



デベロッパーガイド

Amazon Kinesis Video Streams



Amazon Kinesis Video Streams: デベロッパーガイド

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは Amazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有しない商標はすべてそれぞれの所有者に所属します。所有者は必ずしも Amazon と提携していたり、関連しているわけではありません。また、Amazon 後援を受けているとはかぎりません。

Table of Contents

Amazon Kinesis Video Streams とは	1
リージョンの可用性	1
仕組み	3
API およびプロデューサーライブラリ	4
データモデル	8
システム要件	16
カメラの要件	17
テスト済みオペレーティングシステム	17
SDK ストレージ要件	18
クォータ	18
コントロールプレーン API サービスクォータ	18
メディアおよびアーカイブメディア API サービスクォータ	23
フラグメントメタデータクォータとフラグメントメディアクォータ	27
ストリーミングメタデータサービスクォータ	30
プロデューサー SDK クォータ	31
階層型ストレージ	34
概要:	34
主なコンセプト	35
階層型ストレージのセットアップ	35
ベストプラクティス	36
アカウントをセットアップする	36
にサインアップする AWS アカウント	36
管理アクセスを持つユーザーを作成する	37
AWS アカウント キーを作成する	38
利点	38
開始方法	40
Amazon Kinesis ビデオストリームを作成する	40
コンソールを使用してビデオストリームを作成する	40
を使用してビデオストリームを作成する AWS CLI	41
Amazon Kinesis ビデオストリームにデータを送信する	41
SDK とサンプルを構築する	42
サンプルを実行して Kinesis Video Streams にメディアをアップロードする	45
確認オブジェクトを確認する	47
メディアデータの消費	47

コンソールでメディアを表示する	47
HLS を使用してメディアデータを使用する	48
Kinesis Video Streams へのアップロード	49
Kinesis Video Streams プロデューサークライアント	49
Kinesis Video Streams プロデューサーライブラリ	50
プロデューサーライブラリとは何かを理解する	51
Java	51
手順: Java プロデューサー SDK を使用する	52
前提条件	52
コードをダウンロードして設定する	53
コードを記述して調べる	54
リソースをクリーンアップする	56
コードを実行して検証する	56
Android	57
手順: Android プロデューサー SDK を使用する	57
前提条件	58
コードをダウンロードして設定する	61
コードを調べる	63
コードを実行して検証する	65
C++	66
オブジェクトモデル	66
ストリームにメディアを配置する	67
コールバックインターフェイス	67
手順: C++ プロデューサー SDK を使用する	68
前提条件	68
コードをダウンロードして設定する	69
コードを記述して調べる	74
コードを実行して検証する	81
SDK を GStreamer プラグインとして使用する	81
C++ プロデューサー SDK を Docker コンテナの GStreamer プラグインとして使用する	81
ログ記録を使用する	82
C プロデューサー	82
オブジェクトモデル	82
ストリームにメディアを配置する	84
手順: C プロデューサー SDK を使用する	84
前提条件	84

コードをダウンロードする	85
コードを記述して調べる	86
コードを実行して検証する	88
Raspberry Pi の C++	90
前提条件	91
ユーザーの作成	91
ネットワークに参加する	93
リモート接続	94
カメラを設定する	94
ソフトウェアのインストールの前提条件	96
C++ プロデューサー SDK をダウンロードして構築する	97
ライブストリームをストリーミングする	100
メディアを再生する	112
トラブルシューティング	115
エラーコードのリファレンス	120
PutFrame コールバックによって返されるエラーとステータスコード - プラットフォーム独立コード (PIC)	120
PutFrame コールバックによって返されるエラーとステータスコード - C プロデューサーライブラリ	169
NAL 適応フラグ	175
プロデューサー構造	177
DeviceInfo/ DefaultDeviceInfoProvider	177
StorageInfo	177
ストリーム構造	179
StreamDefinition/StreamInfo	179
ClientMetrics	196
StreamMetrics	197
コールバック	199
ClientCallbackProvider	199
StreamCallbackProvider	200
ClientCallbacks	201
ストリーミングを再試行するためのコールバック実装	206
ストリーミングメタデータの使用	207
Kinesis ビデオストリームへのメタデータの追加	208
Kinesis Video Streams で IPv6 を使用する	210
AWS SDK for IPv6 を設定する	210

IPv6 用の Kinesis Video Streams プロデューサー SDK を設定する	212
IPv6 AWS CLI の を設定する	214
設定例	214
考慮事項	214
IPv6 を含むアップグレードの影響を受けるお客様	215
トラブルシューティング	217
動画の再生	218
再生要件	219
GetClip	219
GetDASHStreamingSessionURL	220
GetHLSStreamingSessionURL	221
GetImages	222
HLS での再生	223
を使用して HLS ストリーミングセッション URL AWS CLI を取得する	223
例: HTML および JavaScript で HLS を使用する	227
HLS の問題のトラブルシューティング	231
MPEG-DASH での再生	233
例: HTML および JavaScript での MPEG-DASH の使用	234
通知の設定	239
通知設定の管理	239
UpdateNotificationConfiguration	239
DescribeNotificationConfiguration	240
プロデューサー MKV タグについて	240
プロデューサー MKV タグの構文	240
MKV タグの制限	240
Amazon SNS メッセージ	241
Amazon SNS トピックペイロード	241
Amazon SNS メッセージを表示する	242
クロスアカウント Amazon SNS 通知の発行	243
ID ベースのポリシー設定	243
リソースベースのポリシー設定	244
要件と考慮事項	244
ビデオストリームからイメージを抽出する	245
リアルタイムイメージ生成	245
フラグメントへのイメージ生成タグの追加	248
Amazon S3 オブジェクトパス (イメージ)	251

イメージメタデータの取得	251
スロットリングから保護するための Amazon S3 URI の推奨事項	252
トラブルシューティング	252
ビデオ分析にアクセスする	255
メタデータを消費する	255
パーサーライブラリを使用したストリーミング	256
前提条件	257
コードをダウンロードする	257
コードを調べる	258
コードを実行する	265
モニタリング	265
CloudWatch によるメトリクスのモニタリング	266
CloudWatch で Amazon Kinesis Video Streams Edge エージェントをモニタリングする	285
CloudTrail を使用した API コールのログ記録	290
ストリーミングメタデータの制限	295
ビデオ録画とストレージをスケジュールする	296
Amazon Kinesis Video Streams Edge Agent API オペレーション	297
Amazon Kinesis Video Streams Edge エージェントのモニタリング	297
非AWS IoT Greengrass モードでデプロイする	297
依存関係のインストール	298
IP カメラ RTSP URLs のリソースを作成する	300
IAM アクセス権限ポリシーを作成する	302
IAM ロールを作成する	304
AWS IoT ロールエイリアスを作成する	305
AWS IoT ポリシーを作成する	306
AWS IoT モノを作成して AWS IoT Core 認証情報を取得する	308
Edge エージェントの構築	310
CloudWatch エージェントをインストールする	321
Edge エージェントをネイティブプロセスとして実行する	324
にデプロイする AWS IoT Greengrass	326
Ubuntu インスタンスを作成する	327
AWS IoT Greengrass コアデバイスをセットアップする	329
IP カメラ RTSP URLs のリソースを作成する	330
TES ロールにアクセス許可を追加する	332
Secret Manager コンポーネントをインストールする	335
デバイスに Edge エージェントをデプロイする	338

AWS IoT Greengrass ログマネージャーコンポーネントをインストールする	347
よくある質問	351
Amazon Kinesis Video Streams Edge Agent はどのオペレーティングシステムをサポートしていますか?	351
Amazon Kinesis Video Streams Edge Agent は H.265 メディアをサポートしていますか? ..	352
Amazon Kinesis Video Streams Edge Agent は AL2 で動作しますか?	352
AWS IoT モノまたはデバイス内で複数のストリームを実行するにはどうすればよいですか?	352
送信StartEdgeConfigurationUpdate後に を編集するにはどうすればよいですか?	352
一般的な の例はありますかScheduleConfigs?	352
ストリームの上限はありますか?	353
エラーが発生したジョブを再起動するにはどうすればよいですか?	353
Amazon Kinesis Video Streams Edge エージェントの状態をモニタリングするにはどうすればよいですか?	354
VPC 経由でビデオをストリーミングする	355
追加情報	355
VPC エンドポイントの手順	355
例	358
例: Kinesis Video Streams へのデータの送信	358
例: Kinesis Video Streams からデータを取得する	358
例: 動画データの再生	358
前提条件	358
GStreamer プラグイン - kvssink	359
GStreamer 要素をダウンロード、構築、設定する	360
GStreamer 要素を実行する	360
起動コマンド	361
Docker コンテナで GStreamer 要素を実行する	363
パラメータリファレンス	366
PutMedia API	379
コードをダウンロードして設定する	380
コードを記述して調べる	381
コードを実行して検証する	383
RTSP および Docker	384
ビデオチュートリアル	384
前提条件	385
Docker イメージを作成します。	385

RTSP サンプルアプリケーションを実行する	386
レンダラー	387
前提条件	388
レンダラーの実行例	388
仕組み	389
API リファレンス	391
アクション	391
Amazon Kinesis Video Streams	393
Amazon Kinesis Video Streams Media	525
Amazon Kinesis Video Streams Archived Media	543
Amazon Kinesis Video Signaling Channels	592
Amazon Kinesis Video WebRTC ストレージ	601
データ型	610
Amazon Kinesis Video Streams	612
Amazon Kinesis Video Streams Media	653
Amazon Kinesis Video Streams Archived Media	656
Amazon Kinesis Video Signaling Channels	674
Amazon Kinesis Video WebRTC ストレージ	676
共通エラー	676
共通パラメータ	678
セキュリティ	681
データ保護	682
Kinesis Video Streams のサーバー側の暗号化とは	682
コスト、リージョン、およびパフォーマンスに関する考慮事項	682
サーバー側の暗号化の使用開始方法	683
カスタマーマネージドキーの作成と使用	684
カスタマーマネージドキーを使用するためのアクセス許可	684
IAM を使用した Kinesis Video Streams リソースへのアクセスの制御	686
ポリシー構文	687
Kinesis Video Streams のアクション	688
Kinesis Video Streams の Amazon リソースネーム (ARN)	688
他の IAM アカウントに Kinesis ビデオストリームへのアクセスを許可する	689
ポリシーの例	692
を使用した Kinesis Video Streams リソースへのアクセスの制御 AWS IoT	695
ストリーム名としてのAWS IoT ThingName	696
ストリーム名としてのAWS IoT CertificateId	702

AWS IoT 認証情報を使用してハードコードされたストリーム名にストリーミングする	704
コンプライアンス検証	705
耐障害性	705
インフラストラクチャセキュリティ	706
セキュリティのベストプラクティス	706
最小特権アクセスの実装	706
IAM ロールの使用	707
CloudTrail を使用した API コールのモニタリング	707
トラブルシューティング	708
一般的な問題	708
レイテンシーが高すぎる	708
API の問題	709
エラー: 「未知のオプション」	709
エラー: "承認するサービス/オペレーション名を特定できませんでした"	709
エラー: "ストリームにフレームを配置できませんでした"	710
エラー: "最終 AckEvent が受信される前に、サービスによって接続が終了されました"	710
エラー: 「STATUS_STORE_OUT_OF_MEMORY」	710
エラー: 「認証情報は有効なリージョンにスコープする必要があります」	711
HLS の問題	711
Java の問題	711
Java ログの有効化	711
プロデューサーライブラリの問題	712
プロデューサー SDK をコンパイルできない	713
ビデオストリームはコンソールには表示されません。	713
エラー: GStreamer デモアプリケーションを使用したデータのストリーミング時の "リクエストに含まれているセキュリティトークンが無効です"	714
エラー: 「Kinesis Video クライアントにフレームを送信できませんでした」	714
GStreamer アプリケーションが、OS X で "ストリーミングが中止されました。ネゴシエーションされていないという理由です" というメッセージで停止する	715
エラー: Raspberry Pi の GStreamer デモで Kinesis ビデオクライアントを作成するときの "ヒープを割り当てできませんでした"	715
エラー: Raspberry Pi での GStreamer デモの実行時の "無効な命令"	715
カメラで Raspberry Pi のロードに失敗する	716
カメラが macOS High Sierra で見つからない	716
macOS High Sierra でコンパイルするときに、jni.h ファイルが見つかりません	717
GStreamer デモアプリケーションを実行中の Curl エラー	717

Raspberry Pi での実行時のタイムスタンプ/範囲アサーション	717
Raspberry Pi の <code>gst_value_set_fraction_range_full</code> でのアサーション	717
Android での <code>STATUS_MKV_INVALID_ANNEXB_NALU_IN_FRAME_DATA(0x3200000d)</code> エラー	717
最大フラグメント期間に達したエラー	718
IoT 認可の使用中に "無効なモノの名前が渡されました (Invalid thing name passed)" エラーが発生	718
ストリームパーサーライブラリの問題	718
ストリームから 1 つのフレームにアクセスできない	719
フラグメントのデコードエラー	719
ネットワークの問題	719
ドキュメント履歴	721
.....	dccxxvi

Amazon Kinesis Video Streams とは

フルマネージド型の Amazon Kinesis Video Streams を使用すると AWS のサービス、デバイスからライブビデオをストリーミングしたり AWS クラウド、リアルタイムのビデオ処理やバッチ指向のビデオ分析用のアプリケーションを構築したりできます。

Kinesis Video Streams は、ビデオデータのストレージではありません。これを使用すると、ビデオストリームをクラウドで受信しながらリアルタイムで視聴できます。でライブストリームをモニタリングするか AWS マネジメントコンソール、Kinesis Video Streams API ライブラリを使用してライブビデオを表示する独自のモニタリングアプリケーションを開発できます。

Kinesis Video Streams を使用すると、スマートフォン、セキュリティカメラ、ウェブカメラ、車、ドローンやその他のソースに設置されるカメラのような何百万ものソースからライブ動画データの膨大な量を取得できます。オーディオデータ、熱画像、深度データ、RADAR データなど、ビデオ以外の時系列データを送信することもできます。これらのソースから Kinesis ビデオストリームへのライブビデオストリームとして、低レイテンシー処理のために frame-by-frame リアルタイムでデータにアクセスするアプリケーションを構築できます。Kinesis Video Streams はソースに依存しません。[GStreamer プラグイン - kvssink](#) ライブラリを使用してコンピュータのウェブカメラから、またはリアルタイムストリーミングプロトコル (RTSP) を使用してネットワーク上のカメラからビデオをストリーミングできます。

また、指定する保持期間でメディアデータを永続的に保存するように Kinesis のビデオストリームを設定することもできます。Kinesis Video Streams は、このデータを自動的に保存し、保管時には暗号化します。さらに、Kinesis Video Streams は、プロデューサータイムスタンプと取り込みタイムスタンプの両方に基づいて、保存されたデータのタイムインデックスを作成します。動画データを定期的にバッチ処理するアプリケーションを構築したり、さまざまなユースケースの履歴データへの 1 回限りのアクセスを必要とするアプリケーションを作成したりできます。

リアルタイムまたはバッチ指向のカスタムアプリケーションは、Amazon EC2 インスタンスで実行できます。これらのアプリケーションは、オープンソース、深層学習アルゴリズム、または Kinesis Video Streams と統合するサードパーティーアプリケーションを使用してデータを処理できます。

リージョンの可用性

Amazon Kinesis Video Streams は、次のリージョンで利用できます。

リージョン名	AWS リージョンコード
米国東部 (オハイオ)	us-east-2
米国東部 (バージニア北部)	us-east-1
米国西部 (オレゴン)	us-west-2
AWS GovCloud (米国東部)	us-gov-east-1
AWS GovCloud (米国西部)	us-gov-west-1
アフリカ (ケープタウン)	af-south-1
アジアパシフィック (香港)	ap-east-1
アジアパシフィック (ムンバイ)	ap-south-1
アジアパシフィック (ソウル)	ap-northeast-2
アジアパシフィック (シンガポール)	ap-southeast-1
アジアパシフィック (シドニー)	ap-southeast-2
アジアパシフィック (マレーシア)	ap-southeast-3
アジアパシフィック (東京)	ap-northeast-1
カナダ (中部)	ca-central-1
中国 (北京)	cn-north-1
欧州 (フランクフルト)	eu-central-1
欧州 (アイルランド)	eu-west-1
欧州 (ロンドン)	eu-west-2
欧州 (パリ)	eu-west-3
欧州 (スペイン)	eu-south-2

リージョン名	AWS リージョンコード
中東 (バーレーン)	me-south-1
南米 (サンパウロ)	sa-east-1

Kinesis Video Streams: 仕組み

トピック

- [Kinesis Video Streams API とプロデューサーライブラリのサポート](#)
- [Kinesis Video Streams データモデル](#)

フルマネージド型の Amazon Kinesis Video Streams を使用して AWS のサービス、デバイスから AWS クラウド にライブビデオをストリーミングし、永続的に保存できます。その後、リアルタイムで動画を処理するために独自のアプリケーションを構築するか、バッチ指向の動画分析を実行できます。

次の図表は、Kinesis Video Streams の仕組みの概要を示しています。

この図は、次のコンポーネント間のやり取りを示しています。

- Producer - Kinesis のビデオストリームにデータを送る任意のソース。プロデューサーは、セキュリティカメラ、ボディウェアカメラ、スマートフォンカメラ、ダッシュボードカメラなどのビデオ生成デバイスです。プロデューサーは、音声フィード、イメージ、RADAR データなどの動画以外のデータも送信できます。

1 つのプロデューサーで複数のビデオストリームを生成できます。例えば、ビデオカメラは動画データを 1 つの Kinesis のビデオストリームにプッシュし、音声データを別のストリームにプッシュすることができます。

- Kinesis Video Streams プロデューサーライブラリ - デバイスにインストールして設定できるソフトウェアとライブラリのセット。これらのライブラリを使用して、動画を安全に接続し、リアルタイムで、数秒間バッファした後、事後after-the-factメディアアップロードとしてなど、さまざまな方法で確実にストリーミングできます。

- Kinesis ビデオストリーム – ライブビデオデータを転送し、オプションで保存して、リアルタイム、バッチベース、または 1 回限りのデータを使用できるようにするリソース。一般的な設定の場合、Kinesis のビデオストリームには、それに対してデータを発行するプロデューサーが 1 つだけ用意されています。

音声、動画のほか、奥行き感知フィードや RADAR フィードなどの、時間がエンコードされた類似のデータストリームを扱うことができます。を使用して、AWS マネジメントコンソール または SDKs を使用して AWS プログラムで Kinesis ビデオストリームを作成します。

複数の独立したアプリケーションでは、Kinesis のビデオストリームを並列で消費できます。

- コンシューマー - フラグメントやフレームなどのデータを Kinesis のビデオストリームから取得して、表示、処理、または分析します。一般的に、これらのコンシューマーは Kinesis Video Streams アプリケーションと呼ばれます。Kinesis Video Streams でデータをリアルタイムで消費して処理するアプリケーション、または低レイテンシー処理が不要な場合にデータを保存して時間インデックスを作成するアプリケーションを作成できます。これらのコンシューマーアプリケーションを作成して Amazon EC2 インスタンス上で実行できます。
- [パーサーライブラリを使用してカメラからの出力を監視する](#) – Kinesis Video Streams アプリケーションが Kinesis ビデオストリームから低レイテンシーでメディアを確実に取得できるようにします。また、メディア内のフレームの境界を解析し、アプリケーションでフレーム自体の処理や分析を集中的に実行できるようにします。

Kinesis Video Streams API とプロデューサーライブラリのサポート

Kinesis Video Streams には、ストリームを作成および管理し、ストリーム間でメディアデータの読み取りまたは書き込みを行うための API が用意されています。Kinesis Video Streams コンソールは管理機能に加えて、ライブ再生およびビデオオンデマンドの再生もサポートしています。Kinesis Video Streams は、アプリケーションコードで使用する、データをメディアソースから抽出したり、Kinesis のビデオストリームにアップロードすることができる一連のプロデューサーライブラリも提供します。

トピック

- [Kinesis Video Streams API](#)
- [エンドポイント検出パターン](#)
- [プロデューサーライブラリ](#)

Kinesis Video Streams API

Kinesis Video Streams には APIs が用意されています。また、メディアデータをストリームから読み取ったり、ストリームに書き込むための API も用意されています。

- **Producer API** - Kinesis Video Streams には、メディアデータを Kinesis のビデオストリームに書き込むための PutMedia API が用意されています。PutMedia リクエストで、プロデューサーはメディアフラグメントのストリームを送信します。フラグメントとは、自己完結型のフレームのシーケンスです。フラグメントに属するフレームは、他のフラグメントからのフレームに依存していないことが求められます。詳細については、「[PutMedia](#)」を参照してください。

フラグメントが届くと、Kinesis Video Streams では一意のフラグメント番号を昇順で割り当てます。また、各フラグメントのプロデューサー側とサーバー側のタイムスタンプを Kinesis Video Streams 固有のメタデータとして保存します。

- **コンシューマー APIs** – コンシューマーは、次の APIs を使用してストリームからデータを取得できます。
- **GetMedia** - この API を使用するとき、コンシューマーは開始フラグメントを識別する必要があります。次に、API はストリームに追加された順番 (昇順のフラグメント番号) でフラグメントを返します。フラグメント内のメディアデータは、[Matroska \(MKV\)](#) などの構造化された形式にまとめられています。詳細については、「[GetMedia](#)」を参照してください。

Note

GetMedia では、フラグメントの場所を認識します (データストア内にアーカイブされているか、リアルタイムで利用可能)。たとえば、開始フラグメントがアーカイブされていることを GetMedia が判断すると、フラグメントがデータストアから返され始めます。まだアーカイブされていない新しいフラグメントを返す必要がある場合、はインメモリストリームバッファからのフラグメントの読み取り GetMedia に切り替えます。

これは、ストリームによって取り込まれた順番でフラグメントを処理する継続的なコンシューマーの例です。

GetMedia では、動画処理アプリケーションが失敗したり、遅延した後でも、問題なく処理を挽回することができます。GetMedia を使用すると、アプリケーションでは、データストアにアーカイブされているデータを処理でき、アプリケーションが処理に追いついてきたところで、届いたメディアデータを GetMedia がリアルタイムで引き続き配信するようになります。

- `GetMediaFromFragmentList` (および `ListFragments`) - バッチ処理アプリケーションはオフラインコンシューマーと見なされます。オフラインコンシューマーは、`ListFragments` と `GetMediaFromFragmentList` の API を組み合わせることで、特定のメディアフラグメントまたは動画の範囲を明示的にフェッチできます。`ListFragments` および `GetMediaFromFragmentList` を使用すると、アプリケーションは、特定の時間範囲またはフラグメント範囲の動画のセグメントを識別し、これらのフラグメントを順番に、または並行して処理するためにフェッチできます。このアプローチは、大量のデータを並行して迅速に処理する必要がある MapReduce アプリケーションに適しています。

たとえば、コンシューマーが 1 日分の動画フラグメントを処理する必要があるとします。コンシューマーは次のことを行います。

1. `ListFragments` API を呼び出し、時間範囲を指定して目的のフラグメントのコレクションを選択することで、フラグメントのリストを取得します。

API は、指定された時間範囲内のすべてのフラグメントからメタデータを返します。メタデータは、フラグメント番号、プロデューサー側、サーバー側のタイムスタンプなどの情報を提供します。

2. フラグメントのメタデータリストを使用して、フラグメントを任意の順序で取得します。例えば、その日のすべてのフラグメントを処理するために、コンシューマーはリストをサブリストに分割し、ワーカー (複数の Amazon EC2 インスタンスなど) に を使用してフラグメントを並行してフェッチさせ `GetMediaFromFragmentList`、並行して処理することを選択できます。

次の図は、これらの API コール中のフラグメントとチャンクの詳細なデータフローを示しています。

プロデューサーが `PutMedia` リクエストを送信するときは、ペイロード内のメディアメタデータを送信してから、メディアデータフラグメントのシーケンスを送信します。Kinesis Video Streams はデータを受け取ると、Kinesis Video Streams のチャンクとして着信メディアデータを保存します。各チャンクは以下で構成されています。

- メディアメタデータのコピー
- フラグメント
- Kinesis Video Streams 固有のメタデータ。フラグメント番号、サーバー側およびプロデューサー側のタイムスタンプなど

コンシューマーがメディアメタデータをリクエストすると、Kinesis Video Streams は、リクエストで指定されたフラグメント番号から始まるチャンクのストリームを返します。

ストリームのデータの永続性を有効にした場合、ストリームでフラグメントを受け取った後に、Kinesis Video Streams もフラグメントのコピーをデータストアに保存します。

エンドポイント検出パターン

コントロールプレーン REST APIs

[Kinesis Video Streams コントロールプレーン REST APIs](#) にアクセスするには、[Kinesis Video Streams サービスエンドポイント](#)を使用します。

データプレーン REST APIs

Kinesis Video Streams は、セル[ラーアーキテクチャ](#)を使用して構築され、スケーリングとトラフィック分離のプロパティを向上させます。各ストリームはリージョン内の特定のセルにマッピングされるため、アプリケーションはストリームがマッピングされた正しいセル固有のエンドポイントを使用する必要があります。Data Plane REST APIs にアクセスするときは、適切なエンドポイントを自分で管理してマッピングする必要があります。このプロセスを示したエンドポイント検出パターンの詳細は次のとおりです。

1. エンドポイント検出パターンは、いずれかのGetEndpointsアクションの呼び出しから始まります。これらのアクションはコントロールプレーンに属します。
 1. [the section called “Amazon Kinesis Video Streams Media”](#) または [the section called “Amazon Kinesis Video Streams Archived Media”](#) サービスのエンドポイントを取得する場合は、[the section called “GetDataEndpoint”](#)を使用します。
 2. [the section called “Amazon Kinesis Video Signaling Channels”](#)、[the section called “Amazon Kinesis Video WebRTC ストレージ”](#)または [Kinesis Video Signaling](#) のエンドポイントを取得する場合は、[the section called “GetSignalingChannelEndpoint”](#)を使用します。
2. エンドポイントをキャッシュして再利用します。
3. キャッシュされたエンドポイントが機能しなくなった場合は、[the section called “GetEndpoints”](#)を新たに呼び出しGetEndpointsでエンドポイントを更新します。

プロデューサーライブラリ

Kinesis のビデオストリームの作成後は、ストリームへのデータの送信を開始できます。アプリケーションコードで、これらのライブラリを使用してデータをメディアソースから抽出したり、Kinesis

のビデオストリームにアップロードすることができます。使用可能なプロデューサーライブラリの詳細については、「[Kinesis Video Streams へのアップロード](#)」を参照してください。

Kinesis Video Streams データモデル

[Kinesis Video Streams へのアップロード](#) および [the section called “パーサーライブラリを使用したストリーミング”](#) は、動画データに伴う情報の埋め込みをサポートする形式で動画データを送受信します。この形式は Matroska (MKV) 仕様に基いています。

[MKV 形式](#) は、メディアデータに対するオープン仕様です。Amazon Kinesis Video Streams Developer Guide 内のすべてのライブラリおよびコードの例は、MKV 形式でデータを送受信します。

[Kinesis Video Streams へのアップロード](#) は StreamDefinition および Frame タイプを使用して、MKV ストリームヘッダー、フレームヘッダー、フレームデータを生成します。

MKV 仕様の詳細については「[Matroska Specification](#)」を参照してください。

以下のセクションでは、[C++](#) によって作成された MKV 形式データのコンポーネントについて説明します。

トピック

- [ストリームヘッダー要素](#)
- [トラックデータをストリーミングする](#)
- [フレームヘッダー要素](#)
- [MKV フレームデータ](#)

ストリームヘッダー要素

StreamDefinition では次の MKV ヘッダー要素が使用されます (StreamDefinition.h で定義)。

Element	説明	一般的な値
stream_name	Kinesis のビデオストリームの名前。	my-stream

Element	説明	一般的な値
retention_period	ストリームデータが Kinesis Video Streams によって保持される時間単位の期間。データを保持しないストリーム0にを指定します。	24
[タグ]	ユーザーデータのキーと値のコレクション。このデータは AWS マネジメントコンソールに表示され、クライアントアプリケーションにより読み取ることによってストリームに関する情報をフィルタリングまたは取得できます。	
kms_key_id	存在する場合、ユーザー定義 AWS KMS キーを使用してストリーム上のデータを暗号化します。存在しない場合、データは Kinesis が提供するキー () によって暗号化されますaws/kinesisvideo 。	01234567-89ab-cdef-0123-456789ab
streaming_type	現在、有効なストリーミングタイプは STREAMING_TYPE_REALTIME のみです。	STREAMING_TYPE_REALTIME
content_type	ユーザー定義のコンテンツタイプ。動画データをストリーミングしてコンソールで再生する場合、コンテンツタイプは video/h264 である必要があります。	video/h264

Element	説明	一般的な値
max_latency	この値は現在使用されていないため、0 に設定する必要があります。	0
fragment_duration	フラグメントの継続時間 (推定)。この時間が最適化に使用されます。実際のフラグメント継続時間は、ストリーミングデータによって決定します。	2
timecode_scale	フレームタイムスタンプで使 用されるスケールを示しま す。デフォルト値は 1 ミリ秒 です。0 を指定すると、デフ ォルト値の 1 ミリ秒も割り当 てられます。この値は 100 ナ ノ秒～1 秒の範囲で指定でき ます。 詳細については、Matroska の ドキュメントの「TimecodeS cale」を参照してください。	
key_frame_fragmentation	true の場合、ストリームは キーフレームが受信された場 合に新たなクラスターを開始 します。	true

Element	説明	一般的な値
frame_timecodes	の場合true、Kinesis Video Streams は受信したフレームの表示タイムスタンプ (pts) とデコードタイムスタンプ (dts) の値を使用します。の場合false、Kinesis Video Streams はシステム生成の時間値でフレームを受信したときにフレームをスタンプします。	true
absolute_fragment_time	true の場合、クラスタータイムコードは絶対時間 (プロデューサーのシステムクロックなど) を用いて解釈されます。false の場合、クラスタータイムコードはストリームの開始時刻の相対値として解釈されます。	true
fragment_acks	true の場合、Kinesis Video Streams がデータを受信した際に確認 (ACK) が送信されます。ACK は KinesisVideoStreamFragmentAck または KinesisVideoStreamParseFragmentAck コールバックを用いて受信できます。	true
restart_on_error	ストリームのエラーが発生した後、ストリームが送信を再開すべきかどうかを示します。	true

Element	説明	一般的な値
nal_adaptation_flags	コンテンツ内に NAL (Network Abstraction Layer) 適応またはコーデックプライベートデータが含まれるかどうかを示します。有効なフラグには NAL_ADAPTATION_ANNEXB_NALS および NAL_ADAPTATION_ANNEXB_CPD_NALS が含まれます。	NAL_ADAPTATION_ANN EXB_NALS
frame_rate	コンテンツの推定フレームレート。この値は最適化に使用され、実際のフレームレートは受信データのレートにより決定されます。0 を指定すると、デフォルトの 24 が割り当てられます。	24
avg_bandwidth_bps	コンテンツ帯域幅の推定値。Mbps 単位。この値は最適化に使用され、実際のレートは受信データの帯域幅により決定されます。たとえば、25 FPS で動作する 720p の解像度のビデオストリームでは、平均 5 Mbps の帯域幅を期待できます。	5

Element	説明	一般的な値
buffer_duration	コンテンツがプロデューサー上でバッファされる時間。ネットワークレイテンシーが低い場合、この値を減らすことができます。ネットワークレイテンシーが高い場合、この値を大きくすると、割り当てが小さいバッファにフレームを配置できないため、送信前にフレームがドロップされるのを防ぎます。	
replay_duration	接続が失われた場合にビデオデータストリームが「巻き戻し」される時間。接続損失によるフレームの損失が懸念されない場合、この値はゼロになります。消費するアプリケーションが冗長フレームを削除できる場合は、値を増やすことができます。この値はバッファ期間より短くする必要があります。そうでない場合はバッファ期間が使用されます。	
connection_staleness	データが受信されない場合に接続を維持する期間。	
codec_id	コンテンツで使用するコーデック。詳細情報については Matroska 仕様の「 CodecID 」を参照してください。	V_MPEG2
track_name	ユーザー定義のトラック名。	my_track

Element	説明	一般的な値
codecPrivateData	多くのダウンストリームコンシューマーにおいて必要となる、フレームデータのデコードのためにエンコーダーが提供するデータ (ピクセル単位でのフレーム幅および高さなど)。 C++ プロデューサーライブラリ では、 <code>gMkvTrackVideoBits</code> 配列にフレームのピクセル幅と高さ <code>MkvStatics.cpp</code> が含まれます。	
codecPrivateDataSize	codecPrivateData パラメータのデータのサイズ。	
track_type	ストリームのトラックのタイプ。	MKV_TRACK_INFO_TYPE_AUDIO または MKV_TRACK_INFO_TYPE_VIDEO
segment_uuid	ユーザー定義のセグメント uuid (16 バイト)。	
default_track_id	トラックの、一意のゼロ以外の数。	1

トラックデータをストリーミングする

StreamDefinition では次の MKV トラック要素が使用されます (StreamDefinition.h で定義)。

Element	説明	一般的な値
track_name	ユーザー定義のトラック名。 たとえば、オーディオトラック用の「audio」。	audio
codec_id	トラック用のコーデック ID。 たとえば、オーディオトラック用の「A_AAC」。	A_AAC
cpd	フレームデータのデコードのためにエンコーダーが提供するデータ。このデータには、フレームの幅と高さ (ピクセル単位) を含めることができます。この情報は多くのダウンストリームコンシューマーで必要となります。 C++ プロデューサーライブラリ では、MkvStatics.cpp の gMkvTrackVideoBits 配列にフレームのピクセル幅と高さが含まれています。MkvStatics.cpp	
cpd_size	codecPrivateData パラメータのデータサイズ。	
track_type	トラックのタイプ。たとえば、オーディオ用の MKV_TRACK_INFO_TYPE_AUDIO の列挙値を使用できます。	MKV_TRACK_INFO_TYPE_AUDIO

フレームヘッダー要素

Frame では次の MKV ヘッダー要素が使用されます (mkvgen/Include.h の KinesisVideoPic パッケージで定義)。

- Frame Index: 一定間隔で増加する値。
- Flags: フレームのタイプ。有効な値は次のとおりです。
 - FRAME_FLAGS_NONE
 - FRAME_FLAG_KEY_FRAME: key_frame_fragmentation がストリーム上で設定されている場合、キーフレームは新たなフラグメントを開始します。
 - FRAME_FLAG_DISCARDABLE_FRAME: デコーダーに対し、デコーディングが遅い場合はこのフレームを破棄できることを通知します。
 - FRAME_FLAG_INVISIBLE_FRAME: このブロックの時間は 0 です。
- デコードタイムスタンプ: このフレームがデコードされた時刻のタイムスタンプ。以前のフレームがデコードのためにこのフレームに依存している場合、このタイムスタンプは以前のフレームよりも前になる可能性があります。この値はフラグメントの開始の相対値です。
- プレゼンテーションタイムスタンプ: このフレームが表示されるときタイムスタンプ。この値はフラグメントの開始の相対値です。
- Duration: フレームの再生時間。
- Size: フレームデータのサイズ (バイト単位)

MKV フレームデータ

frame.frameData 内のデータには、使用されるエンコーディングスキーマに応じ、フレームのメディアデータのみが含まれている場合、あるいはさらにネスト化されたヘッダー情報が含まれている場合があります。に表示するには AWS マネジメントコンソール、データを [H.264](#) コーデックでエンコードする必要がありますが、Kinesis Video Streams はタイムシリアル化されたデータストリームを任意の形式で受信できます。

Amazon Kinesis Video Streams システム要件

以下のセクションで、Amazon Kinesis Video Streams のハードウェア、ソフトウェア、ストレージの要件を説明します。

トピック

- [カメラの要件](#)

- [テスト済みオペレーティングシステム](#)
- [SDK ストレージ要件](#)

カメラの要件

Kinesis Video Streams プロデューサー SDK とサンプルの実行に使用されるカメラには、次のメモリ要件があります。

- SDK コンテンツビューには 16 MB のメモリが必要です。
- サンプルアプリケーションのデフォルト設定は 512 MB です。この値は、ネットワーク接続が良好で追加バッファリングの必要がないプロデューサーに適しています。ネットワーク接続の状態が悪く、必要とするバッファリングが大きい場合、1 秒あたりのフレームレートをフレームメモリサイズで乗算してバッファリング 1 秒あたりに必要なメモリを計算できます。メモリの割り当ての詳細については、「[StorageInfo](#)」を参照してください。

H.264 を使用してデータをエンコードする USB または RTSP (Real Time Streaming Protocol) カメラを使用することをお勧めします。CPU のエンコード負荷がなくなるためです。

現在、デモアプリケーションは RTSP ストリーミングのユーザーデータグラムプロトコル (UDP) をサポートしていません。この機能は今後追加される予定です。

プロデューサー SDK は、次のタイプのカメラをサポートしています。

- ウェブカメラ。
- USB カメラ。
- H.264 エンコードができるカメラ (推奨)。
- H.264 エンコードではないカメラ。
- Raspberry Pi カメラモジュール。Raspberry Pi デバイスでこれが推奨されるのは、ビデオデータ転送では GPU に接続されるため、CPU 処理のオーバーヘッドがないためです。
- RTSP (ネットワーク) カメラ。これらのカメラが推奨されるのは、ビデオストリームが H.264 でエンコードされるためです。

テスト済みオペレーティングシステム

ウェブカメラおよび RTSP カメラは、以下のデバイスとオペレーティングシステムでテストされています。

- Mac mini
 - High Sierra
- MacBook Pro ノートパソコン
 - Sierra (10.12)
 - El Capitan (10.11)
- Ubuntu 16.04 を実行している HP ノートパソコン
- Ubuntu 17.10 (Docker コンテナ)
- Raspberry Pi 3

SDK ストレージ要件

[Kinesis Video Streams へのアップロード](#) をインストールする場合の最小ストレージ要件は 170 MB、推奨ストレージ要件は 512 MB です。

Amazon Kinesis Video Streams サービスクォータ

Kinesis Video Streams には次のサービスクォータがあります。

Important

次のサービスクォータは、サポートチケットを送信してアップグレードできるソフト [s]、または引き上げることができないハード [h] のいずれかです。以下の表では、個々のサービスクォータの横に [s] と [h] が表示されます。

コントロールプレーン API サービスクォータ

次のセクションでは、コントロールプレーン APIs のサービスクォータについて説明します。TPS は Transactions Per Second (1 秒あたりのトランザクション数) の略です。

アカウントレベルまたはリソースレベルのリクエスト制限に達すると、`ClientLimitExceededException` がスローされます。

API	アカウント制限: リクエスト	アカウント制限: ストリーム	ストリームレベルの制限	関連する例外と注意事項
CreateStream	50 TPS [s]	サポートされているすべてのリージョンで、アカウント [s] あたり 10,000 ストリーム。 Note この制限は、アカウントあたり 100,000 ストリーム（またはそれ以上）まで増やすことができます。		デバイス、CLI、SDK 駆動型のアクセス、およびコンソールは、すべてこの API を呼び出します。ストリームが既に存在しない場合、1 つの API コールのみが成功します。

API	アカウント制限: リクエスト	アカウント制限: ストリーム	ストリームレベルの制限	関連する例外と注意事項
		AWS マネ ジメ ント コン ソ ール でに サイ ンイ ンhttps:// console.a ws.amazo .com/し、 この 制限 の引 き 上げ をリ クエ スト しま す。		
DeleteEdgeConfiguration	10 TPS [h]	該当なし	10 TPS [h]	
DeleteStream	50 TPS [h]	該当なし	5 TPS [h]	

API	アカウント制限: リクエスト	アカウント制限: ストリーム	ストリームレベルの制限	関連する例外と注意事項
DescribeEdgeConfiguration	50 TPS [h]	該当なし	5 TPS [h]	
DescribeImageGenerationConfiguration	50 TPS [h]	該当なし	5 TPS [h]	
DescribeMappedResourceConfiguration	50 TPS [h]	該当なし	5 TPS [h]	
DescribeNotificationConfiguration	50 TPS [h]	該当なし	5 TPS [h]	
DescribeStream	300 TPS [h]	該当なし	5 TPS [h]	
GetDataEndpoint	300 TPS [h]	該当なし	5 TPS [h]	ほとんどの PutMedia/GetMedia ユースケースでは、ストリーミングトークンを更新するために 45 分ごとに呼び出されます。データエンドポイントのキャッシュは、アプリケーションで失敗時に再ロードされる場合、安全です。

API	アカウント制限: リクエスト	アカウント制限: ストリーム	ストリームレベルの制限	関連する例外と注意事項
ListEdgeAgentConfigurations	50 TPS [h]	該当なし	該当なし	
ListStreams	50 TPS [h]	該当なし		
ListTagsForStream	50 TPS [h]	該当なし	5 TPS [h]	
StartEdgeConfigurationUpdate	10 TPS [h]	該当なし	10 TPS [h]	
TagStream	50 TPS [h]	該当なし	5 TPS [h]	
UntagStream	50 TPS [h]	該当なし	5 TPS [h]	
UpdateDataRetention	50 TPS [h]	該当なし	5 TPS [h]	
UpdateImageGenerationConfiguration	50 TPS [h]	該当なし	5 TPS [h]	
UpdateNotificationConfiguration	50 TPS [h]	該当なし	5 TPS [h]	
UpdateStream	50 TPS [h]	該当なし	5 TPS [h]	

API	アカウント制限: リクエスト	アカウント制限: ストリーム	ストリームレベルの制限	関連する例外と注意事項
UpdateStreamStorageConfiguration	50 TPS [h]	該当なし	5 TPS [h]	
DescribeStreamStorageConfiguration	50 TPS [h]	該当なし	5 TPS [h]	

メディアおよびアーカイブメディア API サービスクォータ

次のセクションでは、メディア API とアーカイブメディア APIs のサービスクォータについて説明します。

アカウントレベルまたはリソースレベルのリクエスト制限に達すると、`ClientLimitExceededException` がスローされます。

接続レベルのリクエスト制限に到達すると、`ConnectionLimitExceededException` がスローされます。

次のエラーまたは ACK は、フラグメントレベルの制限に達したときにスローされます。

- `MIN_FRAGMENT_DURATION_REACHED ACK` は、最小期間より小さいフラグメントに対して返されます。
- `MAX_FRAGMENT_DURATION_REACHED ACK` は、最大期間より大きいフラグメントに対して返されます。
- `MAX_FRAGMENT_SIZE ACK` は、最大データサイズより大きいフラグメントに対して返されます。
- `GetMediaForFragmentList` オペレーションでフラグメントの制限に達した場合、`FragmentLimitExceeded` の例外がスローされます。

データプレーン API サービスクォータ

API	ストリームレベルの制限	接続レベルの制限	帯域幅制限	フラグメントレベルの制限	関連する例外と注意事項
PutMedia	5 TPS [h]	1 [h]	ストリームあたり 12.5 MB/秒、または 100 Mbps [s]	- 最小フラグメント継続時間: 1 秒 [h] - 最大フラグメント継続時間: 20 秒 [h] - 最大フラグメントサイズ: 50 MB [h] - トラックの最大数: 3 [s] 1 秒あたりに送信されるフラグメントの最大数: 5 [h] - フラグメントメタデータの最大制限: 10 タグ [h]	PutMedia リクエストはストリーミングで長時間実行される接続です。1 つの永続接続で複数のフラグメントを送信できるため、データごとに新しい接続を開く必要はありません。複数の同時PutMedia接続を試みると、Kinesis Video Streams は最新の接続をConnectionLimitExceededException エラーメッセージでスロットリングします。

API	ストリームレベルの制限	接続レベルの制限	帯域幅制限	フラグメントレベルの制限	関連する例外と注意事項
GetClip	該当なし	該当なし	100 MB のサイズ制限 [h]	フラグメントの最大数: 200 [h]	
GetDASHStreamingSessionURL	25 TPS [h]	該当なし	該当なし	該当なし	
GetHLSStreamingSessionURL	25 TPS [h]	該当なし	該当なし	該当なし	
GetImages	該当なし	該当なし	100 MB [h]	該当なし	リクエストあたりのイメージの最大数は 100 [h] です。
					 Note の最小値SamplingInterval は 200 ミリ秒 (ms) で、1 秒あたり 5 イメージです。

API	ストリームレベルの制限	接続レベルの制限	帯域幅制限	フラグメントレベルの制限	関連する例外と注意事項
GetMedia	5 TPS [h]	3 [h]	25 MB/秒 または 200 Mbps [s]	1 秒あたり 最大 5 個 のフラグメントを送信 [h]	GetMedia リクエストはストリーミングで長時間実行される接続です。1 つの永続接続で複数のフラグメントを送信できるため、データごとに新しい接続を開く必要はありません。3 つ以上の同時GetMedia接続を試みると、Kinesis Video Streams は最新の接続をConnectionLimitExceededException エラーメッセージでスロットリングします。 一般的なフラグメントが約 5 MB の場合、この制限は Kinesis ビデオストリームあたり約 75 MBps を意味します。このようなストリームは、ストリームの最大受信ビットレートの 2 倍の送信ビットレートを持ちます。  Note GetMedia は HLS/DASH 再生

API	ストリームレベルの制限	接続レベルの制限	帯域幅制限	フラグメントレベルの制限	関連する例外と注意事項
					には使用されません。
GetMediaFragmentList	該当なし	5 [s]	25 MB/秒 または 200 Mbps [s]	フラグメントの最大数: 1000 [h]	5 つのフラグメントベースの消費アプリケーションが同時に呼び出すことができますGetMediaFragmentList。それ以上の接続は拒否されます。

動画再生プロトコル API サービスクォータ

API	セッションレベルの制限	フラグメントレベルの制限
GetDASHManifestPlaylist	5 TPS [h]	プレイリストあたりのフラグメントの最大数: 5,000 [h]
GetHLSMasterPlaylist	5 TPS [h]	該当なし
GetHLSMediaPlaylist	5 TPS [h]	プレイリストあたりのフラグメントの最大数: 5,000 [h]
GetMP4InitFragment	5 TPS [h]	該当なし
GetMP4MediaFragment	20 TPS [h]	該当なし
GetTSFragment	20 TPS [h]	該当なし

フラグメントメタデータクォータとフラグメントメディアクォータ

Kinesis Video Streams の [アーカイブされたメディアにアクセスするための API](#) は、API コールの回数ではなく、リクエストされたフラグメントの数に基づいてスロットリングされます。APIs は、

フラグメントメタデータの数とリクエストされたフラグメントメディアの数の両方によってレート制限されます。フラグメントメタデータクォータとフラグメントメディアクォータは、ストリームごとに適用されます。つまり、あるストリーム内のフラグメントメタデータまたはフラグメントメディアのリクエストは、別のストリームのクォータには適用されません。ただし、特定のストリーム内では、各クォータは複数の API 間で共有されます。これは、特定のストリームでは、異なる API にまたがるフラグメントに対するリクエストは、同じクォータから消費されるからです。ストリームのフラグメントメタデータまたはフラグメントメディアクォータが制限を超えると、API は `ClientLimitExceededException` を返します。次の表は、2 種類のクォータのそれぞれについて、API がどのように消費するかを示しています。表の 2 番目の列では、あるストリームが N 個のクォータを持つと仮定します。すなわち、API は、そのストリームのそのクォータタイプから消費する N 個のポイントを持ちます。GetClip API は両方のテーブルに表示されます。

フラグメントメタデータクォータの消費

API	リクエストごとに消費されるクォータポイントの数	共有クォータ (N)
ListFragments	MaxResults パラメータの値	ストリームあたり 1 秒あたり 10,000 クォータポイント [h]
GetClip	作成されるクリップ内のフラグメントの数	
GetHLSMediaPlaylist	MaxMediaPlaylistFragmentResults パラメータの値	
GetDASHManifest	MaxManifestFragmentResults パラメータの値	
GetImages	値 400 + リクエストされたイメージの最大数	

フラグメントメディアクォータの消費

API	リクエストごとに消費されるクォータポイントの数	共有クォータ (N)
GetMediaForFragmentList	Fragments パラメータのフラグメントの数	ストリームあたり 1 秒あたり 500 クォータポイント [h]
GetClip	作成されるクリップ内のフラグメントの数	
GetMP4MediaFragment	1	
GetTSFragment	1	
GetImages	リクエストされたイメージの最大数	

例えば、1 秒あたり 500 フラグメントメディアのクォータの場合、特定のストリームについて次のような呼び出しパターンがサポートされています。

- 各クリップに 100 個のフラグメントの GetClip に 1 秒あたり 5 個のリクエスト。
- 各クリップに 5 個のフラグメントの GetClip に 1 秒あたり 100 個のリクエスト。
- 各クリップに 100 個のフラグメントの GetClip に 1 秒あたり 2 個のリクエスト、および、各クリップの GetMediaForFragmentList に 1 秒あたり 3 個のリクエスト。
- GetMP4MediaFragment に 1 秒あたり 400 個のリクエスト、および GetTSFragment に 1 秒あたり 100 個のリクエスト。

これらのクォータは、ストリームごとにサポートできる HLS および MPEG-DASH セッションの数に関して重要な意味を持ちます。メディアプレーヤーが一度に使用できる HLS セッションと DASH セッションの数に制限はありません。したがって、再生アプリケーションで同時に使用できるセッションが多すぎないことが重要です。次の 2 つの例は、サポートできる同時再生セッションの数を決定する方法を示しています。

例 1: ライブストリーミング

1 秒の再生時間フラグメント、オーディオトラックとビデオトラック、を 5

MaxMediaPlaylistFragmentResults に設定して HLS を使用するライブストリーミングシナリオでは、メディアプレーヤーは通常、1 秒 GetHLSMediaPlaylist あたり 2 回の呼び出しを行います。

す。1つの呼び出しは最新のビデオメタデータ用で、もう1つの呼び出しは対応するオーディオメタデータ用です。2つの呼び出しは、それぞれ5つのフラグメントメタデータクォータポイントを消費します。また、1秒GetMP4MediaFragmentあたりを2回呼び出します。1回は最新のビデオを、もう1回は対応するオーディオを呼び出します。各呼び出しは1つのフラグメントメディアトークンを消費するため、合計2つのトークンが消費されます。

このシナリオでは、最大250の同時再生セッションをサポートできます。250セッションの場合、このシナリオでは、1秒あたり2,500のフラグメントメタデータクォータポイント(10,000クォータを大幅に下回る)および1秒あたり500のフラグメントメディアクォータポイントが消費されます。

例 2: オンデマンド再生

音声と動画のトラックがあり、MaxManifestFragmentResultsは1000に設定し、MPEG-DASHを使用した過去のイベントをオンデマンド再生するシナリオでは、メディアプレイヤーは通常GetDASHManifestをセッションの開始時に1回呼び出し(フラグメントメタデータクォータポイントを1,000消費)、GetMP4MediaFragmentをすべてのフラグメントがロードされるまで1秒あたり最大5回の割合で呼び出します(フラグメントメディアクォータポイントを5消費)。このシナリオでは、1秒あたり最大10個の新しいセッションを開始でき(ちょうど1秒あたり10,000のフラグメントメタデータクォータ)、最大100セッションでフラグメントメディアを1秒あたり5のレートでアクティブにロードできます(ちょうど1秒あたり500のフラグメントメディアクォータ)。

フラグメントメタデータポイント、およびフラグメントメディアクォータポイントの消費量をモニタリングするには、ArchivedFragmentsConsumed.MetadataとArchivedFragmentsConsumed.Mediaをそれぞれ使用します。モニタリングの詳細については、「[the section called “モニタリング”](#)」を参照してください。

ストリーミングメタデータサービスクォータ

Kinesis ビデオストリームへのストリーミングメタデータの追加には、次のサービスクォータが適用されます。

- フラグメントの先頭には10個までメタデータ項目を追加できます。
- フラグメントのメタデータの名前の長さは、最大128バイトです。
- フラグメントのメタデータの値の長さは、最大256バイトです。
- フラグメントメタデータ名は文字列AWS「」で始めることはできません。そのようなメタデータ項目が追加された場合は、PICのputFragmentMetadataメソッドはSTATUS_INVALID_METADATA_NAMEエラー(エラーコード: 0x52000077)を返します。アプリケーションは、エラーを無視する(PICはメタデータ項目を追加しません)か、エラーに対応します。

プロデューサー SDK クォータ

次の表に、SDK の値の現在のクォータを示します。詳細については「[Kinesis Video Streams へのアップロード](#)」を参照してください。

Note

これらの値を設定する前に、入力値を検証する必要があります。SDK ではこれらの制限は検証されません。制限を超えた場合はランタイムエラーが表示されます。

値	制限	注意事項
最大ストリームカウント	128	プロデューサーオブジェクトが作成できるストリームの最大数です。これはソフト制限です (増加をリクエストできます)。これにより、プロデューサーが誤って再帰的にストリームを作成しなくなります。
デバイス名の最大の長さ	128 文字	
最大タグカウント	ストリームあたり 50	
ストリーム名の最大の長さ	256 文字	
最小ストレージサイズ	10 MiB = 10 x 1024 x 1024 バイト	
最大ストレージサイズ	10 GiB = 10 x 1024 x 1024 x 1024 バイト	
最大のルートディレクトリの長さ	4,096 文字	
最大の auth info の長さ	10,000 バイト	

値	制限	注意事項
最大の URI 文字列の長さ	10,000 文字	
タグ名の最大の長さ	128 文字	
タグ値の最大の長さ	1,024 文字	
最小のセキュリティトークン期間	30 秒	
セキュリティトークン猶予期間	40 分	指定された期間が長い場合は、この値に制限されます。
保持期間	0 または 1 時間以上	0 は保持なしを示します。
最小クラスター期間	1 秒	この値は、100 NS ユニットに指定されます (SDK 標準)。
最大クラスター期間	30 秒	この値は、100 NS ユニットに指定されます (SDK 標準)。バックエンド API は、より短いクラスター期間を適用できます。
ラグメントの最大サイズ	50MB	詳細については、「 Amazon Kinesis Video Streams サービススクォータ 」を参照してください。
最大フラグメント期間	20 秒	詳細については、「 Amazon Kinesis Video Streams サービススクォータ 」を参照してください。

値	制限	注意事項
最大接続時間	45 分	この時間後にバックエンドは接続を終了します。SDK はトークンをローテーションして、この時間内で新しい接続を確立します。
ACK セグメントの最大の長さ	1,024 文字	ACK パーサー関数に送信される確認セグメントの最大の長さ。
コンテンツタイプ文字列の最大の長さ	128 文字	
コーデック ID 文字列の最大の長さ	32 文字	
トラック名文字列の最大の長さ	32 文字	
コーデックプライベートデータの最大の長さ	1 MiB = 1 x 1024 x 1024 バイト	
タイムコードスケール値の最小の長さ	100 ns	生成した MKV クラスターのフレームタイムスタンプを表す最小のタイムコードスケール値です。この値は、100 NS 増加して指定されます (SDK 標準)。
タイムコードスケール値の最大の長さ	1 秒	生成した MKV クラスターのフレームタイムスタンプを表す最大のタイムコードスケール値です。この値は、100 NS 増加して指定されます (SDK 標準)。

値	制限	注意事項
コンテンツビュー項目の最小数	10	
最小のバッファ時間	20 秒	この値は、100 NS 増加して指定されます (SDK 標準)。
アップデートバージョンの最大の長さ	128 文字	
ARN の最大の長さ	1024 文字	
フラグメントシーケンスの最大の長さ	128 文字	
最大保持期間	10 年	

Amazon Kinesis Video Streams 階層型ストレージ

Amazon Kinesis Video Streams 階層型ストレージを使用すると、予想される消費ニーズ、データアクセス頻度、保持期間に基づいて、メディアの保存と消費に関連するコストを最適化できます。KVS には、コストとパフォーマンスの要件のバランスを取るのに役立つ 2 つのストレージ層があります。

トピック

- [概要:](#)
- [主なコンセプト](#)
- [階層型ストレージのセットアップ](#)
- [ベストプラクティス](#)

概要:

Kinesis Video Streams には 2 つのストレージ階層があります。

- Kinesis Video Streams Standard (ホット) 階層 – 頻繁にアクセスされるメディアに対して、レイテンシーを最小限に抑えながら即時アクセスを提供します。

- Kinesis Video Streams ウォーム階層 – メディアの保存期間が長く、アクセス頻度は低くても、必要に応じて迅速なアクセスを必要とするユースケースにコスト効率的です。

主なコンセプト

これらの概念を理解することは、最適なストレージポリシーを設定するのに役立ちます。

- ストレージコスト – ウォーム階層はホット階層よりも安価です。
- 取り込みコスト – 1000 フラグメントあたりのウォーム階層料金と、取り込まれたメディア 1 GB あたりのホット階層料金。
- 最小ストレージ期間 – ウォーム階層ストレージは、最小ストレージ期間である 30 日間課金されます。この 30 日前にメディアを削除または移動した場合、通常のストレージ使用料と、最小ストレージ期間の残りの期間に対する按分計算された料金が発生します。最小ストレージ期間よりも長く保存されたメディアには、最小料金は発生しません。
- レイテンシー – 取り込みのわずかな増加 (ミリ秒)、メディアがウォーム階層に保存されている場合の消費レイテンシー。

階層型ストレージのセットアップ

ウォーム階層取り込みで新しいストリームを作成する

1. で コンソールを開きます。 <https://console.aws.amazon.com/kinesisvideo/home>
2. [Video streams (ビデオストリーム)] ページで、[Create video stream (ビデオストリームの作成)] を選択します。
3. 新しいビデオストリームの作成ページで、ストリーム名の *YourStreamName* を入力します。
4. カスタム設定を選択します。データ保持期間 (*31 #*など) を設定します。ストレージ階層を *### ###* に設定します。
5. [Create video stream (ビデオストリームの作成)] を選択します。
6. Amazon Kinesis Video Streams がストリームを作成したら、YourStreamName ページで詳細を確認します。

これで、ストリームはメディアをウォーム階層に直接取り込み、レイテンシーを最小限に抑えてすぐにアクセスできるようになります。

ベストプラクティス

- ストレージ階層を選択する前にメディアアクセスパターンを分析する
- 即時アクセスを必要とする頻繁にアクセスされるメディアにホット階層を使用する
- 許容できるわずかなレイテンシーで時折アクセスされるメディアのウォーム階層を設定する
- ストレージおよび取り出しコストを含む総所有コストを考慮する

アカウントをセットアップする

Amazon Kinesis Video Streams を初めて使用する前に、次のタスクを完了してください。

トピック

- [にサインアップする AWS アカウント](#)
- [管理アクセスを持つユーザーを作成する](#)
- [AWS アカウント キーを作成する](#)

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. <https://portal.aws.amazon.com/billing/signup> を開きます。
2. オンラインの手順に従います。

サインアップ手順の一環として、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

にサインアップすると AWS アカウント、AWS アカウントのルートユーザー が作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティベストプラクティスとして、ユーザーに管理アクセス権を割り当て、[ルートユーザーアクセスが必要なタスク](#)の実行にはルートユーザーのみを使用するようにしてください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。 <https://aws.amazon.com/> の [マイアカウント] をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、日常的なタスクにルートユーザーを使用しないように AWS アカウントのルートユーザー、 を保護し AWS IAM Identity Center、 を有効にして管理ユーザーを作成します。

を保護する AWS アカウントのルートユーザー

1. ルートユーザーを選択し、AWS アカウント E メールアドレスを入力して、アカウント所有者 [AWS マネジメントコンソール](#) としてサインインします。次のページでパスワードを入力します。

ルートユーザーを使用してサインインする方法については、「AWS サインイン ユーザーガイド」の「[ルートユーザーとしてサインインする](#)」を参照してください。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

手順については、IAM [ユーザーガイドの AWS アカウント「ルートユーザー \(コンソール\) の仮想 MFA デバイス](#) を有効にする」を参照してください。

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

手順については、「AWS IAM Identity Center ユーザーガイド」の「[AWS IAM Identity Centerの有効化](#)」を参照してください。

2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法的チュートリアルについては、AWS IAM Identity Center 「ユーザーガイド」の「[デフォルトを使用してユーザーアクセスを設定する IAM アイデンティティセンターディレクトリ](#)」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

IAM Identity Center ユーザーを使用してサインインする方法については、AWS サインイン「ユーザーガイド」の [AWS「アクセスポータルにサインインする」](#) を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラクティスに従ったアクセス許可セットを作成します。

手順については、「AWS IAM Identity Center ユーザーガイド」の「[アクセス許可セットを作成する](#)」を参照してください。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

手順については、「AWS IAM Identity Center ユーザーガイド」の「[グループを追加する](#)」を参照してください。

AWS アカウント キーを作成する

Amazon Kinesis Video Streams にプログラムでアクセスするには、AWS アカウント キーが必要です。

AWS アカウント キーを作成するには、以下を実行します。

1. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/iam/> で IAM コンソールを開きます。
2. ナビゲーションバーの [Users] (ユーザー) を選択後、[Administrator] (管理者) ユーザーを選択します。
3. [セキュリティ認証情報] タブを選択し、[アクセスキーの作成] を選択します。
4. [アクセスキー ID] を記録します。[Secret access key] (シークレットアクセスキー) で [Show] (表示) を選択します。[シークレットアクセスキー] を記録します。

利点

Kinesis Video Streams を使用すると、次のような利点があります。

- 何百万ものデバイスに接続してストリーミングする – Kinesis Video Streams を使用して、コンシューマースマートフォン、ドローン、ダッシュカメラなど、何百万ものデバイスからビデオ、オーディオ、その他のデータを接続してストリーミングできます。Kinesis Video Streams プロデューサーライブラリを使用すると、デバイスを設定して、リアルタイムまたは事後メディアアップロードによる安定したストリーミングを行うことができます。
- データの永続的な保存、暗号化とインデックス – カスタムの保持期間でメディアデータを永続的に保管するように Kinesis のビデオストリームを設定できます。Kinesis Video Streams は、プロデューサーが生成したタイムスタンプまたはサービス側のタイムスタンプに基づいて、保存されたデータに対してインデックスも生成します。アプリケーションは、タイムインデックスを使用してストリーム内の指定されたデータを取得できます。
- インフラストラクチャではなくアプリケーションの管理に集中する – Kinesis Video Streams はサーバーレスであるため、セットアップや管理を行うインフラストラクチャはありません。データストリームと消費するアプリケーションの数が増減するため、基盤となるインフラストラクチャのデプロイ、設定、エラスティックスケールアップについて心配する必要はありません。Kinesis Video Streams は、ストリームを管理するために必要なすべての管理や維持を自動的に行うため、インフラストラクチャではなく、アプリケーションに集中することができます。
- データストリームでリアルタイムおよびバッチアプリケーションを構築する – Kinesis Video Streams を使用して、ライブデータストリームで動作するカスタムリアルタイムアプリケーションを構築し、厳密なレイテンシー要件なしで永続的に保持されるデータで動作するバッチまたは 1 回限りのアプリケーションを作成できます。を使用して、オープンソース (Apache MXNet、OpenCV)、自社開発、またはサードパーティーのソリューションなどのカスタムアプリケーションを構築、デプロイ、管理 AWS Marketplace し、ストリームを処理および分析できます。Kinesis Video Streams Get APIs、データをリアルタイムまたはバッチ指向ベースで処理する複数の同時アプリケーションを構築できます。
- データのより安全なストリーミング – Kinesis Video Streams は、サービスを通過するとき、およびデータが保持されるときに、すべてのデータを暗号化します。Kinesis Video Streams は、デバイスからのデータストリーミングに Transport Layer Security (TLS) ベースの暗号化を適用し、AWS Key Management Service () を使用して保管中のすべてのデータを暗号化します AWS KMS。さらに、AWS Identity and Access Management (IAM) を使用してデータへのアクセスを管理できます。
- 従量制料金 – 詳細については、「」を参照してください [AWS 料金見積りツール](#)。

Amazon Kinesis Video Streams の開始方法

このセクションでは、Amazon Kinesis Video Streams で次のタスクを実行する方法を説明します。

- をセットアップ AWS アカウント し、まだ作成していない場合は 管理者を作成します。
- Kinesis ビデオストリームを作成します。
- カメラから Kinesis のビデオストリームにデータを送信し、コンソールでそのメディアを表示します。

Amazon Kinesis Video Streams を初めて使用する場合は、[Kinesis Video Streams: 仕組み](#) 最初に読むことをお勧めします。

Note

開始方法のサンプルに従うと、 に料金は発生しません AWS アカウント。リージョンのデータコストについては、[Amazon Kinesis Video Streams の料金](#)」を参照してください。

トピック

- [Amazon Kinesis ビデオストリームを作成する](#)
- [Amazon Kinesis ビデオストリームにデータを送信する](#)
- [メディアデータの消費](#)

Amazon Kinesis ビデオストリームを作成する

このセクションでは、Kinesis のビデオストリームの作成方法について説明します。

このセクションでは、次の手順を紹介します。

- [the section called “コンソールを使用してビデオストリームを作成する”](#)
- [the section called “を使用してビデオストリームを作成する AWS CLI”](#)

コンソールを使用してビデオストリームを作成する

1. で コンソールを開きます。 <https://console.aws.amazon.com/kinesisvideo/home>

2. [Video streams (ビデオストリーム)] ページで、[Create video stream (ビデオストリームの作成)] を選択します。
3. 新しいビデオストリームの作成ページで、ストリーム名の *YourStreamName* を入力します。デフォルト設定ボタンは選択したままにします。
4. [Create video stream (ビデオストリームの作成)] を選択します。
5. Amazon Kinesis Video Streams がストリームを作成したら、YourStreamName ページで詳細を確認します。

を使用してビデオストリームを作成する AWS CLI

1. AWS CLI がインストールされ、設定されていることを確認します。詳細については、[AWS Command Line Interface](#) ドキュメントを参照してください。
2. AWS CLIで、次の Create-Stream コマンドを実行します。

```
aws kinesisisvideo create-stream --stream-name "YourStreamName" --data-retention-in-hours 24
```

レスポンスは次のようになります。

```
{
  "StreamARN": "arn:aws:kinesisvideo:us-west-2:123456789012:stream/YourStreamName/123456789012"
}
```

Amazon Kinesis ビデオストリームにデータを送信する

このセクションでは、カメラから前のセクションで作成した Kinesis ビデオストリームにメディアデータを送信する方法について説明します。このセクションでは、[C++ プロデューサーライブラリを使用する](#) を [例: Kinesis Video Streams プロデューサー SDK GStreamer プラグイン - kvssink](#) プラグインとして使用します。

このチュートリアルでは、さまざまなオペレーティングシステム上のさまざまなデバイスからメディアを送信するために、Kinesis Video Streams C++ プロデューサーライブラリと、カメラやその他のメディアソースへのアクセスを標準化するオープンソースのメディアフレームワークである [GStreamer](#) を使用します。

トピック

- [SDK とサンプルを構築する](#)
- [サンプルを実行して Kinesis Video Streams にメディアをアップロードする](#)
- [確認オブジェクトを確認する](#)

SDK とサンプルを構築する

SDK とサンプルは、コンピュータまたは で構築できます AWS Cloud9。以下の適切な手順に従ってください。

Build on your computer

[readme ファイル](#)の指示に従って、プロデューサーライブラリとサンプルアプリケーションを構築します。

これには、以下が含まれます。

- 依存関係をインストールする
- リポジトリのクローン作成
- CMake を使用して makefile を生成する
- make を使用したバイナリファイルの構築

Build in AWS Cloud9

Kinesis Video Streams にアップロードするには、次の手順に従います AWS Cloud9。コンピュータに何かをダウンロードする必要はありません。

1. で AWS マネジメントコンソール、を開きます [AWS Cloud9](#)。

環境の作成 を選択します。

2. 環境の作成画面で、以下を完了します。

- 名前 - 新しい環境の名前を入力します。
- プラットフォーム - Ubuntu Server 22.04 LTS を選択します。

他のフィールドはデフォルトの選択のままにすることができます。

3. 環境が作成されたら、Cloud9 IDE 列で Open を選択します。Cloud9

画面の中央下部に が表示されますAdmin:~/environment \$。これは (Amazon EC2) ターミナルです AWS Cloud9。

 Note

ターミナルを誤って閉じた場合は、Window, New Terminal を選択します。

ターミナルで次のコマンドを実行して、ボリュームを 20 GiB に変更します。

- a. スクリプトをダウンロードします。

```
wget https://awsj-iot-handson.s3-ap-northeast-1.amazonaws.com/kvs-workshop/resize_volume.sh
```

- b. スクリプトに実行アクセス許可を付与します。

```
chmod +x resize_volume.sh
```

- c. スクリプトを実行します。

```
./resize_volume.sh
```

4. Advanced Packaging Tool (APT) を使用して、インストールまたは更新できるすべてのソフトウェアに関する最新情報を取得します。

このコマンドはソフトウェア自体を更新しませんが、利用可能な最新バージョンをシステムが把握していることを確認します。

```
sudo apt-get update
```

5. C++ プロデューサー SDK の依存関係をインストールします。

```
sudo apt-get install -y cmake m4 git build-essential pkg-config libssl-dev  
libcurl4-openssl-dev \  
liblog4cplus-dev libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev \  
gstreamer1.0-plugins-base-apps gstreamer1.0-plugins-bad gstreamer1.0-plugins-  
good \  
gstreamer1.0-plugins-ugly gstreamer1.0-tools
```

6. git を使用して C++ プロデューサー SDK のクローンを作成します。

```
git clone https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp.git
```

7. ビルドディレクトリを準備します。

```
cd amazon-kinesis-video-streams-producer-sdk-cpp
mkdir build
cd build
```

8. CMake を使用して makefile を生成します。

```
cmake .. -DBUILD_GSTREAMER_PLUGIN=TRUE -DBUILD_DEPENDENCIES=OFF
```

予想される出力の終わりは次のようになります。

```
-- Build files have been written to: /home/ubuntu/environment/amazon-kinesis-
video-streams-producer-sdk-cpp/build
```

9. make を使用して SDK とサンプルアプリケーションをコンパイルし、最終的な実行可能ファイルを構築します。

```
make
```

予想される出力の終わりは次のようになります。

```
[100%] Linking CXX executable kvs_gstreamer_file_uploader_sample
[100%] Built target kvs_gstreamer_file_uploader_sample
```

10. サンプルファイルが構築されたことを確認します。現在のディレクトリ内のファイルを一覧表示します。

```
ls
```

次のファイルが存在することを確認します。

- kvs_gstreamer_sample
- libgstkvssink.so

11. (オプション) GST_PLUGIN_PATH 環境変数の設定をシェルの起動スクリプトに追加できます。これにより、新しいターミナルセッション中に GST_PLUGIN_PATH が正しく設定されます。では AWS Cloud9、シェルの起動スクリプトは `~/.bashrc`。

次のコマンドを実行して、シェルの起動スクリプトの末尾に コマンドを追加します。

```
echo "export GST_PLUGIN_PATH=~/.environment/amazon-kinesis-video-streams-  
producer-sdk-cpp/build" >> ~/.bashrc
```

次のように入力して、シェルの起動スクリプトを実行します。

```
source ~/.bashrc
```

GST_PLUGIN_PATH が設定されていることを確認します。

```
echo $GST_PLUGIN_PATH
```

出力を正しく設定すると、次の出力が表示されます。出力が空白の場合、環境変数が正しく設定されていません。

```
/home/ubuntu/environment/amazon-kinesis-video-streams-producer-sdk-cpp/build
```

サンプルを実行して Kinesis Video Streams にメディアをアップロードする

サンプルアプリケーションは IMDS 認証情報をサポートしていません。ターミナルで、IAM ユーザーまたはロールの AWS 認証情報と、ストリームがあるリージョンをエクスポートします。

```
export AWS_ACCESS_KEY_ID=YourAccessKey  
export AWS_SECRET_ACCESS_KEY=YourSecretKey  
export AWS_DEFAULT_REGION=YourAWSRegion
```

一時的な AWS 認証情報を使用している場合は、セッショントークンもエクスポートします。

```
export AWS_SESSION_TOKEN=YourSessionToken
```

.mp4 files

サンプル .mp4 ビデオをダウンロードして Kinesis Video Streams にアップロードします。


```
wget https://awsj-iot-handson.s3-ap-northeast-1.amazonaws.com/kvs-workshop/sample.mp4
```

ビデオ仕様:

- 解像度 - 1280 x 720 ピクセル
- フレームレート - 30 フレーム/秒
- 期間 - 14.0 秒
- ビデオエンコーディング - H.264、トラック 1 内
- キーフレーム - 3 秒ごとにフラグメント期間 (写真のグループ (GoP) サイズとも呼ばれます) は 3 秒で、最後のフラグメントは 2 秒です。

以前に作成したストリームの名前で次のコマンドを実行します。ストリームをまだ作成していない場合は、「」を参照してください[the section called “Amazon Kinesis ビデオストリームを作成する”](#)。

```
./kvs_gstreamer_sample YourStreamName ./sample.mp4
```

Sample video from GStreamer

GStreamer を使用してビデオを生成するには、次のコマンドを使用します。

GStreamer プラグインの場所を kvssink GStreamer に伝えます。ビルドディレクトリで、libgstkvssink.so ファイルを含むフォルダへのパスを指定します。

ビルドディレクトリから、次のコマンドを実行します。

```
export GST_PLUGIN_PATH=`pwd`
```

この GStreamer パイプラインは、640 x 480 ピクセルの解像度で 1 秒あたり 10 フレームで実行される標準テストパターンを持つライブテストビデオストリームを生成します。オーバーレイが追加され、現在のシステム日時が表示されます。その後、ビデオは H.264 形式にエンコードされ、キーフレームは最大 10 フレームごとに生成され、フラグメント期間 (写真のグループ (GoP) サイズとも呼ばれます) は 1 秒になります。は H.264 でエンコードされたビデオストリーム kvssink を受け取り、Matroska (MKV) コンテナ形式にパッケージ化して、Kinesis ビデオストリームにアップロードします。

次のコマンドを実行します。

```
gst-launch-1.0 -v videotestsrc is-live=true \  
! video/x-raw,framerate=10/1,width=640,height=480 \  
! clockoverlay time-format="%a %B %d, %Y %I:%M:%S %p" \  
! x264enc bframes=0 key-int-max=10 \  
! h264parse \  
! kvssink stream-name="YourStreamName"
```

GStreamer パイプラインを停止するには、ターミナルウィンドウを選択し、Ctrl+C を押します。

Note

GStreamer プラグインを使用してカメラまたは USB カメラから RTSP ストリームからビデオをストリーミングする方法の詳細については、「」を参照してください [例: Kinesis Video Streams プロデューサー SDK GStreamer プラグイン - kvssink](#)。

確認オブジェクトを確認する

アップロード中、Kinesis Video Streams はアップロードを実行するクライアントに確認オブジェクトを返します。これらはコマンド出力に出力されます。例は次のようになります。

```
{"EventType": "PERSISTED", "FragmentTimecode": 1711124585823, "FragmentNumber": "1234567890123456789"}
```

確認応答の EventType が の場合 PERSISTED、Kinesis Video Streams は取得、分析、長期保存のためにこのメディアのチャンクを永続的に保存して暗号化したことを意味します。

確認の詳細については、「」を参照してください [the section called “PutMedia”](#)。

メディアデータの消費

メディアデータは、コンソールで表示するか、Hypertext Live Streaming (HLS) を使用してストリームからメディアデータを読み取るアプリケーションを作成することで使用できます。

コンソールでメディアを表示する

別のブラウザタブで、を開きます AWS マネジメントコンソール。Kinesis Video Streams Dashboard で、[ビデオストリーム](#)を選択します。

ストリームのリストでストリームの名前を選択します。必要に応じて検索バーを使用します。

メディア再生セクションを展開します。動画がまだアップロードされている場合は、表示されます。アップロードが完了したら、二重左矢印を選択します。

HLS を使用してメディアデータを使用する

HLS を使用して Kinesis ビデオストリームのデータを使用するクライアントアプリケーションを作成できます。HLS でメディアデータを使用するアプリケーションの作成方法については、「[動画の再生](#)」を参照してください。

Kinesis Video Streams へのアップロード

Amazon Kinesis Video Streams プロデューサーライブラリは、Kinesis Video Streams プロデューサー SDK のライブラリのセットです。クライアントはライブラリと SDK を使用して、Kinesis Video Streams に安全に接続し、メディアデータをストリーミングしてコンソールまたはクライアントアプリケーションにリアルタイムで表示するためのオンデバイスアプリケーションを構築します。

メディアデータは次の方法でストリーミングできます。

- リアルタイムで
- 数秒間バッファリングした後
- メディアのアップロード後

Kinesis Video Streams ストリームを作成したら、そのストリームへのデータの送信を開始できます。SDK を使用して、フレームと呼ばれるビデオデータをメディアソースから抽出し、Kinesis Video Streams にアップロードするアプリケーションコードを作成できます。これらのアプリケーションはプロデューサーアプリケーションとも呼ばれます。

プロデューサーライブラリには、次のコンポーネントが含まれています。

- [Kinesis Video Streams プロデューサークライアント](#)
- [Kinesis Video Streams プロデューサーライブラリ](#)

Kinesis Video Streams プロデューサークライアント

Kinesis Video Streams プロデューサークライアントには 1 つの `KinesisVideoClient` クラスが含まれています。このクラスは、メディアソースの管理、ソースからのデータの受信、ストリームのライフサイクルの管理を行います。データはメディアソースから Kinesis Video Streams に流れます。また、Kinesis Video Streams と独自のハードウェアおよびソフトウェア間のやり取りを定義するための `MediaSource` インターフェイスも提供します。

メディアソースはほぼすべてが対象となります。たとえば、カメラのメディアソースまたはマイクのメディアソースを使用できます。メディアソースはオーディオやビデオソースのみには限定されません。たとえば、データログがテキストファイルの場合でも、データのストリームとして送信できます。また、スマートフォンで複数のカメラから同時にデータをストリームすることもできます。

そのほかのソースからデータを取得するには、MediaSource インターフェイスを実装できます。このインターフェイスでは追加のシナリオが可能ですが、ビルトインサポートは提供されません。例えば、次のようなものを Kinesis Video Streams に送信したいとします。

- 診断データストリーム (アプリケーションログとイベントなど)
- 赤外線カメラ、RADAR あるいは深度カメラからのデータ

Kinesis Video Streams は、カメラなどのメディア生成デバイス用の組み込み実装を提供していません。このようなデバイスからデータを抽出するには、カスタムのメディアソース実装によるコードを実装する必要があります。これにより、カスタムメディアソースを KinesisVideoClient に明示的に登録でき、データは Kinesis Video Streams にアップロードされます。

Kinesis Video Streams プロデューサークライアントは、Java および Android アプリケーションで使用できます。詳細については、「[Java プロデューサーライブラリを使用する](#)」および「[Android プロデューサーライブラリを使用する](#)」を参照してください。

Kinesis Video Streams プロデューサーライブラリ

Kinesis Video Streams プロデューサーライブラリは、Kinesis Video Streams プロデューサークライアントに含まれています。このライブラリは、Kinesis Video Streams とより密接に統合することを希望するユーザーが直接使用することもできます。これにより、独自のオペレーティングシステム、ネットワークスタックや制限されたデバイスリソースのデバイスから統合ができるようになります。

Kinesis Video Streams プロデューサーライブラリは、Kinesis Video Streams にストリーミングするためのステートマシンを実装します。これは、独自のトランスポート実装を提供して、各メッセージがこのサービスに行き来するように明示的に指示することが必要なコールバックフックを提供します。

次の理由により、Kinesis Video Streams プロデューサーライブラリを直接使用することもできます。

- アプリケーションを実行するデバイスに Java 仮想マシンがない場合。
- Java 以外の言語でアプリケーションコードを記述する場合。
- メモリや処理能力などの制限により、コードのオーバーヘッド量を減らし、抽象化の最小レベルに制限したい。

現在、Kinesis Video Streams プロデューサーライブラリは Android、C、C++、Java アプリケーションで使用できます。詳細については、以下の関連トピックでサポートされている言語を参照してください。

プロデューサーライブラリとは何かを理解する

[Java プロデューサーライブラリを使用する](#)

[Android プロデューサーライブラリを使用する](#)

[C++ プロデューサーライブラリを使用する](#)

[C プロデューサーライブラリを使用する](#)

[Raspberry Pi で C++ プロデューサー SDK を使用する](#)

Java プロデューサーライブラリを使用する

Amazon Kinesis Video Streams が提供する Java プロデューサーライブラリを使用して、最小限の設定でアプリケーションコードを記述し、デバイスから Kinesis ビデオストリームにメディアデータを送信できます。

以下のステップを実行してコードを Kinesis Video Streams と統合し、アプリケーションが Kinesis ビデオストリームへのデータのストリーミングを開始できるようにします。

1. KinesisVideoClient オブジェクトのインスタンスを作成します。
2. メディアソース情報を指定して MediaSource オブジェクトを作成します。たとえば、カメラのメディアソースを作成する場合、カメラを識別しカメラ使用のエンコードを指定するなどの情報を提供します。

ストリーミングを開始するには、カスタムのメディアソースを作成する必要があります。

3. KinesisVideoClient を使用してメディアソースを登録します。

KinesisVideoClient を使用してメディアソースを登録後、メディアソースでデータが利用可能になると、KinesisVideoClient とデータが呼び出されます。

手順: Java プロデューサー SDK を使用する

この手順では、Java アプリケーションで Kinesis Video Streams Java プロデューサークライアントを使用して Kinesis ビデオストリームにデータを送信する方法を示します。

このステップでは、カメラやマイクなどのメディアソースは必要ありません。代わりに、テスト目的により、このコードは一連のバイトで構成されるサンプルフレームを生成します。カメラやマイクなどの実際のソースからメディアデータを送信する場合に、この同じコードパターンを使用できます。

この手順には、以下のステップが含まれます。

- [コードをダウンロードして設定する](#)
- [コードを記述して調べる](#)
- [コードを実行して検証する](#)

前提条件

Java プロデューサー SDK を設定する前に、次の前提条件を満たしていることを確認してください。

- サンプルコードでは、認証情報プロファイルファイルで設定したプロファイルを指定して AWS、認証情報を指定します。まず、認証情報プロファイルを設定します (まだ設定していない場合)。詳細については、『』の「[開発用の AWS 認証情報とリージョンの設定](#)」を参照してください AWS SDK for Java。

Note

Java の例では、SystemPropertiesCredentialsProvider オブジェクトを使用して認証情報を取得します。プロバイダは aws.accessKeyId および aws.secretKey Java システムプロパティから、この認証情報を取得します。このシステムプロパティを Java 開発環境に設定します。Java システムプロパティを設定する方法についての詳細は、お使いの統合開発環境 (IDE) のドキュメントを参照してください。

- NativeLibraryPath には KinesisVideoProducerJNI ファイル (<https://github.com/aws-labs/amazon-kinesis-video-streams-producer-sdk-cpp> から入手可能) が含まれている必要があります。このファイルのファイル名拡張子は、オペレーティングシステムによって以下のように変化します。
 - KinesisVideoProducerJNI.so: Linux
 - KinesisVideoProducerJNI.dylib: macOS

- KinesisVideoProducerJNI.dll: Windows

Note

macOS、Ubuntu、Windows、および Raspbian 用の構築済みライブラリは、<https://github.com/aws-labs/amazon-kinesis-video-streams-producer-sdk-java> の `src/main/resources/lib` で入手できます。他の環境では、[C++](#) をコンパイルします。

Java プロデューサーライブラリコードをダウンロードして設定する

Java プロデューサーライブラリ手順のこのセクションでは、Java サンプルコードをダウンロードし、プロジェクトを Java IDE にインポートして、ライブラリの場所を設定します。

この例の前提条件およびその他の詳細については、「[Java プロデューサーライブラリの使用](#)」を参照してください。

1. ディレクトリを作成して、GitHub リポジトリのソースコード例からクローンを作成します。

```
git clone https://github.com/aws-labs/amazon-kinesis-video-streams-producer-sdk-java
```

2. 使用する Java 統合開発環境 (IDE) ([Eclipse](#) や [JetBrains IntelliJ IDEA](#) など) を開き、ダウンロードした Apache Maven プロジェクトをインポートします。

- IntelliJ IDEA では: [インポート] を選択します。ダウンロードしたパッケージのルートに含まれる `pom.xml` ファイルに移動します。
- Eclipse では: [ファイル]、[インポート]、[Maven]、[Existing Maven Projects] を選択します。続いて、`kinesis-video-java-demo` ディレクトリに移動します。

詳細については、IDE のドキュメントを参照してください。

3. Java サンプルコードでは、現在の AWS 認証情報を使用します。別の認証情報プロファイルを使用するには、次のコードを `DemoAppMain.java` で見つけます。

```
final KinesisVideoClient kinesisVideoClient = KinesisVideoJavaClientFactory
    .createKinesisVideoClient(
        Regions.US_WEST_2,
        AuthHelper.getSystemPropertiesCredentialsProvider());
```


コードを次に変更します。

```
final KinesisVideoClient kinesisVideoClient = KinesisVideoJavaClientFactory
    .createKinesisVideoClient(
        Regions.US_WEST_2,
        new ProfileCredentialsProvider("credentials-profile-name"));
```

詳細については、<https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/auth/profile/ProfileCredentialsProvider.html> リファレンスで「AWS SDK for Java ProfileCredentialsProvider」を参照してください。

コードを記述して調べる

[Java プロデューサーライブラリ手順](#)のこのセクションでは、前のセクションでダウンロードした Java サンプルコードを記述して調べます。

Java テストアプリケーション ([DemoAppMain](#)) は、次のコードパターンを示します。

- KinesisVideoClient のインスタンスを作成します。
- MediaSource のインスタンスを作成します。
- MediaSource をクライアントと登録します。
- ストリーミングを開始します。を起動MediaSourceすると、クライアントへのデータの送信が開始されます。

詳細については次のセクションで説明します。

KinesisVideoClient のインスタンスを作成する

createKinesisVideoClient オペレーションを呼び出す KinesisVideoClient オブジェクトを作成します。

```
final KinesisVideoClient kinesisVideoClient = KinesisVideoJavaClientFactory
    .createKinesisVideoClient(
        Regions.US_WEST_2,
        AuthHelper.getSystemPropertiesCredentialsProvider());
```

KinesisVideoClient がネットワーク呼び出しを行うには、認証のために認証情報が必要です。SystemPropertiesCredentialsProvider のインスタンスを渡すと、認証情報のデフォルトプロファイルの AWSCredentials を読み込みます。

```
[default]
aws_access_key_id = ABCDEFGHIJKLMOPQRSTU
aws_secret_access_key = AbCd1234EfGh5678IjKl9012MnOp3456QrSt7890
```

MediaSource のインスタンスを作成する

Kinesis のビデオストリームにバイトを送信するには、データを生成する必要があります。Amazon Kinesis Video Streams は MediaSource インターフェイスを提供し、これは、データソースを示します。

例えば、Kinesis Video Streams Java ライブラリは、MediaSource インターフェイスの ImageFileMediaSource 実装を提供します。このクラスが読み込むのは、Kinesis のビデオストリームではなく一連のメディアファイルのデータだけですが、コードのテストに使用することは可能です。

```
final MediaSource bytesMediaSource = createImageFileMediaSource();
```

MediaSource をクライアントに登録する

KinesisVideoClient で作成したメディアソースを再登録すると、クライアントを認識するようになります (そして、クライアントにデータを送信できます)。

```
kinesisVideoClient.registerMediaSource(mediaSource);
```

メディアソースを起動する

メディアソースを起動して、データの生成を開始してクライアントに送信できるようにします。

```
bytesMediaSource.start();
```

リソースをクリーンアップする

メモリリークを回避するには、次の手順を実行してクライアントからメディアソースを登録解除し、クライアントを解放します。

```
try {
    kinesisVideoClient.unregisterMediaSource(mediaSource);
    kinesisVideoClient.free();
} catch (final KinesisVideoException e) {
    throw new RuntimeException(e);
}
```

を使用してキャッシュに項目を追加した場合[CachedInfoMultiAuthServiceCallbacks](#)、次に例を示します。

```
serviceCallbacks.addStreamInfoToCache(streamName, streamInfo);
serviceCallbacks.addStreamingEndpointToCache(streamName, dataEndpoint);
```

完了したらキャッシュをクリアします。

```
serviceCallbacks.removeStreamFromCache(streamName);
```

コードを実行して検証する

Java [プロデューサーライブラリの Java](#) テストハーネスを実行するには、次の手順を実行します。

1. [DemoAppMain] を選択します。
2. [実行]、[Run 'DemoAppMain'] の順に選択します。
3. アプリケーションの JVM 引数に認証情報を追加します。
 - 非一時的な AWS 認証情報の場合: "-Daws.accessKeyId={YourAwsAccessKey} -Daws.secretKey={YourAwsSecretKey} -Djava.library.path={NativeLibraryPath}"
 - 一時的な AWS 認証情報の場合: "-Daws.accessKeyId={YourAwsAccessKey} -Daws.secretKey={YourAwsSecretKey} -Daws.sessionToken={YourAwsSessionToken} -Djava.library.path={NativeLibraryPath}"

4. にサインイン AWS マネジメントコンソール し、[Kinesis Video Streams コンソール](#)を開きます。

[Manage Streams] ページでストリームを選択します。

5. 埋め込みプレーヤーでサンプルビデオが再生されます。フレームが蓄積されビデオが表示されるまでに少し時間がかかることがあります (一般的な帯域幅やプロセッサの状態で最長 10 秒)。

このコード例は、ストリームを作成します。MediaSource としてコードが開始すると、KinesisVideoClient にサンプルフレームの送信を開始します。続いて、クライアントは Kinesis のビデオストリームにデータを送信します。

Android プロデューサーライブラリを使用する

Amazon Kinesis Video Streams が提供する Android プロデューサーライブラリを使用して、最小限の設定でアプリケーションコードを記述し、Android デバイスから Kinesis ビデオストリームにメディアデータを送信できます。

以下のステップを実行してコードを Kinesis Video Streams と統合し、アプリケーションが Kinesis ビデオストリームへのデータのストリーミングを開始できるようにします。

1. KinesisVideoClient オブジェクトのインスタンスを作成します。
2. メディアソース情報を指定して MediaSource オブジェクトを作成します。たとえば、カメラのメディアソースを作成する場合、カメラを識別しカメラ使用のエンコードを指定するなどの情報を提供します。

ストリーミングを開始するには、カスタムのメディアソースを作成する必要があります。

手順: Android プロデューサー SDK を使用する

この手順では、Android アプリケーションで Kinesis Video Streams Android プロデューサークライアントを使用して Kinesis ビデオストリームにデータを送信する方法を示します。

この手順には、以下のステップが含まれます。

- [the section called “前提条件”](#)
- [the section called “コードをダウンロードして設定する”](#)
- [the section called “コードを調べる”](#)

- [the section called “コードを実行して検証する”](#)

前提条件

アプリケーションコードの検査、編集、および実行には、[Android Studio](#) をお勧めします。最新の安定バージョンを使用することをお勧めします。

サンプルコードでは、Amazon Cognito 認証情報を入力します。

Amazon Cognito ユーザープールと ID プールを設定するには、次の手順に従います。

- [ユーザープールを設定する](#)
- [ID プールを設定する](#)

ユーザープールを設定する

ユーザープールをセットアップ

1. [Amazon Cognito コンソール](#)にサインインし、リージョンが正しいことを確認します。
2. 左側のナビゲーションで、ユーザープールを選択します。
3. ユーザープールセクションで、ユーザープールの作成を選択します。
4. 以下のセクションを完了します。
 - a. ステップ 1: サインインエクスペリエンスを設定する - Cognito ユーザープールのサインインオプションセクションで、適切なオプションを選択します。

[次へ] を選択してください。
 - b. ステップ 2: セキュリティ要件を設定する - 適切なオプションを選択します。

[次へ] を選択してください。
 - c. ステップ 3: サインアップエクスペリエンスを設定する - 適切なオプションを選択します。

[次へ] を選択してください。
 - d. ステップ 4: メッセージ配信を設定する - 適切なオプションを選択します。

IAM ロール選択フィールドで、既存のロールを選択するか、新しいロールを作成します。

[次へ] を選択してください。

- e. ステップ 5: アプリを統合する - 適切なオプションを選択します。

初期アプリケーションクライアントフィールドで、機密クライアントを選択します。

[次へ] を選択してください。

- f. ステップ 6: 確認して作成する - 前のセクションの選択内容を確認し、ユーザープールの作成を選択します。

5. ユーザープールページで、先ほど作成したプールを選択します。

ユーザープール ID をコピーし、後で書き留めます。awsconfiguration.json ファイルでは、これは `CognitoUserPool.Default.PoolId`。

6. アプリ統合タブを選択し、ページの下部に移動します。

7. アプリクライアントリストセクションで、先ほど作成したアプリクライアント名を選択します。

クライアント ID をコピーし、後で書き留めます。awsconfiguration.json ファイルでは、これは `CognitoUserPool.Default.AppClientId`。

8. クライアントシークレットを表示し、後で書き留めます。awsconfiguration.json ファイルでは、これは `CognitoUserPool.Default.AppClientSecret`。

ID プールを設定する

ID プールをセットアップ

1. [Amazon Cognito コンソール](#)にサインインし、リージョンが正しいことを確認します。
2. 左側のナビゲーションで、ID プールを選択します。
3. [ID プールを作成] を選択します。
4. ID プールを設定します。

- a. ステップ 1: ID プールの信頼を設定する - 以下のセクションを完了します。

- ユーザーアクセス - 認証されたアクセスを選択する
- 認証された ID ソース - Amazon Cognito ユーザープールを選択する

[次へ] を選択してください。

- b. ステップ 2: アクセス許可を設定する - 認証されたロールセクションで、次のフィールドに入力します。

- IAM ロール - 新しい IAM ロールの作成を選択します
- IAM ロール名 - 名前を入力し、後のステップで書き留めます。

[次へ] を選択してください。

- c. ステップ 3: ID プロバイダーを接続する - ユーザープールの詳細セクションで、次のフィールドに入力します。

- ユーザープール ID - 前に作成したユーザープールを選択します。
- アプリクライアント ID - 前に作成したアプリクライアント ID を選択します。

[次へ] を選択してください。

- d. ステップ 4: プロパティを設定する - ID プール名フィールドに名前を入力します。

[次へ] を選択してください。

- e. ステップ 5: 確認して作成する - 各セクションの選択内容を確認し、ID プールの作成を選択します。

5. ID プールページで、新しい ID プールを選択します。

ID プール ID をコピーし、後で書き留めます。awsconfiguration.json ファイルでは、これは `CredentialsProvider.CognitoIdentity.Default.PoolId`。

6. IAM ロールのアクセス許可を更新します。

- a. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/iam/> で IAM コンソールを開きます。
- b. 左側のナビゲーションで、ロールを選択します。
- c. 上記で作成したロールを見つけて選択します。

 Note

必要に応じて検索バーを使用します。

- d. アタッチされたアクセス許可ポリシーを選択します。

[Edit] (編集) を選択します。

- e. JSON タブを選択し、ポリシーを以下に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cognito-identity:*",
        "kinesisvideo:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

[次へ] を選択してください。

- f. 新しいバージョンがまだ選択されていない場合は、そのバージョンをデフォルトとして設定の横にあるチェックボックスをオンにします。

[変更を保存] を選択します。

Android プロデューサーライブラリコードをダウンロードして設定する

Android プロデューサーライブラリ手順のこのセクションでは、Android サンプルコードをダウンロードし、Android Studio でプロジェクトを開きます。

この例の前提条件およびその他の詳細については、[「Android プロデューサーライブラリ」の使用](#)を参照してください。

1. ディレクトリを作成し、GitHub リポジトリ AWS Mobile SDK for Android から のクローンを作成します。

```
git clone https://github.com/aws-labs/aws-sdk-android-samples
```

2. [Android Studio](#) を開きます。

3. [開く] 画面で、[Open an existing Android Studio project] を選択します。
4. `aws-sdk-android-samples/AmazonKinesisVideoDemoApp` ディレクトリに移動し、[OK] を開始します。
5. `AmazonKinesisVideoDemoApp/src/main/res/raw/awsconfiguration.json` ファイルを開きます。

CredentialsProvider ノードで、[「前提条件」セクション](#)の「ID プールをセットアップするには」の手順から ID プール ID を指定し、AWS リージョン (例: `us-west-2`) を指定します **us-west-2**。

CognitoUserPool ノードで、「前提条件」セクションの「ユーザープールをセットアップするには」からアプリケーションシークレット、アプリケーション ID、プール ID を指定し、を指定します AWS リージョン (例: **us-west-2**)。 <https://docs.aws.amazon.com/kinesisvideostreams/latest/dg/producer-sdk-android.html#producersdk-android-prerequisites>

6. `awsconfiguration.json` ファイルは次のようになります。

```
{
  "Version": "1.0",
  "CredentialsProvider": {
    "CognitoIdentity": {
      "Default": {
        "PoolId": "us-west-2:01234567-89ab-cdef-0123-456789abcdef",
        "Region": "us-west-2"
      }
    }
  },
  "IdentityManager": {
    "Default": {}
  },
  "CognitoUserPool": {
    "Default": {
      "AppClientSecret": "abcdefghijklmnopqrstuvwxyz0123456789abcdefghijklmnopqrstuvwxyz",
      "AppClientId": "0123456789abcdefghijklmnopqrstuvwxyz",
      "PoolId": "us-west-2_qRsTuVwXy",
      "Region": "us-west-2"
    }
  }
}
```

7. リージョンAmazonKinesisVideoDemoApp/src/main/java/com/amazonaws/kinesisvideo/demoapp/KinesisVideoDemoApp.javaで を更新します (次の例では、US_WEST_2 に設定されています)。

```
public class KinesisVideoDemoApp extends Application {  
    public static final String TAG = KinesisVideoDemoApp.class.getSimpleName();  
    public static Regions KINESIS_VIDEO_REGION = Regions.US_WEST_2;
```

AWS リージョン 定数の詳細については、[「リージョン」](#) を参照してください。

コードを調べる

[Android プロデューサーライブラリの手順](#)のこのセクションでは、コード例を調べます。

Android テストアプリケーション (AmazonKinesisVideoDemoApp) は、次のコードパターンを示します。

- KinesisVideoClient のインスタンスを作成します。
- MediaSource のインスタンスを作成します。
- ストリーミングを開始します。を起動するとMediaSource、クライアントへのデータの送信が開始されます。

詳細については次のセクションで説明します。

KinesisVideoClient のインスタンスを作成する

[createKinesisVideoClient](#) オペレーションを呼び出す [KinesisVideoClient](#) オブジェクトを作成します。

```
mKinesisVideoClient = KinesisVideoAndroidClientFactory.createKinesisVideoClient(  
    getActivity(),  
    KinesisVideoDemoApp.KINESIS_VIDEO_REGION,  
    KinesisVideoDemoApp.getCredentialsProvider());
```

KinesisVideoClient がネットワーク呼び出しを行うには、認証のために認証情報が必要です。AWSCredentialsProvider のインスタンスを渡します。これは、前のセクションで変更した awsconfiguration.json ファイルから Amazon Cognito 認証情報を読み込みます。

MediaSource のインスタンスを作成する

Kinesis のビデオストリームにバイトを送信するには、データを生成する必要があります。Amazon Kinesis Video Streams は [MediaSource](#) インターフェイスを提供し、これは、データソースを示します。

例えば、Kinesis Video Streams Android ライブラリは、MediaSource インターフェイスの [AndroidCameraMediaSource](#) 実装を提供します。このクラスは、デバイスのカメラの 1 つからデータを読み取ります。

次のコード例 (「[fragment/StreamConfigurationFragment.java](#)」ファイルから) では、メディアソースの設定が作成されます。

```
private AndroidCameraMediaSourceConfiguration getCurrentConfiguration() {
    return new AndroidCameraMediaSourceConfiguration(
        AndroidCameraMediaSourceConfiguration.builder()
            .withCameraId(mCamerasDropdown.getSelectedItem().getCameraId())

            .withEncodingMimeType(mMimeTypeDropdown.getSelectedItem().getMimeType())

            .withHorizontalResolution(mResolutionDropdown.getSelectedItem().getWidth())

            .withVerticalResolution(mResolutionDropdown.getSelectedItem().getHeight())
                .withCameraFacing(mCamerasDropdown.getSelectedItem().getCameraFacing())
                .withIsEncoderHardwareAccelerated(

mCamerasDropdown.getSelectedItem().isEncoderHardwareAccelerated())
                .withFrameRate(FRAMERATE_20)
                .withRetentionPeriodInHours(RETENTION_PERIOD_48_HOURS)
                .withEncodingBitRate(BITRATE_384_KBPS)
                .withCameraOrientation(-
mCamerasDropdown.getSelectedItem().getCameraOrientation())

            .withNalAdaptationFlags(StreamInfo.NalAdaptationFlags.NAL_ADAPTATION_ANNEXB_CPD_AND_FRAME_NALS)
                .withIsAbsoluteTimecode(false));
}
```

次のコード例 (「[fragment/StreamingFragment.java](#)」ファイルから) では、メディアソースの設定が作成されます。

```
mCameraMediaSource = (AndroidCameraMediaSource) mKinesisVideoClient
    .createMediaSource(mStreamName, mConfiguration);
```

メディアソースを起動する

メディアソースを開始して、データを生成し、それをクライアントに送信できるようにします。次のコード例は [fragment/StreamingFragment.java](#) ファイルからのものです。

```
mCameraMediaSource.start();
```

コードを実行して検証する

Android [プロデューサーライブラリの Android](#) サンプルアプリケーションを実行するには、次の手順を実行します。

1. Android デバイスに接続します。
2. [Run]、[Run]、[Edit configurations...] をクリックします。
3. プラスアイコン (+)、Android アプリを選択します。[名前] フィールドに **AmazonKinesisVideoDemoApp** を入力してください。[Module] プルダウンから、[AmazonKinesisVideoDemoApp] を選択します。[OK] を選択してください。
4. [Run]、[Run] を選択します。
5. [Select a Deployment Target] 画面で、接続されているデバイスを選択し、[OK] を選択します。
6. デバイスの [AWSKinesisVideoDemoApp] アプリケーションで、[Create new account] を選択します。
7. [«1»USERNAME]、[Password]、[Given name]、[Email address]、[Phone number] の値を入力し、[Sign up] を選択します。

Note

これらの値には以下の制約があります。

- パスワード: 大文字と小文字、数字、特殊文字を含む必要があります。これらの制約は、[Amazon Cognito コンソール](#)のユーザープールページで変更できます。

- E メールアドレス: 確認コードを受け取れるように有効なアドレスでなければなりません。
- 電話番号: 次の形式にする必要があります。+<Country code><Number> (例: +12065551212)。

8. E メールで受信したコードを入力し、確認を選択します。[OK] を選択します。
9. 次のページで、デフォルト値のままにし、ストリームを選択します。
10. にサインイン AWS マネジメントコンソール し、米国西部 (オレゴン) リージョンで [Kinesis Video Streams コンソール](#)を開きます。

[Manage Streams] ページで [demo-stream] を選択します。

11. 埋め込みプレーヤーでストリーミングビデオが再生されます。フレームが蓄積されビデオが表示されるまでに少し時間がかかることがあります (一般的な帯域幅やプロセッサの状態で最長 10 秒)。

Note

デバイスの画面が回転された場合 (縦向きから横向きへなど)、アプリケーションはビデオのストリーミングを停止します。

このコード例は、ストリームを作成します。MediaSource としてコードが開始すると、カメラから KinesisVideoClient にフレームが送信を開始します。クライアントは、データを [demo-stream] (デモストリーム) という名前の Kinesis のビデオストリームに送信します。

C++ プロデューサーライブラリを使用する

Amazon Kinesis Video Streams が提供する C++ プロデューサーライブラリを使用して、デバイスから Kinesis ビデオストリームにメディアデータを送信するアプリケーションコードを記述できます。

オブジェクトモデル

C++ ライブラリには、Kinesis のビデオストリームへのデータ送信を管理するために次のオブジェクトが用意されています。

- KinesisVideoProducer: メディアソースと AWS 認証情報に関する情報が含まれ、Kinesis Video Streams イベントを報告するコールバックを維持します。

- **KinesisVideoStream**: Kinesis のビデオストリームを表します。名前、データ保持期間、メディアコンテンツタイプなど、ビデオストリームのパラメータに関する情報が含まれます。

ストリームにメディアを配置する

C++ ライブラリが提供するメソッド (など `PutFrame`) を使用して、**KinesisVideoStream** オブジェクトにデータを配置できます。ライブラリは、データの内部状態も管理します。タスクには以下が含まれる場合があります。

- 認証を実行する。
- ネットワークレイテンシーを監視する。レイテンシーが長すぎると、フレームが停止される場合があります。
- 進行中のストリーミングのステータスを追跡する。

コールバックインターフェイス

このレイヤーでは、一連のコールバックインターフェイスを表示し、アプリケーションレイヤーとやり取りできるようにします。これらのコールバックインターフェイスには以下が含まれます。

- サービスコールバックインターフェイス (**CallbackProvider**): ライブラリは、ストリームの作成、ストリームの説明の取得、ストリームの削除時に、このインターフェイスを通じて取得したイベントを呼び出します。
- クライアント対応状態または低ストレージイベントインターフェイス (**ClientCallbackProvider**): ライブラリは、クライアントの準備ができたとき、または使用可能なストレージまたはメモリが不足していることを検出したときに、このインターフェイスでイベントを呼び出します。
- ストリームイベントコールバックインターフェイス (**StreamCallbackProvider**): ライブラリは、準備完了状態になったストリーム、ドロップフレーム、ストリームエラーなどのストリームイベントが発生したときに、このインターフェイスでイベントを呼び出します。

Kinesis Video Streams には、これらのインターフェイス用のデフォルト実装が用意されています。独自のカスタム実装を提供することもできます。例えば、カスタムネットワークングロジックが必要な場合や、低ストレージ状態をユーザーインターフェイスに表示する場合などです。

プロデューサーライブラリのコールバックの詳細については、「」を参照してください [プロデューサー SDK コールバック](#)。

手順: C++ プロデューサー SDK を使用する

この手順では、C++ アプリケーションで Kinesis Video Streams クライアントおよびメディアソースを使用してデータを Kinesis のビデオストリームに送信する方法について説明します。

この手順には、以下のステップが含まれます。

前提条件

C++ プロデューサー SDK を設定する前に、次の前提条件を満たしていることを確認してください。

- 認証情報: サンプルコードでは、認証情報プロファイルファイルで AWS 設定したプロファイルを指定して、認証情報を指定します。まず、認証情報プロファイルを設定します (まだ設定していない場合)。

詳細については、「[開発用の AWS 認証情報とリージョンの設定](#)」を参照してください。

- 証明書ストアの統合: Kinesis Video Streams プロデューサーライブラリは、呼び出すサービスとの信頼を確立する必要があります。これは、パブリック証明書ストアで認証機関 (CAs) を検証することによって行われます。Linux ベースのモデルの場合、このストアは /etc/ssl/ ディレクトリにあります。

以下の場所から証明書ストアに、証明書をダウンロードしてください。

<https://www.amazontrust.com/repository/SFSRootCAG2.pem>

- macOS 用の次のビルド依存関係をインストールします。
 - [Autoconf 2.69](#) (ライセンス GPLv3+/Autoconf: GNU GPL バージョン 3 以降)
 - [CMake 3.または 3.8](#)
 - [Pkg-Config](#)
 - xCode (macOS) / clang / gcc (xcode-select バージョン 2347)
 - Java Development Kit (JDK) (Java JNI コンパイル用)
 - [Lib-Pkg](#)
- Ubuntu に次のビルド依存関係をインストールします。
 - Git: `sudo apt install git`
 - [CMake](#): `sudo apt install cmake`
 - G++: `sudo apt install g++`
 - pkg-config: `sudo apt install pkg-config`

- OpenJDK: `sudo apt install openjdk-8-jdk`

 Note

これは、Java ネイティブインターフェイス (JNI) を構築している場合にのみ必要です。

- JAVA_HOME 環境変数を設定します: `export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/`

C++ プロデューサーライブラリコードをダウンロードして設定する

C++ プロデューサーライブラリのダウンロードおよび設定方法については、「[Amazon Kinesis Video Streams CPP Producer, GStreamer Plugin and JNI](#)」を参照してください。

この例の前提条件と詳細については、「」を参照してください[the section called “C++”](#)。

CMake 引数

以下は、C++ プロデューサー SDK 固有の CMake 引数のリファレンステーブルです。[標準の CMake オプション](#)を CMake に渡すこともできます。

 Important

これらはすべてオプションです。

特定の機能を含めるか除外するためのフラグ

CMake 引数	タイプ	デフォルト	説明
BUILD_DEPENDENCIES	ブール値	ON	ソースから依存関係を構築します。それ以外の場合は、システムに既にインストールされている依存関係を使用します。必要な依存関係のいずれかが見つからな

CMake 引数	タイプ	デフォルト	説明
			い場合は、エラーが返されます。
BUILD_GSTREAMER_PLUGIN	ブール値	VOFF	kvssink GStreamer プラグインを構築します。
BUILD_JNI	ブール値	VOFF	Java ランタイム環境からこのコードを呼び出すことができるように Java ネイティブインターフェイス (JNI) を構築します。
ALIGNED_MEMORY_MODEL	ブール値	VOFF	メモリ割り当てを 8 バイトの境界に合わせる必要がある場合。一部のアーキテクチャでは、整合性のないメモリアクセスが許可されません。
CONSTRAINED_DEVICE	ブール値	VOFF	Windows 以外のみ。ON の場合、スレッドスタックのサイズを に設定します 0.5 MiB。 Alpine Linux ビルドに必要です。それ以外の場合は、オペレーティングシステムのデフォルトが使用されます。

CMake 引数	タイプ	デフォルト	説明
BUILD_STATIC	ブール値	VOFF	共有 (OFF)、または 静的 (ON) としてライブラリと実行可能ファイルを構築します。
ADD_MUCLIBC	ブール値	VOFF	組み込みシステム用に設計された小さな C 標準ライブラリである標準 C ライブラリの代わりに uClibc にリンクします。
OPEN_SRC_INSTALL_PREFIX	String	../オープンソース/ローカル	ソースから構築する場合にオープンソースの依存関係をインストールする場所。

クロスコンパイルのフラグ

Important

ターゲットマシンとホストマシンの CPU アーキテクチャが異なる場合は、これらを設定します。

CMake 引数	タイプ	デフォルト	説明
BUILD_LOG4CPLUS_HOST	String	""	指定された CPU アーキテクチャの log4cplus 依存関係を構築します。設定されていない場合、log4cplus はホストマシンの CPU アーキテクチャを自

CMake 引数	タイプ	デフォルト	説明
			動検出して使用します。
BUILD_OPENSSL_PLATFORM	String	""	指定された CPU アーキテクチャの OpenSSL 依存関係を構築します。設定されていない場合、OpenSSL はホストマシンの CPU アーキテクチャを自動検出して使用します。

テストに関連するフラグ

CMake 引数	タイプ	デフォルト	説明
BUILD_TEST	ブール値	VOFF	ユニットテストと統合テストを構築します。すべてのテストを実行するには、ビルドディレクトリ ./tst/producerTest から実行します。AWS テストを実行するには認証情報が必要です。
CODE_COVERAGE	ブール値	VOFF	GNU/Clang コンパイラでのみ使用できます。 gcov とレポート生成を使用してコー

CMake 引数	タイプ	デフォルト	説明
			ドカバレッジ収集を有効にします。
COMPILER_WARNINGS	ブール値	VOFF	GNU/Clang コンパイラでのみ使用できます。すべてのコンパイラ警告を有効にします。
ADDRESS_SANITIZER	ブール値	VOFF	GNU/Clang コンパイラでのみ使用できます。 AddressSanitizer を使用してビルドします。
MEMORY_SANITIZER	ブール値	VOFF	GNU/Clang コンパイラでのみ使用できます。 MemorySanitizer を使用してビルドします。
THREAD_SANITIZER	ブール値	VOFF	GNU/Clang コンパイラでのみ使用できます。 ThreadSanitizer を使用して構築します。
UNDEFINED_BEHAVIOR_SANITIZER	ブール値	VOFF	GNU/Clang コンパイラでのみ使用できます。 UndefinedBehaviorSanitizer を使用して構築します。

これらの CMake 引数を使用するには、`cmake ..` コマンドの後に `-Dkey=value` ペアのスペース区切りリストとして渡します。例えば、次のようになります。

```
cmake .. -DBUILD_GSTREAMER_PLUGIN=ON -DBUILD_DEPENDENCIES=OFF -  
DALIGNED_MEMORY_MODEL=ON
```

CMake は、\$PATH変数に従ってコンパイラツールチェーンを探します。CMake を実行する前に、CCおよび CXX環境変数を設定して、クロスコンパイルに使用するツールチェーンを明示的に設定します。

コードを記述して調べる

このセクションでは[the section called “C++”](#)、C++ テストハーネス (tst/ProducerTestFixture.h およびその他のファイル) のコードを調べます。このコードは前のセクションでダウンロードしたものです。

プラットフォームに依存しない C++ 例では、次のコーディングパターンを示します。

- Kinesis Video Streams にアクセスするために、KinesisVideoProducer のインスタンスを作成します。
- KinesisVideoStream のインスタンスを作成します。これにより、同じ名前のストリームがまだ存在しない AWS アカウント 場合、に Kinesis ビデオストリームが作成されます。
- データのフレームをストリームに送信する準備ができたなら、そのたびに putFrame を KinesisVideoStream で呼び出します。

以下のセクションでは、このコーディングパターンについて詳しく説明します。

KinesisVideoProducer のインスタンスを作成する

KinesisVideoProducer::createSync メソッドを呼び出して、KinesisVideoProducer オブジェクトを作成します。次の例では、KinesisVideoProducer を ProducerTestFixture.hファイルに作成します。

```
kinesis_video_producer_ = KinesisVideoProducer::createSync(move(device_provider_),  
    move(client_callback_provider_),  
    move(stream_callback_provider_),  
    move(credential_provider_),  
    defaultRegion_);
```

createSync メソッドは以下のパラメータを使用します。

- DeviceInfoProvider オブジェクト。デバイスまたはストレージ設定に関する情報を含む DeviceInfo オブジェクトを返します。

Note

deviceInfo.storageInfo.storageSize パラメータを使用してコンテンツストアのサイズを設定します。コンテンツストリームは、コンテンツストアを共有します。ストレージサイズの要件を確認するには、平均フレームサイズに、すべてのストリームの最大継続時間に格納されたフレーム数を乗算します。次に、1.2 を掛けてデフラグメンテーションに合わせます。たとえば、アプリケーションの設定が次のとおりであるとします。

- 3 つのストリーム
- 3 分の最大継続時間
- 各ストリームは 30 フレーム/秒 (FPS)
- 各フレームのサイズは 10,000 KB

このアプリケーションのコンテンツストアの要件は、3 (ストリーム) x 3 (分) x 60 (1 分あたり秒) x 10,000 (kb) x 1.2 (デフラグメント許容量) = 194.4 Mb ~ 200 Mb です。

- ClientCallbackProvider オブジェクト。クライアント固有のイベントを報告する関数ポインタを返します。
- StreamCallbackProvider オブジェクト。ストリーム固有のイベントが発生したときにコールバックされる関数ポインタを返します。
- 認証情報環境変数へのアクセス AWS を提供する CredentialProvider オブジェクト。
- (AWS リージョン 「us-west-2」)。サービスエンドポイントはリージョンから決定されます。

KinesisVideoStream のインスタンスを作成する

StreamDefinition パラメータを指定して KinesisVideoProducer::CreateStream メソッドを呼び出すことで、KinesisVideoStream オブジェクトを作成します。この例では、トラックタイプをビデオ、トラック ID を 1 として、ProducerTestFixture.h ファイルで KinesisVideoStream を作成します。

```
auto stream_definition = make_unique<StreamDefinition>(stream_name,
                                                         hours(2),
                                                         tags,
                                                         "",
                                                         STREAMING_TYPE_REALTIME,
```

```
        "video/h264",  
        milliseconds::zero(),  
        seconds(2),  
        milliseconds(1),  
        true,  
        true,  
        true);  
return kinesis_video_producer_->createStream(move(stream_definition));
```

StreamDefinition オブジェクトには以下のフィールドがあります。

- ストリーム名。
- データ保持期間。
- ストリーム用タグ。コンシューマーアプリケーションでこれらのタグを使用すると、適切なストリームの検索や、ストリームに関する詳細情報を取得できます。タグは、AWS マネジメントコンソールで表示することもできます。
- AWS KMS ストリームの暗号化キー。詳細については、「[the section called “データ保護”](#)」を参照してください。
- ストリーミングタイプ。現在、有効な値は STREAMING_TYPE_REALTIME のみです。
- メディアコンテンツタイプ。
- メディアレイテンシー。この値は現在使用されていないため、0 に設定する必要があります。
- 各フラグメントの再生時間。
- メディアのタイムコードスケール。
- メディアがキーフレームを使用して断片化するかどうか。
- メディアがタイムコードを使用するかどうか。
- メディアが絶対フラグメントタイムを使用するかどうか。

Kinesis ビデオストリームにオーディオトラックを追加する

の addTrack メソッドを使用して、オーディオトラックの詳細をビデオトラックストリーム定義に追加できます StreamDefinition。

```
stream_definition->addTrack(DEFAULT_AUDIO_TRACKID, DEFAULT_AUDIO_TRACK_NAME,  
    DEFAULT_AUDIO_CODEC_ID, MKV_TRACK_INFO_TYPE_AUDIO);
```

addTrack メソッドには、次のパラメータが必要です。

- トラック ID (オーディオ用)。これは一意であり、ゼロ以外の値である必要があります。
- ユーザー定義のトラック名 (オーディオトラックの「オーディオ」など)。
- このトラックのコーデック ID (オーディオトラック「A_AAC」など)。
- トラックタイプ (たとえば、オーディオには MKV_TRACK_INFO_TYPE_AUDIO の列挙値を使用します)。

オーディオトラック用のコーデックプライベートデータがある場合は、addTrack 関数を呼び出すときに、このデータを渡すことができます。KinesisVideoStream で start メソッドを呼び出し中に、KinesisVideoStream オブジェクトの作成後、コーデックプライベートデータを送信することもできます。

Kinesis ビデオストリームにフレームを配置する

KinesisVideoStream::putFrame を使用してメディアを Kinesis のビデオストリームに挿入し、ヘッダーとメディアデータを含む Frame オブジェクトに渡します。この例では、ProducerApiTest.cpp ファイル内の putFrame を呼び出します。

```
frame.duration = FRAME_DURATION_IN_MICROS * HUNDREDS_OF_NANOS_IN_A_MICROSECOND;
frame.size = sizeof(frameBuffer_);
frame.frameData = frameBuffer_;
memset(frame.frameData, 0x55, frame.size);

while (!stop_producer_) {
    // Produce frames
    timestamp = std::chrono::duration_cast<std::chrono::nanoseconds>(
        std::chrono::system_clock::now().time_since_epoch()).count() /
    DEFAULT_TIME_UNIT_IN_NANOS;
    frame.index = index++;
    frame.decodingTs = timestamp;
    frame.presentationTs = timestamp;

    // Key frame every 50th
    frame.flags = (frame.index % 50 == 0) ? FRAME_FLAG_KEY_FRAME : FRAME_FLAG_NONE;
    ...

    EXPECT_TRUE(kinesis_video_stream->putFrame(frame));
}
```


Note

前述の C++ プロデューサーの例では、テストデータのバッファを送信します。実際のアプリケーションでは、フレームバッファとフレームのサイズはメディアソース (カメラなど) のフレームデータから取得してください。

Frame オブジェクトには以下のフィールドがあります。

- フレームインデックス。これは一定間隔で増加する値にする必要があります。
- フレームに関連付けられているフラグ。たとえば、エンコーダーがキーフレームを生成するように設定されている場合、このフレームは `FRAME_FLAG_KEY_FRAME` フラグに割り当てられます。
- デコードタイムスタンプ。
- プレゼンテーションのタイムスタンプ。
- フレームの時間 (100 ns 単位)。
- フレームのサイズ (バイト単位)。
- フレームデータ。

フレームの形式の詳細については、「[the section called “データモデル”](#)」を参照してください。

KinesisVideoStream の特定のトラックに KinesisVideoStream を配置する

PutFrameHelper クラスを使用して、フレームデータを特定のトラックに配置することができます。まず、`getFrameDataBuffer` を呼び出して、`KinesisVideoFrame` データを埋めるために事前に割り当てられたバッファの 1 つへのポインタを取得します。次に、`putFrameMultiTrack` を呼び出して、`bool` 値 `KinesisVideoFrame` とともに送信し、フレームデータのタイプを指定できます。ビデオデータの場合は `true`、フレームにオーディオデータが含まれている場合は `false` を使用します。この `putFrameMultiTrack` メソッドはキューイングメカニズムを使用して、MKV フラグメントが一定間隔で増加するフレームタイムスタンプを維持し、2 つのフラグメントが重複しないようにします。たとえば、フラグメントの最初のフレームの MKV タイムスタンプは、常に前のフラグメントの最後のフレームの MKV タイムスタンプより大きい必要があります。

PutFrameHelper には以下のフィールドがあります。

- キュー内のオーディオフレームの最大数。
- キュー内のビデオフレームの最大数。

- 1 つのオーディオフレームに割り当てるサイズ。
- 1 つのビデオフレームに割り当てるサイズ。

アクセスメトリクスとメトリクスのログ記録

C++ プロデューサー SDK には、メトリクスとメトリクスログ記録の機能が含まれています。

`getKinesisVideoMetrics` および `getKinesisVideoStreamMetrics` API オペレーションを使用すると、Kinesis Video Streams とアクティブなストリームに関する情報を取得できます。

以下は `kinesis-video-pic/src/client/include/com/amazonaws/kinesis/video/client/Include.h` ファイルにあるコードです。

```
/**
 * Gets information about the storage availability.
 *
 * @param 1 CLIENT_HANDLE - the client object handle.
 * @param 2 PKinesisVideoMetrics - OUT - Kinesis Video metrics to be filled.
 *
 * @return Status of the function call.
 */
PUBLIC_API STATUS getKinesisVideoMetrics(CLIENT_HANDLE, PKinesisVideoMetrics);

/**
 * Gets information about the stream content view.
 *
 * @param 1 STREAM_HANDLE - the stream object handle.
 * @param 2 PStreamMetrics - Stream metrics to fill.
 *
 * @return Status of the function call.
 */
PUBLIC_API STATUS getKinesisVideoStreamMetrics(STREAM_HANDLE, PStreamMetrics);
```

`getKinesisVideoMetrics` によって入力される `PClientMetrics` オブジェクトには、以下の情報が含まれています。

- `contentStoreSize`: コンテンツストア (ストリーミングデータの保存に使用されるメモリ) 全体のサイズ (バイト単位)。
- `contentStoreAvailableSize`: コンテンツストアで使用可能なメモリ、バイト単位。
- `contentStoreAllocatedSize`: コンテンツストアに割り当てられたメモリ。

- `totalContentViewsSize`: コンテンツビューに使用される合計メモリ。コンテンツビューは、コンテンツストア内の情報の一連のインデックスです。
- `totalFrameRate`: 1 秒あたりのフレーム数を、すべてのアクティブなストリームで集計した数。
- `totalTransferRate`: すべてのストリームで送信される合計ビット数/秒 (bps)。

`getKinesisVideoStreamMetrics` によって入力される `PStreamMetrics` オブジェクトには、以下の情報が含まれています。

- `currentViewDuration`: コンテンツビューの先頭 (フレームがエンコードされている場合) と現在の位置 (フレームデータが Kinesis Video Streams に送信される場合) の 100 ns 単位の差。
- `overallViewDuration`: コンテンツビューの先頭 (フレームのエンコード時) と末尾 (コンテンツビューに割り当てられた合計容量を超過したこと、または Kinesis Video Streams から `PersistedAck` メッセージを受信し、既知の永続的なフレームがフラッシュされたことが原因で、フレームがメモリからフラッシュされた時点) の差異 (100 ns 単位)。
- `currentViewSize`: コンテンツビューの先頭 (フレームがエンコードされた時点) から、現在の位置 (フレームデータが Kinesis Video Streams に送信される時点) までのサイズ (バイト単位)。
- `overallViewSize`: コンテンツビューの合計サイズ (バイト単位)。
- `currentFrameRate`: 最後に測定されたストリームのレート (fps 単位)。
- `currentTransferRate`: 最後に測定されたストリームのレート (bps 単位)。

Teardown:

バッファ内の残りのバイトを送信し、ACK を待機する場合、`stopSync` を使用できます。

```
kinesis_video_stream->stopSync();
```

または、`stop` を呼び出してストリーミングを終了できます。

```
kinesis_video_stream->stop();
```

ストリームを停止したら、次の API を呼び出すことでストリームを解放できます。

```
kinesis_video_producer_->freeStream(kinesis_video_stream);
```

コードを実行して検証する

のコードを実行して検証するには[the section called “C++”](#)、以下の OS 固有の手順を参照してください。

- [Linux](#)
- [macOS](#)
- [Windows](#)
- [Raspberry Pi OS](#)

ストリームに関連付けられている、PutMedia.IncomingBytes などのメトリクスを Amazon CloudWatch コンソールで監視することで、ストリームのトラフィックをモニタリングすることができます。

C++ プロデューサー SDK を GStreamer プラグインとして使用する

[GStreamer](#) は、モジュラープラグインを組み合わせてカスタムメディアパイプラインを作成するために、複数のカメラとビデオソースで利用される一般的なメディアフレームワークです。Kinesis Video Streams GStreamer プラグインは、既存の GStreamer メディアパイプラインと Kinesis Video Streams の統合を合理化します。

C++ プロデューサー SDK を GStreamer プラグインとして使用方法については、「」を参照してください[例: Kinesis Video Streams プロデューサー SDK GStreamer プラグイン - kvssink](#)。

C++ プロデューサー SDK を Docker コンテナの GStreamer プラグインとして使用する

[GStreamer](#) は、モジュラープラグインを組み合わせてカスタムメディアパイプラインを作成するために、複数のカメラとビデオソースで利用される一般的なメディアフレームワークです。Kinesis Video Streams GStreamer プラグインは、既存の GStreamer メディアパイプラインと Kinesis Video Streams の統合を合理化します。

さらに、[Docker](#) を使用して GStreamer パイプラインを作成すると、Kinesis Video Streams の運用環境が標準化され、アプリケーションの構築と実行が合理化されます。

C++ プロデューサー SDK を Docker コンテナの GStreamer プラグインとして使用方法については、「」を参照してください[Docker コンテナで GStreamer 要素を実行する](#)。

C++ プロデューサー SDK でログ記録を使用する

C++ プロデューサー SDK アプリケーションのログ記録は、kinesis-video-native-buildフォルダの kvs_log_configuration ファイルで設定します。

次の例は、デフォルト設定ファイルの最初の行を示しています。この行で、DEBUG レベルのログエントリを AWS マネジメントコンソールに書き込むようにアプリケーションを設定します。

```
log4cplus.rootLogger=DEBUG, KvsConsoleAppender
```

詳細度が低いログ記録の場合は、ログ記録レベルを INFO に設定できます。

ログファイルにログエントリを書き込むようにアプリケーションを設定するには、ファイルの最初の行を次のように更新します。

```
log4cplus.rootLogger=DEBUG, KvsConsoleAppender, KvsFileAppender
```

これにより、kvs.log フォルダの kinesis-video-native-build/log にログエントリを書き込むようにアプリケーションが設定されます。

ログファイルの場所を変更するには、次の行を新しいパスで更新します。

```
log4cplus.appender.KvsFileAppender.File=../log/kvs.log
```

Note

DEBUG レベルのログ記録をファイルに書き込むと、ログファイルはデバイスの使用可能なストレージスペースをすぐに消費してしまう場合があります。

C プロデューサーライブラリを使用する

Amazon Kinesis Video Streams が提供する C プロデューサーライブラリを使用して、デバイスから Kinesis ビデオストリームにメディアデータを送信するアプリケーションコードを記述できます。

オブジェクトモデル

Kinesis Video Streams C プロデューサーライブラリは、プラットフォームに依存しないコードベース (PIC、Platform Independent Codebase) と呼ばれる共通コンポーネントに基づきます。このコー

ドベースは、GitHub (<https://github.com/aws-labs/amazon-kinesis-video-streams-pic/>) で入手できます。PIC には、基盤コンポーネントのプラットフォームに依存しないビジネスロジックが含まれています。Kinesis Video Streams C プロデューサーライブラリは、シナリオおよびプラットフォーム固有のコールバックとイベントを可能にする追加の API レイヤーで PIC をラップします。Kinesis Video Streams C プロデューサーライブラリには、PIC 上に構築された以下のコンポーネントがあります。

- デバイス情報プロバイダー - PIC API に直接提供できる DeviceInfo 構造を公開します。アプリケーションが処理するストリームの数とタイプに基づいてコンテンツストアを最適化できるアプリケーションシナリオ最適化プロバイダーや、使用可能な RAM の量に基づいて設定された必要なバッファリングの量など、一連のプロバイダーを設定できます。
- ストリーム情報プロバイダー - PIC API に直接提供できる StreamInfo 構造を公開します。アプリケーションタイプと一般的なタイプのストリーミングシナリオに固有のプロバイダーのセットがあります。これには、ビデオ、オーディオ、オーディオ、ビデオマルチトラックなどのプロバイダーが含まれます。これらの各シナリオには、アプリケーションの要件に応じてカスタマイズできるデフォルトがあります。
- コールバックプロバイダー - PIC API に直接提供できる ClientCallbacks 構造を公開します。これには、ネットワーキング (CURL ベースの API コールバック)、認可 (AWS 認証情報 API)、およびエラーコールバックの再試行ストリーミング用の一連のコールバックプロバイダーが含まれます。コールバックプロバイダー API は、AWS リージョンや認可情報など、設定する多数の引数を取ります。これは、IoT 証明書を使用するか、AccessKeyId、SecretKey、または SessionToken を使用して AWS 行われます。アプリケーションでアプリケーション固有のロジックを実現するために特定のコールバックをさらに処理する必要がある場合、カスタムコールバックでコールバックプロバイダーを拡張することができます。
- FrameOrderCoordinator - マルチトラックのシナリオ用に音声と動画の同期の処理に役立ちます。デフォルトの動作があり、アプリケーション固有のロジックを処理するようにカスタマイズできます。また、低層 PIC API に送信する前に、PIC フレーム構造でのフレームメタデータのパッケージ化を合理化します。マルチトラック以外のシナリオの場合、このコンポーネントは PIC putFrame API へのパススルーです。

C ライブラリには、Kinesis ストリームへのデータ送信を管理するために次のオブジェクトが用意されています。

- KinesisVideoClient - デバイスに関する情報が含まれ、Kinesis Video Streams イベントを報告するコールバックが維持されます。

- KinesisVideoStream – 名前、データ保持期間、メディアコンテンツタイプなど、ビデオストリームのパラメータに関する情報を表します。

ストリームにメディアを配置する

C ライブラリが提供するメソッド (などPutKinesisVideoFrame) を使用して、KinesisVideoStreamオブジェクトにデータを配置できます。ライブラリは、データの内部状態も管理します。タスクには以下が含まれる場合があります。

- 認証を実行する。
- ネットワークレイテンシーを監視する。レイテンシーが長すぎると、フレームが停止される場合があります。
- 進行中のストリーミングのステータスを追跡する。

手順: C プロデューサー SDK を使用する

この手順では、C アプリケーションで Kinesis Video Streams クライアントおよびメディアソースを使用して H.264 でエンコードされた動画フレームを Kinesis のビデオストリームに送信する方法について説明します。

この手順には、以下のステップが含まれます。

- [C プロデューサーライブラリコードをダウンロードする](#)
- [コードを記述して調べる](#)
- [コードを実行して検証する](#)

前提条件

C プロデューサー SDK を設定する前に、次の前提条件を満たしていることを確認してください。

- 認証情報 – サンプルコードでは、認証情報プロファイルファイルで設定したプロファイルを指定して AWS、認証情報を指定します。まず、認証情報プロファイルを設定します (まだ設定していない場合)。

詳細については、「[開発用の AWS 認証情報とリージョンの設定](#)」を参照してください。

- 証明書ストアの統合 – Kinesis Video Streams プロデューサーライブラリは、呼び出すサービスとの信頼を確立する必要があります。これは、パブリック証明書ストアで認証機関 (CAs) を検証す

ることによって行われます。Linux ベースのモデルの場合、このストアは `/etc/ssl/` ディレクトリにあります。

以下の場所から証明書ストアに、証明書をダウンロードしてください。

<https://www.amazontrust.com/repository/SFSRootCAG2.pem>

- macOS 用の次のビルド依存関係をインストールします。
 - [Autoconf 2.69](#) (ライセンス GPLv3+/Autoconf: GNU GPL バージョン 3 以降)
 - [CMake 3.または 3.8](#)
 - [Pkg-Config](#)
 - xCode (macOS) / clang / gcc (xcode-select バージョン 2347)
 - Java Development Kit (JDK) (Java JNI コンパイル用)
 - [Lib-Pkg](#)
- Ubuntu に次のビルド依存関係をインストールします。
 - Git: `sudo apt install git`
 - [CMake](#): `sudo apt install cmake`
 - G++: `sudo apt install g++`
 - pkg-config: `sudo apt install pkg-config`
 - OpenJDK: `sudo apt install openjdk-8-jdk`
 - JAVA_HOME 環境変数を設定します: `export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/`

C プロデューサーライブラリコードをダウンロードする

このセクションでは、低レベルのライブラリをダウンロードします。前提条件やこの例のその他の詳細については、「[the section called “C++”](#)」を参照してください。

1. ディレクトリを作成して、GitHub リポジトリのソースコード例からクローンを作成します。

```
git clone --recursive https://github.com/aws-labs/amazon-kinesis-video-streams-producer-c.git
```


Note

--recursive を使用して Git クローンを実行しなかった場合は、amazon-kinesis-video-streams-producer-c/open-source ディレクトリで `git submodule update --init` を実行してください。pkg-config、CMake、ビルド環境もインストールする必要があります。

詳細については、<https://github.com/aws-labs/amazon-kinesis-video-streams-producer-c.git> の README.md を参照してください。

2. 任意の統合開発環境 (IDE) ([Eclipse](#) など) でコードを開きます。

コードを記述して調べる

このセクションでは、GitHub の <https://github.com/aws-labs/amazon-kinesis-video-streams-producer-c> レポジトリの `samples` フォルダにあるサンプルアプリケーション `KvsVideoOnlyStreamingSample.c` のコードを調べます。このコードは前のステップでダウンロードしたものです。このサンプルでは、C プロデューサーライブラリを使用して、フォルダ `samples/h264SampleFrames` 内の H.264 エンコードされた動画フレームを Kinesis のビデオストリームに送信する方法を示します。

このサンプルアプリケーションは次の 3 つのセクションで構成されています。

- 初期化と設定:
 - プラットフォーム固有のメディアパイプラインの初期化および設定。
 - パイプラインの `KinesisVideoClient` および `KinesisVideoStream` の初期化および設定、コールバックの設定、シナリオ固有の認証の統合、コーデックプライベートデータの抽出および送信、ストリームの準備完了状態。
- メインループ:
 - タイムスタンプおよびフラグによるメディアパイプラインからのフレームの取得。
 - `KinesisVideoStream` へのフレームの送信。
- Teardown:
 - `KinesisVideoStream` の停止 (同期)、`KinesisVideoStream` の解放、`KinesisVideoClient` の解放。

このサンプルアプリケーションは以下のタスクを実行します。

- `createDefaultDeviceInfo` API を呼び出して、デバイスまたはストレージ設定に関する情報が含まれる `deviceInfo` オブジェクトを作成します。

```
// default storage size is 128MB. Use setDeviceInfoStorageSize after create to change storage size.  
CHK_STATUS(createDefaultDeviceInfo(&pDeviceInfo));  
// adjust members of pDeviceInfo here if needed  
pDeviceInfo->clientInfo.loggerLogLevel = LOG_LEVEL_DEBUG;
```

- `createRealtimeVideoStreamInfoProvider` API を呼び出して、`StreamInfo` オブジェクトを作成します。

```
CHK_STATUS(createRealtimeVideoStreamInfoProvider(streamName,  
    DEFAULT_RETENTION_PERIOD, DEFAULT_BUFFER_DURATION, &pStreamInfo));  
// adjust members of pStreamInfo here if needed
```

- `createDefaultCallbacksProviderWithAwsCredentials` API を呼び出して、静的 AWS 認証情報に基づいてデフォルトのコールバックプロバイダーを作成します。

```
CHK_STATUS(createDefaultCallbacksProviderWithAwsCredentials(accessKey,  
    secretKey,  
    sessionToken,  
    MAX_UINT64,  
    region,  
    cacertPath,  
    NULL,  
    NULL,  
    FALSE,  
    &pClientCallbacks));
```

- `createKinesisVideoClient` API を呼び出して、デバイスのストレージに関する情報を含み、Kinesis Video Streams イベントに関する報告を行うためのコールバックを管理する `KinesisVideoClient` オブジェクトを作成します。

```
CHK_STATUS(createKinesisVideoClient(pDeviceInfo, pClientCallbacks, &clientHandle));
```

- `createKinesisVideoStreamSync` API を呼び出して、`KinesisVideoStream` オブジェクトを作成します。

```
CHK_STATUS(createKinesisVideoStreamSync(clientHandle, pStreamInfo, &streamHandle));
```

- サンプルフレームを設定し、PutKinesisVideoFrame API を呼び出してそのフレームを KinesisVideoStream オブジェクトに送信します。

```
// setup sample frame
memset(frameBuffer, 0x00, frameSize);
frame.frameData = frameBuffer;
frame.version = FRAME_CURRENT_VERSION;
frame.trackId = DEFAULT_VIDEO_TRACK_ID;
frame.duration = HUNDREDS_OF_NANOS_IN_A_SECOND / DEFAULT_FPS_VALUE;
frame.decodingTs = defaultGetTime(); // current time
frame.presentationTs = frame.decodingTs;

while(defaultGetTime() > streamStopTime) {
    frame.index = frameIndex;
    frame.flags = fileIndex % DEFAULT_KEY_FRAME_INTERVAL == 0 ?
FRAME_FLAG_KEY_FRAME : FRAME_FLAG_NONE;
    frame.size = sizeof(frameBuffer);

    CHK_STATUS(readFrameData(&frame, frameFilePath));

    CHK_STATUS(putKinesisVideoFrame(streamHandle, &frame));
    defaultThreadSleep(frame.duration);

    frame.decodingTs += frame.duration;
    frame.presentationTs = frame.decodingTs;
    frameIndex++;
    fileIndex++;
    fileIndex = fileIndex % NUMBER_OF_FRAME_FILES;
}
```

- Teardown:

```
CHK_STATUS(stopKinesisVideoStreamSync(streamHandle));
CHK_STATUS(freeKinesisVideoStream(&streamHandle));
CHK_STATUS(freeKinesisVideoClient(&clientHandle));
```

コードを実行して検証する

のコードを実行して検証するには[the section called “C++”](#)、以下を実行します。

1. 次のコマンドを実行して、[ダウンロードした C SDK](#) に build ディレクトリを作成し、cmake から起動します。

```
mkdir -p amazon-kinesis-video-streams-producer-c/build;  
cd amazon-kinesis-video-streams-producer-c/build;  
cmake ..
```

次のオプションを cmake .. に渡すことができます。

- -DBUILD_DEPENDENCIES - ソースから依存ライブラリを構築するかどうか。
- -DBUILD_TEST=TRUE - ビルドユニットと統合テスト。デバイスのサポートを確認するのに役立つ場合があります。

./tst/webrtc_client_test

- -DCODE_COVERAGE - カバレッジレポートを有効にします。
 - -DCOMPILER_WARNINGS - すべてのコンパイラ警告を有効にします。
 - -DADDRESS_SANITIZER - AddressSanitizer を使用して構築します。
 - -DMEMORY_SANITIZER - MemorySanitizer を使用して構築します。
 - -DTHREAD_SANITIZER - ThreadSanitizer を使用して構築します。
 - -DUNDEFINED_BEHAVIOR_SANITIZER - UndefinedBehaviorSanitizer を使用して構築します。
 - -DALIGNED_MEMORY_MODEL - アライメントされたメモリモデルのみのデバイス用に構築します。デフォルトは OFF です。
2. 前のステップで作成した build ディレクトリに移動し、make を実行して WebRTC C SDK とその提供されたサンプルを構築します。

```
make
```

3. サンプルアプリケーション kinesis_video_cproducer_video_only_sample は、フォルダ samples/h264SampleFrames 内の H.264 でエンコードされた動画フレームを Kinesis Video Streams に送信します。次のコマンドは、10 秒ループの動画フレームを Kinesis Video Streams に送信します。

```
./kinesis_video_cproducer_video_only_sample YourStreamName 10
```

別のフォルダ (などMyH264FramesFolder) から H.264 でエンコードされたフレームを送信する場合は、次の引数を使用してサンプルを実行します。

```
./kinesis_video_cproducer_video_only_sample YourStreamName 10 MyH264FramesFolder
```

4. 詳細なログを有効にするには、CMakeList.txt の適切な行をコメント解除し、HEAP_DEBUG および LOG_STREAMING の C-定義を定義します。

テストスイートの進行状況は、IDE のデバッグ出力で監視できます。Amazon CloudWatch コンソールのストリームに関連付けられている PutMedia.IncomingBytes などのメトリクスを監視することで、ストリームのトラフィックをモニタリングすることもできます。

Note

テストハーネスでは空のバイトのフレームのみを送信するため、コンソールにはビデオストリームとしてのデータは表示されません。

Raspberry Pi で C++ プロデューサー SDK を使用する

Raspberry Pi は、基本的なコンピュータプログラミングスキルを説明して学習するために使用される小さく安価なコンピュータです。このチュートリアルでは、Raspberry Pi デバイスで Amazon Kinesis Video Streams C++ プロデューサー SDK をセットアップして使用方法について説明します。この手順には、GStreamer デモアプリケーションを使用してインストールを検証する方法も含まれています。

トピック

- [前提条件](#)
- [Kinesis Video Streams に書き込むアクセス許可を持つ IAM ユーザーを作成する](#)
- [Raspberry Pi を Wi-Fi ネットワークに結合する](#)
- [Raspberry Pi にリモートで接続する](#)
- [Raspberry Pi カメラを設定する](#)
- [ソフトウェアのインストールの前提条件](#)
- [Kinesis Video Streams C++ プロデューサー SDK をダウンロードして構築する](#)
- [Kinesis ビデオストリームにビデオをストリーミングする](#)

- [Kinesis ビデオストリームからメディアを再生する](#)
- [C++ プロデューサー SDK for Raspberry Pi のビルドに関する問題のトラブルシューティング](#)

前提条件

Raspberry Pi で C++ プロデューサー SDK を設定する前に、次の前提条件を満たしていることを確認してください。

- 以下の設定の Raspberry Pi デバイス
 - Board バージョン: 3 Model B 以降。
 - 接続された[カメラモジュール](#)または接続された USB カメラ (ウェブカメラ)。
 - 少なくとも 8 GB の容量がある SD カード。
 - Raspbian オペレーティングシステム (カーネルバージョン 4.9 以降) がインストールされている。最新の Raspberry Pi OS (以前の Raspbian) イメージは、[Raspberry Pi ウェブサイト](#)からダウンロードできます。Raspberry Pi ガイドの「[SD カードにダウンロードしたイメージをインストールする](#)」に従います。
- Kinesis ビデオストリーム AWS アカウント を持つ。詳細については、「[Kinesis ビデオストリームの使用開始](#)」を参照してください。

Kinesis Video Streams に書き込むアクセス許可を持つ IAM ユーザーを作成する

まだ設定していない場合は、Kinesis ビデオストリームに書き込むアクセス許可を持つ AWS Identity and Access Management (IAM) ユーザーを設定します。

これらの手順は、AWS アクセスキーペアの使用をすばやく開始するのに役立ちます。デバイスは X.509 証明書を使用して接続できます AWS IoT。証明書ベースの認証を使用するようにデバイスを設定する方法の詳細については、[the section called “を使用した Kinesis Video Streams リソースへのアクセスの制御 AWS IoT”](#)「」を参照してください。

1. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/iam/> で IAM コンソールを開きます。
2. 左側のナビゲーションメニューで [ユーザー] を選択します。
3. 新規ユーザーを作成するには、[ユーザーを追加] を選択します。

4. ユーザーにわかりやすい [ユーザー名] を提供します (**kinesis-video-raspberry-pi-producer** など)。
5. [アクセスの種類] で、[プログラムによるアクセス] を選択します。
6. [Next: Permissions] (次のステップ: 許可) を選択します。
7. [Set permissions for kinesis-video-raspberry-pi-producer (kinesis-video-raspberry-pi-producer にアクセス許可を設定する)] で [既存のポリシーを直接アタッチ] を選択します。
8. [Create policy] (ポリシーの作成) を選択します。[ポリシーの作成] ページが新しいウェブブラウザタブで開きます。
9. JSON タブを選択します。
10. 次の JSON ポリシーをコピーし、テキスト欄に貼り付けます。このポリシーは、Kinesis ビデオストリームにデータを作成および書き込むアクセス許可をユーザーに付与します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "kinesisvideo:DescribeStream",
      "kinesisvideo:CreateStream",
      "kinesisvideo:GetDataEndpoint",
      "kinesisvideo:PutMedia"
    ],
    "Resource": [
      "*"
    ]
  }]
}
```

11. [ポリシーの確認] を選択します。
12. など、ポリシーの名前を指定します **kinesis-video-stream-write-policy**。
13. [Create policy] (ポリシーの作成) を選択します。
14. ブラウザで [ユーザーを追加] タブに戻り、[Refresh (更新)] を選択します。
15. 検索ボックスに作成したポリシーの名前を入力します。
16. リストの作成した新しいポリシーの横のチェックボックスを選択します。
17. [Next: Review] を選択します。

18. [Create user] を選択します。
19. コンソールには、新規ユーザーの [アクセスキー ID] が表示されます。[表示] を選択して、[シークレットアクセスキー] を表示します。この値を記録します。アプリケーションを設定するときに、この値が必要となります。

Raspberry Pi を Wi-Fi ネットワークに結合する

アタッチされたモニターおよびキーボードを使用する場合には、「[Raspberry Pi カメラを設定する](#)」に進みます。

これらの手順は、ヘッドレスモードで実行するとき、つまりキーボード、モニター、またはネットワークケーブルがアタッチされていないときに Raspberry Pi を設定するのに役立つように記述されています。以下の手順に従って、指定したネットワークへの接続を自動的に試行するように Raspberry Pi を設定し、ホストマシンがそのネットワークに SSH 接続できるようにします。

1. コンピュータに wpa_supplicant.conf という名前のファイルを作成します。
2. 次のテキストをコピーし、wpa_supplicant.conf ファイルに貼り付けます。

```
country=US
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
  ssid="Your Wi-Fi SSID"
  scan_ssid=1
  key_mgmt=WPA-PSK
  psk="Your Wi-Fi Password"
}
```

ssid と psk 値を使用する Wi-Fi ネットワークの情報に置き換えます。

3. wpa_supplicant.conf ファイルを SD カードにコピーします。boot ボリュームのルートにコピーする必要があります。
4. Raspberry Pi に SD カードを挿入し、デバイスの電源を入れます。Wi-Fi ネットワークに接続し、SSH が有効になります。

Raspberry Pi にリモートで接続する

アタッチされたモニターおよびキーボードを使用する場合には、「[Raspberry Pi カメラを設定する](#)」に進みます。

これらの手順は、ヘッドレスモードで実行するとき、つまりキーボード、モニター、またはネットワークケーブルがアタッチされていないときに Raspberry Pi を設定するのに役立つように記述されています。以下の手順に従って、ネットワーク上の Raspberry Pi を見つけ、ホストマシンからそこに SSH 接続します。

1. Raspberry Pi デバイスにリモートで接続する前に、IP アドレスを確認するために次のいずれかを実行します。
 - ネットワークの Wi-Fi ルーターにアクセスできる場合には、接続した Wi-Fi デバイスを探します。Raspberry Pi という名前のデバイスを検索して、デバイスの IP アドレスを探します。
 - ネットワークの Wi-Fi ルーターにアクセスできない場合には、ネットワーク上のデバイスを検索する他のソフトウェアを使用できます。[Fing](#) は、Android と iOS デバイスのどちらでも利用できる広く使用されているアプリケーションです。このアプリケーションの無償バージョンを使用して、ネットワーク上のデバイスの IP アドレスを見つけることができます。
2. Raspberry Pi デバイスの IP アドレスがわかっている場合には、任意のターミナルアプリケーションを使用して接続できます。
 - macOS や Linux では、ssh を使用します。

```
ssh pi@<IP address>
```

- Windows では、Windows 向けの無料の SSH クライアントである [PuTTY](#) を使用します。

新規にインストールした Raspbian では、ユーザー名は **pi**、パスワードは **raspberry** です。[このデフォルトパスワードを変更する](#)ことが推奨されます。

Raspberry Pi カメラを設定する

デバイスから Kinesis ビデオストリームにビデオを送信するように [Raspberry Pi カメラモジュール](#)を設定するには、次の手順に従います。

Note

USB ウェブカメラを使用している場合は、「」に進みます [the section called “ソフトウェアのインストールの前提条件”](#)。

Camera module 1

以下の手順に従って、モジュールファイルを更新し、カメラインターフェイスを有効にして、カメラの機能を確認します。モジュールファイルを更新すると、起動時にロードするカーネルモジュールが Raspberry Pi に通知されます。カメラを使用していない Raspberry Pi デバイスのシステムリソースを節約するために、カメラドライバはデフォルトではロードされません。

1. エディタを開いてモジュールファイルを変更します。ターミナルを開き、次のコマンドを使用して nano エディタを使用してファイルを編集します。

```
sudo nano /etc/modules
```

2. ファイルの末尾に次の行を追加します (既存しない場合)。

```
bcm2835-v4l2
```

3. ファイルを保存し、エディタを終了します。nano エディタを使用して保存して終了するには、Ctrl+X を使用します。
4. Raspberry Pi の再起動。

```
sudo reboot
```

5. デバイスが再起動したら、リモート接続の場合には、ターミナルアプリケーションから再度接続します。
6. オープン raspi-config:

```
sudo raspi-config
```

7. インターフェイスオプション、レガシーカメラを選択します。Raspbian オペレーティングシステムの古いビルドでは、このメニューオプションはインターフェイスオプション、カメラにある可能性があります。

カメラを有効にしていない場合は有効にし、プロンプトされる場合には再起動します。

8. 次のコマンドを入力して、カメラが正常に機能することを確認します。

```
raspistill -v -o test.jpg
```

カメラが正しく設定されている場合、このコマンドはカメラからイメージをキャプチャし、という名前のファイルに保存してtest.jpg、情報メッセージを表示します。

Camera module 2 or 3

カメラモジュール 2 を使用している場合は、bcm2835-v4l2 (レガシー) または libcamera (モダン) のいずれかを使用します。ただし、サポートと機能を向上させるには、libcamera スタックをお勧めします。以下の手順に従って、libcamera がシステムで up-to-date であることを確認します。

1. [libcamera](#) は Raspberry Pi にプリインストールされている必要があります。バグ修正とセキュリティ更新について、更新がないか確認し、最新バージョンに更新してください。ターミナルを開き、次のコマンドを入力します。

```
sudo apt-get update  
sudo apt-get upgrade
```

2. 更新を有効にするには、システムを再起動します。

```
sudo reboot
```

3. カメラをテストします。このアプリケーションはカメラプレビューストリームを開始し、画面に表示します。

```
libcamera-hello
```

カメラモジュールに問題がある場合は、トラブルシューティングのために [Raspberry Pi のドキュメント](#) を参照してください。

ソフトウェアのインストールの前提条件

C++ プロデューサー SDK では、Raspberry Pi に以下のソフトウェア前提条件をインストールする必要があります。

1. パッケージリストを更新し、SDK の構築に必要なライブラリをインストールします。ターミナルを開き、次のコマンドを入力します。

```
sudo apt-get update
sudo apt-get install -y \
    automake \
    build-essential \
    cmake \
    git \
    gstreamer1.0-plugins-base-apps \
    gstreamer1.0-plugins-bad \
    gstreamer1.0-plugins-good \
    gstreamer1.0-plugins-ugly \
    gstreamer1.0-tools \
    gstreamer1.0-omx-generic \
    libcurl4-openssl-dev \
    libgstreamer1.0-dev \
    libgstreamer-plugins-base1.0-dev \
    liblog4cplus-dev \
    libssl-dev \
    pkg-config
```

2. libcamera スタックを使用している場合は、libcamerasrcGStreamer プラグインもインストールします。この GStreamer プラグインはデフォルトではインストールされません。

```
sudo apt-get install gstreamer1.0-libcamera
```

3. 次の PEM ファイルを /etc/ssl/cert.pem にコピーします。

```
sudo curl https://www.amazontrust.com/repository/AmazonRootCA1.pem -o /etc/ssl/
AmazonRootCA1.pem
sudo chmod 644 /etc/ssl/AmazonRootCA1.pem
```

Kinesis Video Streams C++ プロデューサー SDK をダウンロードして構築する

[Kinesis Video Streams C++ プロデューサー SDK](#) をダウンロードして構築するには、次の手順に従います。ソフトウェアの前提条件がインストールされていることを確認します。これらの手順 [the section called “ソフトウェアのインストールの前提条件”](#) については、「」を参照してください。

1. ダウンロードディレクトリに移動します。ターミナルを開き、任意のダウンロードディレクトリに変更します。

例えば、次のようになります。

```
cd ~/Downloads
```

2. SDK リポジトリのクローンを作成します。git clone コマンドを使用して、GitHub リポジトリから SDK をダウンロードします。タイプ:

```
git clone https://github.com/aws-labs/amazon-kinesis-video-streams-producer-sdk-cpp.git --single-branch -b master kvs-producer-sdk-cpp
```

このコマンドは 1 つのブランチ (master ブランチ) のみをクローンし、ダウンロードサイズと時間を短縮します。また、ダウンロードしたコンテンツを現在のディレクトリ kvs-producer-sdk-cpp 内の というフォルダに配置します。

3. ダウンロードを確認します。クローン作成プロセスが完了したら、kvs-producer-sdk-cpp フォルダの内容を一覧表示して、SDK がダウンロードされたことを確認します。

```
ls kvs-producer-sdk-cpp
```

4. ビルドディレクトリを準備します。タイプ:

```
mkdir -p kvs-producer-sdk-cpp/build  
cd kvs-producer-sdk-cpp/build
```

5. ビルドを設定します。次の cmake コマンドを実行して、ビルド環境を特定のオプションで設定します。

```
cmake .. -DBUILD_GSTREAMER_PLUGIN=ON -DBUILD_DEPENDENCIES=OFF -  
DALIGNED_MEMORY_MODEL=ON
```

[CMake](#) は、次のオプションを使用して適切な を生成します Makefiles。

- プロジェクトフォルダ (..) をソースディレクトリとして使用します。
- ビルド出力に現在のディレクトリ (.) (build/) を使用します。
- -DBUILD_GSTREAMER_PLUGIN=ON では、GStreamer プラグイン kvssink を構築できます。

- `-DBUILD_DEPENDENCIES=OFF` は、ソースからの外部依存関係の構築を無効にします。プロジェクトは、前のステップでインストールされた外部依存関係を見つけて使用します。
- `-DALIGNED_MEMORY_MODEL=ON` は、アライメントされていないメモリモデルを無効にします。アライメントされていないメモリアクセスは、特定の Raspberry Pi デバイスではサポートされていません。

 Note

CMake 引数の完全なリストについては、「」を参照してください [the section called “コードをダウンロードして設定する”](#)。

6. プロジェクトをビルドします。ビルドを設定したら、`make` コマンドを使用して、`cmake` によって Makefile 生成された `cmake` を使用してコンパイルします。

```
make -j$(nproc)
```

引数 `-j` 数 `make` により、複数のコンパイルジョブを並行して実行できます。ビルド時間を短縮するには、`nproc` コマンドを使用して Raspberry Pi の CPU コア数を動的に計算します。

7. `libgstkvssink.so` が存在することを確認します。

現在のディレクトリ内のファイルを一覧表示します。

プロンプト:

```
ls
```

レスポンス:

CMakeCache.txt	dependency	kvs_gstreamer_sample
CMakeFiles	kvs_gstreamer_audio_video_sample	kvssink_gstreamer_sample
Makefile	kvs_gstreamer_file_uploader_sample	libKinesisVideoProducer.so
cmake_install.cmake	kvs_gstreamer_multistream_sample	libgstkvssink.so

8. GStreamer がロードできることを確認します `kvssink`。

`GST_PLUGIN_PATH` 環境変数を 含むディレクトリに設定します `libgstkvssink.so`。

```
export GST_PLUGIN_PATH=`pwd`
```

GStreamer に をロードさせますkvssink。

```
gst-inspect-1.0 kvssink
```

に関するドキュメントがいくつか表示されますkvssink。矢印キーを使用して移動し、qを押して終了します。

9. (オプション) GST_PLUGIN_PATH環境変数の設定を含めるようにシェルの起動スクリプトを更新します。これによりGST_PLUGIN_PATH、新しいターミナルセッション中に が適切に設定されます。Raspberry Pi デバイスの場合、シェルの起動スクリプトは です ~/.bashrc。

次のコマンドを実行して、シェルの起動スクリプトの末尾に コマンドを追加します。

```
echo "export GST_PLUGIN_PATH=~/Downloads/kvs-producer-sdk-cpp/build" >> ~/.bashrc
```

次のように入力してシェルの起動スクリプトを実行するか、現在のシェルを閉じて新しいシェルを開きます。

```
source ~/.bashrc
```

GST_PLUGIN_PATH が設定され、 をロードできることを確認しますkvssink。

```
echo $GST_PLUGIN_PATH
```

```
gst-inspect-1.0 kvssink
```

Kinesis ビデオストリームにビデオをストリーミングする

サンプルアプリケーションを実行するには、次の情報が必要です。

- 「[前提条件](#)」セクションで作成したストリームの名前。
- 「[Kinesis Video Streams に書き込むアクセス許可を持つ IAM ユーザーを作成する](#)」で作成したアカウントの認証情報 (アクセスキー ID およびシークレットアクセスキー)。

- GStreamer はkvssinkプラグインを見つけることができます。詳細については「[the section called “C++ プロデューサー SDK をダウンロードして構築する”](#)」を参照してください。

1. 認証情報とリージョンを設定します。

```
export AWS_ACCESS_KEY_ID=YourAccessKey
export AWS_SECRET_ACCESS_KEY=YourSecretKey
export AWS_DEFAULT_REGION=us-west-2
```

その他の認証方法については、「」を参照してください[the section called “認証情報を に提供する kvssink”](#)。

Note

C++ プロデューサー SDK は、デフォルトで米国西部 (オレゴン) (us-west-2) リージョンを使用します。デフォルトを使用するには、米国西部 (オレゴン) リージョンで Kinesis ビデオストリーム AWS リージョン を作成します。
Kinesis ビデオストリームに別のリージョンを使用するには、次の環境変数をリージョン (*us-east-1* ##) に設定します。

```
export AWS_DEFAULT_REGION=us-east-1
```

2. 入力メディアに応じて、次のいずれかを選択します。

Sample GStreamer video

この GStreamer パイプラインは、640 x 480 ピクセルの解像度で 1 秒あたり 10 フレームで実行される標準テストパターンを持つライブテストビデオストリームを生成します。オーバーレイが追加され、現在のシステム日時が表示されます。その後、ビデオは H.264 形式にエンコードされ、キーフレームは最大 10 フレームごとに生成され、フラグメント期間 (写真のグループ (GoP) サイズとも呼ばれます) は 1 秒になります。kvssink は H.264 エンコードされたビデオストリームを取得し、Matroska (MKV) コンテナ形式にパッケージ化して、Kinesis ビデオストリームにアップロードします。

次のコマンドを実行します。

```
gst-launch-1.0 -v videotestsrc is-live=true \
! video/x-raw,framerate=10/1,width=640,height=480 \
```



```
! clockoverlay time-format="%a %B %d, %Y %I:%M:%S %p" \  
! x264enc bframes=0 key-int-max=10 \  
! h264parse \  
! kvssink stream-name="YourStreamName"
```

GStreamer パイプラインを停止するには、ターミナルウィンドウを選択し、Ctrl+C を押します。

サンプルビデオ GStreamer パイプラインは次のようになります。

USB web cam

次のコマンドを実行して、GStreamer で USB カメラを自動検出します。

```
gst-launch-1.0 autovideosrc \  
! videoconvert \  
! video/x-raw,format=I420,width=640,height=480 \  
! x264enc bframes=0 key-int-max=45 tune=zerolatency byte-stream=true speed-  
preset=ultrafast \  
! h264parse \  
! video/x-h264,stream-format=avc,alignment=au,profile=baseline \  
! kvssink stream-name="YourStreamName"
```

GStreamer パイプラインを停止するには、ターミナルウィンドウを選択し、Ctrl+C を押します。

GStreamer が自動検出するのではなく、特定のデバイス識別子v4l2srcで を使用できます。次のコマンドを実行します。

```
gst-device-monitor-1.0
```

出力には、いくつかのデバイスと、デバイスの使用方法に関する GStreamer パイプラインの開始が表示されます。

Device found:

```
name   : H264 USB Camera: USB Camera  
class  : Video/Source  
caps   : video/x-h264, stream-format=(string)byte-stream,  
alignment=(string)au, width=(int)1920, height=(int)1080, pixel-aspect-
```

```
ratio=(fraction)1/1, colorimetry=(string){ 2:4:7:1 }, framerate=(fraction)
{ 30/1, 25/1, 15/1 };

...
properties:
  device.path = /dev/video4
  udev-probed = false
  device.api = v4l2
  v4l2.device.driver = uvcvideo
  v4l2.device.card = "H264\ USB\ Camera:\ USB\ Camera"
  v4l2.device.bus_info = usb-3f980000.usb-1.3
  v4l2.device.version = 265767 (0x00040e27)
  v4l2.device.capabilities = 2216689665 (0x84200001)
  v4l2.device.device_caps = 69206017 (0x04200001)
gst-launch-1.0 v4l2src device=/dev/video4 ! ...
```

GStreamer パイプラインを停止するには、ターミナルウィンドウを選択し、Ctrl+C を押します。

Raspberry Pi camera module 1

で Pi カメラモジュール 1 または Pi カメラモジュール 2 を使用している場合はbcm2835-v4l2、以下を使用します。

```
gst-launch-1.0 v4l2src device=/dev/video0 \
! videoconvert \
! video/x-raw,format=I420,width=640,height=480 \
! x264enc bframes=0 key-int-max=45 bitrate=500 tune=zerolatency \
! h264parse ! video/x-h264,stream-format=avc,alignment=au,profile=baseline \
! kvssink stream-name="YourStreamname"
```

GStreamer パイプラインを停止するには、ターミナルウィンドウを選択し、Ctrl+C を押します。

Raspberry Pi camera module 2 or 3

最新のlibcameraスタックを使用している場合は、次の GStreamer パイプラインを使用します。

```
gst-launch-1.0 libcamerasrc \
! video/x-raw,width=640,height=480,framerate=30/1,format=I420 \
! videoconvert \
! x264enc speed-preset=ultrafast tune=zerolatency byte-stream=true key-int-max=75 \
```

```
! video/x-h264,level='(string)4' \  
! h264parse \  
! video/x-h264,stream-  
format=avc,alignment=au,width=640,height=480,framerate=30/1 \  
! kvssink stream-name="YourStreamname"
```

GStreamer パイプラインを停止するには、ターミナルウィンドウを選択し、Ctrl+C を押します。

Sample RTSP camera

この例では、Gst-Rtsp-Server を使用してデモ RTSP カメラフィードをローカルでホストします。次に、その RTSP カメラフィードを指定された Kinesis ビデオストリームにアップロードする GStreamer パイプラインを構築します。

Raspberry Pi で Gst-Rtsp-Server を設定するには

1. Gst-Rtsp-Server プロジェクトを構築するために必要な依存関係ライブラリをインストールします。ソフトウェアの前提条件もインストールされていることを確認してください。ターミナルに次のように入力します。

```
sudo apt-get update  
sudo apt-get install libgststrtspserver-1.0
```

2. Raspberry Pi に GStreamer の 1.22 バージョンをダウンロードします。

```
git clone https://gitlab.freedesktop.org/gstreamer/gstreamer.git --single-  
branch -b 1.22
```

3. ディレクトリを gst-rtsp-server の examples ディレクトリに変更します。

```
cd gstreamer  
cd subprojects  
cd gst-rtsp-server  
cd examples
```

4. gcc を使用して test-launch.c を test-launch という実行可能ファイルにコンパイルします。

```
gcc -o test-launch test-launch.c `pkg-config --cflags --libs gstreamer-rtsp-  
server-1.0`
```

5. 次の引数を使用して実行可能ファイルを実行します。注: GStreamer の初回ロードには時間がかかる場合があります。

```
./test-launch "videotestsrc is-live=true ! video/x-raw,height=480,width=640,framerate=10/1 ! videoconvert ! x264enc tune=zerolatency bitrate=512 key-int-max=25 bframes=0 ! h264parse ! rtph264pay ! name=pay0 pt=96"
```

以下の出力が表示されます。

```
stream ready at rtsp://127.0.0.1:8554/test
```

6. RTSP ビデオストリームを確認します。任意の RTSP ビューワーを使用できます。たとえば、VLC メディアプレーヤーです。VLC メディアプレーヤーを使用してライブストリームを表示するには、新しいターミナルを開き、次のように入力します。

```
sudo apt-get install vlc
```

VLC メディアプレーヤーをインストールします。次に、次のように入力します。

```
vlc rtsp://127.0.0.1:8554/test
```

ライブストリームとともに VLC ウィンドウが表示されます。そうでない場合は、テスト起動実行可能ファイルがまだ実行中であることを確認し、エラーがないか出力を確認します。

RTSP ストリームを検証するもう 1 つの方法は、gst-discoverer-1.0 ユーティリティを使用することです。タイプ:

```
gst-discoverer-1.0 "rtsp://127.0.0.1:8554/test"
```

予想される出力は次のようになります。

```
Analyzing rtsp://127.0.0.1:8554/test
Done discovering rtsp://127.0.0.1:8554/test

Properties:
  Duration: 99:99:99.999999999
  Seekable: no
```

```
Live: yes
unknown #0: application/x-rtp
  video #1: H.264 (Constrained Baseline Profile)
    Stream ID:
    359314d7d4bba383223927d7e57d4244d0800e629c626be81c505055c62170e2/
video:0:0:RTP:AVP:96
  Width: 640
  Height: 480
  Depth: 24
  Frame rate: 10/1
  Pixel aspect ratio: 1/1
  Interlaced: false
  Bitrate: 0
  Max bitrate: 0
```

kvssink を使用して RTSP フィードを Kinesis Video Stream に送信するには

この GStreamer パイプラインは `rtspsrc`、を使用して RTSP サーバーに接続し、RTP ビデオストリームを取得します。フレームを `rtph264depay` に渡します。これにより `rtph264depay`、RTP パケットから H.264 でエンコードされたビデオフレームが抽出されます。は、ビデオフレームを形式で `h264parse` グループ化 `kvssink` します。 `kvssink` は H.264 でエンコードされたビデオストリームを取得し、Matroska (MKV) コンテナ形式にパッケージ化して、Kinesis ビデオストリームにアップロードします。

次のコマンドを実行します。

```
gst-launch-1.0 -v rtspsrc location="rtsp://127.0.0.1:8554/test" short-
header=true \
! rtph264depay \
! h264parse \
! video/x-h264,format=avc,alignment=au \
! kvssink stream-name="YourStreamName"
```

GStreamer パイプラインを停止するには、ターミナルウィンドウを選択し、Ctrl+C を押します。

ハードウェアの使用

一部の Raspberry Pi モデルには、ハードウェアアクセラレーション H.264 エンコーダーが付属しています。ソフトウェアエンコーダ `x264enc` である の代わりに使用できます。

1. GStreamer プラグインがインストールされていることを確認します。

```
sudo apt-get install gstreamer1.0-tools gstreamer1.0-plugins-bad
```

2. タイプ:

```
gst-inspect-1.0 | grep h264
```

次の要素が使用可能かどうかを確認します。

- `omxh264enc`
- `v4l2h264enc`

利用可能な場合は、それらを使用できます。これらの要素を使用したパイプラインの例を以下に示します。

omxh264enc:

```
gst-launch-1.0 v4l2src device=/dev/video0 \  
! videoconvert \  
! video/x-raw,format=I420,width=640,height=480 \  
! omxh264enc control-rate=2 target-bitrate=512000 periodicity-idr=45 inline-  
header=FALSE \  
! h264parse ! video/x-h264,stream-format=avc,alignment=au,profile=baseline \  
! kvssink stream-name="raspberry"
```

v4l2h264enc および v4l2convert:

```
gst-launch-1.0 libcamerasrc \  
! video/x-raw,width=640,height=480,framerate=30/1,format=I420 \  
! v4l2convert \  
! v4l2h264enc extra-controls="controls,repeat_sequence_header=1" \  
! video/x-h264,level='(string)4' \  
! h264parse \  

```

```
! video/x-h264,stream-format=avc,alignment=au,width=640,height=480,framerate=30/1
\
! kvssink stream-name="test-stream"
```

ランタイムの問題

以下に、頻繁に発生するランタイムの問題とそのトラブルシューティング方法を示します。

そのような要素「xxxxxxxx」はありません

次のようなエラーが表示された場合は、GStreamer プラグインがないことを意味します。

```
WARNING: erroneous pipeline: no element "videoconvert"
```

解決策:

欠落している要素に基づいて、適切なアクションを決定します。

- kvssink: 「」を参照してください [the section called “C++ プロデューサー SDK をダウンロードして構築する”](#)。
- libcamerasrc: libcamerasrc GStreamer 要素をインストールする [the section called “「バッファプールのアクティベーションに失敗しました」エラー”](#) には、「」を参照してください。
- omxh264enc または v4l2h264enc:

に従って [the section called “ソフトウェアのインストールの前提条件”](#)、すべての GStreamer ライブラリをインストールします。これらをすべてインストールし、これらの要素が表示されない場合は、Raspberry Pi にハードウェアがないことを意味します。x264enc 代わりにソフトウェアエンコーダーを使用します。

- その他: [the section called “ソフトウェアのインストールの前提条件”](#) に従ってすべての GStreamer ライブラリをインストールします。さまざまな GStreamer 要素がさまざまな GStreamer プラグイングループ (良い、悪い、醜い) にあるため、すべてインストールしてください。

「バッファプールのアクティベーションに失敗しました」エラー

次のようなエラーが表示された場合は、使用されているパイプラインが を使用していることを意味しますが v4l2src、libcamerasrc 代わりに を使用する必要があります。

```
ERROR bufferpool gstbufferpool.c:572:gst_buffer_pool_set_active:source:pool0:src start failed
```

```
WARN v4l2src gstv4l2src.c:976:gst_v4l2src_decide_allocation: error: Failed to allocate
required memory.
WARN v4l2src gstv4l2src.c:976:gst_v4l2src_decide_allocation: error: Buffer pool
activation failed
WARN basesrc gstbasesrc.c:3352:gst_base_src_prepare_allocation: Subclass failed to
decide allocation
Error received from element source: Failed to allocate required memory.
WARN basesrc gstbasesrc.c:3132:gst_base_src_loop: error: Internal data stream error.
Debugging information: ../sys/v4l2/gstv4l2src.c(976): gst_v4l2src_decide_allocation
(): /GstPipeline:live-kinesis-pipeline/GstV4l2Src:source:
Buffer pool activation failed
WARN basesrc gstbasesrc.c:3132:gst_base_src_loop: error: streaming stopped, reason not-
negotiated (-4)
```

たとえば、次のパイプラインをカメラモジュール 2 に libcamerasrc インストールせずに使用している場合、GStreamer が使用する要素を自動検出しようとする、このエラーが発生することがあります。

```
gst-launch-1.0 autovideosrc ! videoconvert ! autovideosink
```

解決策:

libcamerasrc がインストールされていることを確認し、ではなくソース要素として使用します v4l2src。libcamerasrc GStreamer 要素をインストールするには、次のように入力します。

```
sudo apt-get update
sudo apt-get install gstreamer1.0-libcamera
```

libcamerasrc をインストールすると、autovideosrc 要素を使用している場合、GStreamer は libcamerasrc の代わりに正しいソースを使用するように自動的に切り替える必要があります v4l2src。

バスエラー

開始直後 kvssink (通常は の HTTP 呼び出し PutMedia が完了した前後) に Bus エラーが表示された場合は、Raspberry Pi がアライメントされていないメモリアクセスをサポートしていないことを意味します。ログは次のようになります。

```
INFO Camera camera.cpp:1197 configuring streams: (0) 640x480-YUV420
INFO RPI pisp.cpp:1450 Sensor: /base/axi/pcie@120000/rp1/i2c@88000/imx708@1a - Selected
sensor format: 1536x864-SBGGR10_1X10 - Selected CFE format: 1536x864-PC1B
```



```
[INFO ] kinesisVideoStreamFormatChanged(): Stream format changed.
[DEBUG] setRequestHeader(): Appending header to request: user-agent -> AWS-SDK-KVS-CPP-CLIENT/3.4.2/1.5.3 GCC/12.2.0 Linux/6.6.51+rpt-rpi-v8 aarch64 CPPSDK
[DEBUG] setRequestHeader(): Appending header to request: x-amzn-stream-name -> demo-stream
[DEBUG] setRequestHeader(): Appending header to request: x-amzn-producer-start-timestamp -> 1732012345.678
[DEBUG] setRequestHeader(): Appending header to request: x-amzn-fragment-acknowledgment-required -> 1
[DEBUG] setRequestHeader(): Appending header to request: x-amzn-fragment-timecode-type -> ABSOLUTE
[DEBUG] setRequestHeader(): Appending header to request: transfer-encoding -> chunked
[DEBUG] setRequestHeader(): Appending header to request: connection -> keep-alive
[INFO ] putStreamResultEvent(): Put stream result event. New upload handle 0
[WARN ] notifyDataAvailable(): [demo-stream] Failed to un-pause curl with error: 43.
    Curl object 0xe2f6f418
    Bus error
```

Kinesis Video Streams PIC は、アライメントされていないメモリアクセスを使用してメモリ使用量を最適化します。これは、すべてのデバイスでサポートされているわけではありません。

解決策:

アラインされたメモリアクセスモードで SDK を使用するには、をコンパイルONするときに `ALIGNED_MEMORY_MODEL` CMake フラグを明示的にに設定する必要があります。これは `kvssink`、デフォルトがであるためですOFF。詳細な手順[the section called “C++ プロデューサー SDK をダウンロードして構築する”](#)については、「」を参照してください。

タイムスタンプがフリーズし、パイプラインが停止する

GStreamer パイプライン `x264enc` でを使用する場合、パイプラインのタイムラインが数秒以内に大幅にまたは完全に停止することがあります。

これは、`x264enc` デフォルト設定によってエンコーディングレイテンシーが高くなり、デフォルトの入力バッファの容量を超える可能性があるためです。その結果、入力バッファがいっぱいになり、アップストリーム要素がブロックされ、パイプラインが停止します。

詳細については、[GStreamer ドキュメント](#)を参照してください。

解決策:

調整オプションx264encを使用して zerolatency を設定します。これにより、リアルタイムのシナリオに合わせて を最適化することでエンコーディングのレイテンシーが大幅に短縮され、フレームの処理と出力がより迅速になります。

設定例:

```
... ! x264enc tune=zerolatency byte-stream=true speed-preset=ultrafast bframes=0 key-int-max=60 ! ...
```

Note

このソリューションはパイプラインの停止を効果的に防止しますが、エンコーディングの効率と品質に影響を与える可能性があります。低レイテンシーと高品質の両方を必要とするシナリオでは、ハードウェア最適化の使用や H.264 を直接出力するウェブカメラの検索など、このエンコーディングステップをスキップする代替アプローチを検討してください。詳細については、「[the section called “ハードウェアの使用”](#)」を参照してください。

同じv412デバイスから複数のパイプラインを同時に実行することはできません

などのデバイスには、一度に 1 つのプロセス/dev/video0でのみアクセスできます。複数のプロセスが同時にアクセスしようとする、2 番目のプロセスは最初のプロセスが完了するまで待機します。

解決策:

ループバックデバイスを作成し、複数のプロセスが同時にループバックインターフェイスを使用できるようにします。詳細については、「[Stack Exchange](#)」を参照してください。

内部データストリームエラー

GStreamer パイプラインを作成するときは、ある要素のソースパッドを別の要素のシンクパッドにリンクして要素を接続します。このリンクプロセスにより、ソース要素からシンク要素へのデータフローが可能になり、データパイプラインが形成されます。

ログのエラーメッセージ「パッドリンク失敗」は、パイプライン内の 2 つの要素のパッド間の接続 (リンク) を確立しようとしたときに GStreamer で問題が発生したことを示します。

```
Pad link failed
```

```
Error received from element udpsrc0: Internal data stream error.
```

解決策:

どの要素が相互にリンクできないかを判断します。パイプラインの範囲を絞り込むには、パイプラインから要素を削除します。右端の要素を置き換え `fakesink`、要素を一度に 1 つずつ削除します。

場合によっては、[capsfilter](#) 要素を調整したり、パイプラインが使用する要素を変更したりする必要があります。

一般的なケースでは、カメラがサポートしていない resolution framerate または を要求しています。ターミナル `gst-device-monitor-1.0` を使用して、サポートされている `framerates`、`resolutions`、および `formats` を取得します。[ビデオスケール](#) GStreamer 要素を使用してビデオ解像度を調整し、[ビデオレート](#) を使用してビデオフレームレートを調整できます。

個々の GStreamer 要素でサポートされている形式を確認するには、ターミナル `gst-inspect-1.0 element-name` に と入力します。

Kinesis ビデオストリームからメディアを再生する

[Kinesis Video Streams コンソール](#)を開き、作成したストリームのストリーム名を選択します。

Raspberry Pi から送信された動画ストリームがコンソールに表示されます。

Note

ビデオがコンソールに表示されるまでに数秒かかる場合があります。

ストリームが再生されたら、コンソールで次の機能を試すことができます。

- [ビデオのプレビュー] セクションから、ナビゲーションコントロールを使用しストリームの巻き戻しまたは早送りを行います。
- ストリーム情報セクションで、ストリームのコーデック、解像度、ビットレートを確認します。このチュートリアルでは、帯域幅の使用を最小限に抑えるために、Raspberry Pi の解像度とビットレートの値は意図的に低く設定されています。

ストリーム用に作成されている Amazon CloudWatch メトリクスを表示するには、CloudWatch でストリームメトリクスを表示するを選択します。

ネットワーク帯域幅の不足やデバイスの制約により、コンソールでのメディアのロードが遅くなることがありますが、ビデオのエンコードとフラグメント化に関連している可能性もあります。

ビデオエンコードの基本:

- H.264 および H.265 エンコーダは、効率的な圧縮のためにキーフレーム (I フレーム) と予測フレーム (P フレーム) を使用します。
- キーフレームには完全なイメージデータが含まれ、P フレームには以前のフレームからの変更のみが含まれます。
- 「キーフレーム間隔」は、ビデオストリームでキーフレームが発生する頻度を決定します。

ストリーミングでのフラグメント化:

- Kinesis Video Streams では、新しいフラグメントは各キーフレームで始まります。詳細については、「[the section called “データモデル”](#)」を参照してください。
- フラグメントの長さ (秒単位) は、キーフレーム間隔 ÷ フレームレートとして推定できます。

例:

キーフレーム間隔が 30 でフレームレートが 15 fps のストリームの場合:

フラグメントの長さ = $30 \div 15 = 2$ 秒

キーフレーム間隔が大きいため、フラグメントが長くなるとストリーミングメディアのレイテンシーが増加します。

解決策:

ロード時間を短縮するには、キーフレーム間隔を短くすることを検討してください。これによりフラグメントが短くなり、レイテンシーが短縮されますが、ビデオファイルのサイズも大きくなります。

x264enc GStreamer 要素の場合、key-int-maxプロパティを使用してキーフレーム間隔を明示的に設定できます。

```
x264enc bframes=0 key-int-max=60
```

ログ出力を確認するときは、アップロードするクライアントが Kinesis Video Streams から ACKs を受信する頻度に注意してください。生成されるキーフレームが多いほど、返される ACKs が増えます。

メディアが歪んでいるか、アーティファクトがある

この問題のトラブルシューティングを行うには、すべてのケーブルがしっかりと接続されていることを確認してください。カメラモジュールの libcamera-hello (またはレガシー Pi カメラ raspistill の場合) の出力を確認します。

GStreamer パイプラインで、kvssink を autovideosink または matroskamux とに置き換えます filesink。例えば、次のようになります。

```
... x264enc tune=zerolatency speed-preset=ultrafast bframes=0 key-int-max=60 byte-stream=true ! h264parse ! matroskamux ! filesink location=output.mkv
```

filesink または の使用時に開くメディアプレーヤーの出力ファイルを確認して autovideosink、アーティファクトが存在するかどうかを確認します。

また、次のパイプラインの出力を確認します。

```
gst-launch-1.0 autovideosrc ! videoconvert ! autovideosink
```

[デワープ](#)などの要素をパイプラインに追加すると、魚眼カメラの出力を修正できます。

カメラでサポートされている出力コーデックを確認し、必要に応じて要素を調整します。

たとえば、USB カメラが JPEG 出力のみをサポートしている場合、jpegparse および jpegdec 要素を使用してメディアを変換してから、を使用して H.264 にエンコードする必要があります x264enc。同様のパイプラインやウェブカメラのセットアップを持つ他のユーザー向けの GStreamer フォーラムでサポートを探します。

C++ プロデューサー SDK for Raspberry Pi のビルドに関する問題のトラブルシューティング

ビルドの問題が発生し、異なる CMake 引数を試す場合は、必ずクリーンなビルドを実行してください。再試行する前に open-source、dependency、および build フォルダを削除します。

OpenSSL に関する問題をビルドする

次のような出力を受信した場合、OpenSSL がシステムアーキテクチャを誤って検出したことを示します。

```
crypto/md5/md5-aarch64.S: Assembler messages:
```

```
crypto/md5/md5-aarch64.S:3: Error: unrecognized symbol type ""  
crypto/md5/md5-aarch64.S:6: Error: bad instruction `stp x19,x20,[sp,#-80]!'  
crypto/md5/md5-aarch64.S:7: Error: bad instruction `stp x21,x22,[sp,#16]'  
crypto/md5/md5-aarch64.S:8: Error: bad instruction `stp x23,x24,[sp,#32]'  
crypto/md5/md5-aarch64.S:9: Error: bad instruction `stp x25,x26,[sp,#48]'
```

この例では、この Raspberry Pi が実際に 32 ビットである場合、64 ビットバージョン (linux-aarch64) を構築しようとしています。一部の Raspberry Pi デバイスには 64 ビットカーネルがありますが、32 ビットのユーザースペースがあります。

OpenSSL が構築しようとしているアーキテクチャを決定します。OpenSSL のconfigureステップ中にログ行を見つけることができます。

```
[ 33%] Performing update step for 'project_libopenssl'  
-- Already at requested tag: OpenSSL_1_1_1t  
[ 44%] No patch step for 'project_libopenssl'  
[ 55%] Performing configure step for 'project_libopenssl'  
Operating system: x86_64-whatever-linux2  
Configuring OpenSSL version 1.1.1t (0x1010114fL) for linux-x86_64  
Using os-specific seed configuration  
Creating configdata.pm  
Creating Makefile
```

システムのアーキテクチャを確認します。

- カーネルビットネスの確認: `run uname -m`
- ユーザースペースのビットネスを確認する: `run getconf LONG_BIT`

`cat /proc/cpuinfo` または `lscpu` コマンドを使用して CPU 情報を確認することもできます。

解決策:

この問題を解決するには、構築時に次の CMake 引数を追加して、OpenSSL が 32 ビット ARM アーキテクチャ用に正しく構築されるようにします。

```
-DBUILD_OPENSSL_PLATFORM=linux-armv4
```

GStreamer でのkvssinkロードに関する問題のトラブルシューティング

確認 GST_PLUGIN_PATH

現在のシェルセッションのGST_PLUGIN_PATH環境変数が、を含むディレクトリを指していることを確認しますkvssink。環境変数はセッション固有であるため、新しいセッションごとに設定する必要があります。この変更を永続的にするには、「GST_PLUGIN_PATH 環境変数の設定を含めるようにシェルの起動スクリプトを更新します」を参照してください。

エラー: 共有オブジェクトファイルを開くことができません: そのようなファイルやディレクトリはありません

エラーが発生した場合はCannot open shared object file: No such file or directory、次のコマンドを実行します。

```
gst-inspect-1.0 /path/to/libgstkvssink.so
```

次の出力が表示された場合は、動的リンカーがに必要なライブラリを見つけられないことを示しますkvssink。これは通常、次の理由で発生します。

- 構築された場所とは異なる場所kvssinkに移動する。
- 間違った CPU アーキテクチャのクロスコンパイル。
- 必要な依存関係がありません。

出力:

```
WARNING: erroneous pipeline: no element "kvssink"  
error while loading shared libraries: libcproducer.so: cannot open shared object file:  
No such file or directory
```

解決策:

移動したライブラリの場合は、欠落しているライブラリを含むディレクトリを に追加しますLD_LIBRARY_PATH。

元のリポジトリのルートディレクトリから、 findユーティリティを使用して欠落しているライブラリを見つけることができます。ターミナルで、次のように入力します。

```
find . -name "*libcproducer*"
```

出力:


```
./build/dependency/libkvscproducer/kvscproducer-src/libcproducer.so
```

Linux デバイスのファイルパス区切り文字は です:。以下のコマンドは、既存のLD_LIBRARY_PATH環境変数に新しいフォルダパスを追加し、以前の値を保持します。

ターミナルで、次のように入力します。

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/build/dependency/libkvscproducer/kvscproducer-src
```

Important

環境変数はセッション固有です。セッション間で変更を保持するには、シェルの起動スクリプトを変更します。

また、open-source/local/libを に追加する必要がある場合もあります\$LD_LIBRARY_PATH。

エラー: ./path/to/libcproducer.so.1: 無効な ELF ヘッダー

共有ライブラリのロード中にこのエラーが発生した場合は、シンボリックリンク () が壊れている可能性がありますsymlinks。ホストマシンのオペレーティングシステムがターゲットマシンのオペレーティングシステムと一致しない場合、シンボリックリンクが壊れる可能性があります。例えば、MacBook for a Raspberry Pi でのクロスコンパイルなどです。

もう 1 つの考えられる原因は、構築されたバイナリが間違ったアーキテクチャ用であったことです。たとえば、バイナリが x86 用に構築されている場合 (Raspberry Pi は ARM CPUsを使用します)。

エラーで指定されたライブラリの場所に移動し、 と入力ls -laしてライブラリを検査しますsymlinks。

レスポンス:

```
drwxr-xr-x 16 me staff 512 Sep 10 17:16 .
drwxr-xr-x 7 me staff 224 Jan 6 23:46 ..
drwxr-xr-x 4 me staff 128 Sep 10 17:16 engines-1.1
-rwxr-xr-x 1 me staff 2294496 Sep 10 17:16 libcrypto.1.1.so
-rw-r--r-- 1 me staff 4002848 Sep 10 17:16 libcrypto.a
lrwxr-xr-x 1 me staff 19 Sep 10 17:16 libcrypto.so -> libcrypto.1.1.so
```

```
-rwxr-xr-x  1 me  staff  631176 Sep 10 17:12 liblog4cplus-2.0.3.so
lrwxr-xr-x  1 me  staff      24 Sep 10 17:12 liblog4cplus.so -> liblog4cplus-2.0.3.so
-rwxr-xr-x  1 me  staff   1012 Sep 10 17:12 liblog4cplus.a
-rwxr-xr-x  1 me  staff  694328 Sep 10 17:12 liblog4cplusU-2.0.3.so
lrwxr-xr-x  1 me  staff      25 Sep 10 17:12 liblog4cplusU.dylib ->
liblog4cplusU-2.0.3.so
-rwxr-xr-x  1 me  staff   1017 Sep 10 17:12 liblog4cplusU.a
-rwxr-xr-x  1 me  staff  536416 Sep 10 17:16 libssl.1.1.1.so
-rw-r--r--  1 me  staff  795184 Sep 10 17:16 libssl.a
lrwxr-xr-x  1 me  staff     16 Sep 10 17:16 libssl.so -> libssl.1.1.1.so
drwxr-xr-x  6 me  staff    192 Sep 10 17:16 pkgconfig
```

上記のサンプル出力では、symlinksは壊れていません。壊れsymlinksた には、ターゲットを指す矢印はありません。

解決策:

シンボリックリンクを修正するには、次の 2 つのオプションがあります。

- 推奨: `ln` コマンドsymlinkを使用して を再作成します。タイプ:

```
ln -s /path/to/actual/library /path/to/symlink
```

- 実際のライブラリファイルをコピーし、と一致するように名前を変更しますsymlink。

Note

このオプションを使用すると、ストレージ使用量が増加します。

ベストプラクティスとして、Docker などのツールを使用して同じオペレーティングシステムでコンパイルし、クロスコンパイルの問題を回避します。

欠落している依存関係:

欠落しているライブラリ名が で始まる場合はlibkvs、上記の「移動したライブラリ」のセクションを参照して、ホストデバイスからターゲットデバイスに Kinesis Video Streams ライブラリをインストールします。

それ以外の場合は、 に従って [the section called “ソフトウェアのインストールの前提条件”](#)、すべてのオープンソースソフトウェアの前提条件がターゲットデバイスにインストールされていることを確認します。

エラーコードのリファレンス

このセクションには、[Kinesis Video Streams へのアップロード](#) のエラーおよびステータスコード情報が含まれています。

一般的な問題のソリューションについては、「[トラブルシューティング](#)」を参照してください。

トピック

- [PutFrame コールバックによって返されるエラーとステータスコード - プラットフォーム独立コード \(PIC\)](#)
- [PutFrame コールバックによって返されるエラーとステータスコード - C プロデューサーライブラリ](#)

PutFrame コールバックによって返されるエラーとステータスコード - プラットフォーム独立コード (PIC)

以下のセクションには、プラットフォーム独立コード (PIC) 内の PutFrame オペレーションのコールバックによって返されるエラーとステータス情報が含まれています。

トピック

- [クライアントライブラリによって返されるエラーコードとステータスコード](#)
- [期間ライブラリによって返されるエラーコードとステータスコード](#)
- [共通ライブラリによって返されるエラーコードとステータスコード](#)
- [ヒープライブラリによって返されるエラーコードとステータスコード](#)
- [MKVGen ライブラリによって返されるエラーコードとステータスコード](#)
- [トレースライブラリによって返されるエラーコードとステータスコード](#)
- [Utils ライブラリによって返されるエラーコードとステータスコード](#)
- [ビュースライブラリによって返されるエラーコードとステータスコード](#)

クライアントライブラリによって返されるエラーコードとステータスコード

次の表に、Kinesis Video Streams Client ライブラリの メソッドによって返されるエラーとステータス情報を示します。

コード	メッセージ	説明	推奨されるアクション
0x52000001	STATUS_MAX_STREAM_COUNT	ストリームの最大数に達しました。	「 プロデューサー SDK クォータ 」で説明するように、DeviceInfo で最大のストリーム数を指定します。
0x52000002	STATUS_MIN_STREAM_COUNT	最小ストリーム数エラー。	で 0 より大きいストリームの最大数を指定します DeviceInfo 。
0x52000003	STATUS_INVALID_DEVICE_NAME_LENGTH	無効なデバイス名の長さ。	で指定された文字での最大デバイス名の長さを参照してください プロデューサー SDK クォータ 。
0x52000004	STATUS_INVALID_DEVICE_INFO_VERSION	無効な DeviceInfo 構造バージョン。	構造体の正しいバージョンを指定します。
0x52000005	STATUS_MAX_TAG_COUNT	タグの最大数に達しました。	で指定されている現在の最大タグ数を参照してください プロデューサー SDK クォータ 。
0x52000006	STATUS_DEVICE_FINGERPRINT_LENGTH		
0x52000007	STATUS_INVALID_CALLBACKS_VERSION	無効な Callbacks 構造バージョン。	構造体の正しいバージョンを指定します。
0x52000008	STATUS_INVALID_STREAM_INFO_VERSION	無効な StreamInfo 構造バージョン。	構造体の正しいバージョンを指定します。

コード	メッセージ	説明	推奨されるアクション
0x52000009	STATUS_INVALID_STREAM_NAME_LENGTH	無効なストリーム名の長さ。	で指定された文字の最大ストリーム名の長さを参照してください プロデューサー SDK クォータ 。
0x5200000a	STATUS_INVALID_STORAGE_SIZE	無効なストレージサイズが指定されました。	バイト単位のストレージサイズは、 プロデューサー SDK クォータ で指定される制限内である必要があります。
0x5200000b	STATUS_INVALID_ROOT_DIRECTORY_LENGTH	ルートディレクトリの文字列の長さが無効です。	で指定されているルートディレクトリパスの最大長を参照してください プロデューサー SDK クォータ 。
0x5200000c	STATUS_INVALID_SPILL_RATIO	無効なスピル比率。	スピル率を 0～100 のパーセンテージで表します。
0x5200000d	STATUS_INVALID_STORAGE_INFO_VERSION	無効な StorageInfo 構造バージョン。	構造体の正しいバージョンを指定します。

コード	メッセージ	説明	推奨されるアクション
0x5200000e	STATUS_INVALID_STREAM_STATE	ストリームが現在のオペレーションを許可しない状態にあります。	通常、このエラーは、SDK がリクエストされたオペレーションの実行に必要な状態に達しなかった場合に発生します。たとえば、GetStreamingEndpoint API 呼び出しが失敗し、クライアントアプリケーションがこれを無視してストリームにフレームを送り続ける場合などに発生します。
0x5200000f	STATUS_SERVICE_CALLBACKS_MISSING	Callbacks 構造に一部の必須関数でエントリポイントの関数が欠落しています。	必須コールバックがクライアントアプリケーションに実装されていることを確認します。このエラーは、プラットフォーム独立コード (PIC) クライアントにのみ公開されます。C++ や他のより高レベルのラッパーはこの呼び出しに対応します。

コード	メッセージ	説明	推奨されるアクション
0x52000010	STATUS_SERVICE_CALL_NOT_AUTHORIZED_ERROR	権限がありません。	セキュリティトークン、証明書、セキュリティトークンの統合、有効期限を確認します。トークンに正しい関連付けられた権限があることを確認します。Kinesis Video Streams サンプルアプリケーションの場合、環境変数が正しく設定されていることを確認します。
0x52000011	STATUS_DESCRIBE_STREAM_CALL_FAILED	DescribeStream API エラー。	DescribeStream API 再試行エラーのあとにこのエラーが返されます。PIC クライアントは、再試行を停止した後にこのエラーを返します。
0x52000012	STATUS_INVALID_DESCRIBE_STREAM_RESPONSE	無効な DescribeStreamResponse 構造体。	DescribeStreamResultEvent に渡された構造体が null あるいは、無効な Amazon リソースネーム (ARN) のような無効な項目を含んでいます。
0x52000013	STATUS_STREAM_IS_BEING_DELETED_ERROR	ストリームが削除されています。	ストリームが削除されているため、API エラーが生じます。ストリームの使用中に他のプロセスがストリームを削除しようとしていないことを確認します。

コード	メッセージ	説明	推奨されるアクション
0x52000014	STATUS_SERVICE_CALL_INVALID_ARG_ERROR	サービス呼び出しに無効な引数が指定されています。	サービス呼び出し引数が無効な場合、または SDK が解釈できないエラーを検出した場合、バックエンドはこのエラーを返します。
0x52000015	STATUS_SERVICE_CALL_DEVICE_NOT_FOUND_ERROR	デバイスが見つかりませんでした。	使用中にデバイスが削除されていないことを確認します。
0x52000016	STATUS_SERVICE_CALL_DEVICE_NOT_PROVISIONED_ERROR	デバイスがプロビジョニングされていません。	デバイスがプロビジョニングされていることを確認します。
0x52000017	STATUS_SERVICE_CALL_RESOURCE_NOT_FOUND_ERROR	このサービスから汎用的なリソースが返されていません。	サービスがリソース (ストリームなど) を検出できない場合にこのエラーが発生します。これには、さまざまな場面での多様な意味を持つ場合がありますが、ストリームが作成される以前の API の使用状況が原因であることがよくあります。SDK を使用すると、ストリームが最初に作成されることを確認します。

コード	メッセージ	説明	推奨されるアクション
0x52000018	STATUS_INVALID_AUTH_LEN	無効な auth info の長さ。	プロデューサー SDK クォータ で指定されている現在の値を参照します。
0x52000019	STATUS_CREATE_STREAM_CALL_FAILED	CreateStream API 呼び出しに失敗しました。	エラー文字列でこのオペレーションが失敗した理由についての詳細情報を参照します。
0x5200002a	STATUS_GET_STREAMING_TOKEN_CALL_FAILED	GetStreamingToken の呼び出しに失敗しました。	エラー文字列でこのオペレーションが失敗した理由についての詳細情報を参照します。
0x5200002b	STATUS_GET_STREAMING_ENDPOINT_CALL_FAILED	GetStreamingEndpoint API 呼び出しに失敗しました。	エラー文字列でこのオペレーションが失敗した理由についての詳細情報を参照します。
0x5200002c	STATUS_INVALID_URI_LEN	GetStreamingEndpoint API から無効な長さの URI 文字列が返されます。	プロデューサー SDK クォータ で指定されている現在の最大値を参照します。
0x5200002d	STATUS_PUT_STREAM_CALL_FAILED	PutMedia API 呼び出しに失敗しました。	エラー文字列でこのオペレーションが失敗した理由についての詳細情報を参照します。

コード	メッセージ	説明	推奨されるアクション
0x5200002e	STATUS_STORE_OUT_OF_MEMORY	コンテンツストアがメモリ不足です。	コンテンツストアはストリーム間で共有され、全ストリーム + ~20% (最適化を考慮して) 分の最大時間を保存するために十分な容量を必要とします。ストレージをオーバーフローしないことは重要です。ストリームごとにストレージサイズとレイテンシー許容値を累積した最大時間の値を選択します。フレームがコンテンツビューウィンドウから外れたときに、単に配置されたときではなく、フレームを削除することをお勧めします (コンテンツストアのメモリ負荷)。これは、フレームを削除するとストリームプレッシャー通知コールバックが開始されるためです。これでアプリケーションがビットレートを低める、フレームをドロップするなどの適切な行為を行うためにアップストリームメディアコンポーネント (エンコーダーなど) を調整できます。

コード	メッセージ	説明	推奨されるアクション
0x5200002f	STATUS_NO_MORE_DATA_AVAILABLE	ストリームには現在利用可能なデータがこれ以上ありません。	これは、ネットワーキングスレッドによってサービスに送信されるフレームの消費よりメディアパイプラインが作成する量が遅い場合の潜在的な有効結果です。高レベルのクライアント (C++、Java、Android など) は内部で処理されるため、この警告は表示されません。
0x52000030	STATUS_INVALID_TAG_VERSION	無効な Tag 構造バージョン。	構造体の正しいバージョンを指定します。
0x52000031	STATUS_SERVICE_CALL_UNKNOWN_ERROR	ネットワーキングスタックから不明な、あるいは汎用的なエラーが返されます。	詳細情報については、ログを参照します。
0x52000032	STATUS_SERVICE_CALL_RESOURCE_IN_USE_ERROR	使用中のリソース。	サービスから返されます。詳細については、「Kinesis Video Streams API Reference」を参照してください。
0x52000033	STATUS_SERVICE_CALL_CLIENT_LIMIT_ERROR	クライアント制限。	サービスから返されます。詳細については、「Kinesis Video Streams API Reference」を参照してください。

コード	メッセージ	説明	推奨されるアクション
0x52000034	STATUS_SERVICE_CALL_DEVICE_LIMIT_ERROR	デバイス制限。	サービスから返されます。詳細については、「Kinesis Video Streams API Reference」を参照してください。
0x52000035	STATUS_SERVICE_CALL_STREAM_LIMIT_ERROR	ストリーム制限。	サービスから返されます。詳細については、「Kinesis Video Streams API Reference」を参照してください。
0x52000036	STATUS_SERVICE_CALL_RESOURCE_DELETED_ERROR	リソースが削除された、あるいは削除中です。	サービスから返されます。詳細については、「Kinesis Video Streams API Reference」を参照してください。
0x52000037	STATUS_SERVICE_CALL_TIMEOUT_ERROR	サービス呼び出しがタイムアウトしました。	特定のサービス API の呼び出しがタイムアウトの結果となりました。有効なネットワーク接続があることを確認します。PIC はオペレーションを自動的に再試行します。
0x52000038	STATUS_STREAM_READY_CALLBACK_FAILED	ストリームの準備完了通知。	非同期ストリームが作成されたことを示す通知が PIC からクライアントに送信されます。
0x52000039	STATUS_DEVICE_TAGS_COUNT_NON_ZERO_TAGS_NULL	無効なタグが指定されています。	タグ数はゼロではありませんが、タグは空です。タグが指定されているか、カウントがゼロであることを確認します。

コード	メッセージ	説明	推奨されるアクション
0x5200003a	STATUS_INVALID_STREAM_DESCRIPTION_VERSION	無効な StreamDescription 構造バージョン。	構造体の正しいバージョンを指定します。
0x5200003b	STATUS_INVALID_TAG_NAME_LEN	無効なタグ名の長さ。	プロデューサー SDK クォータ で指定されるタグ名の制限を参照します。
0x5200003c	STATUS_INVALID_TAG_VALUE_LEN	無効なタグ値の長さ。	プロデューサー SDK クォータ で指定されるタグ値の制限を参照します。
0x5200003d	STATUS_TAG_STREAM_CALL_FAILED	TagResource API は失敗しました。	TagResource API 呼び出しに失敗しました。ネットワーク接続の有効性を確認します。この失敗の詳細については、ログを参照してください。
0x5200003e	STATUS_INVALID_CUSTOM_DATA	無効なカスタムデータによる PIC API 呼び出し。	無効なカスタムデータが PIC API の呼び出しに指定されています。これは、PIC を直接使用するクライアントでのみ発生します。
0x5200003f	STATUS_INVALID_CREATE_STREAM_RESPONSE	無効な CreateStreamResponse 構造体。	この構造体あるいはそのメンバーフィールドが無効です (ARN が Null あるいは プロデューサー SDK クォータ で指定される名前より長い場合)。

コード	メッセージ	説明	推奨されるアクション
0x52000040	STATUS_CLIENT_AUTH_CALL_FAILED	クライアント認証の失敗。	PIC は、複数回の再試行後に適切な認証情報 (AccessKeyId または SecretAccessKey) を取得できませんでした。認証の統合を確認します。サンプルアプリケーションは環境変数を使用して、認証情報を C++ プロデューサーライブラリに渡します。
0x52000041	STATUS_GET_CLIENT_TOKEN_CALL_FAILED	セキュリティトークンを取得する呼び出しに失敗しました。	この状態は、PIC を直接使用するクライアントでのみ発生します。複数回の試行後、呼び出しはこのエラーで失敗します。
0x52000042	STATUS_CLIENT_PROVISION_CALL_FAILED	プロビジョニングエラー。	プロビジョニングは実装されていません。
0x52000043	STATUS_CREATE_CLIENT_CALL_FAILED	プロデューサークライアントの作成に失敗しました。	複数回の再試行後クライアントの作成に失敗すると、PIC は一般的なエラーを返します。
0x52000044	STATUS_CLIENT_READY_CALLBACK_FAILED	READY 状態のプロデューサークライアントの取得に失敗しました。	PIC が READY 状態に移行することに失敗すると、PIC ステートマシンによって返されます。このルート原因の詳細については、ログを参照してください。

コード	メッセージ	説明	推奨されるアクション
0x52000045	STATUS_TAG_CLIENT_CALL_FAILED	プロデューサークライアントの TagResource に失敗しました。	プロデューサークライアントの TagResource API 呼び出しに失敗しました。このルート原因の詳細については、ログを参照してください。
0x52000046	STATUS_INVALID_CREATE_DEVICE_RESPONSE	デバイスあるいはプロデューサーの作成に失敗しました。	高レベルの SDKs (C++ や Java など) では、デバイスまたはプロデューサー作成 API はまだ実装されていません。PIC を直接使用するクライアントは、結果通知を使用して失敗を示すことができます。
0x52000047	STATUS_ACK_TIMESTAMP_NOT_IN_VIEW_WINDOW	受信した ACK のタイムスタンプがビューに表示されません。	このエラーは、受信した ACK に対応するフレームがコンテンツビューウィンドウから落ちる場合に発生します。一般的に、これは ACK 配信が遅い場合に発生します。これは 湊谷として解釈され、ダウンリンクが低速であることを示します。
0x52000048	STATUS_INVALID_FRAGMENT_ACK_VERSION	無効な FragmentAck 構造バージョン。	FragmentAck 構造の正しいバージョンを指定します。

コード	メッセージ	説明	推奨されるアクション
0x52000049	STATUS_INVALID_TOKEN_EXPIRATION	無効なセキュリティトークン期限。	セキュリティトークンの有効期限は、将来の絶対タイムスタンプが現在のタイムスタンプよりも大きく、猶予期間がある必要があります。猶予期間の制限については、「 プロデューサー SDK クォータ 」を参照してください。
0x5200004a	STATUS_END_OF_STREAM	ストリームの終了 (EOS) インジケータです。	GetStreamData API 呼び出しで、現在のアップロード処理セッションは終了したことを示します。これは、セッションが終了あるいはエラーが発生した、あるいはセッショントークンが期限切れとなり、セッションが更新されている場合に発生します。
0x5200004b	STATUS_DUPLICATE_STREAM_NAME	ストリーム名が重複しています。	複数のストリームが同じストリーム名を持つことはできません。ストリームに一意の名前を選択します。

コード	メッセージ	説明	推奨されるアクション
0x5200004c	STATUS_INVALID_RETENTION_PERIOD	無効な保持期間。	StreamInfo 構造に無効な保持期間が指定されています。保持期間の有効な値範囲についての詳細は、「 プロデューサー SDK クォータ 」を参照してください。
0x5200004d	STATUS_INVALID_ACK_KEY_START	無効 FragmentAck 。	フラグメント ACK 文字列を解析できませんでした。無効なキー開始インジケータです。フラグメント ACK 文字列が壊れている可能性があります。これは自己修正できるため、このエラーは警告として捉えることができます。
0x5200004e	STATUS_INVALID_ACK_DUPLICATE_KEY_NAME	無効 FragmentAck 。	フラグメント ACK 文字列を解析できませんでした。複数のキーが同じ名前を持っています。フラグメント ACK 文字列が壊れている可能性があります。これは自己修正できるため、このエラーは警告として捉えることができます。

コード	メッセージ	説明	推奨されるアクション
0x5200004f	STATUS_INVALID_ACK_VALUE_START	無効 FragmentAck。	無効なキー値の開始インジケータにより、フラグメント ACK 文字列を解析できません。フラグメント ACK 文字列が壊れている可能性があります。これは自己修正できるため、このエラーは警告として捉えることができます。
0x52000050	STATUS_INVALID_ACK_VALUE_END	無効 FragmentAck。	無効なキー値の終了インジケータにより、フラグメント ACK 文字列を解析できません。フラグメント ACK 文字列が壊れている可能性があります。これは自己修正できるため、このエラーは警告として捉えることができます。
0x52000051	STATUS_INVALID_PARSED_ACK_TYPE	無効 FragmentAck。	無効な ACK 文字列が指定されたため、ACK フラグメントを解析できません。
0x52000052	STATUS_STREAM_HAS_BEEN_STOPPED	ストリームが停止されました。	ストリームが停止されましたが、フレームは引き続きストリームに処理されています。

コード	メッセージ	説明	推奨されるアクション
0x52000053	STATUS_INVALID_STREAM_METRICS_VERSION	無効な StreamMetrics 構造バージョン。	StreamMetrics 構造の正しいバージョンを指定します。
0x52000054	STATUS_INVALID_CLIENT_METRICS_VERSION	無効な ClientMetrics 構造バージョン。	ClientMetrics 構造の正しいバージョンを指定します。
0x52000055	STATUS_INVALID_CLIENT_READY_STATE	プロデューサーの初期化が READY 状態に到達できませんでした。	プロデューサークライアントの初期化中に、READY 状態に到達できませんでした。詳細については、ログを参照してください。
0x52000056	STATUS_STATE_MACHINE_STATE_NOT_FOUND	内部ステートマシンエラー。	公に表示されるエラーではありません。
0x52000057	STATUS_INVALID_FRAGMENT_ACK_TYPE	FragmentAck 構造で無効な ACK タイプが指定されています。	FragmentAck 構造にはパブリックヘッダーで定義される ACK タイプが含まれている必要があります。
0x52000058	STATUS_INVALID_STREAM_READY_STATE	内部ステートマシントランジションエラー。	公に表示されるエラーではありません。

コード	メッセージ	説明	推奨されるアクション
0x52000059	STATUS_CLIENT_FREE_BEFORE_STREAM	プロデューサーの解放後、ストリームオブジェクトが解放されます。	プロデューサーオブジェクトが解放されると、ストリームの解放が試行されます。これは、PIC を直接使用するクライアントでのみ発生します。
0x5200005a	STATUS_ALLOCATION_SIZE_SMALLER_THAN_REQUESTED	内部ストレージエラー。	コンテンツストアからの実際の割り当てサイズがパッケージ化されたフレームとフラグメントのサイズよりも小さいことを示す内部エラー。
0x5200005b	STATUS_VIEW_ITEM_SIZE_GREATER_THAN_ALLOCATION	内部ストレージエラー。	コンテンツビューの割り当てられる保存サイズがコンテンツストアの割り当てサイズより大きくなっています。
0x5200005c	STATUS_ACK_ERR_STREAM_READ_ERROR	ストリーム読み込みエラー ACK。	ACK がバックエンドから返した、ストリームの読み取りまたは解析エラーを示すエラー。これは一般的に、バックエンドがストリームの取得に失敗したときに発生します。通常の場合、自動再ストリーミングによってこのエラーを修正できます。

コード	メッセージ	説明	推奨されるアクション
0x5200005d	STATUS_ACK_ERR_FRAGMENT_SIZE_REACHED	フラグメントの最大サイズに達しました。	フラグメントの最大サイズ (バイト単位) は、 「プロデューサー SDK クォータ」 で定義されています。このエラーは、非常に大きなフレームがあるか、または管理可能なサイズのフラグメントを作成するキーフレームが存在しないことを示します。エンコーダーの設定を確認し、キーフレームが正しく生成されていることを確認します。非常に密度の高いストリームには、最大限のサイズを管理するためにフラグメントを短い時間で生成するようにエンコーダーを設定します。

コード	メッセージ	説明	推奨されるアクション
0x5200005e	STATUS_ACK_ERR_FRAGMENT_DURATION_REACHED	フラグメントの最大時間に達しました。	フラグメントの最大時間は、「 プロデューサー SDK クォータ 」で定義されています。このエラーは、1 秒間のフレームが非常に低い場合、または管理可能な時間のフラグメントを作成するキーフレームが存在しないことを示します。エンコーダーの設定を確認し、キーフレームが定期的に正しく生成されていることを確認します。
0x5200005f	STATUS_ACK_ERR_CONNECTION_DURATION_REACHED	接続の最大時間に達しました。	Kinesis Video Streams は プロデューサー SDK クォータ で指定されている最大の接続時間を適用します。プロデューサー SDK は、最大値に達する前にストリームまたはトークンを自動的にローテーションします。SDK を使用しているクライアントは、このエラーを受信しないでください。
0x52000060	STATUS_ACK_ERR_FRAGMENT_TIMESTAMP_NOT_MONOTONIC	タイムコードが一定間隔で増加しません。	プロデューサー SDK はタイムスタンプを適用するため、SDK を使用するクライアントはこのエラーを受信しません。

コード	メッセージ	説明	推奨されるアクション
0x52000061	STATUS_AC K_ERR_MUL TI_TRACK_MKV	MKV に複数のトラックがあります。	プロデューサー SDK は単一のトラックストリームを適用するため、SDK を使用するクライアントはこのエラーを受信しません。
0x52000062	STATUS_AC K_ERR_INV ALID_MKV_DATA	無効な MKV データ。	バックエンド MKV パーサーにストリームの解析エラーが発生しました。SDK を使用しているクライアントは、移行中にストリームが破損している場合、このエラーが発生する可能性があります。これは、バッファ圧力が SDK に部分的に送信されたテールフレームを強制的にドロップする場合にも発生する可能性があります。後者の場合は、FPS と解像度を減らすか、圧縮率を上げるか、(「バースト」ネットワークがある場合) 一時的なプレッシャーに対応するためにコンテンツストアとバッファ期間を大きくすることをお勧めします。

コード	メッセージ	説明	推奨されるアクション
0x52000063	STATUS_ACK_ERR_INVALID_PRODUCER_TIMESTAMP	無効なプロデューサータイムスタンプ。	プロデューサークロックに今後大きなドリフトがある場合に、サービスはACKにこのエラーを返します。より高レベルのSDK (Java や C++ など) は、システムクロックの一部のバージョンを使用して PIC の現在の時間コールバックに対応します。システムクロックが正しく設定されていることを確認します。PIC を直接使用するクライアントは、コールバック関数が正しいタイムスタンプを返すことを確認する必要があります。
0x52000064	STATUS_ACK_ERROR_STREAM_NOT_ACTIVE	非アクティブなストリーム。	ストリームが「アクティブ」状態にないときに、バックエンド API への呼び出しが実行されました。これは、クライアントがストリームを作成した直後にフレームを中にプッシュした場合に発生します。SDK はステートマシンおよびリカバリーメカニズムを通してこのシナリオを処理します。

コード	メッセージ	説明	推奨されるアクション
0x52000065	STATUS_AK_ERR_KMS_KEY_ACCESS_DENIED	AWS KMS アクセス拒否エラー。	アカウントに指定されたキーへのアクセスがない場合に返されるエラーです。
0x52000066	STATUS_AK_ERR_KMS_KEY_DISABLED	AWS KMS キーが無効になっています。	指定されたキーが無効になりました。
0x52000067	STATUS_AK_ERR_KMS_KEY_VALIDATION_ERROR	AWS KMS キー検証エラー。	一般的な検証エラー。詳細については、「AWS Key Management Service APIリファレンス https://docs.aws.amazon.com/kms/latest/APIReference/ 」を参照してください。
0x52000068	STATUS_AK_ERR_KMS_KEY_UNAVAILABLE	AWS KMS key は利用できません。	このキーは使用不可です。詳細については、「AWS Key Management Service APIリファレンス https://docs.aws.amazon.com/kms/latest/APIReference/ 」を参照してください。
0x52000069	STATUS_AK_ERR_KMS_KEY_INVALID_USAGE	KMS キーの使用が無効です。	AWS KMS key はこのコンテキストで使用するように設定されていません。詳細については、「AWS Key Management Service APIリファレンス https://docs.aws.amazon.com/kms/latest/APIReference/ 」を参照してください。

コード	メッセージ	説明	推奨されるアクション
0x5200006a	STATUS_ACK_ERR_KMS_KEY_INVALID_STATE	AWS KMS 無効な状態。	詳細については、「AWS Key Management Service APIリファレンス https://docs.aws.amazon.com/kms/latest/APIReference/ 」を参照してください。
0x5200006b	STATUS_ACK_ERR_KMS_KEY_NOT_FOUND	KMS キーが見つかりません。	このキーが見つかりません。詳細については、「AWS Key Management Service APIリファレンス https://docs.aws.amazon.com/kms/latest/APIReference/ 」を参照してください。
0x5200006c	STATUS_ACK_ERR_STREAM_DELETED	ストリームが削除された、または削除中です。	ストリームが別のアプリケーションまたは AWS マネジメントコンソールで削除されています。
0x5200006d	STATUS_ACK_ERR_INTERNAL_ERROR	Internal error。	一般的サービス内部エラー。
0x5200006e	STATUS_ACK_ERR_FRAGMENT_ARCHIVAL_ERROR	フラグメントのアーカイブエラー。	サービスが永続的に継続し、フラグメントをインデックスできないときにこのエラーが返されます。稀に生じるエラーですが、これはさまざまな理由により発生します。デフォルトでは、SDK はフラグメントの送信を再試行します。

コード	メッセージ	説明	推奨されるアクション
0x5200006f	STATUS_ACK_ERR_UNKNOWN_ACK_ERROR	未知のエラー。	サービスによって不明なエラーが返されました。
0x52000070	STATUS_MISSING_ERR_ACK_ID	ACK 情報の欠落。	ACK パーサーは解析を完了しましたが、FragmentAck 情報が欠落しています。
0x52000071	STATUS_INVALID_ACK_SEGMENT_LEN	無効な ACK セグメントの長さ。	無効な長さの ACK セグメント文字列が ACK パーサーで指定されています。詳細については、「 プロデューサー SDK クォータ 」を参照してください。
0x52000074	STATUS_MAX_FRAGMENT_METADATA_COUNT	フラグメントには最大数のメタデータ項目が追加されています。	Kinesis のビデオストリームには、非永続的項目をフラグメントに追加、または永続的項目をメタデータキューに追加することにより、メタデータ項目を 10 個までフラグメントに追加できます。詳細については、「 Kinesis Video Streams でのストリーミングメタデータの使用 」を参照してください。

コード	メッセージ	説明	推奨されるアクション
0x52000075	STATUS_ACK_ERR_FRAGMENT_METADATA_LIMIT_REACHED	制限 (メタデータの最大個数、メタデータの名前の長さ、またはメタデータの値の長さ) に達しました。	プロデューサー SDK では、メタデータ項目の個数とサイズが制限されます。このエラーは、プロデューサー SDK コードの制限が変更されない限り発生しません。詳細については、「 Kinesis Video Streams でのストリーミングメタデータの使用 」を参照してください。
0x52000076	STATUS_BLOCKING_PUT_INTERRUPTED_STREAM_TERMINATED	実装されていません。	
0x52000077	STATUS_INVALID_METADATA_NAME	メタデータの名前が不正です。	メタデータ名は文字列 AWS「」で始めることはできません。このエラーが発生した場合、メタデータ項目はフラグメントキューまたはメタデータキューに追加されません。詳細については、「 Kinesis Video Streams でのストリーミングメタデータの使用 」を参照してください。

コード	メッセージ	説明	推奨されるアクション
0x52000078	STATUS_END_OF_FRAGMENT_FRAME_INVALID_STATE	フラグメントフレームの末尾が無効な状態です。	フラグメントの終了は、non-key-frameフラグメント化されたストリームで送信しないでください。
0x52000079	STATUS_TRACK_ACK_INFO_MISSING	トラック情報がありません。	トラック番号は 0 より大きく、トラック ID と一致する必要があります。
0x5200007a	STATUS_MAX_TRACK_COUNT_EXCEEDED	最大トラック数を超過しています。	ストリームごとに最大 3 つのトラックを設定できます。
0x5200007b	STATUS_OFFLINE_MODE_WITH_ZERO_RETENTION	オフラインストリーミングモードの保持期間を 0 に設定します。	オフラインストリーミングモードの保持時間を 0 に設定しないでください。
0x5200007c	STATUS_ACK_ERR_TRACK_NUMBER_MISMATCH	エラー ACK のトラック番号が一致していません。	
0x5200007d	STATUS_ACK_ERR_FRAMES_MISSING_FOR_TRACK	トラックのフレームがありません。	
0x5200007e	STATUS_ACK_ERR_MORE_THAN_ALLOWED_TRACKS_FOUND	許可される最大のトラック数を超過しました。	

コード	メッセージ	説明	推奨されるアクション
0x5200007f	STATUS_UPLOAD_HANDLED_ABORTED	アップロード処理は中止されます。	
0x52000080	STATUS_INVALID_CERT_PATH_LENGTH	証明書のパスの長さが無効です。	
0x52000081	STATUS_DUPLICATE_TRACK_ID_FOUND	重複するトラック ID が見つかりました。	
0x52000082	STATUS_INVALID_CLIENT_INFO_VERSION		
0x52000083	STATUS_INVALID_CLIENT_ID_STRING_LENGTH		
0x52000084	STATUS_SETTING_KEY_FRAME_FLAG_WHILE_USING_EOFR		
0x52000085	STATUS_MAXIMUM_FRAME_TIMESTAMP_DELTA_BETWEEN_TRACKS_EXCEEDED		

コード	メッセージ	説明	推奨されるアクション
0x52000086	STATUS_STREAM_SHUTTING_DOWN		
0x52000087	STATUS_CLIENT_SHUTTING_DOWN		
0x52000088	STATUS_PUTMEDIA_LAST_PERSIST_ACK_NOT_RECEIVED		
0x52000089	STATUS_NON_ALIGNED_HEAP_WITHIN_CONTENT_STORE_ALLOCATORS		
0x5200008a	STATUS_MULTIPLE_CONSECUTIVE_EOFR		
0x5200008b	STATUS_DUPLICATE_STREAM_EVENT_TYPE		
0x5200008c	STATUS_STREAM_NOT_STARTED		
0x5200008d	STATUS_INVALID_IMAGE_PREFIX_LENGTH		

コード	メッセージ	説明	推奨されるアクション
0x5200008e	STATUS_INVALID_METADATA_KEY_LENGTH		
0x5200008f	STATUS_INVALID_METADATA_VALUE_LENGTH		

期間ライブラリによって返されるエラーコードとステータスコード

次の表に、Durationライブラリ内のメソッドによって返されるエラーとステータス情報を示します。

コード	メッセージ
0xFFFFFFFFFFFFFFFF	INVALID_DURATION_VALUE

共通ライブラリによって返されるエラーコードとステータスコード

次の表に、Commonライブラリ内のメソッドによって返されるエラーとステータス情報を示します。

Note

このエラーと状態の情報コードは多くの API で共通です。

コード	先頭に 0 がないコード	メッセージ	説明
0x00000001	0x1	STATUS_NULL_ARG	NULL が必須の引数として渡されました。

コード	先頭に 0 がないコード	メッセージ	説明
0x00000002	0x2	STATUS_INVALID_ARG	引数に無効な値が指定されています。
0x00000003	0x3	STATUS_INVALID_ARG_LEN	無効な長さの引数が指定されています。
0x00000004	0x4	STATUS_NOT_ENOUGH_MEMORY	十分なメモリを割り当てることができませんでした。
0x00000005	0x5	STATUS_BUFFER_TOO_SMALL	指定されたバッファサイズが小さすぎます。
0x00000006	0x6	STATUS_UNEXPECTED_EOF	予期しないエンドオブファイルに達しました。
0x00000007	0x7	STATUS_FORMAT_ERROR	無効なフォーマットが発生しました。
0x00000008	0x8	STATUS_INVALID_HANDLE_ERROR	無効な処理値です。
0x00000009	0x9	STATUS_OPEN_FILE_FAILED	ファイルを開くことができませんでした。
0x0000000a	0xa	STATUS_READ_FILE_FAILED	ファイルの読み込みに失敗しました。
0x0000000b	0xb	STATUS_WRITE_TO_FILE_FAILED	ファイルの書き込みに失敗しました。

コード	先頭に 0 がないコード	メッセージ	説明
0x0000000c	0xc	STATUS_INTERNAL_ERROR	通常には発生しない内部エラーが生じ、SDK あるいはサービス API のバグである可能性があります。
0x0000000d	0xd	STATUS_INVALID_OPERATION	無効なオペレーションが発生した、またはこのオペレーションは許可されていません。
0x0000000e	0xe	STATUS_NOT_IMPLEMENTED	この機能は実装されていません。
0x0000000f	0xf	STATUS_OPERATION_TIMED_OUT	オペレーションがタイムアウトしました。
0x00000010	0x10	STATUS_NOT_FOUND	必要なリソースが見つかりませんでした。
0x00000011	0x11	STATUS_CREATE_THREAD_FAILED	スレッドの作成に失敗しました。
0x00000012	0x12	STATUS_THREAD_READ_NOT_ENOUGH_RESOURCES	別のスレッドを作成するリソースが不十分であるか、スレッド数にシステムが課す制限が発生しました。

コード	先頭に 0 がないコード	メッセージ	説明
0x00000013	0x13	STATUS_THREAD_READ_INVALID_ARG	無効なスレッド属性が指定されているか、別のスレッドが既にこのスレッドとの結合を待っています。
0x00000014	0x14	STATUS_THREAD_READ_PERMISSIONS	スレッド属性で指定されたスケジューリングポリシーとパラメータを設定するアクセス許可がありません。
0x00000015	0x15	STATUS_THREAD_READ_DEADLOCKED	デッドロックが検出されるか、結合スレッドが呼び出し元のスレッドを指定します。
0x00000016	0x16	STATUS_THREAD_READ_DOES_NOT_EXIST	指定されたスレッド ID を持つスレッドが見つかりません。
0x00000017	0x17	STATUS_JOIN_THREAD_FAILED	スレッド結合オペレーションから不明なエラーまたは一般的なエラーが返されました。
0x00000018	0x18	STATUS_WAIT_FAILED	条件変数を待機する最大時間を超えました。

コード	先頭に 0 がないコード	メッセージ	説明
0x00000019	0x19	STATUS_CANCEL_THREAD_FAILED	スレッドキャンセルオペレーションから不明なエラーまたは一般的なエラーが返されました。
0x0000001a	0x1a	STATUS_THREAD_NOT_JOINABLE	スレッド結合オペレーションは、結合不可能なスレッドでリクエストされます。
0x0000001b	0x1b	STATUS_DETACH_THREAD_FAILED	スレッドデタッチオペレーションから不明なエラーまたは一般的なエラーが返されました。
0x0000001c	0x1c	STATUS_THREAD_ATTR_INIT_FAILED	スレッド属性オブジェクトの初期化に失敗しました。
0x0000001d	0x1d	STATUS_THREAD_ATTR_SET_STACK_SIZE_FAILED	スレッド属性オブジェクトのスタックサイズを設定できませんでした。
0x0000001e	0x1e	STATUS_MEMORY_NOT_FREED	テストでのみ使用されます。リクエストされたすべてのメモリが解放されていないことを示します。

コード	先頭に 0 がないコード	メッセージ	説明
0x0000001f	0x1f	STATUS_INVALID_THREAD_PARAM_VERSION	無効な「ThreadParams」構造バージョン。構造体の正しいバージョンを指定します。

ヒープライブラリによって返されるエラーコードとステータスコード

次の表に、Heapライブラリ内のメソッドによって返されるエラーとステータス情報を示します。

コード	メッセージ	説明
0x10000001	STATUS_HEAP_FLAGS_ERROR	無効なフラグの組み合わせが指定されています。
0x10000002	STATUS_HEAP_NOT_INITIALIZED	ヒープが初期化される前にオペレーションが試行されました。
0x10000003	STATUS_HEAP_CORRUPTED	ヒープが破損している、またはガードバンド (デバッグモード) が上書きされました。クライアントコードのバッファのオーバーフローはヒープの破損を引き起こす場合があります。
0x10000004	STATUS_HEAP_VRAM_LIB_MISSING	VRAM (ビデオ RAM) ユーザーまたはカーネルモードライブラリをロードできないか、欠落しています。基盤のプラットフォームが VRAM の割り当てをサポートしていることを確認します。

コード	メッセージ	説明
0x10000005	STATUS_HEAP_VRAM_L IB_REOPEN	VRAM ライブラリを開くことができませんでした。
0x10000006	STATUS_HEAP_VRAM_I NIT_FUNC_SYMBOL	INIT 関数エクスポートのロードに失敗しました。
0x10000007	STATUS_HEAP_VRAM_A LLOC_FUNC_SYMBOL	ALLOC 関数エクスポートのロードに失敗しました。
0x10000008	STATUS_HEAP_VRAM_F REE_FUNC_SYMBOL	FREE 関数エクスポートのロードに失敗しました。
0x10000009	STATUS_HEAP_VRAM_L OCK_FUNC_SYMBOL	LOCK 関数エクスポートのロードに失敗しました。
0x1000000a	STATUS_HEAP_VRAM_U NLOCK_FUNC_SYMBOL	UNLOCK 関数エクスポートのロードに失敗しました。
0x1000000b	STATUS_HEAP_VRAM_U NINIT_FUNC_SYMBOL	UNINIT 関数エクスポートのロードに失敗しました。
0x1000000c	STATUS_HEAP_VRAM_G ETMAX_FUNC_SYMBOL	GETMAX 関数エクスポートのロードに失敗しました。
0x1000000d	STATUS_HEAP_DIRECT _MEM_INIT	ハイブリッドヒープで主要なヒーププールの初期化に失敗しました。
0x1000000e	STATUS_HEAP_VRAM_I NIT_FAILED	VRAM の動的初期化に失敗しました。
0x1000000f	STATUS_HEAP_LIBRAR Y_FREE_FAILED	VRAM ライブラリの割り当て解除と解放に失敗しました。
0x10000010	STATUS_HEAP_VRAM_A LLOC_FAILED	VRAM の割り当てに失敗しました。

コード	メッセージ	説明
0x10000011	STATUS_HEAP_VRAM_FREE_FAILED	VRAM の解放に失敗しました。
0x10000012	STATUS_HEAP_VRAM_MAP_FAILED	VRAM マッピングに失敗しました。
0x10000013	STATUS_HEAP_VRAM_UNMAP_FAILED	VRAM マッピング解除に失敗しました。
0x10000014	STATUS_HEAP_VRAM_UNINIT_FAILED	VRAM の初期化解除に失敗しました。
0x10000015	STATUS_INVALID_ALLOCATION_SIZE	
0x10000016	STATUS_HEAP_REALLOC_ERROR	
0x10000017	STATUS_HEAP_FILE_HEAP_FILE_CORRUPT	

MKVGen ライブラリによって返されるエラーコードとステータスコード

次の表に、MKVGenライブラリ内のメソッドによって返されるエラーとステータス情報を示します。

コード	メッセージ	説明/推奨されるアクション
0x32000001	STATUS_MKV_INVALID_FRAME_DATA	Frame データ構造の無効なメンバー。期間、サイズ、フレームデータが有効であり、で指定された制限内であることを確認します プロデューサー SDK クォータ 。
0x32000002	STATUS_MKV_INVALID_FRAME_TIMESTAMP	無効なフレームタイムスタンプ。計算された PTS (プレゼ

コード	メッセージ	説明/推奨されるアクション
		ンテーションタイムスタンプ) および DTS (デコードタイムスタンプ) がフラグメントの開始フレームのタイムスタンプ以上です。これは、潜在的なメディアパイプラインあるいはエンコーダーの安定性の問題を指摘しています。トラブルシューティング情報については、「 エラー: 「Kinesis Video クライアントにフレームを送信できませんでした」 」を参照してください。
0x32000003	STATUS_MKV_INVALID_CLUSTER_DURATION	無効なフラグメント時間が指定されています。詳細については、「 プロデューサー SDK クォータ 」を参照してください。
0x32000004	STATUS_MKV_INVALID_CONTENT_TYPE_LENGTH	無効なコンテンツタイプ文字列の長さ。詳細については、「 プロデューサー SDK クォータ 」を参照してください。
0x32000005	STATUS_MKV_NUMBER_TOO_BIG	EBML (拡張可能なバイナリメタ言語) 形式で表記するには大きすぎる数字をエンコードしようとしています。これは、SDK クライアントには公開されません。
0x32000006	STATUS_MKV_INVALID_CODEC_ID_LENGTH	無効なコーデック ID 文字列の長さ。詳細については、「 プロデューサー SDK クォータ 」を参照してください。

コード	メッセージ	説明/推奨されるアクション
0x32000007	STATUS_MKV_INVALID_TRACK_NAME_LENGTH	無効なトラック名文字列の長さ。詳細については、「 プロデューサー SDK クォータ 」を参照してください。
0x32000008	STATUS_MKV_INVALID_CODEC_PRIVATE_LENGTH	無効なコーデックプライベートデータの長さ。詳細については、「 プロデューサー SDK クォータ 」を参照してください。
0x32000009	STATUS_MKV_CODEC_PRIVATE_NULL	コーデックプライベートデータ (CPD) は NULL ですが、CPD サイズは 0 より大きいです。
0x3200000a	STATUS_MKV_INVALID_TIMECODE_SCALE	無効なタイムコードスケール値です。詳細については、「 プロデューサー SDK クォータ 」を参照してください。
0x3200000b	STATUS_MKV_MAX_FRAME_TIMECODE	フレームタイムコードは最大値よりも大きい値である必要があります。詳細については、「 プロデューサー SDK クォータ 」を参照してください。

コード	メッセージ	説明/推奨されるアクション
0x3200000c	STATUS_MKV_LARGE_FRAME_TIMECODE	最大フレームタイムコードに到達しています。MKV 形式は、フレームの相対的なタイムコードとしてクラスターの先頭に 16 ビット符号を使用します。フレームタイムコードが表現できない場合、エラーが生成されます。このエラーは、不正なタイムコードスケールが選択されている、またはクラスター時間が長すぎるため、表現するフレームのタイムコードが 16 ビット符号スペースをオーバーフローすることを示唆しています。

コード	メッセージ	説明/推奨されるアクション
0x3200000d	STATUS_MKV_INVALID_ANNEXB_NALU_IN_FRAME_DATA	無効な Annex-B 開始コードが発生しました。たとえば、Annex-B 順応フラグが指定され、コードに 3 つ以上のゼロがある無効な開始シーケンスが発生した場合などです。有効な Annex-B 形式には、バイトストリームの 3 つ以上のゼロのシーケンスを回避するために、「エミュレーション防御」があります。詳細については、MPEG の仕様を参照してください。Android でのこのエラーの詳細については、「 Android での STATUS_MKV_INVALID_ANNEXB_NALU_IN_FRAME_DATA(0x3200000d) エラー 」を参照してください。
0x3200000e	STATUS_MKV_INVALID_AVCC_NALU_IN_FRAME_DATA	適応 AVCC フラグが指定されている場合、AVCC NALU パッケージングが無効です。バイトストリームが有効な AVCC 形式であることを確認します。詳細については、MPEG の仕様を参照してください。
0x3200000f	STATUS_MKV_BOTH_ANNEXB_AND_AVCC_SPECIFIED	適応する AVCC と Annex-B NALUs ました。いずれか 1 つを指定、あるいは指定なしにします。

コード	メッセージ	説明/推奨されるアクション
0x32000010	STATUS_MKV_INVALID _ANNEXB_NALU_IN_CPD	順応 Annex-B フラグが指定されているときの CPD の無効な Annex-B 形式。CPD が有効な Annex-B 形式であることを確認します。そうでない場合は、CPD Annex-B 適応フラグを削除します。
0x32000011	STATUS_MKV_PTS_DTS _ARE_NOT_SAME	Kinesis Video Streams は、PTS (プレゼンテーションタイムスタンプ) および DTS (デコードタイムスタンプ) に同じフラグメント開始フレームを適用します。これはフラグメントを開始するキーフレームです。
0x32000012	STATUS_MKV_INVALID _H264_H265_CPD	H264/H265 コーデックプライベートデータの貼り付けに失敗しました。
0x32000013	STATUS_MKV_INVALID _H264_H265_SPS_WIDTH	コーデックプライベートデータから幅を抽出できませんでした。
0x32000014	STATUS_MKV_INVALID _H264_H265_SPS_HEIGHT	コーデックプライベートデータから高さを抽出できませんでした。
0x32000015	STATUS_MKV_INVALID _H264_H265_SPS_NALU	H264/H265 SPS NALU が無効です。
0x32000016	STATUS_MKV_INVALID _BIH_CPD	コーデックプライベートデータの無効なビットマップ情報ヘッダー形式。

コード	メッセージ	説明/推奨されるアクション
0x32000017	STATUS_MKV_INVALID_HEVC_NALU_COUNT	高効率ビデオコーディング (HEVC) のネットワーク抽象化レイヤーユニット (NALU) 数が無効です。
0x32000018	STATUS_MKV_INVALID_HEVC_FORMAT	HEVC の形式が無効です。
0x32000019	STATUS_MKV_HEVC_SPS_NALU_MISSING	シーケンスパラメータセット (SPS) に HEVC NALU が見つかりません。
0x3200001a	STATUS_MKV_INVALID_HEVC_SPS_NALU_SIZE	HEVC SPS NALU サイズが無効です。
0x3200001b	STATUS_MKV_INVALID_HEVC_SPS_CHROMA_FORMAT_IDC	クロマ形式 IDC が無効です。
0x3200001c	STATUS_MKV_INVALID_HEVC_SPS_RESERVED	HEVC 予約 SPS が無効です。
0x3200001d	STATUS_MKV_MIN_ANNEX_B_CPD_SIZE	AnnexBb コーデックのプライベートベータ値の最小サイズ。H264 の場合、この値は 11 以上である必要があります。H265 の場合、この値は 15 以上である必要があります。
0x3200001e	STATUS_MKV_ANNEXB_CPD_MISSING_NALUS	Annex-B NALU のコーデックプライベートデータがありません。

コード	メッセージ	説明/推奨されるアクション
0x3200001f	STATUS_MKV_INVALID_ANNEXB_CPD_NALUS	Annex-B NALU のコーデックプライベートベータが無効です。
0x32000020	STATUS_MKV_INVALID_TAG_NAME_LENGTH	無効なタグ名の長さ。有効な値はゼロより大きく、128 未満です。
0x32000021	STATUS_MKV_INVALID_TAG_VALUE_LENGTH	無効なタグ値の長さ。有効な値は 0 より大きく 256 未満です。
0x32000022	STATUS_MKV_INVALID_GENERATOR_STATE_TAGS	ジェネレーター状態タグが無効です。
0x32000023	STATUS_MKV_INVALID_AAC_CPD_SAMPLING_FREQUENCY_INDEX	AAC コーデックのプライベートデータサンプリング頻度インデックスが無効です。
0x32000024	STATUS_MKV_INVALID_AAC_CPD_CHANNEL_CONFIG	AAC コーデックのプライベートデータチャンネル設定が無効です。
0x32000025	STATUS_MKV_INVALID_AAC_CPD	AAC コーデックのプライベートデータが無効です。
0x32000026	STATUS_MKV_TRACK_INFO_NOT_FOUND	トラック情報が見つかりませんでした。
0x32000027	STATUS_MKV_INVALID_SEGMENT_UUID	UUID セグメント UUID が無効です。
0x32000028	STATUS_MKV_INVALID_TRACK_UUID	トラック UUID が無効です。

コード	メッセージ	説明/推奨されるアクション
0x32000029	STATUS_MKV_INVALID_CLIENT_ID_LENGTH	
0x3200002a	STATUS_MKV_INVALID_AMS_ACM_CPD	
0x3200002b	STATUS_MKV_MISSING_SPS_FROM_H264_CPD	
0x3200002c	STATUS_MKV_MISSING_PPS_FROM_H264_CPD	
0x3200002d	STATUS_MKV_INVALID_PARENT_TYPE	

トレースライブラリによって返されるエラーコードとステータスコード

次の表に、Traceライブラリ内のメソッドによって返されるエラーとステータス情報を示します。

コード	メッセージ
0x10100001	STATUS_MIN_PROFILER_BUFFER

Utils ライブラリによって返されるエラーコードとステータスコード

次の表に、Utilsライブラリ内のメソッドによって返されるエラーとステータス情報を示します。

コード	メッセージ
0x40000001	STATUS_INVALID_BASE64_ENCODE
0x40000002	STATUS_INVALID_BASE
0x40000003	STATUS_INVALID_DIGIT
0x40000004	STATUS_INT_OVERFLOW

コード	メッセージ
0x40000005	STATUS_EMPTY_STRING
0x40000006	STATUS_DIRECTORY_OPEN_FAILED
0x40000007	STATUS_PATH_TOO_LONG
0x40000008	STATUS_UNKNOWN_DIR_ENTRY_TYPE
0x40000009	STATUS_REMOVE_DIRECTORY_FAILED
0x4000000a	STATUS_REMOVE_FILE_FAILED
0x4000000b	STATUS_REMOVE_LINK_FAILED
0x4000000c	STATUS_DIRECTORY_ACCESS_DENIED
0x4000000d	STATUS_DIRECTORY_MISSING_PATH
0x4000000e	STATUS_DIRECTORY_ENTRY_STAT_ERROR
0x4000000f	STATUS_STRFTIME_FALIED
0x40000010	STATUS_MAX_TIMESTAMP_FORMAT_STR_LEN_EXCEEDED
0x40000011	STATUS_UTIL_MAX_TAG_COUNT
0x40000012	STATUS_UTIL_INVALID_TAG_VERSION
0x40000013	STATUS_UTIL_TAGS_COUNT_NON_ZERO_TAGS_NULL
0x40000014	STATUS_UTIL_INVALID_TAG_NAME_LEN
0x40000015	STATUS_UTIL_INVALID_TAG_VALUE_LEN

コード	メッセージ
0x4000002a	STATUS_EXPONENTIAL_BACKOFF_INVALID_STATE
0x4000002b	STATUS_EXPONENTIAL_BACKOFF_RETRIES_EXHAUSTED
0x4000002c	STATUS_THREADPOOL_MAX_COUNT
0x4000002d	STATUS_THREADPOOL_INTERNAL_ERROR
0x40100001	STATUS_HASH_KEY_NOT_PRESENT
0x40100002	STATUS_HASH_KEY_ALREADY_PRESENT
0x40100003	STATUS_HASH_ENTRY_ITERATION_ABORT
0x41000001	STATUS_BIT_READER_OUT_OF_RANGE
0x41000002	STATUS_BIT_READER_INVALID_SIZE
0x41100001	STATUS_TIMER_QUEUE_STOP_SCHEDULING
0x41100002	STATUS_INVALID_TIMER_COUNT_VALUE
0x41100003	STATUS_INVALID_TIMER_PERIOD_VALUE
0x41100004	STATUS_MAX_TIMER_COUNT_REACHED
0x41100005	STATUS_TIMER_QUEUE_SHUTDOWN
0x41200001	STATUS_SEMAPHORE_OPERATION_AFTER_SHUTDOWN
0x41200002	STATUS_SEMAPHORE_ACQUIRE_WHEN_LOCKED

コード	メッセージ
0x41300001	STATUS_FILE_LOGGER_INDEX_FILE_INVALID_SIZE

ビューライブラリによって返されるエラーコードとステータスコード

次の表に、Viewライブラリ内のメソッドによって返されるエラーとステータス情報を示します。

コード	メッセージ	説明
0x30000001	STATUS_MIN_CONTENT_VIEW_ITEMS	無効なコンテンツビュー項目数が指定されています。詳細については、「 プロデューサー SDK クォータ 」を参照してください。
0x30000002	STATUS_INVALID_CONTENT_VIEW_DURATION	無効なコンテンツビュー時間が指定されています。詳細については、「 プロデューサー SDK クォータ 」を参照してください。
0x30000003	STATUS_CONTENT_VIEW_NO_MORE_ITEMS	ヘッド位置を超える試みが行われました。
0x30000004	STATUS_CONTENT_VIEW_INVALID_INDEX	無効なインデックスが指定されました。
0x30000005	STATUS_CONTENT_VIEW_INVALID_TIMESTAMP	無効なタイムスタンプがある、あるいはタイムスタンプが重複しています。フレームデコードタイムスタンプは、前のフレームタイムスタンプに前のフレーム期間を加えた値以上である必要があります`DTS(n) >= DTS(n-1)`

コード	メッセージ	説明
		+ Duration(n-1)`。このエラーは、多くの場合「不安定な」エンコーダーを示しています。エンコーダーはエンコードされたフレームを大量に生成し、タイムスタンプが内部フレーム時間よりも小さい値です。あるいは、ストリームが SDK のタイムスタンプを使用するように設定され、このフレームがフレーム時間よりも高速で送信されています。エンコーダーの一部の「不安定」を解消するには、StreamInfo.StreamCaps 構造でより短いフレーム時間を設定します。たとえば、ストリームが 25 FPS の場合、各フレームの再生時間は 40 ミリ秒です。ただし、エンコーダーの「ジッター」を処理するには、そのフレーム期間の半分 (20 ミリ秒) を使用することをお勧めします。一部のストリームでは、エラーを検出するためにより正確な時間制御が必要となります。
0x30000006	STATUS_INVALID_CONTENT_VIEW_LENGTH	無効なコンテンツビュー項目データの長さが指定されています。

PutFrame コールバックによって返されるエラーとステータスコード - C プロデューサーライブラリ

次のセクションには、C プロデューサーライブラリ内の PutFrame オペレーションのコールバックによって返されるエラーとステータス情報が含まれています。

コード	メッセージ	説明	推奨されるアクション
0x15000001	STATUS_STOP_CALLBACK_CHAIN	コールバックチェーンが停止しました。	
0x15000002	STATUS_MAX_CALLBACK_CHAIN	コールバックチェーンの最大値に達しました。	
0x15000003	STATUS_INVALID_PLATFORM_CALLBACKS_VERSION	無効な PlatformCallbacks 構造バージョン。	構造体の正しいバージョンを指定します。
0x15000004	STATUS_INVALID_PRODUCER_CALLBACKS_VERSION	無効な ProducerCallbacks 構造バージョン。	構造体の正しいバージョンを指定します。
0x15000005	STATUS_INVALID_STREAM_CALLBACKS_VERSION	無効な StreamCallbacks 構造バージョン。	構造体の正しいバージョンを指定します。
0x15000006	STATUS_INVALID_AUTH_CALLBACKS_VERSION	無効な AuthCallbacks 構造バージョン。	構造体の正しいバージョンを指定します。

コード	メッセージ	説明	推奨されるアクション
0x15000007	STATUS_INVALID_API_CALLBACKS_VERSION	無効な ApiCallbacks 構造バージョン。	構造体の正しいバージョンを指定します。
0x15000008	STATUS_INVALID_AWS_CREDENTIALS_VERSION	無効な AwsCredentials 構造バージョン。	構造体の正しいバージョンを指定します。
0x15000009	STATUS_MAX_REQUEST_HEADER_COUNT	リクエストヘッダーカウントの最大値に達しました。	
0x1500000a	STATUS_MAX_REQUEST_HEADER_NAME_LEN	リクエストヘッダー名の最大長に達しています。	
0x1500000b	STATUS_MAX_REQUEST_HEADER_VALUE_LEN	リクエストヘッダー値の最大長に達しています。	
0x1500000c	STATUS_INVALID_API_CALL_RETURN_JSON	API コールに無効な戻り値の JSON。	
0x1500000d	STATUS_CURL_INIT_FAILED	Curl の初期化に失敗しました。	
0x1500000e	STATUS_CURL_LIBRARY_INIT_FAILED	Curl lib 初期化に失敗しました。	

コード	メッセージ	説明	推奨されるアクション
0x1500000f	STATUS_INVALID_DESCRIBE_STREAM_RETURN_JSON	DescribeStream に無効な戻り値の JSON。	
0x15000010	STATUS_HMAC_GENERATION_ERROR	HMAC の生成エラー。	
0x15000011	STATUS_IOT_FAILED	IoT 認可に失敗しました。	
0x15000012	STATUS_MAX_ROLE_ALIAS_LEN_EXCEEDED	ロールエイリアスの最大長に達しました。	短いエイリアスの長さを指定してください。
0x15000013	STATUS_MAX_USER_AGENT_NAME_POSTFIX_LEN_EXCEEDED	エージェント名ポストフィックスの最大長に達しました。	
0x15000014	STATUS_MAX_CUSTOM_USER_AGENT_LEN_EXCEEDED	顧客のユーザーエージェントの最大長に達しました。	
0x15000015	STATUS_INVALID_USER_AGENT_LENGTH	無効なユーザーエージェントの長さ。	
0x15000016	STATUS_INVALID_END_POINT_CACHING_PERIOD	エンドポイントの無効なキャッシュ期間。	24 時間未満のキャッシュ期間を指定してください。

コード	メッセージ	説明	推奨されるアクション
0x15000017	STATUS_IOT_EXPIRATION_OCCURRENCE_IN_PAST	IoT の有効期限タイムスタンプは過去に発生します。	
0x15000018	STATUS_IOT_EXPIRATION_PARSING_FAILED	IoT の有効期限解析に失敗しました。	
0x15000019	STATUS_DUPLICATE_PRODUCER_CALLBACK_FREE_FUNC		
0x1500001a	STATUS_DUPLICATE_STREAM_CALLBACK_FREE_FUNC		
0x1500001b	STATUS_DUPLICATE_AUTH_CALLBACK_FREE_FUNC		
0x1500001c	STATUS_DUPLICATE_API_CALLBACK_FREE_FUNC		
0x1500001d	STATUS_FILE_LOGGER_INDEX_FILE_TOO_LARGE		

コード	メッセージ	説明	推奨されるアクション
0x1500001e	STATUS_MAX_IOT_THING_NAME_LENGTH		
0x1500001f	STATUS_IOT_CREATE_LWS_CONTEXT_FAILED		
0x15000020	STATUS_INVALID_CA_CERT_PATH		
0x15000022	STATUS_FILE_CREDENTIAL_PROVIDER_OPEN_FILE_FAILED		
0x15000023	STATUS_FILE_CREDENTIAL_PROVIDER_INVALID_FILE_LENGTH		
0x15000024	STATUS_FILE_CREDENTIAL_PROVIDER_INVALID_FILE_FORMAT		
0x15000026	STATUS_STREAM_BEING_SHUTDOWN		

コード	メッセージ	説明	推奨されるアクション
0x15000027	STATUS_CLIENT_BEING_SHUTDOWN		
0x15000028	STATUS_CONTINUOUS_RETRY_RESET_FAILED		
0x16000001	STATUS_CURL_PERFORM_FAILED	CURL は成功しないコードを返しました。	追加情報については、ログを確認してください。一般的な CURL エラーは「ホスト名を解決できませんでした」です。デバイスのインターネット接続を確認してください。 もう 1 つの一般的なエラーは、403 エラーコードです。これは、IoT 証明書が正しく作成または指定されていないことを示します。IoT 証明書へのファイルパスとアクセス許可が正しく設定されていることを確認します。詳細については「the section called “を使用した Kinesis Video Streams リソースへのアクセスの制御 AWS IoT”」を参照してください。

コード	メッセージ	説明	推奨されるアクション
0x16000002	STATUS_IOT_INVALID_RESPONSE_LENGTH	IoT 認証情報の取得時に 0 の長さのレスポンスを受信しました。	AWS ヘルスダッシュボードを確認して、後でもう一度試してください。
0x16000003	STATUS_IOT_NULL_AWS_CREDS	IoT 認証情報エンドポイントから返された JSON に認証情報オブジェクトが含まれていませんでした。	追加情報については、JSON の「メッセージ」項目を確認してください。
0x16000004	STATUS_IOT_INVALID_URI_LEN	IoT 認証情報の取得関数に渡される URL の長さは 1～10,000 ではありません。	この関数に渡される URL を確認します。
0x16000005	STATUS_TIMESTAMP_UNRECOGNIZED_FORMAT	IoT 認証情報の取得による JSON の「有効期限」項目は、 の形式ではありませんYYYY-MM-DDTHH:mm:SSZ 。	AWS ヘルスダッシュボードを確認して、後でもう一度試してください。

Network Abstraction Layer (NAL) 適応フラグを参照

このセクションでは、StreamInfo.NalAdaptationFlags 列挙に利用可能なフラグに関する情報が含まれています。

アプリケーションの [エレメンタリーストリーム](#) は、Annex-B または AVCC 形式のいずれかにすることができます。

- Annex-B 形式は、2 バイトのゼロで [NALU \(Network Abstraction Layer Units\)](#) を区切り、その後に 1 バイトまたは 3 バイトのゼロと数値 1 が続きます (開始コードと呼ばれる、00000001 など)。
- AVCC 形式はまた、NALU をラップしますが、各 NALU の前に NALU のサイズを示す値 (通常は 4 バイト) があります。

多くのエンコーダーは Annex-B ビットストリーム形式を作成します。一部の上位ビットストリームプロセッサ (の再生エンジンや [Media Source Extensions \(MSE\)](#) プレイヤーなど AWS マネジメントコンソール) は、フレームに AVCC 形式を使用します。

H.264 コーデックの SPS/PPS (シーケンスパラメータセット/ピクチャパラメータセット) であるコーデックプライベートデータ (CPD) は、Annex-B または AVCC 形式にすることもできます。ただし、CPD の場合、形式は前に説明したものとは異なります。

フラグは、次のように、SDK に指示し、フレームデータと CPD の NALU を AVCC または Annex-B に適応させるようにします。

フラグ	適応	
NAL_ADAPTATION_FLAG_NONE	適応なし。	
NAL_ADAPTATION_ANNEXB_NALS	Annex-B NALUs を AVCC NALUs。	
NAL_ADAPTATION_AVCC_NALS	AVCC NALUs を Annex-B NALUs。	
NAL_ADAPTATION_ANNEXB_CPD_NALS	コーデックプライベートデータの Annex-B NALUs を AVCC 形式の NALUs。	
NAL_ADAPTATION_ANNEXB_CPD_AND_FRAME_NALS	コーデックの Annex-B NALUs を適応させ、プライベートデータを AVCC 形式の NALUs。	

NALU タイプの詳細については、RFC 3984 の、[セクション 1.3: ネットワーク抽象化レイヤーユニットタイプ](#)を参照してください。

プロデューサー SDK 構造

このセクションでは、データを Kinesis Video Streams Producer オブジェクトに提供するために使用できる構造について説明します。

トピック

- [DeviceInfo/ DefaultDeviceInfoProvider](#)
- [StorageInfo](#)

DeviceInfo/ DefaultDeviceInfoProvider

DeviceInfo オブジェクトおよび DefaultDeviceInfoProvider オブジェクトは、Kinesis Video Streams Producer オブジェクトの動作を制御します。

メンバーフィールド

- version – 正しいバージョンの構造が現在のバージョンのコードベースで使用されていることを確認するために使用される整数値。現行バージョンは、DEVICE_INFO_CURRENT_VERSION マクロを使用して指定します。
- name – デバイスの人間が読み取れる名前。
- tagCount/tags – 現在使用されていません。
- streamCount – デバイスが処理できるストリームの最大数。これにより、最初にストリームを指すポインターのストレージが事前に割り当てられますが、実際のストリームオブジェクトは後で作成されます。デフォルトは 16 ストリームですが、この数は DefaultDeviceInfoProvider.cpp ファイルで変更できます。
- storageInfo: メインのストレージ設定を説明するオブジェクト。詳細については、「[StorageInfo](#)」を参照してください。

StorageInfo

Kinesis Video Streams のメインストレージの設定を指定します。

デフォルトの実装は、ストリーミング向けに最適化された、断片化の少ない高速なヒープ実装に基づきます。使用する MEMALLOC アロケータは、特定のプラットフォームで上書きできます。一部のプラットフォームにおける仮想メモリの割り当ては、物理ページでバッキングされません。メモリが使

用されると、仮想ページは物理ページでバッキングされます。これにより、ストレージの使用率が低いときは、システム全体のメモリ負荷が低くなります。

デフォルトのストレージサイズを次の式に基づいて計算します。DefragmentationFactor は 1.2 (20 パーセント) に設定する必要があります。

```
Size = NumberOfStreams * AverageFrameSize * FramesPerSecond * BufferDurationInSeconds *  
DefragmentationFactor
```

次の例では、デバイスに音声ストリームとビデオストリームがあります。音声ストリームには 1 秒あたり 512 サンプルがあり、各サンプルは平均 100 バイトです。ビデオストリームには 1 秒あたり 25 サンプルがあり、各サンプルは平均 10,000 バイトです。各ストリームのバッファ期間は 3 分です。

```
Size = (512 * 100 * (3 * 60) + 25 * 10000 * (3 * 60)) * 1.2 = (9216000 + 45000000) *  
1.2 = 65059200 = ~ 66MB.
```

デバイスに使用可能なメモリがある場合は、重大な断片化を避けるために、ストレージにメモリを追加することをお勧めします。

エンコードの複雑さが高い場合 (モーションが大きいためにフレームサイズが大きい場合)、または帯域幅が低い場合、ストレージサイズがすべてのストリームの完全なバッファに対応するのに十分であることを確認します。プロデューサーがメモリプレッシャーに達すると、ストレージオーバーフロープレッシャーコールバック () が出力されます StorageOverflowPressureFunc。ただし、コンテンツストアに使用可能なメモリがない場合は、Kinesis Video Streams 内に挿入されるフレームが破棄され、エラー (STATUS_STORE_OUT_OF_MEMORY = 0x5200002e) になります。詳細については、「[クライアントライブラリによって返されるエラーコードとステータスコード](#)」を参照してください。アプリケーションの確認 (ACK) が利用できない場合、または保持された ACK が遅延した場合にも発生する可能性があります。この場合、バッファは前のフレームがドロップアウトし始める前に「バッファ期間」容量までいっぱいになります。

メンバーフィールド

- version – 正しいバージョンの構造が現在のバージョンのコードベースで使用されることを確認するために使用される整数値。
- storageType – ストレージの基盤となるバッキングと実装を指定する DEVICE_STORAGE_TYPE 列挙。現在、サポートされている値は DEVICE_STORAGE_TYPE_IN_MEM のみです。将来の実装では DEVICE_STORAGE_TYPE_HYBRID_FILE がサポートされます。これは、ファイルに格納されたコンテンツストアにストレージがフォールバックすることを示します。

- `storageSize` – 事前割り当てするストレージサイズ。バイト単位。最小の割り当ては 10 MB です。最大の割り当ては 10 GB です。(今後ファイルに格納されるコンテンツストアの実装に伴って変更される予定です。)
- `spillRatio` – セカンダリオーバーフローストレージ (ファイルストレージ) ではなく、直接メモリストレージタイプ (RAM) から割り当てられるストレージの割合を表す整数値。現在使用されていません。
- `rootDirectory`: ファイルに格納されるコンテンツストアがあるディレクトリへのパス。現在使用されていません。

Kinesis ビデオストリーム構造

次の構造を使用して Kinesis のビデオストリームのインスタンスにデータを提供できます。

トピック

- [StreamDefinition/StreamInfo](#)
- [ClientMetrics](#)
- [StreamMetrics](#)

StreamDefinition/StreamInfo

C++ レイヤーの `StreamDefinition` オブジェクトは、プラットフォームに依存しないコードの `StreamInfo` オブジェクトをラップし、コンストラクタの一部のデフォルト値を提供します。

メンバーフィールド

フィールド	データ型	説明	デフォルトの値
<code>stream_name</code>	<code>string</code>	オプションのストリーム名。ストリーム名の長さの詳細については、「 ブロードユーザー SDK クォータ 」を参照してください。スト	名前を指定しないと、名前がランダムに生成されます。

フィールド	データ型	説明	デフォルトの値
		リームごとに一意の名前が必要です。	
retention_period	duration<uint64_t, ratio<3600>>	ストリームの保持期間 (秒単位)。0 の指定は、保持なしを示します。	3600 (1 時間)
[タグ]	const map<string, string>*	ユーザー情報を含むキー/値ペアのマップ。ストリームに既存のタグセットがある場合、新しいタグは既存のタグセットに追加されます。	タグがありません
kms_key_id	string	ストリームの暗号化に使用される AWS KMS キー ID。詳細については、「 Kinesis Video Streams でのデータ保護 」を参照してください。	デフォルト KMS キー (aws/kinesisvideo)。
streaming_type	STREAMING_TYPE 列挙	STREAMING_TYPE_REALTIME はサポートされる唯一の値です。	
content_type	string	ストリームのコンテンツ形式。Kinesis Video Streams コンソールは、video/h264 形式でコンテンツを再生できます。	video/h264

フィールド	データ型	説明	デフォルトの値
max_latency	duration<uint64_t, milli>	ストリームの最大レイテンシー (ミリ秒)。この時間をバッファ期間を超えると、ストリームのレイテンシープレッシャーコールバック (指定されている場合) が呼び出されます。 0 を指定すると、ストリームのレイテンシープレッシャーコールバックは呼び出されません。	milliseconds::zero()

フィールド	データ型	説明	デフォルトの値
fragment_duration	duration<uint64_t>	フラグメントの有効期間 (秒単位)。この値は、key_frame_fragmentation 値と組み合わせて使用します。この値が false の場合、この継続期間が経過すると、Kinesis Video Streams はキーフレームでフラグメントを生成します。たとえば、Advanced Audio Coding (AAC) オーディオストリームは各フレームをキーフレームとして使用します。key_frame_fragmentation = false を指定すると、この継続期間の経過後に、2 秒のフラグメントが生じます。	2

フィールド	データ型	説明	デフォルトの値
timecode_scale	duration<uint64_t, milli>	MKV タイムコードスケール (ミリ秒単位)。MKV クラスター内でフレームのタイムコードの詳細度を指定します。MKV フレームのタイムコードは、常にクラスターの開始を基準にします。MKV では、符号付きの 16 ビット値 (0 ～ 32767) を使用してクラスター内のタイムコード (フラグメント) を示します。フレームタイムコードが指定されたタイムコードスケールで表現できることを確認します。タイムコードスケール値のデフォルトである 1 ミリ秒の場合、表現できるフレームの最大値は 32,767 ミリ秒 ≈ 32 秒です。これは、 Amazon Kinesis Video Streams サークルウォータ で指定されたフラグメント継続時間の最大値 (10 秒) を超えます。	1

フィールド	データ型	説明	デフォルトの値
key_frame_fragmentation	bool	キーフレームでフラグメントを生成するかどうかを指定します。true の場合、SDK はキーフレームがあるたびにフラグメントの開始を生成します。false の場合、Kinesis Video Streams は少なくとも fragment_duration の期間待機してから、その後のキーフレームで新しいフラグメントを生成します。	true

フィールド	データ型	説明	デフォルトの値
frame_timecodes	bool	フレームのタイムコードを使用するか、現在時刻のコールバックを使用してタイムスタンプを生成するかどうかを指定します。多くのエンコーダーでは、フレームでタイムスタンプを生成しません。したがって、このパラメータに <code>false</code> を指定すると、確実にフレームがタイムスタンプ付きで Kinesis Video Streams に配置されます。	true

フィールド	データ型	説明	デフォルトの値
<code>absolute_fragment_times</code>	<code>bool</code>	Kinesis Video Streams では、基盤となるパッケージング機構として MKV を使用します。MKV の仕様では、クラスターの開始地点 (フラグメント) に相対するフレームのタイムコードは厳格です。ただし、クラスターのタイムコードはストリームの開始時刻に対して相対値の場合と絶対値の場合があります。タイムスタンプが相対値である場合、PutMedia サービスの API コールでは、オプションであるストリームの開始タイムスタンプを使用して、クラスターのタイムスタンプを調整します。このサービスで保存されるフラグメントには常に絶対値のタイムスタンプが使用されます。	<code>true</code>

フィールド	データ型	説明	デフォルトの値
fragment_acks	bool	アプリケーションレベルのフラグメント ACKs (承認) を受け取るかどうか。	true の場合、SDK は ACK を受け取り、相応に対応します。
restart_on_error	bool	特定のエラー発生時に再開するかどうかを指定します。	true の場合、SDK はエラー発生時にストリーミングの再開を試行します。
recalculate_metrics	bool	メトリクスを再計算するかどうかを指定します。メトリクスを取得するための呼び出しごとに、再計算によって最新の「実行中」の値が取得されます。これに伴って CPU に小さな影響が生じます。電力やフットプリントが極端に低いデバイスでは、これを false に設定して、CPU サイクルの消費を抑える必要があります。それ以外の場合は、この値 false にを使用することはお勧めしません。	true

フィールド	データ型	説明	デフォルトの値
nal_adaptation_flags	uint32_t	Network Abstraction Layer unit (NALU) 適応フラグを指定します。ビットストリームが H.264 でエンコードされている場合、NALU で未加工またはパッケージとして処理できます。これは Annex-B 形式または AVCC 形式のいずれかになります。ほとんどの基本ストリームプロデューサーとコンシューマー (読み取りエンコーダーとデコーダー) は、エラー復旧などの利点があるため、Annex-B 形式を使用します。高レベルのシステムでは AVCC 形式を使用します。これは、MPEG、HLS、DASH などのデフォルト形式です。コンソールの再生では、ブラウザの MSE (Media Source Extensions) を使用して、AVCC 形式を使用するストリームをデコードして再生し	デフォルトでは、フレームデータとコーデックプライベートデータの両方で Annex-B 形式を AVCC 形式に適應させます。

フィールド	データ型	説明	デフォルトの値
		ます。H.264 (および M-JPEG と H.265) の場合、SDK には適応機能が用意されています。 多くの基本ストリームは次の形式になります。この例では、Ab は Annex-B 開始コード (001 または 0001) です。 <pre>Ab(Sps)Ab (Pps)Ab(I- frame)Ab(P/B- frame) Ab(P/B-fr ame)... Ab(Sps)Ab (Pps)Ab(I- frame)Ab(P/B- frame) Ab(P/B-fr ame)</pre> H.264 の場合、コーデックプライベートデータ (CPD) は SPS (シーケンスパラメータセット) および PPS (ピクチャパラメータセット) パラメータにあり、AVCC 形式に適應できます。メディアパイプラインで個別に CPD を指定しない限り、アプリケーション	

フィールド	データ型	説明	デフォルトの値
		ンはフレームから CPD を抽出します。これを行うには、最初の IDR フレーム (SPS と PPS を含む必要があります) を探し、2 つの NALUs () を抽出 Ab(Sps)Ab (Pps) して、の CPD に設定します StreamDefinition 。 詳細については、「the section called “NAL 適応フラグ”」を参照してください。	
frame_rate	uint32_t	予想されるフレームレート。この値を使用してバッファリングニーズの計算を効率化できます。	25
avg_bandwidth_bps	uint32_t	ストリームの予想される平均帯域幅。この値を使用してバッファリングニーズの計算を効率化できます。	4 * 1024 * 1024

フィールド	データ型	説明	デフォルトの値
buffer_duration	duration<uint64_t>	ストリームのバッファ期間 (秒単位)。SDK は、最大のフレームをコンテンツストアに保持しbuffer_duration、その後、ウィンドウが進むにつれて前のフレームがドロップされます。ドロップされているフレームがバックエンドに送信されていない場合、ドロップされたフレームコールバックが呼び出されます。現在のバッファ期間がmax_latency より大きい場合、ストリームのレイテンシープレッシャーコールバックが呼び出されます。フラグメントの永続化 ACK が受信されると、バッファはトリミングされて次のフラグメント開始地点に送られます。これは、コンテンツがクラウド内で永続的に保持されるため、コンテンツをローカルデバイス	120

フィールド	データ型	説明	デフォルトの値
		に保存する必要がなくなるためです。	

フィールド	データ型	説明	デフォルトの値
replay_duration	duration<uint64_t>	再起動が有効になっている場合、エラー中に現在のリーダーをロールバックして再生する時間を秒単位で表します。ロールバックはバッファの開始位置で停止します (ストリーミングを開始したばかりであるか、永続化 ACK が受信された場合)。 ロールバックは、フラグメントの開始を示すキーフレームを確定しようとしします。「再起動の原因」というエラーがデッドホストを示していない場合 (ホストがまだ存続していて、内部バッファにフレームデータが含まれている場合)、ロールバックは最後に受信した ACK フレームで停止します。その後、次のキーフレームまでロールフォワードします (フラグメント全体がすでにホストメモリに保存されているため)。	40

フィールド	データ型	説明	デフォルトの値
connection_staleness	duration<uint64_t>	SDK がバッファリング ACK を受信しない場合にストリームの古さコールバックが呼び出される秒単位の時間。これは、フレームがデバイスから送信されているが、バックエンドがフレームを承認していないことを示します。この状況は、中間ホップまたはロードバランサーで接続が切断されていることを示します。	30
codec_id	string	MKV トラックのコーデック ID。	V_MPEG4/ISO/AVC
track_name	string	MKV トラック名。	kinesis_video

フィールド	データ型	説明	デフォルトの値
codecPrivateData	unsigned char*	コーデックプライベートデータ (CPD) のバッファ。ストリームの開始前に CPD に関する情報がメディアパイプラインにある場合は、StreamDefinition.codecPrivateData で設定できます。ビットがコピーされ、バッファを再利用できます。または、ストリームを作成するための呼び出し後にバッファが解放されます。ただし、ストリームの作成時にデータを使用できない場合は、KinesisVideoStream.start(cpd) 関数のオーバーロードのいずれかで設定できます。	null
codecPrivateDataSize	uint32_t	コーデックプライベートデータのバッファサイズ。	0

ClientMetrics

ClientMetrics オブジェクトを満たすには `getKinesisVideoMetrics` を呼び出します。

メンバーフィールド

フィールド	データ型	説明
version	UINT32	構造のバージョン。 CLIENT_METRICS_CURRENT_VERSION マクロで定義します。
contentStoreSize	UINT64	コンテンツストア全体のサイズ (バイト単位)。これは、DeviceInfo.StorageInfo.storageSize での指定値です。
contentStoreAvailableSize	UINT64	現在使用可能なストレージサイズはバイト単位です。
contentStoreAllocatedSize	UINT64	現在割り当てられているサイズ。割り当て済みのサイズ + 使用可能なサイズは、内部のブックキーピングとコンテンツストアの実装のため、ストレージ全体のサイズよりわずかに小さくなります。
totalContentViewsSize	UINT64	すべてのストリームですべてのコンテンツビューに割り当てられているメモリのサイズ。これはストレージサイズにはカウントされません。このメモリは MEMALLOC マクロを使用して割り当てられます。これを上書きしてカス

フィールド	データ型	説明
		タムアロケータを指定できません。
totalFrameRate	UINT64	すべてのストリームで確認された合計フレームレート。
totalTransferRate	UINT64	すべてのストリームで確認された合計ストリームレート (バイト/秒)。

StreamMetrics

StreamMetrics オブジェクトを満たすには `getKinesisVideoMetrics` を呼び出します。

メンバーフィールド

フィールド	データ型	説明
version	UINT32	構造のバージョン。 STREAM_METRICS_CURRENT_VERSION マクロで定義します。
currentViewDuration	UINT64	蓄積されたフレームの時間。高速ネットワーキングの場合、この期間は 0 またはフレーム期間 (フレームの送信中) のいずれかです。期間が <code>max_latency</code> 指定された期間よりも長くなると <code>StreamDefinition</code> 、ストリームレイテンシーコールバックが指定されている場合、ストリームレイテンシーコールバックが呼び出されます。時間は 100ns 単位で指

フィールド	データ型	説明
		定します。これは PIC レイヤーのデフォルトの時間単位です。
overallViewDuration	UINT64	全体の表示時間。ストリームに ACK や永続化が設定されていない場合、この値は Kinesis のビデオストリームに挿入されるフレームが増えるに従って増加し、StreamDefinition に指定された buffer_duration と等しくなります。ACKs が有効で、永続 ACK が受信されると、バッファは次のキーフレームに切り捨てられます。これは、ACK タイムスタンプがフラグメント全体の先頭を示すためです。時間は 100 ns 単位で指定します。これは PIC レイヤーのデフォルトの時間単位です。
currentViewSize	UINT64	現在のバッファのサイズ (バイト単位)。
overallViewSize	UINT64	全体の表示サイズ (バイト単位)。
currentFrameRate	UINT64	現在のストリームで確認されたフレームレート。
currentTransferRate	UINT64	すべてのストリームで確認された転送レート (バイト/秒)。

プロデューサー SDK コールバック

Amazon Kinesis Video Streams プロデューサー SDK のクラスとメソッドは、独自のプロセスを維持しません。その代わりに、受信した関数呼び出しとイベントを使用してコールバックをスケジュールし、アプリケーションと通信します。

アプリケーションが SDK とやり取りするために使用できるコールバックパターンは 2 つあります。

- **CallbackProvider** – このオブジェクトは、プラットフォームに依存しないコード (PIC) コンポーネントからのすべてのコールバックをアプリケーションに公開します。このパターンではすべての機能を使用できますが、実装では C++ レイヤーにあるすべてのパブリック API メソッドと署名を処理する必要があります。
- [StreamCallbackProvider](#) および [ClientCallbackProvider](#) – これらのオブジェクトは、ストリーム固有およびクライアント固有のコールバックを公開し、SDK の C++ レイヤーは残りのコールバックを公開します。これは、プロデューサー SDK とやり取りするために推奨されるコールバックパターンです。

次の図は、コールバックオブジェクトのオブジェクトモデルです。

前の図の DefaultCallbackProvider は CallbackProvider (PIC のすべてのコールバックを公開します) から派生し、StreamCallbackProvider および ClientCallbackProvider が含まれます。

このトピックには、次のセクションが含まれています。

- [ClientCallbackProvider](#)
- [StreamCallbackProvider](#)
- [ClientCallbacks 構造](#)
- [ストリーミングを再試行するためのコールバック実装](#)

ClientCallbackProvider

ClientCallbackProvider オブジェクトはクライアントレベルのコールバック関数を公開します。関数の詳細は「[the section called “ClientCallbacks”](#)」に記載されています。

コールバックメソッド:

- `getClientReadyCallback` – クライアントの準備完了状態をレポートします。
- `getStorageOverflowPressureCallback` – ストレージのオーバーフローまたはプレッシャーを報告します。このコールバックは、ストレージの使用率が以下の `STORAGE_PRESSURE_NOTIFICATION_THRESHOLD` 値 (ストレージ全体のサイズの 5 パーセント) に下がると呼び出されます。詳細については、「[StorageInfo](#)」を参照してください。

StreamCallbackProvider

`StreamCallbackProvider` オブジェクトはストリームレベルのコールバック関数を公開します。

コールバックメソッド:

- `getDroppedFragmentReportCallback`: 削除されたフラグメントを報告します。
- `getDroppedFrameReportCallback` – ドロップされたフレームをレポートします。
- `getFragmentAckReceivedCallback` – ストリームのフラグメント ACK が受信されたことをレポートします。
- `getStreamClosedCallback` – ストリームのクローズ状態を報告します。
- `getStreamConnectionStaleCallback` – 古い接続条件を報告します。この条件では、プロデューサーは サービスにデータを送信していますが、確認を受信していません。
- `getStreamDataAvailableCallback` – データがストリームで利用可能であることをレポートします。
- `getStreamErrorReportCallback` – ストリームエラー状態をレポートします。
- `getStreamLatencyPressureCallback` – ストリームレイテンシー条件をレポートします。これは、蓄積されたバッファサイズが `max_latency` 値より大きい場合です。詳細については、「[StreamDefinition/StreamInfo](#)」を参照してください。
- `getStreamReadyCallback`: – ストリーム準備完了状態を報告します。
- `getStreamUnderflowReportCallback` – ストリームのアンダーフロー条件をレポートします。この関数は現在使用されておらず、将来の使用のために予約されています。

`StreamCallbackProvider` のソースコードについては、[StreamCallbackProvider.h](#) を参照してください。

ClientCallbacks 構造

ClientCallbacks 構造には、特定のイベントが発生したときに PIC によって呼び出されるコールバック関数のエントリポイントが含まれています。またこの構造には、CALLBACKS_CURRENT_VERSION フィールドにバージョン情報が含まれるほか、個別のコールバック関数で返されるユーザー定義データが含まれる customData フィールドが含まれています。

クライアントアプリケーションは this ポインターを custom_data フィールドで使用できます。これは次のコード例のようにメンバー関数を実行時に静的 ClientCallback 関数にマッピングします。

```
STATUS TestStreamCallbackProvider::streamClosedHandler(UINT64 custom_data,
    STREAM_HANDLE stream_handle, UINT64 stream_upload_handle) {
    LOG_INFO("Reporting stream stopped.");

    TestStreamCallbackProvider* streamCallbackProvider =
        reinterpret_cast<TestStreamCallbackProvider*> (custom_data);
    streamCallbackProvider->streamClosedHandler(...);
```

Events

関数	説明	タイプ
CreateDeviceFunc	現在はバックエンドに実装されていません。この呼び出しは Java または C++ から呼び出されると失敗します。その他のクライアントはプラットフォーム固有の初期化を実行します。	バックエンド API
CreateStreamFunc	ストリームを作成したときに呼び出されます。	バックエンド API
DescribeStreamFunc	DescribeStream が呼び出されたときに呼び出されます。	バックエンド API

関数	説明	タイプ
GetStreamingEndpointFunc	GetStreamingEndpoint が呼び出されたときに呼び出されます。	バックエンド API
GetStreamingTokenFunc	GetStreamingToken が呼び出されたときに呼び出されます。	バックエンド API
PutStreamFunc	PutStream が呼び出されたときに呼び出されます。	バックエンド API
TagResourceFunc	TagResource が呼び出されたときに呼び出されます。	バックエンド API
CreateMutexFunc	同期ミューテックスを作成します。	同期
FreeMutexFunc	ミューテックスを解放します。	同期
LockMutexFunc	同期ミューテックスをロックします。	同期
TryLockMutexFunc	ミューテックスをロックするように試みます。現在実装されていません。	同期
UnlockMutexFunc	ミューテックスのロックを解除します。	同期
ClientReadyFunc	クライアントが準備完了状態になると呼び出されます。	Notification

関数	説明	タイプ
DroppedFrameReportFunc	フレームが削除されたときに報告されます。	Notification
DroppedFragmentReportFunc	フラグメントが削除されたときに報告されます。この関数は現在使用されておらず、将来の使用のために予約されています。	Notification
FragmentAckReceivedFunc	フラグメント ACK (バッファリング、受信、保持、エラー) が受信されたときに呼び出されます。	Notification
StorageOverflowPressureFunc	ストレージの使用率が STORAGE_PRESSURE_NOTIFICATION_THRESHOLD 値 (ストレージ全体のサイズの 5 パーセントとして定義) に下がると呼び出されます。	Notification
StreamClosedFunc	残りのフレームの最後のビットがストリーミングされたときに呼び出されます。	Notification
StreamConnectionStaleFunc	ストリームが古い接続状態になると呼び出されます。この状況では、プロデューサーはサービスにデータを送信していますが、送達確認を受信していません。	Notification
StreamDataAvailableFunc	ストリームデータが使用可能になったときに呼び出されます。	Notification

関数	説明	タイプ
StreamErrorReportFunc	ストリームエラーが発生したときに呼び出されます。この状況になると、PIC はストリームを自動的に閉じます。	Notification
StreamLatencyPressureFunc	ストリームがレイテンシー状態になったときに呼び出されます。蓄積されたバッファのサイズが <code>max_latency</code> 値より大きくなった場合です。詳細については、「 StreamDefinition/StreamInfo 」を参照してください。	Notification
StreamReadyFunc	ストリームが準備完了状態になると呼び出されます。	Notification
StreamUnderflowReportFunc	この関数は現在使用されておらず、将来の使用のために予約されています。	Notification
DeviceCertToTokenFunc	接続証明書をトークンとして返します。	プラットフォーム統合
GetCurrentTimeFunc	現在時刻を返します。	プラットフォーム統合
GetDeviceCertificateFunc	デバイス証明書を返します。この関数は現在使用されておらず、将来の使用のために予約されています。	プラットフォーム統合

関数	説明	タイプ
GetDeviceFingerprintFunc	デバイスフィンガープリントを返します。この関数は現在使用されておらず、将来の使用のために予約されています。	プラットフォーム統合
GetRandomNumberFunc	0 から RAND_MAX までの乱数を返します。	プラットフォーム統合
GetSecurityTokenFunc	バックエンド API と通信する関数に渡されるセキュリティトークンを返します。シリアル化された AccessKeyId 、 SecretKeyId 、およびセッショントークンを指定して実装できます。	プラットフォーム統合
LogPrintFunc	タグとログレベルを伴うテキスト行をログ記録します。詳細については、「PlatformUtils.h 」を参照してください。	プラットフォーム統合

前の表のプラットフォーム統合関数の最後のパラメータは ServiceCallContext 構造であり、以下のフィールドがあります。

- version: 構造のバージョン。
- callAfter: 関数を呼び出すまでの絶対時間。
- timeout: オペレーションのタイムアウト (100 ナノ秒単位)。
- customData: クライアントに返されるユーザー定義の値。
- pAuthInfo: 呼び出しの認証情報。詳細については、次の (__AuthInfo) 構造を参照してください。

認可情報は、シリアル化された認証情報またはプロバイダー固有の認証トークンのいずれかである `__AuthInfo` 構造を使用して提供されます。この構造には次のフィールドがあります。

- `version`: `__AuthInfo` 構造のバージョン。
- `type`: 認証情報のタイプを定義する `AUTH_INFO_TYPE` 値 (証明書またはセキュリティトークン)。
- `data`: 認証情報を含むバイト配列。
- `size`: `data` パラメータのサイズ。
- `expiration`: 認証情報の有効期限 (100 ナノ秒単位)。

ストリーミングを再試行するためのコールバック実装

Kinesis Video プロデューサー SDK は、コールバック関数を使用して、ストリーミングのステータスを提供します。ストリーミング中に発生した一時的なネットワーク問題から回復するには、次のコールバックメカニズムを実装することをお勧めします。

- ストリームレイテンシープレッシャーコールバック - このコールバックメカニズムは、SDK がストリームレイテンシー条件に遭遇したときに開始されます。このトリガーは、累積バッファサイズが `MAX_LATENCY` 値より大きい場合に発生します。ストリームが作成されると、ストリーミングアプリケーションによって `MAX_LATENCY` がデフォルト値の 60 秒に設定されます。このコールバックの一般的な実装として、接続をリセットします。必要に応じて、<https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp/blob/master/kinesis-video-c-producer/src/source/StreamLatencyStateMachine.c> のサンプル実装を使用することができます。ネットワーク停止による未配信のフレームを、バックフィル用のセカンダリストレージに保存することはできません。
- ストリームの古さコールバック - このコールバックは、プロデューサーが Amazon Kinesis Data Streams サービス (アップリンク) にデータを送信できるが、確認応答 (バッファされた ACK) を時間に戻すことができない (デフォルトは 60 秒) 場合に開始されます。ネットワーク設定に応じて、ストリームレイテンシープレッシャーコールバックまたはストリームの古さコールバック、またはその両方が開始されます。ストリームのレイテンシープレッシャーコールバックの再試行の実装と同様に、一般的な実装として、接続をリセットし、ストリーミング用に新しい接続を開始します。必要に応じて、<https://github.com/awslabs/amazon-kinesis-video-streams-producer-c/blob/master/src/source/ConnectionStateStateMachine.c> のサンプル実装を使用することができます。
- ストリームエラーコールバック - このコールバックは、SDK が KVS API サービスコールの呼び出し中にネットワーク接続でタイムアウトやその他のエラーが発生したときに開始されます。

- ドロップフレームコールバック - このコールバックは、ネットワーク速度が遅いか、ストリームエラーが原因でストレージサイズがいっぱいになると開始されます。ネットワーク速度によってフレームがドロップされた場合は、ストレージサイズを増やすか、ビデオフレームサイズを減らすか、ネットワーク速度に合わせてフレームレートを下げます。

Kinesis Video Streams でのストリーミングメタデータの使用

Amazon Kinesis Video Streams プロデューサー SDK を使用して、Kinesis ビデオストリームの個々のフラグメントレベルでメタデータを埋め込むことができます。Kinesis Video Streams 内のメタデータは、変更可能なキーバリューのペアです。これを使用して、フラグメントの内容を記述したり、実際のフラグメントと一緒に転送する必要がある関連するセンサー読み取り値を埋め込んだり、その他のカスタムニーズを満たすことができます。メタデータは [the section called “GetMedia”](#) または [the section called “GetMediaForFragmentList”](#) API オペレーションの一部として使用できます。ストリームの保持期間中、フラグメントとともに保存されます。を消費するアプリケーションは、[を使用してメタデータに基づいて読み取り、処理、対応できます](#) [パーサーライブラリを使用してカメラからの出力を監視する](#)。

メタデータをストリーム内のフラグメントを埋め込むモードは 2 つあります。

- 非永続的 - 発生したビジネス固有の基準に基づいて、ストリーム内のフラグメントに 1 回限り、またはアドホックベースでメタデータをアタッチできます。一例として、動きを検出して、Kinesis のビデオストリームに送信する前にその動きを含む対応フラグメントにメタデータを追加するスマートカメラがあります。フラグメントには、以下の形式でメタデータを適用できます。Motion = true
- 永続 - 継続的なニーズに基づいて、ストリーム内の連続するフラグメントにメタデータをアタッチできます。一例として、Kinesis のビデオストリームに送信するすべてのフラグメントに関連付けられた現在の緯度と経度の座標を送信するスマートカメラがあります。すべてのフラグメントには、以下の形式でメタデータを適用できます。Lat = 47.608013N , Long = -122.335167W

アプリケーションのニーズに基づいて、同一のフラグメントに対して同時にこのモードの両方でメタデータを付け加えられます。埋め込まれたメタデータには、検出されたオブジェクト、トラッキングされたアクティビティ、GPS 座標、またはその他のカスタムデータで、ストリームのフラグメントと関連付けるものが含まれる場合があります。メタデータはキーと値の文字列ペアとしてエンコードされます。

Kinesis ビデオストリームへのメタデータの追加

Kinesis のビデオストリームに追加するメタデータは MKV タグとしてモデル化され、キーバリューのペアとして実装されます。

メタデータは、ストリーム内のイベントをマークするなどの一時的なもの、またはあるイベントが発生したフラグメントを識別するなどの永続的なもののいずれかです。永続メタデータ項目は残り、キャンセルされるまで連続する各フラグメントに適用されます。

Note

[Kinesis Video Streams へのアップロード](#) を使用して追加されたメタデータ項目は、[the section called “TagStream”](#)、[the section called “UntagStream”](#)、および[the section called “ListTagsForStream”](#)を使って実装されたストリームレベルのタグ付け API とは異なります。

ストリーミングメタデータ API

プロデューサー SDK で次のオペレーションを使用して、ストリーミングメタデータを実装できます。

トピック

- [PIC](#)
- [C++ プロデューサー SDK](#)
- [Java プロデューサー SDK](#)
- [永続メタデータと非永続メタデータ](#)

PIC

```
PUBLIC_API STATUS putKinesisVideoFragmentMetadata(STREAM_HANDLE streamHandle,  
    PCHAR name,  
    PCHAR value,  
    BOOL persistent);
```

C++ プロデューサー SDK

```
/**
```

```
* Appends a "tag" or metadata - a key/value string pair into the stream.
*/
bool putFragmentMetadata(const std::string& name, const std::string& value, bool
    persistent = true);
```

Java プロデューサー SDK

Java プロデューサー SDK を使用して、MediaSourceを使用して にメタデータを追加できますMediaSourceSink.onCodecPrivateData。

```
void onFragmentMetadata(final @NonNull String metadataName, final @NonNull String
    metadataValue, final boolean persistent)
    throws KinesisVideoException;
```

永続メタデータと非永続メタデータ

非永続メタデータでは、同一の名前を使ったメタデータ項目を複数追加できます。プロデューサー SDK は、次のフラグメントの前に付加されるまで、メタデータキュー内のメタデータ項目を収集します。メタデータキューはストリームにメタデータ項目が適用されるとクリアされます。メタデータを繰り返すには、putKinesisVideoFragmentMetadata または putFragmentMetadata を再度呼び出します。

永続メタデータの場合、プロデューサー SDK は非永続メタデータの場合と同じ方法でメタデータキュー内のメタデータ項目を収集します。ただし、メタデータ項目は、次のフラグメントの先頭に追加されてもキューから削除されません。

putKinesisVideoFragmentMetadata または putFragmentMetadata で persistent を true に設定して呼び出すと、以下のような動作になります。

- API を呼び出すと、キューにメタデータ項目が追加されます。メタデータは、メタデータ項目がキュー内にある間に、フラグメントごとに MKV タグとして追加されます。
- 同一の名前で、以前に追加されたメタデータ項目と異なる値で API を呼び出すと、その項目は上書きされます。
- 空の値で API を呼び出すと、メタデータキューからそのメタデータ項目は削除 (キャンセル) されます。

Kinesis Video Streams で IPv6 を使用する

コントロールプレーンオペレーションとデータプレーンオペレーションの両方で IPv6 を使用するよう Kinesis Video Streams を設定できます。これにより、アプリケーションはデュアルスタックエンドポイントを介して IPv6 アドレスを使用して Kinesis Video Streams サービスと通信できます。

Note

IPv6 サポートには、特定の SDK バージョンと設定が必要です。Kinesis Video Streams SDK および AWS SDK バージョンが IPv6 デュアルスタックエンドポイントをサポートしていることを確認します。デュアルスタックエンドポイントは IPv4 トラフィックと IPv6 トラフィックの両方をサポートし、一部のリージョンの一部のサービスで使用できます。

Kinesis Video Streams は、プロデューサーアプリケーションとコンシューマーアプリケーションの両方でデュアルスタックエンドポイントを介して IPv6 をサポートします。コントロールプレーン API コールとデータプレーンストリーミングオペレーションに IPv6 を使用するようにアプリケーションを設定できます。

AWS SDK for IPv6 を設定する

AWS SDK を使用して本番稼働用セットアップで Kinesis Video Streams コントロールプレーン APIs を呼び出す場合は、デュアルスタックエンドポイントを設定することで IPv6 を有効にできます。AWS SDK には、デュアルスタックエンドポイントを有効にするための標準化されたメソッドがいくつか用意されています。

Important

デュアルスタックエンドポイントを有効にすると、SDK はデュアルスタックエンドポイントを使用してネットワークリクエストを実行しようとします。サービスまたはリージョンにデュアルスタックエンドポイントが存在しない場合、リクエストは失敗します。

環境変数を使用します。

IPv6 デュアルスタックエンドポイントを有効にするには、次の環境変数を設定します。

```
export AWS_USE_DUALSTACK_ENDPOINT=true
```

AWS 設定ファイルを使用する

AWS 設定ファイル (~/.aws/config) に次の設定を追加します。

```
[default]
use_dualstack_endpoint = true
```

JVM システムプロパティを使用する (Java および Kotlin SDKsのみ)

Java および Kotlin アプリケーションの場合は、次の JVM システムプロパティを設定します。

```
-Daws.useDualstackEndpoint=true
```

または、Java コードでプログラマ的に以下を実行します。

```
System.setProperty("aws.useDualstackEndpoint", "true");
```

SDK サポート

次の AWS SDKs、デュアルスタックのエンドポイント設定をサポートしています。

SDK	サポート	設定方法
AWS CLI v2	はい	環境変数、設定ファイル
SDK for C++	はい	環境変数、設定ファイル
SDK for Go V2 (1.x)	はい	環境変数、設定ファイル
SDK for Java 2.x	はい	環境変数、設定ファイル、JVM プロパティ
SDK for Java 1.x	いいえ	サポートされていません
SDK for JavaScript 3.x	はい	環境変数、設定ファイル
SDK for Python (Boto3)	はい	環境変数、設定ファイル

デュアルスタックエンドポイントを設定すると、AWS SDK は Kinesis Video Streams コントロールプレーン APIs 呼び出すときに IPv6 エンドポイントを自動的に使用します。

IPv6 用の Kinesis Video Streams プロデューサー SDK を設定する

Kinesis Video Streams プロデューサー SDK は、コントロールプレーンオペレーションとデータプレーンオペレーションの両方に IPv6 設定オプションを提供します。これらの設定は AWS SDK デュアルスタックエンドポイント設定で機能します。

C/C++ プロデューサー SDK を設定する

デフォルトのエンドポイントと DNS 解決チェーンは、KVS Producer-C SDK バージョン 1.6.0 によって実装されます。これらのパラメータの設定を設定できる各場所を順番にチェックし、最初に設定したパラメータを選択します。事前定義されたシーケンスは次のとおりです。

プロデューサー SDKs の詳細については、[プロデューサー SDK for C](#)、[プロデューサー SDK for C++](#)、[プロデューサー SDK の関連トピック](#)を参照してください。

エンドポイント設定

1. の `controlPlaneUrl` パラメータ `createAbstractDefaultCallbacksProvider`。
2. エンドポイント設定 CMake パラメータ: (`-DAWS_KVS_USE_LEGACY_ENDPOINT_ONLY=TRUE`、`-DAWS_KVS_USE_DUAL_STACK_ENDPOINT_ONLY=TRUE`)
3. 環境変数: (`export AWS_USE_DUALSTACK_ENDPOINT=TRUE`)
 - `AWS_USE_DUALSTACK_ENDPOINT` が `TRUE` (大文字と小文字を区別しない) の場合、デュアルスタックのエンドポイントが使用されます。
4. そうしないと、レガシーエンドポイントが構築されて使用されます。

2、3、4 では、エンドポイントは に提供されたリージョンに基づいて構築されま
す `createAbstractDefaultCallbacksProvider`。

DNS フィルタリング

KVS プロデューサー SDK は、設定に応じて適切な `CURLLOPT_IPRESOLVE` パラメータを設定します。

1. DNS 解決 CMake パラメータ: (`-DAWS_KVS_IPV4_ONLY=TRUE`、`-DAWS_KVS_IPV6_ONLY=TRUE`、`-DAWS_KVS_IPV4_AND_IPV6_ONLY=TRUE`)

2. 環境変数 (`export AWS_KVS_USE_IPV4=TRUE`、`export AWS_KVS_USE_IPV6=TRUE`)
3. それ以外の場合、フィルタリングは行われません。DNS によって返される IPv4 と IPv6 の両方の IP アドレスを使用できます。

Note

DNS フィルター設定が IPv6 IP アドレスをフィルタリングするように設定されていても、SDK 設定がレガシーエンドポイント (IPv4-only アドレスを返す) を使用する場合は、リクエストは失敗します。

C++ プロデューサー SDK バージョン 3.5.0 では、KVS API コールに Producer-C SDK 1.6.0 を使用します。

GStreamer プラグインを設定する

GStreamer プラグインは基盤となる C プロデューサー SDK を使用するため、前述のように C SDK for IPv6 を設定すると IPv6 設定が自動的に処理されます。

コードの変更は必須ではありません。CMake パラメータを使用して SDK を構築するか、前のセクションで説明したように適切な環境変数を設定します。

データプレーンエンドポイントの解決

データプレーンオペレーションでは、`GetDataEndpoint` API を使用して適切なデュアルスタックのデータプレーンエンドポイントを取得します。サービスは、リクエスト URL に応じて対応するエンドポイントを返します。

例:

- API `GetDataEndpoint` リクエストが で終わるレガシーエンドポイントに対して行われた場合 `.amazonaws.com`、Kinesis Video Streams は で終わるレガシーデータプレーンエンドポイントを返します `.amazonaws.com`。
- API `GetDataEndpoint` リクエストが で終わるデュアルスタックエンドポイントに対して行われた場合 `.api.aws`、Kinesis Video Streams は で終わるデュアルスタックデータプレーンエンドポイントを返します `.api.aws`。

IPv6 AWS CLI の を設定する

Kinesis Video Streams オペレーション (通常はproof-of-concept作業用) AWS CLI に を使用している場合は、デュアルスタックエンドポイントを設定することで IPv6 を有効にできます。

環境変数の使用

```
export AWS_USE_DUALSTACK_ENDPOINT=true
```

AWS 設定ファイルを使用する

AWS CLI 設定ファイル (~/.aws/config) に以下を追加します。

```
[default]
use_dualstack_endpoint = true
```

設定例

C SDK の例

KVS Producer-C SDK を IPV6-onlyモードで構築し、環境変数設定を無視するには、次のコマンドを使用して SDK を構築します。

```
cmake .. -DAWS_KVS_USE_DUAL_STACK_ENDPOINT_ONLY=TRUE -DAWS_KVS_IPV6_ONLY=TRUE
make -j
```

Note

SDK が既に構築されている場合は、クリーンビルドを実行する必要があります。ビルドコマンドを実行する前に、既存のビルドフォルダ、オープンソースフォルダ、依存関係フォルダを削除します。

考慮事項

ネットワークの要件

- ネットワークインフラストラクチャが IPv6 接続をサポートしていることを確認します。

- セキュリティグループとネットワーク ACLs IPv6 トラフィックを許可していることを確認します。
- デプロイ環境から AWS IPv6 エンドポイントへの接続をテストする
- デュアルスタックエンドポイントは、一部の リージョンの一部のサービスで利用可能 — ターゲットリージョンの可用性を検証します。

SDK の互換性

- サポートされている AWS SDK バージョンを使用していることを確認します。
- AWS SDK for Java 1.x はデュアルスタックのエンドポイント設定をサポートしていません
- SDK for Go 1.x (V1) では、共有設定ファイル設定を使用するには、設定ファイルからのロードを有効にする必要があります

テストと検証

IPv6-enabled Kinesis Video Streams アプリケーションを本番環境にデプロイする前に:

- コントロールプレーンオペレーションのテスト (ストリームの作成、削除、一覧表示)
- データプレーンオペレーションの検証 (ビデオの取り込みと消費)
- ネットワーク環境でのパフォーマンスと接続を検証する
- Canary テストを実行して、一貫した IPv6 機能を確認する
- デュアルスタックエンドポイントが使用できない場合のフェイルオーバー動作のテスト

IPv6 を含むアップグレードの影響を受けるお客様

Kinesis Video Streams で IPv6 を有効にすると、継続的な機能を確認するために既存の設定とポリシーを更新する必要がある領域がいくつかあります。

IAM ポリシーと IP アドレスフィルタリング

IAM ユーザーポリシー、ロールポリシー、またはリソースベースのポリシーでソース IP アドレスフィルタリングを使用する場合は、これらのポリシーを更新して IPv6 アドレス範囲を含める必要があります。

⚠ Important

IpAddress または NotIpAddress 条件で IPv4 CIDR ブロックを使用する既存の IAM ポリシーは、IPv6 アドレスでは自動的に機能しません。アクセスコントロールを維持するには、IPv6 範囲を明示的に追加する必要があります。

IPv6 の IAM ポリシー更新の例:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "kinesisvideo:*",
      "Resource": "*",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": [
            "192.0.2.0/24",
            "203.0.113.0/24",
            "2001:db8::/32"
          ]
        }
      }
    }
  ]
}
```

ログ記録とモニタリング

IPv6 アドレスの形式は IPv4 アドレスとは異なるため、ログ記録、モニタリング、分析システムに影響を与える可能性があります。

ログ

ログには、リクエストが IPv6 経由で行われると、sourceIpAddress フィールドに IPv6 アドレスが含まれます。IPv6 アドレス形式を処理するようにログ解析ツールとスクリプトを更新します。

ログの IPv6 アドレスの例:

```
{
  "sourceIPAddress": "2001:db8::1",
  "eventName": "CreateStream",
  "eventSource": "kinesisvideo.amazonaws.com"
}
```

トラブルシューティング

一般的な問題

- 接続の失敗 — IPv6 ネットワーク接続と DNS 解決を検証する
- SDK エラー — デュアルスタックエンドポイントをサポートする互換性のある SDK バージョンを使用していることを確認します。
- 認証の問題 — IAM ポリシーと認証情報が IPv6 エンドポイントで動作することを確認する
- エンドポイントを使用できない — サービスまたはリージョンにデュアルスタックエンドポイントが存在しない場合、リクエストは失敗します

検証手順

- `AWS_USE_DUALSTACK_ENDPOINT=true` が設定されているか `use_dualstack_endpoint = true`、設定ファイルにあることを確認します。
- Kinesis Video Streams SDK IPv6 設定フラグが正しく設定されていることを確認します。
- AWS IPv6 エンドポイントへのネットワーク接続をテストする
- IPv6-specific エラーメッセージのアプリケーションログを確認する
- リージョンが Kinesis Video Streams のデュアルスタックエンドポイントをサポートしていることを確認します。

設定の検証

デュアルスタックのエンドポイント設定を確認するには、以下を確認します。

- 環境変数: `echo $AWS_USE_DUALSTACK_ENDPOINT`
- AWS 設定ファイル: `cat ~/.aws/config | grep use_dualstack_endpoint`
- JVM プロパティ (Java): アプリケーションログのシステムプロパティを確認する

Kinesis Video Streams の再生

次の方法を使用して、Kinesis のビデオストリームを表示できます。

- GetMedia – GetMedia API を使用して独自のアプリケーションを構築し、Kinesis Video Streams を処理できます。GetMediaは低レイテンシーのリアルタイム API です。を使用するプレイヤーを作成するにはGetMedia、自分で構築する必要があります。GetMedia を使用して Kinesis のビデオストリームを表示するアプリケーションを開発する方法については、「[パーサーライブラリを使用したストリーミング](#)」を参照してください。
- HLS – [HTTP Live Streaming \(HLS\)](#) は、業界標準の HTTP ベースのメディアストリーミング通信プロトコルです。HLS を使用して、ライブ再生またはアーカイブされたビデオを表示するために、Kinesis ビデオストリームを表示できます。

HLS はライブ再生に使用できます。レイテンシーは通常 3~5 秒ですが、ユースケース、プレイヤー、ネットワーク条件に応じて 1~10 秒です。サードパーティー製のプレイヤー ([Video.js](#) や [Google Shaka Player](#) など) を使用してビデオストリームを表示するには、HLS ストリーミングセッション URL をプログラムまたは手動で指定できます。[Apple Safari](#) または [Microsoft Edge](#) ブラウザの Location バーに HLS ストリーミングセッション URL を入力して、ビデオを再生することもできます。

- MPEG-DASH – MPEG-DASH と呼ばれる [HTTP 経由の動的アダプティブストリーミング \(DASH\)](#) は、従来の HTTP ウェブサーバーから配信されるインターネット経由でメディアコンテンツを高品質にストリーミングできるようにするアダプティブビットレートストリーミングプロトコルです。

MPEG-DASH はライブ再生に使用できます。レイテンシーは通常 3~5 秒ですが、ユースケース、プレイヤー、ネットワーク条件に応じて 1~10 秒です。プログラムまたは手動で MPEG-DASH ストリーミングセッション URL を指定することで、サードパーティープレイヤー ([dash.js](#) や [Google Shaka Player](#) など) を使用してビデオストリームを表示できます。

- GetClip – GetClip API を使用して、アーカイブされたオンデマンドメディアを含むクリップ (MP4 ファイル内) を、指定された時間範囲で指定されたビデオストリームからダウンロードできます。詳細については、[GetClip](#) API リファレンスを参照してください。

トピック

- [動画再生トラックの要件](#)
- [HLS による動画再生](#)

- [MPEG-DASH による動画再生](#)

動画再生トラックの要件

Amazon Kinesis Video Streams は、複数の形式でエンコードされたメディアをサポートしています。Kinesis ビデオストリームで、以下に示す 4 つの APIs のいずれかでサポートされていない形式を使用している場合は、トラックタイプの制限がないため[GetMediaForFragmentList](#)、[GetMedia](#)または を使用します。

トピック

- [GetClip の要件](#)
- [GetDASHStreamingSessionURL の要件](#)
- [GetHLSStreamingSessionURL の要件](#)
- [GetImages の要件](#)

GetClip の要件

この API の詳細については、「[GetClip](#)」を参照してください。

トラック 1 の説明	トラック 1 コーデック ID	トラック 2 の説明	トラック 2 コーデック ID
H.264 ビデオ	V_MPEG4/ISO/AVC	該当なし	該当なし
H.264 ビデオ	V_MPEG4/ISO/AVC	AAC オーディオ	A_AAC
H.264 ビデオ	V_MPEG4/ISO/AVC	G.711 オーディオ (A-Law のみ)	A_MS/ACM
H.265 ビデオ	V_MPEGH/ISO/HEVC	該当なし	該当なし
H.265 ビデオ	V_MPEGH/ISO/HEVC	AAC オーディオ	A_AAC

⚠ Important

各フラグメントに含まれるコーデックプライベートデータ (CPD) には、フラグメントを適切にデコードするために必要なフレームレート、解像度、エンコーディングプロファイルなどのコーデック固有の初期化情報が含まれています。CPD の変更は、結果のクリップのターゲットフラグメント間ではサポートされていません。CPD は、クエリされたメディアを通じて一貫性を保つ必要があります。そうしないと、エラーが返されます。

⚠ Important

トラックの変更はサポートされていません。トラックは、クエリされたメディア全体で一貫性を保つ必要があります。ストリームのフラグメントがビデオのみからオーディオとビデオの両方に変わった場合、または AAC オーディオトラックが A-Law オーディオトラックに変更された場合、エラーが返されます。

GetDASHStreamingSessionURL の要件

この API の詳細については、「[GetDASHStreamingSessionURL](#)」を参照してください。

トラック 1 の説明	トラック 1 コーデック ID	トラック 2 の説明	トラック 2 コーデック ID
H.264 ビデオ	V_MPEG4/ISO/AVC	該当なし	該当なし
H.264 ビデオ	V_MPEG4/ISO/AVC	AAC オーディオ	A_AAC
H.264 ビデオ	V_MPEG4/ISO/AVC	G.711 オーディオ (A-Law のみ)	A_MS/ACM
H.264 ビデオ	V_MPEG4/ISO/AVC	G.711 オーディオ (U-Law のみ)	A_MS/ACM
AAC オーディオ	A_AAC	該当なし	該当なし
H.265 ビデオ	V_MPEGH/ISO/HEVC	該当なし	該当なし
H.265 ビデオ	V_MPEGH/ISO/HEVC	AAC オーディオ	A_AAC

⚠ Important

各フラグメントに含まれるコーデックプライベートデータ (CPD) には、フラグメントを適切にデコードするために必要なフレームレート、解像度、エンコーディングプロファイルなどのコーデック固有の初期化情報が含まれています。CPD の変更は、ストリーミングセッション中はサポートされていません。CPD は、クエリされたメディアを通じて一貫性を維持する必要があります。

⚠ Important

トラックの変更はサポートされていません。トラックは、クエリされたメディア全体で一貫性を保つ必要があります。ストリーム内のフラグメントがビデオのみからオーディオとビデオの両方に変わるか、AAC オーディオトラックが A-Law オーディオトラックに変更されると、ストリーミングは失敗します。

GetHLSStreamingSessionURL の要件

この API の詳細については、「[GetHLSStreamingSessionURL](#)」を参照してください。

HLS Mp4

トラック 1 の説明	トラック 1 コーデック ID	トラック 2 の説明	トラック 2 コーデック ID
H.264 ビデオ	V_MPEG4/ISO/AVC	該当なし	該当なし
H.264 ビデオ	V_MPEG4/ISO/AVC	AAC オーディオ	A_AAC
AAC オーディオ	A_AAC	該当なし	該当なし
H.265 ビデオ	V_MPEGH/ISO/HEVC	該当なし	該当なし
H.265 ビデオ	V_MPEGH/ISO/HEVC	AAC オーディオ	A_AAC

HLS TS

トラック 1 の説明	トラック 1 コーデック ID	トラック 2 の説明	トラック 2 コーデック ID
H.264 ビデオ	V_MPEG4/ISO/AVC	該当なし	該当なし
H.264 ビデオ	V_MPEG4/ISO/AVC	AAC オーディオ	A_AAC
AAC オーディオ	A_AAC	該当なし	該当なし

 Note

各フラグメントに含まれるコーデックプライベートデータ (CPD) には、フラグメントを適切にデコードするために必要なフレームレート、解像度、エンコーディングプロファイルなどのコーデック固有の初期化情報が含まれています。TS と MP4 の両方で、CPD の変更はストリーミングセッション中にサポートされます。したがって、セッション内のフラグメントは、再生を中断することなく CPD で異なる情報を持つことができます。ストリーミングセッションごとに許可される CPD の変更は 500 個のみです。

 Important

トラックの変更はサポートされていません。トラックは、クエリされたメディア全体で一貫性を保つ必要があります。ストリーム内のフラグメントがビデオのみからオーディオとビデオの両方に変わるか、AAC オーディオトラックが A-Law オーディオトラックに変更されると、ストリーミングは失敗します。

GetImages の要件

この API の詳細については、「[GetImages](#)」を参照してください。

 Note

GetImages メディアには、トラック 1 にビデオトラックが含まれている必要があります。

HLS による動画再生

[HTTP Live Streaming \(HLS\)](#) は、業界標準の HTTP ベースのメディアストリーミング通信プロトコルです。HLS を使用して、ライブ再生またはアーカイブされたビデオを表示するために、Kinesis ビデオストリームを表示できます。

HLS はライブ再生に使用できます。レイテンシーは通常 3~5 秒ですが、ユースケース、プレイヤー、ネットワーク条件に応じて 1~10 秒です。サードパーティー製のプレイヤー ([Video.js](#) や [Google Shaka Player](#) など) を使用してビデオストリームを表示するには、HLS ストリーミングセッション URL をプログラムまたは手動で指定できます。また、[Apple Safari](#) または [Microsoft Edge](#) ブラウザの Location バーに HLS ストリーミングセッション URL を入力して、ビデオを再生することもできます。

HLS を使用して Kinesis ビデオストリームを表示するには、まず [GetHLSStreamingSessionURL](#) を使用してストリーミングセッションを作成します。このアクションにより、HLS セッションにアクセスするための URL (セッショントークンを含む) が返されます。次に、この URL をメディアプレイヤーまたはスタンドアロンアプリケーションで使用してストリームを表示できます。

Important

Kinesis Video Streams に送信されたすべてのメディアを HLS 経由で再生できるわけではありません。特定のアップロード要件 [the section called “GetHLSStreamingSessionURL”](#) については、「」を参照してください。

トピック

- [を使用して HLS ストリーミングセッション URL AWS CLI を取得する](#)
- [例: HTML および JavaScript で HLS を使用する](#)
- [HLS の問題のトラブルシューティング](#)

を使用して HLS ストリーミングセッション URL AWS CLI を取得する

次の手順では、を使用して Kinesis ビデオストリームの HLS ストリーミングセッション URL AWS CLI を生成する方法を示します。

インストール手順については、[AWS Command Line Interface 「ユーザーガイド」](#) を参照してください。インストール後、認証情報とリージョンを使用して [を設定します AWS CLI](#)。

または、AWS CLI がインストールされ、設定されている AWS CloudShell ターミナルを開きます。詳細については、『[AWS CloudShell ユーザーガイド](#)』を参照してください。

Kinesis ビデオストリームの HLS URL エンドポイントを取得します。

1. ターミナルに次のように入力します。

```
aws kinesisisvideo get-data-endpoint \  
  --api-name GET_HLS_STREAMING_SESSION_URL \  
  --stream-name YourStreamName
```

次のようなレスポンスが表示されます。

```
{  
  "DataEndpoint": "https://b-1234abcd.kinesisvideo.aws-region.amazonaws.com"  
}
```

2. 返されたエンドポイントに HLS ストリーミングセッション URL リクエストを行います。

Live

ライブ再生の場合、HLS メディアプレイリストは、最新のメディアが利用可能になると継続的に更新されます。このタイプのセッションをメディアプレーヤーで再生する場合、ユーザーインターフェイスには通常「ライブ」通知が表示され、表示する再生ウィンドウの位置を選択するためのスクラバーコントロールはありません。

このコマンドを実行するときは、このストリームにメディアをアップロードしていることを確認してください。

```
aws kinesis-video-archived-media get-hls-streaming-session-url \  
  --endpoint-url https://b-1234abcd.kinesisvideo.aws-region.amazonaws.com \  
  --stream-name YourStreamName \  
  --playback-mode LIVE
```

Live replay

ライブ再生の場合、再生は指定された開始時刻から開始されます。HLS メディアプレイリストは、最新のメディアが利用可能になると継続的に更新されます。セッションには、セッションの有効期限が切れるまで、または指定された終了時刻のいずれか早い方まで、新しく取り込まれたメディアが引き続き含まれます。このモードは、イベントの検出で再生を開始

し、セッションの作成時点でまだ取り込まれていないライブストリーミングメディアを継続できるようにする場合に便利です。

開始タイムスタンプを決定します。

この例では、Unix エポック時間を秒形式で使します。[タイムスタンプのフォーマットの詳細については](#)、AWS Command Line Interface 「ユーザーガイド」の「タイムスタンプ」セクションを参照してください。

変換ツールについては、「[UnixTime.org](#)」を参照してください。

- 1708471800 は 2024 年 2 月 20 日午後 3:30:00 GMT-08:00 に相当します

この例では、終了タイムスタンプを指定しません。つまり、セッションの有効期限が切れるまで、セッションには新しく取り込まれたメディアが引き続き含まれます。

LIVE_REPLAY 再生モードと [HLS フラグメントセクタ](#)を指定して GetHLSStreamingSessionURL API を呼び出します。

```
aws kinesis-video-archived-media get-hls-streaming-session-url \
  --endpoint-url https://b-1234abcd.kinesisvideo.aws-region.amazonaws.com \
  --stream-name YourStreamName \
  --playback-mode LIVE_REPLAY \
  --hls-fragment-selector \
  "FragmentSelectorType=SERVER_TIMESTAMP, TimestampRange={StartTimestamp=1708471800}"
```

On-demand

オンデマンド再生の場合、HLS メディアプレイリストには、HLS フラグメントセクタで指定されたメディアが含まれます。このタイプのセッションがメディアプレーヤーで再生される場合、ユーザーインターフェイスには、通常再生ウィンドウ内の位置を選択するためのスクラバーコントロールが表示されます。

ストリームの特定のセクションの URL を作成するには、まず開始タイムスタンプと終了タイムスタンプを決定します。

この例では、Unix エポック時間を秒形式で使します。[タイムスタンプの形式の詳細については](#)、AWS Command Line Interface 「ユーザーガイド」の「タイムスタンプ」セクションを参照してください。

変換ツールについては、「[UnixTime.org](https://unixtime.org)」を参照してください。

- 1708471800 は 2024 年 2 月 20 日午後 3:30:00 GMT-08:00 に相当します
- 1708471860 は 2024 年 2 月 20 日午後 3:31:00 GMT-08:00 に相当します

ON_DEMAND 再生モードと [HLS フラグメントセレクタ](#)を指定して
GetHLSStreamingSessionURL API を呼び出します。

```
aws kinesis-video-archived-media get-hls-streaming-session-url \
  --endpoint-url https://b-1234abcd.kinesisvideo.aws-region.amazonaws.com \
  --stream-name YourStreamName \
  --playback-mode ON_DEMAND \
  --hls-fragment-selector \

"FragmentSelectorType=SERVER_TIMESTAMP, TimestampRange={StartTimestamp=1708471800, EndTimestamp=1708471860}"
```

 Note

タイムスタンプは、[the section called “HLSTimestampRange”](#) ドキュメントに記載されているように、相互に 24 時間以内である必要があります。

次のようなレスポンスが表示されます。

```
{
  "HLSStreamingSessionURL": "https://b-1234abcd.kinesisvideo.aws-region.amazonaws.com/hls/v1/getHLSMasterPlaylist.m3u8?SessionToken=CiAz...DkRE6M~"
}
```

 Important

許可されていないエンティティがアクセスできる場所に、このトークンを共有したり保存したりしないでください。トークンがストリームのコンテンツへのアクセスを提供します。AWS 認証情報で使用するのと同じ方法でトークンを保護します。

この URL と任意の HLS プレイヤーを使用して、HLS ストリームを表示できます。

たとえば、VLC メディアプレーヤーを使用します。

また、Apple Safari または Microsoft Edge ブラウザの Location バーに HLS ストリーミングセッション URL を入力して、HLS ストリームを再生することもできます。

例: HTML および JavaScript で HLS を使用する

次の例は、AWS SDK for JavaScript v2 を使用して Kinesis ビデオストリームの HLS ストリーミングセッションを取得し、ウェブページで再生する方法を示しています。この例では、動画の再生に以下のプレーヤーを使用します。

- [Video.js](#)
- [Google Shaka Player](#)
- [hls.js](#)

GitHub で[完全なサンプルコード](#)と[ホストされたウェブページ](#)を表示します。この静的ウェブページは、Amazon Kinesis ビデオストリームからの HLS および MPEG-DASH 出力のテストと実験を簡素化します。サンプルページには、次のパラメータの入力フィールドが表示されます。

- AWS リージョン: Kinesis ビデオストリームがあるリージョン
- ストリーム名: Kinesis ビデオストリームの名前
- 再生モード: HLS 再生モード (LIVE、LIVE_REPLAY、または ON_DEMAND)
- フラグメントセレクトタイプ: フラグメントの選択に使用されるメソッド (SERVER_TIMESTAMP または PRODUCER_TIMESTAMP)
- フラグメント番号: 開始フラグメント番号 (該当する場合)
- コンテナ形式: メディアコンテナの形式 (FRAGMENTED_MP4 または MPEG_TS)

アプリケーションは、HTML ページの入力ボックスからこれらの値を取得し、それを使用して、ページに表示される HLS ストリーミングセッションのリクエストを構築します。

コードウォークスルーピックアップ:

- [ブラウザ用の AWS SDK for JavaScript をインポートする](#)
- [Kinesis Video Streams クライアントのセットアップ](#)
- [HLS 再生用のエンドポイントを取得する](#)

- [Kinesis Video Streams アーカイブメディアクライアントを設定する](#)
- [HLS ストリーミングセッション URL を取得する](#)
- [ウェブページに HLS ストリームを表示する](#)

ブラウザ用の AWS SDK for JavaScript をインポートする

ウェブページに次のスクリプトタグを含めて、AWS SDK for JavaScript v2 をプロジェクトにインポートします。

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/aws-sdk/2.490.0/aws-sdk.min.js"></script>
```

詳細については、[AWS SDK for JavaScript](#) のドキュメントを参照してください。

Kinesis Video Streams クライアントのセットアップ

HLS でストリーミングビデオにアクセスするには、まず Kinesis Video Streams クライアントを作成して設定します。その他の認証方法については、[「ウェブブラウザでの認証情報の設定」](#)を参照してください。

```
const clientConfig = {
  accessKeyId: 'YourAccessKey',
  secretAccessKey: 'YourSecretKey',
  region: 'us-west-2'
};
const kinesisVideoClient = new AWS.KinesisVideo(clientConfig);
```

アプリケーションは、HTML ページの入力ボックスから必要な値を取得します。

HLS 再生用のエンドポイントを取得する

Kinesis Video Streams クライアントを使用して [the section called “GetDataEndpoint”](#) API を呼び出し、エンドポイントを取得します。

```
const getDataEndpointOptions = {
  StreamName: 'YourStreamName',
  APIName: 'GET_HLS_STREAMING_SESSION_URL'
};
const getDataEndpointResponse = await kinesisVideoClient
  .getDataEndpoint(getDataEndpointOptions)
```

```
.promise();  
const hlsDataEndpoint = getDataEndpointResponse.DataEndpoint;
```

このコードは、エンドポイントを `hlsDataEndpoint` 変数に保存します。

Kinesis Video Streams アーカイブメディアクライアントを設定する

Kinesis Video Streams アーカイブメディアクライアントのクライアント設定で、前のステップで取得したエンドポイントを指定します。

```
const archivedMediaClientConfig = {  
  accessKeyId: 'YourAccessKey',  
  secretAccessKey: 'YourSecretKey',  
  region: 'us-west-2',  
  endpoint: hlsDataEndpoint  
};  
const kinesisVideoArchivedMediaClient = new  
  AWS.KinesisVideoArchivedMedia(archivedMediaClientConfig);
```

HLS ストリーミングセッション URL を取得する

Kinesis Video Streams アーカイブメディアクライアントを使用して [the section called “GetHLSStreamingSessionURL”](#) API を呼び出し、HLS 再生 URL を取得します。

```
const getHLSStreamingSessionURLOptions = {  
  StreamName: 'YourStreamName',  
  PlaybackMode: 'LIVE'  
};  
const getHLSStreamingSessionURLResponse = await kinesisVideoArchivedMediaClient  
  .getHLSStreamingSessionURL(getHLSStreamingSessionURLOptions)  
  .promise();  
const hlsUrl = getHLSStreamingSessionURLResponse.HLSStreamingSessionURL;
```

ウェブページに HLS ストリームを表示する

HLS ストリーミングセッション URL がある場合、それをビデオプレーヤーに指定します。URL をビデオプレーヤーに指定する方法は、使用するプレーヤーごとに異なります。

Video.js

[Video.js](#) とその CSS クラスをブラウザスクリプトにインポートするには、次の手順を実行します。


```
<link rel="stylesheet" href="https://vjs.zencdn.net/6.6.3/video-js.css">
<script src="https://vjs.zencdn.net/6.6.3/video.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/videojs-contrib-hls/5.14.1/
videojs-contrib-hls.js"></script>
```

動画を表示する video HTML 要素を作成します。

```
<video id="videojs" class="player video-js vjs-default-skin" controls autoplay></
video>
```

HLS URL を HTML 動画要素ソースとして設定します。

```
const playerElement = document.getElementById('videojs');
const player = videojs(playerElement);
player.src({
  src: hlsUrl,
  type: 'application/x-mpegURL'
});
player.play();
```

Shaka

[Google Shaka プレイヤー](#)をブラウザスクリプトにインポートするには、次の手順を実行します。

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/shaka-player/2.4.1/shaka-
player.compiled.js"></script>
```

動画を表示する video HTML 要素を作成します。

```
<video id="shaka" class="player" controls autoplay></video>
```

動画要素を指定して Shaka プレイヤーを作成し、ロードメソッドを呼び出します。

```
const playerElement = document.getElementById('shaka');
const player = new shaka.Player(playerElement);
player.load(hlsUrl);
```

hls.js

ブラウザスクリプトに [hls.js](#) をインポートするには、次の手順を実行します。

```
<script src="https://cdn.jsdelivr.net/npm/hls.js@latest"></script>
```

動画を表示する video HTML 要素を作成します。

```
<video id="hlsjs" class="player" controls autoplay></video>
```

hls.js プレイヤーを作成し、HLS URL を渡して、再生するように指示します。

```
const playerElement = document.getElementById('hlsjs');
const player = new Hls();
player.loadSource(hlsUrl);
player.attachMedia(playerElement);
player.on(Hls.Events.MANIFEST_PARSED, function() {
    video.play();
});
```

HLS の問題のトラブルシューティング

このセクションでは、Kinesis Video Streams で (HLS) HTTP Live Streaming を使用するときが発生する可能性がある問題について説明します。

問題

- [HLS ストリーミングセッション URL の取得は成功するが、ビデオプレーヤーで再生が失敗する](#)
- [プロデューサーとプレーヤー間のレイテンシーが高すぎる](#)

HLS ストリーミングセッション URL の取得は成功するが、ビデオプレーヤーで再生が失敗する

この状況が発生するのは、HLS ストリーミングセッション URL は GetHLSStreamingSessionURL を使用して正常に取得できるが、この URL をビデオプレーヤーに指定したときに動画が再生されない場合です。

この状況のトラブルシューティングを行うには、以下を試します。

- Kinesis Video Streams コンソールでビデオストリームが再生されるかどうかを確認します。コンソールに表示されたエラーを検討します。

- フラグメント継続時間が 1 秒未満の場合は、1 秒に増やします。フラグメントの再生時間が短すぎると、ビデオフラグメントのリクエストが多すぎるため、サービスはプレイヤーをスロットリングする可能性があります。
- 各 HLS ストリーミングセッション URL を 1 つのプレイヤーでのみ使用していることを確認します。1 つの HLS ストリーミングセッション URL を複数のプレイヤーで使用している場合、サービスが受け取るリクエストが多すぎて、プレイヤーがスロットリングされることがあります。
- プレイヤーが HLS ストリーミングセッションに指定するすべてのオプションをサポートしていることを確認します。以下のパラメータでさまざまな値の組み合わせを試します。
 - ContainerFormat
 - PlaybackMode
 - FragmentSelectorType
 - DiscontinuityMode
 - MaxMediaPlaylistFragmentResults

通常、一部のメディアプレイヤー (HTML5 やモバイルプレイヤーなど) は、fMP4 コンテナ形式の HLS のみをサポートします。他のメディアプレイヤー (フラッシュやカスタムプレイヤーなど) は、MPEG TS コンテナ形式の HLS のみをサポートする場合があります。トラブルシューティングを開始するには、ContainerFormat パラメータを試すことをお勧めします。

- 各フラグメントに一貫した数のトラックがあることを確認します。ストリーム内のフラグメントが、オーディオトラックとビデオトラックの両方を持つことと、ビデオトラックのみを持つことの間で変化していないことを確認します。また、各トラックのフラグメント間でエンコーダー設定 (解像度とフレームレート) が変更されていないことを確認します。

プロデューサーとプレイヤー間のレイテンシーが高すぎる

この状況が発生するのは、動画をキャプチャした時点から動画プレイヤーで再生した時点までのレイテンシーが高すぎる場合です。

動画は HLS を介してフラグメント単位で再生されます。そのため、レイテンシーをフラグメント継続時間未満にすることはできません。レイテンシーには、データのバッファリングと転送の所要時間も含まれます。ソリューションで 1 秒未満のレイテンシーが必要な場合は、代わりに GetMedia API を使用することを検討してください。

以下のパラメータを調整してレイテンシー全体を短縮できますが、それに伴って画質が低下したり、再バッファリング率が高くなったりする場合があります。

- フラグメント期間 – フラグメント期間は、ビデオエンコーダーによって生成されたキーフレームの頻度によって制御されるストリーム内の分割間のビデオの量です。推奨される値は 1 秒です。フラグメント継続時間が短いほど、動画データをサービスに転送する前にフラグメントが完了するまで待機する時間が少なくなります。また、フラグメントが短いほど、サービスでの処理が高速になります。ただし、フラグメント継続時間が短すぎると、プレーヤーでコンテンツが不足するため、停止してコンテンツをバッファリングしなければならない確率が高くなります。フラグメント継続時間が 500 ミリ秒未満の場合、プロデューサーで作成されるリクエストが多すぎて、サービスでスロットリングされることがあります。
- ビットレート – ビットレートが低いビデオストリームでは、読み取り、書き込み、送信にかかる時間が短くなります。ただし、通常、ビデオストリームのビットレートが低いほど、画質は低下します。
- メディアプレイリストのフラグメント数 – レイテンシーの影響を受けやすいプレーヤーは、メディアプレイリスト内の最新のフラグメントのみをロードする必要があります。ほとんどのプレーヤーは、代わりに最も早いフラグメントから開始します。プレイリスト内のフラグメントの数を減らすことで、以前のフラグメントと新しいフラグメントの時間分離を短縮できます。プレイリストのサイズが小さいほど、プレイリストへの新しいフラグメントの追加に遅延がある場合、またはプレーヤーが更新されたプレイリストを取得するのに遅延がある場合、再生中にフラグメントがスキップされる可能性があります。プレイリストから最新のフラグメントのみをロードするように設定されたプレーヤーを使用するには、3~5 個のフラグメントを使用することをお勧めします。
- プレーヤーバッファサイズ – ほとんどのビデオプレーヤーには設定可能な最小バッファ期間があり、通常は 10 秒のデフォルトです。最も低いレイテンシーの場合、この値は 0 秒に設定できます。ただし、そうすることで、遅延を吸収するバッファがプレーヤーにないため、フラグメントの生成に遅延が発生した場合にプレーヤーはリバッファリングします。
- プレーヤー「キャッチアップ」 – 遅延したフラグメントによってフラグメントのバックログが再生されるなど、バッファがいっぱいになると、ビデオプレーヤーは通常ビデオバッファの前面まで再生を自動的にキャッチしません。カスタムプレーヤーでは、これを避けるためにフレームをドロップするか、再生スピードを速くして (1.1 倍など)、バッファの先頭までキャッチアップできます。プレーヤーのキャッチアップに伴って、再生が途切れたり、速度が増したりします。また、バッファサイズが短いと、再バッファリングの回数が増える場合があります。

MPEG-DASH による動画再生

MPEG-DASH を使用して Kinesis のビデオストリームを表示するには、まず [GetDASHStreamingSessionURL](#) を使用してストリーミングセッションを作成します。このアクションにより、MPEG-DASH セッションにアクセスするための URL (セッショントークンを含む) が返さ

れます。次に、この URL をメディアプレーヤーまたはスタンドアロンアプリケーションで使用してストリームを表示できます。

Amazon Kinesis のビデオストリームでは、MPEG-DASH を介して動画を提供する場合、以下が要件となります。

- ストリーミングビデオ再生トラックの要件については、「」を参照してください[the section called “GetDASHStreamingSessionURL”](#)。
- データの保持期間が 0 より大きい。
- 各フラグメントの動画トラックに、AVC (Advanced Video Coding) のコーデックプライベートデータが H.264 形式で、および HEVC のコーデックプライベートデータが H.265 形式で含まれている必要があります。詳細については、「[MPEG-4 仕様 ISO/IEC 14496-15](#)」を参照してください。ストリームデータを特定の形式に適応させる方法については、「[NAL 適応フラグ](#)」を参照してください。
- 各フラグメントのオーディオトラック (存在する場合) に、コーデックプライベートデータが AAC 形式 ([AAC 仕様 ISO/IEC 13818-7](#)) または [MS Wave 形式](#) で含まれている必要があります。

例: HTML および JavaScript での MPEG-DASH の使用

次の例では、Kinesis のビデオストリームの MPEG-DASH ストリーミングセッションを取得して、ウェブページで再生する方法を示します。この例では、動画の再生に以下のプレーヤーを使用します。

- [Google Shaka Player](#)
- [dash.js](#)

トピック

- [MPEG-DASH 再生用に Kinesis Video Streams クライアントを設定する](#)
- [MPEG-DASH 再生用の Kinesis Video Streams アーカイブコンテンツエンドポイントを取得する](#)
- [MPEG-DASH ストリーミングセッション URL を取得する](#)
- [MPEG-DASH 再生でストリーミングビデオを表示する](#)
- [完了した例](#)

MPEG-DASH 再生用に Kinesis Video Streams クライアントを設定する

MPEG-DASH でストリーミング動画にアクセスするには、まず、Kinesis Video Streams クライアント (サービスエンドポイントを取得するため) とアーカイブ済みメディアクライアント (MPEG-DASH ストリーミングセッションを取得するため) を作成して設定します。アプリケーションは、HTML ページの入力ボックスから必要な値を取得します。

```
var streamName = $('#streamName').val();

// Step 1: Configure SDK Clients
var options = {
    accessKeyId: $('#accessKeyId').val(),
    secretAccessKey: $('#secretAccessKey').val(),
    sessionToken: $('#sessionToken').val() || undefined,
    region: $('#region').val(),
    endpoint: $('#endpoint').val() || undefined
};
var kinesisVideo = new AWS.KinesisVideo(options);
var kinesisVideoArchivedContent = new AWS.KinesisVideoArchivedMedia(options);
```

MPEG-DASH 再生用の Kinesis Video Streams アーカイブコンテンツエンドポイントを取得する

クライアントの初期化後に、Kinesis Video Streams のアーカイブ済みコンテンツエンドポイントを取得して、次のように MPEG-DASH ストリーミングセッション URL を取得できるようにします。

```
// Step 2: Get a data endpoint for the stream
console.log('Fetching data endpoint');
kinesisVideo.getDataEndpoint({
    StreamName: streamName,
    APIName: "GET_DASH_STREAMING_SESSION_URL"
}, function(err, response) {
    if (err) { return console.error(err); }
    console.log('Data endpoint: ' + response.DataEndpoint);
    kinesisVideoArchivedContent.endpoint = new AWS.Endpoint(response.DataEndpoint);
```

MPEG-DASH ストリーミングセッション URL を取得する

アーカイブ済みコンテンツエンドポイントがある場合は、次のように [GetHLSStreamingSessionURL](#) API を呼び出して MPEG-DASH ストリーミングセッション URL を取得します。

```
// Step 3: Get a Streaming Session URL
var consoleInfo = 'Fetching ' + protocol + ' Streaming Session URL';
console.log(consoleInfo);

if (protocol === 'DASH') {
    kinesisVideoArchivedContent.getDASHStreamingSessionURL({
        StreamName: streamName,
        PlaybackMode: $('#playbackMode').val(),
        DASHFragmentSelector: {
            FragmentSelectorType: $('#fragmentSelectorType').val(),
            TimestampRange: $('#playbackMode').val() === "LIVE" ? undefined : {
                StartTimestamp: new Date($('#startTimestamp').val()),
                EndTimestamp: new Date($('#endTimestamp').val())
            }
        },
        DisplayFragmentTimestamp: $('#displayFragmentTimestamp').val(),
        DisplayFragmentNumber: $('#displayFragmentNumber').val(),
        MaxManifestFragmentResults: parseInt($('#maxResults').val()),
        Expires: parseInt($('#expires').val())
    }, function(err, response) {
        if (err) { return console.error(err); }
        console.log('DASH Streaming Session URL: ' + response.DASHStreamingSessionURL);
    });
}
```

MPEG-DASH 再生でストリーミングビデオを表示する

MPEG-DASH ストリーミングセッション URL がある場合、それをビデオプレーヤーに指定します。URL をビデオプレーヤーに指定する方法は、使用するプレーヤーごとに異なります。

次のコード例では、ストリーミングセッション URL を [Google Shaka](#) プレーヤーに指定する方法を示します。

```
// Step 4: Give the URL to the video player.

//Shaka Player elements
```

```
<video id="shaka" class="player" controls autoplay></video>
<script src="https://cdnjs.cloudflare.com/ajax/libs/shaka-player/2.4.1/shaka-
player.compiled.js">
</script>
...

var playerName = $('#player').val();

if (playerName === 'Shaka Player') {
  var playerElement = $('#shaka');
  playerElement.show();

  var player = new shaka.Player(playerElement[0]);
  console.log('Created Shaka Player');

  player.load(response.DASHStreamingSessionURL).then(function() {
    console.log('Starting playback');
  });
  console.log('Set player source');
}
```

次のコード例では、ストリーミングセッション URL を [dash.js](#) プレーヤーに指定する方法を示します。

```
<!-- dash.js Player elements -->
<video id="dashjs" class="player" controls autoplay=""></video>
<script src="https://cdn.dashjs.org/latest/dash.all.min.js"></script>
...

var playerElement = $('#dashjs');
playerElement.show();

var player = dashjs.MediaPlayer().create();
console.log('Created DASH.js Player');

player.initialize(document.querySelector('#dashjs'), response.DASHStreamingSessionURL,
  true);
console.log('Starting playback');
console.log('Set player source');
}
```


完了した例

GitHub で [完全な例のコードをダウンロードまたは表示](#) できます。

Kinesis Video Streams で通知を設定する

メディアフラグメントが使用可能になると、Kinesis Video Streams は Amazon Simple Notification Service (Amazon SNS) 通知を使用してお客様に通知します。

Note

Amazon Kinesis Video Streams は、通信に Amazon SNS 標準トピックを使用します。FIFO トピックは現在サポートされていません。

以下のトピックでは、通知の使用を開始する方法について説明します。

トピック

- [通知設定の管理](#)
- [プロデューサー MKV タグについて](#)
- [Amazon SNS メッセージ](#)
- [クロスアカウント Amazon SNS 通知の発行](#)

通知設定の管理

通知設定を管理するには、UpdateNotificationConfigurationと を使用し
ますDescribeNotificationConfiguration。詳細については、以下を参照してください。

UpdateNotificationConfiguration

この API オペレーションを使用して、ストリームの通知情報を更新しま
す。UpdateNotificationConfiguration 機能の詳細については、Amazon Kinesis Video
Streamsデベロッパーガイド」の[UpdateNotificationConfiguration](#)」を参照してください。

Note

通知設定を更新してから通知を開始するには、少なくとも 1 分かかります。更新呼び出
しPutMedia後、 を呼び出す前に少なくとも 1 分待ちます。

DescribeNotificationConfiguration

この API を使用して、ストリームにアタッチされた通知設定を記述します。DescribeNotificationConfiguration 機能の詳細については、「Amazon Kinesis Video Streamsデベロッパーガイド」の「[DescribeNotificationConfiguration](#)」を参照してください。

プロデューサー MKV タグについて

Kinesis Video Streams プロデューサー SDK を使用して、SDK で API オペレーションを公開することで、関心のある特定のフラグメントにタグを付けることができます。[このセクションのコードで](#)、この仕組みのサンプルを参照してください。この API を呼び出すと、SDK はフラグメントデータとともに事前定義された MKV タグのセットを追加します。Kinesis Video Streams は、これらの特別な MKV タグを認識し、タグ付けされたフラグメントの通知を開始します。

Notification MKV タグとともに提供されるフラグメントメタデータは、Amazon SNS トピックペイロードの一部として公開されます。

プロデューサー MKV タグの構文

```
|+ Tags
| + Tag
|   // MANDATORY: Predefined MKV tag to trigger the notification for the fragment
|   + Simple
|     + Name: AWS_KINESISVIDEO_NOTIFICATION
|     + String
|   // OPTIONAL: Key value pairs that will be sent as part of the Notification payload
|   + Simple
|     + Name: CUSTOM_KEY_1 // Max 128 bytes
|     + String: CUSTOM_VALUE_1 // Max 256 bytes
|   + Simple
|     + Name: CUSTOM_KEY_2 // Max 128 bytes
|     + String: CUSTOM_VALUE_2 // Max 256 bytes
```

MKV タグの制限

次の表に、メタデータタグに関連する制限を示します。メタデータタグの制限が調整可能な場合は、アカウントマネージャーを通じて引き上げをリクエストできます。

制限	最大値	引き上げ可能
オプションのメタデータキーの長さ	128	いいえ
オプションのメタデータ値の長さ	256	いいえ
オプションのメタデータの最大数	10	[Yes (はい)]

Amazon SNS メッセージ

このトピックには、Amazon SNS メッセージとトピックペイロードに関する詳細情報が含まれています。

トピック

- [Amazon SNS トピックペイロード](#)
- [Amazon SNS メッセージを表示する](#)

Amazon SNS トピックペイロード

前のワークフローで開始された通知は、次の例に示すように、Amazon SNS トピックペイロードを配信します。この例は、Amazon Simple Queue Service (Amazon SQS) キューから通知データを消費した後に発生する Amazon SNS Amazon SQS メッセージです。

```
{
  "Type" : "Notification",
  "MessageId" : Message ID,
  "TopicArn" : SNS ARN,
  "Subject" : "Kinesis Video Streams Notification",
  "Message" : "{\"StreamArn\": \"Stream Arn\", \"FragmentNumber\": \"Fragment Number\",
    \"FragmentStartProducerTimestamp\": FragmentStartProducerTimestamp,
    \"FragmentStartServerTimestamp\": FragmentStartServerTimestamp,
    \"NotificationType\": \"PERSISTED\", \"NotificationPayload\": {\" CUSTOM_KEY_1:
    \"CUSTOM_VALUE_1\",
    \"CUSTOM_KEY_2: \"CUSTOM_VALUE_2\"}}\",
```

```
"Timestamp" : "2022-04-25T18:36:29.194Z",
"SignatureVersion" : Signature Version,
"Signature" : Signature,
"SigningCertURL" : Signing Cert URL,
"UnsubscribeURL" : Unsubscribe URL
}
```

Subject: "Kinesis Video Streams Notification"

Message:

```
{
  "StreamArn":Stream Arn,
  "FragmentNumber":Fragment Number,
  "FragmentStartProducerTimestamp":Fragment Start Producer Timestamp,
  "FragmentStartServerTimestamp":Fragment Start Server Timestamp,
  "NotificationType":"PERSISTED",
  "NotificationPayload":{
    CUSTOM_KEY_1:CUSTOM_VALUE_1,
    CUSTOM_KEY_2:CUSTOM_VALUE_2
  }
}
```

Amazon SNS メッセージを表示する

Amazon SNS トピックからメッセージを直接読み取ることはできません。そのための API がないためです。メッセージを表示するには、SQS キューを SNS トピックにサブスクライブするか、他の [Amazon SNS でサポートされている送信先](#) を選択します。ただし、メッセージを表示するための最も効率的なオプションは、Amazon SQS を使用することです。

Amazon SQS を使用して Amazon SNS メッセージを表示するには Amazon SQS

1. [Amazon SQS キュー](#) を作成します。
2. から AWS マネジメントコンソール、で送信先として設定された Amazon SNS トピックを開きます NotificationConfiguration。
3. サブスクリプションの作成を選択し、最初のステップで作成した Amazon SQS キューを選択します。
4. 通知設定を有効にし、通知 MKV タグをフラグメントに追加して PutMedia セッションを実行します。
5. Amazon SQS コンソールで Amazon SQS キューを選択し、Amazon SQS キューのメッセージの送受信を選択します。

6. メッセージのポーリング。このコマンドには、PutMediaセッションによって生成されたすべての通知が表示されます。ポーリングの詳細については、[Amazon SQSショートポーリングとロングポーリング](#)を参照してください。

クロスアカウント Amazon SNS 通知の発行

別の AWS アカウントのトピックに Amazon SNS 通知を発行するには、アイデンティティベースのポリシーとリソースベースのポリシーの両方を設定する必要があります。この設定により、Kinesis Video Streams は、あるアカウントから別のアカウントの Amazon SNS トピックに通知を発行できます。

ID ベースのポリシー設定

PutMedia API を呼び出す IAM ロールまたはユーザーには、クロスアカウント Amazon SNS トピックに対するsns:Publishアクセス許可が必要です。ID ベースのポリシーに次のポリシーステートメントを追加します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesisvideo:PutMedia",
      "Resource": "arn:aws:kinesisvideo:us-east-1:123456789012:stream/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "arn:aws:sns:us-east-1:123456789012:*"
      ]
    }
  ]
}
```

リソースベースのポリシー設定

送信先アカウントの Amazon SNS トピックには、送信元アカウントにメッセージの発行を許可するリソースベースのアクセスポリシーが必要です。Amazon SNS トピックアクセスポリシーを次のように設定します。

JSON

```
{
  "Version": "2012-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:us-east-1:123456789012:topic-name"
    }
  ]
}
```

を Kinesis Video Streams ストリームがある AWS アカウント ID <kvs_streams_account_id> に置き換え、 を Amazon SNS トピックの ARN <sns_topic_arn> に置き換えます。

要件と考慮事項

- アイデンティティベースのポリシー (ソースアカウント内) とリソースベースのポリシー (宛先アカウント内) の両方を、クロスアカウント発行が機能するように設定する必要があります。
- PutMedia オペレーションに使用される IAM ロールには、ロールエイリアスで IoT 証明書を使用する場合でも、sns:Publish アクセス許可を含める必要があります。

ビデオストリームからイメージを抽出する

Amazon Kinesis Video Streams API と SDKs を使用して、Kinesis ビデオストリームからオンデマンドのイメージ抽出と自動イメージ抽出をリアルタイムで実行できます。APIs これらのイメージは、サムネイルやスクラブなどの拡張再生アプリケーションや、機械学習ワークフローでの使用に使用できます。

Kinesis Video Streams は、ビデオストリームからのイメージの抽出を 2 つの方法でサポートしています。

- [オンデマンドイメージ生成](#) - [the section called “GetImages”](#) API を使用して、Kinesis Video Streams に保存されているビデオから 1 つまたは複数のイメージを抽出します。
- [the section called “リアルタイムイメージ生成”](#) - 取り込み時にビデオのフラグメントタグに基づいてリアルタイムでビデオデータから画像を自動的に抽出し、S3 バケットに画像を配信するように Kinesis Video Streams を設定します。

自動リアルタイムイメージ生成

Amazon Kinesis Video Streams は、イメージをトランスコードして配信する機能を提供します。Kinesis Video Streams は、動画データからリアルタイムで画像を自動的に抽出し、指定した Amazon S3 バケットに画像を配信します。リアルタイムの自動イメージ抽出を実装するには、次のステップを実行します。

- 生成されたイメージを受け取る S3 バケットを作成します。
- [ImageGenerationConfiguration](#) ストリームプロパティを設定して、イメージの作成方法と送信先を Kinesis Video Streams に指示します。
- イメージ生成タグの追加 – Kinesis Video Streams は、イメージ生成タグを持つフラグメントのみを使用してイメージを生成します。これらのタグは、Kinesis Video Streams プロデューサー SDK と `putKinesisVideoEventMetadata` メソッドを使用してビデオをアップロードするときに追加されます。

次の手順では、これらの各ステップを実行する手順を示します。

カスタマーマネージドキーを使用している場合は、PutMedia呼び出しを実行するロール (アップローダー) に、データの暗号化と復号、および Amazon S3 バケットへのアクセスに必要な次のアクセス許可があることを確認します。

- kms:Encrypt
- kms:GenerateDataKey
- kms:Decrypt
- s3:PutObject

詳細については、「[the section called “サーバー側の暗号化の使用開始方法”](#)」を参照してください。

生成されたイメージの送信先を設定するには

1. イメージが送信される S3 送信先バケットを作成します。

[Amazon S3 ユーザーガイド](#)に従って Amazon S3 バケットを作成します。

バケットの URI を書き留めます。これは、ストリームのイメージ生成設定を更新する次のステップで必要になります。

2. AWS CLI がインストールされ、設定されていることを確認します。詳細については、[AWS Command Line Interface 「バージョン 2 ユーザーガイド」](#)を参照してください。
3. 次のコンテンツを入力として という名前 `update-image-generation-input.json` の新しいファイルを作成します。プレースホルダー値を、使用する値で更新します。サポートされる最大値と最小値については、[UpdateImageGenerationConfiguration](#) API を参照してください。

```
{
  "StreamName": "demo-stream",
  "ImageGenerationConfiguration": {
    "Status": "ENABLED",
    "DestinationConfig": {
      "DestinationRegion": "us-east-1",
      "Uri": "s3://my-bucket-name"
    },
    "SamplingInterval": 200,
    "ImageSelectorType": "PRODUCER_TIMESTAMP",
    "Format": "JPEG",
    "FormatConfig": {
      "JPEGQuality": "80"
    },
    "WidthPixels": 320,
    "HeightPixels": 240
  }
}
```

4. 次のコマンドに示すように、[the section called “UpdateImageGenerationConfiguration”](#) API を使用してストリームのイメージ生成設定を更新し、JSON ファイルを入力としてアタッチします。ファイルパスは、現在のディレクトリ内の ファイルを指すことに注意してください。

```
aws kinesisvideo update-image-generation-configuration \  
  --cli-input-json file://./update-image-generation-input.json
```

5. 成功すると、空のレスポンスが返され、ターミナルに何も出力されません。

 Note

イメージ生成設定を更新してからイメージ生成ワークフローを開始するには、少なくとも 1 分かかります。少なくとも 1 分待ってから、ビデオをストリームにアップロードします。

6. 設定を確認します。AWS CLI を使用して、ストリームの [the section called “DescribeImageGenerationConfiguration”](#) API を呼び出します。

```
aws kinesisvideo describe-image-generation-configuration \  
  --stream-name "demo-stream"
```

Kinesis Video Streams は、イメージ生成タグを持つフラグメントのイメージのみを生成して配信します。Amazon S3 イメージ生成タグとともに提供される追加のフラグメントメタデータは、Amazon S3 メタデータとして保存されます。

 Note

イメージ生成タグは、ストリームレベルのタグではなく、フラグメントメタデータタグを指します。

 Important

イメージ生成タグは、フラグメントメタデータタグの制限にカウントされます。詳細については、「[the section called “ストリーミングメタデータサービスクォータ”](#)」を参照してください。

以下は、mkvinfoユーティリティを使用したフラグメントメタデータタグ構造の例です。イメージ生成タグは、キーがで値AWS_KINESISVIDEO_IMAGE_GENERATIONのない MKV シンプルタグです。詳細については、Matroska ドキュメントの「[ビデオタグの例](#)」を参照してください。

```
|+ Tags
| + Tag
| // MANDATORY: Predefined MKV tag to trigger image generation for the fragment
| + Simple
|   + Name: AWS_KINESISVIDEO_IMAGE_GENERATION

| // OPTIONAL: S3 prefix which will be set as prefix for generated image.
| + Simple
|   + Name: AWS_KINESISVIDEO_IMAGE_PREFIX
|   + String: image_prefix_in_s3 // 256 bytes max

| // OPTIONAL: Key value pairs that will be persisted as S3 Image object metadata.
| + Simple
|   + Name: CUSTOM_KEY_1 // Max 128 bytes
|   + String: CUSTOM_VALUE_1 // Max 256 bytes
| + Simple
|   + Name: CUSTOM_KEY_2 // Max 128 bytes
|   + String: CUSTOM_VALUE_2 // Max 256 bytes
```

フラグメントへのイメージ生成タグの追加

Kinesis Video Streams は、イメージ生成タグを持つフラグメントに対してのみイメージを生成および配信します。Kinesis Video Streams は、これらの特別な MKV タグを認識し、ストリームのイメージ処理設定に基づいてイメージ生成ワークフローを開始します。

Kinesis Video Streams プロデューサー SDK を使用してメディアをアップロードする場合、putKinesisVideoEventMetadataメソッドを使用して、タグ付けする各フラグメントにイメージ生成タグを追加します。keyframe フラグを含むフレームで putFrameが呼び出されると、新しいフラグメントが開始されます。

録画済みの動画をアップロードする場合、ネットワーク速度に応じて、録画された動画の速度とは異なる速度でアップロードされる可能性があります。ビデオの元のタイムスタンプに基づいて定期的にイメージを生成し、Amazon Kinesis Video Streams がビデオを受信したレートに基づいて生成されたサーバータイムスタンプを使用しない場合は、プロデューサータイムスタンプを使用してイメージ生成を設定することをお勧めします。

このコードの完全な例を表示するには、GitHub の[VideoOnlyRealtimeStreamingSample](#)コードサンプルを参照してください。

```
// Setup sample frame
memset(frameBuffer, 0x00, frameSize);
frame.frameData = frameBuffer;
frame.version = FRAME_CURRENT_VERSION;
frame.trackId = DEFAULT_VIDEO_TRACK_ID;
frame.duration = HUNDREDS_OF_NANOS_IN_A_SECOND / DEFAULT_FPS_VALUE;
frame.decodingTs = defaultGetTime(); // current time
frame.presentationTs = frame.decodingTs;

Frame eofr = EOFR_FRAME_INITIALIZER;

while(defaultGetTime() > streamStopTime) {
    frame.index = frameIndex;
    frame.flags = fileIndex % DEFAULT_KEY_FRAME_INTERVAL == 0 ? FRAME_FLAG_KEY_FRAME :
FRAME_FLAG_NONE;
    frame.size = sizeof(frameBuffer);

    CHK_STATUS(readFrameData(&frame, frameFilePath));

    // 1. End the previous fragment
    if (frame.flags == FRAME_FLAG_KEY_FRAME && !firstFrame) {
        putKinesisVideoFrame(streamHandle, &eofr);
    }

    // 2. putFrame call
    CHK_STATUS(putKinesisVideoFrame(streamHandle, &frame));

    if (frame.flags == FRAME_FLAG_KEY_FRAME) {
        // 3. Adding the image generation tag
        CHK_STATUS(putKinesisVideoEventMetadata(streamHandle,
STREAM_EVENT_TYPE_IMAGE_GENERATION, NULL));

        // 4. Adding fragment metadata
        for (n = 1; n <= 5; n++) {
            SNPRINTF(metadataKey, METADATA_MAX_KEY_LENGTH, "SAMPLE_KEY_%d", n);
            SNPRINTF(metadataValue, METADATA_MAX_VALUE_LENGTH, "SAMPLE_VALUE_%d",
frame.index + n);
            CHK_STATUS(putKinesisVideoFragmentMetadata(streamHandle, metadataKey,
metadataValue, FALSE));
        }
    }
}
```

```

    }
    defaultThreadSleep(frame.duration);

    frame.decodingTs += frame.duration;
    frame.presentationTs = frame.decodingTs;
    frameIndex++;
    fileIndex++;
    fileIndex = fileIndex % NUMBER_OF_FRAME_FILES;
    firstFrame = TRUE;
}

// 5. End the final fragment
putKinesisVideoFrame(streamHandle, &eofr);

```

サンプルフレームを設定するためのサンプルコードの要素は、次のように説明されています。

1. 各フラグメントはフラグメントの末尾 () で終わる必要がありますeofr。このステートメントでは、次のフレームの先頭を示す新しいキーフレームを受信するたびに、次のフレームをストリームに追加するeofr前に を配置します。
2. 現在のフレームをストリームに配置します。
3. イメージ生成タグを追加します。putKinesisVideoEventMetadata メソッドは、呼び出し後および の前にいつでもputFrame(keyFrame)呼び出すことができますputFrame(eofr)。フラグメントごとに呼び出されるのは最大 1 回のみです。すべてのフラグメントにはキーフレームが 1 つしかないので、現時点ではわかりやすくするために呼び出されます。の戻り値は、成功コード (0) putKinesisVideoEventMetadataをチェックします。
4. Kinesis Video Streams が Amazon S3 オブジェクトメタデータに変換する他のカスタムフラグメントメタデータを追加します。
5. このアップロードセッションの最終フラグメントを終了します。

サンプルを使用して画像生成タグを追加する

コマンドラインオプションでイメージ生成タグを追加する場合は、C++ プロデューサー SDK kvs_gstreamer_audio_video_sampleで を使用できます。次の例に示すように、-e image または -e both引数を追加して、この機能を有効にします。

```

./kvs_gstreamer_audio_video_sample stream-name \
-f video-to-upload.mp4 \
-e both

```

このサンプルアプリケーションの詳細については、GitHub の [Amazon Kinesis Video Streams CPP プロデューサー](#)、[GStreamer プラグイン](#)、[JNI README](#) を参照してください。

Amazon S3 オブジェクトパス (イメージ)

S3 オブジェクトパスは、生成されたイメージが配信される設定済み S3 バケット上の場所を記述します。次の形式を使用します。

```
ImagePrefix_AccountID_StreamName_ImageTimecode_RandomID.file-extension
```

オブジェクトパス要素は次のように定義されます。

- ImagePrefix - 存在するAWS_KINESISVIDEO_IMAGE_PREFIX場合は の値。
- AccountID - ストリームが作成される AWS アカウント ID。
- StreamName - イメージの生成元のストリームの名前。
- ImageTimecode - イメージが生成されるフラグメントのエポックタイムコード (ミリ秒単位)。
- RandomID - ランダム GUID。
- file-extension - リクエストされたイメージ形式に基づく JPG または PNG。

この例では、生成されたイメージのオブジェクトパスは次のようになります。

```
111122223333_demo-stream_16907729324_f20f9add-75e7-4399-a30f-fc7aefb1bab7.jpg
```

イメージメタデータの取得

S3 コンソールまたは CLI を使用して、生成されたイメージのメタデータを取得できます。

Kinesis Video Streams は、生成されたイメージのフラグメント番号、プロデューサーとサーバーのタイムスタンプ、コンテンツタイプのメタデータを、すべて Amazon S3 オブジェクトメタデータとしてフォーマットして設定します。追加の MKV タグが存在する場合、それらのタグは Amazon S3 オブジェクトメタデータとしても追加されます。次の例は、Amazon S3 head-object API コマンドを使用してオブジェクトメタデータを取得する方法を示しています。レスポンスには、Kinesis Video Streams によって作成されたメタデータが含まれます。

```
aws s3api head-object --bucket my-bucket-name --key 111122223333_demo-stream_1690707290324_f20f9add-7e57-4399-a30f-fc7aefb1bab7.jpg
{
  "AcceptRanges": "bytes",
```

```
"LastModified": "2023-07-30T08:54:51+00:00",
"ContentLength": 22693,
"ETag": "\"63e03cb6d57f77e2db984c1d344b1083\"",
"ContentType": "image/jpeg",
"ServerSideEncryption": "AES256",
"Metadata": {
  "aws_kinesisvideo_producer_timestamp": "1690707290324",
  "aws_kinesisvideo_server_timestamp": "1690707289209",
  "aws_kinesisvideo_fragment_number":
    "91343852333182036507421233921329142742245756394"
}
```

S3 オブジェクトメタデータの詳細については、「」を参照してください<https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingMetadata.html>。

スロットリングから保護するための Amazon S3 URI の推奨事項

Amazon S3 に何千ものイメージを書き込むと、スロットリングのリスクがあります。詳細については、[S3 プレフィックスの Put リクエストの制限](#)を参照してください。

Amazon S3 プレフィックスは、1 秒あたり 3,500 PUT リクエストの PUT 制限で始まり、一意のプレフィックスについては時間の経過とともに徐々に増加します。Amazon S3 プレフィックスとして日付と時刻を使用しないでください。タイムコード化されたデータは、一度に 1 つのプレフィックスに影響し、定期的に変更され、以前のプレフィックスのスケールアップが無効になります。

高速で一貫性のある Amazon S3 スケーリングを有効にするには、16 進数コードや UUID などのランダムプレフィックスを Amazon S3 送信先 URI に追加することをお勧めします。たとえば、16 進コードプレフィックスは、リクエストを 16 の異なるプレフィックス (一意の 16 進文字ごとのプレフィックス) にランダムに分割します。これにより、Amazon S3 が自動スケーリングされた後、1 秒あたり 56,000 個の PUT リクエストが許可されます。

トラブルシューティング

Amazon S3 バケットに配信されないイメージ

この問題のトラブルシューティングには、注意すべき点がいくつかあります。

1. アクセス許可がない
2. イメージ生成設定が正しくない
3. タグがフラグメントに追加されませんでした

アクセス許可がない

カスタマーマネージド KMS キーを使用している場合は、次のように、PutMedia呼び出しを実行するロール (アップローダー) に適切な暗号化および復号アクセス許可があり、Amazon S3 バケットにアクセスできることを確認します。

- kms:Encrypt
- kms:GenerateDataKey
- kms:Decrypt
- s3:PutObject

詳細については、「[the section called “サーバー側の暗号化の使用開始方法”](#)」を参照してください。

送信先を確認する

AWS CLI を使用して、ストリームの DescribeImageGenerationConfiguration API を呼び出します。

```
aws kinesisisvideo describe-image-generation-configuration \  
  --stream-name "demo-stream"
```

レスポンス DestinationConfig のを確認し、正しく見えることを確認します。

イメージ生成タグがフラグメントに追加されていることを確認します。

1. putKinesisVideoEventMetadata 呼び出しが成功したことを確認します。

putKinesisVideoEventMetadata メソッドは、成功するとステータスコード 0 を返します。0 の関数の戻り値を確認することをお勧めします。0 以外のステータスコードが返された場合は、それを 16 進数に変換し、[the section called “エラーコードのリファレンス”](#)で詳細を確認してください。

エラーログがオンになっていることを確認し、アプリケーション内の他のエラーがないかログを確認します。アプリケーションのフレーム送信呼び出しパターンを確認して、推奨される実装と比較します。[フラグメントへのイメージ生成タグの追加](#)

2. ローカルで生成された MKV ファイルを検証する

プロデューサー SDK またはサンプルアプリケーションがタグを正しく追加したことを確認します。

- a. KVS_DEBUG_DUMP_DATA_FILE_DIR 環境変数を設定します。この値が設定されている場合、プロデューサー SDK は Kinesis Video Streams に送信したメディアファイルを指定された場所書き込みます。

```
export KVS_DEBUG_DUMP_DATA_FILE_DIR=/path/to/output/directory
```

 Note

パスが存在しない場合、SDK は新しいディレクトリを作成しません。必要に応じてフォルダを作成します。

- b. アプリケーションを再度実行します。指定した出力ディレクトリに書き込まれる .mkv ファイルが表示されます。
- c. MKVToolNix または他のソフトウェアを使用して、タグが存在することを確認します。
 - i. MKVToolNix をインストールします。 `brew install mkvtoolnix`
 - ii. 出力ディレクトリのいずれかの .mkv ファイルを使用して MKVToolNix を実行します。

```
mkvinfo -v ./path/to/video/file
```

- iii. MKVToolNix 出力を確認します。プ
□KinesisVideoStream::PutFragmentMetadataデューサー SDK メソッドが正しく呼び出された場合、次の MKV タグが表示されます。

```
|+ Tags
| + Tag
| + Simple
| + Name: AWS_KINESISVIDEO_IMAGE_GENERATION
```

 Note

タグは、その前のクラスターに属します。

- d. MKV タグが存在しない場合は、プ
□KinesisVideoStream::PutEventMetadataデューサー SDK メソッドが STREAM_EVENT_TYPE_IMAGE_GENERATION 引数で呼び出され、成功 (0) コードを返したことを確認します。

ビデオ分析にアクセスする

このセクションでは、パーサーライブラリと Amazon CloudWatch を使用してビデオ分析にアクセスする方法について説明します。

トピック

- [Kinesis ビデオストリームに埋め込まれたメタデータを消費する](#)
- [パーサーライブラリを使用してカメラからの出力を監視する](#)
- [Amazon Kinesis Video Streams のモニタリング](#)
- [ストリーミングメタデータの制限](#)

Kinesis ビデオストリームに埋め込まれたメタデータを消費する

Kinesis のビデオストリーム内のメタデータを使用するには、MkvTagProcessor の実装を使用します。

```
public interface MkvTagProcessor {
    default void process(MkvTag mkvTag, Optional<FragmentMetadata>
currentFragmentMetadata) {
        throw new NotImplementedException("Default
FragmentMetadataVisitor.MkvTagProcessor");
    }
    default void clear() {
        throw new NotImplementedException("Default
FragmentMetadataVisitor.MkvTagProcessor");
    }
}
```

このインターフェイスは、[パーサーライブラリを使用してカメラからの出力を監視する](#) の [FragmentMetadataVisitor](#) クラスにあります。

FragmentMetadataVisitor クラスには MkvTagProcessor の実装が含まれます。

```
public static final class BasicMkvTagProcessor implements
FragmentMetadataVisitor.MkvTagProcessor {
    @Getter
    private List<MkvTag> tags = new ArrayList<>();
}
```

```
@Override
public void process(MkvTag mkvTag, Optional<FragmentMetadata>
currentFragmentMetadata) {
    tags.add(mkvTag);
}

@Override
public void clear() {
    tags.clear();
}
}
```

KinesisVideoRendererExample クラスには、BasicMkvTagProcessor の使用例があります。以下の例では、BasicMkvTagProcessor があるアプリケーションの MediaProcessingArguments に追加されます。

```
if (renderFragmentMetadata) {
    getMediaProcessingArguments =
    KinesisVideoRendererExample.GetMediaProcessingArguments.create(
        Optional.of(new FragmentMetadataVisitor.BasicMkvTagProcessor()));
}
```

BasicMkvTagProcessor.process メソッドは、フラグメントのメタデータが到着すると呼び出されます。蓄積されたメタデータは GetTags を使って取得できます。1 つのメタデータ項目を取得するには、まず clear を呼び出して収集されたメタデータをクリアしてから、メタデータ項目を再度取得します。

パーサーライブラリを使用してカメラからの出力を監視する

Kinesis ビデオストリームパーサーライブラリは、Java アプリケーションで Kinesis ビデオストリームの MKV データを消費するために使用できる一連のツールです。

このライブラリには次のツールが含まれます。

- [StreamingMkvReader](#): このクラスは指定された MKV 要素をビデオストリームから読み取ります。
- [FragmentMetadataVisitor](#): このクラスはフラグメント (メディア要素) およびトラック (音声や字幕といったメディア情報を含む個々のデータストリーム) からメタデータを取得します。
- [OutputSegmentMerger](#): このクラスは、ビデオストリームの連続したフラグメントまたはチャンクを結合します。

- [KinesisVideoExample](#): これは、Kinesis ビデオストリームパーサーライブラリの使用方法を示すサンプルアプリケーションです。

ライブラリには、ツールの使用方法を示すテストも含まれています。

前提条件

Kinesis ビデオストリームパーサーライブラリを調べて使用するには、以下が必要です。

- Amazon Web Services (AWS) アカウント。をまだお持ちでない場合は AWS アカウント、「」を参照してください[the section called “にサインアップする AWS アカウント”](#)。
- [Eclipse Java Neon](#) や [JetBrains IntelliJ Idea](#) などの Java 統合開発環境 (IDE)。
- [Amazon Corretto 11](#) などの Java 11。

コードをダウンロードする

このセクションでは、Java ライブラリおよびテストコードをダウンロードし、プロジェクトを Java IDE にインポートします。

この手順の前提条件その他の詳細については、「[the section called “パーサーライブラリを使用したストリーミング”](#)」を参照してください。

1. ディレクトリを作成し、GitHub リポジトリ (<https://github.com/aws/amazon-kinesis-video-streams-parser-library>) からライブラリソースコードのクローンを作成します。

```
git clone https://github.com/aws/amazon-kinesis-video-streams-parser-library
```

2. 使用している Java IDE ([Eclipse](#) や [IntelliJ IDEA](#) など) を開き、ダウンロードした Apache Maven プロジェクトをインポートします。
 - Eclipse: [File]、[Import]、[Maven]、[Existing Maven Projects] を選択して `kinesis-video-streams-parser-lib` フォルダに移動します。
 - IntelliJ Idea では: [インポート] を選択します。ダウンロードしたパッケージのルートに含まれる `pom.xml` ファイルに移動します。

詳細については、関連する IDE ドキュメントを参照してください。

コードを調べる

このセクションでは、Java ライブラリとテストコードを検証し、ライブラリに含まれるツールを独自のコードで使用方法について学習します。

Kinesis ビデオストリームパーサーライブラリには、次のツールが含まれています。

- [StreamingMkvReader](#)
- [FragmentMetadataVisitor](#)
- [OutputSegmentMerger](#)
- [KinesisVideoExample](#)

StreamingMkvReader

このクラスは指定された MKV 要素をストリームからブロックしない方法で読み取ります。

次のコード例 (FragmentMetadataVisitorTest から) は、Streaming MkvReader を作成、使用して inputStream と呼ばれる入力ストリームから MkvElement オブジェクトを取得する方法を示しています。

```
StreamingMkvReader mkvStreamReader =
    StreamingMkvReader.createDefault(new
InputStreamParserByteSource(inputStream));
while (mkvStreamReader.mightHaveNext()) {
    Optional<MkvElement> mkvElement = mkvStreamReader.nextIfAvailable();
    if (mkvElement.isPresent()) {
        mkvElement.get().accept(fragmentVisitor);
        ...
    }
}
```

FragmentMetadataVisitor

このクラスはフラグメント (メディア要素) のメタデータを取得し、コーデックのプライベートデータ、ピクセル幅、ピクセルの高さなどのメディア情報を含む個々のデータストリームを追跡します。

次のコード例 (FragmentMetadataVisitorTest ファイルから) は FragmentMetadataVisitor を使って MkvElement オブジェクトからデータを取得する方法を示しています。

```

FragmentMetadataVisitor fragmentVisitor = FragmentMetadataVisitor.create();
StreamingMkvReader mkvStreamReader =
    StreamingMkvReader.createDefault(new InputSteamParserByteSource(in));
int segmentCount = 0;
while(mkvStreamReader.mightHaveNext()) {
    Optional<MkvElement> mkvElement = mkvStreamReader.nextIfAvailable();
    if (mkvElement.isPresent()) {
        mkvElement.get().accept(fragmentVisitor);
        if
(MkvTypeInfoos.SIMPLEBLOCK.equals(mkvElement.get().getElementMetaData().getTypeInfo()))
{
            MkvDataElement dataElement = (MkvDataElement) mkvElement.get();
            Frame frame =
((MkvValue<Frame>)dataElement.getValueCopy()).getVal();
            MkvTrackMetadata trackMetadata =
fragmentVisitor.getMkvTrackMetadata(frame.getTrackNumber());
            assertTrackAndFragmentInfo(fragmentVisitor, frame, trackMetadata);
        }
        if
(MkvTypeInfoos.SEGMENT.equals(mkvElement.get().getElementMetaData().getTypeInfo())) {
            if (mkvElement.get() instanceof MkvEndMasterElement) {
                if (segmentCount < continuationTokens.size()) {
                    Optional<String> continuationToken =
fragmentVisitor.getContinuationToken();
                    Assert.assertTrue(continuationToken.isPresent());
                    Assert.assertEquals(continuationTokens.get(segmentCount),
continuationToken.get());
                }
                segmentCount++;
            }
        }
    }
}
}

```

前述の例は、次のコーディングパターンを示しています。

- データ解析のための `FragmentMetadataVisitor` およびデータ提供のための [StreamingMkvReader](#) を作成します。
- ストリーム内の各 `MkvElement` について、そのメタデータが `SIMPLEBLOCK` タイプかどうかを検証します。
- 該当する場合は `MkvElement` から `MkvDataElement` を取得します。

- MkvDataElement から Frame (メディアデータ) を取得します。
- FragmentMetadataVisitor から Frame 用の MkvTrackMetadata を取得します。
- Frame および MkvTrackMetadata オブジェクトから次のデータを取得して検証します。
 - 追跡番号。
 - フレームのピクセルの高さ。
 - フレームのピクセルの幅。
 - フレームのエンコードに使用するコーデックのコーデック ID。
 - このフレームが順番に到着したこと。前のフレームのトラック番号が存在する場合、現在のフレームのトラック番号より小さいことを確認します。

プロジェクトで FragmentMetadataVisitor を使用するには、ビジターの accept 方法を使って MkvElement オブジェクトをビジターにパスします。

```
mkvElement.get().accept(fragmentVisitor);
```

OutputSegmentMerger

このクラスは、ストリーム内の異なるトラックのメタデータを単一のセグメントを持つストリームにマージします。

次のコード例 (FragmentMetadataVisitorTest ファイルから) は、OutputSegmentMerger を使って inputBytes と呼ばれるバイト配列の追跡メタデータをマージする方法を示しています。

```
FragmentMetadataVisitor fragmentVisitor = FragmentMetadataVisitor.create();

ByteArrayOutputStream outputStream = new ByteArrayOutputStream();

OutputSegmentMerger outputSegmentMerger =
    OutputSegmentMerger.createDefault(outputStream);

CompositeMkvElementVisitor compositeVisitor =
    new TestCompositeVisitor(fragmentVisitor, outputSegmentMerger);

final InputStream in = TestResourceUtil.getTestInputStream("output_get_media.mkv");

StreamingMkvReader mkvStreamReader =
    StreamingMkvReader.createDefault(new InputStreamParserByteSource(in));
```

```
while (mkvStreamReader.mightHaveNext()) {
    Optional<MkvElement> mkvElement = mkvStreamReader.nextIfAvailable();
    if (mkvElement.isPresent()) {
        mkvElement.get().accept(compositeVisitor);
    }
    if
(MkvTypeInfos.SIMPLEBLOCK.equals(mkvElement.get().getElementMetadata().getTypeInfo()))
{
        MkvDataElement dataElement = (MkvDataElement) mkvElement.get();
        Frame frame = ((MkvValue<Frame>) dataElement.getValueCopy()).getVal();
        Assert.assertTrue(frame.getFrameData().limit() > 0);
        MkvTrackMetadata trackMetadata =
fragmentVisitor.getMkvTrackMetadata(frame.getTrackNumber());
        assertTrackAndFragmentInfo(fragmentVisitor, frame, trackMetadata);
    }
}
```

前述の例は、次のコーディングパターンを示しています。

- [FragmentMetadataVisitor](#) を作成してストリームからメタデータを取得する。
- 出力ストリームを作成してマージされたメタデータを取得する。
- `OutputSegmentMerger` を作成し、`ByteArrayOutputStream` に渡す。
- 2 つのビジターを含む `CompositeMkvElementVisitor` を作成する。
- 指定されたファイルを指す `InputStream` を作成する。
- 入力データ内の各要素を出力ストリームにマージします。

KinesisVideoExample

これは、Kinesis ビデオストリームパーサーライブラリの使用法を示すサンプルアプリケーションです。

このクラスは次の操作を実行します。

- Kinesis のビデオストリームを作成します。指定した名前がすでに存在する場合は、ストリームが削除され、再作成されます。
- Kinesis のビデオストリームに、動画フラグメントをストリーミングするために、[PutMedia](#) を呼び出します。
- Kinesis のビデオストリームから動画フラグメントをストリーミングするために、[GetMedia](#) を呼び出します。

- [StreamingMkvReader](#) を使用してストリームで返されたフラグメントを解析し、[FragmentMetadataVisitor](#) を使用してフラグメントを記録します。

ストリームを削除して再作成

次のコード例 (StreamOps.java ファイルから) は、特定の Kinesis のビデオストリーム を削除します。

```
//Delete the stream
amazonKinesisVideo.deleteStream(new
    DeleteStreamRequest().withStreamARN(streamInfo.get().getStreamARN()));
```

次のコード例 (StreamOps.java ファイルから) は、指定された名前の Kinesis のビデオストリームを作成します。

```
amazonKinesisVideo.createStream(new CreateStreamRequest().withStreamName(streamName)
    .withDataRetentionInHours(DATA_RETENTION_IN_HOURS)
    .withMediaType("video/h264"));
```

PutMedia 呼び出し

次のコード例 (PutMediaWorker.java ファイルから) は、ストリームの [PutMedia](#) を呼び出します。

```
putMedia.putMedia(new PutMediaRequest().withStreamName(streamName)
    .withFragmentTimecodeType(FragmentTimecodeType.RELATIVE)
    .withProducerStartTimestamp(new Date())
    .withPayload(inputStream), new PutMediaAckResponseHandler() {
    ...
});
```

GetMedia 呼び出し

次のコード例 (GetMediaWorker.java ファイルから) は、ストリームの [GetMedia](#) を呼び出します。

```
GetMediaResult result = videoMedia.getMedia(new
    GetMediaRequest().withStreamName(streamName).withStartSelector(startSelector));
```

GetMedia の結果を解析

このセクションで

は、[StreamingMkvReader](#)、[FragmentMetadataVisitor](#)、CompositeMkvElementVisitor を使用して、GetMedia から返されたデータを解析し、ファイルに保存して、ログに記録する方法について説明します。

StreamingMkvReader で GetMedia の出力を読み込む

次のコード例 (GetMediaWorker.java ファイルから) は、[StreamingMkvReader](#) を作成し、それを使用して [GetMedia](#) オペレーションの結果を解析します。

```
StreamingMkvReader mkvStreamReader = StreamingMkvReader.createDefault(new
    InputStreamParserByteSource(result.getPayload()));
log.info("StreamingMkvReader created for stream {}", streamName);
try {
    mkvStreamReader.apply(this.elementVisitor);
} catch (MkvElementVisitException e) {
    log.error("Exception while accepting visitor {}", e);
}
```

上記のコード例では、[StreamingMkvReader](#) は GetMedia 結果のペイロードから MKVElement オブジェクトを取得します。次のセクションでは、要素は [FragmentMetadataVisitor](#) に渡されます。

FragmentMetadataVisitor を使用してフラグメントを取得する

次のコード例 (KinesisVideoExample.java および StreamingMkvReader.java ファイルから) は、[FragmentMetadataVisitor](#) を作成します。[StreamingMkvReader](#) で反復された MkvElement オブジェクトは、accept メソッドを使用して訪問者に渡されます。

KinesisVideoExample.java: から

```
FragmentMetadataVisitor fragmentMetadataVisitor = FragmentMetadataVisitor.create();
```

StreamingMkvReader.java: から

```
if (mkvElementOptional.isPresent()) {
    //Apply the MkvElement to the visitor
}
```

```
mkvElementOptional.get().accept(elementVisitor);
}
```

要素を記録し、ファイルに書き込む

次のコード例 (KinesisVideoExample.java ファイルから) は、以下のオブジェクトを作成し、それらを GetMediaProcessingArguments 関数の戻り値の一部として返します。

- システムログに書き込む LogVisitor (MkvElementVisitor の拡張)。
- 受信データを MKV ファイルに書き込む OutputStream。
- OutputStream にバインドされたデータをバッファする BufferedOutputStream。
- 同じトラックと EBML データで GetMedia 結果の連続した要素をマージする [the section called “OutputSegmentMerger”](#)
- [FragmentMetadataVisitor](#)、[the section called “OutputSegmentMerger”](#)、を 1 つの要素の訪問者 LogVisitor に構成 CompositeMkvElementVisitor する。

```
//A visitor used to log as the GetMedia stream is processed.
LogVisitor logVisitor = new LogVisitor(fragmentMetadataVisitor);

//An OutputSegmentMerger to combine multiple segments that share track and ebml
metadata into one
//mkv segment.
OutputStream fileOutputStream =
Files.newOutputStream(Paths.get("kinesis_video_example_merged_output2.mkv"),
    StandardOpenOption.WRITE, StandardOpenOption.CREATE);
BufferedOutputStream outputStream = new BufferedOutputStream(fileOutputStream);
OutputSegmentMerger outputSegmentMerger =
OutputSegmentMerger.createDefault(outputStream);

//A composite visitor to encapsulate the three visitors.
CompositeMkvElementVisitor mkvElementVisitor =
    new CompositeMkvElementVisitor(fragmentMetadataVisitor,
outputSegmentMerger, logVisitor);

return new GetMediaProcessingArguments(outputStream, logVisitor,
mkvElementVisitor);
```

その後、メディア処理引数は に渡され GetMediaWorker、次に に渡され ExecutorService、は別のスレッドでワーカーを実行します。

```
GetMediaWorker getMediaWorker = GetMediaWorker.create(getRegion(),
    getCredentialsProvider(),
    getStreamName(),
    new StartSelector().withStartSelectorType(StartSelectorType.EARLIEST),
    amazonKinesisVideo,
    getMediaProcessingArgumentsLocal.getMkvElementVisitor());
executorService.submit(getMediaWorker);
```

コードを実行する

Kinesis ビデオストリームパーサーライブラリには、独自のプロジェクトで使用するためのツールが含まれています。プロジェクトにはこれらのツールに対するユニットテストが含まれており、これを実行することでインストールを検証できます。

ライブラリには次のユニットテストが含まれます。

- mkv
 - ElementSizeAndOffsetVisitorTest
 - MkvValueTest
 - StreamingMkvReaderTest
- ユーティリティ
 - FragmentMetadataVisitorTest
 - OutputSegmentMergerTest

Amazon Kinesis Video Streams のモニタリング

モニタリングは、Amazon Kinesis Video Streams と AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。マルチポイント障害が発生した場合は、その障害をデバッグできるように、AWS ソリューションのすべての部分からモニタリングデータを収集することをお勧めします。Amazon Kinesis Video Streams のモニタリングを開始する前に、以下の質問に対する回答を含むモニタリング計画を作成することをお勧めします。

- モニタリングの目的は何ですか？
- どのリソースをモニタリングしますか？
- どのくらいの頻度でこれらのリソースをモニタリングしますか？
- どのモニタリングツールを利用しますか？

- 誰がモニタリングタスクを実行しますか？
- 問題が発生したときに誰が通知を受け取りますか？

モニタリング目標を定義し、モニタリング計画を作成したら、次のステップとして、環境での通常の Amazon Kinesis Video Streams パフォーマンスのベースラインを確立します。Amazon Kinesis Video Streams のパフォーマンスは、さまざまな時間や負荷条件で測定する必要があります。Amazon Kinesis Video Streams をモニタリングするときに、収集したモニタリングデータの履歴を保存します。現在の Amazon Kinesis Video Streams のパフォーマンスをこの履歴データと比較することで、通常のパフォーマンスパターンとパフォーマンス異常を特定し、発生する可能性のある問題に対処する方法を考案できます。

トピック

- [CloudWatch を使用して Amazon Kinesis Video Streams メトリクスをモニタリングする](#)
- [CloudWatch で Amazon Kinesis Video Streams Edge エージェントをモニタリングする](#)
- [を使用した Amazon Kinesis Video Streams API コールのログ記録 AWS CloudTrail](#)

CloudWatch を使用して Amazon Kinesis Video Streams メトリクスをモニタリングする

Amazon CloudWatch を使用して Amazon CloudWatch は、Amazon Kinesis Video Streams から raw データを収集し、ほぼリアルタイムの読み取り可能なメトリクスに加工します。これらの統計は 15 か月間記録されるため、履歴情報にアクセスしてウェブアプリケーションまたはサービスのパフォーマンスをよりの確に把握できます。

[Amazon Kinesis Video Streams コンソールでは](#)、Amazon Kinesis ビデオストリームの CloudWatch メトリクスを 2 つの方法で表示できます。

- [Dashboard] (ダッシュボード) ページで、[Account-level metrics for Current Region] (現在のリージョンのアカウントレベルのメトリクス) セクションの [Video streams] (ビデオストリーム) タブを選択します。
- ビデオストリームの詳細ページで、[モニタリング] タブを選択します。

Amazon Kinesis Video Streams には、次のメトリクスが用意されています。

メトリクス	説明
ArchivedFragmentsConsumed.Media	すべての APIs によって消費されたフラグメントメディアクォータポイントの数。クォータポイントの概念の説明については、the section called “フラグメントメタデータクォータとフラグメントメディアクォータ” を参照してください。 単位: カウント
ArchivedFragmentsConsumed.Metadata	すべての APIs によって消費されたフラグメントメタデータクォータポイントの数。クォータポイントの概念の説明については、the section called “フラグメントメタデータクォータとフラグメントメディアクォータ” を参照してください。 単位: カウント
PutMedia.Requests	特定のストリームの PutMedia API リクエストの数。 単位: カウント
PutMedia.IncomingBytes	ストリームPutMediaの の一部として受信したバイト数。 単位: バイト
PutMedia.IncomingFragments	ストリームPutMediaの の一部として受信した完全なフラグメントの数。 単位: カウント
PutMedia.IncomingFrames	ストリームPutMediaの の一部として受信された完全なフレームの数。 単位: カウント
PutMedia.ActiveConnections	サービスホストへの接続の合計数。

メトリクス	説明
	単位: カウント
<code>PutMedia.ConnectionErrors</code>	ストリームPutMediaの接続の確立中にエラーが発生しました。 単位: カウント
<code>PutMedia.FragmentIngestionLatency</code>	Amazon Kinesis Video Streams がフラグメントの最初のバイトを受信したときと最後のバイトを受信したときの時間差。 単位: ミリ秒
<code>PutMedia.FragmentPersistLatency</code>	完全なフラグメントデータが受信されてアーカイブされるまでにかかる時間。 単位: カウント
<code>PutMedia.Latency</code>	接続の確立中の InletService からのリクエストと HTTP レスポンスの時間差。 単位: カウント
<code>PutMedia.BufferingAckLatency</code>	新しいフラグメントの最初のバイトが Amazon Kinesis Video Streams によって受信されてから、フラグメントに対してバッファリング ACK が送信されるまでの時間差。 単位: ミリ秒
<code>PutMedia.ReceivedAckLatency</code>	Amazon Kinesis Video Streams が新しいフラグメントの最後のバイトを受信してから、フラグメントの受信 ACK が送信されるまでの時間差。 単位: ミリ秒

メトリクス	説明
PutMedia.PersistedAckLatency	Amazon Kinesis Video Streams が新しいフラグメントの最後のバイトを受信してから、フラグメントに対して永続 ACK が送信されるまでの時間差。 単位: ミリ秒
PutMedia.ErrorAckCount	ストリームPutMediaの の実行中に送信されたエラー ACKs の数。 単位: カウント
PutMedia.Success	フラグメントが正常に書き込まれるたびに 1。フラグメントが失敗するたびに 0。このメトリクスの平均値は、完全で有効なフラグメントがどれくらい送信されたかを示しています。 単位: カウント
GetMedia.Requests	特定のストリームの GetMedia API リクエストの数。 単位: カウント
GetMedia.OutgoingBytes	特定のストリームの GetMedia API の一部としてサービスから送信された合計バイト数。 単位: バイト
GetMedia.OutgoingFragments	ストリームGetMediaの の実行中に送信されたフラグメントの数。 単位: カウント
GetMedia.OutgoingFrames	特定のストリームGetMediaで 中に送信されたフレームの数。 単位: カウント

メトリクス	説明
GetMedia.MillisBehindNow	現在のサーバーのタイムスタンプと最後に送信されたフラグメントのサーバーのタイムスタンプとの時間差。 単位: ミリ秒
GetMedia.ConnectionErrors	正常に確立されなかった接続の数。 単位: カウント
GetMedia.Success	各フラグメントが正常に送信されると 1、失敗すると 0。平均値は成功率を示しています。  Note 失敗には、400 (ユーザー) エラーと 500 (システム) エラーの両方が含まれます。リクエスト ID を含む AWS リクエストとレスポンスの概要を有効にする方法の詳細については、「リクエスト/レスポンスの概要のログ記録」を参照してください。IDs 単位: カウント
GetMediaForFragmentList.OutgoingBytes	特定のストリームの GetMediaForFragmentList API の一部としてサービスから送信された合計バイト数。 単位: バイト
GetMediaForFragmentList.OutgoingFragments	特定のストリームの GetMediaForFragmentList API の一部としてサービスから送信されたフラグメントの合計数。 単位: カウント

メトリクス	説明
GetMediaForFragmentList.OutgoingFrames	特定のストリームの GetMediaForFragmentList API の一部としてサービスから送信されたフレームの合計数。 単位: カウント
GetMediaForFragmentList.Requests	特定のストリームの GetMediaForFragmentList API リクエストの数。 単位: カウント
GetMediaForFragmentList.Success	各フラグメントが正常に送信されると 1、失敗すると 0。平均値は成功率を示しています。  Note 失敗には、400 (ユーザー) エラーと 500 (システム) エラーの両方が含まれます。リクエスト ID を含む AWS リクエストとレスポンスの概要を有効にする方法の詳細については、「リクエスト/レスポンスの概要のログ記録」を参照してください。IDs 単位: カウント
ListFragments.Latency	指定されたストリーム名の ListFragments API コールのレイテンシー。 単位: ミリ秒
ListFragments.Requests	特定のストリームの ListFragments API リクエストの数。 単位: カウント

メトリクス	説明
ListFragments.Success	各リクエストが成功すると 1、失敗すると 0。平均値は成功率を示しています。  Note 失敗には、400 (ユーザー) エラーと 500 (システム) エラーの両方が含まれます。リクエスト ID を含む AWS リクエストとレスポンスの概要を有効にする方法の詳細については、「リクエスト/レスポンスの概要のログ記録」を参照してください。IDs
	単位: カウント
GetHLSStreamingSessionURL.Latency	指定されたストリーム名の GetHLSStreamingSessionURL API コールのレイテンシー。 単位: ミリ秒
GetHLSStreamingSessionURL.Requests	特定のストリームの GetHLSStreamingSessionURL API リクエストの数。 単位: カウント

メトリクス	説明
<code>GetHLSStreamingSessionURL.Success</code>	各リクエストが成功すると 1、失敗すると 0。平均値は成功率を示しています。  Note 失敗には、400 (ユーザー) エラーと 500 (システム) エラーの両方が含まれます。リクエスト ID を含む AWS リクエストとレスポンスの概要を有効にする方法の詳細については、「リクエスト/レスポンスの概要のログ記録」を参照してください。IDs 単位: カウント
<code>GetHLSMasterPlaylist.Latency</code>	指定されたストリーム名の <code>GetHLSMasterPlaylist</code> API コールのレイテンシー。 単位: ミリ秒
<code>GetHLSMasterPlaylist.Requests</code>	特定のストリームの <code>GetHLSMasterPlaylist</code> API リクエストの数。 単位: カウント

メトリクス	説明
GetHLSMasterPlaylist.Success	各リクエストが成功すると 1、失敗すると 0。平均値は成功率を示しています。  Note 失敗には、400 (ユーザー) エラーと 500 (システム) エラーの両方が含まれます。リクエスト ID を含む AWS リクエストとレスポンスの概要を有効にする方法の詳細については、「リクエスト/レスポンスの概要のログ記録」を参照してください。IDs
	単位: カウント
GetHLSMediaPlaylist.Latency	指定されたストリーム名の GetHLSMediaPlaylist API コールのレイテンシー。 単位: ミリ秒
GetHLSMediaPlaylist.Requests	特定のストリームの GetHLSMediaPlaylist API リクエストの数。 単位: カウント

メトリクス	説明
GetHLSMediaPlaylist.Success	各リクエストが成功すると 1、失敗すると 0。平均値は成功率を示しています。  Note 失敗には、400 (ユーザー) エラーと 500 (システム) エラーの両方が含まれます。リクエスト ID を含む AWS リクエストとレスポンスの概要を有効にする方法の詳細については、「リクエスト/レスポンスの概要のログ記録」を参照してください。IDs 単位: カウント
GetMP4InitFragment.Latency	指定されたストリーム名の GetMP4InitFragment API コールのレイテンシー。 単位: ミリ秒
GetMP4InitFragment.Requests	特定のストリームの GetMP4InitFragment API リクエストの数。 単位: カウント

メトリクス	説明
GetMP4InitFragment.Success	各リクエストが成功すると 1、失敗すると 0。平均値は成功率を示しています。  Note 失敗には、400 (ユーザー) エラーと 500 (システム) エラーの両方が含まれます。リクエスト ID を含む AWS リクエストとレスポンスの概要を有効にする方法の詳細については、「リクエスト/レスポンスの概要のログ記録」を参照してください。IDs 単位: カウント
GetMP4MediaFragment.Latency	指定されたストリーム名の GetMP4MediaFragment API コールのレイテンシー。 単位: ミリ秒
GetMP4MediaFragment.Requests	特定のストリームの GetMP4MediaFragment API リクエストの数。 単位: カウント

メトリクス	説明
GetMP4MediaFragment.Success	各リクエストが成功すると 1、失敗すると 0。平均値は成功率を示しています。  Note 失敗には、400 (ユーザー) エラーと 500 (システム) エラーの両方が含まれます。リクエスト ID を含む AWS リクエストとレスポンスの概要を有効にする方法の詳細については、「リクエスト/レスポンスの概要のログ記録」を参照してください。IDs 単位: カウント
GetMP4MediaFragment.OutgoingBytes	特定のストリームの GetMP4MediaFragment API の一部としてサービスから送信された合計バイト数。 単位: バイト
GetTSFragment.Latency	指定されたストリーム名の GetTSFragment API コールのレイテンシー。 単位: ミリ秒
GetTSFragment.Requests	特定のストリームの GetTSFragment API リクエストの数。 単位: カウント

メトリクス	説明
GetTSFragment.Success	各リクエストが成功すると 1、失敗すると 0。平均値は成功率を示しています。  Note 失敗には、400 (ユーザー) エラーと 500 (システム) エラーの両方が含まれます。リクエスト ID を含む AWS リクエストとレスポンスの概要を有効にする方法の詳細については、「リクエスト/レスポンスの概要のログ記録」を参照してください。IDs 単位: カウント
GetTSFragment.OutgoingBytes	特定のストリームの GetTSFragment API の一部としてサービスから送信された合計バイト数。 単位: バイト
GetDASHStreamingSessionURL.Latency	指定されたストリーム名の GetDASHStreamingSessionURL API コールのレイテンシー。 単位: ミリ秒
GetDASHStreamingSessionURL.Requests	特定のストリームの GetDASHStreamingSessionURL API リクエストの数。 単位: カウント

メトリクス	説明
<code>GetDASHStreamingSessionURL.Success</code>	各リクエストが成功すると 1、失敗すると 0。平均値は成功率を示しています。  Note 失敗には、400 (ユーザー) エラーと 500 (システム) エラーの両方が含まれます。リクエスト ID を含む AWS リクエストとレスポンスの概要を有効にする方法の詳細については、「リクエスト/レスポンスの概要のログ記録」を参照してください。IDs 単位: カウント
<code>GetDASHManifest.Latency</code>	指定されたストリーム名の <code>GetDASHManifest</code> API コールのレイテンシー。 単位: ミリ秒
<code>GetDASHManifest.Requests</code>	特定のストリームの <code>GetDASHManifest</code> API リクエストの数。 単位: カウント

メトリクス	説明
GetDASHManifest.Success	各リクエストが成功すると 1、失敗すると 0。平均値は成功率を示しています。  Note 失敗には、400 (ユーザー) エラーと 500 (システム) エラーの両方が含まれます。リクエスト ID を含む AWS リクエストとレスポンスの概要を有効にする方法の詳細については、「リクエスト/レスポンスの概要のログ記録」を参照してください。IDs 単位: カウント
GetClip.Latency	指定されたビデオストリーム名の GetClip API コールのレイテンシー。 単位: ミリ秒
GetClip.Requests	特定のビデオストリームに対する GetClip API リクエストの数。 単位: カウント

メトリクス	説明
GetClip.Success	各リクエストが成功すると 1、失敗すると 0。平均値は成功率を示しています。  Note 失敗には、400 (ユーザー) エラーと 500 (システム) エラーの両方が含まれます。リクエスト ID を含む AWS リクエストとレスポンスの概要を有効にする方法の詳細については、「リクエスト/レスポンスの概要のログ記録」を参照してください。IDs 単位: カウント
GetClip.OutgoingBytes	特定のビデオストリームの GetClip API の一部としてサービスから送信された合計バイト数。 単位: バイト

CloudWatch メトリクスガイダンス

CloudWatch メトリクスは、以下の質問に対する回答を見つけるのに役立ちます。

トピック

- [データは Amazon Kinesis Video Streams サービスに到達していますか？](#)
- [Amazon Kinesis Video Streams サービスによってデータが正常に取り込まれないのはなぜですか？](#)
- [プロデューサーから送信されるのと同じ速度で Amazon Kinesis Video Streams サービスからデータを読み取れないのはなぜですか？](#)
- [コンソールにビデオが含まれないのはなぜですか？ また、ビデオの再生に遅延があるのはなぜですか？](#)
- [リアルタイムのデータの読み取りの遅延とは何ですか？ また、クライアントがストリームの先頭から遅延するのはなぜですか？](#)

- [クライアントは Kinesis ビデオストリームからデータを読み込んでいますか？ また、そのレートはどれだけですか？](#)
- [クライアントが Kinesis ビデオストリームからデータを読み込むことはできないのはなぜですか？](#)

データは Amazon Kinesis Video Streams サービスに到達していますか？

関連するメトリクス:

- PutMedia.IncomingBytes
- PutMedia.IncomingFragments
- PutMedia.IncomingFrames

アクション項目:

- これらのメトリクスが低下した場合は、アプリケーションがまだサービスにデータを送信しているかどうかを確認します。
- ネットワーク帯域幅を確認します。ネットワーク帯域幅が不十分な場合は、それが原因でサービスがデータを受信するレートが低下している可能性があります。

Amazon Kinesis Video Streams サービスによってデータが正常に取り込まれないのはなぜですか？

関連するメトリクス:

- PutMedia.Requests
- PutMedia.ConnectionErrors
- PutMedia.Success
- PutMedia.ErrorAckCount

アクション項目:

- が増加している場合はPutMedia.ConnectionErrors、プロデューサークライアントが受信した HTTP レスポンスとエラーコードを調べて、接続の確立中に発生しているエラーを確認します。
- の低下PutMedia.Successまたは増加がある場合はPutMedia.ErrorAckCount、サービスによって送信された ack レスポンスの ack エラーコードを参照して、データの取り込みが失敗した理由を確認してください。詳細については、「[AckErrorCode.Values](#)」を参照してください。

プロデューサーから送信されるのと同じ速度で Amazon Kinesis Video Streams サービスからデータを読み取れないのはなぜですか？

関連するメトリクス:

- PutMedia.FragmentIngestionLatency
- PutMedia.IncomingBytes

アクション項目:

- これらのメトリクスが低下した場合は、接続のネットワーク帯域幅を確認してください。低帯域幅接続により、データはより低いレートでサービスに到達する可能性があります。

コンソールにビデオが含まれないのはなぜですか？ また、ビデオの再生に遅延があるのはなぜですか？

関連するメトリクス:

- PutMedia.FragmentIngestionLatency
- PutMedia.FragmentPersistLatency
- PutMedia.Success
- ListFragments.Latency
- PutMedia.IncomingFragments

アクション項目:

- の増加PutMedia.FragmentIngestionLatencyまたは減少がある場合はPutMedia.IncomingFragments、ネットワーク帯域幅とデータがまだ送信されているかどうかを確認します。
- にドロップがある場合はPutMedia.Success、ack エラーコードを確認してください。詳細については、「[AckErrorCode.Values](#)」を参照してください。
- PutMedia.FragmentPersistLatency または が増加するとListFragments.Latency、サービスの問題が発生する可能性が高くなります。条件が長期間続く場合は、カスタマーサービスの連絡先に問い合わせて、サービスに問題があるかどうかを確認します。

リアルタイムのデータの読み取りの遅延とは何ですか？ また、クライアントがストリームの先頭から遅延するのはなぜですか？

関連するメトリクス:

- `GetMedia.MillisBehindNow`
- `GetMedia.ConnectionErrors`
- `GetMedia.Success`

アクション項目:

- が増加した場合 `GetMedia.ConnectionErrors`、コンシューマーはストリームへの再接続を頻繁に試みるため、ストリームの読み取りに遅れている可能性があります。 `GetMedia` リクエストに対して返される HTTP レスポンス/エラーコードを確認します。
- にドロップがある場合 `GetMedia.Success`、サービスがコンシューマーにデータを送信できないために接続が切断され、コンシューマーから再接続され、コンシューマーがストリームの先頭から遅れる可能性があります。
- が増加している場合は `GetMedia.MillisBehindNow`、帯域幅の制限を調べて、帯域幅が小さいためにデータの受信速度が遅いかどうかを確認します。

クライアントは Kinesis ビデオストリームからデータを読み込んでいますか？ また、そのレートはどれだけですか？

関連するメトリクス:

- `GetMedia.OutgoingBytes`
- `GetMedia.OutgoingFragments`
- `GetMedia.OutgoingFrames`
- `GetMediaForFragmentList.OutgoingBytes`
- `GetMediaForFragmentList.OutgoingFragments`
- `GetMediaForFragmentList.OutgoingFrames`

アクション項目:

- これらのメトリクスは、リアルタイムデータとアーカイブデータを読み取る速度を示します。

クライアントが Kinesis ビデオストリームからデータを読み込むことはできないのはなぜですか？

関連するメトリクス:

- `GetMedia.ConnectionErrors`
- `GetMedia.Success`
- `GetMediaForFragmentList.Success`
- `PutMedia.IncomingBytes`

アクション項目:

- が増加している場合は `GetMedia.ConnectionErrors`、`GetMedia` リクエストによって返される HTTP レスポンスとエラーコードを確認します。詳細については、「[AckErrorCode.Values](#)」を参照してください。
- 最新データまたはライブデータを読み取ろうとする場合は、サービスがコンシューマーに送信するデータがストリームに入る `PutMedia.IncomingBytes` かどうかを確認します。
- `GetMedia.Success` または `GetMediaForFragmentList.Success` がドロップされた場合は、サービスがコンシューマーにデータを送信できないことが原因である可能性があります。条件が長期間続く場合は、カスタマーサービスの連絡先に問い合わせ、サービスに問題があるかどうかを確認します。

CloudWatch で Amazon Kinesis Video Streams Edge エージェントをモニタリングする

Amazon CloudWatch を使用して Amazon Kinesis Video Streams Amazon CloudWatch は raw データを収集し、読み取り可能なほぼリアルタイムのメトリクスに加工します。これらの統計は 15 か月間記録されます。この履歴情報を使用すると、ウェブアプリケーションまたは Amazon Kinesis Video Streams Edge Agent サービスのパフォーマンスをよりの確に把握できます。

メトリクスを表示するには、以下を実行します。

1. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/cloudwatch/> で CloudWatch コンソールを開きます。
2. 左側のナビゲーションのメトリクスで、すべてのメトリクスを選択します。
3. 参照タブを選択し、EdgeRuntimeAgent カスタム名前空間を選択します。

Amazon Kinesis Video Streams Edge Agent は、名前空間に次のメトリクスを発行します EdgeRuntimeAgent。

ディメンション	State	説明
ストリー ム名、 RecordJob	実行中	RecordJob の実行時に継続的に発行します。 単位: なし。「1」はRecordJob、がこの状態にある限り発行されます。
	FatalError	RecordJob 致命的なエラーが発生した場合に発行します。 単位: なし。「1」は、このイベントが発生すると 1 回発行されます。  Note 詳細については、「ログ」を参照してください。
	完了	RecordJob が完了すると発行されます。 単位: なし。「1」は、このイベントが発生すると 1 回発行されます。
ストリー ム名、 UploadJob	実行中	UploadJob の実行時に継続的に発行します。 単位: なし。「1」はUploadJob、がこの状態にある限り発行されます。
	FatalError	UploadJob 致命的なエラーが発生した場合に発行します。 単位: なし。「1」は、このイベントが発生すると 1 回発行されます。  Note 詳細については、「ログ」を参照してください。

ディメンション	State	説明
	完了	UploadJob が完了すると発行されます。 単位: なし。「1」は、このイベントが発生すると 1 回発行されます。
ストリーム名	PercentageSpaceUsed	これは、Amazon Kinesis Video Streams Edge Agent 設定で記録メディアに割り当てられた合計領域のうち、使用された割合です。詳細については「the section called “LocalSizeConfig”」を参照してください。 単位: パーセンテージ (スケール 0 ~ 1)。
モノの名前	アライブ	実行中の設定に関係なく、Amazon Kinesis Video Streams Edge Agent から毎分発行します。 これは、Amazon Kinesis Video Streams Edge Agent が稼働していて、設定を受け入れる準備ができているかどうかを理解するために使用できます。 単位: なし。「1」は毎分発行されます。
	RecordJobs.HealthyJobCount	Amazon Kinesis Video Streams Edge Agent で実行されているレコードジョブとスケジュールされたレコードジョブの合計数。 単位: 個
	UploadJobs.HealthyJobCount	Amazon Kinesis Video Streams Edge Agent で実行中およびスケジュールされたアップロードジョブの合計数。 単位: 個
	RecordJobs.UnhealthyJobCount	現在エラーが発生しているレコードジョブの合計数。 単位: 個

ディメンション	State	説明
	UploadJobs.UnhealthyJobCount	現在エラーが発生しているアップロードジョブの合計数。 単位: 個
	RecordJobs.RunningJobCount	アクティブに実行されているレコードジョブの合計数。 単位: 個
	UploadJobs.RunningJobCount	アクティブに実行されているアップロードジョブの合計数。 単位: 個
	RecordJobs.EdgeConfigCount	Amazon Kinesis Video Streams Edge Agent で処理中のレコード設定の合計数。 単位: 個
	UploadJobs.EdgeConfigCount	Amazon Kinesis Video Streams Edge Agent で進行中のアップロード設定の合計数。 単位: 個

Amazon Kinesis Video Streams Edge Agent の CloudWatch メトリクスガイダンス Amazon Kinesis Video Streams

CloudWatch メトリクスは、次の質問に対する回答を見つけるのに役立ちます。

トピック

- [Amazon Kinesis Video Streams Edge エージェントには、記録するのに十分なスペースがありますか？](#)
- [Amazon Kinesis Video Streams Edge エージェントは稼働していますか？](#)
- [異常なジョブはありますか？](#)
- [外部からの介入が必要なジョブはありますか？](#)

Amazon Kinesis Video Streams Edge エージェントには、記録するのに十分なスペースがありますか？

関連するメトリクス: PercentageSpaceUsed

アクション: アクションは必要ありません。

Amazon Kinesis Video Streams Edge エージェントは稼働していますか？

関連するメトリクス: Alive

アクション: このメトリクスの受信を停止すると、Amazon Kinesis Video Streams Edge Agent で次のいずれかまたは複数が発生したことを意味します。

- アプリケーションランタイムの問題: メモリやその他のリソースの制約、バグなど
- シャットダウン、クラッシュ、または終了時にエージェントが実行している AWS IoT デバイス
- AWS IoT デバイスにはネットワーク接続がありません

異常なジョブはありますか？

関連するメトリクス:

- RecordJobs.UnhealthyJobCount
- UploadJobs.UnhealthyJobCount

アクション: ログを検査し、FatalErrorメトリクスを探します。

- FatalError メトリクスが存在する場合、致命的なエラーが発生したため、ジョブを手動で再起動する必要があります。を使用してジョブを手動で再起動する前にStartEdgeConfigurationUpdate、ログを検査し、問題を修正します。
- FatalError メトリクスが存在しない場合、一時的な (致命的ではない) エラーが発生し、Amazon Kinesis Video Streams Edge Agent がジョブを再試行しています。

Note

エージェントが致命的に誤ったジョブを再試行するには、を使用します [the section called “StartEdgeConfigurationUpdate”](#)。

外部からの介入が必要なジョブはありますか？

関連するメトリクス:

- `PercentageSpaceUsed` – これが特定の値を超えると、レコードジョブは一時停止され、スペースが利用可能なとき (メディアの保持期間外になったとき) にのみ再開されます。より高い で更新された設定を送信 `MaxLocalMediaSizeInMB` して、ジョブをすぐに更新できます。
- `RecordJob.FatalError / UploadJob.FatalError` – エージェントのログを調査し、ジョブを再開するための設定を再度送信します。

アクション: 設定を使用して API コールを行い、この問題が発生したジョブを再起動します。

を使用した Amazon Kinesis Video Streams API コールのログ記録 AWS CloudTrail

Amazon Kinesis Video Streams は AWS CloudTrail、Amazon Kinesis Video Streams AWS のサービスのユーザー、ロール、または によって実行されたアクションを記録するサービスであると連携します。Amazon Kinesis Video Streams CloudTrail は、Amazon Kinesis Video Streams へのすべての API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、Amazon Kinesis Video Streams コンソールからの呼び出しと、Amazon Kinesis Video Streams API 操作へのコード呼び出しが含まれます。証跡を作成する場合は、Amazon Kinesis Video Streams のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的なデリバリーを有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。CloudTrail で収集された情報を使用して、Amazon Kinesis Video Streams に対して行ったリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

設定や有効化の方法など、CloudTrail の詳細については、「[AWS CloudTrail ユーザーガイド](#)」を参照してください。

Amazon Kinesis Video Streams と CloudTrail

CloudTrail は、AWS アカウントの作成時にアカウントで有効になります。Amazon Kinesis Video Streams でサポートされているイベントアクティビティが発生すると、そのアクティビティはイベント履歴の他の AWS サービスイベントとともに CloudTrail イベントに記録されます。AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、[CloudTrail イベント履歴でのイベントの表示](#)を参照してください。

Amazon Kinesis Video Streams のイベントなど、AWS アカウントのイベントの継続的な記録については、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョンに適用されます。証跡は、AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理 AWS のサービス するように他の を設定できます。詳細については、次を参照してください:

- [証跡の作成のための概要](#)
- [CloudTrail がサポートするサービスと統合](#)
- [CloudTrail 用 Amazon SNS 通知の構成](#)
- [複数のリージョンから CloudTrail ログファイルを受け取る](#) および [複数のアカウントから CloudTrail ログファイルを受け取る](#)

Amazon Kinesis Video Streams が、CloudTrail ログファイルのイベントとして次のアクションのログ記録をサポートします。

- [CreateStream](#)
- [DeleteStream](#)
- [DescribeStream](#)
- [GetDataEndpoint](#)
- [ListStreams](#)
- [ListTagsForStream](#)
- [TagStream](#)
- [UntagStream](#)
- [UpdateDataRetention](#)
- [UpdateStream](#)

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストが、ルートと ユーザー認証情報のどちらを使用して送信されたか
- リクエストの送信に使用された一時的なセキュリティ認証情報に、ロールとフェデレーテッドユーザーのどちらが使用されたか

- リクエストが、別の AWS のサービスによって送信されたかどうか。

詳細については、「[CloudTrail userIdentity 要素](#)」を参照してください。

例: Amazon Kinesis Video Streams ログファイルエントリ

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントは、任意の出典からの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。CloudTrail ログファイルは、パブリック API 呼び出しの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次の例は、[CreateStream](#) アクションを示す CloudTrail ログエントリです。

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
      },
      "eventTime": "2018-05-25T00:16:31Z",
      "eventSource": "kinesisvideo.amazonaws.com",
      "eventName": "CreateStream",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
      "requestParameters": {
        "streamName": "VideoStream",
        "dataRetentionInHours": 2,
        "mediaType": "mediaType",
        "kmsKeyId": "arn:aws:kms::us-east-1:123456789012:alias",
        "deviceName": "my-device"
      },
      "responseElements": {
        "streamARN": "arn:aws:kinesisvideo:us-east-1:123456789012:stream/VideoStream/12345"
      },
    }
  ]
}
```

```

    "requestID": "db6c59f8-c757-11e3-bc3b-57923b443c1c",
    "eventID": "b7acfc0-6ca9-4ee1-a3d7-c4e8d420d99b"
  },
  {
    "eventVersion": "1.05",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::123456789012:user/Alice",
      "accountId": "123456789012",
      "accessKeyId": "EXAMPLE_KEY_ID",
      "userName": "Alice"
    },
    "eventTime": "2018-05-25:17:06Z",
    "eventSource": "kinesisvideo.amazonaws.com",
    "eventName": "DeleteStream",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
    "requestParameters": {
      "streamARN": "arn:aws:kinesisvideo:us-east-1:012345678910:stream/VideoStream/12345",
      "currentVersion": "keqrjeqkj9"
    },
    "responseElements": null,
    "requestID": "f0944d86-c757-11e3-b4ae-25654b1d3136",
    "eventID": "0b2f1396-88af-4561-b16f-398f8eaea596"
  },
  {
    "eventVersion": "1.05",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::123456789012:user/Alice",
      "accountId": "123456789012",
      "accessKeyId": "EXAMPLE_KEY_ID",
      "userName": "Alice"
    },
    "eventTime": "2014-04-19T00:15:02Z",
    "eventSource": "kinesisvideo.amazonaws.com",
    "eventName": "DescribeStream",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",

```



```

    "requestParameters": {
      "streamName": "VideoStream"
    },
    "responseElements": null,
    "requestID": "a68541ca-c757-11e3-901b-cbcfe5b3677a",
    "eventID": "22a5fb8f-4e61-4bee-a8ad-3b72046b4c4d"
  },
  {
    "eventVersion": "1.05",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::123456789012:user/Alice",
      "accountId": "123456789012",
      "accessKeyId": "EXAMPLE_KEY_ID",
      "userName": "Alice"
    },
    "eventTime": "2014-04-19T00:15:03Z",
    "eventSource": "kinesisvideo.amazonaws.com",
    "eventName": "GetDataEndpoint",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
    "requestParameters": {
      "streamName": "VideoStream",
      "aPIName": "LIST_FRAGMENTS"
    },
    "responseElements": null,
    "requestID": "a6e6e9cd-c757-11e3-901b-cbcfe5b3677a",
    "eventID": "dcd2126f-c8d2-4186-b32a-192dd48d7e33"
  },
  {
    "eventVersion": "1.05",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::123456789012:user/Alice",
      "accountId": "123456789012",
      "accessKeyId": "EXAMPLE_KEY_ID",
      "userName": "Alice"
    },
    "eventTime": "2018-05-25T00:16:56Z",
    "eventSource": "kinesisvideo.amazonaws.com",

```

```
    "eventName": "ListStreams",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
    "requestParameters": {
      "maxResults": 100,
      "streamNameCondition": {"comparisonValue": "MyVideoStream"
comparisonOperator": "BEGINS_WITH"}}
    },
    "responseElements": null,
    "requestID": "e9f9c8eb-c757-11e3-bf1d-6948db3cd570",
    "eventID": "77cf0d06-ce90-42da-9576-71986fec411f"
  }
]
}
```

ストリーミングメタデータの制限

Kinesis [the section called “ストリーミングメタデータサービスクォータ”](#) ビデオストリームへのストリーミングメタデータの追加に適用される制限の詳細については、「」を参照してください。

Amazon Kinesis Video Streams Edge Agent を使用してビデオ録画とストレージをスケジュールする

Amazon Kinesis Video Streams は、お客様のオンプレミスの IP カメラに接続するための効率的で費用対効果の高い方法を提供します。Amazon Kinesis Video Streams Edge Agent を使用すると、カメラからのビデオをローカルで録画して保存し、お客様が定義したスケジュールでビデオをクラウドにストリーミングして、長期保存、再生、分析処理を行うことができます。

Note

Amazon Kinesis Video Streams Edge エージェントにアクセスするには、この[簡単なフォーラム](#)に入力します。

Amazon Kinesis Video Streams Edge エージェントをダウンロードし、オンプレミスのエッジコンピューティングデバイスにデプロイできます。また、Amazon EC2 インスタンスで実行されている Docker コンテナに簡単にデプロイすることもできます。デプロイ後、Amazon Kinesis Video Streams API を使用して、ビデオ録画とクラウドアップロードの設定を更新できます。この機能は、RTSP プロトコル経由でストリーミングできる IP カメラで動作します。カメラに追加のファームウェアデプロイは必要ありません。

Amazon Kinesis Video Streams Edge エージェントには、次のインストールが用意されています。

- AWS IoT Greengrass V2 コンポーネントとして: Amazon Kinesis Video Streams Edge Agent を任意の AWS IoT Greengrass 認定デバイスに AWS IoT Greengrass コンポーネントとしてインストールできます。詳細については AWS IoT Greengrass、[AWS IoT Greengrass Version 2 デベロッパーガイド](#)を参照してください。
- オン AWS Snowball Edge: Snowball Edge デバイスで Amazon Kinesis Video Streams Edge エージェントを実行できます。詳細については、[AWS Snowball Edge 「Edge デベロッパーガイド」](#)を参照してください。
- ネイティブ AWS IoT デプロイの場合: Amazon Kinesis Video Streams Edge エージェントは、任意のコンピューティングインスタンスにネイティブにインストールできます。Edge SDK は、を通じてエッジを管理する[AWS IoT Core](#)ために を使用します [the section called “Amazon Kinesis Video Streams”](#)。

Amazon Kinesis Video Streams Edge Agent の使用を開始するには、以下の手順を実行します。

トピック

- [Amazon Kinesis Video Streams Edge Agent API オペレーション](#)
- [Amazon Kinesis Video Streams Edge エージェントのモニタリング](#)
- [非AWS IoT Greengrass モードでデプロイする](#)
- [Amazon Kinesis Video Streams Edge エージェントを にデプロイする AWS IoT Greengrass](#)
- [Amazon Kinesis Video Streams Edge Agent に関するよくある質問](#)

Amazon Kinesis Video Streams Edge Agent API オペレーション

次の API オペレーションを使用して、Amazon Kinesis Video Streams Edge エージェントを設定します。

- [the section called “StartEdgeConfigurationUpdate”](#)
- [the section called “DescribeEdgeConfiguration”](#)
- [the section called “DeleteEdgeConfiguration”](#)
- [the section called “ListEdgeAgentConfigurations”](#)

Amazon Kinesis Video Streams Edge エージェントのモニタリング

Amazon Kinesis Video Streams Edge エージェントをモニタリングするには、「」を参照してください[the section called “CloudWatch で Amazon Kinesis Video Streams Edge エージェントをモニタリングする”](#)。

非AWS IoT Greengrass モードでデプロイする

このセクションでは、AWS IoT Greengrass 環境外で Amazon Kinesis Video Streams を使用するための包括的なガイドを提供します。エッジデバイスやその他のプラットフォームを使用している場合でも、この情報は Kinesis Video Streams を効果的にセットアップして活用するのに役立ちます。

詳細については、以下を参照してください。

- 開発環境のセットアップ
- Kinesis ビデオストリームの作成
- Kinesis Video Streams プロデューサー SDK のダウンロードとコンパイル

- サンプルアプリケーションの記述と調査
- サンプルアプリケーションの実行

AWS IoT MQTT を使用して Amazon Kinesis Video Streams Edge エージェントをスタンドアロンデプロイとして実行するには、以下の手順に進みます。

トピック

- [依存関係のインストール](#)
- [IP カメラ RTSP URLs のリソースを作成する](#)
- [IAM アクセス権限ポリシーを作成する](#)
- [IAM ロールを作成する](#)
- [AWS IoT ロールエイリアスを作成する](#)
- [AWS IoT ポリシーを作成する](#)
- [AWS IoT モノを作成して AWS IoT Core 認証情報を取得する](#)
- [Amazon Kinesis Video Streams Edge エージェントを構築する](#)
- [デバイスに CloudWatch エージェントをインストールする](#)
- [Amazon Kinesis Video Streams Edge エージェントをネイティブプロセスとして実行する](#)

依存関係のインストール

Amazon Kinesis Video Streams プロデューサー SDK の使用を開始する前に、必要な依存関係を持つ開発環境を設定する必要があります。このページでは、必要なソフトウェアコンポーネントとライブラリをシステムにインストールするプロセスについて説明します。

Note

サポートされているオペレーティングシステムのリストについては、「」を参照してください [the section called “Amazon Kinesis Video Streams Edge Agent はどのオペレーティングシステムをサポートしていますか?”](#)。

デバイスに依存関係をインストールする

1. Amazon Kinesis Video Streams Edge エージェントを実行するには、デバイスに次の適切なライブラリをインストールします。

Ubuntu

タイプ:

```
wget -O- https://apt.corretto.aws/corretto.key | sudo apt-key add -  
sudo add-apt-repository 'deb https://apt.corretto.aws stable main'  
sudo apt-get update  
  
sudo apt-get install -y gcc libssl-dev libcurl4-openssl-dev liblog4cplus-dev \  
libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev \  
gstreamer1.0-plugins-base-apps gstreamer1.0-plugins-bad \  
gstreamer1.0-plugins-good gstreamer1.0-tools \  
unzip java-11-amazon-corretto-jdk maven
```

Amazon Linux 2

タイプ:

```
sudo yum update -y && sudo yum upgrade -y && sudo yum clean all -y  
sudo yum install -y gcc-c++ openssl-devel libcurl-devel gstreamer1* wget \  
java-11-amazon-corretto tar
```

ソースlog4cplus-2.1.0から をインストールします。

```
wget https://github.com/log4cplus/log4cplus/releases/download/REL_2_1_0/  
log4cplus-2.1.0.tar.gz  
tar -xzf log4cplus-2.1.0.tar.gz  
cd log4cplus-2.1.0 && \  
mkdir build && \  
cd build && \  
cmake .. && \  
sudo make && \  
sudo make install
```

ソースapache-maven-3.9.2から をインストールします。

```
wget https://dlcdn.apache.org/maven/maven-3/3.9.2/binaries/apache-maven-3.9.2-  
bin.tar.gz  
RUN tar -xzf apache-maven-3.9.2-bin.tar.gz -C /opt
```

⚠ Important

一部のサービスを再起動する必要があることを示す画面が表示された場合は、Enter キーを押してOK を選択します。

詳細については、[「Amazon Corretto 11 ユーザーガイド」](#)を参照してください。

2. をインストールします AWS Command Line Interface。 [「ユーザーガイド」の最新のバージョンのインストールまたは更新 AWS CLI](#)を参照してください。 AWS Command Line Interface

IP カメラ RTSP URLs のリソースを作成する

以下の手順に従って、に必要なストリームとシークレットを作成します AWS Secrets Manager。最初にこのステップを実行します。これは、ポリシーで作成されたリソースの ARNs が必要なためです。

Amazon Kinesis Video Streams を作成する

AWS マネジメントコンソール AWS CLI、または API を使用して Amazon Kinesis Video Streams を作成します。

で AWS マネジメントコンソール、[Amazon Kinesis Video Streams コンソール](#)を開きます。左側のナビゲーションでビデオストリームを選択します。

詳細については、[「the section called “Amazon Kinesis ビデオストリームを作成する”](#)」を参照してください。

でシークレットを作成する AWS Secrets Manager

で AWS マネジメントコンソール、[AWS Secrets Manager コンソール](#)を開きます。左側のナビゲーションでシークレットを選択します。

適切なリージョンが選択されていることを確認します。

1. 新しいシークレットを保存 を選択します。
 - a. ステップ 1: シークレットタイプを選択する

- [その他のシークレットのタイプ] を選択します。
- 「キーと値のペア」セクションで、キーと値のペアを追加します。

[Key] (キー): MediaURI

 Note

キーは である必要がありますMediaURI。これは大文字と小文字が区別されます。誤って入力すると、アプリケーションは機能しません。

値: *Your MediaURI*。

Example

例: rtsp://<YourCameraIPAddress>:<YourCameraRTSPPort>/
YourCameraMediaURI。

- a. ステップ 2: シークレットを設定する。このシークレットに名前を付けます。任意の名前を付けます。
 - b. ステップ 3: ローテーションを設定する - オプション。[次へ] を選択します。
 - c. ステップ 4: 確認する。[保存する] を選択します。
2. シークレットがすぐに表示されない場合は、更新ボタンを選択します。

シークレットの名前を選択します。シークレット ARN を書き留めます。

3. ストリーミング元の MediaURI ごとにこのプロセスを繰り返します。

 Note

AWS ネットワークは、一部のパブリック RTSP ソースをブロックします。Amazon EC2 インスタンス内から、または VPN への接続中にアンマネージドで実行している場合、これらにアクセスすることはできません。

 Important

カメラ RTSP URL は h.264 形式でビデオをストリーミングする必要があります。フラグメントの期間は、「」で説明されている制限を超えることはできません[the section called “プロデューサー SDK クォータ”](#)。

Amazon Kinesis Video Streams Edge Agent はビデオのみをサポートします。

を実行して `gst-discoverer-1.0 Your RtspUrl`、カメラがデバイスから到達可能であることを確認します。

作成したすべてのストリームとシークレットの ARNs を保存します。次のステップでは、これらが必要です。

IAM アクセス権限ポリシーを作成する

IAM ポリシーを作成するには、次の手順に従います。このアクセス許可ポリシーは、AWS リソースの選択的なアクセスコントロール (サポートされているオペレーションのサブセット) を許可します。この場合、AWS リソースは Amazon Kinesis Video Streams Edge Agent がストリーミングするビデオストリームです。リソースには、Amazon Kinesis Video Streams Edge Agent が取得できる AWS Secrets Manager シークレットも含まれます。詳細については、「[IAM ポリシー](#)」を参照してください。

JSON ポリシーエディタを使用してポリシーを作成する

1. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/iam/> で IAM コンソールを開きます。
2. 左のナビゲーションペインの [ポリシー] を選択します。

初めて [ポリシー] を選択する場合には、[管理ポリシーによるこそ] ページが表示されます。今すぐ始める を選択します。

3. ページの上部で、[ポリシーを作成] を選択します。
4. ポリシーエディタ セクションで、JSON オプションを選択します。
5. 次の JSON ポリシードキュメントを入力します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```

        "Action": [
            "cloudwatch:PutMetricData",
            "kinesisvideo:ListStreams",
            "iot:Connect",
            "iot:Publish",
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource": [
            "*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "kinesisvideo:DescribeStream",
            "kinesisvideo:PutMedia",
            "kinesisvideo:TagStream",
            "kinesisvideo:GetDataEndpoint"
        ],
        "Resource": [
            "arn:aws:kinesisvideo:*:*:stream/streamName1/*",
            "arn:aws:kinesisvideo:*:*:stream/streamName2/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": "secretsmanager:GetSecretValue",
        "Resource": [
            "arn:aws:secretsmanager:*:*:secret:*",
            "arn:aws:secretsmanager:*:*:secret:*"
        ]
    }
]
}

```

Note

arn:aws:kinesisvideo:*:*:stream/streamName1/* と をビデオスト
トリームARNs arn:aws:kinesisvideo:*:*:stream/streamName2/

*に置き換え、 を で作成した MediaURI シークレットを含む ARNs

arn:aws:secretsmanager:*:*:secret:*に置き換えます [the section called “IP カ](#)

[メラ RTSP URLs のリソースを作成する](#)”。Amazon Kinesis Video Streams Edge Agent がアクセスするシークレットの ARNs を使用します。

6. [次へ] を選択します。

 Note

いつでも Visual と JSON エディタオプションを切り替えることができます。ただし、[ビジュアル] エディタで [次へ] に変更または選択した場合、IAM はポリシーを再構成してビジュアルエディタに合わせて最適化することがあります。詳細については、IAM ユーザーガイドの「[ポリシーの再構築](#)」を参照してください。

7. 確認と作成ページで、作成するポリシーのポリシー名とオプションの説明を入力します。このポリシーで定義されているアクセス許可を確認して、ポリシーによって付与されたアクセス許可を確認します。
8. ポリシーを作成 をクリックして、新しいポリシーを保存します。

IAM ロールを作成する

このステップで作成したロールは、AWS Security Token Service () から一時的な認証情報を取得するために AWS IoT によって引き受けることができますAWS STS。これは、Amazon Kinesis Video Streams Edge エージェントから認証情報認可リクエストを実行するときに行われます。

Amazon Kinesis Video Streams のサービスロールを作成する (IAM コンソール)

1. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/iam/> で IAM コンソールを開きます。
2. IAM コンソールのナビゲーションペインで、[ロール]、[ロールを作成] を選択します。
3. カスタム信頼ポリシーのロールタイプを選択し、次のポリシーを貼り付けます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "Service": "credentials.iot.amazonaws.com"
    }
  }
}
```

```
    },  
    "Action": "sts:AssumeRole"  
  }  
}
```

4. で作成した IAM ポリシーの横にあるボックスを選択します [the section called “IAM アクセス権限ポリシーを作成する”](#)。
5. [次へ] を選択します。
6. このロールの目的を特定するのに役立つロール名またはロール名のサフィックスを入力します。

Example

例: KvsEdgeAgentRole

7. (オプション) [説明] には、新しいロールの説明を入力します。
8. (オプション) タグをキーと値のペアとしてアタッチして、ロールにメタデータを追加します。

IAM でのタグの使用の詳細については、IAM ユーザーガイドの [「IAM リソースのタグ付け」](#) を参照してください。

9. ロール情報を確認し、ロールの作成 を選択します。

AWS IoT ロールエイリアスを作成する

で作成した IAM AWS IoT ロールのロールエイリアスを作成するには、次の手順に従います [the section called “IAM ロールを作成する”](#)。ロールエイリアスは、IAM ロールを指す代替データモデルです。AWS IoT 認証情報プロバイダーリクエストには、AWS Security Token Service () から一時的な認証情報を取得するために引き受ける IAM ロールを示すロールエイリアスを含める必要があります AWS STS。詳細については、[「証明書を使用してセキュリティトークンを取得する方法」](#) を参照してください。

AWS IoT ロールエイリアスを作成する

1. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/iot/> で AWS IoT Core コンソールを開きます。
2. 適切なリージョンが選択されていることを確認します。
3. 左側のナビゲーションで、セキュリティを選択し、ロールエイリアスを選択します。
4. ロールエイリアスの作成 を選択します。
5. ロールエイリアスの名前を入力します。

Example

例: KvsEdgeAgentRoleAlias

6. ロールドロップダウンで、で作成した IAM ロールを選択します [the section called “IAM ロールを作成する”](#)。
7. [作成] を選択します。次のページには、ロールエイリアスが正常に作成されたというメモが表示されます。
8. 新しく作成したロールエイリアスを検索して選択します。ロールエイリアス ARN を書き留めます。これは、次のステップで AWS IoT ポリシーに必要です。

AWS IoT ポリシーを作成する

デバイス証明書にアタッチされる AWS IoT ポリシーを作成するには、次の手順に従います。これにより、AWS IoT 機能にアクセス許可が付与され、証明書を使用してロールエイリアスを引き受けることができます。

AWS IoT Core ポリシーを使用すると、AWS IoT Core データプレーンへのアクセスを制御できます。AWS IoT Core データプレーンは、以下を実行するために使用できるオペレーションで構成されます。

- AWS IoT Core メッセージブローカーに接続する
- MQTT メッセージの送受信
- モノのデバイスシャドウを取得または更新する

詳細については、「[AWS IoT Core ポリシー](#)」を参照してください。

AWS IoT ポリシーエディタを使用して AWS IoT ポリシーを作成する

1. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/iot/> で AWS IoT Core コンソールを開きます。
2. 左側のナビゲーションで、セキュリティを選択し、ポリシーを選択します。
3. [Create policy] (ポリシーの作成) を選択します。
4. ポリシーの名前を入力します。

Example

ポリシー名の例としては、KvsEdgeAccessIoTPolicy があります。

5. (オプション) タグをキーと 値のペアとしてアタッチして、メタデータをポリシーに追加します。

IAM でのタグの使用の詳細については、「[AWS IoT Core デベロッパーガイド](#)」の「[AWS IoT リソースのタグ付け](#)」を参照してください。

6. JSON タブを選択します。
7. 以下の JSON ポリシードキュメントを貼り付けます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRoleWithWebIdentity"
      ],
      "Resource": "arn:aws:iot:us-west-2:123456789012:rolealias/your-role-alias"
    }
  ]
}
```

Note

を、で作成したロールエイリアスの ARN `your-role-alias-arn` に置き換えます [the section called “AWS IoT ロールエイリアスを作成する”](#)。

8. 作成を選択して作業を保存します。

AWS IoT モノを作成して AWS IoT Core 認証情報を取得する

この時点で、以下を作成しました。

- IAM アクセス許可ポリシー。「[the section called “IAM アクセス権限ポリシーを作成する”](#)」を参照してください。
- アクセス許可ポリシーがアタッチされた IAM ロール。「[the section called “IAM ロールを作成する”](#)」を参照してください。
- IAM AWS IoT ロールのロールエイリアス。「[the section called “AWS IoT ロールエイリアスを作成する”](#)」を参照してください。
- AWS IoT ポリシー。現在、どの AWS リソースにもアタッチされていません。「[the section called “AWS IoT ポリシーを作成する”](#)」を参照してください。

AWS IoT モノを作成して登録し、AWS IoT Core アクセス認証情報を取得するには

1. デバイスを AWS IoT モノとして登録し、デバイスの X.509 証明書を生成します。
 - a. にサインイン AWS マネジメントコンソールし、<https://console.aws.amazon.com/iot/> で AWS IoT Core コンソールを開きます。
 - b. 適切なリージョンを選択します。
 - c. 左側のナビゲーションで、すべてのデバイスを選択し、モノを選択します。
 - d. モノの作成 を選択します。
 - e. Create single thing を選択し、Next を選択します。
 1. ステップ 1. モノのプロパティを指定する
モノの名前を入力し、次へを選択します。
 2. ステップ 2. デバイス証明書を設定する

新しい証明書の自動生成 (推奨) を選択し、次へ を選択します。

3. ステップ 3. 証明書にポリシーをアタッチする

で作成したアクセス許可ポリシーを検索します [the section called “ AWS IoT ポリシーを作成する”](#)。

ポリシーの横にあるチェックボックスをオンにし、モノの作成を選択します。

f. 表示されるウィンドウで、次のファイルをダウンロードします。

- デバイス証明書。これは X.509 証明書です。
- パブリックキーファイル
- プライベートキーファイル
- Amazon 信頼サービスエンドポイント (RSA 2048 ビットキー: Amazon ルート CA 1)

後のステップで、これらの各ファイルの場所を書き留めます。

g. [Done] (完了) をクリックします。次のページに、モノが正常に作成されたことを示すメモが表示されます。

h. 上記でダウンロードしたファイルを AWS IoT モノにまだ転送していない場合は、転送します。

2. AWS アカウントの認証情報プロバイダーエンドポイントを取得します。

AWS CLI

次のコマンドを実行します。

```
aws iot describe-endpoint --endpoint-type iot:CredentialProvider
```

AWS マネジメントコンソール

で [AWS CloudShell](#)、次のコマンドを実行します。

```
aws iot describe-endpoint --endpoint-type iot:CredentialProvider
```

後のステップでこの情報を書き留めます。

3. AWS アカウントのデバイスデータエンドポイントを取得します。

AWS CLI

次のコマンドを実行します。

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

AWS マネジメントコンソール

以下の操作を実行します。

1. にサインイン AWS マネジメントコンソール し、AWS IoT Core <https://console.aws.amazon.com/iot/> でコンソールを開きます。
2. 左側のナビゲーションで、設定を選択します。
3. デバイスデータエンドポイントを見つけます。

後のステップでこの情報を書き留めます。

4. (オプション) 証明書が正しく生成されたことを確認します。

次のコマンドを実行して、項目が正しく生成されたことを確認します。

```
curl --header "x-amzn-iot-thingname:your-thing-name" \  
  --cert /path/to/certificateID-certificate.pem.crt \  
  --key /path/to/certificateID-private.pem.key \  
  --cacert /path/to/AmazonRootCA1.pem \  
  https://your-credential-provider-endpoint/role-aliases/your-role-alias-name/  
  credentials
```

詳細については、「[証明書を使用してセキュリティトークンを取得する方法](#)」を参照してください。

Amazon Kinesis Video Streams Edge エージェントを構築する

Amazon Kinesis Video Streams Edge エージェントを構築する

1. 提供されたリンクを使用して tar ファイルをダウンロードします。

Amazon Kinesis Video Streams Edge Agent のインタレストフォームに記入した場合は、ダウンロードリンクの E メールを確認してください。フォームを完了していない場合は、[ここで](#)入力します。

2. チェックサムを確認します。
3. デバイス内のバイナリと jar を抽出します。

型: `tar -xvf kvs-edge-agent.tar.gz`。

抽出後、フォルダ構造は次のようになります。

```
kvs-edge-agent/LICENSE
kvs-edge-agent/THIRD-PARTY-LICENSES
kvs-edge-agent/pom.xml
kvs-edge-agent/KvsEdgeComponent
kvs-edge-agent/KvsEdgeComponent/recipes
kvs-edge-agent/KvsEdgeComponent/recipes/recipe.yaml
kvs-edge-agent/KvsEdgeComponent/artifacts
kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/edge_log_config
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/kvs-edge-agent.jar
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/libgstkvssink.so
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/libIngestorPipelineJNI.so
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/lib
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/lib/libcproducer.so
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/lib/libKinesisVideoProducer.so
```

Note

リリースフォルダ名は、最新のバイナリリリース番号を反映するように設定する必要があります。たとえば、1.0.0 リリースでは、フォルダ名は 1.0.0 に設定されます。

4. 依存関係 jar を構築します。

Note

に含まれている jar には依存関係 `kvs-edge-agent.tar.gz` がありません。これらのライブラリを構築するには、次のステップを使用します。

を含む `kvs-edge-agent` フォルダに移動します `pom.xml`。

「`mvn clean package`」と入力します。

これにより、Amazon Kinesis Video Streams Edge Agent が 必要とする依存関係を含む jar ファイルが生成されます `kvs-edge-agent/target/libs.jar`。

5. コンポーネントのアーティファクトを含むフォルダ `libs.jar` に を配置します。

「`mv ./target/libs.jar ./KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/`」と入力します。

6. 前のステップの値を使用して環境変数を設定します。次の表に、変数の説明を示します。

環境変数名	必要	説明
AWS_REGION	はい	使用されるリージョン。 例: <code>us-west-2</code>
AWS_IOT_CA_CERT	はい	TLS を介してバックエンドサービスとの信頼を確立するために使用される CA 証明書へのファイルパス。 例: <code>/file/path/to/AmazonRootCA1.pem</code>
AWS_IOT_CORE_CERT	はい	X.509 証明書へのファイルパス。 例: <code>/file/path/to/certifica</code>

環境変数名	必要	説明
		<code>teID-cert</code> <code>ificate</code> .pem.crt
AWS_IOT_CORE_CREDENTIAL_ENDPOINT	はい	AWS アカウントの AWS IoT Core 認証情報エンドポイントプロバイダー エンドポイント。 例: <code>credential-account-specific-prefix</code> .credentials.iot. <code>aws-region</code> .amazonaws.com
AWS_IOT_CORE_DATA_ATS_ENDPOINT	はい	アカウントの AWS AWS IoT Core データプレーンエンドポイント。 例: <code>data-account-specific-prefix</code> .iot. <code>aws-region</code> .amazonaws.com
AWS_IOT_CORE_PRIVATE_KEY	はい	パブリック/プライベートキーペアで使用されるプライベートキーへのファイルパス。詳細については、「でのキー管理 AWS IoT」を参照してください。 例: <code>/file/path/to/certificateID-private</code> .pem.key

環境変数名	必要	説明
AWS_IOT_CORE_ROLE_ALIAS	はい	接続時に使用する IAM AWS ロールを指すロールエイリアスの名前 AWS IoT Core。 例: kvs-edge-role-alias
AWS_IOT_CORE_THING_NAME	はい	アプリケーションが実行されている AWS IoT モノの名前。 例: my-edge-device-thing
GST_PLUGIN_PATH	はい	gstkvssink およびIngestorPipelineJNI プラットフォーム依存ライブラリを含むフォルダを指すファイルパス。GStreamer がこれらのプラグインをロードできるようにします。詳細については、「the section called “GStreamer 要素をダウンロード、構築、設定する”」を参照してください。 例: <i>/download-location</i> /kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesis.video.KvsEdgeComponent/ <i>EdgeAgent Version</i> /

環境変数名	必要	説明
LD_LIBRARY_PATH	はい	cproducer およ びKinesisVideoProduc er プラットフォーム依存ラ イブラリを含むディレクトリ を指すファイルパス。 例: <i>/download- location</i> /kvs- edge-agent/Kv sEdgeComponent/art ifacts/aws.kinesis video.KvsEdgeCompo nent/ <i>EdgeAgent Version</i> /lib/
AWS_KVS_EDGE_CLOUD WATCH_ENABLED	いいえ	Amazon Kinesis Video Streams Edge Agent がジョ ブのヘルスマトリクスを投稿 するかどうかを決定します Amazon CloudWatch。 使用できる値: TRUE/FALSE (大文字と小文字は区別され ません)。指定しない場合、 デフォルトFALSEは になり ます。 例: FALSE

環境変数名	必要	説明
AWS_KVS_EDGE_LOG_LEVEL	いいえ	Amazon Kinesis Video Streams Edge Agent 出力のログ記録のレベル。 使用できる値: • VOFF • すべて • 致命的 • エラー • WARN • INFO、指定されていない場合はデフォルト • DEBUG • TRACE 例: INFO
AWS_KVS_EDGE_LOG_MAX_FILE_SIZE	いいえ	ログファイルがこのサイズに達すると、ロールオーバーが発生します。 • 最小: 0 • 最大: 10,000 • デフォルト: 指定されていない場合は 20 • 単位: メガバイト (MB) 例: 5

環境変数名	必要	説明
AWS_KVS_EDGE_LOG_OUTPUT_DIRECTORY	いいえ	Amazon Kinesis Video Streams Edge Agent ログが出力されるディレクトリを指すファイルパス。指定しない場合、デフォルト <code>./log</code> になります。 例: <code>/file/path/</code>
AWS_KVS_EDGE_LOG_ROLLOVER_COUNT	いいえ	削除する前に保持するロールオーバーログの数。 • 最小: 1 • 最大: 100 • デフォルト: 指定されていない場合は 10 例: 20
AWS_KVS_EDGE_RECORDING_DIRECTORY	いいえ	ディレクトリに記録されたメディアを指すファイルパスが書き込まれます。指定しない場合、デフォルトは現在のディレクトリになります。 例: <code>/file/path/</code>
GST_DEBUG	いいえ	出力する GStreamer ログのレベルを指定します。詳細については、GStreamer ドキュメント を参照してください。 例: 0

環境変数名	必要	説明
GST_DEBUG_FILE	いいえ	GStreamer デバッグログの出力ファイルを指定します。設定されていない場合、デバッグログは標準エラーに出力されます。詳細については、GStreamer ドキュメントを参照してください。 例: <code>/tmp/gstreamer-logging.log</code>

7. GStreamer キャッシュをクリアします。タイプ:

```
rm ~/.cache/gstreamer-1.0/registry.your-os-architecture.bin
```

詳細については、[GStreamer レジストリドキュメント](#)を参照してください。

8. Java コマンドを準備して実行します。Amazon Kinesis Video Streams Edge エージェントは、次の引数を受け入れます。

Java プロパティ名	必要	説明
java.library.path	いいえ	gstkvssink とIngestorPipelineJNI 依存ライブラリを含むフォルダを指すファイルパス。指定しない場合、Amazon Kinesis Video Streams Edge Agent は現在のディレクトリでそれらを検索します。

 Important

これらのファイルを見つけることがで

Java プロパティ名

必要

説明

きない場合、Amazon Kinesis Video Streams Edge Agent は正しく機能しません。

例: */file/path/*

これらを設定するには、`jar -Djava-property-name=value` の実行に使用する java コマンドに *を* 追加します。

例えば、次のようになります。

```
java -Djava.library.path=/download-location/kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion \  
--add-opens java.base/jdk.internal.misc=ALL-UNNAMED \  
-Dio.netty.tryReflectionSetAccessible=true \  
-cp kvs-edge-agent.jar:libs.jar \  
com.amazonaws.kinesisvideo.edge.controller.ControllerApp
```

 Important

上記の java コマンドを 同じディレクトリから実行します */download-location/kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion*。

9. を使用して設定をアプリケーションに送信します AWS CLI。

a. 新しいファイル を作成します *example-edge-configuration.json*。

ファイルに次のコードを貼り付けます。これは、毎日午前 9:00:00 から午後 4:59:59 まで (AWS IoT デバイスのシステム時間に応じて) を記録するサンプル設定です。また、記録されたメディアを毎日午後 7:00:00 から午後 9:59:59 にアップロードします。

詳細については、「[the section called “StartEdgeConfigurationUpdate”](#)」を参照してください。

```
{
  "StreamARN": "arn:aws:kinesisvideo:your-region:your-account-id:stream/your-stream/0123456789012",
  "EdgeConfig": {
    "HubDeviceArn": "arn:aws:iot:your-region:your-account-id:thing/kvs-edge-agent-demo",
    "RecorderConfig": {
      "MediaSourceConfig": {
        "MediaUriSecretArn": "arn:aws:secretsmanager:your-region:your-account-id:secret:your-secret-dRbHJQ",
        "MediaUriType": "RTSP_URI"
      },
      "ScheduleConfig": {
        "ScheduleExpression": "0 0 9,10,11,12,13,14,15,16 ? * * *",
        "DurationInSeconds": 3599
      }
    },
    "UploaderConfig": {
      "ScheduleConfig": {
        "ScheduleExpression": "0 0 19,20,21 ? * * *",
        "DurationInSeconds": 3599
      }
    },
    "DeletionConfig": {
      "EdgeRetentionInHours": 15,
      "LocalSizeConfig": {
        "MaxLocalMediaSizeInMB": 2800,
        "StrategyOnFullSize": "DELETE_OLDEST_MEDIA"
      },
      "DeleteAfterUpload": true
    }
  }
}
```

- b. Amazon Kinesis Video Streams Edge エージェントにファイルを送信するには、に次のように入力します AWS CLI。

```
aws kinesishvideo start-edge-configuration-update --cli-input-json
"file://example-edge-configuration.json"
```

10. Amazon Kinesis Video Streams Edge Agent のストリームごとに前のステップを繰り返します。

デバイスに CloudWatch エージェントをインストールする

Note

[CloudWatch のクォータ](#)に注意してください。

Amazon Kinesis Video Streams Edge Agent によって生成されたログを CloudWatch に自動的にアップロードするように CloudWatch エージェントをインストールして設定するには、次の手順に従います。これは任意の手順です。

デバイスに CloudWatch エージェントをインストールする[手順](#)については、Amazon CloudWatch ユーザーガイド」を参照してください。

設定を求められたら、次のいずれかの設定を選択します。

Important

次の設定 `file_path` のは、デフォルトのログ記録出力場所が使用されていることを前提としています。

使用するファイルパスは、の場所から Amazon Kinesis Video Streams Edge エージェントを実行していることを前提としています `download-location/kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent/version`。

- ログをアップロードし、デバイスの RAM および CPU メトリクスをポストするように CloudWatch エージェントを設定するには、設定ファイルに以下を貼り付けます。

```
{
  "agent": {
    "run_as_user": "ubuntu",
    "metrics_collection_interval": 60
```

```

},
"metrics": {
  "metrics_collected": {
    "mem": {
      "measurement": [
        "mem_used_percent"
      ],
      "append_dimensions": {
        "IoTThing": "YourIoTThingName"
      }
    },
    "cpu": {
      "resources": [
        "*"
      ],
      "measurement": [
        "usage_active"
      ],
      "totalcpu": true,
      "append_dimensions": {
        "IoTThing": "YourIoTThingName"
      }
    }
  }
},
"logs": {
  "logs_collected": {
    "files": {
      "collect_list": [
        {
          "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/java_kvs.log",
          "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
          "log_stream_name": "YourIoTThingName-java_kvs.log"
        },
        {
          "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/cpp_kvs_edge.log*",
          "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
          "log_stream_name": "YourIoTThingName-cpp_kvs_edge.log"
        },
        {
          "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/cpp_kvs_streams.log*",

```

```

        "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
        "log_stream_name": "YourIotThingName-cpp_kvs_streams.log"
    },
    {
        "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/cpp_kvssink.log*",
        "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
        "log_stream_name": "YourIotThingName-cpp_kvssink.log"
    }
]
}
}
}
}
}

```

- ログのみをアップロードし、デバイスの RAM と CPU を収集しない場合は、次の設定を使用します。

```

{
  "logs": {
    "logs_collected": {
      "files": {
        "collect_list": [
          {
            "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/java_kvs.log",
            "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
            "log_stream_name": "YourIotThingName-java_kvs.log"
          },
          {
            "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/cpp_kvs_edge.log*",
            "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
            "log_stream_name": "YourIotThingName-cpp_kvs_edge.log"
          },
          {
            "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/cpp_kvs_streams.log*",
            "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
            "log_stream_name": "YourIotThingName-cpp_kvs_streams.log"
          }
        ]
      }
    }
  }
}

```

```

        "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/cpp_kvssink.log*",
        "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
        "log_stream_name": "YourIotThingName-cpp_kvssink.log"
    }
}
}
}
}
}
}

```

Amazon Kinesis Video Streams Edge エージェントをネイティブプロセスとして実行する

Amazon Kinesis Video Streams Edge Agent を systemd サービスとしてセットアップします。これは任意の手順です。

systemd は Linux デバイスのシステムおよびサービスマネージャーです。systemd は、プロセスを管理するために推奨される方法です。アプリケーションにエラーが発生したり、アプリケーションを実行しているデバイスが電源を失ったりした場合に、Amazon Kinesis Video Streams Edge Agent を再起動するためです。

以下の操作を実行します。

Amazon Kinesis Video Streams Edge エージェントをネイティブプロセスとして実行する

1. で新しいファイルを作成し/etc/systemd/system、 という名前を付けます *aws.kinesisvideo.edge-runtime-agent.service*。

以下を貼り付けます。

```

[Unit]
Description=AWS Kinesis Video Streams edge agent
After=network.target
StartLimitBurst=3
StartLimitInterval=30

[Service]
Type=simple
Restart=on-failure
RestartSec=10

```

```
WorkingDirectory=/download-location/kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion
Environment="GST_PLUGIN_PATH=/download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion"
Environment="LD_LIBRARY_PATH=/download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/lib"
...
Environment="AWS_IOT_CORE_DATA_ATS_ENDPOINT=data-account-specific-prefix.iot.aws-
region.amazonaws.com"
ExecStart=/usr/lib/jvm/java-11-amazon-corretto/bin/java --add-opens java.base/
jdk.internal.misc=ALL-UNNAMED -Dio.netty.tryReflectionSetAccessible=true -cp kvs-
edge-agent.jar:libs.jar com.amazonaws.kinesisvideo.edge.controller.ControllerApp

[Install]
WantedBy=multi-user.target
```

systemd サービス設定ファイルで受け入れられるパラメータの詳細については、[ドキュメント](#)を参照してください。

Note

「」で指定されているように、必要な環境変数を ... の場所に追加します [the section called “Edge エージェントの構築”](#)。

2. サービスファイルを再ロードして、新しいサービスを含めます。

「sudo systemctl daemon-reload」と入力します。

3. サービスを起動します。

「sudo systemctl start *aws.kinesisvideo.edge-runtime-agent.service*」と入力します。

4. Amazon Kinesis Video Streams Edge Agent サービスのステータスをチェックして、実行中であることを確認します。

「sudo systemctl status *aws.kinesisvideo.edge-runtime-agent.service*」と入力します。

以下は、表示される出力の例です。

```
aws.kinesisvideo.edge-runtime-agent.service - AWS Kinesis Video Streams edge agent
```



```
Loaded: loaded (/etc/systemd/system/aws.kinesisvideo.edge-runtime-agent.service; disabled; vendor preset: enabled)
Active: active (running) since Thu 2023-06-08 19:15:02 UTC; 6s ago
Main PID: 506483 (java)
Tasks: 23 (limit: 9518)
Memory: 77.5M
CPU: 4.214s
CGroup: /system.slice/aws.kinesisvideo.edge-runtime-agent.service
        ##506483 /usr/lib/jvm/java-11-amazon-corretto/bin/java -cp kvs-edge-agent.jar:libs.jar com.amazonaws.kinesisvideo.edge.controller.ControllerApp
```

5. エラーがないかログを検査します。

「`journalctl -e -u aws.kinesisvideo.edge-runtime-agent.service`」と入力します。

6. を使用してプロセスを管理するオプションの完全なリスト `systemctl --help` については、「」と入力します `systemctl`。

以下は、Amazon Kinesis Video Streams Edge エージェントを管理するための一般的なコマンドです。

- 再起動するには、「」と入力します `sudo systemctl restart aws.kinesisvideo.edge-runtime-agent.service`。
- 停止するには、「」と入力します `sudo systemctl stop aws.kinesisvideo.edge-runtime-agent.service`。
- デバイスを再起動するたびに自動的に起動するには、「」と入力します `sudo systemctl enable aws.kinesisvideo.edge-runtime-agent.service`。

Amazon Kinesis Video Streams Edge エージェントを にデプロイする AWS IoT Greengrass

このセクションでは、Amazon Kinesis Video Streams を で使用するための包括的なガイドを提供します AWS IoT Greengrass。これらのサービスを組み合わせることで、エッジデバイスからクラウドにビデオを効率的にストリーミングし、IoT、監視などの幅広いアプリケーションを実現できます。

詳細については、以下を参照してください。

- 開発環境のセットアップ

- Kinesis ビデオストリームの作成
- Lambda 関数の作成とパッケージ化
- Kinesis Video Streams コアデバイスの設定
- コアデバイスにデプロイする
- ストリームの検証

Amazon Kinesis Video Streams Edge Agent を にデプロイ AWS IoT Greengrass して、IP カメラからメディアを記録およびアップロードするには、次の手順に従います。

トピック

- [Ubuntu Amazon EC2 インスタンスを作成する](#)
- [デバイスで AWS IoT Greengrass V2 コアデバイスをセットアップする](#)
- [IP カメラ RTSP URLs の Amazon Kinesis Video Streams と AWS Secrets Manager リソースを作成する](#)
- [トークン交換サービス \(TES\) ロールにアクセス許可を追加する](#)
- [デバイスに AWS IoT Greengrass Secret Manager コンポーネントをインストールする](#)
- [Amazon Kinesis Video Streams Edge Agent AWS IoT Greengrass コンポーネントをデバイスにデプロイする](#)
- [デバイスに AWS IoT Greengrass ログマネージャーコンポーネントをインストールする](#)

Ubuntu Amazon EC2 インスタンスを作成する

Ubuntu Amazon EC2 インスタンスを作成するには、次の手順を実行します。

Ubuntu Amazon EC2 インスタンスを作成する

1. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/ec2/> で Amazon EC2 コンソールを開きます。

適切なリージョンが選択されていることを確認します。

2. [インスタンスの作成] を選択してください。

以下のフィールドに値を入力します。

- 名前 – インスタンスの名前を入力します。

- アプリケーションイメージと OS イメージ (Amazon マシンイメージ) – Ubuntu を選択します。
 - インスタンスタイプ – t2.large を選択します。
 - キーペアログイン – 独自のキーペアを作成します。
 - ネットワーク設定 – デフォルトのままにします。
 - ストレージの設定 – ボリュームを 256 GiB に増やします。
 - 詳細設定 – デフォルトのままにします。
3. インスタンスを起動し、そのインスタンスに SSH を挿入します。

以下の操作を実行します。

1. 左側のナビゲーションでインスタンスを選択し、インスタンス ID を選択します。
 2. 右上で接続を選択します。
 3. SSH クライアントを選択し、画面の指示に従います。
 4. ターミナルを開き、ダウンロードした .pem ファイル (にある可能性が高い) に移動します~/Downloads。
 5. これらの手順を初めて実行すると、「ホストの信頼性 (...) を確立できません」というメッセージが表示されます。「はい」と入力します。
4. システムライブラリをインストールして、Amazon Kinesis Video Streams Edge エージェントをインスタンスに構築します。

```
wget -O- https://apt.corretto.aws/corretto.key | sudo apt-key add -  
sudo add-apt-repository 'deb https://apt.corretto.aws stable main'
```

```
sudo apt-get update
```

```
sudo apt-get install -y gcc libssl-dev libcurl4-openssl-dev liblog4cplus-dev \  
libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev \  
gstreamer1.0-plugins-base-apps gstreamer1.0-plugins-bad \  
gstreamer1.0-plugins-good gstreamer1.0-tools \  
unzip java-11-amazon-corretto-jdk maven
```

Important

一部のサービスを再起動する必要があることを示す画面が表示された場合は、Enter キーを押して OK を選択します。

詳細については、「[Amazon Corretto 11 ユーザーガイド](#)」を参照してください。

デバイスで AWS IoT Greengrass V2 コアデバイスをセットアップする

Amazon EC2 インスタンスに AWS IoT Greengrass コア nucleus ソフトウェアをインストールするには、次の手順に従います。

AWS IoT Greengrass コアデバイスをセットアップする

1. <https://console.aws.amazon.com/iot/> AWS マネジメントコンソールにサインインします。

適切なリージョンが選択されていることを確認します。

2. 左側のナビゲーションで、Greengrass デバイス、Core デバイスを選択します。
3. 1 つのコアデバイスをセットアップを選択します。
4. 画面上のステップを完了します。

- ステップ 1: Greengrass コアデバイスを登録する。デバイスの名前を入力します。
- ステップ 2: をモノのグループに追加して、継続的なデプロイを適用します。グループなしを選択します。
- ステップ 3: Greengrass Core ソフトウェアをインストールします。Linux を選択します。
 - ステップ 3.1: デバイスに Java をインストールする

Java は の一部としてインストールされます [the section called “Ubuntu インスタンスを作成する”](#)。Java がまだインストールされていない場合は、このステップに戻ります。

- ステップ 3.2: デバイスに AWS 認証情報をコピーする

bash/zsh オプションを開き、Amazon EC2 インスタンスにエクスポートコマンドを貼り付けます。

- ステップ 3.3: インストーラを実行する

1. インストーラをダウンロードして実行し、Ubuntu Amazon EC2 インスタンスでインストーラコマンドを実行します。

 Note

Run the installer コマンドは、前のステップで選択した名前に基づいて自動的に更新されます。

2. 作成されたトークン交換サービス (TES) ロールを書き留めます。これは後で必要になります。

 Note

デフォルトでは、作成されたロールは GreengrassV2TokenExchangeRole と呼ばれます。

IP カメラ RTSP URLs の Amazon Kinesis Video Streams と AWS Secrets Manager リソースを作成する

以下の手順に従って、に必要なストリームとシークレットを作成します AWS Secrets Manager。最初にこのステップを実行します。これは、ポリシーで作成されたリソースの ARNs が必要なためです。

Amazon Kinesis Video Streams を作成する

AWS マネジメントコンソール AWS CLI、または API を使用して Amazon Kinesis Video Streams を作成します。

で AWS マネジメントコンソール、[Amazon Kinesis Video Streams コンソール](#)を開きます。左側のナビゲーションでビデオストリームを選択します。

詳細については、「[the section called “Amazon Kinesis ビデオストリームを作成する”](#)」を参照してください。

でシークレットを作成する AWS Secrets Manager

で AWS マネジメントコンソール、[AWS Secrets Manager コンソール](#)を開きます。左側のナビゲーションでシークレットを選択します。

適切なリージョンが選択されていることを確認します。

1. 新しいシークレットを保存 を選択します。

a. ステップ 1: シークレットタイプを選択する

- [その他のシークレットのタイプ] を選択します。
- 「キーと値のペア」セクションで、キーと値のペアを追加します。

[Key] (キー): MediaURI

Note

キーは である必要がありますMediaURI。これは大文字と小文字が区別されます。誤って入力すると、アプリケーションは機能しません。

値: *Your MediaURI*。

Example

例: rtsp://<YourCameraIPAddress>:<YourCameraRTSPPort>/
YourCameraMediaURI。

- b. ステップ 2: シークレットを設定する。このシークレットに名前を付けます。任意の名前を付けます。
 - c. ステップ 3: ローテーションを設定する - オプション。[次へ] を選択します。
 - d. ステップ 4: 確認する。[保存する] を選択します。
- ## 2. シークレットがすぐに表示されない場合は、更新ボタンを選択します。

シークレットの名前を選択します。シークレット ARN を書き留めます。

3. ストリーミング元の MediaURI ごとにこのプロセスを繰り返します。

Note

AWS ネットワークは、一部のパブリック RTSP ソースをブロックします。Amazon EC2 インスタンス内から、または VPN への接続中にアンマネージドで実行している場合、これらにアクセスすることはできません。

⚠ Important

カメラ RTSP URL は h.264 形式でビデオをストリーミングする必要があります。フラグメントの期間は、「」で説明されている制限を超えることはできません [the section called “プロデューサー SDK クォータ”](#)。

Amazon Kinesis Video Streams Edge Agent はビデオのみをサポートします。

を実行して `gst-discoverer-1.0 Your RtspUrl`、カメラがデバイスから到達可能であることを確認します。

作成したすべてのストリームとシークレットの ARNs を保存します。次のステップでは、これらが必要です。

トークン交換サービス (TES) ロールにアクセス許可を追加する

シークレットを確認するアクセス許可を引き受けるデバイスにトークン交換サービス (TES) ロールを付与します。これは、コンポーネントが正しく動作するために必要です AWS Secrets Manager AWS IoT Greengrass。

TES ロールにアクセス許可を追加する

1. にサインイン AWS マネジメントコンソール し、 <https://console.aws.amazon.com/iam/> で IAM コンソールを開きます。
2. 左側のナビゲーションでロールを選択し、プロセスの前半で作成した TES ロールを検索します。
3. アクセス許可の追加ドロップダウンで、ポリシーのアタッチを選択します。
4. [Create policy] (ポリシーの作成) を選択します。
5. 下にスクロールして編集を選択します。
6. ポリシーエディタで JSON を選択し、ポリシーを編集します。

ポリシーを以下に置き換えます。

 Note

arn:aws:kinesisvideo:*:*:stream/streamName1/* と を、前のステップで作成したストリームの ARNs arn:aws:kinesisvideo:*:*:stream/streamName2/* に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:ListStreams"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:DescribeStream",
        "kinesisvideo:PutMedia",
        "kinesisvideo:TagStream",
        "kinesisvideo:GetDataEndpoint"
      ],
      "Resource": [
        "arn:aws:kinesisvideo:*:*:stream/streamName1/*",
        "arn:aws:kinesisvideo:*:*:stream/streamName2/*"
      ]
    }
  ]
}
```

7. [タグの追加] ページで、[次へ: レビュー] を選択します。
8. ポリシーに名前を付け、ポリシーの作成を選択します。

ポリシー名の例としては、KvsEdgeAccessPolicy があります。

9. タブを閉じて、ポリシーを TES ロールにアタッチしていたタブに戻ります。

更新ボタンを選択し、新しく作成したポリシーを検索します。

チェックボックスをオンにし、ポリシーのアタッチを選択します。

次の画面に、ポリシーがロールに正常にアタッチされたことを示すメモが表示されます。

10. 今度はシークレット用に別のポリシーを作成してアタッチします。

ポリシーを以下に置き換えます。

 Note

を、で作成した MediaURI シークレットを含む ARNs
arn:aws:secretsmanager:*:*:secret:*に置き換えます [the section called “IP カメラ RTSP URLs のリソースを作成する”](#)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "secretsmanager:GetSecretValue",
      "Resource": [
        "arn:aws:secretsmanager:*:*:secret:*",
        "arn:aws:secretsmanager:*:*:secret:*"
      ]
    }
  ]
}
```

11. メトリクス用に Amazon CloudWatch 別のポリシーを作成してアタッチします。ポリシーを以下に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
```

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:PutMetricData"
    ],
    "Resource": [
      "*"
    ]
  }
]
```

デバイスに AWS IoT Greengrass Secret Manager コンポーネントをインストールする

Amazon Kinesis Video Streams Edge Agent では、最初に AWS IoT Greengrass Secret Manager コンポーネントをデバイスにインストールする必要があります。

Secret Manager コンポーネントをインストールする

1. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/iot/> で AWS IoT Core コンソールを開きます。適切なリージョンが選択されていることを確認します。
2. 左側のナビゲーションで、Greengrass デバイス、デプロイを選択します。

で作成したものと同一ターゲットを持つデプロイを選択します [the section called “ AWS IoT Greengrass コアデバイスをセットアップする”](#)。

3. 右上隅の Actions ドロップダウンで、Revise を選択します。

表示されるポップアップで、デプロイの修正を選択します。

4. 以下のセクションを完了します。

- ステップ 1: ターゲットを指定します。[次へ] を選択します。
- ステップ 2: コンポーネントを選択します。
 - aws.greengrass.Cli コンポーネントが選択されていることを確認します。このコンポーネントはアンインストールしないでください。
 - 選択したコンポーネントのみを表示の切り替えを切り替え、aws.greengrass.SecretManager を検索します。

- aws.greengrass.SecretManager の横にあるチェックボックスをオンにし、次へを選択します。
- ステップ 3: コンポーネントを設定する。AWS IoT Greengrass 環境内から AWS IoT Greengrass シークレットをダウンロードするように Secret Manager コンポーネントを設定します。

aws.greengrass.SecretManager コンポーネントを選択し、コンポーネントの設定を選択します。

表示される画面で、Configuration の AWS Secrets Manager ARNs を更新してマージします。

 Note

を、ARNs `arn:aws:secretsmanager:*:*:secret:*` に置き換えます [the section called “IP カメラ RTSP URLs のリソースを作成する”](#)。

```
{
  "cloudSecrets": [
    {
      "arn": "arn:aws:secretsmanager:*:*:secret:*"
    },
    {
      "arn": "arn:aws:secretsmanager:*:*:secret:*"
    }
  ]
}
```

 Note

cloudSecrets は、キーを持つオブジェクトのリストですarn。詳細については、「AWS IoT Greengrass Version 2 デベロッパーガイド」の「[シークレットマネージャー設定](#)」セクションを参照してください。

完了したら、確認を選択し、次へを選択します。

- ステップ 4: 詳細設定を構成する。[次へ] を選択してください。

- ステップ 5: 確認する。[Deploy] (デプロイ) を選択します。
5. AWS Secrets Manager コンポーネントとアクセス許可が正しくインストールされていることを確認します。

Ubuntu Amazon EC2 インスタンスで、と入力 `sudo /greengrass/v2/bin/greengrass-cli component details --name aws.greengrass.SecretManager` して、コンポーネントが更新された設定を受信したことを確認します。

6. AWS IoT Greengrass コアログを検査します。

「`sudo less /greengrass/v2/logs/greengrass.log`」と入力します。

デプロイエラーを確認します。

エラーが発生した場合は、デプロイを修正して `aws.greengrass.SecretManager` コンポーネントを削除します。

と入力 `sudo service greengrass restart` して AWS IoT Greengrass コアサービスを再起動します。

デプロイエラーがアクセス許可の欠落に関連している場合は、[the section called “TES ロールにアクセス許可を追加する”](#) セクションを確認して、TES ロールに適切なアクセス許可があることを確認します。次に、このセクションを繰り返します。

7. AWS IoT Greengrass Secret Manager コンポーネントのシークレットを更新する

 Important

AWS IoT Greengrass Secret Manager コンポーネントは、デプロイが更新された場合にのみシークレットを取得してキャッシュします。

AWS IoT Greengrass Secret Manager コンポーネントのシークレットを更新するには、前のステップ 1~6 に従い、次の変更を加えます。

ステップ 3: コンポーネントを設定する。AWS IoT Greengrass 環境内から AWS IoT Greengrass シークレットをダウンロードするように Secret Manager コンポーネントを設定します。

`aws.greengrass.SecretManager` コンポーネントを選択し、コンポーネントの設定を選択します。

表示される画面で、パスの[""]リセットボックスに貼り付け、Configuration to merge ボックスの AWS Secrets Manager ARNs を更新します。

詳細については、[「更新のリセット」](#)を参照してください。

Amazon Kinesis Video Streams Edge Agent AWS IoT Greengrass コンポーネントをデバイスにデプロイする

Amazon Kinesis Video Streams Edge Agent AWS IoT Greengrass コンポーネントをデバイスにデプロイするには、以下を実行します。

コンポーネントのデプロイ

1. 提供されたリンクを使用して tar ファイルをダウンロードします。

Amazon Kinesis Video Streams Edge Agent のインタレストフォームに記入した場合は、ダウンロードリンクの E メールを確認してください。フォームを完了していない場合は、[ここで](#)入力します。

2. チェックサムを確認します。
3. デバイス内のバイナリと jar を抽出します。

型: `tar -xvf kvs-edge-agent.tar.gz`。

抽出後、フォルダ構造は次のようになります。

```
kvs-edge-agent/LICENSE
kvs-edge-agent/THIRD-PARTY-LICENSES
kvs-edge-agent/pom.xml
kvs-edge-agent/KvsEdgeComponent
kvs-edge-agent/KvsEdgeComponent/recipes
kvs-edge-agent/KvsEdgeComponent/recipes/recipe.yaml
kvs-edge-agent/KvsEdgeComponent/artifacts
kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/edge_log_config
```

```
kvs-edge-agent/KvsEdgeComponent/artifacts/  
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/kvs-edge-agent.jar  
kvs-edge-agent/KvsEdgeComponent/artifacts/  
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/libgstkvssink.so  
kvs-edge-agent/KvsEdgeComponent/artifacts/  
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/libIngestorPipelineJNI.so  
kvs-edge-agent/KvsEdgeComponent/artifacts/  
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/lib  
kvs-edge-agent/KvsEdgeComponent/artifacts/  
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/lib/libcproducer.so  
kvs-edge-agent/KvsEdgeComponent/artifacts/  
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/lib/libKinesisVideoProducer.so
```

 Note

リリースフォルダ名は、最新のバイナリリリース番号を反映するように設定する必要があります。たとえば、1.0.0 リリースでは、フォルダ名は 1.0.0 に設定されます。

4. 依存関係 jar を構築します。

 Note

kvs-edge-agent.tar.gz に含まれている jar には依存関係がありません。これらのライブラリを構築するには、次のステップを使用します。

を含むkvs-edge-agentフォルダに移動しますpom.xml。

「mvn clean package」と入力します。

これにより、Amazon Kinesis Video Streams Edge Agent が 必要とする依存関係を含む jar ファイルが生成されますkvs-edge-agent/target/libs.jar。

5. libs.jar をコンポーネントのアーティファクトを含むフォルダに配置します。

「mv ./target/libs.jar ./KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/*EdgeAgentVersion*/」と入力します。

6. オプション。プロパティを設定します。Amazon Kinesis Video Streams Edge エージェントは、モードで次の環境変数を受け入れます AWS IoT Greengrass 。

環境変数名	必要	説明
AWS_REGION	はい	使用されるリージョン。 例: us-west-2 AWS IoT Greengrass Core ソフトウェアは、この値を自動的に設定します。詳細については、「AWS IoT Greengrass Version 2 デベロッパーガイド」の「コンポーネント環境変数リファレンス」トピックを参照してください。

環境変数名	必要	説明
GST_PLUGIN_PATH	はい	gstkvssink およ びIngestorPipelineJN I プラットフォーム依存ラ イブラリを含むフォルダを指 すファイルパス。これによ り、GStreamer はこれらの プラグインをロードできま す。詳細については、「the section called “GStreamer 要 素をダウンロード、構築、設 定する”」を参照してくださ い。 例: <i>/download- location /kvs- edge-agent/Kv sEdgeComponent/art ifacts/aws.kinesis video.KvsEdgeCompo nent/ EdgeAgent Version /</i>

環境変数名	必要	説明
LD_LIBRARY_PATH	はい	cproducer およ びKinesisVideoProduc er プラットフォーム依存ラ イブラリを含むディレクトリ を指すファイルパス。 例: <i>/download- location</i> /kvs- edge-agent/Kv sEdgeComponent/art ifacts/aws.kinesis video.KvsEdgeCompo nent/ <i>EdgeAgent Version</i> /lib/
AWS_KVS_EDGE_CLOUD WATCH_ENABLED	いいえ	Amazon Kinesis Video Streams Edge Agent がジョ ブのヘルスマトリクスを投稿 するかどうかを決定します Amazon CloudWatch。 使用できる値: TRUE/FALSE (大文字と小文字は区別され ません)。指定しない場合、 デフォルトFALSEは になり ます。 例: FALSE

環境変数名	必要	説明
AWS_KVS_EDGE_LOG_LEVEL	いいえ	Amazon Kinesis Video Streams Edge Agent 出力のログ記録のレベル。 使用できる値: • VOFF • すべて • 致命的 • エラー • WARN • INFO、指定されていない場合はデフォルト • DEBUG • TRACE 例: INFO
AWS_KVS_EDGE_LOG_MAX_FILE_SIZE	いいえ	ログファイルがこのサイズに達すると、ロールオーバーが発生します。 • 最小: 1 • 最大: 100 • デフォルト: 指定されていない場合は 20 • 単位: メガバイト (MB) 例: 5

環境変数名	必要	説明
AWS_KVS_EDGE_LOG_OUTPUT_DIRECTORY	いいえ	Amazon Kinesis Video Streams Edge Agent ログが出力されるディレクトリを指すファイルパス。指定しない場合、デフォルト <code>./log</code> になります。 例: <code>/file/path/</code>
AWS_KVS_EDGE_LOG_ROLLOVER_COUNT	いいえ	削除する前に保持するロールオーバーログの数。 • 最小: 1 • 最大: 100 • デフォルト: 指定されていない場合は 10 例: 20
AWS_KVS_EDGE_RECORDING_DIRECTORY	いいえ	ディレクトリに記録されたメディアを指すファイルパスが書き込まれます。指定しない場合、デフォルトは現在のディレクトリになります。 例: <code>/file/path/</code>
GREENGRASS_ROOT_DIRECTORY	いいえ	AWS IoT Greengrass ルートディレクトリへのファイルパス。 指定 <code>/greengrass/v2/</code> しない場合、デフォルトでになります。 例: <code>/file/path/</code>

環境変数名	必要	説明
GST_DEBUG	いいえ	出力する GStreamer ログのレベルを指定します。詳細については、 GStreamer ドキュメント を参照してください。 例: 0
GST_DEBUG_FILE	いいえ	GStreamer デバッグログの出力ファイルを指定します。設定されていない場合、デバッグログは標準エラーに出力されます。詳細については、 GStreamer ドキュメント を参照してください。 例: <code>/tmp/gstreamer-logging.log</code>

実行スクリプトを開いて `kvs-edge-agent/KvsEdgeComponent/recipes/recipe.yaml` を変更し、前述の環境変数を追加します。

Important

変更された実行スクリプトにタブ文字が含まれていないことを確認します。AWS IoT Greengrass コアソフトウェアはレシピを読み取ることができません。

7. Amazon Kinesis Video Streams Edge Agent AWS IoT Greengrass コンポーネントをデプロイします。

タイプ:

```
sudo /greengrass/v2/bin/greengrass-cli deployment create \
  --recipeDir <download location>/kvs-edge-agent/KvsEdgeComponent/recipes/ \
  --artifactDir <download location>/kvs-edge-agent/KvsEdgeComponent/artifacts/ \
  --merge "aws.kinesisvideo.KvsEdgeComponent=EdgeAgentVersion"
```

詳細については、AWS IoT Greengrass Version 2 デベロッパーガイドの以下のセクションを参照してください。

- [AWS IoT Greengrass CLI コマンド](#)
- [AWS IoT Greengrass コンポーネントをデバイスにデプロイする](#)

8. を使用して設定をアプリケーションに送信します AWS CLI。

- a. 新しいファイル を作成します `example-edge-configuration.json`。

ファイルに次のコードを貼り付けます。これは、毎日午前 9:00:00 から午後 4:59:59 まで (AWS IoT デバイスのシステム時間に応じて) を記録するサンプル設定です。また、記録されたメディアを毎日午後 7:00:00 から午後 9:59:59 にアップロードします。

詳細については、「[the section called “StartEdgeConfigurationUpdate”](#)」を参照してください。

```
{
  "StreamARN": "arn:aws:kinesisvideo:your-region:your-account-id:stream/your-stream/0123456789012",
  "EdgeConfig": {
    "HubDeviceArn": "arn:aws:iot:your-region:your-account-id:thing/kvs-edge-agent-demo",
    "RecorderConfig": {
      "MediaSourceConfig": {
        "MediaUriSecretArn": "arn:aws:secretsmanager:your-region:your-account-id:secret:your-secret-dRbHJQ",
        "MediaUriType": "RTSP_URI"
      },
      "ScheduleConfig": {
        "ScheduleExpression": "0 0 9,10,11,12,13,14,15,16 ? * * **",
        "DurationInSeconds": 3599
      }
    },
    "UploaderConfig": {
      "ScheduleConfig": {
        "ScheduleExpression": "0 0 19,20,21 ? * * **",
        "DurationInSeconds": 3599
      }
    },
    "DeletionConfig": {
      "EdgeRetentionInHours": 15,
```

```
        "LocalSizeConfig": {
            "MaxLocalMediaSizeInMB": 2800,
            "StrategyOnFullSize": "DELETE_OLDEST_MEDIA"
        },
        "DeleteAfterUpload": true
    }
}
```

- b. に次のように入力 AWS CLI して、ファイルを Amazon Kinesis Video Streams Edge エージェントに送信します。

```
aws kinesishvideo start-edge-configuration-update --cli-input-json
"file://example-edge-configuration.json"
```

9. Amazon Kinesis Video Streams Edge Agent のストリームごとに前のステップを繰り返します。

デバイスに AWS IoT Greengrass ログマネージャーコンポーネントをインストールする

Note

[CloudWatch クォータ](#)に注意してください。

ログマネージャーコンポーネントを使用して CloudWatch に自動的にアップロードするように Amazon Kinesis Video Streams Edge Agent AWS IoT Greengrass ログを設定するには、次の手順に従います。これは任意の手順です。

AWS IoT Greengrass ログマネージャーコンポーネントをインストールする

1. AWS IoT Greengrass デバイスロールに[適切なアクセス許可](#)があることを確認します。
 - a. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/iam/> で IAM コンソールを開きます。
 - b. 左側のナビゲーションで **ロール** をクリックします。
 - c. で作成された TES ロールの名前を選択します[the section called “ AWS IoT Greengrass コア デバイスをセットアップする”](#)。必要に応じて検索バーを使用します。
 - d. `GreengrassV2TokenExchangeRoleAccess[]` ポリシーを選択します。

- e. JSON タブを選択し、ポリシーが次のようになっていることを確認します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

- f. GreengrassV2TokenExchangeRoleAccess ポリシーが存在しない場合、または必要なアクセス許可が不足している場合は、これらのアクセス許可を持つ新しい IAM ポリシーを作成し、で作成された TES ロールにアタッチします [the section called “ AWS IoT Greengrass コアデバイスをセットアップする”](#)。
2. にサインイン AWS マネジメントコンソール し、 <https://console.aws.amazon.com/iot/> で AWS IoT Core コンソールを開きます。適切なリージョンが選択されていることを確認します。
3. 左側のナビゲーションで、Greengrass デバイス、デプロイを選択します。

で作成したものと同一ターゲットを持つデプロイを選択します [the section called “ AWS IoT Greengrass コアデバイスをセットアップする”](#)。
4. 右上隅で、アクションを選択し、修正を選択します。

表示されるポップアップで、デプロイの修正を選択します。
5. 以下のセクションを完了します。
 - a. ステップ 1: ターゲットを指定します。[次へ] を選択します。
 - b. ステップ 2: コンポーネントを選択します。

- i. aws.greengrass.Cli コンポーネントと aws.greengrass.SecretManager コンポーネントがまだ選択されていることを確認します。

⚠ Important

これらのコンポーネントはアンインストールしないでください。

- ii. 選択したコンポーネントのみを表示の切り替えを切り替え、aws.greengrass.LogManager を検索します。
 - iii. aws.greengrass.LogManager の横にあるボックスを選択し、次へを選択します。
- c. ステップ 3: コンポーネントを設定する。Amazon Kinesis Video Streams Edge Agent によって生成されたログをアップロードするようにログ AWS IoT Greengrass マネージャーコンポーネントを設定します。

aws.greengrass.LogManager コンポーネントを選択し、コンポーネントの設定を選択します。

表示される画面で、次のログマネージャー設定をマージする設定ボックスに貼り付けます。

```
{
  "logsUploaderConfiguration": {
    "componentLogsConfigurationMap": {
      "aws.kinesisvideo.KvsEdgeComponent/java_kvs.log": {
        "diskSpaceLimit": "100",
        "diskSpaceLimitUnit": "MB",
        "logFileDirectoryPath": "/greengrass/v2/work/
aws.kinesisvideo.KvsEdgeComponent/log",
        "logFileRegex": "java_kvs.log\\w*"
      },
      "aws.kinesisvideo.KvsEdgeComponent/cpp_kvs_edge.log": {
        "diskSpaceLimit": "100",
        "diskSpaceLimitUnit": "MB",
        "logFileDirectoryPath": "/greengrass/v2/work/
aws.kinesisvideo.KvsEdgeComponent/log",
        "logFileRegex": "cpp_kvs_edge.log\\w*"
      },
      "aws.kinesisvideo.KvsEdgeComponent/cpp_kvssink.log": {
        "diskSpaceLimit": "100",
        "diskSpaceLimitUnit": "MB",
```



```

        "logFileDirectoryPath": "/greengrass/v2/work/
aws.kinesisvideo.KvsEdgeComponent/log",
        "logFileRegex": "cpp_kvssink.log\\w*"
    },
    "aws.kinesisvideo.KvsEdgeComponent/cpp_kvs_streams.log": {
        "diskSpaceLimit": "100",
        "diskSpaceLimitUnit": "MB",
        "logFileDirectoryPath": "/greengrass/v2/work/
aws.kinesisvideo.KvsEdgeComponent/log",
        "logFileRegex": "cpp_kvs_streams.log\\w*"
    }
}
},
"periodicUploadIntervalSec": "1"
}

```

Important

前述の設定logFileDirectoryPathのは、デフォルトのログ記録出力場所が使用されていることを前提としています。

Note

ログマネージャー設定の各パラメータの詳細については、「AWS IoT Greengrass Version 2 デベロッパーガイド」の「[ログマネージャー](#)」セクションを参照してください。

完了したら、確認を選択し、次へを選択します。

- d. ステップ 4: 詳細設定を構成する。[次へ] を選択してください。
- e. ステップ 5: 確認する。[Deploy] (デプロイ) を選択します。
6. AWS ログマネージャーコンポーネントとアクセス許可が正しくインストールされていることを確認します。
7. Ubuntu Amazon EC2 インスタンスで、と入力 `sudo /greengrass/v2/bin/greengrass-cli component details --name aws.greengrass.LogManager` して、コンポーネントが更新された設定を受信したことを確認します。
8. AWS IoT Greengrass コアログを検査します。

「`sudo less /greengrass/v2/logs/greengrass.log`」と入力します。

デプロイエラーを確認します。

エラーが発生した場合は、デプロイを修正して`aws.greengrass.LogManager`コンポーネントを削除します。

と入力`sudo service greengrass restart`して AWS IoT Greengrass コアサービスを再起動します。

デプロイエラーがアクセス許可の欠落に関連している場合は、[the section called “TES ロールにアクセス許可を追加する”](#)を確認して TES ロールに適切なアクセス許可があることを確認します。次に、このセクションを繰り返します。

Amazon Kinesis Video Streams Edge Agent に関するよくある質問

以下は、Amazon Kinesis Video Streams Edge Agent サービスに関する一般的な質問です。

Amazon Kinesis Video Streams Edge Agent はどのオペレーティングシステムをサポートしていますか？

Amazon Kinesis Video Streams Edge Agent は現在、次のオペレーティングシステムをサポートしています。

Ubuntu

- 22.x
 - AMD64
- 18.x
 - ARM

AL2

- amzn2
 - AMD64 amazonlinux:2.0.20210219.0-amd64 (Snowball)

Amazon Kinesis Video Streams Edge Agent は H.265 メディアをサポートしていますか？

Amazon Kinesis Video Streams Edge Agent は H.264 基本ストリームのみをサポートします。

Amazon Kinesis Video Streams Edge Agent は AL2 で動作しますか？

はい。

AWS IoT モノまたはデバイス内で複数のストリームを実行するにはどうすればよいですか？

別の [the section called “StartEdgeConfigurationUpdate”](#) を同じ に HubDeviceArn、異なる Amazon Kinesis Video Streams/AWS Secrets Manager ARNs。

送信 **StartEdgeConfigurationUpdate** 後に を編集するにはどうすればよいですか？

[the section called “StartEdgeConfigurationUpdate”](#) 同じ Amazon Kinesis Video Streams ARN HubDeviceArn を使用して、更新された を同じ に送信します。Amazon Kinesis Video Streams アプリケーションは、Amazon Kinesis Video Streams からメッセージを受信すると、そのストリームの以前の設定を上書きします。その後、変更が行われます。

一般的な の例はありますか **ScheduleConfigs**？

Amazon Kinesis Video Streams Edge Agent は、実行中のデバイスのシステム時間を使用します。

説明	ScheduleExpression	DurationInSeconds
24 時間 365 日の記録、時間単位のアップロード	(null ScheduleConfig)	
毎日午前 9:00:00 ~ 午後 4:59:59	0 0 9#16 * * ? *	3599
9:00:00 AM - 4:59:59 PM weekdays	0 0 9#16 ? * 2#6 *	3599

説明	ScheduleExpression	DurationInSeconds
	0 0 9-16 ? * 2,3,4,5,6 *	3599
	0 0 9-16 ? * MON-FRI *	3599
	0 0 9-16 ? * MON,TUE,WED,THU,FRI *	3599
週末の午前 9:00:00 ~ 午後 4:59:59	0 0 9-16 ? * SAT,SUN *	3599
平日の午後 10:00:00 ~ 午後 11:59:59	0 0 22,23 ? * MON-FRI *	3599
毎日午前 9:00:00 ~ 午前 10:00:00	0 0 9 * * ? *	3600
毎日午後 4:00:00 ~ 午後 5:59:59	0 0 16-17 * * ? *	3599

その他の例については、[「Quartz ドキュメント」](#)を参照してください。

ストリームの上限はありますか？

Amazon Kinesis Video Streams Edge Agent のハード制限は現在、デバイスあたり 16 ストリームです。[the section called “DeleteEdgeConfiguration”](#) API を使用して、デバイスからストリームを削除します。を使用して同じストリームの設定を更新[the section called “StartEdgeConfigurationUpdate”](#)しても、デバイスのストリーム数は増加しません。

エラーが発生したジョブを再起動するにはどうすればよいですか？

エラーが発生した場合、Amazon Kinesis Video Streams Edge Agent はジョブの再起動を試みます。ただし、一部のエラー (設定エラーなど) では、ジョブを手動で再起動する必要があります。

手動で再起動する必要があるジョブを確認するには、「」のFatalError メトリクス」を参照してください[the section called “CloudWatch で Amazon Kinesis Video Streams Edge エージェントをモニタリングする”](#)。

ストリームのジョブを再起動[the section called “StartEdgeConfigurationUpdate”](#)するには、 を再送信します。

Amazon Kinesis Video Streams Edge エージェントの状態をモニタリングするにはどうすればよいですか？

詳細については、「[the section called “CloudWatch で Amazon Kinesis Video Streams Edge エージェントをモニタリングする”](#)」を参照してください。

VPC 経由でビデオをストリーミングする

このベータ版は、欧州 (パリ) リージョン eu-west-3 のプレビューで利用できます。これらのコンポーネントと入門ガイドにアクセスするには、E [メールでお問い合わせください](#)。

Amazon Kinesis Video Streams VPC エンドポイントサービスを使用すると、パブリックインターネットを通過するデータなしで、Amazon ネットワーク経由でビデオをストリーミングおよび消費できます。

アクセスをリクエストするには、次の情報を E [メールで送信](#)してください。

- アカウント ID
- ストリーム ARNs
- VPC ID

Note

サービスに追加されるまでに最大 1 週間かかる場合があります。

VPC エンドポイントをこれまで使用したことがない場合は、次の情報を確認して概念を理解してください。

- [AWS PrivateLink バックグラウンド](#)
- [VPC 入門ガイド](#)

追加情報

ベータ版に追加されると、この機能に関する追加情報へのリンクが E メールで送信されます。

VPC エンドポイントの手順

クォータ

主なクォータの違いは次のとおりです。

- すべての帯域幅 APIs の下限クォータ (2 mbps):
 - PutMedia
 - GetMedia
 - GetMediaForFragmentList
- お客様あたり 10 個のストリームが許可されます

エンドポイントを作成する

許可が一覧表示されると、Amazon Kinesis Video Streams の VPC エンドポイントサービス名を受け取ります。のようになります `com.amazonaws.region.kinesisvideo`。

Amazon [VPC コンソール](#) または [\(\)](#) を使用して、[Amazon Kinesis Video Streams のインターフェイス VPC エンドポイント](#) を作成します AWS CLI。Amazon Kinesis Video Streams AWS Command Line Interface

に AWS CLI、次のように入力します。

```
aws ec2 create-vpc-endpoint \  
--vpc-id customer-provided-vpc-id \  
--service-name com.amazonaws.eu-west-2.kinesisvideo \  
--private-dns-enabled
```

Important

VPC 内のトラフィックは、プライベート DNS を使用してエンドポイントをルーティングします。これを有効にしない場合は、独自の DNS ロジックを実装する必要があります。プライベート DNS の詳細については、「[AWS PrivateLink ドキュメント](#)」を参照してください。

AWS CLI オプションの詳細については、「[create-vpc-endpoint](#)」を参照してください。

エンドポイントへのアクセスを制御する

Amazon Kinesis Video Streams へのアクセスを制御するエンドポイントポリシーを VPC エンドポイントにアタッチできます。このポリシーでは、以下の情報を指定します。

- アクションを実行できるプリンシパル、

- 実行できるアクション、および
- アクションを実行できるリソース。

詳細については、「[AWS PrivateLink ガイド](#)」の「[エンドポイントポリシーを使用した VPC エンドポイントによるサービスへのアクセスの制御](#)」を参照してください。

以下は、Amazon Kinesis Video Streams のエンドポイントポリシーの例です。エンドポイントにアタッチすると、このポリシーは、すべてのリソースのすべてのプリンシパルに対するリストされたPutMediaアクションへのアクセスを拒否します。

```
{
  "Statement": [
    {
      "Principal": "*",
      "Effect": "Deny",
      "Action": [
        "kinesisvideo:PutMedia"
      ],
      "Resource": "*"
    }
  ]
}
```


Amazon Kinesis Video Streams の例

次のコード例は、Kinesis Video Streams API を使用する方法を示しています。

例: Kinesis Video Streams へのデータの送信

- [例: Kinesis Video Streams プロデューサー SDK GStreamer プラグイン - kvssink](#): GStreamer の送信先として使用する Kinesis Video Streams プロデューサー SDK を構築する方法を示します。
- [例: Docker コンテナで GStreamer 要素を実行する](#): 構築済みの Docker イメージを使用して、IP カメラから Kinesis Video Streams にリアルタイムストリーミングプロトコル (RTSP) ビデオを送信する方法を示します。
- [例: RTSP ソースからのストリーミング](#): 独自の Docker イメージをビルドして、IP カメラから Kinesis Video Streams に RTSP ビデオを送信する方法を示します。
- [例: PutMedia API を使用して Kinesis Video Streams にデータを送信する](#): を使用して [Java プロデューサーライブラリを使用する](#)、[PutMedia API](#) を使用して、既にコンテナ形式 (MKV) になっている Kinesis Video Streams にデータを送信する方法を示します。

例: Kinesis Video Streams からデータを取得する

- [KinesisVideoExample](#): Kinesis Video Streams パーサーライブラリを使用して、ビデオフラグメントを解析およびログ記録する方法を示します。
- [例: Kinesis Video Streams フラグメントの解析とレンダリング](#): [JCodec](#) および [JFrame](#) を使用して、Kinesis ビデオストリームのフラグメントを解析およびレンダリングする方法を示します。

例: 動画データの再生

- [例: HTML および JavaScript で HLS を使用する](#): Kinesis ビデオストリームの HLS ストリーミングセッションを取得して、ウェブページで再生する方法を示します。

前提条件

- サンプルコードでは、認証情報プロファイルファイルで設定したプロファイルを指定するか、統合開発環境 (IDE) の Java システムプロパティで認証情報を指定して AWS、認証情報を指定しま

す。まだ設定していない場合は、まず認証情報を設定します。詳細については、[「開発用の AWS 認証情報とリージョンの設定」](#)を参照してください。

- コードの表示および実行には次のいずれかの Java IDE の使用をお勧めします。
 - [Eclipse Java Neon](#)
 - [JetBrains IntelliJ IDEA](#)

例: Kinesis Video Streams プロデューサー SDK GStreamer プラグイン - kvssink

このトピックでは、GStreamer プラグインとして使用する Amazon Kinesis Video Streams プロデューサー SDK を構築する方法について説明します。

トピック

- [GStreamer 要素をダウンロード、構築、設定する](#)
- [GStreamer 要素を実行する](#)
- [GStreamer 起動コマンドの例](#)
- [Docker コンテナで GStreamer 要素を実行する](#)
- [GStreamer 要素パラメータリファレンス](#)

[GStreamer](#) は、モジュラープラグインを組み合わせることでカスタムメディアパイプラインを作成するために、複数のカメラとビデオソースで 사용되는一般的なメディアフレームワークです。Kinesis Video Streams GStreamer プラグインは、既存の GStreamer メディアパイプラインと Kinesis Video Streams の統合を合理化します。GStreamer を統合すると、ウェブカメラまたはリアルタイムストリーミングプロトコル (RTSP) カメラから Kinesis Video Streams にビデオをストリーミングして、リアルタイムまたはそれ以降の再生、ストレージ、および詳細な分析を行うことができます。

GStreamer プラグインは、Kinesis Video Streams プロデューサー SDK が提供する機能を GStreamer シンク要素 にカプセル化することで、Kinesis Video Streams へのビデオストリームの転送を自動的に管理します。kvssink。GStreamer フレームワークは、カメラや他のビデオソースのようなデバイスからのメディアフローを構築して処理、レンダリング、保存を行うための標準的なマネージド環境を提供します。

GStreamer パイプラインは通常、ソース (ビデオカメラ) とシンクエレメント (ビデオをレンダリングするプレーヤーやオフラインで取得するためのストレージ) 間のリンクで構成されます。この例では、プロデューサー SDK エレメントをビデオソース (ウェブカメラまたは IP カメラ) のシンク、つ

まりメディア送信先として使用します。SDK をカプセル化するプラグイン要素は、ビデオストリームを Kinesis Video Streams に送信します。

このトピックでは、ウェブカメラや RTSP ストリームなどのビデオソースからビデオをストリーミングできる GStreamer メディアパイプラインを構築する方法について説明します。通常、中間エンコードステージ (H.264 エンコードを使用) を介して Kinesis Video Streams に接続されます。動画ストリームが Kinesis 動画ストリームとして利用可能になったら、[the section called “パーサーライブラリを使用したストリーミング”](#) をさらに処理、再生、保存、または分析できます。

GStreamer 要素をダウンロード、構築、設定する

GStreamer プラグインの例は、Kinesis Video Streams C++ プロデューサー SDK に含まれています。SDK の前提条件およびダウンロードの詳細については、「[C++ プロデューサーライブラリコードをダウンロードして設定する](#)」を参照してください。

プロデューサー SDK GStreamer シンクは、macOS、Ubuntu、Raspberry Pi、または Windows で動的ライブラリとして構築できます。GStreamer プラグインは build ディレクトリにあります。このプラグインをロードするには、`GST_PLUGIN_PATH` にある必要があります。次のコマンドを実行します。

```
export GST_PLUGIN_PATH=`pwd`/build
```

Note

macOS では、Docker コンテナで GStreamer を実行する場合にのみネットワークカメラからビデオをストリーミングできます。Docker コンテナで macOS の USB カメラからのビデオのストリーミングはサポートされていません。

GStreamer 要素を実行する

Kinesis Video Streams プロデューサー SDK 要素をシンクとして GStreamer を実行するには、`gst-launch-1.0` コマンドを使用します。GStreamer プラグインが使用するのに適したアップストリーム要素を使用します。たとえば、Linux システム上の v4l2 デバイスには [v4l2src](#) を、RTSP デバイスには [rtspsrc](#) を使用します。kvssink をシンク (パイプラインの最終的な送信先) としてを指定し、ビデオをプロデューサー SDK に送信します。

[認証情報の提供](#)と[リージョンの提供](#)に加えて、kvssink要素には次の必須パラメータがあります。

- stream-name – 送信先 Kinesis Video Streams の名前。

kvssink のオプションのパラメータの詳細については、「[GStreamer 要素パラメータリファレンス](#)」を参照してください。

GStreamer プラグインとパラメータに関する最新情報については、[GStreamer プラグイン](#)」を参照してください。GStreamer 要素またはプラグインの名前のgst-inspect-1.0後にを使用して、その情報を出力し、デバイスで使用可能であることを確認することもできます。

```
gst-inspect-1.0 kvssink
```

構築がkvssink失敗した場合、または GST_PLUGIN_PATH が正しく設定されていない場合、出力は次のようになります。

```
No such element or plugin 'kvssink'
```

GStreamer 起動コマンドの例

次の例は、kvssinkGStreamer プラグインを使用してさまざまなタイプのデバイスからビデオをストリーミングする方法を示しています。

例 1: Ubuntu の RTSP カメラからビデオをストリーミングする

次のコマンドを実行すると、ネットワーク RTSP カメラからストリーミングする GStreamer パイプラインが Ubuntu に作成されます。これは [rtspsrc](#) GStreamer プラグインを使用します。

```
gst-launch-1.0 -v rtspsrc location="rtsp://YourCameraRtspUrl" short-header=TRUE !  
rtph264depay ! h264parse ! kvssink stream-name="YourStreamName" storage-size=128
```

例 2: Ubuntu の USB カメラでビデオをエンコードしてストリーミングする

次のコマンドを実行すると、USB カメラからのストリームを H.264 形式でエンコードし、Kinesis Video Streams にストリーミングする GStreamer パイプラインが Ubuntu に作成されます。この例では [v4l2src](#) GStreamer プラグインを使用します。

```
gst-launch-1.0 v4l2src do-timestamp=TRUE device=/dev/video0 ! videoconvert ! video/x-  
raw,format=I420,width=640,height=480,framerate=30/1 ! x264enc bframes=0 key-int-max=45  
bitrate=500 ! video/x-h264,stream-format=avc,alignment=au,profile=baseline ! kvssink
```

```
stream-name="YourStreamName" storage-size=512 access-key="YourAccessKey" secret-  
key="YourSecretKey" aws-region="YourAWSRegion"
```

例 3: Ubuntu の USB カメラから事前にエンコードされたビデオをストリーミングする

次のコマンドを実行すると、カメラが H.264 形式でエンコード済みのビデオを Kinesis Video Streams にストリーミングする GStreamer パイプラインが Ubuntu に作成されます。この例では [v4l2src](#) GStreamer プラグインを使用します。

```
gst-launch-1.0 v4l2src do-timestamp=TRUE device=/dev/video0 ! h264parse ! video/x-  
h264,stream-format=avc,alignment=au ! kvssink stream-name="plugin" storage-size=512  
access-key="YourAccessKey" secret-key="YourSecretKey" aws-region="YourAWSRegion"
```

例 4: macOS のネットワークカメラからビデオをストリーミングする

次のコマンドを実行すると、ビデオをネットワークカメラから Kinesis Video Streams にストリーミングする GStreamer パイプラインが macOS に作成されます。この例では [rtspsrc](#) GStreamer プラグインを使用します。

```
gst-launch-1.0 rtspsrc location="rtsp://YourCameraRtspUrl" short-header=TRUE !  
rtph264depay ! h264parse ! video/x-h264, format=avc,alignment=au ! kvssink  
stream-name="YourStreamName" storage-size=512 access-key="YourAccessKey" secret-  
key="YourSecretKey" aws-region="YourAWSRegion"
```

例 5: Windows のネットワークカメラからビデオをストリーミングする

次のコマンドを実行すると、ビデオをネットワークカメラから Kinesis Video Streams にストリーミングする GStreamer パイプラインが Windows に作成されます。この例では [rtspsrc](#) GStreamer プラグインを使用します。

```
gst-launch-1.0 rtspsrc location="rtsp://YourCameraRtspUrl" short-header=TRUE !  
rtph264depay ! video/x-h264, format=avc,alignment=au ! kvssink stream-  
name="YourStreamName" storage-size=512 access-key="YourAccessKey" secret-  
key="YourSecretKey" aws-region="YourAWSRegion"
```

例 6: Raspberry Pi のカメラからビデオをストリーミングする

次のコマンドを実行すると、ビデオを Kinesis Video Streams にストリーミングする GStreamer パイプラインが Raspberry Pi に作成されます。この例では [v4l2src](#) GStreamer プラグインを使用します。

```
gst-launch-1.0 v4l2src do-timestamp=TRUE device=/dev/video0 ! videoconvert !  
video/x-raw,format=I420,width=640,height=480,framerate=30/1 !  
omxh264enc control-rate=1 target-bitrate=5120000 periodicity-  
idr=45 inline-header=FALSE ! h264parse ! video/x-h264,stream-  
format=avc,alignment=au,width=640,height=480,framerate=30/1,profile=baseline ! kvssink  
stream-name="YourStreamName" access-key="YourAccessKey" secret-key="YourSecretKey"  
aws-region="YourAWSRegion"
```

例 7: Raspberry Pi と Ubuntu でオーディオとビデオの両方をストリーミングする

[gst-launch-1.0 コマンドを実行して、Raspberry-PI および Ubuntu でオーディオとビデオの両方のストリーミングを開始する方法](#)について説明します。

例 8: macOS のデバイスソースからオーディオとビデオの両方をストリーミングする

[gst-launch-1.0 コマンドを実行して、MacOS でオーディオとビデオの両方のストリーミングを開始する方法](#)について説明します。

例 9: オーディオとビデオの両方を含む MKV ファイルをアップロードする

[gst-launch-1.0 コマンドを実行して、オーディオとビデオの両方を含む MKV ファイルをアップロードする方法](#)について説明します。h.264 および AAC でエンコードされたメディアを含む MKV テストファイルが必要です。

Docker コンテナで GStreamer 要素を実行する

Docker は、コンテナを使用してアプリケーションを開発、デプロイ、実行するためのプラットフォームです。Docker を使用して GStreamer パイプラインを作成すると、Kinesis Video Streams の運用環境が標準化され、アプリケーションの構築と使用が合理化されます。

Docker をインストールして設定するには、以下を参照してください。

- [Docker のダウンロード手順](#)
- [Docker の開始方法](#)

Docker をインストールしたら、次のいずれかの `docker pull` コマンドを使用して、Amazon Elastic Container Registry から Kinesis Video Streams C++ プロデューサー SDK (および GStreamer プラグイン) をダウンロードできます。

Kinesis Video Streams プロデューサー SDK 要素を Docker コンテナのシンクとして GStreamer を実行するには、次の手順を実行します。

トピック

- [Docker クライアントを認証する](#)
- [Ubuntu、macOS、Windows、または Raspberry Pi の Docker イメージのダウンロード](#)
- [Docker イメージを実行する](#)

Docker クライアントを認証する

イメージのプル元になる Amazon ECR レジストリに対して Docker クライアントを認証します。使用するレジストリごとに認証トークンを取得する必要があります。トークンは 12 時間有効です。詳細については、Amazon Elastic Container Registry ユーザーガイドの[レジストリの認証](#)を参照してください。

Example: Amazon ECR を使用して認証する

Amazon ECR で認証するには、次のコマンドをコピーして貼り付けます。

```
sudo aws ecr get-login-password --region us-west-2 | docker login -u AWS --password-stdin https://546150905175.dkr.ecr.us-west-2.amazonaws.com
```

成功すると、Login Succeeded が出力されます。

Ubuntu、macOS、Windows、または Raspberry Pi の Docker イメージのダウンロード

オペレーティングシステムに応じて次のコマンドのいずれかを使用し、Docker イメージを Docker 環境にダウンロードします。

Ubuntu の Docker イメージのダウンロード

```
sudo docker pull 546150905175.dkr.ecr.us-west-2.amazonaws.com/kinesis-video-producer-sdk-cpp-amazon-linux:latest
```

macOS の Docker イメージのダウンロード

```
docker pull 546150905175.dkr.ecr.us-west-2.amazonaws.com/kinesis-video-producer-sdk-cpp-amazon-linux:latest
```


Windows の Docker イメージのダウンロード

```
docker pull 546150905175.dkr.ecr.us-west-2.amazonaws.com/kinesis-video-producer-sdk-cpp-amazon-windows:latest
```

Raspberry Pi の Docker イメージのダウンロード

```
sudo docker pull 546150905175.dkr.ecr.us-west-2.amazonaws.com/kinesis-video-producer-sdk-cpp-raspberry-pi:latest
```

イメージが正常に追加されたことを確認するには、次のコマンドを使用します。

```
docker images
```

Docker イメージを実行する

オペレーティングシステムに応じて、次のコマンドのいずれかを使用して Docker イメージを実行します。

Ubuntu で Docker イメージを実行する

```
sudo docker run -it --network="host" --device=/dev/video0 546150905175.dkr.ecr.us-west-2.amazonaws.com/kinesis-video-producer-sdk-cpp-amazon-linux /bin/bash
```

macOS で Docker イメージを実行する

```
sudo docker run -it --network="host" 546150905175.dkr.ecr.us-west-2.amazonaws.com/kinesis-video-producer-sdk-cpp-amazon-linux /bin/bash
```

Windows で Docker イメージを実行する

```
docker run -it 546150905175.dkr.ecr.us-west-2.amazonaws.com/kinesis-video-producer-sdk-cpp-windows AWS_ACCESS_KEY_ID AWS_SECRET_ACCESS_KEY RTSP_URL STREAM_NAME
```

Raspberry Pi で Docker イメージを実行する

```
sudo docker run -it --device=/dev/video0 --device=/dev/vchiq -v /opt/vc:/opt/vc 546150905175.dkr.ecr.us-west-2.amazonaws.com/kinesis-video-producer-sdk-cpp-raspberry-pi /bin/bash
```


Docker はコンテナを起動し、コンテナ内でコマンドを使用するためのコマンドプロンプトを表示します。

コンテナ内で、次のコマンドを使用して環境変数を設定します。

```
export LD_LIBRARY_PATH=/opt/awssdk/amazon-kinesis-video-streams-producer-sdk-cpp/kinesis-video-native-build/downloads/local/lib:$LD_LIBRARY_PATH
export PATH=/opt/awssdk/amazon-kinesis-video-streams-producer-sdk-cpp/kinesis-video-native-build/downloads/local/bin:$PATH
export GST_PLUGIN_PATH=/opt/awssdk/amazon-kinesis-video-streams-producer-sdk-cpp/kinesis-video-native-build/downloads/local/lib:$GST_PLUGIN_PATH
```

kvssink を使用して gst-launch-1.0 へのストリーミングを開始し、デバイスとビデオソースに適したパイプラインを実行します。パイプラインの例については、「」を参照してください[GStreamer 起動コマンドの例](#)。

GStreamer 要素パラメータリファレンス

Amazon Kinesis Video Streams プロデューサー C++ SDK に動画を送信するには、パイプラインのシンクまたは最終送信先kvssinkとして を指定します。このリファレンスでは、kvssink の必須およびオプションのパラメータに関する情報を提供します。詳細については、「[the section called “GStreamer プラグイン - kvssink”](#)」を参照してください。

トピック

- [the section called “認証情報を に提供する kvssink”](#)
- [the section called “にリージョンを指定する kvssink”](#)
- [the section called “kvssink オプションのパラメータ”](#)

認証情報を に提供する kvssink

kvssink GStreamer 要素がリクエストを実行できるようにするには AWS、Amazon Kinesis Video Streams サービスを呼び出すときに使用する AWS 認証情報を指定します。認証情報プロバイダーチェーンは、次の順序で認証情報を検索します。

1. AWS IoT 認証情報

AWS IoT 認証情報を設定するには、「」を参照してください[the section called “を使用した Kinesis Video Streams リソースへのアクセスの制御 AWS IoT”](#)。

iot-credentials パラメータ値は で始まり、次の`##iot-certificate`,と`##`ペアのカンマ区切りリストが続く必要があります。

Key	必要	説明
ca-path	はい	TLS を介してバックエンドサービスとの信頼を確立するために使用される CA 証明書へのファイルパス。 Example 例: <code>/file/path/to/certificate.pem</code>
cert-path	はい	X.509 証明書へのファイルパス。 Example 例: <code>/file/path/to/certificateID -certificate.pem.crt</code>
endpoint	はい	AWS アカウントの AWS IoT Core 認証情報エンドポイントプロバイダーエンドポイント。「AWS IoT デベロッパーガイド」を参照してください。 Example 例: <code>credential-account-specific-prefix .credentials.iot. aws-region .amazonaws.com</code>

Key	必要	説明
key-path	はい	パブリック/プライベートキーペアで使用されるプライベートキーへのファイルパス。 Example 例: <pre><i>/file/path/to/certificateID</i> -private.pem.key</pre>
role-aliases	はい	への接続時に使用する AWS IAM ロールを指すロールエイリアスの名前 AWS IoT Core。 Example 例: <i>KvsCameraIoTRoleAlias</i>
iot-thing-name	いいえ	iot-thing-name はオプションです。が指定されiot-thing-name ていない場合は、stream-name パラメータ値が使用されます。 Example 例: <i>kvs_example_camera</i>

Example

例:

```
gst-launch-1.0 -v ... ! kvssink stream-name="YourStream" aws-region="YourRegion"  
  iot-certificate="iot-certificate,endpoint=credential-account-specific-prefix.credentials.iot.aws-region.amazonaws.com,cert-path=certificateID-
```

```
certificate.pem.crt,key-path=certificateID-private.pem.key,ca-  
path=certificate.pem,role-aliases=YourRoleAlias,iot-thing-name=YourThingName"
```

2. 環境変数

で環境の認証情報kvssinkを使用するには、次の環境変数を設定します。

環境変数名	必要	説明
AWS_ACCESS_KEY_ID	はい	Amazon Kinesis Video Streams へのアクセスに使用される AWS アクセスキー。
AWS_SECRET_ACCESS_KEY	はい	アクセスキーに関連付けられた AWS シークレットキー。
AWS_SESSION_TOKEN	いいえ	オペレーションから直接 AWS STS 一時的なセキュリティ認証情報を使用する場合は、必要なセッショントークン値を指定します。

環境変数を設定すると使用する値が変更され、その値はシェルセッションが終了するか、または変数に別の値が設定されるまで有効です。変数を今後のセッションで永続化するには、シェルのスタートアップスクリプトで変数を設定します。

3. access-key、secret-keyパラメータ

認証情報をkvssinkパラメータとして直接指定するには、次のパラメータを設定します。

kvssink パラメータ名	必要	説明
access-key	はい	Amazon Kinesis Video Streams へのアクセスに使用される AWS アクセスキー。
secret-key	はい	アクセスキーに関連付けられた AWS シークレットキー。

kvssink パラメータ名	必要	説明
session-token	いいえ	オペレーションから直接 AWS STS 一時的なセキュリティ認証情報を使用する場合は、必要なセッショントークン値を指定します。

Example

静的認証情報の使用:

```
gst-launch-1.0 -v ... ! kvssink stream-name="YourStream" aws-region="YourRegion"  
access-key="AKIDEXAMPLE" secret-key="SKEEXAMPLE"
```

Example

一時的な認証情報の使用:

```
gst-launch-1.0 -v ... ! kvssink stream-name="YourStream" aws-region="YourRegion"  
access-key="AKIDEXAMPLE" secret-key="SKEEXAMPLE" session-token="STEXAMPLE"
```

4. 認証情報ファイル

Important

前述の方法のいずれかを選択した場合、credential-pathkvssinkパラメータを使用することはできません。

kvssink パラメータ名	必要	説明
credential-path	はい	特定の形式の認証情報を含むテキストファイルへのパス。

テキストファイルには、次のいずれかの形式の認証情報が含まれている必要があります。

- CREDENTIALS *YourAccessKey YourSecretKey*
- CREDENTIALS *YourAccessKey ##### YourSecretKey SessionToken*

Example

例: *credentials.txt* ファイルは `/home/ubuntu/` にあり、以下が含まれています。

```
CREDENTIALS AKIDEXAMPLE 2023-08-10T22:43:00Z SKEXAMPLE STEXAMPLE
```

で使用するには `kvssink`、次のように入力します。

```
gst-launch-1.0 -v ... ! kvssink stream-name="YourStream" aws-region="YourRegion"  
credential-path="/home/ubuntu/credentials.txt"
```

Note

有効期限は、少なくとも $5 + 30 + 3 = 38$ 秒先である必要があります。猶予期間は、`IOT_CREDENTIAL_FETCH_GRACE_PERIOD` 変数として定義されます [IotCredentialProvider.h](#)。起動時に認証情報の有効期限に近すぎる場合は `kvssink`、エラーコード `0x52000049 - STATUS_INVALID_TOKEN_EXPIRATION` が表示されます。

Important

`kvssink` は認証情報ファイルを変更しません。一時的な認証情報を使用している場合は、有効期限から猶予期間を引いた時間より前に、外部ソースによって認証情報ファイルを更新する必要があります。

にリージョンを指定する `kvssink`

リージョンのルックアップ順序は次のとおりです。

1. `AWS_DEFAULT_REGION` 環境変数が最初にレビューされます。設定されている場合、そのリージョンを使用してクライアントを設定します。
2. `aws-region` 次に、パラメータを確認します。設定されている場合、そのリージョンを使用してクライアントを設定します。

3. 上記のいずれの方法も使用されなかった場合、はkvssinkデフォルトで になりますus-west-2。

kvssink オプションのパラメータ

kvssink エlementには以下のオプションパラメータがあります。これらのパラメータの詳細については、「[Kinesis ビデオストリーム構造](#)」を参照してください。

パラメータ	説明	単位/タイプ	デフォルト
stream-name	送信先の Amazon Kinesis ビデオストリームの名前。  Important stream-name を指定しない場合、デフォルトのストリーム名「DEFAULT_STREAM」が使用されます。そのデフォルト名を持つストリームがまだ存在しない場合、ストリームが作成されます。		
absolute-fragment-times	絶対フラグメントタイムを使用するかどうか。	ブール値	true

パラメータ	説明	単位/タイプ	デフォルト
access-key	Kinesis Video Streams へのアクセスに使用される AWS アクセスキー。 AWS 認証情報を設定するか、このパラメータを指定する必要があります。この情報を指定するには、次のように入力します。 <pre>export AWS_ACCESS_KEY_ID=</pre>		
avg-bandwidth-bps	ストリームの予想される平均帯域幅。	毎秒ビット	4194304

パラメータ	説明	単位/タイプ	デフォルト
aws-region	AWS リージョン 使用する。  Note リージョンに AWS_DEFAULT_REGION 環境変数を指定することもできます。環境変数と kvssink パラメータの両方が設定されている場合、環境変数が優先されます。  Important 特に指定 us-west-2 しない場合、リージョンはデフォルトになります。	String	"us-west-2"
buffer-duration	ストリームのバッファ期間。	秒	120
codec-id	ストリームのコーデック ID。	String	"V_MPEG4/ISO/AVC"

パラメータ	説明	単位/タイプ	デフォルト
connection-staleness	ストリームの古さコールバックが呼び出されるまでの時間。	秒	60
content-type	ストリームのコンテンツタイプ。	String	"video/h264"
fragment-acks	ACK フラグメントを使用するかどうか。	ブール値	true
fragment-duration	フラグメントの有効期間。	ミリ秒	2000
framerate	予想されるフレームレート。	1 秒あたりのフレーム	25
frame-timecodes	フレームのタイムコードを使用するか、現在時刻のコールバックを使用してタイムスタンプを生成するかどうかを指定します。	ブール値	true
key-frame-fragmentation	キーフレームでフラグメントを生成するかどうかを指定します。	ブール値	true
log-config	ログ設定のパス。	String	"../kvs_log_configuration"
max-latency	ストリームの最大レイテンシー。	秒	60

パラメータ	説明	単位/タイプ	デフォルト
recalculate-metrics	メトリクスを再計算するかどうかを指定します。	ブール値	true
replay-duration	再開が有効になっている場合、エラー発生時の再生まで現在のリーダーをロールバックする期間。	秒	40
restart-on-error	エラーが発生したときに再起動するかどうか。	ブール値	true
retention-period	ストリームが保持される期間。	時間	2
rotation-period	キーの更新間隔。詳細については、 「キーのローテーション AWS KMS」 を参照してください。	秒	3600
secret-key	Kinesis Video Streams へのアクセスに使用される AWS シークレットキー。 AWS 認証情報を設定するか、このパラメータを指定する必要があります。 <pre>export AWS_SECRET_ACCESS_KEY=</pre>		

パラメータ	説明	単位/タイプ	デフォルト
session-token	オペレーションから直接 AWS STS 一時的なセキュリティ認証情報を使用する場合は、必要なセッショントークン値を指定します。		
storage-size	メビバイト (MiB) 単位のデバイスストレージサイズ。デバイスストレージの構成の詳細については、「 StorageInfo 」を参照してください。	メビバイト (MiB)	128
streaming-type	ストリーミングタイプ。有効な値を次に示します。 0: リアルタイム 1: ほぼリアルタイム (現在サポートされていません) 2: オフライン	列挙型 GstKvsSinkStreamingType	0: リアルタイム
timecode-scale	MKV のタイムコードスケール。	ミリ秒	1
track-name	MKV トラック名。	String	"kinesis_video"

パラメータ	説明	単位/タイプ	デフォルト
iot-certificate	AWS IoT kvssink要素で使用する 認証情報。 iot-certificate は、次のキーと値を受け入れます。  Note iot-thing-name はオプションです。が指定されiot-thing-name ていない場合は、stream-name パラメータ値が使用されます。 • endpoint=iotcredentialsproviderendpoint • cert-path=/localdirectorypath/to/certificate • key-path=/localdirectorypath /	String	なし

パラメータ	説明	単位/タイプ	デフォルト
	<code>to/private/ key</code> • <code>ca-path=/ localdir ectorypath/ to/ca-cert</code> • <code>role-alia ses =role-alia ses</code> • <code>iot-thing- name=YourIotTh ingName</code>		

例: PutMedia API を使用して Kinesis Video Streams にデータを送信する

この例は、[PutMedia](#) API を使用して以下を行う方法を説明するものです。既にコンテナ形式 (MKV) のデータを送信する方法を示します。送信する前にデータをコンテナ形式にアセンブルする必要がある場合 (カメラビデオデータをフレームにアセンブルする場合など) は、「」を参照してください[Kinesis Video Streams へのアップロード](#)。

Note

PutMedia オペレーションは C++ および Java SDKs でのみ使用できます。これは、接続、データフロー、確認の全二重管理によるものです。他の言語ではサポートされていません。

この例には以下のステップが含まれます。

- [コードをダウンロードして設定する](#)
- [コードを記述して調べる](#)
- [コードを実行して検証する](#)

コードをダウンロードして設定する

手順に従って、Java サンプルコードをダウンロードし、プロジェクトを Java IDE にインポートし、ライブラリの場所を設定し、認証情報を使用する AWS ようにコードを設定します。

1. ディレクトリを作成し、GitHub リポジトリのソースコード例からクローンを作成します。PutMedia の例は、[Java](#) の一部です。

```
git clone https://github.com/aws-labs/amazon-kinesis-video-streams-producer-sdk-java
```

2. 使用している Java IDE ([Eclipse](#) や [IntelliJ IDEA](#) など) を開き、ダウンロードした Apache Maven プロジェクトをインポートします。

- Eclipse では: [ファイル]、[インポート]、[Maven]、[Existing Maven Projects (既存の Maven プロジェクト)] を選択し、ダウンロードしたパッケージのルートに移動します。pom.xml ファイルを選択します。
- IntelliJ Idea では: [インポート] を選択します。ダウンロードしたパッケージのルートに含まれる pom.xml ファイルに移動します。

詳細については、関連する IDE ドキュメントを参照してください。

3. インポートしたライブラリのロケーションが IDE で見つかるようにするため、プロジェクトを更新します。

- IntelliJ IDEA の場合は以下を実行します。
 - a. プロジェクトの lib ディレクトリのコンテキスト (右クリック) メニューを開き、[Add as library] を選択します。
 - b. File を選択し、Project Structure を選択します。
 - c. [Project Settings] で [Modules] を選択します。
 - d. [Sources] タブで Language Level を 7 以上に設定します。
- Eclipse の場合は以下を実行します。
 - a. プロジェクトのコンテキスト (右クリック) メニューを開き、[プロパティ]、[Java Build Path]、[ソース] の順に選択します。次に、以下の操作を実行します。
 1. [Source] タブで、[Native library location] をダブルクリックします。
 2. [Native Library Folder Configuration] ウィザードで [Workspace] を選択します。
 3. [Native Library Folder] の選択肢からプロジェクトの lib ディレクトリを選択します。

- b. プロジェクトのコンテキスト (右クリック) メニューを開き、[プロパティ] を選択します。次に、以下の操作を実行します。
 1. [Libraries] タブで、[Add Jars] を選択します。
 2. [JAR selection (JAR 選択)] ウィザードで、プロジェクトの lib ディレクトリ内のすべての .jar を選択します。

コードを記述して調べる

PutMedia API の例 () は、次のコーディングパターンを示しています。PutMediaDemo

トピック

- [PutMediaClient を作成する](#)
- [メディアをストリーミングしてスレッドを一時停止する](#)

このセクションのコード例は、PutMediaDemo クラスのものです。

PutMediaClient を作成する

PutMediaClient オブジェクトを作成するには、次のパラメータが必要です。

- PutMedia エンドポイントの URI。
- ストリーミングする MKV ファイルを指す InputStream。
- ストリーム名。この例では、[Java プロデューサーライブラリを使用する](#) (my-stream) で作成されたものと同じストリームを使用します。別のストリームを使用するには、以下のパラメータを変更します。

```
private static final String STREAM_NAME="my-stream";
```

Note

PutMedia API の例では、ストリームを作成しません。ストリームは、Kinesis Video Streams コンソール[Java プロデューサーライブラリを使用する](#)、または のテストアプリケーションを使用して作成する必要があります AWS CLI。

- 現在のタイムスタンプ。

- タイムコードのタイプ。この例では RELATIVE が使用されます。これは、タイムスタンプがコンテナの開始を基準にしていることを示します。
- 受信したパケットが承認済の送信者から送信されたことを確認する `AWSKinesisVideoV4Signer` オブジェクト。
- 最大アップストリーム帯域幅 (Kbps)
- パケットの送達確認を受け取る `AckConsumer` オブジェクト。

`PutMediaClient` オブジェクトは以下のコードを作成します。

```
/* actually URI to send PutMedia request */
final URI uri = URI.create(KINESIS_VIDEO_DATA_ENDPOINT + PUT_MEDIA_API);

/* input stream for sample MKV file */
final InputStream inputStream = new FileInputStream(MKV_FILE_PATH);

/* use a latch for main thread to wait for response to complete */
final CountDownLatch latch = new CountDownLatch(1);

/* a consumer for PutMedia ACK events */
final AckConsumer ackConsumer = new AckConsumer(latch);

/* client configuration used for AWS SigV4 signer */
final ClientConfiguration configuration = getClientConfiguration(uri);

/* PutMedia client */
final PutMediaClient client = PutMediaClient.builder()
    .putMediaDestinationUri(uri)
    .mkvStream(inputStream)
    .streamName(STREAM_NAME)
    .timestamp(System.currentTimeMillis())
    .fragmentTimeCodeType("RELATIVE")
    .signWith(getKinesisVideoSigner(configuration))
    .upstreamKbps(MAX_BANDWIDTH_KBPS)
    .receiveAcks(ackConsumer)
    .build();
```

メディアをストリーミングしてスレッドを一時停止する

クライアントが作成されると、サンプルが `putMediaInBackground` との同時ストリーミングを開始します。AckConsumer が返されるまでメインスレッドは `latch.await` で一時停止し、この時点でクライアントは切断されます。

```
/* start streaming video in a background thread */
    client.putMediaInBackground();

    /* wait for request/response to complete */
    latch.await();

    /* close the client */
    client.close();
```

コードを実行して検証する

PutMedia API の例を実行するには、以下を実行します。

1. Kinesis Video Streams コンソールで、または AWS CLIを使用して `my-stream` という名前のストリームを作成します。
2. 作業ディレクトリを Java プロデューサー SDK ディレクトリに変更します。

```
cd /<YOUR_FOLDER_PATH_WHERE_SDK_IS_DOWNLOADED>/amazon-kinesis-video-streams-
producer-sdk-java/
```

3. Java SDK およびデモアプリケーションをコンパイルします。

```
mvn package
```

4. /tmp ディレクトリに一時ファイル名を作成します。

```
jar_files=$(mktemp)
```

5. ローカルリポジトリからファイルへの依存関係のクラスパス文字列を作成します。

```
mvn -Dmdep.outputFile=$jar_files dependency:build-classpath
```

6. LD_LIBRARY_PATH 環境変数の値を次のように設定します。

```
export LD_LIBRARY_PATH=/<YOUR_FOLDER_PATH_WHERE_SDK_IS_DOWNLOADED>/amazon-kinesis-  
video-streams-producer-sdk-cpp/kinesis-video-native-build/downloads/local/lib:  
$LD_LIBRARY_PATH  
$ classpath_values=$(cat $jar_files)
```

7. 認証情報を指定 AWS して、次のようにコマンドラインからデモを実行します。

```
java -classpath target/kinesisvideo-java-demo-1.0-SNAPSHOT.jar:$classpath_values -  
Daws.accessKeyId=${ACCESS_KEY} -Daws.secretKey=${SECRET_KEY} -Djava.library.path=/  
opt/amazon-kinesis-video-streams-producer-sdk-cpp/kinesis-video-native-build  
com.amazonaws.kinesisvideo.demoapp.DemoAppMain
```

8. [Kinesis Video Streams コンソール](#)を開き、ストリームの管理ページでストリームを選択します。動画が [Video Preview] ペインで再生されます。

例: RTSP ソースからのストリーミング

には、リアルタイムストリーミングプロトコル (RTSP) ネットワークカメラに接続する [Docker](#) コンテナの定義 [C++](#)が含まれています。Docker を使用すると、Kinesis Video Streams の運用環境が標準化され、アプリケーションの構築と使用が合理化されます。

次の手順では、RTSP デモアプリケーションのセットアップ方法と使用方法を示しています。

トピック

- [ビデオチュートリアル](#)
- [前提条件](#)
- [Docker イメージを作成します。](#)
- [RTSP サンプルアプリケーションを実行する](#)

ビデオチュートリアル

この動画では、Raspberry Pi をセットアップして RTSP フィードを AWS クラウドおよび Amazon Kinesis Video Streams に送信する方法を示します。これはend-to-endのデモンストレーションです。

この動画では、フィードからイメージをキャプチャしてコンピュータビジョンと Amazon Rekognition を使用してイメージを処理し、アラートを送信する方法を示します。

前提条件

Kinesis Video Streams RTSP サンプルアプリケーションを実行するには、次を確認する必要があります。

- Docker: Docker のインストールと使用に関する情報については、以下のリンクを参照してください。
 - [Docker のダウンロード手順](#)
 - [Docker の開始方法](#)
- RTSP ネットワークカメラソース: 推奨カメラについては、「[システム要件](#)」を参照してください。

Docker イメージを作成します。

まず、デモアプリケーションが実行される Docker イメージを構築します。

1. Amazon Kinesis Video Streams デモリポジトリのクローンを作成します。

```
git clone https://github.com/aws-samples/amazon-kinesis-video-streams-demos.git
```

2. `Dockerfile` を含むディレクトリに変更します。この場合、[docker-rtsp](#) ディレクトリです。

```
cd amazon-kinesis-video-streams-demos/producer-cpp/docker-rtsp/
```

3. 次のコマンドを使用して Docker イメージを構築します。このコマンドはイメージを作成し、`rtspdockertest` としてタグ付けします。

```
docker build -t rtspdockertest .
```

4. を実行して `docker images`、でタグ付けされたイメージ ID を検索します `rtspdockertest`。

たとえば、以下の出力例では、IMAGE ID は `54f0d65f69b2` です。

REPOSITORY	TAG	IMAGE ID	CREATED	PLATFORM	SIZE
rtspdockertest	latest	54f0d65f69b2	10 minutes ago	linux/arm64	653.1 MiB

これは後のステップで必要になります。

RTSP サンプルアプリケーションを実行する

RTSP サンプルアプリケーションは、Docker コンテナ内または外部から実行できます。以下の適切な手順に従ってください。

トピック

- [Docker コンテナ内](#)
- [Docker コンテナの外](#)

Docker コンテナ内

RTSP サンプルアプリケーションを実行する

1. 次のコマンドを使用して、Amazon Kinesis Video Streams Docker コンテナを起動します。

```
docker run -it YourImageId /bin/bash
```

2. サンプルアプリケーションを起動するには、AWS 認証情報、Amazon Kinesis ビデオストリームの名前、RTSP ネットワークカメラの URL を指定します。

Important

一時的な認証情報を使用している場合は、も指定する必要があります `AWS_SESSION_TOKEN`。以下の 2 番目の例を参照してください。

```
export AWS_ACCESS_KEY_ID=YourAccessKeyId  
export AWS_SECRET_ACCESS_KEY=YourSecretKeyId  
export AWS_DEFAULT_REGION=YourAWSRegion  
./kvs_gstreamer_sample YourStreamName YourRtspUrl
```

一時的な認証情報:

```
export AWS_ACCESS_KEY_ID=YourAccessKeyId  
export AWS_SECRET_ACCESS_KEY=YourSecretKeyId
```

```
export AWS_SESSION_TOKEN=YourSessionToken
export AWS_DEFAULT_REGION=YourAWSRegion
./kvs_gstreamer_sample YourStreamName YourRtspUrl
```

- にサインイン AWS マネジメントコンソール し、[Kinesis Video Streams コンソール](#)を開きます。

ストリームを表示します。

- Docker コンテナを終了するには、ターミナルウィンドウを閉じるか、と入力しますexit。

Docker コンテナの外

Docker コンテナの外部から、次のコマンドを使用します。

```
docker run -it YourImageId /bin/bash -c "export AWS_ACCESS_KEY_ID=YourAccessKeyId;  
export AWS_SECRET_ACCESS_KEY=YourSecretKeyId; export  
AWS_SESSION_TOKEN=YourSessionToken; export AWS_DEFAULT_REGION=Your AWS Region; ./  
kvs_gstreamer_sample YourStreamName YourRtspUrl"
```

例: Kinesis Video Streams フラグメントの解析とレンダリング

[パーサーライブラリを使用したストリーミング](#) には、Amazon Kinesis ビデオストリームフラグメントの解析とレンダリングを説明する KinesisVideoRendererExample という名前のデモアプリケーションが含まれています。この例では、[例: Kinesis Video Streams プロデューサー SDK GStreamer プラグイン - kvssink](#) アプリケーションを使用して取り込まれた H.264 エンコードのフレームを [JCodec](#) を使用してデコードします。JCodec を使用してフレームをデコードすると、視覚イメージは [JFrame](#) を使用してレンダリングされます。

この例は、次を実行する方法を説明しています。

- GetMedia API を使用して Kinesis ビデオストリームからフレームを取得し、表示するためにストリームをレンダリングします。
- Kinesis Video Streams コンソールを使用する代わりに、カスタムアプリケーションでストリームのビデオコンテンツを表示します。

また、表示される前にデコードを必要としない JPEG ファイルのストリームなど、H.264 としてエンコードされていない Kinesis のビデオストリームコンテンツの表示にこの例のクラスを使用することもできます。

次の手順では、レンダラ-デモアプリケーションのセットアップと使用方法を示しています。

前提条件

レンダラ-の例のライブラリを確かめて使用するには、以下が必要です。

- Amazon Web Services (AWS) アカウント。AWS アカウントをまだお持ちでない場合は、「[Kinesis Video Streams の開始方法](#)」を参照してください。
- [Eclipse Java Neon](#) や [JetBrains IntelliJ Idea](#) などの Java 統合開発環境 (IDE)。

レンダラーの実行例

1. ディレクトリを作成して、GitHub リポジトリのソースコード例からクローンを作成します。

```
git clone https://github.com/aws/amazon-kinesis-video-streams-parser-library
```

2. 使用する Java IDE ([Eclipse](#) や [IntelliJ IDEA](#) など) を開き、ダウンロードした Apache Maven プロジェクトをインポートします。
 - Eclipse では: [ファイル]、[インポート]、[Maven]、[Existing Maven Projects] を選択します。kinesis-video-streams-parser-lib ディレクトリに移動します。
 - IntelliJ Idea では: [インポート] を選択します。ダウンロードしたパッケージのルートに含まれる pom.xml ファイルに移動します。

Note

IntelliJ が依存関係を検出できない場合は、以下の手順の実行が必要になることがあります。

- 構築のクリーン: [File (ファイル)]、[Settings (設定)]、[Build, Execution, Deployment (構築、実行、デプロイ)]、[Compiler (コンパイラー)] の順に選択します。再構築時に出カディレクトリをクリアが選択されていることを確認してから、ビルド、ビルドプロジェクトを選択します。
- プロジェクトの再インポート: プロジェクトのコンテキスト (右クリック) メニューを開き、[Maven]、[Reimport (再インポート)] の順に選択します。

詳細については、関連する IDE ドキュメントを参照してください。

3. Java IDE で `src/test/java/com.amazonaws.kinesisvideo.parser/examples/KinesisVideoRendererExampleTest` を開きます。
4. このファイルから `@Ignore` ディレクティブを削除します。
5. `.stream` パラメータを Kinesis ビデオストリームの名前で更新します。
6. `KinesisVideoRendererExample` テストを実行します。

仕組み

例のアプリケーションは次を示します。

- [MKV データの送信](#)
- [MKV フラグメントをフレームに解析する](#)
- [フレームのデコードと表示](#)

MKV データの送信

この例では、動画データを `[rendering_example_video.mkvrender-example-streamPutMedia]` という名前のストリームに送信するために `PutMediaWorker` を使用して ファイルからサンプル MKV データ を送信します。

このアプリケーションは `PutMediaWorker` を作成します。

```
PutMediaWorker putMediaWorker = PutMediaWorker.create(getRegion(),
    getCredentialsProvider(),
    getStreamName(),
    inputStream,
    streamOps.amazonKinesisVideo);
executorService.submit(putMediaWorker);
```

`PutMediaWorker` クラスの詳細については、[パーサーライブラリを使用したストリーミング](#) ドキュメンテーションで「[PutMedia 呼び出し](#)」を参照してください。

MKV フラグメントをフレームに解析する

この例では次に、`GetMediaWorker` を使用してストリームから MKV フラグメントを取得し、解析します。

```
GetMediaWorker getMediaWorker = GetMediaWorker.create(getRegion(),
```



```
getCredentialsProvider(),
getStreamName(),
new StartSelector().withStartSelectorType(StartSelectorType.EARLIEST),
streamOps.amazonKinesisVideo,
getMediaProcessingArgumentsLocal().getFrameVisitor());
executorService.submit(getMediaWorker);
```

GetMediaWorker クラスの詳細については、[パーサーライブラリを使用したストリーミング](#) ドキュメンテーションで「[GetMedia 呼び出し](#)」を参照してください。

フレームのデコードと表示

この例では次に、[JFrame](#) を使用してフレームをデコードし、表示します。

以下は、JFrame を拡張する KinesisVideoFrameViewer クラスからのコード例です。

```
public void setImage(BufferedImage bufferedImage) {
    image = bufferedImage;
    repaint();
}
```

[java.awt.image.BufferedImage](#) のインスタンスとしてイメージが表示されます。BufferedImage を使用する方法を示す例については、「[Reading/Loading an Image \(イメージの読み取りとロード\)](#)」を参照してください。

API リファレンス

このノードのセクションには、API リファレンスドキュメントが含まれています。左側のペインにある目次を使用して、さまざまな API リファレンスのセクションに移動します。

アクション

以下のアクションは、Amazon Kinesis Video Streams でサポートされています。

- [CreateSignalingChannel](#)
- [CreateStream](#)
- [DeleteEdgeConfiguration](#)
- [DeleteSignalingChannel](#)
- [DeleteStream](#)
- [DescribeEdgeConfiguration](#)
- [DescribeImageGenerationConfiguration](#)
- [DescribeMappedResourceConfiguration](#)
- [DescribeMediaStorageConfiguration](#)
- [DescribeNotificationConfiguration](#)
- [DescribeSignalingChannel](#)
- [DescribeStream](#)
- [DescribeStreamStorageConfiguration](#)
- [GetDataEndpoint](#)
- [GetSignalingChannelEndpoint](#)
- [ListEdgeAgentConfigurations](#)
- [ListSignalingChannels](#)
- [ListStreams](#)
- [ListTagsForResource](#)
- [ListTagsForStream](#)
- [StartEdgeConfigurationUpdate](#)
- [TagResource](#)

- [TagStream](#)
- [UntagResource](#)
- [UntagStream](#)
- [UpdateDataRetention](#)
- [UpdateImageGenerationConfiguration](#)
- [UpdateMediaStorageConfiguration](#)
- [UpdateNotificationConfiguration](#)
- [UpdateSignalingChannel](#)
- [UpdateStream](#)
- [UpdateStreamStorageConfiguration](#)

以下のアクションは、Amazon Kinesis Video Streams Media でサポートされています。

- [GetMedia](#)
- [PutMedia](#)

Amazon Kinesis Video Streams Archived Media では、以下のアクションがサポートされています。

- [GetClip](#)
- [GetDASHStreamingSessionURL](#)
- [GetHLSStreamingSessionURL](#)
- [GetImages](#)
- [GetMediaForFragmentList](#)
- [ListFragments](#)

以下のアクションは、Amazon Kinesis Video Signaling Channels でサポートされています。

- [GetIceServerConfig](#)
- [SendAlexaOfferToMaster](#)

Amazon Kinesis Video WebRTC Storage では、次のアクションがサポートされています。

- [JoinStorageSession](#)

- [JoinStorageSessionAsViewer](#)

Amazon Kinesis Video Streams

以下のアクションは、Amazon Kinesis Video Streams でサポートされています。

- [CreateSignalingChannel](#)
- [CreateStream](#)
- [DeleteEdgeConfiguration](#)
- [DeleteSignalingChannel](#)
- [DeleteStream](#)
- [DescribeEdgeConfiguration](#)
- [DescribeImageGenerationConfiguration](#)
- [DescribeMappedResourceConfiguration](#)
- [DescribeMediaStorageConfiguration](#)
- [DescribeNotificationConfiguration](#)
- [DescribeSignalingChannel](#)
- [DescribeStream](#)
- [DescribeStreamStorageConfiguration](#)
- [GetDataEndpoint](#)
- [GetSignalingChannelEndpoint](#)
- [ListEdgeAgentConfigurations](#)
- [ListSignalingChannels](#)
- [ListStreams](#)
- [ListTagsForResource](#)
- [ListTagsForStream](#)
- [StartEdgeConfigurationUpdate](#)
- [TagResource](#)
- [TagStream](#)
- [UntagResource](#)
- [UntagStream](#)
- [UpdateDataRetention](#)

- [UpdateImageGenerationConfiguration](#)
- [UpdateMediaStorageConfiguration](#)
- [UpdateNotificationConfiguration](#)
- [UpdateSignalingChannel](#)
- [UpdateStream](#)
- [UpdateStreamStorageConfiguration](#)

CreateSignalingChannel

サービス : Amazon Kinesis Video Streams

シグナリングチャネルを作成します。

CreateSignalingChannel は非同期の操作です。

リクエストの構文

```
POST /createSignalingChannel HTTP/1.1
Content-type: application/json
```

```
{
  "ChannelName": "string",
  "ChannelType": "string",
  "SingleMasterConfiguration": {
    "MessageTtlSeconds": number
  },
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ]
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

ChannelName

作成しているシグナリングチャネルの名前。これは、AWS アカウント と ごとに一意である必要があります AWS リージョン。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: はい

[ChannelType](#)

作成しているシグナリングチャンネルのタイプ。現在サポートされている唯一のチャンネルタイプは SINGLE_MASTER です。

型: 文字列

有効な値: SINGLE_MASTER | FULL_MESH

必須: いいえ

[SingleMasterConfiguration](#)

SINGLE_MASTER チャンネルタイプの設定を含む構造体。チャンネルメッセージの有効期間のデフォルト設定は 60 秒 (1 分) です。

型: [SingleMasterConfiguration](#) オブジェクト

必須: いいえ

[Tags](#)

このチャンネルに関連付ける一連のタグ (キーと値のペア)。

型: [Tag](#) オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 50 項目です。

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "ChannelARN": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

ChannelARN

新たに作成されたチャネルの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

AccountChannelLimitExceededException

このリージョンで、この AWS アカウント アクティブなシグナリングチャネルの最大制限に達しました。

HTTP ステータスコード: 400

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceInUseException

の入力または `StreamARNChannelARNCLOUD_STORAGE_MODE` が既に別の Kinesis Video Stream リソースにマッピングされている場合、または指定された入力 `StreamARN` または `ChannelARN` がアクティブステータスでない場合は、次のいずれかを試してください。

1. 指定されたチャンネルがマッピングされるストリームを決定する
`DescribeMediaStorageConfiguration` API。
2. 特定のストリームがマッピングされるチャンネルを決定する
`DescribeMappedResourceConfiguration` API。
3. リソースのステータスを決定する `DescribeStream` または `DescribeSignalingChannel` API。

HTTP ステータスコード: 400

TagsPerResourceExceededLimitException

リソースに関連付けることができるタグの上限を超えています。Kinesis ビデオストリームは、最大 50 個のタグをサポートできます。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateStream

サービス : Amazon Kinesis Video Streams

新しい Kinesis ビデオストリームを作成します。

新しいストリームを作成すると、Kinesis Video Streams によってバージョン番号が割り当てられます。ストリームのメタデータを変更すると、Kinesis Video Streams によってバージョンが更新されます。

CreateStream は非同期の操作です。

サービスの仕組みについては、「[仕組み](#)」を参照してください。

KinesisVideo:CreateStream アクションのアクセス許可が必要です。

リクエストの構文

```
POST /createStream HTTP/1.1
Content-type: application/json

{
  "DataRetentionInHours": number,
  "DeviceName": "string",
  "KmsKeyId": "string",
  "MediaType": "string",
  "StreamName": "string",
  "StreamStorageConfiguration": {
    "DefaultStorageTier": "string"
  },
  "Tags": {
    "string" : "string"
  }
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[DataRetentionInHours](#)

ストリームにデータを保持する時間数。Kinesis Video Streams は、ストリームに関連付けられているデータストアにデータを保持します。

デフォルト値は 0 で、ストリームがデータを維持しないことを示します。最小値は 1 時間です。

DataRetentionInHours の値が 0 の場合でも、コンシューマーはサービスホストバッファに残っているフラグメントを消費できます。このバッファには、5 分の保持時間と 200 MB の容量の制限があります。いずれかの制限に達すると、フラグメントはバッファから削除されます。

型: 整数

値の範囲: 最小値 は 0 です。

必須: いいえ

[DeviceName](#)

ストリームに書き込んでいるデバイスの名前。

Note

現在の実装では、Kinesis Video Streams はこの名前を使用しません。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 128 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

[KmsKeyId](#)

Kinesis Video Streams がストリームデータの暗号化に使用する AWS Key Management Service (AWS KMS) キーの ID。

キー ID が指定されていない場合、デフォルトの Kinesis Video 管理キー (aws/kinesisvideo) が使われます。

詳細については、「[DescribeKey](#)」を参照してください。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 2,048 です。

パターン: .+

必須: いいえ

MediaType

ストリームのメディアタイプ。ストリームのコンシューマーは、ストリームの処理時にこの情報を使用できます。メディアタイプの詳細については、「[メディアタイプ](#)」を参照してください。MediaType を指定する場合は、ガイドラインの「[命名要件](#)」を参照してください。

有効な値の例として、"video/h264" や "video/h264,audio/aac" などがあります。

このパラメータはオプションです。デフォルト値は null (JSON の場合は空) です。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 128 です。

パターン: `[\w\-\.\+]+/[\w\-\.\+]+(,[\w\-\.\+]+/[\w\-\.\+]+)*`

必須: いいえ

StreamName

作成しているストリームの名前。

ストリーム名はストリームの識別子であり、アカウントやリージョンごとに一意である必要があります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: はい

StreamStorageConfiguration

ストリームデータのデフォルトのストレージ階層を含む、ストリームのストレージの設定。この設定は、ストリームデータの保存方法とアクセス方法を決定し、さまざまな階層でさまざまなレベルのパフォーマンスとコストの最適化を提供します。

指定しない場合、ストリームは最適なパフォーマンスを得るために HOT 階層でデフォルトのストレージ設定を使用します。

型: [StreamStorageConfiguration](#) オブジェクト

必須: いいえ

Tags

指定されたストリームに関連付けるタグのリスト。各タグはキーと値のペアです (値はオプションです)。

タイプ: 文字列間のマッピング

マップエントリ: 項目の最大数は 50 です。

キーの長さ制限: 最小長さは 1 です。最大長は 128 です。

キーパターン: $^([\backslash p\{L\}\backslash p\{Z\}\backslash p\{N\}_\cdot:/=+\backslash -@]^*)\$$

値の長さの制限: 最小長は 0 です。最大長は 256 です。

値のパターン: $[\backslash p\{L\}\backslash p\{Z\}\backslash p\{N\}_\cdot:/=+\backslash -@]^*$

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json
```

```
{
  "StreamARN": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[StreamARN](#)

ストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccountStreamLimitExceededException

アカウント用に作成されたストリームの数が多すぎます。

HTTP ステータスコード: 400

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

DeviceStreamLimitExceededException

実装されていません。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

InvalidDeviceException

実装されていません。

HTTP ステータスコード: 400

ResourceInUseException

入力 StreamARN または ChannelARN の CLOUD_STORAGE_MODE が既に別の Kinesis Video Stream リソースにマッピングされている場合、または指定された入力 StreamARN または ChannelARN がアクティブステータスでない場合は、次のいずれかを試してください。

1. 指定されたチャンネルがマッピングされるストリームを決定する
DescribeMediaStorageConfiguration API。
2. 特定のストリームがマッピングされるチャンネルを決定する
DescribeMappedResourceConfiguration API。
3. リソースのステータスを決定する DescribeStreamまたは DescribeSignalingChannel API。

HTTP ステータスコード: 400

TagsPerResourceExceededLimitException

リソースに関連付けることができるタグの上限を超えています。Kinesis ビデオストリームは、最大 50 個のタグをサポートできます。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteEdgeConfiguration

サービス : Amazon Kinesis Video Streams

ストリームの既存のエッジ設定と、エッジエージェントから対応するメディアを削除する非同期 API。

この API を呼び出すと、同期ステータスは に設定されます DELETING。削除プロセスが開始され、アクティブなエッジジョブが停止し、すべてのメディアがエッジデバイスから削除されます。削除時間は、保存されたメディアの合計量によって異なります。削除プロセスが失敗すると、同期ステータスは に変わります DELETE_FAILED。削除を再試行する必要があります。

削除プロセスが正常に完了すると、エッジ設定にアクセスできなくなります。

Note

この API は、AWS アフリカ (ケープタウン) リージョン af-south-1 では利用できません。

リクエストの構文

```
POST /deleteEdgeConfiguration HTTP/1.1
Content-type: application/json
```

```
{
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

StreamARN

ストリームの Amazon リソースネーム (ARN)。StreamName または を指定します StreamARN。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

エッジ設定を削除するストリームの名前。StreamName または を指定しますStreamARN。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

StreamEdgeConfigurationNotFoundException

Amazon Kinesis Video Streams が、指定したストリームのエッジ設定を見つけられないときにレンダリングされる例外。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteSignalingChannel

サービス : Amazon Kinesis Video Streams

指定したシグナリングチャンネルを削除します。DeleteSignalingChannel は非同期の操作です。チャンネルの現在バージョンを指定しない場合、最新バージョンは削除されます。

リクエストの構文

```
POST /deleteSignalingChannel HTTP/1.1
Content-type: application/json

{
  "ChannelARN": "string",
  "CurrentVersion": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

ChannelARN

削除するシグナリングチャンネルの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: はい

CurrentVersion

削除するシグナリングチャンネルの現在のバージョン。DescribeSignalingChannel または ListSignalingChannels API 操作を呼び出すことにより、現在のバージョンを取得できます。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長 64

パターン: [a-zA-Z0-9]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceInUseException

の入力または StreamARNChannelARNCLOUD_STORAGE_MODEが既に別の Kinesis Video Stream リソースにマッピングされている場合、または指定された入力StreamARNまたは ChannelARNがアクティブステータスでない場合は、次のいずれかを試してください。

1. 指定されたチャンネルがマッピングされるストリームを決定する
DescribeMediaStorageConfiguration API。
2. 特定のストリームがマッピングされるチャンネルを決定する
DescribeMappedResourceConfiguration API。
3. リソースのステータスを決定する DescribeStreamまたは DescribeSignalingChannel API。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

VersionMismatchException

指定したストリームバージョンは最新バージョンではありません。最新バージョンを入手するには、[DescribeStream](#) API を使用してください。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteStream

サービス : Amazon Kinesis Video Streams

Kinesis ビデオストリームとストリームに含まれるデータを削除します。

このメソッドは、削除対象のストリームにマークを付けることで、直ちにストリーム内のデータにアクセスできないようにします。

ストリームを削除する前にストリームの最新バージョンを確認するために、ストリームバージョンを指定します。Kinesis Video Streams が各ストリームにバージョンを割り当てます。ストリームを更新すると、Kinesis Video Streams が新しいバージョン番号を割り当てます。最新のストリームバージョンを取得するには、DescribeStream APIを使用します

この操作には KinesisVideo:DeleteStream アクションに対するアクセス許可が必要です。

リクエストの構文

```
POST /deleteStream HTTP/1.1
Content-type: application/json

{
  "CurrentVersion": "string",
  "StreamARN": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

CurrentVersion

オプション: 削除するストリームのバージョン。

安全のためバージョンを指定して、正しいストリームを確実に削除します。ストリームバージョンを取得するには、DescribeStream APIを使用します。

指定しない場合、ストリームの削除前に CreationTime のみがチェックされます。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長 64

パターン: [a-zA-Z0-9]+

必須: いいえ

StreamARN

削除するストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+

必須: はい

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

NotAuthorizedException

呼び出し元には、この操作を実行するための権限がありません。

HTTP ステータスコード: 401

ResourceInUseException

入力 StreamARN または ChannelARN が既に別の Kinesis Video Stream リソースに CLOUD_STORAGE_MODE マッピングされている場合、または指定された入力 StreamARN または ChannelARN がアクティブステータスでない場合は、次のいずれかを試してください。

1. 指定されたチャンネルがマッピングされるストリームを決定する
DescribeMediaStorageConfiguration API。
2. 特定のストリームがマッピングされるチャンネルを決定する
DescribeMappedResourceConfiguration API。
3. リソースのステータスを決定する DescribeStream または DescribeSignalingChannel API。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

VersionMismatchException

指定したストリームバージョンは最新バージョンではありません。最新バージョンを入手するには、[DescribeStream](#) API を使用してください。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)

- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeEdgeConfiguration

サービス : Amazon Kinesis Video Streams

StartEdgeConfigurationUpdate API を使用して設定されたストリームのエッジ設定と、エッジエージェントのレコーダーおよびアップローダージョブの最新ステータスについて説明します。この API を使用して設定のステータスを取得し、設定が Edge エージェントと同期しているかどうかを判断します。この API を使用して、エッジエージェントのヘルスを評価します。

Note

この API は、AWS アフリカ (ケープタウン) リージョン af-south-1 では利用できません。

リクエストの構文

```
POST /describeEdgeConfiguration HTTP/1.1
```

```
Content-type: application/json
```

```
{
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

StreamARN

ストリームの Amazon リソースネーム (ARN)。StreamNameまたは を指定しますStreamARN。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

エッジ設定を更新するストリームの名前。StreamName または を指定しますStreamARN。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "CreationTime": number,
  "EdgeAgentStatus": {
    "LastRecorderStatus": {
      "JobStatusDetails": "string",
      "LastCollectedTime": number,
      "LastUpdatedTime": number,
      "RecorderStatus": "string"
    },
    "LastUploaderStatus": {
      "JobStatusDetails": "string",
      "LastCollectedTime": number,
      "LastUpdatedTime": number,
      "UploaderStatus": "string"
    }
  },
  "EdgeConfig": {
    "DeletionConfig": {
      "DeleteAfterUpload": boolean,
      "EdgeRetentionInHours": number,
      "LocalSizeConfig": {
        "MaxLocalMediaSizeInMB": number,
        "StrategyOnFullSize": "string"
      }
    }
  }
}
```

```

    },
    "HubDeviceArn": "string",
    "RecorderConfig": {
      "MediaSourceConfig": {
        "MediaUriSecretArn": "string",
        "MediaUriType": "string"
      },
      "ScheduleConfig": {
        "DurationInSeconds": number,
        "ScheduleExpression": "string"
      }
    },
    "UploaderConfig": {
      "ScheduleConfig": {
        "DurationInSeconds": number,
        "ScheduleExpression": "string"
      }
    }
  },
  "FailedStatusDetails": "string",
  "LastUpdatedTime": number,
  "StreamARN": "string",
  "StreamName": "string",
  "SyncStatus": "string"
}

```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

CreationTime

ストリームのエッジ設定が最初に作成されたタイムスタンプ。

タイプ: タイムスタンプ

EdgeAgentStatus

エッジエージェントのレコーダージョブとアップローダージョブの最新のステータスの詳細を含むオブジェクト。この情報を使用して、エッジエージェントの現在の状態を判断します。

型: EdgeAgentStatus オブジェクト

[EdgeConfig](#)

Edge Agent IoT Greengrass コンポーネントとの同期に使用されるストリームのエッジ設定の説明。Edge Agent コンポーネントは、オンプレミスの IoT Hub デバイスのセットアップで実行されます。

型: [EdgeConfig](#) オブジェクト

[FailedStatusDetails](#)

生成された障害ステータスの説明。

タイプ: 文字列

[LastUpdatedTime](#)

ストリームのエッジ設定が最後に更新されたタイムスタンプ。

タイプ: タイムスタンプ

[StreamARN](#)

ストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

[StreamName](#)

エッジ設定が更新されたストリームの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

[SyncStatus](#)

エッジ設定の更新の最新ステータス。

型: 文字列

有効な値 : SYNCING | ACKNOWLEDGED | IN_SYNC | SYNC_FAILED | DELETING | DELETE_FAILED | DELETING_ACKNOWLEDGED

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

StreamEdgeConfigurationNotFoundException

Amazon Kinesis Video Streams が、指定したストリームのエッジ設定を見つけられないときにレンダリングされる例外。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeImageGenerationConfiguration

サービス : Amazon Kinesis Video Streams

特定の Kinesis ビデオストリーム ImageGenerationConfiguration のを取得します。

リクエストの構文

```
POST /describeImageGenerationConfiguration HTTP/1.1
Content-type: application/json

{
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

StreamARN

イメージ生成設定を取得する Kinesis ビデオストリームの Amazon リソースネーム (ARN)。StreamName または StreamARN のパラメータを指定する必要があります。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

イメージ生成設定を取得するストリームの名前。StreamName または StreamARN のパラメータを指定する必要があります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "ImageGenerationConfiguration": {
    "DestinationConfig": {
      "DestinationRegion": "string",
      "Uri": "string"
    },
    "Format": "string",
    "FormatConfig": {
      "string": "string"
    },
    "HeightPixels": number,
    "ImageSelectorType": "string",
    "SamplingInterval": number,
    "Status": "string",
    "WidthPixels": number
  }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[ImageGenerationConfiguration](#)

Kinesis ビデオストリーム (KVS) イメージ配信に必要な情報を含む構造。この構造が null の場合、設定はストリームから削除されます。

型: [ImageGenerationConfiguration](#) オブジェクト

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeMappedResourceConfiguration

サービス : Amazon Kinesis Video Streams

ストリームに関する最新の情報を返します。streamName または は入力で指定streamARNする必要があります。

リクエストの構文

```
POST /describeMappedResourceConfiguration HTTP/1.1
Content-type: application/json
```

```
{
  "MaxResults": number,
  "NextToken": "string",
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

MaxResults

レスポンスで返される結果の最大数。

タイプ: 整数

有効な範囲: 固定値は 1 です。

必須: いいえ

NextToken

結果の別のバッチを取得するために、次のリクエストで指定するトークン。

タイプ: 文字列

長さの制約: 最小長は 0 です。最大長は 1,024 です。

パターン: [a-zA-Z0-9+/=]*

必須: いいえ

StreamARN

ストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+

必須: いいえ

StreamName

ストリームの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "MappedResourceConfigurationList": [
    {
      "ARN": "string",
      "Type": "string"
    }
  ],
  "NextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[MappedResourceConfigurationList](#)

メディアストレージ設定プロパティをカプセル化または含む構造。

型: [MappedResourceConfigurationListItem](#) オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 1 項目です。

[NextToken](#)

次の結果セットを取得するためにNextTokenリクエストで使用されたトークン。

タイプ: 文字列

長さの制約: 最小長は 0 です。最大長は 1,024 です。

パターン: [a-zA-Z0-9+/=]*

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeMediaStorageConfiguration

サービス : Amazon Kinesis Video Streams

チャンネルに関する最新の情報を返します。入力ChannelARNで ChannelNameまたは を指定します。

リクエストの構文

```
POST /describeMediaStorageConfiguration HTTP/1.1
Content-type: application/json
```

```
{
  "ChannelARN": "string",
  "ChannelName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

ChannelARN

チャンネルの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

ChannelName

チャンネルの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "MediaStorageConfiguration": {
    "Status": "string",
    "StreamARN": "string"
  }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[MediaStorageConfiguration](#)

メディアストレージ設定プロパティをカプセル化または含む構造。

型: [MediaStorageConfiguration](#) オブジェクト

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeNotificationConfiguration

サービス : Amazon Kinesis Video Streams

特定の Kinesis ビデオストリームNotificationConfigurationの を取得します。

リクエストの構文

```
POST /describeNotificationConfiguration HTTP/1.1
Content-type: application/json
```

```
{
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

StreamARN

通知設定を取得する Kinesis ビデオストリームの Amazon リソースネーム (ARN)。StreamName または StreamARN のいずれかを指定する必要があります。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

通知設定を取得するストリームの名前。StreamName または StreamARN のパラメータを指定する必要があります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "NotificationConfiguration": {
    "DestinationConfig": {
      "Uri": "string"
    },
    "Status": "string"
  }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[NotificationConfiguration](#)

通知に必要な情報を含む構造。構造が null の場合、設定はストリームから削除されます。

型: [NotificationConfiguration](#) オブジェクト

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeSignalingChannel

サービス : Amazon Kinesis Video Streams

シグナリングチャンネルに関する最新の情報を返します。説明するチャンネルの名前または Amazon リソースネーム (ARN) のいずれかを指定する必要があります。

リクエストの構文

```
POST /describeSignalingChannel HTTP/1.1
Content-type: application/json
```

```
{
  "ChannelARN": "string",
  "ChannelName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[ChannelARN](#)

説明するシグナリングチャンネルの ARN。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_-.]+/[0-9]+`

必須: いいえ

[ChannelName](#)

説明するシグナリングチャンネルの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "ChannelInfo": {
    "ChannelARN": "string",
    "ChannelName": "string",
    "ChannelStatus": "string",
    "ChannelType": "string",
    "CreationTime": number,
    "SingleMasterConfiguration": {
      "MessageTtlSeconds": number
    },
    "Version": "string"
  }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[ChannelInfo](#)

指定されたシグナリングチャネルのメタデータとプロパティをカプセル化する構造体。

型: [ChannelInfo](#) オブジェクト

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeStream

サービス : Amazon Kinesis Video Streams

指定されたストリームに関する最新の情報を返します。StreamName または StreamARN のパラメータを指定する必要があります。

リクエストの構文

```
POST /describeStream HTTP/1.1
Content-type: application/json
```

```
{
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

StreamARN

ストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

ストリームの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "StreamInfo": {
    "CreationTime": number,
    "DataRetentionInHours": number,
    "DeviceName": "string",
    "KmsKeyId": "string",
    "MediaType": "string",
    "Status": "string",
    "StreamARN": "string",
    "StreamName": "string",
    "Version": "string"
  }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[StreamInfo](#)

ストリームを記述するオブジェクト。

型: [StreamInfo](#) オブジェクト

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

NotAuthorizedException

呼び出し元には、この操作を実行するための権限がありません。

HTTP ステータスコード: 401

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeStreamStorageConfiguration

サービス : Amazon Kinesis Video Streams

指定された Kinesis ビデオストリームの現在のストレージ設定を取得します。

リクエストでは、StreamName または StreamARN のパラメータを指定する必要があります。

KinesisVideo:DescribeStreamStorageConfiguration アクションのアクセス許可が必要です。

リクエストの構文

```
POST /describeStreamStorageConfiguration HTTP/1.1
Content-type: application/json
```

```
{
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

StreamARN

ストレージ設定を取得するストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

ストレージ設定を取得するストリームの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "StreamARN": "string",
  "StreamName": "string",
  "StreamStorageConfiguration": {
    "DefaultStorageTier": "string"
  }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

StreamARN

ストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+

StreamName

ストリームの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

[StreamStorageConfiguration](#)

デフォルトのストレージ階層やその他のストレージ関連の設定を含む、ストリームの現在のストレージ設定。

型: [StreamStorageConfiguration](#) オブジェクト

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

GetDataEndpoint

サービス : Amazon Kinesis Video Streams

読み取りまたは書き込みするために指定されたストリームのエンドポイントを取得します。アプリケーションでこのエンドポイントを使用して、指定されたストリームから読み取る (GetMedia または GetMediaForFragmentList 操作を使用) か、書き込み (PutMedia 操作を使用) します。

Note

返されたエンドポイントには API 名が付けられていません。クライアントは、返されたエンドポイントに API 名を追加する必要があります。

リクエストでは、StreamName または StreamARN でストリームを指定します。

リクエストの構文

```
POST /getDataEndpoint HTTP/1.1
Content-type: application/json
```

```
{
  "APIName": "string",
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

APIName

エンドポイントを取得する API アクションの名前。

型: 文字列

有効な値: PUT_MEDIA | GET_MEDIA | LIST_FRAGMENTS |
GET_MEDIA_FOR_FRAGMENT_LIST | GET_HLS_STREAMING_SESSION_URL |
GET_DASH_STREAMING_SESSION_URL | GET_CLIP | GET_IMAGES

必須: はい

StreamARN

エンドポイントを取得するストリームの Amazon リソースネーム (ARN)。リクエストでは、このパラメータまたは StreamName を指定する必要があります。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+

必須: いいえ

StreamName

エンドポイントを取得するストリームの名前。リクエストでは、このパラメータまたは StreamARN を指定する必要があります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "DataEndpoint": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

DataEndpoint

エンドポイントの値。ストリームからデータを読み取ったり、ストリームにデータを書き込んだりするには、アプリケーションでこのエンドポイントを指定します。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

NotAuthorizedException

呼び出し元には、この操作を実行するための権限がありません。

HTTP ステータスコード: 401

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

GetSignalingChannelEndpoint

サービス : Amazon Kinesis Video Streams

指定されたシグナリングチャネルがメッセージを送受信するためのエンドポイントを提供します。このAPIは、`Protocols` と `Role` のプロパティで構成される `SingleMasterChannelEndpointConfiguration` 入力パラメーターを使用します。

`Protocols` は、通信メカニズムを決定するために使用されます。例えば、プロトコルとして `WSS` を指定すると、このAPIは安全な `WebSocket` エンドポイントを生成します。プロトコルとして `HTTPS` を指定すると、このAPIが `HTTPS` エンドポイントを生成します。プロトコル `WEBRTC` としてを指定しても、シグナリングチャネルが取り込み用に設定されていない場合は、エラーが表示されます `InvalidArgumentException`。

`Role` はメッセージングのアクセス許可を決定します。MASTER ロールにより、このAPIはエンドポイントを生成します。このエンドポイントは、クライアントがチャネル上の任意の閲覧者と通信するために使用できます。VIEWER ロールにより、このAPIはエンドポイントを生成します。このエンドポイントは、クライアントが MASTER とだけ通信するために使用できます。

リクエストの構文

```
POST /getSignalingChannelEndpoint HTTP/1.1
Content-type: application/json

{
  "ChannelARN": "string",
  "SingleMasterChannelEndpointConfiguration": {
    "Protocols": [ "string" ],
    "Role": "string"
  }
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

ChannelARN

エンドポイントを取得するシグナリングチャネルの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: はい

[SingleMasterChannelEndpointConfiguration](#)

SINGLE_MASTER チャネルタイプのエンドポイント設定を含む構造。

型: [SingleMasterChannelEndpointConfiguration](#) オブジェクト

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "ResourceEndpointList": [
    {
      "Protocol": "string",
      "ResourceEndpoint": "string"
    }
  ]
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[ResourceEndpointList](#)

指定されたシグナリングチャネルのエンドポイントのリスト。

型: [ResourceEndpointListItem](#) オブジェクトの配列

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceInUseException

の入力または StreamARNChannelARNCLOUD_STORAGE_MODEが既に別の Kinesis Video Stream リソースにマッピングされている場合、または指定された入力StreamARNまたは ChannelARNがアクティブステータスでない場合は、次のいずれかを試してください。

1. 指定されたチャンネルがマッピングされるストリームを決定する
DescribeMediaStorageConfiguration API。
2. 特定のストリームがマッピングされるチャンネルを決定する
DescribeMappedResourceConfiguration API。
3. リソースのステータスを決定する DescribeStreamまたは DescribeSignalingChannel API。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListEdgeAgentConfigurations

サービス : Amazon Kinesis Video Streams

指定されたエッジエージェントに関連付けられたエッジ設定の配列を返します。

リクエストでは、エッジエージェント を指定する必要がありますHubDeviceArn。

Note

この API は、AWS アフリカ (ケープタウン) リージョン af-south-1 では使用できません。

リクエストの構文

```
POST /listEdgeAgentConfigurations HTTP/1.1
Content-type: application/json
```

```
{
  "HubDeviceArn": "string",
  "MaxResults": number,
  "NextToken": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

HubDeviceArn

エッジエージェントの「モノのインターネット (IoT) モノ」ARN。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:iot:[a-z0-9-]+:[0-9]+:thing/[a-zA-Z0-9_.-]+`

必須: はい

MaxResults

レスポンスで返されるエッジ設定の最大数。デフォルトは 5 です。

タイプ: 整数

値の範囲: 最小値 は 1 です。最大値は 10 です。

必須: いいえ

NextToken

このパラメータを指定すると、ListEdgeAgentConfigurations 操作の結果が切り捨てられ、呼び出しはレスポンスで NextToken を返します。エッジ設定の別のバッチを取得するには、次のリクエストでこのトークンを指定します。

タイプ: 文字列

長さの制約: 最小長は 0 です。最大長は 1,024 です。

パターン: [a-zA-Z0-9+/=]*

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "EdgeConfigs": [
    {
      "CreationTime": number,
      "EdgeConfig": {
        "DeletionConfig": {
          "DeleteAfterUpload": boolean,
          "EdgeRetentionInHours": number,
          "LocalSizeConfig": {
            "MaxLocalMediaSizeInMB": number,
            "StrategyOnFullSize": "string"
          }
        },
        "HubDeviceArn": "string",
        "RecorderConfig": {
```

```

    "MediaSourceConfig": {
      "MediaUriSecretArn": "string",
      "MediaUriType": "string"
    },
    "ScheduleConfig": {
      "DurationInSeconds": number,
      "ScheduleExpression": "string"
    }
  },
  "UploaderConfig": {
    "ScheduleConfig": {
      "DurationInSeconds": number,
      "ScheduleExpression": "string"
    }
  },
  "FailedStatusDetails": "string",
  "LastUpdatedTime": number,
  "StreamARN": "string",
  "StreamName": "string",
  "SyncStatus": "string"
}
],
"NextToken": "string"
}

```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[EdgeConfigs](#)

単一ストリームのエッジ設定の説明。

型: [ListEdgeAgentConfigurationsEdgeConfig](#) オブジェクトの配列

[NextToken](#)

レスポンスが切り捨てられると、呼び出しは指定されたトークンでこの要素を返します。エッジ設定の次のバッチを取得するには、次のリクエストでこのトークンを使用します。

タイプ: 文字列

長さの制約: 最小長は 0 です。最大長は 1,024 です。

パターン: [a-zA-Z0-9+/=]*

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

NotAuthorizedException

呼び出し元には、この操作を実行するための権限がありません。

HTTP ステータスコード: 401

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListSignalingChannels

サービス : Amazon Kinesis Video Streams

ChannelInfo オブジェクトの配列を返します。各オブジェクトは、シグナリングチャンネルを記述します。特定の条件を満たすチャンネルだけを取得するには、ChannelNameCondition を指定できます。

リクエストの構文

```
POST /listSignalingChannels HTTP/1.1
Content-type: application/json
```

```
{
  "ChannelNameCondition": {
    "ComparisonOperator": "string",
    "ComparisonValue": "string"
  },
  "MaxResults": number,
  "NextToken": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[ChannelNameCondition](#)

オプション: 特定の条件を満たすチャンネルだけを返します。

型: [ChannelNameCondition](#) オブジェクト

必須: いいえ

[MaxResults](#)

レスポンスで返されるチャンネルの最大数。デフォルトは 500 です。

型: 整数

値の範囲: 最小値 は 1 です。最大値は 10,000 です。

必須: いいえ

NextToken

このパラメータを指定すると、ListSignalingChannels 操作の結果が切り捨てられ、呼び出しはレスポンスで NextToken を返します。チャンネルの別のバッチを取得するには、次のリクエストでこのトークンを指定します。

タイプ: 文字列

長さの制約: 最小長は 0 です。最大長は 1,024 です。

パターン: [a-zA-Z0-9+/=]*

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "ChannelInfoList": [
    {
      "ChannelARN": "string",
      "ChannelName": "string",
      "ChannelStatus": "string",
      "ChannelType": "string",
      "CreationTime": number,
      "SingleMasterConfiguration": {
        "MessageTtlSeconds": number
      },
      "Version": "string"
    }
  ],
  "NextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[ChannelInfoList](#)

ChannelInfo オブジェクトの配列。

型: [ChannelInfo](#) オブジェクトの配列

[NextToken](#)

レスポンスが切り捨てられた場合、呼び出しはトークンを含む要素を返します。ストリームの次のバッチを取得するには、次のリクエストでこのトークンを使用します。

タイプ: 文字列

長さの制約: 最小長は 0 です。最大長は 1,024 です。

パターン: [a-zA-Z0-9+/=]*

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListStreams

サービス : Amazon Kinesis Video Streams

StreamInfo オブジェクトの配列を返します。各オブジェクトがストリームを記述します。特定の条件を満たすストリームだけを取得するには、StreamNameCondition を指定します。

リクエストの構文

```
POST /listStreams HTTP/1.1
Content-type: application/json

{
  "MaxResults": number,
  "NextToken": "string",
  "StreamNameCondition": {
    "ComparisonOperator": "string",
    "ComparisonValue": "string"
  }
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

MaxResults

レスポンスに返されるストリームの最大数。デフォルトは 10,000 です。

型: 整数

値の範囲: 最小値 は 1 です。最大値は 10,000 です。

必須: いいえ

NextToken

このパラメータを指定すると、ListStreams 操作の結果が切り捨てられ、呼び出しはレスポンスで NextToken を返します。ストリームの別のバッチを取得するには、次のリクエストでこのトークンを指定します。

タイプ: 文字列

長さの制約: 最小長は 0 です。最大長は 1,024 です。

パターン: `[a-zA-Z0-9+/=]*`

必須: いいえ

StreamNameCondition

オプション: 特定の条件を満たすストリームだけを返します。現在、条件としてストリーム名のプレフィックスだけを指定できます。

型: StreamNameCondition オブジェクト

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "NextToken": "string",
  "StreamInfoList": [
    {
      "CreationTime": number,
      "DataRetentionInHours": number,
      "DeviceName": "string",
      "KmsKeyId": "string",
      "MediaType": "string",
      "Status": "string",
      "StreamARN": "string",
      "StreamName": "string",
      "Version": "string"
    }
  ]
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[NextToken](#)

レスポンスが切り捨てられた場合、呼び出しはトークンを含む要素を返します。ストリームの次のバッチを取得するには、次のリクエストでこのトークンを使用します。

タイプ: 文字列

長さの制約: 最小長は 0 です。最大長は 1,024 です。

パターン: [a-zA-Z0-9+/=]*

[StreamInfoList](#)

StreamInfo オブジェクトの配列。

型: [StreamInfo](#) オブジェクトの配列

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)

- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListTagsForResource

サービス : Amazon Kinesis Video Streams

指定されたシグナリングチャンネルに関連付けられたタグのリストを返します。

リクエストの構文

```
POST /ListTagsForResource HTTP/1.1
Content-type: application/json
```

```
{
  "NextToken": "string",
  "ResourceARN": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

NextToken

このパラメーターを指定し、ListTagsForResource の呼び出しの結果が切り捨てられた場合、レスポンスには、タグの次のバッチをフェッチするために次のリクエストで使用できるトークンが含まれます。

タイプ: 文字列

長さの制約: 最小長は 0 です。最大長は 1,024 です。

パターン: [a-zA-Z0-9+/=]*

必須: いいえ

ResourceARN

タグを一覧表示するシグナリングチャンネルの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: はい

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "NextToken": "string",
  "Tags": {
    "string" : "string"
  }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[NextToken](#)

このパラメーターを指定し、`ListTagsForResource` の呼び出しの結果が切り捨てられた場合、レスポンスには、タグの次のセットをフェッチするために次のリクエストで利用できるトークンが含まれます。

タイプ: 文字列

長さの制約: 最小長は 0 です。最大長は 1,024 です。

パターン: `[a-zA-Z0-9+/=]*`

[Tags](#)

指定されたシグナリングチャネルに関連付けられたタグキーと値のマップ。

型: 文字列間のマッピング

マップエントリ: 項目の最大数は 50 です。

キーの長さ制限: 最小長さは 1 です。最大長は 128 です。

キーパターン: `^([\p{L}\p{Z}\p{N}_.: /+=\-\@]*)$`

値の長さの制限: 最小長は 0 です。最大長は 256 です。

値のパターン: `[\p{L}\p{Z}\p{N}_.: /+=\-\@]*`

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)

- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListTagsForStream

サービス : Amazon Kinesis Video Streams

指定されたストリームに関連付けられているタグのリストを返します。

リクエストでは、StreamName または StreamARN のパラメータを指定する必要があります。

リクエストの構文

```
POST /listTagsForStream HTTP/1.1
Content-type: application/json
```

```
{
  "NextToken": "string",
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[NextToken](#)

このパラメーターを指定し、ListTagsForStream の呼び出しの結果が切り捨てられた場合、レスポンスには、タグの次のバッチをフェッチするために次のリクエストで利用できるトークンが含まれます。

タイプ: 文字列

長さの制約: 最小長は 0 です。最大長は 1,024 です。

パターン: [a-zA-Z0-9+/=]*

必須: いいえ

[StreamARN](#)

タグを一覧表示するストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

[StreamName](#)

タグをリスト表示するストリームの名前を指定します。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "NextToken": "string",
  "Tags": {
    "string" : "string"
  }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[NextToken](#)

このパラメーターを指定し、ListTags の呼び出しの結果が切り捨てられた場合、レスポンスには、タグの次のセットをフェッチするために次のリクエストで利用できるトークンが含まれます。

タイプ: 文字列

長さの制約: 最小長は 0 です。最大長は 1,024 です。

パターン: [a-zA-Z0-9+/=]*

Tags

指定されたストリームに関連付けられたタグキーと値のマップ。

タイプ: 文字列間のマッピング

マップエントリ: 項目の最大数は 50 です。

キーの長さ制限: 最小長さは 1 です。最大長は 128 です。

キーパターン: ^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)\$

値の長さの制限: 最小長は 0 です。最大長は 256 です。

値のパターン: [\p{L}\p{Z}\p{N}_.:/=+\-@]*

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

InvalidResourceFormatException

StreamARN の形式が無効です。

HTTP ステータスコード: 400

NotAuthorizedException

呼び出し元には、この操作を実行するための権限がありません。

HTTP ステータスコード: 401

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

StartEdgeConfigurationUpdate

サービス : Amazon Kinesis Video Streams

ストリームの既存のエッジ設定を更新する非同期 API。Kinesis Video Stream は、ストリームのエッジ設定を、オンプレミスでセットアップされた IoT Hub Device で実行される Edge Agent IoT Greengrass コンポーネントと同期します。同期時間は、Hub デバイスの接続によって異なります。SyncStatus エッジ設定が確認され、エッジエージェントと同期されると、 が更新されます。

この API が初めて呼び出されると、ストリームの新しいエッジ設定が作成され、同期ステータスが に設定されます SYNCING。同期ステータスが IN_SYNC、 などの終了状態になるまで待ってから SYNC_FAILED、この API を再度使用する必要があります。同期プロセス中にこの API を呼び出すと、ResourceInUseException がスローされます。ストリームのエッジ設定とエッジエージェントの接続は 15 分間再試行されます。15 分後、ステータスは SYNC_FAILED 状態に移行します。

エッジ設定をあるデバイスから別のデバイスに移動するには、 [DeleteEdgeConfiguration](#) を使用して現在のエッジ設定を削除します。その後、更新された Hub Device ARN を使用して StartEdgeConfigurationUpdate を呼び出すことができます。

Note

この API は、AWS アフリカ (ケープタウン) リージョン af-south-1 では利用できません。

リクエストの構文

```
POST /startEdgeConfigurationUpdate HTTP/1.1
Content-type: application/json
```

```
{
  "EdgeConfig": {
    "DeletionConfig": {
      "DeleteAfterUpload": boolean,
      "EdgeRetentionInHours": number,
      "LocalSizeConfig": {
        "MaxLocalMediaSizeInMB": number,
        "StrategyOnFullSize": "string"
      }
    },
    "HubDeviceArn": "string",
    "RecorderConfig": {
      "MediaSourceConfig": {
```

```

    "MediaUriSecretArn": "string",
    "MediaUriType": "string"
  },
  "ScheduleConfig": {
    "DurationInSeconds": number,
    "ScheduleExpression": "string"
  }
},
"UploaderConfig": {
  "ScheduleConfig": {
    "DurationInSeconds": number,
    "ScheduleExpression": "string"
  }
}
},
"StreamARN": "string",
"StreamName": "string"
}

```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[EdgeConfig](#)

更新プロセスを呼び出すために必要なエッジ設定の詳細。

型: [EdgeConfig](#) オブジェクト

必須: はい

[StreamARN](#)

ストリームの Amazon リソースネーム (ARN)。StreamName または を指定しますStreamARN。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

エッジ設定を更新するストリームの名前。StreamName または を指定しますStreamARN。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "CreationTime": number,
  "EdgeConfig": {
    "DeletionConfig": {
      "DeleteAfterUpload": boolean,
      "EdgeRetentionInHours": number,
      "LocalSizeConfig": {
        "MaxLocalMediaSizeInMB": number,
        "StrategyOnFullSize": "string"
      }
    },
    "HubDeviceArn": "string",
    "RecorderConfig": {
      "MediaSourceConfig": {
        "MediaUriSecretArn": "string",
        "MediaUriType": "string"
      },
      "ScheduleConfig": {
        "DurationInSeconds": number,
        "ScheduleExpression": "string"
      }
    },
    "UploaderConfig": {
      "ScheduleConfig": {
        "DurationInSeconds": number,
```

```
        "ScheduleExpression": "string"
    }
}
},
"FailedStatusDetails": "string",
"LastUpdateTime": number,
"StreamARN": "string",
"StreamName": "string",
"SyncStatus": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[CreationTime](#)

ストリームのエッジ設定が最初に作成されたタイムスタンプ。

タイプ: タイムスタンプ

[EdgeConfig](#)

Edge Agent IoT Greengrass コンポーネントとの同期に使用されるストリームのエッジ設定の説明。Edge Agent コンポーネントは、オンプレミスの IoT Hub デバイスのセットアップで実行されます。

型: [EdgeConfig](#) オブジェクト

[FailedStatusDetails](#)

生成された障害ステータスの説明。

タイプ: 文字列

[LastUpdateTime](#)

ストリームのエッジ設定が最後に更新されたタイムスタンプ。

タイプ: タイムスタンプ

[StreamARN](#)

ストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

StreamName

エッジ設定が更新されたストリームの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

SyncStatus

ストリームのエッジ設定の現在の同期ステータス。この API を呼び出すと、同期ステータスは SYNCING 状態に設定されます。DescribeEdgeConfiguration API を使用して、エッジ設定の最新のステータスを取得します。

型: 文字列

有効な値 : SYNCING | ACKNOWLEDGED | IN_SYNC | SYNC_FAILED | DELETING | DELETE_FAILED | DELETING_ACKNOWLEDGED

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

NoDataRetentionException

時間単位のストリームデータ保持は 0 に等しくなります。

HTTP ステータスコード: 400

ResourceInUseException

の入力または StreamARNChannelARNCLOUD_STORAGE_MODEが既に別の Kinesis Video Stream リソースにマッピングされている場合、または指定された入力StreamARNまたは ChannelARNがアクティブステータスでない場合は、次のいずれかを試してください。

1. 指定されたチャンネルがマッピングされるストリームを決定する
DescribeMediaStorageConfiguration API。
2. 特定のストリームがマッピングされるチャンネルを決定する
DescribeMappedResourceConfiguration API。
3. リソースのステータスを決定する DescribeStreamまたは DescribeSignalingChannel API。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

TagResource

サービス : Amazon Kinesis Video Streams

シグナリングチャンネルにのみ 1 つ以上のタグを追加します。注: ストリームにタグを付けるには、代わりに TagStream API を使用します。タグは、AWS リソースを定義して割り当てることができるキーと値のペア (値はオプション) です。すでに存在するタグを指定すると、タグの値はこのリクエストで指定した値に置き換えられます。詳細については、『』の「[コスト配分タグの使用](#) AWS Billing and Cost Management 」および「コスト管理ユーザーガイド」を参照してください。

リクエストの構文

```
POST /TagResource HTTP/1.1
Content-type: application/json

{
  "ResourceARN": "string",
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ]
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

ResourceARN

タグを追加するシグナリングチャンネルの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: はい

Tags

指定されたシグナリングチャンネルに関連付けるタグのリスト。各タグはキーバリューのペアです。

型: [Tag](#) オブジェクトの配列

配列メンバー: 最小数は 1 項目です。最大数は 50 項目です。

必須: はい

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

TagsPerResourceExceededLimitException

リソースに関連付けることができるタグの上限を超えています。Kinesis ビデオストリームは、最大 50 個のタグをサポートできます。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

TagStream

サービス : Amazon Kinesis Video Streams

1 つまたは複数のタグをストリームに追加します。タグは、AWS リソースを定義して割り当てることのできるキーと値のペア (値はオプション) です。すでに存在するタグを指定すると、タグの値はこのリクエストで指定した値に置き換えられます。詳細については、『』の「[コスト配分タグの使用](#)」AWS Billing and Cost Management」および「コスト管理ユーザーガイド」を参照してください。

StreamName または StreamARN を指定する必要があります。

この操作には KinesisVideo:TagStream アクションに対するアクセス許可が必要です。

Kinesis ビデオストリームは、最大 50 個のタグをサポートできます。

リクエストの構文

```
POST /tagStream HTTP/1.1
Content-type: application/json

{
  "StreamARN": "string",
  "StreamName": "string",
  "Tags": {
    "string" : "string"
  }
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[StreamARN](#)

1 つまたは複数のタグを追加するリソースの Amazon リソースネーム (ARN) です。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

1 つまたは複数のタグを追加するストリームの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: いいえ

Tags

指定されたストリームに関連付けるタグのリスト。各タグはキーと値のペアです (値はオプションです)。

タイプ: 文字列間のマッピング

マップエントリ: 項目の最大数は 50 です。

キーの長さ制限: 最小長さは 1 です。最大長は 128 です。

キーパターン: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

値の長さの制限: 最小長は 0 です。最大長は 256 です。

値のパターン: `[\p{L}\p{Z}\p{N}_.:/=+\-@]*`

必須: はい

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

InvalidResourceFormatException

StreamARN の形式が無効です。

HTTP ステータスコード: 400

NotAuthorizedException

呼び出し元には、この操作を実行するための権限がありません。

HTTP ステータスコード: 401

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

TagsPerResourceExceededLimitException

リソースに関連付けることができるタグの上限を超えています。Kinesis ビデオストリームは、最大 50 個のタグをサポートできます。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UntagResource

サービス : Amazon Kinesis Video Streams

シグナリングチャネルからのみ 1 つ以上のタグを削除します。注: ストリームからタグを削除するには、代わりに UntagStream API を使用します。リクエストでは、1 つまたは複数のタグキーだけを指定します。値は指定しません。存在しないタグキーを指定した場合、そのキーは無視されます。

リクエストの構文

```
POST /UntagResource HTTP/1.1
Content-type: application/json

{
  "ResourceARN": "string",
  "TagKeyList": [ "string" ]
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

ResourceARN

タグを削除するシグナリングチャネルの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: はい

TagKeyList

削除するタグのキーのリスト。

型: 文字列の配列

配列メンバー: 最小数は 1 項目です。最大数は 50 項目です。

長さの制限: 最小長 1、最大長は 128 です。

パターン: `^([\p{L}\p{Z}\p{N}_.:/+\\-@]*)$`

必須: はい

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UntagStream

サービス : Amazon Kinesis Video Streams

ストリームから 1 つまたは複数のタグを削除します。リクエストでは、1 つまたは複数のタグキーだけを指定します。値は指定しません。存在しないタグキーを指定した場合、そのキーは無視されます。

リクエストでは、StreamName または StreamARN を入力する必要があります。

リクエストの構文

```
POST /untagStream HTTP/1.1
Content-type: application/json

{
  "StreamARN": "string",
  "StreamName": "string",
  "TagKeyList": [ "string" ]
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

StreamARN

タグを削除するストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

タグを削除するクラスターの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

TagKeyList

削除するタグのキーのリスト。

型: 文字列の配列

配列メンバー: 最小数は 1 項目です。最大数は 50 項目です。

長さの制限: 最小長 1、最大長は 128 です。

パターン: ^([\p{L}\p{Z}\p{N}_.:/+\\-@]*)\$

必須: はい

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

InvalidResourceFormatException

StreamARN の形式が無効です。

HTTP ステータスコード: 400

NotAuthorizedException

呼び出し元には、この操作を実行するための権限がありません。

HTTP ステータスコード: 401

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateDataRetention

サービス : Amazon Kinesis Video Streams

ストリームのデータ保持期間を指定した値で増減します。データ保持期間を延長するか短縮するかを指定するには、リクエスト本文の `Operation` パラメータを指定します。リクエストでは、`StreamName` または `StreamARN` のパラメータを指定する必要があります。

この操作には `KinesisVideo:UpdateDataRetention` アクションに対するアクセス許可が必要です。

データ保持期間を変更すると、ストリーム内のデータに次のように影響します。

- データ保持期間を延長すると、既存のデータは新しい保持期間で保持されます。例えば、データ保持期間を 1 時間から 7 時間に延長すると、既存のすべてのデータが 7 時間保持されます。
- データ保持期間が短縮されると、既存のデータは新しい保存期間にわたって保持されます。例えば、データ保持期間が 7 時間から 1 時間に短縮すると、既存のすべてのデータが 1 時間保持され、1 時間より古いデータは直ちに削除されます。

リクエストの構文

```
POST /updateDataRetention HTTP/1.1
Content-type: application/json

{
  "CurrentVersion": "string",
  "DataRetentionChangeInHours": number,
  "Operation": "string",
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

CurrentVersion

保持期間を変更するストリームのバージョン。バージョンを取得するには、DescribeStream または ListStreams API を呼び出します。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長 64

パターン: [a-zA-Z0-9]+

必須: はい

DataRetentionChangeInHours

現在の保持期間を調整する時間数。指定した値は、 に応じて、現在の値に加算または減算されます operation。

データ保持の最小値は 0 で、最大値は 87600 (10 年) です。

タイプ: 整数

値の範囲: 最小値 は 1 です。

必須: はい

Operation

保持期間を増減させるかどうかを示します。

型: 文字列

有効な値 : INCREASE_DATA_RETENTION | DECREASE_DATA_RETENTION

必須: はい

StreamARN

保持期間を変更するストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+

必須: いいえ

StreamName

保持期間を変更するストリームの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

NotAuthorizedException

呼び出し元には、この操作を実行するための権限がありません。

HTTP ステータスコード: 401

ResourceInUseException

入力 StreamARN または ChannelARN が既に別の Kinesis Video Stream リソースに CLOUD_STORAGE_MODE マッピングされている場合、または指定された入力 StreamARN または ChannelARN がアクティブステータスでない場合は、次のいずれかを試してください。

1. 指定されたチャンネルがマッピングされるストリームを決定する
DescribeMediaStorageConfiguration API。
2. 特定のストリームがマッピングされるチャンネルを決定する
DescribeMappedResourceConfiguration API。
3. リソースのステータスを決定する DescribeStream または DescribeSignalingChannel API。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

VersionMismatchException

指定したストリームバージョンは最新バージョンではありません。最新バージョンを入手するには、[DescribeStream](#) API を使用してください。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)

- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateImageGenerationConfiguration

サービス : Amazon Kinesis Video Streams

StreamInfo および ImageProcessingConfiguration フィールドを更新します。

リクエストの構文

```
POST /updateImageGenerationConfiguration HTTP/1.1
Content-type: application/json
```

```
{
  "ImageGenerationConfiguration": {
    "DestinationConfig": {
      "DestinationRegion": "string",
      "Uri": "string"
    },
    "Format": "string",
    "FormatConfig": {
      "string" : "string"
    },
    "HeightPixels": number,
    "ImageSelectorType": "string",
    "SamplingInterval": number,
    "Status": "string",
    "WidthPixels": number
  },
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[ImageGenerationConfiguration](#)

KVS イメージの配信に必要な情報を含む構造。構造が null の場合、設定はストリームから削除されます。

型: [ImageGenerationConfiguration](#) オブジェクト

必須: いいえ

[StreamARN](#)

イメージ生成設定を更新する Kinesis ビデオストリームの Amazon リソースネーム (ARN)。StreamName または StreamARN のパラメータを指定する必要があります。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

[StreamName](#)

イメージ生成設定を更新するストリームの名前。StreamName または StreamARN のパラメータを指定する必要があります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

NoDataRetentionException

時間単位のストリームデータ保持は 0 に等しくなります。

HTTP ステータスコード: 400

ResourceInUseException

入力 StreamARN または ChannelARN が既に別の Kinesis Video Stream リソースに CLOUD_STORAGE_MODE マッピングされている場合、または指定された入力 StreamARN または ChannelARN がアクティブステータスでない場合は、次のいずれかを試してください。

1. 指定されたチャネルがマッピングされるストリームを決定する
DescribeMediaStorageConfiguration API。
2. 特定のストリームがマッピングされるチャネルを決定する
DescribeMappedResourceConfiguration API。
3. リソースのステータスを決定する DescribeStream または DescribeSignalingChannel API。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateMediaStorageConfiguration

サービス : Amazon Kinesis Video Streams

をストリームSignalingChannelに関連付けて、メディアを保存します。を指定できるシグナリングモードは 2 つあります。

- StorageStatus が有効になっている場合、データはStreamARN提供された に保存されます。WebRTC Ingestion を機能させるには、ストリームでデータ保持が有効になっている必要があります。
- StorageStatus が無効になっている場合、データは保存されず、StreamARN/パラメータは必要ありません。

Important

StorageStatus が有効になっている場合、peer-to-peer (マスタービューワー) 接続は行われなくなります。ピアはストレージセッションに直接接続します。JoinStorageSession API を呼び出して SDP オファァの送信をトリガーし、ピアとストレージセッション間の接続を確立する必要があります。

リクエストの構文

```
POST /updateMediaStorageConfiguration HTTP/1.1
```

```
Content-type: application/json
```

```
{
  "ChannelARN": "string",
  "MediaStorageConfiguration": {
    "Status": "string",
    "StreamARN": "string"
  }
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[ChannelARN](#)

チャネルの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: はい

[MediaStorageConfiguration](#)

メディアストレージ設定プロパティをカプセル化または含む構造。

型: [MediaStorageConfiguration](#) オブジェクト

必須: はい

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

NoDataRetentionException

時間単位のストリームデータ保持は 0 に等しくなります。

HTTP ステータスコード: 400

ResourceInUseException

入力 StreamARN または ChannelARN の CLOUD_STORAGE_MODE が既に別の Kinesis Video Stream リソースにマッピングされている場合、または指定された入力 StreamARN または ChannelARN がアクティブステータスでない場合は、次のいずれかを試してください。

1. 指定されたチャンネルがマッピングされるストリームを決定する
DescribeMediaStorageConfiguration API。
2. 特定のストリームがマッピングされるチャンネルを決定する
DescribeMappedResourceConfiguration API。
3. リソースのステータスを決定する DescribeStream または DescribeSignalingChannel API。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)

- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateNotificationConfiguration

サービス : Amazon Kinesis Video Streams

ストリームの通知情報を更新します。

リクエストの構文

```
POST /updateNotificationConfiguration HTTP/1.1
Content-type: application/json
```

```
{
  "NotificationConfiguration": {
    "DestinationConfig": {
      "Uri": "string"
    },
    "Status": "string"
  },
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[NotificationConfiguration](#)

通知に必要な情報を含む構造。構造が null の場合、設定はストリームから削除されます。

型: [NotificationConfiguration](#) オブジェクト

必須: いいえ

[StreamARN](#)

通知設定を更新する Kinesis ビデオストリームの Amazon リソースネーム (ARN)。StreamName または StreamARN のパラメータを指定する必要があります。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

通知設定を更新するストリームの名前。StreamName または StreamARN のパラメータを指定する必要があります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

NoDataRetentionException

時間単位のストリームデータ保持は 0 に等しくなります。

HTTP ステータスコード: 400

ResourceInUseException

の入力または StreamARNChannelARNCLOUD_STORAGE_MODEが既に別の Kinesis Video Stream リソースにマッピングされている場合、または指定された入力StreamARNまたは ChannelARNがアクティブステータスでない場合は、次のいずれかを試してください。

1. 指定されたチャンネルがマッピングされるストリームを決定する
DescribeMediaStorageConfiguration API。
2. 特定のストリームがマッピングされるチャンネルを決定する
DescribeMappedResourceConfiguration API。
3. リソースのステータスを決定する DescribeStreamまたは DescribeSignalingChannel API。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateSignalingChannel

サービス : Amazon Kinesis Video Streams

既存のシグナリングチャネルを更新します。これは非同期の操作であり、完了するまでに時間がかかります。

MessageTtlSeconds の値が更新された場合 (増加または減少)、更新後にこのチャネルを介して送信された新しいメッセージだけに適用されます。以前の MessageTtlSeconds の値の通り、既存のメッセージはまだ期限切れになっています。

リクエストの構文

```
POST /updateSignalingChannel HTTP/1.1
Content-type: application/json

{
  "ChannelARN": "string",
  "CurrentVersion": "string",
  "SingleMasterConfiguration": {
    "MessageTtlSeconds": number
  }
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

ChannelARN

更新するシグナリングチャネルの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: はい

[CurrentVersion](#)

更新するシグナリングチャネルの現在のバージョン。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長 64

パターン: [a-zA-Z0-9]+

必須: はい

[SingleMasterConfiguration](#)

更新するシグナリングチャネルの SINGLE_MASTER 型の設定を含む構造。このパラメータとチャネルメッセージのtime-to-liveは、チャネルタイプのSINGLE_MASTERチャネルに必要です。

型: [SingleMasterConfiguration](#) オブジェクト

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceInUseException

の入力または StreamARNChannelARNCLOUD_STORAGE_MODEが既に別の Kinesis Video Stream リソースにマッピングされている場合、または指定された入力StreamARNまたは ChannelARNがアクティブステータスでない場合は、次のいずれかを試してください。

1. 指定されたチャンネルがマッピングされるストリームを決定する
DescribeMediaStorageConfiguration API。
2. 特定のストリームがマッピングされるチャンネルを決定する
DescribeMappedResourceConfiguration API。
3. リソースのステータスを決定する DescribeStreamまたは DescribeSignalingChannel API。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

VersionMismatchException

指定したストリームバージョンは最新バージョンではありません。最新バージョンを入手するには、[DescribeStream](#) API を使用してください。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)

- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateStream

サービス : Amazon Kinesis Video Streams

デバイス名やメディアタイプなどのストリームメタデータを更新します。

ストリーム名またはストリームの Amazon リソースネーム (ARN) を指定する必要があります。

ストリームを更新する前に最新バージョンであることを確保するために、ストリームのバージョンを指定できます。Kinesis Video Streams が各ストリームにバージョンを割り当てます。ストリームを更新すると、Kinesis Video Streams が新しいバージョン番号を割り当てます。最新のストリームバージョンを取得するには、DescribeStream APIを使用します

UpdateStream は非同期の操作で、完了するまでに時間がかかります。

リクエストの構文

```
POST /updateStream HTTP/1.1
Content-type: application/json

{
  "CurrentVersion": "string",
  "DeviceName": "string",
  "MediaType": "string",
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

CurrentVersion

メタデータを更新するストリームのバージョン。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長 64

パターン: [a-zA-Z0-9]+

必須: はい

DeviceName

ストリームに書き込んでいるデバイスの名前。

Note

現在の実装では、Kinesis Video Streams はこの名前を使用しません。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 128 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

MediaType

ストリームのメディアタイプ。MediaType を使用して、ストリームに含まれるコンテンツのタイプをストリームのコンシューマーに指定します。メディアタイプの詳細については、「[メディアタイプ](#)」を参照してください。MediaType を指定する場合は、「[命名要件](#)」を参照してください。

コンソールで動画を再生するには、正しい動画タイプを指定してください。例えば、ストリーム内のビデオが H.264 の場合は、MediaType として「video/h264」を指定します。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 128 です。

パターン: [\w\-\.\+]+/[\w\-\.\+]+([\w\-\.\+]+/[\w\-\.\+]+)*

必須: いいえ

StreamARN

メタデータを更新するストリームの ARN。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

メタデータを更新するストリームの名前。

ストリーム名はストリームの識別子であり、アカウントやリージョンごとに一意である必要があります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

NotAuthorizedException

呼び出し元には、この操作を実行するための権限がありません。

HTTP ステータスコード: 401

ResourceInUseException

入力 StreamARN または ChannelARN が既に別の Kinesis Video Stream リソースに CLOUD_STORAGE_MODE マッピングされている場合、または指定された入力 StreamARN または ChannelARN がアクティブステータスでない場合は、次のいずれかを試してください。

1. 指定されたチャンネルがマッピングされるストリームを決定する
DescribeMediaStorageConfiguration API。
2. 特定のストリームがマッピングされるチャンネルを決定する
DescribeMappedResourceConfiguration API。
3. リソースのステータスを決定する DescribeStream または DescribeSignalingChannel API。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

VersionMismatchException

指定したストリームバージョンは最新バージョンではありません。最新バージョンを入手するには、[DescribeStream](#) API を使用してください。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)

- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateStreamStorageConfiguration

サービス : Amazon Kinesis Video Streams

既存の Kinesis ビデオストリームのストレージ設定を更新します。

このオペレーションでは、ストリームのストレージ階層設定を変更できるため、アクセスパターンに基づいてストレージコストとパフォーマンスを最適化できます。

UpdateStreamStorageConfiguration は非同期オペレーションです。

KinesisVideo:UpdateStreamStorageConfiguration アクションのアクセス許可が必要です。

リクエストの構文

```
POST /updateStreamStorageConfiguration HTTP/1.1
Content-type: application/json
```

```
{
  "CurrentVersion": "string",
  "StreamARN": "string",
  "StreamName": "string",
  "StreamStorageConfiguration": {
    "DefaultStorageTier": "string"
  }
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

CurrentVersion

ストレージ設定を変更するストリームのバージョン。バージョンを取得するには、DescribeStream または ListStreams API を呼び出します。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長 64

パターン: [a-zA-Z0-9]+

必須: はい

[StreamARN](#)

ストレージ設定を更新するストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+

必須: いいえ

[StreamName](#)

ストレージ設定を更新するストリームの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

[StreamStorageConfiguration](#)

ストリームの新しいストレージ設定。これには、ストリームデータの保存方法とアクセス方法を決定するデフォルトのストレージ層が含まれます。

ストレージ階層が異なると、特定のユースケースの要件に合わせてさまざまなレベルのパフォーマンスとコストの最適化が提供されます。

型: [StreamStorageConfiguration](#) オブジェクト

必須: はい

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 401

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceInUseException

入力 StreamARN または ChannelARN の CLOUD_STORAGE_MODE が既に別の Kinesis Video Stream リソースにマッピングされている場合、または指定された入力 StreamARN または ChannelARN がアクティブステータスでない場合は、次のいずれかを試してください。

1. 指定されたチャンネルがマッピングされるストリームを決定する
DescribeMediaStorageConfiguration API。
2. 特定のストリームがマッピングされるチャンネルを決定する
DescribeMappedResourceConfiguration API。
3. リソースのステータスを決定する DescribeStream または DescribeSignalingChannel API。

HTTP ステータスコード: 400

ResourceNotFoundException

Amazon Kinesis Video Streams は、指定したストリームを見つけることができません。

HTTP ステータスコード: 404

VersionMismatchException

指定したストリームバージョンは最新バージョンではありません。最新バージョンを入手するには、[DescribeStream](#) API を使用してください。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

Amazon Kinesis Video Streams Media

以下のアクションは、Amazon Kinesis Video Streams Media でサポートされています。

- [GetMedia](#)
- [PutMedia](#)

GetMedia

サービス : Amazon Kinesis Video Streams Media

この API を使用して、Kinesis ビデオストリームからメディアコンテンツを取得します。リクエストで、ストリーム名またはストリーム Amazon リソースネーム (ARN) と開始チャンクを特定します。Kinesis Video Streams は、フラグメント番号順にチャンクのストリームを返します。

Note

エンドポイントを取得するには、最初に GetDataEndpoint API を呼び出す必要があります。次に、[--endpoint-url parameter](#) を使用して GetMedia リクエストをこのエンドポイントに送信します。

ストリームにメディアデータ (フラグメント) を配置すると、Kinesis Video Streams は、受信する各フラグメントと関連するメタデータを「チャンク」と呼ばれるものに格納します。詳細については、「[PutMedia](#)」を参照してください。GetMedia API は、リクエストで指定されたチャンクから始まるこれらのチャンクのストリームを返します。

GetMedia API は、長時間実行される接続を介したストリーミング API として動作するように設計されています。これは、フラグメントごとに新しい HTTP 接続が確立されて閉じられる従来の RESTful 方式で使用することを目的としたものではありません。GetMedia API を呼び出すと、Kinesis Video Streams は HTTP チャンク転送エンコーディングを使用した永続的な長時間実行接続を通じてフラグメントを継続的に配信します。

GetMedia API を使用するときには以下の制限が適用されます。

- クライアントは、ストリームごとに 1 秒間に最大 5 回 GetMedia を呼び出すことができます。
- Kinesis Video Streams は、GetMedia セッション中に最大 25 メガバイト/秒 (200 メガビット/秒) の速度でメディアデータを送信します。

Note

ストリーミングの長時間実行接続 GetMedia として使用して、1 つの永続接続で複数のフラグメントを取得します。フラグメントごとに新しい HTTP 接続を確立して閉じる従来の RESTful 方式で GetMedia API を使用しないでください。3 つ以上の同時 GetMedia 接続を試みると、Kinesis Video Streams は最新の接続を `ConnectionLimitExceededException` エラーでスロットリングします。

Note

GetMedia HTTP レスポンスステータスコードはすぐに返されますが、取り込まれたフラグメントが再生できない場合、HTTP レスポンスペイロードの読み取りは 3 秒後にタイムアウトします。

Note

Kinesis Video Streams メディア API を呼び出した後にエラーがスローされた場合、HTTP ステータスコードとレスポンス本文に加えて、次の情報が含まれます。

- `x-amz-ErrorType` HTTP ヘッダー — HTTP ステータスコードで提供されるものに加えて、より具体的なエラータイプが含まれます。
- `x-amz-RequestId` HTTP ヘッダー – 問題をレポートする場合 AWS、リクエスト ID を指定すれば、サポートチームが問題をより適切に診断できます。

HTTP ステータスコードと `ErrorType` ヘッダーの両方を使用すれば、エラーが再試行可能かどうか、またはどのような条件下でエラーが再試行可能かについてプログラムで判断したり、クライアントプログラマーが再度試行するために必要なアクションに関する情報を提供したりできます。

詳細については、このトピックの下部にある[Errors] (エラー) セクションおよび「[Common Errors](#)」を参照してください。

リクエストの構文

```
POST /getMedia HTTP/1.1
Content-type: application/json
```

```
{
  "StartSelector": {
    "AfterFragmentNumber": "string",
    "ContinuationToken": "string",
    "StartSelectorType": "string",
    "StartTimestamp": number
  },
  "StreamARN": "string",
  "StreamName": "string"
```

```
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[StartSelector](#)

指定されたストリームから取得する開始チャンクを特定します。

型: [StartSelector](#) オブジェクト

必須: はい

[StreamARN](#)

メディアコンテンツの取得元からのストリームの ARN。streamARN を指定しない場合は、streamName を指定する必要があります。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

[StreamName](#)

メディアコンテンツの取得元からの Kinesis ビデオストリーム名。streamName を指定しない場合は、streamARN を指定する必要があります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-Type: ContentType
```

Payload

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

レスポンスでは、以下の HTTP ヘッダーが返されます。

ContentType

リクエストされたメディアのコンテンツタイプ

長さの制限：最小長 1、最大長は 128 です。

パターン: `^[a-zA-Z0-9_\.\\-]+$`

レスポンスは、HTTP 本文として以下を返します。

Payload

Kinesis Video Streams が返すペイロードは、指定されたストリームからのチャンクのシーケンスです。チャンクの詳細については、「[PutMedia](#)」を参照してください。Kinesis Video Streams が GetMedia の呼び出しで返すチャンクには、次の追加の Matroska (MKV) タグも含まれます。

- AWS_KINESISVIDEO_CONTINUATION_TOKEN (UTF-8 文字列) - GetMedia の呼び出しが終了した場合、次のリクエストでこの継続トークンを使用して、最後のリクエストが終了した次のチャンクを取得できます。
- AWS_KINESISVIDEO_MILIS_BEHIND_NOW (UTF-8 文字列) - クライアントアプリケーションはこのタグ値を使用して、レスポンスで返されるチャンクがストリームの最新のチャンクからどのくらい後ろにあるかを判断できます。
- AWS_KINESISVIDEO_FRAGMENT_NUMBER - チャンクで返されるフラグメント番号。
- AWS_KINESISVIDEO_SERVER_TIMESTAMP - フラグメントのサーバーのタイムスタンプ。
- AWS_KINESISVIDEO_PRODUCER_TIMESTAMP - フラグメントのプロジューサーのタイムスタンプ。

エラーが発生すると、次のタグが表示されます。

- AWS_KINESISVIDEO_ERROR_CODE - GetMedia を停止させたエラーの文字列の説明。
- AWS_KINESISVIDEO_ERROR_ID: エラーの整数コード。

エラーコードは次のとおりです。

- 3002 - Error writing to the stream (ストリームへの書き込みエラー)
- 4000 - Requested fragment is not found (要求されたフラグメントが見つかりません)
- 4500 - Access denied for the stream's KMS key (ストリームの KMS キーに対するアクセスが拒否されました)
- 4501 - Stream's KMS key is disabled (ストリームの KMS キーが無効)
- 4502 - Validation error on the stream's KMS key (ストリームの KMS キーの検証エラー)
- 4503 - KMS key specified in the stream is unavailable (ストリームで指定された KMS キーが使用できません)
- 4504 - Invalid usage of the KMS key specified in the stream (ストリームで指定された KMS キーの使用が無効)
- 4505 - Invalid state of the KMS key specified in the stream (ストリームで指定された KMS キーが無効な状態)
- 4506 - Unable to find the KMS key specified in the stream (ストリームで指定された KMS キーが見つかりません)
- 5000 - Internal error (内部エラー)

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

ConnectionLimitExceededException

許可されたクライアント接続の制限を超えたため、Kinesis Video Streams がリクエストをスロットリングしました。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

InvalidEndpointException

呼び出し元が間違ったエンドポイントを使用してデータをストリームに書き込みました。このような例外を受信すると、ユーザーは APIName を PUT_MEDIA に設定して GetDataEndpoint を呼び出し、応答からのエンドポイントを使用して次の PutMedia コールを呼び出す必要があります。

HTTP ステータスコード: 400

NotAuthorizedException

呼び出し元は、指定されたストリームで操作を実行する権限がないか、トークンの有効期限が切れています。

HTTP ステータスコード: 401

ResourceNotFoundException

ステータスコード: 404 指定された名前のストリームは存在しません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)

- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

PutMedia

サービス : Amazon Kinesis Video Streams Media

この API を使用して、メディアデータを Kinesis ビデオストリームに送信します。

Note

エンドポイントを取得するには、最初に GetDataEndpoint API を呼び出す必要があります。次に、[--endpoint-url parameter](#) を使用して PutMedia リクエストをこのエンドポイントに送信します。

このリクエストでは、HTTP ヘッダーを使用して、ストリーム名、タイムスタンプ、タイムスタンプ値が絶対値あるいはプロデューサが記録を開始した時点からの相対値であるかなどのパラメータ情報を提供します。リクエスト本文を使用してメディアデータを送信します。Kinesis Video Streams では、この API を使用してメディアデータを送信するために Matroska (MKV) コンテナ形式のみがサポートされています。

この API を使用してデータを送信するには、次のオプションがあります。

- メディアデータをリアルタイムで送信する：例えば、セキュリティカメラは、フレームを生成するときにリアルタイムでフレームを送信できます。このアプローチにより、動画録画とネットワーク上で送信されるデータ間のレイテンシーが最小限に抑えられます。これを連続プロデューサーと呼びます。この場合、コンシューマアプリケーションは、リアルタイムで、または必要に応じてストリームを読み込むことができます。
- メディアデータをオフライン（バッチ処理）で送信：例えば、ボディカメラが動画を数時間録画してデバイスに保存する場合があります。後でカメラをドッキングポートに接続すると、カメラは PutMedia セッションを開始して、Kinesis ビデオストリームにデータを送信できます。このシナリオでは、レイテンシーは問題ではありません。

API を使用する場合は、次の考慮事項に注意してください。

- streamName または streamARN のパラメータを指定する必要があります。両方を指定することはできません。
- コンソールまたは HLS を介してメディアを再生できるようにするには、各フラグメントのトラック 1 に h.264 エンコードされたビデオが含まれている必要があり、フラグメントメタデータの CodecID は「V_MPEG/ISO/AVC」である必要があり、更にフラグメントメタデータには、AVCC

形式の h.264 コーデックプライベートデータが含まれている必要があります。オプションで、各フラグメントのトラック 2 には AAC でエンコードされたオーディオが含まれ、フラグメントメタデータの CodecID は「A_AAC」で、フラグメントメタデータには AAC コーデックのプライベートデータが含まれている必要があります。

- PutMedia API は、長時間実行される接続を介したストリーミング API として動作するように設計されています。これは、フラグメントごとに新しい HTTP 接続が確立されて閉じられる従来の RESTful 方式での使用を意図していません。PutMedia API を使用する場合は、HTTP チャンク転送エンコーディングを使用して、永続接続を介してフラグメントを継続的に送信します。
- PutMedia セッションで受信したフラグメントごとに、Kinesis Video Streams は 1 つ以上の確認を送信します。クライアント側のネットワークに関する潜在的な考慮事項により、これらの確認が生成されても、それらの確認がすべて取得されない場合があります。

Note

1 つの永続接続で複数のフラグメントを送信するには、長時間実行されるストリーミング接続 PutMedia としてを使用します。複数の同時 PutMedia 接続を試みると、Kinesis Video Streams は最新の接続を `ConnectionLimitExceededException` エラーでスロットリングします。

PutMedia API を使用するときには以下の制限が適用されます。

- クライアントは、ストリームごとに 1 秒間に最大 5 回 PutMedia を呼び出すことができます。
- クライアントは、ストリームごとに 1 秒あたり最大 5 つのフラグメントを送信できます。
- Kinesis Video Streams は、最大 12.5 MB/秒、つまり PutMedia セッション中に 100 Mbps の速度でメディアデータを読み込みます。

以下の制約があることに注意してください。このような場合、Kinesis Video Streams は応答でエラー確認を送信します。

- 最大許容制限よりも長いタイムコードを持ち、50 MB を超えるデータを含むフラグメントは許可されません。
- 3 つ以上のトラックを含むフラグメントは許可されません。すべてのフラグメントの各フレームには、フラグメントヘッダーで定義されているトラックの 1 つと同じトラック番号が必要です。さらに、すべてのフラグメントには、フラグメントヘッダーで定義されたトラックごとに少なくとも 1 つのフレームが含まれている必要があります。

- 各フラグメントには、フラグメントメタデータで定義された各トラックに少なくとも 1 つのフレームが含まれている必要があります。
- フラグメント内の最初のフレームタイムスタンプは、前のフラグメントの最後のフレームタイムスタンプの後にする必要があります。
- 複数の MKV セグメントを含む、または許可されていない MKV 要素 (track* など) を含む MKV ストリームもエラー確認になります。

Kinesis Video Streams は、受信する各フラグメントと関連メタデータを「チャンク」と呼ばれるものに格納します。フラグメントメタデータには、次のものが含まれます。

- PutMedia 要求の開始時に提供される MKV ヘッダ
- 次のフラグメントの Kinesis Video Streams 固有のメタデータ。
 - server_timestamp - Kinesis Video Streams がフラグメントを受け取った時のタイムスタンプ。
 - producer_timestamp - プロデューサーがフラグメントの記録を開始したときのタイムスタンプ。Kinesis Video Streams は、リクエストで受信した 3 つの情報を使用して、この値を計算します。
 - フラグメントとともにリクエスト本文で受信されたフラグメントのタイムコード値。
 - 2 つのリクエストヘッダー: producerStartTimestamp (プロデューサーの記録開始時間) および fragmentTimeCodeType (ペイロード内フラグメントの絶対タイムコードまたは相対タイムコードの設定)。

Kinesis Video Streams は、フラグメントの producer_timestamp を次のように計算します。

fragmentTimeCodeType が相対の場合:

$$\text{producer\_timestamp} = \text{producerStartTimeStamp} + \text{フラグメントタイムコード}$$

fragmentTimeCodeType が絶対の場合:

$$\text{producer\_timestamp} = \text{フラグメントタイムコード (ミリ秒に変換)}$$

- Kinesis Video Streams によって割り当てられた一意のフラグメント番号。

Note

GetMedia リクエストを行うと、Kinesis Video Streams はこれらのチャンクのストリームを返します。クライアントは必要に応じてメタデータを処理できます。

Note

このオペレーションは AWS SDK for Java でのみ使用できます。他の言語の AWS SDKs ではサポートされていません。

Note

Kinesis Video Streams は、PutMedia API を介した取り込みやアーカイブ中に、コーデックのプライベートデータを解析および検証しません。KVS は、HLS API を介してストリームを消費するときに、MPEG-TS および MP4 フラグメントパッケージングのコーデックプライベートデータから必要な情報を抽出して検証します。

Note

Kinesis Video Streams メディア API を呼び出した後にエラーがスローされた場合、HTTP ステータスコードとレスポンス本文に加えて、次の情報が含まれます。

- `x-amz-ErrorType` HTTP ヘッダー — HTTP ステータスコードで提供されるものに加えて、より具体的なエラータイプが含まれます。
- `x-amz-RequestId` HTTP ヘッダー — 問題を報告したい場合は AWS、リクエスト ID を指定することで、サポートチームが問題をより適切に診断できます。

HTTP ステータスコードと `ErrorType` ヘッダーの両方を使用すれば、エラーが再試行可能かどうか、またはどのような条件下でエラーが再試行可能かについてプログラムで判断したり、クライアントプログラマーが再度試行するために必要なアクションに関する情報を提供したりできます。

詳細については、このトピックの下部にある [Errors] (エラー) セクションおよび「[Common Errors](#)」を参照してください。

リクエストの構文

```
POST /putMedia HTTP/1.1
x-amzn-stream-name: StreamName
x-amzn-stream-arn: StreamARN
x-amzn-fragment-timecode-type: FragmentTimecodeType
x-amzn-producer-start-timestamp: ProducerStartTimestamp

Payload
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

[FragmentTimecodeType](#)

この値を x-amzn-fragment-timecode-type HTTP ヘッダーとして渡します。

フラグメント (ペイロード、HTTP リクエスト本文) 内のタイムコードが producerStartTimestamp に対して絶対値であるか相対値であるかを示します。Kinesis Video Streams は、API の概要で説明されているように、この情報を使用して、リクエストで受信したフラグメントの producer_timestamp を計算します。

有効な値 : ABSOLUTE | RELATIVE

必須: はい

[ProducerStartTimestamp](#)

この値を x-amzn-producer-start-timestamp HTTP ヘッダーとして渡します。

これは、プロデューサーがメディアの記録を開始したプロデューサーのタイムスタンプです (リクエスト内の特定のフラグメントのタイムスタンプではありません)。

[StreamARN](#)

この値を x-amzn-stream-arn HTTP ヘッダーとして渡します。

メディアコンテンツを書き込む Kinesis のビデオストリームの Amazon リソース名 (ARN)。streamARN を指定しない場合は、streamName を指定する必要があります。

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

[StreamName](#)

この値を `x-amzn-stream-name` HTTP ヘッダーとして渡します。

メディアコンテンツを書き込む Kinesis ビデオストリームの名前。streamName を指定しない場合は、streamARN を指定する必要があります。

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

リクエストボディ

リクエストは以下のバイナリデータを受け入れます。

[Payload](#)

Kinesis ビデオストリームに書き込むメディアコンテンツ。現在の実装では、Kinesis Video Streams は、単一の MKV セグメントを含む Matroska (MKV) コンテナ形式のみをサポートしています。セグメントには、1 つまたは複数のクラスターを含めることができます。

Note

各 MKV クラスターは Kinesis ビデオストリームフラグメントにマッピングされます。選択したクラスター期間は、フラグメント期間になります。

レスポンスの構文

HTTP/1.1 200

Payload

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

レスポンスは、HTTP 本文として以下を返します。

Payload

Kinesis Video Streams が PutMedia リクエストを正常に受信した後、サービスがリクエストヘッダーを検証します。次に、サービスはペイロードの読み込みを開始し、最初に HTTP 200 レスポンスを送信します。

次に、サービスは、改行で区切られた一連の JSON オブジェクト (Acknowledgement オブジェクト) を含むストリームを返します。確認応答は、メディアデータが送信されるのと同じ接続で受信されます。PutMedia リクエストには多くの確認応答があります。各 Acknowledgement は、次のキーと値のペアで構成されています。

- **AckEventType** - 確認応答が表すイベントタイプ。
 - **Buffering**: Kinesis Video Streams がフラグメントの受信を開始しました。Kinesis Video Streams は、フラグメントデータの最初のバイトを受信されると、最初の Buffering 確認応答を送信します。
 - **Received**: Kinesis Video Streams がフラグメント全体を受信しました。データを保持するようにストリームを設定しなかった場合、プロデューサーはこの確認応答を受信するとフラグメントのバッファリングを停止できます。
 - **Persisted**: Kinesis Video Streams は、フラグメントを保持しました (Amazon S3 など)。データを保持するようにストリームを設定すると、この確認応答を受け取ります。この確認応答を受信すると、プロデューサーはフラグメントのバッファリングを停止できます。
 - **Error**: フラグメントの処理中に Kinesis Video Streams でエラーが発生しました。エラーコードを確認して、次のアクションを決定できます。
 - **Idle**: PutMediaセッションが進行中です。ただし、Kinesis Video Streams は現在データを受信していません。 Kinesis Video Streams は、最後のデータ受信後最大 30 秒間、この確認応答を定期的に送信します。データが 30 秒以内に受信されない場合、Kinesis Video Streams はリクエストを終了します。

Note

この確認応答は、データを送信していない場合でも、プロデューサーが PutMedia 接続が有効であるかどうかを判断するのに役立ちます。

- **FragmentTimecode** - 確認応答が送信されるフラグメントタイムコード。

AckEventType が Idle の場合、要素が欠落している可能性があります。

- **FragmentNumber** - 確認応答が送信される Kinesis Video Streams が生成するフラグメント番号。

- `ErrorId` および `ErrorCode` - `AckEventType`が `Error`、このフィールドには対応するエラーコードが表示されます。次に、エラー ID とそれに対応するエラーコードおよびエラーメッセージのリストを示します。
- 4000 - `STREAM_READ_ERROR` - データストリームの読み取り中にエラーが発生しました。
- 4001 - `MAX_FRAGMENT_SIZE_REACH` - フラグメントサイズが最大制限の 50 MB を超えています。
- 4002 - `MAX_FRAGMENT_DURATION_REACHED` - フラグメント期間が最大許容制限を超えています。
- 4003 - `MAX_CONNECTION_DURATION_REACH` - 接続時間が最大許容しきい値を超えています。
- 4004 - `FRAGMENT_TIMECODE_LESSER_THAN_PREVIOUS` - フラグメントのタイムコードは、以前のタイムコードのタイムコードよりも小さくなっています (`PutMedia` の呼び出しでは、フラグメントを順不同で送信することはできません)。
- 4005 - `MORE_THAN_ALLOWED_TRACKS_FOUND` - MKV に複数のトラックが見つかりました。(廃止)
- 4006 - `INVALID_MKV_DATA` - 入力ストリームを有効な MKV 形式として解析できませんでした。
- 4007 - `INVALID_PRODUCER_TIMESTAMP` - プロデューサーのタイムスタンプが無効です。
- 4008 - `STREAM_NOT_ACTIVE` - ストリームは存在しません (削除済)。
- 4009 - `FRAGMENT_METADATA_LIMIT_REACH` - フラグメントメタデータの制限に達しました。デベロッパーガイドの [「制限」](#) セクションを参照してください。
- 4010 - `TRACK_NUMBER_MISMATCH` - MKV フレームのトラック番号が MKV ヘッダーのトラックと一致しませんでした。
- 4011 - `FRAMES_MISSING_FOR_TRACK` - フラグメントには、MKV ヘッダーのトラックの少なくとも 1 つのフレームが含まれていませんでした。
- 4012 - `INVALID_FRAGMENT_METADATA` - フラグメントメタデータ名は文字列 で始めることはできません `AWS_`。
- 4500 - `KMS_KEY_ACCESS_DENIED` - ストリームの指定された KMS キーへのアクセスが拒否されました。
- 4501 - `KMS_KEY_DISABLED` - ストリームの指定された KMS キーが無効になっています。
- 4502 - `KMS_KEY_VALIDATION_ERROR` - ストリームの指定された KMS キーの検証に失敗しました。

- 4503 - KMS_KEY_NAVAILABLE - ストリームの指定された KMS キーは使用できません。
- 4504 - KMS_KEY_INVALID_USAGE - ストリームの指定された KMS キーの使用が無効です。
- 4505 - KMS_KEY_INVALID_STATE - ストリームの指定された KMS キーが無効な状態です。
- 4506 - KMS_KEY_NOT_FOUND - ストリームの指定された KMS キーが見つかりません。
- 5000 - INTERNAL_ERROR - 内部サービスエラー。
- 5001 - ARCHIVAL_ERROR - Kinesis Video Streams がデータストアにフラグメントを保持できませんでした。

Note

プロデューサーは、長時間実行される PutMedia リクエストのペイロードを送信するときに、送達確認のレスポンスを読み取る必要があります。中間のプロキシサーバーでのバッファリングが原因で、プロデューサーは同時に確認応答のチャンクを受信する場合があります。タイムリーに確認応答を受信したいプロデューサーは、PutMedia リクエストごとに送信するフラグメントを少なくすることができます。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

ConnectionLimitExceededException

許可されたクライアント接続の制限を超えたため、Kinesis Video Streams がリクエストをスロットリングしました。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

InvalidEndpointException

呼び出し元が間違ったエンドポイントを使用してデータをストリームに書き込みました。このような例外を受信すると、ユーザーは APIName を PUT_MEDIA に設定して GetDataEndpoint を呼び出し、応答からのエンドポイントを使用して次の PutMedia コールを呼び出す必要があります。

HTTP ステータスコード: 400

NotAuthorizedException

呼び出し元は、指定されたストリームで操作を実行する権限がないか、トークンの有効期限が切れています。

HTTP ステータスコード: 401

ResourceNotFoundException

ステータスコード: 404 指定された名前のストリームは存在しません。

HTTP ステータスコード: 404

例

確認応答の形式

確認応答の形式は次のとおりです。

```
{
  Acknowledgement : {
    "EventType": enum
    "FragmentTimecode": Long,
    "FragmentNumber": Long,
    "ErrorId" : String
  }
}
```

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)

- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

Amazon Kinesis Video Streams Archived Media

Amazon Kinesis Video Streams Archived Media では、以下のアクションがサポートされています。

- [GetClip](#)
- [GetDASHStreamingSessionURL](#)
- [GetHLSStreamingSessionURL](#)
- [GetImages](#)
- [GetMediaForFragmentList](#)
- [ListFragments](#)

GetClip

サービス : Amazon Kinesis Video Streams Archived Media

指定した時間範囲において、アーカイブされたオンデマンドメディアを含む MP4 ファイル (クリップ) を、指定したビデオストリームからダウンロードできます。

StreamName パラメーターと StreamARN パラメーターはどちらもオプションですが、この API 操作を呼び出すときは、StreamName または StreamARN のいずれかを指定する必要があります。

Note

エンドポイントを取得するには、最初に GetDataEndpoint API を呼び出す必要があります。次に、[--endpoint-url parameter](#) を使用して GetClip リクエストをこのエンドポイントに送信します。

Amazon Kinesis のビデオストリームには、MP4 を使用してデータを提供するための次の要件があります。

- [動画再生トラックの要件](#)。
- データの保持期間が 0 より大きい。
- 各フラグメントの動画トラックに、AVC (Advanced Video Coding) のコーデックプライベートデータが H.264 形式で、および HEVC のコーデックプライベートデータが H.265 形式で含まれている必要があります。詳細については、「[MPEG-4 仕様 ISO/IEC 14496-15](#)」を参照してください。ストリームデータを特定の形式に適応させる方法については、「[NAL 適応フラグ](#)」を参照してください。
- 各フラグメントのオーディオトラック (存在する場合) に、コーデックプライベートデータが AAC 形式 ([AAC 仕様 ISO/IEC 13818-7](#)) または [MS Wave 形式](#) で含まれている必要があります。

GetClip.OutgoingBytes Amazon CloudWatch メトリックスをモニタリングすることで、送信データの量をモニタリングできます。CloudWatch を使用して Kinesis Video Streams をモニタリングする方法については、「[Monitoring Kinesis Video Streams](#)」を参照してください。料金の詳細については、[Amazon Kinesis Video Streams の料金](#)と[AWS 料金](#)」を参照してください。送信 AWS データには料金が適用されます。

⚠ Important

各フラグメントに含まれるコーデックプライベートデータ (CPD) には、フラグメントを適切にデコードするために必要なフレームレート、解像度、エンコーディングプロファイルなどのコーデック固有の初期化情報が含まれています。CPD の変更は、結果のクリップのターゲットフラグメント間ではサポートされていません。CPD は、クエリされたメディアを通じて一貫性を保つ必要があります。そうしないと、エラーが返されます。

⚠ Important

トラックの変更はサポートされていません。トラックは、クエリされたメディア全体で一貫性を維持する必要があります。ストリームのフラグメントがビデオのみからオーディオとビデオの両方に変わった場合、または AAC オーディオトラックが A-Law オーディオトラックに変更された場合、エラーが返されます。

リクエストの構文

```
POST /getClip HTTP/1.1
Content-type: application/json

{
  "ClipFragmentSelector": {
    "FragmentSelectorType": "string",
    "TimestampRange": {
      "EndTimeStamp": number,
      "StartTimeStamp": number
    }
  },
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

ClipFragmentSelector

要求されたクリップの時間範囲とタイムスタンプのソース。

型: [ClipFragmentSelector](#) オブジェクト

必須: はい

StreamARN

メディアクリップを取得するストリームの Amazon リソースネーム (ARN)。

StreamName または StreamArn を指定する必要があります。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

メディアクリップを取得するストリームの名前。

StreamName または StreamArn を指定する必要があります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-Type: ContentType
```

Payload

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

レスポンスでは、以下の HTTP ヘッダーが返されます。

ContentType

要求されたクリップ内のメディアのコンテンツタイプ。

長さの制限：最小長 1、最大長は 128 です。

パターン: `^[a-zA-Z0-9_\.\\-]+$`

レスポンスは、HTTP 本文として以下を返します。

Payload

指定したビデオストリームのメディアクリップを含む従来の MP4 ファイル。出力には、指定された開始タイムスタンプから (最初の) 100 MB 分のフラグメントまたは 200 個のフラグメントが含まれます。詳細については、「[Kinesis Video Streams のクォータ](#)」を参照してください。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

制限を超えたため、Kinesis Video Streams がリクエストをスロットリングしました。後で呼び出しを試みてください。制限の詳細については、「[Kinesis Video Streams のクォータ](#)」を参照してください。

HTTP ステータスコード: 400

InvalidArgumentException

指定されたパラメータが制限を超えているか、サポートされていない、または使用できません。

HTTP ステータスコード: 400

InvalidCodecPrivateDataException

ビデオストリームの少なくとも 1 つのトラックにあるコーデックのプライベートデータは、この操作には無効です。

HTTP ステータスコード: 400

InvalidMediaFrameException

要求されたクリップの 1 つまたは複数のフレームは、指定されたコーデックに基づいて解析できませんでした。

HTTP ステータスコード: 400

MissingCodecPrivateDataException

ビデオストリームの少なくとも 1 つのトラックにコーデックのプライベートデータがありませんでした。

HTTP ステータスコード: 400

NoDataRetentionException

GetImages は、データを保持しないストリーム (つまり、DataRetentionInHours の が 0) に対してリクエストされました。

HTTP ステータスコード: 400

NotAuthorizedException

ステータスコード: 403 呼び出し元が指定されたストリームで操作を実行する権限がないか、トークンの有効期限が切れています。

HTTP ステータスコード: 401

ResourceNotFoundException

GetImages Kinesis Video Streams が指定したストリームを見つけられない場合、はこのエラーをスローします。

GetHLSStreamingSessionURL リクエストされた時間範囲内にフラグメントがないストリームに対して ON_DEMAND または PlaybackMode のセッションLIVE_REPLAY がリクエストされた場合、または過去 30 秒以内にフラグメントがないストリームに対して PlaybackMode のセッションLIVE がリクエストされた場合、とはこのエラーをGetDASHStreamingSessionURLスローします。

HTTP ステータスコード: 404

UnsupportedStreamMediaTypeException

メディアのタイプ (h.264 または h.265 ビデオ、AAC または G.711 オーディオなど) は、再生セッションの最初のフラグメントのトラックのコーデック IDs から判断できませんでした。トラック 1 のコーデック ID は V_MPEG/ISO/AVC である必要があります。また、オプションでトラック 2 のコーデック ID は A_AAC である必要があります。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

GetDASHStreamingSessionURL

サービス : Amazon Kinesis Video Streams Archived Media

ストリームの DASH (MPEG Dynamic Adaptive HTTP) の URL を取得します。次に、メディアプレーヤーで URL を開いて、ストリームのコンテンツを表示できます。

StreamName と StreamARN のパラメータは両方ともオプションですが、この API 操作を呼び出すときは StreamName または StreamARN を指定する必要があります。

Amazon Kinesis ビデオストリームには、MPEG-DASH を介してデータを提供するための次の要件があります。

- [動画再生トラックの要件](#)。
- データの保持期間が 0 より大きい。
- 各フラグメントの動画トラックに、AVC (Advanced Video Coding) のコーデックプライベートデータが H.264 形式で、および HEVC のコーデックプライベートデータが H.265 形式で含まれている必要があります。詳細については、「[MPEG-4 仕様 ISO/IEC 14496-15](#)」を参照してください。ストリームデータを特定の形式に適応させる方法については、「[NAL 適応フラグ](#)」を参照してください。
- 各フラグメントのオーディオトラック (存在する場合) に、コーデックプライベートデータが AAC 形式 ([AAC 仕様 ISO/IEC 13818-7](#)) または [MS Wave 形式](#) で含まれている必要があります。

以下の手順は、Kinesis Video Streams で MPEG-DASH を使用方法を示しています。

1. GetDataEndpoint API を呼び出してエンドポイントを取得します。次に、[--endpoint-url parameter](#) を使用して GetDASHStreamingSessionURL リクエストをこのエンドポイントに送信します。
2. GetDASHStreamingSessionURL を使用して、MPEG-DASH の URL を取得します。Kinesis Video Streams は、MPEG-DASH プロトコルを使用してストリーム内のコンテンツにアクセスするために使用される MPEG-DASH ストリーミングセッションを作成します。GetDASHStreamingSessionURL は、セッションの MPEG-DASH マニフェスト (MPEG-DASH でのストリーミングに必要なルートリソース) の認証済み URL (暗号化されたセッショントークンを含む) を返します。

Note

許可されていないエンティティがアクセスできる場所に、このトークンを共有したり保存したりしないでください。トークンがストリームのコンテンツへのアクセスを提供します。AWS 認証情報で使用するのと同じ方法でトークンを保護します。

マニフェストを通じて利用できるメディアは、要求されたストリーム、時間範囲、および形式のみで構成されます。他のメディアデータ（リクエストされた画面外のフレーム、代替ビットレートなど）は利用できません。

3. MPEG-DASH プロトコルをサポートするメディアプレーヤーに MPEG-DASH マニフェストの URL（暗号化されたセッショントークンを含む）を指定します。Kinesis Video Streams は、マニフェスト URL を通じて初期化フラグメントとメディアフラグメントを使用できるようにします。初期化フラグメントには、ストリームのコーデックプライベートデータ、およびビデオまたはオーディオデコーダーとレンダラーのセットアップに必要なその他のデータが含まれています。

メディアフラグメントには、エンコードされたビデオフレームまたはエンコードされたオーディオサンプルが含まれます。

4. メディアプレーヤーは、認証された URL を受け取り、ストリームメタデータとメディアデータを通常通りリクエストします。メディアプレーヤーがデータを要求すると、次のアクションが呼び出されます。

- `getDashManifest`: 再生するメディアのメタデータを含む MPEG DASH マニフェストを取得。
- `GetMP4InitFragment`: MP4 初期化フラグメントを取得。通常、メディアプレーヤーがメディアフラグメントをロードする前に、初期化フラグメントをロードします。このフラグメントには、「ftyp」および「moov」MP4 atom、およびメディアプレーヤーデコーダーを初期化するために必要な子 atom が含まれています。

初期化フラグメントは、Kinesis ビデオストリームのフラグメントには対応していません。これには、メディアプレーヤーがメディアフレームをデコードするために必要な、ストリームと各トラックのコーデックプライベートデータだけが含まれます。

- `GetMP4MediaFragment`: MP4 メディアフラグメントを取得。これらのフラグメントは、「moof」および「mdat」MP4 atom とその子 atom で構成され、エンコードされたフラグメントのメディアフレームとそのタイムスタンプを含みます。

⚠ Important

各フラグメントに含まれるコーデックプライベートデータ (CPD) には、フラグメントを適切にデコードするために必要なフレームレート、解像度、エンコーディングプロファイルなどのコーデック固有の初期化情報が含まれています。CPD の変更は、ストリーミングセッション中はサポートされていません。CPD は、クエリされたメディアを通じて一貫性を維持する必要があります。

⚠ Important

トラックの変更はサポートされていません。トラックは、クエリされたメディア全体で一貫性を保つ必要があります。ストリーム内のフラグメントがビデオのみからオーディオとビデオの両方になるか、AAC オーディオトラックが A-Law オーディオトラックに変更されると、ストリーミングは失敗します。

このアクションで取得されたデータは請求対象です。詳細については、「[料金](#)」を参照してください。

i Note

MPEG-DASH セッションに適用される制限については、「[Kinesis Video Streams クォータ](#)」を参照してください。

GetMP4MediaFragment.OutgoingBytes Amazon CloudWatch メトリックスをモニタリングすることで、メディアプレーヤーが消費するデータの量をモニタリングできます。CloudWatch を使用して Kinesis Video Streams をモニタリングする方法については、「[Monitoring Kinesis Video Streams](#)」を参照してください。料金情報については、[Amazon Kinesis Video Streams の料金とAWS 料金](#)」を参照してください。HLS セッションと送信 AWS データの両方の料金が適用されます。

HLSの詳細については、[Apple 開発者サイト](#)の [HTTP ライブストリーミング](#)を参照してください。

⚠ Important

Kinesis Video Streams アーカイブメディア API を呼び出した後にエラーがスローされた場合、HTTP ステータスコードとレスポンス本文に加えて、次の情報が含まれます。

- `x-amz-ErrorType` HTTP ヘッダー — HTTP ステータスコードで提供されるものに加えて、より具体的なエラータイプが含まれます。
- `x-amz-RequestId` HTTP ヘッダー — サポートチームに問題を報告したい場合 AWS 、リクエスト ID を指定すれば問題をより適切に診断できます。

HTTP ステータスコードと `ErrorType` ヘッダーの両方を使用すれば、エラーが再試行可能かどうか、またはどのような条件下でエラーが再試行可能かについてプログラムで判断したり、クライアントプログラマーが再度試行するために必要なアクションに関する情報を提供したりできます。

詳細については、このトピックの下部にある[Errors] (エラー) セクションおよび「[Common Errors](#)」を参照してください。

リクエストの構文

```
POST /getDASHStreamingSessionURL HTTP/1.1
Content-type: application/json
```

```
{
  "DASHFragmentSelector": {
    "FragmentSelectorType": "string",
    "TimestampRange": {
      "EndTimeStamp": number,
      "StartTimeStamp": number
    }
  },
  "DisplayFragmentNumber": "string",
  "DisplayFragmentTimestamp": "string",
  "Expires": number,
  "MaxManifestFragmentResults": number,
  "PlaybackMode": "string",
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[DASHFragmentSelector](#)

要求されたフラグメントの時間範囲とタイムスタンプのソース。

このパラメーターは、PlaybackMode が ON_DEMAND または LIVE_REPLAY の場合に必要です。このパラメーターは、PlaybackMode が LIVE の場合にはオプションになります。PlaybackMode が LIVE の場合、FragmentSelectorType は設定できますが、TimestampRange は設定しないでください。PlaybackMode が ON_DEMAND または LIVE_REPLAY の場合、FragmentSelectorType と TimestampRange の両方を設定する必要があります。

型: [DASHFragmentSelector](#) オブジェクト

必須: いいえ

[DisplayFragmentNumber](#)

フラグメントは、セッション内のシーケンス番号に基づいてマニフェストファイルで識別されます。DisplayFragmentNumber が ALWAYS に設定されている場合、Kinesis Video Streams のフラグメント番号は、属性名「kvs:fn」でマニフェストファイルの各 S 要素に追加されます。これらのフラグメント番号は、ロギングや他の API (例: GetMedia、GetMediaForFragmentList) で使用できます。これらのカスタム属性を活用するには、カスタム MPEG-DASH メディアプレーヤーが必要です。

デフォルト値は NEVER です。

型: 文字列

有効な値 : ALWAYS | NEVER

必須 : いいえ

[DisplayFragmentTimestamp](#)

MPEG-DASH 仕様に従って、マニフェストファイル内のフラグメントのウォールクロック時刻は、マニフェスト自体の属性を使用して派生できます。ただし、通常、MPEG-DASH 互換メ

メディアプレーヤーは、メディアタイムラインのギャップを適切に処理しません。Kinesis Video Streams は、マニフェストファイルのメディアタイムラインを調整して、不連続があるメディアを再生可能にします。したがって、マニフェストファイルから得られるウォールクロック時刻が不正確になる可能性があります。DisplayFragmentTimestamp が ALWAYS に設定されている場合、正確なフラグメントタイムスタンプが属性名「kvs:ts」でマニフェストファイルの各 S 要素に追加されます。このカスタム属性を活用するには、カスタム MPEG-DASH メディアプレーヤーが必要です。

デフォルト値は NEVER です。[DASHFragmentSelector](#) が SERVER_TIMESTAMP の場合、タイムスタンプはサーバーの開始タイムスタンプになります。同様に、[DASHFragmentSelector](#) が PRODUCER_TIMESTAMP の場合、タイムスタンプはプロデューサーの開始タイムスタンプになります。

型: 文字列

有効な値: ALWAYS | NEVER

必須: いいえ

[Expires](#)

要求されたセッションの有効期限が切れるまでの時間 (秒)。この値は 300 (5 分) から 43200 (12 時間) の間です。

セッションの有効期限が切れると、そのセッションに対して GetDashManifest、GetMP4InitFragment、または GetMP4MediaFragment への新しい呼び出しを行うことはできません。

デフォルトは300(5分)です。

型: 整数

値の範囲: 最小値 は 300 です。最大値は 43200 です。

必須: いいえ

[MaxManifestFragmentResults](#)

MPEG-DASH マニフェストで返されるフラグメントの最大数。

PlaybackMode が LIVE の場合、最新のフラグメントがこの値まで返されます。PlaybackMode が ON_DEMAND の場合、この最大数まで、最も古いフラグメントが返されます。

ライブ MPEG-DASH マニフェストで利用可能なフラグメントの数が多い場合、ビデオプレーヤーは再生を開始する前にコンテンツをバッファリングすることがよくあります。バッファサイズを大きくすると再生レイテンシーが増加しますが、再生中にバッファリングが発生する可能性は低くなります。ライブ MPEG-DASH マニフェストには、最低 3 個のフラグメントと最大 10 個のフラグメントを持つことをお勧めします。

デフォルトでは、PlaybackMode が LIVE または LIVE_REPLAY の場合は 5 個のフラグメント、PlaybackMode が ON_DEMAND の場合は 1,000 個のフラグメントです。

1,000 個のフラグメントの最大値は、1 秒のフラグメントを含むストリームで 16 分を超える動画、および 10 秒のフラグメントを含むストリームで 2 時間 30 分を超える動画に対応します。

型: Long

有効範囲: 最小値は 1 です。最大値は 5,000 です。

必須: いいえ

PlaybackMode

ライブ、ライブリプレイ、またはアーカイブ済のオンデマンドデータを取得するかどうか。

3 種類のセッションの機能は次のとおりです。

- **LIVE** : このタイプのセッションの場合、MPEG-DASH マニフェストは、最新のフラグメントが利用可能になると継続的に更新されます。メディアプレーヤーは 1 秒間隔で新しいマニフェストを取得することをお勧めします。このタイプのセッションがメディアプレーヤーで再生される場合、ユーザーインターフェイスには、通常「live (ライブ)」通知が表示されます。再生ウィンドウ内の位置を選択するためのスクラバーコントロールはありません。

Note

LIVEモードの場合、フラグメント間にギャップがある場合でも（つまり、フラグメントが欠落している場合）、使用可能な最新のフラグメントが MPEG-DASH マニフェストに含まれます。このようなギャップにより、メディアプレーヤーが再生中に停止したり、途切れたりすることがあります。このモードでは、再生リストの最新のフラグメントよりも古いフラグメントは MPEG-DASH マニフェストに追加されません。後続のフラグメントがマニフェストに追加された後に欠落フラグメントが使用可能になっても、古いフラグメントは追加されず、ギャップは埋められません。

- **LIVE_REPLAY** : このタイプのセッションの場合、MPEG-DASH マニフェストは、LIVE モードの更新と同様に更新されますが、特定の開始時刻からのフラグメントを含めることによって

開始される点が異なります。フラグメントは、取り込まれるときに追加されるのではなく、次のフラグメントの期間が経過すると追加されます。例えば、セッション内のフラグメントの長さが 2 秒の場合、2 秒ごとに新しいフラグメントがマニフェストに追加されます。このモードは、イベントの検出で再生を開始し、セッションの作成時点でまだ取り込まれていないライブストリーミングメディアを継続できるようにする場合に便利です。また、ON_DEMAND モードの 1,000 フラグメントの制限に制約されることなく、以前にアーカイブされたメディアをストリーミングする場合にも役立ちます。

- **ON_DEMAND** : このタイプのセッションの場合、MPEG-DASH マニフェストには、MaxManifestFragmentResults で指定された数までのセッションのすべてのフラグメントが含まれます。マニフェストは、セッションごとに 1 回だけ取得する必要があります。このタイプのセッションがメディアプレーヤーで再生される場合、ユーザーインターフェイスには、通常再生ウィンドウ内の位置を選択するためのスクラバーコントロールが表示されます。

すべての再生モードで、FragmentSelectorType が PRODUCER_TIMESTAMP で、開始タイムスタンプが同じフラグメントが複数ある場合、フラグメント番号が大きいフラグメント (つまり、新しいフラグメント) が MPEG-DASH マニフェストに含まれます。他のフラグメントは含まれません。タイムスタンプは異なるが、期間が重複しているフラグメントは、MPEG-DASH マニフェストに引き続き含まれます。これにより、メディアプレーヤーで予期しない動作が発生する場合があります。

デフォルトは LIVE です。

型: 文字列

有効な値 : LIVE | LIVE_REPLAY | ON_DEMAND

必須 : いいえ

[StreamARN](#)

MPEG-DASH マニフェスト URL を取得するストリームの Amazon リソースネーム (ARN)。

StreamName または StreamARN のパラメータを指定する必要があります。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

MPEG-DASH マニフェスト URL を取得するストリームの名前。

StreamName または StreamARN のパラメータを指定する必要があります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "DASHStreamingSessionURL": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

DASHStreamingSessionURL

メディアプレーヤーが MPEG-DASH マニフェストを取得するために使用できる URL (セッショントークンを含む)。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

制限を超えたため、Kinesis Video Streams がリクエストをスロットリングしました。後で呼び出しを試みてください。制限の詳細については、[「Kinesis Video Streams のクォータ」](#)を参照してください。

HTTP ステータスコード: 400

InvalidArgumentException

指定されたパラメータが制限を超えているか、サポートされていない、または使用できません。

HTTP ステータスコード: 400

InvalidCodecPrivateDataException

ビデオストリームの少なくとも 1 つのトラックにあるコーデックのプライベートデータは、この操作には無効です。

HTTP ステータスコード: 400

MissingCodecPrivateDataException

ビデオストリームの少なくとも 1 つのトラックにコーデックのプライベートデータがありませんでした。

HTTP ステータスコード: 400

NoDataRetentionException

GetImages は、データを保持しないストリームに対してリクエストされました (つまり、 のは 0 DataRetentionInHours です)。

HTTP ステータスコード: 400

NotAuthorizedException

ステータスコード: 403 呼び出し元が指定されたストリームで操作を実行する権限がないか、トークンの有効期限が切れています。

HTTP ステータスコード: 401

ResourceNotFoundException

GetImages Kinesis Video Streams が指定したストリームを見つけられない場合、はこのエラーをスローします。

GetHLSStreamingSessionURL リクエストされた時間範囲内にフラグメントがないストリームに対して ON_DEMAND または PlaybackMode のセッション LIVE_REPLAY がリクエストされた場合、または過去 30 秒以内にフラグメントがないストリームに対して PlaybackMode のセッション LIVE がリクエストされた場合、 とはこのエラーを GetDASHStreamingSessionURL スローします。

HTTP ステータスコード: 404

UnsupportedStreamMediaTypeException

メディアのタイプ (h.264 または h.265 ビデオ、AAC または G.711 オーディオなど) は、再生セッションの最初のフラグメントのトラックのコーデック IDs から判断できませんでした。トラック 1 のコーデック ID は V_MPEG/ISO/AVC である必要があります。また、オプションでトラック 2 のコーデック ID は A_AAC である必要があります。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

GetHLSStreamingSessionURL

サービス : Amazon Kinesis Video Streams Archived Media

ストリームの HTTP ライブストリーミング (HLS) URL を取得します。その後、ブラウザまたはメディアプレーヤーで URL を開いて、ストリームのコンテンツを表示できます。

StreamName と StreamARN のパラメータは両方ともオプションですが、この API 操作を呼び出すときは StreamName または StreamARN を指定する必要があります。

Amazon Kinesis ビデオストリームには、HLS を介してデータを提供するための次の要件があります。

- [動画再生トラックの要件](#)。
- データの保持期間が 0 より大きい。
- 各フラグメントのビデオトラックには、H.264 形式の場合は AVC (Advanced Video Coding)、H.265 形式の場合は HEVC ([MPEG-4 仕様 ISO/IEC 14496-15](#)) のコーデックプライベートデータが含まれている必要があります。ストリームデータを特定の形式に適応させる方法については、「[NAL 適応フラグ](#)」を参照してください。
- 各フラグメントのオーディオトラック (存在する場合) に、コーデックプライベートデータが AAC 形式 ([AAC specification ISO/IEC 13818-7](#)) で含まれている必要があります。

Kinesis Video Streams HLS セッションには、フラグメント化された MPEG-4 形式 (fmp4 または CMAF と呼ばれる) または MPEG-2 形式 (HLS 仕様でもサポートされている TS チャンクと呼ばれる) のフラグメントが含まれています。HLS フラグメントタイプの詳細については、[HLS の仕様](#)を参照してください。

以下の手順は、Kinesis Video Streams で HLS を使用方法を示しています。

1. GetDataEndpoint API を呼び出してエンドポイントを取得します。次に、[--endpoint-url parameter](#) を使用して GetHLSStreamingSessionURL リクエストをこのエンドポイントに送信します。
2. GetHLSStreamingSessionURL を使用して HLS URL を取得します。Kinesis Video Streams は、HLS プロトコルを使用してストリーム内のコンテンツにアクセスするために使用される HLS ストリーミングセッションを作成します。GetHLSStreamingSessionURL は、セッションの HLS マスタープレイリスト (HLS でのストリーミングに必要なルートリソース) の認証済み URL (暗号化されたセッショントークンを含む) を返します。

 Note

許可されていないエンティティがアクセスできる場所に、このトークンを共有したり保存したりしないでください。トークンがストリームのコンテンツへのアクセスを提供します。認証情報で使用する AWS のと同じ方法でトークンを保護します。

プレイリストを通じて利用できるメディアは、要求されたストリーム、時間範囲、および形式のみで構成されます。他のメディアデータ（リクエストされた画面外のフレーム、代替ビットレートなど）は利用できません。

3. HLS マスタープレイリストの URL（暗号化されたセッショントークンを含む）を、HLS プロトコルをサポートするメディアプレーヤーに指定します。Kinesis Video Streams は、HLS メディアプレイリスト、初期化フラグメント、およびメディアフラグメントをマスタープレイリスト URL から使用できるようにします。初期化フラグメントには、ストリームのコーデックプライベートデータ、およびビデオまたはオーディオデコーダーとレンダラーのセットアップに必要なその他のデータが含まれています。メディアフラグメントには、H.264 エンコードされたビデオフレームまたは AAC でエンコードされたオーディオサンプルが含まれています。
4. メディアプレーヤーは、認証された URL を受け取り、ストリームメタデータとメディアデータを通常通りリクエストします。メディアプレーヤーがデータを要求すると、次のアクションが呼び出されます。
 - GetHLSMasterPlaylist: 各トラックの GetHLSMediaPlaylist アクションの URL と、推定ビットレートや解像度など、メディアプレーヤーの追加のメタデータを含む HLS マスタープレイリストを取得します。
 - GetHLSMediaPlaylist: GetMP4InitFragment アクションで MP4 初期化フラグメントにアクセスするための URL と GetMP4MediaFragment アクションで MP4 メディアフラグメントにアクセスするための URL を含む HLS メディアプレイリストを取得します。HLS メディアプレイリストには、PlaybackMode が LIVE または ON_DEMAND の設定など、プレーヤーが再生するのに必要なストリームに関するメタデータも含まれています。HLS メディアプレイリストは通常、PlaybackType が ON_DEMAND のセッションでは静的です。HLS メディアプレイリストは、PlaybackType が LIVE のセッションの新しいフラグメントで継続的に更新されます。ビデオトラックとオーディオトラック（該当する場合）には、特定のトラックの MP4 メディア URL を含む個別の HLS メディアプレイリストがあります。
 - GetMP4InitFragment: MP4 初期化フラグメントを取得。通常、メディアプレーヤーがメディアフラグメントをロードする前に、初期化フラグメントをロードします。このフラグメントに

は、「fyp」および「moov」MP4 atom、およびメディアプレーヤーデコーダを初期化するために必要な子 atom が含まれています。

初期化フラグメントは、Kinesis ビデオストリームのフラグメントには対応していません。これには、メディアプレーヤーがメディアフレームをデコードするために必要な、ストリームと各トラックのコーデックプライベートデータだけが含まれます。

- GetMP4MediaFragment: MP4 メディアフラグメントを取得。これらのフラグメントは、「moof」および「mdat」MP4 atom とその子 atom で構成され、エンコードされたフラグメントのメディアフレームとそのタイムスタンプを含みます。

Note

各フラグメントに含まれるコーデックプライベートデータ (CPD) には、フラグメントを適切にデコードするために必要なフレームレート、解像度、エンコーディングプロファイルなどのコーデック固有の初期化情報が含まれています。TS と MP4 の両方で、CPD の変更はストリーミングセッション中にサポートされます。したがって、セッション内のフラグメントは、再生を中断することなく CPD で異なる情報を持つことができます。ストリーミングセッションごとに許可される CPD の変更は 500 個のみです。

Important

トラックの変更はサポートされていません。トラックは、クエリされたメディア全体で一貫性を維持する必要があります。ストリーム内のフラグメントがビデオのみからオーディオとビデオの両方に変わった場合、または AAC オーディオトラックが A-Law オーディオトラックに変更された場合、ストリーミングは失敗します。

このアクションで取得されたデータは請求対象です。詳細については、「[料金表](#)」を参照してください。

- GetTSFragment ストリーム内のすべてのトラックの初期化データとメディアデータの両方を含む MPEG TS フラグメントを取得します。

 Note

ContainerFormat が MPEG_TS の場合、GetMP4InitFragment と GetMP4MediaFragment の代わりにこのAPIを使用してストリームメディアを取得します。

このアクションで取得されたデータは請求対象です。詳細については、「[Amazon Kinesis Video Streams の料金表](#)」を参照してください。

ストリーミングセッション URL をプレイヤー間で共有することはできません。複数のメディアプレイヤーがセッションを共有している場合、サービスはセッションをスロットリングする場合があります。接続制限については、「[Kinesis Video Streams クォータ](#)」を参照してください。

GetMP4MediaFragment.OutgoingBytes Amazon CloudWatch メトリックスをモニタリングすることで、メディアプレイヤーが消費するデータの量をモニタリングできます。CloudWatch を使用して Kinesis Video Streams をモニタリングする方法については、「[Monitoring Kinesis Video Streams](#)」を参照してください。料金情報については、[Amazon Kinesis Video Streams の料金とAWS 料金](#)」を参照してください。HLS セッションと送信 AWS データの両方の料金が適用されます。

ドキュメントガイドの動画再生の例: [を使用して HLS ストリーミングセッション URL AWS CLI を取得する](#) および [例: HTML および JavaScript で HLS を使用する](#) を参照してください。

HLSの詳細については、[Apple 開発者サイト](#)の [HTTP ライブストリーミング](#)を参照してください。

 Important

Kinesis Video Streams アーカイブメディア API を呼び出した後にエラーがスローされた場合、HTTP ステータスコードとレスポンス本文に加えて、次の情報が含まれます。

- x-amz-ErrorType HTTP ヘッダー — HTTP ステータスコードで提供されるものに加えて、より具体的なエラータイプが含まれます。
- x-amz-RequestId HTTP ヘッダー — 問題を に報告する場合 AWS、リクエスト ID を指定すれば、サポートチームが問題をより適切に診断できます。

HTTP ステータスコードと `ErrorType` ヘッダーの両方を使用すれば、エラーが再試行可能かどうか、またはどのような条件下でエラーが再試行可能かについてプログラムで判断したり、クライアントプログラマーが再度試行するために必要なアクションに関する情報を提供したりできます。

詳細については、このトピックの下部にある[Errors] (エラー) セクションおよび「[Common Errors](#)」を参照してください。

リクエストの構文

```
POST /getHLSStreamingSessionURL HTTP/1.1
Content-type: application/json

{
  "ContainerFormat": "string",
  "DiscontinuityMode": "string",
  "DisplayFragmentTimestamp": "string",
  "Expires": number,
  "HLSFragmentSelector": {
    "FragmentSelectorType": "string",
    "TimestampRange": {
      "EndTimeStamp": number,
      "StartTimeStamp": number
    }
  },
  "MaxMediaPlaylistFragmentResults": number,
  "PlaybackMode": "string",
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

ContainerFormat

メディアのパッケージ化に使用するフォーマットを指定します。FRAGMENTED_MP4 コンテナ形式を指定すると、メディアが MP4 フラグメント (fMP4 または CMAF) にパッケージ化されます。これは、パッケージのオーバーヘッドが最小限なので、推奨されるパッケージングです。別のコンテナ形式オプションは MPEG_TS です。HLS は、リリースされてから MPEG TS チャンクをサポートしました。MPEG TS は、古い HLS プレーヤーでサポートされている唯一のパッケージである場合があります。MPEG TS は通常、5〜25 % のパッケージングオーバーヘッドがあります。つまり、MPEG TS は通常 fMP4 より 5〜25 % 広い帯域幅とコストを必要とします。

デフォルトは FRAGMENTED_MP4 です。

型: 文字列

有効な値 : FRAGMENTED_MP4 | MPEG_TS

必須 : いいえ

DiscontinuityMode

フラグメント間の不連続性を示すフラグをメディアプレイリストに追加するタイミングを指定します。

メディアプレーヤーは通常、各フラグメントのタイムスタンプに基づいて、再生するメディアコンテンツのタイムラインを作成します。これは、フラグメント間でオーバーラップやギャップがある場合 (一般に、[HLSFragmentSelector](#) が SERVER_TIMESTAMP に設定されている)、メディアプレーヤーのタイムラインでも一部の位置でフラグメント間に小さなギャップがあり、他の位置でフレームが上書きされることを意味します。メディアプレーヤーでタイムラインにギャップがあると、再生が停止したり、オーバーラップによって再生が不安定になる場合があります。フラグメント間に不連続フラグがある場合、メディアプレーヤーはタイムラインをリセットし、前のフラグメントの直後に次のフラグメントを再生します。

次のモードがサポートされています。

- ALWAYS: 不連続マーカは、HLS メディアプレイリストのすべてのフラグメントの間に配置されます。フラグメントのタイムスタンプが正確でない場合は、ALWAYS の値を使用することをお勧めします。
- NEVER: 不連続マーカはどこにでも配置されません。メディアプレーヤーのタイムラインがプロデューサーのタイムスタンプに最適にマップされるように、NEVER の値を使用することをお勧めします。

- **ON_DISCONTINUITY** : 不連続マーカは、50 ミリ秒を超えるギャップまたはオーバーラップを持つフラグメントの間に配置されます。ほとんどの再生シナリオでは、メディアタイムラインに重大な問題 (フラグメントの欠落など) がある場合にのみメディアプレーヤーのタイムラインがリセットされるように、**ON_DISCONTINUITY** の値を使用することをお勧めします。

デフォルトでは、[HLSFragmentSelector](#) が **SERVER_TIMESTAMP** に設定されている場合は **ALWAYS**、**PRODUCER_TIMESTAMP** に設定されている場合は **NEVER** です。

型: 文字列

有効な値 : **ALWAYS** | **NEVER** | **ON_DISCONTINUITY**

必須 : いいえ

[DisplayFragmentTimestamp](#)

フラグメントの開始タイムスタンプを HLS メディアプレイリストに含めるタイミングを指定します。通常、メディアプレーヤーは、再生セッションの最初のフラグメントの開始に対して相対的な時間として再生ヘッドの位置を報告します。ただし、開始タイムスタンプが HLS メディアプレイリストに含まれている場合、一部のメディアプレーヤーは、フラグメントのタイムスタンプに基づいて現在の再生ヘッドを絶対時間として報告することがあります。これは、閲覧者にメディアのウォールクロック時刻を表示する再生エクスペリエンスを作成するのに便利です。

デフォルトは **NEVER** です。[HLSFragmentSelector](#) が **SERVER_TIMESTAMP** の場合、タイムスタンプはサーバーの開始タイムスタンプになります。同様に、[HLSFragmentSelector](#) が **PRODUCER_TIMESTAMP** の場合、タイムスタンプはプロデューサーの開始タイムスタンプになります。

型: 文字列

有効な値 : **ALWAYS** | **NEVER**

必須 : いいえ

[Expires](#)

要求されたセッションの有効期限が切れるまでの時間 (秒)。この値は 300 (5 分) から 43200 (12 時間) の間です。

セッションの有効期限が切れると、そのセッションに対して `GetHLSMasterPlaylist`、`GetHLSMediaPlaylist`、`GetMP4InitFragment`、`GetMP4MediaFragment` または `GetTSFragment` への新しい呼び出しは行われません。

デフォルトは300(5分)です。

型: 整数

値の範囲: 最小値 は 300 です。最大値は 43200 です。

必須: いいえ

[HLSFragmentSelector](#)

要求されたフラグメントの時間範囲とタイムスタンプのソース。

このパラメーターは、PlaybackMode が ON_DEMAND または LIVE_REPLAY の場合に必要です。このパラメーターは、PlaybackMode が LIVE の場合にはオプションになります。PlaybackMode が LIVE の場合、FragmentSelectorType は設定できますが、TimestampRange は設定しないでください。PlaybackMode が ON_DEMAND または LIVE_REPLAY の場合、FragmentSelectorType と TimestampRange の両方を設定する必要があります。

型: [HLSFragmentSelector](#) オブジェクト

必須: いいえ

[MaxMediaPlaylistFragmentResults](#)

HLS メディアプレイリストで返されるフラグメントの最大数。

PlaybackMode が LIVE の場合、最新のフラグメントがこの値まで返されます。PlaybackMode が ON_DEMAND の場合、この最大数まで、最も古いフラグメントが返されません。

ライブ HLS メディアプレイリストでフラグメントの数が多い場合、ビデオプレーヤーは、再生を開始する前にコンテンツをバッファリングすることがよくあります。バッファサイズを大きくすると再生レイテンシーが増加しますが、再生中にバッファリングが発生する可能性は低くなります。ライブ HLS メディアプレイリストには、最低 3 つのフラグメントと最大 10 個のフラグメントを含めることをお勧めします。

デフォルトでは、PlaybackMode が LIVE または LIVE_REPLAY の場合は 5 個のフラグメント、PlaybackMode が ON_DEMAND の場合は 1,000 個のフラグメントです。

5,000 フラグメントの最大値は、1 秒のフラグメントを含むストリームでは 80 分を超える動画に対応し、10 秒のフラグメント含むストリームでは 13 時間を超える動画に相当します。

タイプ: Long

有効範囲: 最小値は 1 です。最大値は 5,000 です。

必須: いいえ

[PlaybackMode](#)

ライブ、ライブリプレイ、またはアーカイブ済のオンデマンドデータを取得するかどうか。

3 種類のセッションの機能は次のとおりです。

- **LIVE** : このタイプのセッションでは、HLS メディアプレイリストは、最新のフラグメントが利用可能になると継続的に更新されます。メディアプレーヤーは 1 秒間隔で新しいプレイリストを取得することをお勧めします。このタイプのセッションがメディアプレーヤーで再生される場合、ユーザーインターフェイスには、通常「live (ライブ)」通知が表示されます。再生ウィンドウ内の位置を選択するためのスクラバーコントロールはありません。

Note

LIVE モードでは、フラグメント間にギャップ (フラグメントの欠落) がある場合でも、利用可能な最新のフラグメントが HLS メディアプレイリストに含まれます。このようなギャップにより、メディアプレーヤーが再生中に停止したり、途切れたりすることがあります。このモードでは、フラグメントがプレイリストの最新のフラグメントよりも古い場合、HLS メディアプレイリストに追加されません。後続のフラグメントがプレイリストに追加された後に欠落フラグメントが使用可能になっても、古いフラグメントは追加されず、ギャップは埋められません。

- **LIVE_REPLAY** : このタイプのセッションでは、HLS メディアプレイリストは、LIVE モードの更新方法と同様に更新されますが、特定の開始時刻からのフラグメントを含めることによって開始される点が異なります。フラグメントは、取り込まれるときに追加されるのではなく、次のフラグメントの期間が経過すると追加されます。例えば、セッション内のフラグメントの長さが 2 秒の場合、2 秒ごとに新しいフラグメントがメディアプレイリストに追加されます。このモードは、イベントの検出で再生を開始し、セッションの作成時点でまだ取り込まれていないライブストリーミングメディアを継続できるようにする場合に便利です。また、ON_DEMAND モードの 1,000 フラグメントの制限に制約されることなく、以前にアーカイブされたメディアをストリーミングする場合にも役立ちます。
- **ON_DEMAND** : このタイプのセッションの場合、HLS メディアプレイリストには、MaxMediaPlaylistFragmentResults で指定された数までのセッションのすべてのフラグメントが含まれます。プレイリストは、セッションごとに 1 回だけ取得する必要があります。

す。このタイプのセッションがメディアプレーヤーで再生される場合、ユーザーインターフェイスには、通常再生ウィンドウ内の位置を選択するためのスクラバーコントロールが表示されます。

すべての再生モードで、FragmentSelectorType が PRODUCER_TIMESTAMP で、開始タイムスタンプが同じフラグメントが複数ある場合、フラグメント番号が大きいフラグメント（つまり、新しいフラグメント）が HLS メディアプレイリストに含まれます。他のフラグメントは含まれません。タイムスタンプは異なるが、期間が重複しているフラグメントは、HLS メディアプレイリストに引き続き含まれます。これにより、メディアプレーヤーで予期しない動作が発生する場合があります。

デフォルトは LIVE です。

型: 文字列

有効な値: LIVE | LIVE_REPLAY | ON_DEMAND

必須: いいえ

[StreamARN](#)

HLS マスタープレイリスト URL を取得するストリームの Amazon リソースネーム (ARN)。

StreamName または StreamARN のパラメータを指定する必要があります。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

[StreamName](#)

HLS マスタープレイリスト URL を取得するストリームの名前。

StreamName または StreamARN のパラメータを指定する必要があります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "HLSStreamingSessionURL": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[HLSStreamingSessionURL](#)

メディアプレーヤーが HLS マスタープレイリストを取得するために使用できる URL (セッショントークンを含む)。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

制限を超えたため、Kinesis Video Streams がリクエストをスロットリングしました。後で呼び出しを試みてください。制限の詳細については、「[Kinesis Video Streams クォータ](#)」を参照してください。

HTTP ステータスコード: 400

InvalidArgumentException

指定されたパラメータが制限を超えているか、サポートされていない、または使用できません。

HTTP ステータスコード: 400

InvalidCodecPrivateDataException

ビデオストリームの少なくとも 1 つのトラックにあるコーデックのプライベートデータは、この操作には無効です。

HTTP ステータスコード: 400

MissingCodecPrivateDataException

ビデオストリームの少なくとも 1 つのトラックにコーデックのプライベートデータがありませんでした。

HTTP ステータスコード: 400

NoDataRetentionException

GetImages は、データを保持しないストリーム (つまり、DataRetentionInHours の が 0) に対してリクエストされました。

HTTP ステータスコード: 400

NotAuthorizedException

ステータスコード: 403 呼び出し元が指定されたストリームで操作を実行する権限がないか、トークンの有効期限が切れています。

HTTP ステータスコード: 401

ResourceNotFoundException

GetImages Kinesis Video Streams が指定したストリームを見つけられない場合、はこのエラーをスローします。

GetHLSStreamingSessionURL リクエストされた時間範囲内にフラグメントがないストリームに対して ON_DEMAND または PlaybackMode のセッションLIVE_REPLAY がリクエストされた場合、または過去 30 秒以内にフラグメントがないストリームに対して PlaybackMode のセッションLIVE がリクエストされた場合、とはこのエラーをGetDASHStreamingSessionURLスローします。

HTTP ステータスコード: 404

UnsupportedStreamMediaTypeException

メディアのタイプ (h.264 または h.265 ビデオ、AAC または G.711 オーディオなど) は、再生セッションの最初のフラグメントのトラックのコーデック IDs から判断できませんでした。ト

トラック 1 のコーデック ID は V_MPEG/ISO/AVC である必要があります。また、オプションでトラック 2 のコーデック ID は A_AAC である必要があります。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

GetImages

サービス : Amazon Kinesis Video Streams Archived Media

イメージのマネージドサポートは、Kinesis Video Streams にストリーミングおよび保存されたビデオデータからイメージを取得するためのフルマネージド型の方法を提供します。イメージを使用して、人、ペット、車両検出などの機械学習 (ML) ワークロードを実行できます。イメージを使用して、モーションイベントのイメージプレビューやビデオクリップのスクラブなど、インタラクティブな要素を再生に追加することもできます。

GetImages は、特定の時間範囲、サンプリング間隔、およびイメージ形式設定の各タイムスタンプに対応するイメージのリストも取得します。

Note

エンドポイントを取得するには、最初に GetDataEndpoint API を呼び出す必要があります。次に、[--endpoint-url parameter](#) を使用して GetImages リクエストをこのエンドポイントに送信します。

動画再生トラックの要件。

リクエストの構文

```
POST /getImages HTTP/1.1
Content-type: application/json

{
  "EndTimeStamp": number,
  "Format": "string",
  "FormatConfig": {
    "string" : "string"
  },
  "HeightPixels": number,
  "ImageSelectorType": "string",
  "MaxResults": number,
  "NextToken": "string",
  "SamplingInterval": number,
  "StartTimestamp": number,
  "StreamARN": "string",
  "StreamName": "string",
  "WidthPixels": number
```

```
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[EndTimeStamp](#)

生成されるイメージの範囲の終了タイムスタンプ。StartTimeStamp から までの時間範囲EndTimeStampが 300 秒を超える場合はStartTimeStamp、を受け取りますIllegalArgumentException。

タイプ: タイムスタンプ

必須: はい

[Format](#)

イメージのエンコードに使用される形式。

型: 文字列

有効な値: JPEG | PNG

必須: はい

[FormatConfig](#)

イメージの生成時に適用できる追加のパラメータを含むキーと値のペア構造のリスト。FormatConfig キーは JPEGQuality、画像の生成に使用される JPEG 品質キーを示します。FormatConfig 値は 1~100 の Ints を受け入れます。値が 1 の場合、イメージは品質が低く、最適な圧縮で生成されます。値が 100 の場合、イメージは最も高品質で圧縮が少ない状態で生成されます。値を指定しない場合、JPEGQualityキーのデフォルト値は 80 に設定されます。

タイプ: 文字列間のマッピング

マップエントリ: アイテムの最大数は 1 です。

有効なキー: JPEGQuality

値の長さの制限: 最小長は 0 です。最大長は 256 です。

値のパターン: `^[a-zA-Z_0-9]+`

必須: いいえ

[HeightPixels](#)

WidthPixels パラメータと組み合わせて使用される出力イメージの高さ。パラメータHeightPixelsと WidthPixelsパラメータの両方を指定すると、イメージは指定されたアスペクト比に合わせて拡張されます。HeightPixels パラメータのみを指定すると、元のアスペクト比がWidthPixels比率の計算に使用されます。どちらのパラメータも指定しない場合、元のイメージサイズが返されます。

タイプ: 整数

有効範囲: 最小値 は 1 です。最大値は 2160 です。

必須: いいえ

[ImageSelectorType](#)

イメージの生成に使用するサーバーまたはプロデューサーのタイムスタンプのオリジン。

型: 文字列

有効な値 : PRODUCER_TIMESTAMP | SERVER_TIMESTAMP

必須: はい

[MaxResults](#)

API によって返されるイメージの最大数。

Note

デフォルトの制限は、API レスポンスあたり 25 イメージです。この値MaxResultsより大きい値を指定すると、ページサイズは 25 になります。追加の結果はページ分割されます。

タイプ: Long

有効範囲: 最小値は 1 です。最大値は 100 です。

必須: いいえ

NextToken

次のイメージセットのページ分割を開始する場所を指定するトークン。これは、以前に切り捨てられた応答からの `GetImages:NextToken` です。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 4,096 です。

パターン: `[a-zA-Z0-9+/]+= {0,2}`

必須: いいえ

SamplingInterval

ストリームからイメージを生成する必要がある時間間隔 (ミリ秒)。指定できる最小値は 200 ミリ秒 (1 秒あたり 5 イメージ) です。タイムスタンプ範囲がサンプリング間隔より小さい場合、使用可能な場合は からのイメージが返され `startTimestamp` ます。

タイプ: 整数

必須: いいえ

StartTimestamp

イメージを生成する開始点。これは、イメージを返すタイムスタンプの包括的な範囲内 `StartTimestamp` である必要があります。

タイプ: タイムスタンプ

必須: はい

StreamARN

イメージを取得するストリームの Amazon リソースネーム (ARN)。StreamName または StreamARN のパラメータを指定する必要があります。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

イメージを取得するストリームの名前。StreamName または StreamARN のパラメータを指定する必要があります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

WidthPixels

HeightPixels パラメータと組み合わせて使用される出力イメージの幅。パラメータWidthPixelsと HeightPixelsパラメータの両方を指定すると、イメージは指定されたアスペクト比に合わせて拡張されます。WidthPixels パラメータのみを指定するか、のみHeightPixelsを指定すると、ValidationException がスローされます。どちらのパラメータも指定しない場合、ストリームの元のイメージサイズが返されます。

タイプ: 整数

有効範囲: 最小値 は 1 です。最大値は 3840 です。

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "Images": [
    {
      "Error": "string",
      "ImageContent": "string",
      "TimeStamp": number
    }
  ],
  "NextToken": "string"
}
```

```
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[Images](#)

ビデオストリームから生成されたイメージのリスト。指定されたタイムスタンプに使用できるメディアがない場合、NO_MEDIAエラーが出力に表示されます。イメージの生成中にエラーが発生した場合、MEDIA_ERROR は欠落したイメージの原因として出力に表示されます。

型: [Image](#) オブジェクトの配列

[NextToken](#)

より多くのイメージを取得するためにリクエストで使用された暗号化されたトークン。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 4,096 です。

パターン: [a-zA-Z0-9+/-]{0,2}

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

制限を超えたため、Kinesis Video Streams がリクエストをスロットリングしました。後で呼び出しを試みてください。制限の詳細については、「[Kinesis Video Streams クォータ](#)」を参照してください。

HTTP ステータスコード: 400

InvalidArgumentException

指定されたパラメータが制限を超えているか、サポートされていない、または使用できません。

HTTP ステータスコード: 400

NoDataRetentionException

GetImages は、データを保持しないストリーム (つまり、DataRetentionInHoursの が 0) に対してリクエストされました。

HTTP ステータスコード: 400

NotAuthorizedException

ステータスコード: 403 呼び出し元が指定されたストリームで操作を実行する権限がないか、トークンの有効期限が切れています。

HTTP ステータスコード: 401

ResourceNotFoundException

GetImages Kinesis Video Streams が指定したストリームを見つけられない場合、はこのエラーをスローします。

GetHLSStreamingSessionURL リクエストされた時間範囲内にフラグメントがないストリームに対して ON_DEMANDまたは PlaybackModeのセッションLIVE_REPLAYがリクエストされた場合、または過去 30 秒以内にフラグメントがないストリームに対して PlaybackModeのセッションLIVEがリクエストされた場合、とはこのエラーをGetDASHStreamingSessionURLスローします。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

GetMediaForFragmentList

サービス : Amazon Kinesis Video Streams Archived Media

Amazon Kinesis ビデオストリームのアーカイブデータからフラグメント (フラグメント番号で指定) のリストのメディアを取得します。

Note

エンドポイントを取得するには、最初に GetDataEndpoint API を呼び出す必要があります。次に、[--endpoint-url parameter](#) を使用して GetMediaForFragmentList リクエストをこのエンドポイントに送信します。

制限については、[「Kinesis Video Streams のクォータ」](#)を参照してください。

Important

Kinesis Video Streams アーカイブメディア API を呼び出した後にエラーがスローされた場合、HTTP ステータスコードとレスポンス本文に加えて、次の情報が含まれます。

- x-amz-ErrorType HTTP ヘッダー — HTTP ステータスコードで提供されるものに加えて、より具体的なエラータイプが含まれます。
- x-amz-RequestId HTTP ヘッダー – 問題を に報告する場合 AWS、リクエスト ID を指定すれば、サポートチームが問題をより適切に診断できます。

HTTP ステータスコードと ErrorType ヘッダーの両方を使用すれば、エラーが再試行可能かどうか、またはどのような条件下でエラーが再試行可能かについてプログラムで判断したり、クライアントプログラマーが再度試行するために必要なアクションに関する情報を提供したりできます。

詳細については、このトピックの下部にある[Errors] (エラー) セクションおよび「[Common Errors](#)」を参照してください。

リクエストの構文

```
POST /getMediaForFragmentList HTTP/1.1
Content-type: application/json
```

```
{  
  "Fragments": [ "string" ],  
  "StreamARN": "string",  
  "StreamName": "string"  
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

Fragments

メディアを取得するフラグメントの数のリスト。これらの値は、[ListFragments](#) で取得します。

型: 文字列の配列

配列メンバー: 最小数は 1 項目です。最大数は 1000 項目です。

長さの制限: 最小長 1、最大長は 128 です。

パターン: `^[0-9]+$`

必須: はい

StreamARN

フラグメントメディアを取得するストリームの Amazon リソースネーム (ARN)。このパラメータ、または StreamName パラメータのいずれかを指定してください。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

フラグメントメディアを取得するストリームの名前。このパラメータ、または StreamARN パラメータのいずれかを指定してください。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-Type: ContentType
```

Payload

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

レスポンスでは、以下の HTTP ヘッダーが返されます。

ContentType

リクエストされたメディアのコンテンツタイプ

長さの制限: 最小長 1、最大長は 128 です。

パターン: ^[a-zA-Z0-9_\.\\-]+\$

レスポンスは、HTTP 本文として以下を返します。

Payload

Kinesis Video Streams が返すペイロードは、指定されたストリームからのチャンクのシーケンスです。チャンクの詳細については、「[PutMedia](#)」を参照してください。Kinesis Video Streams が GetMediaForFragmentList の呼び出しで返すチャンクには、次の追加の Matroska (MKV) タグも含まれます。

- AWS_KINESISVIDEO_FRAGMENT_NUMBER - チャンクで返されるフラグメント番号。
- AWS_KINESISVIDEO_SERVER_SIDE_TIMESTAMP - フラグメントのサーバー側のタイムスタンプ。

- AWS_KINESISVIDEO_PRODUCER_SIDE_TIMESTAMP - フラグメントのプロデューサー側のタイムスタンプ。

例外が発生した場合、次のタグが含まれます。

- AWS_KINESISVIDEO_FRAGMENT_NUMBER - 例外をスローしたフラグメントの数。
- AWS_KINESISVIDEO_EXCEPTION_ERROR_CODE - エラーの整数コード。
- AWS_KINESISVIDEO_EXCEPTION_MESSAGE - 例外のテキスト説明。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

制限を超えたため、Kinesis Video Streams がリクエストをスロットリングしました。後で呼び出しを試みてください。制限の詳細については、「[Kinesis Video Streams のクォータ](#)」を参照してください。

HTTP ステータスコード: 400

InvalidArgumentException

指定されたパラメータが制限を超えているか、サポートされていない、または使用できません。

HTTP ステータスコード: 400

NotAuthorizedException

ステータスコード: 403 呼び出し元が指定されたストリームで操作を実行する権限がないか、トークンの有効期限が切れています。

HTTP ステータスコード: 401

ResourceNotFoundException

GetImages Kinesis Video Streams が指定したストリームを見つけられない場合、はこのエラーをスローします。

GetHLSStreamingSessionURL リクエストされた時間範囲内にフラグメントがないストリームに対して ON_DEMANDまたは PlaybackModeのセッションLIVE_REPLAYがリクエストされた場合、または過去 30 秒以内にフラグメントがないストリームに対して PlaybackModeのセッショ

ンLIVEがリクエストされた場合、 とはこのエラーをGetDASHStreamingSessionURLスローします。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListFragments

サービス : Amazon Kinesis Video Streams Archived Media

アーカイブデータ内の指定したストリームとタイムスタンプ範囲から [Fragment](#) オブジェクトのリストを返します。

フラグメントのリストは、結果整合性があります。これは、フラグメントが保持されているという確認応答をプロデューサーが受信した場合でも、ListFragments へのリクエストから結果がすぐに返されない場合があることを意味します。ただし、結果は通常、1 秒未満で入手できます。

Note

エンドポイントを取得するには、最初に GetDataEndpoint API を呼び出す必要があります。次に、[--endpoint-url parameter](#) を使用して ListFragments リクエストをこのエンドポイントに送信します。

Important

Kinesis Video Streams アーカイブメディア API を呼び出した後にエラーがスローされた場合、HTTP ステータスコードとレスポンス本文に加えて、次の情報が含まれます。

- x-amz-ErrorType HTTP ヘッダー — HTTP ステータスコードで提供されるものに加えて、より具体的なエラータイプが含まれます。
- x-amz-RequestId HTTP ヘッダー — 問題を に報告する場合 AWS、リクエスト ID を指定すれば、サポートチームが問題をより適切に診断できます。

HTTP ステータスコードと ErrorType ヘッダーの両方を使用すれば、エラーが再試行可能かどうか、またはどのような条件下でエラーが再試行可能かについてプログラムで判断したり、クライアントプログラマーが再度試行するために必要なアクションに関する情報を提供したりできます。

詳細については、このトピックの下部にある[Errors] (エラー) セクションおよび「[Common Errors](#)」を参照してください。

リクエストの構文

```
POST /listFragments HTTP/1.1
```

Content-type: application/json

```
{
  "FragmentSelector": {
    "FragmentSelectorType": "string",
    "TimestampRange": {
      "EndTimeStamp": number,
      "StartTimeStamp": number
    }
  },
  "MaxResults": number,
  "NextToken": "string",
  "StreamARN": "string",
  "StreamName": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[FragmentSelector](#)

返すフラグメントの範囲のタイムスタンプ範囲とタイムスタンプオリジンを記述します。

Note

これは、NextTokenが API で渡されていない場合にのみ必要です。

型: [FragmentSelector](#) オブジェクト

必須: いいえ

[MaxResults](#)

返すフラグメントの総数。使用可能なフラグメントの総数が max-results で指定された値を超える場合、ページネーションを再開するために使用できる[ListFragments:NextToken](#) が出力に提供されます。

デフォルト値は 100 です。

タイプ: Long

有効範囲: 最小値は 1 です。最大値は 1000 です。

必須: いいえ

NextToken

ページ分割を始める場所を指定するトークン。これは、以前に切り詰められたレスポンスからの [ListFragments:NextToken](#) です。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 4,096 です。

パターン: [a-zA-Z0-9+/]+= {0,2}

必須: いいえ

StreamARN

フラグメントリストを取得するストリームの Amazon リソースネーム (ARN)。このパラメータ、または StreamName パラメータのいずれかを指定してください。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+

必須: いいえ

StreamName

フラグメントリストを取得するストリームの名前。このパラメータ、または StreamARN パラメータのいずれかを指定してください。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "Fragments": [
    {
      "FragmentLengthInMilliseconds": number,
      "FragmentNumber": "string",
      "FragmentSizeInBytes": number,
      "ProducerTimestamp": number,
      "ServerTimestamp": number
    }
  ],
  "NextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

Fragments

セレクトー基準を満たすストリームからのアーカイブされた [Fragment](#) オブジェクトのリスト。結果は、ページ間でも特定の順序ではありません。

ストリームにセレクト条件を満たすフラグメントがない場合、空のリストが返されます。

型: [Fragment](#) オブジェクトの配列

NextToken

返されたリストが切り捨てられた場合、操作はこのトークンを返します。これは次のページの結果を取得する上で使用します。返す結果がそれ以上存在しない場合、この値は `null` になります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 4,096 です。

パターン: `[a-zA-Z0-9+/]{1,4096}`

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

制限を超えたため、Kinesis Video Streams がリクエストをスロットリングしました。後で呼び出しを試みてください。制限の詳細については、「[Kinesis Video Streams クォータ](#)」を参照してください。

HTTP ステータスコード: 400

InvalidArgumentException

指定されたパラメータが制限を超えているか、サポートされていない、または使用できません。

HTTP ステータスコード: 400

NotAuthorizedException

ステータスコード: 403 呼び出し元が指定されたストリームで操作を実行する権限がないか、トークンの有効期限が切れています。

HTTP ステータスコード: 401

ResourceNotFoundException

GetImages Kinesis Video Streams が指定したストリームを見つけられない場合、はこのエラーをスローします。

GetHLSStreamingSessionURL リクエストされた時間範囲内にフラグメントがないストリームに対して ON_DEMANDまたは PlaybackModeのセッションLIVE_REPLAYがリクエストされた場合、または過去 30 秒以内にフラグメントがないストリームに対して PlaybackModeのセッションLIVEがリクエストされた場合、とはこのエラーをGetDASHStreamingSessionURLスローします。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)

- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

Amazon Kinesis Video Signaling Channels

以下のアクションは、Amazon Kinesis Video Signaling Channels でサポートされています。

- [GetIceServerConfig](#)
- [SendAlexaOfferToMaster](#)

GetIceServerConfig

サービス : Amazon Kinesis Video Signaling Channels

注: この API を使用する前に、GetSignalingChannelEndpoint API を呼び出して HTTPS エンドポイントをリクエストする必要があります。次に、GetIceServerConfig API リクエストでエンドポイントとリージョンを指定します。

WebRTC 接続の設定に使用できる URI、ユーザー名、パスワードなどの ICE (Interactive Connectivity Establishment) サーバー構成情報を取得します。ICE コンポーネントはこの構成情報を使用して WebRTC 接続を設定します。これには、TURN (Traversal Using Relays around NAT) リレーを使用したトラバーサルでの認証も含まれます。

TURN は、ピアツーピアアプリケーションの接続性を向上させるために使用されるプロトコルです。1 つまたは複数のピアが直接ピアツーピア接続できない場合でも、TURN を使用すればクラウドベースのリレーサービスを提供することで接続を確立できるようになります。詳細については、「[A REST API For Access To TURN Services](#)」を参照してください。

どちらの方法でもシグナリングチャネルを介して直接ピアツーピア接続が確立できない場合に、この API を呼び出してフォールバックメカニズムを確立できます。この API を呼び出すには、シグナリングチャネルの Amazon リソースネーム (ARN) を指定する必要があります。

リクエストの構文

```
POST /v1/get-ice-server-config HTTP/1.1
Content-type: application/json

{
  "ChannelARN": "string",
  "ClientId": "string",
  "Service": "string",
  "Username": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

ChannelARN

設定されたピア間のピアツーピア接続に使用されるシグナリングチャネルの ARN。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: はい

ClientId

ビューワー用の一意の識別子。シグナリングチャネル内で一意である必要があります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: いいえ

Service

目的のサービスを指定します。現在、TURN のみが有効な値です。

型: 文字列

有効な値: TURN

必須: いいえ

Username

認証情報に関連付けられるオプションのユーザー ID。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "IceServerList": [
    {
      "Password": "string",
      "Ttl": number,
      "Uris": [ "string" ],
      "Username": "string"
    }
  ]
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

[IceServerList](#)

ICE サーバ情報オブジェクトのリスト。

型: [IceServer](#) オブジェクトの配列

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

許可されたクライアントコールの制限を超えているため、リクエストが調整されました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

InvalidClientException

指定したクライアントは無効です。

HTTP ステータスコード: 400

NotAuthorizedException

呼び出し元には、この操作を実行するための権限がありません。

HTTP ステータスコード: 401

ResourceNotFoundException

指定したリソースは見つかりませんでした。

HTTP ステータスコード: 404

SessionExpiredException

クライアントセッションの有効期限が切れている場合。クライアントが接続されると、セッションは 45 分間有効です。クライアントはチャンネルに再接続して、メッセージの送受信を続行する必要があります。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

SendAlexaOfferToMaster

サービス : Amazon Kinesis Video Signaling Channels

Note

この API を使用する前に、GetSignalingChannelEndpoint API を呼び出しエンドポイントを取得する必要があります。次に、SendAlexaOfferToMaster API リクエストでエンドポイントとリージョンを指定します。

この API を使用すると、WebRTC 対応デバイスを Alexa ディスプレイデバイスに接続できます。起動すると、Alexa セッション記述プロトコル (SDP) オファーがマスターピアに送信されます。マスターが指定されたシグナリングチャンネルに接続されるとすぐに、オファーが配信されます。この API は、接続されたマスターから SDP 回答を返します。マスターがシグナリングチャンネルに接続されていない場合、メッセージの有効期限が切れるまで再配信要求が行われます。

リクエストの構文

```
POST /v1/send-alex-a-offer-to-master HTTP/1.1
Content-type: application/json

{
  "ChannelARN": "string",
  "MessagePayload": "string",
  "SenderClientId": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

ChannelARN

Alexa とマスターピアが通信するためのシグナリングチャンネルの Amazon リソースネーム (ARN)

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: はい

MessagePayload

base64 エンコード後の SDP オファー内容。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 10,000 です。

パターン: `[a-zA-Z0-9+/=]+`

必須: はい

SenderClientId

セッションクライアントの一意の識別子 (ID)。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: はい

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
  "Answer": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

Answer

base64 エンコード後の SDP 回答の内容。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 10,000 です。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

ClientLimitExceededException

許可されたクライアントコールの制限を超えているため、リクエストが調整されました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

NotAuthorizedException

呼び出し元には、この操作を実行するための権限がありません。

HTTP ステータスコード: 401

ResourceNotFoundException

指定したリソースは見つかりませんでした。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

Amazon Kinesis Video WebRTC ストレージ

Amazon Kinesis Video WebRTC Storage では、次のアクションがサポートされています。

- [JoinStorageSession](#)
- [JoinStorageSessionAsViewer](#)

JoinStorageSession

サービス : Amazon Kinesis Video WebRTC Storage

Note

この API を使用する前に、GetSignalingChannelEndpoint API を呼び出して WebRTC エンドポイントをリクエストする必要があります。次に、JoinStorageSession API リクエストでエンドポイントとリージョンを指定します。

進行中の一方方向ビデオおよび/または多方向オーディオ WebRTC セッションを入力チャネルのビデオ生成デバイスとして参加します。チャンネルに既存のセッションがない場合は、新しいストリーミングセッションを作成し、シグナリングチャネルの Amazon リソースネーム (ARN) を指定します。

現在、SINGLE_MASTERタイプでは、ビデオ生成デバイスはオーディオメディアとビデオメディアの両方をストリームに取り込むことができます。ビデオ生成デバイスのみがセッションに参加し、メディアを記録できます。

Important

現在、WebRTC 取り込みにはオーディオトラックとビデオトラックの両方が必要です。
現在の要件:

- ビデオトラック: H.264
- オーディオトラック: Opus

動画:

- 最大解像度は SQFHD (1920x1920)
- KVS WebRTC 取り込みは現在、特定の H264 プロファイルのセットをサポートしています。
 - 制約されたベースライン
 - ベースライン
 - メイン
 - high
 - 制約付き - 高

- プログレッシブ高

Kinesis ビデオストリームに取り込まれたビデオには、H.264 ビデオと AAC オーディオのパラメータがあります。

マスター参加者が WebRTC を介して接続をネゴシエートすると、取り込まれたメディアセッションは Kinesis ビデオストリームに保存されます。その後、複数のビューワーが再生 APIs を使用してリアルタイムメディアを再生できます。

取り込み済み WebRTC メディアでは、HLSやDASH再生、[GetImages](#) を介したイメージ生成などの既存の Kinesis Video Streams 機能を使用することもできます。

 Note

S3 イメージの配信と通知は現在サポートされていません。

 Note

チャネルのセッションに関連付けることができるビデオ生成デバイスクライアントは 1 つだけであるとしします。複数のクライアントがビデオ生成デバイスとして特定のチャネルのセッションに参加する場合、最新のクライアントリクエストが優先されます。

追加情報

- べき等 - この API はべき等ではありません。
- 再試行動作 - これは新しい API コールとしてカウントされます。
- 同時呼び出し - 同時呼び出しは許可されます。オファーは、コールごとに 1 回送信されます。

リクエストの構文

```
POST /joinStorageSession HTTP/1.1
Content-type: application/json
```

```
{
  "channelArn": "string"
```

```
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[channelArn](#)

シグナリングチャネルの Amazon リソースネーム (ARN)。

Important

この入力パラメータの大文字化に注意してください。

タイプ: 文字列

Pattern: `arn:(aws[a-zA-Z-]*):kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: はい

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 403

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceNotFoundException

指定したリソースは見つかりませんでした。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

JoinStorageSessionAsViewer

サービス : Amazon Kinesis Video WebRTC Storage

Note

この API を使用する前に、GetSignalingChannelEndpoint API を呼び出して WebRTC エンドポイントをリクエストする必要があります。次に、JoinStorageSessionAsViewer API リクエストでエンドポイントとリージョンを指定します。

JoinStorageSessionAsViewer を使用すると、ビューワーは進行中のクラウド録画 WebRTC ストリーミングセッションに参加できます。この API は、ビューワーと録画エージェントの間に SDP オファーと ICE 候補を送信することで WebRTC 接続を開始し、ビューワーが録画エージェントを介してマスターからリアルタイムビデオを受信し、録画エージェントを介して双方向音声通信に参加できるようにします。接続すると、ビューワーのオーディオは、提供された場合、マスター参加者を含む他の接続されたすべてのピアに転送され、Kinesis Video ストリームに保存される WebRTC ストリームに組み込まれます。

Important

現在、視聴者は動画トラックを送信できません。ビューワーは、オプションのオーディオトラックを送信することも、トラックをまったく送信することもできません。

現在のビューワー参加者の送信要件:

- ビデオトラック: サポートされていません
- オーディオトラック (オプション): Opus

マスター参加者が現在ビデオ生成デバイスに接続されている場合、Kinesis ビデオストリームに取り込まれたビデオには、H.264 ビデオと AAC オーディオのパラメータが含まれます。

Note

ビューワー参加者は、マスター参加者ではなく、ストレージセッションに直接接続します。ストレージセッションは、メディアの混合、複製、適切な宛先へのルーティングを処理します。

 Note

マスター参加者が存在しない場合、ビューワーは互いに聞くことができません。

ビューワー参加者が WebRTC を介して接続をネゴシエートすると、マスター参加者がストレージセッションにも接続されている限り、取り込まれたオーディオセッションは Kinesis ビデオストリームに保存されます。

取り込み済み WebRTC メディアでは、HLSやDASH再生、[GetImages](#) を介したイメージ生成などの既存の Kinesis Video Streams 機能を使用することもできます。

 Note

S3 イメージの配信と通知は現在サポートされていません。

 Note

チャンネルのセッションに関連付けることができるビデオ生成デバイスクライアントは 1 つだけであるとしします。複数のクライアントがビデオ生成デバイスとして特定のチャンネルのセッションに参加する場合、最新のクライアントリクエストが優先されます。

制限

現在の制限は設定されています。

- ビューワーの最大数: 3
- ビューワー参加者がマスター参加者なしでストレージセッションに接続したままになる最大時間: 3 分

 Important

ビューワーがストレージセッションから切断した場合 (ピア接続を閉じる場合)、そのクォータ (ビューワー制限) は 1 分間消費されたままになります。この 1 分間、ビューワーは同じクライアント ID でこの API を呼び出して、追加のビューワークォータを消費すること

となくセッションに再参加できます。1 分後、ビューワークォータが解放され、他のビューワが参加できるようになります。

追加情報

- べき等 - この API はべき等ではありません。
- 再試行動作 - これは新しい API コールとしてカウントされます。
- 同時呼び出し - 同時呼び出しは許可されます。オファーは、コールごとに 1 回送信されます。

リクエストの構文

```
POST /joinStorageSessionAsViewer HTTP/1.1
Content-type: application/json

{
  "channelArn": "string",
  "clientId": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

[channelArn](#)

シグナリングチャネルの Amazon リソースネーム (ARN)。



Important

この入力パラメータの大文字化に注意してください。

タイプ: 文字列

Pattern: `arn:(aws[a-zA-Z-]*):kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: はい

clientId

セッションクライアントの一意的識別子 (ID)。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: はい

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「[共通エラー](#)」を参照してください。

AccessDeniedException

この操作を実行するために必要なアクセス許可がありません。

HTTP ステータスコード: 403

ClientLimitExceededException

Kinesis Video Streams は、許可されたクライアントコールの制限を超えているため、リクエストをスロットリングしました。後で呼び出しを試みてください。

HTTP ステータスコード: 400

InvalidArgumentException

この入力パラメータの値は無効です。

HTTP ステータスコード: 400

ResourceNotFoundException

指定したリソースは見つかりませんでした。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS コマンドラインインターフェイス V2](#)
- [AWS SDK for .NET V4](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for Kotlin](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

データ型

Amazon Kinesis Video Streams では、次のデータ型がサポートされています。

- [ChannelInfo](#)
- [ChannelNameCondition](#)
- [DeletionConfig](#)
- [EdgeAgentStatus](#)
- [EdgeConfig](#)
- [ImageGenerationConfiguration](#)
- [ImageGenerationDestinationConfig](#)
- [LastRecorderStatus](#)

- [LastUploaderStatus](#)
- [ListEdgeAgentConfigurationsEdgeConfig](#)
- [LocalSizeConfig](#)
- [MappedResourceConfigurationListItem](#)
- [MediaSourceConfig](#)
- [MediaStorageConfiguration](#)
- [NotificationConfiguration](#)
- [NotificationDestinationConfig](#)
- [RecorderConfig](#)
- [ResourceEndpointListItem](#)
- [ScheduleConfig](#)
- [SingleMasterChannelEndpointConfiguration](#)
- [SingleMasterConfiguration](#)
- [StreamInfo](#)
- [StreamNameCondition](#)
- [StreamStorageConfiguration](#)
- [Tag](#)
- [UploaderConfig](#)

Amazon Kinesis Video Streams Media では、次のデータ型がサポートされています。

- [StartSelector](#)

Amazon Kinesis Video Streams Archived Media では、次のデータ型がサポートされています。

- [ClipFragmentSelector](#)
- [ClipTimestampRange](#)
- [DASHFragmentSelector](#)
- [DASHTimestampRange](#)
- [Fragment](#)
- [FragmentSelector](#)
- [HLSFragmentSelector](#)

- [HLSTimestampRange](#)
- [Image](#)
- [TimestampRange](#)

Amazon Kinesis Video Signaling Channels では、次のデータ型がサポートされています。

- [IceServer](#)

Amazon Kinesis Video WebRTC Storage では、次のデータ型がサポートされています。

Amazon Kinesis Video Streams

Amazon Kinesis Video Streams では、次のデータ型がサポートされています。

- [ChannelInfo](#)
- [ChannelNameCondition](#)
- [DeletionConfig](#)
- [EdgeAgentStatus](#)
- [EdgeConfig](#)
- [ImageGenerationConfiguration](#)
- [ImageGenerationDestinationConfig](#)
- [LastRecorderStatus](#)
- [LastUploaderStatus](#)
- [ListEdgeAgentConfigurationsEdgeConfig](#)
- [LocalSizeConfig](#)
- [MappedResourceConfigurationListItem](#)
- [MediaSourceConfig](#)
- [MediaStorageConfiguration](#)
- [NotificationConfiguration](#)
- [NotificationDestinationConfig](#)
- [RecorderConfig](#)
- [ResourceEndpointListItem](#)
- [ScheduleConfig](#)

- [SingleMasterChannelEndpointConfiguration](#)
- [SingleMasterConfiguration](#)
- [StreamInfo](#)
- [StreamNameCondition](#)
- [StreamStorageConfiguration](#)
- [Tag](#)
- [UploaderConfig](#)

ChannelInfo

サービス: Amazon Kinesis Video Streams

シグナリングチャネルのメタデータとプロパティをカプセル化する構造体。

内容

ChannelARN

シグナリングチャネルの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

ChannelName

シグナリングチャネルの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: いいえ

ChannelStatus

シグナリングチャネルの現在のステータス。

型: 文字列

有効な値: CREATING | ACTIVE | UPDATING | DELETING

必須: いいえ

ChannelType

シグナリングチャネルのタイプ。

型: 文字列

有効な値: SINGLE_MASTER | FULL_MESH

必須: いいえ

CreationTime

シグナリングチャネルが作成された時刻。

型: タイムスタンプ

必須: いいえ

SingleMasterConfiguration

SINGLE_MASTER チャネルタイプの設定を含む構造体。

型: [SingleMasterConfiguration](#) オブジェクト

必須: いいえ

Version

シグナリングチャネルの最新バージョン。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長 64

パターン: [a-zA-Z0-9]+

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ChannelNameCondition

サービス: Amazon Kinesis Video Streams

ListSignalingChannels API のオプションの入力パラメータ。ListSignalingChannels の呼び出し中にこのパラメータを指定した場合、API が、ChannelNameCondition で指定した条件を満たすチャンネルのみを返します。

内容

ComparisonOperator

比較演算子。現在指定できるのは、所定のプレフィックスで始まる名前のシグナリングチャンネルを検索する BEGINS_WITH 演算子のみです。

型: 文字列

有効な値: BEGINS_WITH

必須: いいえ

ComparisonValue

比較する値。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DeletionConfig

サービス: Amazon Kinesis Video Streams

Edge エージェントからストリームの接続を削除するために必要な設定の詳細。

内容

DeleteAfterUpload

Kinesis Video Stream クラウドにアップロードされたメディアを削除対象としてマークするかどうかを示すために使用されるboolean値。、または の制限MaxLocalMediaSizeInMBに達した場合などtrue、削除設定値のいずれかが に設定されている場合EdgeRetentionInHours、メディアファイルは削除できます。

デフォルト値は に設定されているためtrue、メディアファイルが最初に AWS クラウドにアップロードされる前に削除されないようにアップローダスケジュールを設定します。

タイプ: ブール値

必須: いいえ

EdgeRetentionInHours

Edge Agent のストリームにデータを保持する時間数。保持時間のデフォルト値は 720 時間で、30 日に変換されます。

タイプ: 整数

有効範囲: 最小値 は 1 です。最大値は 720 です。

必須: いいえ

LocalSizeConfig

エッジ設定を削除するために必要なローカルサイズの値。

型: [LocalSizeConfig](#) オブジェクト

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

EdgeAgentStatus

サービス: Amazon Kinesis Video Streams

エッジエージェントのレコーダージョブとアップローダージョブの最新のステータスの詳細を含むオブジェクト。この情報を使用して、エッジエージェントの現在の状態を判断します。

内容

LastRecorderStatus

ストリームのエッジ記録ジョブの最新ステータス。

型: [LastRecorderStatus](#) オブジェクト

必須: いいえ

LastUploaderStatus

ストリームのエッジからクラウドへのアップローダージョブの最新ステータス。

型: [LastUploaderStatus](#) オブジェクト

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

EdgeConfig

サービス: Amazon Kinesis Video Streams

Edge Agent IoT Greengrass コンポーネントとの同期に使用されるストリームのエッジ設定の説明。Edge Agent コンポーネントは、オンプレミスの IoT Hub Device セットアップで実行されます。

内容

HubDeviceArn

ストリームの「モノのインターネット (IoT) モノ」 ARN。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:iot:[a-z0-9-]+:[0-9]+:thing/[a-zA-Z0-9_.-]+`

必須: はい

RecorderConfig

レコーダー設定はローカルMediaSourceConfigの詳細で構成され、カメラにストリーミングされたローカルメディアファイルにアクセスするための認証情報として使用されます。

型: [RecorderConfig](#) オブジェクト

必須: はい

DeletionConfig

削除設定は、削除に使用される保持時間 (EdgeRetentionInHours) とローカルサイズ設定 (LocalSizeConfig) の詳細で構成されます。

型: [DeletionConfig](#) オブジェクト

必須: いいえ

UploaderConfig

アップローダー設定には、エッジエージェントから Kinesis Video Streams への記録されたメディアファイルのアップロードジョブをスケジュールするために使用されるScheduleExpression詳細が含まれています。

型: [UploaderConfig](#) オブジェクト

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ImageGenerationConfiguration

サービス: Amazon Kinesis Video Streams

KVS イメージの配信に必要な情報を含む構造。null の場合、設定はストリームから削除されます。

内容

DestinationConfig

顧客にイメージを配信するために必要な情報を含む構造。

型: [ImageGenerationDestinationConfig](#) オブジェクト

必須: はい

Format

受け入れられるイメージ形式。

型: 文字列

有効な値: JPEG | PNG

必須: はい

ImageSelectorType

イメージの生成に使用するサーバーまたはプロデューサーのタイムスタンプのオリジン。

型: 文字列

有効な値: SERVER_TIMESTAMP | PRODUCER_TIMESTAMP

必須: はい

SamplingInterval

ストリームからイメージを生成する必要がある時間間隔 (ミリ秒)。指定できる最小値は 200 ミリ秒です。タイムスタンプ範囲がサンプリング間隔より小さい場合、使用可能な場合はからのイメージが返されStartTimestampます。

タイプ: 整数

必須: はい

Status

ContinuousImageGenerationConfigurations API が有効か無効かを示します。

型: 文字列

有効な値: ENABLED | DISABLED

必須: はい

FormatConfig

イメージの生成時に適用できる追加のパラメータを含むキーと値のペア構造のリスト。FormatConfig キーは JPEGQuality、画像の生成に使用される JPEG 品質キーを示します。FormatConfig 値は 1~100 の Ints を受け入れます。値が 1 の場合、イメージは品質が低く、最適な圧縮で生成されます。値が 100 の場合、イメージは最高品質の圧縮率で生成されます。値を指定しない場合、JPEGQualityキーのデフォルト値は 80 に設定されます。

タイプ: 文字列間のマッピング

マップエントリ: アイテムの最大数は 1 です。

有効なキー: JPEGQuality

値の長さの制限: 最小長は 0 です。最大長は 256 です。

値のパターン: `^[a-zA-Z_0-9]+`

必須: いいえ

HeightPixels

WidthPixels パラメータと組み合わせて使用される出力イメージの高さ。パラメータHeightPixelsと WidthPixelsパラメータの両方を指定すると、イメージは指定されたアスペクト比に合わせて拡張されます。HeightPixels パラメータのみを指定すると、元のアスペクト比がWidthPixels比率の計算に使用されます。どちらのパラメータも指定しない場合、元のイメージサイズが返されます。

タイプ: 整数

有効範囲: 最小値 は 1 です。最大値は 2160 です。

必須: いいえ

WidthPixels

HeightPixels パラメータと組み合わせて使用される出力イメージの幅。パラメータWidthPixelsと HeightPixelsパラメータの両方を指定すると、イメージは指定されたアスペクト比に合わせて拡張されます。WidthPixels パラメータのみを指定すると、元のアスペクト比を使用してHeightPixels比率が計算されます。どちらのパラメータも指定しない場合、元のイメージサイズが返されます。

タイプ: 整数

有効範囲: 最小値 は 1 です。最大値は 3840 です。

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ImageGenerationDestinationConfig

サービス: Amazon Kinesis Video Streams

顧客にイメージを配信するために必要な情報を含む構造。

内容

DestinationRegion

イメージが配信される S3 バケットの AWS リージョン。これは、ストリームが配置されているリージョンと一致する DestinationRegion 必要があります。

タイプ: 文字列

長さの制限: 最小長は 9 です。最大長は 14 です。

パターン: `^[a-z]+(-[a-z]+)?-[a-z]+-[0-9]$`

必須: はい

Uri

イメージが配信される場所を識別する Uniform Resource Identifier (URI)。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `^[a-zA-Z_0-9]+:(//)?(^[^/]+)/?([^\*]*)$`

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

LastRecorderStatus

サービス: Amazon Kinesis Video Streams

ストリームのエッジ記録ジョブの最新ステータス。

内容

JobStatusDetails

レコーダージョブの最新ステータスの説明。

タイプ: 文字列

必須: いいえ

LastCollectedTime

レコーダージョブが最後に実行され、メディアがローカルディスクに保存されたタイムスタンプ。

型: タイムスタンプ

必須: いいえ

LastUpdateTime

レコーダーのステータスが最後に更新されたタイムスタンプ。

型: タイムスタンプ

必須: いいえ

RecorderStatus

最新のレコーダージョブのステータス。

型: 文字列

有効な値 : SUCCESS | USER_ERROR | SYSTEM_ERROR

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

LastUploaderStatus

サービス: Amazon Kinesis Video Streams

ストリームのエッジからクラウドへのアップローダージョブの最新のステータス。

内容

JobStatusDetails

アップローダージョブの最新ステータスの説明。

タイプ: 文字列

必須: いいえ

LastCollectedTime

アップローダージョブが最後に実行され、メディアがクラウドに収集されたタイムスタンプ。

型: タイムスタンプ

必須: いいえ

LastUpdatedTime

アップローダーのステータスが最後に更新されたタイムスタンプ。

型: タイムスタンプ

必須: いいえ

UploaderStatus

最新のアップローダージョブのステータス。

型: 文字列

有効な値 : SUCCESS | USER_ERROR | SYSTEM_ERROR

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ListEdgeAgentConfigurationsEdgeConfig

サービス: Amazon Kinesis Video Streams

単一ストリームのエッジ設定の説明。

内容

CreationTime

ストリームがエッジ設定を最初に作成したタイムスタンプ。

型: タイムスタンプ

必須: いいえ

EdgeConfig

Edge Agent IoT Greengrass コンポーネントとの同期に使用されるストリームのエッジ設定の説明。Edge Agent コンポーネントは、オンプレミスの IoT Hub Device セットアップで実行されます。

型: [EdgeConfig](#) オブジェクト

必須: いいえ

FailedStatusDetails

生成された障害ステータスの説明。

タイプ: 文字列

必須: いいえ

LastUpdateTime

ストリームがエッジ設定を最後に更新したタイムスタンプ。

型: タイムスタンプ

必須: いいえ

StreamARN

ストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

ストリームの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: `[a-zA-Z0-9_.-]+`

必須: いいえ

SyncStatus

ストリームのエッジ設定の現在の同期ステータス。

型: 文字列

有効な値 : SYNCING | ACKNOWLEDGED | IN_SYNC | SYNC_FAILED | DELETING | DELETE_FAILED | DELETING_ACKNOWLEDGED

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

LocalSizeConfig

サービス: Amazon Kinesis Video Streams

Edge Agent のストリームに保存するメディアの最大サイズ (MaxLocalMediaSizeInMB) と、ストリームの最大サイズに達したときに使用する戦略 (StrategyOnFullSize) を含む設定の詳細。

内容

MaxLocalMediaSizeInMB

Edge Agent のストリームに保存するメディアの全体的な最大サイズ。

タイプ: 整数

有効な範囲: 最小値は 64 です。最大値は 2000000 です。

必須: いいえ

StrategyOnFullSize

ストリームMaxLocalMediaSizeInMBの制限に達したときに実行する戦略。

型: 文字列

有効な値: DELETE_OLDEST_MEDIA | DENY_NEW_MEDIA

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

MappedResourceConfigurationListItem

サービス: Amazon Kinesis Video Streams

メディアストレージ設定プロパティをカプセル化または含む構造。

内容

ARN

ストリームに関連付けられた Kinesis Video Stream リソースの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

Type

kinesis ビデオストリームの関連リソースのタイプ。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

MediaSourceConfig

サービス: Amazon Kinesis Video Streams

カメラにストリーミングされるメディアファイルにアクセスするために必要な認証情報 (MediaUriSecretArn および MediaUriType) で構成される設定の詳細。

内容

MediaUriSecretArn

カメラのユーザー名とパスワードの AWS Secrets Manager ARN、またはローカルメディアファイルの場所。

タイプ: 文字列

長さの制限: 最小長は 20 です。最大長は 2,048 です。

パターン: `arn:[a-z\d-]+:secretsmanager:[a-z0-9-]+:[0-9]+:secret:[a-zA-Z0-9_.-]+`

必須: はい

MediaUriType

Uniform Resource Identifier (URI) タイプ。FILE_URI 値は、ローカルメディアファイルのストリーミングに使用できます。

Note

プレビューは、RTSP_URI メディアソース URI 形式のみをサポートします。

型: 文字列

有効な値: RTSP_URI | FILE_URI

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

MediaStorageConfiguration

サービス: Amazon Kinesis Video Streams

メディアストレージ設定プロパティをカプセル化または含む構造。

- `StorageStatus` が有効になっている場合、データは `StreamARN` 提供された に保存されます。WebRTC Ingestion を機能させるには、ストリームでデータ保持が有効になっている必要があります。
- `StorageStatus` が無効になっている場合、データは保存されず、`StreamARN` パラメータは必要ありません。

内容

Status

メディアストレージ設定のステータス。

型: 文字列

有効な値: ENABLED | DISABLED

必須: はい

StreamARN

ストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

NotificationConfiguration

サービス: Amazon Kinesis Video Streams

この API を使用して、フラグメントがストリームで使用可能になったときの Amazon Simple Notification Service (Amazon SNS) 通知を設定します。このパラメータが null の場合、設定はストリームから削除されます。

詳細については、[「Kinesis Video Streams の通知」](#)を参照してください。

内容

DestinationConfig

顧客に通知を配信するために必要な送信先情報。

型: [NotificationDestinationConfig](#) オブジェクト

必須: はい

Status

通知設定が有効か無効かを示します。

型: 文字列

有効な値: ENABLED | DISABLED

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

NotificationDestinationConfig

サービス: Amazon Kinesis Video Streams

顧客に通知を配信するために必要な情報を含む構造。

内容

Uri

イメージが配信される場所を識別する Uniform Resource Identifier (URI)。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `^[a-zA-Z_0-9]+:(//)?([^\/]+)/?([^\*]*)$`

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

RecorderConfig

サービス: Amazon Kinesis Video Streams

レコーダー設定は、カメラにストリーミングされたローカルメディアファイルにアクセスするための認証情報として使用されるローカルMediaSourceConfigの詳細で構成されます。

内容

MediaSourceConfig

カメラにストリーミングされるメディアファイルにアクセスするために必要な認証情報 (MediaUriSecretArn および MediaUriType) で構成される設定の詳細。

型: [MediaSourceConfig](#) オブジェクト

必須: はい

ScheduleConfig

とScheduleExpression、カメラまたはローカルメディアファイルから Edge エージェントに記録するスケジューリングを指定するDurationInMinutes詳細で構成される設定。ScheduleExpression 属性が指定されていない場合、エッジエージェントは常に記録モードに設定されます。

型: [ScheduleConfig](#) オブジェクト

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ResourceEndpointListItem

サービス: Amazon Kinesis Video Streams

GetSignalingChannelEndpoint API によって返されるシグナリングチャネルのエンドポイントを記述するオブジェクト。

メディアサーバーエンドポイントはWEBRTCプロトコルに対応します。

内容

Protocol

GetSignalingChannelEndpoint API によって返されるシグナリングチャネルのプロトコル。

型: 文字列

有効な値 : WSS | HTTPS | WEBRTC

必須 : いいえ

ResourceEndpoint

GetSignalingChannelEndpoint API によって返されるシグナリングチャネルのエンドポイント。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ScheduleConfig

サービス: Amazon Kinesis Video Streams

この API を使用すると、カメラまたはローカルメディアファイルが Edge エージェントに記録する期間を指定できます。ScheduleConfig は、ScheduleExpression と DurationInMinutes 属性で構成されます。

ScheduleConfig が で指定されていない場合 RecorderConfig、エッジエージェントは常に記録モードに設定されます。

ScheduleConfig が で指定されていない場合 UploaderConfig、エッジエージェントは定期的に (1 時間ごとに) アップロードします。

内容

DurationInSeconds

メディアを記録する合計期間。ScheduleExpression 属性が指定されている場合は、DurationInSeconds 属性も指定する必要があります。

タイプ: 整数

有効範囲: 最小値 は 60 です。最大値は 3600 です。

必須: はい

ScheduleExpression

カメラまたはローカルメディアファイルから Edge Agent に記録するようにジョブをスケジュールする、Quartz cron 式。に が指定され ScheduleExpression ていない場合 RecorderConfig、エッジエージェントは常に記録モードに設定されます。

Quartz の詳細については、[「Cron Trigger チュートリアル」](#) ページを参照して、有効な式とその使用方法を理解してください。

タイプ: 文字列

長さの制限: 最小長は 11 です。最大長は 100 です。

パターン: `[^\n]{11,100}`

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SingleMasterChannelEndpointConfiguration

サービス: Amazon Kinesis Video Streams

SINGLE_MASTER チャンネルタイプのエンドポイント設定を含むオブジェクト。

内容

Protocols

このプロパティは、この SINGLE_MASTER シグナリングチャンネルを経由する通信の特性を判断するために使用されます。WSS を指定した場合、この API が websocket エンドポイントを返します。HTTPS を指定した場合、この API が HTTPS エンドポイントを返します。

型: 文字列の配列

配列メンバー: 最小数は 1 項目です。最大数は 5 項目です。

有効な値: WSS | HTTPS | WEBRTC

必須: いいえ

Role

このプロパティは、SINGLE_MASTER シグナリングチャンネル内のメッセージングのアクセス許可を決定するために使用されます。MASTER を指定した場合、この API は、クライアントがこのシグナリングチャンネル上で任意のビューワーからオファーを受信したり、任意のビューワーに回答を送信したりするために使用できるエンドポイントを返します。VIEWER を指定した場合、この API は、クライアントがこのシグナリングチャンネル上で別の MASTER クライアントにオファーを送信するためにのみ使用できるエンドポイントを返します。

型: 文字列

有効な値: MASTER | VIEWER

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SingleMasterConfiguration

サービス: Amazon Kinesis Video Streams

SINGLE_MASTER チャンネルタイプの設定を含む構造体。

内容

MessageTtlSeconds

シグナリングチャンネルが、配信されていないメッセージを破棄するまで保持する期間 (秒単位)。 [UpdateSignalingChannel](#) を使用してこの値を更新します。

タイプ: 整数

値の範囲: 最小値 は 5 です。最大値は 120 です。

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

StreamInfo

サービス: Amazon Kinesis Video Streams

Kinesis ビデオストリームを記述するオブジェクト。

内容

CreationTime

ストリームがいつ作成されたかを示すタイムスタンプ。

型: タイムスタンプ

必須: いいえ

DataRetentionInHours

ストリームがデータを保持する期間 (時間単位)。

型: 整数

値の範囲: 最小値 は 0 です。

必須: いいえ

DeviceName

ストリームに関連付けられているデバイスの名前。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 128 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

KmsKeyId

Kinesis Video Streams がストリーム上のデータを暗号化するために使用する AWS Key Management Service (AWS KMS) キーの ID。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 2,048 です。

パターン: .+

必須: いいえ

MediaType

ストリームの MediaType。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 128 です。

パターン: `[\w\-\.\+]+/[\w\-\.\+]+(,[\w\-\.\+]+/[\w\-\.\+]+)*`

必須: いいえ

Status

ストリームのステータス。

型: 文字列

有効な値 : CREATING | ACTIVE | UPDATING | DELETING

必須 : いいえ

StreamARN

ストリームの Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

パターン: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

必須: いいえ

StreamName

ストリームの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

Version

ストリームのバージョン。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長 64

パターン: [a-zA-Z0-9]+

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

StreamNameCondition

サービス: Amazon Kinesis Video Streams

ストリームを一覧表示 (ListStreams API を参照) するときに返される、ストリームが満たす必要がある条件を指定します。条件には、比較演算と値が含まれています。現在指定できるのは、所定のプレフィックスで始まる名前のストリームを検索する BEGINS_WITH 演算子のみです。

内容

ComparisonOperator

比較演算子。現在指定できるのは、所定のプレフィックスで始まる名前のストリームを検索する BEGINS_WITH 演算子のみです。

型: 文字列

有効な値: BEGINS_WITH

必須: いいえ

ComparisonValue

比較する値。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

StreamStorageConfiguration

サービス: Amazon Kinesis Video Streams

ストリームデータのデフォルトのストレージ階層を含む、ストリームストレージの設定。この設定は、ストリームデータの保存方法とアクセス方法を決定し、さまざまな階層でさまざまなレベルのパフォーマンスとコスト最適化を提供します。

内容

DefaultStorageTier

ストリームデータのデフォルトのストレージ階層。この設定により、ストリームデータに使用されるストレージクラスが決まり、パフォーマンス特性とストレージコストの両方に影響します。

使用可能なストレージ階層:

- HOT - レイテンシーが最も低く、パフォーマンスが最も高い頻繁なアクセスに最適化されています。リアルタイムアプリケーションや頻繁にアクセスされるデータに最適です。
- WARM - アクセスが中程度のデータのパフォーマンスとコストのバランスが取れています。定期的にアクセスされるが、継続的にはアクセスされないデータに適しています。

型: 文字列

有効な値: HOT | WARM

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Tag

サービス: Amazon Kinesis Video Streams

指定したシグナリングチャンネルに関連付けられたキーと値のペア。

内容

Key

指定したシグナリングチャンネルに関連付けられたタグのキー。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 128 です。

パターン: `^([\p{L}\p{Z}\p{N}_.: /+=\ -@]*)$`

必須: はい

Value

指定したシグナリングチャンネルに関連付けられたタグの値。

タイプ: 文字列

長さの制約: 最小長は 0 です。最大長は 256 です。

パターン: `[\p{L}\p{Z}\p{N}_.: /+=\ -@]*`

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

UploaderConfig

サービス: Amazon Kinesis Video Streams

とScheduleExpression、カメラまたはローカルメディアファイルから Edge エージェントに記録するスケジューリングを指定するDurationInMinutes詳細で構成される設定。ScheduleConfig が指定されていない場合UploaderConfig、エッジエージェントは定期的に (1 時間ごとに) アップロードします。

内容

ScheduleConfig

とScheduleExpression、カメラまたはローカルメディアファイルから Edge エージェントに記録するスケジューリングを指定するDurationInMinutes詳細で構成される設定。こので指定されていない場合UploaderConfig、エッジエージェントScheduleConfigは定期的に (1 時間ごとに) アップロードします。

型: [ScheduleConfig](#) オブジェクト

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Amazon Kinesis Video Streams Media

Amazon Kinesis Video Streams Media では、次のデータ型がサポートされています。

- [StartSelector](#)

StartSelector

サービス: Amazon Kinesis Video Streams Media

GetMedia API がメディアデータの返送を開始する Kinesis ビデオストリームのチャンクを識別します。開始チャンクを識別するには、次のオプションがあります。

- 最後の (または最も古い) チャンクを選択します。
- 特定のチャンクを識別します。特定のチャンクを識別するには、フラグメント番号またはタイムスタンプ (サーバーまたはプロデューサー) を指定します。
- 各チャンクのメタデータには、Matroska (MKV) タグ (AWS_KINESISVIDEO_CONTINUATION_TOKEN) として継続トークンが含まれています。前の GetMedia リクエストが終了した場合、このタグ値を次の GetMedia リクエストで使用できます。次に、API は、最後の API が終了した場所からチャンクの返送を開始します。

内容

StartSelectorType

データの取得を開始する Kinesis ビデオストリーム上のフラグメントを識別します。

- NOW – ストリームの最後のチャンクから開始します。
- EARLIEST – ストリームの利用可能な最初のチャンクから開始します。
- FRAGMENT_NUMBER – 特定のフラグメントの後のチャンクから開始します。また、AfterFragmentNumber パラメータを指定する必要があります。
- PRODUCER_TIMESTAMP または SERVER_TIMESTAMP – 指定したプロデューサーまたはサーバーのタイムスタンプを持つフラグメントを含むチャンクから開始します。StartTimeStamp を追加してタイムスタンプを指定します。
- CONTINUATION_TOKEN – 指定した継続トークンを使用して読み込みます。

Note

NOW、EARLIEST、または CONTINUATION_TOKEN を startSelectorType として選択する場合、startSelector に追加情報を入力しません。

型: 文字列

有効な値: FRAGMENT_NUMBER | SERVER_TIMESTAMP | PRODUCER_TIMESTAMP | NOW | EARLIEST | CONTINUATION_TOKEN

必須: はい

AfterFragmentNumber

GetMedia API がフラグメントの返送を開始するフラグメント番号を指定します。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 128 です。

パターン: `^[0-9]+$`

必須: いいえ

ContinuationToken

Kinesis Video Streams が前の GetMedia 応答で返した継続トークン。次に、GetMedia API は、継続トークンで識別されるチャンクから開始します。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 128 です。

パターン: `^[a-zA-Z0-9_\.\\-]+$`

必須: いいえ

StartTimestamp

タイムスタンプ値。この値は、PRODUCER_TIMESTAMP または SERVER_TIMESTAMP を startSelectorType として選択した場合に必要です。次に、GetMedia API は、指定したタイムスタンプを持つフラグメントを含むチャンクから開始します。

型: タイムスタンプ

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Amazon Kinesis Video Streams Archived Media

Amazon Kinesis Video Streams Archived Media では、次のデータ型がサポートされています。

- [ClipFragmentSelector](#)
- [ClipTimestampRange](#)
- [DASHFragmentSelector](#)
- [DASHTimestampRange](#)
- [Fragment](#)
- [FragmentSelector](#)
- [HLSFragmentSelector](#)
- [HLSTimestampRange](#)
- [Image](#)
- [TimestampRange](#)

ClipFragmentSelector

サービス: Amazon Kinesis Video Streams Archived Media

フラグメントの範囲のタイムスタンプ範囲とタイムスタンプ発行元について説明します。

プロデューサーのタイムスタンプが重複しているフラグメントが、重複排除されます。つまり、プロデューサーが実際のクロック時間とほぼ等しいプロデューサーのタイムスタンプを持つフラグメントのストリームを生成している場合、クリップには要求されたタイムスタンプ範囲内のすべてのフラグメントが含まれることになります。一部のフラグメントが同じ時間範囲内の非常に異なる時点で取り込まれた場合、取り込まれた最も古いフラグメントのコレクションだけが返されます。

内容

FragmentSelectorType

使用するタイムスタンプ発行元 (サーバーまたはプロデューサー)。

型: 文字列

有効な値: PRODUCER_TIMESTAMP | SERVER_TIMESTAMP

必須: はい

TimestampRange

返されるタイムスタンプの範囲。

型: [ClipTimestampRange](#) オブジェクト

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ClipTimestampRange

サービス: Amazon Kinesis Video Streams Archived Media

フラグメントを返すタイムスタンプの範囲。

内容

EndTimeStamp

リクエストされたメディアのタイムスタンプ範囲の終了。

この値は、指定した StartTimestamp から 24 時間以内、かつ StartTimestamp 値より後である必要があります。リクエストの FragmentSelectorType が SERVER_TIMESTAMP の場合、この値は過去である必要があります。

この値は両端を含みます。EndTimeStamp は、フラグメントの (開始) タイムスタンプと比較されます。EndTimeStamp 値より前に始まり、それを過ぎて継続するフラグメントがセッションに含まれます。

型: タイムスタンプ

必須: はい

StartTimestamp

フラグメントを返すタイムスタンプの範囲にある開始タイムスタンプ。

StartTimestamp 以降で始まるフラグメントだけが、セッションに含まれます。StartTimestamp より前に始まり、それを過ぎて継続するフラグメントはセッションに含まれません。FragmentSelectorType が SERVER_TIMESTAMP の場合、StartTimestamp はストリームの先頭よりも後である必要があります。

型: タイムスタンプ

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DASHFragmentSelector

サービス: Amazon Kinesis Video Streams Archived Media

リクエストされたメディアのタイムスタンプの範囲とタイムスタンプの送信元が含まれます。

内容

FragmentSelectorType

リクエストされたメディアのタイムスタンプの送信元。

FragmentSelectorType が PRODUCER_TIMESTAMP に設定され、[GetDASHStreamingSessionURL:PlaybackMode](#) が ON_DEMAND または LIVE_REPLAY の場合、指定した [FragmentSelector:TimestampRange](#) 内のプロデューサーのタイムスタンプを持った状態で最初に取り込まれたフラグメントが、メディアプレイリストに含まれます。さらに、最初のフラグメントの直後 ([GetDASHStreamingSessionURL:MaxManifestFragmentResults](#) 値まで) に取り込まれた TimestampRange 内のプロデューサーのタイムスタンプを持つフラグメントが含まれます。

プロデューサーのタイムスタンプが重複しているフラグメントが、重複排除されます。つまり、プロデューサーが実際のクロック時間とほぼ等しいプロデューサーのタイムスタンプを持つフラグメントのストリームを生成している場合、MPEG-DASH マニフェストには要求されたタイムスタンプ範囲内のすべてのフラグメントが含まれることになります。一部のフラグメントが同じ時間範囲内の非常に異なる時点で取り込まれた場合、取り込まれた最も古いフラグメントのコレクションだけが返されます。

FragmentSelectorType が PRODUCER_TIMESTAMP に設定され、[GetDASHStreamingSessionURL:PlaybackMode](#) が LIVE の場合、プロデューサーのタイムスタンプが MP4 フラグメント内で重複排除のために使用されます。ただし、サーバーのタイムスタンプに基づいて最後に取り込まれたフラグメントが、MPEG-DASH マニフェストに含まれています。つまり、過去に取り込まれたフラグメントが現在値を含むプロデューサーのタイムスタンプを持つ場合でも、それらのフラグメントは HLS メディアプレイリストに含まれないことになります。

デフォルトは SERVER_TIMESTAMP です。

型: 文字列

有効な値: PRODUCER_TIMESTAMP | SERVER_TIMESTAMP

必須: いいえ

TimestampRange

リクエストされたメディアのタイムスタンプ範囲の開始と終了。

PlaybackType が LIVE の場合、この値を指定する必要はありません。

型: [DASHTimestampRange](#) オブジェクト

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DASHTimestampRange

サービス: Amazon Kinesis Video Streams Archived Media

リクエストされたメディアのタイムスタンプ範囲の開始と終了。

PlaybackType が LIVE の場合、この値を指定する必要はありません。

DASHTimestampRange の値は両端を含みます。開始時刻以降で始まるフラグメントが、セッションに含まれます。開始時刻より前に始まり、それを過ぎて継続するフラグメントはセッションに含まれません。

内容

EndTimeStamp

リクエストされたメディアのタイムスタンプ範囲の終了。この値は、指定した StartTimeStamp から 24 時間以内、かつ StartTimeStamp 値より後である必要があります。

リクエストの FragmentSelectorType が SERVER_TIMESTAMP の場合、この値は過去である必要があります。

EndTimeStamp 値は ON_DEMAND モードでは必須ですが、LIVE_REPLAY モードではオプションです。EndTimeStamp が LIVE_REPLAY モードで設定されていない場合、セッションが期限切れになるまで、新たにに取り込まれたフラグメントが継続してセッションに含まれます。

Note

この値は両端を含みます。EndTimeStamp は、フラグメントの (開始) タイムスタンプと比較されます。EndTimeStamp 値より前に始まり、それを過ぎて継続するフラグメントがセッションに含まれます。

型: タイムスタンプ

必須: いいえ

StartTimeStamp

リクエストされたメディアのタイムスタンプ範囲の開始。

DASHTimestampRange 値を指定した場合、StartTimeStamp 値が必要です。

StartTimeStamp 以降で始まるフラグメントだけが、セッションに含まれます。StartTimeStamp より前に始まり、それを過ぎて継続するフラグメントはセッションに含まれません。FragmentSelectorType が SERVER_TIMESTAMP の場合、StartTimeStamp はストリームの先頭よりも後である必要があります。

型: タイムスタンプ

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Fragment

サービス: Amazon Kinesis Video Streams Archived Media

ビデオなどの時間区切りデータのセグメントを表します。

内容

FragmentLengthInMilliseconds

フラグメントに関連付けられた再生時間などの時間値。

型: Long

必須: いいえ

FragmentNumber

フラグメントの一意の識別子。この値は、取り込み順序に基づいて一定間隔で増加します。

タイプ: 文字列

長さの制限: 最小長は 1 です。最大長は 128 です。

パターン: `^[0-9]+$`

必須: いいえ

FragmentSizeInBytes

フラグメントおよび含まれるメディアデータに関する情報を含む、フラグメントの合計サイズ。

型: Long

必須: いいえ

ProducerTimestamp

フラグメントに対応するプロデューサーからのタイムスタンプ。ミリ秒単位。

型: タイムスタンプ

必須: いいえ

ServerTimestamp

フラグメントに対応する AWS サーバーのタイムスタンプ。ミリ秒単位。

型: タイムスタンプ

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

FragmentSelector

サービス: Amazon Kinesis Video Streams Archived Media

フラグメントの範囲のタイムスタンプ範囲とタイムスタンプ発行元について説明します。

開始タイムスタンプが指定された開始時刻より後または同じ、かつ終了時刻より前または同じであるフラグメントだけが返されます。例えば、ストリームに次の開始タイムスタンプを持つフラグメントが含まれているとします。

- 00:00:00
- 00:00:02
- 00:00:04
- 00:00:06

フラグメントセレクトの範囲を開始時刻 00:00:01 および終了時刻 00:00:04 とすると、開始時刻が 00:00:02 と 00:00:04 のフラグメントを返します。

内容

FragmentSelectorType

使用するタイムスタンプ発行元 (サーバーまたはプロデューサー)。

型: 文字列

有効な値: PRODUCER_TIMESTAMP | SERVER_TIMESTAMP

必須: はい

TimestampRange

返されるタイムスタンプの範囲。

型: [TimestampRange](#) オブジェクト

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

HLSFragmentSelector

サービス: Amazon Kinesis Video Streams Archived Media

リクエストされたメディアのタイムスタンプの範囲とタイムスタンプの送信元が含まれます。

内容

FragmentSelectorType

リクエストされたメディアのタイムスタンプの送信元。

FragmentSelectorType が PRODUCER_TIMESTAMP に設定され、[GetHLSStreamingSessionURL:PlaybackMode](#) が ON_DEMAND または LIVE_REPLAY の場合、指定した [FragmentSelector:TimestampRange](#) 内のプロデューサーのタイムスタンプを持った状態で最初に取り込まれたフラグメントが、メディアプレイリストに含まれます。さらに、最初のフラグメントの直後 ([GetHLSStreamingSessionURL:MaxMediaPlaylistFragmentResults](#) 値まで) に取り込まれた TimestampRange 内のプロデューサーのタイムスタンプを持つフラグメントが含まれます。

プロデューサーのタイムスタンプが重複しているフラグメントが、重複排除されます。つまり、プロデューサーが実際のクロック時間とほぼ等しいプロデューサーのタイムスタンプを持つフラグメントのストリームを生成している場合、HLS メディアプレイリストには、要求されたタイムスタンプ範囲内のすべてのフラグメントが含まれることになります。一部のフラグメントが同じ時間範囲内の非常に異なる時点で取り込まれた場合、取り込まれた最も古いフラグメントのコレクションだけが返されます。

FragmentSelectorType が PRODUCER_TIMESTAMP に設定され、[GetHLSStreamingSessionURL:PlaybackMode](#) が LIVE の場合、プロデューサーのタイムスタンプは MP4 フラグメント内で重複排除のために使用されます。ただし、サーバーのタイムスタンプに基づいて最後に取り込まれたフラグメントが、HLS メディアプレイリストに含まれています。つまり、過去に取り込まれたフラグメントが現在値を含むプロデューサーのタイムスタンプを持つ場合でも、それらのフラグメントは HLS メディアプレイリストに含まれないことになります。

デフォルトは SERVER_TIMESTAMP です。

型: 文字列

有効な値: PRODUCER_TIMESTAMP | SERVER_TIMESTAMP

必須: いいえ

TimestampRange

リクエストされたメディアのタイムスタンプ範囲の開始と終了。

PlaybackType が LIVE の場合、この値を指定する必要はありません。

型: [HLSTimestampRange](#) オブジェクト

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

HLSTimestampRange

サービス: Amazon Kinesis Video Streams Archived Media

リクエストされたメディアのタイムスタンプ範囲の開始と終了。

PlaybackType が LIVE の場合、この値を指定する必要はありません。

内容

EndTimeStamp

リクエストされたメディアのタイムスタンプ範囲の終了。この値は、指定した StartTimeStamp から 24 時間以内、かつ StartTimeStamp 値より後である必要があります。

リクエストの FragmentSelectorType が SERVER_TIMESTAMP の場合、この値は過去である必要があります。

EndTimeStamp 値は ON_DEMAND モードでは必須ですが、LIVE_REPLAY モードではオプションです。EndTimeStamp が LIVE_REPLAY モードで設定されていない場合、セッションが期限切れになるまで、新たに取り込まれたフラグメントが継続してセッションに含まれます。

Note

この値は両端を含みます。EndTimeStamp は、フラグメントの (開始) タイムスタンプと比較されます。EndTimeStamp 値より前に始まり、それを過ぎて継続するフラグメントがセッションに含まれます。

型: タイムスタンプ

必須: いいえ

StartTimeStamp

リクエストされたメディアのタイムスタンプ範囲の開始。

HLSTimestampRange 値を指定した場合、StartTimeStamp 値が必要です。

StartTimeStamp 以降で始まるフラグメントだけが、セッションに含まれます。StartTimeStamp より前に始まり、それを過ぎて継続するフラグメントはセッションに含まれません。FragmentSelectorType が SERVER_TIMESTAMP の場合、StartTimeStamp はストリームの先頭よりも後である必要があります。

型: タイムスタンプ

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Image

サービス: Amazon Kinesis Video Streams Archived Media

Timestamp、Error、および を含む構造 ImageContent。

内容

Error

指定されたタイムスタンプのイメージが再試行不可能なエラーのために抽出されなかったときに表示されるエラーメッセージ。次の場合、エラーが返されます。

- 指定された にはメディアが存在しません Timestamp。
- 指定された時間のメディアでは、イメージを抽出することはできません。この場合、メディアは音声のみであるか、誤ったメディアが取り込まれています。

型: 文字列

有効な値 : NO_MEDIA | MEDIA_ERROR

必須: いいえ

ImageContent

Base64 でエンコードされた Image オブジェクトの属性。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 6291456 です。

必須: いいえ

TimeStamp

ビデオストリームからイメージを抽出するために使用される Image オブジェクトの属性。このフィールドは、イメージのギャップを管理したり、ページ分割ウィンドウをよりよく理解したりするために使用されます。

型: タイムスタンプ

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

TimestampRange

サービス: Amazon Kinesis Video Streams Archived Media

フラグメントを返すタイムスタンプの範囲。

内容

EndTimeStamp

フラグメントを返すタイムスタンプの範囲内にある終了タイムスタンプ。

型: タイムスタンプ

必須: はい

StartTimeStamp

フラグメントを返すタイムスタンプの範囲にある開始タイムスタンプ。

型: タイムスタンプ

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Amazon Kinesis Video Signaling Channels

Amazon Kinesis Video Signaling Channels では、次のデータ型がサポートされています。

- [IceServer](#)

IceServer

サービス: Amazon Kinesis Video Signaling Channels

ICE サーバーの接続データのための構造体。

内容

Password

ICE サーバーにログインするためのパスワード。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

Ttl

ユーザー名とパスワードの有効期間 (秒単位)。

タイプ: 整数

有効範囲: 固定値は 300 です。

必須: いいえ

Uris

[I-D.petithuguenin-behave-turn-uris](#) 仕様で指定された形式の URI の配列。これらの URI では、TURN サーバーに到達するために使用できるさまざまなアドレスおよび/またはプロトコルが指定されます。

型: 文字列の配列

長さの制限: 最小長は 1 です。最大長は 256 です。

必須: いいえ

Username

ICE サーバーにログインするためのユーザー名。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

パターン: [a-zA-Z0-9_.-]+

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用方法の詳細については、以下を参照してください。

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Amazon Kinesis Video WebRTC ストレージ

Amazon Kinesis Video WebRTC Storage では、次のデータ型がサポートされています。

共通エラー

このセクションでは、すべての AWS サービスの API アクションに共通するエラーを一覧表示します。このサービスの API アクションに固有のエラーについては、その API アクションのトピックを参照してください。

AccessDeniedException

このアクションを実行する十分なアクセス権限がありません。

HTTP ステータスコード: 400

IncompleteSignature

リクエスト署名が AWS 標準に準拠していません。

HTTP ステータスコード: 400

InternalFailure

リクエストの処理が、不明なエラー、例外、または障害により実行できませんでした。

HTTP ステータスコード: 500

InvalidAction

リクエストされたアクション、またはオペレーションは無効です。アクションが正しく入力されていることを確認してください。

HTTP ステータスコード: 400

InvalidClientId

提供された X.509 証明書または AWS アクセスキー ID がレコードに存在しません。

HTTP ステータスコード: 403

NotAuthorized

このアクションを実行するためのアクセス許可がありません。

HTTP ステータスコード: 400

OptInRequired

AWS アクセスキー ID には、サービスのサブスクリプションが必要です。

HTTP ステータスコード: 403

RequestExpired

リクエストの日付スタンプの 15 分以上後またはリクエストの有効期限 (署名付き URL の場合など) の 15 分以上後に、リクエストが到着しました。または、リクエストの日付スタンプが現在より 15 分以上先です。

HTTP ステータスコード: 400

ServiceUnavailable

サーバーの一時的な障害により、リクエストは失敗しました。

HTTP ステータスコード: 503

ThrottlingException

リクエストのスロットリングにより、リクエストが拒否されました。

HTTP ステータスコード: 400

ValidationError

入力が AWS サービスで指定された制約を満たしていません。

HTTP ステータスコード: 400

共通パラメータ

次のリストには、すべてのアクションが署名バージョン 4 リクエストにクエリ文字列で署名するために使用するパラメータを示します。アクション固有のパラメータは、アクションのトピックに示されています。署名バージョン 4 の詳細については、IAM ユーザーガイド [の AWS API リクエストの署名](#) を参照してください。

Action

実行するアクション。

型: 文字列

必須: はい

Version

リクエストが想定している API バージョンである、YYYY-MM-DD 形式で表示されます。

型: 文字列

必須: はい

X-Amz-Algorithm

リクエストの署名を作成するのに使用したハッシュアルゴリズム。

条件: HTTP 認可ヘッダーではなくクエリ文字列に認証情報を含める場合は、このパラメータを指定します。

型: 文字列

有効な値 : AWS4-HMAC-SHA256

必須: 条件による

X-Amz-Credential

認証情報スコープの値で、アクセスキー、日付、対象とするリージョン、リクエストしているサービス、および終了文字列 ("aws4_request") を含む文字列です。値は次の形式で表現されます: [access_key/YYYYMMDD/リージョン/サービス/aws4_request]。

詳細については、IAM [ユーザーガイドの「署名付き AWS API リクエストの作成」](#)を参照してください。

条件: HTTP 認可ヘッダーではなくクエリ文字列に認証情報を含める場合は、このパラメータを指定します。

型: 文字列

必須: 条件による

X-Amz-Date

署名を作成するときに使用する日付です。形式は ISO 8601 基本形式の YYYYMMDD'T'HHMMSS'Z' でなければなりません。例えば、日付 20120325T120000Z は、有効な X-Amz-Date の値です。

条件: X-Amz-Date はすべてのリクエストに対してオプションです。署名リクエストで使用する日付よりも優先される日付として使用できます。ISO 8601 ベーシック形式で日付ヘッダーが指定されている場合、X-Amz-Date は必要ありません。X-Amz-Date を使用すると、常に Date ヘッダーの値よりも優先されます。詳細については、IAM ユーザーガイドの[AWS API リクエスト署名の要素](#)を参照してください。

タイプ: 文字列

必須: 条件による

X-Amz-Security-Token

AWS Security Token Service () の呼び出しによって取得された一時的なセキュリティトークン AWS STS。AWS STSの一時的なセキュリティ認証情報をサポートするサービスのリストについては、「IAM ユーザーガイド」の「[IAM と連携するAWS のサービス](#)」を参照してください。

条件: から一時的なセキュリティ認証情報を使用している場合は AWS STS、セキュリティトークンを含める必要があります。

タイプ: 文字列

必須: 条件による

X-Amz-Signature

署名する文字列と派生署名キーから計算された 16 進符号化署名を指定します。

条件: HTTP 認可ヘッダーではなくクエリ文字列に認証情報を含める場合は、このパラメータを指定します。

型: 文字列

必須: 条件による

X-Amz-SignedHeaders

正規リクエストの一部として含まれていたすべての HTTP ヘッダーを指定します。署名付きヘッダーの指定の詳細については、IAM [ユーザーガイドの「署名付き AWS API リクエストの作成」](#)を参照してください。

条件: HTTP 認可ヘッダーではなくクエリ文字列に認証情報を含める場合は、このパラメータを指定します。

型: 文字列

必須: 条件による

Amazon Kinesis Video Streams のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS、最もセキュリティの影響を受けやすい組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、AWS お客様とお客様の間の責任共有です。[責任共有モデル](#)では、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ - クラウドで AWS AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。は、お客様が安全に使用できるサービス AWS も提供します。セキュリティの有効性は、[AWS コンプライアンスプログラム](#)の一環として、サードパーティーの審査機関によって定期的にテストおよび検証されています。Kinesis Video Streams に適用されるコンプライアンスプログラムについては、「[コンプライアンスプログラムによるAWS 対象範囲内のサービス](#)」を参照してください。
- クラウド内のセキュリティ - お客様の責任は、使用する AWS サービスによって決まります。また、お客様は、お客様のデータの機密性、組織の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

このドキュメントは、Kinesis Video Streams を使用する際に責任共有モデルを適用する方法を理解するのに役立ちます。次のトピックでは、セキュリティおよびコンプライアンスの目的を達成するように Kinesis Video Streams を設定する方法を説明します。また、Kinesis Video Streams リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

- [Kinesis Video Streams でのデータ保護](#)
- [IAM を使用した Kinesis Video Streams リソースへのアクセスの制御](#)
- [を使用した Kinesis Video Streams リソースへのアクセスの制御 AWS IoT](#)
- [Amazon Kinesis Video Streams のコンプライアンス検証](#)
- [Amazon Kinesis Video Streams の耐障害性](#)
- [Kinesis Video Streams のインフラストラクチャセキュリティ](#)
- [Kinesis Video Streams のセキュリティのベストプラクティス](#)

Kinesis Video Streams でのデータ保護

Amazon Kinesis Video Streams で保管中のデータを暗号化することで、AWS Key Management Service () キーを使用したサーバー側の暗号化 (SSE AWS KMS) を使用して、厳格なデータ管理要件を満たすことができます。

トピック

- [Kinesis Video Streams のサーバー側の暗号化とは](#)
- [コスト、リージョン、およびパフォーマンスに関する考慮事項](#)
- [サーバー側の暗号化の使用開始方法](#)
- [カスタマーマネージドキーの作成と使用](#)
- [カスタマーマネージドキーを使用するためのアクセス許可](#)

Kinesis Video Streams のサーバー側の暗号化とは

サーバー側の暗号化は、Kinesis Video Streams の機能であり、指定したキーを使用して AWS KMS 保管時にデータを保存する前に自動的に暗号化します。データが Kinesis Video Streams ストリームストレージレイヤーに書き込まれる前に暗号化され、ストレージから取得された後で復号されます。その結果、Kinesis Video Streams サービス内で保管中のデータは常に暗号化されます。

サーバー側の暗号化を使用すると、Kinesis ビデオストリームプロデューサーとコンシューマーは KMS キーや暗号化オペレーションを管理する必要はありません。データ保持が有効になっている場合、データは Kinesis Video Streams に出入りするときに自動的に暗号化されるため、保管中のデータは暗号化されます。は、サーバー側の暗号化機能で使用するすべてのキー AWS KMS を提供します。は AWS、AWS KMS サービスにインポートされたユーザー指定の AWS KMS キーである Kinesis Video Streams の KMS キーの使用を AWS KMS ストリーミングします。

コスト、リージョン、およびパフォーマンスに関する考慮事項

サーバー側の暗号化を適用すると、AWS KMS API の使用とキーコストが適用されます。カスタム AWS KMS キーとは異なり、デフォルトの KMS aws/kinesisvideo キーは無料で提供されます。ただし、お客様に代わって Kinesis Video Streams で発生した API の使用コストを支払う必要があります。

API の使用コストは、カスタムキーを含むすべての KMS キーに適用されます。AWS KMS コストは、データプロデューサーとコンシューマーで使用するユーザー認証情報の数に応じて変動します。各ユーザー認証情報には一意の API コールが必要なためです AWS KMS。

以下は、リソース別の料金の説明です。

キー

- (AWS エイリアス = aws/kinesisvideo) によって管理される Kinesis Video Streams の KMS キーには料金はかかりません。
- ユーザー生成の KMS キーには AWS KMS key コストがかかります。詳細については、「[AWS Key Management Service 料金](#)」を参照してください。

AWS KMS API の使用

新しいデータ暗号化キーを生成する API リクエスト、またはトラフィックの増加に応じて既存の暗号化キーを取得する API リクエスト。AWS KMS 使用コストがかかります。詳細については、[AWS Key Management Service 「料金: 使用状況」](#)を参照してください。

Kinesis Video Streams が、保持期間が「0」(保持期間なし)に設定されている場合でもキーリクエストを生成します。

リージョン別のサーバー側暗号化の利用可能性

Kinesis Video Streams のサーバー側の暗号化は、Kinesis Video Streams AWS リージョン が利用可能なすべての で使用できます。

サーバー側の暗号化の使用開始方法

サーバー側の暗号化は、Kinesis Video Streams で常に有効になります。ストリームの作成時にユーザー提供のキーが指定されていない場合は、AWS マネージドキー (Kinesis Video Streams が提供する) が使用されます。

ユーザー提供の KMS キーは、作成時に Kinesis ビデオストリームに割り当てする必要があります。後で [UpdateStream](#) API を使用してストリームに別のキーを割り当ててはできません。

ユーザー提供の KMS キーを Kinesis ビデオストリームに割り当てするには、次の 2 つの方法があります。

- で Kinesis ビデオストリームを作成するときは AWS マネジメントコンソール、新しいビデオストリームの作成ページの暗号化タブで KMS キーを指定します。
- [CreateStream](#) API を使用して Kinesis ビデオストリームを作成する際に、KmsKeyId パラメータでキー ID を指定します。

カスタマーマネージドキーの作成と使用

このセクションでは、Amazon Kinesis Video Streams によって管理されるキーを使用する代わりに、独自の KMS キーを作成して使用方法について説明します。

カスタマーマネージドキーの作成

独自のキーを作成する方法については、「AWS Key Management Service デベロッパーガイド」の「[キーの作成](#)」を参照してください。アカウントのキーを作成すると、Kinesis Video Streams サービスはこれらのキーをカスタマーマネージドキーリストに返します。

カスタマーマネージドキーを使用する

コンシューマー、プロデューサー、管理者に正しいアクセス許可が適用されたら、独自の AWS アカウント または別の でカスタム KMS キーを使用できます AWS アカウント。アカウントのすべての KMS キーは、コンソールのカスタマーマネージドキーリストに表示されます。

別のアカウントにあるカスタム KMS キーを使用するには、それらのキーを使用するためのアクセス許可が必要です。また、CreateStream API を使用してストリームを作成する必要もあります。コンソールで作成されたストリームでは、異なるアカウントの KMS キーを使用することはできません。

Note

KMS キーは、PutMediaまたは GetMediaオペレーションが実行されるまでアクセスされません。その結果、次のことが起こります。

- 指定したキーが存在しない場合、CreateStreamオペレーションは成功しますが、ストリームに対するGetMediaオペレーションPutMediaは失敗します。
- 指定されたキー (aws/kinesisvideo) を使用する場合、最初の PutMediaまたは GetMediaオペレーションが実行されるまで、キーはアカウントに存在しません。

カスタマーマネージドキーを使用するためのアクセス許可

カスタマーマネージドキーでサーバー側の暗号化を使用する前に、ストリームの暗号化とストリームレコードの暗号化と復号を許可するように KMS キーポリシーを設定する必要があります。アクセス AWS KMS 許可の例と詳細については、[AWS KMS 「API のアクセス許可: アクションとリソースのリファレンス」](#)を参照してください。

 Note

暗号化にデフォルトのサービスキーを使用する場合、カスタム IAM アクセス許可を適用する必要はありません。

カスタマーマネージドキーを使用する前に、Kinesis ビデオストリームプロデューサーとコンシューマー (IAM プリンシパル) が AWS KMS デフォルトのキーポリシーのユーザーであることを確認します。ユーザーになっていない場合は、ストリームに対する読み取りと書き込みが失敗し、最終的にはデータの損失、処理の遅延、またはアプリケーションのハングにつながる可能性があります。IAM ポリシーを使用して KMS キーの許可を管理できます。詳細については、「[での IAM ポリシーの使用 AWS KMS](#)」を参照してください。

プロデューサーのアクセス許可の例

Kinesis ビデオストリームプロデューサーには `kms:GenerateDataKey` アクセス許可が必要です。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey"
      ],
      "Resource": "arn:aws:kms:us-west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:PutMedia"
      ],
      "Resource": "arn:aws:kinesisvideo:*:123456789012:stream/MyStream/*"
    }
  ]
}
```


コンシューマーのアクセス許可の例

Kinesis ビデオストリームコンシューマーには `kms:Decrypt` アクセス許可が必要です。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:us-west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:GetMedia"
      ],
      "Resource": "arn:aws:kinesisvideo:*:123456789012:stream/MyStream/*"
    }
  ]
}
```

IAM を使用した Kinesis Video Streams リソースへのアクセスの制御

Amazon Kinesis Video Streams で AWS Identity and Access Management (IAM) を使用すると、組織内のユーザーが特定の Kinesis Video Streams API オペレーションを使用してタスクを実行できるかどうか、および特定の AWS リソースを使用できるかどうかを制御できます。

IAM の詳細については、以下を参照してください。

- [AWS Identity and Access Management \(IAM\)](#)
- [IAM の開始方法](#)
- [IAM ユーザーガイド](#)

内容

- [ポリシー構文](#)
- [Kinesis Video Streams のアクション](#)
- [Kinesis Video Streams の Amazon リソースネーム \(ARN\)](#)
- [他の IAM アカウントに Kinesis ビデオストリームへのアクセスを許可する](#)
- [Kinesis Video Streams のポリシーの例](#)

ポリシー構文

IAM ポリシーは、1 つ、または複数のステートメントで構成される JSON ドキュメントです。各ステートメントは次のように構成されます。

```
{
  "Statement": [{
    "Effect": "effect",
    "Action": "action",
    "Resource": "arn",
    "Condition": {
      "condition": {
        "key": "value"
      }
    }
  ]
}
```

ステートメントはさまざまなエレメントで構成されています。

- 効果 – 効果は Allow または Deny です。デフォルトでは ユーザーはリソースおよび API アクションを使用するアクセス許可がないため、リクエストはすべて拒否されます。明示的な許可はデフォルトに上書きされます。明示的な拒否はすべての許可に上書きされます。
- アクション – アクションは、アクセス許可を付与または拒否する特定の API アクションです。
- リソース – アクションの影響を受けるリソース。ステートメント内でリソースを指定するには、Amazon リソースネーム (ARN) を使用する必要があります。
- 条件 – 条件はオプションです。ポリシーの発効条件を指定するために使用します。

IAM ポリシーを作成および管理するときは、[IAM ポリシージェネレーター](#)と [IAM ポリシーシミュレーター](#)を使用することをお勧めします。

Kinesis Video Streams のアクション

IAM ポリシーステートメントで、IAM をサポートするすべてのサービスからの任意の API アクションを指定できます。Kinesis Video Streams の場合、API アクションの名前に次のプレフィックス (kinesisvideo:) を使用します。例えば、kinesisvideo:CreateStream、kinesisvideo:ListStreams、および kinesisvideo:DescribeStream のようになります。

単一のステートメントで複数のアクションを指定するには、次のようにカンマで区切ります。

```
"Action": ["kinesisvideo:action1", "kinesisvideo:action2"]
```

ワイルドカードを使用して複数のアクションを指定することもできます。たとえば、Get という単語で始まる名前のすべてのアクションは、以下のように指定できます。

```
"Action": "kinesisvideo:Get*"
```

すべての Kinesis Video Streams の操作を指定するには、次のようにアスタリスク (*) ワイルドカードを使用します。

```
"Action": "kinesisvideo:*"
```

Kinesis Video Streams API アクションの一覧については、「[Kinesis Video Streams API リファレンス](#)」を参照してください。

Kinesis Video Streams の Amazon リソースネーム (ARN)

各 IAM ポリシーステートメントは ARN を使用して指定したリソースに適用されます。

Kinesis Video Streams には、次の ARN リソースフォーマットを使用します。

```
arn:aws:kinesisvideo:region:account-id:stream/stream-name/code
```

例:

```
"Resource": arn:aws:kinesisvideo:*:111122223333:stream/my-stream/0123456789012
```

ストリームの ARN は [DescribeStream](#) を使用することで取得できます。

他の IAM アカウントに Kinesis ビデオストリームへのアクセスを許可する

Kinesis Video Streams のストリームでオペレーションを実行するには、他の IAM アカウントに許可を付与する必要がある場合があります。次の概要では、アカウント間でビデオストリームへのアクセス許可を付与するための一般的なステップを説明します。

1. アカウントで作成されたストリームリソースでオペレーションを実行するアクセス許可を付与するアカウントの 12 桁のアカウント ID を取得します。

例: 次のステップでは、アクセス許可を付与するアカウントのアカウント ID として 111111111111 を使用し、Kinesis Video Streams の ID として 999999999999 を使用します。

2. 付与するアクセスレベルを許可するストリーム (999999999999) を所有するアカウントに IAM 管理ポリシーを作成します。

サンプルポリシー:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:GetDataEndpoint",
        "kinesisvideo:DescribeStream",
        "kinesisvideo:PutMedia"
      ],
      "Resource": "arn:aws:kinesisvideo:us-west-2:999999999999:stream/custom-stream-name/1613732218179"
    }
  ]
}
```

Kinesis Video Streams リソースのその他のポリシーの例については、次のセクション [ポリシーの例](#) の「」を参照してください。

3. ストリーム (999999999999) を所有するアカウントにロールを作成し、(111111111111) のアクセス許可を付与するアカウントを指定します。これにより、信頼されたエンティティがロールに追加されます。

信頼されたポリシーの例:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111111111111:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

前のステップで作成したポリシーをこのロールにアタッチします。

これで、アカウント 999999999999 でロールが作成されました。このロールには DescribeStream、管理ポリシーのストリームリソース ARN PutMedia に対する、GetDataEndpoint、などのオペレーションに対するアクセス許可があります。この新しいロールは、他のアカウントである 111111111111 も信頼してこのロールを引き受けます。

 Important

ロール ARN を書き留めておきます。次のステップで必要になります。

4. 前のステップでアカウント 111111111111 で作成したロールに対する AssumeRole アクションを許可する管理ポリシーを、他のアカウント 999999999999 に作成します。前のステップのロール ARN について言及する必要があります。

サンプルポリシー:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::999999999999:role/CustomRoleName"
  }
}
```

5. 前のステップで作成したポリシーを、ロールやアカウント 111111111111 のユーザーなどの IAM エンティティにアタッチします。このユーザーは、アカウント 999999999999 CustomRoleName でロールを引き受けるアクセス許可を持つようになりました。

このユーザーの認証情報は API を呼び出し AWS STS AssumeRole でセッション認証情報を取得し、その後、アカウント 999999999999 で作成されたストリームで Kinesis Video Streams APIs を呼び出すために使用されます。

```
aws sts assume-role --role-arn "arn:aws:iam::999999999999:role/CustomRoleName" --
role-session-name "kvs-cross-account-assume-role"
{
  "Credentials": {
    "AccessKeyId": "",
    "SecretAccessKey": "",
    "SessionToken": "",
    "Expiration": ""
  },
  "AssumedRoleUser": {
    "AssumedRoleId": "",
    "Arn": ""
  }
}
```

6. 環境内の前のセットに基づいて、アクセスキー、シークレットキー、およびセッション認証情報を設定します。

```
set AWS_ACCESS_KEY_ID=
set AWS_SECRET_ACCESS_KEY=
set AWS_SESSION_TOKEN=
```

7. Kinesis Video Streams APIs を実行して、アカウント 999999999999 のストリームのデータエンドポイントを記述して取得します。

```
aws kinesisvideo describe-stream --stream-arn "arn:aws:kinesisvideo:us-west-2:999999999999:stream/custom-stream-name/1613732218179"
{
  "StreamInfo": {
    "StreamName": "custom-stream-name",
    "StreamARN": "arn:aws:kinesisvideo:us-west-2:999999999999:stream/custom-stream-name/1613732218179",
    "KmsKeyId": "arn:aws:kms:us-west-2:999999999999:alias/aws/kinesisvideo",
    "Version": "abcd",
    "Status": "ACTIVE",
    "CreationTime": "2018-02-19T10:56:58.179000+00:00",
    "DataRetentionInHours": 24
  }
}

aws kinesisvideo get-data-endpoint --stream-arn "arn:aws:kinesisvideo:us-west-2:999999999999:stream/custom-stream-name/1613732218179" --api-name "PUT_MEDIA"
{
  "DataEndpoint": "https://s-b12345.kinesisvideo.us-west-2.amazonaws.com"
}
```

クロスアカウントアクセスを許可する一般的なstep-by-stepについては、[「IAM ロール AWS アカウント を使用した 間のアクセスの委任」](#)を参照してください。

Kinesis Video Streams のポリシーの例

次のポリシー例は、Kinesis Video Streams へのユーザーアクセスを制御する方法を示しています。

Example 1: ユーザーに Kinesis ビデオストリームからのデータの取得を許可する

このポリシーにより、ユーザーまたはグループが任意の Kinesis ビデオストリームに対して DescribeStream、GetDataEndpoint、GetMedia、ListStreams、および ListTagsForStream の操作を実行できます。このポリシーは、任意のビデオストリームからデータを取得できるユーザーに適しています。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:Describe*",
        "kinesisvideo:Get*",
        "kinesisvideo:List*"
      ],
      "Resource": "*"
    }
  ]
}
```

Example 2: ユーザーに Kinesis ビデオストリームの作成とビデオストリームへのデータの書き込みを許可する

このポリシーにより、ユーザーまたはグループは `CreateStream` および `PutMedia` の操作を実行できます。このポリシーは、ビデオストリームを作成し、それにデータを送信できる監視カメラに適しています。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:CreateStream",
        "kinesisvideo:PutMedia"
      ],
      "Resource": "*"
    }
  ]
}
```


Example 3: すべての Kinesis Video Streams リソースへのフルアクセスをユーザーに許可する

このポリシーにより、ユーザーまたはグループが任意のリソースに対して任意の Kinesis Video Streams オペレーションを実行できます。このポリシーは、管理者に適しています。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesisvideo:*",
      "Resource": "*"
    }
  ]
}
```

Example 4: ユーザーに特定の Kinesis ビデオストリームへのデータの書き込みを許可する

このポリシーにより、ユーザーまたはグループは特定のビデオストリームにデータを書き込むことができます。このポリシーは、1 つのストリームにデータを送信できるデバイスに適しています。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesisvideo:PutMedia",
      "Resource": "arn:aws:kinesisvideo:us-west-2:123456789012:stream/your_stream/0123456789012"
    }
  ]
}
```

を使用した Kinesis Video Streams リソースへのアクセスの制御 AWS IoT

このセクションでは、デバイス (カメラなど) がオーディオおよびビデオデータを 1 つの特定の Kinesis ビデオストリームにのみ送信できるようにする方法について説明します。これを行うには、AWS IoT 認証情報プロバイダーと AWS Identity and Access Management (IAM) ロールを使用します。

デバイスは X.509 証明書を使用して、TLS 相互認証プロトコル AWS IoT を使用して に接続できます。他の AWS のサービス (Kinesis Video Streams など) は証明書ベースの認証をサポートしていませんが、AWS 署名バージョン 4 形式の認証情報を使用して AWS 呼び出すことができます。署名バージョン 4 アルゴリズムでは、通常、呼び出し元にアクセスキー ID とシークレットアクセスキーが必要です。には、組み込みの X.509 証明書を一意のデバイス ID として使用して AWS リクエスト (Kinesis Video Streams へのリクエストなど) を認証できる認証情報プロバイダー AWS IoT があります。これにより、アクセスキー ID とシークレットアクセスキーをデバイスに保存する必要がなくなります。

認証情報プロバイダーは、X.509 証明書を使用してクライアント (この場合は、ビデオストリームにデータを送信するカメラで実行されている Kinesis Video Streams SDK) を認証し、権限が制限された一時的なセキュリティトークンを発行します。トークンを使用して、任意の AWS リクエスト (この場合は Kinesis Video Streams への呼び出し) に署名して認証できます。詳細については、「[AWS サービスへの直接呼び出しの承認](#)」を参照してください。

このようにしてカメラの Kinesis Video Streams へのリクエストを認証するには、IAM ロールを作成して設定し、適切な IAM ポリシーをロールにアタッチして、AWS IoT 認証情報プロバイダーがユーザーに代わってロールを引き受けられるようにする必要があります。

詳細については AWS IoT、[AWS IoT Core 「ドキュメント」](#) を参照してください。IAM の詳細については、[AWS Identity and Access Management \(IAM\)](#) を参照してください。

トピック

- [ストリーム名としてのAWS IoT ThingName](#)
- [ストリーム名としてのAWS IoT Certificateld](#)
- [AWS IoT 認証情報を使用してハードコードされたストリーム名にストリーミングする](#)

ストリーム名としてのAWS IoT ThingName

トピック

- [ステップ 1: AWS IoT モノのタイプと AWS IoT モノを作成する](#)
- [ステップ 2: 引き受ける IAM ロールを作成する AWS IoT](#)
- [ステップ 3: X.509 証明書を作成して設定する](#)
- [ステップ 4: Kinesis ビデオストリームで AWS IoT 認証情報をテストする](#)
- [ステップ 5: カメラのファイルシステムに AWS IoT 証明書と認証情報をデプロイし、データをビデオストリームにストリーミングする](#)

ステップ 1: AWS IoT モノのタイプと AWS IoT モノを作成する

では AWS IoT、モノは特定のデバイスまたは論理エンティティを表します。この場合、AWS IoT モノは、リソースレベルのアクセスコントロールを設定する Kinesis ビデオストリームを表します。モノを作成するには、まず AWS IoT モノのタイプを作成する必要があります。AWS IoT モノのタイプを使用して、同じモノのタイプに関連付けられているすべてのモノに共通の説明と設定情報を保存できます。

1. 次のコマンド例では、モノのタイプ `kvs_example_camera` が作成されます。

```
aws --profile default iot create-thing-type --thing-type-name kvs_example_camera >
iot-thing-type.json
```

2. このコマンド例では、`kvs_example_camera_stream` モノタイプの `kvs_example_camera` モノを作成します。

```
aws --profile default iot create-thing --thing-name kvs_example_camera_stream --
thing-type-name kvs_example_camera > iot-thing.json
```

ステップ 2: 引き受ける IAM ロールを作成する AWS IoT

IAM ロールはユーザーと似ています。ロールは、AWS ID ができることとできないことを決定するアクセス許可ポリシーを持つ ID です AWS。ロールは、そのロールを必要とするどのユーザーでも引き受けることができます。ロールを引き受けると、ロールセッション用の一時的なセキュリティ認証情報が提供されます。

このステップで作成するロールは、 が引き受け AWS IoT で、クライアントから認証情報認可リクエストを実行するときに、セキュリティトークンサービス (STS) から一時的な認証情報を取得できます。この場合、クライアントはカメラで実行されている Kinesis Video Streams SDK です。

この IAM ロールを作成して設定するには、以下のステップを実行します。

1. IAM ロールを作成します。

次のコマンド例では、KVSCameraCertificateBasedIAMRoleという IAM ロールが作成されます。

```
aws --profile default iam create-role --role-name KVSCameraCertificateBasedIAMRole
--assume-role-policy-document 'file://iam-policy-document.json' > iam-role.json
```

iam-policy-document.json には、次の信頼ポリシー JSON を使用できます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 次に、以前に作成した IAM ロールにアクセス許可ポリシーをアタッチします。このアクセス許可ポリシーは、AWS リソースの選択的なアクセスコントロール (サポートされているオペレーションのサブセット) を許可します。この場合、AWS リソースはカメラがデータを送信するビデオストリームです。つまり、すべての設定ステップが完了すると、このカメラはこのビデオストリームにのみデータを送信できるようになります。

```
aws --profile default iam put-role-policy --role-name
KVSCameraCertificateBasedIAMRole --policy-name KVSCameraIAMPolicy --policy-
document 'file://iam-permission-document.json'
```

には、次の IAM ポリシー JSON を使用できます `iam-permission-document.json`。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:DescribeStream",
        "kinesisvideo:PutMedia",
        "kinesisvideo:TagStream",
        "kinesisvideo:GetDataEndpoint"
      ],
      "Resource": "arn:aws:kinesisvideo:*:*:stream/${credentials-
        iot:ThingName}/*"
    }
  ]
}
```

このポリシーは、プレースホルダーによって指定されたビデオストリーム (AWS リソース) でのみ、指定されたアクションを承認することに注意してください(`${credentials-iot:ThingName}`)。このプレースホルダーは、認証情報プロバイダーがリクエストでビデオストリーム名を送信する `ThingName` ときに AWS IoT、AWS IoT thing 属性の値を受け取ります。

- 次に、IAM ロールのロールエイリアスを作成します。ロールエイリアスは、IAM ロールをポイントする代替データモデルです。AWS IoT 認証情報プロバイダーリクエストには、STS から一時的な認証情報を取得するために引き受ける IAM ロールを示すロールエイリアスを含める必要があります。

次のサンプルコマンドでは、`KvsCameraIoTRoleAlias` というロールエイリアスが作成されます。

```
aws --profile default iot create-role-alias --role-alias KvsCameraIoTRoleAlias --
role-arn $(jq --raw-output '.Role.Arn' iam-role.json) --credential-duration-seconds
3600 > iot-role-alias.json
```

4. これで、ガロールエイリアスを使用して (アタッチされた後に) 証明書でロールを AWS IoT 引き受けることができるポリシーを作成できます。

次のサンプルコマンドは、AWS IoT という のポリシーを作成しますKvsCameraIoTPolicy。

```
aws --profile default iot create-policy --policy-name KvsCameraIoTPolicy --policy-document 'file:///iot-policy-document.json'
```

次のコマンドを使用して、iot-policy-document.jsonドキュメント JSON を作成できます。

```
cat > iot-policy-document.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:AssumeRoleWithCertificate"
      ],
      "Resource": "$(jq --raw-output '.roleAliasArn' iot-role-alias.json)"
    }
  ]
}
EOF
```

ステップ 3: X.509 証明書を作成して設定する

デバイス (ビデオストリーム) と 間の通信 AWS IoT は、X.509 証明書を使用して保護されます。

1. AWS IoT 以前に作成した のポリシーをアタッチする必要がある証明書を作成します。

```
aws --profile default iot create-keys-and-certificate --set-as-active --certificate-pem-outfile certificate.pem --public-key-outfile public.pem.key --private-key-outfile private.pem.key > certificate
```

2. AWS IoT (KvsCameraIoTPolicy以前に作成した) のポリシーをこの証明書にアタッチします。

```
aws --profile default iot attach-policy --policy-name KvsCameraIoTPolicy --target  
$(jq --raw-output '.certificateArn' certificate)
```

3. 作成した証明書に AWS IoT モノ (kvs_example_camera_stream) をアタッチします。

```
aws --profile default iot attach-thing-principal --thing-name  
kvs_example_camera_stream --principal $(jq --raw-output '.certificateArn'  
certificate)
```

4. AWS IoT 認証情報プロバイダーを介してリクエストを承認するには、AWS アカウント ID に固有の AWS IoT 認証情報エンドポイントが必要です。次のコマンドを使用して、AWS IoT 認証情報エンドポイントを取得できます。

```
aws --profile default iot describe-endpoint --endpoint-type iot:CredentialProvider  
--output text > iot-credential-provider.txt
```

5. 以前に作成した X.509 証明書に加えて、TLS を介してバックエンドサービスとの信頼を確立するための CA 証明書も必要です。CA 証明書は、次のコマンドを使用して取得できます。

```
curl --silent 'https://www.amazontrust.com/repository/SFSRootCAG2.pem' --output  
cacert.pem
```

ステップ 4: Kinesis ビデオストリームで AWS IoT 認証情報をテストする

これで、これまでに設定した AWS IoT 認証情報をテストできます。

1. まず、この設定のテストに使用する Kinesis ビデオストリームを作成します。

Important

前のステップ () で作成した AWS IoT モノの名前と同じ名前のビデオストリームを作成します kvs_example_camera_stream。

```
aws kinesisanalytics create-stream --data-retention-in-hours 24 --stream-name  
kvs_example_camera_stream
```

2. 次に、AWS IoT 認証情報プロバイダーを呼び出して一時的な認証情報を取得します。

```
curl --silent -H "x-amzn-iot-thingname:kvs_example_camera_stream" --cert
certificate.pem --key private.pem.key https://IOT_GET_CREDENTIAL_ENDPOINT/role-
aliases/KvsCameraIoTRoleAlias/credentials --cacert ./cacert.pem > token.json
```

Note

次のコマンドを使用して、`IOT_GET_CREDENTIAL_ENDPOINT`を取得できます。

```
IOT_GET_CREDENTIAL_ENDPOINT=`cat iot-credential-provider.txt`
```

出力 JSON には、`accessKey`、`secretKey`、および `sessionToken` が含まれており、Kinesis Video Streams へのアクセスに使用できます。

- テストでは、これらの認証情報を使用して、サンプル `kvs_example_camera_stream` ビデオストリームの Kinesis Video Streams `DescribeStream` API を呼び出すことができます。

```
AWS_ACCESS_KEY_ID=$(jq --raw-output '.credentials.accessKeyId' token.json)
AWS_SECRET_ACCESS_KEY=$(jq --raw-output '.credentials.secretAccessKey' token.json)
AWS_SESSION_TOKEN=$(jq --raw-output '.credentials.sessionToken' token.json) aws
kinesisvideo describe-stream --stream-name kvs_example_camera_stream
```

ステップ 5: カメラのファイルシステムに AWS IoT 証明書と認証情報をデプロイし、データをビデオストリームにストリーミングする

Note

このセクションのステップでは、`libcamera` を使用しているカメラから Kinesis ビデオストリームにメディアを送信する方法について説明します [the section called “C++”](#)。

- X.509 証明書、プライベートキー、および前のステップで生成された CA 証明書をカメラのファイルシステムにコピーします。これらのファイルが保存される場所のパス、ロールエイリアス名、コマンド `gst-launch-1.0` またはサンプルアプリケーションを実行するための AWS IoT 認証情報エンドポイントを指定します。

2. 次のサンプルコマンドは、AWS IoT 証明書認可を使用して Kinesis Video Streams に動画を送信します。

```
gst-launch-1.0 rtspsrc location=rtsp://YourCameraRtspUrl short-header=TRUE !
rtph264depay ! video/x-h264,format=avc,alignment=au ! h264parse ! kvssink stream-
name="kvs_example_camera_stream" aws-region="YourAWSRegion" iot-certificate="iot-
certificate,endpoint=credential-account-specific-prefix.credentials.iot.aws-
region.amazonaws.com,cert-path=/path/to/certificate.pem,key-path=/path/to/
private.pem.key,ca-path=/path/to/cacert.pem,role-aliases=KvsCameraIoTRoleAlias"
```

ストリーム名としてのAWS IoT CertificateId

AWS IoT モノを介してデバイス (カメラなど) を表すが、別のストリーム名を承認するには、属性を AWS IoT certificateId ストリーム名として使用し、を使用してストリームに対する Kinesis Video Streams アクセス許可を付与します AWS IoT。これを実現する手順は、前述の手順と似ていますが、いくつかの変更があります。

- アクセス許可ポリシーを IAM ロール (iam-permission-document.json) に次のように変更します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:DescribeStream",
        "kinesisvideo:PutMedia",
        "kinesisvideo:TagStream",
        "kinesisvideo:GetDataEndpoint"
      ],
      "Resource": "arn:aws:kinesisvideo:*:*:stream/${credentials-
iot:AwsCertificateId}/*"
    }
  ]
}
```

Note

リソース ARN では、ストリーム ID のプレースホルダーとして証明書 ID を使用します。IAM アクセス許可は、証明書 ID をストリーム名として使用すると機能します。証明書から証明書 ID を取得して、次のストリーム API コールでストリーム名として使用できます。

```
export CERTIFICATE_ID=`cat certificate | jq --raw-output '.certificateId'`
```

- Kinesis Video Streams の describe-stream CLI コマンドを使用して、この変更を確認します。

```
AWS_ACCESS_KEY_ID=$(jq --raw-output '.credentials.accessKeyId' token.json)
AWS_SECRET_ACCESS_KEY=$(jq --raw-output '.credentials.secretAccessKey' token.json)
AWS_SESSION_TOKEN=$(jq --raw-output '.credentials.sessionToken' token.json)
aws kinesisvideo describe-stream --stream-name ${CERTIFICATE_ID}
```

- `certificateId` を Kinesis Video Streams C++ SDK [のサンプルアプリケーションの](#) AWS IoT 認証情報プロバイダーに渡します。

```
credential_provider =
    make_unique<IotCertCredentialProvider>(iot_get_credential_endpoint,
        cert_path,
        private_key_path,
        role_alias,
        ca_cert_path,
        certificateId);
```

Note

を認証情報プロバイダー `thingname` に AWS IoT 渡すことに注意してください。を使用して `getenv`、他の AWS IoT 属性を渡すのと同様に、モノの名前をデモアプリケーションに渡すことができます。サンプルアプリケーションを実行するときに、コマンドラインパラメータでストリーム名として証明書 ID を使用します。

AWS IoT 認証情報を使用してハードコードされたストリーム名にストリーミングする

AWS IoT モノを介してデバイス (カメラなど) を表し、特定の Amazon Kinesis ビデオストリームへのストリーミングを許可するには、を使用してストリームに対する Amazon Kinesis Video Streams アクセス許可を付与します AWS IoT。このプロセスは前のセクションと似ていますが、いくつかの変更があります。

アクセス許可ポリシーを IAM ロール (iam-permission-document.json) に次のように変更します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:DescribeStream",
        "kinesisvideo:PutMedia",
        "kinesisvideo:TagStream",
        "kinesisvideo:GetDataEndpoint"
      ],
      "Resource": "arn:aws:kinesisvideo:*:*:stream/YourStreamName/*"
    }
  ]
}
```

前のステップで生成された X.509 証明書、プライベートキー、および CA 証明書をカメラのファイルシステムにコピーします。

これらのファイルが保存される場所のパス、ロールエイリアス名、AWS IoT モノの名前、コマンド `gst-launch-1.0` またはサンプルアプリケーションを実行するための AWS IoT 認証情報エンドポイントを指定します。

次のサンプルコマンドは、AWS IoT 証明書認可を使用して Amazon Kinesis Video Streams に動画を送信します。

```
gst-launch-1.0 rtspsrc location=rtsp://YourCameraRtspUrl short-header=TRUE !
rtph264depay ! video/x-h264,format=avc,alignment=au ! h264parse ! kvssink
stream-name="YourStreamName" aws-region="YourAWSRegion" iot-certificate="iot-
certificate,endpoint=credential-account-specific-prefix.credentials.iot.aws-
region.amazonaws.com,cert-path=/path/to/certificate.pem,key-path=/path/to/
private.pem,key-ca-path=/path/to/cacert.pem,role-aliases=KvsCameraIoTRoleAlias,iot-
thing-name=YourThingName"
```

Amazon Kinesis Video Streams のコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、[AWS のサービス「コンプライアンスプログラムによる対象範囲内」のコンプライアンス](#)を参照し、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、[AWS「コンプライアンスプログラム」](#)を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、[「Downloading Reports in AWS Artifact」](#)を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。を使用する際のコンプライアンス責任の詳細については AWS のサービス、[AWS「セキュリティドキュメント」](#)を参照してください。

Amazon Kinesis Video Streams の耐障害性

AWS グローバルインフラストラクチャは、AWS リージョンとアベイラビリティーゾーンを中心に構築されています。AWS リージョンは、低レイテンシー、高スループット、高度に冗長なネットワークで接続された、物理的に分離された複数のアベイラビリティーゾーンを提供します。アベイラビリティーゾーンでは、アベイラビリティーゾーン間で中断せずに、自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、およびスケーラビリティが優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、[AWS「グローバルインフラストラクチャ」](#)を参照してください。

Kinesis Video Streams のインフラストラクチャセキュリティ

マネージドサービスである Amazon Kinesis Video Streams は、ホワイトペーパー「[Amazon Web Services: セキュリティプロセスの概要](#)」に記載されている AWS グローバルネットワークセキュリティ手順で保護されています。

AWS 公開された API コールを使用して、ネットワーク経由で Kinesis Video Streams にアクセスします。クライアントは、Transport Layer Security (TLS) 1.2 以降をサポートする必要があります。また、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもクライアントでサポートされている必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストは、アクセスキー ID および、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、[AWS Security Token Service](#) (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

Kinesis Video Streams のセキュリティのベストプラクティス

Amazon Kinesis Video Streams には、独自のセキュリティポリシーを策定および実装する際に考慮すべきさまざまなセキュリティ機能が用意されています。以下のベストプラクティスは一般的なガイドラインであり、完全なセキュリティソリューションを説明するものではありません。これらのベストプラクティスはお客様の環境に適切ではないか、十分ではない場合があるため、これらは指示ではなく、有用な考慮事項と見なしてください。

お客様のリモートデバイスのセキュリティベストプラクティスについては、[デバイスエージェントのセキュリティベストプラクティス](#)を参照してください。

最小特権アクセスの実装

アクセス許可を付与する場合、どのユーザーにどの Kinesis Video Streams リソースに対するアクセス許可を付与するかは、お客様が決定します。これらのリソースで許可したい特定のアクションを有効にするのも、お客様になります。このため、タスクの実行に必要なアクセス許可のみを付与する必要があります。最小特権アクセスの実装は、セキュリティリスクと、エラーや悪意によってもたらされる可能性のある影響の低減における基本になります。

例えば、Kinesis Video Streams にデータを送るプロデューサーに必要なのは、PutMedia、GetStreamingEndpoint、および DescribeStream のみです。このため、すべ

てのアクション (*) や GetMedia などの他のアクションに必要なアクセス権限を、プロデューサーアプリケーションに付与しないでください。

詳細については、[What Is Least Privilege & Why Do You Need It?](#)を参照してください。

IAM ロールの使用

Kinesis Video Streams にアクセスするには、プロデューサーアプリケーションとクライアントアプリケーションに有効な認証情報が必要です。AWS 認証情報は、クライアントアプリケーションや Amazon S3 バケットに直接保存しないでください。これらは、自動的にローテーションされない長期的な認証情報であり、侵害された場合にビジネスに大きな影響を与える可能性があります。

代わりに、IAM ロールを使用して、プロデューサーおよびクライアントアプリケーションが Kinesis Video Streams にアクセスするための一時的な認証情報を管理する必要があります。ロールを使用する場合、他のリソースにアクセスするために長期的な認証情報 (ユーザー名とパスワードやアクセスキーなど) を使用する必要はありません。

詳細については、IAM ユーザーガイドにある下記のトピックを参照してください。

- [IAM ロール](#)
- [ロールの一般的なシナリオ: ユーザー、アプリケーション、およびサービス](#)

CloudTrail を使用した API コールのモニタリング

Kinesis Video Streams は AWS CloudTrail、Kinesis Video Streams AWS のサービスのユーザー、ロール、または によって実行されたアクションを記録するサービスであると連携します。

CloudTrail で収集された情報を使用して、Kinesis Video Streams に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

詳細については、「[the section called “CloudTrail を使用した API コールのログ記録”](#)」を参照してください。

Kinesis Video Streams のトラブルシューティング

次の情報を使用して、Amazon Kinesis Video Streams で発生する一般的な問題をトラブルシューティングします。

トピック

- [一般的な問題](#)
- [API の問題](#)
- [HLS の問題](#)
- [Java の問題](#)
- [プロデューサーライブラリの問題](#)
- [ストリームパーサーライブラリの問題](#)
- [ネットワークの問題](#)

一般的な問題

このセクションでは、Kinesis Video Streams を操作するときに発生する可能性がある一般的な問題について説明します。

問題

- [レイテンシーが高すぎる](#)

レイテンシーが高すぎる

レイテンシーは、Kinesis Video Streams サービスに送信されるフラグメント継続時間が原因で発生する場合があります。プロデューサーとサービスの間のレイテンシーを減らす方法の 1 つとして、より短いフラグメント継続時間を作成するようにメディアパイプラインを設定します。

各フラグメントで送信されるフレーム数を減らすには、で次の値を減らし、`kinesis_video_gstreamer_sample_app.cpp`。

```
g_object_set(G_OBJECT (data.encoder), "bframes", 0, "key-int-max", 45, "bitrate", 512, NULL);
```

Note

Mozilla Firefox ブラウザではビデオレンダリングを内部実装しているため、レイテンシーが高くなります。

API の問題

このセクションでは、Kinesis Video Streams を操作するときが発生する可能性がある API の問題について説明します。

問題

- [エラー: 「未知のオプション」](#)
- [エラー: "承認するサービス/オペレーション名を特定できませんでした"](#)
- [エラー: "ストリームにフレームを配置できませんでした"](#)
- [エラー: "最終 AckEvent が受信される前に、サービスによって接続が終了されました"](#)
- [エラー: 「STATUS_STORE_OUT_OF_MEMORY」](#)
- [エラー: 「認証情報は有効なリージョンにスコープする必要があります」](#)

エラー: 「未知のオプション」

GetMedia と GetMediaForFragmentList は、次のエラーで失敗する場合があります。

```
Unknown options: <filename>.mkv
```

このエラーは、output のタイプを AWS CLI に設定した場合に発生します json。をデフォルトの出カタイプ () AWS CLI で再設定します none。の設定の詳細については AWS CLI、AWS CLI 「コマンドリファレンス」の [「設定」](#) を参照してください。

エラー: "承認するサービス/オペレーション名を特定できませんでした"

GetMedia は、次のエラーで失敗する場合があります。

```
Unable to determine service/operation name to be authorized
```


このエラーが発生する可能性があるのは、エンドポイントが適切に指定されていない場合です。エンドポイントを取得するときは、GetDataEndpoint呼び出す API に応じて、必ず次のパラメータを呼び出しに含めてください。

```
--api-name GET_MEDIA
--api-name PUT_MEDIA
--api-name GET_MEDIA_FOR_FRAGMENT_LIST
--api-name LIST_FRAGMENTS
```

エラー: "ストリームにフレームを配置できませんでした"

PutMedia は、次のエラーで失敗する場合があります。

```
Failed to put a frame in the stream
```

このエラーが発生する可能性があるのは、サービスで接続またはアクセス許可が利用できない場合です。で以下を実行し AWS CLI、ストリーム情報を取得できることを確認します。

```
aws kinesisanalytics describe-stream --stream-name StreamName --endpoint https://ServiceEndpoint.kinesisvideo.region.amazonaws.com
```

呼び出しが失敗した場合の詳細については、[AWS CLI 「エラーのトラブルシューティング」](#)を参照してください。

エラー: "最終 AckEvent が受信される前に、サービスによって接続が終了されました"

PutMedia は、次のエラーで失敗する場合があります。

```
com.amazonaws.SdkClientException: Service closed connection before final AckEvent was received
```

このエラーが発生する可能性があるのは、PushbackInputStream が不適切に実装されている場合です。unread() メソッドが正しく実装されていることを確認します。

エラー: 「STATUS_STORE_OUT_OF_MEMORY」

PutMedia は、次のエラーで失敗する場合があります。

The content store is out of memory.

コンテンツストアに十分なサイズが割り当てられていない場合、このエラーが発生します。コンテンツストアのサイズを増やすには、`StorageInfo.storageSize` の値を増やします。詳細については、「[StorageInfo](#)」を参照してください。

エラー：「認証情報は有効なリージョンにスコープする必要があります」

このエラーは、署名リージョンがエンドポイントリージョンと一致しない場合に発生します。

たとえば、署名リージョン `us-west-2` として を指定し、`kinesisvideo.us-east-1.amazonaws.com` (`us-east-1`) エンドポイントに接続しようとする、このエラーが表示されます。

[kvssink](#) などの一部のアプリケーションでは、リージョンフォールバックチェーンのデフォルトは `us-west-2` です。使用しているアプリケーションに従ってリージョンが正しく設定されていることを確認します。

HLS の問題

ビデオストリームが正しく再生されない場合は、「」を参照してください [the section called “HLS の問題のトラブルシューティング”](#)。

Java の問題

このセクションでは、Kinesis Video Streams を操作するときに発生する一般的な Java の問題のトラブルシューティング方法について説明します。

問題

- [Java ログの有効化](#)

Java ログの有効化

Java サンプルとライブラリに関する問題をトラブルシューティングするには、デバッグログを有効にして調べることをお勧めします。デバッグログを有効にするには、次の操作を行います。

1. `log4j` を `pom.xml` ノードの `dependencies` ファイルに追加します。

```
<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.17</version>
</dependency>
```

2. target/classes ディレクトリで、次の内容で log4j.properties というファイルを作成します。

```
# Root logger option
log4j.rootLogger=DEBUG, stdout

# Redirect log messages to console
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:
%L - %m%n

log4j.logger.org.apache.http.wire=DEBUG
```

デバッグログが、IDE コンソールに出力されます。

プロデューサーライブラリの問題

このセクションでは、[Kinesis Video Streams へのアップロード](#) を使用するときが発生する可能性がある問題について説明します。

問題

- [プロデューサー SDK をコンパイルできない](#)
- [ビデオストリームはコンソールには表示されません。](#)
- [エラー: GStreamer デモアプリケーションを使用したデータのストリーミング時の "リクエストに含まれているセキュリティトークンが無効です"](#)
- [エラー: 「Kinesis Video クライアントにフレームを送信できませんでした」](#)
- [GStreamer アプリケーションが、OS X で "ストリーミングが中止されました。ネゴシエーションされていないという理由です" というメッセージで停止する](#)

- [エラー: Raspberry Pi の GStreamer デモで Kinesis ビデオクライアントを作成するときの "ヒープを割り当てできませんでした"](#)
- [エラー: Raspberry Pi での GStreamer デモの実行時の "無効な命令"](#)
- [カメラで Raspberry Pi のロードに失敗する](#)
- [カメラが macOS High Sierra で見つからない](#)
- [macOS High Sierra でコンパイルするときに、jni.h ファイルが見つかりません](#)
- [GStreamer デモアプリケーションを実行中の Curl エラー](#)
- [Raspberry Pi での実行時のタイムスタンプ/範囲アサーション](#)
- [Raspberry Pi の gst_value_set_fraction_range_full でのアサーション](#)
- [Android での STATUS_MKV_INVALID_ANNEXB_NALU_IN_FRAME_DATA\(0x3200000d\) エラー](#)
- [最大フラグメント期間に達したエラー](#)
- [IoT 認可の使用中に "無効なモノの名前が渡されました \(Invalid thing name passed\)" エラーが発生](#)

プロデューサー SDK をコンパイルできない

パスに必要なライブラリがあることを確認します。これを確認するには、次のコマンドを使用します。

```
env | grep LD_LIBRARY_PATH
LD_LIBRARY_PATH=/home/local/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp/
kinesis-video-native-build/downloads/local/lib
```

ビデオストリームはコンソールには表示されません。

コンソールでビデオストリームを表示するには、H.264 を使用して AvCC 形式でエンコードする必要があります。ストリームが表示されない場合は、以下の点を確認してください。

- 元のストリームが Annex-B 形式である場合、[NAL 適応フラグ](#) が NAL_ADAPTATION_ANNEXB_NALS | NAL_ADAPTATION_ANNEXB_CPD_NALS に設定されている。これは StreamDefinition コンストラクタのデフォルト値です。
- コーデックのプライベートデータを正しく提供しています。H.264 では、これはシーケンスパラメータセット (SPS) とピクチャパラメータセット (PPS) です。メディアソースに応じて、このデータはメディアソースから個別に取得されるか、フレームにエンコードされています。

多くの基本ストリームの形式は次のとおりです。ここで、Ab は Annex-B の開始コード (001 または 0001) です。

```
Ab(Sps)Ab(Pps)Ab(I-frame)Ab(P/B-frame) Ab(P/B-frame)... Ab(Sps)Ab(Pps)Ab(I-frame)Ab(P/B-frame) Ab(P/B-frame)
```

H.264 が SPS および PPS としてストリームにある場合、CPD (コーデックプライベートデータ) は AvCC 形式に適応できます。メディアパイプラインが CPD を個別に指定しない限り、アプリケーションは最初の Idr フレーム (SPS と PPS を含む必要があります) を検索してフレームから CPD を抽出し、2 つの NALUs (Ab(Sps)Ab(Pps) になります) を抽出し、 の CPD に設定できます StreamDefinition。

エラー: GStreamer デモアプリケーションを使用したデータのストリーミング時の "リクエストに含まれているセキュリティトークンが無効です"

このエラーが発生した場合は、認証情報に問題があります。以下について確認します。

- 一時的なセキュリティ認証情報を使用している場合は、セッショントークンを指定する必要があります。
- 一時的な認証情報が失効していないことを確認します。
- 適切な権限が設定されていることを確認します。
- macOS では、Keychain にキャッシュされた認証情報がないことを確認します。

エラー: 「Kinesis Video クライアントにフレームを送信できませんでした」

このエラーが発生した場合、タイムスタンプはソースストリームに正しく設定されません。次の操作を試してください：

- この問題を解決する最新の SDK サンプルを使用してください。
- 高品質のストリームをより高いビットレートに設定し、カメラがサポートしている場合はソースストリームのジッターを修正します。

GStreamer アプリケーションが、OS X で "ストリーミングが中止されました。ネゴシエーションされていないという理由です" というメッセージで停止する

OS X では、ストリーミングが停止し、次のメッセージが表示される場合があります。

```
Debugging information: gstbasesrc.c(2939): void gst_base_src_loop(GstPad *) (): /
GstPipeline:test-pipeline/GstAutoVideoSrc:source/GstAVFVideoSrc:source-actual-src-
avfvide:
streaming stopped, reason not-negotiated (-4)
```

これを回避するには、の `gst_caps_new_simple` 呼び出しからフレームレートパラメータを削除します `kinesis_video_gstreamer_sample_app.cpp`。

```
GstCaps *h264_caps = gst_caps_new_simple("video/x-h264",
                                          "profile", G_TYPE_STRING, "baseline",
                                          "stream-format", G_TYPE_STRING, "avc",
                                          "alignment", G_TYPE_STRING, "au",
                                          "width", GST_TYPE_INT_RANGE, 320, 1920,
                                          "height", GST_TYPE_INT_RANGE, 240, 1080,
                                          "framerate", GST_TYPE_FRACTION_RANGE, 0,
                                          1, 30, 1,
                                          NULL);
```

エラー: Raspberry Pi の GStreamer デモで Kinesis ビデオクライアントを作成するときの "ヒープを割り当てできませんでした"

GStreamer サンプルアプリケーションは、512 MB の RAM を割り当てようとしてますが、これがシステムで使用できない場合があります。 `KinesisVideoProducer.cpp` で次の値を減らすことによって、この割り当てを減らすことができます。

```
device_info.storageInfo.storageSize = 512 * 1024 * 1024;
```

エラー: Raspberry Pi での GStreamer デモの実行時の "無効な命令"

GStreamer デモの実行中に次のエラーが発生した場合は、デバイスの正しいバージョン用にアプリケーションをコンパイルしたことを確認してください。(例えば、Raspberry Pi 2 で実行しているときに、Raspberry Pi 3 用にコンパイルしていないことを確認します)。

```
INFO - Initializing curl.  
Illegal instruction
```

カメラで Raspberry Pi のロードに失敗する

カメラがロード済みかどうか確認するには、次のコマンドを実行します。

```
ls /dev/video*
```

何も見つからない場合は、次のコマンドを実行します。

```
vcgencmd get_camera
```

出力は次の例に類似したものになります：

```
supported=1 detected=1
```

ドライバでカメラが検出されない場合は、次のコマンドを実行します。

1. 物理的なカメラの設定を確認し、適切に接続されていることを確認します。
2. 以下を実行してファームウェアをアップグレードします。

```
sudo rpi-update
```

3. デバイスを再起動します。
4. 以下を実行してドライバをロードします。

```
sudo modprobe bcm2835-v4l2
```

5. カメラが検出されたことを確認します。

```
ls /dev/video*
```

カメラが macOS High Sierra で見つからない

macOS High Sierra で複数のカメラが利用できる場合、デモアプリケーションはカメラを見つけることができません。

macOS High Sierra でコンパイルするときに、jni.h ファイルが見つかりません

このエラーを解決するには、Xcode のインストールを最新バージョンに更新してください。

GStreamer デモアプリケーションを実行中の Curl エラー

エラーを解決するには、GStreamer デモアプリケーションを実行するとき、[この証明書ファイル](#)を /etc/ssl/cert.pem にコピーします。

Raspberry Pi での実行時のタイムスタンプ/範囲アサーション

実行時にタイムスタンプの範囲アサーションが発生した場合は、ファームウェアを更新し、デバイスを再起動します。

```
sudo rpi-update
$ sudo reboot
```

Raspberry Pi の gst_value_set_fraction_range_full でのアサーション

uv4l サービスが実行中の場合は、次のアサーションが表示されます。

```
gst_util_fraction_compare (numerator_start, denominator_start, numerator_end,
denominator_end) < 0' failed
```

これが発生した場合は、uv4l サービスを停止し、アプリケーションを再起動します。

Android での

STATUS_MKV_INVALID_ANNEXB_NALU_IN_FRAME_DATA(0x3200000d) エラー

メディア・ストリームの [NAL 適応フラグ](#) が間違っている場合、次のエラーが表示されます。

```
putKinesisVideoFrame(): Failed to put a frame with status code 0x3200000d
```

このエラーが発生した場合は、メディアの .withNalAdaptationFlags フラグを正しく入力します (例 : NAL_ADAPTATION_ANNEXB_CPD_NALS)。このフラグを [Android](#) の次の行に入力します。

<https://github.com/aws-labs/aws-sdk-android-samples/blob/master/AmazonKinesisVideoDemoApp/src/main/java/com/amazonaws/kinesisvideo/demoapp/fragment/StreamConfigurationFragment.java#L169>

最大フラグメント期間に達したエラー

このエラーは、ストリーム内のメディアフラグメントが最大フラグメント継続期間の制限を超えると発生します。[the section called “メディアおよびアーカイブメディア API サービスクォータ”](#) セクションのフラグメントの最大期間制限を参照してください。

この問題を解決するには、以下の手順を実行します。

- ウェブカメラ/USB カメラを使用している場合は、次のいずれかの操作を行います。
 - キーフレームベースのフラグメント化を使用している場合は、10 秒以内にキーフレームを提供するようにエンコーダーを設定します。
 - キーフレームベースのフラグメント化を使用していない場合は、でストリームを定義するときに[コードを記述して調べる](#)、最大フラグメント期間制限を 10 秒未満の値に設定します。
 - GStreamer パイプラインでソフトウェアエンコーダー (x264 など) を使用している場合は、key-int-max 属性を 10 秒以内に値に設定できます。たとえば、key-int-max を 60 に設定し、fps を 30 に設定して、2 秒ごとにキーフレームを有効にします。
- RPI カメラを使用している場合は、keyframe-interval 属性を 10 秒未満に設定します。
- IP (RTSP) カメラを使用している場合は、GOP サイズを 60 に設定します。

IoT 認可の使用中に "無効なモノの名前が渡されました (Invalid thing name passed)" エラーが発生

認可に IoT 認証情報を使用している場合にこのエラー (HTTP Error 403: Response: {"message": "Invalid thing name passed"}) を回避するには、stream-name (kvssink要素の必須パラメータ) の値が の値と同じであることを確認しますiot-thingname。詳細については、「[GStreamer 要素パラメータリファレンス](#)」を参照してください。

ストリームパーサーライブラリの問題

このセクションでは、[パーサーライブラリを使用したストリーミング](#) を使用するときが発生する可能性がある問題について説明します。

問題

- [ストリームから 1 つのフレームにアクセスできない](#)
- [フラグメントのデコードエラー](#)

ストリームから 1 つのフレームにアクセスできない

コンシューマーアプリケーションのストリーミングソースから 1 つのフレームにアクセスするには、ストリームに正しいコーデックプライベートデータが含まれていることを確認します。ストリームのデータの形式の詳細については、「[データモデル](#)」を参照してください。

コーデックプライベートデータを使用してフレームにアクセスする方法については、GitHub ウェブサイトで [KinesisVideoRendererExampleTest.java](#) テストファイルを参照してください。

フラグメントのデコードエラー

フラグメントが H.264 フォーマットで適切にエンコードされておらず、ブラウザがサポートしているレベルである場合、コンソールでストリームを再生するときに次のエラーが表示されることがあります。

Fragment Decoding Error

There was an error decoding the video data. Verify that the stream contains valid H.264 content

このような場合は、次の点を確認してください。

- フレームの解像度が、コーデックのプライベートデータで指定された解像度に一致している。
- エンコードされたフレームの H.264 プロファイルとレベルが、コーデックのプライベートデータで指定されたプロファイルとレベルに一致している。
- ブラウザがプロファイル/レベルの組み合わせをサポートしている。現在のほとんどのブラウザは、すべてのプロファイルとレベルの組み合わせをサポートしています。
- タイムスタンプは正確で正しい順序であり、重複するタイムスタンプは作成されない。
- お使いのアプリケーションが H.264 形式を使用してフレームデータをエンコードしている。

ネットワークの問題

Kinesis Video Streams に接続しようとする、「接続タイムアウト」や「接続失敗」などの接続エラーが表示される場合は、ネットワーク設定の IP アドレス範囲の制限が原因である可能性があります。

設定に Kinesis Video Streams の IP アドレス範囲の制限がある場合は、ネットワーク設定を更新して Kinesis Video Streams の [IP アドレス範囲](#)を許可リストに登録します。

 Important

IP 範囲リストは、Kinesis Video Streams の IP アドレスの完全なリストではありません。表示される IP アドレス範囲を含め、IP アドレスは時間の経過とともに変化する可能性があることに注意してください。

詳細については、[AWS 「IP 範囲」](#)を参照してください。IP 範囲が変更されたときに通知を受け取るには、[サブスクリプションの手順](#)に従います。

Amazon Kinesis Video Streams のドキュメント履歴

次の表に、Amazon Kinesis Video Streams の前回のリリース以後に行われたドキュメントの重要な変更を示します。

- 最新の API バージョン: 2017 年 11 月 29 日
- ドキュメントの最終更新日: 2025 年 1 月 6 日

変更	説明	日付
Raspberry Pi の C++	Raspberry Pi で C++ プロデューサー SDK を使用するためのドキュメントを更新しました。	2025 年 1 月 6 日
Amazon Kinesis Video Streams Edge Agent Edge-to-Cloud 接続	新機能のリリース。詳細については、「 ビデオ録画とストレージをスケジュールする 」を参照してください。	2023 年 6 月 27 日
開始方法: Kinesis ビデオストリームにデータを送信する	カメラから Kinesis ビデオストリームにメディアデータを送信するための基本チュートリアル。詳細については、「 Amazon Kinesis ビデオストリームにデータを送信する 」を参照してください。	2019 年 1 月 21 日
ストリーミングメタデータ	プロデューサー SDK を使用して、Kinesis ビデオストリームにメタデータを埋め込むことができます。詳細については、「 Kinesis Video Streams でのストリーミングメタデータの使用 」を参照してください。	2018 年 9 月 28 日

変更	説明	日付
C++ プロデューサー SDK ログ記録	C++ プロデューサー SDK アプリケーションのログ記録を設定できます。詳細については、「 C++ プロデューサー SDK でログ記録を使用する 」を参照してください。	2018 年 7 月 18 日
HLS 動画ストリーミング	HTTP ライブストリーミングを使用して、Kinesis ビデオストリームを表示できるようになりました。詳細については、「 Kinesis Video Streams の再生 」を参照してください。	2018 年 7 月 13 日
RTSP ソースからのストリーミング	Docker コンテナ内で実行され RTSP ソースからビデオをストリーミングする Kinesis Video Streams 用のサンプルアプリケーション。詳細については、「 RTSP および Docker 」を参照してください。	2018 年 6 月 20 日
C++ プロデューサー SDK GStreamer プラグイン	C++ を GStreamer 送信先として使用する構築方法を示します。詳細については、「 GStreamer プラグイン - kvssink 」を参照してください。	2018 年 6 月 15 日

変更	説明	日付
プロデューサー SDK コールバックのリファレンスドキュメント	Kinesis Video Streams へのアップロード で使用されるコールバックのリファレンスドキュメント。詳細については、「 プロデューサー SDK コールバック 」を参照してください。	2018 年 12 月 6 日
システム要件	プロデューサーデバイスと SDK のメモリ要件およびストレージ要件のドキュメントです。詳細については、「 Amazon Kinesis Video Streams システム要件 」を参照してください。	2018 年 5 月 30 日
CloudTrail のサポート	CloudTrail を使用して API の使用状況を監視するためのドキュメントです。詳細については、「 を使用した Amazon Kinesis Video Streams API コールのログ記録 AWS CloudTrail 」を参照してください。	2018 年 5 月 24 日
プロデューサー SDK 構造のリファレンスドキュメント	Kinesis Video Streams へのアップロード 構造で使用される構造のリファレンスドキュメント。詳細については、「 プロデューサー SDK 構造 」および「 Kinesis ビデオストリーム構造 」を参照してください。	2018 年 5 月 7 日

変更	説明	日付
レンダラーの例に関するドキュメント	レンダラーのサンプルアプリケーションのドキュメントです。Kinesis ビデオストリームからフレームをデコードして表示する方法を示しています。詳細については、「 例: Kinesis Video Streams フラグメントの解析とレンダリング 」を参照してください。	2018 年 3 月 15 日
プロデューサー SDK 制限の参照ドキュメント	オペレーションの制限についての情報は、「 C++ 」を参照してください。詳細については、「 プロデューサー SDK クォータ 」を参照してください。	2018 年 3 月 13 日
モニタリング	Amazon CloudWatch および AWS CloudTrailを使用した Kinesis Video Streams メトリクスと API コールのモニタリングに関する情報。詳細については、「 Amazon Kinesis Video Streams のモニタリング 」を参照してください。	2018 年 2 月 5 日
Network Abstraction Layer (NAL) 適応フラグを参照	長時間のストリーミングビデオに NAL 適応フラグを設定するための情報。詳細については、「 NAL 適応フラグ 」を参照してください。	2018 年 1 月 15 日

変更	説明	日付
ストリーミングビデオの Android サポート	Kinesis Video Streams で、Android デバイスからのビデオのストリーミングがサポートされるようになりました。詳細については、「 Android 」を参照してください。	2018 年 1 月 12 日
Kinesis ビデオのサンプルドキュメント	Kinesis ビデオのサンプルアプリケーションのドキュメントです。アプリケーションで パーサーライブラリを使用してカメラからの出力を監視する を使用する方法を示しています。詳細については、「 KinesisVideoExample 」を参照してください。	2018 年 1 月 9 日
Kinesis Video Streams のドキュメントをリリース	これは Amazon Kinesis Video Streams 開発者ガイドの最初の一般リリースです。	2017 年 11 月 29 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。