Tencent Real-Time Communication

Video Calling (Including UI)

製品ドキュメント

カタログ：

Video Calling (Including UI)
コンポーネントの説明（TUICallKit）

最終更新日：：2024-07-19 14:53:21

コンポーネントの説明

TUICallKitはTencent Cloudがリリースするオーディオビデオ通話UIコンポーネントです。この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にオーディオビデオ通話機能を追加することができ、さらにオフライン通知機能もサポートします。TUICallKitはAndroid、iOS、Webなどの複数の開発プラットフォームでサポートしています。基本機能は下図のとおりです。

ご注意:

開発者からのフィードバックと市場ニーズのリサーチを基に、2022年8月、TUICallingはTUICallKitへとアップグレードしました。新しいコンポーネントはグループ内の多人数通話やオフライン通知などの機能をサポートしま

す。また、より割引率の高いオーディオビデオ通話SDKパッケージおよび60日間の無料トライアル期間をご用意しています。この機会にぜひご利用ください。

機能のメリット

便利なアクセス：UI付きのオープンソースコンポーネントを提供することで、開発時間を90%節約でき、オーディオビデオ通話アプリケーションをスピーディーにリリースできます。
プラットフォーム相互運用性：各プラットフォームのTUICallKitコンポーネントは通話の発信、応答、終了などを相互にサポートし、相互運用が可能です。
多人数通話：1対1のビデオ通話だけでなく、グループ内で多人数ビデオ通話を開始することもできます。
オフラインプッシュ：Android&iOSのオフラインプッシュをサポートしています。着呼側ユーザーのAppはオフラインのときでも新しい着信通知を受け取ることができます。

適用ケース

2人または複数人のオーディオビデオ通話が行えます。ゲームSNS、オンラインカスタマーサービス、ビデオカスタマーサービス、オンライン問診、保険相談などのシーンに適します。

TUICallKitのダウンロード

オープンソース構築

アップグレード前のTUICallingオープンソースプロジェクトはここで見つけることができます。

クイックアクセス（TUICallKit）
Android

最終更新日：：2024-07-19 14:53:21

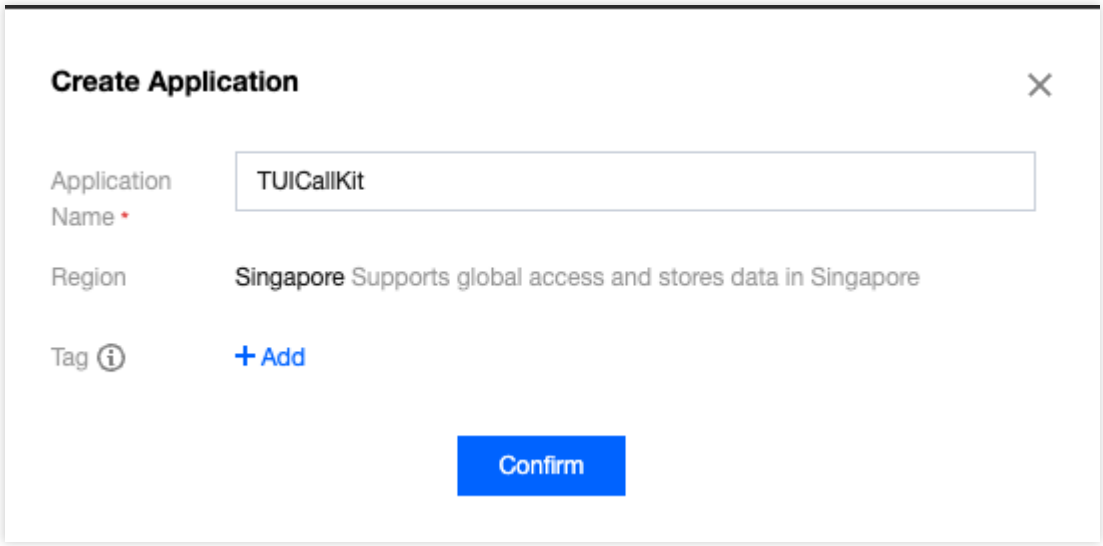
ここでは、最短の時間でTUICallKitコンポーネントの統合を完了する方法についてご説明します。このドキュメントに沿って操作することで、1時間以内に次の重要手順を完了し、最終的にUIインターフェースを完備したビデオ通話機能を手にすることができます。

環境の準備

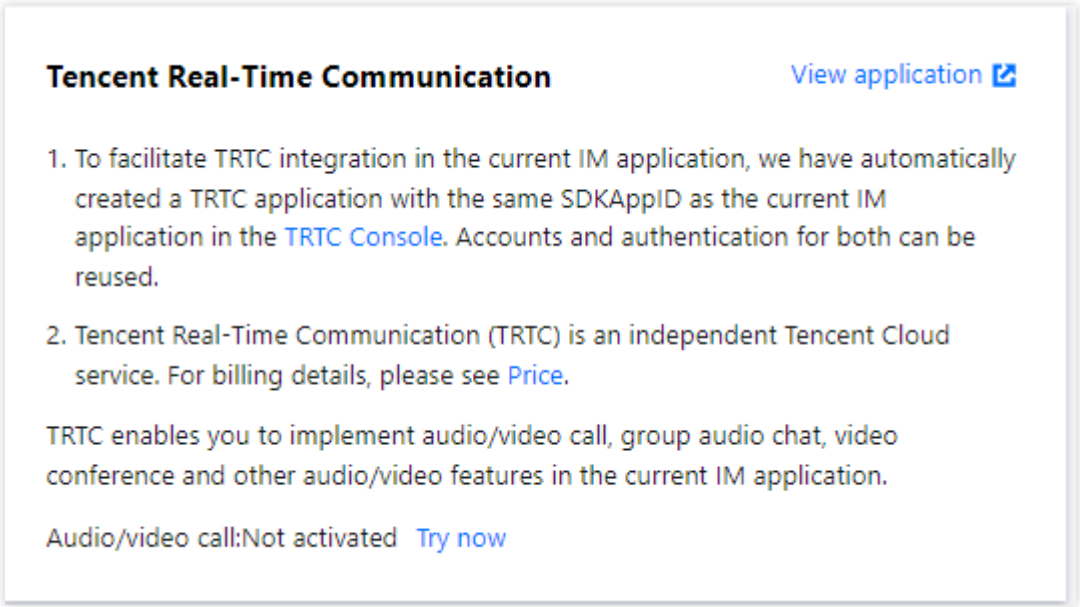
互換性のある最低バージョンはAndroid 4.1（SDK API Level 16）です。Android 5.0（SDK API Level 21）およびそれ以降のバージョンの使用を推奨します。
Android Studio 3.5およびそれ以降のバージョン（Gradle 3.5.4およびそれ以降のバージョン）。
Android 4.1およびそれ以降のスマートフォンデバイス。

ステップ1：サービスのアクティブ化

TUICallKitはTencent CloudのIMと、TRTCという2つの有料PaaSサービスをベースに構築したオーディオビデオ通信コンポーネントです。以下の手順で関連のサービスをアクティブ化し、60日間の無料トライアルサービスを体験することができます。
1. IMコンソールにログインし、**新しいアプリケーションの作成**をクリックし、ポップアップしたダイアログボックスにアプリケーション名を入力して**OK**をクリックします。



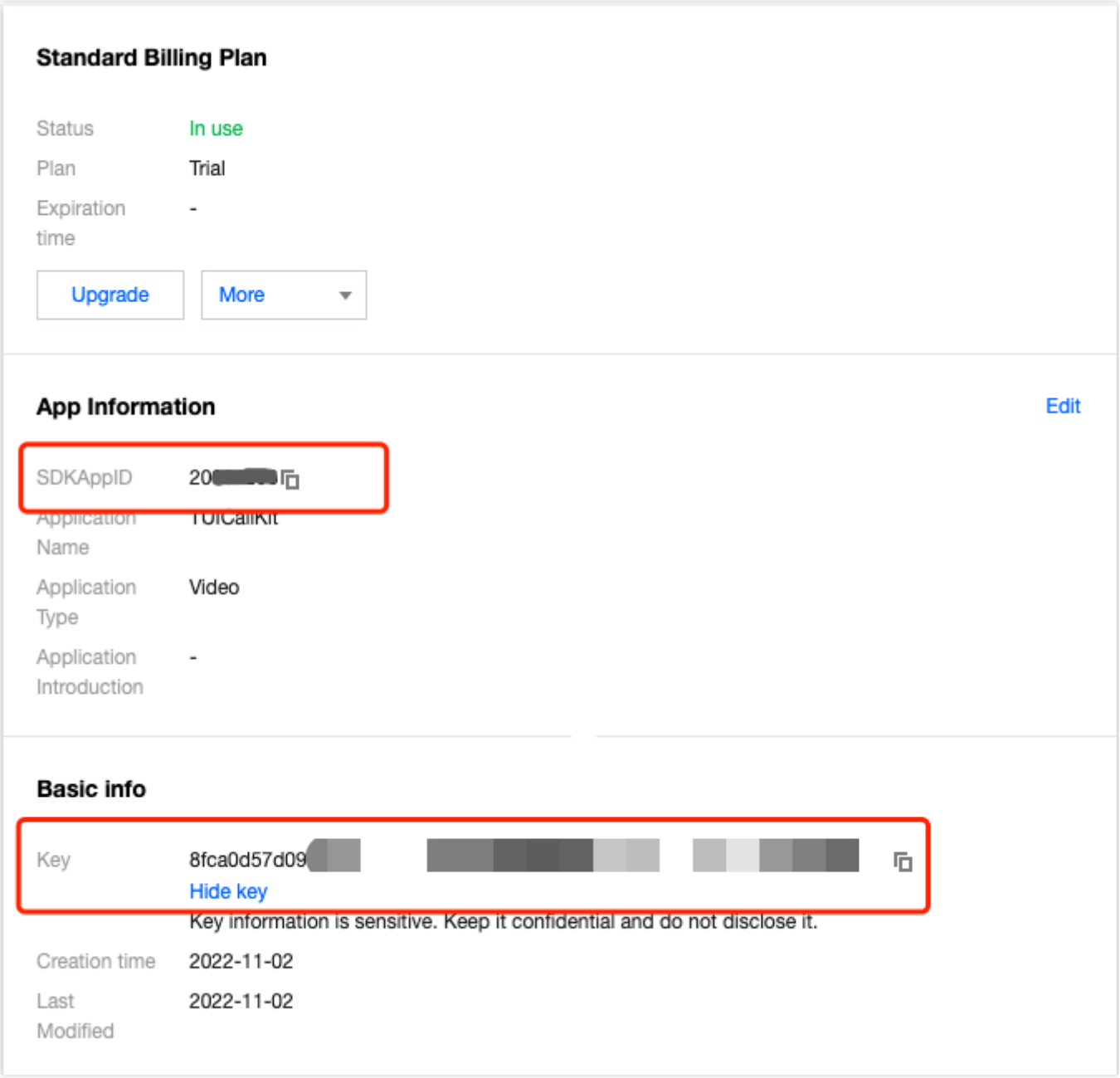
2. 作成したアプリケーションをクリックし、基本設定ページに進み、ページ右下隅のTRTCサービスのアクティブ化機能エリアで無料体験をクリックすると、TUICallKitの7日間無料トライアルサービスをアクティブ化することができます。正式アプリケーションのリリースが必要な場合は、追加購入をクリックすると購入ページに進むことができます。



ご注意：

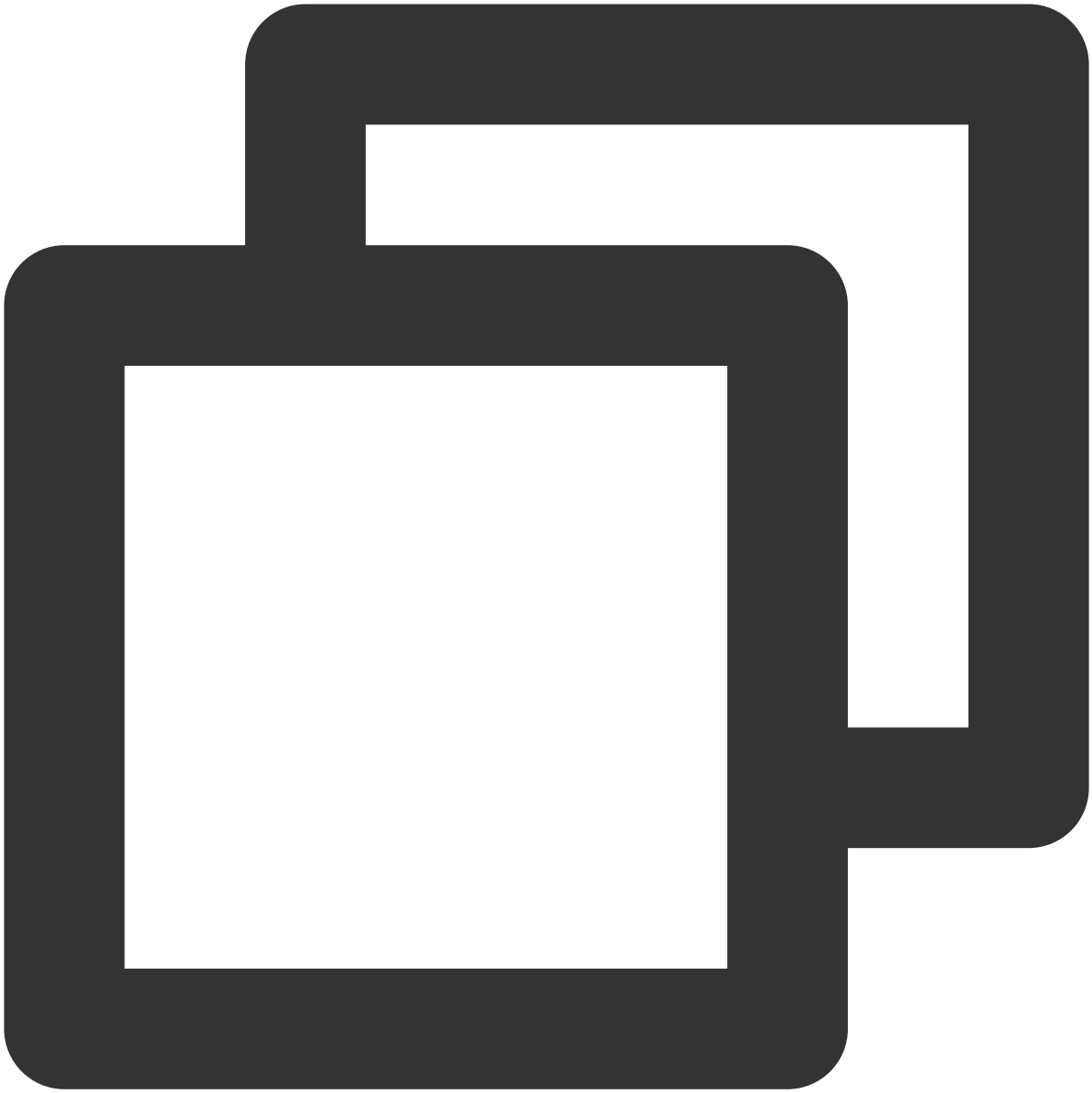
IMのオーディオビデオ通話機能は業務ニーズに応じて差別化した有料バージョンをご提供しています。含まれる機能をIM購入ページでご確認の上、ご自身に合ったバージョンを選択してご購入いただけます。

3. 同じページで**SDKAppID**と**キー(SecretKey)**を見つけて記録します。この2つはその後のステップ4：TUIコンポーネントへのログインで使用します。



説明：

お知らせ：**無料体験**をクリックすると、それまでにTRTCサービスを使用したことがある一部のユーザーには、次のような内容が表示されることがあります。

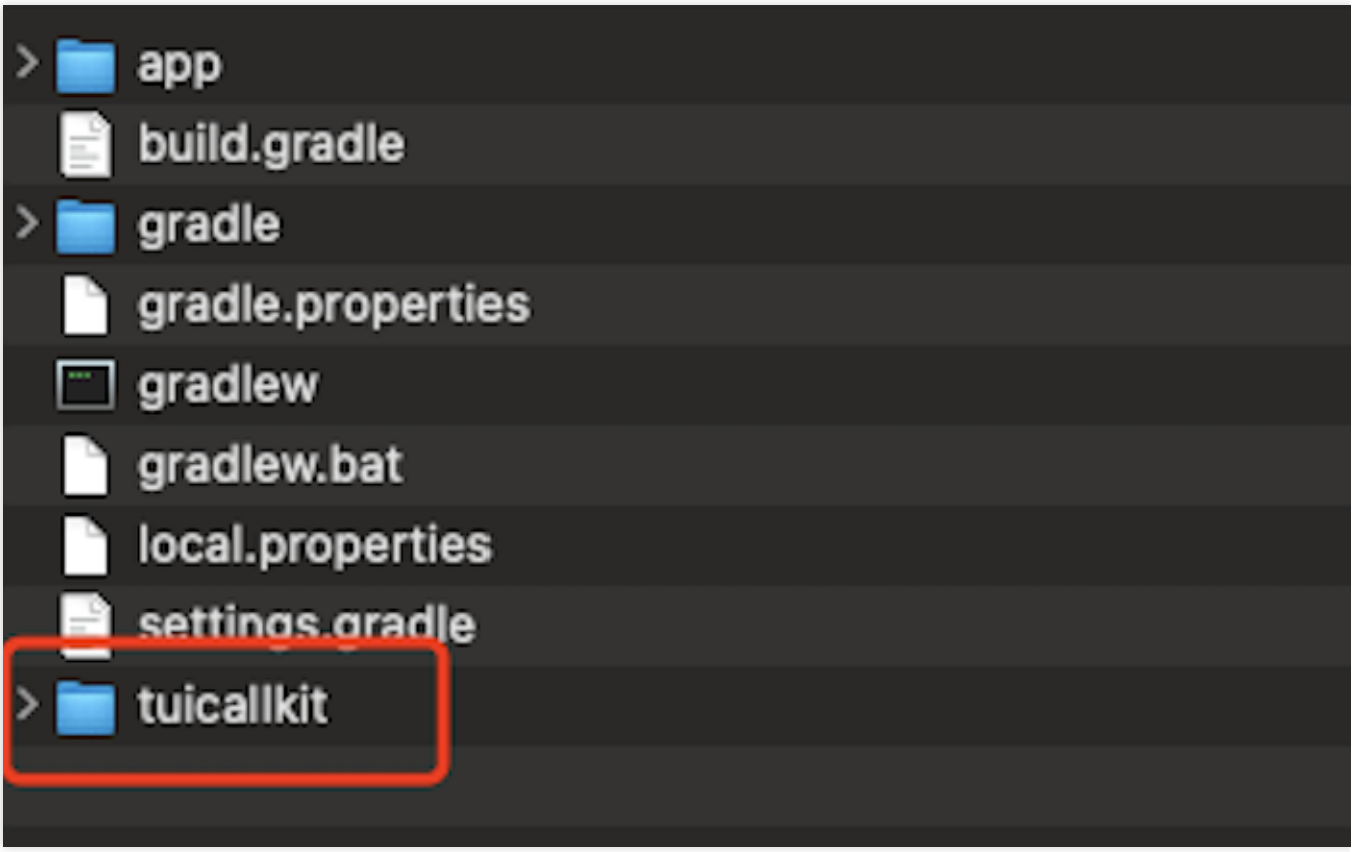


```
[-100013]:TRTC service is  suspended. Please check if the package balance is 0 or t
```

新しいIMオーディオビデオ通話機能は、Tencent CloudのTRTCとIMという2つの基本PaaSサービスを統合したもののため、TRTCの無料利用枠（10000分）が期限切れまたは使用済みの場合、このサービスをアクティブ化しようとすると失敗します。その場合は、TRTCコンソールをクリックし、SDKAppIDに対応するアプリケーション管理ページを見つけ、図のように後払い機能をアクティブ化してから**アプリケーションを有効にする**ことで、オーディオビデオ通話機能を正常に体験することができるようになります。

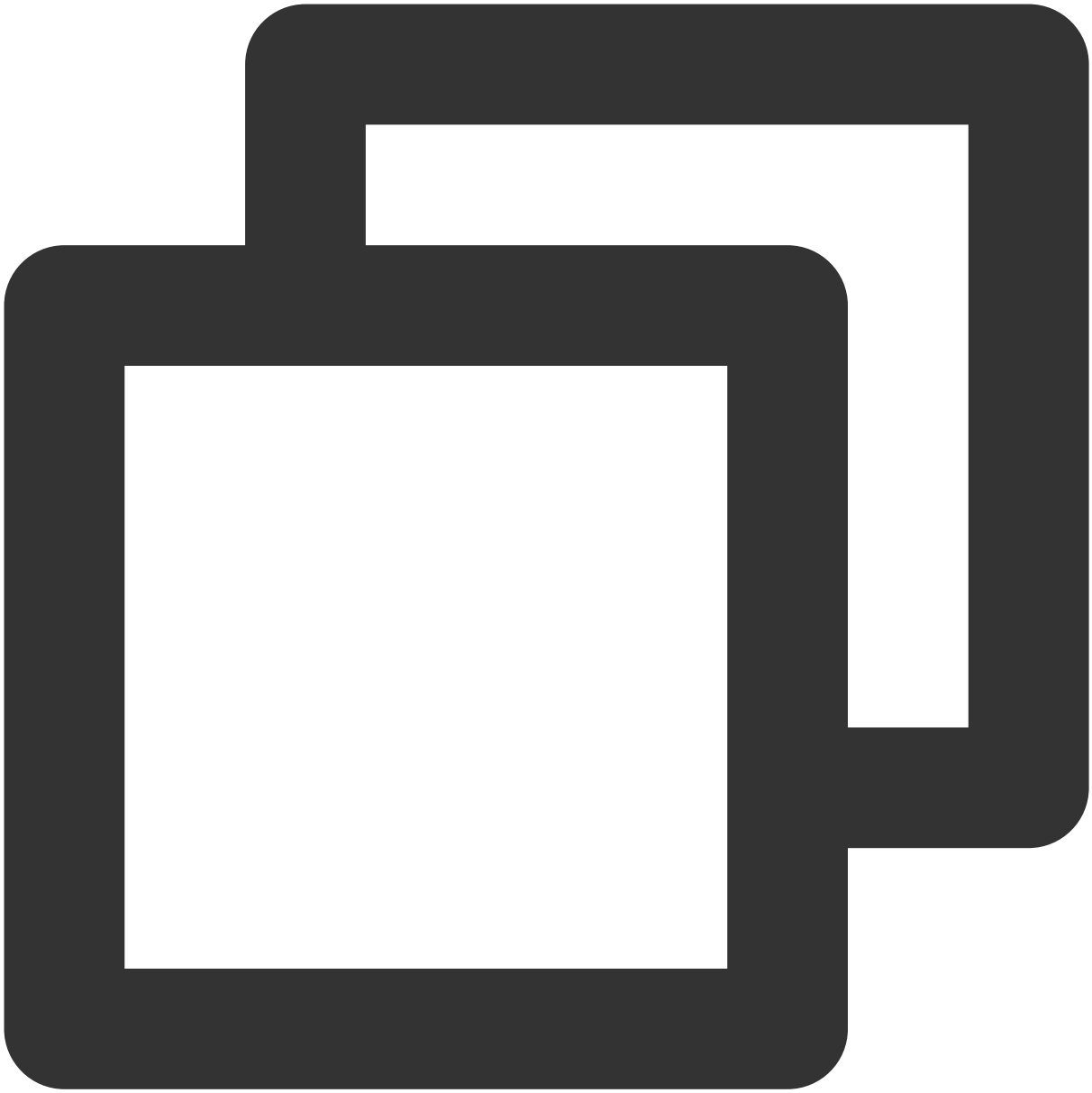
ステップ2：コンポーネントのダウンロードとインポート

Githubでコードをクローン/ダウンロードした後、下図のように、Androidディレクトリ下のtuicallkitサブディレクトリを現在のプロジェクトのappの同階層のディレクトリにコピーします。



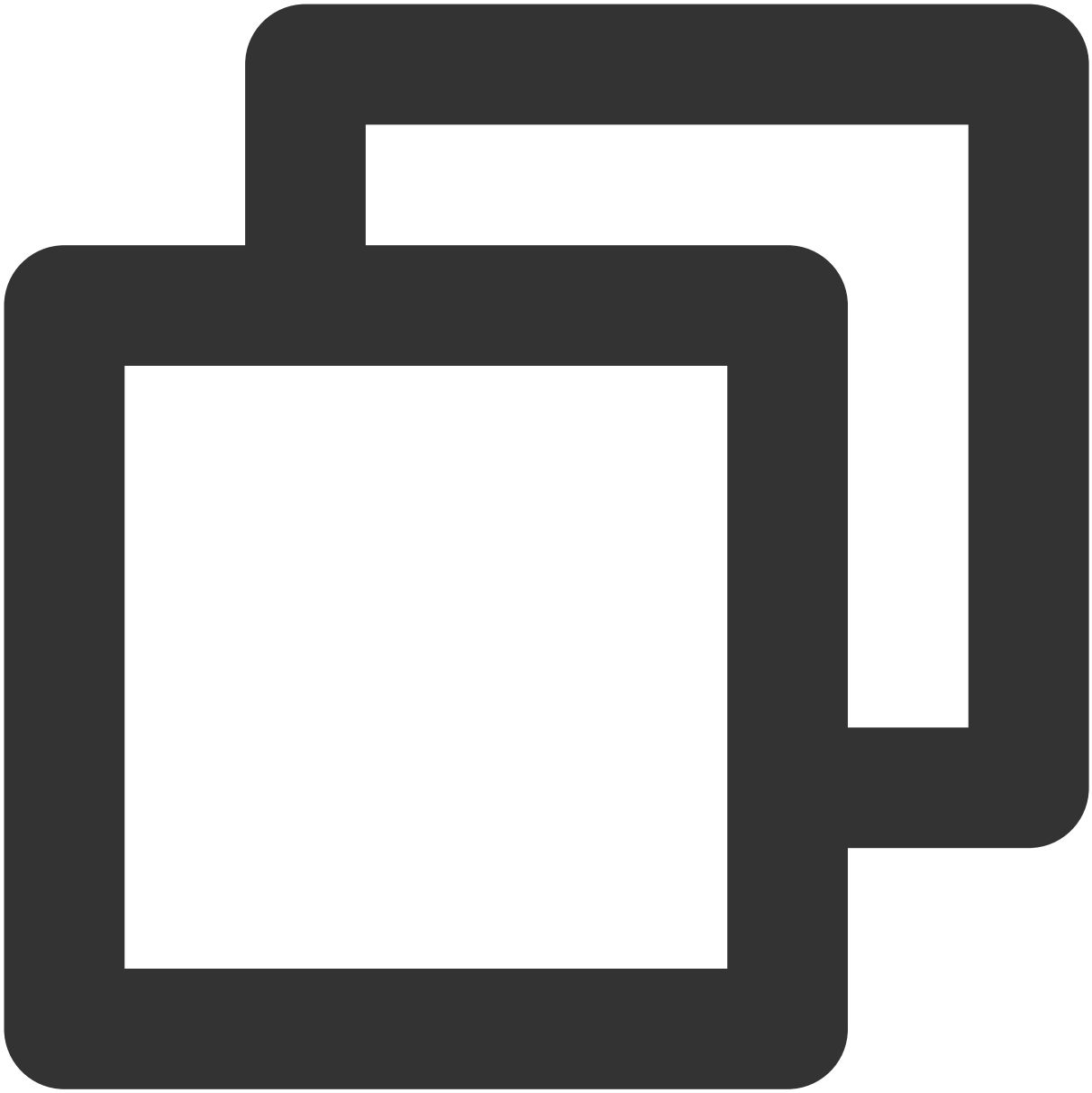
ステップ3：プロジェクト設定の完了

1. プロジェクトのルートディレクトリ下で `setting.gradle` ファイルを見つけ、その中に次のコードを追加します。その役割は、ステップ2でダウンロードしたtuicallkitコンポーネントを現在のプロジェクトにインポートするものです。



```
include ':tuicallkit'
```

2. appディレクトリ下で `build.gradle` ファイルを見つけ、その中に次のコードを追加します。その役割は、現在のappの新たに追加したtuicallkitコンポーネントへの依存を明確に述べるものです。

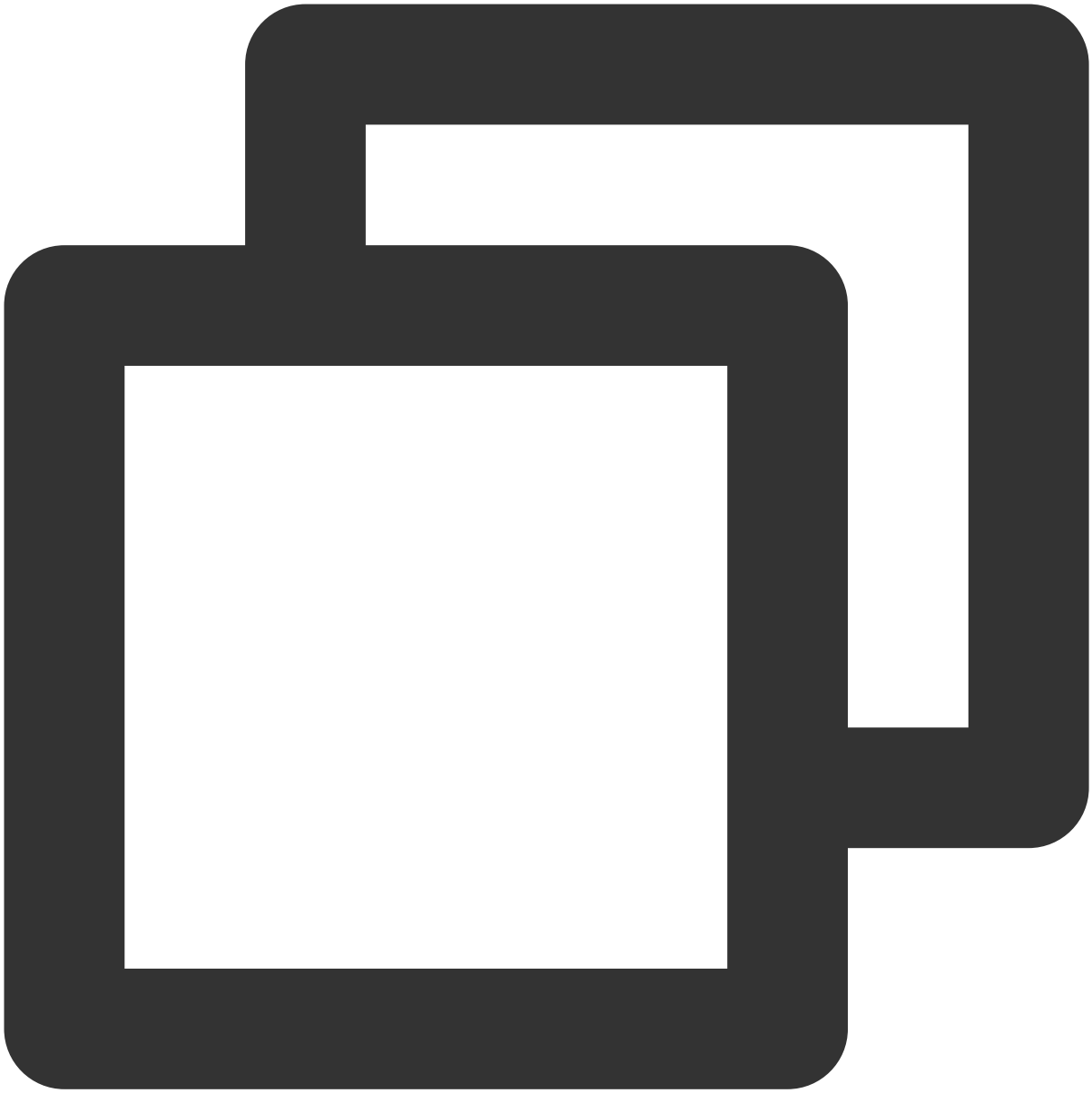


```
api project(':tuicallkit')
```

説明：

TUICallKitプロジェクトの内部はデフォルトで `TRTC SDK` 、 `IM SDK` 、 `tuicallengine` およびパブリックコーパスの `tuicore` に依存しており、開発者が単独で設定する必要はありません。バージョンアップが必要な場合は、 `tuicallkit/build.gradle` ファイルを変更するだけで済みます。

3. SDKの内部ではJavaのリフレクション機能を使用しているため、SDK内の一部のクラスに非難読化リストを追加する必要があります。このため、 `proguard-rules.pro` ファイルに次のコードを追加する必要があります。



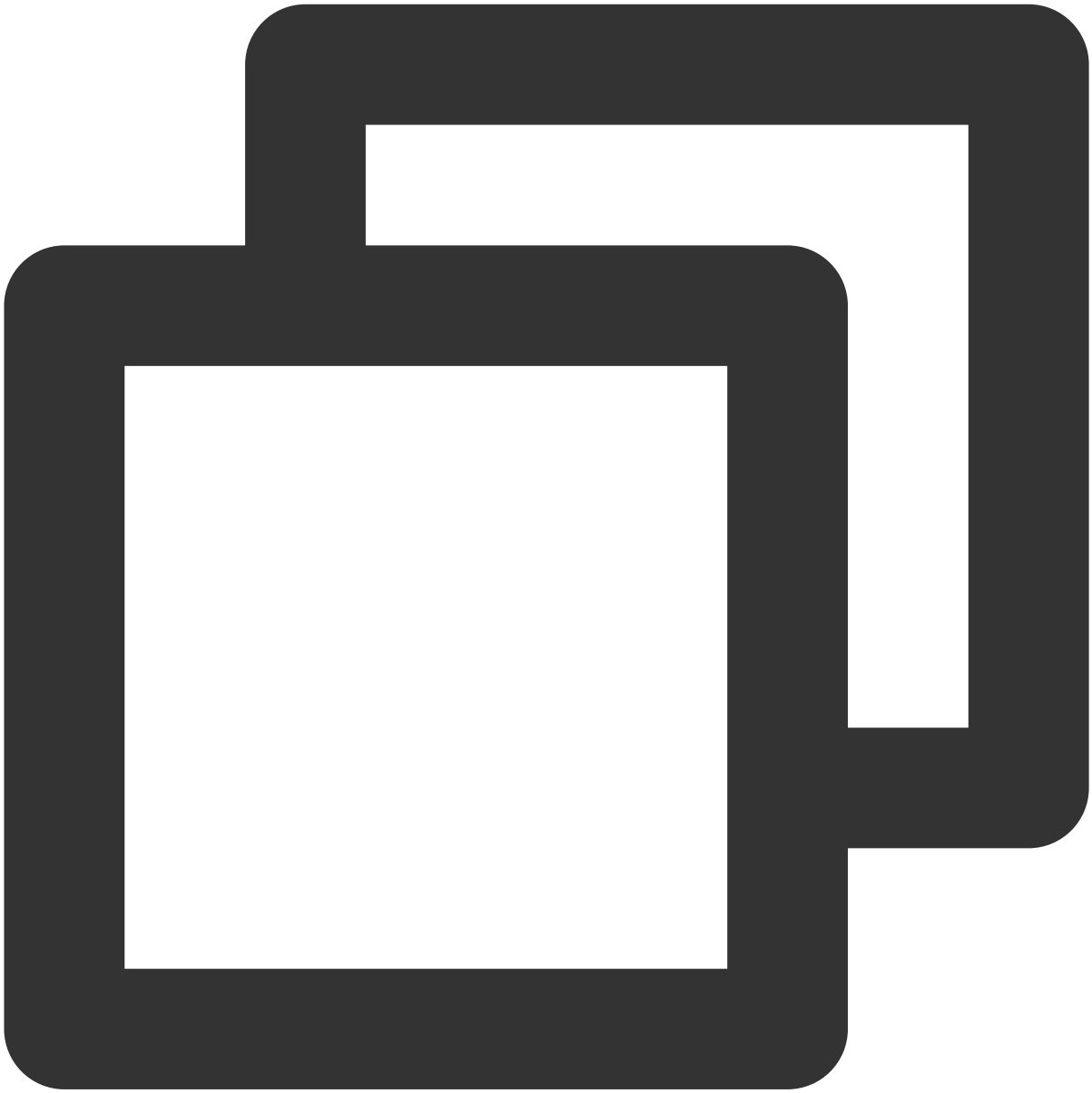
```
-keep class com.tencent.** { *; }
```

ご注意：

TUICallKitは内部で、カメラ、マイク、ストレージ読み取り権限などの動的申請を補助します。業務上、削減が必要な場合は `tuicallkit/src/main/AndroidManifest.xml` を変更することができます。

ステップ4：TUIコンポーネントへのログイン

プロジェクトに次のコードを追加します。この役割は、TUICore内の関連インターフェースを呼び出して、TUIコンポーネントへのログインを完了することです。ログインに成功しなければTUICallKitの各機能を正常に使用できないため、この手順に異常がないことが重要です。関連のパラメータが正しく設定されているかどうかを注意深く確認してください。



```
//ログイン結果に対するリスナーを設定します
private final TUILoginListener mLoginListener = new TUILoginListener() {
    @Override
    public void onKickedOffline() {
        super.onKickedOffline();
        Log.i(TAG, "You have been kicked off the line. Please login again!");
```

```
            //logout();
        }
        @Override
        public void onUserSigExpired() {
            super.onUserSigExpired();
            Log.i(TAG, "Your user signature information has expired");
            //logout();
        }
    };
    TUILogin.addLoginListener(mLoginListener);

    //ログイン
    TUILogin.login(context,
        1400000001,     // ステップ1で取得したSDKAppIDに置き換えてください
        "denny",        // ご自身のUserIDに置き換えてください
        "xxxxxxxxxxx",  // コンソールでUserSigを計算し、この位置に入力することができます
        new TUICallback() {
        @Override
        public void onSuccess() {
            Log.i(TAG, "login success");
        }

        @Override
        public void onError(int errorCode, String errorMessage) {
            Log.e(TAG, "login failed, errorCode: " + errorCode + " msg:" + errorMessage
        }
    });
```

パラメータの説明 ここではlogin関数内で使用するいくつかの重要パラメータについて詳しくご説明します。

SDKAppID：ステップ1の最後の段階で取得済みです。ここでは説明を省略します。

UserID：現在のユーザーIDです。文字列タイプでは、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：ステップ1の第3段階で取得したSecretKeyを使用してSDKAppID、userIDなどの情報を暗号化し、UserSigを取得します。これは認証用の証明書であり、現在のユーザーがTRTCサービスを使用できるかどうかをTencent Cloudが識別するために用いられます。コンソールの**支援ツール**で一時的に使用可能なUserSigを生成することができます。

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。

説明：

この手順は、現在開発者から最も多くのフィードバックが寄せられる手順でもあります。よくみられる問題には次のようなものがあります。

SDKAppIDの設定に誤りがある。国内サイトのSDKAppIDは一般的に140で始まる10桁の整数です。

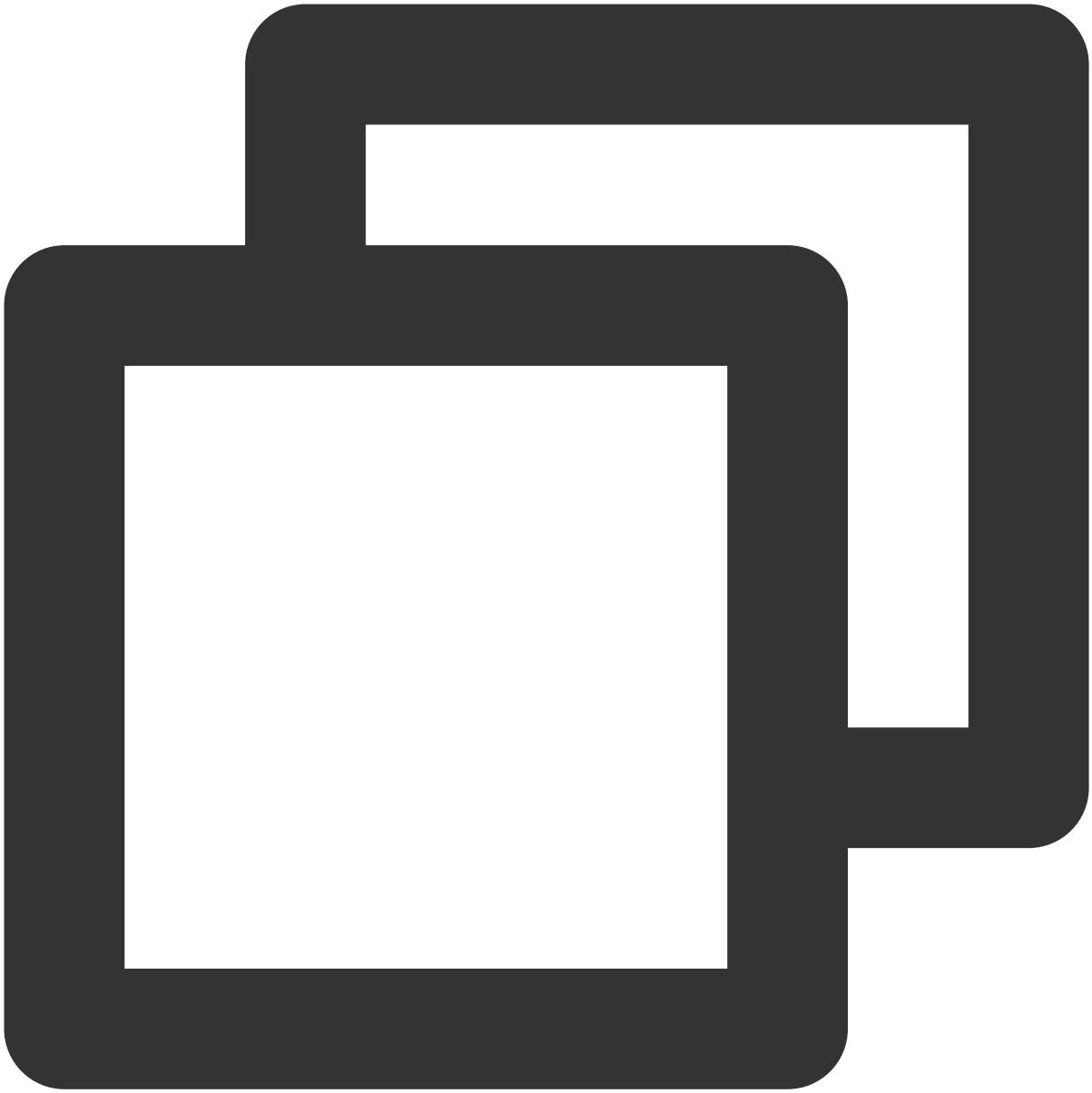
UserSigを誤って暗号化鍵（SecretKey）に設定している。UserSigはSecretKeyを使用して、SDKAppID、UserIDおよび有効期限などの情報を暗号化するためのものであり、SecretKeyは直接UserSigに設定するものではありま

せん。
UserIDが「1」、「123」、「111」などの簡単な文字列で設定されている。**TRTCは同一のUserIDによる複数端末へのログインをサポートしていない**ため、複数人によるコラボレーション開発の場合、「1」、「123」、「111」のようなUserIDは同僚に占有されていることが多く、ログイン失敗につながりやすいです。このため、デバッグの際に、識別度の高いUserIDを設定することをお勧めします。
GithubのサンプルコードでgenTestUserSig関数を使用してローカルでUserSigを計算すると、現在のアクセスフローをさらに速くスタートさせることができますが、この方法ではSecretKeyがAppのコード内で開示されてしまい、その後のアップグレードおよびSecretKeyの保護にとって不利益になります。そのため、UserSigの計算ロジックはサーバーに置いて実行し、またAppがTUICallKitコンポーネントを使用するたびに、リアルタイムに計算したUserSigをサーバーにリクエストするようにすることを強く推奨します。

ステップ5：通話する

1対1ビデオ通話

TUICallKitのcall関数を呼び出し、 通話タイプと着呼者のuserIdを指定すると、音声またはビデオ通話を開始することができます。

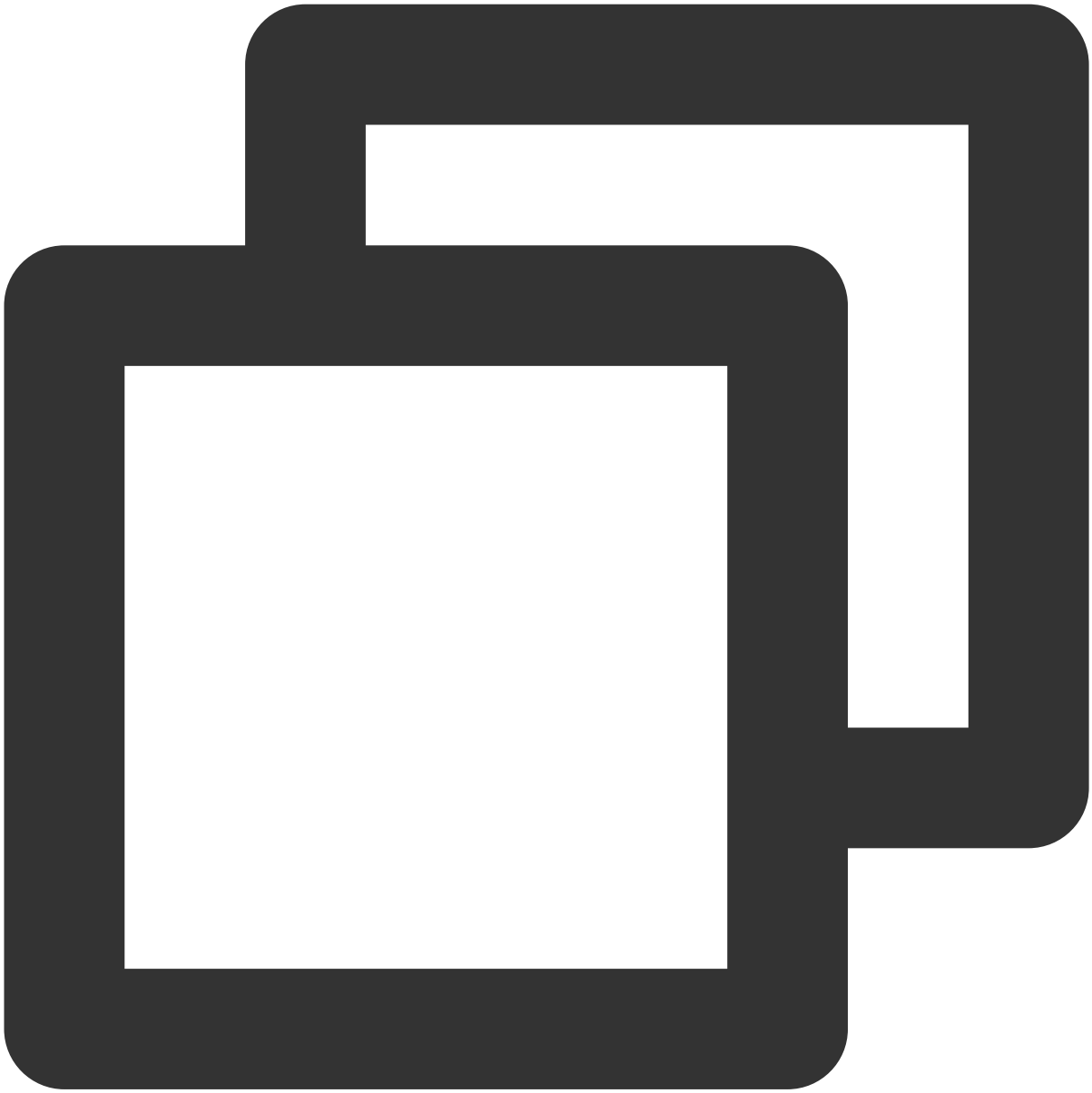


```
// 1対1ビデオ通話を開始します(UserIDはmikeとします)
TUICallKit.createInstance(context).call("mike", TUICallDefine.MediaType.Video);
```

パラメータ	タイプ	意味
userId	String	ターゲットユーザーのUserID： `"mike"`
callMediaType	TUICallDefine.MediaType	通話のメディアタイプ。 例： `TUICallDefine.MediaType.Video`

グループ内ビデオ通話

TUICallKitのgroupCall関数を呼び出し、 通話タイプと着呼者のUserIDリストを指定すると、グループ内で音声またはビデオ通話を開始することができます。



```
TUICallKit.createInstance(context).groupCall("12345678", Arrays.asList("jane", "mik
```

パラメータ	タイプ	意味
groupId	String	グループID。例： `"12345678"`

userIdList	List	ターゲットユーザーのUserIDリスト。例：`{"jane", "mike", "tommy"}`
callMediaType	TUICallDefine.MediaType	通話のメディアタイプ。 例：`TUICallDefine.MediaType.Video`

説明：
グループの作成の詳細についてはIMグループ管理をご参照ください。もしくはIM TUIKitを直接使用し、チャット、通話などのシナリオをワンストップで統合することもできます。
TUICallKitでは現在、グループ以外の多人数ビデオ通話はサポートしていません。もし必要な場合は、colleenyu@tencent.comまでお問い合わせください。

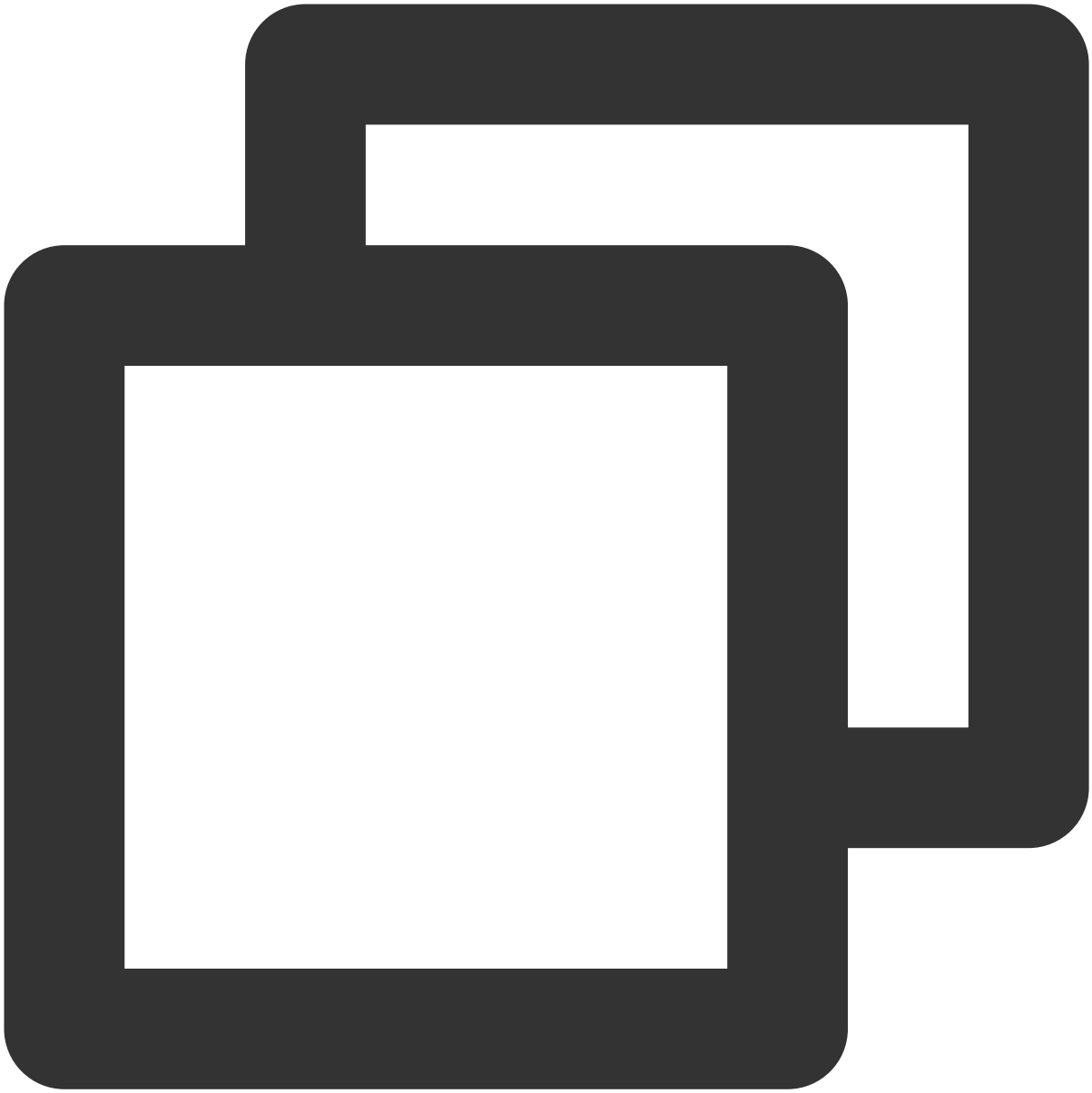
ステップ6：通話に応答する

通話リクエストを受信した後、TUICallKitコンポーネントは着信を通知する応答画面を自動的に呼び出しますが、Androidシステムの権限上の理由により、状況は次のいくつかに分かれます。
Appがフロントエンドにある場合は、招待を受信した際に呼び出し画面が自動的にポップアップし、着信音が再生されます。
Appがバックエンドにあっても、`フローティングウィンドウ権限`または`バックエンドポップアップアプリケーション`などの権限が承認されている場合は、呼び出し画面が自動的にポップアップし、着信音が再生されます。
Appがバックエンドにあり、なおかつ`フローティングウィンドウ権限`または`バックエンドポップアップアプリケーション`などの権限が承認されていない場合は、TUICallKitが着信音を再生し、ユーザーに応答または終了を促します。
Appのプロセスがすでに破棄されている場合は、オフライン通知（Android）にアクセスし、通知欄のメッセージによってユーザーに応答または終了を促すのみとなります。

ステップ7：その他の特徴

1、ニックネーム&プロフィール画像の設定

カスタムニックネームまたはプロフィール画像を設定したい場合は、次のインターフェースを使用して更新できます。



```
TUICallKit.createInstance(context).setSelfInfo("jack", "https:/****/user_avatar.png
```

ご注意：

ユーザーのプライバシー上の制限により、フレンド以外との通話では、呼ばれるニックネームとプロフィール画像の更新が遅れる可能性があります。一度通話が成功するとスムーズに更新されるようになります。

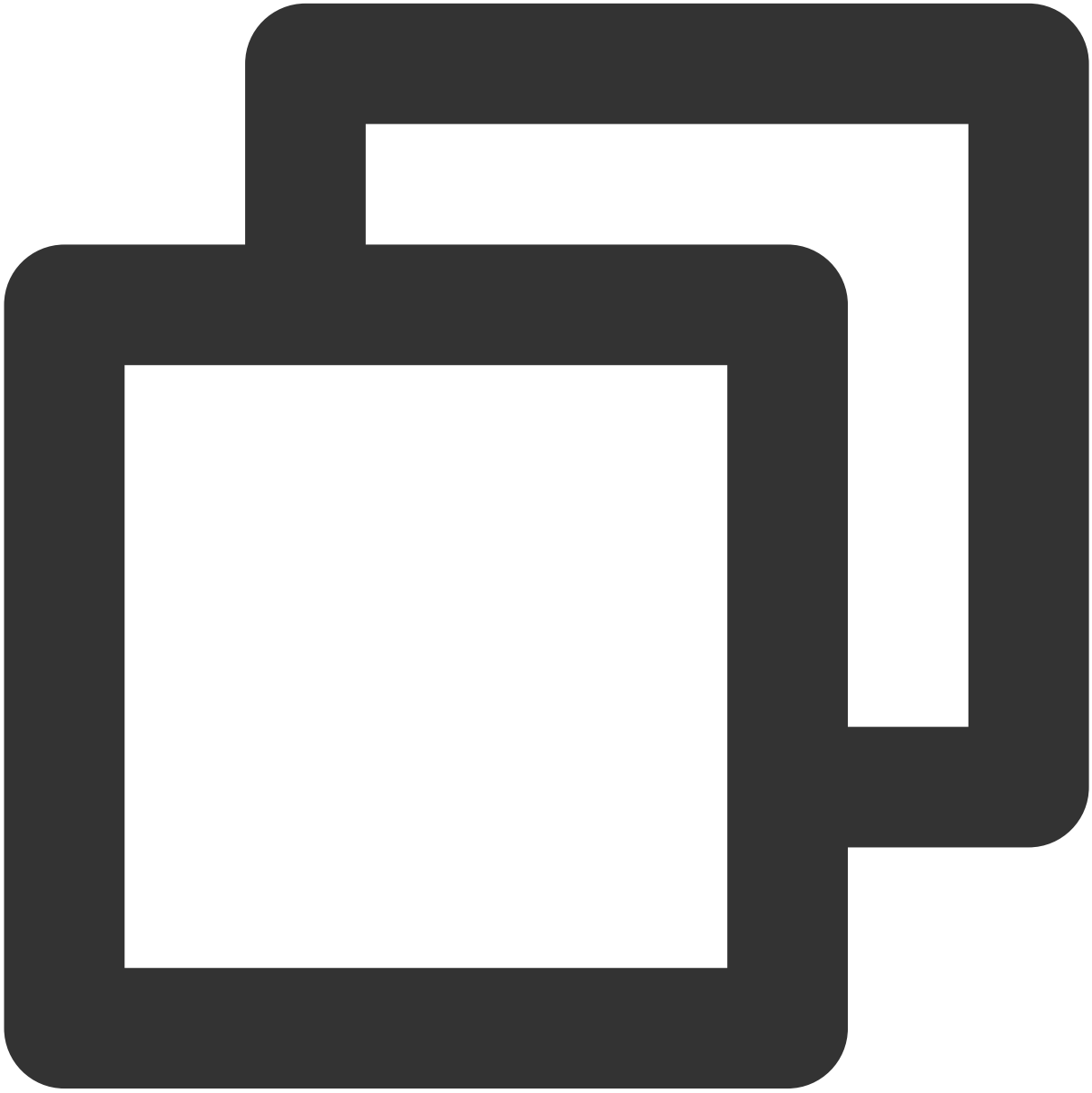
2、オフライン通知

上記の手順が完了すると、オーディオビデオ通話の発信と接続が実現できますが、業務上「アプリケーションのプロセスが強制終了された後」または「アプリケーションがバックエンドに戻った後」もオーディオビデオ通話

リクエストを正常に受信できる必要がある場合は、オフライン通知機能を追加する必要があります。詳細についてはオフライン通知（Android）をご参照ください。

3、フローティングウィンドウ機能

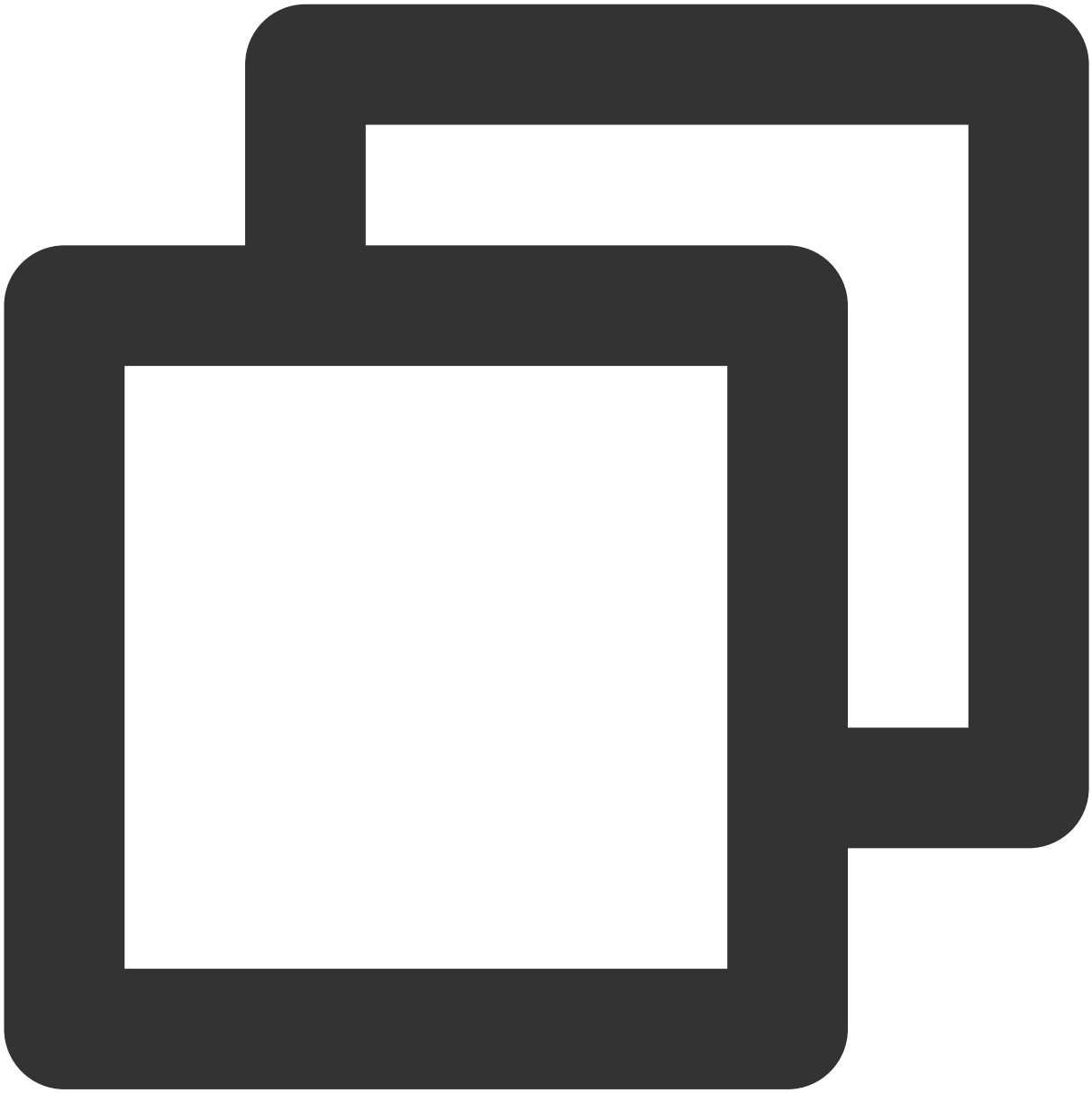
業務上、フローティングウィンドウ機能を有効にする必要がある場合は、TUICallKitコンポーネントの初期化の際に次のインターフェースを呼び出してこの機能を有効化することができます。



```
TUICallKit.createInstance(context).enableFloatWindow(true);
```

4、通話ステータスの監視

業務上、通話の開始、終了、通話中のネットワーク品質などの**通話ステータスの監視**が必要な場合は、次のイベントを監視することができます。

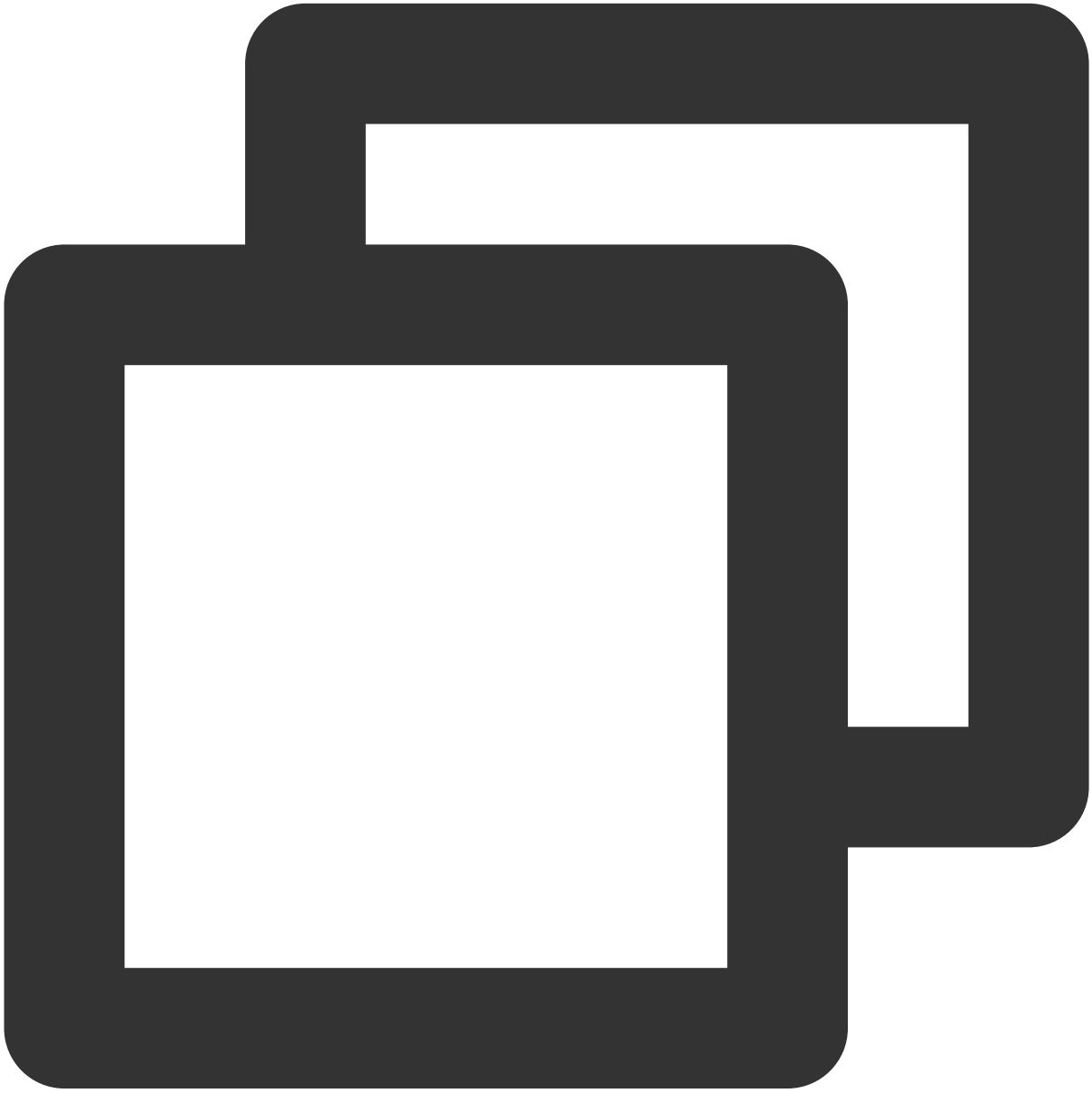


```
TUICallEngine.createInstance(context).addObserver(new TUICallObserver() {
    @Override
    public void onCallBegin(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType
    }
    public void onCallEnd(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType ca
    }
    public void onUserNetworkQualityChanged(List<TUICommonDefine.NetworkQualityInfo
```

```
    }
});
```

5、カスタム着信音

着信音をカスタマイズしたい場合は、次のインターフェースによって設定できます。



```
TUICallKit.createInstance(context).setCallingBell(filePath);
```

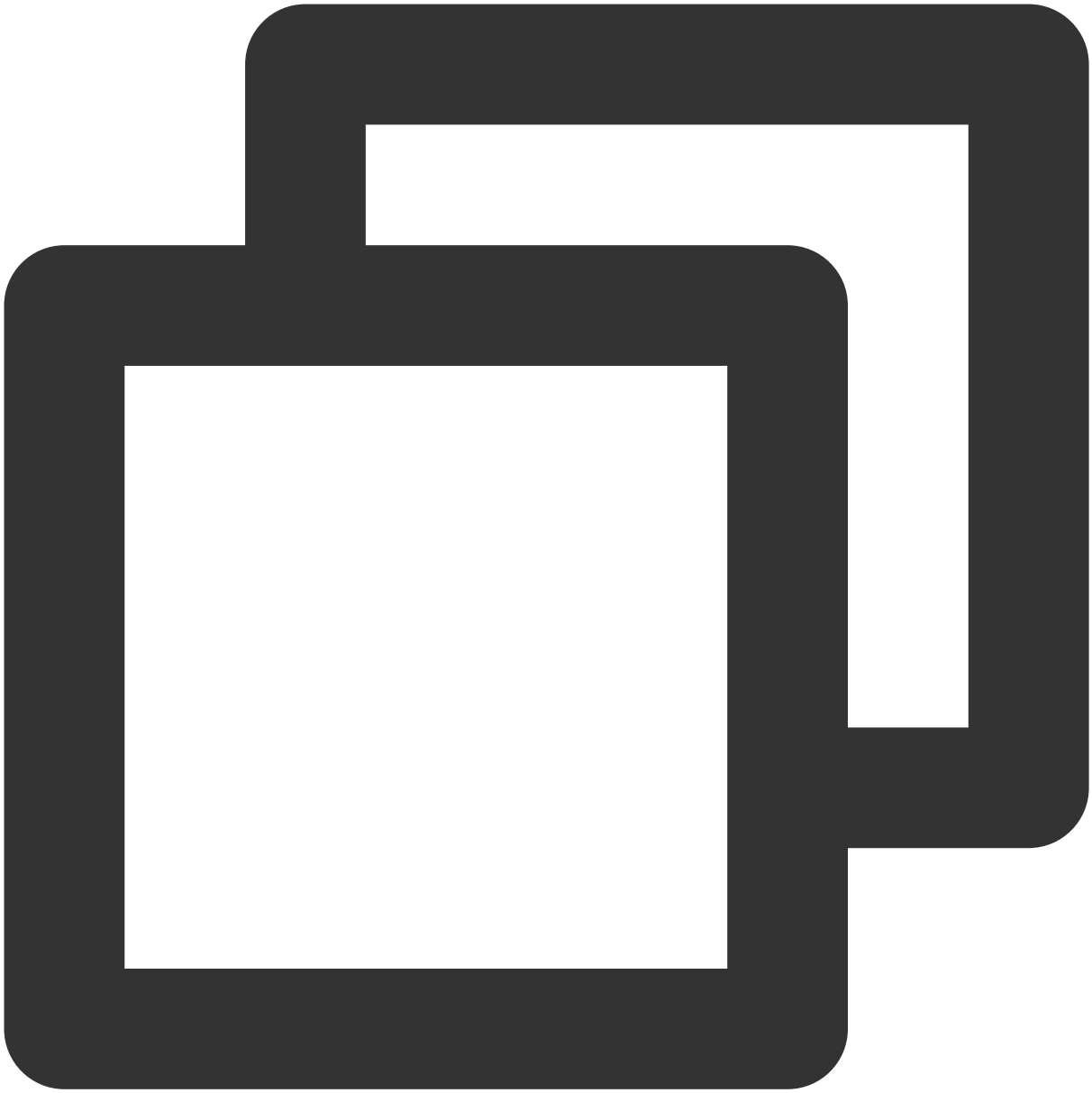
よくあるご質問

1、「The package you purchased does not support this ability」というエラーが表示されました。

上記のエラーが表示された場合は、現在のアプリケーションのオーディオビデオ通話機能パッケージが期限切れまたはアクティブ化されていないことを表します。ステップ1：サービスのアクティブ化を参照してオーディオビデオ通話機能を取得またはアクティブ化し、引き続きTUICallKitコンポーネントをご利用ください。

2、TUICallKitがクラッシュし、クラッシュログが"No implementation found for xxxx"となっています。

スタック情報は次のとおりです。



```
java.lang.UnsatisfiedLinkError: No implementation found for void com.tencent.liteav
       at com.tencent.liteav.base.Log.nativeWriteLogToNative(Native Method)
       at com.tencent.liteav.base.Log.i(SourceFile:177)
       at com.tencent.liteav.basic.log.TXCLog.i(SourceFile:36)
       at com.tencent.liteav.trtccalling.model.impl.base.TRTCLogger.i(TRTCLogger.ja
       at com.tencent.liteav.trtccalling.model.impl.ServiceInitializer.init(Service
       at com.tencent.liteav.trtccalling.model.impl.ServiceInitializer.onCreate(Ser
       at android.content.ContentProvider.attachInfo(ContentProvider.java:2097)
       at android.content.ContentProvider.attachInfo(ContentProvider.java:2070)
       at android.app.ActivityThread.installProvider(ActivityThread.java:8168)
       at android.app.ActivityThread.installContentProviders(ActivityThread.java:77
```

```
        at android.app.ActivityThread.handleBindApplication(ActivityThread.java:7573
        at android.app.ActivityThread.access$2600(ActivityThread.java:260)
        at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2435)
        at android.os.Handler.dispatchMessage(Handler.java:110)
        at android.os.Looper.loop(Looper.java:219)
        at android.app.ActivityThread.main(ActivityThread.java:8668)
        at java.lang.reflect.Method.invoke(Native Method)
        at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.j
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:1109)
```

TUICallkitは仮想マシンをサポートしていませんので、実機でご体験ください。実機で上記の異常が生じる原因は、TUICallKitの依存するTRTCなどのSDKの一部インターフェースにJNIを使用しているため、Android Studioがいくつかの条件でAPKコンパイルを行う際にNativeの.soライブラリをパッケージ化できず、このようなエラーが発生します。再Cleanを行ってください。

説明：

その他のヘルプ情報についての詳細は、Androidについてのよくあるご質問をご参照ください。

ご意見とフィードバック

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。

iOS

最終更新日：：2024-07-19 14:53:21

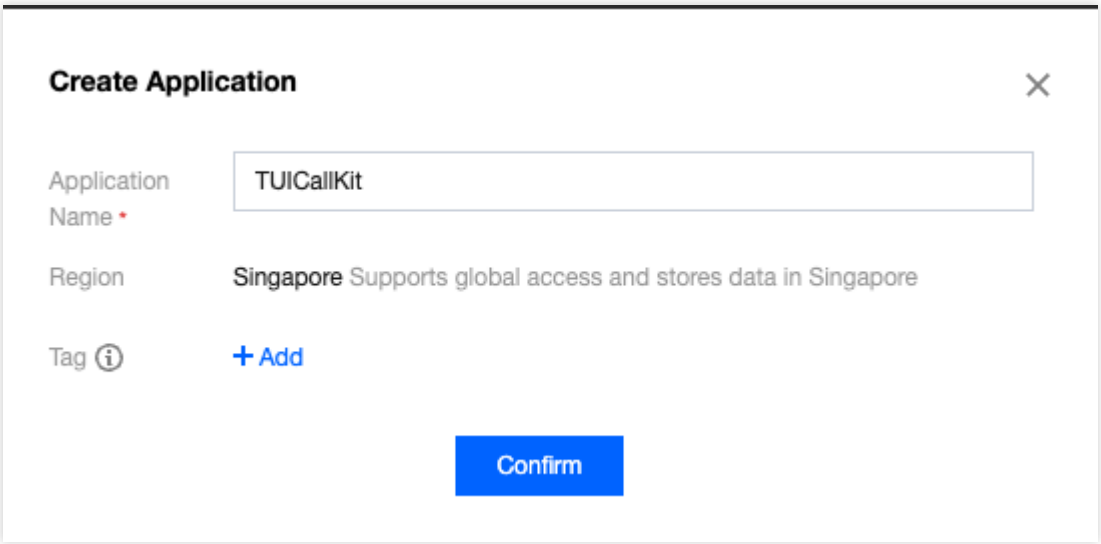
ここでは、最短の時間でTUICallKitコンポーネントの統合を完了する方法についてご説明します。このドキュメントに沿って操作することで、1時間以内に次の重要手順を完了し、最終的にUIインターフェースを完備したビデオ通話機能を手にすることができます。

環境の準備

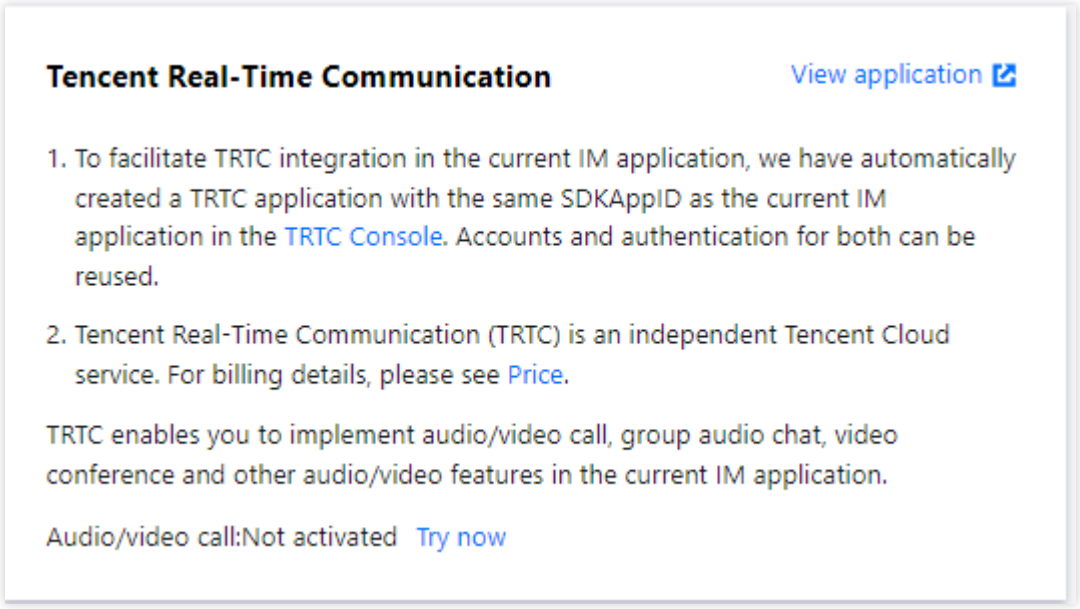
iOS 9.0 (API level 16)およびそれ以上。

ステップ1：サービスのアクティブ化

TUICallKitはTencent CloudのIMと、TRTCという2つの有料PaaSサービスをベースに構築したオーディオビデオ通信コンポーネントです。以下の手順で関連のサービスをアクティブ化し、60日間の無料トライアルサービスを体験することができます。
1. IMコンソールにログインし、**新しいアプリケーションの作成**をクリックし、ポップアップしたダイアログボックスにアプリケーション名を入力して**OK**をクリックします。



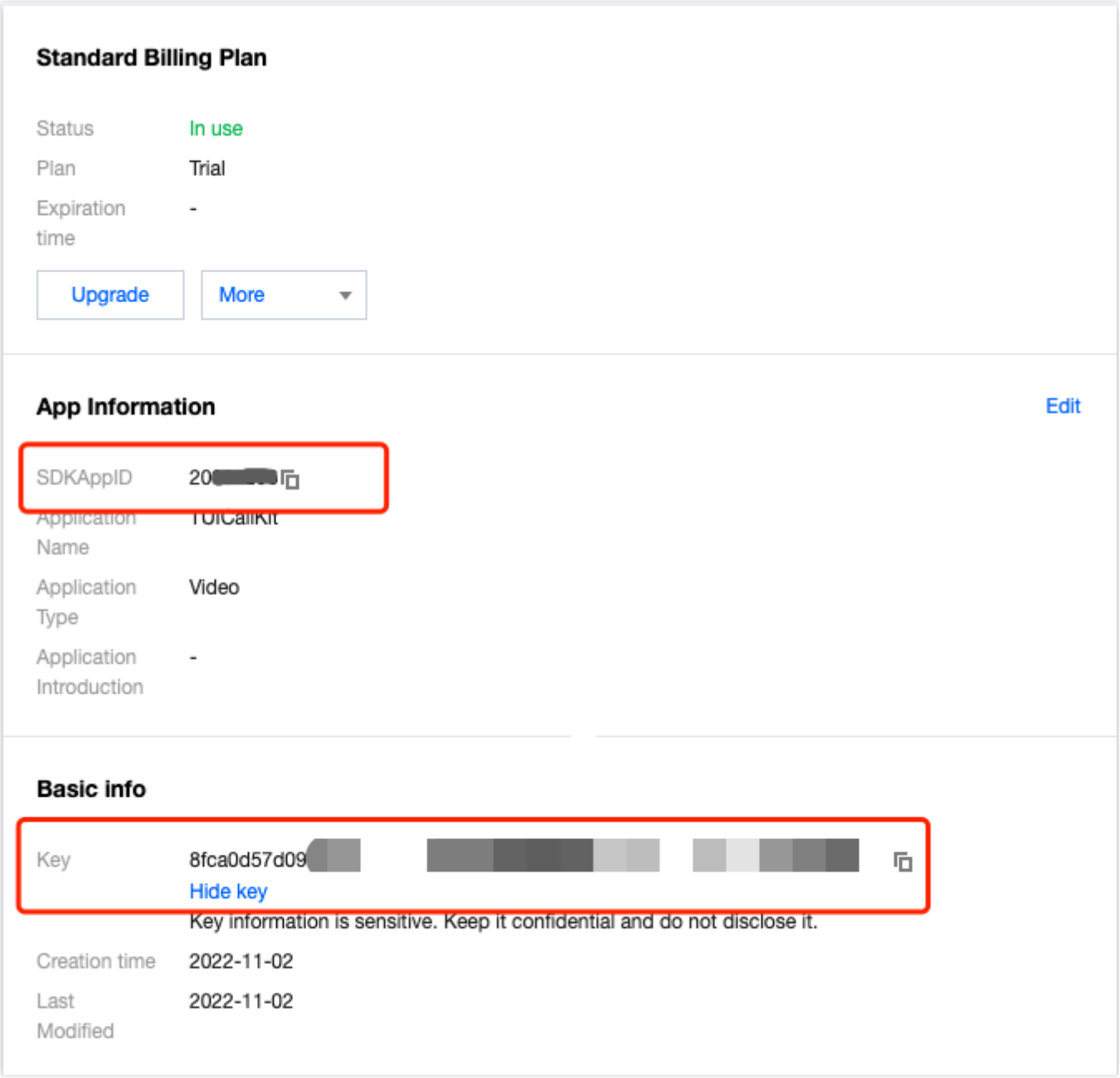
2. 作成したアプリケーションをクリックし、基本設定ページに進み、ページ右下隅のTRTCサービスのアクティブ化機能エリアで無料体験をクリックすると、TUICallKitの7日間無料トライアルサービスをアクティブ化することができます。正式アプリケーションのリリースが必要な場合は、お問い合わせください。



ご注意：

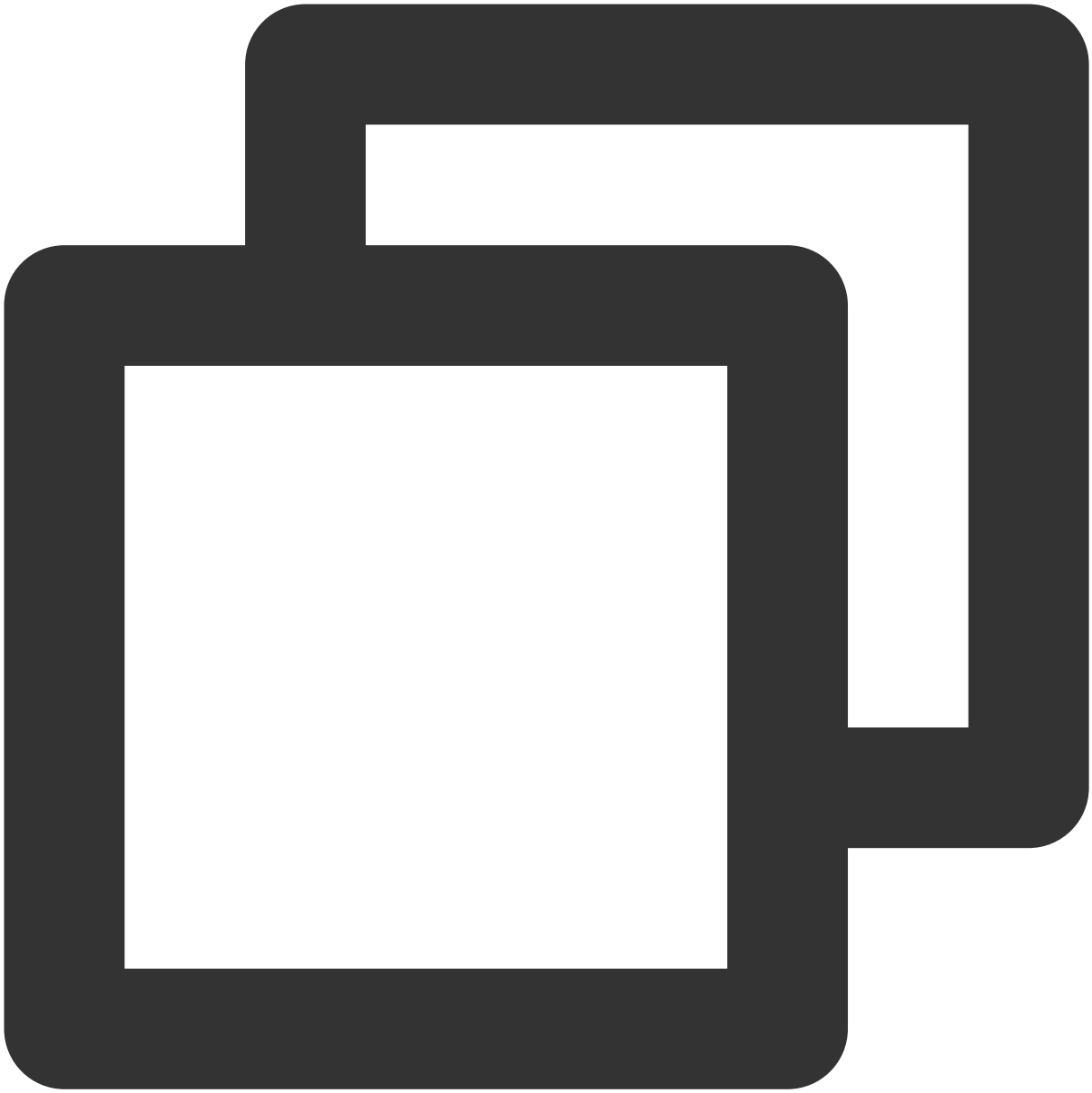
IMのオーディオビデオ通話機能は業務ニーズに応じて差別化した有料バージョンをご提供しています。お問い合わせいただくことで、含まれる機能をご確認の上、ご自身に合ったバージョンを選択してご購入いただけます。

3. 同じページで**SDKAppID**と**キー(SecretKey)**を見つけて記録します。この2つはその後のステップ4：TUIコンポーネントへのログインで使用します。



説明：

お知らせ：**無料体験**をクリックすると、それまでにTRTCサービスを使用したことがある一部のユーザーには、次のような内容が表示されることがあります。



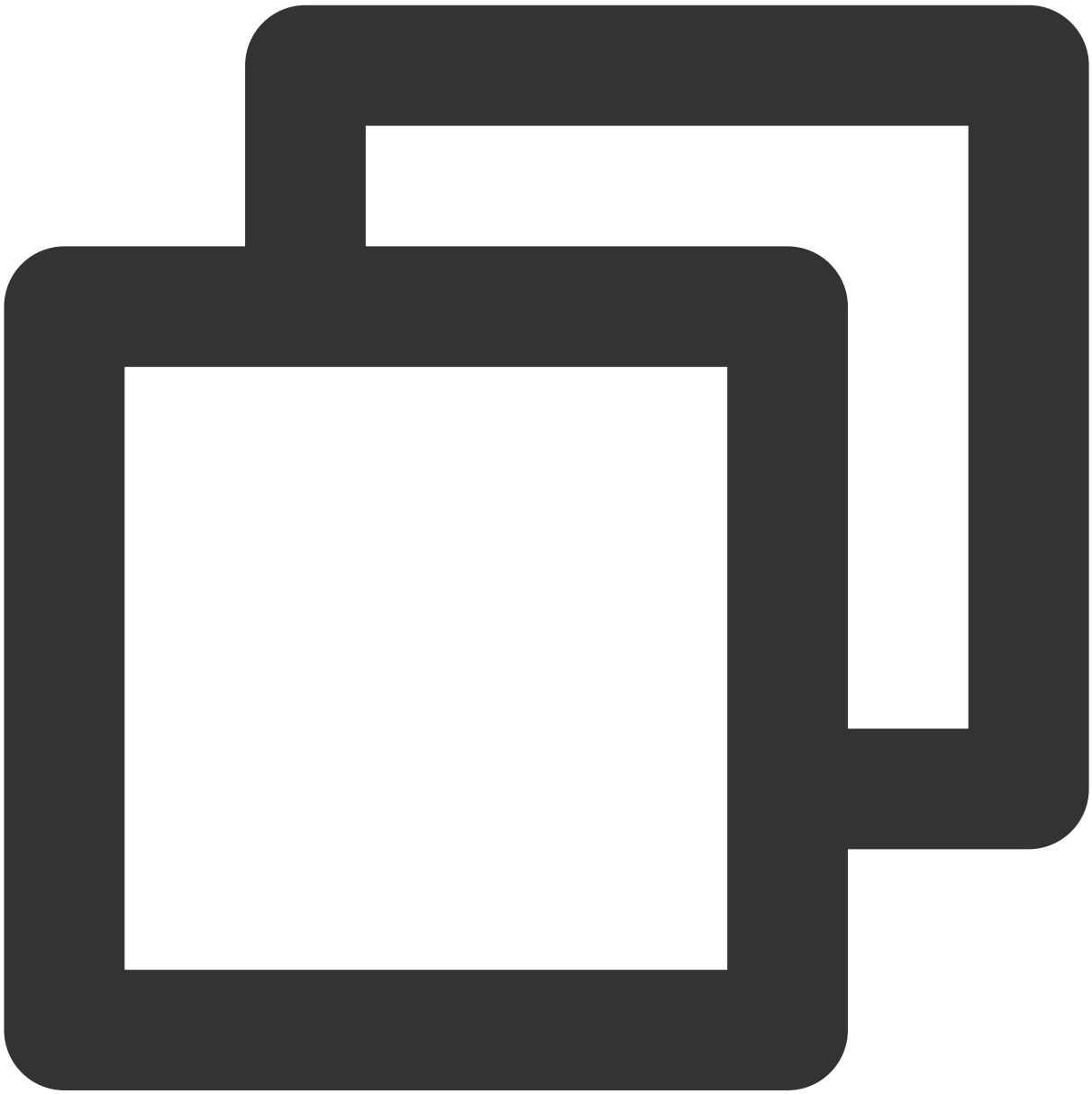
```
[-100013]:TRTC service is  suspended. Please check if the package balance is 0 or t
```

新しいIMオーディオビデオ通話機能は、Tencent CloudのTRTCとIMという2つの基本PaaSサービスを統合したもののため、TRTCの無料利用枠（10000分）が期限切れまたは使用済みの場合、このサービスをアクティブ化しようとすると失敗します。その場合は、TRTCコンソールをクリックし、SDKAppIDに対応するアプリケーション管理ページを見つけ、図のように後払い機能をアクティブ化してから**アプリケーションを有効にする**ことで、オーディオビデオ通話機能を正常に体験することができるようになります。

ステップ2：コンポーネントのインポート

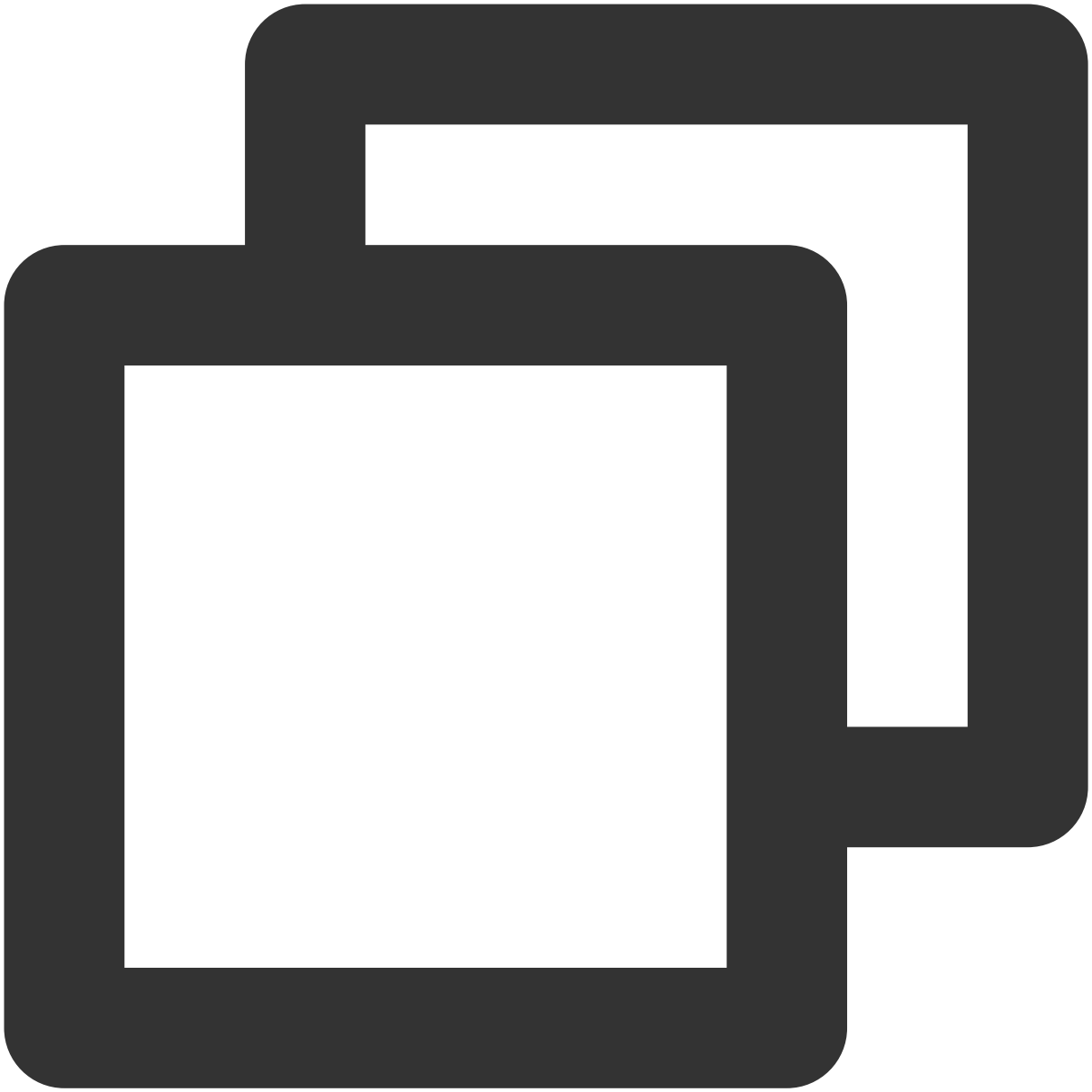
CocoaPodsを使用してコンポーネントをインポートします。具体的な手順は次のとおりです。

1. `Podfile` ファイルに次の依存を追加します。



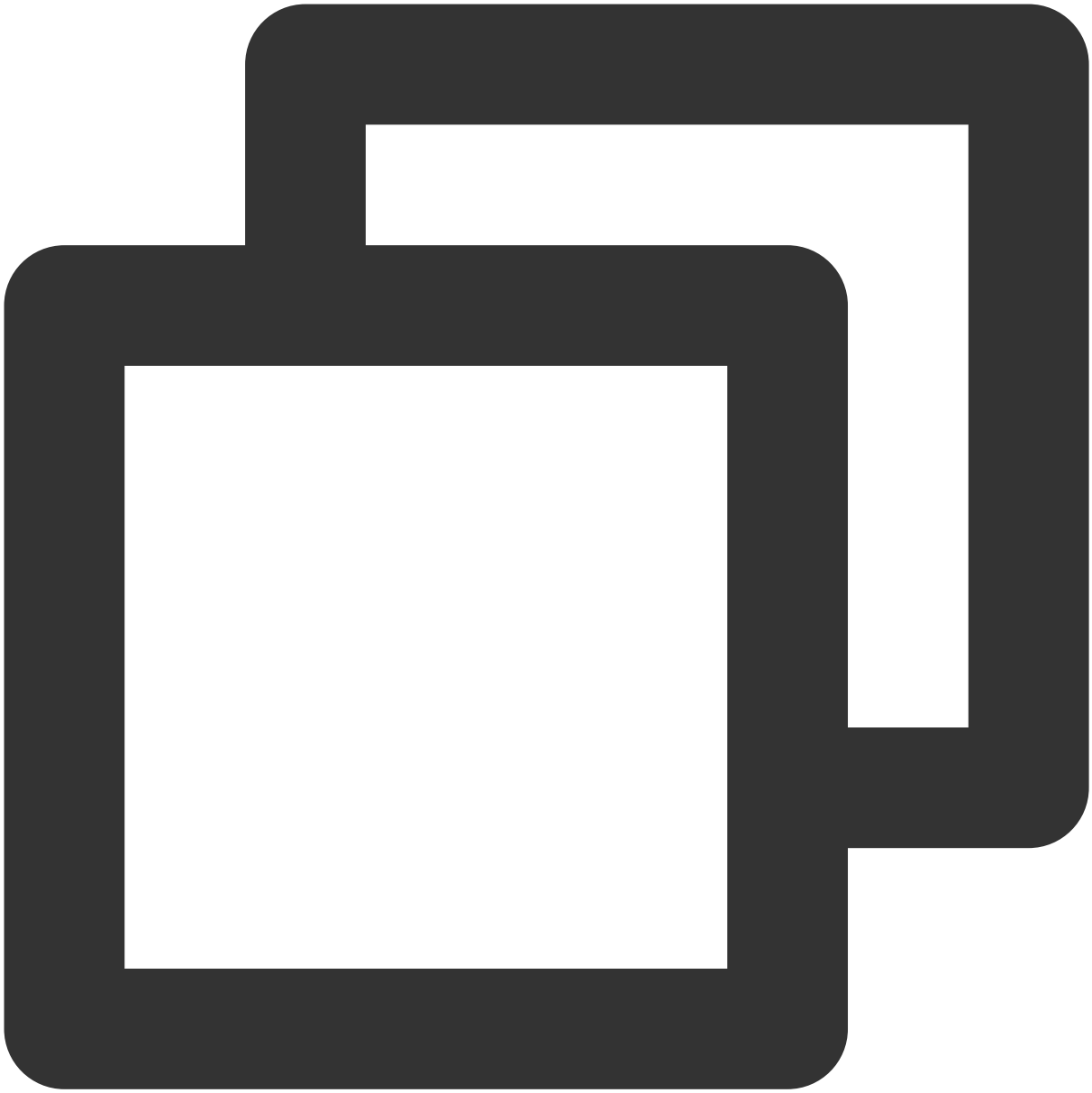
```
pod 'TUICallKit'
```

2. 次のコマンドを実行し、コンポーネントをインストールします。



```
pod install
```

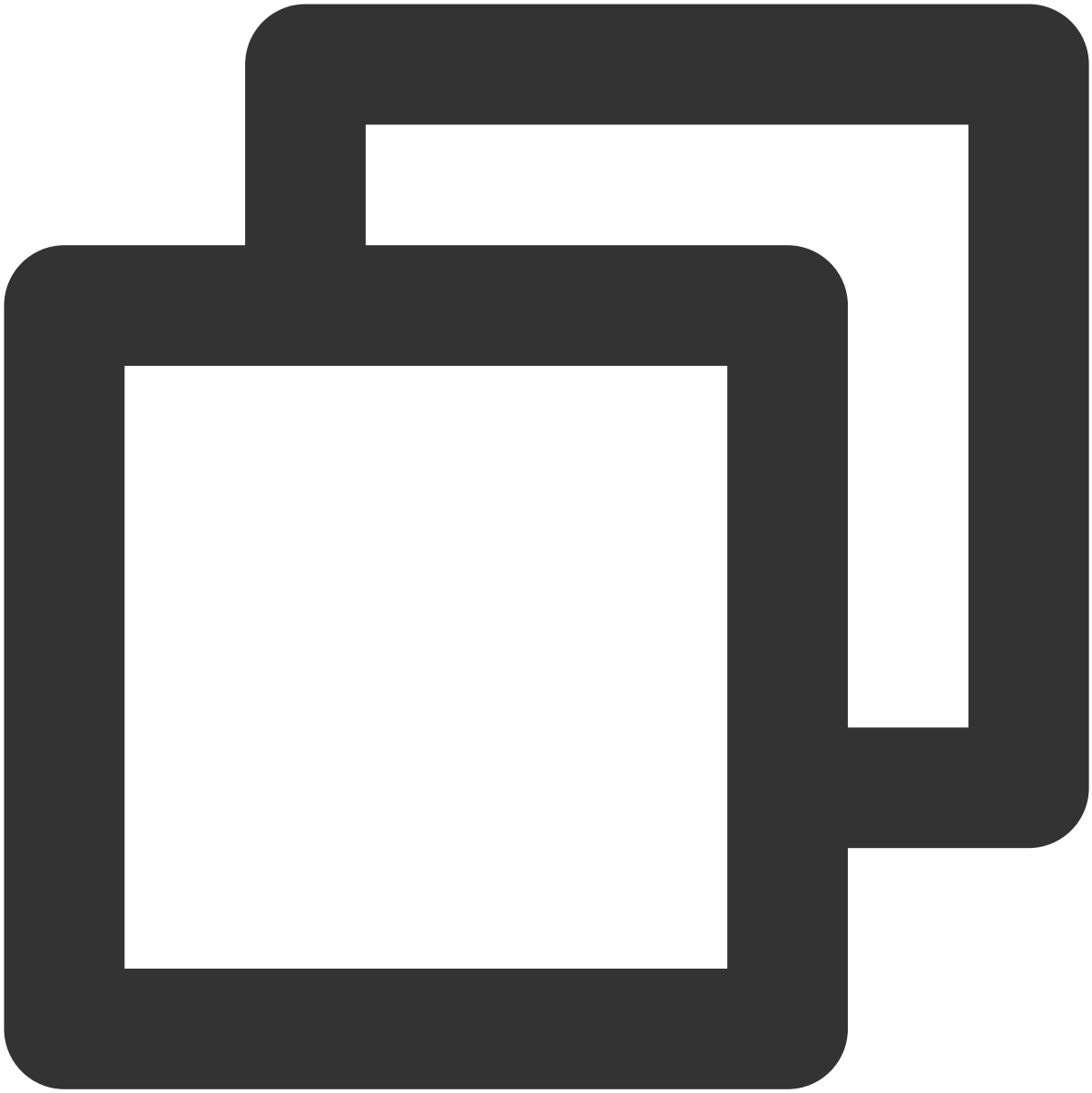
TUICallKitの最新バージョンをインストールできない場合は、次のコマンドを実行して、ローカルのCocoaPodsリポジトリリストを更新します。



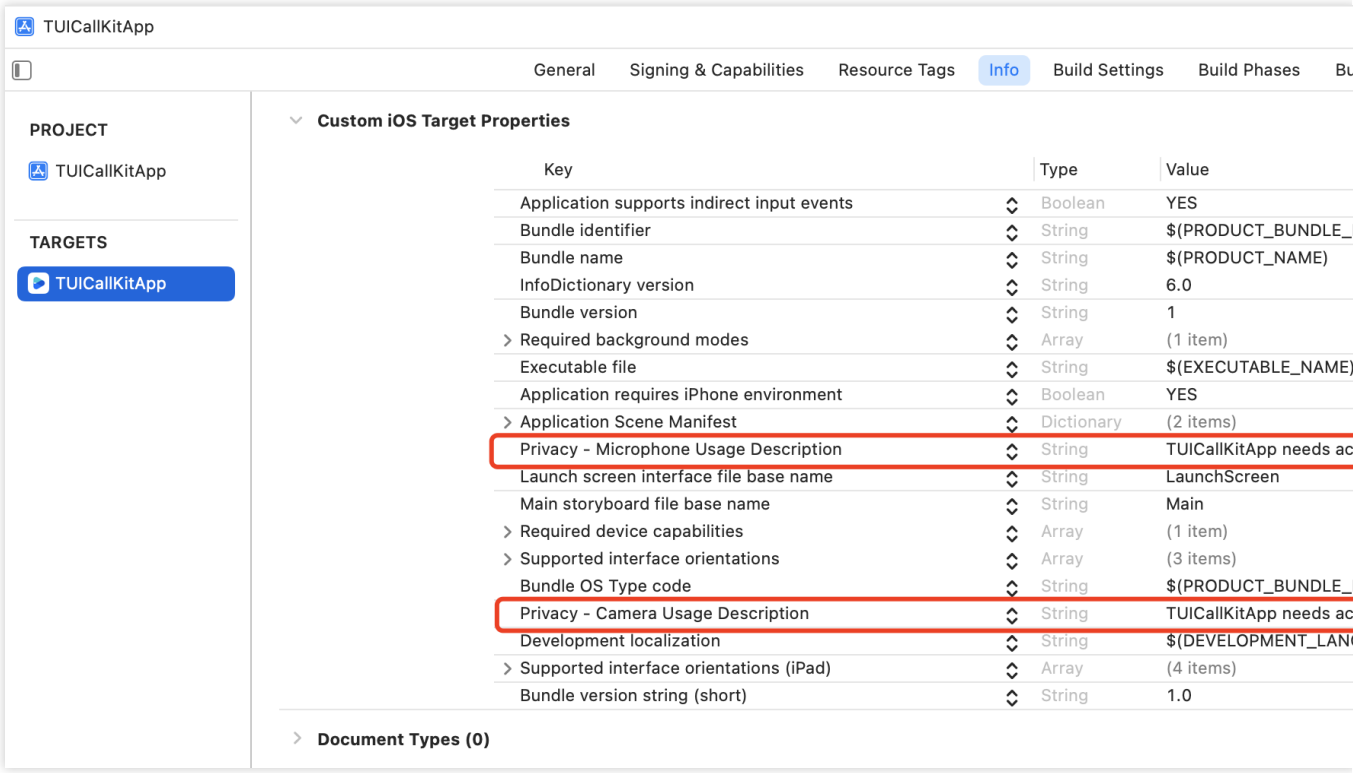
```
pod repo update
```

ステップ3：プロジェクト設定の完了

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>CallingAppはカメラ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な
<key>NSMicrophoneUsageDescription</key>
<string>CallingApp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な
```

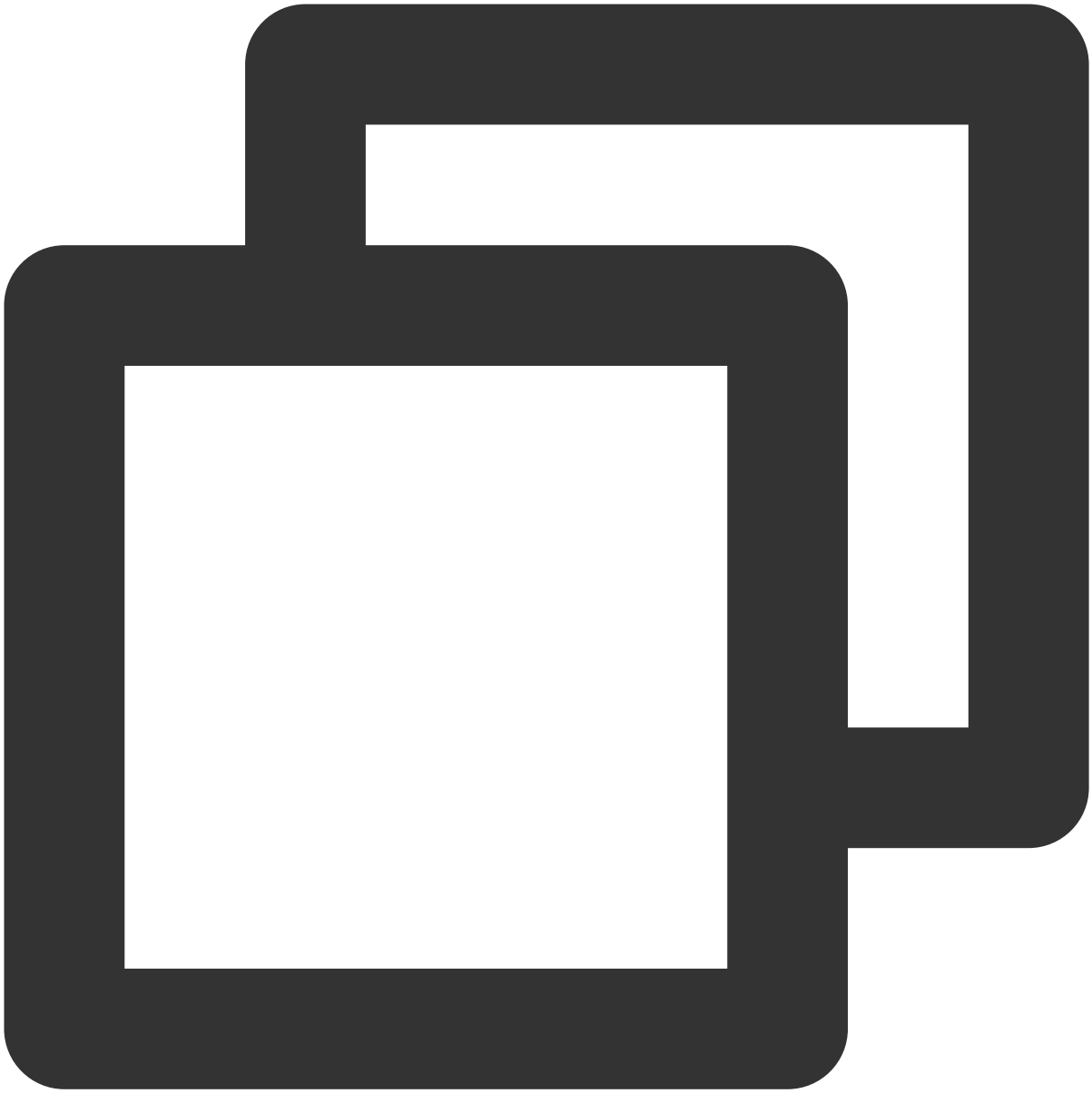


ステップ4：TUIコンポーネントへのログイン

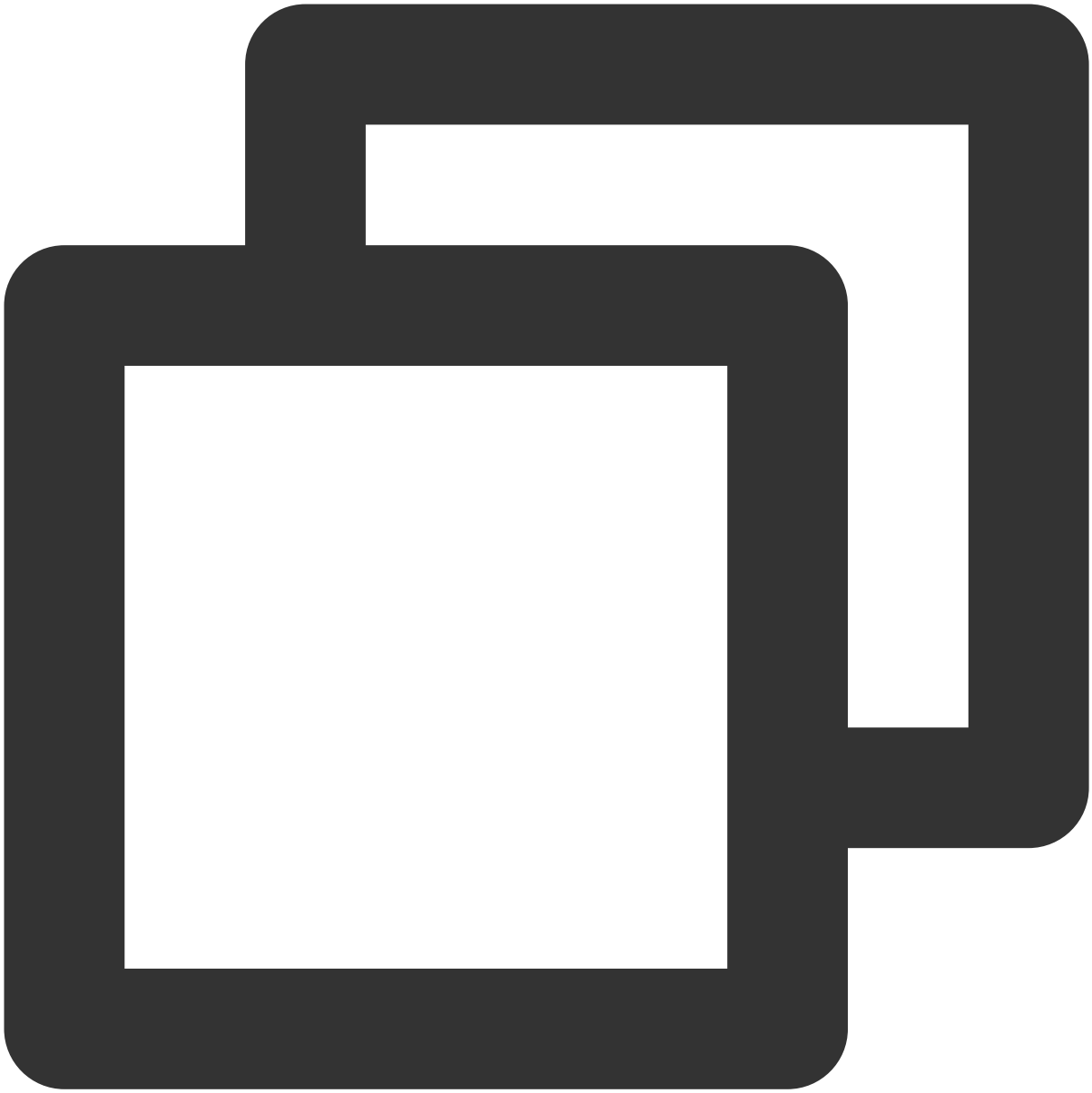
プロジェクトに次のコードを追加します。この役割は、TUICore内の関連インターフェースを呼び出して、TUIコンポーネントへのログインを完了することです。ログインに成功しなければTUICallKitの各機能を正常に使用できないため、この手順に異常がないことが重要です。関連のパラメータが正しく設定されているかどうかを注意深く確認してください。

Objective-C

Swift



```
// コンポーネントのログイン
[TUILogin login:1400000001           // ステップ1で取得したSDKAppIDに置き換えてください
         userID:@"denny"             // ご自身のUserIDに置き換えてください
         userSig:@"xxxxxxxxxxx"      // コンソールでUserSigを計算し、この位置に入力すること
            succ:^{
    NSLog(@"login success");
} fail:^(int code, NSString *msg) {
    NSLog(@"login failed, code: %d, error: %@", code, msg);
}
```



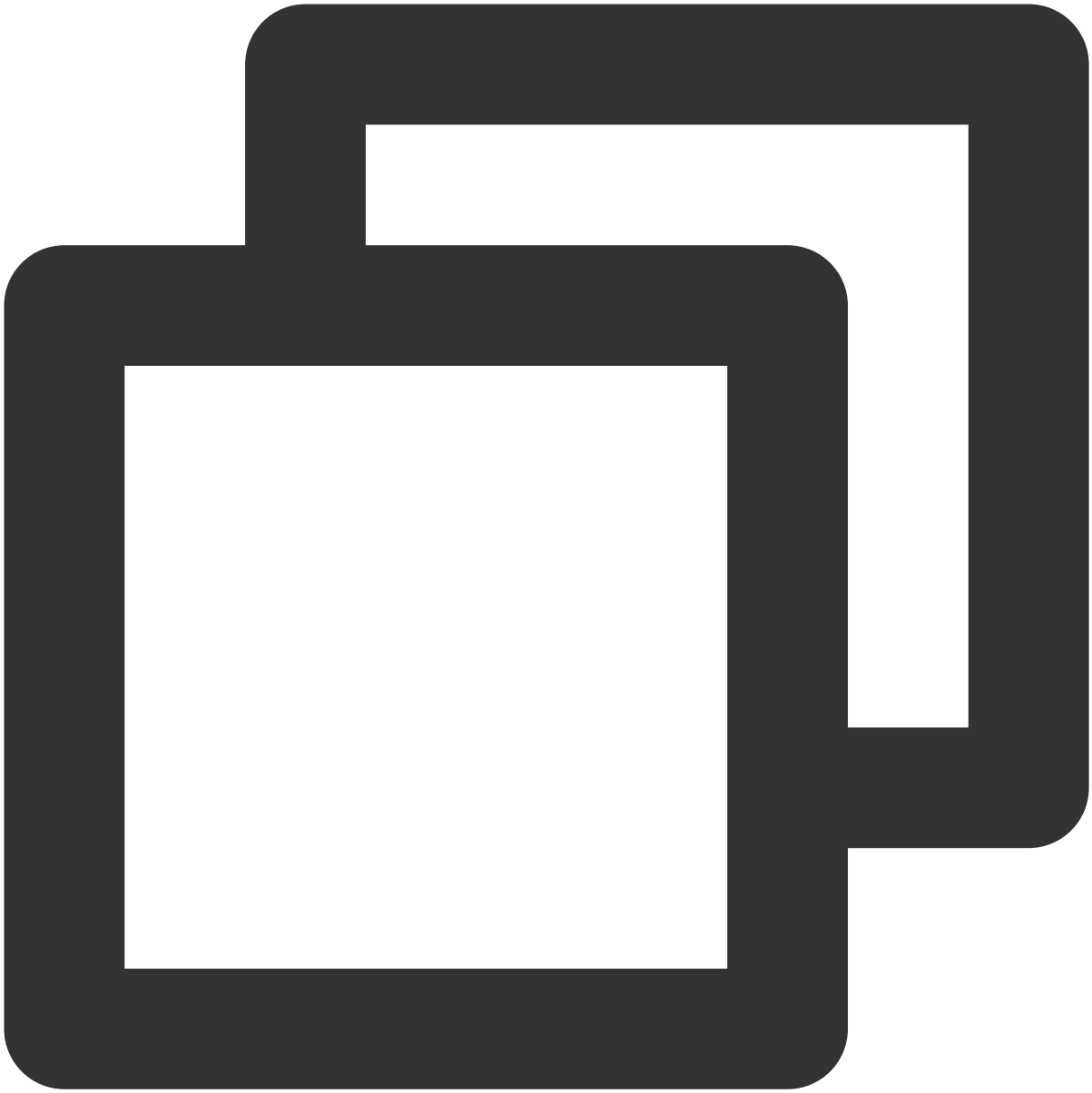
```
// コンポーネントのログイン
TUILogin.login(1400000001,                    // ステップ1で取得したSDKAppIDに置き換えてく
            userID: "denny",                  // ご自身のUserIDに置き換えてください
            userSig: "xxxxxxxxxxx") {         // コンソールでUserSigを計算し、この位置に入
    print("login success")
} fail: { (code, message) in
    print("login failed, code: \\(code), error: \\(message ?? "nil")")
}
```

パラメータの説明：
ここではlogin関数内で使用するいくつかの重要パラメータについて詳しくご説明します。
sdkAppId：ステップ1の最後の段階で取得済みです。ここでは説明を省略します。
userId：現在のユーザーIDです。文字列タイプでは、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：ステップ3で取得したSecretKeyを使用してSDKAppID、UserIDなどの情報を暗号化し、UserSigを取得します。これは認証用の証明書であり、現在のユーザーがTRTCサービスを使用できるかどうかをTencent Cloudが識別するために用いられます。コンソールのUserSigの生成ボタンで、一時的に使用可能なUserSigを生成することができます。
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。
ご注意：
この手順は、現在開発者から最も多くのフィードバックが寄せられる手順でもあります。よくみられる問題には次のようなものがあります。
sdkAppIdの設定に誤りがある。国内サイトのSDKAppIDは一般的に140で始まる10桁の整数です。
userSigを誤って暗号化鍵（Secretkey）に設定している。userSigはSecretKeyを使用して、sdkAppId、userIdおよび有効期限などの情報を暗号化するためのものであり、Secretkeyは直接userSigに設定するものではありません。
userIdが「1」、「123」、「111」などの簡単な文字列で設定されている。**TRTCは同一のUserIDによる複数端末へのログインをサポートしていない**ため、複数人によるコラボレーション開発の場合、「1」、「123」、「111」のようなuserIdは同僚に占有されていることが多く、ログイン失敗につながりやすいです。このため、デバッグの際に、識別度の高いuserIdを設定することをお勧めします。
GithubのサンプルコードでgenTestUserSig関数を使用してローカルでuserSigを計算すると、現在のアクセスフローをさらに速くスタートさせることができますが、この方法ではSecretKeyがAppのコード内で開示されてしまい、その後のアップグレードおよびSecretKeyの保護にとって不利益になります。そのため、userSigの計算ロジックはサーバーに置いて実行し、またAppがTUICallKitコンポーネントを使用するたびに、リアルタイムに計算したuserSigをサーバーにリクエストするようにすることを強く推奨します。

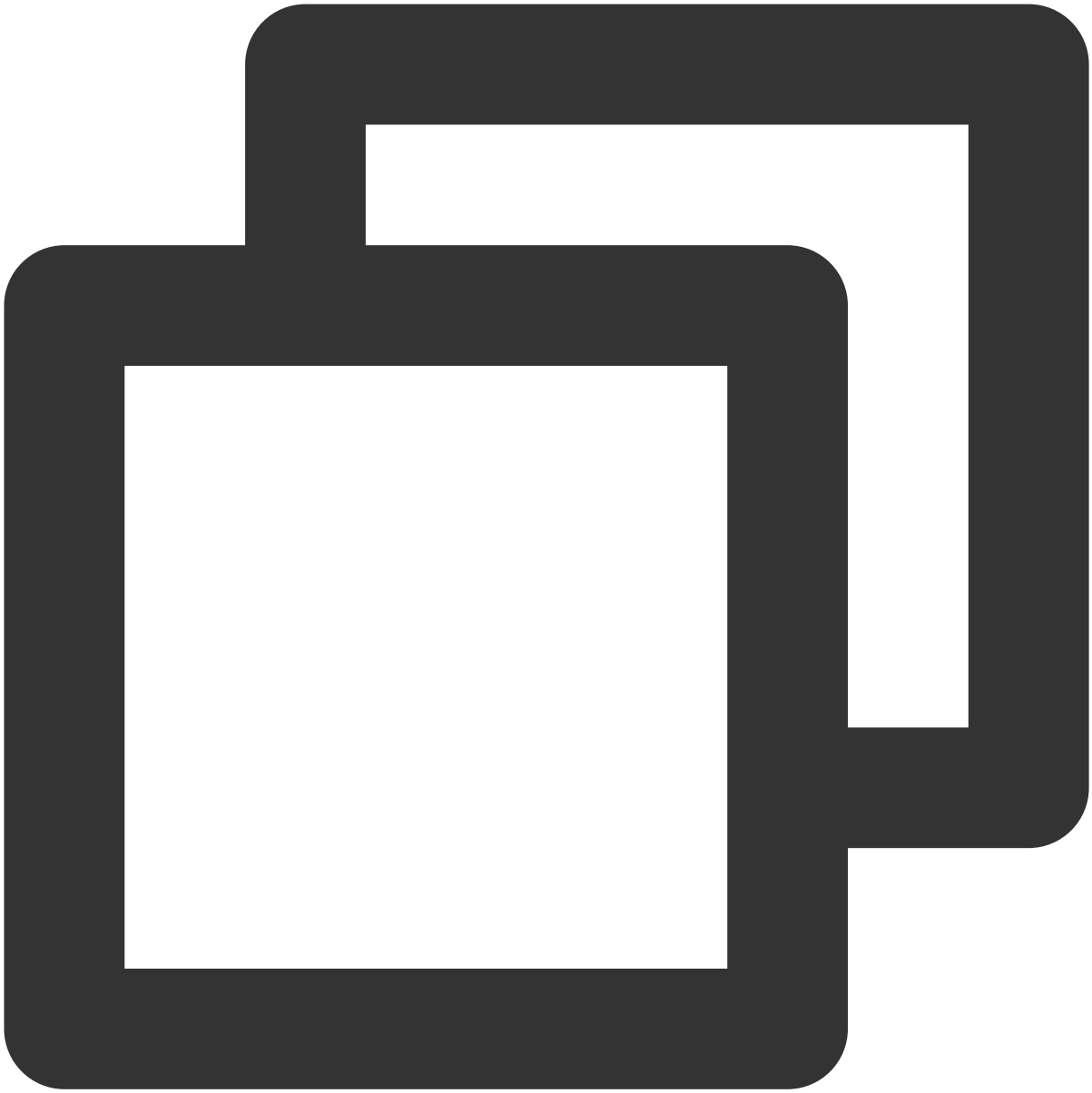
ステップ5：通話する

1対1ビデオ通話

TUICallKitのcall関数を呼び出し、 通話タイプと着呼者のuserIdを指定すると、音声またはビデオ通話を開始することができます。
Objective-C
Swift



```
// 1対1ビデオ通話を開始します(userIdはmikeとします)
[[TUICallKit createInstance] call:@"mike" callMediaType:TUICallMediaTypeVideo];
```



```
// 1対1ビデオ通話を開始します(userIdはmikeとします)
TUICallKit.createInstance().call(userId: "mike", callMediaType: .video)
```

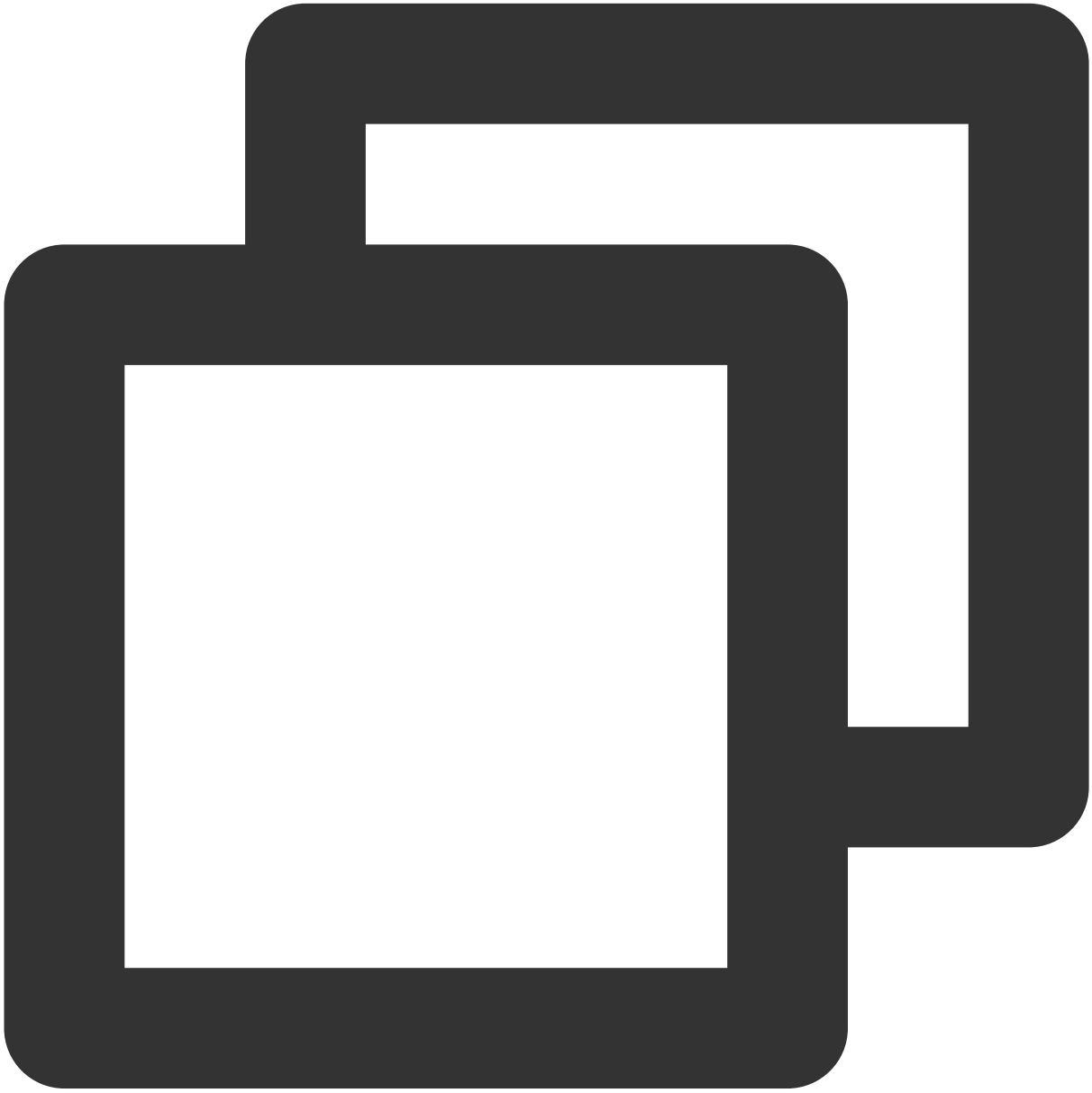
パラメータ	タイプ	意味
userId	String	ターゲットユーザーのUserID：`"mike"`
callMediaType	TUICallMediaType	通話のメディアタイプ。例：`TUICallMediaTypeVideo`

グループ内ビデオ通話

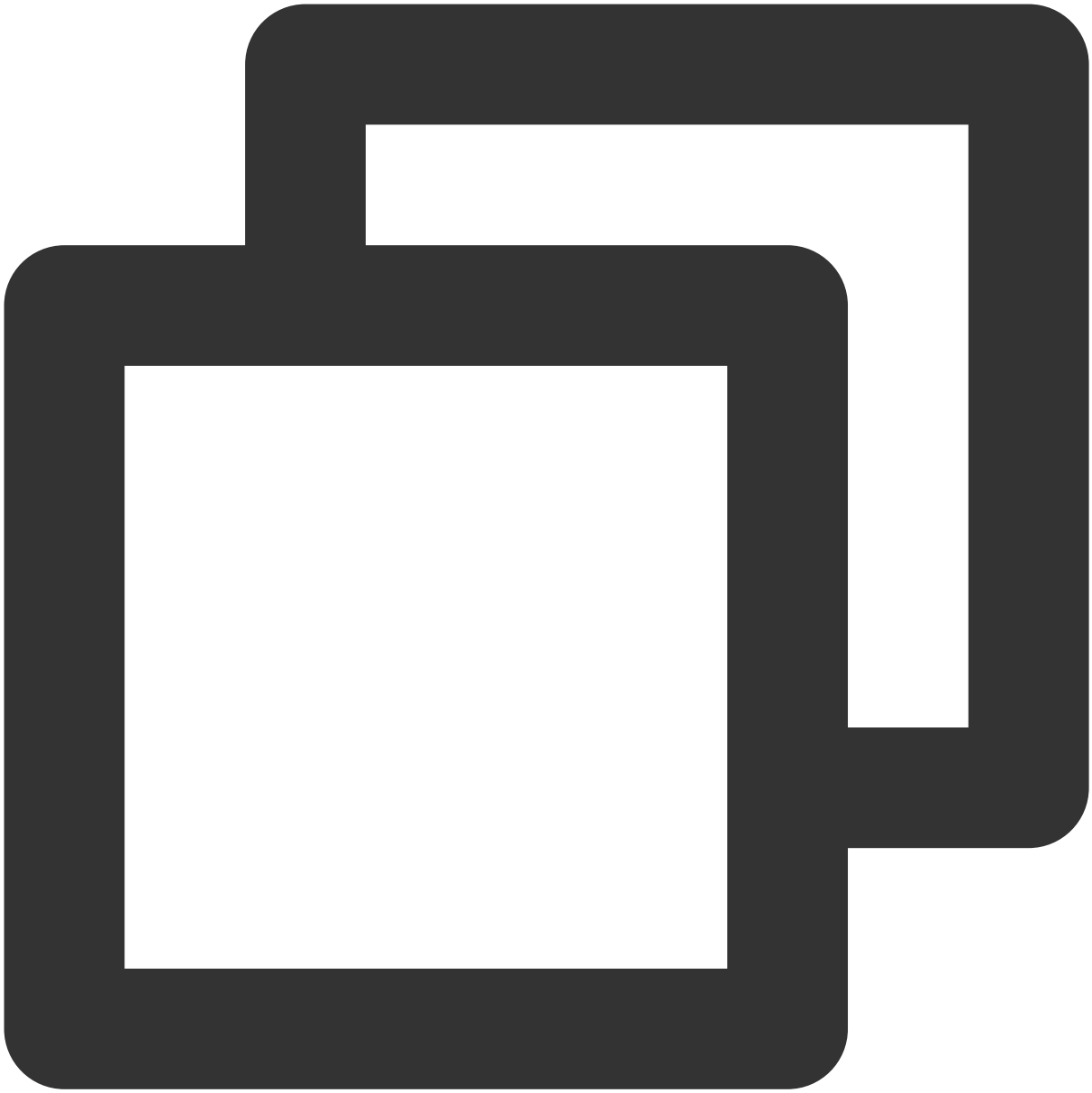
TUICallKitのgroupCall関数を呼び出し、 通話タイプと着呼者のUserIDリストを指定すると、グループ内で音声またはビデオ通話を開始することができます。

Objective-C

Swift



```
[[TUICallKit createInstance] groupCall:@"12345678" userIdList:@[@"denny", @"mike",
```



```
TUICallKit.createInstance().groupCall(groupId: "12345678", userIdList: ["denny", "m
```

パラメータ	タイプ	意味
groupId	String	グループID。例： `@"12345678"`
userIdList	Array	ターゲットユーザーのuserIdリスト。例： `@[@"denny", @"mike", @"tommy"]`
callMediaType	TUICallMediaType	

		通話のメディアタイプ。例：`TUICallMediaTypeVideo`

説明：
グループの作成の詳細についてはIMグループ管理をご参照ください。もしくはIM TUIKitを直接使用し、チャット、通話などのシナリオをワンストップで統合することもできます。
TUICallKitでは現在、グループ以外の多人数ビデオ通話はサポートしていません。もし必要な場合は、colleenyu@tencent.comまでぜひフィードバックをお寄せください。

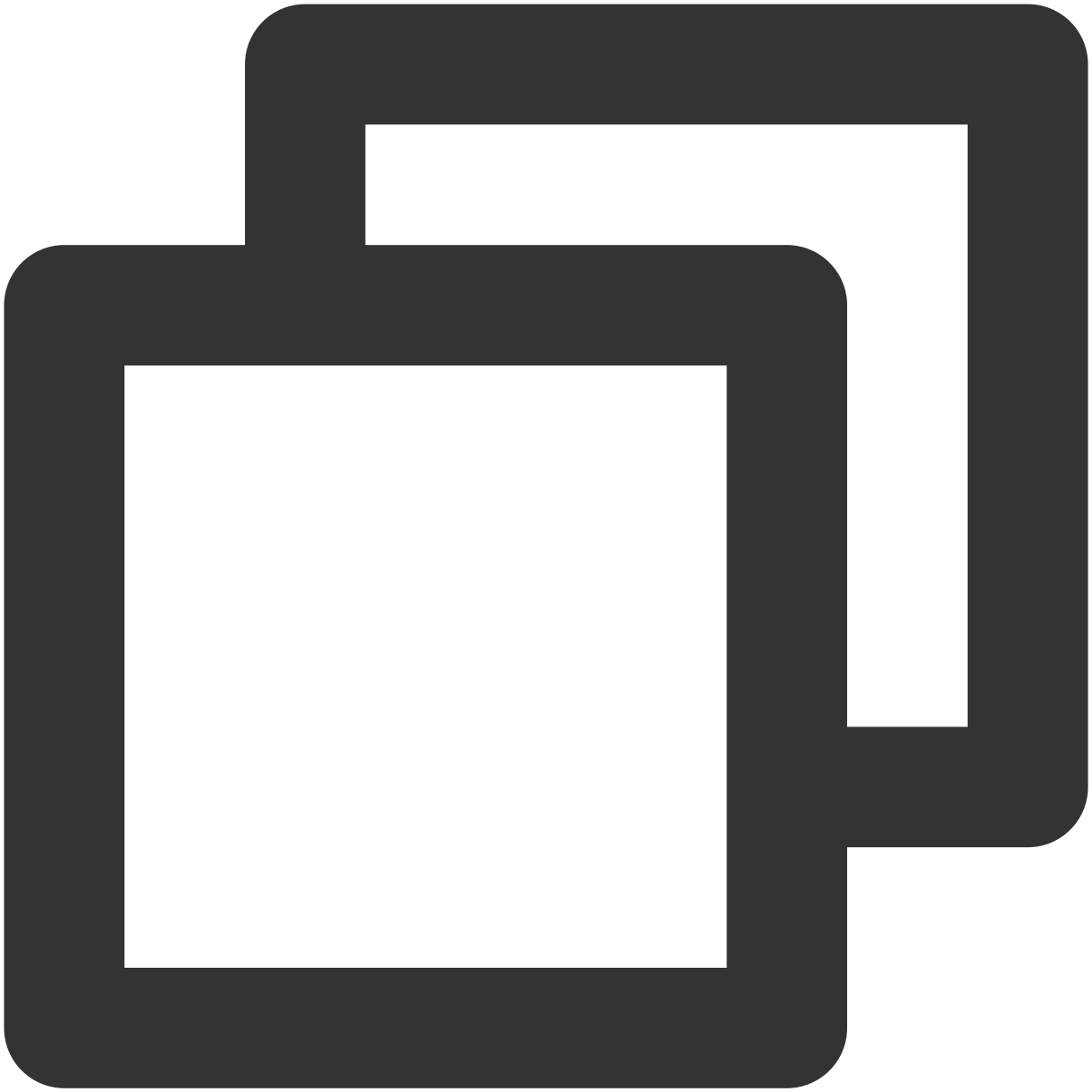
ステップ6：通話に応答する

ステップ4が完了し、通話リクエストを受信すると、TUICallKitコンポーネントは対応する応答画面を自動的に起動します。

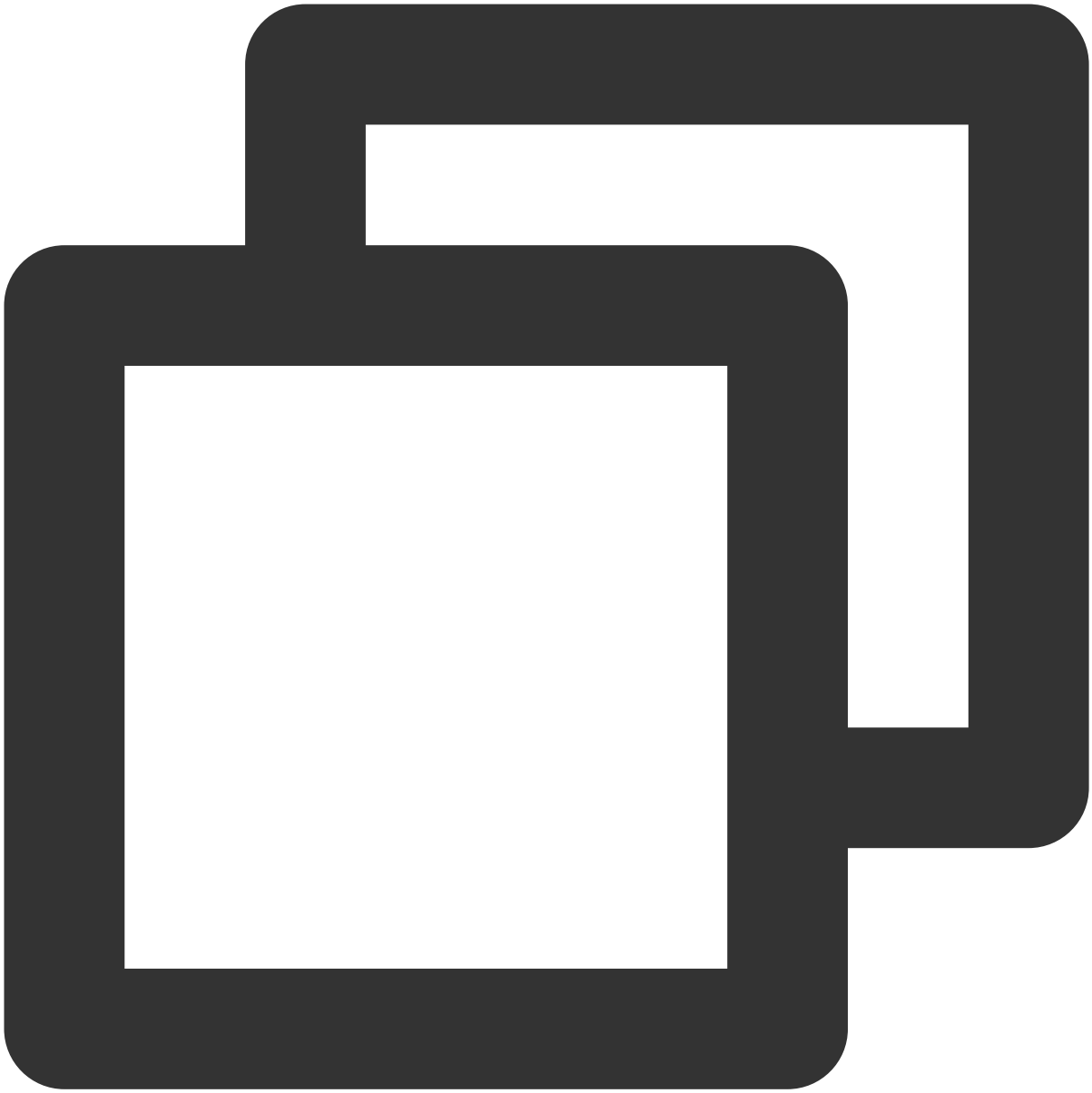
ステップ7：その他の特徴

1、ニックネーム&プロフィール画像の設定

カスタムニックネームまたはプロフィール画像を設定したい場合は、次のインターフェースを使用して更新できます。
Objective-C
Swift



```
[[TUICallKit createInstance] setSelfInfo:@"ニックネーム" avatar:@"プロフィール画像URL" s

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

}];
```



```
TUICallKit.createInstance().setSelfInfo(nickname: "ニックネーム", avatar: "プロフィール

}) { code, desc in

}
```

ご注意：

ユーザーのプライバシー上の制限により、フレンド以外との通話では、呼ばれるニックネームとプロフィール画像の更新が遅れる可能性があります。一度通話が成功するとスムーズに更新されるようになります。

2、オフライン通知

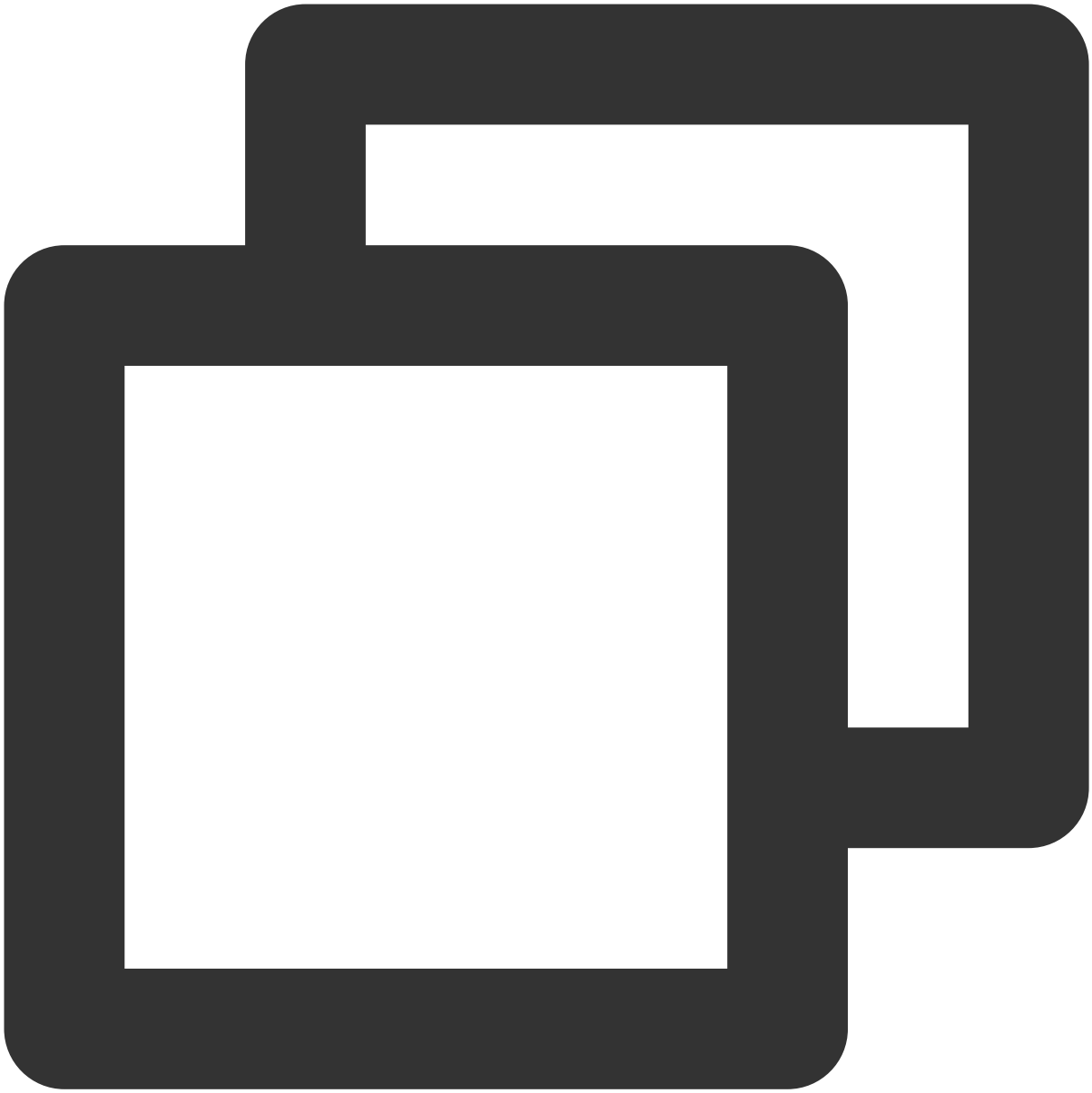
上記の手順が完了すると、オーディオビデオ通話の発信と接続が実現できますが、業務上「Appのプロセスが強制終了された後」または「APPがバックエンドに戻った後」もオーディオビデオ通話リクエストを正常に受信できる必要がある場合は、オフライン通知機能を追加する必要があります。詳細については**オフライン通知（iOS）**をご参照ください。

3、フローティングウィンドウ機能

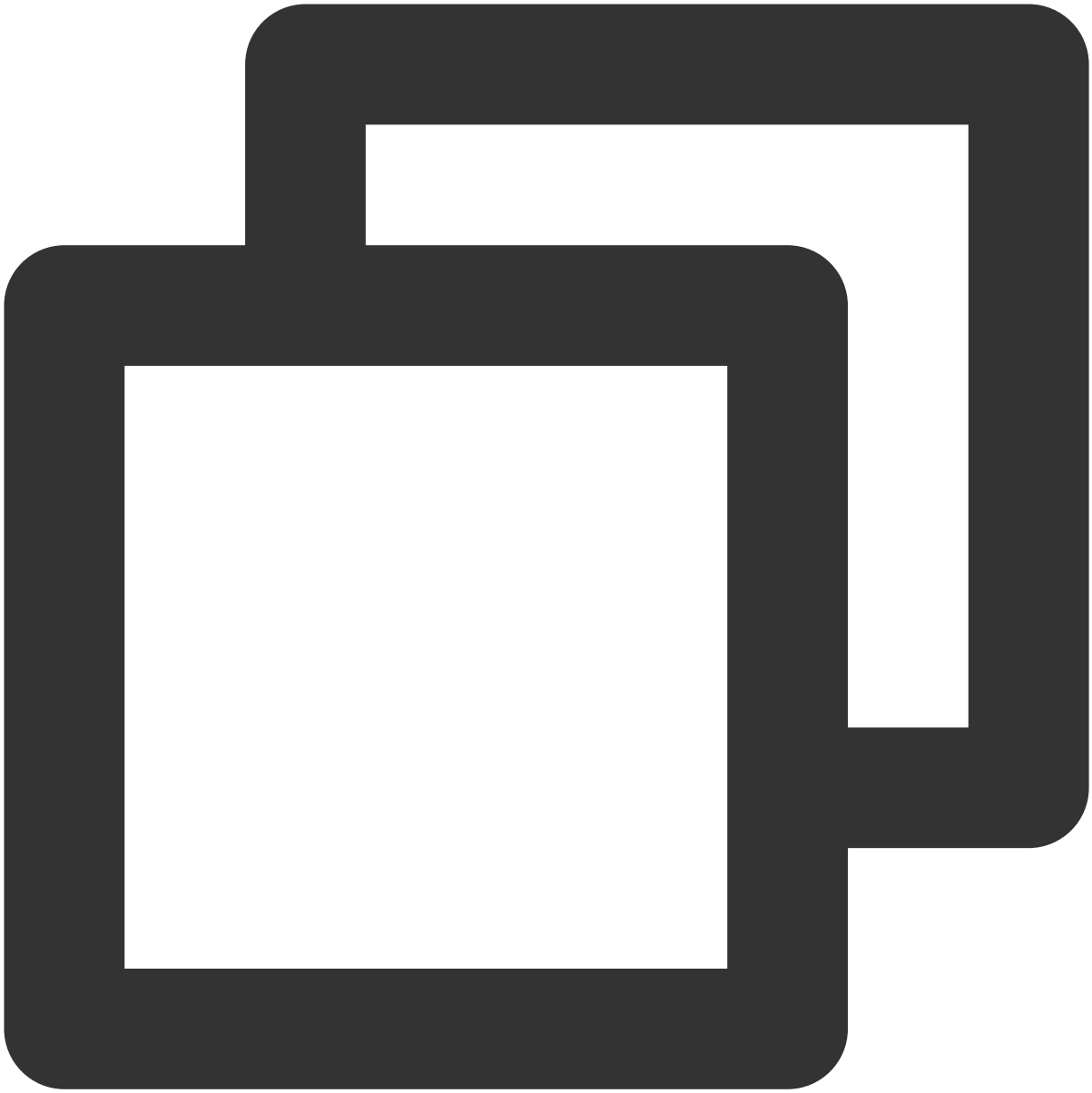
業務上、フローティングウィンドウ機能を有効にする必要がある場合は、TUICallKitコンポーネントの初期化の際に次のインターフェースを呼び出してこの機能を有効化することができます。

Objective-C

Swift



```
[[TUICallKit createInstance] enableFloatWindow:YES];
```



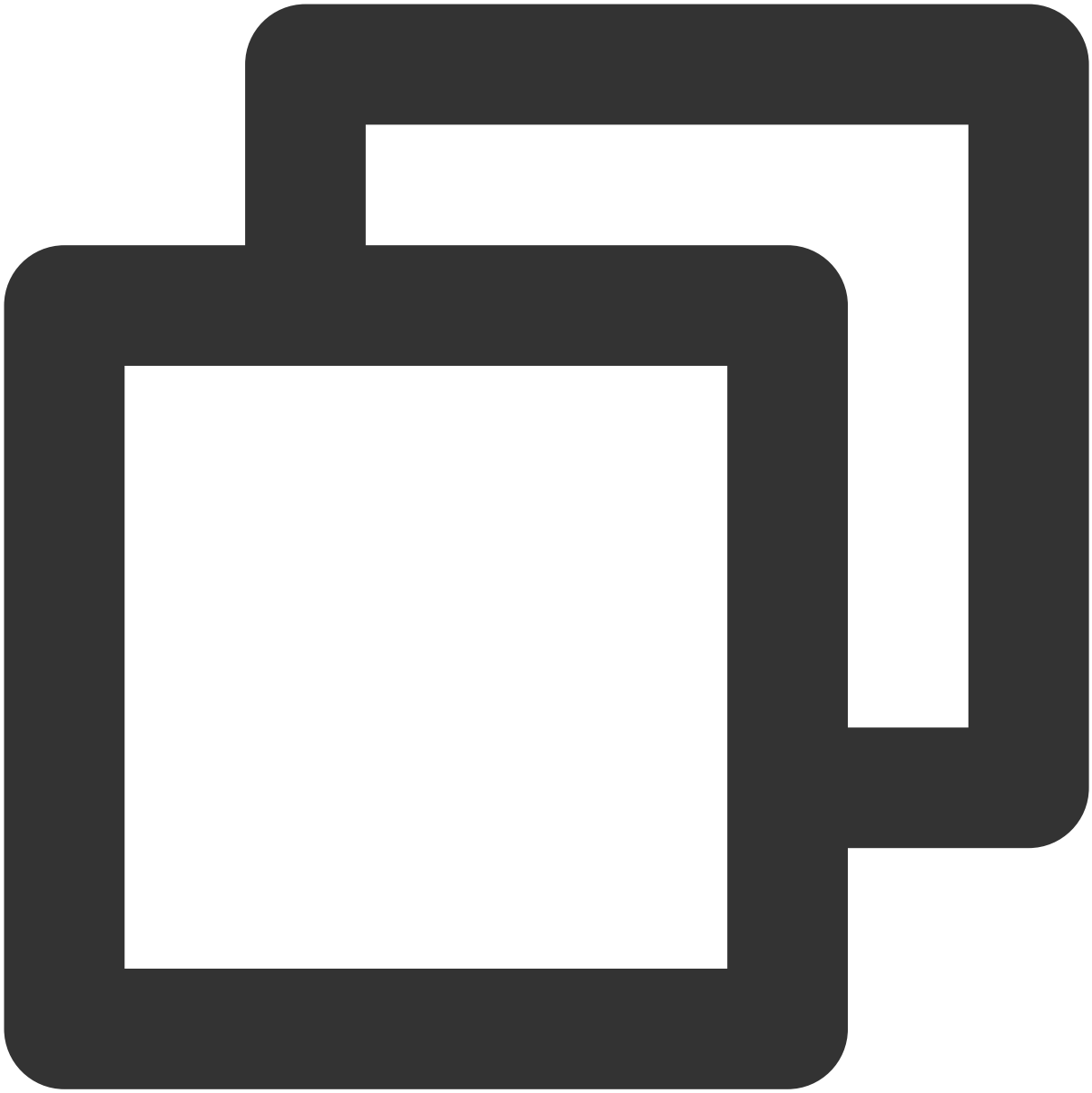
```
TUICallKit.createInstance().enableFloatWindow(true)
```

4. 通話ステータスの監視

業務上、通話の開始、終了、通話中のネットワーク品質などの**通話ステータスの監視**が必要な場合は、次のイベントを監視することができます。

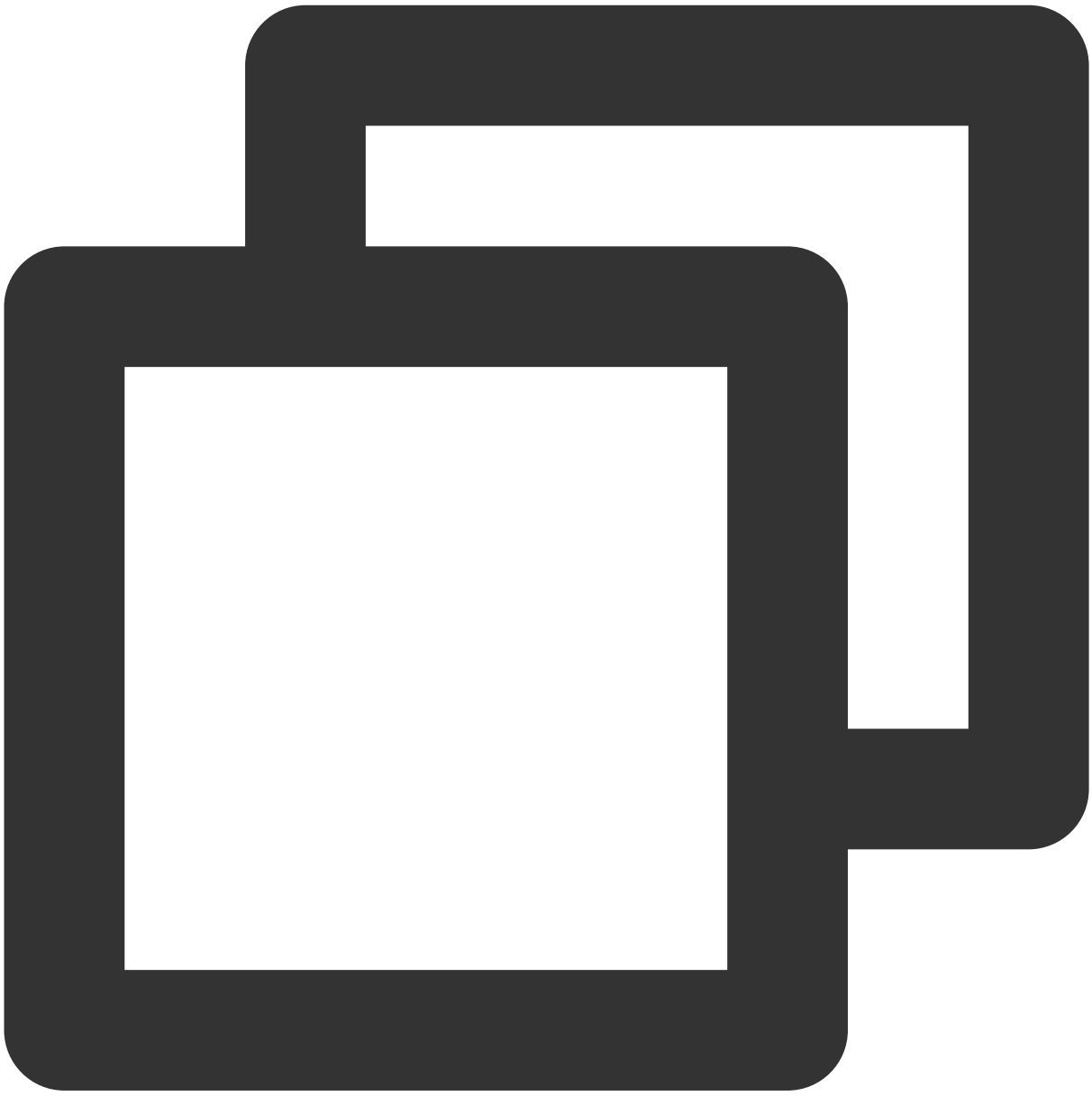
Objective-C

Swift



```
[[TUICallEngine createInstance] addObserver:self];

- (void)onCallBegin:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaTy


}
- (void)onCallEnd:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaType


}
- (void)onUserNetworkQualityChanged:(NSArray<TUINetworkQualityInfo *> *)networkQual
```

```
}
```



```
TUICallEngine.createInstance().add(self)

public func onCallBegin(roomId: TUIRoomId, callMediaType: TUICallMediaType, callRol

}
public func onCallEnd(roomId: TUIRoomId, callMediaType: TUICallMediaType, callRole:
```

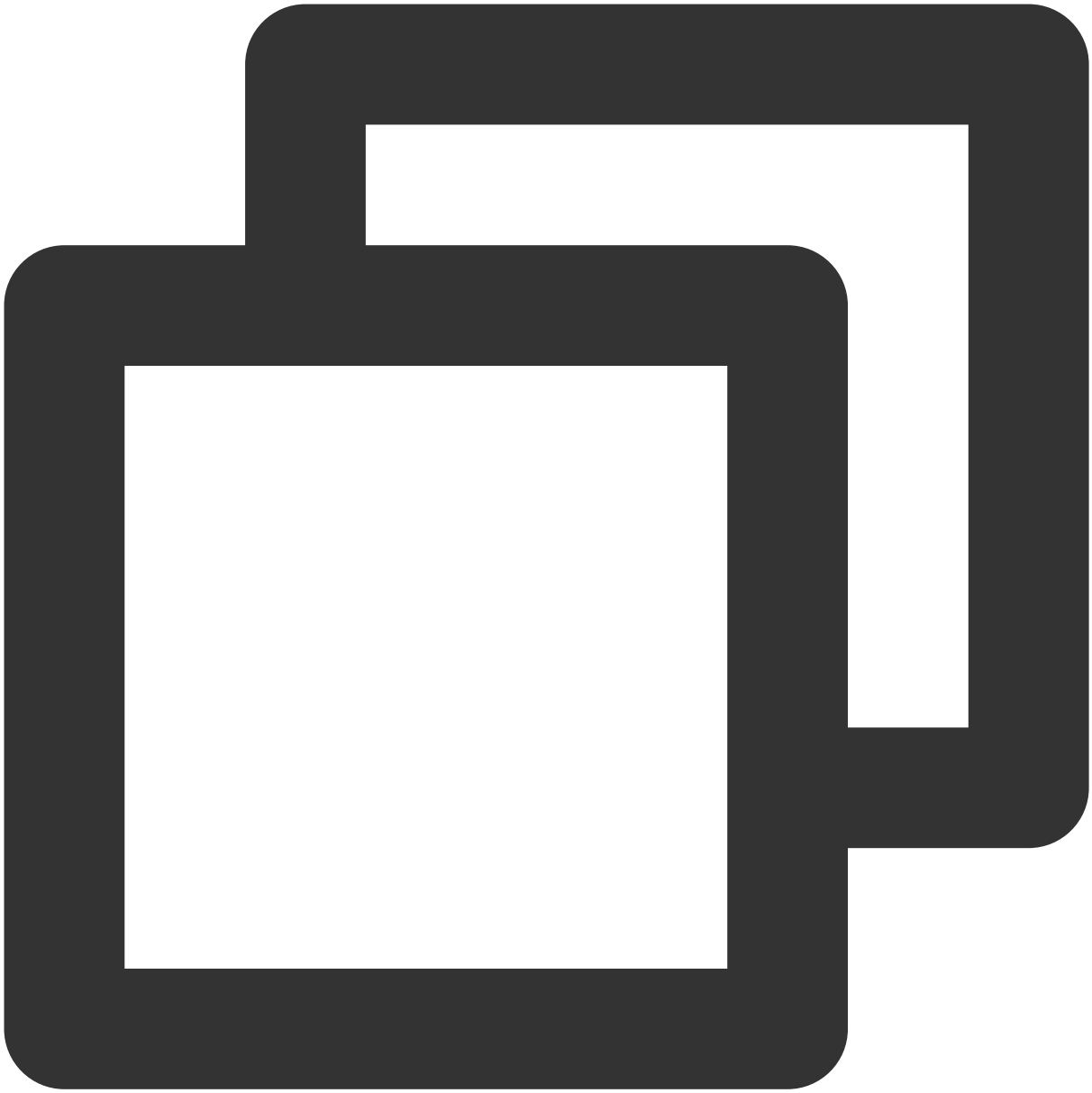
```
}
public func onUserNetworkQualityChanged(networkQualityList: [TUINetworkQualityInfo]

}
```

5、カスタム着信音

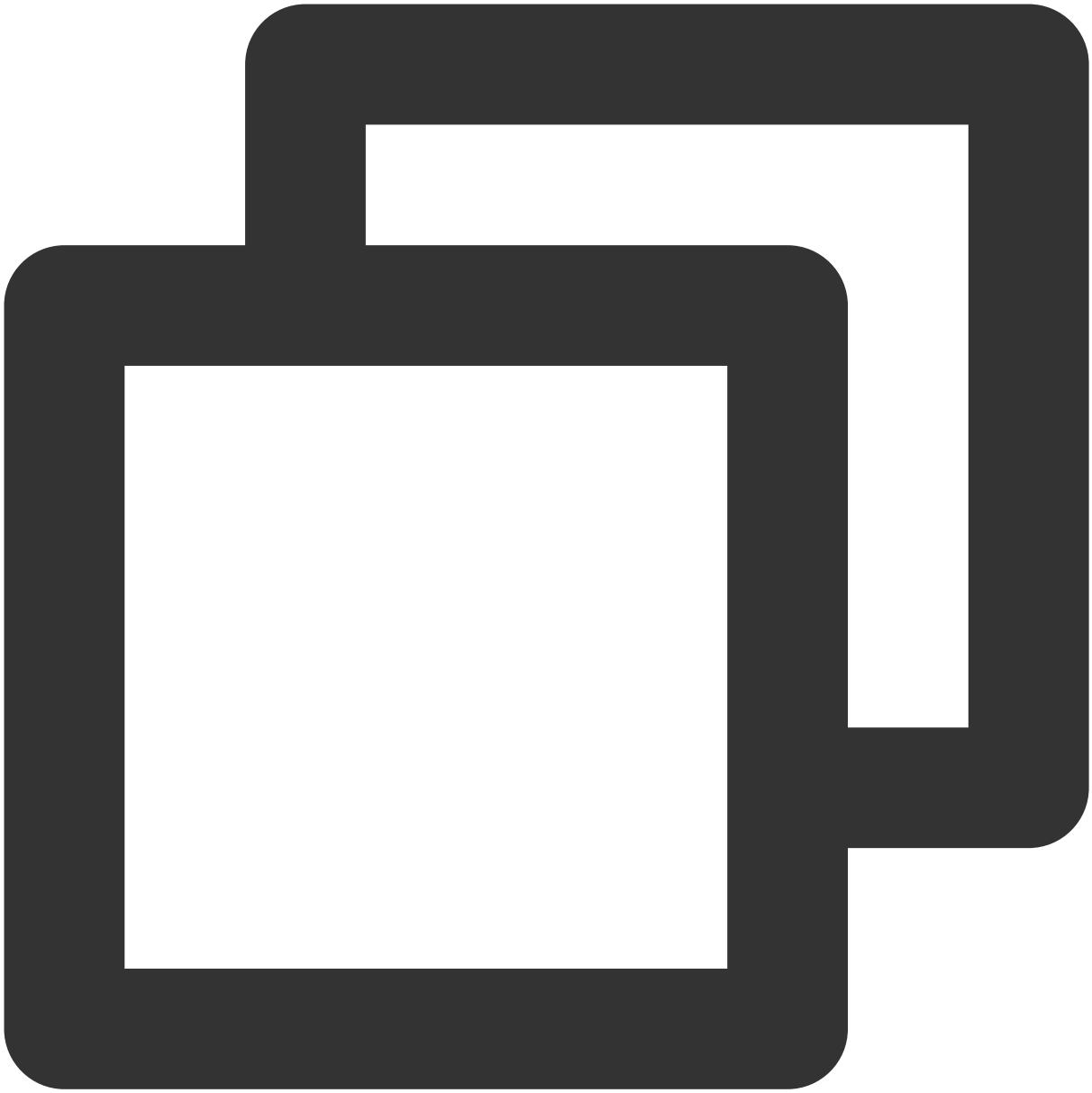
着信音をカスタマイズしたい場合は、次のインターフェースによって設定できます。

Objective-C

Swift



```
[[TUICallKit createInstance] setCallingBell:filePath];
```



```
TUICallKit.createInstance().setCallingBell(filePath: filePath)
```

よくあるご質問

「The package you purchased does not support this ability」というエラーが表示されました。

上記のエラーが表示された場合は、現在のアプリケーションのオーディオビデオ通話機能パッケージが期限切れまたはアクティブ化されていないことを表します。ステップ1を参照してオーディオビデオ通話機能を取得またはアクティブ化し、引き続きTUICallKitコンポーネントをご利用ください。

ご意見とフィードバック

ご利用中にご提案やご意見などがございましたら、colleenyu@tencent.comまでフィードバックをお願いいたします。

Web

最終更新日：：2024-07-19 14:53:21

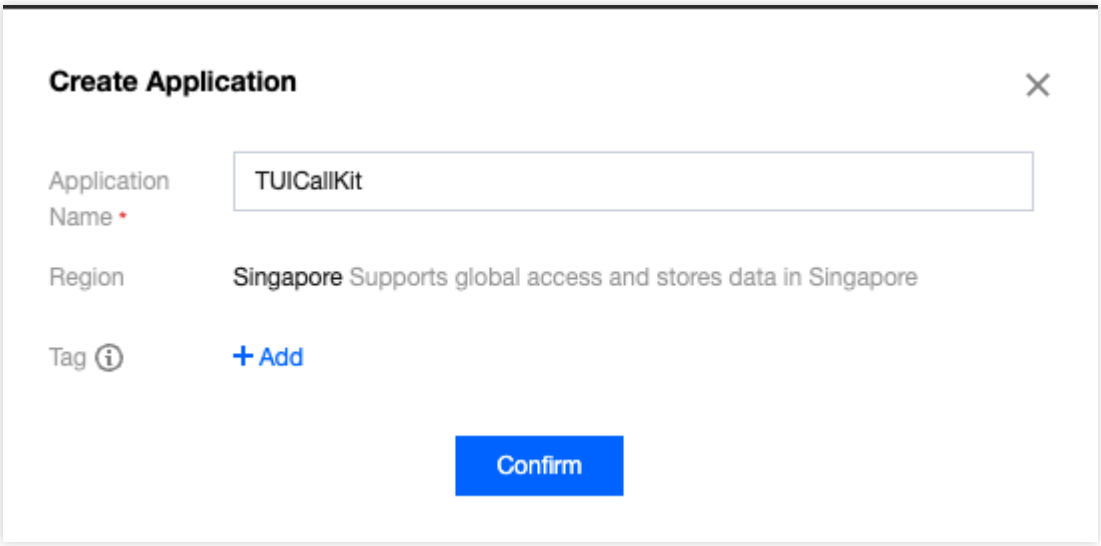
TUICallKitコンポーネントのクイックアクセス

ここでは、最短の時間でTUICallKitコンポーネントの統合を完了する方法についてご説明します。このドキュメントに沿って操作することで、次の重要手順を完了し、最終的にUIを完備したビデオ通話機能を手にすることができます。
コンポーネントの効果を体験したい場合は、TUICallKit demoクイックスタートをご参照ください。
アクセス前にデスクトップブラウザがオーディオビデオサービスをサポートしているかどうかを確認してください。詳しい要件は環境要件をご参照ください。

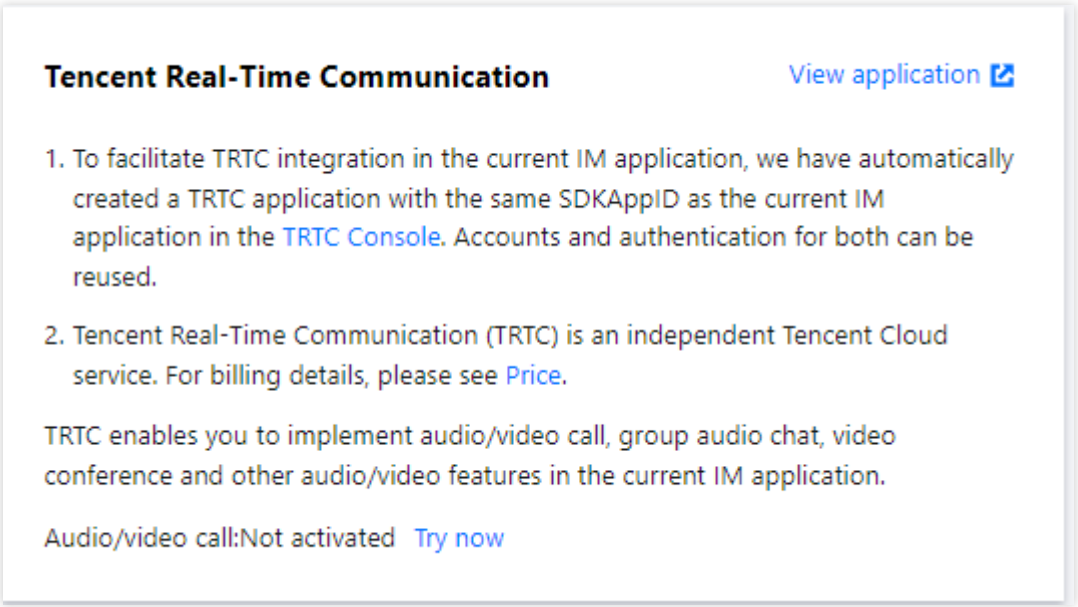
目次

ステップ1：サービスのアクティブ化

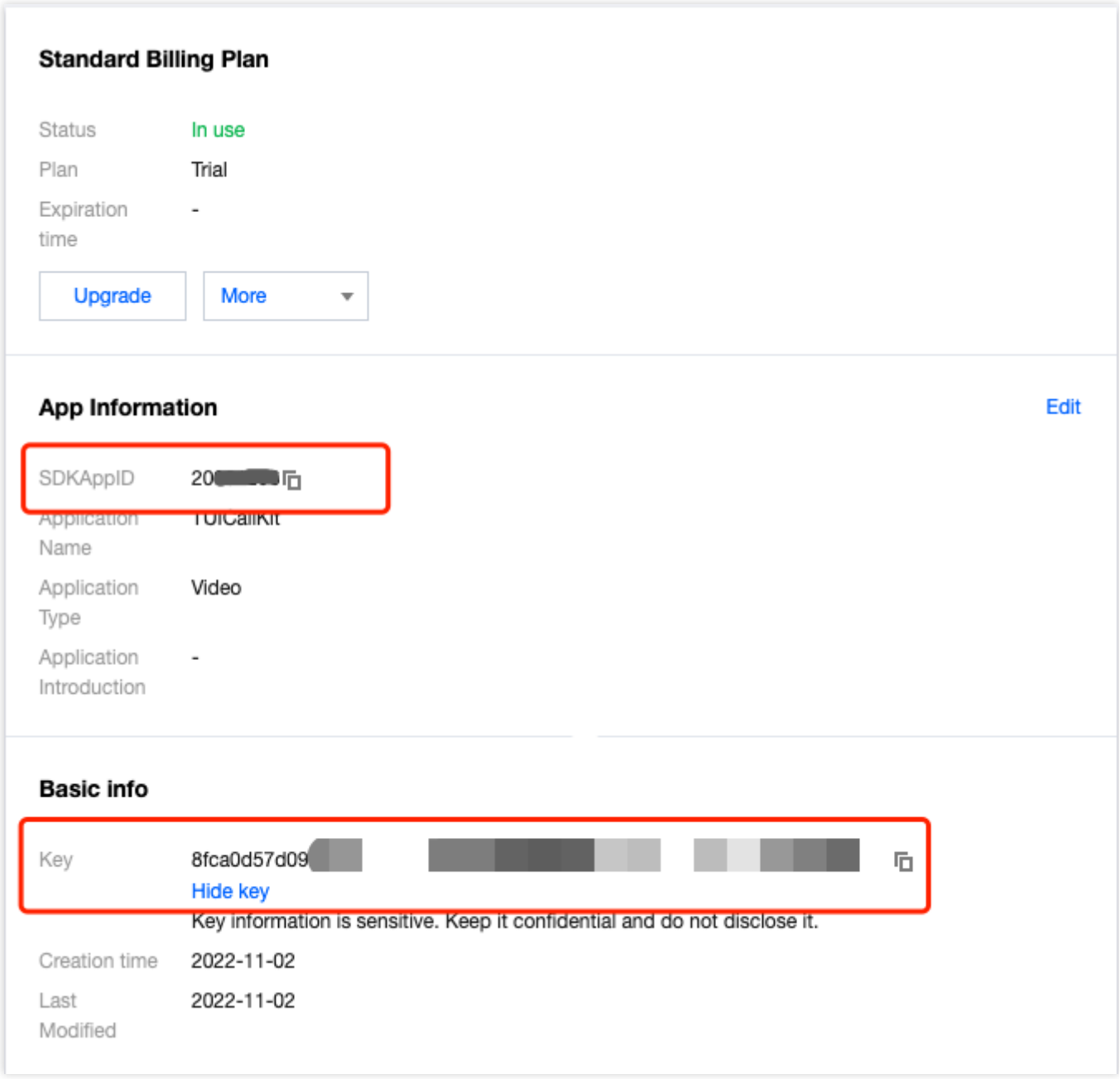
TUICallKitはTencent CloudのIMと、TRTCという2つの有料PaaSサービスをベースに構築したオーディオビデオ通信コンポーネントです。以下の手順で関連のサービスをアクティブ化し、7日間の無料トライアルサービスを体験することができます。
1. IMコンソールにログインし、**新しいアプリケーションの作成**をクリックし、ポップアップしたダイアログボックスにアプリケーション名を入力して**OK**をクリックします。



2. 作成したアプリケーションをクリックし、**基本設定**ページに進み、ページ右下隅の**TRTCサービスのアクティブ化**機能エリアで**無料体験**をクリックすると、TUICallKitの7日間無料トライアルサービスをアクティブ化することができます。



3. 同じページで**SDKAppID**と**キー**を見つけて記録します。これらは後ほど使用します。



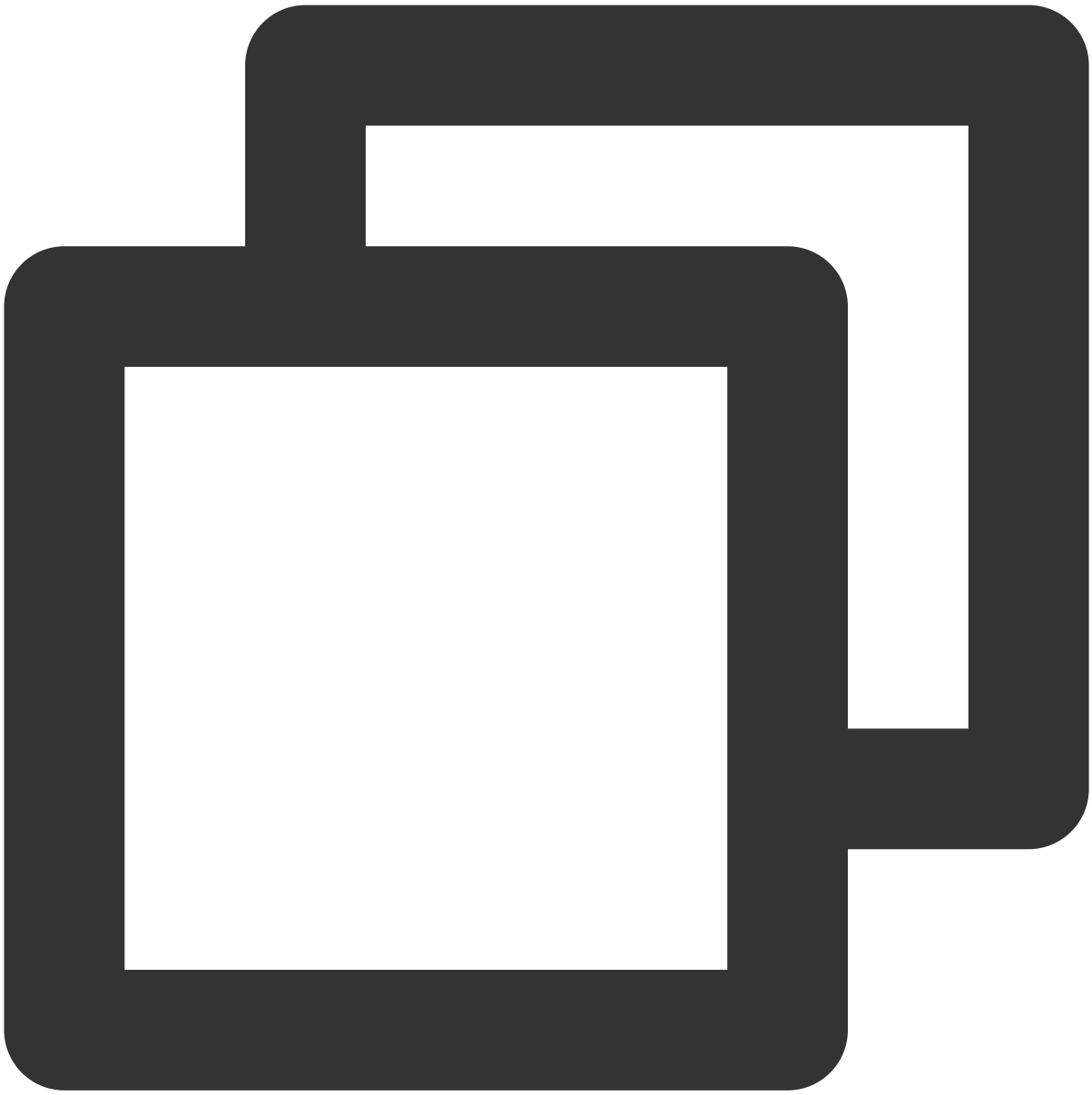
SDKAppID：IMのアプリケーションIDです。業務の分離、すなわち異なるSDKAppIDの通話が互いに接続できないようにするために用いられます。

Secretkey：IM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、IM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。このKeyは次のステップ5で使用します。

ステップ2：TUICallKitコンポーネントのインポート

方式1：npm統合

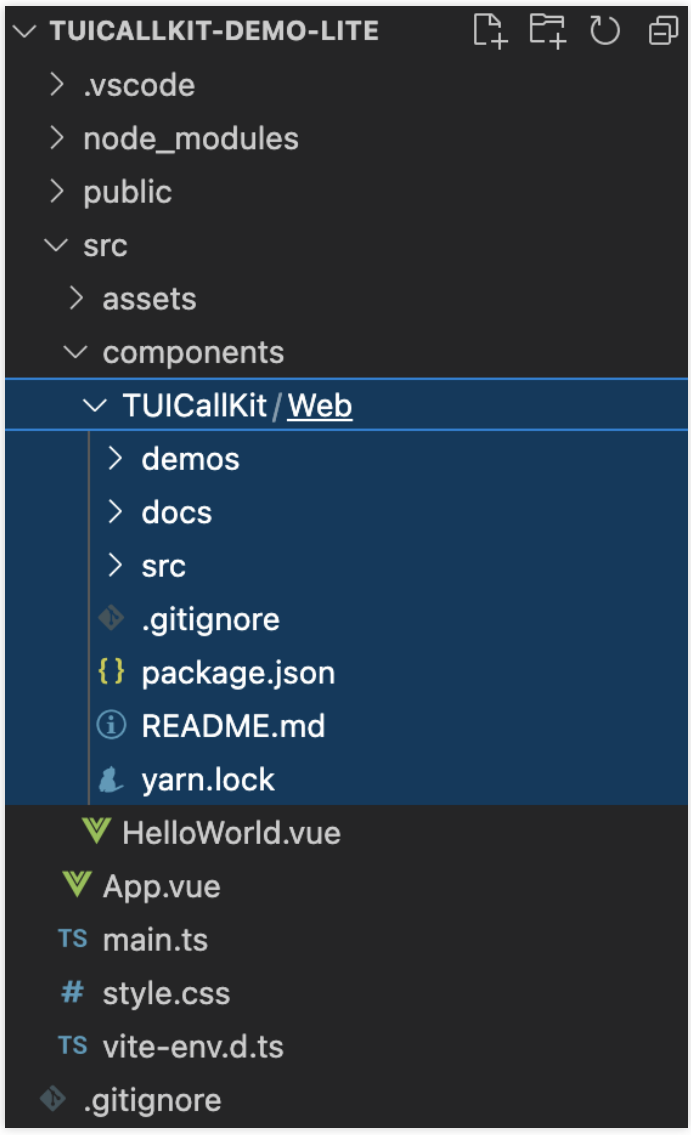
npm方式でTUICallKitコンポーネントをダウンロードします。その後の拡張をしやすくするために、TUICallKitコンポーネントをプロジェクトの `src/components/` ディレクトリにコピーすることをお勧めします。



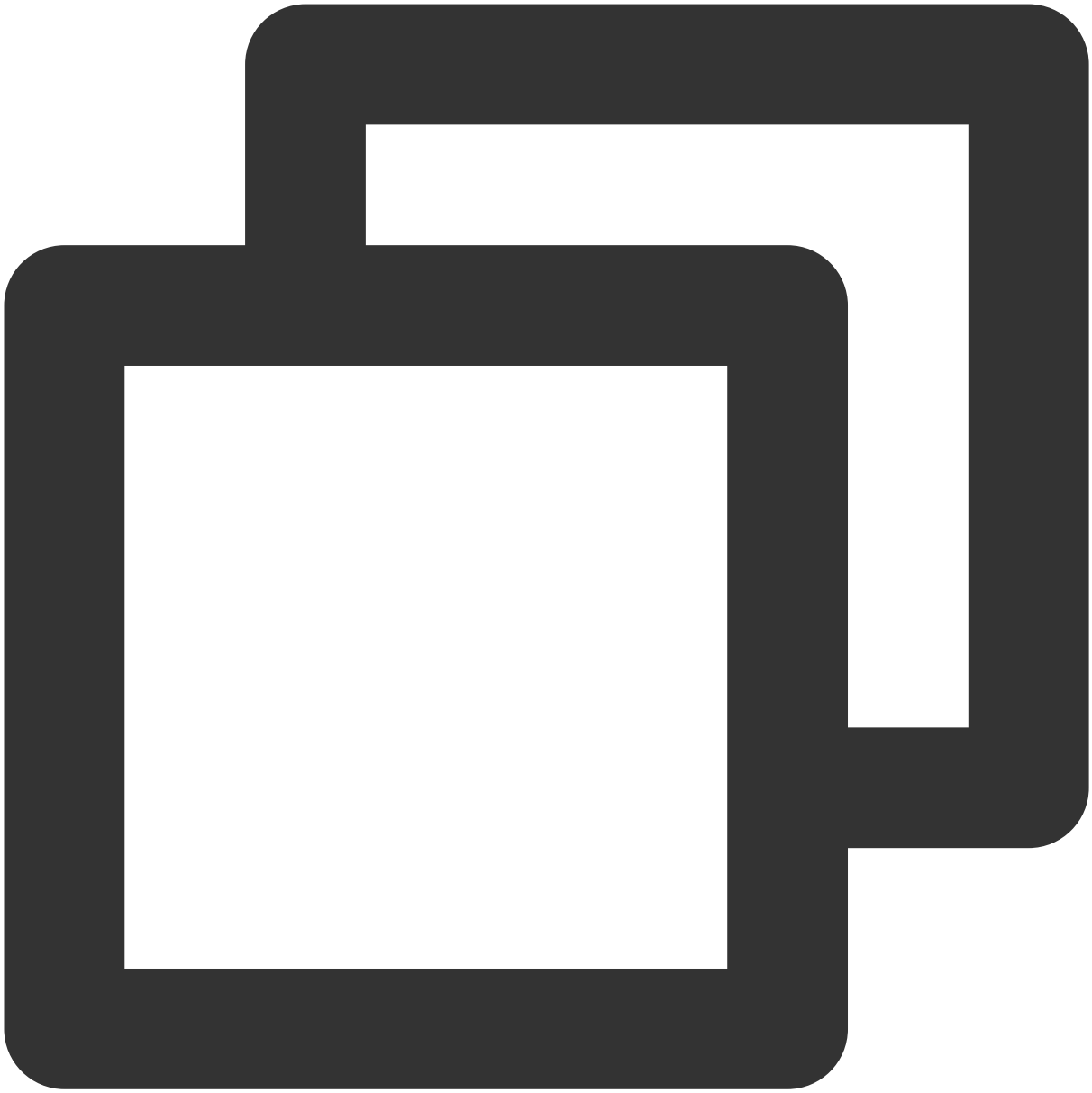
```
# macOS
yarn add @tencentcloud/call-uikit-vue  # yarnがインストールされていない場合、まず以下を実行
mkdir -p ./src/components/TUICallKit/Web && cp -r ./node_modules/@tencentcloud/call

# windows
yarn add @tencentcloud/call-uikit-vue  # yarnがインストールされていない場合、まず以下を実行
xcopy .\\node_modules\\@tencentcloud\\call-uikit-vue  .\\src\\components\\TUICallKi
```

方式2：ソースコードの統合

1. GitHubからTUICallKitソースコードをダウンロードします。 `TUICallKit/Web` フォルダをコピーして自身のプロジェクトの `src/components` フォルダに置きます。例えば次のようになります。



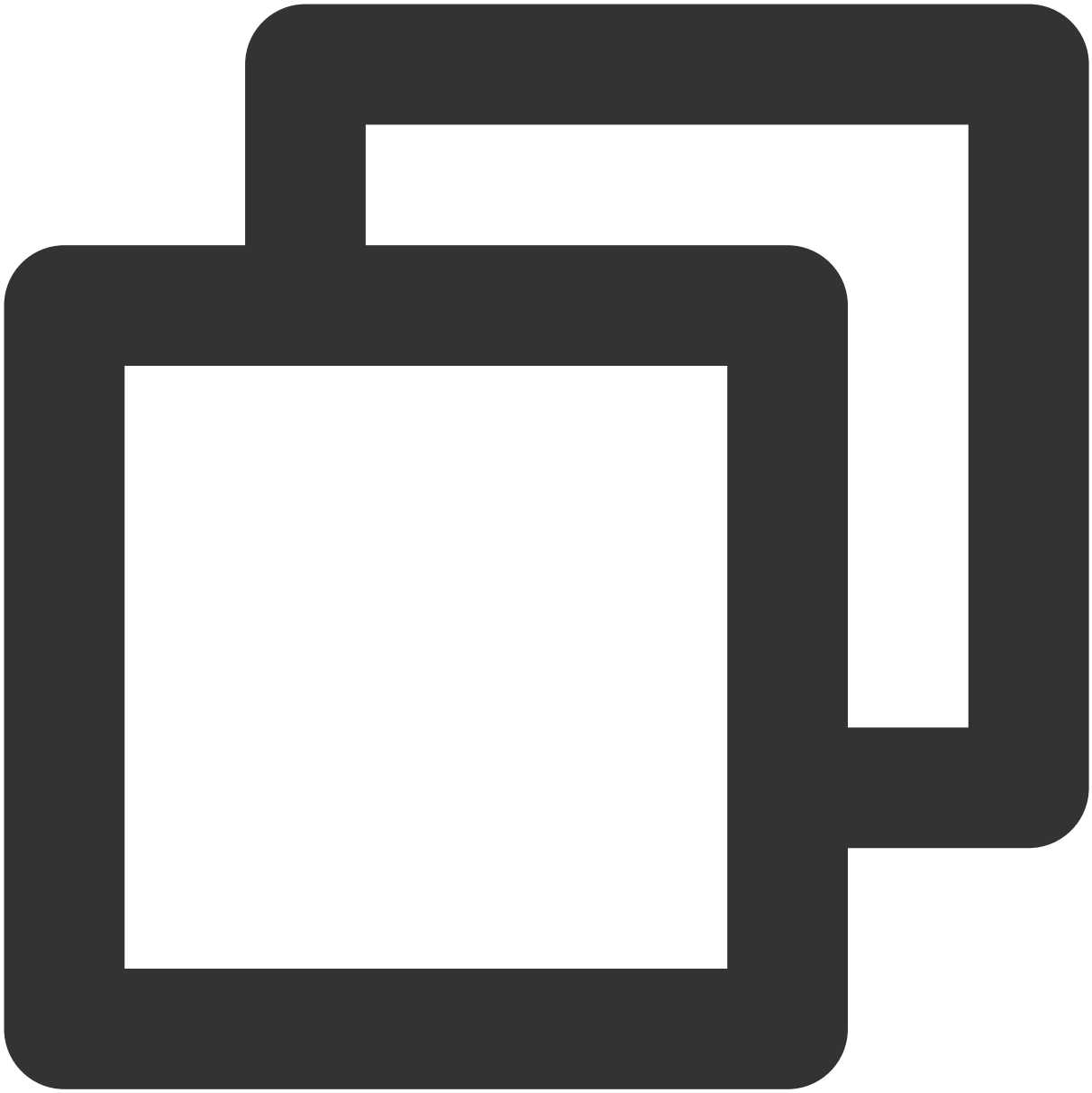
2. このフォルダに進み、実行に必要な依存をインストールします



```
cd ./src/components/TUICallKit/Web
yarn          # yarnをインストールしていない場合、先に以下を実行できます。 npm install -g yar
```

ステップ3：TUICallKitコンポーネントの呼び出し

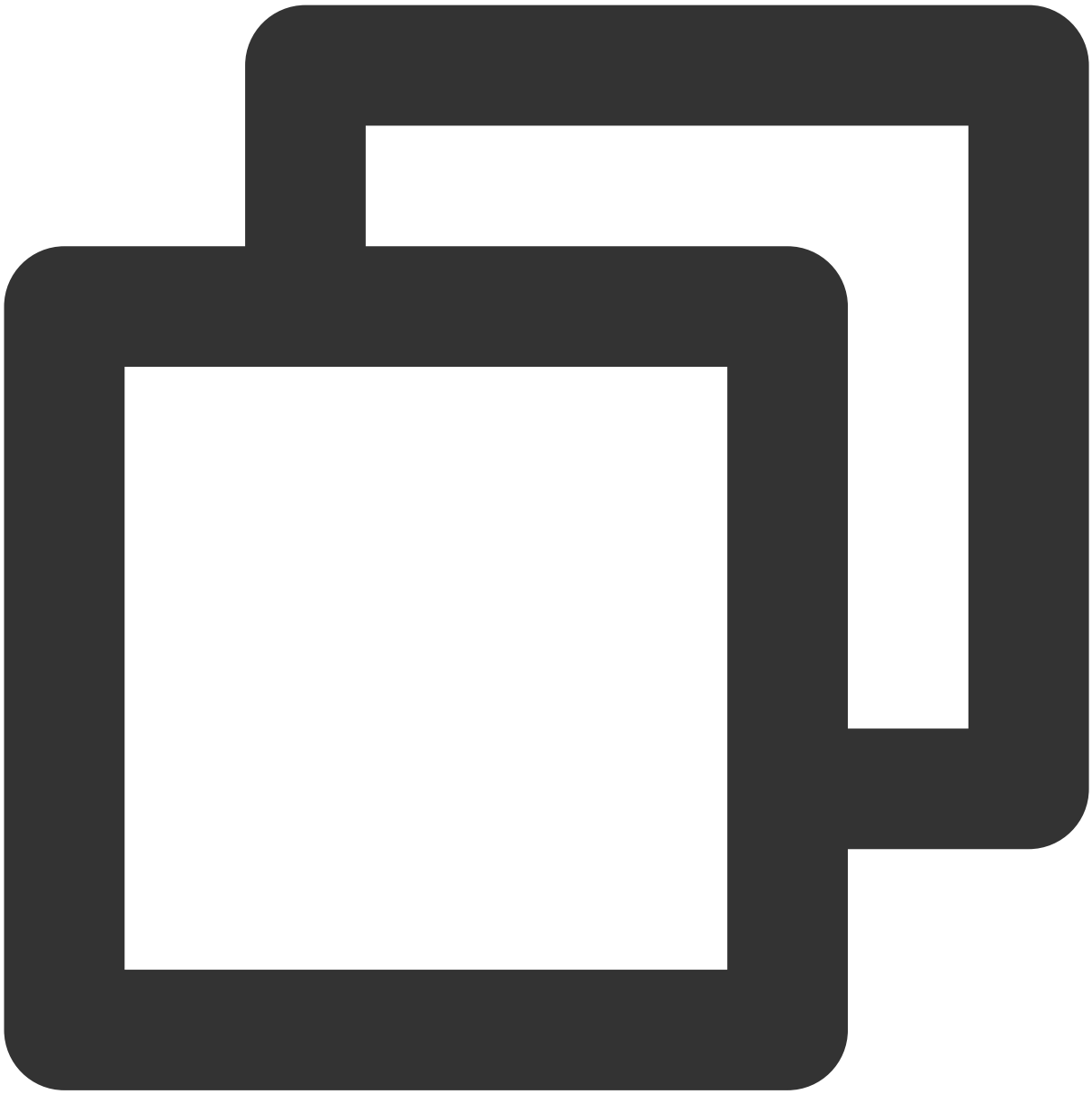
表示したいページでTUICallKitのコンポーネントを呼び出すと、通話ページを表示することができます。
1. 次のようにTUICallKit UIをインポートします。



```
<script lang="ts" setup>
import { TUICallKit } from "./components/TUICallKit/Web";
</script>

<template>
  <TUICallKit />
</template>
```

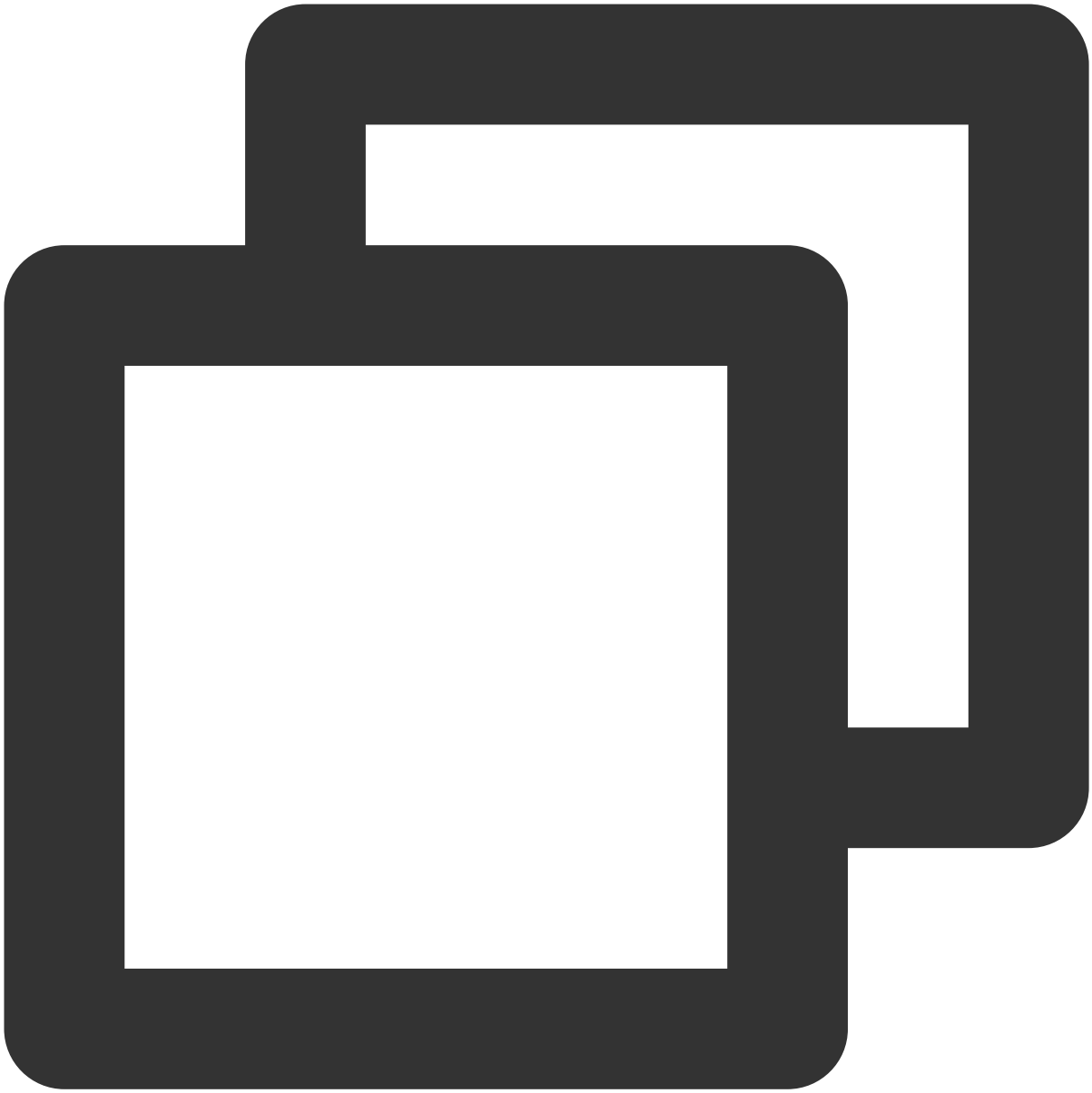
2. ユーザーログインし、電話をかけます

2.1 TUIKitのキットをすでに使用している場合は下記のコードをインポートし、TUICallKitがプラグインであることを宣言する必要があります。使用していない場合、このステップは2.2までスキップすることができます。



```
import { TUICallKit } from './components/TUICallKit/Web';
TUIKit.use(TUICallKit);
```

2.2 電話をかけるには初期化コンポーネントを保持する必要があります

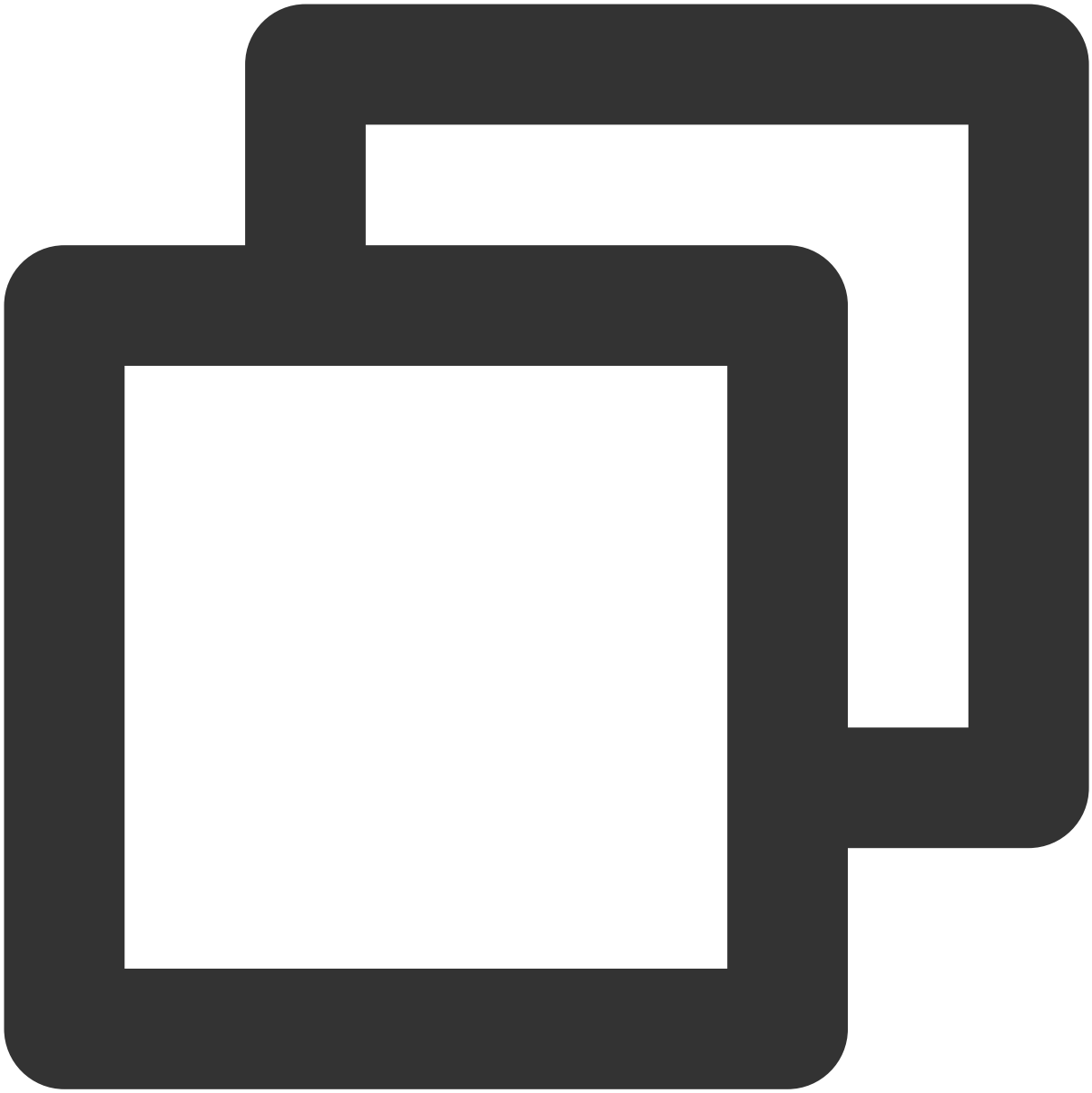


```
import { TUICallKitServer } from './components/TUICallKit/Web';
TUICallKitServer.init({ SDKAppID, userID, userSig });
```

説明：

SDKAppID、 SecretKeyの取得はステップ1をご参照ください

userSigは**一時的に** `GenerateTestUserSig.js` の `genTestUserSig(userID)` 関数を使用して計算することができます。例えば次のようになります。



```
import * as GenerateTestUserSig from "./components/TUICallKit/Web/demos/basic/publi
const { userSig } = GenerateTestUserSig.genTestUserSig(userID, SDKAppID, SecretKey)
```

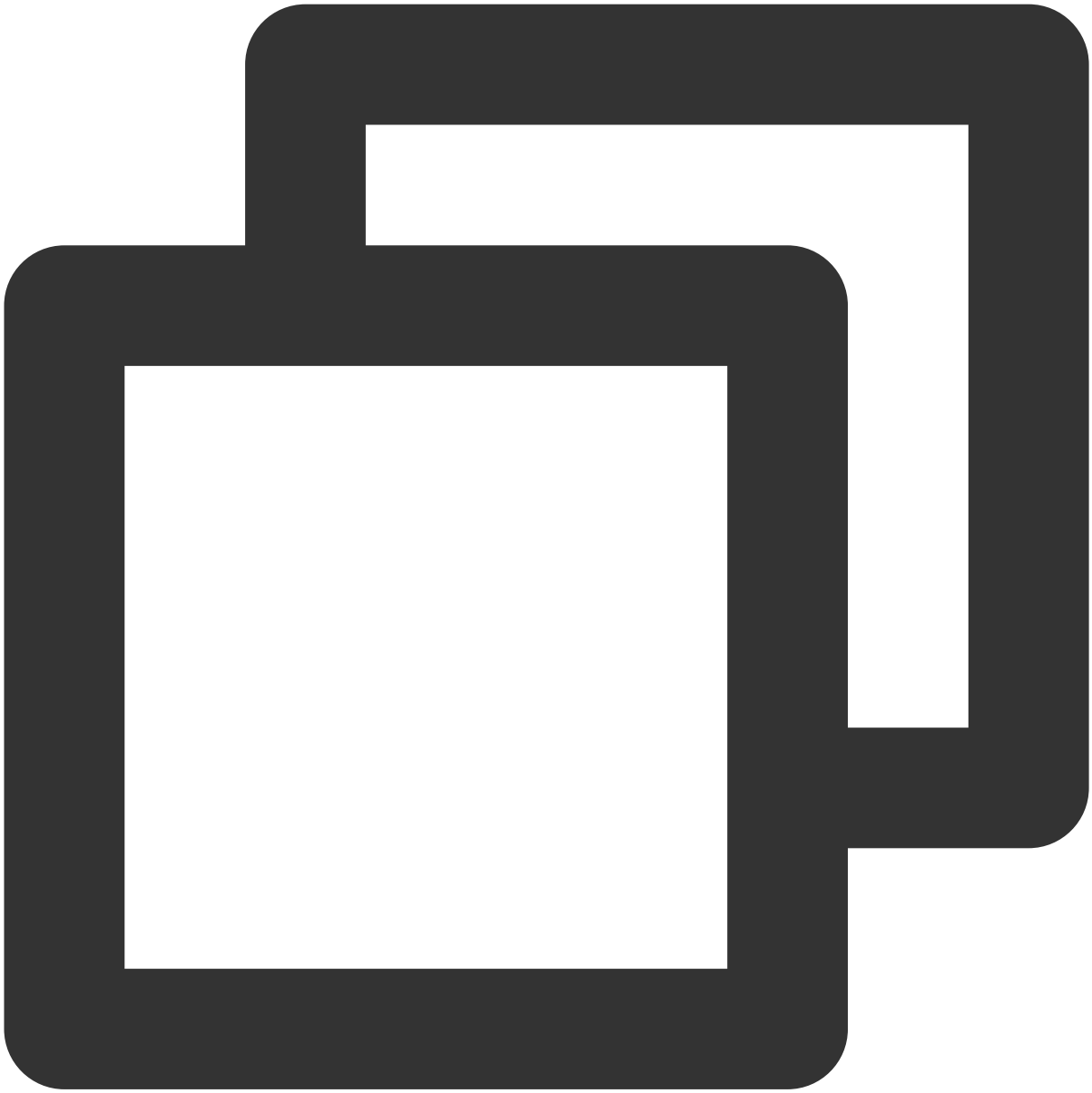
viteを起動ツールとして使用する場合、viteのインポートの問題に注意してください

ご注意：

ここで言及するUserSigの取得方法は、フロントエンドコードにSECRETKEYを設定しますが、この手法のSECRETKEYは逆コンパイルによって逆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、ローカルのクイックスタート機能のデバッグにのみ適しています**。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のイン

ターフェース向けに提供することです。 UserSigが必要なときは、Appから業務サーバーにリクエストを送信し動的にUserSigを取得します。詳細については、サーバーでのUserSig新規作成をご参照ください。

2.3 電話をかけたい場所で、次を実行します。



```
import { TUICallKitServer } from './components/TUICallKit/Web';
TUICallKitServer.call({ userID: "123", type: 2 }); // 2人通話
TUICallKitServer.groupCall({ userIDList: ["xxx"], groupID: "xxx", type: 2 }); // 多
```

その後すぐに最初の電話をかけることができます。より詳細なインターフェースパラメータについては、インターフェースドキュメントをご参照ください

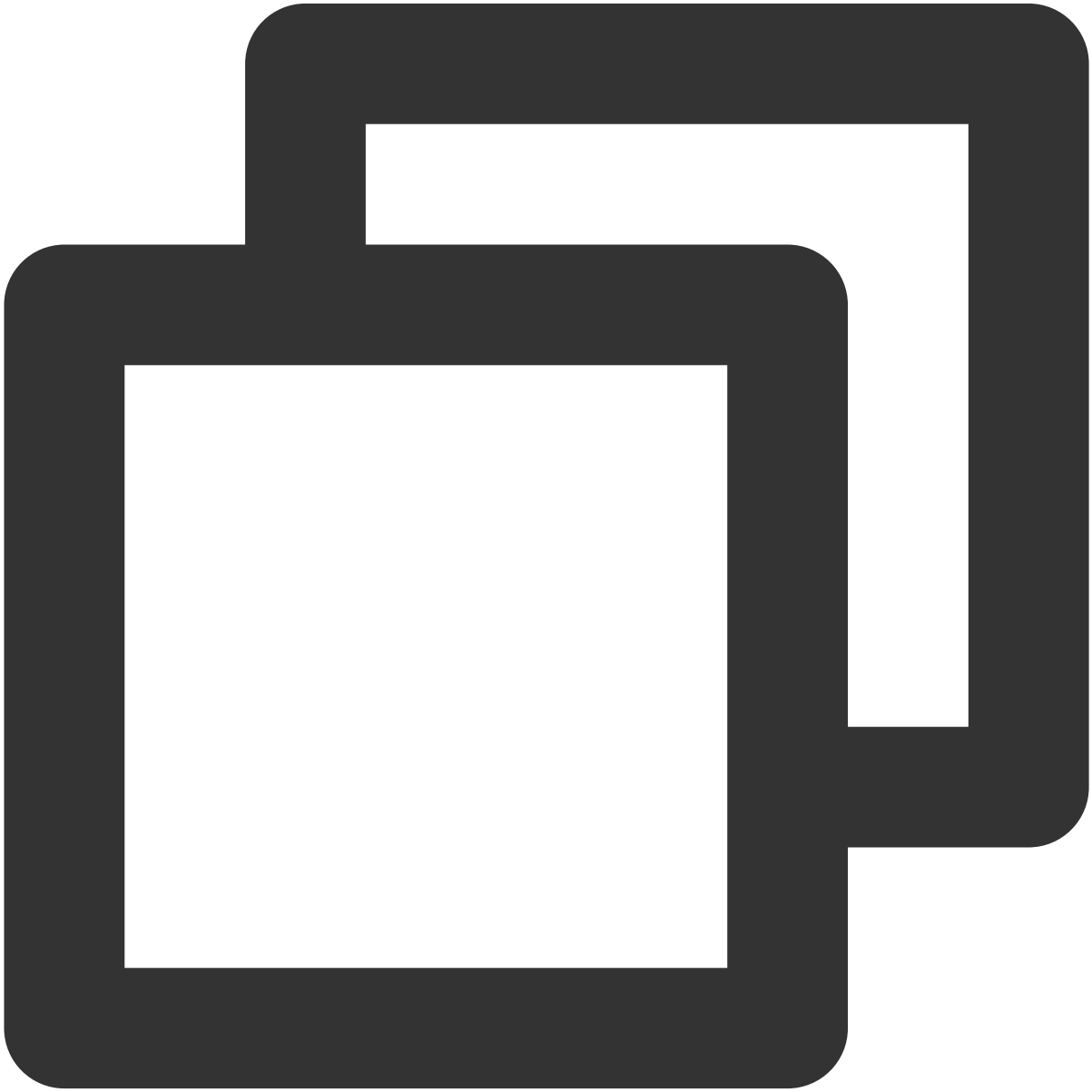
3. 発展インターフェース

このコンポーネントでは `beforeCalling` および `afterCalling` という2つのコールバックを提供しており、業務側に現在の通話状態を通知するために用いられます。

`beforeCalling` : 通話前に実行します

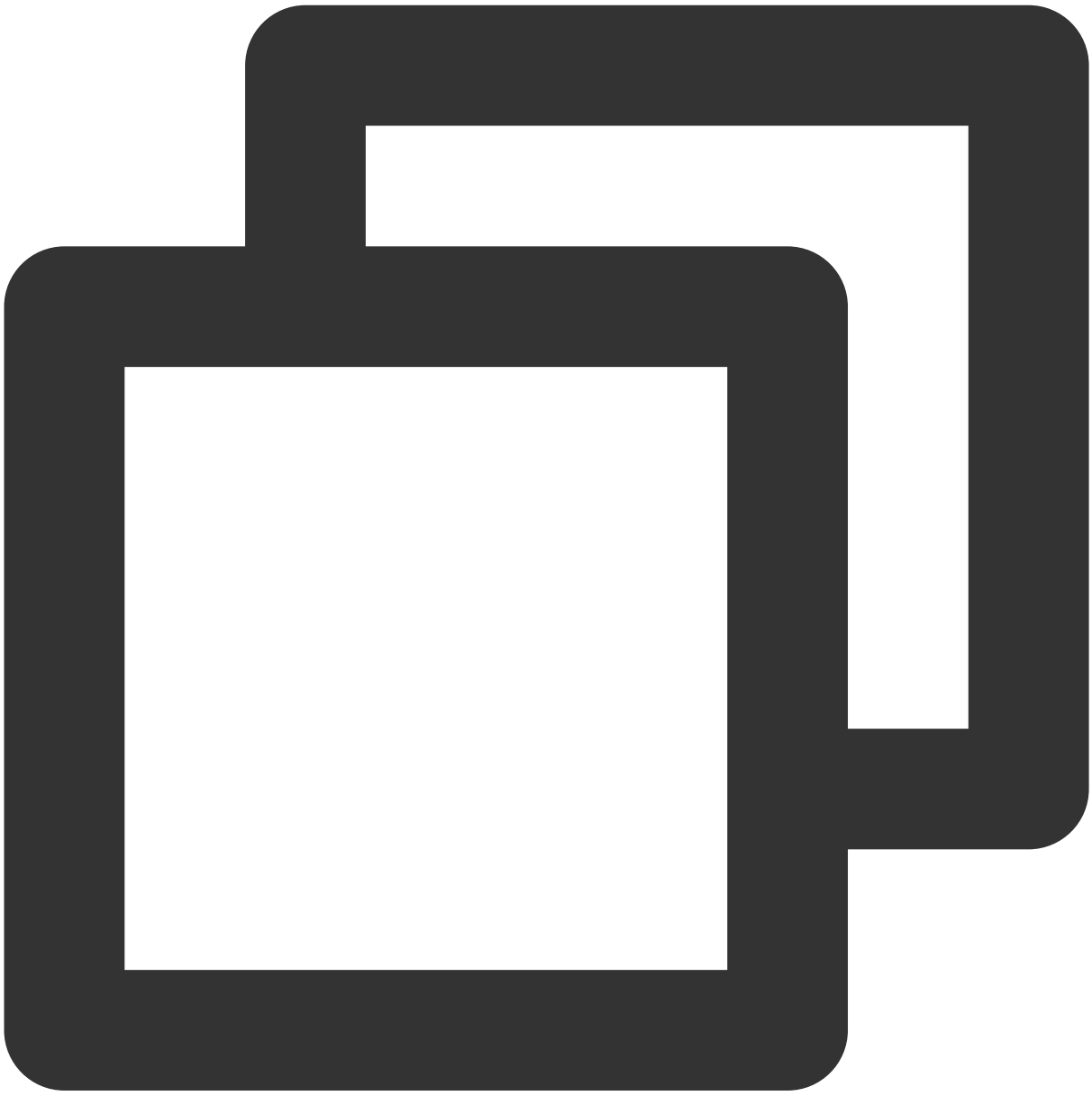
`afterCalling` : 通話後に実行します

例えばサンプルdemoのように、 `<TUICallKit />` コンポーネントの表示と折りたたみに用いることができます。



```
function beforeCalling() {
  console.log("通話前にこの関数を実行");
```

```
}
function afterCalling() {
  console.log("通話後にこの関数を実行");
}
```



```
<TUICallKit :beforeCalling="beforeCalling" :afterCalling="afterCalling"/>
```

その他のドキュメント

TUICallKit API
TUICallKit demoクイックスタート
TUICallKitインターフェースのカスタマイズガイド
TUICallKit (Web) に関するよくあるご質問

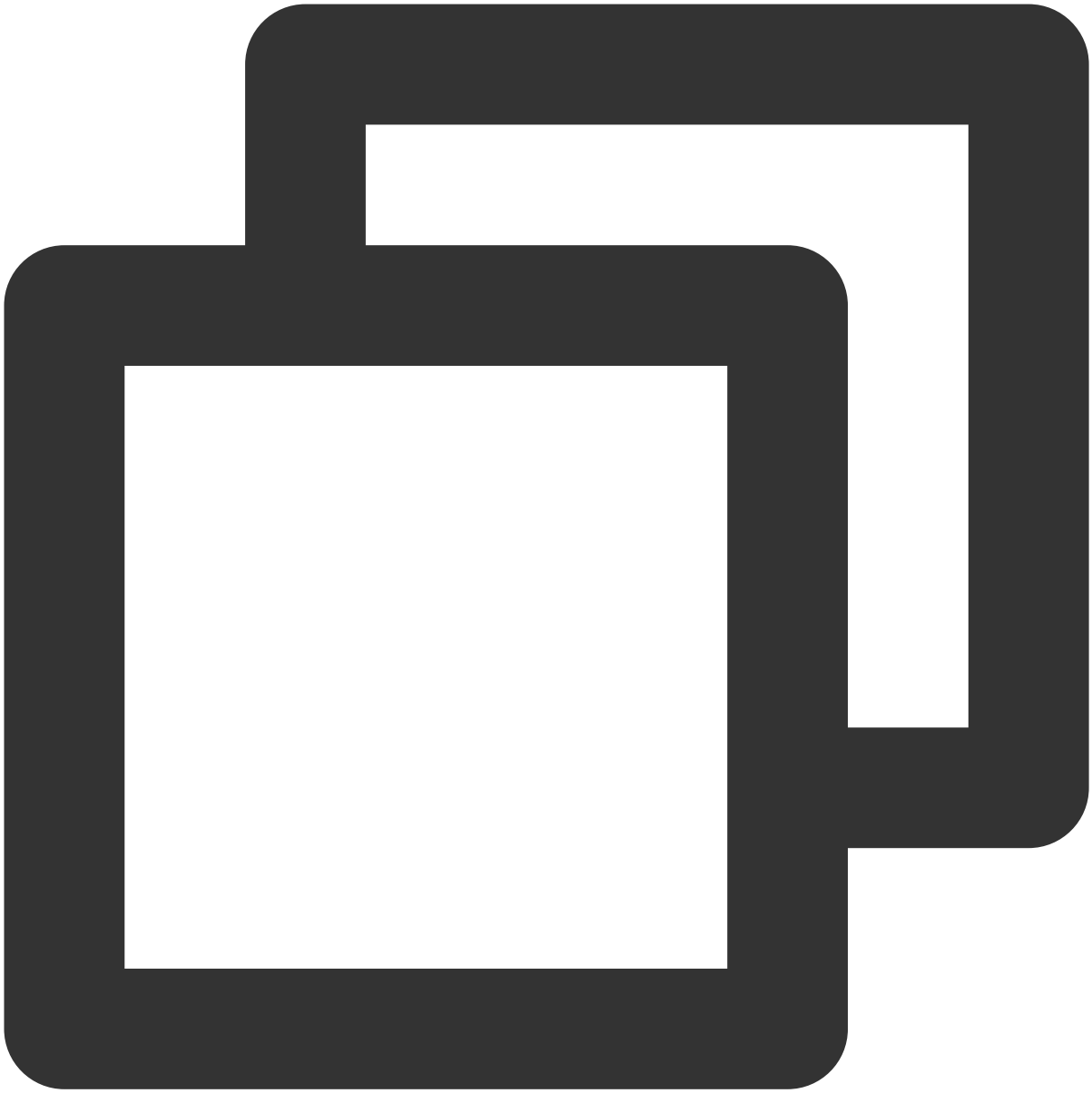
よくあるご質問

1. UserSigはどのように生成しますか。

UserSigの発行方法は、UserSigの計算コードをサーバーに統合し、プロジェクトのインターフェース向けに提供する方法となります。UserSigが必要な時は、プロジェクトから業務サーバーにリクエストを送信し動的にUserSigを取得します。詳細については、サーバーでのUserSig新規作成をご参照ください。

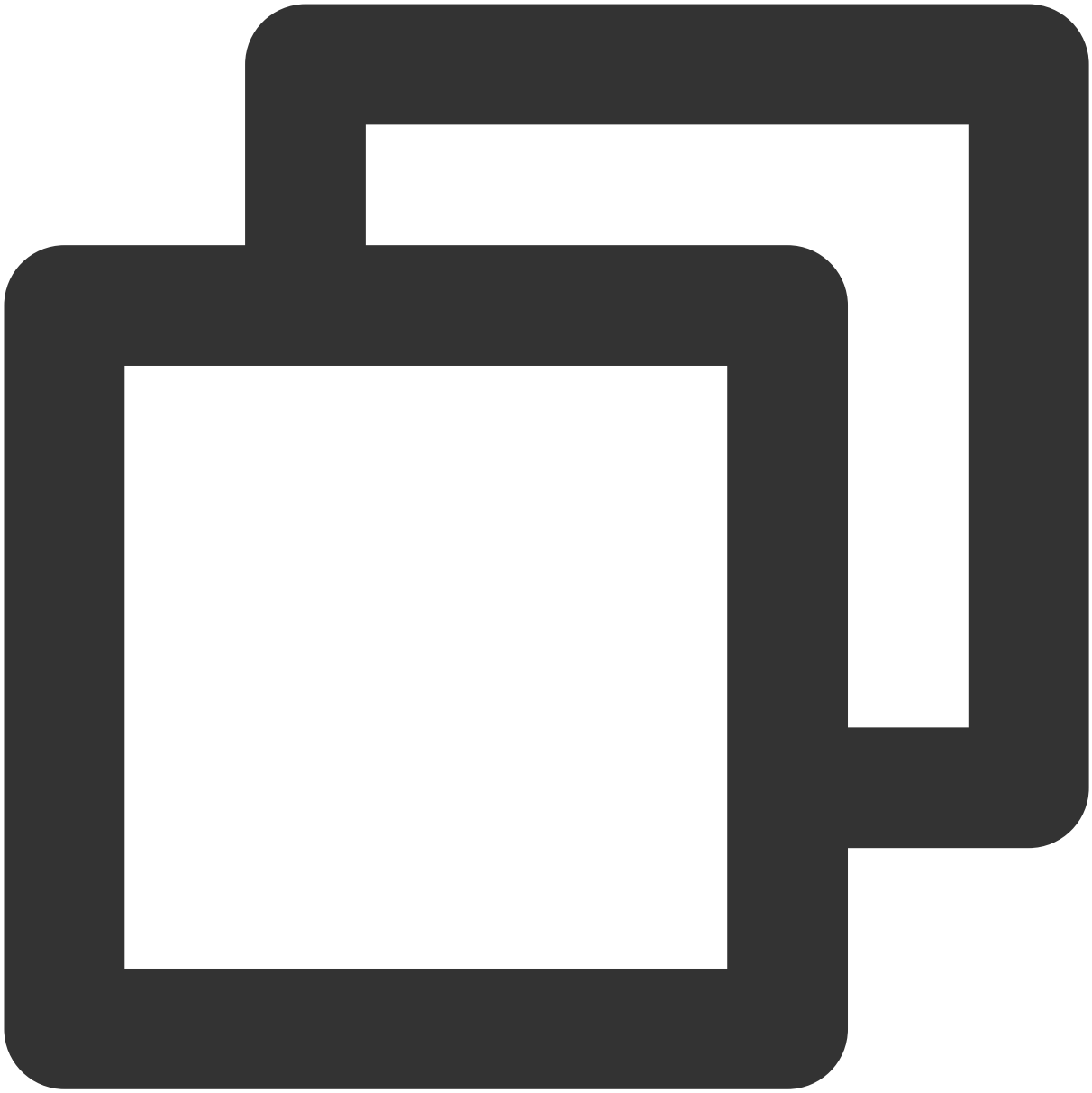
2. viteのインポートの問題

プロジェクトをviteで作成した場合は、ファイルパッケージの違いにより、 `index.html` に `lib-generate-test-usersig.min.js` をインポートする必要があります



```
// index.html
<script src="./src/components/TUICallKit/Web/demos/basic/public/debug/lib-generate-
```

また、元の `GenerateTestUserSig.js` にimportしたメソッドのアノテーションを行います



```
// import * as LibGenerateTestUserSig from './lib-generate-test-usersig.min.js'
```

3. 環境要件

(1) ブラウザのバージョン要件

最新バージョンのChromeブラウザを使用して体験してください。このドキュメントで連結するコンポーネントの、現在の主流デスクトップブラウザに対するサポート状況は次のとおりです。

オペレーティングシステム	ブラウザタイプ	ブラウザの最小バージョン要件

Mac OS	デスクトップ版Safariブラウザ	11+
Mac OS	デスクトップ版Chromeブラウザ	56+
Mac OS	デスクトップ版Firefoxブラウザ	56+
Mac OS	デスクトップ版Edgeブラウザ	80+
Windows	デスクトップ版Chromeブラウザ	56+
Windows	デスクトップ版QQブラウザ（クイックコア）	10.4+
Windows	デスクトップ版Firefoxブラウザ	56+
Windows	デスクトップ版Edgeブラウザ	80+

説明：
詳細な互換性の照会については、ブラウザサポート状況をご参照ください。また、TRTC検査ページでオンライン検査することも可能です。

(2) ネットワーク環境の要件

TUICallKitを使用する際、ユーザーはファイアウォールの制限によって正常にオーディオビデオ通話が行えない可能性があります。ファイアウォール制限の対応関連 をご参照の上、対応するポートおよびドメイン名をファイアウォールのホワイトリストに追加してください。

(3) ウェブサイトドメイン名プロトコルの要件

ユーザーのセキュリティ、プライバシーなどの問題を考慮し、ブラウザは本ドキュメントの連結コンポーネントの全機能をHTTPSプロトコルでしか正常に使用できないようにウェブページを制限しています。本番環境のユーザーが製品機能をスムーズに体験できるようにするため、ウェブサイトはhttps:// プロトコルのドメイン名にデプロイしてください。

ユースケース	プロトコル	受信（再生）	送信（マイク・オン）	画面共有	備考
本番環境	HTTPSプロトコル	サポートしています	サポートしています	サポートしています	推奨
本番環境	HTTPプロトコル	サポートしています	サポートしていません	サポートしていません	-
ローカル開発環境	http://localhost	サポートしています	サポートしています	サポートしています	推奨

ローカル開発環境	http://127.0.0.1	サポートしています	サポートしています	サポートしています	-
ローカル開発環境	http://[ローカルマシンIP]	サポートしています	サポートしていません	サポートしていません	-
ローカル開発環境	file:///	サポートしています	サポートしています	サポートしています	-

インターフェースのカスタマイズ（TUICallKit）

Android

最終更新日：：2024-07-19 14:53:21

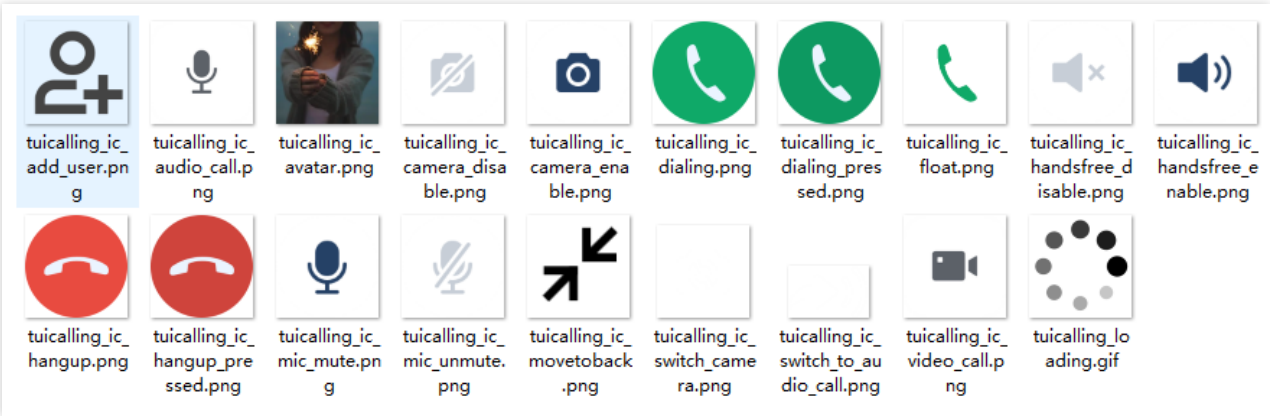
ここではTUICallKitのユーザーインターフェースをカスタマイズする方法についてご説明します。**インターフェース微調整**と**自身でのUI実装**という2つのソリューションをご用意しています。

方法1：インターフェース微調整ソリューション

ご提供するUIソースコードを直接変更することで、TUICallKitのユーザーインターフェースを調整します。
TUICallKitのインターフェースソースコードはGithubの `Android/tuicallkit` フォルダにあります。

アイコンの置き換え

`res\\drawable-xxhdpi` フォルダ内のアイコンを直接置き換え、App全体のアイコンの色やスタイルに統一性を持たせることができます。置き換える際にアイコンファイルの名前を変更しないようにしてください。



着信音の置き換え

`res\\raw` フォルダ内の3つのオーディオファイルを置き換えることで、着信音の置き換えができます。

ファイル名	用途
phone_dialing.mp3	呼び出し開始時の音声
phone_hangup.mp3	通話終了の音声

phone_ringing.mp3	呼び出し応答時の音声

テキストの置き換え

values-zhおよびvalues-en内の `strings.xml` ファイルを変更することで、ビデオ通話画面内の文字列の内容を変更できます。

方法2：自身でのUI実装ソリューション

TUICallKitの通話機能全体はTUICallEngineというUIレスコンポーネントをベースにして実装されています。tuicallkitフォルダを削除し、TUICallEngineを完全にベースにしてご自身のUIを実装することが可能です。

TUICallEngine

TUICallEngine APIは通話コンポーネント全体の基盤インターフェースです。主に1対1オーディオビデオ通話およびグループ内通話の開始、応答、拒否、終了、デバイス操作などの重要なインターフェースを提供します。

API	説明
createInstance	TUICallEngineインスタンスの作成（シングルトンモード）
destroyInstance	TUICallEngineインスタンスの破棄（シングルトンモード）
init	オーディオビデオ通話基本機能の認証完了
addObserver	イベントコールバックの追加
removeObserver	コールバックインターフェースの削除
call	1v1通話の開始
groupCall	グループ通話の開始
accept	通話応答
reject	通話拒否
hangup	通話終了
ignore	通話を無視
inviteUser	グループ通話中に他の人を招待
joinInGroupCall	現在のグループ通話に自主的に参加
switchCallMediaType	通話メディアタイプの切り替え。ビデオ通話からオーディオ通話への切り替

	えなど
startRemoteView	リモートユーザービデオストリームのサブスクリプション開始
stopRemoteView	リモートユーザービデオストリームのサブスクリプション停止
openCamera	カメラの起動
closeCamera	カメラの終了
switchCamera	フロント/リアカメラの切り替え
openMicrophone	マイクをオンにする
closeMicrophone	マイクをオフにする
selectAudioPlaybackDevice	オーディオ再生デバイスの選択（ヘッドホン/スピーカー）
setSelfInfo	ユーザーのニックネーム、プロフィール画像の設定
enableMultiDeviceAbility	TUICallEngineのマルチデバイスログインモードのオン/オフ （プレミアム版パッケージのみサポート）

TUICallObserver

TUICallObserverはTUICallEngineに対応するコールバックイベントクラスです。このコールバックによって、関心のあるコールバックイベントを監視することができます。

API	説明
onError	通話中のエラーコールバック
onCallReceived	通話リクエストのコールバック
onCallCancelled	通話キャンセルのコールバック
onCallBegin	通話接続のコールバック
onCallEnd	通話終了のコールバック
onCallMediaTypeChanged	通話メディアタイプ変更発生のコールバック
onUserReject	xxxxユーザーによる通話拒否のコールバック
onUserNoResponse	xxxxユーザーの応答なしのコールバック
onUserLineBusy	xxxxユーザーが通話中である場合のコールバック
onUserJoin	xxxxユーザーの通話参加のコールバック

onUserLeave	xxxxユーザーの通話からの退出のコールバック
onUserVideoAvailable	xxxユーザーのビデオストリームの有無のコールバック
onUserAudioAvailable	xxxユーザーのオーディオストリームの有無のコールバック
onUserVoiceVolumeChanged	全ユーザーの音量レベルフィードバックのコールバック
onUserNetworkQualityChanged	全ユーザーのネットワーク品質フィードバックのコールバック

主要なタイプの定義

API	説明
TUICallDefine.MediaType	通話のメディアタイプ。列挙タイプ：ビデオ通話、音声通話
TUICallDefine.Role	通話のロール。列挙タイプ：発呼側、着呼側
TUICallDefine.Status	通話の状態。列挙タイプ：アイドル状態、応答待ち、応答中
TUICommonDefine.RoomId	オーディオビデオルームID。数字、文字列の2種類をサポートしています
TUICommonDefine.Camera	カメラIDパラメータ。列挙タイプ：フロントカメラ、リアカメラ
TUICommonDefine.AudioPlaybackDevice	音声再生デバイス。列挙タイプ：スピーカー、ヘッドホン
TUICommonDefine.NetworkQualityInfo	現在のネットワーク品質情報

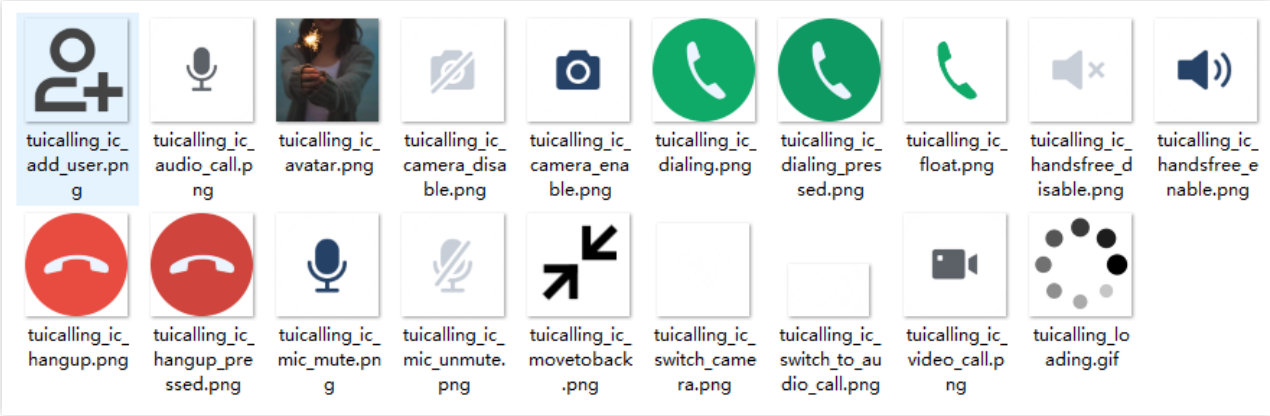
iOS

最終更新日：：2024-07-19 14:53:21

ここではTUICallKitのユーザーインターフェースをカスタマイズする方法についてご説明します。**インターフェース微調整**と**自身でのUI実装**という2つのソリューションをご用意しています。

方法1：インターフェース微調整ソリューション

ご提供するUIソースコードを直接変更することで、TUICallKitのユーザーインターフェースを調整します。
TUICallKitのインターフェースソースコードはGithubの `iOS/TUICallKit` フォルダにあります。
アイコンの置き換え `Resources\\Calling.xcassets` フォルダ内のアイコンを直接置き換え、app全体のアイコンの色やスタイルに統一性を持たせることができます。置き換える際にアイコンファイルの名前を変更しないようにしてください。



着信音の置き換え `Resources\\AudioFile` フォルダ内の3つのオーディオファイルを置き換えることで、着信音の置き換えができます。

ファイル名	用途
phone_dialing.m4a	呼び出し開始時の音声
phone_hangup.mp3	通話終了の音声
phone_ringing.flac	呼び出し応答時の音声

テキストの置き換え zh-Hans.lprojおよびen.lproj内の `CallingLocalized.strings` ファイルを変更することで、ビデオ通話画面内の文字列の内容を変更できます。

方法2：自身でのUI実装ソリューション

TUICallKitの通話機能全体はTUICallEngineというUIレスコンポーネントをベースにして実装されています。tuicallkitフォルダを削除し、TUICallEngineを完全にベースにしてご自身のUIを実装することが可能です。

TUICallEngine

TUICallEngine APIは通話コンポーネント全体の基盤インターフェースです。主に1対1オーディオビデオ通話およびグループ内通話の開始、応答、拒否、終了、デバイス操作などの重要なインターフェースを提供します。

API	説明
createInstance	TUICallEngineインスタンスの作成（シングルトンモード）
destroyInstance	TUICallEngineインスタンスの破棄（シングルトンモード）
init	オーディオビデオ通話基本機能の認証完了
addObserver	イベントコールバックの追加
removeObserver	コールバックインターフェースの削除
call	1v1通話の開始
groupCall	グループ通話の開始
accept	通話応答
reject	通話拒否
hangup	通話終了
ignore	通話を無視
inviteUser	グループ通話中に他の人を招待
joinInGroupCall	現在のグループ通話に自主的に参加
switchCallMediaType	通話メディアタイプの切り替え。ビデオ通話からオーディオ通話への切り替えなど
startRemoteView	リモートユーザービデオストリームのサブスクリプション開始
stopRemoteView	リモートユーザービデオストリームのサブスクリプション停止
openCamera	カメラの起動
closeCamera	カメラの終了

switchCamera	フロント/リアカメラの切り替え
openMicrophone	マイクをオンにする
closeMicrophone	マイクをオフにする
selectAudioPlaybackDevice	オーディオ再生デバイスの選択（ヘッドホン/スピーカー）
setSelfInfo	ユーザーのニックネーム、プロフィール画像の設定
enableMultiDeviceAbility	TUICallEngineのマルチデバイスログインモードのオン/オフ （プレミアム版パッケージのみサポート）

TUICallObserver

TUICallObserverはTUICallEngineに対応するコールバックイベントクラスです。このコールバックによって、関心のあるコールバックイベントを監視することができます。

API	説明
onError	通話中のエラーコールバック
onCallReceived	通話リクエストのコールバック
onCallCancelled	通話キャンセルのコールバック
onCallBegin	通話接続のコールバック
onCallEnd	通話終了のコールバック
onCallMediaTypeChanged	通話メディアタイプ変更発生のコールバック
onUserReject	xxxxユーザーによる通話拒否のコールバック
onUserNoResponse	xxxxユーザーの応答なしのコールバック
onUserLineBusy	xxxxユーザーが通話中である場合のコールバック
onUserJoin	xxxxユーザーの通話参加のコールバック
onUserLeave	xxxxユーザーの通話からの退出のコールバック
onUserVideoAvailable	xxxユーザーのビデオストリームの有無のコールバック
onUserAudioAvailable	xxxユーザーのオーディオストリームの有無のコールバック
onUserVoiceVolumeChanged	全ユーザーの音量レベルフィードバックのコールバック
onUserNetworkQualityChanged	全ユーザーのネットワーク品質フィードバックのコールバック

主要なタイプの定義

API	説明
TUICallMediaType	通話のメディアタイプ。列挙タイプ：ビデオ通話、音声通話
TUICallRole	通話のロール。列挙タイプ：発呼側、着呼側
TUICallStatus	通話の状態。列挙タイプ：アイドル状態、応答待ち、応答中
TUIRoomId	オーディオビデオルームID。数字、文字列の2種類をサポートしています
TUICallCamera	カメラIDパラメータ。列挙タイプ：フロントカメラ、リアカメラ
TUIAudioPlaybackDevice	音声再生デバイス。列挙タイプ：スピーカー、ヘッドホン
TUINetworkQualityInfo	現在のネットワーク品質情報

Web

最終更新日：：2024-07-19 14:53:21

TUICallKitインターフェースのカスタマイズガイド

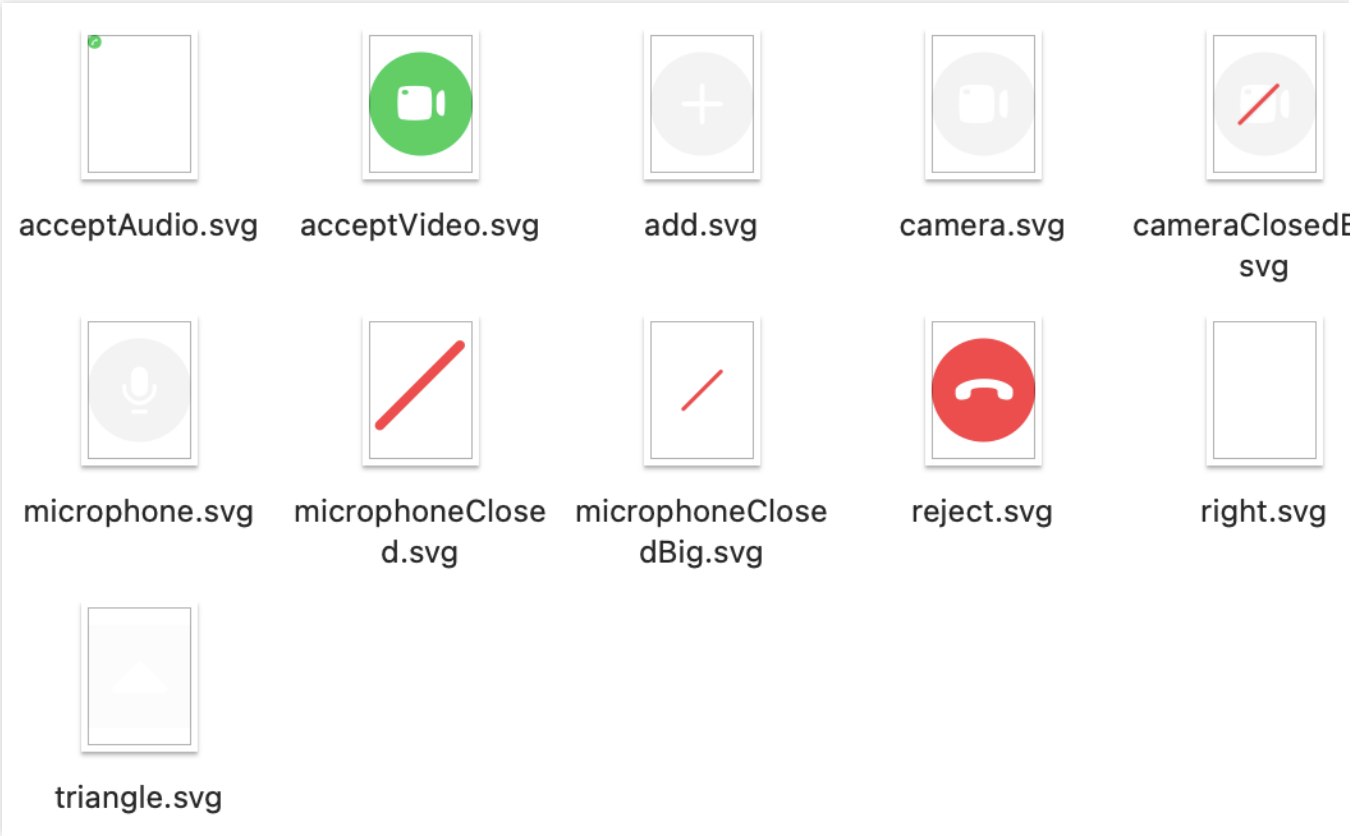
ここではTUICallKitのユーザーインターフェースをカスタマイズする方法についてご説明します。インターフェース微調整と自身でのUI実装という2つのソリューションをご用意しています。

方法1：インターフェース微調整ソリューション

ご提供するUIソースコードを直接変更することで、TUICallKitのユーザーインターフェースを調整します。
TUICallKitのインターフェースソースコードはGithubの `Web/` フォルダにあります。

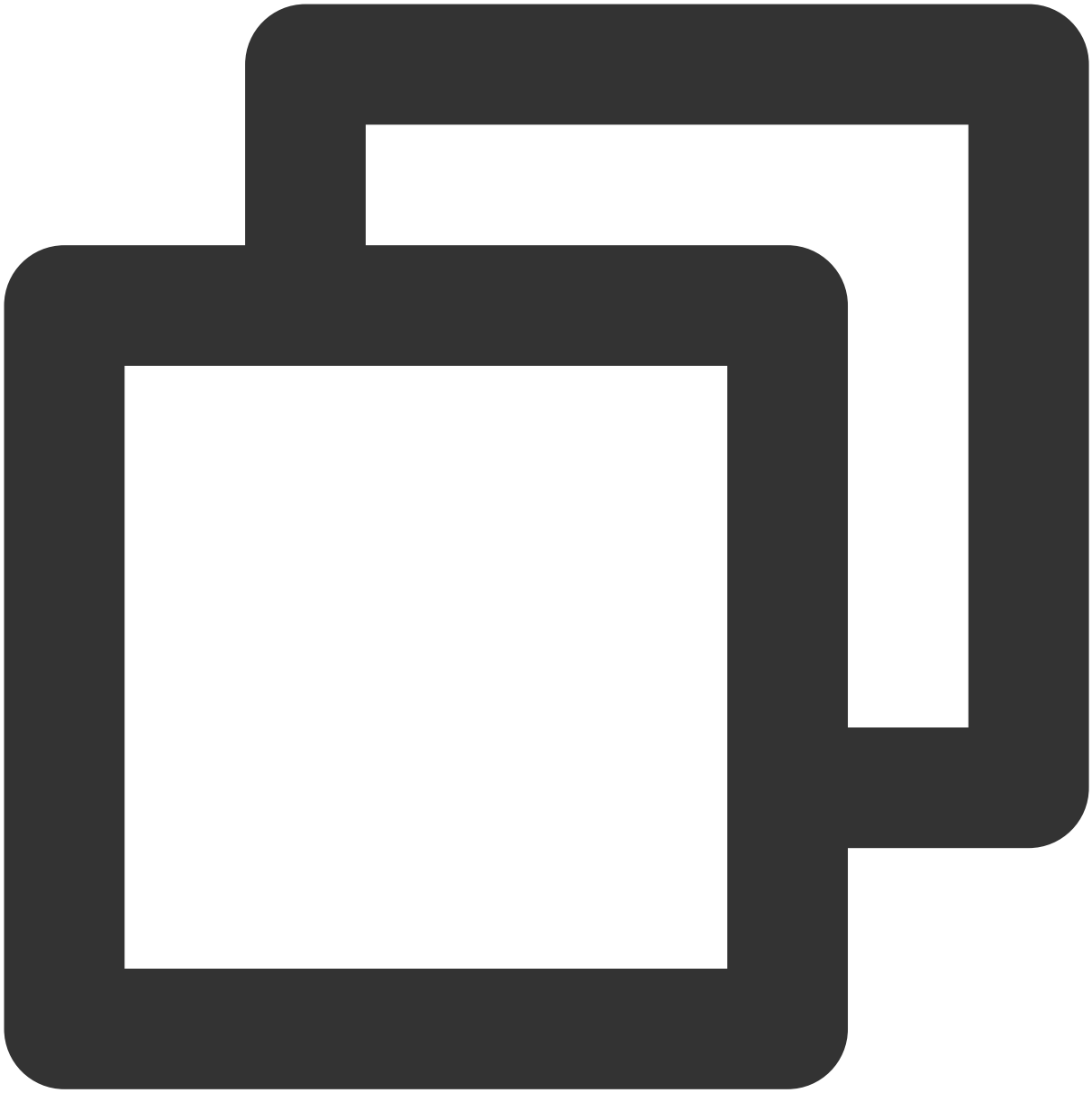
アイコンの置き換え

`src/icons` フォルダ内のアイコンコンポーネントを直接変更し、app全体のアイコンの色やスタイルに統一性を持たせることができます。置き換える際にアイコンファイルの名前を変更しないようにしてください。アイコンのプレビューは `src/assets` で参照できます。



UIレイアウトの調整

src/components/ファイルのそれぞれのページを変更することで、オーディオビデオ通話インターフェースを変更できます



```
- components/
    - Calling-C2CVideo.vue          2人ビデオ通話
    - Calling-Group.Vue             多人数オーディオ・ビデオ通話
    - ControlPanel.vue              コントロールパネル
    - ControlPanelItem.vue          コントロールパネルサブ項目
    - Dialing.vue                   電話発信ページ、着信ページ、2人オーディオ通話
    - MicrophoneIcon.vue            音量の変化を表示できるマイクIcon
    - TUICallKit.vue                TUICallKit全体コンポーネント
```

スタイルの変更

スタイルファイルは `src/style.css` にあります。UIのスタイルに合わせてご自身で調整できます。

方法2：自身でのUI実装ソリューション

TUICallKitのすべての通話機能は、TUICallEngineというUIレスSDKをベースにして実装されており、TUICallEngineを完全にベースにしてご自身のUIを実装することが可能です。詳細については以下をご参照ください

TUICallEngineアクセスガイド

TUICallEngine APIインターフェースアドレス

オフライン通知（TUICallKit）
Android

最終更新日：：2024-07-19 14:53:21

オフライン通知機能は、アプリケーションが「バックエンドで実行中」または「オフライン状態」の場合でも、オーディオビデオ通話の着信を受信できる機能です。TUICallKitはTUIOfflinePushコンポーネントを統合することでオフライン通知機能を実現しています。ここでは、オーディオビデオ通話プロジェクトで `TUIOfflinePush` コンポーネントを統合する方法についてご説明します。

事前準備

1. **アプリケーションをメーカーのプッシュプラットフォームに登録する** オフラインプッシュ機能はメーカーオリジナルのチャネルに依存します。ご自身のアプリケーションを各メーカーのプッシュプラットフォームに登録し、APPIDおよびAPPKEYなどのパラメータを取得する必要があります。

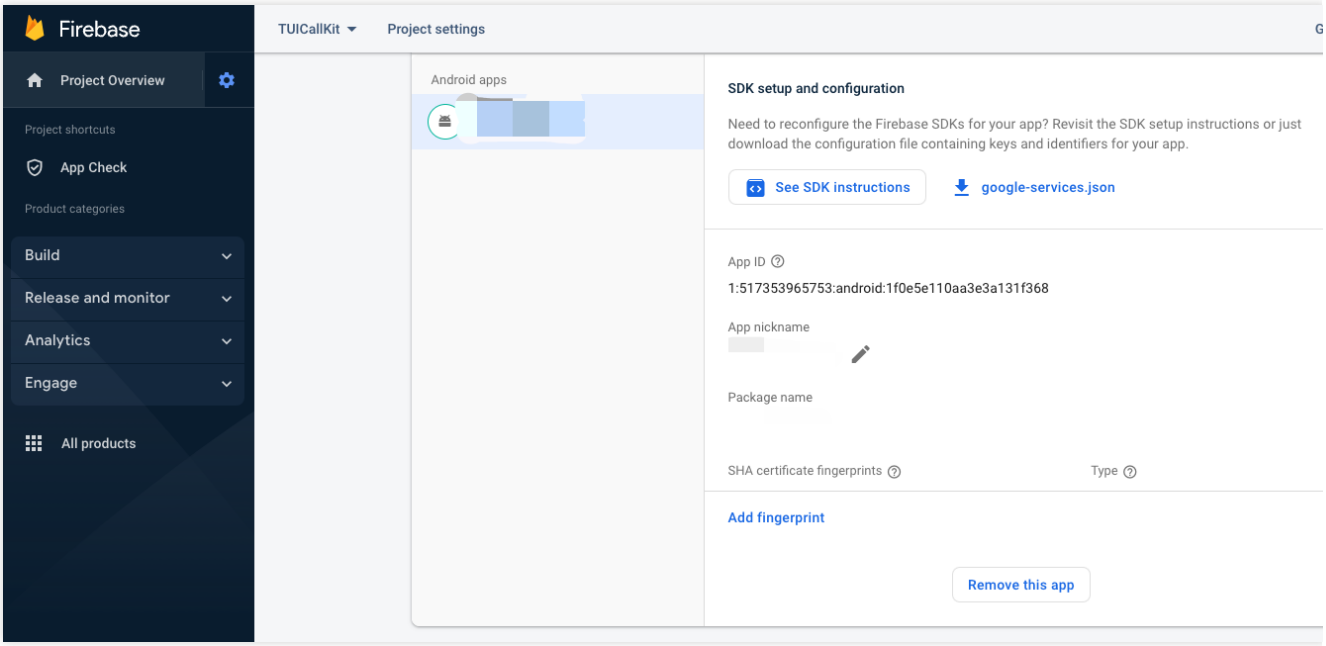
ご注意：

次の2つのファイルはこの後の手順で使用します

Huaweiプラットフォームに登録する場合は、 `agconnect-services.json` ファイルをダウンロードして保存します。

Googleプラットフォームに登録する場合は、 `google-services.json` ファイルをダウンロードして保存します。

Google FCM

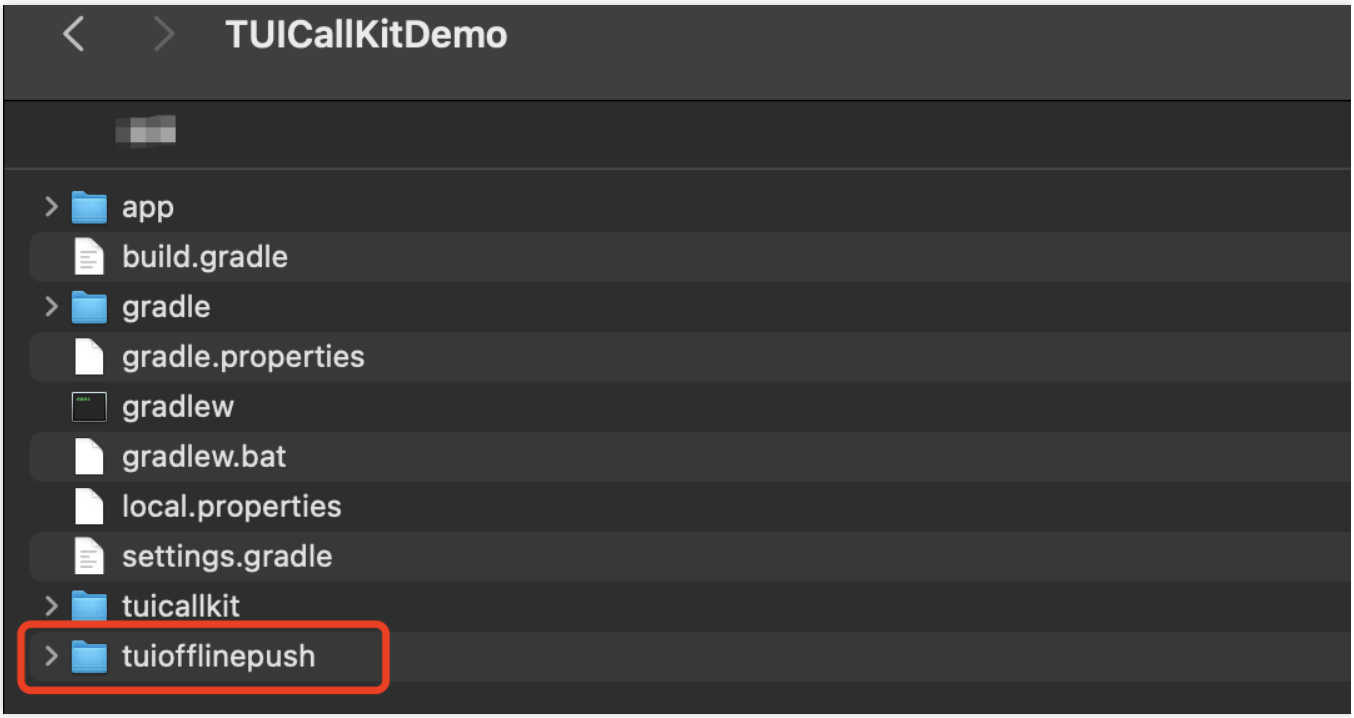


2. **IMコンソールで設定を行う**

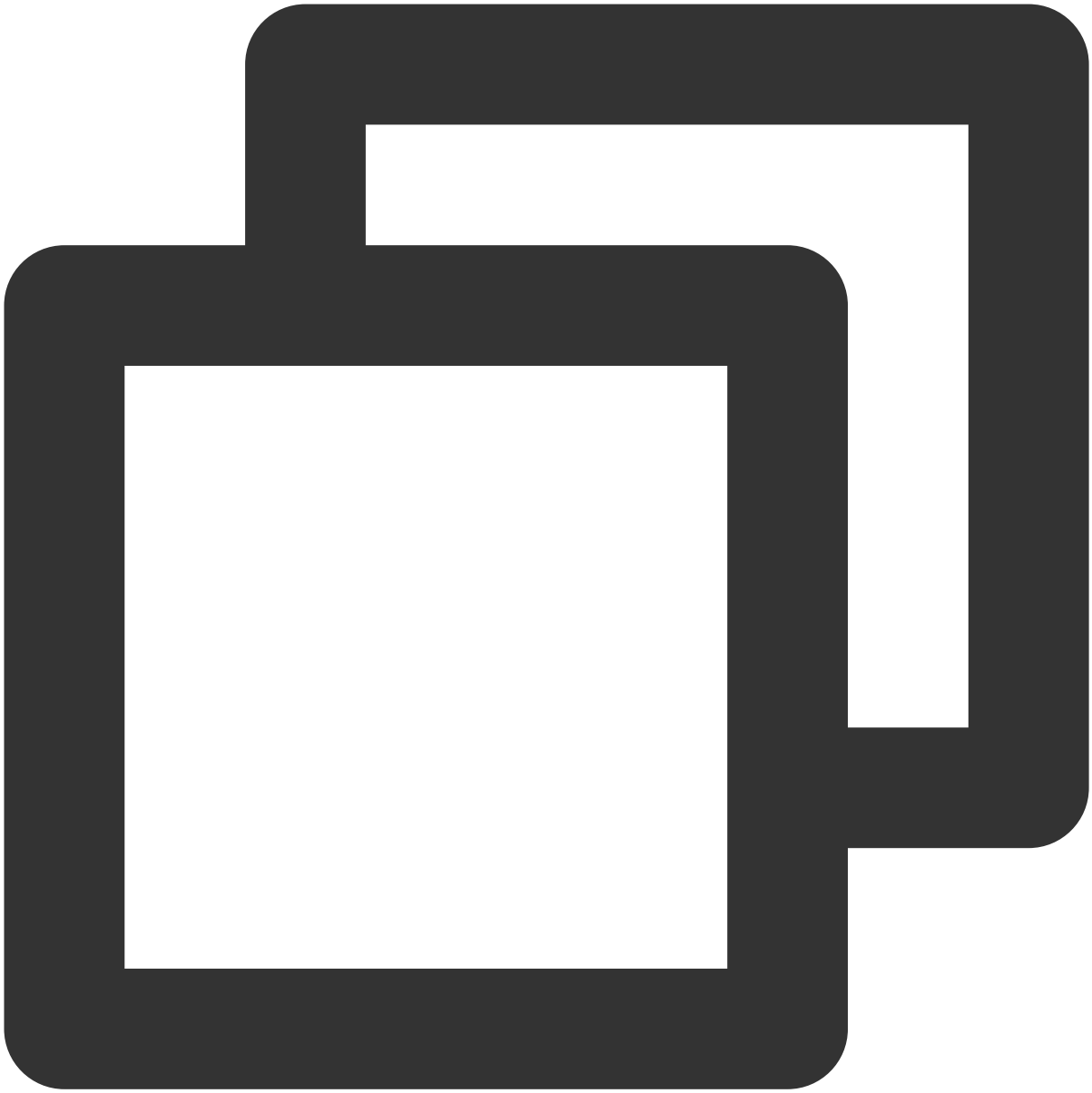
メーカーのチャネルに登録するにはご自身のパッケージ名を渡す必要があります。メッセージの相互運用のため、各メーカーで入力するパッケージ名は一致させる必要があります。生成したID、APPIDおよびAPPKEYを記録します。これらはこの後の手順で使用します。

ステップ1：コンポーネントのダウンロードとインポート

1. Githubでコードをクローン/ダウンロードした後、下図のように、Androidディレクトリ下のtuiofflinepushサブディレクトリを現在のプロジェクトのappの同階層のディレクトリにコピーします。

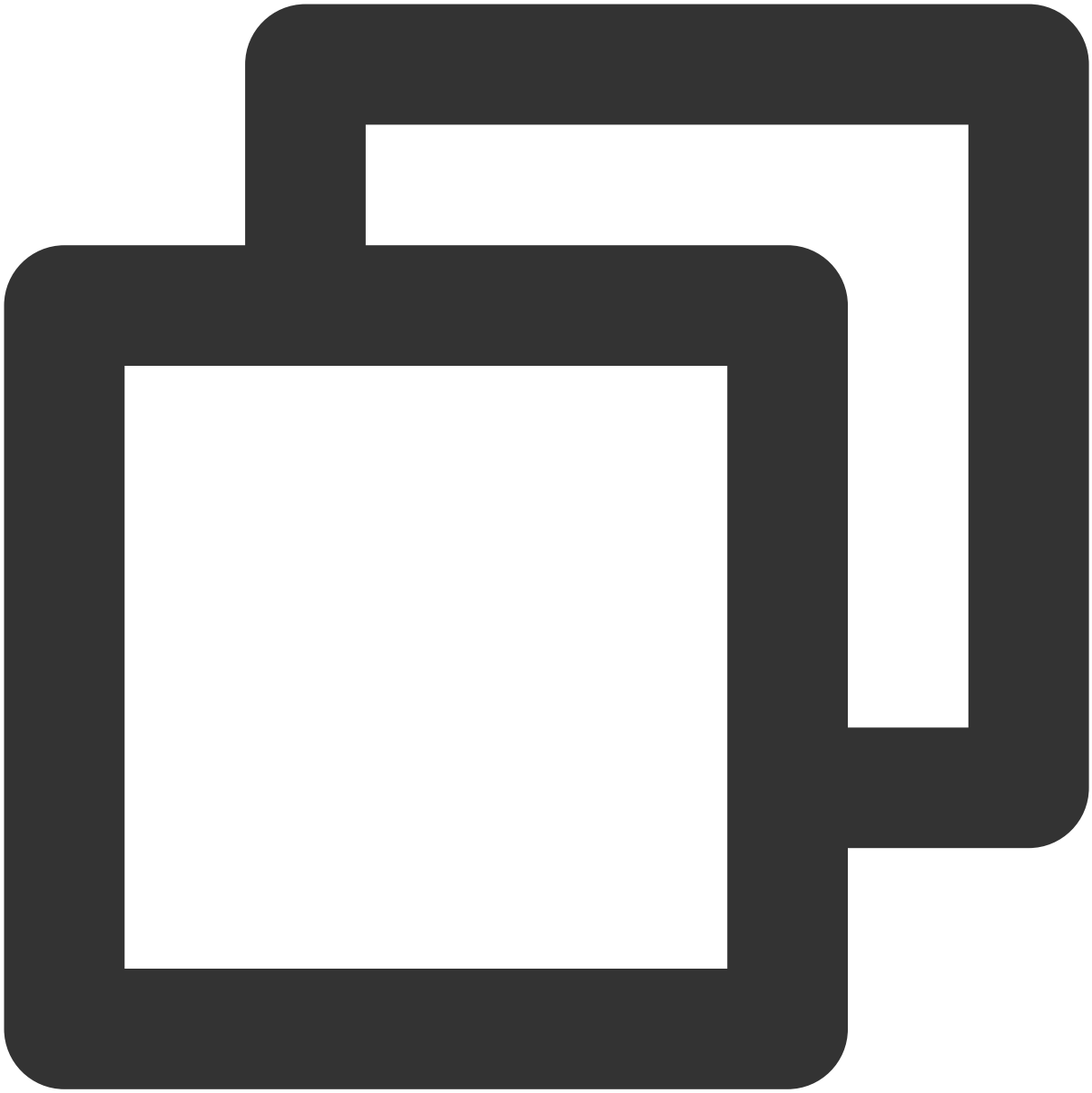


2. プロジェクトのルートディレクトリ下で `setting.gradle` ファイルを見つけ、その中に次のコードを追加します。



```
include ':tuiofflinepush'
```

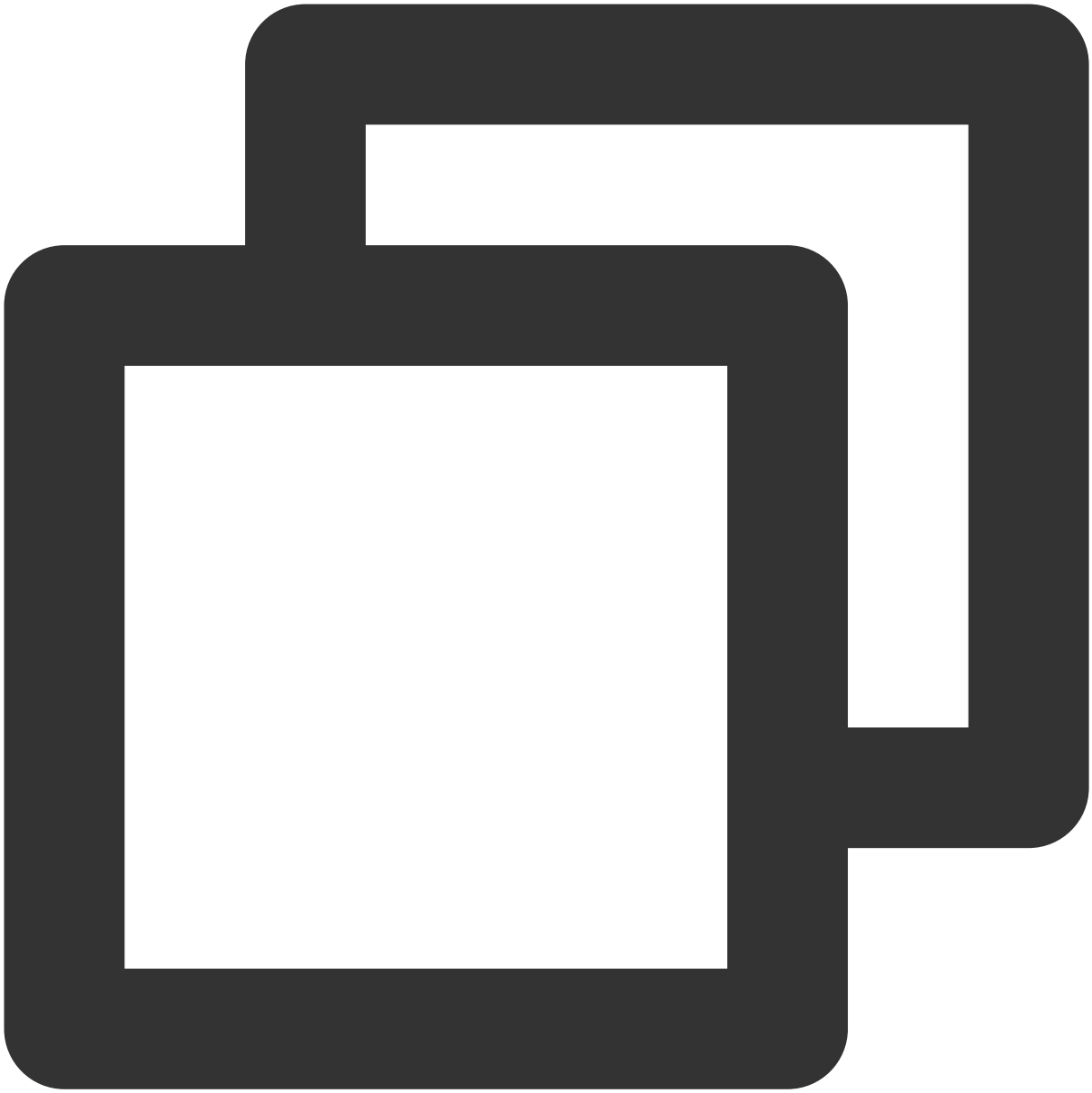
3. appディレクトリ下で `build.gradle` ファイルを見つけ、その中に次のコードを追加します。その役割は、現在のappの新たに追加したtuiofflinepushコンポーネントへの依存を宣言するものです。



```
api project(':tuiofflinepush')
```

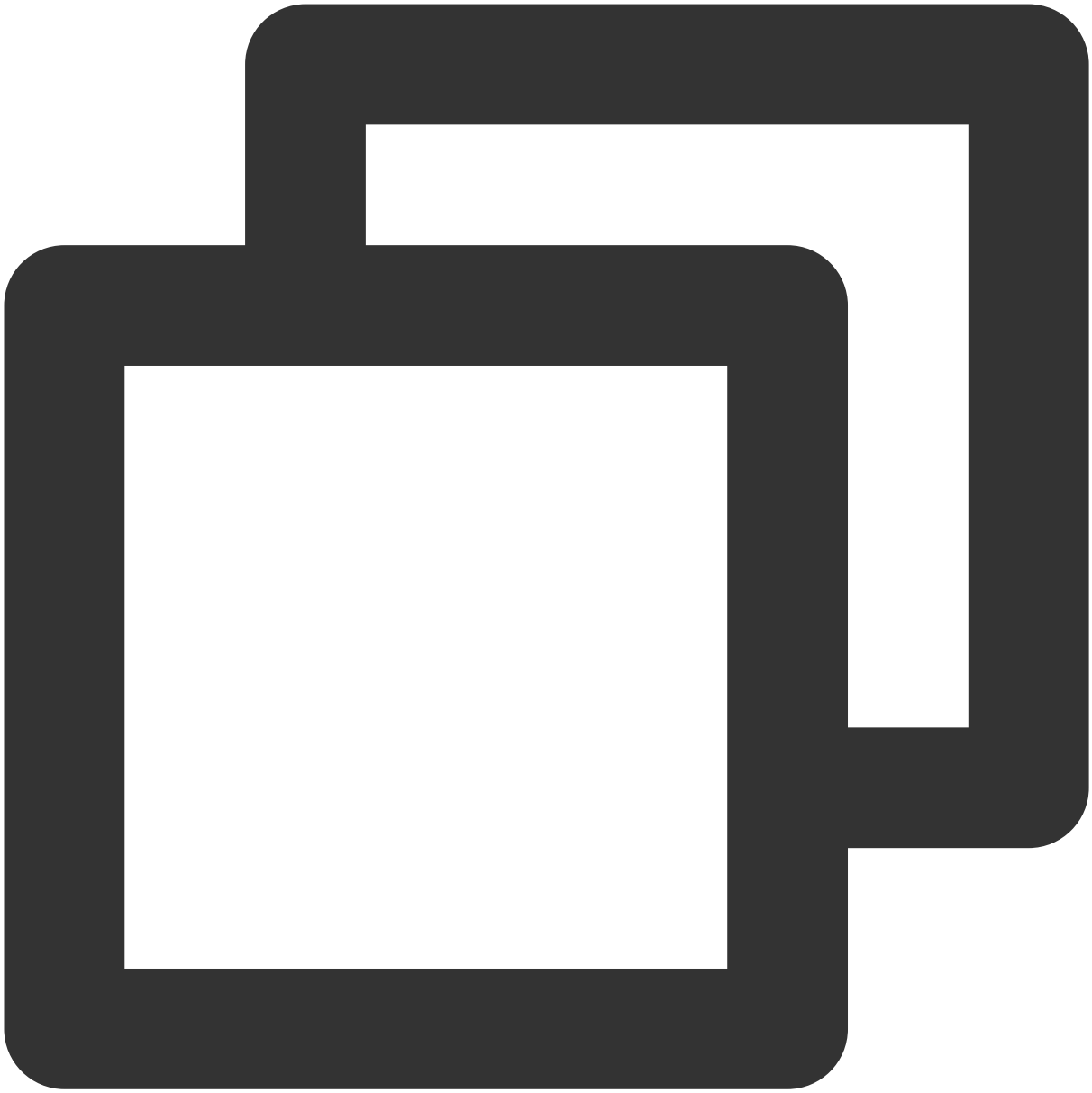
ステップ2：プロジェクト設定の完了

1. appディレクトリ下で `build.gradle` ファイルを見つけ、アプリケーションパッケージ名をご自身のパッケージ名に変更します。



```
applicationId 'com.****.trtc'
```

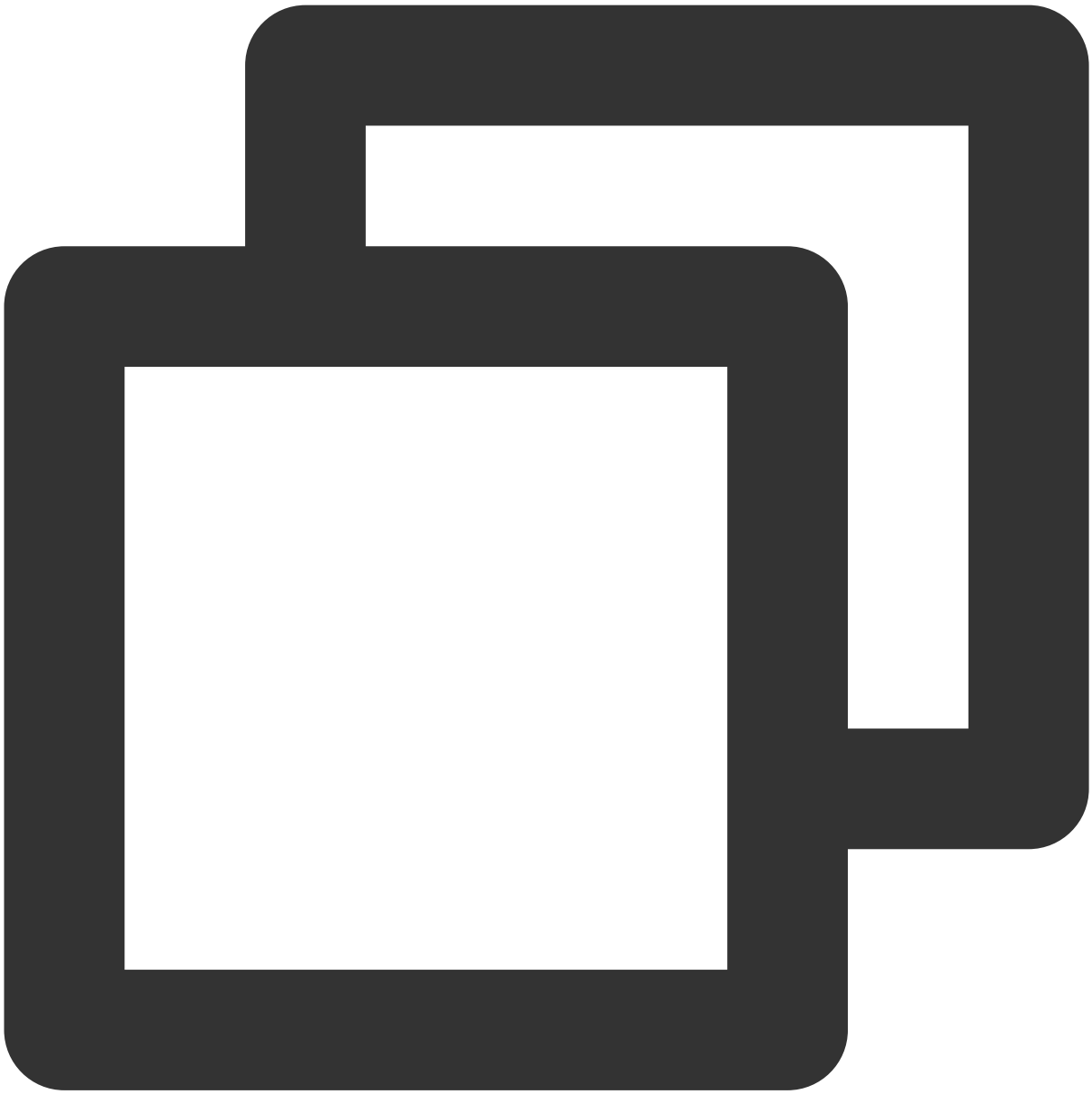
2. appディレクトリ下で `build.gradle` ファイルを見つけ、 `ViVo` アクセスパラメータの `VIVO_APPKEY` 、 `VIVO_APPID` および `HONOR_APPID` を設定し、コンパイルまたは実行エラーを避けます。



```
manifestPlaceholders = [
    "VIVO_APPKEY": "PLACEHOLDER",
    "VIVO_APPID" : "PLACEHOLDER",
    "HONOR_APPID": "PLACEHOLDER"
]
```

3. **Huawei**および**Google**ファイルの設定： appディレクトリ下で `google-services.json` ファイルを置き換えます。このファイルは事前準備でGoogleプラットフォームに登録した際に保存したファイルです。 appディレクトリ下に `agconnect-services.json` ファイルを追加します。このファイルは事前準備でHuaweiプラットフォームに登録した際に保存したファイルです。

4. 事前準備で記録したID、APPID、APPKEYを `PrivateConstants` ファイルに入力し、パラメータの設定が正しいかどうかを確認します。**入力するパラメータは次のとおりです。**



```
public class PrivateConstants {
    /****** Xiaomiオフラインプッシュパラメータstart ******/
    // Tencent Cloudコンソールで、サードパーティの証明書プッシュ後に割り当てられた証明書IDをア
    public static final long XM_PUSH_BUZID = アプリケーションが割り当てた証明書ID;
    // Xiaomiオープンプラットフォームが割り当てたアプリケーションAPPIDおよびAPPKEY
    public static final String XM_PUSH_APPID = "アプリケーションが割り当てたAPPID";
    public static final String XM_PUSH_APPKEY = "アプリケーションが割り当てたAPPKEY";
    /****** Xiaomiオフラインプッシュパラメータend ******/
```

```
}
```

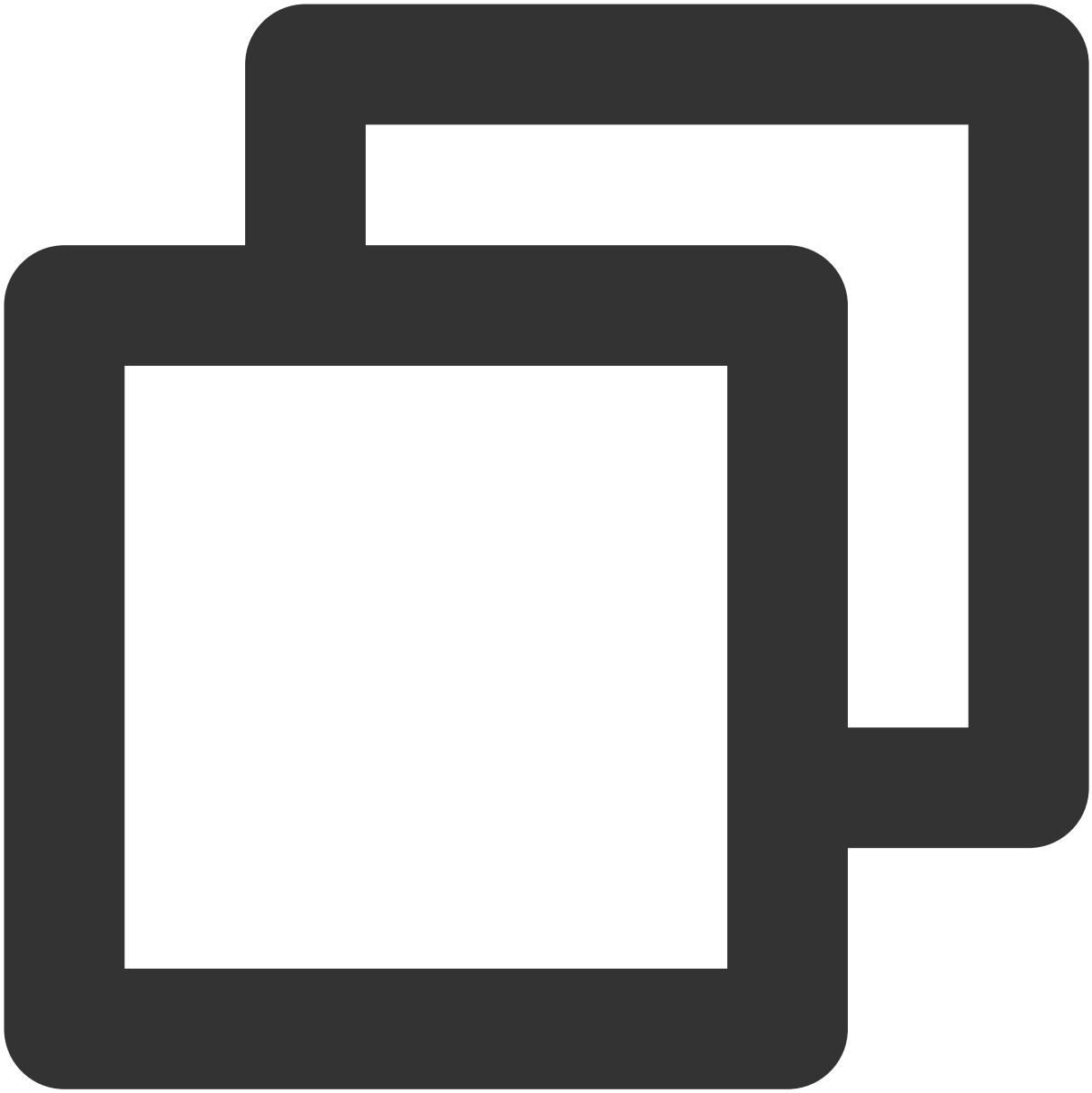
ご注意：

この手順は非常に重要です。パラメータが正しく設定されているかどうかをよく確認してください。

上記の手順が完了すると、プロジェクトに `TUICallKit` のオフライン通知機能が実装されます。

ステップ3：オフライン通知内容のカスタマイズ

TUICallKitはデフォルトの通知スタイルを提供していますが、通知内容をカスタマイズしたい場合は、OfflinePushInfoConfig.javaファイルを変更してください。



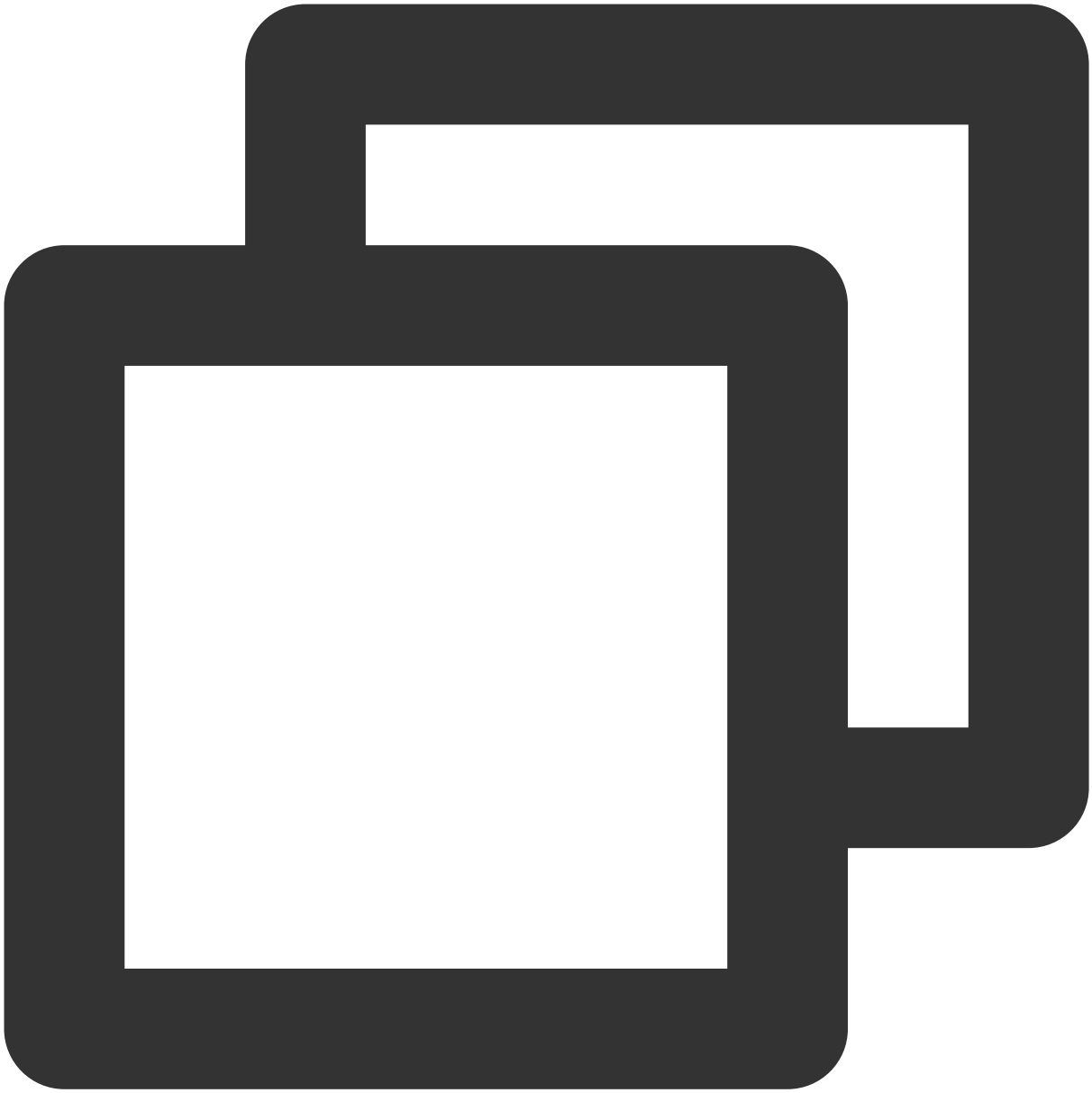
```
public static TUIOfflinePushInfo createOfflinePushInfo(Context context) {
    TUIOfflinePushInfo pushInfo = new TUIOfflinePushInfo();
    pushInfo.setTitle("mike");
    pushInfo.setDesc("You have receive a new call");
    // OPPOはChannelIDを設定するとプッシュメッセージを受信することができます。このchannelIDはこ
    // OPPO must set a ChannelID to receive push messages. This channelID needs to
    pushInfo.setAndroidOPPOChannelID("tuikit");
    pushInfo.setIgnoreIOSBadge(false);
    pushInfo.setIOSSound("phone_ringing.mp3");
    return pushInfo;
}
```

よくあるご質問

オフラインプッシュ通知が受信できない場合は、お調べしますのでお問い合わせください。

1、通知が受信できない

メーカーのコンソールを使用してプッシュテストを行い、成功した場合はメーカーのチャネルに問題がないことがわかります。さらにTUIOfflinePushコンソールのメーカーパラメータ設定が正しいかどうかを確認し、要件に従って入力します。（テストの結果、vivo x9はコンソールでメッセージのカテゴリーを設定する必要があります）。
一部のスマートフォンでは通知が「重要でない通知」に入ることがあります。ステータスバーをプルダウンし、「重要でない通知」に分類されていないかを確認してください。
TUIOfflinePushの登録が正常に行われているかどうかを確認します。次のログをフィルタリングします。



```
TUIOfflinePush
```

2、画面ロック時にディスプレイが点灯しない

Androidスマートフォンはメーカーとプラットフォームの制限により、画面ロック状態での必要な権限が異なります。状況に応じて以下のトラブルシューティングを行ってください。
メーカーのロック画面通知権限がオンになっているかを確認する 一部のメーカーではルールの統一化を行っています。例えばXiaomiではロック画面中のオフライン通知到着時にディスプレイが点灯しない場合、**設定 > ロック画面**で、**ロック画面中の通知受信時にディスプレイを点灯**スイッチをクリックしてオンにします。

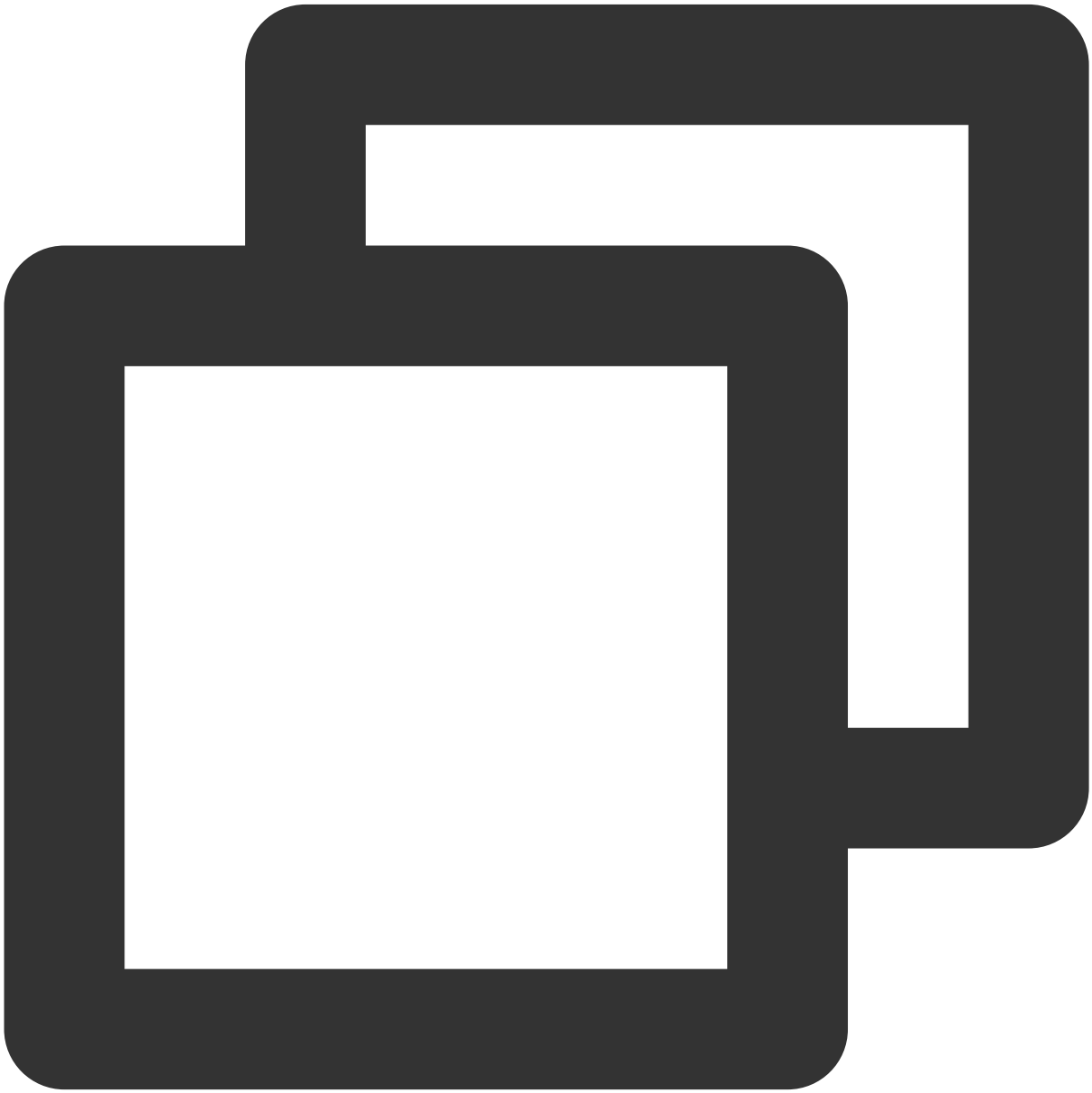
アプリケーションのロック画面通知権限がオンになっているかを確認する 例：Xiaomiではロック画面表示権限が必要です。

説明：

この問題が起こった場合は互換性の処理が必要です。TencentのQQグループ（592465424）に参加すると、問い合わせとフィードバックを行うことができます。

3、オフラインプッシュ通知をクリックしても通話画面が開かない

通話リクエストが見つかるかどうかを確認する必要があります。次のログをフィルタリングできます。



```
onReceiveNewInvitation
```

4、アプリケーションがバックエンドにあるとき、通話画面をフロントエンドに自動的にプルできない

アプリケーションをバックエンドからフロントエンドに自動的にプルする場合、Appが「バックエンド自動起動」または「フローティングウィンドウ」権限を有効にしているかどうかを確認する必要があります。
メーカーが異なる場合や、同じメーカーであってもAndroidのバージョンが異なる場合は、アプリケーションに許可する権限や権限名が一致しないことがありますのでご注意ください。例えば、Xiaomi 6では**バックエンドの画面ポップアップ**権限を有効にすればよいだけですが、Redmiでは**バックエンドの画面ポップアップ**と**フローティングウィンドウの表示**の権限を同時に有効にする必要があります。
説明：
手動ですべての権限を有効化しても、通話画面のフロントエンドへの自動プルができないことをテスト中に発見した場合は、互換性の処理が必要です。

クラウドレコーディング（TUICallKit）

最終更新日：：2024-07-19 14:53:21

ここではTUICallKitのクラウドレコーディングを起動し、重要な通話の保存、審査などに役立てる方法についてご説明します。自動レコーディングソリューションとREST APIレコーディングソリューションという2つのソリューションをご用意しています。

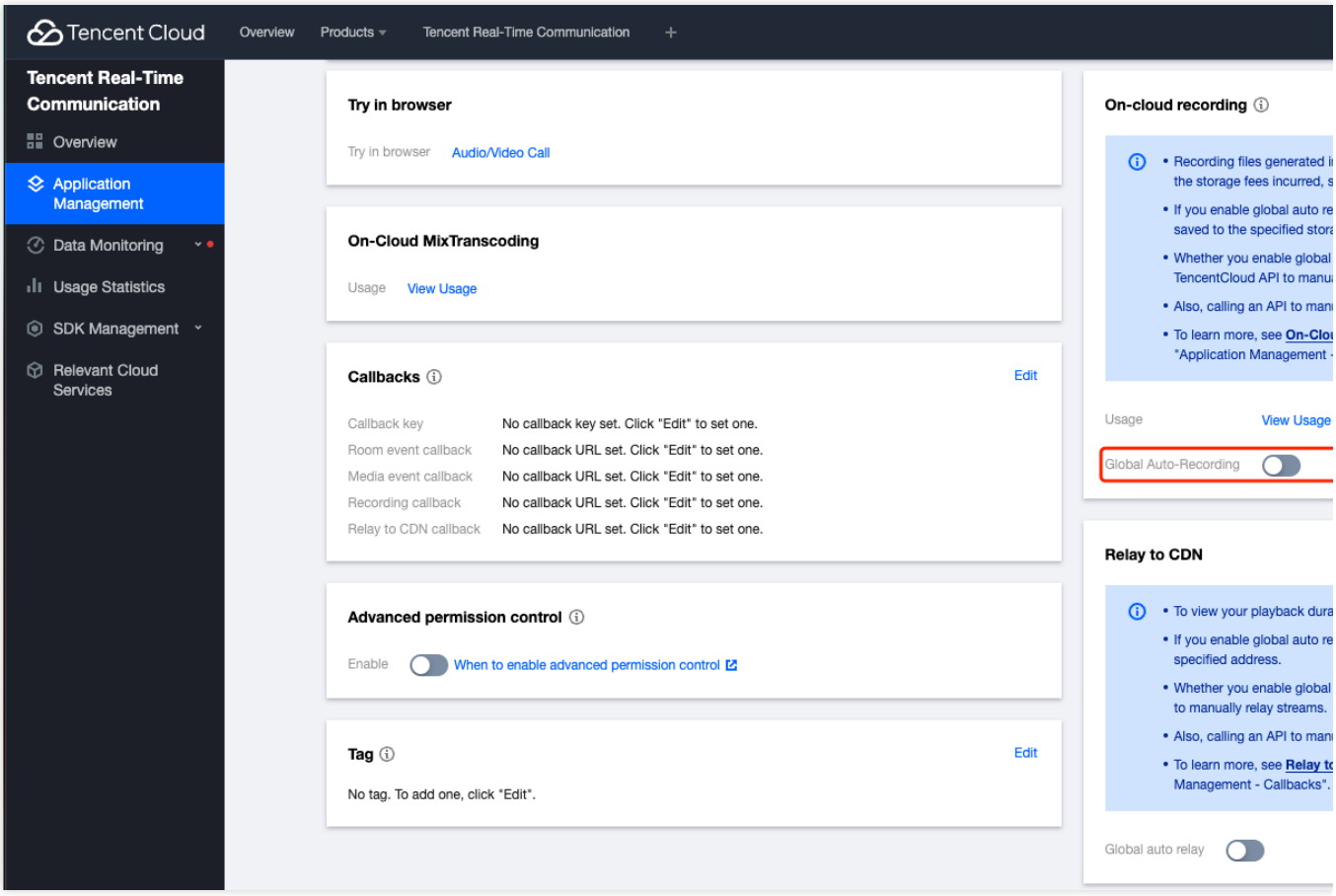
説明：

TUICallKitはTencent Cloudの基本的なPaaSサービスを複数統合しています。このうち、オーディオビデオ関連機能はTRTCに依存しています。このためTUICallKitのクラウドレコーディング機能はTRTCコンソールで設定する必要があります。

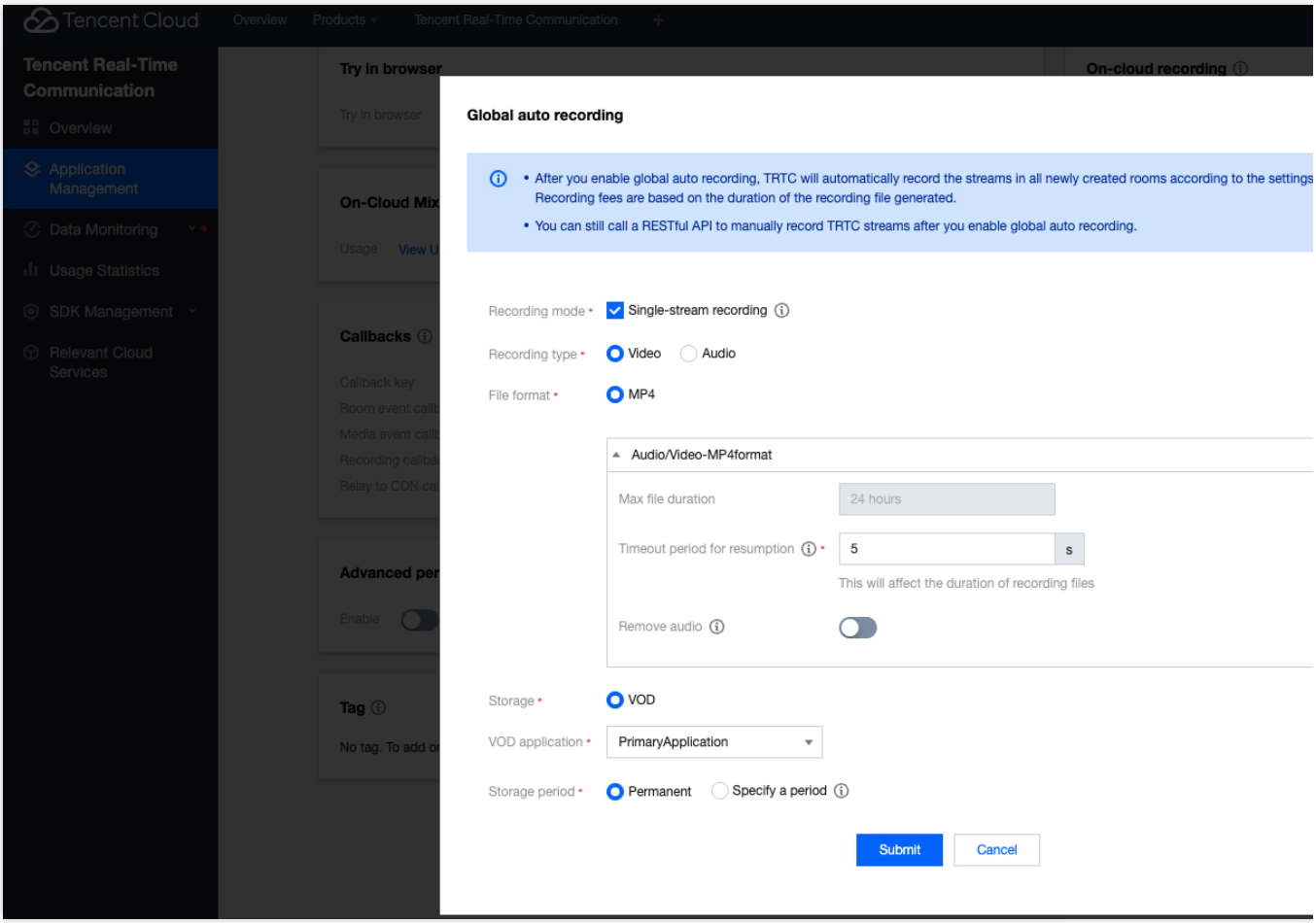
方法1：自動レコーディングソリューション（推奨）

自動レコーディングソリューションの使用を推奨します。業務側でレコーディングを起動および停止する必要がなく、レコーディングタスクはTencent Cloud TRTCバックエンドが管理します。通話中にオーディオビデオストリームのアップストリームがあれば自動的にレコーディングし、**スピーディーかつ簡単にアクセスできます**。次のいくつかの手順で完成させることができます。

1. TRTCコンソール > アプリケーション管理でSDKAppIdに対応するアプリケーションを見つけ、機能設定ページに進みます。
2. 機能設定ページでクラウドレコーディング設定のカードを確認できます。**クラウドレコーディング機能を有効にした後、Global Auto-Recordingテンプレートの作成**をクリックします。



3. オーディオビデオ通話の業務シナリオ（1v1通話、グループ通話）に応じて、次のようにパラメータを設定することを推奨しますが、ご自身の業務ニーズに応じてカスタムレコーディングテンプレートを設定することも可能です。



ご注意：

グローバルレコーディングがサポートする最大ミクスストリーミング人数は8人です。通話人数が8人を超える場合（自分を含めて）、最後のユーザーのストリームはレコーディングされません。

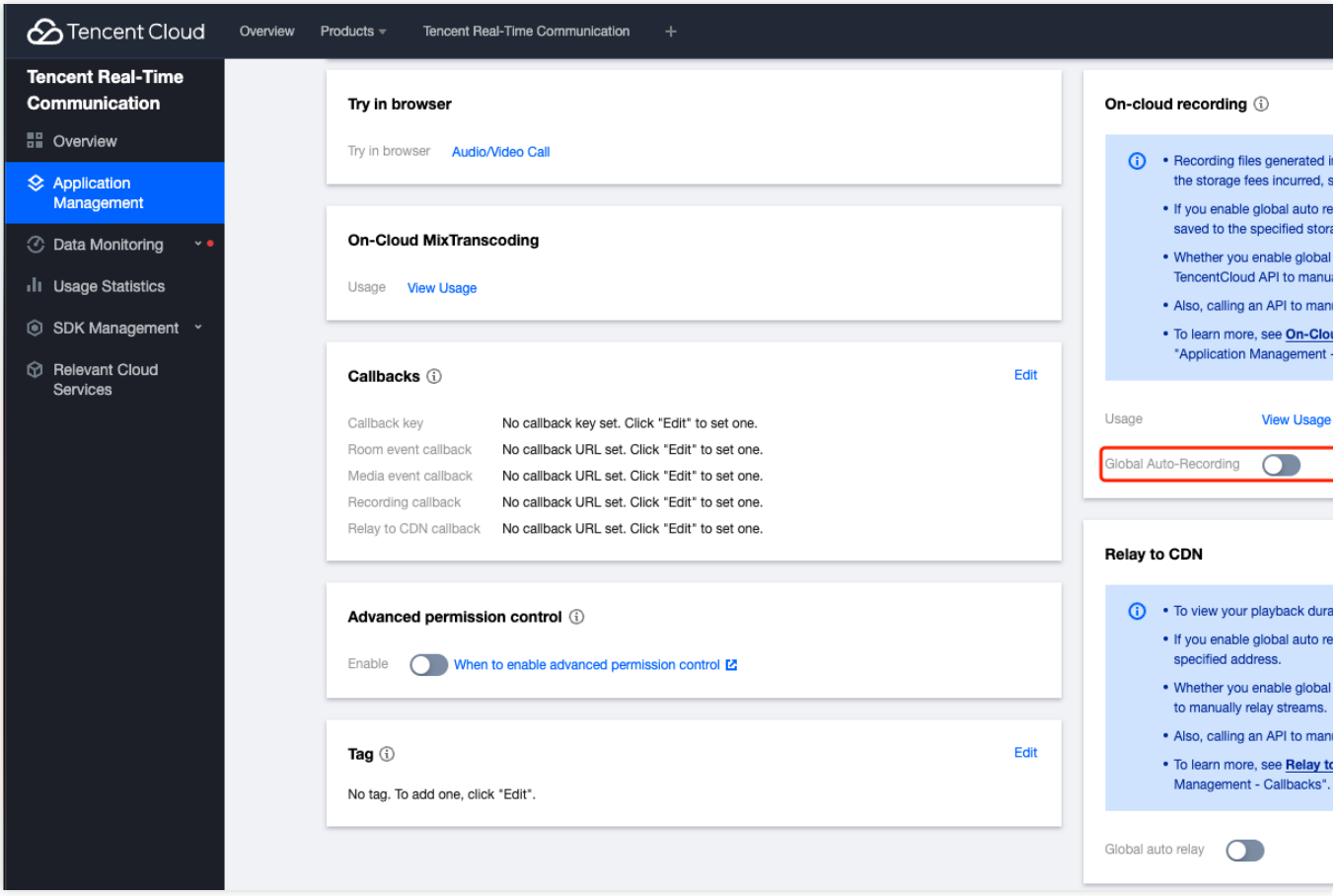
Global Auto-Recording機能を有効にすると、通話応答後かつオーディオビデオアップストリームが存在する場合にレコーディングタスクの自動起動がトリガーされ、通話終了後に自動的にレコーディングが停止します。ネットワークまたはその他の異常によって退室になった場合、ご自身の設定したMaxIdleTime値（アイドル状態の待機時間。デフォルトでは5秒）に基づいてバックエンドが自動的にレコーディングタスクを停止し、それ以上の料金ロスが発生しないようにします。

4. **テンプレートの作成が完了した後、**Global Auto-Recordingにチェックを入れれば完了です。

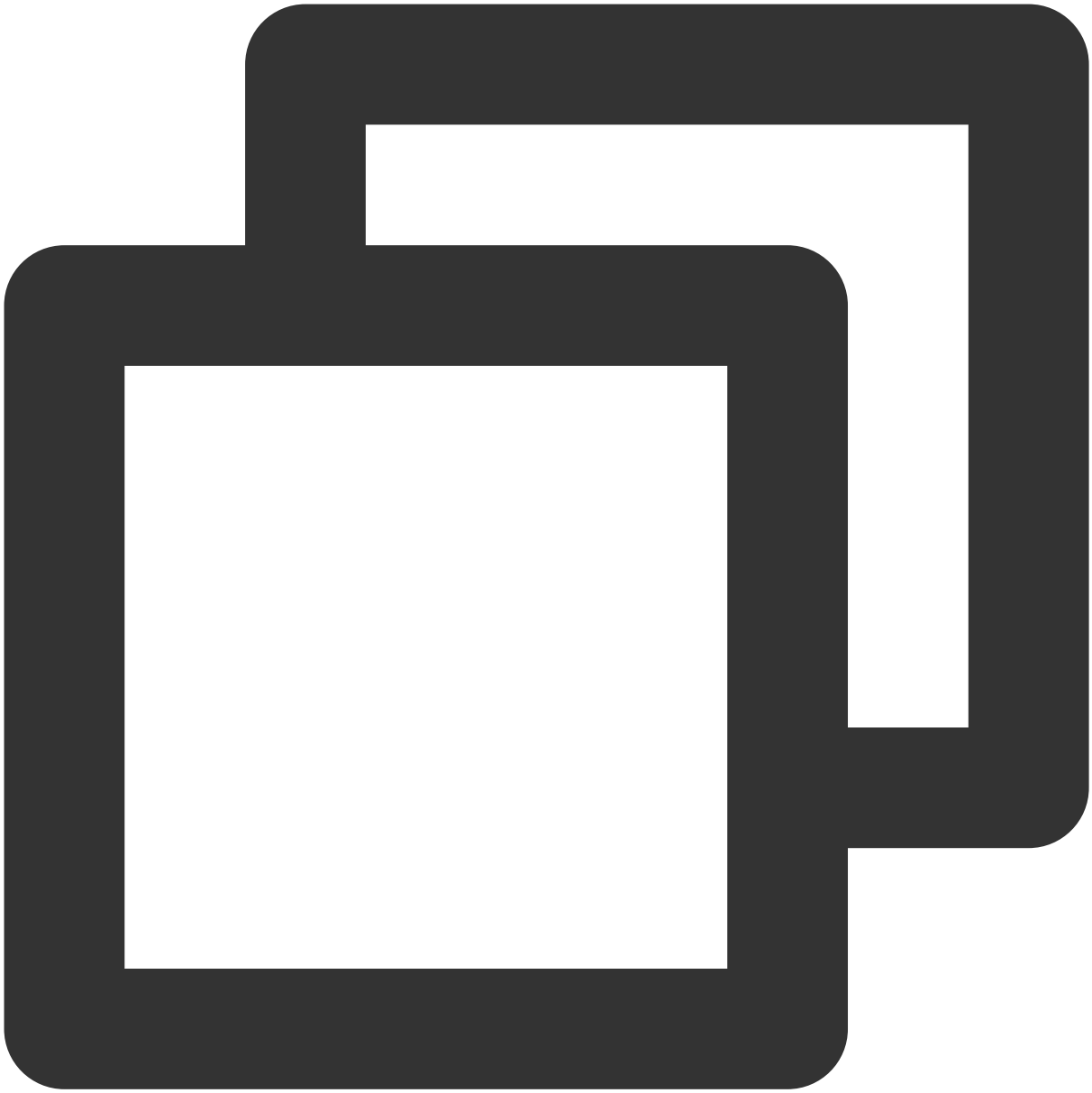
方法2：REST APIレコーディングソリューション

自動レコーディングソリューションで業務ニーズを満たせない場合は、よりフレキシブルなREST APIレコーディングソリューションを使用することもできます。レコーディングサブスクリプションルーム内のキャスターの指定、ミクスストリーミングレイアウトのカスタマイズ、レコーディング中のレイアウトおよびサブスクリプション更新などが可能ですが、**業務バックエンドサービスとの組み合わせが必要で、接続がより複雑になり、機能は強力になります。重要な手順は次のとおりです。**

1. TRTCコンソール > アプリケーション管理でSDKAppIdに対応するアプリケーションを見つけ、機能設定ページに進みます。
2. 機能設定ページでクラウドレコーディング設定のカードを確認できます。**クラウドレコーディング機能を有効にする**と、この時点で**レコーディングを手動でカスタマイズ**、すなわちREST APIモードにデフォルトでチェックが入っています。



3. その後、REST API（CreateCloudRecording）を呼び出してクラウドレコーディングを起動することができます。ここではTUICallObserverの通知イベントを監視することで、オーディオビデオ通話の開始時にレコーディングを起動できるようにすることをお勧めします。Javaコードの例を挙げます。



```
TUICallEngine.createInstance(context).addObserver(new TUICallObserver() {
    @Override
    public void onCallBegin(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType
        // REST APIを使用してレコーディングタスクを起動することを業務バックエンドに通知します。
    }
});
```

4. クライアントにネットワーク不良、プロセスキルなどが発生して通話が異常終了する可能性を考慮し、レコーディング終了の方法については、TRTCルームステータスのコールバック（詳細についてはサーバーイベントコールバックの監視をご参照ください）をサブスクライブし、TRTCルームのステータスが解散となったコールバック

を受信した時点でREST API（DeleteCloudRecording）を呼び出してクラウドのレコーディングタスクを停止するようにすることをお勧めします。

よくあるご質問

1. レコーディング時間の明細を確認するにはどうすればよいですか。

TRTCコンソール > クラウドレコーディングでレコーディングの時間明細を確認できます。

2. レコーディングしたファイルを確認するにはどうすればよいですか。

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、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、検索ボックスにキーワードを入力します。レコーディングファイルの命名ルールは次のとおりです。
シングルストリーミングレコーディングMP4ファイル名のルール：
`<SdkAppId>_<RoomId>_UserId_s_<UserId>_UserId_e_<MediaId>_<Index>.mp4`
ミクスストリーミングレコーディングMP4ファイル名のルール： `<SdkAppId>_<RoomId>_<Index>.mp4`

Client APIs (TUICallKit)
Android
API概要

最終更新日：：2024-07-19 14:53:21

TUICallKit (UIインターフェースあり)

TUICallKit APIはオーディオビデオ通話コンポーネントの**UIインターフェース付き**のものです。TUICallKit APIを使用することで、WeChatのようなオーディオビデオ通話シーンをシンプルなインターフェースでスピーディーに実現できます。

API	説明
createInstance	TUICallKitインスタンスの作成（シングルトンモード）
setSelfInfo	ユーザーのプロフィール画像、ニックネームの設定
call	1v1通話の開始
groupCall	グループ通話の開始
joinInGroupCall	現在のグループ通話に自主的に参加
setCallingBell	カスタム着信音の設定
enableMuteMode	ミュートモードのオン/オフ
enableFloatWindow	フローティングウィンドウ機能のオン/オフ

TUICallEngine (UIインターフェースなし)

TUICallEngine APIはオーディオビデオ通話コンポーネントの**UIインターフェースがない**ものです。TUICallKitのインタラクションではニーズを満たせない場合はこのAPIを使用し、業務ニーズに応じてパッケージをカスタマイズすることができます。

API	説明
createInstance	TUICallEngineインスタンスの作成（シングルトンモード）
destroyInstance	TUICallEngineインスタンスの破棄（シングルトンモード）

init	オーディオビデオ通話基本機能の認証完了
addObserver	イベントコールバックの追加
removeObserver	コールバックインターフェースの削除
call	1v1通話の開始
groupCall	グループ通話の開始
accept	通話応答
reject	通話拒否
hangup	通話終了
ignore	通話を無視
inviteUser	グループ通話中に他の人を招待
joinInGroupCall	現在のグループ通話に自主的に参加
switchCallMediaType	通話メディアタイプの切り替え。ビデオ通話からオーディオ通話への切り替えなど
startRemoteView	リモートユーザービデオストリームのサブスクリプション開始
stopRemoteView	リモートユーザービデオストリームのサブスクリプション停止
openCamera	カメラの起動
closeCamera	カメラの終了
switchCamera	フロント/リアカメラの切り替え
openMicrophone	マイクをオンにする
closeMicrophone	マイクをオフにする
selectAudioPlaybackDevice	オーディオ再生デバイスの選択（ヘッドホン/スピーカー）
setSelfInfo	ユーザーのニックネーム、プロフィール画像の設定
enableMultiDeviceAbility	TUICallEngineのマルチデバイスログインモードのオン/オフ （プレミアム版パッケージのみサポート）

TUICallObserver

TUICallObserverはTUICallEngineに対応するコールバックイベントクラスです。このコールバックによって、関心のあるコールバックイベントを監視することができます。

API	説明
onError	通話中のエラーコールバック
onCallReceived	通話リクエストのコールバック
onCallCancelled	通話キャンセルのコールバック
onCallBegin	通話接続のコールバック
onCallEnd	通話終了のコールバック
onCallMediaTypeChanged	通話メディアタイプ変更発生のコールバック
onUserReject	xxxxユーザーによる通話拒否のコールバック
onUserNoResponse	xxxxユーザーの応答なしのコールバック
onUserLineBusy	xxxxユーザーが通話中である場合のコールバック
onUserJoin	xxxxユーザーの通話参加のコールバック
onUserLeave	xxxxユーザーの通話からの退出のコールバック
onUserVideoAvailable	xxxユーザーのビデオストリームの有無のコールバック
onUserAudioAvailable	xxxユーザーのオーディオストリームの有無のコールバック
onUserVoiceVolumeChanged	全ユーザーの音量レベルフィードバックのコールバック
onUserNetworkQualityChanged	全ユーザーのネットワーク品質フィードバックのコールバック

主要なタイプの定義

API	説明
TUICallDefine.MediaType	通話のメディアタイプ。列挙タイプ：ビデオ通話、音声通話
TUICallDefine.Role	通話のロール。列挙タイプ：発呼側、着呼側
TUICallDefine.Status	通話の状態。列挙タイプ：アイドル状態、応答待ち、応答中
TUICommonDefine.RoomId	オーディオビデオルームId。数字、文字列の2種類をサポートしています

TUICommonDefine.Camera	カメラIdパラメータ。列挙タイプ：フロントカメラ、リアカメラ
TUICommonDefine.AudioPlaybackDevice	音声再生デバイス。列挙タイプ：スピーカー、ヘッドホン
TUICommonDefine.NetworkQualityInfo	現在のネットワーク品質情報

TUICallKit

最終更新日：：2024-07-19 14:53:21

TUICallKit APIの概要

TUICallKit APIはオーディオビデオ通話コンポーネントの**UIインターフェース付き**のものです。TUICallKit APIを使用することで、WeChatのようなオーディオビデオ通話シーンをシンプルなインターフェースでスピーディーに実現できます。より詳細なアクセス手順については、TUICallKitクイックアクセスをご参照ください。

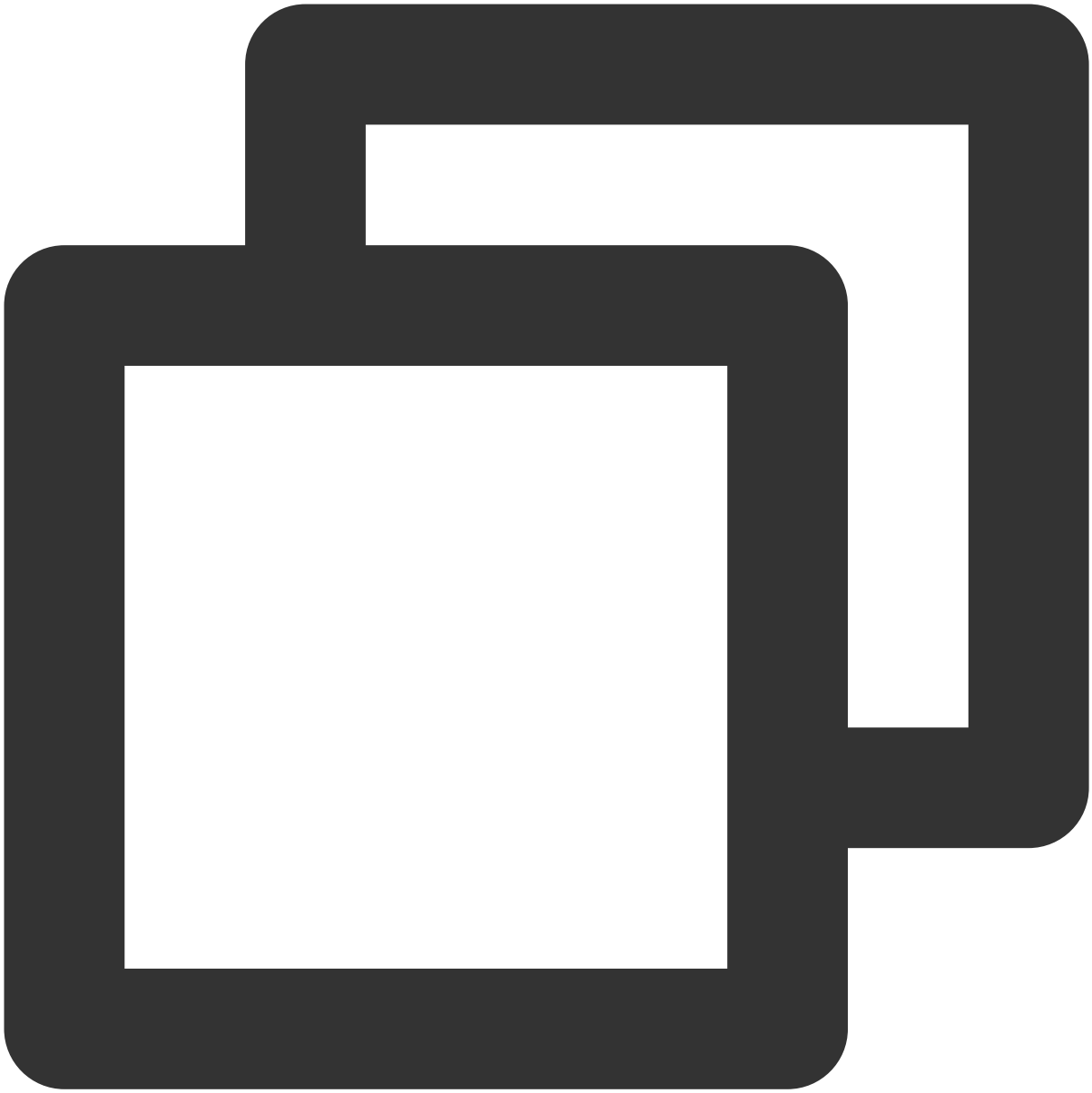
APIの概要

API	説明
createInstance	TUICallKitインスタンスの作成（シングルトンモード）
setSelfInfo	ユーザーのニックネーム、プロフィール画像の設定
call	1v1通話の開始
groupCall	グループ通話の開始
joinInGroupCall	現在のグループ通話に自主的に参加
setCallingBell	カスタム着信音の設定
enableMuteMode	ミュートモードのオン/オフ
enableFloatWindow	フローティングウィンドウ機能のオン/オフ

APIの詳細

createInstance

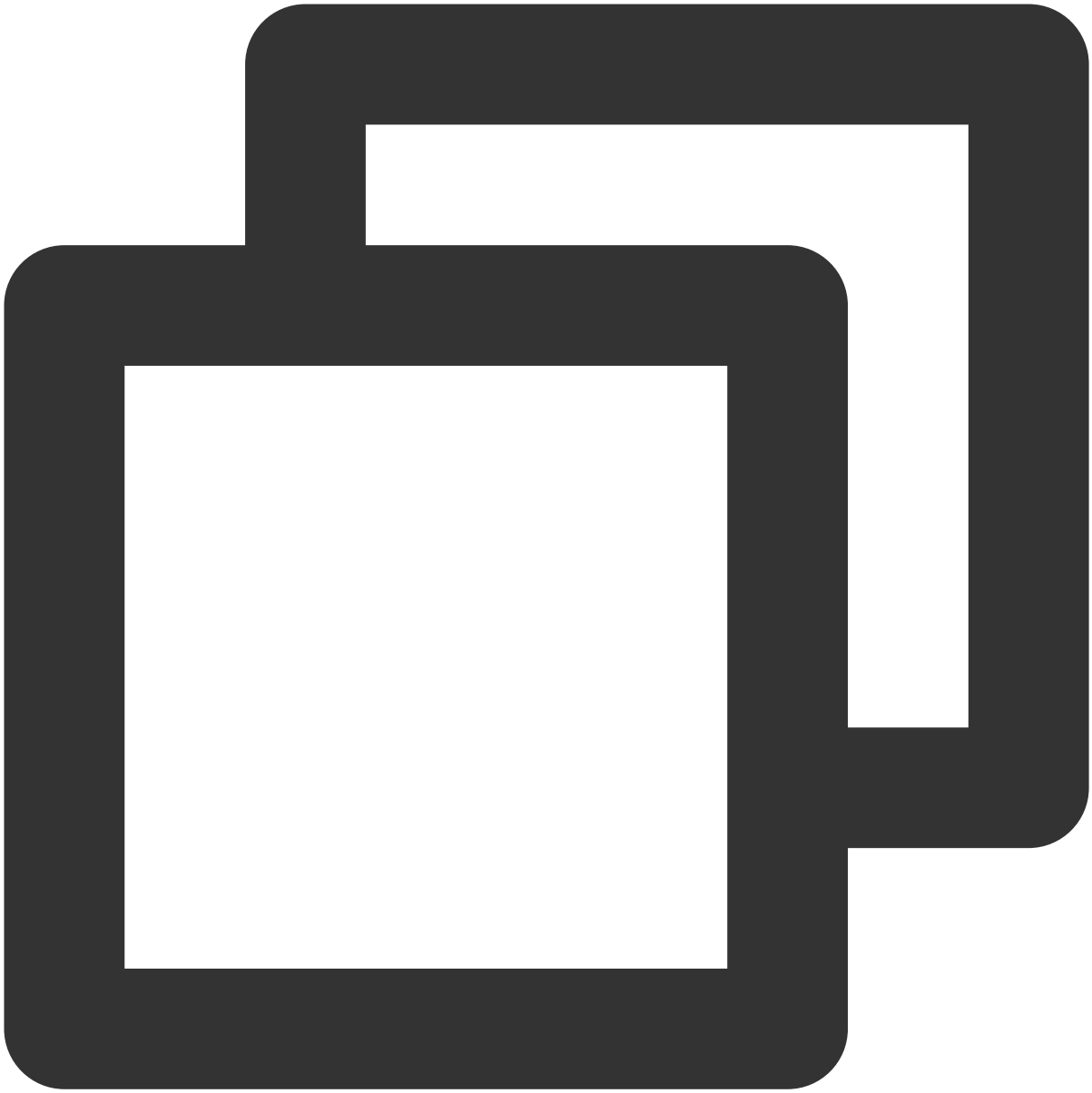
TUICallKitのシングルトンを作成します。



```
TUICallKit createInstance(Context context)
```

setSelfInfo

ユーザーニックネーム、プロフィール画像を設定します。ユーザーニックネームは500バイト以内、ユーザープロフィール画像はURL形式でなければなりません。



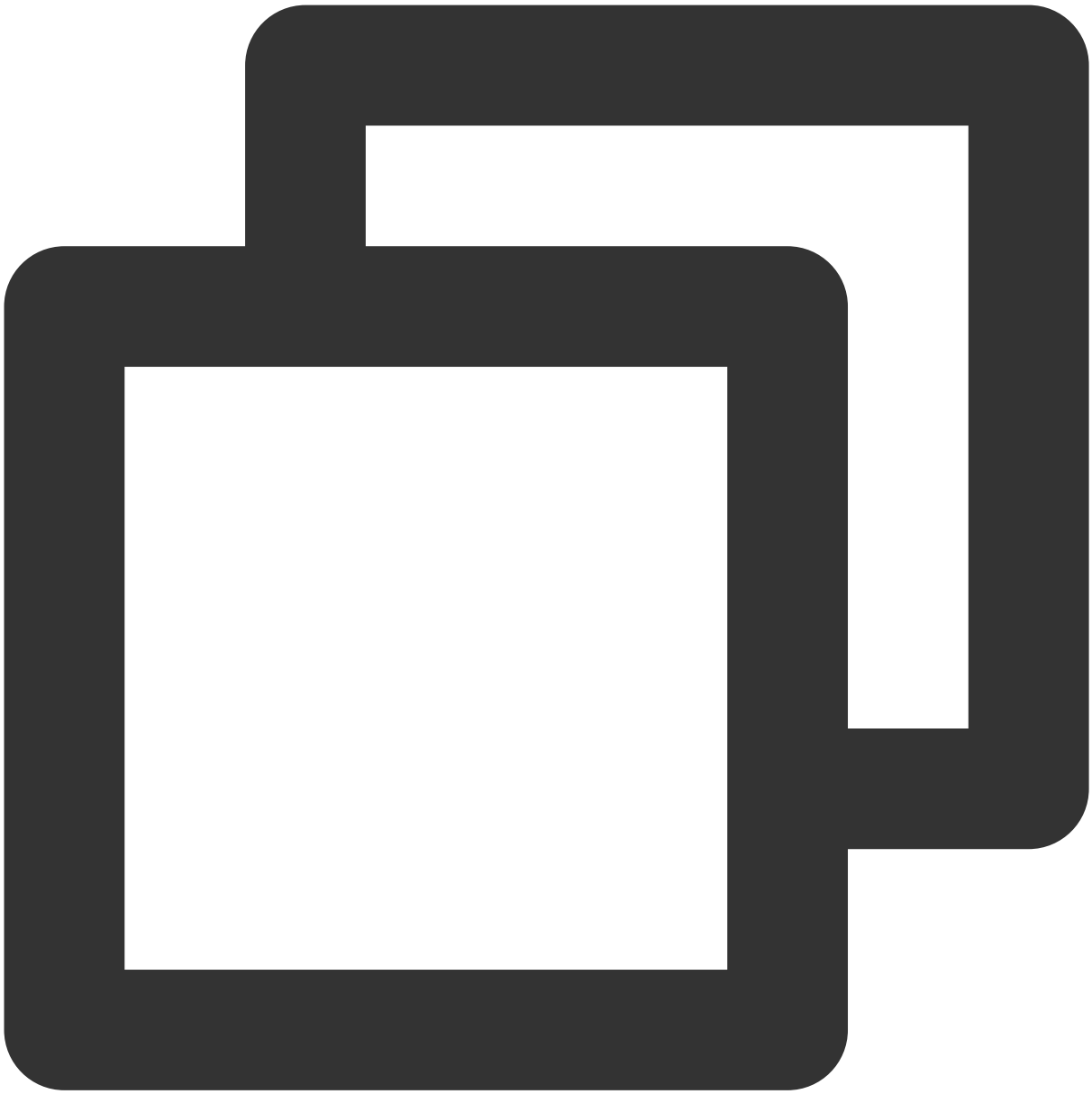
```
void setSelfInfo(String nickname, String avatar, TUICommonDefine.Callback callback)
```

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
nickname	String	ターゲットユーザーのニックネーム
avatar	String	ターゲットユーザーのプロフィール画像

call

電話をかけます（1v1通話）。



```
void call(String userId, TUICallDefine.MediaType callMediaType)
```

パラメータは下表に示すとおりです。

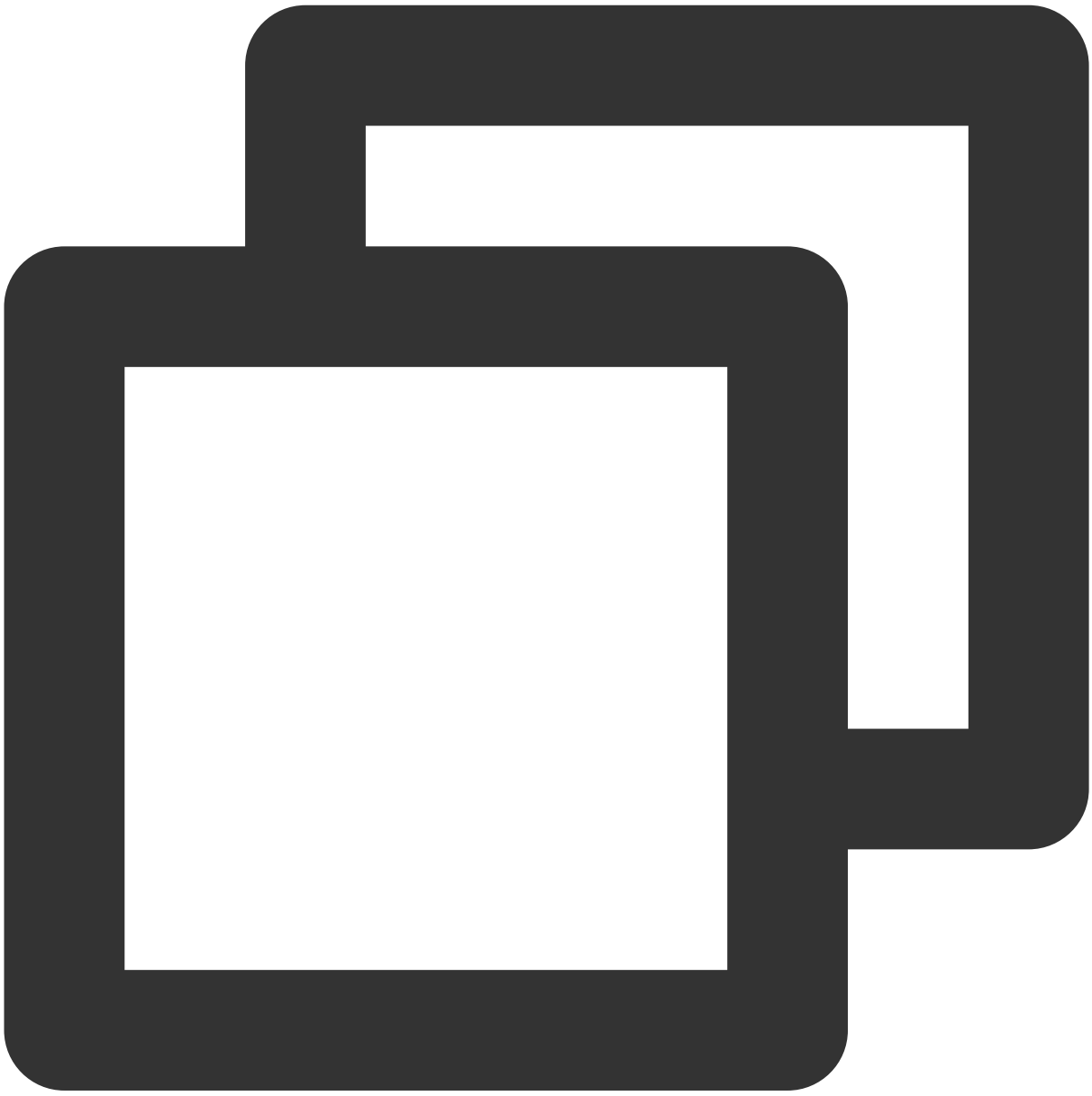
パラメータ	タイプ	意味
userId	String	ターゲットユーザーのuserId
callMediaType	TUICallDefine.MediaType	通話のメディアタイプ。ビデオ通話、音声通話など

groupCall

グループ通話を開始します。

ご注意：

グループ通話を使用する前にIMグループを作成する必要があります。作成済みの場合は無視してください。



```
void groupCall(String groupId, List<String> userIdList, TUICallDefine.MediaType 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

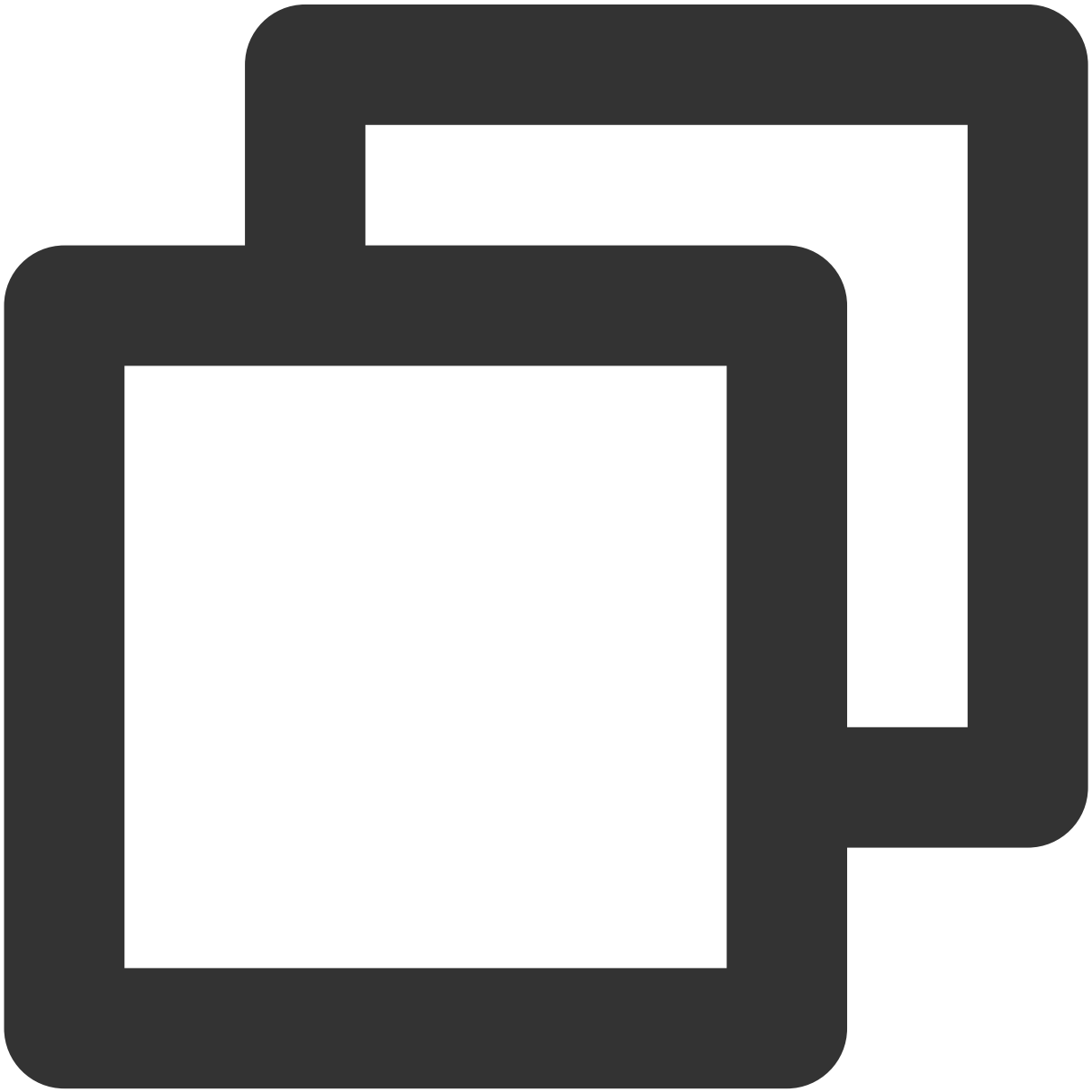
groupId	String	今回のグループ通話のグループID
userIdList	List	ターゲットユーザーのuserIdリスト
callMediaType	TUICallDefine.MediaType	通話のメディアタイプ。ビデオ通話、音声通話など

joinInGroupCall

グループ通話を開始します。

ご注意：

グループ通話を使用する前にIMグループを作成する必要があります。作成済みの場合は無視してください。



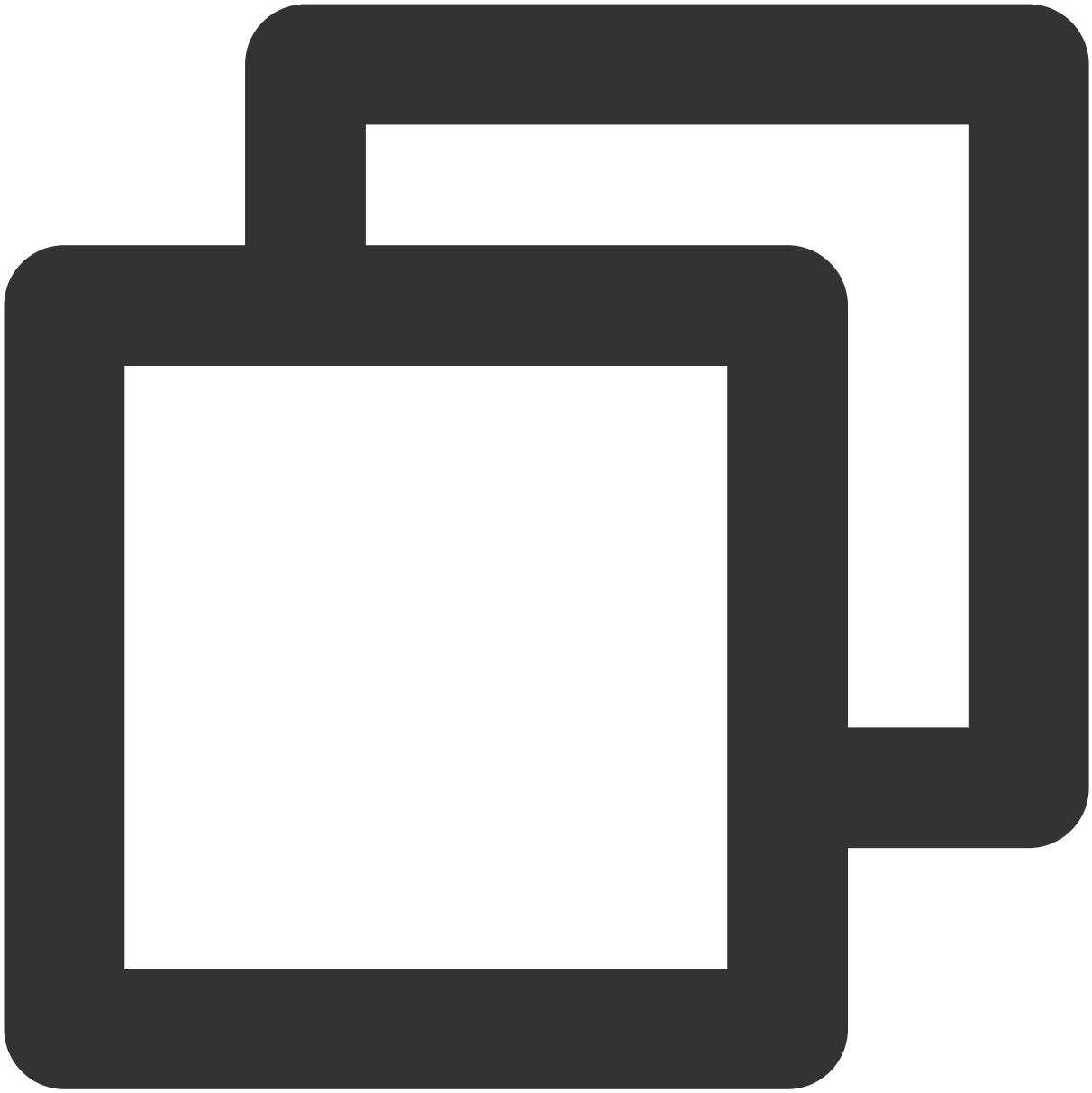
```
void joinInGroupCall(TUICommonDefine.RoomId roomId, String groupId, TUICallDefine.M
```

パラ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	TUICommonDefine.RoomId	今回の通話のオーディオビデオルームID。現在は数字のルームナンバーのみサポートしています。文字列のルームナンバーは今後のバージョンでサポート予定です
groupId	String	今回のグループ通話のグループID
callMediaType	TUICallDefine.MediaType	通話のメディアタイプ。ビデオ通話、音声通話など

setCallingBell

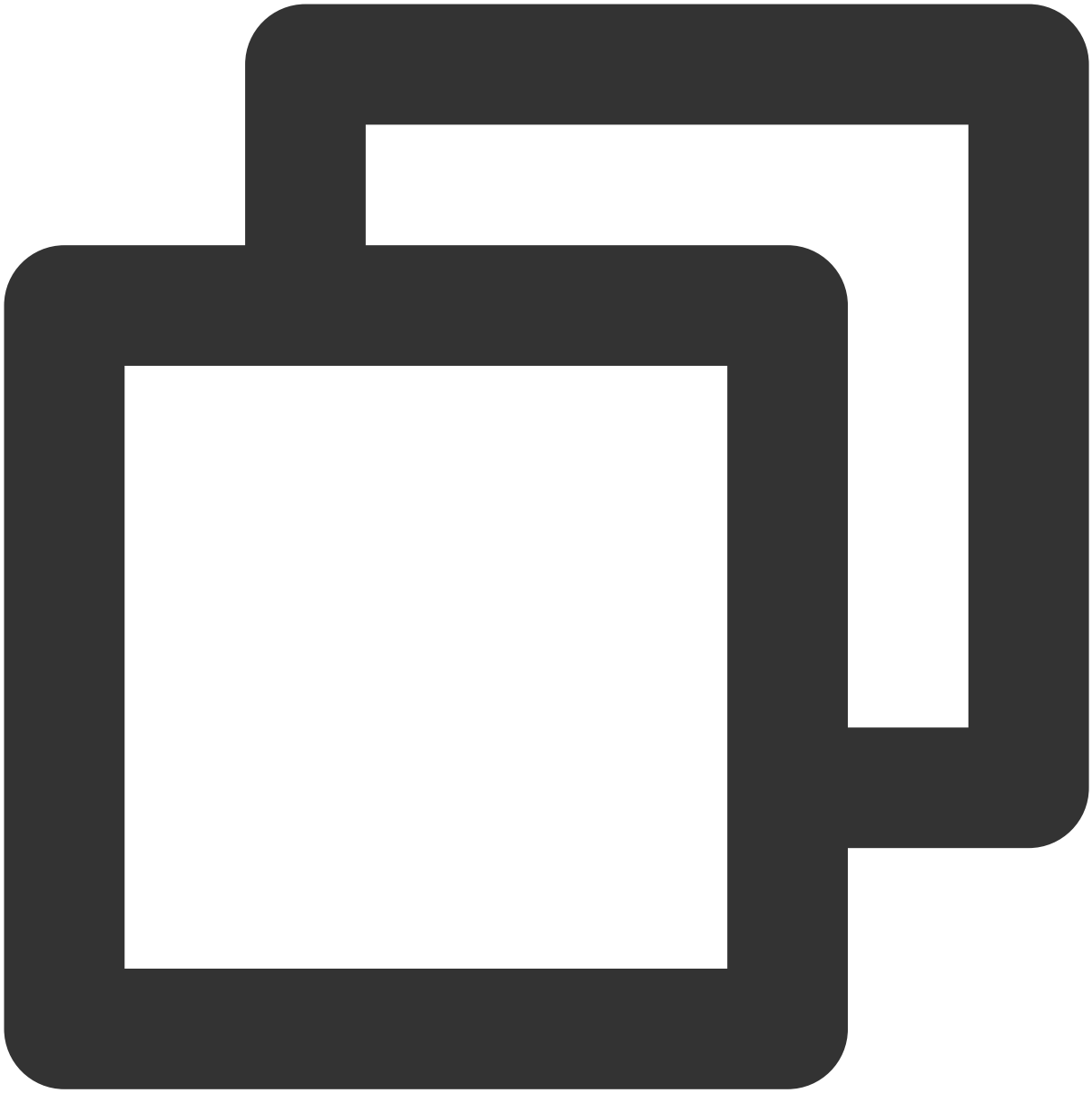
カスタム着信音を設定します。 ここではローカルファイルアドレスのみ渡すことができます。このファイルディレクトリにアプリケーションがアクセスできることを確認する必要があります。
着信音を設定後、デバイスにバインドします。ユーザーを変更しても着信音はそのまま有効です。
デフォルトの着信音に戻したい場合は、 `filePath` にnullを渡します。



```
void setCallingBell(String filePath);
```

enableMuteMode

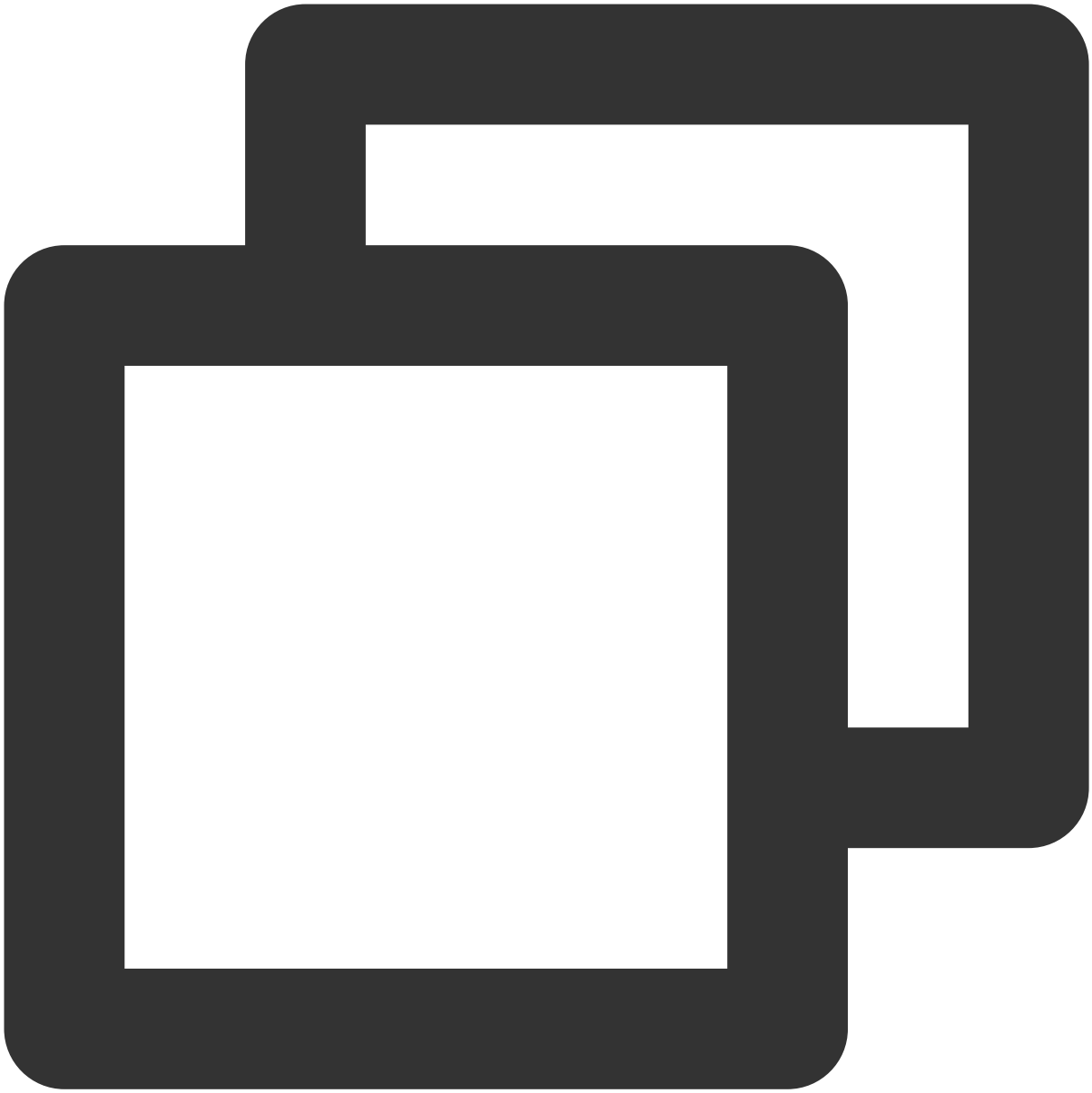
ミュートモードをオン/オフします。



```
void enableMuteMode(boolean enable);
```

enableFloatWindow

フローティングウィンドウ機能をオン/オフします。
デフォルトでは `false` で、通話画面の左上隅のフローティングウィンドウボタンは非表示になっており、trueに設定すると表示されます。



```
void enableFloatWindow(boolean enable);
```

TUICallEngine

最終更新日：：2024-07-19 14:53:21

TUICallEngine APIの概要

TUICallEngine APIはオーディオビデオ通話コンポーネントの**UIインターフェースがない**ものです。TUICallKitのインタラクションではニーズを満たせない場合はこのインターフェースを使用し、パッケージのインタラクションをカスタマイズすることができます。

APIの概要

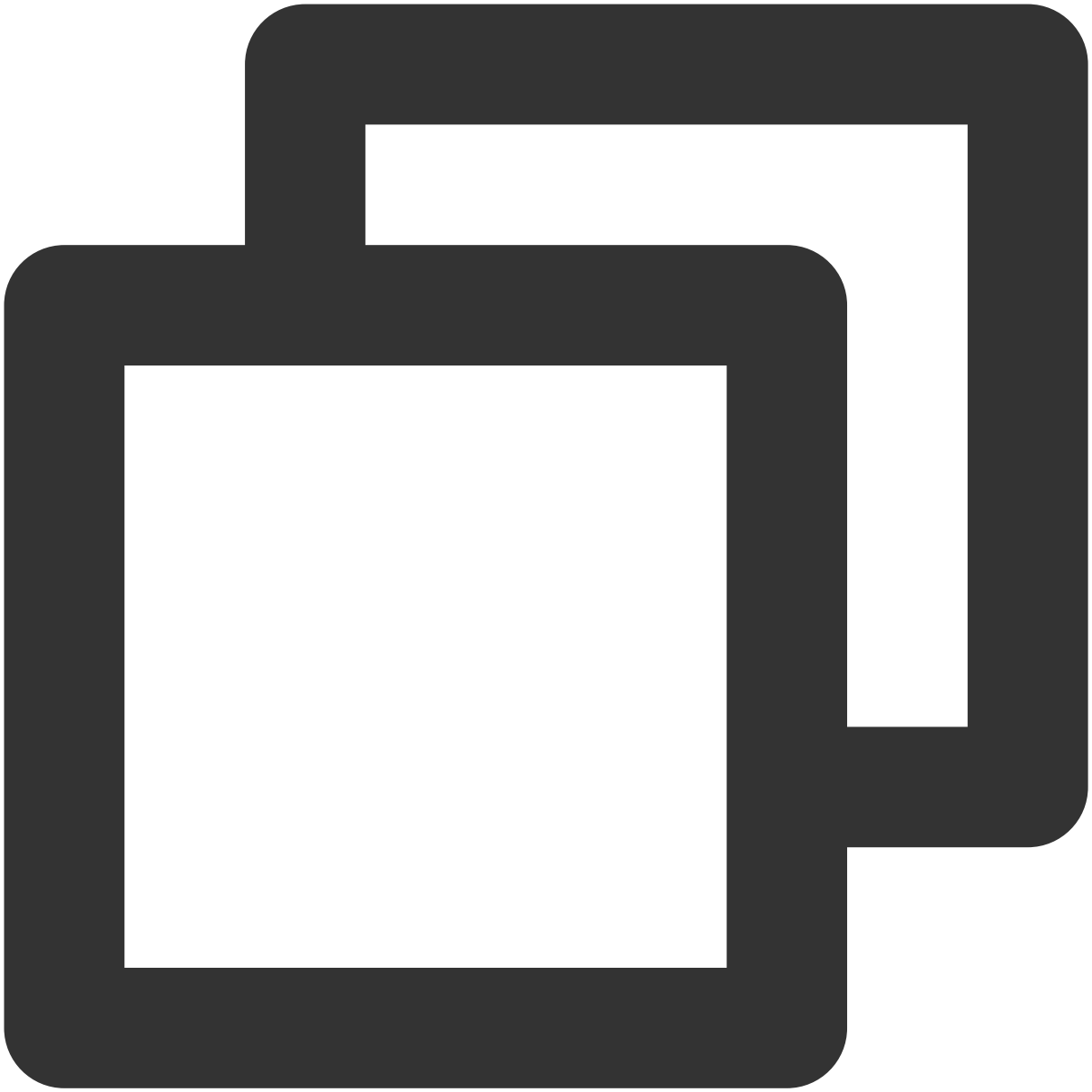
API	説明
createInstance	TUICallEngineインスタンスの作成（シングルトンモード）
destroyInstance	TUICallEngineインスタンスの破棄（シングルトンモード）
init	オーディオビデオ通話基本機能の認証完了
addObserver	イベントコールバックの追加
removeObserver	コールバックインターフェースの削除
call	1v1通話の開始
groupCall	グループ通話の開始
accept	通話応答
reject	通話拒否
hangup	通話終了
ignore	通話を無視
inviteUser	グループ通話中に他の人を招待
joinInGroupCall	現在のグループ通話に自主的に参加
switchCallMediaType	通話メディアタイプの切り替え。ビデオ通話からオーディオ通話への切り替えなど
startRemoteView	リモートユーザービデオストリームのサブスクリプション開始

stopRemoteView	リモートユーザービデオストリームのサブスクリプション停止
openCamera	カメラの起動
closeCamera	カメラの終了
switchCamera	フロント/リアカメラの切り替え
openMicrophone	マイクをオンにする
closeMicrophone	マイクをオフにする
selectAudioPlaybackDevice	オーディオ再生デバイスの選択（ヘッドホン/スピーカー）
setSelfInfo	ユーザーのニックネーム、プロフィール画像の設定
enableMultiDeviceAbility	TUICallEngineのマルチデバイスログインモードのオン/オフ （プレミアム版パッケージのみサポート）

APIの詳細

createInstance

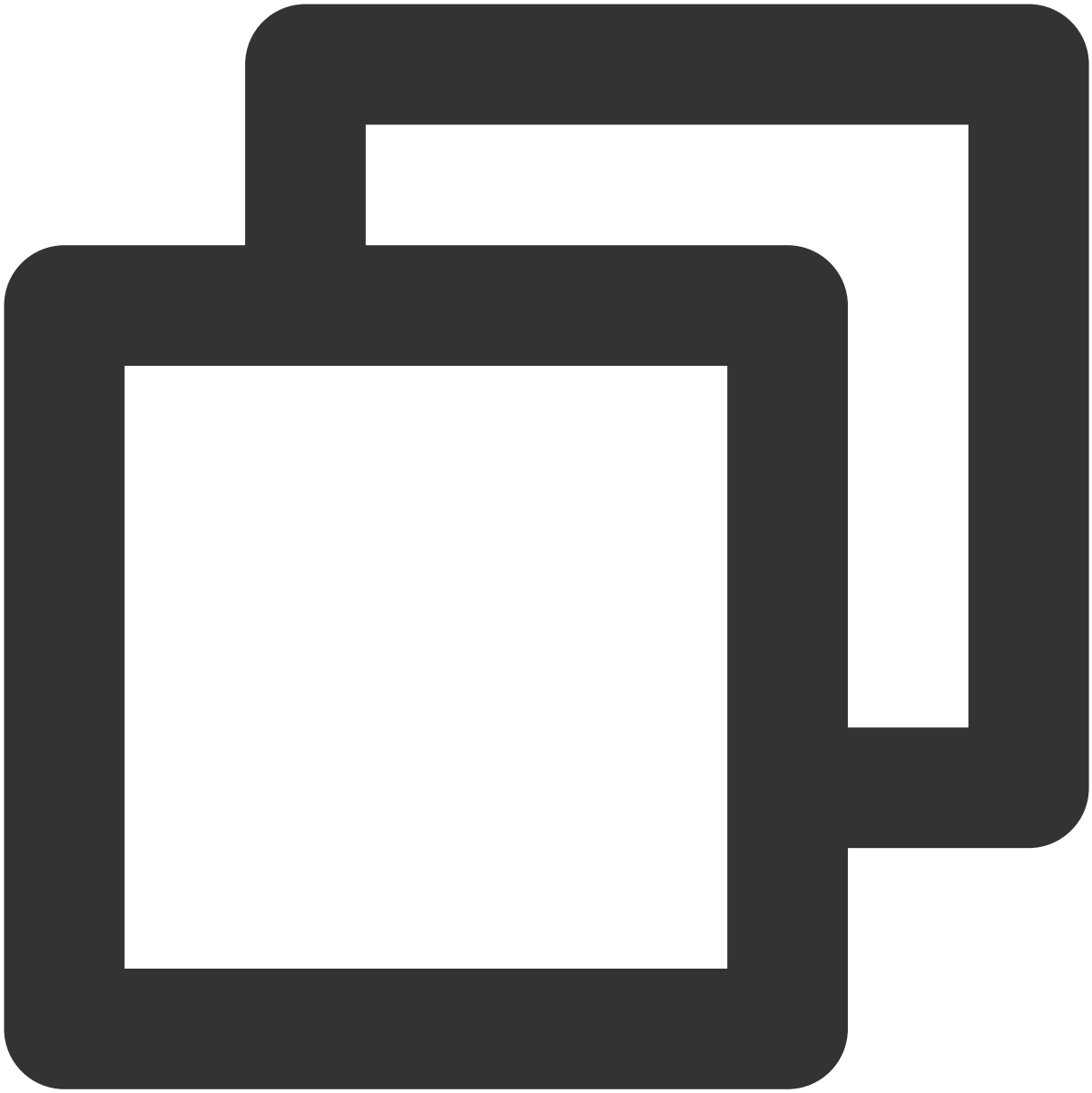
TUICallEngineのシングルトンを作成します。



```
TUICallEngine createInstance(Context context)
```

destroyInstance

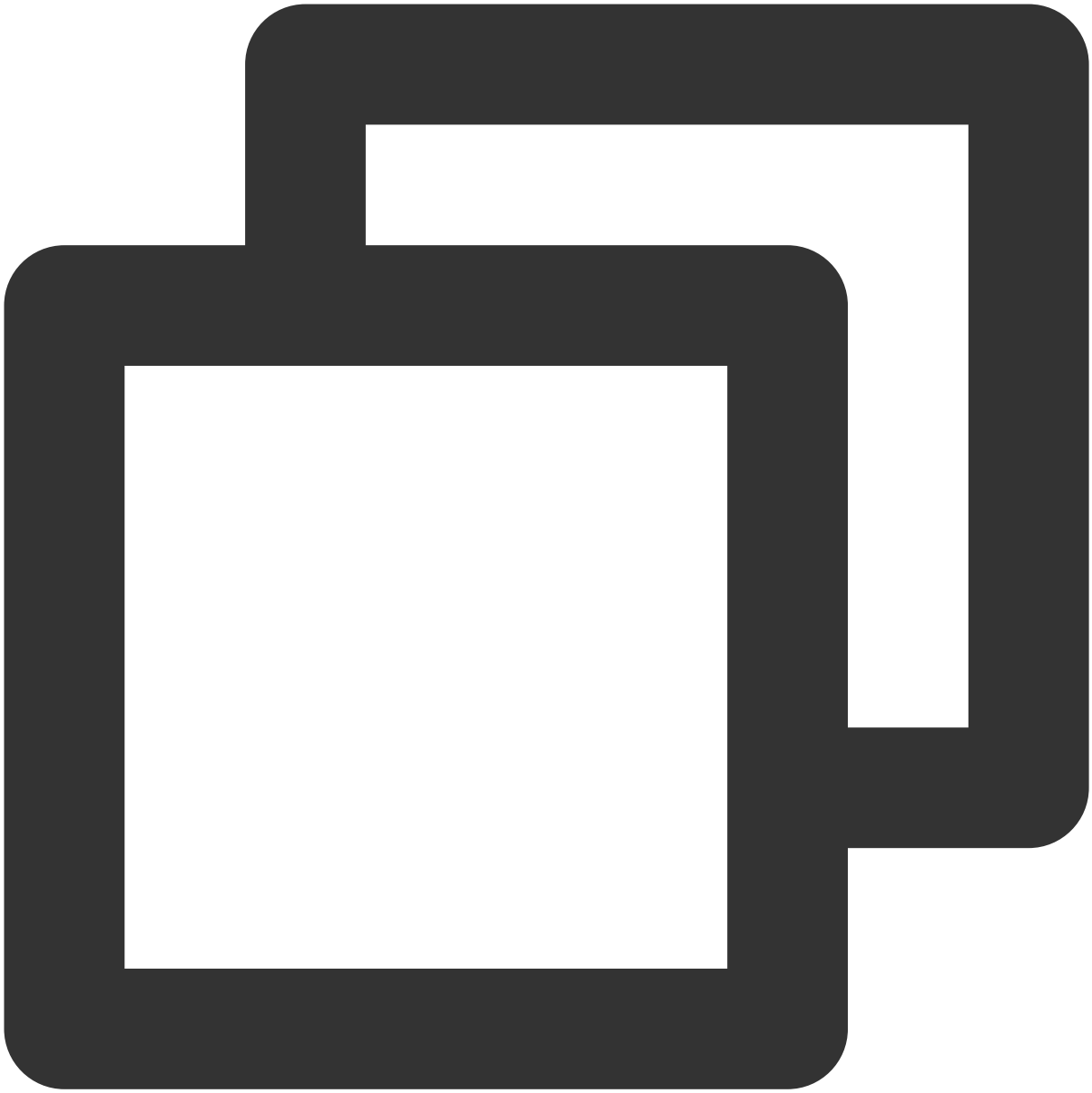
TUICallEngineのシングルトンを破棄します。



```
void destroyInstance();
```

init

関数を初期化します。すべての機能を使用する前にこの関数を呼び出し、通話サービス認証を含む初期化操作を完了させてください。



```
void init(int sdkAppId, String userId, String userSig, TUICommonDefine.Callback cal
```

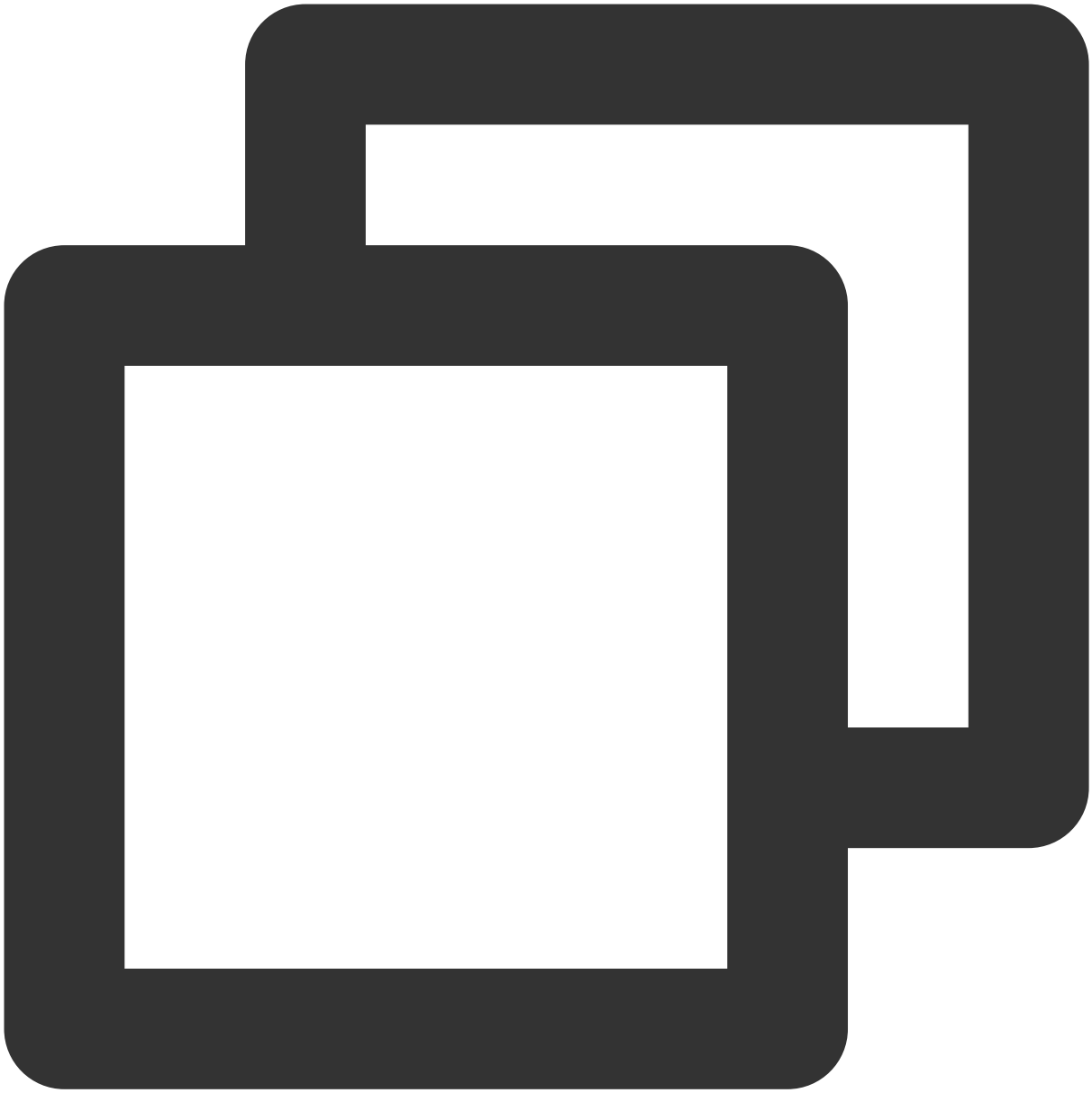
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
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。文字列タイプでは、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
callback	TUICommonDefine.Callback	初期化コールバック。 `onSuccess` は初期化に成功したことを表します

addObserver

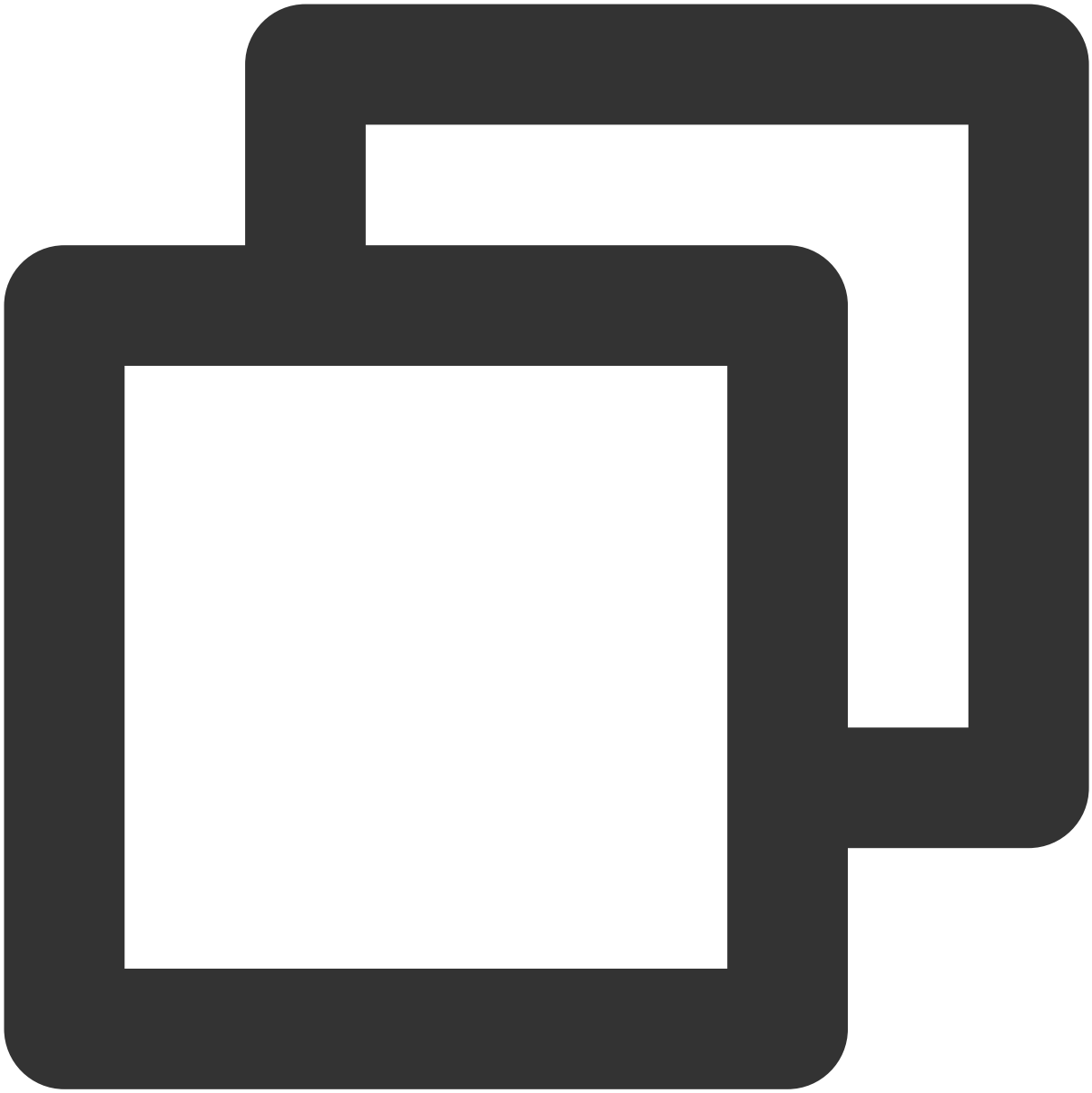
このコールバックインターフェースを追加すると、この応答を通じて `TUICallObserver` 関連のイベントコールバックを監視できます。



```
void addObserver(TUICallObserver observer);
```

removeObserver

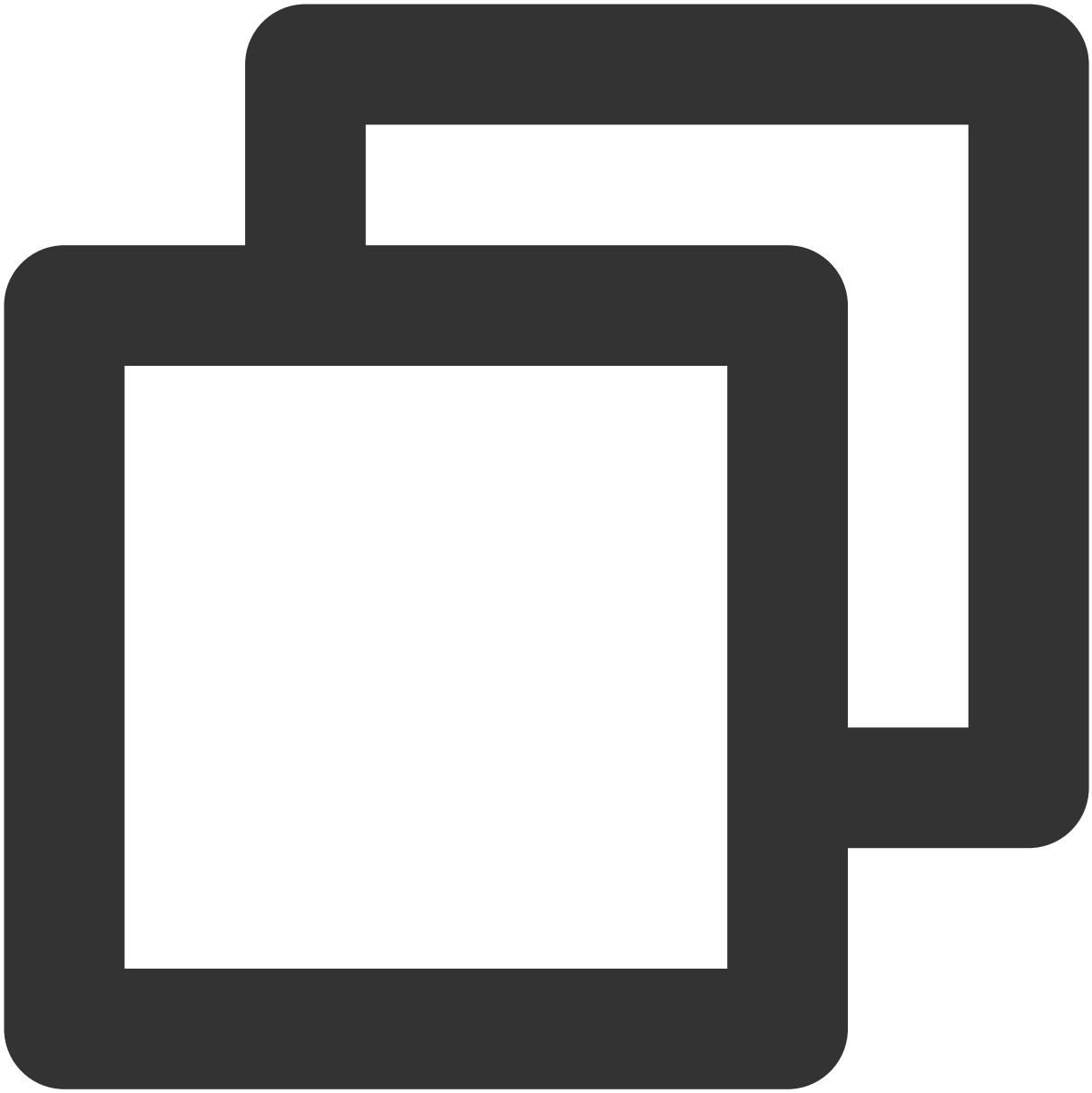
コールバックインターフェースを削除します。



```
void removeObserver(TUICallObserver observer);
```

call

電話をかけます（1v1通話）。



```
void call(TUICommonDefine.RoomId roomId, String userId, TUICallDefine.MediaType cal
          TUICallDefine.CallParams params, TUICommonDefine.Callback callback);
```

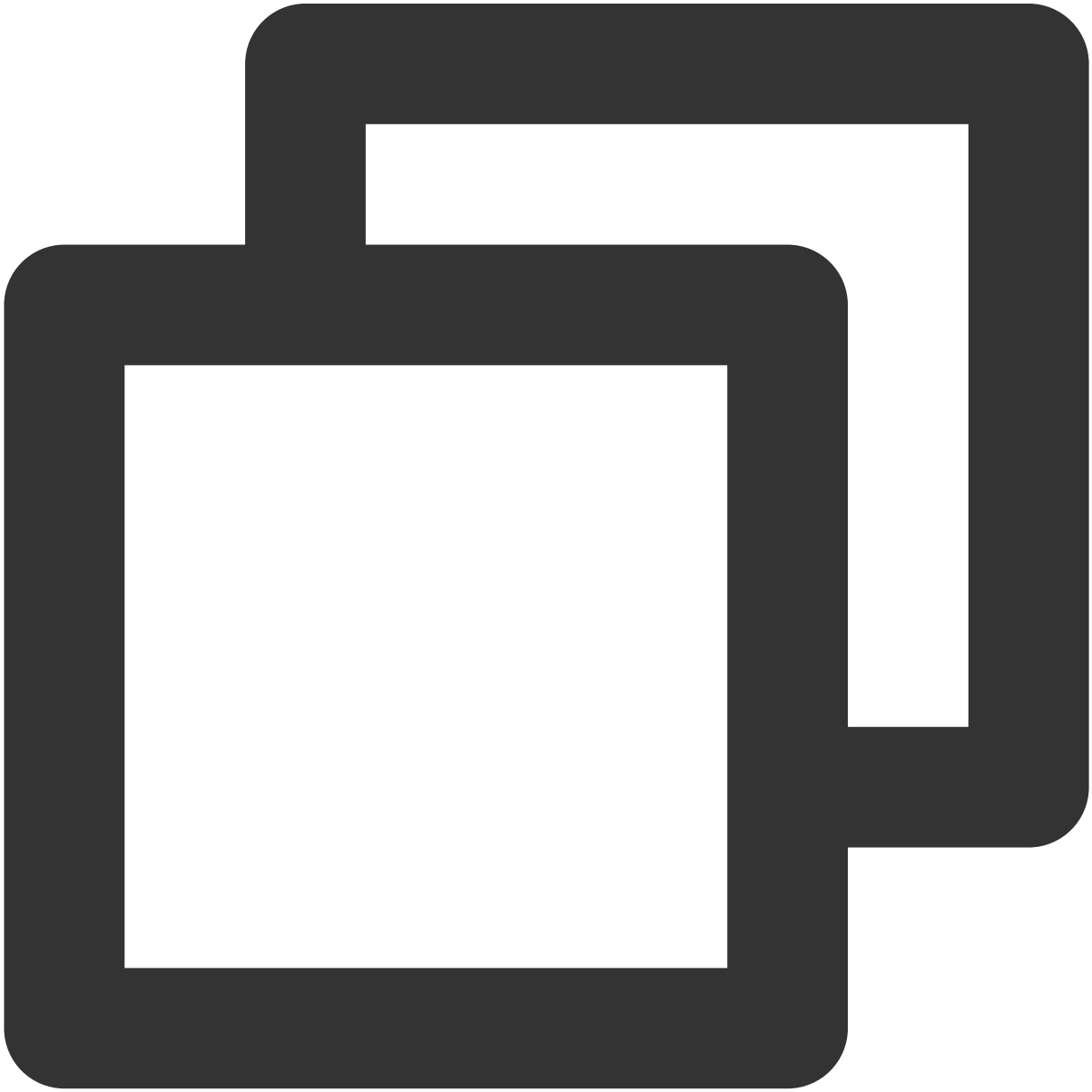
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
roomId	TUICommonDefine.RoomId	今回の通話のオーディオビデオルームID。現在は数字のルームナンバーのみサポートしています。文字列のルームナンバーは今後のバージョンでサポート予定です

userId	String	ターゲットユーザーのuserId
callMediaType	TUICallDefine.MediaType	通話のメディアタイプ。例：ビデオ通話、音声通話など
params	TUICallDefine.CallParams	通話パラメータ拡張フィールド。例：カスタムコンテンツのオフラインプッシュ

groupCall

グループ通話を開始します。注意：グループ通話を使用する前にIMグループを作成する必要があります。作成済みの場合は無視してください。



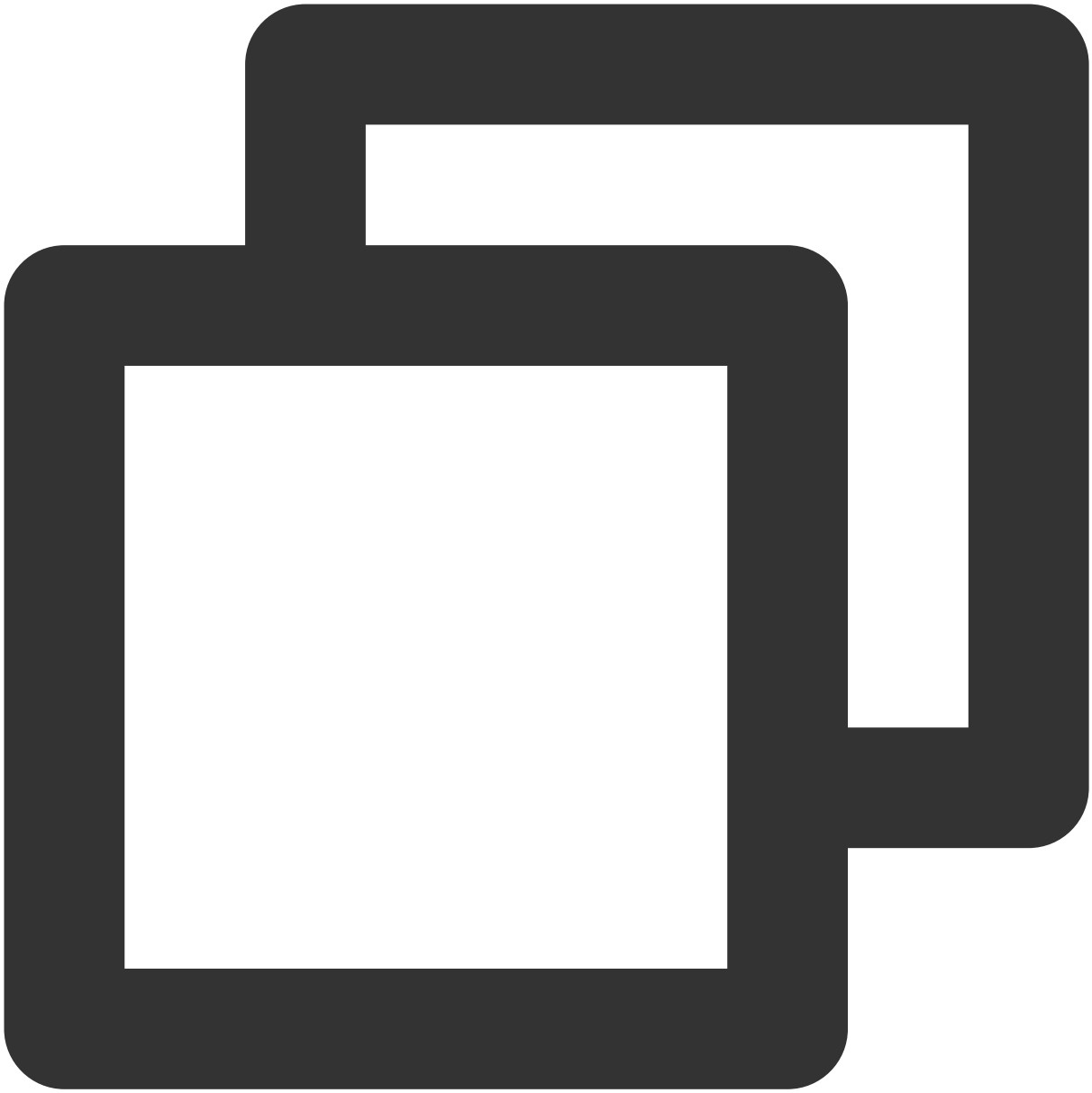
```
void groupCall(TUICommonDefine.RoomId roomId, String groupId, List<String> userIdLi
               TUICallDefine.MediaType callMediaType, TUICallDefine.CallParams para
               TUICommonDefine.Callback callback);
```

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
roomId	TUICommonDefine.RoomId	今回の通話のオーディオビデオルームID。現在は数字のルームナンバーのみサポートしています。文字列のルームナンバーは今後のバージョンでサポート予定です
groupId	String	今回のグループ通話のグループID
userIdList	List	ターゲットユーザーのuserIdリスト
callMediaType	TUICallDefine.MediaType	通話のメディアタイプ。例：ビデオ通話、音声通話など
params	TUICallDefine.CallParams	通話パラメータ拡張フィールド。例：カスタムコンテンツのオフラインプッシュ

accept

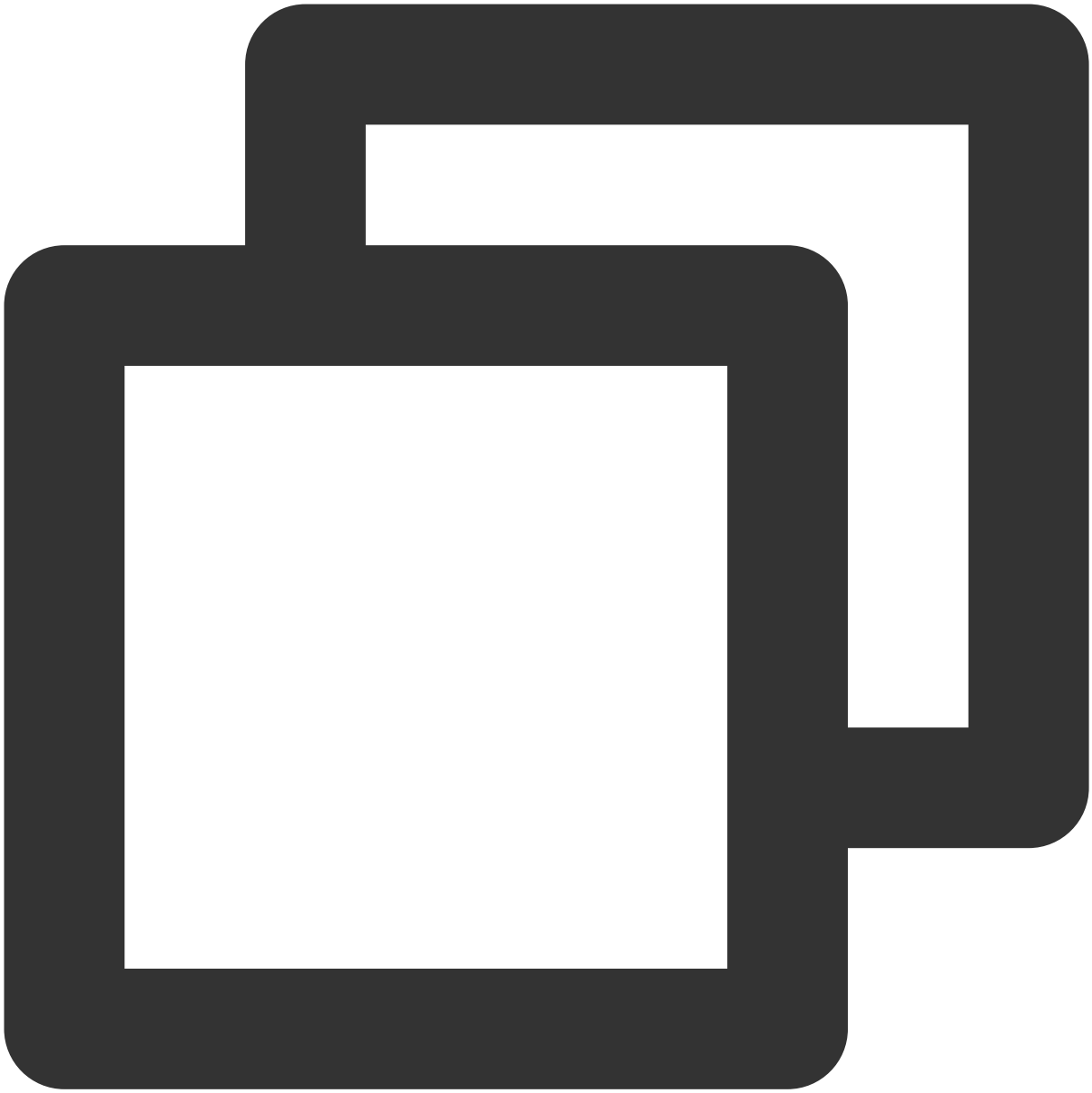
現在の通話を受信します。着呼側として `onCallReceived()` のコールバックを受信した場合は、この関数を呼び出して通話に応答することができます。



```
void accept(TUICommonDefine.Callback callback);
```

reject

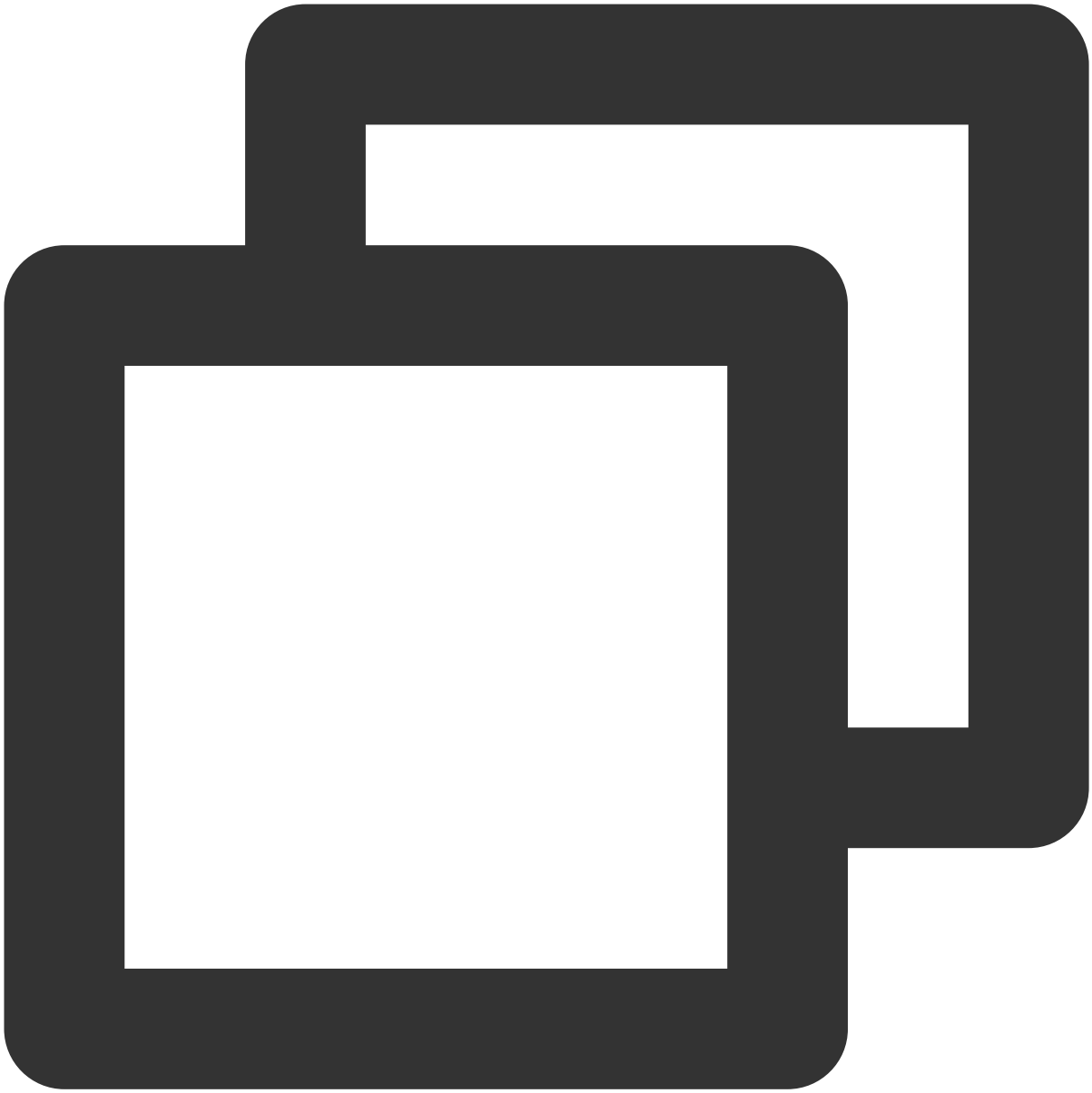
現在の通話を拒否します。着呼側として `onCallReceived()` のコールバックを受信した場合は、この関数を呼び出して通話を拒否することができます。



```
void reject(TUICommonDefine.Callback callback);
```

ignore

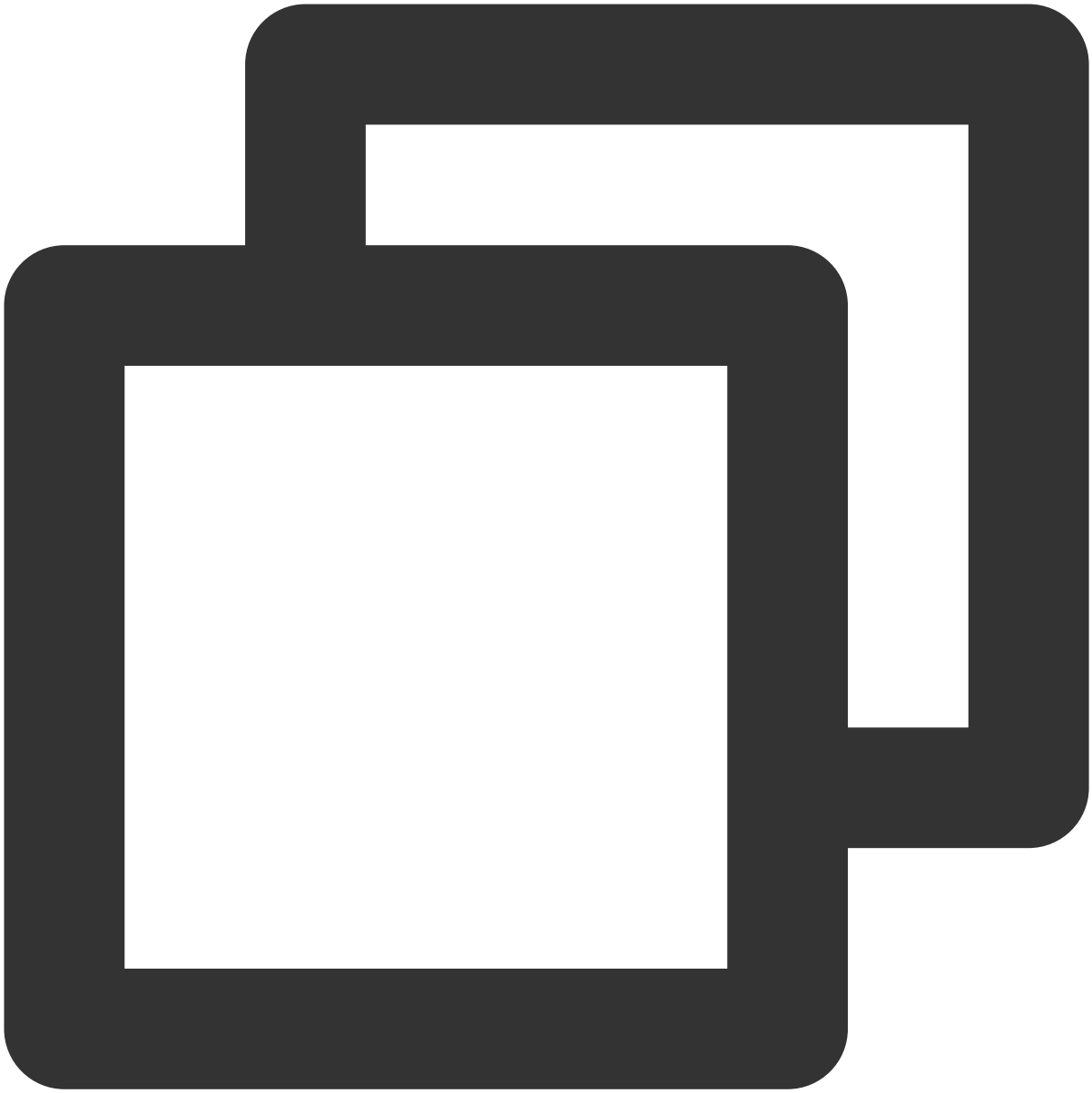
現在の通話を無視します。着呼側としてonCallReceived()のコールバックを受信した場合は、この関数を呼び出して通話を無視することができ、このとき発呼側はonUserLineBusyのコールバックを受信します。
備考：業務内にライブストリーミング、ミーティングなどのシーンがある場合、ライブストリーミング/ミーティング中の場合もこの関数を呼び出して通話を無視することができます。



```
void ignore(TUICommonDefine.Callback callback);
```

hangup

現在の通話を終了します。通話中である場合は、この関数を呼び出して通話を終了できます。

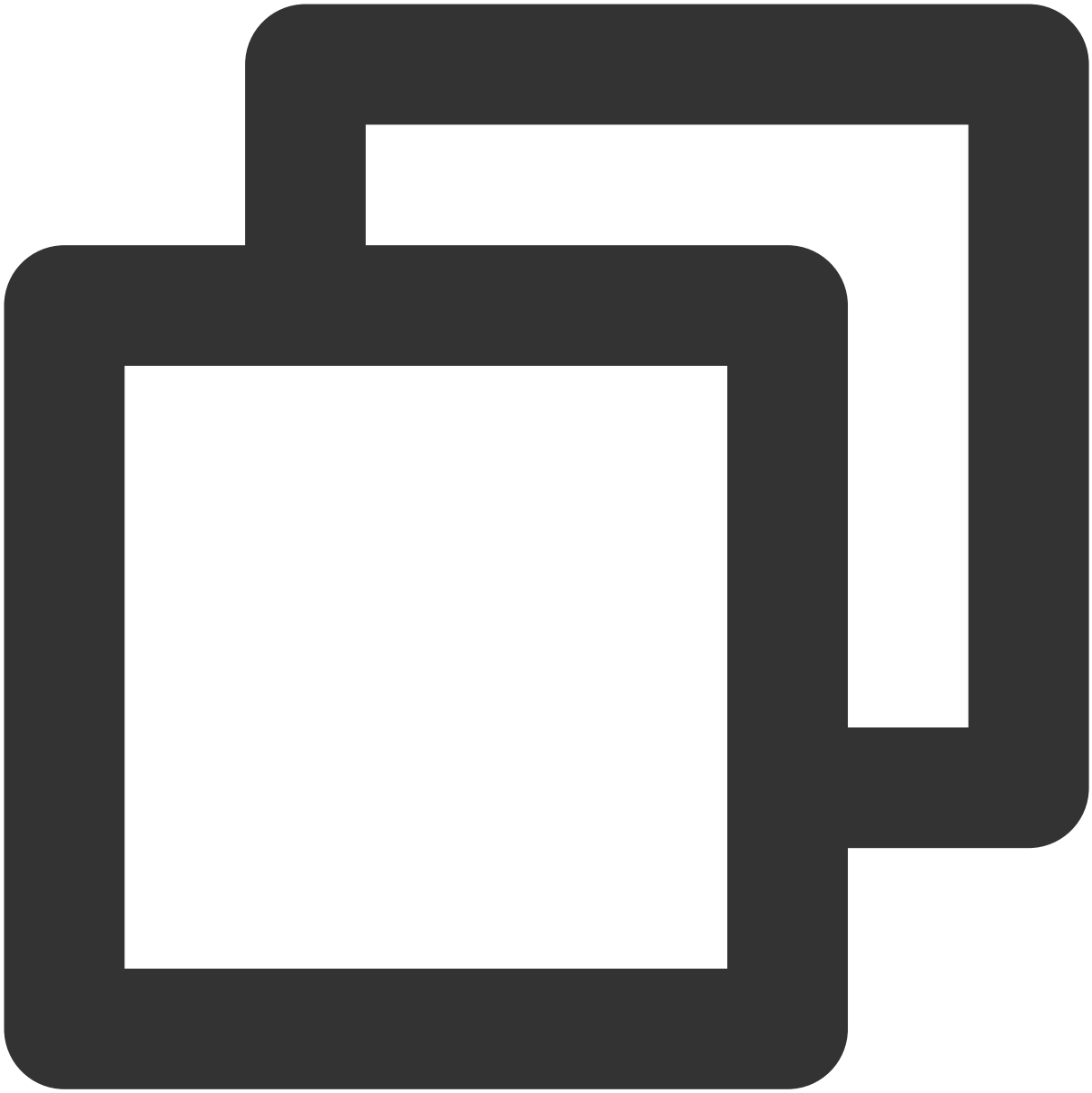


```
void hangup(TUICommonDefine.Callback callback);
```

inviteUser

今回のグループ通話にユーザーを招待します。

ユースケース：グループ通話中のユーザーが自主的に他の人を招待する場合に使用します。



```
void inviteUser(List<String> userIdList, TUICallDefine.CallParams params,
                TUICommonDefine.ValueCallback callback);
```

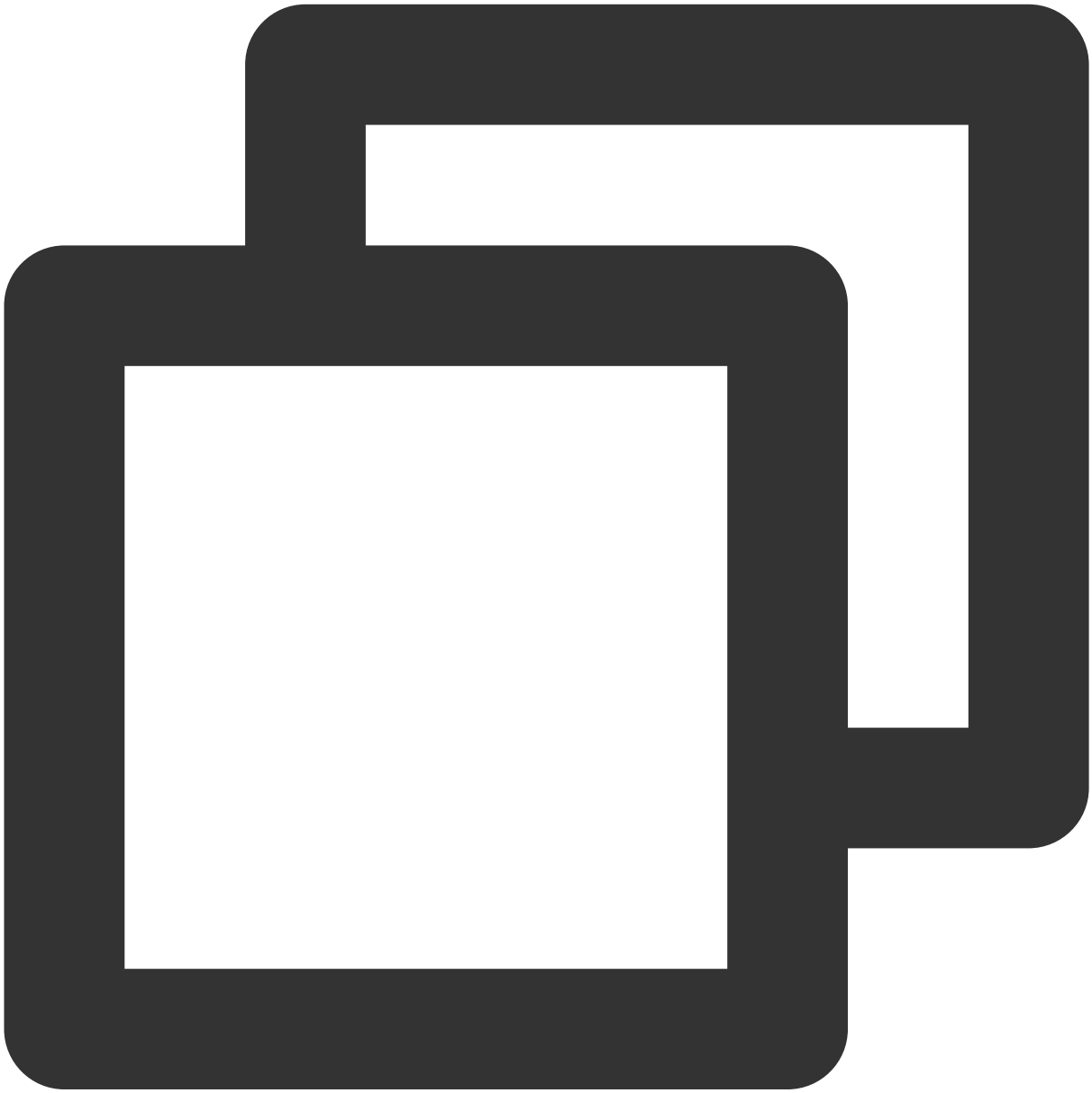
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userIdList	List	ターゲットユーザーのuserIdリスト
params	TUICallDefine.CallParams	通話パラメータ拡張フィールド。例：カスタムコンテンツのオフ

		ラインプッシュ

joinInGroupCall

今回のグループ通話に自主的に参加します。
ユースケース：グループ内のユーザーが今回のグループ通話に自主的に参加する場合に使用します。



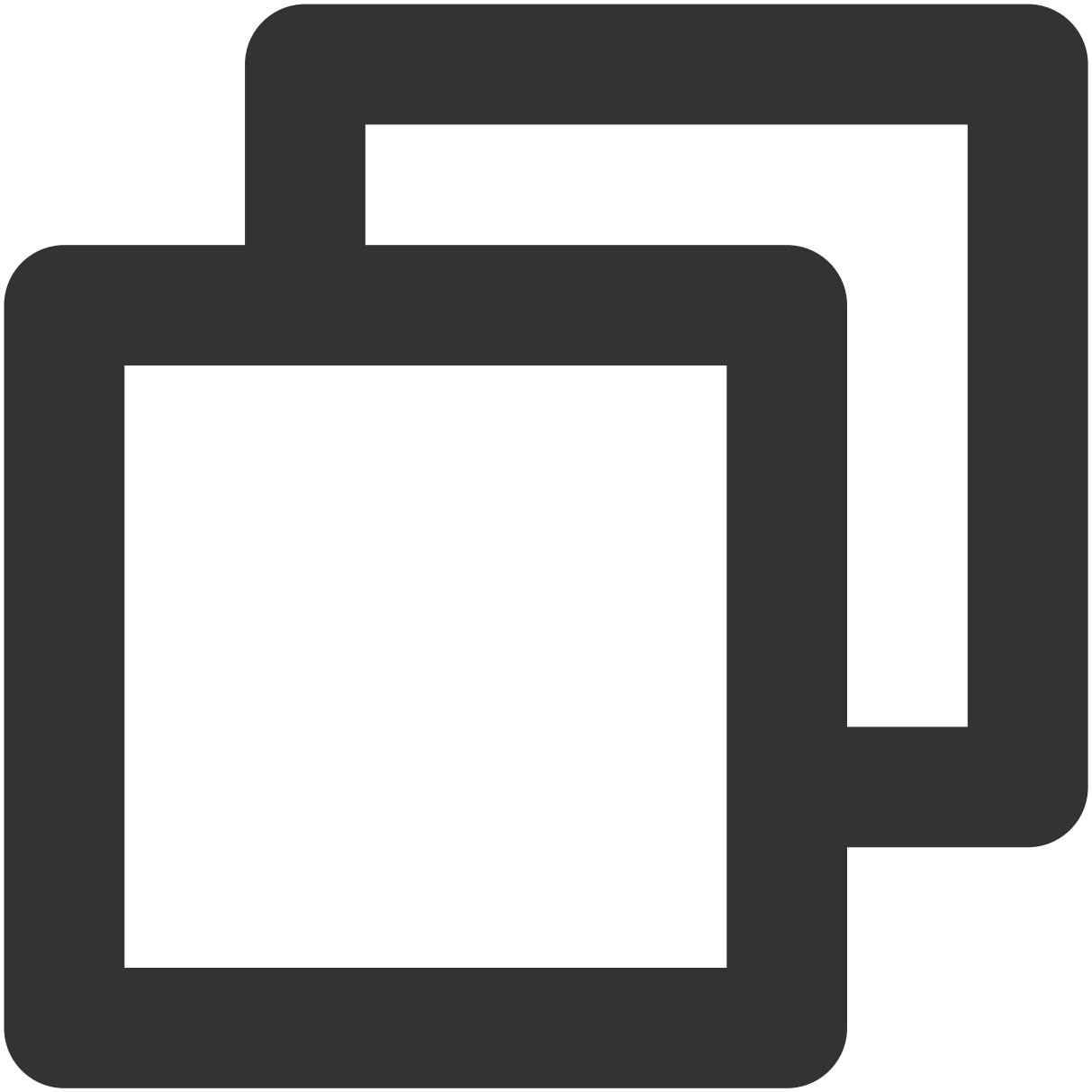
```
void joinInGroupCall(TUICommonDefine.RoomId roomId, String groupId,
                     TUICallDefine.MediaType callMediaType, TUICommonDefine.Callbac
```

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
roomId	TUICommonDefine.RoomId	今回の通話のオーディオビデオルームID。現在は数字のルームナンバーのみサポートしています。文字列のルームナンバーは今後のバージョンでサポート予定です
groupId	String	今回のグループ通話のグループID
callMediaType	TUICallDefine.MediaType	通話のメディアタイプ。ビデオ通話、音声通話など

switchCallMediaType

ビデオ通話を音声通話へ切り替えます。



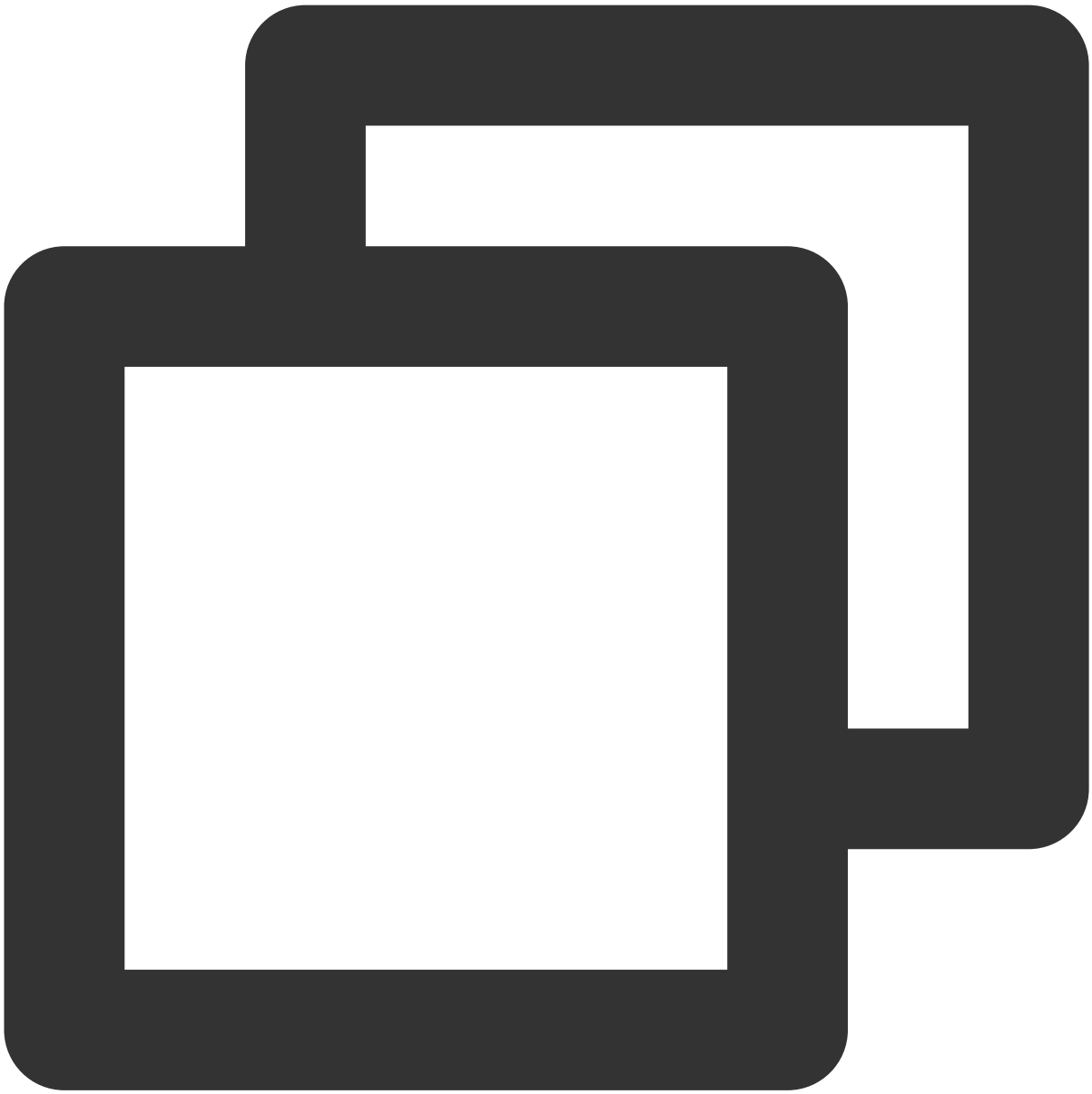
```
void switchCallMediaType(TUICallDefine.MediaType callMediaType);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callMediaType	TUICallDefine.MediaType	通話のメディアタイプ。ビデオ通話、音声通話など

startRemoteView

リモートユーザーのビデオデータのサブスクリプションを開始します。このインターフェースはsetRenderViewの後に呼び出します。



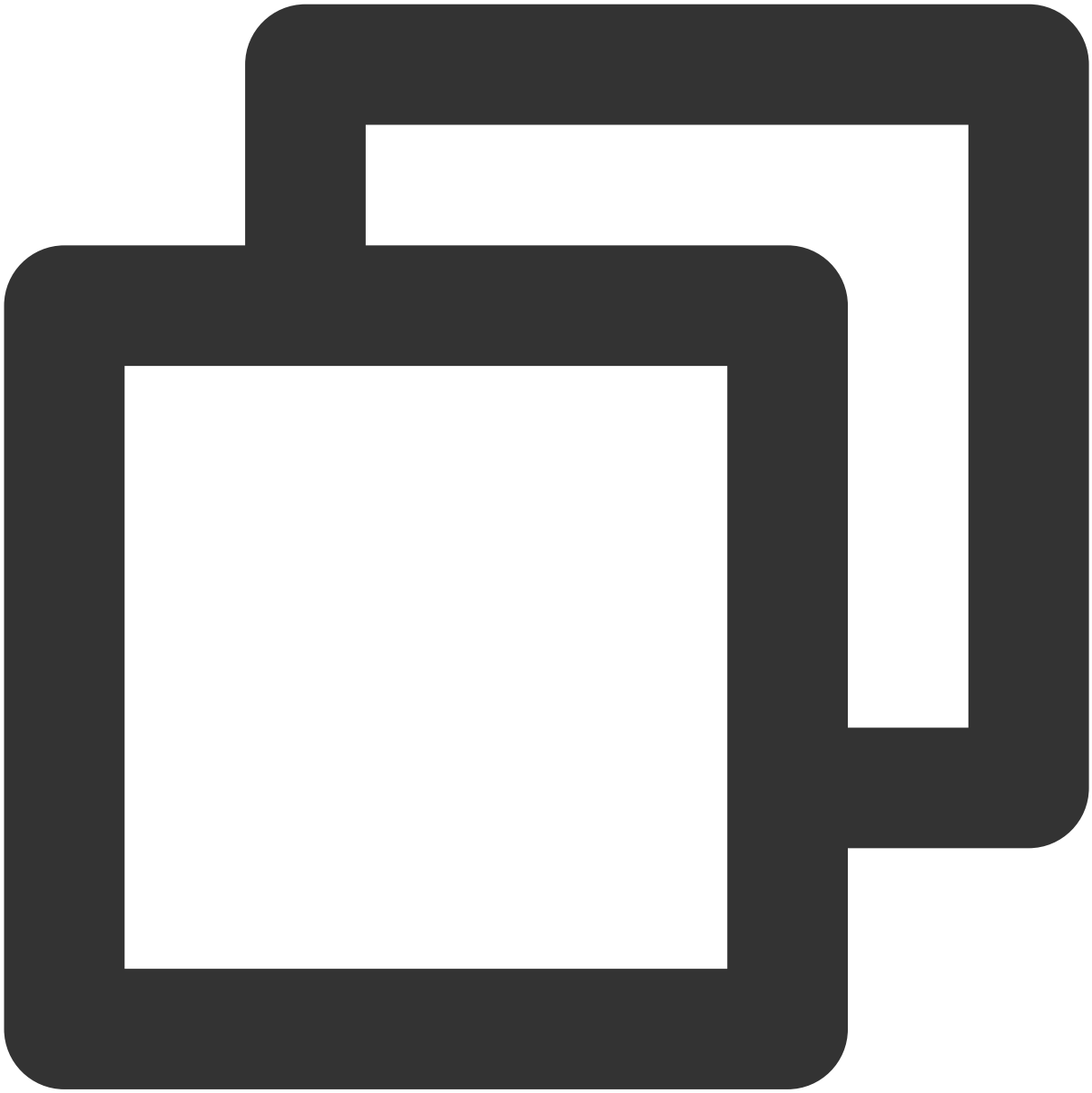
```
void startRemoteView(String userId, TUIVideoView videoView, TUICommonDefine.PlayCal
```

パラ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	ターゲットユーザーのuserId
videoView	TUIVideoView	レンダリング対象のビュー

stopRemoteView

リモートユーザーのビデオデータのサブスクリプションを停止します。



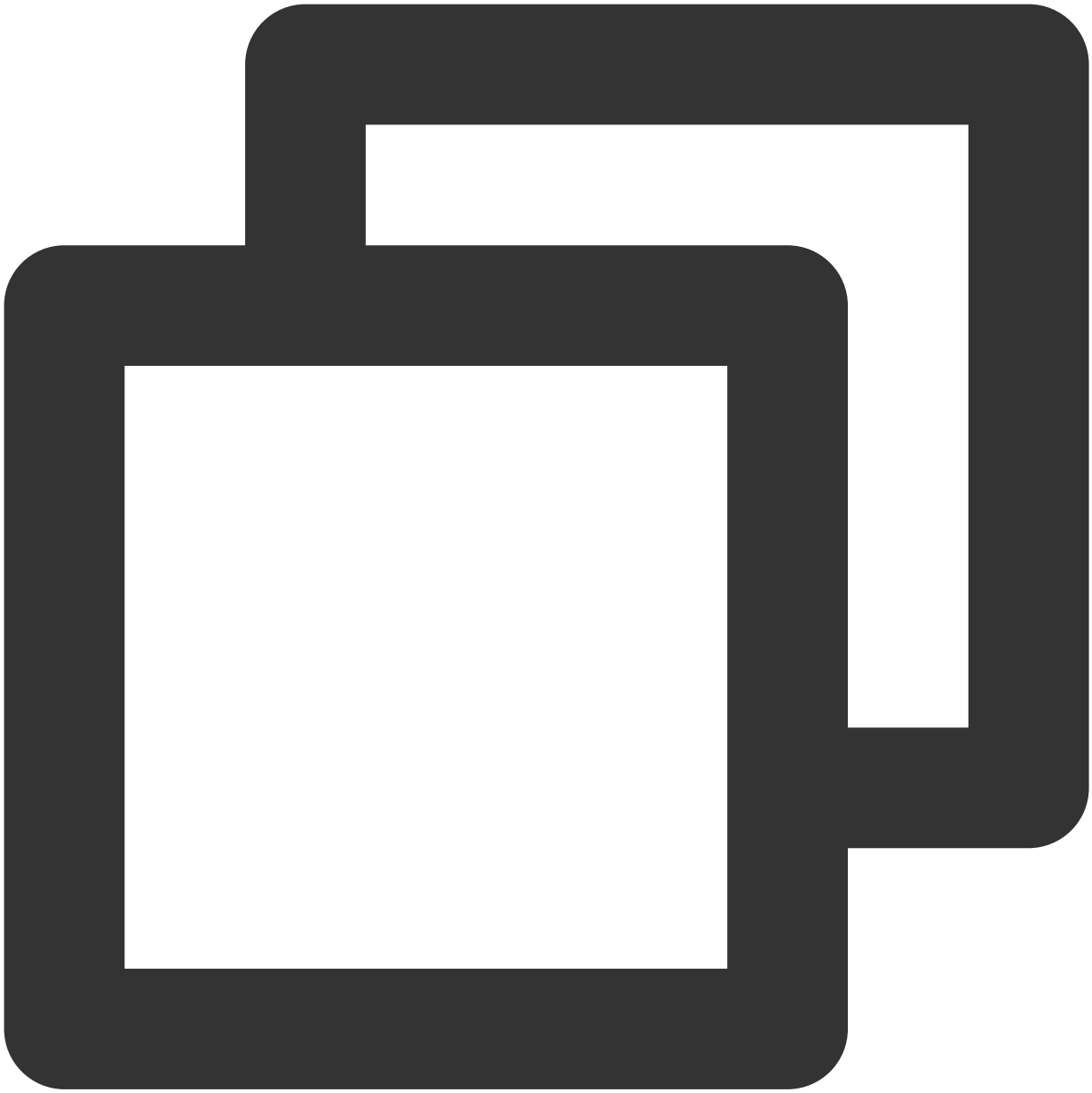
```
void stopRemoteView(String userId);
```

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	ターゲットユーザーのuserId

openCamera

カメラをオンにする



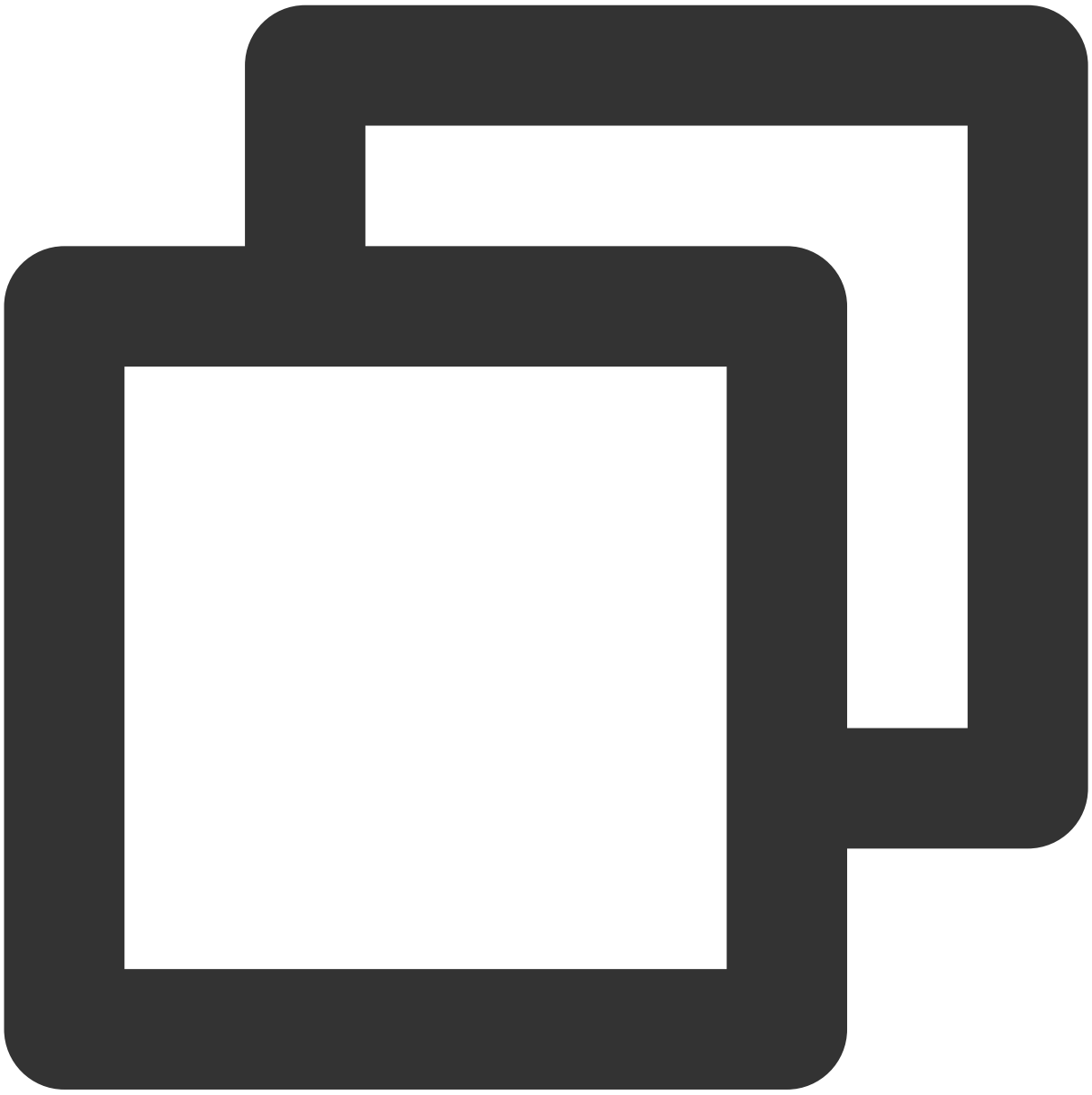
```
void openCamera(TUICommonDefine.Camera camera, TUIVideoView videoView, TUICommonDef
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
camera	TUICommonDefine.Camera	フロントカメラ/リアカメラ
videoView	TUIVideoView	レンダリング対象のビュー

closeCamera

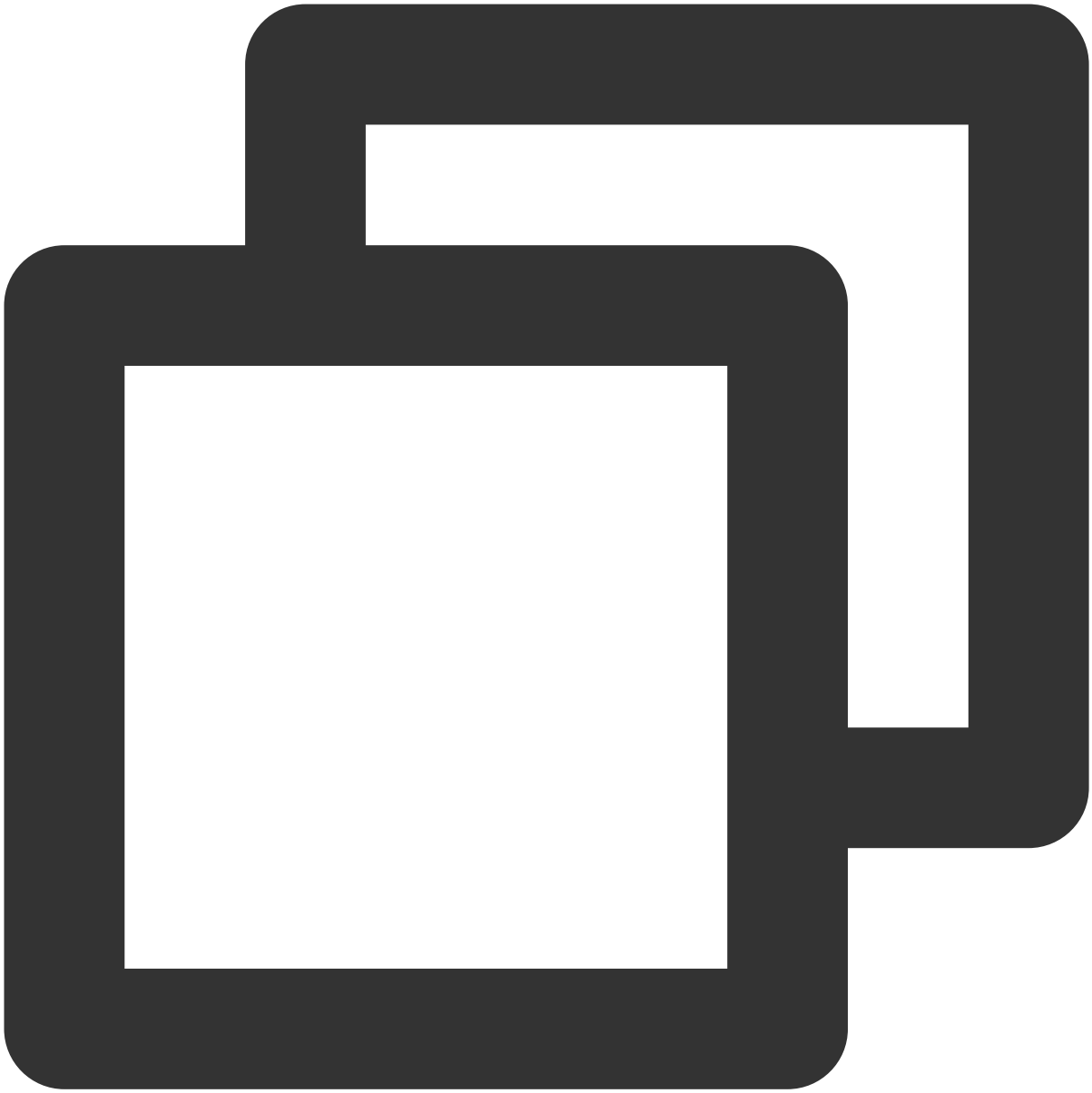
カメラをオフにします。



```
void closeCamera();
```

switchCamera

フロント/リアカメラを切り替えます。



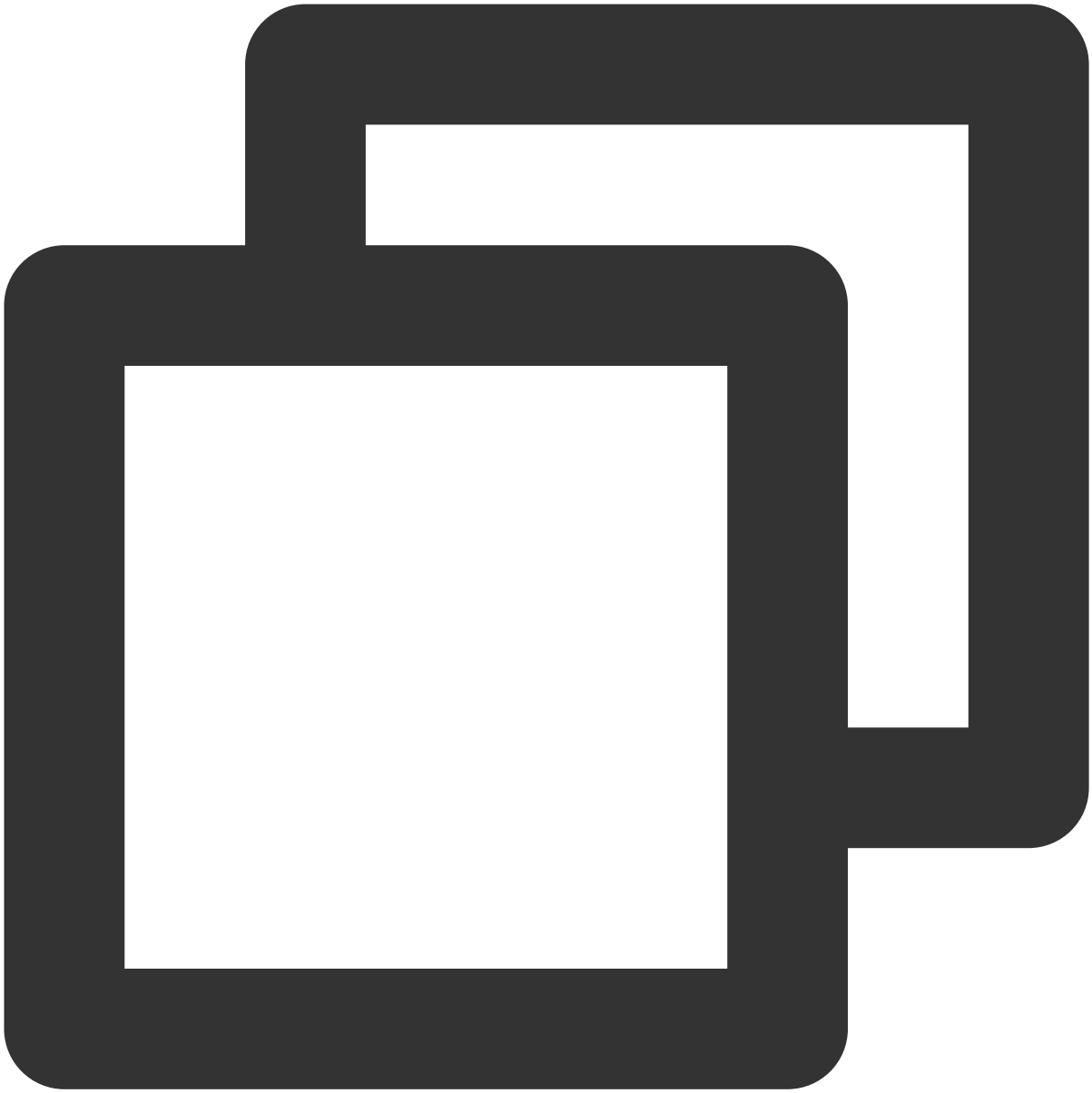
```
void switchCamera(TUICommonDefine.Camera camera);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
camera	TUICommonDefine.Camera	フロントカメラ/リアカメラ

openMicrophone

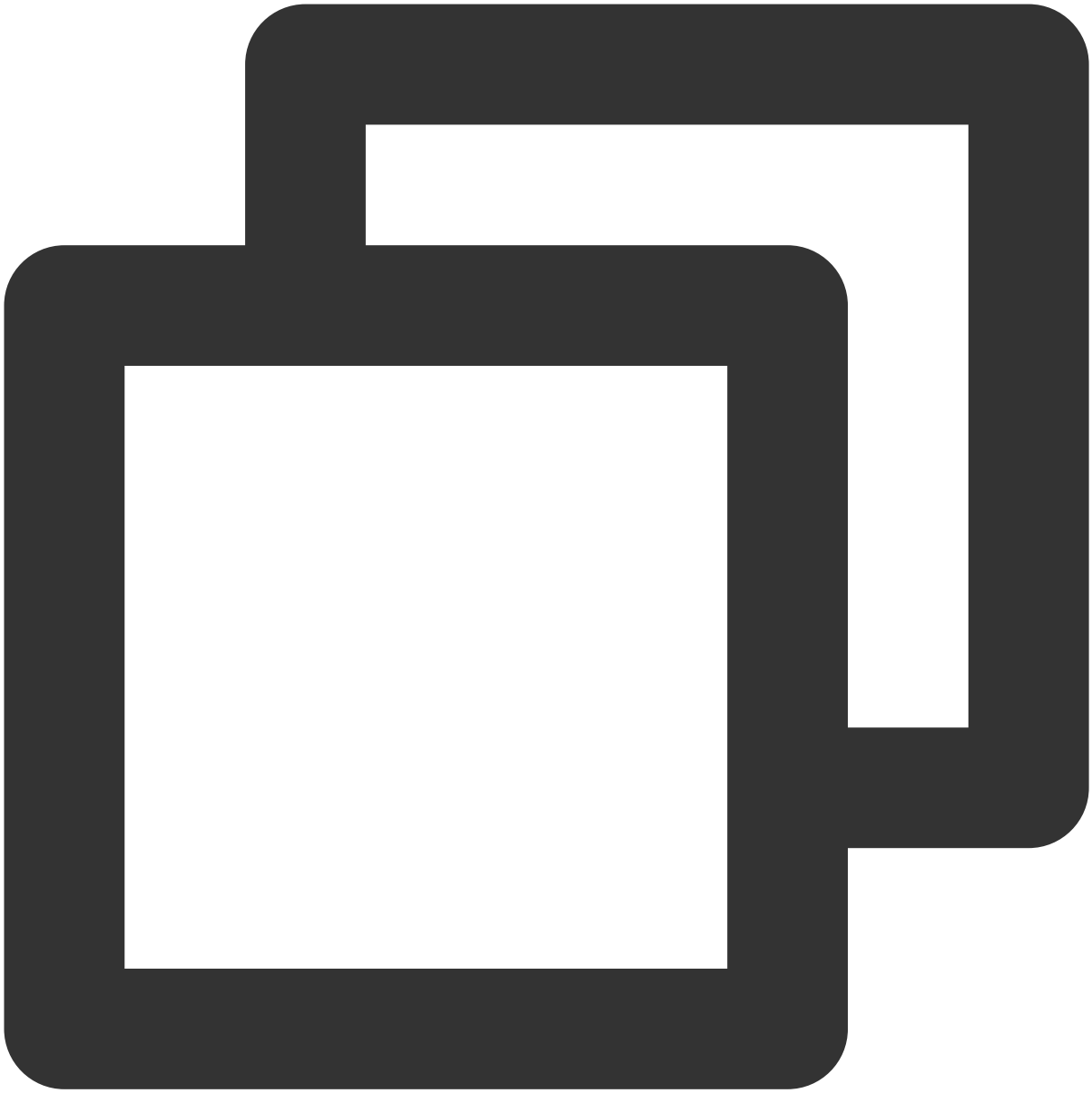
マイクをオンにします



```
void openMicrophone(TUICommonDefine.Callback callback);
```

closeMicrophone

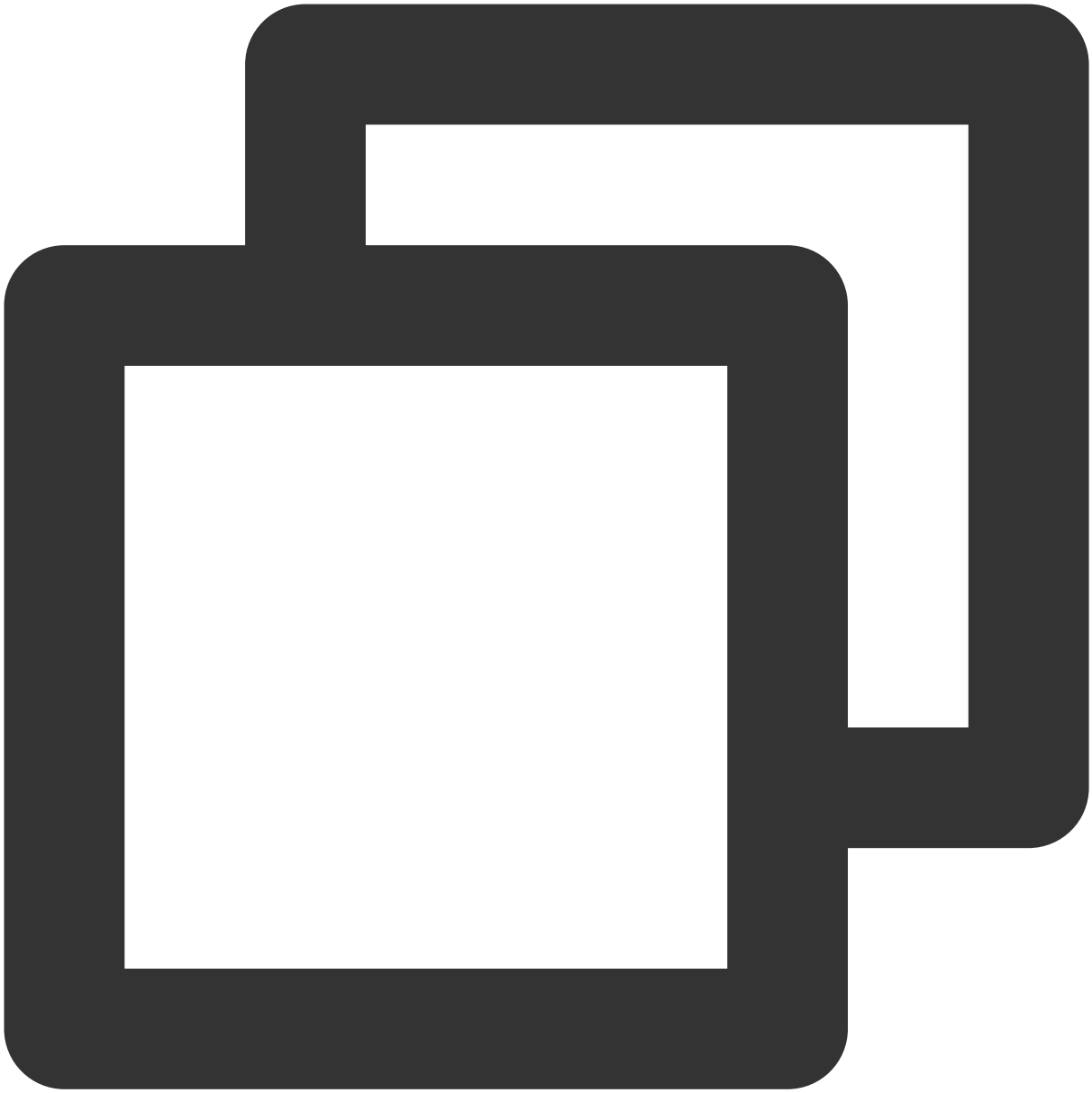
マイクをオフにします。



```
void closeMicrophone();
```

selectAudioPlaybackDevice

オーディオ再生デバイスを選択します。 現在はヘッドホン、スピーカーをサポートしています。通話のシーンでは、このインターフェースを使用してハンズフリーモードのオン/オフが行えます。



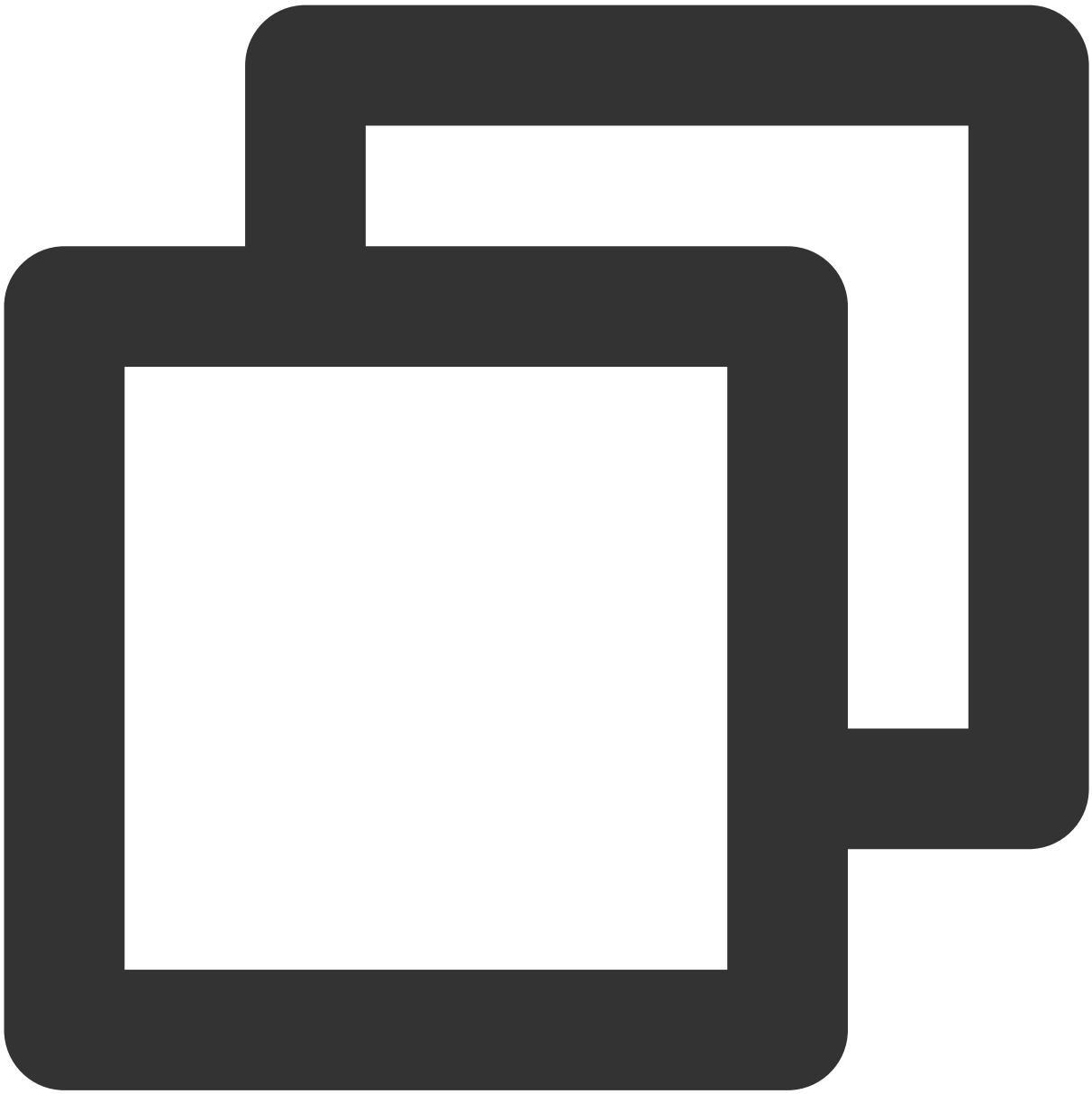
```
void selectAudioPlaybackDevice(TUICommonDefine.AudioPlaybackDevice device);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
device	TUICommonDefine.AudioPlaybackDevice	ヘッドホン/スピーカー

setSelfInfo

ユーザーニックネーム、プロフィール画像を設定します。ユーザーニックネームは500バイト以内、ユーザープロフィール画像はURL形式でなければなりません。



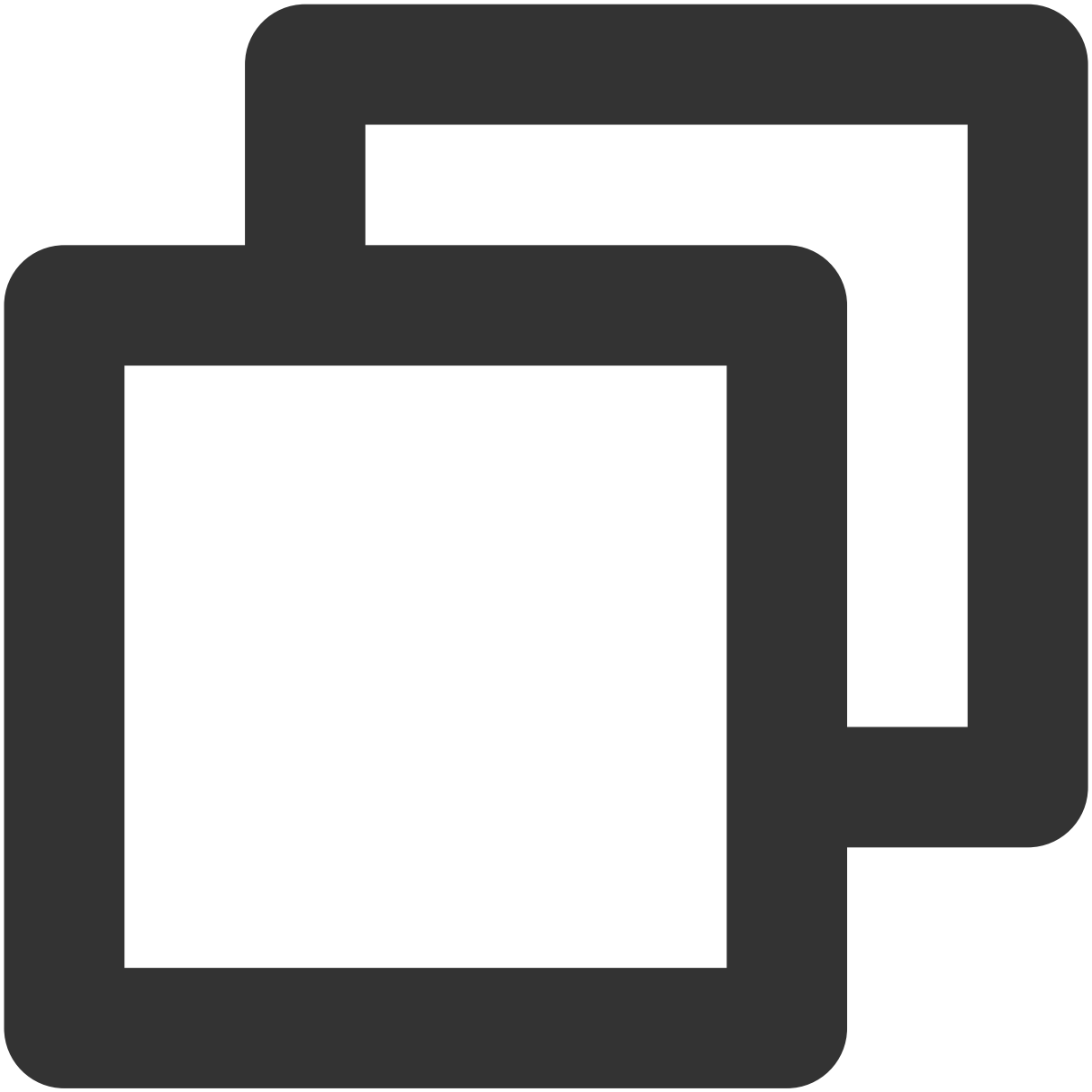
```
void setSelfInfo(String nickname, String avatar, TUICommonDefine.Callback callback)
```

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
nickname	String	ユーザーニックネーム
avatar	String	ユーザープロフィール画像（形式はURL）

enableMultiDeviceAbility

TUICallEngineのマルチデバイスログインモードをオン/オフします（プレミアム版パッケージのみサポート）。



```
void enableMultiDeviceAbility(boolean enable, TUICommonDefine.Callback callback);
```

TUICallObserver

最終更新日：：2024-07-19 14:53:21

TUICallObserver APIの概要

TUICallObserverはTUICallEngineに対応するコールバックイベントクラスです。このコールバックによって、関心のあるコールバックイベントを監視することができます。

コールバックイベントの概要

API	説明
onError	通話中のエラーコールバック
onCallReceived	通話リクエストのコールバック
onCallCancelled	通話キャンセルのコールバック
onCallBegin	通話接続のコールバック
onCallEnd	通話終了のコールバック
onCallMediaTypeChanged	通話メディアタイプ変更発生のコールバック
onUserReject	xxxxユーザーによる通話拒否のコールバック
onUserNoResponse	xxxxユーザーの応答なしのコールバック
onUserLineBusy	xxxxユーザーが通話中である場合のコールバック
onUserJoin	xxxxユーザーの通話参加のコールバック
onUserLeave	xxxxユーザーの通話からの退出のコールバック
onUserVideoAvailable	xxxユーザーのビデオストリームの有無のコールバック
onUserAudioAvailable	xxxユーザーのオーディオストリームの有無のコールバック
onUserVoiceVolumeChanged	全ユーザーの音量レベルフィードバックのコールバック
onUserNetworkQualityChanged	全ユーザーのネットワーク品質フィードバックのコールバック

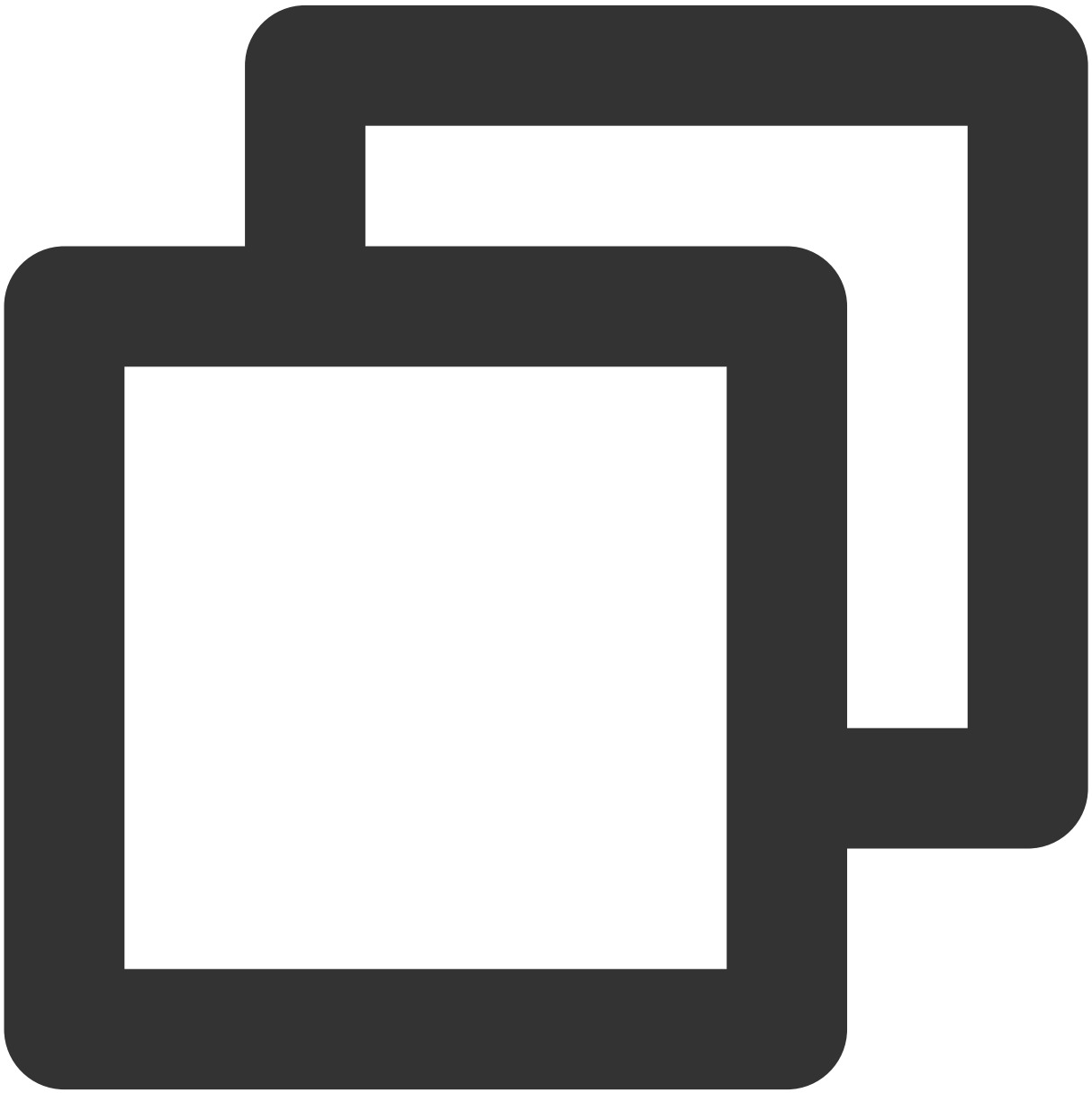
コールバックイベントの詳細

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。



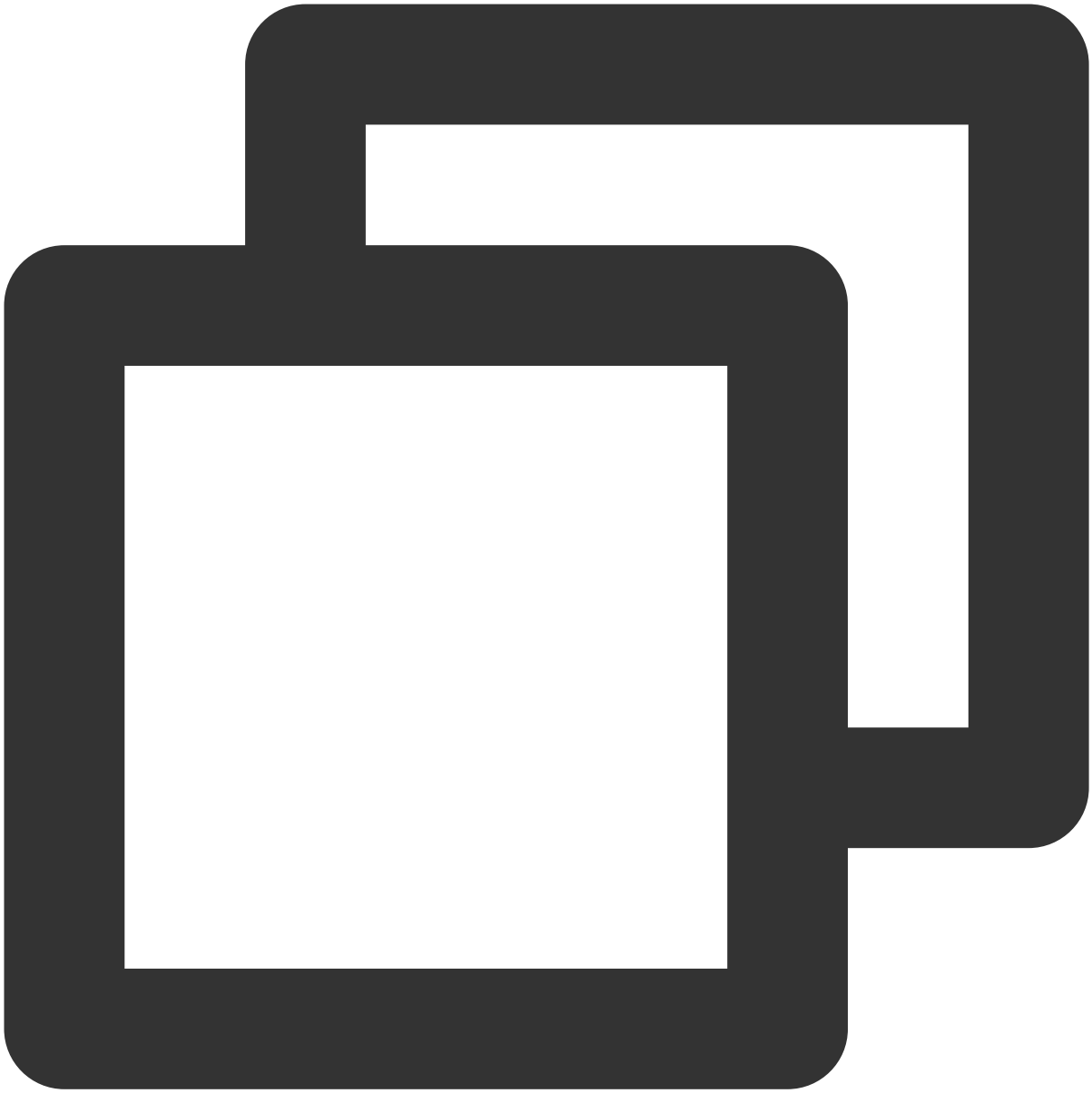
```
void onError(int code, String msg);
```

パラ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
msg	String	エラー情報

onCallReceived

新しい通話リクエストコールバックを受信します。着呼側が受信します。このイベントを監視することで、通話応答画面を表示するかどうかを決定できます。



```
void onCallReceived(String callerId, List<String> calleeIdList, String groupId,
                    TUICallDefine.MediaType callMediaType);
```

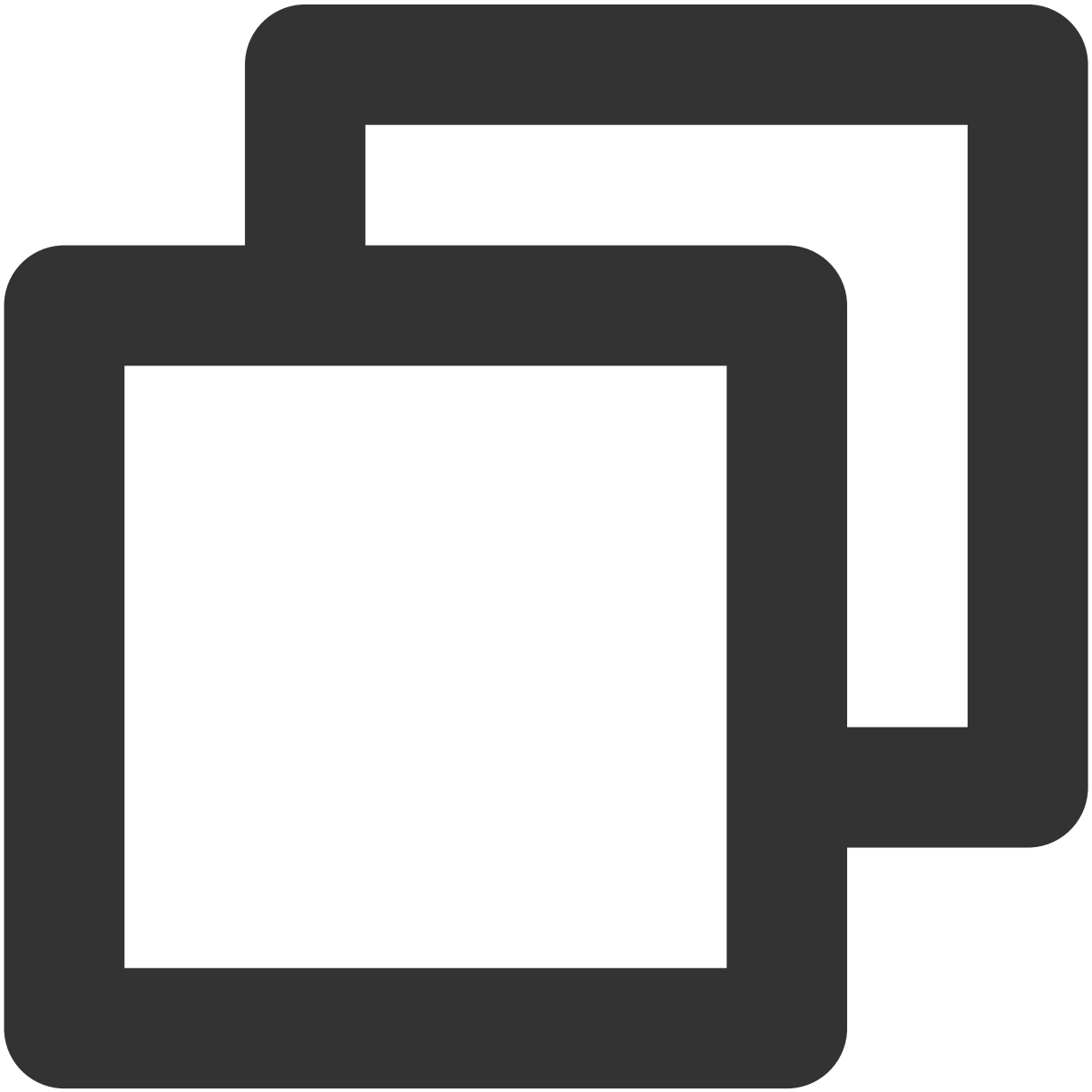
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callerId	String	発呼側ID（招待者）
calleeIdList	List	着呼側IDリスト（被招待者）

groupId	String	グループ通話ID
callMediaType	TUICallDefine.MediaType	通話のメディアタイプ。ビデオ通話、音声通話など

onCallCancelled

今回の通話が発呼側からキャンセルされたことを表します（キャンセルの原因は発呼側の自主的なキャンセル、または通話タイムアウトによるキャンセルの両方の可能性があります）。着呼側が受信します。このイベントを監視することで、未応答通話などに類似した表示ロジックを実現できます。



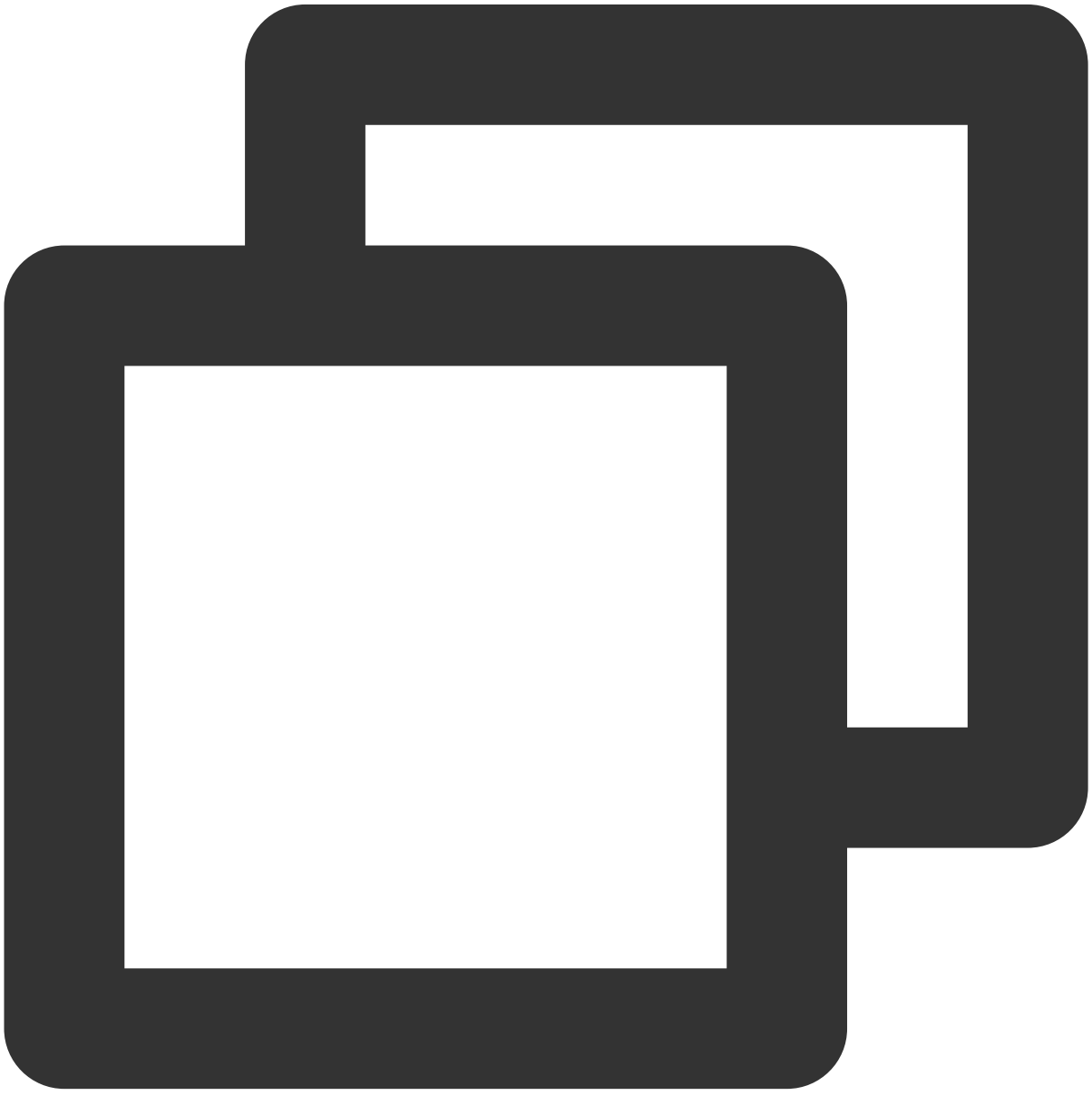
```
void onCallCancelled(String callerId);
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callerId	String	キャンセルしたユーザーのID

onCallBegin

通話接続を表します。発呼側と着呼側がどちらも受信できます。このイベントを監視することで、クラウドレコーディング、コンテンツ審査などのフローを開始できます。



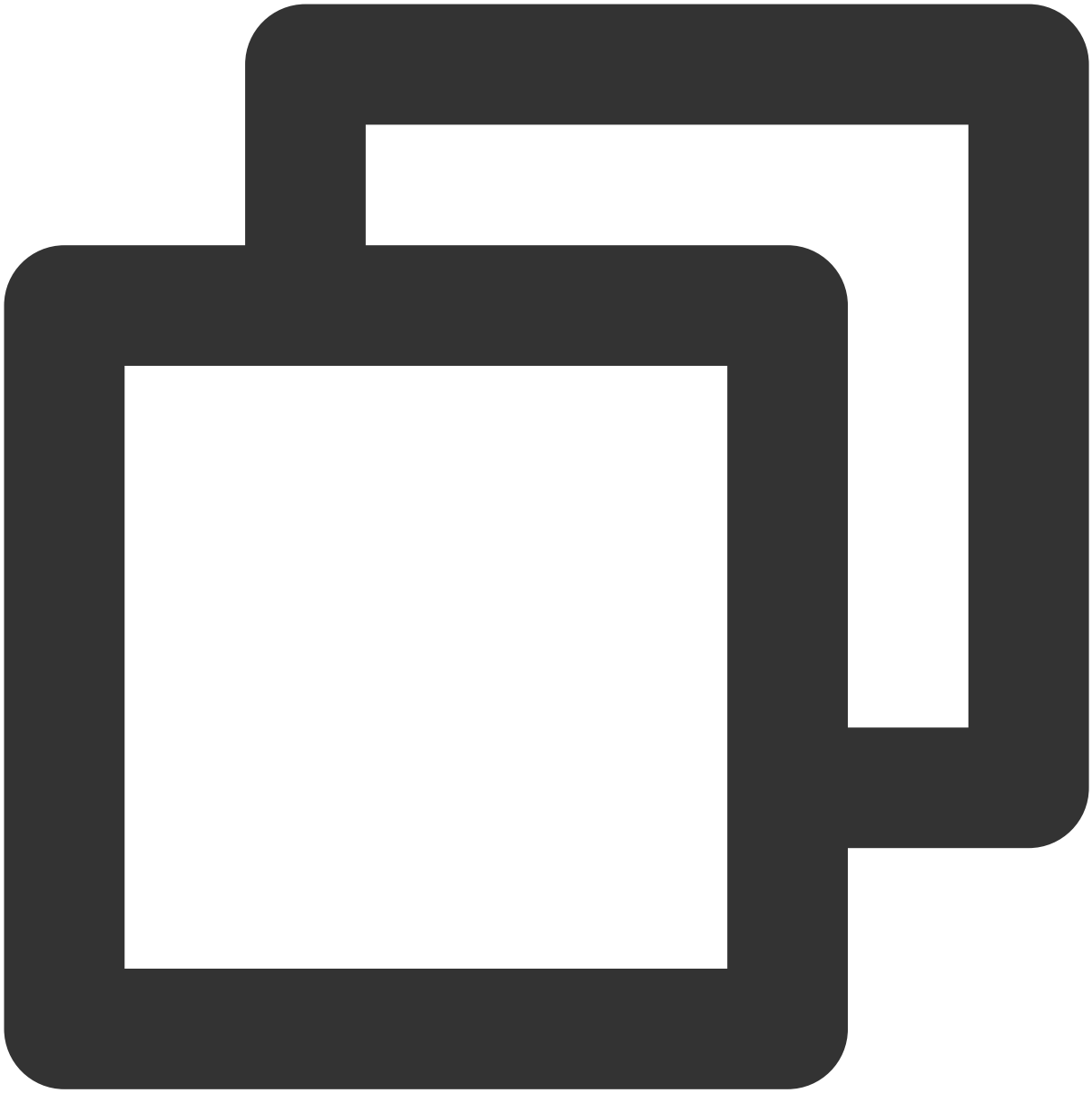
```
void onCallBegin(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType callMediaTy
```

パラ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	TUICommonDefine.RoomId	今回の通話のオーディオビデオルームID。現在は数字のルームナンバーのみサポートしています。文字列のルームナンバーは今後のバージョンでサポート予定です
callMediaType	TUICallDefine.MediaType	通話のメディアタイプ。ビデオ通話、音声通話
callRole	TUICallDefine.Role	ロール。列挙タイプ：発呼側、着呼側

onCallEnd

通話の終了を表します。発呼側と着呼側がどちらも受信できます。このイベントを監視することで、通話時間、通話タイプなどの情報を表示したり、クラウドのレコーディングフローを停止したりすることができます。



```
void onCallEnd(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType callMediaType
```

パラメータは下表に示すとおりです。

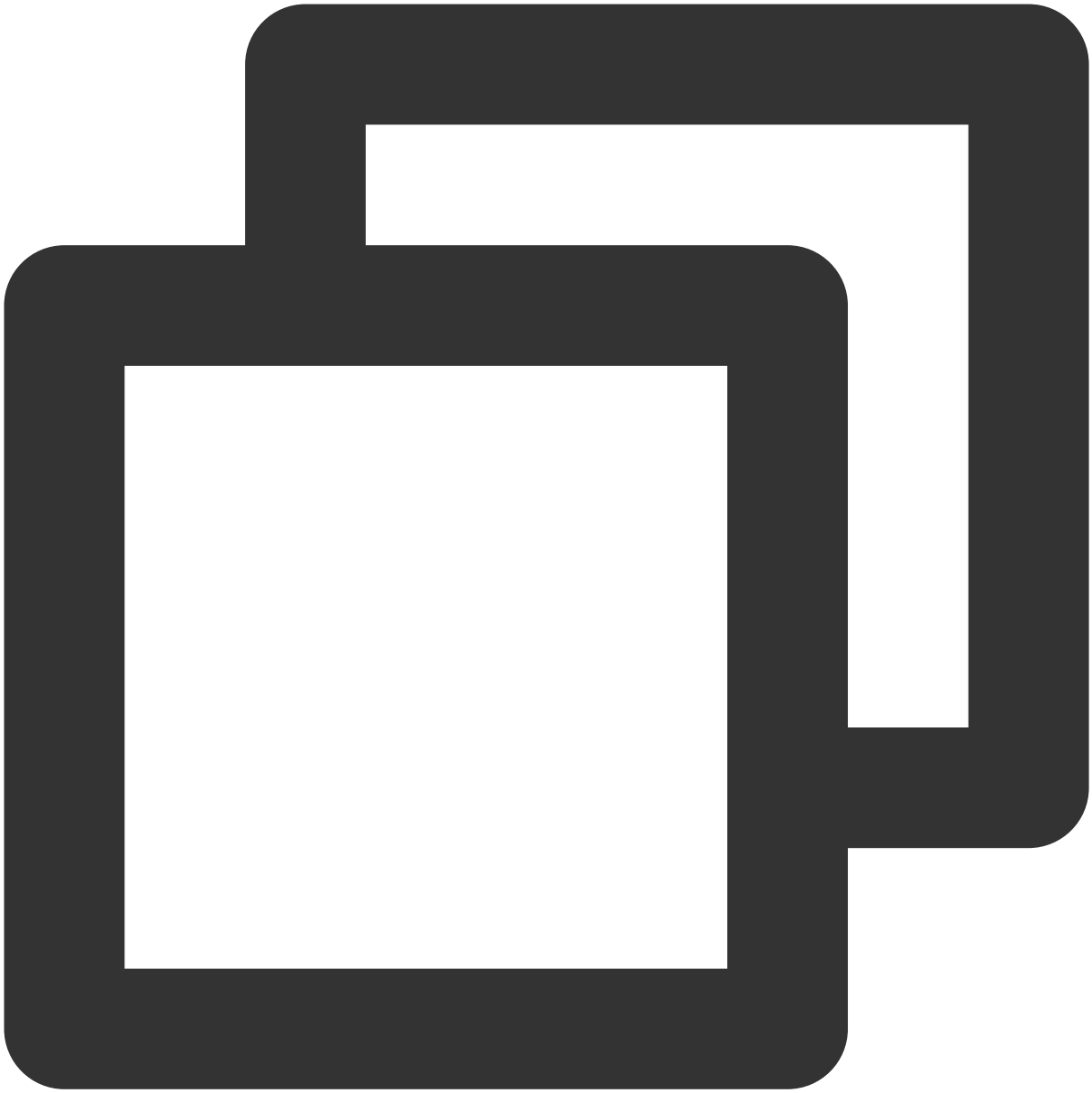
パラメータ	タイプ	意味
roomId	TUICommonDefine.RoomId	今回の通話のオーディオビデオルームID。現在は数字のルームナンバーのみサポートしています。文字列のルームナンバーは今後のバージョンでサポート予定です
callMediaType	TUICallDefine.MediaType	通話のメディアタイプ。ビデオ通話、音声通話

callRole	TUICallDefine.Role	ロール。列挙タイプ：発呼側、着呼側
totalTime	long	今回の通話時間

ご注意：
クライアントのイベントは一般的にすべて、プロセスキルなどの異常イベントに伴って消失します。通話時間の監視によって料金計算などのロジックを完成させたい場合は、REST APIを使用してこの種のフローを完成させることをお勧めします。

onCallMediaTypeChanged

通話のメディアタイプに変更が発生したことを表します。



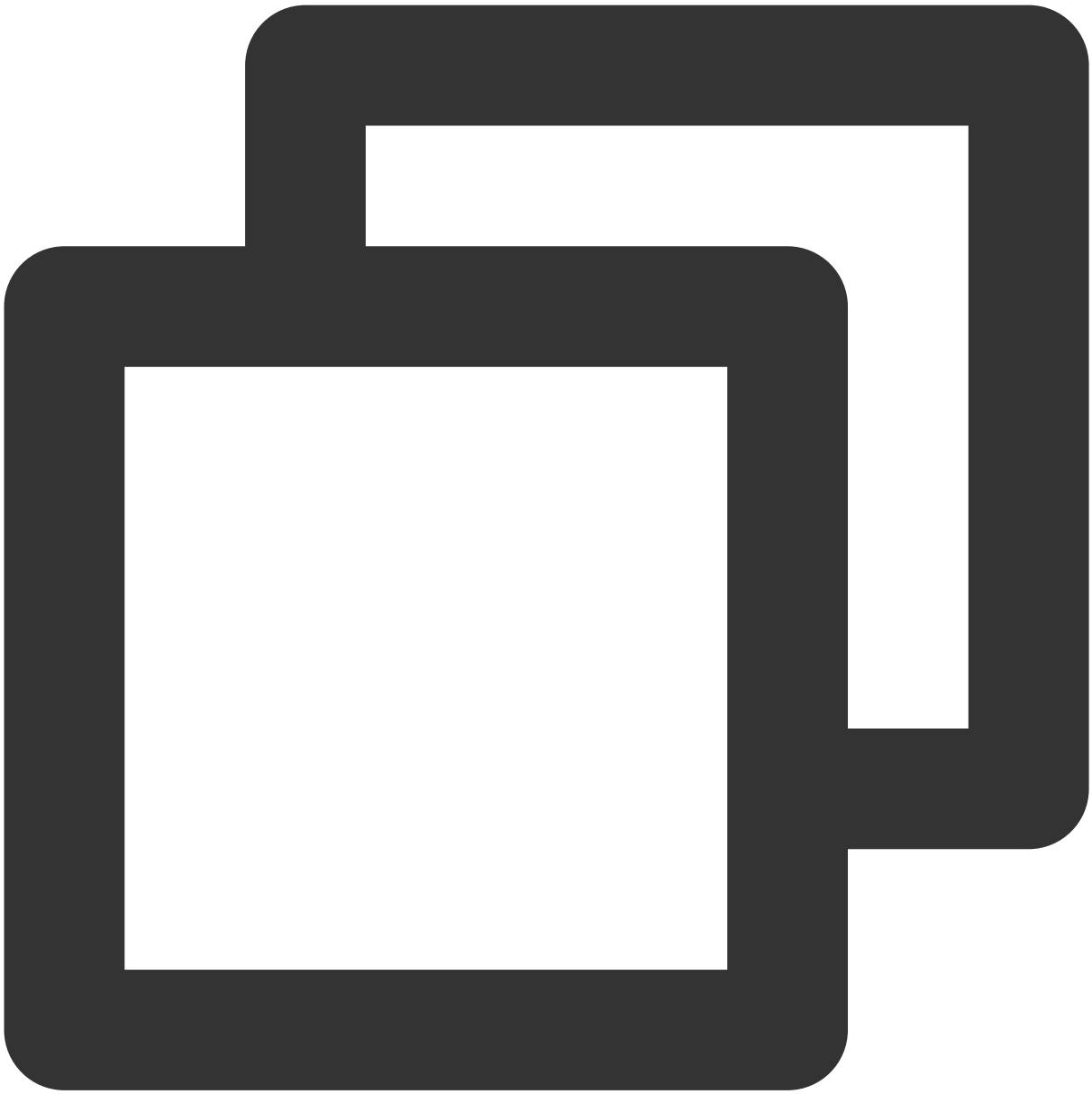
```
void onCallMediaTypeChanged(TUICallDefine.MediaType oldCallMediaType,TUICallDefine.
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
oldCallMediaType	TUICallDefine.MediaType	変更前の通話タイプ
newCallMediaType	TUICallDefine.MediaType	変更後の通話タイプ

onUserReject

通話が拒否された場合のコールバックです。1v1通話では、発呼側のみが拒否のコールバックを受信します。グループ通話では、すべての被招待者がこのコールバックを受信することができます。



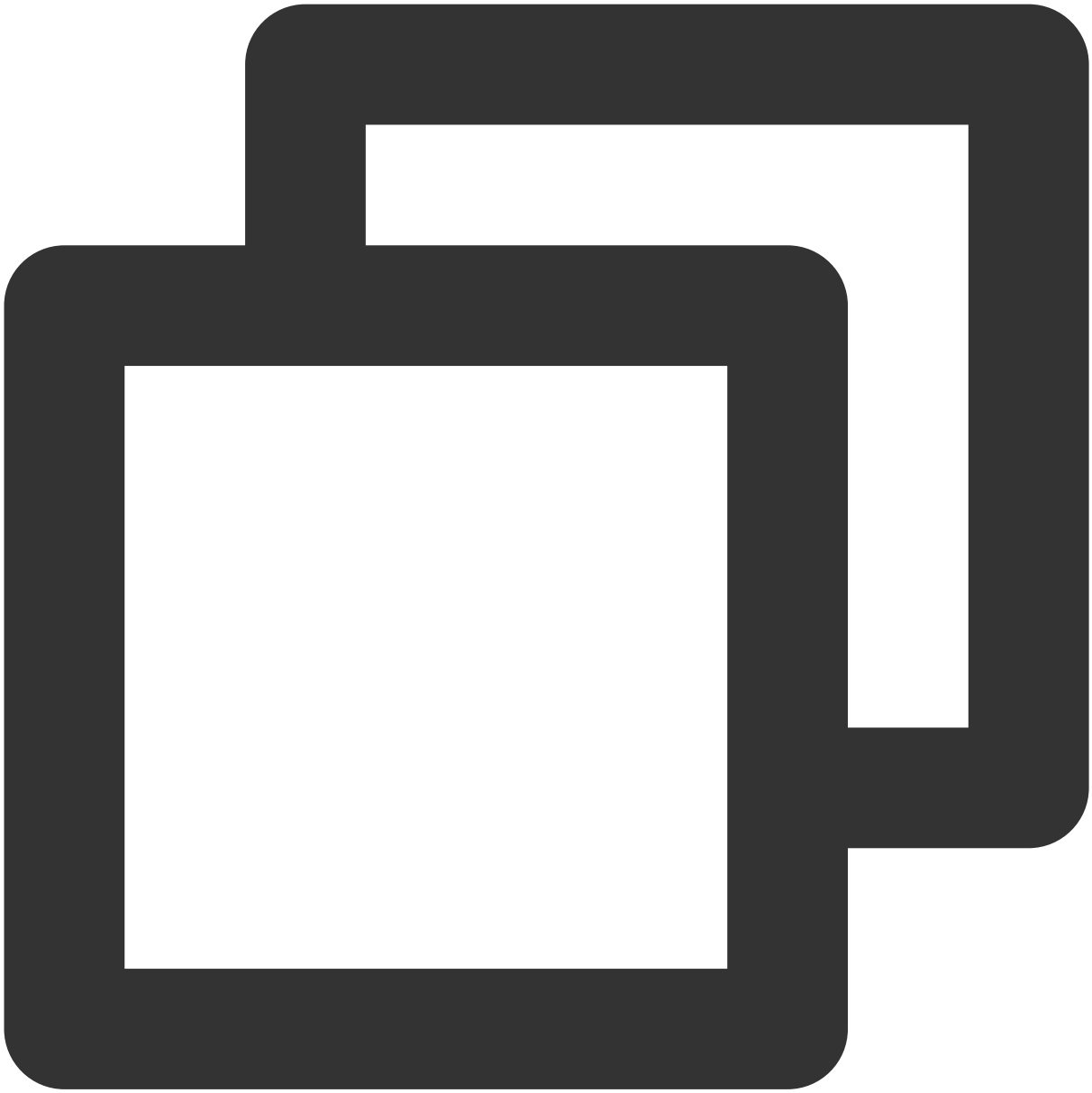
```
void onUserReject(String userId);
```

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	拒否したユーザーのID

onUserNoResponse

相手方が応答しない場合のコールバック。



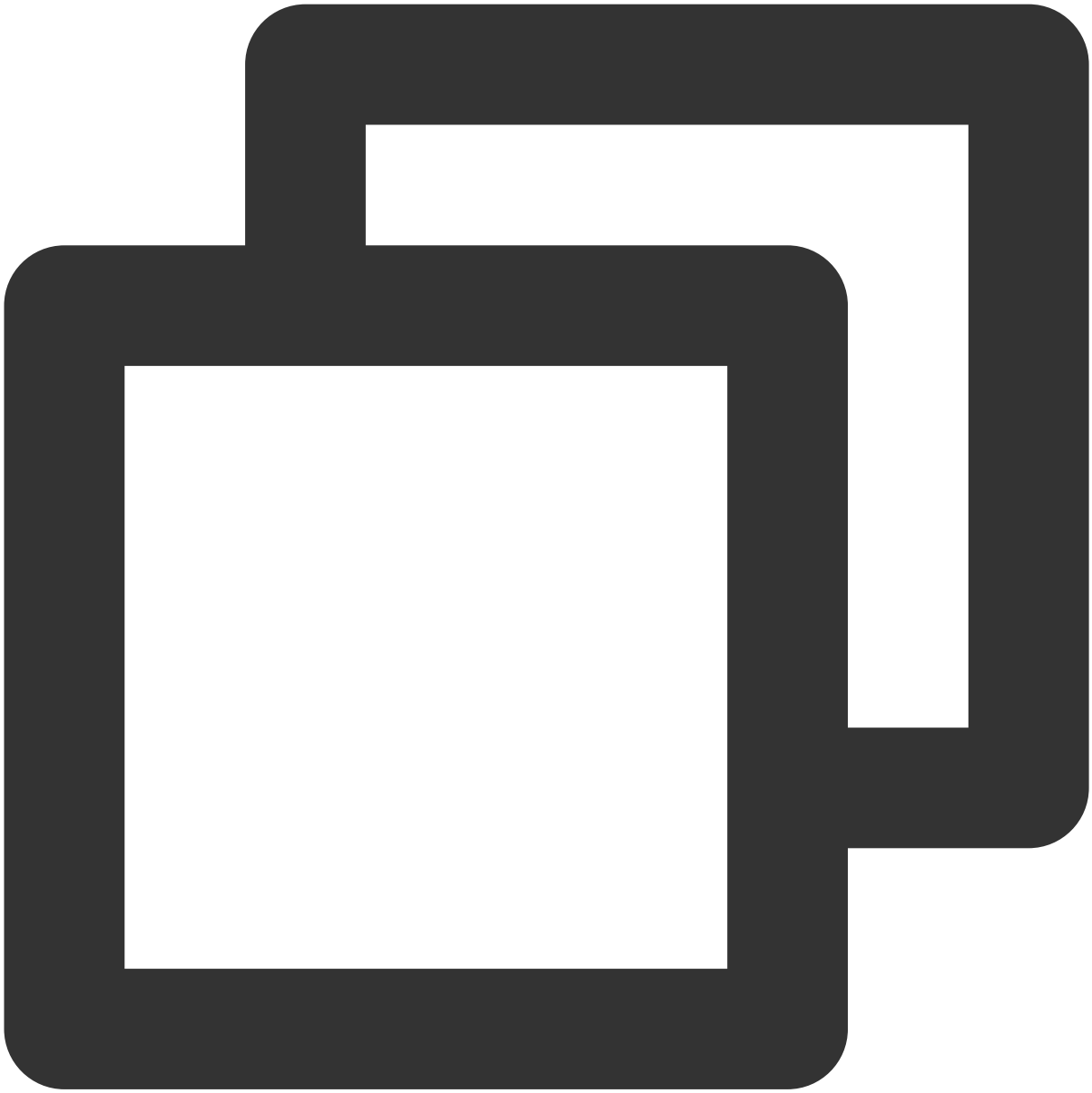
```
void onUserNoResponse(String userId);
```

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	応答しないユーザーのID

onUserLineBusy

通話中である場合のコールバック。



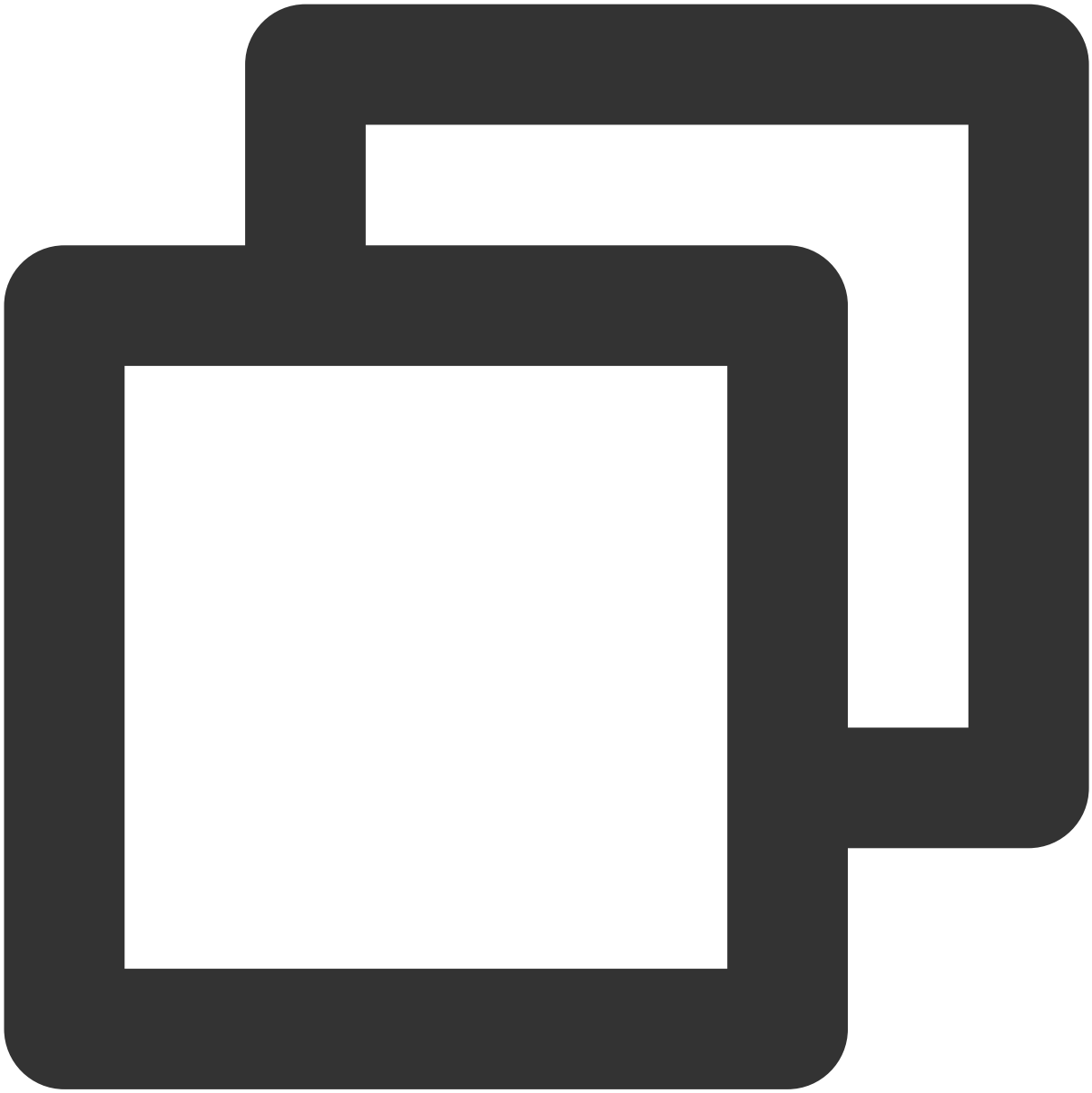
```
void onUserLineBusy(String userId);
```

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	通話中のユーザーのID

onUserJoin

今回の通話に参加したユーザーがいる場合のコールバック。



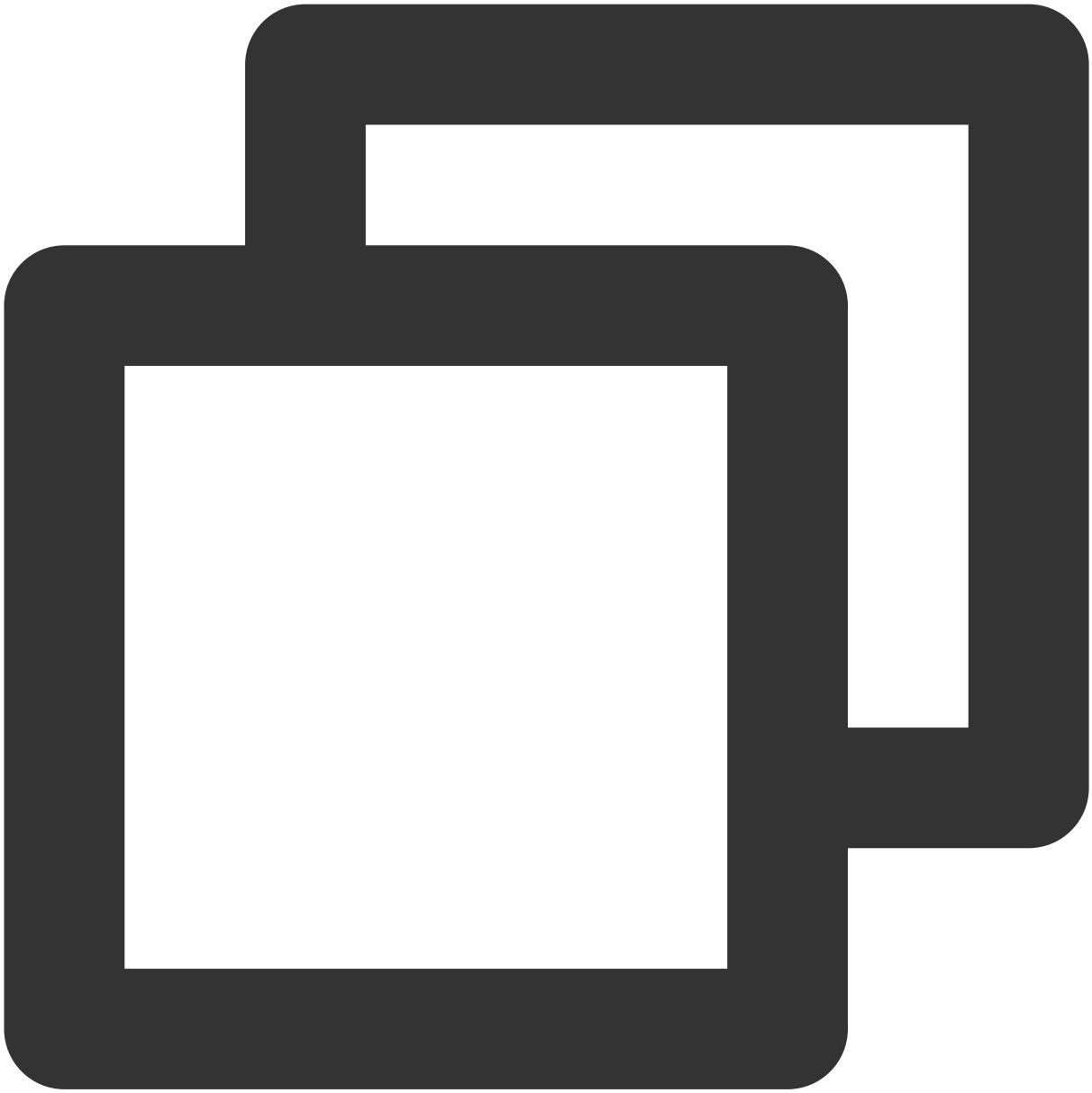
```
void onUserJoin(String userId);
```

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	現在の通話に参加したユーザーのID

onUserLeave

今回の通話から退出したユーザーがいる場合のコールバック。



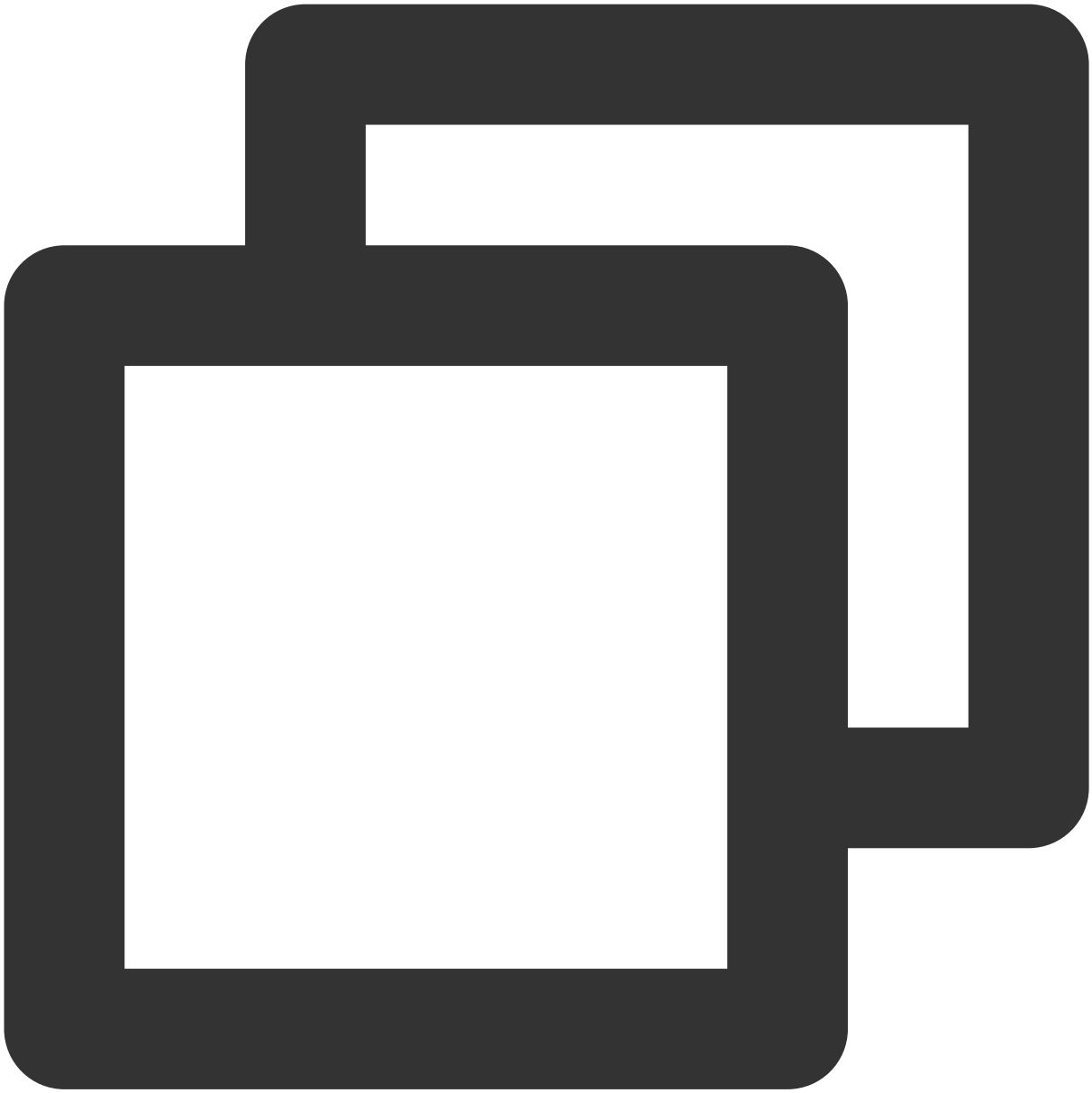
```
void onUserLeave(String userId);
```

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	現在の通話から退出したユーザーのID

onUserVideoAvailable

ユーザーがビデ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。



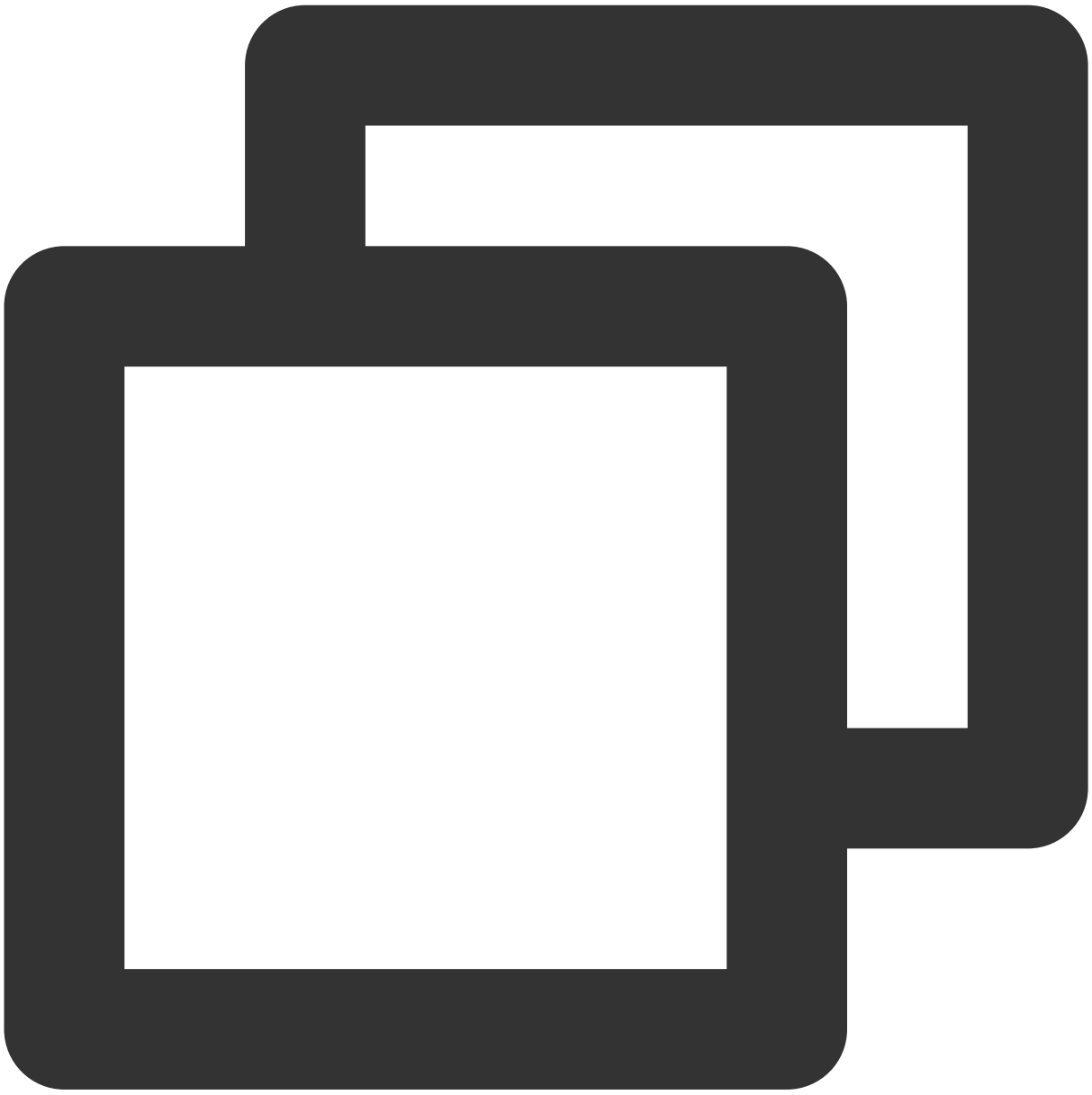
```
void onUserVideoAvailable(String userId, boolean isVideoAvailable);
```

パラ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	通話ユーザーID
isVideoAvailable	boolean	ユーザーのビデオが使用可能かどうか

onUserAudioAvailable

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 onUserAudioAvailable(String userId, boolean isAudioAvailable);
```

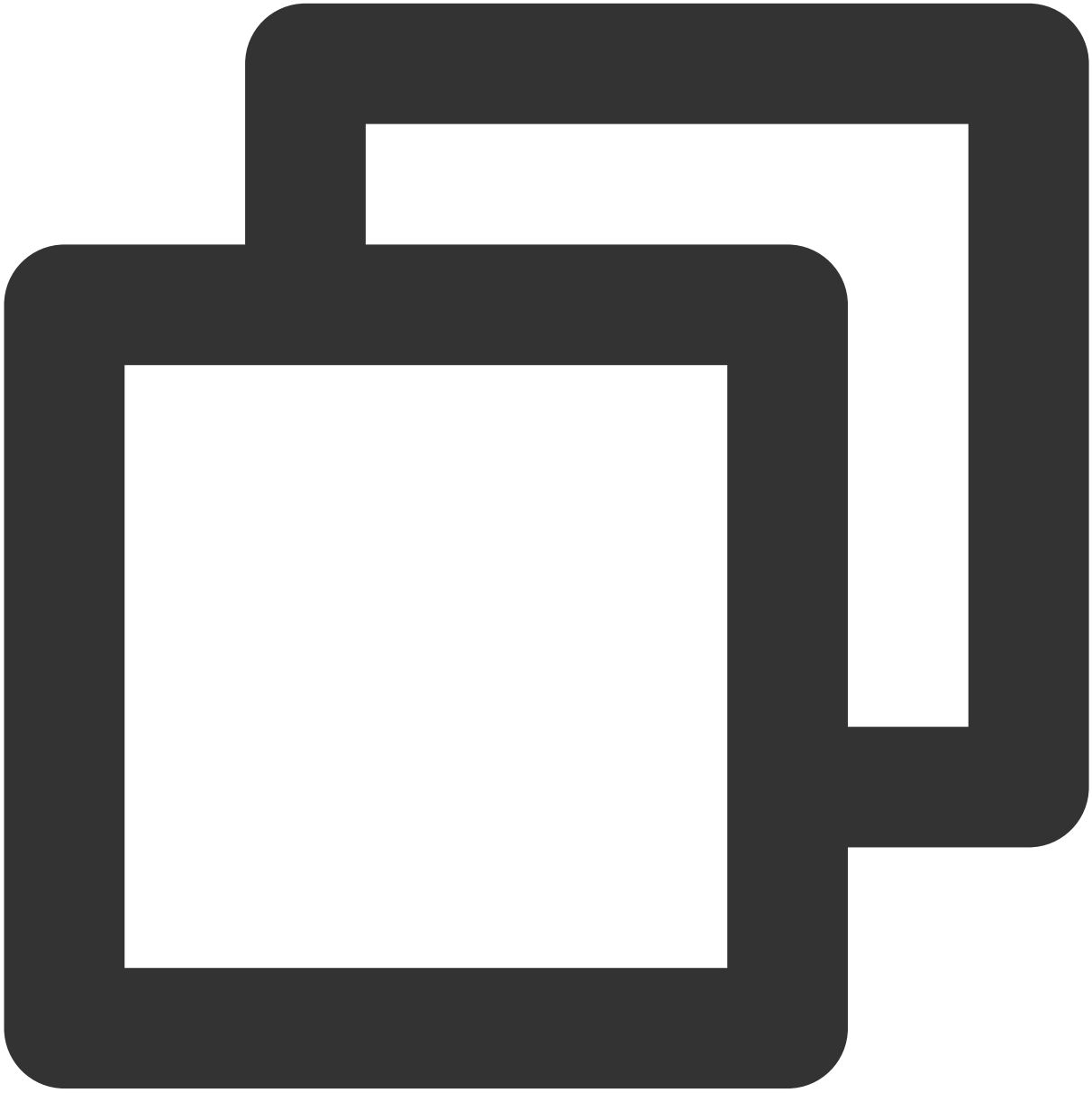
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
userId	String	ユーザーID

isAudioAvailable	boolean	ユーザーのオーディオが使用可能かどうか

onUserVoiceVolumeChanged

ユーザーの通話音量のコールバック。



```
void onUserVoiceVolumeChanged(Map<String, Integer> volumeMap);
```

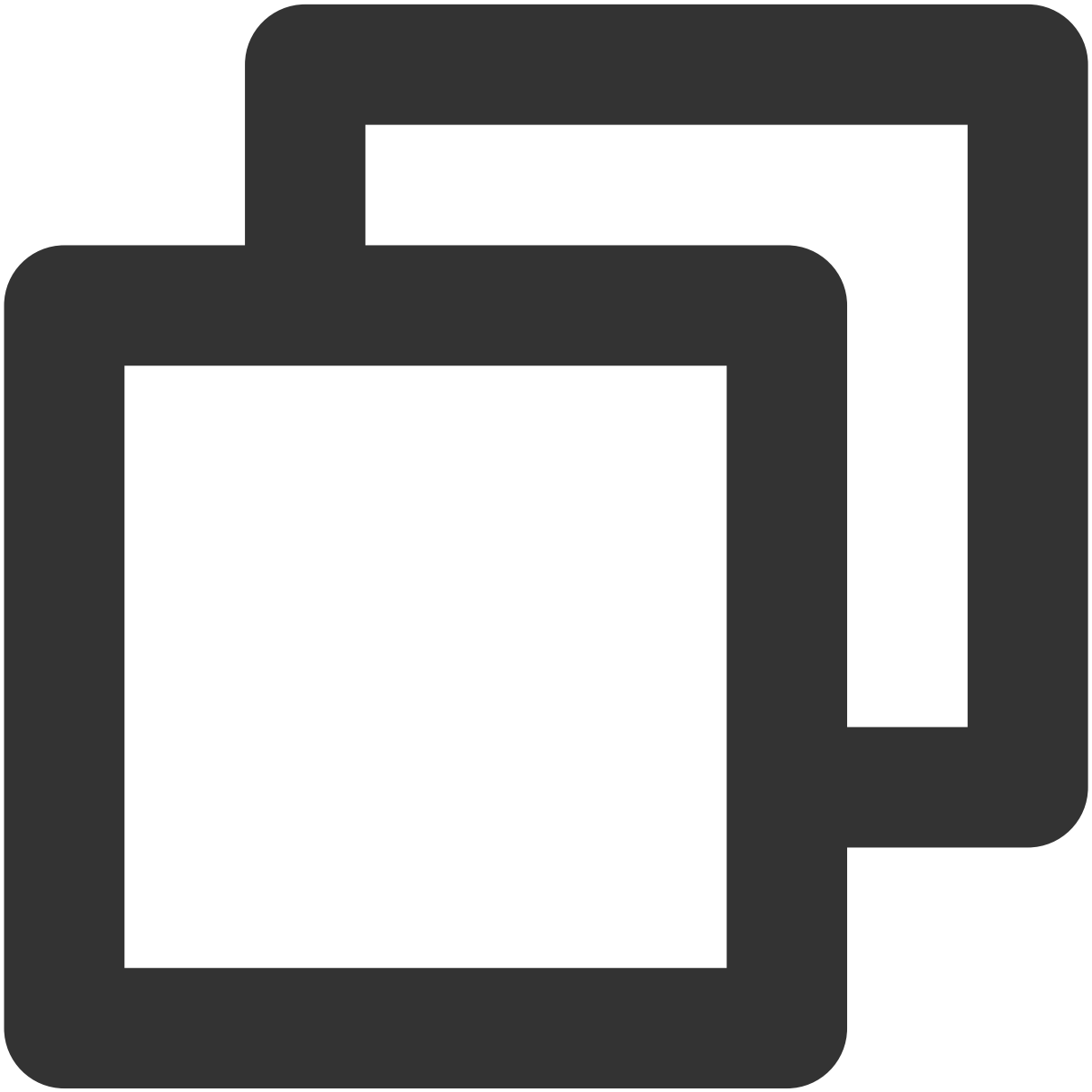
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

volumeMap	Map< String, Integer>	ボリュームメーター。各useridにつき、対応するユーザーの音量レベルを取得できます。音量の最小値は0、最大値は100です

onUserNetworkQualityChanged

ユーザーのネットワーク品質のコールバック。



```
void onUserNetworkQualityChanged(List<TUICallDefine.NetworkQualityInfo> networkQual
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
networkQualityList	List	ネットワーク状態。各userIdにつき、対応するユーザーの現在のネットワーク品質を取得できます

iOS
API概要

最終更新日：：2024-07-19 14:53:21

TUICallKit (UIインターフェースあり)

TUICallKit APIはオーディオビデオ通話コンポーネントの**UIインターフェース付き**のものです。TUICallKit APIを使用することで、WeChatのようなオーディオビデオ通話シーンをシンプルなインターフェースでスピーディーに実現できます。

API	説明
createInstance	TUICallKitインスタンスの作成（シングルトンモード）
setSelfInfo	ユーザーのプロフィール画像、ニックネームの設定
call	1v1通話の開始
groupCall	グループ通話の開始
joinInGroupCall	現在のグループ通話に自主的に参加
setCallingBell	カスタム着信音の設定
enableMuteMode	ミュートモードのオン/オフ
enableFloatWindow	フローティングウィンドウ機能のオン/オフ

TUICallEngine (UIインターフェースなし)

TUICallEngine APIはオーディオビデオ通話コンポーネントの**UIインターフェースがない**ものです。TUICallKitのインタラクションではニーズを満たせない場合はこのAPIを使用し、業務ニーズに応じてパッケージをカスタマイズすることができます。

API	説明
createInstance	TUICallEngineインスタンスの作成（シングルトンモード）
destroyInstance	TUICallEngineインスタンスの破棄（シングルトンモード）
init	オーディオビデオ通話基本機能の認証完了

addObserver	イベントコールバックの追加
removeObserver	コールバックインターフェースの削除
call	1v1通話の開始
groupCall	グループ通話の開始
accept	通話応答
reject	通話拒否
hangup	通話終了
ignore	通話を無視
inviteUser	グループ通話中に他の人を招待
joinInGroupCall	現在のグループ通話に自主的に参加
switchCallMediaType	通話メディアタイプの切り替え。ビデオ通話からオーディオ通話への切り替えなど
setRenderView	ビデオ画面に表示するViewオブジェクトの設定
startRemoteView	ビデオ画面に表示するViewオブジェクトの設定
stopRemoteView	ビデオ画面に表示するViewオブジェクトの設定
openCamera	カメラの起動
closeCamera	カメラの終了
switchCamera	フロント/リアカメラの切り替え
openMicrophone	マイクをオンにする
closeMicrophone	マイクをオフにする
selectAudioPlaybackDevice	オーディオ再生デバイスの選択（ヘッドホン/ハンズフリー）
setSelfInfo	ユーザーのプロフィール画像、ニックネームの設定
enableMultiDeviceAbility	TUICallEngineのマルチデバイスログインモードのオン/オフ （プレミアム版パッケージのみサポート）

TUICallObserver

TUICallObserverはTUICallEngineに対応するコールバックイベントクラスです。このコールバックによって、関心のあるコールバックイベントを監視することができます。

API	説明
onError	通話中のエラーコールバック
onCallReceived	通話リクエストのコールバック
onCallCancelled	通話キャンセルのコールバック
onCallBegin	通話接続のコールバック
onCallEnd	通話終了のコールバック
onCallMediaTypeChanged	通話メディアタイプ変更発生のコールバック
onUserReject	xxxxユーザーによる通話拒否のコールバック
onUserNoResponse	xxxxユーザーの応答なしのコールバック
onUserLineBusy	xxxxユーザーが通話中である場合のコールバック
onUserJoin	xxxxユーザーの通話参加のコールバック
onUserLeave	xxxxユーザーの通話からの退出のコールバック
onUserVideoAvailable	xxxユーザーのビデオストリームの有無のコールバック
onUserAudioAvailable	xxxユーザーのオーディオストリームの有無のコールバック
onUserVoiceVolumeChanged	全ユーザーの音量レベルフィードバックのコールバック
onUserNetworkQualityChanged	全ユーザーのネットワーク品質フィードバックのコールバック

主要なタイプの定義

API	説明
TUICallMediaType	通話のメディアタイプ。列挙タイプ：ビデオ通話、音声通話
TUICallRole	通話のロール。列挙タイプ：発呼側、着呼側
TUICallStatus	通話の状態。列挙タイプ：アイドル状態、応答待ち、応答中
TUIRoomId	オーディオビデオルームId。数字、文字列の2種類をサポートしています

TUICallCamera	カメラIdパラメータ。列挙タイプ：フロントカメラ、リアカメラ
TUIAudioPlaybackDevice	音声再生デバイス。列挙タイプ：スピーカー、ヘッドホン
TUINetworkQualityInfo	現在のネットワーク品質情報

TUICallKit

最終更新日：：2024-07-19 14:53:21

TUICallKit APIの概要

TUICallKit APIはオーディオビデオ通話コンポーネントの**UIインターフェース付き**のものです。TUICallKit APIを使用することで、WeChatのようなオーディオビデオ通話シーンをシンプルなインターフェースでスピーディーに実現できます。より詳細なアクセス手順については、TUICallKitクイックアクセスをご参照ください。

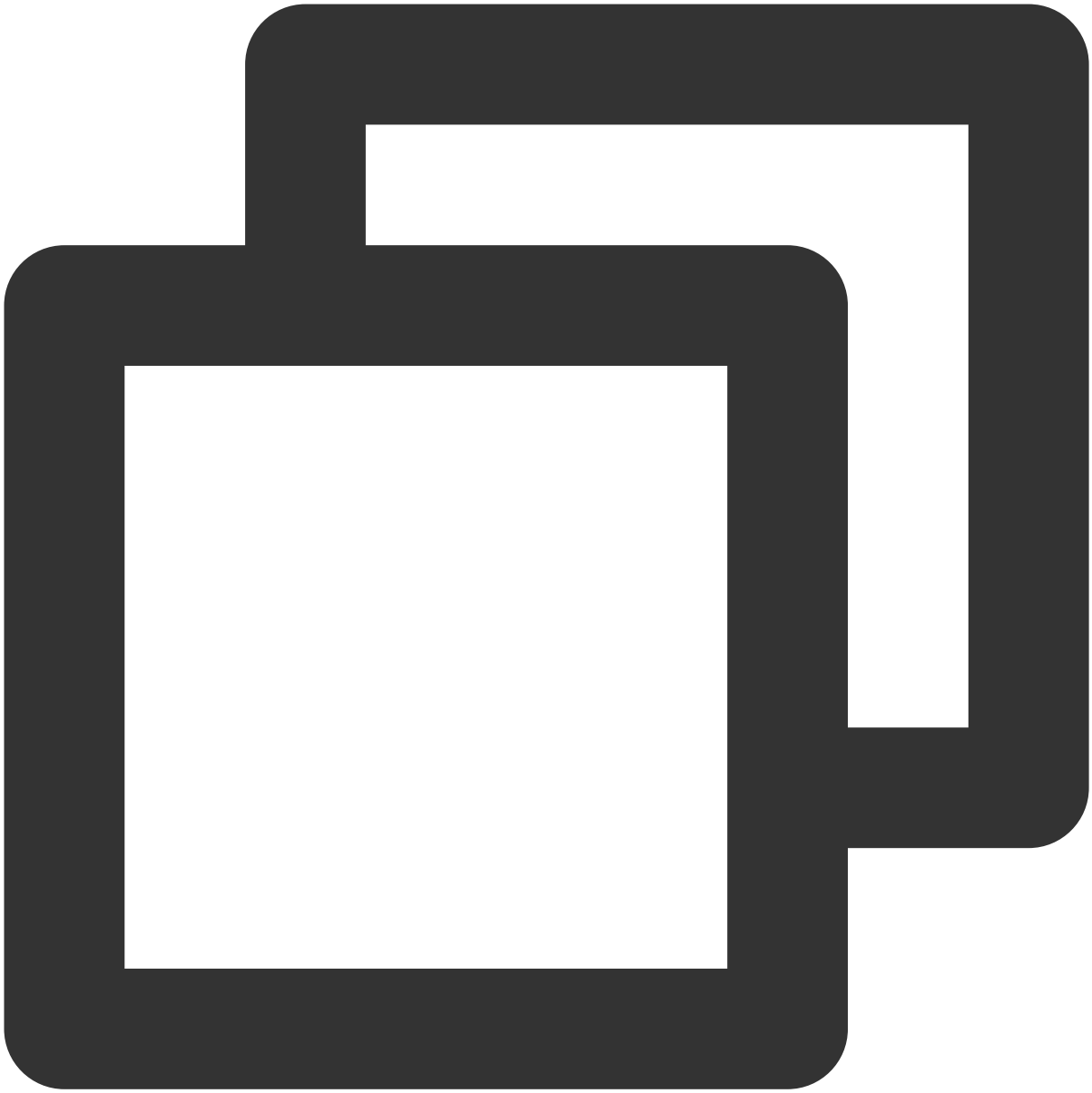
APIの概要

API	説明
createInstance	TUICallKitインスタンスの作成（シングルトンモード）
setSelfInfo	ユーザーのニックネーム、プロフィール画像の設定
call	1v1通話の開始
groupCall	グループ通話の開始
joinInGroupCall	現在のグループ通話に自主的に参加
setCallingBell	カスタム着信音の設定
enableMuteMode	ミュートモードのオン/オフ
enableFloatWindow	フローティングウィンドウ機能のオン/オフ

APIの詳細

createInstance

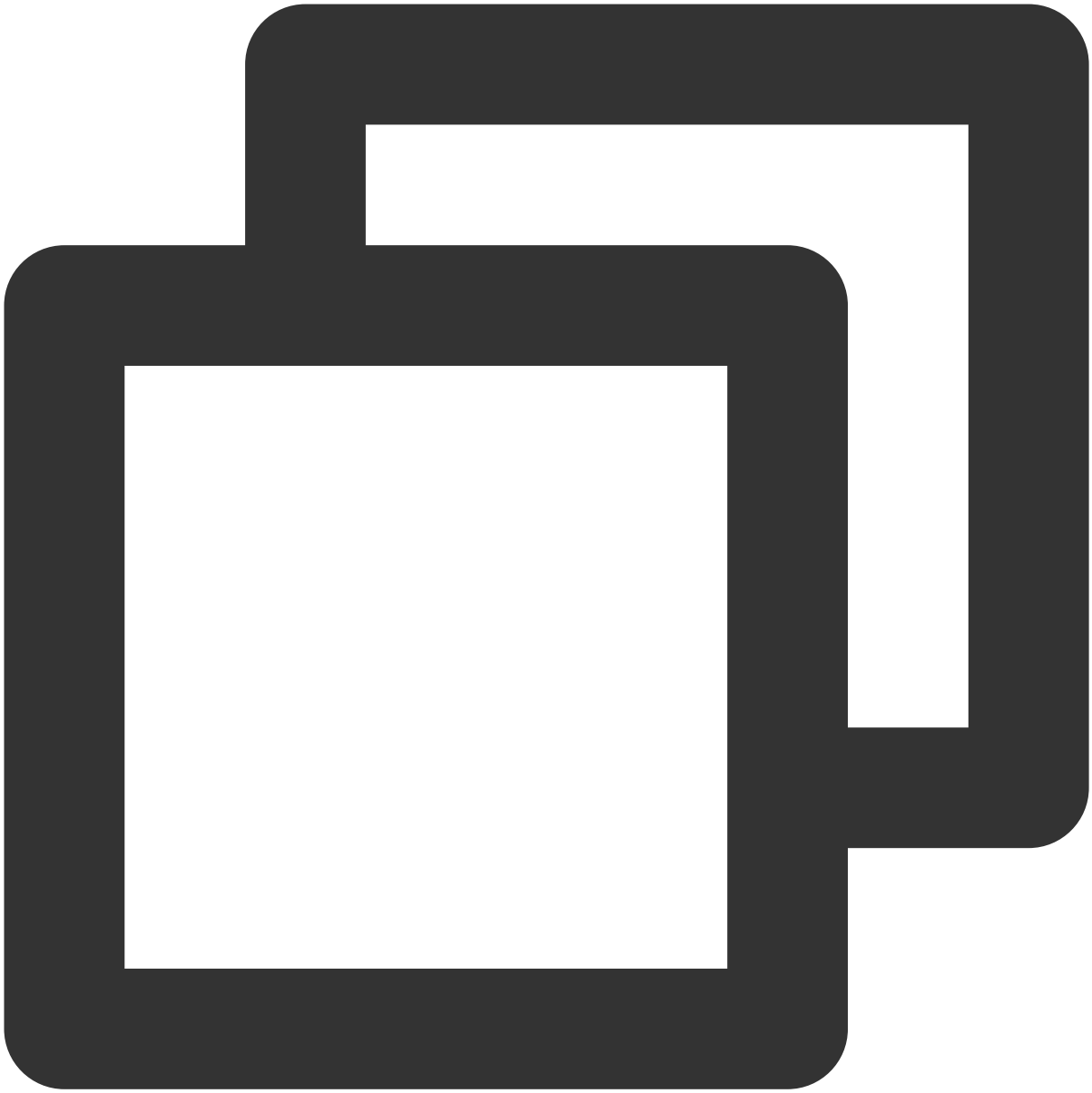
TUICallKitのシングルトンを作成します。



```
- (instancetype)createInstance;
```

setSelfInfo

ユーザーニックネーム、プロフィール画像を設定します。ユーザーニックネームは500バイト以内、ユーザープロフィール画像はURL形式でなければなりません。

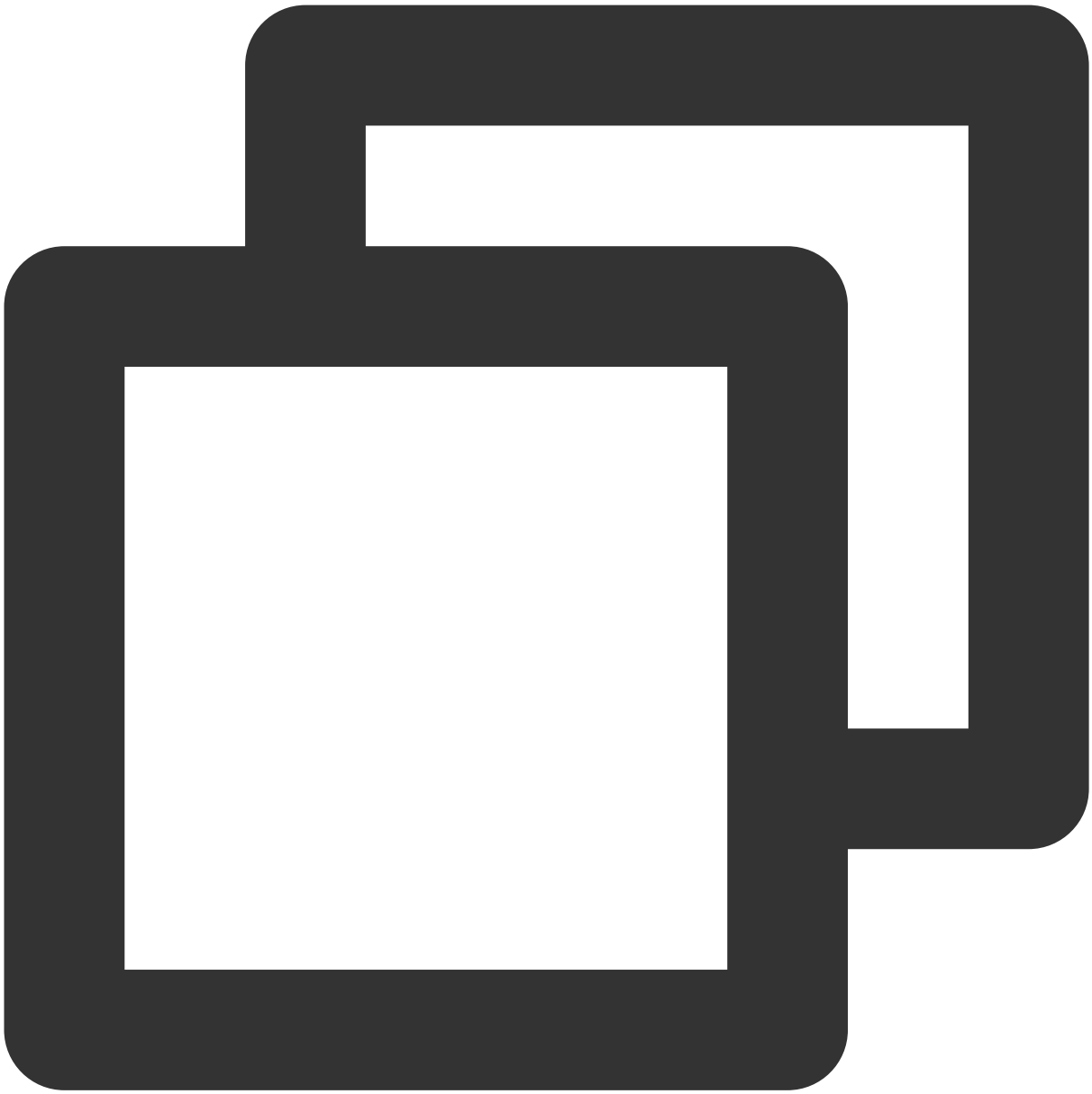


```
- (void)setSelfInfo:(NSString * _Nullable)nickname avatar:(NSString * _Nullable)ava
```

パラメータ	タイプ	意味
nickName	NSString	ターゲットユーザーのニックネーム
avatar	NSString	ターゲットユーザーのプロフィール画像

call

電話をかけます（1v1通話）



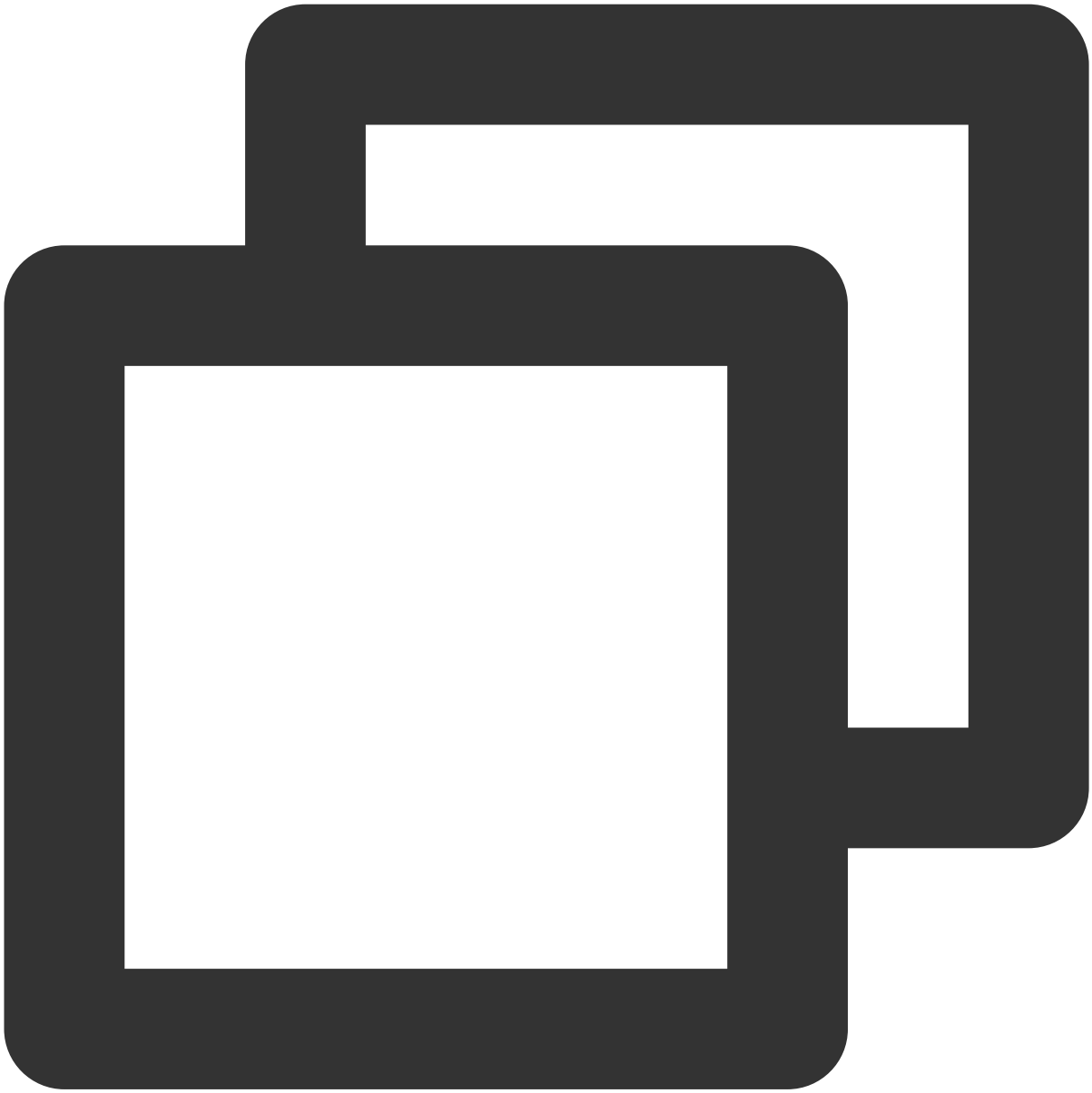
```
- (void)call:(NSString *)userId callMediaType:(TUICallMediaType)callMediaType;
```

パラ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	ターゲットユーザーのuserId
callMediaType	TUICallMediaType	通話のメディアタイプ。ビデオ通話、音声通話など

groupCall

グループ通話を開始します。注意：グループ通話を使用する前にIMグループを作成する必要があります。作成済みの場合は無視してください。



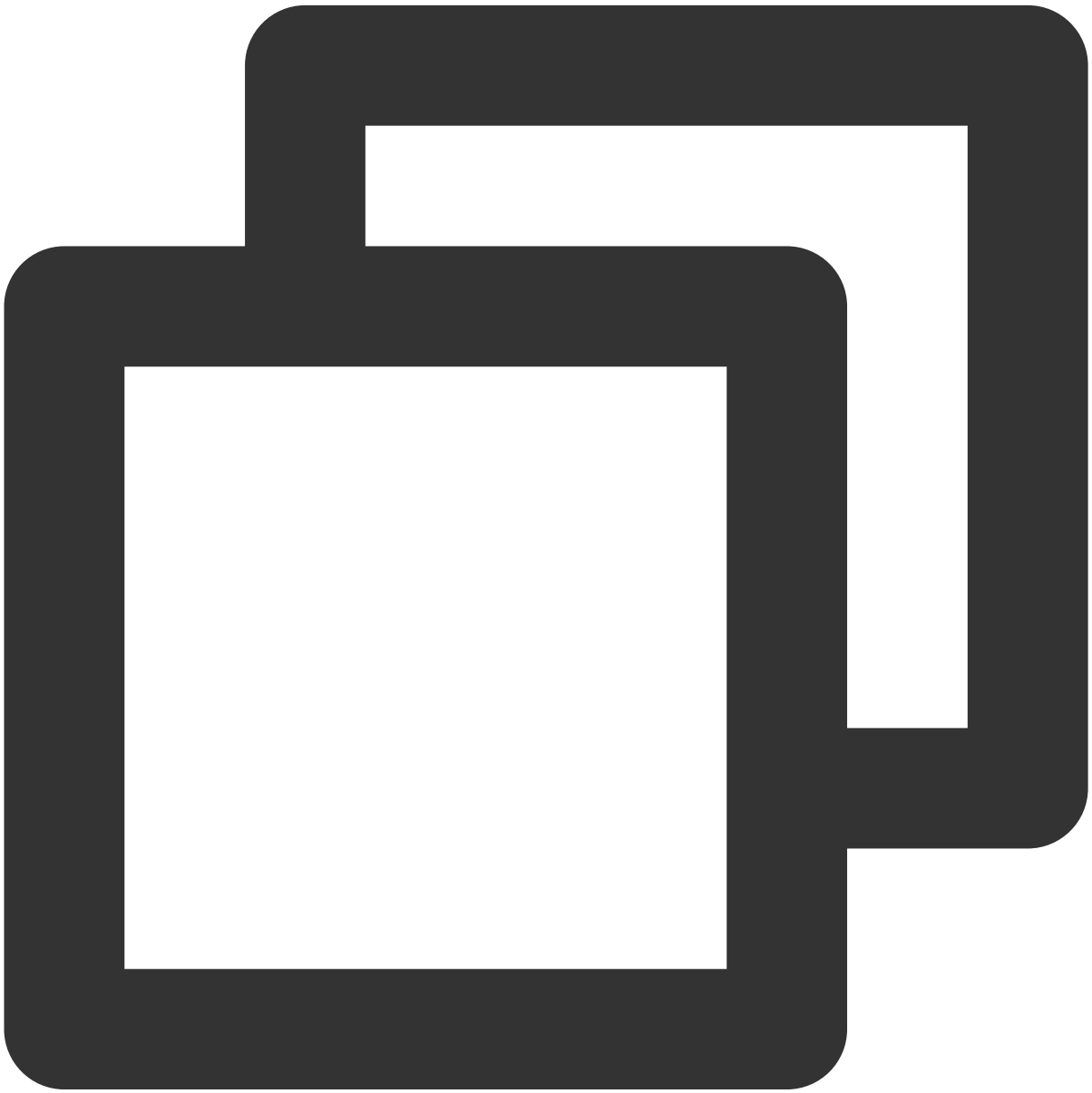
```
- (void)groupCall:(NSString *)groupId userIdList:(NSArray<NSString *> *)userIdList
```

パラメータ	タイプ	意味
groupId	NSString	今回のグループ通話のグループID
userIdList	NSArray	ターゲットユーザーのuserIdリスト

callMediaType	TUICallMediaType	通話のメディアタイプ。ビデオ通話、音声通話など

joinInGroupCall

グループ通話を開始します。注意：グループ通話を使用する前にIMグループを作成する必要があります。作成済みの場合は無視してください。



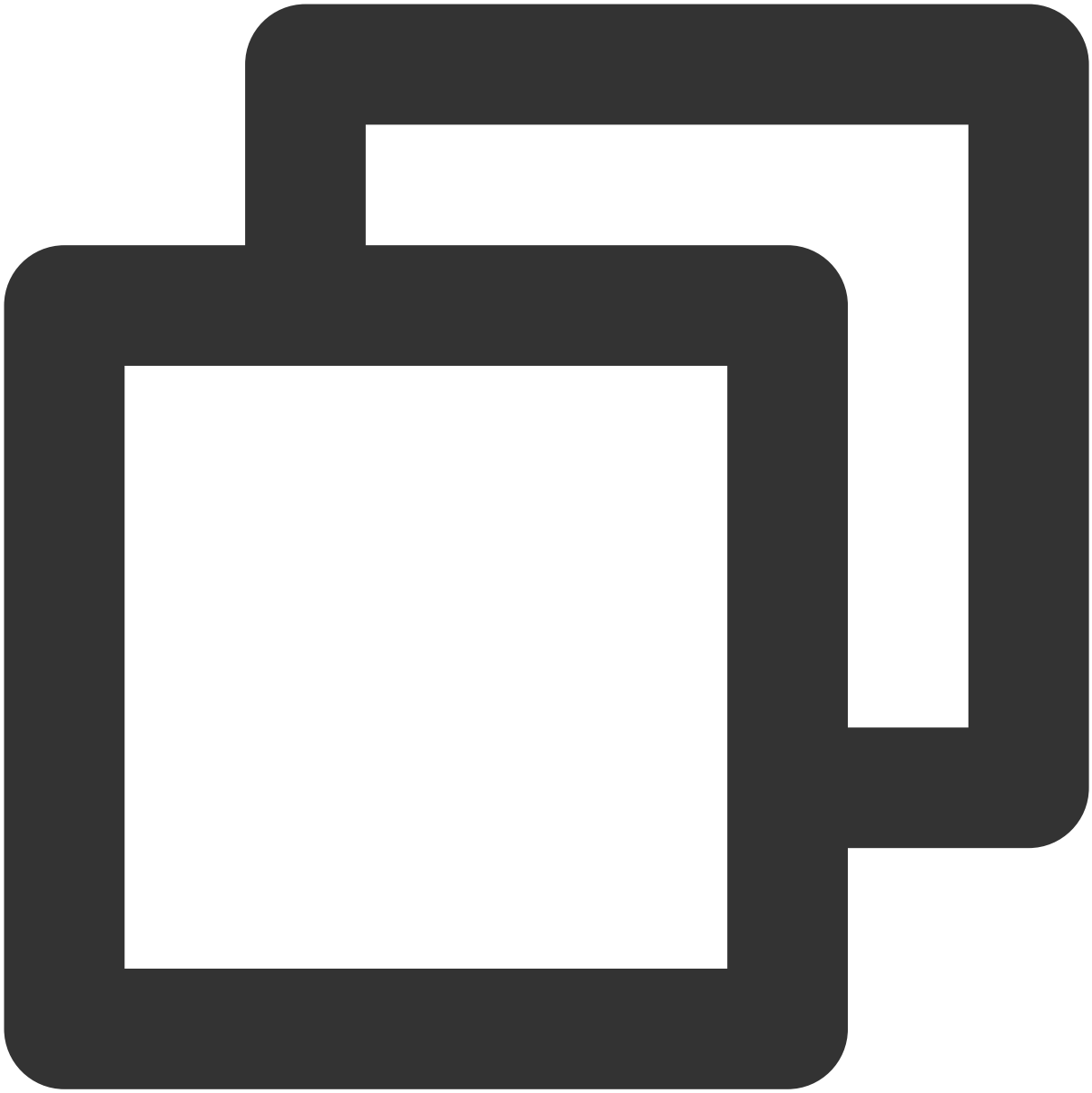
```
- (void)joinInGroupCall:(TUIRoomId *)roomId groupId:(NSString *)groupId callMediaTy
```

パラメータ	タイプ	意味

roomId	TUIRoomId	今回の通話のオーディオビデオルームID。現在は数字のルームナンバーのみサポートしています。文字列のルームナンバーは今後のバージョンでサポート予定です
groupId	NSString	今回のグループ通話のグループID
callMediaType	TUICallMediaType	通話のメディアタイプ。ビデオ通話、音声通話など

setCallingBell

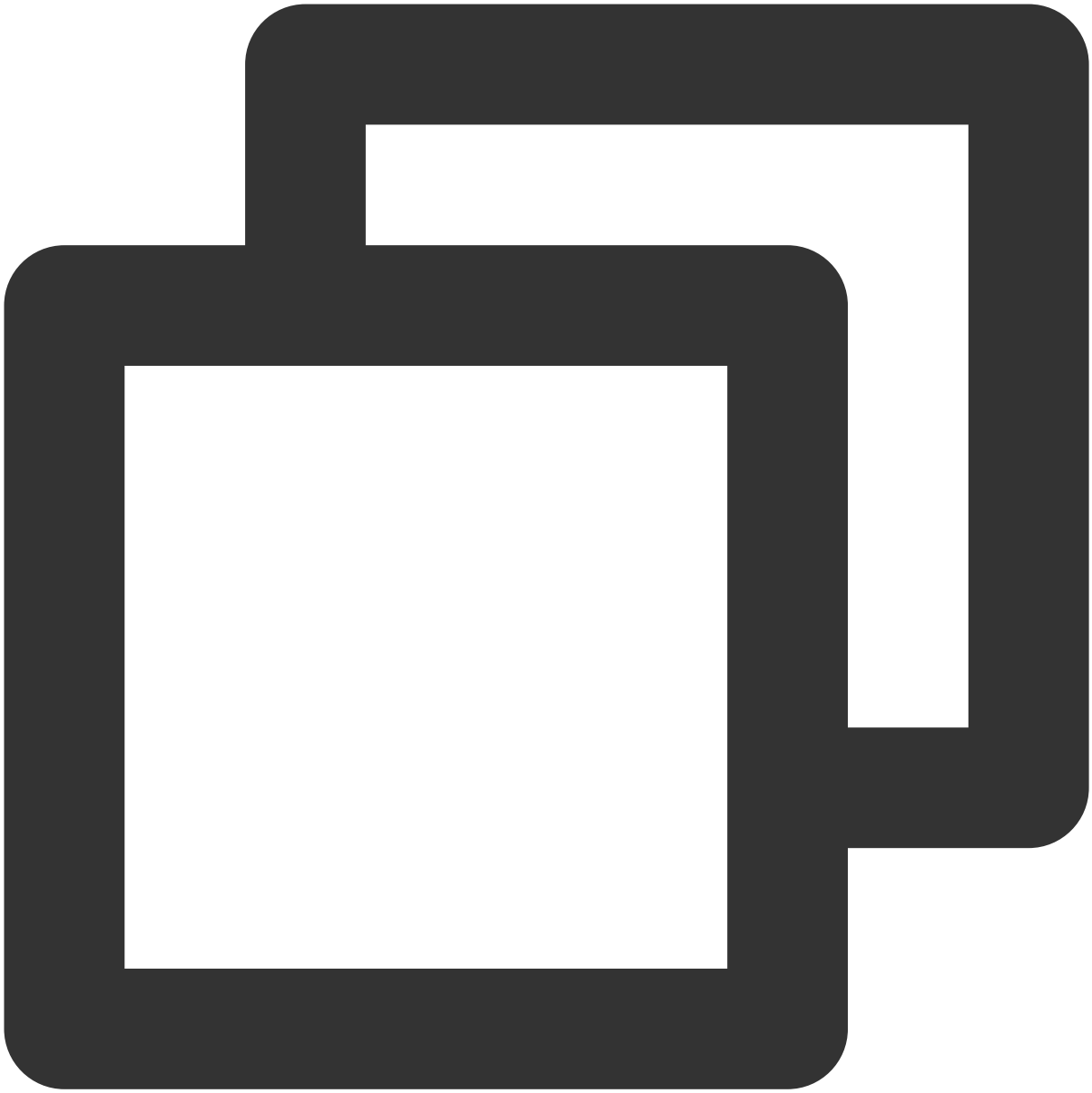
カスタム着信音を設定します。 ここではローカルファイルアドレスのみ渡すことができます。このファイルディレクトリにアプリケーションがアクセスできることを確認する必要があります。
着信音を設定後、デバイスにバインドします。ユーザーを変更しても着信音はそのまま有効です。
デフォルトの着信音に戻したい場合は、 `filePath` にnullを渡します。



```
- (void)setCallingBell:(NSString *)filePath;
```

enableMuteMode

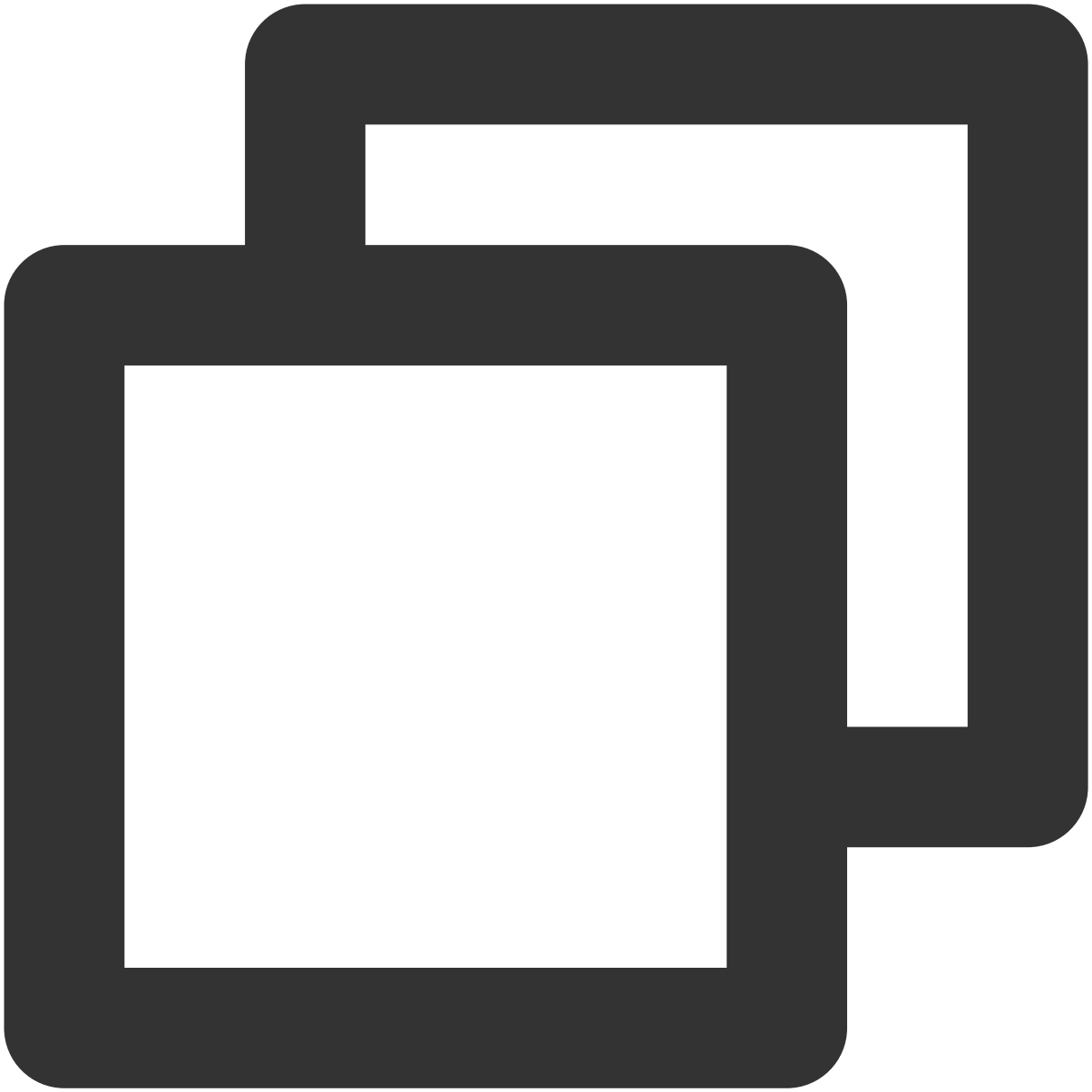
ミュートモードをオン/オフします。



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

enableFloatWindow

フローティングウィンドウ機能をオン/オフします。デフォルトでは `false` で、通話画面の左上隅のフローティングウィンドウボタンは非表示になっており、trueに設定すると表示されます。



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

TUICallEngine

最終更新日：：2024-07-19 14:53:21

TUICallEngine APIの概要

TUICallEngine APIはオーディオビデオ通話コンポーネントの**UIインターフェースがない**ものです。TUICallKitのインタラクションではニーズを満たせない場合はこのインターフェースを使用し、パッケージのインタラクションをカスタマイズすることができます。

APIの概要

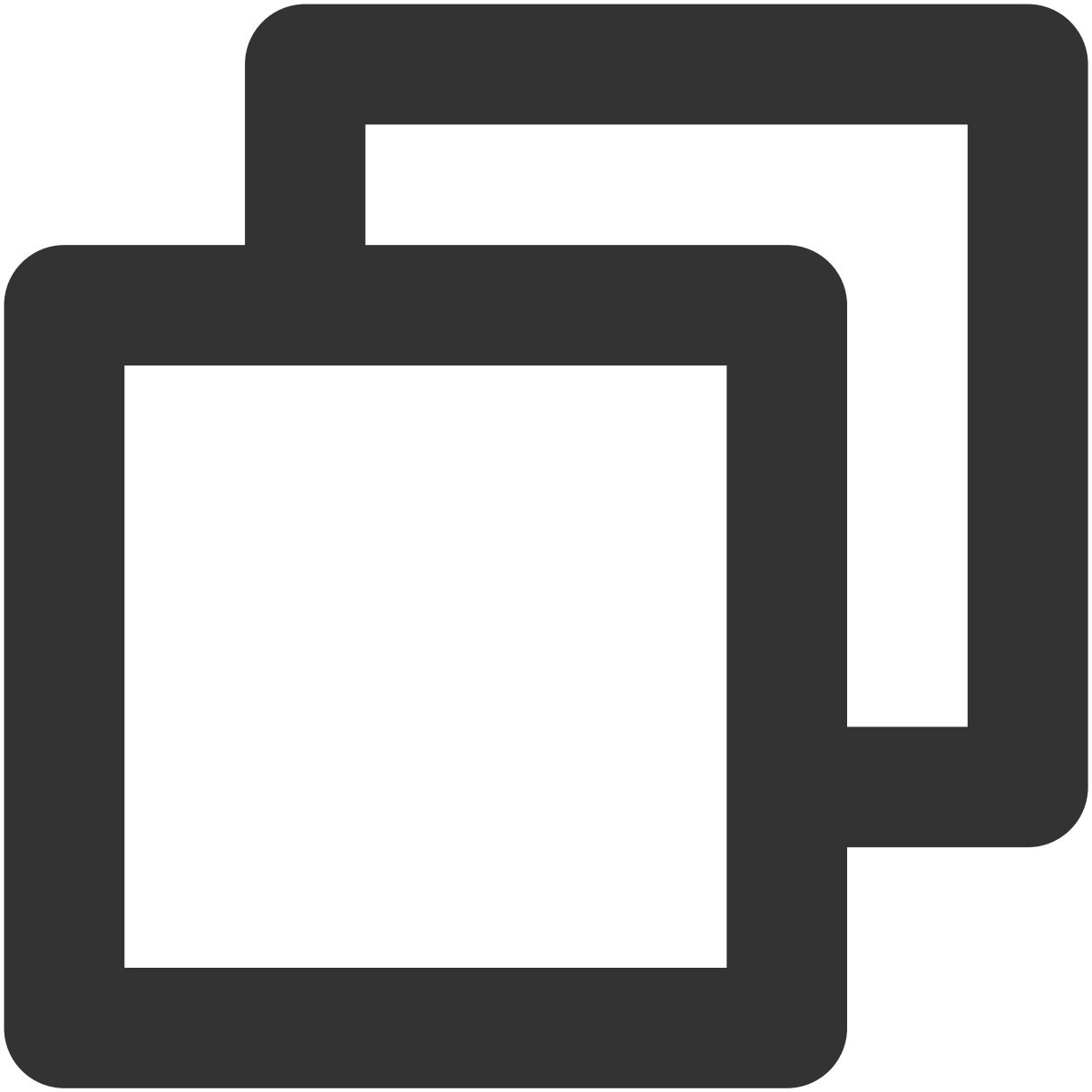
API	説明
createInstance	TUICallEngineインスタンスの作成（シングルトンモード）
destroyInstance	TUICallEngineインスタンスの破棄（シングルトンモード）
init	オーディオビデオ通話基本機能の認証完了
addObserver	イベントコールバックの追加
removeObserver	コールバックインターフェースの削除
call	1v1通話の開始
groupCall	グループ通話の開始
accept	通話応答
reject	通話拒否
hangup	通話終了
ignore	通話を無視
inviteUser	グループ通話中に他の人を招待
joinInGroupCall	現在のグループ通話に自主的に参加
switchCallMediaType	通話メディアタイプの切り替え。ビデオ通話からオーディオ通話への切り替えなど
startRemoteView	リモートユーザービデオストリームのサブスクリプション開始

stopRemoteView	リモートユーザービデオストリームのサブスクリプション停止
openCamera	カメラの起動
closeCamera	カメラの終了
switchCamera	フロント/リアカメラの切り替え
openMicrophone	マイクをオンにする
closeMicrophone	マイクをオフにする
selectAudioPlaybackDevice	オーディオ再生デバイスの選択（ヘッドホン/スピーカー）
setSelfInfo	ユーザーのニックネーム、プロフィール画像の設定
enableMultiDeviceAbility	TUICallEngineのマルチデバイスログインモードのオン/オフ （プレミアム版パッケージのみサポート）

APIの詳細

createInstance

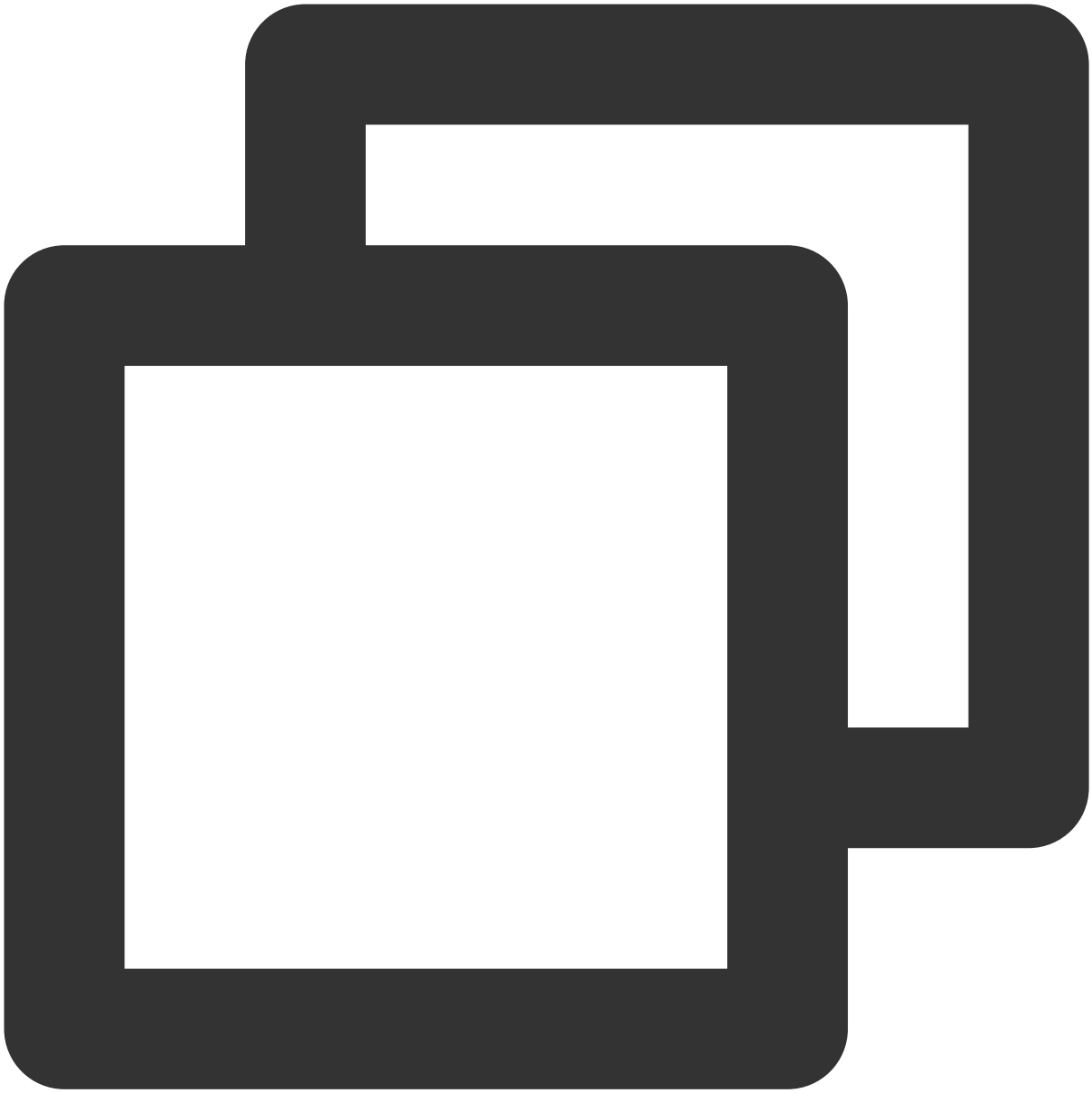
TUICallEngineのシングルトンを作成します。



```
- (TUICallEngine *)createInstance;
```

destroyInstance

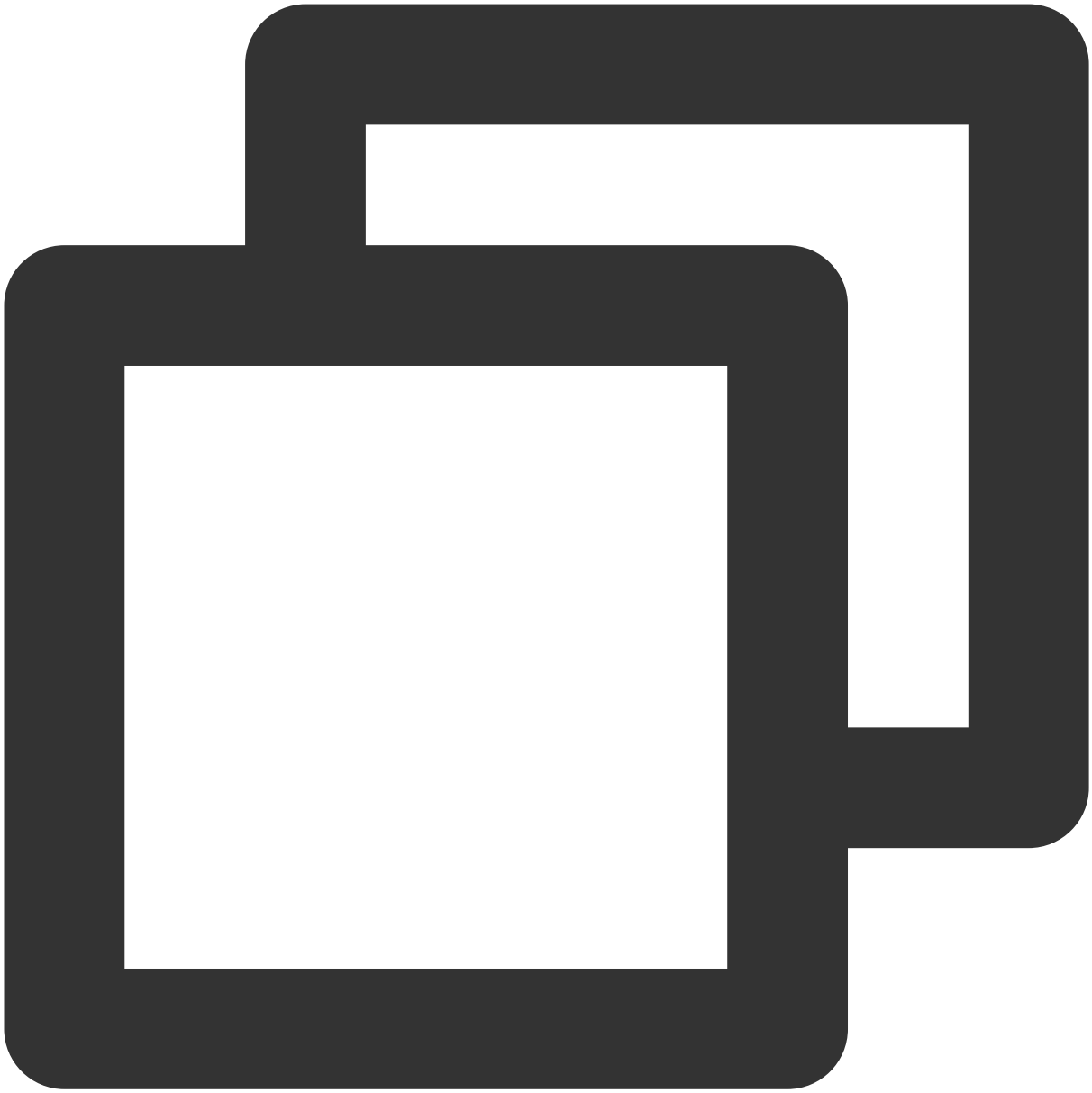
TUICallEngineのシングルトンを破棄します。



```
- (void)destroyInstance;
```

init

関数を初期化します。すべての機能を使用する前にこの関数を呼び出し、通話サービス認証を含む初期化操作を完了させてください。



```
- (void)init:(NSString *)sdkAppID userId:(NSString *)userId userSig:(NSString *)use
```

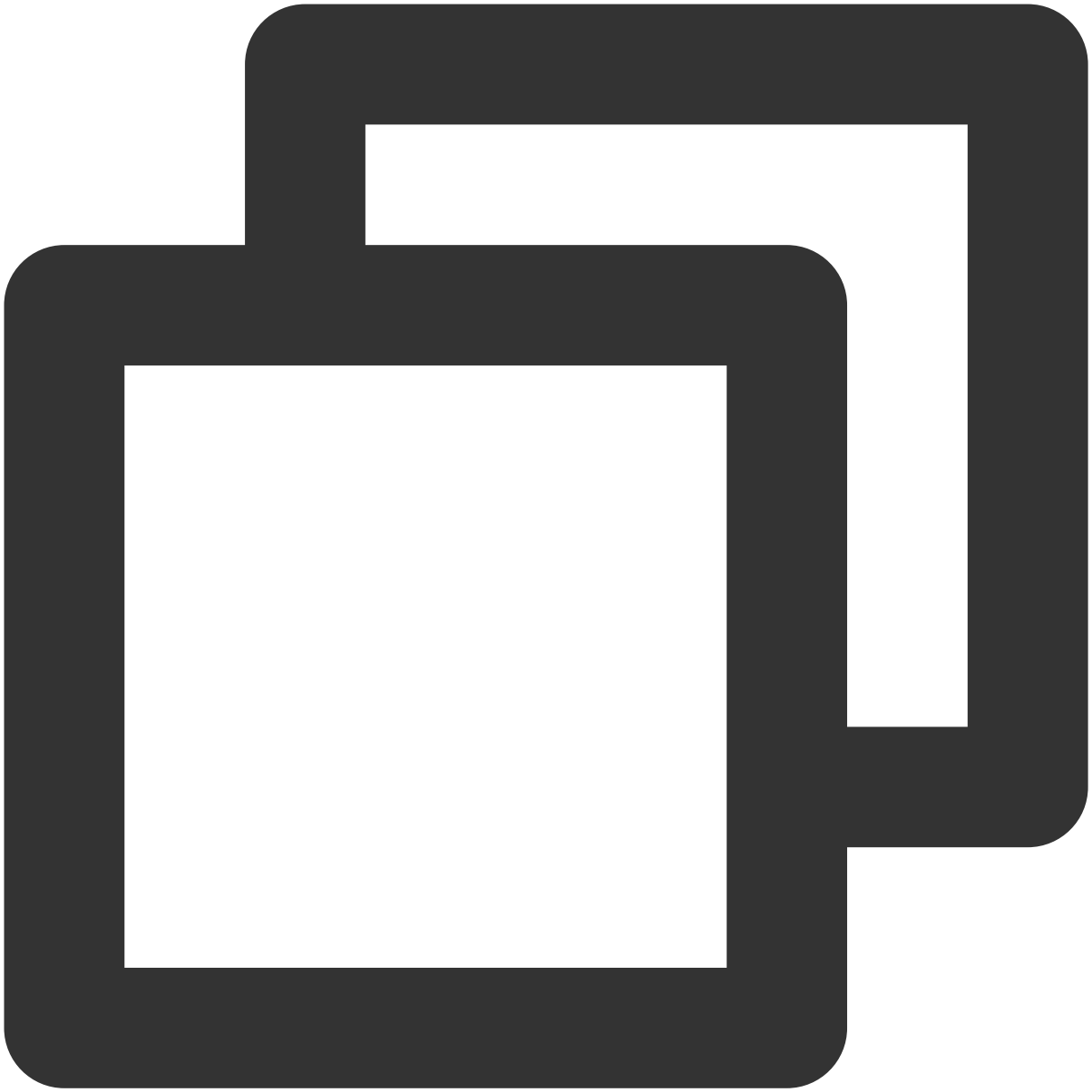
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
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。文字列タイプでは、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
callback	TUIDefine.Callback	初期化コールバック。 `onSuccess` は初期化に成功したことを表します

addObserver

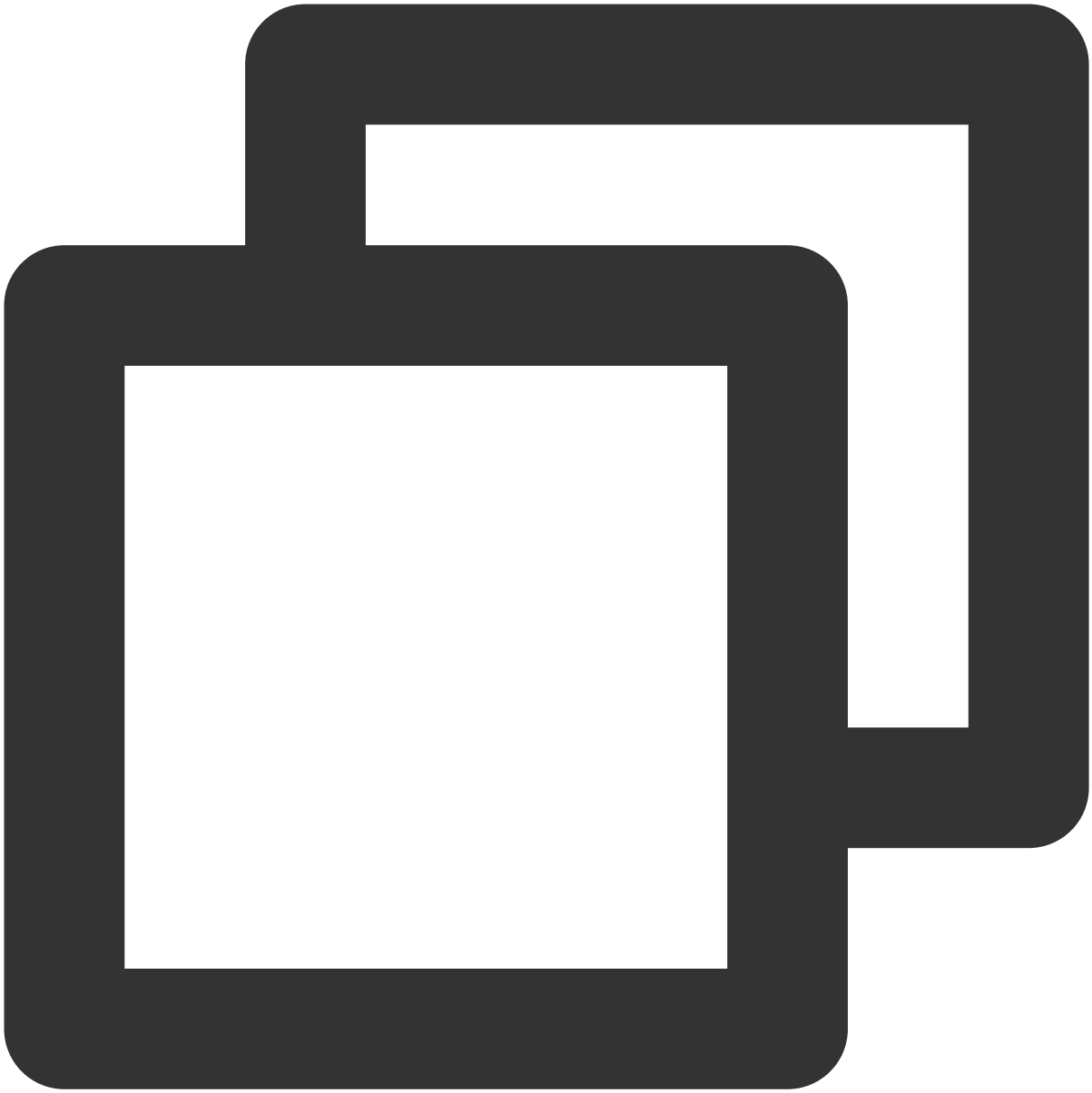
このコールバックインターフェースを追加すると、この応答を通じて `TUICallObserver` 関連のイベントコールバックを監視できます。



```
- (void)addObserver:(id<TUICallObserver>)observer;
```

removeObserver

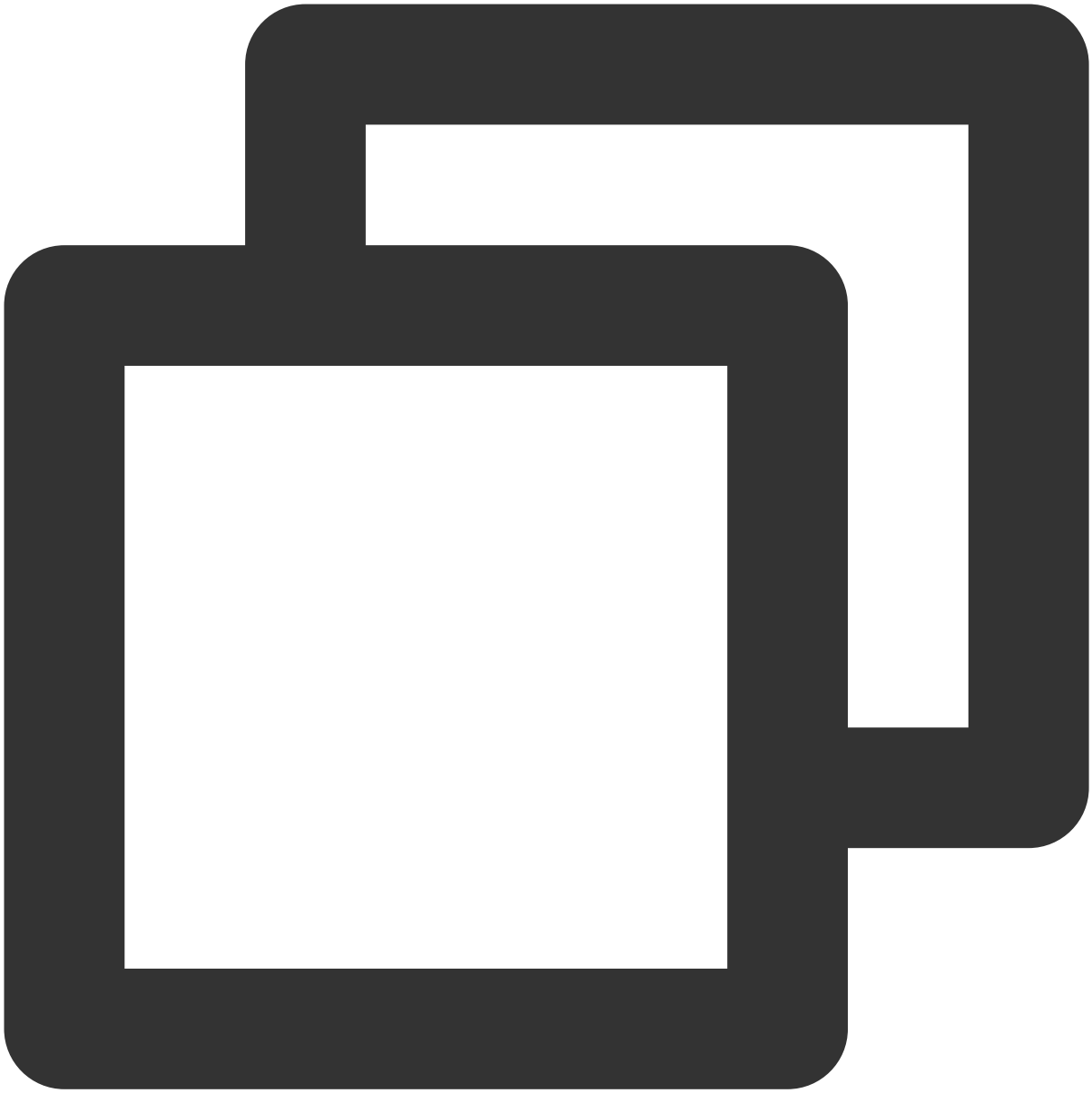
コールバックインターフェースを削除します。



```
- (void)removeObserver:(id<TUICallObserver>)observer;
```

call

電話をかけます（1v1通話）



```
- (void)call:(TUIRoomId *)roomId userId:(NSString *)userId callMediaType:(TUICallMe
```

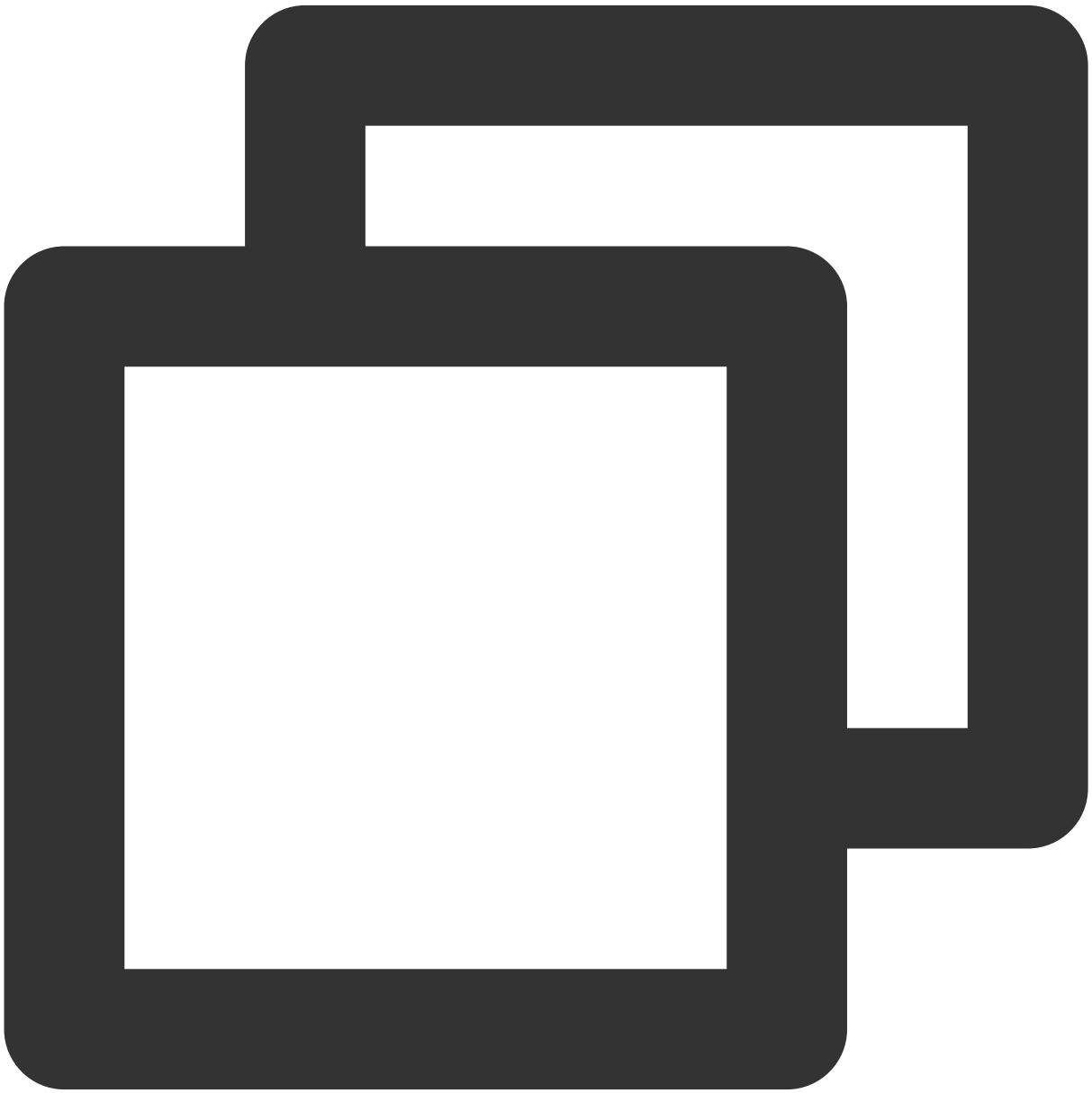
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
roomId	TUIRoomId	今回の通話のオーディオビデオルームID。現在は数字のルームナンバーのみサポートしています。文字列のルームナンバーは今後のバージョンでサポート予定です
userId	NSString	ターゲットユーザーのuserId

callMediaType	TUICallMediaType	通話のメディアタイプ。ビデオ通話、音声通話など
params	TUICallParams	通話パラメータ拡張フィールド。例：カスタムコンテンツのオフラインプッシュ

groupCall

グループ通話を開始します。注意：グループ通話を使用する前にIMグループを作成する必要があります。作成済みの場合は無視してください。

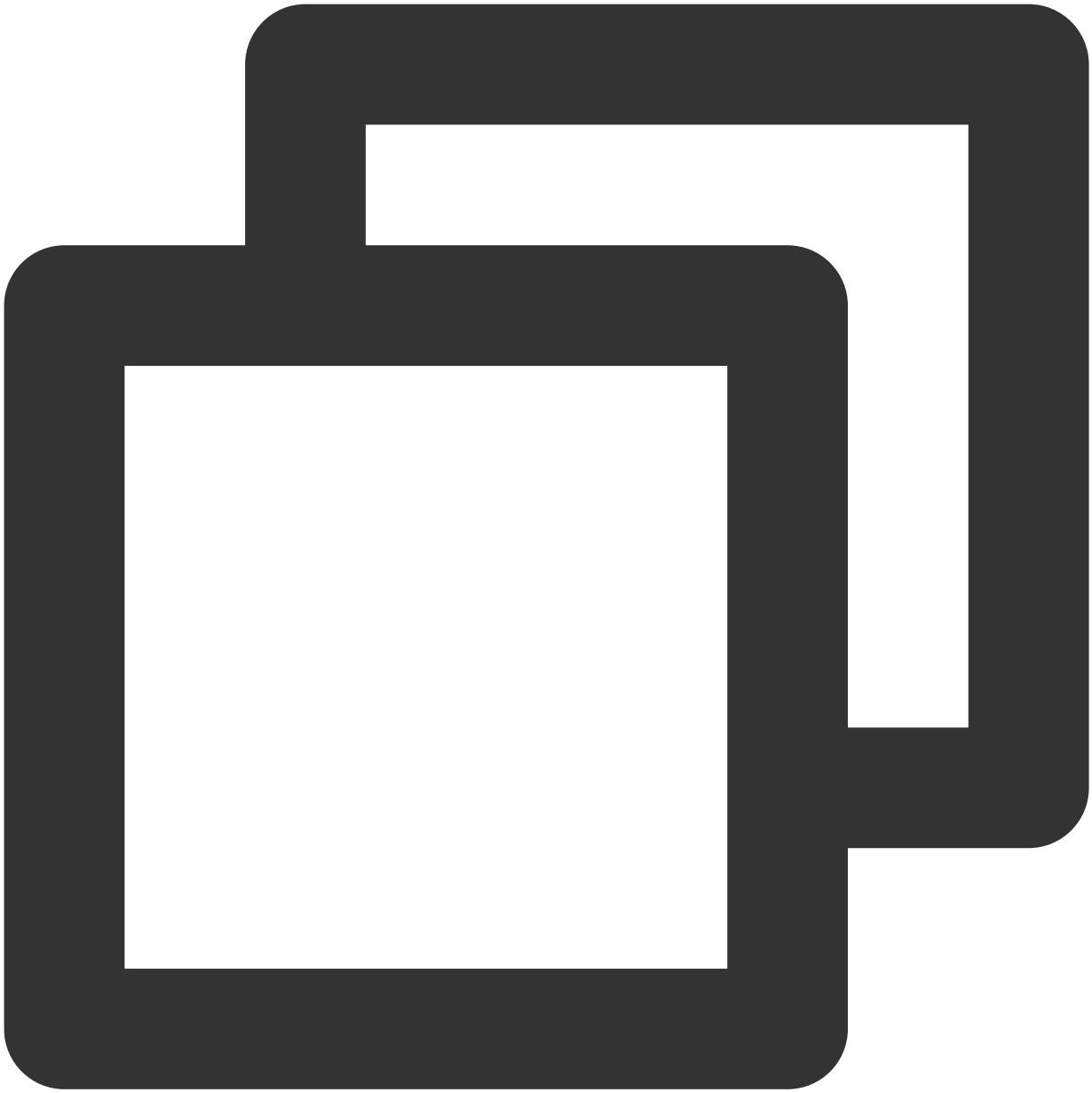


```
- (void)groupCall:(TUIRoomId *)roomId groupId:(NSString *)groupId userIdList:(NSArr
```

パラメータ	タイプ	意味
roomId	TUIRoomId	今回の通話のオーディオビデオルームID。現在は数字のルームナンバーのみサポートしています。文字列のルームナンバーは今後のバージョンでサポート予定です
groupId	NSString	今回のグループ通話のグループID
userIdList	NSArray	ターゲットユーザーのuserIdリスト
callMediaType	TUICallMediaType	通話のメディアタイプ。ビデオ通話、音声通話など
params	TUICallParams	通話パラメータ拡張フィールド。例：カスタムコンテンツのオフラインプッシュ

accept

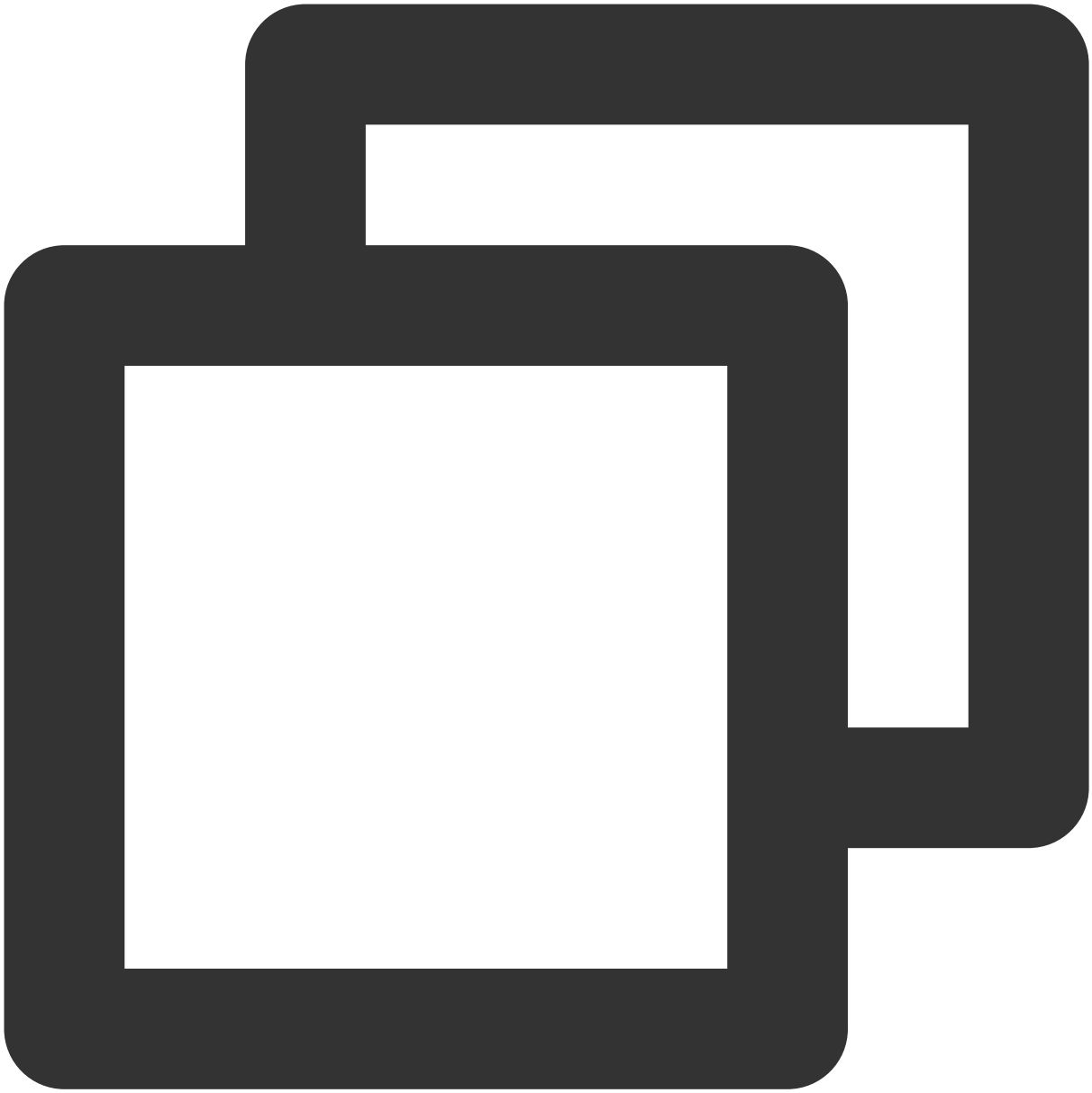
現在の通話を受信します。着呼側として `onCallReceived()` のコールバックを受信した場合は、この関数を呼び出して通話に応答することができます。



```
- (void)accept:(TUICallSucc)succ fail:(TUICallFail)fail;
```

reject

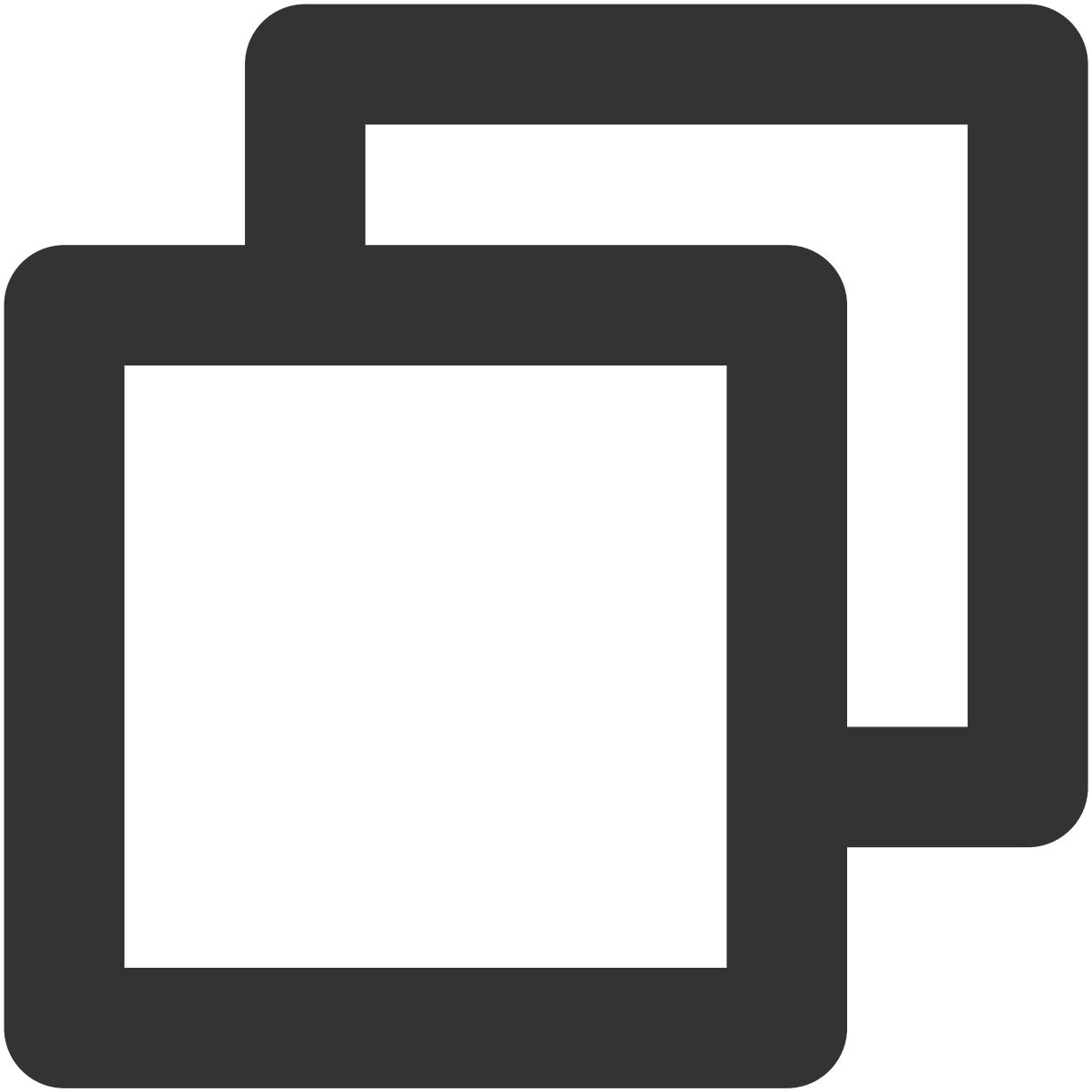
現在の通話を拒否します。着呼側として `onCallReceived()` のコールバックを受信した場合は、この関数を呼び出して通話を拒否することができます。



```
- (void)reject:(TUICallSucc)succ fail:(TUICallFail)fail;
```

ignore

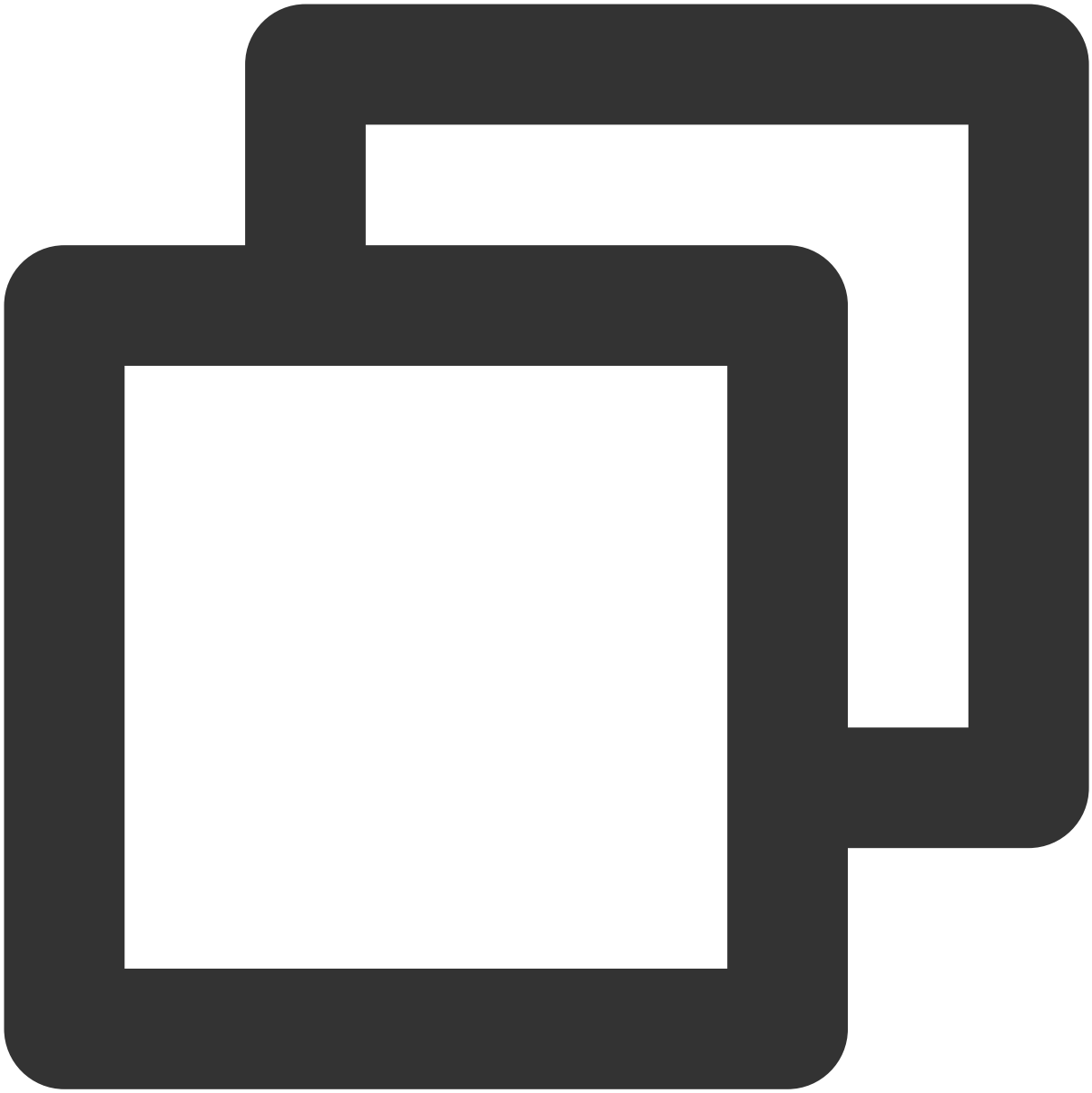
現在の通話を無視します。着呼側として `onCallReceived()` のコールバックを受信した場合は、この関数を呼び出して通話を無視することができ、このとき発呼側は `onUserLineBusy` のコールバックを受信します。備考：業務内にライブストリーミング、ミーティングなどのシーンがある場合、ライブストリーミング/ミーティング中の場合もこの関数を呼び出して通話を無視することができます。



```
- (void)ignore:(TUICallSucc)succ fail:(TUICallFail)fail;
```

hangup

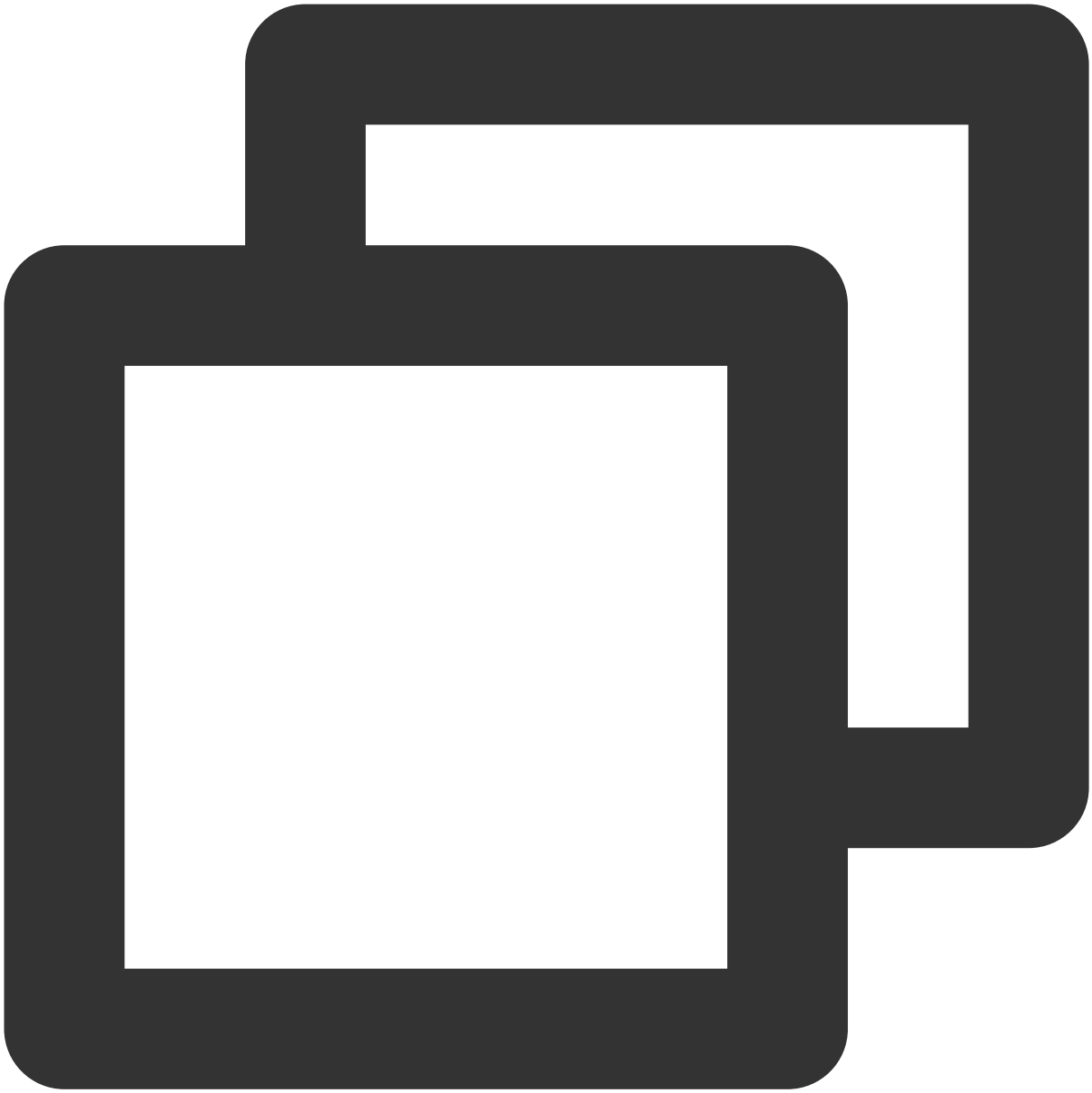
現在の通話を終了します。通話中である場合は、この関数を呼び出して通話を終了できます。



```
- (void)hangup:(TUICallSucc)succ fail:(TUICallFail)fail;
```

inviteUser

今回のグループ通話にユーザーを招待します。ユースケース：グループ通話中のユーザーが自主的に他の人を招待する場合に使用します。

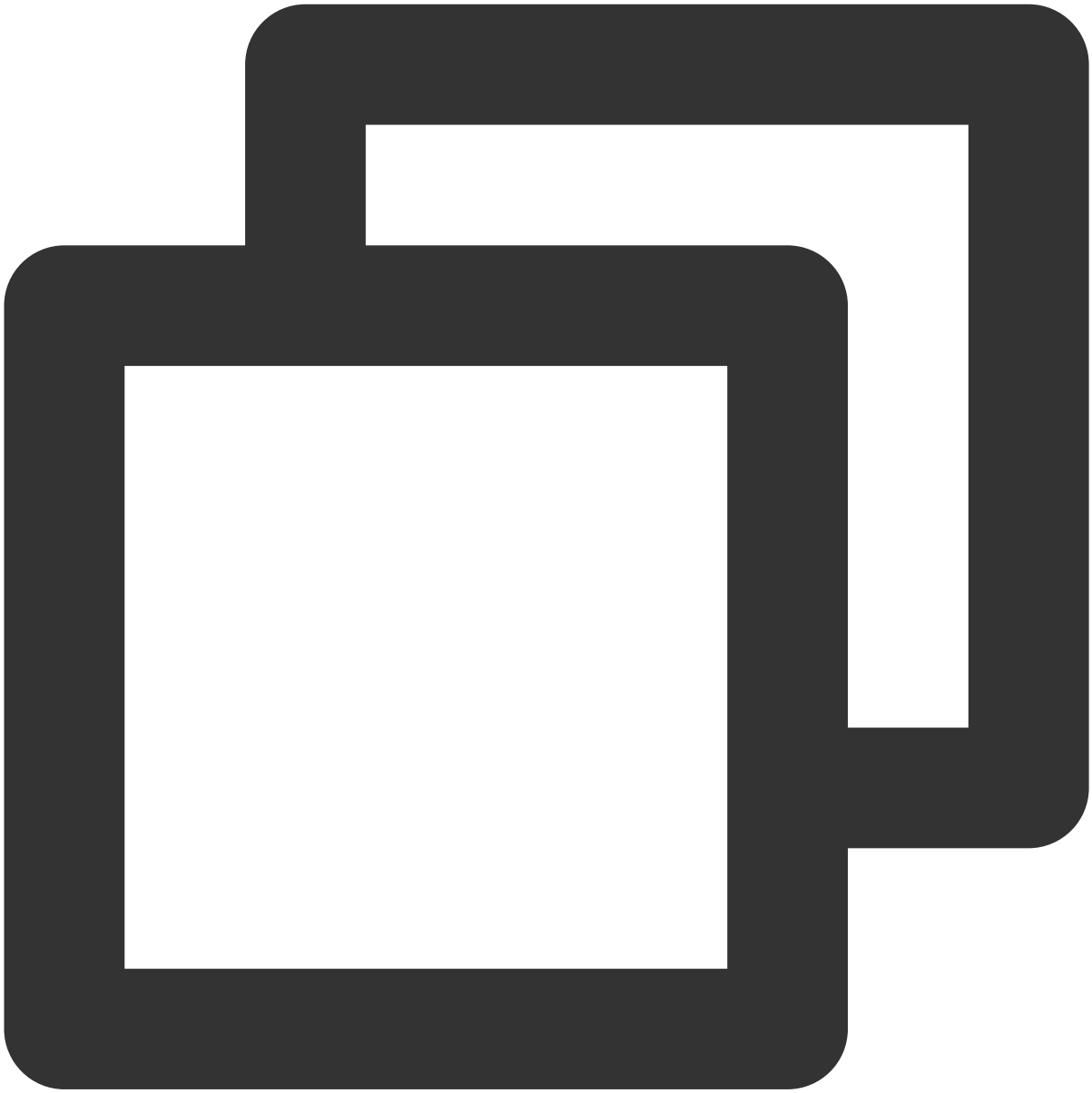


```
- (void)inviteUser:(NSArray<NSString *> *)userIdList params:(TUICallParams *)params
```

パラメータ	タイプ	意味
userIdList	NSArray	ターゲットユーザーのuserIdリスト
params	TUICallParams	通話パラメータ拡張フィールド。例：カスタムコンテンツのオフラインプッシュ

joinInGroupCall

今回のグループ通話に自主的に参加します。ユースケース：グループ内のユーザーが今回のグループ通話に自主的に参加する場合に使用します。



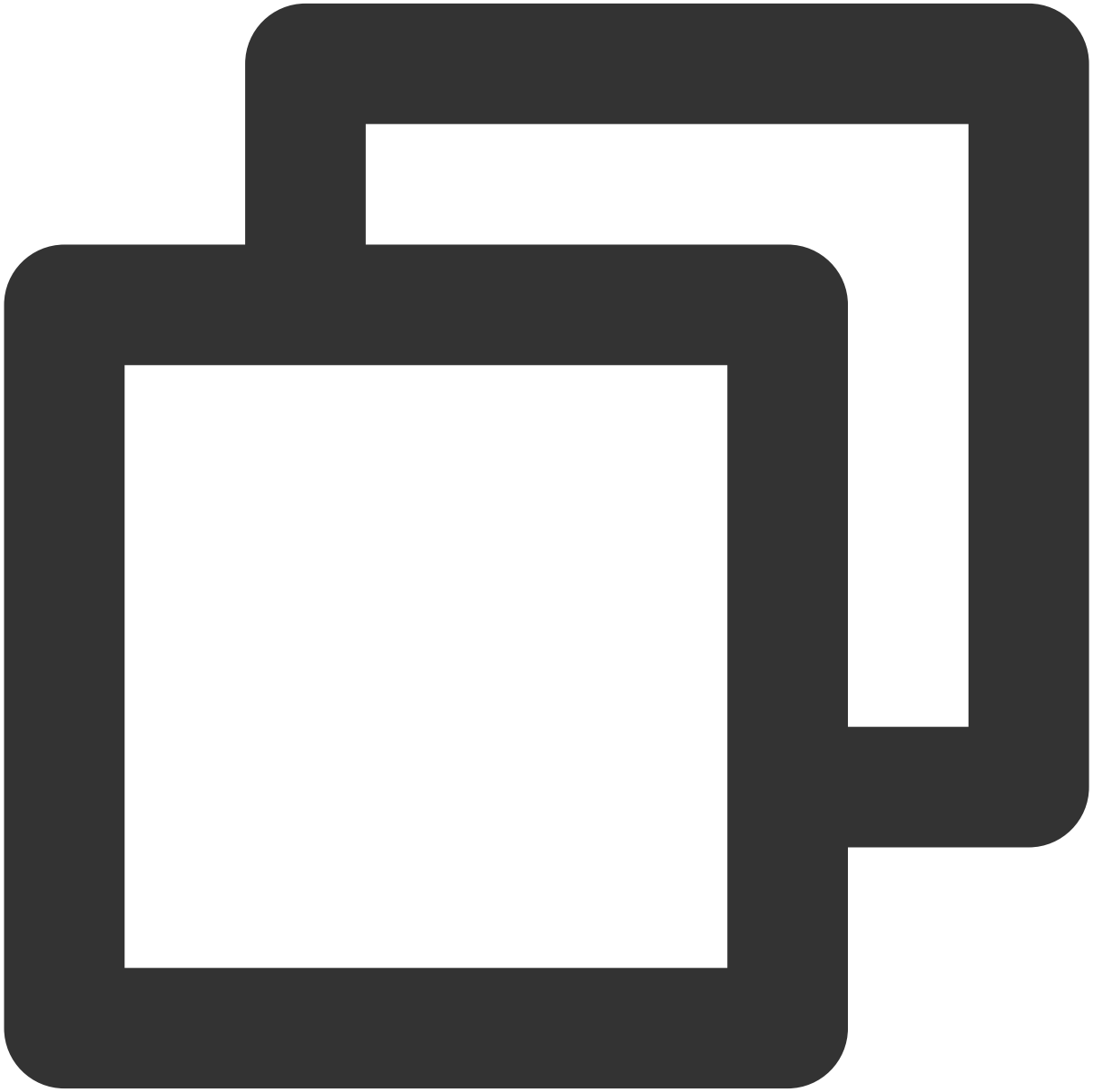
```
- (void)joinInGroupCall:(TUIRoomId *)roomId groupId:(NSString *)groupId callMediaTy
```

パラメータ	タイプ	意味
roomId	TUIRoomId	今回の通話のオーディオビデオルームID。現在は数字のルームナンバーのみサポートしています。文字列のルームナンバーは今後の

		バージョンでサポート予定です
groupId	NSString	今回のグループ通話のグループID
callMediaType	TUICallMediaType	通話のメディアタイプ。ビデオ通話、音声通話など

switchCallMediaType

ビデオ通話を音声通話へ切り替えます。

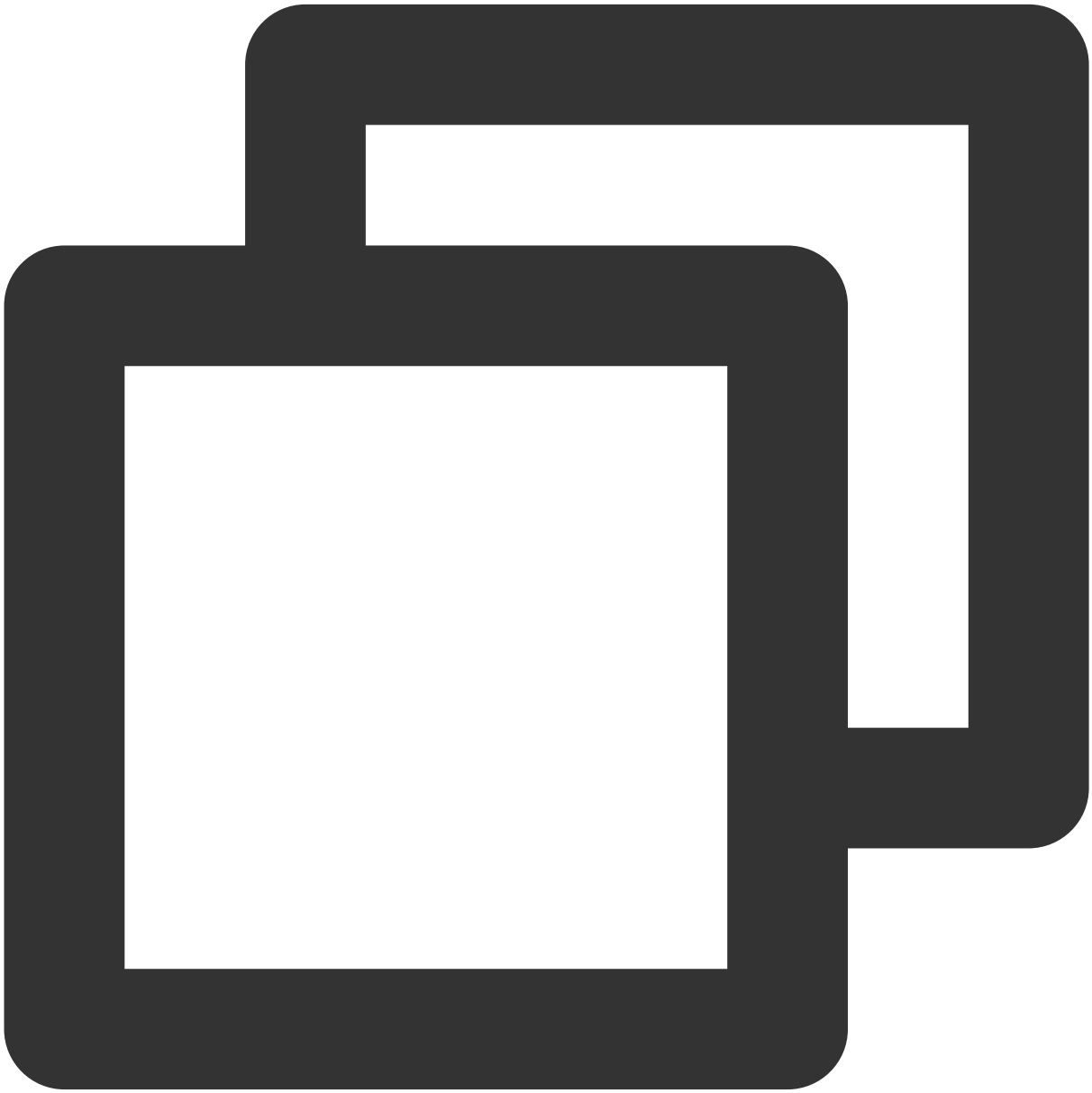


```
- (void)switchCallMediaType:(TUICallMediaType)newType;
```

パラメータ	タイプ	意味
callMediaType	TUICallMediaType	通話のメディアタイプ。ビデオ通話、音声通話など

startRemoteView

ビデオ画面に表示するViewオブジェクトを設定します



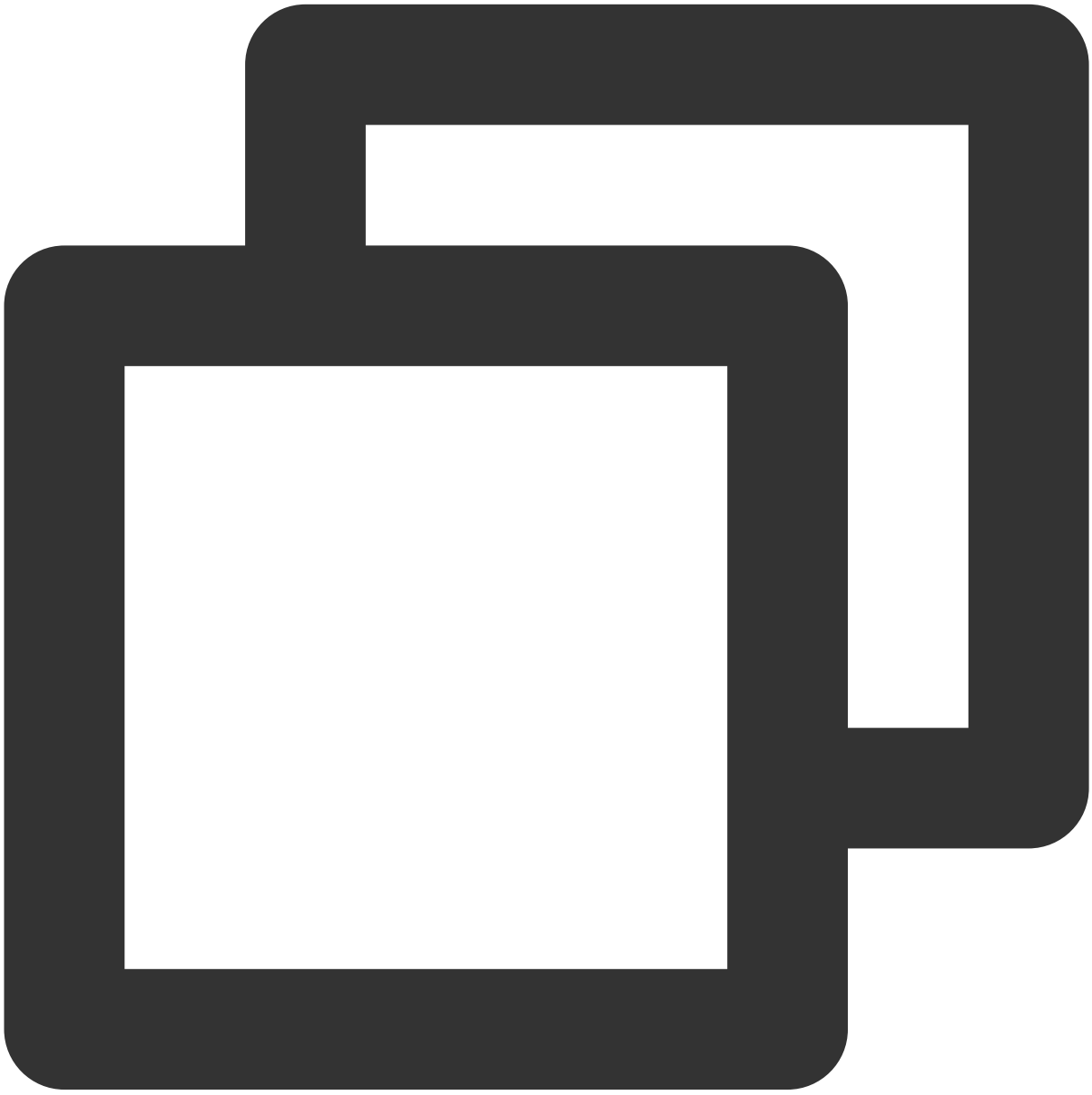
```
- (void)startRemoteView:(NSString *)userId videoView:(TUIVideoView *)videoView onPl
```

パラメータ	タイプ	意味

userId	NSString	ターゲットユーザーのuserId
videoView	TUIVideoView	レンダリング対象のビュー

stopRemoteview

リモートユーザーのビデオデータのサブスクリプションを停止します。

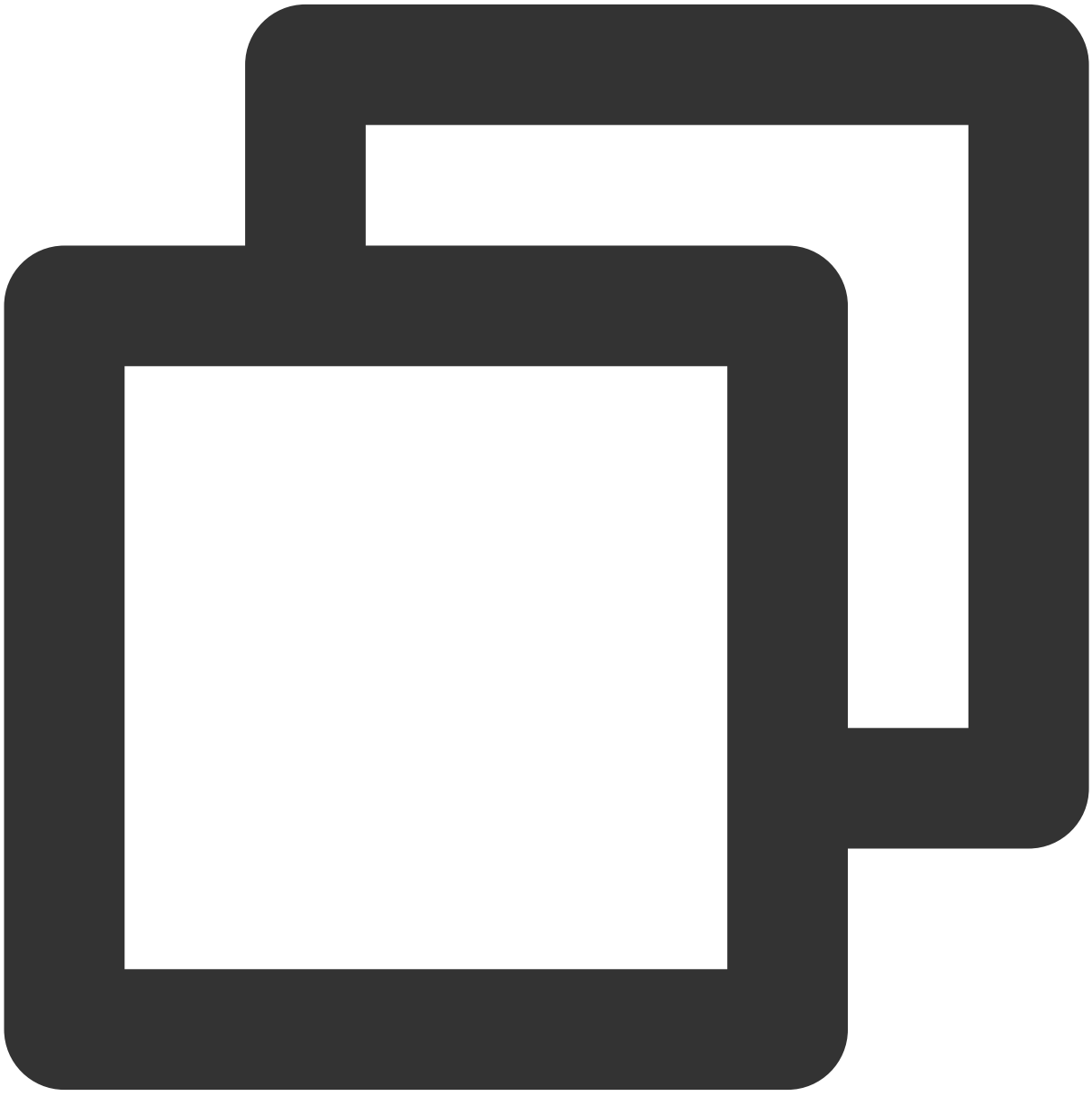


```
- (void)stopRemoteView:(NSString *)userId;
```

パラメータ	タイプ	意味
userId	NSString	ターゲットユーザーのuserId

openCamera

カメラをオンにする



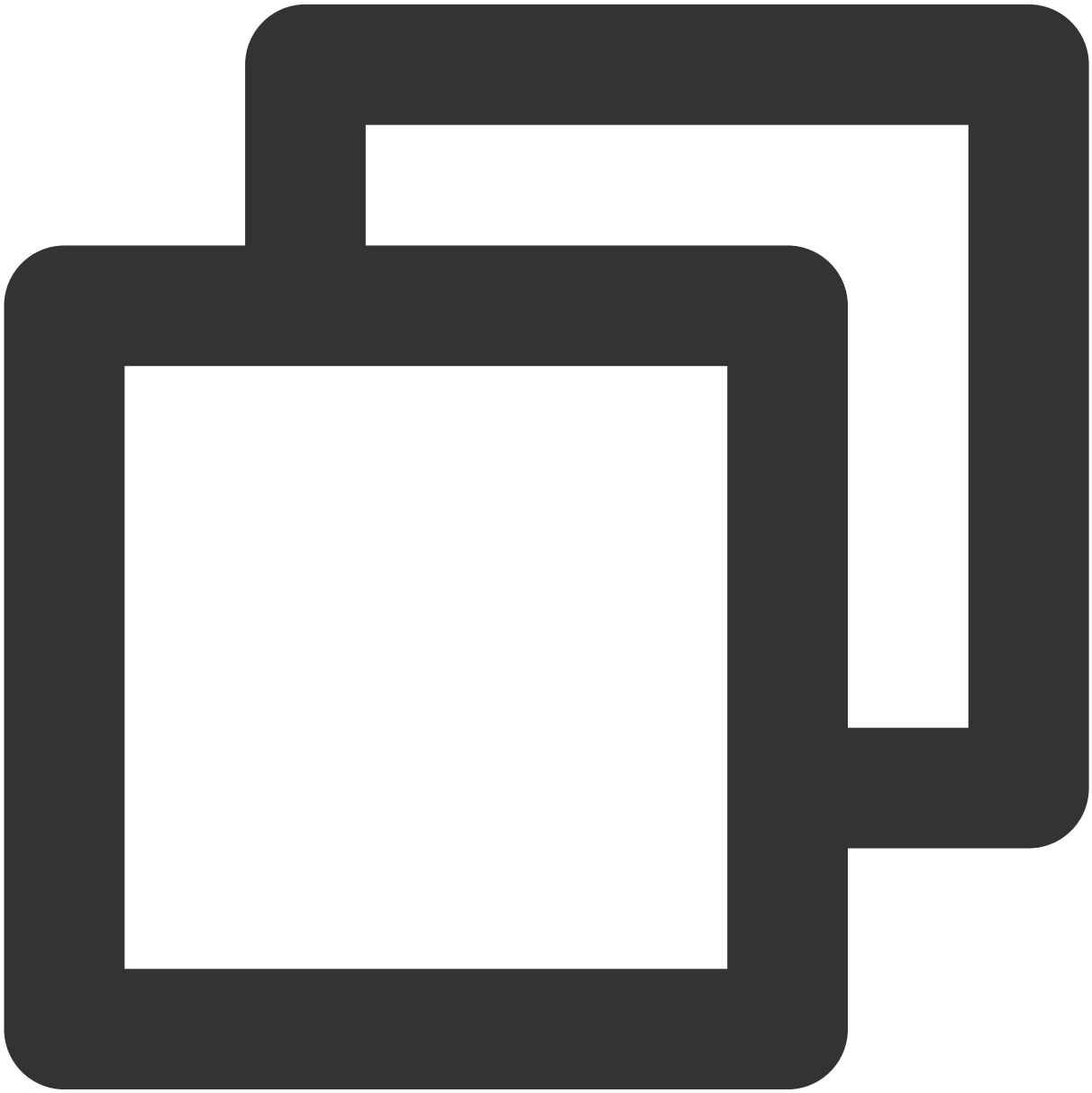
```
- (void)openCamera:(TUICallCamera)camera videoView:(TUIVideoView *)videoView succ:(
```

パラメータ	タイプ	意味

camera	TUICallCamera	フロントカメラ/リアカメラ
videoView	TUIVideoView	レンダリング対象のビュー

closeCamera

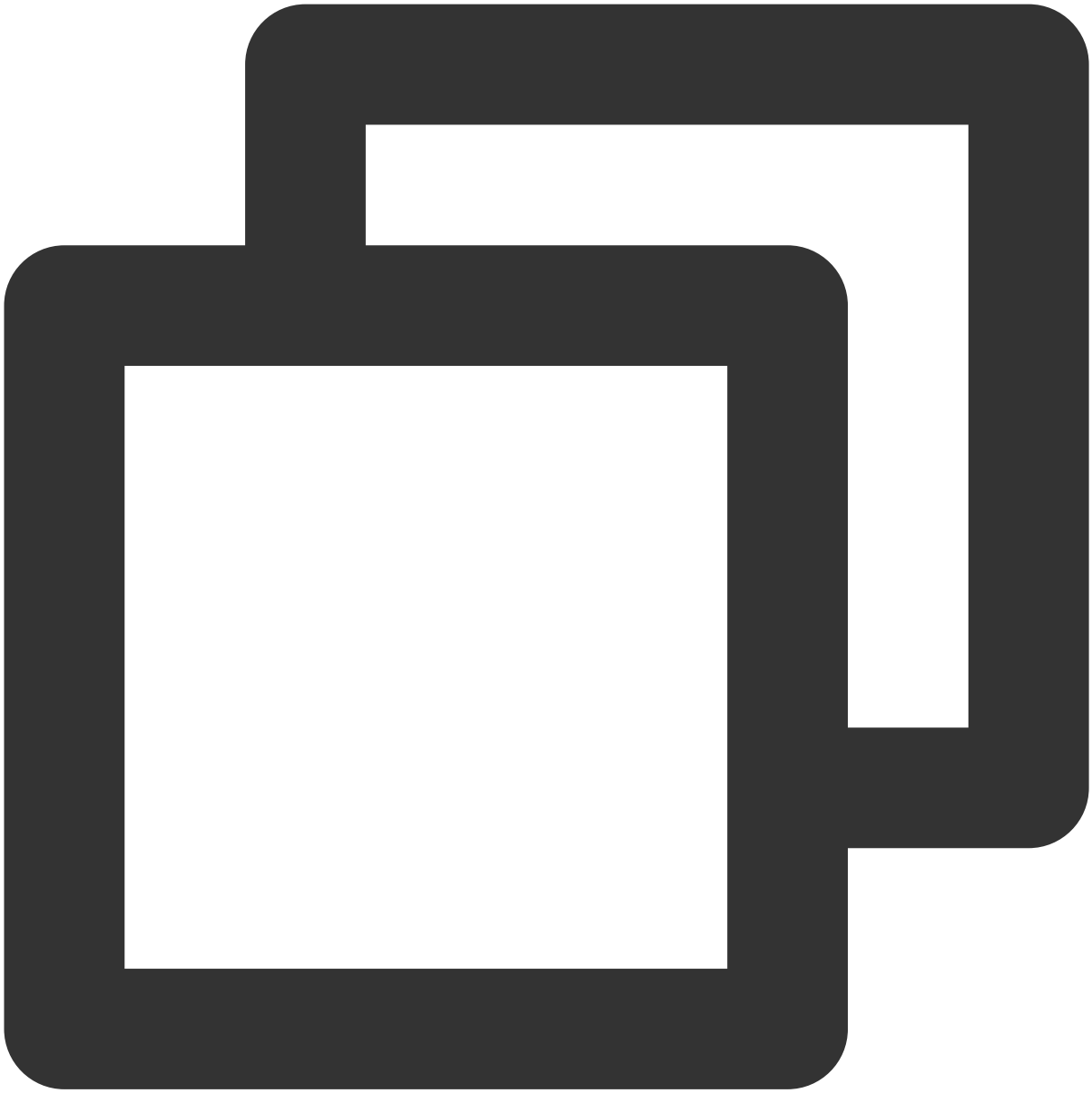
カメラをオフにします。



```
- (void)closeCamera;
```

switchCamera

フロント/リアカメラを切り替えます。

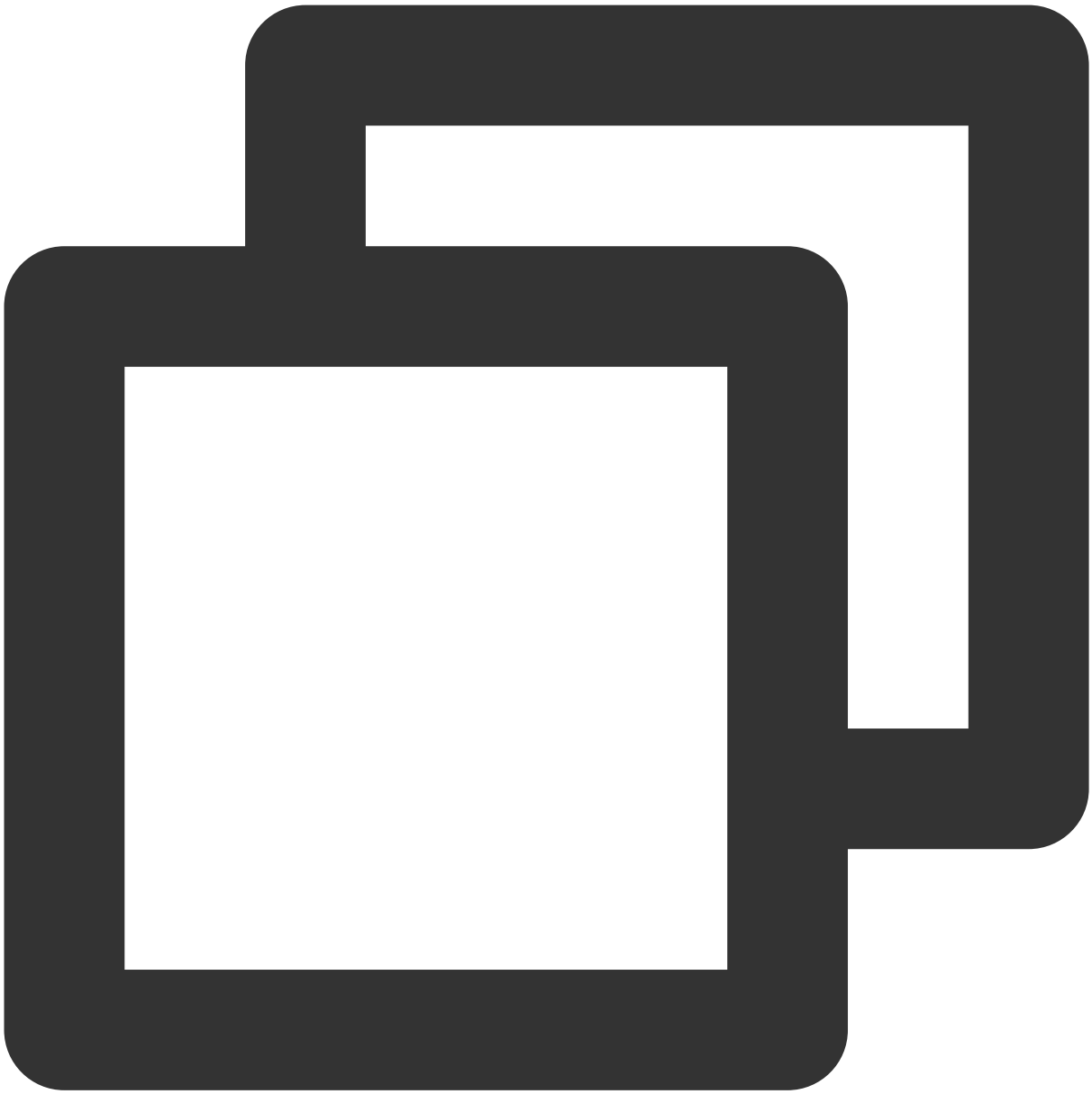


```
- (void)switchCamera:(TUICallCamera)camera;
```

パラメータ	タイプ	意味
camera	TUICallCamera	フロントカメラ/リアカメラ

openMicrophone

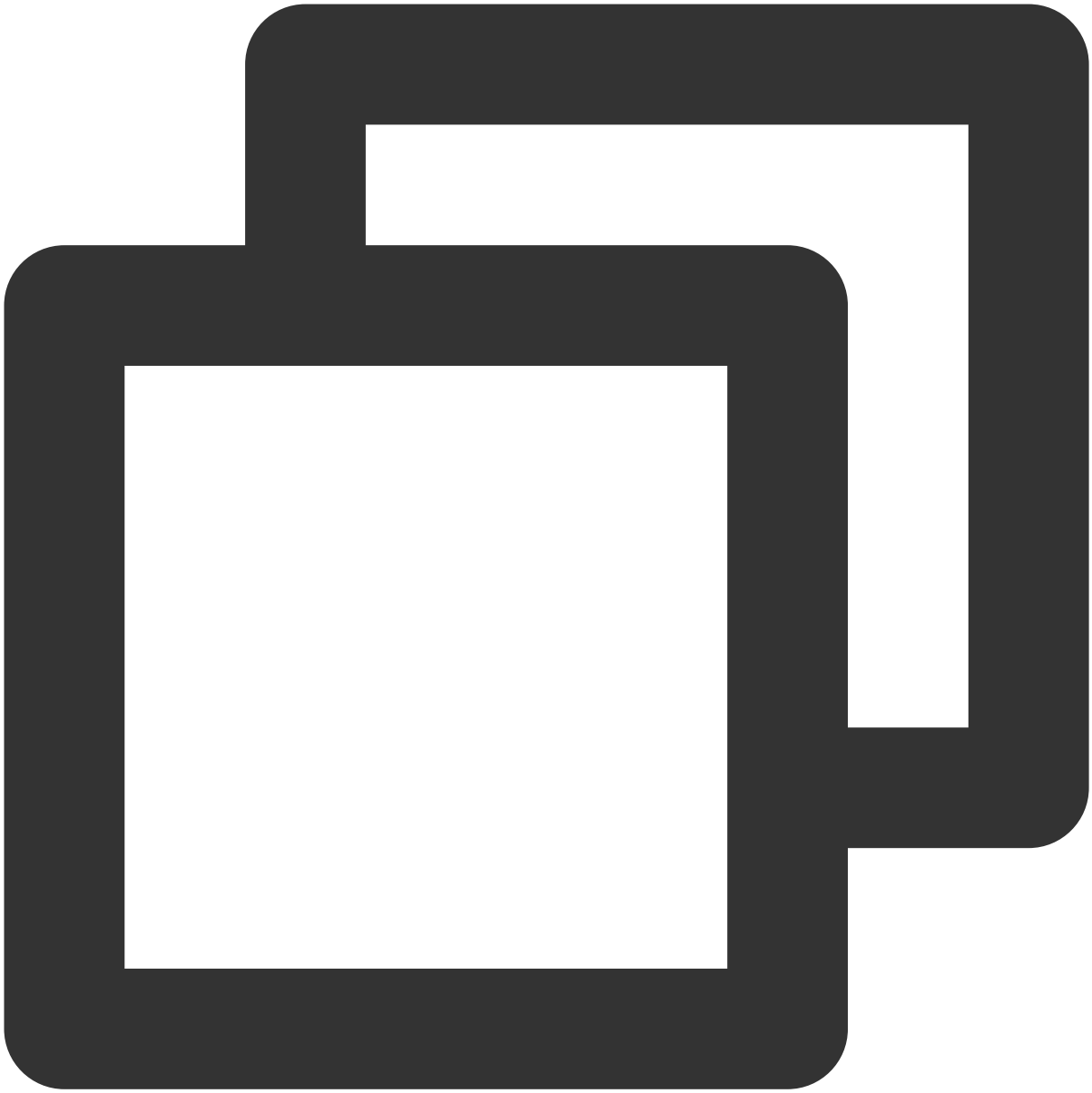
マイクをオンにします



```
- (void)openMicrophone:(TUICallSucc)succ fail:(TUICallFail)fail;
```

closeMicrophone

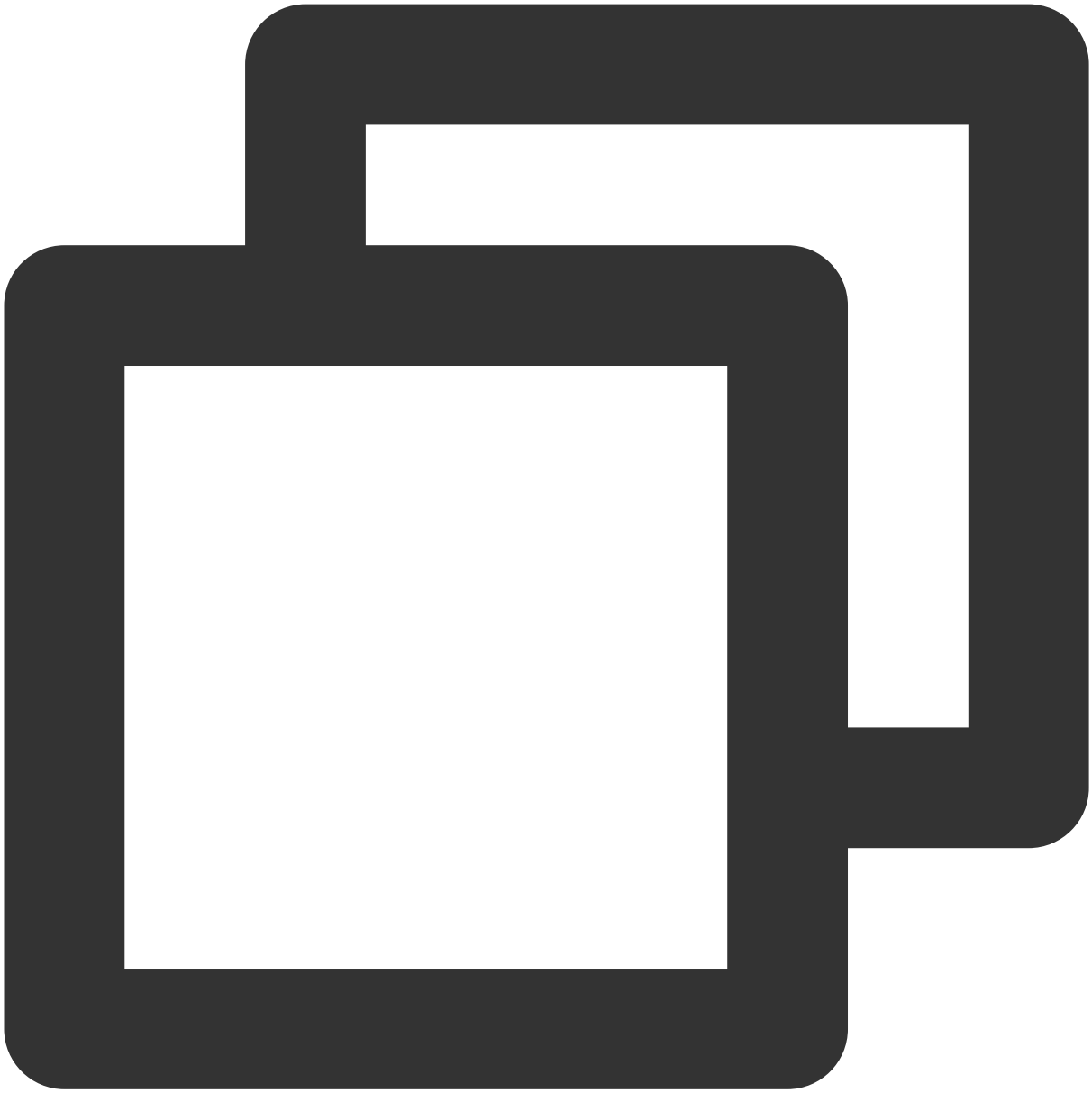
マイクをオフにします。



```
- (void)closeMicrophone;
```

selectAudioPlaybackDevice

オーディオ再生デバイスを選択します。 現在はヘッドホン、スピーカーをサポートしています。通話のシーンでは、このインターフェースを使用してハンズフリーモードのオン/オフが行えます。

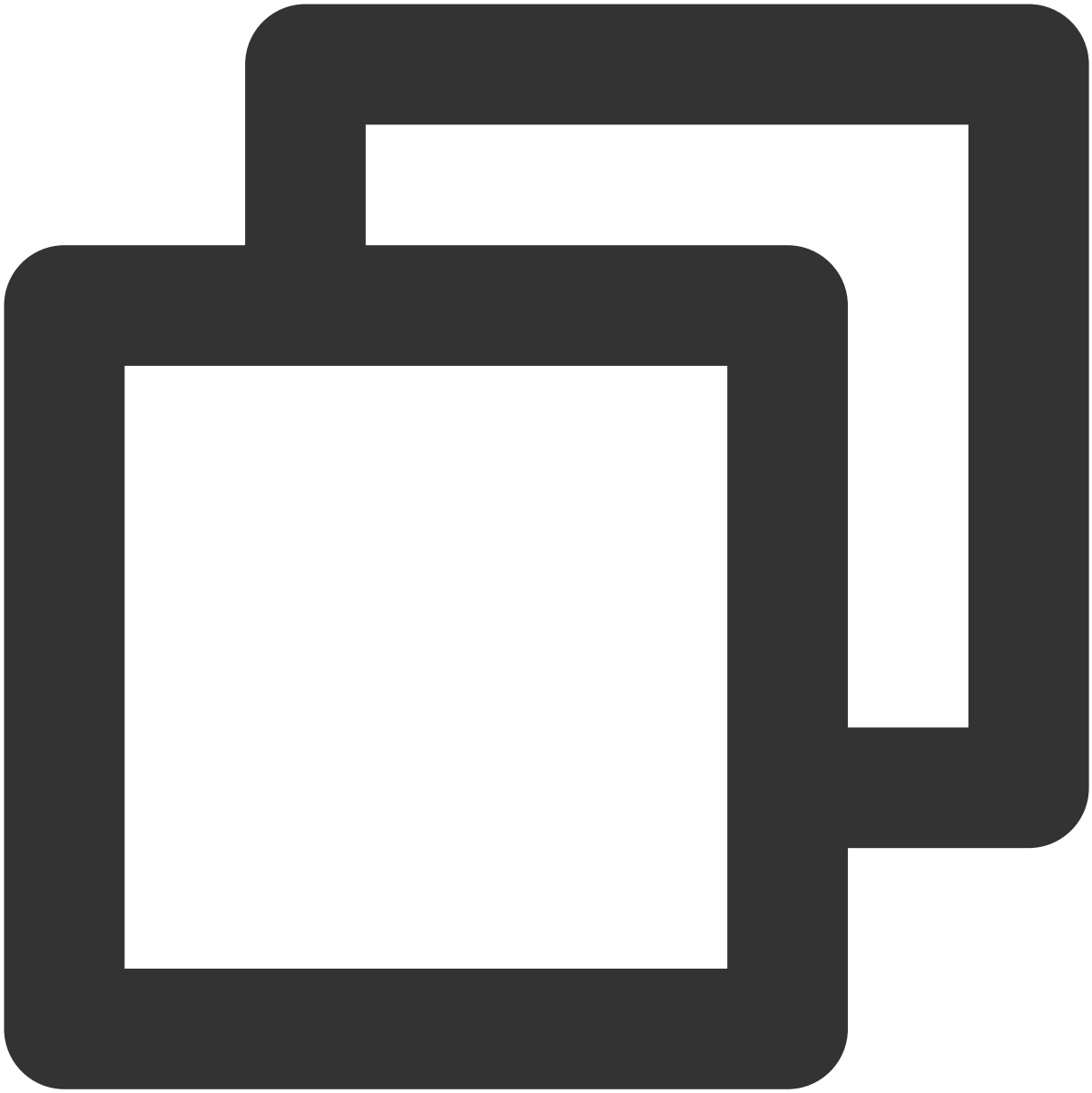


```
- (void)selectAudioPlaybackDevice:(TUIAudioPlaybackDevice)device;
```

パラメータ	タイプ	意味
device	TUIAudioPlaybackDevice	ヘッドホン/スピーカー

setSelfInfo

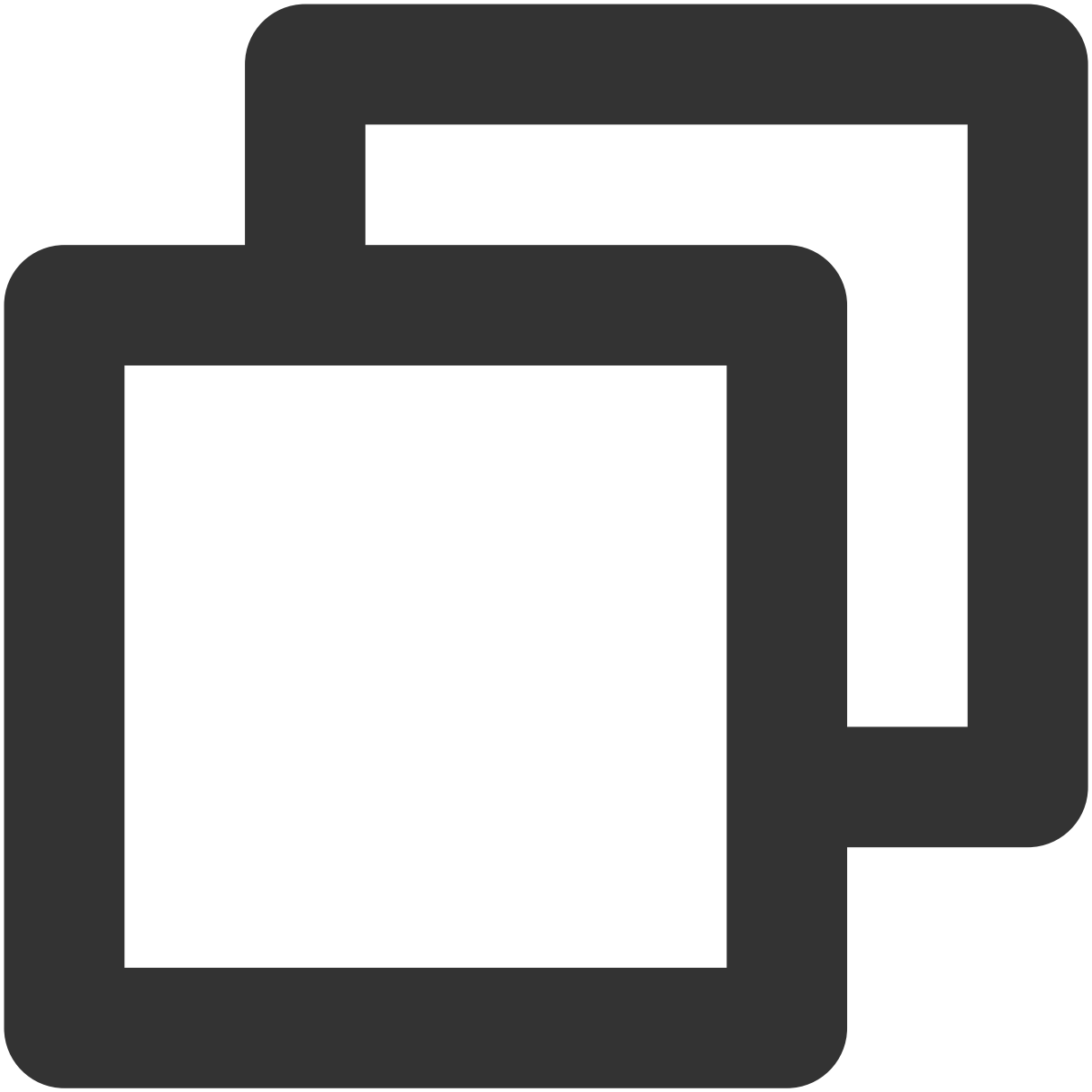
ユーザーニックネーム、プロフィール画像を設定します。ユーザーニックネームは500バイト以内、ユーザープロフィール画像はURL形式でなければなりません。



```
- (void)setSelfInfo:(NSString * _Nullable)nickName avatar:(NSString * _Nullable)ava
```

enableMultiDeviceAbility

TUICallEngineのマルチデバイスログインモードをオン/オフします （プレミアム版パッケージのみサポート）



```
- (void)enableMultiDeviceAbility:(BOOL)enable succ:(TUICallSucc)succ fail:(TUICallF
```

TUICallObserver

最終更新日：：2024-07-19 14:53:21

TUICallObserver APIの概要

TUICallObserverはTUICallEngineに対応するコールバックイベントクラスです。このコールバックによって、関心のあるコールバックイベントを監視することができます。

コールバックイベントの概要

API	説明
onError	通話中のエラーコールバック
onCallReceived	通話リクエストのコールバック
onCallCancelled	通話キャンセルのコールバック
onCallBegin	通話接続のコールバック
onCallEnd	通話終了のコールバック
onCallMediaTypeChanged	通話メディアタイプ変更発生のコールバック
onUserReject	xxxxユーザーによる通話拒否のコールバック
onUserNoResponse	xxxxユーザーの応答なしのコールバック
onUserLineBusy	xxxxユーザーが通話中である場合のコールバック
onUserJoin	xxxxユーザーの通話参加のコールバック
onUserLeave	xxxxユーザーの通話からの退出のコールバック
onUserVideoAvailable	xxxユーザーのビデオストリームの有無のコールバック
onUserAudioAvailable	xxxユーザーのオーディオストリームの有無のコールバック
onUserVoiceVolumeChanged	全ユーザーの音量レベルフィードバックのコールバック
onUserNetworkQualityChanged	全ユーザーのネットワーク品質フィードバックのコールバック

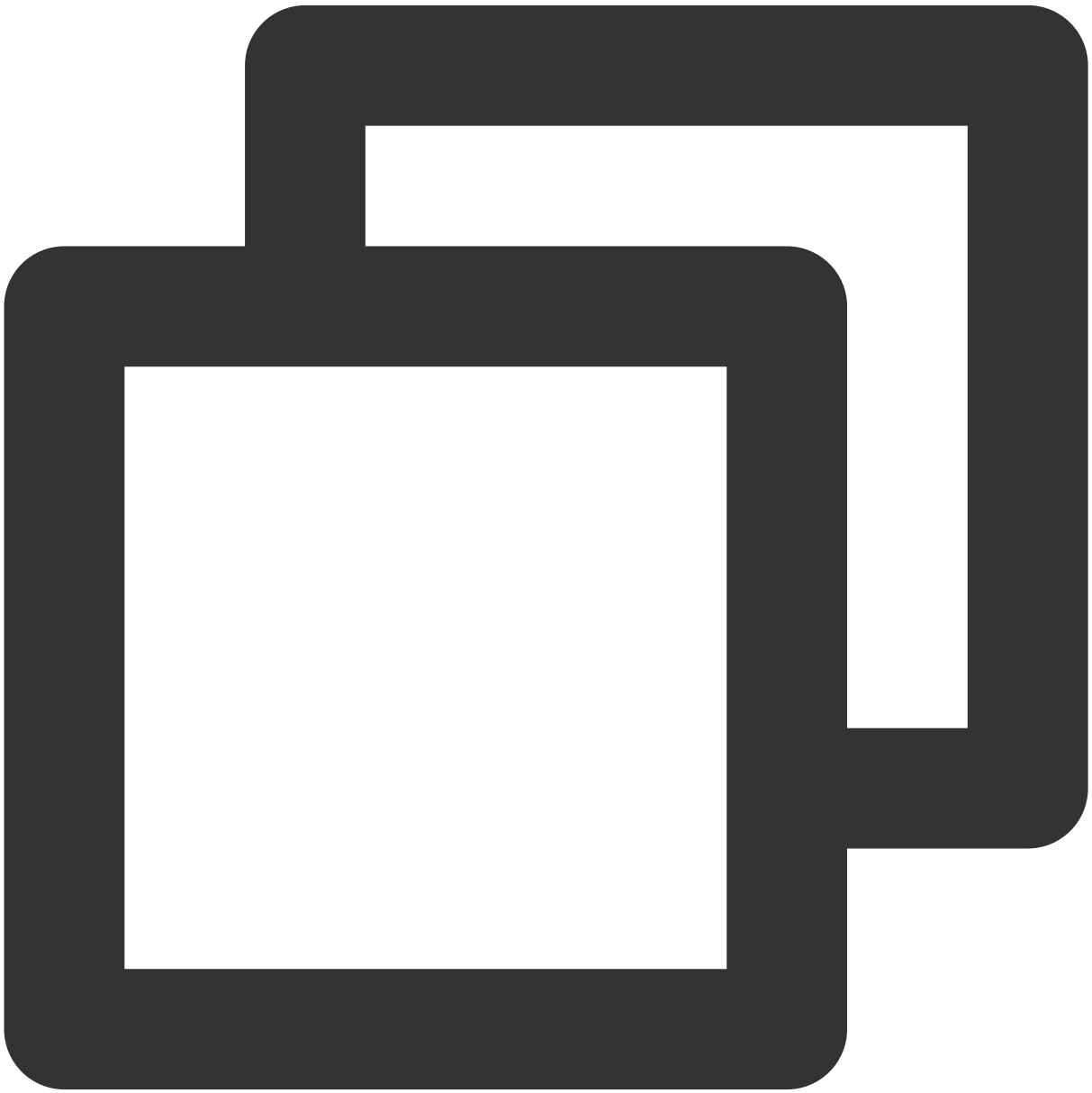
コールバックイベントの詳細

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。



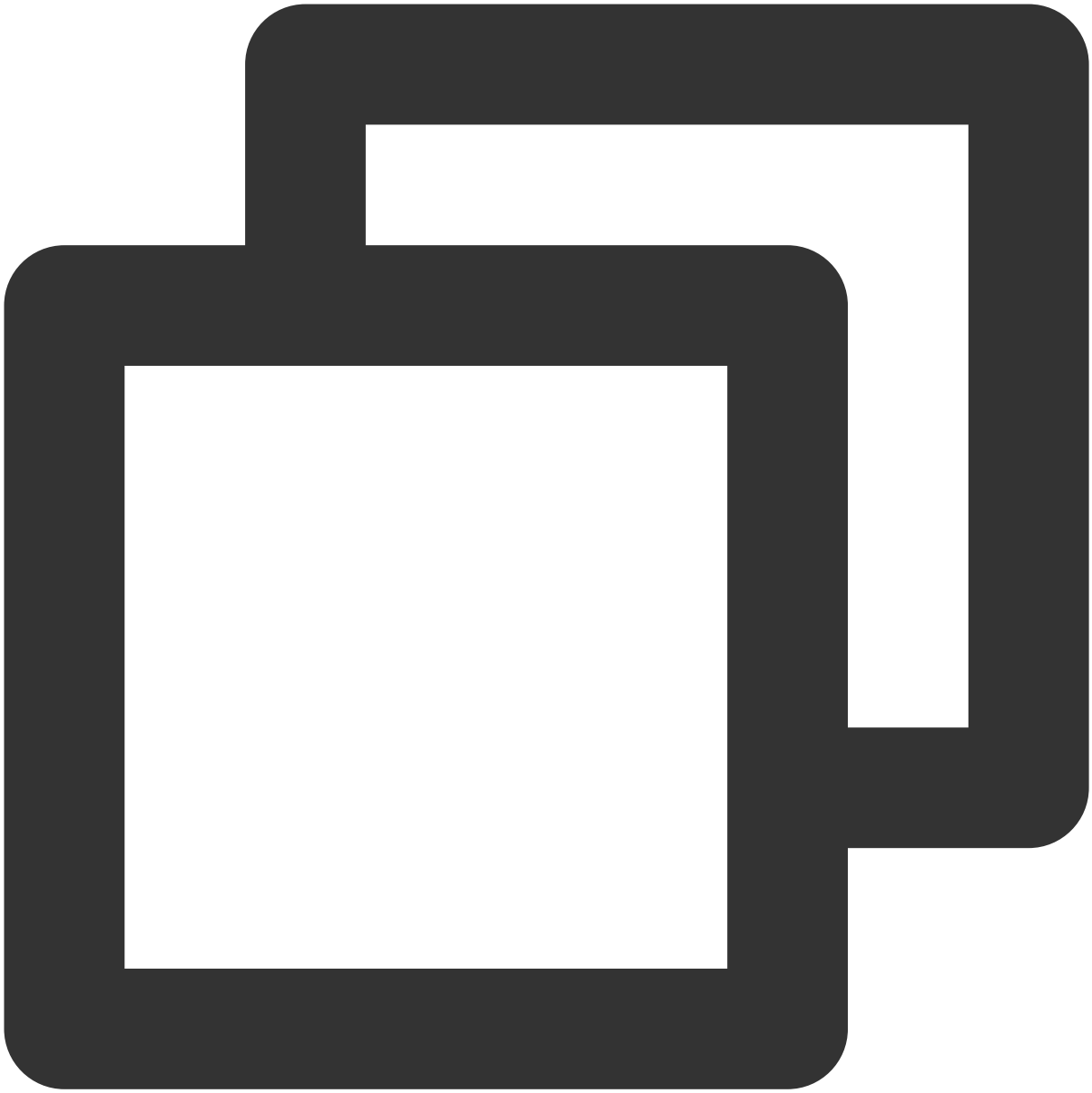
```
- (void)onError:(int)code message:(NSString * _Nullable)message;
```

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

onCallReceived

新しい通話リクエストコールバックを受信します。着呼側が受信します。このイベントを監視することで、通話応答画面を表示するかどうかを決定できます。



```
- (void)onCallReceived:(NSString *)callerId calleeIdList:(NSArray<NSString *> *)cal
```

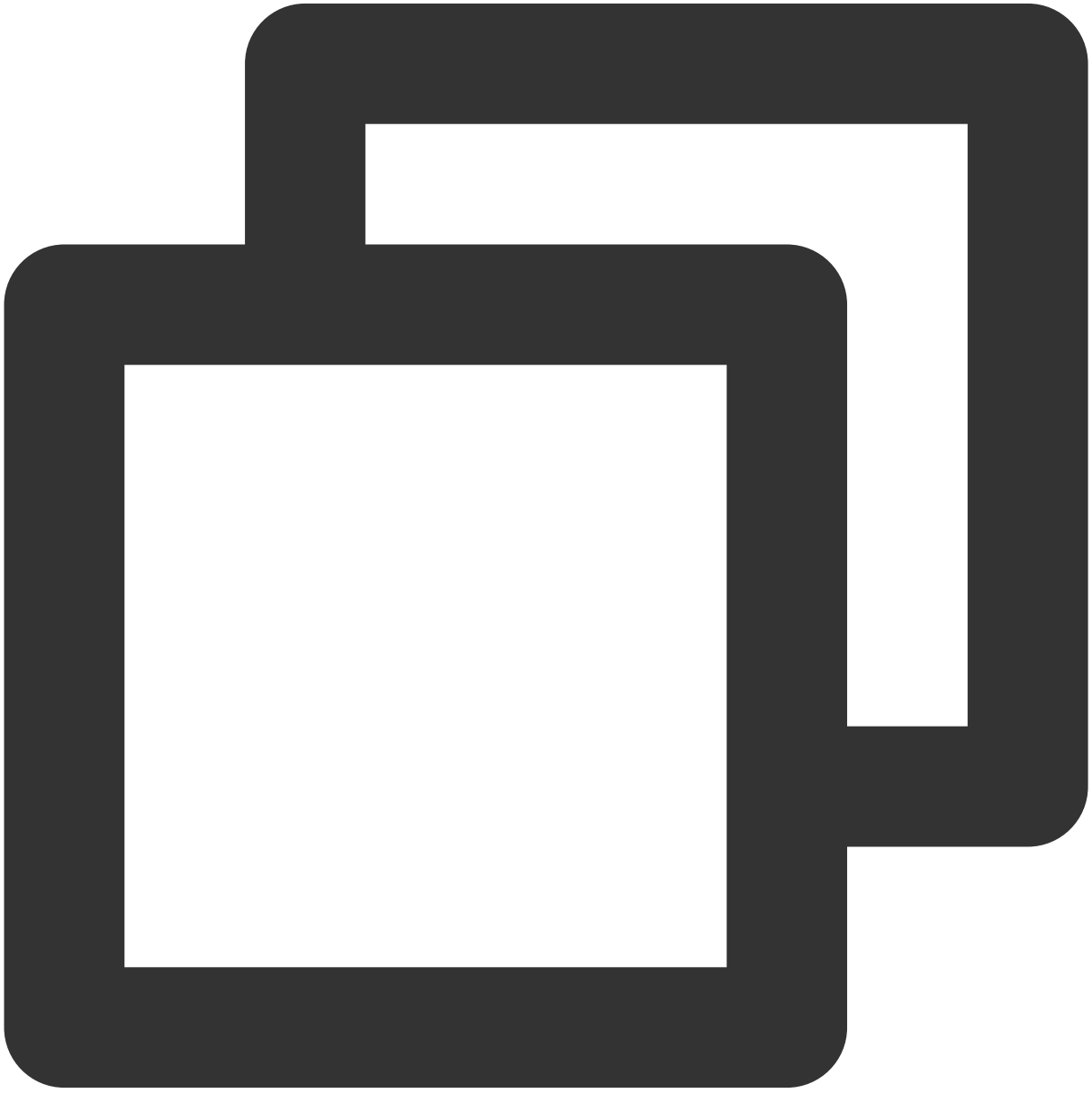
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callerId	NSString	発呼側ID（招待者）
calleeIdList	NSArray	着呼側IDリスト（被招待者）
groupId	NSString	グループ通話ID

callMediaType	TUICallMediaType	通話のメディアタイプ。ビデオ通話、音声通話など

onCallCancelled

今回の通話が発呼側からキャンセルされたことを表します（キャンセルの原因は発呼側の自主的なキャンセル、または通話タイムアウトによるキャンセルの両方の可能性があります）。着呼側が受信します。このイベントを監視することで、未応答通話などに類似した表示ロジックを実現できます。



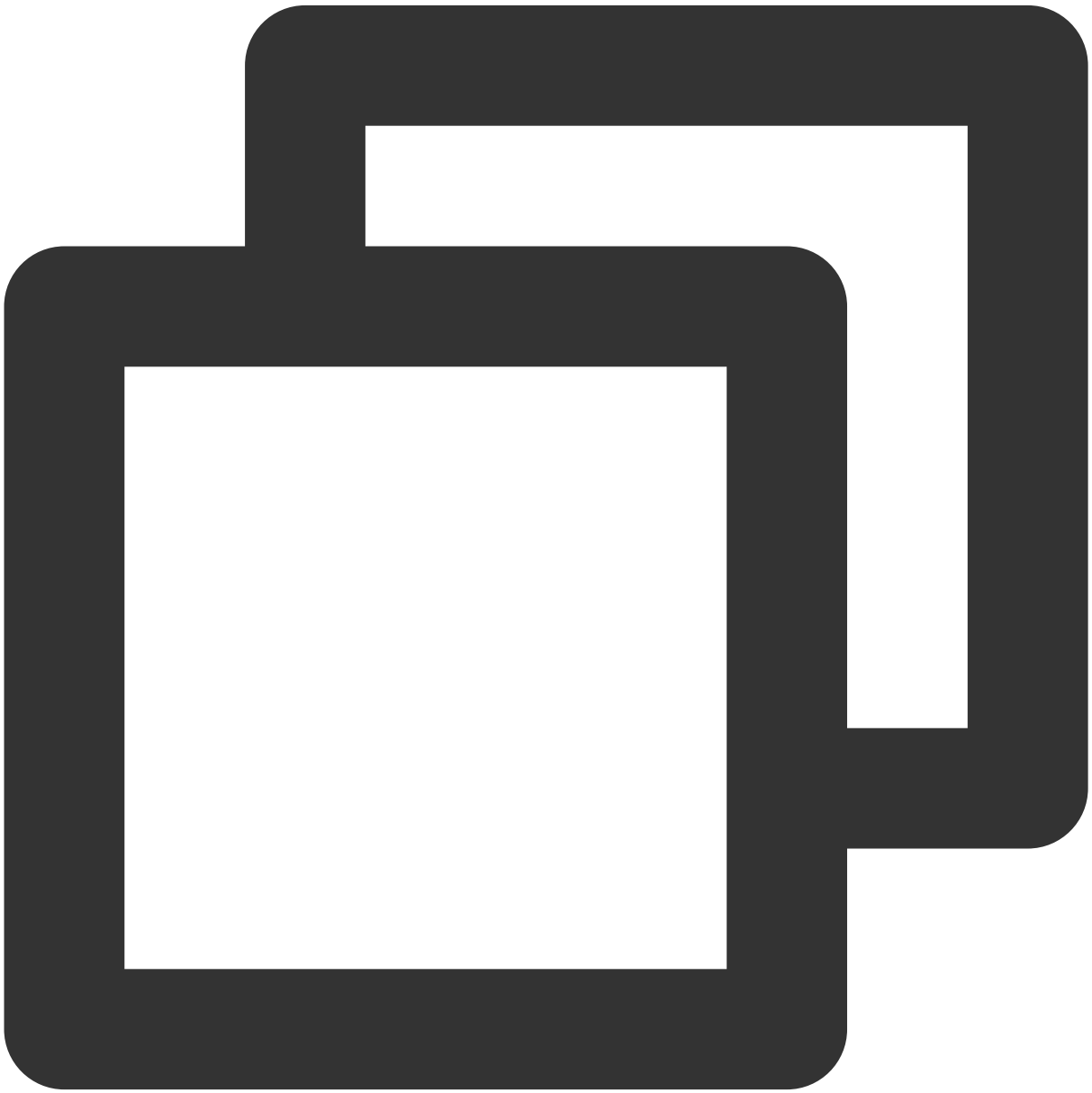
```
- (void)onCallCancelled:(NSString *)callerId;
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
callerId	NSString	キャンセルしたユーザーのID

onCallBegin

通話接続を表します。発呼側と着呼側がどちらも受信できます。このイベントを監視することで、クラウドレコーディング、コンテンツ審査などのフローを開始できます。



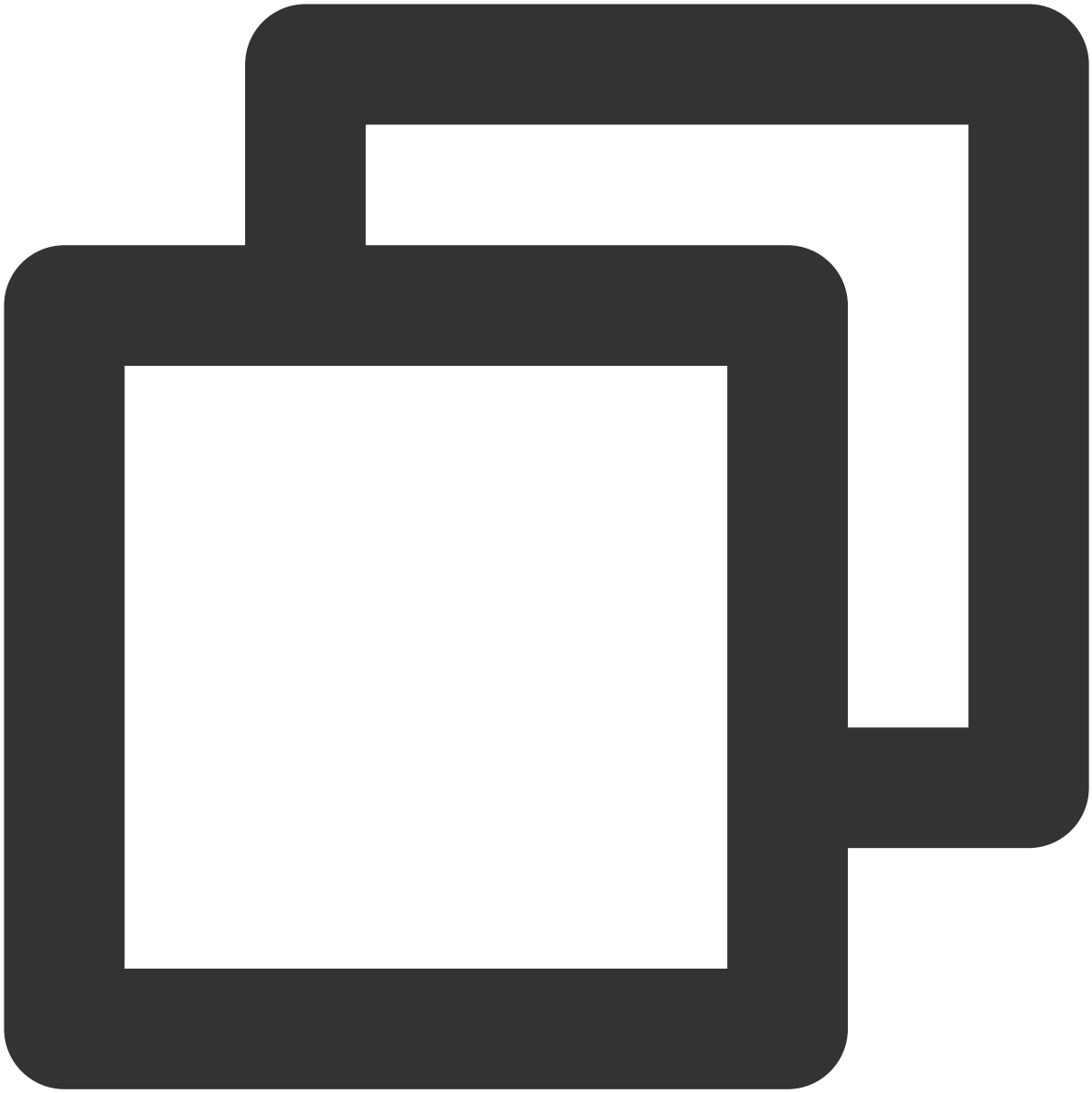
```
- (void)onCallBegin:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaTy
```

パラ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	TUIRoomId	今回の通話のオーディオビデオルームID。現在は数字のルームナンバーのみサポートしています。文字列のルームナンバーは今後のバージョンでサポート予定です
callMediaType	TUICallMediaType	通話のメディアタイプ。ビデオ通話、音声通話
callRole	TUICallRole	ロール。列挙タイプ：発呼側、着呼側

onCallEnd

通話の終了を表します。発呼側と着呼側がどちらも受信できます。このイベントを監視することで、通話時間、通話タイプなどの情報を表示したり、クラウドのレコーディングフローを停止したりすることができます。



```
- (void)onCallEnd:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaType
```

パラ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	TUIRoomId	今回の通話のオーディオビデオルームID。現在は数字のルームナンバーのみサポートしています。文字列のルームナンバーは今後のバージョンでサポート予定です
callMediaType	TUICallMediaType	通話のメディアタイプ。ビデオ通話、音声通話

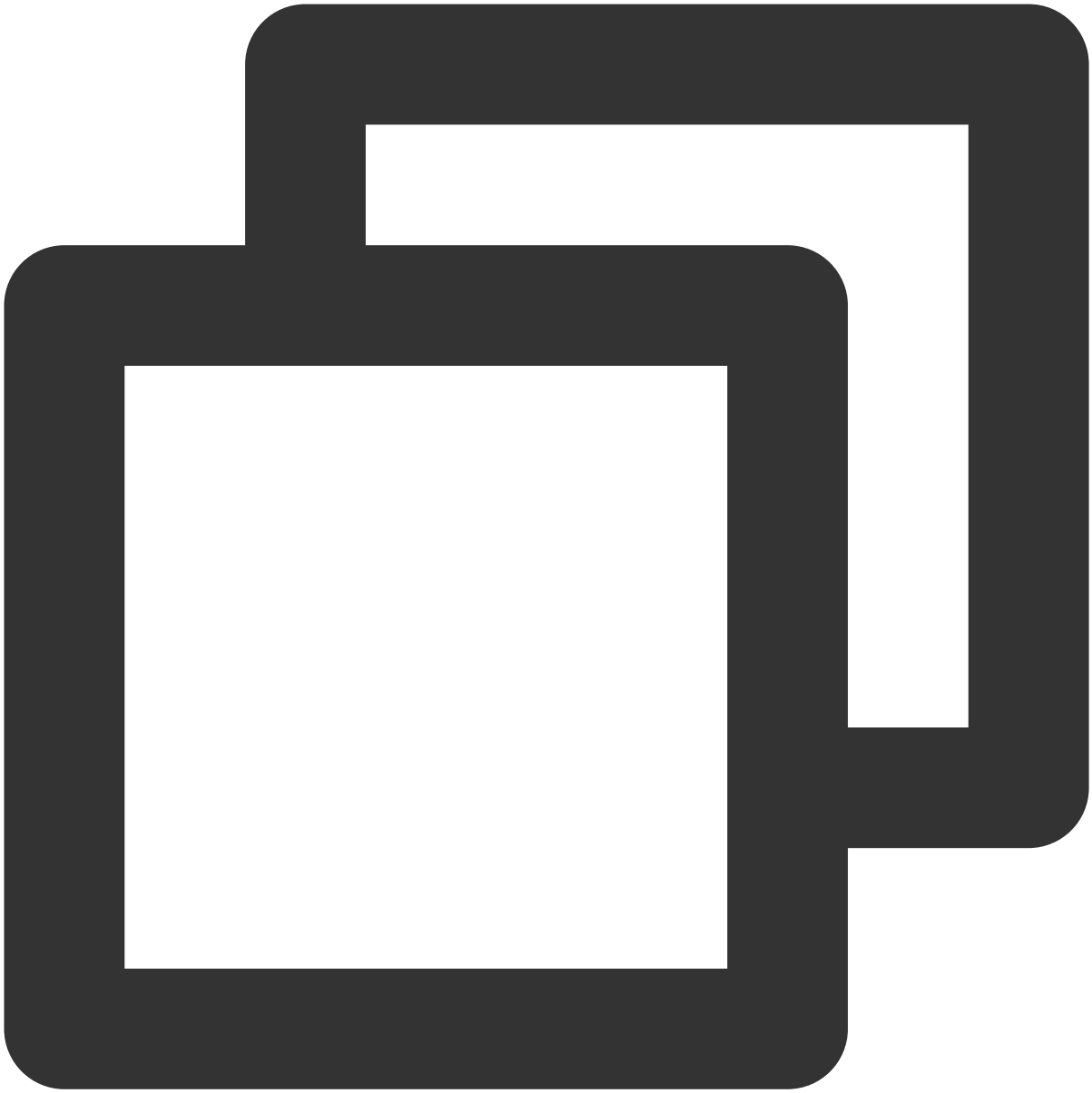
callRole	TUICallRole	ロール。列挙タイプ：発呼側、着呼側
totalTime	float	今回の通話時間

ご注意：

クライアントのイベントは一般的にすべて、プロセスキルなどの異常イベントに伴って消失します。通話時間の監視によって料金計算などのロジックを完成させたい場合は、REST APIを使用してこの種のフローを完成させることをお勧めします。

onCallMediaTypeChanged

通話のメディアタイプに変更が発生したことを表します。



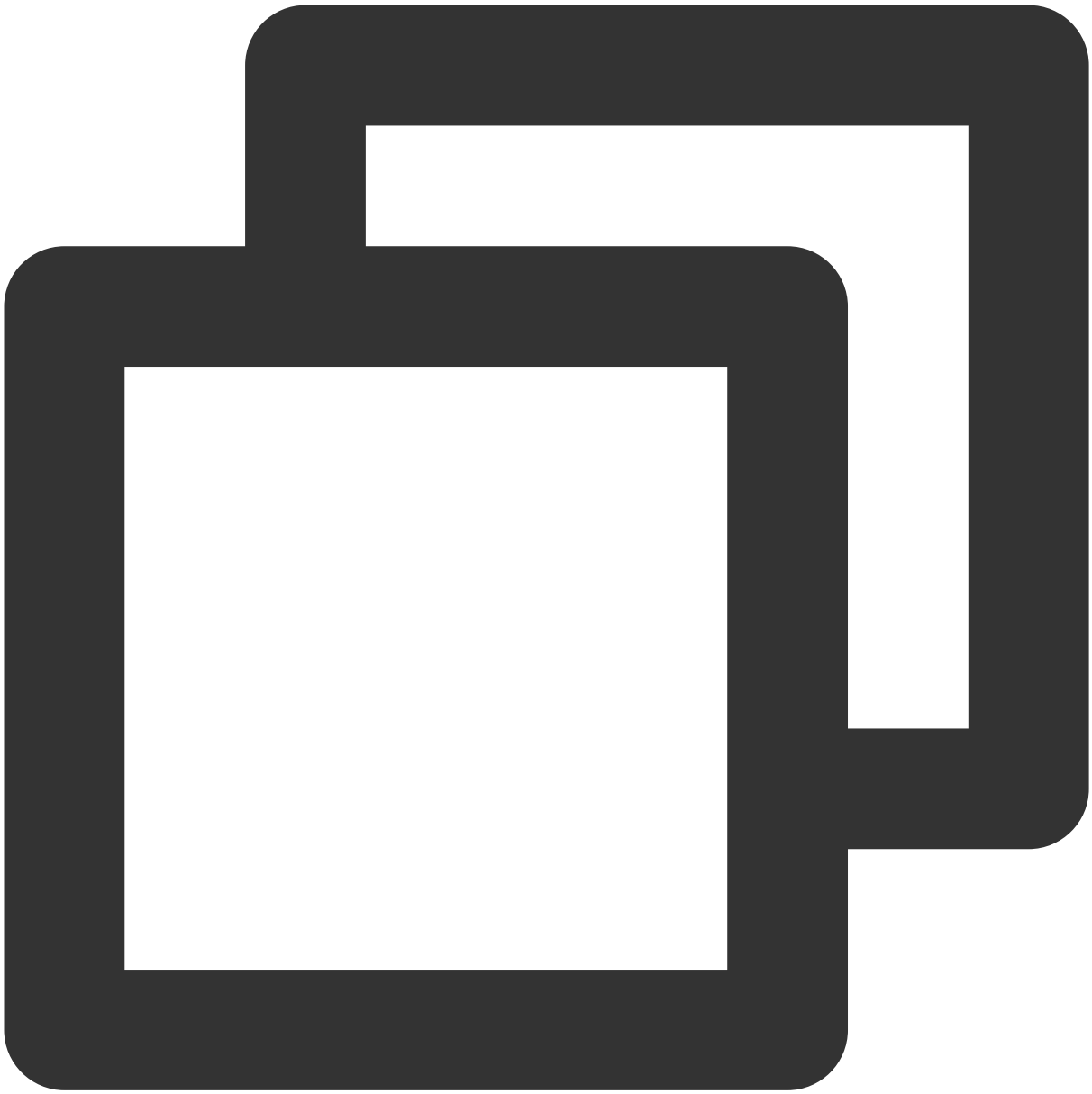
```
- (void)onCallMediaTypeChanged:(TUICallMediaType)oldCallMediaType newCallMediaType:
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
oldCallMediaType	TUICallMediaType	変更前の通話タイプ
newCallMediaType	TUICallMediaType	変更後の通話タイプ

onUserReject

通話が拒否された場合のコールバックです。1v1通話では、発呼側のみが拒否のコールバックを受信します。グループ通話では、すべての被招待者がこのコールバックを受信することができます。



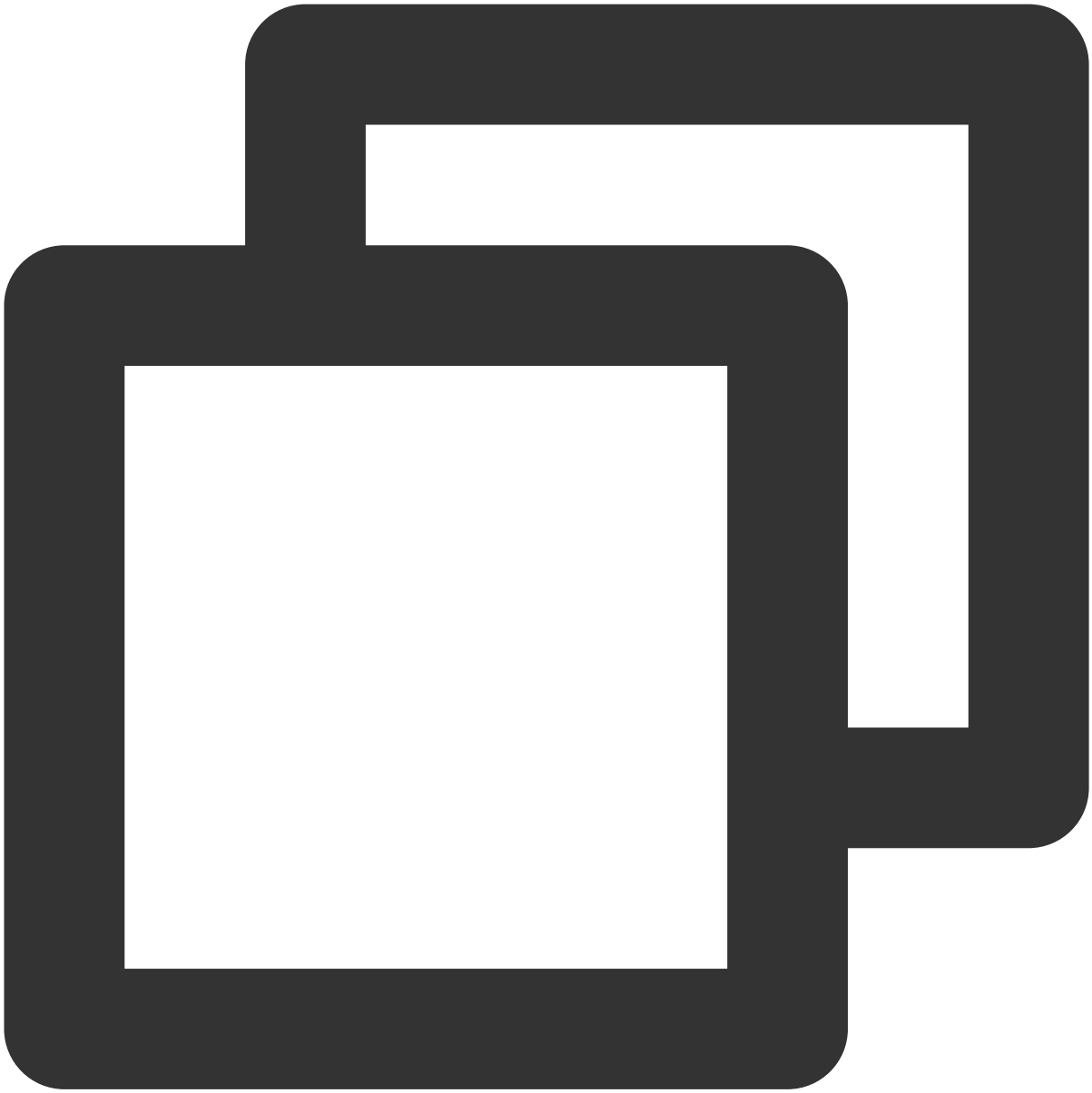
```
- (void)onUserReject:(NSString *)userId;
```

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	拒否したユーザーのID

onUserNoResponse

相手方が応答しない場合のコールバック。



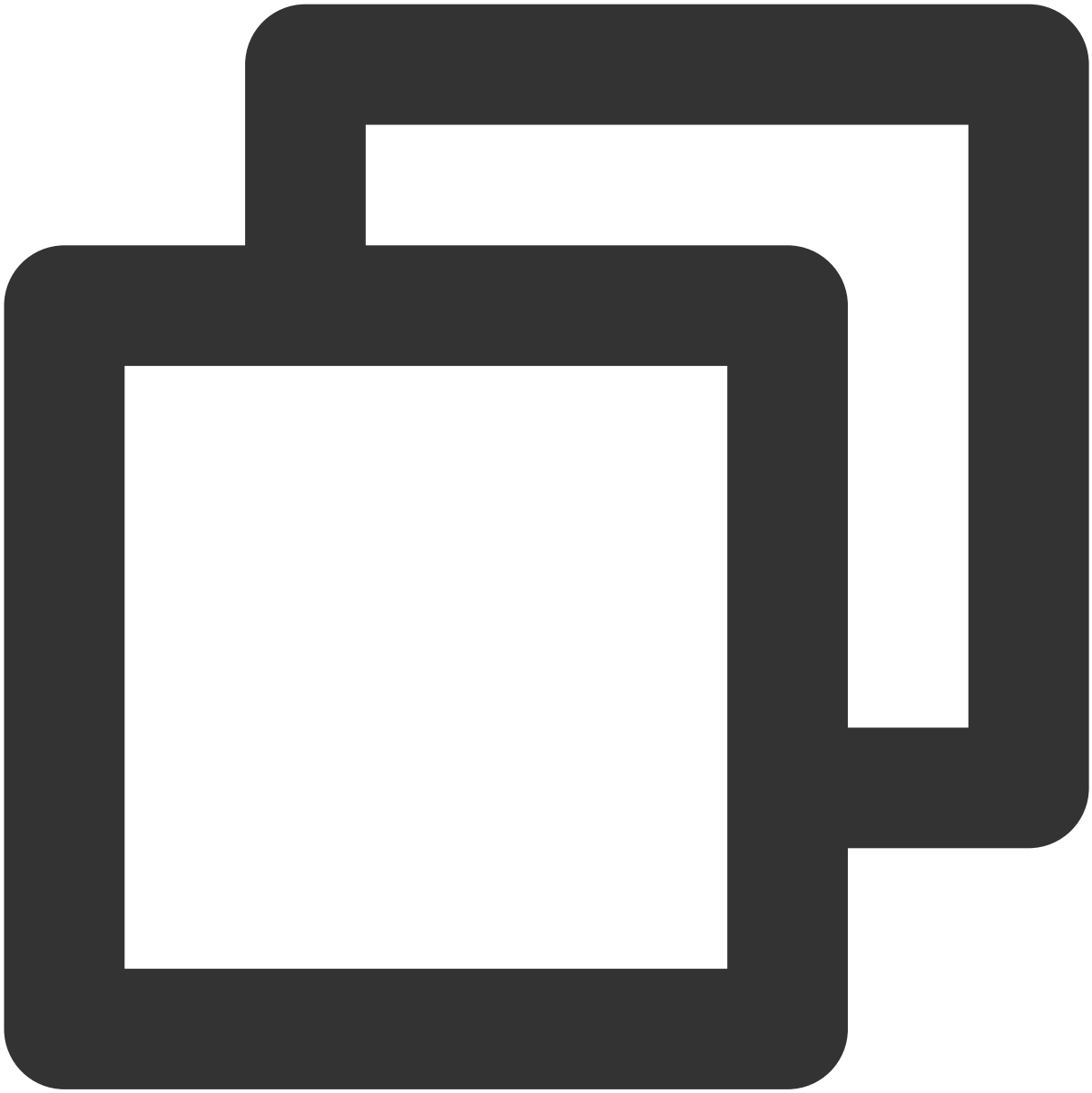
```
- (void)onUserNoResponse:(NSString *)userId;
```

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	応答しないユーザーのID

onUserLineBusy

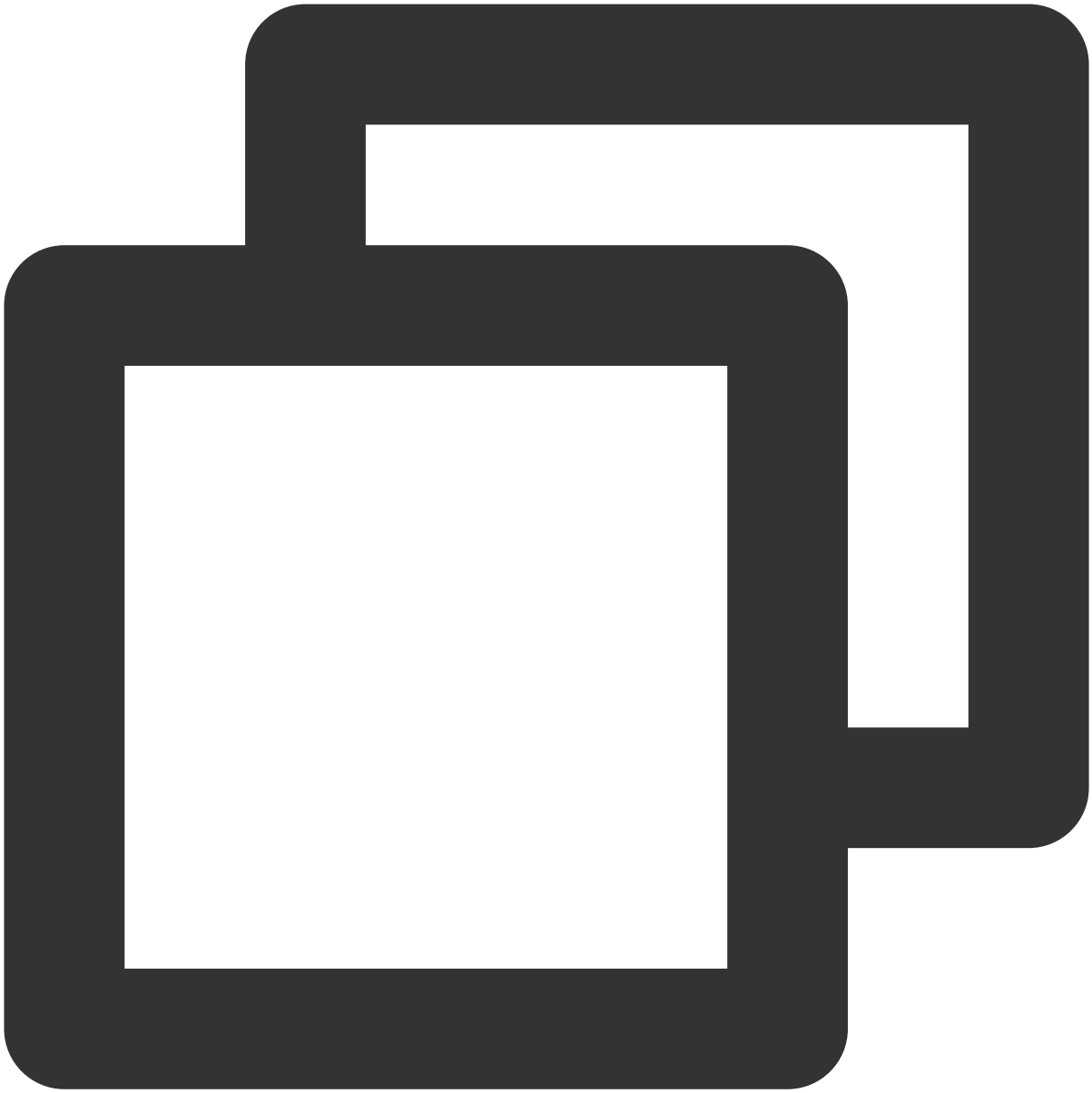
通話中である場合のコールバック。



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

onUserJoin

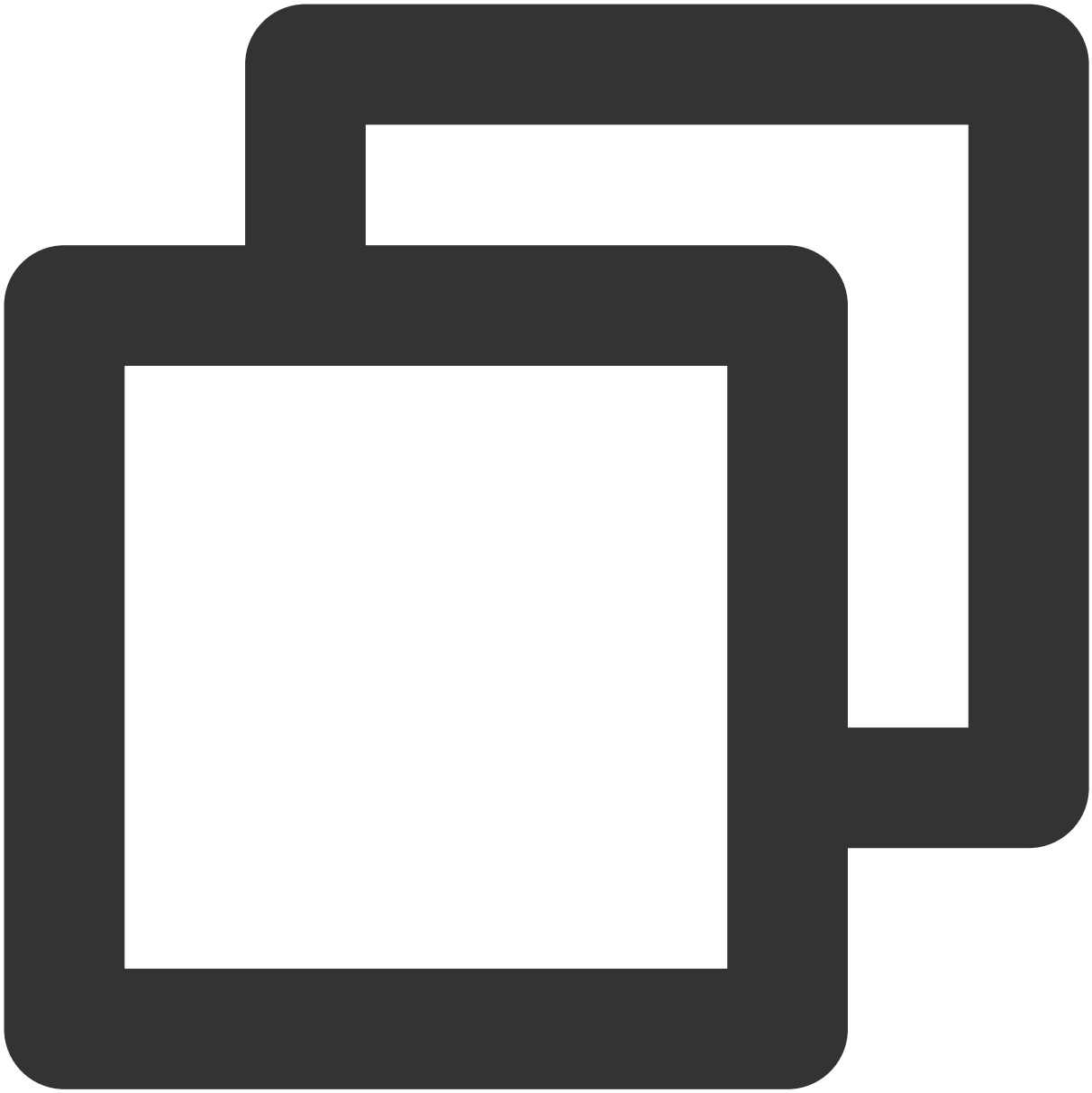
今回の通話に参加したユーザーがいる場合のコールバック。



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

onUserLeave

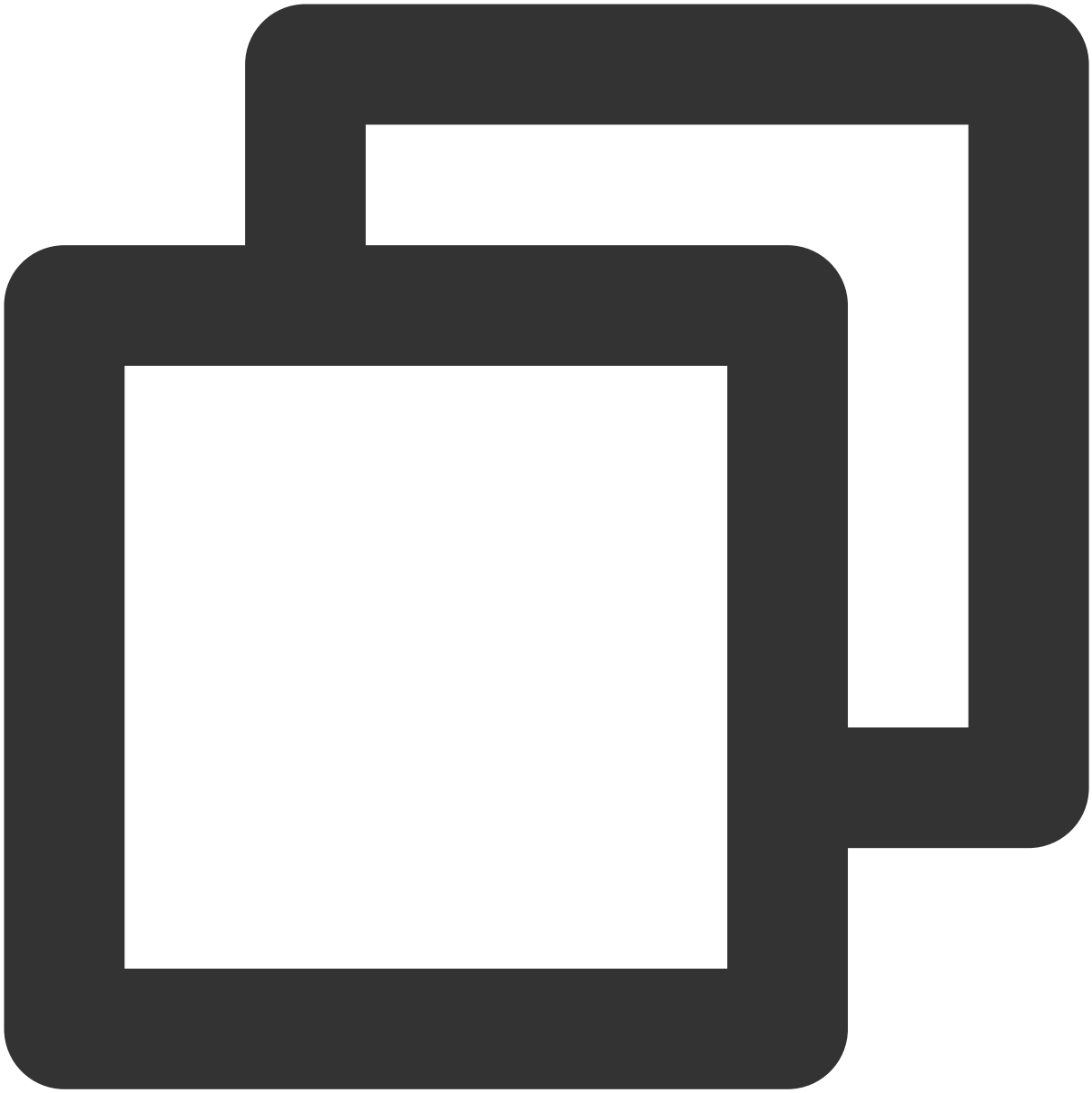
今回の通話から退出したユーザーがいる場合のコールバック。



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

onUserAudioAvailable

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。



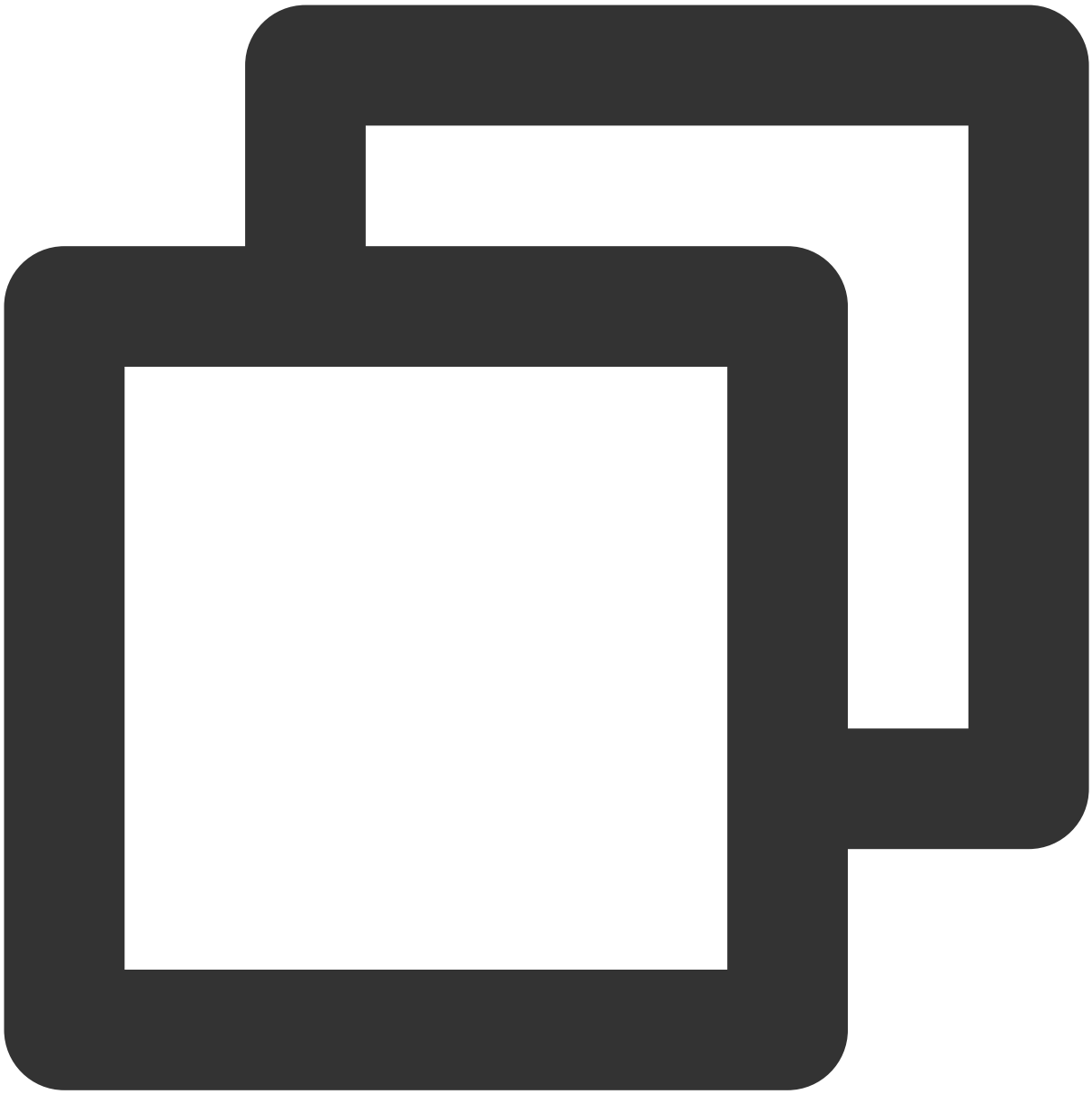
```
- (void)onUserAudioAvailable:(NSString *)userId isAudioAvailable:(BOOL)isAudioAvail
```

パラ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
isAudioAvailable	BOOL	ユーザーのオーディオが使用可能かどうか

onUserVideoAvailable

ユーザーがビデ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。



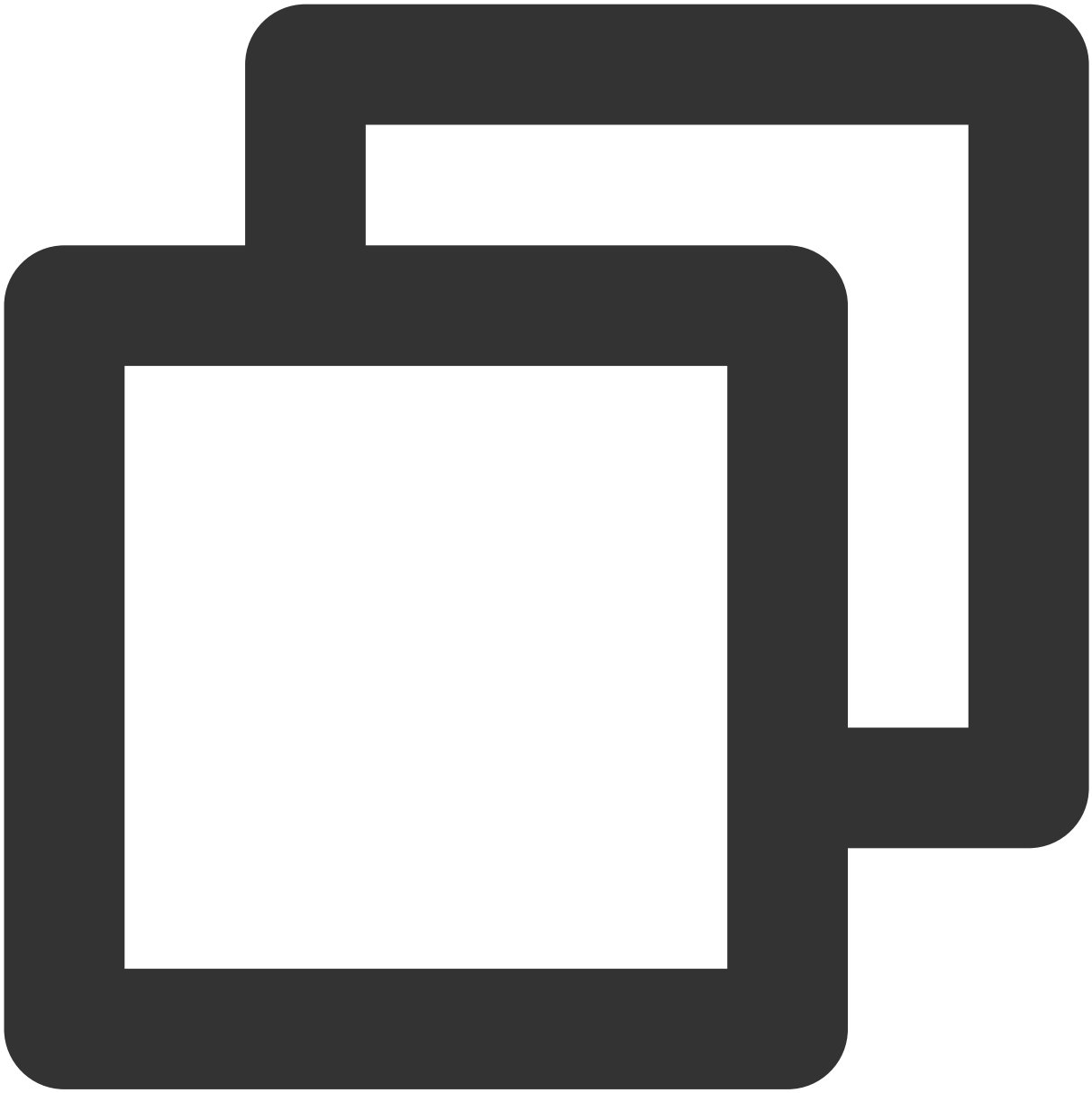
```
- (void)onUserVideoAvailable:(NSString *)userId isVideoAvailable:(BOOL)isVideoAvail
```

パラ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	通話ユーザーID
isVideoAvailable	BOOL	ユーザーのビデオが使用可能かどうか

onUserVoiceVolumeChanged

ユーザーの通話音量のコールバック。



```
- (void)onUserVoiceVolumeChanged:(NSDictionary <NSString *, NSNumber *> *)volumeMap
```

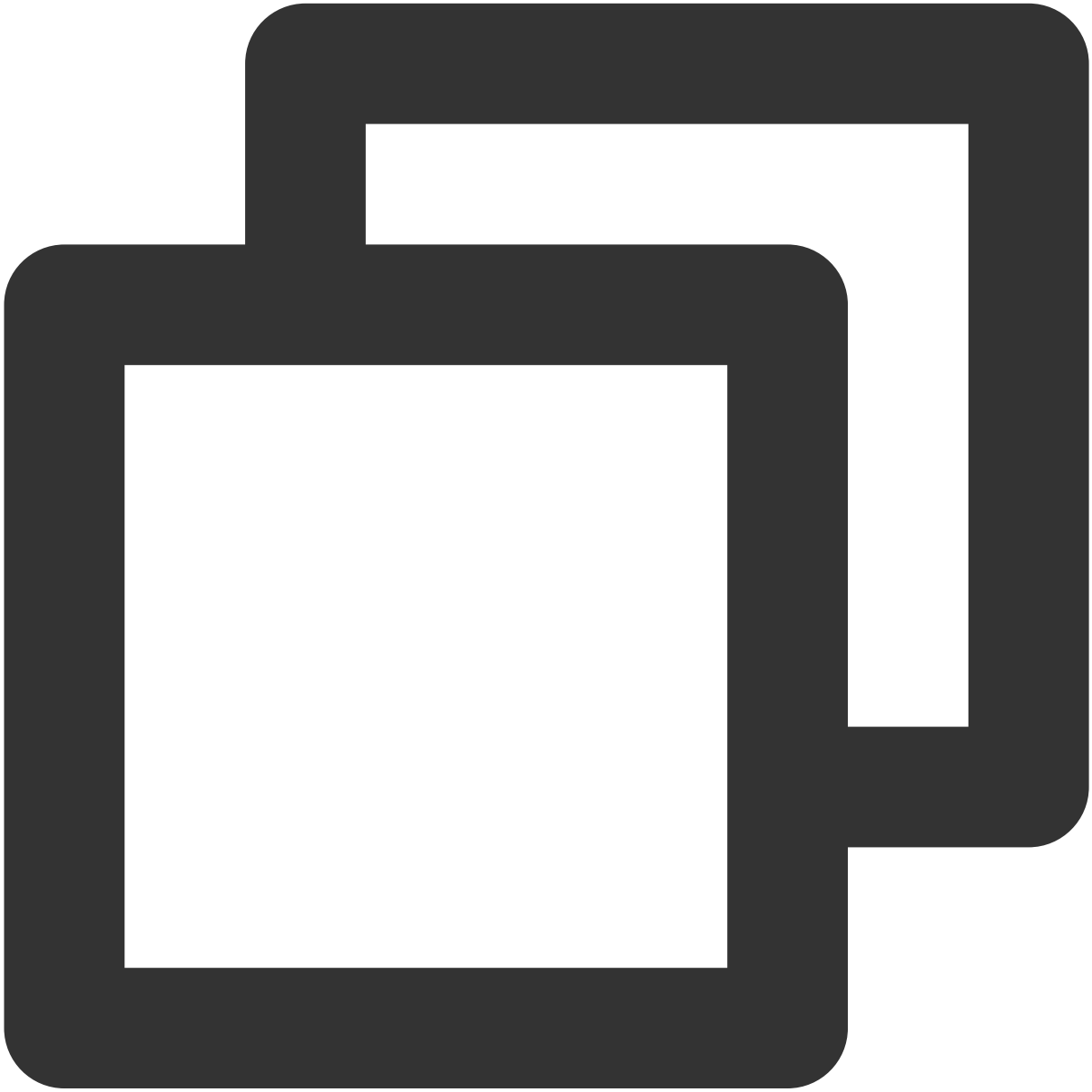
パラメータは下表に示すとおりです。

パラメータ	タイプ	意味
volumeMap	NSDictionary <NSString *, NSNumber *>	ボリュームメーター。各useridに応じて、対応する音量レベルを取得できます。音量の最小値は0、最大値は100です

volumeMap	NSDictionary	ボリュームメーター。各userIdに応じて、対応する音量レベルを取得できます。音量の最小値は0、最大値は100です

onUserNetworkQualityChanged

ユーザーのネットワーク品質のコールバック。



```
- (void)onUserNetworkQualityChanged:(NSArray<TUINetworkQualityInfo *> *)networkQual
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	意味

networkQualityList	NSArray	ネットワーク状態。各userIdにつき、対応するユーザーの現在のネットワーク品質を取得できます

Web
API概要

最終更新日：：2024-07-19 14:53:21

TUICallEngine (UIインターフェースなし)

TUICallEngine APIはオーディオビデオ通話コンポーネントの**UIインターフェースがない**ものです。このAPIを使用し、業務ニーズに応じてパッケージをカスタマイズすることができます。

API	説明
createInstance	TUICallEngineインスタンスの作成（シングルトンモード）
destroyInstance	TUICallEngineインスタンスの破棄（シングルトンモード）
on	イベントの監視
off	イベント監視のキャンセル
login	ログインインターフェース
logout	ログアウトインターフェース
setSelfInfo	ユーザーニックネームおよびプロフィール画像の設定
call	C2C通話への招待
groupCall	グループチャット通話への招待
accept	通話応答
reject	通話拒否
hangup	通話終了
switchCallingType	オーディオビデオ通話の切り替え
startRemoteView	リモート画面レンダリングの起動
stopRemoteView	リモート画面レンダリングの停止
startLocalView	ローカル画面レンダリングの起動
stopLocalView	ローカル画面レンダリングの停止

openCamera	カメラの起動
closeCamara	カメラの終了
openMicrophone	マイクをオンにする
closeMicrophone	マイクをオフにする
setMicMute	デバイスマイクのミュートの有無
setVideoQuality	ビデオ画質の設定
getDeviceList	デバイスリストの取得
switchDevice	カメラまたはマイクデバイスの切り替え

イベントタイプの定義

TUICallEventはTUICallEngineに対応するコールバックイベントクラスです。このコールバックによって、関心のあるコールバックイベントを監視することができます。

EVENT	説明
TUICallEvent.ERROR	SDKの内部でエラーが発生しました
TUICallEvent.SDK_READY	SDKがready状態に入ったときにこのコールバックを受信します
TUICallEvent.KICKED_OUT	重複ログインです。このコールバックを受信した場合は、ルームからの強制退出を意味します
TUICallEvent.USER_ACCEPT	応答したユーザーがいる場合に、このコールバックを受信します
TUICallEvent.USER_ENTER	通話への参加に同意したユーザーがいる場合に、このコールバックを受信します
TUICallEvent.USER_LEAVE	通話からの退出に同意したユーザーがいる場合に、このコールバックを受信します
TUICallEvent.REJECT	ユーザーが通話を拒否
TUICallEvent.NO_RESP	招待したユーザーからの応答なし
TUICallEvent.LINE_BUSY	招待者が通話中
TUICallEvent.CALLING_TIMEOUT	被招待者が受信します。このコールバックを受信

	した場合は、今回の通話に応答せずタイムアウトしたことを意味します
TUICallEvent.USER_VIDEO_AVAILABLE	リモートユーザーによるカメラのオン/オフがあった場合に、このコールバックを受信します
TUICallEvent.USER_AUDIO_AVAILABLE	リモートユーザーによるマイクのオン/オフがあった場合に、このコールバックを受信します
TUICallEvent.USER_VOICE_VOLUME	リモートユーザーがスピーカーの音量調整を行った場合に、このコールバックを受信します
TUICallEvent.GROUP_CALL_INVITEE_LIST_UPDATE	グループチャットの招待リストが更新された場合にこのコールバックを受信します
TUICallEvent.INVITED	通話に招待されました
TUICallEvent.CALLING_CANCEL	被招待者が受信します。このコールバックを受信した場合は、今回の通話がキャンセルされたことを意味します
TUICallEvent.CALLING_END	このコールバックを受信した場合は、今回の通話が終了したことを意味します
TUICallEvent.DEVICED_UPDATED	デバイスリストが更新された場合にこのコールバックを受信します
TUICallEvent.CALL_TYPE_CHANGED	通話タイプが切り替わった場合にこのコールバックを受信します

TUICallKit

最終更新日：：2024-07-19 14:53:21

APIの概要

TUICallKit APIはオーディオビデオ通話コンポーネントの**UIインターフェース付き**のものです。TUICallKit APIを使用することで、WeChatのようなオーディオビデオ通話シーンをシンプルなインターフェースでスピーディーに実現できます。より詳細なアクセス手順については、TUICallKitクイックアクセスをご参照ください。

APIの概要

`<TUICallKit/>` ：UI通話コンポーネントを主体としています
`<TUICallKitMini/>` ：UI通話フローティングウィンドウ
で、 `<TUICallKit/>allowedMinimized` が `true` に設定されている場合、 `<TUICallKitMini/>` はページ内に配置する必要があります
`TUICallKitServer` ：通話インスタンス、メンバーの関数です。
init TUICallKitの初期化
call 1v1通話の開始
groupCall グループ通話の開始
destroyed TUICallKitの破棄

`<TUICallKit/>` APIの詳細

属性

パラメータ	説明	タイプ	入力必須かどうか	デフォルト値
allowedMinimized	最小化を許可するかどうか。最小化ボタンは非表示	boolean	いいえ	false
allowedFullScreen	フルスクリーンを許可するかどうか。フルスクリーンボタンは非表示	boolean	いいえ	true

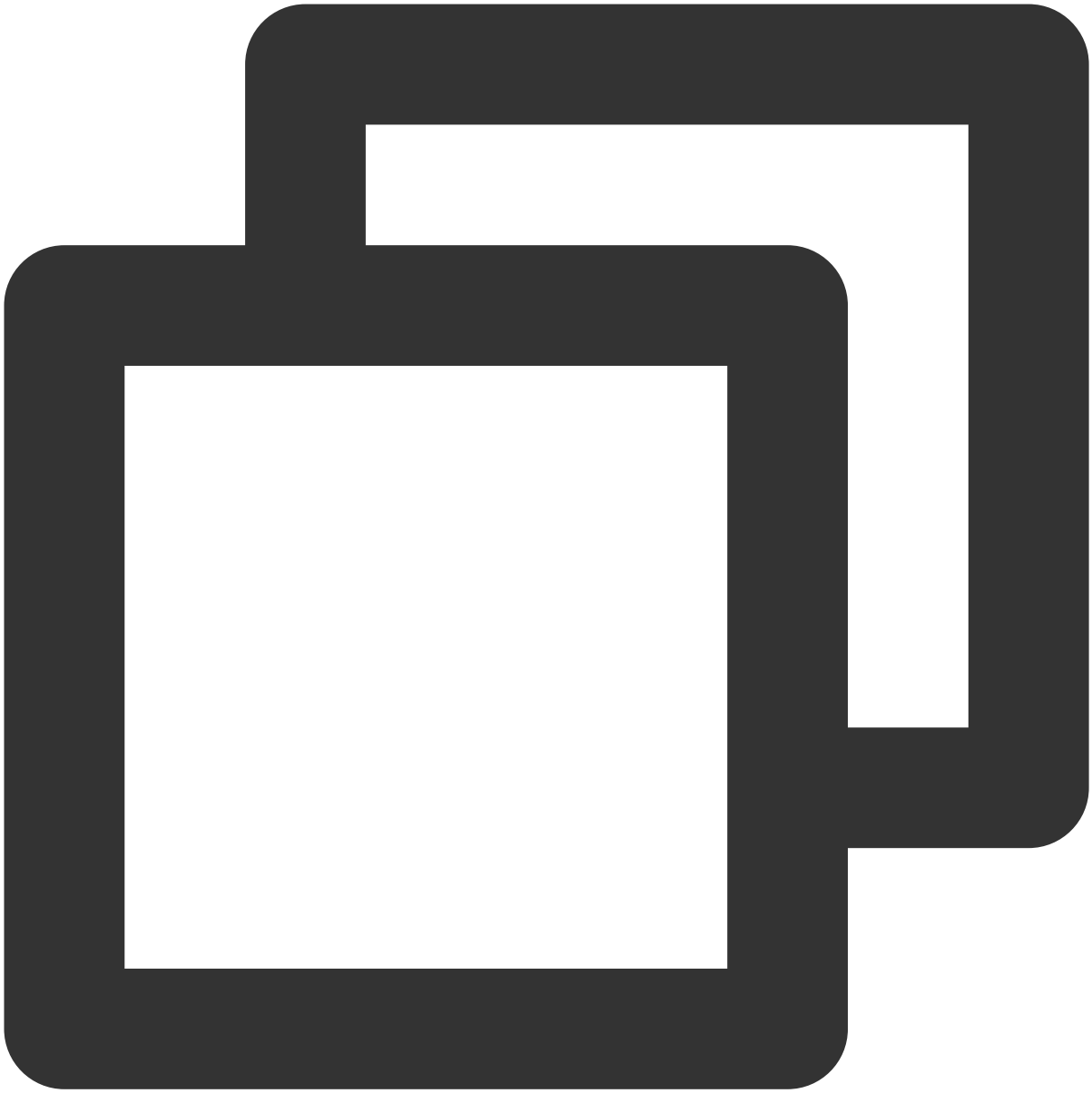
方法

パラメータ	説明	タイプ	入力必須かどうか	デフォルト値
beforeCalling	電話をかける前と通話の招待を受信する前にこの関数を実行	function(type、error)	いいえ	-
afterCalling	通話終了後にこの関数を実行	function()	いいえ	-
onMinimized	コンポーネントを最小化状態に切り替えた時にこの関数を実行	function(oldStatus、newStatus)	いいえ	-

`<TUICallKitMini/>` APIの詳細

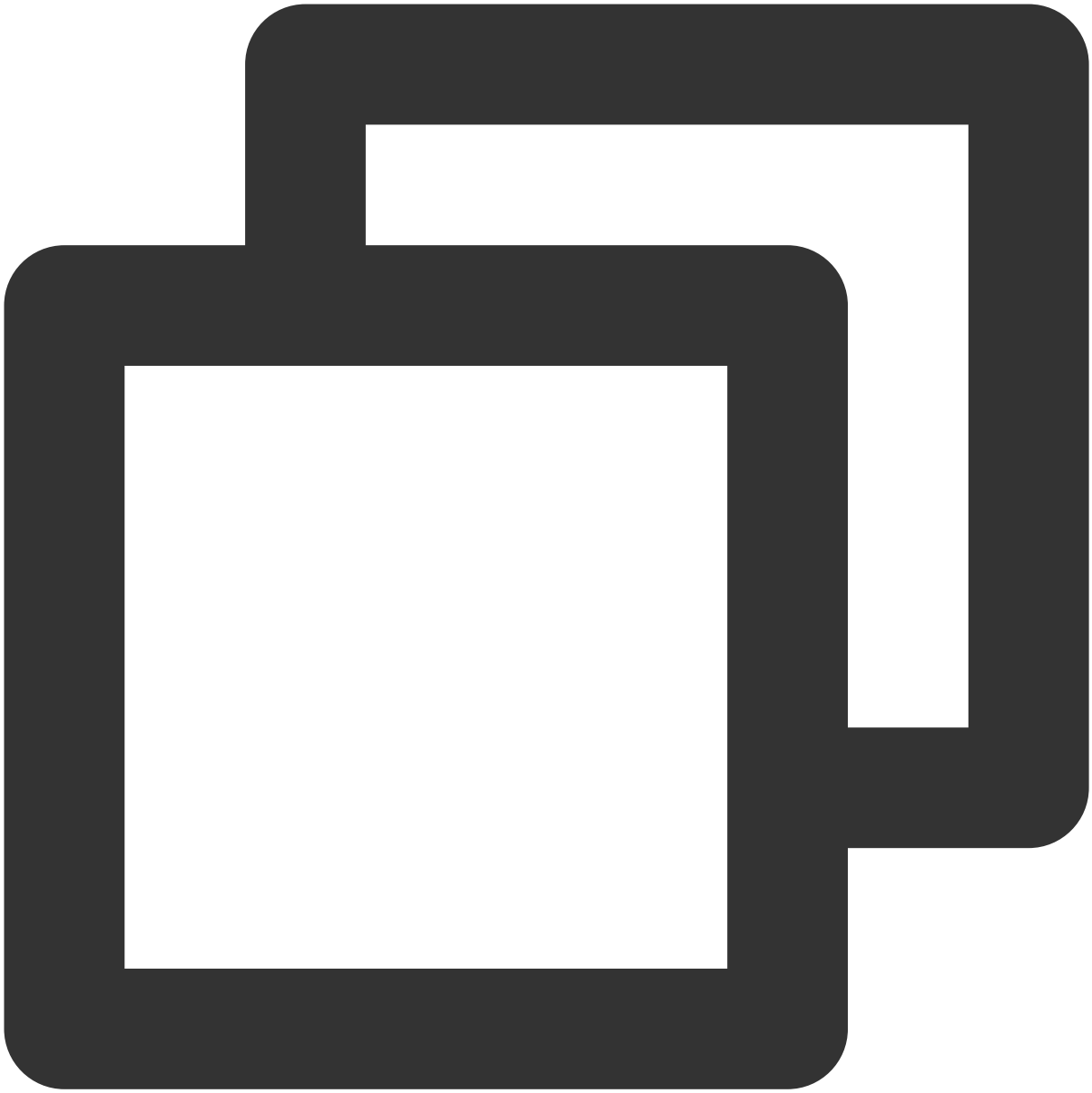
なし

サンプルコード



```
/**
* beforeCalling
 * @param { string } type値は"invited"です | "call" | "groupCall"は、 着信と発信の区別
 * @param { number } error.code エラーコード
 * @param { string } error.type エラータイプ
 * @param { string } error.code エラー情報
 */
function beforeCalling(type, error) {
  console.log("通話前にこの関数を実行、タイプ: "、type、error);
}
function afterCalling() {
```

```
  console.log("通話後にこの関数を実行");
}
/**
 * onMinimized
 * @param { boolean } oldStatus
 * @param { boolean } newStatus
 */
function onMinimized(oldStatus, newStatus) {
  if (newStatus === true) {
    console.log("TUICallKit 最小化状態に進む");
  }else{
    console.log("TUICallKit 最小化状態を終了");
  }
}
```

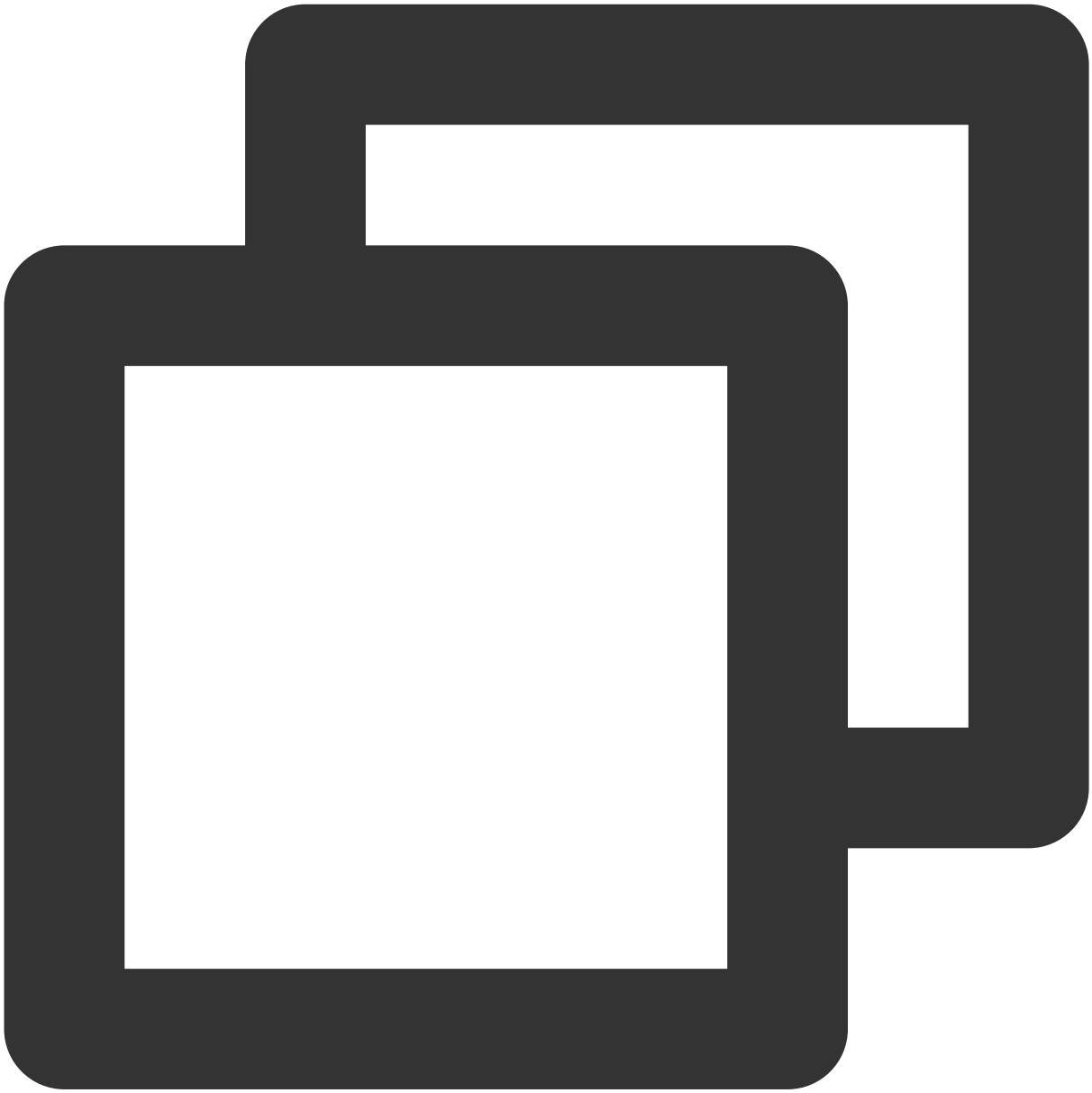


```
<TUICallKit
  :allowedMinimized="true"
  :allowedFullScreen="true"
  :beforeCalling="beforeCalling"
  :afterCalling="afterCalling"
  :onMinimized="onMinimized"
/>
<TUICallKitMini />
```

TUICallKitServer APIの詳細

init

TUICallKitの初期化はcall、groupCallの前に行う必要があります。



```
import { TUICallKitServer } from "./components/TUICallKit/Web";
TUICallKitServer.init({
  SDKAppID,
  userID,
  userSig,
```

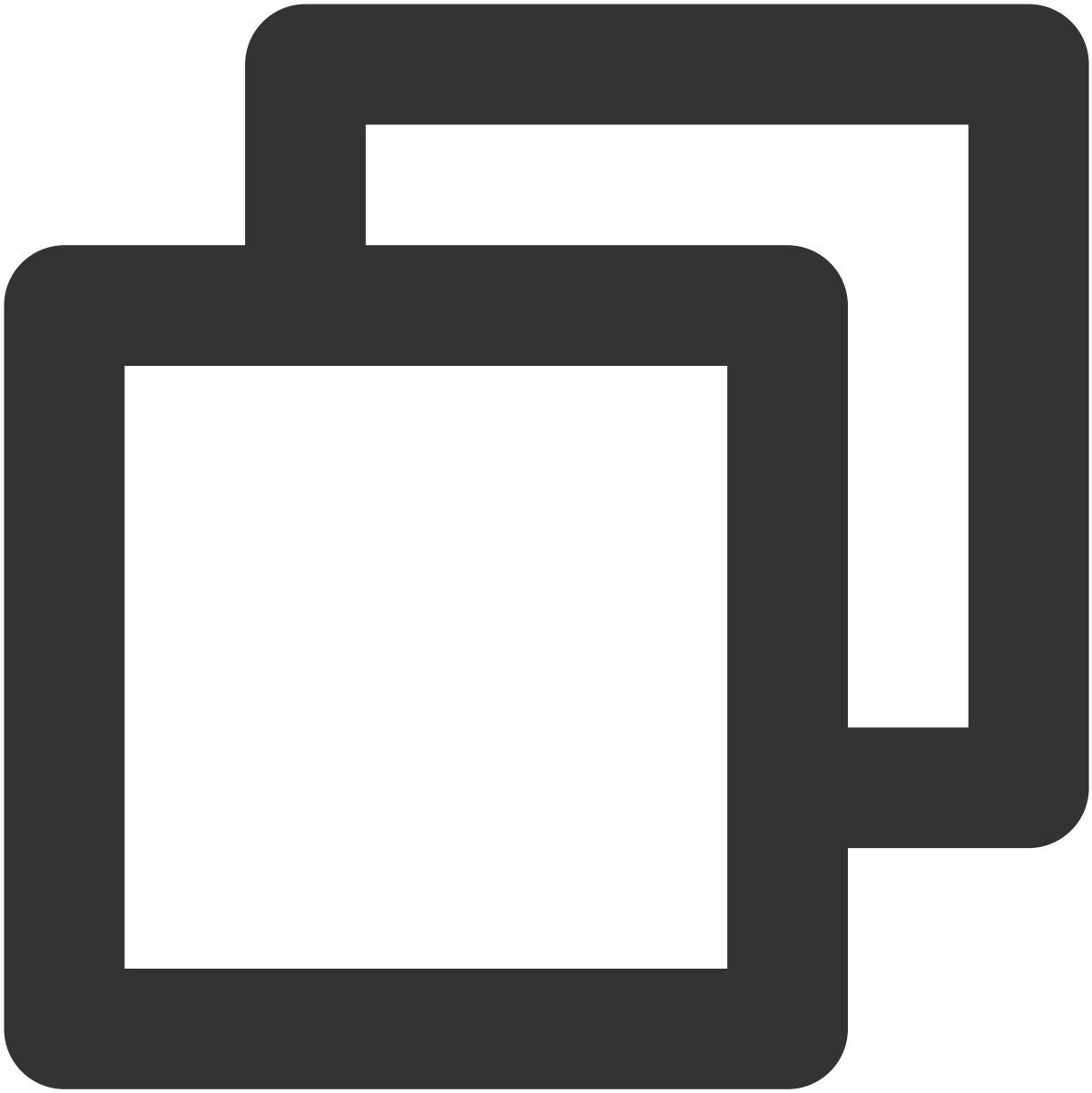
```
  tim,
});
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	入力必須かどうか	意味
SDKAppID	Number	はい	IMアプリケーションのSDKAppID
userID	String	はい	現在のユーザーID、文字列タイプでは、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	はい	Tencent Cloudによって設計されたセキュリティ保護署名です。取得方法については、UserSigをご参照ください
TIMインスタンス	Any	いいえ	timパラメータはサービス内にすでに存在するTIMインスタンスに適用され、TIMインスタンスの一意性を保証します

call

電話をかけます（1v1通話）。



```
import { TUICallKitServer } from "./components/TUICallKit/Web";
TUICallKitServer.call({
  userID: 'jack',
  type: 1,
});
```

パラメータは下表に示すとおりです。

パラメータ	タイプ	入力必須かどうか	意味

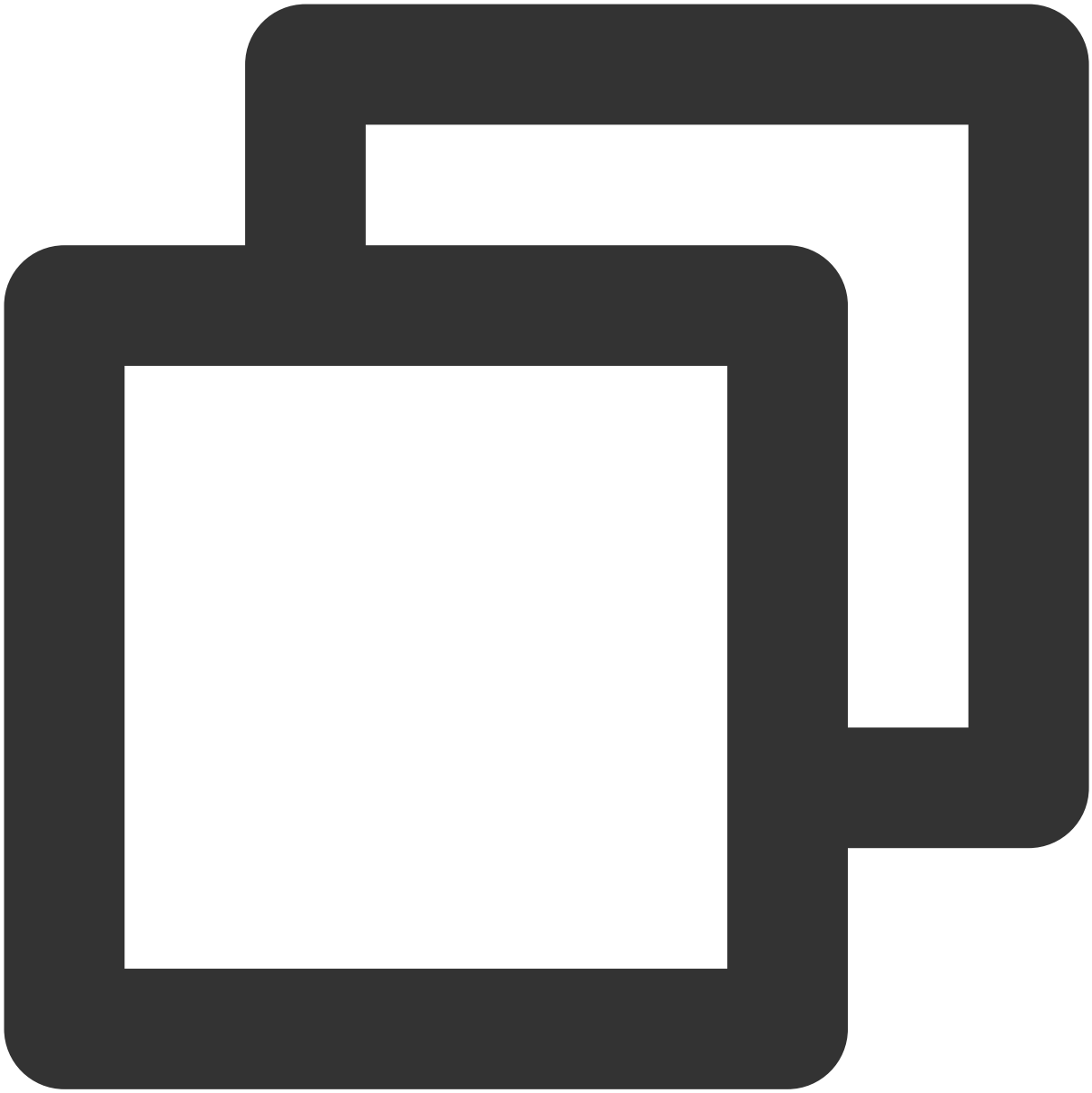
userID	String	はい	ターゲットユーザーのuserId
type	Number	はい	通話のメディアタイプ。音声通話(type = 1)、ビデオ通話(type = 2)
timeout	Number	いいえ	通話のタイムアウト時間。0はタイムアウトしていないことを意味し、単位はs（秒）（オプション） - デフォルト 30s
offlinePushInfo	Object	いいえ	メッセージのオフラインプッシュをカスタマイズします（オプション）-- tsignaling バージョンは >= 0.8.0が必要です

そのうち `offlinePushInfo` について

パラメータ	タイプ	入力必須かどうか	意味
offlinePushInfo.title	String	いいえ	タイトルのオフラインプッシュ（オプション）
offlinePushInfo.description	String	いいえ	コンテンツのオフラインプッシュ（オプション）
offlinePushInfo.androidOPPOChannelID	String	いいえ	オフラインプッシュでは、OPPO携帯（8.0およびそれ以降のシステム）のチャンネルIDを設定します（オプション）
offlinePushInfo.extension	String	いいえ	オフラインプッシュによるコンテンツのパススルー（オプション）（tsignalingバージョン >= 0.9.0）

groupCall

グループ通話を開始します。



```
import { TUICallKitServer } from "./components/TUICallKit/Web";
TUICallKitServer.groupCall({
  userIDList: ['jack', 'tom'],
  groupID: 'xxx',
  type: 1,
});
```

パラメータは下表に示すとおりです。

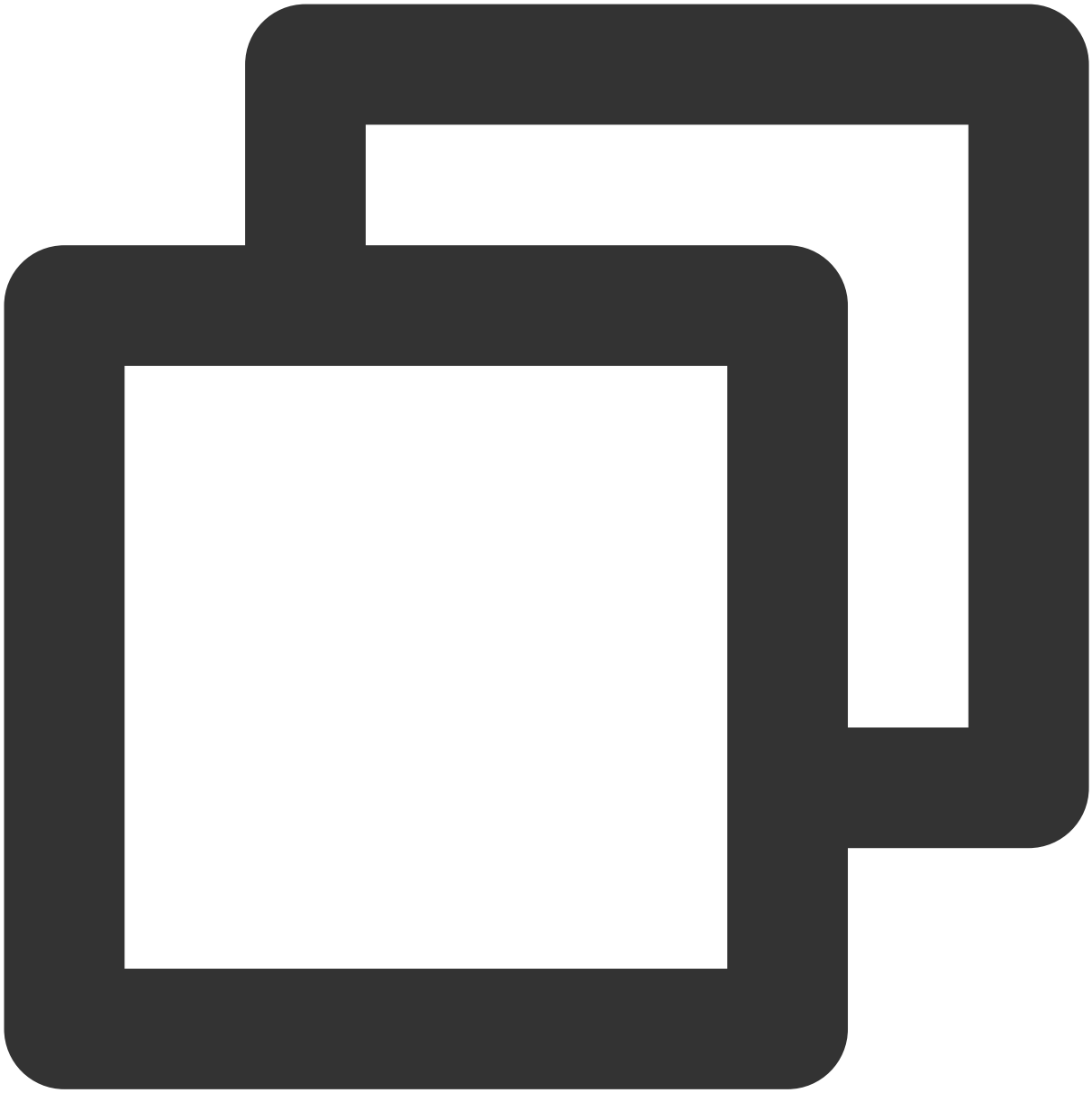
パラメータ	タイプ	入力必須かどうか	意味

userIDList	Array	はい	招待リストメンバーリスト
type	Number	はい	通話のメディアタイプ。音声通話(type = 1)、ビデオ通話(type = 2)
groupID	String	はい	グループIDの呼び出し
timeout	Number	いいえ	通話のタイムアウト時間。0はタイムアウトしていないことを意味し、単位はs（秒）（オプション） - デフォルト 30s
offlinePushInfo	Object	いいえ	メッセージのオフラインプッシュをカスタマイズします（オプション）-- tsignaling バージョンは >= 0.8.0が必要です

そのうち `offlinePushInfo` は、 `call` インターフェースと一致します。

destroyed

TUICallKitを破棄します。



```
import { TUICallKitServer } from "./components/TUICallKit/Web";
TUICallKitServer.destroyed();
```

TUICallEngine

最終更新日：：2024-07-19 14:53:21

TUICallEngine APIの概要

TUICallEngine APIはオーディオビデオ通話コンポーネントの**UIインターフェースがない**ものです。

API概要

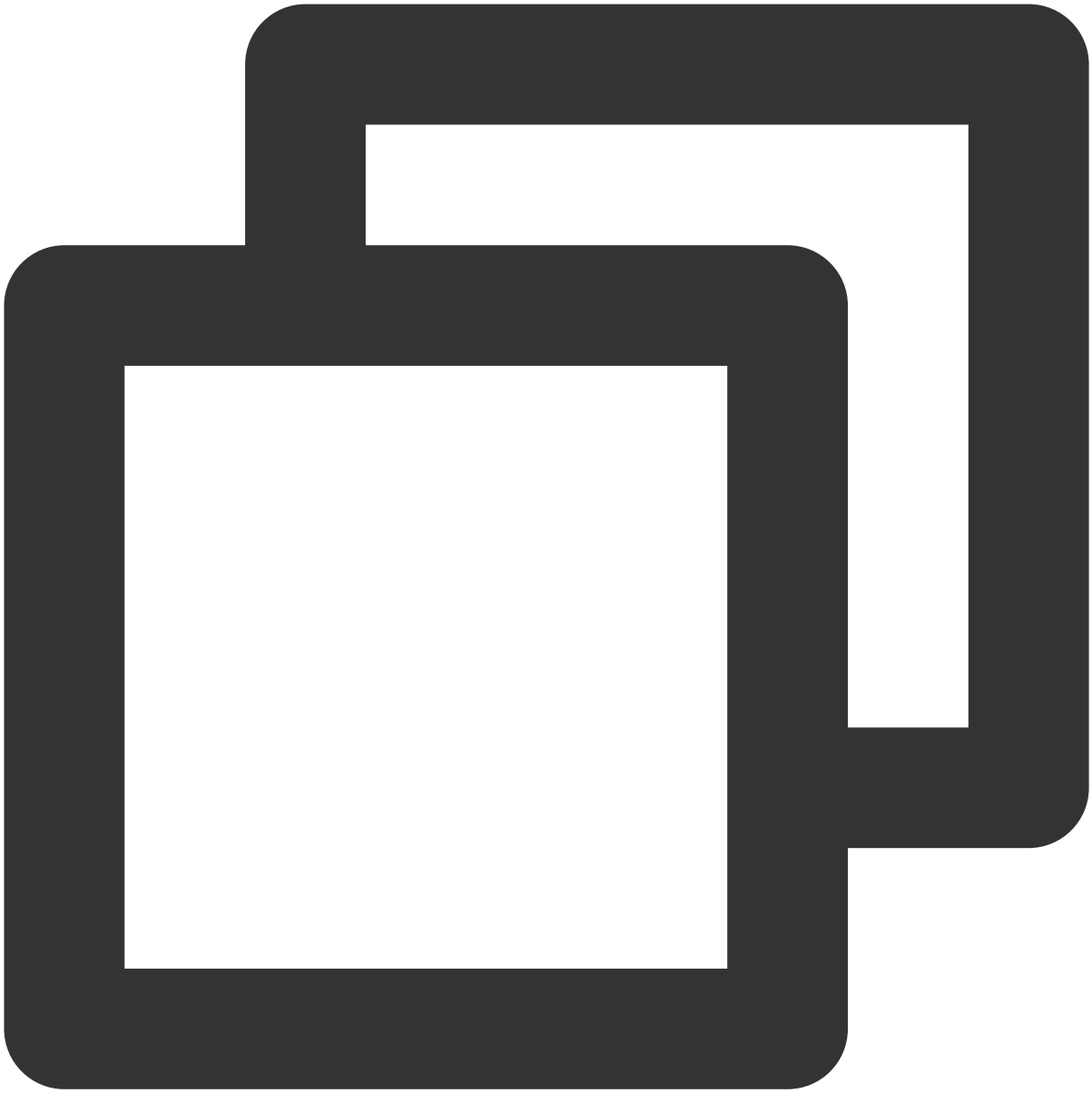
API	説明
createInstance	TUICallEngineのインスタンス（シングルトンモード）を作成します
destroyInstance	TUICallEngineのインスタンス（シングルトンモード）を破棄します
on	イベントの監視
off	イベントの監視をキャンセルします
login	インターフェースにログインします
logout	インターフェースからログアウトします
setSelfInfo	ユーザーニックネームおよびプロフィール画像を設定します
call	C2C通話に招待します
groupCall	グループチャット通話に招待します
accept	通話に応答します
reject	通話を拒否します
hangup	通話を終了します
inviteUser	グループ通話で、他の人を招待します
joinInGroupCall	現在のグループ通話に自主的に参加します
switchCallMediaType	現在の通話タイプを切り替えます

startRemoteView	リモート画面のレンダリングを起動します
stopRemoteView	リモート画面のレンダリングを停止します
startLocalView	ローカル画面のレンダリングを起動します
stopLocalView	ローカル画面のレンダリングを停止します
openCamera	カメラをオンにします
closeCamara	カメラをオフにします
openMicrophone	マイクをオンにします
closeMicrophone	マイクをオフにします
setVideoQuality	ビデオ画質を設定します
getDeviceList	デバイスリストを取得します
switchDevice	カメラまたはマイクデバイスを切り替えます
enableAIVoice	AIノイズリダクションのオン/オフ

APIの詳細

createInstance

TUICallEngineのシングルトンを作成します。



```
const tuiCallEngine = TUICallEngine.createInstance({
    SDKAppID: 0, // アクセス時に、0をIMアプリケーションのSDKAppIDに置き換える必要があります
    tim: tim          // timパラメータはサービス内にすでに存在するTIMインスタンスに使用され、T
});
```

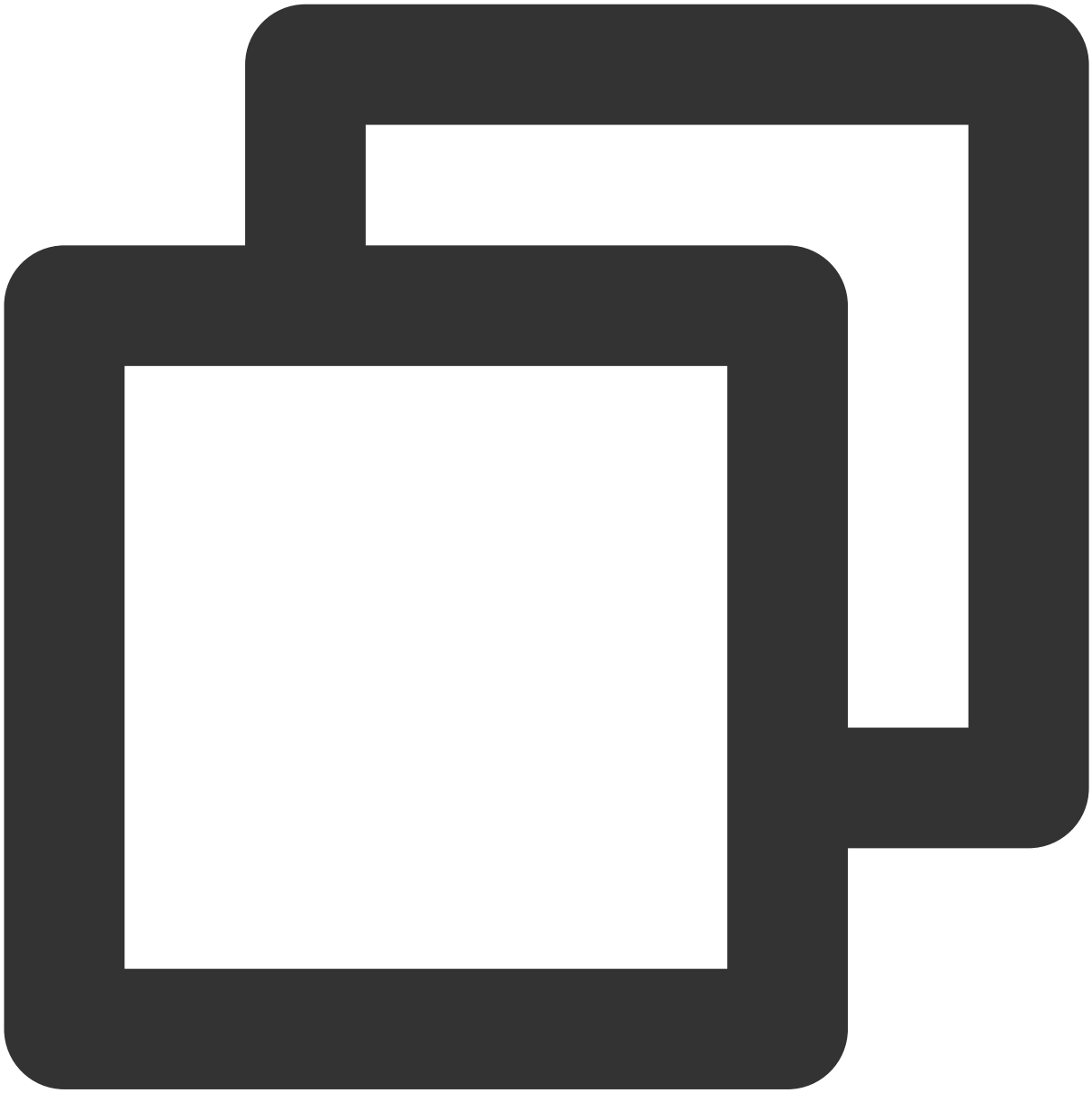
パラメータは下表に示すとおりです

パラメータ	タイプ	意味
SDKAppID	Number	Instant MessagingアプリケーションのSDKAppID

tim	Any	TIMインスタンス（オプション）

destroyInstance

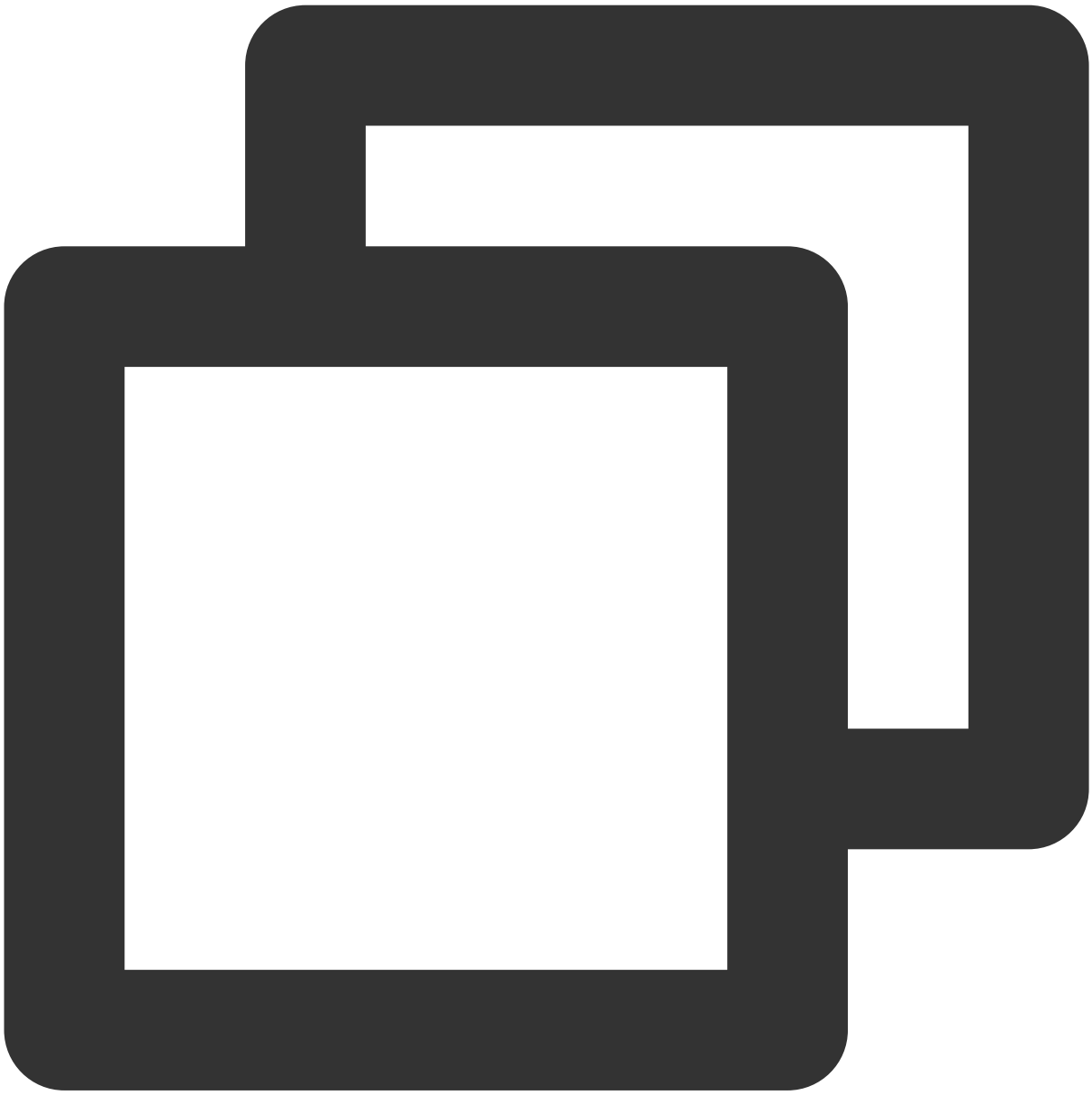
TUICallEngineのシングルトンを破棄します。



```
tuiCallEngine.destroyInstance().then(() => {
    //success
}).catch(error => {
    console.warn('destroyInstance error:', error);
});
```

on

イベントを監視します。



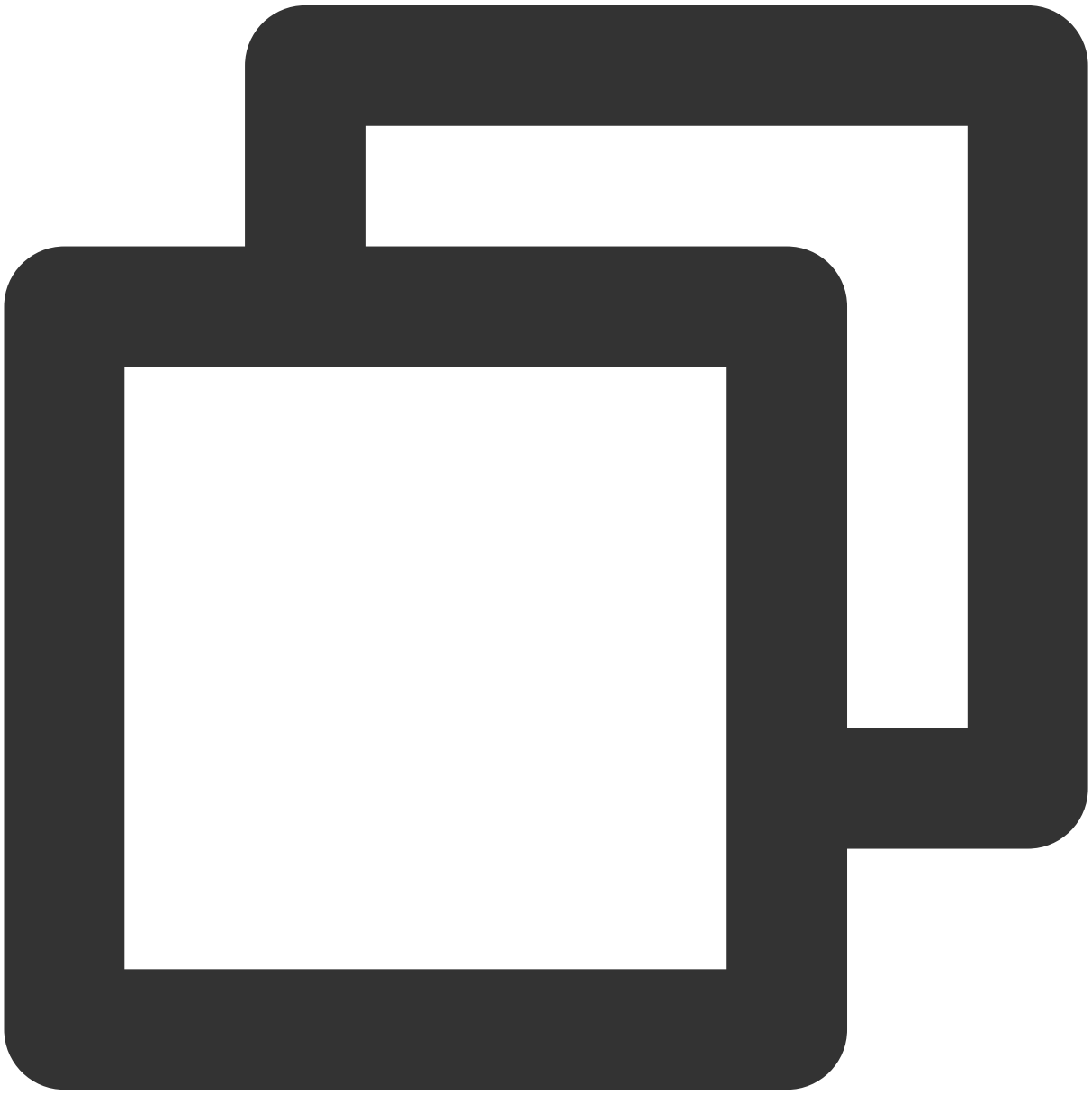
```
let onError = function(error) {
    console.log(error);
};
tuiCallEngine.on(TUICallEvent.ERROR, onError, this);
```

パラメータは下表に示すとおりです

パラメータ	タイプ	意味
eventName	String	イベント名
callback	function	イベントレスポンスコールバック
context	Any	callback実行希望時のコンテキスト

off

イベントの監視をキャンセルします。



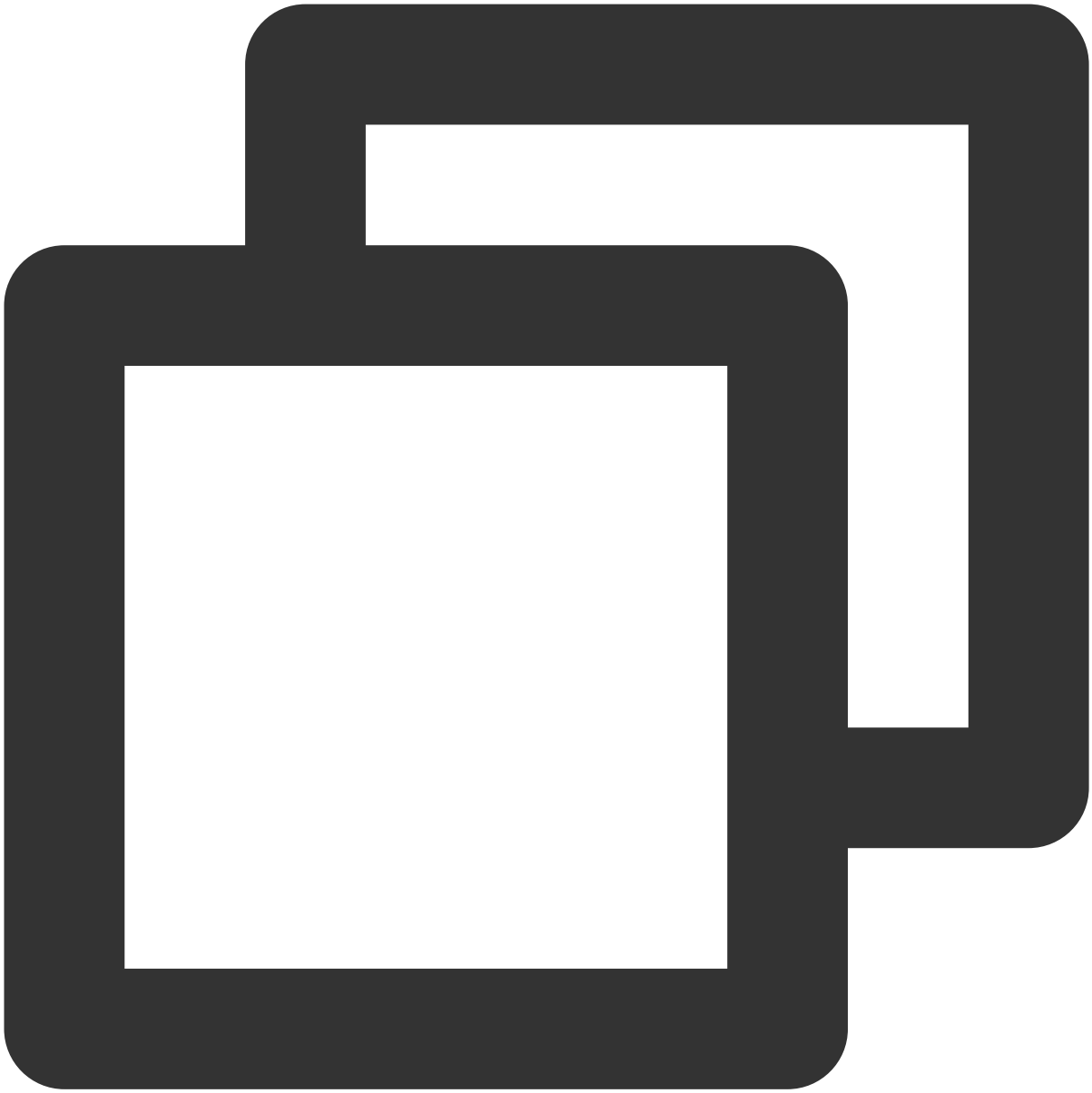
```
let onError = function(error) {
    console.log(error);
};
tuiCallEngine.off(TUICallEvent.ERROR, onError, this);
```

パラメータは下表に示すとおりです

パラメータ	タイプ	意味
eventName	String	イベント名
callback	function	イベントレスポンスコールバック
context	Any	callback実行希望時のコンテキスト

login

インターフェースにログインします。



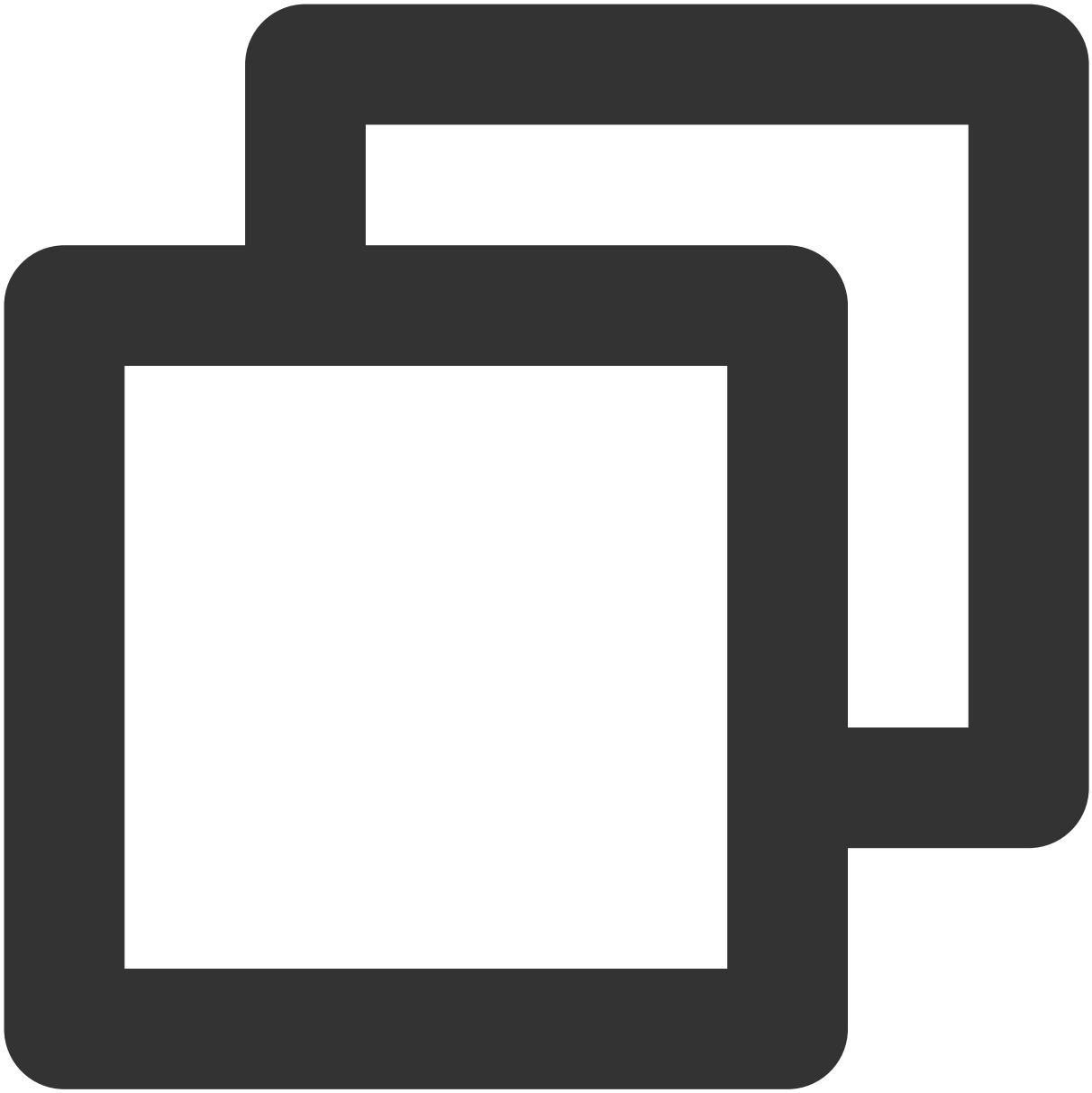
```
const params = {
    userID: 'john', // your userID
    userSig: 'xxxx', // 'your userSig'
    assetsPath: 'https://xx/'
};
let promise = tuiCallEngine.login(params);
promise.then(() => {
    //success
}).catch(error => {
    console.warn('login error:', error);
});
```

パラ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	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	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の計算方法をご参照ください。
assetsPath	String	AIノイズリダクションの依存関係denoiser-wasm.jsファイルがデプロイされるCDNまたは静的リソースサーバーへのパス。詳細については、AIノイズリダクションの使用をご参照ください。

logout

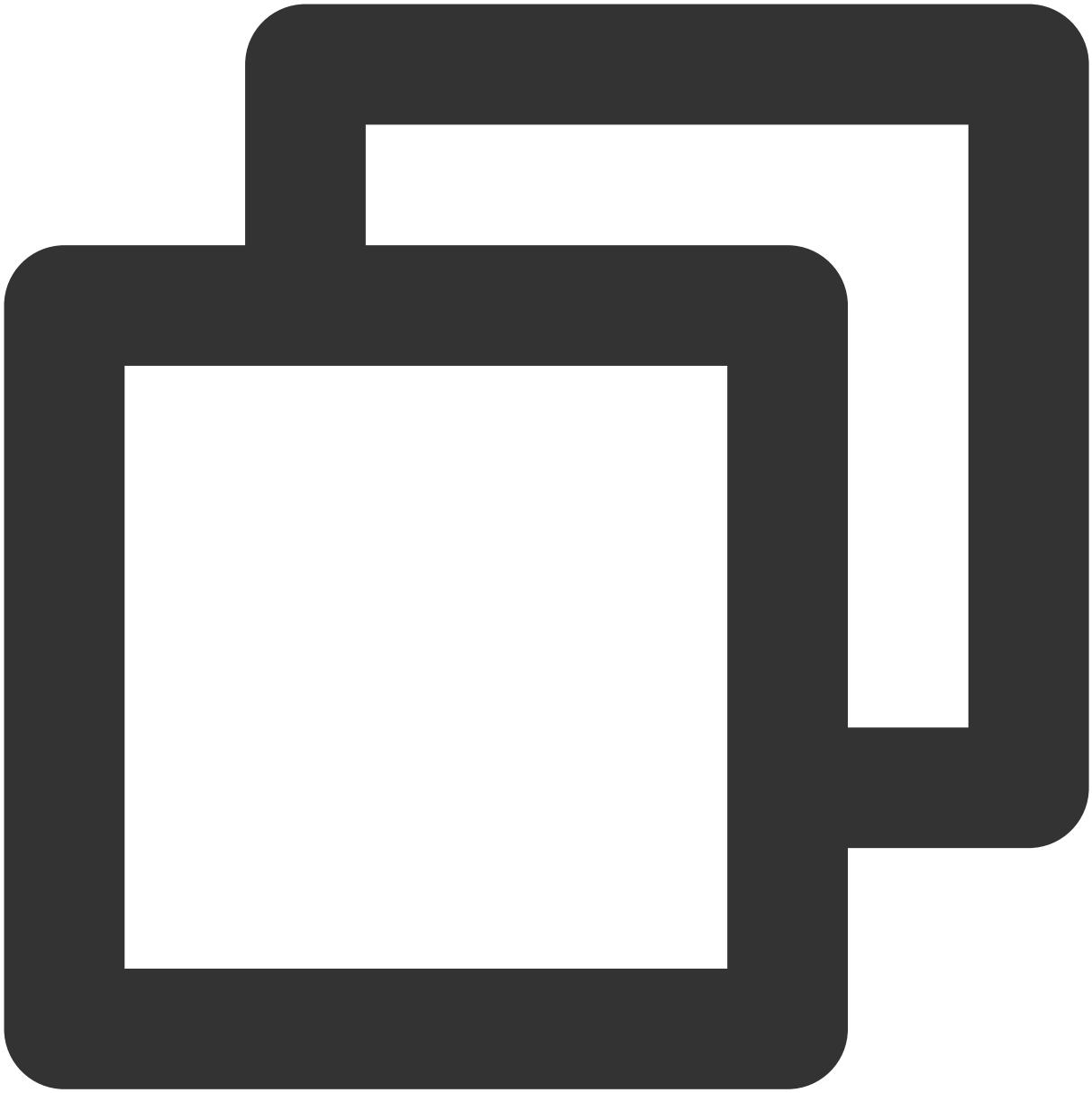
インターフェースからログアウトします。



```
let promise = tuiCallEngine.logout();
promise.then(() => {
    //success
}).catch(error => {
    console.warn('logout error:', error);
});
```

setSelfInfo

ユーザーニックネームおよびプロフィール画像を設定します。



```
let promise = tuiCallEngine.setSelfInfo({
    nickName: 'video',
    avatar:'http(s)://url/to/image.jpg'
});
promise.then(() => {
    //success
}).catch(error => {
    console.warn('setSelfInfo error:', error);
});
```

パラメータは下表に示すとおりです

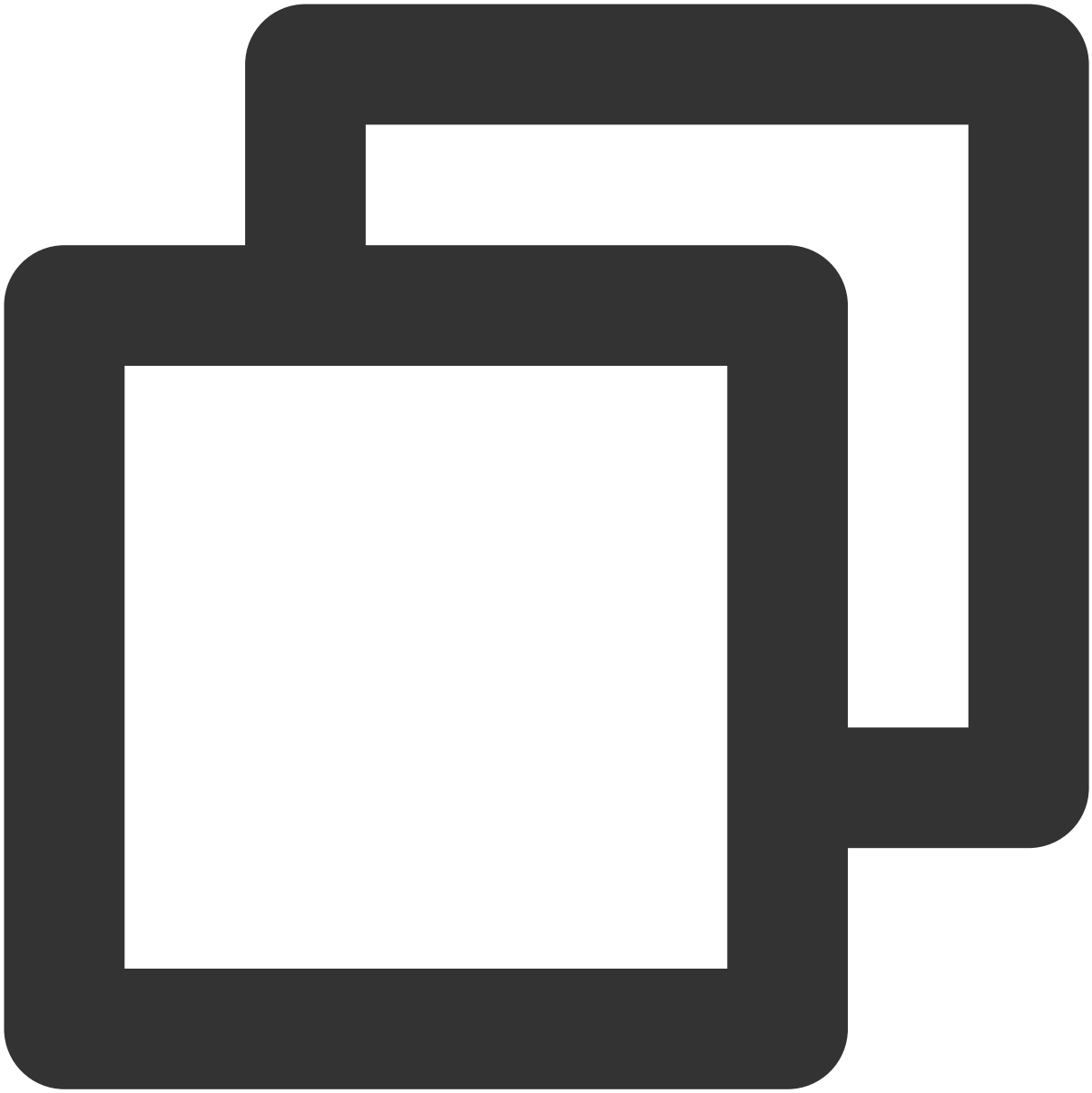
パラメータ	タイプ	意味
nickName	String	ニックネーム
avatar	String	プロフィール画像アドレス

call

C2Cの通話に招待します。被招待者はTUICallEvent.INVITEDイベントを受信します。

ご注意：

オフラインプッシュは端末（AndroidまたはiOS)にのみ適用され、WebおよびWeChat Mini Programではサポートされません。



```
let promise = tuiCallEngine.call({
    userID: 'user1',
    type: 1,
});
promise.then(() => {
    //success
}).catch(error => {
    console.warn('call error:', error);
});
```

パラメータは下表に示すとおりです

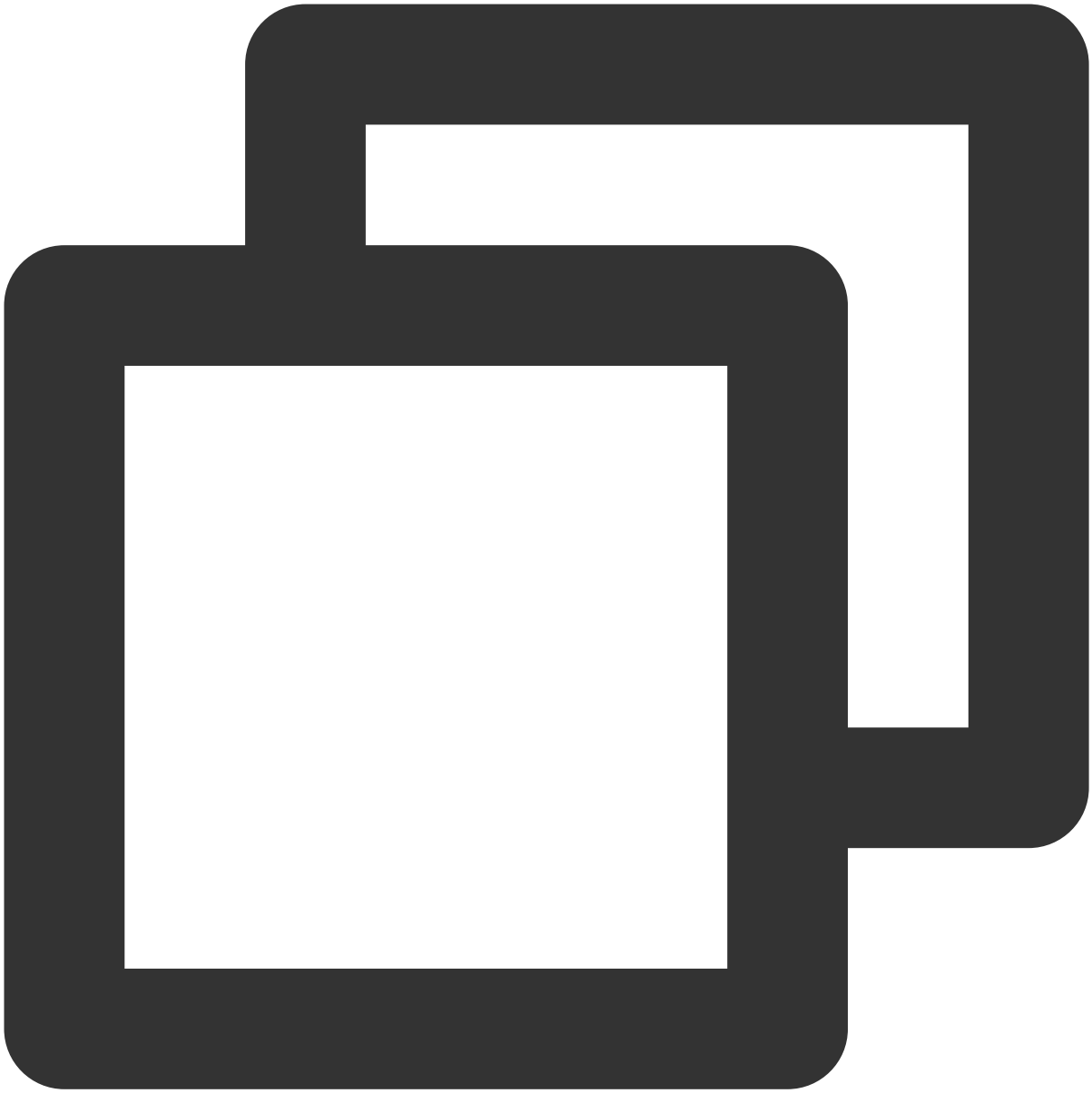
パラメータ	タイプ	意味
userID	String	招待された側のuserID
type	Number	0-不明、1-音声通話、2-ビデオ通話

groupCall

IMグループの通話に招待します。被招待者は `EVENT.INVITED` イベントを受信します。

ご注意：

オフラインプッシュは端末（AndroidまたはiOS)にのみ適用され、WebおよびWeChat Mini Programではサポートされません。



```
let promise = tuiCallEngine.groupCall({
    userIDList: ['user1', 'user2'],
    type: 1,
    groupID: 'IMグループID',
});
promise.then(() => {
    //success
}).catch(error => {
    console.warn('groupCall error:', error);
});
```

パラメータは下表に示すとおりです

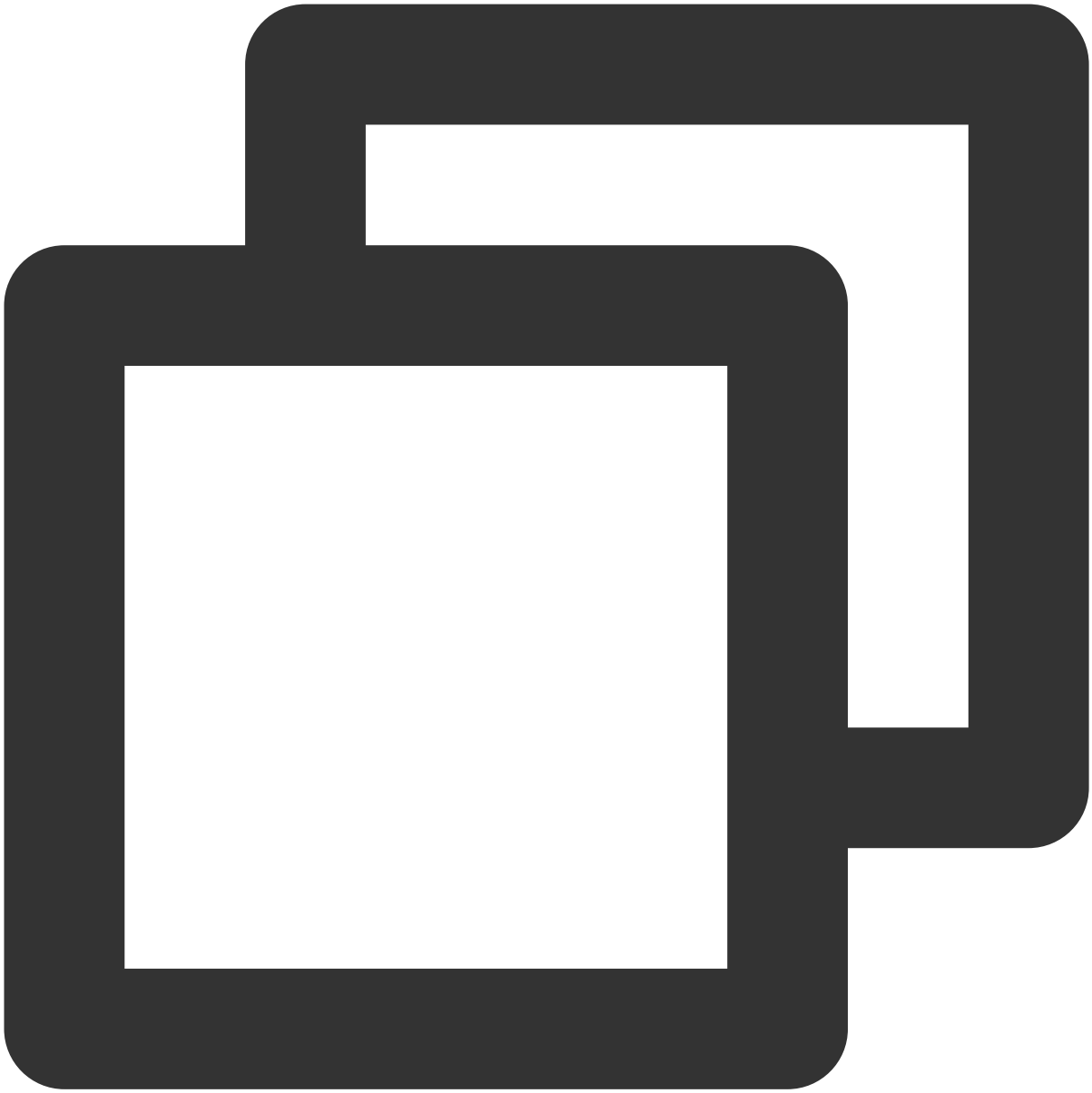
パラメータ	タイプ	意味
userIDList	Array	招待リスト
type	Number	0-不明、1-音声通話、2-ビデオ通話
groupID	String	IMグループ
timeout	String	タイムアウト時間（オプション）
roomID	String	ルームID（オプション）
offlinePushInfo	Object	カスタムオフラインメッセージプッシュ（オプション、tsignalingバージョン>=0.8.0が必要です）

offlinePushInfo

パラメータ	タイプ	意味
title	string	オフラインプッシュタイトル（オプション）
description	string	オフラインプッシュ内容（オプション）
androidOPPOChannelID	string	オフラインプッシュでOPPO携帯電話8.0システム以上のチャンネルIDを設定します（オプション）
extension	string	オフラインプッシュパススルー内容（オプション）（tsignalingバージョン>=0.9.0）

accept

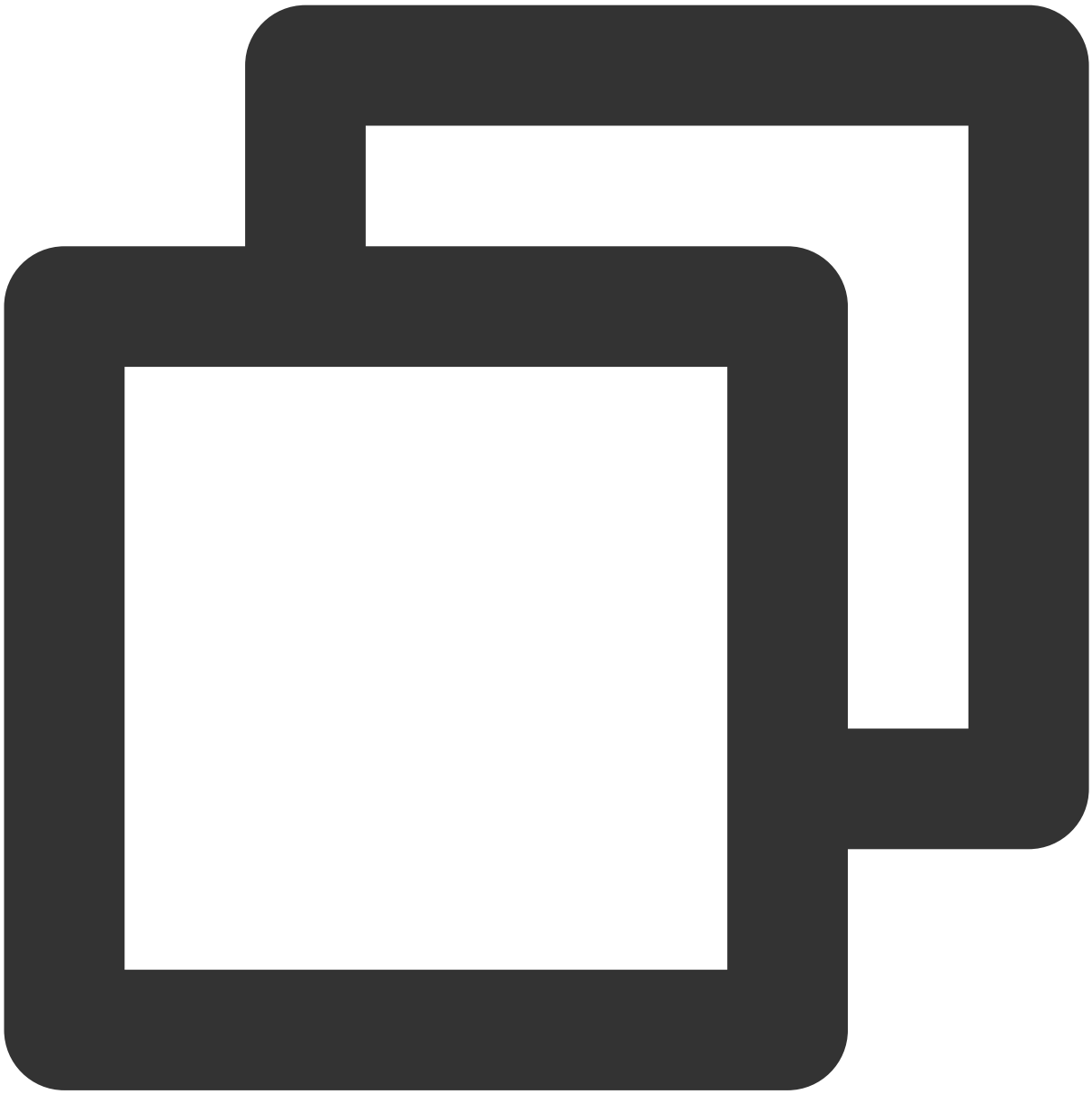
被招待者として `TUICallEvent.INVITED` イベントのコールバックを受信した場合は、このインターフェースを呼び出して通話に応答することができます。



```
tuiCallEngine.on(TUICallEvent.INVITED, () => {
    tuiCallEngine.accept().promise.then(() => {
        //success
    }).catch(error => {
        console.warn('accept error:', error);
    });
});
```

reject

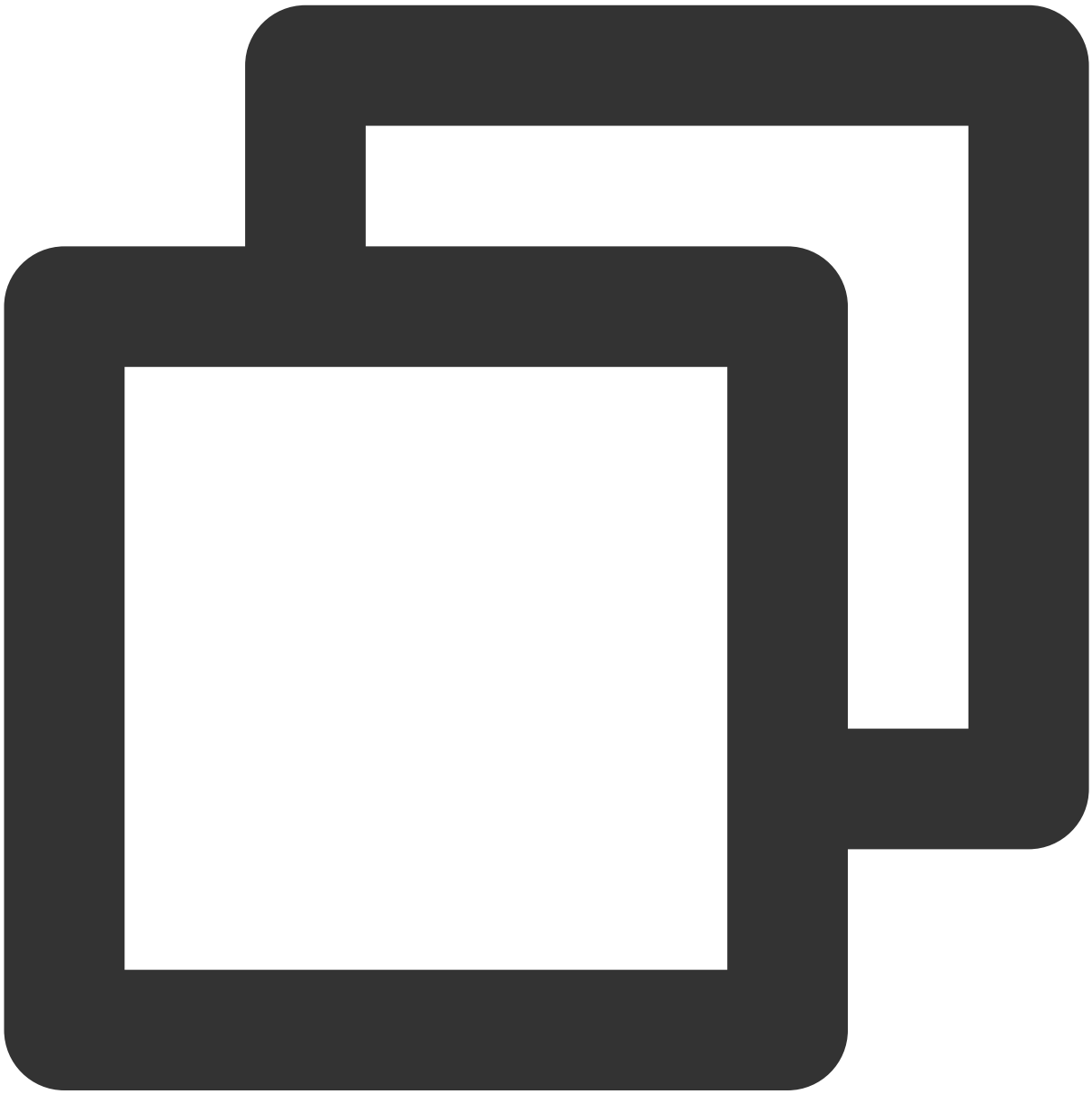
現在の通話を拒否します。着呼側として `TUICallEvent.INVITED` のコールバックを受信した場合は、この関数を呼び出して通話を拒否することができます。



```
tuiCallEngine.on(TUICallEvent.INVITED, () => {
    tuiCallEngine.reject().then(() => {
        //success
    }).catch(error => {
        console.warn('reject error:', error);
    });
});
```

hangup

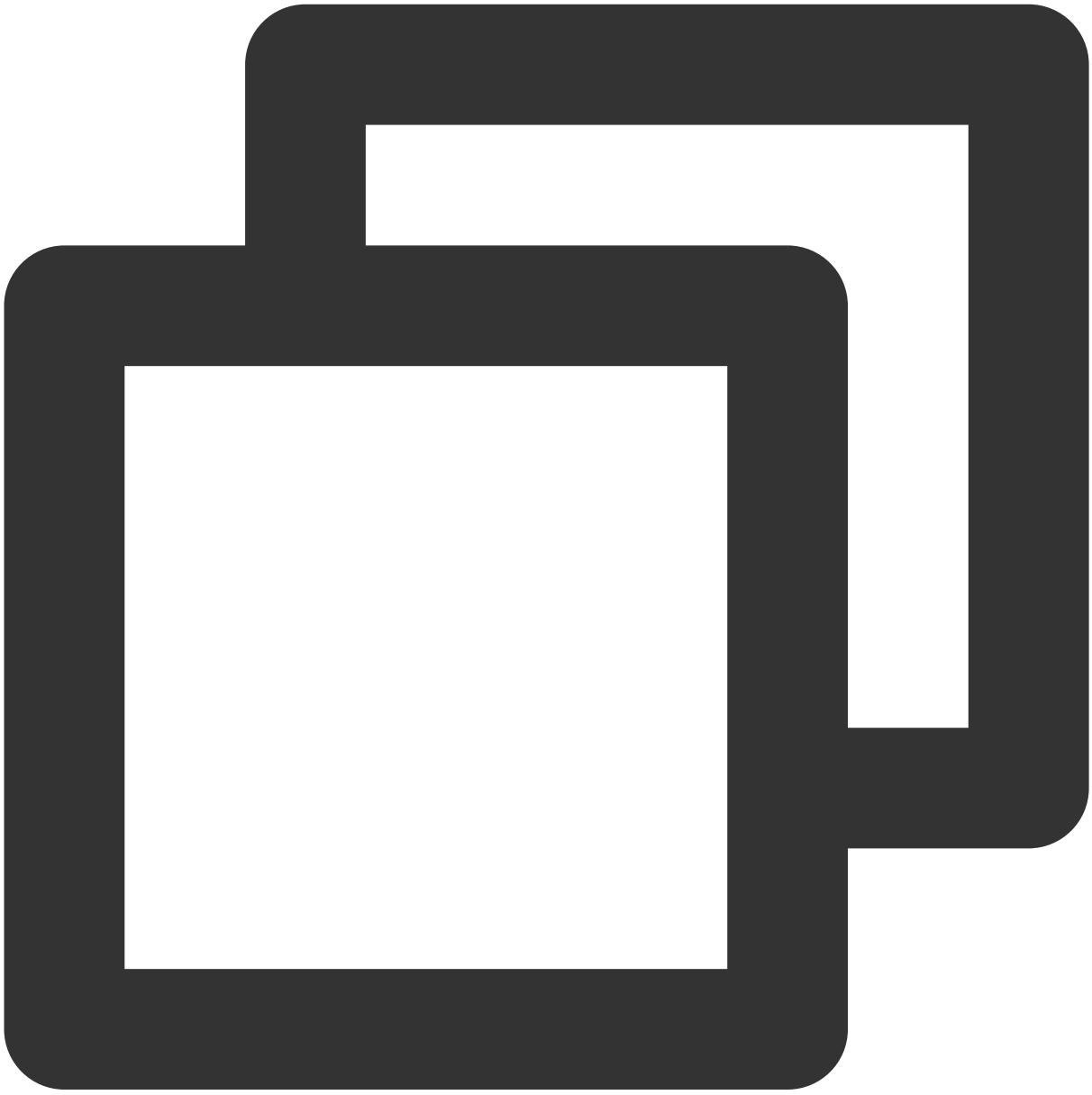
現在の通話を終了します。通話中である場合は、この関数を呼び出して通話を終了できます。
通話中にこのインターフェースを呼び出せば、通話を終了することができます
ダイヤルしていない状態では、通話をキャンセルするために用いることができます



```
tuiCallEngine.hangup().then(() => {
    //success
 }).catch(error => {
    console.warn('hangup error:', error);
 });
```

inviteUser

今回のグループ通話にユーザーを招待します。
ユースケース：グループ通話中のユーザーが自主的に他の人を招待する場合に使用します。



```
const userIDList = ['jack', 'john'];
const params = {
    userIDList
};
tuiCallEngine.inviteUser(params).then(() => {
    // success
```

```
}).catch(error => {
    console.error('inviteUser error:', error);
});
```

パラ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	Array	ターゲットユーザーのuserIDリスト
offlinePushInfo	Object	カスタムオフラインメッセージプッシュ（オプション、tsignalingバージョン>=0.8.0が必要です）、オプション

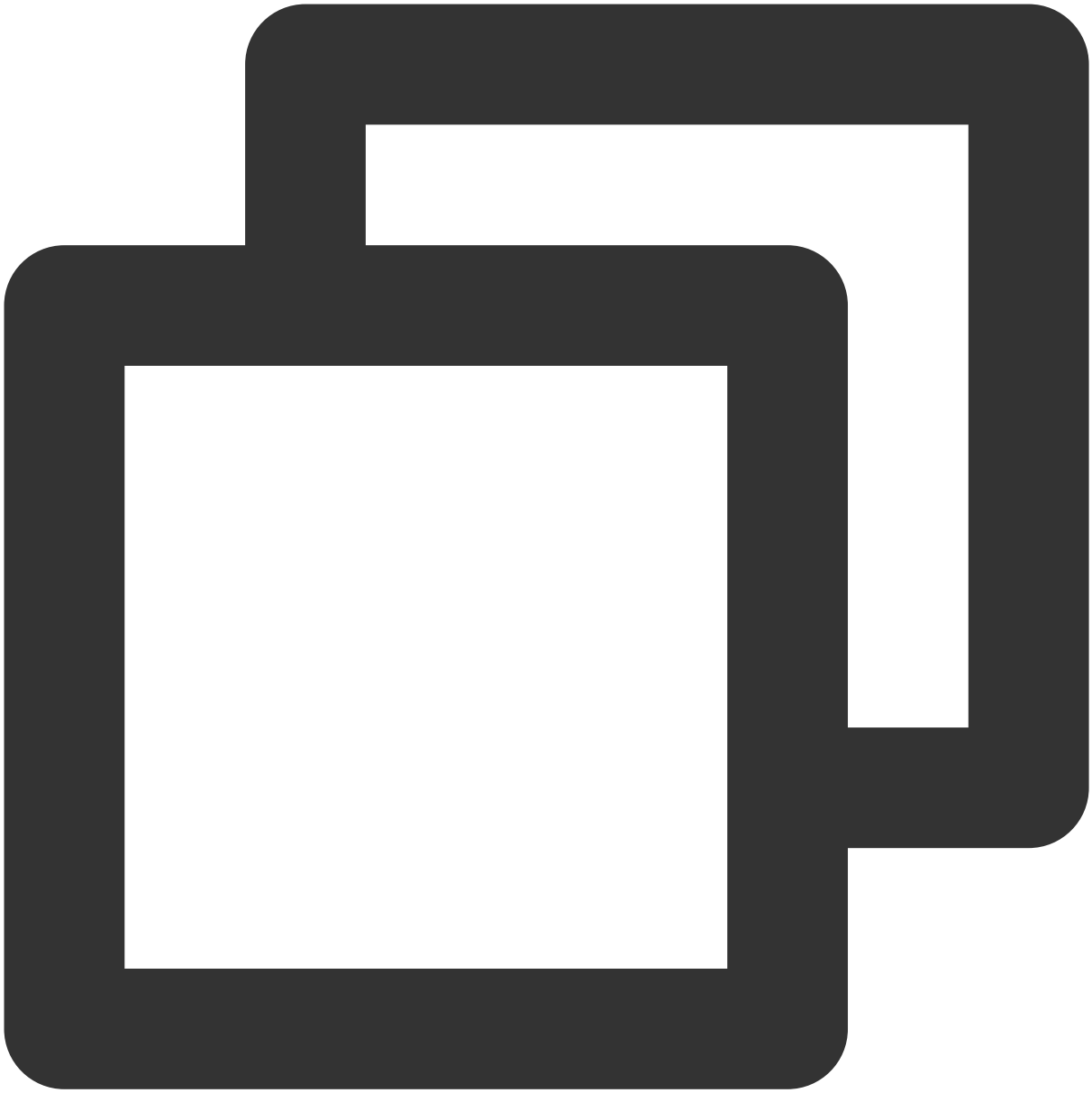
offlinePushInfo

パラメータ	タイプ	意味
title	string	オフラインプッシュタイトル（オプション）
description	string	オフラインプッシュ内容（オプション）
androidOPPOChannelID	string	オフラインプッシュでOPPO携帯電話8.0システム以上のチャンネルIDを設定します（オプション）
extension	string	オフラインプッシュパススルー内容（オプション）（tsignalingバージョン>=0.9.0）

joinInGroupCall

今回のグループ通話に自主的に参加します。

ユースケース：グループ内のユーザーが今回のグループ通話に自主的に参加する場合に使用します。



```
const params = {
    roomID: 123,
    type: 1,
    groupID: 111
};
tuiCallEngine.joinInGroupCall(params).then(() => {
    // success
}).catch(error => {
    console.error('joinInGroupCall error:', error);
});
```

パラメータは下表に示すとおりです。

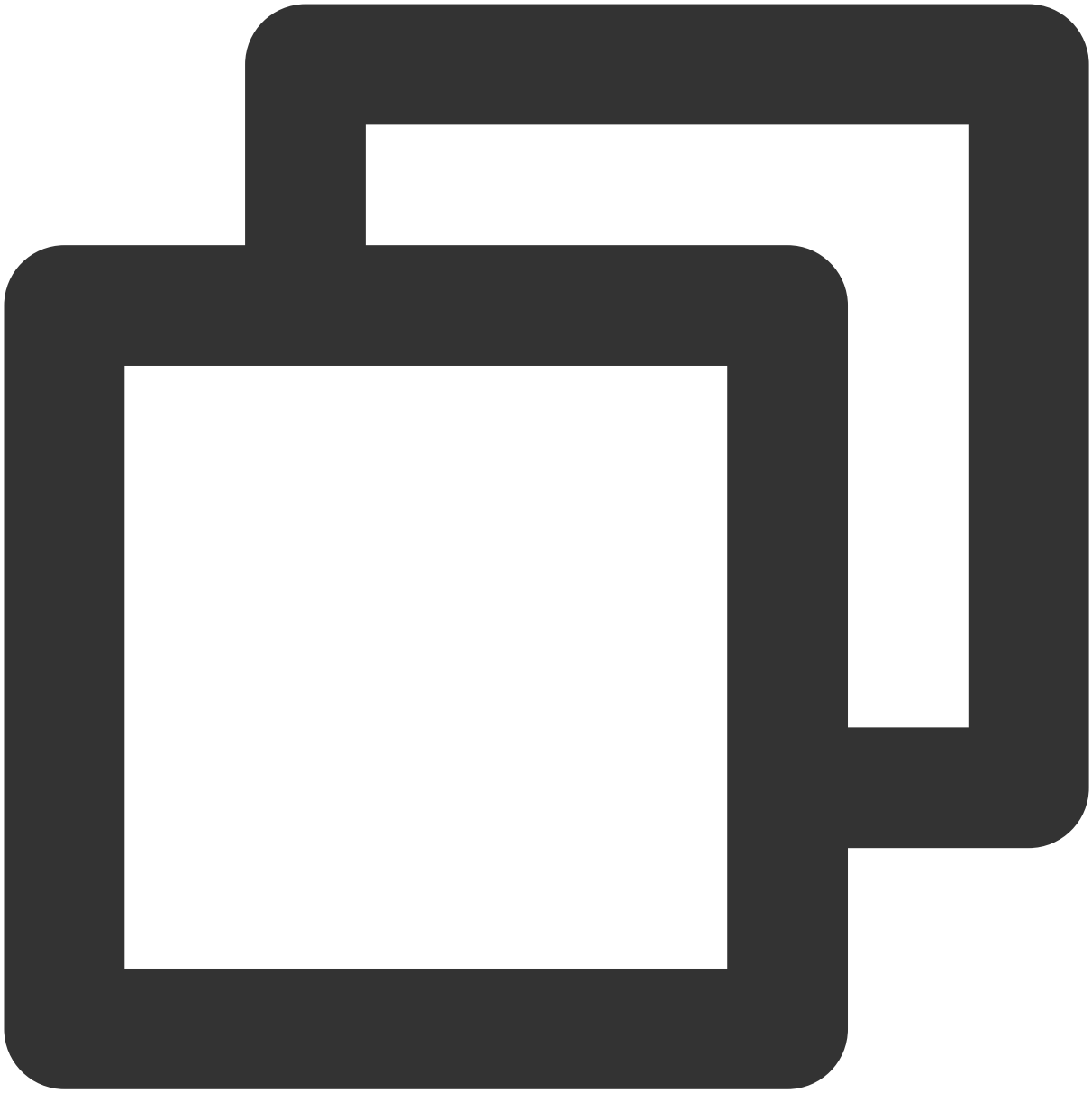
パラメータ	タイプ	意味
roomID	Number	今回の通話のオーディオビデオルームID。現在は数字のルームナンバーのみサポートしています。文字列のルームナンバーは今後のバージョンでサポート予定です
groupID	String	このグループ通話のグループID
type	Number	通話のメディアタイプ。例：1-音声通話、2-ビデオ通話

switchCallMediaType

現在の通話タイプを切り替えます。

1v1通話中のみ使用をサポートします

ERRORイベントの監視に失敗しました。code：60001



```
// 1は音声通話、2はビデオ通話を意味します
tuiCallEngine.switchCallMediaType(2).then(() => {
  //success
}).catch(error => {
  console.warn('switchCallMediaType error:', error);
});
```

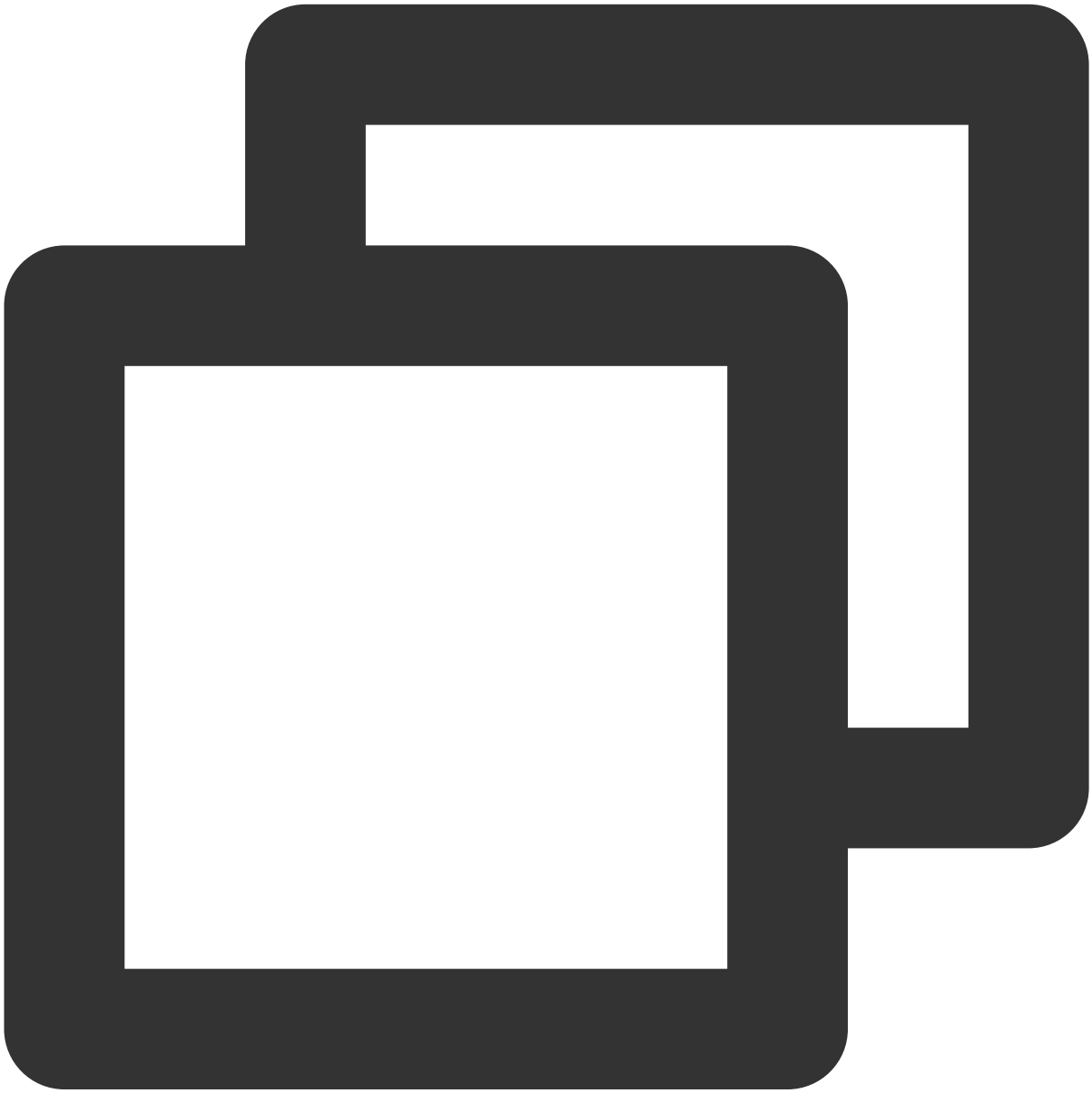
パラメータは下表に示すとおりです

パラメータ	タイプ	意味

newMediaType	Number	1-音声通話、2-ビデオ通話

startRemoteView

リモート画面レンダリングを起動します。USER_VIDEO_AVAILABLEイベントを受信してから、対応するリモート画面を再レンダリングしてください。



```
let promise = tuiCallEngine.startRemoteView({
    userID: 'user1',
    videoViewDomID: 'video_1',
});
```

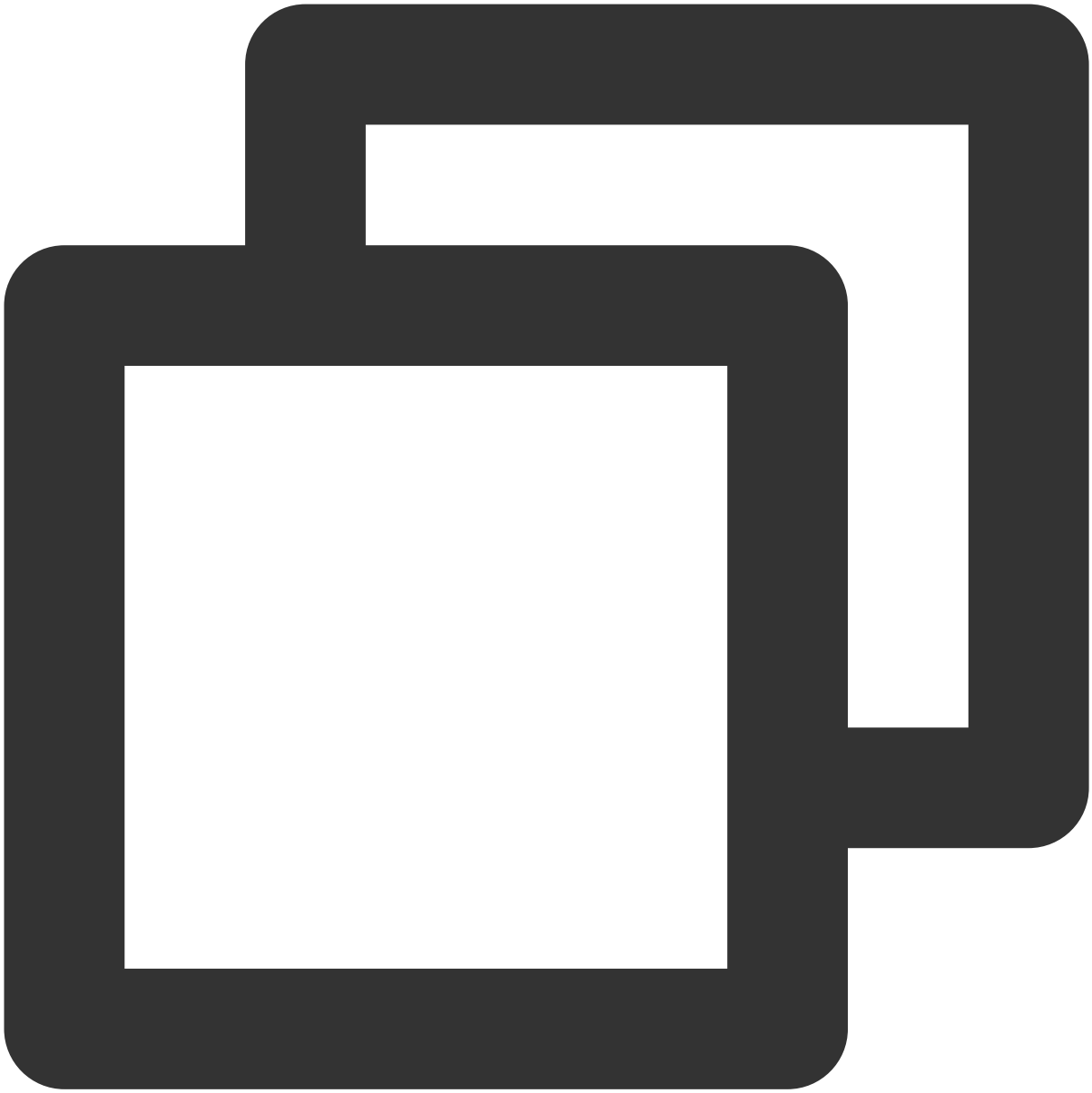
```
promise.then(() => {
    //success
}).catch(error => {
    console.warn('startRemoteView error:', error);
});
```

パラ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
videoViewDomID	String	このユーザーデータはdom idノードにレンダリングされます

stopRemoteView

リモート画面のレンダリングを停止します



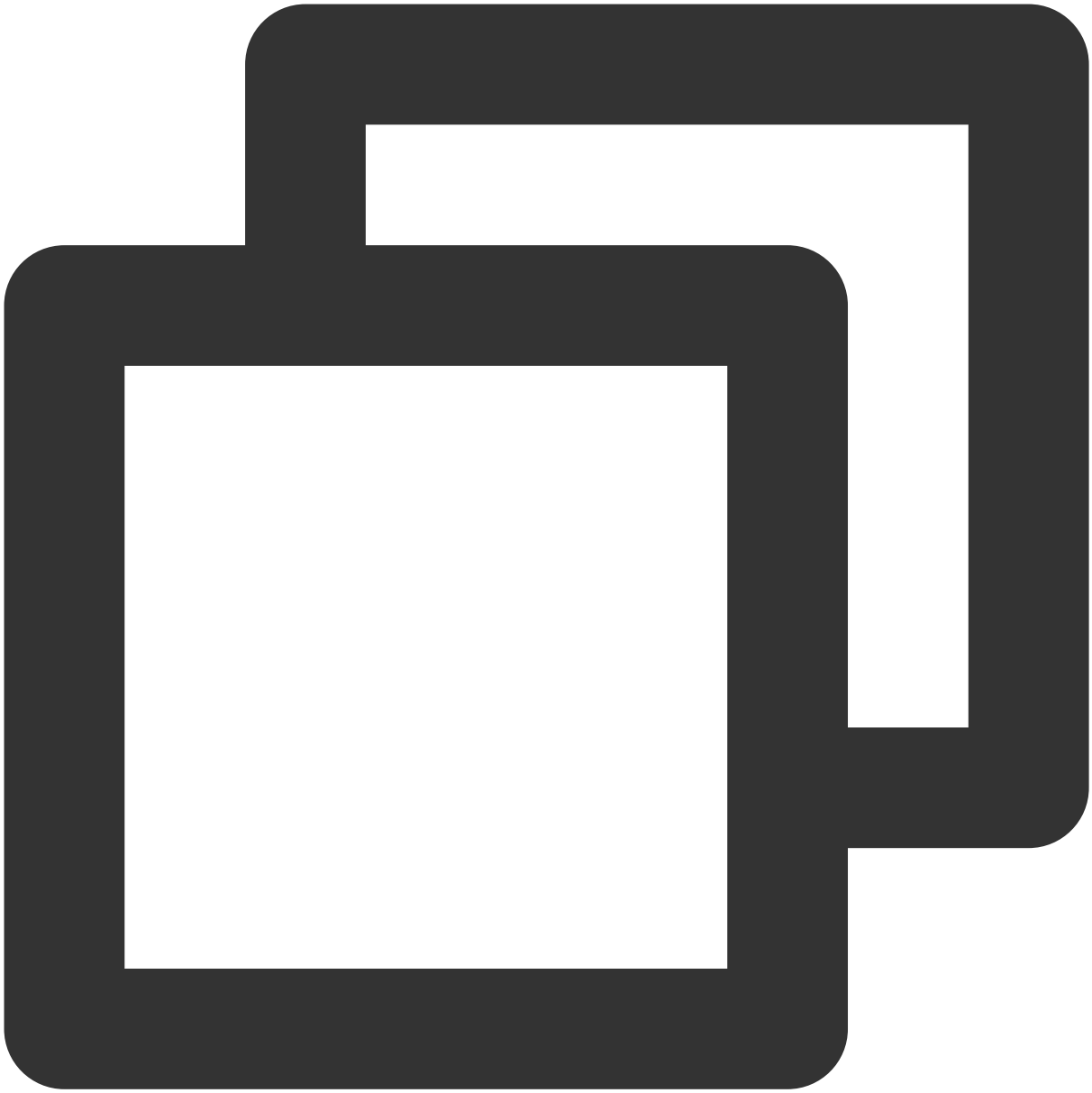
```
tuiCallEngine.stopRemoteView({userID: 'user1'});
```

パラ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

startLocalView

ローカル画面のレンダリングを起動します



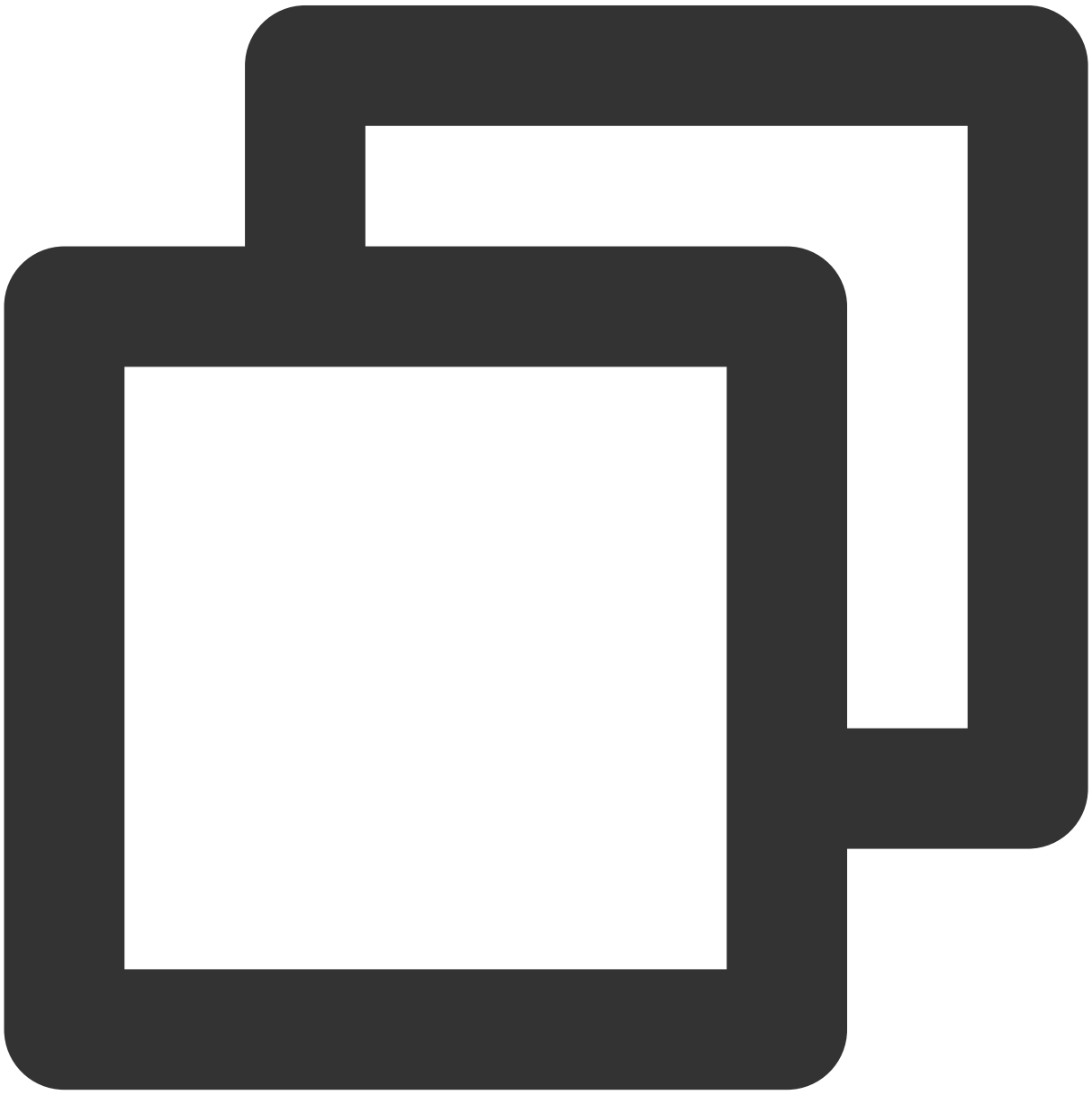
```
let promise = tuiCallEngine.startLocalView({
    userID: 'user1',
    videoViewDomID: 'video_1'
});
promise.then(() => {
    //success
}).catch(error => {
    console.warn('startLocalView error:', error);
});
```

パラ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
videoViewDomID	String	このユーザーデータはdom idノードにレンダリングされます

stopLocalView

ローカル画面のレンダリングを停止します



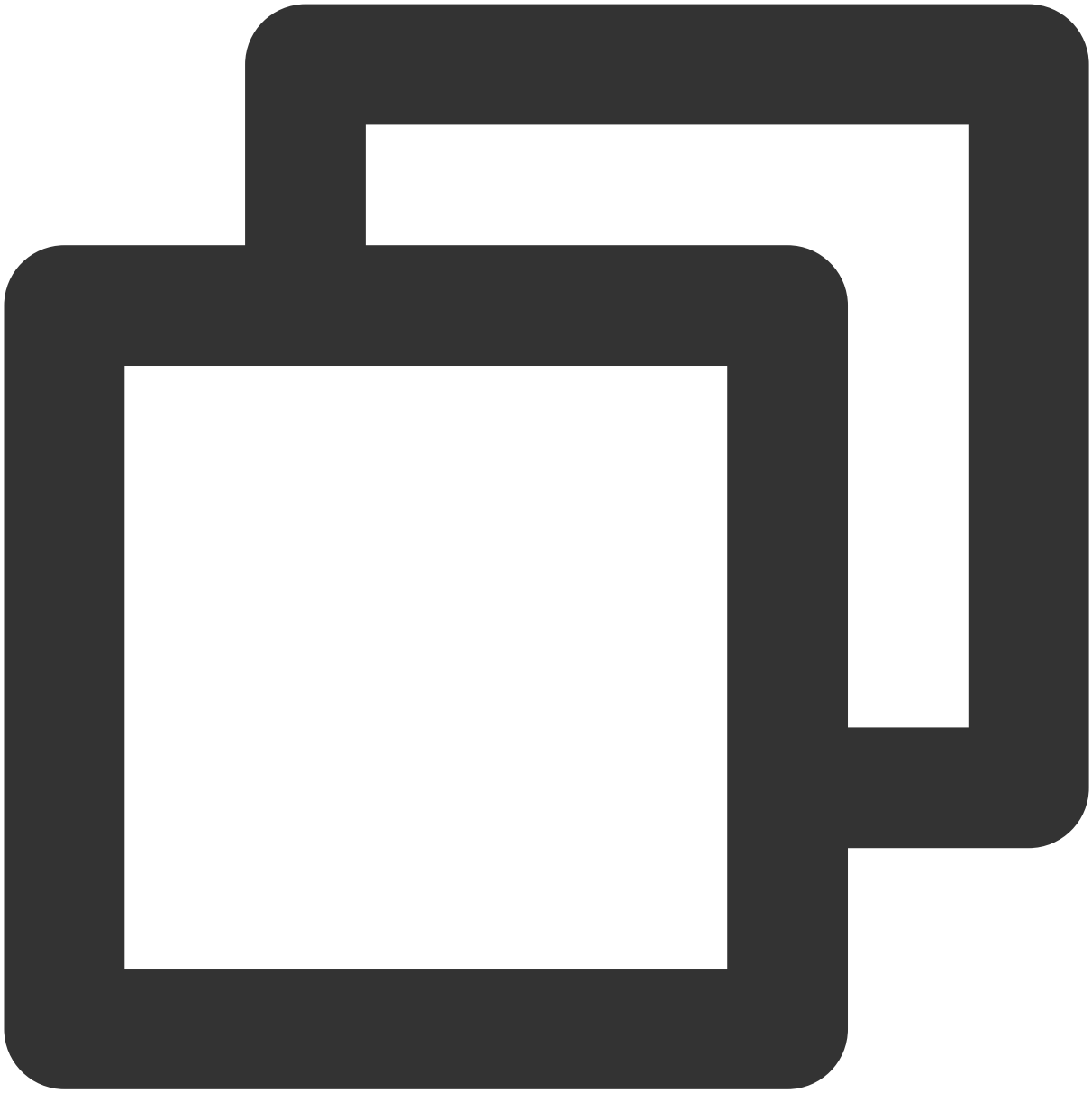
```
let promise = tuiCallEngine.stopLocalView({userID: 'user1'});
promise.then(() => {
    //success
}).catch(error => {
    console.warn('stopLocalView error:', error)
});
```

パラ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

openCamera

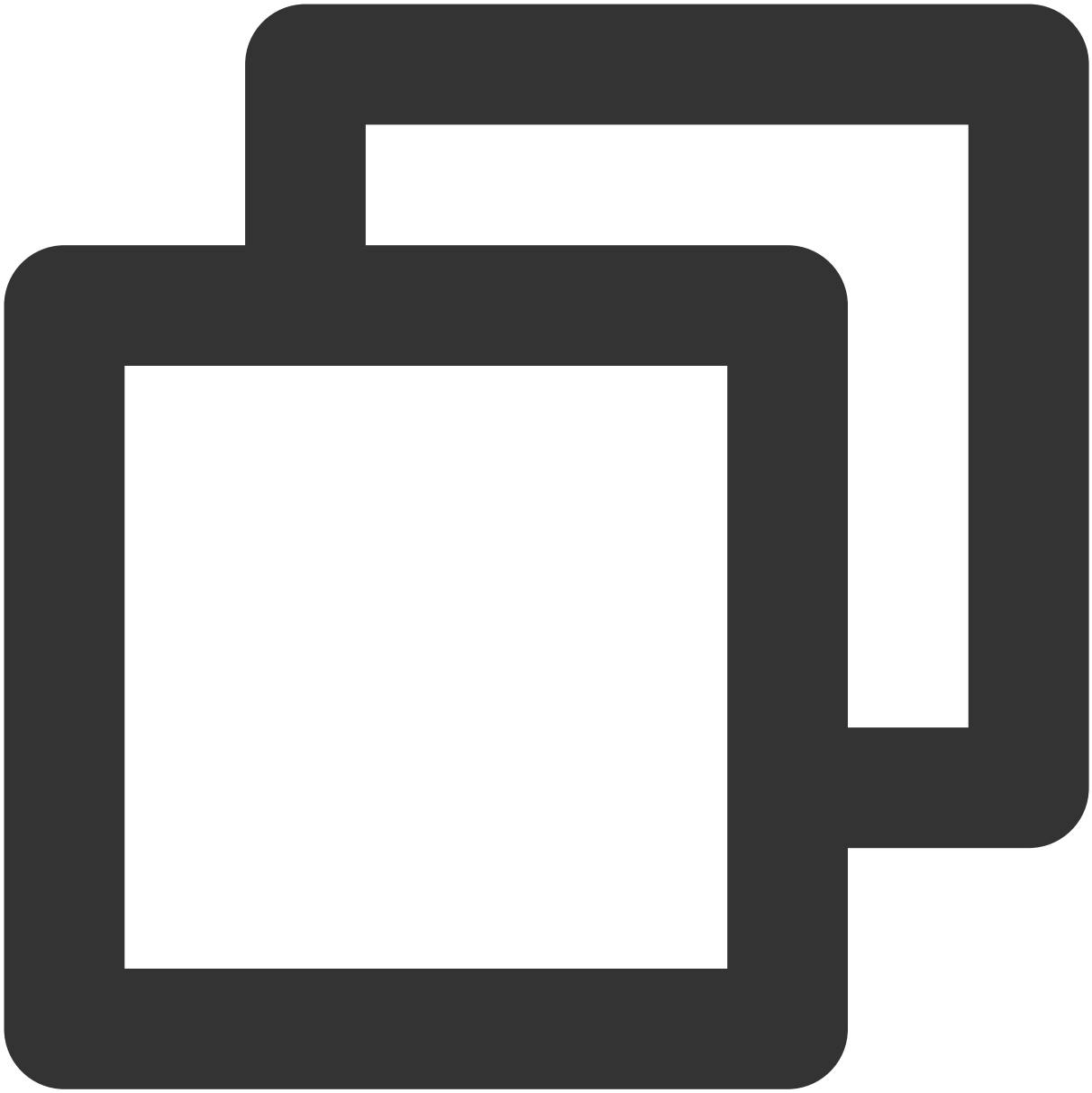
カメラをオンにします。



```
tuiCallEngine.openCamera().then(() => {
    //success
}).catch(error => {
    console.warn('openCamera error:', error);
});
```

closeCamara

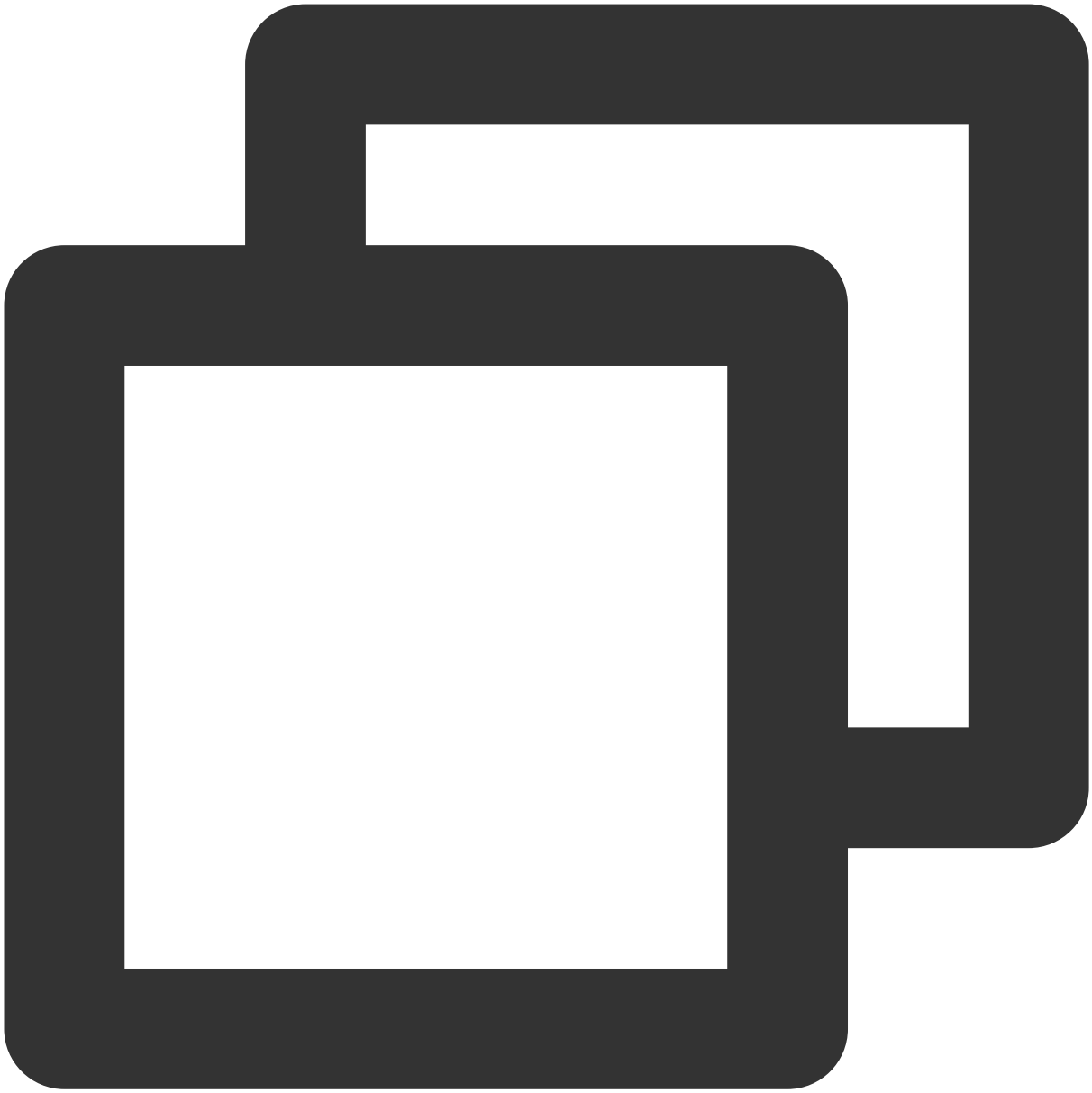
カメラをオフにします



```
tuiCallEngine.closeCamera().then(() => {
    //success
}).catch(error => {
    console.warn('closeCamara error:', error);
});
```

openMicrophone

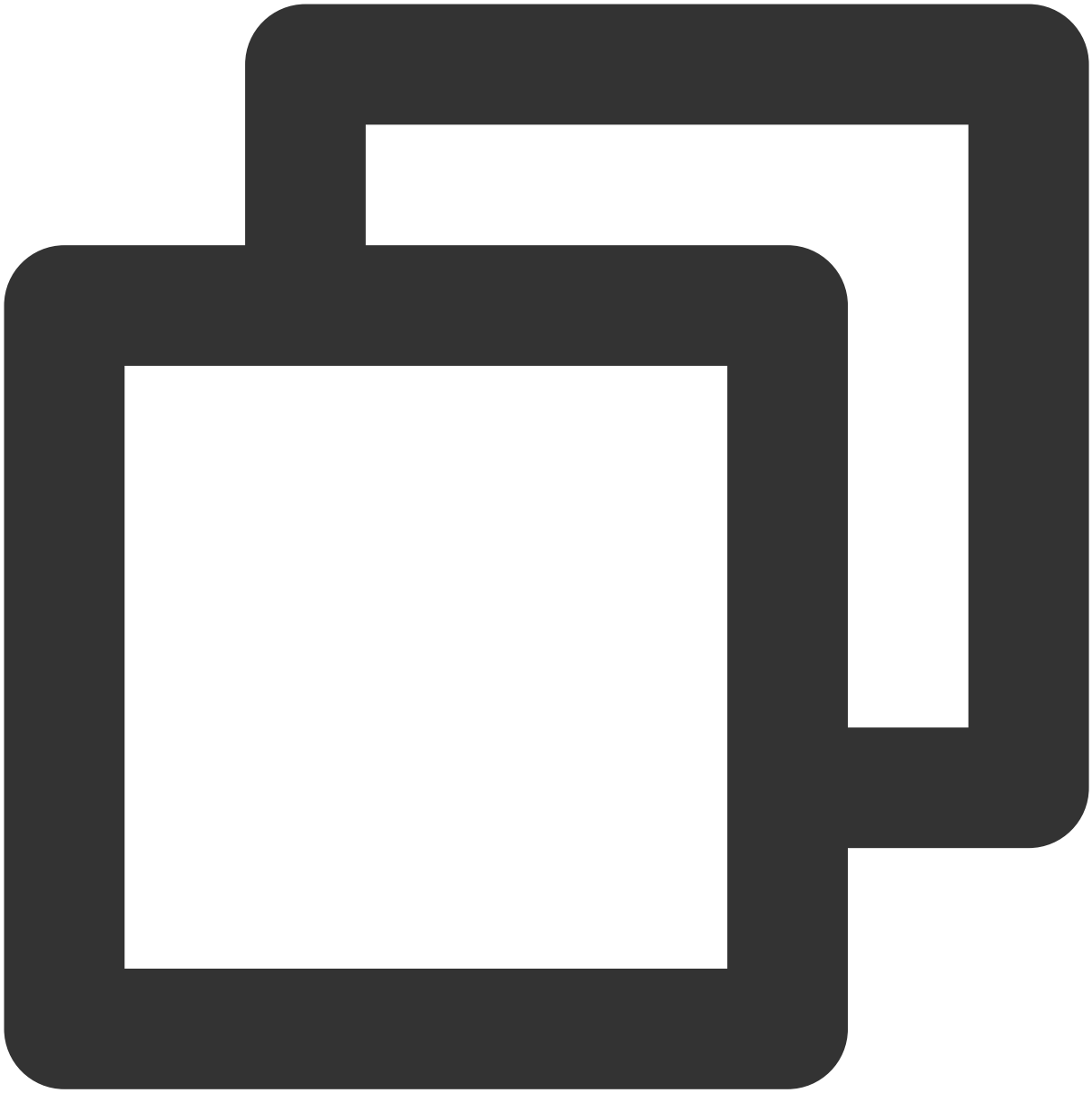
マイクをオンにします。



```
tuiCallEngine.openMicrophone().then(() => {
    //success
}).catch(error => {
    console.warn('openMicrophone error:', error);
});
```

closeMicrophone

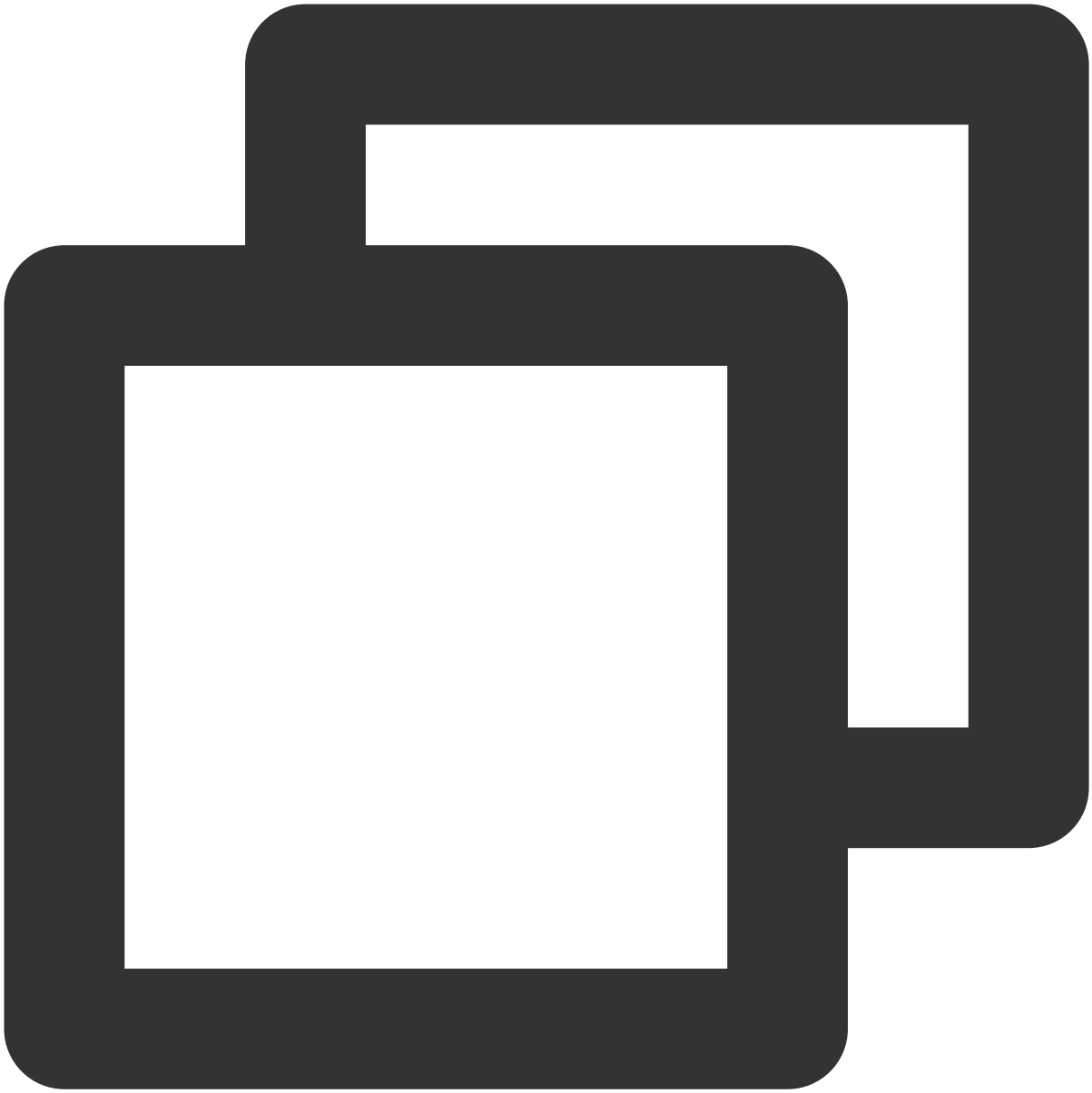
マイクをオフにします。



```
tuiCallEngine.closeMicrophone().then(() => {
    //success
}).catch(error => {
    console.warn('closeMicrophone error:', error);
});
```

setVideoQuality

ビデオ画質を設定します。



```
const profile = '720p';
tuiCallEngine.setVideoQuality(profile).then(() => {
    //success
}).catch(error => {
    console.warn('setVideoQuality error:', error)
});    // ビデオ画質を720pに設定します
```

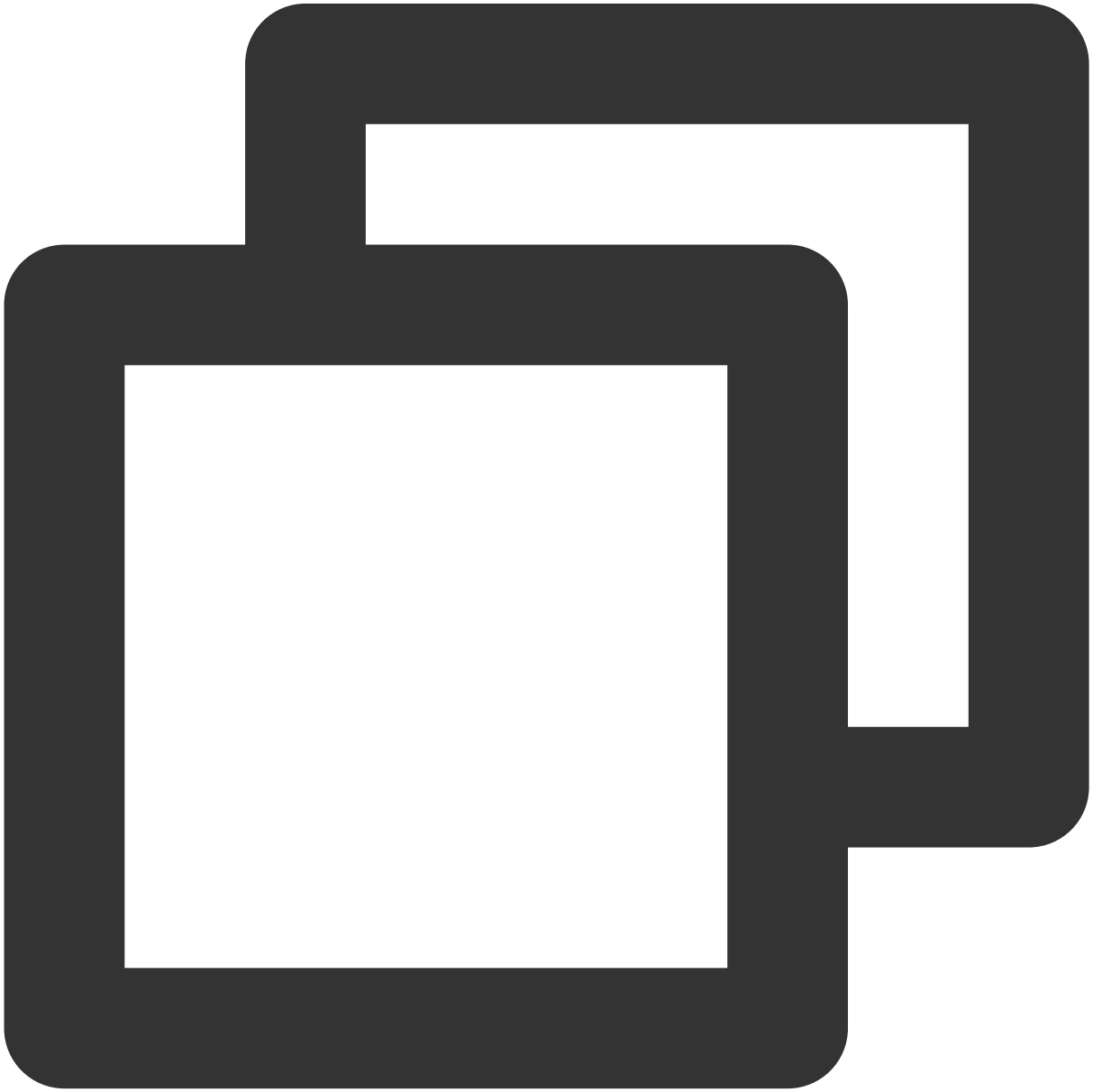
パラメータは下表に示すとおりです

ビデオProfile	解像度（幅 x 高さ）

480p	640 × 480
720p	1280 × 720
1080p	1920 × 1080

getDeviceList

デバイスリストを取得します。



```
tuiCallEngine.getDeviceList("camera").then((devices) => {
```

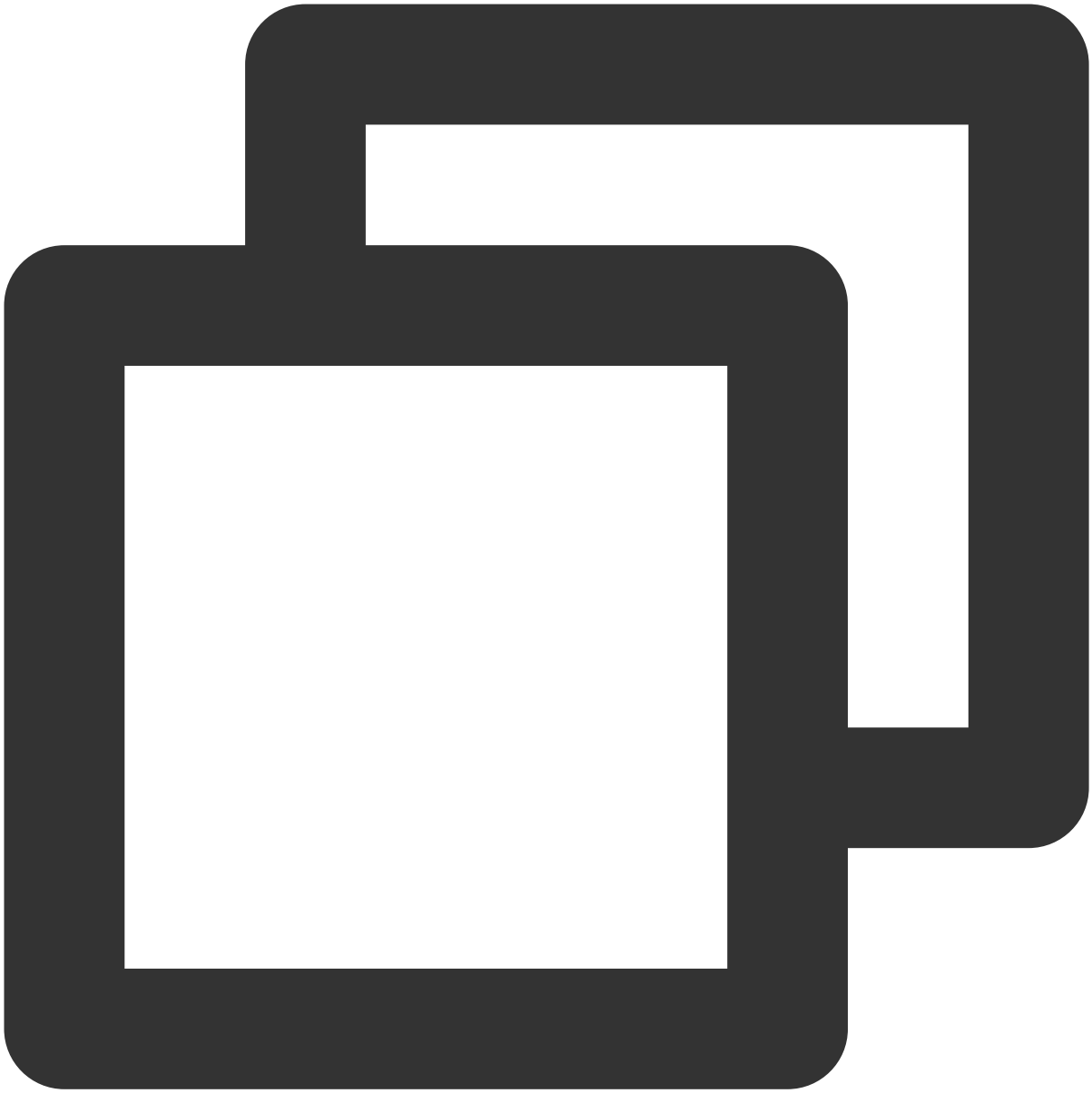
```
    console.log(devices);
}).catch(error => {
    console.warn('getDeviceList error:', error);
});
```

パラメータは下表に示すとおりです

パラメータ	タイプ	意味
deviceType	String	'camera'-カメラ、'microphones'-マイク

switchDevice

カメラまたはマイクデバイスを切り替えます。



```
let promsie = tuiCallEngine.switchDevice({
    deviceType: 'video',
    deviceId: cameras[0].deviceId
});
promise.then(() => {
    //success
}).catch(error => {
    console.warn('switchDevice error:', error)
});
```

パラメータは下表に示すとおりです

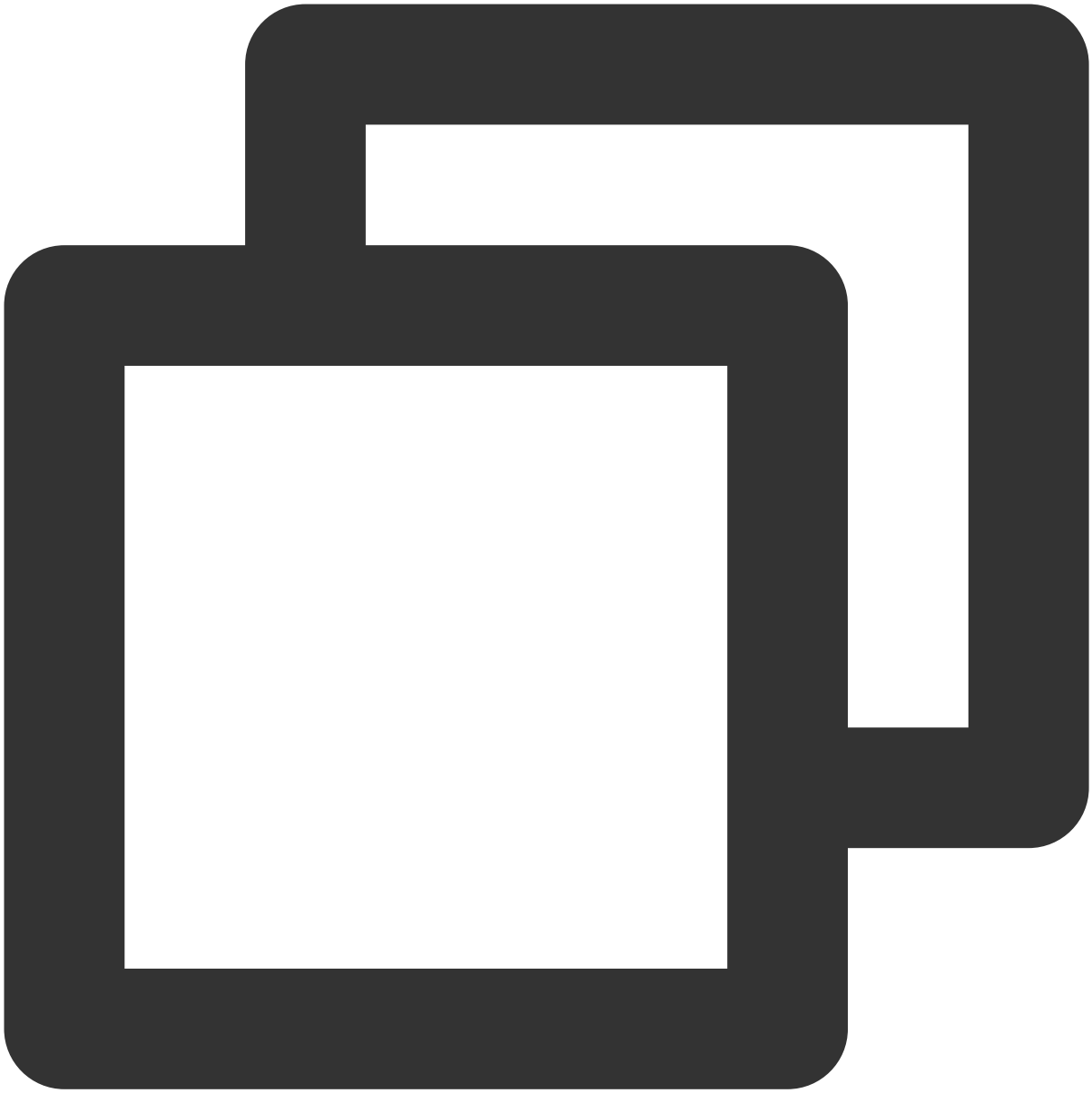
パラメータ	タイプ	意味
deviceType	String	切り替えが必要なデバイスタイプ 'video'カメラ 'audio'マイク
deviceId	String	切り替えが必要なデバイスID カメラデバイスの識別子は、getCameras()で取得します マイクデバイスの識別子は、getMicrophones()で取得します

enableAIVoice

AIノイズリダクションのオン/オフ。

ご注意：

バージョン4.12.1以降でサポートしています。使用方法の詳細については、AIノイズリダクションの使用をご参照ください。



```
let promsie = tuiCallEngine.enableAIVoice(true);
promise.then(() => {
    // success
}).catch(error => {
    console.warn('enableAIVoice error:', error)
});
```

パラ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	AIノイズリダクションのオン/オフ。enable=trueはオンを示し、デフォルトではAIノイズリダクションはオンになっています

TUICallEvent

最終更新日：：2024-07-19 14:53:21

TUICallEvent APIの概要

TUICallEvent APIはオーディオビデオ通話コンポーネントの**イベントインターフェース**です。

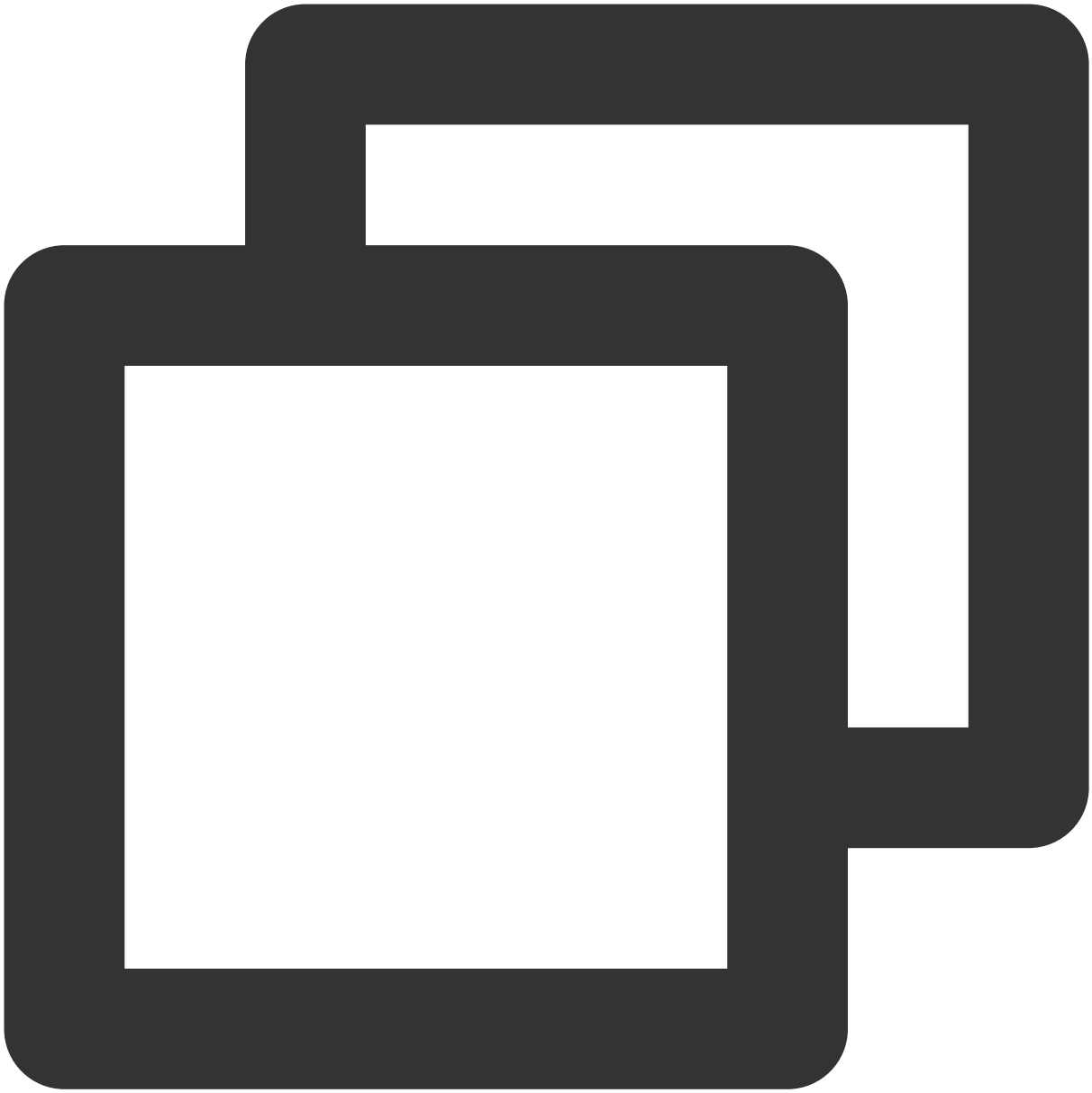
イベントリスト

EVENT	説明
TUICallEvent.ERROR	SDKの内部でエラーが発生しました
TUICallEvent.SDK_READY	SDKがready状態に入ったときにこのコールバックを受信します
TUICallEvent.KICKED_OUT	重複ログインです。このコールバックを受信した場合は、ルームからの強制退出を意味します
TUICallEvent.USER_ACCEPT	応答したユーザーがいる場合に、このコールバックを受信します
TUICallEvent.USER_ENTER	通話への参加に同意したユーザーがいる場合に、このコールバックを受信します
TUICallEvent.USER_LEAVE	通話からの退出に同意したユーザーがいる場合に、このコールバックを受信します
TUICallEvent.REJECT	ユーザーが通話を拒否
TUICallEvent.NO_RESP	招待したユーザーからの応答なし
TUICallEvent.LINE_BUSY	招待者が通話中
TUICallEvent.CALLING_TIMEOUT	被招待者が受信します。このコールバックを受信した場合は、今回の通話に応答せずタイムアウトしたことを意味します
TUICallEvent.USER_VIDEO_AVAILABLE	リモートユーザーによるカメラのオン/オフがあった場合に、このコールバックを受信します
TUICallEvent.USER_AUDIO_AVAILABLE	リモートユーザーによるマイクのオン/オフがあっ

	た場合に、このコールバックを受信します
TUICallEvent.USER_VOICE_VOLUME	リモートユーザーがスピーカーの音量調整を行った場合に、このコールバックを受信します
TUICallEvent.GROUP_CALL_INVITEE_LIST_UPDATE	グループチャットの招待リストが更新された場合にこのコールバックを受信します
TUICallEvent.INVITED	通話に招待されました
TUICallEvent.CALLING_CANCEL	被招待者が受信します。このコールバックを受信した場合は、今回の通話がキャンセルされたことを意味します
TUICallEvent.CALLING_END	このコールバックを受信した場合は、今回の通話が終了したことを意味します
TUICallEvent.DEVICED_UPDATED	デバイスリストが更新された場合にこのコールバックを受信します
TUICallEvent.CALL_TYPE_CHANGED	通話タイプが切り替わった場合にこのコールバックを受信します

ERROR

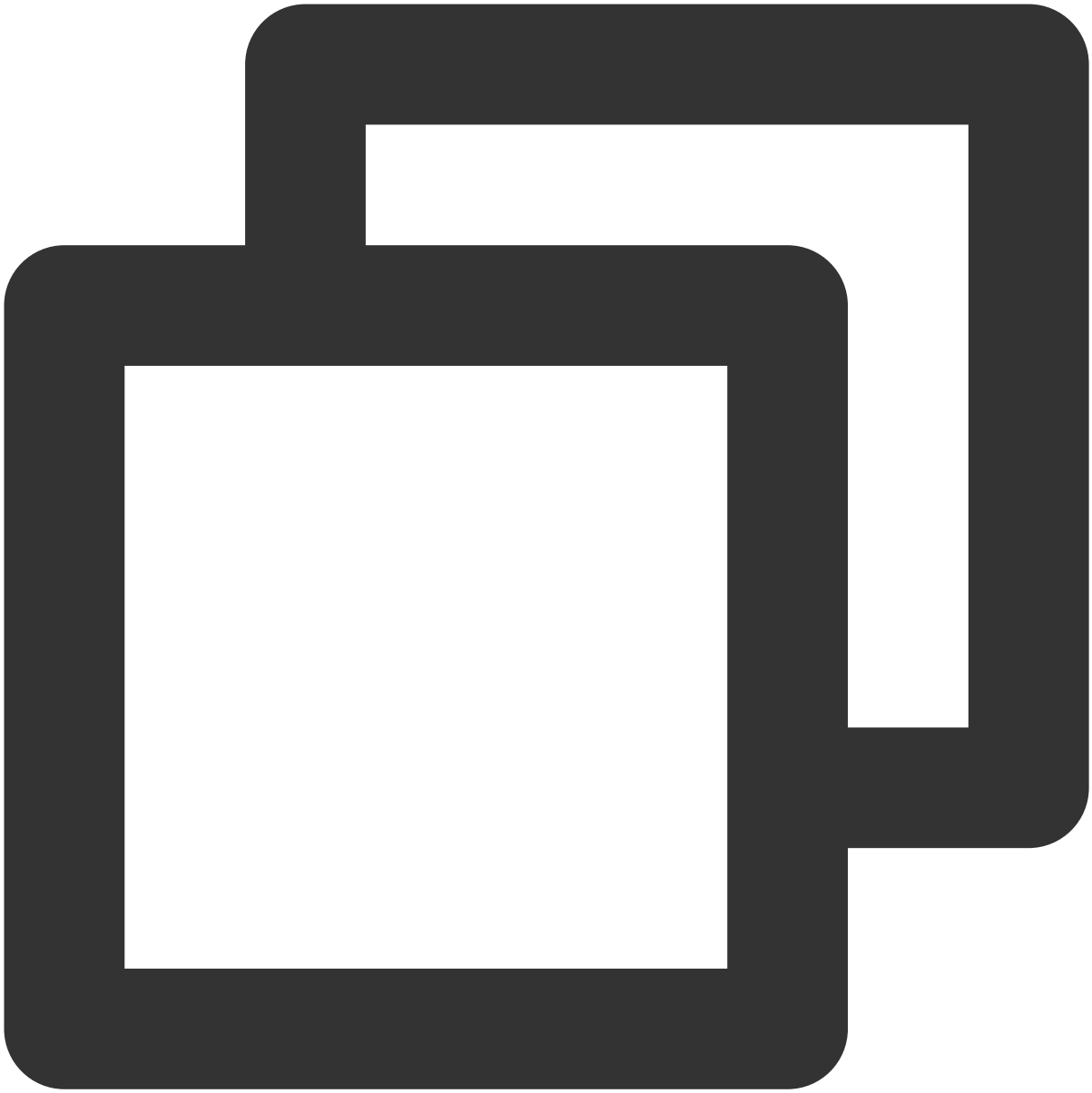
SDK内部にエラーが発生しました。



```
let onError = function(error) {
  console.log(error)
};
tuiCallEngine.on(TUICallEvent.ERROR, onError);
```

SDK_READY

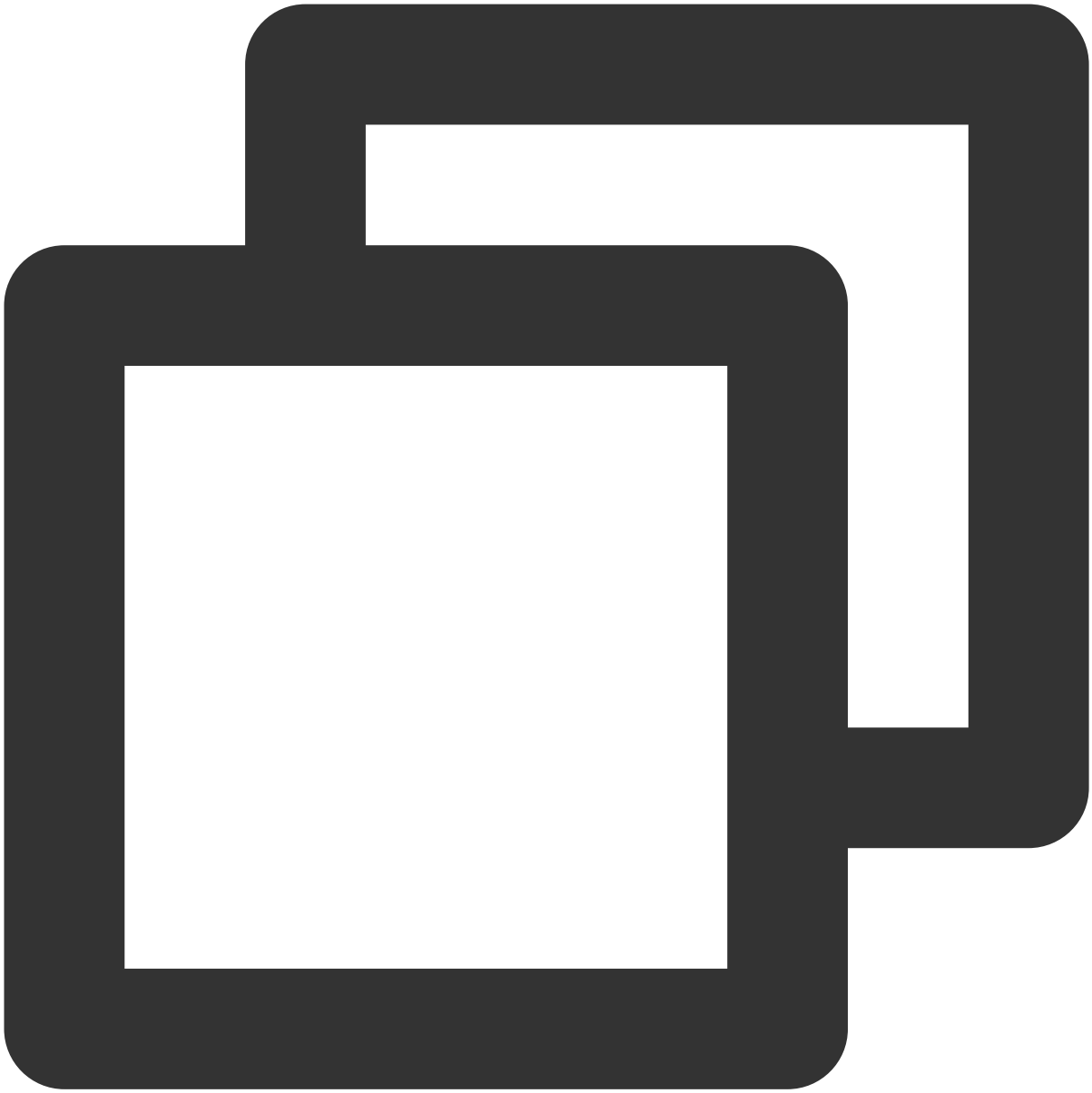
SDKがready状態に入るとこのコールバックを受信します



```
let onSDKReady = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.SDK_READY, onSDKReady);
```

KICKED_OUT

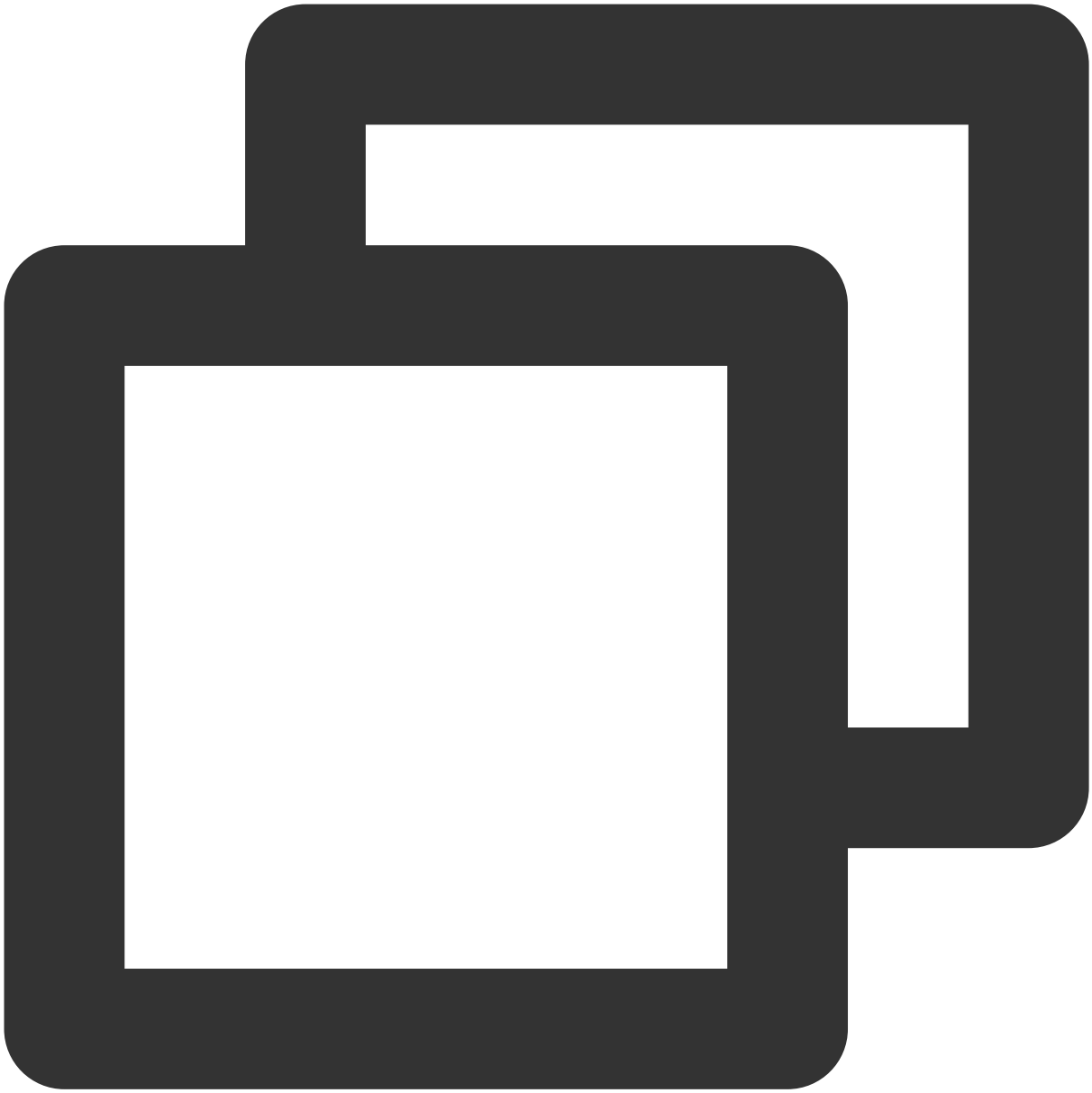
重複ログインです。このコールバックを受信した場合は、ルームからの強制退出を意味します。



```
let handleOnKickedOut = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.KICKED_OUT, handleOnKickedOut);
```

USER_ACCEPT

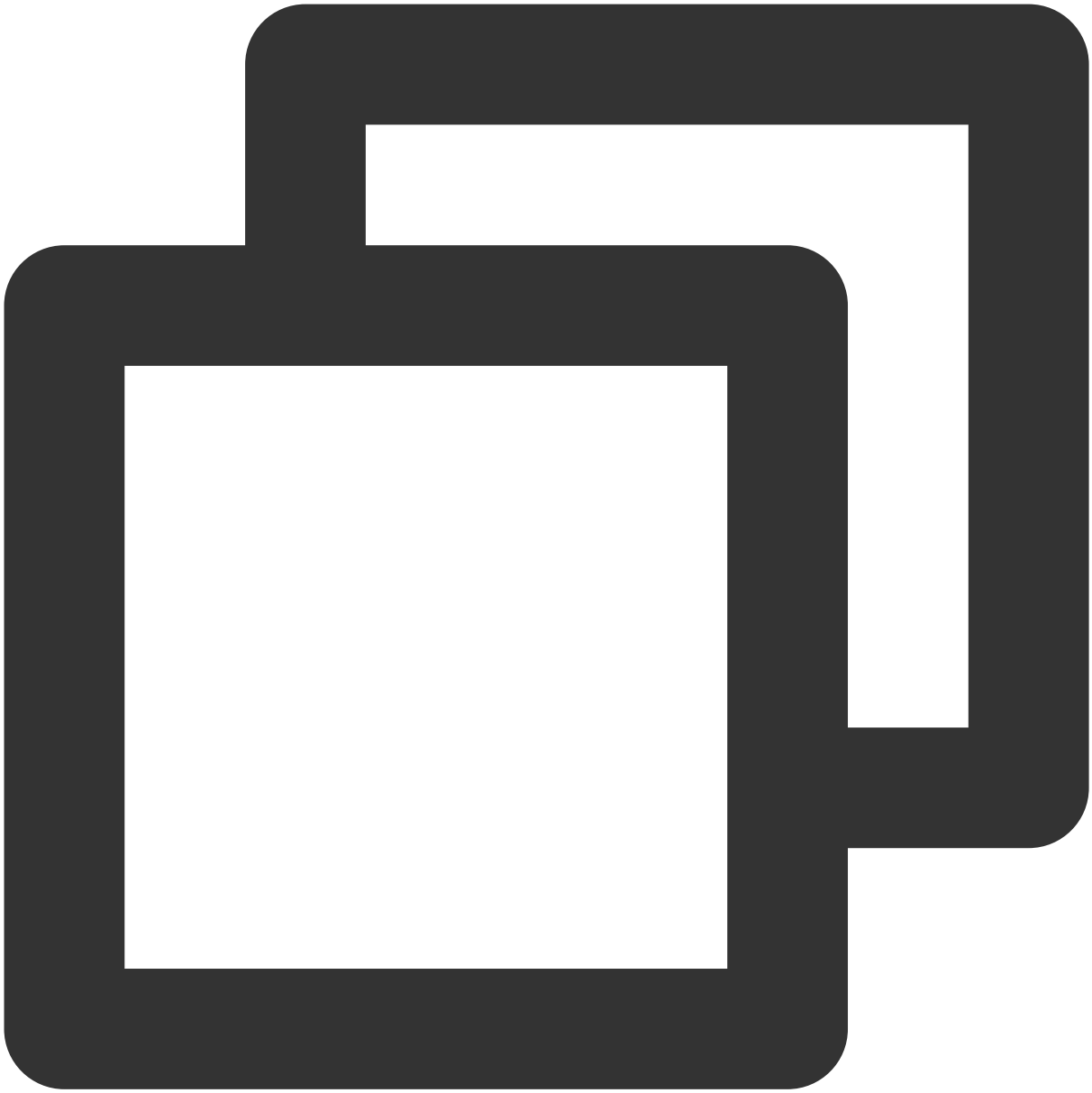
応答したユーザーがいる場合に、このコールバックを受信します。



```
let handleUserAccept = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.USER_ACCEPT, handleUserAccept);
```

USER_ENTER

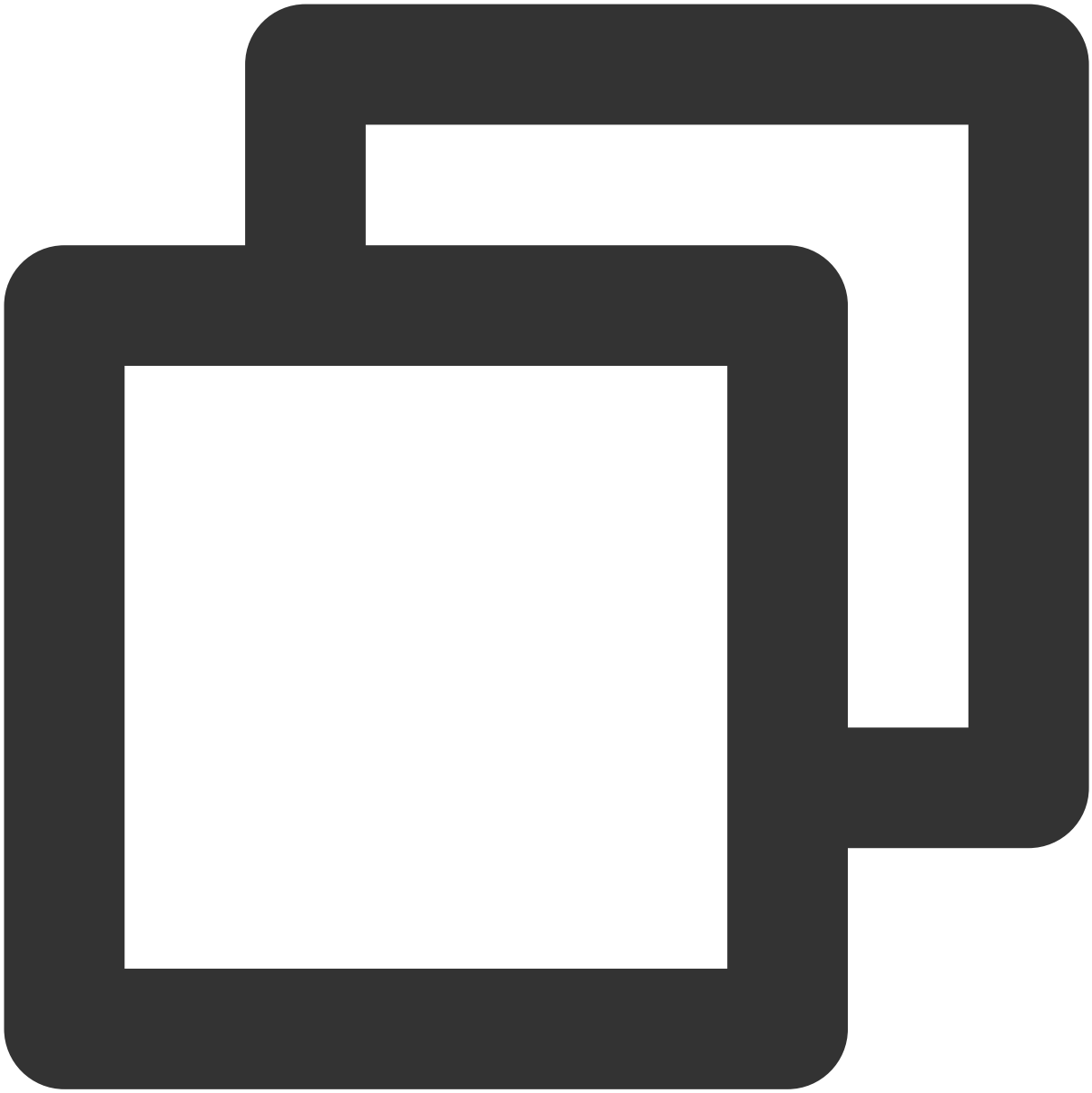
通話への参加に同意したユーザーがいる場合に、このコールバックを受信します。



```
let handleUserEnter = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.USER_ENTER, handleUserEnter);
```

USER_LEAVE

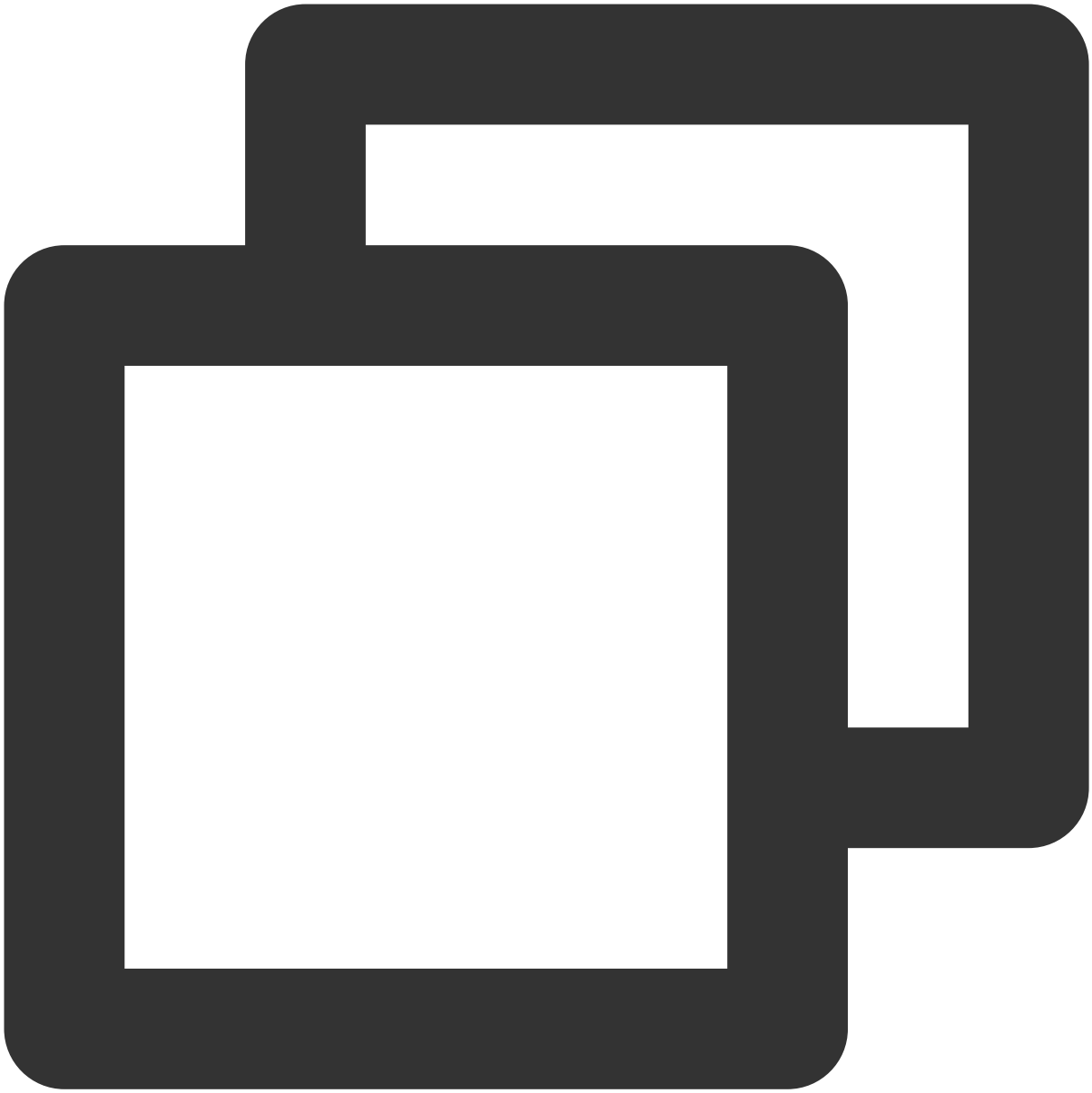
通話からの退出に同意したユーザーがいる場合に、このコールバックを受信します。



```
let handleUserLeave = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.USER_LEAVE, handleUserLeave);
```

REJECT

ユーザーが通話を拒否しました。

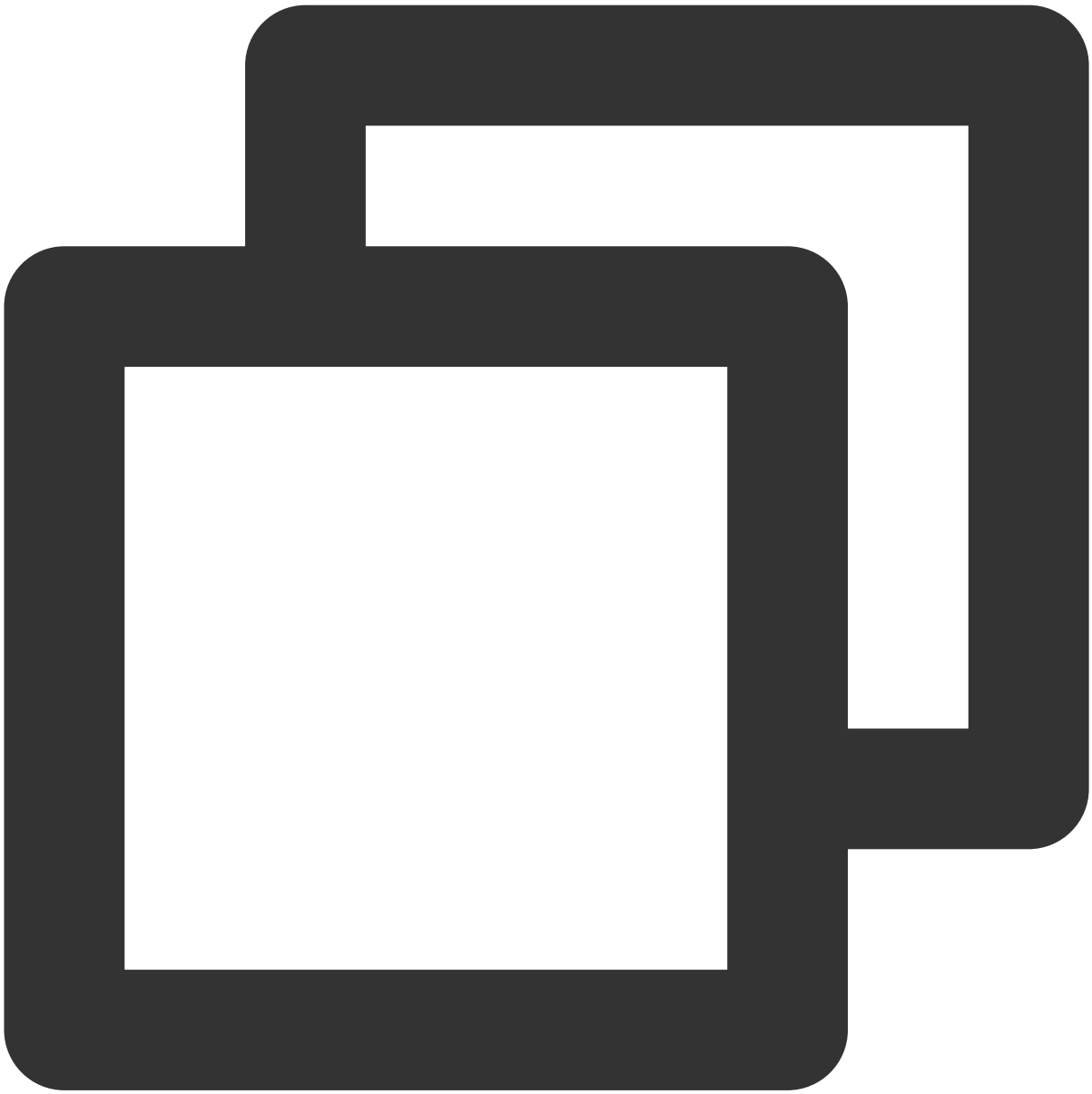


```
let handleInviteeReject = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.REJECT, handleInviteeReject);
```

NO_RESP

招待されたユーザーは応答しませんでした。
C2C通話では、発信者のみが応答なしのコールバックを受信します。例えばA がB、C を通話に招待し、Bが応答しなかった場合、Aはこのコールバックを受信できますが、Cは受信できません。

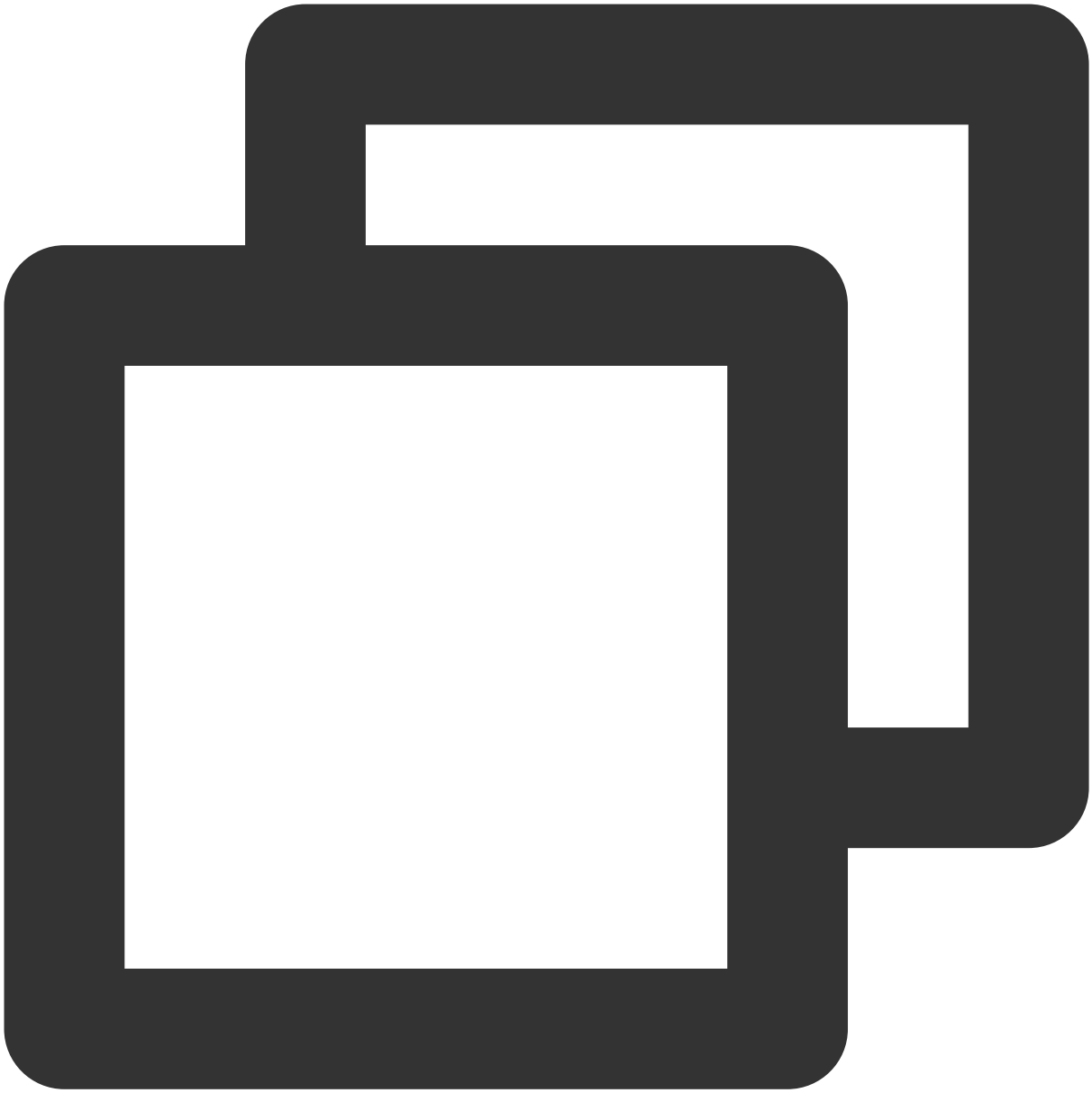
IMグループ通話では、すべての被招待者がこのコールバックを受信できます。例えばA がB、C を通話に招待し、Bが応答しなかった場合、A、Cのどちらもこのコールバックを受信できます。



```
let handleNoResponse = function(event) {
console.log(event)
};
tuiCallEngine.on(TUICallEvent.NO_RESP, handleNoResponse);
```

LINE_BUSY

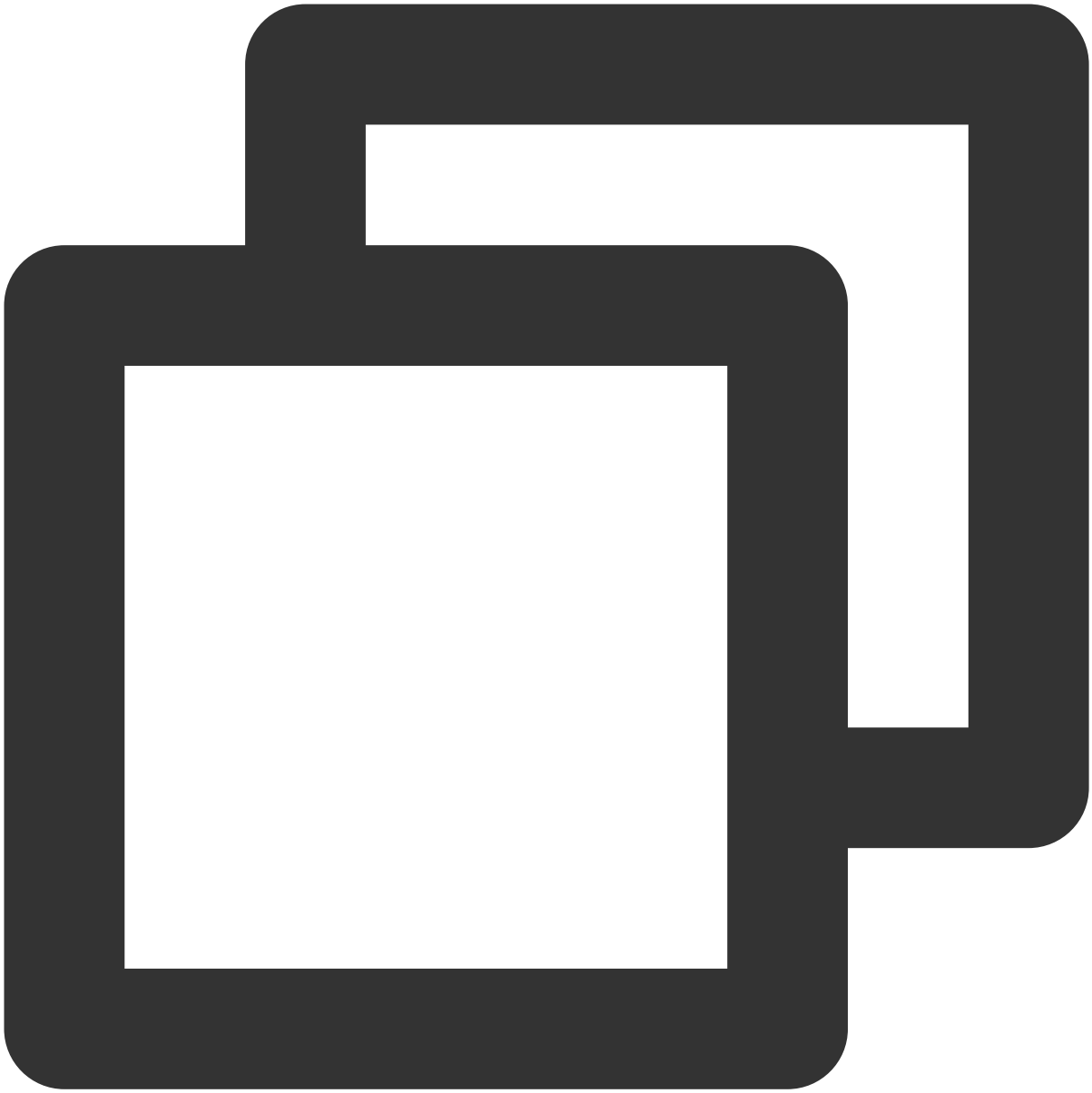
招待者が通話中です。



```
let handleLineBusy = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.LINE_BUSY, handleLineBusy);
```

CALLING_TIMEOUT

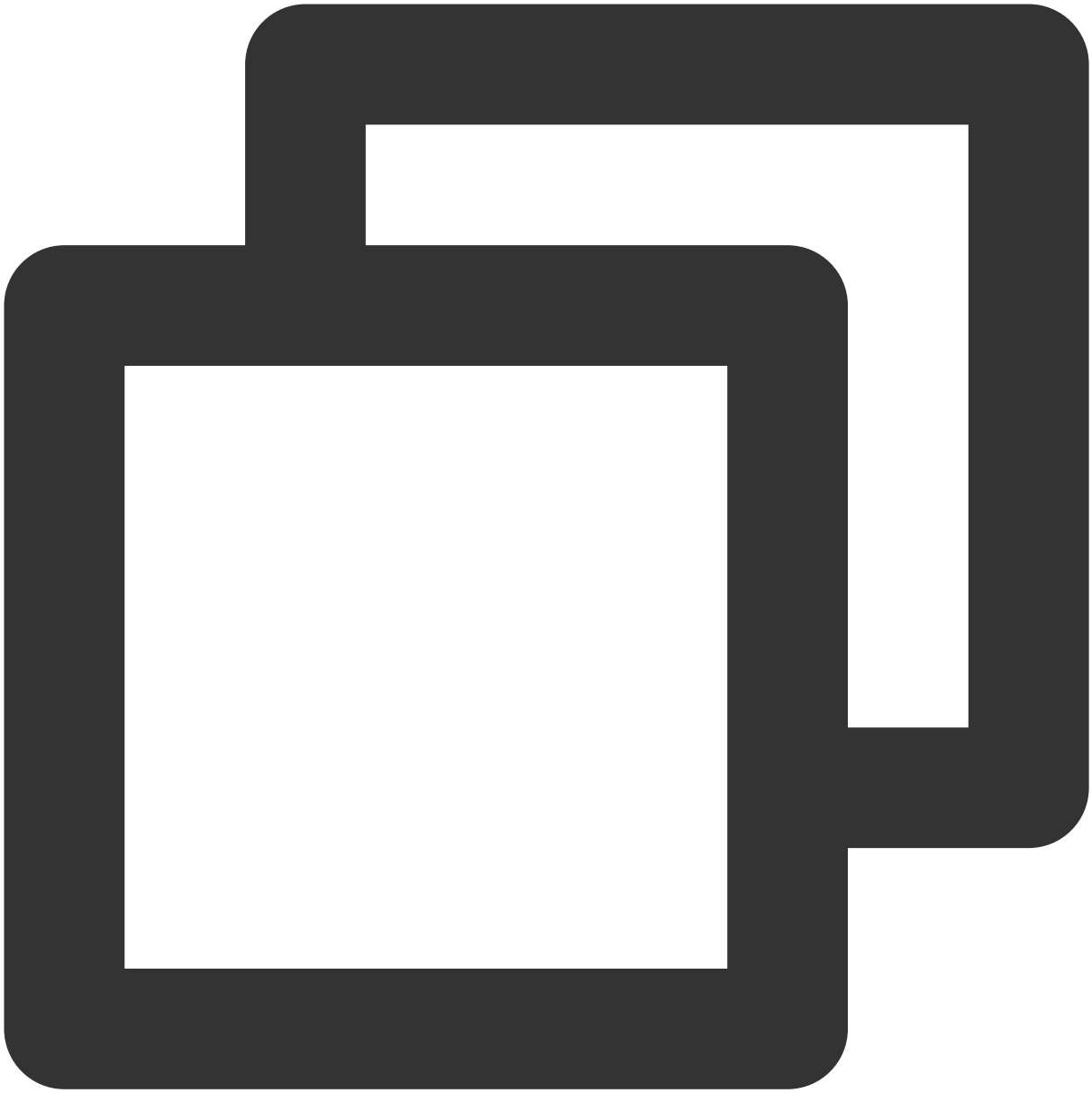
被招待者が受信します。このコールバックを受信した場合は、今回の通話に応答せずタイムアウトしたことを意味します。



```
let handleCallingTimeout = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.CALLING_TIMEOUT, handleCallingTimeout);
```

USER_VIDEO_AVAILABLE

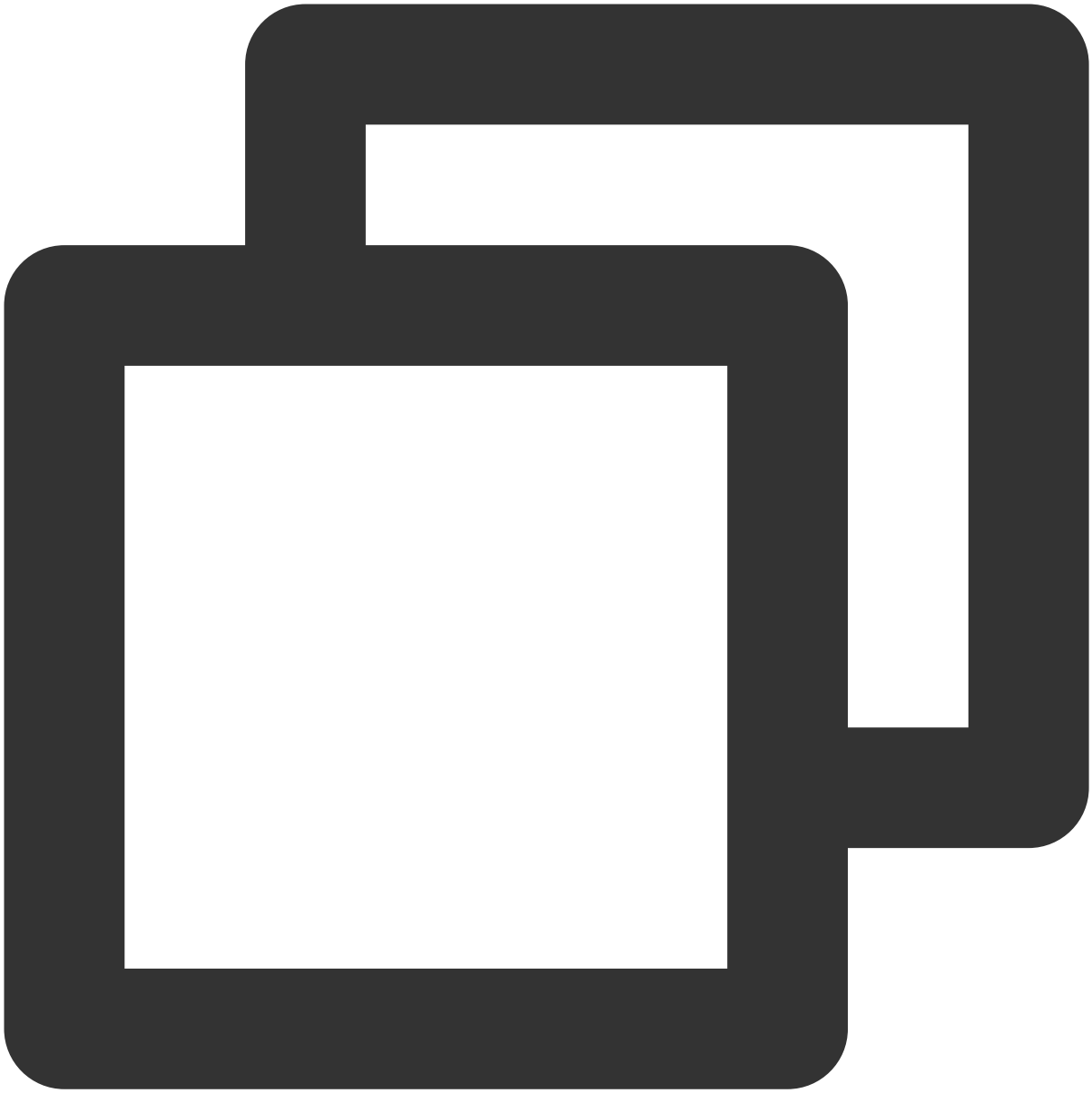
リモートユーザーによるカメラのオン/オフがあった場合に、このコールバックを受信します。



```
let handleUserVideoChange = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.USER_VIDEO_AVAILABLE, handleUserVideoChange);
```

USER_AUDIO_AVAILABLE

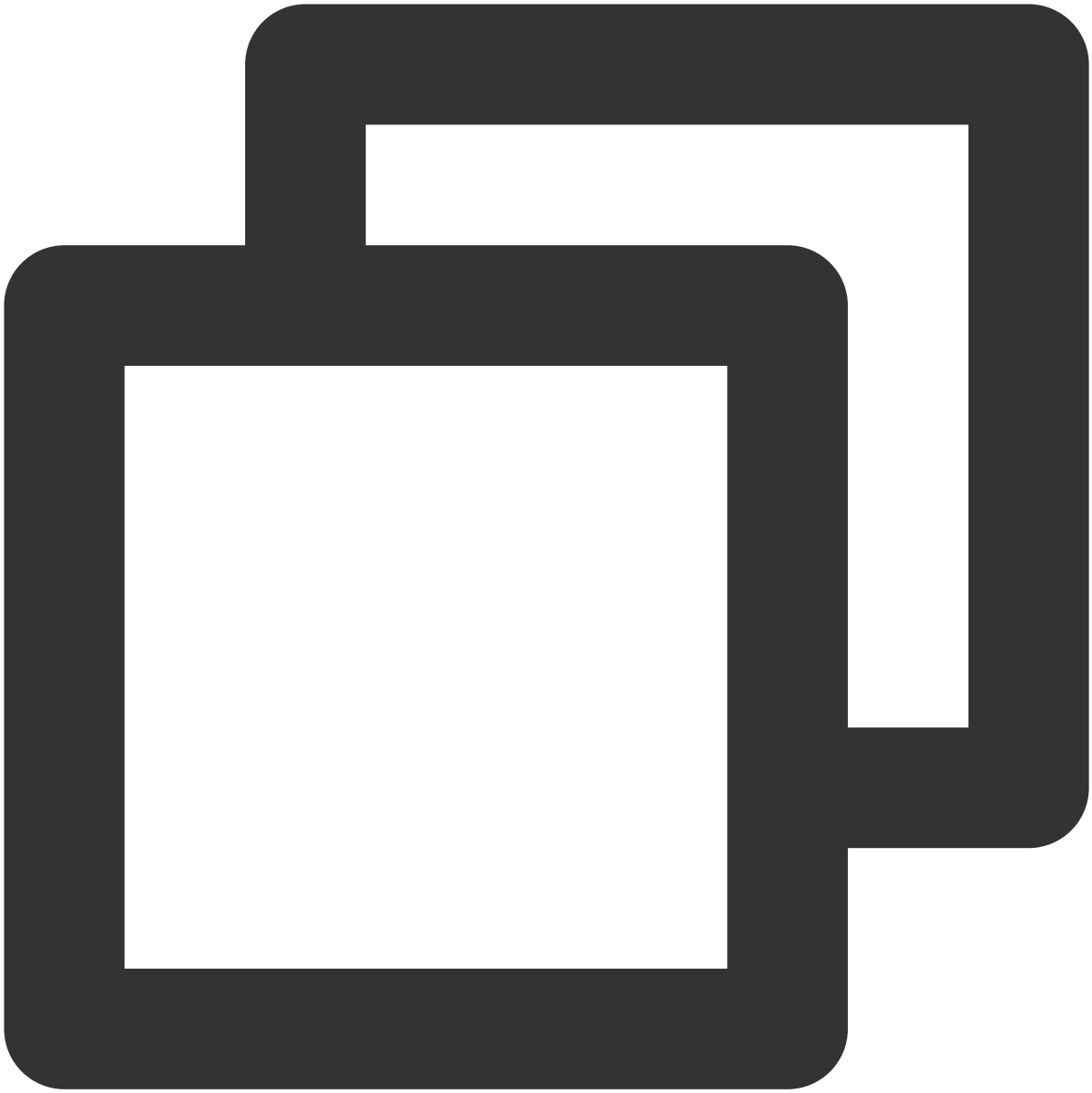
リモートユーザーによるマイクのオン/オフがあった場合に、このコールバックを受信します。



```
let handleUserAudioChange = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.USER_AUDIO_AVAILABLE, handleUserAudioChange);
```

USER_VOICE_VOLUME

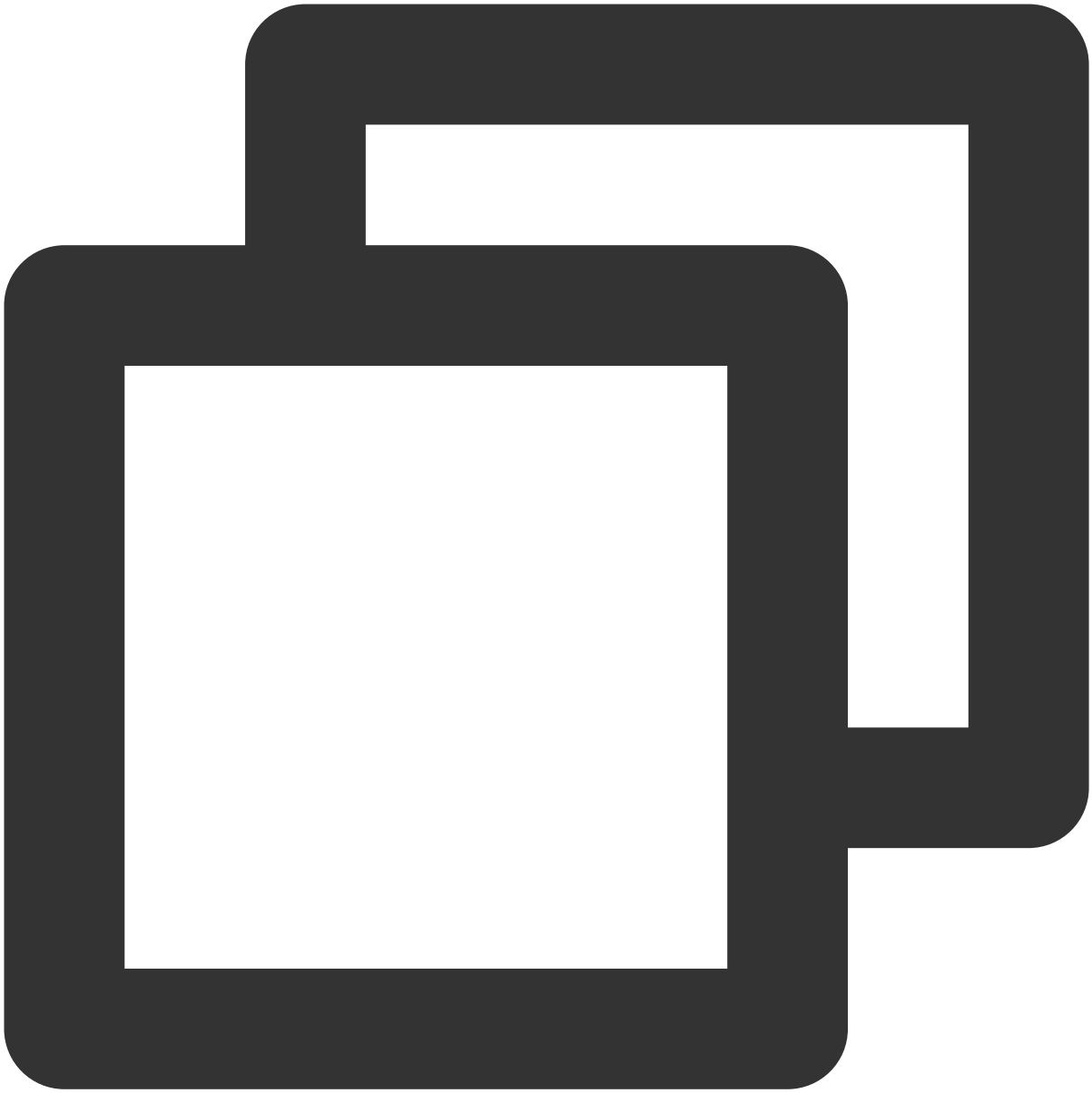
リモートユーザーがスピーカーの音量調整を行った場合に、このコールバックを受信します。



```
let handleUserVoiceVolumeChange = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.USER_VOICE_VOLUME, handleUserVoiceVolumeChange);
```

GROUP_CALL_INVITEE_LIST_UPDATE

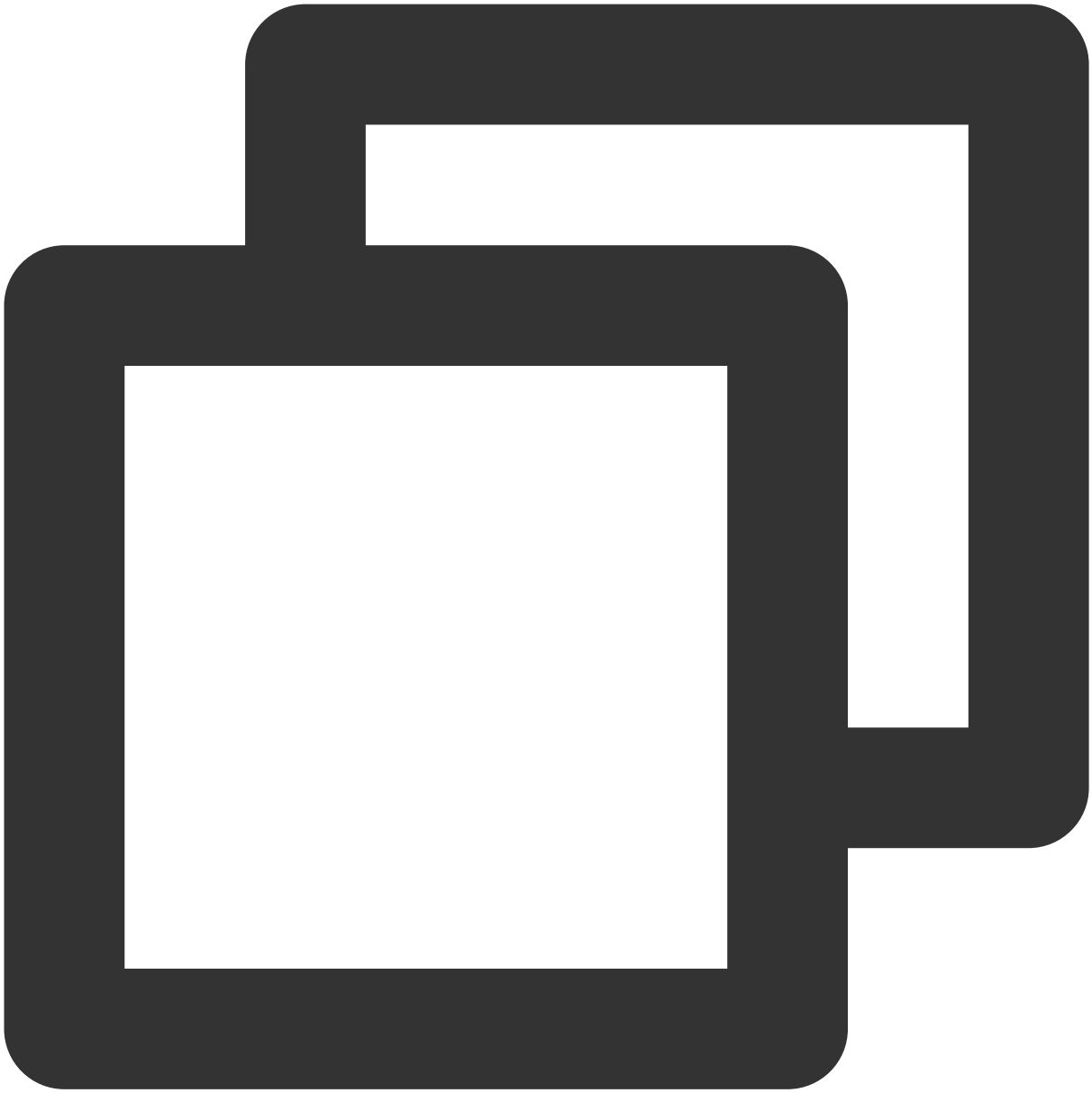
グループチャットで招待リストを更新するとこのコールバックを受信します



```
let handleGroupInviteeListUpdate = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.GROUP_CALL_INVITEE_LIST_UPDATE, handleGroupInviteeLis
```

INVITED

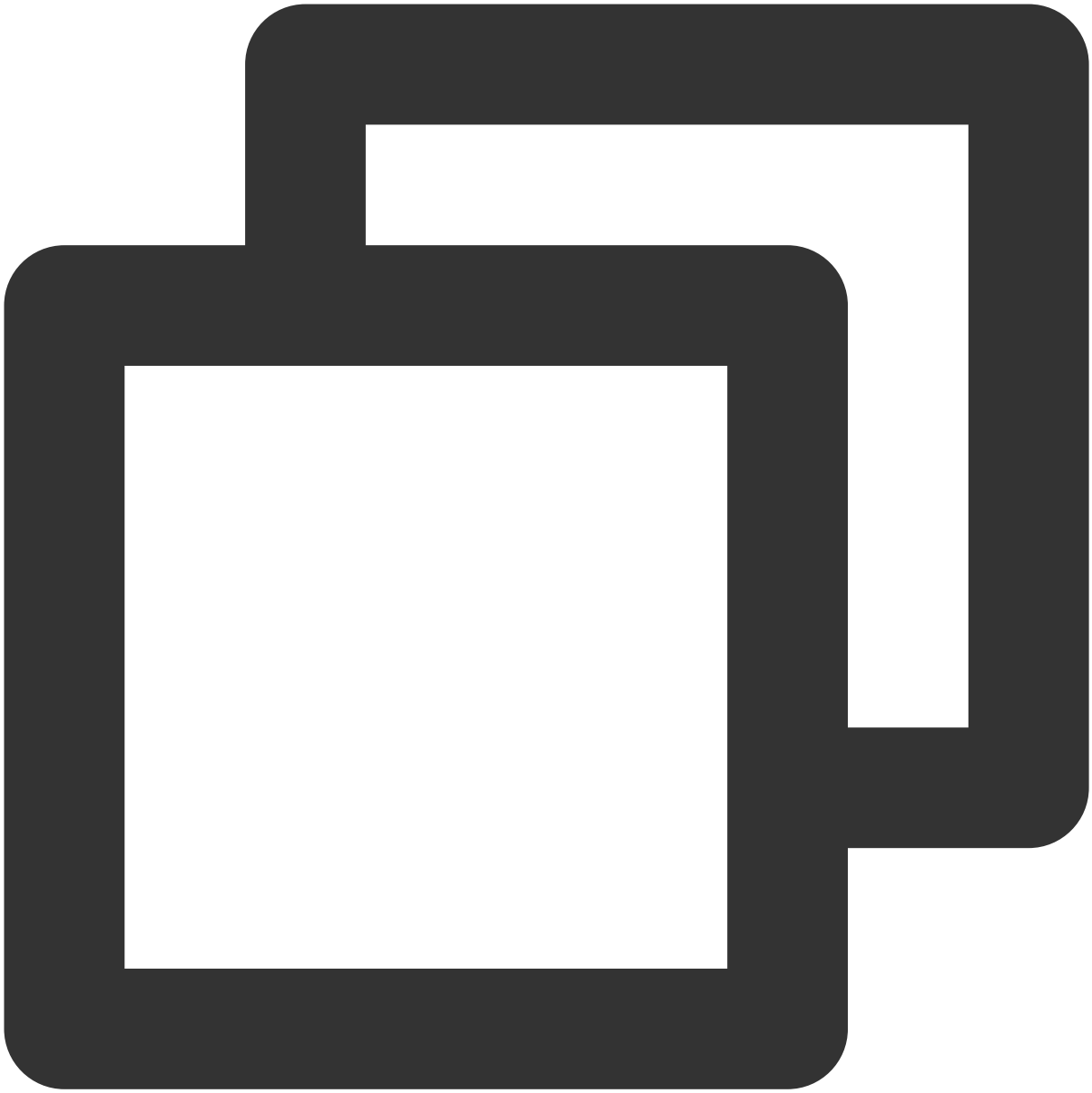
通話に招待されました。



```
let handleNewInvitationReceived = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.INVITED, handleNewInvitationReceived);
```

CALLING_CANCEL

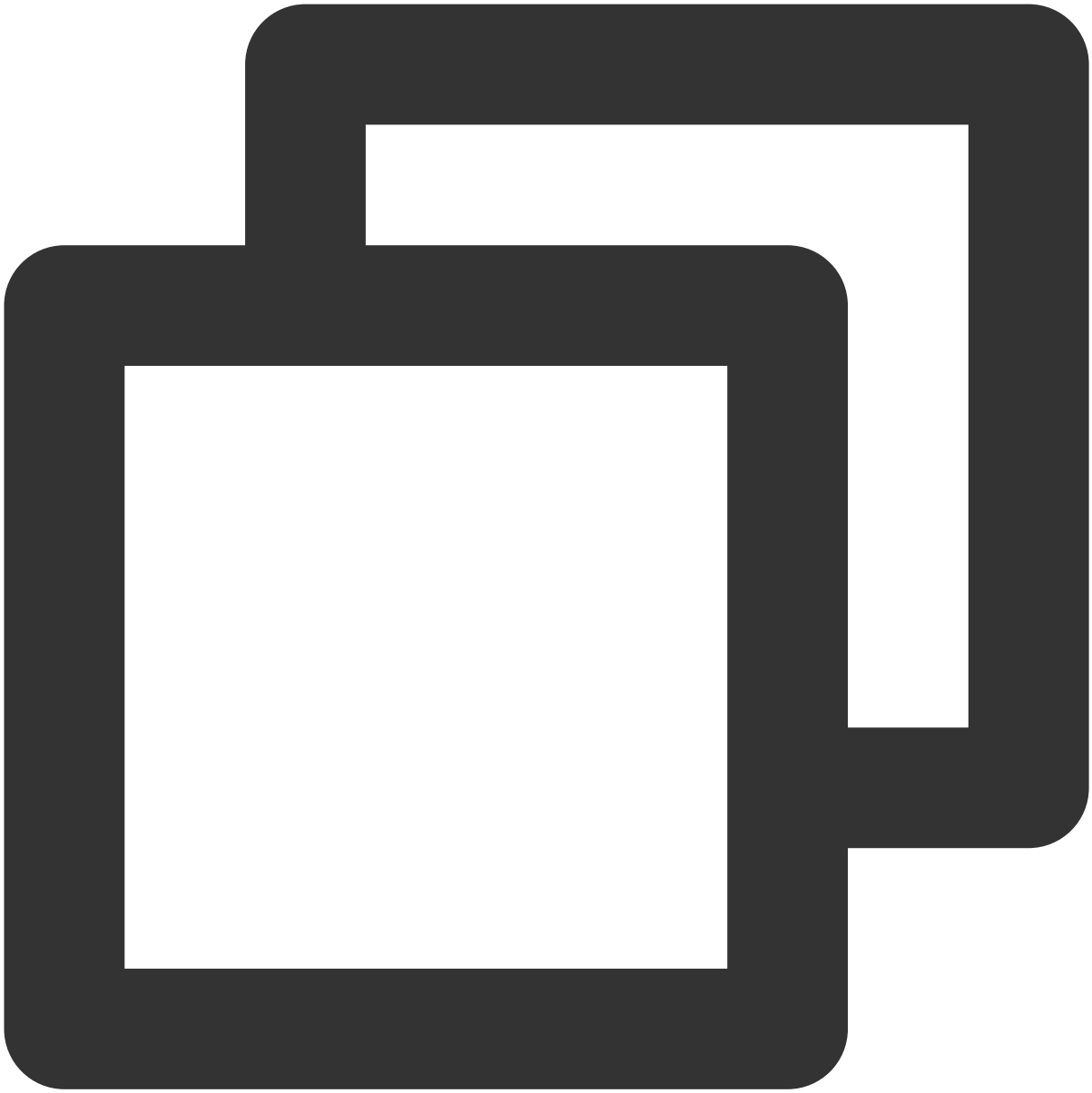
被招待者が受信します。このコールバックを受信した場合は、今回の通話がキャンセルされたことを意味します。



```
let handleCallingCancel = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.CALLING_CANCEL, handleCallingCancel);
```

CALLING_END

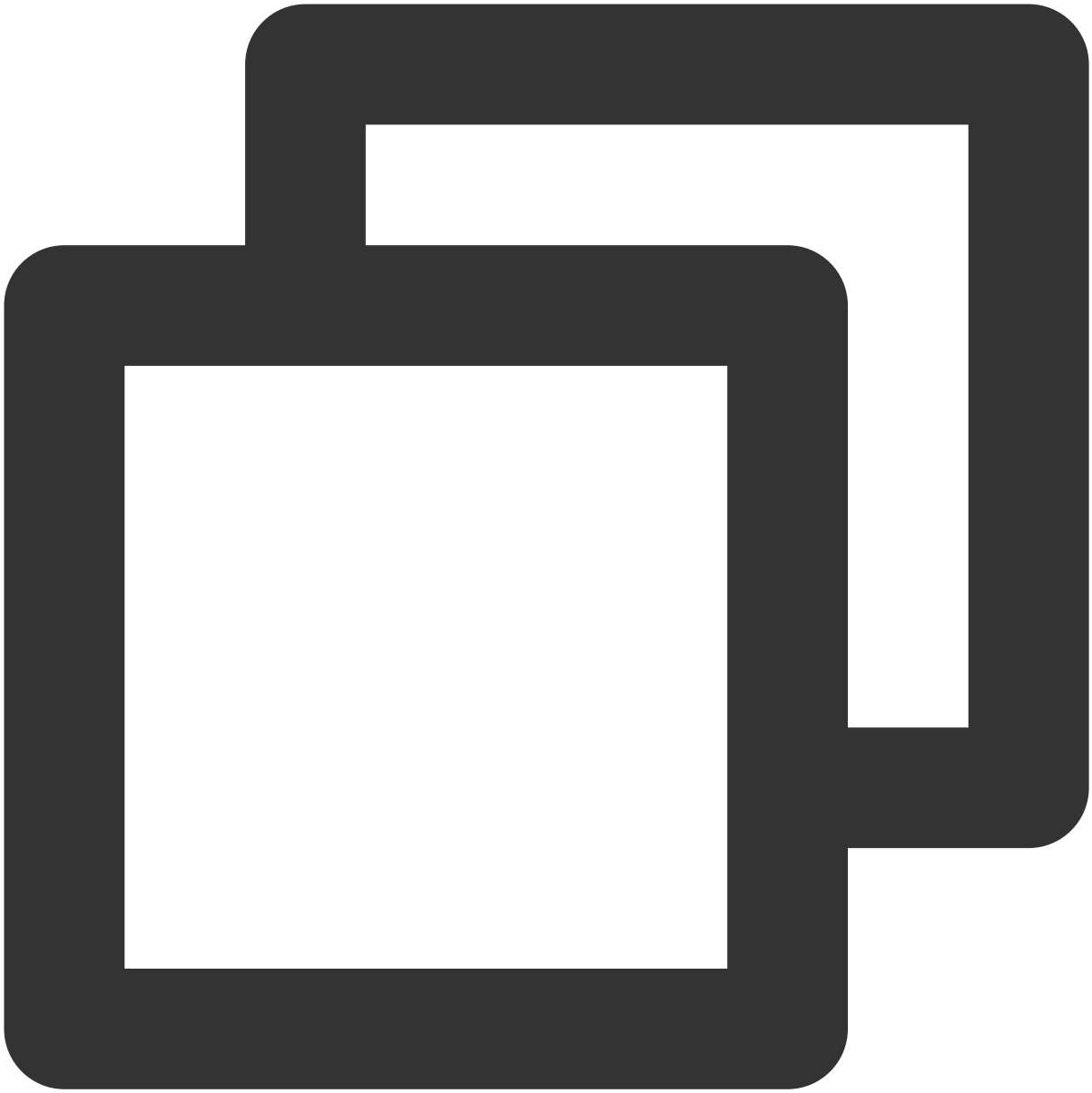
このコールバックを受信した場合は、今回の通話が終了したことを意味します。



```
let handleCallingEnd = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.CALLING_END, handleCallingEnd);
```

DEVICED_UPDATED

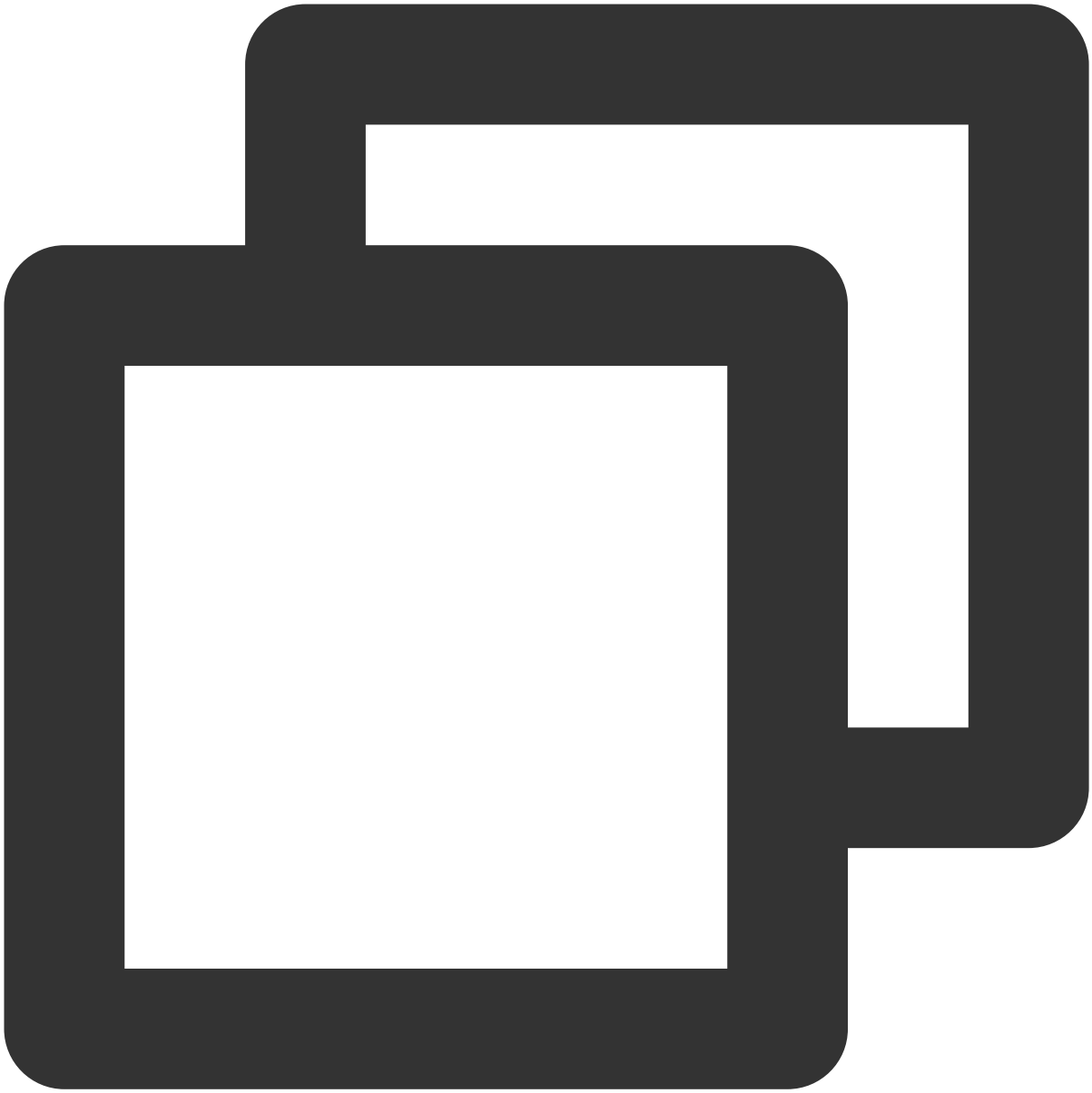
デバイスリストが更新された場合にこのコールバックを受信します。



```
let handleDeviceUpdated = function({ microphoneList, cameraList, currentMicrophoneI
  console.log(microphoneList, cameraList, currentMicrophoneID, currentCameraID)
};
tuiCallEngine.on(TUICallEvent.DEVICED_UPDATED, handleDeviceUpdated);
```

CALL_TYPE_CHANGED

通話タイプが切り替わった場合にこのコールバックを受信します。



```
let handleCallTypeChanged = function({ oldCallType, newCallType }) {
  console.log(oldCallType, newCallType)
};
tuiCallEngine.on(TUICallEvent.CALL_TYPE_CHANGED, handleDeviceUpdated);
```

公開ログ（TUICallKit）

Web

最終更新日：：2023-02-17 16:34:15

Version 1.3.1 @2022.11.29

注意

このバージョンはSDK `tuicall-engine-webrtc@1.2.1` バージョンに依存しています。速やかに更新してください。

追加

スタイルの詳細を最適化します。

相手が通話に応答しない場合、相手を監視して通話タイプを変更することをサポートしています。

basic demoにデバイス検査機能を追加しました。

修正

通話終了時の内部ロジックによるエラーを修正しました。

Version 1.3.0 @2022.11.14

注意

このバージョンにアップグレードする際に、ご参照ください：アップグレードガイドライン。

追加

携帯電話H5の使用時に、自動的に縦画面スタイルに切り替えることをサポートしました。

電話をかけた時にローカルカメラをプレビューすることをサポートしました。

basic demoに電話をかける前のデバイス検査を追加しました。

修正

`TUICallKitServer.destroy()` を呼び出した後にtimインスタンスが完全に終了しない問題を修正しました。

相手が通話中に応答なしと表示される問題を修正しました。

vite環境下でtypescriptタイプのパッケージ化が成功しない問題を修正しました。

インターフェースの変更

`TUICallKitServer.call()` または `TUICallKitServer.groupCall()` を自発的に呼び出した時に、エラーが発生した場合、 `beforeCalling` コールバックを呼び出さなくなります。直接try catchを使用してエラーを捕捉してください。

Version 1.2.0 @2022.11.03

追加

新しいバージョンのTUICallEngine SDKに適応します。

Version 1.1.0 @2022.10.21

追加

通話中に、通話ページをフルスクリーンにすることができます。

通話中に、 `<TUICallKitMini/>` を使用して最小化することができます。

修正

既知の問題を修正し、安定性を向上させました。

Version 1.0.3 @2022.10.14

追加

basic demoにUserIDのクイックコピーを追加したため、ワンクリックで新しいウィンドウを開くことができます。

Version 1.0.2 @2022.09.30

追加

アクセスドキュメントを最適化し、デモ画像および詳細のガイドを追加しました。

修正

初めて入室した時に、デバイスのステータスビットが無効になる問題を修正しました。

webpackツールをパッケージ化する時に、アイコンのロードに失敗することがある問題を修正しました。

既知のスタイルの問題を修正しました。

Version 1.0.1 @2022.09.26

追加

電話をかけた時に、相手のマイクアイコンが非表示になります。

修正

basic demo SDKAppID入力ボックスが数字になる問題を修正しました。

Version 1.0.0 @2022.09.23

TUICallKit demoクイックスタート
TUICallKit クイックアクセス
TUICallKit API
TUICallKitインターフェースのカスタマイズガイド
TUICallKit (Web) に関するよくあるご質問

Android & iOS

最終更新日：：2022-11-21 17:43:39

説明：
新バージョン（1.1.0.103）のTUICallEngineインターフェースに関連する変更：

- 開発中に、maven latest更新メカニズムが原因でAndroid構築エラーが発生した場合は、次の2つの方法で解決できます。
  - TUICallKitを最新バージョンにアップグレードする。
  - tuicallkit/build.gradleのtuicallengineを1.0の固定バージョン：`com.tencent.liteav.tuikit:tuicallengine:1.0.0.53`に依存するよう変更する。
- 開発中に、pod updateの実行が原因でiOS構築エラーが発生した場合は、次の2つの方法で解決できます。
  - TUICallKitを最新バージョンにアップグレードする。
  - Podfileファイルに`pod 'TUICallEngine'、'1.0.0.53'`を追加する。

Version 1.1.0.103 @ 2022.09.30

- Android&iOS：グループ通話中のグループメンバー招待機能を最適化しました。
- Android&iOS：通話フローで、通話応答前にレコーディング、審査などの料金が発生しないように改善しました。
- Android&iOS：カスタムオーディオビデオ通話のオフラインプッシュメッセージをサポートしました。
- Android&iOS：一部の**TUICallEngine**インターフェースパラメータを変更しました。詳細については、call()、groupCall()、inviteUser()、およびコールバックインターフェースonCallReceived()をご参照ください。
- Android&iOS：グループ通話中に、少量のコールバックの異常が発生することがある問題を修正しました。
- Android&iOS：重複ログインおよびUserSigの期限切れによるステータス異常の問題を修正しました。
- iOS：TUCallKit Objective-CとSwiftの混合コンパイルの際に、`init`インターフェースエラーが発生する問題を修正しました。
- Android：Kotlinでのフローティングウィンドウ機能統合の際にコンパイルエラーが発生する問題を修正しました。

Version 1.0.0.53 @ 2022.08.15

新規リリース：

- Android&iOS：1v1オーディオビデオ通話、グループオーディオビデオ通話をサポートしました。
- Android&iOS：主要メーカーのオフライン通知をサポートしました。
- Android&iOS：プロフィール画像のカスタマイズ、ニックネームのカスタマイズをサポートしました。

- Android&iOS：通話中のフローティングウィンドウ有効化をサポートしました。
- Android&iOS：カスタム着信音の設定をサポートしました。
- Android&iOS：複数のプラットフォームにログインした状態での着信サービスをサポートしました。