Tencent Real-Time Communication

UIを含まない統合ソリューション

製品ドキュメント

カタログ：

UIを含まない統合ソリューション
SDKのダウンロード
SDKのダウンロード

最終更新日：：2023-05-09 16:17:03

TRTC SDK
MLVB SDK
Player SDK
UGSV SDK
IM SDK
全機能SDK

機能リスト

機能	機能説明	一般的なユースケース
ビデオ通話	2人または多人数ビデオ通話です。720P、1080PのHD画質をサポートしています 1つのルームで最大300人のオンライン同時接続と、最大50人のカメラ同時起動をサポートしています	ビデオ通話、ビデオミーティング、フレンドとのビデオチャット、オンライン問診、ビデオカスタマーサービス、ビデオ面接など
音声通話	2人または多人数での音声チャットです。48kHz、ダブルサウンドチャンネルをサポートしています 1つのルームで最大300人のオンライン同時接続と、最大50人のマイク同時起動をサポートしています	1対1音声通話、多人数音声通話、ボイスチャット、音声インタラクティブゲームなど
ビデオインタラクティブストリーミング	キャスターと視聴者の双方向音声マイク接続をサポートしています キャスターのルーム間（ライブストリーミングルーム間）PKではスムーズなマイクのオン・オフをサポートし、切り替えプロセスで待つ必要がありません。キャスターの遅延は300ms未満です 1つのルームでのマイク接続人数は無制限で、最大50人のマイク同時接続をサポートしています 低遅延ライブストリーミングモードでは、視聴者10万人の同時再生をサポートし、再生の遅延を1000msまで低減させています	インタラクティブ授業、eコマースライブストリーミング、パーティーライブストリーミング、ライブストリーミングマイク接続、ライブストリーミングPK
ボイスインタラクティ	キャスターと視聴者の双方向音声マイク接続をサポートしています キャスターのルーム間（ライブストリーミングルーム間）PKを	双方向音声ポッドキャスト、音声インタラクティブゲーム、ボイス

ブストリーミング	サポートしています スムーズなマイクのオン・オフをサポートし、切り替えプロセスで待つ必要がありません。キャスターの遅延は300ms未満です 1つのルームでのマイク接続人数は無制限で、最大50人のマイク同時接続をサポートしています 低遅延ライブストリーミングモードでは、視聴者10万人の同時再生をサポートし、再生の遅延を1000msまで低減させています	チャットルーム、音声ライブストリーミングマイク接続、音声ライブストリーミングPK、カラオケルーム、ラジオFMなど

高度な機能

機能	機能説明	一般的なユースケース
インタラクティブマイク接続	インタラクティブなマイク接続をサポートしています。視聴者は自由かつスムーズにマイクをオン・オフでき、切り替えのプロセスを待つ必要がありません	インタラクティブライブストリーミング、オンライン授業、チャットルームなど
ルーム間PK	別名「ライブストリーミングルーム間PK」。複数のキャスターがルームを跨いでインタラクティブなPKを行い、視聴者はこれを視聴します	ショーライブストリーミング、PKマイク接続、クロスルーム授業など
画面共有	別名「スクリーンシェア」。ローカルコンピュータのデスクトップ、ウィンドウ、画面を他人と共有する機能をサポートしています（例：Microsoft PowerPointのPPTを再生するウィンドウ）	オンライン授業、PPT共有、リモート支援など
サーバーでのローカルレコーディング	サーバーでレコーディングするにはLinux SDKを使用する必要があります。現在Linux SDKはまだ全面的にはリリースされていません。お問い合わせまたは関連サービスの利用をご希望の場合は、colleenyu@tencent.comまでご連絡ください。	レコーディング、ファイル保存、コンプライアンスなど
クラウドレコーディング	Relayed Pushの方式を採用し、CSSの機能を利用して、全プロセスにおけるクラウドレコーディング機能（録音/録画）を提供します。さらにレコーディングしたファイルはVODプラットフォームに保存し、レコーディングプロセスの信頼性とリアルタイム性を保証します。	ダブルレコーディング、アーカイブ、コンプライアンスなど。
Cloud MixTranscoding	MCUクラスターを使用して、TRTCルーム内の各アップストリームのオーディオビデオストリーミングを必要に応じてミクスストリーミングトランスコードします。トランスコード後に出力されたオーディオビデオストリーミングはCloud Streaming ServicesへRelayed Pushされ、クラウドレコーディング、またはCDNによるライブストリーミング視聴を実現することが可能です	必要に応じたマルチ画面の合成、レコーディング形式の変換など

高音質	サンプルレート48kHz、フルリンク192kbpsの高音質、リアルな左右サウンドチャンネルステレオオーディオをサポートしており、ルームユーザーにクリアで没入感のあるインタラクティブな体験を提供します	音声通話、ビデオ通話、インタラクティブライブストリーミング、ボイスチャットルーム、高音質FM、音楽教室、カラオケルーム、オンライン授業など
高画質	720P、1080PのHD画質ビデオをサポートしています	ビデオ通話、インタラクティブライブストリーミング、オンライン授業など
3A処理	業界をリードするTencent Ethereal Audio Labが提供する3A処理アルゴリズムで、ダブルトーク、ノイズリダクションなどのシナリオにおいてより優れた音質を提供します。3Aとは、AEC（エコーキャンセル）、ANS（自動ノイズ除去）、AGC（自動利得制御）を指します	すべての音声シナリオ
AIノイズリダクション	AIノイズリダクションでは、従来のノイズ軽減では消しきれなかった音声も消去できます（例：咳、くしゃみ、自動車のクラクションなどの非定常な雑音）	音声通話、ビデオ通話、インタラクティブライブストリーミング、ボイスチャットルーム、オンライン授業など
ベーシック美顔	ベーシックな美顔機能をサポートしています。美白、美肌、肌色補正、ならびに基本的なフィルター効果の設定が含まれます	ビデオ通話、インタラクティブライブストリーミング、オンライン授業など
BGM	ローカルのMP3、AAC、WAVなどの形式のミュージックファイルを人の声のBGMにする機能をサポートしています	音声通話、ビデオ通話、インタラクティブライブストリーミング、インタラクティブ授業、ボイスチャットルーム、オンラインカラオケ、FMラジオなど
サウンドエフェクト	通話中に効果音を追加します（例：拍手、掛け声、口笛、ブーイングなど）	音声通話、ビデオ通話、インタラクティブライブストリーミング、ボイスチャット

		ルーム、カラオケルーム、FMラジオなど
伴唱・伴奏	ローカルで再生した音声を他の人に送ります（例：パソコン上のQQオーディオプレーヤーで再生した音声など）	インタラクティブライブストリーミング、オンライン授業、ボイスチャットルーム、FMラジオなど
ボイスチェンジ	ボイスチェンジのエフェクトを提供します（例：ロリータボイス、おじさん声、ヘビーメタルなどのボイスエフェクト）	音声通話、ビデオ通話、インタラクティブライブストリーミング、ボイスチャットルーム、カラオケルーム、FMラジオなど
リバーブ	リバーブ効果を提供します（例：カラオケ、小部屋、音楽ホール、浴室などのリバーブ効果）	音声通話、ビデオ通話、インタラクティブライブストリーミング、ボイスチャットルーム、カラオケルーム、FMラジオなど
音量の大きさのコールバック	波形のアニメーションまたはインジケータなどで表示できるように、音量の大きさの数値を提供しています	音声通話、ビデオ通話、ボイスチャットルーム、FMラジオ、カラオケルーム、人の声の検出など
インイヤーモニタリング	ローカルでレコーディングした音声をローカルのイヤホンで再生でき、自分で自分が発した声を聴くことができます。一般的に言い間違いの訂正や音程の正しさのチェックに利用します	インタラクティブライブストリーミング、ショーライブストリーミング、カラオケルームなど
オーディオデータのカスタマイズ	オーディオを自分でキャプチャするコールバックをサポートしています。開発者はオリジナルデータに対して処理を行い、カスタマイズ操作を行うことができます（例：外付けの非標準デバイス、オーディオファイルなど）	非標準デバイスの接続、オーディオエフェクトのカスタマイズ、音声処理、音声認識など
ビデオデータのカスタマイズ	カスタマイズしたビデオソースとレンダリングエンジンをサポートし、ビデオファイル、外付けデバイス、サードパーティのカスタムデータソースなど、カメラ以外のビデオソースを使用できるようにしています	美顔のカスタマイズ、データキャプチャソースのカスタマイズ、マルチデバイス管理、ビデオ認識、画像処理など

SEI情報	SEIフレームによってカスタマイズ情報をビデオストリームに埋め込み、他のユーザーにも同時に配信します（例：歌詞、タイトルなど）	カラオケルーム、QA形式のライブストリーミング、インタラクティブライブストリーミングなど

詳細機能

機能モジュール	機能項目	機能概要
ビデオキャプチャレコーディング	パラメータキャプチャ設定	解像度、フレームレート、オーディオサンプルレート、GOP、ビットレートなどの複数のパラメータキャプチャ設定をサポートし、さまざまなシナリオでの画面キャプチャのニーズを満たします
	アスペクト比	16:9、4:3、1:1などの複数のアスペクト比での撮影をサポートしています
	縦横画面	縦向き（portrait）、左側横向き（landscape left）および右側横向き（landscape right）の3方向のプッシュ送信をサポートしています
	解像度	SD、HD、FHDでの撮影をサポートしています
	フラッシュのサポート	フラッシュのオンまたはオフをサポートしています
	カメラの切り替えおよびズーム	撮影時のフロントカメラとリアカメラの切り替えおよびズーム機能をサポートしています
	自動および手動フォーカス	自動および手動フォーカス機能のオンまたはオフをサポートしています
	写真撮影のサポート	写真撮影をサポートしています
	イメージ	カメラキャプチャイメージとプッシュイメージをそれぞれ設定できます。フロントカ

		メラはデフォルトでイメージ機能を有効にしておく必要があります
	ウォーターマーク	撮影時のウォーターマーク追加をサポートしています
	スクリーンショットのサポート	スマートフォンのスクリーンショットをサポートしています
	ビデオ一時停止	ライブストリーミング中のオーディオまたはビデオの単独での一時停止をサポートしています
	フィルター	カスタムフィルターおよびフィルターの程度の設定をサポートしています
	ベーシック美顔	撮影時に顔の美肌、美白、肌色補正のベーシック美顔機能を設定します
オーディオキャプチャレコーディング	バックグラウンドミュージック	撮影前にローカルのMP3をBGMとして選択できます
	ミュージックの変調	ミュージックの変調をサポートしています
	音声ミキシング	ミュージックと人の声のミキシングをサポートし、それぞれの音量を調整できます
	インイヤーモニタリング	インイヤーモニタリング機能をサポートしています。インイヤーモニタリングとはイヤホンキャプチャのリスニングであり、デバイス上にイヤホン（一般のイヤホンまたはBluetooth対応イヤホン）を差し込むと、マシンのイヤホン側でそのデバイスのマイクがキャプチャした音声を聴くことができる機能です
	ステレオ音声	ダブルサウンドチャンネル技術をハイレベルに使用することで、出力音源の各位置と角度を仮想化し、ステレオ音声、3Dサラウンド、音の定位感などの効果を実現できます
	ノイズリダクション	エンジンにノイズ抑制機能が付属しています。音響心理学的モデルを取り入れることでS/N比を20dB以上向上させることができ、言語音質も損ないません
	ボイスチェンジ	撮影時にレコーディング音声のボイスチェンジを行います（ロリータボイス、おじさん声など）

	リバーブ	音声に対し特殊な処理を行い、ボイスチェンジ、リバーブと組み合わせることで、透き通るような声やロボットのような声など、さまざまなサウンドエフェクトをカスタマイズできます
	ミュート	プッシュ時にマイクをオフにし、ビデオ画面だけをプッシュする機能をサポートしています
	音量調整	SDKがマイク音量を自動調節します。遠近の集音に適し、安定した音量を維持します
CSSプッシュ	RTMPプッシュ	RTMP およびRTMPSプロトコルによるCSSプッシュをサポートしています。解像度は180P～1080Pをサポートします
	WebRTCプッシュ	WebRTCプロトコルベースのプッシュをサポートしています
	SRTプッシュ	SRTプロトコルベースのプッシュをサポートしています
	QUICプッシュ	QUICプロトコルベースのプッシュをサポートしています
	スクリーンキャプチャプッシュ（画面共有）	スクリーンキャプチャライブストリーミングをサポートしています。画面内容を共有します
	SEI機能のサポート	SEI（Supplemental Enhancement Information、メディア補足拡張情報）はストリーミングメディアチャネルを通じてテキスト情報とオーディオビデオコンテンツをパッケージ化し、キャスター側（プッシュ側）からプッシュし、視聴者側（プル側）で受信するもので、これによってテキストデータとオーディオビデオコンテンツの正確な同期を実現することを目的としています
	動的ビットレートプッシュ	ネットワーク状態に応じたプッシュビットレートの自動調整をサポートしています。複数のモードの設定をサポートし、ライブストリーミングをよりスムーズにします
	ピュアオーディオプッシュ	オーディオストリームのみをキャプチャしてプッシュを開始する機能をサポートしています。オーディオのみのシナリオの場合に帯域幅トラフィックを節約できます
	外部ソースプッシュ	外部のオーディオビデオデータストリームを入力してライブストリーミングを行うことができます
	プッシュ	プッシュのリンク失敗後の自動再接続をサポートしています

	自動再接続	
CSS再生	RTMP再生	RTMP形式の再生をサポートしています
	FLV再生	FLV形式の再生をサポートしています
	HLS再生	HLS形式の再生をサポートしています
	DASH再生	DASH形式の再生をサポートしています
	WebRTC再生	WebRTCプロトコルでの再生をサポートしています
ライブストリーミングマイク接続	インタラクティブマイク接続	キャスターと視聴者間の1vnビデオマイク接続インタラクションの実現に用います
	キャスターPK	キャスターとキャスター間の1v1ビデオPKの実現に用います
品質モニタリング	品質モニタリング	プッシュ・プルの状況に対する品質モニタリングをサポートしています

機能モジュール	機能項目	機能概要	Web	iOS & Android	Flutter
再生プロトコル/形式	VODまたはライブストリーミングをサポート	VOD再生とCSS再生を同時にサポートしています	✓	✓	✓
	サポートするCSS再生形式	RTMP、FLV、HLS、DASH、WebRTCなどのラ	WebRTC,FLV,HLS,DASH	RTMP,FLV,HLS,DASH	RTMP,FLV,HL

	イブストリーミングビデオ形式をサポートしています			
サポートするVOD再生形式	HLS、DASH、MP4、MP3などのVODオーディオビデオ形式をサポートしています	HLS,MP4,MP3,FLV,DASH	MP4,MP3,HLS,DASH	MP4,MP3,HLS
ライブイベントストリーミング	VOD再生とCSS再生を同時にサポートしています	✓	×	×
DASHプロトコルサポート	標準プロトコルのDASHによるビデオ再生をサポートしています	✓	✓	✓
Quicアクセラレーション	Quic通信プロトコルをサポートし、ビデオ伝送効率をアップします	-	✓	✓
SDR/HDRビデオ	SDRビデオおよびHDR	-	✓	✓

	10/HLG規格のHDRビデオの再生をサポートしています			
H.264再生およびソフトウェアデコードとハードウェアデコード	H.264ビデオソースの再生およびソフトウェアデコードとハードウェアデコードをサポートしています	✓	✓	✓
H.265ハードウェアデコード	H.265ビデオソースのハードウェアデコード再生をサポートしています	-	✓	✓
ピュアオーディオ再生	MP3などのファイルのオーディオのみの再生をサポートしています	✓	✓	✓
ダブルサウンドチャンネルオーディオ	ダブルサウンドチャンネルオーディオの再生をサポートしています	×	✓	✓

	Http Headerの設定	ビデオリソースをリクエストする際、HTTP Headersコンテンツをカスタマイズします	×	✓	✓
	HTTPSをサポート	HTTPSのビデオリソースの再生をサポートしています	✓	✓	✓
	HTTP 2.0	HTTP 2.0プロトコルをサポートしています	✓	✓	✓
再生パフォーマンス	事前ダウンロード	指定したビデオファイルのコンテンツの事前ダウンロードをサポートするとともに、事前ダウンロードするビデオファイルのサイズと解像度の設定をサポートしています。最初のフレー	✓	✓	✓

		ムの所要時間を大幅に短縮でき、さらに低消費電力に向けた最適化により、パフォーマンス向上につながります			
	再生とキャッシュの同時実行	再生中での後続コンテンツの同時キャッシュとダウンロードをサポートしています。ネットワーク使用量を減らし、キャッシュポリシーを設定することができます	✓	✓	✓
	正確なseek	プログレスバー上の指定位置にジャンプして再生する機能をサポートしています。モバイル端末	✓	✓	✓

		ではフレームレベル、Web端末ではミリ秒レベルの正確さを保証します			
	ダウンロード速度のリアルタイム取得	ダウンロード速度のリアルタイム取得をサポートしています。これにより、消費者市場ユーザーが業務上の必要性に応じてラグが発生している場合に、ダウンロード速度を表示することができます。また、アダプティブビットレート帯域幅予測モジュールを使用するための前提条件でもあります	✓	✓	✓
	マルチイ	1つの画面	✓	✓	✓

	ンスタンス	に複数のプレーヤーを追加して同時再生を行うことができます			
	ダイナミックフレームトラッキング	ラグが発生した場合、「早送り」のような方法で現在のライブストリーミングの進捗に追いつくことができ、ライブストリーミング画面のリアルタイム性を確保します	✓	×	×
再生制御	URL再生	オンラインビデオのURL方式による再生をサポートしています。URLは、オンデマンド再生アドレスまたはライブストリーミングのプルアド	✓	✓	✓

	レスにすることができます			
FileID再生	VODファイルの識別子、FileIDによるビデオ再生をサポートしています。これには、複数の解像度のビデオ、サムネイル、キーモーメントなどの情報が含まれます	✓	✓	✓
ローカルビデオ再生	ローカルに保存されたビデオファイルの再生をサポートしています	-	✓	✓
基本制御	開始、終了、一時停止、再開などの再生制御機能をサポートしています	✓	✓	✓
ピクチャーインピクチャー	ピクチャーインピクチャーに	✓	✓	✓

（ミニウィンドウ）再生	切り替えてミニウィンドウ形式で再生することができます。モバイル端末では、App内またはApp外でのピクチャーインピクチャー再生の統合を同時にサポートしています			
cache内seek	キャッシュされたビデオコンテンツをseek時にクリアせず、高速seekする機能をサポートしています	✓	×	×
CSSタイムシフト	CSSタイムシフトのビデオストリーム再生をサポートしており、開始、終了、および現在の	✓	✓	×

	サポート時間を設定でき、ドラッグに対応しています			
プログレスバーマークおよびサムネイルプレビュー	プログレスバーへのマーク情報の追加とサムネイル（スプライトイメージ）のプレビューをサポートしています	✓	✓	×
カバーの設定	再生するビデオのカバーの設定をサポートしています	✓	✓	✓
リプレイ	ビデオ再生終了後の手動でのリプレイトリガーをサポートしています	✓	✓	✓
ループ再生	ビデオ再生終了後の自動再生をサポートしています	✓	✓	✓
リスト再	ビデオリ	✓	✓	✓

生	スト内のビデオの連続再生や繰り返し再生をサポートしています。これはすなわち、ビデオリストの最後のビデオの再生が終了した後、リストの最初のビデオが再生されるということです			
中断からの再生再開	前回の再生終了位置からの再生開始をサポートしています	✓	✓	✓
再生開始時間のカスタマイズ	ビデオの再生開始時間のカスタマイズをサポートしています	✓	✓	✓
倍速再生	0.5~3倍の変速再生をサポートしています。変調せずにオーディ	✓	✓	✓

	オの変速を実現できます			
バックエンド再生	画面をバックエンドに切り替えてもオーディオおよびビデオの再生を継続できます	-	✓	✓
再生コールバック	再生ステータスコールバック、最初のフレームコールバック、再生完了または失敗コールバックをサポートしています	✓	✓	✓
再生失敗リトライ	再生失敗時の自動リトライ、ライブストリーミングの自動リトライ機能をサポートしています	✓	✓	✓
音量の設定	システム音量のリアルタイ	✓	✓	✓

	ム調整とミュート操作をサポートしています			
解像度の切り替え	ユーザーによるHLSビデオの複数の解像度ストリームのスムーズかつラグのない切り替えをサポートしています	✓	✓	✓
解像度の命名	さまざまな解像度ストリームのカスタム命名をサポートしています	✓	✓	✓
スクリーンキャプチャ機能	再生画面の任意のフレームのキャプチャをサポートしています	-	✓	x
プレビュー機能	プレビュー機能を有効にしたビデオ再生をサポートしています	✓	✓	x

	弾幕	ビデオ上方での弾幕表示をサポートしています	✓	✓	x
	字幕インポート	カスタム字幕ファイルのインポートをサポートしています	✓	×	×
ビデオセキュリティ	refererブラックリスト/ホワイトリスト	再生リクエストに含まれるRefererフィールドを介してリクエストの送信元を識別し、ブラックリストまたはホワイトリスト方式で送信元リクエストを制御する機能をサポートしています	✓	✓	✓
	Keyリンク不正アクセス防止	再生リンクへの制御パラメータの追加、リンクの有効時間、プレビュー時	✓	✓	✓

		間、再生が許可されるIP数などの制御をサポートしています			
	HLS暗号化	HLSに基づいて提供されるAES encryptionスキームをサポートし、キーペアを使用してビデオデータを暗号化します	✓	✓	✓
	HLSプライベート暗号化	VODのプライベートプロトコルによるビデオの暗号化をサポートし、暗号化後のビデオはPlayer SDKでのみ復号して再生できるようにします。さまざまなブラウザのプラグインやグレーツールによる	✓	✓	✓

		クラッキングを効果的に防止できます			
	商用DRM	Apple Fairplay、Google Widevineのネイティブ暗号化方式を提供しています	✓	✓	x
	セキュアなダウンロード	暗号化されたビデオのオフラインダウンロードをサポートした後は、ビデオはPlayer SDKを介してのみ復号と再生ができます	-	✓	✓
	動的ウォーターマーク	再生インターフェースに不規則に動くテキストウォーターマークを追加し、効果的な不正録画防止をサポー	✓	✓	x

		トしています			
	トレーサビリティウォーターマーク	不正録画者の追跡をわずかなコストでサポートしています	✓	✓	✓
エフェクトの表示	カスタムUI	SDKはUIを含む統合方式と、UIを含む共通の再生コンポーネントを提供していますので、ご自身のニーズに応じて選択することができます	✓	✓	✓
	画面の塗りつぶし	スクリーンサイズに合わせたビデオ画面の各種塗りつぶしモードの選択をサポートしています	✓	✓	x
	プレーヤーサイズの設定	プレーヤーの幅と高さのカスタマイズをサ	✓	✓	✓

	ポートしています			
ロール画像	一時停止した際の広告表示用のピクチャロール画像の追加をサポートしています	✓	✓	x
ビデオイメージ	水平、垂直方向などのイメージをサポートしています	✓	✓	x
ビデオの回転	角度に応じたビデオ画面の回転をサポートするとともに、ビデオファイルの内部rotateパラメータに基づいたビデオの自動回転をサポートしています	x	✓	x
画面のロック	回転のロックやインターフェース要素の非表示など、画面	-	✓	x

		のロック機能をサポートしています			
	明るさ調節	ビデオ再生時のシステムの明るさ調節をサポートしています	-	✓	✓
パッケージサイズ			-	Android： arm64：4.4 M armv7：4.2 M dex: 573 KB iOS： arm64：5.3M	-

機能モジュール	機能項目	機能概要	ライト版	標準版
インターフェース	カスタムUI	開発者のカスタムUIです。ミニビデオAppで完全なUIインタラクションのソースコードを提供しており、再利用またはカスタマイズが可能です。	✓	✓
キャプチャ・撮影	アスペクト比	16:9、4:3、1:1などの複数のアスペクト比での撮影をサポートしています。	✓	✓
	解像度	SD、HD、FHDでの撮影をサポートしています。カスタムビットレート、フレームレー	✓	✓

	ト、gopをサポートしています。		
撮影制御	撮影時のフロントカメラとリアカメラの切り替え、フラッシュの制御をサポートしています。	✓	✓
時間の設定	撮影の最短および最長時間をカスタマイズします。	×	✓
ウォーターマーク	撮影時のウォーターマーク追加をサポートしています。	×	✓
焦点距離	撮影時の焦点距離調節をサポートしています。	✓	✓
フォーカスモード	手動フォーカスおよび自動フォーカスをサポートしています。	✓	✓
セグメントレコーディング	撮影中にセグメントを一時停止し、削除することができます。	✓	✓
写真撮影	写真撮影をサポートしています。	×	✓
変速レコーディング	撮影時の低速および高速レコーディングをサポートしています。	×	✓
バックグラウンドミュージック	撮影前にローカルのMP3をBGMとして選択できます。	×	✓
ボイスチェンジおよびリバーブ	撮影前にレコーディング音声のボ	×	✓

	イスチェンジ（ロリータボイス、おじさん声など）およびリバーブ効果（KTV、ホールなど）を追加します。		
フィルター	フィルター効果のリアルタイムプレビューとスワイプ切り替えをサポートしています。カスタムフィルターおよびフィルターの程度の設定をサポートしています	✓	✓
ベーシック美顔	撮影時に顔の美肌、美白、肌色補正を設定し、強度を調節します。	✓	✓
プレミアム美顔	撮影時にデカ目、顔やせ、V顔、下あご調整、面長補正、小鼻効果を設定し、強度の調節もサポートします。	×	×
動的エフェクトステッカー	顔を認識した上で、変形、カバーステッカースタンプなどの効果を追加します。	×	×
AIトリミング	人の輪郭を識別し、背景を除去し、動的背景/PPTのような他の要素に置き換えます。	×	×
クロマキー	画面内の緑色の要素（緑一色の背景など）を除去し、	×	×

		動的背景/PPTのような他の要素に置き換えます。		
エフェクト編集	クイックインポート	Androidでビデオのクイックインポートをサポートしています。	✓	✓
	ビデオトリミング	指定した時間の範囲に従ってビデオを正確にトリミングします。	✓	✓
	ビットレート設定	ビットレートを指定してビデオを生成できます。	✓	✓
	カバーの取得	時間に基づいてフレーム画像を取得します。	✓	✓
	フレームごとのプレビュー	タイムラインの移動の際、基準カーソルが止まっている位置のフレーム画像をプレビューウィンドウに表示させます。	✓	✓
	フィルター	ビデオにフィルターを追加します。フィルターの強度の設定もサポートしています	×	✓
	タイムエフェクト	ビデオに逆再生、リピート、スローモーションのタイムエフェクトを追加します。	×	✓
	フィルターエフェクト	ビデオに幽体離脱、発光、分裂、幻影などのエフェクトを追加します。	×	✓

	バックグラウンドミュージック	付属の音声ファイルまたはユーザーのスマートフォンにあるローカルMP3をBGMとして選択できます。BGMのトリミングおよび音量レベルの設定をサポートしています。	×	✓
	動的または静的ステッカー	動的または静的ステッカーを追加します。ビデオ画面内の表示位置および開始時間の設定をサポートしています。	×	✓
	字幕	字幕を追加し、字幕枠の背景のスタイル（ポップオーバーなど）を選択できます。ビデオ画面内の表示位置および開始時間の設定をサポートしています。	×	✓
	画像のトランジション	複数の画像をインポートし、回転、フェードイン・フェードアウトなどのトランジション効果を選択してビデオを生成します。	×	✓
ビデオスプライシング	マルチビデオスプライシング	複数のビデオの前後のスプライシングをサポートしています。	×	✓
	マルチウィンドウ	1つのビデオ画面を2つに分割して、既存のビデオを再生	×	✓

		しながらのビデオ撮影をサポートします。		
ビデオのアップロード	VODへのアップロード	VODはメディア資産管理、コンテンツ審査などの機能をサポートしています。	✓	✓
ビデオ再生	VODプレーヤー	VODプレーヤーをベースにして実装した、ビデオ情報プル、縦横画面切り替え、解像度選択、弾幕、CSSタイムシフトなどの機能を一体化したソリューションであり、完全オープンソースです。	✓	✓
パッケージサイズ			Android： arm64：4.4 M armv7：4.2 M dex: 573 KB iOS： arm64：5.3M	

アカウントの機能

機能タイプ	機能説明
アカウントインポート	アカウント一括インポート
アカウントの無効化	UserSigの失効
アカウントの削除	アカウントの一括削除
ユーザーのオンラインステータス	オンラインおよびオフラインステータスの管理（ユーザーがログインしていることが前提）
アカウントの照会	アカウントがインポートされているかどうかの一括照会

複数端末ログイン

機能タイプ	機能説明
シングルプラットフォームログイン	Android、iPhone、iPad、Windows、Mac、Webは1種類のプラットフォームのみオンライン可能
デュアルプラットフォームログイン	Android、iPhone、iPad、Windows、Macは1つの端末がオンライン可能。Webは同時オンライン可能
トリプルプラットフォームログイン	Android、iPhone、iPadは1種類のプラットフォームがオンライン可能。Windows、Macは1種類のプラットフォームがオンライン可能。Webは同時オンライン可能
マルチプラットフォームログイン	Android、iPhone、iPad、Windows、Mac、Webは全プラットフォームが同時オンライン可能

メッセージタイプ

機能タイプ	機能説明
テキストメッセージ	メッセージの内容は通常のテキストです
画像メッセージ	メッセージの内容は画像URLアドレス、サイズ、画像サイズなどの情報です
顔絵文字メッセージ	顔絵文字メッセージは開発者がカスタマイズできます
音声メッセージ	音声データは秒単位で時間情報を提供する必要があります
地理的位置メッセージ	メッセージの内容は、地名、経度、緯度の情報です
ファイルメッセージ	メッセージの内容は、ファイルのURLアドレス、サイズ、形式などの情報であり、形式に制限はなく、最大100Mをサポートします
UGSVメッセージ	メッセージの内容は、ビデオファイルのURLアドレス、長さ、サイズ、形式などの情報で、最大100Mをサポートします
カスタムメッセージ	ラッキーマネーメッセージ、ジャンケン形式のメッセージなど、開発者がカスタマイズするメッセージタイプです

システム通知メッセージ	内蔵されているシステム通知メッセージと開発者がカスタマイズするシステム通知メッセージがあります
グループTipsメッセージ	メンバーのグループ入退出、グループの説明情報の変更、グループメンバープロファイルの変更があった場合などの、システムからの通知メッセージです
メッセージのマージ	最大300通のメッセージのマージをサポートします

メッセージ機能

機能タイプ	機能説明
メッセージのダウンロード	App管理者は、このインターフェースを介して、App内の過去7日間のすべてのシングルまたはグループメッセージ記録を取得できます
オフラインメッセージ	ユーザーがログイン後にバックエンドに戻り、その状態でメッセージを送信するユーザーがいた場合、IMはオフラインプッシュをサポートします
ローミングメッセージ	新しいデバイスにログインするとき、サーバー（クラウド）が記録した履歴メッセージのストレージを同期します。これはデフォルトで7日間保存され、有料で延長できます
複数端末同期	複数端末メッセージの同期で、メッセージを同時に受信できます
履歴メッ	ローカルメッセージ履歴とクラウドメッセージ履歴をサポートしています

セージ	
メッセージの取り消し	配信に成功したメッセージを取り消すと、デフォルトで2分以内にメッセージが取り消されます。取り消し操作は、シングルチャットおよびグループチャットメッセージのみでサポートしており、ライブストリーミンググループ(AVChatRoom)の取り消しはサポートしていません
開封確認	ピアツーピアセッションで相手の既読・未読のステータスを確認します
メッセージ転送	他のユーザーまたはグループへメッセージを転送します
@機能	グループ内の@メッセージと通常のメッセージに本質的な違いはありませんが、@の付いた人がメッセージを受信すると、UIで特殊な処理を行う必要があります
入力中	オンラインメッセージで実現できます
オフラインプッシュ	Apple APN、Xiaomiプッシュ、Huaweiプッシュ、Meizuプッシュ、OPPOプッシュ、vivoプッシュ、GoogleFCMプッシュをサポートしています
メッセージの削除	メッセージのremove方法を使用すると、メッセージをローカルで削除することができます
ラッキーマネー機能	ラッキーマネーメッセージは@メッセージに類似しており、TIMCustomElemによって実現できます
全メンバープッシュ	IM通信アーキテクチャに基づいて実現するREST APIのセットで、Appアプリケーションの全メンバープッシュ、タグプッシュ、属性プッシュなどのメッセージプッシュ要件のサポートに用います。クライアントはSDKオンラインプッシュ、オフラインプッシュ（Androidバックグラウンド通知およびAPNs）によってプッシュメッセージを受信できます
ロー	フレンドの検索、グループやグループメンバーの検索をサポートしています。メッセージの検索は

カルメッセージ検索	セッショングループごとに行います

プロファイル機能

機能タイプ	機能説明
ユーザープロファイルの設定	ユーザーが自分のニックネーム、検証方法、プロフィール画像、性別、年齢、署名、場所などのプロファイルを設定します
ユーザープロファイルの取得	ユーザーが自分やフレンド、知らない人に関するプロファイルを確認します
フィールドごとのユーザープロファイルの取得	特定のフィールドに従ってユーザープロファイルを取得します
カスタムユーザープロファイル	カスタムユーザープロファイルフィールドは最大20件までです

リレーションシップチェーン機能

機能タイプ	機能説明
フレンドを探す	ユーザーアカウントIDでフレンドを探すことができます
フレンドの追加申請	デフォルトで申請理由が必要かどうかを選択します。現在、デフォルトでは必要ありません
フレンドの追加	フレンド追加のリクエストを送信します
フレンドのインポート	単方向のフレンドの一括インポートをサポートしています
フレンドの更新	同じユーザーの複数のフレンドのリレーションシップチェーンデータの一括更新をサポートしています
フレンドの削除	フレンドにした後にフレンドを削除することができます
すべてのフレンドの取得	すべてのフレンドを取得します。デフォルトでは基本プロファイルのみを取得します
フレンドを同意/拒否	システムからフレンド追加のリクエスト通知を受け取った後、それを承認または拒否することができます
ユーザーをブラックリストに	任意のユーザーをブラックリストに追加します。それまでフレンドだった場

追加	合はフレンドシップが解除されます
ブラックリストから除外	ブラックリストからユーザーを除外します
ブラックリストの取得	ユーザーブラックリストをプルします
フレンドノート	フレンドになると、 フレンドノートを送ることができます
フレンドカスタムプロファイルの設定	フレンドカスタムフィールドは最大20個までです
フレンドグループの作成	グループを作成するときに、追加するユーザーを同時に指定したり、同じユーザーを複数のグループに追加したりできます
フレンドグループの削除	フレンドグループの削除
フレンドのチェック	フレンドシップの一括チェックをサポートしています
ブラックリストのチェック	ブラックリストの一括チェックをサポートしています
フレンドのグループ追加	フレンドをフレンドグループへ追加します
フレンドをグループから削除	フレンドをフレンドグループから削除します
フレンドグループのリネーム	フレンドグループのリネーム
指定したフレンドグループ情報の取得	指定したフレンドグループを取得します
すべてのフレンドグループの取得	すべてのグループ情報を取得します。また、すべてのフレンドを取得することで、グループ情報を取得することもできます
リレーションシップチェーンデータストレージ	SDKはリレーションシップチェーンデータをストレージできます
フレンドプロファイル変更のシステム通知	フレンドプロファイルが変更されると、システム通知を受け取ることができます
リレーションシップチェーン変更のシステム通知	リレーションシップチェーンが変更されると、システム通知を受け取ることができます

グループ機能

グループタイプ	グループ説明
フレンドワークグ	一般的なWeChatグループに似ています。作成後はすでにグループ内にいるフレンドのみを招待して参加させることができ、かつ被招待者側の同意またはグループマスターの承認は不要で

ループ	す
知らない人とのソーシャルグループ	QQグループに似ています。作成後はグループマスターがグループ管理者を指定できます。ユーザーはグループIDを検索してグループ参加申請を送信した後、グループマスターまたは管理者が申請を承認してからでないとグループに参加できません
臨時ミーティンググループ	作成後は自由に参加・退出でき、かつグループ参加前のメッセージを確認する機能をサポートしています。音声/ビデオミーティングのシナリオ、eラーニングのシナリオなどのTRTC製品と連携させたシナリオに適しています
ライブストリーミンググループ	作成後は自由に参加・退出ができ、グループ参加者数の上限はありませんが、メッセージ履歴の保存はサポートしていません。CSS製品との連携に適しており、弾幕チャットのシナリオに使用します
コミュニティ	作成後は自由に参加・退出ができ、最大100000人をサポートし、メッセージ履歴保存もサポートしています。ユーザーはグループIDを検索してグループ参加申請を送信した後、管理者の承認なしにグループに参加できます

機能/SDK	全機能版SDK	TRTC SDK	MLVB SDK	Player SDK	UGSV SDK	IM SDK
キャスター配信開始	✓	✓	✓	-	-	-
キャスターと視聴者のマイク接続/キャスターのルーム間PK	✓	✓	✓	-	-	-
ビデオのレコーディング編集/ビデオのアップロードと公開	✓	-	-	-	✓	-
オーディオビデオ通話	✓	✓	✓	-	-	-
CSS再生	✓	-	✓	✓	-	-
VOD再生	✓	-	-	✓	✓	-
セッションチャット	-	-	-	-	-	✓

リリースノート（App）

最終更新日：：2023-02-10 18:12:04

オーディオビデオ端末SDKにはTRTC SDK、MLVB SDK、Player SDK、UGSV SDKがあり、ユースケースに応じて、サブ製品版SDKまたは全機能版SDKを選択できます。

リリースノート10.8 @ 2022.10.27

ライブストリーミング機能モジュール

新機能

Android：V2TXLivePusherプッシュの際に、システム音声を共有できるようになりました。

機能の最適化

すべてのプラットフォーム：ライブイベントストリーミングプルの成功率を改善しました。
Android：プルの秒速開始にかかる時間を最適化しました。

バグの修正

すべてのプラットフォーム：V2TXLivePusherのプッシュの際に、脆弱なネットワーク環境で1101アラートがコールバックされない問題を修正しました。
すべてのプラットフォーム：TXLivePusher\\V2TXLivePusherで手動フォーカスが無効になる問題を修正しました。

UGSV機能モジュール

機能の最適化

Android&iOS：UGSVの編集中にBGMの再生にラグが発生する問題を改善しました。

バグの修正

Android：UGSVの編集中にBGMを繰り返し切り替えると、無音になる場合がある問題を修正しました。
Android：画像のトランジションの際、Huawei Mate50でブラックスクリーンになる問題を修正しました。
iOS：iOS 14システムバージョンで合成したビデオのビットレートが大きくなる問題を修正しました。

TRTC機能モジュール

新機能

すべてのプラットフォーム：オーディオエフェクトに「スクラッチ」を追加し、オンラインカラオケ体験をさらに充実させました。詳細はTXAudioEffectManager.setMusicScratchSpeedRateをご参照ください。

機能の最適化

Android：ビデオデコードの起動速度を最適化し、画面の秒速開始速度を最速で50msにまでアップしました。
すべてのプラットフォーム：NTP時刻の正確性を改善しました。詳細はTXLiveBase.updateNetworkTimeをご参照ください。

バグの修正

すべてのプラットフォーム：特定のシナリオ（オーディオビデオアップストリームがない場合）において、ミクスストリーミングロボットがTRTCルームにプッシュバックする場合に、プルの異常およびコールバックエラーが発生することがある問題を修正しました。
すべてのプラットフォーム：視聴者が入室後にロールを切り替える際、ネットワークタイプが変化することにより、オーディオビデオアップストリームが失敗することがある問題を修正しました。
すべてのプラットフォーム：ネットワークの切断と再接続の過程で音質切り替えが無効になる問題を修正しました。
すべてのプラットフォーム：ネットワークの切断と再接続の過程でアップストリームの音声が無音になることがある問題を修正しました。
Android & iOS：muteRemoteVideoStreamを呼び出した際に、ビデオ画面の最後の1フレームが削除される問題を修正しました。

プレーヤー機能モジュール

機能の最適化:

Android&iOS：ループ再生の1ループ終了にVOD_PLAY_EVT_LOOP_ONCE_COMPLETEイベントを追加しました。
Android：起動時にNetworkInfo.getExtraInfoを2回呼び出す問題を、コンプライアンスに従って改善しました。

バグの修正

Android&iOS：特殊なケースでプライベート暗号化ビデオの再生に失敗する問題を修正しました。
Android&iOS：一部のビデオをgzipで転送すると再生に失敗する問題を修正しました。
Android&iOS：再生終了後のプログレスバーの長さが合わない問題を修正しました。
iOS：appid&fileid再生でのv2プロトコルのビデオソースアドレス取得エラーの問題を修正しました。

機能リリース 10.7 @ 2022.09.20

ライブストリーミング機能モジュール

新機能

iOS&Android：TXLivePlayer\\V2TXLivePlayerのstartPlayメソッド名をstartLivePlayに変更し、Licenseチェックを強制的に有効化しました。Licenseの申請については、ビデオ再生Licenseをご参照ください。

機能の最適化

すべてのプラットフォーム：AudioJitterBufferのキャッシュポリシーを最適化しました。

バグの修正

すべてのプラットフォーム：V2TXLivePlayerのライブイベントストリーミングでプルする際、hev2オーディオ形式で再生した音声に異常が発生する問題を修正しました。
すべてのプラットフォーム：TXLivePusher\\V2TXLivePusherがRTMPプッシュする際に、IPプッシュを使用すると異常が発生する問題を修正しました。
Windows：C#にV2TXLivePlayerStatisticsコンストラクタが見つからず、コンパイルに失敗する問題を修正しました。
iOS：iPadの一部デバイスでのキャプチャ音量が小さい問題を修正しました。
Android：Bluetoothイヤホンを接続しているのに音声がスピーカー再生になることがある問題を修正しました。

UGSV機能モジュール

機能の最適化

Android&iOS：レコーディング中にオーディオミュートが発生する問題を最適化し、製品体験を向上させました。

バグの修正

Android：UGSVの前処理中に終了するとcrashが発生する問題を修正しました。
Android：プレビューのない状態でステッカーを設定すると無効になる問題を修正しました。
Android：逆再生状態でエフェクトを設定すると無効になる問題を修正しました。
Android：UGSV編集中にビデオ処理コールバックを設定すると無効になる問題を修正しました。
iOS：UGSV保存の際のオーディオビデオ同期の問題を改善しました。
iOS：オーディオのないビデオをスプライシングすると失敗する問題を修正しました。
Android&iOS：撮影を何度も行うとノイズが発生する問題を修正しました。

Tencent Real-Time Communication （TRTC）

新機能

すべてのプラットフォーム：クラウドミクスストリーミングで各チャネルの入力ストリームの音量を調整できるようになりました。詳細はTRTCMixUser.soundLevelをご参照ください。
すべてのプラットフォーム：onRemoteAudioStatusUpdatedコールバックインターフェースを追加し、リモートオーディオストリームの状態をより適切に認識および監視できるようになりました。

機能の最適化

すべてのプラットフォーム：エンコードカーネルをアップグレードし、画面共有シーンでの画質を向上させました。
すべてのプラットフォーム：脆弱なネットワーク下でのエンコードのコード制御効果を最適化しました。

バグの修正

iOS：iPadの一部デバイスでのキャプチャ音量が小さい問題を修正しました。
Android：Bluetoothイヤホンを接続しているのに音声がスピーカー再生になることがある問題を修正しました。
すべてのプラットフォーム：入退室を頻繁に行った場合にcrashが発生することがある問題を修正しました。

プレーヤー機能モジュール

機能の最適化

Android&iOS：VOD再生のstartPlayインターフェースをstartVodPlayに変更しました。
Android&iOS：ライブストリーミング再生のstartPlayインターフェースをstartLivePlayに変更しました。
iOS：長時間バックエンドに戻ってからプレーヤーに戻した場合に、再生が継続されない問題を修正しました。
Android：バージョンの低いAndroidシステムで、一部のビデオの再生に失敗する問題を修正しました。

リリースノート10.6 @ 2022.09.09

ライブストリーミング機能モジュール

新機能

iOS&Android&Mac：プロフェッショナル版TXLivePlayer\\V2TXLivePlayerは、HLS再生、アダプティブ再生およびマルチビットレートによるシームレスなストリームの切り替えをサポートします。

機能の最適化

すべてのプラットフォーム：Music音質下で音量が小さいという問題を最適化しました。
Android&iOS：通話音量での音漏れの問題を最適化しました。
Android：エコーリークが発生することがあるという問題を最適化しました。

バグの修正

すべてのプラットフォーム：V2TXLivePlayerがライブイベントストリーミングでプルする際、ネットワークが切断された状態から再開するときに直ちに再接続されないという問題を修正しました。
すべてのプラットフォーム：V2TXLivePlayerがライブイベントストリーミングでプルする際、UDPのリクエストが失敗し、TCPチャネルへのダウングレードが成功しないという問題を修正しました。
Mac：マイクを切り替えると、時折エコーキャンセルに失敗するという問題を修正しました。

UGSV機能モジュール

バグの修正

Android：UGSVコーデックを最適化し、Highコーデックで保存しました。
Android：BGM形式がプロンプトなしに対応していないという問題を修正しました。

iOS：低速で再生するとBGMにノイズが生じる問題を修正しました。

オーディオビデオ通話機能モジュール

機能の最適化

すべてのプラットフォーム：IPv6ネットワーク環境での入室速度を向上させました。
すべてのプラットフォーム：脆弱なネットワーク環境でのオーディオの復元効率および音声と映像の同期効果を最適化し、通話体験を向上させました。
すべてのプラットフォーム：脆弱なネットワーク環境での接続保持機能を最適化し、切断・再接続の確率を低減しました。
すべてのプラットフォーム：Musicレベル（startLocalAudioにおいて指定）での音量が小さいという問題を最適化し、ユーザー体験を向上させました。
Mac：Bluetoothヘッドセット使用時の通信体験を最適化し、ノイズをさらに減らし、よりクリアなサウンドを実現しました。
Android：ステレオキャプチャの互換性を最適化し、より多くのモデルをサポートしています。
Android：時折発生するエコーリークの問題を改善し、通信体験を向上させました。

バグの修正

Android & iOS：Speechレベル（startLocalAudioにおいて指定）で時折発生する音漏れの問題を修正しました。
Mac：マイクを切り替えると、時折エコーキャンセルに失敗するという問題を修正しました。

プレーヤー機能モジュール

機能の最適化

Android&iOS：fileid再生メソッドに、スプライトイメージやURLといった情報のコールバックが新たに追加されました。
Android&iOS：パッケージサイズを最適化しました。

バグの修正

iOS：一部のユースケースにおいて、プライベート暗号化ビデオのオフラインダウンロードや再生が失敗するという問題を修正しました。

全機能版10.5 @ 2022.08.24

ライブストリーミング機能モジュール

機能の最適化

Android：メモリヒープの発生を防止するために、ビデオデコードのメモリ管理を最適化しました。

Windows：内蔵マイクのノイズ低減効果を最適化し、中でもミュージックモードでのパフォーマンスをより改善しました。
Mac：マイクキャプチャをオンにした際、高確率で発生するノイズの問題を最適化しました。

バグの修正

すべてのプラットフォーム：V2TXLivePlayerがライブイベントストリーミングでプルする際、SEIが受信されないことがあるという問題を修正しました。
すべてのプラットフォーム：V2TXLivePlayerがライブイベントストリーミングでプルする際、タイムスタンプのロールバックによって無音になる問題を修正しました。

UGSV機能モジュール

バグの修正

Android：UGC HarmonyOSで画像トランジションにより生成されたビデオのスクリーンが緑色になる問題を修正しました。
Android：編集で生成されたビデオの長さがおかしくなる問題を修正しました。
Android：マルチサウンドチャンネルビデオが再生および再エンコードができないという問題を修正しました。
Android：動的光波のエフェクトが選択した時間帯に1回しか効果を発揮しないという問題を修正しました。
Android&iOS：UGSVレコーディングクリップを削除した後、BGMの再生進捗がおかしくなる問題を修正しました。

オーディオビデオ通話機能モジュール

機能の最適化

すべてのプラットフォーム：脆弱なネットワーク体験を向上させるため、qosポリシーを最適化しました。
iOS&Android：フルリンクの遅延を低減させ、インイヤーモニタリング体験を最適化しました。
Android：メモリヒープの発生を防止するために、ビデオデコードのメモリ管理を最適化しました。
Windows：内蔵マイクのノイズ低減効果を最適化し、中でもミュージックモードでのパフォーマンスをより改善しました。
Mac：マイクキャプチャをオンにした際、高確率で発生するノイズの問題を最適化しました。

バグの修正

すべてのプラットフォーム：頻繁に異なるルームに入室する際のコールバックイベント：OnUserVideoAvailableおよびOnUserAudioAvailableで、異常が発生することがある問題を修正しました。

プレーヤー機能モジュール

バグの修正

Android&iOS：ビデオフォーマットの拡張子がショートリンクでない場合、再生に失敗する異常を修正しました。

全機能版10.4 @ 2022.07.25

ライブストリーミング機能モジュール

新機能

iOS&Android：V2TXLivePlayerでは、再生終了時に最後のフレームを保持できるようになりました。

機能の最適化

すべてのプラットフォーム：TXLivePlayer\\V2TXLivePlayerがFLVでプルする際に、メモリが占有される問題を最適化しました。
Android：TXLivePlayer\\V2TXLivePlayerがプルする際に、ラグが発生することがある問題を修正しました。
Android：低遅延インイヤーモニタリングおよびダブルサウンドチャンネルキャプチャの互換性を最適化しました。
Android：ハードウェアデコードとソフトウェアデコードの切り替えポリシーを最適化し、デコードのパフォーマンスを向上させました。
iOS：iPadでのキャプチャ音量が小さい問題を改善しました。

バグの修正

Android：TXLivePlayer\\V2TXLivePlayerがプルする際に、ソフトウェアデコードに切り替わることがあるという問題を修正しました。

UGSV機能モジュール

機能の最適化

Android：UGSV編集のためのsetBGMLoopインターフェースを追加しました。

バグの修正

Android：ビデオsetWaterMarkが機能しないという問題を修正しました。
Android：TXVideoEditorのプレビューのレンダリングモードがおかしくなる問題を修正しました。

オーディオビデオ通話機能モジュール

新機能

iOS&Android：カスタムビデオキャプチャでRGBA32形式をサポートしました。詳細についてはsendCustomVideoDataをご覧ください。
Windows&Mac：ウォーターマーク設定でローカルプレビューをサポートしました。詳細についてはsetWaterMarkをご参照ください。

機能の最適化

Android：低遅延インイヤーモニタリングおよびダブルサウンドチャンネルキャプチャの互換性を最適化しました。
Android：ハードウェアデコードとソフトウェアデコードの切り替えポリシーを最適化し、デコードのパフォーマンスを向上させました。
iOS：iPadでのキャプチャ音量が小さい問題を改善しました。

バグの修正

すべてのプラットフォーム：入退室コールバックに異常が発生することがある問題を修正しました。
Windows：共有ウィンドウに切り替えると、新しいウィンドウの内容がトリミングされる問題を修正しました。

プレーヤー機能モジュール

機能の最適化

Android&iOS：HLSライブストリーミングは、アダプティブ再生をサポートしています。

バグの修正

Android：onNetStatusと進捗のコールバック間隔の異常を修正しました。
Android：プレーヤーがsetConfigを呼び出さないことで発生するNULLポインタの異常を修正しました。
iOS：一部のユースケースにおけるリプレイラグの問題を修正しました。

全機能版10.3 @ 2022.07.08

ライブストリーミング機能モジュール

新機能：
すべてのプラットフォーム：TXLivePlayer\\V2TXLivePlayerはHLS再生をサポートしています。
機能の最適化：
すべてのプラットフォーム：Music音質下の音声効果を最適化しました。
すべてのプラットフォーム：TXLivePlayer\\V2TXLivePlayerのSEI解析ロジックを最適化し、一部の非標準SEIと互換性を持たせました。
すべてのプラットフォーム：TXLivePlayer\\V2TXLivePlayerがFLV、RTMPでプルする際に、タイムスタンプのロールバックによって発生する音声と映像が同期しないという問題を最適化しました。
バグ修正：
すべてのプラットフォーム：TXLivePlayer\\V2TXLivePlayerで一部のライブイベントストリーミングのAAC-HEv2ストリームを再生する際における、音声の異常の問題を修正しました。
すべてのプラットフォーム：TXLivePlayerのビデオキャッシュの計算が不正確である問題を修正しました。

UGSV機能モジュール

バグ修正：

Android：ビデオレコーディングのsetZoomが機能しないという問題を修正しました。
Android：Samsung s22のレコーディングの失敗の問題を修正しました。
iOS：カスタムビデオの前処理がコールバックされないという問題を修正しました。

オーディオビデオ通話機能モジュール

新機能：
Windows：レコーディングにローカルレコーディング機能を追加し、インタラクティブライブストリーミングやオーディオビデオ通話の完全な内容をローカルでレコーディングできるようになりました。詳細についてはITXLiteAVLocalRecordをご参照ください。
Windows&Mac：startMicDeviceTestインターフェース内でマイクチェック時にマイクがキャプチャした音声の再生オン/オフをサポートするパラメータを追加しました。詳細についてはstartMicDeviceTestをご参照ください。
機能の最適化：
すべてのプラットフォーム：Music音質下の音声効果を最適化しました。
バグ修正：
すべてのプラットフォーム：ルームユーザーリストでコールバックにエラーが発生することがある問題を修正しました。
Windows：ビデオ再生中に画面がフリーズすることがある問題を修正しました。
Windows：ビデオ再生中に再生が失敗することがある問題を修正しました。
Windows：オーディオのユーザー定義キャプチャのシーンでエコーが発生する問題を修正しました。

プレーヤー機能モジュール

新機能：
iOS：ビデオ再生はピクチャーインピクチャーモードをサポートしています。
バグ修正：
Android：ハードウェアデコードのバックエンドで連続再生されるビデオリストが中断する問題を修正しました。
Android&iOS：Seek完了イベントがコールバックされないという問題を修正しました。

全機能版10.2 @ 2022.06.26

ライブストリーミング機能モジュール

新機能：
すべてのプラットフォーム：TXLivePlayer\\V2TXLivePlayerがプルする際に、Licenseチェックを追加しました。
すべてのプラットフォーム：V2TXLivePlayerがFLVでプルする際に、HTTP Headersの設定をサポートしています。
すべてのプラットフォーム：TXLivePusher\\V2TXLivePusherがRTMPプッシュする際に、オーディオコーデックパラメータを途中で変更できるようになりました。
機能の最適化：

すべてのプラットフォーム：V2TXLivePlayerがライブイベントストリーミングでプルする際、アダプティブビットレートのインターフェースを最適化しました。
すべてのプラットフォーム：V2TXLivePlayerがライブイベントストリーミングでプルする際、再接続に時間がかかりすぎる問題を最適化しました。
すべてのプラットフォーム：TXLivePlayer\\V2TXLivePlayerがFLV、RTMPでプルする際、ローカルキャッシュが少ないという問題を改善しました。
Android：TXLivePlayer\\V2TXLivePlayerがプルする際に、最初のフレームの秒速開始速度を最適化しました。
iOS：iOS SDKのボリュームを最適化しました。
iOS：LiteaVSDK Live版にTXLiveBase.hをパッケージしました。
バグ修正：
すべてのプラットフォーム：TXLivePlayerのラグ閾値の設定が機能しない問題を修正しました。
すべてのプラットフォーム：V2TXLivePusherがRTCプッシュする際に、オーディオビデオの最初のフレームのコールバックタイミングに異常がある問題を修正しました。
Android：TXLivePlayer\\V2TXLivePlayerがプルする際に、クイックstop、startをした場合、ブラックスクリーンが発生することがある問題を修正しました。

UGSV機能モジュール

新機能：
Android：オーディオトラックなしのビデオ編集をサポートしています。
機能の最適化：
Android：UGSVの編集・再生の初動速度を最適化しました。
バグ修正：
Android：レコーディングトリミング領域がおかしくなる問題を修正しました。
Android：H265ビデオのハードウェアデコードにおいて、フレームの幅と高さがおかしくなる問題を修正しました。
iOS：UGSVのトリミング時間が不正確な問題を修正しました。
iOS：iOS 14以降のモデルで発生する可能性のあるレコーディングのノイズの問題を修正しました。
iOS：ビデオ撮影の完了後に、レコーディングに戻るとクラッシュすることがある問題を修正しました。

オーディオビデオ通話機能モジュール

新機能：
すべてのプラットフォーム：よりフレキシブルで強力な機能のミクスストリーミングリレーAPIを新たにリリースしました。詳細についてはstartPublishMediaStreamをご参照ください。
すべてのプラットフォーム：3Dオーディオエフェクト機能を追加しました。詳細については enable3DSpatialAudioEffectをご参照ください。
すべてのプラットフォーム：人の声の検出機能を追加しました。muteLoalAudioとsetAudioCaptureVolume が0の場合は人の声の検出結果に影響を与えません。詳細についてはenableAudioVolumeEvaluationをご参照ください。
Tips：ユーザーのマイクオンの表示に便利です。

すべてのプラットフォーム：ロールの切り替え時の権限チェックサポート機能を追加しました。詳細については `switchRole(TRTCRoleType role, const char* privateMapKey)` をご参照ください。
iOS&Mac：前処理をカスタマイズしたC++インターフェースで、テクスチャ方式によるビデオのドッキング処理をサポートしました。
機能の最適化：
Android：インイヤーモニタリング効果を最適化し、遅延を低減しました。
Android：オーディオのキャプチャリンクを最適化し、一部モデルに存在するノイズの問題を解決しました。
iOS：アップストリームのMPSリンクを最適化し、CPU、GPUの占有率を低減しました。
Windows&Mac：ウィンドウ共有時のエンコード性能を最適化し、エンコードの幅と高さがキャプチャウィンドウのサイズの影響を受けないようにしました。
Windows：パフォーマンスを最適化し、メモリフラグメントおよびその割り当ての際に発生するパフォーマンスオーバーヘッドを減少させました。
バグ修正：
すべてのプラットフォーム：ネットワークタイプの切り替えの際に、アップストリームが失敗することがある問題を修正しました。
iOS：一部のiOS14システムでローカルレコーディングファイルに存在するノイズの問題を修正しました。

プレーヤー機能モジュール

機能の最適化：
Android&iOS：再生プロセスにおけるcachedBytes、IPアドレスなどのパラメータのコールバックを最適化しました。
バグ修正：
Android&iOS：H265形式ビデオのハードウェアデコード再生に失敗する問題を修正しました。
Android&iOS：HLSライブストリーミングを再生する際の異常を修正しました。
iOS：特定のユースケースにおけるsupportedBitratesを取得する際の異常を修正しました。

ログの発行(Electron)

最終更新日：：2022-10-31 14:42:13

Version 10.3.401 @ 2022.07.20

改善

- パフォーマンスを最適化しました。
- 下層データベースをアップグレードしました。

Version 9.3.201 @ 2022.01.05

機能追加

Windows & Mac：onSpeedTestResultネットワークスピードテストの結果のコールバックを追加しました。

改善

- Windows & Mac：startSpeedTestネットワークスピードテストの開始を改善しました。
- Windows & Mac：muteLocalVideoローカルのビデオストリームの公開の一時停止/再開を改善し、streamTypeパラメータを追加しました。
- Windows & Mac：muteRemoteVideoStream指定されたリモートビデオストリームの受信一時停止を改善し、streamTypeパラメータを追加しました。
- Windows & Mac：startScreenCapture画面共有の起動を改善し、paramsパラメータを追加しました。

問題の修正

- Mac：MacOS 12新システムでのカメラキャプチャの問題。
- Windows & Mac：脆弱なネットワークの制御ポリシーを最適化しました。同じシナリオでもよりスムーズです。
- Windows：AGCアルゴリズムを最適化して、音量の小さすぎ・大きすぎといった問題の発生確率を低減します。
- Winodws：画面共有時のフレームレート取得異常の問題を修正しました。

Version 8.9.102 @ 2021.08.11

機能追加

Windows & Mac：onStatisticsコールバックに新しいフィールドgatewayRttを追加しました onStatistics。

問題の修正

- Mac：特殊モデルのログ書き込みによって引き起こされるcrashを修正しました。
- Mac：マイク禁止操作を修正し、APIインターフェースsetAudioCaptureVolume(0)を使用した後、マイク検出音量が0であることが確認されました。
- Windows：性能を最適化し、カメラを起動後に画面が黒くなる現象を修正しました。
- Windows：解像度を自動的に下げた後、スクリーンキャプチャが復元されない問題を修正しました。
- Windows & Mac：その他のバグ修正。

Version 8.6.101 @ 2021.05.28

機能追加

- Windows & Mac：画面共有中のアプリケーションウィンドウの遮断をサポートするインターフェースの追加：addExcludedShareWindow、removeExcludedShareWindow、removeAllExcludedShareWindow。
- Windows & Mac：共有可能なウィンドウリストインターフェースgetScreenCaptureSourcesを取得し、戻り値リスト要素に新しいisMinimizeWindowフィールドを追加します。
- Windows & Mac：パラメータを渡すコンストラクタをサポートします。

問題の修正

- Windows：プラグインのロードが中国語パスをサポートしていないという問題があります。
- Windows & Mac：webgl context lostの問題を修正しました。
- Windows & Mac：デュアルチャネルエンコーディングをオンにし、ルームに参加した後、小画面のビデオストリームに切り替えると、ローカルに表示されているリモート参加者の画面が動かなくなります。
- Windows & Mac：クライアントがルームに参加してストリームをプルする際、リモート参加者の画面がぼやけた後、徐々に鮮明になる問題が発生します。

Version 8.4.1 @ 2021.03.26

機能追加

- Mac：Macオペレーティングシステムの音声出力キャプチャ機能のサポートを開始しました startSystemAudioLoopbackこれは、Windows側と同じSystemLoopback機能です。この機能により、SDKは現在のシステム音声をキャプチャすることができます。この機能をオンにすれば、キャスターは音楽や映画ファイルを他のユーザーに簡単にライブストリーミングすることができます。
- Mac：システムオーディオキャプチャコールバック onSystemAudioLoopbackError、システムオーディオドライバーの実行ステータスを取得できます。
- Mac：画面共有ではローカルプレビュー機能のサポートを開始しました。画面共有のプレビュー内容を、小さなウィンドウを介してユーザーに表示することができます。

- すべてのプラットフォーム：美顔プラグインメカニズムをサポートします。

品質の最適化

- すべてのプラットフォーム：Musicモードの音質を最適化し、cloubhouse等の音声ライブストリーミングケースに更に適するようになりました。
- すべてのプラットフォーム：オーディオビデオリンクのネットワーク耐性を最適化しました。著しく劣るネットワーク環境でも、そのうち70%のオーディオビデオは比較的スムーズなままです。
- Windows：一部のシーンでのライブストリーミングの音質を最適化し、音声障害の問題を大幅に減少させました。
- Windows：パフォーマンスを最適化しました。一部のユースケースでパフォーマンスが旧バージョンより20%～30%向上しました。

問題の修正

- Mac：Mac mini (m1)で全画面共有に切り替えてから特定ウィンドウに切り替えた後も、リモート側では依然として全画面共有ウィンドウが表示される問題を修正しました。
- Mac：Macでは画面共有がハイライト表示されない問題を解決しました（Macシステム11.1、10.14.5では緑の枠が表示されず、Macシステム10.3.2では緑の枠が表示されますが、拡大されたウィンドウがちらついていました）。
- Mac：Mac mini m1が画面共有リストを取得するとcrashし、最下層のsourceNameがnullの場合、上位層が「」を返される問題を修正しました。
- Mac：Mac mini m1でgetCurrentMicDeviceがcrashを引き起こし、(sourceName)が空になる恐れがあるという問題を修正しました。
- Windows： Windows Server 2019 Datacenter x64システムでデスクトップ共有を開始するとcrashする問題を修正しました。
- Windows：共有ウィンドウでターゲットウィンドウのサイズを同時に変更すると、偶発的に共有が停止するBUGを修正しました。
- Windows：一部のモデルのカメラで画面がキャプチャされない問題を修正しました。

Version 8.2.7 @ 2021.01.06

追加

- Windows & Mac：ルーム切り替えのため、switchRoomを追加しました。
- Windows & Mac：ローカル画像（プライマリストリーム）のレンダリングパラメータを設定するため、setLocalRenderParamsを追加しました。
- Windows & Mac：リモート画像のレンダリングパラメータを設定するため、setRemoteRenderParamsを追加しました。
- Windows & Mac： バックグラウンドミュージックの再生を開始するため、startPlayMusicを追加しました。

- Windows & Mac： バックグラウンドミュージックの再生を停止するため、stopPlayMusicを追加しました。
- Windows & Mac： バックグラウンドミュージックの再生を一時停止するため、pausePlayMusicを追加しました。
- Windows & Mac： バックグラウンドミュージックの再生を再開するため、resumePlayMusicを追加しました。
- Windows & Mac：バックグラウンドミュージックファイルの合計継続時間をミリ秒単位で取得するため、getMusicDurationInMSを追加しました。
- Windows & Mac：バックグラウンドミュージックの再生の進行状況を設定するため、seekMusicToPosInTimeを追加しました。
- Windows & Mac：バックグラウンドミュージックの音量を設定するため、setAllMusicVolumeを追加しました。バックグラウンドミュージックの再生・ミキシング時に使用し、バックグラウンド音声の音量を制御するために用います。
- Windows & Mac：バックグラウンドミュージックのローカル再生音量を設定するため、setMusicPlayoutVolumeを追加しました。
- Windows & Mac：バックグラウンドミュージックのリモート再生音量を設定するため、setMusicPublishVolumeを追加しました。
- Windows & Mac：ルーム切り替えコールバックのため、onSwitchRoomを追加しました。
- Windows & Mac：リモートユーザーの再生音量を設定するため、setRemoteAudioVolumeを追加しました。
- Windows & Mac：ビデオ画面をキャプチャするため、snapshotVideoを追加しました。
- Windows & Mac：キャプチャの完了時のコールバックのため、onSnapshotCompleteを追加しました。

改善

- Windows & Mac：enterRoomとswitchRoomは、stringタイプのstrRoomIdをサポートします。
- Windows & Mac：その他のバグ修正。

Version 7.9.348 @ 2020.11.12

改善

- Windows：録音パス設定が中国語のパスファイルをサポートしていないという問題を修正しました。
- Windows：ウィンドウキャプチャの指定エリアキャプチャは、アンチオクルージョンをサポートします。

Version 7.8.342 @ 2020.10.10

追加

- Windows & Mac：現在のオーディオキャプチャデバイスの音量変化のコールバックのため、onAudioDeviceCaptureVolumeChangedを追加しました。

- Windows & Mac：現在のオーディオ再生デバイスの音量変化のコールバックのため、onAudioDevicePlayoutVolumeChangedを追加しました。

Version 7.7.330 @ 2020.09.11

追加

Windows & Mac：オーディオ品質を設定するため、setAudioQualityを追加しました。

改善

- Windows：ローエンドのカメラでのCPU使用率が高すぎる問題を最適化しました。
- Windows： 複数のUSBカメラおよびマイクの互換性を最適化して、デバイスのアクティブ化の成功率を向上させました。
- Windows：カメラおよびマイクデバイスの選択ポリシーを最適化して、カメラまたはマイクを使用中に抜き差しすることでキャプチャに不具合が生じる問題を回避させました。
- Windows & Mac：その他のバグ修正。

Version 7.6.300 @ 2020.08.26

追加

Windows & Mac：PCのマイクとスピーカーを制御するため、setCurrentMicDeviceMute 、getCurrentMicDeviceMute、setCurrentSpeakerDeviceMute、getCurrentSpeakerDeviceMuteを追加しました。

Version 7.5.210 @ 2020.08.11

改善

- Windows & Mac：SDKコールバックが順不同になる問題を修正しました。
- Windows & Mac：レンダリングモードの切り替えによって引き起こされるクラッシュの問題を解決しました。
- Windows & Mac：一部の解像度でレンダリングが失敗する問題を修正しました。
- Windows & Mac：その他のバグ修正。

Version 7.4.204 @ 2020.07.01

改善

- Windows：Windowsプラットフォームでのエコーキャンセル（AEC）の効果を最適化しました。

- Windows：Windowsプラットフォームでのカメラキャプチャのデバイスの互換性を強化しました。
- Windows：Windowsプラットフォームでのオーディオデバイス（マイクとスピーカー）のデバイスの互換性を強化しました。
- Windows：WindowsバージョンのonPlayAudioFrameコールバックのUserIDが不正確である問題を修正しました。
- Windows：64ビットバージョンではシステムミキシングがサポートされています。

Version 7.2.174 @ 2020.04.20

改善

- Mac：Macでローカルカスタムレンダリング解像度の一貫性が失われることがある問題を修正しました。
- Windows：Windows側のgetCurrentCameraDeviceのロジックを最適化しました。カメラを使用しない場合は、最初のデバイスがデフォルトのデバイスとして返されます。
- Windows：画面を共有すると、ハイライト表示されたウィンドウがグレースクリーンとして表示される問題を修正しました。
- Windows：画面共有サムネイルを取得するときにWin10システムがスタックすることがある問題を修正しました。
- Windows & Mac：ロールの切り替え時に、カスタムストリームIDがタイミングよく有効にならないことがある問題を修正しました。
- Windows & Mac：画面共有設定のエンコードパラメータが有効にならない問題を修正しました。
- Windows：Windows側で画面を共有した場合、webrtcが画面を表示するのに時間がかかる問題を修正しました。

Version 7.1.157 @ 2020.04.02

追加

ビッグストリームを使用した画面共有をサポートします。

改善

- ミクスストリーミングプリセットテンプレートの使いやすさを最適化しました。
- ミクスストリーミングを最適化して、成功率を向上させました。
- Windowsの画面共有を最適化しました。

Version 7.0.149 @ 2020.03.019

追加

typescriptを使用する開発者向けに、trtc.d.tsファイルを追加しました。

ログの発行（Web）

最終更新日：：2023-02-20 15:54:12

バージョン番号「major.minor.patch」の具体的なルールは次のとおりです。
major：メジャーバージョン番号。メジャーバージョンのリファクタリングがある場合、このフィールドはインクリメントされます。通常、メジャーバージョン間のインターフェースには互換性がありません。
minor：マイナーバージョン番号。マイナーバージョン番号間のインターフェースには互換性があります。新しいインターフェースまたは最適化されたインターフェースがある場合、フィールドはインクリメントされます。
patch：パッチ番号。機能の改善または欠陥の修正がある場合、このフィールドはインクリメントされます。
注意
製品の安定性を高め、より良いオンラインサポートをご利用いただくため、速やかに最新バージョンに更新することをお勧めします。
バージョンアップに関する注意事項については、アップグレードガイドラインをご参照ください。

Version 4.15.2 @2022.12.30

Improvement

再接続ロジックを最適化しました。
入室パラメータ検証ロジックを最適化しました。

Version 4.15.1 @2022.12.09

Feature

クラウドミクスストリーミングおよびリレーCDNインターフェースはサブストリームをサポートしています。

Improvement

スモールストリームのレンダリングロジックを最適化し、スモールストリームのアスペクト比がラージストリームと一致しないことがある問題を修正しました。
切断・再接続ロジックを最適化し、切断・再接続後にサブストリームが失われ、キックアウトされることがある問題を修正しました。
switchDeviceのロジックを最適化し、デバイス切り替え後にアップストリームのプッシュに異常が発生する問題を修正しました。

Version 4.15.0 @2022.11.21

Feature

サブストリームのプッシュをサポートしました。1つのClientでカメラ+画面共有の同時プッシュが可能です。画面共有チュートリアルをご参照ください。
スペースオーディオをサポートしました。スペースオーディオの実装をご参照ください。
LocalStream.removeTrackはremove audioTrackをサポートしました。

Bug Fixed

リモートユーザーがスモールストリームを有効にしたとき、publish & unpublishが頻繁に発生し、プルが異状になることがある問題を修正しました。

Version 4.14.7 @2022.11.04

Improvement

MacOS VenturaのSafari 16で画面共有すると、ブラックスクリーンが発生する問題を回避しました。

Version 4.14.6 @2022.10.28

Improvement

デュアルストリームを有効にすると同時に、localStreamでtrackを操作するインターフェースを呼び出します(addTrack/removeTrack/replaceTrack/switchDevice)。
デバイスキャプチャロジックを最適化しました。指定のdeviceIdに対応するデバイスが利用できない場合、ブラウザは自動的に利用可能なデバイスを選択してキャプチャします。
デバイスのキャプチャ回復ロジックを最適化しました。

Version 4.14.4 @2022.09.30

Improvement

Mac Firefoxのカメラが352×288より低い解像度をキャプチャするとエラーが報告される問題を回避しました。Firefox issue。
タスクのスケジューリングを最適化し、リソースの消費を削減します。

Bug Fixed

localStream.replaceTrackを使用してvideoTrackを置換した後に、カメラを抜き差しするとvideoTrackが無効になることがある問題を修正しました。

Version 4.14.3 @2022.09.09

Bug Fixed

厳格モードで、Chromeのプライバシーモードがデバイスリストの取得に失敗することがある問題を修正しました。
ページにインタラクションが生成されていない状況で入室すると、音量が異常に取得される問題を修正しました。

Version 4.14.2 @2022.08.26

Feature

TRTC.getCamerasおよびTRTC.getMicrophoneで取得したデバイスリストは、ネイティブインターフェースgetCapabilitiesの呼び出しをサポートしています。

Improvement

iOS端末の（着信、アラーム、Siriの呼び出し）などのシーンで、デバイスのキャプチャ異常に対するリカバリロジックを最適化しました。
H264サポート検出ロジックを最適化し、Android Chrome 88以下で初めてブラウザ検出を開いた時にH264をサポートしていないという問題を回避しました。

Version 4.14.1 @2022.08.11

Feature

SEIメッセージの送信Client.sendSEIMessageをサポートしました。

Improvement

Client.switchRoleインターフェースの堅牢性を向上させました。

Version 4.14.0 @2022.08.05

Breaking Change

Client.on('client-banned')イベントのコールバックパラメータを変更しました。アップグレードの際はご注意ください。

Feature

キャプチャ音量の設定LocalStream.setAudioCaptureVolumeをサポートしました。

Improvement

Client.on('client-banned')イベントコールバックパラメータを変更し、受動的な退室の理由を追加しました。
Firefoxの画面共有キャプチャの解像度が想定と異なる問題を回避しました。
iOS Safariでローカルストリームの再生時にブラックスクリーンが発生することがある問題を回避しました。
ローエンドPCデバイスでのスモールストリームエンコードのパフォーマンスを最適化しました。

Bug Fixed

特定のシーンで画面共有ビットレートが想定に達しない問題を修正しました。

Version 4.13.0 @2022.07.08

Feature

インターフェースClient.destroyを追加し、Clientのライフサイクルを改善しました。

Improvement

スモールストリームエンコードのパフォーマンスを最適化し、スムーズさを向上させました。
キャプチャ自動回復の成功イベントDEVICE_AUTO_RECOVEREDを追加しました。

Bug Fixed

Client.startMixTranscodeインターフェースでクラウドミクスストリーミングを開始した際に、30秒を超えた切断からネットワークが復旧すると、ミクスストリーミングが続行しない不具合を修正しました。
Client.startPublishCDNStreamインターフェースがCDNにストリームをプッシュする際に、30秒を超えた切断からネットワークが復旧すると、CDNストリームが切断される不具合を修正しました。

Version 4.12.7 @2022.06.17

Improvement

Client.startMixTranscodemixUser.renderModeパラメータで設定されたミクスインストリームレンダリングモードをサポートしています。
デフォルトのvideo profileを `480p_2` に変更することで、画質を維持したままアップストリーム帯域幅の消費を抑えることができるようになりました。LocalStream.setVideoProfileをご参照ください。
オートレジューム再生の成功率を引き上げます。
切断、再接続後のmute状態の精度を向上させます。
Mac Safari 15.1 muteVideoページがクラッシュする問題を回避しました。webkit bug。

Bug Fixed

デュアルストリームのシナリオ（非オートサブスクリプション）で `stream-added` イベントがスローされないことがある問題を修正しました。

Version 4.12.6 @2022.06.10

Improvement

client.joinに、重複入室防止ロジックが追加されました。詳細については、アップグレードガイドラインをご参照ください。

Version 4.12.5 @2022.05.20

Bug Fixed

IEにnpmパッケージのtrtc.umd.jsファイルをロードする際にエラーが発生する問題を修正しました。
サブスクリプション状態を変更すると、プルスモールストリームに異常が発生することがある問題を修正しました。
Chrome 70以下のバージョンで、divコンテナを移動すると再生が一時停止する問題を回避しました。

Version 4.12.4 @2022.05.07

Improvement

入室のプロセスを最適化し、入室にかかる時間を短縮しました。
スモールストリームの切り替えロジックを最適化しました。

Bug Fixed

stream-addedイベントがスローされないことがある問題を修正しました。
FirefoxでLogitechカメラを使用して480pでキャプチャを行えない問題を回避しました。
iPad WKWebviewへのsdkインポート時にエラーが発生する問題を修正しました。

Version 4.12.3 @2022.04.19

Improvement

iOS 13、14の高解像度キャプチャロジックを最適化しました。
イベント監視ロジックを最適化し、SDKが業務側をキャプチャするエラーを回避しました。

ダッシュボードのトラブルシューティングに役立つよう、Safari音量レポートを追加しました。

Bug Fixed

liveモードでネットワーク切断後の再接続時に異常が発生することがある問題を修正しました。
iOS 11で音量取得に異常が発生する問題を修正しました。

Version 4.12.2 @2022.04.02

Improvement

音量計算ロジックを最適化し、メモリ占有率およびパフォーマンスのオーバーヘッドを低減させました。

Bug Fixed

ページを長時間バックエンドに切り替えるとキックアウトされることがある問題を修正しました（client-bannedイベントを受信）。
iOS 15.2～15.4でカメラを切り替えた後、エコーが発生する問題を回避しました。iOS Safariの既知の問題 case 11をご参照ください。

Version 4.12.1 @2022.03.18

Note

このバージョンへのアップグレードの際は注意が必要です。アップグレードガイドラインをご参照ください。

Improvement

stream.play呼び出しの繰り返し、イメージ再生の設定、再生パラメータの動的変更をサポートしました。
キャプチャ自動再開ロジックを改善し、プッシュ時にウォーターマークが脱落することがある問題を回避しました。

Bug Fixed

muteVideo/unmuteVideo後に、リモートのプルスモールストリームがブラックスクリーンになる問題を修正しました。
スモールストリームの切断後にstream-subscribedイベントがスローされる問題を修正しました。

Breaking Change

TRTC.createStreamインターフェースのmirrorプロパティを破棄しました。stream.play(elementId, { mirror: true })をご使用ください。

Version 4.12.0 @2022.03.04

Note

このバージョンへのアップグレードの際は注意が必要です。アップグレードガイドラインをご参照ください。

Feature

client.setRemoteVideoStreamTypeを非同期に変更して、Promiseを返すようにし、Promiseの状態によってデュアルストリームの切り替えに成功したかどうかを判断できるようになりました。

Improvement

海外サービスのスケジューリングの正確性を最適化しました。

Bug Fixed

user_time_outを受信するとキックアウトされることがある問題を修正しました。

Version 4.11.13 @2022.02.17

Improvement

npmパッケージのTypescriptステートメントファイルを更新しました。
stream.playパラメータの検証ロジックを最適化しました。

Bug Fixed

iOS 13で権限承認前にLocalStream.initialize失敗のエラーが報告されることがある問題を修正しました。
AUDIO_VOLUMEイベントで値が0になることがある問題を修正しました。

Version 4.11.12 @2022.01.11

Improvement

npmパッケージのTypescriptステートメントファイルを提供しました。
stream.close()インターフェース実装ロジックを最適化しました。
プッシュ側の頻繁なpublish/unpublishのシグナル交信ロジックを最適化しました。

Bug Fixed

iOS 15.1で、デスクトップ版のWeb通話を起動した際にページがcrashする問題を修正しました。詳細については、iOS Safariの既知の問題 case 7をご参照ください。
LocalStream.setAudioProfile('high')でビットレートが192kbpsに設定される問題を修正しました。

Version 4.11.11 @2021.12.17

Improvement

キャプチャ自動再開ロジックを最適化し、一部のローエンドAndroid端末でキャプチャ異常が復旧できない問題を回避しました。
自動再生ポップアップウィンドウの形式を最適化しました。

Version 4.11.10 @2021.12.03

Bug Fixed

`enableAutoPlayDialog: false` の設定で、SDKの自動再生ポップアップウィンドウを無効化できない問題を修正しました。
stream.playを繰り返し呼び出してもSDKによってブロックされない問題を修正しました。

Version 4.11.9 @2021.11.26

Note

このバージョンへのアップグレードの際は注意が必要です。アップグレードガイドラインをご参照ください。

Improvement

SDKが自動再生に失敗した際に、自動再生失敗の問題を解決するためのインタラクティブポップアップウィンドウの表示をサポートしました。詳細については、制限された自動再生の処理に関するアドバイスをご参照ください。
【iOS 15.1でプッシュにcrashが起きる問題】の回避ロジックを最適化しました。詳細については、iOS Safariの既知の問題 case 7をご参照ください。
無音になることがある問題を回避するため、TRTC.getMicrophonesは、deviceIdが'communications'であるマイクを返さないようにしました。詳細については、Chromeの既知の問題 case 8 & 9をご参照ください。
switchDeviceポリシーを最適化しました。
webview環境におけるコーデックサポート検出の精度を向上させました。
client.startPublishCDNStream、client.stopPublishCDNStream、client.startMixTranscodeおよびclient.stopMixTranscodeインターフェースのパラメータ検証を改善しました。

Bug Fixed

client.publishでTRTCをサポートしていないというエラーが表示されることがある問題を修正しました。

Version 4.11.8 @2021.11.05

Improvement

iOS 15.0で、ビデオ再生がブラックスクリーンになることがある問題を回避しました。詳細については、iOS Safariの既知の問題 case 6をご参照ください。
iOS 15.1 で、必ずプッシュにcrashが起きるという問題を回避しました。詳細については、[iOS Safariの既知の問題 case 7]https://web.sdk.qcloud.com/trtc/webrtc/doc/zh-cn/tutorial-02-info-webrtc-issues.html#h2-4!86108718756fa2dfc837d73a463d196e)をご参照ください。

Version 4.11.7 @2021.09.30

Improvement

キーインターフェースは、パラメータタイプの強力な検証を追加します。
開発モード（LogLevelはDebug）での中国語のエラーメッセージプロンプトをサポートしました。
デバイスのキャプチャが異常な場合の、キャプチャの自動復旧の成功率を引き上げます。
システムがスリープ状態になり、再起動した後の通話再開ロジックを最適化しました。
trtc.esm.jsおよびtrtc.umd.jsを追加して、さまざまなシナリオのニーズに対応しました。参考ガイドをご参照ください。

Version 4.11.6 @2021.09.10

Improvement

シグナリングスケジューリングロジックを最適化し、弱いネットワーク環境での入室成功率を向上させました。
v4.11.5はこのバージョンにアップグレードすることをお勧めします。

Version 4.11.5 @2021.09.04

Improvement

シグナリングチャネルの動的スケジューリングをサポートし、弱いネットワーク環境での接続成功率を向上させました。
ルーム間のミクスストリーミングをサポートしました。Client.startMixTranscodeをご参照ください。

Bug Fixed

ネットワーク切断と再接続後に、stream-addedイベントを受信できないことがある問題を修正しました。
長時間の画面共有時にフレームレートが0に低下することがある問題を修正しました。

Version 4.11.4 @2021.08.20

Improvement

oppo＆vivo内蔵ブラウザでのH.264のサポートレベル検出の精度を向上させました。
キャプチャ自動回復ロジックを追加しました（デバイスのキャプチャが異常な場合にトリガーされます）。
subscribeインターフェースのタイムアウトロジックを追加しました。エラーコードについては、
API_CALL_TIMEOUTをご参照ください。

Bug Fixed

旧バージョンの一部iOS Safariのプルが失敗することがある問題を修正しました。
デバイスの切り替え後、mute状態が不正確になる問題を修正しました。
入室タイムアウト時に、入室インターフェースをもう一度呼び出すと、異常が発生することがある問題を修正しました。
リモートがプッシュをキャンセルした後、速やかにオーディオビデオプレーヤーが破棄されない問題を修正しました。

Version 4.11.3 @2021.07.30

Improvement

publish & subscribeインターフェースのトラブルシューティングロジックを最適化しました。
ミキシングプラグインの復旧ポリシーを最適化しました。

Bug Fixed

peer-leave通知が不正確なことがある問題を修正しました。

Version 4.11.2 @2021.07.23

Improvement

turn serverスケジューリングをサポートし、接続成功率を向上させました。
Client.getRemoteMutedStateにhasSmallプロパティを追加し、リモートのスモールプッシュの有無を識別します。

Bug Fixed

LocalStorageが無効である場合に、SDKを使用できない問題を修正しました。
publish異常が発生した場合に、インターフェースがrejectedされない問題を修正しました。

Version 4.11.1 @2021.06.25

Improvement

美顔プラグインをサポートしました。美顔を有効にするをご参照ください。
データ統計の正確性を最適化しました。

Version 4.11.0 @2021.06.18

Feature

デュアルストリームをサポートしました。チュートリアル：デュアルストリームの伝送を有効にするをご参照ください。

Improvement

イベント通知シーケンスを最適化しました。

Version 4.10.3 @2021.06.11

Improvement

品質データ統計ロジックを最適化し、サーバーAPIによる通話品質データの取得をサポートしました。
ClientEvent.NETWORK_QUALITYのイベントで、rttとlossデータを返します。
インターフェース検証ロジックを最適化し、呼び出しに異常が繰り返し発生することを防止しました。
再生ロジックを最適化し、オーディオ再生消費時間を削減しました。

Version 4.10.2 @2021.05.24

Improvement

switchDeviceインターフェースの実装ロジックを最適化して、Huaweiブラウザがフロントカメラを切り替えられないことがある問題を回避しました。
StreamEvent.CONNECTION_STATE_CHANGEDイベント通知の精度を向上させました。

Bug Fixed

Native画面共有が再生できないことがある問題を修正しました。
再接続後にstream-removedイベントを受信できないことがある問題を修正しました。

Version 4.10.1 @2021.04.30

Feature

Stream接続ステータスの変更の監視をサポートするStreamEvent.CONNECTION_STATE_CHANGEDイベントを追加しました。
ダウンストリームRTT統計データを取得するため、Client.getTransportStatsインターフェースをサポートしました。
サブストリーム（画面共有）の統計データを取得するため、Client.getRemoteVideoStatsインターフェースをサポートしました。

Improvement

Client.switchRoleインターフェースの実装ロジックを最適化しました。

Bug Fixed

stream-addedが追加される前に、mute関連イベントがトリガーされることがある問題を修正しました。
ルーム入室時に無音になることがある問題を修正しました。

Version 4.10.0 @2021.04.16

Feature

インターフェースClient.startPublishCDNStreamを追加し、ストリームをTencent Cloud CDNおよびサードパーティのCDNにプッシュできるようにしました。
インターフェースClient.stopPublishCDNStreamを追加し、Tencent Cloud CDNおよびサードパーティのCDNへのプッシュを停止できるようにしました。

Improvement

LocalStream.switchDevice、LocalStream.addTrack、LocalStream.removeTrackインターフェースのパラメータ検証ロジックを最適化しました。

Version 4.9.0 @ 2021.03.19

Feature

クラウドミクスストリーミングがプリセットレイアウトモードをサポートしました。Client.startMixTranscodeインターフェースをご参照ください。
音量コールバックをサポートしました。Client.enableAudioVolumeEvaluationインターフェースをご参照ください。

Improvement

Websocketのデフォルトポートは443に変更されました。

Bug Fixed

liveモードで、視聴者がキャスターの入退室通知を受信できない問題を修正しました。
文字列のルームナンバーを使用する場合、再接続に失敗する場合がある問題を修正しました。

Breaking Change

TRTC.checkSystemRequirementsは、詳細なテスト結果を返します。詳細については、インターフェースドキュメントおよびアップグレードガイドラインをご参照ください。

Version 4.8.6 @ 2021.03.01

Improvement

プルストリームステレオ再生をサポートしました。
注意
iOSプラットフォームでは、まだサポートされていません。

Bug Fixed

モバイル端末で静止画像がミュートされているときに、Webがstream-removedイベントを受信するという問題を修正しました。

Version 4.8.5 @ 2021.01.29

Improvement

Client.setTurnServerで複数のturn serverのセットアップをサポートしました。
userId検証ロジックを最適化しました。

Bug Fixed

ストリームがプッシュされた後、mute状態が不正確になることがある問題を修正しました。

Version 4.8.4 @ 2021.01.15

Improvement

LocalStream.setVideoProfileインターフェースで動的呼び出しをサポートしました。
ダッシュボードのデータレポートロジックを最適化しました。

自動再生が制限されている場合の処理ロジックを最適化しました。制限された自動再生の処理に関するアドバイスをご参照ください。
デバイスプラグインと自動リカバリが失敗した場合の処理ロジックを最適化しました。
DEVICE_AUTO_RECOVER_FAILEDをご参照ください。

Bug Fixed

ストリームが再度プッシュされた後、mute状態が不正確になることがある問題を修正しました。

Version 4.8.3 @ 2021.01.07

Improvement

入室インターフェースのroomIdパラメータ検証ロジックを最適化しました。

Bug Fixed

v4.8.2バージョンにサードパーティとの依存関係がないという問題を修正しました。
オーディオのみをサブスクリプションすると、再生時に無音になることがある問題を修正しました。
iOSの自動再生が制限された場合、resumeが無音になり、getAudioLevelが0になることがある問題を修正しました。

Version 4.8.2 @ 2020.12.31

Improvement

入室インターフェースのroomIdパラメータ検証ロジックを最適化しました。詳細については、インターフェースドキュメントおよびアップグレードガイドラインをご参照ください。
peer-joinおよびpeer-leaveイベントの通知タイミングを最適化しました。

Bug Fixed

退室時に、`Cannot read property 'isConnected' of null`というエラーが報告されることがある問題を修正しました。

Breaking Change

破棄されたインターフェース：setDefaultMuteRemoteStreamsを削除し、代わりに、TRTC.createClientのautoSubscribeパラメータをご使用ください。

Version 4.8.1 @ 2020.12.25

Bug Fixed

Windowsでリモートユーザーの声が聞こえなくなることがある問題を修正しました。
client.getRemoteVideoStats()インターフェースが空のデータを返す問題を修正しました。

Version 4.8.0 @ 2020.12.18

Feature

Cloud MixTranscodingをサポートしました。
すべてのプラットフォームで文字列のルーム番号をサポートしました。TRTC.createClientのuseStringRoomIdパラメータをご参照ください。
自動サブスクリプションの無効化をサポートしました。TRTC.createClientのautoSubscribeパラメータをご参照ください。

Improvement

h264サポート検出ロジックを最適化しました。
デバイススイッチングロジックを最適化しました。
hasAudio/hasVideoインターフェースステータス判定ロジックを最適化しました。

Bug Fixed

ネットワーク切断後、再接続時に失敗のエラー報告が偶発的に発生する問題を修正しました。
iOS Safariで頻繁にadd/remove trackのブラックスクリーンが発生する問題を修正しました。

Version 4.7.1 @ 2020.11.27

Improvement

メディアデバイスが変更された場合（例えば、ポートが緩んでいる、デバイスの抜き差しなど）の自動リカバリキャプチャロジックを最適化しました。
エラーコードDEVICE_AUTO_RECOVER_FAILEDを追加しました。デバイスのリカバリが失敗した際の表示に用いられます。

Bug Fixed

Chrome/87バージョンでエラーのブラックスクリーンが偶発的に発生する問題を修正しました。
Nativeプッシュカメラ+画面共有、Webのサブスクリプションの重複、サブスクリプションのキャンセル、画面共有ストリームが消失するという問題を修正しました。

Version 4.7.0 @ 2020.11.20

Feature

デスクトップFirefoxM56+およびデスクトップEdgeM80+をサポートしました。

Improvement

アップストリームビットレート制御ロジックを最適化しました。
メディアデバイスを取得するための再試行ロジックを最適化しました。
Websocket再接続ロジックを最適化しました。
デバイスが変更されたときのプッシュ自動リカバリロジックを最適化し、マイクがミキシング状態で抜き差しされたときのプッシュ自動リカバリをサポートしました。

Breaking Change

TRTC.checkSystemRequirementsは、詳細なテスト結果を返します。詳細については、インターフェースドキュメントおよびアップグレードガイドラインをご参照ください。

Version 4.6.7 @ 2020.11.05

Bug Fixed

Chromeがハードウェアアクセラレーションで有効になっているときに、プルストリームの視聴でちらつきが生じることがある問題を修正しました。
iOSのWeChatの内蔵ブラウザの場合に、入室してストリームをプルできない問題を修正しました。

Version 4.6.6 @ 2020.10.23

Improvement

アップストリームpeerConnection再接続ロジックを最適化しました。
ダウンストリームpeerConnection再接続ロジックを最適化しました。
TRTC.checkSystemRequirements検出ロジックを最適化しました。
Safariの画面共有をサポートしました。詳細については、画面共有チュートリアルをご参照ください。

Bug Fixed

自動再生ポリシーの制限により、オーディオ再生を手動で再開した後、getAudioLevel値が0になる問題を修正しました。

Version 4.6.5 @ 2020.10.14

Improvement

WebSocketシグナリングチャネルの再接続ロジックを最適化し、接続の安定性を向上させました。
ログ出力ロジックを最適化しました。

Bug Fixed

Chromeの再サブスクリプション後にgetAudioLevelインターフェースの戻り値が0になる問題を修正しました。
Safariが再サブスクリプションした後に再生が無音になるという問題を修正しました。
replaceTrackを使用してアップストリームオーディオトラックを置き換えた後、getLocalVideoStatsインターフェースがundefinedを返す問題を修正しました。
モバイルデバイスの通話中、ネットワークタイプを切り替えるときに、WebSocketの接続が切断されることがある問題を修正しました。

Version 4.6.4 @ 2020.09.24

Improvement

退室後にネットワーク品質統計を停止します。

Bug Fixed

Chrome 56で入室するときにエラーを報告する問題を修正しました。
モバイル端末でバイパスをプッシュするときに画面が回転する問題を修正しました。
オーディオストリーミングのみプッシュされたとき、クラウドレコーディングが異常になる問題を修正しました。
解像度に一貫性がないため、カメラを抜いた後、自動的にプッシュを再開できないという問題を修正しました。

Version 4.6.3 @ 2020.08.28

Improvement

互換性検出ロジックを最適化しました。
ログレポートロジックを最適化しました。
アップストリームビットレート制御ロジックを最適化しました。

Version 4.6.2 @ 2020.08.14

Improvement

アップストリームビットレート調整ロジックを最適化しました。
switchRoleパラメータ検証ロジックを最適化しました。
アップストリームネットワーク品質の計算ロジックを最適化しました。
エラーメッセージを最適化しました。
現在のプッシュキャプチャデバイスの変更が検出されると、プッシュのステータスが自動的にリカバリされます。

Bug Fixes

unpublishが成功した後、すぐにpublish失敗のエラーが再度報告されるという問題を修正しました。

Version 4.6.1 @ 2020.07.28

Improvement

TRTC.isScreenShareSupportedはSafariでの画面共有をサポートしていません。
subscribe & unsubscribeインターフェースのパラメータ検証ロジックを改善しました。
ネットワーク品質ログを追加しました。

Bug Fixes

メディアデバイスが承認されておらず、また、TRTC.createStreamインターフェースで渡されるデバイスIDが空文字列である場合、SDKが OverconstrainedErrorを報告するという問題を修正しました。
アップストリームのpeerConnectionが切断されたときにログが出力されない問題を修正しました。

Version 4.6.0 @ 2020.07.16

Feature

NETWORK_QUALITYイベントを追加しました。

Version 4.5.0 @ 2020.07.02

Feature

createStreamインターフェースにscreenAudioパラメータを追加しました。

Bug Fixes

Androidブラウザでエコーキャンセレーションが機能しない問題を修正しました。
getTransportStatsインターフェースによって返されるrtt値がNANであるという問題を修正しました。

Version 4.4.0 @ 2020.05.28

Feature

Chrome >=74の画面共有およびキャプチャシステム(Windows)または現在のTabページ(Mac)の音声をサポートしました。

Version 4.3.14 @ 2020.04.29

Bug Fixes

ミニプログラムのオーディオmuted unmuteイベントを修正しました。

Version 4.3.13 @ 2020.04.16

Improvement

可用性検出ロジックを最適化しました。

Version 4.3.12 @ 2020.04.13

Bug Fixes

潜在的なRTCPeerConnectionのステータス変更異常を修正しました。

Version 4.3.11 @ 2020.03.28

Improvement

スマホのQQブラウザ検出を追加しました。スマホのQQブラウザは、現時点ではTRTCデスクトップブラウザSDKをサポートできません。

Bug Fixes

Boolean戻り値タイプを修正しました。

Version 4.3.10 @ 2020.03.17

Improvement

環境検出ロジックを最適化しました。
RtcErrorにname codeを追加しました。

Version 4.3.9 @ 2020.03.13

Improvement

デプロイ環境の自動検出を追加しました。
ログを最適化しました。

Version 4.3.8 @ 2020.02.24

Improvement

createClientにstreamId userdefinerecordidフィールドを追加しました。

Version 4.3.7 @ 2020.02.21

Improvement

画面共有中にデバイスを切り替えると、異常がスローされます。

Bug Fixes

デバイスを切り替えるときにMediaStreamをリリースして、デバイスが占有してしまう問題を解決しました。
潜在的なエラーを処理するためのサブスクリプションインターフェースを追加しました。

Version 4.3.6 @ 2020.02.05

Bug Fixes

Stream.resume()オーディオビデオの再生シーケンスを調整し、iOSのWeChatブラウザでの自動再生異常の問題を修正しました

Version 4.3.5 @ 2020.02.05

Improvement

publishタイムアウトチェックを追加して、シグナリング送信の成功率を向上させました

Version 4.3.4 @ 2020.01-06

Improvement

core-jsをv3.6.1にアップグレードしました。

Bug Fixes

unpublishは、タイムアウト後に外部に異常イベントをスローします。
サードパーティライブラリによって引き起こされるV8の最適化に失敗する問題を修正しました。

Version 4.3.3 @ 2019.12.25

Improvement

環境がwebrtc機能をサポートしているかどうかを自動的に検出する機能を追加しました
sdp応答メカニズムを最適化しました
レポートロジックを最適化しました

Bug Fixes

turn URLプロトコル形式を修正しました

Version 4.3.2 @ 2019.12.09

Improvement

ICEがダウンストリーム接続から切断されたときの自動再接続メカニズムを追加しました。
STUNホールパンチングのプロセスを削除し、プライベートネットワークユーザーの接続成功率を高め、接続速度を引き上げました。
-<p>ログレポートのタイムスタンプは、サーバーによって修正されたUTC時間を一律使用します。
ICEエラーレポートを最適化しました。
avmonitorモニタリングに報告する重要なイベントをさらに追加しました。
Bug Fixes

WebSocketシグナリングチャネル1005の異常な再接続と再接続エラー処理を修正しました。

ダウンストリームパケットロス率レポートの問題を修正しました。
Version 4.3.1 @ 2019.11.23

Improvement

通話中にアップストリームリンクICEが切断された場合の自動再接続メカニズムを追加しました。

Bug Fixes

STUNホールパンチングが失敗すると、hostパブリックIPタイプICE Candidateが有効にならない問題を修正しました。

Version 4.3.0 @ 2019.11.15

Feature

Client.getTransportStats() APIを追加しました。

Improvement

より詳細なレポートログを追加しました。

イベントのバインド解除はワイルドカードをサポートします。
接続タイムアウト時間を5sに増加しました。
リリースタイムアウト時間を5sに増加しました。
Bug Fixes

zone.jsによるプロトタイプチェーンの変更によって引き起こされるSDKの判断異常の問題を修正しました。

Version 4.2.0 @ 2019.11.04

Feature

Client.off()インターフェースを追加して、クライアントイベントのバインドをキャンセルしました。

Improvement

通話ステータス統計を最適化しました。

Client.publish()に権限チェックを追加しました。
Stream.play()/resume()自動再生エラープロンプトを追加しました。
Bug Fixes

LocalStream.switchDevice()は、カメラ切り替え時のブラックスクリーンの問題を修正しました。

Version 4.1.1 @ 2019.10.24

Bug Fixes

ログ損失の問題を修正しました。

ネットワークの切断と再接続後にリモートユーザーが消失する問題を修正しました。
Version 4.1.0 @ 2019.10.17

Feature

Stream.play()インターフェースは、HTMLDivElementオブジェクトのインプットをサポートしています。

オーディオビットレート制御設定を追加しました。開発者はLocalStream.setAudioProfile()を介して、オーディオプロパティを設定することができます。現在、standardとhighという2つのProfileがサポートされています。
Bug Fixes
-<p>旧バージョンのChromeでのWebAudio Contextの数量制限の問題を修正しました。

replaceTrack()がローカルオーディオビデオプレーヤーを再起動しない問題を修正しました。

LocalStream.setScreenProfile()のカスタムプロパティ設定が有効にならない問題を修正しました。
audio/video playerの再起動とステータスレポートの問題を修正しました。
Version 4.0.0 @ 2019.10.11
リファクタリングされたバージョンのTRTCデスクトップブラウザSDKは、Client/Streamモードのインターフェースを提供します。各オブジェクトの責任がより明確になり、セマンティクスがより簡潔明瞭になりました。
リファクタリングされたバージョンは旧バージョンと互換性がありません。インターフェースの変更だけでなく、次の機能も提供しています。

ビデオのプロパティ（解像度、フレームレートおよびビットレート）は、SDKのLocalStream.setVideoProfile()インターフェースを介してAppによって完全に制御されます。旧バージョンは、Tencent Cloudコンソールの「画面設定(Spear Role)」ではサポートされなくなりました。

SDKはオーディオビデオプレーヤーをStreamオブジェクトにカプセル化し、オーディオビデオの再生はSDKによって完全に制御されます。
リモートストリーミングのサブスクリプションおよびサブスクリプション解除機能を提供しています。開発者はClient.subscribe()/unsubscribe()インターフェースを介して、リモートストリームのオーディオ、ビデオまたはオーディオビデオデータストリームの受信をフレキシブルにコントロールできます。

提供远端流的订阅与取消订阅功能，开发者可以通过 Client.subscribe()/unsubscribe() 接口灵活控制远端流的音频、视频或音视频数据流的接收。

APIコードサンプル
iOS&Mac

最終更新日：：2022-12-26 10:43:35

ここでは、主にTRTC-API-Example（iOS&Mac）を素早く実行する方法をご紹介します。

環境要件

- Xcode 11.0およびそれ以降のバージョン。
- プロジェクトが有効な開発者による署名を設定済みであることを確認してください。
- Qt Creator 4.13.3（Mac）およびそれ以降のバージョン。

前提条件

Tencent Cloudの登録によりアカウントを登録している。

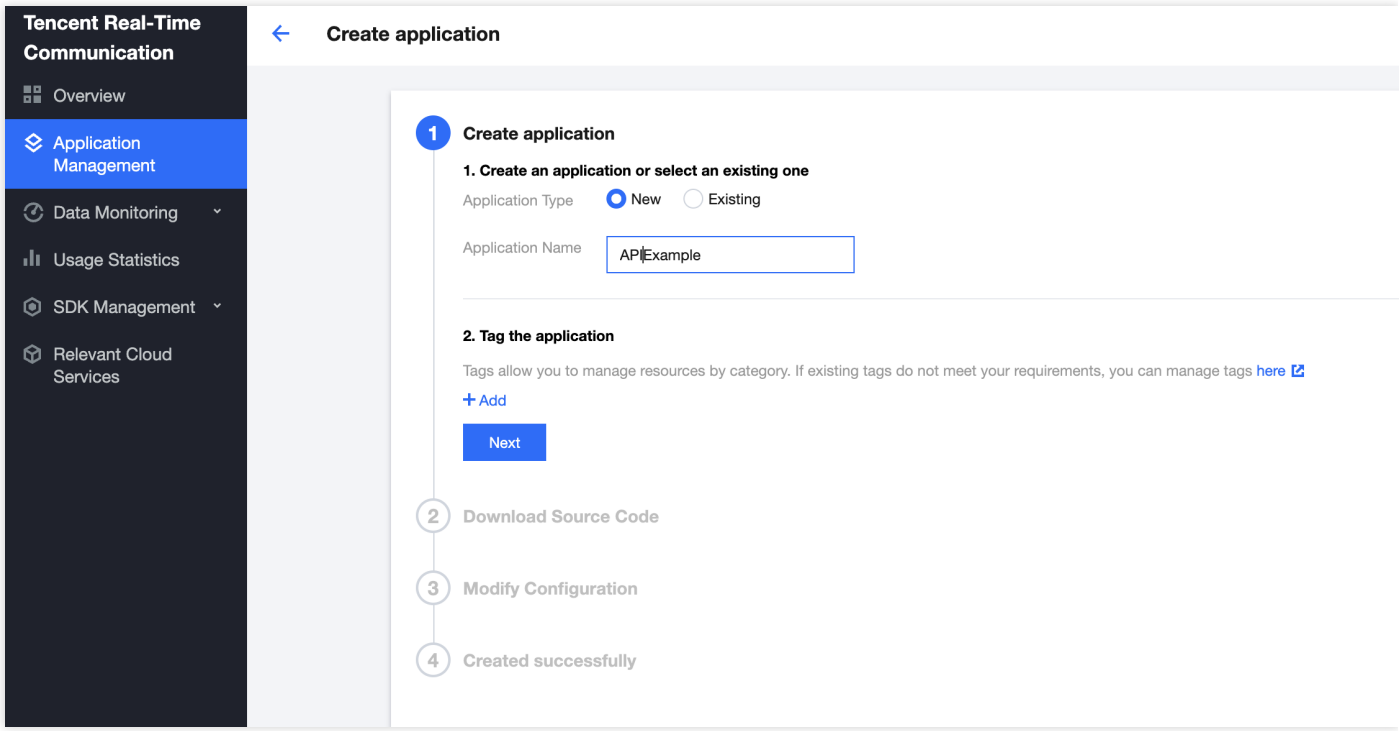
操作手順

手順1：新規アプリケーションの作成

1. TRTCコンソールにログインし、【アプリケーション管理】を選択します。

2. 【アプリケーションの作成】をクリックし、`APIExample`などのアプリケーション名を入力します。すでにアプリケーションを作成している場合は、【既存のアプリケーションを選択】にチェックを入れ、【次のス

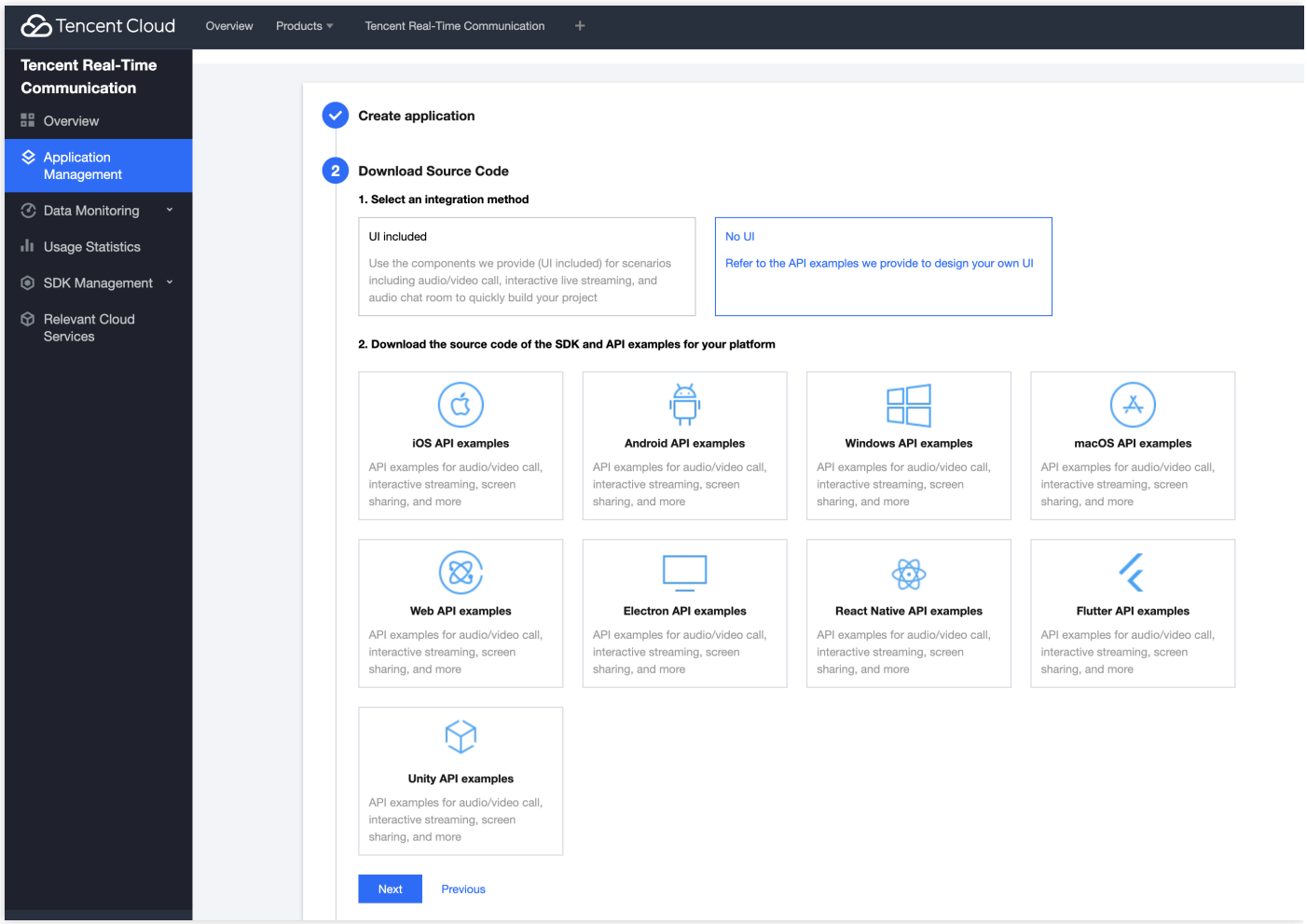
テップ】をクリックします。



ステップ2：サンプルコードのダウンロード

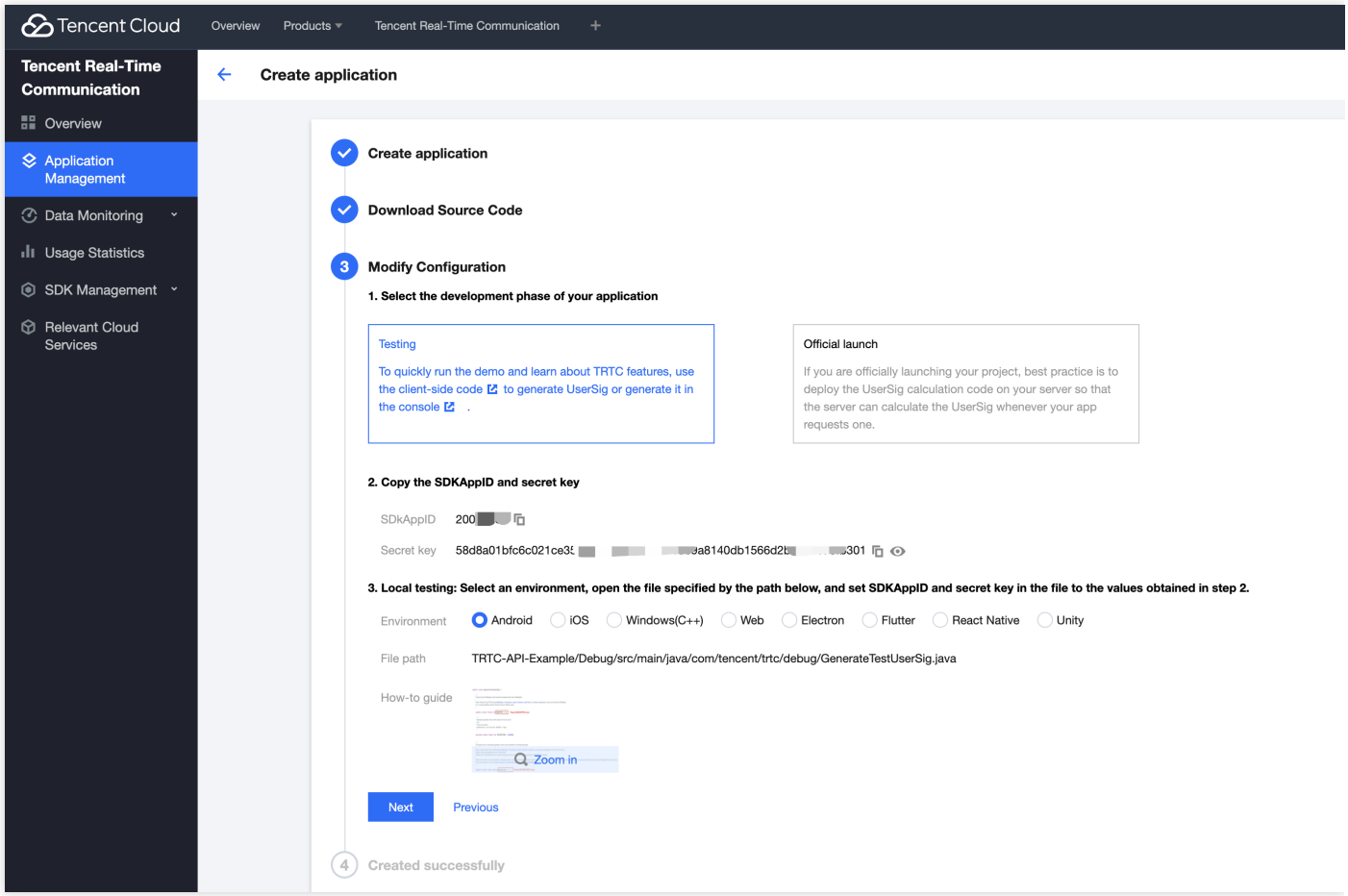
1. UIなしの統合を選択後、ご自身の業務プラットフォームに合わせて、Githubで対応するプラットフォームのサンプルコードをダウンロードします。

2. ダウンロード完了後、【次のステップ】をクリックします。



ステップ3：プロジェクトの設定

1. サンプルプロジェクトのクイックスタート段階で【デバッグ段階】を選択します。その後、ご自身のSDKAppID、Secret keyを記録しておきます。



2. ダウンロードしたサンプルコードを開き、図の指示に従って対応するファイルの場所を探し、ご自身のSDKAppID、Secret keyに変更します。プロジェクトの設定はこれで完了です。【次のステップ】をクリックしてください。

説明：

- ここで言及したUserSigの作成法は、クライアントコードに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TRTC-API-Exampleクイックスタートおよび機能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方法となります。 UserSigが必要なときは、Appから業務サーバーにリクエストを送信し動的にUserSigを取得します。詳細はサーバーでのUserSig新規作成をご参照ください。

手順4：コンパイルと実行

XCode（11.0およびそれ以降のバージョン）を使用してソースディレクトリ配下の TRTC-API-Example-OC.xcworkspace プロジェクトを開き、 TRTC-API-Exampleプロジェクトをコンパイルして実行します。

よくあるご質問

1. 2台の携帯電話で同時にプロジェクトを実行しているのに、お互いの画面が表示されないのはなぜですか。

2台の携帯電話でプロジェクトを実行するとき、UserIDが異なるものを使用してください。TRTCでは、同一のUserID（SDKAppIDが異なる場合を除く）が2つの端末で同時に使用することをサポートしていません。

2. ファイアウォールにはどのような制限がありますか。

SDK が UDP プロトコルを使用してオーディオビデオ伝送を行っていることから、 UDPに対してブロックがあるオフィスネットワークでは使用することができません。類似した問題がおありの際は、 企業ファイアウォール制限の対応をご参照の上、問題及び原因解決にお役立てください。

Android

最終更新日：：2022-12-26 10:43:35

このドキュメントでは、主にTRTC-API-Example（Android）をすばやく実行する方法について説明します。

環境要件

- 互換性のある最低バージョンはAndroid 4.1（SDK API Level 16）。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。
- AppにはAndroid4.1およびそれ以降のデバイスが必要です。

前提条件

Tencent Cloudの登録によりアカウントを登録している。

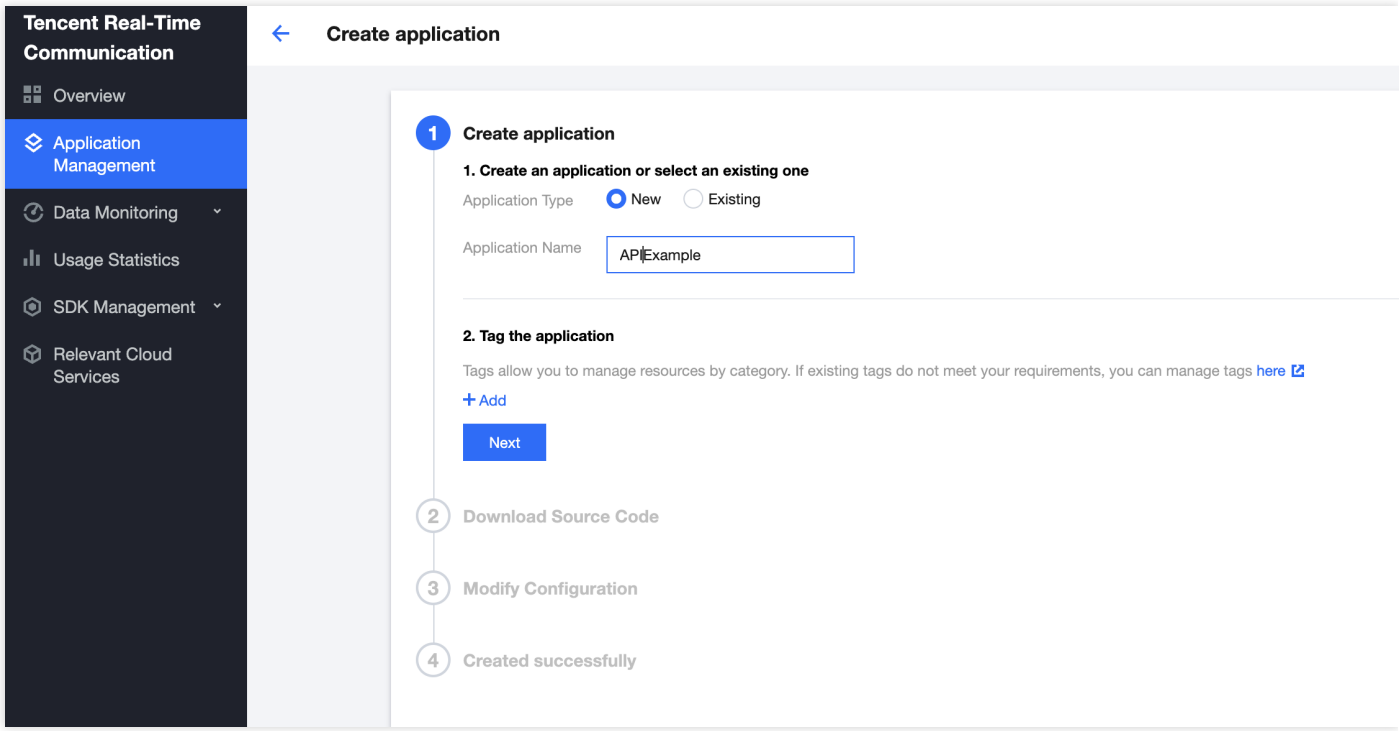
操作手順

手順1：新規アプリケーションの作成

1. TRTCコンソールにログインし、【アプリケーション管理】を選択します。

2. 【アプリケーションの作成】をクリックし、`APIExample`などのアプリケーション名を入力します。すでにアプリケーションを作成している場合は、【既存のアプリケーションを選択】にチェックを入れ、【次のス

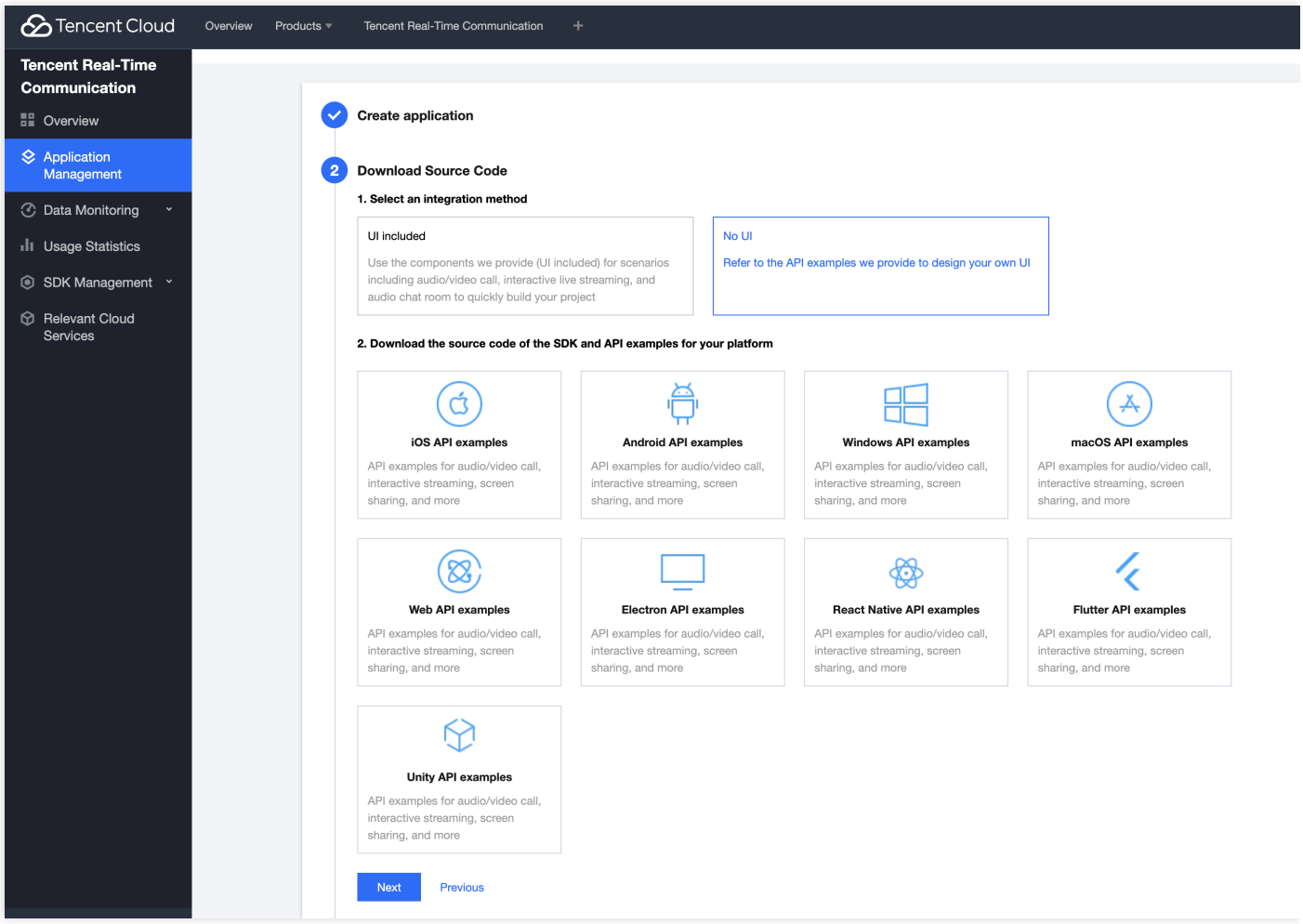
テップ】をクリックします。



ステップ2：サンプルコードのダウンロード

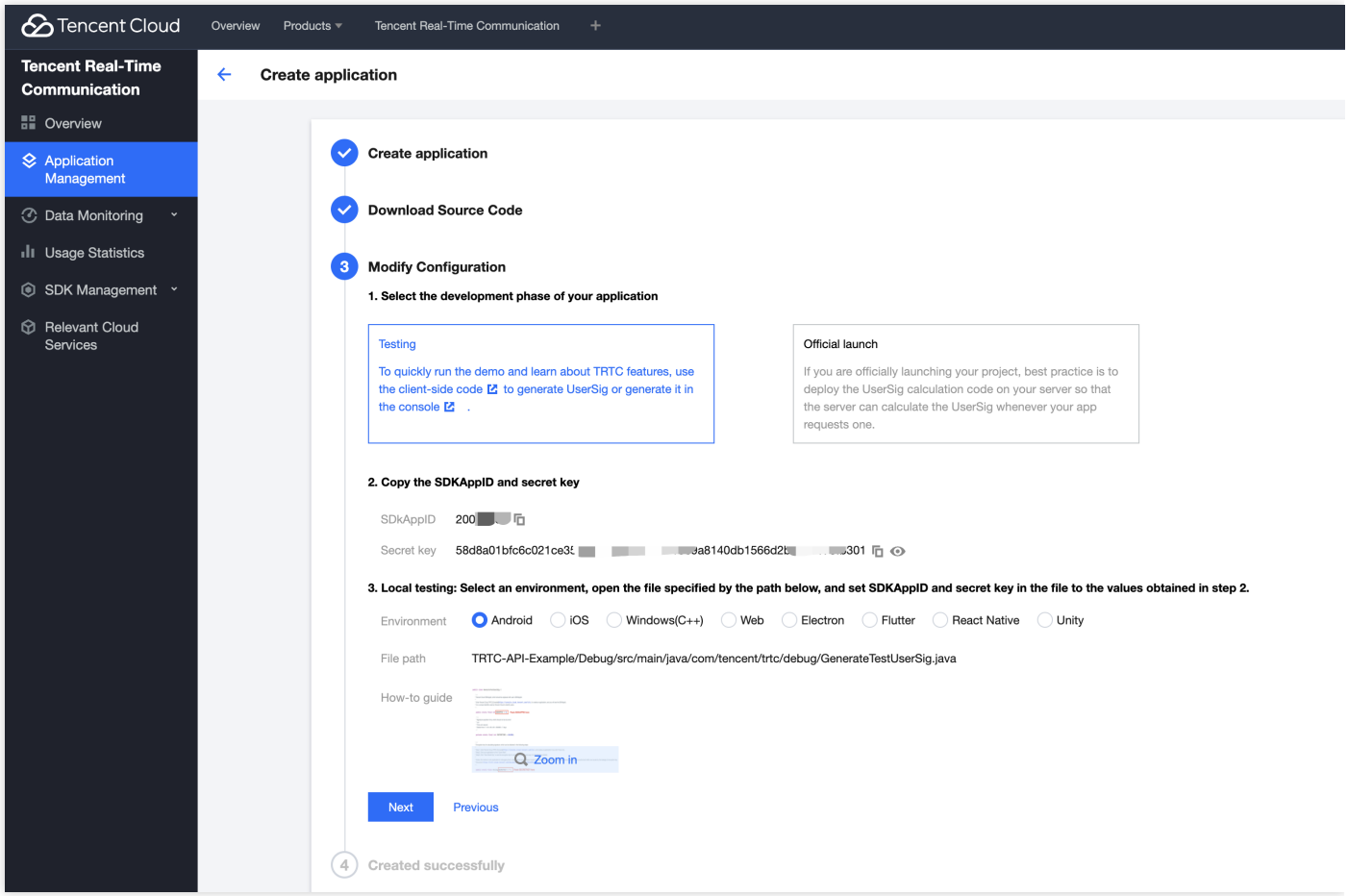
1. UIなしの統合を選択後、ご自身の業務プラットフォームに合わせて、Githubで対応するプラットフォームのサンプルコードをダウンロードします。

2. ダウンロード完了後、【次のステップ】をクリックします。



ステップ3：プロジェクトの設定

1. サンプルプロジェクトのクイックスタート段階で【デバッグ段階】を選択します。その後、ご自身のSDKAppID、Secret keyを記録しておきます。



2. ダウンロードしたサンプルコードを開き、図の指示に従って対応するファイルの場所を探し、ご自身のSDKAppID、Secret keyに変更します。プロジェクトの設定はこれで完了です。【次のステップ】をクリックしてください。

説明：

- ここで言及したUserSigの作成法は、クライアントコードに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TRTC-API-Exampleクイックスタートおよび機能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方法となります。 UserSigが必要なときは、Appから業務サーバーにリクエストを送信し動的にUserSigを取得します。詳細はサーバーでのUserSig新規作成をご参照ください。

手順4：コンパイルと実行

設定完了後、Android Studio（3.5およびそれ以降のバージョン）を使用してソースプロジェクト `TRTC-API-Example` を開き、【実行】をクリックすれば体験できます。

よくあるご質問

1. 2台の携帯電話で同時にAppを実行しているのに、お互いの画面が表示されないのはなぜですか。

2台の携帯電話でAppを操作するとき、UserIDが異なるものを使用してください。TRTCでは、同一のUserID（SDKAppIDが異なる場合を除く）が2つの端末で同時に使用することをサポートしていません。

2. ファイアウォールにはどのような制限がありますか。

SDK が UDP プロトコルを使用してオーディオビデオ伝送を行っていることから、 UDPに対してブロックがあるオフィスネットワークでは使用することができません。類似した問題がおありの際は、 企業ファイアウォール制限の対応をご参照の上、問題及び原因解決にお役立てください。

Windows C++

最終更新日：：2022-12-26 10:43:35

このドキュメントでは、主にTencent Cloud TRTC Demo (Windows C++）を素早く実行する方法を紹介します。

環境要件

- Microsoft Visual Studio 2017バージョン以上、Microsoft Visual Studio 2019の使用を推奨します。
- QT（QT 5.14.xバージョン）開発環境をダウンロードしてインストールします。
- .vsixプラグインファイルをダウンロードしてインストールします。公式サイトで対応するプラグインのバージョンを探してインストールすると完了できます。
- VSを開いてツールバーから `QT VS Tools -> Qt Options -> Qt Versions` をさがし、自分のQtコンパイラ msvcをadd（追加）します。
- `SDK/CPlusPlus/Win64/lib` 下のすべての `.dll` ファイルをプロジェクトディレクトリ下の `debug` / `release` フォルダ下にコピーする必要があります。

> 注意：
> `Debug/release` フォルダはいずれもVSの環境構成後に自動的に生成されます。32bitプログラムならば、 `SDK/CPlusPlus/Win64/lib` 下のすべての `.dll` を `debug` / `release` フォルダ下にコピーする必要があります。

前提条件

Tencent Cloudの登録によりアカウントを登録している。

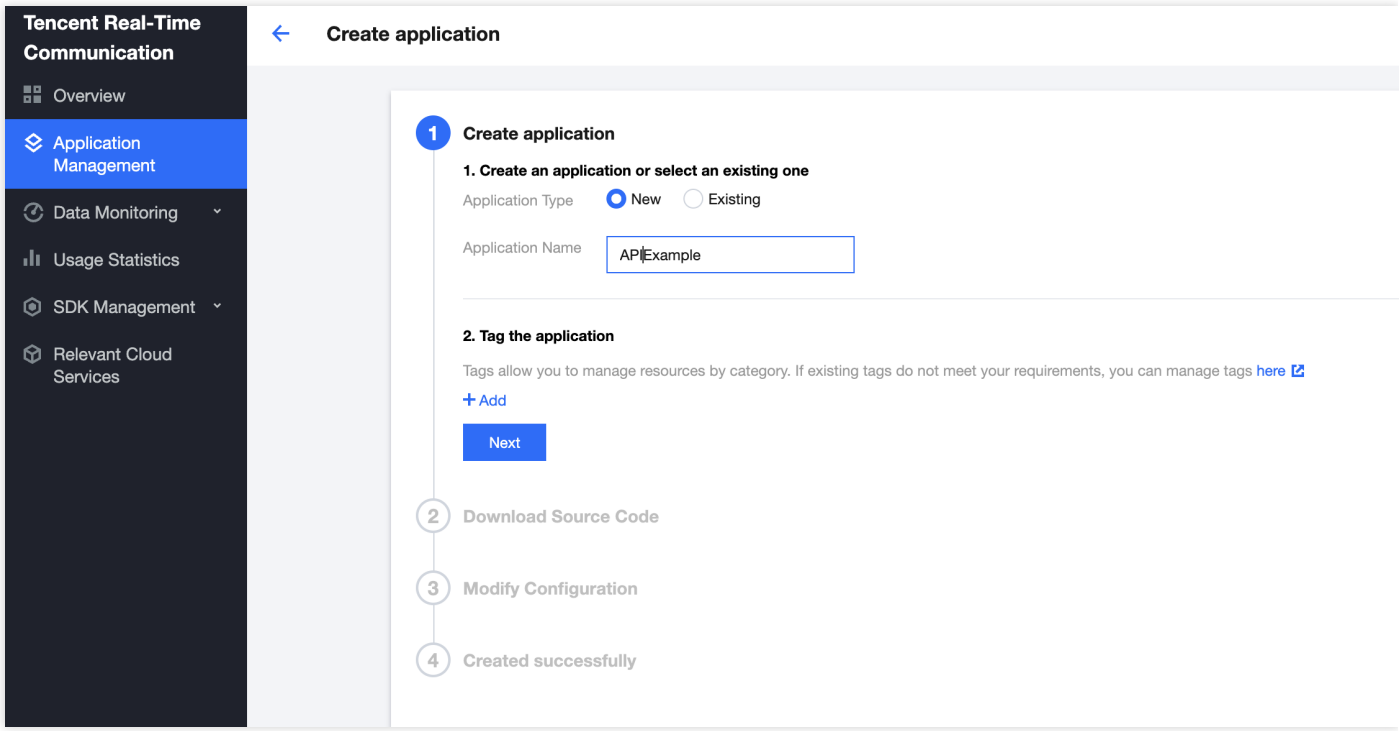
操作手順

手順1：新規アプリケーションの作成

1. TRTCコンソールにログインし、【アプリケーション管理】を選択します。

2. 【アプリケーションの作成】をクリックし、 `APIExample` などのアプリケーション名を入力します。すでにアプリケーションを作成している場合は、【既存のアプリケーションを選択】にチェックを入れ、【次のス

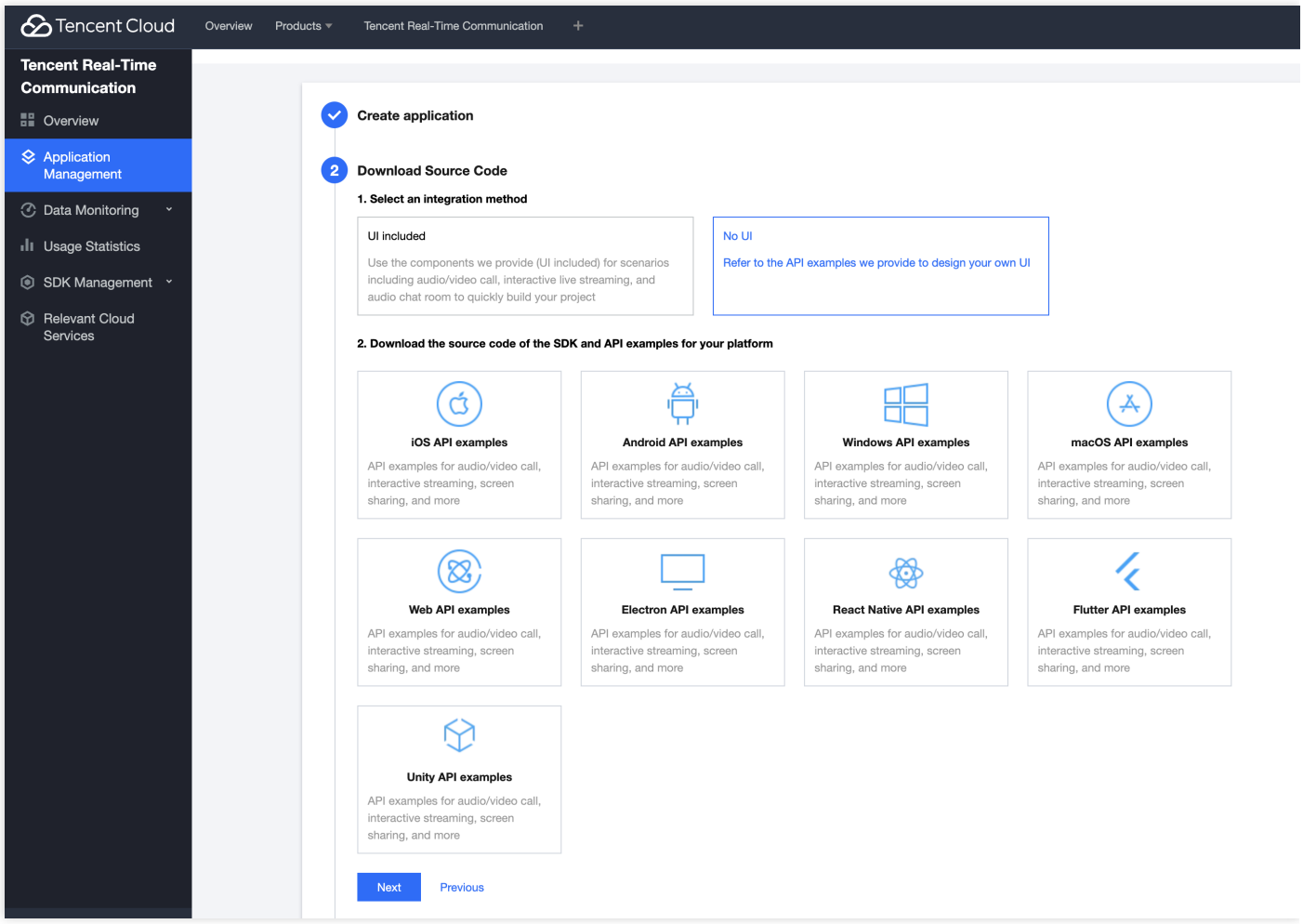
テップ】をクリックします。



ステップ2：サンプルコードのダウンロード

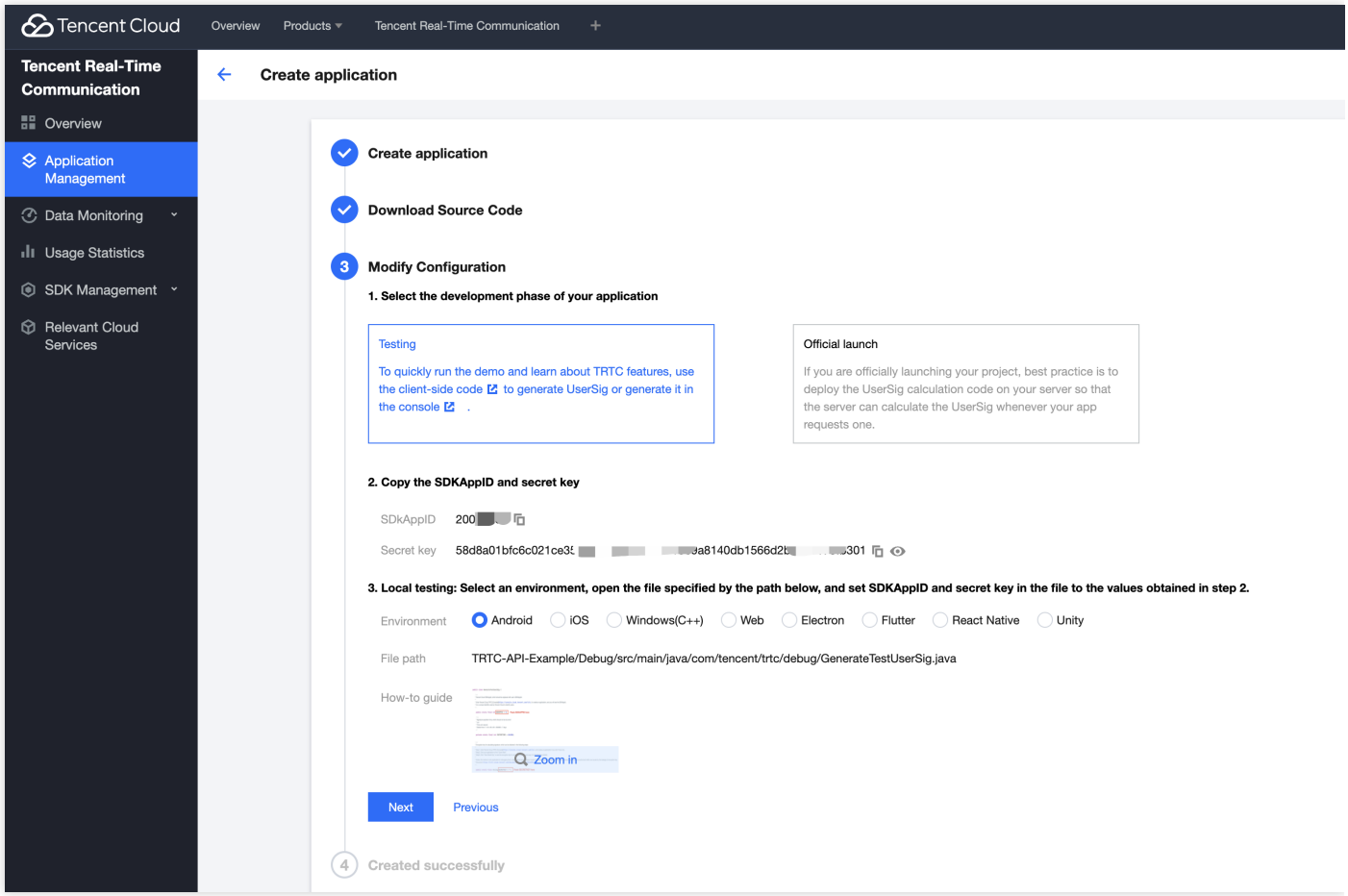
1. UIなしの統合を選択後、ご自身の業務プラットフォームに合わせて、Githubで対応するプラットフォームのサンプルコードをダウンロードします。

2. ダウンロード完了後、【次のステップ】をクリックします。



ステップ3：プロジェクトの設定

1. サンプルプロジェクトのクイックスタート段階で【デバッグ段階】を選択します。その後、ご自身のSDKAppID、Secret keyを記録しておきます。



2. ダウンロードしたサンプルコードを開き、図の指示に従って対応するファイルの場所を探し、ご自身のSDKAppID、Secret keyに変更します。プロジェクトの設定はこれで完了です。【次のステップ】をクリックしてください。

説明：

- ここで言及したUserSigの作成法は、クライアントコードに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TRTC-API-Exampleクイックスタートおよび機能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方法となります。 UserSigが必要なときは、Appから業務サーバーにリクエストを送信し動的にUserSigを取得します。詳細はサーバーでのUserSig新規作成をご参照ください。

手順4：コンパイルと実行

設定完了後、Microsoft Visual Studio（Microsoft Visual Studio 2019の使用を推奨）を使用して、TRTC-API-Example-Qtディレクトリ下のソースコードプロジェクト `QTDemo.sln` を開いてQT環境を設定します（QT5.14バージョンの使用を推奨）。【実行】をクリックすれば体験できます。

よくあるご質問

1. 2台のデバイスで同時にDemoを実行しているのに、お互いの画面が表示されないのはなぜですか。

2台のデバイスでDemoを実行するときは、異なるUserIDを使用していることを確認してください。TRTCでは、2台のデバイスでの同一UserID（SDKAppIDが異なる場合を除く）の同時使用をサポートしていません。

2. ファイアウォールにはどのような制限がありますか。

SDKがUDPプロトコルを使用してオーディオ・ビデオ伝送を行っているため、UDPをブロックするオフィスネットワークでは使用できません。類似した問題がおありの際は、企業ファイアウォール制限の対応をご参照の上、問題及び原因解決にお役立てください。

Web

最終更新日：：2022-12-26 10:43:35

ここでは、主にTencent Cloud TRTC Web SDK Demoを素早く実行する方法をご紹介します。

準備作業

TRTC Web SDK Demoを実行する前に理解すべきの事項。

サポートするプラットフォーム

TRTC Web SDKはWebRTCに基づき実現され、現在、デスクトップとモバイル端末の主流ブラウザをサポートしています。詳細なサポートレベルの表については、サポートしているプラットフォームをご参照ください。
お客様のユースケースは対応表に記載されていない場合は、TRTC Web SDK機能テスト画面を開けて、WebViewなど、現在の環境がWebRTCのすべての機能をサポートしているかどうかをチェックすることができます。

- お客様のユースケースが主に教育シーンである場合は、教師用端末ではElectronソリューションの使用をお勧めし、大小2チャネル画面、よりフレキシブルな画面共有方法およびより強力な弱ネットワークリカバリー機能をサポートしています。

URLドメイン名プロトコルの制限

ブラウザセキュリティポリシーの制限により、WebRTC機能を使用したページへのアクセスプロトコルには厳しい要件があります。以下の表を参照してアプリケーションの開発とデプロイを行ってください。

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	サポートあり	サポートあり	サポートあり	**推奨**
本番環境	HTTPプロトコル	サポートあり	サポートなし	サポートなし	-
ローカル開発環境	http://localhost	サポートあり	サポートあり	サポートあり	**推奨**
ローカル開発環境	http://127.0.0.1	サポートあり	サポートあり	サポートあり	-
ローカル開発環境	http://[ローカルマシンIP]	サポートあり	サポートなし	サポートなし	-

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
ローカル開発環境	file:///	サポートあり	サポートあり	サポートあり	-

ファイアウォールの制限

TRTC Web SDK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。

前提条件

Tencent Cloudの登録によりアカウントを登録している。

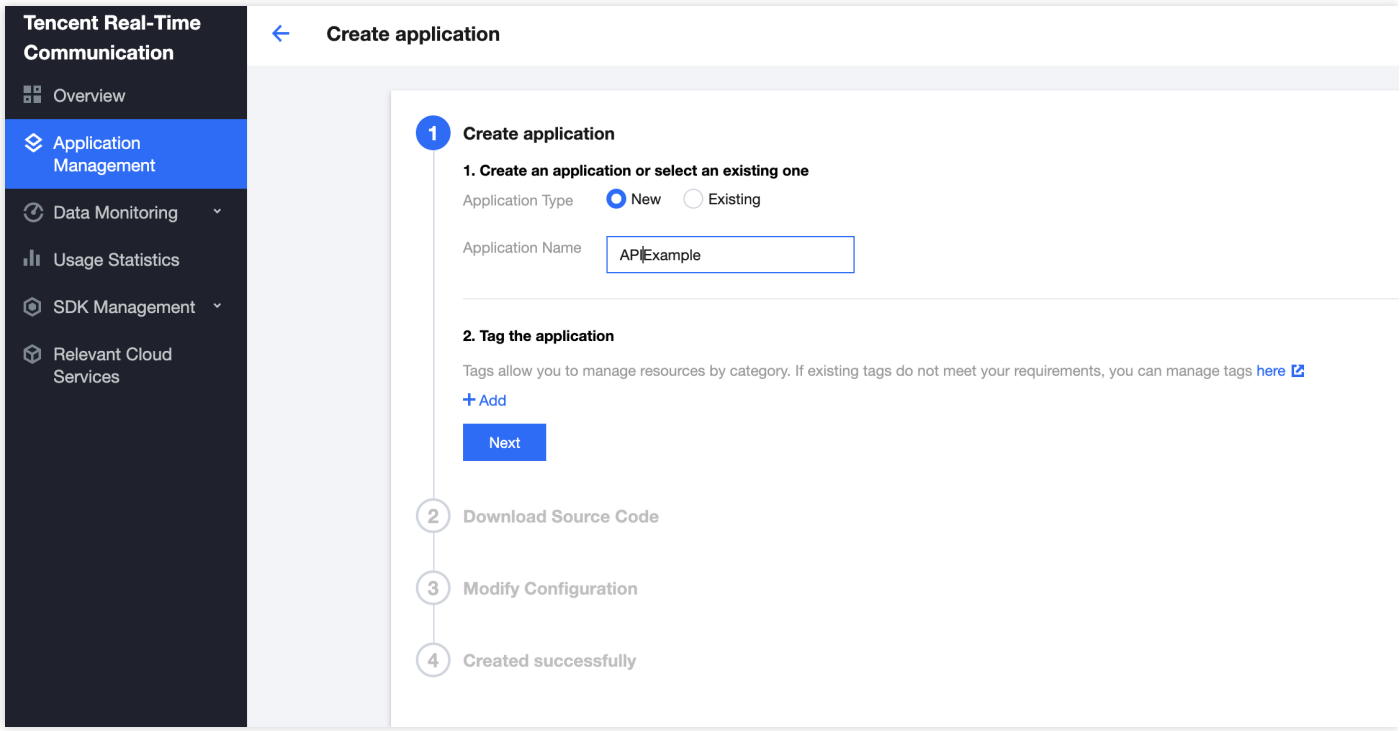
操作手順

手順1：新規アプリケーションの作成

1. TRTCコンソールにログインし、【アプリケーション管理】を選択します。

2. 【アプリケーションの作成】をクリックし、`APIExample` などのアプリケーション名を入力します。すでにアプリケーションを作成している場合は、【既存のアプリケーションを選択】にチェックを入れ、【次のス

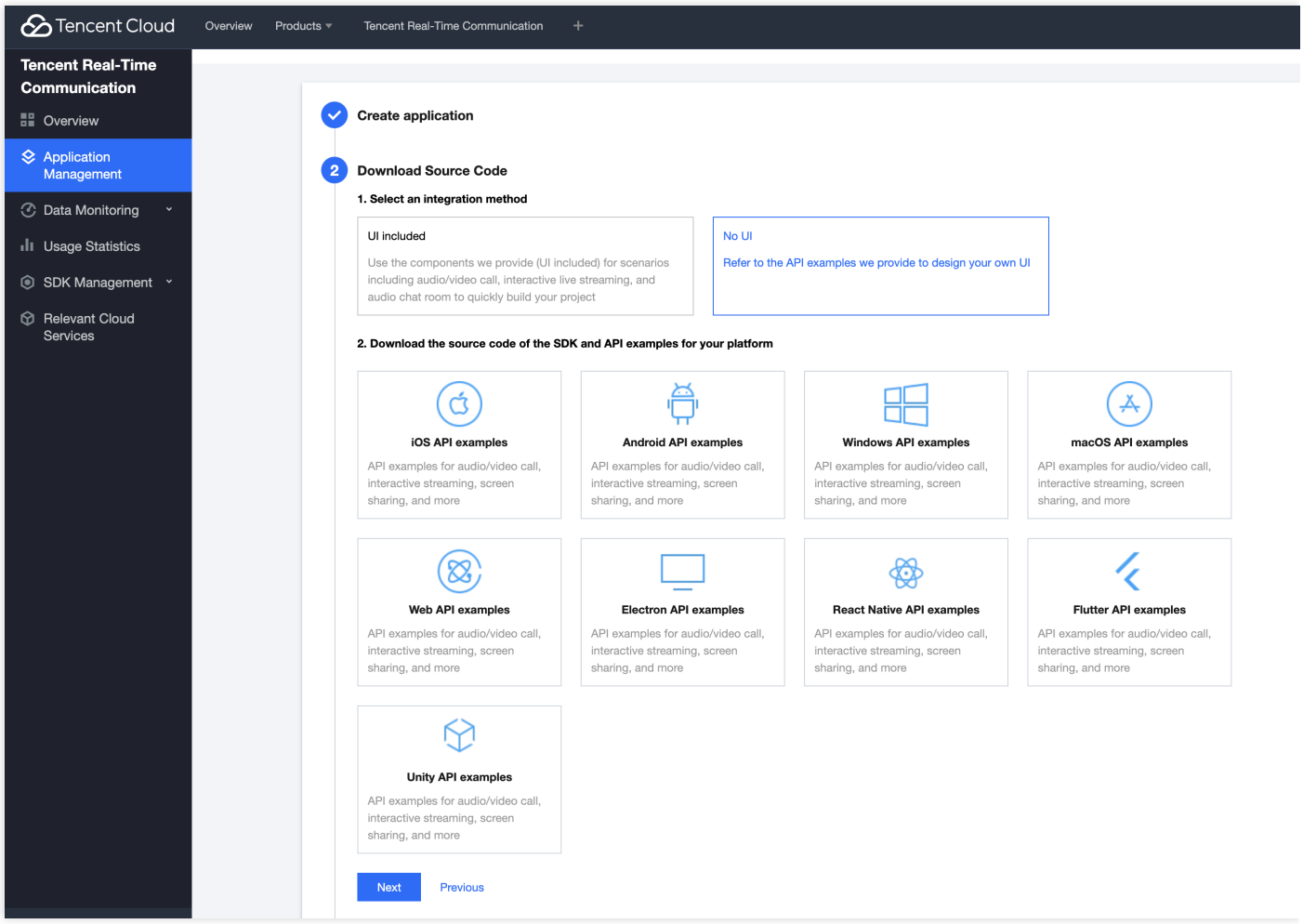
テップ】をクリックします。



ステップ2：サンプルコードのダウンロード

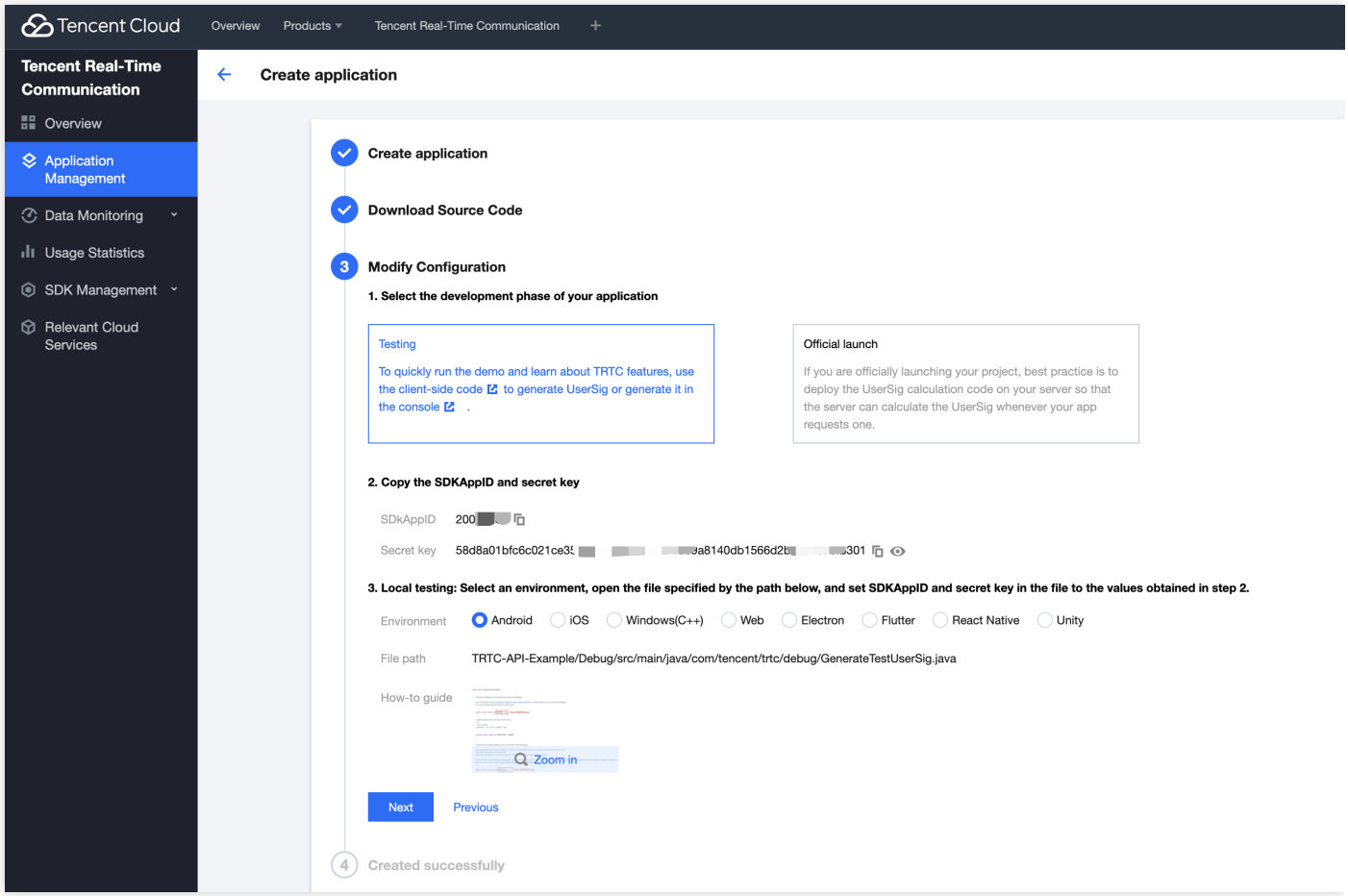
1. UIなしの統合を選択後、ご自身の業務プラットフォームに合わせて、Githubで対応するプラットフォームのサンプルコードをダウンロードします。

2. ダウンロード完了後、【次のステップ】をクリックします。



ステップ3：プロジェクトの設定

1. サンプルプロジェクトのクイックスタート段階で【デバッグ段階】を選択します。その後、ご自身のSDKAppID、Secret keyを記録しておきます。



2. ダウンロードしたサンプルコードを開き、図の指示に従って対応するファイルの場所を探し、ご自身のSDKAppID、Secret keyに変更します。プロジェクトの設定はこれで完了です。【次のステップ】をクリックしてください。

説明：

- ここで言及したUserSigの作成法は、クライアントコードに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TRTC-API-Exampleクイックスタートおよび機能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方法となります。 UserSigが必要なときは、Appから業務サーバーにリクエストを送信し動的にUserSigを取得します。詳細はサーバーでのUserSig新規作成をご参照ください。

よくあるご質問

1. クライアントエラーの発生：“RtcError: no valid ice candidate found”にはどう対処すればよいでしょうか。

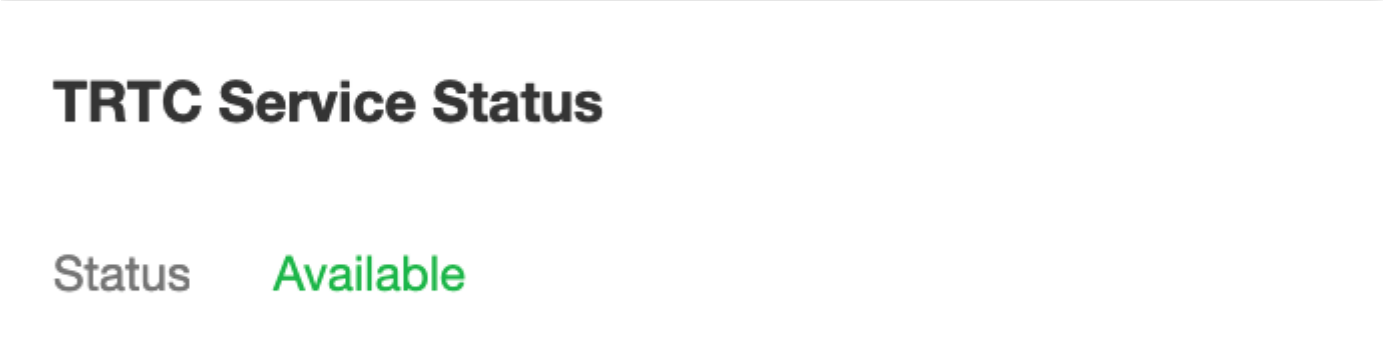
このエラーの発生は、TRTC Web SDKがSTUNホールパンチングに失敗したことを意味します。ファイアウォール制限の対応関連に基づいてファイアウォール設定をチェックしてください。

2. クライアントエラーの発生：「RtcError: ICE/DTLS Transport connection failed"」または「RtcError: DTLS Transport connection timeout」にはどう対処すればよいでしょうか。

このエラーの発生は、TRTC Web SDKがメディア伝送チャネルの確立の際に失敗したことを意味します。ファイアウォール制限の対応関連に基づいてファイアウォール設定をチェックしてください。

3. 10006 error が出現したときの対処方法は？

「Join room failed result: 10006 error: service is suspended,if charge is overdue,renew it」が発生した場合は、Tencent Real-Time Communicationアプリケーションのサーバー状態が正常かどうかご確認ください。
TRTCコンソールにログインし、作成したアプリケーションをクリックし、**アプリケーション情報**をクリックすると、アプリケーション情報パネルでサービスステータスを確認できます。



説明：
その他のよくあるご質問については、Web端末に関するご質問をご参照ください。

Electron

最終更新日：：2022-12-26 10:43:35

ここでは、主にTencent CloudのTRTC-API-Example（Electron）を素早く実行する方法をご紹介します。

前提条件

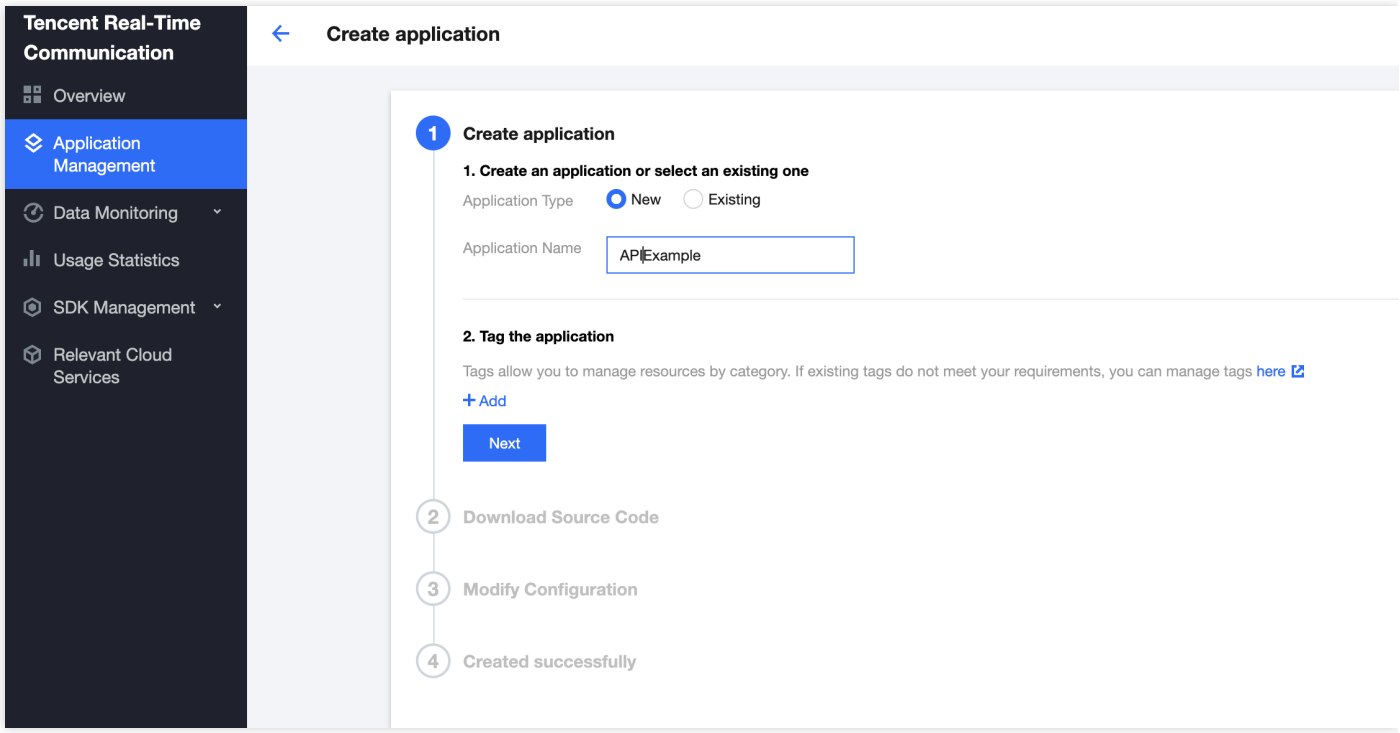
Tencent Cloudの登録によりアカウントを登録している。

操作手順

手順1：新規アプリケーションの作成

1. TRTCコンソールにログインし、【アプリケーション管理】を選択します。

2. 【アプリケーションの作成】をクリックし、`APIExample` などのアプリケーション名を入力します。すでにアプリケーションを作成している場合は、【既存のアプリケーションを選択】にチェックを入れ、【次のステップ】をクリックします。

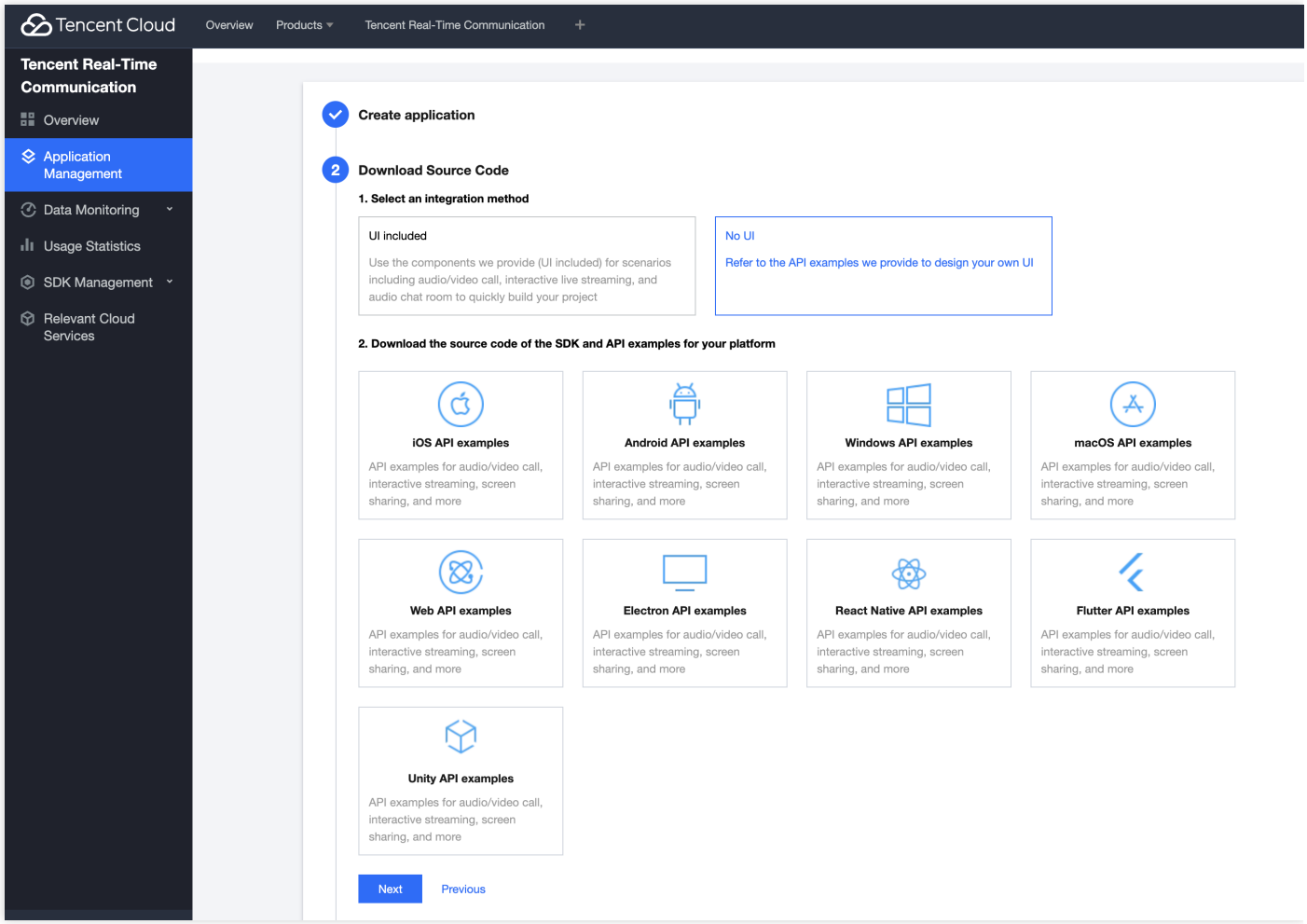


ステップ2：サンプルコードのダウンロード

1. UIなしの統合を選択後、ご自身の業務プラットフォームに合わせて、Githubで、対応するプラットフォームのサンプルコードをダウンロードします。

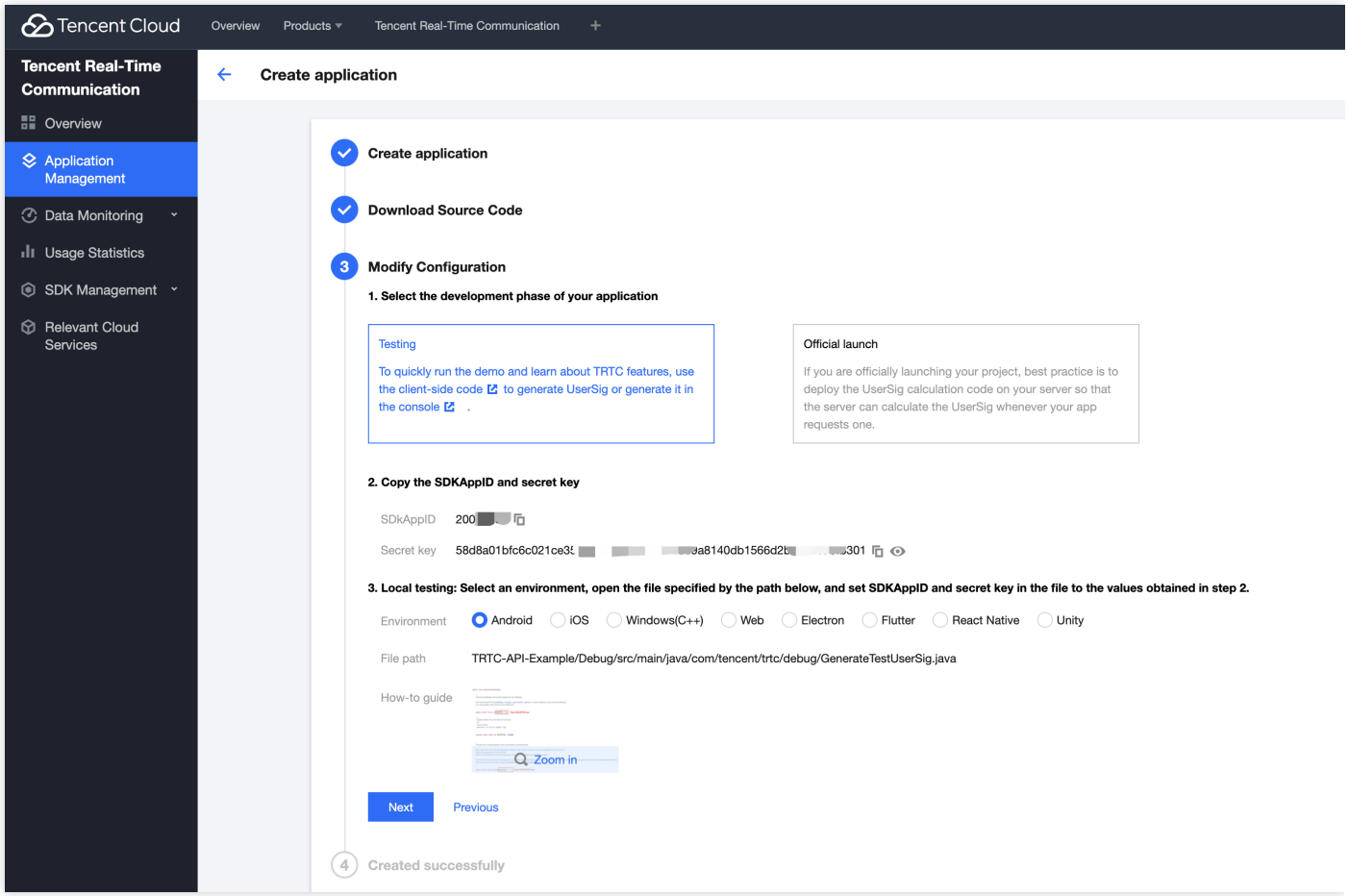
```
git clone https://github.com/tencentyun/TRTCSDK
```

2. ダウンロード完了後、【次のステップ】をクリックします。



ステップ3：プロジェクトの設定

1. サンプルプロジェクトのクイックスタート段階で【デバッグ段階】を選択します。その後、ご自身のSDKAppID、Secret keyを記録しておきます。



2. Electron/TRTC-API-Example/assets/debug/gen-test-user-sig.jsファイルを見つけて開き、ご自身のSDKAppID、Secret keyに変更します。プロジェクトの設定はこれで完了です。【次のステップ】をクリックしてください。

説明：

- ここで言及したUserSigの作成法は、クライアントコードに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TRTC-API-Exampleクイックスタートおよび機能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方法となります。 UserSigが必要なときは、Appから業務サーバーにリクエストを送信し動的にUserSigを取得します。詳細はサーバーでのUserSig新規作成をご参照ください。

よくあるご質問

1. ファイアウォールにはどのような制限がありますか。

SDK が UDP プロトコルを使用してオーディオビデオ伝送を行っていることから、 UDPに対してブロックがあるオフィスネットワークでは使用することができません。類似した問題がおありの際は、 企業ファイアウォール制限の対応をご参照の上、問題及び原因解決にお役立てください。

Flutter

最終更新日：：2022-12-26 10:48:58

ここでは、主にTRTC Demo(Flutter)を素早く実行する方法をご紹介します。

> 注意：
> 現在Windows/MacOs端末は画面共有およびデバイス選択の機能をサポートしていません。

環境要件

- Flutter 2.0およびそれ以降のバージョン。
- **Android端末向け開発：**
  - Android Studio 3.5およびそれ以降のバージョン。
  - AppにはAndroid 4.1およびそれ以降のバージョンのデバイスが必要です。

iOS & macOS端末向け開発：

- Xcode 11.0およびそれ以降のバージョン。
- osxシステムには10.11およびそれ以降のバージョンが必要です。
- プロジェクトが有効な開発者による署名を設定済みであることを確認してください。
- **Windows 端末向け開発：**
  - OS:Windows 7 SP1およびそれ以降のバージョン(x86-64に基づく64ビットOS）。
  - ディスク容量：IDEと一部のツールのインストールに必要な容量を除く、少なくとも1.64 GB以上の空き容量を確保するようにしてください。
  - Visual Studio 2019をインストールします。

前提条件

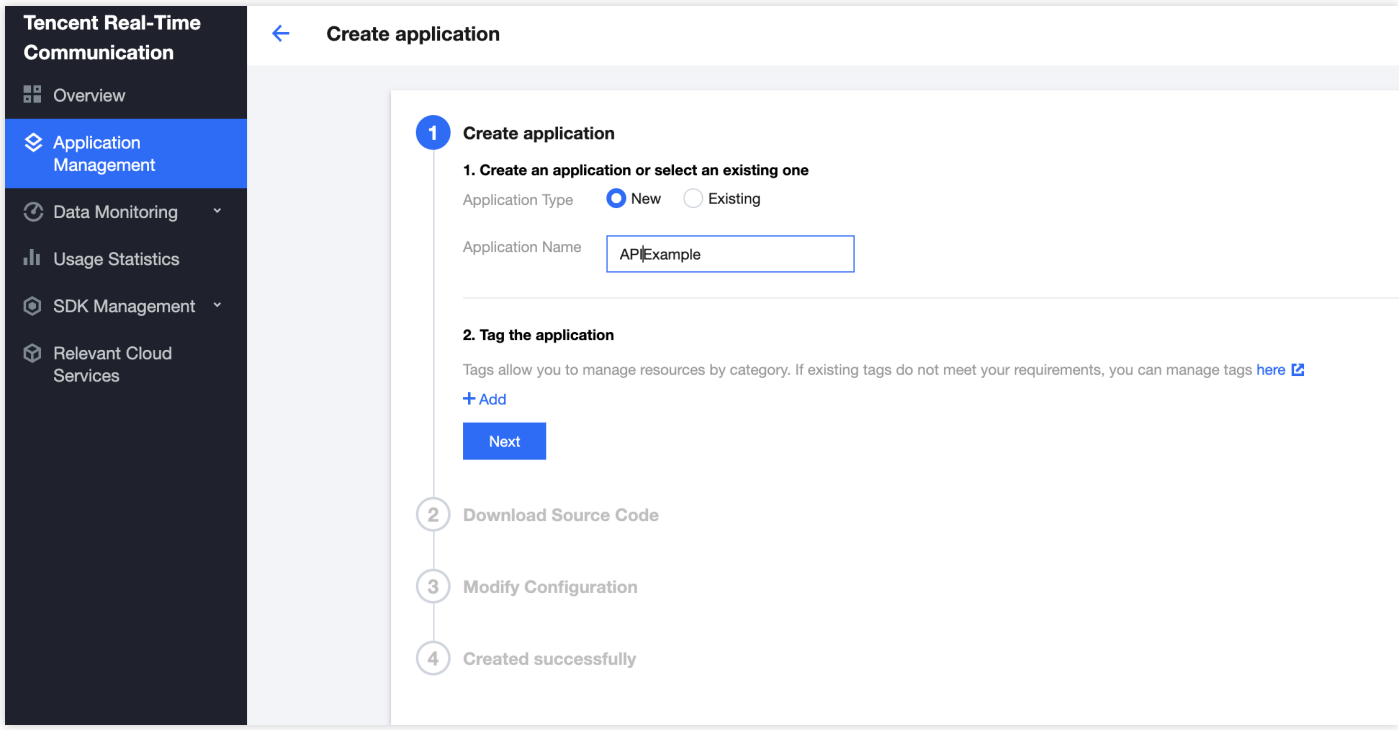
Tencent Cloudアカウントの登録をすでに行っていること。

操作手順

ステップ1：アプリケーションの新規作成

1. TRTCコンソールにログインし、【アプリケーション管理】を選択します。

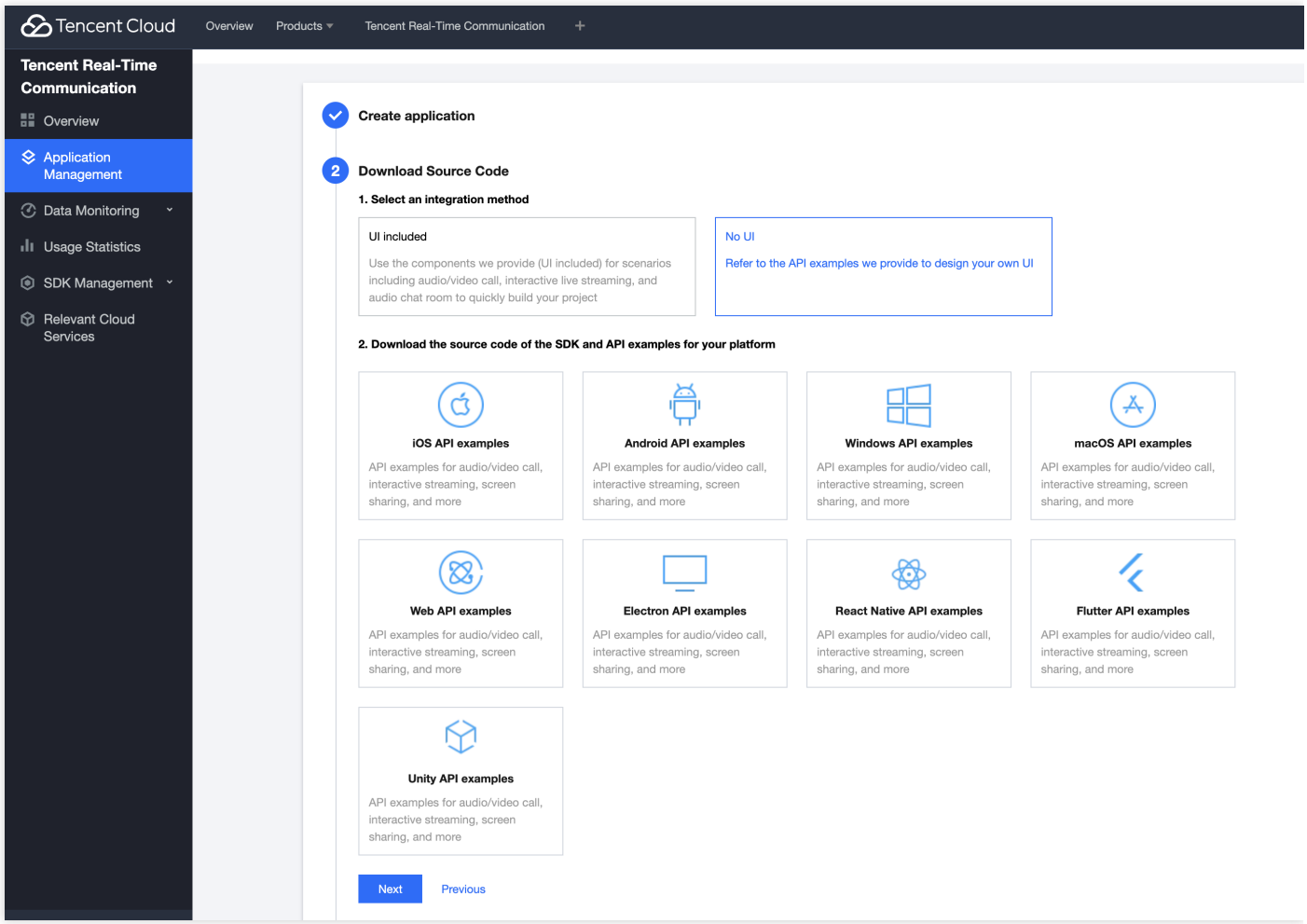
2. 【アプリケーションの作成】をクリックし、`APIExample` などのアプリケーション名を入力します。すでにアプリケーションを作成している場合は、【既存のアプリケーションを選択】にチェックを入れ、【次のステップ】をクリックします。



ステップ2：サンプルコードのダウンロード

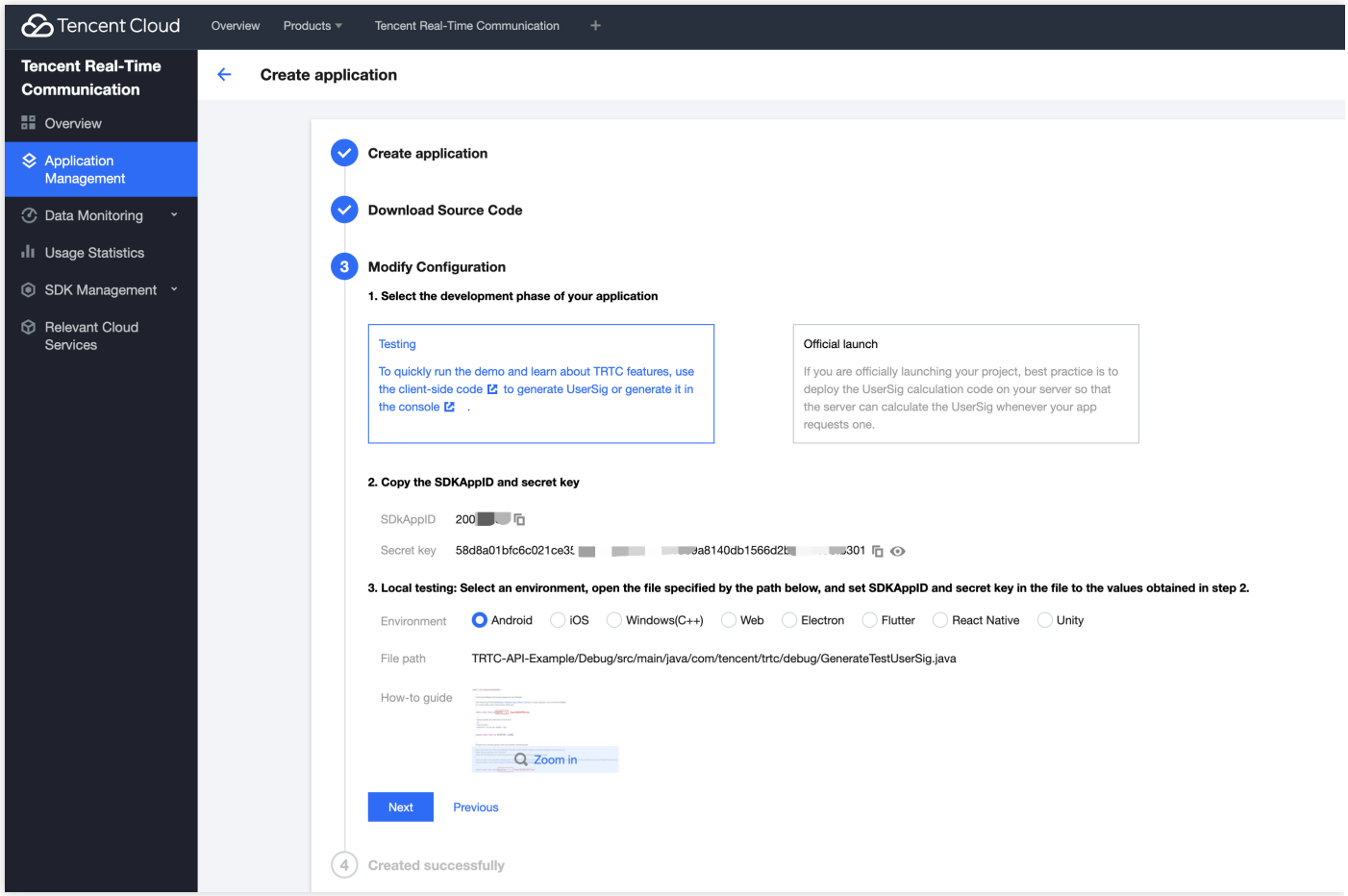
1. UIなしの統合を選択後、ご自身の業務プラットフォームに合わせて、【Github】で、関連するSDKと付属のDemoソースコードをダウンロードします。

2. ダウンロード完了後、【次のステップ】をクリックします。



ステップ3：プロジェクトの設定

1. サンプルプロジェクトのクイックスタート段階で【デバッグ段階】を選択します。その後、ご自身のSDKAppID、Secret keyを記録しておきます。



2. ダウンロードしたサンプルコードを開き、`/lib/debug/GenerateTestUserSig.dart` ファイルを見つけて開き、`GenerateTestUserSig.dart` ファイル内の関連パラメータを設定します。
   - SDKAPPID：デフォルトはPLACEHOLDERです。実際のSDKAppIDを設定してください。
   - SECRETKEY：デフォルトはPLACEHOLDERです。実際のSecret keyを設定してください。
3. プロジェクトの設定はこれで完了です。【次のステップ】をクリックしてください。

説明：

- ここで言及したUserSigの発行方法は、クライアントコードに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TRTC-Simple-Demoクイックスタートおよび機能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方法となります。 UserSigが必要なときは、Appから業務サーバーにリクエストを送信し動的にUserSigを取得します。詳細はサーバーでのUserSig新規作成をご参照ください。

ステップ4：コンパイル実行

1. `flutter pub get` を実行します
2. プロジェクトをコンパイル、実行またはデバッグします。

Androidのデバッグ：

1. `flutter run` を実行できます
2. Android Studio（3.5およびそれ以降のバージョン）を使用して、ソースプロジェクトを開くことができます。【実行】をクリックすれば完了です。

iOSのデバッグ：

1. `cd ios` を実行します
2. `pod install` を実行します
3. XCode（バージョン11.0以上）を使用してソースコードディレクトリ下の `/ios` プロジェクトを開き、Demoプロジェクトをコンパイルして実行すれば完了です。

windowsのデバッグ：

1. windowsサポート `flutter config --enable-windows-desktop` を有効にします
2. `flutter run -d windows` を実行します

macOSのデバッグ

1. macOSサポートを有効にする： `flutter config --enable-macos-desktop` 。
2. `cd macos` を実行します
3. `pod install` を実行します
4. `flutter run -d macos` を実行します

よくあるご質問

TRTCのログはどうやって確認しますか。

TRTCログは、デフォルトで圧縮および暗号化され、接尾辞は `.xlog` です。アドレスは次のとおりです。

- **iOS 端末**：sandbox の `Documents/log` 。
- **Android 端末**：
  - 6.7とそれ以前のバージョン： `/sdcard/log/tencent/liteav`
  - 6.8以後のバージョン： `/sdcard/Android/data/パッケージ名/files/log/tencent/liteav/`

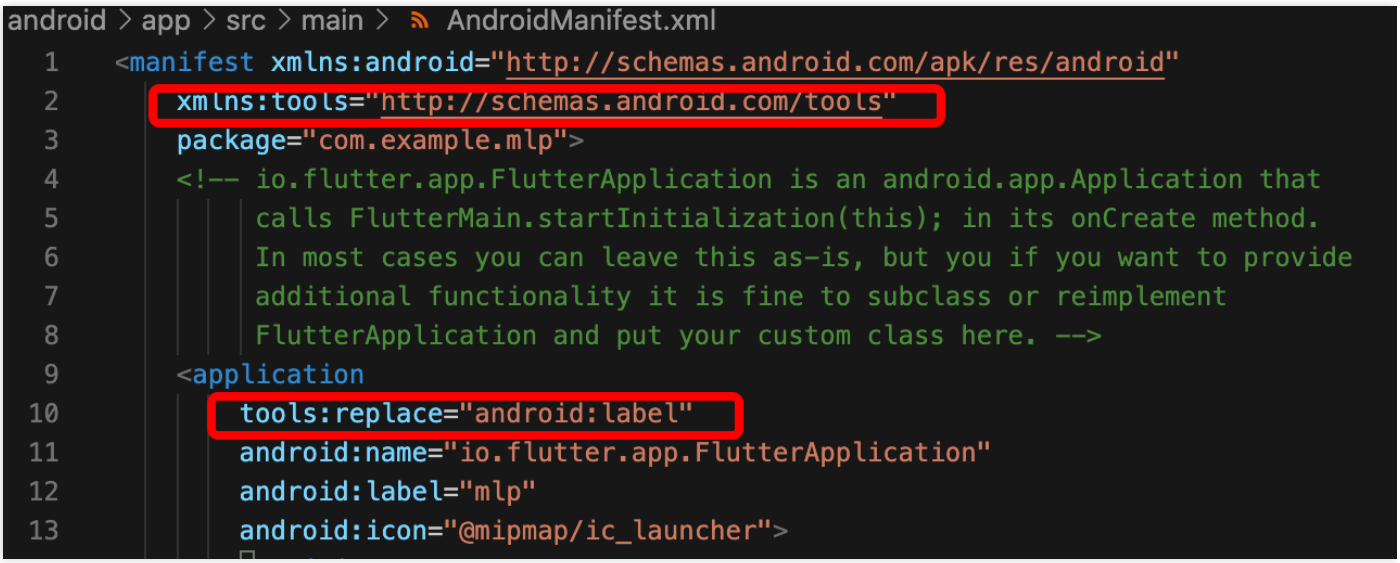
iOSでビデオが表示できない（Androidは正常）場合はどうすればよいですか？

お客様のプロジェクトの info.plist の `io.flutter.embedded_views_preview` の値がYESになっていることを確認してください。

Android Manifest merge failedでコンパイルに失敗した場合は？

`/example/android/app/src/main/AndroidManifest.xml` ファイルを開いてください。

1. `xmlns:tools="http://schemas.android.com/tools"` をmanifestの中に追加します。
2. `tools:replace="android:label"をapplicationの中に追加します。



説明：
詳細については、 Flutterに関するご質問をご参照ください。

Unity

最終更新日：：2022-12-26 10:43:35

このサンプルプロジェクトはUnity内でTRTC SDKのクイックインテグレーションを行い、ゲーム中のオーディオビデオ通話を実現する方法を示しています。

このサンプルプロジェクトには以下の機能が含まれています。

- 通話の参加および通話からの退出。
- ビデオレンダリングのカスタマイズ。
- デバイス管理、音楽の特殊効果および声の特殊効果。

説明：

- 具体的なAPI機能パラメータの説明については、Unity API概要をご参照ください。
- Unityの推奨バージョン： 2020.2.1f1c1。
- 現在、Android、iOS、Windows、Mac（Macはベータ版テスト中です）プラットフォームをサポートしています。
- `Android Build Support`、`iOS Build Support`、`Winodows Build Support`および`MacOs Build Support`モジュールが含まれている必要があります。
- そのうち、iOS端末の開発には以下が必要です。

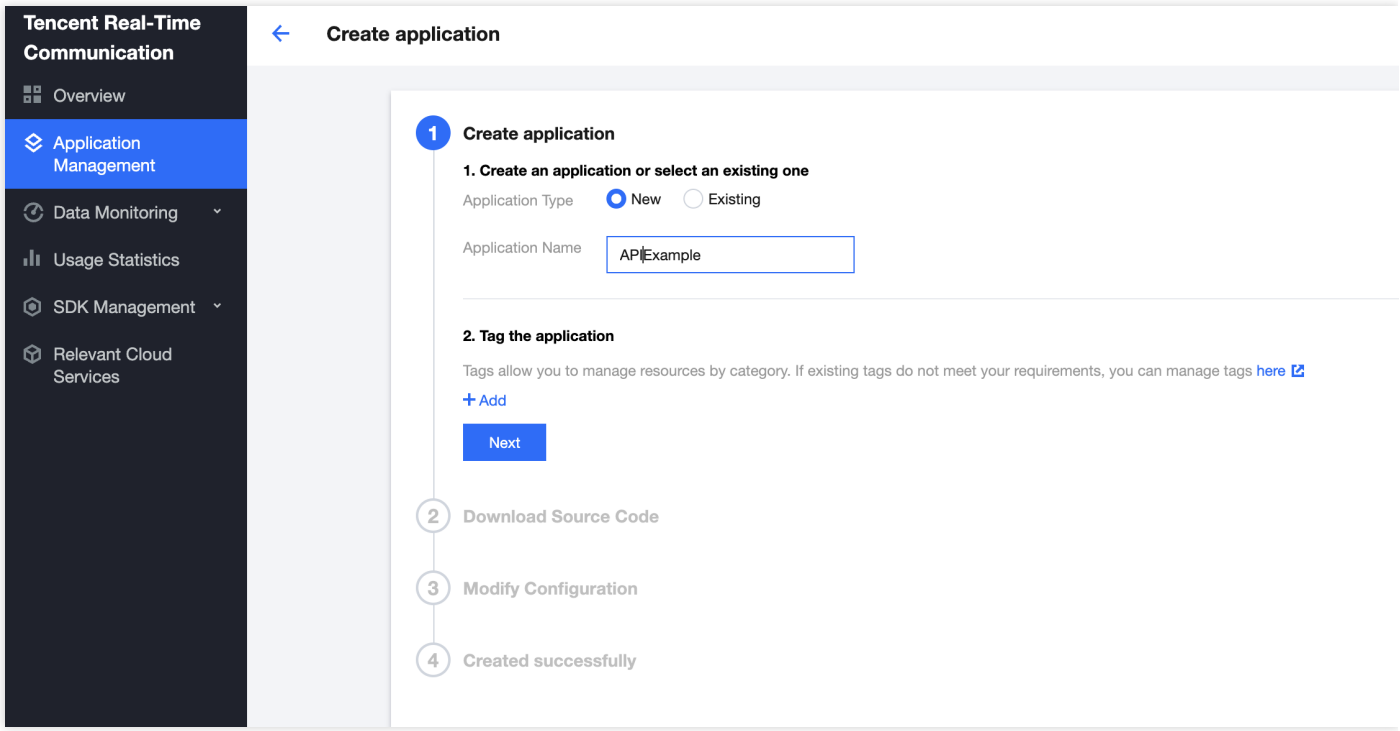
- Xcode 11.0およびそれ以降のバージョン。
- プロジェクトが有効な開発者による署名を設定済みであることを確認してください。

操作手順

ステップ1：アプリケーションの新規作成

1. TRTCコンソールにログインし、【アプリケーション管理】を選択します。
2. 【アプリケーションの作成】をクリックし、`APIExample`などのアプリケーション名を入力します。すでにアプリケーションを作成している場合は、【既存のアプリケーションを選択】にチェックを入れ、【次のス

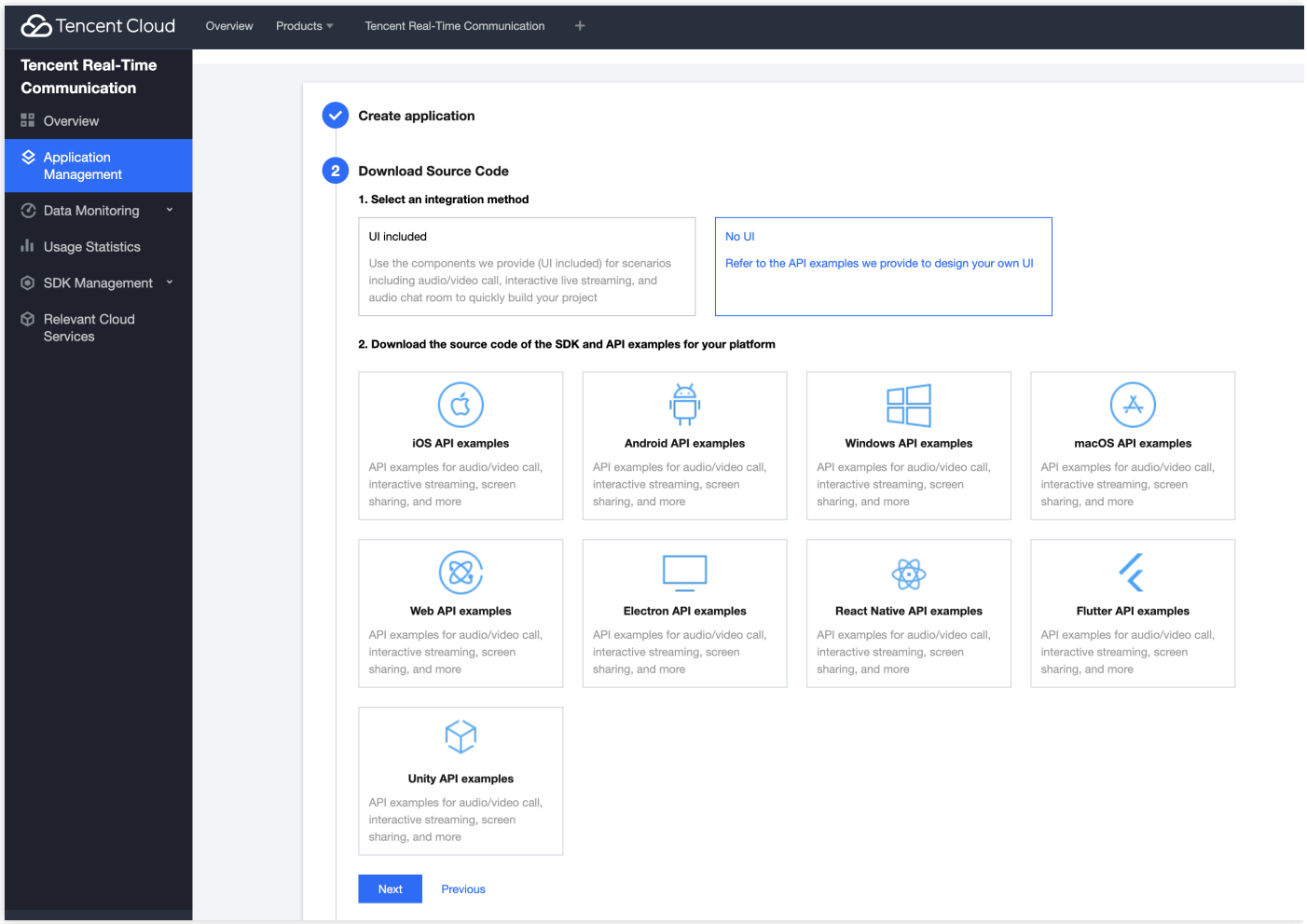
テップ】をクリックします。



ステップ2：サンプルコードのダウンロード

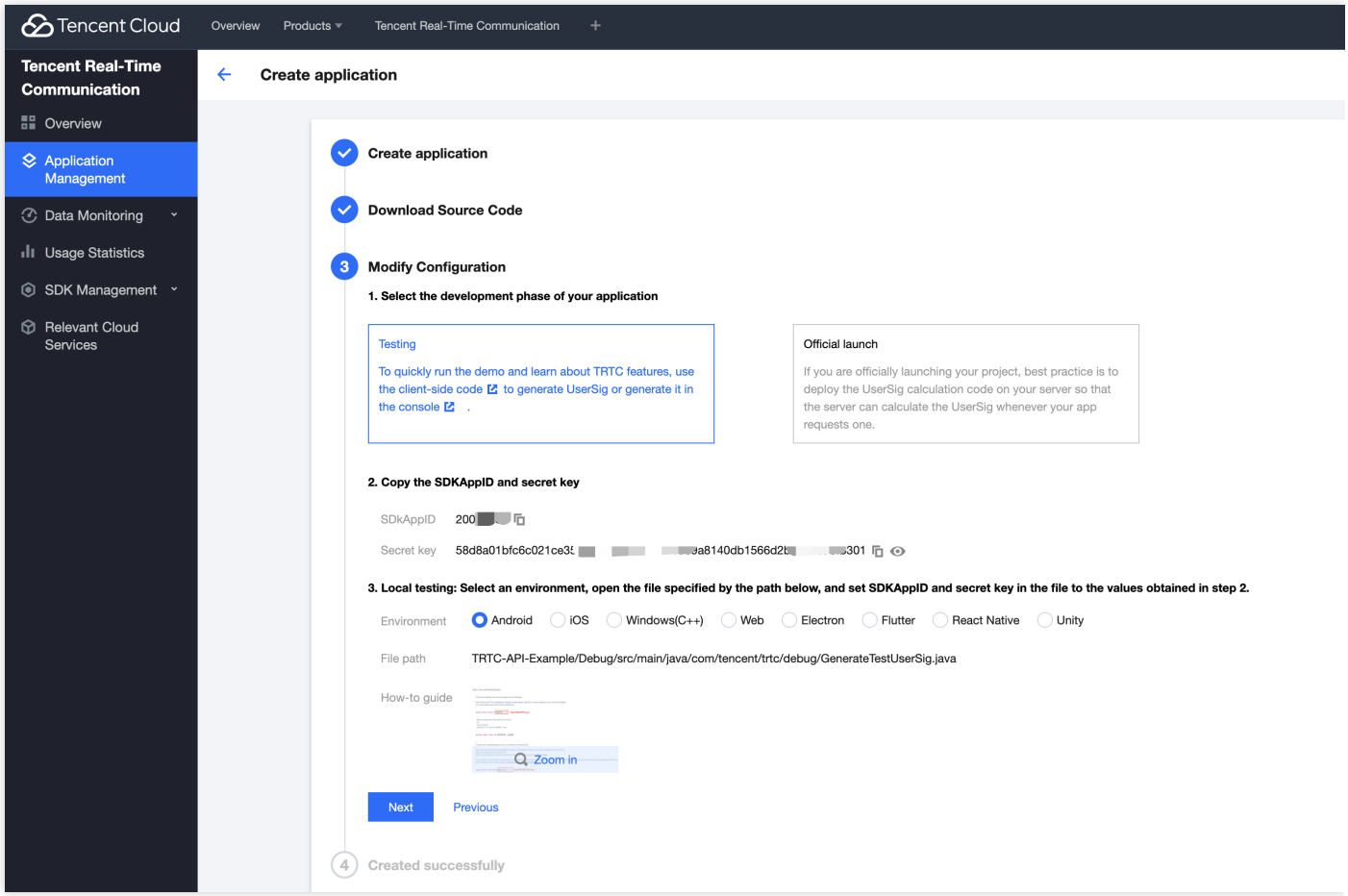
1. UIなしの統合を選択後、ご自身の業務プラットフォームに合わせて、【Github】で、関連するSDKと付属のDemoソースコードをダウンロードします。

2. ダウンロード完了後、【次のステップ】をクリックします。



ステップ3：プロジェクトの設定

1. サンプルプロジェクトのクイックスタート段階で【デバッグ段階】を選択します。その後、ご自身のSDKAppID、Secret keyを記録しておきます。



2. ダウンロードしたサンプルコードを開き、`TRTC-Simple-Demo/Assets/TRTCSDK/Demo/Tools/GenerateTestUserSig.cs` ファイルを見つけて開き、`GenerateTestUserSig.cs` ファイル内の関連パラメータを設定します。

- SDKAPPID：デフォルトはPLACEHOLDERです。実際のSDKAppIDを設定してください。
- SECRETKEY：デフォルトはPLACEHOLDERです。実際のSecret keyを設定してください。

3. プロジェクトの設定はこれで完了です。【次のステップ】をクリックしてください。

説明：

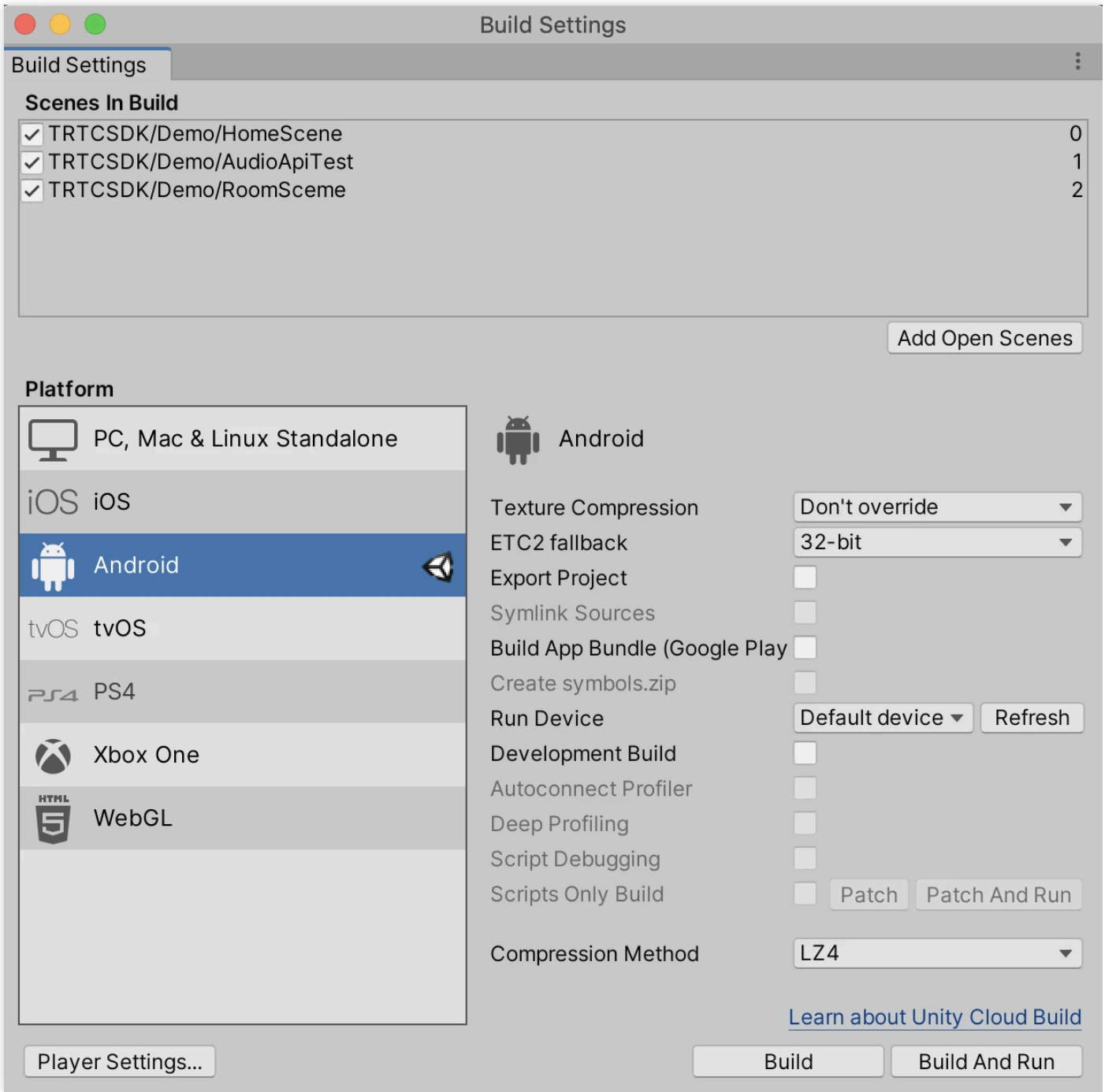
- ここで言及したUserSigの発行方法は、クライアントコードにSECRETKEYを設定しますが、この手法のSECRETKEYは逆コンパイルによって逆クラッキングされやすく、キーがいったん漏洩すると、攻撃者はTencent Cloudトラフィックを盗用できるようになります。そのため**この手法は、ローカルのTRTC-Simple-Demoクイックスタートおよび機能デバッグにのみ適しています**。
- UserSigの正しい発行方法は、UserSigの計算コードをサーバーに統合し、Appのインターフェース向けに提供する方法となります。 UserSigが必要なときは、Appから業務サーバーにリクエストを送信し動的

にUserSigを取得します。詳細はサーバーでのUserSig新規作成をご参照ください。

ステップ4：コンパイル実行

Androidプラットフォーム

1. Unity Editorを設定して、【File】>【Build Setting】をクリックし、Androidに切り替えます。

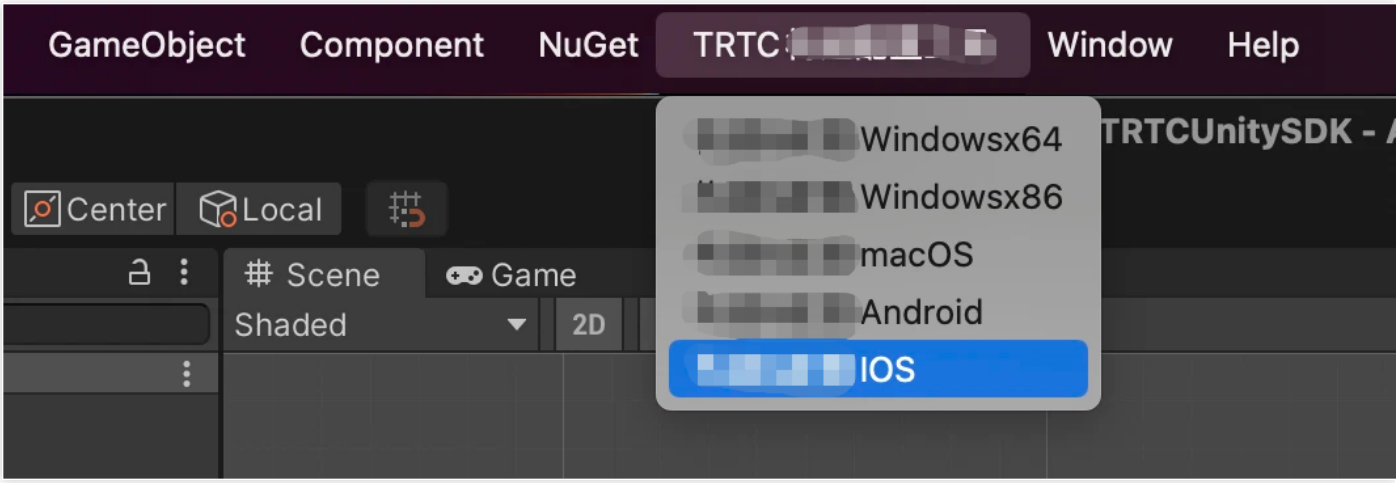


2. Androidの実機に接続して、【Build And Run】をクリックすると、Demoを実行できます。

3. インターフェーステストでは、まずenterRoomの呼び出しをクリックしてから他の関連テストを実行します。データ表示ウィンドウには呼び出しのクリックが成功したことが表示され、もう1つのウィンドウには呼び出した情報が表示されます。

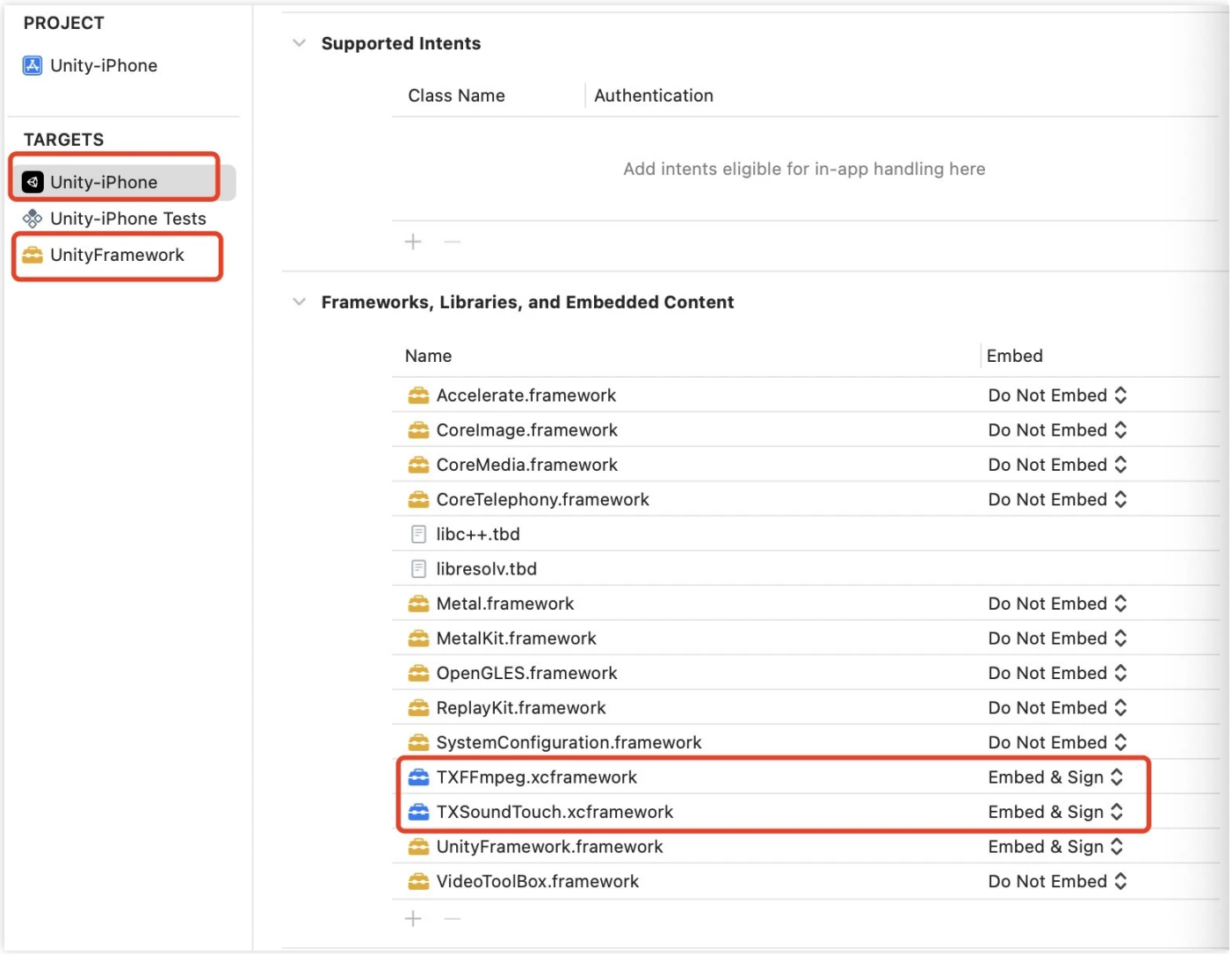
iOSプラットフォーム

1. `TRTC構築設定ツール`（Unityエディタ上部のナビゲーションバーにあります）を開きます
2. `iOSの構築&設定` をクリックし、プロジェクトの生成が完了するのを待ちます



3. 生成したUnity-iPhone.xcodeprojプロジェクトをxcodeで開きます
4. TRTC基盤sdkをダウンロードし、Generalをクリックし、Frameworks,Libraries,and Embedded Contentを選択し、一番下の「+」のアイコンをクリックして必要な動的ライブラリFFmpeg.xcframework、

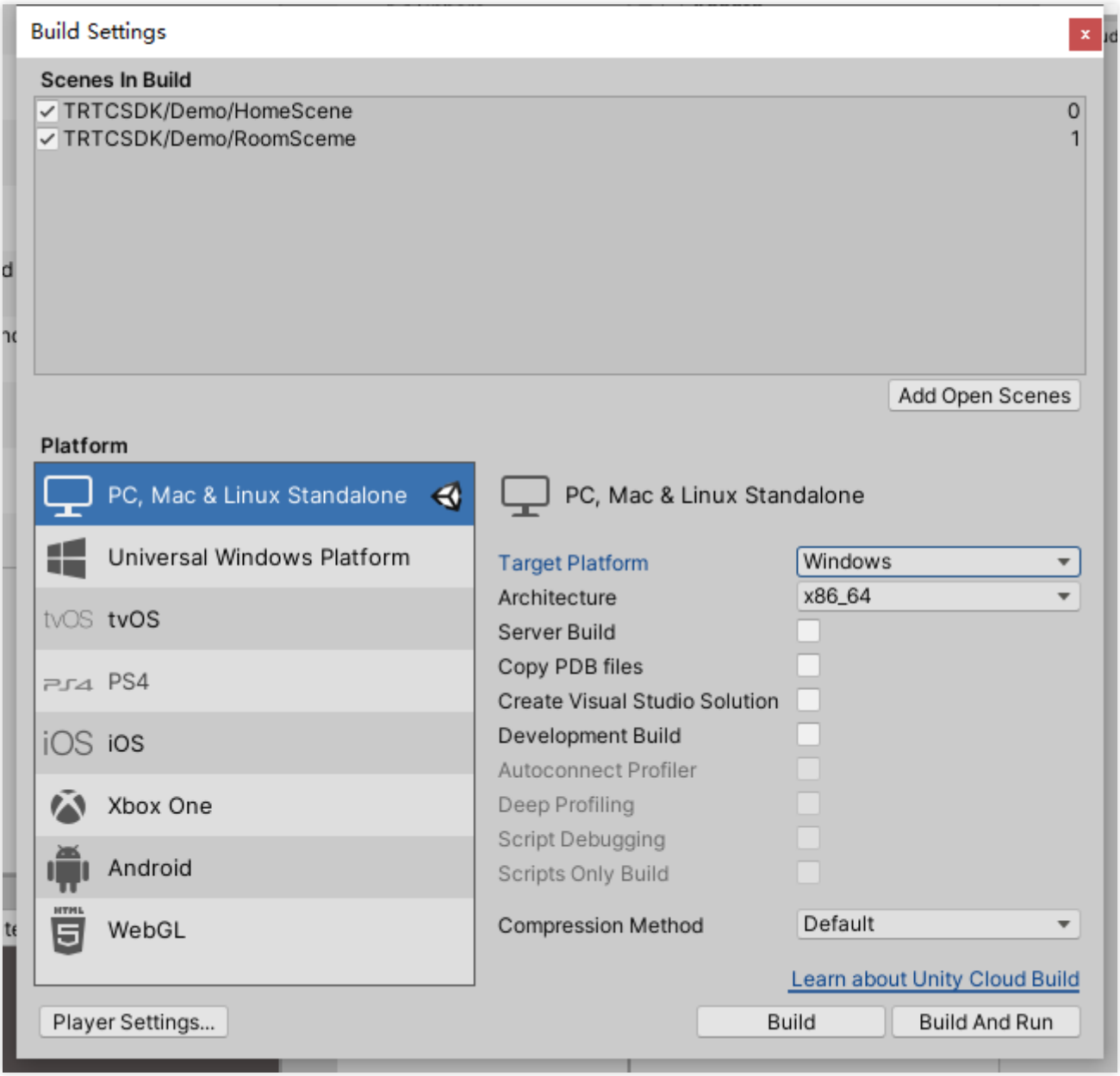
SoundTouch.xcframeworkを順に追加し、Embed & Signを選択します。



5. iOSの実機に接続してデバッグを行います

Windowsプラットフォーム

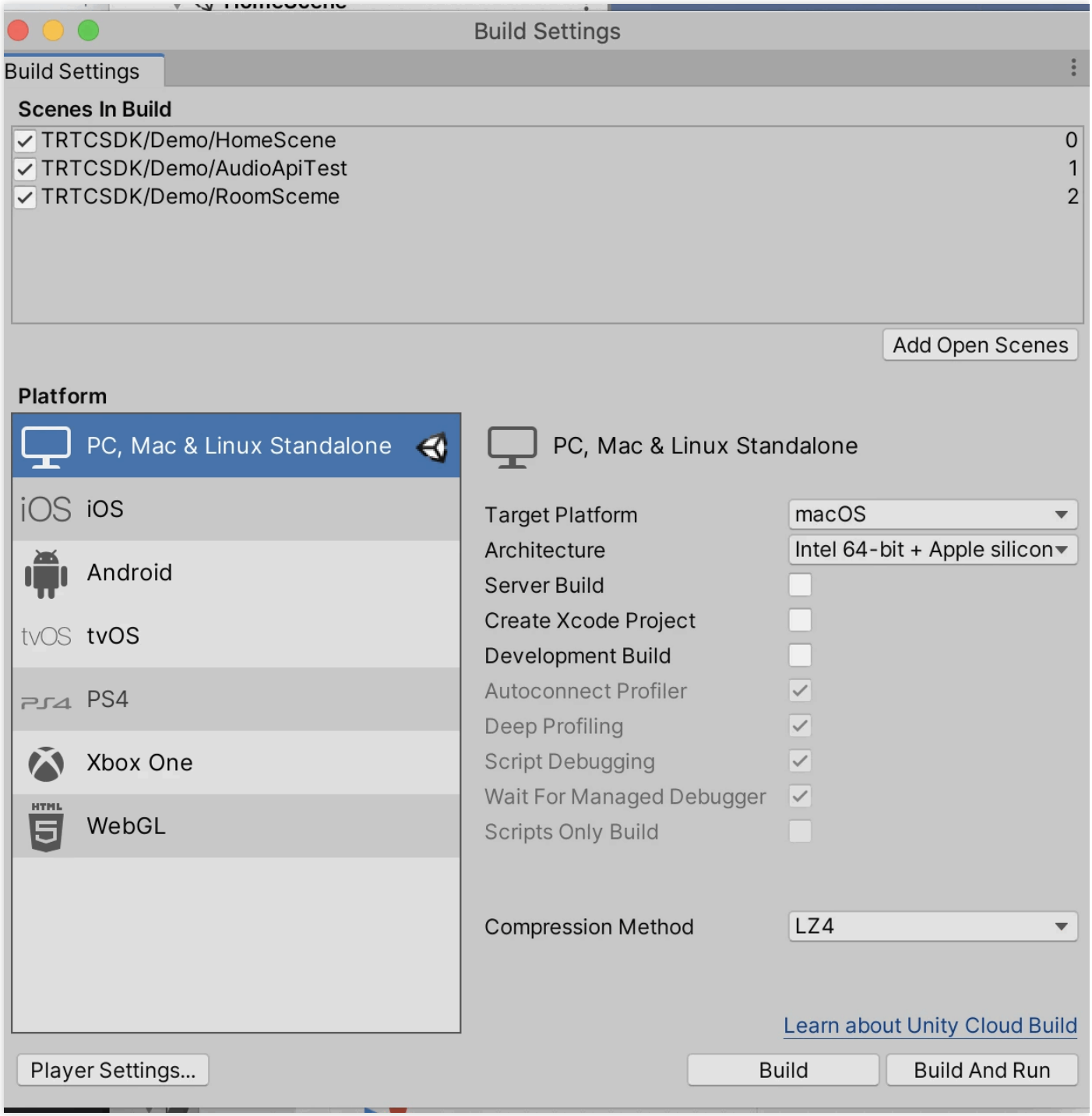
1. Unity Editorを設定して、【File】>【Build Setting】をクリックし、`PC, Mac & Linux Standalone`に切り替え、Target PlatformにWindowsを選択します。



2. 【Build And Run】をクリックすると、Demoを実行できます。

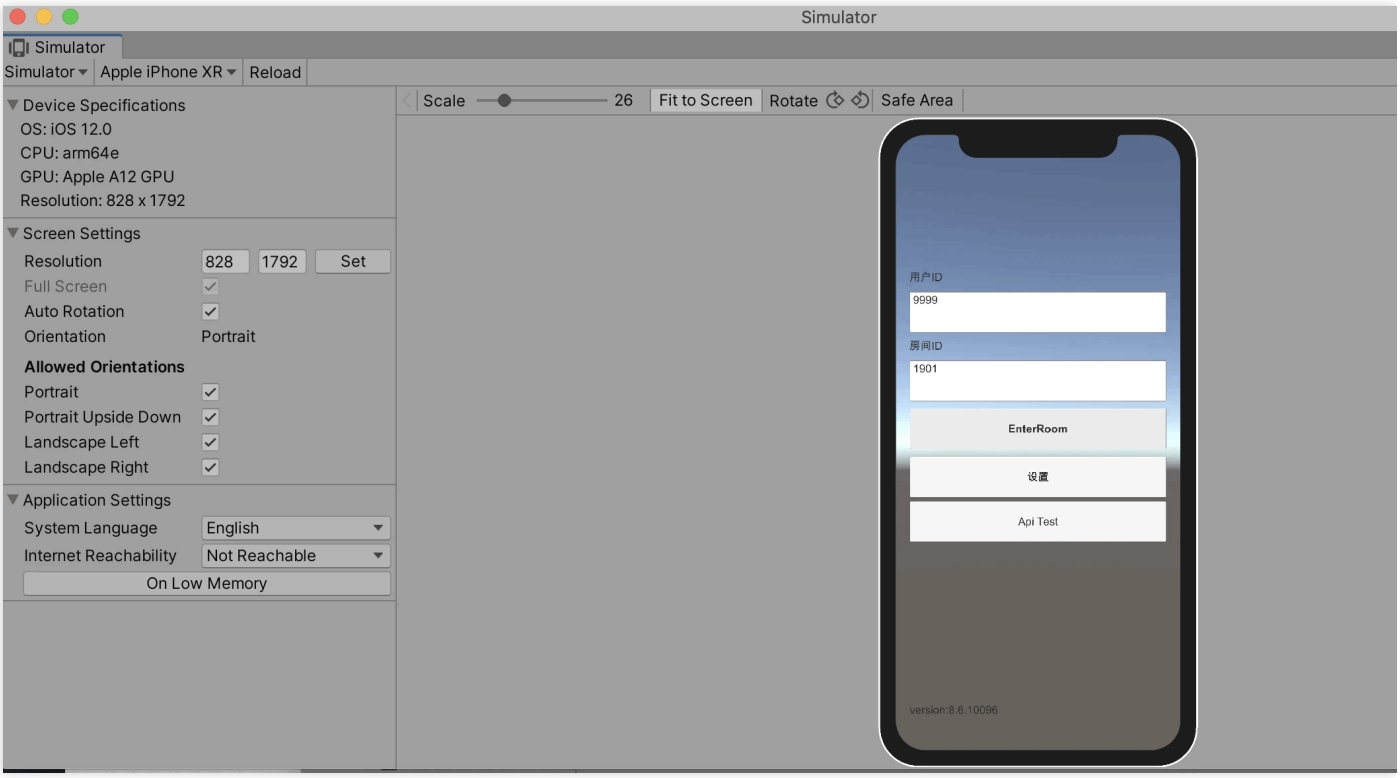
macOSプラットフォーム

1. Unity Editorを設定して、【File】>【Build Setting】をクリックし、`PC, Mac & Linux Standalone`に切り替え、Target PlatformにmacOSを選択します。



2. 【Build And Run】をクリックすると、Demoを実行できます。
3. Unity Editorエミュレーターランタイムを使用して、最初に `Device Simulator Package` をインストールします。

4. 【Window】>【General】>【Device Simulator】をクリックします



Demoサンプル

Demo内にはアップロード済みのAPIの大部分が含まれており、テストおよび参考のために呼び出すことができます。APIドキュメントについてはSDK API（Unity）をご参照ください。

> 説明：
> UIは部分的に調整され更新される可能性があります。最新バージョンを基準としてください。

ディレクトリ構造

```
├─Assets
├── Editor // Unityエディタスクリプト
│ ├── BuildScript.cs // Unityエディタbuildメニュー
│ ├── IosPostProcess.cs // Unityエディタビルドiosアプリケーションスクリプト
├── Plugins
│ ├── Android
```

```
| | ├── AndroidManifest.xml //Androidアプリケーションプロファイル
├── StreamingAssets // Unity Demoオーディオ・ビデオストリーミングファイル
├── TRTCSDK
├── Demo // UnityサンプルDemo
├── SDK // TRTC Unity SDK
├── Implement // TRTC Unity SDK実装
├── Include // TRTC Unity SDKヘッダーファイル
└── Plugins // TRTC Unity SDKの異なるプラットフォーム基盤を実現
```

Integration Guide
1. SDKのプロジェクトへのインポート
iOS

最終更新日：：2022-10-31 14:41:31

ここではSDKをプロジェクトにインポートする方法をご紹介します。

開発環境要件

- Xcode 9.0+。
- iOS 9.0以上のiPhoneまたはiPadの実機。
- プロジェクトには有効な開発者の署名を設定。

第1段階：SDKのインポート

CocoaPodsソリューションを使用するか、またはまずSDKをローカルにダウンロードしてから手動で現在のプロジェクトにインポートするかを選択できます。

方法1：CocoaPodsを使用

1. **CocoaPodsのインストール**
   ターミナルポートに以下のコマンド（事前にMacにRuby環境をインストールする必要があります）を入力します。

```
sudo gem install cocoapods
```

2. **Podfileファイルの新規作成**
   項目のあるパスに入り、以下のコマンドラインを入力すると項目パスに一つのPodfileファイルが現れます。

```
pod init
```

3. **Podfileファイルの編集**
プロジェクトのニーズに応じて適切なバージョンを選択し、Podfileファイルを編集します。

- **オプション1：TRTC簡易版**
インストールパッケージのボリューム増分が最も小さく、Tencent Real-Time Communication（TRTC）とライブストリーミングプレーヤー（TXLivePlayer）という2つの機能だけをサポートしています。このバージョンを選択した場合は、次の方法でPodfileファイルの編集を行ってください。

```
platform :ios, '8.0'

target 'App' do
pod 'TXLiteAVSDK_TRTC', :podspec => 'https://liteav.sdk.qcloud.com/pod/liteavsd
kspec/TXLiteAVSDK_TRTC.podspec'
end
```

- **オプション2：プロフェッショナル版（Professional）**
Tencent Real-Time Communication（TRTC）、ライブストリーミングプレーヤー（TXLivePlayer）、RTMPプッシュ（TXLivePusher）、VODプレーヤー（TXVodPlayer）およびショートビデオレコーディングおよび編集（UGSV）などの多くの機能が含まれます。このバージョンを選択した場合は、次の方法でPodfileファイルの編集を行ってください。

```
platform :ios, '8.0'

target 'App' do
pod 'TXLiteAVSDK_Professional', :podspec => 'https://liteav.sdk.qcloud.com/pod/
liteavsdkspec/TXLiteAVSDK_Professional.podspec'
end
```

4. **SDKの更新およびインストール**

- ターミナルポートに以下のコマンドを入力し、ローカルライブラリファイルを更新し、SDKをインストールします。

```
pod install
```

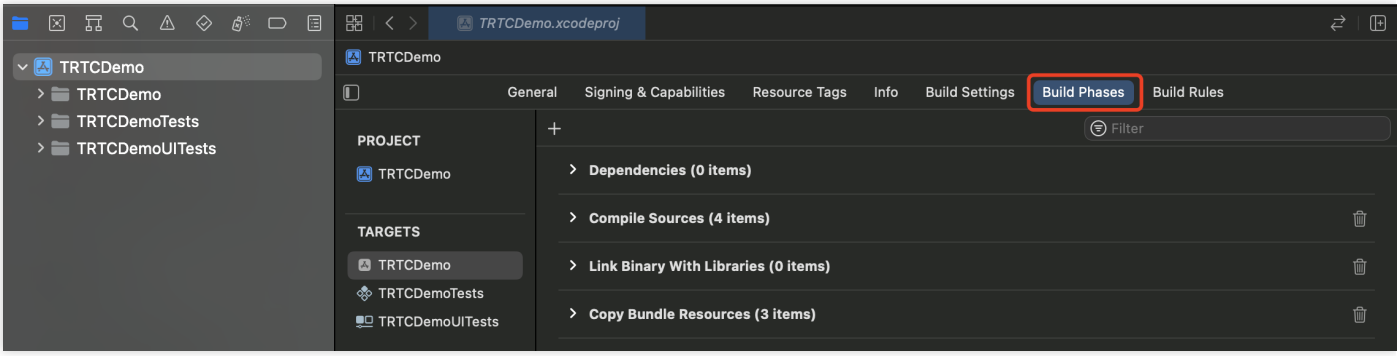
- または以下のコマンドを使用しローカルライブラリバージョンを更新します。

```
pod update
```

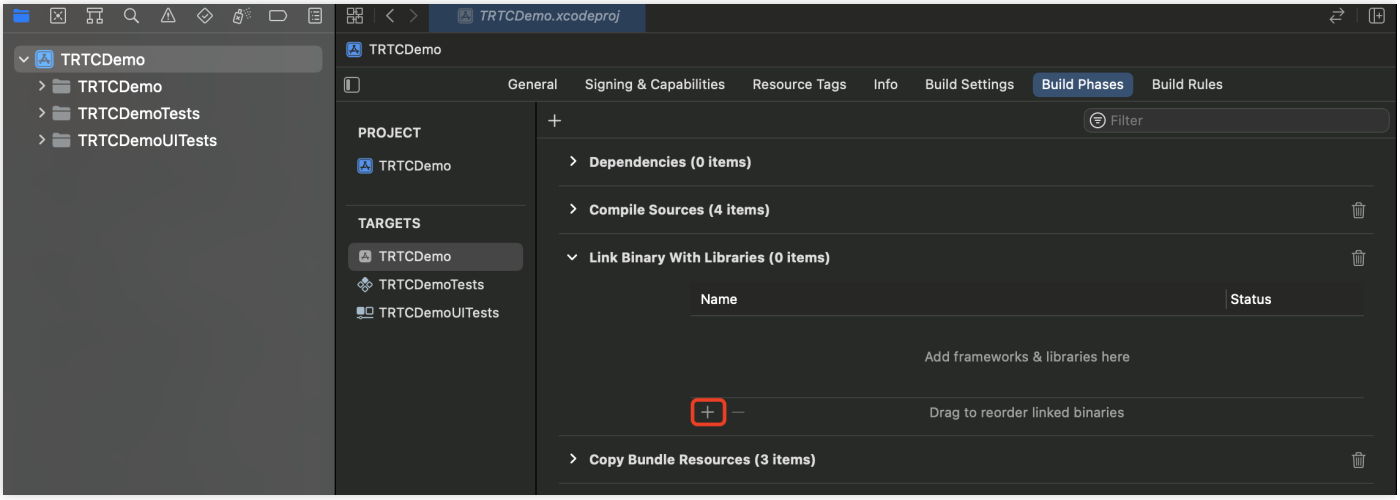
podコマンドが実行されると、SDKに統合された.xcworkspaceサフィックスが付いたプロジェクトファイルが作成され、ダブルクリックして開くことができます。

方法2：SDKのダウンロードと手動でのインポート

1. SDKをダウンロードしてローカルに解凍します。
2. Xcodeプロジェクトを開き、動作させたいtargetを選択し、**Build Phases** 項目を選択します。

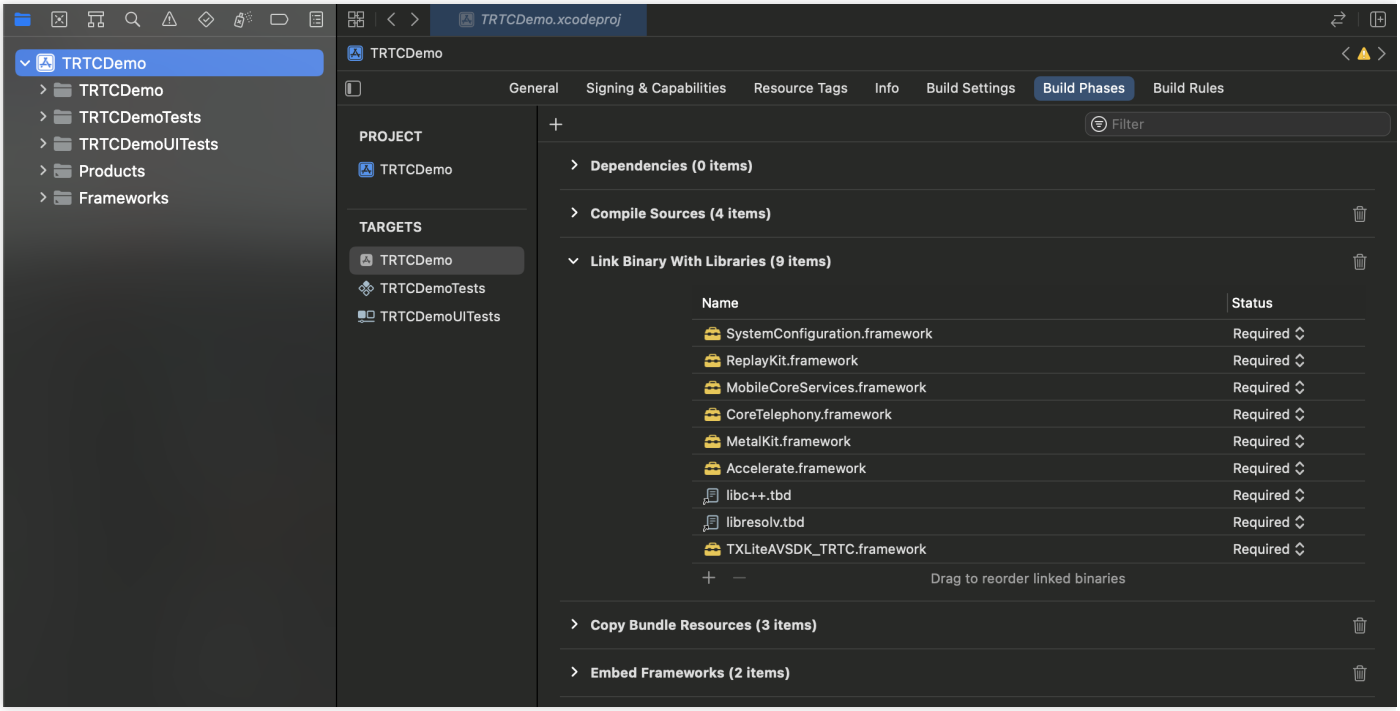


3. **Link Binary with Libraries** をクリックして展開し、下辺の“+”のアイコンをクリックして依存ライブラリを追加します。

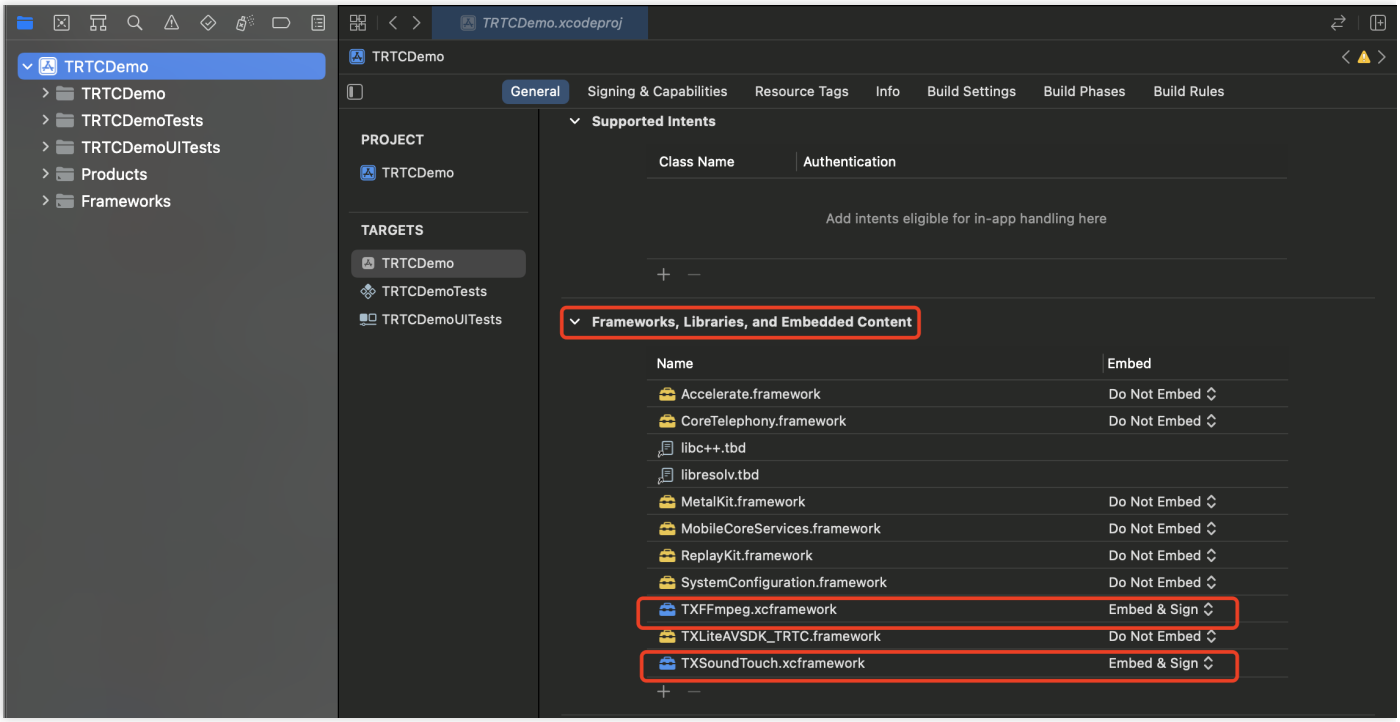


4. ダウンロードした `TXLiteAVSDK_TRTC.Framework`（または `TXLiteAVSDK_Professional.Framework`）、`TXFFmpeg.xcframework`、`TXSoundTouch.xcframework`、およびその必要な依存ライブラリ `GLKit.framework`、`AssetsLibrary.framework`、`SystemConfiguration.framework`、`libsqlite3.0.tbd`、`CoreTelephony.framework`、`AVFoundation.framework`、`OpenGLES.fra`

`mework` 、 `Accelerate.framework` 、 `MetalKit.framework` 、 `libresolv.tbd` 、 `MobileCore Services.framework` 、 `libc++.tbd` 、 `CoreMedia.framework` を順に追加します。

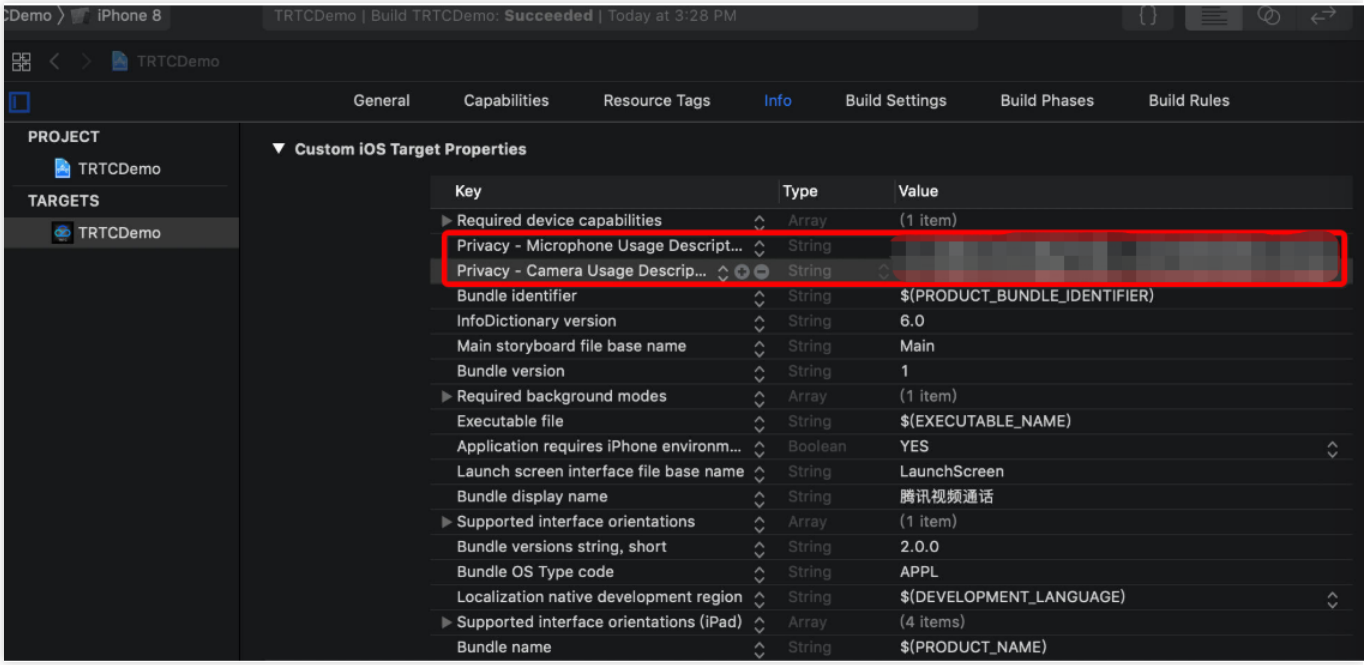


5. **General**をクリックして**Frameworks,Libraries,and Embedded Content**を選択し、TXLiteAVSDK_TRTC.frameworkに必要な動的ライブラリ**TXFFmpeg.xcframework**、**TXSoundTouch.xcframework**がすでに追加されているか、**Embed & Sign**が正しく選択されているかを確認し、まだであれば一番下の“**+**”のアイコンをクリックして順に追加します。



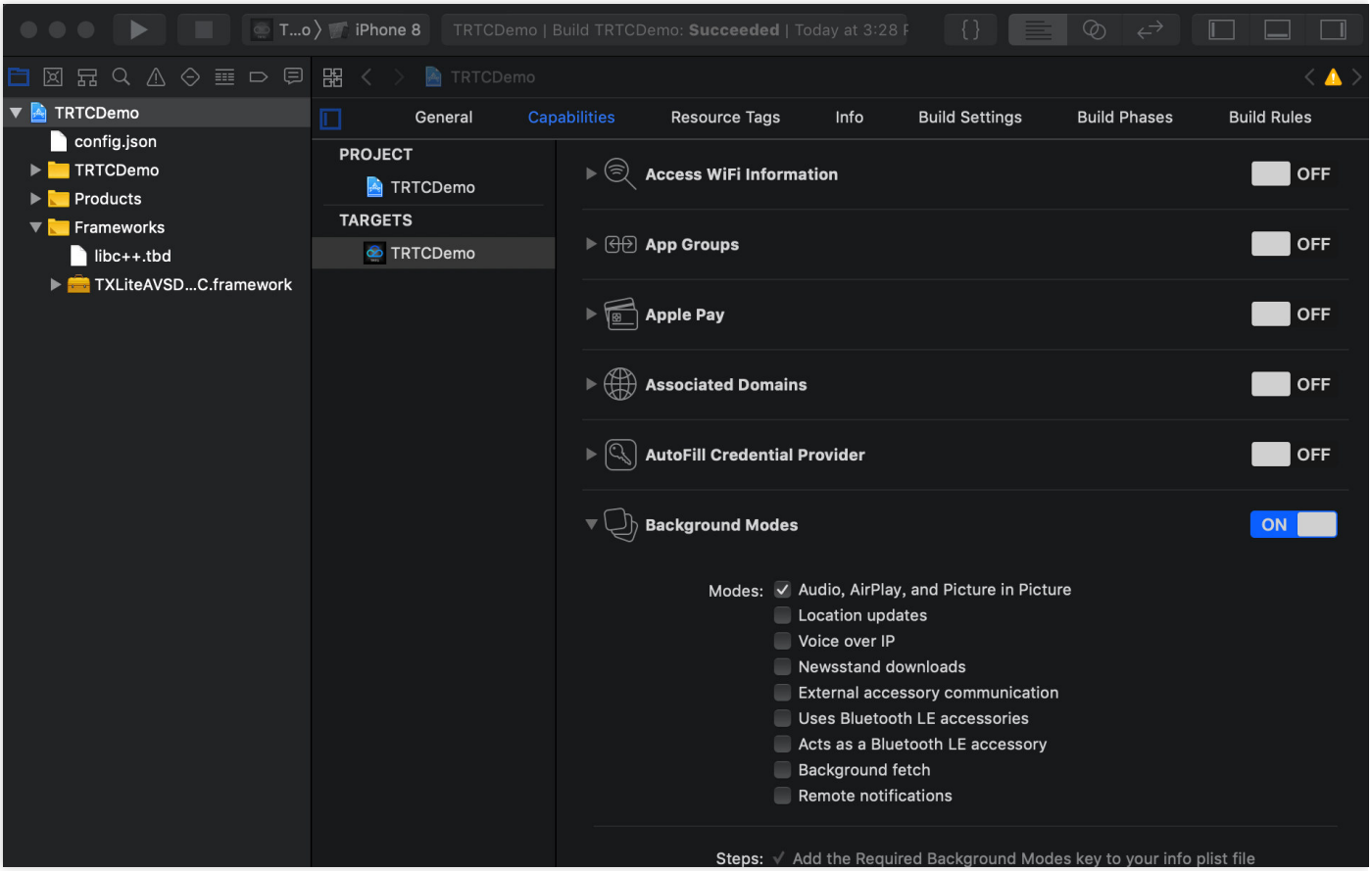
第2段階：App権限の設定

1. SDKの提供するオーディオビデオ機能を使用したい場合は、Appにマイクとカメラの使用権限を承認する必要があります。AppのInfo.plistに次の2項目を追加します。これらはそれぞれマイクとカメラについて、システムから権限承認ダイアログボックスがポップアップされる際に表示される情報に対応します。
   - **Privacy - Microphone Usage Description**、かつマイクの使用目的のメッセージを記入します。
   - **Privacy - Camera Usage Description**、かつカメラの使用目的のメッセージを入力します。



2. Appがバックエンドに入っても関連の機能を実行するようにしたい場合は、下図に示すように、XCodeで現在のプロジェクトを選択し、**Capabilities**で設定項目**Background Modes** を**ON**に設定し、**Audio，AirPlay and**

Picture in Picture にチェックを入れます。



第3段階：プロジェクトにおけるSDKの参照

第1段階のインポートと第2段階のデバイスへの権限承認が完了すると、SDKの提供するインターフェースAPIをプロジェクトで参照できるようになります。

Objective-Cまたは Swiftインターフェースによる参照

Objective-CまたはSwiftコードの中でSDKを使用する方法は2つあります。

- **モジュールの引用**：プロジェクトでSDK APIを使用する必要のあるファイルにモジュール参照を追加します。

```
@import TXLiteAVSDK_TRTC;
```

- **ヘッダーファイルの引用**：プロジェクトでSDK APIを使用する必要のあるファイルに特定のヘッダーファイルを導入します。

```
#import "TXLiteAVSDK_TRTC/TRTCCloud.h"
```

説明：
Objective-Cインターフェースの使用方法については、 iOS&Mac APIの概要をご参照ください。

C++インターフェースによる参照(オプション)

プロジェクトがQTまたはElectronのようなクロスプラットフォームフレームワークによってSDKを参照している場合は、 `TXLiteAVSDK_TRTC.framework/Headers/cpp_interface` ディレクトリ下のヘッダーファイルを参照してください。

```
#include "TXLiteAVSDK_TRTC/cpp_interface/ITRTCCloud.h"
```

説明：
C++インターフェースの使用方法については、 全プラットフォーム（C++）APIの概要をご参照ください。

Android

最終更新日：：2022-07-11 11:30:27

このドキュメントでは、SDKのプロジェクトへのインポート方法を紹介します：

開発環境要件

- Android Studio 3.5+。
- Android 4.1 (SDK API 16）)以上のシステム。

手順1：SDKのインポート

方法１：自動ロード(aar)

TRTC SDK はmavenCentralライブラリにリリースされています。gradleを構成することにより、自動的にダウンロードと更新できます。

1. dependenciesの中にTRTCSDKの依存を追加します。

```
dependencies{
implementation 'com.tencent.liteav:LiteAVSDK_TRTC:latest.release'
}
```

2. defaultConfigでAppが使用するCPUアーキテクチャを指定します。

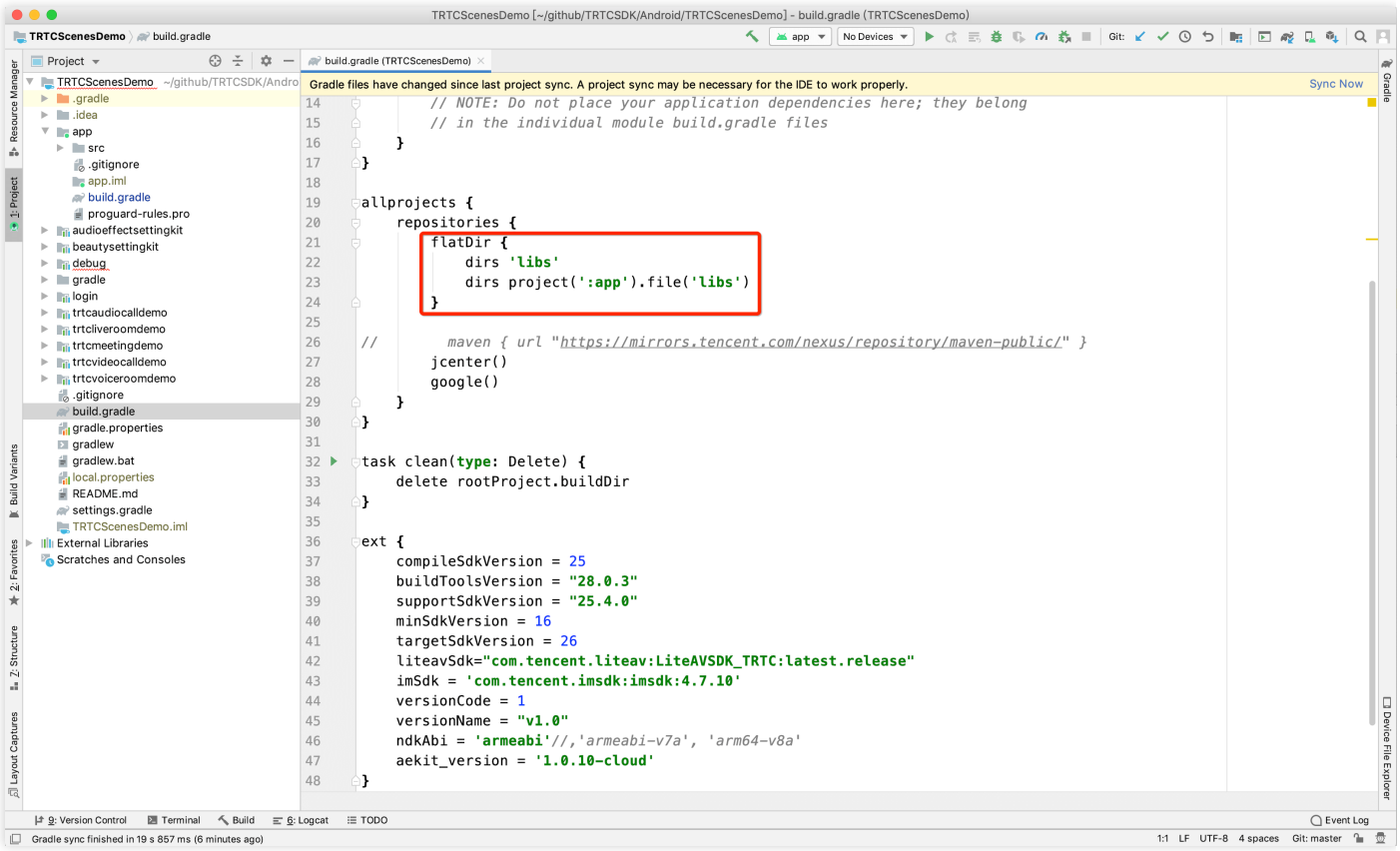
```
defaultConfig {
ndk {
abiFilters "armeabi-v7a", "arm64-v8a"
}
}
```

説明：
現在 TRTC SDKは、armeabi-v7a、arm64-v8aをサポートしています。

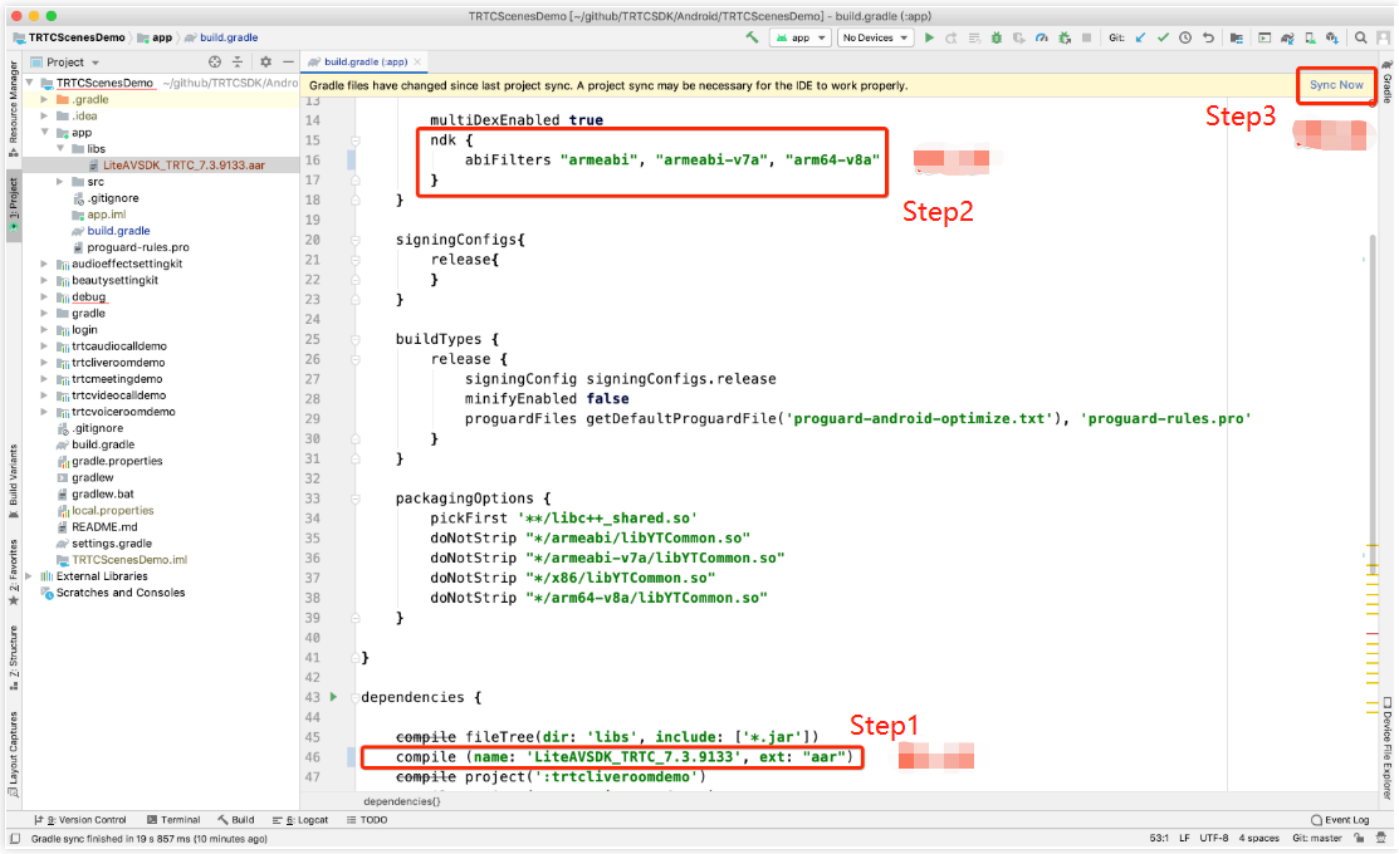
3. **Sync Now**をクリックして、SDKを自動的にダウンロードし、プロジェクトに統合します。

方法2：SDKのダウンロードと手動インポート

1. SDKをダウンロードし、ローカルに解凍して開きます。
2. 解凍したaarファイルをプロジェクトの**app/libs**ディレクトリの下にコピーします。
3. プログラムのルートディレクトリの下の build.gradle の中に、**flatDir**を追加し、ローカルライブラリのパスを指定します。



4. app/build.gradleの中に、aar パッケージのコードを追加して引用します。



5. app/build.gradle的defaultConfigの中で、Appが使用するCPUアーキテクチャを指定します。

```
defaultConfig {
ndk {
abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
}
}
```

説明：
現在 TRTC SDKは、armeabi、armeabi-v7a、arm64-v8aをサポートしています。

6. **Sync Now**をクリックして、TRTC SDKの統合を完了します。

手順2：App権限の構成

AndroidManifest.xml の中で Appの権限を設定します。TRTC SDKには次の権限が必要となります：

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

注意：
`android:hardwareAccelerated="false"` は設定しないでください。ハードウェアアクセラレーションを無効にすると、相手側のビデオストリームがレンダリングできなくなります。

手順3：混同規則の設定

proguard-rules.pro ファイルで、 TRTC SDK関連のクラスを非混同リストに追加します：

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

C++ インターフェースを介してSDKを使用（オプション）

開発にJavaではなくC++ インターフェースを使用したい場合は、この手順を実行できます。Java言語のみを使用してTRTC SDKを呼び出す場合は、この手順を無視してください。
1.まず、上記の指示に従ってjarおよび soライブラリをインポートする方式で TRTC SDKを統合します。
2.ヘッダーファイルをコピー：SDK中のC++ヘッダーファイルをプロジェクトにコピーし（パス： `SDK/LiteAVSDK_TRTC_xxx/libs/include` ）、CMakeLists.txtでincludeフォルダーパスと soライブラリの動的リンクを設定します。

```
cmake_minimum_required(VERSION 3.6)
# C++ インターフェースヘッダーファイルパスの設定
include_directories(
${CMAKE_CURRENT_SOURCE_DIR}/include # SDK/LiteAVSDK_TRTC_xxx/libs/includeからコピーする
)
```

```
add_library(
native-lib
SHARED
native-lib.cpp)
# libliteavsdk.so動的ライブラリパスの設定
add_library(libliteavsdk SHARED IMPORTED)
set_target_properties(libliteavsdk PROPERTIES IMPORTED_LOCATION ${CMAKE_CURRENT_S
OURCE_DIR}/../../../libs/${ANDROID_ABI}/libliteavsdk.so)
find_library(
log-lib
log)
# libliteavsdk.so 動的リンクの設定
target_link_libraries(
native-lib
libliteavsdk
${log-lib})
```

3. ネームスペースの利用：C++ のすべてのプラットフォームのインターフェースのメソッド、タイプなどはいずれも trtc ネームスペースで定義されています。コードをより簡潔にするため、trtc ネームスペースを直接使用することをお勧めします

```
using namespace trtc;
```

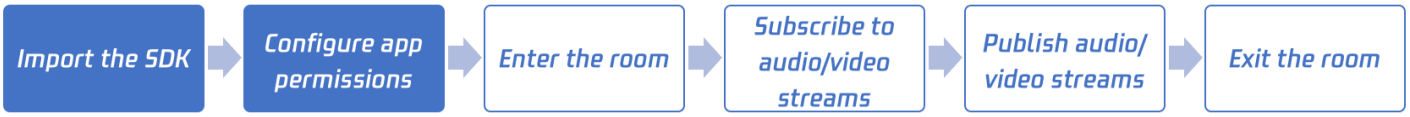
説明：

- Android Studio C/C++ 開発環境の設定方法は、Android Studioの公式ドキュメントをご参照ください：プロジェクトへのC/C++コードの追加 。
- 現在、TRTCバージョンのSDKのみがC++インターフェースをサポートしています。C++ インターフェースの使用方法については、 全プラットフォーム（C++）APIの概要をご参照ください。

Mac

最終更新日：：2022-07-07 15:10:00

このドキュメントでは、主にTRTC SDK (Mac)を迅速にプロジェクトに統合する方法を紹介します。以下のステップにしたがって設定するだけで、SDK統合のタスクが完了します。



開発環境要件

- Xcode 9.0+。
- OS X10.10+ のMac 実機。
- プロジェクトに有効な開発者の署名が設定してあること。

手順1：SDKのインポート

CocoaPodsを使用して自動でローディングするか、または手動で、先ず SDKをダウンロードして、それを現在のプログラムのプロジェクトにインポートする方式を選択できます。

スキーム1：CocoaPods

1. **CocoaPodsのインストール**
   端末のウィンドウに次のコマンドを入力します（事前にMac にRuby環境をインストールしておく必要があります）：

```
sudo gem install cocoapods
```

2. **Podfileファイルの作成**
   プロジェクトが存在するパスに入り、次のコマンドラインを入力するとプロジェクトパスの下にPodfile ファイルが現れます。

```
pod init
```

3. **Podfileファイルの編集**
   Podfile ファイルを編集します。次の2種類の設定方法があります：

- **方式1**：Tencent Cloud LiteAV SDKのpodパスを使用します。

```
platform :osx, '10.10'
target 'Your Target' do
pod 'TXLiteAVSDK_TRTC_Mac', :podspec => 'https://liteav.sdk.qcloud.com/pod/liteavsdkspec/TXLiteAVSDK_TRTC_Mac.podspec'
end
```

- **方式2**：CocoaPodの公式ソースを使用します。バージョンナンバーの選択をサポートしています。

```
platform :osx, '10.10'
source 'https://github.com/CocoaPods/Specs.git'
target 'Your Target' do
pod 'TXLiteAVSDK_TRTC_Mac'
end
```

4. **SDKのインストールおよび更新**

- 端末のウィンドウに次のコマンドを入力して、TRTC SDKのインストールを実行します。

```
pod install
```

- または次のコマンドを使用してローカルライブラリのバージョンを更新します。

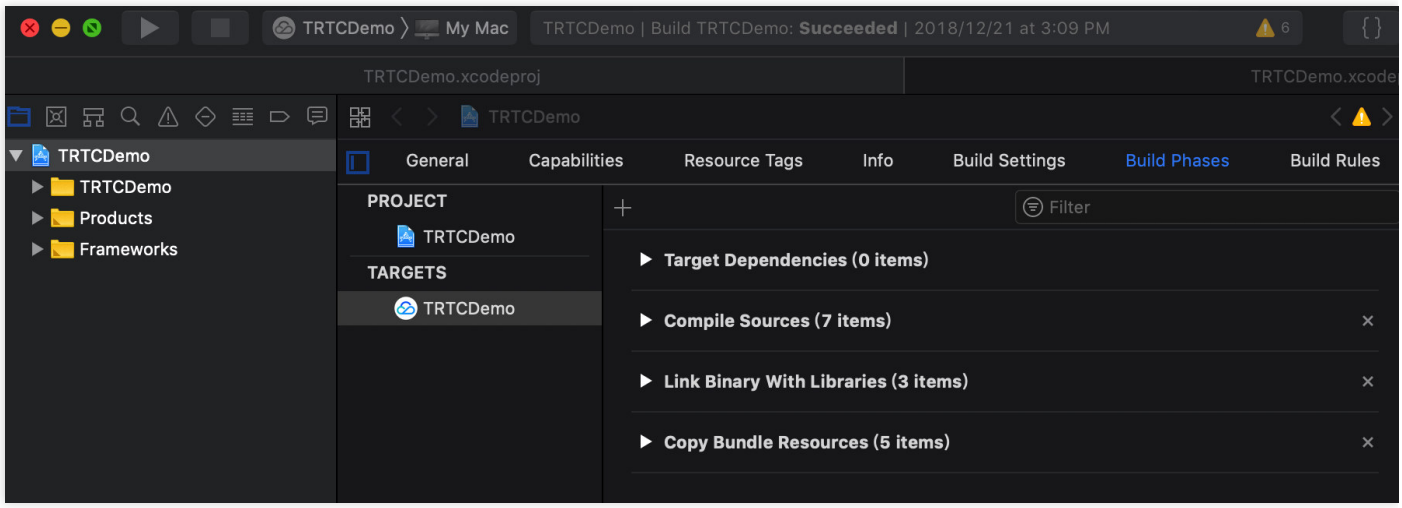
```
pod update
```

podコマンドの実行が完了すると、SDKを統合した `.xcworkspace` という拡張子のプログラムファイルが生成されますので、これをダブルクリックして開きます。

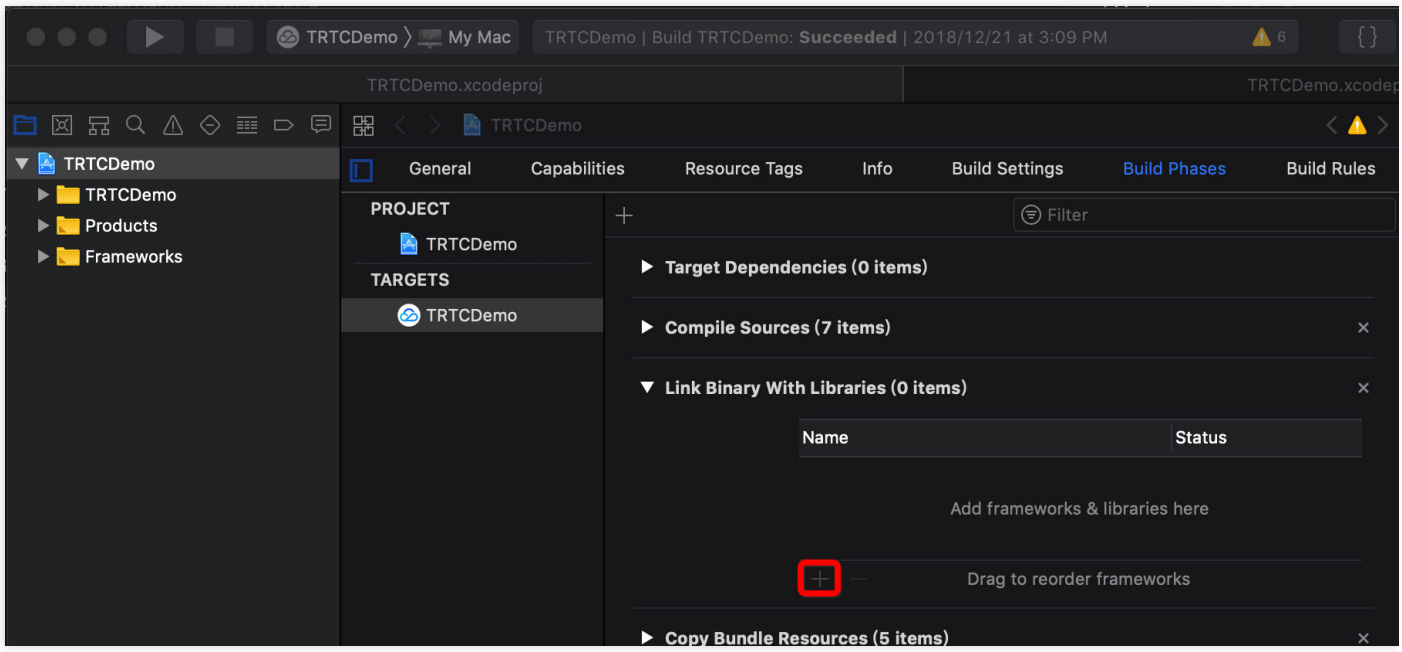
スキーム2：手動統合

1. TRTC-SDK のMacバージョンをダウンロードします。
2. お客様のXcodeのプロジェクトを開き、第1ステップでダウンロードした framework をプログラムにインポートします。

3. 動作させたいtargetを選択し、Build Phasesの項目を選択します。

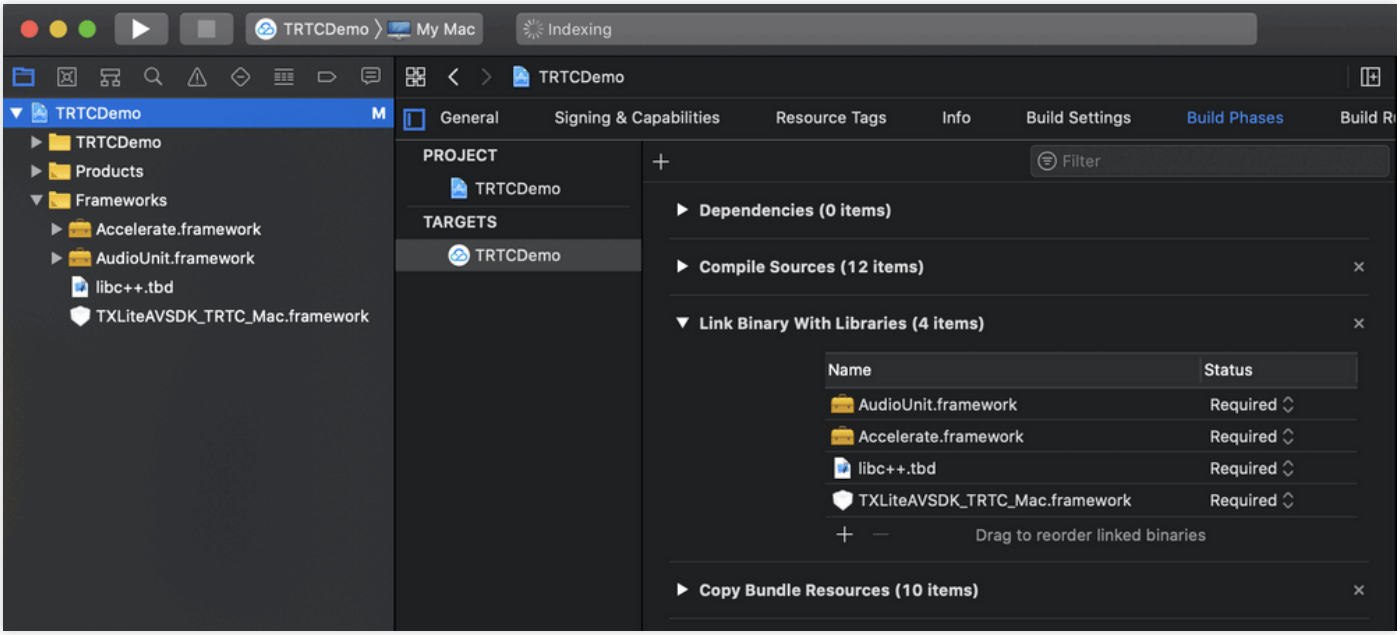


4. **Link Binary with Libraries**の項目をクリックして展開し、一番下の「+」アイコンをクリックして依存ライブラリを追加します。



5. ダウンロードしたSDK Frameworkおよび必要な依存ライブラリ `TXFFmpeg.xcframework` 、 `TXSoundTouch.xcframework` および `libc++.tbd` 、 `Accelerate.framework` 、 `SystemConfiguration.framework` 、 `MetalKit.framework` を順に追加します。

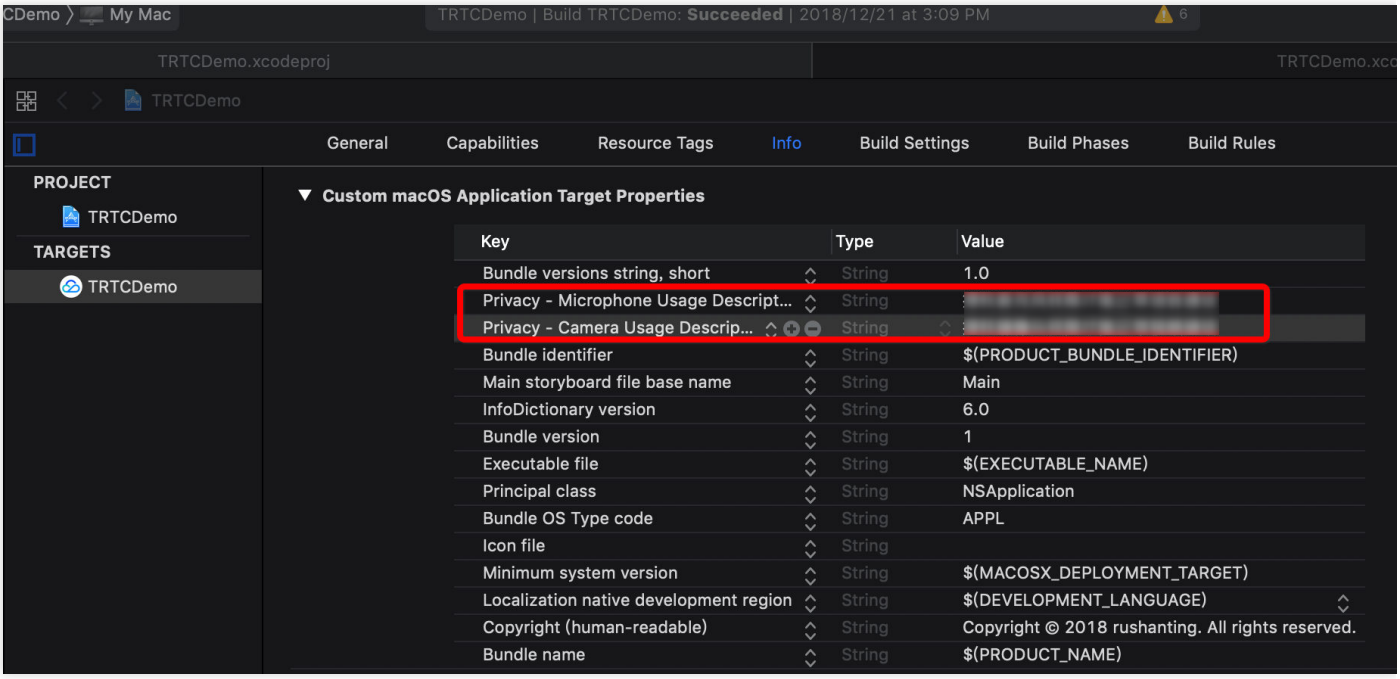
追加した後は次の図のようになります：



手順2：App権限の構成

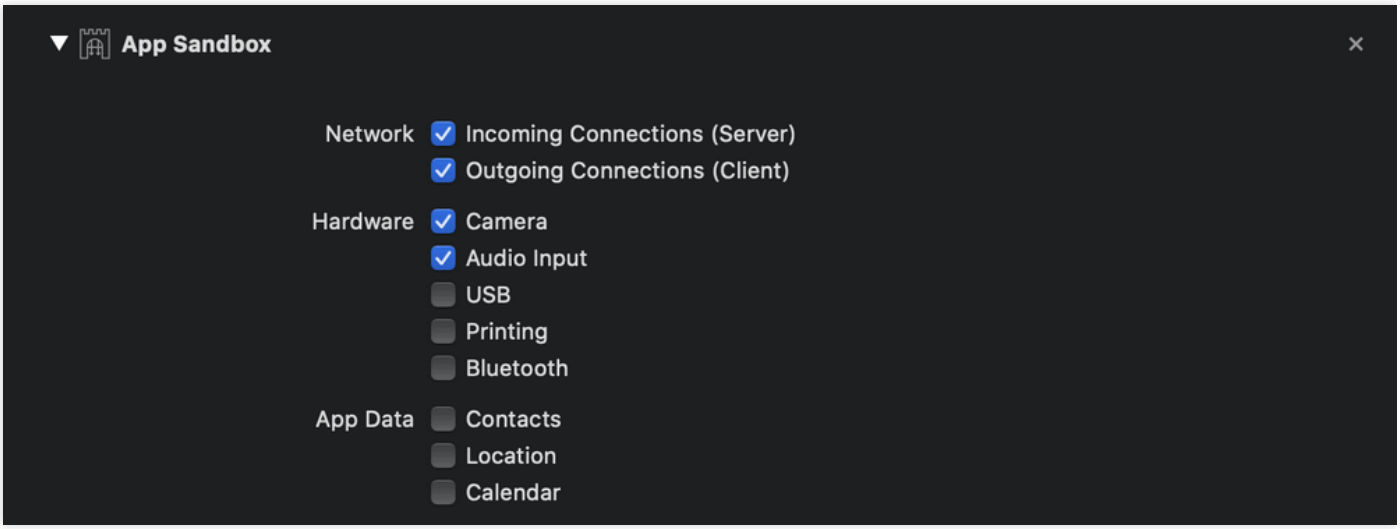
SDK の音声ビデオ機能を使用するには、マイクとカメラの使用権限を許可する必要がありますので、Appの Info.plistの中に次の2項目を追加します。それぞれマイクとカメラに対応し、システムが使用許可のダイヤログボックスをポップアップするときに表示される情報となります。

- **Privacy - Microphone Usage Description**、さらにマイク使用目的のプロンプトを記入します。
- **Privacy - Camera Usage Description**、さらにカメラ使用目的のプロンプトを記入します。
  下図に示すように：



Appで **App Sandbox**または**Hardened Runtime**を有効にしている場合は、 `Network` 、 `Camera` 、 `Audio Input` の選択項目にチェックを入れてください。

- App Sandbox の設定は次の図のとおりです：



- Hardened Runtime の設定は次の図のとおりです：

手順3：プロジェクトへのSDKのインポート

手順1のインポートと手順2のデバイス権限の承認を完了すると、プロジェクトへSDKにより提供されているAPIを参照できます。

Objective-CまたはSwiftインターフェースによる TRTC SDKの参照

Objective-CまたはSwiftコードの中でSDKを使用する方式は2種類あります：

- **コンポーネントの引用**：プロジェクトのSDK APIを使用したいファイルの中に、コンポーネントを追加して引用します。

```
@import TXLiteAVSDK_TRTC_Mac;
```

- **ヘッダーファイルの引用**：プロジェクトのSDK APIを使用したいファイルの中に、具体的なヘッダーファイルをインポートします。

```
#import TXLiteAVSDK_TRTC_Mac/TRTCCloud.h
```

C++インターフェースによる TRTC SDKの参照（オプション）

1. **ヘッダーファイルの引用**：C++インターフェースを使用してMacアプリケーションを開発したい場合は、`TXLiteAVSDK_TRTC_Mac.framework/Headers/cpp_interface` ディレクトリの下のヘッダーファイルをインポートしてください。

```
#include TXLiteAVSDK_TRTC_Mac/cpp_interface/ITRTCCloud.h
```

2. **ネームスペースの利用**：C++の全プラットフォームのインターフェースのメソッド、タイプなどはいずれもtrtcネームスペースの中に定義されています。コードをより簡潔にするため、trtcネームスペースを直接使用することをお勧めします。

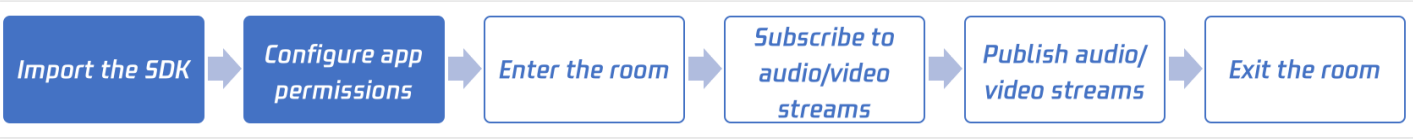
```
using namespace trtc;
```

説明：
C++ インターフェースの使用方法については、 全プラットフォーム（C++）APIの概要をご参照ください。

Windows C++

最終更新日：：2022-07-07 15:10:00

このドキュメントでは、Tencent Cloud TRTC SDK（Windows C++ 版）をMFCプロジェクトに素早く統合する方法を紹介します。



開発環境要件

- OS：Windows 7以上のバージョン。
- 開発環境：Visual Studio 2010およびそれ以上のバージョン、Visual Studio 2015の使用を推奨します。

MFCプロジェクトによるC++ SDKの統合

このセグメントでは、簡単なMFC プロジェクトを作成し、Visual Studio プログラムの中で C++ SDKを統合する方法を紹介します。

手順1：SDKのダウンロード

SDKダウンロードし、解凍して開きます。この例では、SDKディレクトリ下のC++バージョンのSDKファイルを参照するだけで済みます。64ビットを例にとると、SDKの場所は `./SDK/CPlusPlus/Win64/` です。これには主に次の部分が含まれます：

ディレクトリ名	説明
include	詳細なインターフェース注釈つきのAPIヘッダファイル
lib	コンパイル用の.libファイルと実行時にローディングされる.dllファイル

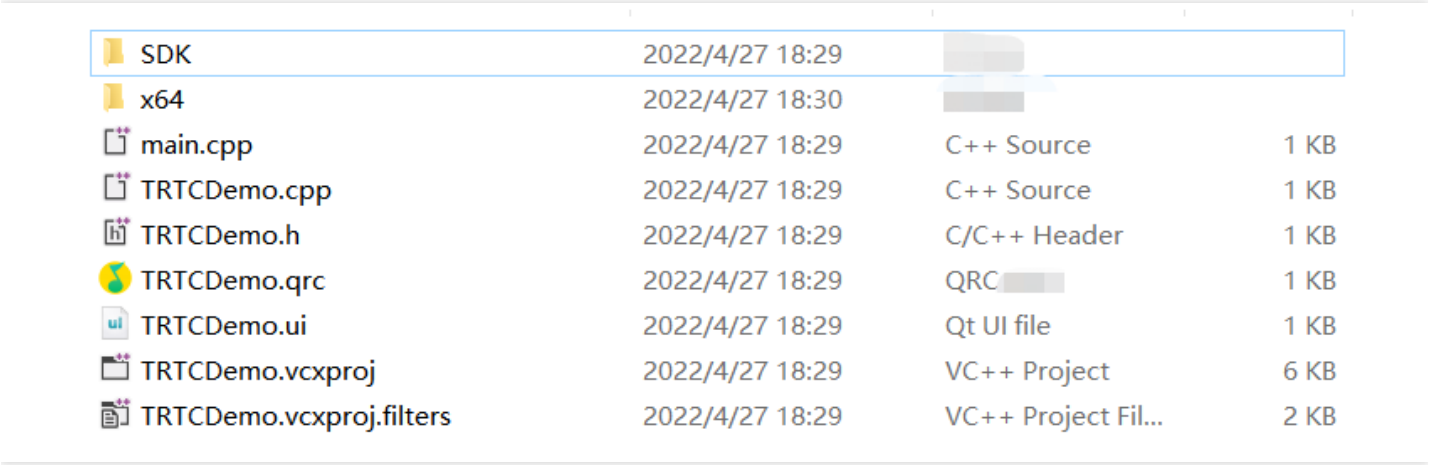
手順2：プロジェクトの新規作成

Visual Studioを開き、TRTCDemoという名前のMFC アプリケーションプログラムを作成します。
迅速に統合する方法を紹介しやすいように、ガイドの**アプリケーションプログラムのタイプ**の画面では、比較的簡単な**ダイアログベース**のタイプを選択しています。
その他のガイドの設定は、デフォルトの設定を選択してください。

手順3：**ファイルのコピー

解凍後のSDKフォルダを `TRTCDemo.vcxproj` が存在するディレクトリ下にコピーします。下図のとおりです：

> 説明：
> 現在はC++ SDKがあれば、SDKパス下のCsharpディレクトリを削除できます。



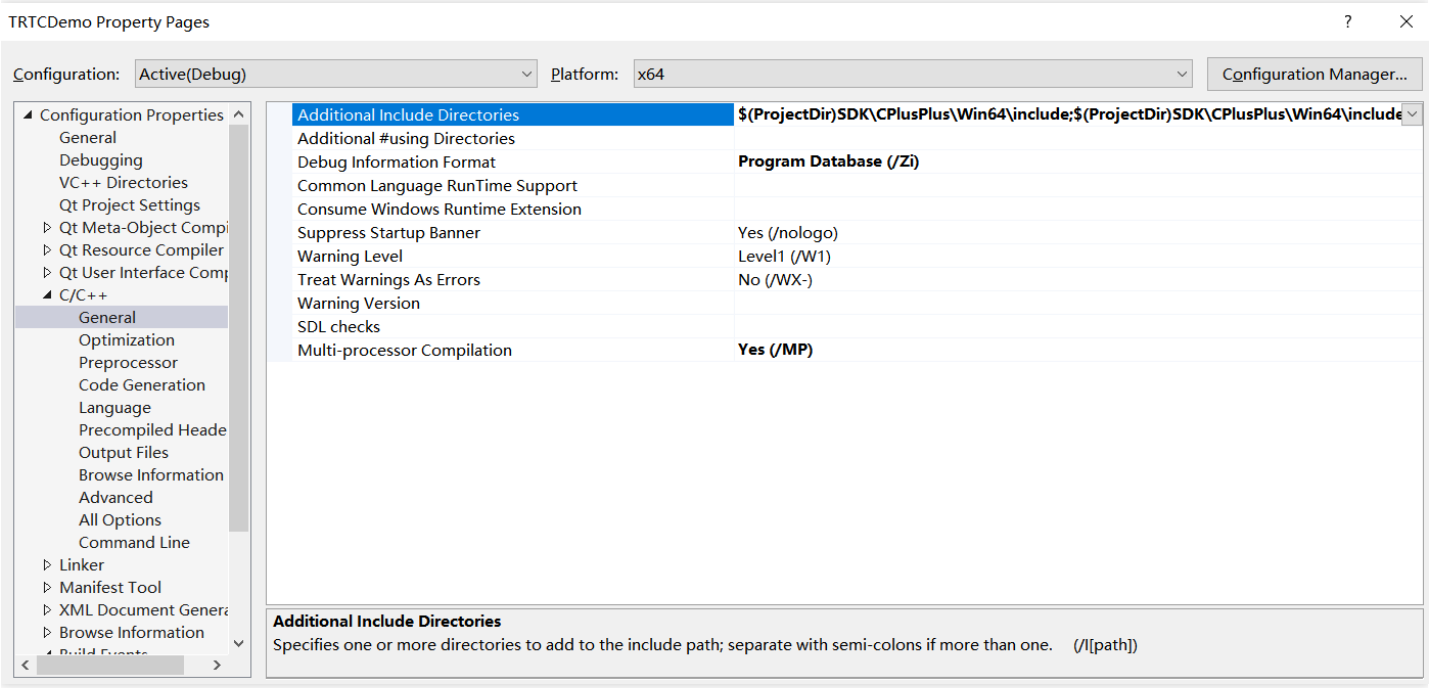
手順4：プロジェクト構成の変更

TRTCDemoの属性ページを開きます。**ソリューションのResource Manager >TRTCDemoプロジェクトのメニュー>属性**を右クリックして、次の手順にしたがって設定してください：

1. **Includeのディレクトリの追加：**

C/C++ > 常規 > 添付ファイル付きディレクトリから、以下のSDKヘッダファイルディレクトリを追加します。なお、64bitの例： `$(ProjectDir)SDK\CPlusPlus\Win64\include` 和 `$(ProjectDir)SDK\CPlusPlus\Win64\include\TRTC` 、下図のとおりです：

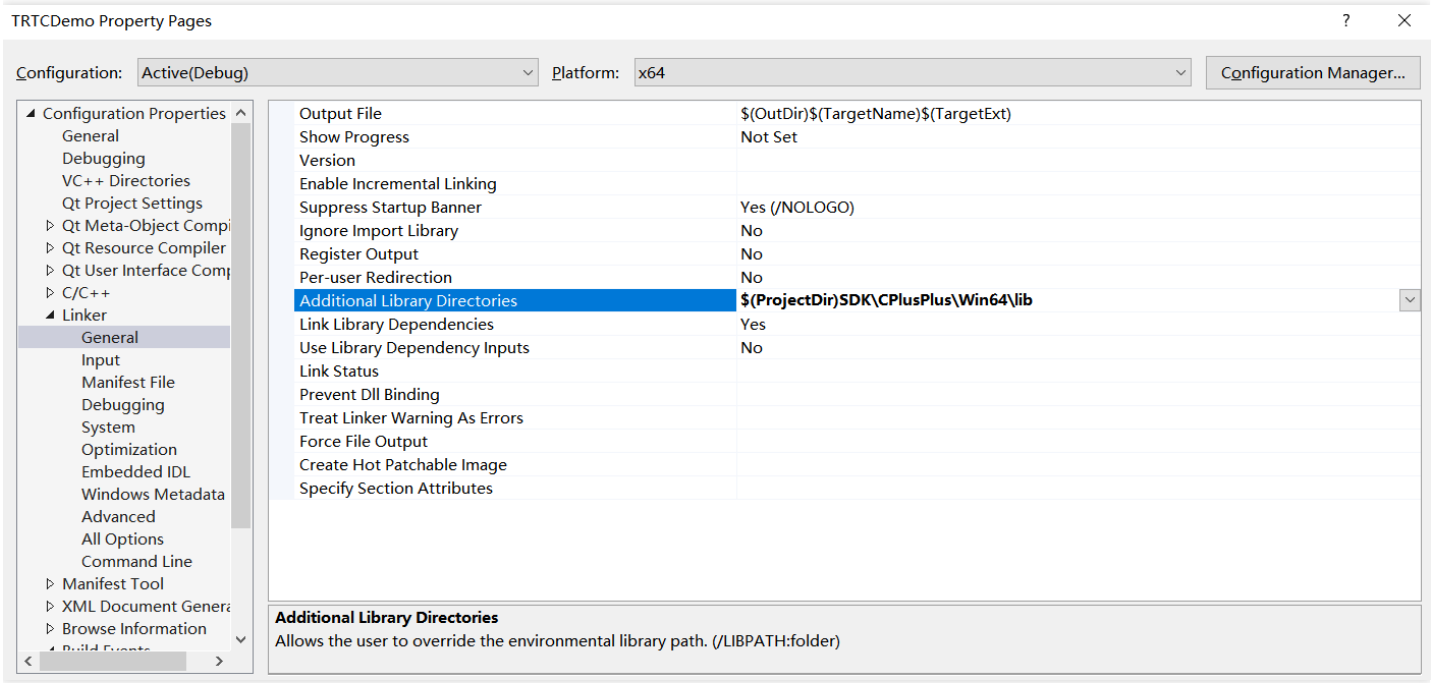
> 説明：
> 32bitの場合、SDKヘッダファイルディレクトリを `$(ProjectDir)SDK\CPlusPlus\Win32\include` と `$(ProjectDir)SDK\CPlusPlus\Win32\include\TRTC` として設定してください。



2. **ライブラリのディレクトリの追加：**

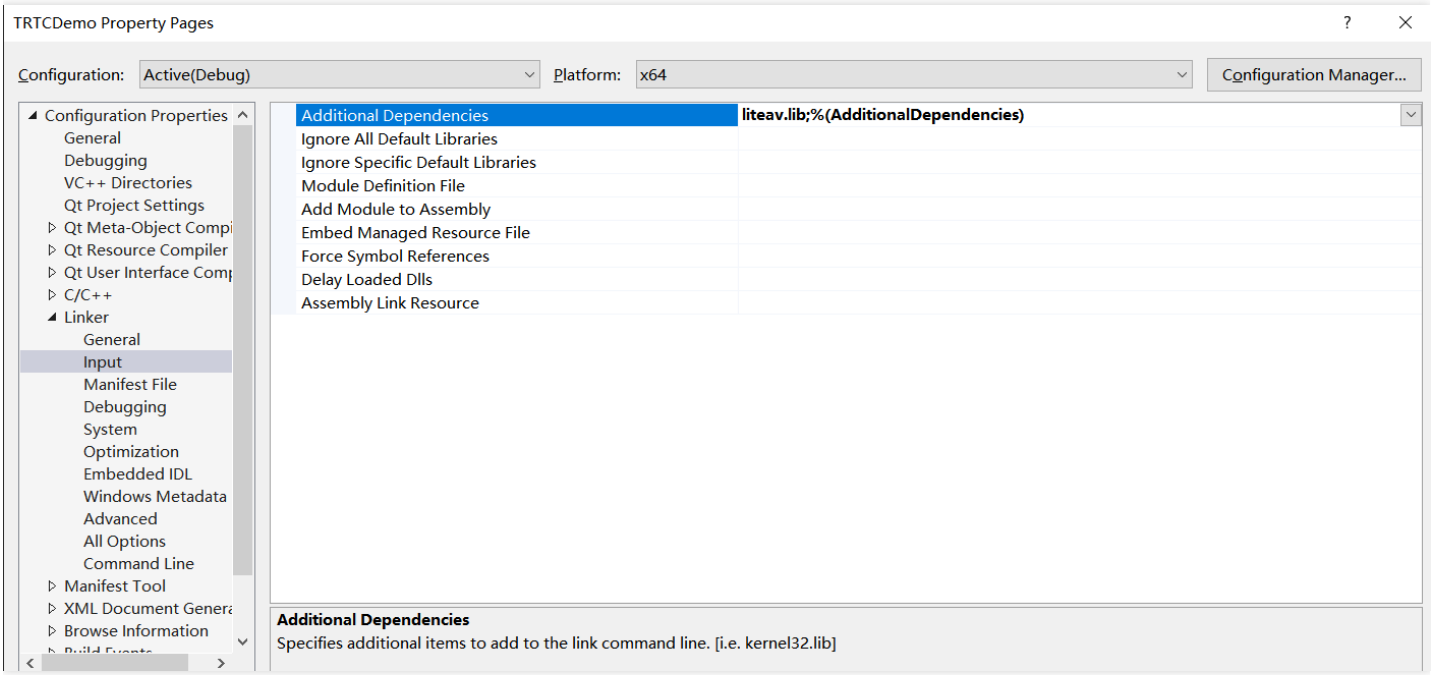
リンカ > 常規 > 添付ライブラリ付きディレクトリから、以下のSDKヘッダファイルディレクトリを追加します。なお、64bitの例： `$(ProjectDir)SDK\CPlusPlus\Win64\lib` 、下図のとおりです：

> 説明：
> 32bitの場合、SDKライブラリディレクトリを `$(ProjectDir)SDK\CPlusPlus\Win32\lib` として設定してください。



3. **ライブラリファイルの追加：**

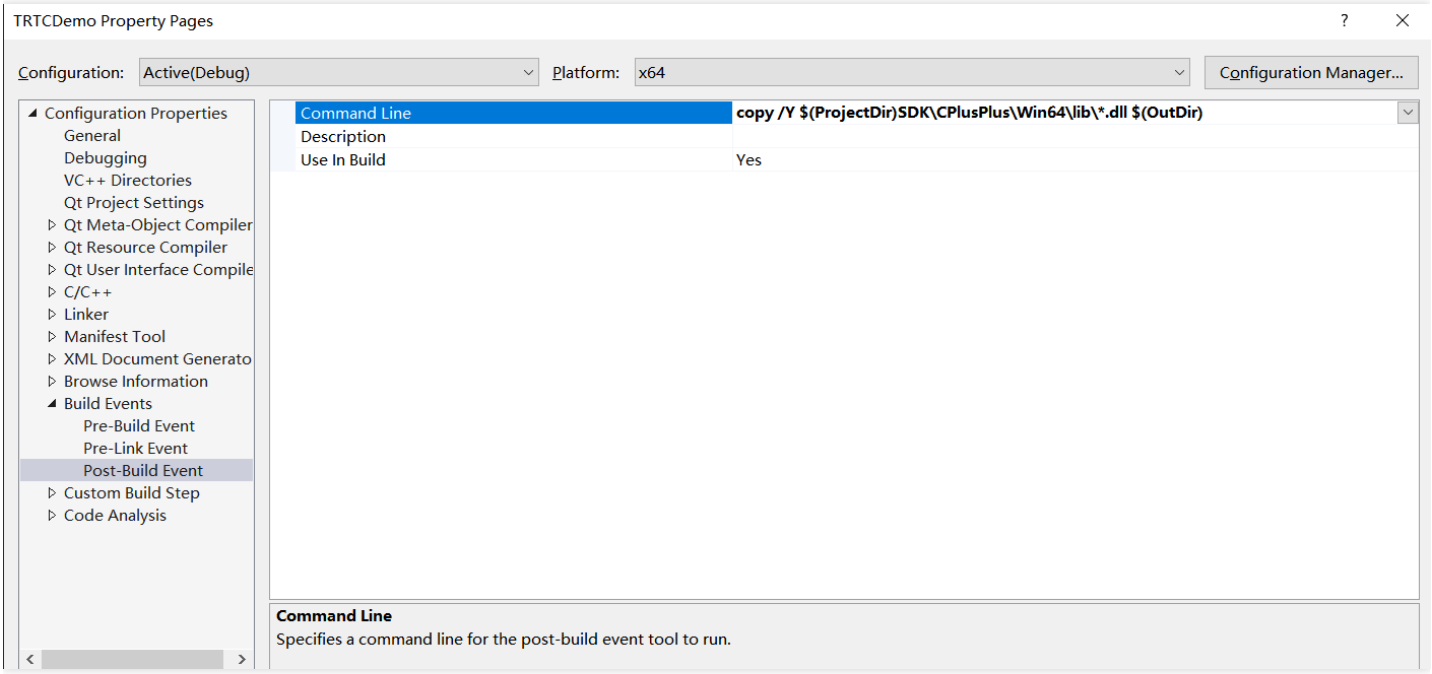
リンカ > 入力 > 依存項目の追加からSDKライブラリファイル `liteav.lib` を追加します。下図のとおりです：



4. **copy コマンドの追加：**

イベントの生成 > 後続イベントの生成 > コマンドラインから、コピーコマンド `copy /Y $(ProjectDir)SDK\CPlusPlus\Win64\lib\*.dll $(OutDir)` を追加します。コンパイルが完了すると、自動的にSDKの.dllファイルがプログラムの実行ディレクトリの下にコピーされます。下図のとおりです。

説明：
32bitの場合、コピーコマンド `copy /Y $(ProjectDir)SDK\CPlusPlus\Win32\lib\*.dll $(OutDir)` を追加します。



手順5： SDKバージョン番号のプリント

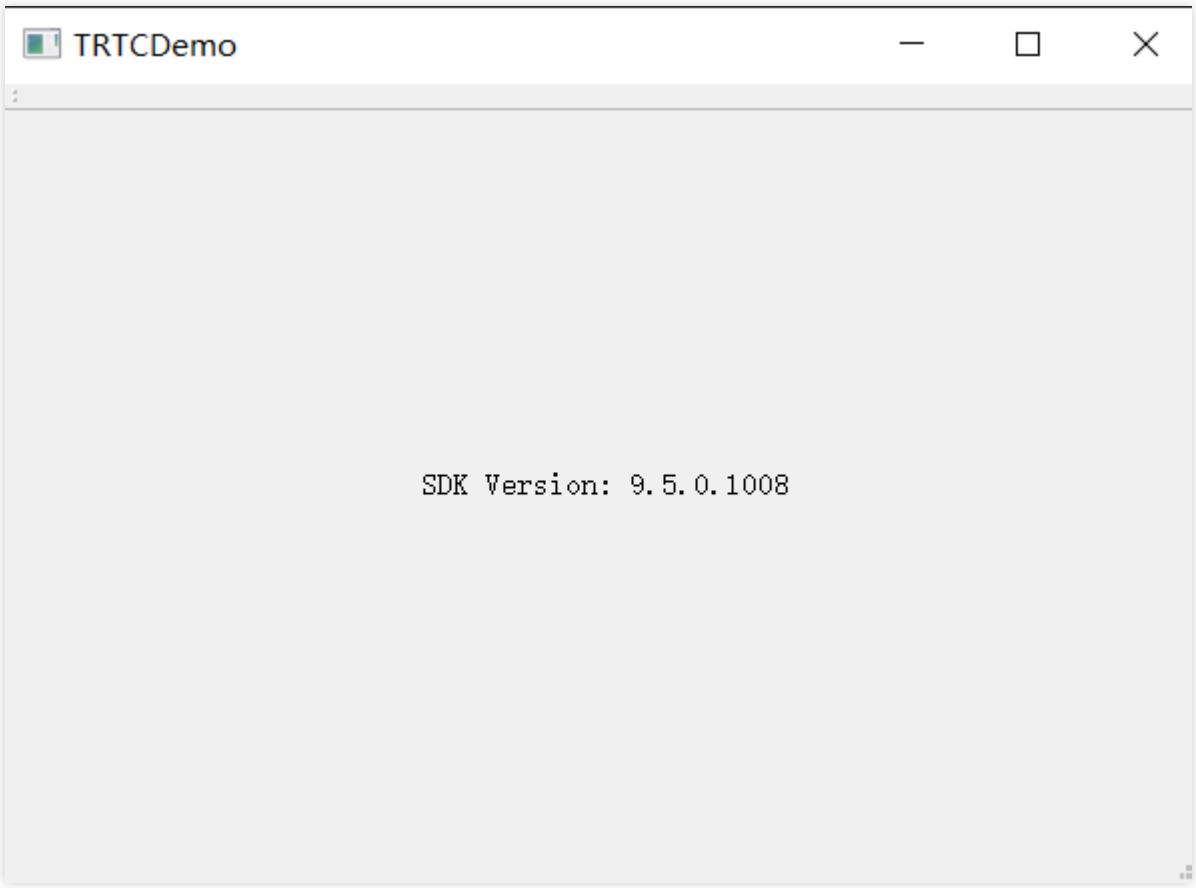
1. TRTCDemoDlg.cppファイルのヘッドにヘッダファイルを追加してください。コードは以下のとおりです：

```
#include "ITRTCCloud.h"
```

2. `CTRTCDemoDlg::OnInitDialog` 関数の中に、以下のテストコードを追加します：

```
ITRTCCloud * pTRTCCloud = getTRTCShareInstance();
CString szText;
szText.Format(L"SDK version: %hs", pTRTCCloud->getSDKVersion());
CWnd *pStatic = GetDlgItem(IDC_STATIC);
pStatic->SetWindowTextW(szText);
```

3. F5キーを押して実行すると、SDKのバージョン番号がプリントされます。下図のとおりです：



よくあるご質問

- 次のエラーが生じた場合は、前述のプログラム設定にしたがって、 SDKヘッダーファイルのディレクトリが正しく追加されているかチェックしてください。

```
fatal error C1083: include ファイルを開くことができません: “TRTCCloud.h”: No such file or directory
```

- 次のエラーが生じた場合は、前述のプログラム設定にしたがって、SDKライブラリのディレクトリとライブラリファイルが正しく追加されているかチェックしてください。

```
error LNK2019: 解析できない外部シンボル "__declspec(dllimport) public: static class TXString __cdecl TRTCCloud::getSDKVersion(void)" (__imp_?getSDKVersion@TRTCCloud@@SA?AVTXString@@XZ)、この記号が関数 "protected: virtual int __thiscall CTRTCDemoDlg::OnInitDialog(void)" (?OnInitDialog@CTRTCDemoDlg@@MAEHXZ) の中に引用されています
```

Web

最終更新日：：2022-11-09 11:17:37

ここでは、主にTencent Cloud TRTC Web SDK をプロジェクトに素早く統合する方法を紹介します。

サポートするプラットフォーム

WebRTCのテクノロジーはGoogleが初めて提唱し、Chrome、Edge、Firefox、Safari、Operaなどのブラウザですべてサポートされています。Tencent Cloud TRTC Web SDKはWebRTCパッケージをベースにしたものです。Tencent Cloud TRTC Web SDKのサポートの詳細については、サポートするプラットフォームをご参照ください。

- ユーザーのユースケースが主に教育シーンである場合は、教師用端末では安定性がより優れたElectron ソリューションの使用をお勧めし、大小2チャネル画面、よりフレキシブルなスクリーンシェアリング方法およびより強力な弱ネットワークリカバリー能力をサポートしています。

Tencent Cloud TRTC Web SDKの詳細なサポートレベルの表については、サポートしているプラットフォームをご参照ください。

注意：

- ブラウザでTRTC Web SDK機能テスト画面を開けば、現在のブラウザがWebRTCのすべての機能をサポートしているかどうかチェックすることができます。例:WebViewなどのブラウザ環境。
- H.264 の著作権上の制限により、Huawei Chrome 88 より前のバージョンは H264 エンコーディングを使用できません（つまり、ストリームをプッシュできません）。HuaweiデバイスのChromeブラウザで TRTC Web SDK を使用してストリームをプッシュするには、チケットを提出 し、VP8コーデックの有効化を申請してください。

URLドメイン名プロトコルの制限

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	サポートあり	サポートあり	サポートあり	推奨

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	HTTPプロトコル	サポートあり	サポートなし	サポートなし	
ローカル開発環境	http://localhost	サポートあり	サポートあり	サポートあり	推奨
ローカル開発環境	http://127.0.0.1	サポートあり	サポートあり	サポートあり	
ローカル開発環境	http://[ローカルマシンIP]	サポートあり	サポートなし	サポートなし	
ローカル開発環境	file:///	サポートあり	サポートあり	サポートあり	

ファイアウォールの制限

TRTC Web SDKを使用する時、ユーザはファイアーフォールの制限でオーディオビデオ通話を利用できない可能性がありますので、ファイアウォール制限への対応 を参照し、関連するポートと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。

TRTC Web SDKの統合

NPM統合

1. プロジェクトにSDKパッケージをインストールするには、npmを使用する必要があります。

```
npm install trtc-js-sdk --save
```

2. プロジェクトのスクリプトでモジュールを導入します。

```
import TRTC from 'trtc-js-sdk';
```

Script統合

下記のコードをWebページに追加するだけで完了です。

```
<script src="trtc.js"></script>
```

関連リソース

SDKダウンロードアドレス：クリックしてダウンロード 。

初期化フローおよびAPIの使用法の詳細については、以下のガイドをご参照ください。

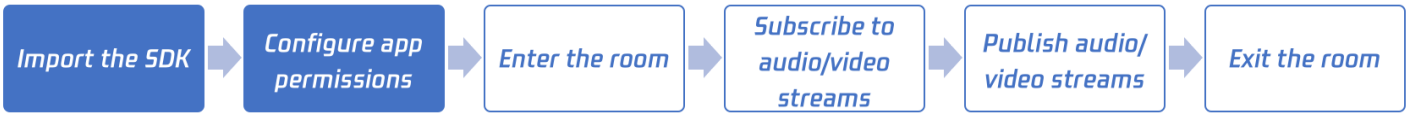
機能	Sample Codeガイド
基本的なオーディオビデオ通話	ガイドリンク
インタラクティブライブストリーミング	ガイドリンク
カメラおよびマイクの切り替え	ガイドリンク
ローカルビデオのプロパティの設定	ガイドリンク
ローカルオーディオまたはビデオの動的な停止と開始	ガイドリンク
画面共有	ガイドリンク
音量計測	ガイドリンク
ユーザー定義キャプチャとカスタマイズ再生レンダリング	ガイドリンク
ルーム内アップリンクユーザー数の制限	ガイドリンク
バックグラウンドミュージックと効果音の実装ソリューション	ガイドリンク
通話前の環境およびデバイステスト	ガイドリンク
通話前のネットワーク品質テスト	ガイドリンク
デバイス挿抜動作チェック	ガイドリンク
CDNへのプッシュの実現	-
ビッグスモールストリームの伝送を有効にする	ガイドリンク
美顔を有効にする	ガイドリンク
ウォーターマークを有効にする	ガイドリンク
ルーム間マイク接続の実現	ガイドリンク

説明：
その他の機能についてはクリックして確認してください。

Electron

最終更新日：：2022-08-08 15:25:06

ここでは、主にTencent Cloud TRTC Electron SDKをプロジェクトに素早く統合する方法を紹介します。以下の手順に従って設定すれば、 SDK の統合作業を完了できます。



サポートするプラットフォーム

- Windows(PC)
- Mac

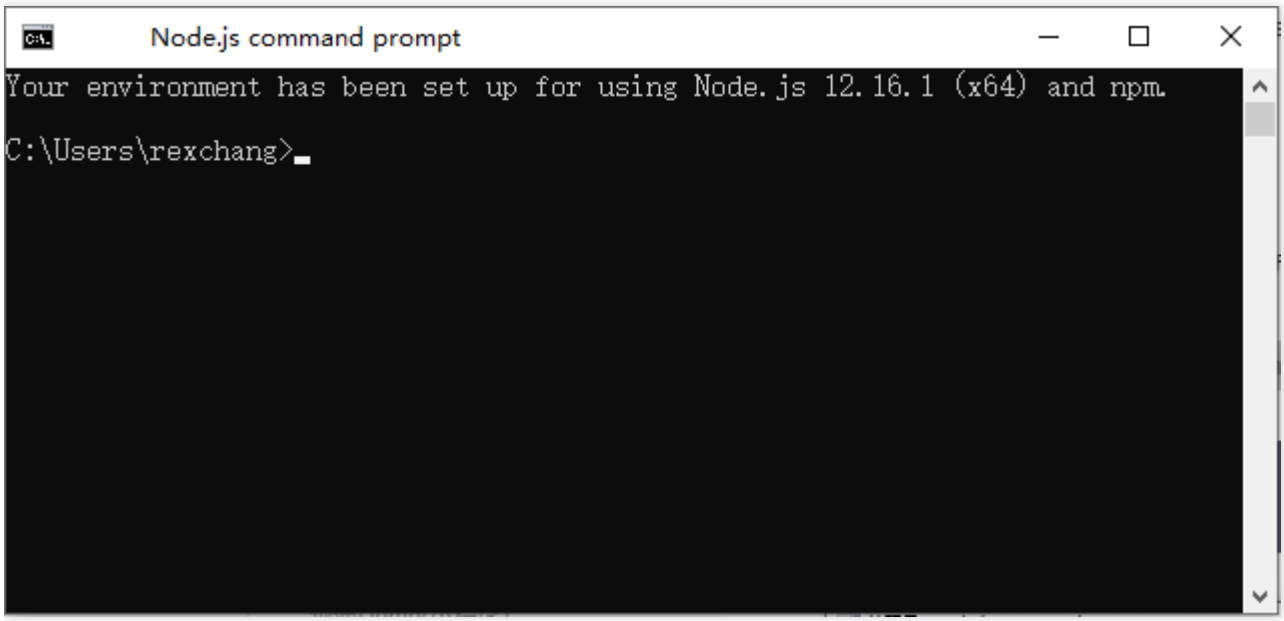
SDKのインポート

ステップ1. Node.jsのインストール

- Windowsプラットフォームのインストールガイド
- Mac OSプラットフォームのインストールガイド

1. WindowsOSに従って、最新バージョンのNode.js インストールパッケージ `Windows Installer (.msi) 64-bit` を選択、ダウンロードします。
2. アプリケーションプログラムリストにあるNode.js command promptを開き、コマンドラインのポートを起動

し、その後の手順における各コマンドの入力に使用します。



ステップ2： Electronのインストール

コマンドラインポートで以下のコマンドを実行し、Electronをインストールします。バージョンは>=4.0.0を推奨します。

```
$ npm install electron@latest --save-dev
```

ステップ3： Electronバージョン TRTC SDKのインストール

1. Electron項目でnpmコマンドを使用してSDKパッケージをインストールします：

```
$ npm install trtc-electron-sdk@latest --save
```

説明：
TRTC Electron SDK最新版では、 trtc-electron-sdk でクエリーできます。
2. プロジェクトのスクリプトでモジュールを導入して、使用します。

```
const TRTCCloud = require('trtc-electron-sdk').default;
// import TRTCCloud from 'trtc-electron-sdk';
this.rtcCloud = new TRTCCloud();
// SDKバージョン番号の取得
this.rtcCloud.getSDKVersion();
```

v7.9.348から、TRTC Electron SDKはtrtc.d.tsファイルを追加しており、 TypeScriptを使用する開発者の操作性が向上しました。

```
import TRTCCloud from 'trtc-electron-sdk';
const rtcCloud: TRTCCloud = new TRTCCloud();
// SDKバージョン番号の取得
rtcCloud.getSDKVersion();
```

ステップ4：パッケージで実行可能なプログラム

パッケージツールのインストール：パッケージツール `electron-builder` を使用してパッケージングすることを推奨します。以下のコマンドを実行して `electron-builder` をインストールできます。

```
$ npm install electron-builder@latest --save-dev
```

Electronバージョンの TRTC SDK（すなわち `trtc_electron_sdk.node` ファイル）を正しくパッケージするために、次のコマンドを実行して `native-ext-loader` ツールもインストールする必要があります。

```
$ npm install native-ext-loader@latest --save-dev
```

ステップ5：パッケージ設定の変更（webpack.config.jsの例）

`webpack.config.js` は、項目アーキテクチャの設定情報を含んでいます。 `webpack.config.js` ファイルの位置は以下のとおりです。

- 通常、 `webpack.config.js` は項目のルートディレクトリにあります。
- `create-react-app` 使用して項目を新規作成する状況では、この設定ファイルは `node_modules/react-scripts/config/webpack.config.js` です。
- `vue-cli` を使用して項目を新規作成する状況では、webpack の設定は `vue.config.js` 設定の `configureWebpack` のプロパティにあります。
- プロジェクトファイルがカスタマイズされている場合は、ご自身でwebpack設定を検索してください。

1. 初めに `webpack.config.js` を構築するときは、 受信名を `--target_platform` のコマンドラインパラメータにできるため、コードの構築プロセスで異なる目標プラットフォームの特徴に応じて正しくパッケージできます。 `module.exports` の前に以下のコードを追加します。

```
const os = require('os');
const targetPlatform = (function(){
let target = os.platform();
for (let i=0; i<process.argv.length; i++) {
if (process.argv[i].includes('--target_platform=')) {
target = process.argv[i].replace('--target_platform=', '');
```

```
break;
}
}
if (!['win32', 'darwin'].includes) target = os.platform();
return target;
})();
```

注意：
`os.platform()` によって返される結果では、"darwin" は Macのプラットフォームを表します。64ビットまたは 32ビットに関わらず、"win32" は Windowsプラットフォームを表します。
2. その後 `rules` オプションでは以下の設定を追加します。 `targetPlatform` 変数は `rewritePath` を使って、異なる目標プラットフォームに従って違う設定に切り替えることができます。

```
rules: [
{
test: /\.node$/,
loader: 'native-ext-loader',
options: {
rewritePath: targetPlatform === 'win32' ? './resources' : '../Resources'
// 開発環境について
// rewritePath: './node_modules/trtc-electron-sdk/build/Release'
}
},
]
```

この設定の意味は以下のとおりです。

- Windowsの `.exe` ファイルをパッケージにするときは、 `native-ext-loader` に `[アプリケーションプログラムルートディレクトリ]/resources` ディレクトリでTRTC SDKをアップロードします。
- Macの `.dmg` をパッケージするときは、 `native-ext-loader` に `[アプリケーションディレクトリ]/Contents/Frameworsk/../Resources` ディレクトリでTRTC SDKをロードします。
- ローカル開発を実行する際は、 `native-ext-loader` に `./node_modules/trtc-electron-sdk/build/Release` ディレクトリ下でTRTC SDKをロードさせます。simple demoの設定をご参照ください。

`package.json` のスクリプト構築で、 `--target_platform` パラメータを追加するには、以下を実行します。

ステップ6：package.json設定の修正

`package.json` は項目のルートディレクトリにあり、そのうち項目のパッケージに必要な情報を含みます。しかし、デフォルトの状況では、 `package.json` の中のパスを修正しなければ順調にパッケージできません。以下の手順に従ってこのファイルを修正できます。

1. `main` 設定の修正。

```
// 多くの状況では、mainファイル名は任意に設定できます。例えば、 TRTCSimpleDemoのものは次のように設定できます。
"main": "main.electron.js",

// しかし、 create-react-app のスキャフォールディングを使用して新規作成した項目、mainファイルは次のように設定する必要があります：
"main": "public/electron.js",
```

2. 以下の `build` 設定をコピーし、 `package.json` ファイルに追加します。これは `electron-builder` が読み取る必要のある設定情報です。

```
"build": {
"appId": "[appId はご自身で定義してください]",
"directories": {
"output": "./bin"
},
"win": {
"extraFiles": [
{
"from": "node_modules/trtc-electron-sdk/build/Release/",
"to": "./resources",
"filter": ["**/*"]
}
]
},
"mac": {
"extraFiles": [
{
"from": "node_modules/trtc-electron-sdk/build/Release/trtc_electron_sdk.node",
"to": "./Resources"
}
]
}
},
```

3. `scripts` ノードにおいて以下のアーキテクチャおよびパッケージしたコマンドスクリプトを追加します。ここでは、 `create-react-app` および `vue-cli` 項目を例に、その他のツールで作成した項目もこの設定

を参考にすることができます。

```
// create-react-app項目にはこの設定を使用してください
"scripts": {
"build:mac": "react-scripts build --target_platform=darwin",
"build:win": "react-scripts build --target_platform=win32",
"compile:mac": "node_modules/.bin/electron-builder --mac",
"compile:win64": "node_modules/.bin/electron-builder --win --x64",
"pack:mac": "npm run build:mac && npm run compile:mac",
"pack:win64": "npm run build:win && npm run compile:win64"
}
// vue-cli項目にはこの設定を使用してください
"scripts": {
"build:mac": "vue-cli-service build --target_platform=darwin",
"build:win": "vue-cli-service build --target_platform=win32",
"compile:mac": "node_modules/.bin/electron-builder --mac",
"compile:win64": "node_modules/.bin/electron-builder --win --x64",
"pack:mac": "npm run build:mac && npm run compile:mac",
"pack:win64": "npm run build:win && npm run compile:win64"
}
```

パラメータ	説明
main	Electron のエントリーファイルは、一般には自由に設定できます。しかし、項目を `create-react-app` のスキャフォールディングを使用して作成した場合は、エントリーファイルは `public/electron.js` に設定する必要があります
build.win.extraFiles	Windowsプログラムをパッケージする場合は、`electron-builder` は `from` が示すディレクトリ下のすべてのファイルをbin/win-unpacked/resources（すべて小文字表記）にコピーします
build.mac.extraFiles	Macプログラムをパッケージする場合は、`electron-builder` は `from` が示す `trtc_electron_sdk.node` ファイルをbin/mac/your-app-name.app/Contents/Resources（頭文字は大文字表記）にコピーします
build.directories.output	パッケージファイルの出力パス。例えば、この設定を `bin` ディレクトリに出力する場合は、実際のニーズに従って修正します
build.scripts.build:mac	Macプラットフォームを目標にスクリプトを構築します
build.scripts.build:win	Windowsプラットフォームを目標にスクリプトを構築します
build.scripts.compile:mac	Macの .dmgインストールファイルをコンパイルします

パラメータ	説明
build.scripts.compile:win64	Windowsの .exeインストールファイルをコンパイルします
build.scripts.pack:mac	まず build:macアーキテクチャコードをコールしてからcompile:macをコールして .dmgインストールファイルをパッケージします
build.scripts.pack:win64	まず build:winアーキテクチャコードをコールしてからcompile:win64をコールして .exeインストールファイルをパッケージします

ステップ7：パッケージコマンドの実行

- Mac .dmgインストールファイルのパッケージ：

```
$ cd [プロジェクトディレクトリ]
$ npm run pack:mac
```

実行に成功すると、パッケージツールは `bin/your-app-name-0.1.0.dmg` インストールファイルを新規作成しますので、このファイルのリリースを選択してください。

- Windows .exeインストールファイルのパッケージ：

```
$ cd [プロジェクトディレクトリ]
$ npm run pack:win64
```

実行に成功すると、パッケージツールは `bin/your-app-name Setup 0.1.0.exe` インストールファイルを新規作成しますので、このファイルのリリースを選択してください。

注意：
TRTC Electron SDK は、プラットフォームパッケージ（Mac下でWindowsをパッケージする.exe ファイル、またはWindowsプラットフォーム下でMacをパッケージする .dmgファイルなど）を一時的にサポートしません。現在プラットフォームを跨がるパッケージ方法を検討中ですのでご期待ください。

よくあるご質問

ファイアウォールにはどのような制限がありますか。

SDKがUDPプロトコルを使用してオーディオビデオ伝送を行っているため、UDPをブロックするオフィスネットワークでは使用できません。類似した問題がおありの際は、企業ファイアウォール制限の対応をご参照ください。

Electronのインストールまたはパッケージ化にトラブルが生じました。

Electron統合中にトラブルが生じた場合、例えばインストールのタイムアウトまたは失敗、パッケージ後にtrtc_electron_sdk.nodeファイルのロード失敗などの状況が生じた場合は、お問い合わせまでご連絡いただければご質問にお答えします。

参考ドキュメント

- SDK APIマニュアル
- SDK更新ログ
- Simple Demoソースコード
- API Exampleソースコード
- Electronについてのよくあるご質問

クイックインテグレーション(Flutter)

最終更新日：：2022-05-30 16:32:37

ここでは、主にTencent CloudのTRTC SDK(Flutter)をプロジェクトに素早く統合する方法を紹介します。以下の手順に従って設定すれば、 SDK の統合作業を完了できます。

> 注意：
> 現在Windows/MacOs端末は画面共有およびデバイス選択の機能をサポートしていません。

環境要件

- Flutter 2.0以降のバージョン。
- **Android端末向け開発：**
  - Android Studio 3.5以降のバージョン。
  - AppにはAndroid 4.1以降のバージョンのデバイスが必要です。

iOS & macOS端末向け開発：

- Xcode 11.0以降のバージョン。
- osxシステムには10.11以降のバージョンが必要です。
- プロジェクトが有効な開発者による署名を設定済みであることを確認してください。
- **Windows端末向け開発：**
  - OS:Windows 7 SP1以降のバージョン(x86-64に基づく64ビットOS）。
  - ディスク容量：IDEと一部のツールのインストールに必要な容量を除く、少なくとも1.64 GB以上の空き容量を確保するようにしてください。
  - Visual Studio 2019をインストールします。

SDKの統合

Flutter SDKはpubライブラリに公開されています。 `pubspec.yaml` を設定することで、更新を自動的にダウンロードできます。

1. プロジェクトの `pubspec.yaml` に次の依存関係を記述します。

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

2. **カメラ**と**マイク**の許可が有効になると、音声通話機能が起動します。

- iOS 端末
- macOS 端末
- Android 端末
- Windows 端末

i. `Info.plist` にカメラとマイクの許可申請を追加する必要があります。

```
<key>NSCameraUsageDescription</key>
<string>通常のビデオ通話が行えるようにカメラを許可します</string>
<key>NSMicrophoneUsageDescription</key>
<string>通常の音声通話が行えるようにマイクの権限を承認します</string>
```

ii. フィールド `io.flutter.embedded_views_preview` を追加し、値をYESに設定します。

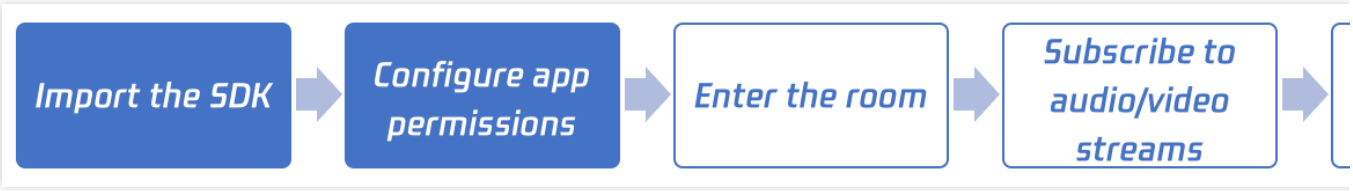
よくあるご質問

- iOSのパッケージングの実行時にCrashした場合はどうすればいいですか。
- iOSでビデオが表示できない場合（Androidでは正常）はどうすればいいですか。
- SDKのバージョンを更新した後、iOS CocoaPodsの実行にエラーが出た場合はどうすればいいですか。
- Android Manifest merge failedでコンパイルに失敗した場合はどうすればいいですか。
- 署名がないため、実機でのデバック時にエラーが出た場合はどうすればいいですか。
- プラグインの中のswiftファイルを追加・削除した後、build（ビルド）時に対応するファイルが見つからないのはなぜですか。
- Run（実行）時にエラー：「Info.plit, error: No value at that key path or invalid key path: NSBonjourServices」が出た場合はどうすればいいですか。
- Pod install時にエラーが出た場合はどうすればいいですか。
- Run（実行）時にiOS版で依存エラーが出た場合はどうすればいいですか。

QT

最終更新日：：2023-07-10 10:35:18

ここでは、主にTencent Cloud TRTC SDK(QTのWindowsバージョンおよびMacバージョン) をプロジェクトに素早く統合する方法を紹介します。以下の手順に従って設定すれば、SDKの統合作業を素早く完了できます。



Windows端末の統合

開発環境要件

OS：Windows 7以上のバージョン。
開発環境：Visual Studio 2015以上のバージョン、Visual Studio 2015の使用をお勧めします。VS関連のQT開発環境をすでに設定していることが前提になります。

explain

VS関連のQT開発環境の設定手順に詳しくない場合は、README の操作手順のステップ4の関連内容をご参照ください。

操作手順

このセグメントでは、簡単なQTプロジェクトを作成し、Visual Studioプログラムの中でC++ SDKを統合する方法を紹介します。

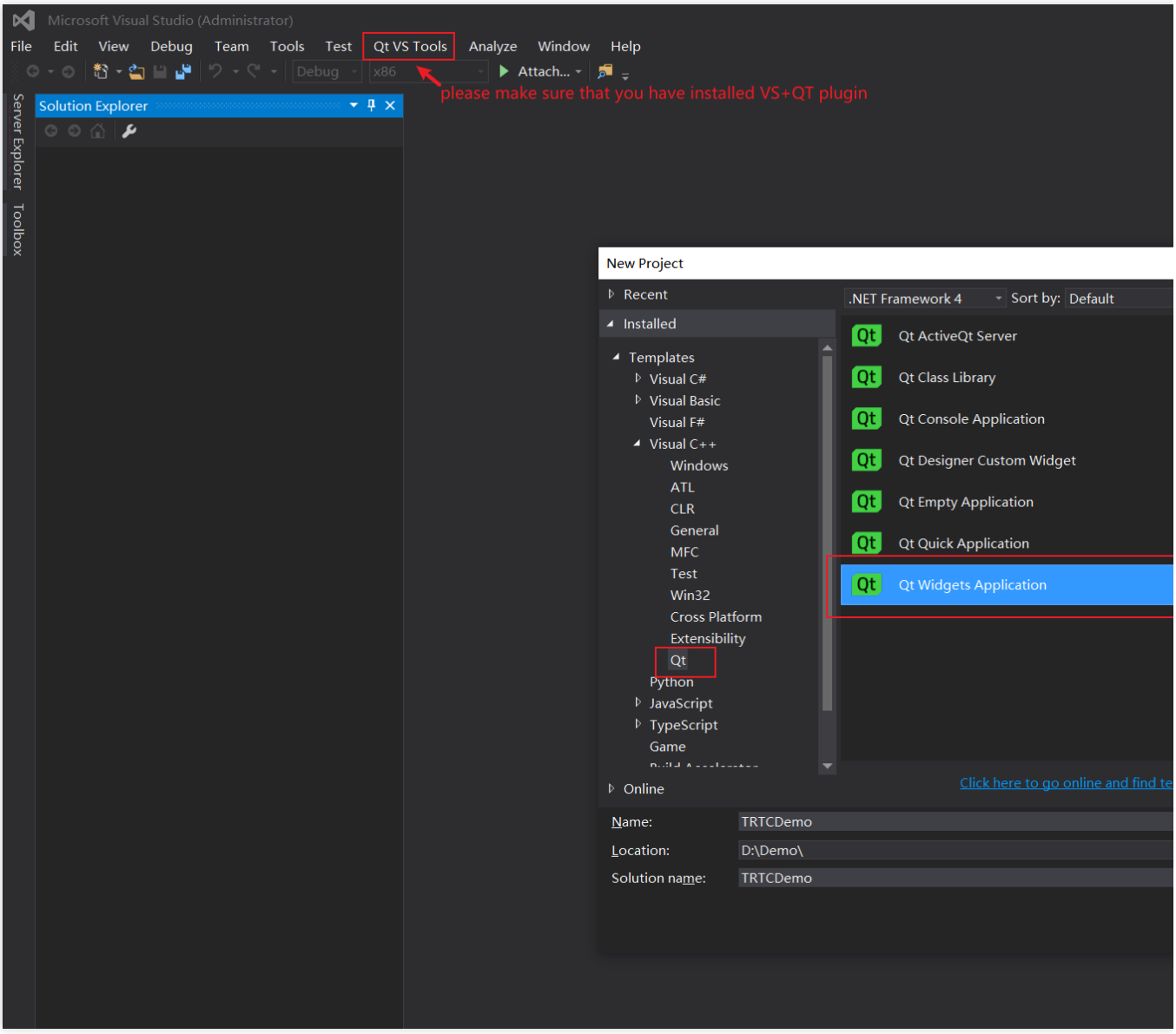
ステップ1： SDKのダウンロード

1. SDKのダウンロードを行い、解凍して開きます。
ここの例では、 SDKディレクトリのC++バージョンのSDKファイルをインポートしさえすればOKです。64ビットの場合、SDKは `./SDK/CPlusPlus/Win64/` にあり、主に次のいくつかの部分が含まれます。

ディレクトリ名	説明
include	詳細なインターフェースの説明がついたAPIヘッダーファイル
lib	編集用の.libファイルおよび実行時にローディングする.dllファイル

ステップ2：プロジェクトの新規作成

Visual Studio 2015を例にとると、ローカルですでにQTおよびVS開発プラグインをインストール済みという前提で、Visual Studioを開きます。下図のように、 `TRTCDemo` という名前のQTアプリケーションを新規作成します。



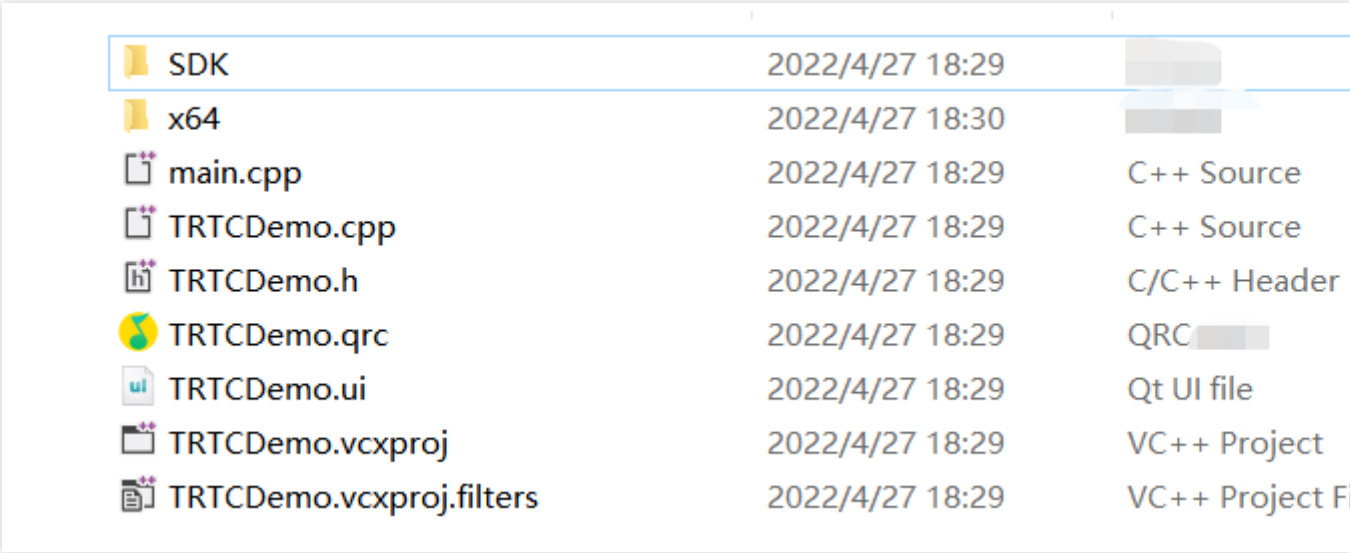
クイックインテグレーションの方法を説明しやすくするため、ガイドの中で**Qt Widgets Application**タイプを選択し、**OK**をクリックします。次のページで**Next**をクリックし、プロジェクトを作成すれば完了です。

ステップ3：ファイルのコピー

下図のように、解凍後のSDKフォルダをTRTCDemo.vcxprojが存在するディレクトリ下にコピーします。

explain

この時点ではC++ SDKだけが必要なため、 SDKパスのCSharpディレクトリを削除しても構いません。



ステップ4：プロジェクト設定の修正

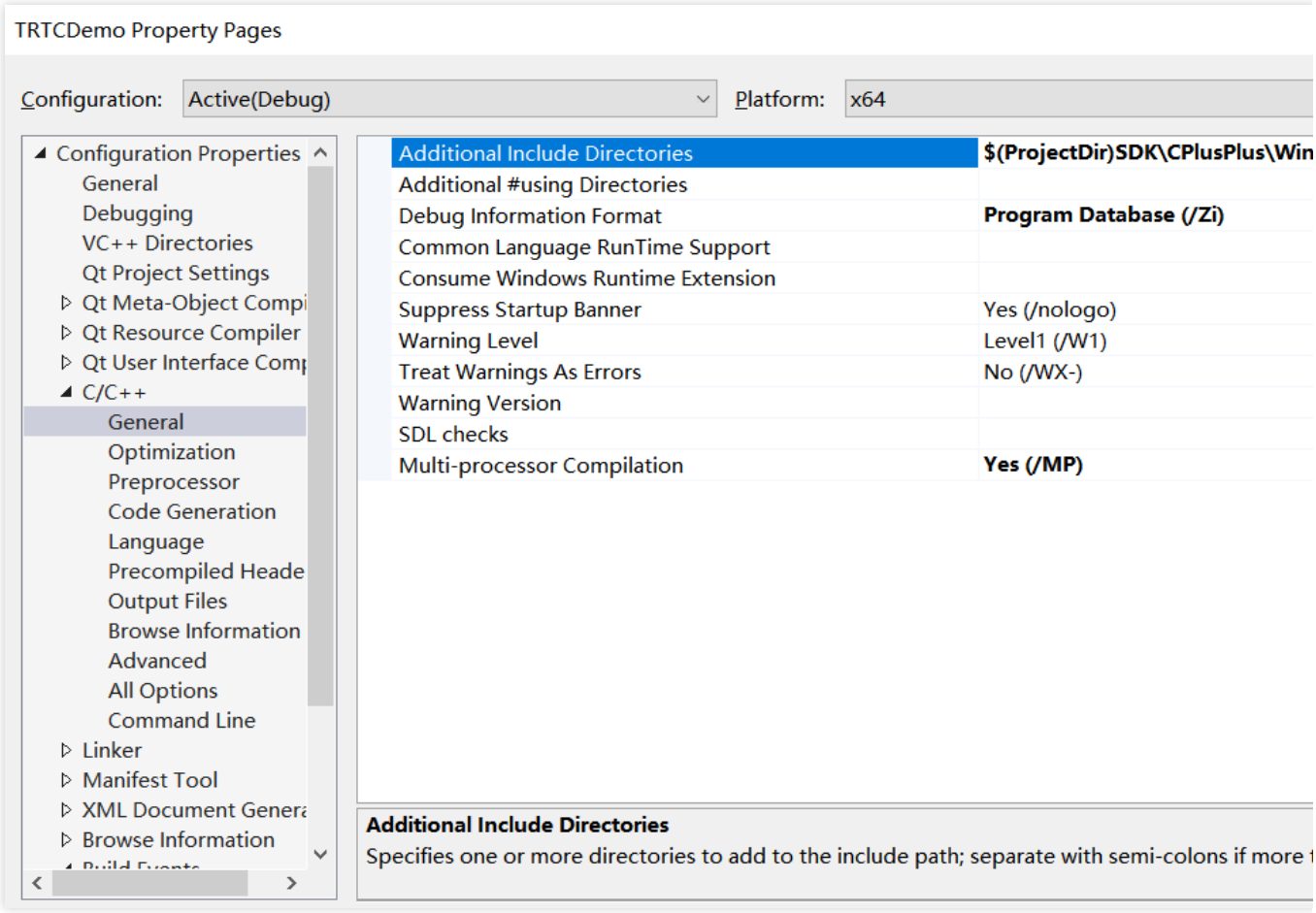
TRTCDemoの属性のページを開きます。**ソリューションのResource Manager >TRTCDemoプログラムの右クリックメニュー>属性**と進みます。次のステップにしたがって設定してください。

1. **Includeのディレクトリの追加：**

下図のように、**C/C++ > 通常 > Includeディレクトリの追加**で、64ビットの場合は、SDKヘッダーファイルディレクトリ `$(ProjectDir)SDK\\CPlusPlus\\Win64\\include` と `$(ProjectDir)SDK\\CPlusPlus\\Win64\\include\\TRTC` を追加します。

explain

32ビットの場合は、SDKヘッダーファイルディレクトリを `$(ProjectDir)SDK\\CPlusPlus\\Win32\\include` と `$(ProjectDir)SDK\\CPlusPlus\\Win32\\include\\TRTC` に設定する必要があります。

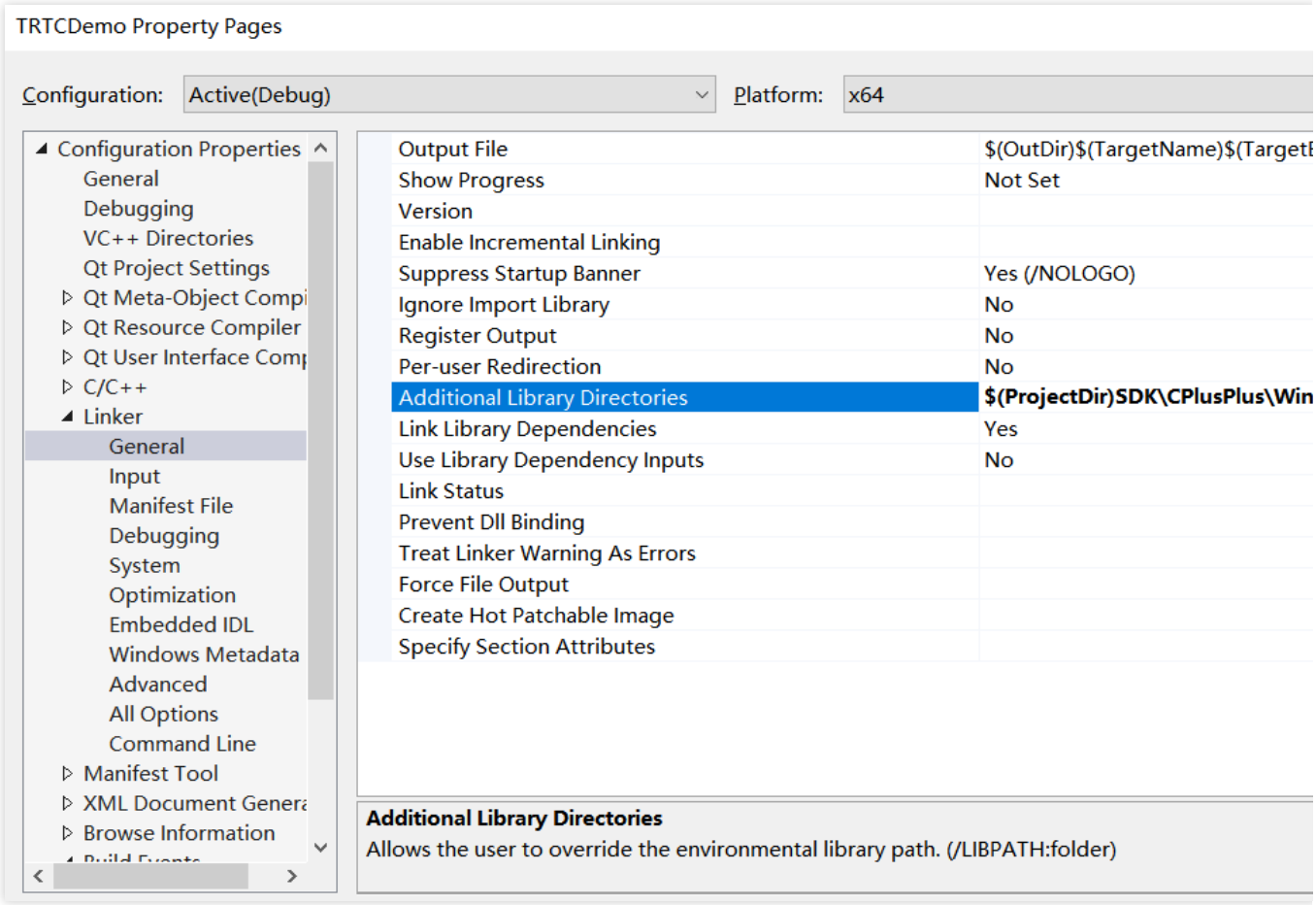


2. **ライブラリのディレクトリの追加：**

下図のように、**リンカー>通常>ライブラリディレクトリの追加**で、64ビットの場合は、SDKライブラリディレクトリ `$(ProjectDir)SDK\\CPlusPlus\\Win64\\lib` を追加します。

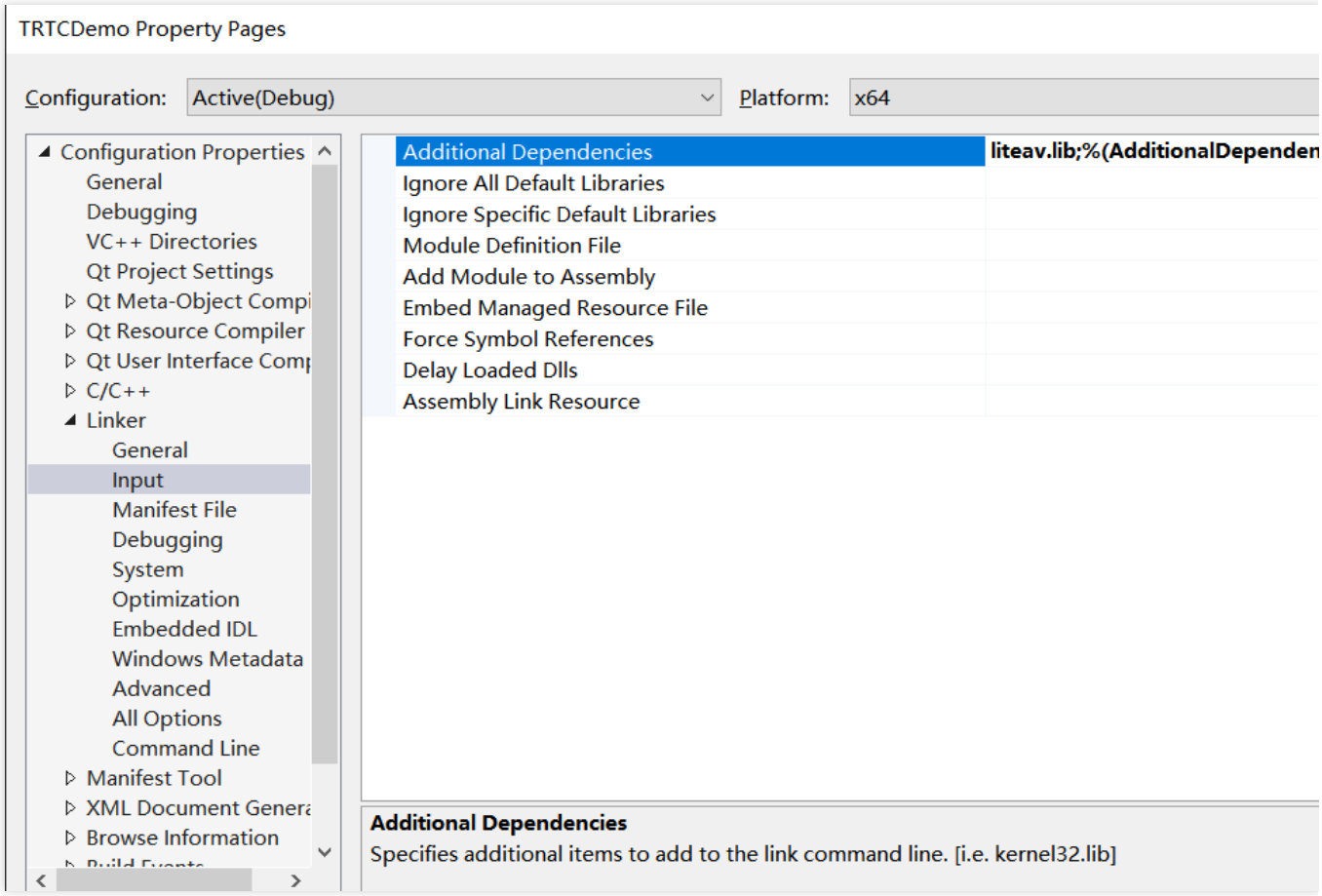
explain

32ビットの場合は、SDKライブラリディレクトリを `$(ProjectDir)SDK\\CPlusPlus\\Win32\\lib` に設定する必要があります。



3. **ライブラリファイルの追加：**

下図のように、**リンカー>入力>依存プロジェクトの追加**で、 SDKライブラリファイル `liteav.lib` を追加します。

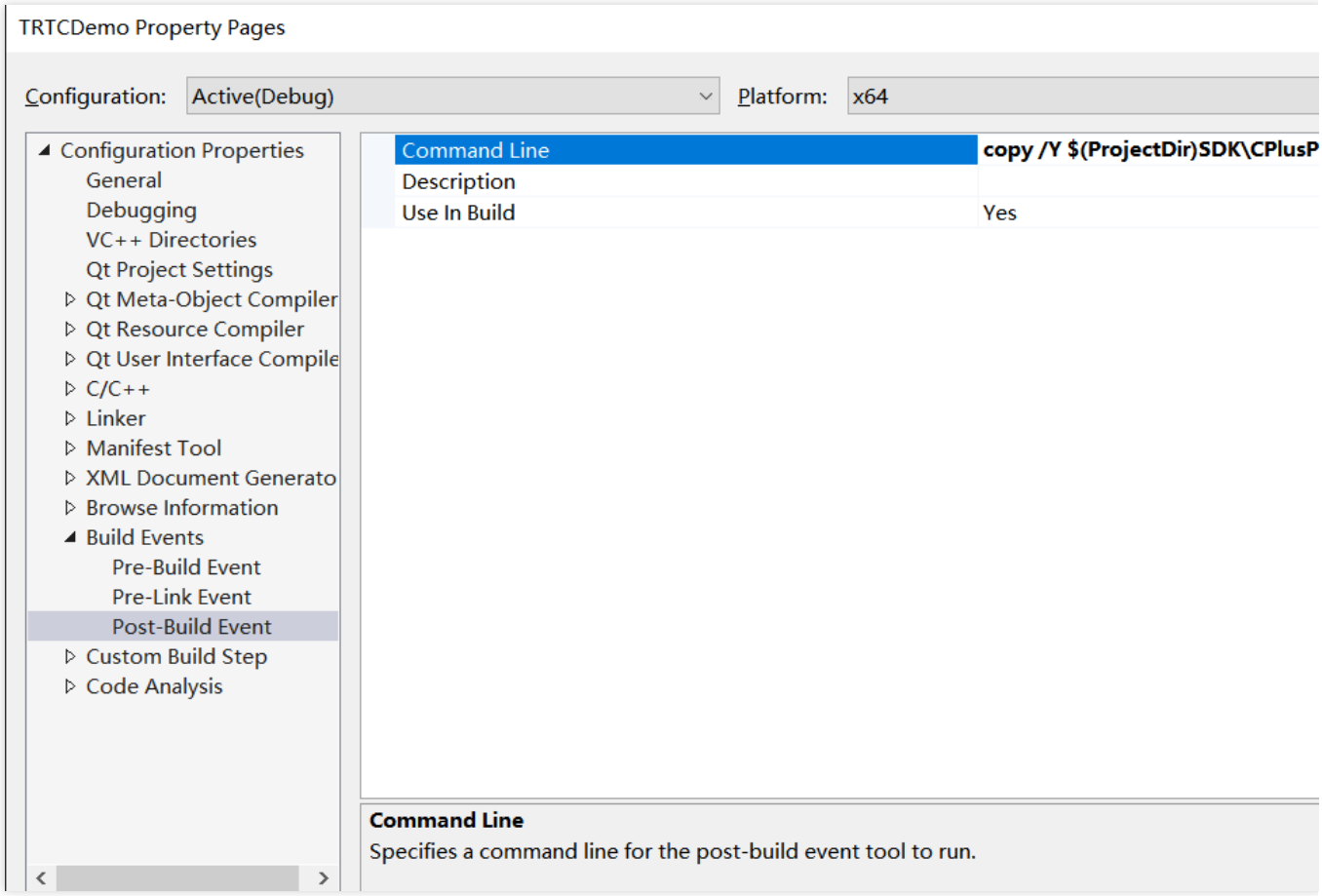


4. **copyコマンドの追加：**

下図のように、**イベントの生成>後続のイベントの生成>コマンドライン**で、コピーコマンド `copy /Y $(ProjectDir)SDK\\CPlusPlus\\Win64\\lib\\*.dll $(OutDir)` を追加します。編集が完了すると、自動で SDKの.dll ファイルがプログラムの実行ディレクトリの下にコピーされます。

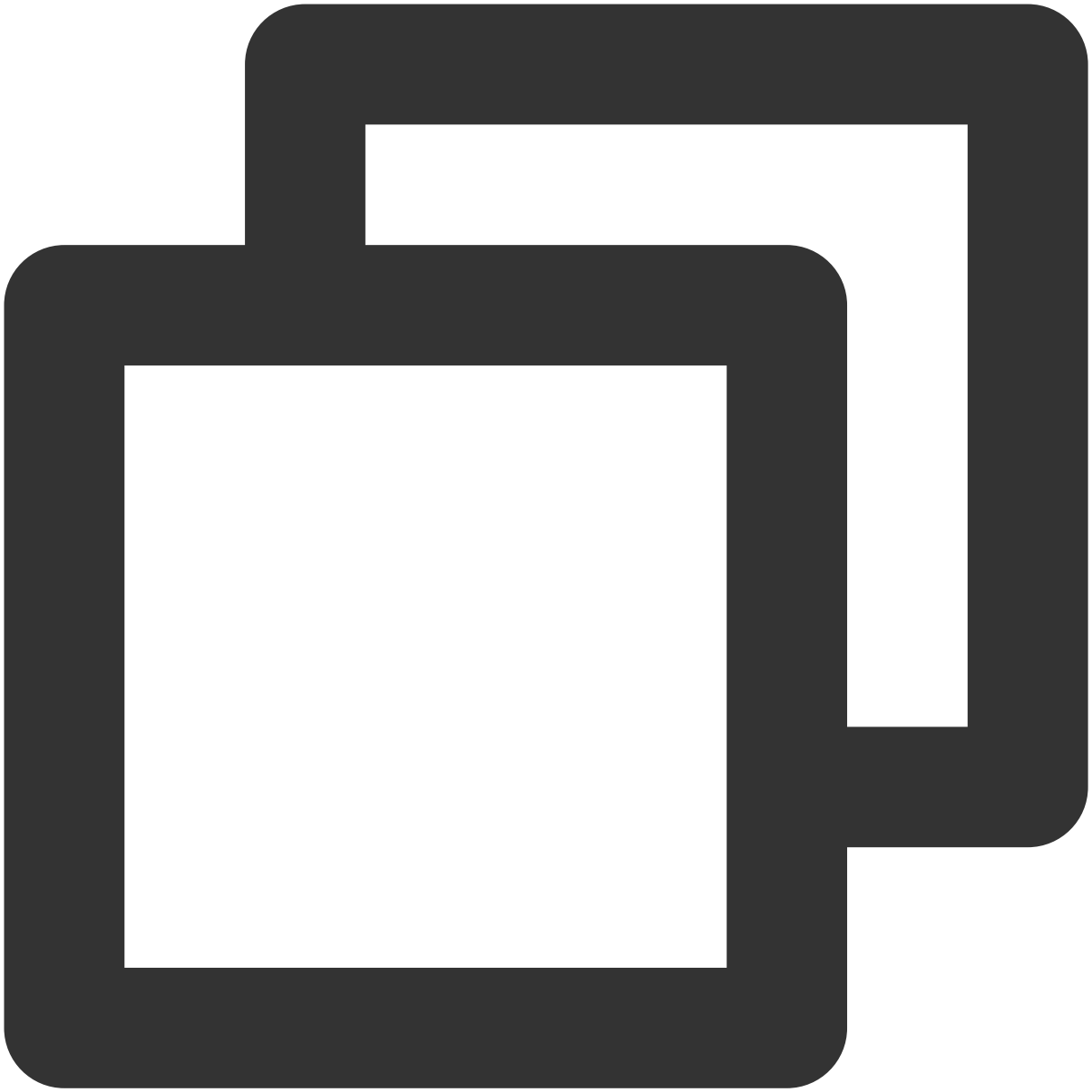
explain

32ビットの場合は、追加するコピーコマンドは `copy /Y $(ProjectDir)SDK\\CPlusPlus\\Win32\\lib\\*.dll $(OutDir)` となります。



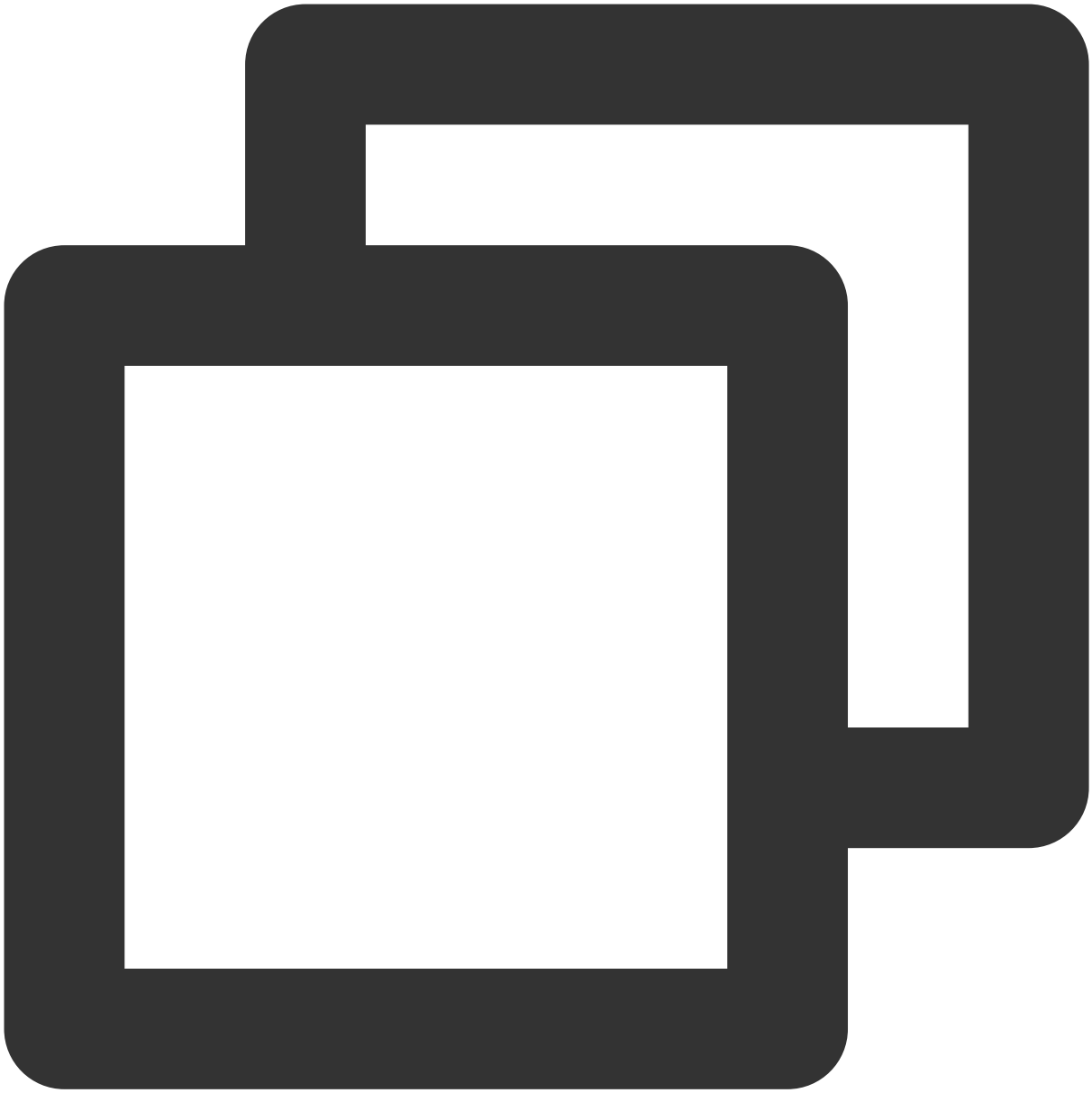
ステップ5： SDKバージョン番号の印刷

1. `TRTCDemo.cpp` ファイルのトップにヘッダーファイルをインポートして追加します。コードは次のとおりです。



```
#include "ITRTCCloud.h"
#include <QLabel>
```

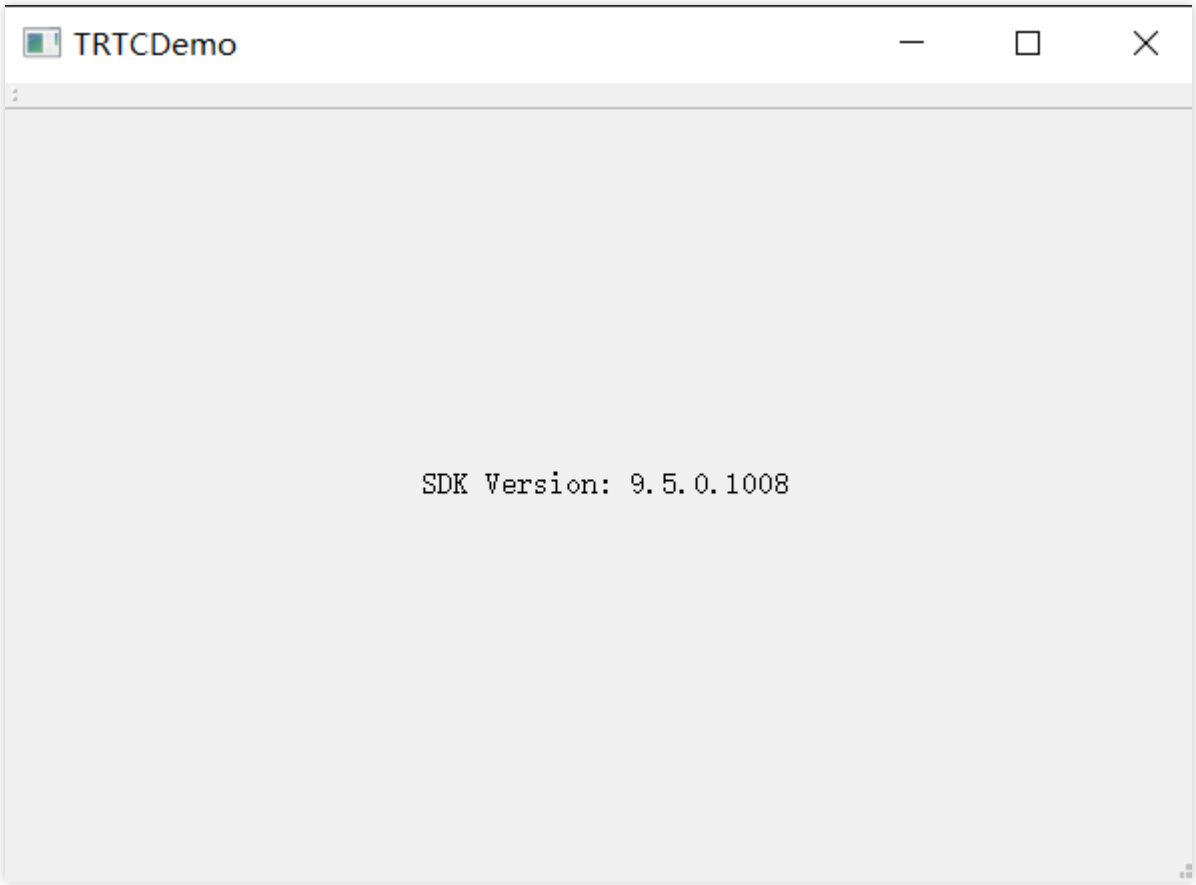
2. `TRTCDemo.cpp` ファイルの `TRTCDemo::TRTCDemo` コンストラクタの中に、以下のテストコードを追加します。



```
ITRTCCloud * pTRTCCloud = getTRTCShareInstance();
std::string version(pTRTCCloud->getSDKVersion());

QString sdk_version = QString("SDK Version: %1").arg(version.c_str());
QLabel* label_text = new QLabel(this);
label_text->setAlignment(Qt::AlignCenter);
label_text->resize(this->width(), this->height());
label_text->setText(sdk_version);
```

3. F5を押下して動作させ、下図のとおり、SKDのバージョン番号を印刷します。



Mac端末統合

開発環境要件

オペレーティングシステム：Mac10.10以上のバージョン。
開発環境：Qt Creator 4.10.3以上のバージョン。Qt Creator 4.13.3以上のバージョンの使用を推奨。
開発フレームワーク：Based on Qt 5.10以上。

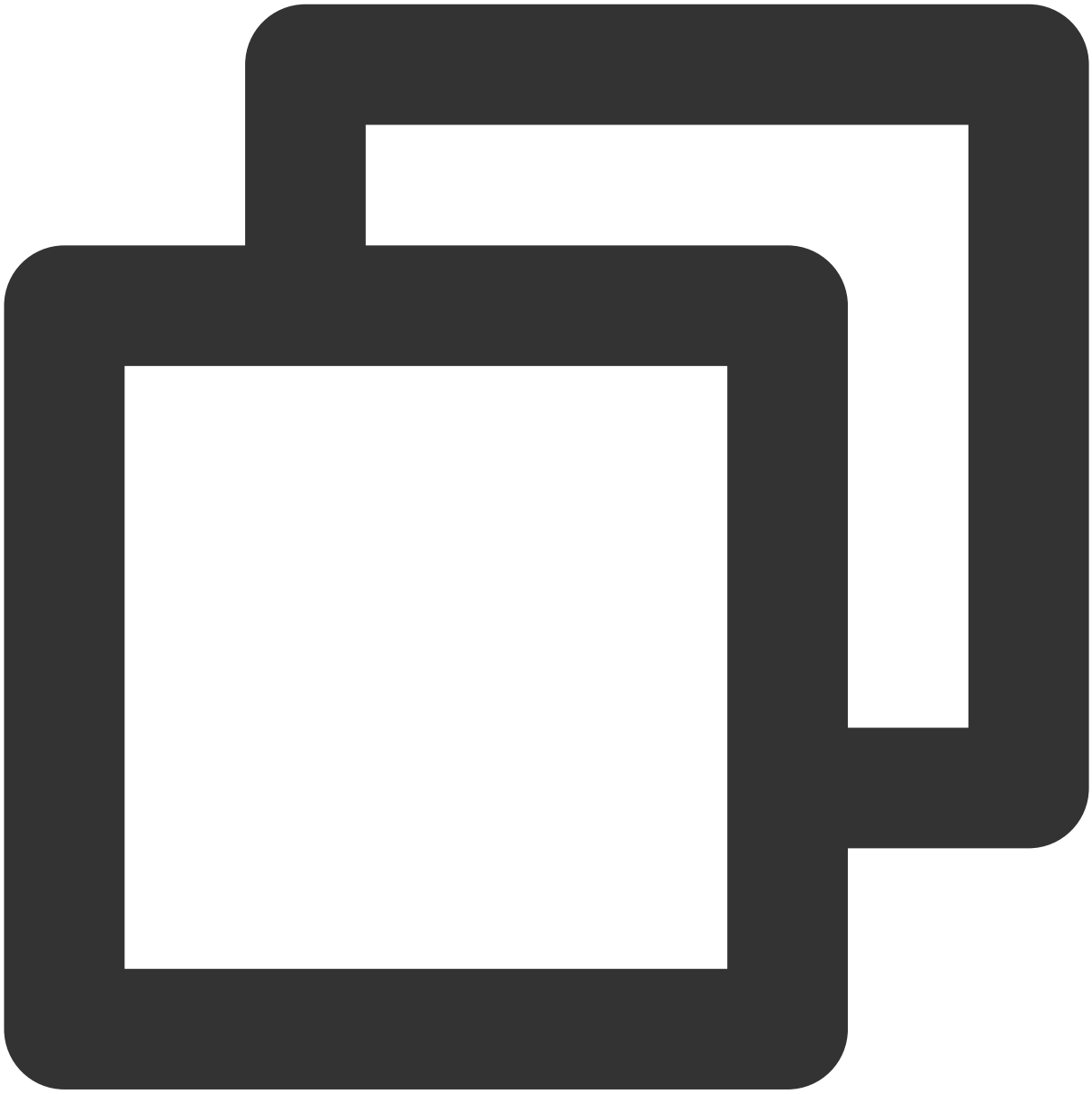
操作手順

ここでは簡単なQTTest項目の作成を例に、Ｑｔ Creatorプロジェクトで C++クロスプラットフォームSDKを統合する方法をご紹介します。

1. **C++クロスプラットフォームSDKのダウンロード**
1.1 SDKをダウンロードし、ファイルを解凍して開きます。
1.2 QTTestと同じクラスのディレクトリ下で空のSDKフォルダを作成し、SDK内の `TXLiteAVSDKTRTCMacx.x.x/SDK/TXLiteAVSDKTRTC_Mac.framework` を、QTTestプロジェクトディレクトリと同じクラスのディレクトリにあるSDKフォルダにコピーします。

2. **QTTest.proの設定**

QTTest プロジェクトディレクトリを開き、任意のテキストエディタを使用して `QTTest.pro` ファイルを開いてから、SDK関連の引用を追加します。



```
INCLUDEPATH += $$PWD/.
DEPENDPATH += $$PWD/.

LIBS += "-F$$PWD/base/util/mac/usersig"
LIBS += "-F$$PWD/../SDK"
LIBS += -framework TXLiteAVSDK_TRTC_Mac
LIBS += -framework Accelerate
```

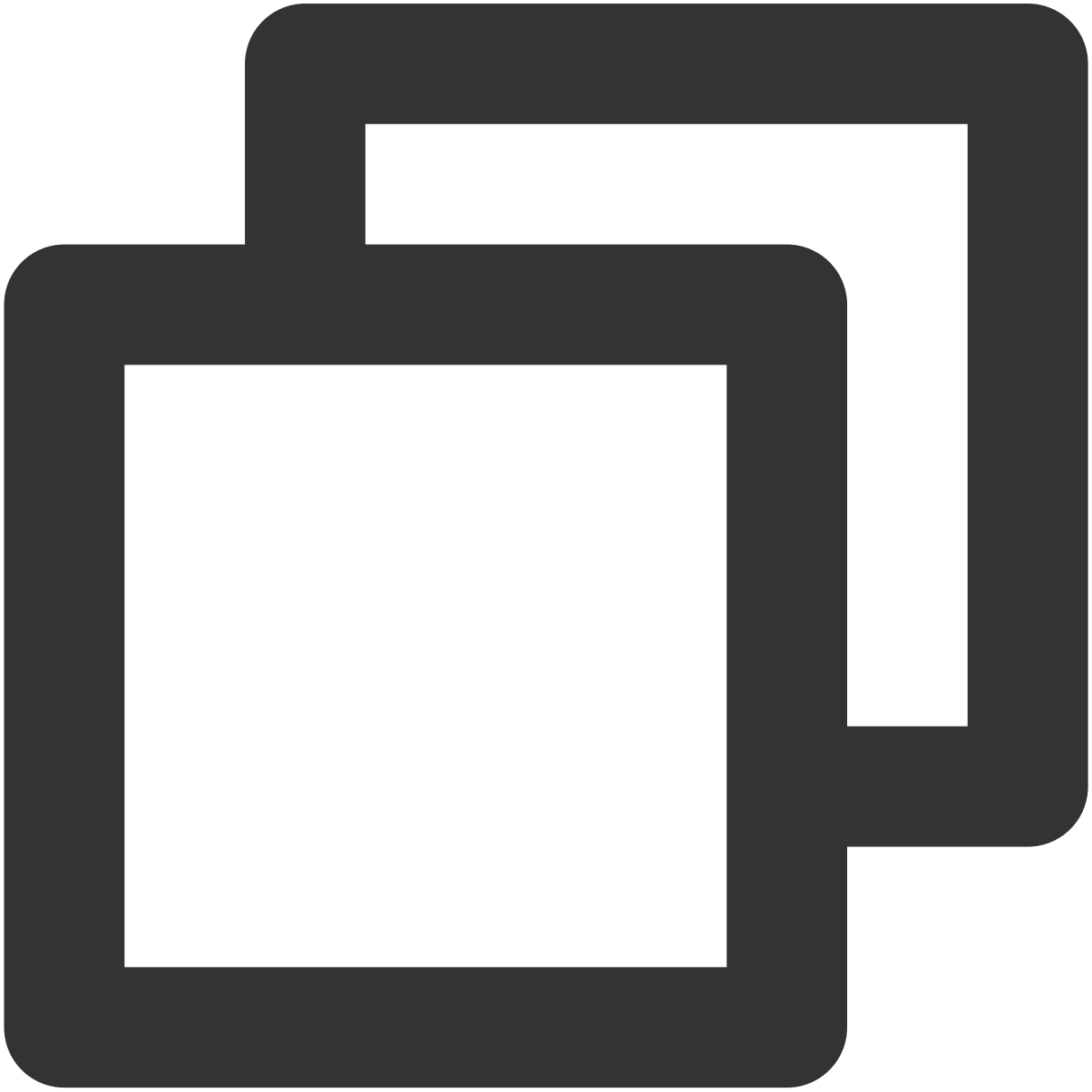
```
LIBS += -framework AudioUnit

INCLUDEPATH += $$PWD/../SDK/TXLiteAVSDK_TRTC_Mac.framework/Headers/cpp_interface

INCLUDEPATH += $$PWD/base/util/mac/usersig/include
DEPENDPATH += $$PWD/base/util/mac/usersig/include
```

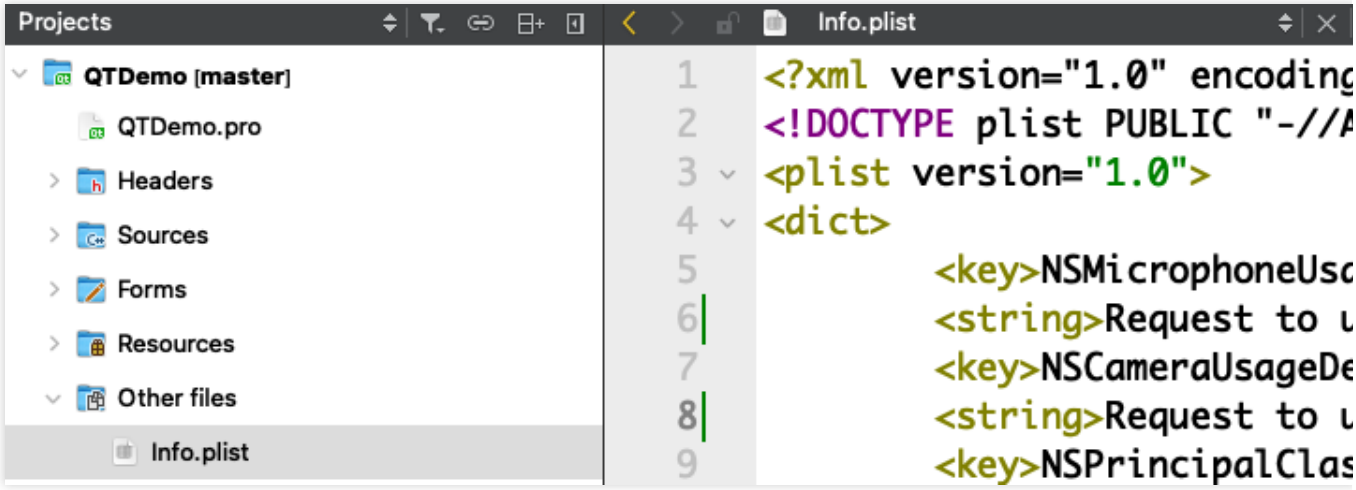
3. **カメラおよびマイクの使用権限の承認**

SDKではカメラおよびマイクを使用しますので、対応する `Info.plist` に該当する権限申請説明を追加する必要があります。



```
NSMicrophoneUsageDescription：マイクの使用申請
NSCameraUsageDescription：カメラの使用申請
```

下図に示すとおり：



4. TRTC SDKの引用

ヘッダーファイル `#include "ITRTCCloud.h"` によって直接引用することができます。

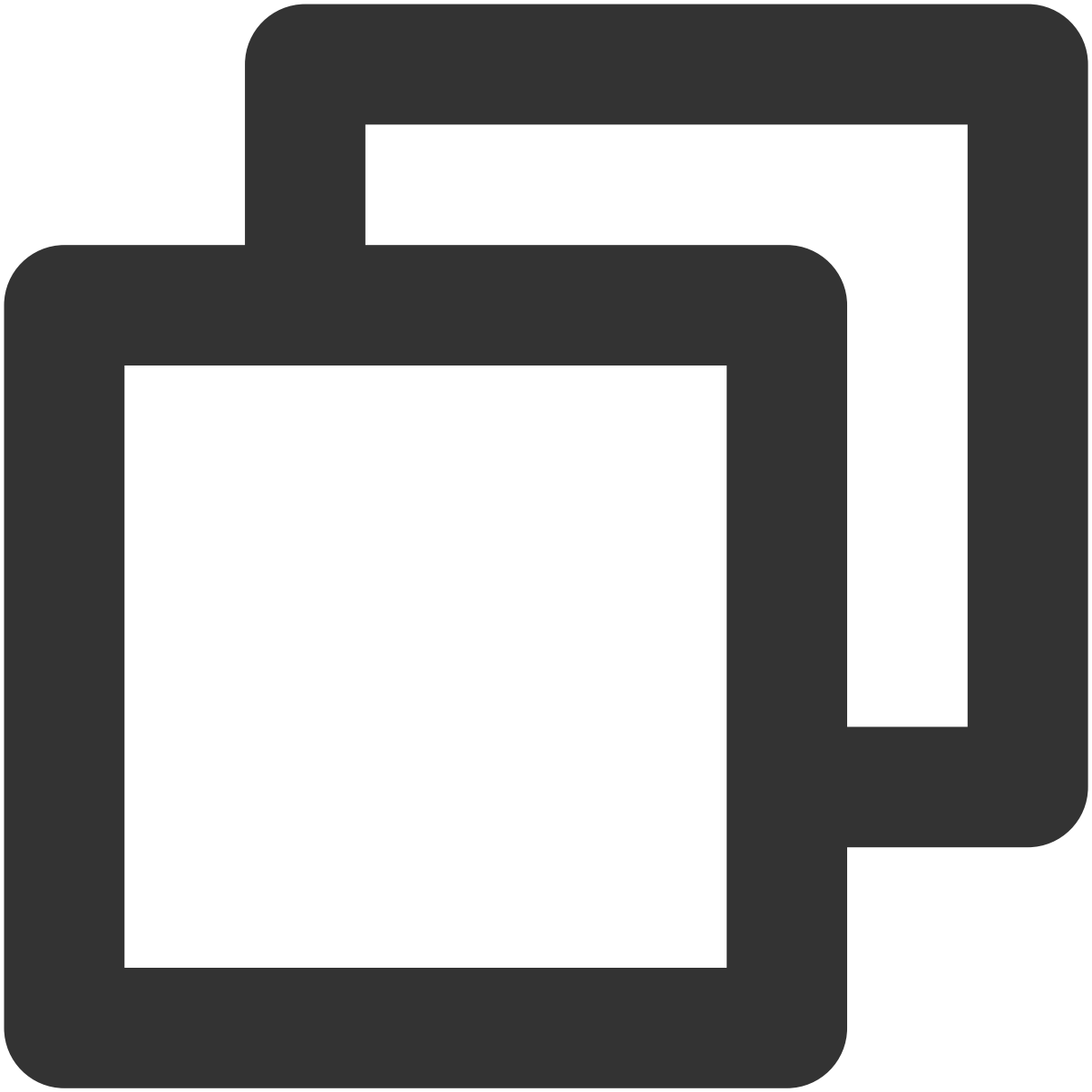
ネームスペースの利用：C++のすべてのプラットフォームのインターフェースのメソッド、タイプなどはいずれもtrtcネームスペースで定義されています。コードをより簡潔にするため、trtcネームスペースを直接使用することをお勧めします。

explain

ここまでで統合作業はすでに完了していますので、プロジェクトをコンパイルして実行できます。 Demoを使用するためのクロスプラットフォームSDKのAPIの詳細については、QTDemoをダウンロードしてご参照ください。

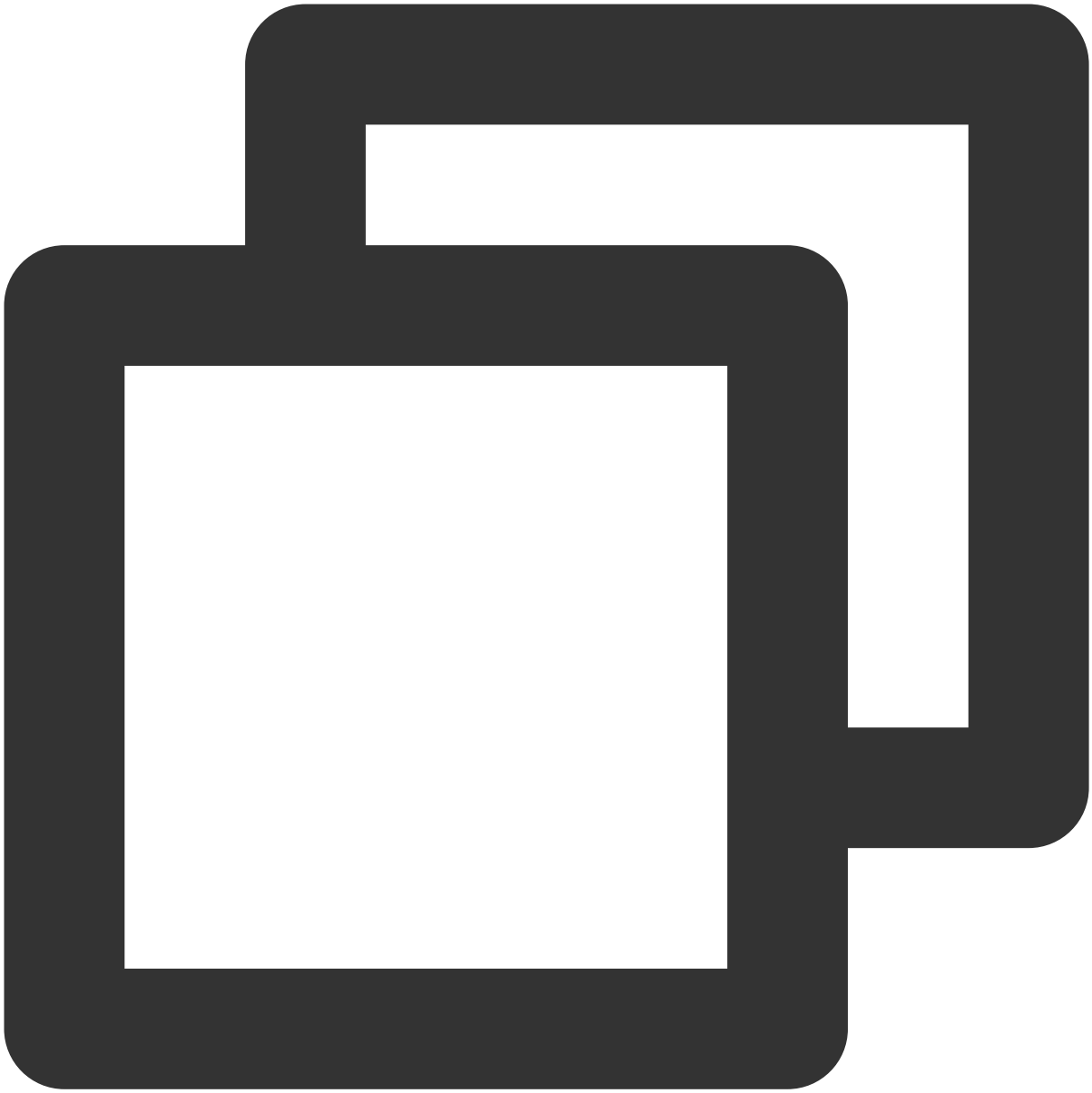
よくあるご質問

次のエラーが生じた場合は、前述のプログラム設定にしたがって、SDKヘッダーファイルのディレクトリが正しく追加されているかチェックしてください。



```
fatal error C1083: includeファイルを開くことができません: “TRTCCloud.h”: No such file or
```

次のエラーが生じた場合は、前述のプログラム設定にしたがって、SDKディレクトリとライブラリファイルが正しく追加されているかチェックしてください。



```
error LNK2019: 解析できない外部シンボル "__declspec(dllimport) public: static class TXS
```

クイックインテグレーション(Unity)

最終更新日：：2022-07-20 16:48:11

ここでは、主にTencent Cloud TRTC SDK（Unity）をプロジェクトに素早く統合する方法をご紹介します。以下の手順に従って設定すれば、SDKの統合作業を完了できます。

環境要件

- Unityの推奨バージョン： 2020.2.1f1c1。
- 現在、Android、iOS、Windows、Mac（Macはベータ版テスト中です)プラットフォームをサポートしています。
- `Android Build Support` 、 `iOS Build Support` 、 `Winodows Build Support` および `MacOs Build Support` モジュールが含まれている必要があります。

- その内、iOS端末の開発には以下が必要です。
  - Xcode 11.0およびそれ以降のバージョン。
  - プロジェクトが有効な開発者による署名を設定済みであることを確認してください。

SDKの統合

1. SDKおよび付属するDemoソースコードをダウンロードします。
2. 解凍後、プロジェクト内の `TRTCUnitySDK/Assets/TRTCSDK/SDK` フォルダを自分のプロジェクトのAssetsディレクトリ下にコピーします。

よくあるご質問

Androidではネットワーク権限の問題が表示されますか。

プロジェクト内の `/Assets/Plugins/AndroidManifest.xml` ファイルを同じレベルのディレクトリ下に配置してください。

Androidではオーディオビデオの権限はないのですか。

Android端末のマイク、カメラの権限は手動で申請します。具体的な方法については以下のコードをご参照ください。

```
#if PLATFORM_ANDROID
if (!Permission.HasUserAuthorizedPermission(Permission.Microphone))
{
Permission.RequestUserPermission(Permission.Microphone);
}
if (!Permission.HasUserAuthorizedPermission(Permission.Camera))
{
Permission.RequestUserPermission(Permission.Camera);
}
#endif
```

2. 入室

Android&iOS&Windows&Mac

最終更新日：：2022-07-07 15:10:00

このドキュメントでは、主にTRTCルームに入る方法を紹介します。オーディオビデオルームに入った後でのみ、ユーザーはルーム内の他のユーザーのオーディオビデオストリームをサブスクリプションしたり、自分のオーディオビデオストリームをルーム内の他のユーザーに公開したりできます。

呼び出しガイド

手順1：SDKをインポートし、Appの権限を設定します

SDKのプロジェクトへのインポートを参照して、SDKをインポートします。

手順2：SDKインスタンスを作成し、イベント監視装置（リスナー）を設定します

各プラットフォームの初期化インターフェースを呼び出して、TRTCのオブジェクトインスタンスを作成します。

- Android java
- iOS&Mac ObjC
- Windows C++

```
// SDKインスタンス（シングルトンモード）を作成し、イベント監視装置（リスナー）を設定します
// Create trtc instance(singleton) and set up event listeners
mCloud = TRTCCloud.sharedInstance(getApplicationContext());
mCloud.setListener(this);
```

手順3：SDKイベントを監視します

イベントコールバックインターフェイスを設定することにより、SDKの実行中に発生するエラー情報、警告情報、トラフィック統計情報、ネットワーク品質情報、およびさまざまなオーディオビデオイベントを監視できます。

- Android

- iOS&Mac ObjC
- Windows C++

独自のクラスに**TRTCCloudListener**を継承させ、onError関数をオーバーロードし、最後に** set Listener **インターフェイスを介してthisポインターをSDKに設定して、現在のクラスでSDKからのコールバックイベントを監視できます。
<dx-code-holder data-codeindex="0"></dx-code-holder>

手順4：入室パラメータTRTCParamsを準備します

enterRoomインターフェースを呼び出すとき、2つの主要なパラメーター、つまり `TRTCParams` と `TRTCAppScene` を入力してください。これらについては、以下で詳しく説明します：

パラメータ1：TRTCAppScene

このパラメータは、ユースケース、つまり**オンラインライブストリーミング**または**リアルタイム通話**を指定するために使用されます：

- **リアルタイム通話：**
  `TRTCAppSceneVideoCall` と `TRTCAppSceneAudioCall` の2つのオプションがあり、それぞれビデオ通話と音声通話です。このモードは、1対1のオーディオビデオ通話、または参加者が300人未満のオンライン会議に適しています。

- **オンラインライブストリーミング：**
  `TRTCAppSceneLIVE` と `TRTCAppSceneVoiceChatRoom` の2つのオプションがあり、それぞれビデオライブストリーミングとオーディオライブストリーミングです。このモードは、10万人未満のオンラインライブストリーミングシナリオに適していますが、以下に説明するTRTCParamsパラメータで**ロール（role）**のフィールドを指定する必要があります。つまり、ルーム内のユーザーは、**ホスト(anchor)**と視聴者(audience)**の2つの異なる役割に分けられます。

パラメータ2：TRTCParams

TRTCParamsは多くのフィールドで構成されていますが、通常は次のフィールドへの入力のみを気にしてください：

パラメータ名	フィールドの意味	補足説明	データのタイプ	記入例
SDKAppID	アプリケーションID	このSDKAppIDはTencent Real-Time Communicationコンソールにあります。見つからない場合は、「アプリケーションの作成」ボタンをクリックして新しいアプリケーションを作成してください。	数字	140000

パラメータ名	フィールドの意味	補足説明	データのタイプ	記入例
userId	ユーザーID	すなわちユーザー名です。大文字と小文字の英字（a-z、A-Z）、数字（0-9）、アンダースコア、およびハイフンのみが許可されます。TRTCは、2つの異なるデバイスで同時に入室する同じuserIdをサポートしていないことに注意してください。そうでない場合には、互いに干渉します。	文字列	「denr 「1233
userSig	入室認証証明書	SDKAppIDとuserIdを使用してuserSigを計算できます。計算方法については、UserSigの計算と使用をご参照ください。	文字列	eJyrVa
roomId	ルーム番号	数値タイプのルーム番号。strRoomIdとroomIdを混在させることはできないため、文字列タイプのルーム番号を使用する場合は、roomIdフィールドの代わりに**strRoomId**フィールドを使用することに注意してください。	数字	29834
strRoomId	ルーム番号	文字列タイプのルーム番号。strRoomIdとroomIdを混在させることはできません。「123」と123は、TRTCバックグラウンドサービスの同じ部屋ではありません。。	数字	29834
role	ロール	「キャスター」と「視聴者」の2つの役割に分かれています。このフィールドは、TRTCAppSceneが`TRTCAppSceneLIVE`または`TRTCAppSceneVoiceChatRoom`として指定されている場合にのみ指定してください。	列挙値	TRTCF はTRT

注意：

- TRTCは、2つの異なるデバイスで同時に入室する同じuserIdをサポートしません。そうでない場合には、互いに干渉します。
- 各端末のユースケースappSceneについては、統一してください。統一していない場合、想定外のトラブルが生じる恐れがあります。

步骤5：入室（enterRoom）

手順4の2つのパラメーター（TRTCAppSceneとTRTCParams）を準備した後、enterRoomインターフェース関数を呼び出して入室できます。

- Android Java
- iOS&Mac objectivec
- Windows C++

```
mCloud = TRTCCloud.sharedInstance(getApplicationContext());
mCloud.setListener(mTRTCCloudListener);

// TRTC入室パラメータを組み立てるには、TRTCParamsの各フィールドを独自のパラメータに置き換えてください
// Please replace each field in TRTCParams with your own parameters
TRTCCloudDef.TRTCParams param = new TRTCCloudDef.TRTCParams();
params.sdkAppId = 1400000123; // Please replace with your own SDKAppID
params.userId = "denny"; // Please replace with your own userid
params.roomId = 123321; // Please replace with your own room number
params.userSig = "xxx"; // Please replace with your own userSig
params.role = TRTCCloudDef.TRTCRoleAnchor;

// 「オンラインライブストリーミング」のシーンの場合、ユースケースをTRTC_APP_SCENE_LIVEに設定してください
// If your application scenario is a video call between several people, please use "TRTC_APP_SCENE_LIVE"
mCloud.enterRoom(param, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
```

イベントコールバック

正常に成功したら、SDKはonEnterRoom (result)イベントをコールバックします。ここで、resultは、入室にかかった時間をミリ秒(ms)の単位で表す0より大きい値になります。

入室に失敗したら、SDKはonEnterRoom(result)イベントもコールバックしますが、パラメータ `result` は負の数になり、その値は入室失敗のエラーコードです。

- Android Java
- iOS&Mac ObjC
- Windows C++

```
// SDKのonEnterRoomイベントを監視し、入室に成功したかどうかを確認します
// Listen to the onEnterRoom event of the SDK and learn whether the room is successfully entered
@Override
public void onEnterRoom(long result) {
```

```
if (result > 0) {
Log.d(TAG, "Enter room succeed");
}else{
Log.d(TAG, "Enter room failed");
}
}
```

Web

最終更新日：：2022-08-08 15:25:07

ここでは主に、TRTCルームへの入室方法についてご説明します。オーディオビデオルームに入室すると、ユーザーはルーム内の他のユーザーのオーディオビデオストリーミングをサブスクリプションしたり、ルーム内の他のユーザーに自分のオーディオビデオストリーミングを公開したりすることができます。

TRTC Web SDKの使用中には、以下のオブジェクトが頻繁に登場します。

- Client オブジェクト。ローカルクライアントを表します。Clientクラスのメソッドにより、通話ルームへの入室、ローカルストリームの公開、リモートストリームのサブスクリプションなどの機能を提供します。
- Streamオブジェクト。オーディオビデオストリーミングオブジェクトを表し、ローカルのオーディオビデオストリーミングオブジェクトLocalStream、およびリモート側のオーディオビデオストリーミングオブジェクトRemoteStreamが含まれます。Streamクラスのメソッドでは主に、オーディオビデオストリーミングオブジェクトのアクションを提供し、これにはオーディオおよびビデオの再生コントロールが含まれます。

ステップ1： Clientオブジェクトの新規作成

TRTC.createClient()メソッドによって、Clientオブジェクトを作成します。主なパラメータは次のとおりです。

パラメータ名	フィールドの意味	補足説明	データタイプ	入力例	デフォ

パラメータ名	フィールドの意味	補足説明	データタイプ	入力例	デフォ
mode	ユースケース	リアルタイム通話モードでは `rtc` に設定します。このモードは1対1のオーディオビデオ通話、または参加者が300人以内のオンラインミーティングに適しています。オンラインライブストリーミングモードでは `live` に設定します。このモードは10万人以内のオンラインライブストリーミングのシーンに適しています	string	rtc	rtc

パラメータ名	フィールドの意味	補足説明	データタイプ	入力例	デフォ
sdkAppId	アプリケーションID	TRTCコンソールでこのsdkAppIdを確認できます。確認できない場合は、「アプリケーションの作成」ボタンをクリックして、新しいアプリケーションを作成します。	number	1400000123	なし
userId	ユーザーID	ユーザー名には、大文字と小文字のアルファベット（a-z、A-Z）、数字（0-9）およびアンダースコアとハイフンのみが使用可能です。 **注意：** TRTCは、同じuserIdによる2つの異なるデバイスの同時入室をサポートしていません。同時に入室した場合は相互に干渉します。	string	「denny」または「123321」	なし

パラメータ名	フィールドの意味	補足説明	データタイプ	入力例	デフォ
userSig	入室認証証書	計算方法については UserSigの計算、使用方法をご参照ください。	string	eJyrVareCeYrSy1SsII...	なし
useStringRoomId	文字列のルームナンバーの有効化または無効化	stringタイプのroomIdを使用するかどうか。	boolean	true	false

パラメータのより詳細な説明については、TRTC.createClient()をご参照ください。

```
// リアルタイム通話モードでのクライアントオブジェクト作成
const client = TRTC.createClient({
mode: 'rtc',
sdkAppId,
userId,
userSig
});
// インタラクティブライブストリーミングモードでのクライアントオブジェクト作成
const client = TRTC.createClient({
mode: 'live',
sdkAppId,
userId,
userSig
});
```

ステップ2：オーディオビデオ通話ルームへの参加

Client.join()を呼び出して、オーディオビデオ通話ルームに参加します。主なパラメータは次のとおりです。

パラメータ名	フィールドの意味	補足説明	データタイプ	入力例	デフォルト

パラメータ名	フィールドの意味	補足説明	データタイプ	入力例	デフォルト
roomId	ルームナンバー	デフォルトではnumberタイプであり、stringタイプのroomIdを使用したい場合は、createClient()でuseStringRoomIdパラメータをtrueに設定してください - roomIdがnumberタイプの場合、値の要件は[1, 4294967294] の整数となります。 - roomIdがstringタイプの場合、長さは64文字までに制限され、かつ次の範囲の文字セットのみサポートされます。 大文字と小文字のアルファベット（a-zA-Z）、数字（0-9）、スペース、!、#、$、%、&、(、)、+、-、:、;、<、=、.、>、?、@、[、]、^、_、{、}、|、~、，	number / string	3364 または class-room	なし
role	ロール	ユーザーのロールは `live` モードの場合のみ設定が必要です。現在は、 `anchor` （キャスター）、 `audience` （視聴者）という2種類のロールをサポートしています	string	anchor	audience

より詳細なパラメータの説明はClient.join()をご参照ください。

```
// Promise構文の使用
client
.join({ roomId })
.then(() => {
console.log('入室成功');
})
.catch(error => {
console.error('入室に失敗しました。しばらくしてからもう一度お試しください'+ error);
```

```
});
// 同様の効果を得るには、async/await構文の使用をお勧めします
try {
await client.join({ roomId });
console.log('入室成功');
} catch (error) {
console.error('入室に失敗しました。しばらくしてからもう一度お試しください'+ error);
}
// キャスターのロールで入室します
try {
await client.join({
roomId,
role: 'anchor'
});
console.log('入室成功');
} catch (error) {
console.error('入室に失敗しました。しばらくしてからもう一度お試しください'+ error);
}
```

Electron

最終更新日：：2022-07-26 15:46:58

ここでは主に、TRTCルームへの入室方法についてご説明します。オーディオビデオルームに入室すると、ユーザーはルーム内の他のユーザーのオーディオビデオストリーミングをサブスクリプションしたり、ルーム内の他のユーザーに自分のオーディオビデオストリーミングを公開したりすることができます。

呼び出しガイド

ステップ1：SDKのインポート

ドキュメントSDKのプロジェクトへのインポートを参照し、SDKのインポートを完了してください。

ステップ2： SDKインスタンスの作成

```
import TRTCCloud from 'trtc-electron-sdk';
const rtcCloud = new TRTCCloud();
```

ステップ3：SDKイベントの監視

イベントコールバックインターフェースを設定することで、SDKの実行中に発生するエラー情報、アラート情報、トラフィック統計情報、ネットワーク品質情報および各種オーディオビデオイベントを監視することができます。

```
function onError(errCode, errMsg) {
// errorCodeについては、https://cloud.tencent.com/document/product/647/32257#.E9.94.99.E8.AF.AF.E7.A0.81.E8.A1.A8をご参照ください
console.log(errCode, errMsg);
}
function onWarning(warningCode, warningMsg) {
// warningCodeについては、https://cloud.tencent.com/document/product/647/32257#.E8.AD.A6.E5.91.8A.E7.A0.81.E8.A1.A8をご参照ください
console.log(warningCode, warningMsg);
}
rtcCloud.on('onError', onError);
rtcCloud.on('onWarning', onWarning);
```

ステップ4：入室パラメータTRTCParamsの準備

enterRoomインターフェースを呼び出す際には、`TRTCParams`と`TRTCAppScene`という2つのキーパラメータを入力する必要があります。次に詳しくご説明します。

パラメータ1：TRTCAppScene

このパラメータは、お客様のユースケース、すなわち**オンラインライブストリーミング**または**リアルタイム通話**を指定するために使用します。

- **リアルタイム通話**：ビデオ通話用の`TRTCAppSceneVideoCall`と音声通話用の`TRTCAppSceneAudioCall`という2つのオプションがあります。このモードは、1対1のオーディオビデオ通話や参加者300人以内のオンラインミーティングに適しています。
- **オンラインライブストリーミング**：ビデオライブストリーミング用の`TRTCAppSceneLIVE`と音声ライブストリーミング用の`TRTCAppSceneVoiceChatRoom`という2つのオプションがあります。このモードは、最大10万人規模のライブストリーミングシナリオに適していますが、次にご紹介するTRTCParamsパラメータに**ロール(role)**というフィールドを指定する必要があります。これは、ルーム内のユーザーがキャスター(anchor)と視聴者(audience)**という2つのロールに区別されることを意味します。

パラメータ2：TRTCParams

TRTCParamsはたくさんのフィールドから構成されていますが、通常は、以下のフィールドについてのみ入力する必要があります。

パラメータ名	フィールドの意味	補足説明	データタイプ	入力例
SDKAppID	アプリケーションID	TRTCコンソールでこのSDKAppIDを確認できます。確認できない場合は、「アプリケーションの作成」ボタンをクリックして、新しいアプリを作成します。	数字	1400000
userId	ユーザーID	ユーザー名には、大文字と小文字のアルファベット（a-z、A-Z）、数字（0-9）およびアンダースコアとハイフンのみが使用可能です。TRTCは、同じuserIdによる2つの異なるデバイスの同時入室をサポートしていません。同時に入室した場合は相互に干渉します。	文字列	「denny」 「12332
userSig	入室認証証書	SDKAppIDとuserIdを使用してuserSigを算出できます。計算方法については、UserSigの計算、使用方法をご参照ください。	文字列	eJyrVare

パラメータ名	フィールドの意味	補足説明	データタイプ	入力例
roomId	ルームナンバー	数字タイプのルームナンバーです。文字列タイプのルームナンバーを使用したい場合は、strRoomIdとroomIdは混在して使用できないため、roomIdフィールドではなく、**strRoomId**フィールドを使用するようご注意ください。	数字	29834
strRoomId	ルームナンバー	文字列タイプのルームナンバーです。strRoomIdとroomIdは混在して使用できないので、ご注意ください。「123」と123は、TRTCバックエンドサービスでは同じルームになりません。	数字	29834
role	ロール	「キャスター」と「視聴者」という2つのロールがあります。TRTCAppSceneが `TRTCAppSceneLIVE` または `TRTCAppSceneVoiceChatRoom` という2つのライブストリーミングシナリオに指定されている場合のみ、このフィールドを指定する必要があります。	列挙値	TRTCRoはTRTCF

注意：

- TRTCは、同じuserIdによる2つの異なるデバイスの同時入室をサポートしていません。同時に入室した場合は相互に干渉します。
- 各端末のユースケースappSceneについては、統一する必要があります。統一していない場合、想定外のトラブルが生じる恐れがあります。

ステップ5：入室(enterRoom)

ステップ4で2つのパラメータ（TRTCAppSceneとTRTCParams）を準備すると、enterRoomインターフェース関数を呼び出して入室することができます。

```
import { TRTCParams, TRTCRoleType, TRTCAppScene } from 'trtc-electron-sdk';
const param = new TRTCParams();
param.sdkAppId = 1400000123;
param.userId = "denny";
param.roomId = 123321;
param.userSig = "xxx";
param.role = TRTCRoleType.TRTCRoleAnchor;
```

```
// シナリオが「オンラインライブストリーミング」の場合、ユースケースをTRTC_APP_SCENE_LIVEに設定してください
rtcCloud.enterRoom(param, TRTCAppScene.TRTCAppSceneLIVE);
```

イベントコールバック：

- 入室に成功すると、SDKはonEnterRoom(result)イベントをコールバックします。そのうちresultは0以上の値で、入室にかかった時間をミリ秒(ms)単位で表します。
- 入室に失敗した場合、SDKはonEnterRoom(result)イベントをコールバックしますが、パラメータ `result` は負の数となり、その値はルームエントリーに失敗したときのエラーコードとなります。

```
function onEnterRoom(result) {
// onEnterRoomについては、https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/TRTCCallback.html#event:onEnterRoomをご参照ください
if (result > 0) {
console.log('Enter room succeed');
} else {
// 入室エラーコード https://www.tencentcloud.com/document/product/647/35124をご参照ください
console.log('Enter room failed');
}

}
rtcCloud.on('onEnterRoom', onEnterRoom);
```

3. オーディオビデオストリームのサブスクリプション

Android&iOS&Windows&Mac

最終更新日：：2022-07-07 15:10:00

このドキュメントでは、主に、ルーム内の他のユーザーのオーディオビデオストリームのサブスクリプション方法、つまり、他のユーザーのオーディオビデオの再生方法を紹介します。以下のドキュメントでは、便宜上、「ルーム内の他のユーザー」をまとめて「リモートユーザー」と呼びます。

呼び出しガイド

手順1：事前手順の完了

SDKのプロジェクトへのインポートを参照して、SDKのインポートおよびApp権限の設定を完了します。

手順2：サブスクリプションモードの設定（非必須）

TRTCCloudの**setDefaultStreamRecvMode**インターフェースを呼び出して、サブスクリプションモードを設定できます。TRTCは2つのサブスクリプションモードを提供します：

- 自動サブスクリプション：SDKは、ユーザー側で追加の動作を実行することなく、リモートユーザーの音声を自動的に再生します。これは、SDKのデフォルトの動作です。
- 手動サブスクリプション：SDKはリモートユーザーの音声を自動的にプルして再生しません。音声の再生をトリガーするには、手動で**muteRemoteAudio(userId, false)**を呼び出してください。

> 注意：
> SDKのデフォルトの動作は、自動サブスクリプションであるので、setDefaultStreamRecvModeを呼び出さなくても問題ないことに注意してください。ただし、手動サブスクリプションに設定した場合は、setDefaultStreamRecvModeがenterRoomの前に呼び出された場合にのみ有効であることに注意してください。

手順3：TRTCの入室

入室を参照して現在のユーザーが入室します。正常に入室した後のみ、他のユーザーのオーディオビデオストリームをサブスクリプションできます。

手順4：オーディオストリームの再生

インターフェイスmuteRemoteAudio("denny"，true)を呼び出すことにより、リモートユーザーdennyの音声をミュートできます。次に、インターフェイスmuteRemoteAudio("denny"，false)を呼び出すことにより、リモートユーザーdennyのミュートを解除できます。

- Android Java
- iOS&Mac ObjC
- Windows C++

```
// Mute user with id denny
mCloud.muteRemoteAudio("denny", true);
// Unmute user with id denny
mCloud.muteRemoteAudio("denny", false);
```

手順5：ビデオストリームの再生

1. 再生の開始と停止（startRemoteView + stopRemoteView）

インターフェースstartRemoteViewを呼び出すことにより、リモートユーザーのビデオ画面を再生できますが、ユーザーのビデオ画面を運ぶためのレンダリングコントロールとして使用されるビューオブジェクトをSDKに渡す必要があることが前提です。

startRemoteViewの最初のパラメータはリモートユーザーのuserId、2番目のパラメータはリモートユーザーのストリームタイプ、3番目のパラメータは渡す必要のあるviewオブジェクトです。2番目のパラメータstreamType（ストリームタイプ）には、次の3つのオプション値があります。

- TRTCVideoStreamTypeBig：ユーザーのビッグストリーム画面です。通常、ユーザーのカメラ画面を送信するために使用されます。
- TRTCVideoStreamTypeSub：ユーザーのサブストリーム画面です。通常、ユーザーの画面共有の画面を送信するために使用されます。
- TRTCVideoStreamTypeSmall：ビッグストリーム画像を基準にしたユーザーの低解像度の小さな画面です。リモートユーザーが「デュアルチャネルエンコーディング（enableEncSmallVideoStream）」を有効にした場合にのみ、ユーザーの低解像度画面を再生できます。また、同時に、ビッグストリーム画面と低解像度の小さな画面の両方から1つのみ選択できます。

stopRemoteViewインターフェースを呼び出すことにより、1つのリモートユーザーのビデオの再生を停止できます。また、stopAllRemoteViewインターフェースを介してすべてのリモートユーザーのビデオの再生を停止できます。

- Android java
- iOS&Mac ObjC
- Windows C++

```
// dennyのカメラ画面（「ビッグストリーム」と呼ばれる）を再生します
mCloud.startRemoteView("denny", TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, cameraVi
ew);
// dennyの画面共有画面（「サブストリーム」と呼ばれる）を再生します
mCloud.startRemoteView("denny", TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_SUB, screenVi
ew);
// 低解像度画面（ビッグストリームと低解像度の両方から1つのみ選択できる）を再生します
mCloud.startRemoteView("denny", TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_SMALL, camera
View);
// dennyのカメラ画面の再生を停止します
mCloud.stopRemoteView("denny", TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, cameraVie
w);
// すべてのビデオ画面の再生を停止します
mCloud.stopAllRemoteView();
```

2. 再生パラメータの設定（updateRemoteView + setRemoteRenderParams）

updateRemoteViewインターフェースを呼び出すことにより、再生中にviewオブジェクトを変更できます。これは、ビデオレンダリングコントロールを切り替えるときに役立ちます。
setRemoteRenderParamsを使用すると、画面の塗りつぶしモード、回転角度、イメージモードを設定できます。。

- 塗りつぶしモード：塗りつぶしまたは適応に分かれます。画面は2つのモードで元のアスペクト比を維持できます。その違いは、黒い境界線があるかどうかにあります。
- 回転角度：0度、90度、180度および270度など、4つの回転角度に設定できます。
- イメージモード：つまり、画面の左右のイメージモードです。

- Android java
- iOS&Mac ObjC
- Windows C++

```
// dennyのビッグストリーム画面を小さなフローティングウィンドウに切り替えます（ミニウィンドウがm
iniFloatingViewである場合）
```

```
mCloud.updateRemoteView("denny", TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, miniFloatingView);

// リモートユーザーdennyのビッグストリーム画面を塗りつぶしモードに設定し、左右のイメージモードを有効にします
TRTCCloudDef.TRTCRenderParams param = new TRTCCloudDef.TRTCRenderParams();
param.fillMode = TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL;
param.mirrorType = TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_DISABLE;
mCloud.setRemoteRenderParams("denny", TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, param);
```

手順6：ルーム内のリモートユーザーのオーディオビデオ状態の検知

手順4と手順5では、リモートユーザーへのオーディオビデオの再生を制御できますが、十分な情報がないと以下を知ることができません：

- 現在のルームのユーザーは誰ですか？
- カメラとマイクがオンになっているかどうか？

この問題を解決するために、SDKからのいくつかのイベントコールバックを監視してください：

オーディオ状態変化通知（onUserAudioAvailable）

リモートユーザーがマイクをオンまたはオフにしたときに、onUserAudioAvailable(userId，boolean)を監視することにより、この状態の変化を検知できます。

ビデオ状態変化通知（onUserVideoAvailable）

リモートユーザーがビデオ画面をオンまたはオフにしたときに、onUserVideoAvailable(userId，boolean)を監視することにより、この状態の変化を検知できます。

リモートユーザーが画面共有画面をオンまたはオフにしたときに、onUserSubStreamAvailable(userId，boolean)を監視することにより、この状態の変化を検知できます。

ユーザーの入退室の通知（onRemoteUserEnter/LeaveRoom）

リモートユーザーが現在のルームに入るとき、onRemoteUserEnterRoom(userId)を介してユーザーのuserIdを検知できます。リモートユーザーが現在のモードを離れるとき、onRemoteUserLeaveRoom(userId, reason)を介してこのユーザーのuserIdと離れる理由を検知できます。

> 注意：
> 正確にいえば、onRemoteUserEnter/LeaveRoomは、キャスターをロール（role）とするユーザーの入退室通知のみを検知できます。このように設計する理由は、ルーム内のオンライン視聴者が多い場合に、頻繁に入退室する人がいるため、ルーム内のすべてのユーザーが他のユーザーの「シグナルストーム」によって攻撃されることを回避します。

これらのイベントコールバックを使用すると、ルームにいるユーザーと、カメラとマイクをオンにしたかどうかを知ることができます。次のサンプルコードを参照してください。このサンプルコードでは、mCameraUserList、mMicrophoneUserList、およびmUserListを使用してそれぞれ以下を個別に管理します。

- ルーム内のユーザー（正確にいえば、キャスターである）は誰ですか。
- カメラをオンにしたユーザーは誰ですか
- マイクをオンにしたユーザーは誰ですか

- Android java
- iOS&Mac ObjC
- Windows C++

```
// リモートユーザーのビデオ状態の変化を検知し、カメラをオンにしたユーザーリストを更新します（mCameraUserList）
@Override
public void onUserVideoAvailable(String userId, boolean available) {
available ? mCameraUserList.add(userId) : mCameraUserList.remove(userId);
}

// リモートユーザーのオーディオ状態の変化を検知し、マイクをオンにしたユーザーリストを更新します（mMicrophoneUserList）
@Override
public void onUserAudioAvailable(String userId, boolean available) {
available ? mMicrophoneUserList.add(userId) : mMicrophoneUserList.remove(userId);
}

// リモートユーザーの入室通知を感知し、リモートユーザーリストを更新します（mUserList）
@Override
public void onRemoteUserEnterRoom(String userId) {
mUserList.add(userId) ;
}

// リモートユーザーの退室通知を感知し、リモートユーザーリストを更新します（mUserList）
@Override
public void onRemoteUserLeaveRoom(String userId, int reason) {
mUserList.remove(userId) ;
}
```

拡張機能ガイド

1. 同じ「ミュート」ですが、その違いは何ですか？

サービスニーズの成長につれて、3種類の「ミュート」があり、それらはすべて「ミュート」と呼ばれますが、カウントの原則は完全に異なります。

- **1番目：再生側はオーディオストリームのサブスクリプションを停止します**
  muteRemoteAudio("denny", true)関数を呼び出す場合、リモートユーザーdennyの音声を再生したくないことを意味し、このとき、SDKはdennyのオーディオデータストリームのプルを停止します。このモードでは、より多くのトラフィックを節約します。そのとき、dennyの音声をもう一度再生したい場合、SDKはオーディオデータのプルプロセスを再開する必要があるため、「ミュート」から「ミュート解除」への回復速度が遅くなります。

- **2番目：再生ボリュームをゼロに調整します**
  サービスシーンでミュート切り替えの応答時間を短縮する必要がある場合、setRemoteAudioVolume("denny", 0)を使用して、リモートユーザーdennyの再生ボリュームをゼロに設定できます。このインターフェースにはネットワーク操作が含まれないため、応答速度が非常に速くなります。

- **3番目：リモートユーザーが自分でマイクをオフにします**
  このドキュメントで説明されているすべての操作は、再生側の操作に対するものです。これらの操作の効果は、現在のユーザーにのみ有効です。たとえば、muteRemoteAudio("denny", true)を使用して、リモートユーザーdennyをミュートできますが、ルーム内の他のユーザーは依然として、dennyの音声を聞くことができます。
  Dennyを完全に「シャットダウン」する場合、dennyのオーディオリリース動作に影響を与える必要があります。これについては、次のドキュメントオーディオビデオストリームのリリースで詳しく紹介します。

Web

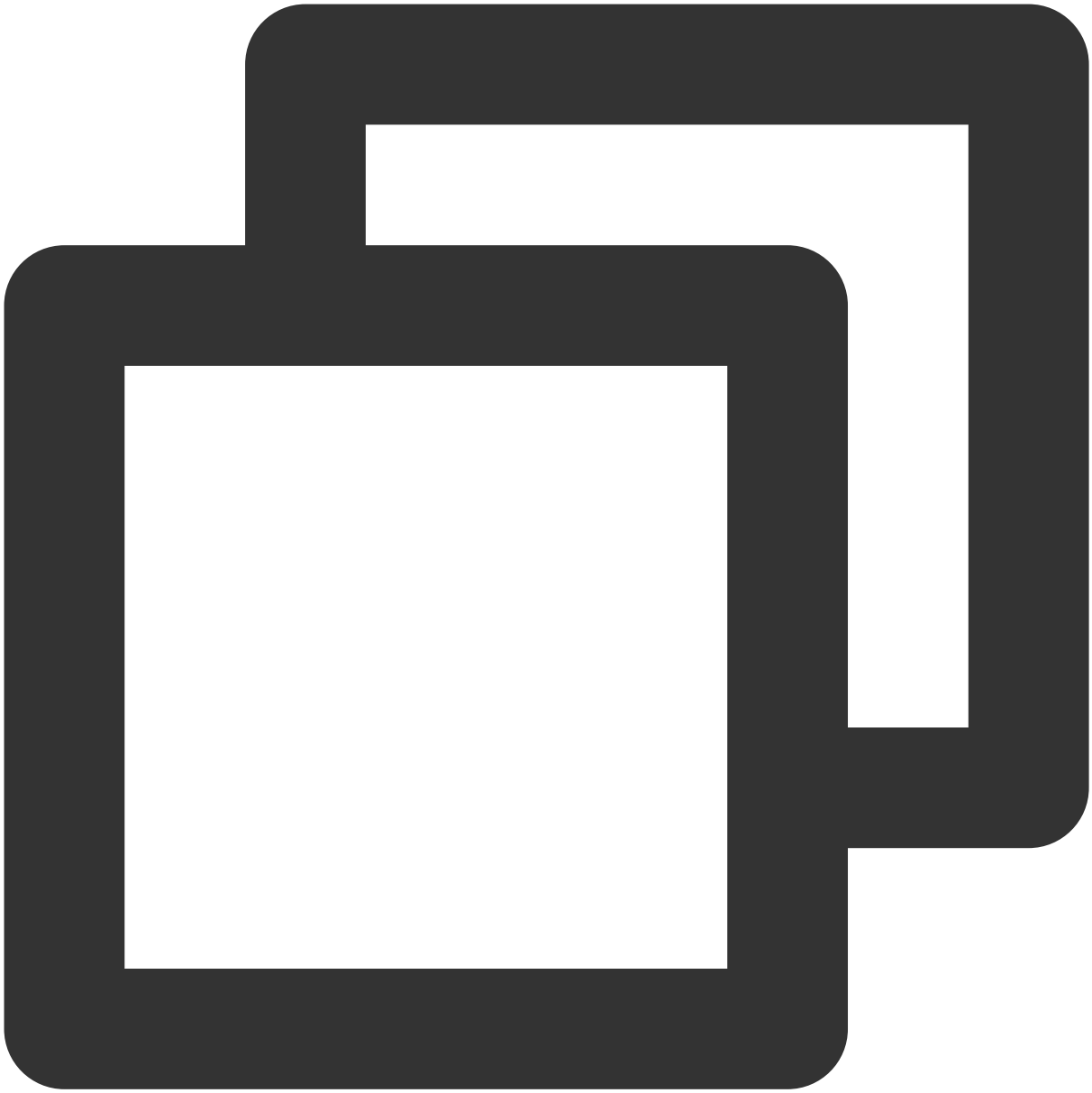
最終更新日：：2023-02-20 16:03:40

このドキュメントでは、主に、ルーム内の他のユーザーのオーディオビデオストリームのサブスクリプション方法、つまり、他のユーザーのオーディオビデオの再生方法を紹介します。以下のドキュメントでは、便宜上、「ルーム内の他のユーザー」をまとめて「リモートユーザー」と呼びます。

TRTC Web SDKの使用中には、以下のオブジェクトが頻繁に登場します。
Client オブジェクト。ローカルクライアントを表します。Clientクラスのメソッドにより、通話ルームへの入室、ローカルストリーミングの公開、リモートストリームの閲覧などの機能を提供します。
Streamオブジェクト。オーディオビデオストリーミングオブジェクトを表し、ローカルのオーディオビデオストリーミングオブジェクトLocalStreamおよびリモートのオーディオビデオストリーミングオブジェクトRemoteStreamが含まれます。Streamクラスのメソッドでは主に、オーディオビデオストリーミングオブジェクトのアクションを提供し、これにはオーディオおよびビデオの再生コントロールが含まれます。

ステップ1： Clientオブジェクトの新規作成

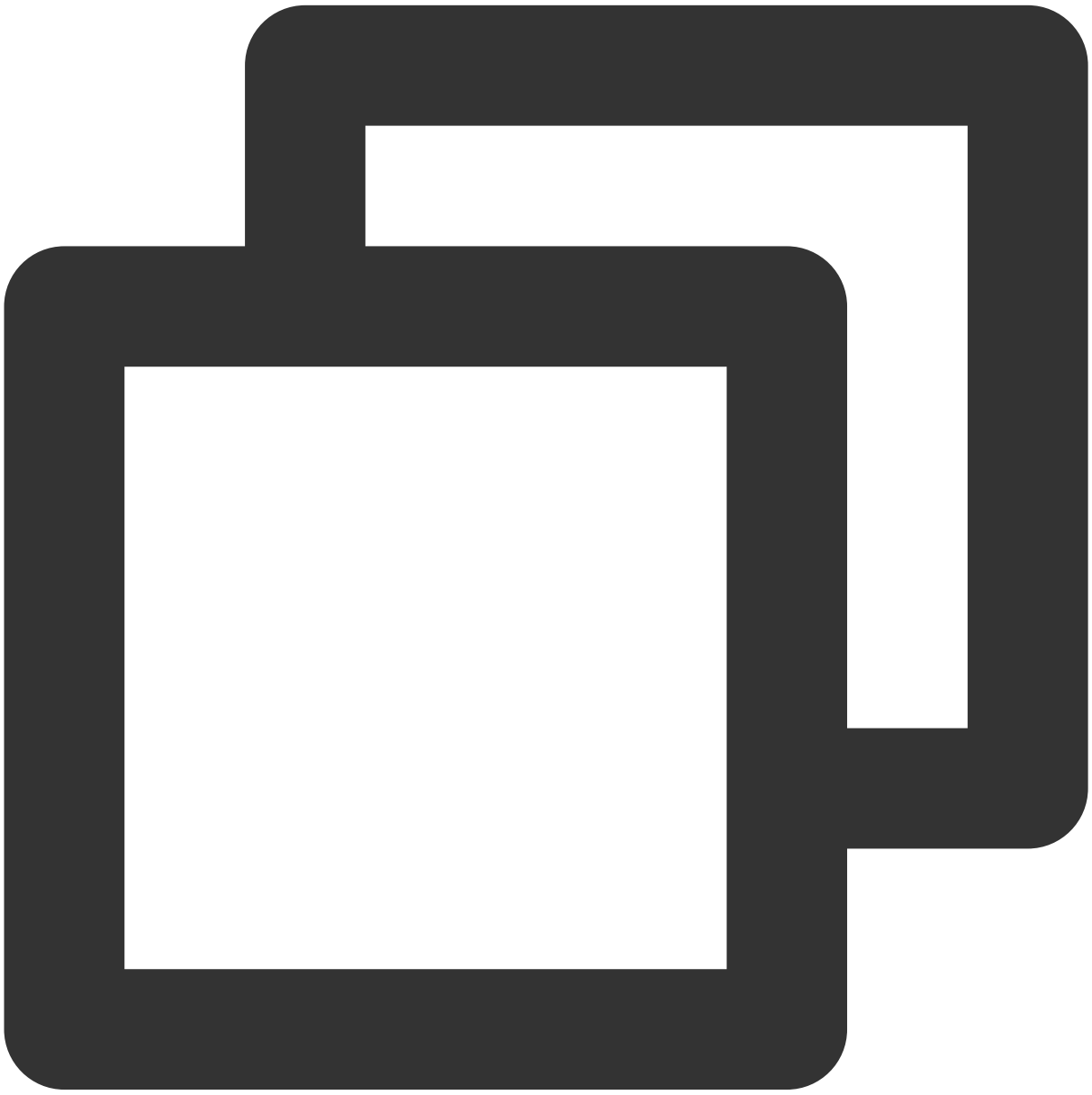
ドキュメント入室-ステップ1を参照し、clientを作成します。
Clientの作成時にサブスクリプションモードの設定を選択できることにご注意ください。TRTCは、2つのサブスクリプションモードを提供しています。
自動サブスクリプションです。stream-addedイベントを受信すると、SDKがすぐにこのリモートストリームに含まれるオーディオビデオデータを受け取ってデコードします。これはSDKのデフォルトのアクションです。
手動サブスクリプション。画面共有を担うclientはプッシュのみを必要とし、プルは必要ないため、画面共有中のclientは自動サブスクリプションを無効にすることができます。



```
const client = TRTC.createClient({
  ...,
 autoSubscribe: false // デフォルトではtrue、すなわち自動サブスクリプションです
});
```

ステップ2：リモートストリーム参加イベントの監視およびリモートストリームの閲覧

リモートストリームのサブスクリプションには、まずどのリモートストリームがサブスクリプション可能かを知る必要があります。イベントClient.on('stream-added')の監視によってルーム内のリモートストリームを取得できます。このイベントを受信すれば、そのリモートストリームはサブスクリプション可能ということになり、イベントコールバックでClient.subscribe()によってリモートオーディオビデオストリーミングをサブスクリプションすることができます。



```
client.on('stream-added', event => {
  const remoteStream = event.stream;
  console.log('リモートストリームの増加: ' + remoteStream.getId());
  //リモートストリームのサブスクリプション
```

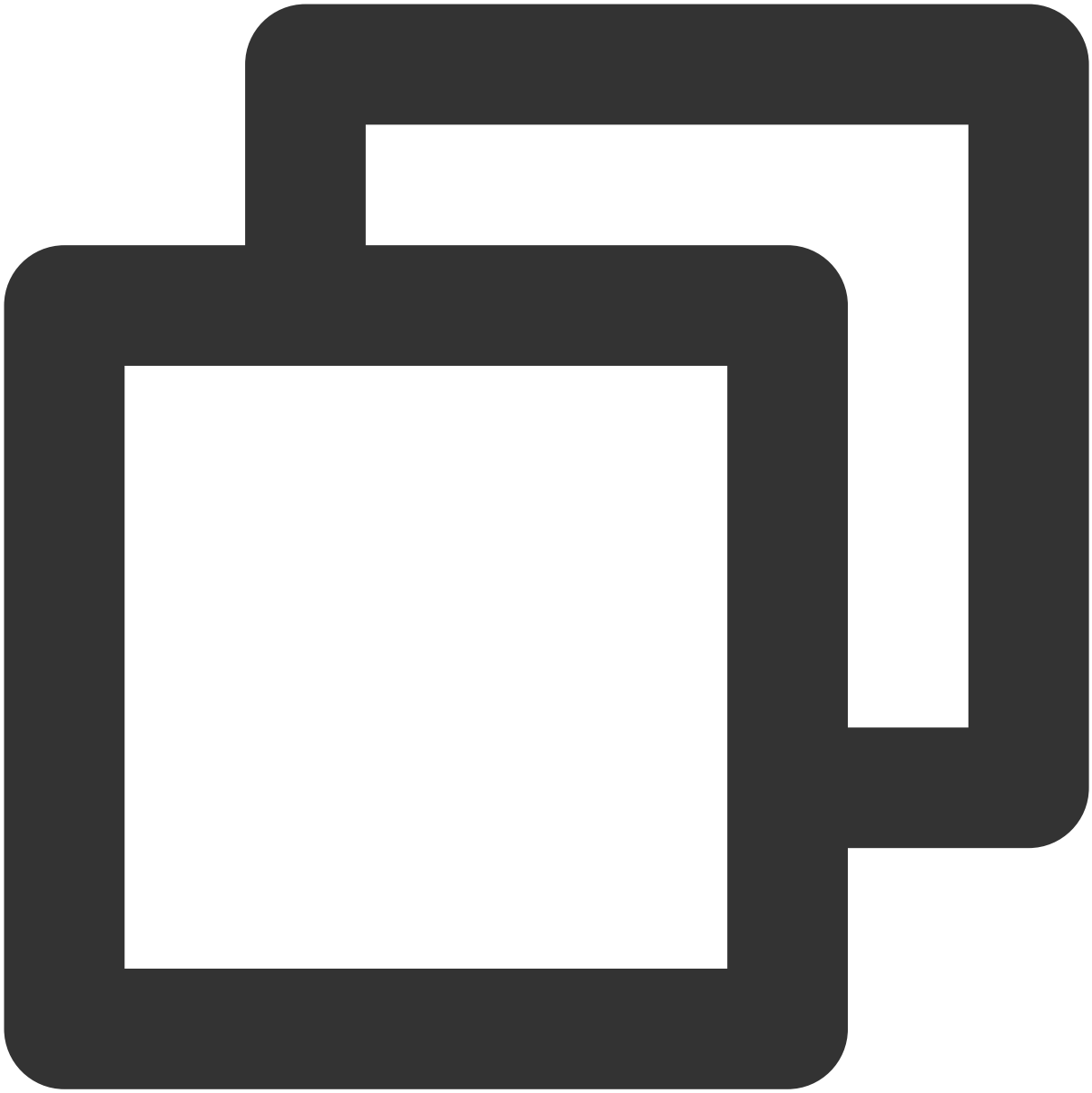
```
    client.subscribe(remoteStream);
  });
```

注意

すでにルーム内にいるユーザーからのリモートストリーム通知を確実に受け取れるよう、入室前にClient.on('stream-added')イベントを監視します。
リモートストリームから退出するなどその他のイベントは、API詳細ドキュメントで確認できます。

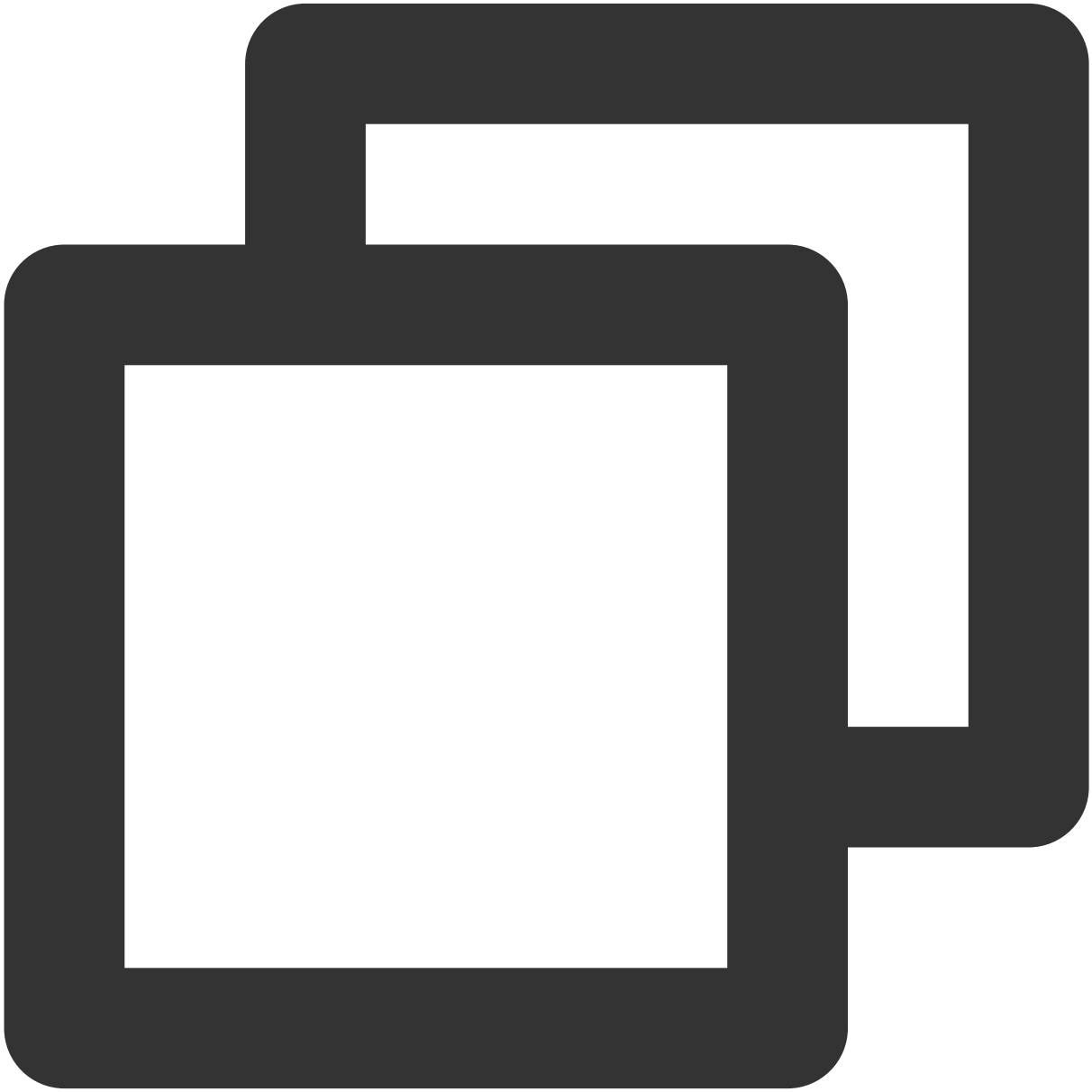
ステップ3：サブスクリプション成功イベントの監視とリモートストリームの再生

リモートストリームの成功イベントのコールバックでは、Stream.play()メソッドを呼び出すことで、Webページ上でオーディオビデオを再生します。 `play` メソッドがdivエレメントのIDまたはHTMLDivElementオブジェクトをパラメータとして受け入れると、SDKがそのdivエレメント下に対応するオーディオビデオタグを自動作成し、オーディオビデオを再生します。
`play` メソッドのより詳細なパラメータ説明については、Stream.play()をご参照ください。



```
client.on('stream-subscribed', event => {
  const remoteStream = event.stream;
  console.log('リモートストリームのサブスクリプション成功: ' + remoteStream.getId());
  // 再生コンテナの作成
  let remotePlayerElement = document.createElement('div');
  remotePlayerElement.id = 'remote-stream-' + remoteStream.getId();
  document.body.appendChild(remotePlayerElement);
  // リモートストリームの再生を開始する場合、playメソッドに渡されるElement IDは、画面に存在するd
  remoteStream.play(remotePlayerElement.id);
});
```

ブラウザによる自動再生ポリシーの制限の影響により、 `play` メソッドを呼び出すとPLAY_NOT_ALLOWEDのエラーが返される場合があることに特にご注意ください。このときSDKはポップアップウィンドウを表示して、ユーザーとページのインタラクションを促します。インタラクションが発生すると、SDKは自動的にインターフェースを呼び出して再生を再開します。
TRTC.createClient()インターフェースでenableAutoPlayDialogパラメータをfalseに設定することで、SDKのポップアップウィンドウ機能を無効にし、さらにユーザーがクリックなどの操作でStream.resume()を呼び出し、オーディオビデオ再生を再開できるように実装することができます。



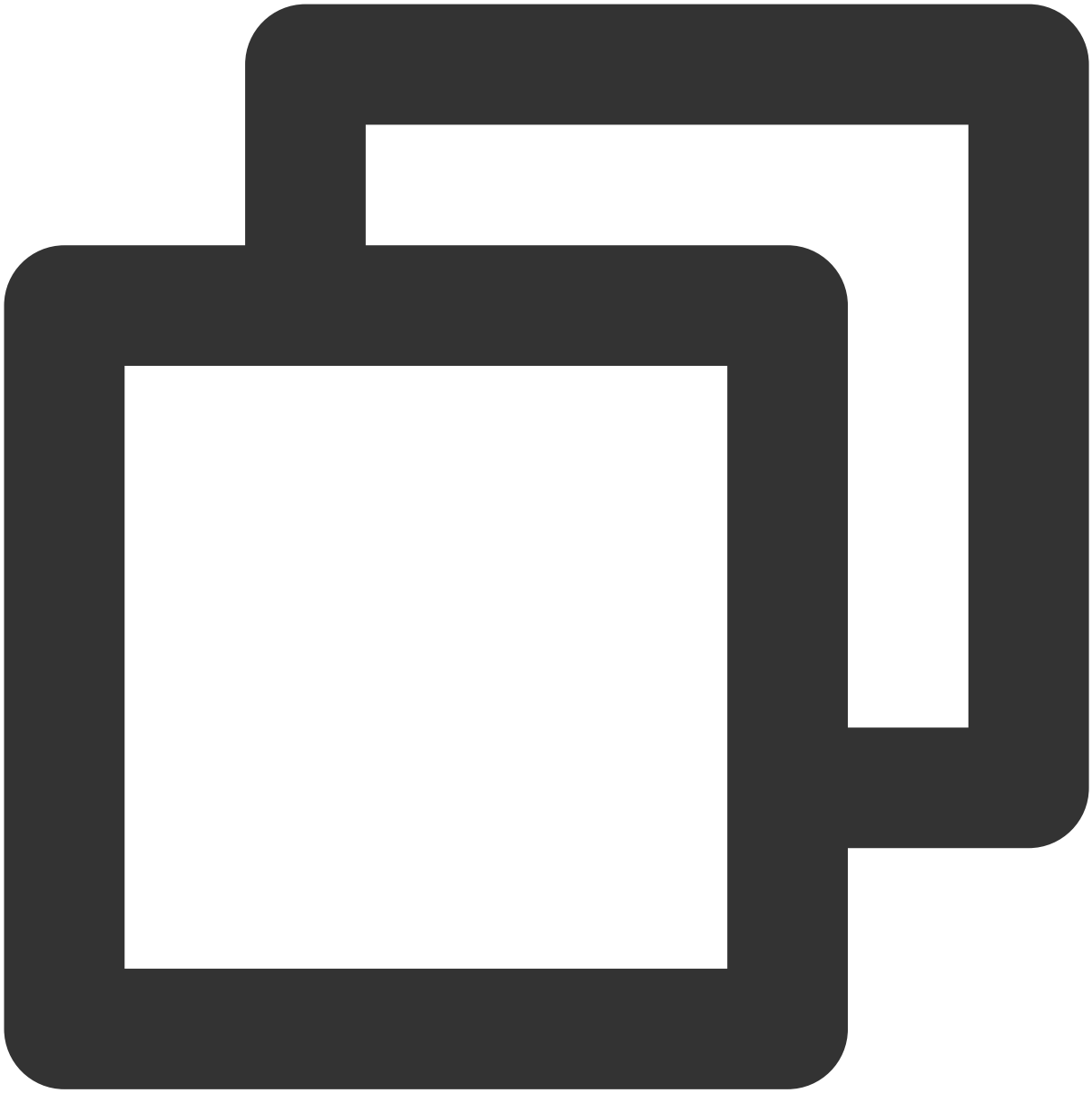
```
client.on('stream-subscribed', event => {
  const remoteStream = event.stream;
```

```
  console.log('リモートストリームのサブスクリプション成功: ' + remoteStream.getId());
  // remoteStreamを使用してerrorを監視する方法で、0x4043エラーをキャッチして処理します
  remoteStream.on('error', error => {
    const errorCode = error.getCode();
    if (errorCode === 0x4043) {
      // PLAY_NOT_ALLOWED、ジェスチャー操作でstream.resumeを呼び出してオーディオビデオ再生を
      // remoteStream.resume()
    }
  });
  // 再生コンテナの作成
  let remotePlayerElement = document.createElement('div');
  remotePlayerElement.id = 'remote-stream-' + remoteStream.getId();
  document.body.appendChild(remotePlayerElement);
  // リモートストリームの再生を開始する場合、playメソッドに渡されるElement IDは、画面に存在するd
  remoteStream.play(remotePlayerElement.id);
});
```

ステップ4：オーディオビデオ通話ルームへの参加

イベントの監視後、Client.join()を呼び出してオーディオビデオ通話ルームに入室できます。ドキュメント入室-ステップ2を参照できます。

完全なコード



```
const client = TRTC.createClient({
  mode: 'rtc',
  sdkAppId,
  userId,
  userSig
});

client.on('stream-added', event => {
  const remoteStream = event.stream;
  console.log('リモートストリームの増加: ' + remoteStream.getId());
```

```
  //リモートストリームのサブスクリプション
  client.subscribe(remoteStream);
});

client.on('stream-subscribed', event => {
  const remoteStream = event.stream;
  console.log('リモートストリームのサブスクリプション成功: ' + remoteStream.getId());
  // remoteStreamを使用してerrorを監視する方法で、0x4043エラーをキャッチして処理します
  remoteStream.on('error', error => {
    const errorCode = error.getCode();
    if (errorCode === 0x4043) {
      // PLAY_NOT_ALLOWED、ジェスチャー操作でstream.resumeを呼び出してオーディオビデオ再生を
      // remoteStream.resume()
    }
  });
  // 再生コンテナの作成
  let remotePlayerElement = document.createElement('div');
  remotePlayerElement.id = 'remote-stream-' + remoteStream.getId();
  document.body.appendChild(remotePlayerElement);
  // リモートストリームの再生を開始する場合、playメソッドに渡されるElement IDは、画面に存在するd
  remoteStream.play(remotePlayerElement.id);
});

try {
  await client.join({ roomId });
console.log('入室成功');
} catch (error) {
  console.error('入室に失敗しました。しばらくしてからもう一度お試しください'+ error);
}
```

Electron

最終更新日：：2022-07-26 15:46:58

ここでは主に、ルーム内の他のユーザーのオーディオビデオストリーミングをサブスクリプションする方法、すなわち他のユーザーのオーディオとビデオを再生する方法についてご説明します。便宜上、これ以降は「ルーム内にいる他のユーザー」を「リモートユーザー」と総称します。

呼び出しガイド

ステップ1：前のステップの完了

ドキュメントSDKのプロジェクトへのインポートを参照し、SDKのインポートを完了してください。

ステップ2：サブスクリプションモードの設定（必須ではありません）

TRTCCloudの**setDefaultStreamRecvMode**インターフェースを呼び出すことでサブスクリプションモードを設定することができます。TRTCは、2つのサブスクリプションモードを提供しています。

- 自動サブスクリプション：SDKは、お客様側で追加の操作をしなくても、自動的にリモートユーザーの音声を再生します。これはSDKのデフォルトの動作です。
- 手動サブスクリプション：SDKは、自動的にリモートユーザーの音声をプルして再生しないため、手動で**muteRemoteAudio(userId, false)**を呼び出して音声を再生させる必要があります。

> 注意：
> setDefaultStreamRecvModeを呼び出さなくても、SDKのデフォルトの動作は自動サブスクリプションである点にご注意ください。ただし、手動サブスクリプションに設定したい場合は、**setDefaultStreamRecvModeはenterRoomの前に呼び出した場合にのみ有効であることにご注意ください。**

ステップ3：TRTCルームへの入室

ドキュメント入室をご参照のうえ、現在のユーザーをTRTCルームに入室させます。入室に成功した場合にのみ、他のユーザーのオーディオビデオストリーミングをサブスクリプションすることができます。

ステップ4：オーディオストリームの再生

インターフェースmuteRemoteAudio("denny"，true)を呼び出すことでリモートユーザーdennyの音声をミュートでき、その後、インターフェースmuteRemoteAudio('denny'，false)を呼び出すことでミュートを解除できます。

```
import TRTCCloud from 'trtc-electron-sdk';
const rtcCloud = new TRTCCloud();
// Reference https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/TRTCCloud.html#muteRemoteAudio
// Mute user with id denny
rtcCloud.muteRemoteAudio('denny', true);
// Unmute user with id denny
rtcCloud.muteRemoteAudio('denny', false);
```

ステップ5：ビデオストリームの再生

1. 再生の開始と停止(startRemoteView + stopRemoteView)

インターフェースstartRemoteViewを呼び出すことで、リモートユーザーのビデオ画面を再生することができます。ただし、ユーザーのビデオ画面のレンダリングウィジェットとして使用するviewオブジェクトをSDKに渡した場合に限ります。

startRemoteViewの最初のパラメータはリモートユーザーのuserId、2番目のパラメータはリモートユーザーのストリームタイプ、3番目のパラメータは渡す必要のあるviewオブジェクトです。そのうち2番目のパラメータstreamType（ストリームタイプ）には、次の3つのオプション値があります。

- **TRTCVideoStreamTypeBig**：ユーザーのメインチャネル画面です。通常、ユーザーのカメラ画面を送信するために使用します。
- **TRTCVideoStreamTypeSub**：ユーザーのサブチャネル画面です。通常、ユーザー画面と共有している画面を送信するために使用します。
- **TRTCVideoStreamTypeSmall**：ユーザーの低解像度小画面です。これは、メインチャネルと相対するもので、リモートユーザーが「デュアルチャネルエンコーディング(enableEncSmallVideoStream)」を有効にした場合にのみ、そのユーザーの低解像度画面を再生することができるようになります。また、メインチャネル画面と低解像度小画面のうち、どちらか一方しか選択できません。

stopRemoteViewインターフェースを呼び出すと、特定のリモートユーザーのビデオ再生を停止することができます。また、stopAllRemoteViewインターフェースを呼び出すと、すべてのリモートユーザーのビデオ再生を停止することもできます。

```
// https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/TRTCCloud.html#startRemoteViewをご参照ください
import TRTCCloud, { TRTCVideoStreamType } from 'trtc-electron-sdk';
const cameraView = document.querySelector('.user-dom');
```

```
const screenView = document.querySelector('.screen-dom');
const rtcCloud = new TRTCCloud();
// dennyのカメラ画面の再生（「メインチャネル画面」と呼びます）
rtcCloud.startRemoteView('denny', cameraView, TRTCVideoStreamType.TRTCVideoStream
TypeBig);
// dennyの画面共有画面の再生（「サブチャネル画面」と呼びます）
rtcCloud.startRemoteView('denny', screenView, TRTCVideoStreamType.TRTCVideoStream
TypeSub);
// dennyの低解像度画面の再生（メインチャネル画面と低解像度画面のうちどちらか一方のみ選択可能）
rtcCloud.startRemoteView('denny', cameraView, TRTCVideoStreamType.TRTCVideoStream
TypeSmall);
// dennyのカメラ画面の再生停止
rtcCloud.stopRemoteView('denny', TRTCVideoStreamType.TRTCVideoStreamTypeBig);
// すべてのビデオ画面の再生停止
rtcCloud.stopAllRemoteView();
```

2. 再生パラメータの設定（setRemoteRenderParams）

setRemoteRenderParamsによって、画面のフィルモード、回転角度およびイメージモードを設定することができます。

- フィルモード：フィルとフィットがあり、どちらも画面の元のアスペクト比を維持できます。違いは黒い縁取りの有無です。
- 回転角度：0度、90度、180度および270度など、4つの回転角度が設定できます。
- イメージモード：画面の左右イメージモードのことです。

```
// https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/TRTCClou
d.html#setRemoteRenderParamsをご参照ください
// リモートユーザーdennyのメインチャネル画面をフィルモードに設定し、左右のイメージモードを有効にし
ます
import TRTCCloud, {
TRTCRenderParams, TRTCVideoStreamType, TRTCVideoRotation,
TRTCVideoFillMode, TRTCVideoMirrorType
} from 'trtc-electron-sdk';
const param = new TRTCRenderParams(
TRTCVideoRotation.TRTCVideoRotation0,
TRTCVideoFillMode.TRTCVideoFillMode_Fill,
TRTCVideoMirrorType.TRTCVideoMirrorType_Enable
);
const rtcCloud = new TRTCCloud();
rtcCloud.setRemoteRenderParams('denny', TRTCVideoStreamType.TRTCVideoStreamTypeBi
g, param);
```

ステップ6：ルーム内にいるリモートユーザーのオーディオビデオステータスの感知

ステップ4とステップ5では、リモートユーザーに対して音声やビデオの再生を制御することができますが、十分な情報がないと以下の事項がわからなくなります。

- 現在、ルームにいるユーザーは誰か。
- ユーザーたちはカメラやマイクをオンにしているか。

この問題を解決するには、SDKからのイベントコールバックのいくつかを監視する必要があります。

- **オーディオステータス変更通知（onUserAudioAvailable）**
  リモートユーザーがマイクをオンやオフした場合、onUserAudioAvailable(userId，boolean)を監視することで、このステータスの変化を感知することができます。

- **ビデオステータス変更通知（onUserVideoAvailable）**
  リモートユーザーがカメラをオンやオフにした場合、onUserVideoAvailable(userId，boolean)を監視することで、このステータスの変化を感知することができます。
  リモートユーザーが画面共有画面をオンやオフにした場合、onUserVideoAvailable(userId，boolean)を監視することで、このステータスの変化を感知することができます。

- **ユーザーの入退室通知（onRemoteUserEnter/LeaveRoom）**
  リモートユーザーが現在のルームに入るとき、onRemoteUserEnterRoom(userId)によってそのユーザーのuserIdを感知することができます。リモートユーザーが現在のルームから退出するとき、onRemoteUserLeaveRoom(userId, reason)によってそのユーザーのuserIdとそのユーザーの退出した理由を感知することができます。

> 注意：
> 正確に言うと、onRemoteUserEnter/LeaveRoomは、ロール(role)はキャスター(anchor)のユーザーの入退室通知のみを感知することができます。このような設計になっているのは、多数の視聴者(audience)がオンラインになっているルームでは人の出入りが頻繁であるため、ルームにいるすべてのユーザーが他のユーザーの入退出によって、「シグナリングストーム」攻撃を受けるのを避けるためです。

これらのイベントコールバックにより、どのユーザーがルームにいるか、彼らのカメラとマイクがオンになっているかどうかを把握することができます。次のサンプルコードをご参照ください。このサンプルコードでは、mCameraUserList、mMicrophoneUserListおよびmUserListを使用して次の事項を管理しています。

- ルーム内にいるユーザー（正確にはキャスター）は誰なのか
- そのうちカメラがオンになっているのはどのユーザーか
- そのうちマイクがオンになっているのはどのユーザーか

```
import TRTCCloud from 'trtc-electron-sdk';
let openCameraUserList = [];
let openMicUserList = [];
let roomUserList = [];
function onUserVideoAvailable(userId, available) {
if (available === 1) {
openCameraUserList.push(userId);
} else {
openCameraUserList = openCameraUserList.filter((item) => item !== userId);
}
}
function onUserAudioAvailable(userId, available) {
if (available === 1) {
openMicUserList.push(userId);
} else {
openMicUserList = openMicUserList.filter((item) => item !== userId);
}
}
function onRemoteUserEnterRoom(userId) {
roomUserList.push(userId);
}
function onRemoteUserLeaveRoom(userId, reason) {
roomUserList = roomUserList.filter((item) => item !== userId);
}
const rtcCloud = new TRTCCloud();
rtcCloud.on('onUserVideoAvailable', onUserVideoAvailable);
rtcCloud.on('onUserAudioAvailable', onUserAudioAvailable);
rtcCloud.on('onRemoteUserEnterRoom', onRemoteUserEnterRoom);
rtcCloud.on('onRemoteUserLeaveRoom', onRemoteUserLeaveRoom);
```

アドバンスガイドライン

同じ「ミュート」ですが、違いは何ですか。

ビジネスニーズを深掘りしていくに従い、「ミュート」には3つのタイプがあることがわかると思います。
「ミュート」という名称は同じでも、技術的な原理はまったく異なります。

- **1つめ：再生側のオーディオストリームのサブスクリプションを停止する**
  muteRemoteAudio("denny", true)関数を呼び出すと、リモートユーザーdennyの音声をもう再生したくないという意味になり、SDKはdennyのオーディオデータストリームのプルを停止します。このモードはトラフィックの節約になります。ただし、再度dennyの音声を再生したい場合は、SDKがオーディオデータをプルするプロセスを再起動する必要があるため、「ミュート」状態から「アンミュート」状態への復帰速度が遅くなります。

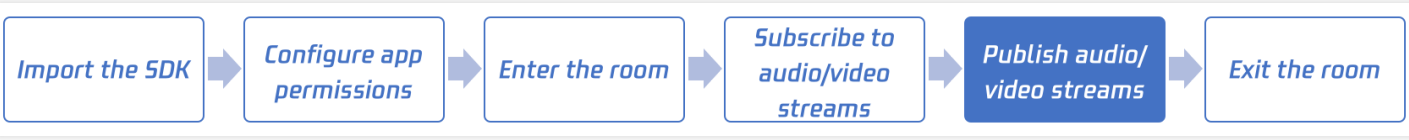
- **2つめ：再生音量をゼロに調整する**
  ビジネスシナリオでミュートの切り替えに高速なレスポンス速度が求められる場合、setRemoteAudioVolume("denny", 0)によってリモートユーザーdennyの再生音量をゼロに設定することができます。このインターフェースはネットワーク操作を伴わないため、影響速度は非常に速くなります。

- **3つめ：リモートユーザーが自分でマイクをオフにする**
  ここでご説明する操作はすべて再生側の操作で、これらの操作によって生じる効果は現在のユーザーにのみ有効です。例えば、muteRemoteAudio("denny", true)によってリモートユーザーdennyをミュートに設定しても、ルーム内の他のユーザーにはdennyの音声が聞こえます。
  dennyを完全にミュートにするには、dennyのオーディオ公開の動作に影響を与える必要があります。これについては、次のドキュメントオーディオビデオストリーミングの公開で詳しくご説明しています。

4. オーディオビデオストリームのリリース Android&iOS&Windows&Mac

最終更新日：：2022-07-12 17:02:05

このドキュメントでは、主にキャスターが自分のオーディオビデオストリームをリリースする方法を紹介します。いわゆる「リリース」とは、マイクとカメラをオンにして、ルーム内の他のユーザーが自分の音声とビデオを聞いたり見たりできるようにすることを意味します。



呼び出しガイド

手順1：事前手順の完了

SDKのプロジェクトへのインポートを参照して、SDKのインポートおよびApp権限の設定を完了します。

手順2：カメラプレビューをオンにする

startLocalPreviewインターフェースを呼び出して、カメラプレビューをオンにすることができます。このとき、SDKはシステムにカメラの使用権限を申請し、カメラキャプチャプロセスはユーザーの承認が渡された後にのみ開始されます。

ローカル画面のレンダリングパラメータを設定する場合は、**setLocalRenderParams**インターフェースを呼び出してローカルプレビューのレンダリングパラメータを設定できます。最初にプレビューをオンにしてからプレビューパラメータを設定すると画面がジッターするのを防ぐために、プレビューパラメータを設定する必要がある場合は、プレビューを開始する前に呼び出すことをお勧めします。

カメラのさまざまな制御パラメータを制御する場合は、**TXDeviceManager**インターフェースを呼び出して、「フロントカメラとリアカメラの切り替え」、「フォーカスモードの設定」、「フラッシュのオン・オフの切り替え」などの一連の操作を実行できます。

美顔効果や画質調節をご希望の場合は、画質設定で詳しく紹介します。

- Android java
- iOS ObjC
- Mac ObjC
- Windows C++

```
// ローカル画面のプレビューモードを設定します。左右のイメージをオンにし、画面を塗りつぶしモードに設定します
TRTCCloudDef.TRTCRenderParams param = new TRTCCloudDef.TRTCRenderParams();
param.fillMode = TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL;
param.mirrorType = TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_AUTO;
mCloud.setLocalRenderParams(param);

// ローカルカメラのプレビューを開始します（localCameraVideoはローカルレンダリング画面をレンダリングするためのコントロールです）
TXCloudVideoView cameraVideo = findViewById(R.id.txcvv_main_local);
mCloud.startLocalPreview(true, cameraVideo);

// TXDeviceManagerを介して自動フォーカスをオンにし、フラッシュをオンにします
TXDeviceManager manager = mCloud.getDeviceManager();
if (manager.isAutoFocusEnabled()) {
manager.enableCameraAutoFocus(true);
}
manager.enableCameraTorch(true);
```

手順3：マイクのキャプチャをオンにする

startLocalAudioを呼び出してマイクのキャプチャをオンにするができます。このインターフェースは、`quality`パラメータを使用してキャプチャモードを決定してください。このパラメータの名前はqualityと呼ばれますが、品質が高いほどより良いという意味ではありません。さまざまなサービスシーンで最適なパラメータを選択できます（このパラメータのより正確な意味はsceneです）。

- **SPEECH**
  このモードのSDKオーディオモジュールは、音声信号の抽出に重点を置き、周囲の環境ノイズを可能な限りフィルタリングします。同時に、このモードのオーディオデータは、低品質ネットワークに対して最高の耐性を獲得するため、このモードは「ビデオ通話」や「オンライン会議」など、音声通信に重点を置いたシーンに特に適しています。
- **MUSIC**
  このモードのSDKは、高いオーディオ処理帯域幅とステレオモードを使用します。キャプチャ品質を最大化すると同時に、オーディオDSP処理モジュールを最も弱いレベルに調整して音質を最大化します。したがって、このモードは「音楽ライブストリーミング」シーン、特にキャスターがプロのサウンドカードを使用するライブストリーミングシーンに適しています。
- **DEFAULT**
  このモードのSDKは、スマート認識アルゴリズムを有効にして現在の環境を認識し、目的を明確にして最適な処理モードを選択します。ただし、最高の認識アルゴリズムでさえ不正確である場合もあります。製品の位置付けが明確な場合は、音声通信に重点を置いたSPEECHと音楽の音質に重点を置いたMUSICの両方から1つを選択することをお勧めします。

- Android java
- iOS&Mac ObjC
- Windows C++

```
// マイクのキャプチャをオンにし、現在のシーンを音声モード（ノイズ抑制能力が高く、弱いネットワークに対する耐性が強い）に設定します
mCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_SPEECH );

// マイクのキャプチャをオンにし、現在のシーンを音楽モード（ハイファイキャプチャ、低音質損失、プロのサウンドカードでの使用を推薦）に設定します
mCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_MUSIC);
```

手順4：TRTCの入室

入室を参照して現在のユーザーが入室します。入室すると、SDKはルーム内の他のユーザーに自分のオーディオストリームをリリースします。

> 注意：
> 当然ながら、入室（enterRoom）後にカメラプレビューとマイクキャプチャを開始することもできますが、ライブストリーミングシーンでは、キャスターにマイクのテストと美顔調整の時間を与える必要があるため、カメラとマイクを起動してから入室する方法は一般的です。

- Android java
- iOS&Mac ObjC
- Windows C++

```
mCloud = TRTCCloud.sharedInstance(getApplicationContext());
mCloud.setListener(mTRTCCloudListener);

// TRTC入室パラメータを組み立てるには、TRTCParamsの各フィールドを独自のパラメータに置き換えてください
// Please replace each field in TRTCParams with your own parameters
TRTCCloudDef.TRTCParams param = new TRTCCloudDef.TRTCParams();
params.sdkAppId = 1400000123; // Please replace with your own SDKAppID
params.userId = "denny"; // Please replace with your own userid
params.roomId = 123321; // Please replace with your own room number
params.userSig = "xxx"; // Please replace with your own userSig
params.role = TRTCCloudDef.TRTCRoleAnchor;

// 「オンラインライブストリーミング」のシーンの場合、ユースケースをTRTC_APP_SCENE_LIVEに設定
```

```
してください
// If your application scenario is a video call between several people, please u
se "TRTC_APP_SCENE_LIVE"
mCloud.enterRoom(param, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
```

手順5：ロール間の切り替え

TRTCの「ロール」

- 「ビデオ通話」(TRTC_APP_SCENE_VIDEOCALL)と「音声威通話」(TRTC_APP_SCENE_AUDIOCALL)の2つのシーンでは、入室時にロールを設定する必要はありません。その理由として、これらの2つのモードでは、各ユーザーはデフォルトでキャスター(Anchor)であるからです。
- 「ビデオライブストリーミング」(TRTC_APP_SCENE_LIVE)と「音声ライブストリーミング」(TRTC_APP_SCENE_VOICE_CHATROOM)の2つのシーンでは、各ユーザーは、入室時に「キャスター(Anchor)」または「視聴者(Audience)」のいずれかの自分の「ロール」を指定する必要があります。

ロールの切り替え

TRTCでは、「キャスター(Anchor)」のみがオーディオビデオストリームをリリースする権限を持っています。「視聴者(Audience(」はオーディオビデオストリームをリリースできません。
したがって、入室時に使用するロールが「視聴者(Audience)」である場合は、オーディオビデオストリームのリリース（「マイク・オン」とも呼ばれる）前に、**switchRole**インターフェースを呼び出してロールを「キャスター(Anchor)」に切り替える必要があります。

- Android java
- iOS ObjC
- Mac ObjC
- Windows C++

```
// 現在のロールが視聴者(Audience)の場合は、最初にswitchRoleを呼び出してキャスター(Anchor)
に切り替えてください
// If your current role is 'audience', you need to call switchRole to switch to
'anchor' first
mCloud.switchRole(TRTCCloudDef.TRTCRoleAnchor);
mCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);
mCloud.startLocalPreview(true, cameraVide);

// ロールの切り替えに失敗した場合、onSwitchRoleコールバックのエラーコードは0ではありません
// If switching operation failed, the error code of the 'onSwitchRole' is not ze
ro
@Override
public void onSwitchRole(final int errCode, final String errMsg) {
if (errCode != 0) {
```

```
Log.d(TAG, "Switching operation failed...");
}
}
```

注：ルームにキャスターが多すぎると、switchRoleはロールの切り替えに失敗し、エラーコードがTRTCのonSwitchRoleを介して通知されます。したがって、オーディオビデオストリームをリリースする必要がなくなったら（「マイク・オフ」とも呼ばれる）、switchRoleを再度呼び出して、「視聴者(Audience)」に切り替えてください。

説明：
質問がある場合があるかもしれませんが、オーディオビデオストリームをリリースできるのはキャスターなら、すべてのユーザーにキャスターのロールで入室させることはできますか？答えは間違いなく「いいえ」です。その理由については、1つのルームに同時に存在するオーディオとビデオのチャネルはいくつですか？ドキュメントをご参照ください。

拡張機能ガイド

1. 1つのルームに同時に存在するオーディオとビデオのチャネルはいくつですか？

1つのTRTCルームに最大**50**チャネルのオーディオビデオストリームが同時に許可されます。「先着順」の原則に従って、余分なオーディオビデオストリームは破棄されます。
大部分のシーンでは、2人の間のビデオ通話から、数万人が同時に視聴するオンラインライブストリーミングまで、50のオーディオビデオストリームがユースケースのニーズを満たすことができますが、**ロール管理**を適切に行うことが前提です。

いわゆる「ロール管理」とは、入室するユーザーにロールをアサインする必要があるということです。

- ユーザー自体は、ライブストリーミングシーンの「キャスター」、またはオンライン教育シーンの「教師」、またはオンライン会議シーンの「ホスト」である場合、そのユーザーに「キャスター（Anchor）」のロールをアサインすることができます。
- ユーザー自体は、ライブストリーミングシーンの「視聴者」、またはオンライン教育シーンの「学生」、またはオンライン会議シーンの「オーディエンス」である場合、「視聴者」のロールをアサインする必要があります。そうしないと、膨大な数の人々が即時キャスターの制限数を「絞り出します」。

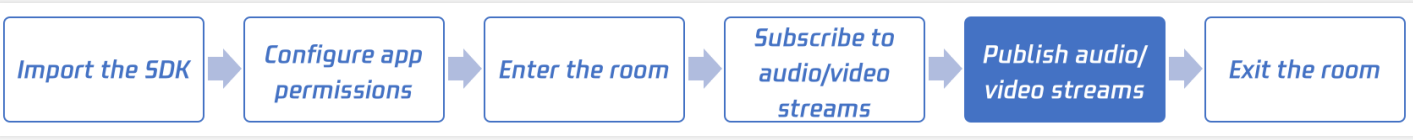
「視聴者」がオーディオビデオストリームをリリースする必要がある（「マイク・オン」）場合にのみ、switchRoleを介して「キャスター」のロールに切り替える必要があります。オーディオビデオストリームをリリースする必要がなくなった（「マイク・オフ」）場合は、すぐ視聴者のロールに戻してください。

合理的なロール管理により、通常、ルームでオーディオビデオストリームを同時にリリースする必要のある「キャスター」は50人以下であることがわかります。そうしないと、ルーム全体が「乱雑」になります。同時音声数が6チャンネルを超えると、一般の人が現在音声から話している相手を特定することは困難です。

Web

最終更新日：：2022-08-08 15:25:07

ここでは主に、キャスターが自分のオーディオビデオストリーミングを公開する方法についてご説明します。いわゆる「公開」とは、マイクとカメラをオンにして、自分の音声やビデオをルーム内にいる他のユーザーに聞かせたり、見せたりすることです。



TRTC Web SDKの使用中には、以下のオブジェクトが頻繁に登場します。

- Client オブジェクト。ローカルクライアントを表します。Clientクラスのメソッドにより、通話ルームへの参加、ローカルストリーミングの公開、リモートストリーミングの閲覧などの機能を提供します。
- Streamオブジェクト。オーディオビデオストリーミングオブジェクトを表し、ローカルのオーディオビデオストリーミングオブジェクトLocalStream、およびリモート側のオーディオビデオストリーミングオブジェクトRemoteStreamが含まれます。Streamクラスのメソッドでは主に、オーディオビデオストリーミングオブジェクトのアクションを提供し、これにはオーディオおよびビデオの再生コントロールが含まれます。

ステップ1：前のステップの完了

ドキュメント入室を参照し、Clientを作成して入室し、その後のオーディオビデオストリーミングの公開のための準備を行います。

ステップ2：ローカルストリームの作成

TRTC.createStream()メソッドを使用して、ローカルのオーディオビデオストリームを作成します。パラメータは次のように設定します。

パラメータ名	フィールドの意味	補足説明	データタイプ	入力例	デフォルト値
userId	ユーザーID	ユーザー名です。作成したClientのuserIdと一致させます	string	“denny”または“123321”	なし**入力必須**

パラメータ名	フィールドの意味	補足説明	データタイプ	入力例	デフォルト値
audio	オーディオキャプチャの有無	マイクを通じてオーディオをキャプチャするかどうかを選択します	boolean	true	なし**入力必須**
video	ビデオキャプチャの有無	カメラを通じてビデオをキャプチャするかどうかを選択します	boolean	true	なし**入力必須**

パラメータのより詳細な説明については、TRTC.createStream()をご参照ください。

```
const localStream = TRTC.createStream({ userId, audio: true, video: true });
```

公開前にLocalStream.initialize()を呼び出して、ローカルのオーディオビデオストリームを初期化する必要があります。
初期化中に、ブラウザを通じてカメラおよびマイクの使用権限を取得する場合があり、デバイスの使用を許可する必要が生じることがありますのでご注意ください。

初期化成功のコールバックにおいて、LocalStream.play()メソッドを呼び出して、ページ内でローカルオーディオビデオのプレビュー再生を行うことができます。

```
// Promise構文の使用
localStream
.initialize()
.then(() => {
console.log('ローカルストリーミング初期化の成功');
localStream.play('local_stream');
})
.catch(error => {
console.error('ローカルストリーミング初期化の失敗 ' + error);
});
// 同様の効果を得るには、async/await構文の使用をお勧めします
try {
await localStream.initialize();
localStream.play('local_stream');
console.log('ローカルストリーミング初期化の成功');
} catch (error) {
console.error('ローカルストリーミング初期化の失敗 ' + error);
}
```

ステップ3：ローカルストリームの公開

ローカルストリームの初期化に成功したら、Client.publish()メソッドを呼び出して、ローカルストリームを公開します。

```
// Promise構文の使用
client
.publish(localStream)
.then(() => {
console.log('ローカルストリーミング公開の成功');
})
.catch(error => {
console.error('ローカルストリーミング公開の失敗 ' + error);
});
// 同様の効果を得るには、async/await構文の使用をお勧めします
try {
await client.publish(localStream);
console.log('ローカルストリーミング公開の成功');
} catch (error) {
console.error('ローカルストリーミング公開の失敗 ' + error);
}
```

完全なコード

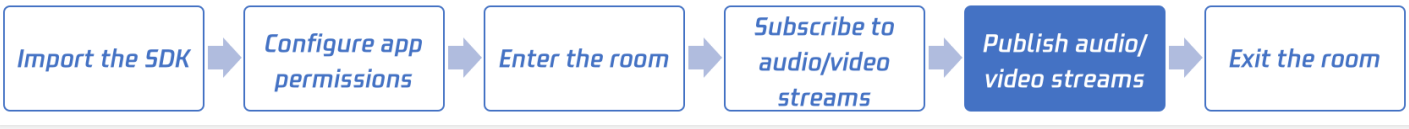
```
const client = TRTC.createClient({
mode: 'rtc',
sdkAppId,
userId,
userSig
});
const localStream = TRTC.createStream({ userId, audio: true, video: true });
try {
await client.join({ roomId });
console.log('入室成功');
} catch (error) {
console.error('入室に失敗しました。しばらくしてからもう一度お試しください'+ error);
}
try {
await localStream.initialize();
localStream.play('local_stream');
console.log('ローカルストリーミング初期化の成功');
} catch (error) {
```

```
console.error('ローカルストリーミング初期化の失敗 ' + error);
}
try {
await client.publish(localStream);
console.log('ローカルストリーミング公開の成功');
} catch (error) {
console.error('ローカルストリーミング公開の失敗 ' + error);
}
```

Electron

最終更新日：：2022-07-26 15:46:58

ここでは主に、キャスターが自分のオーディオビデオストリーミングを公開する方法についてご説明します。いわゆる「公開」とは、マイクとカメラをオンにして、自分の音声やビデオをルーム内にいる他のユーザーに聞かせたり、見せたりすることです。



呼び出しガイド

ステップ1：前のステップの完了

ドキュメントSDKのプロジェクトへのインポートを参照し、SDKのインポートと設定を完了してください。

ステップ2：カメラのプレビューを開く

startLocalPreviewインターフェースを呼び出すと、カメラのプレビューを開くことができます。このとき、SDKはシステムにカメラの使用権限を申請し、ユーザーは承認を受けることでカメラのキャプチャ処理を開始することができます。

ローカル画面のレンダリングパラメータを設定したい場合は、**setLocalRenderParams**インターフェースを呼び出して、ローカルプレビューのレンダリングパラメータを設定することで行えます。プレビューを有効にしてからプレビューのパラメータを設定すると画面が飛んでしまうことを防ぐため、プレビューのパラメータを設定する必要がある場合は、プレビューを有効にする前にこのコマンドを呼び出すことをお勧めします。

```
// ローカル画面のプレビューモードの設定：左右のイメージを有効にし、画面をフィルモードに設定します
import TRTCCloud, {
TRTCRenderParams, TRTCVideoRotation,
TRTCVideoFillMode, TRTCVideoMirrorType
} from 'trtc-electron-sdk';
const param = new TRTCRenderParams(
TRTCVideoRotation.TRTCVideoRotation0,
TRTCVideoFillMode.TRTCVideoFillMode_Fill,
TRTCVideoMirrorType.TRTCVideoMirrorType_Auto
);
const rtcCloud = new TRTCCloud();
```

```
rtcCloud.setLocalRenderParams(param);
const cameraVideoDom = document.querySelector('.camera-dom');
rtcCloud.startLocalPreview(cameraVideoDom);
```

ステップ3：マイクキャプチャを有効にする

startLocalAudioを呼び出すことによって、マイクのキャプチャを有効にすることができます。このインターフェースでは、`quality` パラメータを介してキャプチャモードを決定する必要があります。このパラメータの名前はqualityですが、品質が高いほど良いというわけではなく、ビジネスシナリオによって選択すべき最適なパラメータがあります（このパラメータのより正確な意味はsceneです）。

- **SPEECH**
  このモードのSDKオーディオモジュールは、音声信号をブラッシュアップし、周囲のノイズを可能な限り除去することに重点を置いています。また、このモードのオーディオデータは、質の悪いネットワークに対しても最高の耐性が得られるため、「ビデオ通話」や「オンラインミーティング」といった音声通信に焦点を当てたシナリオに特に適しています。
- **MUSIC**
  このモードのSDKは、高い音声処理帯域幅とステレオモードを使用してキャプチャの品質を最大限に高めながら、オーディオのDSP処理モジュールを可能な限り弱いレベルに調整して音質を最大限に高めています。そのためこのモードは、「音楽ライブストリーミング」のシナリオ、特にキャスターがプロ用サウンドカードを使用して音楽ライブを行っているシナリオに適しています。
- **DEFAULT**
  このモードのSDKは、インテリジェントな認識アルゴリズムが現在の環境を識別し、最適な処理モードを選択することができます。ただし、どんなに優れた認識アルゴリズムでも不正確な場合はあります。自社製品のポジショニングがはっきりしている場合は、音声通信に特化したSPEECHと音楽の音質に特化したMUSICのうちどちらかを選択することをお勧めします。

```
import { TRTCAudioQuality } from 'trtc-electron-sdk';
// マイクキャプチャをオンにして、現在のシナリオを音声モード（高いノイズ抑制能力と脆弱なネットワーク耐性）に設定します
rtcCloud.startLocalAudio(TRTCAudioQuality.TRTCAudioQualitySpeech);
// マイクキャプチャをオンにして、現在のシナリオを音楽モード（高忠実度のキャプチャ、低音質損失、プロ用サウンドカードとの使用を推奨）に設定します
rtcCloud.startLocalAudio(TRTCAudioQuality.TRTCAudioQualityMusic);
```

ステップ4：TRTCルームへの入室

ドキュメント入室をご参照のうえ、現在のユーザーをTRTCルームに入室させます。SDKはルームにいる他のユーザーに対し自分のオーディオストリームの公開を開始します。

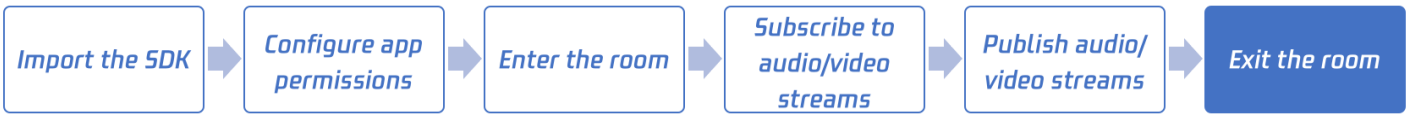
注意：
もちろん、入室(enterRoom)後にカメラプレビューやマイクキャプチャを開始することもできますが、ライブストリーミングシナリオでは、まずキャスターにマイクテストや美顔のエフェクト調整の時間を与える必要があるので、入室前にカメラとマイクを先に開始することがより一般的な方法です。

```
import { TRTCParams, TRTCRoleType, TRTCAppScene } from 'trtc-electron-sdk';
// TRTC入室パラメータを組み立てる場合、TRTCParamsの各フィールドをご自分のパラメータに置き換えてください
// Please replace each field in TRTCParams with your own parameters
const param = new TRTCParams();
params.sdkAppId = 1400000123; // Please replace with your own SDKAppID
params.userId = "denny"; // Please replace with your own userid
params.roomId = 123321; // Please replace with your own room number
params.userSig = "xxx"; // Please replace with your own userSig
params.role = TRTCRoleType.TRTCRoleAnchor;
// シナリオが「オンラインライブストリーミング」の場合、ユースケースをTRTC_APP_SCENE_LIVEに設定してください
// If your application scenario is a video call between several people, please use "TRTC_APP_SCENE_LIVE"
rtcCloud.enterRoom(param, TRTCAppScene.TRTCAppSceneLIVE);
```

5. 退室
Android&iOS&Windows&Mac

最終更新日：：2022-07-07 15:10:00

このドキュメントでは、主に現在のTRTCからの自発的な退室方法を紹介し、どのような状況で強制的に退室するかについても紹介します：



呼び出しガイド

手順1：事前手順の完了

SDKのプロジェクトへのインポートを参照して、SDKのインポートおよびApp権限の設定を完了します。
ドキュメント入室を参照して、入室プロセスを完了してください。

手順2：現在のルームからの自発的な退室

exitRoomインターフェースを呼び出すことで現在のルームから退室できます。SDKは、退室後にonExitRoom(int reason)コールバックイベントを介して通知します。

- Android java
- iOS&Mac swift
- Windows C++

```
// 現在のルームから退室します
mCloud.exitRoom();
```

exitRoomインターフェースを呼び出すと、SDKは、退室プロセスに入ります。これには、2つの非常に重要なタスクがあります：

- **キータスク1：退室の公開**
  ルーム内の他のユーザーに、現在のルームを離れようすることを通知します。ルーム内の他のユーザーは、ユーザーから**onRemoteUserLeaveRoom**コールバックを受け取ります。そうでない場合には、他のユーザーは当該ユーザーが「デッドロック」していると誤解する可能性があります。

- **キータスク2：デバイス権限のリリース**
  ユーザーが退室する前にオーディオビデオストリームを公開している場合、退室プロセスでは、カメラとマイクをオフにしたり、デバイスの使用権をリリースしたりする必要もあります。

したがって、TRTCCloudインスタンスをリリースしたい場合は、onExitRoomコールバックを受信してからリリースすることをお勧めします。

手順3：現在のルームからの強制退出

ユーザーの自発的な退室を除いて、onExitRoom(int reason)コールバックを受け取る2つのケースがあります：

- **ケース1：現在のルームからの強制退室**
  サービス側のRemoveUser | RemoveUserByStrRoomIdインターフェースを介して、特定のユーザーを特定のTRTCルームから強制退室させます。当該ユーザーを強制退室させると、当該ユーザーはonExitRoom(1)のコールバックを受け取ります。

- **ケース2：現在のルームが解散される**
  サーバー側のDismissRoom | DismissRoomByStrRoomIdインターフェースを介して、特定のTRTCルームを解散できます。ルームを解散すると、ルーム内のすべてのユーザーはonExitRoom(2)のコールバックを受け取ります。

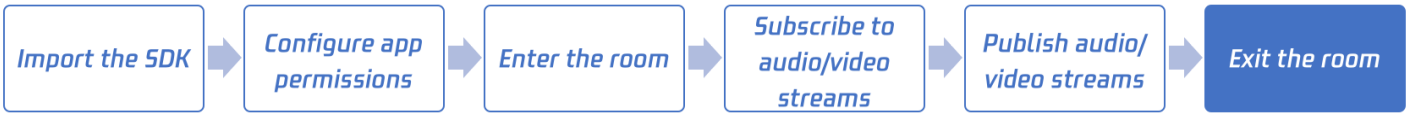
- Android java
- iOS&Mac ObjC
- Windows C++

```
// onExitRoomコールバックを監視することで、退室の理由を確認できます
@Override
public void onExitRoom(int reason) {
if (reason == 0) {
Log.d(TAG, "Exit current room by calling the 'exitRoom' api of sdk ...");
} else if (reason == 1) {
Log.d(TAG, "Kicked out of the current room by server through the restful api..."
);
} else if (reason == 2) {
Log.d(TAG, "Current room is dissolved by server through the restful api...");
}
}
```

Web

最終更新日：：2022-08-08 15:25:07

ここでは主に、現在のTRTCルームから自主的に退出する方法と、どのような状況下で強制的に退出させられるかについてご説明します。



TRTC Web SDKの使用中には、以下のオブジェクトが頻繁に登場します。

- Client オブジェクト。ローカルクライアントを表します。Clientクラスのメソッドにより、通話ルームへの参加、ローカルストリーミングの公開、リモートストリーミングの閲覧などの機能を提供します。
- Streamオブジェクト。オーディオビデオストリーミングオブジェクトを表し、ローカルのオーディオビデオストリーミングオブジェクトLocalStream、およびリモート側のオーディオビデオストリーミングオブジェクトRemoteStreamが含まれます。Streamクラスのメソッドでは主に、オーディオビデオストリーミングオブジェクトのアクションを提供し、これにはオーディオおよびビデオの再生コントロールが含まれます。

ステップ1：前のステップの完了

ドキュメント入室を参照し、clientを作成して入室します。

ステップ2：現在のルームから自主的に退出する

通話終了時は、Client.leave()メソッドを呼び出して、オーディオビデオ通話ルームを退出し、すべてのオーディオビデオ通話によるセッションが終了します。

```
await client.leave();
```

ステップ3：現在のルームから退出させられる

ユーザーが自主的に退室する以外に、次のような場合、ユーザーは `CLIENT_BANNED` イベントを受信します。これは、そのユーザーが退室させられたことを表します。

```
client.on('client-banned', error => {
console.error('client-banned observed: ' + error.message);
// client-banned observed: client was banned because of duplicated userId joining the room.
// client-banned observed: client was banned because of you got banned by account admin
});
```

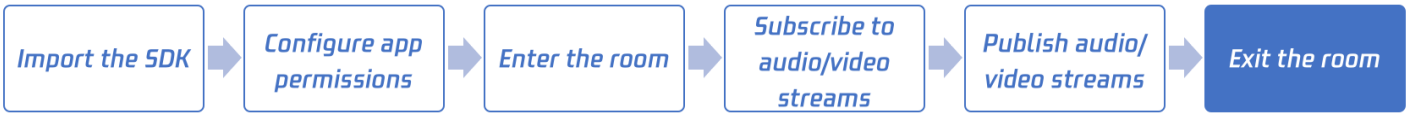
- **状況1：同名のユーザーによって現在のルームから強制退出させられる**
  1つのルーム内に、userIdが同じで、ロールがどちらもキャスターであるユーザーが同時に現れた場合、先に入室していたユーザーがルームから強制退出させられます。
  例えば、2名のユーザーA、Bが、同一の `userId` で相次いで入室した場合、AはBによって退室させられます。
  同名ユーザーによる同一ルームへの同時入室は許可されない行為であり、双方のオーディオビデオ通話に異常を起こすおそれがあるため、このような状況を避けなければなりません。

- **状況2：サーバーAPIによって現在のルームから強制退出させられる、または現在のルームが解散される**
  サーバー側のRemoveUser | RemoveUserByStrRoomIdインターフェースによって、あるユーザーをあるTRTCルームから強制退出させることができます。このユーザーは退室させられると、 `CLIENT_BANNED` イベントを受信します。あるいは、サーバー側のDismissRoom | DismissRoomByStrRoomIdインターフェースによって、あるTRTCルームを解散させることができます。ルームの解散後、このルームのすべてのユーザーは `CLIENT_BANNED` イベントを受信します。

Electron

最終更新日：：2022-07-21 16:47:57

このドキュメントでは、主に現在のTRTCからの自発的な退室方法を紹介し、どのような状況で強制的に退室するかについても紹介します：



呼び出しガイド

手順1：事前手順の完了

1. SDKのプロジェクトへのインポートを参照して、SDKのインポートおよびApp権限の設定を完了します。
2. ドキュメント入室を参照して、入室プロセスを完了してください。

手順2：現在のルームからの自発的な退室

exitRoomインターフェースを呼び出すことで現在のルームから退室できます。SDKは、退室後にonExitRoom(int reason)コールバックイベントを介して通知します。

```
import TRTCCloud from 'trtc-electron-sdk';
const trtcCloud = new TRTCCloud();
// 現在のルームから退室します
trtcCloud.exitRoom();
```

exitRoomインターフェースを呼び出すと、SDKは、退室プロセスに入ります。これには、2つの非常に重要なタスクがあります：

- **キータスク1：退室の公開**
  ルーム内の他のユーザーに、現在のルームを離れようすることを通知します。ルーム内の他のユーザーは、ユーザーから**onRemoteUserLeaveRoom**コールバックを受け取ります。そうしないと、他のユーザーは当該ユーザーが「デッドロック」していると誤解する可能性があります。
- **キータスク2：デバイス権限のリリース**
  ユーザーが退室する前にオーディオビデオストリームを公開している場合、退室プロセスでは、カメラとマイクをオフにしたり、デバイスの使用権をリリースしたりする必要もあります。

したがって、TRTCCloudインスタンスをリリースしたい場合は、onExitRoomコールバックを受信してからリリースすることをお勧めします。

手順3：現在のルームからの強制退出

ユーザーの自発的な退室を除いて、onExitRoomコールバックを受け取る2つのケースがあります：

- **ケース1：現在のルームからの強制退室**
  サービス側のRemoveUser | RemoveUserByStrRoomIdインターフェースを介して、特定のユーザーを特定のTRTCルームから強制退室させます。当該ユーザーを強制退室させると、当該ユーザーはonExitRoom(1)のコールバックを受け取ります。

- **ケース2：現在のルームが解散される**
  サーバー側のDismissRoom | DismissRoomByStrRoomIdインターフェースを介して、特定のTRTCルームを解散できます。ルームを解散すると、ルーム内のすべてのユーザーはonExitRoom(2)のコールバックを受け取ります。

```
// onExitRoomコールバックを監視することで、退室の理由を確認できます
function onExitRoom(reason) {
console.log(`onExitRoom reason: ${reason}`);
}
trtcCloud.on('onExitRoom', onExitRoom);
```

6. Advanced Guide
ネットワーク品質の検出
Android&iOS&Windows&Mac

最終更新日：：2022-07-07 15:10:00

このドキュメントでは、主に現在のネットワーク品質の検出方法を紹介します。

WeChatビデオ通話を使用しているときに、ネットワーク環境が悪い場合（たとえば、エレベーターに入った後）、WeChatビデオ通話のインターフェースで「現在のネットワーク品質が悪い」と表示されます。このドキュメントでは、主にTRTCを介して同じインタラクションを実行する方法について説明します。

呼び出しガイド

TRTCは、**onNetworkQuality**と呼ばれるコールバックイベントを提供します。このイベントは、現在のネットワーク品質を2秒ごとに報告します。そのパラメーターには、localQualityとremoteQualityの2つの部分が含まれます：

- **localQuality** ：それぞれExcellent、Good、Poor、Bad、VeryBadとDownの6つのレベルに分けて、現在のネットワーク品質を表します。
- **remoteQuality**：リモートユーザーのネットワーク品質を表します。これはプライムグループであり、プライムグループのそれぞれの要素はリモートユーザーのそれぞれのネットワーク品質を表します。

Quality	名称	説明
0	Unknown	検出されていません
1	Excellent	現在のネットワークは非常に良い
2	Good	現在のネットワークは比較的良い
3	Poor	現在のネットワークは一般です
4	Bad	現在のネットワークは悪い。明らかなラグや通話の遅延が発生する可能性があります
5	VeryBad	現在のネットワークは非常に悪い。TRTCは接続をほぼ維持できませんが、通信品質を保証することはできません
6	Down	現在のネットワークはTRTCの最小要件を満たしていないため、通常のオーディオビデオ通話を行うことはできません。

TRTCのonNetworkQualityを監視し、インターフェースで対応するプロンプトを表示するだけでよい。

- Android java
- iOS&Mac ObjC
- Windows C++

```
// onNetworkQualityコールバックを監視し、現在のネットワーク状態変化を検知します
@Override
public void onNetworkQuality(TRTCCloudDef.TRTCQuality localQuality,
ArrayList<trtcclouddef.trtcquality> remoteQuality)
{
// Get your local network quality
switch(localQuality) {
case TRTCQuality_Unknown:
Log.d(TAG, "SDK has not yet sensed the current network quality.");
break;
case TRTCQuality_Excellent:
Log.d(TAG, "The current network is very good.");
break;
case TRTCQuality_Good:
```

```
Log.d(TAG, "The current network is good.");
break;
case TRTCQuality_Poor:
Log.d(TAG, "The current network quality barely meets the demand.");
break;
case TRTCQuality_Bad:
Log.d(TAG, "The current network is poor, and there may be significant freezes an
d call delays.");
break;
case TRTCQuality_VeryBad:
Log.d(TAG, "The current network is very poor, the communication quality cannot b
e guaranteed");
break;
case TRTCQuality_Down:
Log.d(TAG, "The current network does not meet the minimum requirements.");
break;
default:
break;
}
// Get the network quality of remote users
for (TRTCCloudDef.TRTCQuality info : arrayList) {
Log.d(TAG, "remote user : = " + info.userId + ", quality = " + info.quality);
}
}
```

Web

最終更新日：：2022-08-08 15:25:07

入室前または通話中にユーザーのネットワーク品質のテストを行い、ユーザーの現在のネットワーク品質状況を事前に判断することができます。ユーザーのネットワーク品質があまりに悪い場合は、正常な通話品質を保証するため、ネットワーク環境を切り替えるようユーザーに推奨する必要があります。

ここでは、NETWORK_QUALITYイベントに基づいて、通話前にネットワーク品質テストを実施する方法についてご説明します。通話中のネットワーク品質感知は、NETWORK_QUALITYイベントを監視するだけで可能です。

実施フロー

1. TRTC.createClientを呼び出して2つのClientを作成し、それぞれをuplinkClientとdownlinkClientと呼びます。
2. これらのClientで同一のルームに入室します。
3. uplinkClientを使用してプッシュし、NETWORK_QUALITYイベントを監視してアップストリームネットワーク品質をテストします。
4. downlinkClientを使用してプルし、NETWORK_QUALITYイベントを監視してダウンストリームネットワーク品質をテストします。
5. プロセス全体を15秒程度継続し、最後にネットワーク品質の平均をとることで、アップストリームとダウンストリームネットワークの状況をおおよそ判断することができます。

> 注意：
> テストの過程で少額の基本サービス料金が発生します。プッシュの解像度を指定しない場合、デフォルトでは640*480の解像度でプッシュを行います。

コード例

```
let uplinkClient = null; // アップストリームネットワーク品質のテストに使用します
let downlinkClient = null; // ダウンストリームネットワーク品質のテストに使用します
let localStream = null; // テスト用のストリームです
let testResult = {
// アップストリームネットワーク品質データを記録します
uplinkNetworkQualities: [],
// ダウンストリームネットワーク品質データを記録します
downlinkNetworkQualities: [],
```

```
average: {
uplinkNetworkQuality: 0,
downlinkNetworkQuality: 0
}
}
// 1. アップストリームネットワーク品質をテストします
async function testUplinkNetworkQuality() {
uplinkClient = TRTC.createClient({
sdkAppId: 0, // sdkAppIdを入力
userId: 'user_uplink_test',
userSig: '', // uplink_testのuserSig
mode: 'rtc'
});
localStream = TRTC.createStream({ audio: true, video: true });
// 実際の業務シナリオに応じてvideo profileを設定します
localStream.setVideoProfile('480p');
await localStream.initialize();
uplinkClient.on('network-quality', event => {
const { uplinkNetworkQuality } = event;
testResult.uplinkNetworkQualities.push(uplinkNetworkQuality);
});
// テスト用のルームに入室します。競合防止のため、ルームナンバーはランダムにします
await uplinkClient.join({ roomId: 8080 });
await uplinkClient.publish(localStream);
}
// 2. ダウンストリームネットワーク品質をテストします
async function testDownlinkNetworkQuality() {
downlinkClient = TRTC.createClient({
sdkAppId: 0, // sdkAppIdを入力
userId: 'user_downlink_test',
userSig: '', // userSig
mode: 'rtc'
});
downlinkClient.on('stream-added', async event => {
await downlinkClient.subscribe(event.stream, { audio: true, video: true });
// サブスクリプション成功後、ネットワーク品質イベントの監視を開始します
downlinkClient.on('network-quality', event => {
const { downlinkNetworkQuality } = event;
testResult.downlinkNetworkQualities.push(downlinkNetworkQuality);
});
})
// テスト用のルームに入室します。競合防止のため、ルームナンバーはランダムにします
await downlinkClient.join({ roomId: 8080 });
}
// 3. テストを開始します
testUplinkNetworkQuality();
testDownlinkNetworkQuality();
```

```
// 4. 15秒後にテストを停止し、平均ネットワーク品質を計算します
setTimeout(() => {
// アップストリーム平均ネットワーク品質を計算します
if (testResult.uplinkNetworkQualities.length > 0) {
testResult.average.uplinkNetworkQuality = Math.ceil(
testResult.uplinkNetworkQualities.reduce((value, current) => value + current, 0)
/ testResult.uplinkNetworkQualities.length
);
}
if (testResult.downlinkNetworkQualities.length > 0) {
// ダウンストリーム平均ネットワーク品質を計算します
testResult.average.downlinkNetworkQuality = Math.ceil(
testResult.downlinkNetworkQualities.reduce((value, current) => value + current, 0
) / testResult.downlinkNetworkQualities.length
);
}

// テストを終了し、関連のステータスを消去します。
uplinkClient.leave();
downlinkClient.leave();
localStream.close();
}, 15 * 1000);
```

結果の分析

上記のステップにより、アップストリーム平均ネットワーク品質、ダウンストリーム平均ネットワーク品質を取得できます。ネットワーク品質の列挙値は次のとおりです。

数値	意味
0	ネットワークの状況は不明。現在のclientインスタンスが アップストリーム/ダウンストリーム接続を確立していないことを表します
1	ネットワーク状況は極めて良好
2	ネットワーク状況は比較的良好
3	ネットワーク状況は普通
4	ネットワーク状況は不良
5	ネットワーク状況は極めて不良

数値	意味
6	ネットワーク接続が切断されている。**注意：ダウンストリームネットワーク品質がこの値の場合は、すべてのダウンストリーム接続が切断されていることを表します**

推奨事項：
ネットワーク品質の値が3より大きい場合は、ネットワークをチェックしてネットワーク環境の切り替えを試すようユーザーに促す必要があり、そうしなければ正常なオーディオビデオ通話品質を保証することは困難です。
あるいは、下記のポリシーによって帯域幅の消費を抑えることも可能です。

- アップストリームネットワーク品質の値が3より大きい場合は、LocalStream.setVideoProfile()インターフェースによってビットレートを下げるか、またはLocalStream.muteVideo()メソッドによってビデオをオフにし、アップストリーム帯域幅の消費を抑えることができます。
- ダウンストリームネットワーク品質の値が3より大きい場合は、スモールストリームのサブスクリプション（参考：デュアルストリームの伝送を有効にする）またはオーディオのみのサブスクリプションを行う方法で、ダウンストリーム帯域幅の消費を抑えることができます。

Electron

最終更新日：：2022-07-21 16:51:16

このドキュメントでは、主に現在のネットワーク品質の検出方法を紹介します。

WeChatビデオ通話を使用しているときに、ネットワーク環境が悪い場合（たとえば、エレベーターに入った後）、WeChatビデオ通話のインターフェースで「現在のネットワーク品質が悪い」と表示されます。このドキュメントでは、主にTRTCを介して同じインタラクションを実行する方法について説明します。

呼び出しガイド

TRTCは、**onNetworkQuality**と呼ばれるコールバックイベントを提供します。このイベントは、現在のネットワーク品質を2秒ごとに報告します。そのパラメーターには、localQualityとremoteQualityの2つの部分が含まれます：

- **localQuality** ：それぞれExcellent、Good、Poor、Bad、VeryBadとDownの6つのレベルに分けて、現在のネットワーク品質を表します。

- **remoteQuality**：リモートユーザーのネットワーク品質を表します。これはプライムグループであり、プライムグループのそれぞれの要素はリモートユーザーのそれぞれのネットワーク品質を表します。

Quality	名称	説明
0	Unknown	検出されていません
1	Excellent	現在のネットワークは非常に良い
2	Good	現在のネットワークは比較的良い
3	Poor	現在のネットワークは一般です
4	Bad	現在のネットワークは悪い。明らかなラグや通話の遅延が発生する可能性があります
5	VeryBad	現在のネットワークは非常に悪い。TRTCは接続をほぼ維持できませんが、通信品質を保証することはできません
6	Down	現在のネットワークはTRTCの最小要件を満たしていないため、通常のオーディオビデオ通話を行うことはできません。

TRTCのonNetworkQualityを監視し、インターフェースで対応するプロンプトを表示するだけでよい。

```
import TRTCCloud, { TRTCQuality } from 'trtc-electron-sdk';
const rtcCloud = new TRTCCloud();
function onNetworkQuality(localQuality, remoteQuality) {
switch(localQuality.quality) {
case TRTCQuality.TRTCQuality_Unknown:
console.log('SDK has not yet sensed the current network quality.');
break;
case TRTCQuality.TRTCQuality_Excellent:
console.log('The current network is very good.');
break;
case TRTCQuality.TRTCQuality_Good:
console.log('The current network is good.');
break;
case TRTCQuality.TRTCQuality_Poor:
console.log('The current network quality barely meets the demand.');
break;
case TRTCQuality.TRTCQuality_Bad:
console.log('The current network is poor, and there may be significant freezes an
d call delays.');
break;
case TRTCQuality.TRTCQuality_Vbad:
```

```
console.log('The current network is very poor, the communication quality cannot b
e guaranteed.');
break;
case TRTCQuality.TRTCQuality_Down:
console.log('The current network does not meet the minimum requirements.');
break;
default:
break;
}
for (let i = 0; i < remoteQuality.length; i++) {
console.log(`remote user: ${remoteQuality[i].userId}, quality: ${remoteQuality
[i].quality}`);
}
}
rtcCloud.on('onNetworkQuality', onNetworkQuality);
```

画面共有の有効化
iOS

最終更新日：：2022-07-07 15:10:00

Tencent CloudのTRTCは、iOSプラットフォームで2つの異なる画面共有ソリューションをサポートしています：

- **アプリ内共有**
  現在のアプリの画面のみを共有できます。この特性をサポートするには、iOSのバージョン13以降のOSが必要です。現在のアプリ以外の画面コンテンツを共有できないため、高度なプライバシー保護が必要なシーンに適しています。
- **アプリケーション間共有**
  AppleのReplaykitソリューションをベースとして、システム全体の画面コンテンツを共有できます。ただし、現在のアプリはExtension拡張コンポーネントを追加で提供する必要があるため、結合手順はアプリ内共有よりもやや多くなります。

サポートするプラットフォーム

iOS	Android	Mac OS	Windows	Electron	Chromeブラウザ
✓	✓	✓	✓	✓	✓

アプリ内共有

アプリケーション内の共有ソリューションは非常にシンプルです。TRTC SDKが提供するインターフェースstartScreenCaptureInAppを呼び出し、エンコードパラメータ `TRTCVideoEncParam` を渡すだけでOKです。このパラメータはnilに設定することができます。この場合、SDKは画面共有が開始される前のエンコードパラメータを引き続き使用します。

iOS画面共有に推奨されるエンコードパラメータは次のとおりです：

パラメータ項目	パラメータ名	通常の推奨値	テキスト教育シナリオ
解像度	videoResolution	1280 × 720	1920 × 1080
フレームレート	videoFps	10 FPS	8 FPS
最高ビットレート	videoBitrate	1600 kbps	2000 kbps

パラメータ項目	パラメータ名	通常の推奨値	テキスト教育シナリオ
解像度アダプティブ	enableAdjustRes	NO	NO

- 画面共有されるコンテンツには通常、大幅な変更がなく、FPSを高く設定するのは経済的ではないため、10FPSを推奨します。
- 共有したい画面コンテンツに大量のテキストが含まれている場合、解像度とビットレートを適宜引き上げることができます。
- 最高ビットレート(videoBitrate)とは、画面が大きく変化したときの最高出力ビットレートのことです。画面コンテンツの変化が少ない場合、実際のエンコードビットレートは低くなります。

アプリケーション間共有

iOSシステムでアプリケーション間画面共有を行うには、**Broadcast Upload Extension**スクリーンキャプチャプロセスを追加して、メインAppプロセスと連携してプッシュを行う必要があります。Extensionスクリーンキャプチャプロセスは、システムによってスクリーンキャプチャが必要なときに作成され、システムが収集した画面イメージの受信を担当します。従って、以下を実行してください：

1. App Groupを作成し、XCodeにおいて設定を行います（オプション）。このステップは、Extensionスクリーンキャプチャプロセスが、メインAppプロセスとプロセス間通信できるようにすることを目的としています。
2. プロジェクトにおいて、**Broadcast Upload Extension**のTargetを新規作成し、SDK圧縮パッケージに拡張モジュール用としてカスタマイズされた `TXLiteAVSDK_ReplayKitExt.framework` を統合します。
3. メインApp側の受信ロジックに対して、メインApp**Broadcast Upload Extension**からのスクリーンキャプチャデータを待機させます。

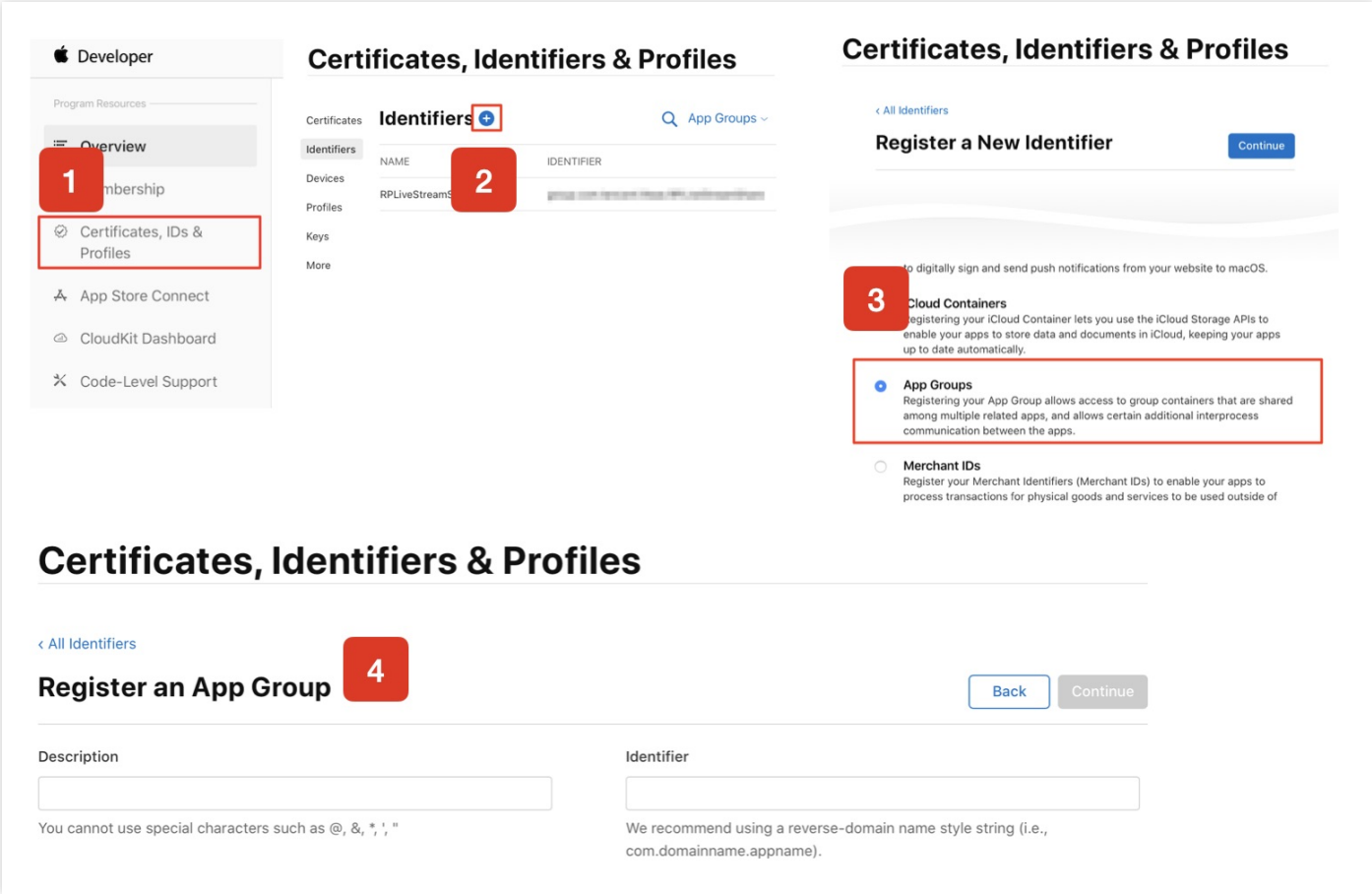
> 注意：
> 手順1をスキップした場合、すなわちApp Groupsを設定しない場合は（インターフェースはnilを渡します）、画面共有は実行できますが、安定性が若干損なわれます。手順はやや多いですが、できる限り正しいApp Groupsを設定して、画面共有機能の安定性を確保してください。

手順1：App Groupsの作成

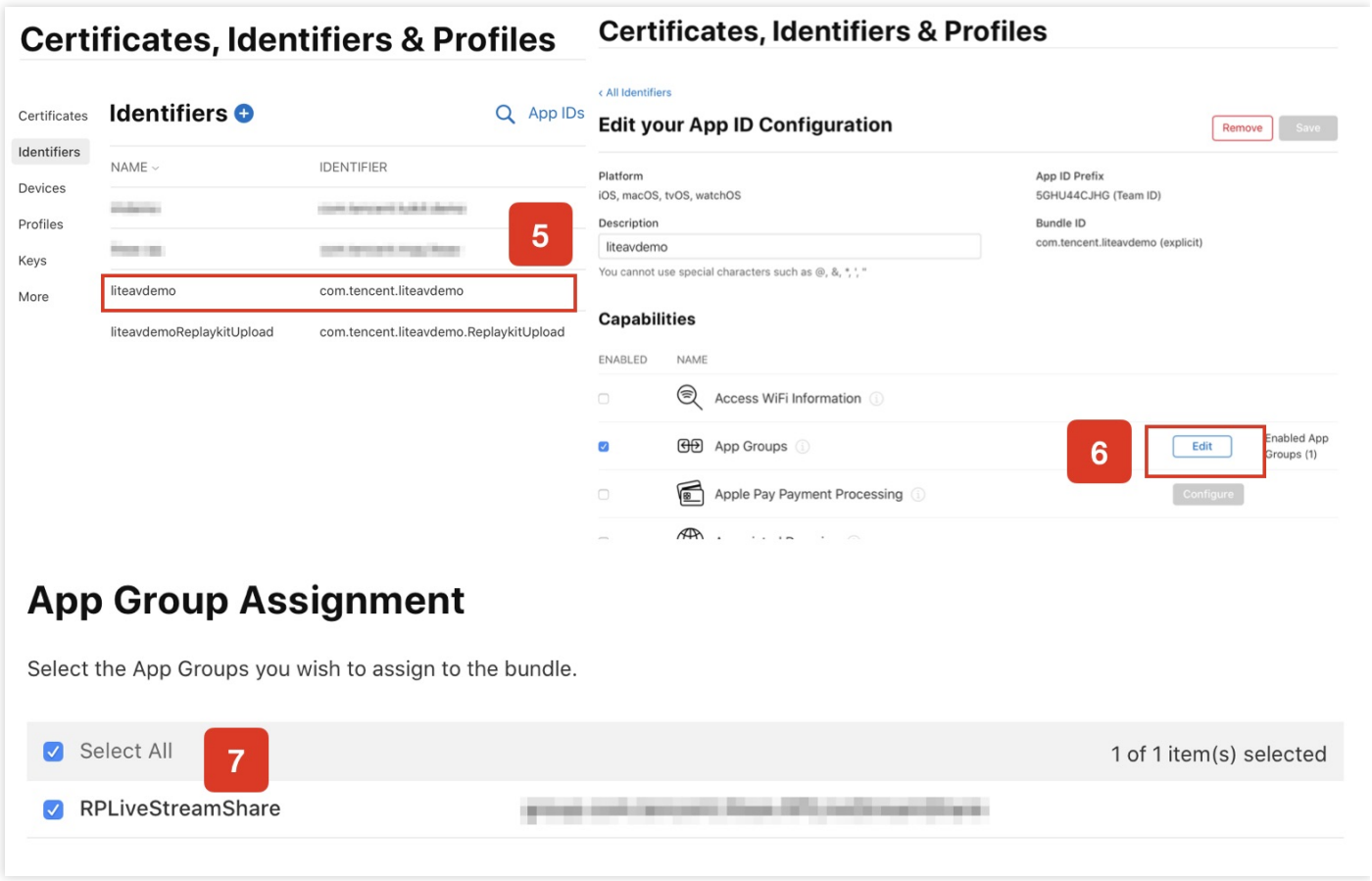
お客様のアカウントを使用して**https://developer.apple.com/**にログインし、次の操作を実行します。**完了後は、対応するProvisioning Profileを再ダウンロードする必要がありますので、ご注意ください。**

1. **Certificates, IDs & Profiles**をクリックします。
2. 右側のインターフェースでプラス記号をクリックします。
3. **App Groups**を選択し、**Continue**をクリックします。

4. ポップアップされたフォームにDescriptionとIdentifierを入力します。そのうちIdentifierは、インターフェースにおいて対応するAppGroupパラメータに渡す必要があります。完了したら、**Continue**をクリックします。



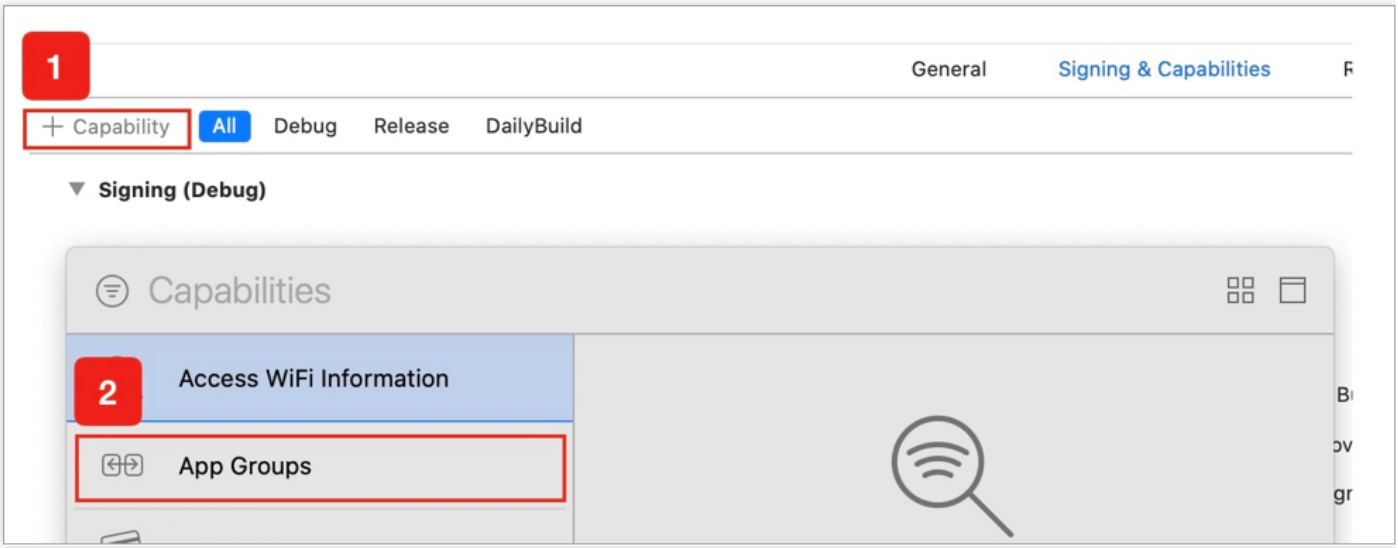
5. Identifierページに戻り、左上のメニューから**App IDs**を選択し、お客様のApp IDをクリックします（メインAppとExtensionのApp IDに同様の設定が必要です）。
6. **App Groups**を選択し、**Edit**をクリックします。
7. ポップアップされたフォームからこの前に作成したApp Groupを選択し、**Continue**をクリックして編集ページに戻り、**Save**をクリックして保存します。



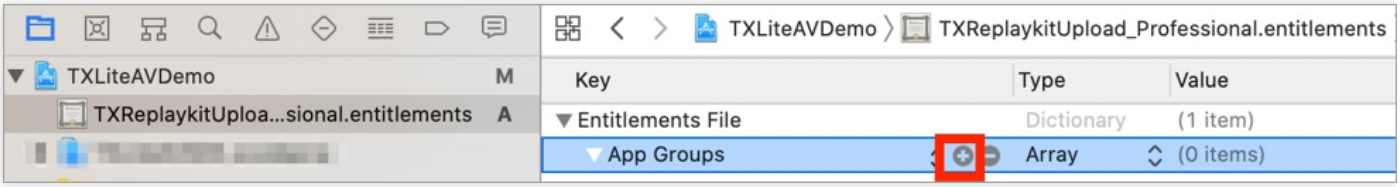
8. Provisioning Profileを再度ダウンロードし、XCodeに設定します。

手順2：Broadcast Upload Extensionの作成

1. Xcodeﾒﾆｭｰで、順に**File**、**New**、**Target...**をクリックし、Broadcast Upload Extensionを選択します。
2. ポップアップされたダイアログボックスに関連情報を入力し、**Include UI Extension**にチェックを**入れず**、**Finish**をクリックして作成を完了します。
3. ダウンロードしたSDK圧縮パッケージのプロジェクトにTXLiteAVSDK_ReplayKitExt.frameworkをドラッグし、先ほど作成したTargetにチェックを入れます。
4. 新しく追加したTargetを選択し、順に**+ Capabilityをクリックし、App Groups**をダブルクリックします。下図のとおりです：



操作が完了すると、下図のようにファイルリストに `Target名.entitlements` という名前のファイルが生成されます。このファイルを選択し、+記号をクリックして上記の手順のApp Groupを入力してください。



5. メインアプリのTargetを選択し、**上記の手順に従って、メインアプリのTargetに同様の処理を行います。**
6. 新規作成したTargetにおいて、Xcodeは「SampleHandler.h」という名前のファイルを自動的に作成し、それを次のコードに置き換えます。**コード内のAPPGROUPを先ほど作成したApp Group Identifierに変更してください。**

```
#import "SampleHandler.h"
@import TXLiteAVSDK_ReplayKitExt;

#define APPGROUP @"group.com.tencent.liteav.RPLiveStreamShare"

@interface SampleHandler() <txreplaykitextdelegate>

@end

@implementation SampleHandler
// 注意：ここでのAPPGROUPは、先ほど作成したApp Group Identifierに変更してください。
- (void)broadcastStartedWithSetupInfo:(NSDictionary<nsstring *,nsobject="" *="">
*)setupInfo {
[[TXReplayKitExt sharedInstance] setupWithAppGroup:APPGROUP delegate:self];
}
```

```
- (void)broadcastPaused {
// User has requested to pause the broadcast. Samples will stop being delivered.
}

- (void)broadcastResumed {
// User has requested to resume the broadcast. Samples delivery will resume.
}

- (void)broadcastFinished {
[[TXReplayKitExt sharedInstance] finishBroadcast];
// User has requested to finish the broadcast.
}

#pragma mark - TXReplayKitExtDelegate
- (void)broadcastFinished:(TXReplayKitExt *)broadcast reason:(TXReplayKitExtReas
on)reason
{
NSString *tip = @"";
switch (reason) {
case TXReplayKitExtReasonRequestedByMain:
tip = @"画面共有は終了しました";
break;
case TXReplayKitExtReasonDisconnected:
tip = @"アプリケーションは切断されました";
break;
case TXReplayKitExtReasonVersionMismatch:
tip = @"統合エラー（SDKバージョン番号が一致しません）";
break;
}

NSError *error = [NSError errorWithDomain:NSStringFromClass(self.class)
code:0
userInfo:@{
NSLocalizedFailureReasonErrorKey:tip
}];
[self finishBroadcastWithError:error];
}

- (void)processSampleBuffer:(CMSampleBufferRef)sampleBuffer withType:(RPSampleBu
fferType)sampleBufferType {
switch (sampleBufferType) {
case RPSampleBufferTypeVideo:
[[TXReplayKitExt sharedInstance] sendVideoSampleBuffer:sampleBuffer];
break;
case RPSampleBufferTypeAudioApp:
// Handle audio sample buffer for app audio
```

```
break;
case RPSampleBufferTypeAudioMic:
// Handle audio sample buffer for mic audio
break;

default:
break;
}
}
@end
```

手順3：メインアプリ側の受信ロジックとの結合

以下の手順に従って、メインアプリ側の受信ロジックと結合します。すなわち、ユーザーが画面共有をトリガーする前に、メインアプリを「待機」状態にして、Broadcast Upload Extensionプロセスからスクリーンキャプチャデータをいつでも受信できるようにします。

1. TRTCCloudがカメラのキャプチャをオフにしていることを確認します。オフになっていない場合は、stopLocalPreviewを呼び出して、カメラのキャプチャをオフにしてください。

2. startScreenCaptureByReplaykit:appGroup:メソッドを呼び出し、手順1で設定したAppGroupを渡して、SDKを「待機」状態にします。

3. ユーザーが画面共有をトリガーするまで待機します。手順4における「トリガーボタン」が実装されていない場合、ユーザーがiOSシステムのコントロールセンターにおいて、スクリーンキャプチャボタンを長押しして画面共有をトリガーしてください。この操作手順は下図のようになります：

4. stopScreenCaptureインターフェースを呼び出すことにより、いつでも画面共有を停止できます。

```
// 画面共有を開始するには、APPGROUPを上記の手順で作成したApp Group Identifierに置き換えてください。
- (void)startScreenCapture {
TRTCVideoEncParam *videoEncConfig = [[TRTCVideoEncParam alloc] init];
videoEncConfig.videoResolution = TRTCVideoResolution_1280_720;
videoEncConfig.videoFps = 10;
videoEncConfig.videoBitrate = 2000;
//APPGROUPを上記の手順で作成したApp Group Identifierに置き換えてください。
[[TRTCCloud sharedInstance] startScreenCaptureByReplaykit:videoEncConfig
appGroup:APPGROUP];
}

// 画面共有を停止します
- (void)stopScreenCapture {
```

```
[[TRTCCloud sharedInstance] stopScreenCapture];
}

// 画面共有の開始イベントの通知は、TRTCCloudDelegateを介して受信できます
- (void)onScreenCaptureStarted {
[self showTip:@"画面共有の開始"];
}
```

手順4：画面共有のトリガーボタンの追加（オプション）

手順3までに、ユーザーがコントロールセンターからスクリーンキャプチャボタンを長押しして、画面共有を手動で開始する必要があります。以下の方法で、VooVMeetingのように、ボタンをクリックすることでトリガーする効果を実現できます：

1. DemoのTRTCBroadcastExtensionLauncherファイルは、画面共有の起動を実装し、それを見つけてプロジェクトに追加します。
2. インターフェースにボタンを設置し、ボタンの応答関数において `TRTCBroadcastExtensionLauncher` の `launch` 関数を呼び出すと、画面共有機能を呼び出すことができます。

```
// カスタムボタンによる応答メソッド
- (IBAction)onScreenButtonTapped:(id)sender {
[TRTCBroadcastExtensionLauncher launch];
}
```

注意：

- AppleはiOS12.0に `RPSystemBroadcastPickerView` を追加しました。これによって、ユーザーがアプリケーションからランチャーをポップアップして、画面共有の開始を確認することができます。現時点では、 `RPSystemBroadcastPickerView` はインターフェースのカスタマイズをサポートしておらず、公式の呼び出しメソッドもありません。
- TRTCBroadcastExtensionLauncherの原理とは、 `RPSystemBroadcastPickerView` のサブViewをスキャンしてUIButtonを見つけ、そのクリックイベントをトリガーするというものです。
- **ただし、このソリューションはAppleで公式に推奨されているものではなく、システムが新たにアップデートされた場合、無効になる可能性があります。従って、手順4はオプションのソリューションにすぎず、お客様の自己責任でこのソリューションをご利用ください。**

画面共有の確認

- **Mac / Windows画面共有の確認**
  ルームにいるMac/Windowsユーザーが画面共有を起動し、サブストリームを介して共有を実行します。ルームにいるその他のユーザーは、TRTCCloudDelegate内のonUserSubStreamAvailableイベントを介してこの通知を受け取ります。
  画面共有を確認したいユーザーはstartRemoteSubStreamViewインターフェースを介してリモートユーザーのサブストリーム画面のレンダリングを起動することができます。

- **Android / iOS画面共有の確認**
  ユーザーがAndroid / iOSを介して画面共有を実行する場合は、メインストリームを介して共有を実行します。ルームにいるその他ユーザーはTRTCCloudDelegateの中のonUserVideoAvailableイベントを介してこの通知を受け取ります。
  画面共有を確認したいユーザーはstartRemoteViewインターフェースを介してリモートユーザーのメインストリーム画面のレンダリングを起動することができます。

Android

最終更新日：：2022-07-07 15:10:00

このドキュメントでは、主に画面共有の使用方法を紹介します。現在、TRTCオーディオビデオルームで使用できる画面共有は1つだけです。

呼び出しガイド

画面共有の有効化

手順1：Activityの追加

次のactivityをmanifestファイルに貼り付けます（項目コードに存在する場合は追加する必要はありません）。

```
<activity
android:name="com.tencent.rtmp.video.TXScreenCapture$TXScreenCaptureAssistantActi
vity"
android:theme="@android:style/Theme.Translucent"/>
```

手順2：画面共有の有効化

Android側での画面共有は、 `TRTCCloud` のstartScreenCapture()インターフェースを呼び出すだけで開始することができます。

startScreenCapture()で最初のパラメータ `encParams` を設定することにより、画面共有のエンコード品質を指定することができます。 `encParams` をnullに指定した場合、SDKは以前に設定されたエンコードパラメータを自動的に使用します。推奨されるパラメータの設定は、次のとおりです：

パラメータ項目	パラメータ名	通常の推奨値	テキスト教育シナリオ
解像度	videoResolution	1280 × 720	1920 × 1080
フレームレート	videoFps	10 FPS	8 FPS
最高ビットレート	videoBitrate	1600 kbps	2000 kbps
解像度アダプティブ	enableAdjustRes	NO	NO

- 画面共有されるコンテンツには通常、大幅な変更がなく、FPSを高く設定するのは経済的ではないため、10FPSを推奨します。

- 共有したい画面コンテンツに大量のテキストが含まれている場合、解像度とビットレートを適宜引き上げることができます。
- 最高ビットレート(videoBitrate)とは、画面が大きく変化したときの最高出力ビットレートのことです。画面コンテンツの変化が少ない場合、実際のエンコードビットレートは低くなります。

手順3：フローティングウィンドウをポップアップして、アプリケーションが強制終了されないようにします（オプション）

Android 7.0以降のシステムから、バックグラウンドで実行されている通常のアプリプロセスに切り替わりますが、CPUのアクティビティがある場合、常にシステムによって強制終了されやすくなります。従って、アプリをバックグラウンドに切り替えてサイレントで画面共有すると、フローティングウィンドウのポップアップというソリューションによって強制終了を回避することができます。また、携帯電話の画面にフローティングウィンドウを表示することは、ユーザーが今、画面共有を行っていることをユーザーに知らせ、プライバシー情報の漏えい防止につながります。

フローティングウィンドウをポップアップするには、FloatingView.javaの実装を参照してください：

```
public void showView(View view, int width, int height) {
mWindowManager = (WindowManager) mContext.getSystemService(Context.WINDOW_SERVIC
E);
int type = WindowManager.LayoutParams.TYPE_TOAST;
//TYPE_TOASTは、4.4以降のシステムにのみ適用できます。下位バージョンをサポートするには、TYPE_S
YSTEM_ALERTを使用します（manifestで権限を宣言してください）
//7.1（7.1を含む）以降のシステムは、TYPE_TOASTに対して制限を設けています
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
type = WindowManager.LayoutParams.TYPE_APPLICATION_OVERLAY;
} else if (Build.VERSION.SDK_INT > Build.VERSION_CODES.N) {
type = WindowManager.LayoutParams.TYPE_PHONE;
}
mLayoutParams = new WindowManager.LayoutParams(type);
mLayoutParams.flags = WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE;
mLayoutParams.flags |= WindowManager.LayoutParams.FLAG_WATCH_OUTSIDE_TOUCH;
mLayoutParams.width = width;
mLayoutParams.height = height;
mLayoutParams.format = PixelFormat.TRANSLUCENT;
mWindowManager.addView(view, mLayoutParams);
}
```

画面共有の確認

- ** Mac / Windows画面共有の確認**
  ルームにいるMac / Windowsユーザーが画面共有を開始する場合、サブストリームを介して共有を実行します。

ルームにいるその他のユーザーは、TRTCCloudListener内のonUserSubStreamAvailableイベントを介してこの通知を受け取ります。

- **Android / iOS画面共有の確認**
  ユーザーがAndroid / iOSを介して画面共有を開始する場合、ビッグストリームを介して共有を実行します。
  ルームにいるその他のユーザーは、TRTCCloudListener内のonUserVideoAvailableイベントを介してこの通知を受け取ります。

画面共有を確認したいユーザーはstartRemoteViewインターフェースを介してリモートユーザーのビッグストリーム画面のレンダリングを開始することができます。

リアルタイム画面共有(Mac)

最終更新日：：2022-06-08 16:00:49

Tencent Cloud TRTC は画面共有機能をサポートし、Macプラットフォームにおける画面共有はビッグストリーム共有とサブストリーム共有の2つのスキームをサポートします。

- **サブストリーム共有**
  TRTCでは、画面共有のための「サブストリーム（**substream**）」という1チャネルのアップストリームのビデオストリームを個別にスタートできます。サブストリーム共有は、キャスターがカメラ画面とスクリーン画面の両方を同時にアップロードします。これはTencentMeetingの使用スキームであり、`startScreenCapture` インターフェースを呼び出す場合に、`TRTCVideoStreamType` パラメータを `TRTCVideoStreamTypeSub` に指定して、このモードをイネーブルできます。このストリーム画面を閲覧するには専用の `startRemoteSubStreamView` インターフェースを使用する必要があります。

- **ビッグストリーム共有**
  TRTCでは、通常、カメラが動くチャネルを「ビッグストリーム（**bigstream**）」といい、ビッグストリーム共有は、カメラチャネルを使用して画面を共有します。このモードでは、キャスターは、アップストリームのカメラ画面、もしくはアップストリームのスクリーン画面のいずれかのアップストリームのビデオストリームを1チャンネルのみ有し、両者は相互に排他的です。`startScreenCapture` インターフェースを呼び出す場合に、`TRTCVideoStreamType` パラメータを `TRTCVideoStreamTypeBig` に指定し、このモードをイネーブルできます。

サポートするプラットフォーム

iOS	Android	Mac OS	Windows	Electron	Chromeブラウザ
✓	✓	✓	✓	✓	✓

共有ターゲットの取得

getScreenCaptureSourcesWithThumbnailSizeを介して共有可能なウィンドウをリストアップでき、共有可能な各ターゲットはいずれも `TRTCScreenCaptureSourceInfo` オブジェクトです。

Mac OS内のデスクトップスクリーンも共有可能なターゲットであり、通常のMacウィンドウのtypeは `TRTCScreenCaptureSourceTypeWindow` で、デスクトップスクリーンのtypeは `TRTCScreenCaptureSourceTypeScreen` です。

typeに加え、各 `TRTCScreenCaptureSourceInfo` には次のフィールド情報があります。

フィールド	タイプ	意味
type	TRTCScreenCaptureSourceType	キャプチャソースタイプ:指定タイプはウィンドウまたはスクリーン
sourceId	NSString	キャプチャソースID：ウィンドウの場合、このフィールドはウィンドウハンドルを示します； スクリーンの場合、このフィールドはスクリーンIDを示します
sourceName	NSString	ウィンドウ名、画面である場合は、Screen0 Screen1...を返します
extInfo	NSDictionary	共有ウィンドウの追加情報
thumbnail	NSImage	ウィンドウサムネイル
icon	NSImage	ウィンドウアイコン

上述のこれらの情報があれば、ユーザーの選択のために共有可能なターゲットを一覧表示するシンプルなリストページを実現できます。

共有ターゲットの選択

TRTC SDKは3種類の共有モードをサポートしており、selectScreenCaptureTargetを介して指定できます。

- **全画面の共有**：
  全スクリーンウィンドウの共有であり、マルチモニター分割画面をサポートします。1つのtypeが `TRTCScreenCaptureSourceTypeScreen` のscreenSourceパラメータを指定し、rectを{ 0, 0, 0, 0 }に設定する必要があります。

- **指定領域の共有**：
  スクリーンの特定領域の共有であり、ユーザーが領域の位置座標を決定する必要があります。1つのtypeが `TRTCScreenCaptureSourceTypeScreen` のscreenSourceパラメータを指定し、captureRectを{100、100、300、300}などの非NULLに設定する必要があります。

- **指定ウィンドウの共有**：
  ターゲットウィンドウのコンテンツを共有するには、ユーザーが共有したいウィンドウを指定する必要があり

ます。1つのtypeが `TRTCScreenCaptureSourceTypeWindow` のscreenSourceパラメータを指定し、captureRectを{ 0, 0, 0, 0 }に設定する必要があります。

説明：
2つの追加パラメータ：

- パラメータcapturesCursorはマウスポインタをキャプチャするかどうかを指定するために使用されます。
- パラメータhighlightは共有するウィンドウを強調表示するかどうかを指定し、キャプチャされた画像がブロックされた場合に、オクルージョンを移動するようユーザーに促すために使用されます。（この部分のUI特殊効果は、SDK内で実現されます）

画面共有の開始

- 共有ターゲットを選択した後、startScreenCaptureインターフェースを使用して画面共有を起動することができます。
- 2つの関数pauseScreenCaptureとstopScreenCaptureの違いは、pauseはスクリーンコンテンツのキャプチャを停止し、その瞬間のスクリーンパッドを一時停止するため、resumeScreenCaptureまで最後の画面がリモート側に表示され続けることです。

```
/**
* 7.6 【画面共有】画面共有の起動
* @param viewレンダリングウィジェットが配置される親ウィジェット
*/
- (void)startScreenCapture:(NSView *)view;
/**
* 7.7 【画面共有】画面キャプチャの停止
* @return 0：成功 <0:失敗
*/
- (int)stopScreenCapture;
/**
* 7.8 【画面共有】画面共有の一時停止
* @return 0：成功 <0:失敗
*/
- (int)pauseScreenCapture;
/**
* 7.9 【画面共有】画面共有のリカバー
*
* @return 0：成功 <0:失敗
```

```
*/
- (int)resumeScreenCapture;
```

画質の設定

setSubStreamEncoderParamインターフェースを介して解像度、ビットレートとフレームレートを含む画面共有の画面品質を設定できます。推奨する参考値を以下に提示します。

解像度レベル	解像度	フレームレート	ビットレート
超高精細（HD+）	1920 × 1080	10	800kbps
高精細（HD）	1280 × 720	10	600kbps
標準（SD）	960 × 720	10	400kbps

画面共有の確認

- **Mac/Windows画面共有の確認**
  ルームにいるMac/Windowsユーザーが画面共有を起動し、サブストリームを介して共有を実行します。ルームにいるその他のユーザーは、TRTCCloudDelegate内のonUserSubStreamAvailableイベントを介してこの通知を受け取ります。
  画面共有を確認したいユーザーはstartRemoteSubStreamViewインターフェースを介して、リモートユーザーのサブストリーム画面のレンダリングを起動することができます。

- **Android/iOS画面共有の確認**
  ユーザーがAndroid / iOSを介して画面共有を実行する場合は、メインストリームを介して共有を実行します。ルームにいるその他ユーザーはTRTCCloudDelegateの中のonUserVideoAvailableイベントを介してこの通知を受け取ります。
  画面共有を確認したいユーザーはstartRemoteViewインターフェースを介して、リモートユーザーのメインストリーム画面のレンダリングを起動することができます。

```
//サンプルコード：画面共有の画面を見る
- (void)onUserSubStreamAvailable:(NSString *)userId available:(BOOL)available {
if (available) {
[self.trtcCloud startRemoteSubStreamView:userId view:self.capturePreviewWindow.co
ntentView];
} else {
```

```
[self.trtcCloud stopRemoteSubStreamView:userId];
}
}
```

よくあるご質問

1つのルームで同時にいくつの画面を共有できますか。
現在、1つのTRTCオーディオ・ビデオルームで共有できる画面は1つだけです。

ウィンドウの共有（SourceTypeWindow）を指定し、ウィンドウのサイズが変化した場合は、ビデオストリームの解像度も変化しますか。
デフォルトでは、SDK内で共有ウィンドウのサイズに従ってエンコーディングパラメータを自動的に調整します。
解像度を固定する必要がある場合は、setSubStreamEncoderParamインターフェースを呼び出し画面共有のエンコーディングパラメータを設定するか、またはstartScreenCaptureを呼び出すときに、対応するエンコーディングパラメータを指定する必要があります。

Web

最終更新日：：2022-08-08 15:25:08

TRTC Web SDKの画面共有のサポート状況については、ブラウザサポート状況をご覧ください。また、SDK も TRTC.isScreenShareSupportedインターフェースを提供し、現在のブラウザが画面共有をサポートしているかどうかを判断できるようにしています。
ここでは、シーンごとの実装プロセスについてご説明します。

注意：

- Web端末では現時点ではサブストリームの公開をサポートしておらず、画面共有の公開はメインストリームの公開によって行います。リモートの画面共有ストリームがWebユーザーからの場合、RemoteStream.getType()の戻り値は'main'となります。通常は、それがWeb端末からの画面共有ストリームであることはuserIdによって識別します。
- Native（iOS、Android、Mac、Windowsなど）端末ではサブストリームの公開をサポートしており、画面共有の公開はサブストリームの公開によって行います。リモートの画面共有ストリームがNativeユーザーからの場合、RemoteStream.getType()の戻り値は'auxiliary'となります。

画面共有ストリームの作成とリリース

ステップ1：画面共有ストリームの作成

画面共有ストリームにはビデオストリームとオーディオストリームが含まれます。そのうちオーディオストリームはマイクオーディオとシステムオーディオに分けられます。

```
// 通常は、userIdにプレフィックス`share_`を加え、これが画面共有用のクライアントオブジェクトだということを識別するために用いることが推奨されます。
const userId = 'share_userId';
const roomId = 'roomId';
// good 正しい用法
// 画面のビデオストリームのみをキャプチャ
const shareStream = TRTC.createStream({ audio: false, screen: true, userId });
// or マイクオーディオおよび画面のビデオストリームをキャプチャ
const shareStream = TRTC.createStream({ audio: true, screen: true, userId });
// or システムオーディオおよび画面のビデオストリームをキャプチャ
const shareStream = TRTC.createStream({ screenAudio: true, screen: true, userId
});
// bad 間違った用法
const shareStream = TRTC.createStream({ camera: true, screen: true });
```

```
// or
const shareStream = TRTC.createStream({ camera: true, screenAudio: true });
```

注意：

- audioとscreenAudioの属性を同時にtrueに設定することはできず、cameraとscreenAudioの属性を同時にtrueに設定することはできません。screenAudioについてのその他の情報は、このドキュメントのパート5でご説明します。
- cameraとscreenの属性を同時にtrueに設定することはできません。

ステップ2：画面共有ストリームの初期化

初期化の際、ブラウザはユーザーに対し画面共有の内容と権限をリクエストします。ユーザーが権限承認を拒否した場合、またはシステムがブラウザに画面共有の権限を与えなかった場合、コードは `NotReadableError` または `NotAllowedError` のエラーをキャッチします。このとき、ブラウザの設定またはシステムの設定を行って画面共有の権限を有効化し、画面共有ストリームの初期化を再度行うよう、ユーザーに対して促す必要があります。

注意：
Safariの制限により、 画面共有ストリームの初期化操作は必ずクリックしたイベントのコールバック内で完了しなければなりません。この問題についての詳細な説明は、よくあるご質問をご参照ください

```
try {
await shareStream.initialize();
} catch (error) {
// 画面共有ストリームの初期化に失敗したとき、ユーザーに通知してその後の入室と公開のプロセスを停止させる
switch (error.name) {
case 'NotReadableError':
// ユーザーに通知して、システムが現在のブラウザによる画面内容取得を許可するよう確認させる
return;
case 'NotAllowedError':
if (error.message.includes('Permission denied by system')) {
// ユーザーに通知して、システムが現在のブラウザによる画面内容取得を許可するよう確認させる
} else {
// ユーザーが画面共有を拒否/キャンセル
}
return;
default:
```

```
// 画面共有ストリームの初期化の際の不明なエラーが発生したため、リトライするようユーザーに通知する
return;
}
}
```

ステップ3：画面共有を担うクライアントオブジェクトの作成

通常は、userIdにプレフィックス `share_` を加え、これが画面共有用のクライアントオブジェクトだということを識別するために用いることが推奨されます。

```
const shareClient = TRTC.createClient({
mode: 'rtc',
sdkAppId,
userId, // 'share_teacher'など
userSig
});
// クライアントオブジェクトの入室
try {
await shareClient.join({ roomId });
// ShareClient join room success
} catch (error) {
// ShareClient join room failed
}
```

ステップ4：画面共有ストリームの公開

ステップ1で作成したクライアントオブジェクトを通じて公開します。公開に成功すると、画面共有ストリームをリモートで受信できます。

```
try {
await shareClient.publish(shareStream);
} catch (error) {
// ShareClient failed to publish local stream
}
```

完全なコード

```
// 通常は、userIdにプレフィックス`share_`を加え、これが画面共有用のクライアントオブジェクトだということを識別するために用いることが推奨されます。
const userId = 'share_userId';
const roomId = 'roomId';
// 画面のビデオストリームのみをキャプチャ
const shareStream = TRTC.createStream({ audio: false, screen: true, userId });
```

```
// or マイクオーディオおよび画面のビデオストリームをキャプチャ
// const shareStream = TRTC.createStream({ audio: true, screen: true, userId });
// or システムオーディオおよび画面のビデオストリームをキャプチャ
// const shareStream = TRTC.createStream({ screenAudio: true, screen: true, userI
d });
try {
await shareStream.initialize();
} catch (error) {
// 画面共有ストリームの初期化に失敗したとき、ユーザーに通知してその後の入室と公開のプロセスを停
止させる
switch (error.name) {
case 'NotReadableError':
// ユーザーに通知して、システムが現在のブラウザによる画面内容取得を許可するよう確認させる
return;
case 'NotAllowedError':
if (error.message.includes('Permission denied by system')) {
// ユーザーに通知して、システムが現在のブラウザによる画面内容取得を許可するよう確認させる
} else {
// ユーザーが画面共有を拒否/キャンセル
}
return;
default:
// 画面共有ストリームの初期化の際の不明なエラーが発生したため、リトライするようユーザーに通知す
る
return;
}
}
const shareClient = TRTC.createClient({
mode: 'rtc',
sdkAppId,
userId, // 'share_teacher'など
userSig
});
// クライアントオブジェクトの入室
try {
await shareClient.join({ roomId });
// ShareClient join room success
} catch (error) {
// ShareClient join room failed
}
try {
await shareClient.publish(shareStream);
} catch (error) {
// ShareClient failed to publish local stream
}
```

画面共有パラメータ設定

設定可能なパラメータには解像度、フレームレート、ビットレートがあります。必要に応じ、setScreenProfile()インターフェースでprofileを指定することができ、各profileは1組の解像度、フレームレート、ビットレートに対応します。SDKはデフォルトで'1080p'の設定を使用します。

```
const shareStream = TRTC.createStream({ audio: false, screen: true, userId });
// setScreenProfile()はinitialize()の前に呼び出す必要があります。
shareStream.setScreenProfile('1080p');
await shareStream.initialize();
```

あるいは、カスタマイズした解像度、フレームレート、ビットレートを使用します。

```
const shareStream = TRTC.createStream({ audio: false, screen: true, userId });
// setScreenProfile()はinitialize()の前に呼び出す必要があります。
shareStream.setScreenProfile({ width: 1920, height: 1080, frameRate: 5, bitrate:
1600 /* kbps */});
await shareStream.initialize();
```

画面共有属性推奨リスト：

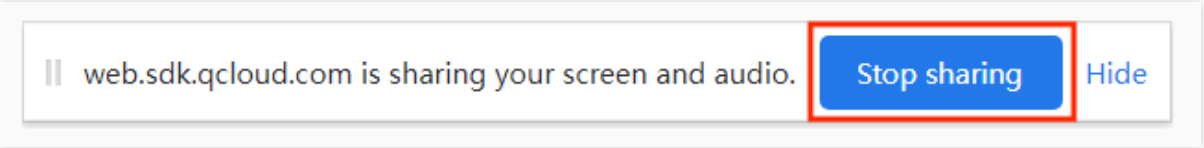
profile	解像度（幅 x 高）	フレームレート（fps）	ビットレート (kbps)
480p	640 x 480	5	900
480p_2	640 x 480	30	1000
720p	1280 x 720	5	1200
720p_2	1280 x 720	30	3000
1080p	1920 x 1080	5	1600
1080p_2	1920 x 1080	30	4000

注意：
高すぎるパラメータを設定することで予測不可能な問題が発生することを避けるため、推奨するパラメータに従って設定することをお勧めします。

画面共有の停止

```
// 画面共有クライアントがストリーム公開をキャンセルする
await shareClient.unpublish(shareStream);
// 画面共有ストリームを閉じる
shareStream.close();
// 画面共有クライアントが退室
await shareClient.leave();
// 上記の3ステップは必須ではなく、シーンのニーズに応じて必要なコードを実行します。通常は、入室済みかどうか、ストリームを公開済みかどうかの判断を追加する必要があります。より詳細なコードの例については、[demoソースコード](https://github.com/LiteAVSDK/TRTC_Web/blob/main/base-js/js/share-client.js)をご参照ください。
```

また、ユーザーがブラウザ付属のボタンで画面共有を停止する可能性もあるため、画面共有ストリームでは画面共有停止イベントを監視し、それに応じた処理を行う必要があります。




```
// 画面共有ストリームの画面共有停止イベント監視
shareStream.on('screen-sharing-stopped', event => {
// 画面共有クライアントがプッシュを停止する
await shareClient.unpublish(shareStream);
// 画面共有ストリームを閉じる
shareStream.close();
// 画面共有クライアントが退室
await shareClient.leave();
});
```

カメラからのビデオと画面共有の同時公開

1つのClientは1つのオーディオと1つのビデオまでしか同時に公開できません。カメラからのビデオと画面共有を同時に公開する場合は、 2つのClientを作成し、それぞれの用途に用いる必要があります。
例えば2つのClientを作成し、それぞれ次のようにします。

- **client**：ローカルオーディオビデオストリーミングの公開を担い、shareClient以外のすべてのリモートストリームを閲覧します。
- **shareClient**：画面共有ストリームの公開を担い、どのリモートストリームも閲覧しません。

注意：

- 画面共有を担うshareClientでは自動サブスクリプションを無効にしておかなければ、リモートストリームのサブスクリプションの重複が生じます。APIの説明をご参照ください。
- ローカルオーディオビデオストリーミングの公開を担うclientでは、shareClientが公開するストリームのサブスクリプションをキャンセルする必要があります。そうしなければ、自分で自分を閲覧するという状況が発生してしまいます。

サンプルコード：

```
const client = TRTC.createClient({ mode: 'rtc', sdkAppId, userId, userSig });
// shareClientでリモートストリームの自動サブスクリプションを無効化、すなわちautoSubscribe: falseとする必要があります
const shareClient = TRTC.createClient({ mode: 'rtc', sdkAppId, `share_${userId}`, userSig, autoSubscribe: false,});
// ローカルオーディオビデオストリーミングの公開を担うclientでは、shareClientが公開するストリームのサブスクリプションをキャンセルする必要があります。
client.on('stream-added', event => {
const remoteStream = event.stream;
const remoteUserId = remoteStream.getUserId();
if (remoteUserId === `share_${userId}`) {
// 自身の画面共有ストリームのサブスクリプションをキャンセル
client.unsubscribe(remoteStream);
} else {
//その他一般的なリモートストリームのサブスクリプション
client.subscribe(remoteStream);
}
});
await client.join({ roomId });
await shareClient.join({ roomId });
const localStream = TRTC.createStream({ audio: true, video: true, userId });
const shareStream = TRTC.createStream({ audio: false, screen: true, userId });
// ... 初期化および関連コードの公開を省略し、必要に応じてストリームを公開します。
```

画面共有のシステム音声キャプチャ

システム音声キャプチャはChrome M74+のみサポートしています。WindowsおよびChrome OS上ではシステム全体のオーディオをキャプチャすることができ、LinuxおよびMacではオプションタブのオーディオのみキャプチャできます。その他のChromeのバージョン、その他のシステム、その他のブラウザはいずれもサポートしていません。

```
// 画面共有ストリームを作成する際、screenAudioはtrueに設定してください、システム音声とマイク音声の同時キャプチャはサポートしていないため、audio属性を同時にtrueに設定しないでください
const shareStream = TRTC.createStream({ screenAudio: true, screen: true, userId });
await shareStream.initialize();
...
```

ポップアップしたダイアログボックスで「オーディオ共有」にチェックを入れると、公開されるstreamにはシステム音声が入ります。

よくあるご質問

1. **Safariの画面共有で `getDisplayMedia must be called from a user gesture handler` というエラーが表示されます。**
   これは、Safariが `getDisplayMedia` スクリーンキャプチャインターフェースを制限しているためであり、ユーザーがイベントのコールバック関数をクリックして実行してから1秒以内でなければ呼び出すことができません。詳細については、webkit issueをご参照ください。

```
// good
async function onClick() {
// onClickで実行する場合は、先にキャプチャロジックを実行することをお勧めします
const screenStream = TRTC.createStream({ screen: true });
await screenStream.initialize();
await client.join({ roomId: 123123 });
}
// bad
async function onClick() {
await client.join({ roomId: 123123 });
// 入室には1秒以上かかることがあり、キャプチャに失敗する可能性があります
const screenStream = TRTC.createStream({ screen: true });
await screenStream.initialize();
}
```

2. **Mac Chromeで、スクリーンレコーディングの権限が得られている状況で画面共有に失敗し、"NotAllowedError: Permission denied by system"または"NotReadableError: Could not start video source"というエラーメッセージが表示されましたが、Chrome bugですか。**
   対処方法：**設定**> **セキュリティとプライバシー**をクリック> **プライバシー**をクリック> **スクリーンレコーディング**をクリック > Chromeのスクリーンレコーディング権限を無効化 > Chromeのスクリーンレコーディング権限を再度有効化 > Chromeブラウザを無効化 > Chromeブラウザを再度有効化。

3. **WebRTC画面共有の既知の問題と回避策**

4. **ElectronによるTRTC Web SDK画面共有の使用**

5. **ユーザーが選択した画面共有のタイプの判断：画面全体、ウィンドウ、Chromeタブ。**

```
// 画面共有キャプチャの成功後
const shareStream = TRTC.createStream({ screenAudio: true, screen: true, userId
});
await shareStream.initialize();
// displaySurfaceによってキャプチャのタイプを判断します。
const { displaySurface } = shareStream.getVideoTrack().getSettings();
// 例えば、monitorは画面全体、windowはあるアプリケーションのウィンドウ、browserはChromeの
あるタブ
```

詳細については、displaySurfaceをご参照ください。

Windows

最終更新日：：2022-10-14 17:18:04

このドキュメントでは、主に画面共有の使用方法を紹介します。現在、TRTCオーディオビデオルームで使用できる画面共有は1つだけです。

Windowsプラットフォームでの画面共有は、ビッグストリーム共有とサブストリーム共有の2つのスキームをサポートます。

- **サブストリーム共有**
  TRTCでは、画面共有のための「サブストリーム(**substream**)」という1チャネルのアップリンクのビデオストリームを個別にスタートできます。サブストリーム共有は、キャスターがカメラ画面とスクリーン画面の両方を同時にアップロードします。これはTencentMeetingの使用スキームであり、`startScreenCapture` インターフェースを呼び出す場合に、`TRTCVideoStreamType` パラメータを `TRTCVideoStreamTypeSub` に指定して、このモードを有効にすることができます。

- **ビッグストリーム共有**
  TRTCでは、通常、カメラが動くチャネルを「ビッグストリーム(**bigstream**)」といい、ビッグストリーム共有は、カメラチャネルを使用して画面を共有します。このモードでは、キャスターは、アップストリームのカメラ画面、もしくはアップストリームのスクリーン画面のいずれかのアップストリームのビデオストリームを1チャンネルのみ有し、両者は相互に排他的です。 `startScreenCapture` インターフェースを呼び出す場合に、 `TRTCVideoStreamType` パラメータを `TRTCVideoStreamTypeBig` に指定し、このモードをイネーブルできます。

依存するAPI

API機能	C++バージョン	C#バージョン	Electronバージョン
共有ターゲットの選択	selectScreenCaptureTarget	selectScreenCaptureTarget	selectScreenCaptureTarget
画面共有を開始	startScreenCapture	startScreenCapture	startScreenCapture
画面共有の一時停止	pauseScreenCapture	pauseScreenCapture	pauseScreenCapture
画面共有のリカバー	resumeScreenCapture	resumeScreenCapture	resumeScreenCapture

API機能	C++バージョン	C#バージョン	Electronバージョン
画面共有の終了	stopScreenCapture	stopScreenCapture	stopScreenCapture

共有ターゲットの取得

`getScreenCaptureSources` を介して 共有可能なウィンドウのリストをリストアップでき、リストは出力パラメータ sourceInfoList を介して戻されます。

> 説明：
> Windowsのデスクトップ画面もウィンドウの1つであり、デスクトップウィンドウ（Desktop）と呼ばれ、モニターが2台ある場合は、各モニターに対応するデスクトップウィンドウがあります。したがって、getScreenCaptureSourcesを介して返されるウィンドウリストにもDesktopウィンドウがあります。

取得したウィンドウ情報に基づき、ユーザーの選択のために共有できるターゲットを一覧表示するシンプルなリストページを実現できます：

画面共有の開始

- 共有ターゲットを選択した後、 `startScreenCapture` インターフェースを使用して画面共有を起動することができます。
- 共有プロセスにおいても、 `selectScreenCaptureTarget` を呼び出し、共有ターゲットを変更することができます。
- `pauseScreenCapture` と `stopScreenCapture` の違いは、pauseはスクリーンコンテンツのキャプチャを停止し、その瞬間の画面を一時停止するため、resumeするまで最後の画面がリモート側に表示され続けます。

画質の設定

`setSubStreamEncoderParam` インターフェースを介して解像度、ビットレートとフレームレートを含む画面共有の画面品質を設定できます。推奨する参考値を以下に提示します：

解像度レベル	解像度	フレームレート	ビットレート

解像度レベル	解像度	フレームレート	ビットレート
超高精細（HD+）	1920 × 1080	10	800kbps
高精細（HD）	1280 × 720	10	600kbps
標準（SD）	960 × 720	10	400kbps

画面共有の確認

ルームにいるユーザーが画面共有を起動し、サブストリームを介して共有を実行します。ルームにいるその他のユーザーは、TRTCCloudDelegate内のonUserSubStreamAvailableイベントを介してこの通知を受け取ります。画面共有を視聴したいユーザーはstartRemoteViewインターフェースを介してリモートユーザーのサブストリーム画面のレンダリングを起動することができます。

```
//サンプルコード：画面共有の画面を見る
void CTRTCCloudSDK::onUserSubStreamAvailable(const char * userId, bool available)
{
LINFO(L"onUserSubStreamAvailable userId[%s] available[%d]\n", UTF82Wide(userId).c
_str(), available);
liteav::ITRTCCloud* trtc_cloud_ = getTRTCShareInstance();
if (available) {
trtc_cloud_->startRemoteView(userId, liteav::TRTCVideoStreamTypeSub, hWnd);
}else{
trtc_cloud_->stopRemoteView(userId, liteav::TRTCVideoStreamTypeSub);
}
}
```

よくあるご質問

1つのルームで同時に複数の画面を共有できますか？

現在、1つのTRTCオーディオ・ビデオルームで共有できる画面は1つだけです。

ウィンドウの共有(SourceTypeWindow)を指定し、ウィンドウのサイズが変化した場合は、ビデオストリームの解像度も変化しますか？

デフォルトでは、SDK内で共有ウィンドウのサイズに従ってエンコーディングパラメータを自動的に調整します。
解像度を固定する必要がある場合は、setSubStreamEncoderParamインターフェースを呼び出し画面共有のエン

コーディングパラメータを設定するか、またはstartScreenCaptureを呼び出すときに、対応するエンコーディングパラメータを指定してください。

Electron

最終更新日：：2022-07-21 16:57:38

このドキュメントでは、主に画面共有の使用方法を紹介します。現在、TRTCオーディオビデオルームで使用できる画面共有は1つだけです。

Electronプラットフォームでの画面共有は、ビッグストリーム共有とサブストリーム共有の2つのスキームをサポートます：

- **サブストリーム共有**
  TRTCでは、画面共有のための「サブストリーム（**substream**）」という1チャネルのアップリンクのビデオストリームを個別にスタートできます。サブストリーム共有は、キャスターがカメラ画面とスクリーン画面の両方を同時にアップロードします。これはTencentMeetingの使用スキームであり、`startScreenCapture` インターフェースを呼び出す場合に、`TRTCVideoStreamType` パラメータを `TRTCVideoStreamTypeSub` に指定して、このモードを有効にすることができます。

- **ビッグストリーム共有**
  TRTCでは、通常、カメラが動くチャネルを「ビッグストリーム（**bigstream**）」といい、ビッグストリーム共有は、カメラチャネルを使用して画面を共有します。このモードでは、キャスターは、アップストリームのカメラ画面、もしくはアップストリームのスクリーン画面のいずれかのアップストリームのビデオストリームを1チャンネルのみ有し、両者は相互に排他的です。`startScreenCapture` インターフェースを呼び出す場合に、`TRTCVideoStreamType` パラメータを `TRTCVideoStreamTypeBig` に指定し、このモードをイネーブルできます。

手順1：共有ターゲットの取得

`getScreenCaptureSources` を介して 共有可能なウィンドウのリストをリストアップでき、リストは出力パラメータ sourceInfoList を介して戻されます。

> 説明：
> Electronのデスクトップ画面もウィンドウの1つであり、デスクトップウィンドウ（Desktop）と呼ばれ、モニターが2台ある場合は、各モニターに対応するデスクトップウィンドウがあります。したがって、getScreenCaptureSourcesを介して返されるウィンドウリストにもDesktopウィンドウがあります。

取得したウィンドウ情報に基づき、ユーザーの選択のために共有できるターゲットを一覧表示するシンプルなリストページを実現できます：

```
import TRTCCloud from 'trtc-electron-sdk';
const rtcCloud = new TRTCCloud();
// https://web.sdk.qcloud.com/trtc/electron/doc/en-us/trtc_electron_sdk/TRTCCloud.html#getScreenCaptureSources
const screenList = rtcCloud.getScreenCaptureSources();
```

手順2：画面共有の開始

- selectScreenCaptureTargetを呼び出すことにより、共有ターゲットを選択できます。
- 共有ターゲットを選択すると、startScreenCaptureインターフェースを使用して画面共有を開始できます。
- 共有プロセスにおいても、selectScreenCaptureTargetを呼び出すことにより、共有ターゲットを変更できます。
- pauseScreenCaptureとstopScreenCaptureの区別は、pauseが画面コンテンツのキャプチャを停止し、その時点の画像スペーサーを一時停止するため、リモート側に表示される画像は、resumeまで常に最後のフレームになります。

```
import TRTCCloud, {
Rect, TRTCScreenCaptureProperty, TRTCVideoStreamType, TRTCVideoEncParam,
TRTCVideoResolution, TRTCVideoResolutionMode
} from 'trtc-electron-sdk';
const rtcCloud = new TRTCCloud();
// https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/TRTCCloud.html#getScreenCaptureSources
const screenList = rtcCloud.getScreenCaptureSources();
// https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/Rect.html
const captureRect = new Rect(0, 0, 0, 0);
// https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/TRTCScreenCaptureProperty.html
const property = new TRTCScreenCaptureProperty(
true, true, true, 0, 0, false
);
if (screenList.length > 0) {
rtcCloud.selectScreenCaptureTarget(screenList[0], captureRect, property)
}
const screenshareDom = document.querySelector('screen-dom');
// https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/TRTCVideoEncParam.html
const encParam = new TRTCVideoEncParam(
```

```
TRTCVideoResolution.TRTCVideoResolution_1920_1080,
TRTCVideoResolutionMode.TRTCVideoResolutionModeLandscape,
15,
2000,
0,
false
);
rtcCloud.startScreenCapture(screenshareDom, TRTCVideoStreamType.TRTCVideoStreamTy
peSub, encParam);
```

手順3：画面品質の設定

`startScreenCapture` インターフェースの3番目のパラメータ `encParam` を介して、 解像度、ビットレート、フレームレートを含む画面共有の画質を設定できます（手順2を参照）。以下の推奨基準値を提供します：

解像度レベル	解像度	フレームレート	ビットレート
超高精細（HD+）	1920 × 1080	10	2000kbps
高精細（HD）	1280 × 720	10	600kbps
標準（SD）	960 × 720	10	400kbps

手順4：画面共有の確認

ルームにいるユーザーが画面共有を起動し、サブストリームを介して共有を実行します。ルームにいるその他のユーザーは、onUserSubStreamAvailableイベントを介してこの通知を受け取ります。
画面共有を視聴したいユーザーはstartRemoteViewインターフェースを介してリモートユーザーのサブストリーム画面のレンダリングを起動することができます。

```
import TRTCCloud, {
TRTCVideoStreamType
} from 'trtc-electron-sdk';
const rtcCloud = new TRTCCloud();
const remoteDom = document.querySelector('.remote-user');
function onUserSubStreamAvailable(userId, available) {
if (available === 1) {
rtcCloud.startRemoteView(userId, remoteDom, TRTCVideoStreamType.TRTCVideoStreamTy
peSub);
}else{
```

```
rtcCloud.stopRemoteView(userId, TRTCVideoStreamType.TRTCVideoStreamTypeSub);
}
}
rtcCloud.on('onUserSubStreamAvailable', onUserSubStreamAvailable);
```

よくある質問

1. 1つのルームで同時に複数の画面を共有できますか？

現在、1つのTRTCオーディオ・ビデオルームで共有できる画面は1つだけです。

2. ウィンドウの共有(SourceTypeWindow)を指定し、ウィンドウのサイズが変化した場合は、ビデオストリームの解像度も変化しますか？

デフォルトでは、SDK内で共有ウィンドウのサイズに従ってエンコーディングパラメータを自動的に調整します。
解像度を固定する必要がある場合は、setSubStreamEncoderParamインターフェースを呼び出し画面共有のエンコーディングパラメータを設定するか、またはstartScreenCaptureを呼び出すときに、対応するエンコーディングパラメータを指定してください。

リアルタイム画面共有(Flutter)

最終更新日：：2022-09-13 10:50:15

Androidプラットフォームの場合

Tencent CloudのTRTCは、Androidシステムでの画面共有をサポートしており、現在のシステムの画面コンテンツは、TRTC SDKを介してルーム内の他のユーザーと共有されます。この機能については、次のような2つの注意点があります。

- TRTC Android版の画面共有は、デスクトップ版のように「サブチャネルの共有」をサポートしていません。従って、画面共有を開始するときは、あらかじめカメラのキャプチャを停止する必要があります。停止しない場合、衝突が生じます。
- Androidシステムのバックグラウンドアプリが引き続きCPUを使用すると、システムによって強制終了されやすくなり、画面共有自体が必然的にCPUを消費してしまいます。この矛盾するような衝突を解消するには、アプリで画面共有を開始するとともに、Androidシステムにフローティングウィンドウをポップアップする必要があります。Androidはフォアグラウンドを含むアプリプロセスを強制終了しないため、このソリューションによって、お客様のアプリはシステムに自動回収されることなく画面を共有し続けることができます。

画面共有を開始

Androidデバイスでの画面共有は、 `TRTCCloud` のstartScreenCapture()インターフェースを呼び出すだけで開始することができます。ただし、明瞭で安定した共有効果を実現するには、次の3つの点に注意する必要があります。

Activityの追加

次のactivityをmanifestファイルに貼り付けます（項目コードに存在する場合は追加する必要はありません）。

```
<activity
android:name="com.tencent.rtmp.video.TXScreenCapture$TXScreenCaptureAssistantActi
vity"
android:theme="@android:style/Theme.Translucent"/>
```

ビデオコーデックパラメータの設定

startScreenCapture()で最初のパラメータ `encParams` を設定することにより、画面共有のエンコード品質を指定することができます。 `encParams` をnullに指定した場合、SDKは以前に設定されたエンコードパラメータを自動的に使用します。推奨されるパラメータの設定は、次のとおりです。

パラメータ項目	パラメータ名	通常の推奨値	テキスト教育シナリオ

パラメータ項目	パラメータ名	通常の推奨値	テキスト教育シナリオ
解像度	videoResolution	1280 × 720	1920 × 1080
フレームレート	videoFps	10 FPS	8 FPS
最高ビットレート	videoBitrate	1600 kbps	2000 kbps
解像度アダプティブ	enableAdjustRes	NO	NO

説明：

- 画面共有されるコンテンツには通常、大幅な変更がなく、FPSを高く設定するのは経済的ではないため、10FPSを推奨します。
- 共有したい画面コンテンツに大量のテキストが含まれている場合、解像度とビットレートを適宜引き上げることができます。
- 最高ビットレート(videoBitrate)とは、画面が大きく変化したときの最高出力ビットレートのことです。画面コンテンツの変化が少ない場合、実際のエンコードビットレートは低くなります。

iOSプラットフォームの場合

- **アプリケーション内の共有**
  現在のアプリの画面のみを共有できます。この特性をサポートするには、iOSのバージョン13以降のOSが必要です。現在のアプリ以外の画面コンテンツを共有できないため、高度なプライバシー保護が必要なシーンに適しています。
- **アプリケーション間共有**
  AppleのReplaykitソリューションをベースとして、システム全体の画面コンテンツを共有できます。ただし、現在のアプリはExtension拡張コンポーネントを追加で提供する必要があるため、結合手順はアプリ内共有よりもやや多くなります。

注意：
注意すべき点としては、モバイル版のTRTC SDKは、デスクトップ版のように、「サブストリームの共有」をサポートしていません。これは、iOSとAndroidシステムは、どちらもバックグラウンドで実行されているアプリに対し、カメラの使用権を制限しているからです。従って、サブストリームの共有をサポートする意義はあまりありません。

方法1：iOSプラットフォームアプリケーション内の共有

アプリ内共有のソリューションは非常にシンプルです。TRTC SDKが提供するインターフェース startScreenCaptureを呼び出し、エンコードパラメータ `TRTCVideoEncParam` と `appGroup` を渡して `''` にセットするだけです。 `TRTCVideoEncParam` パラメータはnullに設定することができます。この場合、SDKは画面共有が開始される前のエンコードパラメータを引き続き使用します。

iOS画面共有に推奨されるエンコードパラメータは次のとおりです。

パラメータ項目	パラメータ名	通常の推奨値	テキスト教育シナリオ
解像度	videoResolution	1280 × 720	1920 × 1080
フレームレート	videoFps	10 FPS	8 FPS
最高ビットレート	videoBitrate	1600 kbps	2000 kbps
解像度アダプティブ	enableAdjustRes	NO	NO

説明：

- 画面共有されるコンテンツには通常、大幅な変更がなく、FPSを高く設定するのは経済的ではないため、10FPSを推奨します。
- 共有したい画面コンテンツに大量のテキストが含まれている場合、解像度とビットレートを適宜引き上げることができます。
- 最高ビットレート(videoBitrate)とは、画面が大きく変化したときの最高出力ビットレートのことです。画面コンテンツの変化が少ない場合、実際のエンコードビットレートは低くなります。

方法2：iOSプラットフォームアプリケーション間の共有

サンプルコード

Githubの**trtc_demo/ios**ディレクトリに、アプリケーション間共有用のサンプルコードを設置しています。これには、次のようなテキストが含まれています。

```
├── Broadcast.Upload //スクリーンキャプチャのプロセスBroadcast Upload Extensionコードの詳細は手順2をご参照ください。
| ├── Broadcast.Upload.entitlements //プロセス間通信の設定に使用されるAppGroup情報
| ├── Broadcast.UploadDebug.entitlements //プロセス間通信の設定に使用されるAppGroup情報（debug環境）
| ├── Info.plist
| └── SampleHandler.swift // システムからのスクリーンキャプチャデータを受信するために使用されます
```

```
├── Resource // リソースファイル
├── Runner // TRTC簡易化Demo
├── TXLiteAVSDK_ReplayKitExt.framework //TXLiteAVSDK_ReplayKitExt SDK
```

READMEのガイドによって、このサンプルDemoを実行することができます。

結合手順

iOSシステムでアプリケーション間画面共有を行うには、Extensionスクリーンキャプチャプロセスを追加し、メインアプリプロセスと連携してプッシュを行う必要があります。Extensionスクリーンキャプチャプロセスは、システムによってスクリーンキャプチャが必要なときに作成され、システムが収集した画面イメージの受信を担当します。従って、次の事項を行う必要があります。

1. App Groupを作成し、XCodeにおいて設定を行います（オプション）。このステップは、Extensionスクリーンキャプチャプロセスが、同じメインアプリプロセスとプロセス間通信できるようにすることを目的としています。
2. お客様のプロジェクトにおいて、Broadcast Upload ExtensionのTargetを新規作成し、拡張コモジュール用としてカスタマイズされた `TXLiteAVSDK_ReplayKitExt.framework` をSDK圧縮パッケージに統合します。
3. メインアプリ側の受信ロジックと結合し、メインアプリにBroadcast Upload Extensionからのスクリーンキャプチャデータを待機させます。
4. `pubspec.yaml` ファイルを編集して `replay_kit_launcher` プラグインを導入し、TRTC Demo Screenのようにボタンをクリックすれば画面共有の呼び出しを実現します（オプション）。

```
# trtc sdkとreplay_kit_launcherの導入
dependencies:
tencent_trtc_cloud: ^0.2.1
replay_kit_launcher: ^0.2.0+1
```

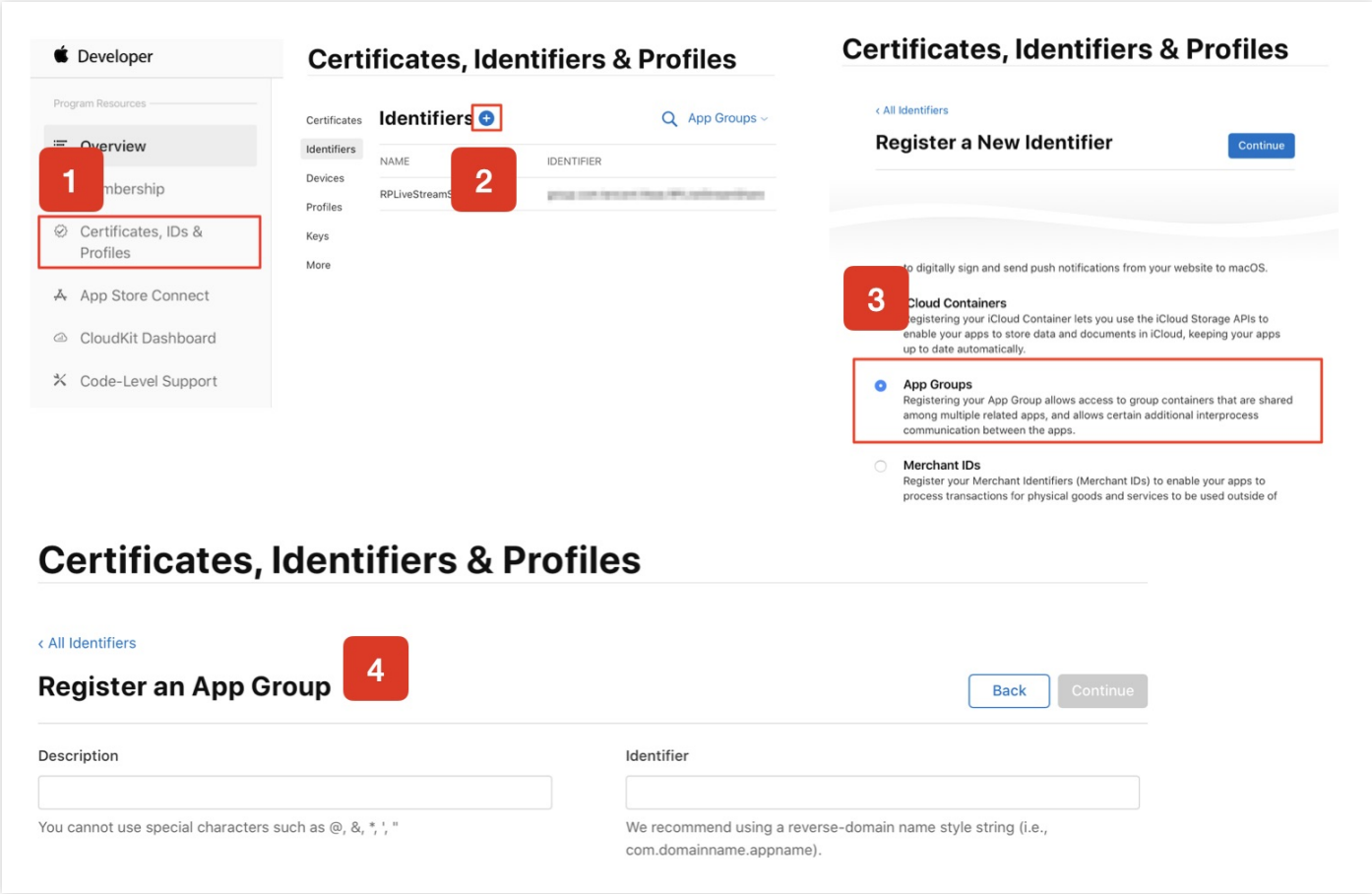
注意：
手順1をスキップした場合、すなわちApp Groupを設定しない場合は（インターフェースはnullを渡します）、画面共有は実行できますが、安定性が若干損なわれます。手順はやや多いですが、できる限り正しいApp Groupを設定して、画面共有機能の安定性を確保してください。

手順1：App Groupsの作成

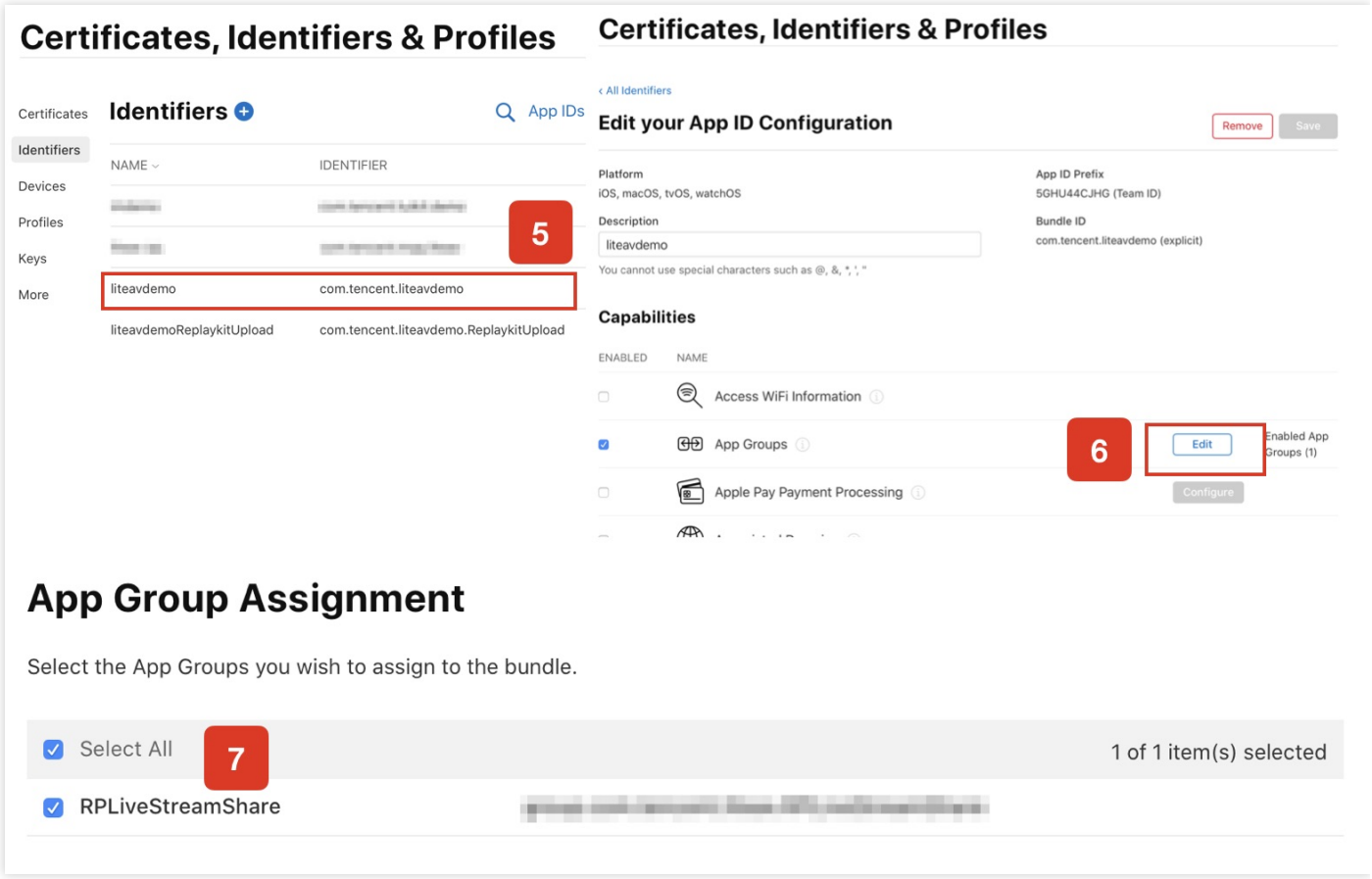
お客様のアカウントを使用して**https://developer.apple.com/**にログインし、次の操作を実行します。**完了後は、対応するProvisioning Profileを再ダウンロードする必要がありますので、ご注意ください。**

1. 【Certificates, IDs & Profiles】をクリックします。
2. 右側のインターフェースでプラス記号をクリックします。
3. 【App Groups】を選択して、【Continue】をクリックします。

4. ポップアップされたフォームにDescriptionとIdentifierを入力します。そのうちIdentifierは、インターフェースにおいて対応するAppGroupパラメータに渡す必要があります。完了したら、【Continue】をクリックします。



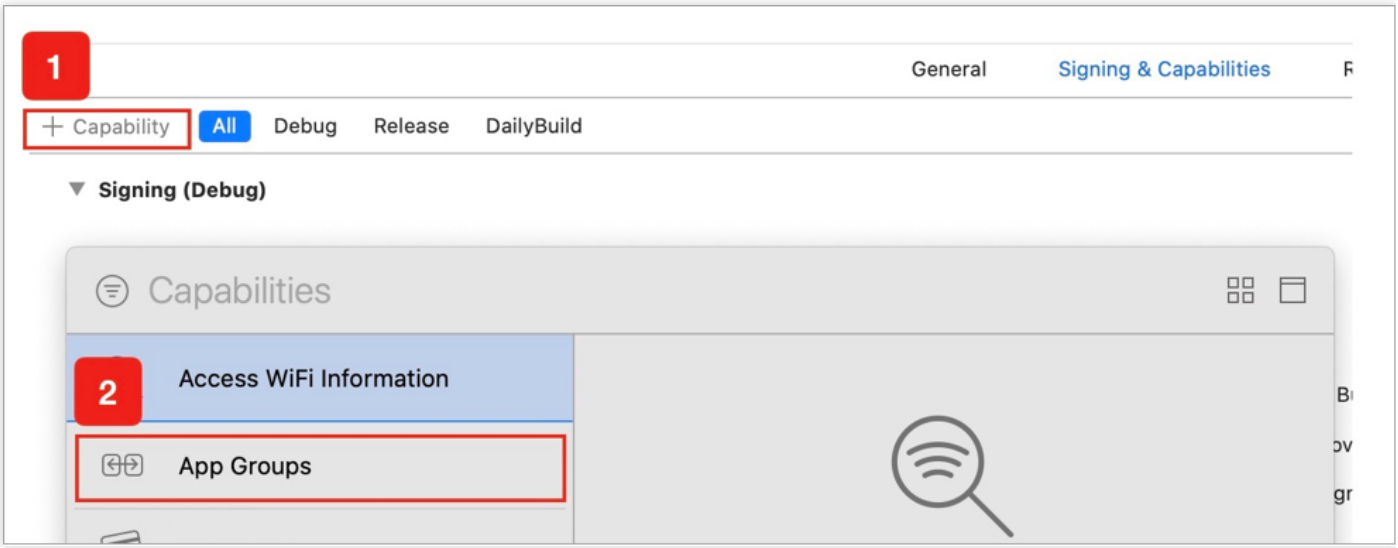
5. Identifierページに戻り、左上のメニューから【App IDs】を選択し、お客様のアプリIDをクリックします（メインアプリとExtensionのアプリIDにも同様の設定が必要です）。
6. 【App Groups】を選択し、【Edit】をクリックします。
7. ポップアップされたフォームからお客様が作成したApp Groupを選択し、【Continue】をクリックして編集ページに戻り、【Save】をクリックして保存します。



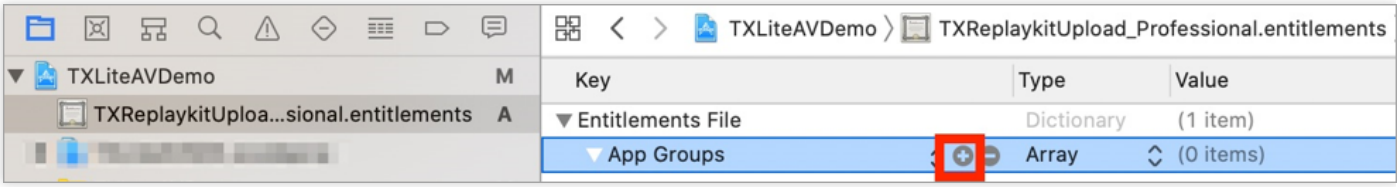
8. Provisioning Profileを再度ダウンロードし、XCodeに設定します。

手順2：Broadcast Upload Extensionの作成

1. Xcodeメニューの【File】 > 【New】 > 【Target...】を順にクリックし、【Broadcast Upload Extension】を選択します。
2. ポップアップされたダイアログボックスに関連情報を入力し、**【Include UI Extension】にはチェック"を入れずに**【Finish】をクリックして作成を完了します。
3. ダウンロードしたSDK圧縮パッケージのプロジェクトにTXLiteAVSDK_ReplayKitExt.frameworkをドラッグし、先ほど作成したTargetにチェックを入れます。
4. 新しく追加したTargetを選択し、【+ Capability】を順番にクリックして、【App Groups】をダブルクリックします。下図のとおりです。



操作が完了すると、下図のようにファイルリストに `Target名.entitlements` という名前のファイルが生成されます。このファイルを選択し、+記号をクリックして上記の手順のApp Groupを入力すればOKです。



5. メインアプリのTargetを選択し、**上記の手順に従って、メインアプリのTargetに同様の処理を行います。**
6. 新規作成したTargetにおいて、Xcodeは「SampleHandler.swift」という名前のファイルを自動的に作成し、それを次のコードに置き換えます。**コード内のAPPGROUPを先ほど作成したApp Group Identifierに変更する必要があります。**

```
import ReplayKit
import TXLiteAVSDK_ReplayKitExt


let APPGROUP = "group.com.tencent.comm.trtc.demo"

class SampleHandler: RPBroadcastSampleHandler, TXReplayKitExtDelegate {

let recordScreenKey = Notification.Name.init("TRTCRecordScreenKey")

override func broadcastStarted(withSetupInfo setupInfo: [String : NSObject]?)
{
// User has requested to start the broadcast. Setup info from the UI extension
can be supplied but optional.
TXReplayKitExt.sharedInstance().setup(withAppGroup: APPGROUP, delegate: self)
}
```

```
override func broadcastPaused() {
// User has requested to pause the broadcast. Samples will stop being delivere
d.
}

override func broadcastResumed() {
// User has requested to resume the broadcast. Samples delivery will resume.
}

override func broadcastFinished() {
// User has requested to finish the broadcast.
TXReplayKitExt.sharedInstance() .finishBroadcast()
}

func broadcastFinished(_ broadcast: TXReplayKitExt, reason: TXReplayKitExtReas
on) {
var tip = ""
switch reason {
case TXReplayKitExtReason.requestedByMain:
tip = 「画面共有は終了しました」
break
case TXReplayKitExtReason.disconnected:
tip = 「アプリケーションは切断されました」
break
case TXReplayKitExtReason.versionMismatch:
tip = 「統合エラー（SDKバージョン番号が一致しません）」
break
default:
break
}

let error = NSError(domain: NSStringFromClass(self.classForCoder), code: 0, us
erInfo: [NSLocalizedFailureReasonErrorKey:tip])
finishBroadcastWithError(error)
}

override func processSampleBuffer(_ sampleBuffer: CMSampleBuffer, with sampleB
ufferType: RPSampleBufferType) {
switch sampleBufferType {
case RPSampleBufferType.video:
// Handle video sample buffer
TXReplayKitExt.sharedInstance() .sendVideoSampleBuffer(sampleBuffer)
break
case RPSampleBufferType.audioApp:
// Handle audio sample buffer for app audio
break
```

```
case RPSampleBufferType.audioMic:
// Handle audio sample buffer for mic audio
break
@unknown default:
// Handle other sample buffer types
fatalError("Unknown type of sample buffer")
}
}
}
```

手順3：メインアプリ側の受信ロジックとの結合

以下の手順に従って、メインアプリ側の受信ロジックと結合します。すなわち、ユーザーが画面共有をトリガーする前に、メインアプリを「待機」状態にして、Broadcast Upload Extensionプロセスからスクリーンキャプチャデータをいつでも受信できるようにします。

1. TRTCCloudがカメラのキャプチャをオフにしていることを確認します。オフになっていない場合は、stopLocalPreview を呼び出して、カメラのキャプチャをオフにしてください。

2. startScreenCaptureメソッドを呼び出し、手順1で設定したAppGroupを渡して、SDKを「待機」状態にします。

3. ユーザーが画面共有をトリガーするまで待機します。手順4における「トリガーボタン」が実装されていない場合、ユーザーがiOSシステムのコントロールセンターにおいて、スクリーンキャプチャボタンを長押しして画面共有をトリガーする必要があります。

4. stopScreenCaptureインターフェースを呼び出すことにより、いつでも画面共有を停止できます。

```
// 画面共有を開始するには、APPGROUPを上記の手順で作成したApp Groupに置き換える必要があります
trtcCloud.startScreenCapture(
TRTCVideoEncParam(
videoFps: 10,
videoResolution: TRTCCloudDef.TRTC_VIDEO_RESOLUTION_1280_720,
videoBitrate: 1600,
videoResolutionMode: TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT,
),
iosAppGroup,
);

// 画面共有を停止します
await trtcCloud.stopScreenCapture();
```

```
// 画面共有の開始イベントの通知は、TRTCCloudListenerを介して受信できます
onRtcListener(type, param){
if (type == TRTCCloudListener.onScreenCaptureStarted) {
//画面共有の開始
}
}
```

手順4：画面共有のトリガーボタンの追加（オプション）

手順3までに、ユーザーがコントロールセンターからスクリーンキャプチャボタンを長押しして、画面共有を手動で開始する必要があります。以下の方法で、TRTC Demo Screenのように、ボタンをクリックすることでトリガーする効果を実現できます。

1. `replay_kit_launcher` プラグインをご使用のプロジェクトに導入します。
2. インターフェースにボタンを設置し、ボタンの応答関数において `ReplayKitLauncher.launchReplayKitBroadcast(iosExtensionName);` 関数を呼び出すと、画面共有機能を呼び出すことができます。

```
// カスタムボタンによる応答メソッド
onShareClick() async {
if (Platform.isAndroid) {
if (await SystemAlertWindow.requestPermissions) {
MeetingTool.showOverlayWindow();
}
} else {
//画面共有機能は実機でのみテストすることができます
ReplayKitLauncher.launchReplayKitBroadcast(iosExtensionName);
}
}
```

画面共有の確認

- **Android/iOS画面共有の確認**
  ユーザーがAndroid / iOSを介して画面共有を実行する場合は、メインストリームを介して共有を実行することができます。ルームにいるその他ユーザーはTRTCCloudListener中のonUserVideoAvailable イベントを介してこの通知を受け取ります。
  画面共有を確認する場合は、startRemoteViewインターフェースを介してリモートユーザーのメインストリーム画面のレンダリングを起動することができます。

よくあるご質問

1つのルームで同時にいくつの画面を共有できますか。

現在、1つの TRTC オーディオ・ビデオルームで共有できる画面は1つだけです。

システム音声の共有
Mac

最終更新日：：2022-07-07 15:24:23

シーンのペインポイントとソリューション

画面共有などのユースケースでは、システムオーディオを相互に共有する必要があります。Macコンピュータのデフォルトのサウンドカードはシステムオーディオのキャプチャをサポートしていないため、Macコンピューターでシステムオーディオを共有することは困難です。これに基づいて、TRTCは、このシーンのニーズを満たすためにMac側でシステムオーディオをレコーディングする機能を提供します。具体的なアクセス手順は次のとおりです。

統合の説明

手順1：TRTCPrivilegedTaskライブラリの統合

SDKは、TRTCPrivilegedTaskライブラリを使用して、仮想サウンドカードプラグインTRTCAudioPlugin.driverをシステムディレクトリ `/ Library / Audio / Plug-Ins/HAL` にインストールするためのroot権限を取得してください。

- CocoaPodsによる統合
- 手動統合

1. 現在のプロジェクトのルートディレクトリの `Podfile` ファイルを開き、以下のコンテンツを追加します：

```
platform :osx, '10.10'

target 'Your Target' do
pod 'TRTCPrivilegedTask', :podspec => 'https://pod-1252463788.cos.ap-guangzhou.myqcloud.com/liteavsdkspec/TRTCPrivilegedTask.podspec'
end
```

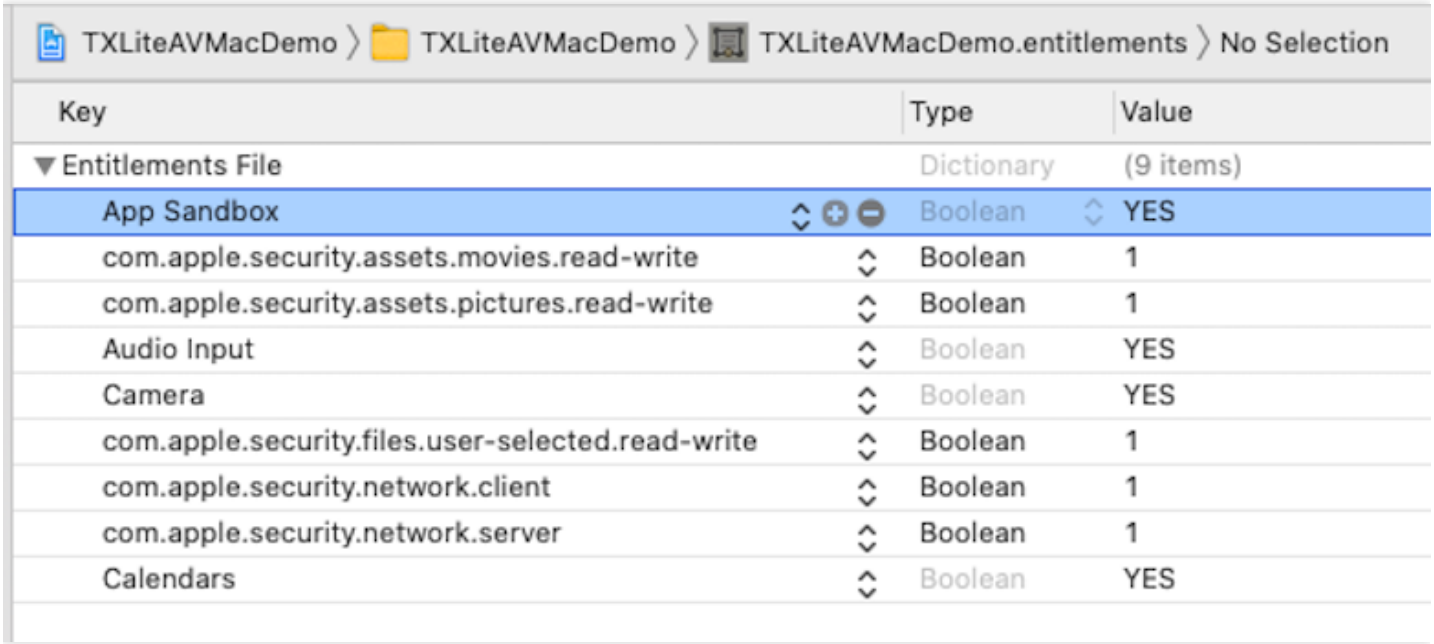
2. `pod install` コマンドを実行し、**TRTCPrivilegedTask**ライブラリをインストールします。

説明

- プロジェクトのルートディレクトリに `Podfile` ファイルがない場合は、まず `pod init` コマンドを実行しファイルを新規作成してから、以下の内容を追加してください。
- CocoaPodsのインストール方法については、CocoaPods公式サイトインストールの説明をご参照ください。

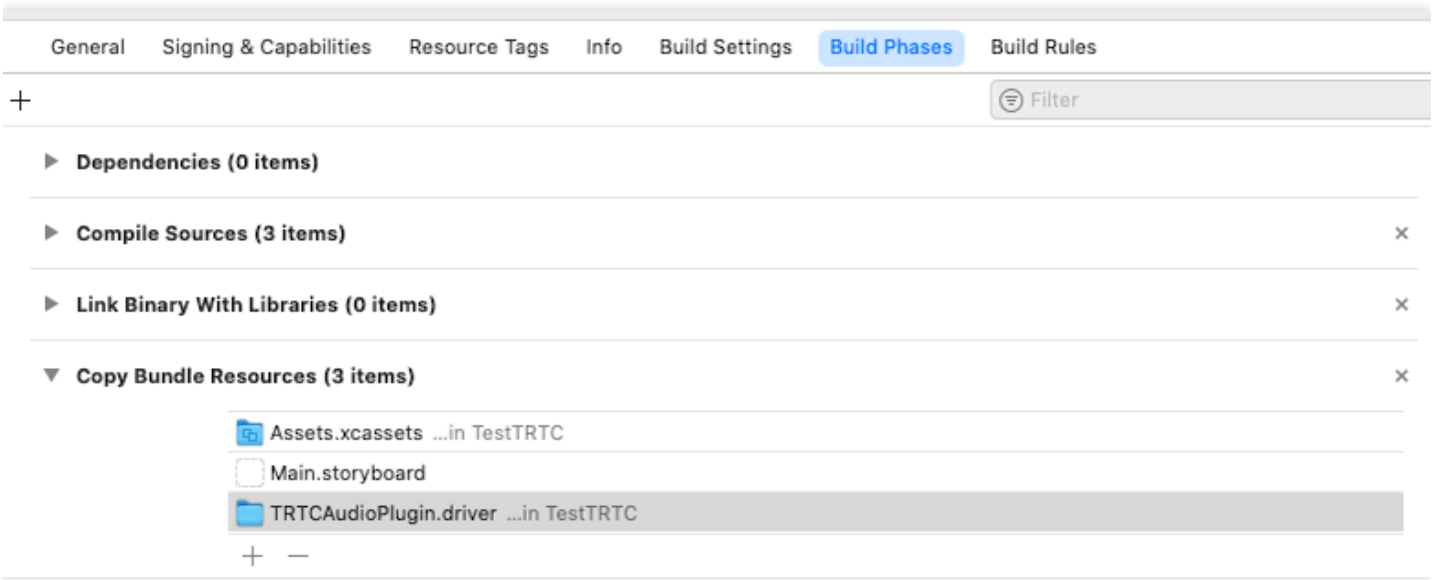
手順2：App Sandbox機能の無効化

Appのentitlements説明ファイルから、**App Sandbox**エントリを削除します。

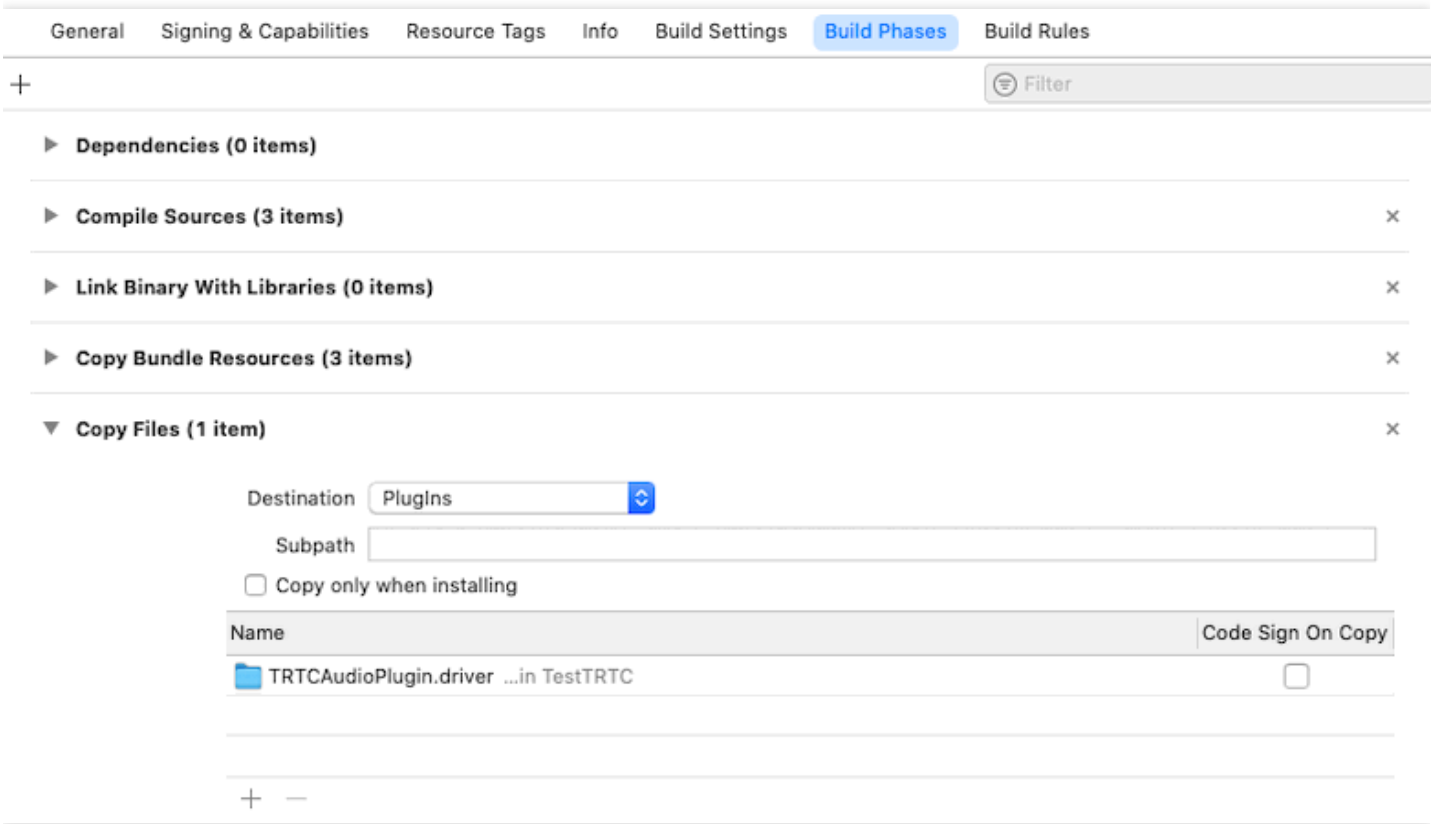


手順3：仮想サウンドカードプラグインのパッケージ化

TRTCPrivilegedTaskライブラリの統合とApp Sandbox機能の無効化の後、システムオーディオレコーディング機能を初めて使用する場合、SDKは仮想サウンドカードプラグインをネットワークからダウンロードしてインストールします。このプロセスをアクセラレーションする場合は、 `TXLiteAVSDK_TRTC_Mac.framework` のPlugInsディレクトリにある仮想サウンドカードプラグイン `TRTCAudioPlugin.driver` をAppBundleのResourcesディレクトリにパッケージ化できます。下図のとおりです：



またはApp BundleのPlugInsディレクトリにコピーします。下図のとおりで：



手順4：システム音声のキャプチャを開始します

startSystemAudioLoopbackインターフェースを呼び出して、システム音声のキャプチャを開始し、アップリンクオーディオストリームに混合します。インターフェースが実行された後、成功または失敗の結果はonSystemAudioLoopbackErrorを介してコールバックされます。

```
TRTCCloud *trtcCloud = [TRTCCloud sharedInstance];
[trtcCloud startLocalAudio];
[trtcCloud startSystemAudioLoopback];
```

注意：
TRTCPrivilegedTaskライブラリの統合とApp Sandbox機能の無効化の後、startSystemAudioLoopbackを最初に呼び出す場合は、root権限を取得します。
ユーザーが**OK**をクリックすると、仮想サウンドカードプラグインが自動的にインストールされます。

手順5：システム音声のキャプチャを停止します

stopSystemAudioLoopbackインターフェースを呼び出して、システム音声のキャプチャを終了します。

```
TRTCCloud *trtcCloud = [TRTCCloud sharedInstance];
[trtcCloud stopSystemAudioLoopback];
```

手順6：システム音声のキャプチャボリュームを設定します

setSystemAudioLoopbackVolumeインターフェースを呼び出して、システム音声のキャプチャボリュームを設定します。

```
TRTCCloud *trtcCloud = [TRTCCloud sharedInstance];
[trtcCloud setSystemAudioLoopbackVolume:80];
```

統合のまとめ

- Mac側では、TRTCは仮想サウンドカードプラグイン `TRTCAudioPlugin.driver` を使用してシステムオーディオをレコーディングします。この仮想サウンドカードプラグインをシステムディレクトリ `/Library/Audio/Plug-Ins/HAL` にコピーし、オーディオサービスを再起動して有効にする必要があります。仮想サウンドカードプラグインが正常にインストールされているかどうかは、 `Launchpad` の `その他` フォルダにある `オーディオMIDI設定` アプリケーションで確認できます。このアプリケーションのデバイスリストに「TRTC Audio Device」という名前のデバイスがある場合は、TRTCの仮想サウンドカードプラグインが正常にインストールされていることを示します。
- 前の手順でのTRTCPrivilegedTaskライブラリの統合とApp Sandbox機能の無効化は、仮想サウンドカードプラグインを自動的にインストールするためのTRTC SDKのroot権限を提供するためのものです。
TRTCPrivilegedTaskライブラリが統合されておらず、App Sandbox機能が保持されている場合、SDKは仮想サ

ウンドカードプラグインを自動的にインストールしませんが、仮想サウンドカードプラグインがシステムにインストールされている場合、システムオーディオレコーディング機能は引き続き正常に使用できます。

> 説明：
> 上記のスキームに加えて、仮想サウンドカードプラグインを手動でインストールして、この機能を統合することもできます。
>
> i. `TXLiteAVSDK_TRTC_Mac.framework` のPlugInsディレクトリにある `TRTCAudioPlugin.driver` をシステムディレクトリ `/Library/Audio/Plug-Ins/HAL` にコピーします。
> ii. システムのオーディオサービスを再起動します。

```
sudo cp -R TXLiteAVSDK_TRTC_Mac.framework/PlugIns/TRTCAudioPlugin.driver /Library/Audio/Plug-Ins/HAL
sudo kill -9 `ps ax|grep 'coreaudio[a-z]' |awk '{print $1}'`
```

注意事項

- App Sandbox機能を無効にすると、Appで取得したユーザーパスが変更されます。
  NSSearchPathForDirectoriesInDomainsなどのシステムメソッドによって取得された `~/Documents` 、`~/Library` などのディレクトリは、サンドボックスディレクトリからユーザーディレクトリ `/Users/ユーザー名/Documents` 、 `/Users/ユーザー名/Library` に切り替えられます。
- TRTCPrivilegedTaskライブラリを統合すると、AppがMac App Storeにリストされなくなる可能性があります。
  SDKが仮想サウンドカードプラグインを自動的にインストールする場合、App Sandbox機能を閉じてルート権限を取得する必要があります。AppがMac App Storeにリストされなくなる可能性があります。詳細については、App Store Review Guidelinesをご参照ください。
  App Storeにリストしたり、Sandbox機能を使用したりする場合は、仮想オーディオプラグインを手動でインストールするスキームを選択することをお勧めします。

Electron

最終更新日：：2022-07-21 17:02:01

シーンのペインポイントとソリューション

画面共有などのユースケースでは、システムオーディオを相互に共有してください。Electronを使用してMacアプリケーションをパッケージ化する場合、Macコンピュータのデフォルトのサウンドカードはシステムオーディオのキャプチャをサポートしていないため、Macコンピューターでシステムオーディオを共有することは困難です。これに基づいて、TRTCは、このシーンのニーズを満たすためにMac側でシステムオーディオをレコーディングする機能を提供します。具体的なアクセス手順は次のとおりです。

手順1：システム音声のキャプチャを開始します

startSystemAudioLoopbackインターフェースを呼び出して、システム音声のキャプチャを開始し、アップリンクオーディオストリームに混合します。インターフェースが実行された後、成功または失敗の結果はonSystemAudioLoopbackErrorを介してコールバックされます。

```
import TRTCCloud, { TRTCAudioQuality } from 'trtc-electron-sdk';
const rtcCloud = new TRTCCloud();
function onSystemAudioLoopbackError(errCode) {
if(errCode === 0) {
console.log(‘起動成功');
}
if (errCode === -1330) {
console.log('システム音声のレコーディングを有効にできませんでした。たとえば、オーディオドライバプラグインが利用できません');
}
if (errCode === -1331) {
console.log('オーディオドライバプラグインのインストールが許可されていません');
}
if (errCode === -1332) {
console.log('オーディオドライバプラグインのインストールに失敗しました');
}
}
trtcCloud.on('onSystemAudioLoopbackError', onSystemAudioLoopbackError);
trtcCloud.startLocalAudio(TRTCAudioQuality.TRTCAudioQualityDefault);
trtcCloud.startSystemAudioLoopback();
```

注意：

startSystemAudioLoopbackを最初に呼び出すと、root権限が取得され（下図を参照）、ユーザーが**OK**をクリックすると、仮想サウンドカードプラグインが自動的にインストールされます。

手順2：システム音声のキャプチャを停止します

stopSystemAudioLoopbackインターフェースを呼び出して、システム音声のキャプチャを終了します。

```
trtcCloud.stopSystemAudioLoopback();
```

手順3：システム音声のキャプチャボリュームを設定します

setSystemAudioLoopbackVolumeインターフェースを呼び出して、システム音声のキャプチャボリュームを設定します。

```
trtcCloud.setSystemAudioLoopbackVolume(60);
```

統合のまとめ

Mac側では、TRTCは仮想サウンドカードプラグイン `TRTCAudioPlugin.driver` を使用してシステムオーディオをレコーディングします。この仮想サウンドカードプラグインをシステムディレクトリ `/Library/Audio/Plug-Ins/HAL` にコピーし、オーディオサービスを再起動して有効にする必要があります。仮想サウンドカードプラグインが正常にインストールされているかどうかは、 `Launchpad` の `その他` フォルダにある `オーディオMIDI設定` アプリケーションで確認できます。このアプリケーションのデバイスリストに「TRTC Audio Device」という名前のデバイスがある場合は、TRTCの仮想サウンドカードプラグインが正常にインストールされていることを示します。

画面品質の設定
Android&iOS&Windows&Mac

最終更新日：：2022-09-06 10:40:53

内容紹介

TRTRTCCloudでは、以下の方式で画質を調整できます。

- TRTCCloud.enterRoom のTRTCAppScene パラメータ：アプリケーションユースケースを使用するのに使用します。
- TRTCCloud.setVideoEncoderParam：エンコードのパラメータを設定するのに使用します。
- TRTCCloud.setNetworkQosParam：ネットワークの制御ポリシーを設定するのに使用します。

ここでは、主に上述のパラメータを設定し、TRTC SDKの画質効果を項目のニーズに適合させる方法を紹介します。
以下のDemoを参照することもできます。

- iOS：SetVideoQualityViewController.m
- Android：SetVideoQualityActivity.java
- Windows：TRTCMainViewController.cpp

サポートするプラットフォーム

iOS	Android	Mac OS	Windows	Web	Electron	Flutter
✓	✓	✓	✓	✓	✓	✓

Web端末で画質の詳細操作を設定します。 設定ガイドをご参照ください。

TRTCAppScene

- **VideoCall**
  ビデオ通信のユースケース、すなわち、長時間二人以上のビデオ通信に対応するユースケースです。内部のエンコーダーおよびネットワークプロトコルを最適化して流暢性に特化し、通話ディレーおよび遅延率を低下させます。

- **LIVE**
  ライブストリーミングのユースケース、すなわち、長時間一人でのライブストリーミングに対応し、時々多人数でのビデオインタラクティブのユースケースに対応します。内部エンコーダーおよびネットワークプロトコルは最適化して性能及び互換性に特化し、性能及び解像度をさらに良くします。

TRTCVideoEncParam

推奨する設定

ユースケース	videoResolution	videoFps	videoBitrate
ビデオ通話（携帯電話）	640x360	15	550kbps
ビデオミーティング（メイン画面 @ Mac Win）	1280x720	15	1200kbps
ビデオミーティング（メイン画面 @ 携帯電話）	640x360	15	900kbps
ビデオミーティング（小画面）	320x180	15	250kbps
eラーニング（教師 @ Mac Win）	960x540	15	850kbps
eラーニング（教師 @ iPad）	640x360	15	550kbps
eラーニング（学生）	320x180	15	250kbps

各フィールドの詳細説明

- **(TRTCVideoResolution) videoResolution**
  エンコード解像度は、 例えば、640 x 360 はエンコードした画面の幅（画素） x 高さ（画素）を指し、TRTCVideoResolution で列挙した定義では、幅は高さ以上の長さとの横型スクリーン（Landscape）の解像度を定義しただけです。縦型スクリーン解像度を使用したい場合は、 resModeを Portraitに設定する必要があります。

> 注意：
> 多くのハードウェアのコーデックが 16 で割り切れる画素数の幅のみをサポートすることから、 SDKが実際にエンコードする解像度は、必ずしもパラメータに従って自ら設定したわけではなく、 16 で割って自動で修正したものです。例えば、 640 x 360 の解像度なら、 SDK 内部では 640 x 368に対応することがありえます。

- **(TRTCVideoResolutionMode) resMode**
  横スクリーンまたは縦スクリーンの解像度を指し、TRTCVideoResolution には横スクリーンの解像度しか定義していないことから、 360 x 640 のような縦スクリーン解像度を使用したい場合は、 resMode を TRTCVideoResolutionModePortraitに指定する必要があります。一般に PCおよび Macは、横スクリーン（Landscape）解像度を採用し、携帯電話は縦スクリーン（Portrait）解像度を採用しています。

- **(int) videoFps**
  フレームレート（FPS）も、毎秒どれだけのフレーム画面をエンコードするかを示すものです。15 FPSに設定することをお薦めします。そうすれば、画面が十分にスムーズに流れ、毎秒のフレーム数が多すぎることによりフレーム画面の明瞭度を下げることはありません。

  スムーズさへの要求が高い場合は、20FPSまたは25FPSに設定することができます。しかし、25FPSより大きい数値には設定しないでください。映画の通常のフレームレートも24FPSしかないためです。

- **(int) videoBitrate**
  ビットレート（Bitrate）とは、毎秒エンコーダーがどれだけかのKbit のエンコードを出力した後のバイナリーデータのことです。 videoBitrate を 800kbpsに設定した場合は、エンコーダーが毎秒 800kbit のビデオデータを産生することになります。これらのデータを一つのファイルに保存すると、ファイル容量は 800kbit、つまり 100KB、0.1MBになります。

  ビデオビットレートは高ければ良いというものではなく、解像度との間にやや適切なマッピング関係があり、下表の示すとおりになります。

解像度ビットレート参照表

解像度の定義	アスペクト比	推奨ビットレート（VideoCall）	推奨ビットレート（LIVE）
TRTCVideoResolution_120_120	1:1	80kbps	120kbps
TRTCVideoResolution_160_160	1:1	100kbps	150kbps
TRTCVideoResolution_270_270	1:1	200kbps	300kbps
TRTCVideoResolution_480_480	1:1	350kbps	525kbps
TRTCVideoResolution_160_120	4:3	100kbps	150kbps
TRTCVideoResolution_240_180	4:3	150kbps	225kbps
TRTCVideoResolution_280_210	4:3	200kbps	300kbps

解像度の定義	アスペクト比	推奨ビットレート（VideoCall）	推奨ビットレート（LIVE）
TRTCVideoResolution_320_240	4:3	250kbps	375kbps
TRTCVideoResolution_400_300	4:3	300kbps	450kbps
TRTCVideoResolution_480_360	4:3	400kbps	600kbps
TRTCVideoResolution_640_480	4:3	600kbps	900kbps
TRTCVideoResolution_960_720	4:3	1000kbps	1500kbps
TRTCVideoResolution_160_90	16:9	150kbps	250kbps
TRTCVideoResolution_256_144	16:9	200kbps	300kbps
TRTCVideoResolution_320_180	16:9	250kbps	400kbps
TRTCVideoResolution_480_270	16:9	350kbps	550kbps
TRTCVideoResolution_640_360	16:9	550kbps	900kbps
TRTCVideoResolution_960_540	16:9	850kbps	1300kbps
TRTCVideoResolution_1280_720	16:9	1200kbps	1800kbps
TRTCVideoResolution_1920_1080	16:9	2000kbps	3000kbps

TRTCNetworkQosParam

QosPreference

ネットワークの帯域幅に比較的余裕がある状況では、明瞭度とスムーズさは同時に対応できます。しかし、ユーザーのネットワークが満足のいく状態でない場合は、最終的に明瞭度とスムーズさのどちらを優先するかを、TRTCNetworkQosParam の preference パラメータを指定することで選択することができます。

- **スムーズさを優先（TRTCVideoQosPreferenceSmooth）**
  ユーザーのネットワーク環境が脆弱な場合は、画面がぼやけ、モザイクが多くかかることがあります。しかし、スムーズさを維持しラグを軽減することができます。

- **解像度を優先（TRTCVideoQosPreferenceClear）**
  ユーザーのネットワーク環境が脆弱な場合は、画面はできるだけ明瞭さを維持しますが、ラグが現れやすくなります。

ControlMode

controlMode パラメータでは、 **TRTCQosControlModeServer** を選択すればOKです。
TRTCQosControlModeClient は、Tencent Cloud研究開発チームが内部テスト用としたものですので、無視してください。

よく見られるエラー箇所

1. 解像度は高いほど良いのですか？
やや高い解像度も、やや高いビットレートでのサポートが必要です。1280 x 720の解像度を選択した場合は、ビットレートを200kbpsに指定すると、画面に大量のモザイクが出てしまいます。 解像度ビットレート参照リストを参照して設定することをお薦めします。

2. フレームレートは高いほど良いですか？
カメラで撮影する画面は、露出段階ですべての現実の物体の完全なマッピングですので、フレームレートが高いほど、スムーズに感じるわけではありません。この点でゲームのFPSと異なります。正反対に、フレームレートが高すぎると、各フレームの画質を損ない、カメラの露出時間も減少させ、効果はさらに落ちることになります。

3. ビットレートは高いほど良いですか？
高いビットレートも高い解像度によって対応します。 320 x 240 の解像度に、1000kbps のビットレートは大きすぎますので、 解像度ビットレート参照リスト を参照して設定することをお薦めします。

4. Wi-Fi を使用する場合、高い解像度およびビットレートを設定可能
Wi-Fiのネット速度がいつも変わらないということはありません。無線ルーターから遠い場合、またはルーター回線が占有されている場合は、ネット速度は4Gよりも劣ることになります。
こうした状況では、TRTC SDKは速度計測機能を提供し、ビデオ通信前に速度を計測し、測定値からネットワークの良し悪しを判断します。

Web

最終更新日：：2022-08-08 15:25:08

ここでは主に、ビデオ通話またはインタラクティブライブストリーミングで画質を設定する方法についてご説明します。開発者は具体的な業務上のニーズに基づいて、ビデオ画面の解像度とスムーズさを調整し、ユーザー体験をより良いものにすることができます。
ビデオの属性には解像度、フレームレート、ビットレートが含まれます。

実現方式

ローカルオーディオビデオストリーミングStreamオブジェクトの `{@link LocalStream#setVideoProfile setVideoProfile()}` のメソッドでビデオの属性を設定します。

- 事前に定義したProfileを指定します。各Profileは1組の推奨される解像度、フレームレート、ビットレートに対応します。

```
const localStream = TRTC.createStream({ userId, audio: true, video: true });
// ビデオ属性Profileを‘480p’に設定します
localStream.setVideoProfile('480p');
localStream.initialize().then(() => {
console.log('local stream init success');
localStream.play('local_stream');
});
```

- カスタマイズした解像度、フレームレート、ビットレートを指定します

```
const localStream = TRTC.createStream({ userId, audio: true, video: true });
// ビデオの解像度、フレームレート、ビットレートをカスタマイズ
localStream.setVideoProfile({ width: 640, height: 480, frameRate: 15, bitrate:
900 /* kpbs */});
localStream.initialize().then(() => {
console.log('local stream init success');
localStream.play('local_stream');
});
```

注意：

- v4.8.4およびそれ以降のバージョンでは、 `{@link LocalStream#setVideoProfile setVideoProfile()}` メソッドで動的呼び出しをサポートしています。詳細な情報について

は、`{@link LocalStream#setVideoProfile setVideoProfile()}` インターフェースの説明をご参照ください。
- v4.8.4より前のバージョンでは、`{@link LocalStream#setVideoProfile setVideoProfile()}` はローカルストリームで `{@link LocalStream#initialize initialize()}` を呼び出して初期化する前に呼び出す必要があり、通話中にビデオ属性を動的に調整することはできません。

ビデオ属性Profileリスト

ビデオprofile	解像度（幅 x 高さ）	フレームレート（fps）	ビットレート（kbps）
120p	160 × 120	15	200
180p	320 × 180	15	350
240p	320 × 240	15	400
360p	640 × 360	15	800
480p	640 × 480	15	900
720p	1280 × 720	15	1500
1080p	1920 × 1080	15	2000
1440p	2560 × 1440	30	4860
4K	3840 × 2160	30	9000

デバイスとブラウザの制限があるため、ビデオの解像度が完全にマッチするとは限りません。マッチしない場合、ブラウザは自動的に解像度を調整し、Profileに対応する解像度に近づけます。

Electron

最終更新日：：2022-07-21 17:08:12

このドキュメントでは、主にビデオ通話やInteractive Live Video Broadcastingで画質を設定する方法を紹介します。開発者は、具体的なのサービスニーズに応じてビデオ画像の鮮明さと滑らかさを調整して、ユーザーエクスペリエンスを向上させることができます。
ビデオのプロパティには、解像度、フレームレート、ビットレートが含まれます。

内容紹介

Electron SDKにおいて、以下の方法で画質を調整できます：

- enterRoomのTRTCAppSceneパラメータ：ユースケースの選択に使用されます。
- setVideoEncoderParam：エンコードパラメータの設定に使用されます。
- setNetworkQosParam：ネットワークチューニングポリシーの設定に使用されます。

このドキュメントでは、主にTRTC SDKの画質をプロジェクトのニーズに合わせるために、上記のパラメーターを設定する方法を紹介します。
Electron API Example：video-qualityDemoを参照することもできます。

TRTCAppScene

- **VideoCall**：ビデオ通話シーン、つまり、ほとんどの時間において2人以上がビデオ通話を行うシーンに対応して、内部エンコーダとネットワークプロトコルの最適化は、滑らかさ、通話遅延の低減、ラグ率に重点を置いています。
- **LIVE**：ライブストリーミングシーン、つまり、ほとんど1人がライブストリーミングし、時々に複数の人のビデオインタラクションがあるシーンに対応して、内部エンコーダとネットワークプロトコルの最適化は、性能と互換性に重点を置き、性能と鮮明さが向上します。

TRTCVideoEncParam

推奨される構成

ユースケース	videoResolution	videoFps	videoBitrate
ビデオ会議（メイン画面 @ Mac Win）	1280x720	15	1200kbps

ユースケース	videoResolution	videoFps	videoBitrate
オンライン教育（教師 @ Mac Win）	960x540	15	850kbps

各フィールドの詳細な説明

- **(TRTCVideoResolution) videoResolution**
  640 x 360などのエンコード解像度は、エンコードされた画像の幅（ピクセル）x高さ（ピクセル）を指します。TRTCVideoResolution列挙定義で幅>=高さの横向きの解像度のみを定義します。縦向きを使用する場合はresModeをPortraitに設定してください。

  > 注意：
  > 多くのハードウェアコーデックは16で割り切れるピクセル幅のみをサポートするため、SDKによって実際にエンコードされる解像度は、必ずしもパラメータに従って完全にカスタマイズされるとは限りませんが、16で割り切れる値によって自動的に修正されます。たとえば、640x360の解像度は、SDK内部で640x368に適合させることができます。

- **(TRTCVideoResolutionMode) resMode**
  横向きまたは縦向きの解像度を指します。TRTCVideoResolutionでは横向きの解像度のみが定義されているため、360x640などの縦向きの解像度を使用する場合は、resModeをTRTCVideoResolutionModePortraitとして指定してください。一般的に、PCとMacは横向き(Landscape)の解像度を使用し、携帯電話は縦向き(Portrait)の解像度を使用します。

- **(int) videoFps**
  フレームレート(FPS)も、毎秒どれだけのフレーム画面をエンコードするかを示すものです。15 FPSに設定することをお勧めします。そうすれば、画面が十分にスムーズに流れ、毎秒のフレーム数が多すぎることによりフレーム画面の解像度を下げることはありません。

  滑らかさの要件が高い場合は、20FPSまたは25FPSに設定できます。ただし、映画の通常のフレームレートは24FPSのみであるため、25FPS以上の値を設定しないでください。

- **(int) videoBitrate**
  ビデオビットレートは、エンコーダが1秒間に出力するエンコードされたバイナリデータのKbit数です。videoBitrateを800kbpsに設定すると、エンコーダは1秒間に800kbitのビデオデータを生成します。これらのデータをファイルとして保存すると、ファイルサイズは800kbit、つまり100KB、つまり0.1Mになります。

ビデオのビットレートは高いほどよいとは限りません。次の表に示すように、ビデオのビットレートと解像度との間にはより適切なマッピング関係が必要です。

解像度とビットレートの参照テーブル

解像度の定義	アスペクト比	推奨ビットレート（VideoCall）	推奨ビットレート（LIVE）
TRTCVideoResolution_120_120	1:1	80kbps	120kbps
TRTCVideoResolution_160_160	1:1	100kbps	150kbps
TRTCVideoResolution_270_270	1:1	200kbps	300kbps
TRTCVideoResolution_480_480	1:1	350kbps	525kbps
TRTCVideoResolution_160_120	4:3	100kbps	150kbps
TRTCVideoResolution_240_180	4:3	150kbps	225kbps
TRTCVideoResolution_280_210	4:3	200kbps	300kbps
TRTCVideoResolution_320_240	4:3	250kbps	375kbps
TRTCVideoResolution_400_300	4:3	300kbps	450kbps
TRTCVideoResolution_480_360	4:3	400kbps	600kbps
TRTCVideoResolution_640_480	4:3	600kbps	900kbps
TRTCVideoResolution_960_720	4:3	1000kbps	1500kbps
TRTCVideoResolution_160_90	16:9	150kbps	250kbps
TRTCVideoResolution_256_144	16:9	200kbps	300kbps
TRTCVideoResolution_320_180	16:9	250kbps	400kbps
TRTCVideoResolution_480_270	16:9	350kbps	550kbps
TRTCVideoResolution_640_360	16:9	550kbps	900kbps
TRTCVideoResolution_960_540	16:9	850kbps	1300kbps
TRTCVideoResolution_1280_720	16:9	1200kbps	1800kbps
TRTCVideoResolution_1920_1080	16:9	2000kbps	3000kbps

TRTCNetworkQosParam

QosPreference

ネットワーク帯域幅が十分な場合は、鮮明さと滑らかさの両方を考慮に入れることができますが、ユーザーのネットワークが理想的でない場合、鮮明さと滑らかさを確保するための優先事項はどちらですか？TRTCNetworkQosParamのpreferenceパラメータを指定して選択できます。

- **滑らかさを優先とする（TRTCVideoQosPreferenceSmooth）**
  ユーザーが弱いネットワーク環境に遭遇すると、画像がぼやけてモザイクが増えますが、滑らかなままになるか、わずかにラグする可能性があります。

- **鮮明さを優先とする（TRTCVideoQosPreferenceClear）**
  ユーザーが弱いネットワーク環境に遭遇すると、画像は可能な限り鮮明に保たれますが、ラグしやすくなる可能性があります。

ControlMode

controlModeパラメーターには**TRTCQosControlModeServer**を選択してよいです。TRTCQosControlModeClientは、Tencent Cloud R&Dチームが内部デバッグに使用されます。注意しないでください。

よくある間違い

1. 解像度が高いほど良いですか。

解像度が高いほど、サポートするにはビットレートも高くなります。解像度が1280x720で、ビットレートが200kbpsに指定されている場合、画像には多くのモザイクが現れます。設定については、解像度とビットレートの参照テーブルを参照することをお勧めします。

2. フレームレートが高いほど良いですか。

カメラで取得した画像は、露光段階のすべての実物を完全にマッピングしたものであるため、フレームレートが高いほど感覚が滑らかになるわけではなく、これはゲームのFPSとは異なります。逆に、フレームレートが高すぎると、各フレームの画質が低下し、カメラの露光時間も短くなり、効果がより低下する場合があります。

3. ビットレートが高いほど良いですか。

ビットレートが高いほど、一致する解像度も高くなります。320x240などの解像度の場合、1000kbpsのビットレートは無駄です。設定については、解像度とビットレートの参照テーブルを参照することをお勧めします。

4. Wi-Fi使用時に高い解像度とビットレートを設定できますか。

Wi-Fiの速度が一定というわけではありません。無線ルーターから遠く離れている場合やルーターのチャネルが占有されている場合は、ネットワーク速度が4Gほど良くない場合があります。
このような状況に対応するために、TRTC SDKは速度測定機能を提供しており、ビデオ通話前に速度を測定し、採点値に応じてネットワークの品質を判断することができます。

ビデオ画面の回転とスケーリング Android&iOS&Windows&Mac

最終更新日：：2022-07-07 15:29:37

内容紹介

Tencent Real-Time Communication(TRTC)は、携帯電話のライブストリーミングのような縦画面による変化に乏しいユーザーエクスペリエンスとは異なり、横画面と縦画面という2つのユースケースを両立させることが必要です。このため、横画面と縦画面の処理ロジックを対応させる必要があります。ここでは主に、次の事項についてご説明します：

- 縦画面モードを実装する方法。例：WeChatのビデオ通話は典型的な縦画面体験モードです。
- 横画面モードを実装する方法。例：多人数インタラクティブオーディオビデオルームApp（XYLink（小魚易連）など）は多くの場合、横画面モードを採用しています。
- ローカル画面およびリモート画面の回転方向およびフィルモードをカスタマイズする方法。

サポートしているプラットフォーム

iOS	Android	Mac OS	Windows	Electron	Web端末
✓	✓	✓	✓	✓	×

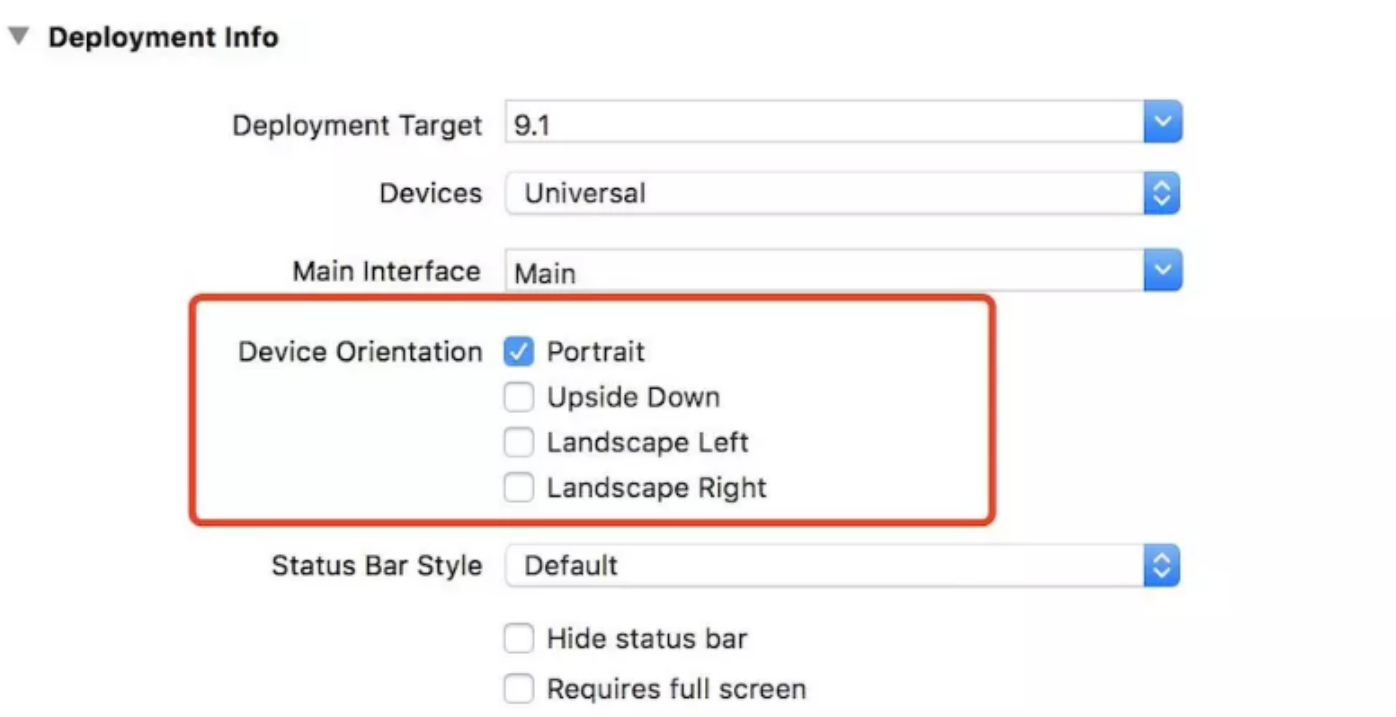
縦画面モード

WeChatのビデオ通話のような体験モードを実装したい場合は、2つの作業を行ってください：

1. アプリのUIインターフェースを縦画面に設定する

- iOSプラットフォーム
- Androidプラットフォーム

XCodeの**General**>**Deployment Info**>**Device Orientation**で直接設定することができます：



Appdelegateの `supportedInterfaceOrientationsForWindow` メソッドを実装することにより、設定することもできます：

```
- (UIInterfaceOrientationMask)application:(UIApplication *)application
supportedInterfaceOrientationsForWindow:(UIWindow *)window
```

```
{

return UIInterfaceOrientationMaskPortrait ;

}
```

説明
CSDNに掲載されている、iOSの縦横画面の回転およびその基本的な適用方法という記事に、iOSプラットフォームで画面の向きに関する開発工程の一部を詳細に説明しています。

2. SDKで使用する縦画面解像度の設定

TRTCCloudを使用したsetVideoEncoderParamインターフェースにビデオコーデックパラメータを設定する場合は、resModeを `TRTCVideoResolutionModePortrait` に指定します。
サンプルコードは次のとおりです。

- iOS swift
- Android java

```
TRTCVideoEncParam* encParam = [TRTCVideoEncParam new];
encParam.videoResolution = TRTCVideoResolution_640_360;
encParam.videoBitrate = 600;
encParam.videoFps = 15;
encParam.resMode = TRTCVideoResolutionModePortrait; //解像度モードを縦画面モードに設定

[trtc setVideoEncoderParam: encParam];
```

横画面モード

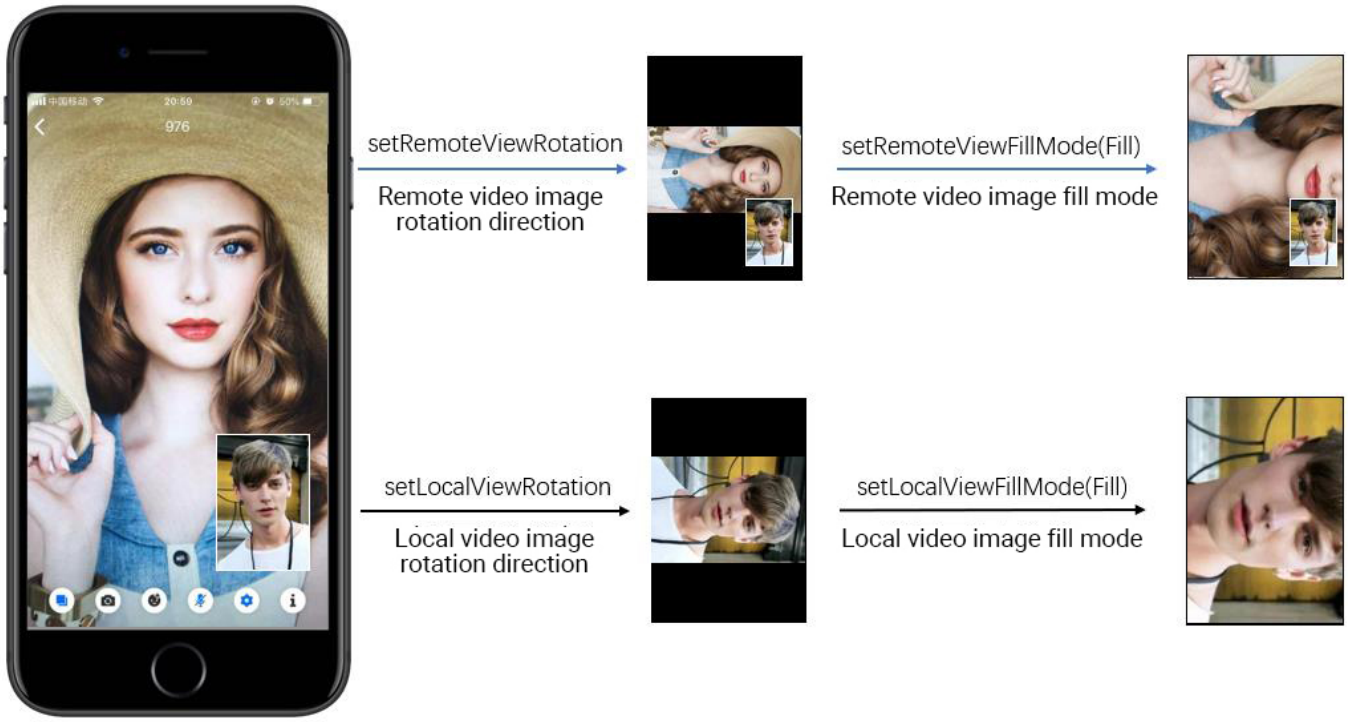
アプリに横画面のユーザーエクスペリエンスを提供する場合、縦画面モードと同様の作業を行う必要がありますが、手順1と2のパラメータに対応する調整を行うだけでOKです。
特に手順2では、TRTCVideoEncParamのresMode値は次のとおりとなります：

- iOSプラットフォームでは `TRTCVideoResolutionModeLandscape` と指定してください。
- Androidプラットフォームでは `TRTC_VIDEO_RESOLUTION_MODE_LANDSCAPE` と指定してください。

カスタマイズ制御

TRTC SDKにより提供された多数のインターフェース関数は、ローカルおよびリモートの画面の回転方向およびフィルモードをコントロールできます：

インターフェース関数	機能の役割	備考説明
setLocalViewRotation	ローカルプレビュー画面での時計回りの回転角度	時計回りの回転を90度、180度、270度という3方向でサポート
setLocalViewFillMode	ローカルプレビュー画面のフィルモード	トリミングか黒枠を残すか
setRemoteViewRotation	リモートビデオ画面での時計回りの回転角度	時計回りの回転を90度、180度、270度という3方向でサポート
setRemoteViewFillMode	リモートビデオ画面のフィルモード	トリミングか黒枠を残すか
setVideoEncoderRotation	エンコーダーを設定して出力した画面の時計回りの回転角度	時計回りの回転を90度、180度、270度という3方向でサポート



GSensorMode

画面の回転はレコーディングやCDN Relayed live streamingの調節に関するさまざまな項目に影響することを考慮すると、TRTC SDKはシンプルな重力センサーの自動調節機能を提供するだけであるので、TRTCCloudの `setGSensorMode` インターフェースを介してオンにすることができます。

この機能は、現在180度の上下回転の自動調節のみをサポートしています。つまり、ユーザー自身の携帯電話が上下180度逆さまになっている場合、相手が見る画面の向きが変わらないように維持します（90度または270度回転への適応はまだサポートしていません）。この自動調節はエンコーダーの方向調整に基づいて実装されているため、レコーディングしたビデオやH5端末に表示されるビデオ画面も、元の向きを維持することができます。

> 注意：
> 重力センサーの自動調節のもう一つの実装ソリューションでは、各フレームのビデオ情報ごとに現在のビデオの重力方向を対応させた上で、リモートユーザー側でレンダリング方向を自動調節します。ただしこのソリューションでは、レコーディングしたビデオの向きと求めるビデオの向きの一致を維持させるために、追加でトランスコードリソースを導入する必要があるためお勧めできません。

Electron

最終更新日：：2022-07-21 17:11:01

Tencent Real-Time Communication (TRTC)は、ローカル画面とリモート画面の回転方向と塗りつぶしモードのカスタム制御をサポートします。

ローカル画面のカスタム制御

setLocalRenderParamsを呼び出すことにより、ローカルレンダリングパラメータを設定できます。

```
import TRTCCloud, {
TRTCRenderParams, TRTCVideoRotation, TRTCVideoFillMode,
TRTCVideoMirrorType
} from 'trtc-electron-sdk';
const trtcCloud = new TRTCCloud();
const param = new TRTCRenderParams(
TRTCVideoRotation.TRTCVideoRotation90,
TRTCVideoFillMode.TRTCVideoFillMode_Fill,
TRTCVideoMirrorType.TRTCVideoMirrorType_Enable
);
trtcCloud.setLocalRenderParams(param);
const localUserDom = document.querySelector('local-user');
trtcCloud.startLocalPreview(localUserDom);
```

リモート画面のカスタム制御

setRemoteRenderParamsを呼び出すことにより、リモートレンダリングパラメータを設定できます。

```
import TRTCCloud, {
TRTCRenderParams, TRTCVideoRotation, TRTCVideoFillMode,
TRTCVideoMirrorType, TRTCVideoStreamType
} from 'trtc-electron-sdk';
const trtcCloud = new TRTCCloud();
const param = new TRTCRenderParams(
TRTCVideoRotation.TRTCVideoRotation180,
TRTCVideoFillMode.TRTCVideoFillMode_Fill,
TRTCVideoMirrorType.TRTCVideoMirrorType_Disable
);
const remoteUserId = 'remoteUser';
trtcCloud.setRemoteRenderParams(remoteUserId, TRTCVideoStreamType.TRTCVideoStream
```

```
TypeBig, param);
const remoteUserDom = document.querySelector('remote-user');
trtcCloud.startRemoteView(remoteUserId, remoteUserDom, TRTCVideoStreamType.TRTCVi
deoStreamTypeBig);
```

7.FAQs
入門者のよくあるご質問

最終更新日：：2022-05-18 12:13:16

UserSigとは。

UserSigとは、悪意ある攻撃者によるクラウドサービスの使用権の盗用を防ぐために、Tencent Cloudによって設計されたセキュリティ保護された署名です。
現在、TRTC、IMおよびMLVBなどのサービスには、すべてこの一連のセキュリティ保護メカニズムが採用されています。これらのサービスを使用する場合、対応するSDKの初期化またはログイン関数において、SDKAppID、UserIDおよびUserSigという3つの重要情報を提供する必要があります。
このうちSDKAppIDはお客様のアプリケーションの識別に、UserIDはユーザーの識別に使用されます。UserSigは、** HMAC SHA256 **暗号化アルゴリズムによって算出される最初の2つをもとに計算されたセキュリティ署名です。攻撃者がUserSigを偽造できない限り、クラウドサービスのトラフィックを盗用することはできません。
UserSigの計算原理を次の図に示します。SDKAppID、UserID、ExpireTimeなどの重要情報をハッシュ化・暗号化することがUserSigの本質です。

```
//UserSigの計算式。このうちsecretkeyは、usersigを算出するための暗号化鍵です。
usersig = hmacsha256(secretkey, (userid + sdkappid + currtime + expire +
base64(userid + sdkappid + currtime + expire)))
```

説明：

- `currtime` は現在のシステムの時間で、 `expire` は署名の期限切れの時間です。
- UserSigの具体的な計算取得方法を知りたい場合は、 UserSigの詳細な説明をご参照ください。

TRTCではルームを同時にいくつ作成できますか。

4,294,967,294室のルームの同時併存をサポートします。ルームの累計数は無制限です。

TRTCの遅延はどのくらいですか。

グローバルなエンドツーエンドの平均遅延は300ms未満です。

TRTCはPCでの画面共有をサポートしていますか。

サポートしています。次のドキュメントをご参照ください。

- 画面共有(Windows)

- 画面共有(Mac)
- 画面共有（Web）

画面共有インターフェースの詳細についてはWindows（C++）APIをご参照ください。また、Electronインターフェースもご利用いただけます。

TRTCはどのプラットフォームをサポートしますか。

サポートしているプラットフォームは、iOS、Android、Windows(C++)、Windows(C#)、Mac、Web、Electronです。詳細については、プラットフォームのサポートをご参照ください。

TRTCは最大何人の同時通話をサポートできますか。

- 通話モードでは、1ルームあたり最大300人の同時接続、最大50人のカメラまたはマイクの同時使用をサポートしています。
- ライブストリーミングモードでは、1ルームあたり視聴者10万人のオンライン視聴と、キャスター50人のカメラまたはマイクの使用をサポートしています。

TRTCはライブストリーミングのシーン類のアプリケーションをどのように実現しますか。

TRTCは、特にオンラインライブストリーミングのシーン向けに、10万人向けの低遅延ライブストリーミングソリューションをリリースしました。これにより、キャスターとマイク接続したキャスター間の最小遅延が200ms、通常の視聴者の遅延が1秒以内になるだけでなく、脆弱なネットワークに極めて高い耐性を持つこととなり、モバイル端末の複雑なネットワーク環境に対応します。
操作ガイドの詳細はライブストリーミングクイックスタートモードをご参照ください。

TRTCライブストリーミングはどのようなロールをサポートしていますか。ロールの違いは何ですか。

ライブストリーミングのシーン（TRTCAppSceneLIVEとTRTCAppSceneVoiceChatRoom）は、TRTCRoleAnchor（キャスター）とTRTCRoleAudience（視聴者）という2つのロールをサポートしています。違いとしては、キャスターのロールはオーディオ・ビデオデータのアップロードとダウンロードを同時に行うことができますが、視聴者のロールは他人のデータのダウンロードと再生のみをサポートしていることです。switchRole()を呼び出すことにより、ロールを切り替えることができます。

TRTCルームは、キックアウト、発言の禁止、ミュートをサポートしていますか。

サポートしています。

- 簡単なシグナリング操作の場合、TRTCのカスタムシグナルインターフェースsendCustomCmdMsgを使用できます。開発者が対応する制御シグナリングをカスタマイズして、制御シグナリングを受信した通話者が対応する操作を実行すればOKです。例えば、キックアウトとは、追い出しシグナリングを定義することであり、この信号を受信したユーザーは自動的に退室します。

- より完全な操作ロジックを実行する必要がある場合は、開発者がIMを使用して関連のロジックを実行し、TRTCルームとIMグループとのマッピングを行い、IMグループでカスタムメッセージを送受信して、対応する操作を実行することをお勧めします。

TRTCオーディオ・ビデオストリームは、CDNを介したプルストリームによる視聴をサポートしていますか。

サポートしています。

iOS端末はSwiftの統合をサポートしていますか？

サポートしています。サードパーティライブラリのフローにそのまま従ってSDKを統合すればOKです。また、Demoクイックスタート(iOS&Mac)も参照することができます。

デスクトップブラウザSDKはどのブラウザをサポートしていますか。

現在、デスクトップ版Chromeブラウザ、デスクトップ版Safariブラウザおよびモバイル版Safariブラウザのサポート状態は比較的万全です。その他のプラットフォーム（Androidプラットフォームのブラウザなど）のサポート状態はまだ不十分です。詳細については、サポートするプラットフォームをご参照ください。
ブラウザでWebRTC能力テストを開き、WebRTC機能を完全にサポートしているかテストすることができます。

Web端末SDKログのエラーメッセージのうち、NotFoundError、NotAllowedError、NotReadableError、OverConstrainedErrorおよびAbortErrorは、それぞれどういう意味ですか。

エラー名	説明	推奨する対処方法
NotFoundError	リクエストを満たすパラメータのメディアタイプ（オーディオ、ビデオ、画面共有を含む）が見つかりません。 例えば、PCにカメラがないのに、ブラウザにビデオストリームを取得するようリクエストがあった場合、このエラーが発生します。	ユーザーが通話を開始する前に、通話に必要なカメラやマイクなどのデバイスを確認することをお勧めします。カメラがなく、音声通話を行う必要がある場合は、TRTC.createStream({ audio: true, video: false }) で、マイクのみをキャプチャするように指定できます。
NotAllowedError	ユーザーが、現在のブラウザ・インスタンスのオーディオ、ビデオおよび画面共有へのアクセスのリクエストを拒否しました。	ユーザーに対し、カメラ/マイクへのアクセス権限を承認しないと、オーディオビデオ通話を行うことができません、というプロンプトが表示されます。

エラー名	説明	推奨する対処方法
NotReadableError	権限が付与されたユーザーが対応するデバイスを使用していますが、OS上のいずれかのハードウェア、ブラウザまたはWebページの階層に発生したエラーのため、デバイスにアクセスできません。	ブラウザのエラーメッセージに従って処理すると、ユーザーに対し、「現在カメラ/マイクにアクセスできません。他のアプリケーションがカメラ/マイクへのアクセスをリクエストしていないことを確認してから、もう一度お試しください」というプロンプトが表示されます。
OverConstrainedError	cameraId/microphoneIdのパラメータの値が無効です。	cameraId/microphoneIdの渡された値が正しく有効であることを確認してください。
AbortError	何らかの理由により、デバイスを使用できません。	-

詳細については、initializeをご参照ください。

Web端末SDKがデバイス（カメラ/マイク）リストを正常に取得できるかどうか確認するにはどうすればいいですか。

1. ブラウザがデバイスを正常に使用できるかチェックします。
   ページ上でコンソールを直接開き、 `navigator.mediaDevices.enumerateDevices()` と入力して、デバイスリストを取得できるかどうか確認します。

- デバイスを正常に取得できるとPromiseが返ってきて、その中に MediaDeviceInfoのオブジェクトの配列があります。配列の中の各オブジェクトが1つの使用可能なメディアデバイスに対応します。
- 列挙に失敗すると、Promiseは rejectedを戻し、これは、ブラウザがデバイスを識別できなかったことを表します。ブラウザまたはデバイスをチェックする必要があります。

2. デバイスリストを取得できる場合は、 `navigator.mediaDevices.getUserMedia({ audio: true, video: true })` と入力して、MediaStreamオブジェクトが正常に返されるかどうか確認します。正常に返されない場合、ブラウザがデータを取得していないことを示しているので、ブラウザの設定をチェックする必要があります。

ライブストリーミング、インタラクティブライブストリーミング、TRTCおよびRelayed live streamingの違いと関係性は何ですか。

- **LVB**（キーワード：一対多、RTMP/HLS/HTTP-FLV、CDN）
  LVBは、プッシュ端末、再生端末およびライブストリーミングクラウドサービスに分かれます。クラウドサービスはCDNを使用してライブストリームを配信します。プッシュには一般的な標準プロトコルRTMPが使用さ

れます。CDNによって配信された場合、再生するときには通常、RTMP、HTTP-FLVまたはHLS（H5サポート）を選択して視聴することができます。

- **ILV**（キーワード：マイク接続、PK）
  ILVBは、業務形式の1種で、キャスターと視聴者の間のインタラクティブなマイク接続や、キャスターとキャスターの間のインタラクティブなPKを行うライブストリーミングのタイプの1つです。
- **TRTC**（キーワード：マルチプレイヤーインタラクション、UDPプライベートプロトコル、低遅延）
  TRTCの主なユースケースは、オーディオとビデオのインタラクションと低遅延のライブストリーミングです。UDPベースのプライベートプロトコルを使用し、ディレイは100ミリ秒まで引き下げることができます。典型的なシーンは、zoom meeting、FaceTime、大規模セミナーなどです。TRTCはプラットフォーム全体をカバーし、iOS/Android/Windowsに加えて、WebRTCの相互運用性、クラウドミクスストリーミングという方法で、CDNへの画面のRelayed live streamingもサポートします。
- **Relayed live streaming**（キーワード：クラウドミクスストリーミング、RTCバイパス・プッシュ転送、CDN）
  Relayed live streamingとは、低遅延のマイク接続ルームにおけるマルチチャンネルのプッシュ画面をコピーして、クラウド内で画面を混合して一つのチャネルにし、ミクスストリーミング後の画面をライブCDNにプッシュして配信、再生する技術のことです。

TRTCは、通話時間と使用量をどのように確認すればいいですか。

TRTCコンソールの【使用量の統計】ページで確認することができます。

TRTCにラグが発生した場合はどのように調査すればいいですか。

通話品質は、対応するRoomIDとUserIDを用いて、TRTCコンソールの【監視ダッシュボード】ページで確認することができます。

- 受信端末の視点から送信端末と受信端末ユーザーの状況を確認します。
- 送信端末と受信端末のパケット損失率が高いかどうか確認します。パケット損失率が通常より高い場合、ネットワーク状態が不安定なためにラグが発生しています。
- フレームレートとCPU使用率を確認します。フレームレートが低くCPU使用率が高いと、ラグが発生します。

TRTCに画質の不良、ぼやけ、モザイクなどが発生する場合はどのように調査すればいいですか。

- 解像度は主にビットレートに関係しています。SDKのビットレートが低く設定されているか確認してください。解像度が高く、ビットレートが低いとモザイク現象が起こりやすくなります。
- TRTCは、クラウドQOSトラフィックコントロールポリシーを通じて、ネットワークの状態に応じ、ビットレートと解像度を動的に調整します。ネットワークが貧弱な場合、ビットレートが下がりやすくなり、解像度が低下します。
- 入室時にVideoCallモードとLiveモードのどちらを使用しているかチェックします。通話シーン向けのVideoCallモードは低遅延とスムーズさの維持に重点を置いています。したがって脆弱なネットワークの場合、スムーズさ

を確保するために画質が犠牲になりやすくなります。画質の方が大事なシーンには、Liveモードの使用をお勧めします。

SDKの最新のバージョン番号はどのように確認しますか。

- 自動ロードを使用する場合は、`latest.release`により最新バージョンとマッチングされて自動ロードが実行されるため、バージョン番号を変更する必要はありません。具体的な統合方法についてはSDKクイックインテグレーションをご参照ください。
- 現在のSDKの最新バージョン番号はリリースノートから確認することができます。以下をご参照ください。
  - iOS & Androidでは、リリースノート（App）をご参照ください。
  - デスクトップブラウザでは、リリースノート（Web）をご参照ください。
  - Electronでは、リリースノート（Electron）をご参照ください。

クライアント側 API
iOS and macOS
API概要

最終更新日：：2022-02-21 17:01:28

TRTCCloud @ TXLiteAVSDK

インスタンスの作成およびイベントコールバック

API	説明
sharedInstance	TRTCCloudインスタンスの作成（シングルトンモード）
destroySharedIntance	TRTCCloudインスタンスの破棄（シングルトンモード）
delegate	TRTCイベントコールバックを設定
delegateQueue	TRTCCloudDelegate イベントコールバックを起動するキューの設定

ルーム関連インターフェース関数

API	説明
enterRoom	ルームに入室
exitRoom	ルームを退室
switchRole	ロールの切り替え
switchRoom	ルームの切り替え
connectOtherRoom	ルーム間通話のリクエスト
disconnectOtherRoom	ルーム間通話を退出
setDefaultStreamRecvMode	サブスクライブモードを設定（有効にするには入室前に設定する必要があります）
createSubCloud	サブルーム事例の作成（複数のルームで同時視聴するために使用されます）
destroySubCloud	サブルーム事例の破棄

CDN関連インターフェース関数

API	説明
startPublishing	Tencent Cloud CSS CDNへのオーディオビデオストリーミングの公開を開始
stopPublishing	Tencent Cloud CSS CDNへのオーディオビデオストリーミングの公開を停止
startPublishCDNStream	非Tencent Cloud CDNへのオーディオビデオストリーミングの公開を開始
stopPublishCDNStream	非Tencent Cloud CDNへのオーディオビデオストリーミングの公開を停止
setMixTranscodingConfig	クラウドミクスストリーミングのレイアウトおよびトランスコードパラメータを設定

ビデオ関連インターフェース関数

API	説明
startLocalPreview	ローカルカメラのプレビュー画面を有効化（モバイル端末）
startLocalPreview	ローカルカメラのプレビュー画面を有効化（デスクトップ）
updateLocalView	ローカルカメラのプレビュー画面を更新
stopLocalPreview	カメラのプレビューを停止
muteLocalVideo	ローカルのビデオストリームの公開を一時停止/再開
setVideoMuteImage	ローカル画面の一時停止中の代替画像を設定
startRemoteView	リモートユーザーのビデオストリームをサブスクライブし、ビデオレンダリングウィジェットをバインド
updateRemoteView	リモートユーザーのビデオレンダリングウィジェットを更新
stopRemoteView	リモートユーザーのビデオストリームのサブスクライブを停止し、レンダリングウィジェットをリリース
stopAllRemoteView	すべてのリモートユーザーのビデオストリームのサブスクライブを停止し、すべてのレンダリングリソースをリリース
muteRemoteVideoStream	リモートユーザーのビデオストリームのサブスクライブを一時停止/再開
muteAllRemoteVideoStreams	すべてのリモートユーザーのビデオストリームのサブスクライブを一時停止/再開
setVideoEncoderParam	ビデオエンコーダのエンコードパラメータを設定

API	説明
setNetworkQosParam	ネットワーク品質モニタリングの関連パラメータを設定
setLocalRenderParams	ローカル画面のレンダリングパラメータを設定
setRemoteRenderParams	リモート画面のレンダリングモードを設定
setVideoEncoderRotation	ビデオエンコーダが出力する画面の方向を設定
setVideoEncoderMirror	エンコーダが出力する画面のイメージモードを設定
setGSensorMode	重力センサーの適合モードを設定
enableEncSmallVideoStream	大小画面のデュアルチャンネルコーディングモード を有効化
setRemoteVideoStreamType	指定リモートユーザーの画面サイズを切り替え
snapshotVideo	ビデオ画面のスクリーンキャプチャ

オーディオ関連インターフェース関数

API	説明
startLocalAudio	ローカルオーディオのキャプチャおよび公開を有効化
stopLocalAudio	ローカルオーディオのキャプチャおよび公開を停止
muteLocalAudio	ローカルのオーディオストリームの公開を一時停止/再開
muteRemoteAudio	リモートのオーディオストリームの再生を一時停止/再開
muteAllRemoteAudio	すべてのリモートユーザーのオーディオストリームの再生を一時停止/再開
setAudioRoute	オーディオルートを設定
setRemoteAudioVolume	特定リモートユーザーの音声再生音量を設定
setAudioCaptureVolume	ローカルオーディオのキャプチャ音量を設定
getAudioCaptureVolume	ローカルオーディオのキャプチャ音量を取得
setAudioPlayoutVolume	リモートオーディオの再生音量を設定
getAudioPlayoutVolume	リモートオーディオの再生音量を取得
enableAudioVolumeEvaluation	音声レベルのプロンプトを起動

API	説明
startAudioRecording	録音を開始
stopAudioRecording	録音を停止
startLocalRecording	ローカルメディアのレコーディングを有効化
stopLocalRecording	ローカルメディアのレコーディングを停止
setRemoteAudioParallelParams	リモートオーディオストリームのインテリジェント同時再生ポリシーを設定

デバイス管理関連インターフェース

API	説明
getDeviceManager	デバイス管理タイプ（TXDeviceManager）を取得

美顔・特殊効果および画像ウォーターマーク

API	説明
getBeautyManager	美顔管理タイプ（TXBeautyManager）を取得
setWatermark	ウォーターマークの追加

BGMおよび音声の特殊効果

API	説明
getAudioEffectManager	オーディオエフェクトマネージャー（TXAudioEffectManager）を取得
startSystemAudioLoopback	システム音声キャプチャ を有効化（Macシステムのみに適用）
stopSystemAudioLoopback	システム音声キャプチャを停止（デスクトップシステムのみに適用）
setSystemAudioLoopbackVolume	システム音声のキャプチャ音量を設定

画面共有関連インターフェース

API	説明
startScreenCaptureInApp	アプリケーション内の画面共有を開始（ iOS 13.0以上のシステムのみをサポート）

API	説明
startScreenCaptureByReplaykit	全システムの画面共有 を開始（ iOS 11.0以上のシステムのみをサポート）
startScreenCapture	デスクトップ画面共有を開始（このインターフェースはデスクトップシステムのみをサポート）
stopScreenCapture	画面共有を停止
pauseScreenCapture	画面共有を一時停止
resumeScreenCapture	画面共有を再開
getScreenCaptureSourcesWithThumbnailSize	共有可能な画面およびウィンドウを列挙（このインターフェースはMac OSシステムのみをサポート ）
selectScreenCaptureTarget	共有したい画面またはウィンドウを選択（このインターフェースはMac OSシステムのみをサポート）
setSubStreamEncoderParam	画面共有（サブストリーム）のビデオコーデックパラメータを設定（デスクトップシステムとモバイルシステムの両方をサポート）
setSubStreamMixVolume	画面共有時の音声ミキシングの音量レベルを設定（このインターフェースはデスクトップシステムのみをサポート）
addExcludedShareWindow	指定のウィンドウを画面共有のexcludeリストに追加（このインターフェースはデスクトップシステムのみをサポート）
removeExcludedShareWindow	指定のウィンドウを画面共有のexcludeリストから削除（このインターフェースはデスクトップシステムのみをサポート）
removeAllExcludedShareWindows	すべてのウィンドウを画面共有のexcludeリストから削除（このインターフェースはデスクトップシステムのみをサポート）
addIncludedShareWindow	指定のウィンドウを画面共有のincludeリストに追加（このインターフェースはデスクトップシステムのみをサポート）
removeIncludedShareWindow	指定のウィンドウを画面共有のincludeリストから削除（このインターフェースはデスクトップシステムのみをサポート）

API	説明
removeAllIncludedShareWindows	すべてのウィンドウを画面共有のincludeリストから削除（このインターフェースはデスクトップシステムのみをサポート）

ユーザー定義キャプチャおよびカスタムレンダリング

API	説明
enableCustomVideoCapture	ビデオユーザー定義キャプチャモード の起動/終了
sendCustomVideoData	自身がキャプチャしたビデオフレームをSDKに送信
enableCustomAudioCapture	オーディオのユーザー定義キャプチャモードを起動
sendCustomAudioData	自身がキャプチャしたオーディオデータをSDKに送信
enableMixExternalAudioFrame	ユーザー定義のオーディオトラックの起動/終了
mixExternalAudioFrame	ユーザー定義のオーディオトラックをSDKにミキシング
setMixExternalAudioVolume	プッシュ時にミキシングする外部オーディオのプッシュ音量および再生音量を設定
generateCustomPTS	ユーザー定義キャプチャ時のタイムスタンプを発行
setLocalVideoProcessDelegete	サードパーティによる美顔のビデオデータコールバックを設定
setLocalVideoRenderDelegate	ローカルビデオカスタムレンダリングコールバックを設定
setRemoteVideoRenderDelegate	リモートビデオカスタムレンダリングコールバックを設定
setAudioFrameDelegate	オーディオデータカスタムコールバックを設定
setCapturedRawAudioFrameDelegateFormat	ローカルマイクによってキャプチャされたオリジナルオーディオフレームコールバック形式を設定
setLocalProcessedAudioFrameDelegateFormat	前処理後のローカルオーディオフレームコールバック形式を設定
setMixedPlayAudioFrameDelegateFormat	最終的にシステムから再生したいオーディオフレームコールバック形式を設定
enableCustomAudioRendering	オーディオカスタム再生を有効化

API	説明
getCustomAudioRenderingFrame	再生可能なオーディオデータを取得

カスタムメッセージ送信インターフェース

API	説明
sendCustomCmdMsg	UDPチャネルを利用してカスタムメッセージをルーム内のすべてのユーザーに送信
sendSEIMsg	SEIチャネルを利用して送信カスタムメッセージをルーム内のすべてのユーザーに送信

ネットワークテストインターフェース

API	説明
startSpeedTest	ネットワークスピードテストを開始（入室前に使用）
stopSpeedTest	ネットワークスピードテストを停止

デバック関連インターフェース

API	説明
getSDKVersion	SDKのバージョン情報を取得
setLogLevel	Log出力レベルを設定
setConsoleEnabled	コンソールのログプリント を有効化/無効化
setLogCompressEnabled	ログのローカル圧縮を有効化/無効化
setLogDirPath	ローカルログの保存パスを設定
setLogDelegate	ログコールバックを設定
showDebugView	ダッシュボードを表示
setDebugViewMargin	ダッシュボードのマージンを設定
callExperimentalAPI	試験的インターフェースの呼び出し

破棄されたインターフェース

API	説明
setMicVolumeOnMixing	マイクの音量レベルを設定
setBeautyStyle	美顔、美白および肌の色調補正エフェクトレベルを設定
setEyeScaleLevel	デカ目レベルを設定
setFaceScaleLevel	小顔レベルを設定
setFaceVLevel	フェイスシェイプレベルを設定
setChinLevel	下あご引き伸ばしまたは縮小幅を設定
setFaceShortLevel	面長修正レベルを設定
setNoseSlimLevel	小鼻レベルを設定
selectMotionTmpl	動的エフェクトステッカーを設定
setMotionMute	動的エフェクトミュートを設定
startScreenCapture	画面共有を起動
setFilter	カラーフィルターエフェクトを設定
setFilterConcentration	カラーフィルター濃度を設定
setGreenScreenFile	クロマキー背景ビデオを設定
playBGM	BGMの再生を起動
stopBGM	BGMの再生を停止
pauseBGM	BGMの再生を停止
resumeBGM	BGMの再生を停止
getBGMDuration	BGMの総時間を取得（単位：ミリ秒）
setBGMPosition	BGM再生の進捗を設定
setBGMVolume	BGMの音量レベルを設定
setBGMPlayoutVolume	BGMのローカル再生音量を設定
setBGMPublishVolume	BGMのリモート再生音量を設定
setReverbType	リバーブエフェクトを設定

API	説明
setVoiceChangerType	ボイスチェンジタイプを設定
playAudioEffect	オーディオエフェクトを再生
setAudioEffectVolume	オーディオエフェクトの音量を設定
setAudioEffectVolume	オーディオエフェクトの再生を停止
stopAllAudioEffects	すべてのオーディオエフェクトを停止
setAllAudioEffectsVolume	すべてのオーディオエフェクト音量を設定
pauseAudioEffect	オーディオエフェクトを一時停止
resumeAudioEffect	オーディオエフェクトを一時停止
enableAudioEarMonitoring	インイヤーモニタリングを有効化（または無効化）
startRemoteView	リモートビデオ画面の表示を開始
stopRemoteView	リモートビデオ画面の表示を停止すると同時に、このリモートユーザーのビデオデータストリームのプルを停止
setRemoteViewFillMode	リモート画像のレンダリングモードを設定
setRemoteViewRotation	リモート画像の時計回りの回転角度を設定
setLocalViewFillMode	ローカル画像のレンダリングモードを設定
setLocalViewRotation	ローカル画像の時計回りの回転角度を設定
setLocalViewMirror	ローカルカメラプレビュー画面のイメージモードを設定
startRemoteSubStreamView	リモートユーザーのサブストリーム画面の表示を開始
stopRemoteSubStreamView	リモートユーザーのサブストリーム画面の表示を停止
setRemoteSubStreamViewFillMode	サブストリーム画面の塗りつぶしモードを設定
setRemoteSubStreamViewRotation	サブストリーム画面の時計回りの回転角度を設定
setPriorRemoteVideoStreamType	大画面または小画面の視聴優先順位を設定
setAudioQuality	オーディオ品質を設定
startLocalAudio	オーディオ品質を設定

API	説明
switchCamera	カメラの切り替え
isCameraZoomSupported	現在のカメラがズームをサポートしているかどうかを照会
setZoom	カメラズームの倍数（フォーカス距離）を設定
isCameraTorchSupported	フラッシュの切り替えをサポートしているかどうかを照会
enbaleTorch	フラッシュのオン/オフ
isCameraFocusPositionInPreviewSupported	カメラがフォーカスの設定をサポートしているかどうかを照会
setFocusPosition	カメラのフォーカス座標位置を設定
isCameraAutoFocusFaceModeSupported	顔の位置の自動認識をサポートしているかどうかを照会
enableAutoFaceFoucs	顔追尾フォーカスの有効化/無効化
startCameraDeviceTestInView	カメラテストを開始
stopCameraDeviceTest	カメラテストを開始
startMicDeviceTest	マイクテストを開始
stopMicDeviceTest	マイクテストを開始
startSpeakerDeviceTest	スピーカーテストを開始
stopSpeakerDeviceTest	スピーカーテストを停止
getMicDevicesList	マイクデバイスリストを取得
getCurrentMicDevice	現在のマイクデバイスを取得
setCurrentMicDevice	現在使用するマイクを選択
getCurrentMicDeviceVolume	現在のマイクのデバイス音量を取得
setCurrentMicDeviceVolume	現在のマイクのデバイス音量を設定
getCurrentMicDeviceMute	現在のシステムのマイクデバイスがミュートされているかどうかを取得
setCurrentMicDeviceMute	システムの現在のマイクデバイスのミュートステータスを設定

API	説明
getSpeakerDevicesList	スピーカーデバイスリストを取得
getCurrentSpeakerDevice	現在のスピーカーデバイスを取得
setCurrentSpeakerDevice	使用したいスピーカーを設定
getCurrentSpeakerDeviceVolume	現在のスピーカーのデバイス音量を取得
setCurrentSpeakerDeviceVolume	現在のスピーカーのデバイス音量を設定
getCurrentSpeakerDeviceMute	システムの現在のスピーカーデバイスがミュートされているかどうかを確認
setCurrentSpeakerDeviceMute	システムの現在のスピーカーデバイスのミュートステータスを設定
getCameraDevicesList	カメラデバイスリストを取得
getCurrentCameraDevice	現在使用しているカメラを取得
setCurrentCameraDevice	現在使用したいカメラを選択
setSystemVolumeType	システムの音量タイプを設定
snapshotVideo	ビデオスクリーンキャプチャ
enableCustomVideoCapture	ビデオのユーザー定義キャプチャモードを起動
sendCustomVideoData	自身がキャプチャしたビデオデータを送信
startScreenCaptureInApp	アプリケーション内の画面共有を開始（iOS）
startScreenCaptureByReplaykit	全システムの画面共有を開始（iOS）
muteLocalVideo	ローカルのビデオストリームの公開を一時停止/再開
muteRemoteVideoStream	リモートユーザーのビデオストリームのサブスクライブを一時停止/再開
startSpeedTest	ネットワークスピードテストを開始（入室前に使用）

エラーおよび警告イベント

API	説明
onError	エラーイベントコールバック

API	説明
onWarning	警告イベントコールバック

ルーム関連イベントコールバック

API	説明
onEnterRoom	入室成功または失敗のイベントコールバック
onExitRoom	退室のイベントコールバック
onSwitchRole	ロール切り替えのイベントコールバック
onSwitchRoom	ルーム切り替え結果のコールバック
onConnectOtherRoom	ルーム間通話リクエスト結果のコールバック
onDisconnectOtherRoom	ルーム間通話終了結果のコールバック

ユーザー関連イベントコールバック

API	説明
onRemoteUserEnterRoom	ユーザーが現在のルームに入室
onRemoteUserLeaveRoom	ユーザーが現在のルームを退室
onUserVideoAvailable	リモートユーザーが公開/キャンセルしたビッグストリームのビデオ画面
onUserSubStreamAvailable	リモートユーザーが公開/キャンセルしたサブストリームのビデオ画面
onUserAudioAvailable	リモートユーザーが公開/キャンセルした自身のオーディオ
onFirstVideoFrame	SDKが自身のローカルユーザーまたはリモートユーザーの最初のフレーム画面のレンダリングを開始
onFirstAudioFrame	SDKがリモートユーザーの最初のフレームのオーディオの再生を開始
onSendFirstLocalVideoFrame	自身のローカルの最初のビデオフレームが公開済み
onSendFirstLocalAudioFrame	自身のローカルの最初のオーディオフレームが公開済み
onRemoteVideoStatusUpdated	リモートビデオステータス変更のイベントコールバック

ネットワークおよび技術指標統計のコールバック

API	説明
onNetworkQuality	ネットワーク品質のリアルタイム統計のコールバック
onStatistics	オーディオビデオ技術指標のリアルタイム統計のコールバック
onSpeedTestResult	ネットワークスピードテストの結果のコールバック

クラウドとの接続状況のイベントコールバック

API	説明
onConnectionLost	SDKがクラウドとの接続を切断済み
onTryToReconnect	SDKがクラウドとの再接続を試行中
onConnectionRecovery	SDKがクラウドとの接続を再開済み

ハードウェアデバイス関連イベントコールバック

API	説明
onCameraDidReady	カメラの準備完了
onMicDidReady	マイクの準備完了
onAudioRouteChanged	現在のオーディオルートに変更発生（モバイルデバイスのみに適用）
onUserVoiceVolume	音量レベルフィードバックのコールバック
onDevice	ローカルデバイスのオン/オフステータスに変更発生（デスクトップシステムのみに適用）
onAudioDeviceCaptureVolumeChanged	現在のマイクのシステムキャプチャ音量に変更発生
onAudioDevicePlayoutVolumeChanged	現在のシステムの再生音量に変更発生
onSystemAudioLoopbackError	システム音声キャプチャが正常に開始されたかどうかのイベントコールバック（Macシステムのみに適用）

カスタムメッセージ受信イベントコールバック

API	説明
onRecvCustomCmdMsgUserId	カスタムメッセージ受信のイベントコールバック

API	説明
onMissCustomCmdMsgUserId	カスタムメッセージ消失のイベントコールバック
onRecvSEIMsg	SEIメッセージ受信のコールバック

CDN関連イベントコールバック

API	説明
onStartPublishing	Tencent Cloud CSS CDN上へのオーディオビデオストリーミングのイベントコールバックの公開を開始
onStopPublishing	Tencent Cloud CSS CDNへのオーディオビデオストリーミングのイベントコールバックの公開を停止
onStartPublishCDNStream	非Tencent Cloud CDNへのオーディオビデオストリーミングのイベントコールバックの公開を開始
onStopPublishCDNStream	非Tencent Cloud CDNへのオーディオビデオストリーミングのイベントコールバックの公開を停止
onSetMixTranscodingConfig	クラウドミクスストリーミングのレイアウトおよびトランスコードパラメータ設定のイベントコールバック

画面共有関連イベントコールバック

API	説明
onScreenCaptureStarted	画面共有開始のイベントコールバック
onScreenCapturePaused	画面共有一時停止のイベントコールバック
onScreenCaptureResumed	画面共有再開のイベントコールバック
onScreenCaptureStoped	画面共有停止のイベントコールバック

ローカルレコーディングおよびローカルスクリーンキャプチャのイベントコールバック

API	説明
onLocalRecordBegin	ローカルレコーディングタスク開始済みのイベントコールバック
onLocalRecording	ローカルレコーディングタスク実行中の進捗のイベントコールバック
onLocalRecordComplete	ローカルレコーディングタスク完了済みのイベントコールバック

破棄されたイベントコールバック

API	説明
onUserEnter	キャスターが現在のルームに入室（破棄済み）
onUserExit	キャスターが現在のルームを退室（破棄済み）
onAudioEffectFinished	オーディオエフェクト再生が完了済み（破棄済み）

ビデオデータカスタムコールバック

API	説明
onRenderVideoFrame	カスタムビデオレンダリングのコールバック
onProcessVideoFrame	サードパーティによる美顔コンポーネントを結合するためのビデオ処理のコールバック
onGLContextDestory	SDK内部 OpenGL環境破棄の通知

オーディオデータカスタムコールバック

API	説明
onCapturedRawAudioFrame	ローカルがキャプチャし、オーディオモジュールで前処理したオーディオデータのコールバック
onLocalProcessedAudioFrame	ローカルがキャプチャし、オーディオモジュールで前処理、音響処理およびBGMミキシングを行ったオーディオデータのコールバック
onRemoteUserAudioFrame	音声ミキシング前のリモートユーザーごとのオーディオデータ
onMixedPlayAudioFrame	各再生待ちオーディオをミキシングし、最終的にシステムに送信して再生する前のデータコールバック
onMixedAllAudioFrame	SDKのすべてのオーディオミキシング後のオーディオデータ（キャプチャしたデータおよび再生待ちのデータを含む）

その他イベントコールバックインターフェース

API	説明
onLog	ローカルLOG のプリントコールバック

ビデオ関連列挙値の定義

API	説明
TRTCVideoResolution	ビデオ解像度
TRTCVideoResolutionMode	ビデオアスペクト比モード
TRTCVideoStreamType	ビデオストリームタイプ
TRTCVideoFillMode	ビデオ画面塗りつぶしモード
TRTCVideoRotation	ビデオ画面回転方向
TRTCBeautyStyle	美顔（美肌）アルゴリズム
TRTCVideoPixelFormat	ビデオピクセル形式
TRTCVideoBufferType	ビデオデータ伝達方式
TRTCVideoMirrorType	ビデオのイメージタイプ
TRTCSnapshotSourceType	ローカルビデオスクリーンキャプチャのデータソース

ネットワーク関連列挙値の定義

API	説明
TRTCAppScene	ユースケース
TRTCRoleType	ロール
TRTCQosControlMode	トラフィックコントロールモード（破棄済み）
TRTCVideoQosPreference	画質の好み
TRTCQualityInfo	ネットワーク品質
TRTCAVStatusType	ビデオステータスタイプ
TRTCAVStatusChangeReason	ビデオステータス変更理由のタイプ

オーディオ関連列挙値の定義

API	説明
TRTCAudioSampleRate	オーディオサンプルレート

API	説明
TRTCAudioQuality	音声音質
TRTCAudioRoute	オーディオルート（音声の再生モード）
TRTCReverbType	音声リバーブモード
TRTCVoiceChangerType	ボイスチェンジタイプ
TRTCSystemVolumeType	システム音量タイプ（モバイルデバイスのみに適用）

その他列挙値の定義

API	説明
TRTCLogLevel	Logレベル
TRTCGSensorMode	重力センサースイッチ（モバイル端末のみに適用）
TRTCScreenCaptureSourceType	画面共有のターゲットタイプ（デスクトップのみに適用）
TRTCTranscodingConfigMode	クラウドミクスストリーミングのレイアウトモード
TRTCRecordType	メディアレコーディングタイプ
TRTCMixInputType	ミクスストリーミング入力タイプ
TRTCMediaDeviceType	デバイスタイプ（デスクトッププラットフォームのみに適用）
TRTCAudioRecordingContent	オーディオレコーディングコンテンツタイプ

TRTCコアタイプの定義

API	説明
TRTCParams	入室パラメータ
TRTCVideoEncParam	ビデオコーデックパラメータ
TRTCNetworkQosParam	ネットワークトラフィックコントロール（Qos）パラメータセット
TRTCRenderParams	ビデオ画面のレンダリングパラメータ
TRTCQualityInfo	ネットワーク品質
TRTCVolumeInfo	音量レベル

API	説明
TRTCSpeedTestParams	スピードテストのパラメータ
TRTCSpeedTestResult	ネットワークスピードテスト結果
TRTCVideoFrame	ビデオフレーム情報
TRTCAudioFrame	オーディオフレームデータ
TRTCMixUser	クラウドミクスストリーミングにおける各画面の説明情報
TRTCTranscodingConfig	クラウドミクスストリーミングのレイアウトおよびトランスコードパラメータ
TRTCPublishCDNParam	非Tencent Cloud CDNへのオーディオビデオストリーミングの公開時に設定が必要な転送パラメータ
TRTCAudioRecordingParams	ローカルオーディオファイルのレコーディングパラメータ
TRTCLocalRecordingParams	ローカルメディアファイルのレコーディングパラメータ
TRTCAudioEffectParam	オーディオエフェクトパラメータ（破棄済み）
TRTCSwitchRoomConfig	ルーム切り替えパラメータ
TRTCAudioFrameDelegateFormat	オーディオカスタムコールバックの形式パラメータ
TRTCScreenCaptureSourceInfo	画面共有のターゲット情報（デスクトップのみに適用）

Android
API概要

最終更新日：：2022-02-21 17:01:28

TRTCCloud @ TXLiteAVSDK

インスタンスの作成およびイベントコールバック

API	説明
sharedInstance	TRTCCloudインスタンスの作成（シングルトンモード）
destroySharedInstance	TRTCCloudインスタンスを破棄（シングルトンモード）
setListener	TRTCイベントコールバックを設定
setListenerHandler	TRTCCloudDelegate イベントコールバックを起動するキューを設定

ルーム関連インターフェース関数

API	説明
enterRoom	ルームに入室
exitRoom	ルームを退室
switchRole	ロールの切り替え
switchRoom	ルームの切り替え
ConnectOtherRoom	ルーム間通話のリクエスト
DisconnectOtherRoom	ルーム間通話を退出
setDefaultStreamRecvMode	サブスクライブモードを設定（有効にするには入室前に設定する必要があります）
createSubCloud	サブルーム事例の作成（複数のルームで同時視聴するために使用されます）
destroySubCloud	サブルーム事例の破棄

CDN関連インターフェース関数

API	説明
startPublishing	Tencent Cloud CSS CDNへのオーディオビデオストリーミングの公開を開始
stopPublishing	Tencent Cloud CSS CDNへのオーディオビデオストリーミングの公開を停止
startPublishCDNStream	非Tencent Cloud CDNへのオーディオビデオストリーミングの公開を開始
stopPublishCDNStream	非Tencent Cloud CDNへのオーディオビデオストリーミングの公開を停止
setMixTranscodingConfig	クラウドミクスストリーミングのレイアウトおよびトランスコードパラメータを設定

ビデオ関連インターフェース関数

API	説明
startLocalPreview	ローカルカメラのプレビュー画面を有効化（モバイル端末）
updateLocalView	ローカルカメラのプレビュー画面を更新
stopLocalPreview	カメラのプレビューを停止
muteLocalVideo	ローカルのビデオストリームの公開を一時停止/再開
setVideoMuteImage	ローカル画面の一時停止中の代替画像を設定
startRemoteView	リモートユーザーのビデオストリームをサブスクライブし、ビデオレンダリングウィジェットをバインド
updateRemoteView	リモートユーザーのビデオレンダリングウィジェットを更新
stopRemoteView	リモートユーザーのビデオストリームのサブスクライブを停止し、レンダリングウィジェットをリリース
stopAllRemoteView	すべてのリモートユーザーのビデオストリームのサブスクライブを停止し、すべてのレンダリングリソースをリリース
muteRemoteVideoStream	リモートユーザーのビデオストリームのサブスクライブを一時停止/再開
muteAllRemoteVideoStreams	すべてのリモートユーザーのビデオストリームのサブスクライブを一時停止/再開
setVideoEncoderParam	ビデオエンコーダのエンコードパラメータを設定
setNetworkQosParam	ネットワーク品質モニタリングの関連パラメータを設定
setLocalRenderParams	ローカル画面のレンダリングパラメータを設定

API	説明
setRemoteRenderParams	リモート画面のレンダリングモードを設定
setVideoEncoderRotation	ビデオエンコーダが出力する画面の方向を設定
setVideoEncoderMirror	エンコーダが出力する画面のイメージモードを設定
setGSensorMode	重力センサーの適合モードを設定
enableEncSmallVideoStream	大小画面のデュアルチャンネルコーディングモード を有効化
setRemoteVideoStreamType	指定リモートユーザーの大小画面を切り替え
snapshotVideo	ビデオ画面のスクリーンキャプチャ

オーディオ関連インターフェース関数

API	説明
startLocalAudio	ローカルオーディオのキャプチャおよび公開を有効化
stopLocalAudio	ローカルオーディオのキャプチャおよび公開を停止
muteLocalAudio	ローカルのオーディオストリームの公開を一時停止/再開
muteRemoteAudio	リモートのオーディオストリームの再生を一時停止/再開
muteAllRemoteAudio	すべてのリモートユーザーのオーディオストリームの再生を一時停止/再開
setAudioRoute	オーディオルートを設定
setRemoteAudioVolume	特定リモートユーザーの音声再生音量を設定
setAudioCaptureVolume	ローカルオーディオのキャプチャ音量を設定
getAudioCaptureVolume	ローカルオーディオのキャプチャ音量を取得
setAudioPlayoutVolume	リモートオーディオの再生音量を設定
getAudioPlayoutVolume	リモートオーディオの再生音量を取得
enableAudioVolumeEvaluation	音声レベルのプロンプトを起動
startAudioRecording	録音を開始
stopAudioRecording	録音を停止

API	説明
startLocalRecording	ローカルメディアのレコーディングを有効化
stopLocalRecording	ローカルメディアのレコーディングを停止
checkAudioCapabilitySupport	オーディオのある機能をサポートしているかどうかを照会（Androidのみに適用）
setRemoteAudioParallelParams	リモートオーディオストリームのインテリジェント同時再生ポリシーを設定

デバイス管理関連インターフェース

API	説明
getDeviceManager	デバイス管理タイプ（TXDeviceManager）を取得

美顔・特殊効果および画像ウォーターマーク

API	説明
getBeautyManager	美顔管理タイプ（TXBeautyManager）を取得
setWatermark	ウォーターマークの追加

BGMおよび音声の特殊効果

API	説明
getAudioEffectManager	オーディオエフェクトマネージャー（TXAudioEffectManager）を取得

画面共有関連インターフェース

API	説明
startScreenCapture	画面共有を起動
stopScreenCapture	画面共有を停止
pauseScreenCapture	画面共有を一時停止
resumeScreenCapture	画面共有を再開

API	説明
setSubStreamEncoderParam	画面共有（サブストリーム）のビデオコーデックパラメータを設定（デスクトップシステムとモバイルシステムの両方をサポート）

ユーザー定義キャプチャおよびカスタムレンダリング

API	説明
enableCustomVideoCapture	ビデオユーザー定義キャプチャモード の起動/終了
sendCustomVideoData	自身がキャプチャしたビデオフレームをSDKに送信
enableCustomAudioCapture	オーディオのユーザー定義キャプチャモードを起動
sendCustomAudioData	自身がキャプチャしたオーディオデータをSDKに送信
enableMixExternalAudioFrame	ユーザー定義のオーディオトラックの起動/終了
mixExternalAudioFrame	ユーザー定義のオーディオトラックをSDKにミキシング
setMixExternalAudioVolume	プッシュ時にミキシングする外部オーディオのプッシュ音量および再生音量を設定
generateCustomPTS	ユーザー定義キャプチャ時のタイムスタンプを発行
setLocalVideoProcessListener	サードパーティによる美顔のビデオデータコールバックを設定
setLocalVideoRenderListener	ローカルビデオカスタムレンダリングコールバックを設定
setRemoteVideoRenderListener	リモートビデオカスタムレンダリングコールバックを設定
setAudioFrameListener	オーディオデータカスタムコールバックを設定
setCapturedRawAudioFrameCallbackFormat	ローカルマイクによってキャプチャされたオリジナルオーディオフレームコールバック形式を設定
setLocalProcessedAudioFrameCallbackFormat	前処理後のローカルオーディオフレームコールバック形式を設定
setMixedPlayAudioFrameCallbackFormat	最終的にシステムから再生したいオーディオフレームコールバック形式を設定
enableCustomAudioRendering	オーディオカスタム再生を有効化
getCustomAudioRenderingFrame	再生可能なオーディオデータを取得

カスタムメッセージ送信インターフェース

API	説明
sendCustomCmdMsg	UDPチャネルを利用してカスタムメッセージをルーム内のすべてのユーザーに送信
sendSEIMsg	SEIチャネルを利用して送信カスタムメッセージをルーム内のすべてのユーザーに送信

ネットワークテストインターフェース

API	説明
startSpeedTest	ネットワークスピードテストを開始（入室前に使用）
stopSpeedTest	ネットワークスピードテストを停止

デバック関連インターフェース

API	説明
getSDKVersion	SDKのバージョン情報を取得
setLogLevel	Log出力レベルを設定
setConsoleEnabled	コンソールのログプリントを有効化/無効化
setLogCompressEnabled	ログのローカル圧縮を有効化/無効化
setLogDirPath	ローカルログの保存パスを設定
setLogListener	ログコールバックを設定
showDebugView	ダッシュボードを表示
setDebugViewMargin	ダッシュボードのマージンを設定
callExperimentalAPI	試験的インターフェースの呼び出し
setNetEnv	TRTCのバックエンドクラスターを設定（Tencent Cloud研究開発チームのみに適用）

破棄されたインターフェース

API	説明

API	説明
setMicVolumeOnMixing	マイクの音量レベルを設定
setBeautyStyle	美顔、美白および肌の色調補正エフェクトレベルを設定
setEyeScaleLevel	デカ目レベルを設定
setFaceSlimLevel	小顔レベルを設定
setFaceVLevel	フェイスシェイプレベルを設定
setChinLevel	下あご引き伸ばしまたは縮小幅を設定
setFaceShortLevel	面長修正レベルを設定
setNoseSlimLevel	小鼻レベルを設定
selectMotionTmpl	動的エフェクトステッカーを設定
setMotionMute	動的エフェクトミュートを設定
setFilter	カラーフィルターエフェクトを設定
setFilterConcentration	カラーフィルター濃度を設定
setGreenScreenFile	クロマキー背景ビデオを設定
playBGM	BGMの再生を起動
stopBGM	BGMの再生を停止
pauseBGM	BGMの再生を停止
resumeBGM	BGMの再生を停止
getBGMDuration	BGMの総時間を取得（単位：ミリ秒）
setBGMPosition	BGM再生の進捗を設定
setBGMVolume	BGMの音量レベルを設定
setBGMPlayoutVolume	BGMのローカル再生音量を設定
setBGMPublishVolume	BGMのリモート再生音量を設定
setReverbType	リバーブエフェクトを設定
setVoiceChangerType	ボイスチェンジタイプを設定

API	説明
playAudioEffect	オーディオエフェクトを再生
setAudioEffectVolume	オーディオエフェクトの音量を設定
setAudioEffectVolume	オーディオエフェクトの再生を停止
stopAllAudioEffects	すべてのオーディオエフェクトを停止
setAllAudioEffectsVolume	すべてのオーディオエフェクト音量を設定
pauseAudioEffect	オーディオエフェクトを一時停止
resumeAudioEffect	オーディオエフェクトを一時停止
enableAudioEarMonitoring	インイヤーモニタリングを有効化（または無効化）
startRemoteView	リモートビデオ画面の表示を開始
stopRemoteView	リモートビデオ画面の表示を停止すると同時に、このリモートユーザーのビデオデータストリームのプルを停止
setRemoteViewFillMode	リモート画像のレンダリングモードを設定
setRemoteViewRotation	リモート画像の時計回りの回転角度を設定
setLocalViewFillMode	ローカル画像のレンダリングモードを設定
setLocalViewRotation	ローカル画像の時計回りの回転角度を設定
setLocalViewMirror	ローカルカメラプレビュー画面のイメージモードを設定
startRemoteSubStreamView	リモートユーザーのサブストリーム画面の表示を開始
stopRemoteSubStreamView	リモートユーザーのサブストリーム画面の表示を停止
setRemoteSubStreamViewFillMode	サブストリーム画面の塗りつぶしモードを設定
setRemoteSubStreamViewRotation	サブストリーム画面の時計回りの回転角度を設定
setPriorRemoteVideoStreamType	大画面または小画面の視聴優先順位を設定
setAudioQuality	オーディオ品質を設定
startLocalAudio	オーディオ品質を設定
switchCamera	カメラの切り替え

API	説明
isCameraZoomSupported	現在のカメラがズームをサポートしているかどうかを照会
setZoom	カメラズームの倍数（フォーカス距離）を設定）
isCameraTorchSupported	フラッシュの切り替えをサポートしているかどうかを照会
enableTorch	フラッシュのオン/オフ
isCameraFocusPositionInPreviewSupported	カメラがフォーカスの設定をサポートしているかどうかを照会
setFocusPosition	カメラのフォーカス座標位置を設定
isCameraAutoFocusFaceModeSupported	顔の位置の自動認識をサポートしているかどうかを照会
setSystemVolumeType	システムの音量タイプを設定
enableCustomVideoCapture	ビデオユーザー定義キャプチャモード の起動
sendCustomVideoData	自身がキャプチャしたビデオデータを送信
startScreenCapture	画面共有を起動（Android）
muteLocalVideo	ローカルのビデオストリームの公開を一時停止/再開
muteRemoteVideoStream	リモートユーザーのビデオストリームのサブスクライブを一時停止/再開
startSpeedTest	ネットワークスピードテストを開始（入室前に使用）

エラーおよび警告イベント

API	説明
onError	エラーイベントコールバック
onWarning	警告イベントコールバック

ルーム関連イベントコールバック

API	説明
onEnterRoom	入室成功または失敗のイベントコールバック
onExitRoom	退室のイベントコールバック

API	説明
onSwitchRole	ロール切り替えのイベントコールバック
onSwitchRoom	ルーム切り替え結果のコールバック
onConnectOtherRoom	ルーム間通話リクエスト結果のコールバック
onDisConnectOtherRoom	ルーム間通話終了結果のコールバック

ユーザー関連イベントコールバック

API	説明
onRemoteUserEnterRoom	ユーザーが現在のルームに入室
onRemoteUserLeaveRoom	ユーザーが現在のルームを退室
onUserVideoAvailable	リモートユーザーが公開/キャンセルしたビッグストリームのビデオ画面
onUserSubStreamAvailable	リモートユーザーが公開/キャンセルしたサブストリームのビデオ画面
onUserAudioAvailable	リモートユーザーが公開/キャンセルした自身のオーディオ
onFirstVideoFrame	SDKが自身のローカルユーザーまたはリモートユーザーの最初のフレーム画面のレンダリングを開始
onFirstAudioFrame	SDKがリモートユーザーの最初のフレームのオーディオの再生を開始
onSendFirstLocalVideoFrame	自身のローカルの最初のビデオフレームが公開済み
onSendFirstLocalAudioFrame	自身のローカルの最初のオーディオフレームが公開済み
onRemoteVideoStatusUpdated	リモートビデオステータス変更のイベントコールバック

ネットワークおよび技術指標統計のコールバック

API	説明
onNetworkQuality	ネットワーク品質のリアルタイム統計のコールバック
onStatistics	オーディオビデオ技術指標のリアルタイム統計のコールバック
onSpeedTestResult	ネットワークスピードテストの結果のコールバック

クラウドとの接続状況のイベントコールバック

API	説明
onConnectionLost	SDKがクラウドとの接続を切断済み
onTryToReconnect	SDKがクラウドとの再接続を試行中
onConnectionRecovery	SDKがクラウドとの接続を再開済み

ハードウェアデバイス関連イベントコールバック

API	説明
onCameraDidReady	カメラの準備完了
onMicDidReady	マイクの準備完了
onAudioRouteChanged	現在のオーディオルートに変更発生（モバイルデバイスのみに適用）
onUserVoiceVolume	音量レベルフィードバックのコールバック

カスタムメッセージ受信イベントコールバック

API	説明
onRecvCustomCmdMsg	カスタムメッセージ受信のイベントコールバック
onMissCustomCmdMsg	カスタムメッセージ消失のイベントコールバック
onRecvSEIMsg	SEIメッセージ受信のコールバック

CDN関連イベントコールバック

API	説明
onStartPublishing	Tencent Cloud CSS CDNへのオーディオビデオストリーミングのイベントコールバックの公開を開始
onStopPublishing	Tencent Cloud CSS CDNへのオーディオビデオストリーミングのイベントコールバックの公開を停止
onStartPublishCDNStream	非Tencent Cloud CDNへのオーディオビデオストリーミングのイベントコールバックの公開を開始
onStopPublishCDNStream	非Tencent Cloud CDNへのオーディオビデオストリーミングのイベントコールバックの公開を停止

API	説明
onSetMixTranscodingConfig	クラウドミクスストリーミングのレイアウトおよびトランスコードパラメータ設定のイベントコールバック

画面共有関連イベントコールバック

API	説明
onScreenCaptureStarted	画面共有開始のイベントコールバック
onScreenCapturePaused	画面共有一時停止のイベントコールバック
onScreenCaptureResumed	画面共有再開のイベントコールバック
onScreenCaptureStopped	画面共有停止のイベントコールバック

ローカルレコーディングおよびローカルスクリーンキャプチャのイベントコールバック

API	説明
onLocalRecordBegin	ローカルレコーディングタスク開始済みのイベントコールバック
onLocalRecording	ローカルレコーディングタスク実行中の進捗のイベントコールバック
onLocalRecordComplete	ローカルレコーディングタスク完了済みのイベントコールバック

破棄されたイベントコールバック

API	説明
onUserEnter	キャスターが現在のルームに入室（破棄済み）
onUserExit	キャスターが現在のルームを退室（破棄済み）
onAudioEffectFinished	オーディオエフェクト再生が完了済み（破棄済み）
onSpeedTest	サーバースピードテストの結果のコールバック（破棄済み）

ビデオデータカスタムコールバック

API	説明
onRenderVideoFrame	カスタムビデオレンダリングのコールバック

API	説明
onGLContextCreated	SDK内部のOpenGL環境作成済みの通知
onProcessVideoFrame	サードパーティによる美顔コンポーネントを結合するためのビデオ処理のコールバック
onGLContextDestory	SDK内部のOpenGL環境破棄の通知

オーディオデータカスタムコールバック

API	説明
onCapturedRawAudioFrame	ローカルがキャプチャし、オーディオモジュールで前処理したオーディオデータのコールバック
onLocalProcessedAudioFrame	ローカルがキャプチャし、オーディオモジュールで前処理、音響処理およびBGMミキシングを行ったオーディオデータのコールバック
onRemoteUserAudioFrame	音声ミキシング前のリモートユーザーごとのオーディオデータ
onMixedPlayAudioFrame	各再生待ちオーディオをミキシングし、最終的にシステムに送信して再生する前のデータコールバック
onMixedAllAudioFrame	SDKのすべてのオーディオミキシング後のオーディオデータ（キャプチャしたデータおよび再生待ちのデータを含む）

その他イベントコールバックインターフェース

API	説明
onLog	ローカルLOGのプリントコールバック

ビデオ関連列挙値の定義

API	説明
TRTCVideoResolution	ビデオ解像度
TRTCVideoResolutionMode	ビデオアスペクト比モード
TRTCVideoStreamType	ビデオストリームタイプ
TRTCVideoFillMode	ビデオ画面塗りつぶしモード
TRTCVideoRotation	ビデオ画面回転方向

API	説明
TRTCBeautyStyle	美顔（美肌）アルゴリズム
TRTCVideoPixelFormat	ビデオピクセル形式
TRTCVideoBufferType	ビデオデータ伝達方式
TRTCVideoMirrorType	ビデオのイメージタイプ
TRTCSnapshotSourceType	ローカルビデオスクリーンキャプチャのデータソース

ネットワーク関連列挙値の定義

API	説明
TRTCAppScene	ユースケース
TRTCRoleType	ロール
TRTCQosControlMode	トラフィックコントロールモード（破棄済み）
TRTCVideoQosPreference	画質の好み
TRTCQuality	ネットワーク品質
TRTCAVStatusType	ビデオステータスタイプ
TRTCAVStatusChangeReason	ビデオステータス変更理由のタイプ

オーディオ関連列挙値の定義

API	説明
TRTCAudioSampleRate	オーディオサンプルレート
TRTCAudioQuality	音声品質
TRTCAudioRoute	オーディオルート（音声の再生モード）
TRTCReverbType	音声リバーブモード
TRTCVoiceChangerType	ボイスチェンジタイプ
TRTCSystemVolumeType	システム音量タイプ（モバイルデバイスのみに適用）

API	説明
TRTCAudioCapabilityType	システムがサポートするオーディオ機能タイプ（Androidデバイスのみに適用）

その他列挙値の定義

API	説明
TRTCLogLevel	Logレベル
TRTCGSensorMode	重力センサースイッチ（モバイル端末のみに適用）
TRTCTranscodingConfigMode	クラウドミクスストリーミングのレイアウトモード
TRTCRecordType	メディアレコーディングタイプ
TRTCMixInputType	ミクスストリーミング入力タイプ
TRTCAudioRecordingContent	オーディオレコーディングコンテンツタイプ

TRTCコアタイプの定義

API	説明
TRTCParams	入室パラメータ
TRTCVideoEncParam	ビデオコーデックパラメータ
TRTCNetworkQosParam	ネットワークトラフィックコントロール（Qos）パラメータセット
TRTCRenderParams	ビデオ画面のレンダリングパラメータ
TRTCQualityInfo	ネットワーク品質
TRTCVolumeInfo	音量レベル
TRTCSpeedTestParams	スピードテストパラメータ
TRTCSpeedTestResult	ネットワークスピードテスト結果
TRTCTexture	ビデオテクスチャデータ（Androidプラットフォームのみに適用。テクスチャIDおよびEGL環境を含む）
TRTCVideoFrame	ビデオフレーム情報
TRTCAudioFrame	オーディオフレームデータ

API	説明
TRTCMixUser	クラウドミクスストリーミングにおける各画面の説明情報
TRTCTranscodingConfig	クラウドミクスストリーミングのレイアウトおよびトランスコードパラメータ
TRTCPublishCDNParam	非Tencent Cloud CDNへのオーディオビデオストリーミングの公開時に設定が必要な転送パラメータ
TRTCAudioRecordingParams	ローカルオーディオファイルのレコーディングパラメータ
TRTCLocalRecordingParams	ローカルメディアファイルのレコーディングパラメータ
TRTCAudioEffectParam	オーディオエフェクトパラメータ（破棄済み）
TRTCSwitchRoomConfig	ルーム切り替えパラメータ
TRTCAudioFrameCallbackFormat	オーディオカスタムコールバックの形式パラメータ
TRTCScreenShareParams	画面共有パラメータ（Androidプラットフォームのみに適用）

エラーコード

最終更新日：：2022-06-28 16:42:03

エラーコードリスト

基本的なエラーコード

記号	値	意味
ERR_NULL	0	エラーなし

入室関連のエラーコード

TRTCCloud.enterRoom() 入室に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.onEnterRoom()、TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_ENTER_FAIL	-3301	入室の失敗
ERR_ENTER_ROOM_PARAM_NULL	-3316	入室パラメータが空です。TRTCCloud.enterRoom():インターフェースの呼び出しが有効なparamで渡されるかどうか確認してください
ERR_SDK_APPID_INVALID	-3317	入室パラメータsdkAppIdエラー
ERR_ROOM_ID_INVALID	-3318	入室パラメータroomIdエラー
ERR_USER_ID_INVALID	-3319	入室パラメータuserIDが正しくありません
ERR_USER_SIG_INVALID	-3320	入室パラメータuserSigが正しくありません
ERR_ROOM_REQUEST_ENTER_ROOM_TIMEOUT	-3308	入室リクエストがタイムアウトしました。ネットワークをご確認ください
ERR_SERVER_INFO_PRIVILEGE_FLAG_ERROR	-100006	パーミッションビットのチェックに失敗しました。privateMapKeyが正しいかどうか確認してください

記号	値	意味
ERR_SERVER_INFO_SERVICE_SUSPENDED	-100013	サービスをご利用いただけません。パッケージの残りの分数が0より大きいかどうか、Tencent Cloudアカウントの支払いが遅延していないかどうかご確認ください
ERR_SERVER_INFO_ECDH_GET_TINYID	-100018	userSigのチェックに失敗しました。userSigが正しいかどうか確認してください

退室関連のエラーコード

TRTCCloud.exitRoom() 入室に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.onEnterRoom()、TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_REQUEST_QUIT_ROOM_TIMEOUT	-3325	退室リクエストのタイムアウト

デバイス（カメラ、マイク、スピーカー）関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_CAMERA_START_FAIL	-1301	カメラの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、カメラのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください
ERR_CAMERA_NOT_AUTHORIZED	-1314	カメラデバイスの権限が承認されていません。通常はモバイルデバイスに表示されます。ユーザーから権限承認を拒否されている可能性があります
ERR_CAMERA_SET_PARAM_FAIL	-1315	カメラパラメータ設定エラー（パラメータがサポートされていないか、その他）
ERR_CAMERA_OCCUPY	-1316	カメラが使用中なので、他のカメラの起動を試みることができます

記号	値	意味
ERR_MIC_START_FAIL	-1302	マイクの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、マイクのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください
ERR_MIC_NOT_AUTHORIZED	-1317	マイクデバイスの権限が承認されていません。通常はモバイルデバイスに表示されます。ユーザーから権限承認を拒否されている可能性があります
ERR_MIC_SET_PARAM_FAIL	-1318	マイクパラメータの設定に失敗しました
ERR_MIC_OCCUPY	-1319	マイクが使用中です。例えば、モバイルデバイスが通話中の場合、マイクの起動は失敗します
ERR_MIC_STOP_FAIL	-1320	マイクの停止に失敗しました
ERR_SPEAKER_START_FAIL	-1321	スピーカーの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、スピーカーのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください
ERR_SPEAKER_SET_PARAM_FAIL	-1322	スピーカーパラメータの設定に失敗しました
ERR_SPEAKER_STOP_FAIL	-1323	スピーカーの停止に失敗しました

画面共有関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味

記号	値	意味
ERR_SCREEN_CAPTURE_START_FAIL	-1308	画面録画の開始に失敗しました。モバイルデバイスに表示されるときは、ユーザーから権限承認を拒否されている可能性があります。WindowsまたはMacシステムのデバイスに表示される場合は、画面録画インターフェースのパラメータが要件を満たしているかご確認ください
ERR_SCREEN_CAPTURE_UNSURPORT	-1309	画面録画に失敗しました。Androidプラットフォームでは5.0以降のシステム、iOSプラットフォームでは11.0以降のシステムが必要です
ERR_SERVER_CENTER_NO_PRIVILEDGE_PUSH_SUB_VIDEO	-102015	サブチャネルのアップストリームの権限がありません
ERR_SERVER_CENTER_ANOTHER_USER_PUSH_SUB_VIDEO	-102016	他のユーザーがサブチャネルのアップストリームを行っています
ERR_SCREEN_CAPTURE_STOPPED	-7001	画面録画がシステムによって停止されました

コーデック関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_VIDEO_ENCODE_FAIL	-1303	ビデオフレームのエンコードに失敗しました。例えば、iOSデバイスを他のアプリケーションに切り替えると、システムによってハードウェアエンコーダが解放される場合があります。切り替えて元に戻すと、ハードウェアエンコーダが再起動する前にスローされる場合があります

記号	値	意味
ERR_UNSUPPORTED_RESOLUTION	-1305	サポートされていないビデオ解像度
ERR_AUDIO_ENCODE_FAIL	-1304	オーディオフレームのエンコードに失敗しました。例えば、渡されたカスタムオーディオデータを、SDKで処理することはできません
ERR_UNSUPPORTED_SAMPLERATE	-1306	サポートされていないオーディオサンプルレート

ユーザー定義キャプチャ関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_PIXEL_FORMAT_UNSUPPORTED	-1327	設定されたpixel formatはサポートされていません
ERR_BUFFER_TYPE_UNSUPPORTED	-1328	設定されたbuffer typeはサポートされていません

CDNバインディングおよびミクスストリーミング関連のエラーコード

コールバック関数TRTCCloudDelegate.onStartPublishing()、TRTCCloudDelegate.onSetMixTranscodingConfig()によって関連通知をキャプチャできます。

記号	値	意味
ERR_PUBLISH_CDN_STREAM_REQUEST_TIME_OUT	-3321	バイパス転送リクエストのタイムアウト
ERR_CLOUD_MIX_TRANSCODING_REQUEST_TIME_OUT	-3322	クラウドミクスストリーミングリクエストのタイムアウト
ERR_PUBLISH_CDN_STREAM_SERVER_FAILED	-3323	バイパス転送リターンパケットの異常
ERR_CLOUD_MIX_TRANSCODING_SERVER_FAILED	-3324	クラウドミクスストリーミング・リターンパケットの異常
ERR_ROOM_REQUEST_START_PUBLISHING_TIMEOUT	-3333	Tencent CloudへのライブCDNのプッシュ開始シグナリングのタイムアウト
ERR_ROOM_REQUEST_START_PUBLISHING_ERROR	-3334	Tencent CloudへのライブCDNのプッシュ開始シグナリングの異常

記号	値	意味
ERR_ROOM_REQUEST_STOP_PUBLISHING_TIMEOUT	-3335	Tencent Cloudへのライブ CDNのプッシュ停止シグナリングのタイムアウト
ERR_ROOM_REQUEST_STOP_PUBLISHING_ERROR	-3336	Tencent Cloudへのライブ CDNのプッシュ停止シグナリングの異常

ルーム間マイク接続関連のエラーコード

TRTCCloud.ConnectOtherRoom() ルーム間マイク接続に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.onConnectOtherRoom()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_REQUEST_CONN_ROOM_TIMEOUT	-3326	マイク接続リクエストのタイムアウト
ERR_ROOM_REQUEST_DISCONN_ROOM_TIMEOUT	-3327	マイク接続からの退出リクエストのタイムアウト
ERR_ROOM_REQUEST_CONN_ROOM_INVALID_PARAM	-3328	無効なパラメータ
ERR_CONNECT_OTHER_ROOM_AS_AUDIENCE	-3330	現在のロールは視聴者であり、ルーム間マイク接続をリクエストまたは切断することはできません。まずキャスターにswitchRole()する必要があります
ERR_SERVER_CENTER_CONN_ROOM_NOT_SUPPORT	-102031	ルーム間マイク接続はサポートされていません

記号	値	意味
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_NUM	-102032	ルーム間マイク接続の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_RETRY_TIMES	-102033	ルーム間マイク接続の再試行回数の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_REQ_TIMEOUT	-102034	ルーム間マイク接続リクエストのタイムアウト
ERR_SERVER_CENTER_CONN_ROOM_REQ	-102035	ルーム間マイク接続リクエストの形式エラー
ERR_SERVER_CENTER_CONN_ROOM_NO_SIG	-102036	ルーム間マイク接続に署名がありません
ERR_SERVER_CENTER_CONN_ROOM_DECRYPT_SIG	-102037	ルーム間マイク接続の署名復号に失敗しました
ERR_SERVER_CENTER_CONN_ROOM_NO_KEY	-102038	ルーム間マイク接続の署名復号キーが見つかりませんでした
ERR_SERVER_CENTER_CONN_ROOM_PARSE_SIG	-102039	ルーム間マイク接続の署名解析エラー
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SIG_TIME	-102040	ルーム間マイク接続の署名タイムスタンプエラー
ERR_SERVER_CENTER_CONN_ROOM_SIG_GROUPID	-102041	ルーム間マイク接続の署名ルーム番号が一致しません

記号	値	意味
ERR_SERVER_CENTER_CONN_ROOM_NOT_CONNED	-102042	ルーム間マイク接続の署名ユーザー名が一致しません
ERR_SERVER_CENTER_CONN_ROOM_USER_NOT_CONNED	-102043	このユーザーはマイク接続を開始していません
ERR_SERVER_CENTER_CONN_ROOM_FAILED	-102044	ルーム間マイク接続に失敗しました
ERR_SERVER_CENTER_CONN_ROOM_CANCEL_FAILED	-102045	ルーム間マイク接続の取り消しに失敗しました
ERR_SERVER_CENTER_CONN_ROOM_CONNED_ROOM_NOT_EXIST	-102046	マイク接続されたルームが存在しません
ERR_SERVER_CENTER_CONN_ROOM_CONNED_REACH_MAX_ROOM	-102047	マイク接続されたルームがマイク接続の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_NOT_EXIST	-102048	マイク接続されたユーザーが存在しません
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_DELETED	-102049	マイク接続されたユーザーが削除されました
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_FULL	-102050	マイク接続されたユーザーがリソースの上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SEQ	-102051	マイク接続リクエストの番号が乱れています

アラートコード表

アラートコードを特に注意する必要はありません。必要に応じて、現在のユーザーに対応するプロンプトを表示するかどうかを選択できます。

記号	値	意味
WARNING_HW_ENCODER_START_FAIL	1103	ハードウェアエンコーディングの起動に問題が発生した場合、自動的にソフトウェアエンコーディングに切り替わります
WARNING_VIDEO_ENCODER_SW_TO_HW	1107	現在のCPU使用率が高すぎてソフトウェアエンコーディング要件を満たせないため、自動的にハードウェアエンコーディングに切り替わります
WARNING_INSUFFICIENT_CAPTURE_FPS	1108	カメラのキャプチャフレームレートが不足しています。美顔アルゴリズムを備えた一部のAndroidスマートフォンには表示されます
WARNING_SW_ENCODER_START_FAIL	1109	ソフトウェアエンコーディングの開始に失敗しました
WARNING_REDUCE_CAPTURE_RESOLUTION	1110	現在のフレームレートとパフォーマンスの最適解を満たすため、カメラのキャプチャ解像度が低下します。
WARNING_CAMERA_DEVICE_EMPTY	1111	使用可能なカメラデバイスが検出されません
WARNING_CAMERA_NOT_AUTHORIZED	1112	ユーザーは、現在のアプリケーションにカメラ使用権限を承認していません
WARNING_MICROPHONE_DEVICE_EMPTY	1201	使用可能なマイクデバイスが検出されません
WARNING_SPEAKER_DEVICE_EMPTY	1202	使用可能なスピーカーデバイスが検出されません
WARNING_MICROPHONE_NOT_AUTHORIZED	1203	ユーザーは、現在のアプリケーションにマイク使用権限を承認していません

記号	値	意味
WARNING_MICROPHONE_DEVICE_ABNORMAL	1204	オーディオキャプチャデバイスが利用できません（使用中であるなど）
WARNING_SPEAKER_DEVICE_ABNORMAL	1205	オーディオ再生デバイスが利用できません（使用中であるなど）
WARNING_VIDEO_FRAME_DECODE_FAIL	2101	現在のビデオフレームのデコードに失敗しました
WARNING_AUDIO_FRAME_DECODE_FAIL	2102	現在のオーディオフレームのデコードに失敗しました
WARNING_VIDEO_PLAY_LAG	2105	現在、ビデオ再生時にラグが発生しています
WARNING_HW_DECODER_START_FAIL	2106	ハードウェアデコードの開始に失敗しました。ソフトウェアデコードを使用してください
WARNING_VIDEO_DECODER_HW_TO_SW	2108	現在のストリームの最初のIフレームのハードウェアデコードが失敗しました。SDKは自動的にソフトウェアデコードに切り替えます
WARNING_SW_DECODER_START_FAIL	2109	ソフトウェアデコーダの起動に失敗しました
WARNING_VIDEO_RENDER_FAIL	2110	ビデオレンダリングに失敗しました
WARNING_START_CAPTURE_IGNORED	4000	キャプチャ中のため、キャプチャの開始は無視されます
WARNING_AUDIO_RECORDING_WRITE_FAIL	7001	オーディオ記録のファイルへの書き込みに失敗しました
WARNING_ROOM_DISCONNECT	5101	ネットワーク接続が切断されました
WARNING_IGNORE_UPSTREAM_FOR_AUDIENCE	6001	現在、視聴者ロールです。アップストリームのオーディオビデオデータを無視します
WARNING_NET_BUSY	1101	ネットワーク状態の不良：アップストリーム帯域幅が小さすぎて、データのアップロード時に問題が生じます

記号	値	意味
WARNING_RTMP_SERVER_RECONNECT	1102	CSSプッシュのときにネットワークが切断しましたが、自動再接続が開始されました（自動再接続は連続3回を超えて失敗すると、それ以上は行われません）
WARNING_LIVE_STREAM_SERVER_RECONNECT	2103	ライブストリーミングのプルのときにネットワークが切断しましたが、自動再接続が開始されました（自動再接続は連続3回を超えて失敗すると、それ以上は行われません）
WARNING_RECV_DATA_LAG	2104	不安定なネットワークパケット：ダウンストリーム帯域幅が不足しているか、キャスター側からのストリーミングが不均一であることが原因と思われます
WARNING_RTMP_DNS_FAIL	3001	ライブストリーミング、DNS解決に失敗しました
WARNING_RTMP_SEVER_CONN_FAIL	3002	ライブストリーミング、サーバーの接続に失敗しました
WARNING_RTMP_SHAKE_FAIL	3003	ライブストリーミング、RTMPサーバーとのハンドシェイクに失敗しました
WARNING_RTMP_SERVER_BREAK_CONNECT	3004	ライブストリーミング、サーバーが自動的に切断しました
WARNING_RTMP_READ_WRITE_FAIL	3005	ライブストリーミング、RTMPの読み取り/書き込みに失敗し、接続が切断されます
WARNING_RTMP_WRITE_FAIL	3006	ライブストリーミング、RTMP書き込みエラー（SDK内部エラーコードは外部にスローされません）
WARNING_RTMP_READ_FAIL	3007	ライブストリーミング、RTMP読み取りエラー（SDK内部エラーコードは外部にスローされません）

記号	値	意味
WARNING_RTMP_NO_DATA	3008	ライブストリーミング、30秒以上データが送信されない場合、自動的に接続が切断されます
WARNING_PLAY_LIVE_STREAM_INFO_CONNECT_FAIL	3009	ライブストリーミング、connectサーバー呼び出しに失敗しました（SDK内部エラーコードは外部にスローされません）
WARNING_NO_STEAM_SOURCE_FAIL	3010	ライブストリーミング、接続に失敗しました。このストリームアドレスにビデオはありません。（SDK内部エラーコードは外部にスローされません）
WARNING_ROOM_RECONNECT	5102	ネットワークが切断され、自動再接続が開始されました
WARNING_ROOM_NET_BUSY	5103	ネットワーク状態の不良：アップストリーム帯域幅が小さすぎて、データのアップロード時に問題が生じます

All Platforms (C++)
API概要

最終更新日：：2022-02-21 17:01:28

C++ 全プラットフォームインターフェースの紹介

バージョン8.0より、従来のWindows（C++）インターフェースをベースとした、新しいC++ インターフェースを提供しています。Windows、iOS、Mac、Androidプラットフォームに適用します。
C++ インターフェースを統合する方法がわからない場合は、各プラットフォームの統合ガイドをご参照ください。

- iOS C++インターフェース統合ガイド
- Android C++インターフェース統合ガイド
- Mac C++インターフェース統合ガイド
- Windows統合ガイド

説明：

- 現在C++インターフェースは簡易版（TRTC）の中でのみ提供されています。
- Windowsプラットフォームでは、TRTCヘッダーファイルで自動的に「trtc」ネームスペースを使用するようになっていますので、あらためて指定する必要はありません。

ITRTCCloud @ TXLiteAVSDK

インスタンスの作成およびイベントコールバック

API	説明
getTRTCShareInstance	TRTCCloudインスタンスの作成（シングルトンモード）
destroyTRTCShareInstance	TRTCCloudインスタンスを破棄（シングルトンモード）
addCallback	TRTCイベントコールバックを設定
removeCallback	TRTCイベントコールバックを削除

ルーム関連インターフェース関数

API	説明
enterRoom	ルームに入室
exitRoom	ルームを退室
switchRole	ロールの切り替え
switchRoom	ルームの切り替え
connectOtherRoom	ルーム間通話のリクエスト
disconnectOtherRoom	ルーム間通話を退出
setDefaultStreamRecvMode	サブスクライブモードを設定（有効にするには入室前に設定する必要があります）
createSubCloud	サブルーム事例の作成（複数のルームで同時視聴するために使用されます）
destroySubCloud	サブルーム事例の破棄

CDN関連インターフェース関数

API	説明
startPublishing	Tencent Cloud CSS CDNへのオーディオビデオストリーミングの公開を開始
stopPublishing	Tencent Cloud CSS CDNへのオーディオビデオストリーミングの公開を停止
startPublishCDNStream	非Tencent Cloud CDNへのオーディオビデオストリーミングの公開を開始
stopPublishCDNStream	非Tencent Cloud CDNへのオーディオビデオストリーミングの公開を停止
setMixTranscodingConfig	クラウドミクスストリーミングのレイアウトおよびトランスコードパラメータを設定

ビデオ関連インターフェース関数

API	説明
startLocalPreview	ローカルカメラのプレビュー画面を有効化（モバイル端末）
startLocalPreview	ローカルカメラのプレビュー画面を有効化（デスクトップ）
updateLocalView	ローカルカメラのプレビュー画面を更新

API	説明
stopLocalPreview	カメラのプレビューを停止
muteLocalVideo	ローカルのビデオストリームの公開を一時停止/再開
startRemoteView	リモートユーザーのビデオストリームをサブスクライブし、ビデオレンダリングウィジェットをバインド
updateRemoteView	リモートユーザーのビデオレンダリングウィジェットを更新
stopRemoteView	リモートユーザーのビデオストリームのサブスクライブを停止し、レンダリングウィジェットをリリース
stopAllRemoteView	すべてのリモートユーザーのビデオストリームのサブスクライブを停止し、すべてのレンダリングリソースをリリース
muteRemoteVideoStream	リモートユーザーのビデオストリームのサブスクライブを一時停止/再開
muteAllRemoteVideoStreams	すべてのリモートユーザーのビデオストリームのサブスクライブを一時停止/再開
setVideoEncoderParam	ビデオエンコーダのエンコードパラメータを設定
setNetworkQosParam	ネットワーク品質モニタリングの関連パラメータを設定
setLocalRenderParams	ローカル画面のレンダリングパラメータを設定
setRemoteRenderParams	リモート画面のレンダリングモードを設定
setVideoEncoderRotation	ビデオエンコーダが出力する画面の方向を設定
setVideoEncoderMirror	エンコーダが出力する画面のイメージモードを設定
enableSmallVideoStream	大小画面のデュアルチャンネルコーディングモードを有効化
setRemoteVideoStreamType	指定リモートユーザーの大小画面を切り替え
snapshotVideo	ビデオ画面のスクリーンキャプチャ

オーディオ関連インターフェース関数

API	説明
startLocalAudio	ローカルオーディオのキャプチャおよび公開を有効化
stopLocalAudio	ローカルオーディオのキャプチャおよび公開を停止

API	説明
muteLocalAudio	ローカルのオーディオストリームの公開を一時停止/再開
muteRemoteAudio	リモートのオーディオストリームの再生を一時停止/再開
muteAllRemoteAudio	すべてのリモートユーザーのオーディオストリームの再生を一時停止/再開
setRemoteAudioVolume	特定リモートユーザーの音声再生音量を設定
setAudioCaptureVolume	ローカルオーディオのキャプチャ音量を設定
getAudioCaptureVolume	ローカルオーディオのキャプチャ音量を取得
setAudioPlayoutVolume	リモートオーディオの再生音量を設定
getAudioPlayoutVolume	リモートオーディオの再生音量を設定
enableAudioVolumeEvaluation	音量レベルのプロンプトを起動
startAudioRecording	録音を開始
stopAudioRecording	録音を停止
startLocalRecording	ローカルメディアのレコーディングを有効化
stopLocalRecording	ローカルメディアのレコーディングを停止
setRemoteAudioParallelParams	リモートオーディオストリームのインテリジェント同時再生ポリシーを設定

デバイス管理関連インターフェース

API	説明
getDeviceManager	デバイス管理タイプ（TXDeviceManager）を取得

美顔・特殊効果および画像ウォーターマーク

API	説明
setBeautyStyle	美顔、美白および肌の色調補正エフェクトレベルを設定
setWaterMark	ウォーターマークの追加

BGMおよび音声の特殊効果

API	説明
getAudioEffectManager	オーディオエフェクトマネージャー（TXAudioEffectManager）を取得
startSystemAudioLoopback	システム音声キャプチャを有効化（デスクトップシステムのみに適用）
stopSystemAudioLoopback	システム音声キャプチャを停止（デスクトップシステムのみに適用）
setSystemAudioLoopbackVolume	システム音声のキャプチャ音量を設定

画面共有関連インターフェース

API	説明
startScreenCapture	デスクトップ画面共有を開始（このインターフェースはデスクトップシステムのみをサポート）
stopScreenCapture	画面共有を停止
pauseScreenCapture	画面共有を一時停止
resumeScreenCapture	画面共有を再開
getScreenCaptureSources	共有可能な画面およびウィンドウを列挙（このインターフェースはデスクトップシステムのみをサポート）
selectScreenCaptureTarget	共有したい画面またはウィンドウを選択（このインターフェースはデスクトップシステムのみをサポート）
setSubStreamEncoderParam	画面共有（サブストリーム）のビデオコーデックパラメータを設定（デスクトップシステムとモバイルシステムの両方をサポート）
setSubStreamMixVolume	画面共有時の音声ミキシングの音量レベルを設定（このインターフェースはデスクトップシステムのみをサポート）
addExcludedShareWindow	指定のウィンドウを画面共有のexcludeリストに追加（このインターフェースはデスクトップシステムのみをサポート）
removeExcludedShareWindow	指定のウィンドウを画面共有のexcludeリストから削除（このインターフェースはデスクトップシステムのみをサポート）
removeAllExcludedShareWindow	すべてのウィンドウを画面共有のexcludeリストから削除（このインターフェースはデスクトップシステムのみをサポート）
addIncludedShareWindow	指定のウィンドウを画面共有のincludeリストに追加（このインターフェースはデスクトップシステムのみをサポート）

API	説明
removeIncludedShareWindow	指定のウィンドウを画面共有のincludeリストから削除（このインターフェースはデスクトップシステムのみをサポート）
removeAllIncludedShareWindow	すべてのウィンドウを画面共有のincludeリストから削除（このインターフェースはデスクトップシステムのみをサポート）

ユーザー定義キャプチャおよびカスタムレンダリング

API	説明
enableCustomVideoCapture	ビデオユーザー定義キャプチャモードの起動/終了
sendCustomVideoData	自身がキャプチャしたビデオフレームをSDKに送信
enableCustomAudioCapture	オーディオのユーザー定義キャプチャモードを起動
sendCustomAudioData	自身がキャプチャしたオーディオデータをSDKに送信
enableMixExternalAudioFrame	ユーザー定義のオーディオトラックの起動/終了
mixExternalAudioFrame	ユーザー定義のオーディオトラックをSDKにミキシング
setMixExternalAudioVolume	プッシュ時にミキシングする外部オーディオのプッシュ音量および再生音量を設定
generateCustomPTS	ユーザー定義キャプチャ時のタイムスタンプを発行
setLocalVideoProcessCallback	サードパーティによる美顔のビデオデータコールバックを設定
setLocalVideoRenderCallback	ローカルビデオカスタムレンダリングコールバックを設定
setRemoteVideoRenderCallback	リモートビデオカスタムレンダリングコールバックを設定
setAudioFrameCallback	オーディオデータカスタムコールバックを設定
setMixedPlayAudioFrameCallbackFormat	最終的にシステムから再生したいオーディオフレームコールバック形式を設定
enableCustomAudioRendering	オーディオカスタム再生を有効化
getCustomAudioRenderingFrame	再生可能なオーディオデータを取得

カスタムメッセージ送信インターフェース

API	説明

API	説明
sendCustomCmdMsg	UDPチャネルを利用してカスタムメッセージをルーム内のすべてのユーザーに送信
sendSEIMsg	SEIチャネルを利用して送信カスタムメッセージをルーム内のすべてのユーザーに送信

ネットワークテストインターフェース

API	説明
startSpeedTest	ネットワークスピードテストを開始（入室前に使用）
stopSpeedTest	ネットワークスピードテストを停止

デバック関連インターフェース

API	説明
getSDKVersion	SDKのバージョン情報を取得
setLogLevel	Log出力レベルを設定
setConsoleEnabled	コンソールのログプリントを有効化/無効化
setLogCompressEnabled	ログのローカル圧縮を有効化/無効化
setLogDirPath	ローカルログの保存パスを設定
setLogCallback	ログコールバックを設定
showDebugView	ダッシュボードを表示
callExperimentalAPI	試験的インターフェースの呼び出し

破棄されたインターフェース

API	説明
enableCustomVideoCapture	ビデオのユーザー定義キャプチャモード を起動
sendCustomVideoData	自身がキャプチャしたビデオデータを送信
muteLocalVideo	ローカルのビデオストリームの公開を一時停止/再開
muteRemoteVideoStream	リモートユーザーのビデオストリームのサブスクライブを一時停止 / 再開

API	説明
startSpeedTest	ネットワークスピードテストを開始（入室前に使用）

エラーおよび警告イベント

API	説明
onError	エラーイベントコールバック
onWarning	警告イベントコールバック

ルーム関連イベントコールバック

API	説明
onEnterRoom	入室成功または失敗のイベントコールバック
onExitRoom	退室のイベントコールバック
onSwitchRole	ロール切り替えのイベントコールバック
onSwitchRoom	ルーム切り替え結果のコールバック
onConnectOtherRoom	ルーム間通話リクエスト結果のコールバック
onDisconnectOtherRoom	ルーム間通話終了結果のコールバック

ユーザー関連イベントコールバック

API	説明
onRemoteUserEnterRoom	ユーザーが現在のルームに入室
onRemoteUserLeaveRoom	ユーザーが現在のルームを退室
onUserVideoAvailable	リモートユーザーが公開/キャンセルしたビッグストリームのビデオ画面
onUserSubStreamAvailable	リモートユーザーが公開/キャンセルしたサブストリームのビデオ画面
onUserAudioAvailable	リモートユーザーが公開/キャンセルした自身のオーディオ
onFirstVideoFrame	SDKが自身のローカルユーザーまたはリモートユーザーの最初のフレーム画面のレンダリングを開始
onFirstAudioFrame	SDKがリモートユーザーの最初のフレームのオーディオの再生を開始

API	説明
onSendFirstLocalVideoFrame	自身のローカルの最初のビデオフレームが公開済み
onSendFirstLocalAudioFrame	自身のローカルの最初のオーディオフレームが公開済み
onRemoteVideoStatusUpdated	リモートビデオステータス変更のイベントコールバック

ネットワークおよび技術指標統計のコールバック

API	説明
onNetworkQuality	ネットワーク品質のリアルタイム統計のコールバック
onStatistics	オーディオビデオ技術指標のリアルタイム統計のコールバック
onSpeedTestResult	ネットワークスピードテストの結果のコールバック

クラウドとの接続状況のイベントコールバック

API	説明
onConnectionLost	SDKがクラウドとの接続を切断済み
onTryToReconnect	SDKがクラウドとの再接続を試行中
onConnectionRecovery	SDKがクラウドとの接続を再開済み

ハードウェアデバイス関連イベントコールバック

API	説明
onCameraDidReady	カメラの準備完了
onMicDidReady	マイクの準備完了
onUserVoiceVolume	音量レベルフィードバックのコールバック
onDeviceChange	ローカルデバイスのオン/オフステータスに変更発生（デスクトップシステムのみに適用）
onAudioDeviceCaptureVolumeChanged	現在のマイクのシステムキャプチャ音量に変更発生
onAudioDevicePlayoutVolumeChanged	現在のシステムの再生音量に変更発生

API	説明
onSystemAudioLoopbackError	システム音声キャプチャが正常に開始されたかどうかのイベントコールバック（Macシステムのみに適用）
onTestMicVolume	マイクテスト時の音量のコールバック
onTestSpeakerVolume	スピーカーテスト時の音量のコールバック

カスタムメッセージ受信イベントコールバック

API	説明
onRecvCustomCmdMsg	カスタムメッセージ受信のイベントコールバック
onMissCustomCmdMsg	カスタムメッセージ消失のイベントコールバック
onRecvSEIMsg	SEIメッセージ受信のコールバック

CDN関連イベントコールバック

API	説明
onStartPublishing	Tencent Cloud CSS CDNへのオーディオビデオストリーミングのイベントコールバックの公開を開始
onStopPublishing	Tencent Cloud CSS CDNへのオーディオビデオストリーミングの公開停止のイベントコールバック
onStartPublishCDNStream	非Tencent Cloud CDNへのオーディオビデオストリーミングのイベントコールバックの公開を開始
onStopPublishCDNStream	非Tencent Cloud CDNへのオーディオビデオストリーミングの公開停止のイベントコールバック
onSetMixTranscodingConfig	クラウドミクスストリーミングのレイアウトおよびトランスコードパラメータ設定のイベントコールバック

画面共有関連イベントコールバック

API	説明
onScreenCaptureStarted	画面共有開始のイベントコールバック
onScreenCapturePaused	画面共有一時停止のイベントコールバック

API	説明
onScreenCaptureResumed	画面共有再開のイベントコールバック
onScreenCaptureStoped	画面共有停止のイベントコールバック
onScreenCaptureCovered	画面共有のターゲットウィンドウブロックのイベントコールバック（Windows OSのみに適用）

ローカルレコーディングおよびローカルスクリーンキャプチャのイベントコールバック

API	説明
onLocalRecordBegin	ローカルレコーディングタスク開始済みのイベントコールバック
onLocalRecording	ローカルレコーディングタスク実行中の進捗のイベントコールバック
onLocalRecordComplete	ローカルレコーディングタスク完了済みのイベントコールバック
onSnapshotComplete	ローカルスクリーンキャプチャ完了のイベントコールバック

破棄されたイベントコールバック

API	説明
onUserEnter	キャスターが現在のルームに入室（破棄済み）
onUserExit	キャスターが現在のルームを退室（破棄済み）
onAudioEffectFinished	オーディオエフェクト再生が完了済み（破棄済み）
onPlayBGMBegin	BGMの再生を開始（破棄済み）
onPlayBGMProgress	BGMの再生進捗のコールバック（破棄済み）
onPlayBGMComplete	BGMの再生が完了済み（破棄済み）
onSpeedTest	サーバースピードテストの結果のコールバック（破棄済み）

ビデオデータカスタムコールバック

API	説明
onRenderVideoFrame	カスタムビデオレンダリングのコールバック

API	説明
onProcessVideoFrame	サードパーティによる美顔コンポーネントを結合するためのビデオ処理のコールバック

オーディオデータカスタムコールバック

API	説明
onCapturedRawAudioFrame	ローカルがキャプチャし、オーディオモジュールで前処理したオーディオデータのコールバック
onLocalProcessedAudioFrame	ローカルがキャプチャし、オーディオモジュールで前処理、音響処理およびBGMミキシングを行ったオーディオデータのコールバック
onPlayAudioFrame	音声ミキシング前のリモートユーザーごとのオーディオデータ
onMixedPlayAudioFrame	各再生待ちオーディオをミキシングし、最終的にシステムに送信して再生する前のデータコールバック

その他イベントコールバックインターフェース

API	説明
onLog	ローカルLOGのプリントコールバック

ビデオ関連列挙値の定義

API	説明
TRTCVideoResolution	ビデオ解像度
TRTCVideoResolutionMode	ビデオアスペクト比モード
TRTCVideoStreamType	ビデオストリームタイプ
TRTCVideoFillMode	ビデオ画面塗りつぶしモード
TRTCVideoRotation	ビデオ画面回転方向
TRTCBeautyStyle	美顔（美肌）アルゴリズム
TRTCVideoPixelFormat	ビデオピクセル形式
TRTCVideoBufferType	ビデオデータ伝達方式

API	説明
TRTCVideoMirrorType	ビデオのイメージタイプ
TRTCSnapshotSourceType	ローカルビデオスクリーンキャプチャのデータソース

ネットワーク関連列挙値の定義

API	説明
TRTCAppScene	ユースケース
TRTCRoleType	ロール
TRTCQosControlMode	トラフィックコントロールモード（破棄済み）
TRTCVideoQosPreference	画質の好み
TRTCQualityInfo	ネットワーク品質
TRTCAVStatusType	ビデオステータスタイプ
TRTCAVStatusChangeReason	ビデオステータス変更理由のタイプ

オーディオ関連列挙値の定義

API	説明
TRTCAudioQuality	音声音質

その他列挙値の定義

API	説明
TRTCLogLevel	Logレベル
TRTCScreenCaptureSourceType	画面共有のターゲットタイプ（デスクトップのみに適用）
TRTCTranscodingConfigMode	クラウドミクスストリーミングのレイアウトモード
TRTCLocalRecordType	メディアレコーディングタイプ
TRTCMixInputType	ミクスストリーミング入力タイプ
TRTCDeviceType	デバイスタイプ（デスクトッププラットフォームのみに適用）

API	説明
TRTCAudioRecordingContent	オーディオレコーディングコンテンツタイプ

TRTCコアタイプの定義

API	説明
TRTCParams	入室パラメータ
TRTCVideoEncParam	ビデオコーデックパラメータ
TRTCNetworkQosParam	ネットワークトラフィックコントロール（Qos）パラメータセット
TRTCRenderParams	ビデオ画面のレンダリングパラメータ
TRTCQualityInfo	ネットワーク品質
TRTCVolumeInfo	音量レベル
TRTCSpeedTestParams	スピードテストのパラメータ
TRTCSpeedTestResult	ネットワークスピードテスト結果
TRTCVideoFrame	ビデオフレーム情報
TRTCAudioFrame	オーディオフレームデータ
TRTCMixUser	クラウドミクスストリーミングにおける各画面の説明情報
TRTCTranscodingConfig	クラウドミクスストリーミングのレイアウトおよびトランスコードパラメータ
TRTCPublishCDNParam	非Tencent Cloud CDNへのオーディオビデオストリーミングの公開時に設定が必要な転送パラメータ
TRTCAudioRecordingParams	ローカルオーディオファイルのレコーディングパラメータ
TRTCLocalRecordingParams	ローカルメディアファイルのレコーディングパラメータ
TRTCAudioEffectParam	オーディオエフェクトパラメータ（破棄済み）
TRTCSwitchRoomConfig	ルーム切り替えパラメータ
TRTCAudioFrameCallbackFormat	オーディオカスタムコールバックの形式パラメータ
TRTCScreenCaptureSourceInfo	画面共有のターゲット情報（デスクトップのみに適用）

API	説明
ITRTCScreenCaptureSourceList	共有可能な画面およびウィンドウのリスト

エラーコード

最終更新日：：2022-06-28 16:43:46

エラーコードリスト

基本的なエラーコード

記号	値	意味
ERR_NULL	0	エラーなし

入室関連のエラーコード

TRTCCloud.enterRoom() 入室に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.onEnterRoom()、TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_ENTER_FAIL	-3301	入室の失敗
ERR_ENTER_ROOM_PARAM_NULL	-3316	入室パラメータが空です。TRTCCloud.enterRoom():インターフェースの呼び出しが有効なparamで渡されるかどうか確認してください
ERR_SDK_APPID_INVALID	-3317	入室パラメータsdkAppIdエラー
ERR_ROOM_ID_INVALID	-3318	入室パラメータroomIdエラー
ERR_USER_ID_INVALID	-3319	入室パラメータuserIDが正しくありません
ERR_USER_SIG_INVALID	-3320	入室パラメータuserSigが正しくありません
ERR_ROOM_REQUEST_ENTER_ROOM_TIMEOUT	-3308	入室リクエストがタイムアウトしました。ネットワークをご確認ください
ERR_SERVER_INFO_PRIVILEGE_FLAG_ERROR	-100006	パーミッションビットのチェックに失敗しました。privateMapKeyが正しいかどうか確認してください

記号	値	意味
ERR_SERVER_INFO_SERVICE_SUSPENDED	-100013	サービスをご利用いただけません。パッケージの残りの分数が0より大きいかどうか、Tencent Cloudアカウントの支払いが遅延していないかどうかご確認ください
ERR_SERVER_INFO_ECDH_GET_TINYID	-100018	userSigのチェックに失敗しました。userSigが正しいかどうか確認してください

退室関連のエラーコード

TRTCCloud.exitRoom() 入室に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.onEnterRoom()、TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_REQUEST_QUIT_ROOM_TIMEOUT	-3325	退室リクエストのタイムアウト

デバイス（カメラ、マイク、スピーカー）関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_CAMERA_START_FAIL	-1301	カメラの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、カメラのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください
ERR_CAMERA_NOT_AUTHORIZED	-1314	カメラデバイスの権限が承認されていません。通常はモバイルデバイスに表示されます。ユーザーから権限承認を拒否されている可能性があります
ERR_CAMERA_SET_PARAM_FAIL	-1315	カメラパラメータ設定エラー（パラメータがサポートされていないか、その他）
ERR_CAMERA_OCCUPY	-1316	カメラが使用中なので、他のカメラの起動を試みることができます

記号	値	意味
ERR_MIC_START_FAIL	-1302	マイクの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、マイクのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください
ERR_MIC_NOT_AUTHORIZED	-1317	マイクデバイスの権限が承認されていません。通常はモバイルデバイスに表示されます。ユーザーから権限承認を拒否されている可能性があります
ERR_MIC_SET_PARAM_FAIL	-1318	マイクパラメータの設定に失敗しました
ERR_MIC_OCCUPY	-1319	マイクが使用中です。例えば、モバイルデバイスが通話中の場合、マイクの起動は失敗します
ERR_MIC_STOP_FAIL	-1320	マイクの停止に失敗しました
ERR_SPEAKER_START_FAIL	-1321	スピーカーの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、スピーカーのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください
ERR_SPEAKER_SET_PARAM_FAIL	-1322	スピーカーパラメータの設定に失敗しました
ERR_SPEAKER_STOP_FAIL	-1323	スピーカーの停止に失敗しました

画面共有関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味

記号	値	意味
ERR_SCREEN_CAPTURE_START_FAIL	-1308	画面録画の開始に失敗しました。モバイルデバイスに表示されるときは、ユーザーから権限承認を拒否されている可能性があります。WindowsまたはMacシステムのデバイスに表示される場合は、画面録画インターフェースのパラメータが要件を満たしているかご確認ください
ERR_SCREEN_CAPTURE_UNSURPORT	-1309	画面録画に失敗しました。Androidプラットフォームでは5.0以降のシステム、iOSプラットフォームでは11.0以降のシステムが必要です
ERR_SERVER_CENTER_NO_PRIVILEDGE_PUSH_SUB_VIDEO	-102015	サブチャネルのアップストリームの権限がありません
ERR_SERVER_CENTER_ANOTHER_USER_PUSH_SUB_VIDEO	-102016	他のユーザーがサブチャネルのアップストリームを行っています
ERR_SCREEN_CAPTURE_STOPPED	-7001	画面録画がシステムによって停止されました

コーデック関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_VIDEO_ENCODE_FAIL	-1303	ビデオフレームのエンコードに失敗しました。例えば、iOSデバイスを他のアプリケーションに切り替えると、システムによってハードウェアエンコーダが解放される場合があります。切り替えて元に戻すと、ハードウェアエンコーダが再起動する前にスローされる場合があります

記号	値	意味
ERR_UNSUPPORTED_RESOLUTION	-1305	サポートされていないビデオ解像度
ERR_AUDIO_ENCODE_FAIL	-1304	オーディオフレームのエンコードに失敗しました。例えば、渡されたカスタムオーディオデータを、SDKで処理することはできません
ERR_UNSUPPORTED_SAMPLERATE	-1306	サポートされていないオーディオサンプルレート

ユーザー定義キャプチャ関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_PIXEL_FORMAT_UNSUPPORTED	-1327	設定されたpixel formatはサポートされていません
ERR_BUFFER_TYPE_UNSUPPORTED	-1328	設定されたbuffer typeはサポートされていません

CDNバインディングおよびミクスストリーミング関連のエラーコード

コールバック関数TRTCCloudDelegate.onStartPublishing()、TRTCCloudDelegate.onSetMixTranscodingConfig()によって関連通知をキャプチャできます。

記号	値	意味
ERR_PUBLISH_CDN_STREAM_REQUEST_TIME_OUT	-3321	バイパス転送リクエストのタイムアウト
ERR_CLOUD_MIX_TRANSCODING_REQUEST_TIME_OUT	-3322	クラウドミクスストリーミングリクエストのタイムアウト
ERR_PUBLISH_CDN_STREAM_SERVER_FAILED	-3323	バイパス転送リターンパケットの異常
ERR_CLOUD_MIX_TRANSCODING_SERVER_FAILED	-3324	クラウドミクスストリーミング・リターンパケットの異常
ERR_ROOM_REQUEST_START_PUBLISHING_TIMEOUT	-3333	Tencent CloudへのライブCDNのプッシュ開始シグナリングのタイムアウト
ERR_ROOM_REQUEST_START_PUBLISHING_ERROR	-3334	Tencent CloudへのライブCDNのプッシュ開始シグナリングの異常

記号	値	意味
ERR_ROOM_REQUEST_STOP_PUBLISHING_TIMEOUT	-3335	Tencent Cloudへのライブ CDNのプッシュ停止シグナリングのタイムアウト
ERR_ROOM_REQUEST_STOP_PUBLISHING_ERROR	-3336	Tencent Cloudへのライブ CDNのプッシュ停止シグナリングの異常

ルーム間マイク接続関連のエラーコード

TRTCCloud.ConnectOtherRoom() ルーム間マイク接続に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.onConnectOtherRoom()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_REQUEST_CONN_ROOM_TIMEOUT	-3326	マイク接続リクエストのタイムアウト
ERR_ROOM_REQUEST_DISCONN_ROOM_TIMEOUT	-3327	マイク接続からの退出リクエストのタイムアウト
ERR_ROOM_REQUEST_CONN_ROOM_INVALID_PARAM	-3328	無効なパラメータ
ERR_CONNECT_OTHER_ROOM_AS_AUDIENCE	-3330	現在のロールは視聴者であり、ルーム間マイク接続をリクエストまたは切断することはできません。まずキャスターにswitchRole()する必要があります
ERR_SERVER_CENTER_CONN_ROOM_NOT_SUPPORT	-102031	ルーム間マイク接続はサポートされていません

記号	値	意味
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_NUM	-102032	ルーム間マイク接続の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_RETRY_TIMES	-102033	ルーム間マイク接続の再試行回数の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_REQ_TIMEOUT	-102034	ルーム間マイク接続リクエストのタイムアウト
ERR_SERVER_CENTER_CONN_ROOM_REQ	-102035	ルーム間マイク接続リクエストの形式エラー
ERR_SERVER_CENTER_CONN_ROOM_NO_SIG	-102036	ルーム間マイク接続に署名がありません
ERR_SERVER_CENTER_CONN_ROOM_DECRYPT_SIG	-102037	ルーム間マイク接続の署名復号に失敗しました
ERR_SERVER_CENTER_CONN_ROOM_NO_KEY	-102038	ルーム間マイク接続の署名復号キーが見つかりませんでした
ERR_SERVER_CENTER_CONN_ROOM_PARSE_SIG	-102039	ルーム間マイク接続の署名解析エラー
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SIG_TIME	-102040	ルーム間マイク接続の署名タイムスタンプエラー
ERR_SERVER_CENTER_CONN_ROOM_SIG_GROUPID	-102041	ルーム間マイク接続の署名ルーム番号が一致しません

記号	値	意味
ERR_SERVER_CENTER_CONN_ROOM_NOT_CONNED	-102042	ルーム間マイク接続の署名ユーザー名が一致しません
ERR_SERVER_CENTER_CONN_ROOM_USER_NOT_CONNED	-102043	このユーザーはマイク接続を開始していません
ERR_SERVER_CENTER_CONN_ROOM_FAILED	-102044	ルーム間マイク接続に失敗しました
ERR_SERVER_CENTER_CONN_ROOM_CANCEL_FAILED	-102045	ルーム間マイク接続の取り消しに失敗しました
ERR_SERVER_CENTER_CONN_ROOM_CONNED_ROOM_NOT_EXIST	-102046	マイク接続されたルームが存在しません
ERR_SERVER_CENTER_CONN_ROOM_CONNED_REACH_MAX_ROOM	-102047	マイク接続されたルームがマイク接続の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_NOT_EXIST	-102048	マイク接続されたユーザーが存在しません
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_DELETED	-102049	マイク接続されたユーザーが削除されました
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_FULL	-102050	マイク接続されたユーザーがリソースの上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SEQ	-102051	マイク接続リクエストの番号が乱れています

アラートコード表

アラートコードを特に注意する必要はありません。必要に応じて、現在のユーザーに対応するプロンプトを表示するかどうかを選択できます。

記号	値	意味
WARNING_HW_ENCODER_START_FAIL	1103	ハードウェアエンコーディングの起動に問題が発生した場合、自動的にソフトウェアエンコーディングに切り替わります
WARNING_VIDEO_ENCODER_SW_TO_HW	1107	現在のCPU使用率が高すぎてソフトウェアエンコーディング要件を満たせないため、自動的にハードウェアエンコーディングに切り替わります
WARNING_INSUFFICIENT_CAPTURE_FPS	1108	カメラのキャプチャフレームレートが不足しています。美顔アルゴリズムを備えた一部のAndroidスマートフォンには表示されます
WARNING_SW_ENCODER_START_FAIL	1109	ソフトウェアエンコーディングの開始に失敗しました
WARNING_REDUCE_CAPTURE_RESOLUTION	1110	現在のフレームレートとパフォーマンスの最適解を満たすため、カメラのキャプチャ解像度が低下します。
WARNING_CAMERA_DEVICE_EMPTY	1111	使用可能なカメラデバイスが検出されません
WARNING_CAMERA_NOT_AUTHORIZED	1112	ユーザーは、現在のアプリケーションにカメラ使用権限を承認していません
WARNING_MICROPHONE_DEVICE_EMPTY	1201	使用可能なマイクデバイスが検出されません
WARNING_SPEAKER_DEVICE_EMPTY	1202	使用可能なスピーカーデバイスが検出されません
WARNING_MICROPHONE_NOT_AUTHORIZED	1203	ユーザーは、現在のアプリケーションにマイク使用権限を承認していません

記号	値	意味
WARNING_MICROPHONE_DEVICE_ABNORMAL	1204	オーディオキャプチャデバイスが利用できません（使用中であるなど）
WARNING_SPEAKER_DEVICE_ABNORMAL	1205	オーディオ再生デバイスが利用できません（使用中であるなど）
WARNING_VIDEO_FRAME_DECODE_FAIL	2101	現在のビデオフレームのデコードに失敗しました
WARNING_AUDIO_FRAME_DECODE_FAIL	2102	現在のオーディオフレームのデコードに失敗しました
WARNING_VIDEO_PLAY_LAG	2105	現在、ビデオ再生時にラグが発生しています
WARNING_HW_DECODER_START_FAIL	2106	ハードウェアデコードの開始に失敗しました。ソフトウェアデコードを使用してください
WARNING_VIDEO_DECODER_HW_TO_SW	2108	現在のストリームの最初のIフレームのハードウェアデコードが失敗しました。SDKは自動的にソフトウェアデコードに切り替えます
WARNING_SW_DECODER_START_FAIL	2109	ソフトウェアデコーダの起動に失敗しました
WARNING_VIDEO_RENDER_FAIL	2110	ビデオレンダリングに失敗しました
WARNING_START_CAPTURE_IGNORED	4000	キャプチャ中のため、キャプチャの開始は無視されます
WARNING_AUDIO_RECORDING_WRITE_FAIL	7001	オーディオ記録のファイルへの書き込みに失敗しました
WARNING_ROOM_DISCONNECT	5101	ネットワーク接続が切断されました
WARNING_IGNORE_UPSTREAM_FOR_AUDIENCE	6001	現在、視聴者ロールです。アップストリームのオーディオビデオデータを無視します
WARNING_NET_BUSY	1101	ネットワーク状態の不良：アップストリーム帯域幅が小さすぎて、データのアップロード時に問題が生じます

記号	値	意味
WARNING_RTMP_SERVER_RECONNECT	1102	CSSプッシュのときにネットワークが切断しましたが、自動再接続が開始されました（自動再接続は連続3回を超えて失敗すると、それ以上は行われません）
WARNING_LIVE_STREAM_SERVER_RECONNECT	2103	ライブストリーミングのプルのときにネットワークが切断しましたが、自動再接続が開始されました（自動再接続は連続3回を超えて失敗すると、それ以上は行われません）
WARNING_RECV_DATA_LAG	2104	不安定なネットワークパケット：ダウンストリーム帯域幅が不足しているか、キャスター側からのストリーミングが不均一であることが原因と思われます
WARNING_RTMP_DNS_FAIL	3001	ライブストリーミング、DNS解決に失敗しました
WARNING_RTMP_SEVER_CONN_FAIL	3002	ライブストリーミング、サーバーの接続に失敗しました
WARNING_RTMP_SHAKE_FAIL	3003	ライブストリーミング、RTMPサーバーとのハンドシェイクに失敗しました
WARNING_RTMP_SERVER_BREAK_CONNECT	3004	ライブストリーミング、サーバーが自動的に切断しました
WARNING_RTMP_READ_WRITE_FAIL	3005	ライブストリーミング、RTMPの読み取り/書き込みに失敗し、接続が切断されます
WARNING_RTMP_WRITE_FAIL	3006	ライブストリーミング、RTMP書き込みエラー（SDK内部エラーコードは外部にスローされません）
WARNING_RTMP_READ_FAIL	3007	ライブストリーミング、RTMP読み取りエラー（SDK内部エラーコードは外部にスローされません）

記号	値	意味
WARNING_RTMP_NO_DATA	3008	ライブストリーミング、30秒以上データが送信されない場合、自動的に接続が切断されます
WARNING_PLAY_LIVE_STREAM_INFO_CONNECT_FAIL	3009	ライブストリーミング、connectサーバー呼び出しに失敗しました（SDK内部エラーコードは外部にスローされません）
WARNING_NO_STEAM_SOURCE_FAIL	3010	ライブストリーミング、接続に失敗しました。このストリームアドレスにビデオはありません。（SDK内部エラーコードは外部にスローされません）
WARNING_ROOM_RECONNECT	5102	ネットワークが切断され、自動再接続が開始されました
WARNING_ROOM_NET_BUSY	5103	ネットワーク状態の不良：アップストリーム帯域幅が小さすぎて、データのアップロード時に問題が生じます

Web
API概要

最終更新日：：2022-11-11 11:58:54

サポートするプラットフォーム

TRTC Web SDKはWebRTCをベースに実現しており、現在デスクトップおよびモバイル端末の主流ブラウザをサポートしています。サポートの程度の詳細については下の表をご参照ください。
サポート対象の表内にユースケースが見当たらない場合は、TRTC Web SDK機能テスト画面を開き、現在の環境がWebRTCのすべての機能をサポートしているかどうかチェックすることができます（例:WebViewなどの環境）。

OS	ブラウザタイプ	ブラウザの最小バージョン要件	SDKのバージョン要件	受信（再生）	送信（マイク・オン）	画面共有
Windows	デスクトップ版Chromeブラウザ	56+	-	サポートしています	サポートしています	Chrome72以降のバージョンをサポート
	デスクトップ版QQブラウザ（クイックコア）	10.4+	-	サポートしています	サポートしています	サポートしていません
	デスクトップ版Firefoxブラウザ	56+	v4.7.0+	サポートしています	サポートしています	Firefox66以降のバージョンをサポート

OS	ブラウザタイプ	ブラウザの最小バージョン要件	SDKのバージョン要件	受信（再生）	送信（マイク・オン）	画面共有
	デスクトップ版Edgeブラウザ	80+	v4.7.0+	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
	デスクトップ版Sogouブラウザ（高速モード）	11+	v4.7.0+	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
	デスクトップ版Sogouブラウザ（互換モード）	-	-	サポートしていません	サポートしていません	サポートしていません
	デスクトップ版Operaブラウザ	46+	v4.7.0+	サポートしています	サポートしています	Opera60以降のバージョンをサポート
	デスクトップ版360SEブラウザ（超高速モード）	13+	v4.7.0+	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
	デスクトップ版360SEブラウザ（互換モード）	-	-	サポートしていません	サポートしていません	サポートしていません

OS	ブラウザタイプ	ブラウザの最小バージョン要件	SDKのバージョン要件	受信（再生）	送信（マイク・オン）	画面共有
	デスクトップ版 WeChat Embeddedブラウザ	-	-	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
	デスクトップ版 WeCom Embeddedブラウザ	-	-	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
Mac OS	デスクトップ版 Safariブラウザ	11+	-	サポートしています	サポートしています	Safari13以降のバージョンをサポート
	デスクトップ版 Chromeブラウザ	56+	-	サポートしています	サポートしています	Chrome72以降のバージョンをサポート
	デスクトップ版 Firefoxブラウザ	56+	v4.7.0+	サポートしています	サポートしています	Firefox66以降のバージョンをサポート（注意[3]）
	デスクトップ版 Edgeブラウザ	80+	v4.7.0+	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

OS	ブラウザタイプ	ブラウザの最小バージョン要件	SDKのバージョン要件	受信（再生）	送信（マイク・オン）	画面共有
	デスクトップ版Operaブラウザ	46+	v4.7.0+	サポートしています	サポートしています	Opera60以降のバージョンをサポート
	デスクトップ版WeChat Embeddedブラウザ	-	-	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
	デスクトップ版WeCom Embeddedブラウザ	-	-	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
Android	WeChat Embeddedブラウザ（TBSコア）	-	-	サポートしています	サポートしています	サポートしていません
	WeChat Embeddedブラウザ（XWEBコア）	-	-	サポートしています	サポートしています	サポートしていません
	WeCom Embeddedブラウザ	-	-	サポートしています	サポートしています	サポートしていません

OS	ブラウザタイプ	ブラウザの最小バージョン要件	SDKのバージョン要件	受信（再生）	送信（マイク・オン）	画面共有
	モバイル版Chromeブラウザ	-	-	サポートしています	サポートしています	サポートしていません
	モバイル版QQブラウザ	-	-	サポートしていません	サポートしていません	サポートしていません
	モバイル版UCブラウザ	-	-	サポートしていません	サポートしていません	サポートしていません
iOS 12.1.4+	WeChat Embeddedブラウザ	-	-	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
iOS 14.3+	WeChat Embeddedブラウザ	6.5+（WeChatバージョン）	-	サポートしています	サポートしています	サポートしていません
iOS	WeCom Embeddedブラウザ	-	-	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

OS	ブラウザタイプ	ブラウザの最小バージョン要件	SDKのバージョン要件	受信（再生）	送信（マイク・オン）	画面共有
iOS 11.1.2+	モバイル版 Safariブラウザ	11+	-	サポートしています	サポートしています	サポートしていません
iOS 12.1.4+	モバイル版 Chromeブラウザ	-	-	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
iOS 14.3+	モバイル版 Chromeブラウザ	-	-	サポートしています	サポートしています	サポートしていません

注意：

- H.264の著作権上の制限により、Huawei Chrome 88以前のバージョンではH264コーデックを使用できません（プッシュできません）。HuaweiデバイスのChromeブラウザでTRTC Web SDKプッシュをご利用になりたい場合は、チケットを提出し、VP8コーデックの有効化を申請してください。
- Mac OSでのFirefoxの画面共有機能はあまり効果的ではなく、現時点では対処方法もありません。そのため、画面共有にはChromeまたはSafariの使用をお勧めします。
- Web端末でのプッシュ時のダブルサウンドチャンネルコーデックのサポートを希望される場合は、チケットを提出し、WebRTCダブルサウンドチャンネルコーデックを申請してください。
- より良い製品安定性とオンラインサポートを得るために、TRTC Web SDKは適時に最新バージョンに更新することをお勧めします。バージョンアップに関する注意事項については、アップグレードガイドラインをご参照ください。

URLドメイン名プロトコルの制限

ブラウザのセキュリティポリシー上の制限により、WebRTC機能を使用したページへのアクセスプロトコルには厳格な要件があります。 以下の表を参照し、アプリケーションの開発とデプロイを行ってください。

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	サポートあり	サポートあり	サポートあり	**推奨**
本番環境	HTTPプロトコル	サポートあり	サポートなし	サポートなし	
ローカル開発環境	http://localhost	サポートあり	サポートあり	サポートあり	**推奨**
ローカル開発環境	http://127.0.0.1	サポートあり	サポートあり	サポートあり	
ローカル開発環境	http://[ローカルマシンIP]	サポートあり	サポートなし	サポートなし	
ローカル開発環境	file:///	サポートあり	サポートあり	サポートあり	

API使用ガイド

初期化フローおよびAPIの使用法の詳細については、以下のガイドをご参照ください。

機能	Sample Codeガイド
基本的なオーディオビデオ通話	ガイドリンク
インタラクティブライブストリーミングマイク接続の実装	ガイドリンク
カメラおよびマイクの切り替え	ガイドリンク
ローカルビデオのプロパティの設定	ガイドリンク
ローカルオーディオまたはビデオの動的な停止と開始	ガイドリンク
画面共有	ガイドリンク
音量計測	ガイドリンク
ユーザー定義キャプチャとカスタマイズ再生レンダリング	ガイドリンク

機能	Sample Codeガイド
ルーム内アップリンクユーザー数の制限	ガイドリンク
バックグラウンドミュージックと効果音の実装ソリューション	ガイドリンク
通話前の環境およびデバイステスト	ガイドリンク
通話前のネットワーク品質テスト	ガイドリンク
デバイス挿抜動作チェック	ガイドリンク
CDNへのプッシュの実現	-
デュアルストリームの伝送を有効にする	ガイドリンク
美顔を有効にする	ガイドリンク
ウォーターマークを有効にする	ガイドリンク
ルーム間マイク接続の実現	ガイドリンク
クラウドミクスストリーミングの実装	ガイドリンク
クラウドレコーディングの実装	ガイドリンク

説明：

- その他の機能についてはクリックして確認してください。
- よくあるご質問についてはWeb端末関連をご参照ください。

APIの説明

TRTC

注意：
このドキュメントは4.x.xバージョンのTRTC Web SDKに適用されます。

TRTCはTRTC Web SDKのメインエントリで、TRTCメソッドによってTRTCのクライアントオブジェクト（Client）とローカルオーディオビデオストリーミングオブジェクト（Stream）を作成することができます。ま

た、TRTCメソッドはブラウザの互換性や、画面共有をサポートするかをチェックしたり、ログレベルやログのアップロードを設定したりすることもできます。

API	説明
VERSION	TRTC Web SDKバージョン番号。
checkSystemRequirements	ブラウザがTRTC Web SDKと互換性があるかチェックします。現在のブラウザとTRTC Web SDKとの互換性がない場合は、Chromeブラウザの最新バージョンをダウンロードするようユーザーに促すことをお勧めします。
isScreenShareSupported	ブラウザが画面共有をサポートしているかをチェックします。画面共有ストリームを作成する前に、このメソッドを呼び出して、現在のブラウザが画面共有をサポートしているか確認してください。
isSmallStreamSupported	ブラウザがデュアルストリームモードの有効化をサポートしているかをチェックします。デュアルストリームモードを有効化する前に、このメソッドを呼び出して、現在のブラウザがデュアルストリームの有効化をサポートしているか確認してください。
getDevices	メディアの入出力デバイスリストを返します。
getCameras	カメラのデバイスリストを返します。
getMicrophones	マイクのデバイスリストを返します。
getSpeakers	スピーカーのデバイスリストを返します。
createClient	TRTC通話のクライアントオブジェクトを作成します。入退室、オーディオビデオストリーミングの公開、サブスクリプションなどの機能の実装に使用します。
createStream	ローカルストリームStreamオブジェクトを作成します。ローカルストリームStreamオブジェクトはpublish()メソッドによってオーディオビデオストリーミングをリリースします。

TRTC.Logger

ログ出力レベルの設定を含め、ログの設定方法を提供します。ログのアップロードを起動または停止します。

API	説明
setLogLevel	ログ出力レベルを設定します。
enableUploadLog	ログのアップロードを起動します。
disableUploadLog	ログのアップロードを停止します。

Client

オーディオビデオ通話のクライアントオブジェクトClientはcreateClient()によって作成し、1回のオーディオビデオセッションを表します。

API	説明
setProxyServer	プロキシサーバーを設定します。このメソッドは、nginx+coturn方式など、企業が自らプロキシサーバーをデプロイする場合に適用されます。
setTurnServer	TURNサーバーを設定します。このメソッドはsetProxyServer()と合わせて使用し、企業が自らプロキシサーバーおよびTURN中継をデプロイする場合に適用されます。
join	オーディオビデオ通話ルームに参加し、入室によってオーディオビデオ通話セッションが始まります。ルームが存在しない場合、システムが自動的に新しいルームを作成します。
leave	現在のオーディオビデオ通話ルームを退出し、オーディオビデオ通話セッションを終了します。
publish	ローカルのオーディオビデオストリーミングを公開します。このメソッドはjoin()で入室後に呼び出す必要があり、1回のオーディオビデオセッションで1度だけローカルストリーミングを公開することができます。
unpublish	ローカルストリーミングの公開を取り消します。
subscribe	リモートストリーミングを閲覧します。
unsubscribe	リモートストリーミングの閲覧を取り消します。
switchRole	ユーザーロールを切り替えます。'live'でインタラクティブライブストリーミングモードの時のみ有効になります。
sendSEIMessage	SEIメッセージを送信します。
on	クライアントオブジェクトイベントを監視します。
off	クライアントオブジェクトイベントの監視を取り消します。
getRemoteMutedState	現在のルーム内にいるリモートユーザーのオーディオビデオのmute状態リストを取得します。
getTransportStats	現在のネットワーク伝送状況の統計データテーブルを取得します。
getLocalAudioStats	現在公開済みのローカルストリーミングのオーディオ統計データを取得します。このメソッドはpublish()後に呼び出す必要があります。

API	説明
getLocalVideoStats	現在公開済みのローカルストリーミングのビデオ統計データを取得します。このメソッドはpublish()後に呼び出す必要があります。
getRemoteAudioStats	現在のすべてのリモートストリーミングのオーディオ統計データを取得します。
getRemoteVideoStats	現在のすべてのリモートストリーミングのビデオ統計データを取得します。
startPublishCDNStream	現在のクライアントのオーディオビデオストリームのCDNへの公開を開始します。
stopPublishCDNStream	現在のクライアントのオーディオビデオストリームのCDNへの公開を停止します。
startMixTranscode	ミクスストリーミングトランスコードを開始します。このインターフェースは入室し、プッシュしてから呼び出してください。
stopMixTranscode	ミクスストリーミングトランスコードを停止します。このインターフェースはローカルストリームの公開（publish）成功後およびミクスストリーミングトランスコードの開始startMixTranscode成功後に呼び出してください。
enableAudioVolumeEvaluation	音量コールバックを有効化または無効化します。
enableSmallStream	プッシュ側のデュアルストリームモードを有効化します。
disableSmallStream	プッシュ側のデュアルストリームモードを無効化します。
setSmallStreamProfile	スモールストリームのパラメータを設定します。
setRemoteVideoStreamType	視聴側でデュアルストリームの属性を切り替えます。リモートでスモールストリームを有効化していなければ切り替えは成功しません。

LocalStream

LocalStreamローカルオーディオビデオストリーミングは、createStreamで作成します。Streamのサブカテゴリーになります。

API	説明
initialize	ローカルのオーディオビデオストリーミングオブジェクトを初期化します。
setAudioProfile	オーディオProfileを設定します。このメソッドはinitialize()を呼び出す前に呼び出す必要があります。

API	説明
setVideoProfile	ビデオProfileを設定します。このメソッドはinitialize()を呼び出す前に呼び出す必要があります。
setScreenProfile	画面共有Profileを設定します。このメソッドはinitialize()を呼び出す前に呼び出す必要があります。
setVideoContentHint	ビデオコンテンツのヒントを設定します。主に、さまざまなシーンにおけるビデオコーデック品質の向上に用いられます。このメソッドはinitialize()の呼び出し完了後に呼び出す必要があります。
switchDevice	メディアの入力デバイスを切り替えます。
addTrack	オーディオまたはビデオのトラックを追加します。
removeTrack	ビデオトラックを削除します。
replaceTrack	オーディオまたはビデオのトラックを変更します。
play	このオーディオビデオストリーミングを再生します。
stop	オーディオビデオストリーミングの再生を停止します。
resume	オーディオビデオの再生を再開します。
close	オーディオビデオストリーミングを終了します。
muteAudio	オーディオトラックを無効にします。
muteVideo	ビデオトラックを無効にします。
unmuteAudio	オーディオトラックを有効にします。
unmuteVideo	ビデオトラックを有効にします。
getId	Streamの固有識別IDを取得します。
getUserId	このストリームが属するユーザーIDを取得します。
setAudioOutput	音声出力デバイスを設定します。
getAudioLevel	現在の音量を取得します。ローカルストリーミングまたはリモートストリーミングにオーディオデータがある場合のみ有効となります。
setAudioCaptureVolume	マイクキャプチャ音量を設定します。
hasAudio	オーディオトラックが含まれているかどうか。

API	説明
hasVideo	ビデオトラックが含まれているかどうか。
getAudioTrack	オーディオトラックを取得します。
getVideoTrack	ビデオトラックを取得します。
getVideoFrame	現在のビデオフレームを取得します。
on	Streamイベントを監視します。
off	Streamイベントの監視を取り消します。

RemoteStream

リモートオーディオビデオストリーミングは、Client.on('stream-added')イベントの監視によって取得します。これはStreamのサブカテゴリーになります。

API	説明
getType	リモートストリーミングのタイプを取得します。主に1つのリモートストリーミングがメインオーディオビデオストリーミングかサブビデオストリームかを判断することに用いられます。サブビデオストリームは通常、画面共有ストリームです。
play	このオーディオビデオストリーミングを再生します。
stop	オーディオビデオストリーミングの再生を停止します。
resume	オーディオビデオの再生を再開します。
close	オーディオビデオストリーミングを終了します。
muteAudio	オーディオトラックを無効にします。
muteVideo	ビデオトラックを無効にします。
unmuteAudio	オーディオトラックを有効にします。
unmuteVideo	ビデオトラックを有効にします。
getId	Streamの固有識別IDを取得します。
getUserId	このストリームが属するユーザーIDを取得します。
setAudioOutput	音声出力デバイスを設定します。
setAudioVolume	再生音量を設定します。

API	説明
getAudioLevel	現在の音量を取得します。ローカルストリーミングまたはリモートストリーミングにオーディオデータがある場合のみ有効となります。
hasAudio	オーディオトラックが含まれているかどうか。
hasVideo	ビデオトラックが含まれているかどうか。
getAudioTrack	オーディオトラックを取得します。
getVideoTrack	ビデオトラックを取得します。
getVideoFrame	現在のビデオフレームを取得します。
on	Streamイベントを監視します。
off	Streamイベントの監視を取り消します。

RtcError

RtcErrorエラーオブジェクト。

API	説明
getCode	エラーコードを取得します。

ClientEvent

Clientがトリガーするイベントのリスト、すなわち `client.on('eventName')` イベント監視中のイベント名 `eventName` です。

API	説明
stream-added	リモートストリーム追加イベントです。リモートユーザーがストリームを公開した場合にこの通知を受信します。
stream-removed	リモートストリーム削除イベントです。リモートユーザーがストリームの公開を取り消した場合にこの通知を受信します。
stream-updated	リモートストリーム更新イベントです。リモートユーザーがオーディオビデオトラックを追加、削除または変更した場合にこの通知を受信します。
stream-subscribed	リモートストリームサブスクリプション成功イベントです。subscribe()を呼び出して成功した場合にこのイベントがトリガーされます。

API	説明
connection-state-changed	ローカルclientとTencent Cloudの接続ステータス変更イベントです。
peer-join	リモートユーザー入室イベントです。
peer-leave	リモートユーザー退室イベントです。
mute-audio	リモートストリームのオーディオ無効化イベントです。リモートユーザーがオーディオを無効にした場合にこのイベントがトリガーされます。
mute-video	リモートストリームのビデオ無効化イベントです。リモートユーザーがビデオを無効にした場合にこのイベントがトリガーされます。
unmute-audio	リモートストリームのオーディオ有効化イベントです。リモートユーザーがオーディオを有効にした場合にこのイベントがトリガーされます。
unmute-video	リモートストリームのビデオ有効化イベントです。リモートユーザーがビデオを有効にした場合にこのイベントがトリガーされます。
client-banned	ユーザーの強制退室イベントです。強制退室の原因には次のものがあります。 • 同名のユーザーが同一のルームに入室した。**注意**：同名のユーザーが同時に同一のルームに入室することは、双方のオーディオビデオ通話に異常が生じることがあるため禁止されています。業務側はこのような状況を避けなければなりません。 • アカウント管理者がサーバーAPIを使用して強制退室させた。
network-quality	ネットワーク品質統計データイベントです。入室後に統計を開始し、2秒に1回トリガーします。アップおよびダウンストリームネットワーク品質データが含まれます。
audio-volume	音量イベントです。 enableAudioVolumeEvaluationインターフェースを呼び出して音量コールバックを有効化すると、SDKは決まった時間にこのイベントをスローし、各userIdの音量を通知します。
sei-message	seiメッセージを受信します。
error	エラーイベントです。復旧できないエラーが発生すると、このイベントがスローされます。エラーコードをご参照ください。

StreamEvent

Streamによってトリガーされるイベントのリストです。

API	説明

API	説明
player-state-changed	Audio/Video Playerのステータス変更イベントです。
screen-sharing-stopped	ローカル画面共有停止イベントです。ローカル画面共有ストリームに対してのみ有効です。
connection-state-changed	Stream接続ステータス変更イベントです。 `stream-added` イベントコールバックでこのイベントを監視し、 `stream-removed` イベントコールバックでこのイベントの監視を取り消してください。
error	エラーイベントです。復旧できないエラーが発生すると、このイベントがスローされます。エラーコードをご参照ください。

お問い合わせ

ご不明な点がございましたら、colleenyu@tencent.comにご連絡ください。

エラーコード

最終更新日：：2022-06-28 11:04:41

注意：
このドキュメントは4.x.xバージョンのTRTC Web SDKに適用されます。

エラーコードの定義

Key	getCode	エラーコード	説明
INVALID_PARAMETER	4096	0x1000	無効なパラメータ 推奨する対処方法：渡したパラメータがSDKの要件を満たしているかご確認ください。例えば、パラメータのタイプが正しいかどうかなどです。
INVALID_OPERATION	4097	0x1001	不正な操作 推奨する対処方法：対応するインターフェースドキュメントに従って、API呼び出しロジックがSDKの要件を満たしているかご確認ください。 例えば、入室していない状態でpublishインターフェースを呼び出すことは禁止されています。

Key	getCode	エラーコード	説明
NOT_SUPPORTED	4098	0x1002	SDKインターフェースを呼び出す際にスローされ、現在のブラウザが対応するインターフェースの呼び出しをサポートしていないことを表します • 説明：SDKインターフェースを呼び出す際にスローされ、現在のブラウザが対応するインターフェースの呼び出しをサポートしていないことを表します • 推奨する対処方法：SDKをサポートしているブラウザを使用するようユーザーに指示します。詳細についてはブラウザサポート状況の確認をご参照ください
DEVICE_NOT_FOUND	4099	0x1003	現在のデバイスにマイクまたはカメラがないにもかかわらず、マイク、カメラキャプチャを試みています。 推奨する対処方法：デバイスのカメラおよびマイクが正常かどうかをチェックするようユーザーに指示し、業務側は入室前のデバイス検査ロジックを追加する必要があります。通話前の環境およびデバイス検査をご参照ください。
INITIALIZE_FAILED	4100	0x1004	LocalStream.initialize()のキャプチャに失敗しました。推奨する対処方法の詳細をご参照ください。

Key	getCode	エラーコード	説明
SIGNAL_CAHNNEL_SETUP_FAILED	16385	0x4001	シグナリングチャネルの確立に失敗しました。通常はTencent Cloudアカウントに関連しています。具体的にはアカウントに関するエラーメッセージをご参照ください。
SIGNAL_CHANNEL_ERROR	16386	0x4002	シグナリングチャネルエラー。
ICE_TRANSPORT_ERROR	16387	0x4003	ICE Transportの接続エラー、すなわちオーディオビデオデータ伝送チャネルのエラーです。主な原因はユーザー側のUDPポートの異常（ユーザーのコンピュータのファイアウォールまたはルーターのファイアウォールポートの制限の可能性があります）によるものです。具体的なポートについてはポートホワイトリストをご参照ください。
JOIN_ROOM_FAILED	16388	0x4004	入室に失敗しました。詳細については入室失敗に関するエラーメッセージをご参照ください
CREATE_OFFER_FAILED	16389	0x4005	sdp offerの作成に失敗しました。

Key	getCode	エラーコード	説明
SIGNAL_CHANNEL_RECONNECTION_FAILED	16390	0x4006	WebSocketシグナリングチャネルの再接続に失敗しました • 説明：WebSocketが切断された際に、SDKが複数回再接続を試行し、すべて失敗した場合にこのエラーがスローされます • 推奨する対処方法：ネットワークを確認してから再度入室するよう、ユーザーに注意喚起します
UPLINK_RECONNECTION_FAILED	16391	0x4007	アップストリームのPeerConnectionの再接続に失敗しました • 説明：アップストリームのPeerConnectionが切断された際に、SDKが複数回再接続を試行し、すべて失敗した場合にこのエラーがスローされます • 推奨する対処方法：ネットワークを確認してから再プッシュまたは再入室するよう、ユーザーに注意喚起します

Key	getCode	エラーコード	説明
DOWNLINK_RECONNECTION_FAILED	16392	0x4008	ダウンストリームのPeerConnectionの再接続に失敗しました • 説明：ダウンストリームのPeerConnectionに異常な切断が起こった際に、SDKが複数回再接続を試行し、すべて失敗した場合にこのエラーがスローされます • 推奨する対処方法：ネットワークを確認してから再度入室するよう、ユーザーに注意喚起します
REMOTE_STREAM_NOT_EXIST	16400	0x4010	リモートストリームが存在しません • 説明：AがBのプッシュするストリームをサブスクライブしている場合、Bがプッシュをキャンセルしたことで、AによるBのサブスクリプションが失敗しました • 推奨する対処方法：正常なインタラクションフローであり、アクセス側が対処する必要はありません

Key	getCode	エラーコード	説明
CLIENT_BANNED	16448	0x4040	ユーザーがルームから強制退室させられました。強制退室の原因には次のものがあります。 • 同名のユーザーが同一のルームに入室した。**注意**：同名のユーザーが同時に同一のルームに入室することは、双方のオーディオビデオ通話に異常が生じることがあるため禁止されています。業務側はこのような状況を避けなければなりません • アカウント管理者がサーバーAPIを使用して強制退室させます
SERVER_TIMEOUT	16449	0x4041	メディア伝送サービスがタイムアウトしました
SUBSCRIPTION_TIMEOUT	16450	0x4042	リモートストリームのサブスクリプションがタイムアウトしました

Key	getCode	エラーコード	説明
PLAY_NOT_ALLOWED	16451	0x4043	自動再生が禁止されるエラー • 説明：`play()` を呼び出して オーディオビデオ再生を行う際、自動再生ポリシーの制限により自動再生できません • 推奨する対処方法：この場合はジェスチャー操作で `resume()` を呼び出し、再生を継続するようユーザーに指示する必要があります。制限された自動再生の処理に関するアドバイスをご参照ください。)

Key	getCode	エラーコード	説明
DEVICE_AUTO_RECOVER_FAILED	16452	0x4044	カメラ、マイクキャプチャの自動再開に失敗しました。このエラーはLocalStreamのerrorイベントでスローされます • ユーザーがプッシュ中のメディアデバイスが変更された場合（例：カメラ、マイクを抜き差しした、接続箇所が緩んでいるなど）、SDKはキャプチャの再開を試行します。このエラーが発生した場合は、再開に失敗しています • 推奨する対処方法：メディアデバイスのキャプチャ自動再開に失敗したことをユーザーに知らせ、カメラ、マイクの接続箇所に緩みがないか確認するよう注意喚起します • ページにリトライボタンを設置し、ユーザーがリトライをクリックした場合に再度カメラ、マイクキャプチャを行うようにすることができます
START_PUBLISH_CDN_FAILED	16453	0x4045	CDNへのプッシュ開始に失敗しました
STOP_PUBLISH_CDN_FAILED	16454	0x4046	CDNへのプッシュ停止に失敗しました
START_MIX_TRANSCODE_FAILED	16455	0x4047	ミクスストリーミングトランスコードの開始に失敗しました

Key	getCode	エラーコード	説明
STOP_MIX_TRANSCODE_FAILED	16456	0x4048	ミクスストリーミングトランスコードの停止に失敗しました
NOT_SUPPORTED_H264	16457	0x4049	現在のデバイスはH.264をサポートしていません
SWITCH_ROLE_FAILED	16458	0x404a	ロールの切り替えに失敗しました
UNKOWN	65535	0xFFFF	不明なエラー

アカウントに関するエラーメッセージ

エラーコード	エラータイプ	説明
-8	アカウントシステム	sdkAppIdが正しくありません。sdkAppIdが正しく入力されているか確認してください
70001	アカウントシステム	UserSigが期限切れです。再度生成をお試しください。生成してすぐに期限切れとなった場合は、入力した有効期間が短すぎないか、もしくは誤って0と入力していないか確認してください
70002	アカウントシステム	userSigの長さが0になっています。署名の計算が正しいか確認してください。sign_srcにアクセスして署名計算用の簡易ソースコードを取得し、パラメータを照合して署名計算の正確性を確認してください
70003	アカウントシステム	userSigのチェックに失敗しました。userSigの内容が分割されていないか、例えばバッファの長さが足りないために内容が分割されていることがないか確認してください
70004	アカウントシステム	userSigのチェックに失敗しました。userSigの内容が分割されていないか、例えばバッファの長さが足りないために内容が分割されていることがないか確認してください
70005	アカウントシステム	userSigのチェックに失敗しました。生成されたuserSigが正しいかどうかをツールで検証します。
70006	アカウントシステム	userSigのチェックに失敗しました。生成されたuserSigが正しいかどうかをツールで検証します

エラーコード	エラータイプ	説明
70007	アカウントシステム	userSigのチェックに失敗しました。生成されたuserSigが正しいかどうかをツールで検証します
70008	アカウントシステム	userSigのチェックに失敗しました。生成されたuserSigが正しいかどうかをツールで検証します
70009	アカウントシステム	業務の公開鍵によるuserSigの検証に失敗しました。生成されたuserSigが使用している秘密鍵とsdkAppIdが対応しているかどうかを確認してください
70010	アカウントシステム	userSigのチェックに失敗しました。生成されたuserSigが正しいかどうかをツールで検証します
70013	アカウントシステム	userSig内のuserIdとリクエスト時のuserIdが一致しません。ログイン時に入力したuserIdがuserSig内のものと一致しているかどうかを確認してください
70014	アカウントシステム	userSig内のsdkAppIdとリクエスト時のsdkAppIdが一致しません。ログイン時に入力したsdkAppIdがuserSig 内のものと一致しているかどうかを確認してください
70015	アカウントシステム	このsdkAppIdとアカウントタイプに対応する検証方法が見つかりません。アカウント統合操作をすでに行っていないか確認してください
70016	アカウントシステム	取得した公開鍵の長さが0です。公開鍵をすでにアップロードしていないか確認してください。再アップロードした公開鍵の場合は10分後にリトライする必要があります
70017	アカウントシステム	内部の第三者証明書の検証がタイムアウトしました。もう一度お試しください。複数回試しても成功しない場合は、お問い合わせください
70018	アカウントシステム	第三者証明書の内部検証に失敗しました
70019	アカウントシステム	HTTPS方式で検証された証明書フィールドが空です。userSigを正しく入力してください
70020	アカウントシステム	sdkAppIdが見つかりません。Tencent Cloudで設定済みかどうか確認してください
70052	アカウントシステム	userSigが無効になっています。再生成してからもう一度お試しください
70101	アカウントシステム	リクエストパケット情報が空です

エラーコード	エラータイプ	説明
70102	アカウントシステム	リクエストパケットのアカウントタイプが正しくありません
70103	アカウントシステム	電話番号の形式が正しくありません
70104	アカウントシステム	メールアドレスの形式が正しくありません
70105	アカウントシステム	TLSアカウントの形式が正しくありません
70106	アカウントシステム	アカウントの形式やタイプが正しくありません
70107	アカウントシステム	userIdが登録されていません
70113	アカウントシステム	バッチ数量が不正です
70114	アカウントシステム	安全上の理由により制限されています
70115	アカウントシステム	uinは対応するsdkAppIdの開発者uinではありません
70140	アカウントシステム	sdkAppIdとacctypeが一致しません
70145	アカウントシステム	アカウントタイプが正しくありません
70169	アカウントシステム	内部エラーが発生しました。もう一度お試しください。複数回試しても成功しない場合は、お問い合わせください
70201	アカウントシステム	内部エラーが発生しました。もう一度お試しください。複数回試しても成功しない場合は、お問い合わせください
70202	アカウントシステム	内部エラーが発生しました。もう一度お試しください。複数回試しても成功しない場合は、お問い合わせください
70203	アカウントシステム	内部エラーが発生しました。もう一度お試しください。複数回試しても成功しない場合は、お問い合わせください

エラーコード	エラータイプ	説明
70204	アカウントシステム	sdkAppIdに対応するacctypeがありません
70205	アカウントシステム	acctypeの検索に失敗しました。もう一度お試しください
70206	アカウントシステム	リクエスト内のバッチ数量が不正です
70207	アカウントシステム	内部エラーが発生しました。もう一度お試しください
70208	アカウントシステム	内部エラーが発生しました。もう一度お試しください
70209	アカウントシステム	開発者uinマークの取得に失敗しました
70210	アカウントシステム	リクエスト内のuinは開発者のuinではありません
70211	アカウントシステム	リクエスト内のuinが正しくありません
70212	アカウントシステム	内部エラーが発生しました。もう一度お試しください。複数回試しても成功しない場合は、お問い合わせください
70213	アカウントシステム	内部データへのアクセスに失敗しました。もう一度お試しください。複数回試しても成功しない場合は、お問い合わせください
70214	アカウントシステム	内部証明書の検証に失敗しました
70221	アカウントシステム	ログイン状態が無効です。UserSigを使用して再認証してください
70222	アカウントシステム	内部エラーが発生しました。もう一度お試しください
70225	アカウントシステム	内部エラーが発生しました。もう一度お試しください。複数回試しても成功しない場合は、お問い合わせください
70231	アカウントシステム	内部エラーが発生しました。もう一度お試しください。複数回試しても成功しない場合は、お問い合わせください

エラーコード	エラータイプ	説明
70236	アカウントシステム	user signatureの検証に失敗しました
70308	アカウントシステム	内部エラーが発生しました。もう一度お試しください。複数回試しても成功しない場合は、お問い合わせください
70346	アカウントシステム	証明書のチェックに失敗しました。
70347	アカウントシステム	証明書が期限切れのためチェックに失敗しました
70348	アカウントシステム	内部エラーが発生しました。もう一度お試しください。複数回試しても成功しない場合は、お問い合わせください
70362	アカウントシステム	内部タイムアウトです。もう一度お試しください。複数回試しても成功しない場合は、お問い合わせください
70401	アカウントシステム	内部エラーが発生しました。もう一度お試しください
70402	アカウントシステム	不正なパラメータです。入力必須フィールドに入力されているか、またはフィールドの入力がプロトコル要件を満たしているかを確認してください
70403	アカウントシステム	操作者がApp管理者ではないため、操作権限がありません
70050	アカウントシステム	失敗とリトライ回数が多すぎるため制限されました。証明書が正しいかどうかをご確認の上、1分後にもう一度お試しください
70051	アカウントシステム	アカウントがブラックリストに登録されました。お問い合わせください

入室失敗に関するエラーメッセージ

エラーコード	エラーメッセージ
10006	お客様のサービス料金のお支払いが遅延しています。TRTCコンソールにログインして、作成したアプリケーションをクリックし、**アプリケーション情報**をクリックすれば、アプリケーション情報パネルでサービス状態を確認できます
-10011	サーバー側の不明なエラーです。もう一度お試しください

エラーコード	エラーメッセージ
-10012	roomIdが渡されていないか、またはroomIdがルールを満たしていません。stringタイプのroomIdを使用する必要がある場合は、`TRTC.createClient`を呼び出す際にuseStringRoomIdをtrueに設定してください
-10013	userSigの認証に失敗しました
-10015	サーバー側でのサーバーノード取得に失敗しました。もう一度お試しください
-10016	サーバー側でのルーム作成に失敗しました。roomIdがトラフィックコントロールの許容範囲内にあるか確認してください
-10018	高度な権限制御を有効にした後、client.joinにprivateMapKeyパラメータが含まれていないか、またはprivateMapKeyが''となっています。高度な権限設定を有効にするをご参照ください
-10019	高度な権限制御を有効にした後、client.joinに含まれるprivateMapKeyパラメータがルールを満たしていません。高度な権限設定を有効にするをご参照ください
-10020	サーバー側がタイムアウトしました。もう一度お試しください
-100027	TRTCの状態が正常かどうか確認し、Tencent CloudのTRTCコンソールでTRTCサービスを有効にしてから入室してください

一般的なエラーと対処方法

このタイプのエラーでは、アプリケーションによる介入が必要です。例えば、カメラ取得権限が拒否された場合、カメラの権限承認を行ってからでなければオーディオビデオ通話が行えないことについて、アプリケーションからユーザーに注意喚起する必要があります。

エラー情報	エラー原因	対処方法
publish timeout	publishタイムアウト	更新して再接続を試行し、再度publish()の操作を行ってください
join room timeout	入室タイムアウト	ページを更新して再入室することをお勧めします
DTLS Transport connection timeout (10s)	DTLS Transport接続タイムアウト	更新して再接続を試行してください

エラー情報	エラー原因	対処方法
failed to connect to remote server via websocket	websocketの接続に失敗しました	更新して再接続を試行してください
ICE/DTLS Transport connection failed	メディア伝送チャネル確立の際に失敗しました	ファイアウォールの設定を確認してください
previous publishing is ongoing, please avoid re-publishing	すでにpublishing状態です	publish後は再度publish()を行わないでください
AbortError	デバイス/システムの何らかの理由により、デバイスが使用できなくなっています	通話前にデバイス検査を行うことをお勧めします
NotReadableError	OS上のいずれかのハードウェア、ブラウザまたはWebページの階層に発生したエラーのため、デバイスにアクセスできません	ブラウザのエラーメッセージに従って処理すると、ユーザーに対し、他のアプリケーションがカメラ/マイクへのアクセスをリクエストしていないことを確認してからリトライするよう表示されます
NotFoundError	リクエストパラメータを満たすメディアタイプ（オーディオ、ビデオ、画面共有など）が見つかりません	通話前にデバイス検査を行うことをお勧めします
NotAllowedError	ユーザーが、現在のブラウザ・インスタンスのオーディオ、ビデオ、画面共有へのアクセスのリクエストを拒否しました	ユーザーはカメラ/マイクへのアクセス権限承認を行ってからでなければオーディオビデオ通話を行うことができません
SignalChannel reconnect failed	websocketが切断されました	更新して再接続を試行してください
duplicate publishing, please unpublish and then re-publish	publishが重複しています	先にunpublish()操作を行ってからpublish()を操作してください
OverconstrainedError	ブラウザがcameraId / microphoneIdを取得できませんでした	cameraId / microphoneIdの値が有効な空でない文字列であることを確認してください
RtcError: no valid ice candidate found	TRTC Web SDKがSTUNトンネリングに失敗しました	ファイアウォールの設定を確認してください

Electron
API概要

最終更新日：：2022-08-08 15:25:08

TRTCCloud @ TXLiteAVSDK

Tencent Cloudビデオ通話機能の主なインターフェース。

- **主なドキュメントアドレス**：TRTC Electron SDK
- **サンプルコードアドレス**：TRTC Electron Demo

TRTCオブジェクトの作成

```
const TRTCCloud = require('trtc-electron-sdk').default;
// import TRTCCloud from 'trtc-electron-sdk';
this.rtcCloud = new TRTCCloud();
```

v7.9.348から、TRTC Electron SDKはtrtc.d.tsファイルを追加しており、 typescriptを使用する開発者の操作性が向上しました。

```
import TRTCCloud from 'trtc-electron-sdk';
const rtcCloud: TRTCCloud = new TRTCCloud();
// SDKバージョン番号の取得
rtcCloud.getSDKVersion();
```

// コールバックの設定

```
subscribeEvents = (rtcCloud) => {
rtcCloud.on('onError', (errcode, errmsg) => {
console.info('trtc_demo: onError :' + errcode + " msg" + errmsg);
});
rtcCloud.on('onEnterRoom', (elapsed) => {
console.info('trtc_demo: onEnterRoom elapsed:' + elapsed);
});
rtcCloud.on('onExitRoom', (reason) => {
console.info('onExitRoom: userenter reason:' + reason);
});
};

subscribeEvents(this.rtcCloud);
```

TRTCCloudシングルトンの作成と破棄

API	説明
getTRTCShareInstance	dllを動的にロードするために使用する場合は、TRTCCloudオブジェクトシングルトンを作成します。
destroyTRTCShareInstance	TRTCCloudシングルトンオブジェクトをリリースし、リソースをクリーンアップします。

ルーム関連インターフェース関数

API	説明
enterRoom	ルームに入室します。ルームが存在しない場合は、システムが新しいルームを自動的に作成します。
exitRoom	ルームを退室します。
switchRoom	ルームを切り替えます。
switchRole	ロールを切り替えます。ライブストリーミングシナリオ（TRTCAppSceneLIVEおよびTRTCAppSceneVoiceChatRoom）のみに適しています。
connectOtherRoom	ルーム間のマイク接続をリクエストします（キャスタールーム間PK）。
disconnectOtherRoom	ルーム間のマイク接続を終了します（キャスタールーム間PK）。
setDefaultStreamRecvMode	オーディオビデオの受信モードを設定します（有効にするには、入室する前に設定する必要があります）。

CDN関連インターフェース関数

API	説明
startPublishing	Tencent Cloudのライブ CDNへのプッシュを開始します。
stopPublishing	Tencent Cloudのライブ CDNへのプッシュを停止します。
startPublishCDNStream	Tencent Cloud以外のライブ CDNへのリレーを開始します。
stopPublishCDNStream	Tencent Cloud以外のライブ CDNへのプッシュを停止します。
setMixTranscodingConfig	クラウドのミクスストリーミングトランスコードパラメータを設定します。

ビデオ関連インターフェース関数

API	説明
startLocalPreview	ローカルカメラのキャプチャとプレビューを起動します。
stopLocalPreview	ローカルカメラのキャプチャとプレビューを停止します。
muteLocalVideo	自身のビデオ画面をブロックするかどうか。
startRemoteView	リモートビデオ画面の表示を開始します。
stopRemoteView	リモートビデオ画面の表示を停止すると同時に、このリモートユーザーのビデオデータトラフィックのプルを停止します。
stopAllRemoteView	すべてのリモートビデオ画面の表示を停止すると同時に、リモートユーザーのビデオデータトラフィックのプルを停止します。
muteRemoteVideoStream	指定のリモートビデオストリームの受信を一時停止します。
muteAllRemoteVideoStreams	すべてのリモートビデオストリームの受信を停止します。
setVideoEncoderParam	ビデオエンコーダの関連パラメータを設定します。
setNetworkQosParam	ネットワークトラフィックコントロールの関連パラメータを設定します。
setLocalRenderParams	ローカル画像（メインストリーム）のレンダリングパラメータを設定します。
setLocalViewFillMode	破棄されたインターフェース：ローカル画像のレンダリングモードを設定します。
setRemoteRenderParams	リモート画像のレンダリングパラメータを設定します。
setRemoteViewFillMode	破棄されたインターフェース：リモート画像のレンダリングモードを設定します。
setLocalViewRotation	破棄されたインターフェース：ローカル画像の時計回りの回転角度を設定します。
setRemoteViewRotation	破棄されたインターフェース：リモート画像の時計回りの回転角度を設定します。
setVideoEncoderRotation	ビデオコーデックが出力する画面（リモートユーザーが視聴する画面およびサーバーが録画する画面）方向を設定します。

API	説明
setLocalViewMirror	破棄されたインターフェース：ローカルカメラプレビュー画面のイメージモードを設定します。
setVideoEncoderMirror	エンコーダが出力する画面のイメージモードを設定します。
enableSmallVideoStream	大小画面のデュアルチャンネルコーディングモードを有効にします。
setRemoteVideoStreamType	指定userIdの大画面または小画面での視聴を選択します。
setPriorRemoteVideoStreamType	破棄されたインターフェース： 視聴者が優先的に選択するビデオ品質を設定します。
snapshotVideo	ビデオ画面のスクリーンキャプチャです。

オーディオ関連インターフェース関数

API	説明
startLocalAudio	ローカルオーディオのキャプチャとアップストリームを開始します。
stopLocalAudio	ローカルオーディオのキャプチャとアップストリームを終了します。
muteLocalAudio	ローカルのオーディオをミュートにします。
muteRemoteAudio	特定ユーザーの音声をミュートにすると同時に、このリモートユーザーのオーディオデータトラフィックのプルを停止します。
muteAllRemoteAudio	すべてのユーザーの音声をミュートにすると同時に、リモートユーザーのオーディオデータトラフィックのプルを停止します。
setAudioCaptureVolume	SDKキャプチャ音量を設定します 。
getAudioCaptureVolume	SDKキャプチャ音量を取得します。
setAudioPlayoutVolume	SDK再生音量を設定します。
getAudioPlayoutVolume	SDK再生音量を取得します。
enableAudioVolumeEvaluation	音量レベルプロンプトを起動または終了します。
startAudioRecording	録音を開始します。
stopAudioRecording	録音を停止します。
setAudioQuality	破棄されたインターフェース：オーディオ品質を設定します。

API	説明
setRemoteAudioVolume	リモートユーザーの再生音量を設定します。

カメラ関連インターフェース関数

API	説明
getCameraDevicesList	カメラデバイスリストを取得します。
setCurrentCameraDevice	使用したいカメラを設定します。
getCurrentCameraDevice	現在使用するカメラ を取得します。

オーディオデバイス関連インターフェース関数

API	説明
getMicDevicesList	マイクデバイスリストを取得します。
getCurrentMicDevice	現在選択しているマイクを取得します。
setCurrentMicDevice	使用したいマイクを設定します。
getCurrentMicDeviceVolume	システムの現在のマイクデバイス音量を取得します。
setCurrentMicDeviceVolume	システムの現在のマイクデバイスの音量を設定します。
setCurrentMicDeviceMute	システムの現在のマイクデバイスのミュートステータスを設定します。
getCurrentMicDeviceMute	システムの現在のマイクデバイスがミュートであるかどうかを取得します。
getSpeakerDevicesList	スピーカーデバイスリストを取得します。
getCurrentSpeakerDevice	現在のスピーカーデバイスを取得します。
setCurrentSpeakerDevice	使用したいスピーカーを設定します。
getCurrentSpeakerVolume	システムの現在のスピーカーデバイス音量を取得します。
setCurrentSpeakerVolume	システムの現在のスピーカーデバイス音量を設定します。
setCurrentSpeakerDeviceMute	システムの現在のスピーカーデバイスのミュートステータスを設定します。
getCurrentSpeakerDeviceMute	システムの現在のスピーカーデバイスがミュートかどうかを取得します。

美顔関連インターフェース関数

API	説明
setBeautyStyle	美顔、美白および肌の色調補正エフェクトレベルを設定します。
setWaterMark	ウォーターマークを設定します。

サブストリーム関連インターフェース関数

API	説明
startRemoteSubStreamView	破棄されたインターフェース：リモートユーザーのサブストリーム（画面共有）画面のレンダリングを開始します。
stopRemoteSubStreamView	破棄されたインターフェース：リモートユーザーのサブストリーム（画面共有）画面のレンダリングを停止します。
setRemoteSubStreamViewFillMode	破棄されたインターフェース：サブストリーム（画面共有）画面のレンダリングモードを設定します。
setRemoteSubStreamViewRotation	破棄されたインターフェース：サブストリーム（画面共有）画面の時計回りの回転角度を設定します。
getScreenCaptureSources	共有可能なウィンドウリストを列挙します。
selectScreenCaptureTarget	画面共有パラメータを設定します。画面共有中にもこのメソッドを呼び出すことができます。
startScreenCapture	画面共有を起動します。
pauseScreenCapture	画面共有を一時停止します。
resumeScreenCapture	画面共有を再開します。
stopScreenCapture	画面共有を停止します。
setSubStreamEncoderParam	サブストリーム（画面共有）のエンコーダパラメータを設定します。
setSubStreamMixVolume	サブストリーム（画面共有）の音声ミキシングの音量レベルを設定します。
addExcludedShareWindow	指定ウィンドウを画面共有のexcludeリストに追加します。excludeリストに追加したウィンドウは共有できなくなります。
removeExcludedShareWindow	指定ウィンドウを画面共有のexcludeリストから削除します。
removeAllExcludedShareWindow	すべてのウィンドウを画面共有のexcludeリストから削除します。

カスタムメッセージの送信

API	説明
sendCustomCmdMsg	カスタムメッセージをルーム内のすべてのユーザーに送信します。
sendSEIMsg	データ量の小さなカスタムデータをビデオフレームに埋め込みます。

BGMミキシング関連インターフェース関数

API	説明
playBGM	破棄されたインターフェース：BGMの再生を起動します。
stopBGM	破棄されたインターフェース：BGMの再生を停止します。
pauseBGM	破棄されたインターフェース：BGMの再生を一時停止します。
resumeBGM	破棄されたインターフェース：BGMの再生を継続します。
getBGMDuration	破棄されたインターフェース：BGMファイルの総時間を取得します。単位はミリ秒です。
setBGMPosition	破棄されたインターフェース：BGM再生の進捗を設定します。
setBGMVolume	破棄されたインターフェース：BGM再生音量レベルを設定します。
setBGMPlayoutVolume	破棄されたインターフェース：BGMローカル再生音量レベルを設定します。
setBGMPublishVolume	破棄されたインターフェース：BGMリモート再生音量レベルを設定します。
startSystemAudioLoopback	システム音声キャプチャを起動します。
stopSystemAudioLoopback	システム音声キャプチャを終了します。
setSystemAudioLoopbackVolume	システム音声キャプチャの音量を設定します。
startPlayMusic	BGMの再生を起動します。
stopPlayMusic	BGMの再生を停止します。
pausePlayMusic	BGMの再生を一時停止します。
resumePlayMusic	BGMの再生を再開します。
getMusicDurationInMS	BGMファイルの総時間を取得します。単位はミリ秒です。

API	説明
seekMusicToPosInTime	BGM再生の進捗を設定します。
setAllMusicVolume	BGMの音量レベルを設定します。BGMの音量レベルを制御するために、BGMの再生や音声ミキシング時に使用します。
setMusicPlayoutVolume	BGMのローカル再生音量レベルを設定します。
setMusicPublishVolume	BGMのリモート再生音量レベルを設定します。

オーディオエフェクト関連インターフェース関数

API	説明
playAudioEffect	破棄されたインターフェース：オーディオエフェクトを再生します。
setAudioEffectVolume	破棄されたインターフェース：オーディオエフェクト音量を設定します。
stopAudioEffect	破棄されたインターフェース：オーディオエフェクトを停止します。
stopAllAudioEffects	破棄されたインターフェース：すべてのオーディオエフェクトを停止します。
setAllAudioEffectsVolume	破棄されたインターフェース：すべてのオーディオエフェクトの音量を設定します。
pauseAudioEffect	破棄されたインターフェース：オーディオエフェクトを一時停止します。
resumeAudioEffect	破棄されたインターフェース：オーディオエフェクトを再開します。

デバイスおよびネットワークテスト

API	説明
startSpeedTest	ネットワークスピードテストを開始します（通話品質への影響を避けるため、ビデオ通話中はテストしないでください）。
stopSpeedTest	ネットワークスピードテストを停止します。
startCameraDeviceTest	カメラテストを開始します。
stopCameraDeviceTest	カメラテストを停止します。
startMicDeviceTest	マイクテストを開始します。
stopMicDeviceTest	マイクテストを停止します。

API	説明
startSpeakerDeviceTest	スピーカーテストを開始します。
stopSpeakerDeviceTest	スピーカーテストを停止します。

LOG関連インターフェース関数

API	説明
getSDKVersion	SDKバージョン情報を取得します。
setLogLevel	Log出力レベルを設定します。
setConsoleEnabled	コンソールのログプリントを有効または無効にします。
setLogCompressEnabled	Logのローカル圧縮を有効または無効にします。
setLogDirPath	ログ保存パスを設定します。
setLogCallback	ログコールバックを設定します。
callExperimentalAPI	試験的APIインターフェースを呼び出します。

使用停止インターフェース関数

API	説明
setMicVolumeOnMixing	v6.9バージョンから破棄します。

TRTCCallback @ TXLiteAVSDK

Tencent Cloudビデオ通話機能のコールバックインターフェース。

エラーイベントおよび警告イベント

API	説明
onError	エラーコールバック：SDKがリカバリー不能なエラーは、監視する必要があり、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。
onWarning	アラートコールバック：ラグやリカバリー不能なデコードの失敗など、非常に重大な問題を通知するために使用されます。

ルームイベントコールバック

API	説明
onEnterRoom	入室済みのコールバックです。
onExitRoom	退室のイベントコールバックです。
onSwitchRole	ロール切り替えのイベントコールバックです。
onConnectOtherRoom	ルーム間マイク接続（キャスタールーム間PK）リクエスト結果のコールバックです。
onDisconnectOtherRoom	ルーム間マイク接続（キャスタールーム間PK）終了結果のコールバックです。
onSwitchRoom	ルームを切り替えます。

メンバーイベントコールバック

API	説明
onRemoteUserEnterRoom	ユーザーが現在のルームに入室します。
onRemoteUserLeaveRoom	ユーザーが現在のルームを退室します。
onUserVideoAvailable	ユーザーがカメラからのビデオを有効にしているかどうか。
onUserSubStreamAvailable	ユーザーが画面共有を有効にしているかどうか。
onUserAudioAvailable	ユーザーがオーディオのアップストリームを有効にしているかどうか。
onFirstVideoFrame	ローカルまたはリモートユーザーの最初のフレーム画面のレンダリングを開始します。
onFirstAudioFrame	リモートユーザーの最初のフレームのオーディオ再生を開始します（現在、ローカル音声は通知しません）。
onSendFirstLocalVideoFrame	最初のフレームのローカルビデオデータが送信されました。
onSendFirstLocalAudioFrame	最初のフレームのローカルオーディオデータが送信されました。
onUserEnter	破棄されたインターフェース：キャスターが現在のルームに入室します。
onUserExit	破棄されたインターフェース： キャスターが現在のルームを退室します。

統計および品質コールバック

API	説明
onNetworkQuality	ネットワーク品質：このコールバックは2秒ごとに1度トリガーされ、現在のネットワークのアップストリームとダウンストリーム品質を統計します。
onStatistics	技術指標統計のコールバックです。

サーバーイベントコールバック

API	説明
onConnectionLost	SDKがサーバーとの接続を切断します。
onTryToReconnect	SDKがサーバーとの再接続を試行中です。
onConnectionRecovery	SDKがサーバーとの接続を再開します。
onSpeedTest	破棄されたインターフェース：サーバースピードテストのコールバックです。SDKは複数のサーバーIPに対するスピードテストを実行し、IPごとのスピードテスト結果をこのコールバックを介して通知します。
onSpeedTestResult	ネットワークスピードテストの結果のコールバックです。

ハードウェアデバイスイベントコールバック

API	説明
onCameraDidReady	カメラの準備が完了しました。
onMicDidReady	マイクの準備が完了しました。
onUserVoiceVolume	音量レベルをリマインドするためのコールバックです。userIdごとの音量とリモートの総音量が含まれます。ローカルユーザーのuseridは''です。
onDeviceChange	ローカルデバイスオン/オフのコールバックです。
onTestMicVolume	マイクテスト音量のコールバックです。
onTestSpeakerVolume	スピーカーテスト音量のコールバックです。
onAudioDeviceCaptureVolumeChanged	現在のオーディオキャプチャデバイス音量変更のコールバックです。
onAudioDevicePlayoutVolumeChanged	現在のオーディオ再生デバイス音量変更のコールバックです。

カスタムメッセージ受信のコールバック

API	説明
onRecvCustomCmdMsg	カスタムメッセージ受信のコールバックです。
onMissCustomCmdMsg	カスタムメッセージ消失のコールバックです。
onRecvSEIMsg	SEIメッセージ受信のコールバックです。

CDNバイパスリレーコールバック

API	説明
onStartPublishing	Tencent CloudのライブCDNへのプッシュ開始のコールバックです。TRTCCloudのstartPublishing()インターフェースに対応します。
onStopPublishing	Tencent CloudのライブCDNへのプッシュ停止のコールバックです。TRTCCloudのstopPublishing()インターフェースに対応します。
onStartPublishCDNStream	CDNへのRelayed Push起動完了のコールバックです。
onStopPublishCDNStream	CDNへのRelayed Push停止完了のコールバックです。
onSetMixTranscodingConfig	クラウドのミクスストリーミングトランスコードパラメータ設定のコールバックです。TRTCCloudのsetMixTranscodingConfig()インターフェースに対応します。

システム音量キャプチャコールバック

API	説明
onSystemAudioLoopbackError	システム音量キャプチャステータスのコールバックです（Macのみで有効）。

オーディオエフェクトコールバック

API	説明
onAudioEffectFinished	破棄されたインターフェース：オーディオエフェクト再生終了のコールバックです。

画面共有コールバック

API	説明

API	説明
onScreenCaptureCovered	SDKは画面共有ウィンドウがブロックされ、正常にキャプチャできないことをこのコールバックを介して通知します。このコールバックでユーザーにウィンドウのブロックを解除するよう通知できます。
onScreenCaptureStarted	SDKは画面共有の開始をこのコールバックを介して通知します。
onScreenCapturePaused	SDKは画面共有の一時停止をこのコールバックを介して通知します。
onScreenCaptureResumed	SDKは画面共有の再開をこのコールバックを介して通知します。
onScreenCaptureStopped	SDKは画面共有の停止をこのコールバックを介して通知します。

スクリーンキャプチャコールバック

API	説明
onSnapshotComplete	SDKはスクリーンキャプチャの完了をこのコールバックを介して通知します。

BGMミキシングイベントコールバック

API	説明
onPlayBGMBegin	破棄されたインターフェース：BGMの再生を開始します。
onPlayBGMProgress	破棄されたインターフェース：BGM再生の進捗です。
onPlayBGMComplete	破棄されたインターフェース：BGMの再生を終了します。

主要なタイプの定義

主要なタイプ

タイプ名	説明
TRTCParams	入室関連パラメータです。
TRTCVideoEncParam	ビデオコーデックパラメータです。
TRTCNetworkQosParam	ネットワークトラフィックコントロール関連パラメータです。
TRTCQualityInfo	ビデオ品質です。

タイプ名	説明
TRTCVolumeInfo	音量レベルです。
TRTCSpeedTestResult	ネットワークスピードテスト結果です。
TRTCMixUser	クラウドミクスストリーミングにおける各サブ画面の位置情報です。
TRTCTranscodingConfig	クラウドミクスストリーミング（トランスコード）の設定です。
TRTCPublishCDNParam	CDN Relayed Pushパラメータです。
TRTCAudioRecordingParams	録音パラメータです。
TRTCLocalStatistics	自身のローカルオーディオビデオ統計情報です。
TRTCRemoteStatistics	リモートメンバーのオーディオビデオ統計情報です。
TRTCStatistics	統計データです。

列挙値

列挙	説明
TRTCVideoResolution	ビデオ解像度です。
TRTCVideoResolutionMode	ビデオ解像度モードです。
TRTCVideoStreamType	ビデオストリームタイプです。
TRTCQuality	画質レベルです。
TRTCVideoFillMode	ビデオ画面塗りつぶしモードです。
TRTCBeautyStyle	美顔（美肌）アルゴリズムです。
TRTCAppScene	ユースケースです。
TRTCRoleType	ロールです。ライブストリーミングシナリオ（TRTCAppSceneLIVE）のみに適しています。
TRTCQosControlMode	トラフィックコントロールモードです。
TRTCVideoQosPreference	画質の好みです。
TRTCDeviceState	デバイスの操作です。
TRTCDeviceType	デバイスのタイプです。

列挙	説明
TRTCWaterMarkSrcType	ウォーターマーク画像のオリジナルタイプです。
TRTCTranscodingConfigMode	ミクスストリーミングパラメータ設定モードです。

エラーコード

最終更新日：：2022-06-28 11:11:04

エラーコードリスト

基本的なエラーコード

記号	値	意味
ERR_NULL	0	エラーなし

入室関連のエラーコード

TRTCCloud.enterRoom() 入室に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.onEnterRoom()、TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_ENTER_FAIL	-3301	入室の失敗
ERR_ENTER_ROOM_PARAM_NULL	-3316	入室パラメータが空です。TRTCCloud.enterRoom():インターフェースの呼び出しが有効なparamで渡されるかどうか確認してください
ERR_SDK_APPID_INVALID	-3317	入室パラメータsdkAppIdエラー
ERR_ROOM_ID_INVALID	-3318	入室パラメータroomIdエラー
ERR_USER_ID_INVALID	-3319	入室パラメータuserIDが正しくありません
ERR_USER_SIG_INVALID	-3320	入室パラメータuserSigが正しくありません
ERR_ROOM_REQUEST_ENTER_ROOM_TIMEOUT	-3308	入室リクエストがタイムアウトしました。ネットワークをご確認ください
ERR_SERVER_INFO_PRIVILEGE_FLAG_ERROR	-100006	パーミッションビットのチェックに失敗しました。privateMapKeyが正しいかどうか確認してください

記号	値	意味
ERR_SERVER_INFO_SERVICE_SUSPENDED	-100013	サービスをご利用いただけません。パッケージの残りの分数が0より大きいかどうか、Tencent Cloudアカウントの支払いが遅延していないかどうかご確認ください
ERR_SERVER_INFO_ECDH_GET_TINYID	-100018	userSigのチェックに失敗しました。userSigが正しいかどうか確認してください

退室関連のエラーコード

TRTCCloud.exitRoom() 入室に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.onEnterRoom()、TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_REQUEST_QUIT_ROOM_TIMEOUT	-3325	退室リクエストのタイムアウト

デバイス（カメラ、マイク、スピーカー）関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_CAMERA_START_FAIL	-1301	カメラの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、カメラのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください
ERR_CAMERA_NOT_AUTHORIZED	-1314	カメラデバイスの権限が承認されていません。通常はモバイルデバイスに表示されます。ユーザーから権限承認を拒否されている可能性があります
ERR_CAMERA_SET_PARAM_FAIL	-1315	カメラパラメータ設定エラー（パラメータがサポートされていないか、その他）
ERR_CAMERA_OCCUPY	-1316	カメラが使用中なので、他のカメラの起動を試みることができます

記号	値	意味
ERR_MIC_START_FAIL	-1302	マイクの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、マイクのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください
ERR_MIC_NOT_AUTHORIZED	-1317	マイクデバイスの権限が承認されていません。通常はモバイルデバイスに表示されます。ユーザーから権限承認を拒否されている可能性があります
ERR_MIC_SET_PARAM_FAIL	-1318	マイクパラメータの設定に失敗しました
ERR_MIC_OCCUPY	-1319	マイクが使用中です。例えば、モバイルデバイスが通話中の場合、マイクの起動は失敗します
ERR_MIC_STOP_FAIL	-1320	マイクの停止に失敗しました
ERR_SPEAKER_START_FAIL	-1321	スピーカーの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、スピーカーのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください
ERR_SPEAKER_SET_PARAM_FAIL	-1322	スピーカーパラメータの設定に失敗しました
ERR_SPEAKER_STOP_FAIL	-1323	スピーカーの停止に失敗しました

画面共有関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味

記号	値	意味
ERR_SCREEN_CAPTURE_START_FAIL	-1308	画面録画の開始に失敗しました。モバイルデバイスに表示されるときは、ユーザーから権限承認を拒否されている可能性があります。WindowsまたはMacシステムのデバイスに表示される場合は、画面録画インターフェースのパラメータが要件を満たしているかご確認ください
ERR_SCREEN_CAPTURE_UNSURPORT	-1309	画面録画に失敗しました。Androidプラットフォームでは5.0以降のシステム、iOSプラットフォームでは11.0以降のシステムが必要です
ERR_SERVER_CENTER_NO_PRIVILEDGE_PUSH_SUB_VIDEO	-102015	サブチャネルのアップストリームの権限がありません
ERR_SERVER_CENTER_ANOTHER_USER_PUSH_SUB_VIDEO	-102016	他のユーザーがサブチャネルのアップストリームを行っています
ERR_SCREEN_CAPTURE_STOPPED	-7001	画面録画がシステムによって停止されました

コーデック関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_VIDEO_ENCODE_FAIL	-1303	ビデオフレームのエンコードに失敗しました。例えば、iOSデバイスを他のアプリケーションに切り替えると、システムによってハードウェアエンコーダが解放される場合があります。切り替えて元に戻すと、ハードウェアエンコーダが再起動する前にスローされる場合があります

記号	値	意味
ERR_UNSUPPORTED_RESOLUTION	-1305	サポートされていないビデオ解像度
ERR_AUDIO_ENCODE_FAIL	-1304	オーディオフレームのエンコードに失敗しました。例えば、渡されたカスタムオーディオデータを、SDKで処理することはできません
ERR_UNSUPPORTED_SAMPLERATE	-1306	サポートされていないオーディオサンプルレート

ユーザー定義キャプチャ関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_PIXEL_FORMAT_UNSUPPORTED	-1327	設定されたpixel formatはサポートされていません
ERR_BUFFER_TYPE_UNSUPPORTED	-1328	設定されたbuffer typeはサポートされていません

CDNバインディングおよびミクスストリーミング関連のエラーコード

コールバック関数TRTCCloudDelegate.onStartPublishing()、TRTCCloudDelegate.onSetMixTranscodingConfig()によって関連通知をキャプチャできます。

記号	値	意味
ERR_PUBLISH_CDN_STREAM_REQUEST_TIME_OUT	-3321	バイパス転送リクエストのタイムアウト
ERR_CLOUD_MIX_TRANSCODING_REQUEST_TIME_OUT	-3322	クラウドミクスストリーミングリクエストのタイムアウト
ERR_PUBLISH_CDN_STREAM_SERVER_FAILED	-3323	バイパス転送リターンパケットの異常
ERR_CLOUD_MIX_TRANSCODING_SERVER_FAILED	-3324	クラウドミクスストリーミング・リターンパケットの異常
ERR_ROOM_REQUEST_START_PUBLISHING_TIMEOUT	-3333	Tencent CloudへのライブCDNのプッシュ開始シグナリングのタイムアウト
ERR_ROOM_REQUEST_START_PUBLISHING_ERROR	-3334	Tencent CloudへのライブCDNのプッシュ開始シグナリングの異常

記号	値	意味
ERR_ROOM_REQUEST_STOP_PUBLISHING_TIMEOUT	-3335	Tencent Cloudへのライブ CDNのプッシュ停止シグナリングのタイムアウト
ERR_ROOM_REQUEST_STOP_PUBLISHING_ERROR	-3336	Tencent Cloudへのライブ CDNのプッシュ停止シグナリングの異常

ルーム間マイク接続関連のエラーコード

TRTCCloud.ConnectOtherRoom() ルーム間マイク接続に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.onConnectOtherRoom()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_REQUEST_CONN_ROOM_TIMEOUT	-3326	マイク接続リクエストのタイムアウト
ERR_ROOM_REQUEST_DISCONN_ROOM_TIMEOUT	-3327	マイク接続からの退出リクエストのタイムアウト
ERR_ROOM_REQUEST_CONN_ROOM_INVALID_PARAM	-3328	無効なパラメータ
ERR_CONNECT_OTHER_ROOM_AS_AUDIENCE	-3330	現在のロールは視聴者であり、ルーム間マイク接続をリクエストまたは切断することはできません。まずキャスターにswitchRole()する必要があります
ERR_SERVER_CENTER_CONN_ROOM_NOT_SUPPORT	-102031	ルーム間マイク接続はサポートされていません

記号	値	意味
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_NUM	-102032	ルーム間マイク接続の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_RETRY_TIMES	-102033	ルーム間マイク接続の再試行回数の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_REQ_TIMEOUT	-102034	ルーム間マイク接続リクエストのタイムアウト
ERR_SERVER_CENTER_CONN_ROOM_REQ	-102035	ルーム間マイク接続リクエストの形式エラー
ERR_SERVER_CENTER_CONN_ROOM_NO_SIG	-102036	ルーム間マイク接続に署名がありません
ERR_SERVER_CENTER_CONN_ROOM_DECRYPT_SIG	-102037	ルーム間マイク接続の署名復号に失敗しました
ERR_SERVER_CENTER_CONN_ROOM_NO_KEY	-102038	ルーム間マイク接続の署名復号キーが見つかりませんでした
ERR_SERVER_CENTER_CONN_ROOM_PARSE_SIG	-102039	ルーム間マイク接続の署名解析エラー
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SIG_TIME	-102040	ルーム間マイク接続の署名タイムスタンプエラー
ERR_SERVER_CENTER_CONN_ROOM_SIG_GROUPID	-102041	ルーム間マイク接続の署名ルーム番号が一致しません

記号	値	意味
ERR_SERVER_CENTER_CONN_ROOM_NOT_CONNED	-102042	ルーム間マイク接続の署名ユーザー名が一致しません
ERR_SERVER_CENTER_CONN_ROOM_USER_NOT_CONNED	-102043	このユーザーはマイク接続を開始していません
ERR_SERVER_CENTER_CONN_ROOM_FAILED	-102044	ルーム間マイク接続に失敗しました
ERR_SERVER_CENTER_CONN_ROOM_CANCEL_FAILED	-102045	ルーム間マイク接続の取り消しに失敗しました
ERR_SERVER_CENTER_CONN_ROOM_CONNED_ROOM_NOT_EXIST	-102046	マイク接続されたルームが存在しません
ERR_SERVER_CENTER_CONN_ROOM_CONNED_REACH_MAX_ROOM	-102047	マイク接続されたルームがマイク接続の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_NOT_EXIST	-102048	マイク接続されたユーザーが存在しません
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_DELETED	-102049	マイク接続されたユーザーが削除されました
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_FULL	-102050	マイク接続されたユーザーがリソースの上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SEQ	-102051	マイク接続リクエストの番号が乱れています

アラートコード表

アラートコードを特に注意する必要はありません。必要に応じて、現在のユーザーに対応するプロンプトを表示するかどうかを選択できます。

記号	値	意味
WARNING_HW_ENCODER_START_FAIL	1103	ハードウェアエンコーディングの起動に問題が発生した場合、自動的にソフトウェアエンコーディングに切り替わります
WARNING_VIDEO_ENCODER_SW_TO_HW	1107	現在のCPU使用率が高すぎてソフトウェアエンコーディング要件を満たせないため、自動的にハードウェアエンコーディングに切り替わります
WARNING_INSUFFICIENT_CAPTURE_FPS	1108	カメラのキャプチャフレームレートが不足しています。美顔アルゴリズムを備えた一部のAndroidスマートフォンには表示されます
WARNING_SW_ENCODER_START_FAIL	1109	ソフトウェアエンコーディングの開始に失敗しました
WARNING_REDUCE_CAPTURE_RESOLUTION	1110	現在のフレームレートとパフォーマンスの最適解を満たすため、カメラのキャプチャ解像度が低下します。
WARNING_CAMERA_DEVICE_EMPTY	1111	使用可能なカメラデバイスが検出されません
WARNING_CAMERA_NOT_AUTHORIZED	1112	ユーザーは、現在のアプリケーションにカメラ使用権限を承認していません
WARNING_MICROPHONE_DEVICE_EMPTY	1201	使用可能なマイクデバイスが検出されません
WARNING_SPEAKER_DEVICE_EMPTY	1202	使用可能なスピーカーデバイスが検出されません
WARNING_MICROPHONE_NOT_AUTHORIZED	1203	ユーザーは、現在のアプリケーションにマイク使用権限を承認していません

記号	値	意味
WARNING_MICROPHONE_DEVICE_ABNORMAL	1204	オーディオキャプチャデバイスが利用できません（使用中であるなど）
WARNING_SPEAKER_DEVICE_ABNORMAL	1205	オーディオ再生デバイスが利用できません（使用中であるなど）
WARNING_VIDEO_FRAME_DECODE_FAIL	2101	現在のビデオフレームのデコードに失敗しました
WARNING_AUDIO_FRAME_DECODE_FAIL	2102	現在のオーディオフレームのデコードに失敗しました
WARNING_VIDEO_PLAY_LAG	2105	現在、ビデオ再生時にラグが発生しています
WARNING_HW_DECODER_START_FAIL	2106	ハードウェアデコードの開始に失敗しました。ソフトウェアデコードを使用してください
WARNING_VIDEO_DECODER_HW_TO_SW	2108	現在のストリームの最初のIフレームのハードウェアデコードが失敗しました。SDKは自動的にソフトウェアデコードに切り替えます
WARNING_SW_DECODER_START_FAIL	2109	ソフトウェアデコーダの起動に失敗しました
WARNING_VIDEO_RENDER_FAIL	2110	ビデオレンダリングに失敗しました
WARNING_START_CAPTURE_IGNORED	4000	キャプチャ中のため、キャプチャの開始は無視されます
WARNING_AUDIO_RECORDING_WRITE_FAIL	7001	オーディオ記録のファイルへの書き込みに失敗しました
WARNING_ROOM_DISCONNECT	5101	ネットワーク接続が切断されました
WARNING_IGNORE_UPSTREAM_FOR_AUDIENCE	6001	現在、視聴者ロールです。アップストリームのオーディオビデオデータを無視します
WARNING_NET_BUSY	1101	ネットワーク状態の不良：アップストリーム帯域幅が小さすぎて、データのアップロード時に問題が生じます

記号	値	意味
WARNING_RTMP_SERVER_RECONNECT	1102	CSSプッシュのときにネットワークが切断しましたが、自動再接続が開始されました（自動再接続は連続3回を超えて失敗すると、それ以上は行われません）
WARNING_LIVE_STREAM_SERVER_RECONNECT	2103	ライブストリーミングのプルのときにネットワークが切断しましたが、自動再接続が開始されました（自動再接続は連続3回を超えて失敗すると、それ以上は行われません）
WARNING_RECV_DATA_LAG	2104	不安定なネットワークパケット：ダウンストリーム帯域幅が不足しているか、キャスター側からのストリーミングが不均一であることが原因と思われます
WARNING_RTMP_DNS_FAIL	3001	ライブストリーミング、DNS解決に失敗しました
WARNING_RTMP_SEVER_CONN_FAIL	3002	ライブストリーミング、サーバーの接続に失敗しました
WARNING_RTMP_SHAKE_FAIL	3003	ライブストリーミング、RTMPサーバーとのハンドシェイクに失敗しました
WARNING_RTMP_SERVER_BREAK_CONNECT	3004	ライブストリーミング、サーバーが自動的に切断しました
WARNING_RTMP_READ_WRITE_FAIL	3005	ライブストリーミング、RTMPの読み取り/書き込みに失敗し、接続が切断されます
WARNING_RTMP_WRITE_FAIL	3006	ライブストリーミング、RTMP書き込みエラー（SDK内部エラーコードは外部にスローされません）
WARNING_RTMP_READ_FAIL	3007	ライブストリーミング、RTMP読み取りエラー（SDK内部エラーコードは外部にスローされません）

記号	値	意味
WARNING_RTMP_NO_DATA	3008	ライブストリーミング、30秒以上データが送信されない場合、自動的に接続が切断されます
WARNING_PLAY_LIVE_STREAM_INFO_CONNECT_FAIL	3009	ライブストリーミング、connectサーバー呼び出しに失敗しました（SDK内部エラーコードは外部にスローされません）
WARNING_NO_STEAM_SOURCE_FAIL	3010	ライブストリーミング、接続に失敗しました。このストリームアドレスにビデオはありません。（SDK内部エラーコードは外部にスローされません）
WARNING_ROOM_RECONNECT	5102	ネットワークが切断され、自動再接続が開始されました
WARNING_ROOM_NET_BUSY	5103	ネットワーク状態の不良：アップストリーム帯域幅が小さすぎて、データのアップロード時に問題が生じます

Flutter
API概要

最終更新日：：2022-04-08 16:58:45

TRTCCloud

基本的な方法

API	説明
sharedInstance	TRTCCloudシングルトンを作成します。
destroySharedInstance	TRTCCloudシングルトンを廃棄します。
registerListener	イベント監視を設定します。
unRegisterListener	イベント監視を削除します。

ルーム関連インターフェース関数

API	説明
enterRoom	ルームに参加し、ルームが存在しない場合、システムが自動的に新しいルームを作成します。
exitRoom	ルームから退出します。
switchRole	ロールを切り替えます。ライブストリーミングシナリオ（TRTC_APP_SCENE_LIVEおよびTRTC_APP_SCENE_VOICE_CHATROOM）のみに適しています。
setDefaultStreamRecvMode	オーディオビデオの受信モードを設定します。有効にするには、入室する前に設定してください。
connectOtherRoom	ルーム間通話（キャスターPK）リクエストします。
disconnectOtherRoom	ルーム間通話から退出します。
switchRoom	ルームを切り替えます。

CDN関連インターフェース関数

API	説明
startPublishing	Tencent CloudへのライブCDNのプッシュを開始します。
stopPublishing	Tencent CloudへのライブCDNのプッシュを停止します。
startPublishCDNStream	他社のクラウドへのライブCDNのリツイートを開始します。
stopPublishCDNStream	Tencent Cloud以外のアドレスへのリツイートを停止します。
setMixTranscodingConfig	クラウドのミクスストリーミングトランスコードパラメータを設定します。

ビデオ関連インターフェース関数

API	説明
startLocalPreview	ローカルビデオのプレビュー画面を開始します。
stopLocalPreview	ローカルビデオの収集およびプレビューを停止します。
muteLocalVideo	ローカルビデオデータのプッシュを一時停止/再開します。
startRemoteView	リモートビデオ画面の表示を開始します。
stopRemoteView	リモートビデオ画面の表示を停止すると同時に、リモートユーザーのビデオデータトラフィックのプルを停止します。
stopAllRemoteView	すべてのリモートビデオ画面の表示を停止すると同時に、リモートユーザーのビデオデータトラフィックのプルを停止します。
muteRemoteVideoStream	指定したたリモートビデオストリームの受信を一時停止/再開します。
muteAllRemoteVideoStreams	すべてのリモートビデオトラフィックを一時停止/再開します。
setVideoEncoderParam	ビデオエンコーダに関するパラメータを設定します。
setNetworkQosParam	ネットワークフロー制御に関するパラメータを設定します。
setLocalRenderParams	ローカル画像のレンダリングモードを設定します。
setRemoteRenderParams	リモート画像に関するパラメータを設定します。
setVideoEncoderRotation	ビデオコーデック出力先の画面方向、すなわちリモートユーザーが見ている画面方向とサーバーで記録された画面方向を設定します。
setVideoEncoderMirror	エンコーダ出力の画面イメージモードを設定します。
setGSensorMode	重力感知の適応モードを設定します。

API	説明
enableEncSmallVideoStream	大小画面の2ウェイコーディングモードを設定します。
setRemoteVideoStreamType	指定したuidの大画面または小画面を選択します。
snapshotVideo	ビデオ画面のスクリーンショット。

オーディオ関連インターフェース関数

API	説明
startLocalAudio	ローカルオーディオの収集およびアップストリームを有効にします。
stopLocalAudio	ローカルオーディオの収集およびアップストリームを無効にします。
muteLocalAudio	ローカルオーディオをミュート/ミュート解除します。
setVideoMuteImage	ローカルビデオのプッシュを一時停止するときにプッシュする画像を設定します。
setAudioRoute	オーディオルーティングを設定します。
muteRemoteAudio	指定したリモートユーザーのサウンドをミュート/ミュート解除します。
muteAllRemoteAudio	すべてのユーザーのサウンドをミュート/ミュート解除します。
setAudioCaptureVolume	SDKの収音量を設定します。
getAudioCaptureVolume	SDKの収音量を取得します。
setAudioPlayoutVolume	SDKの再生音量を設定します。
getAudioPlayoutVolume	SDKの再生音量を取得します。
enableAudioVolumeEvaluation	音量通知を有効にします。
startAudioRecording	録音を開始します。
stopAudioRecording	録音を停止します。
setSystemVolumeType	通話時に使用するシステム音量タイプを設定します。

デバイス管理インターフェース

API	説明

API	説明
getDeviceManager	デバイス管理モジュールを取得し、インターフェースの詳細についてデバイス管理インターフェースドキュメントをご参照ください。

美顔フィルタに関するインターフェース関数

API	説明
getBeautyManager	美顔管理オブジェクトを取得し、インターフェースの詳細について美顔管理ドキュメントをご参照ください。
setWatermark	ウォーターマークを追加します。

音楽の特殊効果および声の特殊効果

API	説明
getAudioEffectManager	音色管理クラス TXAudioEffectManagerを取得し、BGM、短音色および声の特殊効果の管理に使用され、インターフェースの詳細について音色管理ドキュメントをご参照ください。

サブストリーム関連インターフェース関数

API	説明
startScreenCapture	画面共有をオンにします。
stopScreenCapture	画面収集をオフにします。
pauseScreenCapture	画面共有を一時停止します。
resumeScreenCapture	画面共有を再開します。

カスタムメッセージ送信

API	説明
sendCustomCmdMsg	カスタムメッセージをルーム内のすべてのユーザーに送信します。
sendSEIMsg	小さいデータ量のカスタムデータをビデオフレーム内に埋め込みます。

ネットワークテスト

API	説明
startSpeedTest	ネットワーク速度測定（ビデオ通話中にはテストしないでください。通話品質に影響するおそれがあります）。
stopSpeedTest	サーバーの速度測定を停止します。

ログ関連インターフェース関数

API	説明
getSDKVersion	SDKのバージョン情報を取得します。
setLogLevel	Logの出力レベルを設定します。
setLogDirPath	ログ保存パスを変更します。
setLogCompressEnabled	Logのローカル圧縮を有効または無効にします。
setConsoleEnabled	コンソールログの印刷を有効または無効にします。

TRTCCloudListener

Tencent Cloudのビデオ通話機能のイベントコールバックインターフェース。

エラーイベントおよび警告イベント

API	説明
onError	エラーコールバック。SDKの回復不可能なエラーを示します。監視し、状況に応じてユーザーに適切な画面メッセージを表示しなければなりません。
onWarning	警告コールバック。ラグやリカバリ可能なデコードの失敗など、重大度の低い問題を通知します。

ルームイベントのコールバック

API	説明
onEnterRoom	ルーム参加済のコールバック。
onExitRoom	ルーム退出のイベントコールバック。
onSwitchRole	ロール切り替えのイベントコールバック。

API	説明
onConnectOtherRoom	ルーム間通話（キャスターPK）リクエストの結果コールバック。
onDisConnectOtherRoom	ルーム間通話（キャスターPK）終了の結果コールバック。
onSwitchRoom	ルーム切り替え(switchRoom)の結果コールバック。

メンバーイベントコールバック

API	説明
onRemoteUserEnterRoom	現在ルームに参加するユーザーがいます。
onRemoteUserLeaveRoom	現在のルームから退出するユーザーがいます。
onUserVideoAvailable	リモートユーザーに再生可能なメインチャネル画面があるかどうか（通常はカメラに用いられます）。
onUserSubStreamAvailable	リモートユーザーに再生可能なサブチャネル画面があるかどうか（通常は画面共有に用いられます）。
onUserAudioAvailable	リモートユーザーが再生可能なオーディオデータを持っていますか。
onFirstVideoFrame	ローカルまたはリモートユーザーの最初のフレーム画面のレンダリングを開始します。
onFirstAudioFrame	リモートユーザーの最初のフレームオーディオの再生を開始します（ローカルサウンドは通知されません）。
onSendFirstLocalVideoFrame	最初のフレームのローカルビデオデータが送信されました。
onSendFirstLocalAudioFrame	最初のフレームのローカルオーディオデータが送信されました。

BGM再生のコールバックインターフェース

BGM再生のコールバックインターフェースです。

API	説明
onMusicObserverStart	音楽再生開始のコールバック通知。
onMusicObserverPlayProgress	音楽再生の進捗状況のコールバック通知。
onMusicObserverComplete	音楽再生終了のコールバック通知。

統計および品質コールバック

API	説明
onNetworkQuality	ネットワーク品質。このコールバックは2秒ごとに1度トリガーされ、現在のネットワークのアップストリームとダウンストリーム品質の統計を行います。
onStatistics	技術指標統計コールバック。

サーバーイベントコールバック

API	説明
onConnectionLost	SDKとサーバーの接続が切断されました。
onTryToReconnect	SDKがサーバーに再接続しようとします。
onConnectionRecovery	SDKとサーバーの接続が回復しました。
onSpeedTest	サーバースピードテストのコールバックです。SDKは複数のサーバーIPに対するスピードテストを実行し、IPごとのスピードテスト結果をこのコールバックを使用して通知します。

ハードウェアデバイスイベントコールバック

API	説明
onCameraDidReady	カメラの準備ができました。
onMicDidReady	マイクの準備ができました。
onUserVoiceVolume	音量レベルをリマインドするためのコールバックです。userIdごとの音量とリモートの総音量が含まれます。

カスタムメッセージ受信のコールバック

API	説明
onRecvCustomCmdMsg	カスタムメッセージ受信のコールバック。
onMissCustomCmdMsg	カスタムメッセージ紛失のコールバック。
onRecvSEIMsg	SEIメッセージ受信のコールバック。

CDNバイパス転送コールバック

API	説明
onStartPublishing	Tencent CloudへのライブCDN プッシュ開始のコールバック。TRTCCloudのstartPublishing()インターフェースに対応します。
onStopPublishing	Tencent CloudへのライブCDN プッシュ停止のコールバック。TRTCCloudのstopPublishing()インターフェースに対応します。
onStartPublishCDNStream	Relayed Pushの起動からCDN完了までのコールバック。
onStopPublishCDNStream	Relayed Push停止からCDN完了までのコールバック。
onSetMixTranscodingConfig	クラウドのミクスストリーミングトランスコードパラメータ設定のコールバック。TRTCCloudのsetMixTranscodingConfig()インターフェースに対応します。

画面共有コールバック

API	説明
onScreenCaptureStarted	画面共有開始時にSDKがこのコールバックで通知します
onScreenCapturePaused	画面共有がpauseScreenCapture()を呼び出すときに、SDKがこのコールバックで通知します。
onScreenCaptureResumed	画面共有がresumeScreenCapture()を呼び出すときに、SDKがこのコールバックで通知します。
onScreenCaptureStoped	画面共有停止ときにはSDKがこのコールバックで通知します。

スクリーンキャプチャコールバック

API	説明
onSnapshotComplete	スクリーンショット完了時のコールバック。

主要なタイプの定義

タイプ名	説明
TRTCCloudDef	主要なタイプ定義用変数。
TRTCParams	入室パラメータ。

タイプ名	説明
TRTCSwitchRoomConfig	ルームパラメータ切り替え用パラメータ。
TRTCVideoEncParam	ビデオエンコーダパラメータ。
TRTCNetworkQosParam	ネットワークフロー制御に関するパラメータ。
TRTCRenderParams	リモート画像パラメータ。
TRTCMixUser	クラウドミクスストリーミングにおける各チャンネル子画面の位置情報。
TRTCTranscodingConfig	クラウドミクスストリーミング（トランスコード）の構成。
TXVoiceChangerType	ボイス変更タイプの定義（ロリ、オヤジ、ヘビーメタル、外国人など）。
TXVoiceReverbType	ボイス変更タイプの定義（KTV、小ルーム、大会堂、低音、広音など）。
AudioMusicParam	音楽と声設定インターフェースパラメータ。
TRTCAudioRecordingParams	録音パラメータ。
TRTCPublishCDNParam	CDNリツイートパラメータ。

エラーコード

最終更新日：：2022-06-28 11:04:41

エラーコードリスト

基本的なエラーコード

記号	値	意味
ERR_NULL	0	エラーなし

入室関連のエラーコード

TRTCCloud.enterRoom() 入室に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.onEnterRoom()、TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_ENTER_FAIL	-3301	入室の失敗
ERR_ENTER_ROOM_PARAM_NULL	-3316	入室パラメータが空です。TRTCCloud.enterRoom():インターフェースの呼び出しが有効なparamで渡されるかどうか確認してください
ERR_SDK_APPID_INVALID	-3317	入室パラメータsdkAppIdエラー
ERR_ROOM_ID_INVALID	-3318	入室パラメータroomIdエラー
ERR_USER_ID_INVALID	-3319	入室パラメータuserIDが正しくありません
ERR_USER_SIG_INVALID	-3320	入室パラメータuserSigが正しくありません
ERR_ROOM_REQUEST_ENTER_ROOM_TIMEOUT	-3308	入室リクエストがタイムアウトしました。ネットワークをご確認ください
ERR_SERVER_INFO_PRIVILEGE_FLAG_ERROR	-100006	パーミッションビットのチェックに失敗しました。privateMapKeyが正しいかどうか確認してください

記号	値	意味
ERR_SERVER_INFO_SERVICE_SUSPENDED	-100013	サービスをご利用いただけません。パッケージの残りの分数が0より大きいかどうか、Tencent Cloudアカウントの支払いが遅延していないかどうかご確認ください
ERR_SERVER_INFO_ECDH_GET_TINYID	-100018	userSigのチェックに失敗しました。userSigが正しいかどうか確認してください

退室関連のエラーコード

TRTCCloud.exitRoom() 入室に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.onEnterRoom()、TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_REQUEST_QUIT_ROOM_TIMEOUT	-3325	退室リクエストのタイムアウト

デバイス（カメラ、マイク、スピーカー）関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_CAMERA_START_FAIL	-1301	カメラの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、カメラのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください
ERR_CAMERA_NOT_AUTHORIZED	-1314	カメラデバイスの権限が承認されていません。通常はモバイルデバイスに表示されます。ユーザーから権限承認を拒否されている可能性があります
ERR_CAMERA_SET_PARAM_FAIL	-1315	カメラパラメータ設定エラー（パラメータがサポートされていないか、その他）
ERR_CAMERA_OCCUPY	-1316	カメラが使用中なので、他のカメラの起動を試みることができます

記号	値	意味
ERR_MIC_START_FAIL	-1302	マイクの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、マイクのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください
ERR_MIC_NOT_AUTHORIZED	-1317	マイクデバイスの権限が承認されていません。通常はモバイルデバイスに表示されます。ユーザーから権限承認を拒否されている可能性があります
ERR_MIC_SET_PARAM_FAIL	-1318	マイクパラメータの設定に失敗しました
ERR_MIC_OCCUPY	-1319	マイクが使用中です。例えば、モバイルデバイスが通話中の場合、マイクの起動は失敗します
ERR_MIC_STOP_FAIL	-1320	マイクの停止に失敗しました
ERR_SPEAKER_START_FAIL	-1321	スピーカーの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、スピーカーのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください
ERR_SPEAKER_SET_PARAM_FAIL	-1322	スピーカーパラメータの設定に失敗しました
ERR_SPEAKER_STOP_FAIL	-1323	スピーカーの停止に失敗しました

画面共有関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味

記号	値	意味
ERR_SCREEN_CAPTURE_START_FAIL	-1308	画面録画の開始に失敗しました。モバイルデバイスに表示されるときは、ユーザーから権限承認を拒否されている可能性があります。WindowsまたはMacシステムのデバイスに表示される場合は、画面録画インターフェースのパラメータが要件を満たしているかご確認ください
ERR_SCREEN_CAPTURE_UNSURPORT	-1309	画面録画に失敗しました。Androidプラットフォームでは5.0以降のシステム、iOSプラットフォームでは11.0以降のシステムが必要です
ERR_SERVER_CENTER_NO_PRIVILEDGE_PUSH_SUB_VIDEO	-102015	サブチャネルのアップストリームの権限がありません
ERR_SERVER_CENTER_ANOTHER_USER_PUSH_SUB_VIDEO	-102016	他のユーザーがサブチャネルのアップストリームを行っています
ERR_SCREEN_CAPTURE_STOPPED	-7001	画面録画がシステムによって停止されました

コーデック関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_VIDEO_ENCODE_FAIL	-1303	ビデオフレームのエンコードに失敗しました。例えば、iOSデバイスを他のアプリケーションに切り替えると、システムによってハードウェアエンコーダが解放される場合があります。切り替えて元に戻すと、ハードウェアエンコーダが再起動する前にスローされる場合があります

記号	値	意味
ERR_UNSUPPORTED_RESOLUTION	-1305	サポートされていないビデオ解像度
ERR_AUDIO_ENCODE_FAIL	-1304	オーディオフレームのエンコードに失敗しました。例えば、渡されたカスタムオーディオデータを、SDKで処理することはできません
ERR_UNSUPPORTED_SAMPLERATE	-1306	サポートされていないオーディオサンプルレート

ユーザー定義キャプチャ関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_PIXEL_FORMAT_UNSUPPORTED	-1327	設定されたpixel formatはサポートされていません
ERR_BUFFER_TYPE_UNSUPPORTED	-1328	設定されたbuffer typeはサポートされていません

CDNバインディングおよびミクスストリーミング関連のエラーコード

コールバック関数TRTCCloudDelegate.onStartPublishing()、TRTCCloudDelegate.onSetMixTranscodingConfig()によって関連通知をキャプチャできます。

記号	値	意味
ERR_PUBLISH_CDN_STREAM_REQUEST_TIME_OUT	-3321	バイパス転送リクエストのタイムアウト
ERR_CLOUD_MIX_TRANSCODING_REQUEST_TIME_OUT	-3322	クラウドミクスストリーミングリクエストのタイムアウト
ERR_PUBLISH_CDN_STREAM_SERVER_FAILED	-3323	バイパス転送リターンパケットの異常
ERR_CLOUD_MIX_TRANSCODING_SERVER_FAILED	-3324	クラウドミクスストリーミング・リターンパケットの異常
ERR_ROOM_REQUEST_START_PUBLISHING_TIMEOUT	-3333	Tencent CloudへのライブCDNのプッシュ開始シグナリングのタイムアウト
ERR_ROOM_REQUEST_START_PUBLISHING_ERROR	-3334	Tencent CloudへのライブCDNのプッシュ開始シグナリングの異常

記号	値	意味
ERR_ROOM_REQUEST_STOP_PUBLISHING_TIMEOUT	-3335	Tencent Cloudへのライブ CDNのプッシュ停止シグナリングのタイムアウト
ERR_ROOM_REQUEST_STOP_PUBLISHING_ERROR	-3336	Tencent Cloudへのライブ CDNのプッシュ停止シグナリングの異常

ルーム間マイク接続関連のエラーコード

TRTCCloud.ConnectOtherRoom() ルーム間マイク接続に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.onConnectOtherRoom()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_REQUEST_CONN_ROOM_TIMEOUT	-3326	マイク接続リクエストのタイムアウト
ERR_ROOM_REQUEST_DISCONN_ROOM_TIMEOUT	-3327	マイク接続からの退出リクエストのタイムアウト
ERR_ROOM_REQUEST_CONN_ROOM_INVALID_PARAM	-3328	無効なパラメータ
ERR_CONNECT_OTHER_ROOM_AS_AUDIENCE	-3330	現在のロールは視聴者であり、ルーム間マイク接続をリクエストまたは切断することはできません。まずキャスターにswitchRole()する必要があります
ERR_SERVER_CENTER_CONN_ROOM_NOT_SUPPORT	-102031	ルーム間マイク接続はサポートされていません

記号	値	意味
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_NUM	-102032	ルーム間マイク接続の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_RETRY_TIMES	-102033	ルーム間マイク接続の再試行回数の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_REQ_TIMEOUT	-102034	ルーム間マイク接続リクエストのタイムアウト
ERR_SERVER_CENTER_CONN_ROOM_REQ	-102035	ルーム間マイク接続リクエストの形式エラー
ERR_SERVER_CENTER_CONN_ROOM_NO_SIG	-102036	ルーム間マイク接続に署名がありません
ERR_SERVER_CENTER_CONN_ROOM_DECRYPT_SIG	-102037	ルーム間マイク接続の署名復号に失敗しました
ERR_SERVER_CENTER_CONN_ROOM_NO_KEY	-102038	ルーム間マイク接続の署名復号キーが見つかりませんでした
ERR_SERVER_CENTER_CONN_ROOM_PARSE_SIG	-102039	ルーム間マイク接続の署名解析エラー
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SIG_TIME	-102040	ルーム間マイク接続の署名タイムスタンプエラー
ERR_SERVER_CENTER_CONN_ROOM_SIG_GROUPID	-102041	ルーム間マイク接続の署名ルーム番号が一致しません

記号	値	意味
ERR_SERVER_CENTER_CONN_ROOM_NOT_CONNED	-102042	ルーム間マイク接続の署名ユーザー名が一致しません
ERR_SERVER_CENTER_CONN_ROOM_USER_NOT_CONNED	-102043	このユーザーはマイク接続を開始していません
ERR_SERVER_CENTER_CONN_ROOM_FAILED	-102044	ルーム間マイク接続に失敗しました
ERR_SERVER_CENTER_CONN_ROOM_CANCEL_FAILED	-102045	ルーム間マイク接続の取り消しに失敗しました
ERR_SERVER_CENTER_CONN_ROOM_CONNED_ROOM_NOT_EXIST	-102046	マイク接続されたルームが存在しません
ERR_SERVER_CENTER_CONN_ROOM_CONNED_REACH_MAX_ROOM	-102047	マイク接続されたルームがマイク接続の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_NOT_EXIST	-102048	マイク接続されたユーザーが存在しません
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_DELETED	-102049	マイク接続されたユーザーが削除されました
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_FULL	-102050	マイク接続されたユーザーがリソースの上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SEQ	-102051	マイク接続リクエストの番号が乱れています

アラートコード表

アラートコードを特に注意する必要はありません。必要に応じて、現在のユーザーに対応するプロンプトを表示するかどうかを選択できます。

記号	値	意味
WARNING_HW_ENCODER_START_FAIL	1103	ハードウェアエンコーディングの起動に問題が発生した場合、自動的にソフトウェアエンコーディングに切り替わります
WARNING_VIDEO_ENCODER_SW_TO_HW	1107	現在のCPU使用率が高すぎてソフトウェアエンコーディング要件を満たせないため、自動的にハードウェアエンコーディングに切り替わります
WARNING_INSUFFICIENT_CAPTURE_FPS	1108	カメラのキャプチャフレームレートが不足しています。美顔アルゴリズムを備えた一部のAndroidスマートフォンには表示されます
WARNING_SW_ENCODER_START_FAIL	1109	ソフトウェアエンコーディングの開始に失敗しました
WARNING_REDUCE_CAPTURE_RESOLUTION	1110	現在のフレームレートとパフォーマンスの最適解を満たすため、カメラのキャプチャ解像度が低下します。
WARNING_CAMERA_DEVICE_EMPTY	1111	使用可能なカメラデバイスが検出されません
WARNING_CAMERA_NOT_AUTHORIZED	1112	ユーザーは、現在のアプリケーションにカメラ使用権限を承認していません
WARNING_MICROPHONE_DEVICE_EMPTY	1201	使用可能なマイクデバイスが検出されません
WARNING_SPEAKER_DEVICE_EMPTY	1202	使用可能なスピーカーデバイスが検出されません
WARNING_MICROPHONE_NOT_AUTHORIZED	1203	ユーザーは、現在のアプリケーションにマイク使用権限を承認していません

記号	値	意味
WARNING_MICROPHONE_DEVICE_ABNORMAL	1204	オーディオキャプチャデバイスが利用できません（使用中であるなど）
WARNING_SPEAKER_DEVICE_ABNORMAL	1205	オーディオ再生デバイスが利用できません（使用中であるなど）
WARNING_VIDEO_FRAME_DECODE_FAIL	2101	現在のビデオフレームのデコードに失敗しました
WARNING_AUDIO_FRAME_DECODE_FAIL	2102	現在のオーディオフレームのデコードに失敗しました
WARNING_VIDEO_PLAY_LAG	2105	現在、ビデオ再生時にラグが発生しています
WARNING_HW_DECODER_START_FAIL	2106	ハードウェアデコードの開始に失敗しました。ソフトウェアデコードを使用してください
WARNING_VIDEO_DECODER_HW_TO_SW	2108	現在のストリームの最初のIフレームのハードウェアデコードが失敗しました。SDKは自動的にソフトウェアデコードに切り替えます
WARNING_SW_DECODER_START_FAIL	2109	ソフトウェアデコーダの起動に失敗しました
WARNING_VIDEO_RENDER_FAIL	2110	ビデオレンダリングに失敗しました
WARNING_START_CAPTURE_IGNORED	4000	キャプチャ中のため、キャプチャの開始は無視されます
WARNING_AUDIO_RECORDING_WRITE_FAIL	7001	オーディオ記録のファイルへの書き込みに失敗しました
WARNING_ROOM_DISCONNECT	5101	ネットワーク接続が切断されました
WARNING_IGNORE_UPSTREAM_FOR_AUDIENCE	6001	現在、視聴者ロールです。アップストリームのオーディオビデオデータを無視します
WARNING_NET_BUSY	1101	ネットワーク状態の不良：アップストリーム帯域幅が小さすぎて、データのアップロード時に問題が生じます

記号	値	意味
WARNING_RTMP_SERVER_RECONNECT	1102	CSSプッシュのときにネットワークが切断しましたが、自動再接続が開始されました（自動再接続は連続3回を超えて失敗すると、それ以上は行われません）
WARNING_LIVE_STREAM_SERVER_RECONNECT	2103	ライブストリーミングのプルのときにネットワークが切断しましたが、自動再接続が開始されました（自動再接続は連続3回を超えて失敗すると、それ以上は行われません）
WARNING_RECV_DATA_LAG	2104	不安定なネットワークパケット：ダウンストリーム帯域幅が不足しているか、キャスター側からのストリーミングが不均一であることが原因と思われます
WARNING_RTMP_DNS_FAIL	3001	ライブストリーミング、DNS解決に失敗しました
WARNING_RTMP_SEVER_CONN_FAIL	3002	ライブストリーミング、サーバーの接続に失敗しました
WARNING_RTMP_SHAKE_FAIL	3003	ライブストリーミング、RTMPサーバーとのハンドシェイクに失敗しました
WARNING_RTMP_SERVER_BREAK_CONNECT	3004	ライブストリーミング、サーバーが自動的に切断しました
WARNING_RTMP_READ_WRITE_FAIL	3005	ライブストリーミング、RTMPの読み取り/書き込みに失敗し、接続が切断されます
WARNING_RTMP_WRITE_FAIL	3006	ライブストリーミング、RTMP書き込みエラー（SDK内部エラーコードは外部にスローされません）
WARNING_RTMP_READ_FAIL	3007	ライブストリーミング、RTMP読み取りエラー（SDK内部エラーコードは外部にスローされません）

記号	値	意味
WARNING_RTMP_NO_DATA	3008	ライブストリーミング、30秒以上データが送信されない場合、自動的に接続が切断されます
WARNING_PLAY_LIVE_STREAM_INFO_CONNECT_FAIL	3009	ライブストリーミング、connectサーバー呼び出しに失敗しました（SDK内部エラーコードは外部にスローされません）
WARNING_NO_STEAM_SOURCE_FAIL	3010	ライブストリーミング、接続に失敗しました。このストリームアドレスにビデオはありません。（SDK内部エラーコードは外部にスローされません）
WARNING_ROOM_RECONNECT	5102	ネットワークが切断され、自動再接続が開始されました
WARNING_ROOM_NET_BUSY	5103	ネットワーク状態の不良：アップストリーム帯域幅が小さすぎて、データのアップロード時に問題が生じます

Unity
Overview

最終更新日：：2023-10-09 11:55:23

Overview

Basic APIs

API	Description
getTRTCShareInstance	Creates a `TRTCCloud` singleton.
destroyTRTCShareInstance	Releases a `TRTCCloud` singleton.
addCallback	Sets the callback API `TRTCCloudCallback` .
removeCallback	Removes event callback.

Room APIs

API	Description
enterRoom	Enters a room. If the room does not exist, the system will create one automatically.
exitRoom	Exits a room.
switchRole	Switches roles. This API works only in live streaming scenarios (`TRTC_APP_SCENE_LIVE` and `TRTC_APP_SCENE_VOICE_CHATROOM`).
setDefaultStreamRecvMode	Sets the audio/video data receiving mode, which must be set before room entry to take effect.
connectOtherRoom	Requests a cross-room call (anchor competition).
disconnectOtherRoom	Exits a cross-room call.
switchRoom	Switches rooms.

CDN APIs

API	Description

startPublishing	Starts pushing to Tencent Cloud's live streaming CDN.
stopPublishing	Stops pushing to Tencent Cloud's live streaming CDN.
startPublishCDNStream	Starts relaying to the live streaming CDN of a non-Tencent Cloud vendor.
stopPublishCDNStream	Stops relaying to non-Tencent Cloud addresses.
setMixTranscodingConfig	Sets On-Cloud MixTranscoding parameters.

Video APIs

API	Description
startLocalPreview	Enables local video preview (only custom rendering is supported currently).
stopLocalPreview	Stops local video capturing and preview.
muteLocalVideo	Pauses/Resumes sending local video data.
startRemoteView	Starts pulling and displaying the image of a specified remote user (only custom rendering is supported currently).
stopRemoteView	Stops displaying and pulling the video image of a remote user.
stopAllRemoteView	Stops displaying and pulling the video images of all remote users.
muteRemoteVideoStream	Pauses/Resumes receiving the video stream of a specified remote user.
muteAllRemoteVideoStreams	Pauses/Resumes receiving all remote video streams.
setVideoEncoderParam	Sets video encoder parameters.
setNetworkQosParam	Sets QoS control parameters.
setVideoEncoderMirror	Sets the mirror mode of encoded images.

Audio APIs

API	Description
startLocalAudio	Enables local audio capturing and upstream data transfer.
stopLocalAudio	Disables local audio capturing and upstream data transfer.
muteLocalAudio	Mutes/Unmutes local audio.
muteRemoteAudio	Mutes/Unmutes a specified remote user.

muteAllRemoteAudio	Mutes/Unmutes all remote users.
setRemoteAudioVolume	Sets the playback volume of a remote user.
setAudioCaptureVolume	Sets the SDK capturing volume.
getAudioCaptureVolume	Gets the SDK capturing volume.
setAudioPlayoutVolume	Sets the SDK playback volume.
getAudioPlayoutVolume	Gets the SDK playback volume.
enableAudioVolumeEvaluation	Enables volume reminders.
startAudioRecording	Starts audio recording.
stopAudioRecording	Stops audio recording.

Device management APIs

API	Description
getDeviceManager	Gets the device management module. For details, please see Specific Device Management APIs.

Music and voice effect APIs

API	Description
getAudioEffectManager	Gets the audio effect management class `TXAudioEffectManager` , which is used to manage background music, short audio effects, and voice effects. For details, please see Specific Music and Voice Effect APIs.

Custom video rendering APIs

API	Description
setLocalVideoRenderCallback	Sets custom rendering for the local video.
setRemoteVideoRenderCallback	Sets custom rendering for the video of a remote user.

Custom message sending APIs

API	Description

sendSEIMsg	Embeds small-volume custom data in video frames.

Network testing APIs

API	Description
startSpeedTest	Starts network speed testing. This may compromise the quality of video calls and should be avoided during a video call.
stopSpeedTest	Stops server speed testing.

Log APIs

API	Description
getSDKVersion	Gets the SDK version.
setLogLevel	Sets the log output level.
setLogDirPath	Changes the path to save logs.
setLogCompressEnabled	Enables/Disables local log compression.
callExperimentalAPI	Calls the experimental API.

TRTCCloudCallback

Callback APIs for the TRTC audio call feature

Error and warning event callback APIs

API	Description
onError	Error callback. This indicates that the SDK encountered an irrecoverable error. Such errors must be listened for, and UI reminders should be displayed to users if necessary.
onWarning	Warning callback. This alerts you to non-serious problems such as lag or recoverable decoding failure.

Room event callback APIs

API	Description
onEnterRoom	Callback of room entry

onExitRoom	Callback of room exit
onSwitchRole	Callback of role switching
onConnectOtherRoom	Callback of the result of requesting a cross-room call (anchor competition)
onDisConnectOtherRoom	Callback of the result of ending a cross-room call (anchor competition)
onSwitchRoom	Callback of the result of room switching (`switchRoom`)

User event callback APIs

API	Description
onRemoteUserEnterRoom	Callback of the entry of a user
onRemoteUserLeaveRoom	Callback of the exit of a user
onUserVideoAvailable	Callback of whether a user has turned the camera on
onUserAudioAvailable	Callback of whether a remote user has playable audio
onFirstVideoFrame	Callback of rendering the first video frame of the local user or a remote user
onFirstAudioFrame	Callback of playing the first audio frame of a remote user. No notifications are sent for local audio.
onSendFirstLocalVideoFrame	Callback of sending the first local video frame
onSendFirstLocalAudioFrame	Callback of sending the first local audio frame

Callback APIs for statistics on network quality and technical metrics

API	Description
onNetworkQuality	Callback of network quality. This callback is triggered every 2 seconds to collect statistics on the current upstream and downstream data transfer.
onStatistics	Callback of statistics on technical metrics

Server event callback APIs

API	Description
onConnectionLost	Callback of the disconnection of the SDK from the server

onTryToReconnect	Callback of the SDK trying to connect to the server again
onConnectionRecovery	Callback of the reconnection of the SDK to the server
onSpeedTest	Callback of server speed test results. The SDK tests the speed of multiple server addresses, and the result of each test is returned through this callback.

Hardware event callback APIs

API	Description
onCameraDidReady	Callback of the camera being ready
onMicDidReady	Callback of the mic being ready
onUserVoiceVolume	Callback of volume, including the volume of each `userId` and the total remote volume
onDeviceChange	Callback of connecting/disconnecting a local device

Custom message receiving callback APIs

API	Description
onRecvSEIMsg	Callback of receiving an SEI message

Callback APIs for CDN relayed push

API	Description
onStartPublishing	Callback of starting to push to Tencent Cloud's live streaming CDN, which corresponds to the `startPublishing()` API in TRTCCloud
onStopPublishing	Callback of stopping pushing to Tencent Cloud's live streaming CDN, which corresponds to the `stopPublishing()` API in TRTCCloud
onStartPublishCDNStream	Callback of the completion of starting relayed push to CDNs
onStopPublishCDNStream	Callback of the completion of stopping relayed push to CDNs
onSetMixTranscodingConfig	Sets On-Cloud MixTranscoding parameters, which corresponds to the `setMixTranscodingConfig()` API in `TRTCCloud`

Definitions of Key Classes

Class	Description
TRTCParams	Room entry parameters
TRTCVideoEncParam	Video encoding parameters
TRTCTranscodingConfig	On-Cloud MixTranscoding configuration
TRTCSwitchRoomConfig	Room switching parameters
TRTCNetworkQosParam	QoS control parameters
TXVoiceReverbType	Reverb effects (karaoke, room, hall, low and deep, resonant, etc.)
AudioMusicParam	Parameters for music and voice effect setting APIs
TRTCAudioRecordingParams	Audio recording parameters

Specific Device Management APIs

API	Description
isFrontCamera	Gets whether the front camera is being used.
switchCamera	Switches cameras.
getCameraZoomMaxRatio	Gets the maximum zoom level of the current camera.
setCameraZoomRatio	Sets the zoom level of the current camera.
isAutoFocusEnabled	Gets whether automatic facial recognition is supported.
enableCameraAutoFocus	Enables/Disables automatic facial recognition.
setCameraFocusPosition	Sets camera focus.

enableCameraTorch	Enables/Disables flash.
setSystemVolumeType	Sets the system volume type to use during calls.
setAudioRoute	Sets the audio route.

Specific Music and Voice Effect APIs

API	Description
setVoiceReverbType	Sets the voice change effects (karaoke, room, hall, low and deep, resonant, etc.)
setMusicObserver	Sets the callback of the playback progress of background music.
startPlayMusic	Starts playing background music.
stopPlayMusic	Stops playing background music.
pausePlayMusic	Pauses background music.
resumePlayMusic	Resumes playing background music.
setMusicPublishVolume	Sets the remote playback volume of background music, i.e., the volume heard by remote users.
setMusicPlayoutVolume	Sets the local playback volume of background music.
setAllMusicVolume	Sets the local and remote playback volume of background music.
setMusicPitch	Changes the pitch of background music.
setMusicSpeedRate	Changes the playback speed of background music.
getMusicCurrentPosInMS	Gets the playback progress (ms) of background music.
seekMusicToPosInMS	Sets the playback progress (ms) of background music.
getMusicDurationInMS	Gets the length (ms) of the background music file.

エラーコード

最終更新日：：2022-06-02 15:53:19

エラーコードリスト

基本的なエラーコード

記号	値	意味
ERR_NULL	0	エラーなし

入室関連のエラーコード

TRTCCloud.enterRoom() 入室に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.onEnterRoom()、TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_ENTER_FAIL	-3301	入室の失敗
ERR_ENTER_ROOM_PARAM_NULL	-3316	入室パラメータが空です。TRTCCloud.enterRoom():インターフェースの呼び出しが有効なparamで渡されるかどうか確認してください
ERR_SDK_APPID_INVALID	-3317	入室パラメータsdkAppIdエラー
ERR_ROOM_ID_INVALID	-3318	入室パラメータroomIdエラー
ERR_USER_ID_INVALID	-3319	入室パラメータuserIDが正しくありません
ERR_USER_SIG_INVALID	-3320	入室パラメータuserSigが正しくありません
ERR_ROOM_REQUEST_ENTER_ROOM_TIMEOUT	-3308	入室リクエストがタイムアウトしました。ネットワークをご確認ください

記号	値	意味
ERR_SERVER_INFO_SERVICE_SUSPENDED	-100013	サービスをご利用いただけません。パッケージの残りの時間（分）が0より大きいかどうか、Tencent Cloudアカウントの支払いが遅延していないかどうかご確認ください

退室関連のエラーコード

TRTCCloud.exitRoom() 退室に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_REQUEST_QUIT_ROOM_TIMEOUT	-3325	退室リクエストのタイムアウト

デバイス（カメラ、マイク、スピーカー）関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_CAMERA_START_FAIL	-1301	カメラの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、カメラのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください
ERR_CAMERA_NOT_AUTHORIZED	-1314	カメラデバイスの権限が承認されていません。通常はモバイルデバイスに表示されます。ユーザーから権限承認を拒否されている可能性があります
ERR_CAMERA_SET_PARAM_FAIL	-1315	カメラパラメータ設定エラー（パラメータがサポートされていないか、その他）
ERR_CAMERA_OCCUPY	-1316	カメラが使用中なので、他のカメラの起動を試みることができます
ERR_MIC_START_FAIL	-1302	マイクの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、マイクのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください

記号	値	意味
ERR_MIC_NOT_AUTHORIZED	-1317	マイクデバイスの権限が承認されていません。通常はモバイルデバイスに表示されます。ユーザーから権限承認を拒否されている可能性があります
ERR_MIC_SET_PARAM_FAIL	-1318	マイクパラメータの設定に失敗しました
ERR_MIC_OCCUPY	-1319	マイクが使用中です。例えば、モバイルデバイスが通話中の場合、マイクの起動は失敗します
ERR_MIC_STOP_FAIL	-1320	マイクの停止に失敗しました
ERR_SPEAKER_START_FAIL	-1321	スピーカーの起動に失敗しました。例えば、WindowsまたはMacデバイスの場合、スピーカーのコンフィギュレーター（ドライバー）に異常があります。デバイスを無効にしてから再度有効にするか、マシンを再起動するか、またはコンフィギュレーターを更新してください
ERR_SPEAKER_SET_PARAM_FAIL	-1322	スピーカーパラメータの設定に失敗しました
ERR_SPEAKER_STOP_FAIL	-1323	スピーカーの停止に失敗しました

画面共有関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_SCREEN_CAPTURE_START_FAIL	-1308	スクリーンキャプチャの開始に失敗しました。モバイルデバイスに表示されるときは、ユーザーから権限承認を拒否されている可能性があります。WindowsまたはMacシステムのデバイスに表示される場合は、スクリーンキャプチャインターフェースのパラメータが要件を満たしているかご確認ください

記号	値	意味
ERR_SCREEN_CAPTURE_UNSURPORT	-1309	スクリーンキャプチャに失敗しました。Androidプラットフォームでは5.0以降のシステム、iOSプラットフォームでは11.0以降のシステムが必要です
ERR_SERVER_CENTER_NO_PRIVILEDGE_PUSH_SUB_VIDEO	-102015	サブチャネルのアップストリームの権限がありません
ERR_SERVER_CENTER_ANOTHER_USER_PUSH_SUB_VIDEO	-102016	他のユーザーがサブチャネルのアップストリームを行っています
ERR_SCREEN_CAPTURE_STOPPED	-7001	スクリーンキャプチャがシステムによって停止されました

コーデック関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_VIDEO_ENCODE_FAIL	-1303	ビデオフレームのエンコードに失敗しました。例えば、iOSデバイスを他のアプリケーションに切り替えると、システムによってハードウェアエンコーダが解放される場合があります。切り替えて元に戻すと、ハードウェアエンコーダが再起動する前にスローされる場合があります
ERR_UNSUPPORTED_RESOLUTION	-1305	サポートされていないビデオ解像度
ERR_AUDIO_ENCODE_FAIL	-1304	オーディオフレームのエンコードに失敗しました。例えば、渡されたカスタムオーディオデータを、SDKで処理することはできません
ERR_UNSUPPORTED_SAMPLERATE	-1306	サポートされていないオーディオサンプリングレート

カスタマイズキャプチャ関連のエラーコード

コールバック関数TRTCCloudDelegate.OnError()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_PIXEL_FORMAT_UNSUPPORTED	-1327	設定されたpixel formatはサポートされていません
ERR_BUFFER_TYPE_UNSUPPORTED	-1328	設定されたbuffer typeはサポートされていません

CDNバインディングおよびミクスストリーミング関連のエラーコード

コールバック関数TRTCCloudDelegate.onStartPublishing()、TRTCCloudDelegate.onSetMixTranscodingConfig()によって関連通知をキャプチャできます。

記号	値	意味
ERR_PUBLISH_CDN_STREAM_REQUEST_TIME_OUT	-3321	バイパス転送リクエストのタイムアウト
ERR_CLOUD_MIX_TRANSCODING_REQUEST_TIME_OUT	-3322	クラウドミクスストリーミングリクエストのタイムアウト
ERR_PUBLISH_CDN_STREAM_SERVER_FAILED	-3323	バイパス転送リターンパケットの異常
ERR_CLOUD_MIX_TRANSCODING_SERVER_FAILED	-3324	クラウドミクスストリーミング・リターンパケットの異常
ERR_ROOM_REQUEST_START_PUBLISHING_TIMEOUT	-3333	Tencent CloudへのライブCDNのプッシュ開始シグナリングのタイムアウト
ERR_ROOM_REQUEST_START_PUBLISHING_ERROR	-3334	Tencent CloudへのライブCDNのプッシュ開始シグナリングの異常
ERR_ROOM_REQUEST_STOP_PUBLISHING_TIMEOUT	-3335	Tencent CloudへのライブCDNのプッシュ停止シグナリングのタイムアウト
ERR_ROOM_REQUEST_STOP_PUBLISHING_ERROR	-3336	Tencent CloudへのライブCDNのプッシュ停止シグナリングの異常

ルーム間マイク接続関連のエラーコード

TRTCCloud.ConnectOtherRoom() ルーム間マイク接続に失敗したときにこのタイプのエラーコードがトリガーされます。コールバック関数TRTCCloudDelegate.onConnectOtherRoom()によって、関連通知をキャプチャできます。

記号	値	意味
ERR_ROOM_REQUEST_CONN_ROOM_TIMEOUT	-3326	マイク接続リクエストのタイムアウト
ERR_ROOM_REQUEST_DISCONN_ROOM_TIMEOUT	-3327	マイク接続からの退出リクエストのタイムアウト
ERR_ROOM_REQUEST_CONN_ROOM_INVALID_PARAM	-3328	無効なパラメータ
ERR_CONNECT_OTHER_ROOM_AS_AUDIENCE	-3330	現在のロールは視聴者であり、ルーム間マイク接続をリクエストまたは切断することはできません。まずキャスターにswitchRole()する必要があります
ERR_SERVER_CENTER_CONN_ROOM_NOT_SUPPORT	-102031	ルーム間マイク接続はサポートされていません
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_NUM	-102032	ルーム間マイク接続の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_RETRY_TIMES	-102033	ルーム間マイク接続の再試行回数の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_REQ_TIMEOUT	-102034	ルーム間マイク接続リクエストのタイムアウト
ERR_SERVER_CENTER_CONN_ROOM_REQ	-102035	ルーム間マイク接続リクエストの形式エラー

記号	値	意味
ERR_SERVER_CENTER_CONN_ROOM_NO_SIG	-102036	ルーム間マイク接続に署名がありません
ERR_SERVER_CENTER_CONN_ROOM_DECRYPT_SIG	-102037	ルーム間マイク接続の署名復号に失敗しました
ERR_SERVER_CENTER_CONN_ROOM_NO_KEY	-102038	ルーム間マイク接続の署名復号キーが見つかりませんでした
ERR_SERVER_CENTER_CONN_ROOM_PARSE_SIG	-102039	ルーム間マイク接続の署名解析エラー
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SIG_TIME	-102040	ルーム間マイク接続の署名タイムスタンプエラー
ERR_SERVER_CENTER_CONN_ROOM_SIG_GROUPID	-102041	ルーム間マイク接続の署名のルーム番号が一致しません
ERR_SERVER_CENTER_CONN_ROOM_NOT_CONNED	-102042	ルーム間マイク接続の署名のユーザー名が一致しません
ERR_SERVER_CENTER_CONN_ROOM_USER_NOT_CONNED	-102043	このユーザーはマイク接続を開始していません
ERR_SERVER_CENTER_CONN_ROOM_FAILED	-102044	ルーム間マイク接続に失敗しました
ERR_SERVER_CENTER_CONN_ROOM_CANCEL_FAILED	-102045	ルーム間マイク接続の取り消しに失敗しました

記号	値	意味
ERR_SERVER_CENTER_CONN_ROOM_CONNED_ROOM_NOT_EXIST	-102046	マイク接続されたルームが存在しません
ERR_SERVER_CENTER_CONN_ROOM_CONNED_REACH_MAX_ROOM	-102047	マイク接続されたルームがマイク接続の上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_NOT_EXIST	-102048	マイク接続されたユーザーが存在しません
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_DELETED	-102049	マイク接続されたユーザーが削除されました
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_FULL	-102050	マイク接続されたユーザーがリソースの上限に達しました
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SEQ	-102051	マイク接続リクエストの番号が乱れています

アラートコードリスト

アラートコードを特に注意する必要はありません。必要に応じて、現在のユーザーに対応するプロンプトを表示するかどうかを選択できます。

記号	値	意味
WARNING_HW_ENCODER_START_FAIL	1103	ハードウェアエンコーディングの起動に問題が発生した場合、自動的にソフトウェアエンコーディングに切り替わります
WARNING_VIDEO_ENCODER_SW_TO_HW	1107	現在のCPU使用率が高すぎてソフトウェアエンコーディング要件を満たせないため、自動的にハードウェアエンコーディングに切り替わります

記号	値	意味
WARNING_INSUFFICIENT_CAPTURE_FPS	1108	カメラのキャプチャフレームレートが不足しています。美顔アルゴリズムを備えた一部のAndroidスマートフォンには表示されます
WARNING_SW_ENCODER_START_FAIL	1109	ソフトウェアエンコーディングの開始に失敗しました
WARNING_REDUCE_CAPTURE_RESOLUTION	1110	現在のフレームレートとパフォーマンスの最適解を満たすため、カメラのキャプチャ解像度が低下します。
WARNING_CAMERA_DEVICE_EMPTY	1111	使用可能なカメラデバイスが検出されません
WARNING_CAMERA_NOT_AUTHORIZED	1112	ユーザーは、現在のアプリケーションにカメラ使用権限を承認していません
WARNING_MICROPHONE_DEVICE_EMPTY	1201	使用可能なマイクデバイスが検出されません
WARNING_SPEAKER_DEVICE_EMPTY	1202	使用可能なスピーカーデバイスが検出されません
WARNING_MICROPHONE_NOT_AUTHORIZED	1203	ユーザーは、現在のアプリケーションにマイク使用権限を承認していません
WARNING_MICROPHONE_DEVICE_ABNORMAL	1204	オーディオキャプチャデバイスが利用できません（使用中であるなど）
WARNING_SPEAKER_DEVICE_ABNORMAL	1205	オーディオ再生デバイスが利用できません（使用中であるなど）
WARNING_VIDEO_FRAME_DECODE_FAIL	2101	現在のビデオフレームのデコードに失敗しました
WARNING_AUDIO_FRAME_DECODE_FAIL	2102	現在のオーディオフレームのデコードに失敗しました
WARNING_VIDEO_PLAY_LAG	2105	現在、ビデオ再生時にラグが発生しています

記号	値	意味
WARNING_HW_DECODER_START_FAIL	2106	ハードウェアデコードの起動に失敗しました。ソフトウェアデコードを使用してください
WARNING_VIDEO_DECODER_HW_TO_SW	2108	現在のストリームの最初のIフレームのハードウェアデコードに失敗しました。SDKは自動的にソフトウェアデコードに切り替えます
WARNING_SW_DECODER_START_FAIL	2109	ソフトウェアデコーダの起動に失敗しました
WARNING_VIDEO_RENDER_FAIL	2110	ビデオレンダリングに失敗しました
WARNING_START_CAPTURE_IGNORED	4000	キャプチャ中のため、キャプチャの開始は無視されます
WARNING_AUDIO_RECORDING_WRITE_FAIL	7001	オーディオキャプチャのファイルへの書き込みに失敗しました
WARNING_ROOM_DISCONNECT	5101	ネットワーク接続が切断されました
WARNING_IGNORE_UPSTREAM_FOR_AUDIENCE	6001	現在、視聴者ロールです。アップストリームのオーディオビデオデータを無視します
WARNING_NET_BUSY	1101	ネットワーク状態の不良：アップストリーム帯域幅が小さすぎて、データのアップロード時に問題が生じます
WARNING_RTMP_SERVER_RECONNECT	1102	ライブストリーミングのプッシュのときにネットワークが切断しましたが、自動再接続が開始されました（自動再接続は連続3回を超えて失敗すると、それ以上は行われません）
WARNING_LIVE_STREAM_SERVER_RECONNECT	2103	ライブストリーミングのプルのときにネットワークが切断しましたが、自動再接続が開始されました（自動再接続は連続3回を超えて失敗すると、それ以上は行われません）

記号	値	意味
WARNING_RECV_DATA_LAG	2104	不安定なネットワークパケット：ダウンストリーム帯域幅が不足しているか、キャスター側からのストリーミングが不均一であることが原因と思われます
WARNING_RTMP_DNS_FAIL	3001	ライブストリーミング、DNS解決に失敗しました
WARNING_RTMP_SEVER_CONN_FAIL	3002	ライブストリーミング、サーバーの接続に失敗しました
WARNING_RTMP_SHAKE_FAIL	3003	ライブストリーミング、RTMPサーバーとのハンドシェイクに失敗しました
WARNING_RTMP_SERVER_BREAK_CONNECT	3004	ライブストリーミング、サーバーが自動的に切断しました
WARNING_RTMP_READ_WRITE_FAIL	3005	ライブストリーミング、RTMPの読み取り/書き込みに失敗し、接続が切断されます
WARNING_RTMP_WRITE_FAIL	3006	ライブストリーミング、RTMP書き込みに失敗しました（SDK内部エラーコードは外部にスローされません）
WARNING_RTMP_READ_FAIL	3007	ライブストリーミング、RTMP読み取りに失敗しました（SDK内部エラーコードは外部にスローされません）
WARNING_RTMP_NO_DATA	3008	ライブストリーミング、30秒以上データが送信されない場合、自動的に接続が切断されます
WARNING_PLAY_LIVE_STREAM_INFO_CONNECT_FAIL	3009	ライブストリーミング、connectサーバー呼び出しに失敗しました（SDK内部エラーコードは外部にスローされません）

記号	値	意味
WARNING_NO_STEAM_SOURCE_FAIL	3010	ライブストリーミング、接続に失敗しました。このストリームアドレスにビデオはありません。（SDK内部エラーコードは外部にスローされません）
WARNING_ROOM_RECONNECT	5102	ネットワークが切断され、自動再接続が開始されました
WARNING_ROOM_NET_BUSY	5103	ネットワーク状態の不良：アップストリーム帯域幅が小さすぎて、データのアップロード時に問題が生じます