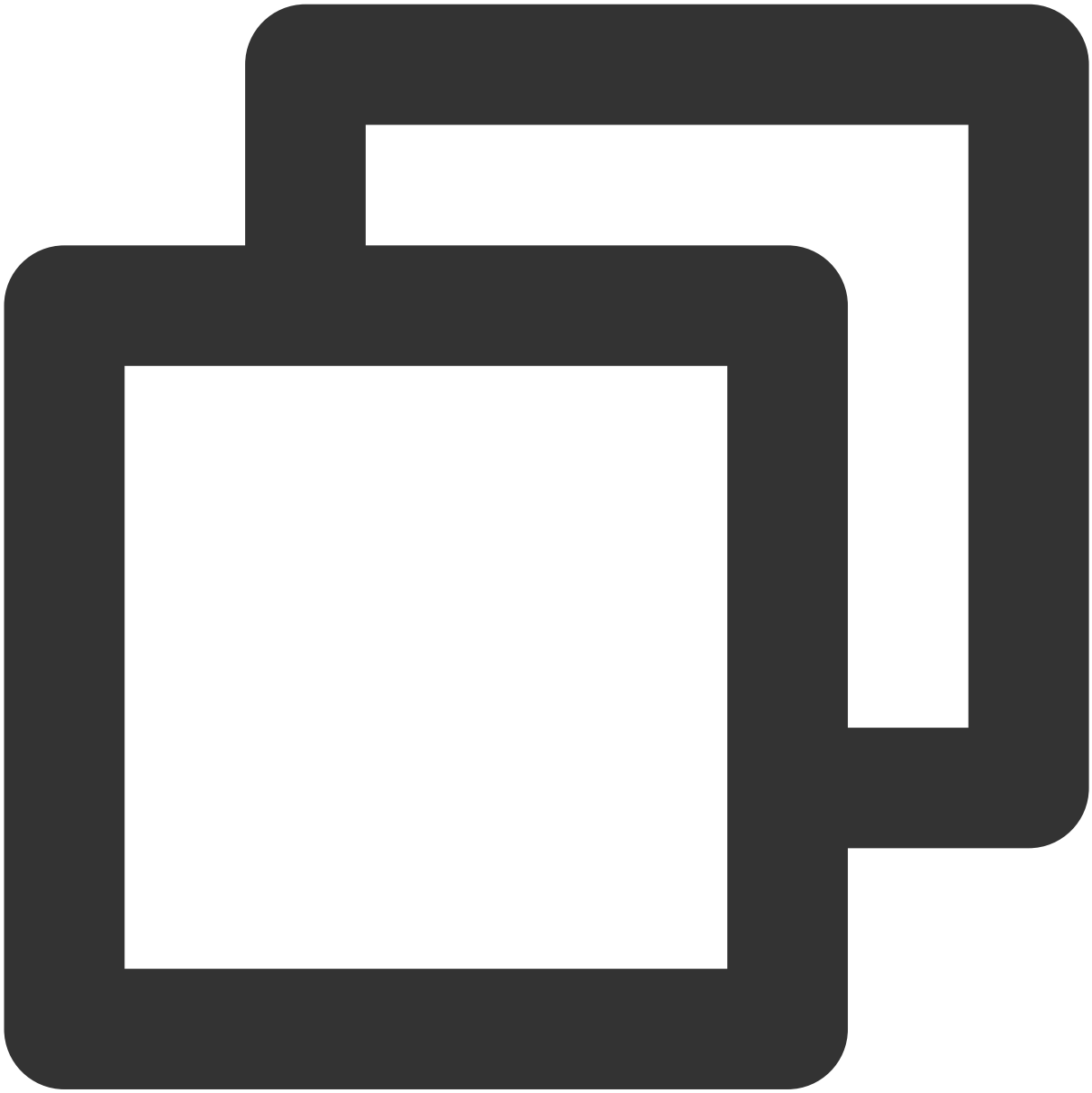
インスタンスの破棄後は、外部にキャッシュされた TRTCVoiceRoomインスタンスの再使用ができません。改めて sharedInstance を呼び出し、インスタンスを新規取得する必要があります。



```
/**
* TRTCVoiceRoom のシングルトンオブジェクトの破棄
*
* - note: インスタンスの破棄後は、外部にキャッシュされたTRTCVoiceRoomインスタンスの再使用ができき
*/
+ (void)destroySharedInstance NS_SWIFT_NAME(destroyShared());
```

setDelegate

TRTCVoiceRoom イベントコールバック、 TRTCVoiceRoomDelegate を介して TRTCVoiceRoom の各種ステータス通知を受け取ることができます。



```
/**
* コンポーネントコールバックインターフェースの設定
*
* TRTCVoiceRoomDelegate によって TRTCVoiceRoom の各種ステータス通知を取得することができます
*
```

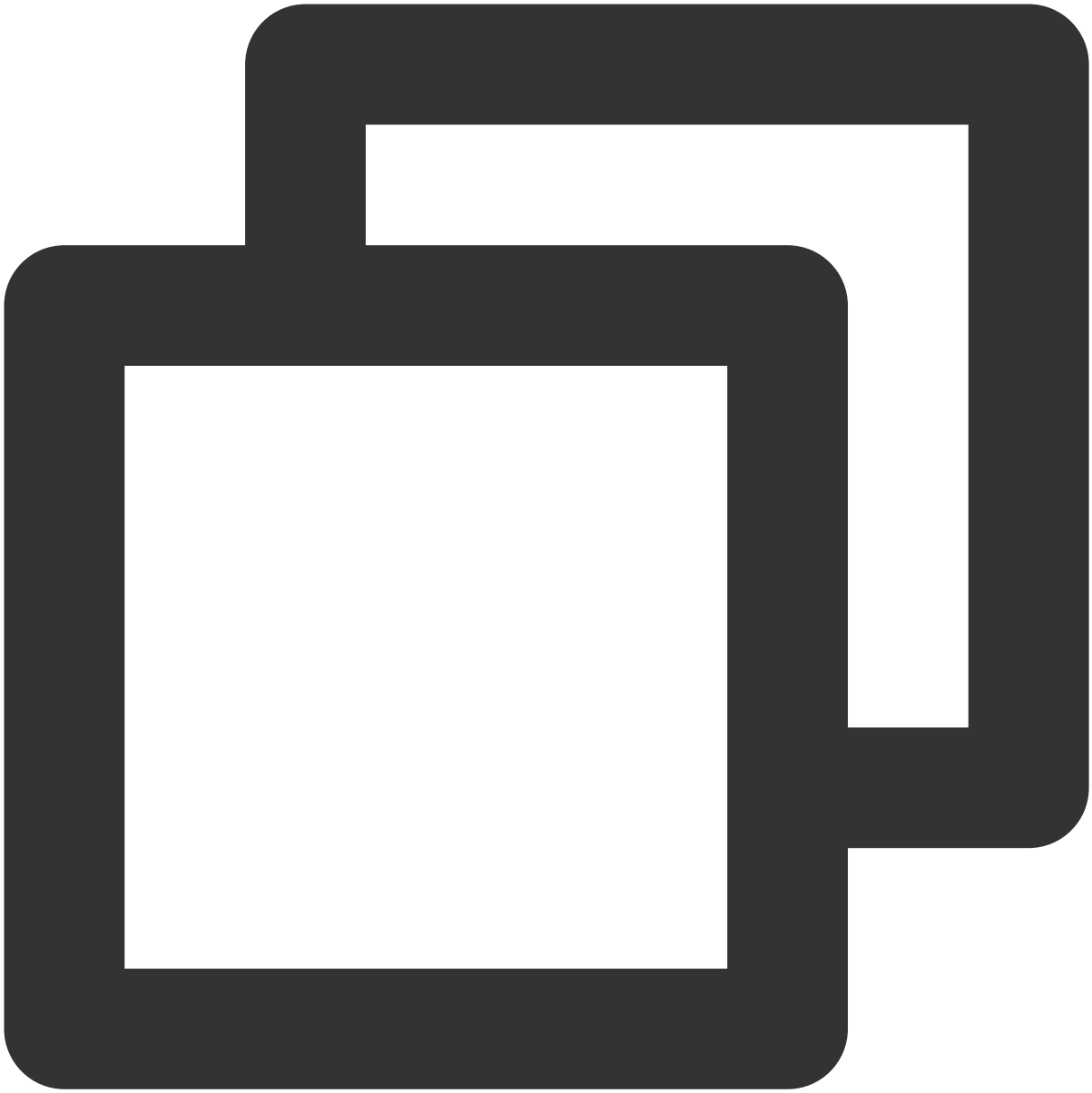
```
 * - parameter delegateコールバックインターフェース
 * - note: TRTCVoiceRoomのコールバックイベントは、デフォルトではMain Queueでコールバックされま
 */
- (void)setDelegate:(id<TRTCVoiceRoomDelegate>)delegate NS_SWIFT_NAME(setDelegate(d
```

説明：

setDelegate は、 TRTCVoiceRoom のプロキシコールバックです。

setDelegateQueue

イベントコールバックが所在するスレッドキューを設定し、デフォルトはメインスレッド MainQueue に送信します。



```
/**
* イベントコールバックが配置されているキューの設定
*
* - parameter queueキュー，TRTCVoiceRoomの各種ステータス通知コールバックは、指定したqueueに発
*/
- (void)setDelegateQueue:(dispatch_queue_t)queue NS_SWIFT_NAME(setDelegateQueue(que
```

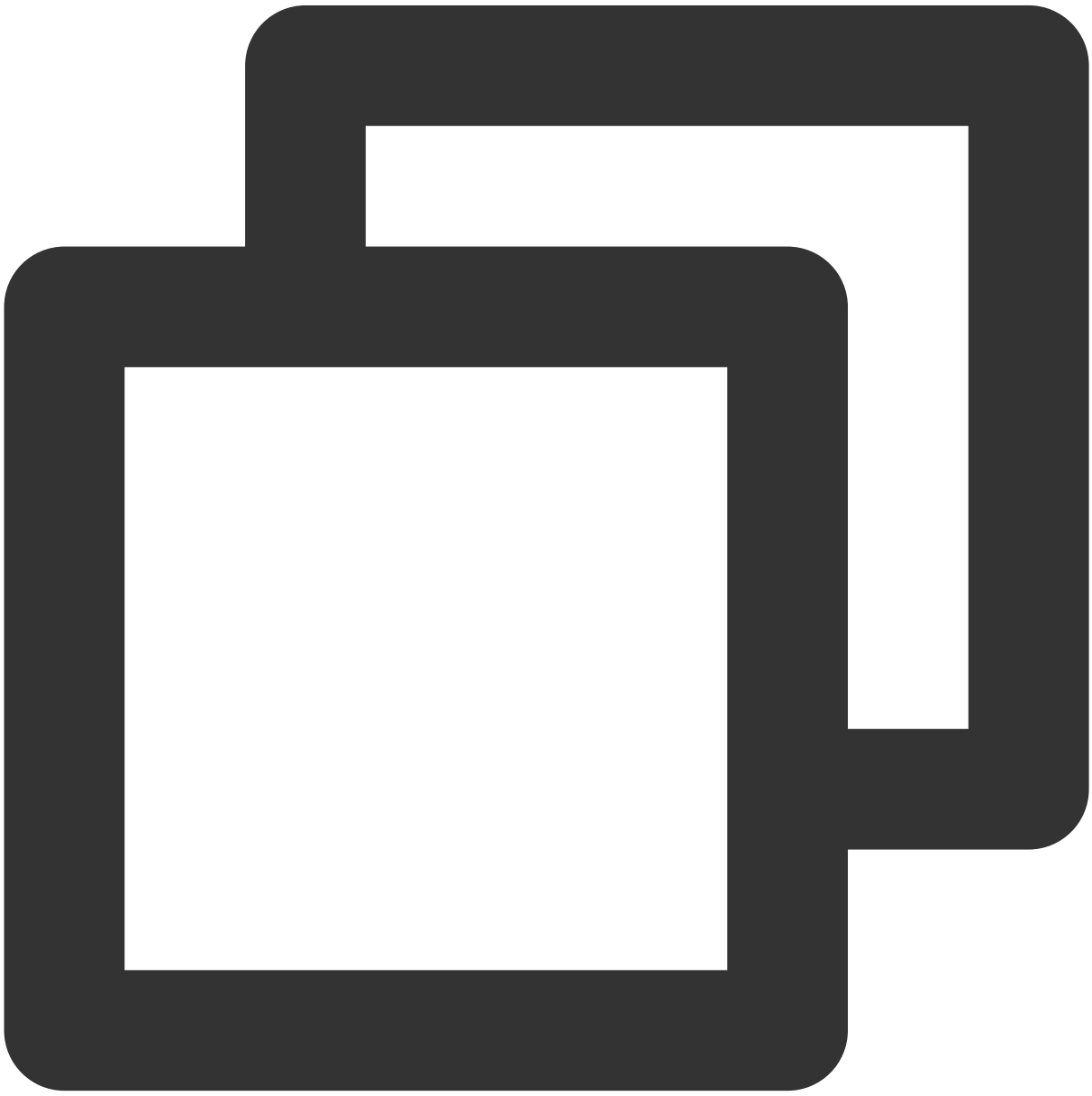
パラメータは下表に示すとおりです：

パラメー	タイプ	意味

タ		
queue	dispatch_queue_t	TRTCVoiceRoomの各種ステータス通知は、指定したスレッドキューに発信されます。

login

ログイン。



```
- (void)login:(int)sdkAppID
      userId:(NSString *)userId
```

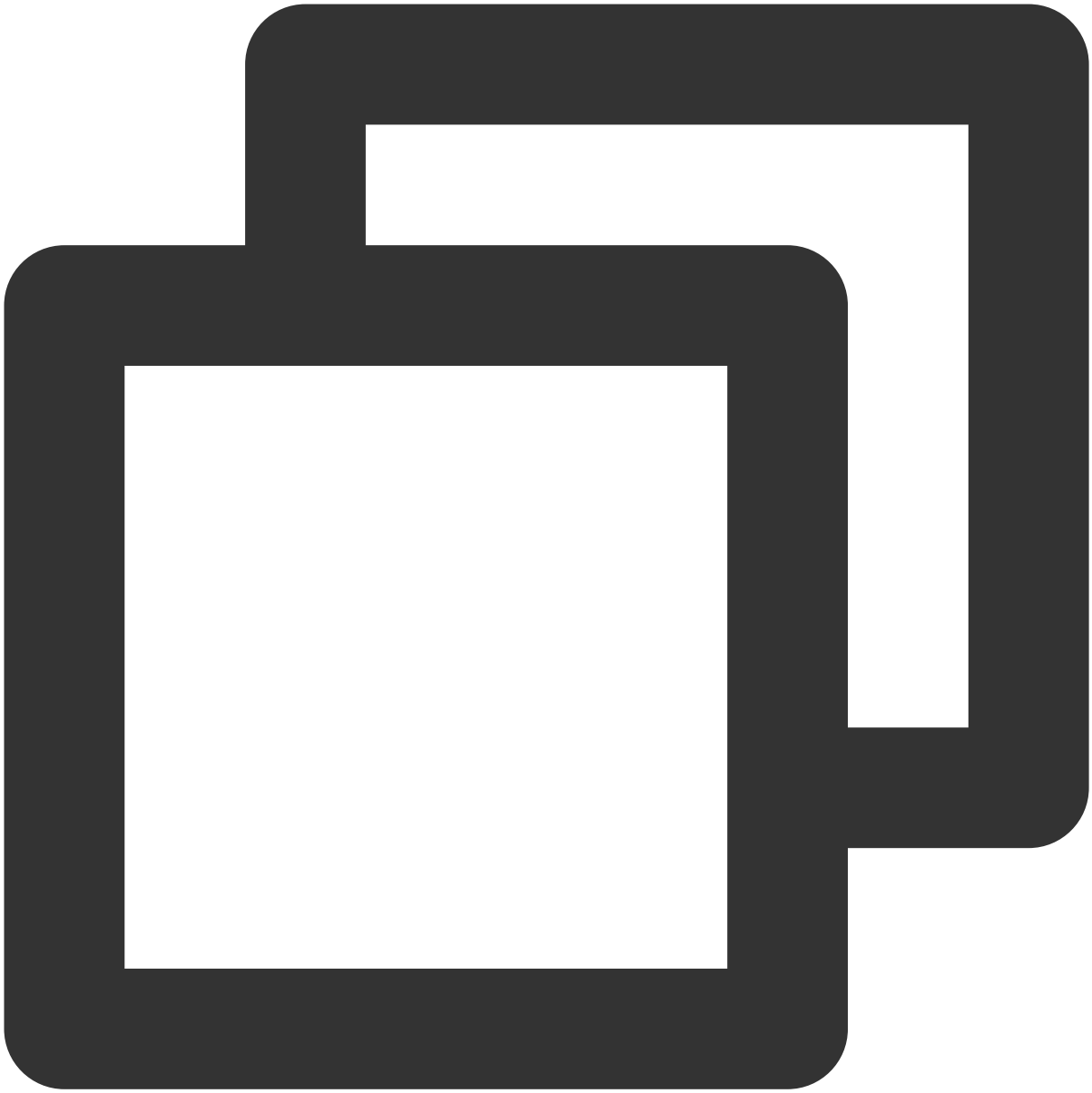
```
        userSig:(NSString *)userSig
      callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(login(sdkAppID:userI
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
sdkAppId	int	**TRTCコンソール >**アプリケーション管理> アプリケーション情報でSDKAppIDを確認できます。
userId	NSString	現在のユーザーID。文字列タイプでは、英語のアルファベット（a-zとA-Z）、数字（0-9）、ハイフン（-）とアンダーライン（_）のみ使用できます。
userSig	NSString	Tencent Cloudによって設計されたセキュリティ保護署名。取得方法については、 UserSigの計算、使用方法をご参照ください。
callback	ActionCallback	ログインのコールバック。成功時にcodeは0になります。

logout

ログアウト。



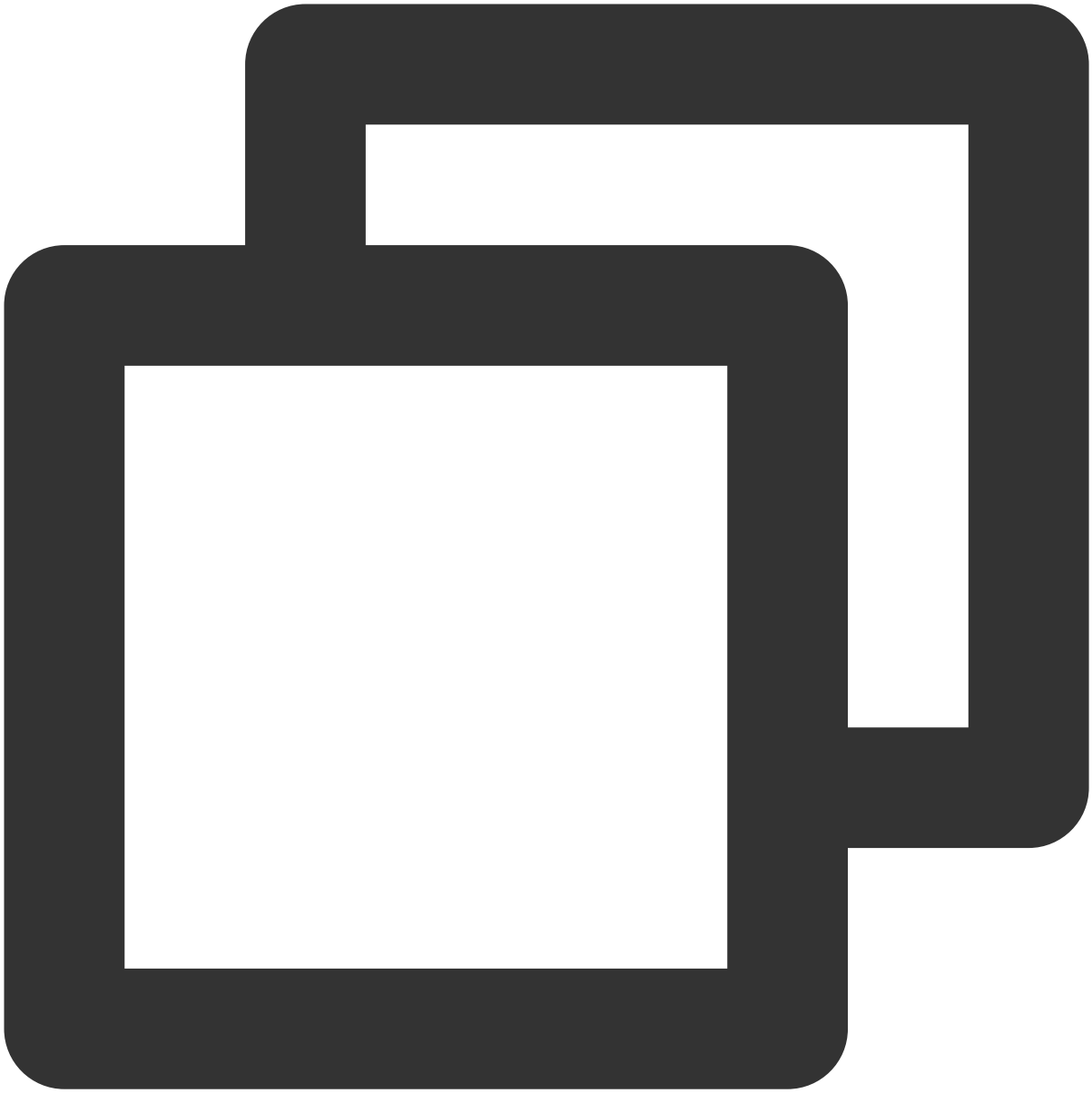
```
- (void)logout:(ActionCallback _Nullable)callback NS_SWIFT_NAME(logout(callback:));
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
callback	ActionCallback	ログアウトのコールバック。成功時にcodeは0になります。

setSelfProfile

個人情報の修正。



```
- (void)setSelfProfile:(NSString *)userName avatarURL:(NSString *)avatarURL callbac
```

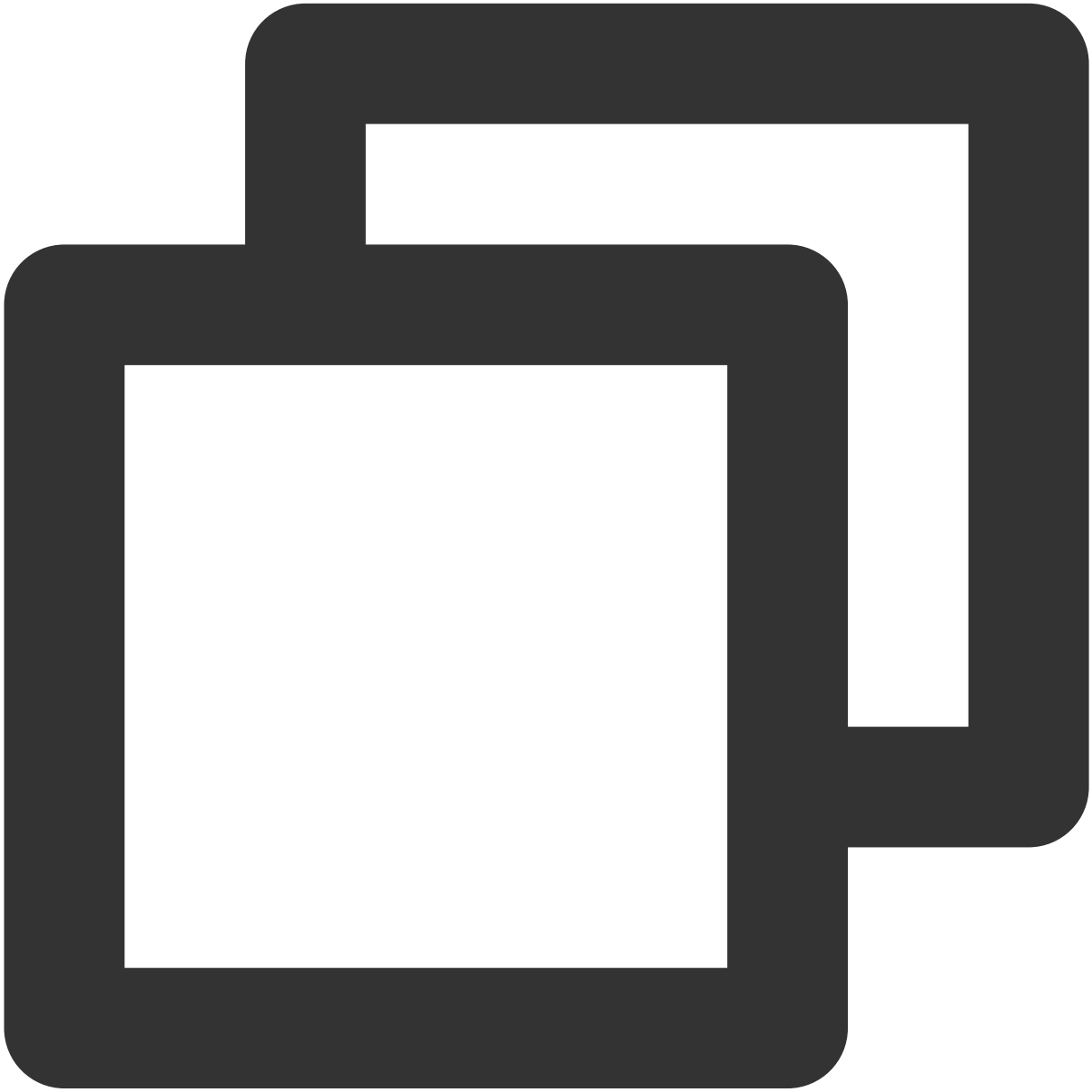
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userName	NSString	ニックネーム。
avatarURL	NSString	プロフィール画像のアドレス。
callback	ActionCallback	個人情報設定のコールバック。成功時にcodeは0になります。

ルーム関連インターフェース関数

createRoom

ルームの作成（管理者が呼び出し）。



```
- (void)createRoom:(int)roomID roomParam:(VoiceRoomParam *)roomParam callback:(Acti
```

パラメータは下表に示すとおりです：

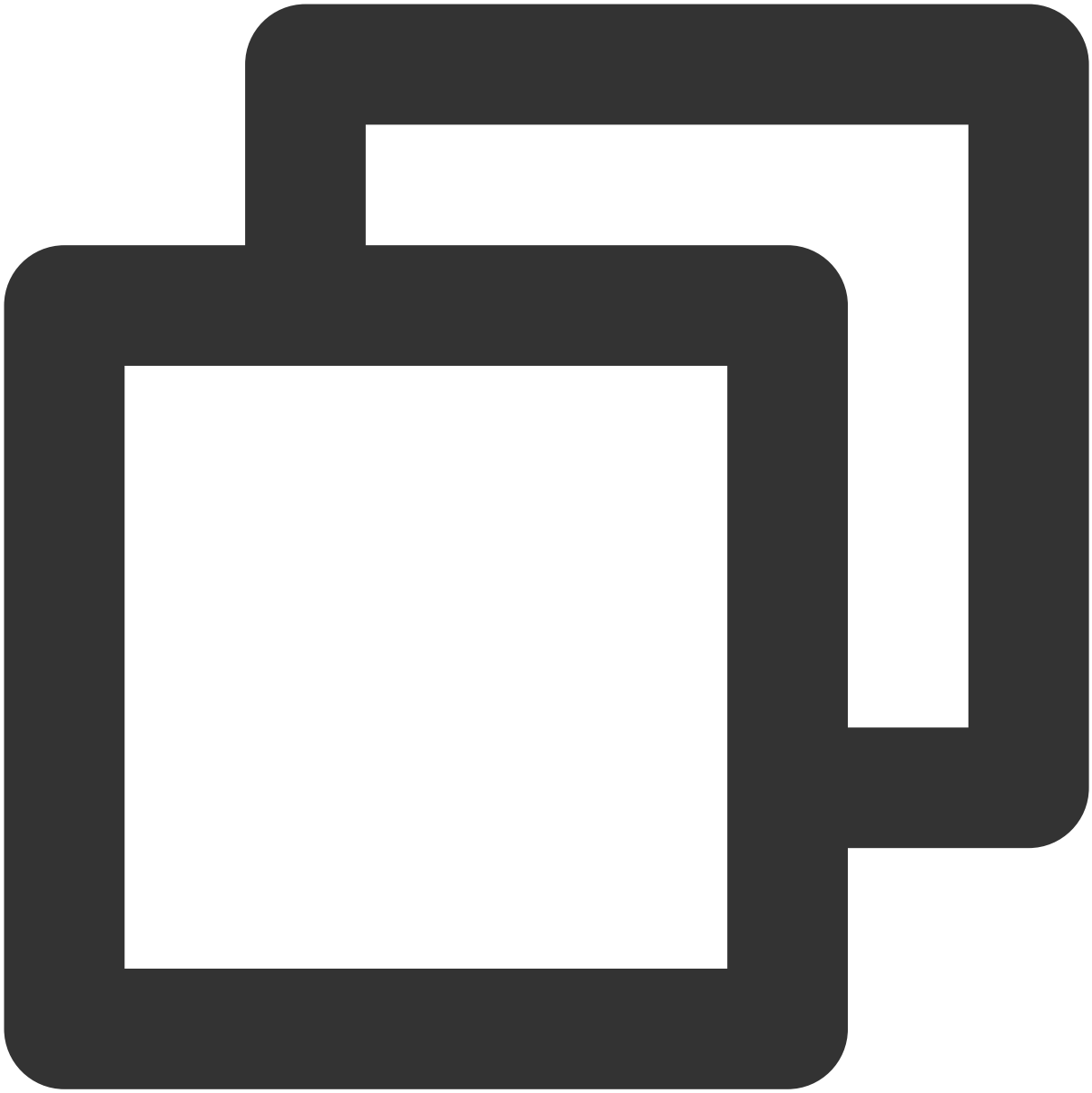
パラメータ	タイプ	意味

roomId	int	ルームIDです。ご自身でアサインし、一元管理してください。複数のroomIDを、1つのボイスチャットルームリストにまとめることができます。Tencent Cloudでは現在、ボイスチャットルームリストの管理サービスを行っていませんので、ご自身でボイスチャットルームリストを管理してください。
roomParam	VoiceRoomParam	ルーム情報は、ルーム名、マイク情報、カバー情報など、ルームを説明するために用いる情報に使用します。マイク管理が必要な場合は、ルームのマイク数を入力してください。
callback	ActionCallback	ルームの新規作成結果のコールバック。成功時にcodeは0になります。

管理者が配信を開始する際の通常の呼び出しプロセスは次のとおりです：
1. 管理者は、 `createRoom` を呼び出して新しいボイスチャットルームを作成します。この時にルームID、マイク・オンに管理者の確認の要否、マイク数などルームのプロパティ情報を渡します。
2. 管理者は、ルーム作成に成功した後、 `enterSeat` を呼び出して参加します。
3. 管理者は、コンポーネントの `onSeatListChange` マイクリスト変更イベント通知を受信します。この時、マイクリスト変更をUI上に更新することができます。
4. 管理者は、マイクリストのメンバーが参加した `onAnchorEnterSeat` というイベント通知も受信します。この時、マイク集音は自動的に開始されます。

destroyRoom

ルームの破棄（管理者が呼び出し）。管理者は、ルーム作成後、この関数を呼び出してルームを破棄します。



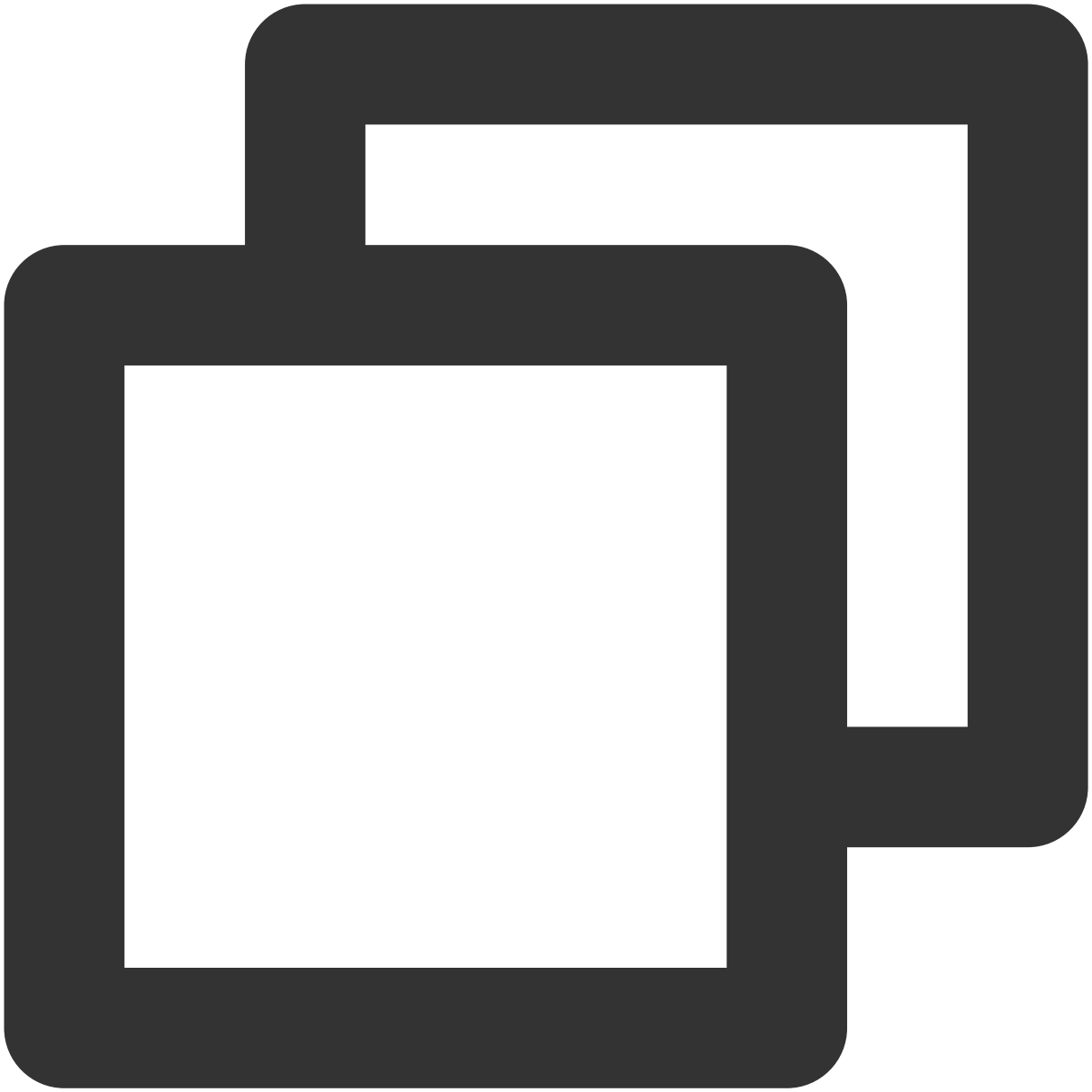
```
- (void)destroyRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(destroyRoom(ca
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
callback	ActionCallback	ルームの廃棄結果のコールバック。成功時にcodeは0になります。

enterRoom

入室（リスナーが呼び出し）。



```
- (void)enterRoom:(NSInteger)roomID callback:(ActionCallback _Nullable)callback NS_
```

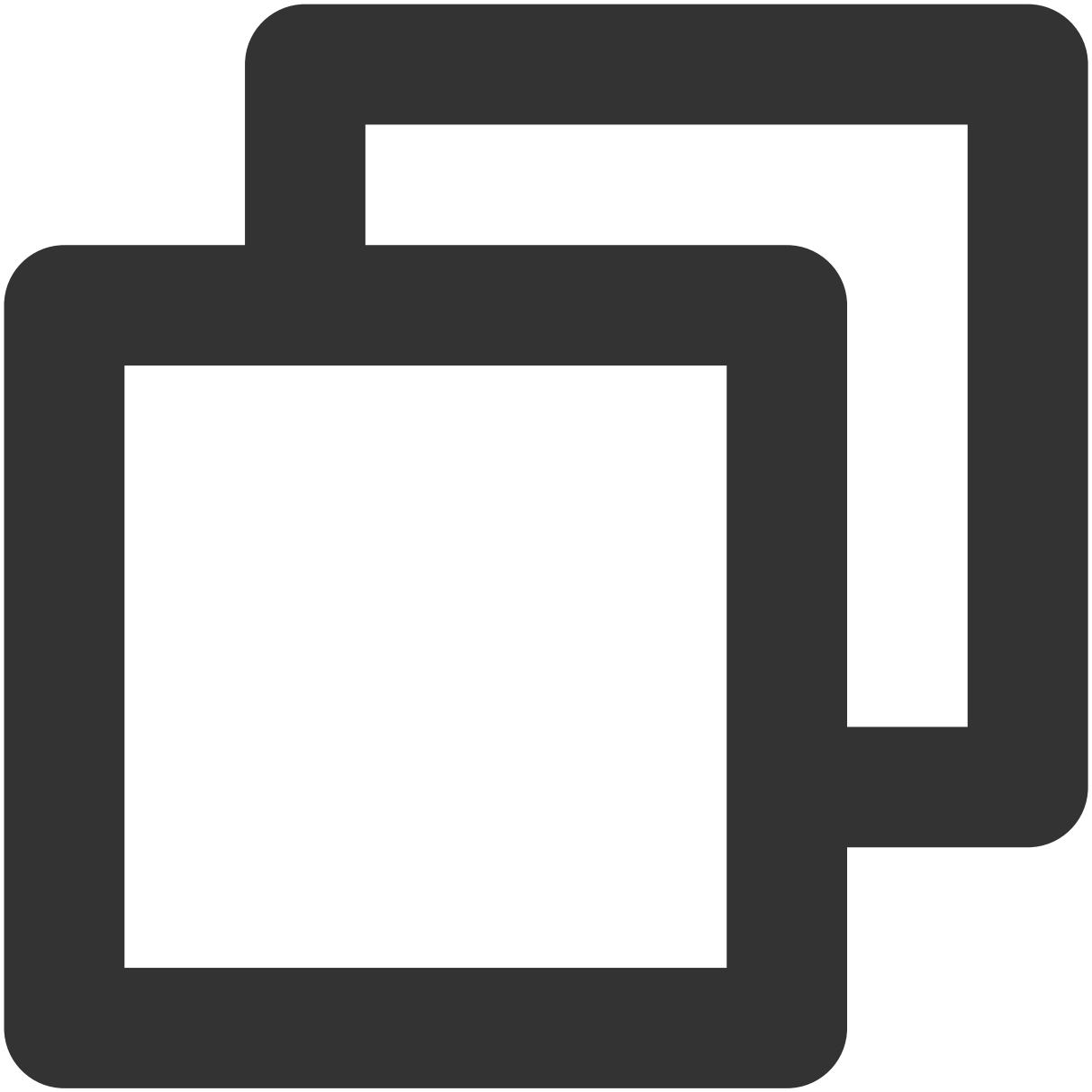
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
roomId	NSInteger	ルームID。
callback	ActionCallback	入室結果のコールバック。成功時にcodeは0になります。

リスナーが入室し聴取する際の通常の呼び出しプロセスは次のとおりです：
1. リスナーがサーバーから取得する最新のボイスチャットルームリストには、多くのボイスチャットルームのroomIdおよびルーム情報が含まれる場合があります。
2. リスナーが1つのボイスチャットルームを選択し、`enterRoom`を呼び出してルームナンバーを渡すと、そのルームに参加できます。
2. 入室後、コンポーネントの`onRoomInfoChange`ルーム属性変更イベント通知を受信します。この時、ルーム属性を記録し、それに応じた修正を行うことができます。例：UIに表示するルーム名、発言者にする際の管理者への同意リクエストの要否の記録など。
3. 入室後は、コンポーネントの`onSeatListChange`マイクリスト変更イベント通知を受信します。この時にマイクリストの変更をUI上に更新することができます。
4. 入室後にマイクリストにキャスターが参加した`onAnchorEnterSeat`のイベント通知も受信します。

exitRoom

ルームから退出します。



```
- (void)exitRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(exitRoom(callback
```

パラメータは下表に示すとおりです：

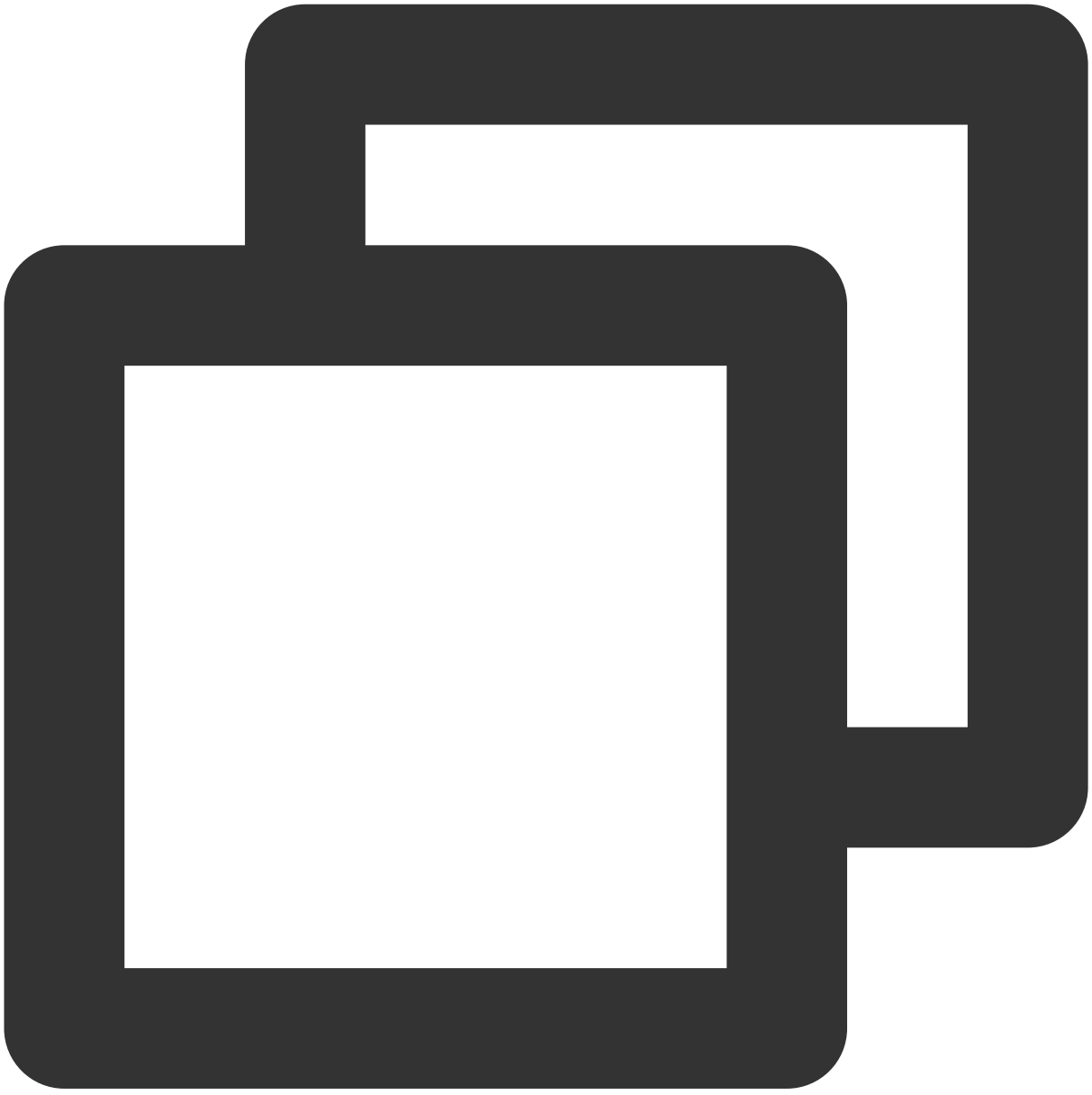
パラメータ	タイプ	意味
callback	ActionCallback	退室結果のコールバック。成功時にcodeは0になります。

getRoomInfoList

ルームリストの詳細情報を取得します。このうち、ルーム名、ルームカバーは、管理者が `createRoom()` 作成時にroomInfoによって設定したものになります。

説明：

ルームリストおよびルーム情報をご自身で管理する場合は、この関数は無視できます。



```
- (void)getRoomInfoList:(NSArray<NSNumber *> *)roomIdList callback:(VoiceRoomInfoCa
```

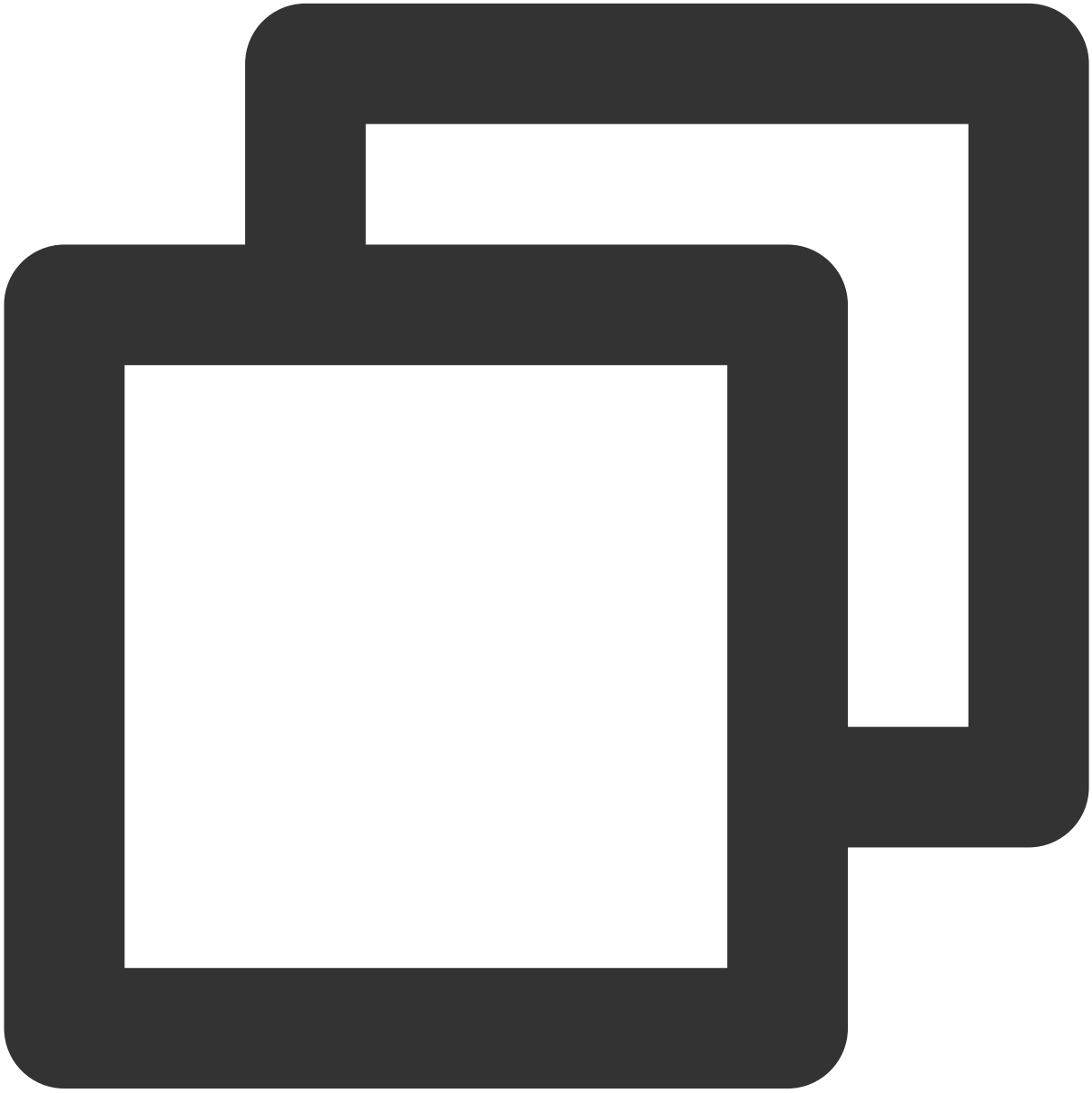
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味

roomIdList	NSArray<NSNumber>	ルームナンバーリスト。
callback	RoomInfoCallback	ルーム詳細情報のコールバック。

getUserInfoList

指定されたuserIdのユーザー情報を取得します。



```
- (void)getUserInfoList:(NSArray<NSString *> * _Nullable)userIDList callback:(Voice
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userIdList	NSArray<NSString>	取得すべきユーザーIDリスト。nullの場合は、ルーム内全員の情報を取得します。
userlistcallback	UserListCallback	ユーザーの詳細情報のコールバック。

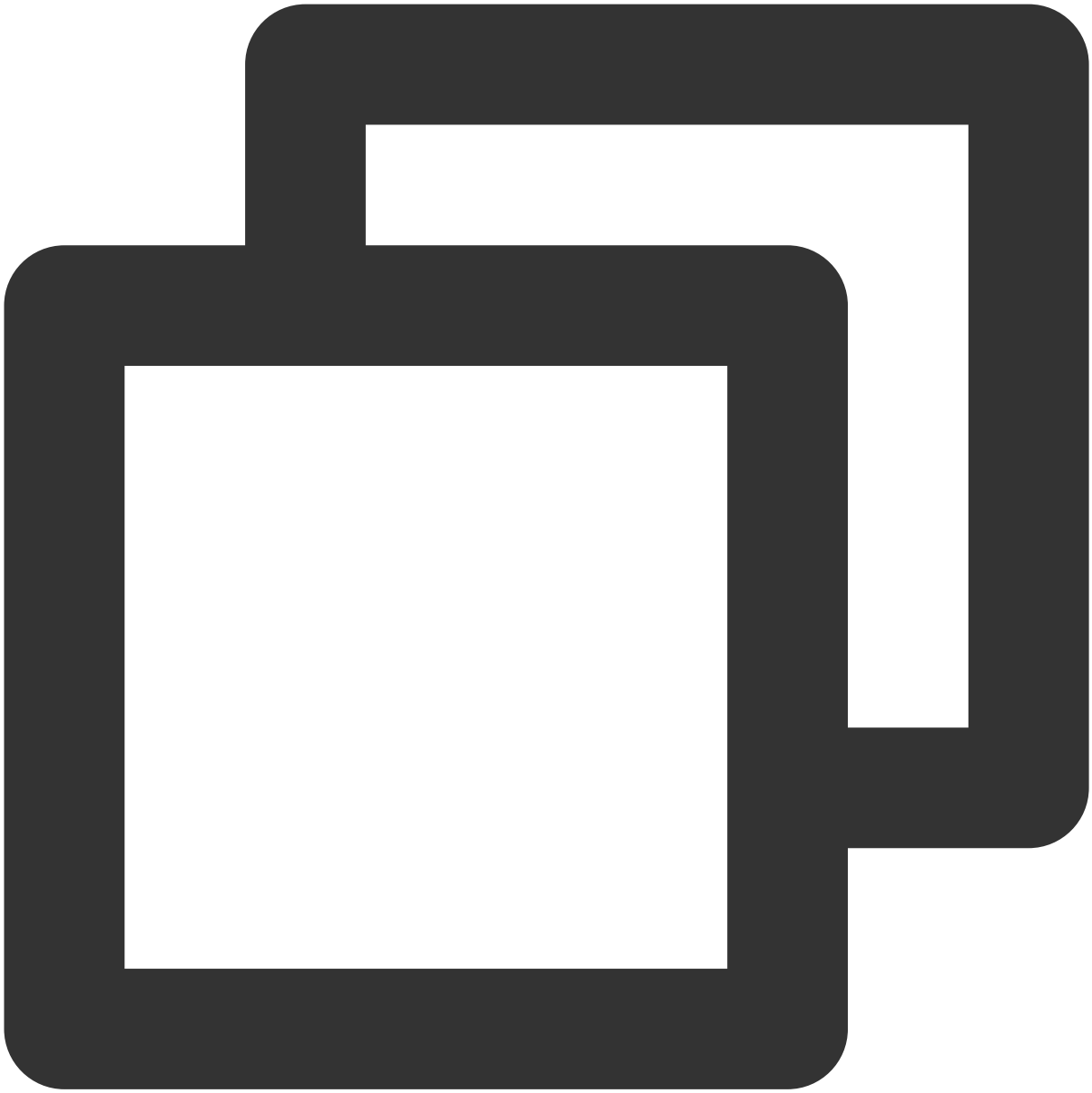
マイク管理インターフェース

enterSeat

ユーザーが発言者になります（リスナー側/管理者ともに呼び出し可）。

説明：

マイク・オンの成功後、ルーム内の全メンバーは、`onSeatListChange` および `onAnchorEnterSeat` というイベント通知を受信します。



```
- (void)enterSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback
```

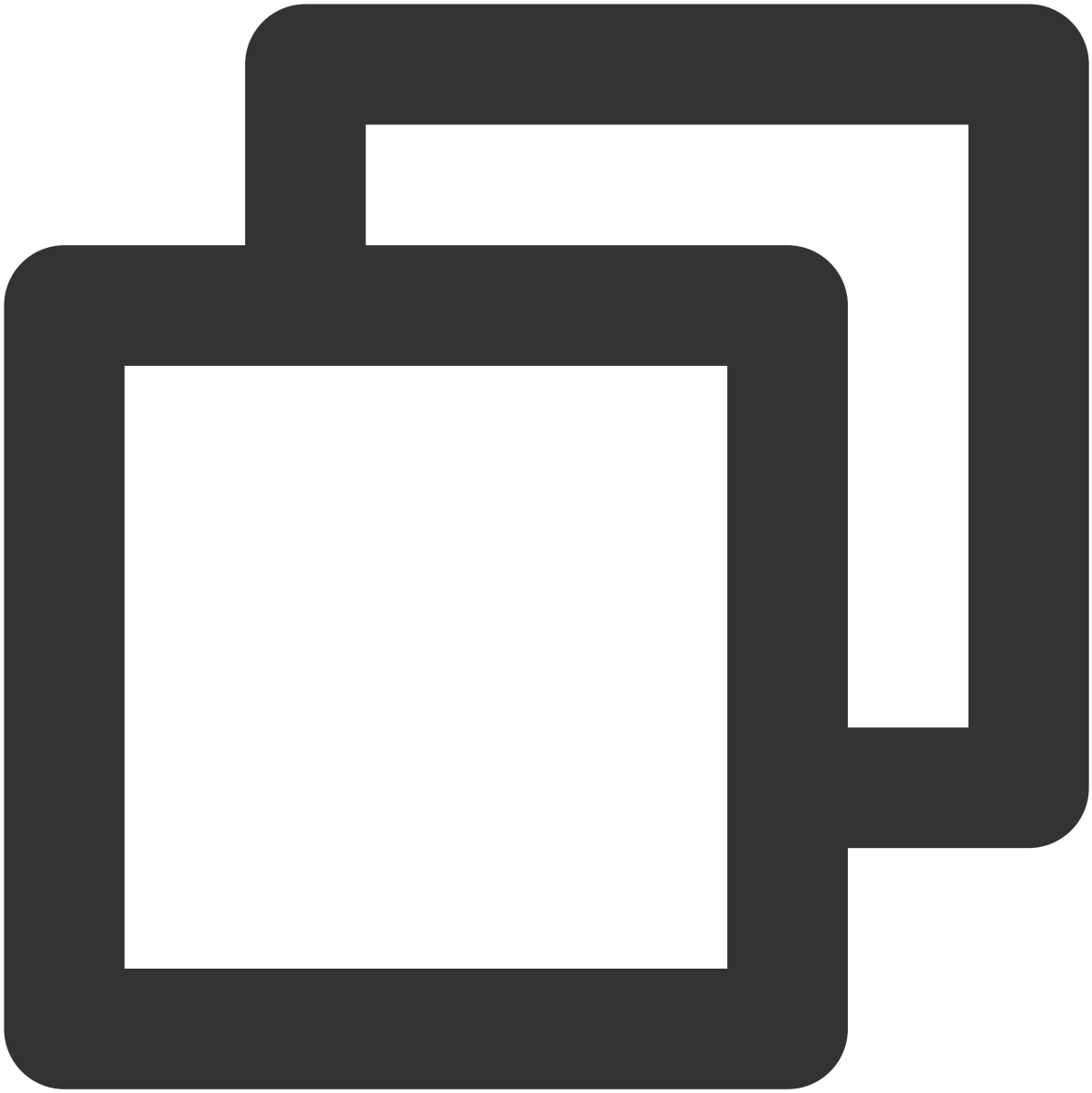
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
seatIndex	NSInteger	マイク・オンの必要があるマイク番号。
callback	ActionCallback	操作コールバック。

このインターフェースを呼び出すと、直ちにマイクリストが変更されます。リスナーによるマイク・オンの申請に管理者の同意が必要となるケースの場合は、まず `sendInvitation` を呼び出してから管理者に申請し、 `onInvitationAccept` を受信するとこの関数を呼び出せるようになります。

moveSeat

マイクの移動 (マイク・オンするキャスター側で呼び出せます)。
説明：
マイクの移動に成功したら、ルーム内のすべてのメンバーが `onSeatListChange` 、 `onAnchorLeaveSeat` 、 `onAnchorEnterSeat` のイベント通知を受け取ります。(キャスターが呼び出した後に変更できるのは、マイクのシート番号情報のみで、ユーザーのキャスターロールを切り替えることはできません。)



```
- (NSInteger)moveSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callb
NS_SWIFT_NAME(moveSeat(seatIndex:callback:))
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
seatIndex	NSInteger	移動の必要があるマイク番号。
callback	ActionCallback	操作コールバック。

戻り値：

戻り値	タイプ	意味
code	NSInteger	マイク移動操作の結果（0は成功、その他は失敗、10001はインターフェース呼び出し頻度制限）。

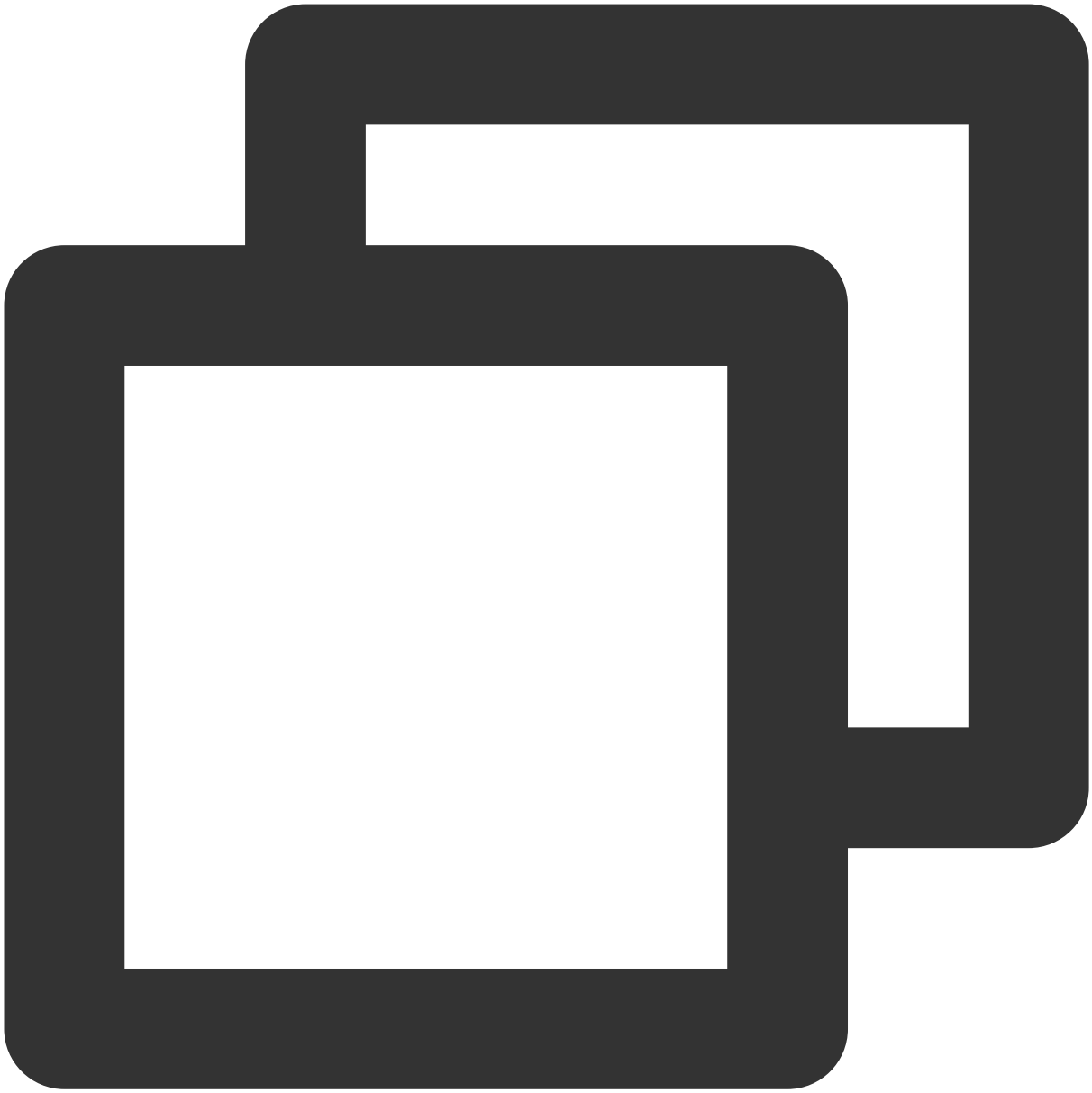
このインターフェースを呼び出すと、直ちにマイクリストが変更されます。リスナーによるマイク・オンの申請に管理者の同意が必要となるケースの場合は、まず `sendInvitation` を呼び出してから管理者に申請し、 `onInvitationAccept` を受信するとこの関数を呼び出せるようになります。

leaveSeat

ユーザーが視聴者になります（キャスターが呼び出し）。

説明：

マイク・オフの成功後、ルーム内の全メンバーは、 `onSeatListChange` および `onAnchorLeaveSeat` というイベント通知を受信します。



```
- (void)leaveSeat:(ActionCallback _Nullable)callback NS_SWIFT_NAME(leaveSeat(callba
```

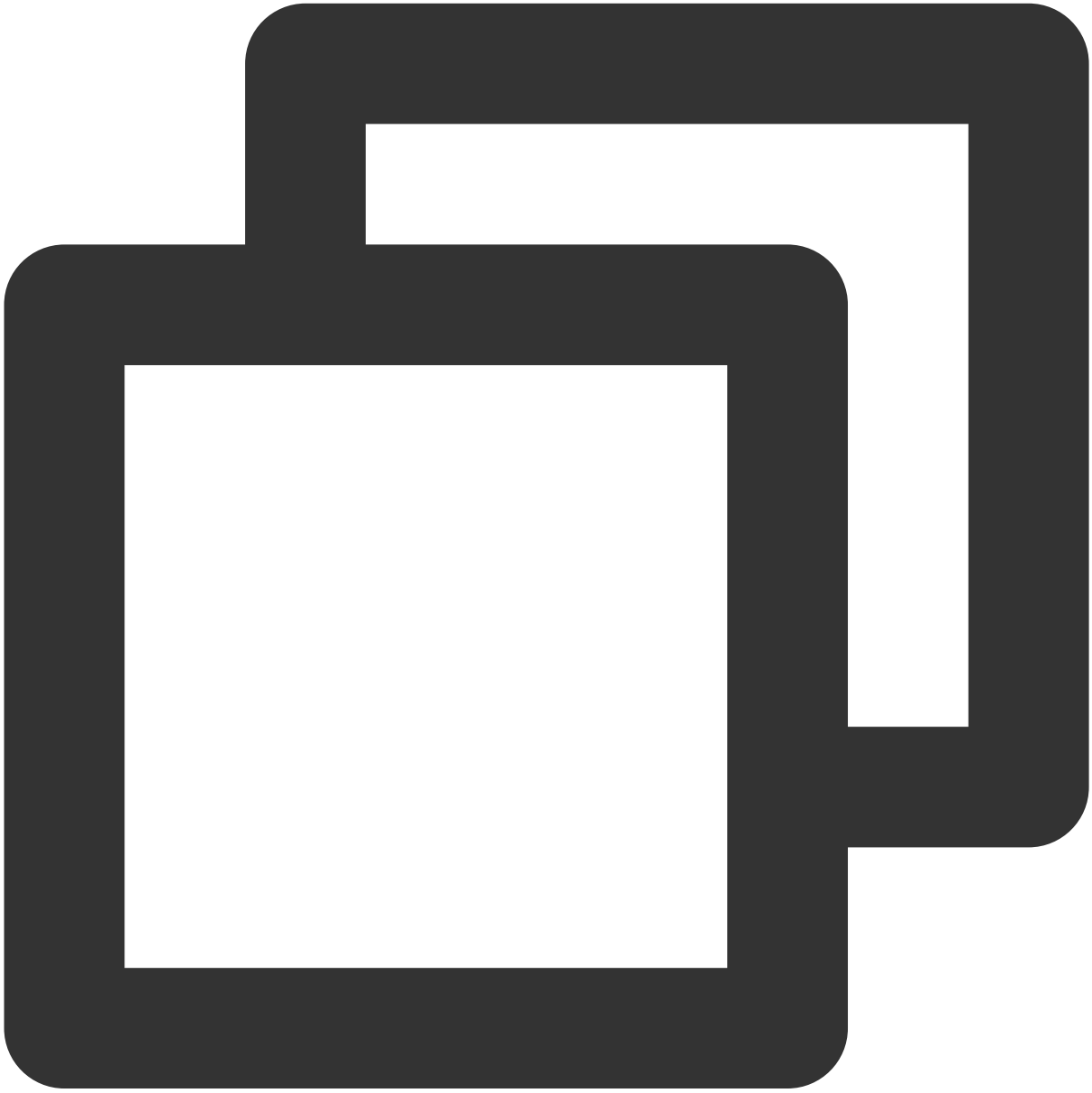
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
callback	ActionCallback	操作コールバック。

pickSeat

視聴者が発言できるように招待（管理者が呼び出し）。

説明：
管理者が視聴者を発言できるように招待すると、ルーム内の全メンバーは、`onSeatListChange` と `onAnchorEnterSeat` というイベント通知を受信します。



```
- (void)pickSeat:(NSInteger)seatIndex userId:(NSString *)userId callback:(ActionCal
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
seatIndex	NSInteger	マイク・オンの必要があるマイク番号。

userId	NSString	ユーザーID。
callback	ActionCallback	操作コールバック。

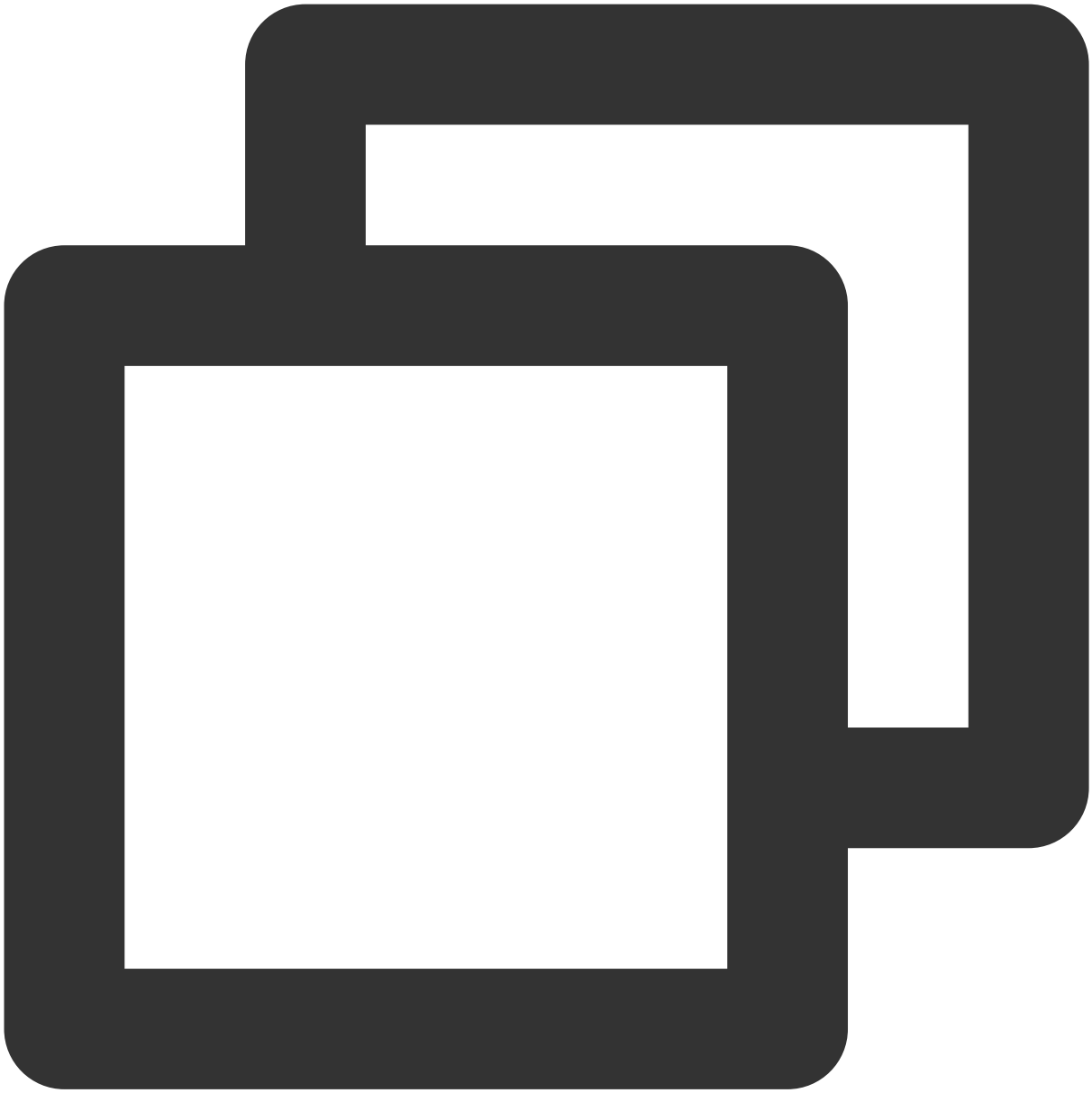
このインターフェースを呼び出すと、すぐにマイクリストが修正されます。管理者がリスナーの同意がなければマイク・オンできないケースの場合は、まず `sendInvitation` を呼び出してからリスナーに申請し、`onInvitationAccept` を受信すると、この関数を呼び出せるようになります。

kickSeat

キックアウトしてマイク・オフ（管理者が呼び出し）。

説明：

管理者がキックアウトしてマイク・オフにすると、ルーム内の全メンバーは、`onSeatListChange` および `onAnchorLeaveSeat` というイベント通知を受信します。



```
- (void)kickSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback N
```

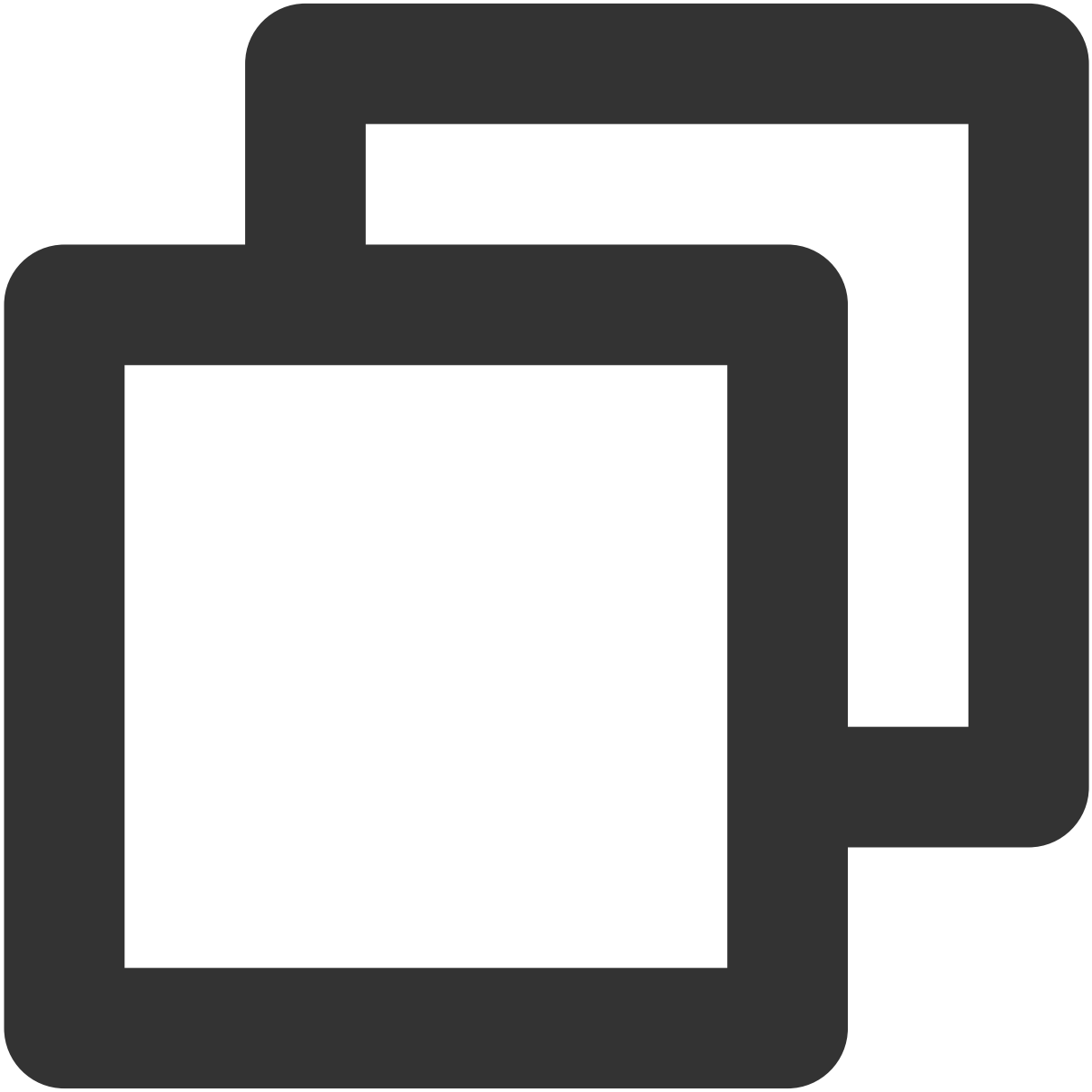
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
seatIndex	NSInteger	キックアウトしてマイク・オフの必要があるマイク番号。
callback	ActionCallback	操作コールバック。

このインターフェースを呼び出すと、すぐにマイクリストが修正されます。

muteSeat

任意のマイクのミュート/ミュート解除（管理者が呼び出し）。
説明：
任意のマイクをミュート/ミュート解除します。ルーム内の全メンバーが `onSeatListChange` および `onSeatMute` というイベント通知を受信します。



```
- (void)muteSeat:(NSInteger)seatIndex isMute:(BOOL)isMute callback:(ActionCallback
```

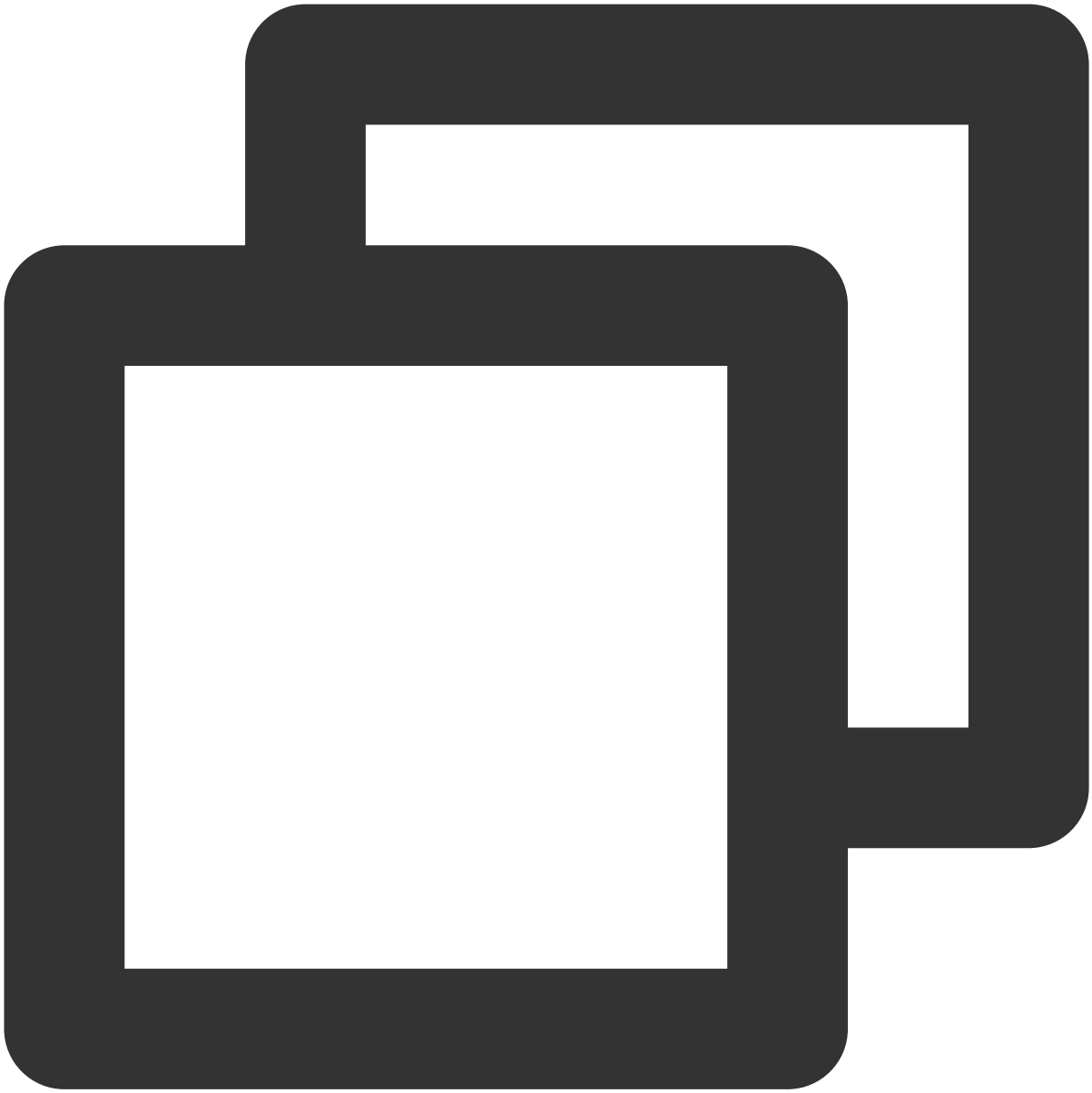
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
seatIndex	NSInteger	操作が必要なマイク番号。
isMute	BOOL	YES：該当するマイクのミュート；NO：該当するマイクのミュート解除。
callback	ActionCallback	操作コールバック。

このインターフェースを呼び出すと、直ちにマイクリストが変更されます。seatIndexの座席に該当するキャスターは、muteAudioを自動的に呼び出してミュート/ミュート解除にします。

closeSeat

任意のマイクのクローズ/解除（管理者が呼び出し）。
説明：
管理者は、該当するマイクをクローズ/解除し、ルーム内の全メンバーは `onSeatListChange` および `onSeatClose` というイベント通知を受信します。



```
- (void)closeSeat:(NSInteger)seatIndex isClose:(BOOL)isClose callback:(ActionCallba
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
seatIndex	NSInteger	操作が必要なマイク番号。
isClose	BOOL	YES：該当するマイクのクローズ； NO：該当するマイクのクローズ解除。

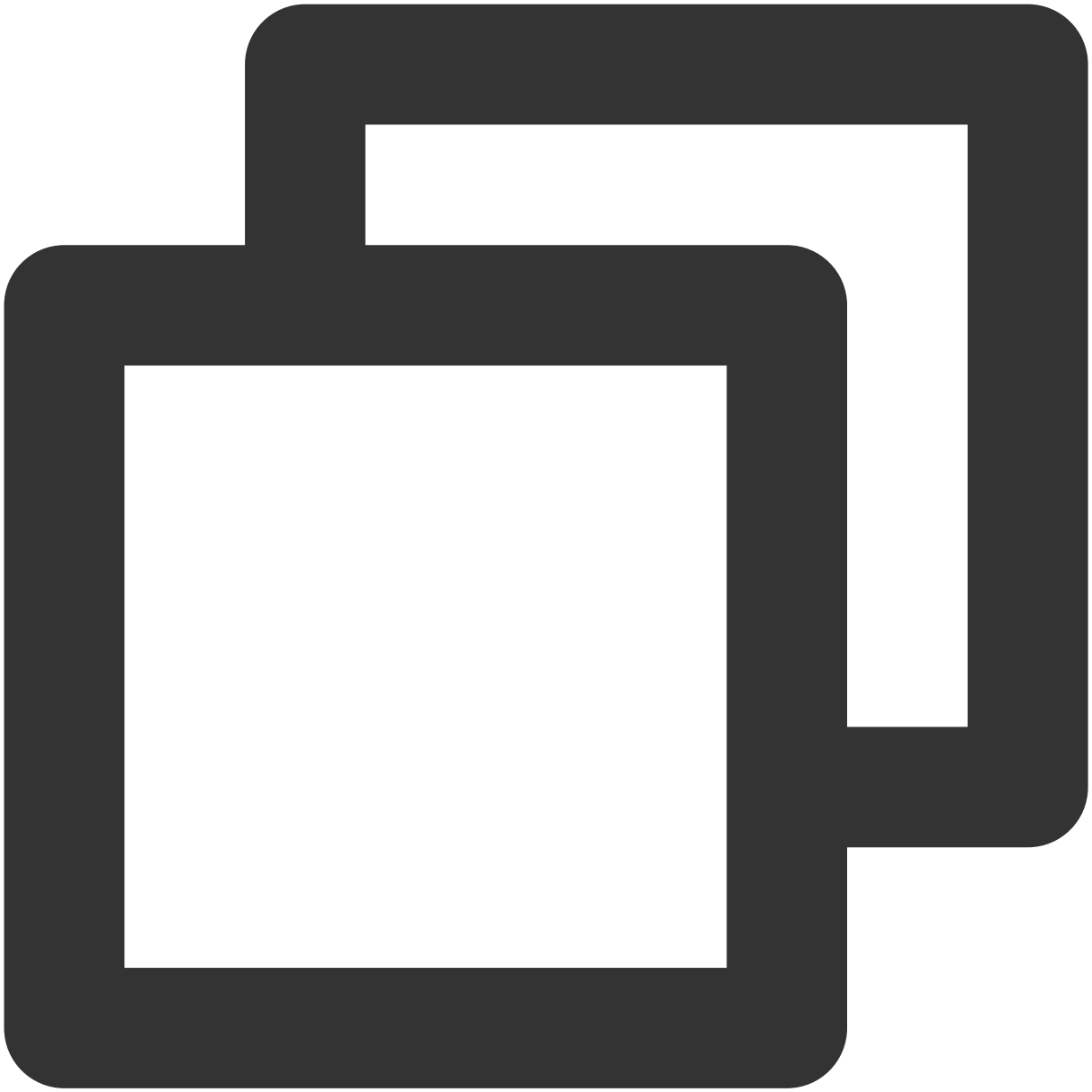
callback	ActionCallback	操作コールバック。

このインターフェースを呼び出すと、すぐにマイクリストが修正されます。該当するseatIndexの座席上のキャスターはクローズされ、自動的にマイク・オフになります。

ローカル音声操作のインターフェース

startMicrophone

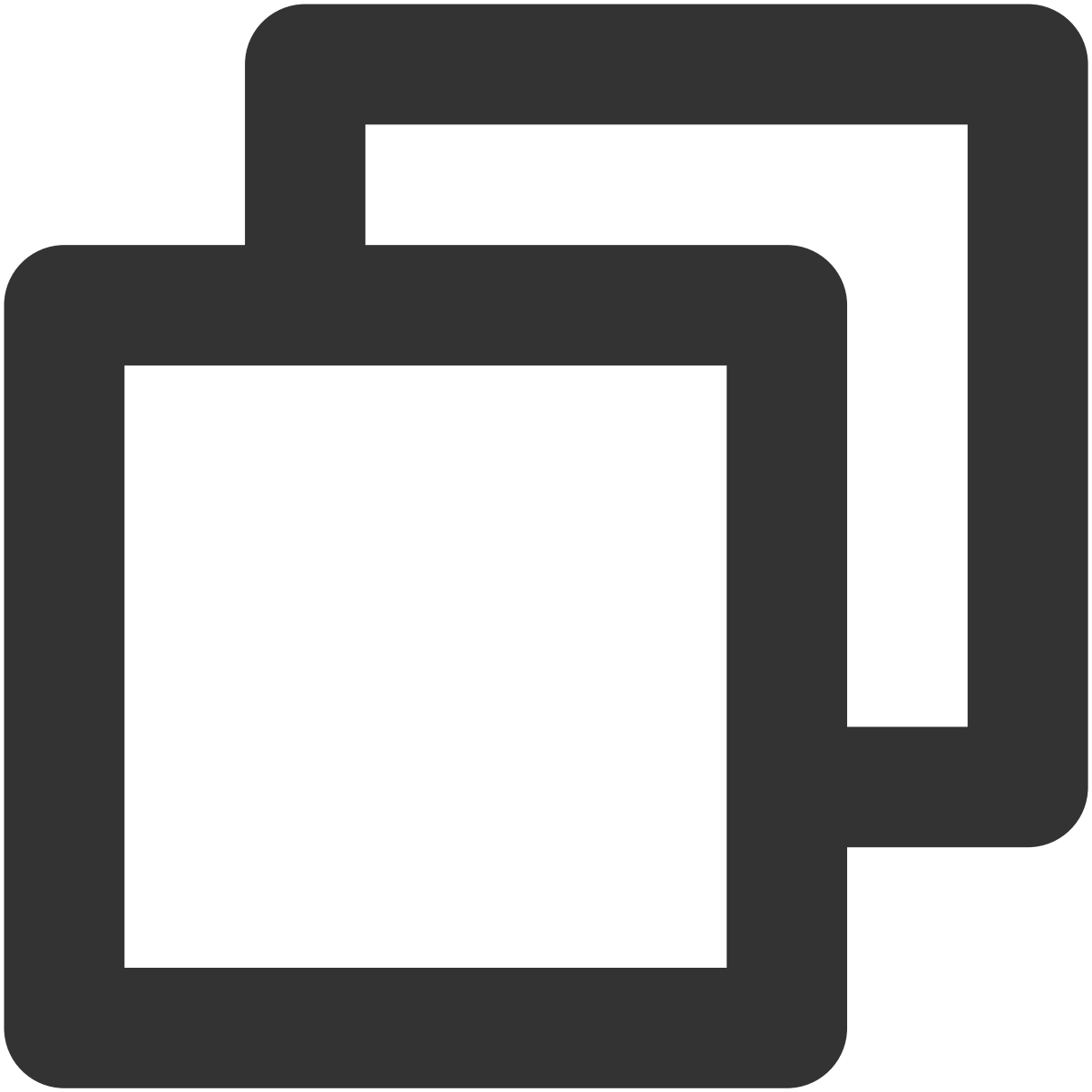
マイクの集音開始。



```
- (void)startMicrophone;
```

stopMicrophone

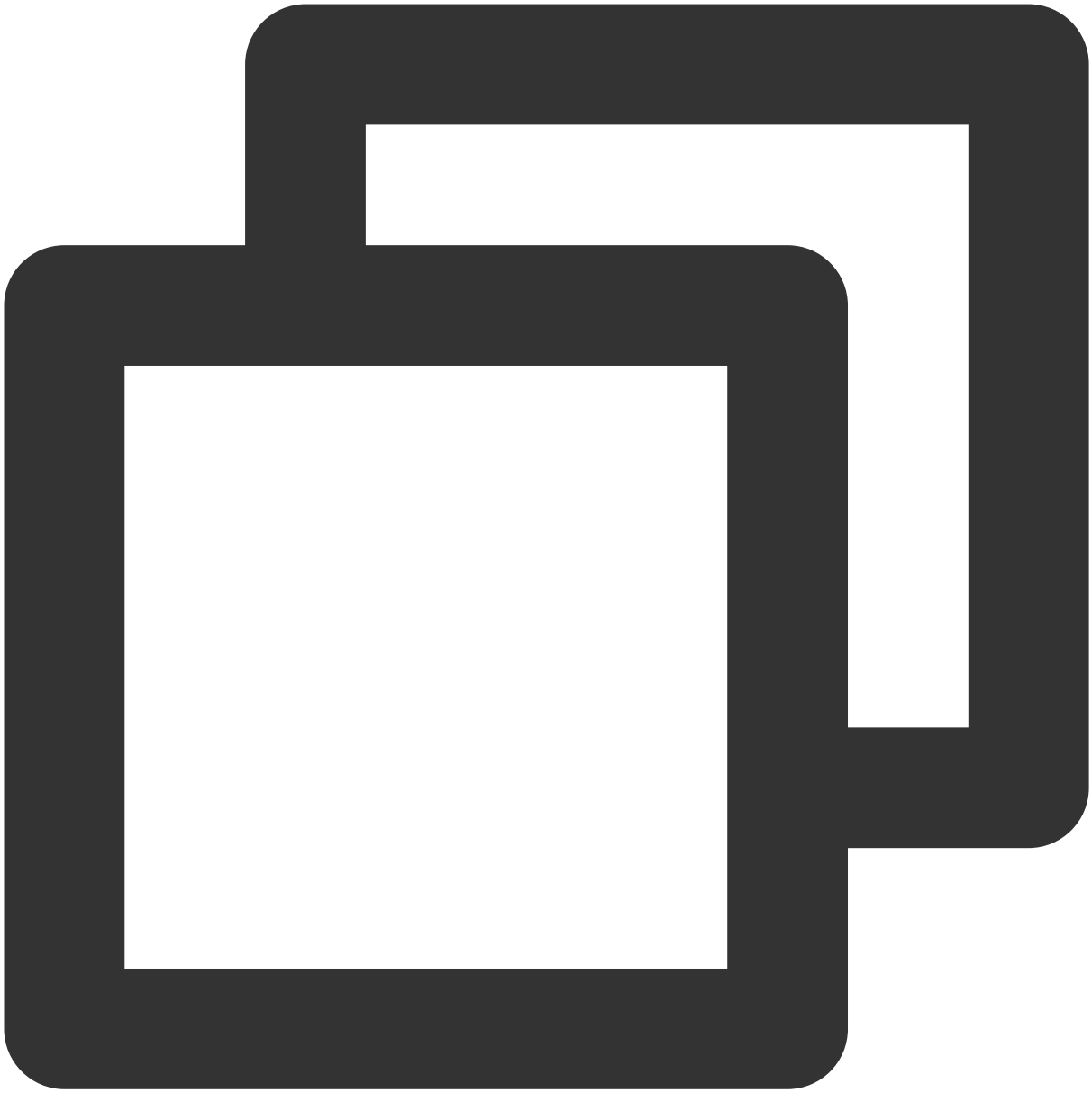
マイクの集音停止。



```
- (void)stopMicrophone;
```

setAudioQuality

音質の設定。



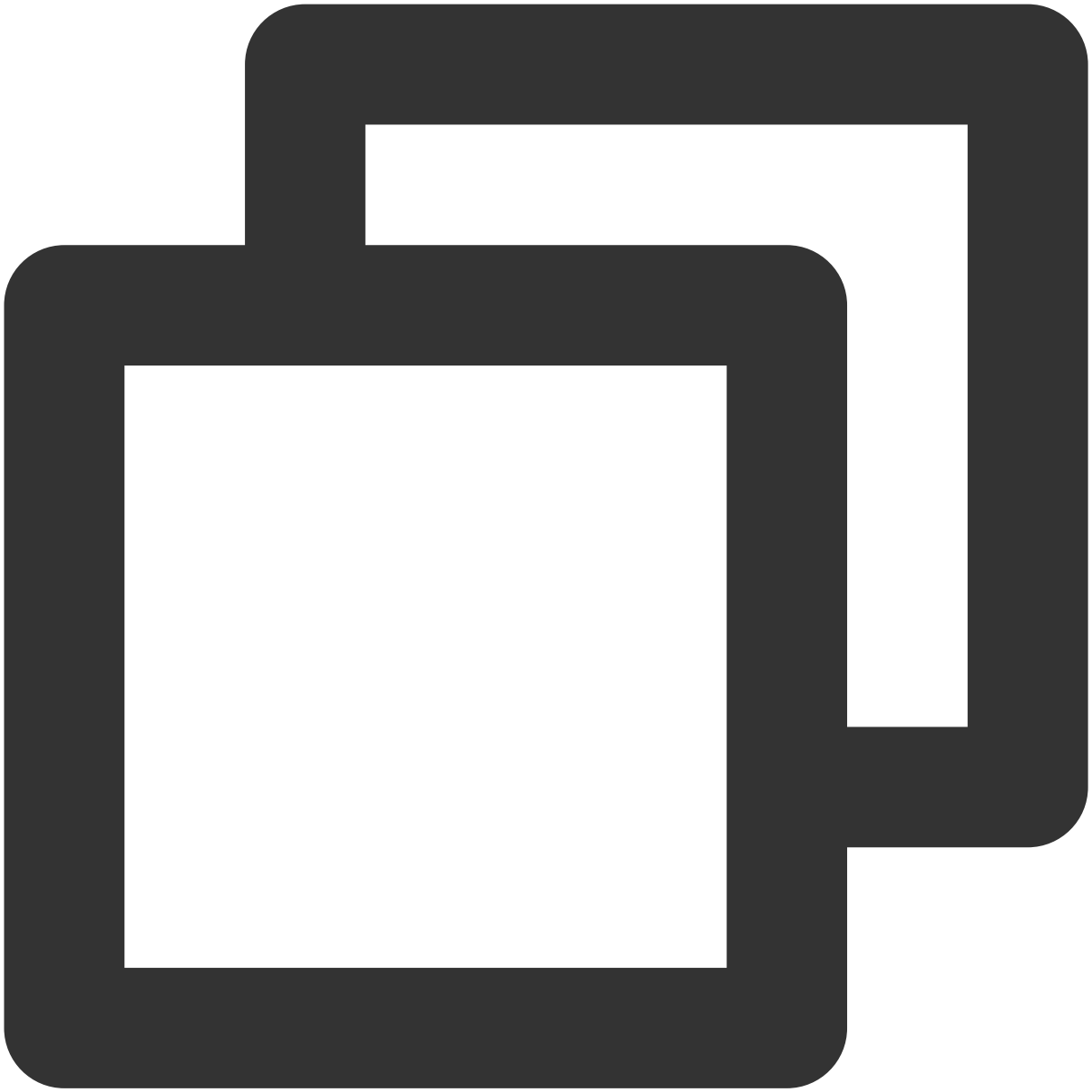
```
- (void)setAuidoQuality:(NSInteger)quality NS_SWIFT_NAME(setAuidoQuality(quality:))
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
quality	NSInteger	オーディオ品質。詳細については、TRTC SDKをご参照ください。

muteLocalAudio

ローカルの音声のミュート/ミュート取り消し。



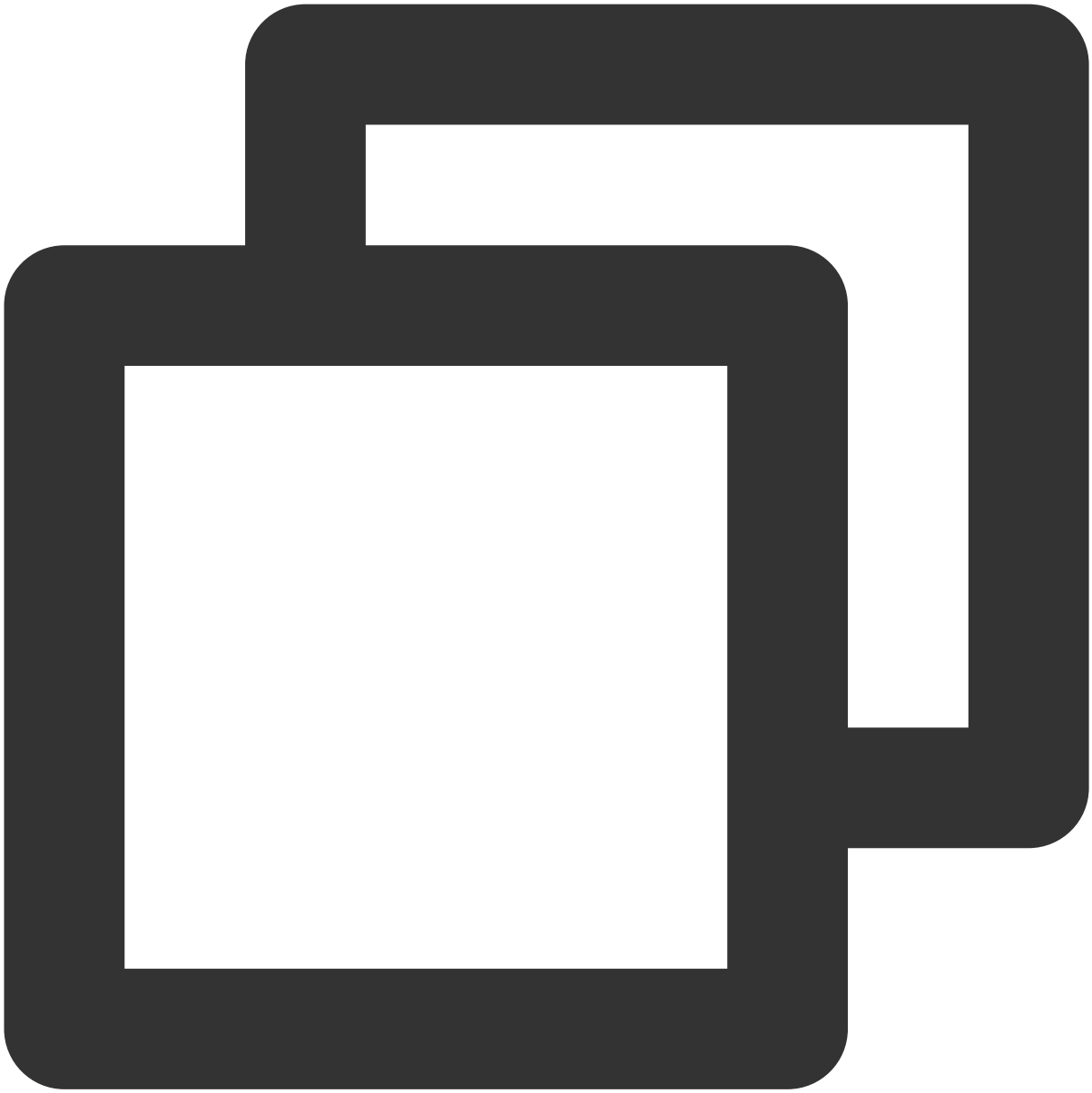
```
- (void)muteLocalAudio:(BOOL)mute NS_SWIFT_NAME(muteLocalAudio(mute:));
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
mute	BOOL	ミュート/ミュートの取り消し。詳細については、TRTC SDKをご参照ください。

setSpeaker

スピーカーの起動設定。



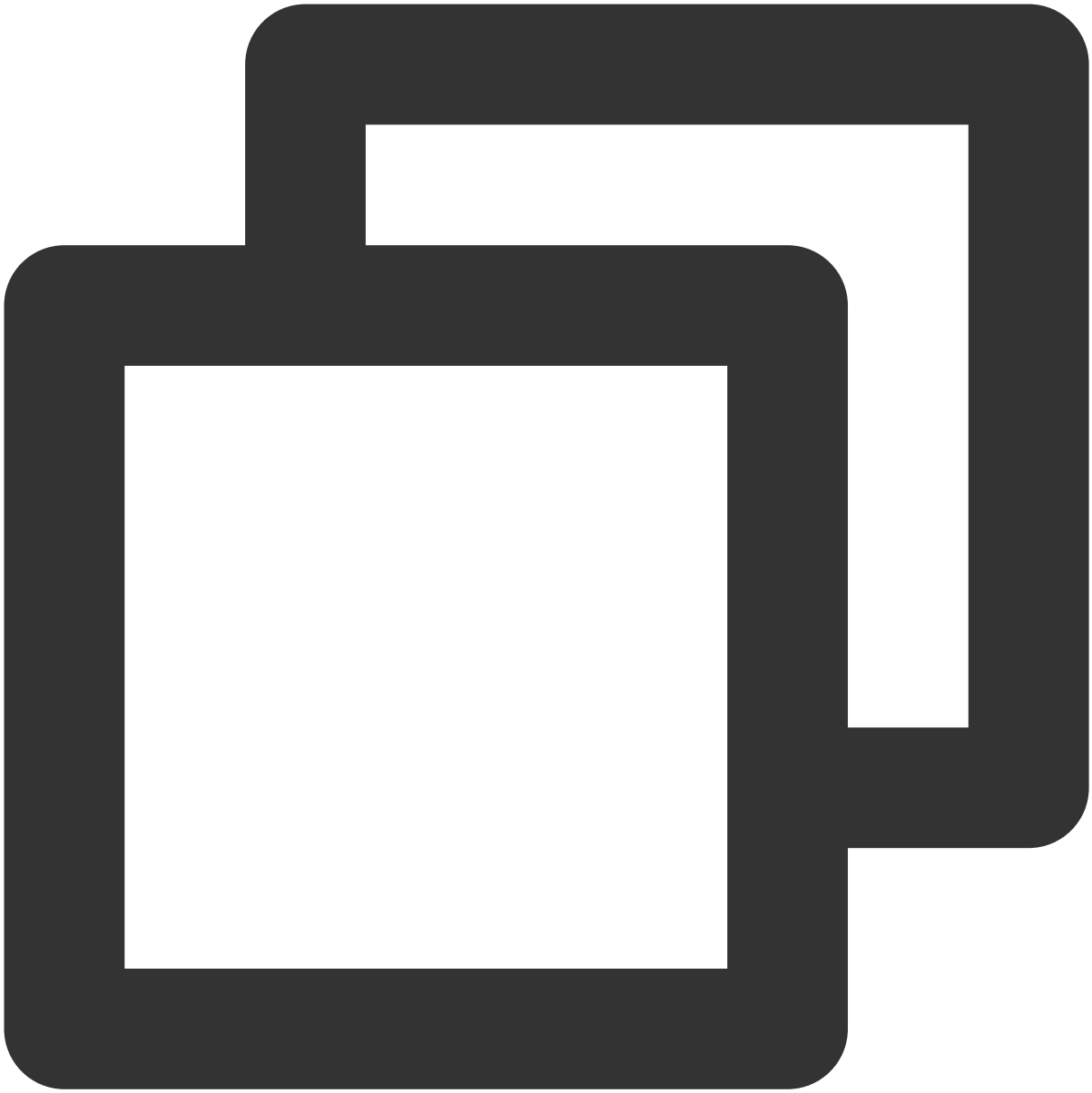
```
- (void)setSpeaker:(BOOL)userSpeaker NS_SWIFT_NAME(setSpeaker(userSpeaker:));
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
useSpeaker	BOOL	YES：スピーカー；NO：ヘッドホン。

setAudioCaptureVolume

マイクの集音音量設定。



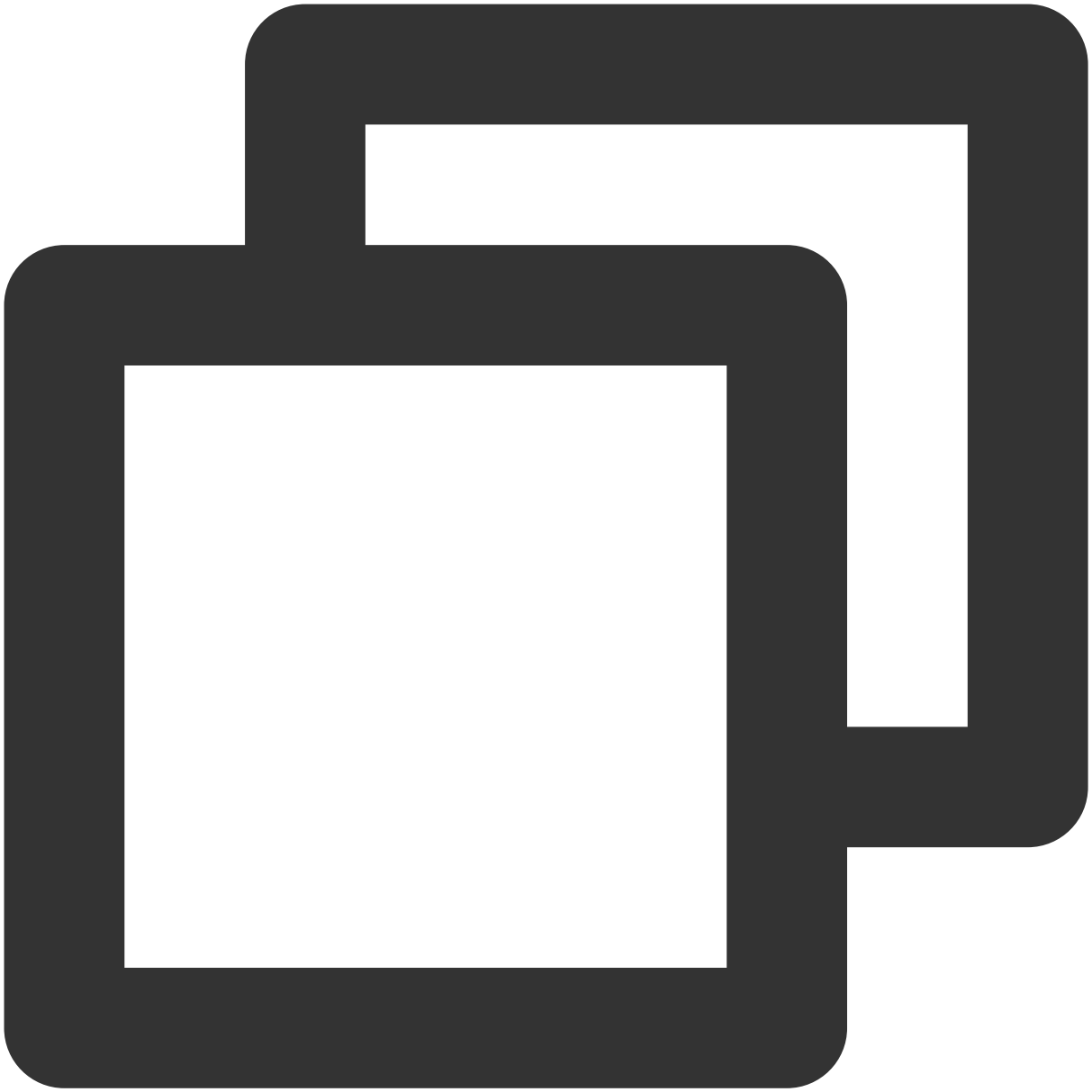
```
- (void)setAudioCaptureVolume:(NSInteger)voluem NS_SWIFT_NAME(setAudioCaptureVolume
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
volume	NSInteger	集音音量、0 - 100、デフォルト100。

setAudioPlayoutVolume

再生音量の設定。



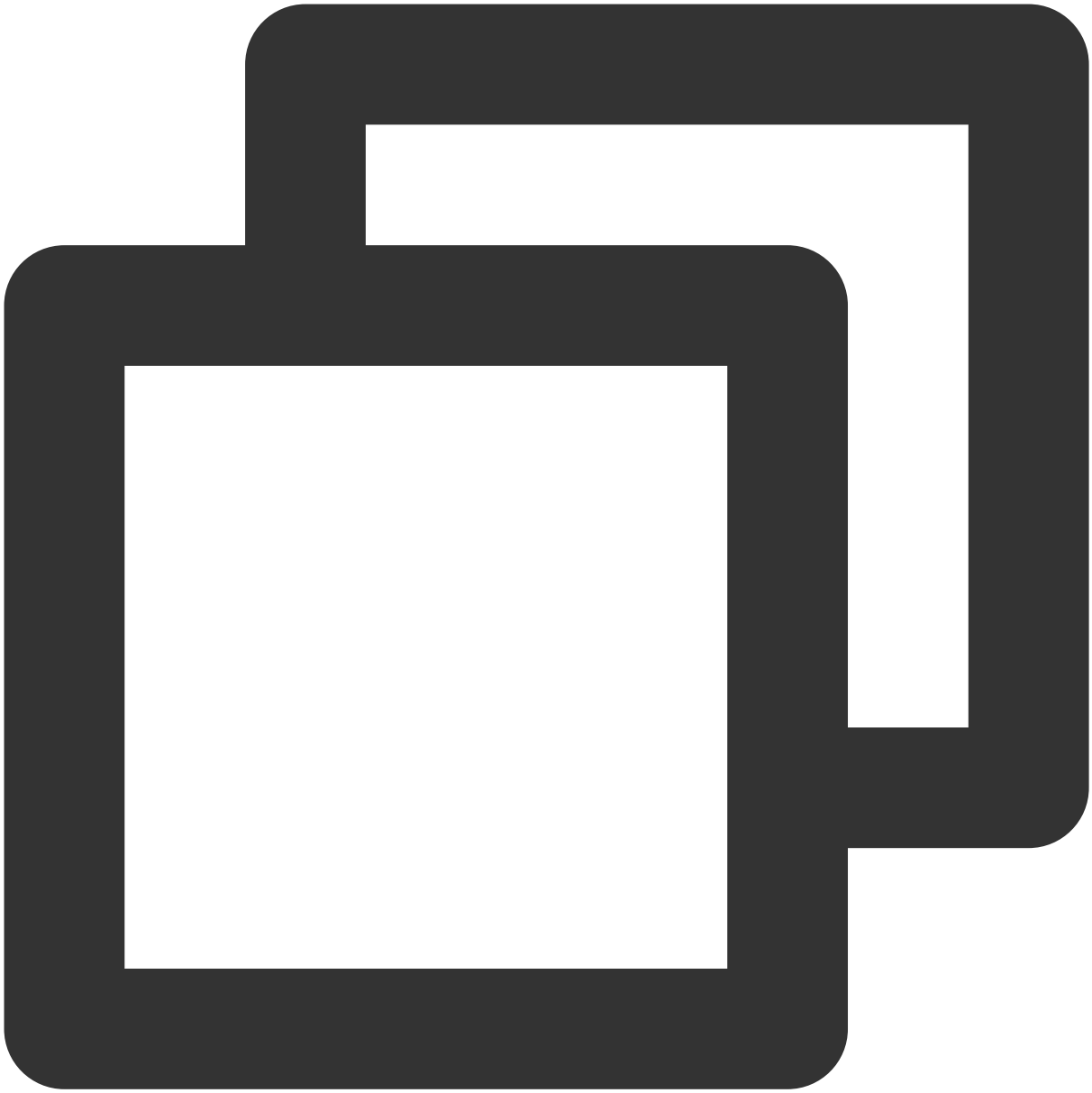
```
- (void)setAudioPlayoutVolume:(NSInteger)volume NS_SWIFT_NAME(setAudioPlayoutVolume
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
volume	NSInteger	再生音量、0 - 100、デフォルト100。

muteRemoteAudio

指定メンバーのミュート/ミュート解除。



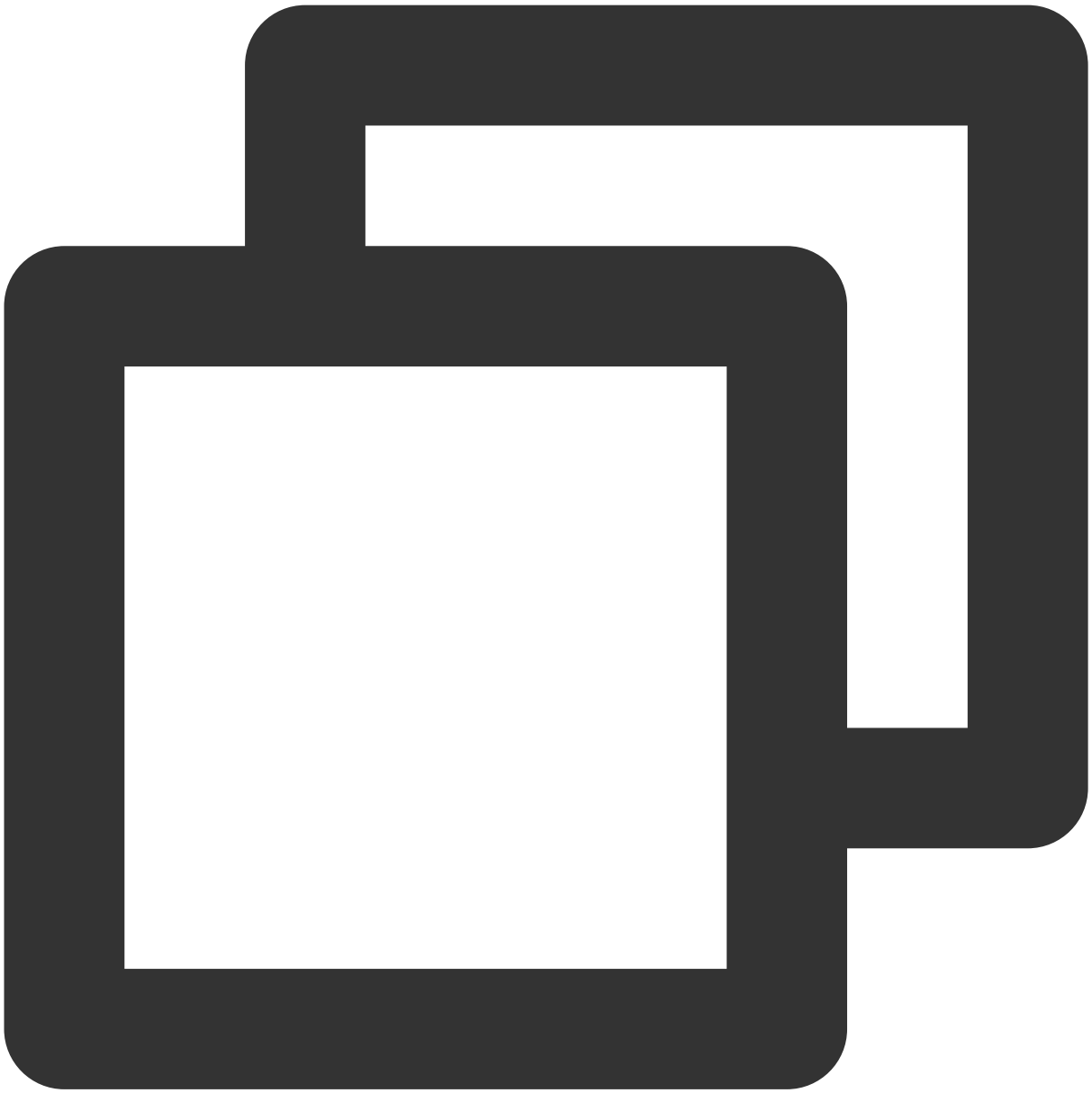
```
- (void)muteRemoteAudio:(NSString *)userId mute:(BOOL)mute NS_SWIFT_NAME(muteRemote
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userId	NSString	指定のユーザーID。
mute	BOOL	YES：ミュート起動；NO：ミュート停止。

muteAllRemoteAudio

全メンバーのミュート/ミュート解除。



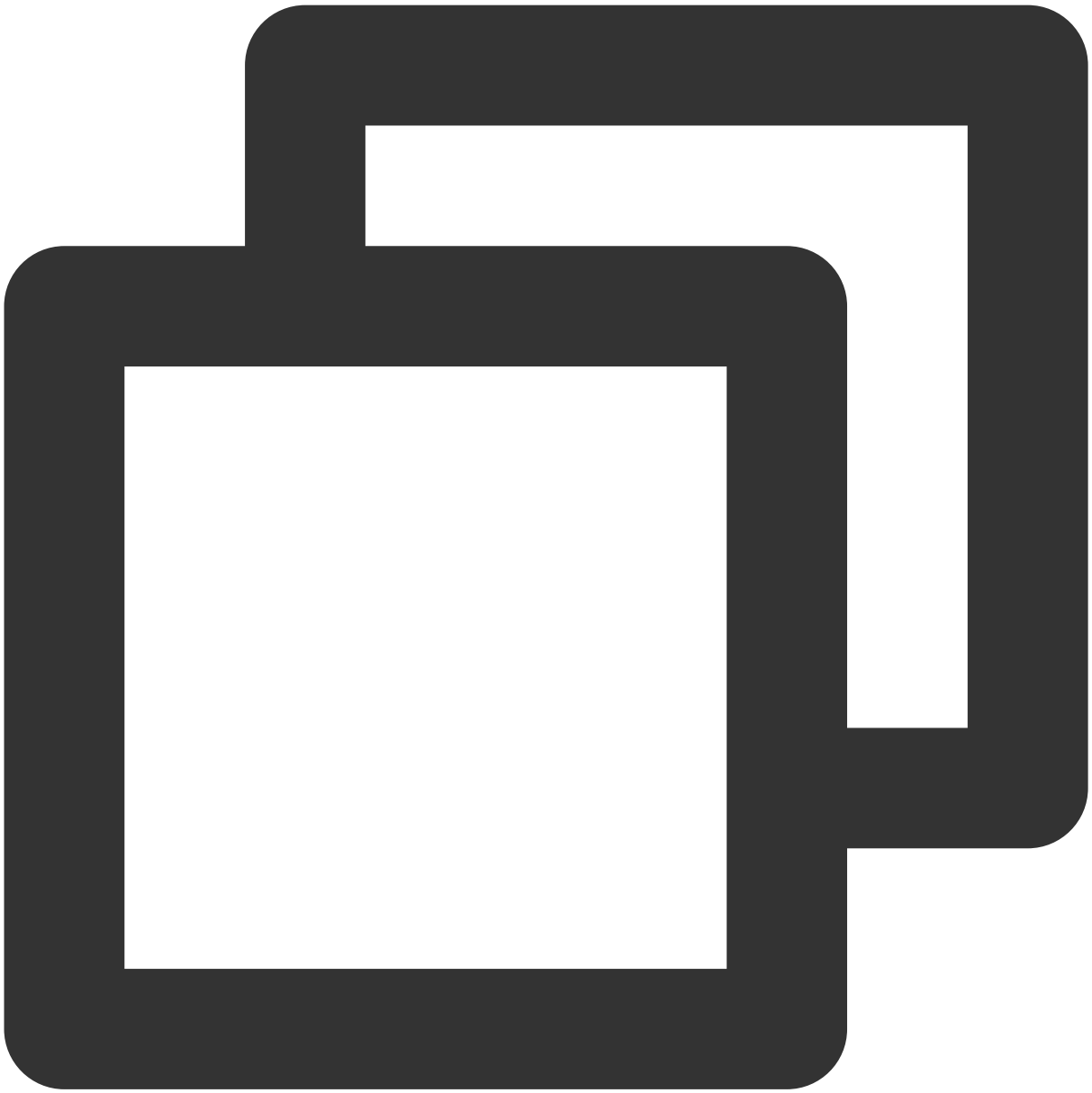
```
- (void)muteAllRemoteAudio:(BOOL)isMute NS_SWIFT_NAME(muteAllRemoteAudio(isMute:));
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
mute	BOOL	YES：ミュート起動；NO：ミュート停止。

setVoiceEarMonitorEnable

インイヤーモニタリングのオン/オフ。



```
- (void)setVoiceEarMonitorEnable:(BOOL)enable NS_SWIFT_NAME(setVoiceEarMonitor(enab
```

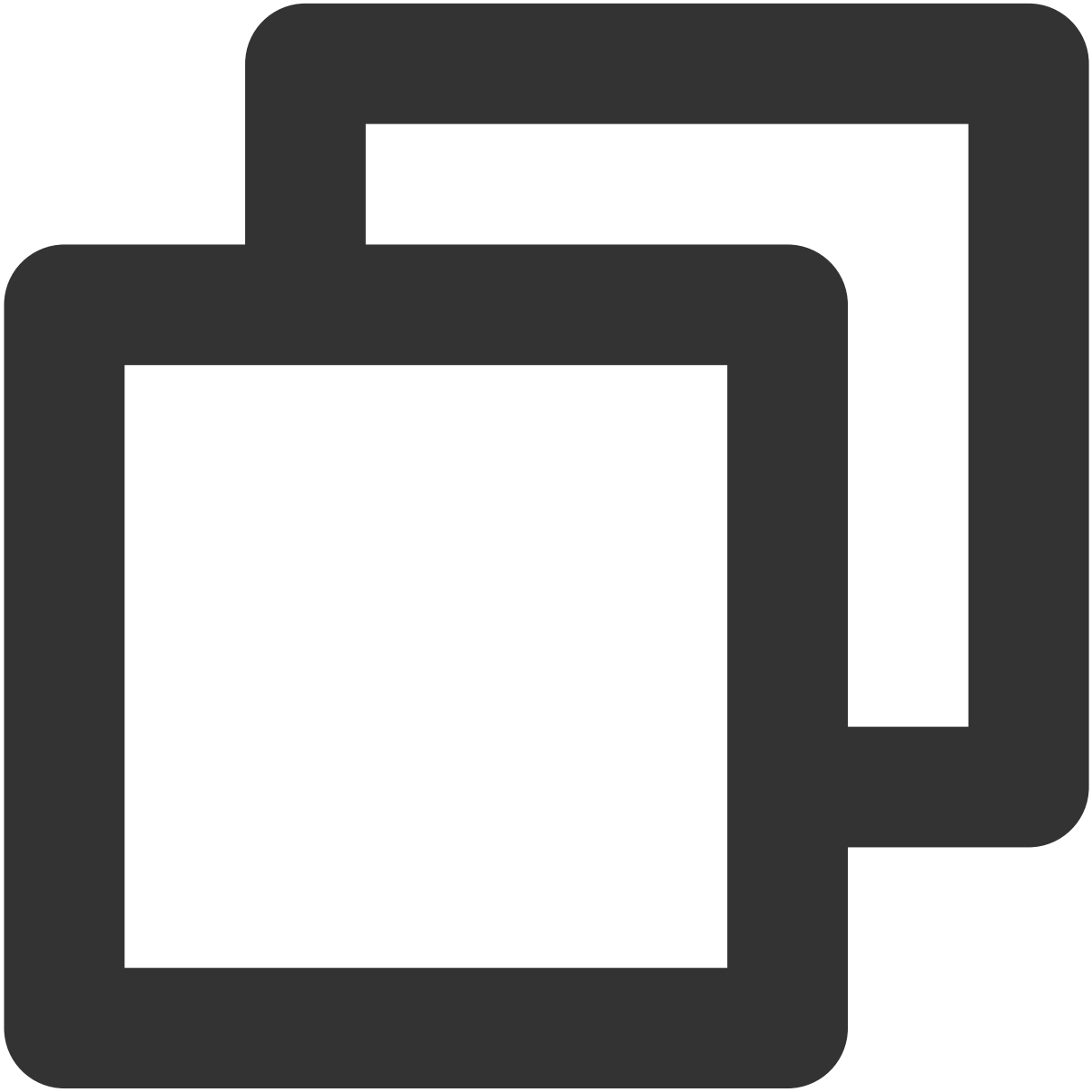
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
enable	BOOL	YES：インイヤーモニタリング起動；NO：インイヤーモニタリング停止。

BGMサウンドエフェクト関連インターフェース関数

getAudioEffectManager

BGMサウンドエフェクト管理オブジェクト TXAudioEffectManagerを取得します。

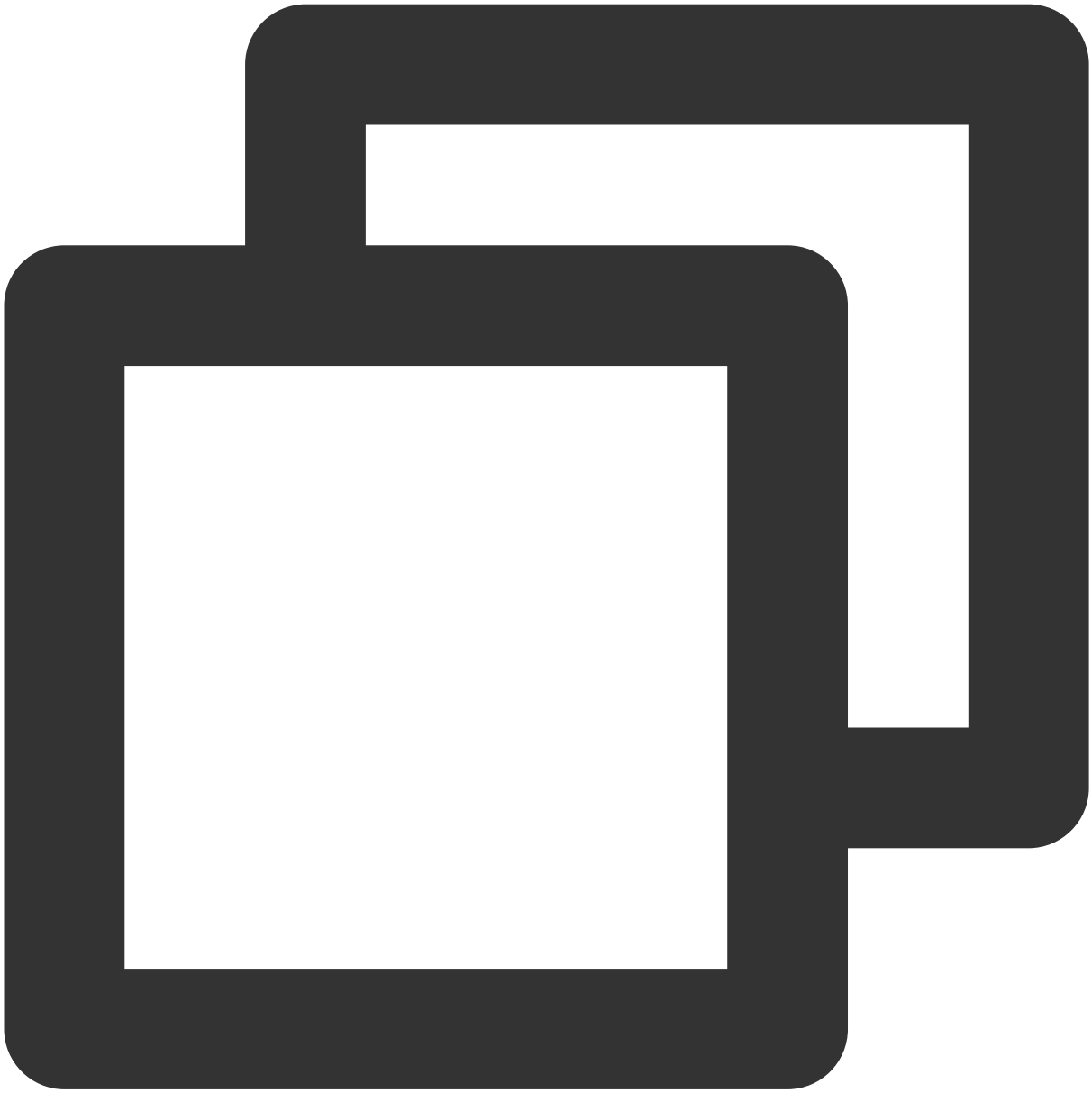


```
- (TXAudioEffectManager * _Nullable)getAudioEffectManager;
```

メッセージ送信関連インターフェース関数

sendRoomTextMsg

ルーム内でテキストメッセージをブロードキャストします。通常、弾幕によるチャットに使用します。



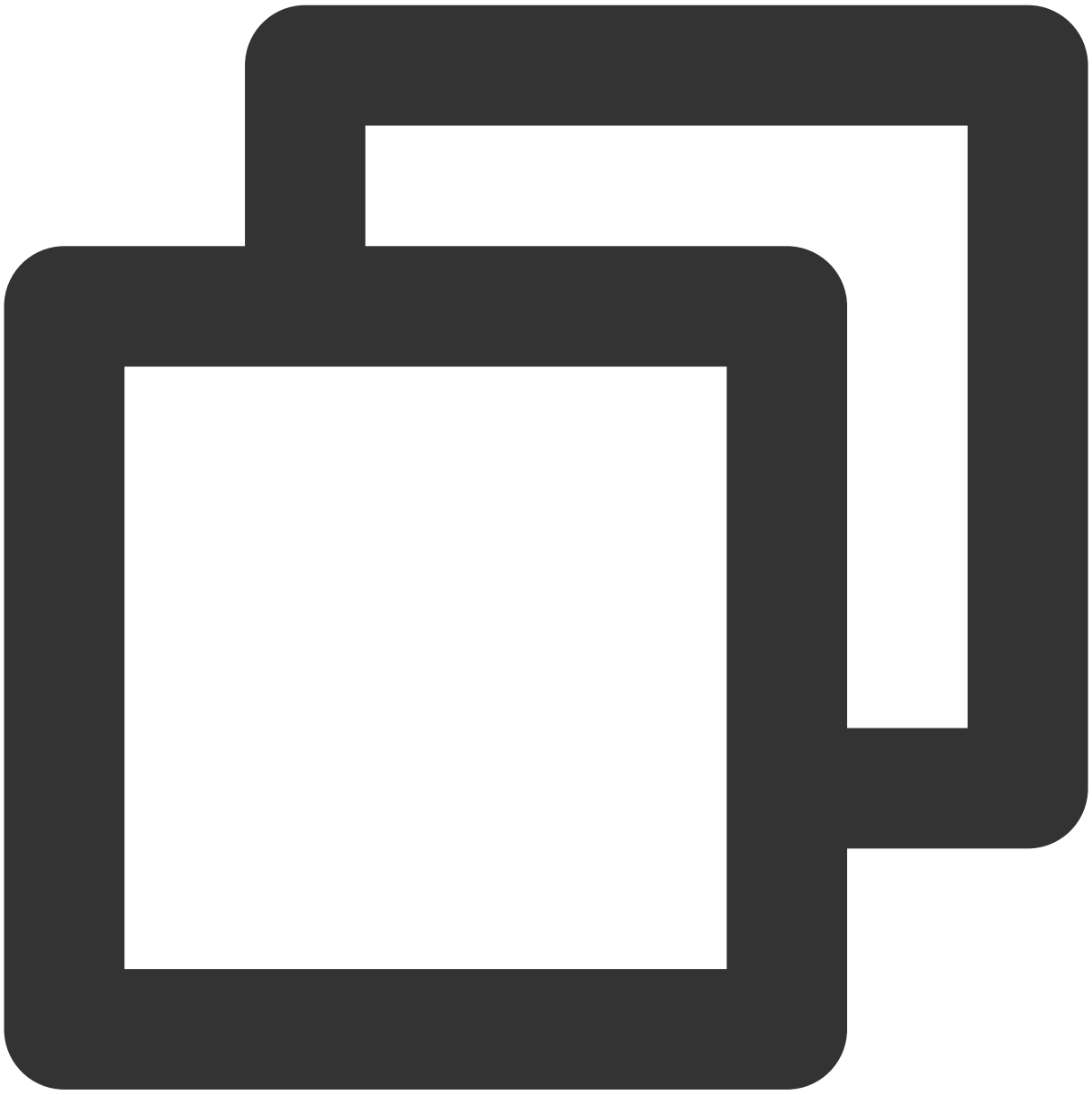
```
- (void)sendRoomTextMsg:(NSString *)message callback:(ActionCallback _Nullable)call
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
message	NSString	テキストメッセージ。
callback	ActionCallback	送信結果のコールバック。

sendRoomCustomMsg

カスタマイズしたテキストメッセージを送信します。



```
- (void)sendRoomCustomMsg:(NSString *)cmd message:(NSString *)message callback:(Act
```

パラメータは下表に示すとおりです：

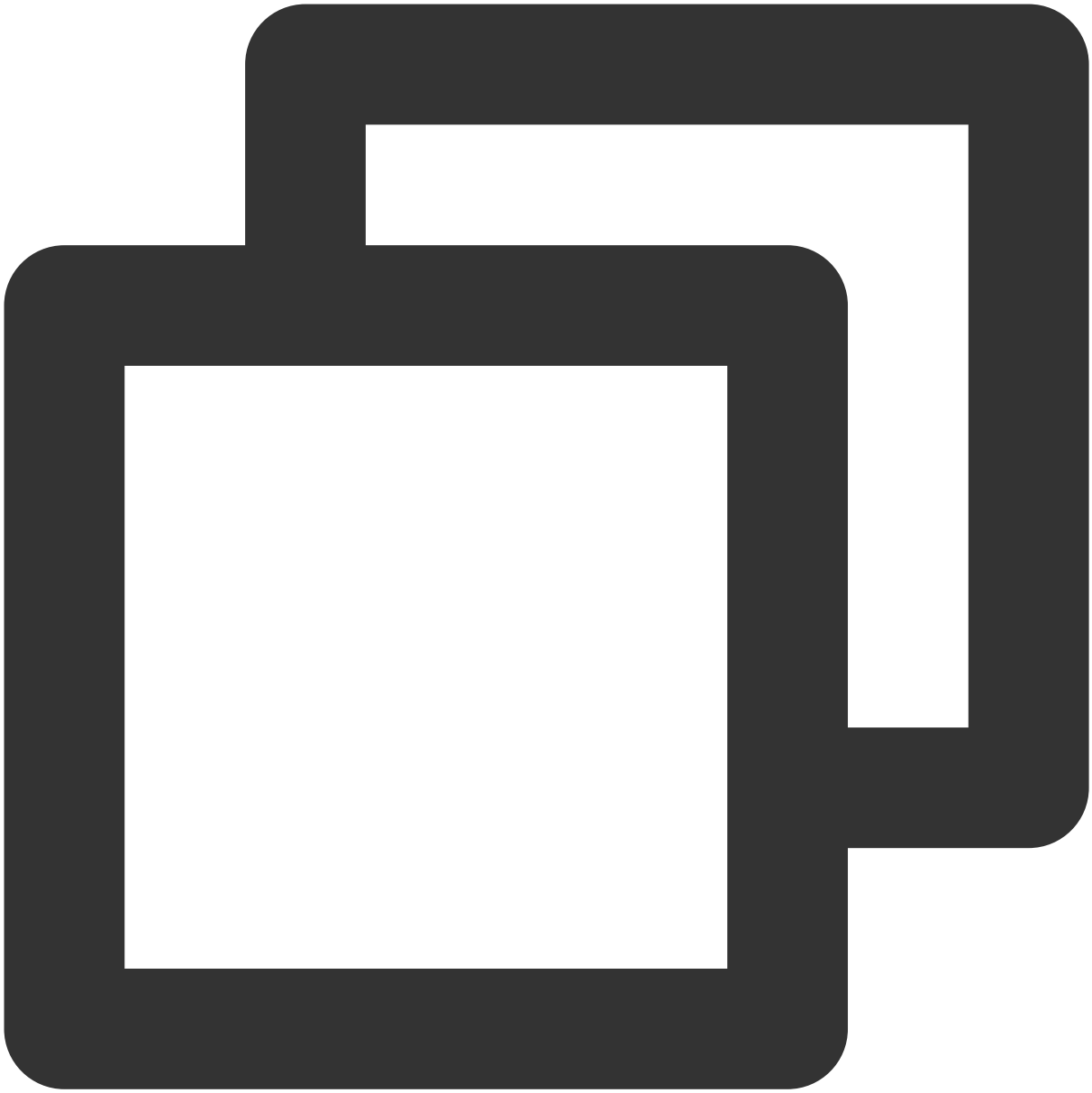
パラメータ	タイプ	意味

cmd	NSString	コマンドワードです。開発者がカスタマイズするものであり、主にさまざまなメッセージタイプを区別するために使用されます。
message	NSString	テキストメッセージ。
callback	ActionCallback	送信結果のコールバック。

招待シグナリング関連インターフェース

sendInvitation

ユーザーに招待を送信。



```
- (NSString *)sendInvitation:(NSString *)cmd
                      userId:(NSString *)userId
                     content:(NSString *)content
                    callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(sendI
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
cmd	NSString	業務カスタマイズコマンド。

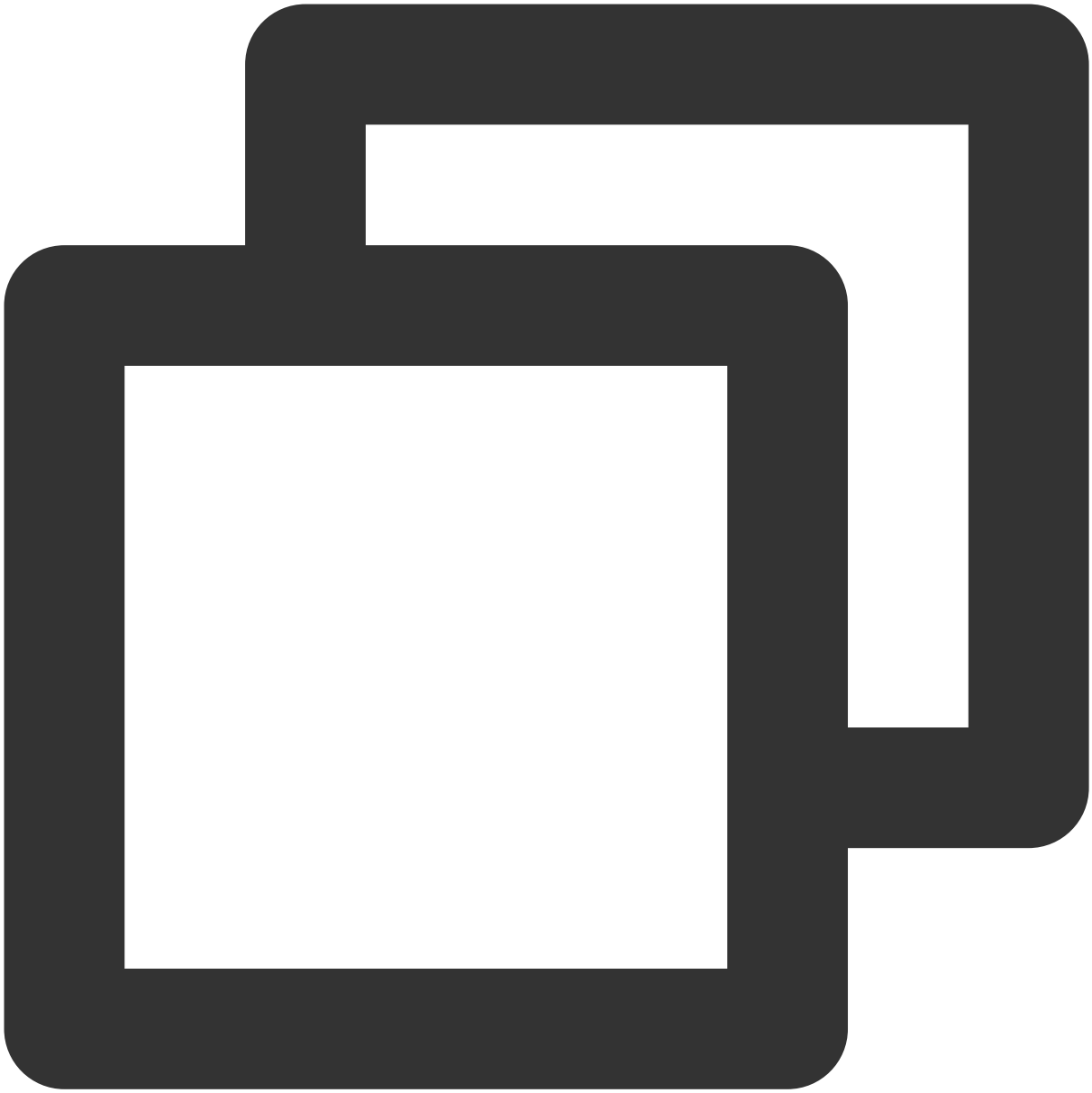
userId	NSString	招待するユーザーID。
content	NSString	招待するコンテンツ。
callback	ActionCallback	送信結果のコールバック。

戻り値：

戻り値	タイプ	意味
inviteId	NSString	今回の招待IDの識別に使用。

acceptInvitation

招待の同意。



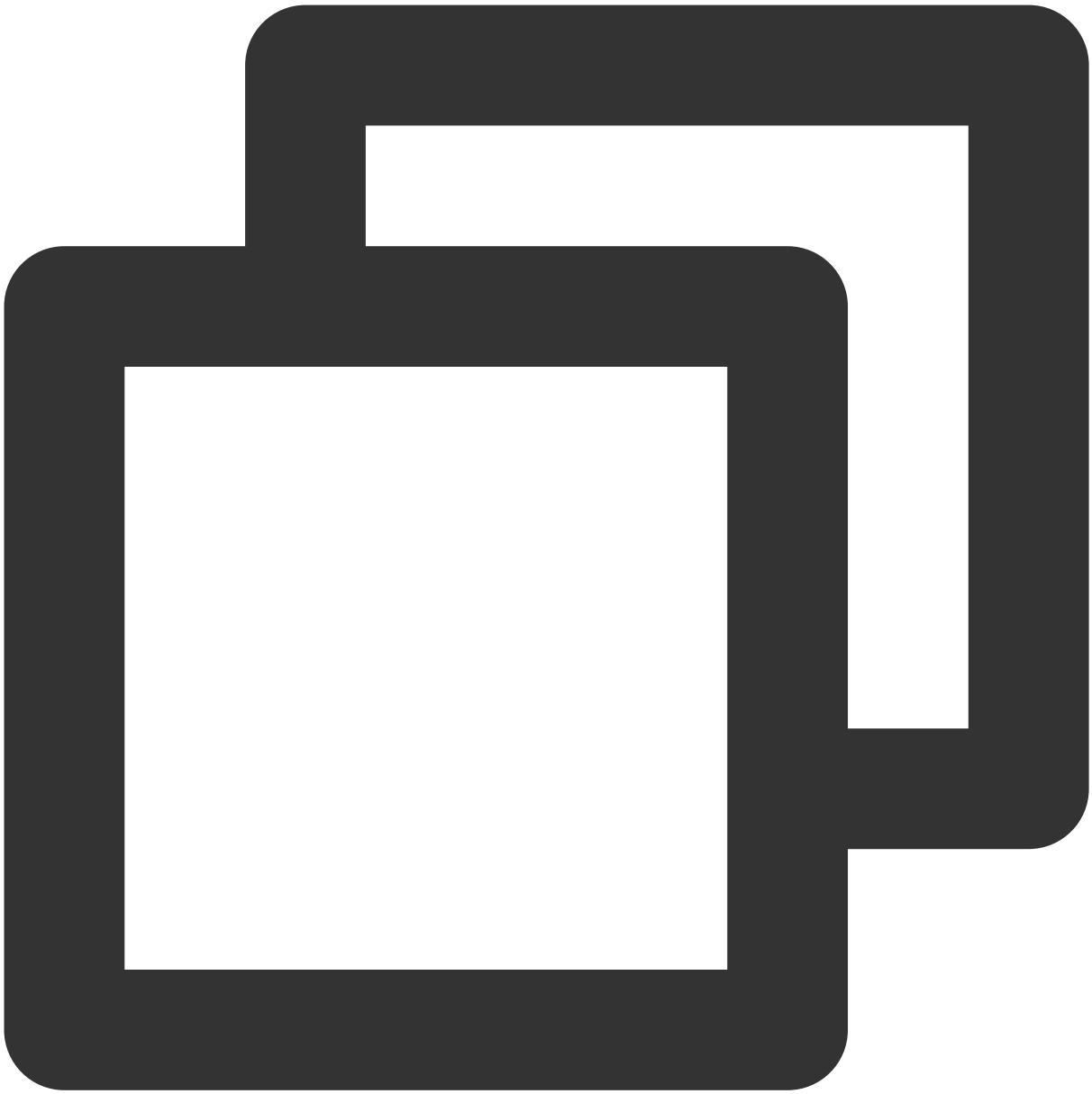
```
- (void)acceptInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
id	NSString	招待ID。
callback	ActionCallback	送信結果のコールバック。

rejectInvitation

招待の拒否。



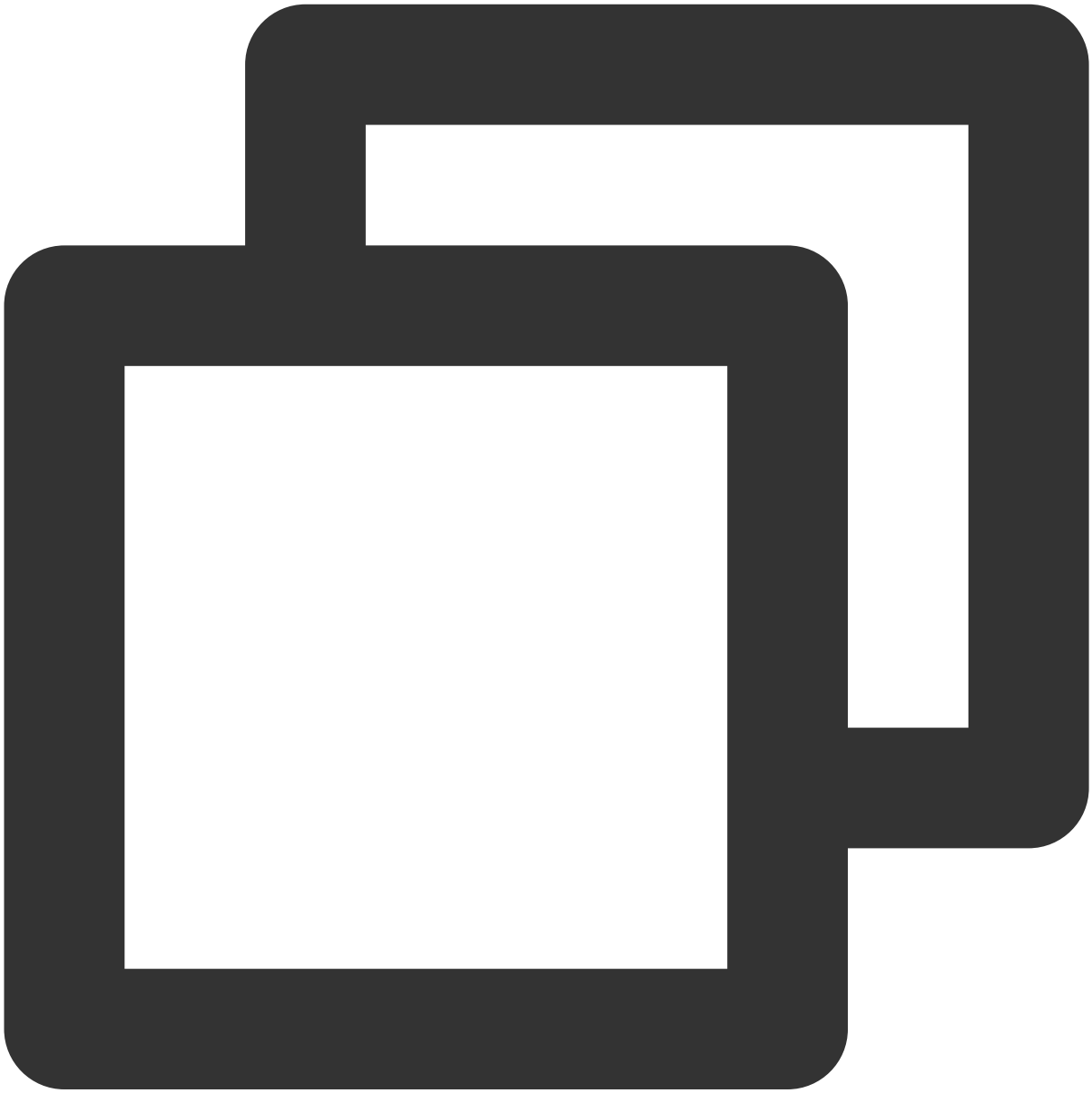
```
- (void)rejectInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
id	NSString	招待ID。
callback	ActionCallback	送信結果のコールバック。

cancelInvitation

招待の取り消し。



```
- (void)cancelInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
id	NSString	招待ID。
callback	ActionCallback	送信結果のコールバック。

TRTCVoiceRoomDelegateイベントのコールバック

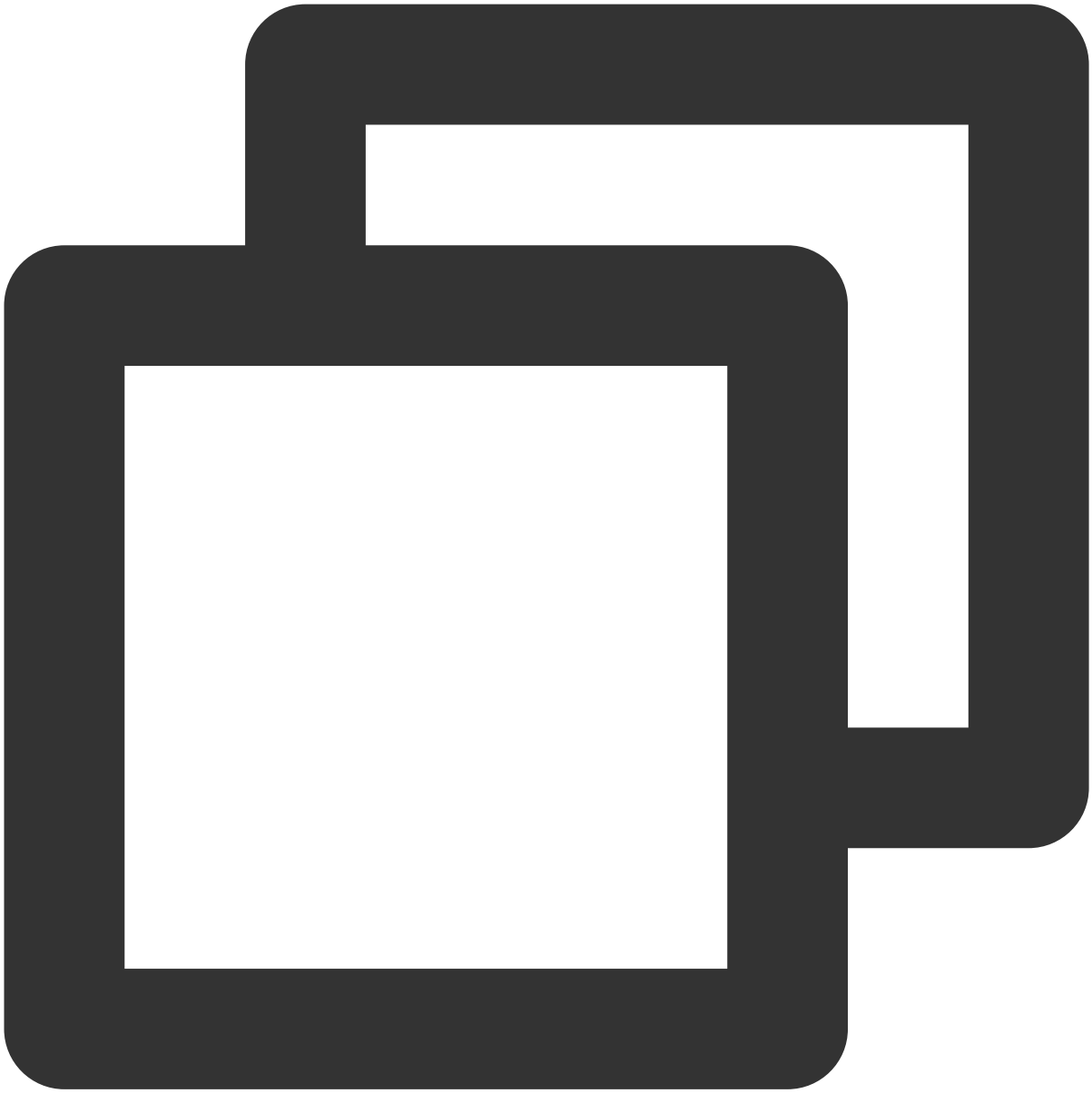
一般的なイベントコールバック

onError

エラーのコールバック。

説明：

SDKリカバリー不能なエラーは必ず監視し、状況に応じてユーザーに適切なインターフェースプロンプトを表示します。



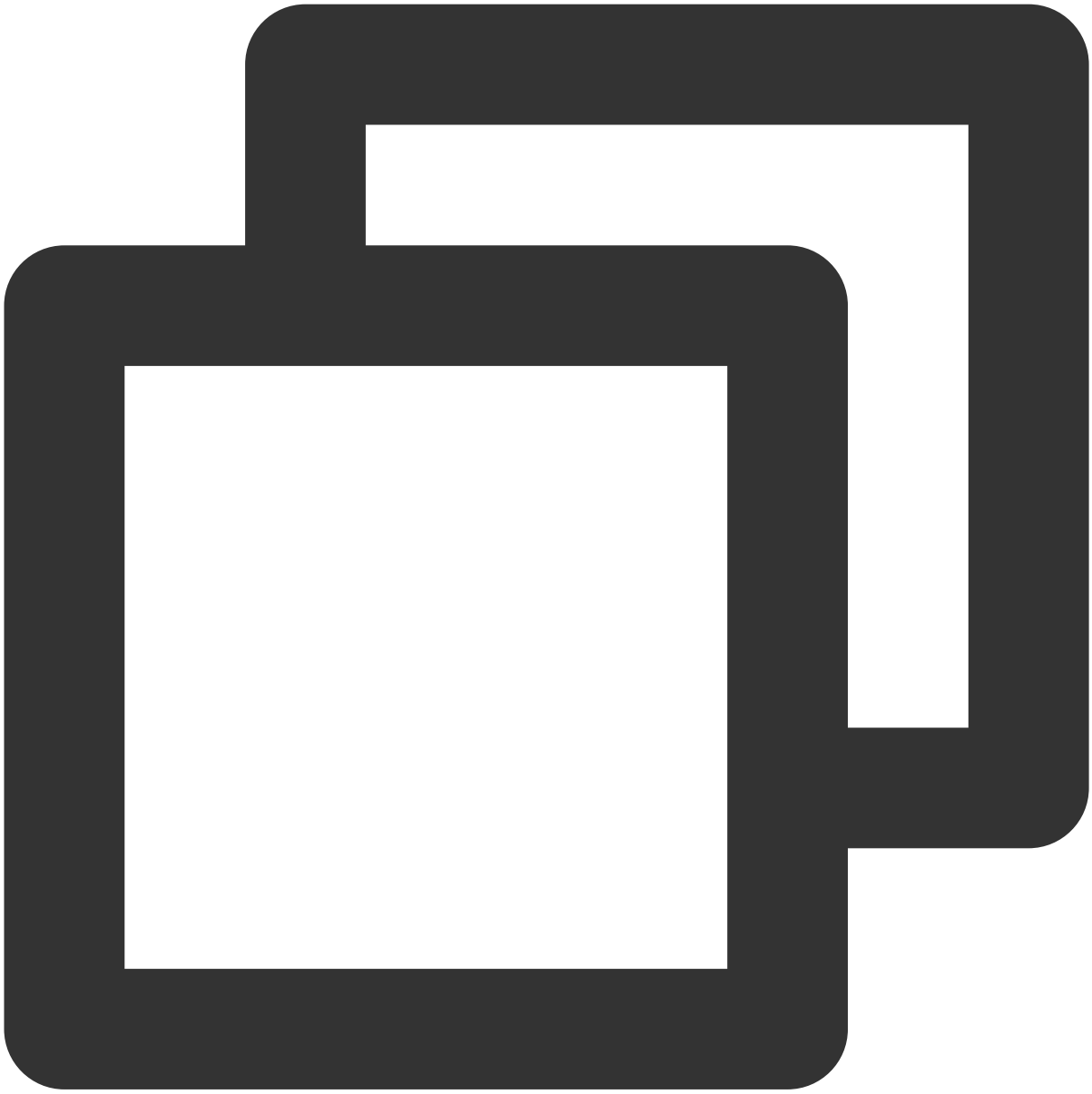
```
- (void)onError:(int)code
                message:(NSString*)message
NS_SWIFT_NAME(onError(code:message:));
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
code	int	エラーコード。
message	NSString	エラー情報。

onWarning

警告のコールバック。



```
- (void)onWarning:(int)code
                  message:(NSString  *)message
NS_SWIFT_NAME(onWarning(code:message:));
```

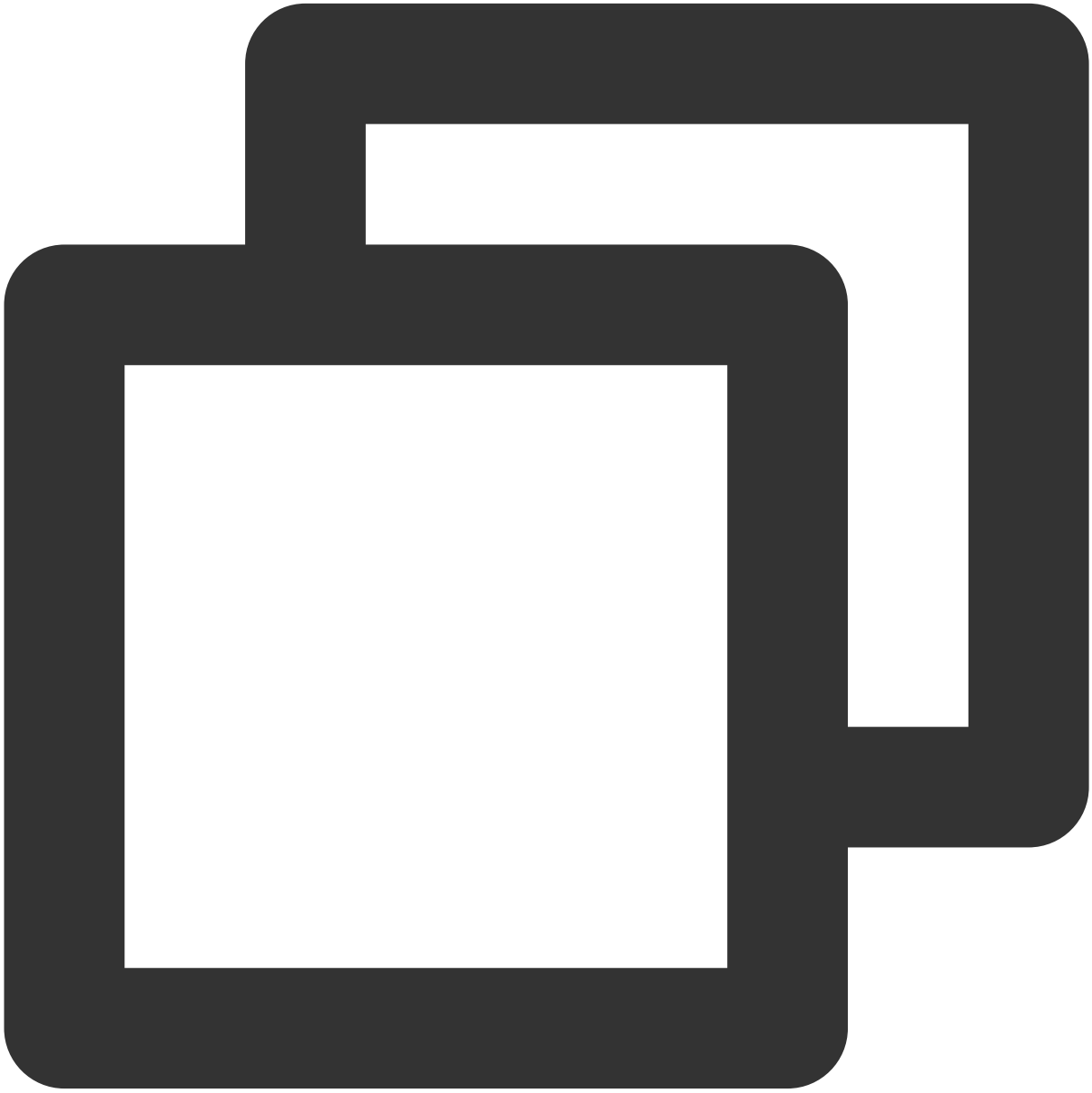
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味

code	int	エラーコード。
message	NSString	警告メッセージ。

onDebugLog

Logコールバック。



```
- (void)onDebugLog:(NSString *)message
NS_SWIFT_NAME(onDebugLog(message:));
```

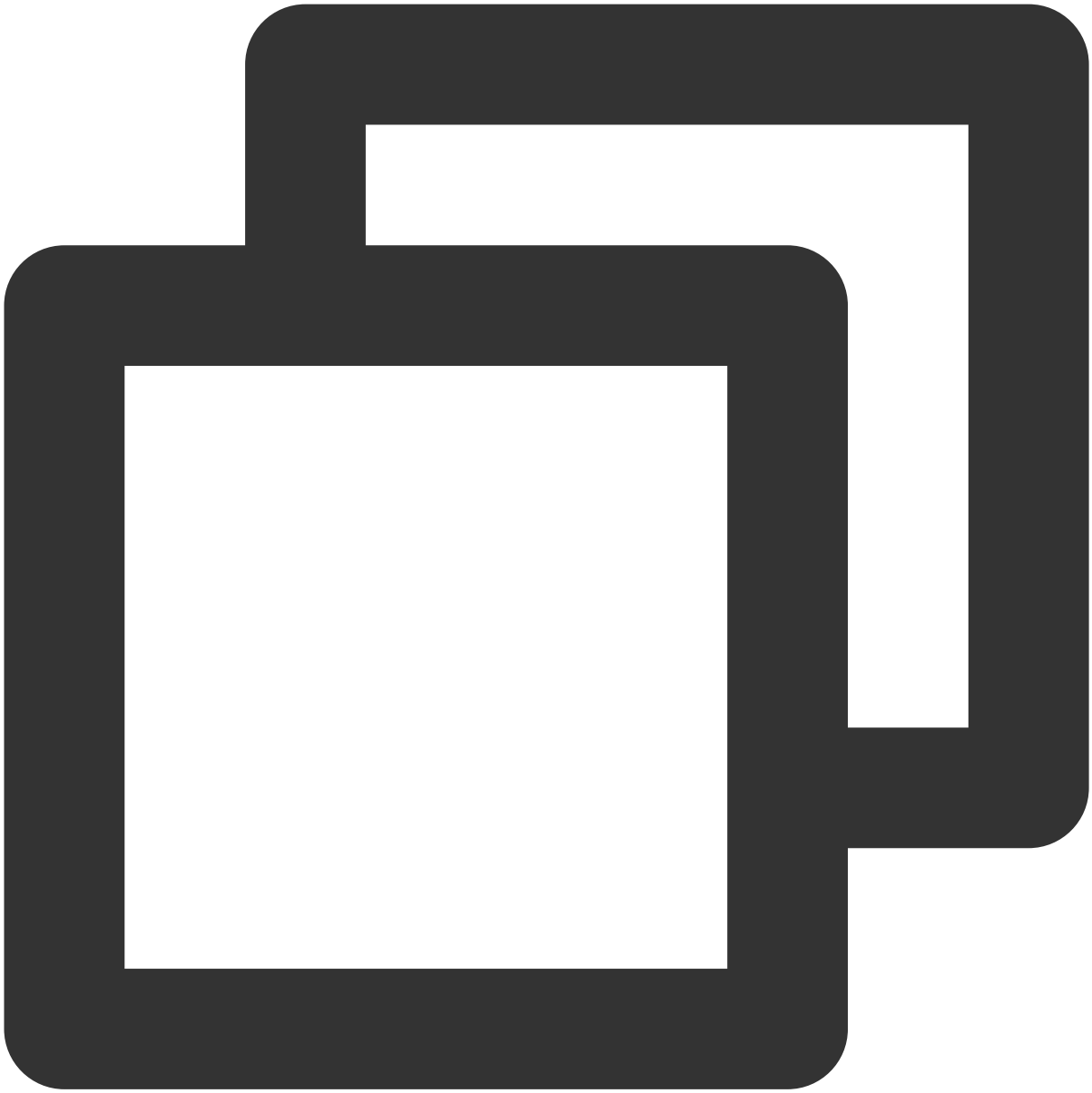
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
message	NSString	ログ情報。

ルームイベントのコールバック

onRoomDestroy

ルーム破棄のコールバック。管理者がルームを解散するとき、ルーム内の全ユーザーはこの通知を受信します。



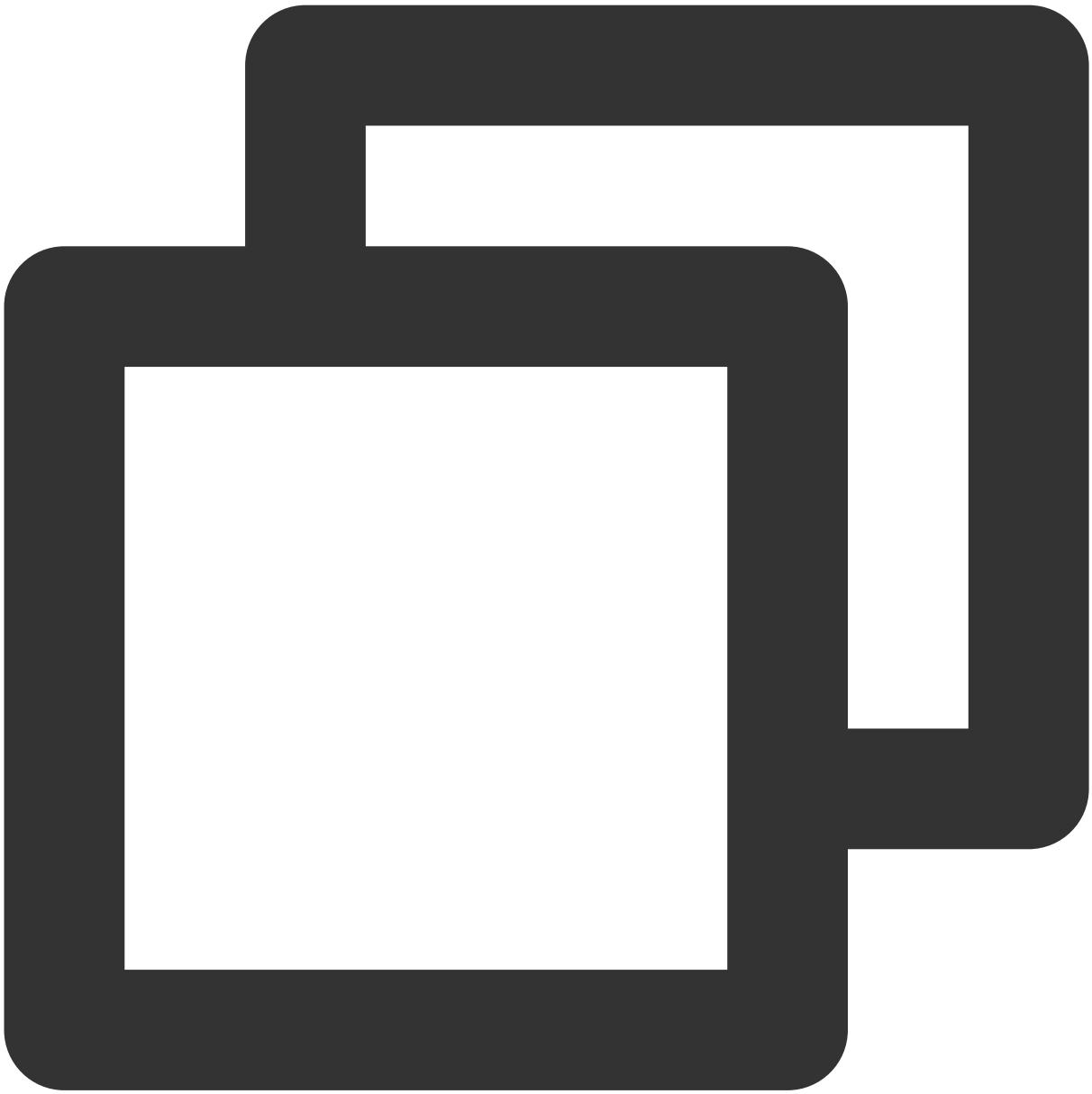
```
- (void)onRoomDestroy:(NSString *)roomId
NS_SWIFT_NAME(onRoomDestroy(roomId:));
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
roomId	NSString	ルーム ID。

onRoomInfoChange

入室に成功後、このインターフェースをコールバックします。roomInfoの情報は、管理者がルームを作成するときに渡されます。



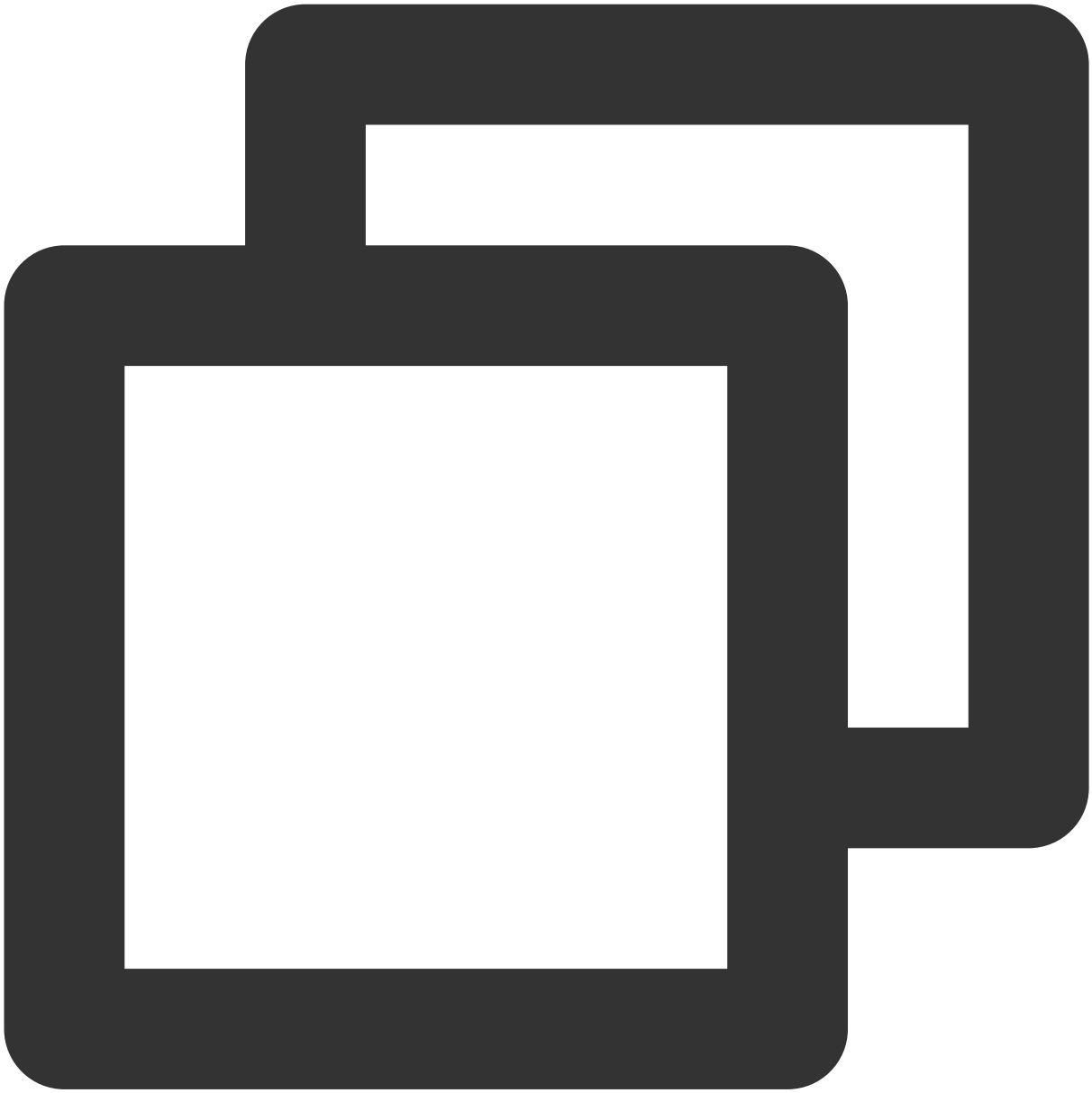
```
- (void)onRoomInfoChange:(VoiceRoomInfo *)roomInfo
NS_SWIFT_NAME(onRoomInfoChange(roomInfo:));
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
roomInfo	VoiceRoomInfo	ルーム情報。

onUserMicrophoneMute

ユーザーのマイクがミュートになっているかどうかのコールバックです。ユーザーがmuteLocalAudioを呼び出すと、ルームの他のユーザーがこの通知を受信します。



```
- (void)onUserMicrophoneMute:(NSString *)userId mute:(BOOL)mute
NS_SWIFT_NAME(onUserMicrophoneMute(userId:mute:));
```

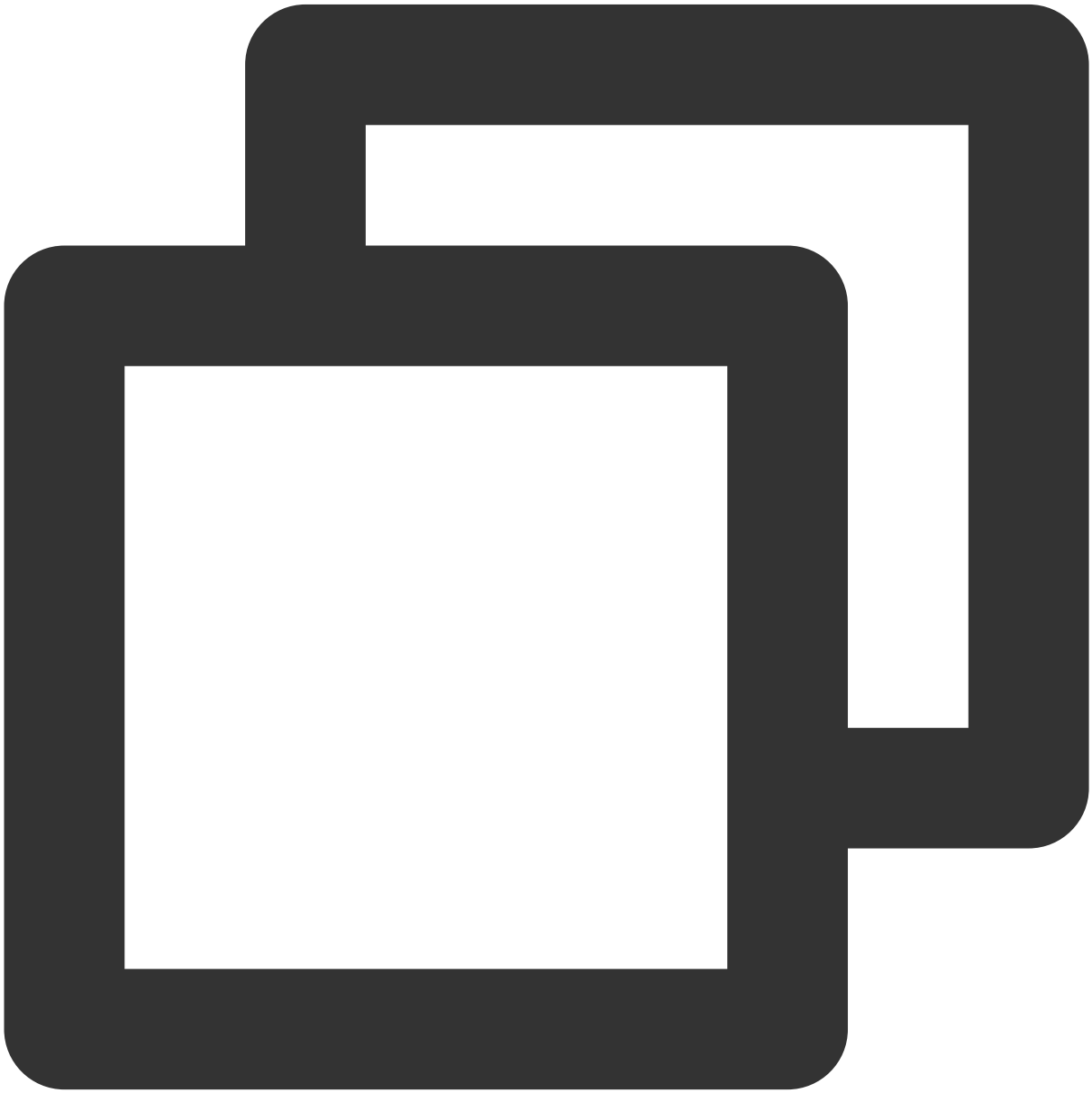
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味

userId	NSString	ユーザーID。
mute	BOOL	YES：マイクミュート； NO：ミュート解除。

onUserVolumeUpdate

音量レベルリマインダを有効にすると、各メンバーの音量を通知します。



```
- (void)onUserVolumeUpdate:(NSArray<TRTCVolumeInfo *> *)userVolumes totalVolume:(NS
NS_SWIFT_NAME(onUserVolumeUpdate(userVolumes:totalVolume:));
```

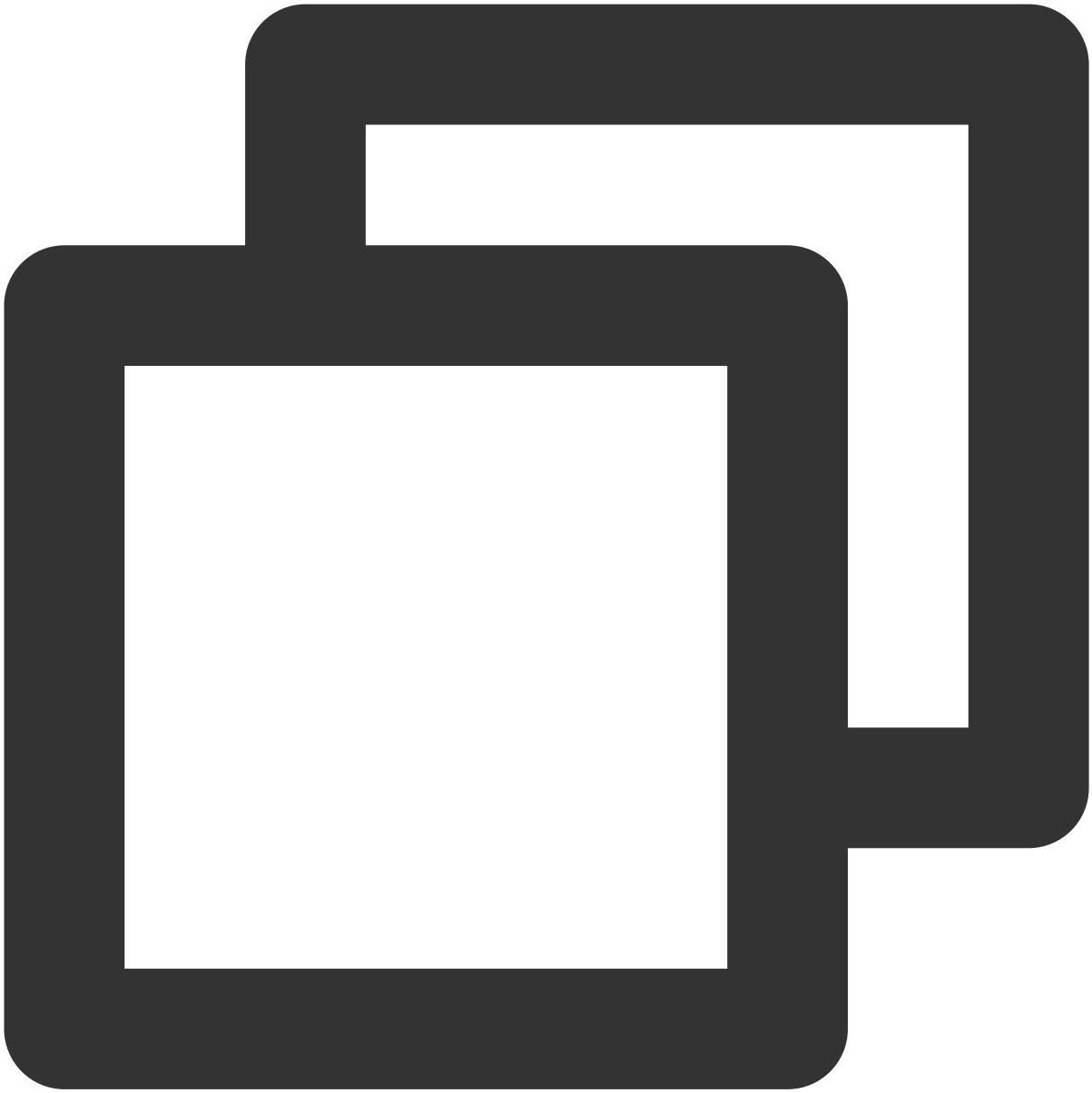
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userVolumes	NSArray	ユーザーリスト。
totalVolume	NSInteger	音量の大きさ。値：0 - 100。

マイクコールバック

onSeatListChange

全量のマイクリストの変更です。全てのマイクリストを含みます。



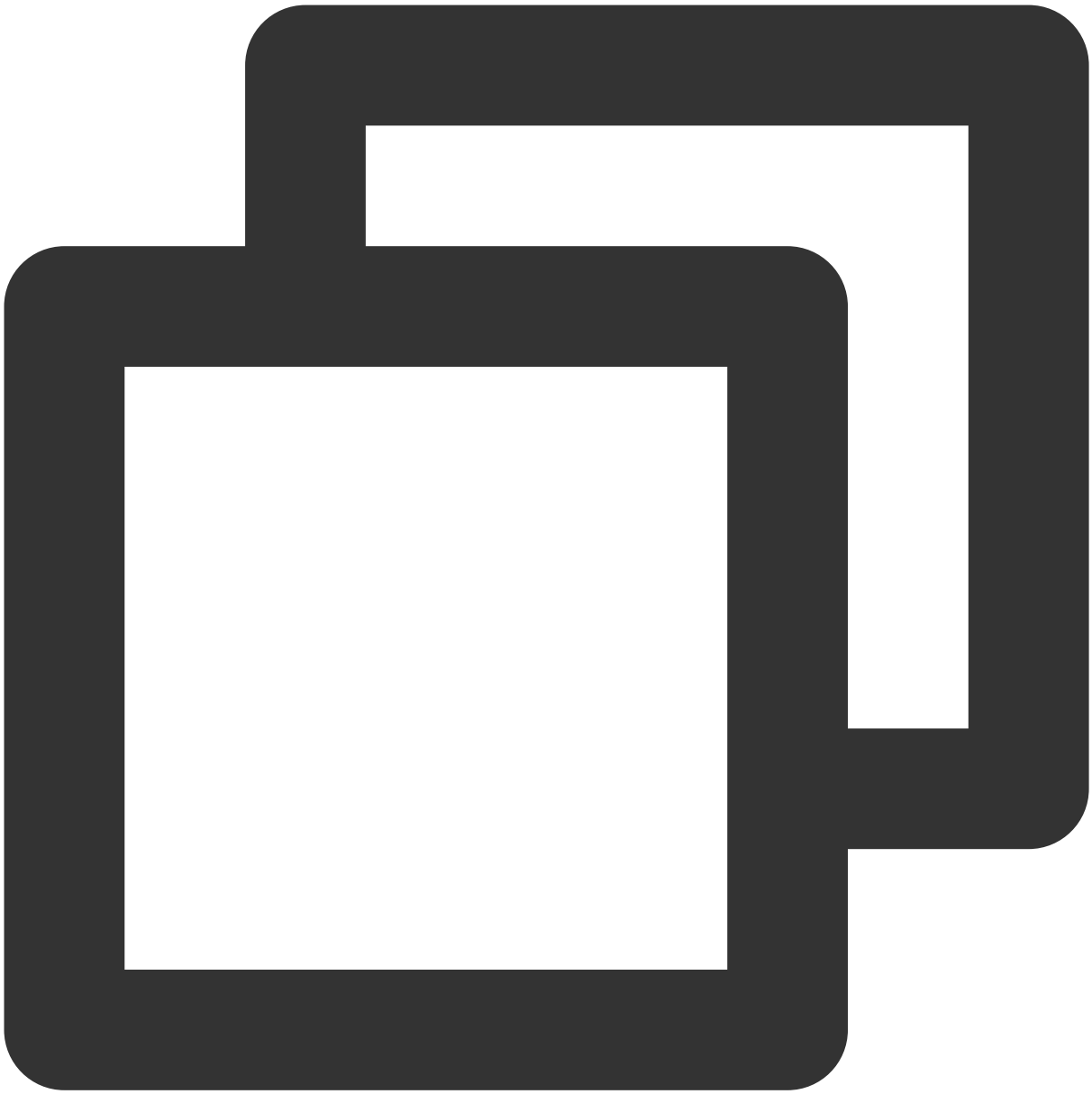
```
- (void)onSeatInfoChange:(NSArray<VoiceRoomSeatInfo *> *)seatInfolist
NS_SWIFT_NAME(onSeatListChange(seatInfoList:));
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
seatInfoList	NSArray<VoiceRoomSeatInfo>	全量のマイクリスト。

onAnchorEnterSeat

発言者のメンバーがいます（ユーザーが発言者になる/管理者が視聴者を発言できるように招待）。



```
- (void)onAnchorEnterSeat:(NSInteger)index
                             user:(VoiceRoomUserInfo *)user
NS_SWIFT_NAME(onAnchorEnterSeat(index:user:));
```

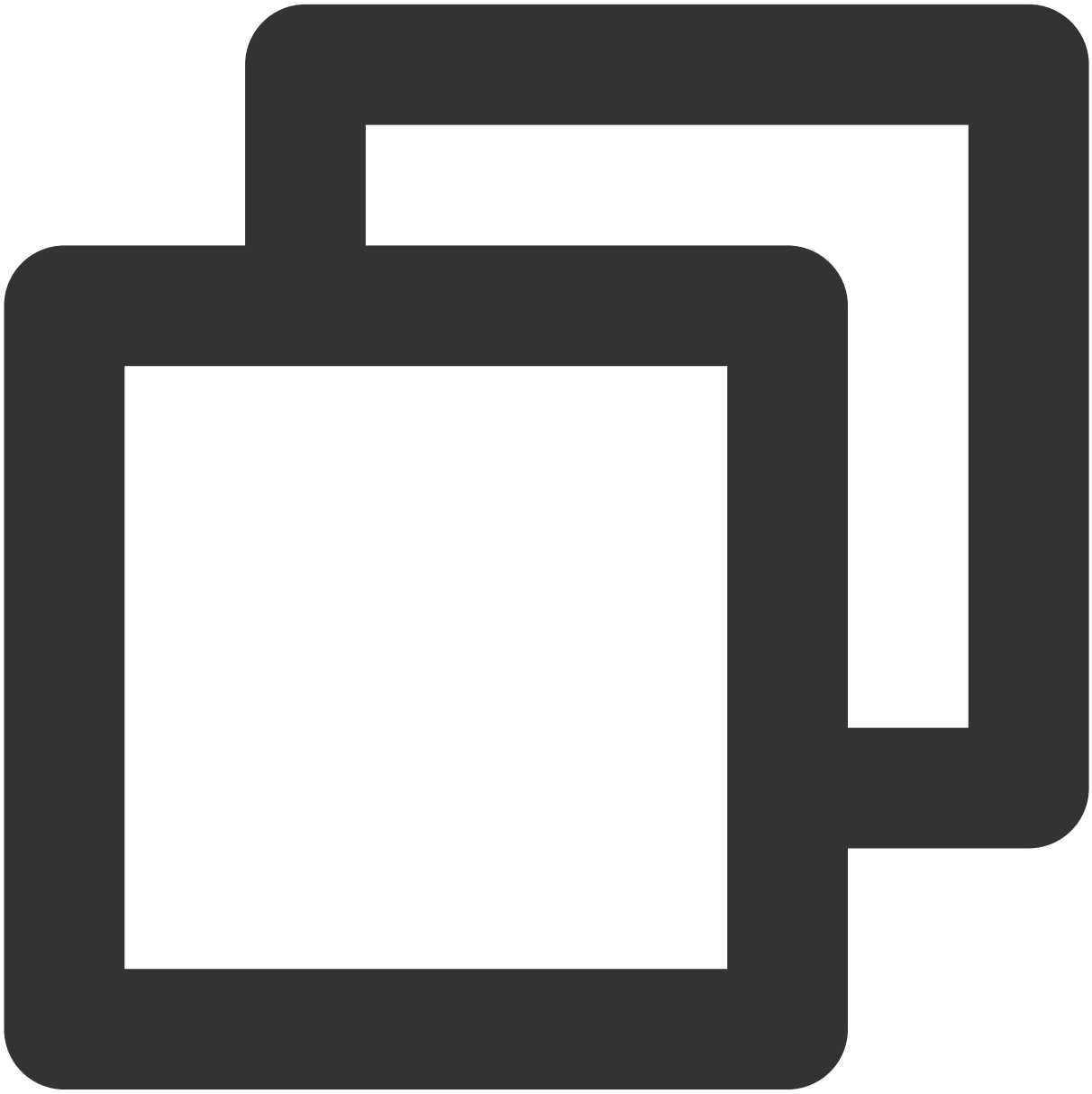
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
index	NSInteger	メンバーがマイク・オンするマイク。

user	VoiceRoomUserInfo	マイク・オンユーザーの詳細情報。

onAnchorLeaveSeat

視聴者のメンバーがいます（ユーザーが視聴者になる/管理者がキックアウトしてマイク・オフ）。



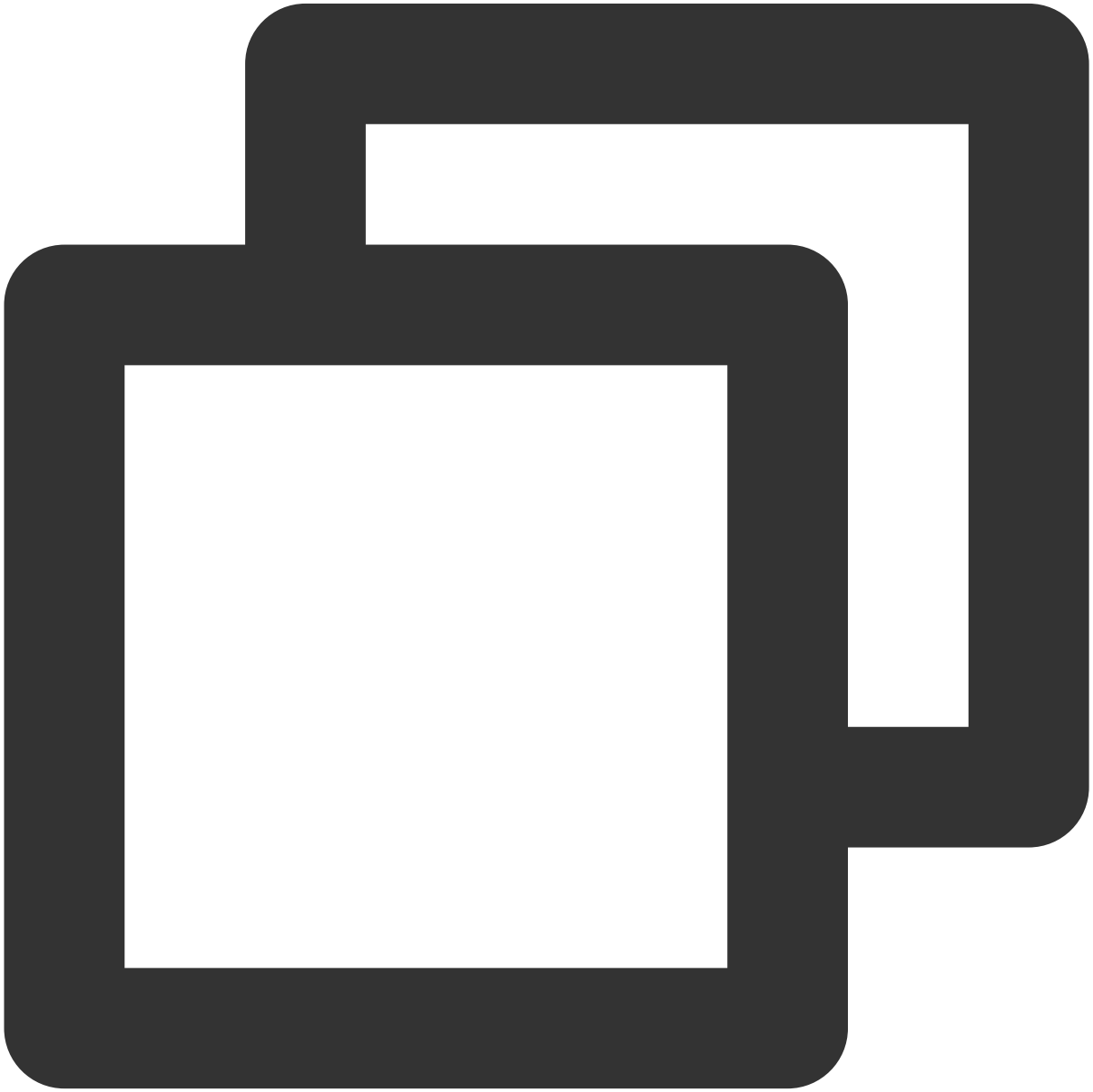
```
- (void)onAnchorLeaveSeat:(NSInteger)index
                     user:(VoiceRoomUserInfo *)user
NS_SWIFT_NAME(onAnchorLeaveSeat(index:user:));
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
index	NSInteger	マイク・オフのマイク。
user	VoiceRoomUserInfo	マイク・オンユーザーの詳細情報。

onSeatMute

管理者のマイクミュート。



```
- (void)onSeatMute:(NSInteger)index
```

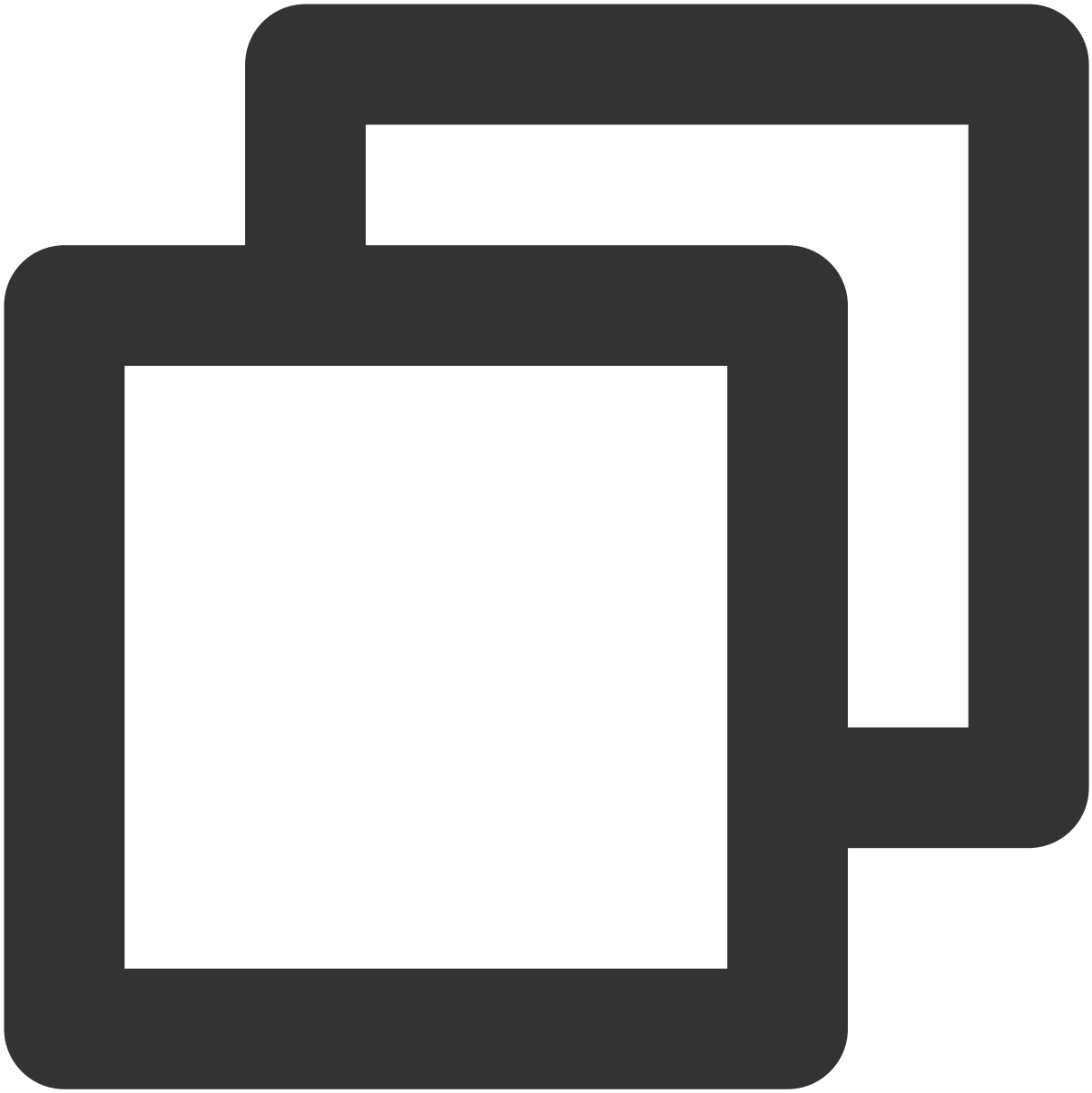
```
                isMute:(BOOL)isMute
NS_SWIFT_NAME(onSeatMute(index:isMute:));
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
index	NSInteger	操作するマイク。
isMute	BOOL	YES：マイクミュート； NO：ミュート解除。

onSeatClose

管理者のマイククローズ。



```
- (void)onSeatClose:(NSInteger)index
            isClose:(BOOL)isClose
NS_SWIFT_NAME(onSeatClose(index:isClose:));
```

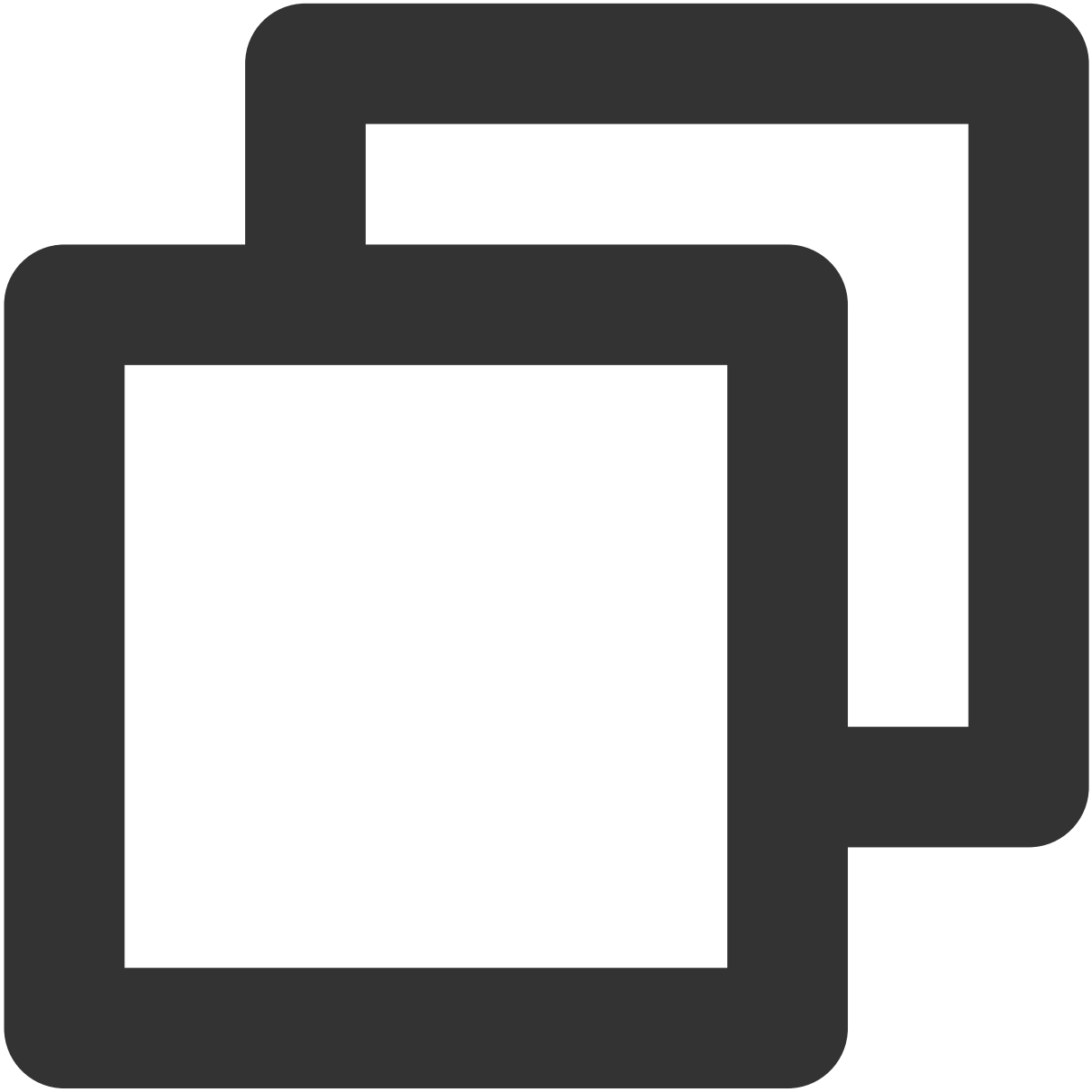
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
index	NSInteger	操作するマイク。
isClose	BOOL	YES：マイクのクローズ； NO： マイクのクローズ解除。

リスナーの入退室イベントのコールバック

onAudienceEnter

リスナー入室通知の受信。



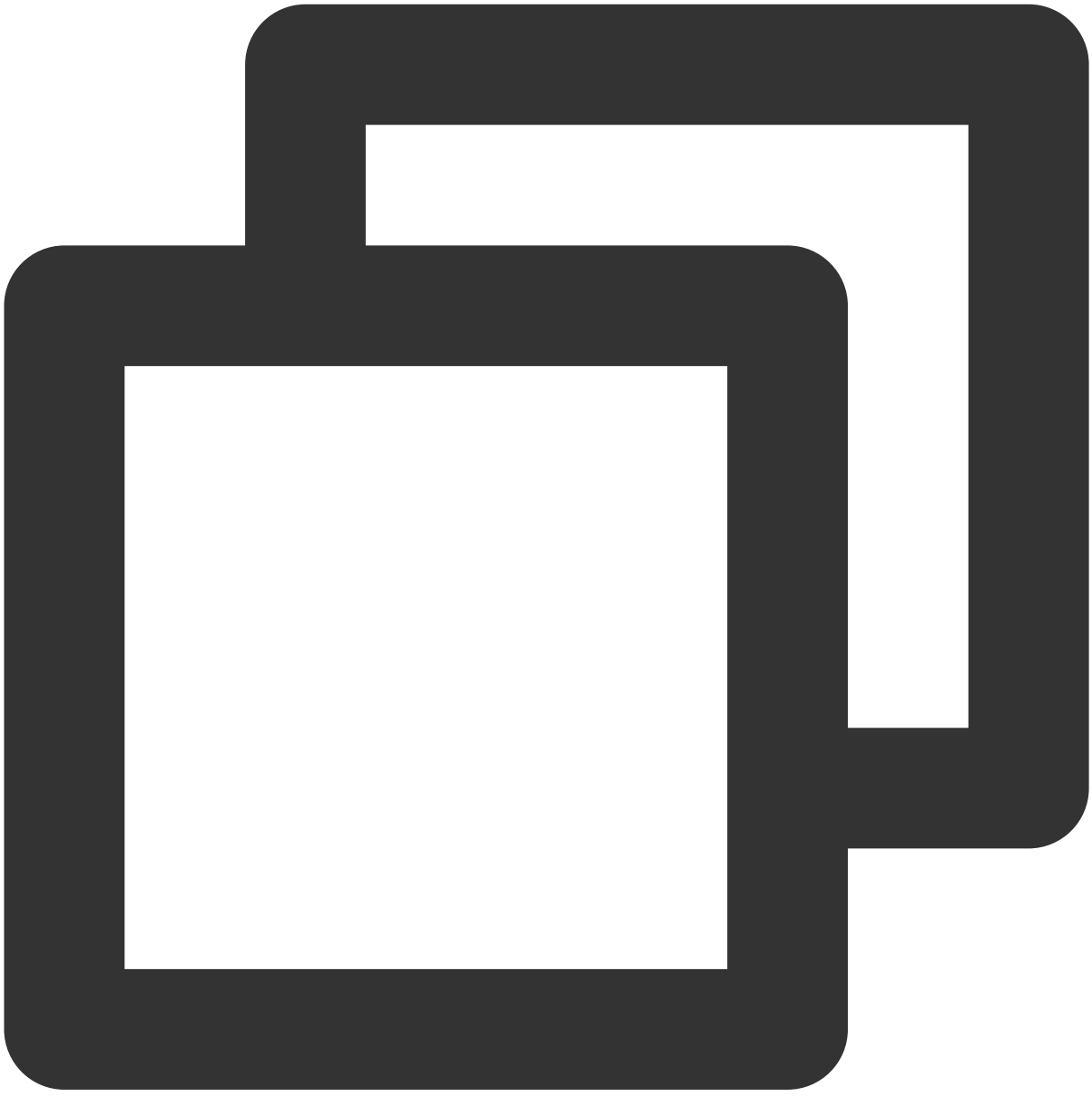
```
- (void)onAudienceEnter:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onAudienceEnter(userInfo:));
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userInfo	VoiceRoomUserInfo	入室したリスナーの情報。

onAudienceExit

リスナー退室通知の受信。



```
- (void)onAudienceExit:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onAudienceExit(userInfo:));
```

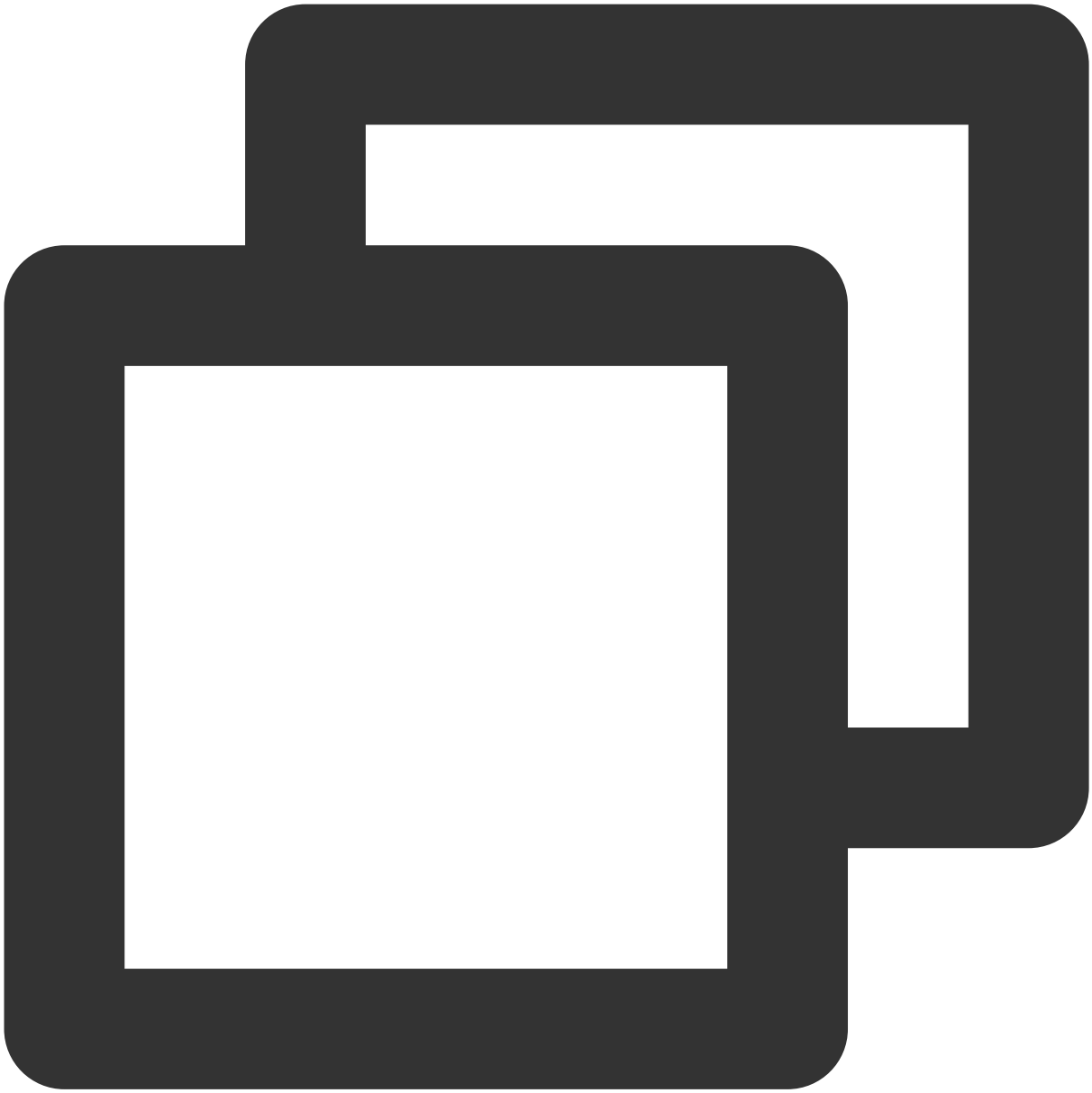
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userInfo	VoiceRoomUserInfo	退室したリスナーの情報。

メッセージイベントのコールバック

onRecvRoomTextMsg

テキストメッセージを受信します。



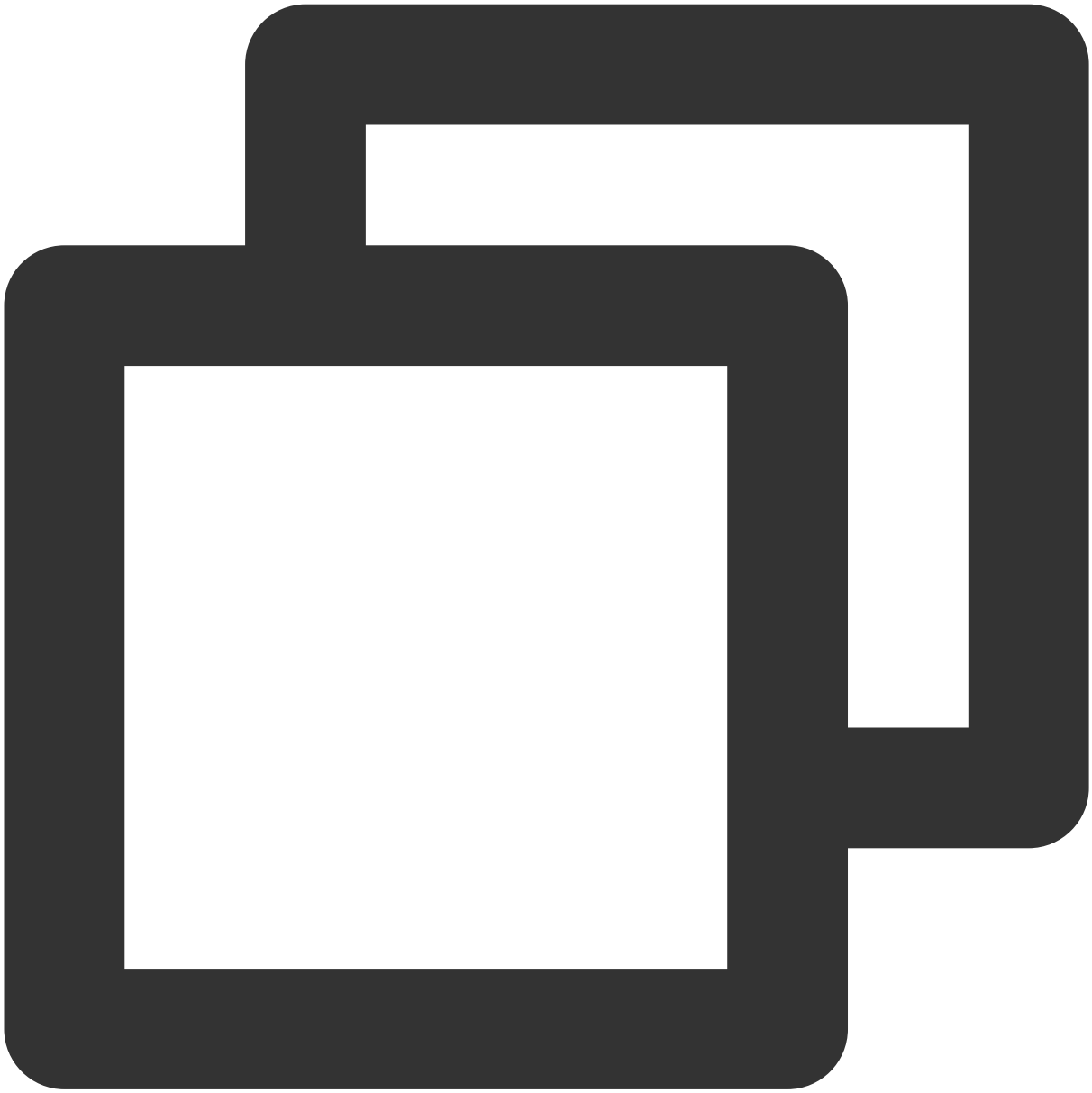
```
- (void)onRecvRoomTextMsg:(NSString *)message
                 userInfo:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomTextMsg(message:userInfo:));
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
message	NSString	テキストメッセージ。
userInfo	VoiceRoomUserInfo	送信者のユーザー情報。

onRecvRoomCustomMsg

カスタムメッセージを受信します。



```
- (void)onRecvRoomCustomMsg:(NSString *)command
                    message:(NSString  *)message
                   userInfo:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomCustomMsg(command:message:userInfo:));
```

パラメータは下表に示すとおりです：

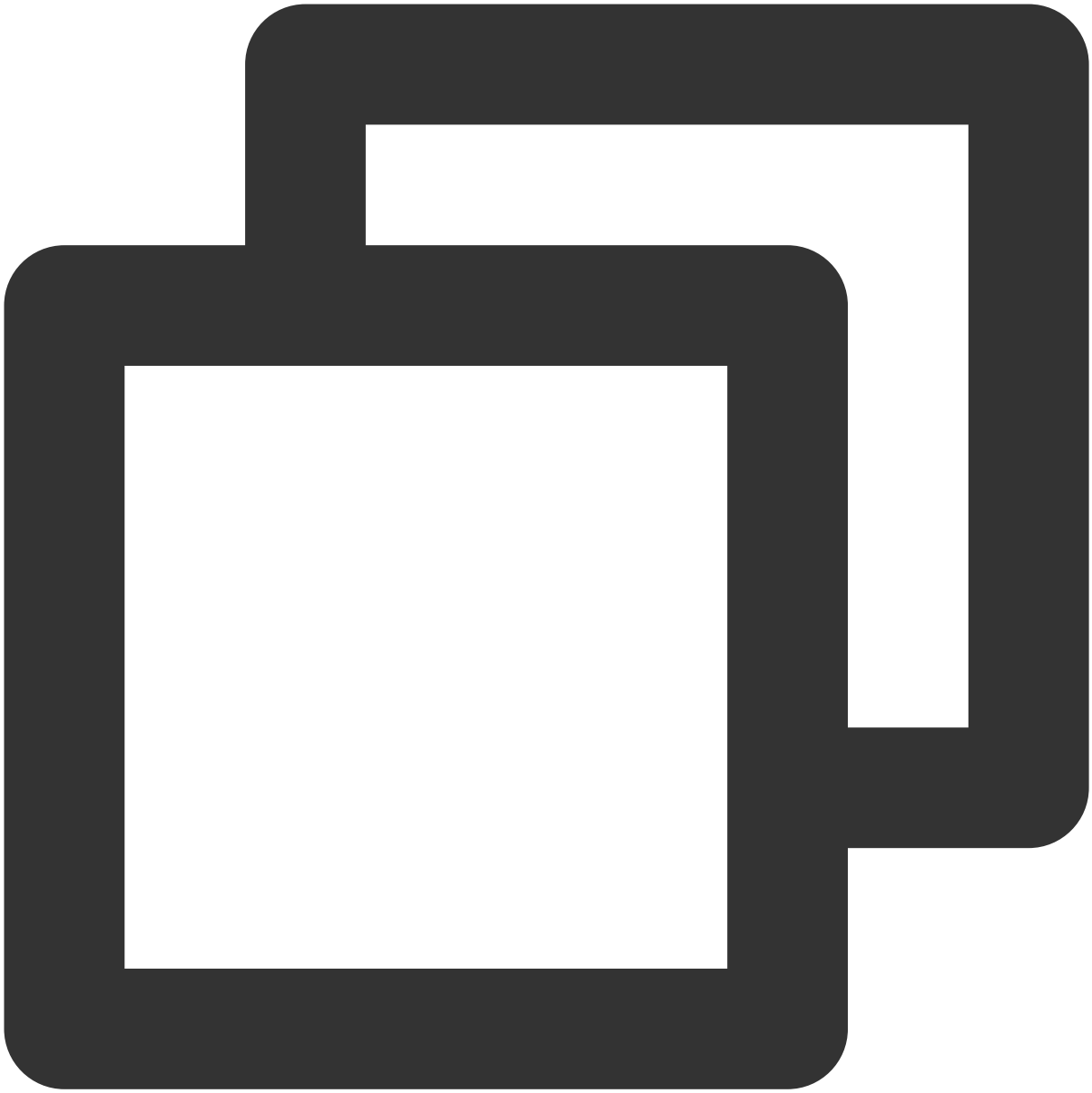
パラメー	タイプ	意味

タ		
command	NSString	コマンドワードです。開発者がカスタマイズするものであり、主にさまざまなメッセージタイプを区別するために使用されます。
message	NSString	テキストメッセージ。
userInfo	VoiceRoomUserInfo	送信者のユーザー情報。

招待シグナリングイベントのコールバック

onReceiveNewInvitation

新規招待リクエストの受信。



```
- (void)onReceiveNewInvitation:(NSString *)identifier
                       inviter:(NSString *)inviter
                           cmd:(NSString *)cmd
                       content:(NSString *)content
NS_SWIFT_NAME(onReceiveNewInvitation(id:inviter:cmd:content:));
```

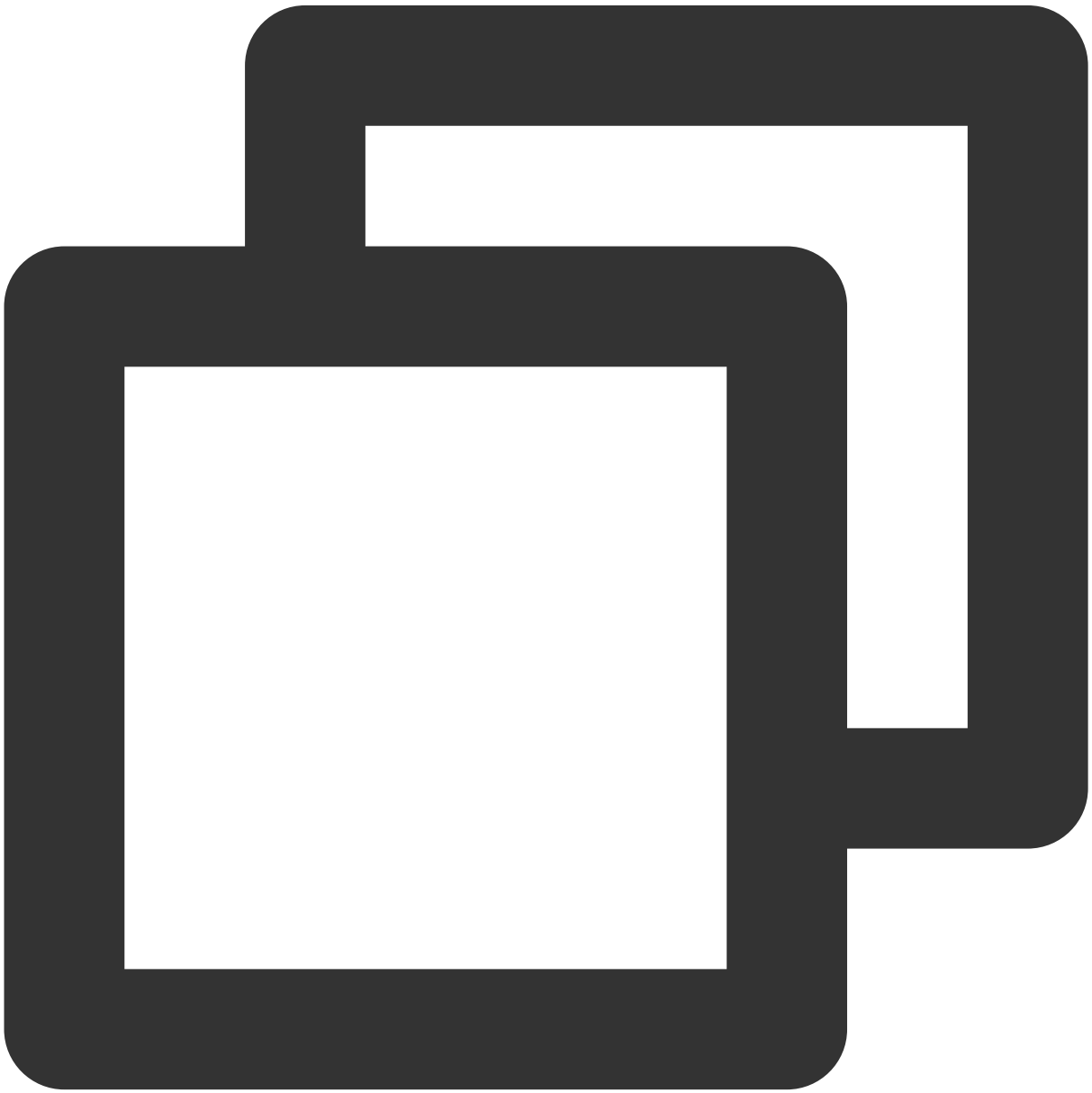
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
id	NSString	招待ID。

inviter	NSString	招待者のユーザーID。
cmd	NSString	開発者がカスタマイズする業務指定のコマンドワードです。
content	NSString	業務指定のコンテンツ。

onInviteeAccepted

被招待者が招待に同意。



```
- (void)onInviteeAccepted:(NSString *)identifier
```

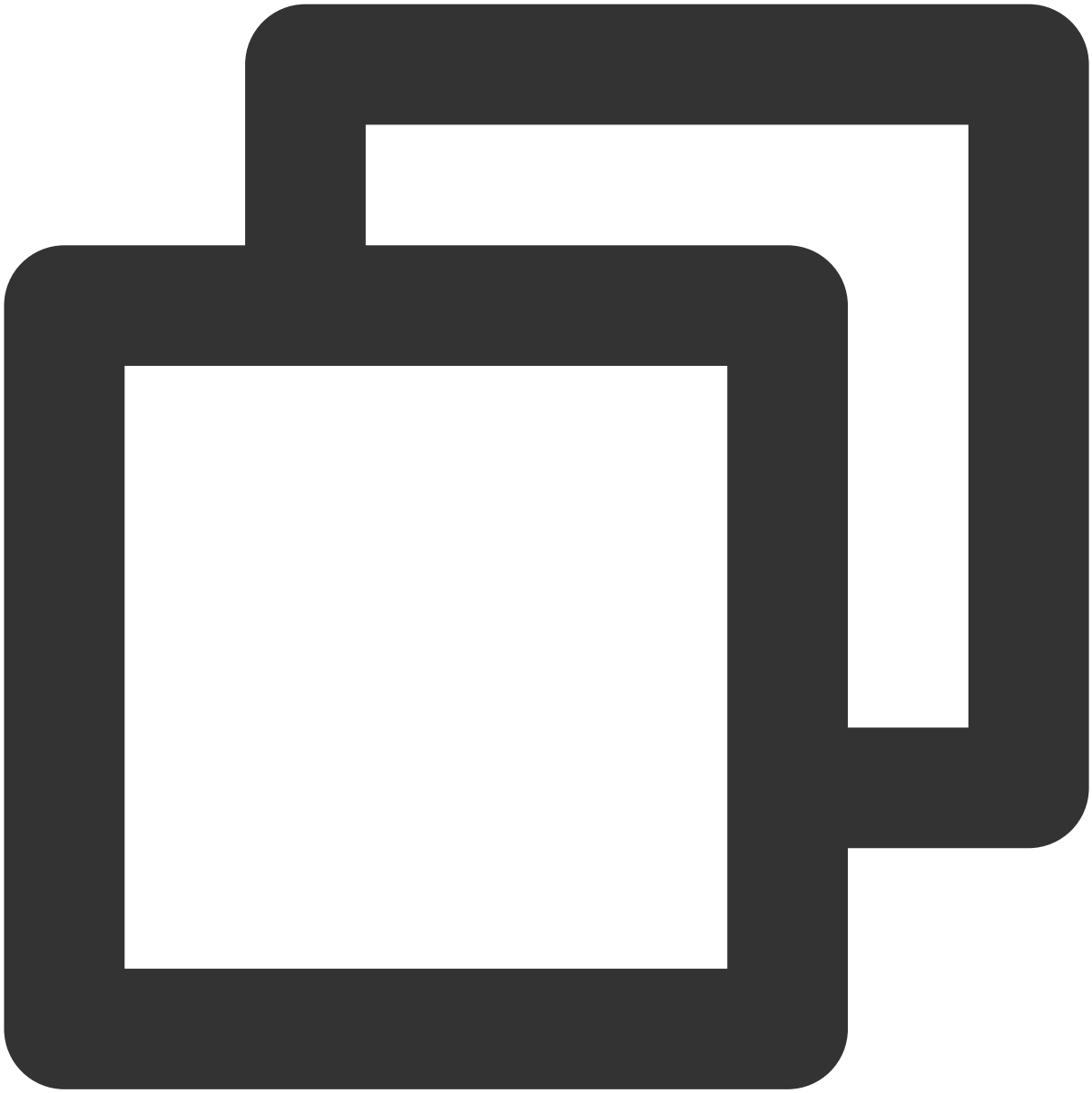
```
                    invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeAccepted(id:invitee:));
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
id	NSString	招待ID。
invitee	NSString	被招待者のユーザーID。

onInviteeRejected

被招待者による招待の拒否。



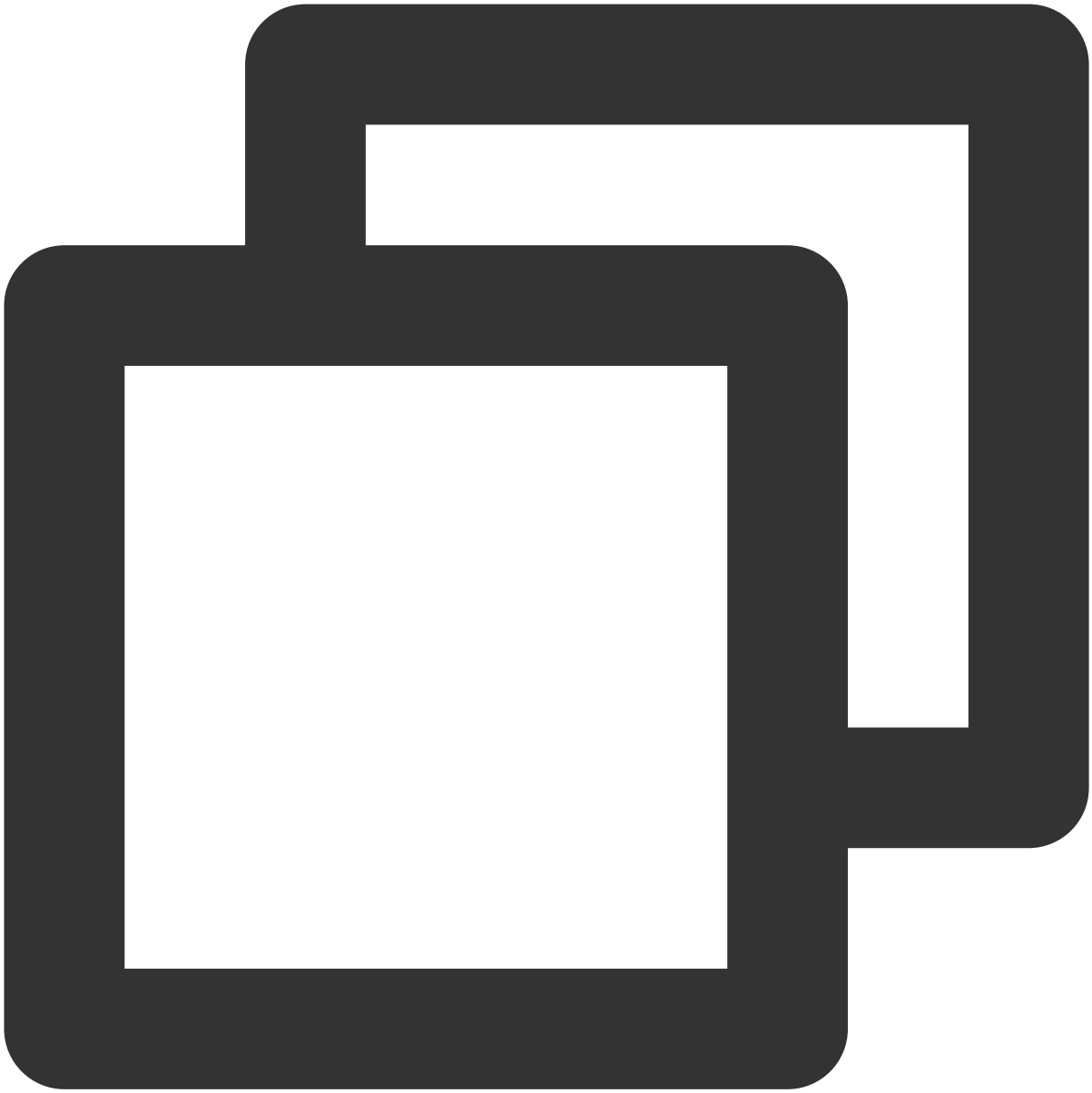
```
- (void)onInviteeRejected:(NSString *)identifier
                  invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeRejected(id:invitee:));
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
id	NSString	招待ID。
invitee	NSString	被招待者のユーザーID。

onInvitationCancelled

招待者が招待を取り消し。



```
- (void)onInvitationCancelled:(NSString *)identifier
                      invitee:(NSString *)invitee NS_SWIFT_NAME(onInvitationCancell
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
id	NSString	招待ID。

inviter	NSString	招待者のユーザーID。

TRTCVoiceRoom (Android)

最終更新日：：2024-07-19 15:35:35

TRTCVoiceRoomは、Tencent CloudのTencent Real-Time Communication (TRTC)およびIMサービスを基に組み合わせたコンポーネントで、以下の機能をサポートしています：
管理者が新しいボイスチャットルームを作成して配信を開始し、リスナーがボイスチャットルームに参加して視聴/インタラクティブなコミュニケーションを行います。
管理者は、リスナーを招待してマイク・オンにしたり、マイク・オンのキャスターをキックアウトしたりすることもできます。
管理者はまた、座席をクローズすることができ、その他のリスナーはマイク・オンを申請することができなくなります。
リスナーはマイク・オンを申請して、マイク・オンのキャスターになり、他者と音声インタラクションを行うことができます。また、いつでもマイク・オフにして、通常のリスナーになることも可能です。
各種のテキストメッセージやカスタマイズメッセージの発信をサポートし、カスタムメッセージは弾幕、「いいね」、ギフトなどに使用することきます。
説明：
TUIKitシリーズコンポーネントはTencent CloudのTRTCとIMという2つの基本的なPaaSサービスを同時に使用し、TRTCをアクティブにした後、IMサービスを同期的にアクティブにすることができます。IMサービスの課金ルールの詳細については、Instant Messagingの料金説明をご参照ください。TRTCをアクティブにすると、デフォルトでは、100DAUまでサポートするIM SDK体験版もアクティブにします。
TRTCVoiceRoomは、オープンソースのClassであり、Tencent Cloudの2つのクローズソースであるSDKに依存しています。具体的な実装プロセスは ボイスチャットルーム（Android）をご参照ください。
TRTC SDK：TRTC SDKを低遅延のボイスチャットコンポーネントとして使用しています。
IM SDK：IM SDKのAVChatroomを使用してチャットルーム機能を実装します。同時にIMの属性インターフェースによって、マイクリストなどのルーム情報を保存し、招待シグナリングはマイク・オン/ピックのリクエストに用いることができます。

TRTCVoiceRoom API 概要

SDK基本関数

API	説明
sharedInstance	シングルトンオブジェクトを取得します。
destroySharedInstance	シングルトンオブジェクトを廃棄します。
setDelegate	イベントコールバックを設定します。

setDelegateHandler	イベントのコールバックが配置されているスレッドを設定します。
login	ログイン。
logout	ログアウト。
setSelfProfile	個人情報を修正します。

ルーム関連インターフェース関数

API	説明
createRoom	ルームの作成（管理者が呼び出し）。ルームが存在しない場合は、システムが新しいルームを自動的に作成します。
destroyRoom	ルームの破棄（管理者が呼び出し）。
enterRoom	入室（リスナーが呼び出し）。
exitRoom	退室（リスナーが呼び出し）。
getRoomInfoList	ルームリストの詳細情報を取得します。
getUserInfoList	指定されたuserIdのユーザー情報を取得します。 nullの場合は、ルーム内全員の情報を取得します。

マイク管理インターフェース

API	説明
enterSeat	ユーザーが発言者になる（リスナー側/管理者ともに呼び出し可）。
moveSeat	マイクの移動（マイク・オンするキャスター側で呼び出せます）。
leaveSeat	ユーザーが視聴者になる（キャスターが呼び出し）。
pickSeat	視聴者が発言できるように招待（管理者が呼び出し）。
kickSeat	キックアウトしてマイク・オフ（管理者が呼び出し）。
muteSeat	任意のマイクのミュート/ミュート解除（管理者が呼び出し）。
closeSeat	任意のマイクのクローズ/解除（管理者が呼び出し）。

ローカルのオーディオ操作インターフェース

API	説明

startMicrophone	マイクの集音開始。
stopMicrophone	マイクの集音停止。
setAudioQuality	音質の設定。
muteLocalAudio	ローカルオーディオミュートの開始/停止。
setSpeaker	スピーカーの起動設定。
setAudioCaptureVolume	マイクの集音音量設定。
setAudioPlayoutVolume	再生音量の設定。
setVoiceEarMonitorEnable	インイヤーモニタリングのオン/オフ。

リモートユーザー音声操作インターフェース

API	説明
muteRemoteAudio	指定メンバーをミュート/ミュート解除。
muteAllRemoteAudio	全メンバーをミュート/ミュート解除。

BGMサウンドエフェクト関連インターフェース

API	説明
getAudioEffectManager	BGMサウンドエフェクト管理オブジェクトTXAudioEffectManagerを取得します。

メッセージ送信関連インターフェース

API	説明
sendRoomTextMsg	ルーム内でのテキストメッセージのブロードキャスト。通常、弾幕によるチャットに使用します。
sendRoomCustomMsg	カスタマイズしたテキストメッセージを送信します。

招待シグナリング関連インターフェース

API	説明
sendInvitation	ユーザーに招待を送信。

acceptInvitation	招待の同意。
rejectInvitation	招待の辞退。
cancelInvitation	招待の取り消し。

TRTCVoiceRoomDelegate API概要

一般的なイベントコールバック

API	説明
onError	エラーのコールバック。
onWarning	警告のコールバック。
onDebugLog	Logコールバック。

ルームイベントのコールバック

API	説明
onRoomDestroy	ルームが廃棄された時のコールバック。
onRoomInfoChange	ボイスチャットルーム情報変更のコールバック。
onUserVolumeUpdate	ユーザー通話音量のコールバック。

マイク変更コールバック

API	説明
onSeatListChange	全量のマイクリストの変更。
onAnchorEnterSeat	発言者のメンバーがいます（ユーザーが発言者になります/管理者が視聴者を発言できるように招待）。
onAnchorLeaveSeat	視聴者のメンバーがいます（ユーザーが視聴者になる/管理者がキックアウトしてマイク・オフ）。
onSeatMute	管理者のマイクミュート。
onUserMicrophoneMute	ユーザーのマイクがミュートされているかどうか。
onSeatClose	管理者のマイククローズ。

リスナーの入退室イベントのコールバック

API	説明
onAudienceEnter	リスナー入室通知の受信。
onAudienceExit	リスナー退室通知の受信。

メッセージイベントのコールバック

API	説明
onRecvRoomTextMsg	テキストメッセージを受信します。
onRecvRoomCustomMsg	カスタムメッセージを受信します。

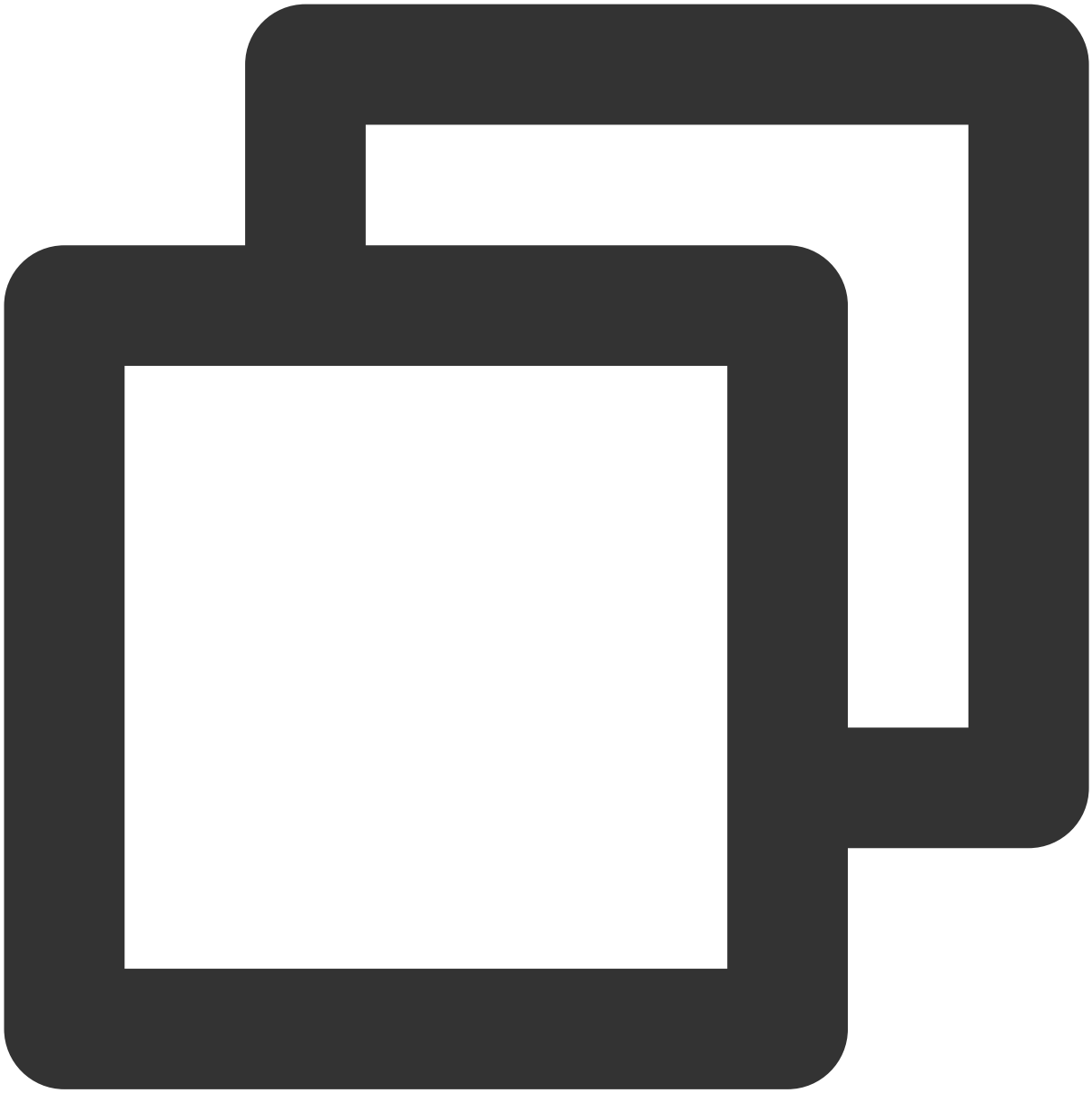
シグナリングイベントのコールバック

API	説明
onReceiveNewInvitation	新規招待リクエストの受信。
onInviteeAccepted	被招待者が招待に同意。
onInviteeRejected	被招待者が招待を拒否。
onInvitationCancelled	招待者が招待を取り消し。

SDK基本関数

sharedInstance

TRTCVoiceRoom シングルトンオブジェクトを取得します。



```
public static synchronized TRTCVoiceRoom sharedInstance(Context context);
```

パラメータは下表に示すとおりです：

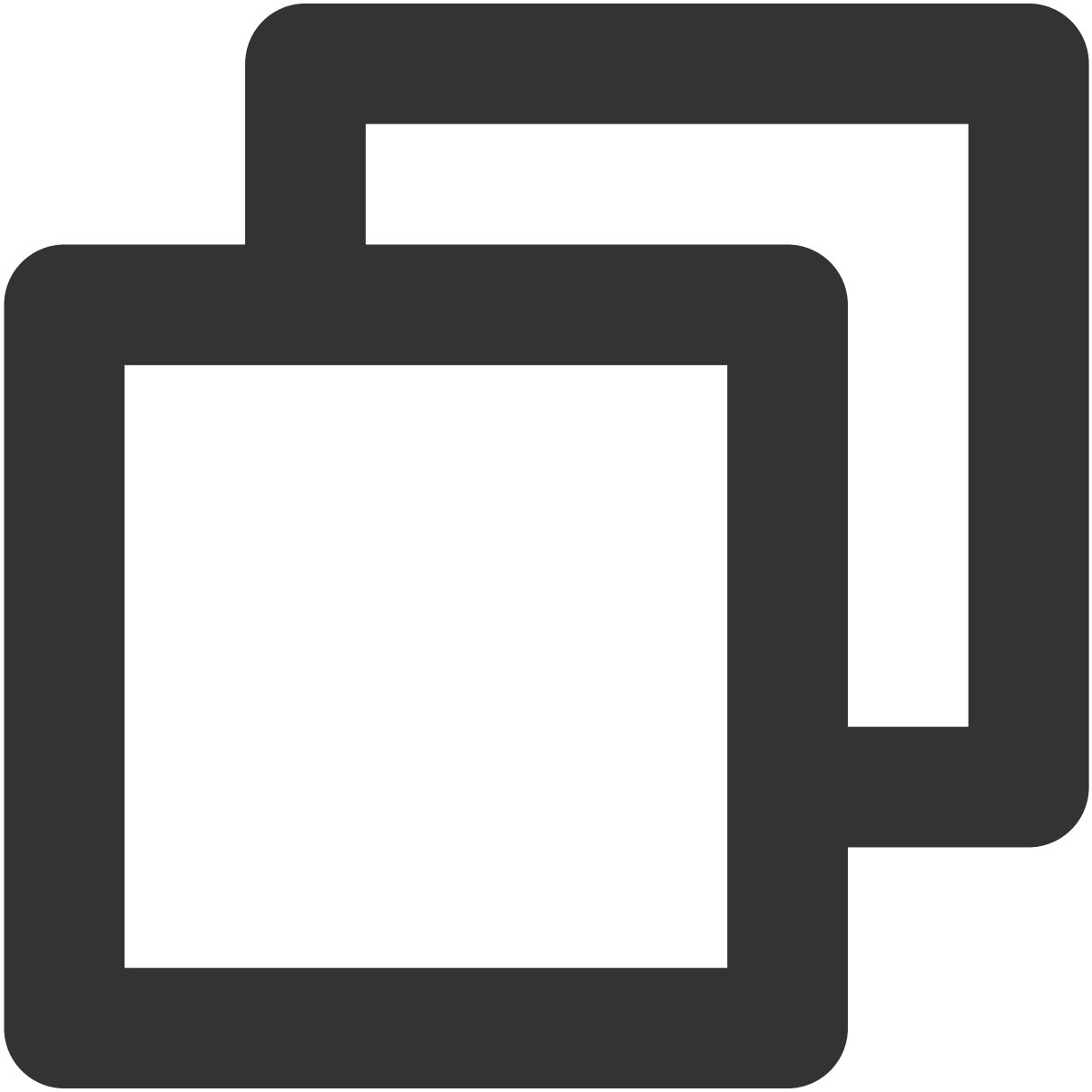
パラメータ	タイプ	意味
context	Context	Androidコンテキスト。内部ではApplicationContextに変換してシステムAPIの呼び出しに使用します

destroySharedInstance

TRTCVoiceRoom シングルトンオブジェクトを破棄します。

説明：

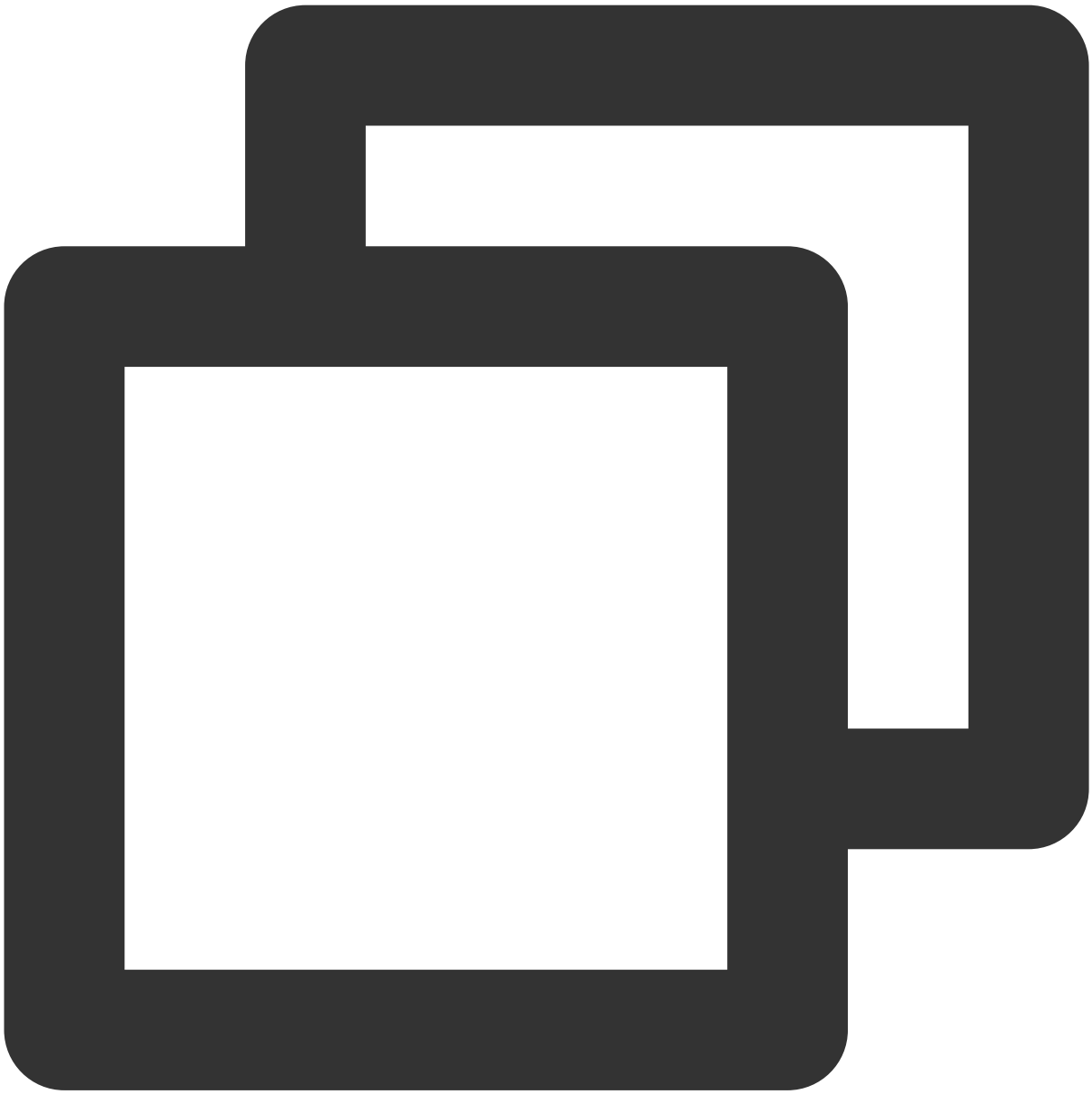
インスタンスの破棄後は、外部にキャッシュされた TRTCVoiceRoomインスタンスの再使用ができません。改めて sharedInstance を呼び出し、インスタンスを新規取得してください。



```
public static void destroySharedInstance();
```

setDelegate

TRTCVoiceRoom イベントコールバック、 TRTCVoiceRoomDelegate を介して TRTCVoiceRoom の各種ステータス通知を受け取ることができます。



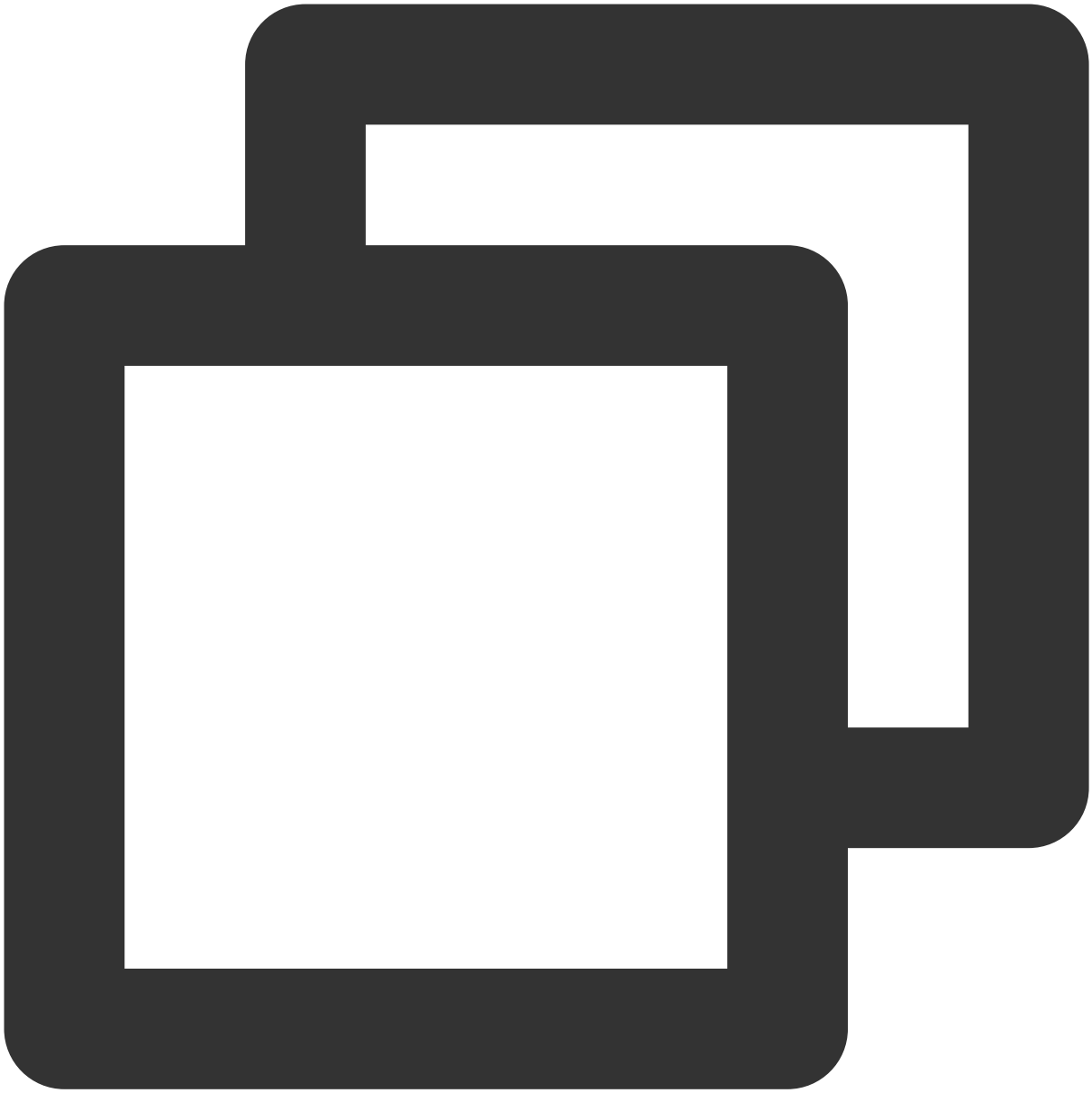
```
public abstract void setDelegate(TRTCVoiceRoomDelegate delegate);
```

説明：

setDelegate は、 TRTCVoiceRoom のプロキシコールバックです。

setDelegateHandler

イベントコールバックが配置されているスレッドを設定します。



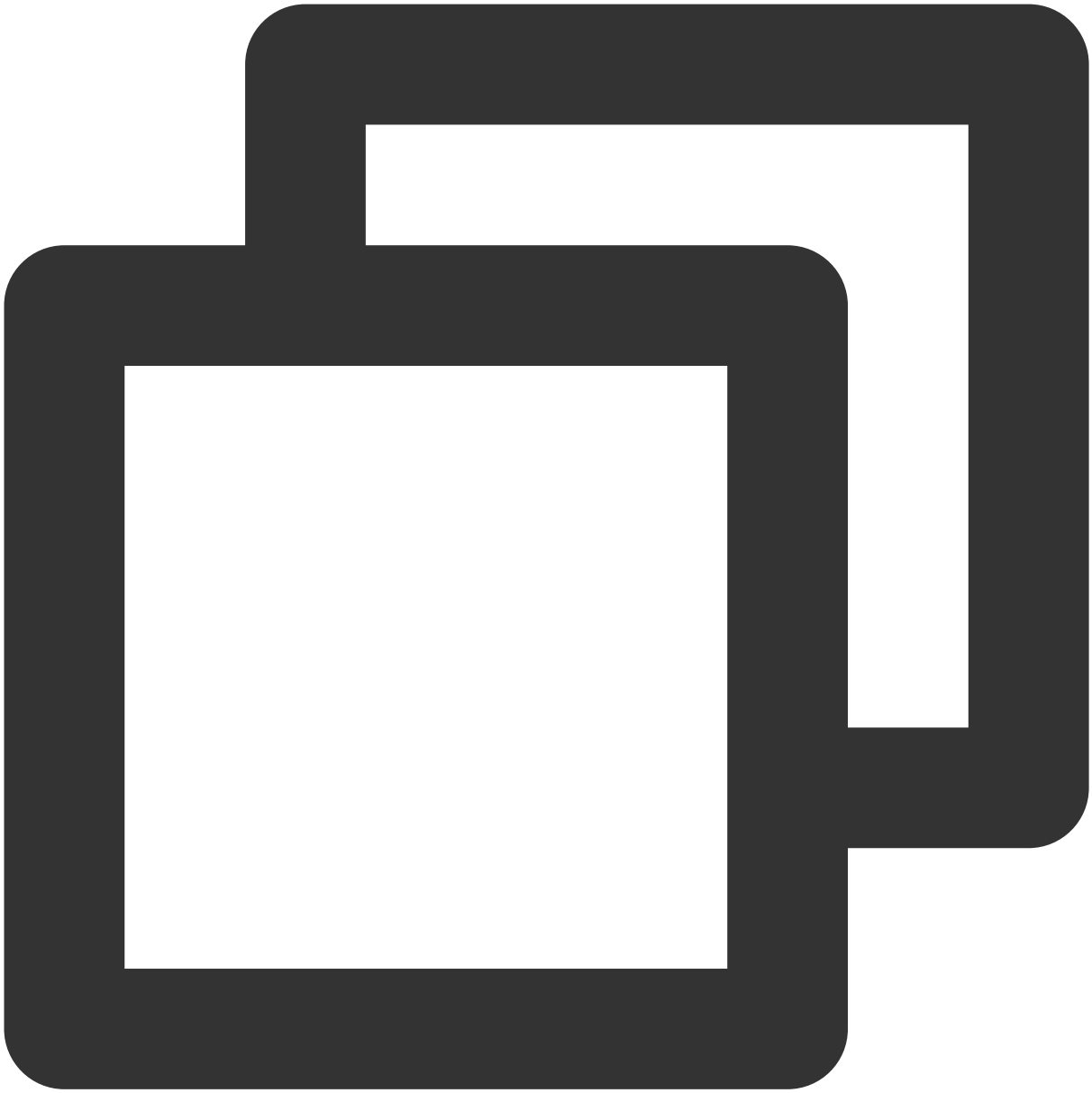
```
public abstract void setDelegateHandler(Handler handler);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
handler	Handler	TRTCChatSalonの各種ステータス通知は、指定したhandlerスレッドに発信されます。

login

ログイン。



```
public abstract void login(int sdkAppId,
 String userId, String userSig,
TRTCVoiceRoomCallback.ActionCallback callback);
```

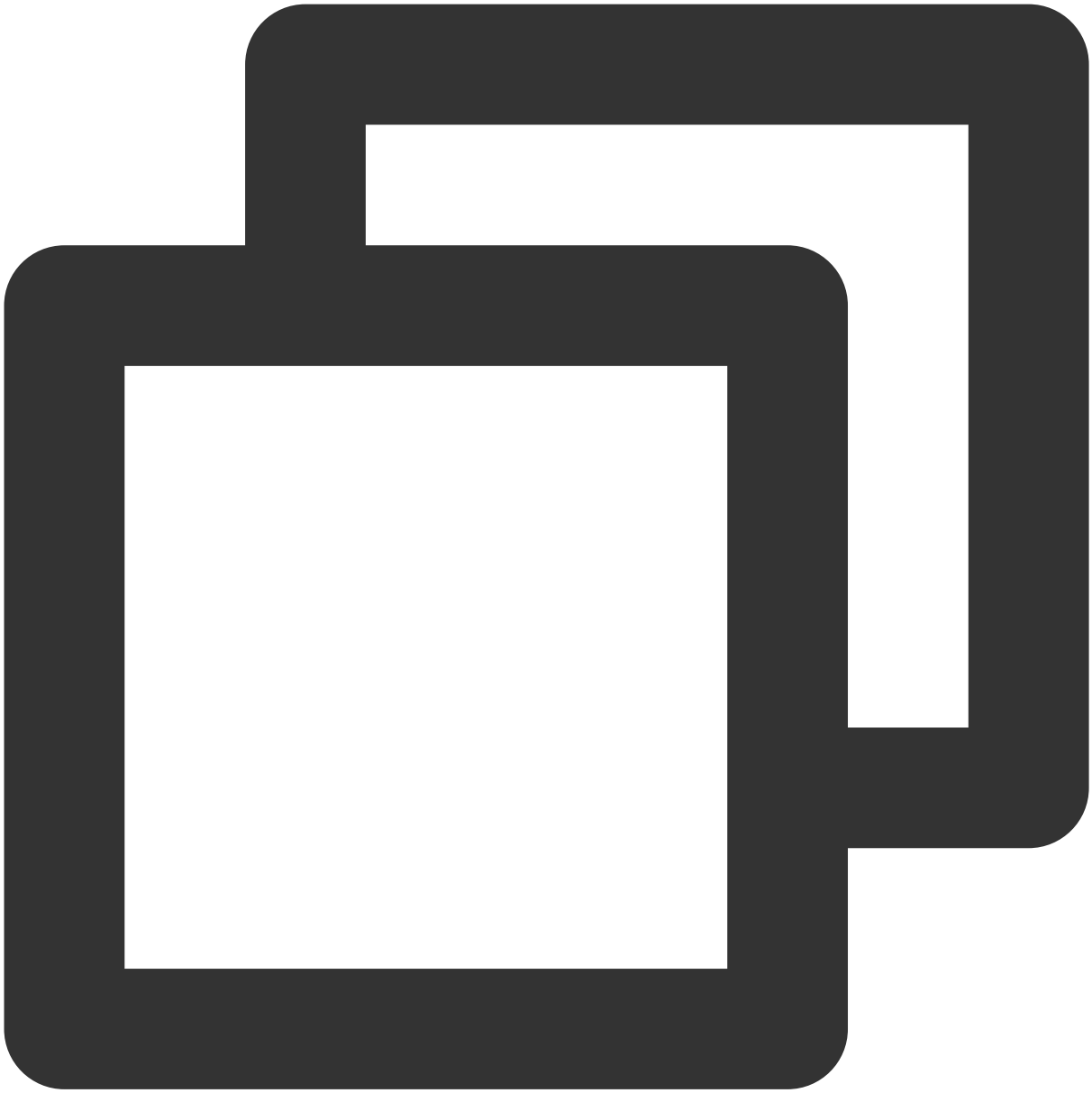
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味

sdkAppId	int	**TRTCコンソール >**アプリケーション管理> アプリケーション情報でSDKAppIDを確認できます。
userId	String	現在のユーザーID。文字列タイプであり、英語のアルファベット（a-zとA-Z）、数字（0-9）、ハイフン（-）とアンダーライン（_）のみ使用できます。
userSig	String	Tencent Cloudによって設計されたセキュリティ保護署名。取得方法については、 UserSigの計算、使用方法をご参照ください。
callback	ActionCallback	ログインのコールバック。成功時にcodeは0になります。

logout

ログアウト。



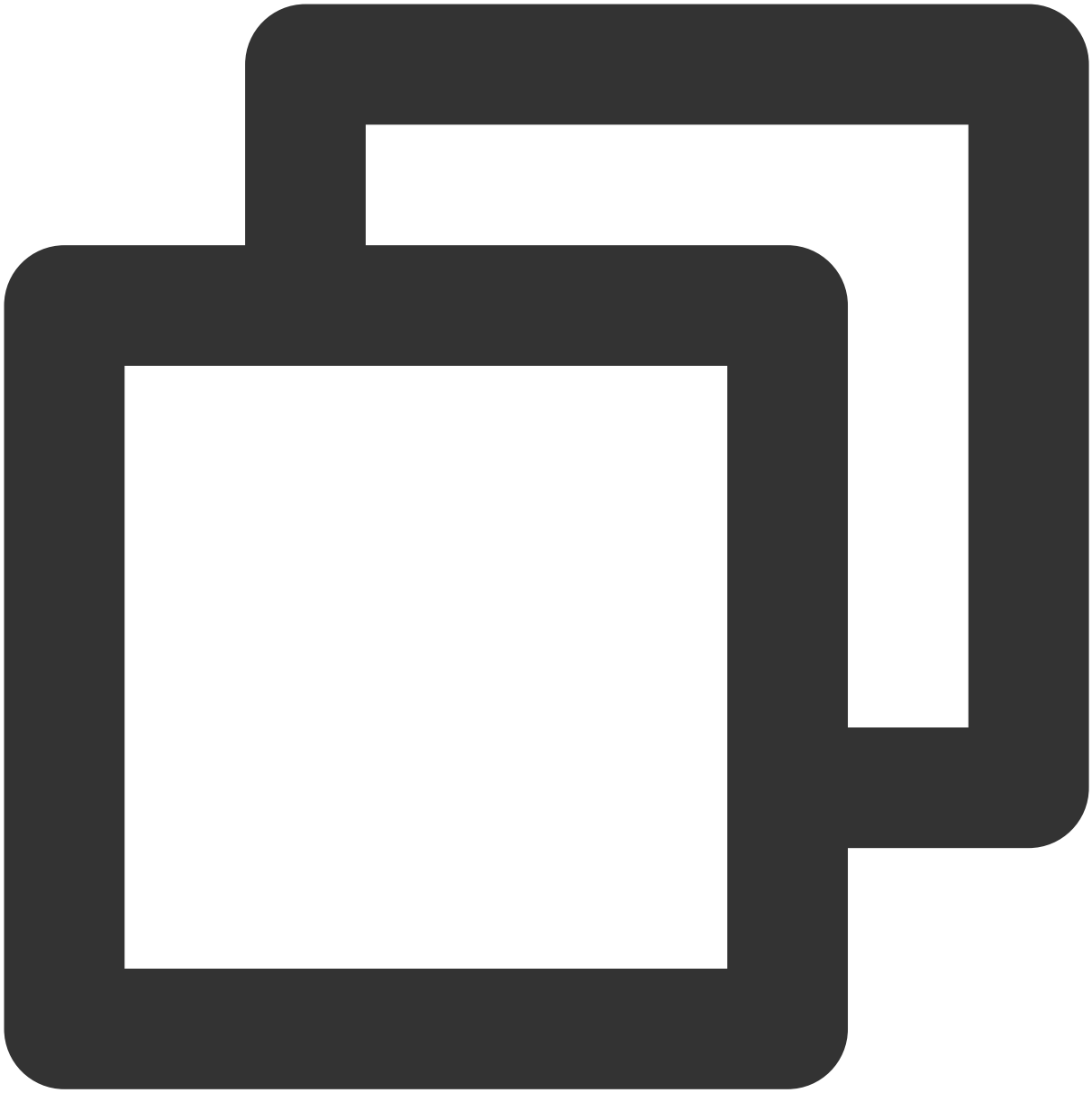
```
public abstract void logout(TRTCVoiceRoomCallback.ActionCallback callback);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
callback	ActionCallback	ログアウトのコールバック。成功時にcodeは0になります。

setSelfProfile

個人情報の修正。



```
public abstract void setSelfProfile(String userName, String avatarURL, TRTCVoiceRoo
```

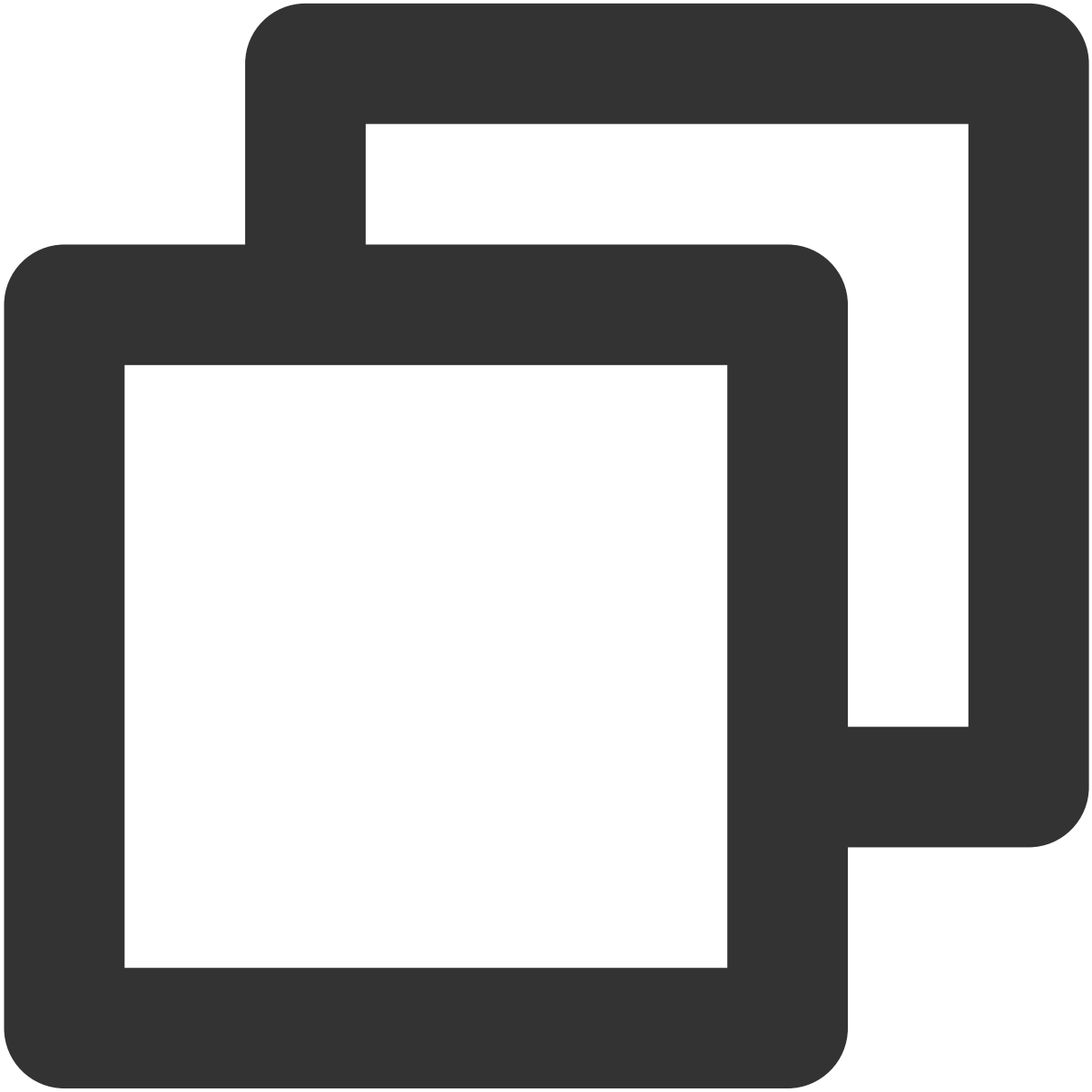
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userName	String	ニックネーム。
avatarURL	String	プロフィール画像のアドレス。
callback	ActionCallback	個人情報設定のコールバック。成功時にcodeは0になります。

ルーム関連インターフェース関数

createRoom

ルームの作成（管理者が呼び出し）。



```
public abstract void createRoom(int roomId, TRTCVoiceRoomDef.RoomParam roomParam, T
```

パラメータは下表に示すとおりです：

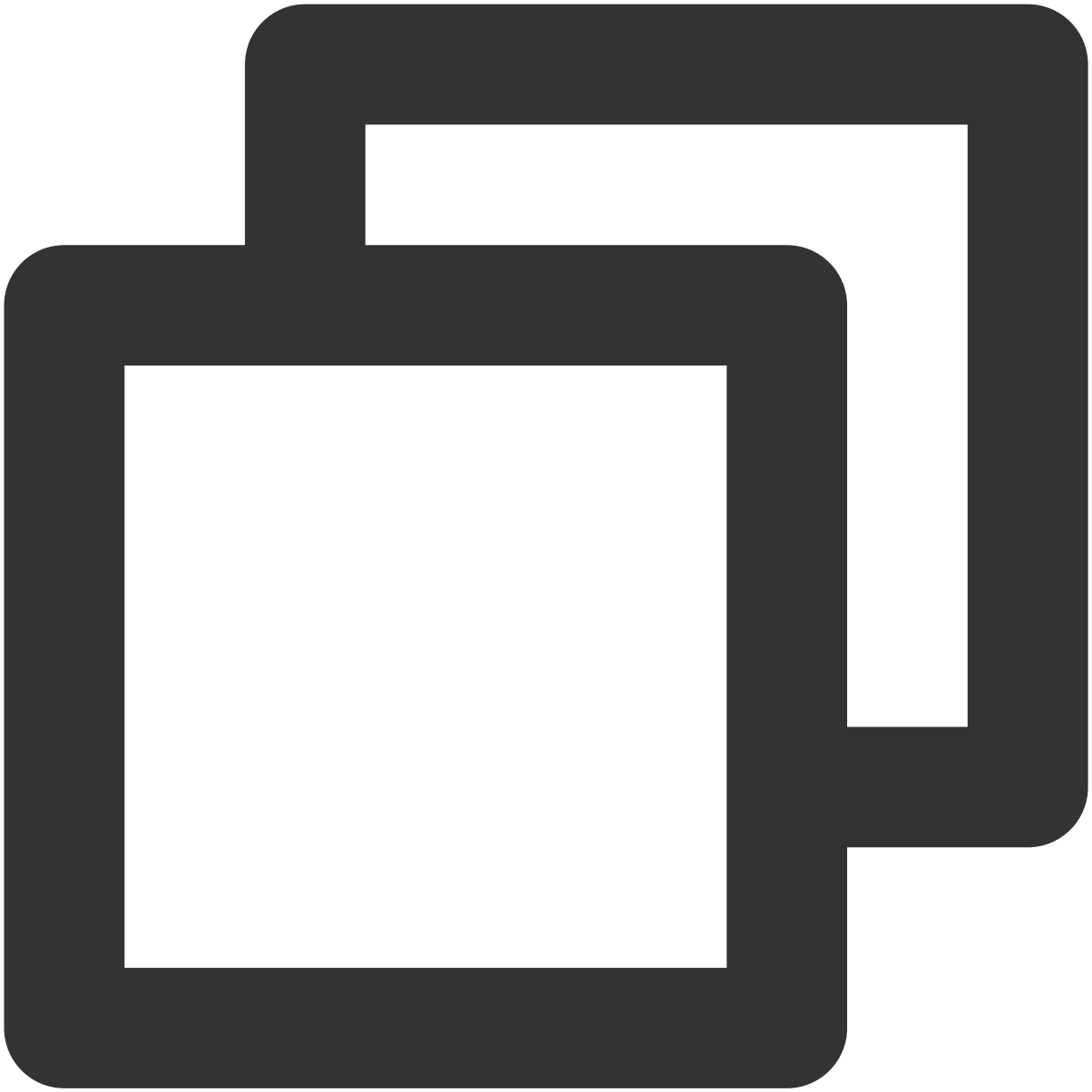
パラメータ	タイプ	意味

roomId	int	ルームIDです。ご自身でアサインし、一元管理する必要があります。複数のroomIDを、1つのボイスチャットルームリストにまとめることができます。Tencent Cloudでは現在、ボイスチャットルームリストの管理サービスを行っていませんので、ご自身でボイスチャットルームリストを管理してください。
roomParam	TRTCCreateRoomParam	ルーム情報です。ルーム名、マイク情報、カバー情報など、ルームを説明するために用いる情報に使用します。マイク管理が必要な場合は、ルームのマイク数を記入してください。
callback	ActionCallback	ルームの新規作成結果のコールバック。成功時にcodeは0になります。

管理者が配信を開始する際の通常の呼び出しプロセスは次のとおりです：
1. 管理者は、 `createRoom` を呼び出して新しいボイスチャットルームを作成します。この時にルームID、マイク・オンに管理者の確認の要否、マイク数などルームのプロパティ情報を渡します。
2. 管理者は、ルーム作成に成功した後、 `enterSeat` を呼び出して参加します。
3. 管理者は、コンポーネントの `onSeatListChange` マイクリスト変更イベント通知を受信します。この時、マイクリストの変更をUI上に更新することができます。
4. 管理者は、マイクリストのメンバーが参加した `onAnchorEnterSeat` というイベント通知も受信します。この時、マイク集音は自動的に開始されます。

destroyRoom

ルームの破棄（管理者が呼び出し）。管理者は、ルーム作成後、この関数を呼び出してルームを破棄します。



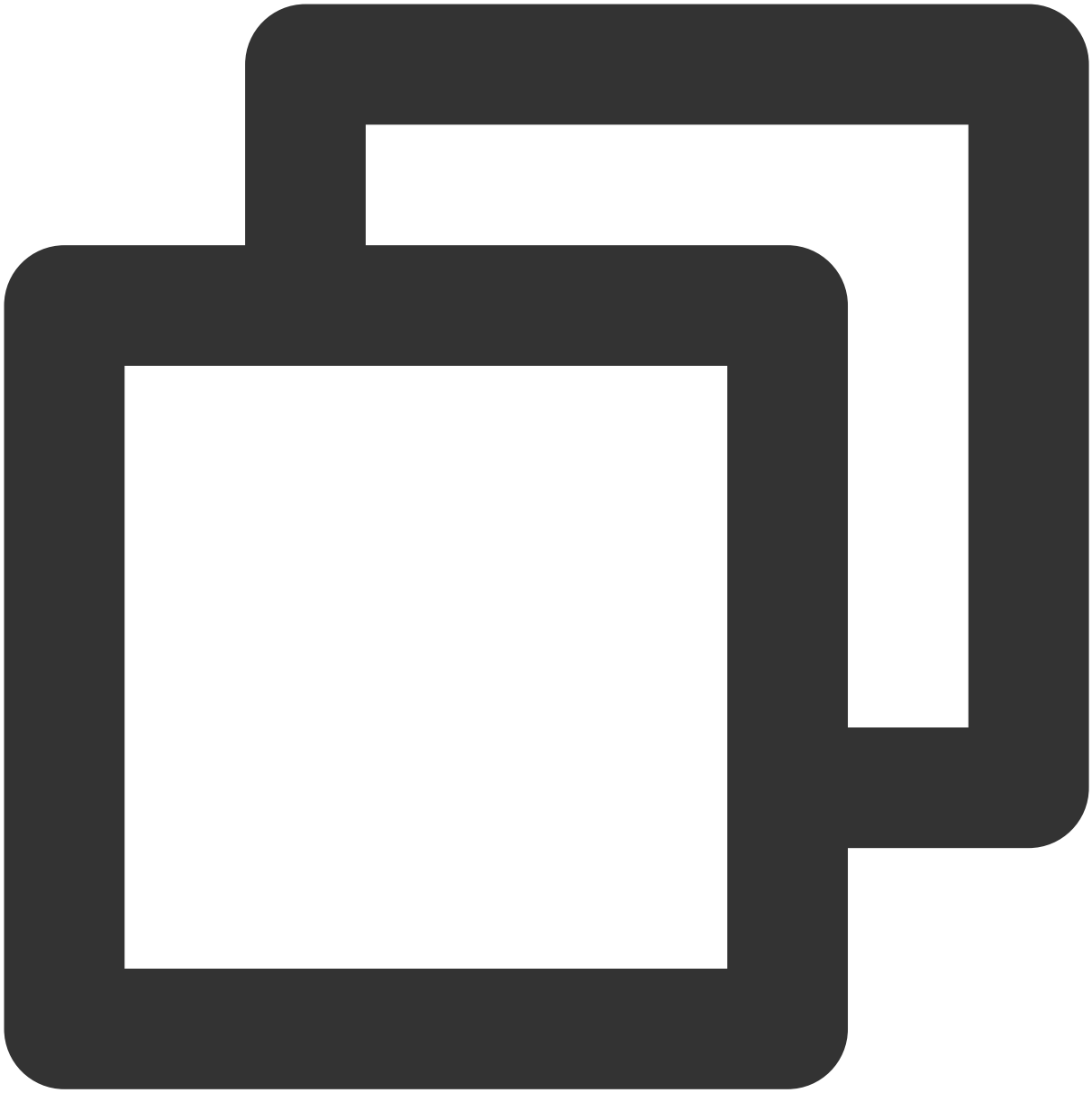
```
public abstract void destroyRoom(TRTCVoiceRoomCallback.ActionCallback callback);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
callback	ActionCallback	ルームの廃棄結果のコールバック。成功時にcodeは0になります。

enterRoom

入室（リスナーが呼び出し）。



```
public abstract void enterRoom(int roomId, TRTCVoiceRoomCallback.ActionCallback cal
```

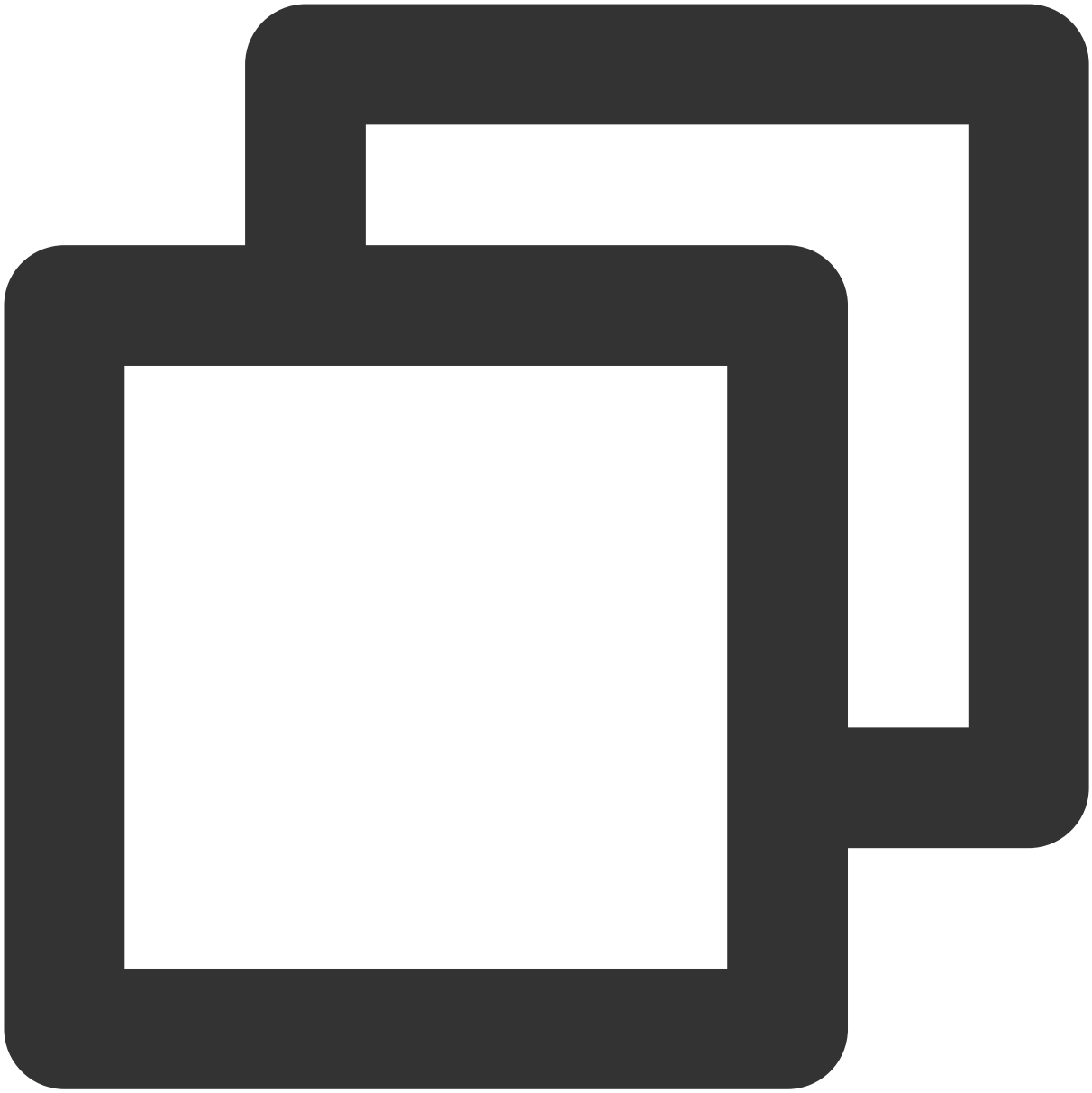
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
roomId	int	ルームID。
callback	ActionCallback	入室結果のコールバック。成功時にcodeは0になります。

リスナーが入室し聴取する際の通常の呼び出しプロセスは次のとおりです：
1. リスナーがサーバーから取得する最新のボイスチャットルームリストには、多くのボイスチャットルームのroomIdおよびルーム情報が含まれる場合があります。
2. リスナーは1つのボイスチャットルームを選択し、`enterRoom`を呼び出してルームナンバーを渡すと、そのルームに参加できます。
2. 入室後、コンポーネントの`onRoomInfoChange`ルーム属性変更イベント通知を受信します。この時、ルーム属性を記録し、それに応じた修正を行うことができます。例：UIに表示するルーム名、マイク・オンの際の管理者への同意リクエストの要否の記録など。
3. 入室後は、コンポーネントの`onSeatListChange`マイクリスト変更イベント通知を受信します。この時、マイクリストの変更をUI上に更新することができます。
4. 入室後に、マイクリストにキャスターが参加した`onAnchorEnterSeat`のイベント通知も受信します。

exitRoom

ルームから退出します。



```
public abstract void exitRoom(TRTCVoiceRoomCallback.ActionCallback callback);
```

パラメータは下表に示すとおりです：

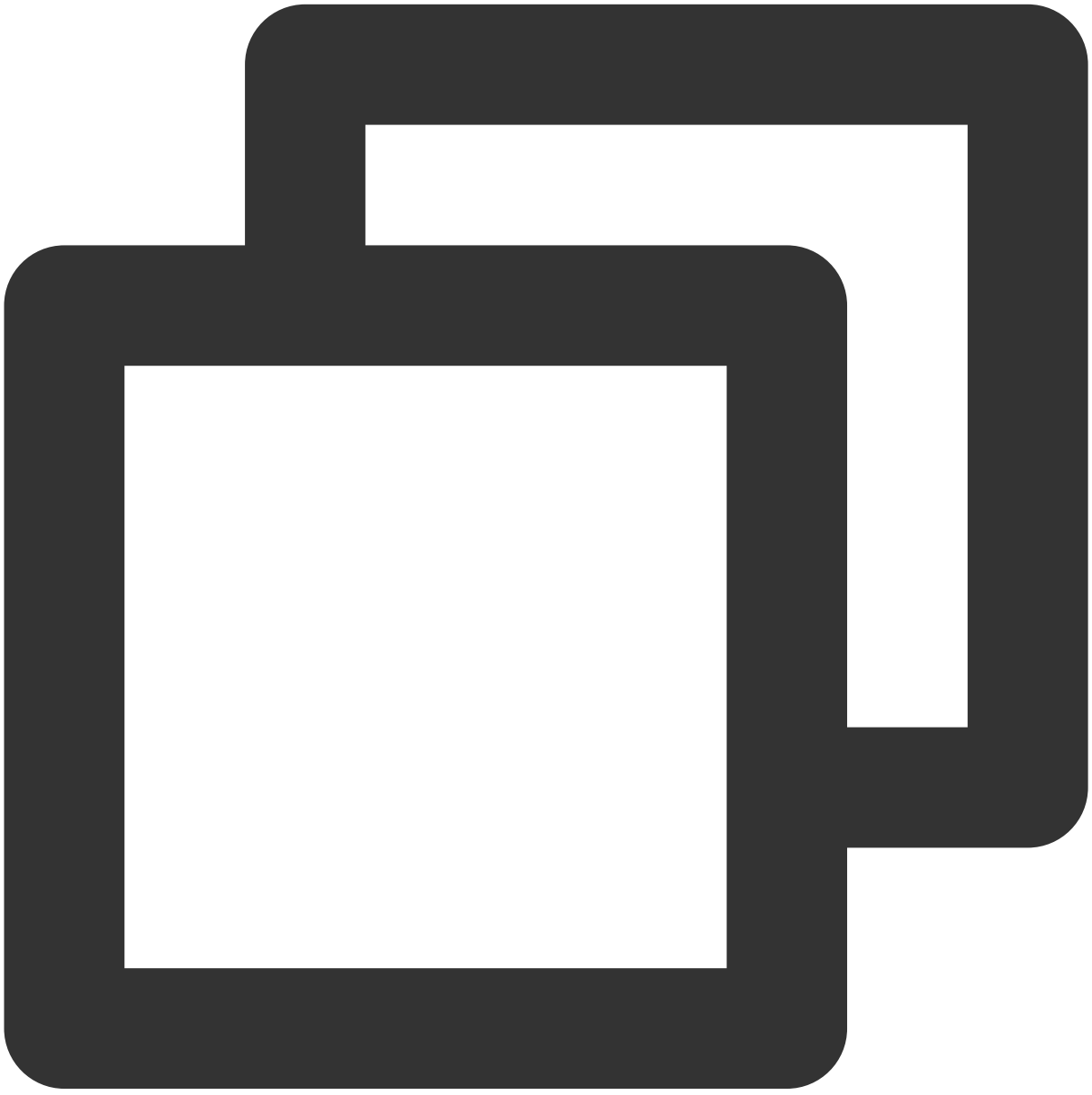
パラメータ	タイプ	意味
callback	ActionCallback	退室結果のコールバック。成功時にcodeは0になります。

getRoomInfoList

ルームリストの詳細情報を取得します。このうち、ルーム名、ルームカバーは、管理者が `createRoom()` 作成時にroomInfoによって設定したものになります。

説明：

ルームリストおよびルーム情報をご自身で管理する場合は、この関数は無視できます。



```
public abstract void getRoomInfoList(List<Integer> roomIdList, TRTCVoiceRoomCallbac
```

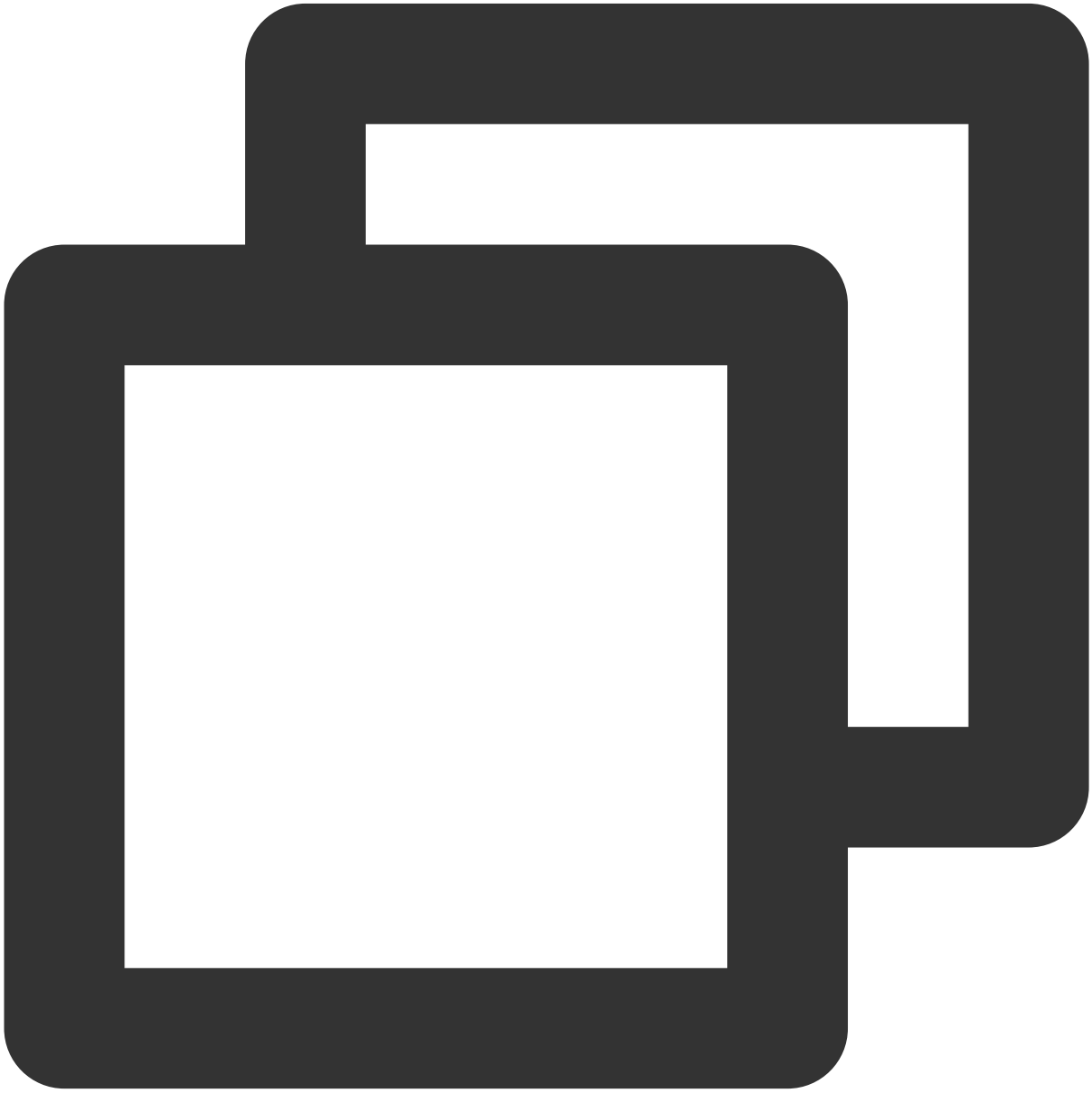
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味

roomIdList	List<Integer>	ルームナンバーリスト。
callback	RoomInfoCallback	ルーム詳細情報のコールバック。

getUserInfoList

指定されたuserIdのユーザー情報を取得します。



```
public abstract void getUserInfoList(List<String> userIdList, TRTCVoiceRoomCallback
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userIdList	List<String>	取得すべきユーザーIDリスト。nullの場合は、ルーム内全員の情報を取得します。
userlistcallback	UserListCallback	ユーザーの詳細情報のコールバック。

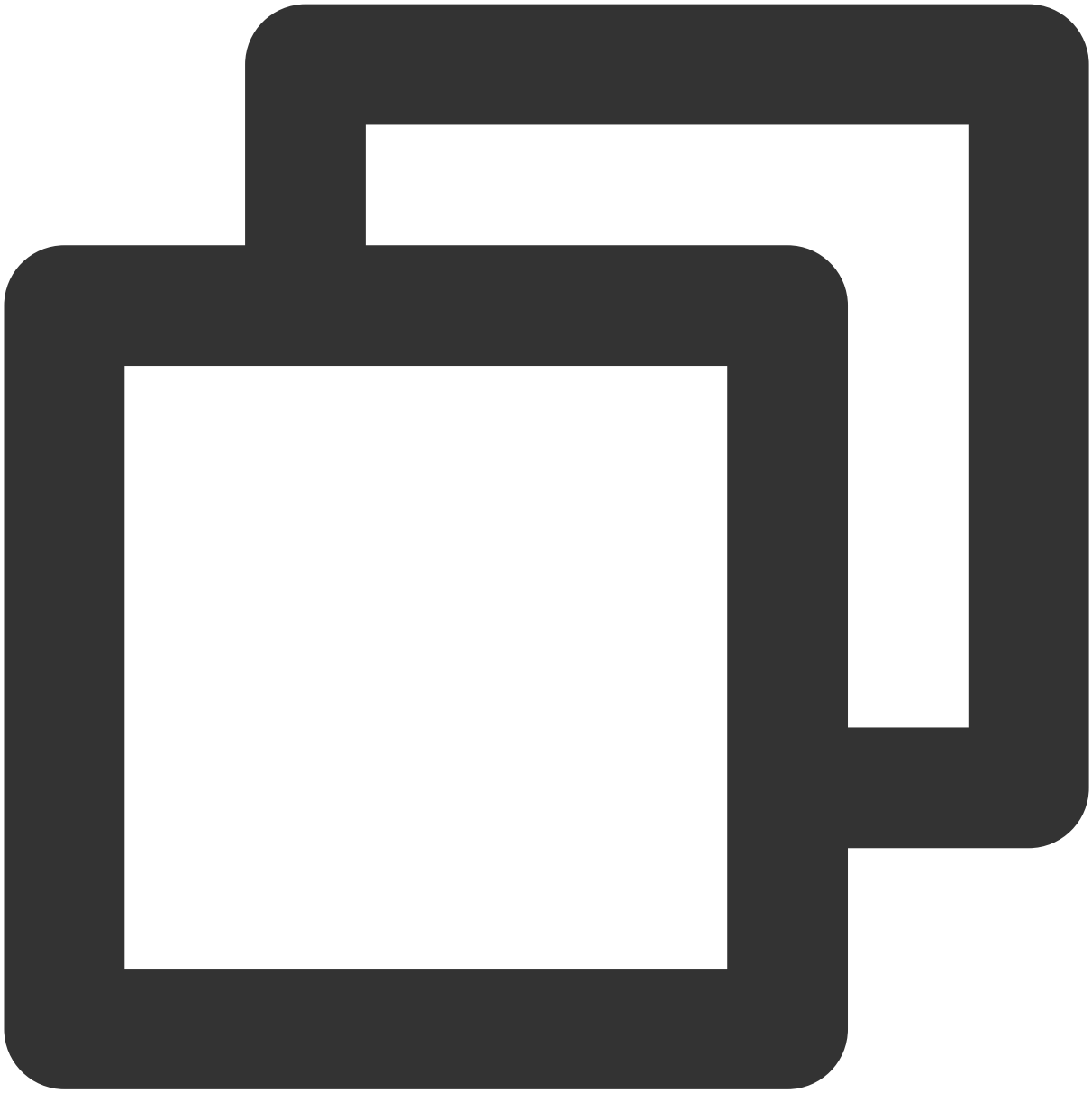
マイク管理インターフェース

enterSeat

ユーザーが発言者になります（リスナー側/管理者ともに呼び出し可）。

説明：

マイク・オンの成功後、ルーム内の全メンバーは、`onSeatListChange` および `onAnchorEnterSeat` というイベント通知を受信します。



```
public abstract void enterSeat(int seatIndex, TRTCVoiceRoomCallback.ActionCallback
```

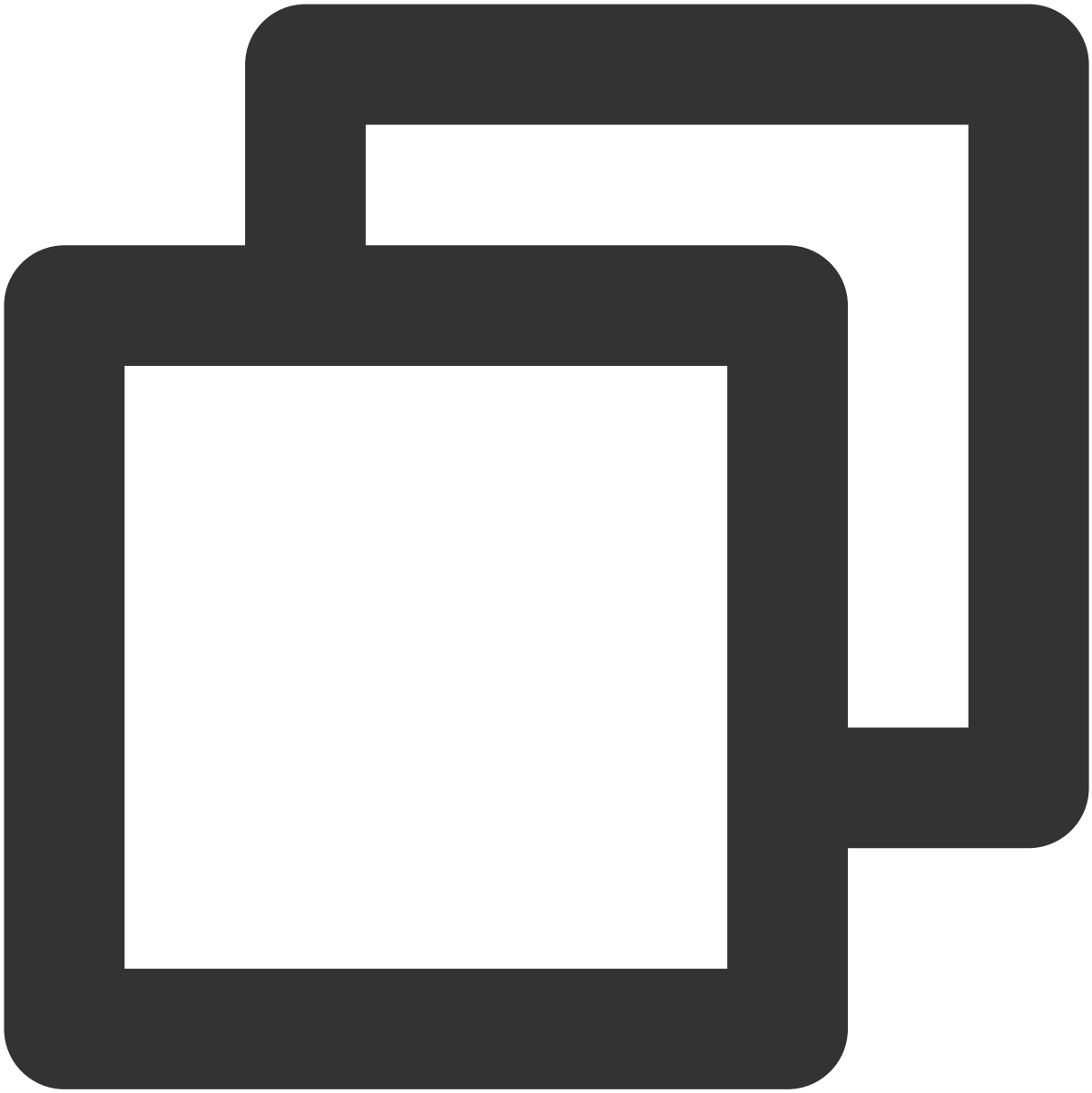
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
seatIndex	int	マイク・オンの必要があるマイク番号。
callback	ActionCallback	操作コールバック。

このインターフェースを呼び出すと、直ちにマイクリストが変更されます。リスナーによるマイク・オンの申請に管理者の同意が必要となるケースの場合は、まず `sendInvitation` を呼び出してから管理者に申請し、 `onInvitationAccept` を受信するとこの関数を呼び出せるようになります。

moveSeat

マイクの移動 (マイク・オンするキャスター側で呼び出せます)。
説明：
マイクの移動に成功したら、ルーム内のすべてのメンバーが `onSeatListChange` 、 `onAnchorLeaveSeat` 、 `onAnchorEnterSeat` のイベント通知を受け取ります。(キャスターが呼び出した後に変更できるのは、マイクのシート番号情報のみで、ユーザーのキャスターロールを切り替えることはできません。)



```
public abstract int moveSeat(int seatIndex, TRTCVoiceRoomCallback.ActionCallback ca
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
seatIndex	int	移動の必要があるマイク番号。
callback	ActionCallback	操作コールバック。

戻り値：

戻り値	タイプ	意味
code	int	マイク移動操作の結果（0は成功、その他は失敗、10001はインターフェース呼び出し頻度制限）。

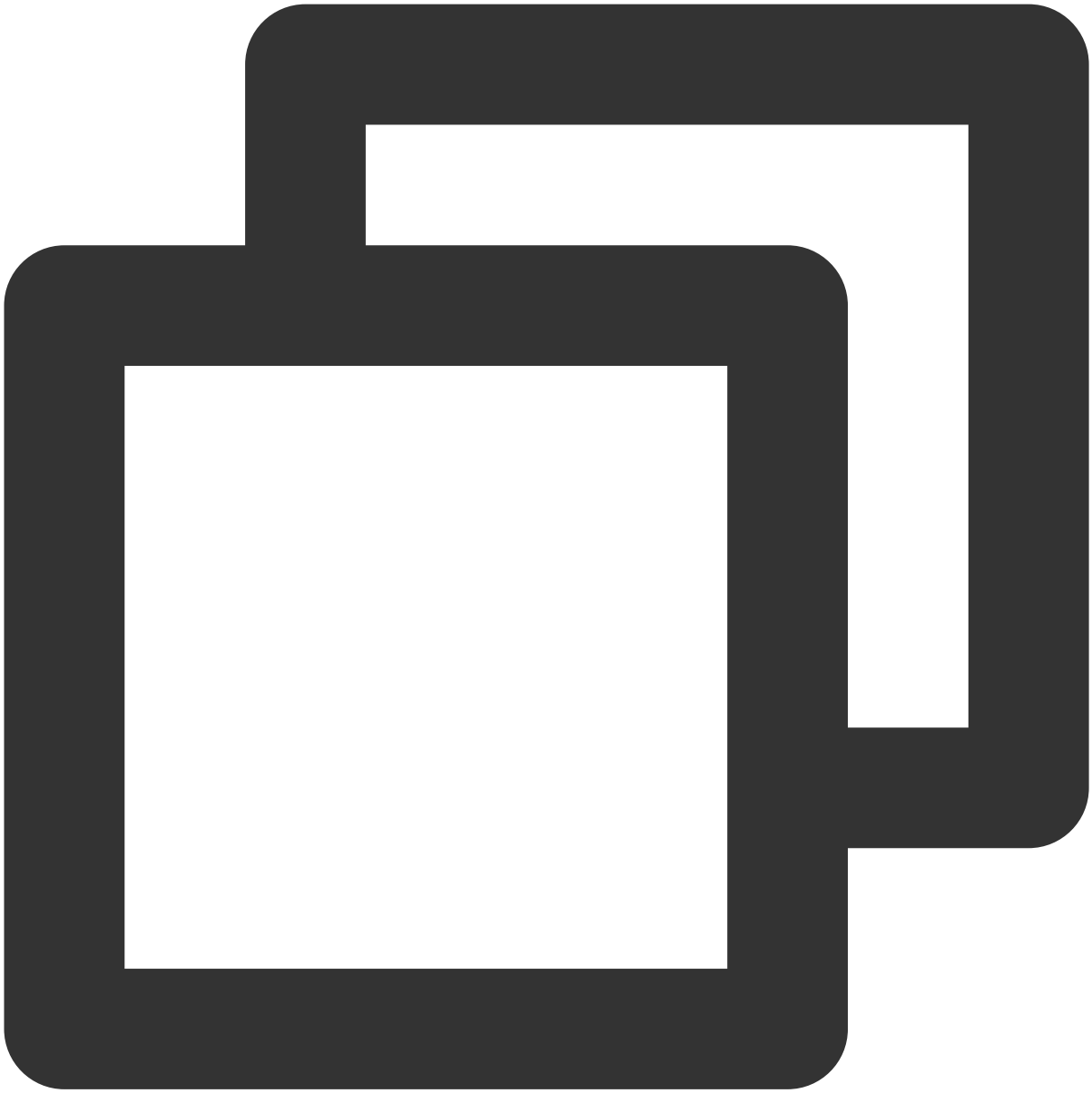
このインターフェースを呼び出すと、直ちにマイクリストが変更されます。リスナーによるマイク・オンの申請に管理者の同意が必要となるケースの場合は、まず `sendInvitation` を呼び出してから管理者に申請し、`onInvitationAccept` を受信するとこの関数を呼び出せるようになります。

leaveSeat

ユーザーが視聴者になります（キャスターが呼び出し）。

説明：

マイク・オフの成功後、ルーム内の全メンバーは、`onSeatListChange` および `onAnchorLeaveSeat` というイベント通知を受信します。



```
public abstract void leaveSeat(TRTCVoiceRoomCallback.ActionCallback callback);
```

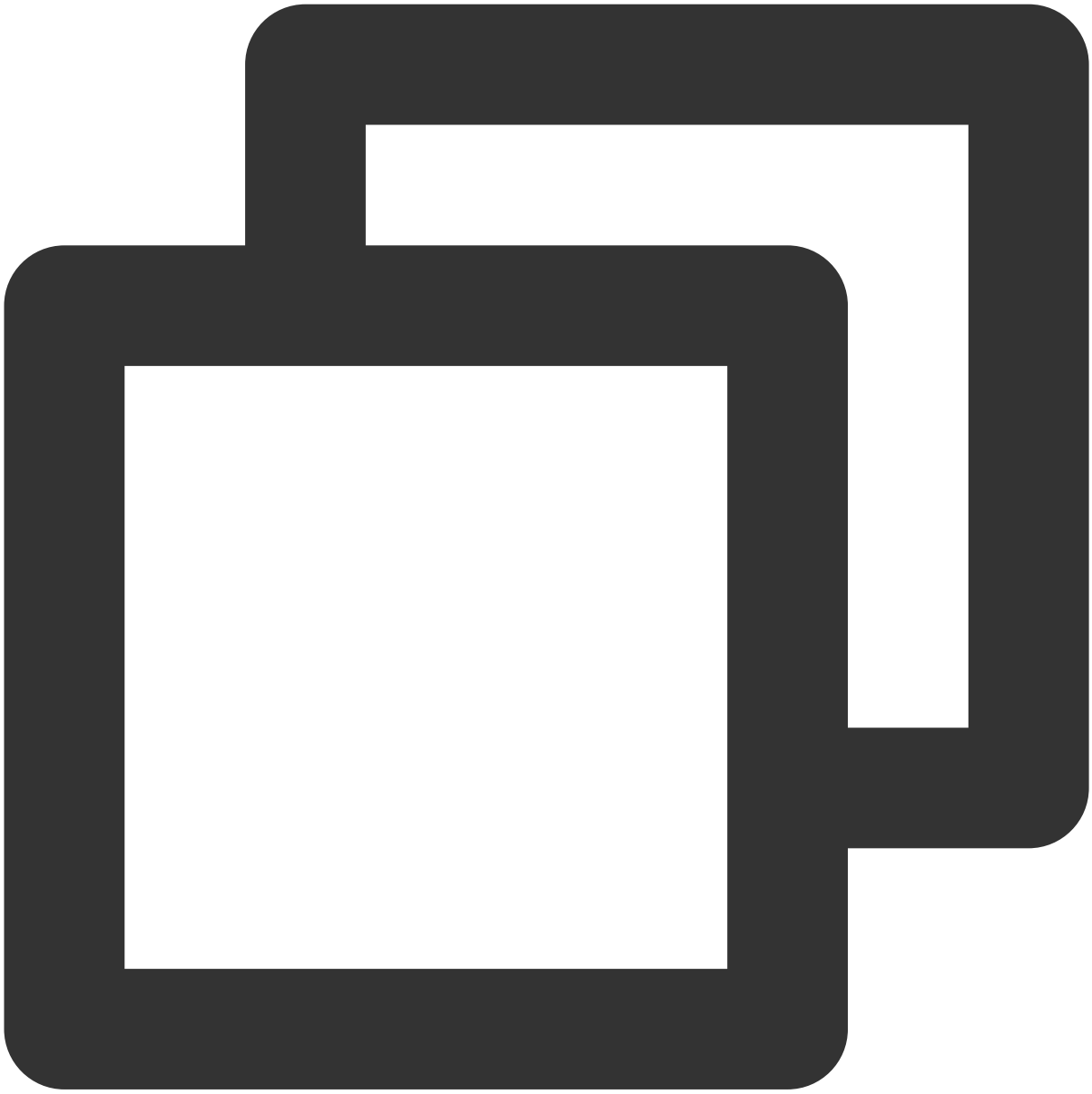
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
callback	ActionCallback	操作コールバック。

pickSeat

視聴者が発言できるように招待（管理者が呼び出し）。

説明：
管理者が視聴者を発言できるように招待すると、ルーム内の全メンバーは、 `onSeatListChange` と `onAnchorEnterSeat` というイベント通知を受信します。



```
public abstract void pickSeat(int seatIndex, String userId, TRTCVoiceRoomCallback.A
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
seatIndex	int	視聴者が発言できるように招待する必要があるマイク番号。

userId	String	ユーザーID。
callback	ActionCallback	操作コールバック。

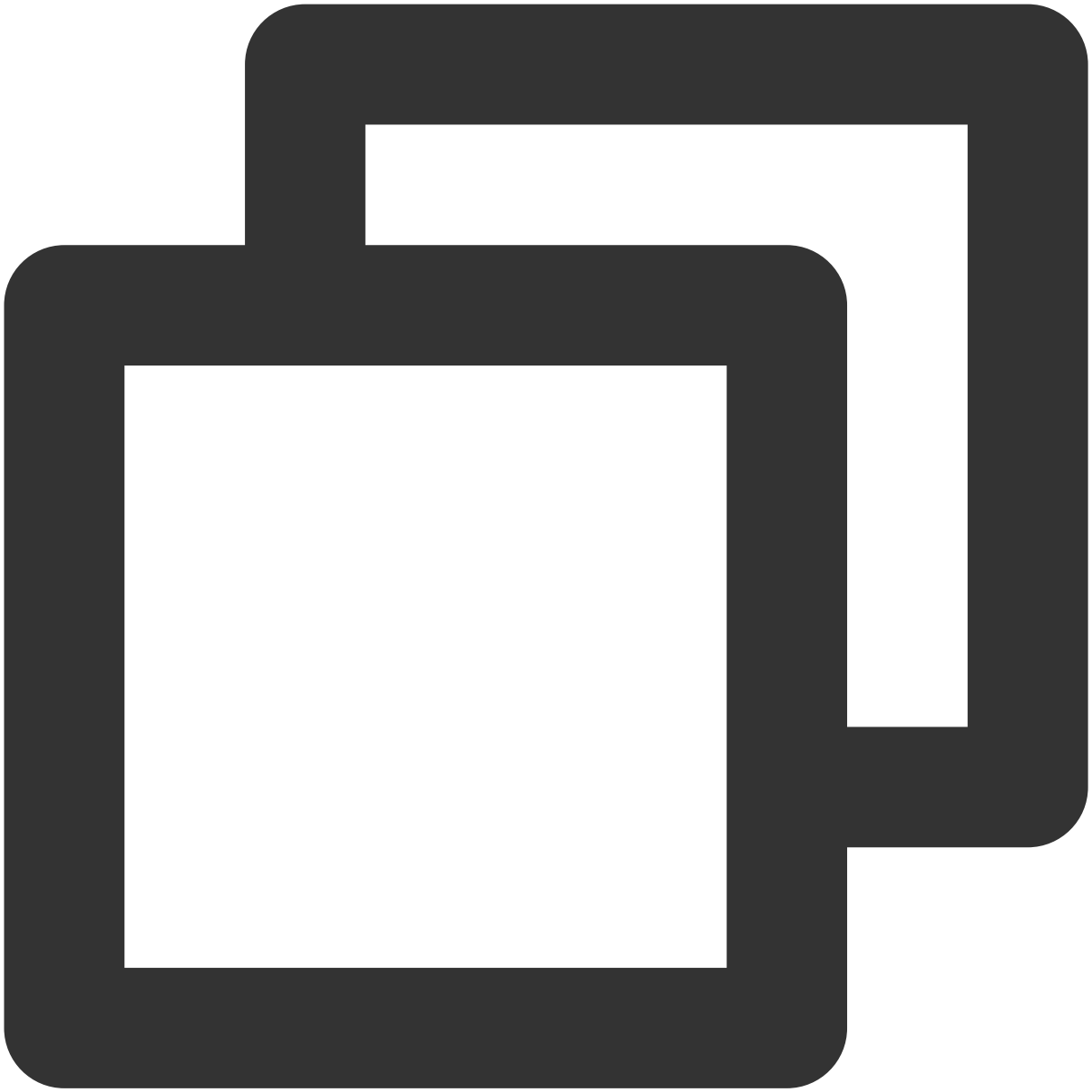
このインターフェースを呼び出すと、すぐにマイクリストが修正されます。管理者がリスナーの同意がなければマイク・オンできないケースの場合は、まず `sendInvitation` を呼び出してからリスナーに申請し、`onInvitationAccept` を受信すると、この関数をコールできるようになります。

kickSeat

キックアウトしてマイク・オフ（管理者が呼び出し）。

説明：

管理者がキックアウトしてマイク・オフにすると、ルーム内の全メンバーは、`onSeatListChange` および `onAnchorLeaveSeat` というイベント通知を受信します。



```
public abstract void kickSeat(int seatIndex, TRTCVoiceRoomCallback.ActionCallback c
```

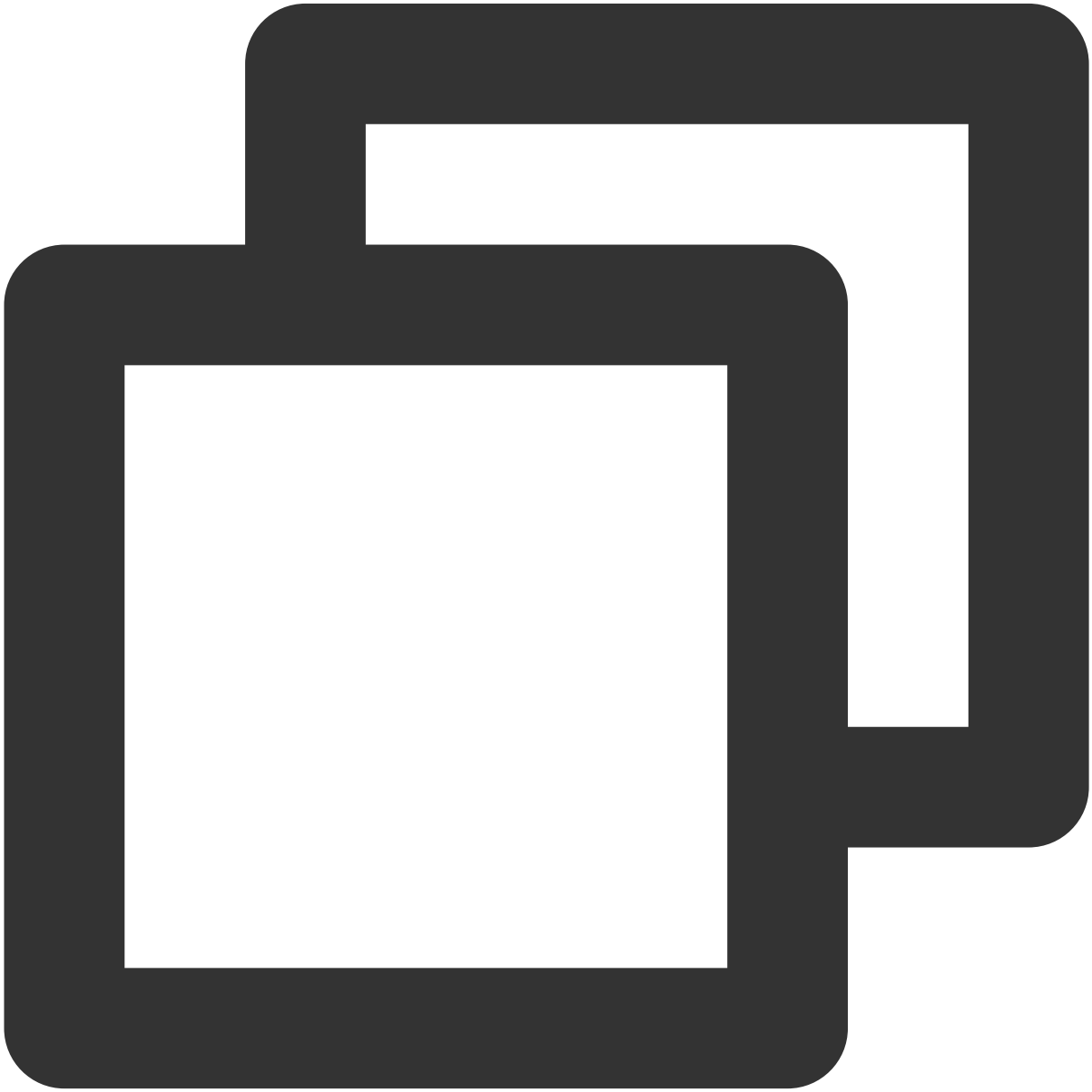
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
seatIndex	int	キックアウトしてマイク・オフの必要があるマイク番号。
callback	ActionCallback	操作コールバック。

このインターフェースを呼び出すと、すぐにマイクリストが修正されます。

muteSeat

任意のマイクのミュート/ミュート解除（管理者が呼び出し）。
説明：
任意のマイクのミュート/ミュート解除では、ルーム内の全メンバーが `onSeatListChange` および `onSeatMute` というイベント通知を受信します。



```
public abstract void muteSeat(int seatIndex, boolean isMute, TRTCVoiceRoomCallback.
```

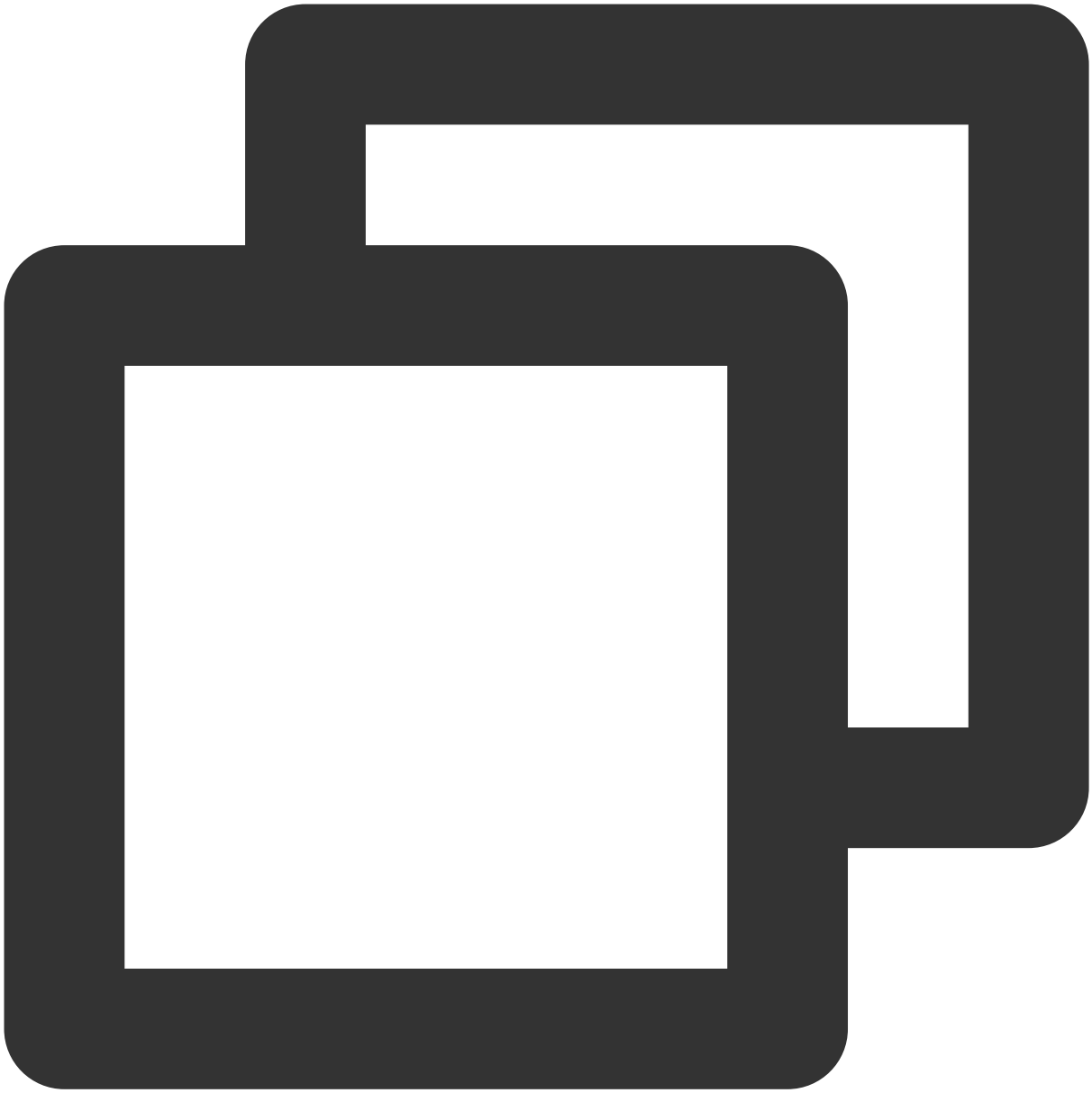
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
seatIndex	int	操作の必要があるマイク番号。
isMute	boolean	true：該当するマイクをミュートします；false：該当するマイクをミュート解除します。
callback	ActionCallback	操作コールバック。

このインターフェースを呼び出すと、直ちにマイクリストが変更されます。seatIndexの座席に該当するキャスターは、muteAudioを自動的に呼び出してミュート/ミュート解除します。

closeSeat

任意のマイクのクローズ/解除（管理者が呼び出し）。
説明：
管理者は、該当するマイクをクローズ/解除し、ルーム内の全メンバーは `onSeatListChange` および `onSeatClose` というイベント通知を受信します。



```
public abstract void closeSeat(int seatIndex, boolean isClose, TRTCVoiceRoomCallbac
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
seatIndex	int	操作の必要があるマイク番号。
isClose	boolean	true：該当するマイクをクローズします； false：該当するマイクをクローズ解除します。

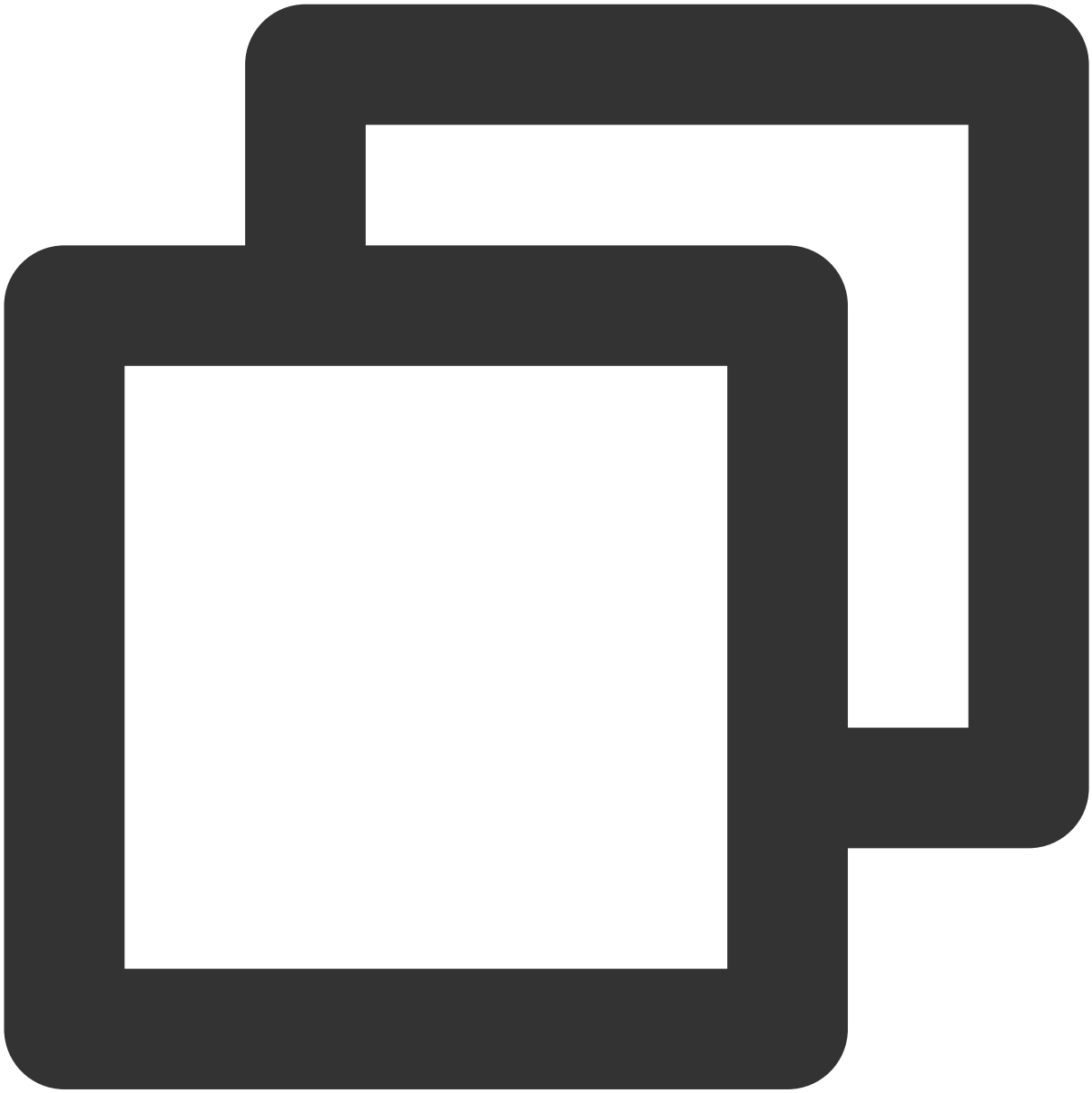
callback	ActionCallback	操作コールバック。

このインターフェースを呼び出すと、すぐにマイクリストが修正されます。該当するseatIndexの座席上のキャスターはクローズされ、自動的にマイク・オフになります。

ローカル音声操作のインターフェース

startMicrophone

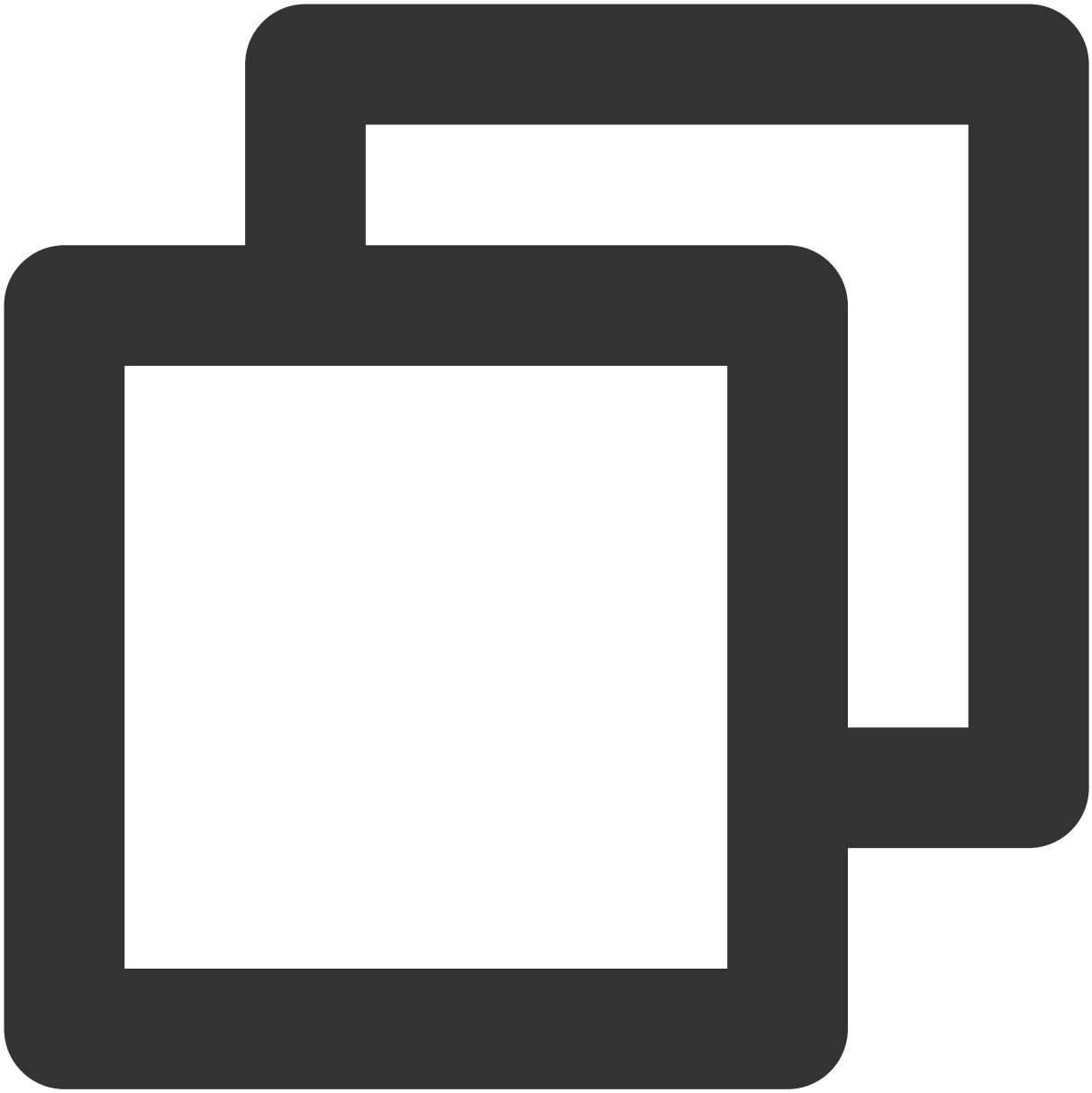
マイクの集音開始。



```
public abstract void startMicrophone();
```

stopMicrophone

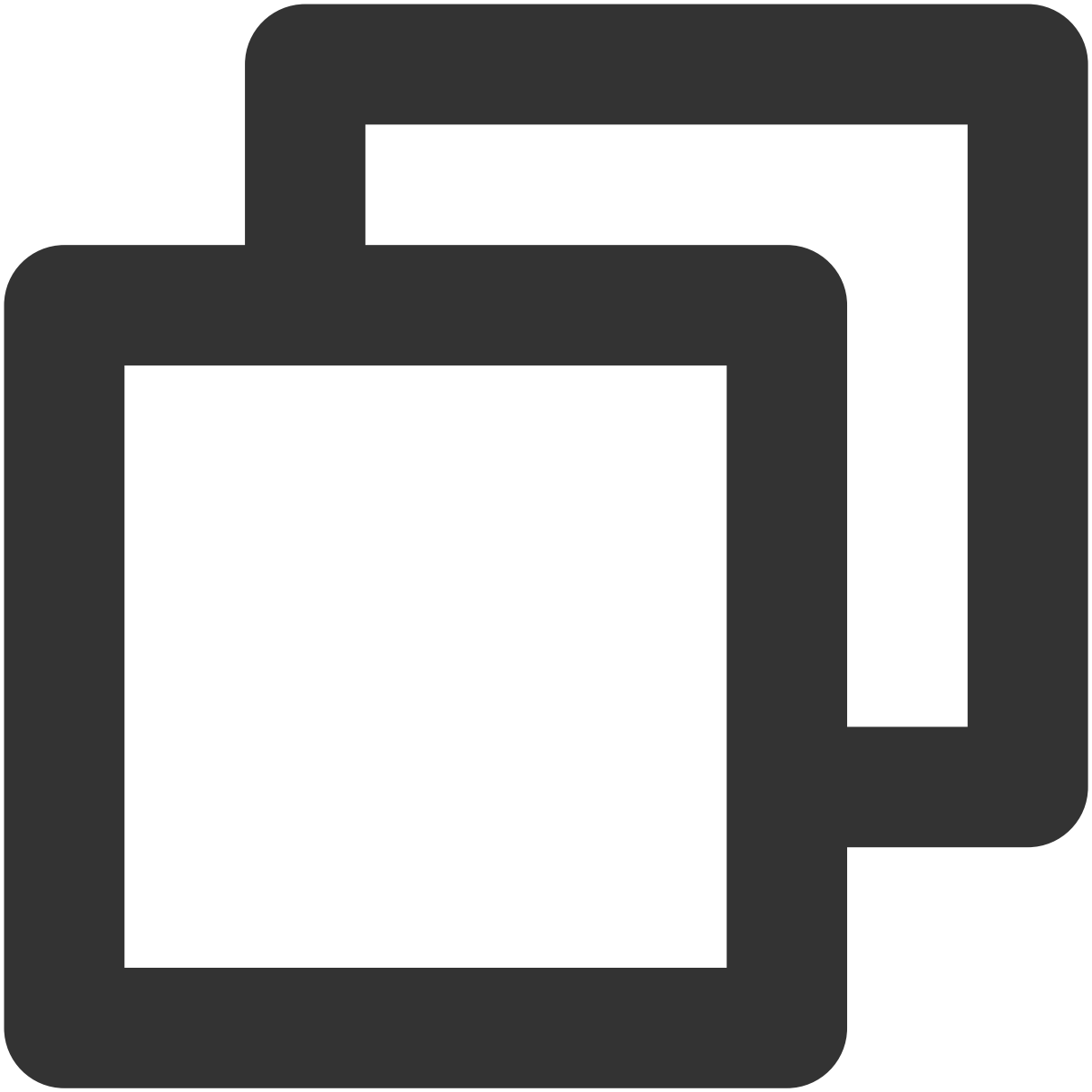
マイクの集音停止。



```
public abstract void stopMicrophone();
```

setAudioQuality

音質の設定。



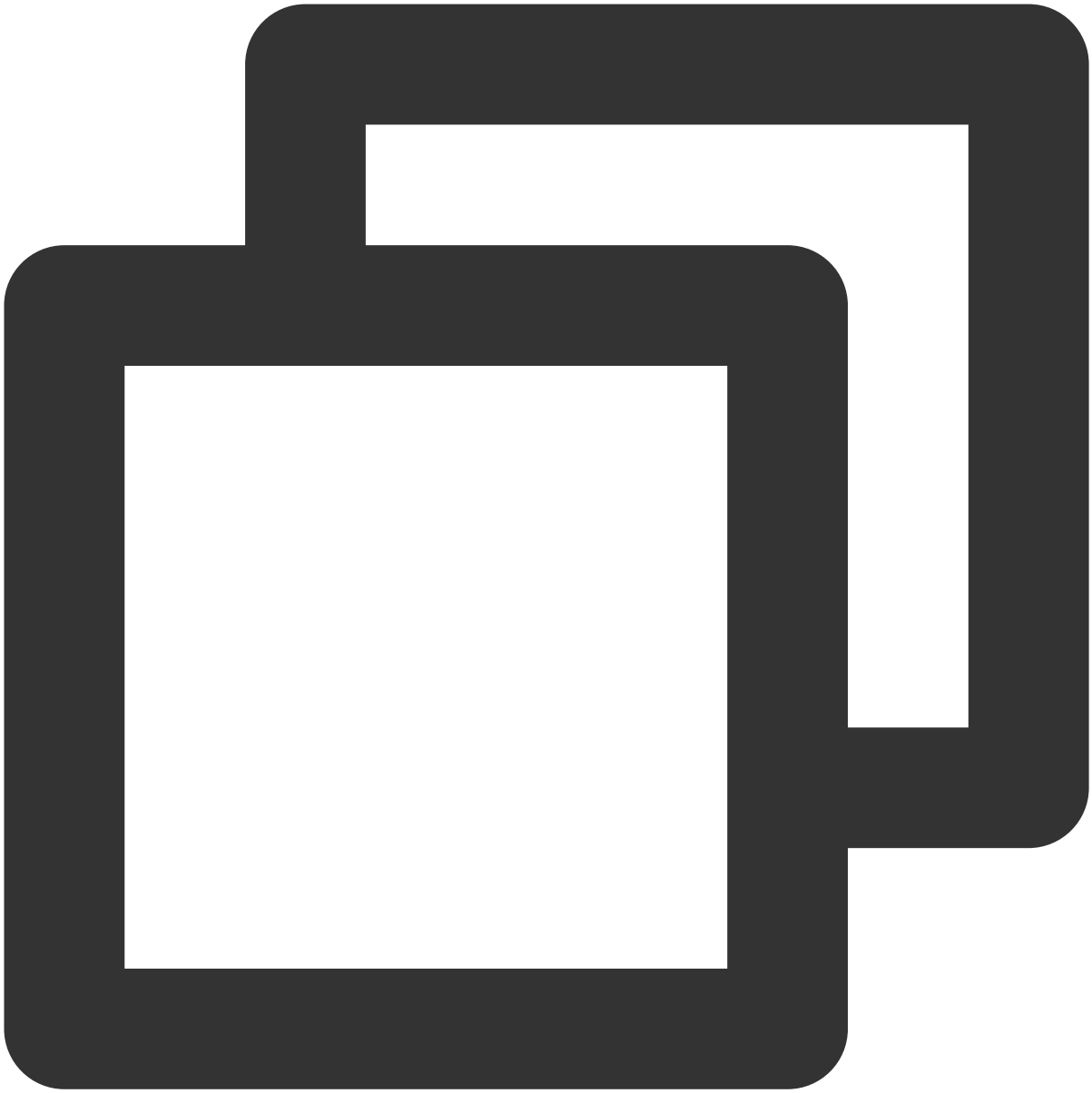
```
public abstract void setAudioQuality(int quality);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
quality	int	音声品質。詳細は TRTC SDKをご参照ください。

muteLocalAudio

ローカルの音声のミュート/ミュート取り消し。



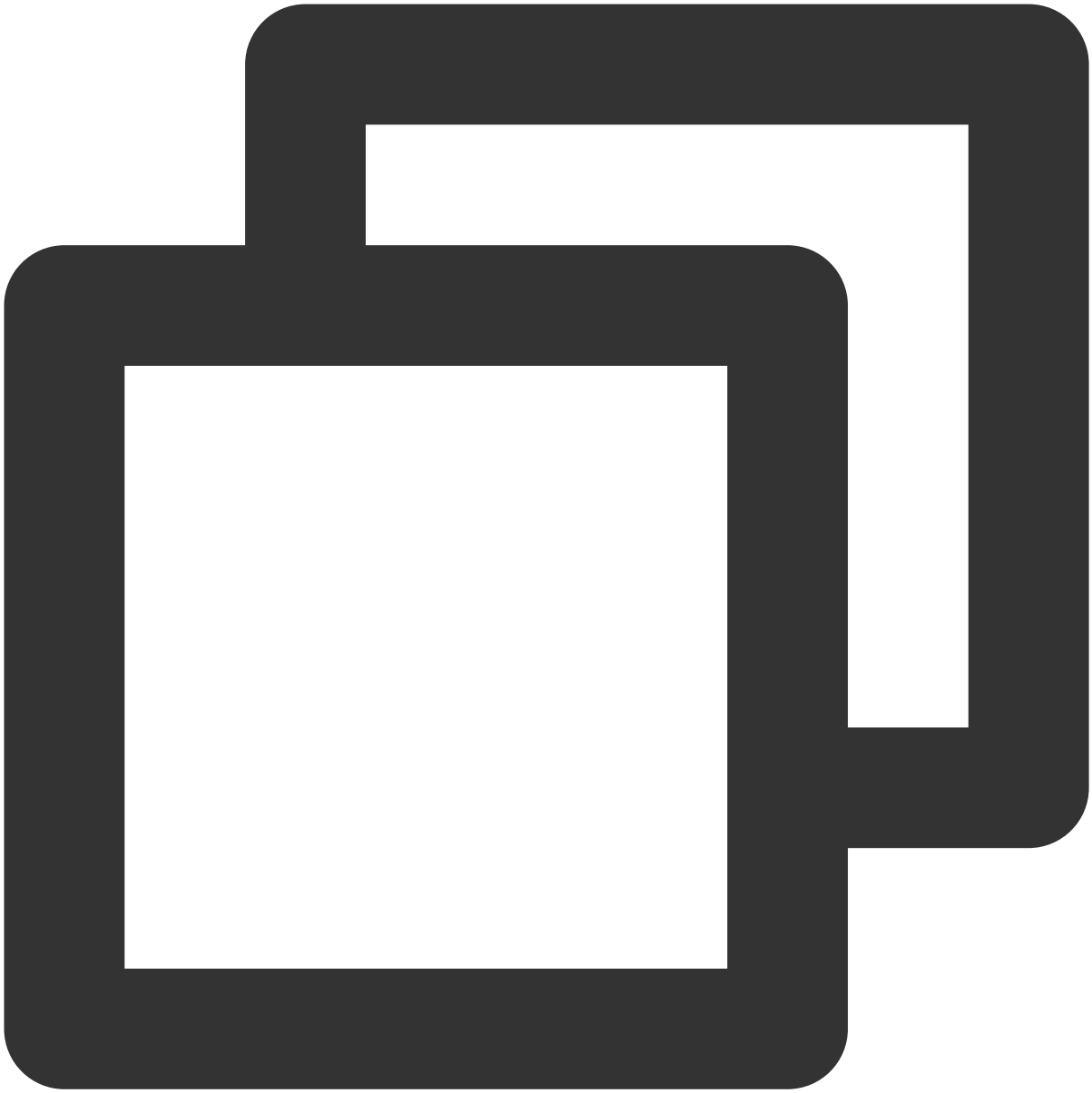
```
public abstract void muteLocalAudio(boolean mute);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
mute	boolean	ミュート/ミュート取り消し。詳細は TRTC SDKをご参照ください。

setSpeaker

スピーカーの起動設定。



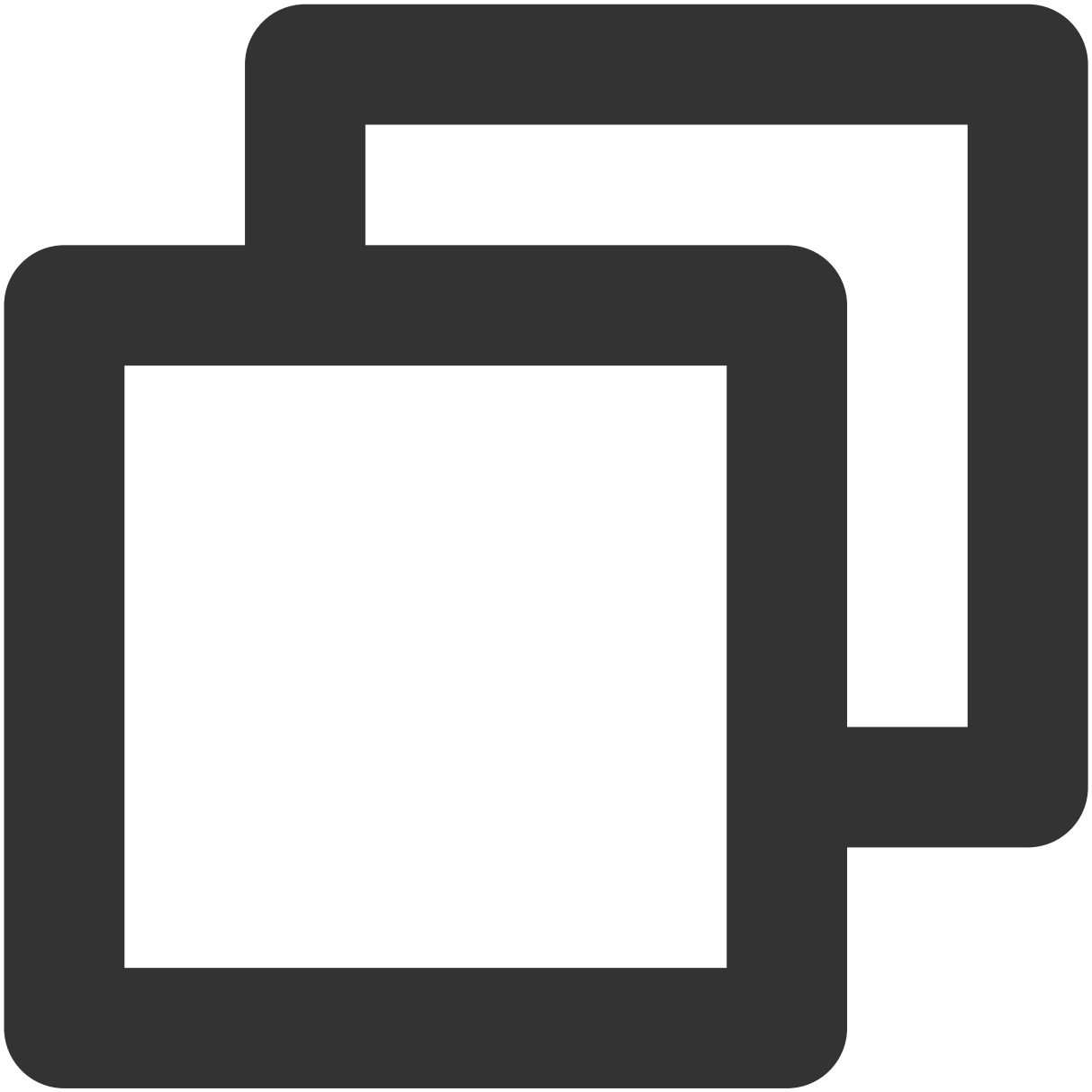
```
public abstract void setSpeaker(boolean useSpeaker);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
useSpeaker	boolean	true：スピーカー、false：ヘッドホン。

setAudioCaptureVolume

マイクの集音音量設定。



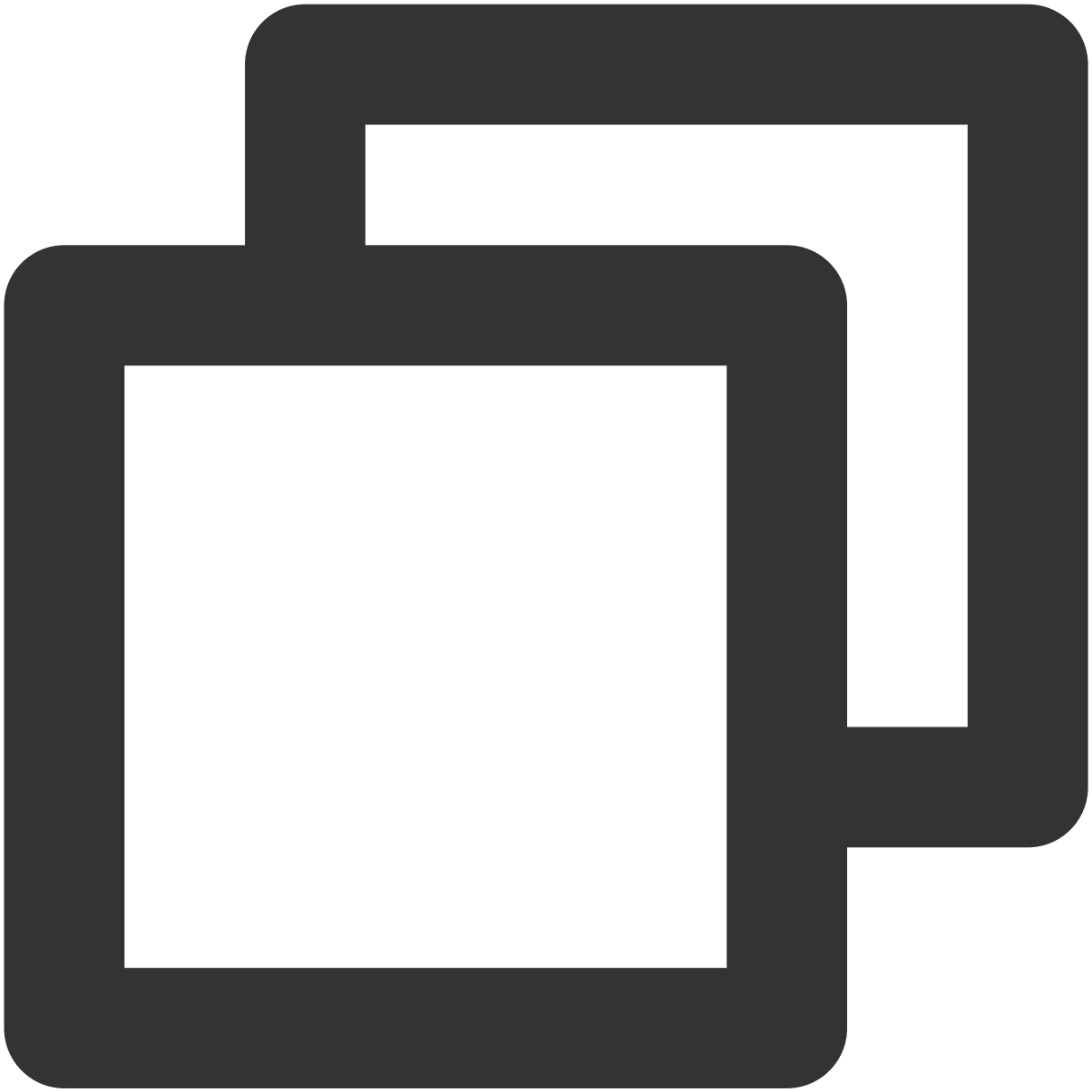
```
public abstract void setAudioCaptureVolume(int volume);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
volume	int	集音音量、0 - 100、 デフォルト100。

setAudioPlayoutVolume

再生音量の設定。



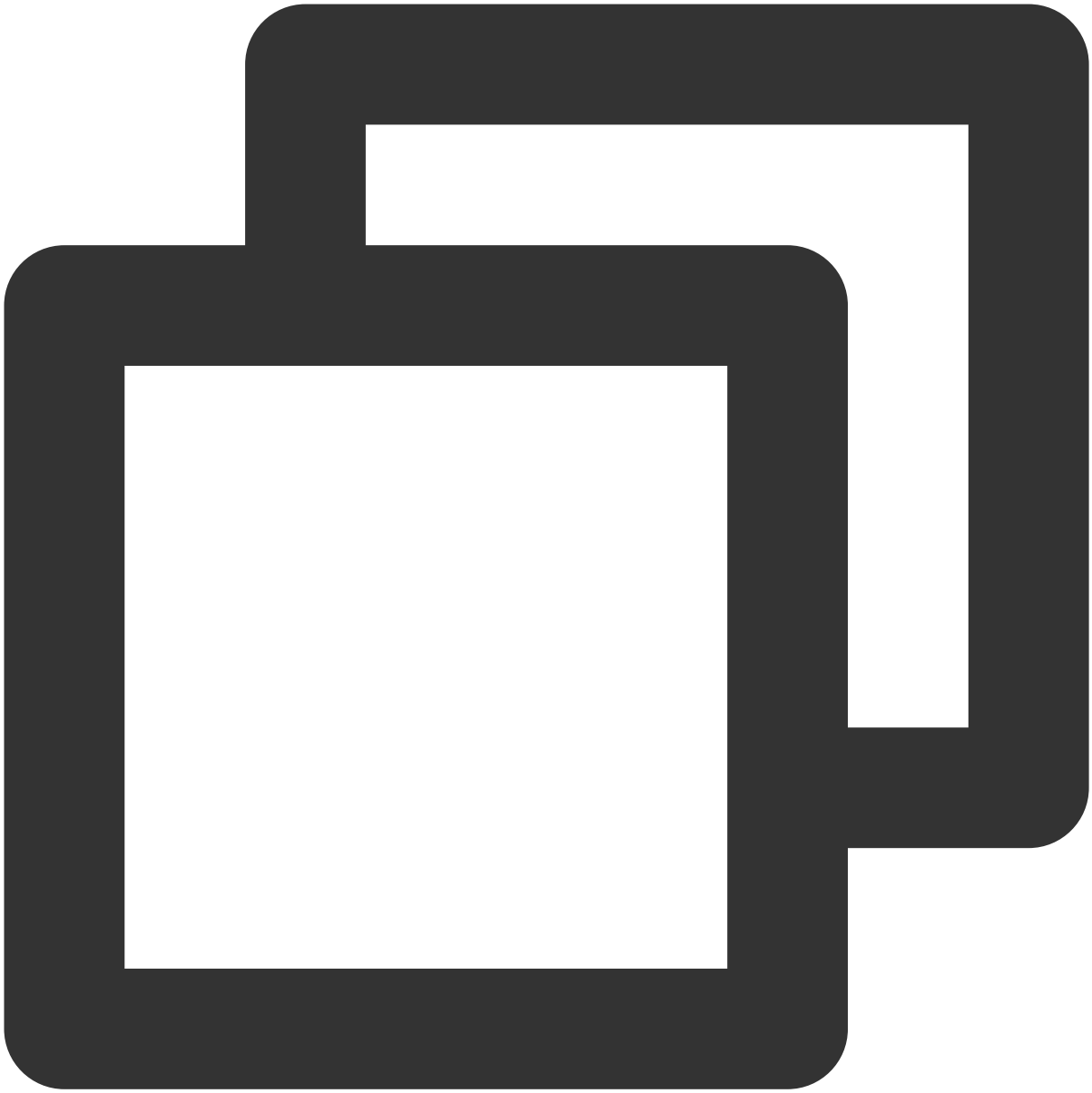
```
public abstract void setAudioPlayoutVolume(int volume);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
volume	int	再生音量、0 - 100、 デフォルト100。

muteRemoteAudio

指定メンバーのミュート/ミュート解除。



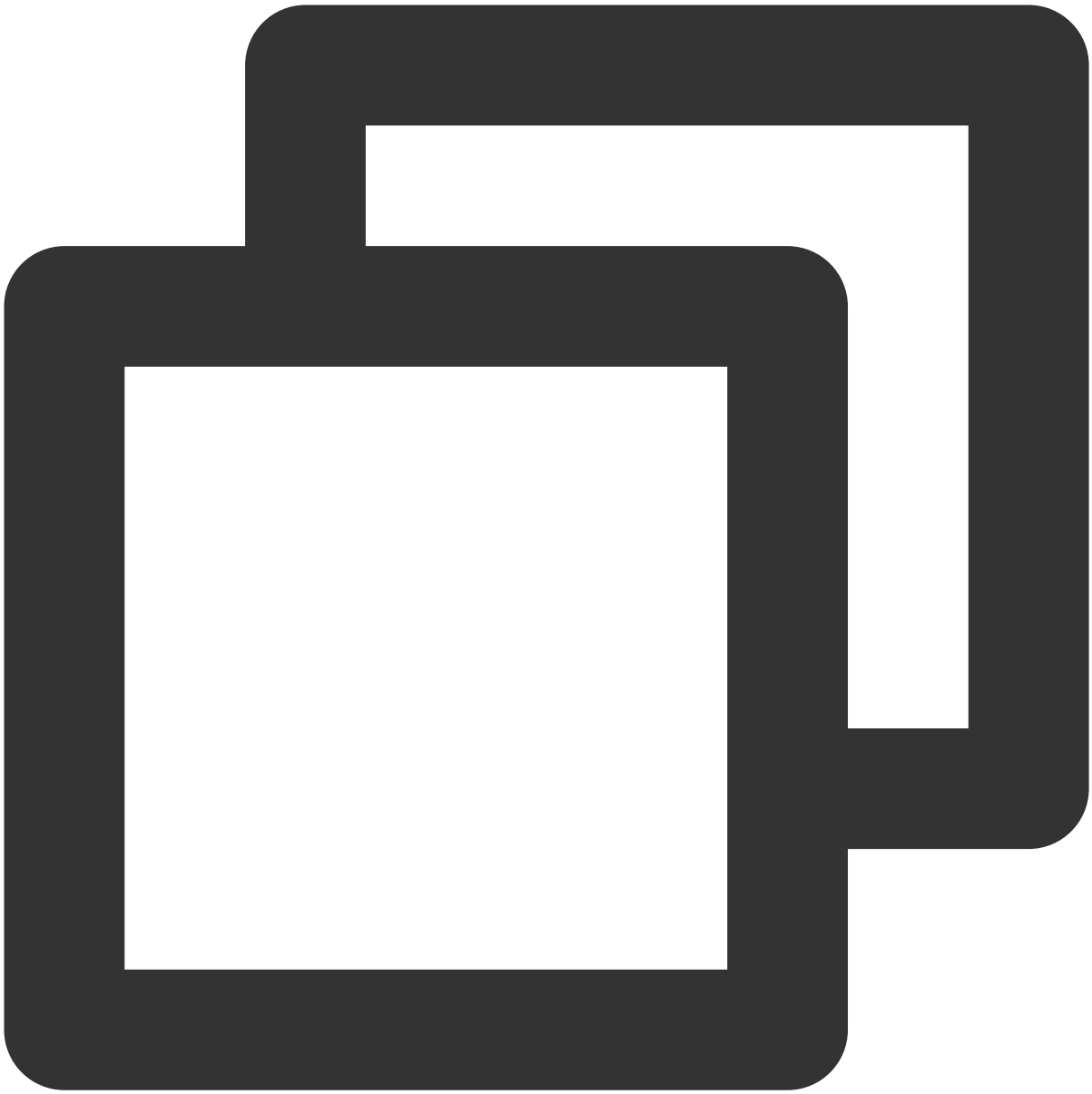
```
public abstract void muteRemoteAudio(String userId, boolean mute);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userId	String	指定ユーザーID。
mute	boolean	true：ミュート起動；false：ミュート停止。

muteAllRemoteAudio

全メンバーのミュート/ミュート解除。



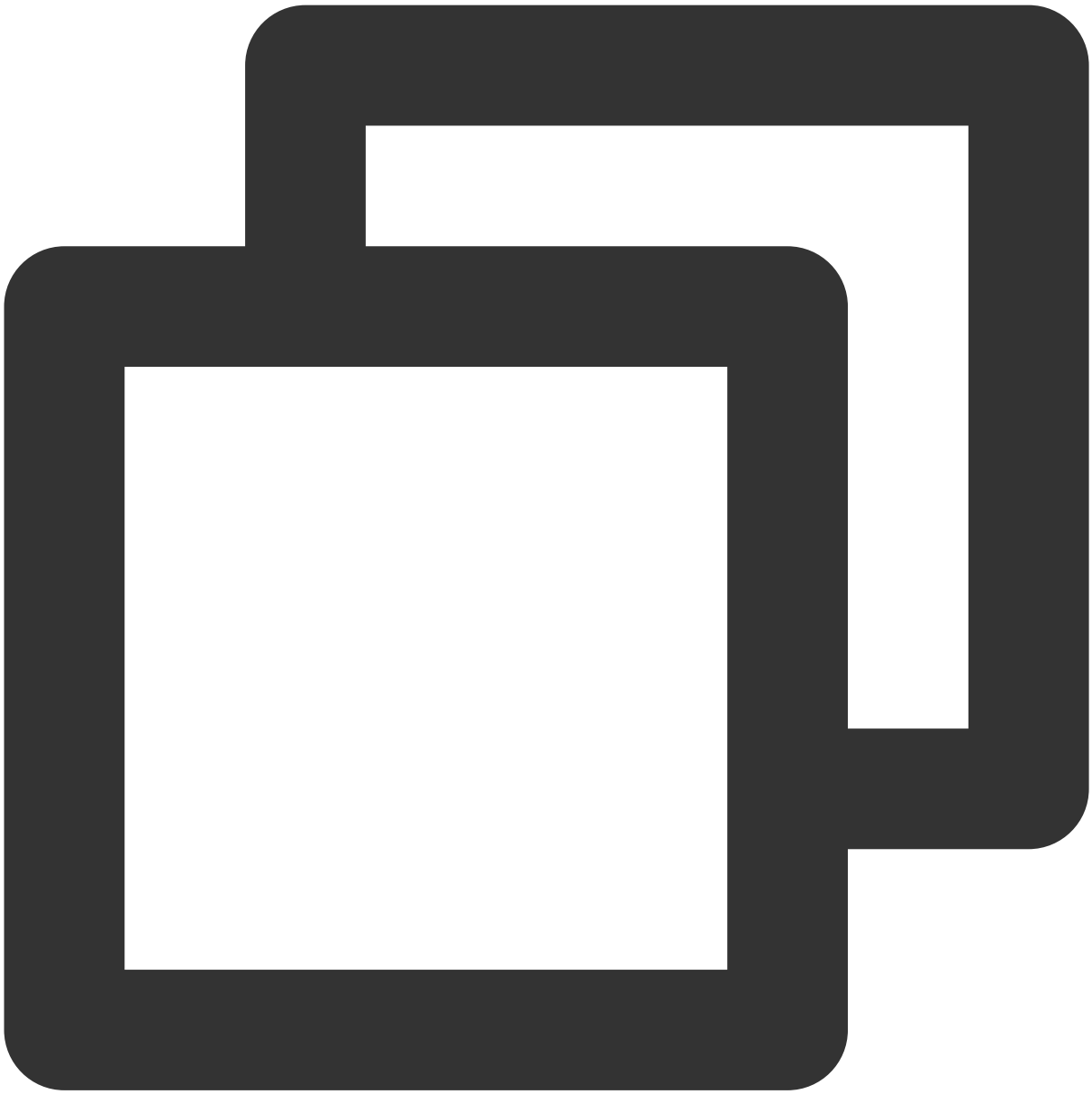
```
public abstract void muteAllRemoteAudio(boolean mute);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
mute	boolean	true：ミュート起動；false：ミュート停止。

setVoiceEarMonitorEnable

インイヤーモニタリングのオン/オフ。



```
public abstract void setVoiceEarMonitorEnable(boolean enable);
```

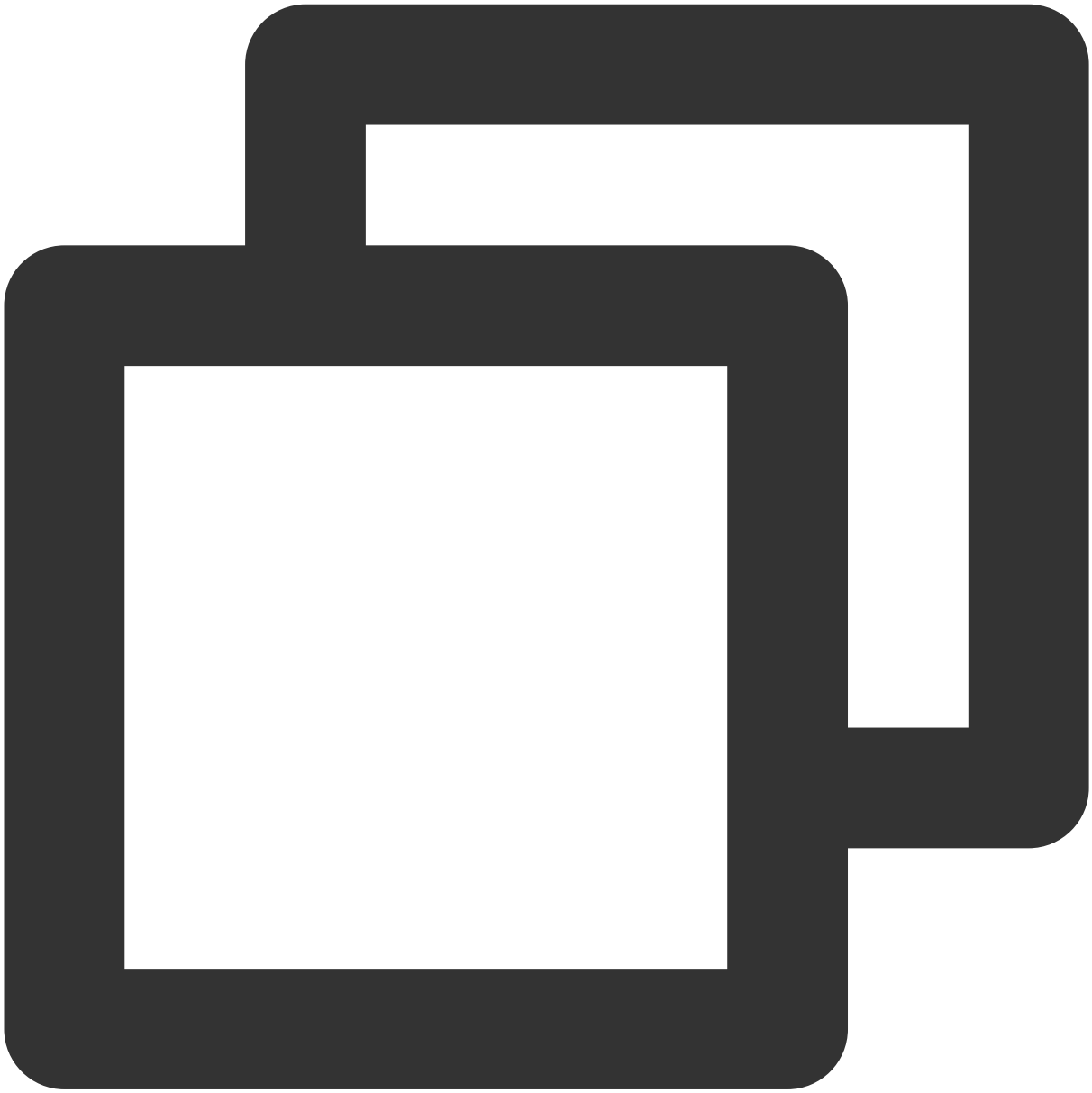
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
enable	boolean	true：インイヤーモニタリングをオン。false：インイヤーモニタリングをオフ。

BGMサウンドエフェクト関連インターフェース関数

getAudioEffectManager

BGMサウンドエフェクト管理オブジェクト TXAudioEffectManagerを取得します。

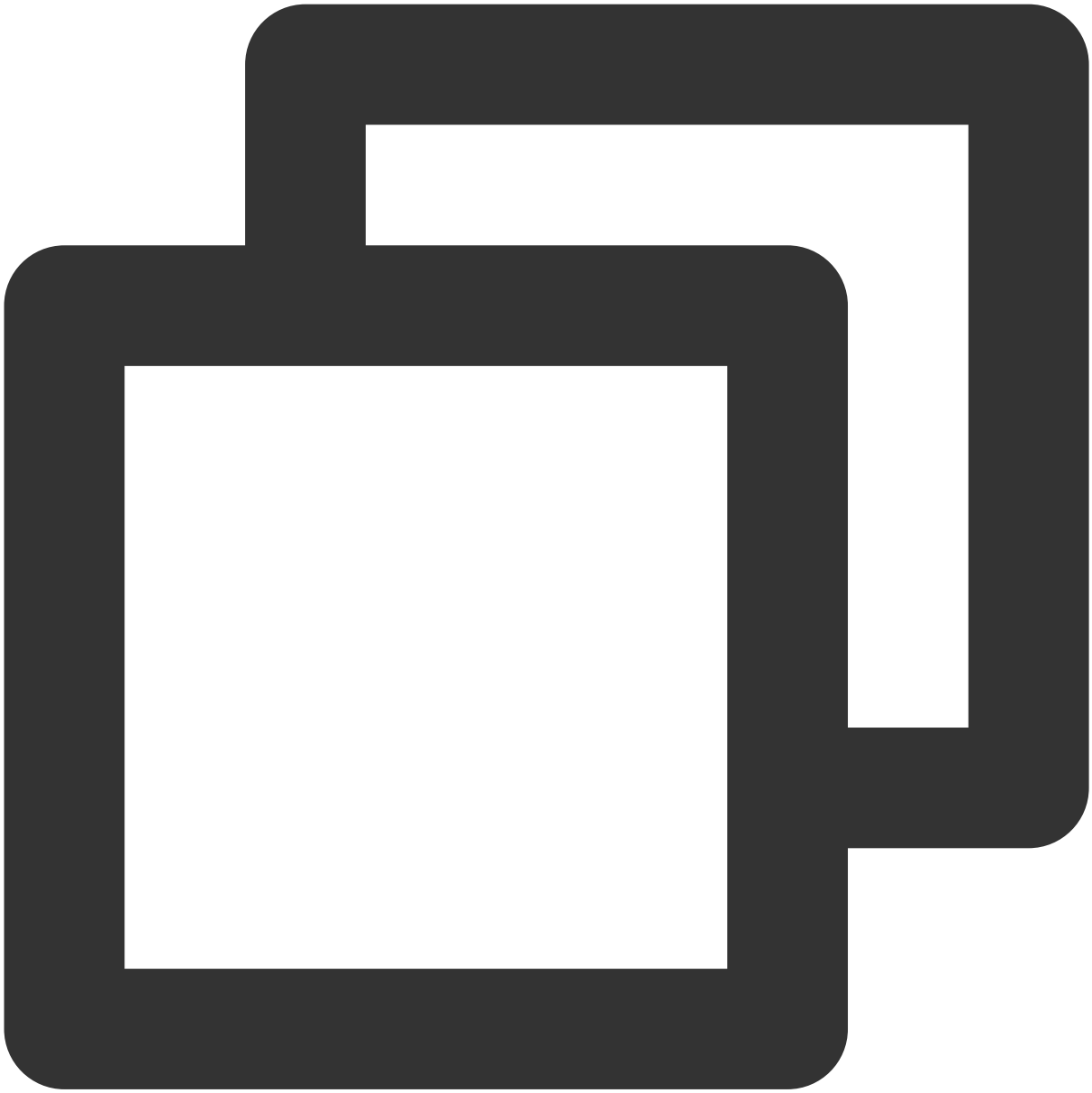


```
public abstract TXAudioEffectManager getAudioEffectManager();
```

メッセージ送信関連インターフェース関数

sendRoomTextMsg

ルーム内でテキストメッセージをブロードキャストします。通常、弾幕によるチャットに使用します。



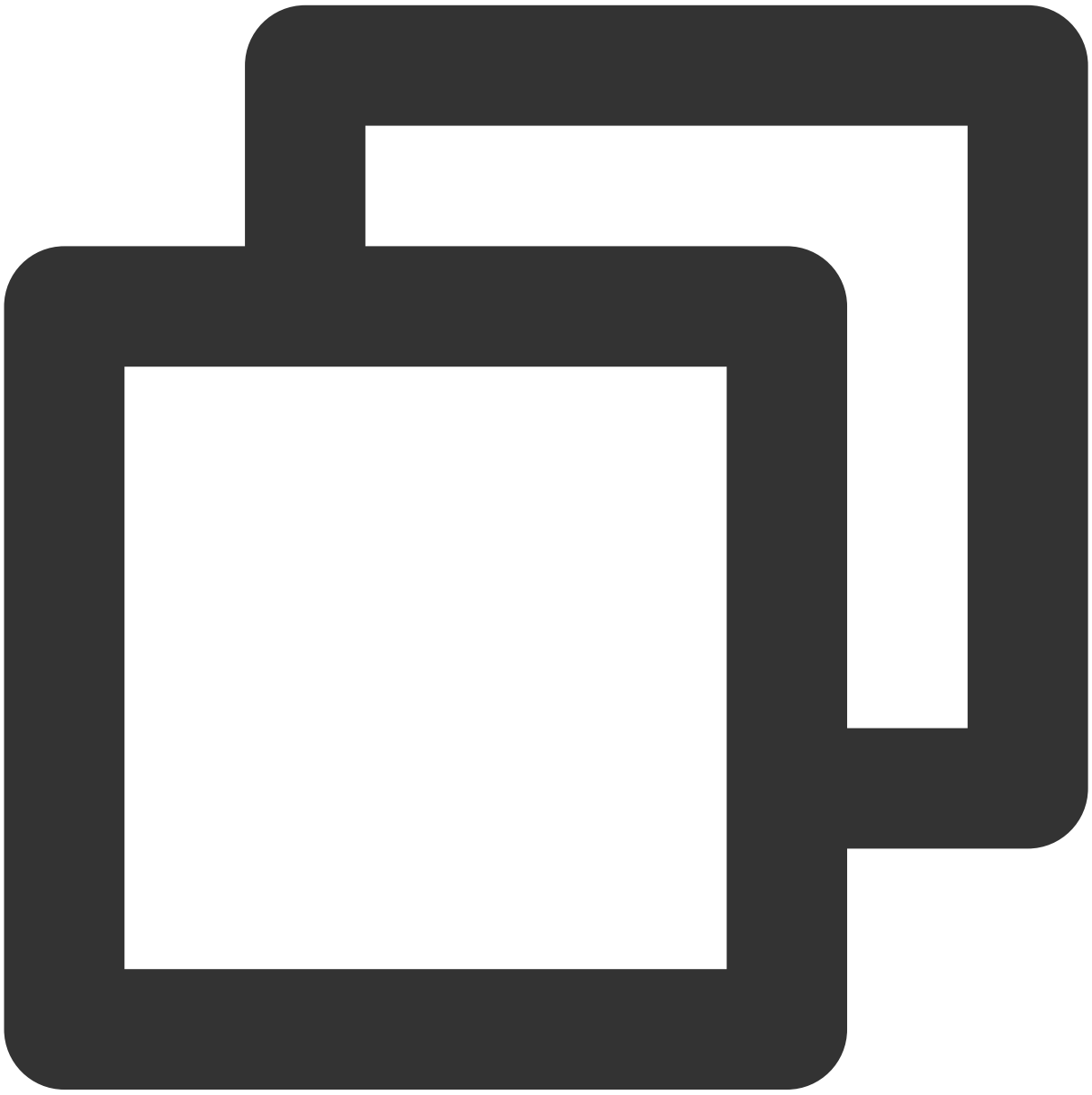
```
public abstract void sendRoomTextMsg(String message, TRTCVoiceRoomCallback.ActionCa
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
message	String	テキストメッセージ。
callback	ActionCallback	送信結果のコールバック。

sendRoomCustomMsg

カスタマイズしたテキストメッセージを送信します。



```
public abstract void sendRoomCustomMsg(String cmd, String message, TRTCVoiceRoomCal
```

パラメータは下表に示すとおりです：

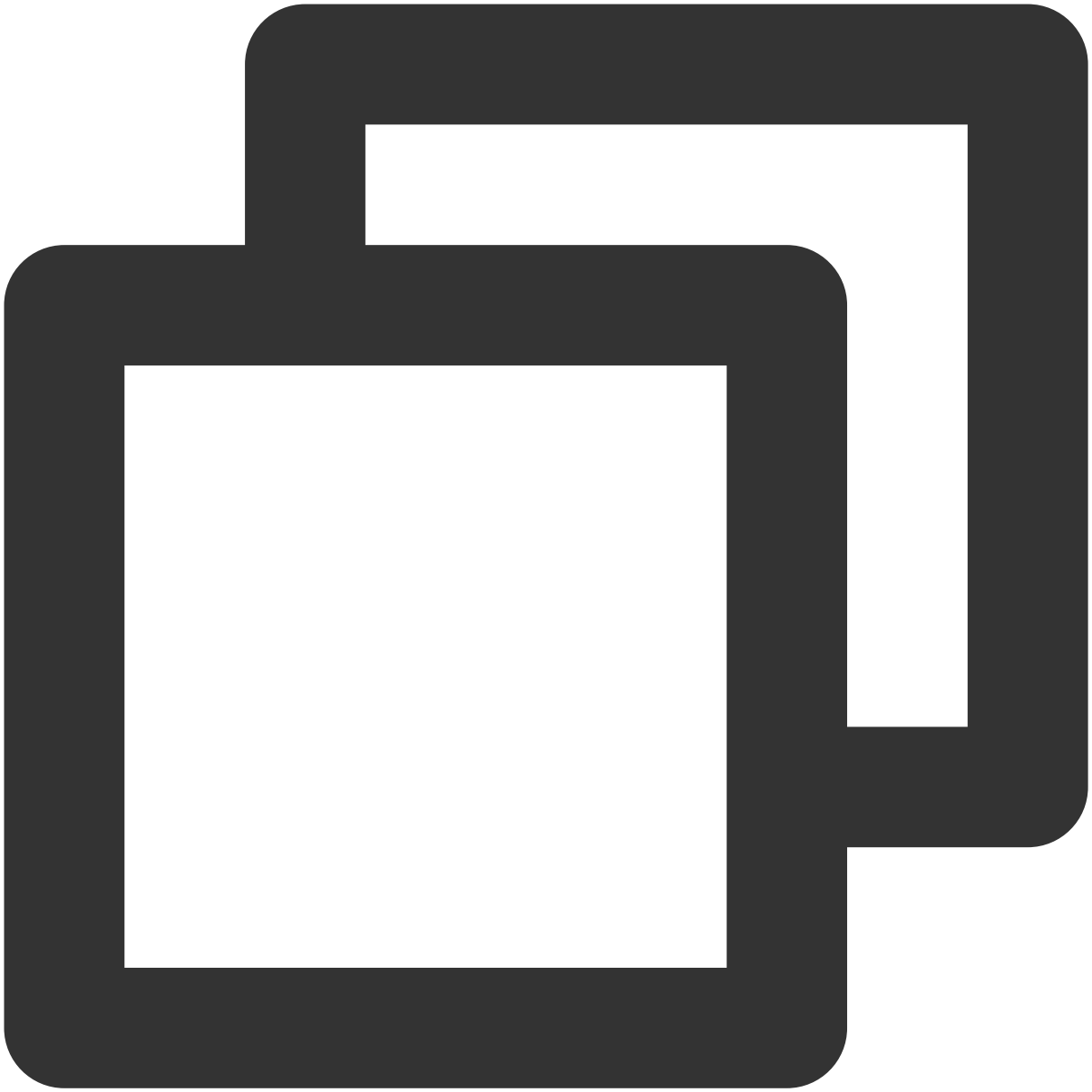
パラメータ	タイプ	意味

cmd	String	コマンドワードです。開発者がカスタマイズするものであり、主にさまざまなメッセージタイプを区別するために使用されます。
message	String	テキストメッセージ。
callback	ActionCallback	送信結果のコールバック。

招待シグナリング関連インターフェース

sendInvitation

ユーザーに招待を送信。



```
public abstract String sendInvitation(String cmd, String userId, String content, TR
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
cmd	String	業務カスタマイズコマンド。
userId	String	招待ユーザーID。
content	String	招待コンテンツ。

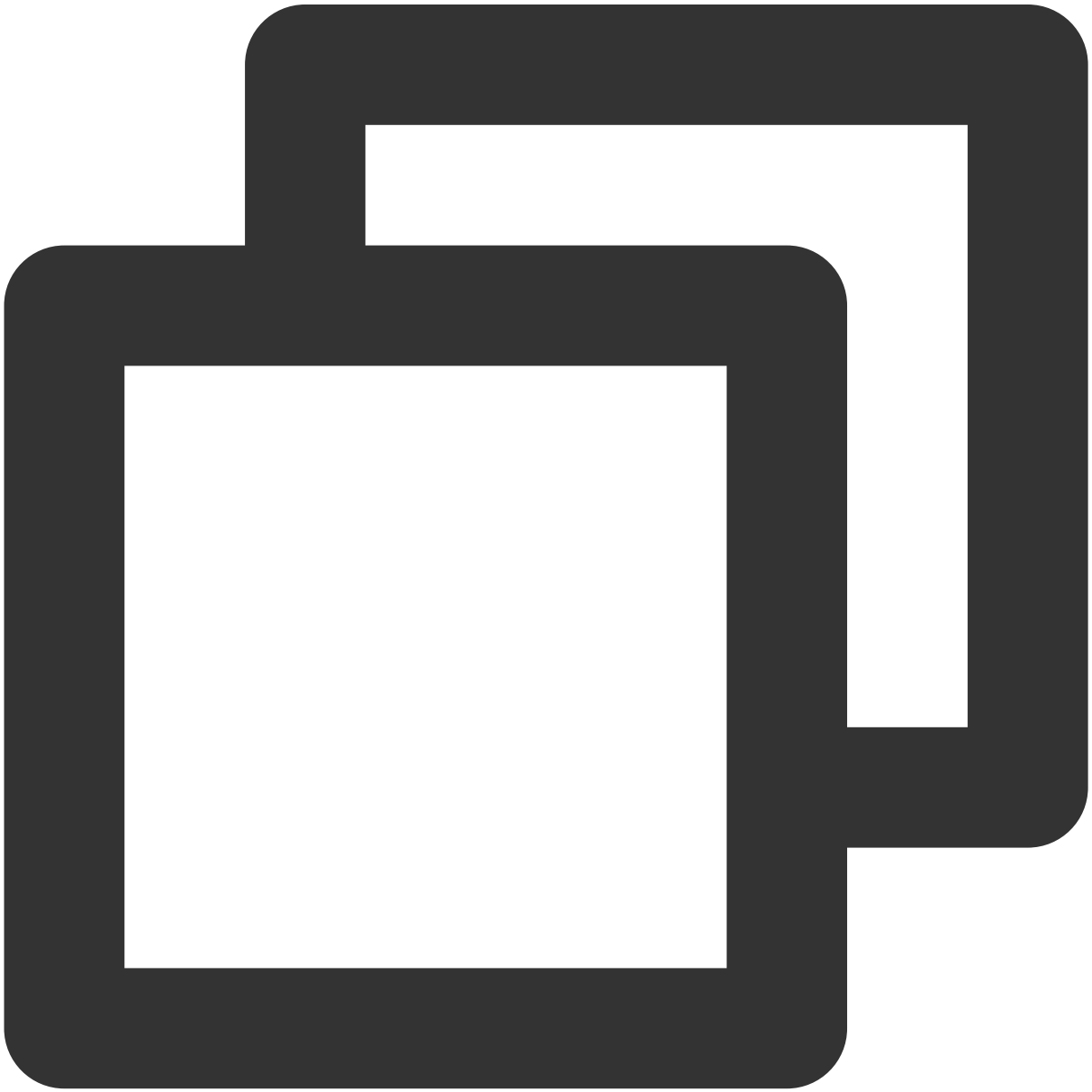
callback	ActionCallback	送信結果のコールバック。

戻り値：

戻り値	タイプ	意味
inviteId	String	今回の招待IDの識別に使用。

acceptInvitation

招待の同意。



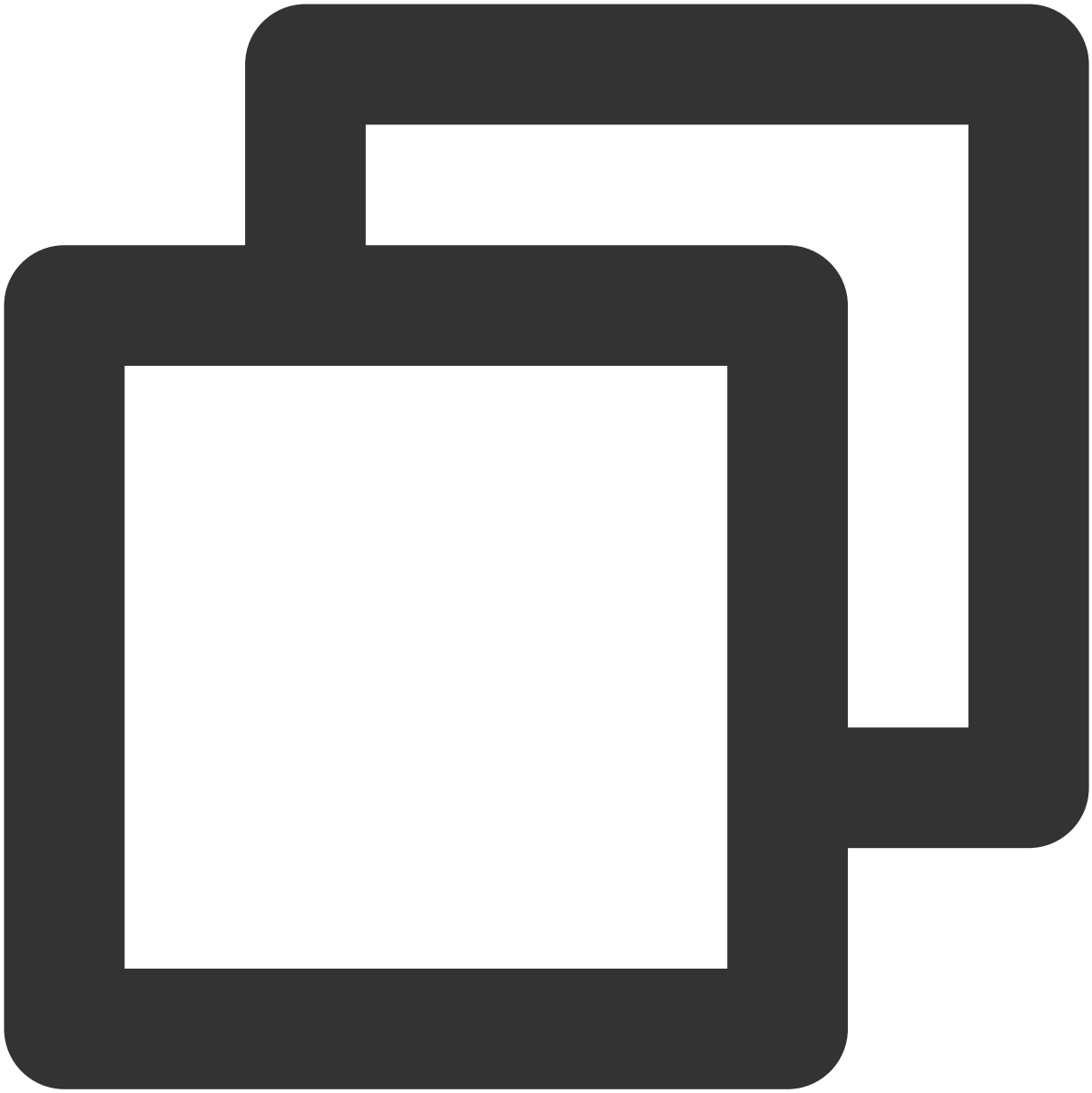
```
public abstract void acceptInvitation(String id, TRTCVoiceRoomCallback.ActionCallba
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
id	String	招待ID。
callback	ActionCallback	送信結果のコールバック。

rejectInvitation

招待の拒否。



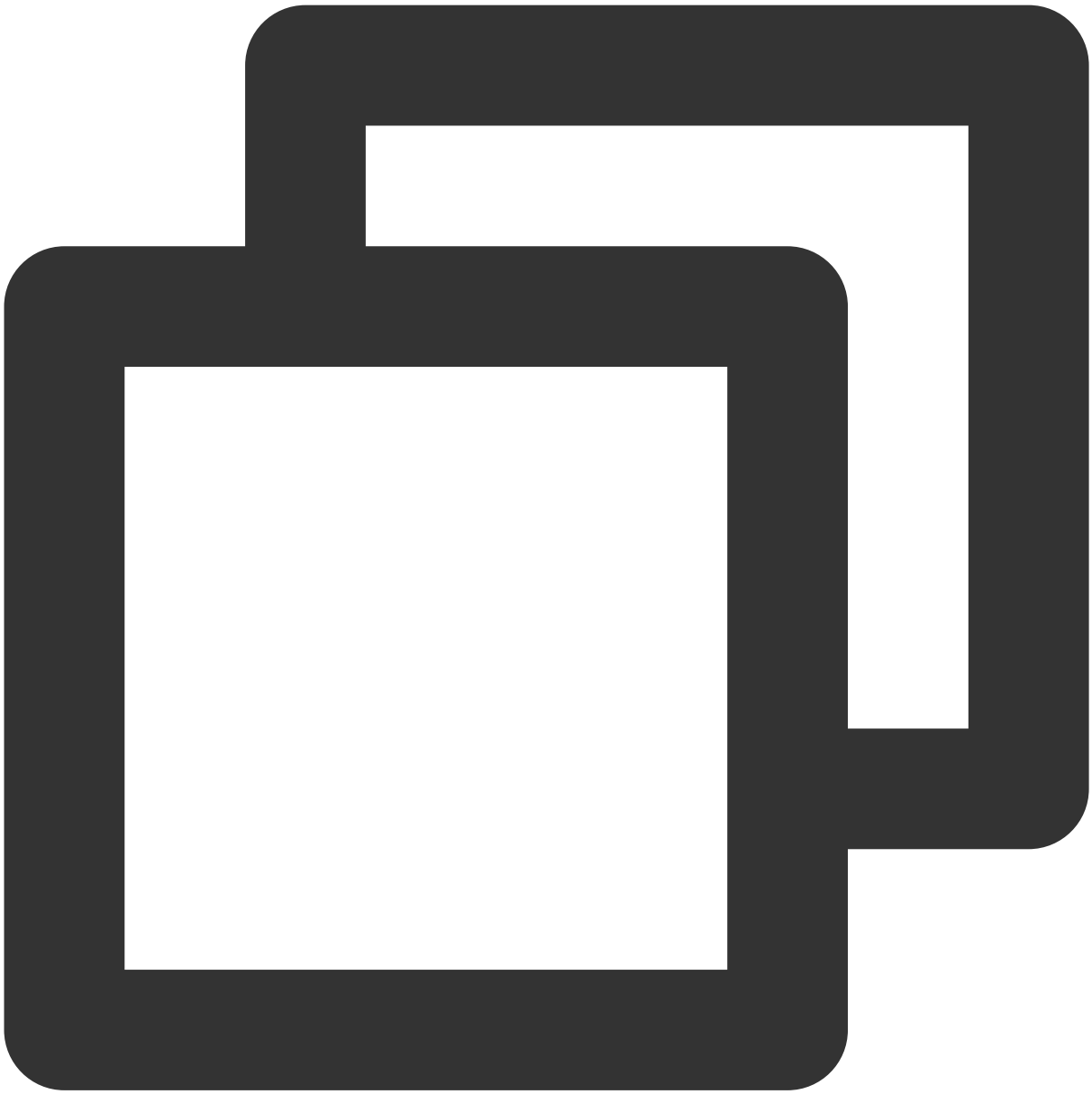
```
public abstract void rejectInvitation(String id, TRTCVoiceRoomCallback.ActionCallba
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
id	String	招待ID。
callback	ActionCallback	送信結果のコールバック。

cancelInvitation

招待の取り消し。



```
public abstract void cancelInvitation(String id, TRTCVoiceRoomCallback.ActionCallba
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
id	String	招待ID。
callback	ActionCallback	送信結果のコールバック。

TRTCVoiceRoomDelegateイベントのコールバック

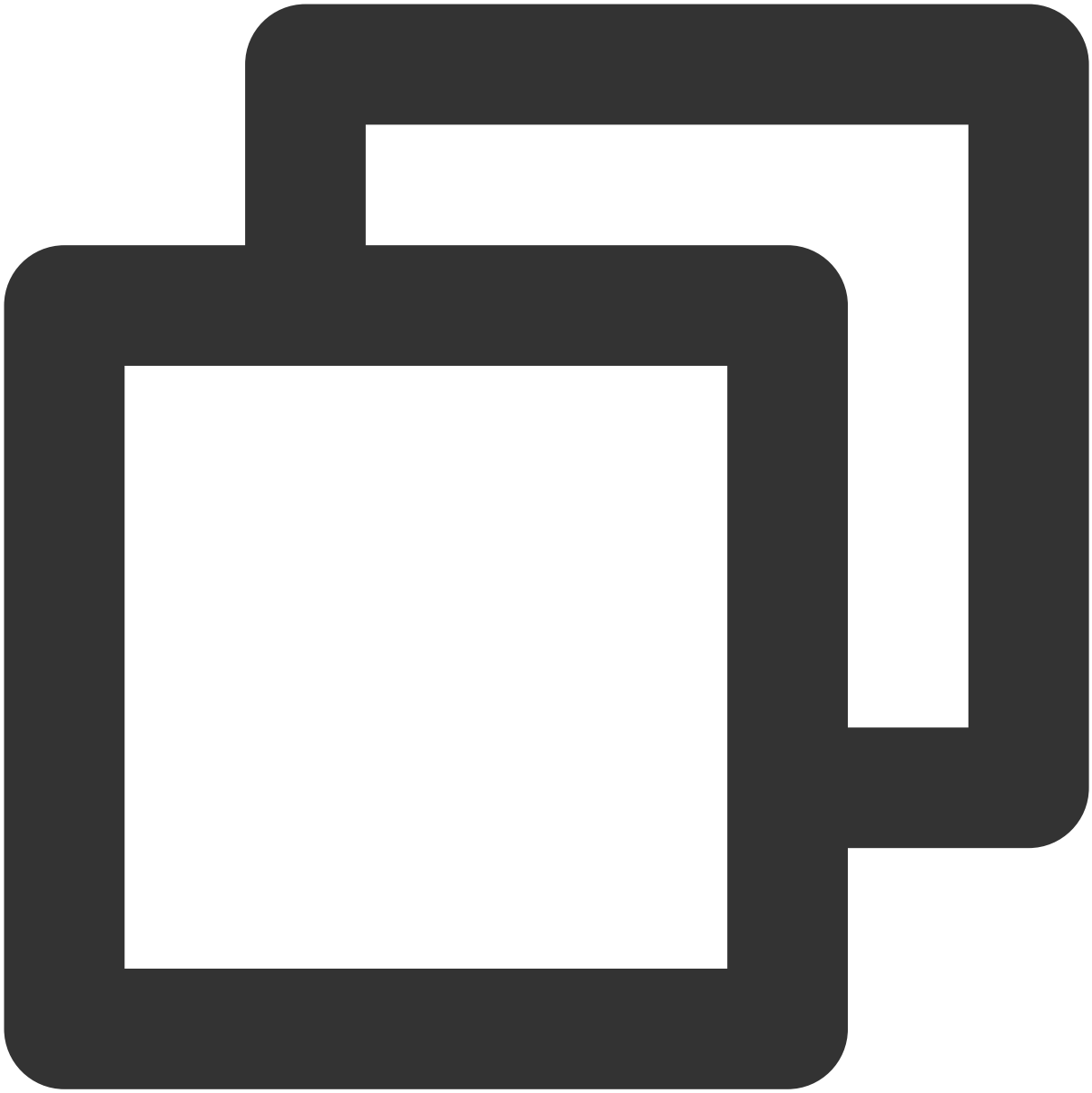
一般的なイベントコールバック

onError

エラーのコールバック。

説明：

SDKリカバリー不能なエラーは必ず監視し、状況に応じてユーザーに適切なインターフェースプロンプトを表示します。



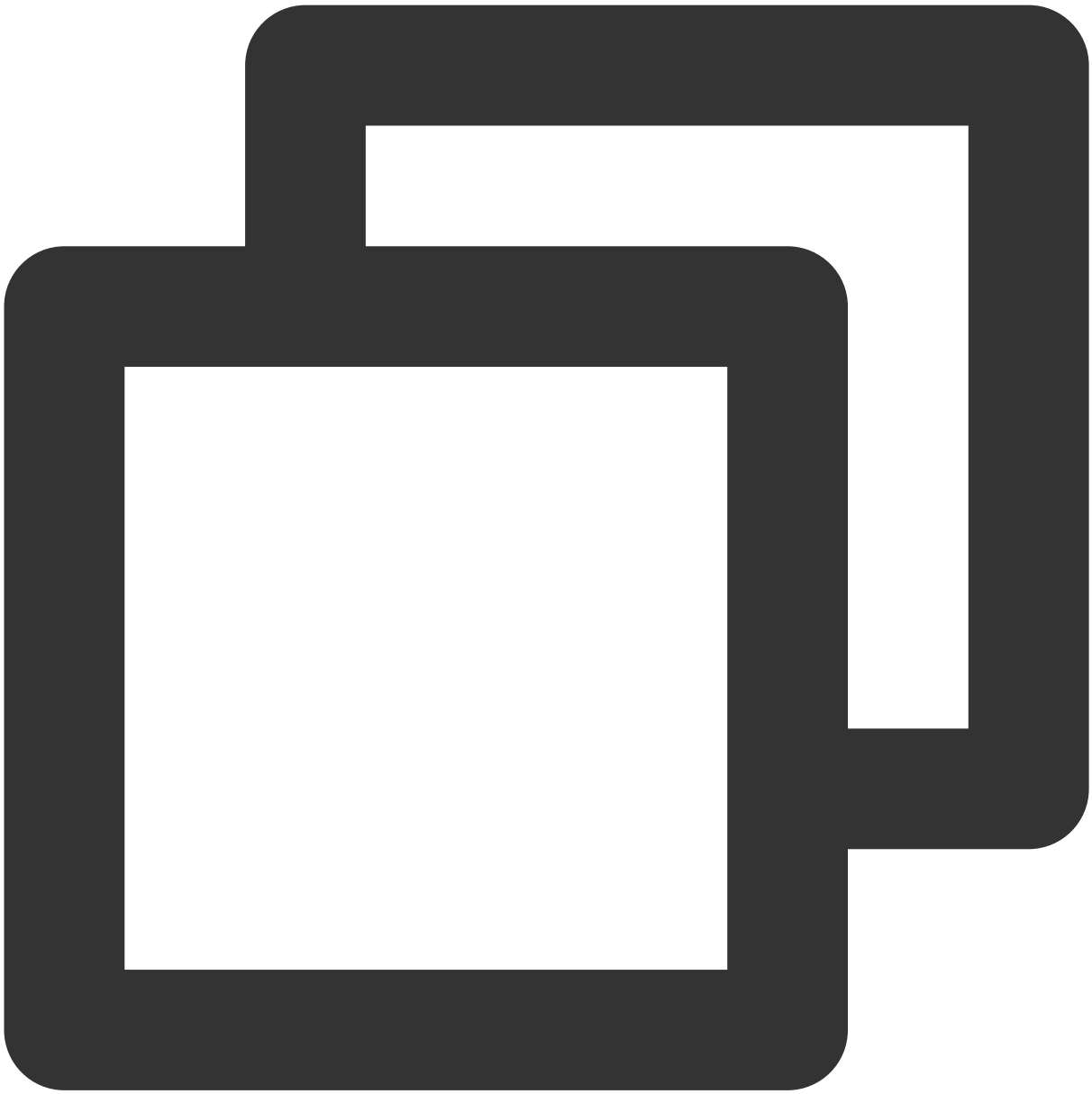
```
void onError(int code, String message);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
code	int	エラーコード。
message	String	エラーメッセージ。

onWarning

警告のコールバック。



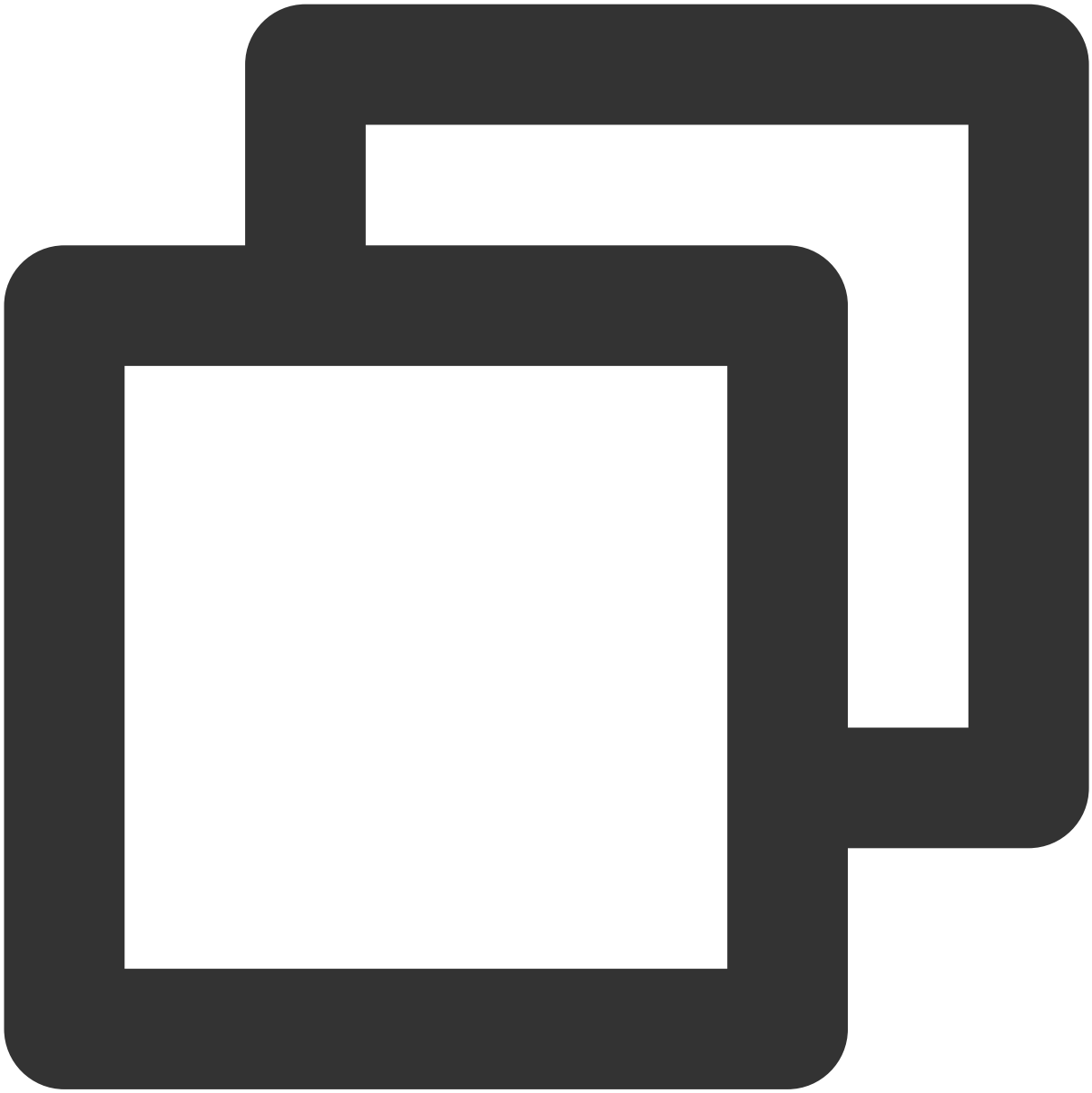
```
void onWarning(int code, String message);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
code	int	エラーコード。
message	String	警告メッセージ。

onDebugLog

Logコールバック。



```
void onDebugLog(String message);
```

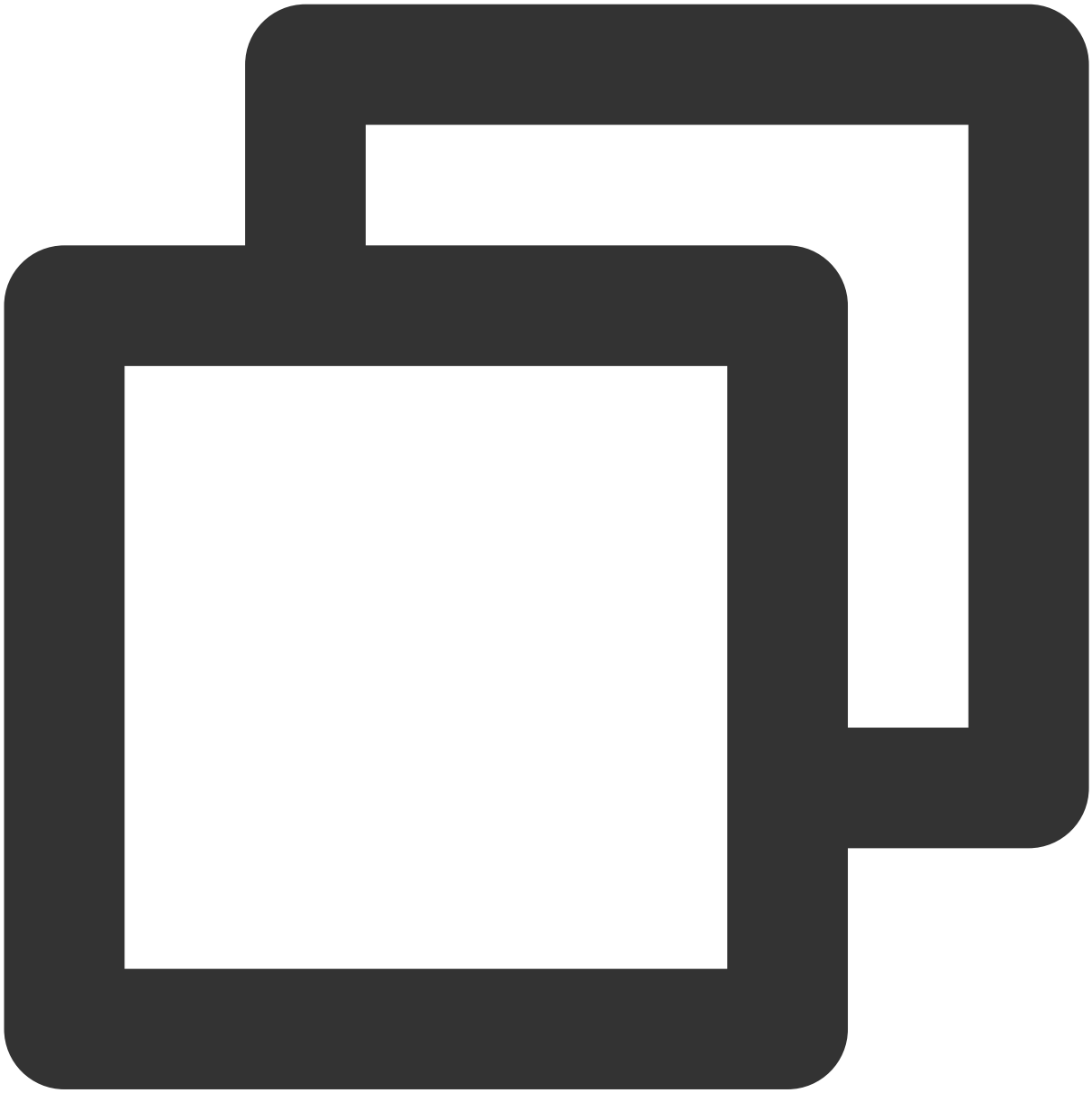
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
message	String	ログ情報。

ルームイベントのコールバック

onRoomDestroy

ルーム破棄のコールバック。管理者がルームを解散するとき、ルーム内の全ユーザーはこの通知を受信します。



```
void onRoomDestroy(String roomId);
```

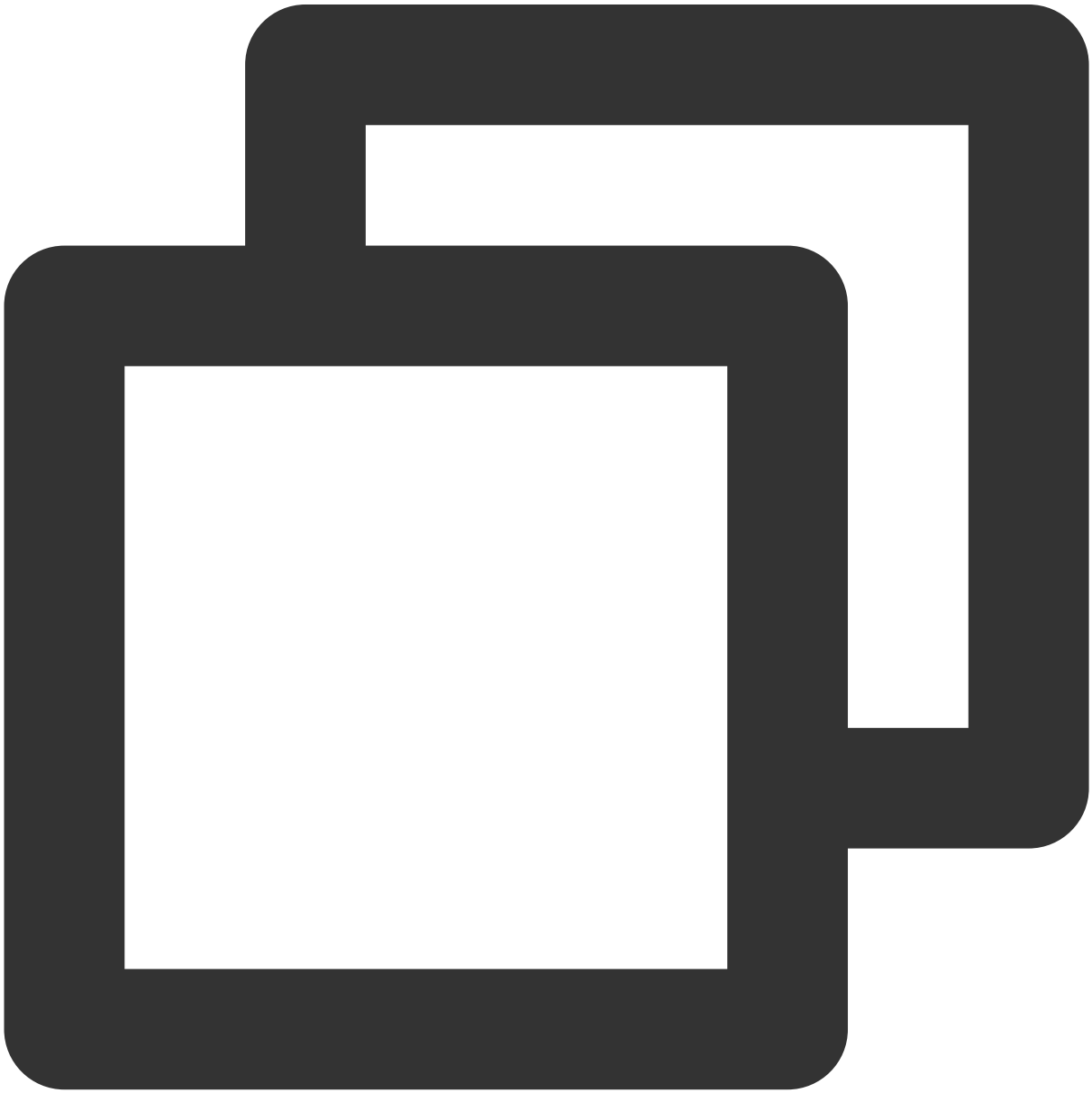
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味

roomId	String	ルームID。

onRoomInfoChange

入室に成功後、このインターフェースをコールバックします。roomInfoの情報は、管理者がルームを作成するときに渡されます。



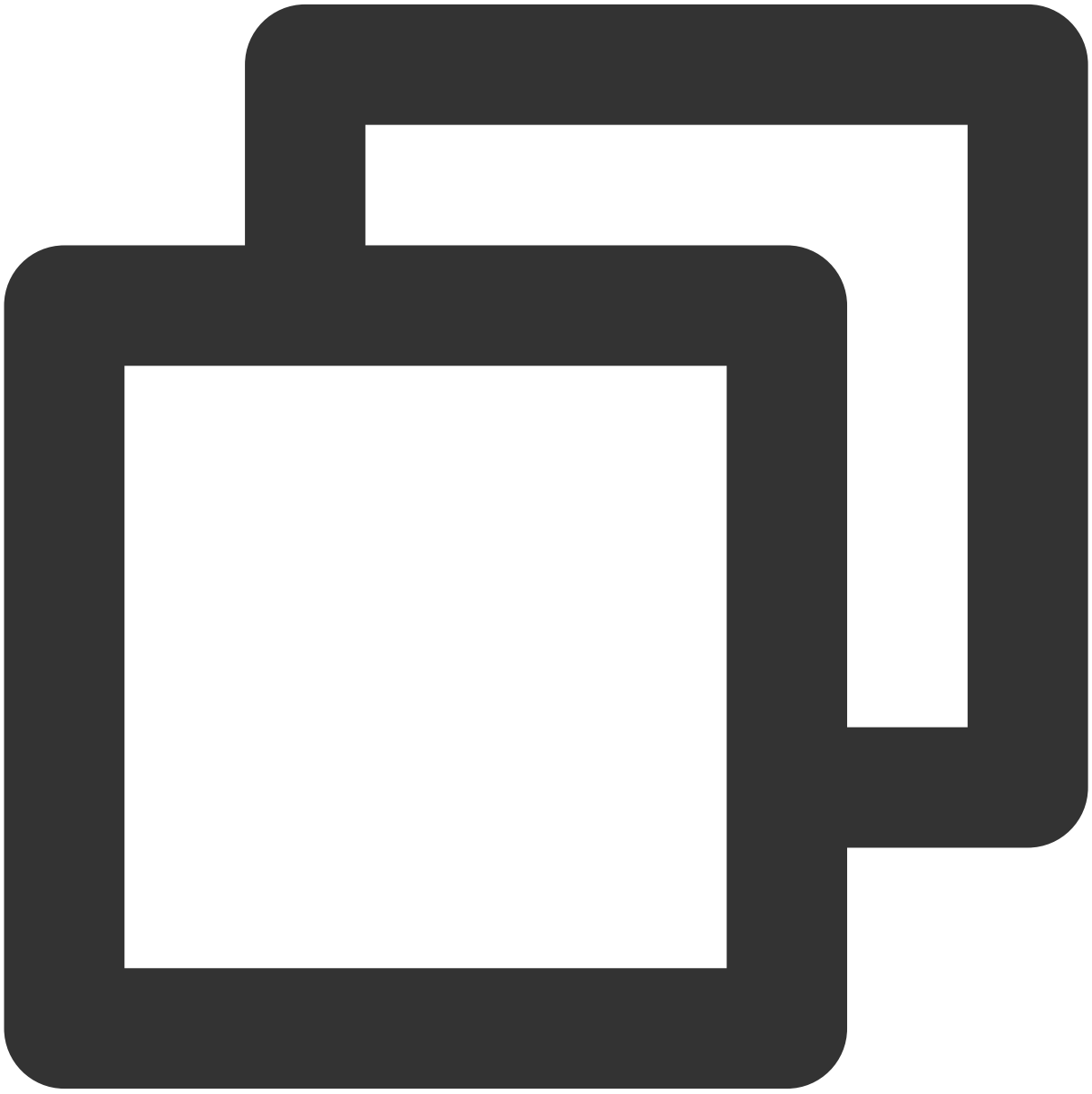
```
void onRoomInfoChange(TRTCVoiceRoomDef.RoomInfo roomInfo);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
roomInfo	RoomInfo	ルーム情報。

onUserMicrophoneMute

ユーザーのマイクがミュートになっているかどうかについて、ユーザーがmuteLocalAudioを呼び出すと、ルームの他のユーザーがこの通知を受信します。



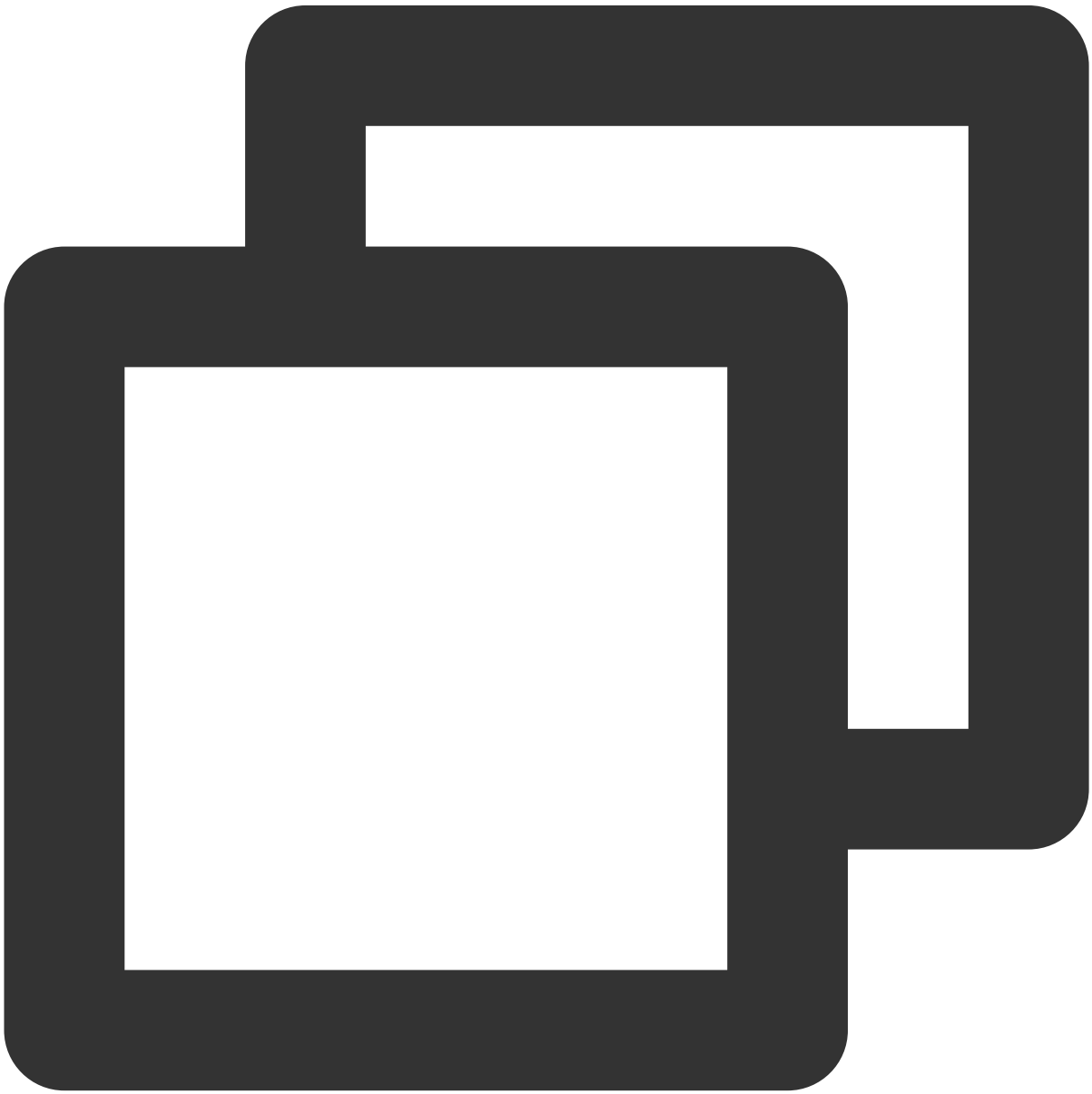
```
void onUserMicrophoneMute(String userId, boolean mute);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userId	String	ユーザーID。
mute	boolean	true：マイクミュート； false：ミュート解除。

onUserVolumeUpdate

音量レベルリマインダを有効にすると、各メンバーの音量を通知します。



```
void onUserVolumeUpdate(List<TRTCCloudDef.TRTCVolumeInfo> userVolumes, int totalVol
```

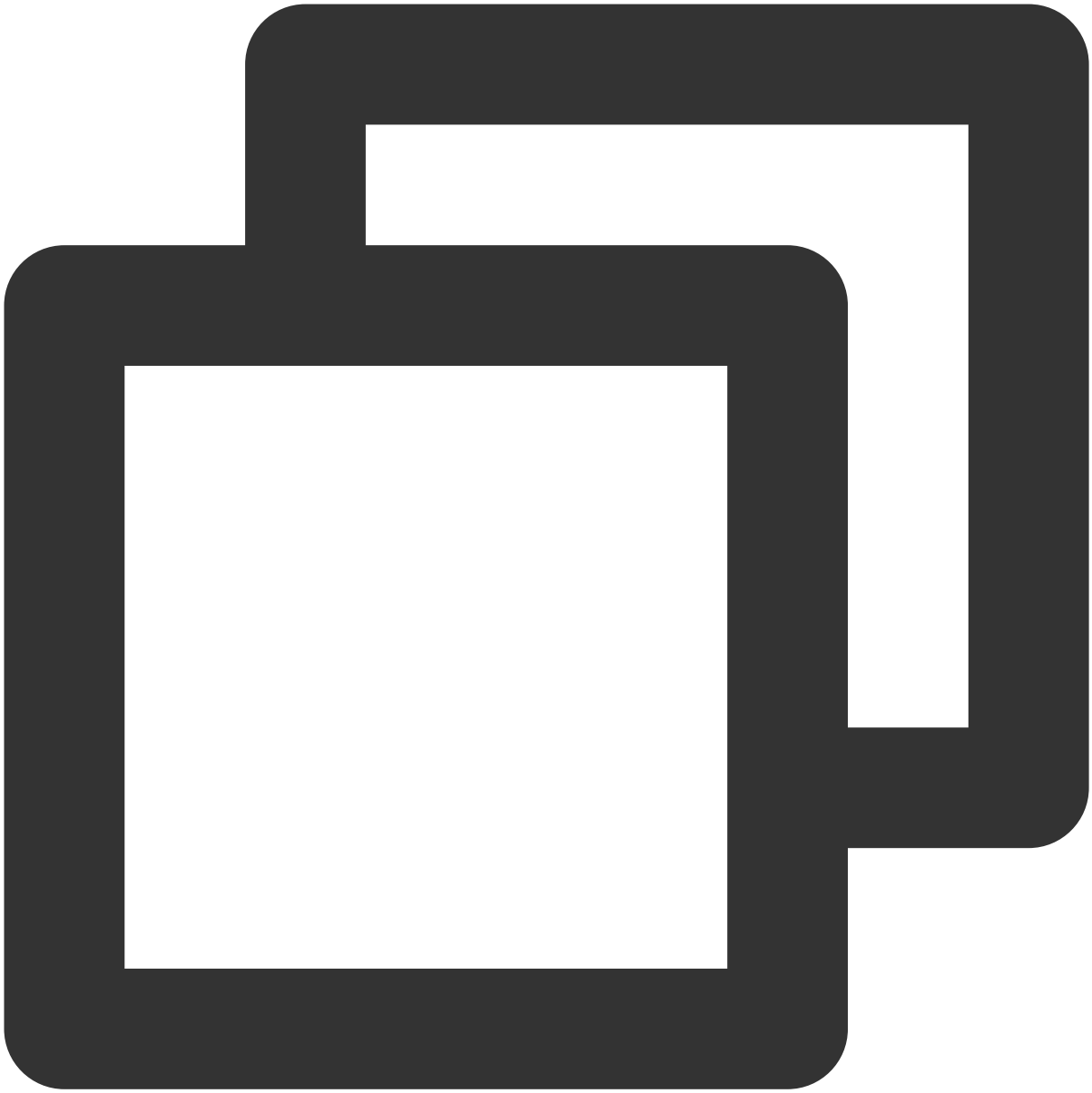
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userVolumes	ListList<TRTCCloudDef.TRTCVolumeInfo>	ユーザーリスト。
totalVolume	int	音量の大きさ。値：0～100。

マイクコールバック

onSeatListChange

全量のマイクリストの変更です。全てのマイクリストを含みます。



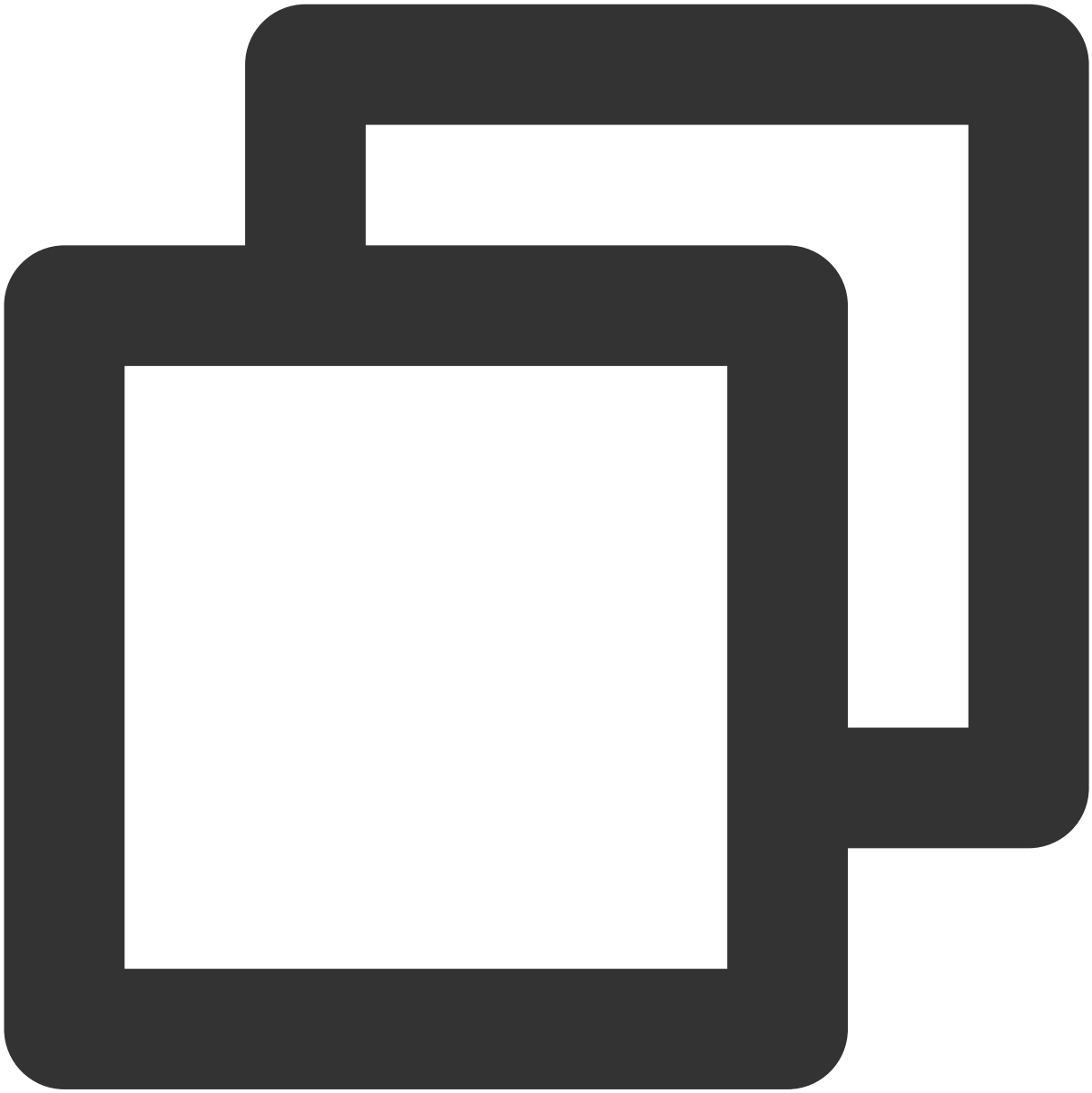
```
void onSeatListChange(List<SeatInfo> seatInfoList);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
seatInfoList	List<SeatInfo>	全量のマイクリスト。

onAnchorEnterSeat

発言者のメンバーがいます（ユーザーが発言者になる/管理者が視聴者を発言できるように招待）。



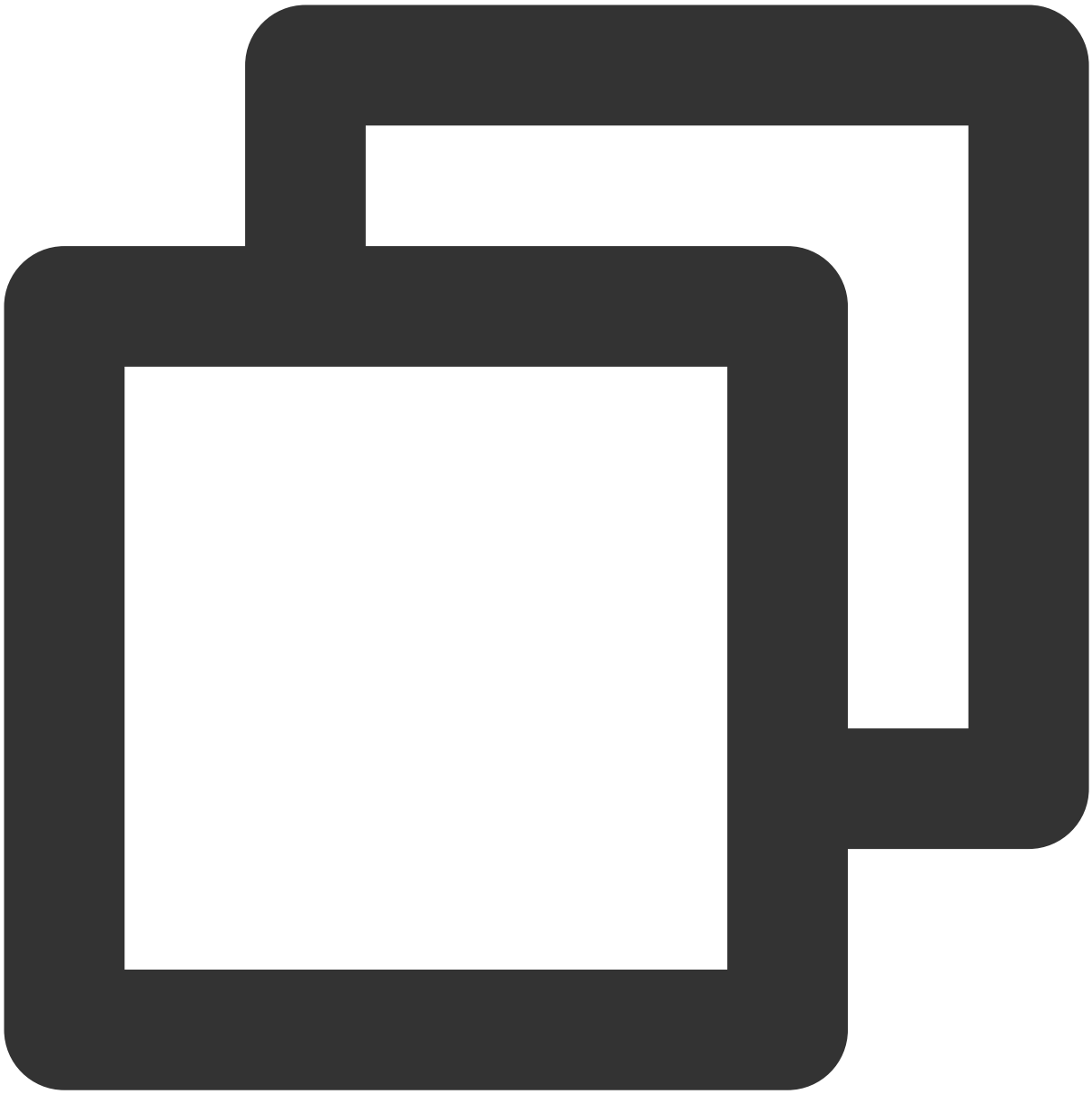
```
void onAnchorEnterSeat(int index, TRTCVoiceRoomDef.UserInfo user);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
index	int	メンバーがマイク・オンのマイク。
user	UserInfo	マイク・オンのユーザーの詳細情報。

onAnchorLeaveSeat

視聴者のメンバーがいます（ユーザーが視聴者になる/管理者がキックアウトしてマイク・オフ）。



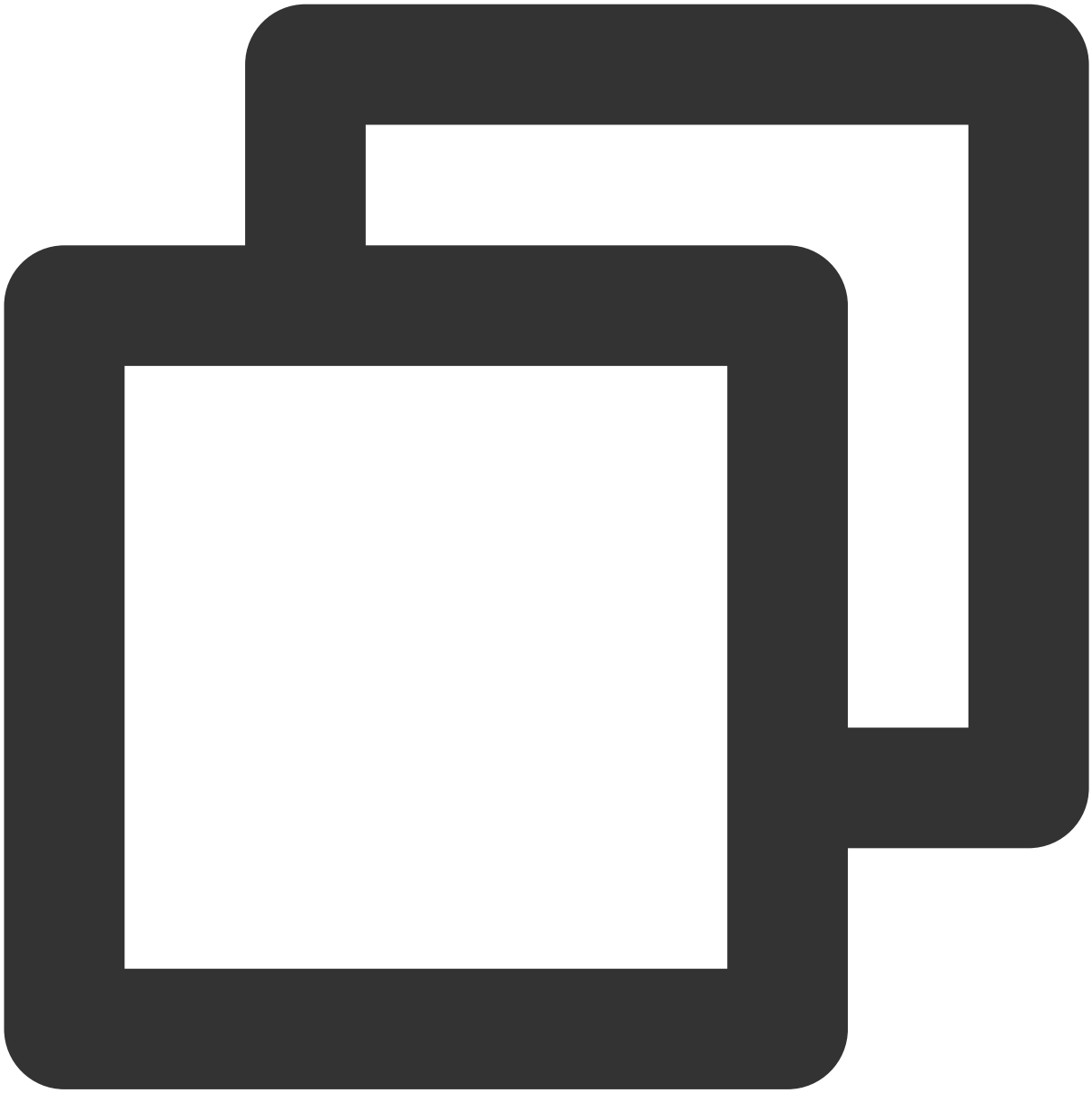
```
void onAnchorLeaveSeat(int index, TRTCVoiceRoomDef.UserInfo user);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
index	int	マイク・オフのマイク。
user	UserInfo	マイク・オンのユーザーの詳細情報。

onSeatMute

管理者のマイクミュート。



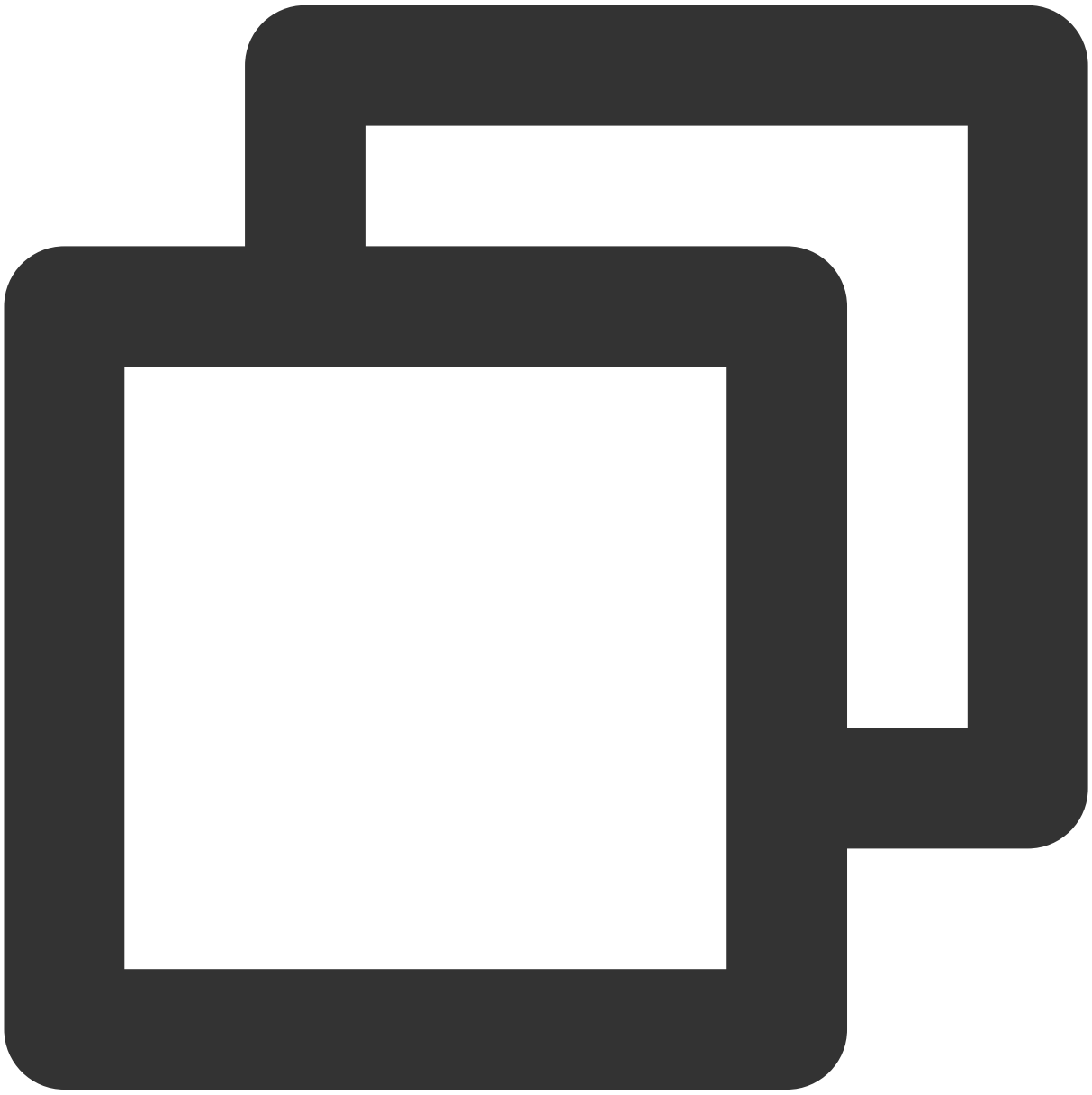
```
void onSeatMute(int index, boolean isMute);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
index	int	操作するマイク。
isMute	boolean	true：マイクミュート； false：ミュート解除。

onSeatClose

管理者のマイククローズ。



```
void onSeatClose(int index, boolean isClose);
```

パラメータは下表に示すとおりです：

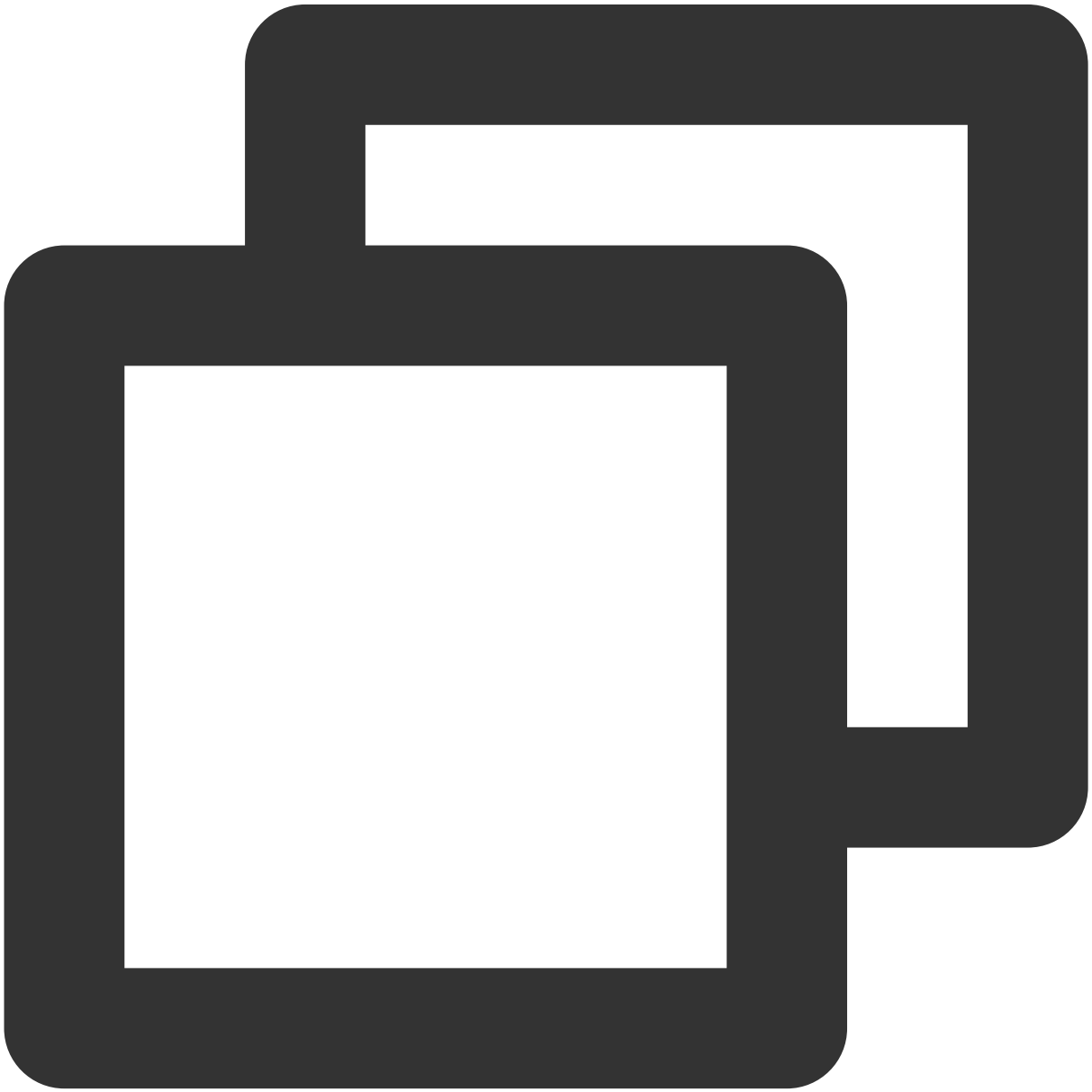
パラメータ	タイプ	意味
index	int	操作するマイク。

isClose	boolean	true：マイクのクローズ； false：マイクのクローズ解除。

リスナーの入退室イベントのコールバック

onAudienceEnter

リスナー入室通知の受信。



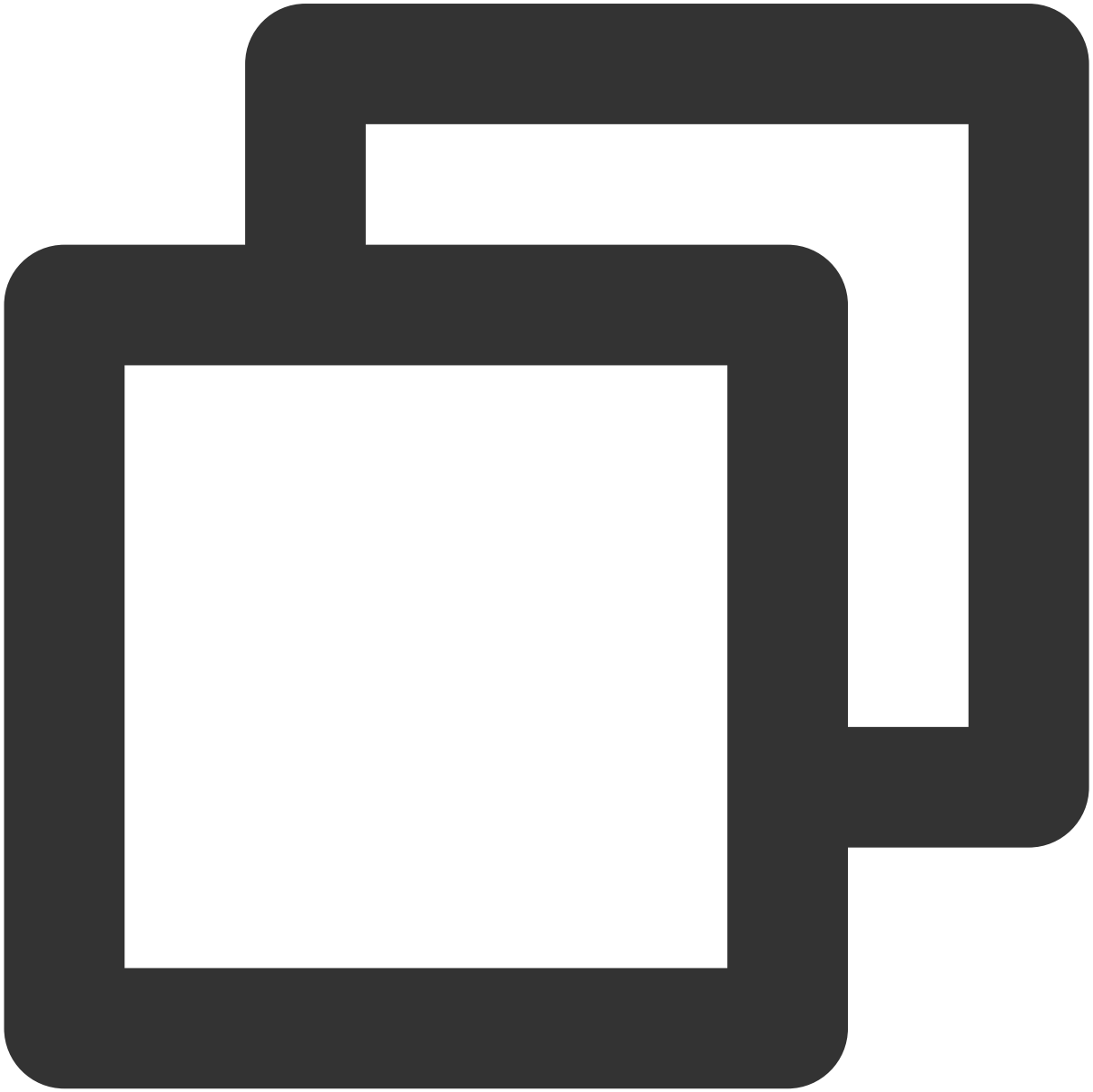
```
void onAudienceEnter(TRTCVoiceRoomDef.UserInfo userInfo);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userInfo	UserInfo	入室したリスナーの情報。

onAudienceExit

リスナー退室通知の受信。



```
void onAudienceExit(TRTCVoiceRoomDef.UserInfo userInfo);
```

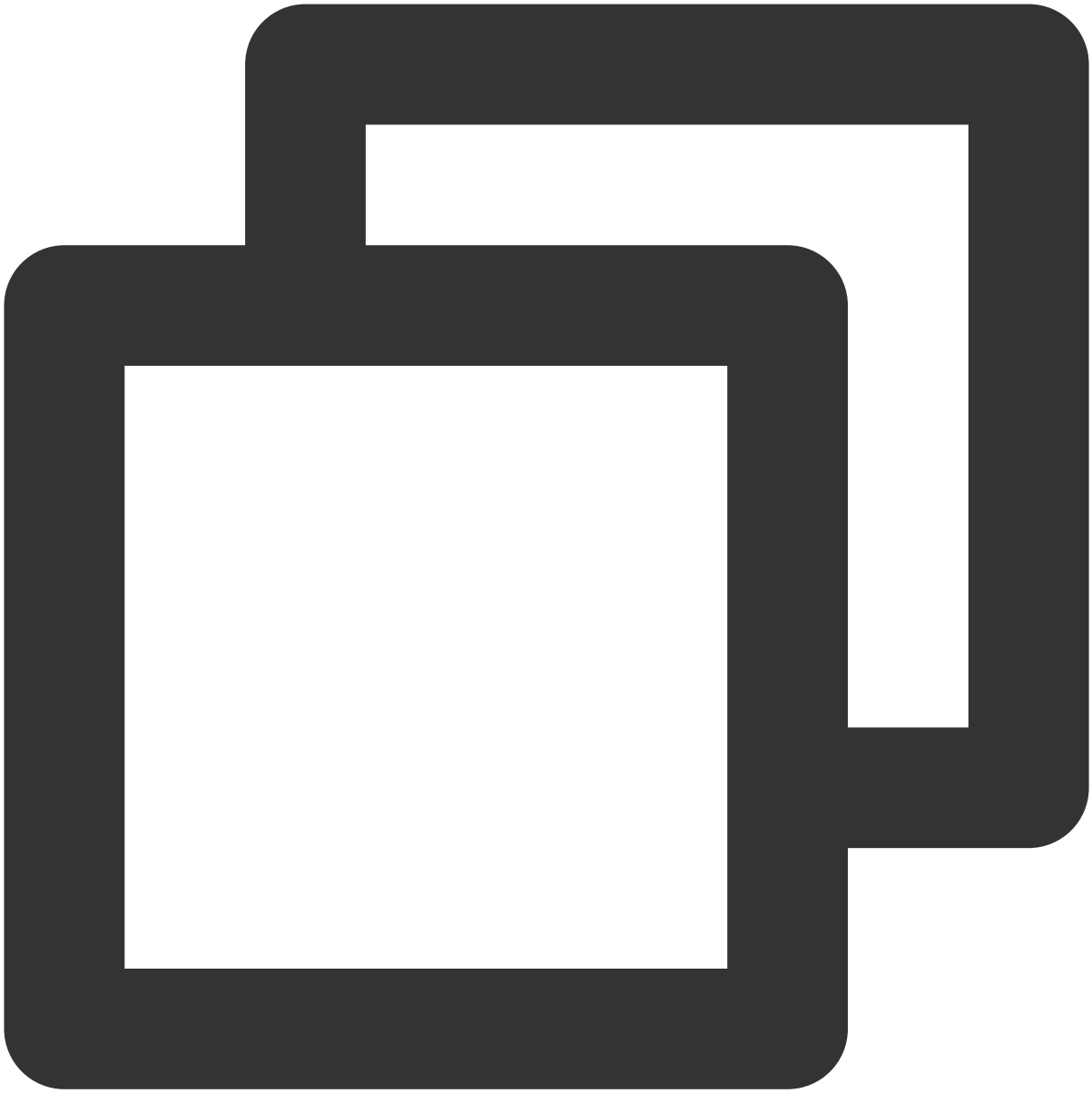
パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
userInfo	UserInfo	退室したリスナーの情報。

メッセージイベントのコールバック

onRecvRoomTextMsg

テキストメッセージを受信します。



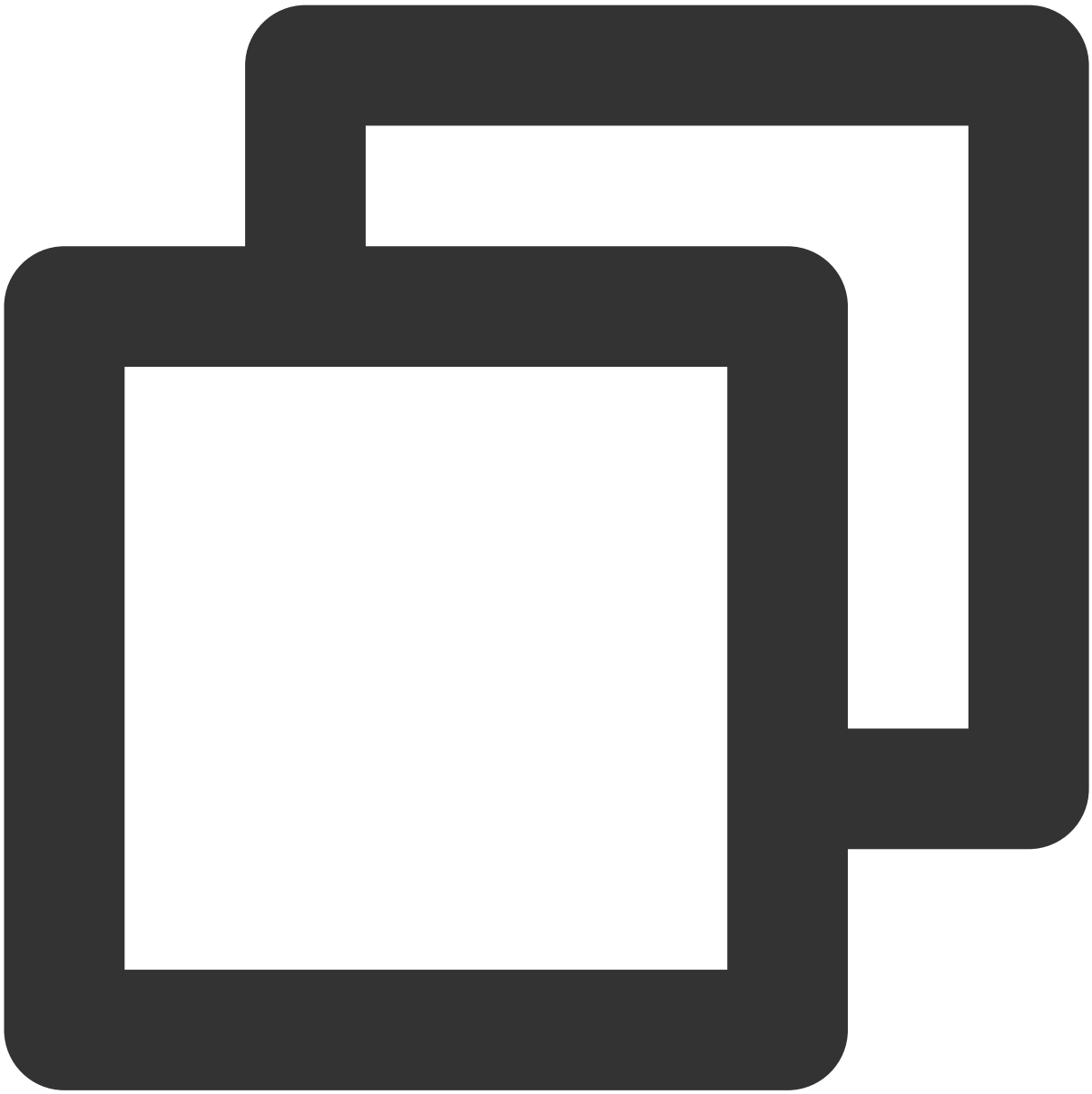
```
void onRecvRoomTextMsg(String message, TRTCVoiceRoomDef.UserInfo userInfo);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
message	String	テキストメッセージ。
userInfo	UserInfo	送信者のユーザー情報。

onRecvRoomCustomMsg

カスタムメッセージを受信します。



```
void onRecvRoomCustomMsg(String cmd, String message, TRTCVoiceRoomDef.UserInfo user
```

パラメータは下表に示すとおりです：

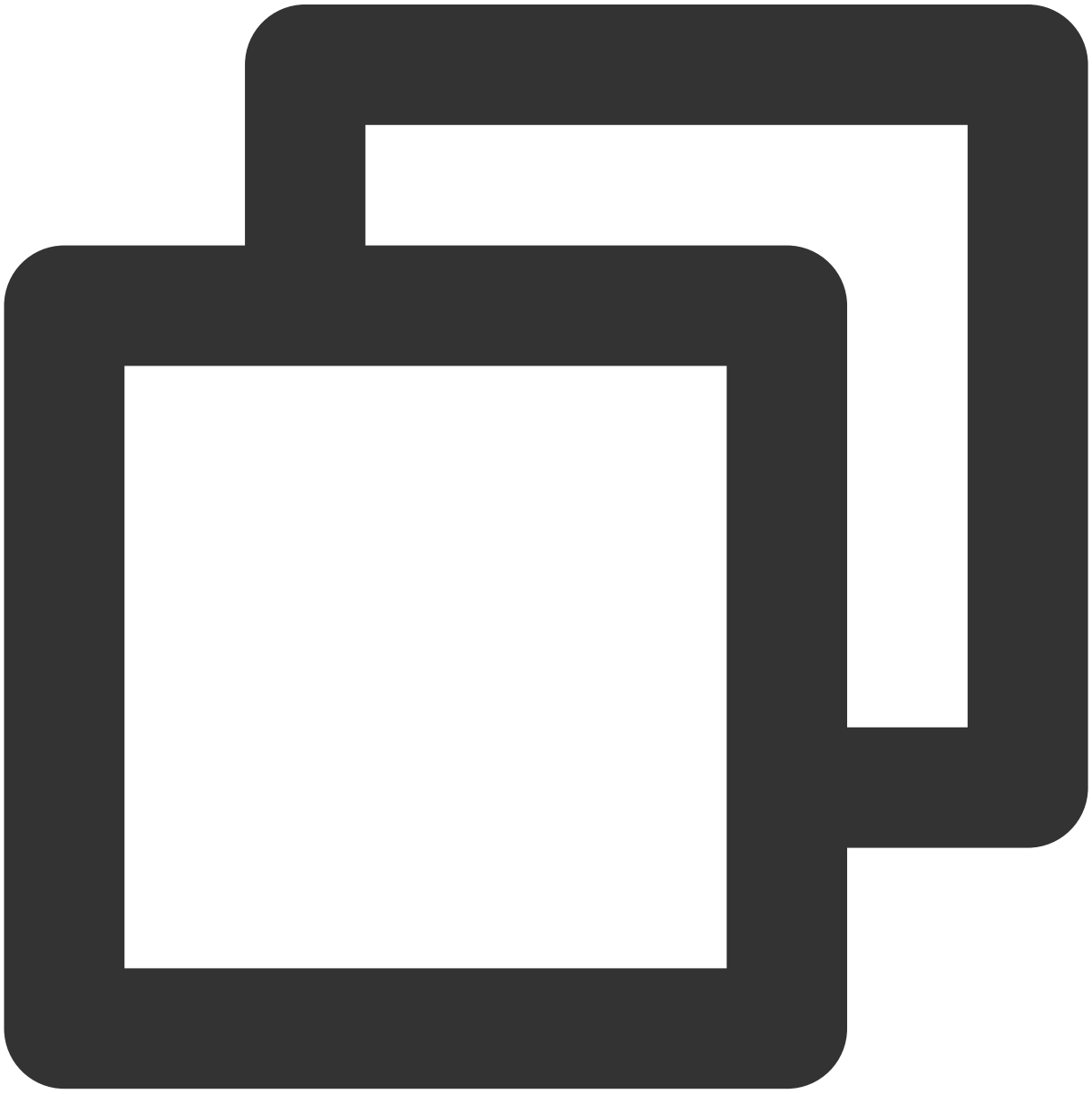
パラメータ	タイプ	意味
cmd	String	コマンドワードです。開発者がカスタマイズするものであり、主にさまざまなメッセージタイプを区別するために使用されます。

message	String	テキストメッセージ。
userInfo	UserInfo	送信者のユーザー情報。

招待シグナリングイベントのコールバック

onReceiveNewInvitation

新規招待リクエストの受信。



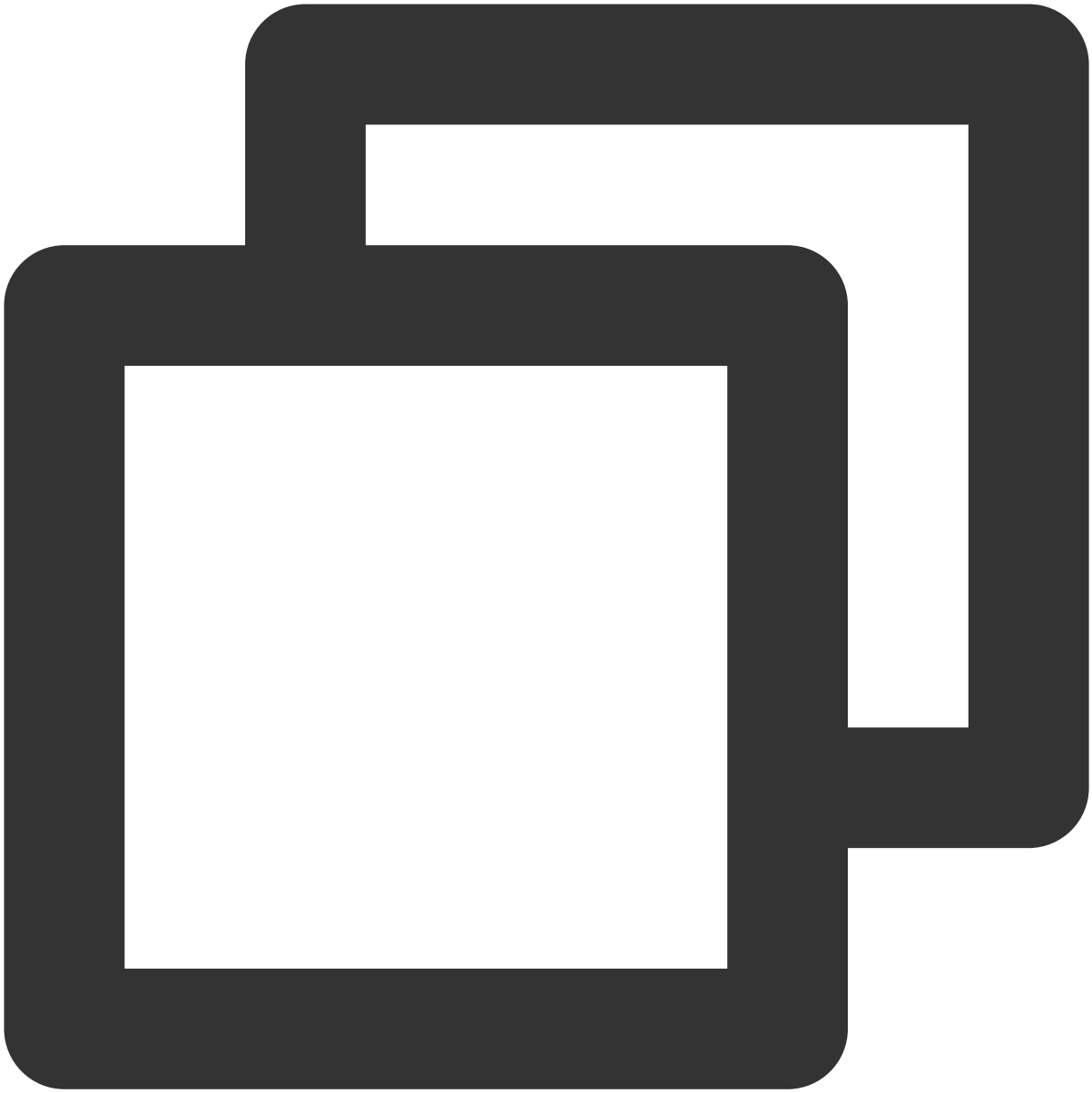
```
void onReceiveNewInvitation(String id, String inviter, String cmd, String content);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
id	String	招待ID。
inviter	String	招待者のユーザーID。
cmd	String	開発者がカスタマイズする業務指定のコマンドワードです。
content	String	業務指定のコンテンツ。

onInviteeAccepted

被招待者が招待に同意。



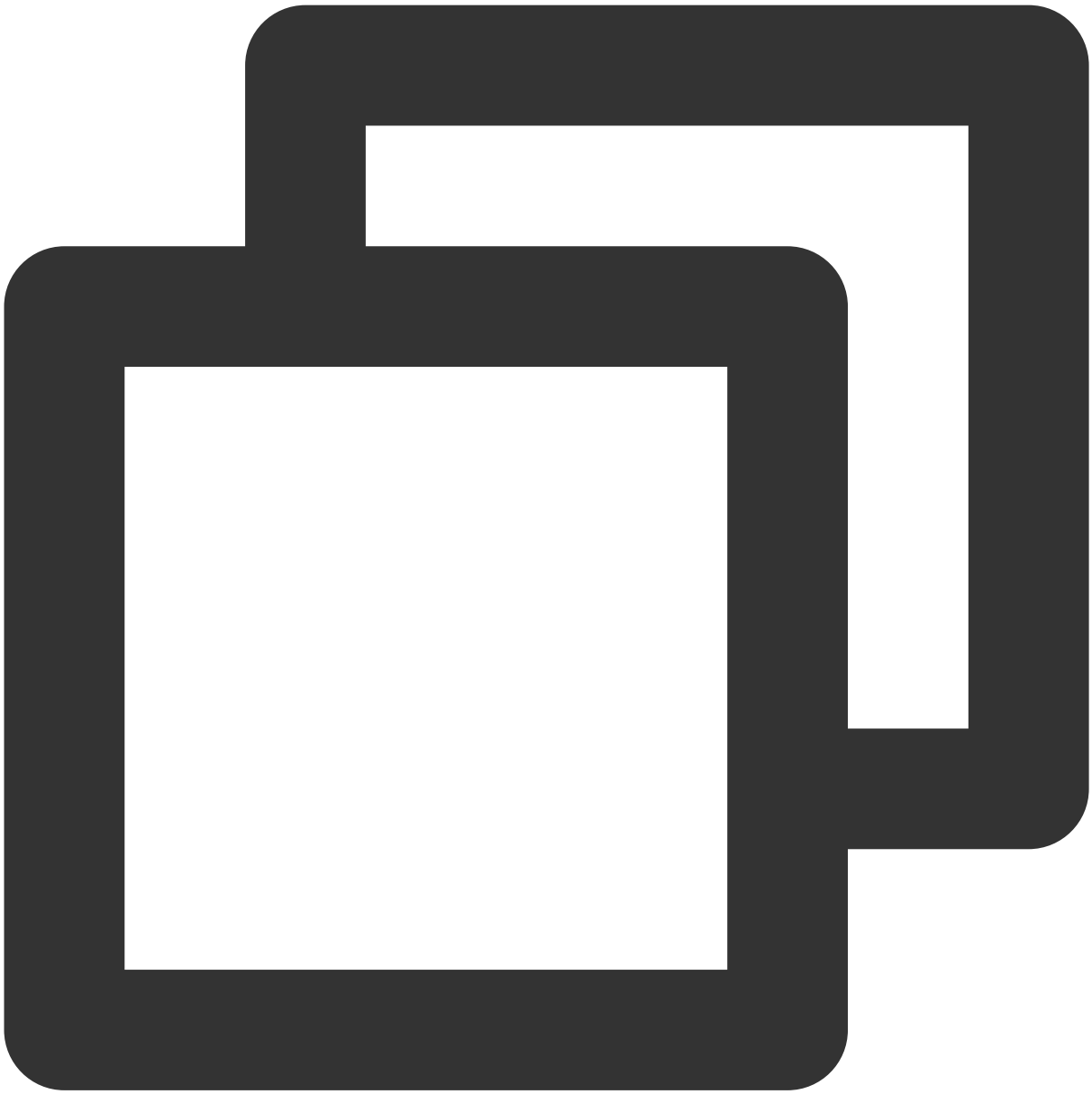
```
void onInviteeAccepted(String id, String invitee);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
id	String	招待ID。
invitee	String	被招待者のユーザーID。

onInviteeRejected

被招待者による招待の拒否。



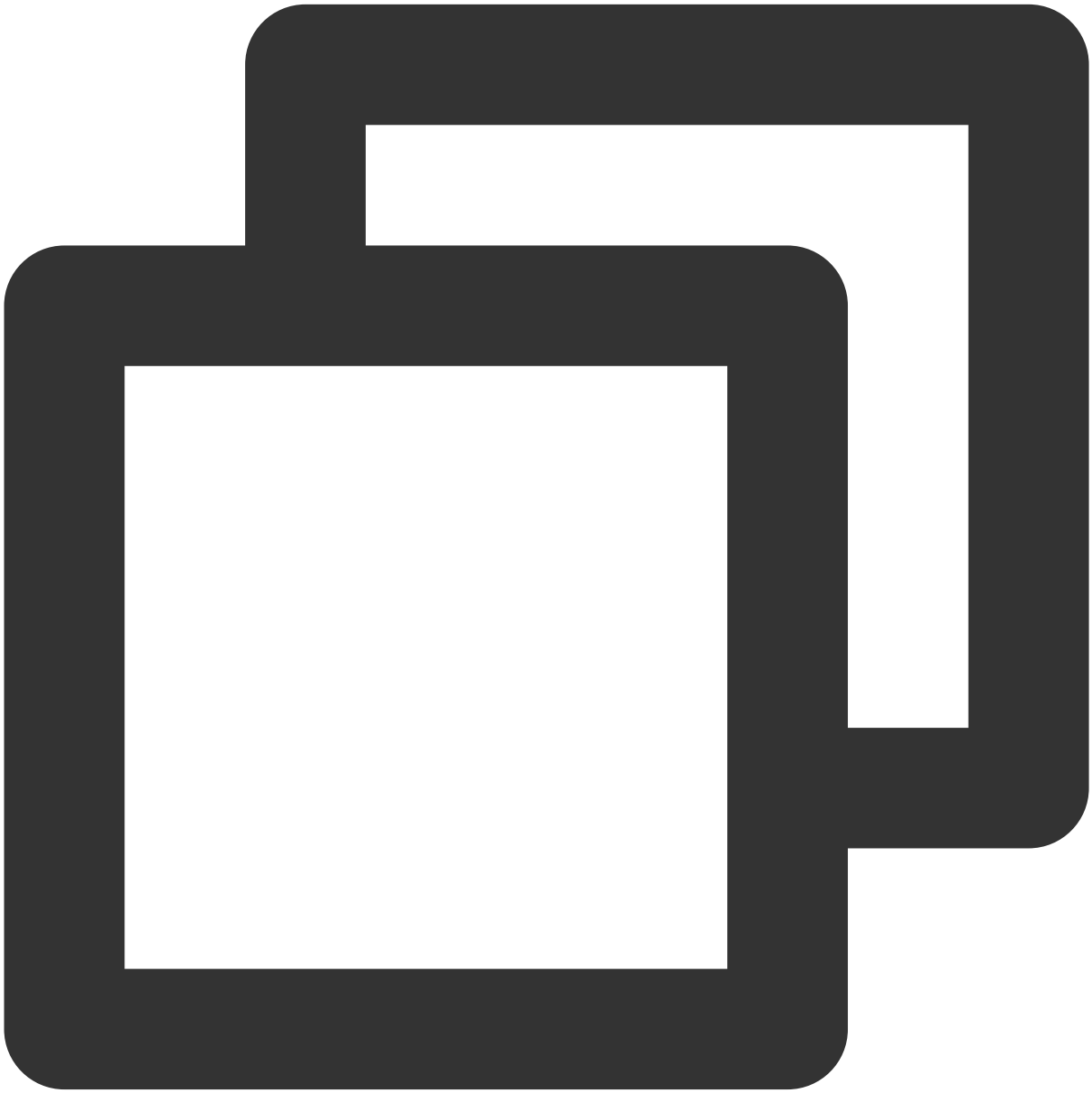
```
void onInviteeRejected(String id, String invitee);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
id	String	招待ID。
invitee	String	被招待者のユーザーID。

onInvitationCancelled

招待者が招待を取り消し。



```
void onInvitationCancelled(String id, String inviter);
```

パラメータは下表に示すとおりです：

パラメータ	タイプ	意味
id	String	招待ID。
inviter	String	招待者のユーザーID。