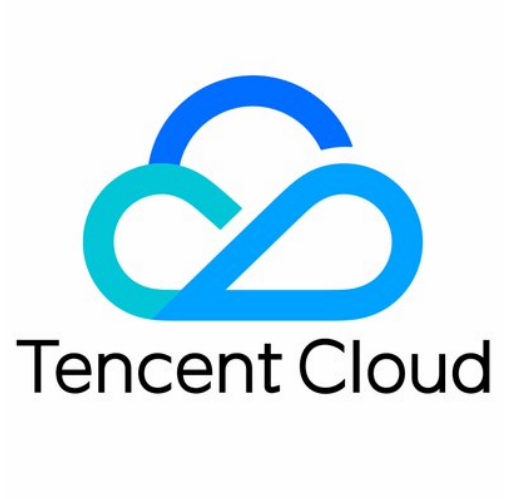
Video on Demand

開発ガイド

製品ドキュメント



カタログ：

開発ガイド
メディアアップロード
メディアアップロードの概要

最終更新日：：2023-10-26 17:31:32

メディアアップロードとは、ビデオ、オーディオ、カバー画像などのメディアファイルをVODのストレージにアップロードし、後続の処理と配信などを実行することをいいます。

アップロード方法

VODでは次のいくつかのアップロード方法をサポートしています。

- コンソールからのローカルアップロード
  VODコンソールのアップロードページで操作して、ローカルメディアファイルをVODにアップロードできます。少量のメディアを直接管理するケースに適しており、簡単かつ迅速、さらに技術的ハードルがないというメリットがあります。
- コンソールからのプルアップロード
  VODコンソールのアップロードページで操作して、アップロードするメディアのURLを指定すると、VODバックエンドがファイルをオフラインでプルします。
- サーバーからのアップロード
  バックエンドサーバーにストレージされたメディアファイルをVODにアップロードできます。自動化、システム化された本番環境に適しています。VODでは次のプログラム言語によるサーバーからのアップロードSDKからのアップロードを提供します。
  - Java SDK
  - C# SDK
  - PHP SDK
  - Python SDK
  - Node.js SDK
  - Golang SDK
- クライアントからのアップロード
  エンドユーザーはクライアントのローカルビデオをVODにアップロードできます。UGC、PGCなどのケースに適しています。VODは次のプラットフォームのクライアントからのアップロードSDKからのアップロードを提供します。
  - AndroidのアップロードSDK
  - iOSのアップロードSDK

- Web端末のアップロードSDK
- APIからのプルアップロード
  VOD が提供するサーバーAPIからのプルアップロードインタフェースを使用するとき、アップロードするメディアのURLを指定すると、VODバックエンドがファイルをオフラインでプルします。大量のまたは自動化されたメディアファイルを移行するケースに適しています。
- CSSレコーディング
  CSSが提供するレコーディング機能を介して、CSSストリームのビデオ内容をVODにストレージし、アーカイブ、トリミングおよびレビューなどを実行します。

ストレージリージョン

サポートされているリージョン

VODでは世界中の複数のリージョンにストレージノードがあります。メディアをアップロードするプロセスではその内の1つのリージョンを選択してストレージします。現在、VODがサポートするストレージリージョンは次のとおりです。

ストレージリージョン	リージョンの英語の略称
北京	ap-beijing
上海	ap-shanghai
広州	ap-guangzhou
重慶	ap-chongqing
天津	ap-beijing-1
南京	ap-nanjing
成都	ap-chengdu
中国香港	ap-hongkong
中国台北	ap-taipei
シンガポール	ap-singapore
インドムンバイ	ap-mumbai
インドネシアジャカルタ	ap-jakarta
韓国ソウル	ap-seoul

ストレージリージョン	リージョンの英語の略称
タイバンコク	ap-bangkok
日本東京	ap-tokyo
アメリカシリコンバレー（アメリカ西部）	na-siliconvalley
アメリカバージニア（アメリカ東部）	na-ashburn
ブラジルサンパウロ	sa-saopaulo
カナダトロント	na-toronto
ドイツフランクフルト	eu-frankfurt
ロシアモスクワ	eu-moscow

ストレージリージョンのアクティブ化

複数のストレージリージョンを構成する重要な目的はメディアアップロードの品質（成功率と速度）を向上させることです。アップロードの実行者とストレージノードの距離はアップロード品質に影響を及ぼし、通常、遠距離よりも近距離の方が高いアップロード品質を得られます。

開発者がVODサービスをアクティブ化すると、VODは**シンガポール**ストレージリージョンを自動的に割り当てます。開発者は業務のニーズに応じてその他のストレージリージョンをアクティブ化することが可能です。具体的な操作については、アップロードストレージ設定をご参照ください。**ストレージリージョンは一度アクティブ化すると無効にできなくなります**。

デフォルトのストレージリージョン

開発者の既存のストレージリージョンの中には、デフォルトのストレージリージョンがあり、かつそれは1つのみとなります。ストレージリージョンが1つのみ（シンガポール）のときは、これがデフォルトのストレージリージョンとなり、複数のストレージリージョンをアクティブ化している場合は、コンソールでその他リージョンをデフォルトのストレージリージョンに選択することができます。具体的な操作については、ストレージリージョン設定をご参照ください。

デフォルトのストレージリージョンの目的：一部のケースでは、このリージョンがメディアアップロードの対象リージョンとして優先的に選択されます。具体的な説明については、次のテキストをお読みください。

ストレージリージョンの選択

メディアのアップロードにはストレージリージョンを選択する必要があります。デフォルトでVODバックエンドによって自動的に選択されることも、またアップロードリクエストで指定することもできます。

- VODバックグラウンドがストレージリージョンを自動的に選択する場合：

- 開発者のストレージリージョンが1つのみ（シンガポール）の場合は、アップロードしたすべてのメディアはこのリージョンに保存されます。
- 開発者が複数のストレージリージョンをアクティブ化している場合、各種アップロード方法の選択ポリシーは次のとおりです。

アップロード方法	リージョン選択ポリシー
コンソールからのローカルアップロード	アップロードする者の地理的位置に基づき、最寄りのストレージリージョンが選択されます
コンソールからのプルアップロード	デフォルトのストレージリージョンが常に選択されます
サーバーからのアップロード	アップロードする者の地理的位置に基づき、最寄りのストレージリージョンが選択されます
クライアントからのアップロード	アップロードする者の地理的位置に基づき、最寄りのストレージリージョンが選択されます
APIプルアップロード	デフォルトのストレージリージョンが常に選択されます
CSSレコーディング	CSSストリームの所在リージョンに基づき、最寄りのストレージリージョンが選択されます

- 開発者がストレージリージョンを指定する場合の各種アップロード方法の指定方法は次のとおりです。

アップロード方法	リージョン指定方法
コンソールからのローカルアップロード	サポートしていません
コンソールからのプルアップロード	サポートしていません
サーバーからのアップロード	Java SDK C# SDK PHP SDK Python SDK Node.js SDK Go SDK
クライアントからのアップロード	クライアントからのアップロード署名パラメータ
APIプルアップロード	プルアップロードインタフェースStorageRegionパラメータ
CSSレコーディング	サポートしていません

機能と制限

メディアのタイプ

VODは次のタイプのメディアファイルのアップロードをサポートしています。

- ビデオ：WMV、RM、MOV、MPEG、MP4、3GP、FLV、AVI、RMVB、TS、ASF、MPG、WEBM、MKV 、M3U8、WM、ASX、RAM、MPE、VOB、DAT、MP4V、M4V、F4V、MXF、QT、OGG。
- オーディオ：MP3、M4A、FLAC、OGG、WAV、RA、AAC、AMR。
- カバー画像：JPG、JPEG、PNG、GIF、BMP、TIFF、AI、CDR、EPS

イベント通知

メディアアップロードが完了すると、VODバックエンドはこのイベント通知を送信します。イベント通知の原理については、 イベントの通知 を、設定方法については、イベント通知の設定 をそれぞれご参照ください。
各種アップロード方法に対応するイベント通知のタイプは次のとおりです。

アップロード方法	イベント通知のタイプ
• コンソールからのローカルアップロード • サーバーからのアップロード • クライアントからのアップロード • CSSレコーディング	ビデオアップロード完了
• コンソールからのプルアップロード • APIからのプルアップロード	URLからのビデオプルアップロードの完了

付属機能

VODのメディアアップロードは、メディア資産管理関連、ビデオ処理とイベント通知関連、アップロード制御関連などの様々な付属機能をサポートしています。

メディア資産管理関連

- カバーの追加：ビデオをアップロードするときに画像を一緒にアップロードします。この画像はVODメディア資産システムでこのビデオのカバーとして自動的に設定されます。
- 有効期限の指定：アップロード時にメディアファイルの有効期限を指定し、指定時間に達した後、VODバックエンドはメディアファイルおよびそれに関連するファイル（トランスコードファイル、スクリーンキャプチャなど）を自動的に削除します。
- 分類の指定：アップロード後にこのメディアファイルの分類を設定します。

各種アップロード方法のサポート状況と使用法は下表のとおりです。

機能	コンソールからのローカルアップロード	コンソールからのプルアップロード	サーバーからのア
カバーの追加	サポートしていません	サポートしていません	• Java SDK • C# SDK • PHP SDK • Python SDK • Node.js SDK • Go SDK
期限の指定	サポートしていません	サポートしていません	• Java SDKインタスExpireTimeパ • C# SDKインタ-ExpireTimeパラ • PHP SDKインタスExpireTimeパ • Python SDKインフェースExpire メータ • Node.js SDKイフェースExpire メータ • Go SDKインタ-ExpireTimeパラ
分類の指定	分類の指定	サポートしていません	• Java SDKインタスClassIdパラメ • C# SDKインタ-ClassIdパラメー • PHP SDKインタスClassIdパラメ • Python SDKインフェース ClassI タ • Node.js SDKイフェースClassI タ • Go SDKインタ-classIdパラメー

ビデオ処理およびイベント通知関連

- 自動ビデオ処理：メディアのアップロードと同時にタスクフロー を指定すると、アップロードの完了後にVODはこのタスクフローを自動的に実行します。よくあるケース：ビデオのスタートフレームの画像をキャプチャしてカバーにする、トランスコード、コンテンツ審査など。

- ビデオ処理イベント通知のパススルーフィールド：自動ビデオ処理が有効になっている場合は、処理が完了した後、VODバックエンドはイベント通知を開始して、このフィールドをパススルーします。
- アップロードイベント通知のパススルーフィールド：アップロードが完了後、VODバックエンドはイベント通知を開始してこのフィールドをパススルーします。

各種アップロード方法のサポート状況と使用法は下表のとおりです。

機能	コンソールからのローカルアップロード	コンソールからのプルアップロード	サーバーからの
自動ビデオ処理	アップロード後に自動的にビデオ処理を実行	サポートしていません	• Java SDK • C# SDK • PHP SDK • Python SDK • Node.js SDK • Go SDK
ビデオ処理イベント通知のパススルーフィールド	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
アップロードイベント通知のパススルーフィールド	サポートしていません	サポートしていません	• Java SDKイ SourceConte • C# SDKイン SourceConte • PHP SDKイ SourceConte • Python SDKイ スSourceCon タ • Node.js SDK スSourceCon タ • Go SDKイン SourceConte

アップロード制御関連

- 中断からの再開 ：アップロードプロセスが予期せず終了し（ネットワークの中断、ブラウザの終了など）、同一ファイルのアップロードを再度実行する場合に、ファイル全体をアップロードし直す必要がなく、中断した

部分から引き続きアップロードすることができます。
- アップロードの一時停止/再開：アップロード中に自主的にアップロードを停止し、自主的にアップロードを再開することができます。
- アップロードのキャンセル：アップロード中に自主的にそのアップロードを終了することができます。
- アップロードの進行状況の取得：メディアのうちVODにアップロードされた部分の割合を取得することができます。
- マルチパートアップロード：アップロード時にメディアファイルを複数のパートに分割して、それぞれ別々にアップロードします。 弱いネットワーク環境の場合、ネットワーク異常による中断の影響を軽減できます。さらに高帯域幅環境の場合は、複数のパートを同時にアップロードして、ネットワーク帯域幅を充分に利用することができます。

各種アップロード方法のサポート状況と使用法は下表のとおりです。

機能	コンソールからのローカルアップロード	コンソールからのプルアップロード	サーバーからのア
中断からの再開	サポートしていません	関連しません	サポートしていま
アップロードの一時停止と再開	サポートしていません	関連しません	サポートしていま
アップロードのキャンセル	ブラウザページの更新または終了	関連しません	サポートしていま

機能	コンソールからのローカルアップロード	コンソールからのプルアップロード	サーバーからのア
アップロード進行状況の取得	デフォルトで進行状況をページに表示	サポートしていません	サポートしていま
マルチパートアップロード	有効化済み	関連しません	• Java SDK • C# SDK • PHP SDK • Python SDK • Node.js SDK • Go SDK

制限

- メディアファイルサイズの制限は次のとおりです。

アップロード方法	メディアサイズの制限
◦ コンソールからのローカルアップロード ◦ クライアントからのアップロード - Web SDK	60GB
◦ サーバーからのアップロード ◦ コンソールからのプルアップロード ◦ APIからのプルアップロード	48.82TB（50,000GB）
◦ クライアントからのアップロード - Android SDK ◦ クライアントからのアップロード - iOS SDK	10GB
CSSレコーディング	◦ MP4/FLV 形式は48.82TB（50,000GB）です ◦ HLS形式はサイズに制限はありません ◦ その他制限は CSSレコーディングに依存します

- ファイル数量：制限がありません。

サーバーからのアップロード
サーバーからのアップロードガイド

最終更新日：：2023-10-26 17:31:32

操作シナリオ

サーバーからのビデオアップロードとは、Appバックエンドがビデオを VODプラットフォームにアップロードすることを指します。ここでは、サーバーAPIを使用してビデオを アップロードする方法を紹介します。

前提条件

1. サービスのアクティブ化

VODサービスをアクティブ化します。詳細については、購入ガイドラインをご参照ください。

2. Tencent Cloud APIキーの取得

サーバーAPIの呼び出しに必要なセキュリティ証明（SecretIdとSecretKey）を取得します。具体的な手順は次のとおりです。

1. コンソールにログインし、【クラウド製品】>【CAM】>【APIキー管理】を選択し、「APIキー管理」ページに進みます。
2. Tencent Cloud APIキーを取得します。キーを作成していない場合は、【キーの作成】をクリックして、SecretIdとSecretKeyのペアを作成することができます。

操作手順

1. アップロードの開始

アップロードの開始方式は、SDKによるアップロード開始とAPIによるアップロード開始に分かれます。

SDKから開始するアップロード方法

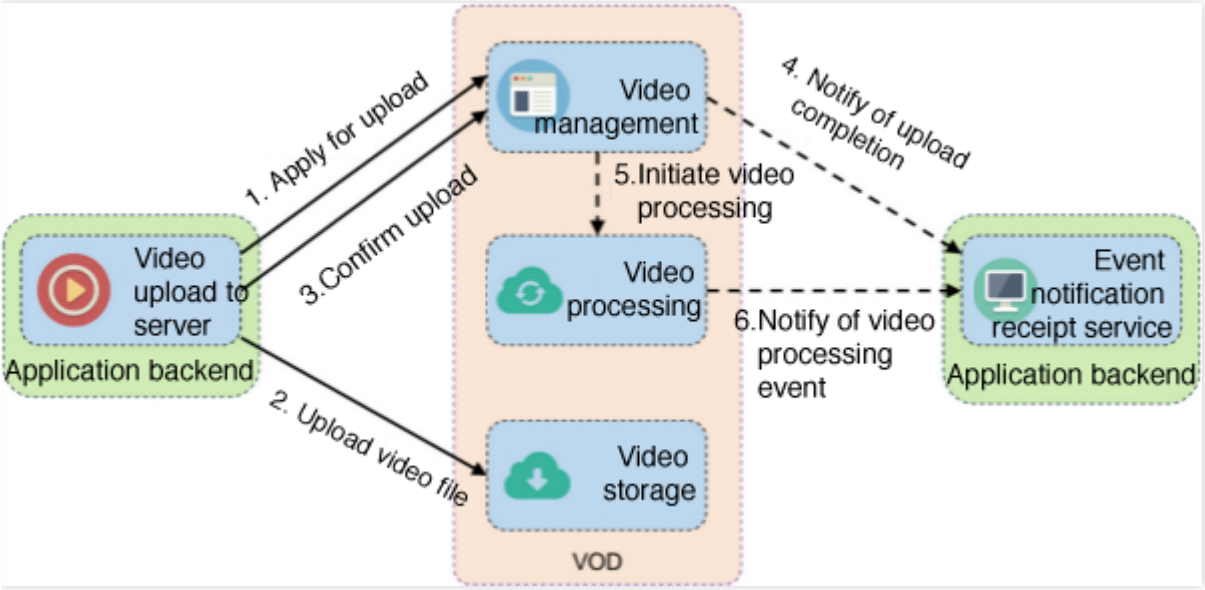
ユーザー開発サイドでアップロード機能を扱いやすくするため、VODでは多様な言語のプラットフォームによるSDKを提供しています。各言語のSDKにはいずれも対応するDemoが含まれています。詳細については、以下をご参照ください。

- PHP SDK
- Java SDK

- Python SDK
- Node.js SDK
- Go SDK
- C# SDK

APIから開始するアップロード方法

VODが提供するアップロードSDKにおいて、Appバックエンドの使用言語がカバーされていない場合は、Appバックエンドで自分でVODのサーバーAPIを呼び出し、ビデオのアップロードを行なう必要があります（この方式は複雑なため、推奨しません）。APIによるアップロードの作業フローチャートは次のとおりです。



SDK方式のアップロードと比べ、API方式によるアップロードは、アップロードの申請やファイルのアップロードなどの手順を自分で実行する必要があります。またファイルのアップロードもSDKの方式ほど便利ではありません。大きなファイルについては、パートアップロードなどのロジックを自分で実行する必要があります。詳細については、以下をご参照ください。

- サーバーAPI - アップロードの申請
- サーバーAPI - ファイルのアップロード
- サーバーAPI - アップロードの確認

高度な機能

- **アップロード時のタスクフローの指定**
  ビデオのアップロードが完了した後、 ビデオ処理タスクフロー（トランスコード、スクリーンキャプチャなど）を自動的に開始したい場合は、サーバーAPI アップロードの申請を呼び出す時に、 `Procedure` パラメータによって実現することができ、このパラメータの値をタスクフローテンプレート名にします。VODでは、タスクフローテンプレートの作成し、テンプレートに命名する機能をサポートしており、タスクフローの開始時

に、タスクフローテンプレート名によって開始したいタスクを表すことができます。VODが提供する多言語のSDKはいずれもタスクフローパラメータの指定が可能です。詳細については以下をご参照ください。

- PHP SDK
- Java SDK
- Python SDK
- Node.js SDKSDK
- Go SDK
- C# SDK

- **アップロード時にストレージリージョンを指定**
  VODが提供するストレージリージョンは、デフォルトでシンガポール に設定されています。他のリージョンに保存する必要がある場合は、コンソールで他のストレージリージョンを自分で追加できます。詳細については、アップロードストレージ設定をご参照ください。設定が完了すると、サーバーAPI アップロードの申請 を呼び出す時に、 `StorageRegion` パラメータによって実現でき、このパラメータの値をストレージリージョンの英語の略称にします。VODが提供する多言語 SDKはいずれもアップロード時のストレージリージョンの指定をサポートしています。詳細については以下をご参照ください。

  - PHP SDK
  - Java SDK
  - Python SDK
  - Node.js SDKSDK
  - Go SDK
  - C# SDK

2. イベント通知

ビデオのアップロードが完了すると、VODはAppバックエンドに イベント通知 - ビデオアップロード完了 を開始します。Appバックエンドは、このイベントを通じてビデオのアップロード動作を認識することができます。イベント通知を受信するには、Appは コンソール - コールバック設定 に移動してイベント通知を有効にする必要があります。イベント通知 - ビデオアップロード完了 には、主に次の情報が含まれています。

- 新しいビデオファイルのFileIdとURL。
- VODは、ビデオアップロード時のパススルーフィールドの指定をサポートしており、イベントが完了すると、パススルーフィールドはAppバックエンドに送信されます。イベント通知には、次のフィールドが含まれています。

- `SourceType` ：このフィールドはTencent Cloudによって `ServerUpload` と定められており、アップロードがサーバーからであることを表しています。
- `SourceContext` ：ユーザー定義のパススルーフィールドです。署名の配布中にAppバックエンドに指定されます。これは署名の `sourceContext` パラメータに対応します。

- VODでは、ビデオアップロード完了時にビデオ処理を自動的に開始する機能をサポートしています。アップロード時にビデオ処理タスクフローを指定すると、イベント通知の内容にタスクID、すなわちイベント通知の中の `data.procedureTaskId` フィールドが添付されてきます。

詳細については、以下をご参照ください。

- タスク管理とイベント通知
- イベント通知 - ビデオアップロード完了

ファイルのアップロード

最終更新日：：2023-10-26 17:31:32

インターフェースの説明

- 小さなファイル（5MB未満）のアップロードに使用するAPIは、シンプルなファイルアップロードをご参照ください。
- 大きなファイル（5MB超過）のアップロードに使用するAPI、パートアップロードの初期化、パートごとのアップロード、パートアップロードの終了をご参照ください。

機能説明

1. メディア（およびカバー）ファイルをアップロードします。
2. APIをサーバーでアップロードする手順は、サーバーからのアップロードの概要をご参照ください。

SDKから

コンテナのSDKを使用してAPIを呼び出すことを推奨します。

APIから

使用方法は、上記APIリンクの中のドキュメントをご参照ください。各APIの構文は次のとおりです。

```
PUT <ObjectName> HTTP/1.1
Host: <BucketName>-<APPID>.cos.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

構文の中の以下の変数は、VODのアップロード申請で戻ってくる結果の値を取ります。

- `<objectname>` は**MediaStoragePath**（カバーファイルに対応するのは **CoverStoragePath**）です。
- `<bucketname>-<appid>` は**StorageBucket**です。
- `<region>` は**StorageRegion**です。

APIリクエストについて、注意すべき点は以下のとおりです。

- Authorizationの署名には、VODのアップロード申請で戻ってくる結果の中の**TempCertificate**のSecretIdとSecretKeyを使用します。署名のドキュメントを参照して計算してください。
- HTTPヘッダーまたはPOSTリクエストパケットのform-dataの中で**x-cos-security-token**フィールド（そのリクエストに使用するセキュリティトークンが表示される）を渡します。戻り値は、**TempCertificate**の中のTokenフィールドとなります。

Java SDK

最終更新日：：2023-10-26 17:31:32

サーバーでのビデオアップロードのシナリオを実現させるために、VODではJava SDKを提供しています。アップロードのフローは、サーバーからのアップロードガイドをご参照ください。

統合方式

Maven依存によるインポート

プロジェクトのpom.xmlファイルにVOD SDKの依存を追加すれば完了です。

```
<dependency>
<groupId>com.qcloud</groupId>
<artifactId>vod_api</artifactId>
<version>2.1.4</version>
</dependency>
```

jarパッケージによるインポート

プロジェクトがMaven方式の採用で依存管理を行なっていない場合は、次の方式を採用し、各々に必要なjarパッケージをダウンロードして、プロジェクトにインポートすることができます。

jarファイル	説明
vod_api-2.1.4.jar	VOD SDK。
jackson-annotations-2.9.0.jar,jackson-core-2.9.7.jar,jackson-databind-2.9.7.jar,gson-2.2.4.jar	オープンソースのJSON関連ライブラリ。
cos_api-5.4.10.jar	Tencent Cloud Cloud Object Storageサービス COS SDK。
tencentcloud-sdk-java-3.1.2.jar	Tencent Cloud API SDK。
commons-codec-1.10.jar,commons-logging-1.2.jar,log4j-1.2.17.jar,slf4j-api-1.7.21.jar,slf4j-log4j12-1.7.21.jar	オープンソースのログ関連ライブラリ。
httpclient-4.5.3.jar,httpcore-4.4.6.jar,okhttp-2.5.0.jar,okio-1.6.0.jar	オープンソースのHTTP処理ライブラリ。
joda-time-2.9.9.jar	オープンソースの時間処理ライブラリ。

jarファイル	説明
jaxb-api-2.3.0.jar	オープンソースのXML処理ライブラリ。
bcprov-jdk15on-1.59.jar	オープンソースの暗号化処理ライブラリ。

Java SDK関連jarパッケージをクリックして、ダウンロードしたjarパッケージをプロジェクトにインポートすれば、使用できます。

シンプルなアップロード

アップロードクライアントオブジェクトの初期化

Tencent Cloud APIキーを使用して、VodUploadClientインスタンスを初期化します。

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
```

アップロードリクエストのオブジェクト作成

メディアのローカルアップロードパスを設定します。

```
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
```

アップロードの呼び出し

アップロードメソッドを呼び出し、アクセスポイントリージョンおよびアップロードリクエストを渡します。

```
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// サービスチームによるトラブルシューティング
logger.error("Upload Err", e);
}
```

説明：

アップロード方法は、ファイルのサイズに応じて、通常アップロードとマルチパートアップロードが自動的に選択されます。マルチパートアップロードの各手順を気にすることなく、マルチパートアップロードを行うことができます。

高度な機能

カバーの付加

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
request.setCoverFilePath("/data/videos/Wildlife.jpg");
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// サービスチームによるトラブルシューティング
logger.error("Upload Err", e);
}
```

タスクフローの指定

まず、タスクフローテンプレートの作成、およびテンプレートに対する命名を行います。タスクフロー時に、このタスクフローテンプレート名を使用して `Procedure` パラメータを設定すれば、アップロード成功後、タスクフローを自動的に実行することができます。

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
request.setProcedure("Your Procedure Name");
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// サービスチームによるトラブルシューティング
logger.error("Upload Err", e);
}
```

サブアプリケーションのアップロード

サブアプリケーションIDを渡します。アップロード成功後、リソースは具体的なサブアプリケーションにのみ属します。

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
request.setSubAppId(101);
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// サービスチームによるトラブルシューティング
logger.error("Upload Err", e);
}
```

ストレージリージョンの指定

コンソールで目標ストレージリージョンがアクティブ化されているか確認します。アクティブ化されていない場合は、アップロードストレージ設定を参考とすることができます。最後に、 `StorageRegion` の属性によって、ストレージリージョンの 英語の略称を設定します。

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
request.setStorageRegion("ap-chongqing");
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// サービスチームによるトラブルシューティング
logger.error("Upload Err", e);
}
```

パート同時実行数の指定

パート同時実行数は、大きなファイルに対応するもので、複数のパートに分割すると同時にアップロードするものです。パート同時アップロードのメリットは、1個のファイルのアップロードを素早く完了させられることで、SDKはファイルのサイズを基に、通常アップロードとパートアップロードを自動的に選択します。ユーザーは、パートアップロードの各手順に気を遣う必要もなく、パートでのアップロードを実現できます。こうしてファイルの同時パート数は、 `ConcurrentUploadNumber` パラメータによって指定します。

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
```

```
request.setMediaFilePath("/data/videos/Wildlife.wmv");
request.setConcurrentUploadNumber(5);
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// サービスチームによるトラブルシューティング
logger.error("Upload Err", e);
}
```

臨時証明書の使用によるアップロード

臨時証明書の関連キー情報を渡し、臨時証明書を使用して身分を検証し、アップロードします。

```
VodUploadClient client = new VodUploadClient("Credentials TmpSecretId", "Credenti
als TmpSecretKey", "Credentials Token");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// サービスチームによるトラブルシューティング
logger.error("Upload Err", e);
}
```

アップロードのプロキシ設定

アップロードのプロキシ設定では、関係するプロトコルおよびデータはいずれもプロキシ経由で処理します。開発者はプロキシの助けを借りて、ファイルを自社イントラネットからTencent Cloudにアップロードすることができます。

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/Wildlife.wmv");
HttpProfile httpProfile = new HttpProfile();
httpProfile.setProxyHost("your proxy ip");
httpProfile.setProxyPort(8080); //your proxy port
client.setHttpProfile(httpProfile);
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// サービスチームによるトラブルシューティング
```

```
logger.error("Upload Err", e);
}
```

アダプティブビットレートストリーミング（ABS）ファイルのアップロード

このSDKでアップロードをサポートするABS形式のパッケージにはHLSとDASHがあります。manifest（M3U8またはMPD）で引用するメディアファイルは、必ず相対パスとし（URLおよび絶対パスは不可）、同時にmanifestと同じクラスのディレクトリまたは下の階層のディレクトリに置く必要があります（`../` は使用不可）。SDKのアップロードインターフェースを呼び出す時に、`MediaFilePath` パラメータにmanifest パスを入力すると、SDKが関連するメディアファイルリストを解析し、一緒にアップロードします。

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.setMediaFilePath("/data/videos/prog_index.m3u8");
try {
VodUploadResponse response = client.upload("ap-guangzhou", request);
logger.info("Upload FileId = {}", response.getFileId());
} catch (Exception e) {
// サービスチームによるトラブルシューティング
logger.error("Upload Err", e);
}
```

インターフェースの説明

アップロードクライアントクラス `VodUploadClient`

属性名	属性説明	タイプ	入力必須
secretId	Tencent Cloud APIキーID。	String	はい
secretKey	Tencent Cloud API Key。	String	はい

アップロードリクエストクラス `VodUploadRequest`

属性名	属性説明	タイプ	入力必須
MediaFilePath	アップロード予定のメディアファイルパス。ローカルパスにする必要があります。URLはサポートしていません。	String	はい

属性名	属性説明	タイプ	入力必須
SubAppId	VOD サブアプリケーションID。サブアプリケーションの中のリソースにアクセスしたい場合は、このフィールドにサブアプリケーションIDを入力します。アクセスしない場合、このフィールドは入力不要です。	Integer	いいえ
MediaType	アップロード予定のメディアファイルタイプ。選択可能なタイプの詳細は、ビデオアップロードの概要をご参照ください。MediaFilePathに拡張子が付いている場合は入力不要です。	String	いいえ
MediaName	アップロード後のメディアの名前。入力しない場合は、デフォルトでMediaFilePathのファイル名を採用します。	String	いいえ
CoverFilePath	アップロード予定のカバーファイルパス。ローカルパスにする必要があります。URLはサポートしていません。	String	いいえ
CoverType	アップロード予定のカバーファイルタイプ。選択可能なタイプの詳細は、ビデオアップロードの概要をご参照ください。CoverFilePathに拡張子が付いている場合は入力不要です。	String	いいえ
Procedure	アップロード後に自動的に実行させたいタスクフロー名。このパラメータは、タスクフローの作成（API方式 またはコンソール方式）時にユーザーが指定します。具体的な内容は、タスクフロー概要をご参照ください。	String	いいえ
ExpireTime	メディアファイルの期限切れ時間。表記形式はISO 8601規格に準拠します。詳細については、ISO日時表記形式の説明をご参照ください。	String	いいえ
ClassId	カテゴリーID。メディアのカテゴリー管理に使用します。カテゴリー作成 インターフェースによってカテゴリーを作成し、カテゴリーIDを取得することができます。	Integer	いいえ
SourceContext	ソースコンテキスト。ユーザーリクエスト情報のパススルーに使用します。アップロードコールバックインターフェースは、このフィールドの値を戻します。最長250文字。	String	いいえ

属性名	属性説明	タイプ	入力必須
StorageRegion	ストレージリージョン。ストレージを予定/希望するリージョンを指定します。このフィールドにはストレージリージョンの英語の略称を入力します。	String	いいえ
ConcurrentUploadNumber	パート同時実行数。大きなファイルを対象にパートアップロードする時に有効となります。	Integer	いいえ

アップロードレスポンスクラス `VodUploadResponse`

属性名	属性説明	タイプ
FileId	メディアファイルの一意の標識。	String
MediaUrl	メディア再生アドレス。	String
CoverUrl	メディアカバーアドレス。	String
RequestId	一意のリクエストID。リクエストごとに返されます。問題を特定する時はその回のリクエストのRequestIdを提供する必要があります。	String

アップロードメソッド `VodUploadClient.upload(String region, VodUploadRequest request)`

パラメータ名	パラメータの説明	タイプ	入力必須
region	アクセスポイントリージョン。どのリージョンのVODサーバーにリクエストするかであり、ストレージリージョンとは異なります。具体的な内容は、サポートするリージョンリストをご参照ください。	String	はい
request	アップロードリクエスト。	VodUploadRequest	はい

エラーコードリスト

ステータスコード	意味
InternalError	内部エラー。
InvalidParameter.ExpireTime	パラメータ値のエラー：期限切れ時間。
InvalidParameterValue.CoverType	パラメータ値のエラー：カバーのタイプ。

ステータスコード	意味
InvalidParameterValue.MediaType	パラメータ値のエラー：メディアタイプ。
InvalidParameterValue.SubAppId	パラメータ値のエラー：サブアプリケーションID。
InvalidParameterValue.VodSessionKey	パラメータ値のエラー：VODセッション。
ResourceNotFound	リソースがありません。

C# SDK

最終更新日：：2023-10-26 17:32:09

サーバーでのビデオアップロードのシナリオを実現させるために、VODではC# SDKを提供しています。アップロードのフローは、サーバーからのアップロードガイドをご参照ください。

統合方式

nugetによるインストール

1. コマンドラインによるインストール：

```
dotnet add package VodSDK --version 1.0.1
```

2. Visual Studioのnugetパッケージ管理ツールによって、VodSDKを検索し、インストールします。

ソースコードパッケージによるインストール

プロジェクトの中でnugetツールを使用していない場合は、ソースコードを直接ダウンロードし、プロジェクトの中にインポートして使用することができます。

- Githubからアクセス
- クリックしてC# SDKをダウンロード

最新のコードがダウンロードされ、解凍後、プロジェクトの作業ディレクトリ下にインストールされますので、Visual Studio 2017を使用して開き、コンパイルします。このSDKはまだ外部のパッケージに依存しているため、以下のSDKを同時にインストールする必要があります。

- Tencent Cloud API SDK
- Cloud Object Storage SDK

シンプルなアップロード

アップロードクライアントオブジェクトの初期化

Tencent Cloud APIキーを使用して、VodUploadClientインスタンスを初期化します。

```
using System;
using VodSDK;
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
```

アップロードリクエストのオブジェクト作成

メディアのローカルアップロードパスを設定します。

```
VodUploadRequest request = new VodUploadRequest();
request.MediaFilePath = "/data/videos/Wildlife.wmv";
```

アップロードの呼び出し

アップロードメソッドを呼び出し、アクセスポイントリージョンおよびアップロードリクエストを渡します。

```
try
{
VodUploadResponse response = client.Upload("ap-guangzhou", request);
// メディアFileIdの出力
Console.WriteLine(response.FileId);
}
catch (Exception e)
{
// サービスチームによるトラブルシューティング
Console.WriteLine(e);
}
```

アップロード方法は、ファイルのサイズに応じて、通常アップロードとマルチパートアップロードが自動的に選択されます。マルチパートアップロードの各手順を気にすることなく、マルチパートアップロードを行うことができます。

高度な機能

カバーの付加

```
using System;
using VodSDK;
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
```

```
request.MediaFilePath = "/data/videos/Wildlife.wmv";
request.CoverFilePath = "/data/videos/Wildlife.jpg";
try
{
VodUploadResponse response = client.Upload("ap-guangzhou", request);
// メディアFileIdの出力
Console.WriteLine(response.FileId);
}
catch (Exception e)
{
// サービスチームによるトラブルシューティング
Console.WriteLine(e);
}
```

タスクフローの指定

まず、タスクフローテンプレートの作成、およびテンプレートに対する命名を行います。タスクフロー時に、このタスクフローテンプレート名を使用して `Procedure` パラメータを設定すれば、アップロード成功後、タスクフローを自動的に実行することができます。

```
using System;
using VodSDK;
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.MediaFilePath = "/data/videos/Wildlife.wmv";
request.Procedure = "Your Procedure Name";
try
{
VodUploadResponse response = client.Upload("ap-guangzhou", request);
// メディアFileIdの出力
Console.WriteLine(response.FileId);
}
catch (Exception e)
{
// サービスチームによるトラブルシューティング
Console.WriteLine(e);
}
```

サブアプリケーションのアップロード

サブアプリケーションIDを渡します。アップロード成功後、リソースは具体的なサブアプリケーションにのみ属します。

```
using System;
using VodSDK;
```

```
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.MediaFilePath = "/data/videos/Wildlife.wmv";
request.SubAppId = 101;
try
{
VodUploadResponse response = client.Upload("ap-guangzhou", request);
// メディアFileIdの出力
Console.WriteLine(response.FileId);
}
catch (Exception e)
{
// サービスチームによるトラブルシューティング
Console.WriteLine(e);
}
```

ストレージリージョンの指定

コンソールで目標ストレージリージョンがアクティブ化されているか確認します。アクティブ化されていない場合は、アップロードストレージ設定を参考とすることができます。最後に、 `StorageRegion` の属性によって、ストレージリージョンの 英語の略称を設定します。

```
using System;
using VodSDK;
VodUploadClient client = new VodUploadClient("your secretId", "your secretKey");
VodUploadRequest request = new VodUploadRequest();
request.MediaFilePath = "/data/videos/Wildlife.wmv";
request.StorageRegion = "ap-chongqing";
try
{
VodUploadResponse response = client.Upload("ap-guangzhou", request);
// メディアFileIdの出力
Console.WriteLine(response.FileId);
}
catch (Exception e)
{
// サービスチームによるトラブルシューティング
Console.WriteLine(e);
}
```

インターフェースの説明

アップロードクライアントクラス `VodUploadClient`

属性名	属性説明	タイプ	入力必須
secretId	Tencent Cloud APIキーID。	String	はい
secretKey	Tencent Cloud API Key。	String	はい

アップロードリクエストクラス `VodUploadRequest`

属性名	属性説明	タイプ	入力必須
MediaFilePath	アップロード予定のメディアファイルパス。ローカルパスにする必要があります。URLはサポートしていません。	String	はい
SubAppId	VOD サブアプリケーションID。サブアプリケーションの中のリソースにアクセスしたい場合は、このフィールドにサブアプリケーションIDを入力します。アクセスしない場合、このフィールドは入力不要です。	Integer	いいえ
MediaType	アップロード予定のメディアファイルタイプ。選択可能なタイプの詳細は、ビデオアップロードの概要をご参照ください。MediaFilePathに拡張子が付いている場合は入力不要です。	String	いいえ
MediaName	アップロード後のメディアの名前。入力しない場合は、デフォルトでMediaFilePathのファイル名を採用します。	String	いいえ
CoverFilePath	アップロード予定のカバーファイルパス。ローカルパスにする必要があります。URLはサポートしていません。	String	いいえ
CoverType	アップロード予定のメディアファイルタイプ。選択可能なタイプの詳細は、ビデオアップロードの概要をご参照ください。CoverFilePathに拡張子が付いている場合は入力不要です。	String	いいえ
Procedure	アップロード後に自動的に実行させたいタスクフロー名。このパラメータは、タスクフローの作成（API方式 またはコンソール方式）時にユーザーが指定します。具体的な内容は、タスクフロー概要をご参照ください。	String	いいえ
ExpireTime	メディアファイルの期限切れ時間。表記形式はISO 8601規格に準拠します。詳細については、ISO日時表記形式の説明をご参照ください。	String	いいえ
ClassId	カテゴリーID。メディアのカテゴリー管理に使用します。カテゴリー作成インターフェースによってカテゴリーを作成し、カテゴリーIDを取得することができます。	Integer	いいえ

属性名	属性説明	タイプ	入力必須
SourceContext	ソースコンテキスト。ユーザーリクエスト情報のパススルーに使用します。アップロードコールバックインターフェースは、このフィールドの値を戻します。最長250文字。	String	いいえ
StorageRegion	ストレージリージョン。ストレージを予定/希望するリージョンを指定します。このフィールドにはストレージリージョンの英語の略称を入力します。	String	いいえ

アップロードレスポンスクラス `VodUploadResponse`

属性名	属性説明	タイプ
FileId	メディアファイルの一意の標識。	String
MediaUrl	メディア再生アドレス。	String
CoverUrl	メディアカバーアドレス。	String
RequestId	一意のリクエストID。リクエストごとに返されます。問題を特定する時はその回のリクエストのRequestIdを提供する必要があります。	String

アップロードメソッド `VodUploadClient.Upload(String region, VodUploadRequest request)`

パラメータ名	パラメータの説明	タイプ	入力必須
region	アクセスポイントリージョン。どのリージョンのVODサーバーにリクエストするかであり、ストレージリージョンとは異なります。具体的な内容は、サポートするリージョンリストをご参照ください。	String	はい
request	アップロードリクエスト。	VodUploadRequest	はい

エラーコードリスト

ステータスコード	意味
InternalError	内部エラー。
InvalidParameter.ExpireTime	パラメータ値のエラー：期限切れ時間。
InvalidParameterValue.CoverType	パラメータ値のエラー：カバーのタイプ。

ステータスコード	意味
InvalidParameterValue.MediaType	パラメータ値のエラー：メディアタイプ。
InvalidParameterValue.SubAppId	パラメータ値のエラー：サブアプリケーションID。
InvalidParameterValue.VodSessionKey	パラメータ値のエラー：VODセッション。
ResourceNotFound	リソースがありません。

PHP SDK

最終更新日：：2023-10-26 17:31:32

サーバーでのビデオアップロードのシナリオを実現させるために、VODではPHP SDKを提供しています。アップロードのフローは、サーバーからのアップロードガイドをご参照ください。

統合方式

composerを使用してインポート

```
{
"require": {
"qcloud/vod-sdk-v5": "v2.4.0"
}
}
```

ソースコードパッケージによるインストール

1. プロジェクトの中で依存管理用のcomposerツールを使用していない場合は、Githubからアクセスでソースコードを直接ダウンロードして、プロジェクトにインポートすると使用できます。
2. vod-sdk.zipファイルをプロジェクトの中で解凍し、autoload.phpファイルをインポートすれば使用できます。

シンプルビデオアップロード

アップロードオブジェクトの初期化

Tencent Cloud APIキーを使用して、VodUploadClientインスタンスを初期化します。

composerを使用してインポート

```
<?php
require 'vendor/autoload.php';

use Vod\VodUploadClient;

$client = new VodUploadClient("your secretId", "your secretKey");
```

ソースコードを使用してインポート

```
<?php
require 'vod-sdk-v5/autoload.php';

use Vod\VodUploadClient;

$client = new VodUploadClient("your secretId", "your secretKey");
```

アップロードリクエストのオブジェクト作成

```
use Vod\Model\VodUploadRequest;

$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/Wildlife.wmv";
```

アップロードの呼び出し

アップロードメソッドを呼び出し、アクセスポイントリージョンおよびアップロードリクエストを渡します。

```
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
} catch (Exception $e) {
// アップロード異常の処理
echo $e;
}
```

説明：
アップロード方法は、ファイルのサイズに応じて、通常アップロードとマルチパートアップロードが自動的に選択されます。マルチパートアップロードの各手順を気にすることなく、マルチパートアップロードを行うことができます。

高度な機能

カバーの付加

```
<?php
require 'vendor/autoload.php';
```

```
use Vod\VodUploadClient;
use Vod\Model\VodUploadRequest;

$client = new VodUploadClient("your secretId", "your secretKey");
$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/Wildlife.wmv";
$req->CoverFilePath = "/data/videos/Wildlife-Cover.png";
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
echo "CoverUrl -> ". $rsp->CoverUrl . "\n";
} catch (Exception $e) {
// アップロード異常の処理
echo $e;
}
```

タスクフローの指定

まず、タスクフローテンプレートの作成、およびテンプレートに対する命名を行います。タスクフロー時に、このタスクフローテンプレート名を使用して `Procedure` パラメータを設定すれば、アップロード成功後、タスクフローを自動的に実行することができます。

```
<?php
require 'vendor/autoload.php';

use Vod\VodUploadClient;
use Vod\Model\VodUploadRequest;

$client = new VodUploadClient("your secretId", "your secretKey");
$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/Wildlife.wmv";
$req->Procedure = "Your Procedure Name";
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
} catch (Exception $e) {
// アップロード異常の処理
echo $e;
}
```

サブアプリケーションのアップロード

サブアプリケーションIDを渡します。アップロード成功後、リソースは具体的なサブアプリケーションにのみ属します。

```
<?php
require 'vendor/autoload.php';

use Vod\VodUploadClient;
use Vod\Model\VodUploadRequest;

$client = new VodUploadClient("your secretId", "your secretKey");
$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/Wildlife.wmv";
$req->SubAppId = 101;
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
} catch (Exception $e) {
// アップロード異常の処理
echo $e;
}
```

ストレージリージョンの指定

コンソールで目標ストレージリージョンがアクティブ化されているか確認します。アクティブ化されていない場合は、アップロードストレージ設定をご参照ください。最後に、 `StorageRegion` の属性によって、ストレージリージョンの 英語の略称を設定します。

```
<?php
require 'vendor/autoload.php';

use Vod\VodUploadClient;
use Vod\Model\VodUploadRequest;

$client = new VodUploadClient("your secretId", "your secretKey");
$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/Wildlife.wmv";
$req->StorageRegion = "ap-chongqing";
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
} catch (Exception $e) {
// アップロード異常の処理
```

```
echo $e;
}
```

臨時証明書の使用によるアップロード

臨時証明書の関連キー情報を渡し、臨時証明書を使用して身分を検証し、アップロードします。

```
<?php
require 'vendor/autoload.php';

use Vod\VodUploadClient;
use Vod\Model\VodUploadRequest;

$client = new VodUploadClient("Credentials TmpSecretId", "Credentials TmpSecretKe
y", "Credentials Token");
$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/Wildlife.wmv";
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
} catch (Exception $e) {
// アップロード異常の処理
echo $e;
}
```

アップロードのプロキシ設定

アップロードのプロキシ設定では、関係するプロトコルおよびデータはいずれもプロキシ経由で処理します。開発者はプロキシを参照しながら、ファイルを自社インターネットからテンセントクラウドにアップロードします。

```
<?php
require 'vendor/autoload.php';

use Vod\VodUploadClient;
use Vod\Model\VodUploadRequest;
use Vod\Model\VodUploadHttpProfile;

$client = new VodUploadClient("your secretId", "your secretKey");
$uploadHttpProfile = new VodUploadHttpProfile("your proxy addr");
$client->setHttpProfile($uploadHttpProfile);
$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/Wildlife.wmv";
try {
$rsp = $client->upload("ap-guangzhou", $req);
```

```
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
} catch (Exception $e) {
// アップロード異常の処理
echo $e;
}
```

アダプティブビットレートストリーミング（ABS）ファイルのアップロード

このSDKでアップロードをサポートするABS形式のパッケージにはHLSとDASHがあります。manifest（M3U8またはMPD）で引用するメディアファイルは、必ず相対パスとし（URLおよび絶対パスは不可）、同時にmanifestと同じクラスのディレクトリまたは下の階層のディレクトリに置く必要があります（ `../` は使用不可）。SDKのアップロードインターフェースを呼び出す時に、 `MediaFilePath` パラメータにmanifest パスを入力すると、SDKが関連するメディアファイルリストを解析し、一緒にアップロードされます。

```
<?php
require 'vendor/autoload.php';

use Vod\VodUploadClient;
use Vod\Model\VodUploadRequest;

$client = new VodUploadClient("your secretId", "your secretKey");
$req = new VodUploadRequest();
$req->MediaFilePath = "/data/videos/prog_index.m3u8";
try {
$rsp = $client->upload("ap-guangzhou", $req);
echo "FileId -> ". $rsp->FileId . "\n";
echo "MediaUrl -> ". $rsp->MediaUrl . "\n";
} catch (Exception $e) {
// アップロード異常の処理
echo $e;
}
```

インターフェースの説明

アップロードクライアントクラス `VodUploadClient`

属性名	属性説明	タイプ	入力必須
secretId	Tencent Cloud APIキーID。	String	はい
secretKey	Tencent Cloud API Key。	String	はい

アップロードリクエストクラス `VodUploadRequest`

属性名	属性説明	タイプ	入力必須
MediaFilePath	アップロード予定のメディアファイルパス。ローカルパスにしてください。URLはサポートしていません。	String	はい
SubAppId	VOD サブアプリケーションID。サブアプリケーションの中のリソースにアクセスしたい場合は、このフィールドにサブアプリケーションIDを入力します。アクセスしない場合、この項目は入力不要です。	Integer	いいえ
MediaType	アップロード予定のメディアファイルタイプ。選択可能なタイプの詳細は、ビデオアップロードの概要をご参照ください。MediaFilePathに拡張子が付いている場合は入力不要です。	String	いいえ
MediaName	アップロード後のメディアの名前。入力しない場合は、デフォルトでMediaFilePathのファイル名を採用します。	String	いいえ
CoverFilePath	アップロード予定のカバーファイルパス。ローカルパスにしてください。URLはサポートしていません。	String	いいえ
CoverType	アップロード予定のカバーファイルタイプ。選択可能なタイプの詳細は、ビデオアップロードの概要をご参照ください。CoverFilePathに拡張子が付いている場合は入力不要です。	String	いいえ
Procedure	アップロード後に自動的に実行させたいタスクフロー名。このパラメータは、タスクフローの作成（API方式 またはコンソール方式）時にユーザーが指定します。詳細内容は、タスクフロー概要をご参照ください。	String	いいえ
ExpireTime	メディアファイルの期限。表記形式はISO 8601規格に準拠します。詳細については、ISO日時表記形式の説明をご参照ください。	String	いいえ
ClassId	カテゴリーID。メディアのカテゴリー管理に使用します。カテゴリー作成 インターフェースによってカテゴリーを作成し、カテゴリーIDを取得することができます。	Integer	いいえ
SourceContext	ソースコンテキスト。ユーザーリクエスト情報のパススルーに使用します。アップロードコールバックインターフェースは、このフィールドの値を戻します。最長250文字。	String	いいえ
StorageRegion	ストレージリージョン。ストレージを予定/希望するリージョンを指定します。このフィールドにはストレージリージョンの英語の略称を入力します。	String	いいえ

アップロードレスポンスクラス `VodUploadResponse`

属性名	属性説明	タイプ
FileId	メディアファイルの一意の標識。	String
MediaUrl	メディア再生アドレス。	String
CoverUrl	メディアカバーアドレス。	String
RequestId	一時的なリクエストID。リクエストごとに返されます。問題を特定する時はその回のリクエストのRequestIdを提供してください。	String

アップロードメソッド `VodUploadClient.upload(String region, VodUploadRequest request)`

パラメータ名	パラメータの説明	タイプ	入力必須
region	アクセスポイントリージョン。どのリージョンのVODサーバーにリクエストするかであり、ストレージリージョンとは異なります。詳細内容は、サポートするリージョンリストをご参照ください。	String	はい
request	アップロードリクエスト。	VodUploadRequest	はい

エラーコードリスト

ステータスコード	意味
InternalError	内部エラー。
InvalidParameter.ExpireTime	パラメータ値のエラー：期限。
InvalidParameterValue.CoverType	パラメータ値のエラー：カバーのタイプ。
InvalidParameterValue.MediaType	パラメータ値のエラー：メディアタイプ。
InvalidParameterValue.SubAppId	パラメータ値のエラー：サブアプリケーションID。
InvalidParameterValue.VodSessionKey	パラメータ値のエラー：VODセッション。
ResourceNotFound	リソースがありません。

Python SDK

最終更新日：：2023-10-26 17:31:32

サーバーでのビデオアップロードのシナリオを実現させるために、VODではPython SDKを提供しています。アップロードのフローは、サーバーからのアップロードガイドをご参照ください。

統合方式

pipを使用してインストール

```
pip install vod-python-sdk
```

ソースコードパッケージによるインストール

プロジェクトの中でpipツールを使用していない場合は、ソースコードを直接ダウンロードし、プロジェクトの中にインポートして使用することができます。

- Githubからアクセス
- クリックしてPython SDKをダウンロード

最新コードをダウンロードして解凍後：

```
$ cd vod-python-sdk
$ python setup.py install
```

シンプルビデオアップロード

アップロードオブジェクトの初期化

Tencent Cloud APIキーを使用して、VodUploadClientインスタンスを初期化します。

```
from qcloud_vod.vod_upload_client import VodUploadClient

client = VodUploadClient("your secretId", "your secretKey")
```

アップロードリクエストのオブジェクト作成

```
from qcloud_vod.model import VodUploadRequest

request = VodUploadRequest()
request.MediaFilePath = "/data/file/Wildlife.mp4"
```

アップロードの呼び出し

アップロードメソッドを呼び出し、アクセスポイントリージョンおよびアップロードリクエストを渡します。

```
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
except Exception as err:
// サービス異常の処理
print(err)
```

説明：
アップロード方法は、ファイルのサイズに応じて、通常アップロードとマルチパートアップロードが自動的に選択されます。マルチパートアップロードの各手順を気にすることなく、マルチパートアップロードを行うことができます。

高度な機能

カバーの付加

```
from qcloud_vod.vod_upload_client import VodUploadClient
from qcloud_vod.model import VodUploadRequest

client = VodUploadClient("your secretId", "your secretKey")
request = VodUploadRequest()
request.MediaFilePath = "/data/file/Wildlife.mp4"
request.CoverFilePath = "/data/file/Wildlife-Cover.png"
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
print(response.CoverUrl)
except Exception as err:
```

```
// サービス異常の処理
print(err)
```

タスクフローの指定

まず、タスクフローテンプレートの作成、およびテンプレートに対する命名を行います。タスクフロー時に、このタスクフローテンプレート名を使用して `Procedure` パラメータを設定すれば、アップロード成功後、タスクフローを自動的に実行することができます。

```
from qcloud_vod.vod_upload_client import VodUploadClient
from qcloud_vod.model import VodUploadRequest

client = VodUploadClient("your secretId", "your secretKey")
request = VodUploadRequest()
request.MediaFilePath = "/data/file/Wildlife.mp4"
request.Procedure = "Your Procedure Name"
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
except Exception as err:
// サービス異常の処理
print(err)
```

サブアプリケーションのアップロード

サブアプリケーションIDを渡します。アップロード成功後、リソースは具体的なサブアプリケーションにのみ属します。

```
from qcloud_vod.vod_upload_client import VodUploadClient
from qcloud_vod.model import VodUploadRequest

client = VodUploadClient("your secretId", "your secretKey")
request = VodUploadRequest()
request.MediaFilePath = "/data/file/Wildlife.mp4"
request.SubAppId = 101
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
except Exception as err:
// サービス異常の処理
print(err)
```

ストレージリージョンの指定

コンソールで目標ストレージリージョンがアクティブ化されているか確認します。アクティブ化されていない場合は、アップロードストレージ設定を参考とすることができます。最後に、 `StorageRegion` の属性によって、ストレージリージョンの 英語の略称を設定します。

```
from qcloud_vod.vod_upload_client import VodUploadClient
from qcloud_vod.model import VodUploadRequest

client = VodUploadClient("your secretId", "your secretKey")
request = VodUploadRequest()
request.MediaFilePath = "/data/file/Wildlife.mp4"
request.StorageRegion = "ap-chongqing"
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
except Exception as err:
// サービス異常の処理
print(err)
```

パート同時実行数の指定

パート同時実行数は、大きなファイルに対応するもので、複数のパートに分割すると同時にアップロードするものです。パート同時アップロードのメリットは、1個のファイルのアップロードを素早く完了させられることで、SDKはファイルのサイズを基に、通常アップロードとパートアップロードを自動的に選択します。ユーザーは、パートアップロードの各手順に気を遣う必要もなく、パートでのアップロードを実現できます。こうしてファイルの同時パート数は、 `ConcurrentUploadNumber` パラメータによって指定します。

```
from qcloud_vod.vod_upload_client import VodUploadClient
from qcloud_vod.model import VodUploadRequest

client = VodUploadClient("your secretId", "your secretKey")
request = VodUploadRequest()
request.MediaFilePath = "/data/file/Wildlife.mp4"
request.ConcurrentUploadNumber = 5
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
except Exception as err:
// サービス異常の処理
print(err)
```

一時的な証明書の使用によるアップロード

一時的な証明書の関連キー情報を渡し、一時的な証明書を使用して身分を検証し、アップロードします。

```
from qcloud_vod.vod_upload_client import VodUploadClient
from qcloud_vod.model import VodUploadRequest

client = VodUploadClient("Credentials TmpSecretId", "Credentials TmpSecretKey",
"Credentials Token")
request = VodUploadRequest()
request.MediaFilePath = "/data/file/Wildlife.mp4"
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
except Exception as err:
// サービス異常の処理
print(err)
```

アダプティブビットレートストリーミング（ABS）ファイルのアップロード

このSDKでアップロードをサポートするABS形式のパッケージにはHLSとDASHがあります。manifest（M3U8またはMPD）で引用するメディアファイルは、必ず相対パスとし（URLおよび絶対パスは不可）、同時にmanifestと同じクラスのディレクトリまたは下の階層のディレクトリに置く必要があります（ `../` は使用不可）。SDKのアップロードインターフェースを呼び出す時に、 `MediaFilePath` パラメータにmanifest パスを入力すると、SDKが関連するメディアファイルリストを解析し、一緒にアップロードします。

```
from qcloud_vod.vod_upload_client import VodUploadClient
from qcloud_vod.model import VodUploadRequest

client = VodUploadClient("your secretId", "your secretKey")
request = VodUploadRequest()
request.MediaFilePath = "/data/file/prog_index.mp4"
try:
response = client.upload("ap-guangzhou", request)
print(response.FileId)
print(response.MediaUrl)
except Exception as err:
// サービス異常の処理
print(err)
```

インターフェースの説明

アップロードクライアントクラス `VodUploadClient` ：

属性名	属性説明	タイプ	入力必須
secretId	Tencent Cloud APIキーID。	String	はい
secretKey	Tencent Cloud API Key。	String	はい

アップロードリクエストクラス `VodUploadRequest` ：

属性名	属性説明	タイプ	入力必須
MediaFilePath	アップロード予定のメディアファイルパス。ローカルパスにする必要があります。URLはサポートしていません。	String	はい
SubAppId	VOD サブアプリケーションID。サブアプリケーションの中のリソースにアクセスしたい場合は、このフィールドにサブアプリケーションIDを入力します。アクセスしない場合、このフィールドは入力不要です。	Integer	いいえ
MediaType	アップロード予定のメディアファイルタイプ。選択可能なタイプの詳細は、ビデオアップロードの概要をご参照ください。MediaFilePathに拡張子が付いている場合は入力不要です。	String	いいえ
MediaName	アップロード後のメディアの名前。入力しない場合は、デフォルトでMediaFilePathのファイル名を採用します。	String	いいえ
CoverFilePath	アップロード予定のカバーファイルパス。ローカルパスにする必要があります。URLはサポートしていません。	String	いいえ
CoverType	アップロード予定のカバーファイルタイプ。選択可能なタイプの詳細は、ビデオアップロードの概要をご参照ください。CoverFilePathに拡張子が付いている場合は入力不要です。	String	いいえ
Procedure	アップロード後に自動的に実行させたいタスクフロー名。このパラメータは、タスクフローの作成（API方式 またはコンソール方式）時にユーザーが指定します。具体的な内容は、タスクフロー概要をご参照ください。	String	いいえ

属性名	属性説明	タイプ	入力必須
ExpireTime	メディアファイルの期限。表記形式はISO 8601規格に準拠します。詳細については、ISO日時表記形式の説明をご参照ください。	String	いいえ
ClassId	カテゴリーID。メディアのカテゴリー管理に使用します。カテゴリー作成 インターフェースによってカテゴリーを作成し、カテゴリーIDを取得することができます。	Integer	いいえ
SourceContext	ソースコンテキスト。ユーザーリクエスト情報のパススルーに使用します。アップロードコールバックインターフェースは、このフィールドの値を戻します。最長250文字。	String	いいえ
StorageRegion	ストレージリージョン。ストレージを予定/希望するリージョンを指定します。このフィールドにはストレージリージョンの英語の略称を入力します。	String	いいえ
ConcurrentUploadNumber	パート同時実行数。大きなファイルを対象にパートアップロードする時に有効となります。	Integer	いいえ

アップロードレスポンスクラス `VodUploadResponse`

属性名	属性説明	タイプ
FileId	メディアファイルの一意の標識。	String
MediaUrl	メディア再生アドレス。	String
CoverUrl	メディアカバーアドレス。	String
RequestId	一意のリクエストID。リクエストごとに返されます。問題を特定する時はその回のリクエストのRequestIdを提供する必要があります。	String

アップロードメソッド `VodUploadClient.upload(String region, VodUploadRequest request)`

パラメータ名	パラメータの説明	タイプ	入力必須
region	アクセスポイントリージョン。どのリージョンのVODサーバーにリクエストするかであり、ストレージリージョンとは異なります。具体的な内容は、サポートするリージョンリストをご参照ください。	String	はい

パラメータ名	パラメータの説明	タイプ	入力必須
request	アップロードリクエスト。	VodUploadRequest	はい

エラーコードリスト

ステータスコード	意味
InternalError	内部エラー。
InvalidParameter.ExpireTime	パラメータ値のエラー：期限。
InvalidParameterValue.CoverType	パラメータ値のエラー：カバーのタイプ。
InvalidParameterValue.MediaType	パラメータ値のエラー：メディアタイプ。
InvalidParameterValue.SubAppId	パラメータ値のエラー：サブアプリケーションID。
InvalidParameterValue.VodSessionKey	パラメータ値のエラー：VODセッション。
ResourceNotFound	リソースが存在しません。

Node.js SDK

最終更新日：：2023-10-26 17:31:32

サーバーでのビデオアップロードのシナリオを実現させるために、VODではNode SDKを提供しています。アップロードのフローは、サーバーからのアップロードガイドをご参照ください。

統合方式

npmを使用してインストール

```
npm i vod-node-sdk --save
```

ソースコードパッケージによるインストール

プロジェクトの中でnpmツールを使用して依存管理を行なっていない場合は、ソースコードを直接ダウンロードし、プロジェクトの中にインポートして使用することができます。

- Githubからアクセス
- クリックしてNode.js SDKをダウンロード

シンプルビデオアップロード

アップロードオブジェクトの初期化

Tencent Cloud APIキーを使用して、VodUploadClientインスタンスを初期化します。

```
const { VodUploadClient, VodUploadRequest } = require('vod-node-sdk');

client = new VodUploadClient("your secretId", "your secretKey");
```

アップロードリクエストのオブジェクト作成

```
let req = new VodUploadRequest();
req.MediaFilePath = "/data/file/Wildlife.mp4";
```

アップロードの呼び出し

アップロードメソッドを呼び出して、アクセスポイントリージョンおよびアップロードリクエストを渡し、コールバック方式によって戻ってきた情報を取得します。

```
client.upload("ap-guangzhou", req, function (err, data) {
if (err) {
// サービス異常の処理
console.log(err)
} else {
// アップロード成功後の情報の取得
console.log(data.FileId);
console.log(data.MediaUrl);
}
});
```

説明：
アップロード方法は、ファイルのサイズに応じて、通常アップロードとマルチパートアップロードが自動的に選択されます。マルチパートアップロードの各手順を気にすることなく、マルチパートアップロードを行うことができます。

高度な機能

カバーの付加

```
const { VodUploadClient, VodUploadRequest } = require('vod-node-sdk');

client = new VodUploadClient("your secretId", "your secretKey");
let req = new VodUploadRequest();
req.MediaFilePath = "/data/file/Wildlife.mp4";
req.CoverFilePath = "/data/file/Wildlife-cover.png";
client.upload("ap-guangzhou", req, function (err, data) {
if (err) {
// サービス異常の処理
console.log(err)
} else {
// アップロード成功後の情報の取得
console.log(data.FileId);
console.log(data.MediaUrl);
console.log(data.CoverUrl);
}
});
```

タスクフローの指定

まず、タスクフローテンプレートの作成、およびテンプレートに対する命名を行います。タスクフロー時に、このタスクフローテンプレート名を使用して `Procedure` パラメータを設定すれば、アップロード成功後、タスクフローを自動的に実行することができます。

```
const { VodUploadClient, VodUploadRequest } = require('vod-node-sdk');

client = new VodUploadClient("your secretId", "your secretKey");
let req = new VodUploadRequest();
req.MediaFilePath = "/data/file/Wildlife.mp4";
req.Procedure = "Your Procedure Name";
client.upload("ap-guangzhou", req, function (err, data) {
if (err) {
// サービス異常の処理
console.log(err)
} else {
// アップロード成功後の情報の取得
console.log(data.FileId);
console.log(data.MediaUrl);
}
});
```

サブアプリケーションのアップロード

サブアプリケーションIDを渡します。アップロード成功後、リソースは具体的なサブアプリケーションにのみ属します。

```
const { VodUploadClient, VodUploadRequest } = require('vod-node-sdk');

client = new VodUploadClient("your secretId", "your secretKey");
let req = new VodUploadRequest();
req.MediaFilePath = "/data/file/Wildlife.mp4";
req.SubAppId = 101;
client.upload("ap-guangzhou", req, function (err, data) {
if (err) {
// サービス異常の処理
console.log(err)
} else {
// アップロード成功後の情報の取得
console.log(data.FileId);
console.log(data.MediaUrl);
}
});
```

ストレージリージョンの指定

コンソールで目標ストレージリージョンがアクティブ化されているか確認します。アクティブ化されていない場合は、アップロードストレージ設定を参考とすることができます。最後に、 `StorageRegion` の属性によって、ストレージリージョンの 英語の略称を設定します。

```
const { VodUploadClient, VodUploadRequest } = require('vod-node-sdk');

client = new VodUploadClient("your secretId", "your secretKey");
let req = new VodUploadRequest();
req.MediaFilePath = "/data/file/Wildlife.mp4";
req.StorageRegion = "ap-chongqing";
client.upload("ap-guangzhou", req, function (err, data) {
if (err) {
// サービス異常の処理
console.log(err)
} else {
// アップロード成功後の情報の取得
console.log(data.FileId);
console.log(data.MediaUrl);
}
});
```

一時的な証明書の使用によるアップロード

一時的な証明書の関連キー情報を渡し、一時的な証明書を使用して身分を検証し、アップロードします。

```
const { VodUploadClient, VodUploadRequest } = require('vod-node-sdk');

client = new VodUploadClient("Credentials TmpSecretId", "Credentials TmpSecretKe
y", "Credentials Token");
let req = new VodUploadRequest();
req.MediaFilePath = "/data/file/Wildlife.mp4";
client.upload("ap-guangzhou", req, function (err, data) {
if (err) {
// サービス異常の処理
console.log(err)
} else {
// アップロード成功後の情報の取得
console.log(data.FileId);
console.log(data.MediaUrl);
}
});
```

アダプティブビットレートストリーミング（ABS）ファイルのアップロード

このSDKでアップロードをサポートするABS形式のパッケージにはHLSとDASHがあります。manifest（M3U8またはMPD）で引用するメディアファイルは、必ず相対パスとし（URLおよび絶対パスは不可）、同時にmanifestと同じクラスのディレクトリまたは下の階層のディレクトリに置く必要があります（`../` は使用不可）。SDKのアップロードインターフェースを呼び出す時に、`MediaFilePath` パラメータにmanifest パスを入力すると、SDKが関連するメディアファイルリストを解析し、一緒にアップロードします。

```
const { VodUploadClient, VodUploadRequest } = require('vod-node-sdk');

client = new VodUploadClient("your secretId", "your secretKey");
let req = new VodUploadRequest();
req.MediaFilePath = "/data/file/prog_index.m3u8";
client.upload("ap-guangzhou", req, function (err, data) {
if (err) {
// サービス異常の処理
console.log(err)
} else {
// アップロード成功後の情報の取得
console.log(data.FileId);
console.log(data.MediaUrl);
}
});
```

インターフェースの説明

アップロードクライアントクラス `VodUploadClient`

属性名	属性説明	タイプ	入力必須
secretId	Tencent Cloud APIキーID。	String	はい
secretKey	Tencent Cloud API Key。	String	はい

アップロードリクエストクラス `VodUploadRequest`

属性名	属性説明	タイプ	入力必須
MediaFilePath	アップロード予定のメディアファイルパス。ローカルパス（ユーザーサーバー上のパス）にする必要があります。URLはサポートしていません。	String	はい

属性名	属性説明	タイプ	入力必須
SubAppId	VOD サブアプリケーションID。サブアプリケーションの中のリソースにアクセスしたい場合は、このフィールドにサブアプリケーションIDを入力します。アクセスしない場合、このフィールドは入力不要です。	Integer	いいえ
MediaType	アップロード予定のメディアファイルタイプ。選択可能なタイプの詳細は、ビデオアップロードの概要をご参照ください。MediaFilePathに拡張子が付いている場合は入力不要です。	String	いいえ
MediaName	アップロード後のメディアの名前。入力しない場合は、デフォルトでMediaFilePathのファイル名を採用します。	String	いいえ
CoverFilePath	アップロード予定のカバーファイルパス。ローカルパス（ユーザーサーバー上のパス）にする必要があります。URLはサポートしていません。	String	いいえ
CoverType	アップロード予定のカバーファイルタイプ。選択可能なタイプの詳細は、ビデオアップロードの概要をご参照ください。CoverFilePathに拡張子が付いている場合は入力不要です。	String	いいえ
Procedure	アップロード後に自動的に実行させたいタスクフロー名。このパラメータは、タスクフローの作成（API方式 またはコンソール方式）時にユーザーが指定します。具体的な内容は、タスクフロー概要をご参照ください。	String	いいえ
ExpireTime	メディアファイルの''。表記形式はISO 8601規格に準拠します。詳細については、ISO日時表記形式の説明をご参照ください。	String	いいえ
ClassId	カテゴリーID。メディアのカテゴリー管理に使用します。カテゴリー作成 インターフェースによってカテゴリーを作成し、カテゴリーIDを取得することができます。	Integer	いいえ
SourceContext	ソースコンテキスト。ユーザーリクエスト情報のパススルーに使用します。アップロードコールバックインターフェースは、このフィールドの値を戻します。最長250文字。	String	いいえ
StorageRegion	ストレージリージョン。ストレージを予定/希望するリージョンを指定します。このフィールドにはストレージリージョンの英語の略称を入力します。	String	いいえ

アップロードレスポンスクラス `VodUploadResponse`

属性名	属性説明	タイプ

属性名	属性説明	タイプ
FileId	メディアファイルの一意の標識。	String
MediaUrl	メディア再生アドレス。	String
CoverUrl	メディアカバーアドレス。	String
RequestId	一意のリクエストID。リクエストごとに返されます。問題を特定する時はその回のリクエストのRequestIdを提供する必要があります。	String

アップロードメソッド `VodUploadClient.upload(String region, VodUploadRequest request, function callback)`

パラメータ名	パラメータの説明	タイプ	入力必須
region	アクセスポイントリージョン。どのリージョンのVODサーバーにリクエストするかであり、ストレージリージョンとは異なります。具体的な内容は、サポートするリージョンリストをご参照ください。	String	はい
request	アップロードリクエスト。	VodUploadRequest	はい
callback	アップロード完了コールバック関数。	function	はい

アップロード完了コールバック関数 `function(err, data)`

パラメータ名	パラメータの説明	タイプ	入力必須
err	エラー情報。	Exception	はい
data	アップロードレスポンスの結果。	VodUploadResponse	はい

エラーコードリスト

ステータスコード	意味
InternalError	内部エラー。
InvalidParameter.ExpireTime	パラメータ値のエラー：期限。
InvalidParameterValue.CoverType	パラメータ値のエラー：カバーのタイプ。

ステータスコード	意味
InvalidParameterValue.MediaType	パラメータ値のエラー：メディアタイプ。
InvalidParameterValue.SubAppId	パラメータ値のエラー：サブアプリケーションID。
InvalidParameterValue.VodSessionKey	パラメータ値のエラー：VODセッション。
ResourceNotFound	リソースが存在しません。

Go SDK

最終更新日：：2023-10-26 17:31:32

サーバーでのビデオアップロードのシナリオを実現させるために、VODではGo SDKを提供しています。アップロードのフローは、サーバーからのアップロードガイドをご参照ください。

統合方式

go getを使用してインポート

```
go get -u github.com/tencentcloud/tencentcloud-sdk-go
go get -u github.com/tencentyun/cos-go-sdk-v5
go get -u github.com/tencentyun/vod-go-sdk
```

ソースコードパッケージによるインストール

プロジェクトの中にソースコードを直接インポートする必要がある場合は、ソースコードを直接ダウンロードしてプロジェクトにインポートし、使用することができます。

- Githubからアクセス
- クリックしてGo SDKをダウンロード

シンプルビデオアップロード

アップロードオブジェクトの初期化

Tencent Cloud APIキーを使用して、VodUploadClientインスタンスを初期化します。

```
import (
"github.com/tencentyun/vod-go-sdk"
)

client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"
```

アップロードリクエストのオブジェクト作成

```
import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
)

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")
```

アップロードの呼び出し

アップロードメソッドを呼び出し、アクセスポイントリージョンおよびアップロードリクエストを渡します。

```
rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
```

説明：
アップロード方法は、ファイルのサイズに応じて、通常アップロードとマルチパートアップロードが自動的に選択されます。マルチパートアップロードの各手順を気にすることなく、マルチパートアップロードを行うことができます。

高度な機能

カバーの付加

```
package main

import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
)

func main() {
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"
```

```
req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")
req.CoverFilePath = common.StringPtr("/data/video/Wildlife-cover.png")

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
fmt.Println(*rsp.Response.CoverUrl)
}
```

タスクフローの指定

まず、タスクフローテンプレートの作成、およびテンプレートに対する命名を行います。タスクフロー時に、このタスクフローテンプレート名を使用して `Procedure` パラメータを設定すれば、アップロード成功後、タスクフローを自動的に実行することができます。

```
package main

import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
)

func main() {
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")
req.Procedure = common.StringPtr("Your Proceducre Name")

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
}
```

サブアプリケーションのアップロード

サブアプリケーションIDを渡します。アップロード成功後、リソースは具体的なサブアプリケーションにのみ属します。

```
package main

import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
)

func main() {
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")
req.SubAppId = common.Uint64Ptr(101)

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
}
```

ストレージリージョンの指定

コンソールで目標ストレージリージョンがアクティブ化されているか確認します。アクティブ化されていない場合は、アップロードストレージ設定をご参照ください。最後に、 `StorageRegion` の属性によって、ストレージリージョンの 英語の略称を設定します。

```
package main

import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
)
```

```
func main() {
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")
req.StorageRegion = common.StringPtr("ap-chongqing")

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
}
```

パート同時実行数の指定

パート同時実行数は、大きなファイルに対応するもので、複数のパートに分割すると同時にアップロードするものです。パート同時アップロードのメリットは、1個のファイルのアップロードを素早く完了させられることで、SDKはファイルのサイズを基に、通常アップロードとパートアップロードを自動的に選択します。ユーザーは、パートアップロードの各手順に気を遣う必要もなく、パートでのアップロードを実現できます。こうしてファイルの同時パート数は、 `ConcurrentUploadNumber` パラメータによって指定します。

```
package main

import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
)

func main() {
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")
req.ConcurrentUploadNumber = common.Uint64Ptr(5)

rsp, err := client.Upload("ap-guangzhou", req)
```

```
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
}
```

一時的な証明書の使用によるアップロード

一時的な証明書の関連キー情報を渡し、一時的な証明書を使用して身分を検証し、アップロードします。

```
package main

import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
)

func main() {
client := &vod.VodUploadClient{}
client.SecretId = "Credentials TmpSecretId"
client.SecretKey = "Credentials TmpSecretKey"
client.Token = "Credentials Token"

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
}
```

アップロードのプロキシ設定

アップロードのプロキシ設定では、関係するプロトコルおよびデータはいずれもプロキシ経由で処理します。開発者はプロキシを参照しながら、ファイルを自社インターネットからテンセントクラウドにアップロードします。

```
package main
```

```
import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
"net/http"
"net/url"
)

func main() {
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"
proxyUrl, _ := url.Parse("your proxy url")
client.Transport = &http.Transport{
Proxy: http.ProxyURL(proxyUrl),
}

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/Wildlife.mp4")

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
}
```

アダプティブビットレートストリーミング（ABS）ファイルのアップロード

このSDKでアップロードをサポートするABS形式のパッケージにはHLSとDASHがあります。manifest（M3U8またはMPD）で引用するメディアファイルは、必ず相対パスとし（URLおよび絶対パスは不可）、同時にmanifestと同じクラスのディレクトリまたは下の階層のディレクトリに置く必要があります（ `../` は使用不可）。SDKのアップロードインターフェースを呼び出す時に、 `MediaFilePath` パラメータにmanifest を入力すると、SDKが関連するメディアファイルリストを解析し、一緒にアップロードされます。

```
package main

import (
"github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
"github.com/tencentyun/vod-go-sdk"
"fmt"
)
```

```
func main() {
client := &vod.VodUploadClient{}
client.SecretId = "your secretId"
client.SecretKey = "your secretKey"

req := vod.NewVodUploadRequest()
req.MediaFilePath = common.StringPtr("/data/video/prog_index.m3u8")

rsp, err := client.Upload("ap-guangzhou", req)
if err != nil {
fmt.Println(err)
return
}
fmt.Println(*rsp.Response.FileId)
fmt.Println(*rsp.Response.MediaUrl)
fmt.Println(*rsp.Response.CoverUrl)
}
```

インターフェースの説明

アップロードクライアントクラス `VodUploadClient`

属性名	属性説明	タイプ	入力必須
SecretId	Tencent Cloud APIキーID。	String	はい
SecretKey	Tencent Cloud API Key。	String	はい

アップロードリクエストクラス `VodUploadRequest`

属性名	属性説明	タイプ	入力必須
MediaFilePath	アップロード予定のメディアファイルパス。ローカルパスにする必要があります。URLはサポートしていません。	Stringポインタ	はい
SubAppId	VOD サブアプリケーションID。サブアプリケーションの中のリソースにアクセスしたい場合は、このフィールドにサブアプリケーションIDを入力します。アクセスしない場合、この項目は入力不要です。	uint64ポインタ	いいえ

属性名	属性説明	タイプ	入力必須
MediaType	アップロード予定のメディアファイルタイプ。選択可能なタイプの詳細は、メディアアップロードの概要をご参照ください。MediaFilePathに拡張子が付いている場合は入力不要です。	Stringポインタ	いいえ
MediaName	アップロード後のメディアの名前。入力しない場合は、デフォルトでMediaFilePathのファイル名を採用します。	Stringポインタ	いいえ
CoverFilePath	アップロード予定のカバーファイルパス。ローカルパスにする必要があります。URLはサポートしていません。	Stringポインタ	いいえ
CoverType	アップロード予定のカバーファイルタイプ。選択可能なタイプの詳細は、メディアアップロードの概要をご参照ください。CoverFilePathに拡張子が付いている場合は入力不要です。	Stringポインタ	いいえ
Procedure	アップロード後に自動的に実行させたいタスクフロー名。このパラメータは、タスクフローの作成（API方式 またはコンソール方式）時にユーザーが指定します。具体的な内容は、タスクフロー概要をご参照ください。	Stringポインタ	いいえ
ExpireTime	メディアファイルの期限。表記形式はISO 8601規格に準拠します。詳細については、ISO日時表記形式の説明をご参照ください。	Stringポインタ	いいえ
ClassId	カテゴリーID。メディアのカテゴリー管理に使用します。カテゴリー作成 インターフェースによってカテゴリーを作成し、カテゴリーIDを取得することができます。	int64ポインタ	いいえ
SourceContext	ソースコンテキスト。ユーザーリクエスト情報のパススルーに使用します。アップロードコールバックインターフェースは、このフィールドの値を戻します。最長250文字。	Stringポインタ	いいえ
StorageRegion	ストレージリージョン。ストレージを予定/希望するリージョンを指定します。このフィールドにはストレージリージョンの英語の略称を入力します。	Stringポインタ	いいえ

属性名	属性説明	タイプ	入力必須
ConcurrentUploadNumber	パート同時実行数。大きなファイルを対象にパートアップロードする時に有効となります。	Integer	いいえ

アップロードレスポンスクラス `VodUploadResponse`

属性名	属性説明	タイプ
Response	アップロードの結果情報を戻します。	struct
Response.FileId	メディアファイルの一意の標識。	String ポインタ
Response.MediaUrl	メディア再生アドレス。	String ポインタ
Response.CoverUrl	メディアカバーアドレス。	String ポインタ
Response.RequestId	一時的なリクエストID。リクエストごとに返されます。問題を特定する時はその回のリクエストのRequestIdを提供する必要があります。	String ポインタ

アップロードメソッド `VodUploadClient.Upload(region string, request *VodUploadRequest)`

パラメータ名	パラメータの説明	タイプ	入力必須
region	アクセスポイントリージョン。どのリージョンのVODサーバーにリクエストするかであり、ストレージリージョンとは異なります。具体的な内容は、サポートするリージョンリストをご参照ください。	String	はい
request	アップロードリクエスト。	VodUploadRequest ポインタ	はい

エラーコードリスト

ステータスコード	意味

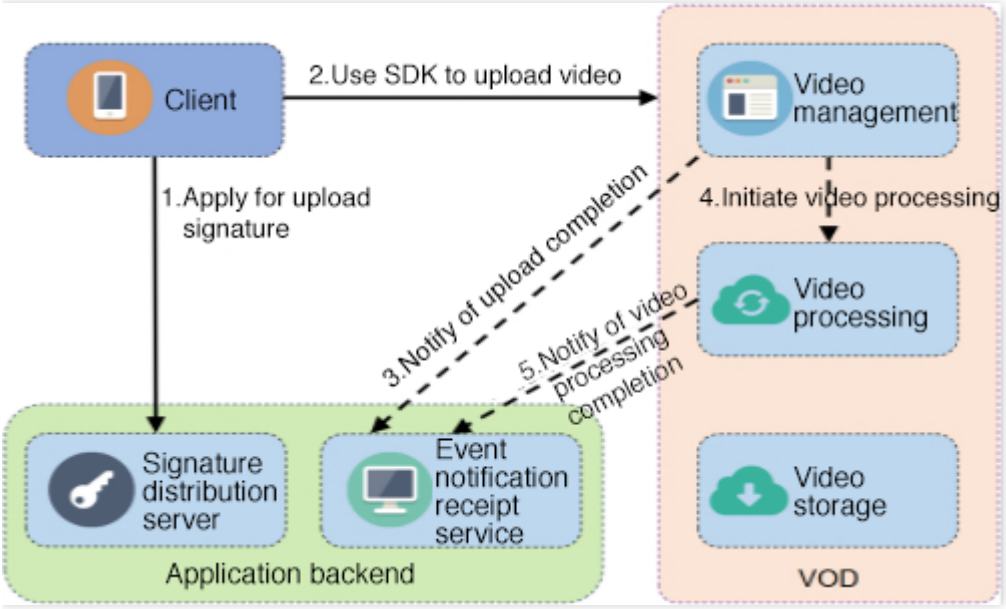
ステータスコード	意味
InternalError	内部エラー。
InvalidParameter.ExpireTime	パラメータ値のエラー：期限。
InvalidParameterValue.CoverType	パラメータ値のエラー：カバーのタイプ。
InvalidParameterValue.MediaType	パラメータ値のエラー：メディアタイプ。
InvalidParameterValue.SubAppId	パラメータ値のエラー：サブアプリケーションID。
InvalidParameterValue.VodSessionKey	パラメータ値のエラー：VODセッション。
ResourceNotFound	リソースがありません。

クライアントからのアップロード
クライアントアップロードガイド

最終更新日：：2023-10-26 17:31:32

操作シナリオ

クライアントからのビデオアップロードとは、Appのエンドユーザーがローカルビデオを VODプラットフォームにアップロードすることです。フローチャートは次のとおりです。ここでは、クライアントを使用してビデオをアップロードする方法をご説明します。



前提条件

1. サービスのアクティブ化

VODサービスをアクティブ化します。詳細については、購入ガイドラインをご参照ください。

2. Tencent Cloud APIキーの取得

サーバーAPIの呼び出しに必要なセキュリティ証明（SecretIdとSecretKey）を取得します。具体的な手順は次のとおりです。

1. コンソールにログインし、【クラウド製品】>【CAM】>【APIキー管理】を選択し、「APIキー管理」ページに進みます。

2. Tencent Cloud APIキーを取得します。キーを作成していない場合は、【キーの作成】をクリックして、SecretIdとSecretKeyのペアを作成することができます。

操作手順

1. アップロード用署名の申請

クライアントは、アップロード署名をAppの署名配布サーバーに申請する必要があります。署名の生成手順については、クライアントからのアップロード署名 をご参照ください。さまざまな言語で署名を生成する例については、以下をご参照ください。

- PHP署名の例
- Java署名の例
- Node.js署名の例
- C#署名の例
- Python署名の例

説明：

- クライアントからのアップロードは、クライアントからVODプラットフォームにビデオファイルを直接アップロードします。Appサーバーで中継する必要がないため、VODはリクエストを開始するクライアントを必ず認証する必要があります。
- 重大なセキュリティの問題を回避するため、Appは最終的なアクセス許可を持つSecretKeyをクライアントに開示しないようにします。クライアントはアップロードを開始する前にアップロード署名を申請する必要があります。

2. SDKを使用したビデオのアップロード

VODは、クライアントからビデオを手軽にアップロードできるよう、顧客アクセスを容易にするマルチプラットフォームのSDKを提供しています。詳細については、以下をご参照ください。

- AndroidのアップロードSDK
- iOSのアップロードSDK
- WebのアップロードSDK

よくあるご質問

- **ビデオのアップロード完了後、どのようにしてトランスコードを自動的に開始しますか。
  クライアントからアップロードするための署名の `procedure` パラメータを介して、ビデオのアップロードが完了した後の処理方法を指定します。詳細については、アップロード時のタスクフロー指定 をご参照ください。
- **Appバックエンドはビデオのアップロード完了通知を受信したとき、どのクライアントがアップロードしたのか、どうすれば特定できますか。**
  クライアントからアップロードするための署名に、 `sourceContext` パラメータを追加して、ユーザーID情報を伝達できます。アップロード完了通知は、このパラメータをAppバックエンドに渡します。詳細については、イベントの通知をご参照ください。

3. イベントの通知

ビデオのアップロードが完了すると、VODはAppバックエンドに イベント通知 - ビデオアップロード完了 を開始します。Appバックエンドは、このイベントを通じてビデオのアップロード動作を認識することができます。イベント通知を受信するには、Appは コンソール - コールバック設定 に移動してイベント通知を有効にする必要があります。イベント通知 - ビデオアップロード完了 には、主に次の情報が含まれています。

- 新しいビデオファイルのFileIdとURL。
- VODは、ビデオアップロード時のパススルーフィールドの指定をサポートしており、イベントが完了すると、パススルーフィールドはAppバックエンドに送信されます。イベント通知には、次のフィールドが含まれています。
  - `SourceType` ：このフィールドはTencent Cloudにより `ServerUpload` に固定されています。アップロードソースがサーバーからのアップロードであることを意味します。
  - `SourceContext` ：ユーザー定義のパススルーフィールドです。署名の配布中にAppバックエンドに指定されます。これは署名の `sourceContext` パラメータに対応します。
- VODでは、ビデオアップロード完了時にビデオ処理を自動的に開始する機能をサポートしています。アップロード時にビデオ処理タスクフローを指定すると、イベント通知の内容にタスクID、すなわちイベント通知の中の `data.procedureTaskId` フィールドが添付されてきます。

詳細については、以下をご参照ください。

- タスク管理とイベント通知
- イベント通知 - ビデオアップロード完了

高度な機能

- **アップロード時のタスクフローの指定**

開発者がビデオのアップロード完了後にビデオ処理タスクフロー（例：トランスコード、スクリーンキャプチャなど）を自動的に開始したい場合は、アップロード署名の生成時に `procedure` パラメータによって実現することができ、パラメータの値をタスクフローテンプレート名にします。VODではタスクフローテンプレートの作成およびテンプレートの命名をサポートし、タスクフローの開始時に、タスクフローテンプレート名によって開始が必要なタスクを示すことができます。

- **アップロード時のストレージリージョンの指定**

  VODがデフォルトで提供するストレージリージョンは、シンガポール に設定されています。その他のエリアに保存したい場合は、コンソール上でご自身でその他ストレージリージョンを追加することができます。詳細については、アップロードストレージ設定をご参照ください。設定完了後、アップロード署名の生成時に `storageRegion` パラメータによって指定でき、パラメータの値をストレージリージョンの英語の略称にします。

- **アップロード時のカバー追加**

  VODでは、アップロードSDKインターフェースに関連するカバーへのパスを入力することで、カバー付きのビデオをアップロードできます。詳細については、以下をご参照ください。
  - AndroidのアップロードSDK
  - iOSのアップロードSDK
  - WebのアップロードSDK

- **ワンタイム署名**

  ビデオのアップロード中に、Appのバックエンドによって配布された署名は、デフォルトで有効期間中、複数回使用できます。Appにビデオアップロードのセキュリティに関する高い要件がある場合は、ワンタイム署名を指定することができます。
  ワンタイム署名の使用方法：Appのバックエンドが署名を配布するときに、パラメータ `oneTimeValid` を1に指定する必要があります。詳細については、クライアントからのアップロード署名をご参照ください。

  > 説明：
  > ワンタイム署名は1回しか使用できないため、この署名方式はより安全にはなりますが、Appで規定外のエラー処理が必要となります。例えば、アップロードにエラーが発生した場合、簡単にSDKを再利用してビデオをアップロードできず、アップロード署名も再度申請する必要があります。

- **中断からの再開**

  ビデオのアップロード中にアップロードが予期せず終了した場合、中断したポイントからファイルを再度アップロードできます。

> 説明：
> 中断からの再開の有効期限は1日です。つまり同じビデオのアップロードが中断された場合、1日以内に再度アップロードすると中断ポイントからそのままアップロードできます。1日を超えるとデフォルトでは、完全なビデオを再度アップロードします。

Appで中断からの再開機能を有効にする方法は、次のとおりです。

- AndroidのアップロードSDK、アップロード時に `enableResume` フィールドをTrueに設定します。
- iOSのアップロードSDK、アップロード時に `enableResume` フィールドをTrueに設定します。
- WebのアップロードSDK、中断からの再開は組み込まれており、ユーザーによる操作は不要です。

- **アップロードの一次停止/再開/キャンセル**

VOD SDKでは、ビデオのアップロード中に、アップロードの一時停止、再開またはキャンセルをすることができます。詳細については、以下をご参照ください。

- AndroidのアップロードSDK
- iOSのアップロードSDK
- WebのアップロードSDK

クライアントからのアップロードの高速化

最終更新日：：2023-10-26 17:31:32

クライアントのアップロードアクセラレーションにより、より高品質なアップロードサービスをお客様に提供することができます。この機能は、Tencent Cloudがグローバルにデプロイするアクセラレーションネットワークをベースとして、ユーザーのリクエストに応じて最適なアクセスポイントと最適なリンクをインテリジェントに選択し、アップロードの速度と成功率を引き上げます。また、QUICプロトコルによるデータ伝送もサポートし、データ伝送の効率化と脆弱なネットワーク環境下での安定性を向上させます。

アップロードの品質に影響する主な要因

長距離伝送

オンデマンドは世界中多くの地域にストレージセンターを設けており、お客様はオプションでこのストレージセンターを利用して、アップロード時にお客様の所在地の近くでストレージすることができます。詳細については、近くでのアップロードをご参照ください。しかしこのサービスを利用しても、エンドユーザーがストレージセンターから遠すぎて、地域や海を越えてアップロードする必要があるビジネスシナリオを持つお客様もいらっしゃいます。長距離のデータアップロードは通常、ネットワークリンクが長くなり、伝送遅延が大きくなることを意味します。さらに中間リンクの1つでネットワークジッターやパケットロスが発生すると、リンク全体のアップロード速度や成功率は低下します。

脆弱なネットワークの問題

脆弱なネットワークとは、簡単に言えば、遅延やパケットロス率の多いネットワークなど、質の悪いネットワーク環境のことです。現在、モバイルネットワークが非常に普及しており、アップロードに占めるモバイル端末のお客様の割合が非常に高くなっています。モバイルネットワークでは、ユーザーが基地局からの電波が届きにくい場所にいる場合や、ネットワーク接続された機器が動作中に頻繁にネットワークを切り替える場合など、ネットワークの脆弱性が問題となることがよくあります。アップロードの品質を向上させるには、脆弱なネットワーク環境下でいかに安定したデータ伝送を維持するかが大きな課題となります。

非効率なネットワークプロトコル

オンデマンドクライアントがアップロードするファイルの多くは、データ量の大きいビデオファイルです。現在、アップロードに最も頻繁に使用されるネットワークプロトコルは、依然としてHTTP 1.1です。このプロトコルはシリアルモデルをベースとしているため、HOLブロッキングなどの問題があり、大規模データ伝送シナリオでは性能ボトルネックになりやすくなっています。

クライアントからのアップロードの高速化方式

グローバルリンクアクセラレーション、高信頼性チャネル

Video on Demandは、長距離伝送シナリオにおける長いネットワークリンクによるアップロードの品質低下の問題に対応するため、Tencent Cloudのグローバルなアクセラレーションネットワークとエッジノードをベースとしたデータアップロード用の一連のグローバルリンクのアクセラレーションチャンネルをお客様に提供しています。Tencent Cloudのインテリジェントなグローバルトラフィック管理プラットフォームを介して、ユーザーのアップロードリクエストはユーザーに最も近いエッジノードに送信され、ユーザーのデータを近くで受信することができます。さらにTencent Cloudが長年磨き上げてきたアクセラレーションネットワークを通じて最適なリンクを選択し、ストレージセンターにデータを伝送します。

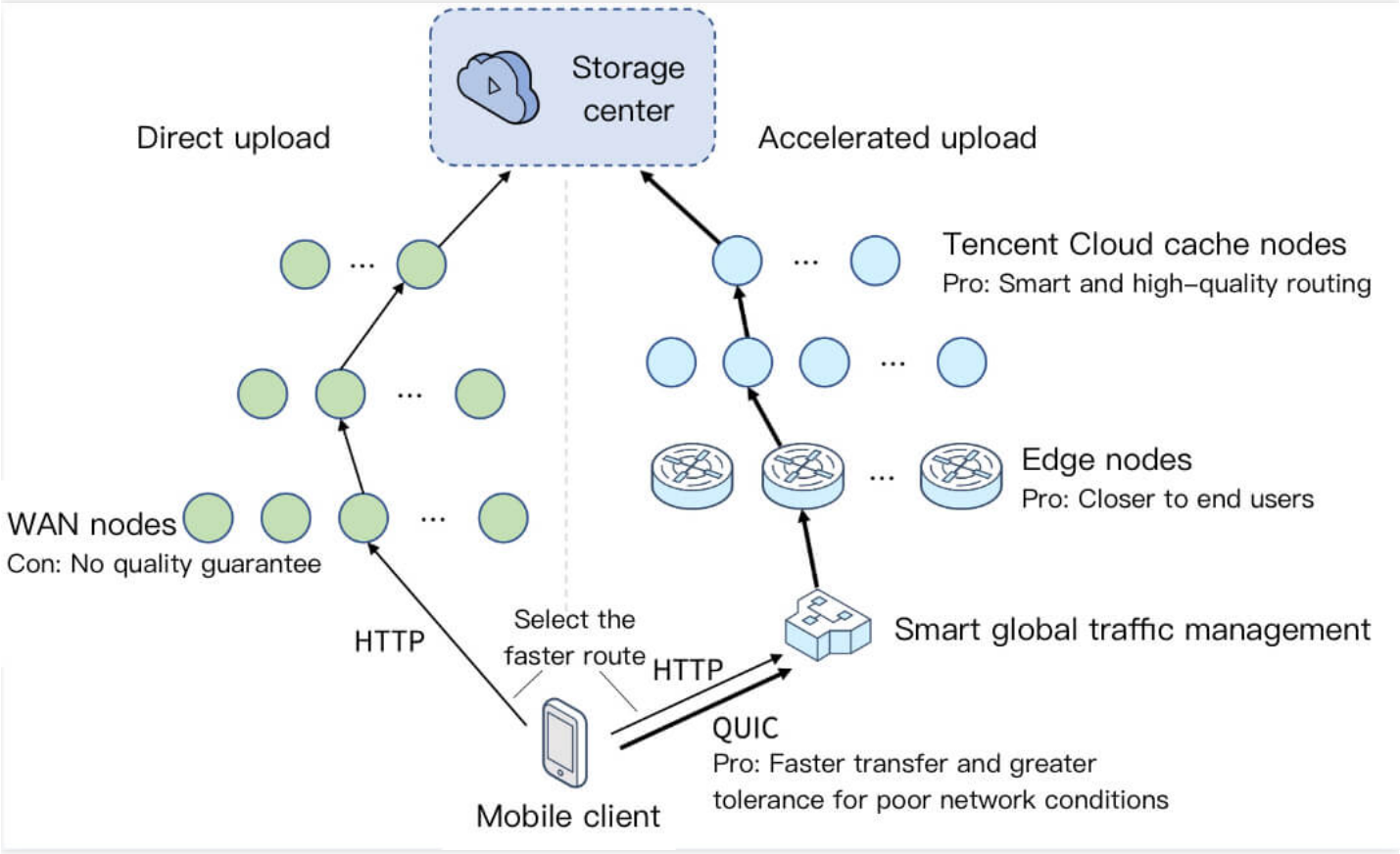
より高速に、より安定したQUICプロトコル

VODでは、脆弱なネットワーク環境や非効率なネットワークプロトコルに対応するため、現在の標準的なHTTP3プロトコルのベースとなっているUDPベースの低遅延・高信頼通信プロトコルであるQUICプロトコルに対応したクライアントアップロードを実装しています。QUICは、0-RTTの接続の確立とHOLブロッキングのない多重化をサポートしています。ネットワーク帯域幅を実際のデータ伝送に最大限活用できるため、パケットロス率やネットワーク遅延の大きい脆弱なネットワーク環境でも高品質のデータ通信を実現できます。また、QUICは、モバイル端末が頻繁にネットワークを切り替えるようなシナリオでも、接続移行をサポートしてスムーズな移行と中断のないネットワークを確保します。

インテリジェントな選択、手軽な使用

Video on Demandは、お客様がコンソールで関連機能を有効にするだけで、簡単に使用できるアップロードアクセラレーション方式を提供します。SDKを使用してアップロードする場合、通常チャネルとアクセラレーションチャネルの間でインテリジェントに速度の競争と優先順位付けを行い、データアップロードにQUICプロトコルを

使用するかどうかを自動的に検出して決定します。



使用方法

以下の2つの簡単なステップにより、クライアントのアップロードアクセラレーション機能がオンになります。

1. クライアントアップロードのアクセラレーションについては、コンソール操作のガイドをご参照のうえ、「グローバルリンクアクセラレーション」をオンにし、必要に応じて「QUIC伝送」をオンにしてください。
2. AndroidとiOSプラットフォームでは、Appの起動時にプリアップロードが呼び出されるようにする必要があります。「QUIC伝送」をオンにする場合は、Androidプラットフォームは SDKバージョン9.6以上、iOSプラットフォームはSDKバージョン10.4以上を使用する必要があります

> 説明：
>
> - Android、iOSでSDKをアップロードする場合、アップロードアクセラレーションとQUIC伝送が同時にサポートされます。
> - Web端末、ミニプログラム端末でSDKをアップロードする場合、アップロードアクセラレーションのみをサポートしており、今のところQUIC伝送はサポートしていません。

料金関連

クライアントのアップロードアクセラレーションを使用する場合、以下のような料金が発生します。

- グローバルリンクアクセラレーション料金：グローバルリンクアクセラレーション使用時に発生するアップロードアクセラレーショントラフィック料金
- QUIC伝送料金：QUIC伝送を使用する際に発生するアップロードトラフィックのアクセラレーション料金

上記の料金の具体的な価格については、購入ガイドをご参照ください。

クライアントアップロード署名

最終更新日：：2023-10-26 17:31:32

クライアントでアップロードを開始する前に、アップロード操作中に実行する必要があるアップロード署名を、Appの署名配布サーバーに申請する必要があります。これによって、VODはクライアントにアップロードの権限が付与されているかを確認できます。

署名の生成手順

1. **TencentCloud APIの取得**
   サーバーAPIの呼び出しに必要なセキュリティ証明（SecretIdとSecretKey）を取得します。具体的な手順は次のとおりです。
   i. コンソールにログインし、【クラウドサービス】>【CAM】>【APIキー管理】を選択し、「APIキー管理」ページに進みます。
   ii. Tencent Cloud APIキーを取得します。キーを作成していない場合は、【キーの作成】をクリックして、SecretIdとSecretKeyのペアを作成することができます。
2. **平文文字列originalの結合**
   URL QueryStringのフォーマット要件に従って、平文署名文字列originalを結合します。そのフォーマットは次のとおりです。

```
secretId=[secretId]¤tTimeStamp=[currentTimeStamp]&expireTime=[expireTime]&random=[random]
```

```
- 上記のoriginalにおける`[secretId]`、`[currentTimeStamp]`、`[expireTime]`、`[random]`は、特定のパラメータ値にご自身で置き換える必要があります。
- originalには、`secretId`、`currentTimeStamp`、`expireTime`、`random`の4つの必須パラメータが含まれており、オプションのパラメータならいくつでも含めることができます。詳細については、[署名パラメータ](#p2) をご参照ください。
- パラメータ値は、必ずUrlEncodeを行う必要があります。行わない場合、QueryStringの解析が失敗する恐れがあります。
```

3. **平文文字列を最終的な署名に変換します**（Javaコードの一部を例とします）

i. 取得したSecretKeyを使用して、平文文字列originalをHMAC-SHA1 で暗号化し、signatureTmpを取得します。

```
Mac mac = Mac.getInstance("HmacSHA1");
SecretKeySpec secretKey = new SecretKeySpec(this.secretKey.getBytes("UTF-8"),
mac.getAlgorithm());
mac.init(secretKey);
byte[] signatureTmp = mac.doFinal(original.getBytes("UTF-8"));
```

signatureTmpは、UTF-8でエンコードされ、HMAC-SHA1で暗号化されたバイト配列です。

ii. UTF-8を使用して平文文字列originalをバイト配列にエンコードし、次に配列をsignatureTmpでマージし、最後にマージされた結果をBase64 にエンコードして、次のような署名signatureを取得します。

```
String signature = base64Encode(byteMerger(signatureTmp, original.getBytes("u
tf8")));
```

byteMergerとbase64Encodeは、それぞれ配列のマージとBase64エンコードのメソッドです。詳細については、Java署名サンプルコード をご参照ください。

署名生成の例

VODでは、参考と検証のために、**署名生成サンプルコード**と署名ツールも提供します。

- クライアントからのアップロード - 署名生成サンプルコード
- オンデマンドクライアントからのアップロード - 署名生成ツール
- オンデマンドクライアントからのアップロード - 署名検証ツール

署名パラメータの説明

パラメータ名	記入必須	タイプ	説明
secretId	はい	String	Tencent Cloud APIキーのSecretId。取得方法については、クライアントからのアップロードガイド - Tencent Cloud APIキーの取得 をご参照ください。
currentTimeStamp	はい	Integer	現在のUnixタイムスタンプ。
expireTime	はい	Integer	署名の有効期限が切れたUnixタイムスタンプ。`expireTime = currentTimeStamp + 署名の有効期間` 署名の有効期間の最大値は7776000、つまり90日です。
random	はい	Integer	平文署名文字列を作成するためのパラメータ。10進数で、最大値は `xxxxx` （32ビットの符号なしのバイナリー最大値）です。
classId	いいえ	Integer	ビデオファイルのカテゴリー。デフォルトは0です。
procedure	いいえ	String	後続のビデオタスク操作、すなわち、ビデオのアップロードが完了した後、タスクフロー操作が自動的に開始されます。このパラメータ値はタスクフローテンプレート名です。VODは、タスクフローテンプレートの作成 とテンプレートの命名をサポートします。
taskPriority	いいえ	Integer	後続のビデオタスクの優先度（procedureが指定されている場合にのみ有効）。値の範囲は[-10，10]、デフォルトは0です。
taskNotifyMode	いいえ	String	タスクフロー状態の変更通知モード（procedureが指定されている場合にのみ有効）。 • Finish：タスクフローが完全に実行された場合にのみ、イベント通知が開始されます。 • Change：タスクフロー内の各サブタスクのステータスが変更された場合に限り、イベント通知が行われます。 • None：タスクフローのコールバックは受け入れられません。 デフォルトはFinishです。
sourceContext	いいえ	String	ソースコンテキストは、ユーザーリクエスト情報を渡すために使用されます。アップロード完了コールバック は、このフィールドの値を返します。最大250文字を入力することが可能です。

パラメータ名	記入必須	タイプ	説明
oneTimeValid	いいえ	Integer	署名がワンタイム署名かどうかについては、クライアントからのアップロードガイド - ワンタイム署名 をご参照ください。 デフォルトは0で、ワンタイム署名が無効であることを表し、1はワンタイム署名が有効であることを表します。 関連するエラーコードについては、ワンタイム署名の説明 をご参照ください。
vodSubAppId	いいえ	Integer	サブアプリケーション ID。このパラメータを入力しない、0を入力した、またはTencent Cloud AppIdを入力した場合、操作するサブアプリケーションが「メインアプリケーション」となります。
sessionContext	いいえ	String	セッションコンテキストは、ユーザーリクエスト情報を渡すために使用されます。procedureパラメータが指定されている場合、タスクフロー状態の変更コールバック は、フィールド値を返します。最大1000文字を含めることができます。
storageRegion	いいえ	String	ストレージリージョンを指定します。コンソールでストレージリージョンを追加することができます。詳細については、アップロードストレージ設定 をご参照ください。このフィールドには、ストレージリージョンの英語の略称 を入力してください。

ワンタイム署名の説明

- ワンタイム署名を有効にした場合、署名サーバーは、ユーザーに配布される署名が毎回異なることを確認する必要があります（同じタイミングで配布される署名 `random` が繰り返されないようにするなど）。確認しない場合、署名重複によるエラーが発生します。
- 署名エラーによってアップロードに失敗し、再試行する場合は、新しい署名を取得する必要があります。
- AndroidおよびJava SDKが原因の署名エラーのエラーコードは `1001` です。

署名生成の例

最終更新日：：2023-10-26 17:31:32

PHP署名の例

```
<?php
// AppのTencentCloud APIキーを決定します
$secret_id = "XXXXXXXXXXXXXXXXXXX";
$secret_key = "AAAAAAAAAAAAAAAAAAA";

// 署名の現在の時刻と有効期間を決定します
$current = time();
$expired = $current + 86400; // 署名の有効期間：1日

// パラメータリストにパラメータを入力します
$arg_list = array(
"secretId" => $secret_id,
"currentTimeStamp" => $current,
"expireTime" => $expired,
"random" => rand());

// 署名計算
$original = http_build_query($arg_list);
$signature = base64_encode(hash_hmac('SHA1', $original, $secret_key, true).$origi
nal);

echo $signature;
echo "\n";
?>
```

Java署名の例

```
import java.util.Random;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import sun.misc.BASE64Encoder;

// 署名ツール類
class Signature {
private String secretId;
```

```
private String secretKey;
private long currentTime;
private int random;
private int signValidDuration;

private static final String HMAC_ALGORITHM = "HmacSHA1"; //署名アルゴリズム
private static final String CONTENT_CHARSET = "UTF-8";

public static byte[] byteMerger(byte[] byte1, byte[] byte2) {
byte[] byte3 = new byte[byte1.length + byte2.length];
System.arraycopy(byte1, 0, byte3, 0, byte1.length);
System.arraycopy(byte2, 0, byte3, byte1.length, byte2.length);
return byte3;
}

// 署名を取得します
public String getUploadSignature() throws Exception {
String strSign = "";
String contextStr = "";

// オリジナルのパラメータ文字列を生成します
long endTime = (currentTime + signValidDuration);
contextStr += "secretId=" + java.net.URLEncoder.encode(secretId, "utf8");
contextStr += "¤tTimeStamp=" + currentTime;
contextStr += "&expireTime=" + endTime;
contextStr += "&random=" + random;

try {
Mac mac = Mac.getInstance(HMAC_ALGORITHM);
SecretKeySpec secretKey = new SecretKeySpec(this.secretKey.getBytes(CONTENT_CHARS
ET), mac.getAlgorithm());
mac.init(secretKey);

byte[] hash = mac.doFinal(contextStr.getBytes(CONTENT_CHARSET));
byte[] sigBuf = byteMerger(hash, contextStr.getBytes("utf8"));
strSign = base64Encode(sigBuf);
strSign = strSign.replace(" ", "").replace("\n", "").replace("\r", "");
} catch (Exception e) {
throw e;
}
return strSign;
}

private String base64Encode(byte[] buffer) {
BASE64Encoder encoder = new BASE64Encoder();
return encoder.encode(buffer);
}
```

```
public void setSecretId(String secretId) {
this.secretId = secretId;
}

public void setSecretKey(String secretKey) {
this.secretKey = secretKey;
}

public void setCurrentTime(long currentTime) {
this.currentTime = currentTime;
}

public void setRandom(int random) {
this.random = random;
}

public void setSignValidDuration(int signValidDuration) {
this.signValidDuration = signValidDuration;
}
}

public class Test {
public static void main(String[] args) {
Signature sign = new Signature();
// AppのTencentCloud APIキーを設定します
sign.setSecretId("APIキーのSecret Id");
sign.setSecretKey("APIキーのSecret Key");
sign.setCurrentTime(System.currentTimeMillis() / 1000);
sign.setRandom(new Random().nextInt(java.lang.Integer.MAX_VALUE));
sign.setSignValidDuration(3600 * 24 * 2); // 署名の有効期間：2日

try {
String signature = sign.getUploadSignature();
System.out.println("signature : " + signature);
} catch (Exception e) {
System.out.print("署名の取得に失敗しました");
e.printStackTrace();
}
}
}
```

Java1.9以降のバージョンでは、`sun.misc.BASE64Encoder` に関連するパッケージが除去されており、`base64Encode` メソッドの対応する実装を `java.util.Base64` で置き換えることができます。詳細については、次のコードをご参照ください。

```
import java.util.Base64;

private String base64Encode(byte[] buffer) {
Base64.Encoder encoder = Base64.getEncoder();
return encoder.encodeToString(buffer);
}
```

Node.js署名の例

```
var querystring = require("querystring");
var crypto = require('crypto');

// appのTencentCloud APIキーを決定します
var secret_id = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
var secret_key = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA";

// 署名の現在の時刻と有効期間を決定します
var current = parseInt((new Date()).getTime() / 1000)
var expired = current + 86400; // 署名の有効期間：1日

// パラメータリストにパラメータを入力します
var arg_list = {
secretId : secret_id,
currentTimeStamp : current,
expireTime : expired,
random : Math.round(Math.random() * Math.pow(2, 32))
}

// 署名計算
var orignal = querystring.stringify(arg_list);
var orignal_buffer = new Buffer(orignal, "utf8");

var hmac = crypto.createHmac("sha1", secret_key);
var hmac_buffer = hmac.update(orignal_buffer).digest();

var signature = Buffer.concat([hmac_buffer, orignal_buffer]).toString("base64");

console.log(signature);
```

C# 署名の例

```
using System;
using System.Security.Cryptography;
using System.Text;
using System.Threading;

class Signature
{
public string m_strSecId;
public string m_strSecKey;
public int m_iRandom;
public long m_qwNowTime;
public int m_iSignValidDuration;
public static long GetIntTimeStamp()
{
TimeSpan ts = DateTime.UtcNow - new DateTime(1970, 1, 1);
return Convert.ToInt64(ts.TotalSeconds);
}
private byte[] hash_hmac_byte(string signatureString, string secretKey)
{
var enc = Encoding.UTF8; HMACSHA1 hmac = new HMACSHA1(enc.GetBytes(secretKey));
hmac.Initialize();
byte[] buffer = enc.GetBytes(signatureString);
return hmac.ComputeHash(buffer);
}
public string GetUploadSignature()
{
string strContent = "";
strContent += ("secretId=" + Uri.EscapeDataString((m_strSecId)));
strContent += ("¤tTimeStamp=" + m_qwNowTime);
strContent += ("&expireTime=" + (m_qwNowTime + m_iSignValidDuration));
strContent += ("&random=" + m_iRandom);

byte[] bytesSign = hash_hmac_byte(strContent, m_strSecKey);
byte[] byteContent = System.Text.Encoding.Default.GetBytes(strContent);
byte[] nCon = new byte[bytesSign.Length + byteContent.Length];
bytesSign.CopyTo(nCon, 0);
byteContent.CopyTo(nCon, bytesSign.Length);
return Convert.ToBase64String(nCon);
}
}
class Program
{
static void Main(string[] args)
{
Signature sign = new Signature();
sign.m_strSecId = "APIキーのSecret Id";
```

```
sign.m_strSecKey = "APIキーのSecret Key";
sign.m_qwNowTime = Signature.GetIntTimeStamp();
sign.m_iRandom = new Random().Next(0, 1000000);
sign.m_iSignValidDuration = 3600 * 24 * 2;

Console.WriteLine(sign.GetUploadSignature());
}
}
```

Python署名の例

```
#!/usr/local/bin/python3
#coding=utf-8

import time
import random
import hmac
import hashlib
import base64

SecretId = 'IamSecretId'
SecretKey = 'IamSecretKey'
#TimeStamp = int(time.time())
TimeStamp = 1571215095
ExpireTime = TimeStamp + 86400 * 365 * 10
#Random = random.randint(0, 999999)
Random = 220625

Original = "secretId=" + SecretId + "¤tTimeStamp=" + str(TimeStamp) + "&expireTim
e=" + str(ExpireTime) + "&random=" + str(Random)

Hmac = hmac.new(bytes(SecretKey, 'utf-8'), bytes(Original, 'utf-8'), hashlib.sha
1)
Sha1 = Hmac.digest()
Signature = bytes(Sha1) + bytes(Original, 'utf-8')
Signature2 = base64.b64encode(Signature)

#return str(signature2, 'UTF-8')

print("Original: ", Original)
print("HMAC-SHA1: ", Sha1)
print("Signature before BASE64: ", Signature)
print("Signature after BASE64: ", str(Signature2))
```

Go署名の例

```
package main

import (
"crypto/hmac"
"crypto/sha1"
"encoding/base64"
"fmt"
"math/rand"
"strconv"
"time"
)

func generateHmacSHA1(secretToken, payloadBody string) []byte {
mac := hmac.New(sha1.New, []byte(secretToken))
sha1.New()
mac.Write([]byte(payloadBody))
return mac.Sum(nil)
}

func main() {
rand.Seed(time.Now().Unix())
secretId := "IamSecretId"
secretKey := "IamSecretKey"
// timestamp := time.Now().Unix()
timestamp := int64(1571215095)
expireTime := timestamp + 86400*365*10
timestampStr := strconv.FormatInt(timestamp, 10)
expireTimeStr := strconv.FormatInt(expireTime, 10)

random := 220625
randomStr := strconv.Itoa(random)
original := "secretId=" + secretId + "¤tTimeStamp=" + timestampStr + "&expireTime
=" + expireTimeStr + "&random=" + randomStr
signature := generateHmacSHA1(secretKey, original)
signature = append(signature, []byte(original)...)
signatureB64 := base64.StdEncoding.EncodeToString(signature)
fmt.Println(signatureB64)
}
```

WebアップロードSDK

最終更新日：：2023-10-26 17:31:32

VODは、オーディオとビデオをブラウザからアップロードするシナリオ向けに、Webアップロード用SDKを提供しています。SDKソースコードが必要な場合は、SDKソースコード にアクセスしてください。

シンプルビデオアップロード

SDKのインポート

scriptのインポート方法

webpackを使用しない場合は、scriptタグ方式によってインポートすることができます。この方法では、グローバル変数 `TcVod` が公開されます。scriptをインポートするには、次の2つの方法があります。

- **ローカルへのダウンロード**
  SDKソースコードをローカルにダウンロードし、次のようにインポートします。

```
<script src="./vod-js-sdk-v6.js"></script>
```

> 説明：
> インポートパスはローカル保存先のパスにご自身で変更してください。

- **CDNリソースの使用**
  CDNリソースを使用すると、次の方法で直接インポートできます。

```
<script src="https://cdn-go.cn/cdn/vod-js-sdk-v6/latest/vod-js-sdk-v6.js"></scr
ipt>
```

ここをクリックして、scriptによってインポートされたDemoをご確認ください。ここをクリックして、Demoソースコードをご確認ください。

npmのインポート方法

webpackを使用する場合（VueやReactなど）は、npmを介してインポートできます。

```
// npm install vod-js-sdk-v6を実行した後に、importを実行してページに直接インポートします。
import TcVod from 'vod-js-sdk-v6'
```

ここをクリック して、npmによってインポートされたDemoソースコードをご確認ください。

注意：
SDKはPromiseに依存しています。低いバージョンのブラウザではご自身でインポートしてください。

アップロード署名を取得するための関数の定義

```
function getSignature() {
return axios.post(url).then(function (response) {
return response.data.signature;
})
};
```

説明：

- `url` は、署名配布サービスのURLです。その他の関連情報については、クライアントからのアップロードガイド をご参照ください。
- `signature` の計算ルールについては、クライアントからのアップロード署名をご参照ください。
- **サブアプリケーション**、**ビデオファイルカテゴリー**、**タスクフロー**などの情報はすべてアップロード署名に含まれています。その他の関連情報については、署名パラメータの説明をご参照ください。

ビデオアップロードの例

```
// importによってインポートした場合、new TcVod(opts) を実行します
// new TcVod.default(opts) は、scriptによってインポートされます
const tcVod = new TcVod.default({
getSignature: getSignature // 前文で説明したアップロード署名を取得する関数です
})

const uploader = tcVod.upload({
mediaFile: mediaFile, // Fileタイプのメディアファイル（ビデオ、オーディオまたは画像）
})
uploader.on('media_progress', function(info) {
console.log(info.percent) // 進行状況
})

// コールバック結果の説明
// type doneResult = {
// fileId: string,
```

```
// video: {
// url: string
// },
// cover: {
// url: string
// }
// }
uploader.done().then(function (doneResult) {
// deal with doneResult
}).catch(function (err) {
// deal with error
})
```

説明：

- `new TcVod(opts)` のoptsは、このインターフェースに関連するパラメータを指します。 詳細については、TcVodインターフェースの説明 をご参照ください。
- アップロード方法は、ファイルのサイズに応じて、通常アップロードとマルチパートアップロードが自動的に選択されます。マルチパートアップロードの各手順を気にすることなく、マルチパートアップロードを行うことができます。
- 指定されたサブアプリケーションにアップロードする必要がある場合は、サブアプリケーションシステム-クライアントからのアップロードをご参照ください。

高度な機能

ビデオとカバーの同時アップロード

```
const uploader = tcVod.upload({
mediaFile: mediaFile,
coverFile: coverFile,
})

uploader.done().then(function (doneResult) {
// deal with doneResult
})
```

アップロードの進行状況の取得

SDKは、現在のアップロード進行状況をコールバック形式で表示します。

```
const uploader = tcVod.upload({
mediaFile: mediaFile,
coverFile: coverFile,
})
// ビデオのアップロードが完了したとき
uploader.on('media_upload', function(info) {
uploaderInfo.isVideoUploadSuccess = true;
})
// ビデオのアップロード進行状況
uploader.on('media_progress', function(info) {
uploaderInfo.progress = info.percent;
})
// カバーをアップロードしたとき
uploader.on('cover_upload', function(info) {
uploaderInfo.isCoverUploadSuccess = true;
})
// カバーのアップロード進行状況
uploader.on('cover_progress', function(info) {
uploaderInfo.coverProgress = info.percent;
})

uploader.done().then(function (doneResult) {
// deal with doneResult
})
```

`xxx_upload` と `xxx_progress` のコールバック値については、マルチパートアップロード/タスクのコピー操作をご参照ください。

アップロードのキャンセル

SDKは、進行中のビデオまたはカバーのアップロードのキャンセルをサポートしています。

```
const uploader = tcVod.upload({
mediaFile: mediaFile,
coverFile: coverFile,
})

uploader.cancel()
```

中断からの再開

SDKでは、自動的にアップロードを再開できます。追加の操作は必要ありません。アップロードが予期せず終了した場合（ブラウザが閉じた、ネットワークが中断したなど）、中断したポイントから再度アップロードを続行

できるため、アップロードを繰り返す時間が短縮されます。

インターフェースの説明

TcVod

パラメータ名	入力必須	タイプ	パラメータの説明
getSignature	はい	Function	アップロード署名を取得するための関数。
appId	いいえ	number	入力後、組み込みの統計レポートに自動的に反映されます。
reportId	いいえ	number	入力後、組み込みの統計レポートに自動的に反映されます。

TcVod.upload

パラメータ名	入力必須	タイプ	パラメータの説明
mediaFile	いいえ	File	メディアファイル（ビデオ、オーディオまたは画像）。
coverFile	いいえ	File	カバーファイル。
mediaName	いいえ	string	メディアファイルのメタデータを上書きするファイル名。
fileId	いいえ	string	カバーを変更するときに渡されます。
reportId	いいえ	number	入力後、組み込みの統計レポートに自動的に反映されます。コンストラクタの設定を上書きします。
fileParallelLimit	いいえ	number	同じインスタンスでアップロードされたファイルの並列処理の数。デフォルト値は3です。
chunkParallelLimit	いいえ	number	同じアップロードファイルのマルチパートの並列処理の数。デフォルト値は6です。
chunkRetryTimes	いいえ	number	マルチパートアップロードの場合、エラーの再試行回数。デフォルト値は2です（最初に追加すると、リクエストは合計3回になります）
chunkSize	いいえ	number	マルチパートアップロードの場合、パーティションあたりのバイトのサイズ。デフォルト値は8388608 (8MB)です
progressInterval	いいえ	number	アップロード進捗のコールバックメソッドonProgressのコールバック頻度。単位：ms、デフォルト値は1000です

イベント

イベント名	入力必須	イベントの説明
media_upload	いいえ	メディアファイルのアップロードが成功したとき。
cover_upload	いいえ	カバーのアップロードが成功したとき。
media_progress	いいえ	メディアファイルのアップロードの進行状況。
cover_progress	いいえ	カバーファイルのアップロードの進行状況。

よくあるご質問

1. **Fileオブジェクトを取得する方法とは**
   `type` が `file` 型の `input` タグを使用すると、 `File` オブジェクトを取得できます。
2. **アップロードするファイルのサイズに制限はありますか？**
   最大60GBをサポートしています。
3. **SDKがサポートしているブラウザのバージョンとは？**
   Chrome、Firefoxなどは、HTML5の主流ブラウザををサポートします。IEは、IE10以降のバージョンをサポートします。
4. **アップロードの一時停止や再開などの機能を実装する方法とは？**
   SDKの基盤層には中断からの再開機能が自動的に実装されているため、一時停止とは、本質的には `uploader.cancel()` メソッドの呼び出しとなります。同様に、一時停止後のアップロード再開も、最初の `tcVod.upload` メソッドを呼び出すことで実行されます。違いは、アップロードが再開されるときに呼び出されるメソッドのパラメータは、以前にキャッシュされたパラメータでなければならないという点です（例えば、グローバル変数を使用してアップロードの開始時にパラメータを保存して、アップロードの完了後にそれをクリアすることができます）。
5. **Web端末のアップロードSDKは、https: ドメイン名を使用してアップロードしますか？**
   サポートできます。デフォルトでは、現在のページのドメイン名がhttp:であるとWeb端末が判断した場合、http: ドメイン名を使用してアップロードします。ドメイン名がhttp:ではないと判断した場合、https: ドメイン名を使用してアップロードします。

AndroidのアップロードSDK

最終更新日：：2023-10-26 17:31:32

VODは、Androidプラットフォームでビデオをアップロードするシナリオ向けに、AndroidアップロードSDKを提供しています。アップロードのフローについては、クライアントからのアップロードガイド をご参照ください。

ソースコードのダウンロード

1. AndroidアップロードDemoとソースコードをクリックしてダウンロードします。
2. ダウンロードした圧縮パッケージを解凍すると、Demoディレクトリが表示されます。アップロードソースコードは、 `Demo/app/src/main/java/com/tencent/ugcupload/demo/videoupload` ディレクトリにあります。

アップロードライブラリとソースコードの統合

1. ソースコードディレクトリ `Demo/app/src/main/java/com/tencent/ugcupload/demo/videoupload` をプロジェクトディレクトリにコピーして、package名を手動で変更する必要があります。
2. `Demo/app/build.gradle` を参照して、プロジェクトに依存関係を追加します。

```
implementation 'com.qcloud.cos:cos-android-nobeacon:5.8.5'
implementation 'com.qcloud.cos:quic:1.5.38'
```

説明：
手動統合 を参照して、対応するバージョンの依存ライブラリを統合することもできます。

3. ビデオアップロードを使用するには、ネットワークとストレージ関連のアクセス許可が必要です。 `AndroidManifest.xml` に次の許可ステートメントを追加することができます。

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<receiver android:name=".videoupload.impl.TVCNetWorkStateReceiver">
```

```
<intent-filter>
<!--ネットワーク変更を検出するaction-->
<action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
<category android:name="android.intent.category.DEFAULT" />
</intent-filter>
</receiver>
```

シンプルビデオアップロード

アップロードオブジェクトの初期化

```
TXUGCPublish mVideoPublish = new TXUGCPublish(this.getApplicationContext(), "inde
pendence_android")
```

アップロードオブジェクトのコールバックの設定

```
mVideoPublish.setListener(new TXUGCPublishTypeDef.ITXVideoPublishListener() {
@Override
public void onPublishProgress(long uploadBytes, long totalBytes) {
mProgress.setProgress((int) (100*uploadBytes/totalBytes));
}

@Override
public void onPublishComplete(TXUGCPublishTypeDef.TXPublishResult result) {
mResultMsg.setText(result.retCode + " Msg:" + (result.retCode == 0 ? result.video
URL : result.descMsg));
}
});
```

アップロードパラメータの作成

```
TXUGCPublishTypeDef.TXPublishParam param = new TXUGCPublishTypeDef.TXPublishParam
();

param.signature = "xxx";
param.videoPath = "xxx";
```

`signature` の計算ルールについては、クライアントからのアップロード署名 をご参照ください。

アップロードの呼び出し

```
int publishCode = mVideoPublish.publishVideo(param);
```

画像のシンプルアップロード

アップロードオブジェクトの初期化

```
TXUGCPublish mVideoPublish = new TXUGCPublish(this.getApplicationContext(), "inde
pendence_android")
```

アップロードオブジェクトのコールバックの設定

```
mVideoPublish.setListener(new TXUGCPublishTypeDef.ITXMediaPublishListener() {
@Override
public void onMediaPublishProgress(long uploadBytes, long totalBytes) {
mProgress.setProgress((int) (100*uploadBytes/totalBytes));
}
@Override
public void onMediaPublishComplete(TXUGCPublishTypeDef.TXMediaPublishResult media
Result) {
mResultMsg.setText(result.retCode + " Msg:" + (result.retCode == 0 ? result.video
URL : result.descMsg));
}
});
```

アップロードパラメータの作成

```
TXUGCPublishTypeDef.TXMediaPublishParam param = new TXUGCPublishTypeDef.TXMediaPu
blishParam();

param.signature = "xxx";
param.mediaPath = "xxx";
```

`signature` の計算ルールについては、クライアントからのアップロード署名 をご参照ください。

アップロードの呼び出し

```
int publishCode = mVideoPublish.publishMedia(param);
```

説明：

- アップロード方法は、ファイルのサイズに応じて、通常アップロードとマルチパートアップロードが自動的に選択されます。マルチパートアップロードの各手順を気にすることなく、マルチパートアップロードを行うことができます。
- 指定のサブアプリケーションにアップロードしたい場合は、サブアプリケーションシステム - クライアントからのアップロードをご参照ください。

高度な機能

カバーの付加

アップロードパラメータとしてカバーへのパスを追加します。

```
TXUGCPublishTypeDef.TXPublishParam param = new TXUGCPublishTypeDef.TXPublishParam
();
param.signature = "xxx";
param.videoPath = "xxx";
param.coverPath = "xxx";
```

`signature` の計算ルールについては、クライアントからのアップロード署名 をご参照ください。

アップロードのキャンセルと再開

アップロードをキャンセルするには、 `TXUGCPublish` の `canclePublish()` インターフェースを呼び出します。

```
mVideoPublish.canclePublish();
```

アップロードを再開するには、同じアップロードパラメータを使用し（ビデオパスとカバーパスは変更されません） `TXUGCPublish` の `publishVideo` を再度呼び出します。

中断からの再開

VODはビデオのアップロード中、中断からの再開をサポートします。アップロードが予期せず終了した場合に、中断ポイントからアップロードを再開できるため、アップロード時間を短縮できます。

中断からの再開の有効期限は1日です。つまり同じビデオのアップロードが中断された場合、1日以内に再度アップロードすると中断ポイントからそのままアップロードできます。1日を超えるとデフォルトでは、完全なビデオを再度アップロードします。

アップロードパラメータの `enableResume` は、中断からの再開のスイッチであり、デフォルトで有効になっています。

事前アップロード

実際のアップロード中に発生するエラーのほとんどは、ネットワーク接続の障害またはタイムアウトに起因しています。このような問題を最適化するために、最適化された事前アップロードロジックを追加しました。事前アップロードには、HTTPDNSの解決 、推奨されるアップロードリージョンの取得および最適なアップロードリージョンの検出が含まれます。

Appを起動するときに `TXUGCPublishOptCenter.getInstance().prepareUpload(signature)` を呼び出すことをお勧めします。事前アップロードモジュールは、 `<ドメイン名，IP>` マッピングテーブルと最適なアップロードリージョンをローカルにキャッシュします。ネットワークスイッチが検出されると、キャッシュを自動的にパージして更新します。

> 注意：
> 必ず `AndroidManifest.xml` にネットワーク監視モジュールを登録してください。

```
<receiver android:name=".videoupload.impl.TVCNetWorkStateReceiver">
<intent-filter>
<!--ネットワーク変更を検出するaction-->
<action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
<category android:name="android.intent.category.DEFAULT" />
</intent-filter>
</receiver>
```

> `signature` の計算ルールについては、クライアントからのアップロード署名 をご参照ください。

httpsアップロードの有効化

アップロードパラメータのTXPublishParamの中のenableHTTPSをtrueに設定すれば完了です。デフォルトではfalseになっています。

```
TXUGCPublishTypeDef.TXPublishParam param = new TXUGCPublishTypeDef.TXPublishParam
();
param.enableHttps = true;
```

ビデオアップロードインターフェースの説明

アップロードオブジェクトの初期化： `TXUGCPublish`

パラメータ名	パラメータの説明	タイプ	入力必須
context	applicationコンテキスト。	Context	はい
customKey	ユーザーを区別するために使用されます。今後の問題特定を容易にするためにAppのアカウントIDを使用することをお勧めします。	String	いいえ

VOD appIdの設定：`TXUGCPublish.setAppId`

パラメータ名	パラメータの説明	タイプ	入力必須
appId	VOD appId。	int	はい

アップロードビデオ：`TXUGCPublish.publishVideo`

パラメータ名	パラメータの説明	タイプ	入力必須
param	アップロードパラメータ。	TXUGCPublishTypeDef.TXPublishParam	はい

アップロードパラメータ：`TXUGCPublishTypeDef.TXPublishParam`

パラメータ名	パラメータの説明	タイプ	入力必須
signature	クライアントからのアップロード署名。	String	はい
videoPath	ローカルビデオファイルパス。	String	はい
coverPath	ローカルカバーファイルパス。デフォルトではカバーファイルは含まれません。	String	いいえ
enableResume	中断ポイントからの再開の有効無効を指定。デフォルトでは有効になっています。	boolean	いいえ
enableHttps	HTTPSの有効無効を指定。デフォルトでは無効になっています。	boolean	いいえ
fileName	Tencent Cloudにアップロードされたビデオファイル名です。空のままの場合、デフォルトでローカルファイル名が使用されます。	String	いいえ

アップロードコールバックの設定：`TXUGCPublish.setListener`

パラメータ名	パラメータの説明	タイプ	入力必須

パラメータ名	パラメータの説明	タイプ	入力必須
listener	アップロードの進行状況と結果のコールバックを監視します。	TXUGCPublishTypeDef.ITXVideoPublishListener	はい

進行状況コールバック：`TXUGCPublishTypeDef.ITXVideoPublishListener.onPublishProgress`

変数名	変数の説明	タイプ
uploadBytes	アップロード済みのバイト数。	long
totalBytes	合計バイト数。	long

結果コールバック：`TXUGCPublishTypeDef.ITXVideoPublishListener.onPublishComplete`

変数名	変数の説明	タイプ
result	アップロード結果。	TXUGCPublishTypeDef.TXPublishResult

アップロード結果：`TXUGCPublishTypeDef.TXPublishResult`

メンバー変数名	変数の説明	タイプ
retCode	結果コード。	int
descMsg	アップロード失敗のエラー説明。	String
videoId	VODビデオファイルID。	String
videoURL	ビデオストレージアドレス。	String
coverURL	カバーストレージアドレス。	String

事前アップロード：`TXUGCPublishOptCenter.prepareUpload`

パラメータ名	パラメータの説明	タイプ	入力必須
signature	クライアントからのアップロード署名。	String	はい

画像アップロードインターフェースの説明

アップロードオブジェクトの初期化：`TXUGCPublish`

パラメータ名	パラメータの説明	タイプ	入力必須
context	applicationコンテキスト。	Context	はい
customKey	ユーザーを区別するために使用されます。今後の問題特定を容易にするためにAppのアカウントIDを使用することをお勧めします。	String	いいえ

VOD appIdの設定：`TXUGCPublish.setAppId`

パラメータ名	パラメータの説明	タイプ	入力必須
appId	VOD appId。	int	はい

アップロード画像：`TXUGCPublish.publishMedia`

パラメータ名	パラメータの説明	タイプ	入力必須
param	アップロードパラメータ。	TXUGCPublishTypeDef.TXMediaPublishParam	はい

アップロードパラメータ：`TXUGCPublishTypeDef.TXMediaPublishParam`

パラメータ名	パラメータの説明	タイプ	入力必須
signature	クライアントからのアップロード署名。	String	はい
mediaPath	ローカル画像ファイルパス。	String	はい
enableResume	中断ポイントからの再開の有効無効を指定。デフォルトでは有効になっています。	boolean	いいえ
enableHttps	HTTPSの有効無効を指定。デフォルトでは無効になっています。	boolean	いいえ
fileName	Tencent Cloudにアップロードされたビデオファイル名です。空のままの場合、デフォルトでローカルファイル名が使用されます。	String	いいえ

アップロードコールバックの設定：`TXUGCPublish.setListener`

パラメータ名	パラメータの説明	タイプ	入力必須

パラメータ名	パラメータの説明	タイプ	入力必須
listener	アップロードの進行状況と結果のコールバックを監視します。	TXUGCPublishTypeDef.ITXMediaPublishListener	はい

進行状況コールバック：`TXUGCPublishTypeDef.ITXMediaPublishListener.onPublishProgress`

変数名	変数の説明	タイプ
uploadBytes	アップロード済みのバイト数。	long
totalBytes	合計バイト数。	long

結果コールバック：`TXUGCPublishTypeDef.ITXMediaPublishListener.onPublishComplete`

変数名	変数の説明	タイプ
result	アップロード結果。	TXUGCPublishTypeDef.TXPublishResult

アップロード結果：`TXUGCPublishTypeDef.TXMediaPublishResult`

メンバー変数名	変数の説明	タイプ
retCode	結果コード。	int
descMsg	アップロード失敗のエラー説明。	String
mediaId	VODビデオファイルID。	String
mediaURL	メディアリソースストレージアドレス。	String

事前アップロード：`TXUGCPublishOptCenter.prepareUpload`

パラメータ名	パラメータの説明	タイプ	入力必須
signature	クライアントからのアップロード署名。	String	はい

エラーコード

SDKは、`TXUGCPublishTypeDef.ITXVideoPublishListene\ITXMediaPublishListener` インターフェースを介してビデオのアップロードのステータスを監視します。従って、`TXUGCPublishTypeDef.TXPublishResult\TXMediaPublishResult` の `retCode` を使用して、ビデオのアップロード状況を確認することができます。

ステータスコード	TVCConstantsにおいて対応する定数	意味
0	NO_ERROR	アップロードに成功しました。
1001	ERR_UGC_REQUEST_FAILED	アップロードリクエストが失敗しました。通常、クライアントの署名が期限切れまたは無効になっており、Appに別の署名を再申請する必要があります。
1002	ERR_UGC_PARSE_FAILED	リクエスト情報の解析に失敗しました。
1003	ERR_UPLOAD_VIDEO_FAILED	ビデオのアップロードに失敗しました。
1004	ERR_UPLOAD_COVER_FAILED	カバーのアップロードに失敗しました。
1005	ERR_UGC_FINISH_REQUEST_FAILED	アップロード終了リクエストに失敗しました。
1006	ERR_UGC_FINISH_RESPONSE_FAILED	アップロード終了の応答に失敗しました。
1007	ERR_CLIENT_BUSY	クライアントはビジーです（オブジェクトはそれ以上のリクエストを処理できません）。
1008	ERR_FILE_NOEXIT	アップロードファイルが存在しません。
1009	ERR_UGC_PUBLISHING	ビデオのアップロード中です。
1010	ERR_UGC_INVALID_PARAM	アップロードパラメータが空です。
1012	ERR_UGC_INVALID_SIGNATURE	ビデオアップロードのsignatureが空です。
1013	ERR_UGC_INVALID_VIDOPATH	ビデオファイルのパスが空です。
1014	ERR_UGC_INVALID_VIDEO_FILE	現在のパスにビデオファイルが存在しません。
1015	ERR_UGC_FILE_NAME	動画アップロードファイル名が長すぎる（40を超える）か、または特殊文字が含まれています。

ステータスコード	TVCConstantsにおいて対応する定数	意味
1016	ERR_UGC_INVALID_COVER_PATH	ビデオファイルのカバーパスが間違っており、ファイルが存在しません。
1017	ERR_USER_CANCEL	ユーザーがアップロードをキャンセルしました。
1018	ERR_UPLOAD_VOD	5M未満のファイルのVODへの直接アップロードに失敗しました。

iOSアップロードSDK

最終更新日：：2023-10-26 17:31:32

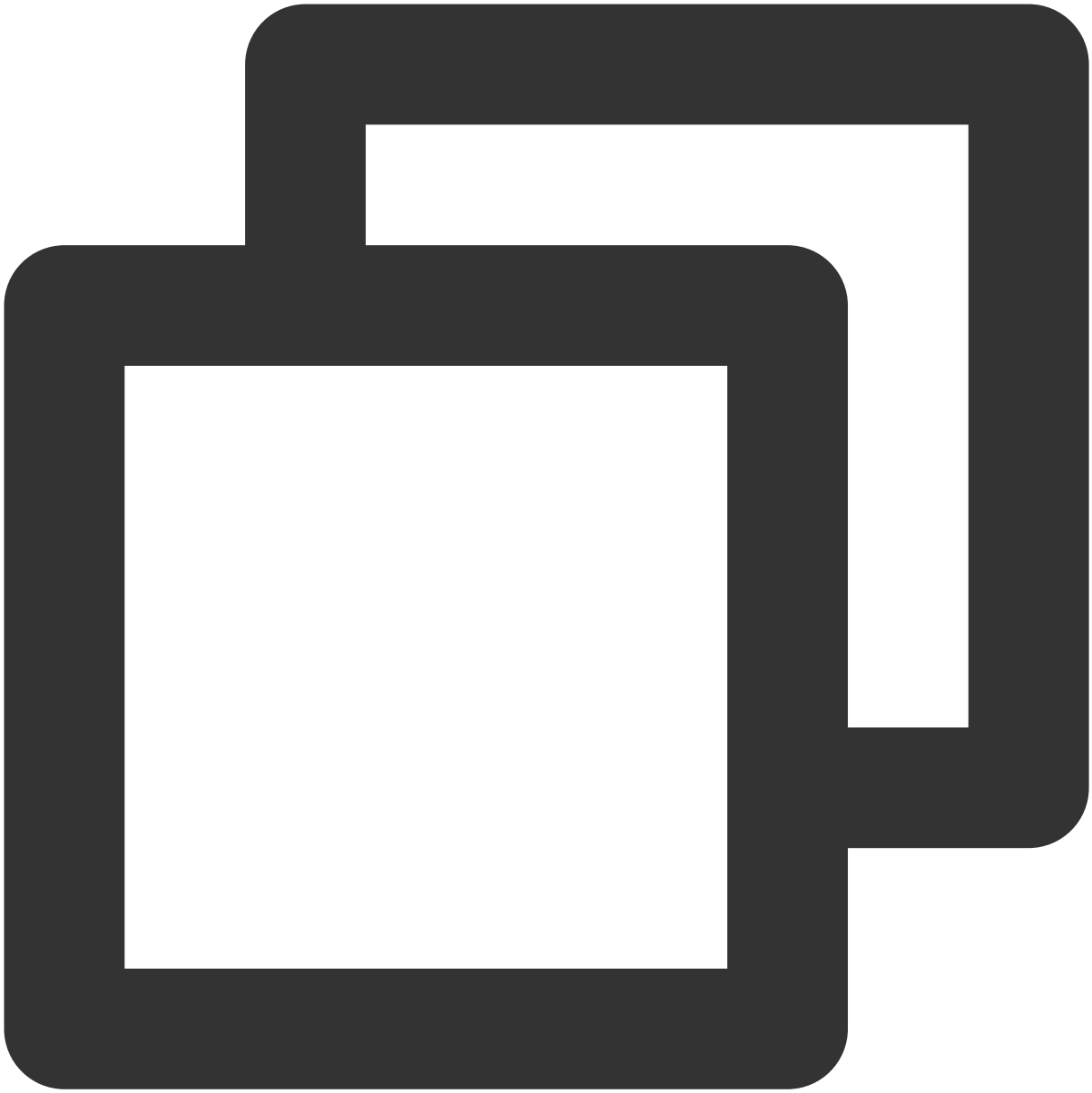
VODは、iOSプラットフォームでビデオをアップロードするシナリオ向けに、iOSアップロードSDKを提供しています。アップロードのフローについては、クライアントからのアップロードガイドをご参照ください。

ソースコードのダウンロード

1. iOSアップロードDemoとソースコードをクリックしてダウンロードします。
2. ダウンロードした圧縮パッケージを解凍すると、TXUGCUploadDemoディレクトリが表示されます。アップロードソースコードは、`TXUGCUploadDemo/upload` ディレクトリにあります。

アップロードライブラリとソースコードの統合

1. ソースコードディレクトリ `TXUGCUploadDemo/upload` をプロジェクトディレクトリにコピーします。
2. 動的ライブラリ `QCloudCore.framework` 、 `QCloudCOSXML.framework` （ `TXUGCUploadDemo/upload/COSSDK/` ディレクトリにあります）をプロジェクトにインポートし、次の依存ライブラリを追加します。

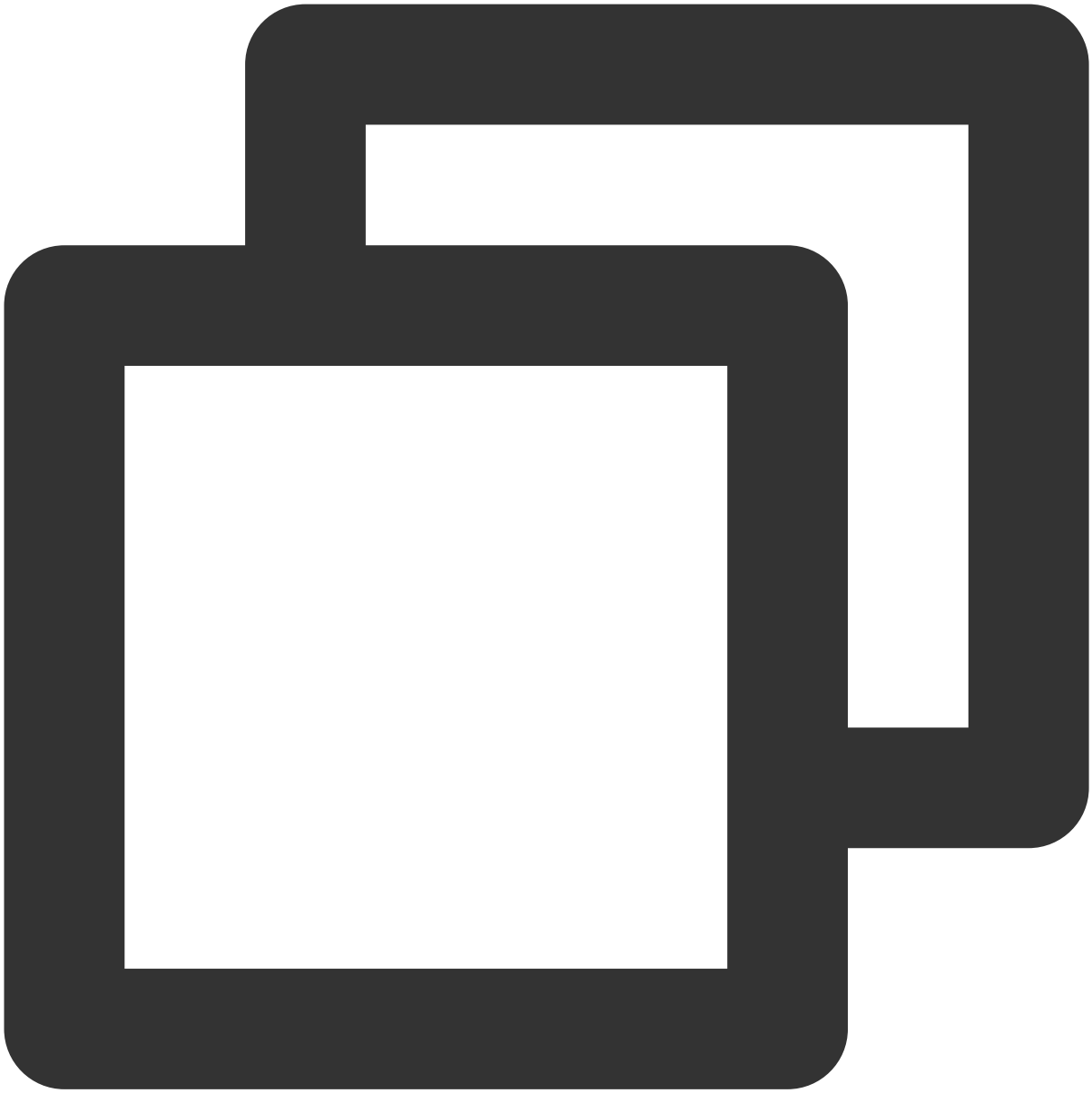


```
1. CoreTelephony.framework
2. Foundation.framework
3. SystemConfiguration.framework
4. libc++.tbd
```

3. Build Settingsにおいて、Other Linker Flagsを設定し、パラメータ `-ObjC` を追加します。

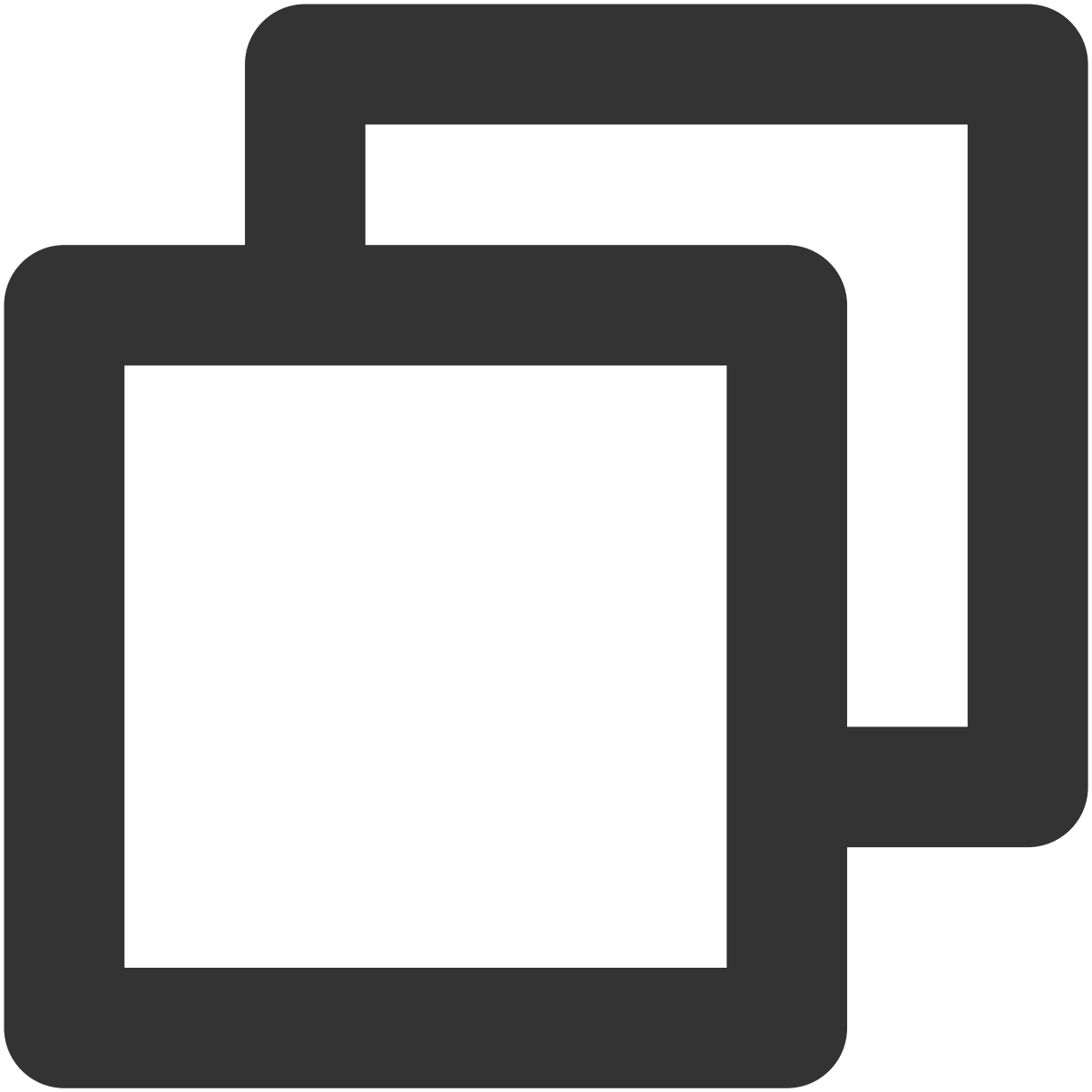
シンプルビデオアップロード

アップロードオブジェクトの初期化

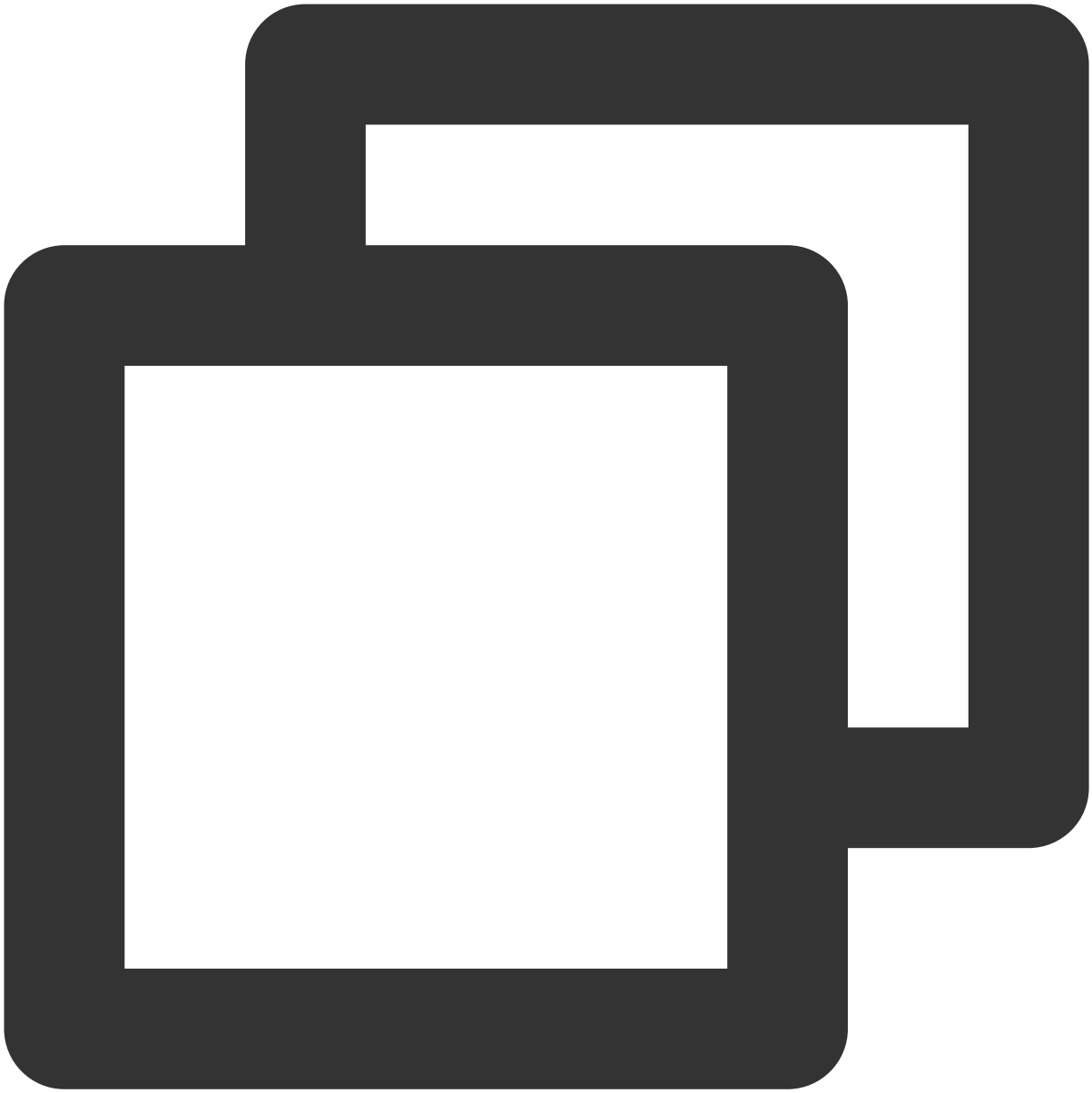


```
TXUGCPublish *_videoPublish = [[TXUGCPublish alloc] initWithUserID:@"upload_video_u
```

アップロードオブジェクトのコールバックの設定



```
_videoPublish.delegate = self;
```



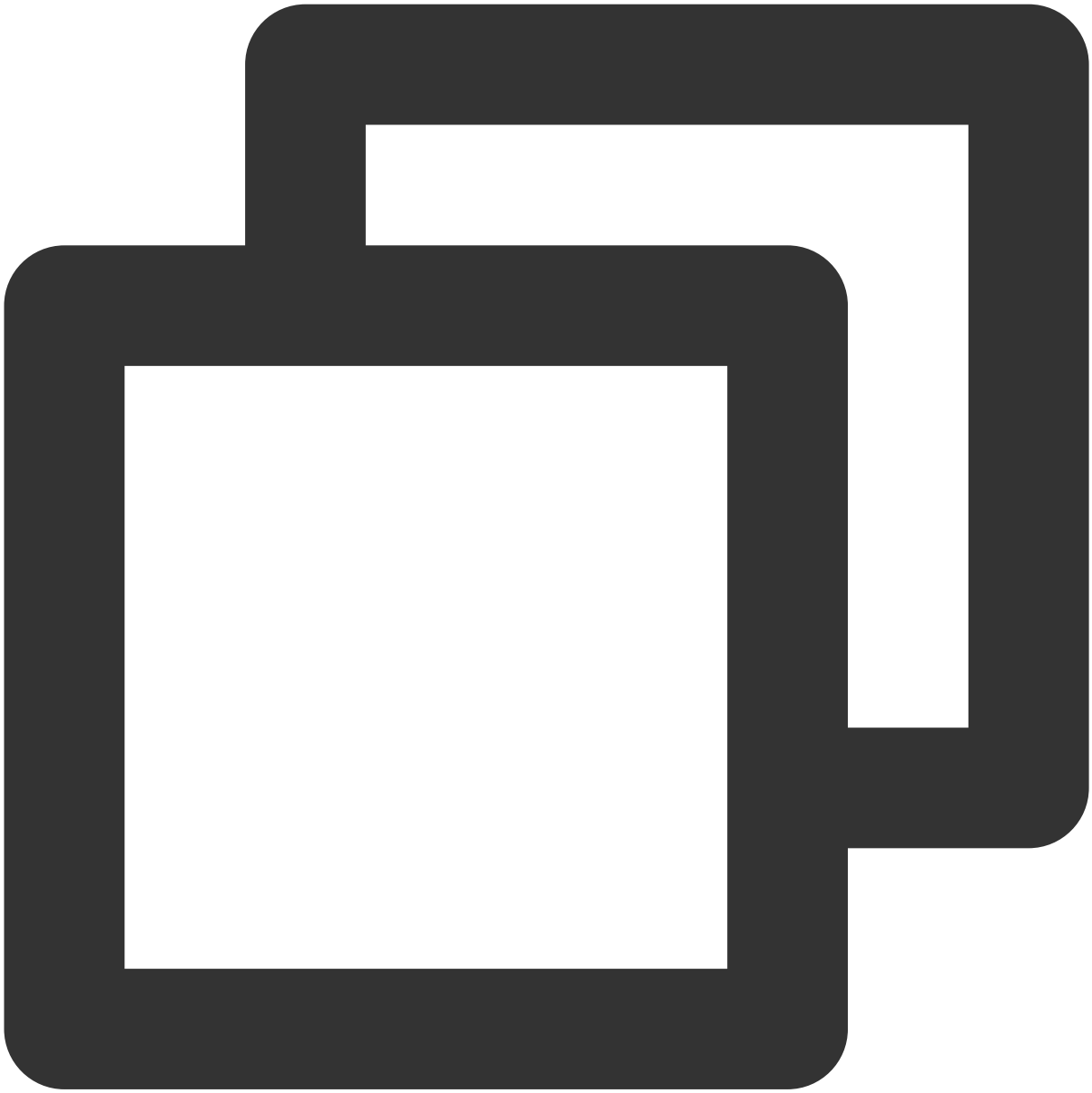
```
#pragma mark - TXVideoPublishListener

- (void)onPublishProgress:(NSInteger)uploadBytes totalBytes:(NSInteger)totalBytes {
    self.progressView.progress = (float)uploadBytes/totalBytes;
    NSLog(@"onPublishProgress [%ld/%ld]", uploadBytes, totalBytes);
}

- (void)onPublishComplete:(TXPublishResult*)result {
    NSString *string = [NSString stringWithFormat:@"アップロード完了、エラーコード[%d],
    [self showErrorMessage:string];
    NSLog(@"onPublishComplete [%d/%@]", result.retCode, result.retCode == 0? result
```

```
}
```

アップロードパラメータの作成

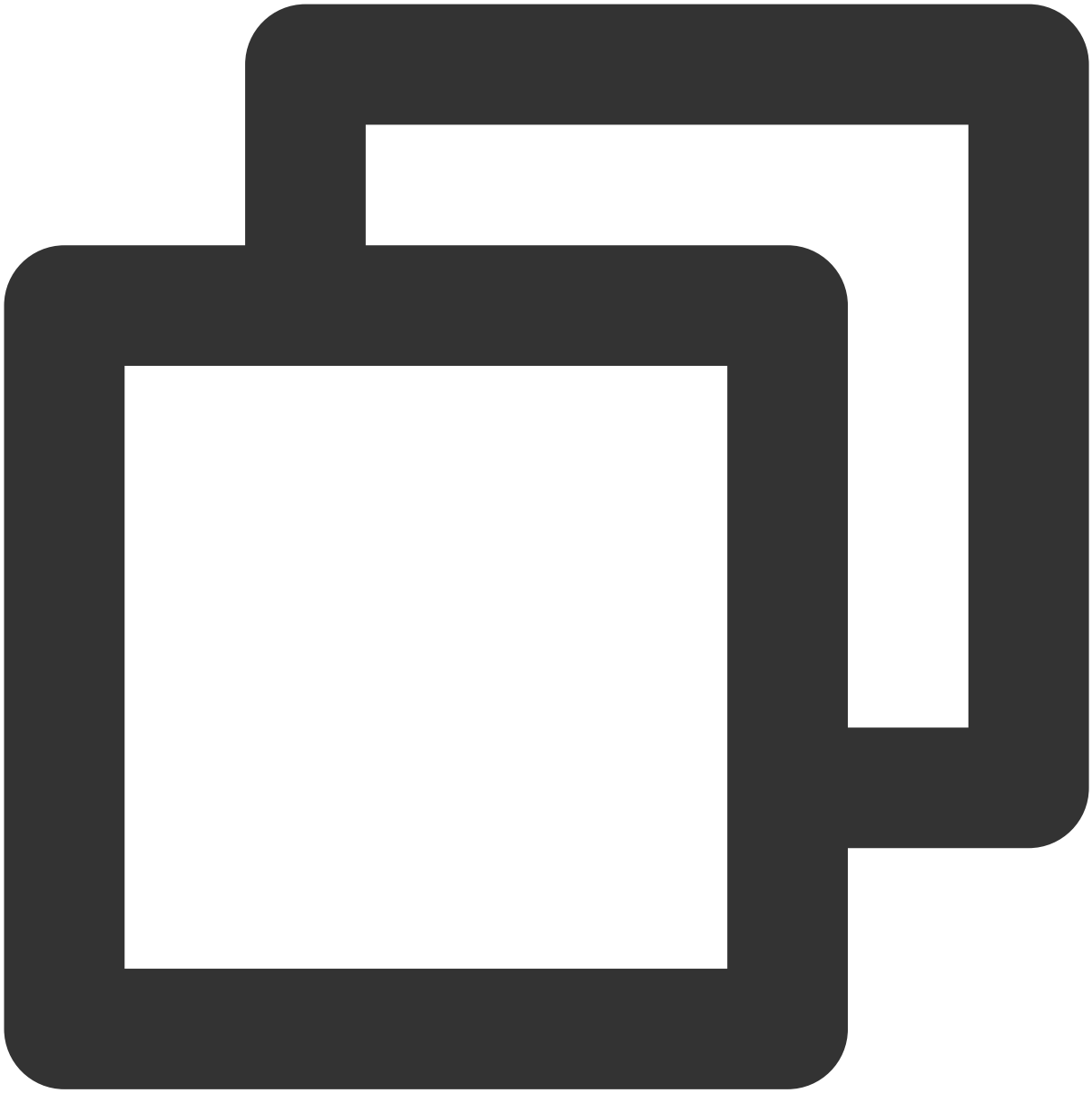


```
TXPublishParam *publishParam = [[TXPublishParam alloc] init];

publishParam.signature  = @"業務バックエンドで生成された署名";
publishParam.videoPath  = @"ビデオファイルパス";
```

`signature` の計算ルールについては、クライアントからのアップロード署名をご参照ください。

アップロードの呼び出し



```
[_videoPublish publishVideo:publishParam];
```

説明：

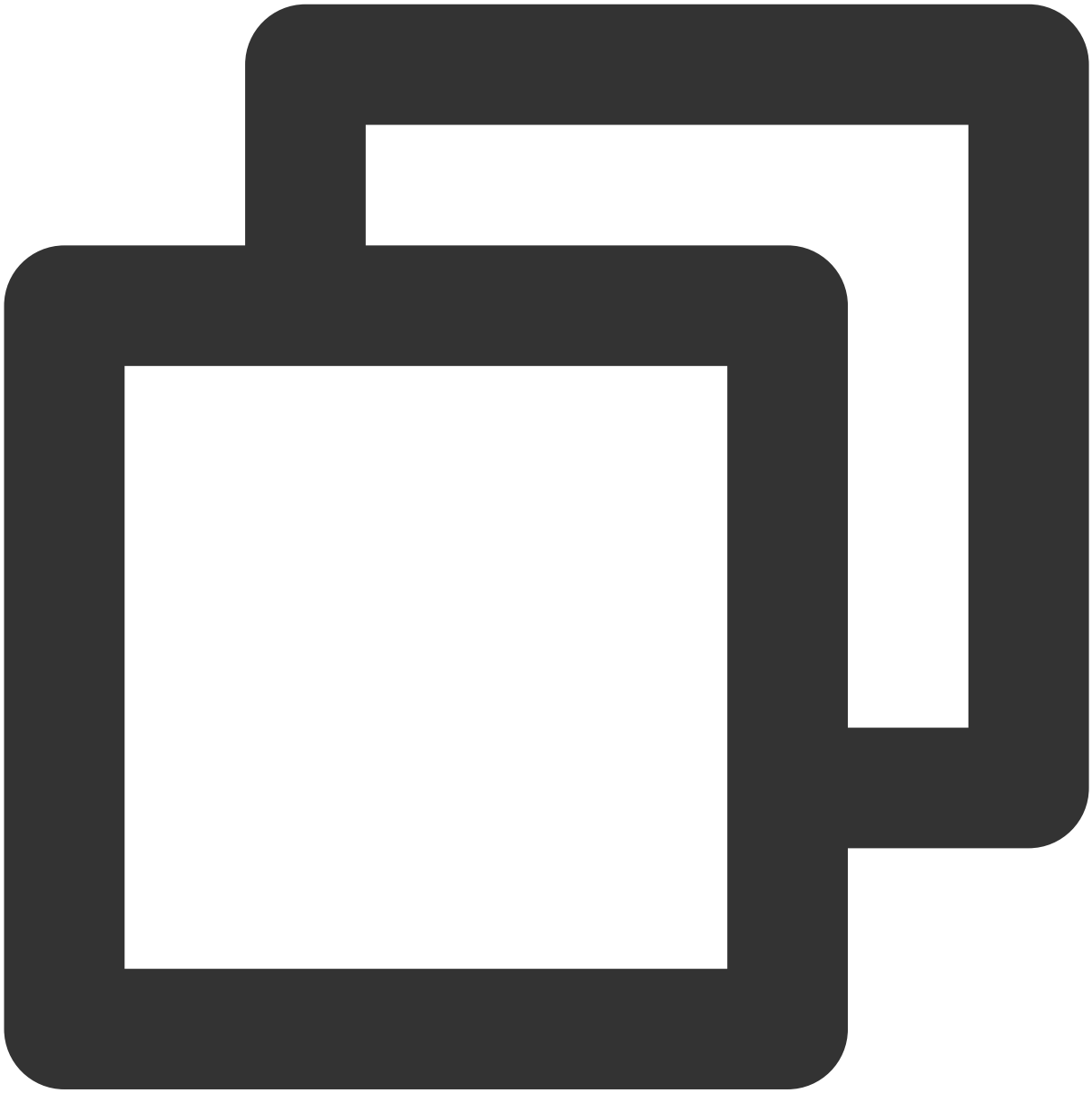
アップロード方法は、ファイルのサイズに応じて、通常アップロードとマルチパートアップロードが自動的に選択されます。マルチパートアップロードの各手順を気にすることなく、マルチパートアップロードを行うことができます。

指定のサブアプリケーションにアップロードしたい場合は、サブアプリケーションシステム - クライアントからのアップロードをご参照ください。

高度な機能

カバー画像の付加

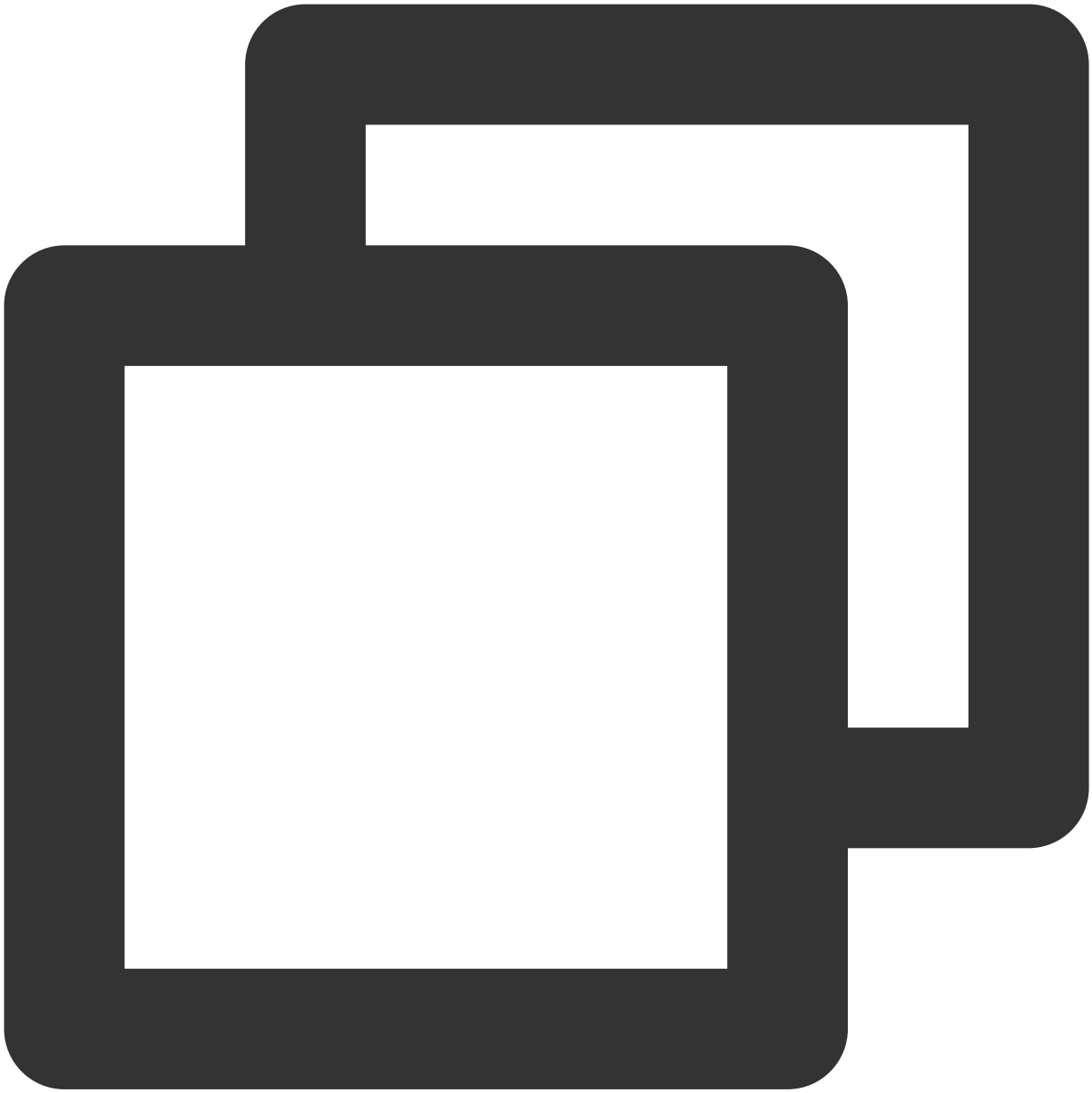
アップロードパラメータにカバー画像を追加します。



```
TXPublishParam *publishParam = [[TXPublishParam alloc] init];
publishParam.signature  = @"業務バックエンドで生成された署名";
publishParam.coverPath = @"カバー画像ファイルパス";
publishParam.videoPath  = @"ビデオファイルパス";
```

アップロードのキャンセルと再開

アップロードをキャンセルするには、 `canclePublish` インターフェースを呼び出します。



```
[_videoPublish canclePublish];
```

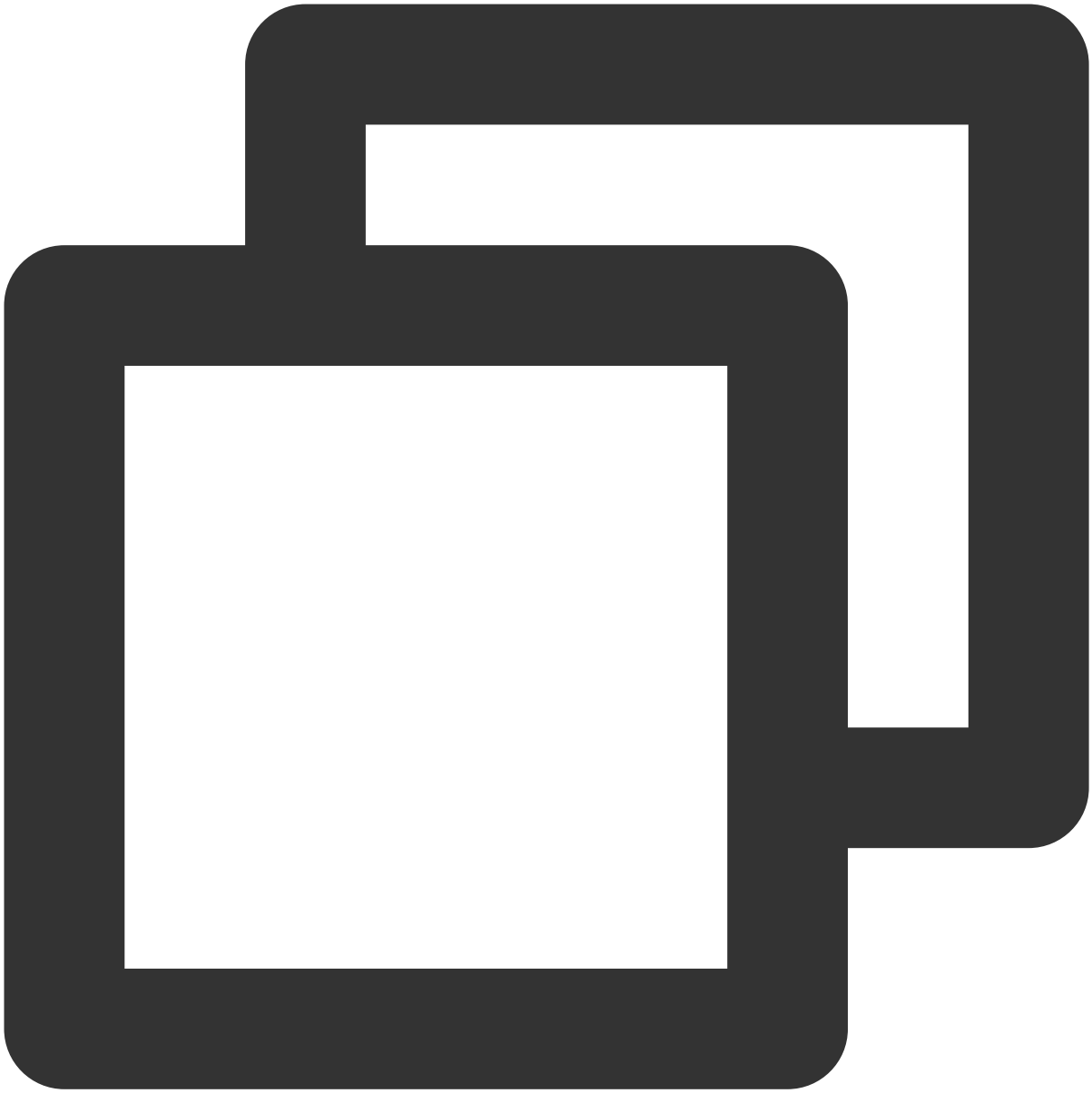
アップロードを再開するには、同じアップロードパラメータを使用し（ビデオパスとカバーパスは変更されません） `TXUGCPublish` の `publishVideo` を再度呼び出します。

中断からの再開

VODはビデオのアップロード中、中断からの再開をサポートします。アップロードが予期せず終了した場合に、中断ポイントからアップロードを再開できるため、アップロード時間を短縮できます。中断からの再開の有効期限は1日です。つまり同じビデオのアップロードが中断された場合、1日以内に再度アップロードすると中断ポイントからそのままアップロードできます。1日を超えるとデフォルトでは、完全なビデオを再度アップロードします。
アップロードパラメータの `enableResume` は、中断からの再開のスイッチであり、デフォルトで有効になっています。

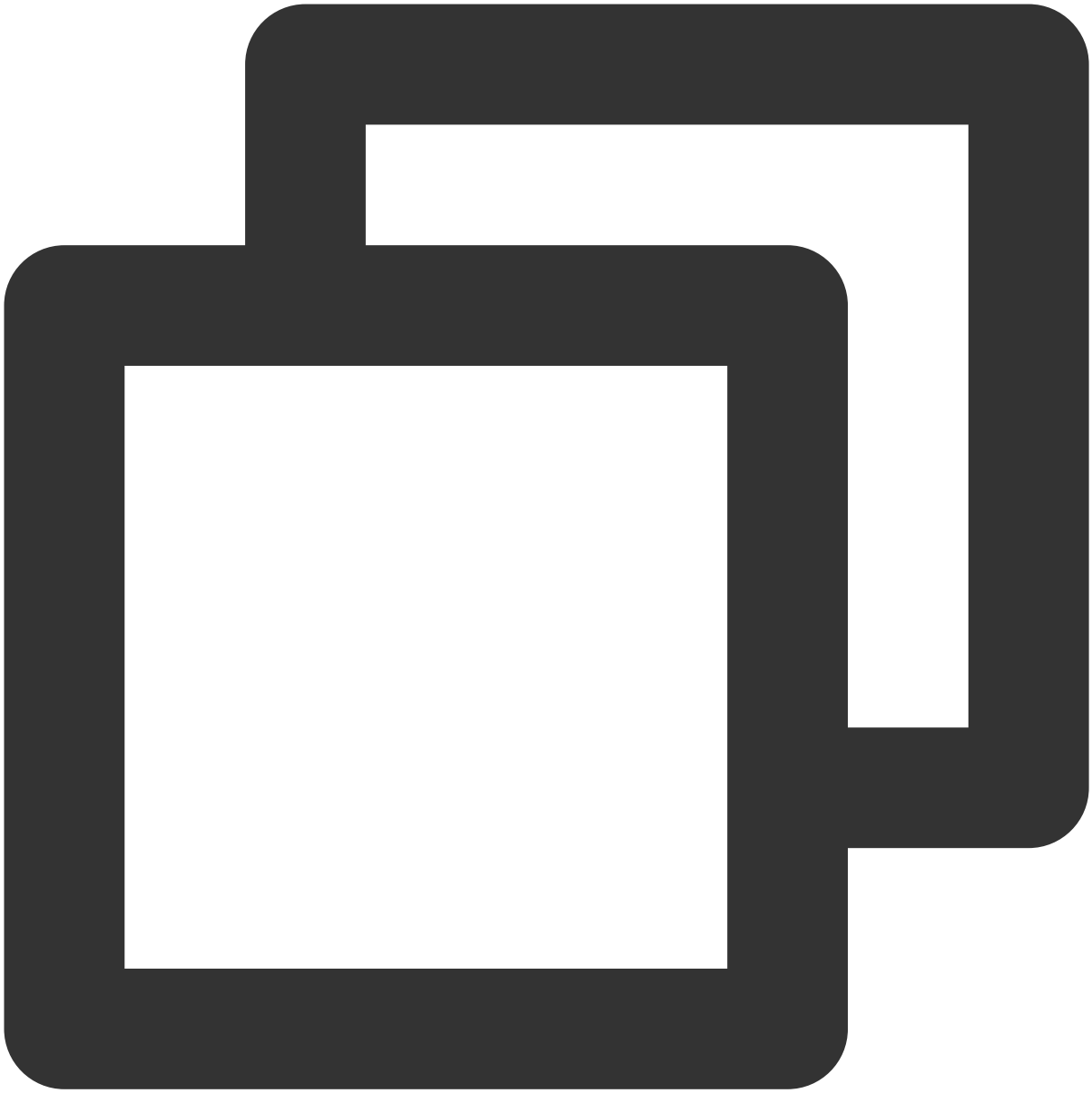
httpsアップロードの有効化

アップロードパラメータのTXPublishParamの中のenableHTTPSをtrueに設定すれば完了です。デフォルトではfalseになっています。



```
TXPublishParam *publishParam = [[TXPublishParam alloc] init];
publishParam.enableHTTPS  = true;
```

画像とメディアのアップロード



```
// オブジェクトの作成
TXUGCPublish *_imagePublish = [[TXUGCPublish alloc] initWithUserID:@"upload_image_u

// コールバックの設定
_imagePublish.mediaDelegate = self;

// アップロードパラメータの作成
TXMediaPublishParam *publishParam = [[TXMediaPublishParam alloc] init];
publishParam.signature  = @"業務バックエンドで生成された署名";
publishParam.mediaPath = @"画像ファイルへのパス";
```

```
// 画像またはメディアファイルをアップロードします
[_imagePublish publishMedia:publishParam];
```

ビデオアップロードインターフェースの説明

アップロードオブジェクトの初期化：`TXUGCPublish::initWithUserID`

パラメータ名	パラメータの記述	タイプ	入力必須
userID	ユーザーのuserID。ユーザーを区別するために使用されます。	NSString	いいえ

アップロードの開始：`TXUGCPublish.publishVideo`

パラメータ名	パラメータの記述	タイプ	入力必須
param	パラメータをリリースします。	TXPublishParam	はい

アップロードパラメータ：`TXPublishParam`

パラメータ名	パラメータの記述	タイプ	入力必須
signature	クライアントからのアップロード署名を行います。	NSString*	はい
videoPath	ローカルビデオファイルパスです。	NSString*	はい
coverPath	カバー画像のローカルパスです。設定しなくてもかまいません。	NSString*	いいえ
fileName	Tencent Cloudにアップロードするビデオファイル名です。入力しない場合はデフォルトでローカルファイル名を使用します。	NSString*	いいえ
enableResume	中断からの再開を有効にするか。デフォルトでは有効になっています。	BOOL	いいえ
enableHttps	HTTPSを有効にするか。デフォルトでは無効になっています。	BOOL	いいえ

アップロードコールバックの設定：`TXUGCPublish.delegate`

メンバー変数名	変数の記述	タイプ	入力必

			須
delegate	アップロードの進捗状況と結果のコールバックを監視します。	TXVideoPublishListener	はい

アップロード進行状況のコールバック：`onPublishProgress`

変数名	変数の記述	タイプ
uploadBytes	アップロード済みのバイト数です。	NSInteger
totalBytes	総バイト数です。	NSInteger

アップロード結果のコールバック：`onPublishComplete`

変数名	変数の記述	タイプ
result	アップロードの結果です。	TXPublishResult

アップロードイベントのコールバック：`onPublishEvent`

変数名	変数の記述	タイプ
evt	イベント。デバッグプリントに使用します。	NSDictionary

アップロード結果：`TXPublishResult`

メンバー変数名	変数の説明	タイプ
retCode	エラーコード	int
descMsg	アップロード失敗のエラー記述。	NSString
videoId	VODビデオファイルID。	NSString
videoURL	ビデオストレージアドレス。	NSString
coverURL	カバー画像のストレージアドレス。	NSString

事前アップロード：`TXUGCPublishOptCenter.prepareUpload`

パラメータ名	パラメータの記述	タイプ	入力必須
signature	クライアントからのアップロード署名	NSString	はい

エラーコード

SDKは、`TXVideoPublishListener` インターフェースによってビデオのアップロードのステータスを監視します。従って、`TXPublishResult` の `retCode` を使用して、ビデオの公開状況を確認することができます。

エラーコード	TVCCommonにおいて対応する定数	意味
0	TVC_OK	アップロードに成功しました。
1001	TVC_ERR_UGC_REQUEST_FAILED	アップロードリクエストが失敗しました。通常、クライアントの署名が期限切れまたは無効になっており、Appに署名を再申請する必要があります。
1002	TVC_ERR_UGC_PARSE_FAILED	リクエスト情報の解析に失敗しました。
1003	TVC_ERR_VIDEO_UPLOAD_FAILED	ビデオのアップロードに失敗しました。
1004	TVC_ERR_COVER_UPLOAD_FAILED	カバー画像のアップロードに失敗しました。
1005	TVC_ERR_UGC_FINISH_REQ_FAILED	アップロード終了リクエストが失敗しました。
1006	TVC_ERR_UGC_FINISH_RSP_FAILED	アップロード終了レスポンスのエラーです。

画像とメディアのアップロードインターフェースの記述

アップロードオブジェクトの初期化：`TXUGCPublish::initWithUserID`

パラメータ名	パラメータの記述	タイプ	入力必須
userID	ユーザーのuserID。ユーザーを区別するために使用されます。	NSString	いいえ

アップロードの開始：`TXUGCPublish.publishMedia`

パラメータ名	パラメータの記述	タイプ	入力必須

param	パラメータをリリースします。	TXMediaPublishParam	はい

アップロードパラメータ：`TXMediaPublishParam`

パラメータ名	パラメータの記述	タイプ	入力必須
signature	クライアントからのアップロード署名を行います。	NSString*	はい
mediaPath	ローカル画像/メディアファイルパスです。	NSString*	はい
fileName	Tencent Cloudにアップロードする画像/メディアファイル名です。入力しない場合はデフォルトでローカルファイル名を使用します。	NSString*	いいえ
enableResume	中断からの再開を有効にするか。デフォルトでは有効になっています。	BOOL	いいえ
enableHttps	HTTPSを有効にするか。デフォルトでは無効になっています。	BOOL	いいえ

アップロードコールバックの設定：`TXUGCPublish.TXMediaPublishListener`

メンバー変数名	変数の記述	タイプ	入力必須
mediaDelegate	アップロードの進捗状況と結果のコールバックを監視します。	TXMediaPublishListener	はい

アップロード進行状況のコールバック：`onMediaPublishProgress`

変数名	変数の記述	タイプ
uploadBytes	アップロード済みのバイト数です。	NSInteger
totalBytes	総バイト数です。	NSInteger

アップロード結果のコールバック：`onMediaPublishComplete`

変数名	変数の記述	タイプ
result	アップロードの結果です。	TXMediaPublishResult

アップロードイベントのコールバック：`onMediaPublishEvent`

変数名	変数の記述	タイプ
evt	イベント。デバッグプリントに使用します。	NSDictionary

アップロード結果：`TXMediaPublishResult`

メンバー変数名	変数の説明	タイプ
retCode	エラーコード	int
descMsg	アップロード失敗のエラー記述。	NSString
mediaId	画像/メディアファイルIDです。	NSString
mediaURL	画像/メディアストレージアドレスです。	NSString

事前アップロード：`TXUGCPublishOptCenter.prepareUpload`

パラメータ名	パラメータの記述	タイプ	入力必須
signature	クライアントからのアップロード署名	NSString	はい

エラーコード

SDKは、`TXMediaPublishListener` インターフェースによって画像/メディアのアップロードのステータスを監視します。従って、`TXMediaPublishResult` の `retCode` を使用して、画像/メディアの公開状況を確認することができます。

エラーコード	TVCCommonにおいて対応する定数	意味
0	MEDIA_PUBLISH_RESULT_OK	アップロードに成功しました。
1001	MEDIA_PUBLISH_RESULT_UPLOAD_REQUEST_FAILED	アップロードリクエストが失敗しました。通常、クライアントの署名が期限切れまたは無効になっており、Appに署名を再申請する必要があります。
1002	MEDIA_PUBLISH_RESULT_UPLOAD_RESPONSE_ERROR	リクエスト情報の解析に失敗しまし

		た。
1003	MEDIA_PUBLISH_RESULT_UPLOAD_VIDEO_FAILED	画像/メディアのアップロードに失敗しました。
1005	MEDIA_PUBLISH_RESULT_PUBLISH_REQUEST_FAILED	アップロード終了リクエストが失敗しました。
1006	MEDIA_PUBLISH_RESULT_PUBLISH_RESPONSE_ERROR	アップロード終了レスポンスのエラーです。

Upload SDK for Flutter

最終更新日：：2024-01-18 10:25:31

VOD provides an SDK for uploading videos from Flutter clients. For details about the upload process, see Guide.

Environment Setup

Flutter：

Flutter 2.5.0 and above

Dart 2.19.2 and below 3.0

Android:

Android Studio 3.5 and above

Android 4.1 and above

iOS：

Xcode 11.0 and above

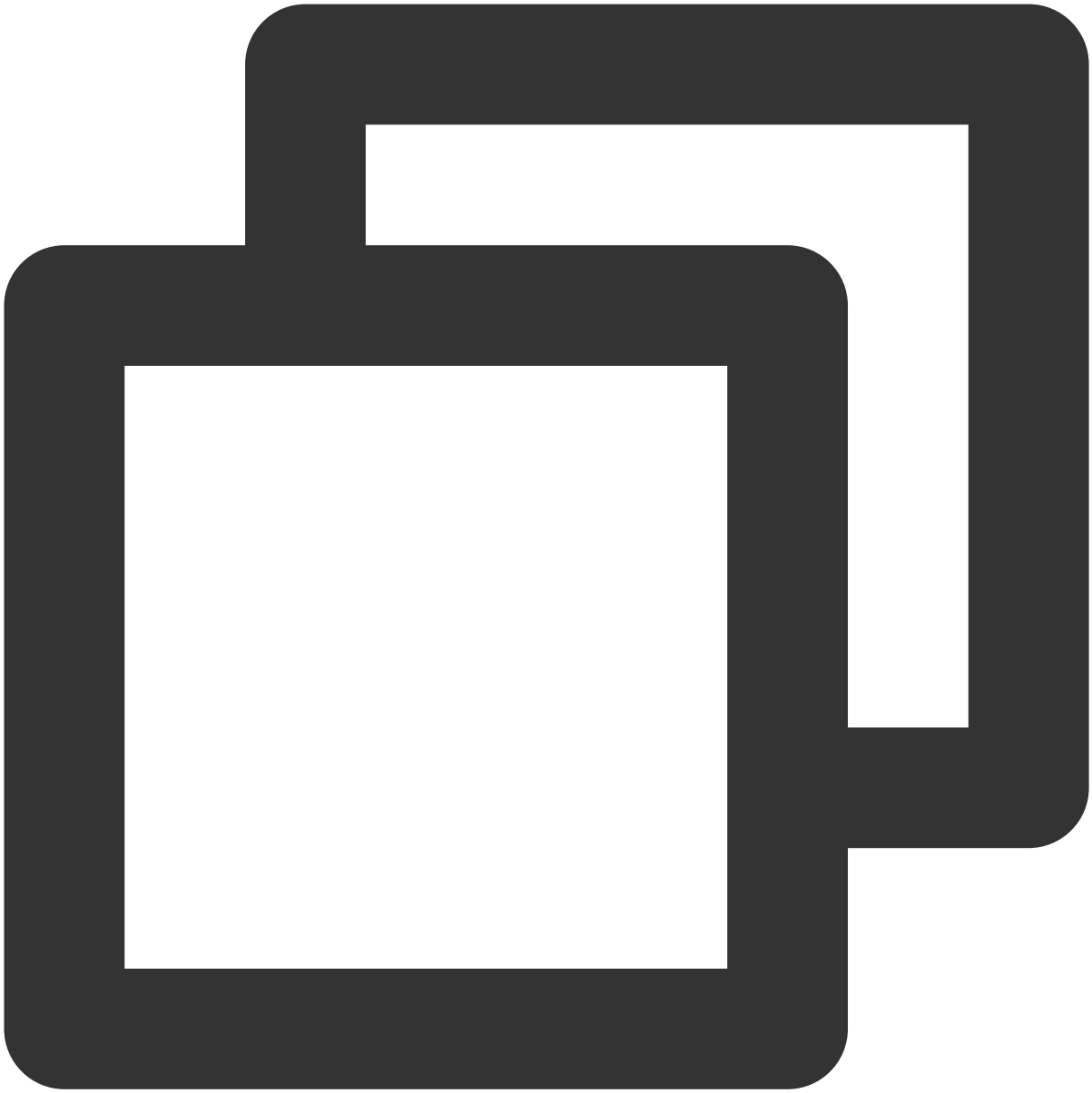
iOS 9.0 and above

Make sure your project has a valid developer signature set up

Quick Integration

Add Dependencies

1. Copy the SDK source code to your project directory.
2. Add the SDK to `pubspec.yaml`



```
vod_upload_flutter:
  path: ./vod_upload
```

3. Run the command `flutter pub get` in the root directory of your project to refresh the dependencies.

Note：

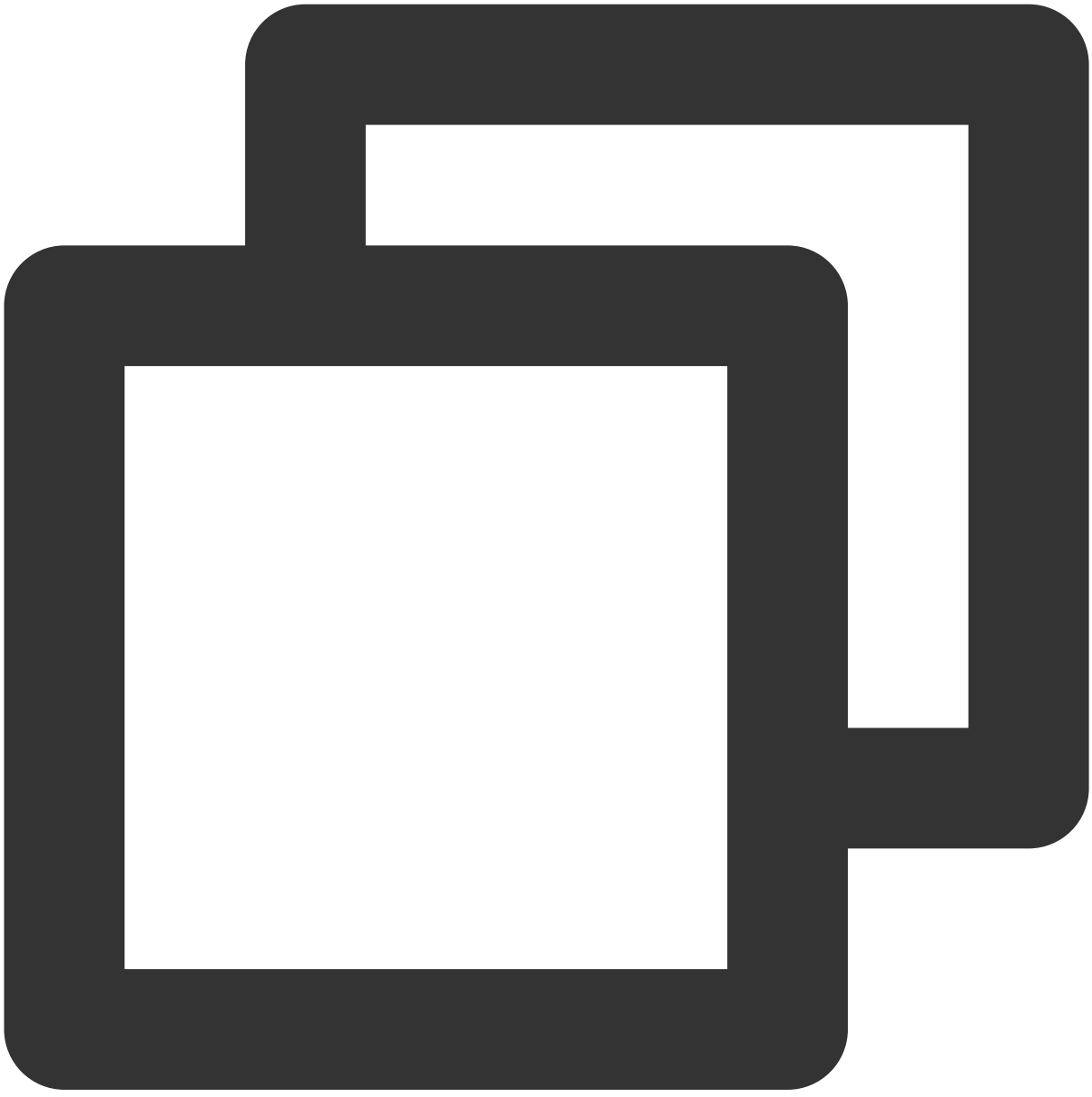
1. It is recommended to run `flutter pub get` command separately in the `root directory` , `SDK directory` , and `SDK Example directory` to avoid potential errors.

2. The `SDK Example directory` is the test project for the SDK. You can delete it if not needed.

Add Native Configurations

Android

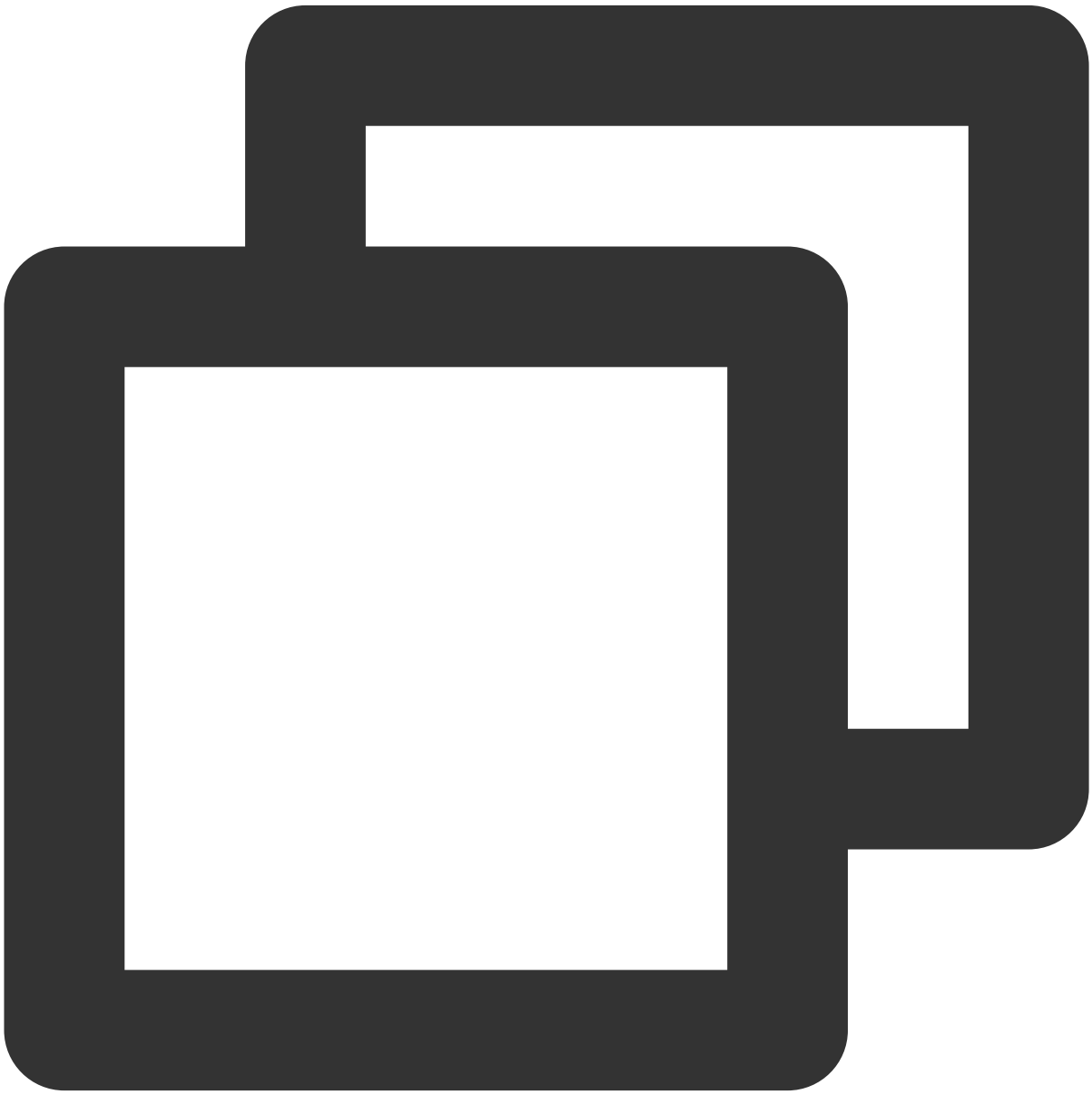
Add the following configurations to `AndroidManifest.xml`



```
<!-- Network permissions -->
<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" />
```

Allow App to Send HTTP Requests: Starting from Android P, Google requires that app requests use encrypted connections for security reasons. The SDK requires network requests for uploading. If your app has `targetSdkVersion >= 28` and needs to use the HTTP protocol for uploading, you can enable sending HTTP requests through network security configuration. Otherwise, you may encounter the `java.io.IOException: Cleartext HTTP traffic to xxx not permitted` error, which prevents uploading. Follow these steps to configure it:

1. Create a `res/xml/network_security_config.xml` file in your project to set up network security configuration

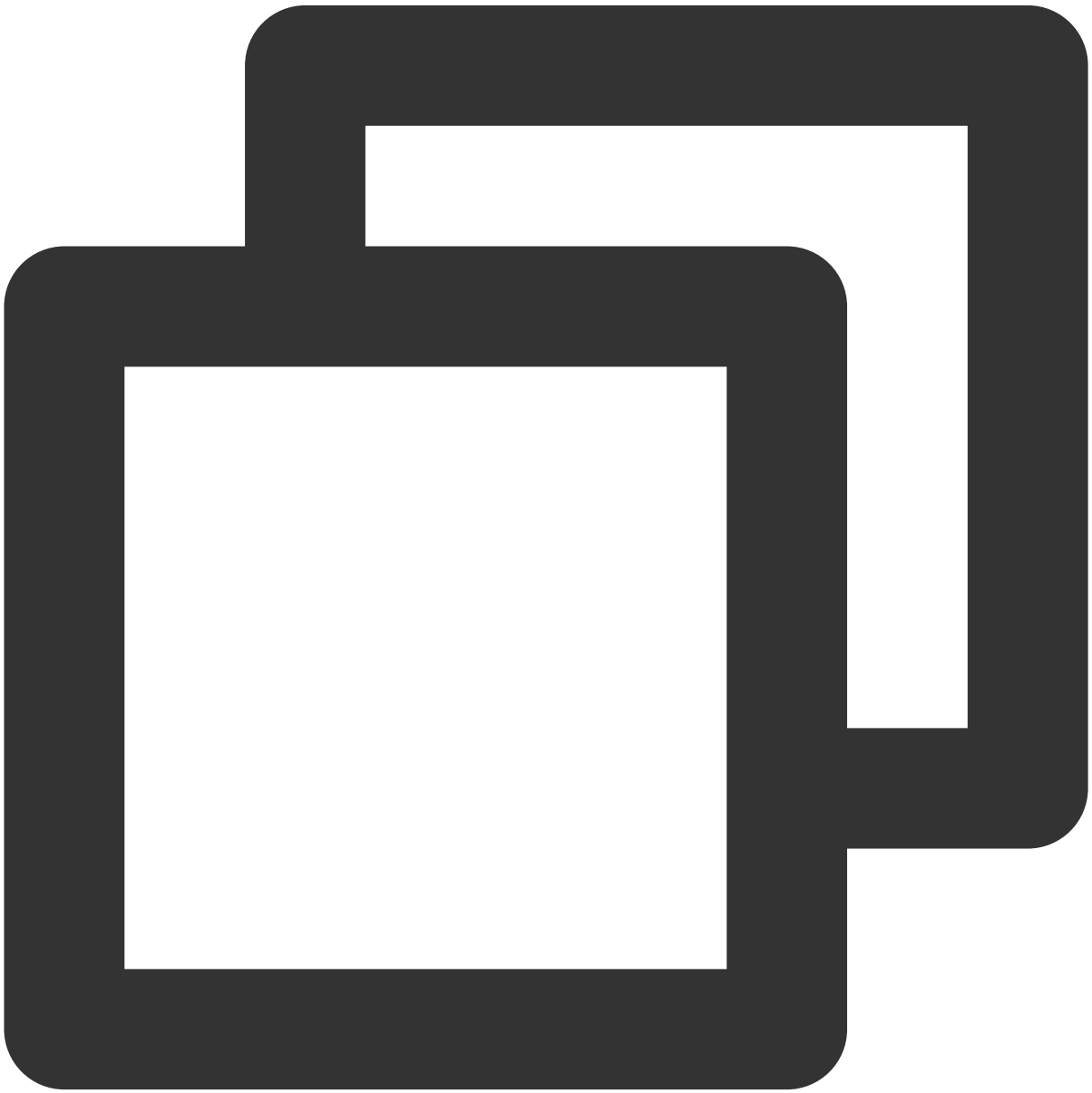


```
<?xml version="1.0" encoding="UTF-8" ?>
<network-security-config>
    <base-config cleartextTrafficPermitted="ture" />
</network-security-config>
```

Note：

The above configuration is only for `testing environments` . In production environments, you should configure network security as needed.

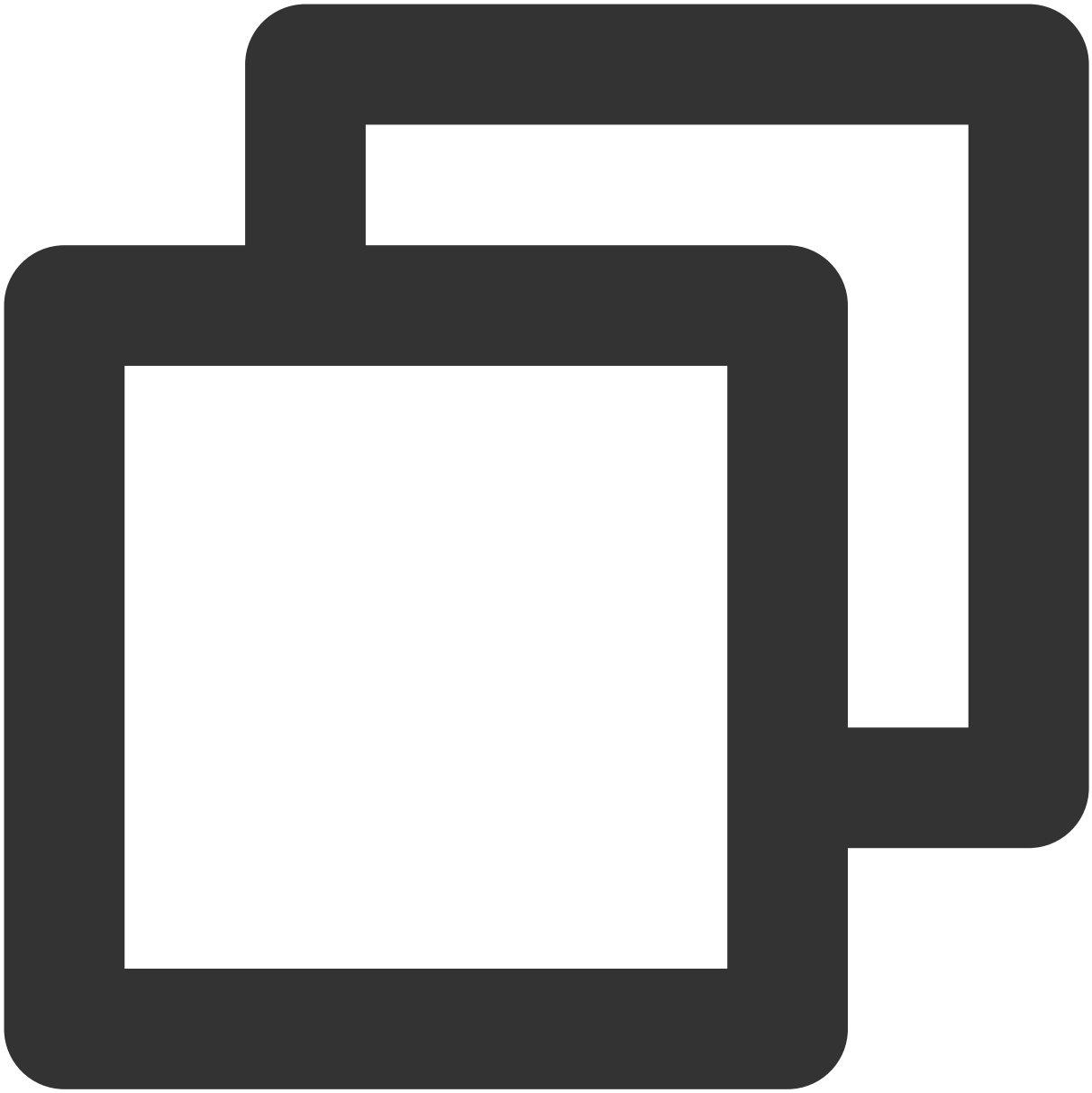
2. Add the following attribute to the `application` tag in the `AndroidManifest.xml` file



```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:networkSecurityConfig="@xml/network_security_config"
                    ... >
        ...
    </application>
</manifest>
```

iOS

Add the following configuration to `Info.plist` in `iOS`



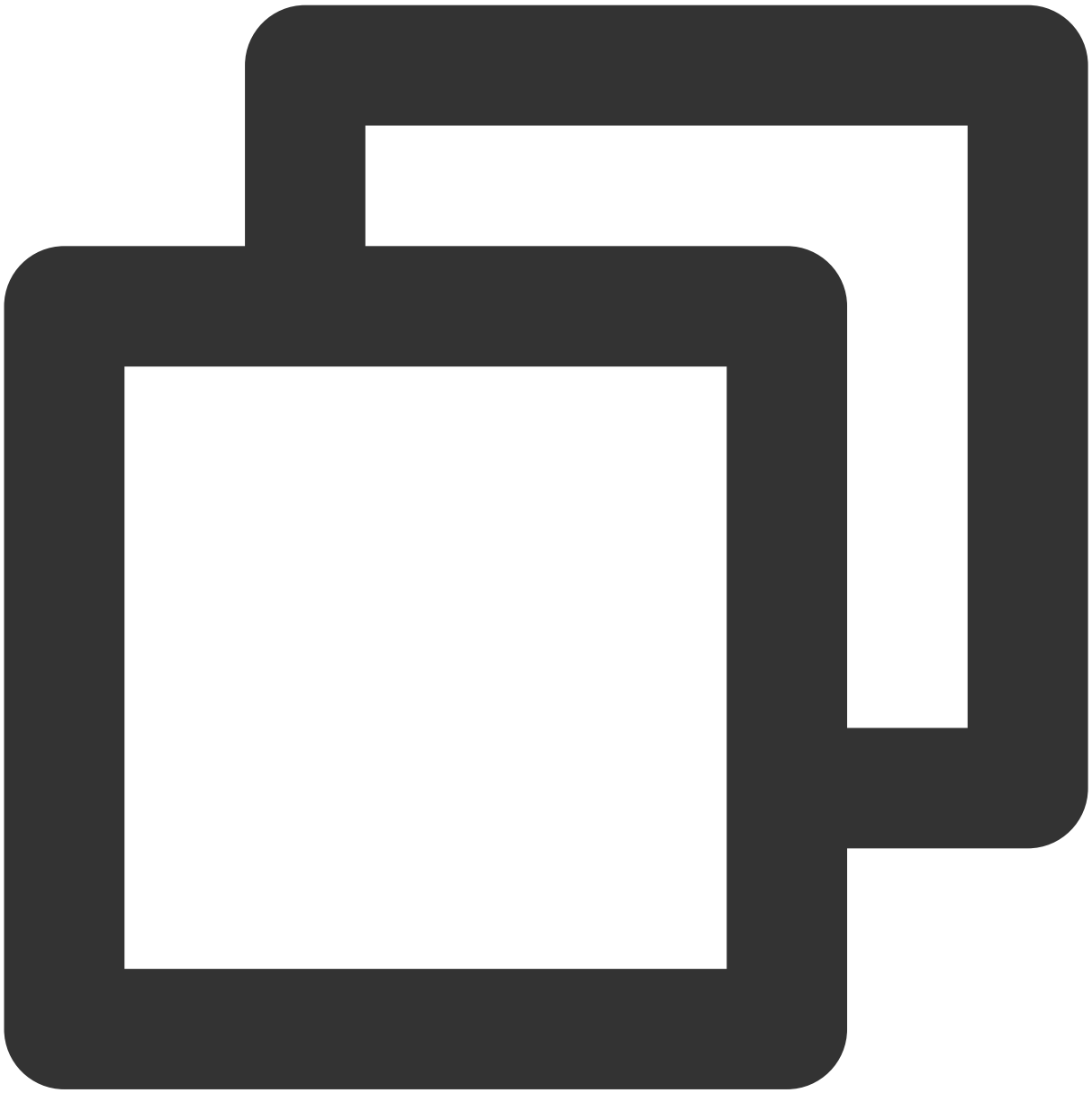
```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```

Note：

If you want to run the provided `Demo` in the `SDK`, you should also declare permission to use the photo library.

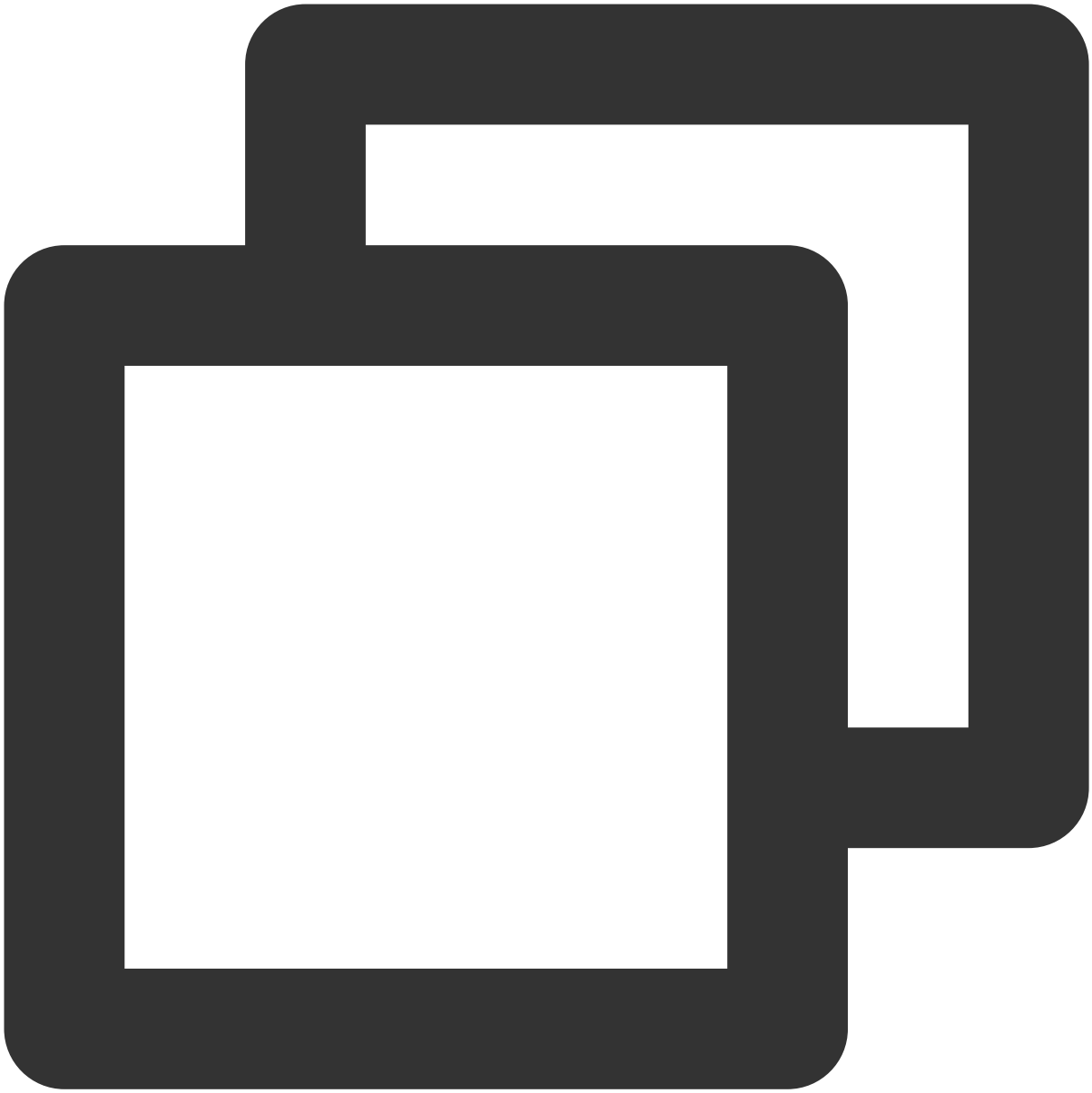
Usage

1. Import the file



```
import 'package:vod_upload_flutter/txugc_publish.dart';
```

2. Create an object



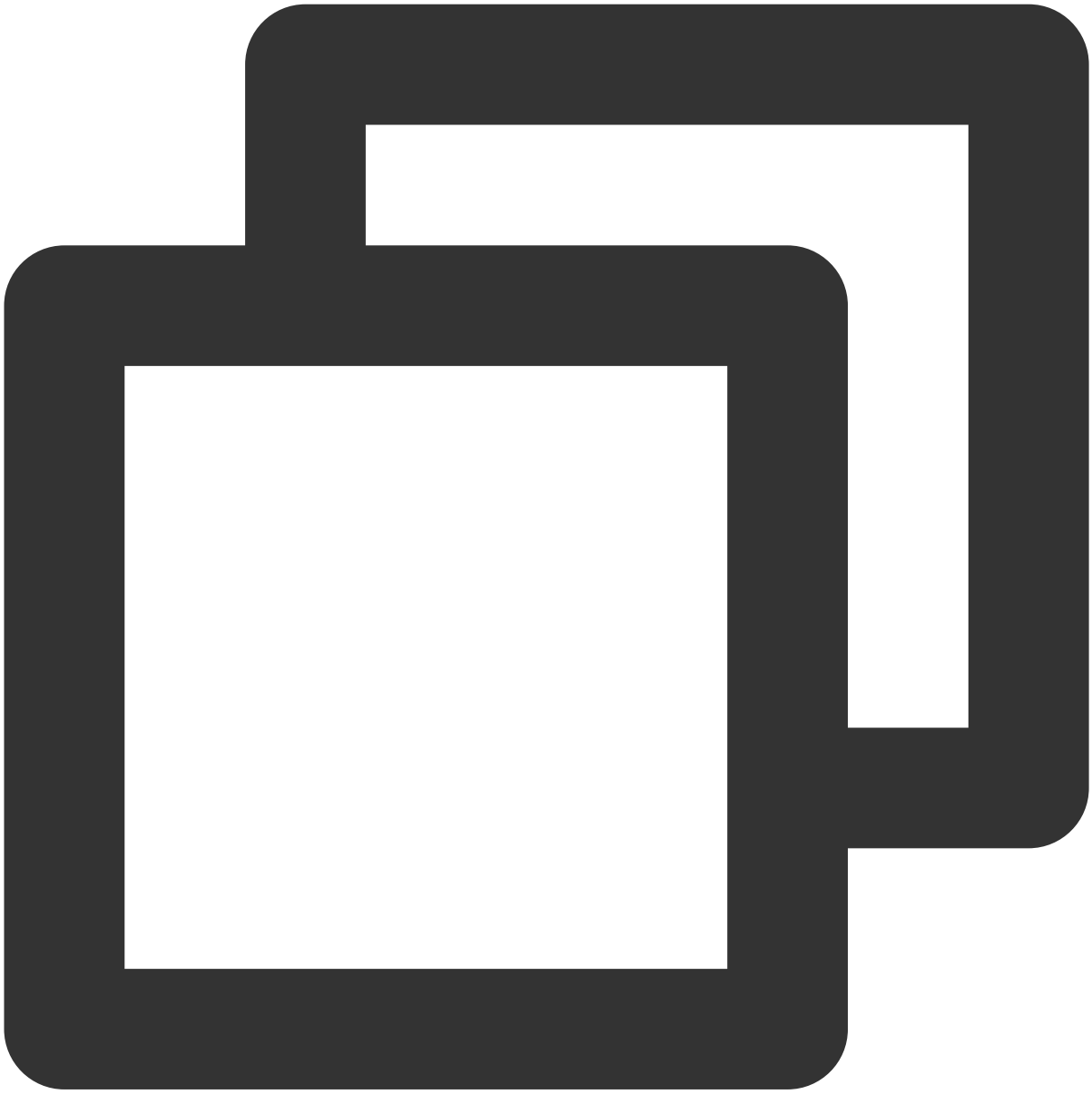
```
var uploader = TXUGCPublish(
    id: "",
);
```

Note：

The `id` can be any string as long as it is `unique`. The main purpose is to map the Flutter object to the native layer object.

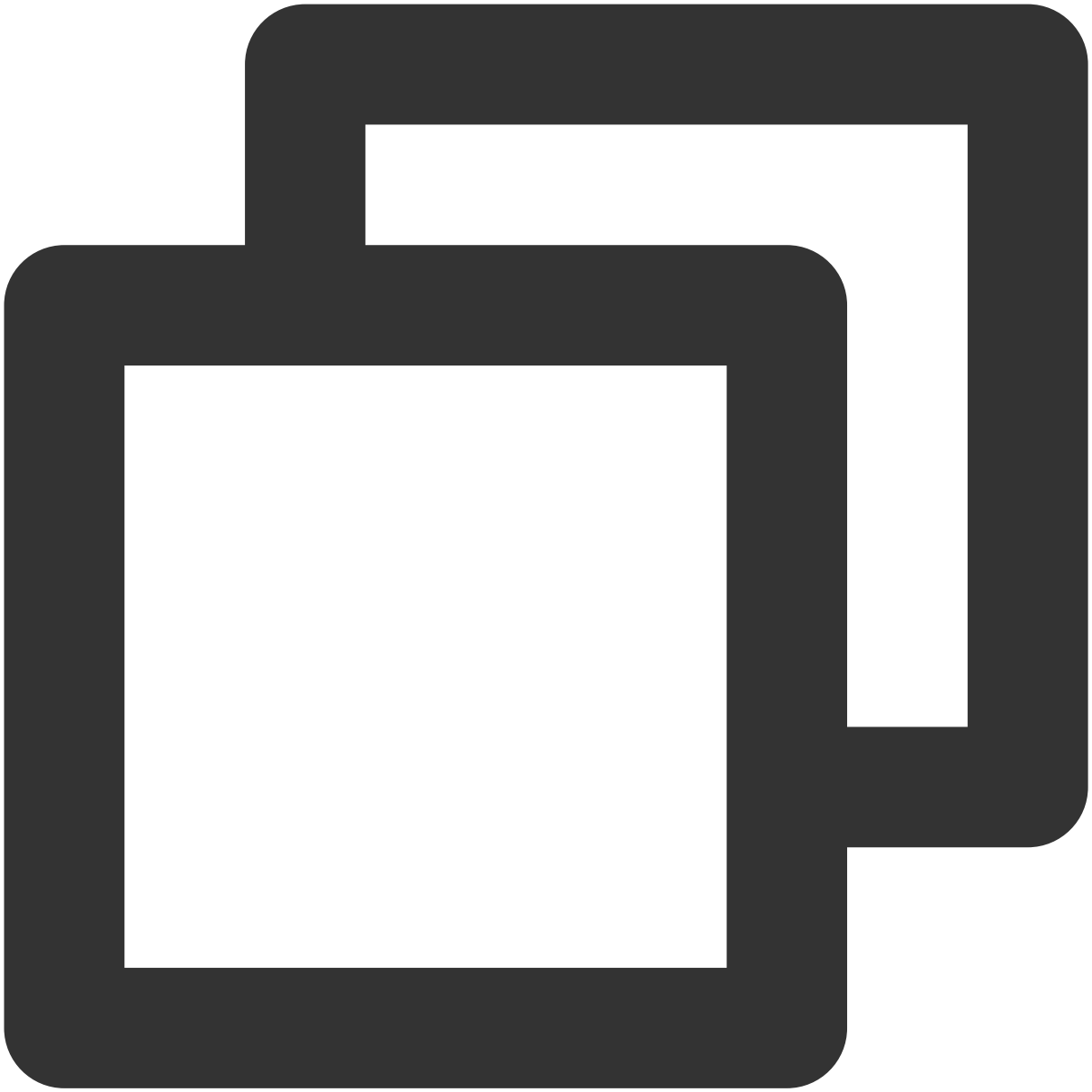
API

Upload Video



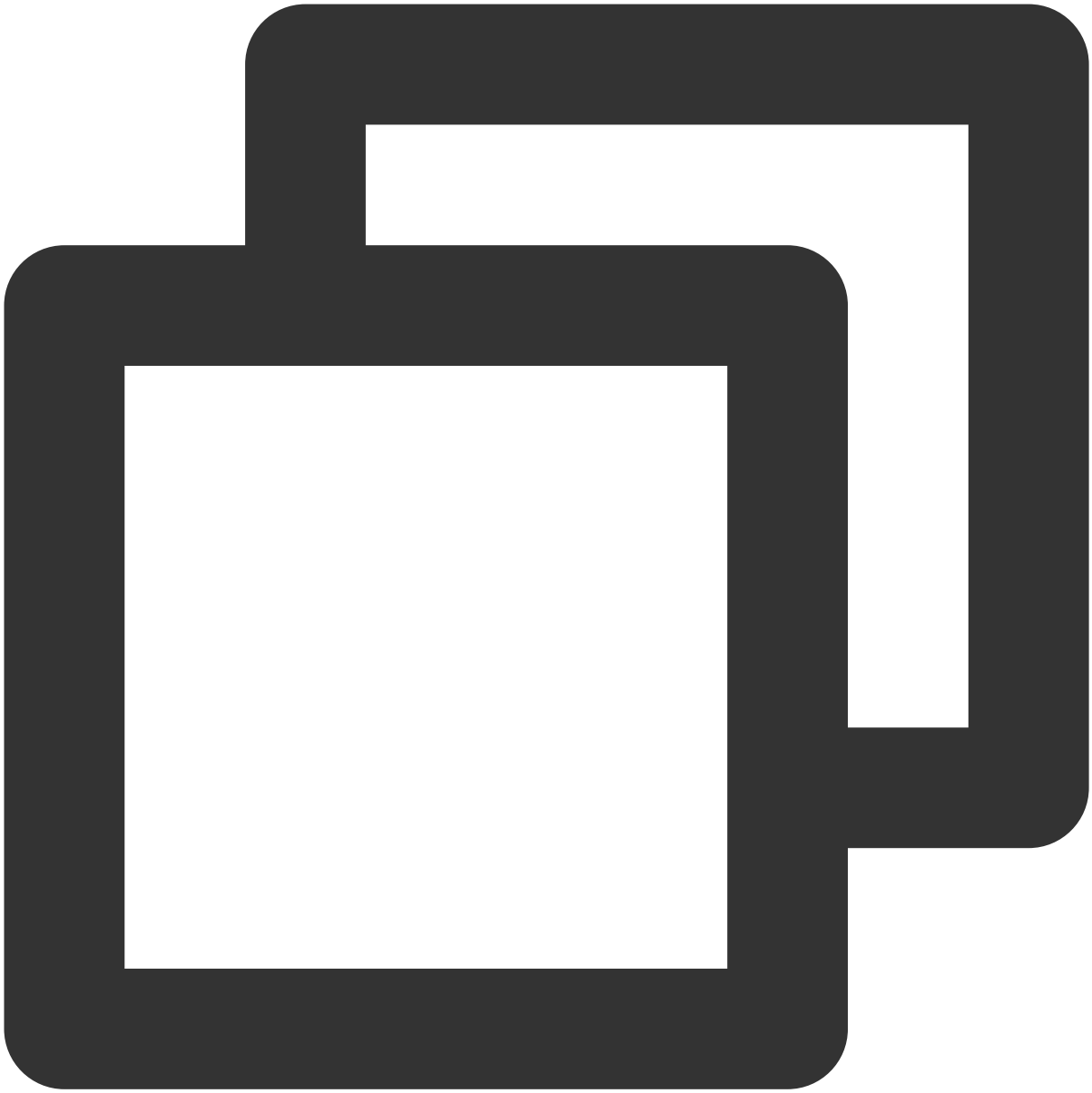
```
uploader.publishVideo(TXPublishParam(
      signature: "",
      videoPath: "",
      fileName: "",
));
```

Cancel Video Upload



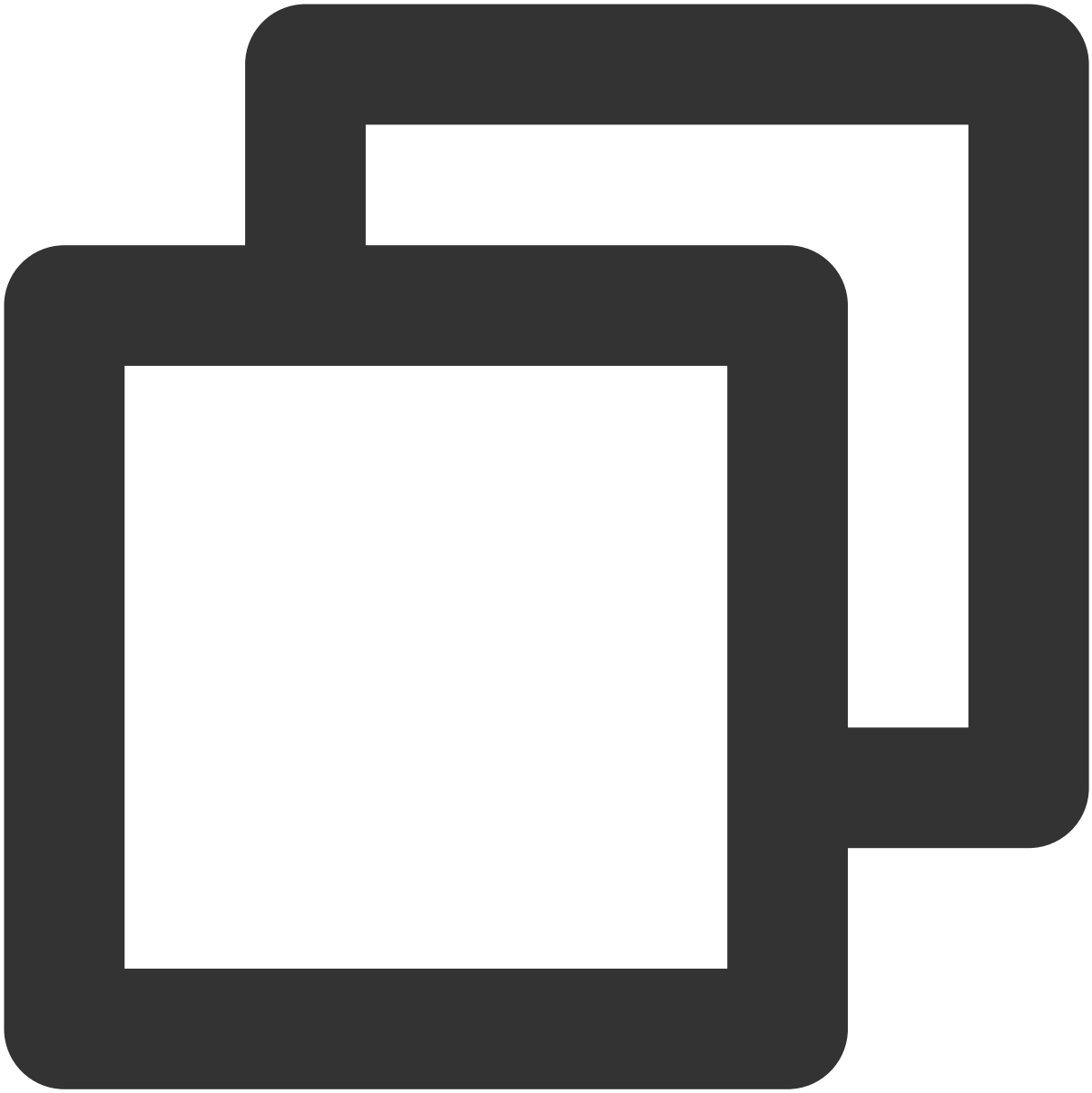
```
uploader.cancelUploadVideo();
```

Resume Video Upload



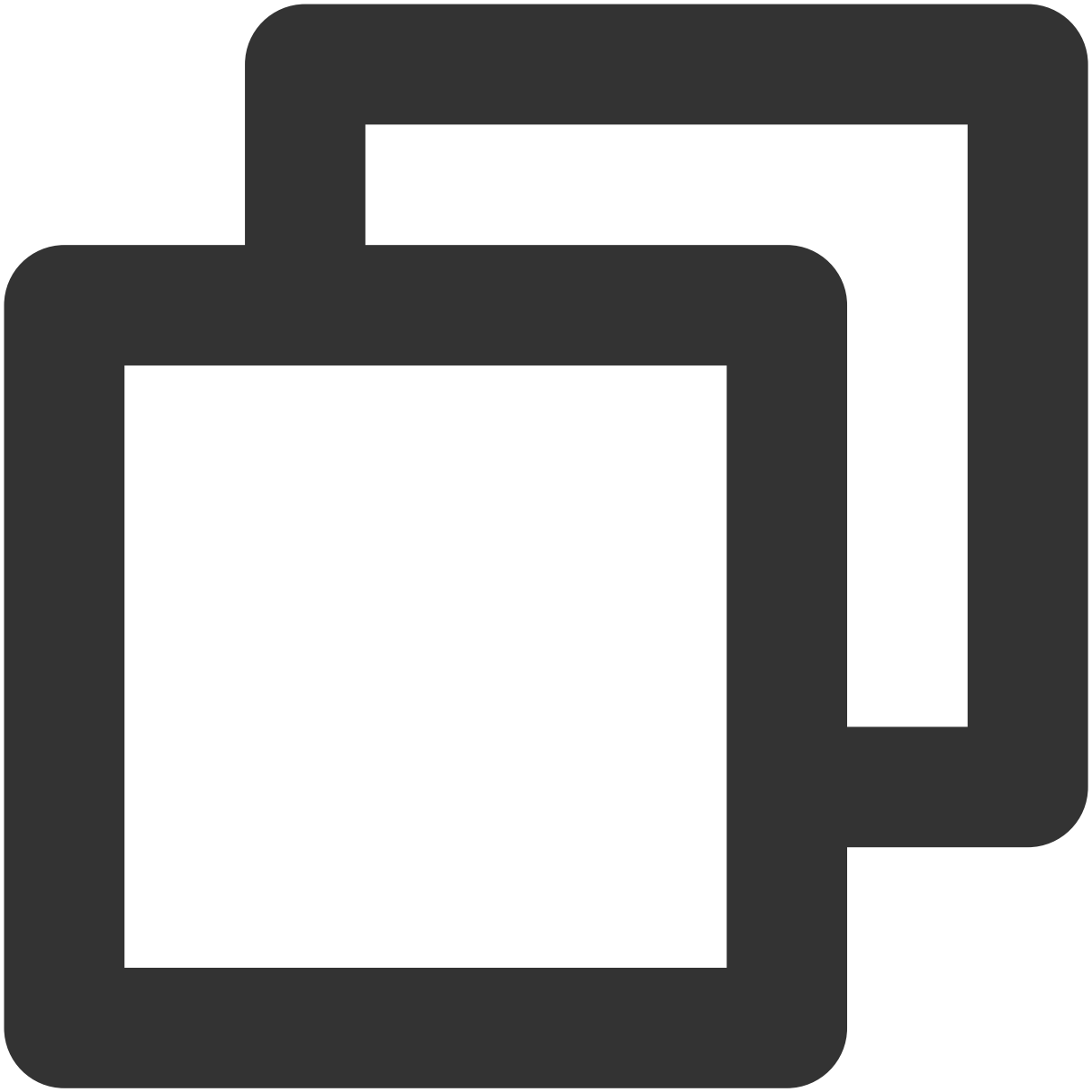
```
uploader.resumeUploadVideo(TXPublishParam(
      signature: "",
      videoPath: "",
      fileName: "",
));
```

Upload Media File



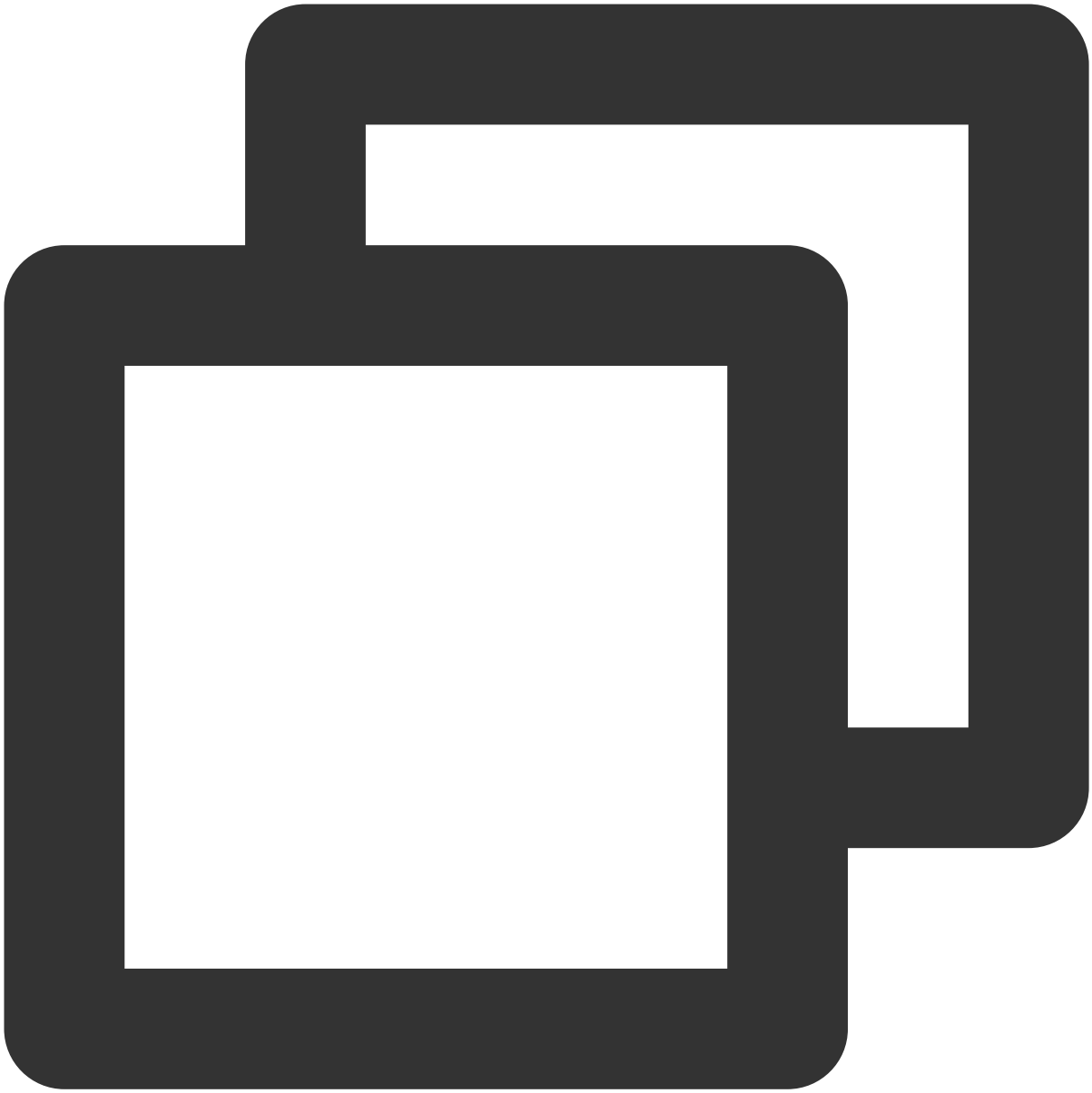
```
uploader.publishMedia(TXMediaPublishParam(
      signature: "",
      mediaPath: "",
      fileName: "",
));
```

Cancel Media File Upload



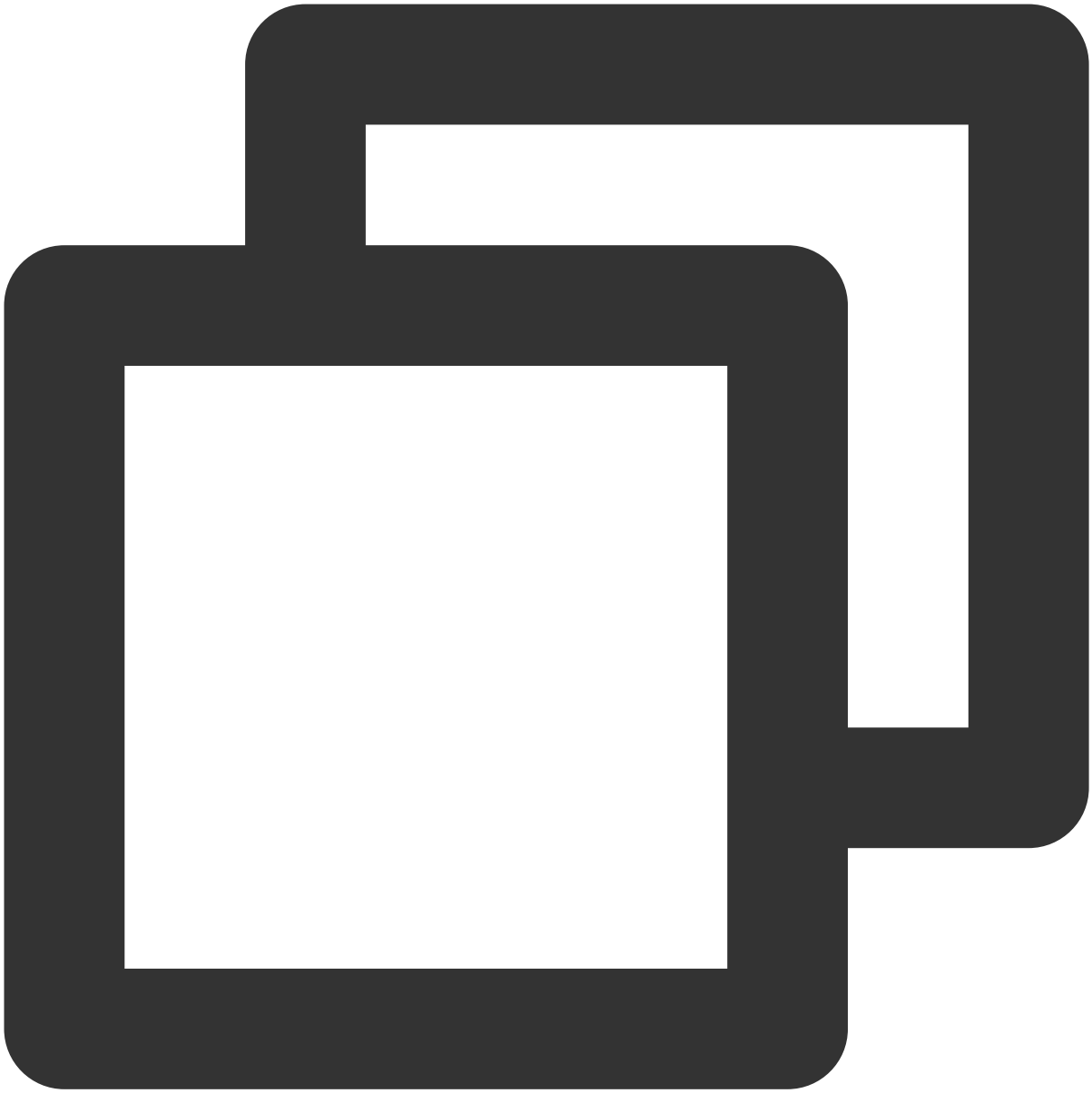
```
uploader.cancelUploadMedia();
```

Resume Media File Upload



```
uploader.resumeUploadMedia(TXMediaPublishParam(
      signature: "",
      mediaPath: "",
      fileName: "",
));
```

Prepare Upload

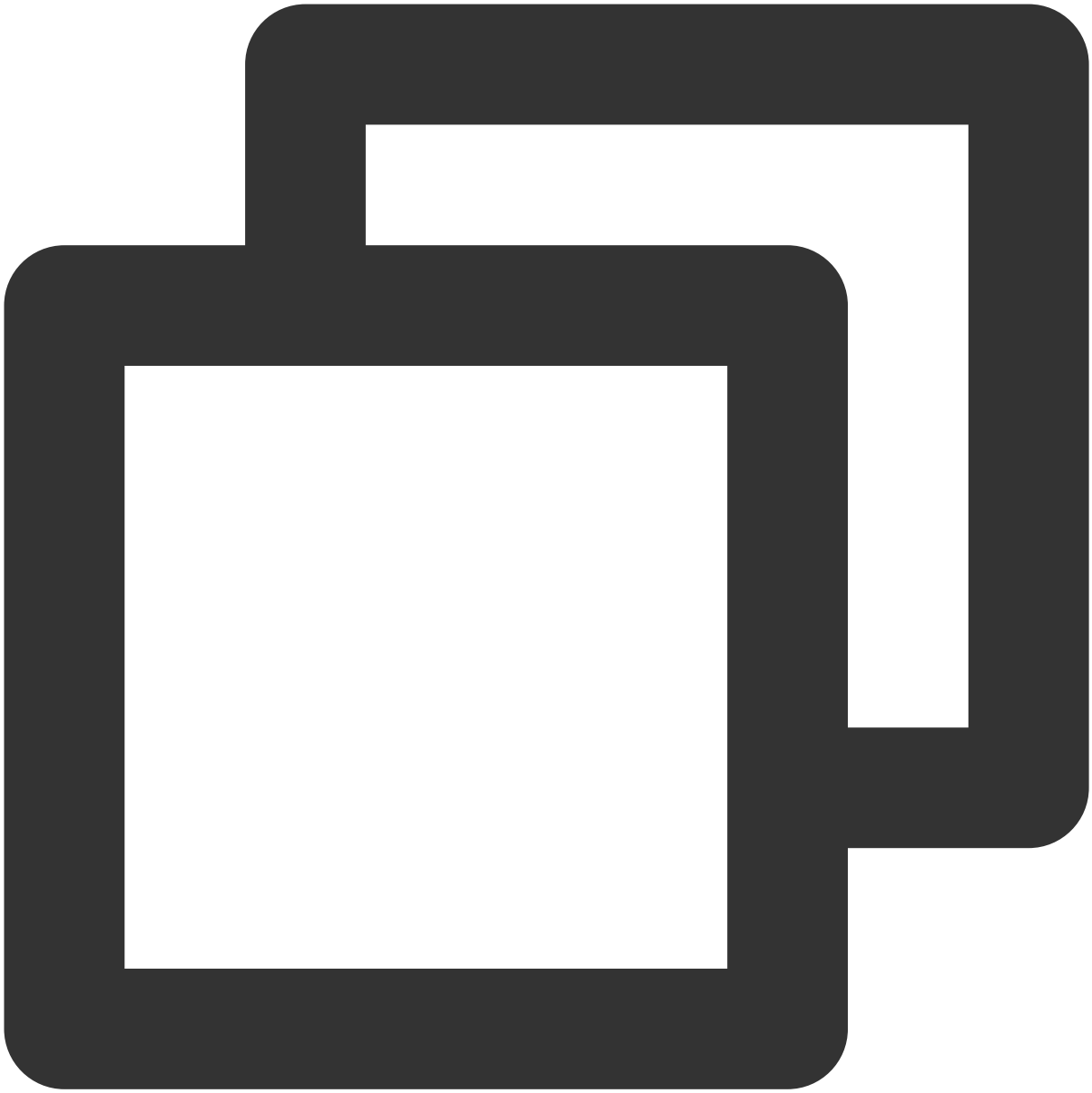


```
TXUGCPublish.prepareUpload(signature, callback);
```

Note：

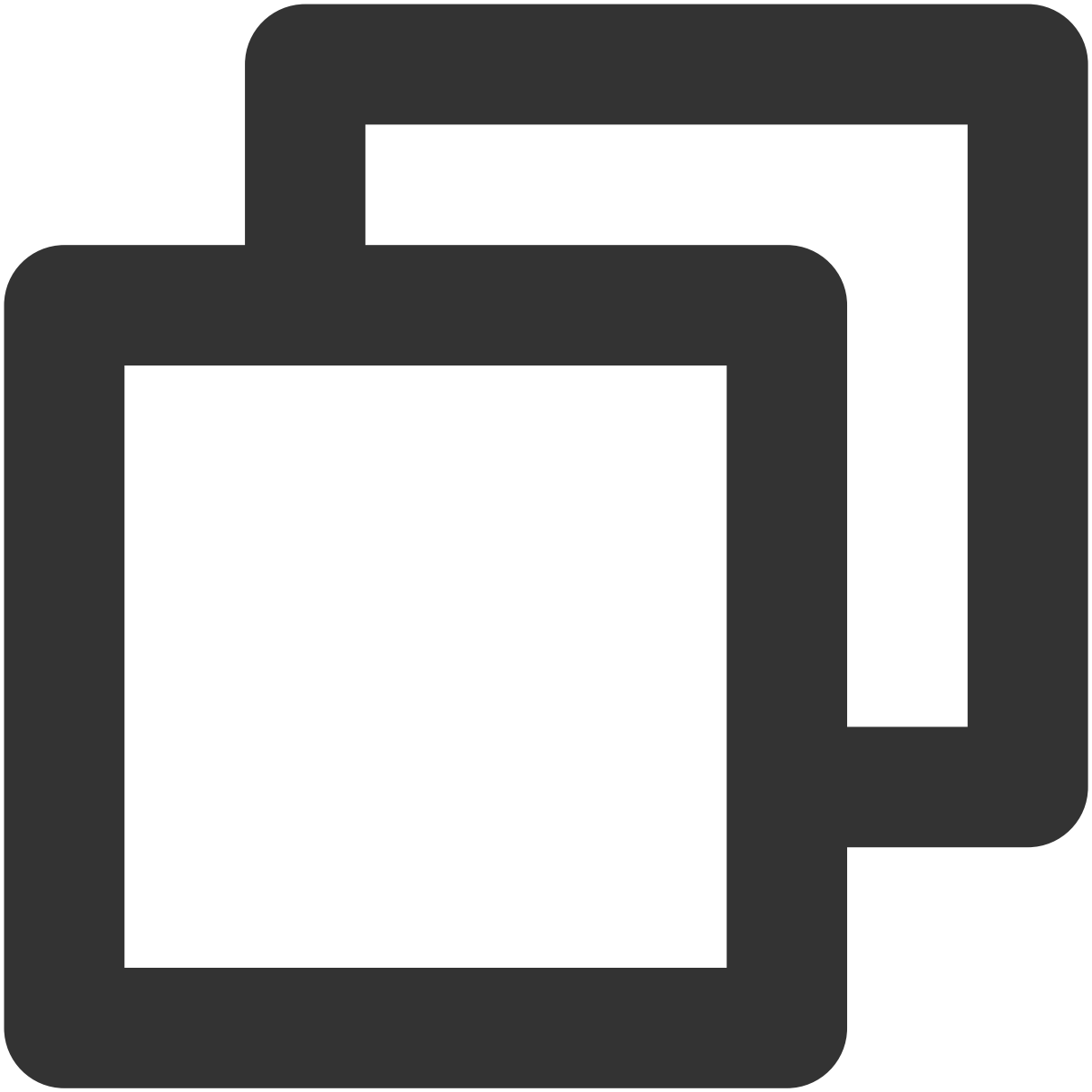
Prepare upload is a `static method`

Get Upload Information



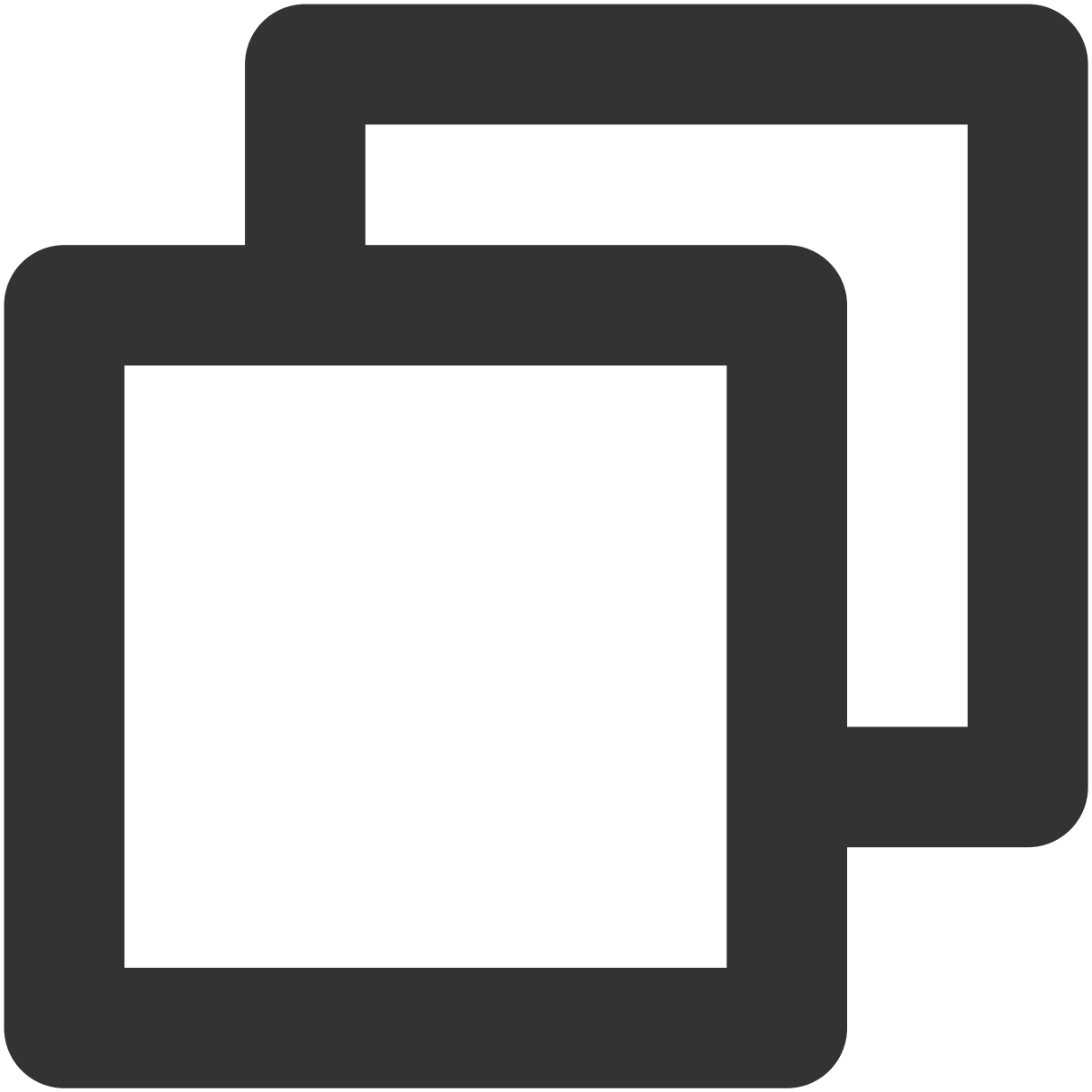
```
// On Android, you can only get information during the upload process, while on iOS
uploader.getStatusInfo();
```

Report `AppId`



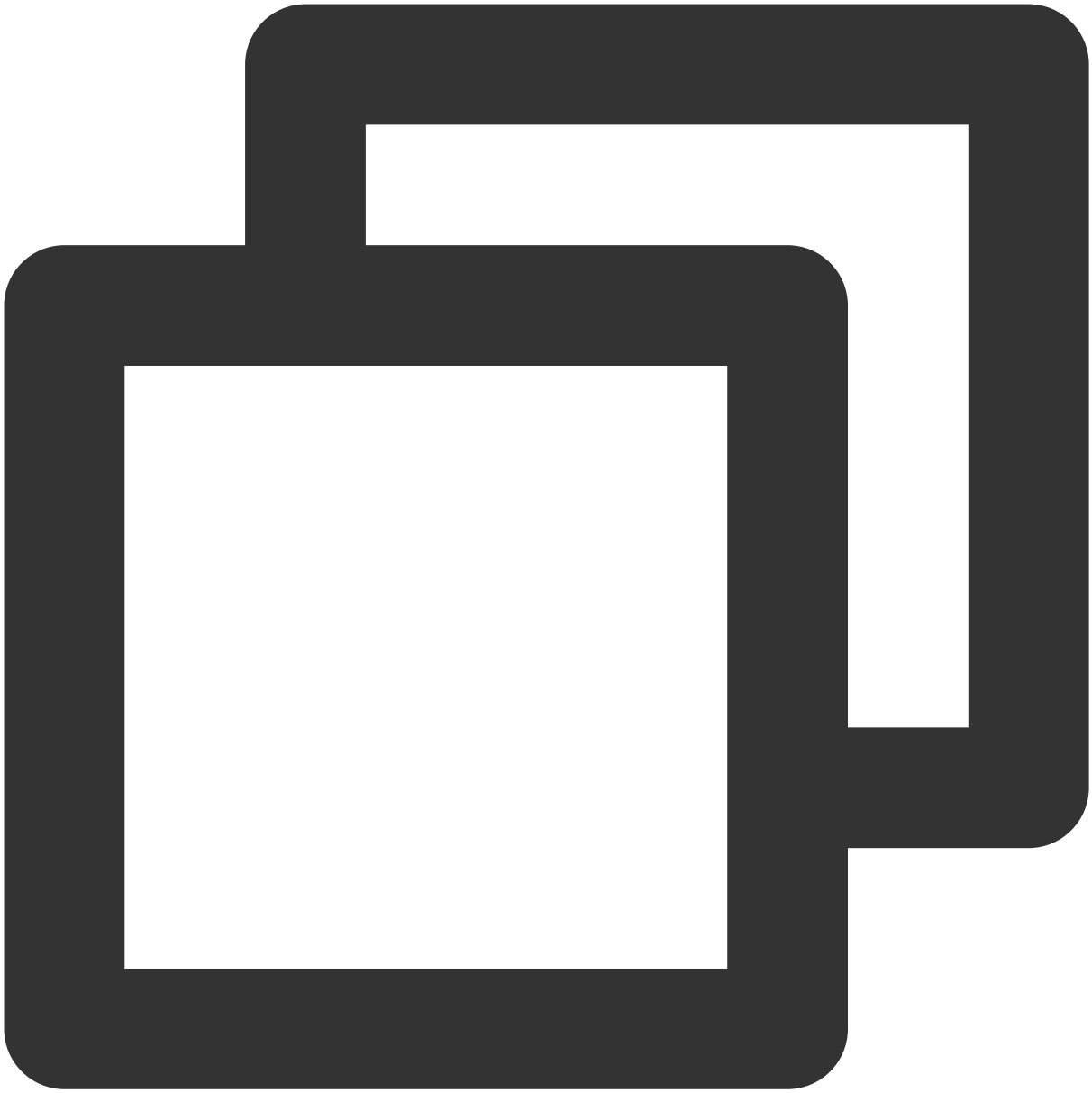
```
uploader.setAppId(appId);
```

Set Video Upload Callback



```
uploader.setVideoListener(listener);
```

Set Media Upload Callback



```
uploader.setMediaListener(listener);
```

Callback Interfaces and Parameter Explanations

Video Upload Parameters

`TXPublishParam`

Field	Type	Required	Explanation	Default

				Value
signature	string	✓	Signature	null
videoPath	string	✓	Video path	null
fileName	string	✓	File name	null
enableResume	boolean	✗	Enable resumable upload	true
enableHttps	boolean	✗	Enable HTTPS	false
coverPath	string	✗	Cover image	null
enablePreparePublish	boolean	✗	Enable prepare upload (can be manually triggered if disabled)	true
sliceSize	integer	✗	Chunk size (minimum 1M, maximum 10M, default 0, which means the file size divided by 10)	0
concurrentCount	integer	✗	Concurrent number of chunk uploads (if <=0, the default value of 2 will be used)	-1

Media Upload Parameters

`TXMediaPublishParam`

Field	Type	Required	Explanation	Default Value
signature	string	✓	Signature	null
mediaPath	string	✓	Media file path	null
fileName	string	✓	File name	null
enableResume	boolean	✗	Enable resumable upload	true
enableHttps	boolean	✗	Enable HTTPS	false
coverPath	string	✗	Cover image	null
enablePreparePublish	boolean	✗	Enable prepare upload (can be manually triggered if disabled)	true
sliceSize	integer	✗	Chunk size (minimum 1M, maximum 10M, default 0, which means the file size divided by 10)	0

concurrentCount	integer	✖	Concurrent number of chunk uploads (if <=0, the default value of 2 will be used)	-1

Video Upload Callback

`ITXVideoPublishListener`

Method	Return Type	Explanation
onPublishProgress	void	Upload progress callback
onPublishComplete	void	Upload completion callback

Parameter explanation:

`onPublishProgress`

Parameter	Type	Explanation
uploadBytes	integer	Number of bytes uploaded
totalBytes	integer	Total number of bytes

`onPublishComplete`

Parameter	Type	Explanation
result	TXPublishResult	Upload result

`TXPublishResult`

Parameter	Type	Explanation
retCode	integer	Error code
descMsg	string	Error description
videoId	string	Video file ID
videoURL	string	Video playback URL
coverURL	string	Cover image storage URL

Media File Upload Callback

`ITXMediaPublishListener`

Method	Return Type	Explanation
onMediaPublishProgress	void	Upload progress callback
onMediaPublishComplete	void	Upload completion callback

Parameter explanation:

`onMediaPublishProgress`

Parameter	Type	Explanation
uploadBytes	integer	Number of bytes uploaded
totalBytes	integer	Total number of bytes

`onMediaPublishComplete`

Parameter	Type	Explanation
result	TXMediaPublishResult	Upload result

`TXMediaPublishResult`

Parameter	Type	Explanation
retCode	integer	Error code
descMsg	string	Error description
mediaId	string	Media file ID
mediaURL	string	Media file URL

Prepare Upload Callback

`IPrepareUploadCallback`

Method	Return Type	Explanation
onLoading	void	Prepare upload start callback
onFinish	void	Prepare upload completion callback

Upload Status Information

`ReportInfo`

Field	Type	Explanation
reqType	string	Request type, indicating the current step
errCode	string	Error code
cosErrCode	string	COS upload error code
errMsg	string	Error message
reqTime	string	Request start time for the current step
reqTimeCost	string	Time spent on the current step
fileSize	string	File size
fileType	string	File type
fileName	string	File name
fileId	string	File ID
appId	string	VOD App ID set through TXUGCPublish
reqServerIp	string	IP address accessed during the current step
reportId	string	Custom report ID provided by the customer, can be passed through the TXUGCPublish constructor
reqKey	string	Request key, usually composed of the last modification time of the file and the start time of this upload
vodSessionKey	string	Session key from the VOD server, obtained from the upload request interface
cosRegion	string	Region accessed during the current upload
requestId	string	Request ID for the current COS upload
cosVideoPath	string	Path for the current COS video upload
vodErrCode	integer	Signaling request error code
useHttpDNS	integer	Whether to use httpDns for domain name resolution
useCosAcc	integer	Whether COS domain name acceleration is enabled
tcpConnTimeCost	integer	Time spent on connecting to the server in the current step

recvRespTimeCost	integer	Time spent on receiving server response in the current step

ビデオAI
オーディオビデオコンテンツ審査

最終更新日：：2023-10-26 17:39:30

オーディオビデオコンテンツ審査は、AIの力を借りて、オーディオビデオのコンテンツに対してインテリジェントな審査を行う機能であり、オフラインタスクです。タスクの実行結果の中には、審査の評点、審査のアドバイス、疑わしいビデオの断片などが含まれています。「審査のアドバイス」に基づき、ビデオ管理者はビデオの公開を許可するかを決定でき、不正なビデオによる法的リスクやブランドのダメージを効果的に回避することができます。
VODでは画面上の画像、画面内のテキスト、音声内のテキスト、声のコンテンツの4種類を対象に審査を行うことができます。

オブジェクト	審査タグ
画面上の画像	ポルノ（Porn）
	暴力（Terror）
音声	喘ぎ声（Moan）
音声内のテキスト（ASR）	ポルノ（Porn）
	暴力（Terror）
画面内のテキスト（OCR）	ポルノ（Porn）
	暴力（Terror）

オーディオビデオ審査結果部分のフィールドの説明：

フィールド名	タイプ	意味
Confidence	Float	審査の評点（0～100）。評点が高いほど、疑わしさが大きくなります。
Suggestion	String	審査のアドバイスには、pass、review、blockの3種類があります。 pass：疑わしさが低いため、そのまま合格させることをお勧めします。 review：疑わしさが高いため、手動で再審査を行うことをお勧めします。 block：非常に疑わしいため、そのままブロックすることをお勧めします。

Form	String	審査形式には、以下の種類があります。 Image：画面上の画像。 Voice：音声。 OCR：画面内の文字。 ASR：音声内の文字。
Label	String	審査タグには、以下の種類があります。 Porn：ポルノ。 Terror：暴力。 Moan：喘ぎ声。

オーディオビデオ審査テンプレート

オーディオビデオ審査のパラメータによって、審査タスクで具体的にどの審査タグを検出するかを制御できます。VODでは、ビデオ審査テンプレートを使用して審査のパラメータグループを表示し、ビデオ審査テンプレートによって、審査タスクの中で以下のどのタグ（1つまたは複数）を検出するかを指定できます。
ポルノ（Porn）
暴力（Terror）
喘ぎ声（Moan）
一般的な操作の組み合わせを対象に、VODでは、プリセットのオーディオビデオ審査テンプレートを提供しています。また、サーバーAPIを呼び出してカスタマイズしたビデオ審査テンプレートを作成し、管理することができます。

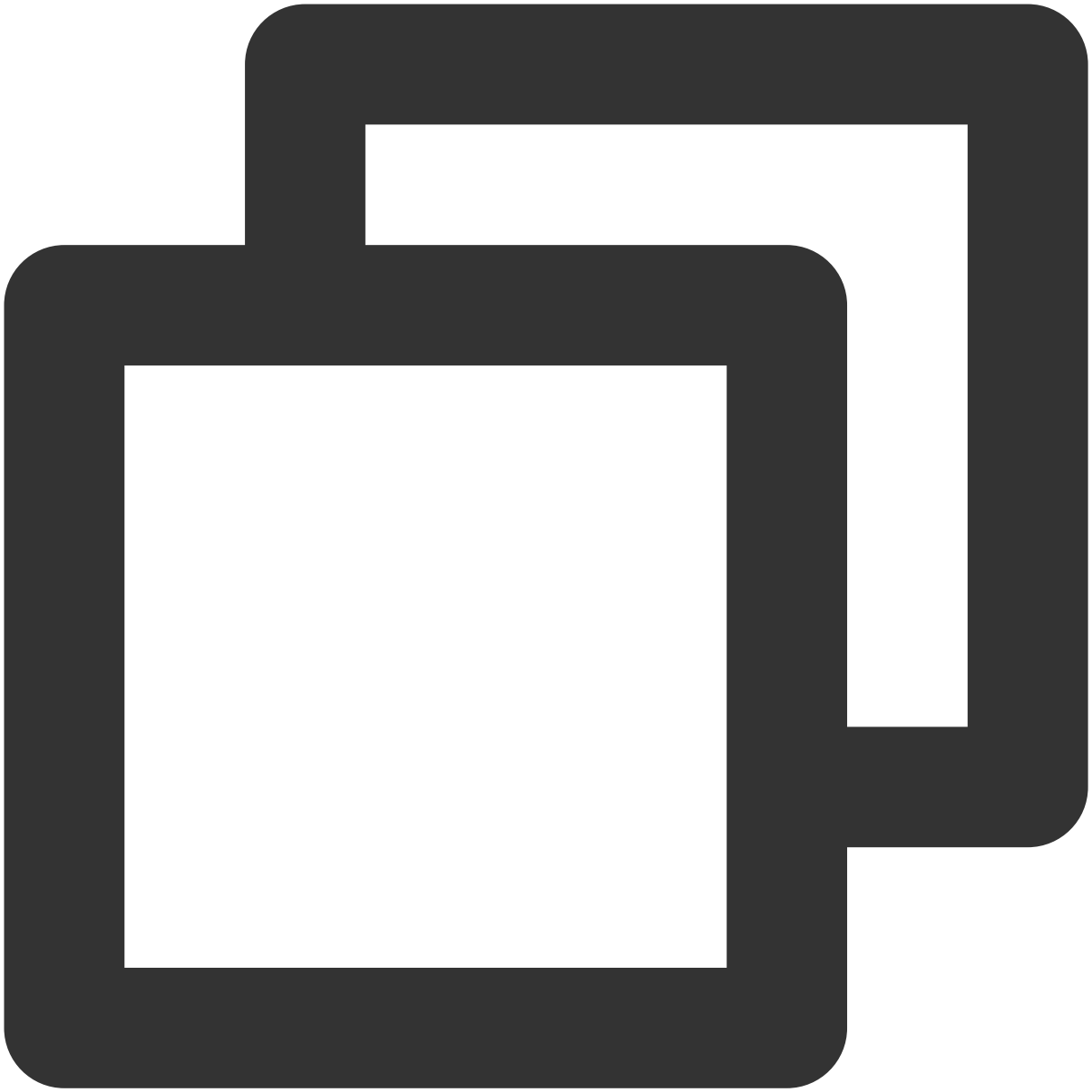
タスクの開始

オーディオビデオコンテンツ審査タスクの開始には、「サーバーAPIから直接開始」、「コンソールから直接開始」、「アップロード時に実行したいタスクを指定」の3種類の方法があります。具体的には、ビデオ処理のタスクの開始をご参照ください。
以下は、各方法のオーディオビデオコンテンツ審査タスク開始についての説明です。
サーバーAPIによって直接開始します。サーバーAPIReviewAudioVideoを呼び出してタスクを開始します。
コンソールから直接開始します。コンソールガイドのオーディオビデオ審査をご参照ください。
サーバーからのアップロード時にタスクを指定：コンソールでタスクフローを追加し、タスクフローの中でコンテンツ審査を有効にします。アップロードの申請の中の `procedure` パラメータでこのタスクフローを指定します。
クライアントからのアップロード時にタスクを指定：コンソールでタスクフローを追加し、タスクフローの中でコンテンツ審査を有効にします。クライアントからのアップロード署名の中の `procedure` でこのタスクフローを指定します。

コンソールからのアップロード：コンソールでタスクフローを追加し、タスクフローの中でコンテンツ審査を有効にします。コンソールでビデオをアップロードし、アップロードと同時にビデオに対する処理操作を実行を選択し、ビデオのアップロードを指定してからこのタスクフローを実行します。

結果の取得

ビデオ審査タスクを開始した後、非同期のオーディオビデオ審査の完了を待機するか、またはタスク確認を同期的に実行する方法のどちらかでビデオ審査タスクの実行結果を取得できます。以下は、審査タスクの開始後、通常のコールバック方式での結果通知の例となります（値がnullのフィールドは省略されます）。



```
{
    "EventType": "ReviewAudioVideoComplete",
    "ReviewAudioVideoCompleteEvent": {
        "TaskId": "125xxxx-ReviewAudioVideo-07edbc78ba20563cdf2362cffbf4aa0ct",
        "Status": "FINISH",
        "ErrCodeExt": "",
        "Message": "SUCCESS",
        "Input":{
            "FileId": "387702130626135215"
        },
        "Output":{
```

```
            "Suggestion": "block",
            "Label": "Porn",
            "Form": "Image",
            "SegmentSet":[
                {
                    "StartTimeOffset": 0,
                    "EndTimeOffset": 1,
                    "Confidence": 99,
                    "Suggestion": "block",
                    "Label": "Porn",
                    "SubLabel": "SexyBehavior",
                    "Form": "Image",
                    "AreaCoordSet": [],
                    "Text": "",
                    "KeywordSet": [],
                    "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                    "PicUrlExpireTime": "2023-01-16T03:06:16.039Z"
                },
                {
                    "StartTimeOffset": 1,
                    "EndTimeOffset": 2,
                    "Confidence": 99,
                    "Suggestion": "block",
                    "Label": "Porn",
                    "SubLabel": "SexyBehavior",
                    "Form": "Image",
                    "AreaCoordSet": [],
                    "Text": "",
                    "KeywordSet": [],
                    "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                    "PicUrlExpireTime": "2023-01-16T03:06:17.039Z"
                },
                {
                    "StartTimeOffset": 2,
                    "EndTimeOffset": 3,
                    "Confidence": 99,
                    "Suggestion": "block",
                    "Label": "Porn",
                    "SubLabel": "SexyBehavior",
                    "Form": "Image",
                    "AreaCoordSet": [],
                    "Text": "",
                    "KeywordSet": [],
                    "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                    "PicUrlExpireTime": "2023-01-16T03:06:18.039Z"
                },
                {
```

```
                    "StartTimeOffset": 3,
                    "EndTimeOffset": 4,
                    "Confidence": 99,
                    "Suggestion": "block",
                    "Label": "Porn",
                    "SubLabel": "SexyBehavior",
                    "Form": "Image",
                    "AreaCoordSet": [],
                    "Text": "",
                    "KeywordSet": [],
                    "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                    "PicUrlExpireTime": "2023-01-16T03:06:19.039Z"
                },
                {
                    "StartTimeOffset": 4,
                    "EndTimeOffset": 5,
                    "Confidence": 99,
                    "Suggestion": "block",
                    "Label": "Porn",
                    "SubLabel": "SexyBehavior",
                    "Form": "Image",
                    "AreaCoordSet": [],
                    "Text": "",
                    "KeywordSet": [],
                    "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                    "PicUrlExpireTime": "2023-01-16T03:06:20.039Z"
                },
                {
                    "StartTimeOffset": 5,
                    "EndTimeOffset": 6,
                    "Confidence": 99,
                    "Suggestion": "block",
                    "Label": "Porn",
                    "SubLabel": "SexyBehavior",
                    "Form": "Image",
                    "AreaCoordSet": [],
                    "Text": "",
                    "KeywordSet": [],
                    "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                    "PicUrlExpireTime": "2023-01-16T03:06:21.039Z"
                },
                {
                    "StartTimeOffset": 6,
                    "EndTimeOffset": 7,
                    "Confidence": 99,
                    "Suggestion": "block",
                    "Label": "Porn",
```

```
                    "SubLabel": "SexyBehavior",
                    "Form": "Image",
                    "AreaCoordSet": [],
                    "Text": "",
                    "KeywordSet": [],
                    "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                    "PicUrlExpireTime": "2023-01-16T03:06:22.039Z"
                },
                {
                    "StartTimeOffset": 7,
                    "EndTimeOffset": 8,
                    "Confidence": 99,
                    "Suggestion": "block",
                    "Label": "Porn",
                    "SubLabel": "SexyBehavior",
                    "Form": "Image",
                    "AreaCoordSet": [],
                    "Text": "",
                    "KeywordSet": [],
                    "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                    "PicUrlExpireTime": "2023-01-16T03:06:23.039Z"
                },
                {
                    "StartTimeOffset": 8,
                    "EndTimeOffset": 9,
                    "Confidence": 99,
                    "Suggestion": "block",
                    "Label": "Porn",
                    "SubLabel": "SexyBehavior",
                    "Form": "Image",
                    "AreaCoordSet": [],
                    "Text": "",
                    "KeywordSet": [],
                    "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                    "PicUrlExpireTime": "2023-01-16T03:06:24.039Z"
                },
                {
                    "StartTimeOffset": 9,
                    "EndTimeOffset": 10,
                    "Confidence": 99,
                    "Suggestion": "block",
                    "Label": "Porn",
                    "SubLabel": "SexyBehavior",
                    "Form": "Image",
                    "AreaCoordSet": [],
                    "Text": "",
                    "KeywordSet": [],
```

```
                    "Url": "https://251000800.vod2.myqcloud.com/1a168d62vodcq251000
                    "PicUrlExpireTime": "2023-01-16T03:06:25.039Z"
                }
            ],
            "SegmentSetFileUrl": "http://251000800.vod2.myqcloud.com/a8800b40vodtra
            "SegmentSetFileUrlExpireTime": "2022-10-12T07:01:07.695Z"
        },
        "SessionContext": "",
        "SessionId": ""
    }
}
```


コールバック結果の中で、`ReviewAudioVideoCompleteEvent.Output` はオーディオビデオ審査結果のアウトプットであり、`Output.Suggestion` は全体的な審査のアドバイスを表します。ここでは `block` すなわちそのままブロックすることを推奨しています。`Output.Label=Porn` と `Output.Form=Image` は、違反の内容がビデオ画面に含まれる性的な情報である可能性が最も高いことを示しています。

1つのオーディオビデオに複数の違反部分が存在する可能性があり、`Output.SegmentSet` ではそのうち最初の10部分をリストアップします。（全体の違反結果はリンクの有効期間内に `Output.SegmentSetFileUrl` から取得できます）。

各違反部分の `StartTimeOffset` と `EndTimeOffset` は、その部分の元のビデオ内での開始・終了時間を表します。`SubLabel` はその部分の具体的な違反内容を表します。

画面内の文字と音声内の文字の認識：

`Text` はその部分から認識された完全な文字コンテンツを表します。

`KeywordSet` はヒットした違反キーワードのリストを表します。

ビデオ画面（人および物体）と画面内の文字の認識：

`AreaCoordSet` は違反対象のエリア座標を表します。

`Url` は違反画面のスクリーンキャプチャのリンクです。

`PicUrlExpireTime` は `Url` の有効期限であり、これを過ぎるとリンクにアクセスできなくなります。

ビデオコンテンツの分析

最終更新日：：2023-10-26 17:39:30

ビデオコンテンツ分析は、AIの力を借りてオーディオビデオコンテンツに対してインテリジェントな分析を行う機能であり、オフラインタスクです。オーディオビデオコンテンツ分析を使用することで、ビデオのクラシフィケーション（分類）、タグづけ、カバー画像のキャプチャなどに対してインテリジェントなアドバイスを与え、ビデオプラットフォームでビデオを的確かつ効率的に管理ができるようにサポートします。
オーディオビデオコンテンツ分析には、以下の機能が含まれます。

機能名	説明
スマート分類	ビデオが属するカテゴリーを提案します。現在は次の10余りのカテゴリーがあります。 ニュース、エンターテインメント、ゲーム、テクノロジー、グルメ、スポーツ、旅行、アニメ漫画、ダンス、ミュージック、映画TV、自動車など。
インテリジェントタグ	ビデオに付けることができるタグを提案します。現在は計3000種余りのタグがあり、例えば次のようなものがあります。 ゲーム、交通手段、ミュージシャン、レース、ペット、ドラム、自転車、World of Warcraft、コンピュータ、学校、ジャケットなど。
サムネイル画像のスマート生成	ビデオの中から1枚または何枚かのスクリーンキャプチャを選定し、カバー画像としての採用を推奨します。
フレーム別インテリジェントタグ	ビデオの1フレームの画面ごとに付けることができるタグを提案します。現在は計1000種余りのタグがあり、例えば次のようなものがあります。 モダンダンス、水上スポーツ、ステーキ、ベビー、子猫、一年生植物、駆逐艦、漫画、芝生、ウェディングドレス、多機能ホール、パスポートなど。

オーディオビデオコンテンツ分析テンプレート

オーディオビデオコンテンツ分析のパラメータによって、分析タスクで具体的にどの項目の分析操作を実行するかを制御することができます。VODではオーディオビデオコンテンツ分析テンプレートを使用して、インテリジェント分析のパラメータグループを表示します。
インテリジェントクラシフィケーションのアクティブ化の有無。
インテリジェントタグのアクティブ化の有無。
インテリジェントカバー画像のアクティブ化の有無。
フレーム別インテリジェントタグのアクティブ化の有無。
一般的な操作の組み合わせを対象に、Video on Demandでは、プリセットオーディオビデオコンテンツ分析テンプレートを提供しています。その他、サーバーAPIを呼び出してカスタマイズしたオーディオビデオコンテンツ分析

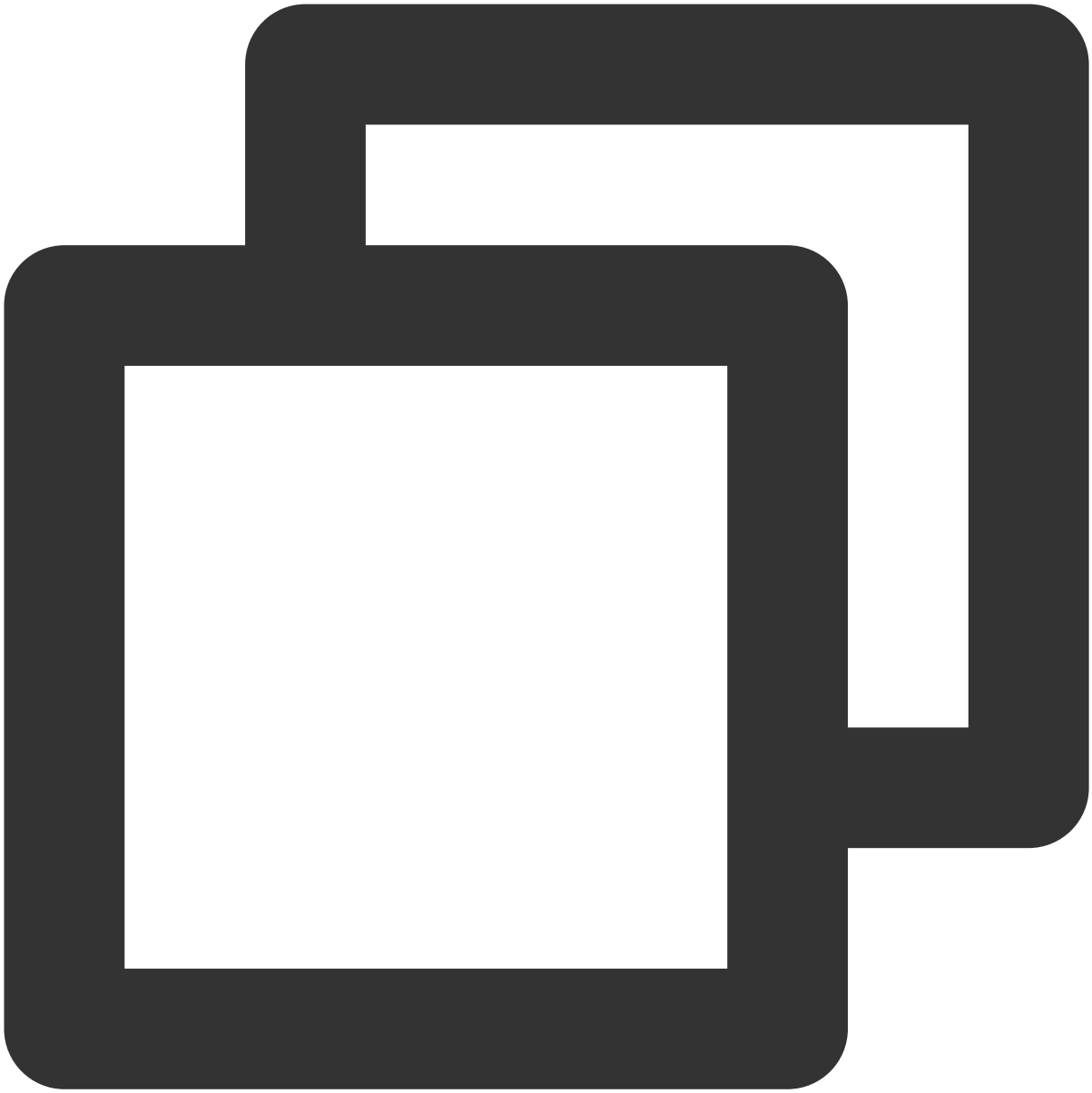
テンプレートを作成し、管理することができます。

タスクの開始

オーディオビデオコンテンツ分析タスクの開始には、「サーバーAPIから直接開始」、「コンソールから直接開始」、「アップロード時に実行したいタスクを指定」の3種類の方法があります。詳細内容は、ビデオ処理のタスクの開始をご参照ください。
以下は、各方法のオーディオビデオコンテンツ分析タスク開始についての説明です。
サーバーAPIProcessMediaの呼び出しによるタスク開始：リクエストの中の `AiAnalysisTask` パラメータでオーディオビデオコンテンツ分析テンプレートのテンプレートIDを指定します。
コンソールでのビデオに対するタスク開始：サーバーAPIを呼び出してタスクフローを作成し、タスクフローの中でオーディオビデオコンテンツ分析タスクを設定します（ `MediaProcessTask.AiAnalysisTask` の中で指定）。コンソールでこのタスクフローを使用してビデオ処理を開始します。
サーバーからのアップロード時にタスクを指定： サーバーAPIを呼び出してタスクフローを作成し、タスクフローの中でオーディオビデオコンテンツ分析タスクを設定します（ `MediaProcessTask.AiAnalysisTask` の中で指定）。アップロードの申請の `procedure` パラメータでこのタスクフローを指定します。
クライアントからのアップロード時にタスクを指定： サーバーAPIを呼び出してタスクフローを作成し、タスクフローの中でオーディオビデオコンテンツ分析タスクを設定します（ `MediaProcessTask.AiAnalysisTask` の中で指定）。クライアントからのアップロード署名の `procedure` でこのタスクフローを指定します。
コンソールからのアップロード：サーバーAPIを呼び出してタスクフローを作成し、タスクフローの中でオーディオビデオコンテンツ分析タスクを設定します（ `MediaProcessTask.AiAnalysisTask` の中で指定）。コンソールでビデオをアップロードし、アップロードと同時にビデオに対する処理操作を実行を選択して、ビデオアップロード後にこのタスクフローの実行を指定します。

結果の取得

オーディオビデオコンテンツ分析タスクを開始した後、非同期の結果通知を待機するか、またはタスク確認を同期的に実行する方法のどちらかでビデオコンテンツ分析タスクの実行結果を取得できます。以下は、ビデオコンテンツ分析タスクの開始後、通常のコールバック方式での結果通知の例となります（値がnullのフィールドは省略）。




```
{
    "EventType":"ProcedureStateChanged",
    "ProcedureStateChangeEvent":{
        "TaskId":"1256768367-Procedure-2e1af2456351812be963e309cc133403t0",
        "Status":"FINISH",
        "FileId":"5285890784246869930",
        "FileName":"アニマルワールド",
        "FileUrl":"http://1256768367.vod2.myqcloud.com/xxx/xxx/AtUCmy6gmIYA.mp4",
        "MetaData":{
            "AudioDuration":60,
            "AudioStreamSet":[
```

```
                {
                    "Bitrate":383854,
                    "Codec":"aac",
                    "SamplingRate":48000
                }
            ],
            "Bitrate":1021028,
            "Container":"mov,mp4,m4a,3gp,3g2,mj2",
            "Duration":60,
            "Height":480,
            "Rotate":0,
            "Size":7700180,
            "VideoDuration":60,
            "VideoStreamSet":[
                {
                    "Bitrate":637174,
                    "Codec":"h264",
                    "Fps":23,
                    "Height":480,
                    "Width":640
                }
            ],
            "Width":640
        },
        "AiAnalysisResultSet":[
            {
                "Type":"Classification",
                "ClassificationTask":{
                    "Status":"SUCCESS",
                    "ErrCode":0,
                    "Message":"",
                    "Input":{
                        "Definition":10
                    },
                    "Output":{
                        "ClassificationSet":[
                            {
                                "Classification":"動物",
                                "Confidence":80
                            },
                            {
                                "Classification":"旅行",
                                "Confidence":34
                            }
                        ]
                    }
                }
```

```
            },
            {
                "Type":"Cover",
                "CoverTask":{
                    "Status":"SUCCESS",
                    "ErrCode":0,
                    "Message":"",
                    "Input":{
                        "Definition":10
                    },
                    "Output":{
                        "CoverSet":[
                            {
                                "CoverUrl":"http://1256768367.vod2.myqcloud.com/xxx
                                "Confidence":79
                            },
                            {
                                "CoverUrl":"http://1256768367.vod2.myqcloud.com/xxx
                                "Confidence":70
                            },
                            {
                                "CoverUrl":"http://1256768367.vod2.myqcloud.com/xxx
                                "Confidence":66
                            }
                        ]
                    }
                }
            },
            {
                "Type":"Tag",
                "TagTask":{
                    "Status":"SUCCESS",
                    "ErrCode":0,
                    "Message":"",
                    "Input":{
                        "Definition":10
                    },
                    "Output":{
                        "TagSet":[
                            {
                                "Tag":"馬",
                                "Confidence":34
                            },
                            {
                                "Tag":"鳥",
                                "Confidence":27
                            },
```

```
                                {
                                    "Tag":"植物",
                                    "Confidence":13
                                },
                                {
                                    "Tag":"ビーチ",
                                    "Confidence":11
                                }
                            ]
                        }
                    }
                }
            ],
            "TasksPriority":0,
            "TasksNotifyMode":""
        }
    }
```

コールバックの結果の中で、`ProcedureStateChangeEvent.AiAnalysisResultSet` に `Type` が `Classification` 、 `Cover` 、 `Tag` の3種類の分析結果がありますが、それぞれビデオインテリジェントクラシフィケーション、ビデオインテリジェントカバー画像、ビデオインテリジェントタグを表します。

`Type` が `Classification` の結果では、 `Output.ClassificationSet` の信頼度が最も高いカテゴリーが `動物` 、その次のカテゴリーが `旅行` と示されています。

`Type` が `Cover` の結果 `Output.CoverSet` には、3つの採用を推奨するカバー画像が示されています。 `CoverUrl` がカバー画像に対応するダウンロードアドレスです。

`Type` が `Tag` の結果 `Output.TagSet` には、4つの採用を推奨するビデオのタグが示され、信頼度の順に上から下へ配列されています。

ビデオコンテンツ認識

最終更新日：：2023-10-26 17:39:30

Tencent Cloud Video on Demand(VOD)サービスは2022年8月1日より、オーディオビデオコンテンツ認識の課金項目を新設し、ユーザーが開始したオーディオビデオコンテンツ認識タスクに対する課金を正式に開始します。詳細については、オーディオビデオコンテンツ認識の正式商用化に関するお知らせ をご参照ください。
オーディオビデオコンテンツ認識は、AIの力を借りてオーディオビデオコンテンツに対してインテリジェントな認識を行う機能であり、オフラインタスクです。オーディオビデオコンテンツ認識を使用することで、ビデオ画面の中の人の顔、文字、始点と終点、音声の中の文字を認識することができ、このオーディオビデオコンテンツ認識の結果に基づき、ビデオを的確かつ効果的に管理することができます。オーディオビデオコンテンツ認識には以下の機能が含まれます。

機能名	機能説明	活用例
フェイスレコグニション	画面内に登場する顔の認識	スターが画面に登場する位置のタグ付けを行います。 画面に登場した話題の人物について調べます。
音声全文認識	音声内に登場する全テキストの認識	スピーチ内容について字幕を生成します。 ビデオの音声内容に対するデータ分析を行います。
テキスト全文認識	画面内に登場する全テキストの認識	画面内のテキストに対するデータ分析を行います。
音声キーワード認識	音声内に存在するキーワードの認識	音声内のセンシティブワードを調べます。 音声内に出てきた特定のキーワードを検索します。
テキストキーワード認識	画面内に存在するキーワードの認識	画面内のセンシティブワードを調べます。 画面内に登場した特定のキーワードを検索します。
ビデオ先頭末尾認識	ビデオの先頭と末尾の認識	プログレスバーの中の先頭、末尾、本編の位置にタグ付けを行います。 ビデオの前後の不要な部分を一括削除します。

一部のコンテンツ認識機能は、素材コーパスに依存する必要があります。これにはパブリックコーパスとカスタマイズコーパスの2種類があります。
パブリックコーパス：VODのプリセットの素材コーパス。
カスタマイズコーパス：ユーザー自身で作成、管理する素材コーパス。

識別タイプ	パブリックコーパス	カスタマイズコーパス
フェイスレコグニション	サポートしています。素材の人物は主にエンターテイメントのスター、スポーツのスター、話題の人物です。	サポートしています。 サーバーAPIを呼び出して、人の顔のカスタマイズコーパスを管理します。

音声単語認識	この機能は現在サポートされていません。	サポートしています。サーバーAPIを呼び出してキーワードのコーパスを管理します。
文字単語認識	この機能は現在サポートされていません。	サポートしています。サーバーAPIを呼び出してキーワードのコーパスを管理します。

オーディオビデオコンテンツ認識テンプレート

オーディオビデオコンテンツ認識は複数の認識機能を統合しており、パラメータによって細かく制御する必要があります。制御のターゲットは以下のとおりです。
有効にする認識タイプ：コンテンツ認識の中のどの機能を有効にするか。
使用する素材コーパス：Face Recognitionに対して使用するのはパブリックコーパスかカスタマイズコーパスか。
フィルタリング点数の指定：Face Recognitionの信頼度が何点に達したら結果を返すか。
フィルタリングタグの指定：人の顔のタグがどの範囲にあれば結果を返すか。
一般的な操作の組み合わせを対象に、Video on Demandでは、プリセットオーディオビデオコンテンツ認識テンプレートを提供しています。その他、サーバーAPIを呼び出してカスタマイズしたオーディオビデオコンテンツ認識テンプレートを作成し、管理することができます。

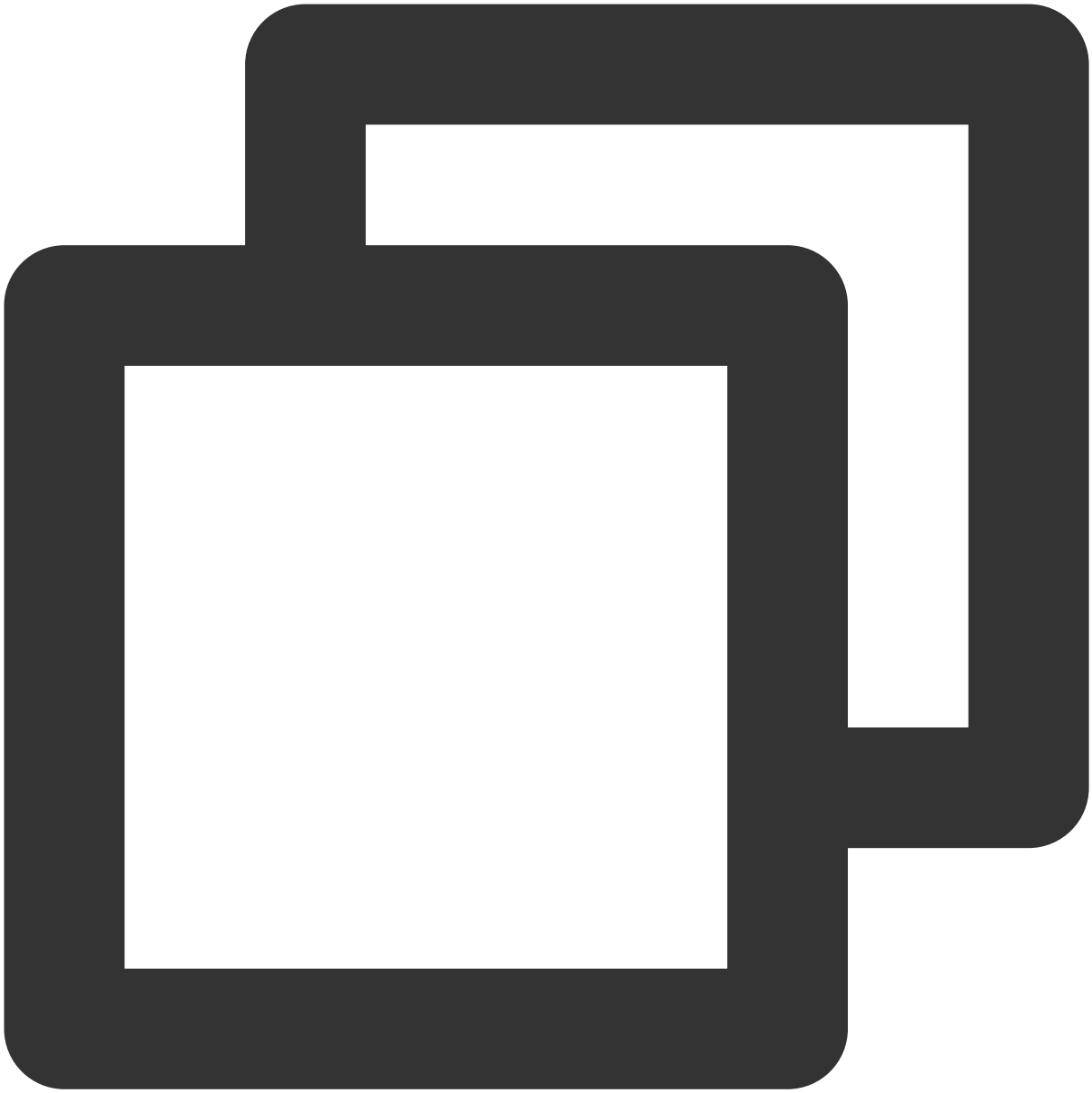
タスクの開始

オーディオビデオコンテンツ認識タスクの開始には、「サーバーAPIから直接開始」、「コンソールから直接開始」、「アップロード時に実行したいタスクを指定」の3種類の方法があります。詳細内容は、ビデオ処理のタスクの開始をご参照ください。
以下は、各方法のオーディオビデオコンテンツ認識タスク開始についての説明です。
サーバーAPIProcessMediaの呼び出しによるタスク開始：リクエストの中の `AiRecognitionTask` パラメータでオーディオビデオコンテンツ認識テンプレートのテンプレートIDを指定します。
コンソールでのビデオに対するタスク開始：サーバーAPIを呼び出してタスクフローを作成し、タスクフローの中でオーディオビデオコンテンツ認識タスクを設定します（ `MediaProcessTask.AiRecognitionTask` の中で指定）。コンソールでこのタスクフローを使用してビデオ処理を開始します。
サーバーからのアップロード時にタスクを指定： サーバーAPIを呼び出してタスクフローを作成し、タスクフローの中でオーディオビデオコンテンツ認識タスクを設定します（ `MediaProcessTask.AiRecognitionTask` の中で指定）。アップロードの申請の `procedure` パラメータでこのタスクフローを指定します。
クライアントからのアップロード時にタスクを指定： サーバーAPIを呼び出してタスクフローを作成し、タスクフローの中でオーディオビデオコンテンツ認識タスクを設定します

（ `MediaProcessTask.AiRecognitionTask` の中で指定）。クライアントからのアップロード署名の `procedure` でこのタスクフローを指定します。
コンソールからのアップロード：サーバーAPIを呼び出してタスクフローを作成し、タスクフローの中でオーディオビデオコンテンツ認識タスクを設定します（ `MediaProcessTask.AiRecognitionTask` の中で指定）。コンソールでビデオをアップロードし、アップロードと同時にビデオに対する処理操作を実行を選択して、ビデオアップロード後にこのタスクフローを実行するよう指定します。

結果の取得

オーディオビデオコンテンツ認識タスクを開始した後、非同期の結果通知を待機するか、またはタスク確認を同期的に実行する方法のどちらかでビデオコンテンツ認識タスクの実行結果を取得できます。以下は、ビデオコンテンツ認識タスクの開始後、通常のコールバック方式での結果通知の例となります（値がnullのフィールドは省略）。



```
{
    "EventType":"ProcedureStateChanged",
    "ProcedureStateChangeEvent":{
        "TaskId":"1400155958-Procedure-2e1af2456351812be963e309cc133403t0",
        "Status":"FINISH",
        "FileId":"5285890784363430543",
        "FileName":"名作選",
        "FileUrl":"http://1400155958.vod2.myqcloud.com/xxx/xxx/aHjWUx5Xo1EA.mp4",
        "MetaData":{
            "AudioDuration":243,
            "AudioStreamSet":[
```

```
                {
                    "Bitrate":125599,
                    "Codec":"aac",
                    "SamplingRate":48000
                }
            ],
            "Bitrate":1459299,
            "Container":"mov,mp4,m4a,3gp,3g2,mj2",
            "Duration":243,
            "Height":1080,
            "Rotate":0,
            "Size":44583593,
            "VideoDuration":243,
            "VideoStreamSet":[
                {
                    "Bitrate":1333700,
                    "Codec":"h264",
                    "Fps":29,
                    "Height":1080,
                    "Width":1920
                }
            ],
            "Width":1920
        },
        "AiRecognitionResultSet":[
            {
                "Type":"FaceRecognition",
                "FaceRecognitionTask":{
                    "Status":"SUCCESS",
                    "ErrCode":0,
                    "Message":"",
                    "Input":{
                        "Definition":10
                    },
                    "Output":{
                        "ResultSet":[
                            {
                                "Id":183213,
                                "Type":"Default",
                                "Name":"張三",
                                "SegmentSet":[
                                    {
                                        "StartTimeOffset":10,
                                        "EndTimeOffset":12,
                                        "Confidence":97,
                                        "AreaCoordSet":[
                                            830,
```



```
                                    783,
                                    1030,
                                    599
                                ]
                            },
                            {
                                "StartTimeOffset":12,
                                "EndTimeOffset":14,
                                "Confidence":97,
                                "AreaCoordSet":[
                                    844,
                                    791,
                                    1040,
                                    614
                                ]
                            }
                        ]
                    },
                    {
                        "Id":236099,
                        "Type":"Default",
                        "Name":"lisi",
                        "SegmentSet":[
                            {
                                "StartTimeOffset":120,
                                "EndTimeOffset":122,
                                "Confidence":96,
                                "AreaCoordSet":[
                                    579,
                                    903,
                                    812,
                                    730
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ],
    "TasksPriority":0,
    "TasksNotifyMode":""
  }
}
```

コールバックの結果の中で、`ProcedureStateChangeEvent.AiRecognitionResultSet` に `Type` が `FaceRecognition` となる認識結果があり、顔認識を表します。

`Type` が `FaceRecognition` の結果では、`Output.ResultSet` の中に認識した人物が2人含まれており、それぞれ `張三` とlisiとなっています。`SegmentSet` には人の顔がビデオに登場した時間帯（`StartTimeOffset` と `EndTimeOffset` により確定）および画面の中の座標（`AreaCoordSet` により確定）が示されています。

画像審査

最終更新日：：2023-10-26 17:39:31

VOD画像審査は、AIの力を借りて違反情報を審査します。審査の結果には、審査の評点、審査のアドバイスが含まれます。「審査のアドバイス」に基づき、メディア管理者は画像の公開を許可するかを決定でき、不正な画像による法的リスクやブランドのイメージダウンを効果的に回避することができます。
VODでは画面上の画像、画面内のテキストの2種類を対象に審査を行うことができます。審査タグには、ポルノ、暴力、不適切な情報、違法、罵詈雑言、広告などが含まれます。

オブジェクト	審査タグ
画面上の画像	ポルノ（Porn）
	暴力（Terror）
	不適切な情報（Polity）
	広告（Ad）
	違法（Illegal）
画面内のテキスト（OCR）	ポルノ（Porn）
	暴力（Terror）
	不適切な情報（Polity）
	広告（Ad）
	違法（Illegal）
	罵詈雑言（Abuse）

審査結果は下表に示すとおりです。

フィールド名	タイプ	意味
Confidence	Float	審査の評点（0～100）。評点が高いほど、疑わしさが大きくなります。
Suggestion	String	審査のアドバイスには、pass、review、blockの3種類があります。 pass：疑わしさが低いため、そのまま合格させることをお勧めします。 review：疑わしさが高いため、手動で再審査を行うことをお勧めします。

		block：非常に疑わしいため、そのままブロックすることをお勧めします。

処理の開始

2種類の方法で画像審査を開始することができます。VODコンソールから操作またはサーバーAPI画像審査から呼び出しをすることができます。

結果の取得

どの方法で処理を開始しても、審査結果はすぐに返されます（同期され返されます）。
VODコンソールによって開始した場合、出力結果は、VODコンソールをご参照ください。サーバーAPI画像審査によって呼び出して開始した場合、出力結果のデータ形式は、画像審査 - 3. 出力パラメータをご参照ください。

イベント通知
イベント通知の概要

最終更新日：：2023-10-26 17:39:31

VODのビデオに対して開始されるアップロード、削除、ビデオ処理などの操作は、すべてイベントと呼ぶことができます。イベントの実行は、完了するまでに時間がかかります。VODはイベントが終了すると、直ちにAppサービスに実行結果、つまりイベント通知を送信します。

VODは、次のタイプのイベント通知をサポートします。

カテゴリー	イベント通知
アップロード削除	ビデオアップロードの完了
	URLからのビデオプルアップロードの完了
	ビデオ削除の完了
ビデオ処理	タスクフロー状態の変更
	ビデオ編集の完了
	ビデオ合成の完了

イベント通知方法には、「通常のコールバック」と「信頼できるコールバック」があります。 VODコンソール にログインしてコールバックモードを設定し、コールバックを受信するイベントを設定します。具体的な操作については、 コールバック設定 をご参照ください。

- 通常のコールバック：コンソールでコールバックURLを設定すると、システムはイベントの完了後にそのURLに対し、HTTPリクエストを送信します。リクエストボディには通知内容が含まれています。
- 信頼できるコールバック：イベントが完了した後、VODシステムは通知を組み込みのキューに入れ、Appサービスはサーバー APIを介してキュー内の通知を消費します。

通常のコールバック

通常のコールバックは、Appサービスがイベント通知を受動的に受信するモードです。コールバックURLを設定し、通常のコールバックモードを選択すると、VODはイベントの完了後にコールバックURLへのコールバックを開始します。

VODが開始する通常のコールバックの形式はHTTPリクエストであり、リクエストボディはJSON形式です。コンテンツに EventHandle パラメータのEventContentの構造 は含まれません。
タスクステータス変更通知を例に取った場合、コールバックの `EventType` パラメータは `ProcedureStateChanged` であり、情報は `ProcedureStateChangeEvent` パラメータ（ProcedureTask構造）によって表されます。

信頼できるコールバック

信頼できるコールバックは、Appサービスがイベント通知をVODに対してアクティブにプルするモードです。信頼できるコールバックモードを選択した場合、VODシステムはイベント通知をキューに入れ、Appサービスはサーバーを介してキュー内のイベント通知を順番に消費します。
Appサービスはプルイベント通知 APIを介してメッセージを取得した場合、イベント通知の確認 APIを呼び出して確認する必要があります。メッセージは必ず確認された上でVODのキューから削除されるので、「信頼できるコールバック」は「通常のコールバック」よりもの信頼性が高くなります。**イベント通知の信頼性の高い要件では、「信頼できるコールバック」モードを使用することをお勧めします。**

イベント通知の入門チュートリアル

最終更新日：：2023-10-26 17:39:31

このチュートリアルでは、「通常のコールバック」と「信頼できるコールバック」という2つのメソッドをはじめとするVODのイベント通知の使用方法について、詳しくご説明します。

前提条件

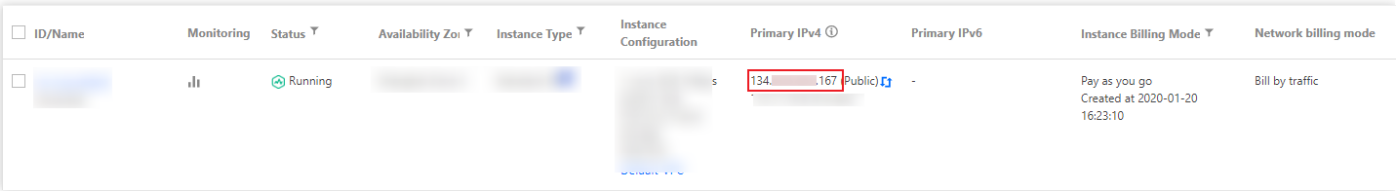
- Tencent Cloudアカウントの登録および実名認証を完了していること。実名認証を行っていないユーザーは、中国国内のVODインスタンスを購入できません。
- 通常のコールバックでは、Python 2 .7ランタイム環境が必要です。

通常のコールバック

コールバック受信サービスのデプロイ

通常のコールバック を使用してイベント通知を取得するには、パブリックIPのあるサーバーにコールバック受信サービスをデプロイする必要があります。ここでは例としてCVMインスタンスにこのサービスをデプロイする方法をご説明します。

1. CVMコンソールのインスタンスリスト ページ に入り、【新規作成】をクリックします。
2. 【クイックコンフィグレーション】メニューを選択し、【イメージ】で“**Ubuntu Server**”または“**CentOS**”を選択して、【パブリックネットワーク帯域幅】で“**1Mbps**”を選択し、“**無料のパブリックIPの割り当て **”にチェックを入れて、【今すぐ購入】をクリックします。
3. インスタンスリスト ページに再度入り、作成に成功したCVMインスタンス を見つけ、【プライマリIPアドレス】のパブリックIPをコピーします（この例では**134.XXX.XXX.167**）。



4. 購入したCVMにログインし、ソースコード圧縮パッケージ をダウンロードして作業ディレクトリに解凍し、次のコマンドを実行します。

```
python NotificationReceiveServer.py
```

コマンドが実行された後、CVMの標準出力で `Started httpserver on port 8080` が出力されます。サービスプロセスが開始されてポート8080を監視していることを意味します。

5. ブラウザに `http://134.XXX.XXX.167:8080` と入力すると、CVMの標準出力で次のHTTPリクエスト情報が出力されます。

```
[root@yanchuxiong notification]# python NotificationReceiveServer.py
Started httpserver on port  8080

----- Request Start ----->

/
Host: [redacted]:8080
Proxy-Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.131 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
connection: keep-alive

<----- Request End -----

103.7.29.7 - - [28/May/2019 07:59:56] "GET / HTTP/1.1" 200 -
```

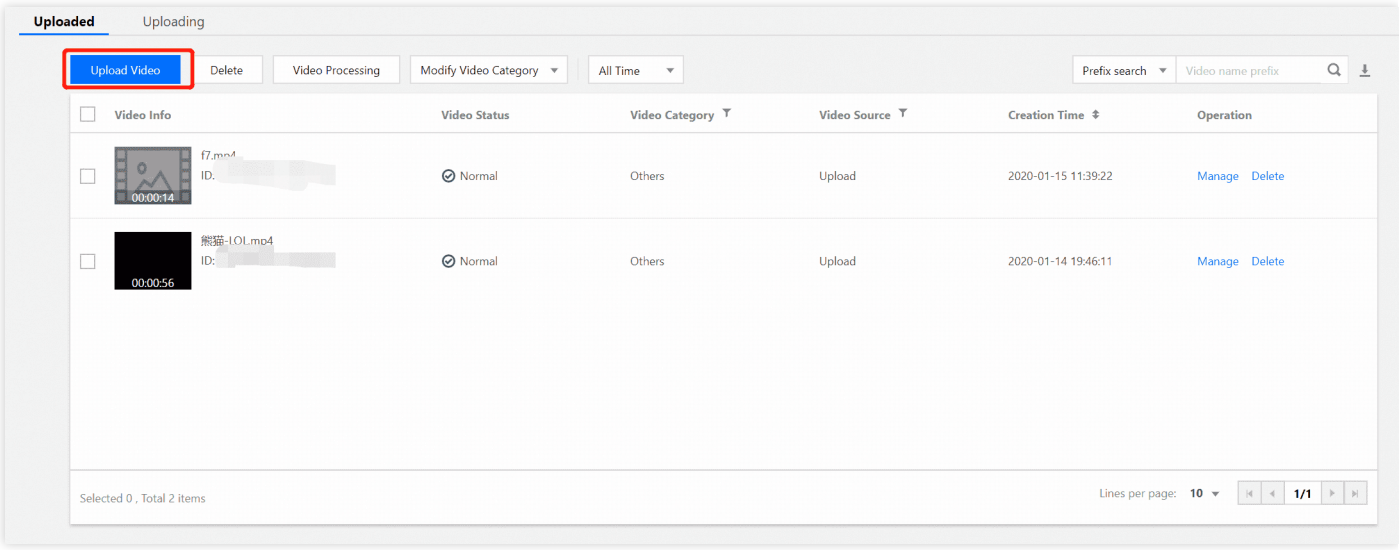
通常のコールバックのコンフィグレーション

1. VODコンソールにログインし、左側メニューバーの【コールバック設定】をクリックします。
2. 【設定】をクリックします。
   - コールバックモード：「**通常のコールバック**」を選択します。
   - コールバックURL：`http://134.XXX.XXX.167:8080` を入力します。
   - コールバックイベント：「**ビデオアップロード完了コールバック**」にチェックを入れます。
3. 【OK】をクリックして設定を完了してください。

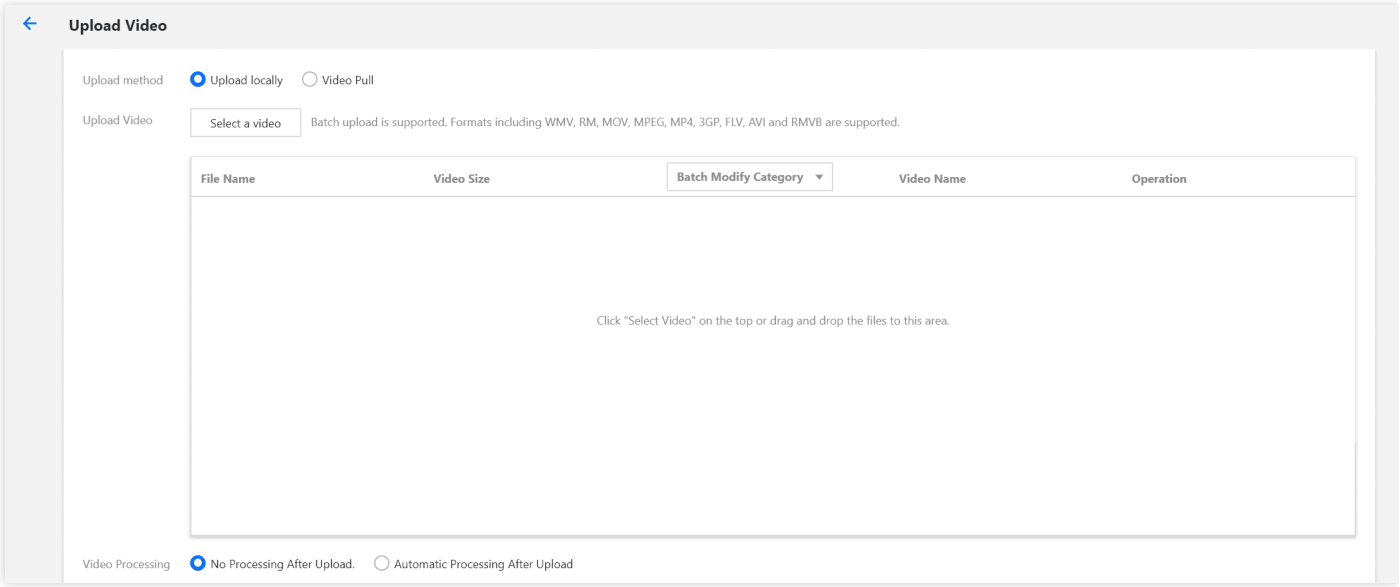
通常のコールバックの開始と受信

デモビデオ をローカルにダウンロードして、手引きとしてご利用ください。

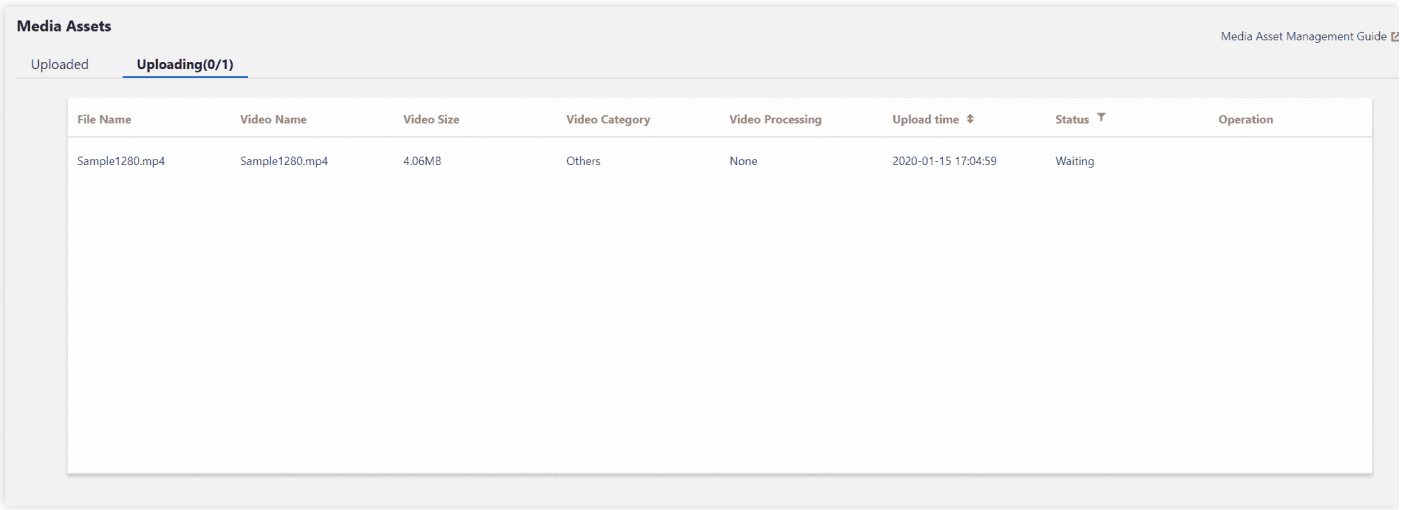
1. 左側メニューバーの【メディア資産管理】をクリックし、【アップロード済み】を選択して、【ビデオのアップロード】をクリックします。



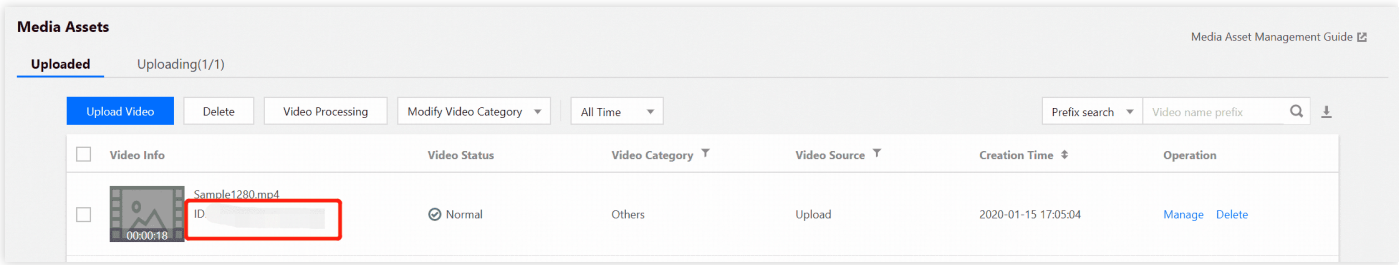
2. 「**ビデオのアップロード**」ダイアログボックスが表示されたら、【ローカルからのアップロード】を選択し、【ビデオの選択】をクリックして**デモビデオ**をVODプラットフォームにアップロードします。



アップロードを実行すると、“**アップロード中**”バーにビデオアップロードの進行状況が表示されます。



アップロードが完了すると、“**アップロード済み**”バーのビデオリストに、アップロードされたビデオとビデオに対応するID（FileId）が表示されます。



3. CVMをチェックして、標準出力でビデオアップロード完了 通知の内容を出力します。

```
-----> Request Start <-----

/
Host: [redacted]
User-Agent: Go-http-client/1.1
Content-Length: 898
Accept-Encoding: gzip
Content-Type: application/json
Vod-Forwardproxy-Begin-Time: 1559045788
Vod-Forwardproxy-Out-To-Host: [redacted]:8080
Vod-Forwardproxy-Routetype: host/[redacted]/8080

{"EventType":"NewFileUpload","FileUploadEvent":{"FileId":"5[redacted]0","MediaBasicInfo":{"Name":"Wildlife.wm
v","Description":"","CreateTime":"2019-05-28T12:16:26Z","UpdateTime":"2019-05-28T12:16:28Z","ExpireTime":"9999-12-31
T23:59:59Z","ClassId":0,"ClassName":"其他","ClassPath":"-1","CoverUrl":"","Type":"wmv","MediaUrl":"http://1255566954
.vod2.myqcloud.com/ca75586fvodgzp1255566954/fb3a6191[redacted]0/KfxUIIhsre0A.wmv","TagSet":[],"StorageRegion
":"ap-guangzhou-2","SourceInfo":{"SourceType":"Upload","SourceContext":""},"Vid":"5[redacted]0"},"ProcedureTa
skId":""},"ProcedureStateChangeEvent":null,"FileDeleteEvent":null,"PullCompleteEvent":null,"EditMediaCompleteEvent":
null,"WechatPublishCompleteEvent":null,"TranscodeCompleteEvent":null,"ConcatCompleteEvent":null,"ClipCompleteEvent":
null,"CreateImageSpriteCompleteEvent":null,"SnapshotByTimeOffsetCompleteEvent":null}
<----- Request End ----->

123.207.100.120 - - [28/May/2019 08:16:30] "POST / HTTP/1.1" 200 -
```

4. 【メディア資産管理】の「**アップロード済み**」バーで、先ほどアップロードしたビデオにチェックを入れ、【ビデオ処理】をクリックします。【処理タイプ】で「**トランスコードテンプレートの手動選択**」オプションを選択し、【トランスコードテンプレート】の「**MP4-LD-FLU (10)**」にチェックを入れ、【ビデオカバー】にチェックを入れたまま、【OK】をクリックします。
5. 10分間待った後、CVMをチェックし、標準出力でタスクフローステータスの変更 通知の内容を出力します。ここには、トランスコードの結果（`Type` が `Transcode` の場合）やカバー生成時点のスクリーンキャプチャの結果（`Type` が `CoverBySnapshot` の場合）が含まれます。

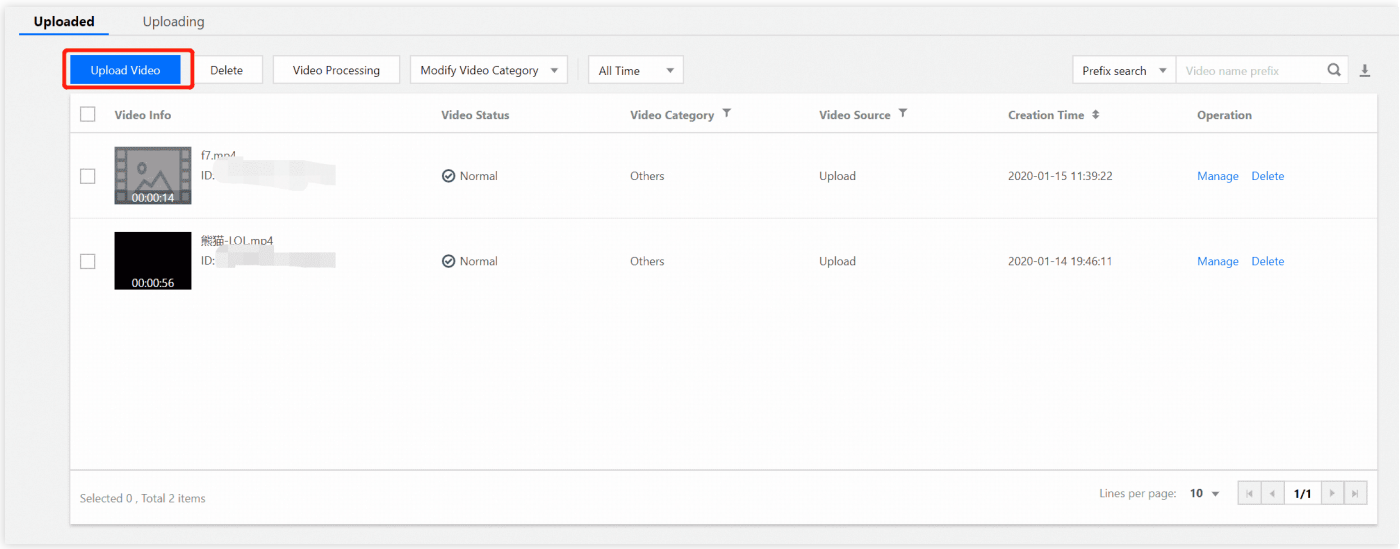
この時点で、ビデオをアップロードしてトランスコードタスク実行が完了しています。アップロードとトランスコードが完了すると、コールバック受信サービスは「**ビデオのアップロードの完了**」と「タスクフロー状態の変更」通知を受信します。

信頼できるコールバック

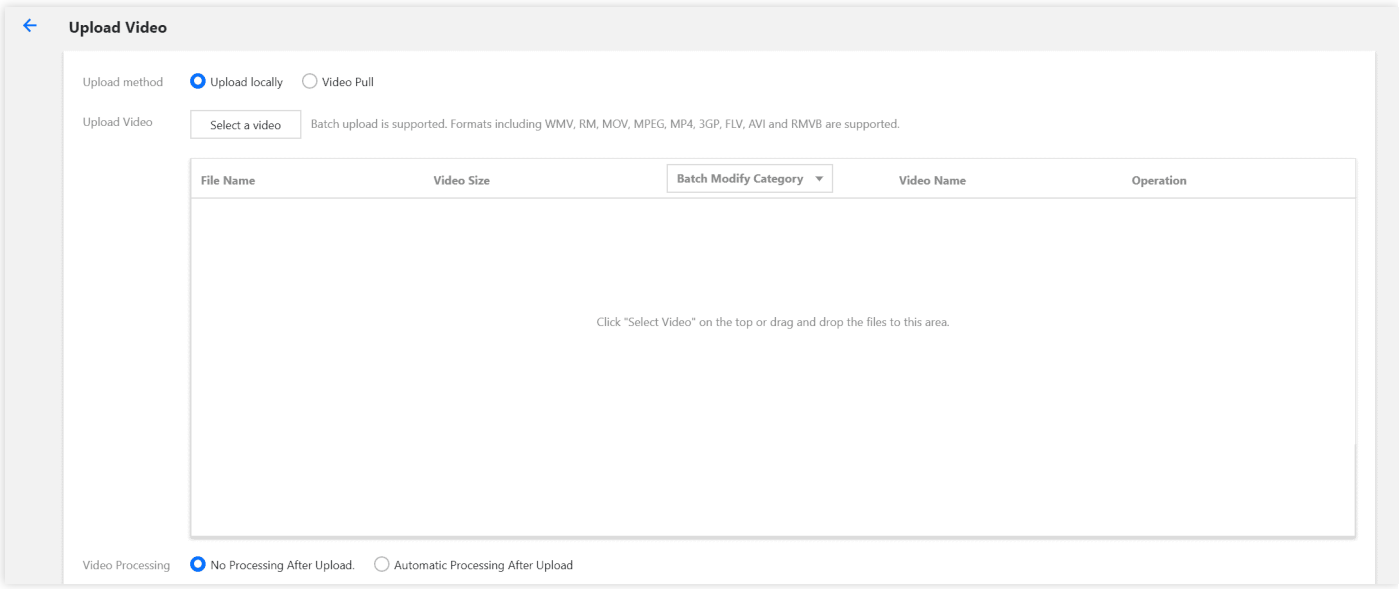
1. VODコンソールにログインし、左側メニューバーの【コールバック設定】をクリックします。
2. 【設定】をクリックします。
   - コールバックモード：「**信頼できるコールバック**」を選択します。
   - コールバックイベント：「**ビデオアップロード完了コールバック**」にチェックを入れます。
3. 【OK】をクリックして設定を完了してください。

信頼できるコールバックの開始

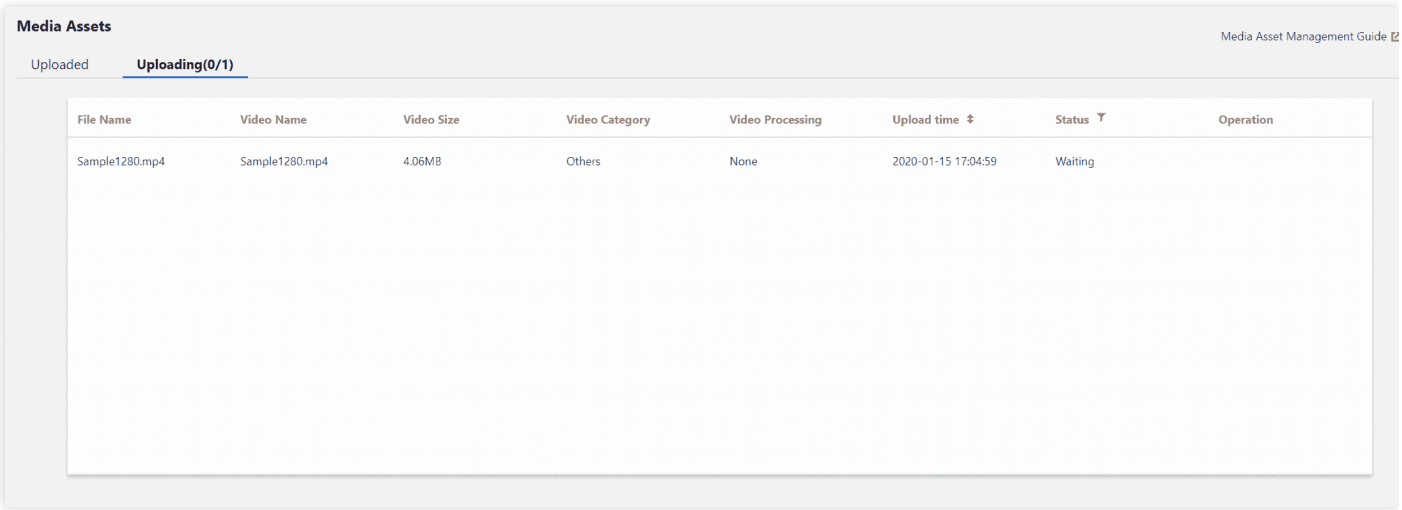
1. 左側メニューバーの【メディア資産管理】をクリックし、【アップロード済み】を選択して、【ビデオのアップロード】をクリックします。



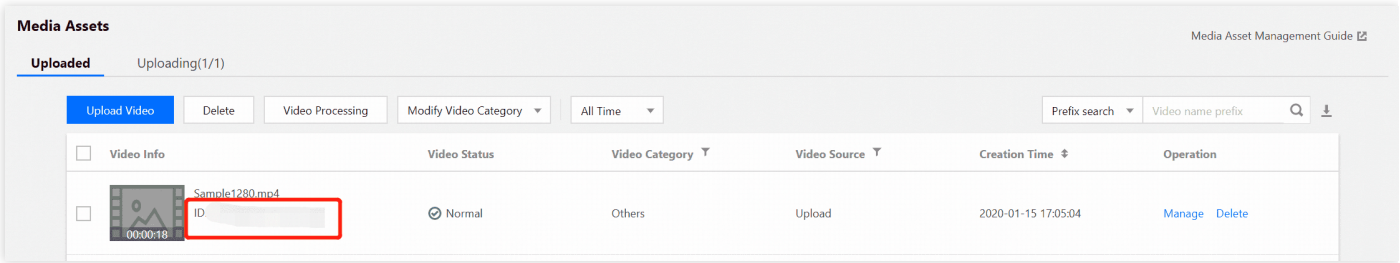
2. 「**ビデオのアップロード**」ダイアログボックスが表示されたら、【ローカルからのアップロード】を選択し、【ビデオの選択】をクリックして**デモビデオ**をVODプラットフォームにアップロードします。



アップロードを実行すると、“**アップロード中**”バーにビデオアップロードの進行状況が表示されます。



アップロードが完了すると、“**アップロード済み**”バーのビデオリストに、アップロードされたビデオとビデオに対応するID（FileId）が表示されます。



3. 【メディア資産管理】の「**アップロード済み**」バーで、先ほどアップロードしたビデオにチェックを入れ、【ビデオ処理】 をクリックします。【処理タイプ】で「**トランスコードテンプレートの手動選択**」オプションを選択し、【トランスコードテンプレート】の「**MP4-LD-FLU (10)**」にチェックを入れ、【ビデオカバー】にチェックを入れたまま、【OK】をクリックします。

この時点で、ビデオは再度アップロードされ、ビデオのトランスコードタスクが開始されました。これらの操作によりイベント通知がトリガーされます。

信頼できるコールバックのプル

イベント通知がトリガーされたら、信頼できるコールバックを自発的にプルする必要があります。APIツールを介して `PullEvents` コマンドを実行し、消費されていないイベント通知をクエリーすることができます。具体的な手順は次のとおりです。

1. VOD PullEventsコマンドのAPI 3.0 Explorerに入り、お客様の「**パーソナルキー**」を入力します（【キーの確認】をクリックし、APIキー管理コンソールから取得できます）。
2. 右側のバーの中から【オンラインコール】を選択し、【リクエスト送信】をクリックします。

リクエストを送信した後、もどってきた結果の中のイベントリスト `EventSet` を見ます。イベントリストの中には2つのイベント通知が含まれ、タイプは「**ビデオアップロード完了**」と「**タスクフロー状態の変更**」で、対応するイベントハンドラはそれぞれ `8078360634045903972` と `8078360634045903972` です。

```
{
  "Response": {
    "EventSet": [
      {
        "EventHandle": "8078571180371850078",
        "EventType": "NewFileUpload",
        "FileUploadEvent": {
          "FileId": "528_____________207",
          "MediaBasicInfo": {
            "Name": "Wildlife.wmv",
            "Description": "",
```

```
      {
        "EventHandle": "8078360634045903972",
        "EventType": "ProcedureStateChanged",
        "FileUploadEvent": null,
        "ProcedureStateChangeEvent": {
          "TaskId": "1255566954-Procedure-fa015168bf78fb8de4c9e504c4fe079ct
0",
          "Status": "FINISH",
          "ErrCode": 0,
          "Message": "",
```


この時点で、信頼できるコールバックの方式で、「**ビデオアップロード完了**」と「**タスクフロー状態の変更**」の2つの通知をプルしました。続いて、プルしたイベント通知をすみやかに確認する必要があります。

信頼できるコールバックの確認

プルした信頼できるコールバックは、**30秒以内**に確認を終える必要があります。そうでない場合、

- イベントハンドラが30秒後に再びAPIを呼び出して確認を行い、呼び出しに失敗します。
- イベントは30秒を超えても確認がないと、次の回の信頼できるコールバックのプルの時に再びプルされます。

APIツールを使用して `ConfirmEvents` コマンドを実行し、消費されていないイベント通知をクエリーすることができます。具体的な手順は次のとおりです。

1. VOD ConfirmEvents コマンドのAPI 3.0 Explorerに入り、お客様の「**パーソナルキー**」を入力します。

2. 「**EventHandles.N**」の入力ボックスに、イベント通知をプルした時に取得したイベントハンドラを入力します。

3. 右側のバーの中から【オンラインコール】を選択し、【リクエスト送信】をクリックします。

リクエストに成功すると、次の情報が返ってきます。

レスポンスに失敗した場合は、次をご確認ください。

- イベントハンドラの入力ミス。
- イベントがプルされ30秒以上経過していないか。

4. 5分経ってから、再びAPIツールを使用して `PullEvents` コマンドを実行し、イベント通知をプルします。

ResourceNotFoundのエラーが戻ってきた場合は、プルできる通知がすでにないことを意味します。

ビデオアップロードの完了

最終更新日：：2023-10-26 17:39:30

イベント名

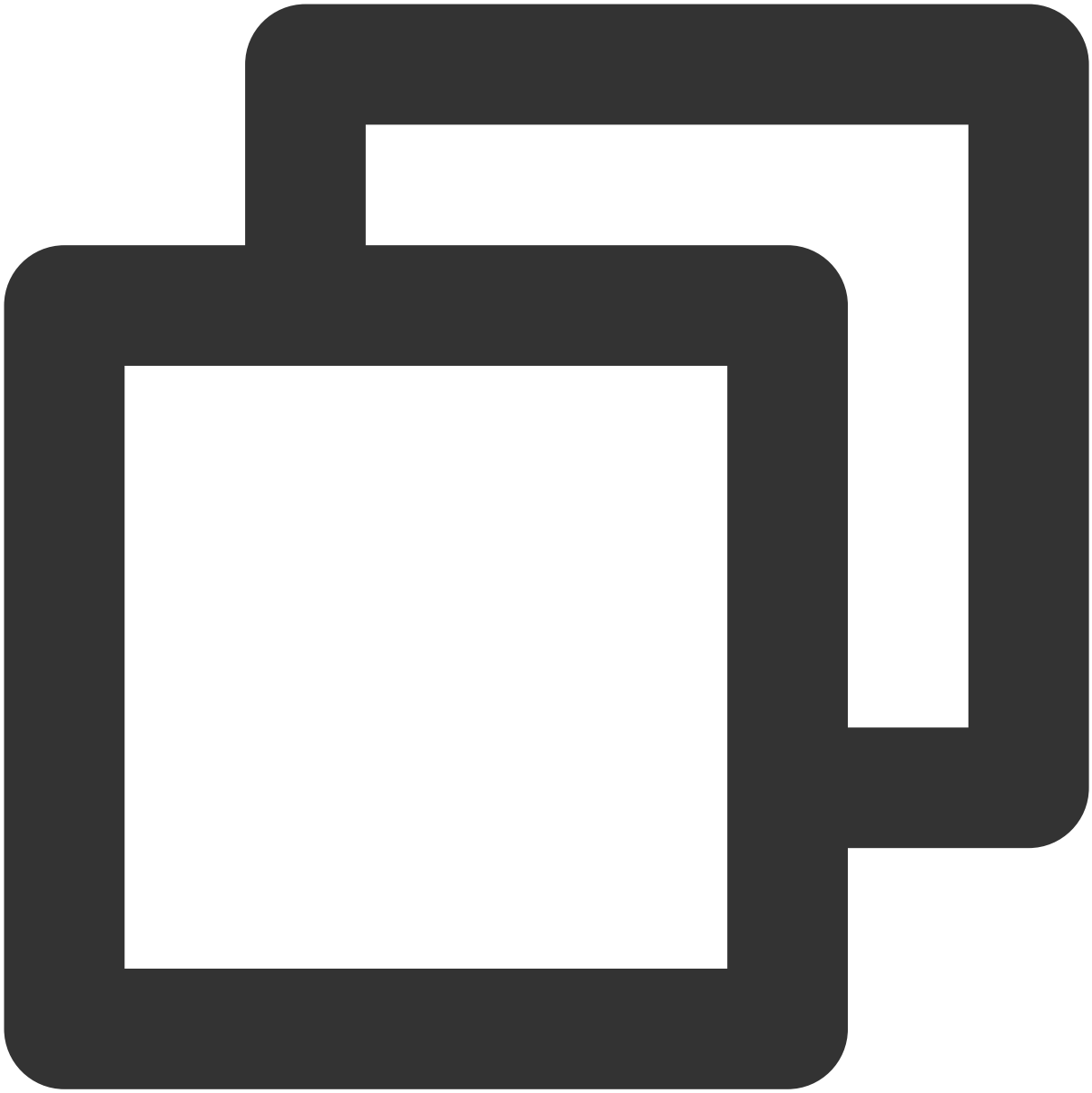
NewFileUpload

イベントの説明

Appがイベント通知を構成し、かつビデオをクライアントまたはサーバーからアップロードした場合、Appバックエンドは「通常のコールバック」または「信頼できるコールバック」によりこのイベント通知を取得することができます。イベント通知の内容は、FileUploadTaskの構造です。

通常のコールバック

通常のコールバックモードを選択すると、コールバックURLはVODからのHTTPリクエストを受信します。リクエストはPOSTメソッドを使用します。リクエストの内容は、以下に示すようにBODYにあります（null値のフィールドは省略されています）。

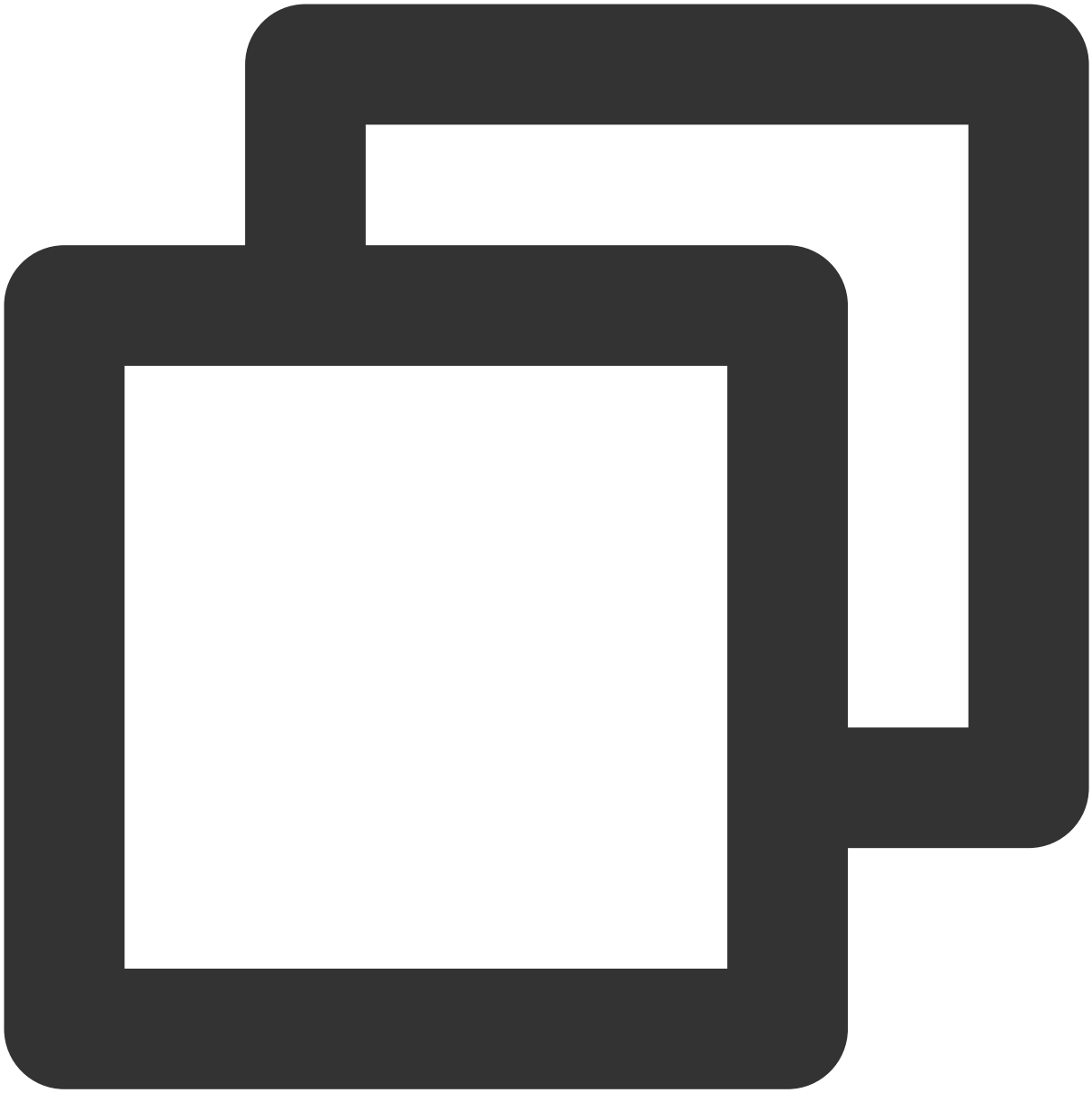


```
{
    "EventType":"NewFileUpload",
    "FileUploadEvent":{
        "FileId":"5285890784273533167",
        "MediaBasicInfo":{
            "Name":"アニマルワールド",
            "Description":"",
            "CreateTime":"2019-01-09T16:36:22Z",
            "UpdateTime":"2019-01-09T16:36:24Z",
            "ExpireTime":"9999-12-31T23:59:59Z",
            "ClassId":0,
```

```
            "ClassName":"その他",
            "ClassPath":"その他",
            "CoverUrl":"",
            "Type":"mp4",
            "MediaUrl":"http://125676836723.vod2.myqcloud.com/xxx/xxx/q1BORBPQH1IA.
            "TagSet":[
            ],
            "StorageRegion":"ap-guangzhou-2",
            "SourceInfo":{
                "SourceType":"Upload",
                "SourceContext":""
            },
            "Vid":"5285890784273533167"
        },
        "ProcedureTaskId":"",
        "ReviewAudioVideoTaskId":""
    }
}
```


信頼できるコールバック

信頼できるコールバックモードを選択し、プルイベント通知APIが呼び出された後に、次の形式のHTTPレスポンスを受信します（null値のフィールドは省略されています）。




```
{
    "Response": {
        "EventSet": [
    {
                "EventHandle": "EventHandle.N",
                "EventType": "NewFileUpload",
                "FileUploadEvent": {
                    "FileId": "5285890784273533167",
                    "MediaBasicInfo": {
                        "Name": "アニマルワールド",
                        "Description": "",
```

```
                        "CreateTime": "2019-01-09T16:36:22Z",
                        "UpdateTime": "2019-01-09T16:36:24Z",
                        "ExpireTime": "9999-12-31T23:59:59Z",
                        "ClassId": 0,
                        "ClassName": "その他",
                        "ClassPath": "その他",
                        "CoverUrl": "",
                        "Type": "mp4",
                        "MediaUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/
                        "TagSet": [],
                        "StorageRegion": "ap-guangzhou-2",
                        "SourceInfo": {
                            "SourceType": "Upload",
                            "SourceContext": ""
                        },
                        "Vid": "5285890784273533167"
                    },
                    "ProcedureTaskId": "",
                    "ReviewAudioVideoTaskId":""
                }
    }
        ],
        "RequestId": "335bdaa3-db0e-46ce-9946-51941d9cb0f5"
    }
}
```

URLからのビデオプルアップロードの完了

最終更新日：：2023-10-26 17:39:31

イベント名

PullComplete

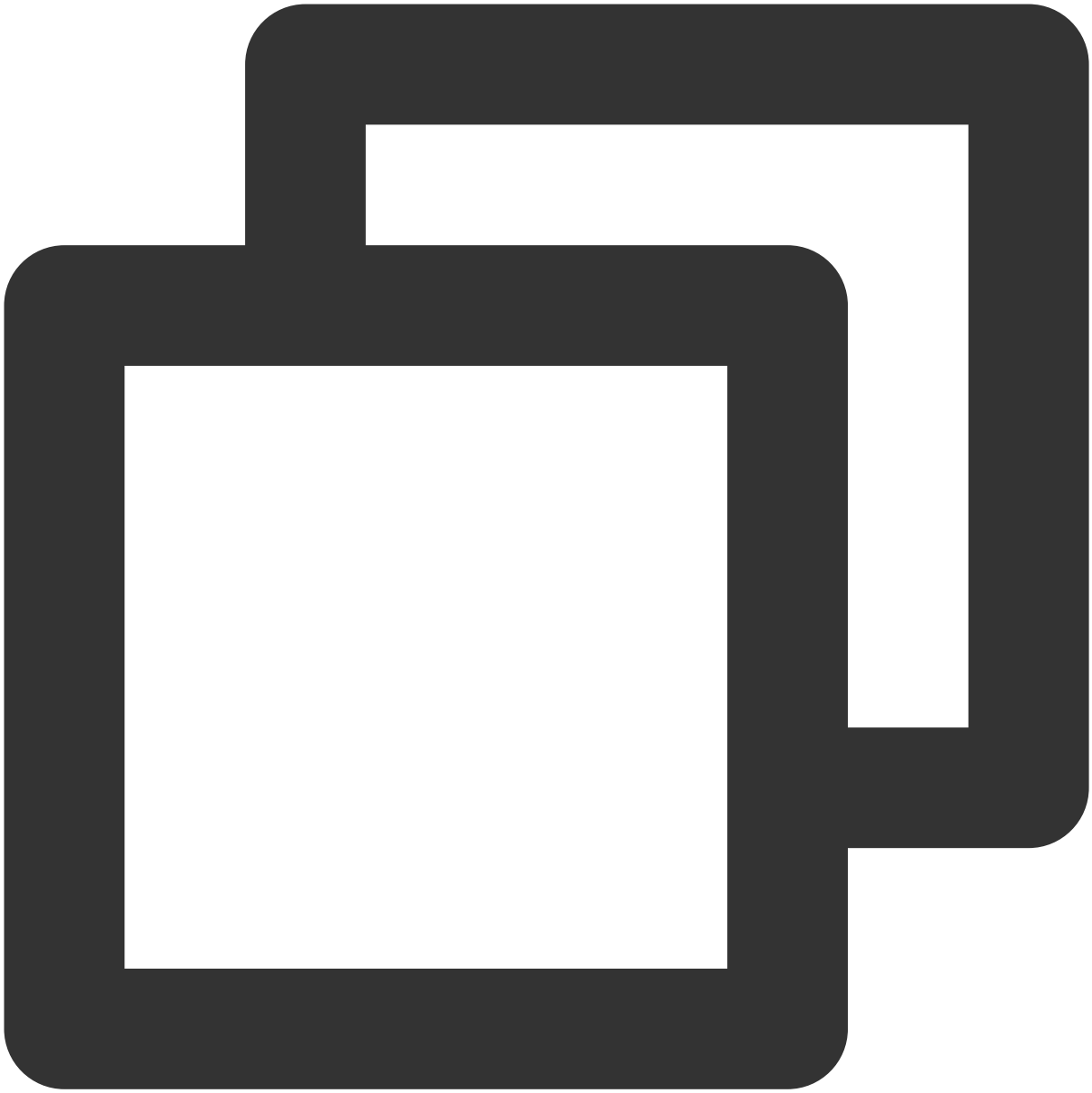
イベントの説明

Appがイベント通知を構成し、かつビデオのプルアップロードが完了している場合、Appバックエンドは「通常のコールバック」または「信頼できるコールバック」によりこのイベント通知を取得することができます。イベント通知の内容は、PullCompleteの構造です。

事例

通常のコールバック

通常のコールバックモードを選択すると、コールバックURLはVODからのHTTPリクエストを受信します。リクエストはPOSTメソッドを使用します。リクエストの内容は、以下に示すようにBODYにあります（null値のフィールドは省略されています）。

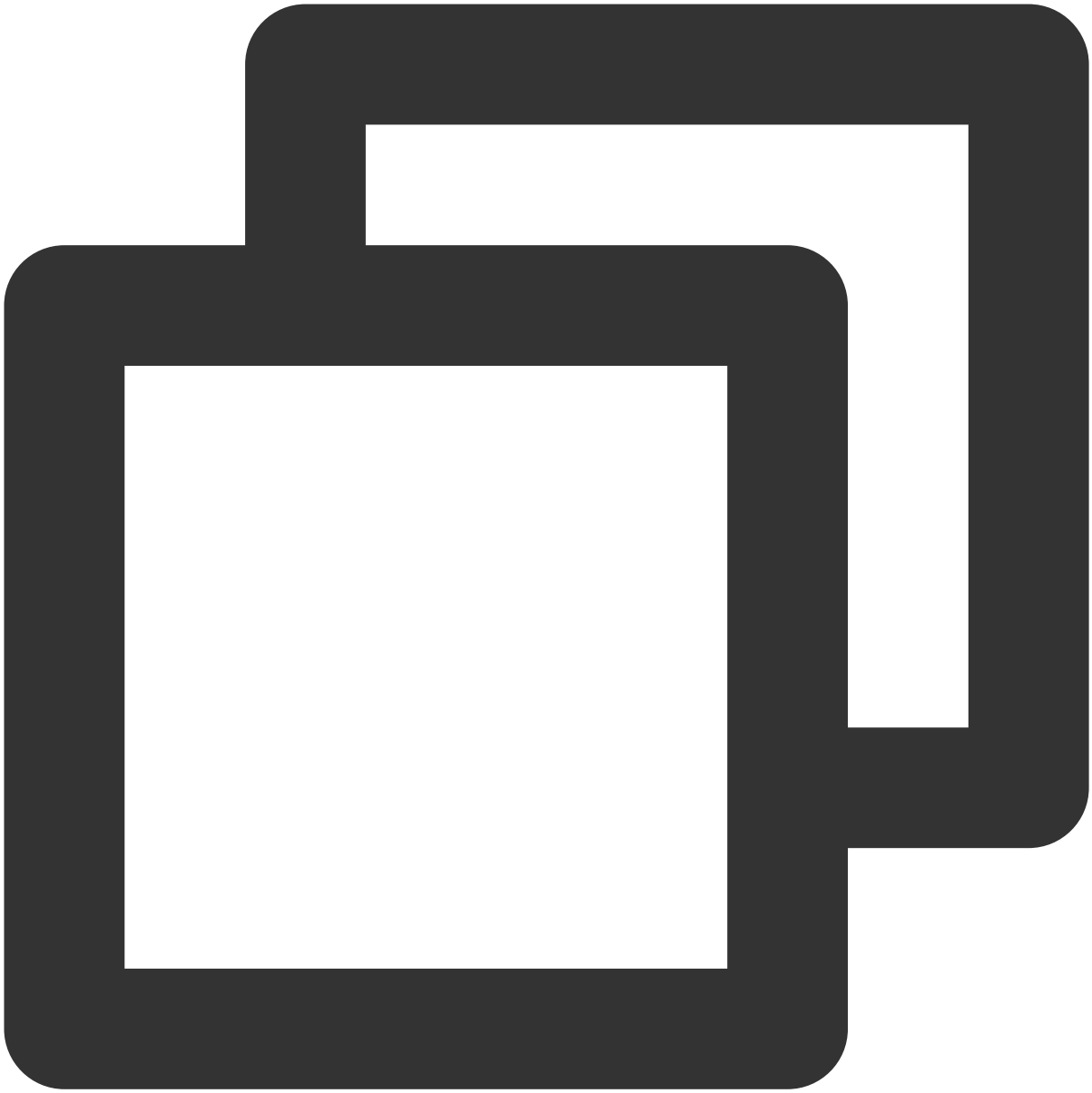



```
{
    "EventType": "PullComplete",
    "PullCompleteEvent": {
        "TaskId": "125676836723-Pull-f5ac8127b3b6b85cdc13f237c6005d8",
        "Status": "FINISH",
        "ErrCode": 0,
        "Message": "SUCCESS",
        "FileId": "14508071098244959037",
        "MediaBasicInfo": {
            "Name": "アニマルワールド",
            "Description": "",
```

```
            "CreateTime": "2019-01-09T16:36:22Z",
            "UpdateTime": "2019-01-09T16:36:24Z",
            "ExpireTime": "9999-12-31T23:59:59Z",
            "ClassId": 0,
            "ClassName": "その他",
            "ClassPath": "その他",
            "CoverUrl": "",
            "Type": "mp4",
            "MediaUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/xxx.mp4",
            "TagSet": [ ],
            "StorageRegion": "ap-guangzhou-2",
            "SourceInfo": {
                "SourceType": "Upload",
                "SourceContext": ""
            },
            "Vid": ""
        },
        "FileUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/xxx.mp4",
        "ProcedureTaskId": "",
        "ReviewAudioVideoTaskId":"",
        "SessionContext": "",
        "SessionId": ""
    }
}
```


信頼できるコールバック

信頼できるコールバックモードを選択し、プルイベント通知APIが呼び出された後に、次の形式のHTTPレスポンスを受信します（null値のフィールドは省略されています）。




```
{
    "Response": {
        "EventSet": [
    {
                "EventHandle": "EventHandleX",
                "EventType": "PullComplete",
                "PullCompleteEvent": {
                    "TaskId": "125676836723-Pull-f5ac8127b3b6b85cdc13f237c6005d8",
                    "Status": "FINISH",
                    "ErrCode": 0,
                    "Message": "SUCCESS",
```



```
                    "FileId": "14508071098244959037",
                    "MediaBasicInfo": {
                        "Name": "アニマルワールド",
                        "Description": "",
                        "CreateTime": "2019-01-09T16:36:22Z",
                        "UpdateTime": "2019-01-09T16:36:24Z",
                        "ExpireTime": "9999-12-31T23:59:59Z",
                        "ClassId": 0,
                        "ClassName": "その他",
                        "ClassPath": "その他",
                        "CoverUrl": "",
                        "Type": "mp4",
                        "MediaUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/
                        "TagSet": [ ],
                        "StorageRegion": "ap-guangzhou-2",
                        "SourceInfo": {
                            "SourceType": "Upload",
                            "SourceContext": ""
                        },
                        "Vid": ""
                    },
                    "FileUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/xxx.m
                    "ProcedureTaskId": "",
                    "ReviewAudioVideoTaskId":"",
                    "SessionContext": "",
                    "SessionId": ""
                }
    }
        ]
    }
}
```

ビデオ削除の完了

最終更新日：：2023-10-26 17:39:30

イベント名

FileDeleted

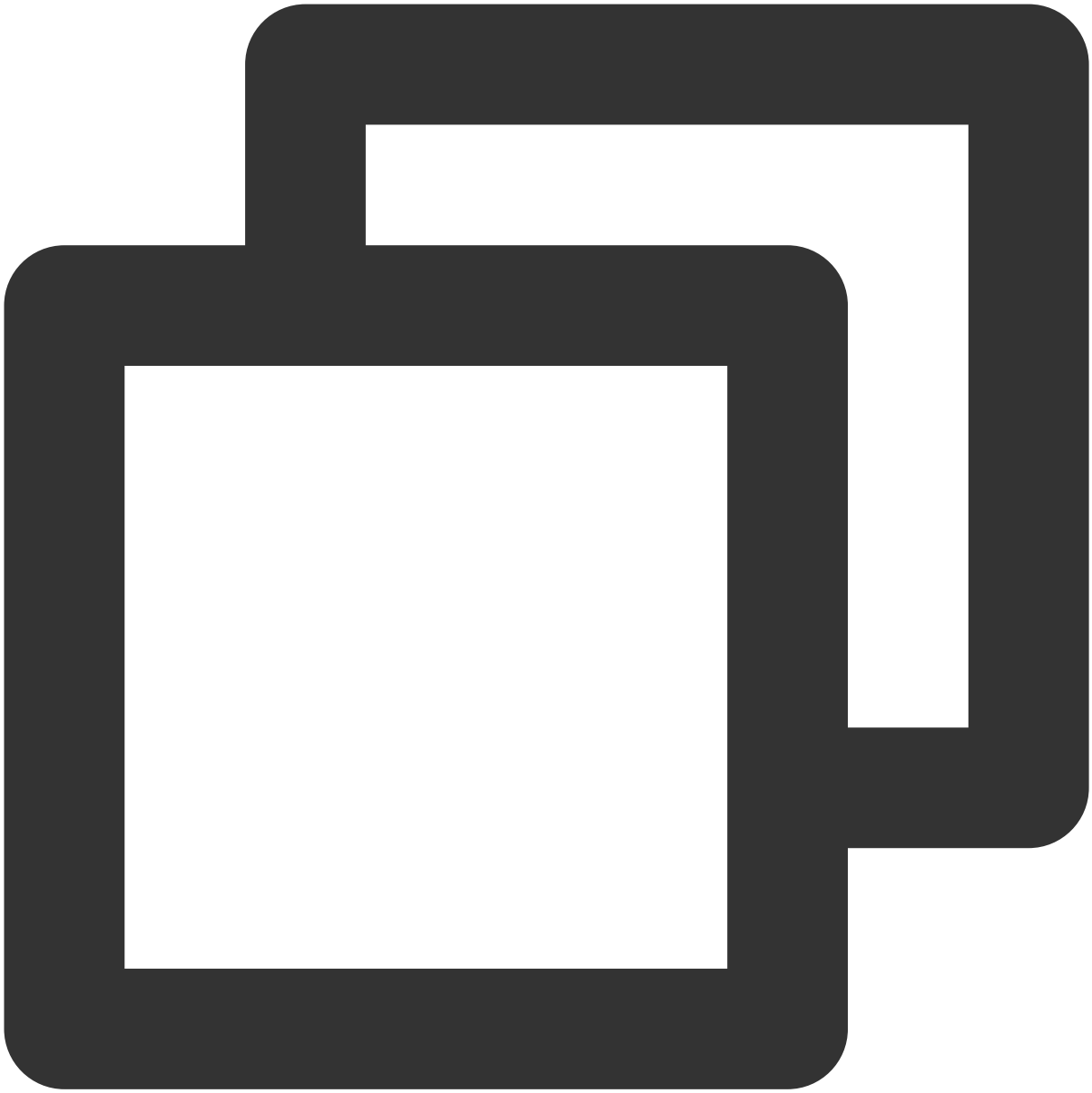
イベントの説明

Appがイベント通知を構成し、かつビデオを削除した場合、Appバックエンドは「通常のコールバック」または「信頼できるコールバック」によりこのイベント通知を取得することができます。イベント通知の内容は、FileDeleteTaskの構造です。

事例

通常のコールバック

通常のコールバックモードを選択すると、コールバックURLはVODからのHTTPリクエストを受信します。リクエストはPOSTメソッドを使用します。リクエストの内容は、以下に示すようにBODYにあります（null値のフィールドは省略されています）。

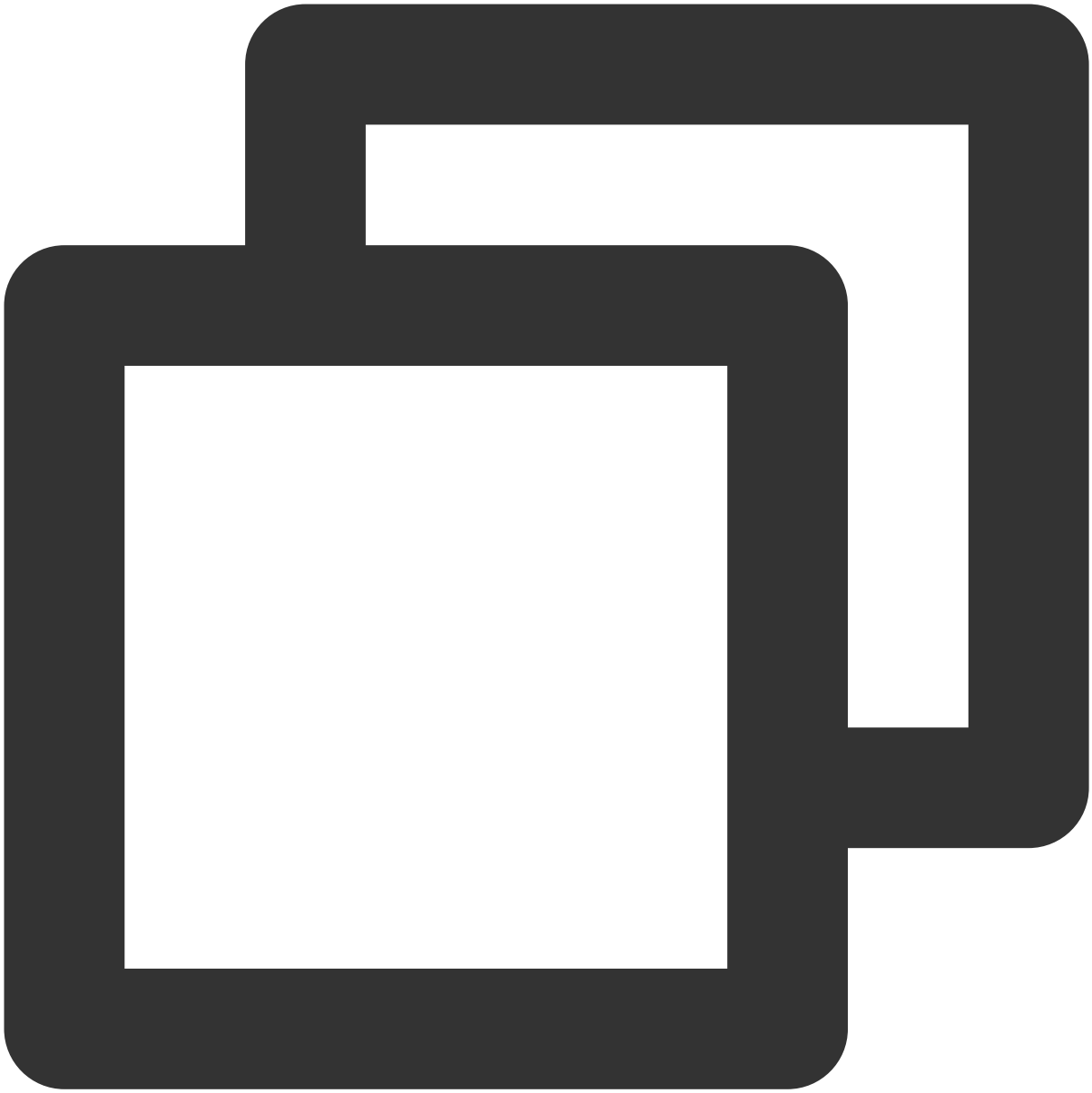


```
{
    "EventType":"FileDeleted",
    "FileDeleteEvent":{
        "FileIdSet":[
            "24961954183381008"
        ],
        "FileDeleteResultInfo":[
    {
                "FileId":"2496195418338 1008",
                "DeleteParts":[
                    {
```

```
                        "Type":"TranscodeFiles",
                        "Definition":0
                }
                 ]
    }
         ]
    }
}
```

信頼できるコールバック

信頼できるコールバックモードを選択し、プルイベント通知APIが呼び出された後に、次の形式のHTTPレスポンスを受信します（null値のフィールドは省略されています）。



```
{
    "Response":{
        "EventSet":[
    {
                "EventHandle":"EventHandle.N",
                "EventType":"FileDeleted",
                "FileDeleteEvent":{
                    "FileIdSet":[
                        "2496195418338100 8"
                    ],
                    "FileDeleteResultInfo":[
```

```
                        {
                            "FileId":"24961954183381008",
                            "DeleteParts":[
                                {
                                    "Type":"TranscodeFiles",
                                    "Definition":0
                }
                            ]
                }
                    ]
                }
    }
        ],
        "RequestId":"335bdaa3-db0e-46ce-9946-51941d9cb0f5"
    }
}
```

タスクフローステータスの変更

最終更新日：：2023-10-26 17:39:30

イベント名

ProcedureStateChanged

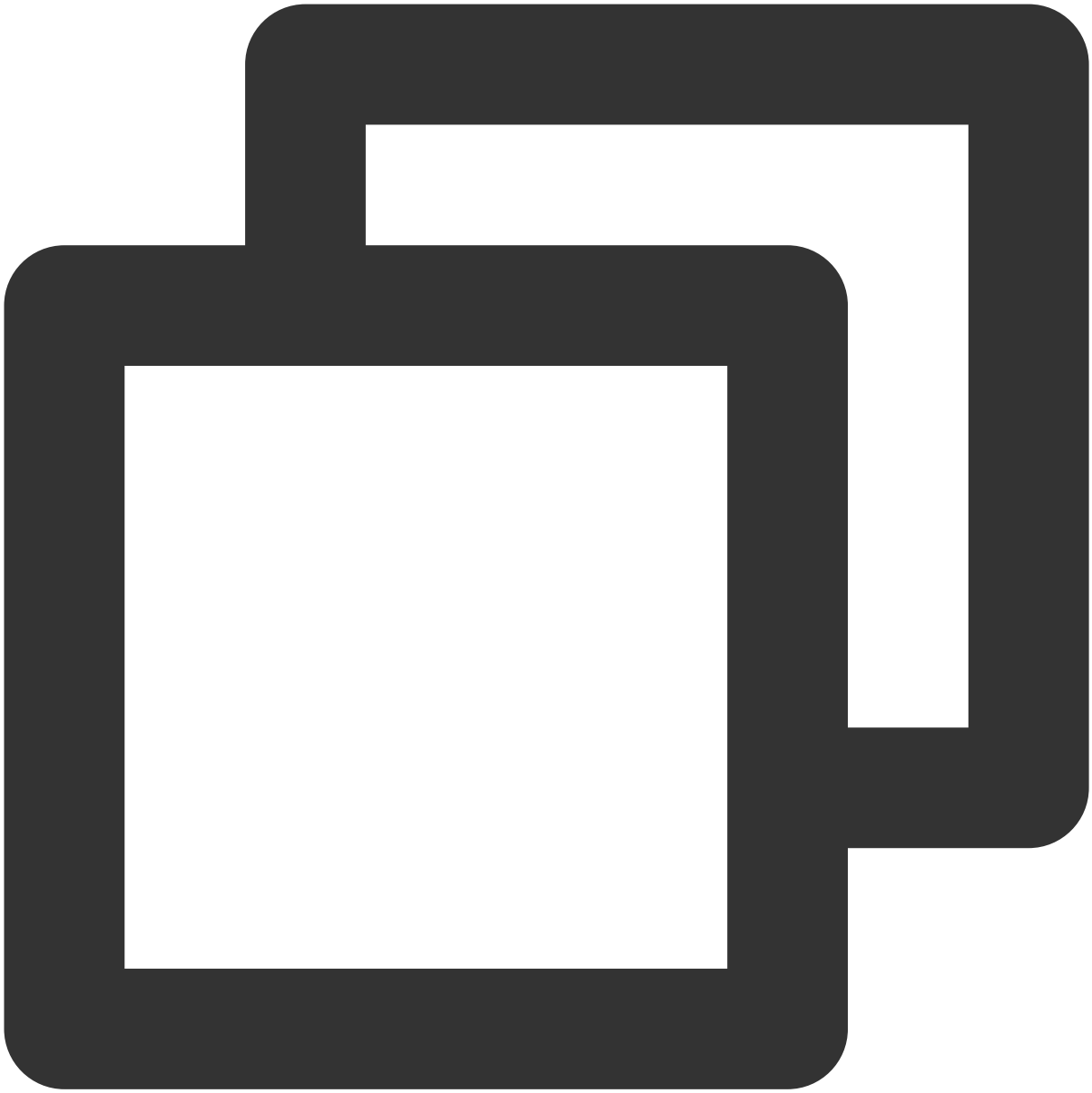
イベントの説明

Appがイベント通知を構成した後で、タスクフローのステータスが変更された場合、Appバックエンドは「通常のコールバック」または「信頼できるコールバック」によりこのイベント通知を取得することができます。イベント通知の内容は、ProcedureTaskの構造です。

事例

通常のコールバック

通常のコールバックモードを選択すると、コールバックURLはVODからのHTTPリクエストを受信します。リクエストはPOSTメソッドを使用します。リクエストの内容は、以下に示すようにBODYにあります（null値のフィールドは省略されています）。



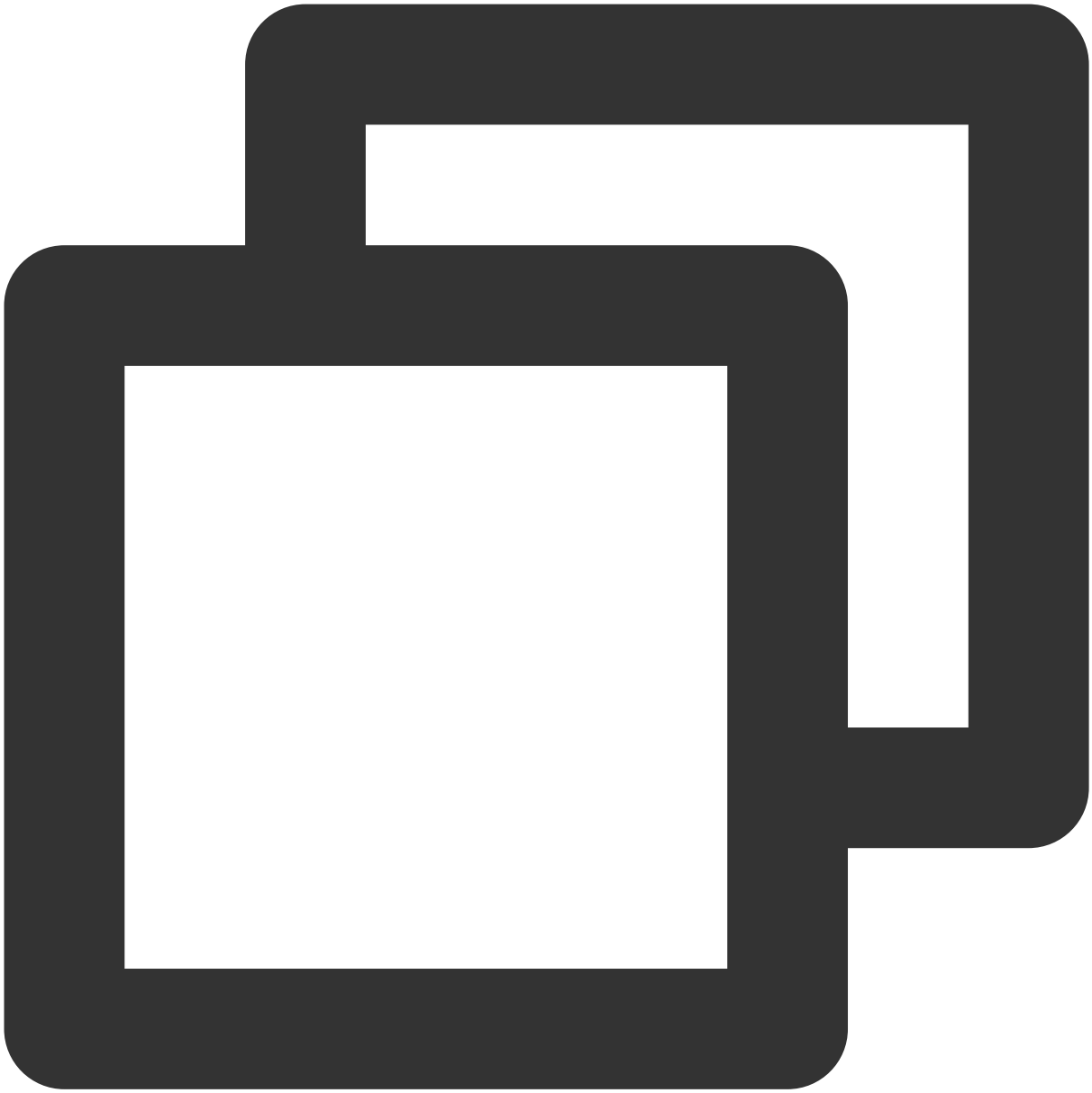

```
{
    "EventType":"ProcedureStateChanged",
    "ProcedureStateChangeEvent":{
        "TaskId":"1256768367-Procedure-475b72xxxcb177t1",
        "Status":"FINISH",
        "ErrCode":0,
        "Message":"",
        "FileId":"5285890784246869930",
        "FileName":"アニマルワールド",
        "FileUrl":"https://1256768367.vod2.myqcloud.com/xxx/xxx/xxx.mp4",
        "MetaData":{
```

```
            "AudioDuration":59.990001678467,
            "AudioStreamSet":[
                {
                    "Bitrate":383854,
                    "Codec":"aac",
                    "SamplingRate":48000
                }
            ],
            "Bitrate":1021028,
            "Container":"mov,mp4,m4a,3gp,3g2,mj2",
            "Duration":60,
            "Height":480,
            "Rotate":0,
            "Size":7700180,
            "VideoDuration":60,
            "VideoStreamSet":[
                {
                    "Bitrate":637174,
                    "Codec":"h264",
                    "Fps":23,
                    "Height":480,
                    "Width":640
                }
            ],
            "Width":640
        },
        "MediaProcessResultSet":[
    {
                "Type":"Transcode",
                "TranscodeTask":{
                    "Status":"SUCCESS",
                    "ErrCode":0,
                    "Message":"SUCCESS",
                    "Input":{
                        "Definition":20
                    },
                    "Output":{
                        "Url":"https://1256768367.vod2.myqcloud.com/xxx/xxx/v.f20.m
                        "Size":4189073,
                        "Container":"mov,mp4,m4a,3gp,3g2,mj2",
                        "Height":480,
                        "Width":640,
                        "Bitrate":552218,
                        "Md5":"eff7031ad7877865f9a3240e9ab165ad",
                        "Duration":60.04700088501,
                        "VideoStreamSet":[
                            {
```

```
                                "Bitrate":503727,
                                "Codec":"h264",
                                "Fps":24,
                                "Height":480,
                                "Width":640
                    }
                            ],
                            "AudioStreamSet":[
                                {
                                    "Bitrate":48491,
                                    "Codec":"aac",
                                    "SamplingRate":44100
                    }
                            ],
                            "Definition":0
                    }
                    }
                },
        {
                     "Type":"CoverBySnapshot",
                     "CoverBySnapshotTask":{
                         "Status":"SUCCESS",
                         "ErrCode":0,
                         "Message":"SUCCESS",
                         "Input":{
                             "Definition":10,
                             "PositionType":"Time",
                             "PositionValue":0
                         },
                         "Output":{
                             "CoverUrl":"http://1256768367.vod2.myqcloud.com/xxx/xxx/xxx
                    }
                    }
        }
            ]
        }
    }
```


信頼できるコールバック

信頼できるコールバックモードを選択し、プルイベント通知APIが呼び出された後に、次の形式のHTTPレスポンスを受信します（null値のフィールドは省略されています）。




```
{
    "Response": {
        "EventSet": [
    {
                "EventHandle": "EventHandleX",
                "EventType": "ProcedureStateChanged",
                "ProcedureStateChangeEvent": {
                    "TaskId": "1256768367-Procedure-475b72xxxcb177t1",
                    "Status": "FINISH",
                    "FileId": "5285890784246869930",
                    "FileName": "アニマルワールド",
```

```
                    "FileUrl": "https://1256768367.vod2.myqcloud.com/xxx/xxx/xxx.mp
                    "MetaData": {
                        "AudioDuration": 59.990001678467,
                        "AudioStreamSet": [{
                            "Bitrate": 383854,
                            "Codec": "aac",
                            "SamplingRate": 48000
                        }],
                        "Bitrate": 1021028,
                        "Container": "mov,mp4,m4a,3gp,3g2,mj2",
                        "Duration": 60,
                        "Height": 480,
                        "Rotate": 0,
                        "Size": 7700180,
                        "VideoDuration": 60,
                        "VideoStreamSet": [{
                            "Bitrate": 637174,
                            "Codec": "h264",
                            "Fps": 23,
                            "Height": 480,
                            "Width": 640
                        }],
                        "Width": 640
                    },
                    "MediaProcessResultSet": [{
                            "Type": "Transcode",
                            "TranscodeTask": {
                                "Status": "SUCCESS",
                                "ErrCode": 0,
                                "Message": "SUCCESS",
                                "Input": {
                                    "Definition": 20
                                },
                                "Output": {
                                    "Url": "https://1256768367.vod2.myqcloud.com/xx
                                    "Size": 4189073,
                                    "Container": "mov,mp4,m4a,3gp,3g2,mj2",
                                    "Height": 480,
                                    "Width": 640,
                                    "Bitrate": 552218,
                                    "Md5": "eff7031ad7877865f9a3240e9ab165ad",
                                    "Duration": 60.04700088501,
                                    "VideoStreamSet": [{
                                        "Bitrate": 503727,
                                        "Codec": "h264",
                                        "Fps": 24,
                                        "Height": 480,
```

```
                                    "Width": 640
                                }],
                                "AudioStreamSet": [{
                                    "Bitrate": 48491,
                                    "Codec": "aac",
                                    "SamplingRate": 44100
                                }],
                                "Definition": 0
                }
                }
                        },
                        {
                            "Type": "CoverBySnapshot",
                            "CoverBySnapshotTask": {
                                "Status": "SUCCESS",
                                "ErrCode": 0,
                                "Message": "SUCCESS",
                                "Input": {
                                    "Definition": 10,
                                    "PositionType": "Time",
                                    "PositionValue": 0
                                },
                                "Output": {
                                    "CoverUrl": "http://1256768367.vod2.myqcloud.co
                }
                }
                }
                     ]
                }
    }
        ],
        "RequestId": "335bdaa3-db0e-46ce-9946-51941d9cb0f5"
    }
}
```

ビデオ編集の完了

最終更新日：：2023-10-26 17:39:31

イベント名

EditMediaComplete

イベントの説明

Appがイベント通知で構成されている場合、かつビデオの編集が完了した場合、Appバックエンドは「通常のコールバック」または「信頼できるコールバック」を介してイベント通知を取得することができます。APIイベント通知の内容は、EditMediaTaskの構造です。

事例

通常のコールバック

通常のコールバックモードを選択すると、コールバックURLはVODからのHTTPリクエストを受信します。リクエストはPOSTメソッドを使用します。リクエストの内容は、以下に示すようにBODYにあります（null値のフィールドは省略されています）。

```
{
"EventType":"EditMediaComplete",
"EditMediaCompleteEvent":{
"TaskId":"1256768367-EditMedia-f5ac8127b3b6b85cdc13f237c6005d8",
"Status":"FINISH",
"ErrCode":0,
"Message":"SUCCESS",
"Input":{
"InputType":"File",
"FileInfoSet":[
{
"FileId":"24961954183381008",
"StartTimeOffset":0,
"EndTimeOffset":0
},
{
"FileId":"24961954183381009",
"StartTimeOffset":0,
```

```
"EndTimeOffset":0
},
{
"FileId":"24961954183381010",
"StartTimeOffset":0,
"EndTimeOffset":0
}
]
},
"Output":{
"FileType":"mp4",
"FileId":"24961954183923290",
"FileUrl":"http://125676836723.vod2.myqcloud.com/xxx/xxx/f0.mp4"
},
"ProcedureTaskId":"",
"ReviewAudioVideoTaskId":""
}
}
```


信頼できるコールバック

信頼できるコールバックモードを選択し、プルイベント通知APIが呼び出された後に、次の形式のHTTPレスポンスを受信します（null値のフィールドは省略されています）。


```
{
"Response": {
"EventSet": [
{
"EventHandle": "EventHandle.N",
"EventType": "EditMediaComplete",
"EditMediaCompleteEvent": {
"TaskId": "EditMedia-f5ac8127b3b6b85cdc13f237c6005d8",
"Status": "FINISH",
"ErrCode": 0,
"Message": "SUCCESS",
"Input": {
"InputType": "File",
"FileInfoSet": [
{
"FileId": "24961954183381008",
"StartTimeOffset": 0,
"EndTimeOffset": 0
},
{
"FileId": "24961954183381009",
"StartTimeOffset": 0,
```

```
"EndTimeOffset": 0
},
{
"FileId": "24961954183381010",
"StartTimeOffset": 0,
"EndTimeOffset": 0
}
]
},
"Output": {
"FileType": "mp4",
"FileId": "24961954183923290",
"FileUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/f0.mp4"
},
"ProcedureTaskId": "",
"ReviewAudioVideoTaskId": ""
}
}
],
"RequestId": "335bdaa3-db0e-46ce-9946-51941d9cb0f5"
}
}
```

ビデオ合成の完成

最終更新日：：2023-10-26 17:39:31

イベント名

ComposeMediaComplete

イベントの説明

Appがイベント通知で構成されている場合、かつビデオの合成が完了した場合、Appバックエンドは「通常のコールバック」または「信頼できるコールバック」を介してイベント通知を取得することができます。APIイベント通知の内容は、ComposeMediaTaskの構造です。

事例

通常のコールバック

通常のコールバックモードを選択すると、コールバックURLはVODからのHTTPリクエストを受信します。リクエストはPOSTメソッドを使用します。リクエストの内容は、以下に示すようにBODYにあります（null値のフィールドは省略されています）。

```
{
"EventType": "ComposeMediaComplete",
"ComposeMediaCompleteEvent": {
"TaskId": "1256768367-ComposeMedia-f5ac8127b3b6b85cdc13f237c6005d8",
"Status": "FINISH",
"ErrCode": 0,
"Message": "SUCCESS",
"Input": {
"Tracks": [{
"Type": "Video",
"TrackItems": [{
"Type": "Video",
"SourceMedia": "5285485487985271487",
"AudioOperations": [{
"Type": "Volume",
"VolumeParam": {
"Mute": 1
}
```

```
}]
}]
},
{
"Type": "Audio",
"TrackItems": [{
"Type": "Empty",
"EmptyItem": {
"Duration": 5
}
},
{
"Type": "Audio",
"AudioItem": {
"SourceMedia": "5285485487985271488",
"Duration": 15
}
},
{
"Type": "Audio",
"AudioItem": {
"SourceMedia": "5285485487985271489",
"SourceMediaStartTime": 2,
"Duration": 14
}
}
]
}
],
"Output": {
"FileName": "ビデオ合成効果テスト",
"Container": "mp4"
}
},
"Output": {
"FileType": "mp4",
"FileId": 5285485487985271490,
"FileUrl": "http://125676836723.vod2.myqcloud.com/xxx/xxx/xxx.mp4"
}
}
}
```


信頼できるコールバック

信頼できるコールバックモードを選択し、プルイベント通知APIが呼び出された後に、次の形式のHTTPレスポンスを受信します（null値のフィールドは省略されています）。

```
{
"Response":{
"EventSet":[
{
"EventHandle":"EventHandle.N",
"ComposeMediaCompleteEvent":{
"TaskId":"1256768367-ComposeMedia-f5ac8127b3b6b85cdc13f237c6005d8",
"Status":"FINISH",
"ErrCode":0,
"Message":"SUCCESS",
"Input":{
"Tracks":[
{
"Type":"Video",
"TrackItems":[
{
"Type":"Video",
"SourceMedia":"5285485487985271487",
"AudioOperations":[
{
"Type":"Volume",
"VolumeParam":{
"Mute":1
}
}
]
}
]
},
{
"Type":"Audio",
"TrackItems":[
{
"Type":"Empty",
"EmptyItem":{
"Duration":5
}
},
{
"Type":"Audio",
"AudioItem":{
"SourceMedia":"5285485487985271488",
"Duration":15
}
},
{
```



```
"Type":"Audio",
"AudioItem":{
"SourceMedia":"5285485487985271489",
"SourceMediaStartTime":2,
"Duration":14
}
}
]
}
],
"Output":{
"FileName":"ビデオ合成効果テスト",
"Container":"mp4"
}
},
"Output":{
"FileType":"mp4",
"FileId":5285485487985271490,
"FileUrl":"http://125676836723.vod2.myqcloud.com/xxx/xxx/xxx.mp4"
}
}
}
],
"RequestId":"335bdaa3-db0e-46ce-9946-51941d9cb0f5"
}
}
```

ビデオ取得の完了

最終更新日：：2023-10-26 17:39:30

イベント名

RestoreMediaComplete

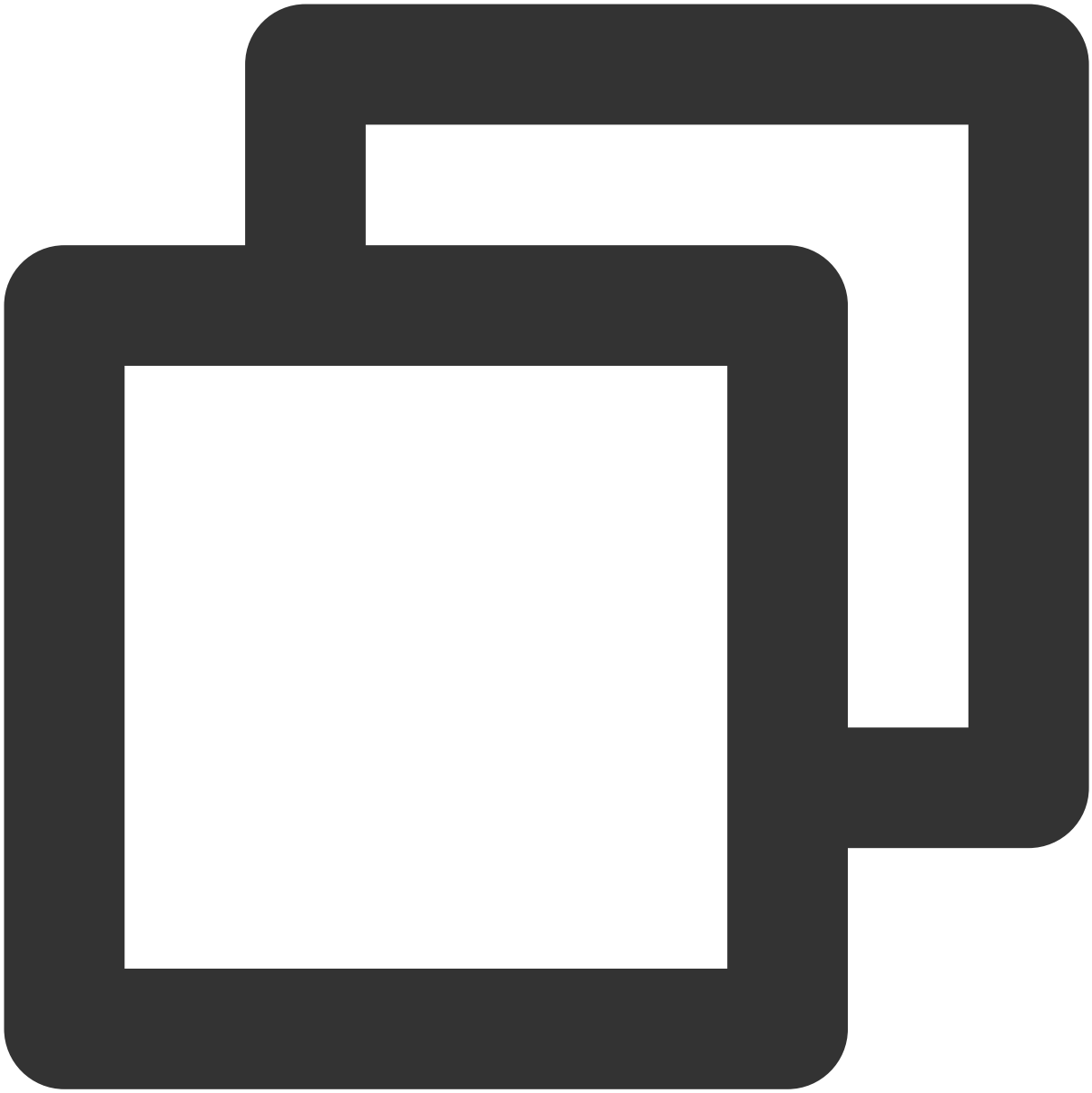
イベントの説明

Appがイベント通知を構成し、かつアーカイブまたはディープアーカイブされたメディアファイルの解凍または取得が完了している場合、Appバックエンドは「通常のコールバック」または「信頼できるコールバック」によりイベント通知を取得することができます。イベント通知の内容は、RestoreMediaTaskの構造です。

事例

通常のコールバック

通常のコールバックモードを選択すると、コールバックURLは次の形式のHTTPリクエストを受信します。

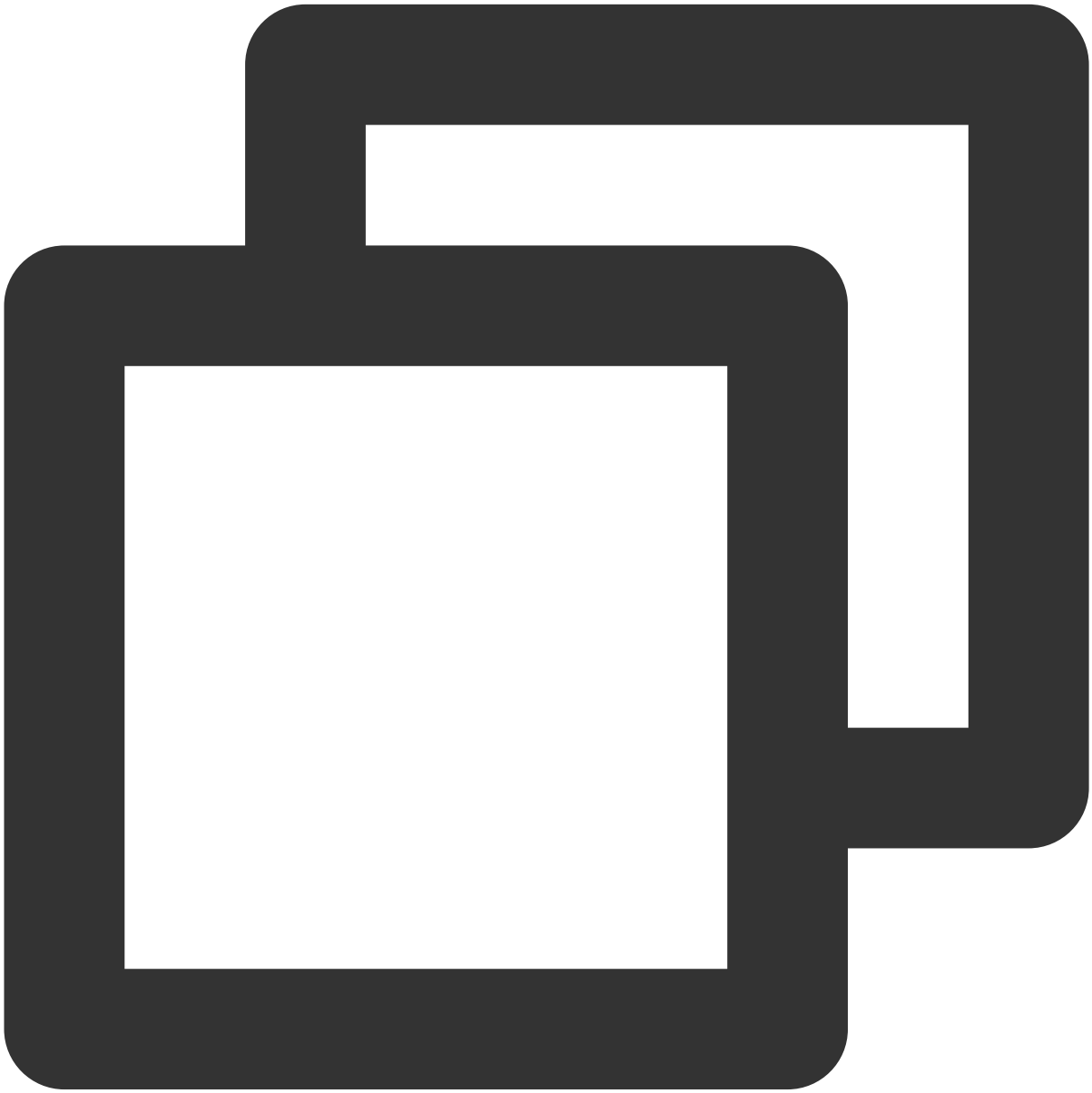


```
{
    "EventType":"RestoreMediaComplete",
    "RestoreMediaCompleteEvent":{
        "FileId":"2496195418338100 8",
        "OriginalStorageClass":"ARCHIVE",
        "TargetStorageClass":"STANDARD",
        "RestoreTier":"Standard",
        "RestoreDay":0,
        "Status":0,
        "Message":"Restore success!"
    }
```

```
}
```

信頼できるコールバック

信頼できるコールバックモードを選択し、プルイベント通知APIが呼び出された後に、次の形式のHTTPレスポンスを受信します。



```
{
    "Response":{
        "EventSet":[
    {
```

```
                "EventHandle":"EventHandle.N",
                "EventType":"RestoreMediaComplete",
                "RestoreMediaCompleteEvent":{
                    "FileId":"24961954183381008",
                    "OriginalStorageClass":"ARCHIVE",
                    "TargetStorageClass":"STANDARD",
                    "RestoreTier":"Standard",
                    "RestoreDay":0,
                    "Status":0,
                    "Message":"Restore success!"
                }
    }
        ],
        "RequestId":"335bdaa3-db0e-46ce-9946-51941d9cb0f5"
    }
}
```

オーディオビデオ審査完了

最終更新日：：2023-10-26 17:39:31

イベント名

ReviewAudioVideoComplete

イベントの説明

Appにイベント通知が設定され、かつオーディオビデオ審査が完了した場合、Appバックエンドは「通常のコールバック」または「信頼できるコールバック」からイベント通知を取得することができます。イベント通知の内容はReviewAudioVideoTaskの構造です。

事例

通常のコールバック

通常のコールバックモードを選択すると、コールバックURLはVODからのHTTPリクエストを受信します。リクエストはPOSTメソッドを使用します。リクエストの内容は、以下に示すようにBODYにあります（null値のフィールドは省略されています）。

```
{
"EventType": "ReviewAudioVideoComplete",
"ReviewAudioVideoCompleteEvent": {
"TaskId": "125xxxx-ReviewAudioVideo-07edbc78ba20563cdf2362cffbf4aa0ct",
"Status": "FINISH",
"ErrCodeExt": "",
"Message": "SUCCESS",
"Input":{
"FileId": "387702130626135215"
},
"Output":{
"Suggestion": "block",
"Label": "porn",
"Form": "Image",
"SegmentSet":[
{
"StartTimeOffset": 0,
"EndTimeOffset": 1,
```

```
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 1,
"EndTimeOffset": 2,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 2,
"EndTimeOffset": 3,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 3,
"EndTimeOffset": 4,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 4,
```

```
"EndTimeOffset": 5,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 5,
"EndTimeOffset": 6,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 6,
"EndTimeOffset": 7,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 7,
"EndTimeOffset": 8,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
```

```
"StartTimeOffset": 8,
"EndTimeOffset": 9,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 9,
"EndTimeOffset": 10,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
}
],
"SegmentSetFileUrl": "http://251000800.vod2.myqcloud.com/a8800b40vodtranssgp251000800/0f9bd2b0-34a8-4642-f481-001894d93019.txt",
"SegmentSetFileUrlExpireTime": "2022-10-12T07:01:07.695Z"
},
"SessionContext": "",
"SessionId": ""
}
}
```


信頼できるコールバック

信頼できるコールバックモードを選択し、プルイベント通知 APIが呼び出された後に、次の形式のHTTPレスポンスを受信します（null値のフィールドは省略されています）。


```
{
"Response":{
"EventSet":[
{
"EventHandle": "EventHandle.N",
"EventType": "ReviewAudioVideoComplete",
"ReviewAudioVideoCompleteEvent": {
"TaskId": "125xxxx-ReviewAudioVideo-07edbc78ba20563cdf2362cffbf4aa0ct",
```

```
"Status": "FINISH",
"ErrCodeExt": "",
"Message": "SUCCESS",
"Input":{
"FileId": "387702130626135215"
},
"Output":{
"Suggestion": "block",
"Label": "porn",
"Form": "Image",
"SegmentSet":[
{
"StartTimeOffset": 0,
"EndTimeOffset": 1,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 1,
"EndTimeOffset": 2,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 2,
"EndTimeOffset": 3,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
```



```
{
"StartTimeOffset": 3,
"EndTimeOffset": 4,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 4,
"EndTimeOffset": 5,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 5,
"EndTimeOffset": 6,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 6,
"EndTimeOffset": 7,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
```

```
},
{
"StartTimeOffset": 7,
"EndTimeOffset": 8,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 8,
"EndTimeOffset": 9,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
},
{
"StartTimeOffset": 9,
"EndTimeOffset": 10,
"Confidence": 99,
"Suggestion": "block",
"Label": "Porn",
"SubLabel": "porn",
"Form": "Image",
"AreaCoordSet": [],
"Text": "",
"KeywordSet": []
}
],
"SegmentSetFileUrl": "http://251000800.vod2.myqcloud.com/a8800b40vodtranssgp251000800/0f9bd2b0-34a8-4642-f481-001894d93019.txt",
"SegmentSetFileUrlExpireTime": "2022-10-12T07:01:07.695Z"
},
"SessionContext": "",
"SessionId": ""
}
}
],
```

```
"RequestId": "335bdaa3-db0e-46ce-9946-51941d9cb0f5"
}
}
```

ビデオ再生
ビデオ再生の概要

最終更新日：：2023-10-26 17:39:30

VODは、アップロードおよびトランスコードされたビデオを再生するための複数の形式をサポートしています。再生は主に、短編ビデオの再生、長編ビデオの再生および暗号化ビデオ再生といったいくつかのシナリオに分けられます。

短編ビデオの再生

短編ビデオとは、通常は次のような視聴時間が5分以内のビデオです。

- UGSVソーシャルプラットフォーム（微視（WeSee）、快手（Kuaishou）、抖音 （Douyin））で共有されるビデオ。
- eコマースショッピングプラットフォーム（京東（JD.com）、拼多多 （Pinduoduo））の商品PRビデオ。
- WeChat公式アカウント、メディア で共有されるショートムービー。

長編ビデオの再生

長編ビデオは、通常、次のような専門機関によって制作され、動画サイトを介して公開されるビデオのことです。

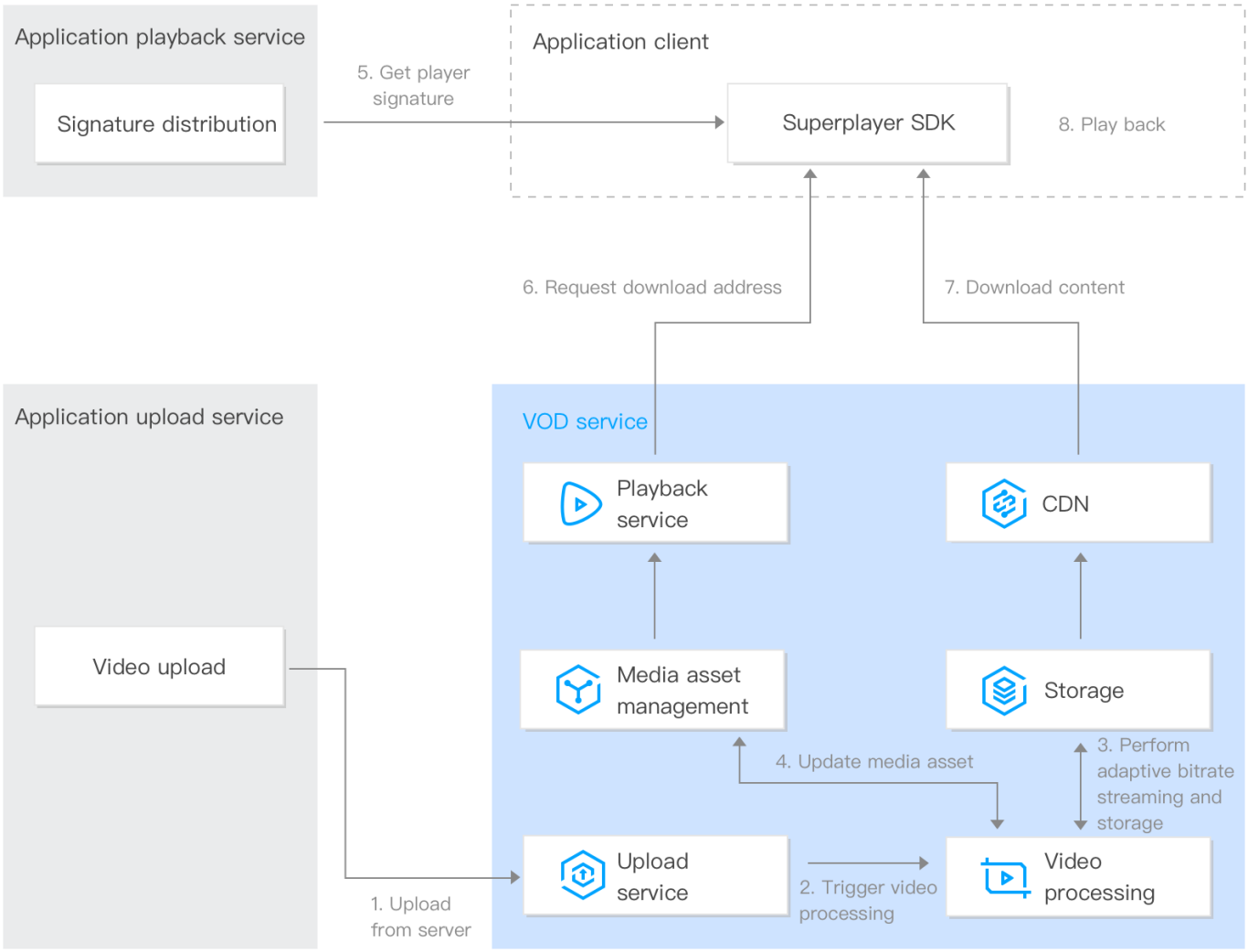
- ビデオメディアプラットフォーム（Tencent Video、優酷（Youku）、愛奇芸（iQIYI））で公開される独占放送のドラマ、バラエティ番組。
- eラーニングWebサイト（Tencent Class、Penguin Tutoring ）の学習ビデオ。
- ネットワークTVプラットフォーム（CNTV、Mango TV）のテレビ番組ビデオの再生。

暗号化されたビデオの再生

ビデオの暗号化は、長編ビデオのシーンでは特定のシーンに該当します。ビデオプラットフォームの独占ドラマ、オンライン学習などの著作権で保護されたビデオに対し、ビデオが違法にダウンロードや配布されるのを防止するために暗号化を採用しています。

再生アーキテクチャ

さまざまなビデオ再生シナリオにおいて、VODでは**Player+**を使用して、アダプティブビットレートストリーミングに変換された出力ビデオを再生することをお勧めします。再生の全体的なアーキテクチャの流れは、以下のとおりです。



1. **サーバーからのアップロード**：ビジネスバックエンドでは、コンソールやサーバーAPIといった形式により、ビデオをVODにアップロードします。
2. **ビデオ処理のトリガー**：ビデオをアップロードすると同時に、アダプティブビットレートストリーミングが指定されます。アップロード後、ビデオ処理が開始されます。
3. **アダプティブビットレートストリーミングとストレージへの書き込み**：ビデオがアダプティブビットレートストリーミングに変換された後、出力されたビデオコンテンツはVODのストレージに書き込まれます。
4. **メディア資産の更新**：アダプティブビットレートストリーミングに変換されたビデオ情報は、メディア資産管理モジュールに書き込まれます。
5. **署名配布**：ビジネスバックエンドは、プレーヤーの署名計算ルールに従ってプレーヤーの再生署名を配布します。
6. **ダウンロードアドレスのリクエスト**：プレーヤーは、再生するビデオのFileIdを指定した後、VODの再生サービスからビデオのダウンロードアドレスを取得します。
7. **コンテンツのダウンロード**：プレーヤーは、ダウンロードアドレスを介してVOD CDNからコンテンツをダウンロードします。

8. **再生**：プレーヤーは、出力されたアダプティブビットレートストリーミングの再生を開始します。

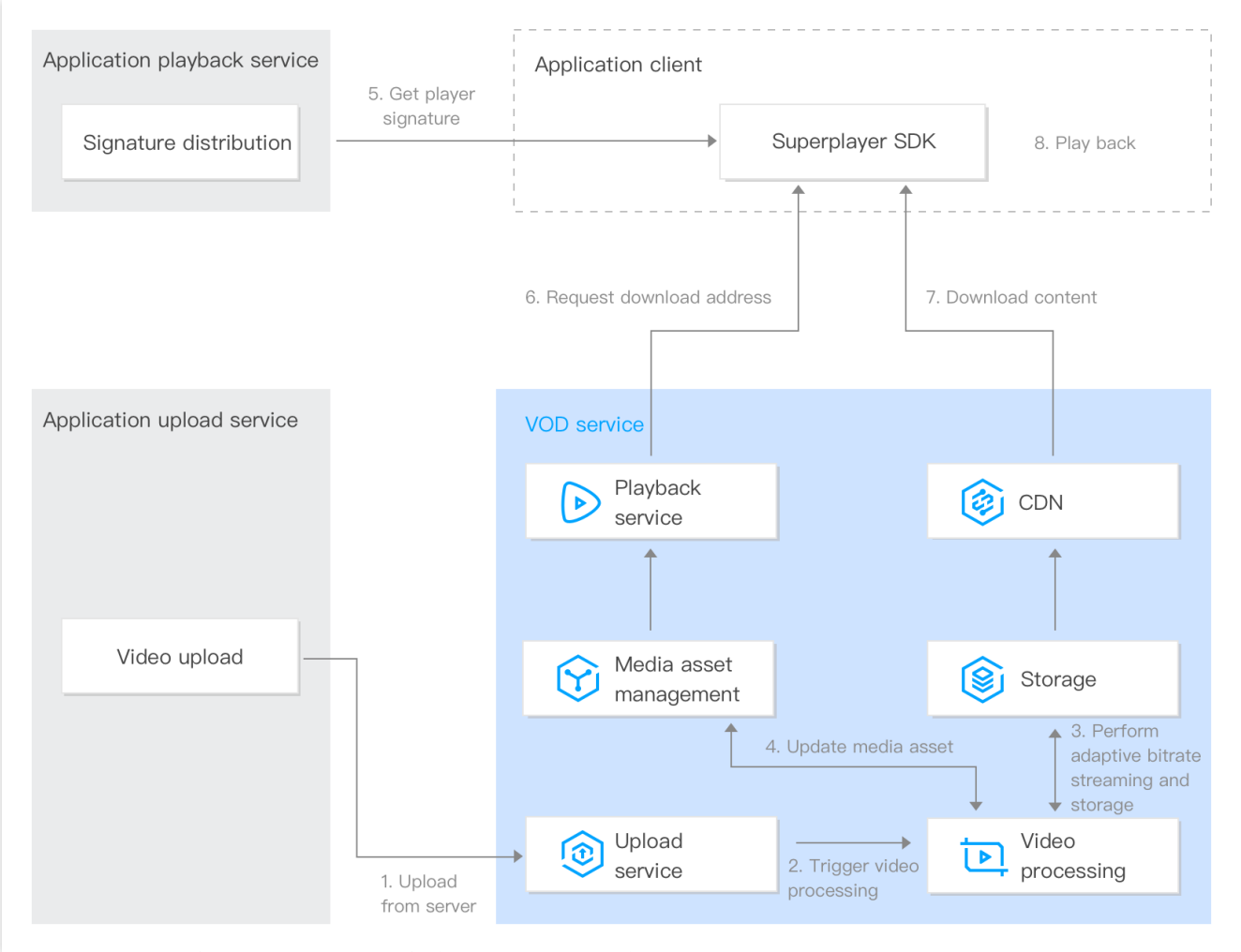
ドキュメントガイド

- Player+がサポートする機能については、機能説明を、統合方法については、SDKダウンロードをご参照ください。
- VODのプレーヤーを素早く統合できるように、Player+のアクセスガイドを提供し、統合手順について例示して説明しています。
- ビデオ暗号化再生シナリオの場合、VODはビデオ暗号化の概要とビデオ暗号化の統合ガイドで、それぞれビデオ暗号化の原理と統合方法について説明しています。

プレーヤーの署名

最終更新日：：2023-10-26 17:39:31

プレーヤー署名は、App再生サービスが端末の再生権限を承認するために使用されます。下図の手順6に示すとおり、 App再生サービスが端末での再生を承認すると、有効な署名が配布されます。端末では、署名の有効時間内にビデオコンテンツを再生することができます。



以下で、プレーヤー署名のパラメータと生成規則について説明します。

署名パラメータ

パラメータ名	記入必須	タイプ	説明
appId	はい	Integer	VODアプリケーションのappId。
fileId	はい	String	VODファイルID。

contentInfo	はい	Object	VODファイルIDに対応して再生される具体的なコンテンツ。ContentInfoタイプです。次の3種類のいずれかを再生できます。 アダプティブビットレートストリーミングへのトランスコードを行った出力オーディオビデオ。暗号化はされていてもいなくてもかまいません。 トランスコードされた出力オーディオビデオ。 アップロードされたオリジナルオーディオビデオ。
currentTimeStamp	はい	Integer	署名配布時点のUnixタイムスタンプ。
expireTimeStamp	いいえ	Integer	署名配布有効期限のUnixタイムスタンプ。入力しない場合は有効期限がないことを表します。
urlAccessInfo	いいえ	Object	再生リンクアクセス設定パラメータ。Keyリンク不正アクセス防止設定、再生ドメイン名、プロトコルパラメータが含まれます。UrlAccessInfoタイプです。
drmLicenseInfo	いいえ	Object	DRM License設定パラメータ。DrmLicenseInfoタイプです。

ContentInfoタイプ

パラメータ名	記入必須	タイプ	説明
audioVideoType	はい	String	再生するオーディオビデオタイプ。オプション値は次のとおりです。 RawAdaptive：暗号化されていないアダプティブビットレートストリーミングへのトランスコード出力。 ProtectedAdaptive：プライベート暗号化またはDRMで保護されたアダプティブビットレートストリーミングへのトランスコード出力。 Transcode：トランスコード後の出力。 Original：アップロードされたオリジナルオーディオビデオ。
rawAdaptiveDefinition	いいえ	Integer	出力が許可される、暗号化されていないABS生成テンプレートのID。audioVideoTypeがRawAdaptiveの場合のみ、このパラメータが入力必須かつ有効になります。
drmAdaptiveInfo	いいえ	Object	出力が許可される、暗号化によって保護されたABS生成テンプレートのID。audioVideoTypeがProtectedAdaptiveの場合のみ、このパラメータが入力必須かつ有効になります。DRMAdaptiveInfoタイプです。

transcodeDefinition	いいえ	Integer	出力が許可されるトランスコードテンプレートのID。audioVideoTypeがTranscodeの場合のみ、このパラメータが入力必須かつ有効になります。
imageSpriteDefinition	いいえ	Integer	プログレスバープレビューに使用するスプライトイメージテンプレートのID。
resolutionNames	いいえ	Array of Object	プレーヤーの各解像度に対応するサブストリーム表示名。ResolutionNameInfoタイプの配列です。空のままにするか、またはNULL配列を入力するとデフォルト設定を使用します。 MinEdgeLength：240，Name：240P。 MinEdgeLength：480，Name：480P。 MinEdgeLength：720，Name：720P。 MinEdgeLength：1080，Name：1080P。 MinEdgeLength：1440，Name：2K。 MinEdgeLength：2160，Name：4K。 MinEdgeLength：4320，Name：8K。

DRMAdaptiveInfoタイプ

パラメータ名	記入必須	タイプ	説明
privateEncryptionDefinition	いいえ	Integer	保護タイプDrmTypeがSimpleAESであるABS生成テンプレートのID。
widevineDefinition	いいえ	Integer	保護タイプDrmTypeがWidevineであるABS生成テンプレートのID。
fairPlayDefinition	いいえ	Integer	保護タイプDrmTypeがFairPlayであるABS生成テンプレートのID。

ResolutionNameInfoタイプ

パラメータ名	記入必須	タイプ	説明
MinEdgeLength	はい	Integer	ビデオ短辺の長さ。単位：ピクセル。
Name	はい	String	表示名。

UrlAccessInfo タイプ

パラメータ名	記入必須	タイプ	説明
t	いいえ	String	16進数文字列。リンクの有効期限を表します。

			具体的な意味と値についてはリンク不正アクセス防止パラメータのtパラメータをご参照ください。 入力しない場合は有効期限がないことを表します。
exper	いいえ	Integer	プレビュー時間。単位は秒で、10進数で表示されます。 プレビュー時間を指定したい場合、時間は30秒以上にする必要があります。 具体的な意味と値についてはリンク不正アクセス防止パラメータのexperパラメータをご参照ください。
rlimit	いいえ	Integer	再生可能な最大端末IP数。10進数で表示されます。 具体的な意味と値についてはリンク不正アクセス防止パラメータのrlimitパラメータをご参照ください。
us	いいえ	String	リンクID。ユーザーのリンクの一意性を強化します。 具体的な意味と値についてはリンク不正アクセス防止パラメータのusパラメータをご参照ください。
domain	いいえ	String	再生時に使用するドメイン名。入力しない場合またはDefaultを入力した場合は、デフォルト配信設定のドメイン名を使用することを表します。
scheme	いいえ	String	再生時に使用するScheme。入力しない場合またはDefaultを入力した場合は、デフォルト配信設定のSchemeを使用することを表します。その他のオプション値は次のとおりです。 HTTP。 HTTPS
uv	いいえ	String	6桁の16進数の文字。トレーサビリティウォーターマークのシーンで使用します。

DrmLicenseInfo タイプ

パラメータ名	記入必須	タイプ	説明

persistent	いいえ	String	端末に商用DRM再生ライセンスの永続的な保存を許可するかどうか。値の範囲： ON: 永続的な保存を許可します。 OFF: 永続的な保存を許可しません。 デフォルトの値はOFFになっています。
rentalDuration	いいえ	Integer	persistentがONの場合に、商用DRM再生ライセンスによって許可される永続的な保存時間。単位は秒で、入力しない場合は時間制限がないことを表します。
forceL1TrackTypes	いいえ	Array of String	Widevineを使用する際に、端末に必ずL1セキュリティレベルを使用して処理するよう要求するTrackタイプ。このうち、未指定のTrackタイプはデフォルトでL3セキュリティレベルを使用して処理します。 値の範囲は次のとおりです。 AUDIO: オーディオサブストリーム。 SD: 短辺が720未満のサブストリーム。 HD: 短辺が720以上かつ2160未満のサブストリーム。 UHD1: 短辺が2160以上かつ4320未満のサブストリーム。 UHD2: 短辺が4320以上のサブストリーム。

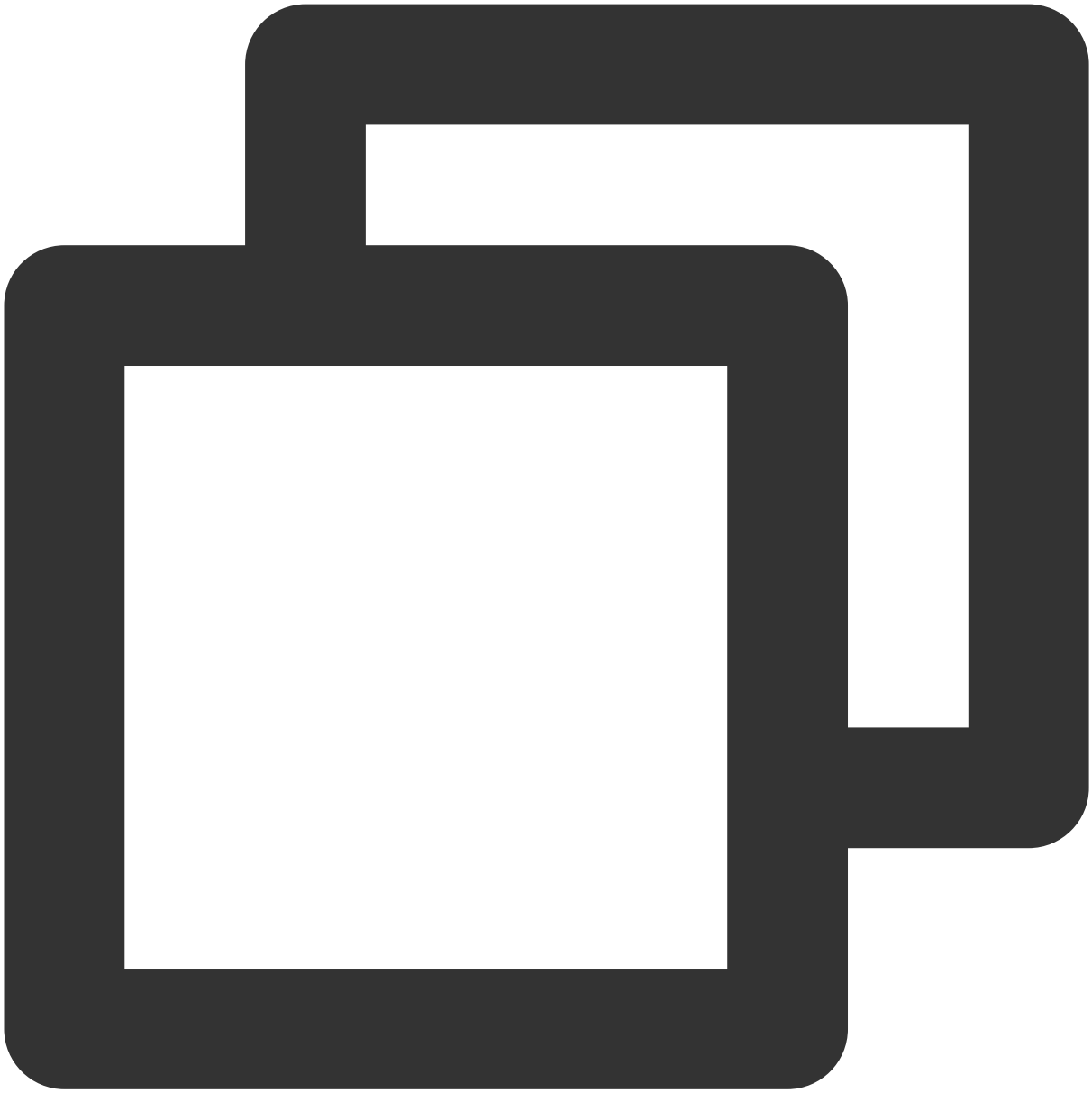
説明:
サブアプリケーションを使用する場合、appIdパラメータにはサブアプリケーションAppIdを入力する必要があります。
署名パラメータの `t` 、 `exper` 、 `rlimit` 、 `us` の説明と値は、 リンク不正アクセス防止パラメータの同名パラメータと完全に一致します。

署名計算

VODプレーヤーの署名には、Header、PayLoad、Key によって計算され組み合わせられたデジタルトークンである JWT（JSON Web Token）を採用します。

Header

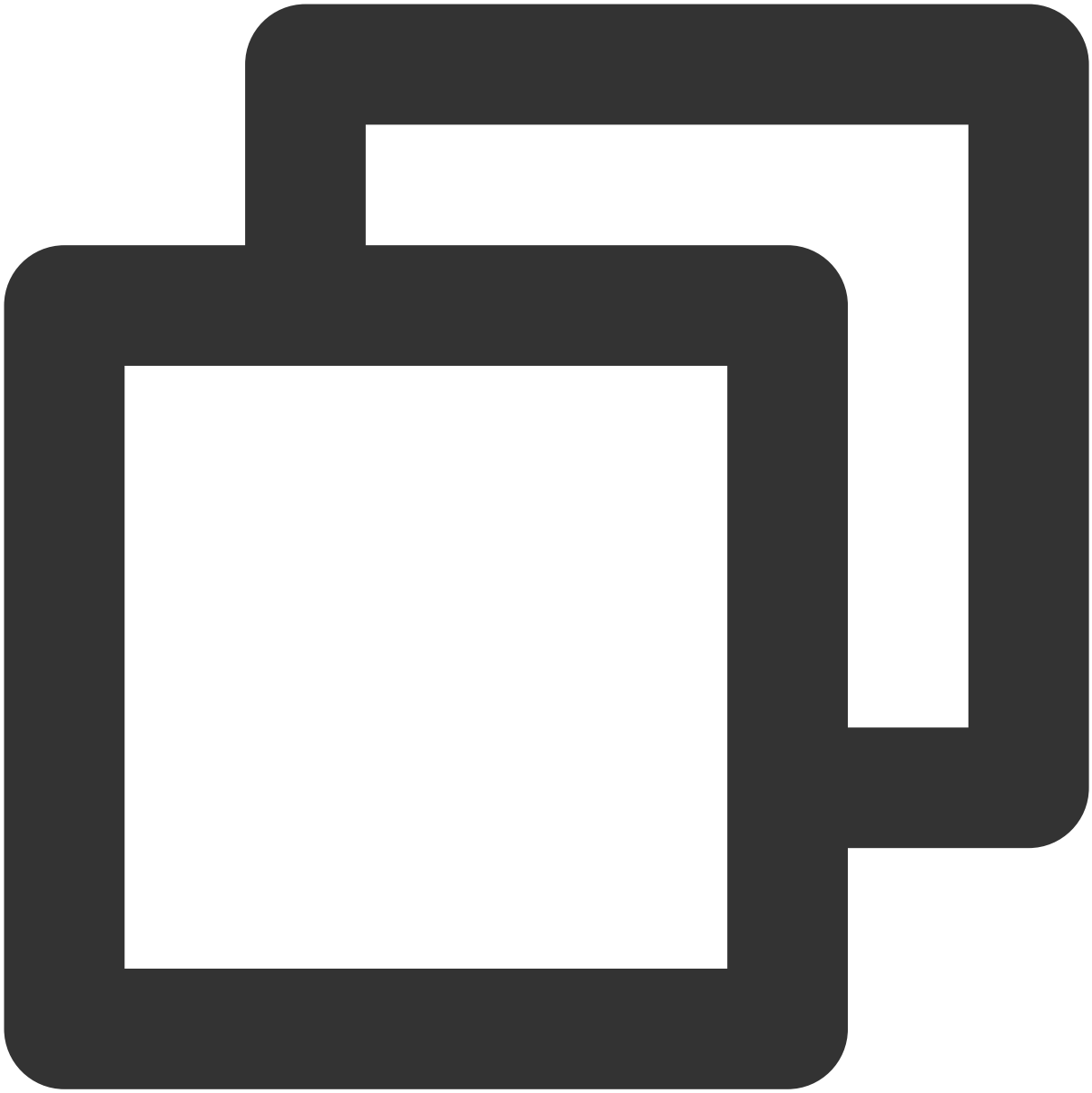
HeaderはJSON形式であり、JWTで使用されるアルゴリズム情報を表し、その内容は次のとおり固定的 に使用されます：



```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PayLoad

Payload はJSON形式であり、次に例示するようにプレーヤーの署名パラメータのコンテンツです。




```
{
  "appId": 1255566655,
  "fileId": "4564972818519602447",
  "contentInfo": {
    "audioVideoType": "RawAdaptive",
    "rawAdaptiveDefinition": 10,
    "imageSpriteDefinition": 10
  },
  "currentTimeStamp": 1663064276,
  "expireTimeStamp": 1663294210,
  "urlAccessInfo": {
```

```
      "t": "6323e6b0",
      "rlimit": 3,
      "us": "72d4cd1101"
    }
  }
```

Key

Keyは署名の計算時に使用するキーです。ここではデフォルト配信設定の `再生キー` を使用します。

計算式

1. Signatureの計算：

`Signature = HMACSHA256(base64UrlEncode(Header) + "." + base64UrlEncode(Payload), Key)`

2. Tokenの計算：

`Token = base64UrlEncode(Header) + '.' + base64UrlEncode(Payload) + '.' + base64UrlEncode(Signature)`

最終的に得られたTokenが、VODプレーヤー署名となります。

説明:

HMACSHA256については、RFC - HMACSHA256をご参照ください。base64UrlEncodeについては、RFC - base64UrlEncodeをご参照ください。

署名の計算および署名の検証を容易にするため、VODコンソールでは署名発行ツールおよび検証ツールを提供しています。

プレーヤー署名ツール。

計算例

例えばあるユーザーの、appIdが `1255566655` 、fileIdが `4564972818519602447` のビデオについてプレーヤー署名を生成する場合で、かつ以下のようであったとします。

再生キーが `TxtyhLlgo7J3iOADIron` 。

プレーヤー署名の配布時間が2022-09-13 18:17:56、対応するUnix時間が `1663064276` 。

プレーヤー署名の有効期限が2022-09-16 10:10:10、対応するUnix時間が `1663294210` 。

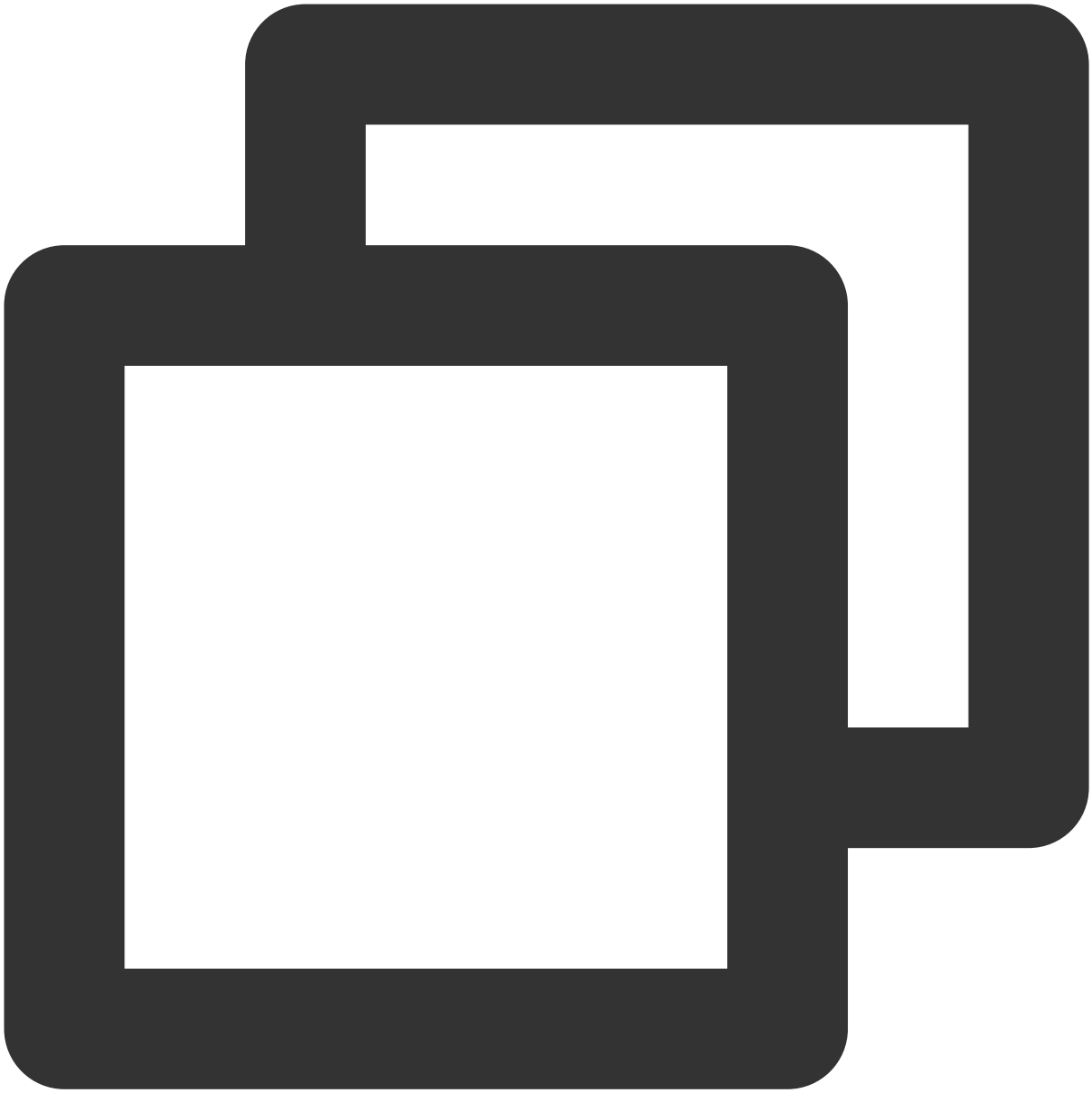
リンク不正アクセス防止の有効期限が2022-09-16 11:00:00、対応するUnix時間が `6323e6b0` 。

URLでの再生を最大3つの異なるIPで許可。

生成されたランダムな文字列が `72d4cd1101` 。

この場合の署名手順は、次のとおりです.

1. Headerのコンテンツ：

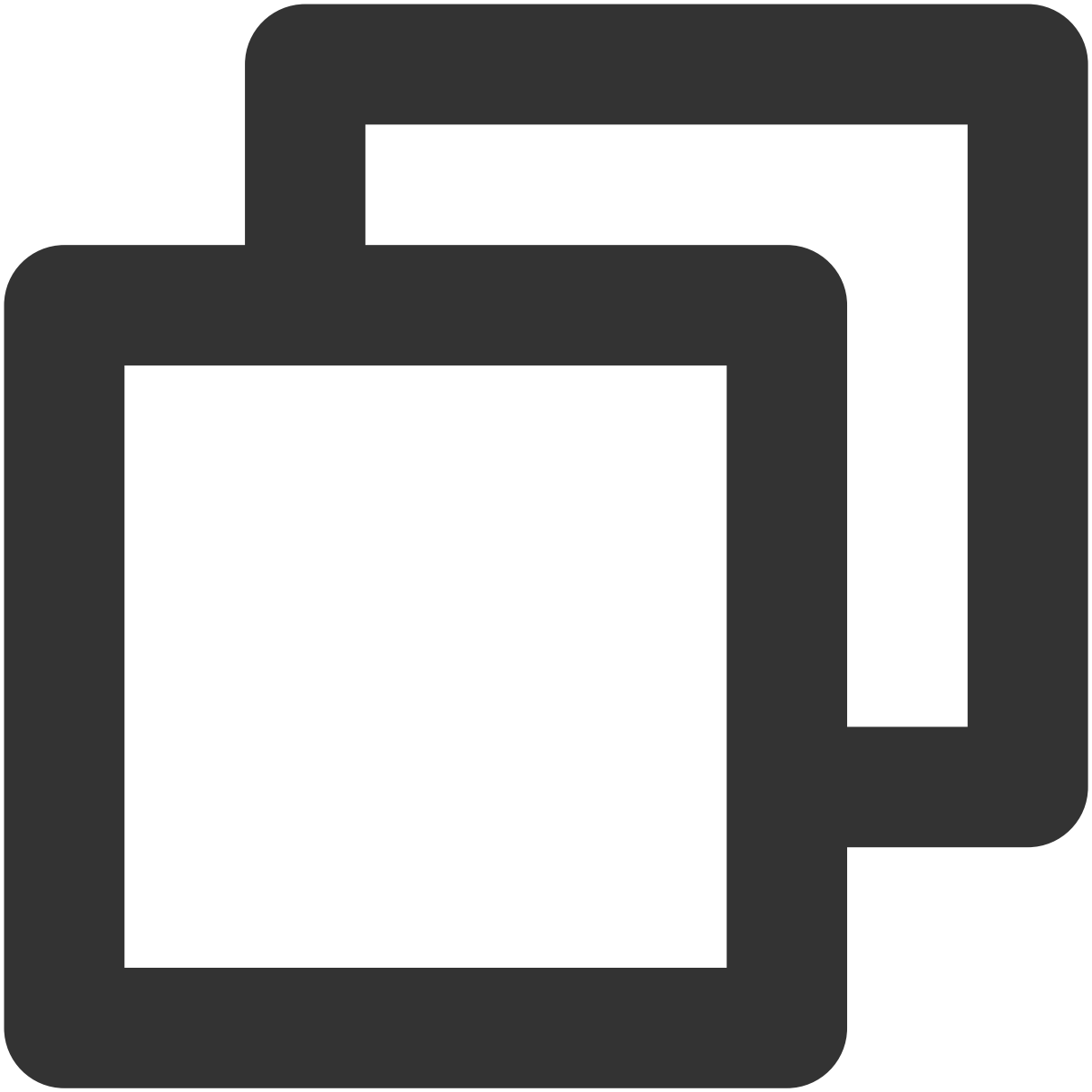


```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

base64UrlEncode で処理した後の結果：

`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9` 。

2. Payloadのコンテンツ：




```
{
  "appId": 1255566655,
  "fileId": "4564972818519602447",
  "contentInfo": {
    "audioVideoType": "RawAdaptive",
    "rawAdaptiveDefinition": 10,
    "imageSpriteDefinition": 10
  },
  "currentTimeStamp": 1663064276,
  "expireTimeStamp": 1663294210,
  "urlAccessInfo": {
```

```
      "t": "6323e6b0",
      "rlimit": 3,
      "us": "72d4cd1101"
    }
  }
```

base64UrlEncode で処理した後の結果：

`eyJhcHBJZCI6MTI1NTU2NjY1NSwiZmlsZUlkIjoiNDU2NDk3MjgxODUxOTYwMjQ0NyIsImNvbnRlbnRJbmZvMSI6eyJhdWRpb1ZpZGVvVHlwZSI6IlJhd0FkYXB0`
`aXZlIiwicmF3QWRhcHRpdmVEZWZpbml0aW9uIjoxMCwiaW1hZ2VTcHJpdGVEZWZpbml0aW9uIjoxMH0sImN1cnJlbnRUaW1lU3RhbXAiOjE2NjMwNjQyNzYsImV4`
`cGlyZVRpbWVTdGFtcCI6MTY2MzI5NDIxMCwidXJsQWNjZXNzSW5mbyI6eyJ0IjoiNjMyM2U2YjAiLCJybGltaXQiOjMsInVzIjoiNzJkNGNkMTEwMSJ9fQ` 。

3. 再生キーをKey（ `TxtyhLlgo7J3iOADIron` ）としてHMAC計算を実行した場合、Signatureは次のようになります。

`QFcBX9830ysTzJIyZxoOlRmNb2Gqy2fns9yOfriaDI8` 。

4. 最終的にTokenはこのようになります。

`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTI1NTU2NjY1NSwiZmlsZUlkIjoiNDU2NDk3MjgxODUxOTYwMjQ0NyIsImNvbnRlbnRJbmZvMSI6eyJhdWRpb1ZpZGVvVHlwZSI6IlJhd0FkYXB0aXZlIiwicmF3QWRhcHRpdmVEZWZpbml0aW9uIjoxMCwiaW1hZ2VTcHJpdGVEZWZpbml0aW9uIjoxMH0sImN1cnJlbnRUaW1lU3RhbXAiOjE2NjMwNjQyNzYsImV4cGlyZVRpbWVTdGFtcCI6MTY2MzI5NDIxMCwidXJsQWNjZXNzSW5mbyI6eyJ0IjoiNjMyM2U2YjAiLCJybGltaXQiOjMsInVzIjoiNzJkNGNkMTEwMSJ9fQ.QFcBX9830ysTzJIyZxoOlRmNb2Gqy2fns9yOfriaDI8` 。

サンプルコード

VODはPython、Java、Go、C#、PHP、Node.jsなどの複数の言語用にプレーヤーのサンプルコードを提供しています。詳細についてはプレーヤー署名 - 署名の例をご参照ください。

よくあるエラー

プレーヤー署名を使用した際、Player SDKがエラーコードを返した場合、よくある原因は次のとおりです。

署名の計算 KEYが間違っている。 デフォルト配信設定内の `再生キー` を使用する必要があります。KEYリンク不正アクセス防止パラメータ内の `KEY` パラメータを誤って使用していないかどうかをチェックすることができます。

署名パラメータの**入力が間違っている**。例えば次のようなものがあります。

パラメータタイプのエラー：例えばappIdが整数で、誤って `appId:"125000123"` （文字列型）と入力している。または `contentInfo` 内のトランスコードテンプレートパラメータが整数で、誤って `transcodeDefinition: "14011"` （文字列型）と入力している。
パラメータ値が有効範囲を超えている：例えば `contentInfo` 内の再生するオーディオビデオタイプのパラメータを、誤って `audioVideoType: "Transocde"` （スペルミス、有効な列挙値ではない）と入力している。

プレーヤーの署名例

最終更新日：：2023-10-26 17:39:31

Python署名の例

pyjwt ライブラリを使用して署名を計算します。 `pip install pyjwt` を使用してインストールしてください。

```
#!/usr/bin/python
#coding=utf-8

import jwt

AppId = 1255566655
FileId = "4564972818519602447"
AudioVideoType = "RawAdaptive"
RawAdaptiveDefinition = 10
ImageSpriteDefinition = 10
CurrentTime = 1546340400
PsignExpire = 1546344000
UrlTimeExpire = "5c2b5640"
PlayKey = "TxtyhLlgo7J3iOADIron"

Original = {
"appId": AppId,
"fileId": FileId,
"contentInfo": {
"audioVideoType": AudioVideoType,
"rawAdaptiveDefinition": RawAdaptiveDefinition,
"imageSpriteDefinition": ImageSpriteDefinition
},
"currentTimeStamp": CurrentTime,
"expireTimeStamp": PsignExpire,
"urlAccessInfo": {
"t": UrlTimeExpire
}
}

Signature = jwt.encode(Original, PlayKey, algorithm='HS256')

print("Original: ", Original)
print("Signature: ", Signature)
```

Java署名の例

java-jwt ライブラリを使用して署名を計算します。

```
import java.util.*;
import com.auth0.jwt.algorithms.Algorithm;
import com.auth0.jwt.exceptions.JWTCreationException;
import com.auth0.jwt.JWT;

class Main {
public static void main(String[] args) {
Integer AppId = 1255566655;
String FileId = "4564972818519602447";
String AudioVideoType = "RawAdaptive";
Integer RawAdaptiveDefinition = 10;
Integer ImageSpriteDefinition = 10;
Integer CurrentTime = 1589448067;
Integer PsignExpire = 1589548067;
String UrlTimeExpire = "5ebe9423";
String PlayKey = "TxtyhLlgo7J3iOADIron";
HashMap<String, Object> urlAccessInfo = new HashMap<String, Object>();
urlAccessInfo.put("t", UrlTimeExpire);
HashMap<String, Object> contentInfo = new HashMap<String, Object>();
contentInfo.put("audioVideoType", AudioVideoType);
contentInfo.put("rawAdaptiveDefinition", RawAdaptiveDefinition);
contentInfo.put("imageSpriteDefinition", ImageSpriteDefinition);

try {
Algorithm algorithm = Algorithm.HMAC256(PlayKey);
String token = JWT.create().withClaim("appId", AppId).withClaim("fileId", FileId)
.withClaim("contentInfo", contentInfo)
.withClaim("currentTimeStamp", CurrentTime).withClaim("expireTimeStamp", PsignExp
ire)
.withClaim("urlAccessInfo", urlAccessInfo).sign(algorithm);
System.out.println("token:" + token);
} catch (JWTCreationException exception) {
// Invalid Signing configuration / Couldn't convert Claims.
}
}
}
```

Go署名の例

jwt-go ライブラリを使用して署名を計算します。コマンドgo get github.com/dgrijalva/jwt-go`を使用してインストールしてください。

```
package main

import (
"fmt"
"time"
"strconv"
"github.com/dgrijalva/jwt-go"
)

func main(){
appId := 1255566655 // ユーザー appid
fileId := "4564972818519602447" // ターゲット FileId
audioVideoType := "RawAdaptive" // 再生するオーディオビデオのタイプ
rawAdaptiveDefinition := 10 // 出力が許可される、暗号化されていないABSテンプレートID
imageSpriteDefinition := 10 // プログレスバープレビュー用のスプライトイメージテンプレートID
currentTime := time.Now().Unix()
psignExpire := currentTime + 3600 // 有効期限は、1hなど任意に設定できます
urlTimeExpire := strconv.FormatInt(psignExpire, 16) // 有効期限は、1hなどの16進文字列で任意に設定できます
playKey := []byte("TxtyhLlgo7J3iOADIron")

// Create a new token object, specifying signing method and the claims
// you would like it to contain.
token := jwt.NewWithClaims(jwt.SigningMethodHS256, jwt.MapClaims{
"appId": appId,
"fileId": fileId,
"contentInfo": {
"audioVideoType": audioVideoType,
"rawAdaptiveDefinition": rawAdaptiveDefinition,
"imageSpriteDefinition": imageSpriteDefinition,
},
"currentTimeStamp": currentTime,
"expireTimeStamp": psignExpire,
"urlAccessInfo": map[string]string{
"t": urlTimeExpire,
},
})

// Sign and get the complete encoded token as a string using the secret
tokenString, err := token.SignedString(playKey)
```

```
fmt.Println(tokenString, err)
}
```

C# 署名の例

jose-jwt を使用して署名を計算します。NuGetコマンド `Install-Package jose-jwt` を使用してインストールしてください。

```
using System;
using System.Text;
using System.Collections.Generic;
using Jose;

public class Program
{
public static void Main()
{
var appId = 1255566655; // ユーザー appid
var fileId = "4564972818519602447"; // ターゲット FileId
var audioVideoType = "RawAdaptive"; // 再生するオーディオビデオのタイプ
var rawAdaptiveDefinition = 10; // 出力が許可される、暗号化されていないABSテンプレートID
var imageSpriteDefinition = 10; // プログレスバープレビュー用のスプライトイメージテンプレートID
var currentTime = DateTimeOffset.UtcNow.ToUnixTimeSeconds();
var psignExpire = currentTime + 3600; // 有効期限は、1hなど任意に設定できます
var urlTimeExpire = psignExpire.ToString("X4"); // 有効期限は、1hなどの16進文字列で任意に設定できます
var playKey = "TxtyhLlgo7J3iOADIron";
var playKeyBytes = Encoding.ASCII.GetBytes(playKey);
var payload = new Dictionary<string, object>()
{
{"appId", appId},
{"fileId", fileId},
{"contentInfo": new Dictionary<string, object>()
{
{"audioVideoType": audioVideoType},
{"rawAdaptiveDefinition": rawAdaptiveDefinition},
{"imageSpriteDefinition": imageSpriteDefinition}
}
},
{"currentTimeStamp", currentTime},
{"expireTimeStamp", psignExpire},
{"urlAccessInfo", new Dictionary<string, object>()
{
```

```
{"t", urlTimeExpire}
}
}
};
string token = Jose.JWT.Encode(payload, playKeyBytes, JwsAlgorithm.HS256);
Console.WriteLine(token);
}
}
```

PHP署名の例

php-jwt を使用して署名を計算します。コマンド `composer require firebase/php-jwt` を使用してインストールしてください。

```
<?php
require 'vendor/autoload.php';
use \Firebase\JWT\JWT;

$appId = 1255566655; // ユーザー appid
$fileId = "4564972818519602447"; // ターゲット FileId
$audioVideoType = "RawAdaptive"; // 再生するオーディオビデオのタイプ
$rawAdaptiveDefinition = 10; // 出力が許可される、暗号化されていないABSテンプレートID
$imageSpriteDefinition = 10; // プログレスバープレビュー用のスプライトイメージテンプレート
ID
$currentTime = time();
$psignExpire = $currentTime + 3600; // 有効期限は、1hなど任意に設定できます
$urlTimeExpire = dechex($psignExpire); // 有効期限は、1hなどの16進文字列で任意に設定でき
ます
$playKey = "TxtyhLlgo7J3iOADIron";

$payload = array(
"appId" => $appId,
"fileId" => $fileId,
"contentInfo" => array(
"audioVideoType"=> $audioVideoType,
"rawAdaptiveDefinition"=> $rawAdaptiveDefinition,
"imageSpriteDefinition"=> $imageSpriteDefinition
),
"currentTimeStamp" => $currentTime,
"expireTimeStamp" => $psignExpire,
"urlAccessInfo" => array(
"t" => $urlTimeExpire
)
);
```

```
$jwt = JWT::encode($payload, $playKey, 'HS256');
print_r($jwt);
?>
```

Node.js署名の例

jsonwebtoken を使用して署名を計算します。コマンド `npm install jsonwebtoken` を使用してインストールしてください。

```
var jwt = require('jsonwebtoken');

var appId = 1255566655 // ユーザー appid
var fileId = "4564972818519602447" // ターゲット FileId
var audioVideoType = "RawAdaptive" // 再生するオーディオビデオのタイプ
var rawAdaptiveDefinition = 10 // 出力が許可される、暗号化されていないABSテンプレートID
var imageSpriteDefinition = 10 // プログレスバープレビュー用のスプライトイメージテンプレートID
var currentTime = Math.floor(Date.now()/1000)
var psignExpire = currentTime + 3600 // 有効期限は、1hなど任意に設定できます
var urlTimeExpire = psignExpire.toString(16) // 有効期限は、1hなどの16進文字列で任意に設定できます
var playKey = 'TxtyhLlgo7J3iOADIron'

var payload = {
appId: appId,
fileId: fileId,
contentInfo: {
audioVideoType: audioVideoType,
rawAdaptiveDefinition: rawAdaptiveDefinition,
imageSpriteDefinition: imageSpriteDefinition
},
currentTimeStamp: currentTime,
expireTimeStamp: psignExpire,
urlAccessInfo: {
t: urlTimeExpire
}
}
var token = jwt.sign(payload, playKey);
console.log(token);
```

ホットリンク防止の設定
ホットリンク防止の概要

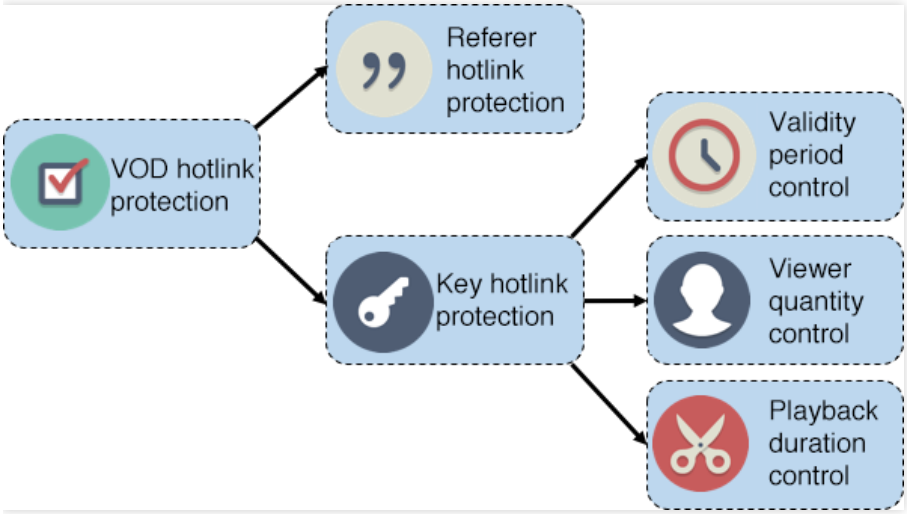
最終更新日：：2023-10-26 17:39:31

概要

ビデオ再生の権限制御をサポートするため、VODはホットリンク防止のソリューションを提供します。ホットリンク防止を有効にした後、Tencent Cloud CDNノードは、再生リクエストの重要な情報を検査し、検査に合格したリクエストにビデオデータを返します。このソリューションにはプレーヤーに対する制限はなく、VODのプレーヤーSDKであるか一般的なプレーヤーであるかを問わず、いずれにも使用できます。

タイプと機能

VODホットリンク防止はRefererホットリンク防止およびKeyホットリンク防止をサポートします。



Refererホットリンク防止

HTTPに基づくRefererメカニズムは、再生リクエストHeaderのRefererフィールドを介してリクエストのソースを識別します。指定したドメイン名をブラックリストまたはホワイトリストに追加することができ、これに基づいてCDNノードは認証を行い、再生リクエストを許可または拒否します。

Keyホットリンク防止

ビデオの再生制御パラメータをQueryStringの形式でビデオURLにスプライスできるようにし、CDNノードはURLの再生制御パラメータを検査し、そのパラメータに応じてビデオの再生を制御します。現在、Keyホットリンク防

止は、「有効期限パラメータ」、「再生可能なIP数パラメータ」 と「プレビュー時間パラメータ」を介して「ホットリンク防止有効期限の制御」、「ホットリンク防止再生視聴者数の制御」と「ビデオ再生時間の制御」をサポートします。

ホットリンク防止有効期限の制御

ビデオURLで有効期限を指定します。リクエストされたビデオURLがすでに期限切れである場合は、ビデオを再生することができません。この方式を介して、ビデオURLに有効期限を設定し、第三者がビデオURLを他のWebサイトに転送して長期的に使用することを防止できます。

ホットリンク防止再生視聴者数の制御

ビデオURLにアクセスできる最大視聴者数を指定します。異なるプライベートネットワークの再生端末では、通常、パブリックネットワークIPは異なります。URLを再生できるパブリックIPの最大数を制限することで、同一のURLにアクセスできる視聴者数を制限できます。こうすることで、第三者がビデオURLを他のWebサイトに転送し、視聴人数の制限なく配布することを防止できます。

ビデオ再生許可時間の制御

ビデオURLでプレビュー時間を指定します（ビデオの最初の5分間のみ再生を許可するなど）。この方式を介して無料ユーザー向けのプレビュー機能を実現することができます。

説明：

- Refererホットリンク防止の詳細については、Refererホットリンク防止をご参照ください。
- Keyホットリンク防止の詳細については、Keyホットリンク防止をご参照ください。

Refererホットリンク防止

最終更新日：：2023-10-26 17:39:31

機能の説明

- HTTPプロトコルでサポートされるRefererメカニズムに基づき、HTTPヘッダーのRefererフィールドを介してリクエストのソースを識別します。Refererブラックリスト/ホワイトリストを設定して、ビデオをリクエストするソースを識別し認証します。
- ブラックリストとホワイトリストの両モードをサポートします。ビデオ再生リクエストがCDNノードに伝達されると、ノードは設定されたRefererブラックリスト/ホワイトリストに基づき、リクエストのソースを認証します。ルールに適合するリクエストについては、CDNはビデオデータを返し、適合しない場合は、403レスポンスコードを返し、再生リクエストを拒否します。

> 説明：
> Refererホットリンク防止の有効化については、 ホットリンク防止の設定 をご参照ください。

注意事項

- この機能はオプションであり、デフォルトでは有効ではありません。
- 機能を有効にした後、ブラックリストまたはホワイトリストを選択して設定します。ブラックリストとホワイトリストは相互に排他的であるため、同時に1つのモードしかサポートできません。
- ブラックリストまたはホワイトリストのドメイン名は1個～10個をサポートし、1行につき1個を記録します。
- ドメイン名の前にプロトコル名（ `http://` と `https://` ）を追加しないでください。ドメイン名の照合はプレフィックスに基づいています（ `abc.com` と入力すると、 `abc.com/123` と `abc.com.cn` も一致します）。また、ワイルドカード `（*.abc.com）` をサポートします。

Keyホットリンク防止

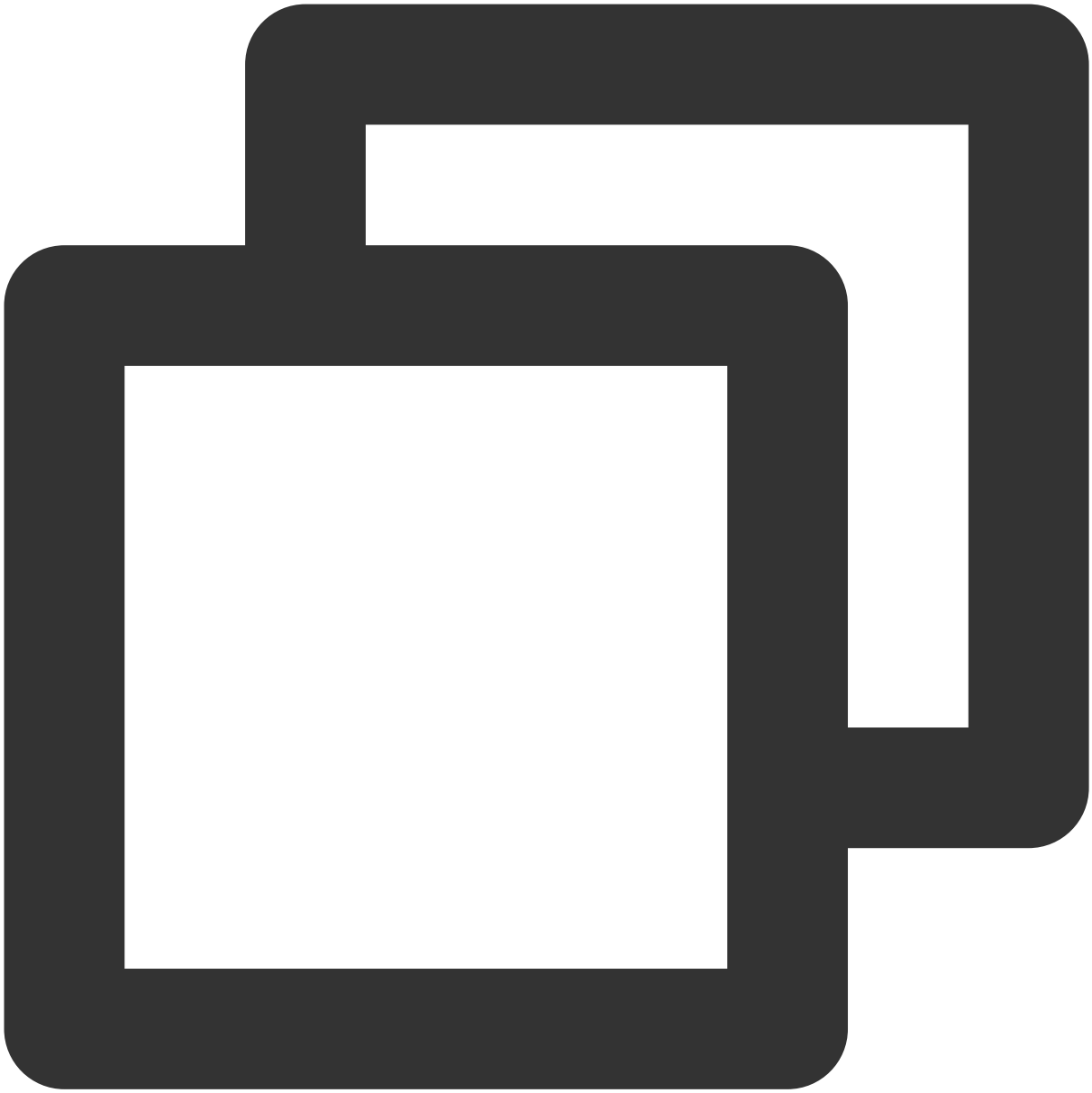
最終更新日：：2023-10-26 17:39:31

機能の説明

第三者がビデオURLを取得して、長期的に使用することができないように、ビデオURLの有効期限を指定できます。
第三者がビデオURLを取得して視聴人数の制限なく配布することができないように、ビデオURLで再生可能な最大IP数を指定できます。
ビデオURLでのプレビュー時間の指定をサポートし、プレビュー機能を実現します。
ビデオURLでのリージョン指定によるアクセス制限をサポートします。ブラックリストとホワイトリストの2つの方式をサポートしています。
ビデオURLでのRefererブラックリスト/ホワイトリストの指定をサポートしています。
`KEY` を使用してビデオURLに署名し、URLに署名結果を含めることができます。ユーザーキーを開示しない限り、他のユーザーはビデオURLを偽造することができません。
CDNノードは、ビデオURLのパラメータと署名を検査し、ビデオ再生リクエストを制御します。リクエストが検査に不合格となった場合は、403レスポンスコードが返されます。
サポートするファイルタイプ：MP4、TS、M3U8、FLV、AAC、MOV、WMV、AVI、MP3、RMVB、MKV、MPG、3GP、WEBM、M4V、ASF、F4V、WAV、MPEG、VOB、RM、WMA、DAT、M4A、MPD、M4S。
説明：
Keyリンク不正アクセス防止の有効化については、 リンク不正アクセス防止の設定をご参照ください。
現時点ではリンク不正アクセス防止Keyのプレビュー機能はオーディオ形式のファイルをサポートしていません。

ホットリンク防止URLの生成方法

VODでのビデオにはすべて**ビデオオリジナルURL**が存在します。リンク不正アクセス防止が有効になっていない場合は、ビデオオリジナルURLを使用して、ビデオを再生することができます。
Keyリンク不正アクセス防止を有効にすると、ビデオオリジナルURLは再生できなくなり、この場合、ビデオの**リンク不正アクセス防止URL**を生成する必要があります。
ホットリンク防止URLの生成ルールは、オリジナルURLの末尾に、QueryStringの形式でホットリンク防止パラメータを追加することです。形式は次のとおりです。



```
http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4?t=[t]&exper=[exper]&rlimit=[
```

リンク不正アクセス防止URLの各パラメータの定義と値の取得方法について、以下で詳細に説明します。

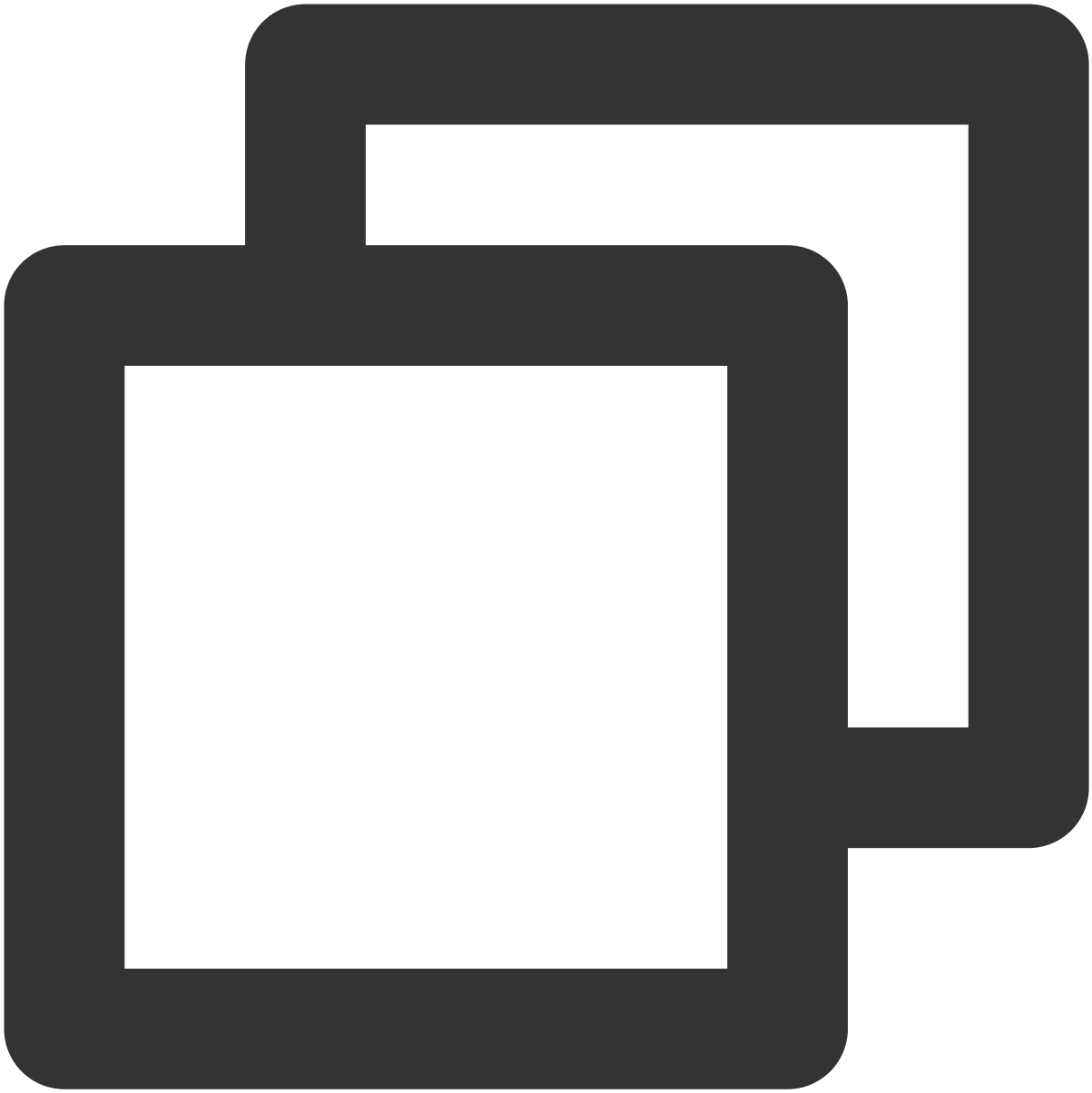
ホットリンク防止パラメータ

パラメータ名	記入必須	説明
KEY	はい	Keyリンク不正アクセス防止を有効にする場合に入力するキーです。アルファベットの

		大文字小文字（a～Z）または数字（0～9）で構成され、長さは8～20文字とする必要があります。コンソールで【KEY生成】をクリックして生成することを推奨します。具体的な操作手順については、 リンク不正アクセス防止の設定 をご参照ください。
Dir	はい	ビデオオリジナルURLのPATHのファイル名を削除した部分のパスです。オリジナルURLがhttp://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4であれば、再生パスは/dir1/dir2/となります。
t	はい	再生アドレスの有効期限タイムスタンプです。Unix時間の16進数の小文字形式で表示します。 有効期限が過ぎるとURLは失効し、403レスポンスコードが返されます。マシンの間に時間差が存在する可能性を考慮し、リンク不正アクセス防止URLの実際の有効期限は通常、指定した有効期限より5分長くなります。つまり、300秒の許容誤差時間が追加されます。 有効期限タイムスタンプの時間を短くしすぎず、ビデオを完全に再生できる充分な時間を確保することを推奨します。
exper	いいえ	プレビュー時間。単位は秒で、10進数で表示されます。空のままにするか、または0を入力するとプレビューは無効になります（つまり、完全なビデオが返されます）。 プレビュー時間はビデオのオリジナルの時間を超えないようにしてください。超えてしまうと、再生に失敗する可能性があります。
rlimit	いいえ	再生可能な最大端末IP数。10進数で表示され、最大値は9です。空のままにした場合、制限はありません。 URLを1人しか再生できないように制限する場合は、rlimitを厳密に1に制限しないことを推奨します（3に設定するなど）。モバイル端末ではネットワーク切断後の再接続の際にIPが変わる可能性があるためです。
us	いいえ	リンクID。リンクの一意性を高めるため、リンク不正アクセス防止URLのランダム化に使用されます。 リンク不正アクセス防止URLを生成するときは毎回 、ランダムなus値を指定することを推奨します。
whreg	いいえ	アクセスを許可するリージョンのリスト。1～10個とすることができ、半角のコンマで区切ります。値はISO 3166-1国名コードを用います。
bkreg	いいえ	アクセスを禁止するリージョンのリスト。1～10個とすることができ、半角のコンマで区切ります。値はISO 3166-1国名コードを用います。
whref	いいえ	アクセスを許可するドメイン名のリスト。1～10個とすることができ、半角のコンマで区切ります。ドメイン名の前にプロトコル名（http://とhttps://）を追加しないでください。ドメイン名はプレフィックスマッチング（abc.comと入力すると、abc.com/123とabc.com.cnも一致します）であり、ワイルドカード（*.abc.comなど）もサポートします。
bkref	いいえ	アクセスを禁止するドメイン名のリスト。1～10個とすることができ、半角のコンマで区切ります。ドメイン名の前にプロトコル名（http://とhttps://）を追加しないでくださ

		い。ドメイン名はプレフィックスマッチング（abc.comと入力すると、abc.com/123とabc.com.cnも一致します）であり、ワイルドカード（*.abc.comなど）もサポートします。
sign	はい	リンク不正アクセス防止署名。32文字の長さの16進数で表示され、リンク不正アクセス防止URLの有効性の検証に使用されます 署名の検証に失敗すると403レスポンスコードが返されます。署名の計算式について以下でご説明します。
uv	いいえ	6桁の16進数の文字。トレーサビリティウォーターマークのシーンで使用します。

署名の計算式



```
sign = md5(KEY + Dir + t + exper + rlimit + us + whref + bkref + whreg + bkreg + uv
```

式の `+` は文字列のスプライスを意味し、オプションのパラメータを空白の文字列にすることができます。

ホットリンク防止URLの生成例

VODにビデオがあり、ビデオのオリジナル再生URL
が `http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4` で、Keyリンク不正アクセス防止を

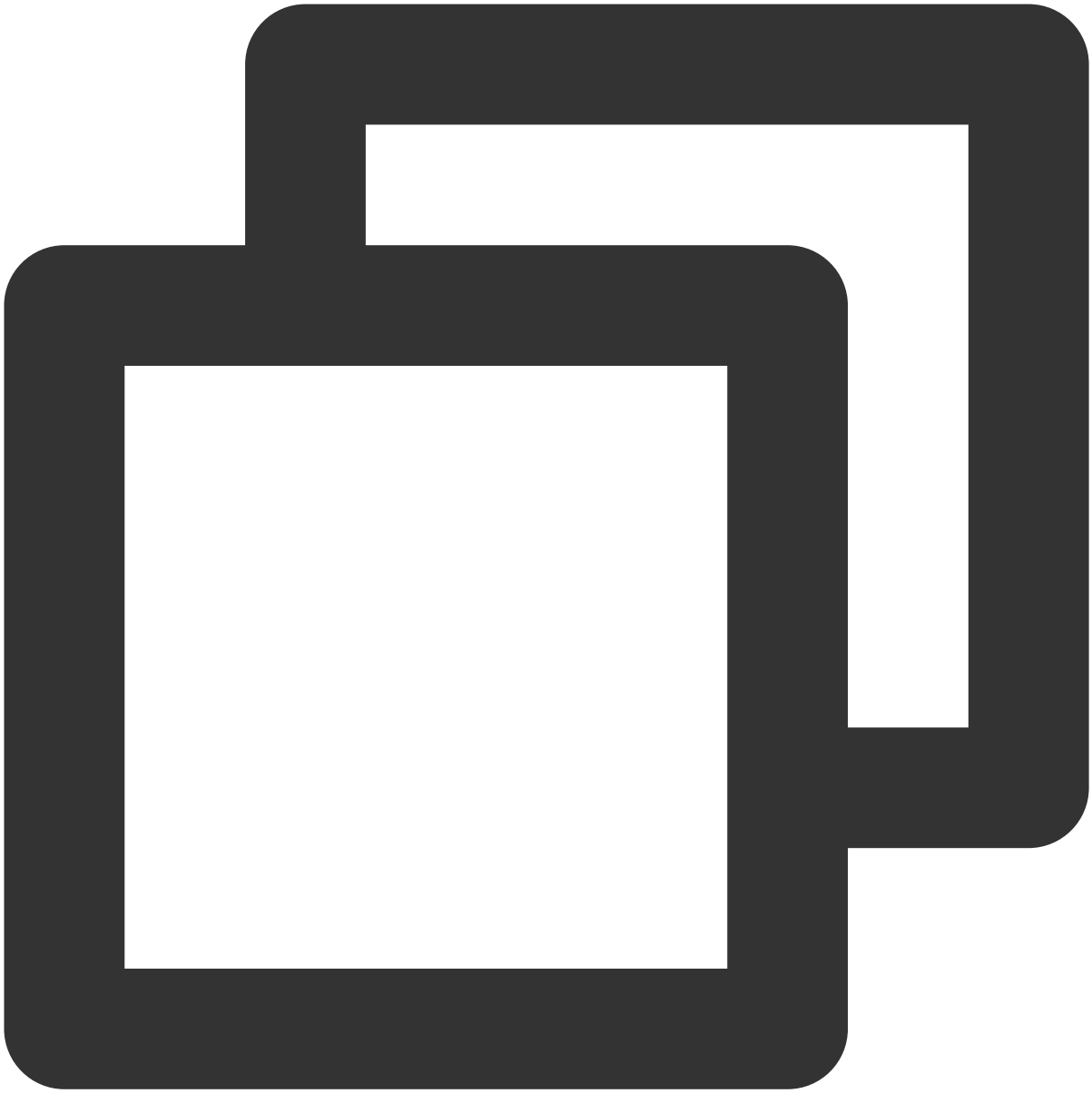
有効にし、生成されたキーが `24FEQmTzro4V5u3D5epW` 、生成されたランダム文字列が `72d4cd1101` であり、次の条件を満たす必要がある場合。
1. このビデオのためにリンク不正アクセス防止URLを生成し、URLの有効期限を2018年1月31日20:00（Unix時間では1517400000）とします。
2. プレビューURLを生成し、プレビュー時間をビデオの最初の5分間（オリジナルのビデオの時間は5分より長いこと）とします。
3. URLで再生可能なIP数を最大で3つの異なるIPに制限し、端末でこのURLの再生を許可します。
「ビデオ再生アドレス有効期限の制御」、「ビデオ再生アドレスでの最大再生許可IP数」と「ビデオ再生許可時間の制御」について、それぞれのホットリンク防止URLの生成方法を、以下に説明します。

例1：再生アドレス有効期限の制御

手順1：ホットリンク防止パラメータの確定

パラメータ名	値	説明
KEY	24FEQmTzro4V5u3D5epW	開発者がKeyリンク不正アクセス防止をアクティブにするときに選択したキー
Dir	/dir1/dir2/	元の再生URLのPATHからmyVideo.mp4を除いた残りの部分
t	5a71afc0	有効期限タイムスタンプ1517400000を16進数で表した結果
us	72d4cd1101	生成されたランダムな文字列

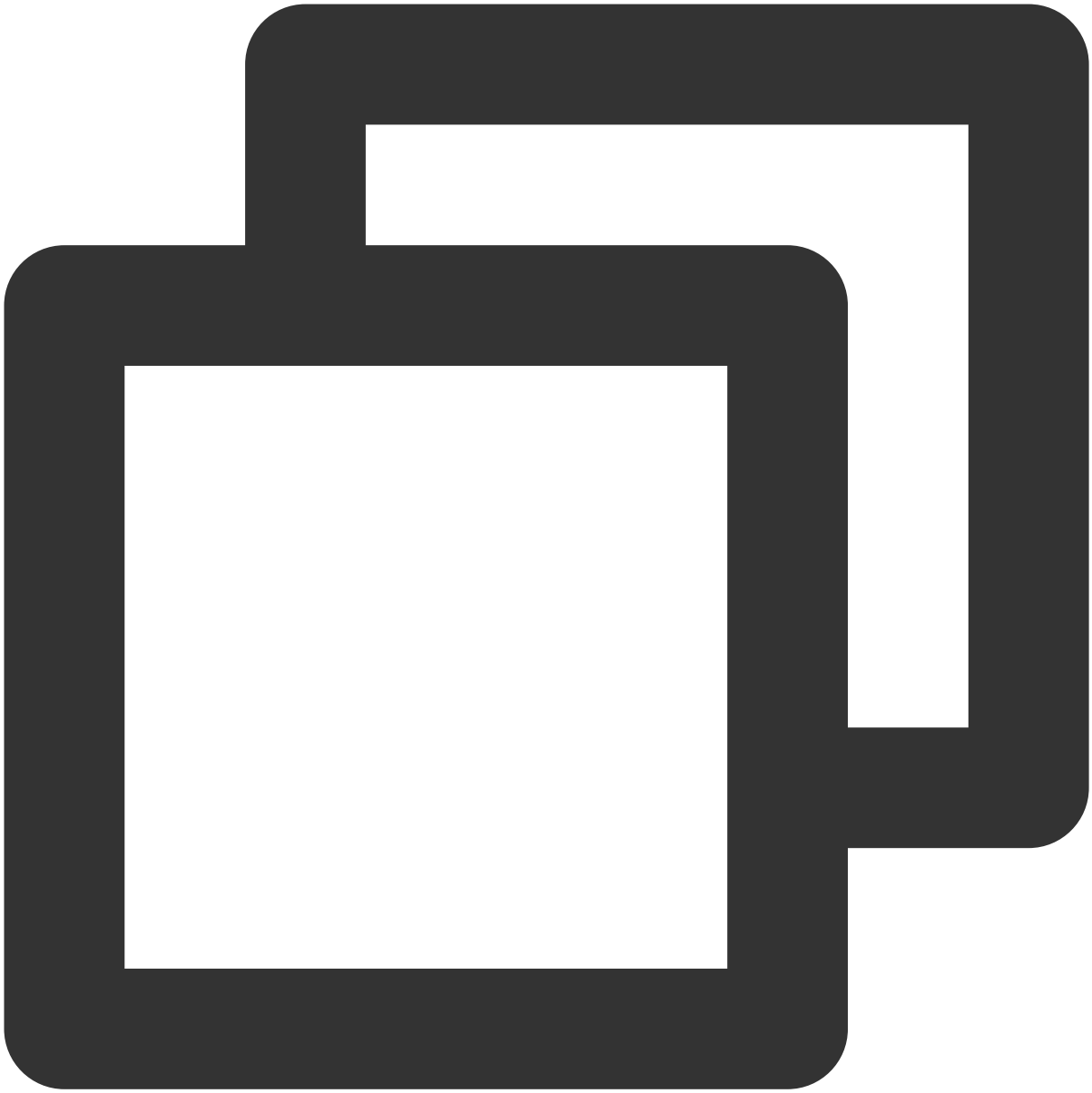
手順2：署名の計算



```
sign = md5("24FEQmTzro4V5u3D5epW/dir1/dir2/5a71afc072d4cd1101") = "3d8488faeb37d52d
```

手順3：ホットリンク防止URLの生成

ホットリンク防止パラメータをビデオオリジナルURLのQueryStringにスプライスし、ビデオホットリンク防止URLを取得します。



```
http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4?t=5a71afc0&us=72d4cd1101&sig
```

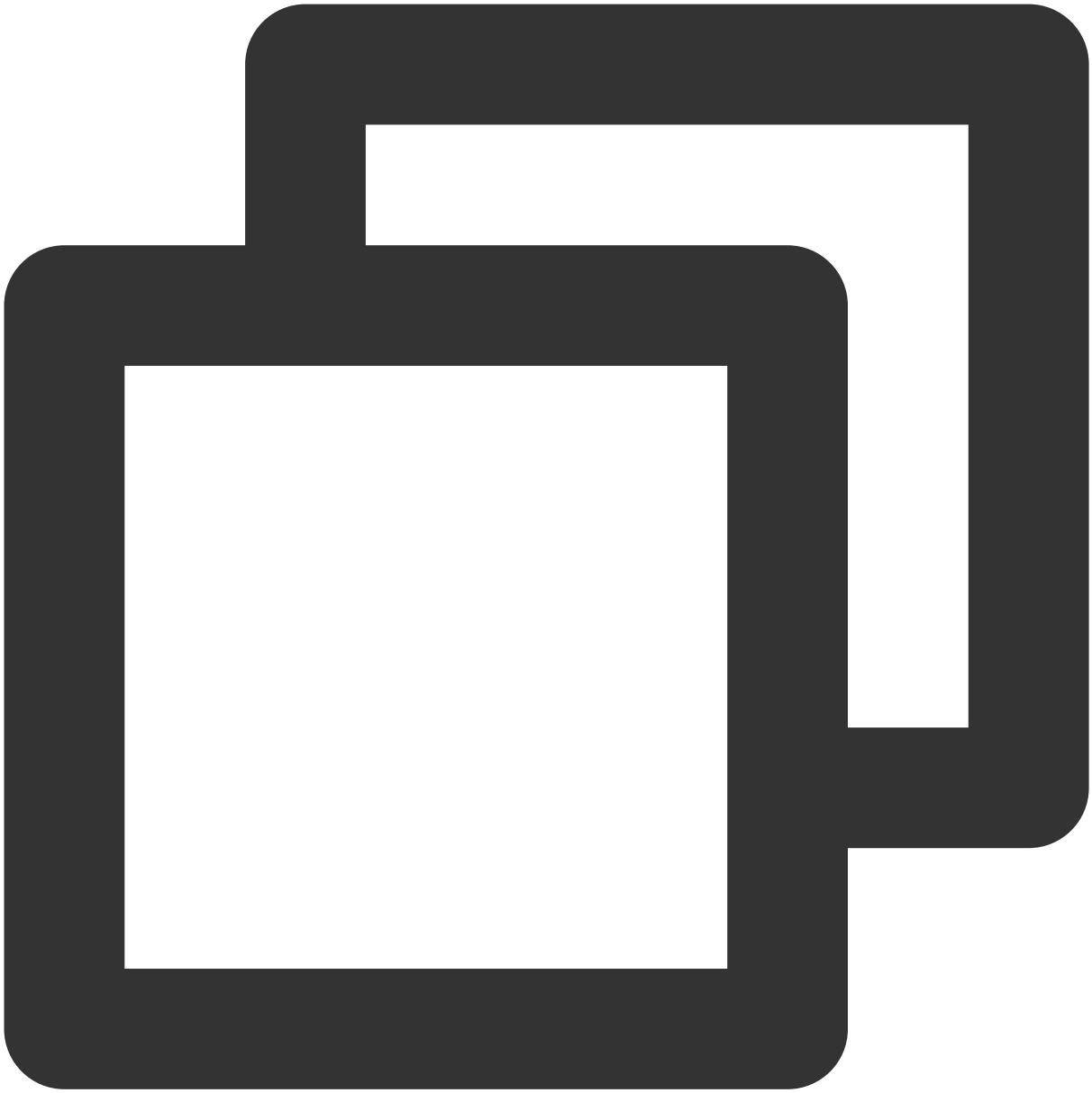
例2：再生アドレスでの最大再生可能IP数

手順1：ホットリンク防止パラメータの確定

パラメータ名	値	説明
`KEY`	`24FEQmTzro4V5u3D5epW`	開発者がKeyリンク不正アクセス防止をアクティブにするときに選択したキー

`Dir`	`/dir1/dir2/`	元の再生URLのPATHから `myVideo.mp4` を除いた残りの部分
`t`	`5a71afc0`	有効期限タイムスタンプ1517400000を16進数で表した結果
`rlimit`	`3`	URLでの再生を最大3つの異なるIPで許可
`us`	`72d4cd1101`	生成されたランダムな文字列

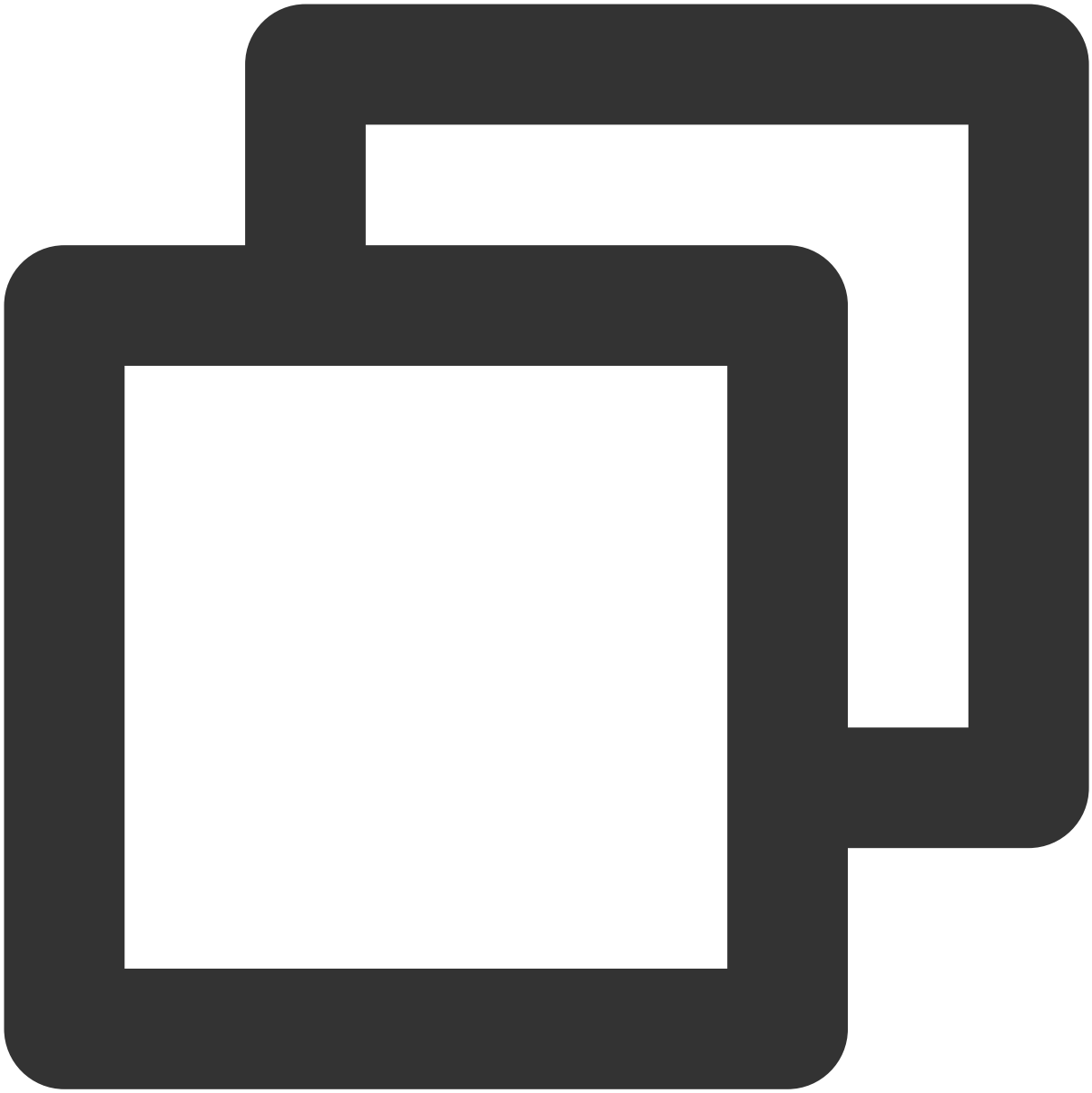
手順2：署名の計算



```
sign = md5("24FEQmTzro4V5u3D5epW/dir1/dir2/5a71afc0372d4cd1101") = "c5214f0d5961b13
```

手順3：ホットリンク防止URLの生成

ホットリンク防止パラメータをビデオオリジナルURLのQueryStringにスプライスし、ビデオホットリンク防止URLを取得します。



```
http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4?t=5a71afc0&rlimit=3&us=72d4c
```

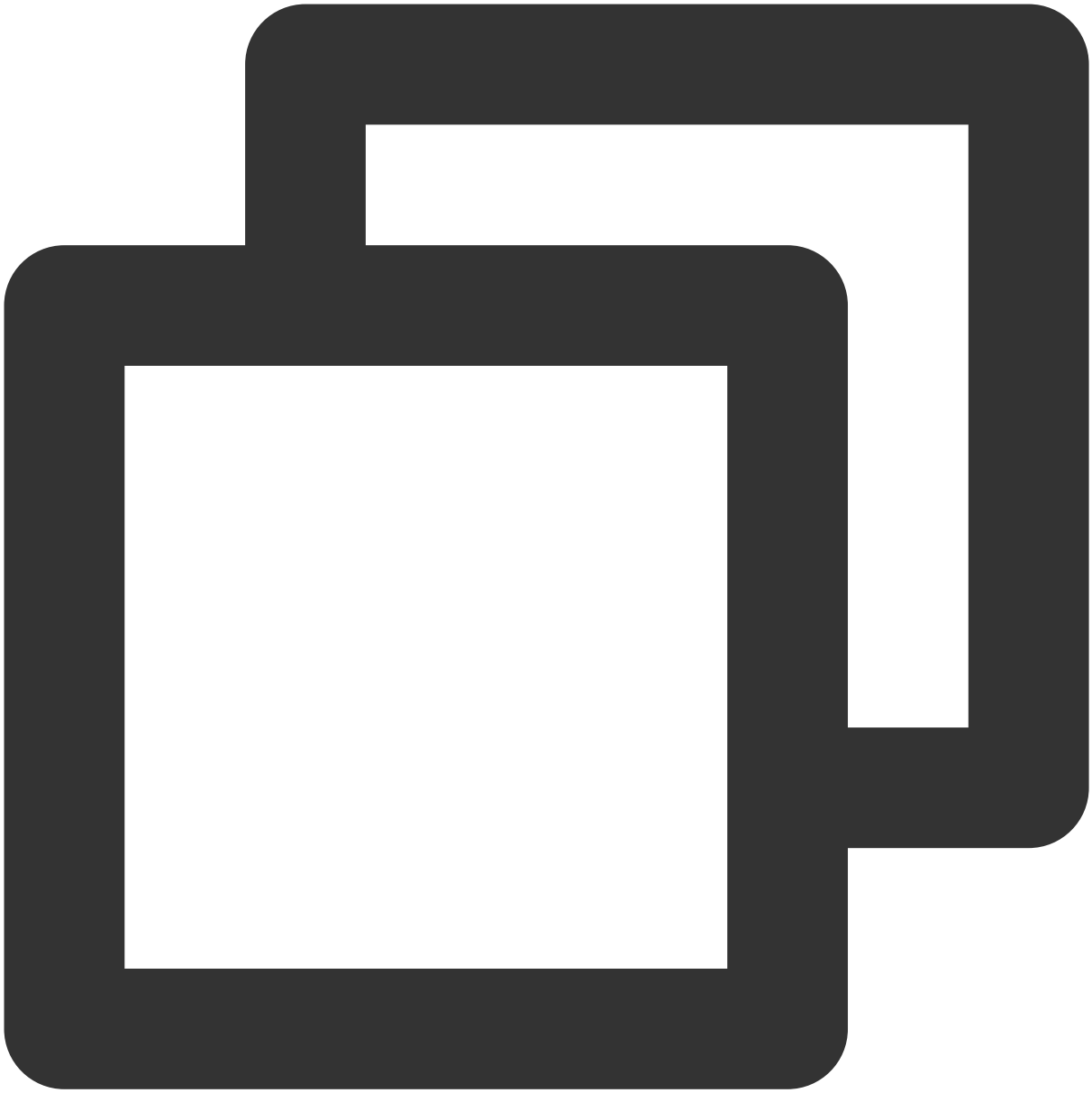
例3：再生許可時間の制御

手順1：ホットリンク防止パラメータの確定

パラメータ名	値	説明
KEY	24FEQmTzro4V5u3D5epW	開発者がKeyリンク不正アクセス防止をアクティブに

		するときに選択したキー
Dir	/dir1/dir2/	元の再生URLのPATHからmyVideo.mp4を除いた残りの部分
t	5a71afc0	有効期限タイムスタンプ1517400000を16進数で表した結果
exper	300	最初の5分、すなわち300秒をプレビュー
us	72d4cd1101	生成されたランダムな文字列

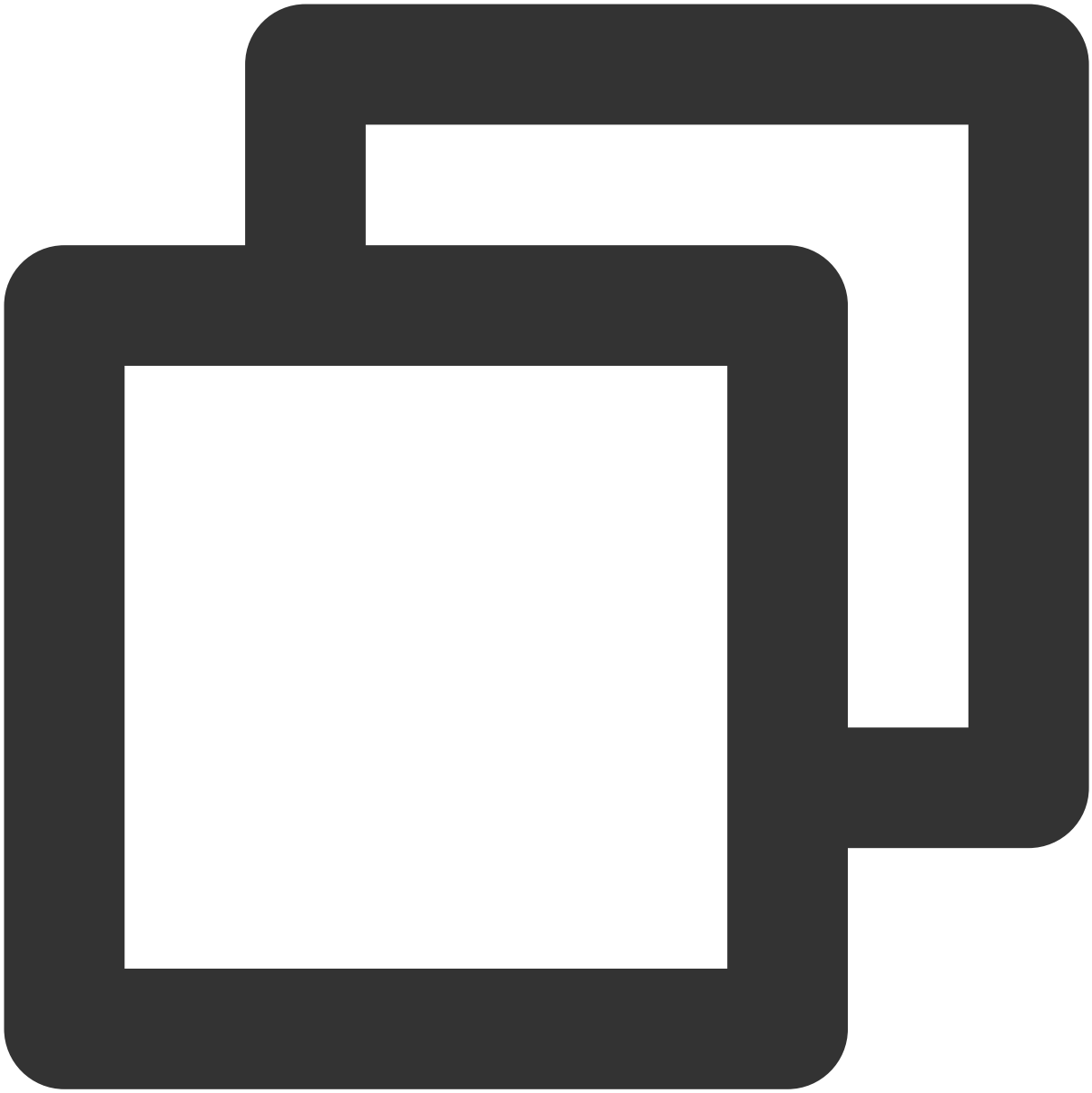
手順2：署名の計算



```
sign = md5("24FEQmTzro4V5u3D5epW/dir1/dir2/5a71afc030072d4cd1101") = "547d98c4b91e8
```

手順3：ホットリンク防止URLの生成

ホットリンク防止パラメータをビデオオリジナルURLのQueryStringにスプライスし、ビデオホットリンク防止URLを取得します。



```
http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4?t=5a71afc0&exper=300&us=72d4
```

Keyホットリンク防止の生成と検証のツール

VODはKeyホットリンク防止URLの生成ツールと検証ツールを提供し、開発者はそれらツールを使用して迅速かつ正確に要件に適合するホットリンク防止URLを生成し検証することができます。

Keyリンク不正アクセス防止生成ツール

Keyリンク不正アクセス防止検証ツール

注意事項

この機能はオプションであり、デフォルトでは有効ではありません。
この機能を有効にした後は、ビデオオリジナルURLでは直接再生できなくなるため、ルールに従い、有効なリンク不正アクセス防止URLを生成する必要があります。
キー `KEY` は、大文字および小文字のアルファベット（a～Z）または数字（0～9）で構成され、長さは8～20文字とする必要があります。
リンク不正アクセス防止URLが期限切れである場合、または署名が合格しない場合は、ビデオを再生できず、403レスポンスコードが返されます。
リンク不正アクセス防止URLのQueryStringのパラメータは、 `t` 、 `exper` 、 `rlimit` 、 `us` 、 `sign` の順に出現する必要があり、順序が正しくない場合は、ビデオを再生することができません。
プレビュー機能を使用する場合、プレビュー時間はビデオ時間よりも短いことを確認する必要があり、ビデオ時間より長い場合は、ビデオを再生することができません。
プレビューではビデオの形式に対する厳格な要件（例えばH.264のみ。ビデオメタ情報はビデオファイルのヘッダーに含まれている必要がある等）があり、形式要件に適合しないオリジナルビデオをプレビューすると異常が発生するおそれがあります。プレビュー機能を設定する前に、VODトランスコード機能を使用して、トランスコードすることをお勧めします（トランスコード後の形式がすべてプレビュー形式の要件に適合するようにします）。

メディアファイルのダウンロード

最終更新日：：2023-10-26 17:46:07

VODはクラウドに保存したメディアファイルのローカルディスクまたはその他のストレージ上へのダウンロードをサポートしています。

メディアファイル

VODサービスを使用すると、ソースファイル、トランスコードファイル、ビデオスクリーンキャプチャ、カバー画像など、ダウンロード可能なさまざまなメディアファイルが生成される場合があります。VODでは各種メディアファイルのダウンロードをご提供しています。メディアファイルのカテゴリーは次のとおりです。

- オーディオビデオ
  - オーディオビデオソースファイル：VODにアップロードしたオーディオビデオソースファイル。
  - メディア処理タスクで生成されたオーディオビデオファイル：通常のトランスコードファイル、アダプティブビットレートストリーミングなど。
- 画像
  - 画像ソースファイル：VODにアップロードした画像ソースファイル。
  - メディア処理タスクで生成された画像ファイル：スクリーンキャプチャ、スプライトイメージ、アニメーションなど。

コンソールからのダウンロードアドレス取得

- VODコンソールにログインし、**メディア資産管理**に進み、**オーディオビデオ管理**または**画像管理**を選択し、対応するファイルの右側の**管理**をクリックすると、ソースファイルとメディア処理出力ファイルのダウンロードアドレスを取得できます。具体的には、オーディオビデオの確認および画像の管理をご参照ください。
- VODコンソールにログインし、**メディア資産管理>オーディオビデオ管理**と進み、**その他のバッチ操作**で、オーディオビデオソースファイルとメディア処理出力ファイルのダウンロードアドレスを一括エクスポートできます。具体的には、オーディオビデオのエクスポートをご参照ください。

APIによるダウンロードアドレス取得

VODではAPIによって対応するファイルダウンロードアドレスを取得することもできます。以下をご参照ください。

- メディア詳細情報の取得
- メディア情報検索

ダウンロードファイル名のカスタマイズ

通常、ブラウザを通じてメディアファイルURLにアクセスすると、ブラウザはこのファイルをダウンロードするのではなく、直接開こうとします。例えば、ブラウザでビデオファイルURLにアクセスすると、ブラウザはこのビデオの再生を直接開始します。URLによってファイルをダウンロードしたい場合は、QueryStringにパラメータ `download_name` を追加すると、ブラウザにこのファイルをダウンロードさせ、同時にダウンロードしたファイルの名前をカスタマイズすることができるようになります。例えば次のようにします。

```
http://example.vod2.myqcloud.com/dir1/dir2/myVideo.mp4?download_name=[download_name]
```

注意：

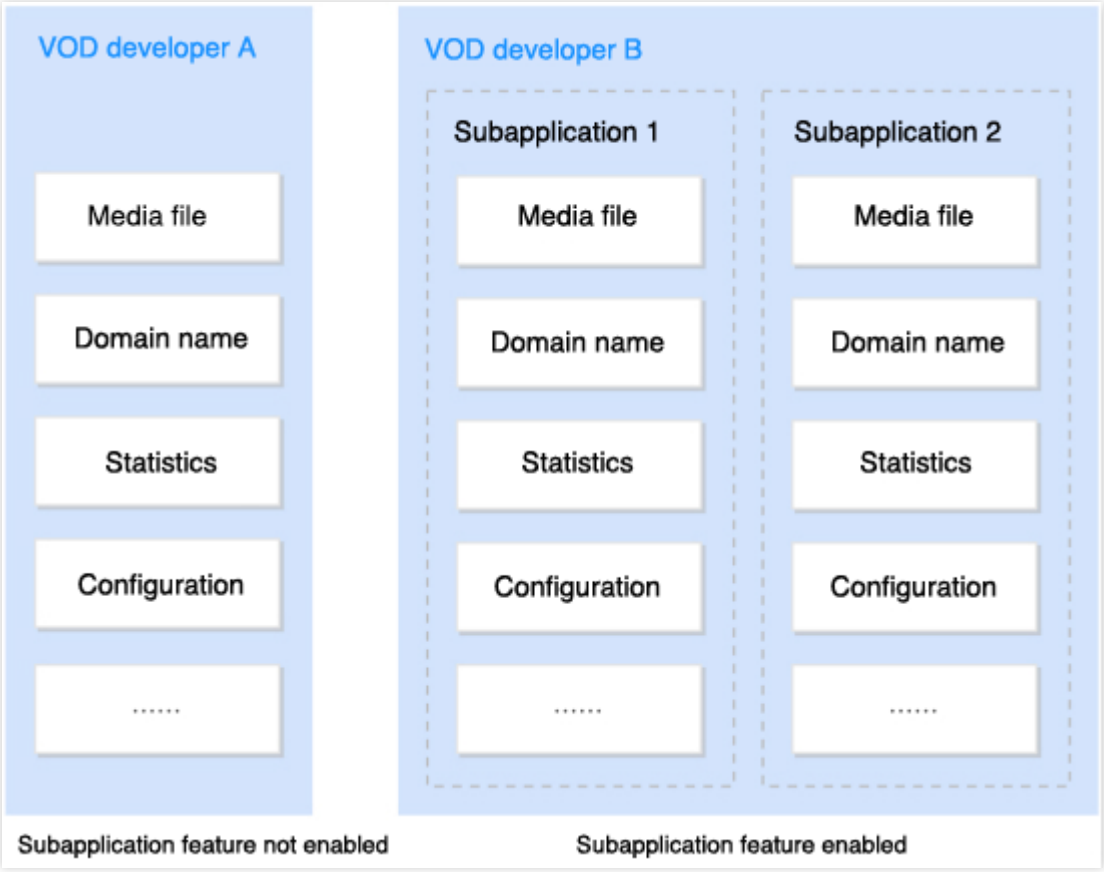
- リンク不正アクセス防止を設定している場合は、ダウンロードの際にRefererリストを受信します。URLの有効期限などの制限については、リンク不正アクセス防止の概要をご参照ください。
- ビデオ暗号化機能を使用後にダウンロードしたトランスコードファイルは暗号化ファイルとなり、再生するには復号が必要となります。暗号化したビデオの再生をご参照ください。
- HLSファイルをダウンロードする際は、インデックスファイルと各セグメントファイルをそれぞれダウンロードする必要があります。ビデオはトランスコード操作を行ってMP4形式に変換することができます。

サブアプリケーションシステム

最終更新日：：2023-10-26 17:46:07

概要

VODでは、VODでリソースを隔離できるように、**サブアプリケーション機能**を提供しています。サブアプリケーションとはVODの内的概念であり、リソースを分割するための方法です。サブアプリケーションの外的パフォーマンスは、独立したVODアカウントに似ています。サブアプリケーションを作成した後のVODリソースの帰属形式を、次の図に示します。



説明：

ここでいう**リソース**には、VOD内のメディアファイルとその属性、メディアファイルから派生したその他のファイル、各種設定、CDNドメイン名、VODサービスの使用状況の統計情報などが含まれます。

ユースケース

VODサブアプリケーションの一般的なユースケースは、次のとおりです。

複数部門/複数業務の隔離：ある企業はTencent Cloudをベースとして独自の製品を開発しています。その中で部門Aは、VODを使用してUGSV Appを開発する必要があります。部門Bは、VODを使用して映画とテレビのWebサイトを開発する必要があります。これら2つのVOD業務は互いに隔離する必要がありますが、財務管理上の理由か

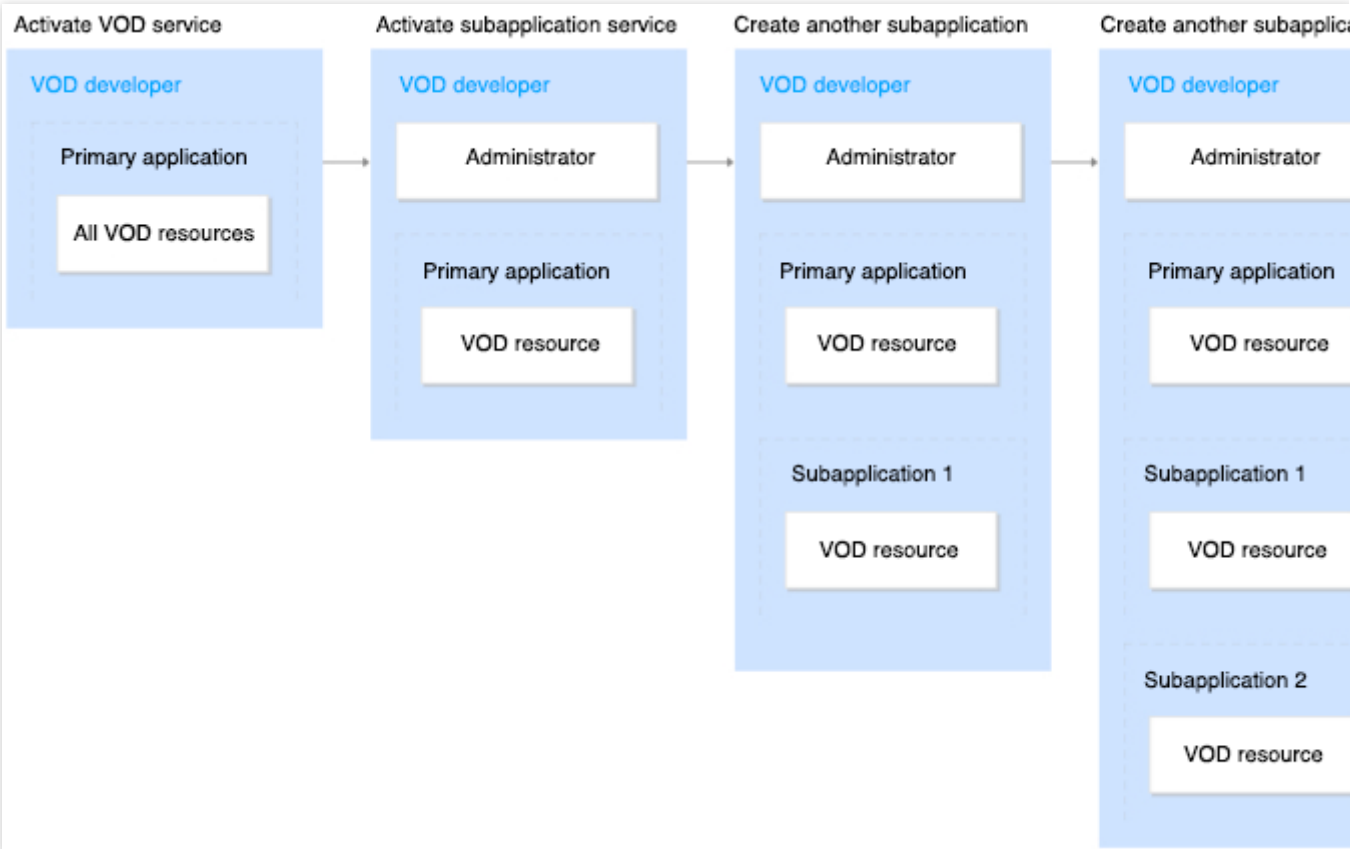
ら、この会社は部門AとBに個別のTencent Cloudアカウントを作成することができません。このような場合、VODのサブアプリケーション機能を使用して、部門Aと部門Bに1つずつサブアプリケーションを割り当てることができます。

権限制御：上記の複数部門/複数業務の隔離シナリオにおいて、開発者は、権限制御の要件をさらに設定する場合があります。例えば、各部門に対して自部門の業務に関連付けられたサブアプリケーションにのみアクセスを許可し、他のサブアプリケーションへのアクセス権限を持たせないといったことです。この場合、アカウント管理者は、部門Aと部門Bにサブユーザーを1つずつ割り当てて、対応するVODサブアプリケーションへのアクセス権限を付与することができます。操作の詳細については、CAMをご参照ください。

本番環境とテスト環境の分離：開発者はあるVOD機能をテストしたいと思っていますが、それらがオンライン運用に影響を与えることを懸念しています（例えば、イベントの通知方法の変更やリンク不正アクセス防止を有効にすることなど）。開発者は、本番環境とテスト環境に対して、それぞれ1つのサブアプリケーションをアクティブにすることで、新しい機能を最初にテスト環境で検証し、誤りのないことを確認してから、オンライン環境に変更することができます。

ロールの定義と識別

サブアプリケーションシステムには、管理者、メインアプリケーションおよびサブアプリケーションという3種類のロールがあります。次の図を使用して、それらの定義を説明します。



1. 開発者がVODサービスをアクティブにした場合、デフォルトでは**メインアプリケーション**ロールとなります。この場合のすべてのVODリソースは、メインアプリケーションに帰属します。メインアプリケーションの識別子は開発者のTencent Cloud APPIDであり、コンソールのアカウント情報で確認できます。

2. 開発者がVODサブアプリケーション機能をアクティブにすると、別の**管理者**ロールが生成されます。管理者はVODリソースを所有せず、すべてのリソースは引き続きメインアプリケーションに属します。
3. 開発者が管理者ロールを使用して**サブアプリケーション**を作成すると、新規作成されたサブアプリケーションは、独立したVODリソースを持ちます。サブアプリケーションはメインアプリケーションと対等の存在であり、メインアプリケーションとは相互に隔離されているので、メインアプリケーションは特殊なサブアプリケーションであると考えることができます。サブアプリケーションを作成すると、VODはサブアプリケーションIDと呼ばれる、プラットフォーム全体で一意の識別子を割り当てます。確認方法については、コンソールの使用説明 - アプリケーション管理をご参照ください。
4. 開発者が管理者ロールを使用して**サブアプリケーション**を再度作成すると、この新規作成されたサブアプリケーションも独立したVODリソースを持ちます。このサブアプリケーションもメインアプリケーションや他のサブアプリケーションと対等の存在であり、相互に隔離されています。以下同様です。

説明：
特に説明のない限り、以下ではメインアプリケーションとサブアプリケーションを区別せず、すべて**サブアプリケーション**と表現します。

機能

VODサブアプリケーションシステムは、次のような機能を提供します。
サブアプリケーションの作成および設定：開発者はVODサブアプリケーション機能をアクティブにすると、コンソールで管理者ロールによってサブアプリケーションを作成し、各サブアプリケーションの名前と説明を設定することができます。
サブアプリケーションの無効化：メインアプリケーション以外のサブアプリケーションは無効にすることができます。サブアプリケーションを無効にしても、サブアプリケーション下のVODリソースはクリアされず、そのドメイン名のみが無効になります。他の機能（アップロード、トランスコードなど）は影響を受けません。
リソースの隔離：サブアプリケーションとサブアプリケーション間のVODリソースは、相互に隔離されています。
サブアプリケーションのVODリソースはコンソールまたはサーバーAPIによって操作できます。
ストレージ、帯域幅/トラフィック、トランスコード時間、ビデオインテリジェント認識時間、再生データなど、サブアプリケーションごとに個別のデータ統計情報を生成します。
すべてのサブアプリケーションに対して概要データ統計情報を生成します。

制限

VODサブアプリケーションシステムには、次のような制限があります。
メインアプリケーションの名前と説明は変更できません。
サブアプリケーションは削除できません。
各VODアカウントは、最大50個のサブアプリケーションを作成することができます。
サブアプリケーションには個別の課金ロジックを設定できません（課金方式の設定、個別の課金生成、専用リソースパックの購入など）。1つのVODアカウント下のすべてのサブアプリケーションは、いずれも同じVODアカウントに属しています。すべてのサブアプリケーションのVOD使用量データ（ストレージ、トラフィック、トラン

スコード時間、ビデオインテリジェント認識時間などのVOD課金項目を含みますが、これらに限定されない）はすべて合算され、一括で課金されます。

コンソールの使用説明

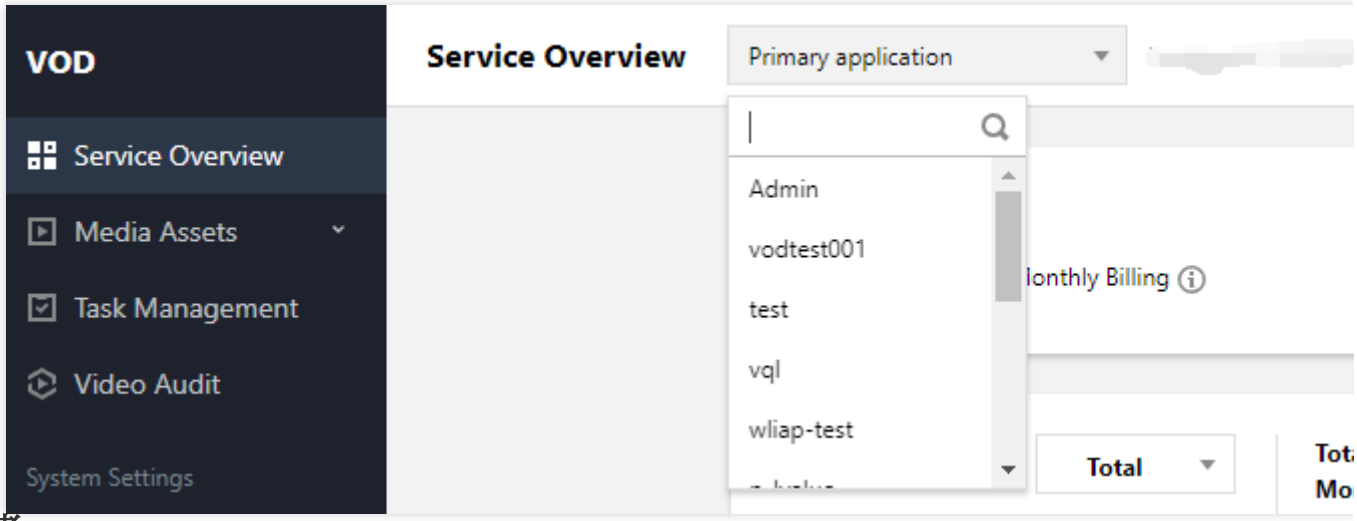
サブアプリケーションのアクティブ化

1. VODコンソールにログインします。
2. 左側ナビゲーションバーの**サブアプリケーションのアクティブ化**をクリックして、アクティブ化ページに進みます。
3. このページの**今すぐ使用する**をクリックすると、VODのサブアプリケーション機能がアクティブになります。

説明：
サブアプリケーションの機能がすでにアクティブ化されている場合、左側ナビゲーションバーの**サブアプリケーションのアクティブ化**は表示されません。

ロールの選択

サブアプリケーション機能をアクティブにすると、VODコンソールの**アプリケーション管理**の左上隅にロールを選択するドロップダウンリストが表示されるので、開発者はここでロールを選択できます。サブアプリケーション機能をアクティブにした直後は、ドロップダウンリストには「メインアプリケーション」という1つのロールしかありません。サブアプリケーションを作成すると、対応するロールがドロップダウンリストに追加されます。



管理者

管理者ロールの場合、左側ナビゲーションバーには、**サービスの概要、アプリケーション管理、使用量統計、リソースパック管理、Licnese管理**が表示されます。
サービスの概要：このページには、VOD課金方式、すべてのサブアプリケーションを集約した後の主要な業務データおよび各サブアプリケーションの主要な業務データが表示されます。

アプリケーション管理：このページでは、サブアプリケーションの確認、作成、編集および無効化を行うことができます。各サブアプリケーションの識別子（サブアプリケーションID）もこのページに表示されます。
使用量統計：このページでは、各アカウント下で使用する各製品機能の消費量を確認できます。
リソースパック管理：このページでは、各タイプのリソースパックの使用状況を確認できます。
License管理：このページでは、View CubeビデオLicenseおよびUGSVミニプログラムプラグインLicenseの使用状況を確認できます。

サブアプリケーション

サブアプリケーションロールの場合、VODコンソールの使用方法はサブアプリケーション機能が無効のときとほぼ同じです。サブアプリケーションに属するVODリソースを表示、操作することができます。2つの主な相違点は、サブアプリケーションには独自の課金設定がないところです。

サーバーAPIの使用説明

VODサブアプリケーション機能をアクティブ化後、開発者がVODサーバーAPIを使用する際は、どのサブアプリケーションのリソースにアクセスするかを指定する必要があります。

サーバーAPIでサブアプリケーションを指定

VODサーバーAPIは、Tencent Cloud API 3.0にアップグレードされました。ユーザーは各APIの `SubAppId` パラメータでアクセスしたいサブアプリケーションを指定できます。メインアプリケーションにアクセスする場合は、メインアプリケーションのIDを入力するか、もしくは空のままでもかまいません。

###サーバーAPI2017でサブアプリケーションを指定

サーバーAPI 2017もサブアプリケーションをサポートしています。使用する場合は、リクエストに `SubAppId` パラメータを追加する必要があります（大文字と小文字を区別して入力してください）。このパラメータとサーバーAPI 2017のパブリックリクエストパラメータは同じレベルであり、その値はサブアプリケーションIDです。メインアプリケーションにアクセスする場合は、メインアプリケーションIDを入力するか、もしくは空のままでもかまいません。

説明：
サーバーAPI 2017のドキュメントに `SubAppId` パラメータは公開されていませんが、このパラメータの使用には影響しません。
`SubAppId` パラメータはサーバーAPIの署名計算にも関与しますが、計算ルールは同じです。

ファイルアップロードの説明

VODサブアプリケーション機能をアクティブにした後、メディアファイルをVODにアップロードするサブアプリケーションを指定する必要があります。

CSSレコーディング

CSSレコーディング は、指定したサブアプリケーションでレコーディングを生成できます。指定方法とは、CSSプッシュパラメータに、 `vod_sub_app_id=xxx` （ `xxx` とはサブアプリケーションIDのこと）を追加することです。メインアプリケーションでレコーディングする場合は、このパラメータを空のままにします。

サーバーからのアップロード

サーバーからのアップロード は、指定されたサブアプリケーションへのアップロードをサポートします。具体的なパラメータの設定方法については、以下のリンクをご参照ください。メインアプリケーションにアップロードする場合は、メインアプリケーションのIDを入力するか、空のままでもかまいません。

SDKから

Java SDK
PHP SDK
Python SDK
Node.js SDK
Golang SDK

サーバーAPIから

API方式でアップロードする場合はApplyUploadとCommitUploadという2つのインターフェースが必要です。具体的な使用方法については、サーバーAPIでサブアプリケーションを指定をご参照ください。
SDKを使用してアップロードすることを強くお勧めします。

クライアントからのアップロード

クライアントからのアップロード では指定したサブアプリケーションへのアップロードをサポートします。指定方法とは、クライアントからのアップロードの署名 にパラメータ `vodSubAppId=xxx` （ `xxx` はサブアプリケーションIDのこと）を追加することです。メインアプリケーションにアップロードする場合は、メインアプリケーションのIDを入力するか、空のままでもかまいません。
説明：
`vodSubAppId` パラメータはクライアントからのアップロードの署名計算にも関与しますが、計算ルールは同じです。

URLからのプルアップロード

URLからのプルアップロードでは、指定したサブアプリケーションへのアップロードをサポートします。
コンソールから：具体的な使用法については、 コンソールの使用説明をご参照ください。

サーバーAPIから：PullUpload インターフェースを使用します。具体的な使用方法については、サーバーAPIでサブアプリケーションを指定をご参照ください。

権限管理

VODはCAMに接続されており、サブアプリケーションレベルの権限付与をサポートしています。詳細については、CAMをご参照ください。

FAQ

サブアプリケーション機能をアクティブにすると、オンラインになっている既存の業務ロジックに影響を与えますか。

影響は与えません。サブアプリケーションシステムは互換性を考慮して設計されています。すべてのサーバーAPIインターフェースは、サブアプリケーションIDが指定されていない場合、デフォルトでメインアプリケーションを操作します。

サブアプリケーション機能をアクティブ化するには料金がかかりますか。

サブアプリケーション機能自体は無料ですが、各サブアプリケーションによって発生した消費はすべてそのVODアカウントに計上され、課金ロジックに従って課金されます。

サブアプリケーション機能を使用して業務の隔離を実装していますが、各業務ではどのように内部決済/コストを実装するのですか。

前述の制限で説明したように、VODは、アカウント全体を取りまとめた請求書のみを発行します。複数の内部業務でコストを分担する必要がある場合は、VODが提供するサブアプリケーションレベルの統計データに基づいて、内部コストの分担をカスタマイズし、計算することができます。

サービスを一時停止した場合、サブアプリケーションにどのような影響を与えますか。

VODサービスに料金未払いによるサービス停止が発生した場合、このアカウント下のすべてのサブアプリケーションのサービスが停止します。

あるサブアプリケーションのビデオを、別のサブアプリケーションに移行できますか。

サブアプリケーション間のリソースは隔離されており、移行できません。

エラーコード

最終更新日：：2023-10-26 17:46:07

エラーコードリスト

ビデオ処理エラーコード

エラーコード	意味
InvalidInput	入力パラメータが不正のため、入力パラメータをチェックしてください。
InvalidInput.InvalidTimeOffset	入力パラメータが不正：指定されたタイムポイントが不正です。
InvalidInput.DefinitionNotExist	入力パラメータが不正：指定されたテンプレートIDが存在しません。
InvalidInput.ConfigurationUnsupported	入力パラメータが不正：無効な設定。以下を含むがこれらに限定されない： • ユーザーが未登録。 • 入力パラメータ値が不正（形式、値の範囲など）。 • パラメータテンプレートの設定に問題があります。 • ビデオ処理タスクを指定しません。
InvalidInput.TaskDuplicated	入力パラメータ値が不正：重複するタスクが存在します。
InvalidInput.PermissionDenied	入力パラメータ値が不正：権限がありません。この製品を使う権限を申請してください
InvalidInput.ResultFileSizeTooLarge	入力パラメータ値が不正：ファイルサイズが大きすぎます。
SourceFileError	ソースファイルエラー：ビデオデータが破損など、ソースファイルが正常かどうか確認してください。
SourceFileError.NoVideoMedia	ソースファイルエラー：ビデオトラック画面がありません。
SourceFileError.NoVideoResolution	ソースファイルエラー：ソースファイルの解像度を取得できません。
SourceFileError.ContentMalformed	ソースファイルエラー：入力内容（ファイルが存在しない、ファイルが破損している、メディアファイルをデコードできないなど）が正しくありません。

エラーコード	意味
SourceFileError.ContentUnsupported	ソースファイルエラー：入力形式（サポートされていないファイル形式、ファイルサイズ、ファイル期間など）が正しくありません。
SourceFileError.DownloadNotAccessible	ソースファイルエラー：入力ファイルをダウンロードしようとすると、これらのファイルにアクセスできません。ソースの可用性を確認してください。
InternalError	内部サービスエラーのため、リトライをお勧めします。